参考
Trident

NetApp
February 02, 2026

目录

参考

Trident端口

详细了解Trident用于通信的端口。

Trident端口

Trident使用以下端口在 Kubernetes 内部进行通信：

Port	目的
8443	后通道 HTTPS
8001	Prometheus 指标端点
8000	Trident REST 服务器
17546	Trident demonset Pod 使用的活动性 / 就绪性探测端口

在安装期间、可以使用更改活跃度/就绪性探测端口 `--probe-port` 标志。请务必确保此端口未被工作节点上的其他进程使用。

Trident REST API

虽然"tridentctl 命令和选项"这是与Trident REST API交互的最简单方式、但您也可以根据需要直接使用REST端点。

何时使用REST API

对于在非Kubnetes部署中使用Trident作为独立二进制文件的高级安装、REST API非常有用。

为了提高安全性、默认情况下、在Pod中运行时、Trident `REST API`仅限于本地主机。要更改此行为、您需要在Pod配置中设置Trident的 `-address`参数。

使用REST API

有关如何调用这些API的示例，请传递debug (`-d`)标志。有关详细信息，请参阅 "使用trdentctrd管理Trident"。

API 的工作原理如下：

获取

`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>`

获得命名对象的详细信息。

发布

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

创建指定类型的对象。

- 需要为要创建的对象配置 JSON 。有关每种对象类型的规范，请参见"使用trdentctrd管理Trident"。
- 如果对象已存在，则行为会有所不同：后端更新现有对象，而所有其他对象类型将使操作失败。

删除

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

删除命名资源。

与后端或存储类关联的卷将继续存在；必须单独删除这些卷。有关详细信息，请参阅 "使用trdentctrd管理Trident"。

命令行选项

Trident为Trident流程编排程序提供了多个命令行选项。您可以使用这些选项修改部署。

日志记录

-debug

启用调试输出。

-loglevel <level>

设置日志记录级别(调试、信息、警告、错误、致命)。默认为 INFO 。

Kubernetes

-k8s_pod

使用此选项或 `-k8s_api_server` 以启用Kubernetes支持。如果设置此值，则 Trident 将使用其所属 POD 的 Kubernetes 服务帐户凭据来联系 API 服务器。只有当 Trident 在启用了服务帐户的 Kubernetes 集群中作为 POD 运行时，此功能才有效。

-k8s_api_server <insecure-address:insecure-port>

使用此选项或启用Kubornetes `-k8s_pod`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。这样、`Trident`便可部署在`POD`之外；但是、它仅支持与`API`服务器的不安全连接。要安全连接、请使用选项在`POD`中部署`Trident` `-k8s_pod`。

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>

侦听此端口、而不侦听UNIX域套接字。

-config <file>

必需；您必须指定后端配置文件的此路径。

REST

-address <ip-or-host>

指定要侦听的三端存储服务器的地址。默认为 localhost 。在本地主机上侦听并在 Kubernetes Pod 中运行时，无法从 Pod 外部直接访问 REST 接口。使用 … `-address ""` 可从Pod IP地址访问REST接口。

> 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。

-port <port-number>

指定应侦听的三端存储服务器的端口。默认为 8000 。

-rest

启用REST接口。默认为 true 。

Kubernetes 和 Trident 对象

您可以通过读取和写入资源对象来使用 REST API 与 Kubernetes 和 Trident 进行交互。Kubernetes 与 Trident ， Trident 与存储以及 Kubernetes 与存储之间的关系由多个资源对象决定。其中一些对象通过 Kubernetes 进行管理，而另一些对象则通过 Trident 进行管理。

