参考
Trident

NetApp
January 15, 2026

目录

参考

Trident港口

了解更多关于Trident用于通信的端口的信息。

Trident港口

Trident使用以下端口在 Kubernetes 内部进行通信：

端口	目的
8443	反向通道 HTTPS
8001	Prometheus 指标端点
8000	Trident REST 服务器
17546	Trident守护进程集 pod 使用的存活/就绪探测端口

安装过程中可以使用以下方法更改活性/就绪探针端口： `--probe-port`旗帜。务必确保工作节点上的其他进程没有使用该端口。

Trident REST API

尽管"tridentctl 命令和选项"这是与Trident REST API 交互的最简单方法，如果您愿意，也可以直接使用 REST 端点。

何时使用 REST API

REST API 适用于在非 Kubernetes 部署中使用Trident作为独立二进制文件的高级安装。

为了更好的安全性，Trident `REST API`在 pod 内运行时，默认仅限于 localhost。要改变这种行为，你需要设置 Trident 的 `-address`在其 pod 配置中设置参数。

使用 REST API

要查看这些 API 的调用示例，请传递调试信息。(`-d`） 旗帜。更多信息，请参阅"使用 tridentctl 管理Trident"。

API 的工作原理如下：

GET

`GET <trident-address>/trident/v1/<object-type>`

列出该类型的所有对象。

`GET <trident-address>/trident/v1/<object-type>/<object-name>`

获取指定对象的详细信息。

POST

`POST <trident-address>/trident/v1/<object-type>`

创建指定类型的对象。

- 创建对象需要 JSON 配置。有关每种对象类型的详细说明，请参阅："使用 tridentctl 管理Trident" 。
- 如果对象已存在，则行为会有所不同：后端会更新现有对象，而所有其他对象类型都会使操作失败。

DELETE

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`

删除指定的资源。

> 与后端或存储类关联的卷将继续存在；这些卷必须单独删除。更多信息，请参阅"使用 tridentctl 管理Trident"。

命令行选项

Trident为Trident编排器提供了几个命令行选项。您可以使用这些选项来修改您的部署。

日志记录

`-debug`

启用调试输出。

`-loglevel <level>`

设置日志级别（debug、info、warn、error、fatal）。默认显示信息。

Kubernetes

`-k8s_pod`

使用此选项或 `-k8s_api_server`启用 Kubernetes 支持。设置此项后， Trident将使用其所在 pod 的 Kubernetes 服务帐户凭据来联系 API 服务器。只有当Trident作为 Pod 在启用了服务帐户的 Kubernetes 集群中运行时，此方法才有效。

`-k8s_api_server <insecure-address:insecure-port>`

使用此选项或 `-k8s_pod`启用 Kubernetes 支持。指定后， Trident将使用提供的不安全地址和端口连接到 Kubernetes API 服务器。这使得Trident可以部署在 pod 之外；但是，它仅支持与 API 服务器的不安全连接。为了安全连接，请将Trident部署在带有以下组件的 pod 中： `-k8s_pod`选项。

Docker

`-volume_driver <name>`

注册 Docker 插件时使用的驱动程序名称。默认为 `netapp`。

`-driver_port <port-number>`

监听此端口，而不是 UNIX 域套接字。

-config <file>

必填项；您必须指定后端配置文件的路径。

休息

-address <ip-or-host>

指定 Trident REST 服务器应该监听的地址。默认为本地主机。当监听本地主机并在 Kubernetes pod 内运行时，REST 接口无法从 pod 外部直接访问。使用 `-address ""`使 REST 接口能够通过 pod IP 地址访问。

> Trident REST 接口可以配置为仅监听和提供服务于 127.0.0.1（对于 IPv4）或 [::1]（对于 IPv6）。

-port <port-number>

指定 Trident REST 服务器应监听的端口。默认值为 8000。

-rest

启用 REST 接口。默认为真。

Kubernetes 和Trident对象

您可以使用 REST API 通过读取和写入资源对象与 Kubernetes 和Trident进行交互。有几个资源对象决定了 Kubernetes 与Trident、 Trident与存储以及 Kubernetes 与存储之间的关系。其中一些对象通过 Kubernetes 进行管理，另一些对象通过Trident进行管理。

这些物体之间是如何相互作用的?

