参考
Trident

NetApp
September 26, 2025

This PDF was generated from https://docs.netapp.com/zh-cn/trident-2410/trident-reference/ports.html on September 26, 2025. Always check docs.netapp.com for the latest.

目录

参考 1
Trident端口 1
Trident端口 1
Trident REST API 1
何时使用REST API 1
使用REST API 1
命令行选项 2
日志记录 2
Kubernetes 2
Docker 2
REST 3
Kubernetes 和 Trident 对象 3
对象如何相互交互? 3
Kubbernetes `PersistentVolumeClaim`对象 4
Kubbernetes `PersistentVolume`对象 5
Kubbernetes `StorageClass`对象 5
Kubbernetes `VolumeSnapshotClass`对象 9
Kubbernetes `VolumeSnapshot`对象 9
Kubbernetes `VolumeSnapshotContent`对象 9
Kubbernetes `CustomResourceDefinition`对象 10
Trident `StorageClass`对象 10
Trident 后端对象 10
Trident `StoragePool`对象 11
Trident `Volume`对象 11
Trident `Snapshot`对象 12
Trident `ResourceQuota`对象 12
POD安全标准(PSS)和安全上下文限制(SCC) 13
所需的Kubernetes安全上下文和相关字段 14
POD安全标准(PSS) 14
POD安全策略(PSP) 14
安全上下文限制(SCC) 16

参考

Trident端口

详细了解Trident用于通信的端口。

Trident端口

Trident通过以下端口进行通信：

端口	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

可以在安装期间使用标志更改活动性/就绪探测端口 `--probe-port`。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

使用此选项或启用Kubornetes `-k8s_api_server`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。 `-address ""`用于使REST接口可从POD IP地址访问。

> 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. 用户创建了 `PersistentVolumeClaim`、请求从管理员先前配置的KubureNet获取特定大小的 `StorageClass`新 `PersistentVolume`。
2. Kubernetes `StorageClass`将Trident标识为其配置程序、并包含一些参数、用于告知Trident如何为请求的类配置卷。
3. Trident使用相同的名称查找自己的 `StorageClass`卷、该名称用于标识匹配项 `Backends`、并 `StoragePools`可用于为类配置卷。
4. Trident会在匹配的后端配置存储并创建两个对象：一个 `PersistentVolume`位于`KubeNet`中、用于告知`KubeNet`如何查找、挂载和处理卷；另一个位于`Trident`中、用于保留与实际存储之间的关系 `PersistentVolume`。
5. Kubnetes会将绑定 `PersistentVolumeClaim`到新 `PersistentVolume`。包含在运行此持久卷的任何主机上挂载此持久卷的Pod `PersistentVolumeClaim`。
6. 用户使用指向Trident的创建 `VolumeSnapshot`现有`PVC`的 `VolumeSnapshotClass`。

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`用于指示Kubbernetes如何识别快照。
8. 用户可以使用 `VolumeSnapshot`创建 `PersistentVolumeClaim`作为源。
9. Trident会确定所需的快照，并执行与创建和 `Volume`相同的一组步骤 `PersistentVolume。

要进一步阅读有关Kubbernetes对象的信息、我们强烈建议您阅读 "永久性卷"Kubbernetes文档的章节。

Kubbernetes `PersistentVolumeClaim`对象

Kubbernetes `PersistentVolumeClaim`对象是由Kubbernetes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 ， gcp-cvs ， 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 ， GCP-CVS
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

如果创建的PV具有 `Delete`回收策略、则在PV释放后(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使用此 `Retain`策略、则`Trident`会忽略此策略、并假定管理员将从`Kubbernetes`和后端对其进行清理、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`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 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 。查看 `On-Demand Volume Snapshots`以了解其工作原理。

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

以下是一个示例：如果用户已经有一个名为的 `mysql`PVC，则用户可以使用标注创建一个名为的新PVC `mysqlclone，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：