对象如何相互交互？

了解对象，对象的用途以及对象交互方式的最简单方法可能是，遵循 Kubernetes 用户的单个存储请求：

1. 用户创建一个 `PersistentVolumeClaim` ，请求管理员先前配置的 Kubernetes `StorageClass` 中具有特定大小的新 `PersistentVolume` 。
2. Kubernetes `StorageClass` 将 Trident 标识为其配置程序，并包含一些参数，用于指示 Trident 如何为请求的类配置卷。
3. Trident 会查看自己的 `StorageClass` 并使用相同的名称来标识匹配的 `Backend` 和 `StoragePools` ，它可以使用这些卷为该类配置卷。
4. Trident 会在匹配的后端配置存储并创建两个对象： Kubernetes 中的 A `PersistentVolume` 用于告知 Kubernetes 如何查找，挂载和处理卷； Trident 中的一个卷用于保留 `PersistentVolume` 与实际存储之间的关系。
5. Kubernetes 会将 `PersistentVolumeClaim` 绑定到新的 `PersistentVolume` 。包含 `PersistentVolumeClaim` 的 Pod 会将此 PersistentVolume 挂载到其运行所在的任何主机上。
6. 用户使用指向 Trident 的 `VolumeSnapshotClass` 创建现有 PVC 的 `VolumeSnapshot` 。

7. Trident 标识与 PVC 关联的卷，并在其后端创建卷的快照。此外，它还会创建一个 `VolumeSnapshotContent` ，指示 Kubernetes 如何识别快照。
8. 用户可以使用 `VolumeSnapshot` 作为源创建 `PersistentVolumeClaim` 。
9. Trident 可识别所需的快照，并执行与创建 `PersistentVolume` 和 `Volume` 相同的一组步骤。

要进一步了解 Kubernetes 对象，强烈建议您阅读 "永久性卷" Kubernetes 文档的一节。

Kubernetes `PersistentVolumeClaim` 对象

Kubernetes `PersistentVolumeClaim` 对象是 Kubernetes 集群用户发出的存储请求。

除了标准规范之外，如果用户要覆盖在后端配置中设置的默认值， Trident 还允许用户指定以下特定于卷的标注：

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	cloneSourceVolume	ontap-nas、ontap-san、solidfire-san、azure-netapp-files、ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-NAS ， ontap-san
trident.netapp.io/protocol	协议	任意
trident.netapp.io/exportPolicy	导出策略	ontap-nas ， ontap-nas-economy-、 ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	snapshotPolicy	ontap-nas ， ontap-nas-economy. ontap-nas-flexgroup ， ontap-san
trident.netapp.io/snapshotReserve	SnapshotReserve	ontap-nas、ontap-nas-flexgroup 、ontap-san
trident.netapp.io/snapshotDirectory	snapshotDirectory	ontap-nas ， ontap-nas-economy-、 ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ontap-nas ， ontap-nas-economy-、 ontap-nas-flexgroup
trident.netapp.io/blockSize	块大小	solidfire-san
trident.netapp.io/skipRecoveryQueue	跳过恢复队列	ontap-nas、ontap-nas-economy 、ontap-nas-flexgroup、ontap-san、ontap-san-economy

如果创建的 PV 具有 `Delete` reclaime 策略，则在释放 PV 时（即用户删除 PVC 时）， Trident 会同时删除 PV 和后备卷。如果删除操作失败， Trident 会将 PV 标记为相应的 PV ，并定期重试此操作，直到操作成功或 PV 手动删除为止。如果 PV 使用 ` retain` 策略， Trident 会忽略该策略，并假定管理员将从 Kubernetes 和后端清理该策略，以便在删除卷之前对其进行备份或检查。请注意，删除 PV 不会通过发生原因 Trident 删除后备卷。您应使用 REST API （`tridentctl` ）将其删除。

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 PVC 。这样， PV 的时

间点副本就可以以快照的形式公开给 Kubernetes 。然后，可以使用快照创建新的 PV 。请查看 ` + 按需卷快照 +` 了解其工作原理。

Trident还提供 `cloneFromPVC` 和 `splitOnClone` 用于创建克隆的标注。您可以使用这些标注克隆PVC、而无需使用CSI实施。

例如：如果用户已经有一个名为 `mysql` 的 PVC ，则用户可以使用标注创建一个名为 `mysqlclone` 的新 PVC ，例如 `trident.netapp.io/cloneFromPVC: mysql` 。设置了此标注后， Trident 将克隆与 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。
- 一个 PVC 及其克隆应位于同一个 Kubernetes 命名空间中，并具有相同的存储类。
- 使用 `ontap-nas` 和 `ontap-san` 驱动程序时，可能需要将 PVC 标注 `trident.netapp.io/splitOnClone` 与 `trident.netapp.io/cloneFromPVC` 结合使用。当 `trident.netapp.io/splitOnClone` 设置为 `true` 时， Trident 会将克隆的卷与父卷拆分，从而将克隆的卷与其父卷的生命周期完全分离，从而降低存储效率。如果不设置 `trident.netapp.io/splitOnClone` 或将其设置为 `false` ，则会减少后端的空间占用，而会在父卷和克隆卷之间创建依赖关系，因此除非先删除克隆，否则无法删除父卷。拆分克隆是有意义的一种情形，即克隆空数据库卷时，该卷及其克隆会发生很大的差异，无法从 ONTAP 提供的存储效率中受益。