要了解这些对象、它们的用途以及它们如何交互，最简单的方法或许是跟踪 Kubernetes 用户发出的单个存储请求：

1. 用户创建 `PersistentVolumeClaim`请求一个新的 `PersistentVolume`来自 Kubernetes 的特定大小 `StorageClass`这是管理员之前配置好的。
2. Kubernetes `StorageClass`将其配置器标识为Trident ，并包含告诉Trident如何为请求的类配置卷的参数。
3. Trident审视自身 `StorageClass`名称相同，用于标识匹配项 `Backends`和 `StoragePools`它可以用来为该类配置卷。
4. Trident在匹配的后端配置存储并创建两个对象：a `PersistentVolume`在 Kubernetes 中，它告诉 Kubernetes 如何查找、挂载和处理卷；在Trident中，它维护着两者之间的关系。 `PersistentVolume`以及实际存储。
5. Kubernetes 绑定了 `PersistentVolumeClaim`新的 `PersistentVolume`。包含以下部件的舱体 `PersistentVolumeClaim`将该持久卷挂载到它运行的任何主机上。
6. 用户创建 `VolumeSnapshot`利用现有的PVC管材， `VolumeSnapshotClass`这表明Trident存在问题。
7. Trident识别与 PVC 关联的卷，并在其后端创建该卷的快照。它还创造了一个 `VolumeSnapshotContent`指示 Kubernetes 如何识别快照。
8. 用户可以创建 `PersistentVolumeClaim`使用 `VolumeSnapshot`作为来源。
9. Trident会识别所需的快照，并执行与创建快照相同的步骤。 `PersistentVolume`和 `Volume`。

如需进一步了解 Kubernetes 对象，我们强烈建议您阅读以下内容： "持久卷" Kubernetes 文档的这一部分。

Kubernetes `PersistentVolumeClaim`对象

Kubernetes `PersistentVolumeClaim` object 是 Kubernetes 集群用户发出的存储请求。

除了标准规范之外， Trident还允许用户指定以下卷特定的注释，以便覆盖您在后端配置中设置的默认值：

标注	卷选项	支持的驱动程序
trident.netapp.io/fileSystem	文件系统	ontap-san、solidfire-san、ontap-san-economy
trident.netapp.io/cloneFromPVC	克隆源卷	ontap-nas、ontap-san、solidfire-san、azure-netapp-files、gcp-cvs、ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas，ontap-san
trident.netapp.io/protocol	protocol	任何
trident.netapp.io/exportPolicy	出口政策	ontap-nas、ontap-nas-economy 、ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	快照策略	ontap-nas、ontap-nas-economy 、ontap-nas-flexgroup、ontap-san
trident.netapp.io/snapshotReserve	快照储备	ontap-nas、ontap-nas-flexgroup 、ontap-san、gcp-cvs
trident.netapp.io/snapshotDirectory	快照目录	ontap-nas、ontap-nas-economy 、ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unix权限	ontap-nas、ontap-nas-economy 、ontap-nas-flexgroup
trident.netapp.io/blockSize	块大小	solidfire-san

如果创建的PV具有 `Delete`根据回收策略，当 PV 被释放时（即用户删除 PVC 时）， Trident会同时删除 PV 和支持卷。如果删除操作失败， Trident会将 PV 标记为失败，并定期重试该操作，直到操作成功或手动删除 PV 为止。如果光伏发电使用 `Retain`策略方面， Trident会忽略它，并假定管理员会从 Kubernetes 和后端清理它，从而允许在删除卷之前对其进行备份或检查。请注意，删除 PV 不会导致Trident删除备份卷。您应该使用 REST API 将其删除。（`tridentctl`）。

Trident支持使用 CSI 规范创建卷快照：您可以创建卷快照并将其用作数据源来克隆现有的 PVC。这样，就可以将 PV 的特定时间点副本以快照的形式暴露给 Kubernetes。然后可以使用这些快照创建新的PV。看看 `On-Demand Volume Snapshots`看看这种方法是否可行。

Trident还提供 `cloneFromPVC`和 `splitOnClone`用于创建克隆的注释。您可以使用这些注解来克隆 PVC，而无需使用 CSI 实现。

例如：如果用户已经有一个名为 PVC 的 `mysql`用户可以创建一个名为"新PVC"的 `mysqlclone`通过使用注释，例如 `trident.netapp.io/cloneFromPVC: mysql`。有了这组注释， Trident会克隆与 mysql PVC 对应的卷，而不是从头开始配置卷。