。 `sample-input` 目录包含用于Trident的PVC定义示例。请参见 有关与三项技术的卷关联的参数和设置的完整问题描述。

Kubernetes `PersistentVolume` 对象

Kubernetes `PersistentVolume` 对象表示可供 Kubernetes 集群使用的一段存储。它的生命周期与使用它的 POD 无关。

> Trident 会创建 `PersistentVolume` 对象，并根据其配置的卷自动将其注册到 Kubernetes 集群中。您不应自行管理它们。

创建引用基于 Trident 的 `S`存储类 的 PVC 时， Trident 会使用相应的存储类配置新卷并为该卷注册新的 PV 。在配置已配置的卷和相应的 PV 时， Trident 会遵循以下规则：

- Trident 会为 Kubernetes 生成 PV 名称及其用于配置存储的内部名称。在这两种情况下，它都可以确保名称在其范围内是唯一的。
- 卷的大小与 PVC 中请求的大小尽可能匹配，但可能会根据平台将其取整为最接近的可分配数量。

Kubernetes `StorageClass` 对象

Kubernetes `StorageClass` 对象在 `PersistentVolumeClass` 中按名称指定，用于使用一组属性配置存储。存储类本身可标识要使用的配置程序，并按配置程序所了解的术语定义该属性集。

它是需要由管理员创建和管理的两个基本对象之一。另一个是 Trident 后端对象。

使用 Trident 的 Kubernetes `StorageClass` 对象如下所示：

```
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 专用的，可告诉 Trident 如何为类配置卷。

存储类参数包括：

属性	Type	Required	Description
属性	map[string]string	否	请参见下面的属性部分
存储池	map[string]StringList	否	后端名称映射到中的存储池列表
附加 StoragePools	map[string]StringList	否	后端名称映射到中的存储池列表
排除 StoragePools	map[string]StringList	否	后端名称映射到中的存储池列表

存储属性及其可能值可以分类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	Type	值	优惠	请求	支持
介质[1]	string	HDD ，混合，SSD	Pool 包含此类型的介质；混合表示两者	指定的介质类型	ontap-nas ，ontap-nas-economy. ontap-nas-flexgroup ，ontap-san ，solidfire-san
配置类型	string	精简，厚	Pool 支持此配置方法	指定的配置方法	Thick：All ONTAP ；Thin ：All ONTAP & solidfire-san

属性	Type	值	优惠	请求	支持
后端类型	string	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、solidfire-san 、azure-netapp-files、ontap-san-economy	池属于此类型的后端	指定后端	所有驱动程序
snapshots	池	true false	Pool 支持具有快照的卷	启用了快照的卷	ontap-nas 、ontap-san 、solidfire-san
克隆	池	true false	Pool 支持克隆卷	启用了克隆的卷	ontap-nas 、ontap-san 、solidfire-san
加密	池	true false	池支持加密卷	已启用加密的卷	ontap-nas ， ontap-nas-economy-、 ontap-nas-flexgroups ， ontap-san
IOPS	内部	正整数	Pool 能够保证此范围内的 IOPS	卷保证这些 IOPS	solidfire-san

[1] ： ONTAP Select 系统不支持

在大多数情况下，请求的值直接影响配置；例如，请求厚配置会导致卷配置较厚。但是， Element 存储池会使用其提供的 IOPS 最小值和最大值来设置 QoS 值，而不是请求的值。在这种情况下，请求的值仅用于选择存储池。

理想情况下，您可以单独使用 `attributes` 来为满足特定类需求所需的存储质量建模。Trident 会自动发现并选择与您指定的 **属性** 的 *all* 匹配的存储池。

如果您发现自己无法使用 `attributes` 自动为某个类选择合适的池，则可以使用 `storagePools` 和 `addtionalStoragePools` 参数进一步细化池，甚至可以选择一组特定的池。

您可以使用 `storagePools` 参数进一步限制与任何指定的 `attributes` 匹配的池集。换言之， Trident 会使用 `attributes` 和 `storagePools` 参数标识的池的交叉点进行配置。您可以单独使用参数，也可以同时使用这两者。

您可以使用 `addtionalStoragePools` 参数扩展 Trident 用于配置的池集，而不管 `attributes` 和 `storagePools` 参数选择的任何池如何。

您可以使用 `excludeStoragePools` 参数筛选 Trident 用于配置的池集。使用此参数将删除任何匹配的池。

在 `storagePools` 和 `addtionalStoragePools` 参数中，每个条目的格式为 ` <backend>: <storagePoolList>` ，其中 ` <storagePoolList>` 是指定后端的存储池列表，以逗号分隔。例如， `addtionalStoragePools` 的值可能类似于 `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` 来获取后端及其池的列表。