请考虑以下几点：

- NetApp建议克隆空闲卷。
- PVC 及其克隆应该位于同一个 Kubernetes 命名空间中，并且具有相同的存储类。
- 随着 `ontap-nas`和 `ontap-san`对于驱动程序来说，设置 `PVC` 注释可能是有益的。`trident.netapp.io/splitOnClone`与 `trident.netapp.io/cloneFromPVC`。和 `trident.netapp.io/splitOnClone`设置为 `true`Trident将克隆卷与父卷分离，从而将克隆卷的生命周期与其父卷完全解耦，但代价是损失了一些存储效率。未设置 `trident.netapp.io/splitOnClone`或者将其设置为 `false`这样做可以减少后端空间占用，但代价是在父卷和克隆卷之间创建依赖关系，使得除非先删除克隆卷，否则无法删除父卷。在克隆空数据库卷时，拆分克隆是有意义的，因为预计该卷及其克隆卷将有很大的不同，并且无法从ONTAP提供的存储效率中受益。

这 `sample-input`目录包含可用于Trident的 PVC 定义示例。请参阅有关Trident音量相关参数和设置的完整说明。

Kubernetes `PersistentVolume`对象

Kubernetes `PersistentVolume`该对象代表一块可供 Kubernetes 集群使用的存储空间。它的生命周期与使用它的舱体无关。

> Trident创造 `PersistentVolume`根据其配置的卷，自动将对象注册到 Kubernetes 集群。您无需自行管理它们。

当您创建 PVC 时，指的是基于 Trident 的 `StorageClass`Trident使用相应的存储类配置一个新卷，并为该卷注册一个新的 PV。在配置已配置卷和相应的 PV 时， Trident遵循以下规则：

- Trident会为 Kubernetes 生成一个 PV 名称，以及一个用于配置存储的内部名称。无论哪种情况，名称在其范围内都是独一无二的，这一点都令人放心。
- 容量大小与 PVC 中要求的容量大小尽可能接近，但可能会向上取整到最接近的可分配数量，具体取决于平台。

Kubernetes `StorageClass`对象

Kubernetes `StorageClass`对象通过名称指定。 `PersistentVolumeClaims`为存储配置一组属性。存储类本身标识要使用的配置器，并以配置器能够理解的方式定义该组属性。

它是管理员需要创建和管理的两个基本对象之一。另一个是Trident后端对象。

Kubernetes `StorageClass`使用Trident的对象看起来像这样：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters: <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

这些参数是 Trident 特有的，用于告诉Trident如何为该类配置卷。

存储类参数如下：

属性	类型	必填项	描述
属性	映射[字符串]字符串	不可以	请参阅以下属性部分。
存储池	map[string]StringList	不可以	后端名称到存储池列表的映射
附加存储池	map[string]StringList	不可以	后端名称到存储池列表的映射
排除存储池	map[string]StringList	不可以	后端名称到存储池列表的映射

存储属性及其可能的值可以分为存储池选择属性和 Kubernetes 属性。

存储池选择属性

这些参数决定了应使用哪些 Trident 管理的存储池来配置给定类型的卷。

属性	类型	价值观	提供	要求	由.....支持
媒体[1]	string	机械硬盘、混合硬盘、固态硬盘	Pool 包含此类媒体；混合型媒体是指两者兼具。	指定的媒体类型	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、solidfire-san
供应类型	string	薄的，厚的	池支持这种配置方法	指定的配置方法	厚：全部 ontap；薄：全部 ontap 和 solidfire-san

属性	类型	价值观	提供	要求	由……支持
后端类型	string	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、solidfire-san 、gcp-cvs 、azure-netapp-files、ontap-san-economy	池属于这种类型的后端	指定的后端	所有司机
snapshots	布尔值	真，假	存储池支持带快照的卷	已启用快照的卷	ontap-nas 、ontap-san 、solidfire-san 、gcp-cvs
个克隆	布尔值	真，假	存储池支持卷克隆	已启用克隆的卷	ontap-nas 、ontap-san 、solidfire-san 、gcp-cvs
加密	布尔值	真，假	存储池支持加密卷	已启用加密的卷	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroups、ontap-san
IOPS	整数	正整数	Pool 能够保证在此范围内的 IOPS	容量保证了这些IOPS	solidfire-san

[1]： ONTAP Select系统不支持此系统

在大多数情况下，所请求的值会直接影响配置；例如，请求厚配置会导致厚配置卷的生成。但是，Element 存储池使用其提供的最小和最大 IOPS 来设置 QoS 值，而不是使用请求的值。在这种情况下，请求的值仅用于选择存储池。

理想情况下，你可以使用 `attributes`单独建模，以满足特定类别的需求所需的存储质量。 Trident会自动发现并选择符合所有条件的存储池 `attributes`您指定的。

如果您发现自己无法使用 `attributes`要自动为课程选择合适的池子，您可以使用 `storagePools`和 `additionalStoragePools`参数用于进一步细化池子，甚至选择一组特定的池子。

您可以使用 `storagePools`参数用于进一步限制与任何指定参数匹配的池集合。 `attributes`。换句话说， Trident使用了由以下方式识别的池的交集： `attributes`和 `storagePools`配置参数。您可以单独使用其中一个参数，也可以同时使用两个参数。

您可以使用 `additionalStoragePools`此参数用于扩展Trident用于配置的池集，而不管 Trident 选择的任何池。 `attributes`和 `storagePools`参数。

您可以使用 `excludeStoragePools`用于筛选Trident用于资源配置的池集合的参数。使用此参数会移除所有匹配的池。

在 `storagePools`和 `additionalStoragePools`参数，每个条目都采用以下形式 `<backend>:<storagePoolList>`，在哪里 `<storagePoolList>`是指定后端存储池的逗号分隔列表。例如，一个值 `additionalStoragePools`可能看起来像 `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`。这些列表接受后端值和列表值的正则表达式值。您可以使用 `tridentctl get backend`获取后端及其连接池的列表。

Kubernetes属性

这些属性对Trident在动态配置期间选择存储池/后端没有任何影响。相反，这些属性只是提供 Kubernetes 持久卷支持的参数。工作节点负责文件系统创建操作，可能需要文件系统实用程序，例如 xfsprogs。

属性	类型	价值观	描述	相关驱动因素	**Kubernetes** 版本
文件系统类型	string	ext4、ext3、xfs	块卷的文件系统类型	solidfire-san 、ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、ontap-san-economy	全部
允许卷扩展	布尔值	真，假	启用或禁用对增大PVC尺寸的支持	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、ontap-san-economy、solidfire-san、gcp-cvs、azure-netapp-files	1.11+
容量绑定模式	string	立即，等待首位消费者	选择何时进行卷绑定和动态配置	全部	1.19 - 1.26

- 这 `fsType`该参数用于控制 SAN LUN 所需的文件系统类型。此外，Kubernetes 还利用了以下信息： `fsType`在存储类中表示文件系统存在。可以通过以下方式控制卷所有权： `fsGroup`仅当 `fsType`已设置。请参阅"Kubernetes：为 Pod 或容器配置安全上下文"有关如何使用设置卷所有权的概述 `fsGroup`语境。 Kubernetes 将应用 `fsGroup`仅当满足以下条件时才有值：
  - `fsType`设置在存储类中。
  - PVC接入方式为RWO。

  对于 NFS 存储驱动程序，文件系统已作为 NFS 导出的一部分存在。为了使用 `fsGroup`存储类仍然需要指定一个 `fsType`您可以将其设置为 `nfs`或任何非空值。
- 请参阅"扩大规模"有关扩容的更多详情。
- Trident安装程序包提供了几个示例存储类定义，可供Trident使用。`sample-input/storage-class-*.yaml` 。删除 Kubernetes 存储类会导致相应的Trident存储类也被删除。

Kubernetes `VolumeSnapshotClass`对象

Kubernetes `VolumeSnapshotClass`物体类似于 `StorageClasses`。它们有助于定义多种存储类别，并被卷快照引用，以将快照与所需的快照类别关联起来。每个卷快照都与一个卷快照类相关联。

一个 `VolumeSnapshotClass`应由管理员定义以创建快照。创建卷快照类时，定义如下：

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

这 `driver`向 Kubernetes 指定请求卷快照的 `csi-snapclass`类由Trident处理。这 `deletionPolicy`指定必须删除快照时要执行的操作。什么时候 `deletionPolicy`设置为 `Delete`当删除快照时，存储集群上的卷快照对象以及底层快照也会被删除。或者，将其设置为 `Retain`意味着 `VolumeSnapshotContent`并保留物理快照。

Kubernetes `VolumeSnapshot`对象

Kubernetes `VolumeSnapshot` object 是创建卷快照的请求。 PVC 代表用户对卷的请求，卷快照则是用户对现有 PVC 创建快照的请求。

当收到卷快照请求时， Trident会自动在后端创建卷快照，并通过创建唯一标识符来公开该快照。`VolumeSnapshotContent`目的。您可以从现有的 PVC 创建快照，并在创建新的 PVC 时将这些快照用作数据源。

> VolumeSnapshot 的生命周期与源 PVC 无关：即使源 PVC 被删除，快照仍然会存在。删除具有关联快照的 PVC 时， Trident会将此 PVC 的后备卷标记为“正在删除”状态，但不会将其完全删除。当所有关联的快照都被删除后，该卷也会被移除。