Kubernetes 属性

这些属性不会影响 Trident 在动态配置期间选择的存储池 / 后端。相反，这些属性仅提供 Kubernetes 永久性卷支持的参数。工作节点负责文件系统创建操作，并且可能需要文件系统实用程序，例如 xfsprogs 。

属性	Type	值	Description	相关驱动程序	Kubernetes 版本
FSType	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	boolean	true false	启用或禁用对增加 PVC 大小的支持	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、ontap-san-economy、solidfire-san、azure-netapp-files	1.11 及更高版本
卷绑定模式	string	即时，WaitForFirstConsumer"	选择何时进行卷绑定和动态配置	全部	1.19 - 1.26

- 。 `fsType` 参数用于控制SAN LUN所需的文件系统类型。此外、Kubernetes还会使用 `fsType` 在存储类中以指示文件系统已存在。可以使用控制卷所有权 `fsGroup` 仅当出现此情况时、Pod的安全上下文才会显示 `fsType` 已设置。请参见 "Kubernetes ： 为 Pod 或容器配置安全上下文" 有关使用设置卷所有权的概述 `fsGroup` 环境。Kubernetes将应用 `fsGroup` 只有在以下情况下才为值：

  - 在存储类中设置 `FSType` 。
  - PVC 访问模式为 RW 。

  对于 NFS 存储驱动程序， NFS 导出中已存在文件系统。要使用 `fsGroup` ，存储类仍需要指定 `FSType` 。您可以将其设置为 `NFS` 或任何非空值。

- 请参见 "展开卷" 有关卷扩展的更多详细信息。
- Trident 安装程序包提供了几个示例存储类定义，可用于 `sample-input/storage-class-*` 。 `yaml` 中的 Trident 。删除 Kubernetes 存储类也会删除相应的 Trident 存储类。

Kubernetes `VolumeSnapshotClass` 对象

Kubernetes `VolumeSnapshotClass` 对象类似于 `StorageClasses` 。它们有助于定义多个存储类，并由卷快

照引用以将快照与所需的快照类关联。每个卷快照都与一个卷快照类相关联。

要创建快照，管理员应定义 `VolumeSnapshotClass` 。此时将使用以下定义创建卷快照类：

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

`driver` 指定给 Kubernetes ，由 Trident 处理对 `csI-snapclass` 类的卷快照请求。`deeltionPolicy` 指定必须删除快照时要执行的操作。如果将 `deletionPolicy` 设置为 `Delete` ，则在删除快照时，卷快照对象以及存储集群上的底层快照将被删除。或者，如果将其设置为 `Retain` ，则表示保留 `VolumeSnapshotContent` 和物理快照。

Kubernetes `VolumeSnapshot` 对象

Kubernetes `VolumeSnapshot` 对象是创建卷快照的请求。就像 PVC 代表用户对卷发出的请求一样，卷快照也是用户为现有 PVC 创建快照的请求。

收到卷快照请求后， Trident 会自动管理在后端为卷创建快照的操作，并通过创建唯一的 `VolumeSnapshotContent` 对象公开快照。您可以从现有 PVC 创建快照，并在创建新 PVC 时将这些快照用作 DataSource 。

> VolumeSnapshot 的生命周期与源 PVC 无关：即使源 PVC 被删除，快照仍然存在。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。删除所有关联快照后，卷将被删除。

Kubernetes `VolumeSnapshotContent` 对象

Kubernetes `VolumeSnapshotContent` 对象表示从已配置的卷创建的快照。它类似于 `PersistentVolume` ，表示存储集群上配置的快照。与 `PersistentVolumeClaim` 和 `PersistentVolume` 对象类似，创建快照时， `VolumeSnapshotContent` 对象会与请求创建快照的 `VolumeSnapshot` 对象保持一对一映射。

`VolumeSnapshotContent` 对象包含用于唯一标识快照的详细信息，例如 `snapshotHandle` 。此 `snapshotHandle` 是 PV 名称和 `VolumeSnapshotContent` 对象名称的唯一组合。

收到快照请求后， Trident 会在后端创建快照。创建快照后， Trident 会配置一个 `VolumeSnapshotContent` 对象，从而将快照公开到 Kubernetes API 。

> 通常、您不需要管理 `VolumeSnapshotContent`对象。但是、如果要在Trident外部创建、则会出现一个例外情况"导入卷快照"。

Kubbernetes `VolumeGroupSnapshotClass`对象

Kubbernetes `VolumeGroupSnapshotClass`对象类似于 `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。每个卷组快照都与单个卷组快照类别相关联。

一个 `VolumeGroupSnapshotClass`应由管理员定义，以便创建快照组。卷组快照类使用以下定义创建：