Kubernetes `VolumeSnapshotContent`对象

Kubernetes `VolumeSnapshotContent`该对象表示从已配置的卷中获取的快照。它类似于 `PersistentVolume`表示存储集群上已配置的快照。类似于 `PersistentVolumeClaim`和 `PersistentVolume`当创建快照时，对象会……`VolumeSnapshotContent`对象与……保持一对一的映射关系 `VolumeSnapshot`该对象请求创建快照。

这 `VolumeSnapshotContent`对象包含唯一标识快照的详细信息，例如： `snapshotHandle`。这 `snapshotHandle`是PV名称和名称的独特组合 `VolumeSnapshotContent`目的。

当收到快照请求时， Trident会在后端创建快照。快照创建完成后， Trident会进行配置。`VolumeSnapshotContent`对象，从而将快照暴露给 Kubernetes API。

> 通常情况下，你不需要管理 `VolumeSnapshotContent`目的。但也有例外情况，比如你想……"导入卷快照"在Trident之外创建。

Kubernetes `VolumeGroupSnapshotClass`对象

Kubernetes `VolumeGroupSnapshotClass`物体类似于 `VolumeSnapshotClass`。它们有助于定义多种存储类别，并被卷组快照引用，以将快照与所需的快照类别关联起来。每个卷组快照都与一个卷组快照类相关联。

一个 `VolumeGroupSnapshotClass`应由管理员定义以创建快照组。使用以下定义创建卷组快照类：

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

这 `driver`向 Kubernetes 指定请求卷组快照的权限。 `csi-group-snap-class`类由Trident处理。这 `deletionPolicy`指定当必须删除组快照时要执行的操作。什么时候 `deletionPolicy`设置为 `Delete`当删除快照时，卷组快照对象以及存储集群上的基础快照也会被删除。或者，将其设置为 `Retain`意味着 `VolumeGroupSnapshotContent`并保留物理快照。

Kubernetes `VolumeGroupSnapshot`对象

Kubernetes `VolumeGroupSnapshot`该对象是创建多个卷的快照的请求。就像 PVC 代表用户对卷的请求一样，卷组快照是用户对现有 PVC 创建快照的请求。

当收到卷组快照请求时， Trident会自动在后端管理卷的组快照创建，并通过创建唯一标识符来公开该快照。 `VolumeGroupSnapshotContent`目的。您可以从现有的 PVC 创建快照，并在创建新的 PVC 时将这些快照用作数据源。

> VolumeGroupSnapshot 的生命周期与源 PVC 无关：即使源 PVC 被删除，快照仍然会保留。删除具有关联快照的 PVC 时， Trident会将此 PVC 的后备卷标记为“正在删除”状态，但不会将其完全删除。当所有关联的快照都被删除时，卷组快照也会被删除。

Kubernetes `VolumeGroupSnapshotContent`对象

Kubernetes `VolumeGroupSnapshotContent`该对象表示从已配置的卷中获取的组快照。它类似于 `PersistentVolume`表示存储集群上已配置的快照。类似于 `PersistentVolumeClaim`和 `PersistentVolume`当创建快照时，对象会…… `VolumeSnapshotContent`对象与……保持一对一的映射关系 `VolumeSnapshot`该对象请求创建快照。

这 `VolumeGroupSnapshotContent`对象包含用于标识快照组的详细信息，例如： `volumeGroupSnapshotHandle`以及存储系统上存在的各个 volumeSnapshotHandles。

当收到快照请求时， Trident会在后端创建卷组快照。卷组快照创建完成后， Trident会进行配置。 `VolumeGroupSnapshotContent`对象，从而将快照暴露给 Kubernetes API。

Kubernetes `CustomResourceDefinition`对象

Kubernetes 自定义资源是 Kubernetes API 中的端点，由管理员定义，用于对类似对象进行分组。 Kubernetes 支持创建自定义资源来存储对象集合。您可以通过运行以下命令来获取这些资源定义。 `kubectl get crds` 。

Kubernetes 将自定义资源定义 (CRD) 及其关联的对象元数据存储在其元数据存储中。这样就无需为Trident单独开设商店了。

Trident的使用 `CustomResourceDefinition`用于保存Trident对象标识的对象，例如Trident后端、 Trident存储类和Trident卷。这些对象由Trident管理。此外，CSI 卷快照框架引入了一些定义卷快照所需的 CRD。

CRD 是 Kubernetes 的一种构造。上述资源的对象由Trident创建。举个简单的例子，当使用以下方式创建后端时 `tridentctl`相应的 `tridentbackends`创建 CRD 对象供 Kubernetes 使用。

关于Trident的CRD，需要记住以下几点：

- 安装Trident时，会创建一组 CRD，可以像使用任何其他资源类型一样使用这些 CRD。
- 使用以下方式卸载Trident时 `tridentctl uninstall`使用命令后， Trident pod 会被删除，但创建的 CRD 不会被清理。请参阅"卸载Trident"了解如何将Trident完全移除并从头开始重新配置。

Trident `StorageClass`对象

Trident为 Kubernetes 创建匹配的存储类 `StorageClass`指定对象 `csi.trident.netapp.io`在他们的供应领域。存储类名称与 Kubernetes 的名称匹配。 `StorageClass`它所代表的对象。

> 使用 Kubernetes 时，这些对象会在 Kubernetes 集群启动时自动创建。 `StorageClass`已注册使用Trident作为配置器的配置器。

存储类别包含对卷的一系列要求。 Trident会将这些要求与每个存储池中存在的属性进行匹配；如果匹配，则该存储池是使用该存储类别配置卷的有效目标。

您可以使用 REST API 创建存储类配置，直接定义存储类。但是，对于 Kubernetes 部署，我们期望在注册新的 Kubernetes 实例时创建它们。 `StorageClass`物体。

Trident后端对象

后端代表存储提供商， Trident在其上配置卷；单个Trident实例可以管理任意数量的后端。

> 这是您可以自行创建和管理的两种对象类型之一。另一个是 Kubernetes。 `StorageClass`目的。

有关如何构建这些对象的更多信息，请参阅："配置后端" 。

Trident `StoragePool`对象

存储池代表每个后端可用于配置的不同位置。对于ONTAP而言，这些对应于 SVM 中的聚合。对于NetApp HCI/SolidFire，这些对应于管理员指定的 QoS 频段。对于Cloud Volumes Service，这些对应于云提供商区域。每个存储池都有一组独特的存储属性，这些属性定义了其性能特征和数据保护特征。

与此处的其他对象不同，存储池候选对象始终会自动发现和管理。

Trident `Volume`对象

卷是配置的基本单元，包括后端端点（例如 NFS 共享）以及 iSCSI 和 FC LUN。在 Kubernetes 中，这些直接对应于 `PersistentVolumes`。创建卷时，请确保它具有存储类（决定卷的部署位置）和大小。

- 在 Kubernetes 中，这些对象是自动管理的。您可以查看这些信息，了解Trident 的部署情况。
- 删除带有关联快照的 PV 时，相应的Trident卷将更新为 正在删除 状态。要删除Trident卷，您应该删除该卷的快照。

卷配置定义了已配置卷应具有的属性。

属性	类型	必填项	描述
version	string	不可以	Trident API 版本（“1”）
name	string	可以	要创建的卷的名称
存储类	string	可以	配置卷时要使用的存储类
大小	string	可以	要配置的卷的大小（以字节为单位）
protocol	string	不可以	使用的协议类型：“文件”或“块”
内部名称	string	不可以	存储系统中对象的名称；由Trident生成
克隆源卷	string	不可以	ontap（nas、san）和 solidfire-*：要克隆的卷的名称
splitOnClone	string	不可以	ontap（nas，san）：将克隆体从其父体中分离出来
快照策略	string	不可以	ontap-*：要使用的快照策略
快照储备	string	不可以	ontap-*：为快照预留的卷百分比
出口政策	string	不可以	ontap-nas*：要使用的导出策略
快照目录	布尔值	不可以	ontap-nas*：快照目录是否可见
unix权限	string	不可以	ontap-nas*：初始 UNIX 权限
块大小	string	不可以	solidfire-*：块/扇区大小
文件系统	string	不可以	文件系统类型

Trident生成 `internalName`创建卷时。这包括两个步骤。首先，它会添加存储前缀（默认值）。`trident`或后端配置中的前缀）添加到卷名称，从而得到如下形式的名称 `<prefix>-<volume-name>`。然后它会对名称进行清理，替换后端不允许的字符。对于ONTAP后端，它会将连字符替换为下划线（因此，内部名称变为 `<prefix>_<volume-name>`）。对于 Element 后端，它会将下划线替换为连字符。

您可以使用卷配置通过 REST API 直接配置卷，但在 Kubernetes 部署中，我们预计大多数用户将使用标准的 Kubernetes 管理配置。 `PersistentVolumeClaim`方法。 Trident会在配置过程中自动创建此卷对象。