```
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
```

由 Trident 处理。 `deletionPolicy`指定必须删除组快照时要采取的操作。当 `deletionPolicy`设置为 `Delete`，删除快照时，卷组快照对象以及存储集群上的底层快照也将被删除。或者、将其设置为 `Retain`表示将 `VolumeGroupSnapshotContent`保留和物理快照。

Kubbernetes `VolumeGroupSnapshot`对象

Kubernetes `VolumeGroupSnapshot`对象是创建多个卷快照的请求。正如 PVC 代表用户对卷的请求一样，卷组快照是用户为现有 PVC 创建快照的请求。

当卷组快照请求到达时，Trident 会自动管理后端卷的组快照的创建，并通过创建唯一的 `VolumeGroupSnapshotContent`目的。您可以从现有 PVC 创建快照，并在创建新 PVC 时将这些快照用作 DataSource 。

> VolumeGroupSnapshot 的生命周期与源 PVC 无关：即使源 PVC 被删除，快照仍然有效。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，卷组快照也会被移除。

Kubbernetes `VolumeGroupSnapshotContent`对象

Kubernetes `VolumeGroupSnapshotContent`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。与和 `PersistentVolume`对象类似 `PersistentVolumeClaim`、创建快照时、 `VolumeSnapshotContent`对象会与请求创建快照的对象保持一对一映射 `VolumeSnapshot`。

这 `VolumeGroupSnapshotContent`对象包含识别快照组的详细信息，例如 `volumeGroupSnapshotHandle`以及存储系统上现有的各个volumeSnapshotHandles。

当快照请求到达时，Trident 会在后端创建卷组快照。创建卷组快照后，Trident 会配置一个 `VolumeGroupSnapshotContent`对象，从而将快照公开给 Kubernetes API。

Kubernetes `CustomResourceDefinition` 对象

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

自定义资源定义（ CRD ）及其关联的对象元数据由 Kubernetes 存储在其元数据存储中。这样就无需为 Trident 创建单独的存储。

Trident使用 `CustomResourceDefinition`对象保留Trident对象的身份、例如Trident后端、Trident存储类和Trident卷。这些对象由 Trident 管理。此外， CSI 卷快照框架还引入了一些定义卷快照所需的 CRD 。

CRD 是一种 Kubernetes 构造。上述资源的对象由 Trident 创建。例如，使用 `tridentctl` 创建后端时，会创建一个对应的 `tridentbackend` CRD 对象，供 Kubernetes 使用。

有关 Trident 的 CRD ，请注意以下几点：

- 安装 Trident 时，系统会创建一组 CRD ，并可像使用任何其他资源类型一样使用。
- 使用卸载Trident时 `tridentctl uninstall` 命令中、Trident Pod会被删除、但创建的CRD不会被清理。请参见 "卸载 Trident" 了解如何从头开始完全删除和重新配置 Trident 。

Trident `StorageClass`对象

Trident会为Kubernetes创建匹配的存储类 `StorageClass` 指定的对象 `csi.trident.netapp.io` 在其配置程序字段中。存储类名称与Kubernetes的名称匹配 `StorageClass` 它所代表的对象。

使用 Kubernetes 时，如果注册了使用 Trident 作为配置程序的 Kubernetes `StorageClass` ，则会自动创建这些对象。

存储类包含一组卷要求。Trident 会将这些要求与每个存储池中的属性进行匹配；如果匹配，则该存储池是使用该存储类配置卷的有效目标。

您可以使用 REST API 创建存储类配置以直接定义存储类。但是，对于 Kubernetes 部署，我们希望在注册新的 Kubernetes `StorageClass` 对象时创建这些部署。

Trident 后端对象

后端表示存储提供程序，其中 Trident 配置卷；单个 Trident 实例可以管理任意数量的后端。

这是您自己创建和管理的两种对象类型之一。另一个是 Kubernetes `StorageClass` 对象。

有关如何构建这些对象的详细信息、请参见 "正在配置后端"。

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。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