Trident `Snapshot`对象

快照是卷在特定时间点的副本，可用于配置新卷或恢复状态。在 Kubernetes 中，这些直接对应于 `VolumeSnapshotContent`物体。每个快照都与一个卷相关联，该卷是快照数据的来源。

每个 `Snapshot`对象包含以下属性：

属性	类型	必填项	描述
version	字符串	是	Trident API 版本（“1”）
name	字符串	是	Trident快照对象的名称
内部名称	字符串	是	存储系统上Trident快照对象的名称
volumeName	字符串	是	创建快照的持久卷的名称
volumeInternalName	字符串	是	存储系统上关联的Trident卷对象的名称

在 Kubernetes 中，这些对象是自动管理的。您可以查看这些信息，了解Trident 的部署情况。

当 Kubernetes `VolumeSnapshot`创建对象请求后， `Trident` 的工作原理是在后端存储系统上创建快照对象。这 `internalName`此快照对象是通过组合前缀生成的。 `snapshot-`和 `UID`的 `VolumeSnapshot`对象（例如， `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`）。 `volumeName`和 `volumeInternalName`通过获取支持卷的详细信息来填充。

Trident `ResourceQuota`目的

Trident守护进程消耗一个 `system-node-critical`优先级等级——Kubernetes 中可用的最高优先级等级——以确保Trident能够在节点优雅关闭期间识别和清理卷，并允许Trident daemonset pod 在资源压力高的集群中抢占优先级较低的工作负载。

为了实现这一目标， Trident采用了一种 `ResourceQuota`确保Trident守护进程集上的“system-node-critical”优先级类得到满足。在部署和创建守护进程集之前， Trident会查找 `ResourceQuota`对象，如果未发现，则应用它。

如果您需要对默认资源配额和优先级类别进行更多控制，您可以生成一个 `custom.yaml`或配置 `ResourceQuota`使用 Helm Chart 的对象。

以下是一个 `ResourceQuota` 对象优先考虑Trident守护进程集的示例。

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

有关资源配额的更多信息，请参阅"Kubernetes：资源配额"。

清理 `**ResourceQuota**`如果安装失败

在极少数情况下，如果安装失败， `ResourceQuota`对象已创建，首次尝试"卸载"然后重新安装。

如果这样不行，就手动删除。 `ResourceQuota`目的。

消除 `ResourceQuota`

如果您希望自行控制资源分配，您可以移除Trident。 `ResourceQuota`使用以下命令对象：

```
kubectl delete quota trident-csi -n trident
```

Pod 安全标准 (PSS) 和安全上下文约束 (SCC)

Kubernetes Pod 安全标准 (PSS) 和 Pod 安全策略 (PSP) 定义了权限级别并限制了 Pod 的行为。 OpenShift 安全上下文约束 (SCC) 类似地定义了 OpenShift Kubernetes Engine 特有的 pod 限制。为了实现这种自定义功能， Trident会在安装过程中启用某些权限。以下各节详细介绍了Trident设置的权限。

PSS 取代了 Pod 安全策略 (PSP)。 PSP 在 Kubernetes v1.21 中已被弃用，并将于 v1.25 中移除。更多信息，请参阅"Kubernetes：安全性"。

必需的 **Kubernetes** 安全上下文和相关字段

权限	描述
特权	CSI 要求挂载点是双向的，这意味着Trident节点 pod 必须运行特权容器。更多信息，请参阅"Kubernetes：挂载传播"。

权限	描述
主机网络	iSCSI守护进程需要它。 `iscsiadm`管理 iSCSI 挂载点，并使用主机网络与 iSCSI 守护进程通信。
主机 IPC	NFS 使用进程间通信 (IPC) 与 NFSD 进行通信。
主机 PID	开始需要 `rpc-statd`适用于 NFS。 Trident会查询主机进程以确定是否 `rpc-statd`在挂载 NFS 卷之前运行。

功能	这 `SYS_ADMIN`该功能作为特权容器的默认功能的一部分提供。例如，`Docker` 为特权容器设置了以下功能：
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

赛康普	在特权容器中，Seccomp 配置文件始终为“Unconfined”；因此，它无法在Trident中启用。
SELinux	在 OpenShift 上，特权容器运行在 `spc_t`（“超级特权容器”）域，而非特权容器则运行在……域中。`container_t`领域。在 `containerd`， 和 `container-selinux`安装完成后，所有容器都在运行。`spc_t`域，这实际上禁用了 SELinux。因此， Trident不会增加 `seLinuxOptions`到容器中。
DAC	特权容器必须以 root 用户身份运行。非特权容器以 root 用户身份运行，以访问 CSI 所需的 Unix 套接字。