卷是基本配置单元、由后端端端点(例如NFS共享以及iSCSI和FC LUN)组成。在Kubnetes中，这些直接对应于 `PersistentVolumes`。创建卷时，请确保其具有存储类，此类可确定可配置该卷的位置以及大小。

- 在 Kubernetes 中，这些对象会自动进行管理。您可以查看它们以查看 Trident 配置的内容。
- 删除具有关联快照的 PV 时，相应的 Trident 卷将更新为 * 正在删除 * 状态。要删除 Trident 卷，您应删除该卷的快照。

卷配置定义了配置的卷应具有的属性。

属性	Type	Required	Description
version	string	否	Trident API 版本（ "1" ）
name	string	是的。	要创建的卷的名称
存储类	string	是的。	配置卷时要使用的存储类
size	string	是的。	要配置的卷大小（以字节为单位）
协议	string	否	要使用的协议类型； "file" 或 "block"
内部名称	string	否	存储系统上的对象名称；由 Trident 生成
cloneSourceVolume	string	否	ONTAP （ NAS ， SAN ）和 SolidFire — * ： 要从中克隆的卷的名称
splitOnClone	string	否	ONTAP （ NAS ， SAN ）： 将克隆从其父级拆分
snapshotPolicy	string	否	Snapshot-* ： 要使用的 ONTAP 策略
SnapshotReserve	string	否	Snapshot-* ： 为快照预留的卷百分比 ONTAP
导出策略	string	否	ontap-nas* ： 要使用的导出策略
snapshotDirectory	池	否	ontap-nas* ： 是否显示快照目录
unixPermissions	string	否	ontap-nas* ： 初始 UNIX 权限
块大小	string	否	SolidFire — * ： 块 / 扇区大小
文件系统	string	否	文件系统类型
跳过恢复队列	string	否	删除卷时，绕过存储中的恢复队列，立即删除卷。

创建卷时， Trident 会生成 `internalName` 。这包括两个步骤。首先，它会将存储前缀（默认值 `trident` 或后端配置中的前缀）预先添加到卷名称中，从而使名称格式为 ` <prefix>-<volume-name>` 。然后，它将继续清理名称，替换后端不允许使用的字符。对于 ONTAP 后端，它会将连字符替换为下划线（因此，内部名称将变为 ` <prefix>_<volume-name>` ）。对于 Element 后端，它会将下划线替换为连字符。

您可以使用卷配置使用 REST API 直接配置卷，但在 Kubernetes 部署中，我们希望大多数用户使用标准的

Kubernetes `PersistentVolumeClaim` 方法。Trident 会在配置过程中自动创建此卷对象。

Trident `Snapshot`对象

快照是卷的时间点副本，可用于配置新卷或还原状态。在 Kubernetes 中，这些对象直接对应于 `VolumeSnapshotContent` 对象。每个快照都与一个卷相关联，该卷是快照的数据源。

每个 `Snapshot` 对象包括以下属性：

属性	Type	Required	Description
version	string	是的。	Trident API 版本（ "1" ）
name	string	是的。	Trident Snapshot 对象的名称
内部名称	string	是的。	存储系统上 Trident Snapshot 对象的名称
volumeName	string	是的。	为其创建快照的永久性卷的名称
volumeInternalName	string	是的。	存储系统上关联的 Trident 卷对象的名称

在 Kubernetes 中，这些对象会自动进行管理。您可以查看它们以查看 Trident 配置的内容。

创建 Kubernetes `VolumeSnapshot` 对象请求时， Trident 会在备用存储系统上创建 Snapshot 对象。此快照对象的 `internalName` 是通过将前缀 `snapshot-` 与 `VolumeSnapshot` 对象的 `UID` （例如， `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660` ）组合而生成的。`VolumeName` 和 `volumeInternalName` 可通过获取后备卷的详细信息来填充。

Trident `ResourceQuota`对象

Trident守护进程使用优先级类(KubeNet中可用的最高优先级类)、以确保Trident可以在正常节点关闭期间识别和清理卷、并允许Trident守护进程 `system-node-critical`Pod抢占资源压力较高的集群中优先级较低的工作负载。

为此、Trident会使用一个 `ResourceQuota`对象来确保满足Trident守护程序集上的"system-node critical"优先级类。在部署和创建守护进程之前、Trident会查找对象、如果未发现、则会应用该 `ResourceQuota`对象。

如果您需要对默认资源配额和优先级类进行更多控制、可以使用Helm图表生成`custom.yaml`或配置`ResourceQuota`对象。

以下是一个`ResourceQuota`对象的示例、该对象会优先处理Trident子集。

```
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
```

有关资源配额的详细信息、请参见 "Kubernetes：资源配额"。

清理 `ResourceQuota` 如果安装失败

在极少数情况下、如果在创建`ResourceQuota`对象后安装失败、请先尝试 "正在卸载" 然后重新安装。

如果不起作用、请手动删除`ResourceQuota`对象。

删除 `ResourceQuota`

如果您希望控制自己的资源分配、可以使用以下命令删除Trident `ResourceQuota`对象：

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

POD安全标准(PSS)和安全上下文限制(SCC)

Kubernetes Pod安全标准(PSS)和Pod安全策略(PSP)定义权限级别并限制Pod的行为。OpenShift安全上下文约束(SCC)同样定义了特定于OpenShift Kubernetes引擎的POD限制。为了提供此自定义功能、Trident会在安装期间启用某些权限。以下各节详细介绍了Trident设置的权限。

PSS将取代Pod安全策略(PSP)。PSP已在Kubernetes v1.21中弃用、并将在v1.25中删除。有关详细信息、请参见 "Kubernetes：安全性"。

所需的Kubernetes安全上下文和相关字段

权限	Description
特权	CSI要求挂载点为双向挂载点、这意味着Trident节点POD必须运行特权容器。有关详细信息，请参见"Kubernetes：挂载传播"。
主机网络连接	iSCSI守护进程所需的。`iscsiadm`可管理iSCSI挂载、并使用主机网络与iSCSI守护进程进行通信。
主机IPC	NFS使用进程间通信(Interprocess Communication、IPC)与NFSD进行通信。
主机PID	启动NFS时需要此参数 `rpc-statd`。Trident会在挂载NFS卷之前查询主机进程以确定是否 `rpc-statd`正在运行。
功能	`SYS_ADMIN`功能是特权容器的默认功能的一部分。例如、`Docker`会为有权限的容器设置这些功能：`CapPrm：00003fffffffff``CapEff：00003fffffffff`
Seccomp	Seccomp配置文件始终处于"非受限"状态、因此无法在Trident中启用。
SELinux	在OpenShift上、有权限的容器在("超级特权容器")域中运行 `spc_t`、无权限的容器在域中运行 `container_t`。在上 `containerd`，安装后 `container-selinux`，所有容器都在域中运行 `spc_t`，从而有效地禁用SELinux。因此、Trident不会添加 `seLinuxOptions`到容器中。
DAC	有权限的容器必须以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。

POD安全标准(PSS)

Label	Description	Default
`po-security.Kubernetes IO/enforce``po-security.Kubernetes IO/enforce`版本	允许将Trident控制器和节点收入安装命名空间。请勿更改命名空间标签。	`enforce：privileged``enice-version`：<当前集群的版本或测试的最高`PSS`版本。>

更改命名空间标签可能会导致Pod未计划、出现"创建时出错：..."或"警告：Trident CSI -..."。如果发生这种情况、请检查`privileged`的命名空间标签是否已更改。如果是、请重新安装Trident。

POD安全策略(PSP)

字段	Description	Default
`allowPrivilegeEscalation`	有权限的容器必须允许权限升级。	`true`
允许`CSIDrivers`	Trident不使用实时CSI临时卷。	空

字段	Description	Default
`allowedCapabilities`	非特权Trident容器所需的功能不会超过默认设置、而特权容器会获得所有可能的功能。	空
支持`FlexVolumes`	Trident不使用 "FlexVolume驱动程序"、因此它们不会包含在允许的卷列表中。	空
`allowedHostPaths`	Trident节点POD挂载节点的根文件系统、因此设置此列表没有好处。	空