舱体安全标准 (PSS)

标签	描述	默认
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	允许将Trident控制器和节点添加到安装命名空间中。请勿更改命名空间标签。	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

更改命名空间标签可能会导致 Pod 无法调度，出现“创建时出错：…”或“警告：trident-csi-…”等错误。如果发生这种情况，请检查命名空间标签是否为 `privileged`已更改。如果是这样，请重新安装Trident。

Pod 安全策略 (PSP)

字段	描述	默认
`allowPrivilegeEscalation`	特权容器必须允许权限提升。	`true`
`allowedCSIDrivers`	Trident不使用内联 CSI 临时卷。	空
`allowedCapabilities`	非特权Trident容器不需要比默认集更多的功能，而特权容器将被授予所有可能的功能。	空

字段	描述	默认
allowedFlexVolumes	Trident不使用"FlexVolume驱动器"因此，它们不包含在允许的音量列表中。	空
allowedHostPaths	Trident节点 pod 会挂载节点的根文件系统，因此设置此列表没有任何好处。	空
allowedProcMountTypes	Trident不使用任何 ProcMountTypes。	空
allowedUnsafeSysctls	Trident不需要任何不安全措施 sysctls。	空
defaultAddCapabilities	特权容器无需添加任何功能。	空
defaultAllowPrivilegeEscalation	权限提升权限是在每个Trident pod中处理的。	false
forbiddenSysctls	不 `sysctls`允许。	空
fsGroup	Trident容器以root权限运行。	RunAsAny
hostIPC	挂载 NFS 卷需要主机 IPC 与主机通信 nfsd	true
hostNetwork	iscsiadm 需要主机网络才能与 iSCSI 守护进程通信。	true
hostPID	需要主机 PID 来进行检查 `rpc-statd`正在节点上运行。	true
hostPorts	Trident不使用任何主机端口。	空
privileged	Trident节点 pod 必须运行特权容器才能挂载卷。	true
readOnlyRootFilesystem	Trident节点 pod 必须写入节点文件系统。	false
requiredDropCapabilities	Trident节点 pod 运行的是特权容器，不能放弃任何功能。	none
runAsGroup	Trident容器以root权限运行。	RunAsAny
runAsUser	Trident容器以root权限运行。	runAsAny
runtimeClass	Trident不使用 RuntimeClasses。	空
seLinux	Trident不设置 `seLinuxOptions`因为目前容器运行时和 Kubernetes 发行版在处理 SELinux 方面存在差异。	空
supplementalGroups	Trident容器以root权限运行。	RunAsAny
volumes	Trident pods 需要这些音量插件。	hostPath, projected, emptyDir

安全上下文约束 (SCC)

标签	描述	默认
allowHostDirVolumePlugin	Trident节点 pod 会挂载节点的根文件系统。	true
allowHostIPC	挂载 NFS 卷需要主机 IPC 与主机通信 nfsd。	true
allowHostNetwork	iscsiadm 需要主机网络才能与 iSCSI 守护进程通信。	true
allowHostPID	需要主机 PID 来进行检查 `rpc-statd`正在节点上运行。	true
allowHostPorts	Trident不使用任何主机端口。	false
allowPrivilegeEscalation	特权容器必须允许权限提升。	true
allowPrivilegedContainer	Trident节点 pod 必须运行特权容器才能挂载卷。	true
allowedUnsafeSysctls	Trident不需要任何不安全措施 sysctls。	none
allowedCapabilities	非特权Trident容器不需要比默认集更多的功能，而特权容器将被授予所有可能的功能。	空
defaultAddCapabilities	特权容器无需添加任何功能。	空
fsGroup	Trident容器以root权限运行。	RunAsAny
groups	此 SCC 专用于Trident ，并与其用户绑定。	空
readOnlyRootFilesystem	Trident节点 pod 必须写入节点文件系统。	false
requiredDropCapabilities	Trident节点 pod 运行的是特权容器，不能放弃任何功能。	none
runAsUser	Trident容器以root权限运行。	RunAsAny
seLinuxContext	Trident不设置 `seLinuxOptions`因为目前容器运行时和 Kubernetes 发行版在处理 SELinux 方面存在差异。	空
seccompProfiles	特权容器始终以“非限制”模式运行。	空
supplementalGroups	Trident容器以root权限运行。	RunAsAny
users	提供了一个条目，用于将此 SCC 绑定到Trident命名空间中的Trident用户。	不适用
volumes	Trident pods 需要这些音量插件。	hostPath, downwardAPI, projected, emptyDir