`allowedProcessMountTypes`	Trident不使用任何`ProcMountTypes`。	空
允许`UnsafeSysctls`	Trident不需要任何不安全的`sysctls`。	空
`defaultAddCapabilities`	无需向有权限的容器添加任何功能。	空
`defaultAllowPrivilegeEscalation`	允许权限升级在每个Trident POD中进行处理。	`false`
`forbiddenSysctls`	不允许使用`sysctls`。	空
`fsGroup`	Trident容器以root身份运行。	`RunAsAny`
`hostIPC`	挂载NFS卷需要主机IPC才能与`nfsd`进行通信	`true`
`hostNetwork`	iscsiadm要求主机网络与iSCSI守护进程进行通信。	`true`
`hostPID`	需要使用主机PID来检查节点上是否正在运行`rpc-statd`。	`true`
`hostPorts`	Trident不使用任何主机端口。	空
特权	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运行有权限的容器、无法删除功能。	无
`runAsGroup`	Trident容器以root身份运行。	`RunAsAny`
`runAsUser`	Trident容器以root身份运行。	`runAsAny`
`runtimeClass`	Trident不使用`RuntimeClasses`。	空
`seLinux`	Trident未设置`seLinuxOptions`、因为容器运行时间和Kubernetes分发程序处理SELinux的方式目前存在差异。	空
`supplementalGroups`	Trident容器以root身份运行。	`RunAsAny`

字段	Description	Default
卷	Trident Pod需要这些卷插件。	`hostPath、projected、emptyDir`

安全上下文限制(SCC)

标签	Description	Default
`allowHostDirVolumePlugin`	Trident节点Pod挂载节点的根文件系统。	`true`
`allowHostIPC`	挂载NFS卷需要主机IPC才能与`nfsd`进行通信。	`true`
`allowHostNetwork`	iscsiadm要求主机网络与iSCSI守护进程进行通信。	`true`
`allowHostPID`	需要使用主机PID来检查节点上是否正在运行`rpc-statd`。	`true`
`allowHostPorts`	Trident不使用任何主机端口。	`false`
`allowPrivilegeEscalation`	有权限的容器必须允许权限升级。	`true`
`allowPrivilegedContainer`	Trident节点Pod必须运行特权容器才能挂载卷。	`true`
允许`UnsafeSysctls`	Trident不需要任何不安全的`sysctls`。	无
`allowedCapabilities`	非特权Trident容器所需的功能不会超过默认设置、而特权容器会获得所有可能的功能。	空
`defaultAddCapabilities`	无需向有权限的容器添加任何功能。	空
`fsGroup`	Trident容器以root身份运行。	`RunAsAny`
组	此SCC专用于Trident并绑定到其用户。	空
`readOnlyRootFilesystem`	Trident节点Pod必须写入节点文件系统。	`false`
`requiredDropCapabilities`	Trident节点Pod运行有权限的容器、无法删除功能。	无
`runAsUser`	Trident容器以root身份运行。	`RunAsAny`
`seLinuxContext`	Trident未设置`seLinuxOptions`、因为容器运行时间和Kubernetes分发程序处理SELinux的方式目前存在差异。	空
`seccompProfile`	有权限的容器始终运行"无限制"。	空
`supplementalGroups`	Trident容器以root身份运行。	`RunAsAny`

标签	Description	Default
用户	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
卷	Trident Pod需要这些卷插件。	`hostPath`、`downwardAPI`、`projected`、`emptyDir`