- 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。
- 对于 `ontap-nas`和 `ontap-san`驱动程序，可能需要将PVC标注 `trident.netapp.io/splitOnClone`与结合使用 `trident.netapp.io/cloneFromPVC`。将设置为 `true`时 `trident.netapp.io/splitOnClone`，Trident会将克隆的卷从父卷中分离出来，从而使克隆卷的生命周期与其父卷完全分离，从而牺牲一些存储效率。如果不将其设置 `trident.netapp.io/splitOnClone`或设置为 `false`、则会减少后端的空间消耗、而这会影响在父卷和克隆卷之间创建依赖关系、从而导致无法删除父卷、除非先删除克隆。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。

该 `sample-input`目录包含用于Trident的PVC定义示例。有关与Trident卷关联的参数和设置的完整说明、请参见。

Kubbernetes `PersistentVolume`对象

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

> Trident会根据所配置的卷自动创建 `PersistentVolume`对象并将其注册到Kubbernetes集群中。您不应自行管理它们。

创建引用基于Trident的PVC时 `StorageClass`，Trident会使用相应的存储类配置新卷，并为该卷注册新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。

Kubbernetes `StorageClass`对象

Kubnetes `StorageClass`对象在中按名称指定 `PersistentVolumeClaims`、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Kubenetes `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	键入	必填	说明
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	键入	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	键入	值	优惠	请求	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 ， gcp-cvs
克隆	池	true false	Pool 支持克隆卷	启用了克隆的卷	ontap-nas ， ontap-san ， solidfire-san ， gcp-cvs
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`来模拟满足特定类需求所需的存储质量。`Trident`会自动发现并选择与您指定的_`all`_匹配的存储池 `attributes`。

如果您发现自己无法使用 `attributes`自动为类选择合适的池、则可以使用和 `additionalStoragePools`参数进一步细化池、甚至可以 `storagePools`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匹配的池集 `attributes`。换言之、Trident使用和 `storagePools`参数标识的池的交叉点 `attributes`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 `additionalStoragePools`参数扩展`Trident`用于配置的池集、而不管和 `storagePools`参数选择了哪些池 `attributes`。

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。

在和 `additionalStoragePools`参数中 `storagePools`，每个条目的格式为

`<backend>:<storagePoolList>`，其中 `<storagePoolList>`是指定后端的存储池的逗号分隔列表。例如，的值 `additionalStoragePools`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`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	键入	值	说明	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 ，gcp-cvs ，azure-netapp-files	1.11多个
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`参数用于控制所需的SAN LUN文件系统类型。此外、Kubnetes还会使用存储类中存在的来指示文件系统存在 `fsType`。只有在设置了后、才能使用POD的安全上下文 `fsType`控制卷所有权 `fsGroup`。有关使用上下文设置卷所有权的概述、 `fsGroup`请参见"Kubernetes： 为 Pod 或容器配置安全上下文"。只有在以下情况下、Kubnetes才会应用此 `fsGroup`值：
  - `fsType`在存储类中设置。
  - 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`。您可以将其设置为或任何非空值。 `nfs`
- 有关卷扩展的详细信息、请参见"展开卷"。
- Trident安装程序包提供了几个示例存储类定义`sample-input/storage-class-*.yaml`，用于中的Trident。删除 Kubernetes 存储类也会删除相应的 Trident 存储类。

Kubbernetes `VolumeSnapshotClass`对象

Kubbernetes `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。

`VolumeSnapshotClass`要创建快照、管理员应定义。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`用于向Kub联网 指定由Trident处理对类的卷快照的请求 `csi-snapclass`。`deletionPolicy`指定在必须删除快照时要执行的操作。如果 `deletionPolicy`将设置为 `Delete`，则在删除快照后，系统将删除卷快照对象以及存储集群上的底层快照。或者、将其设置为 `Retain`表示将 `VolumeSnapshotContent`保留和物理快照。

Kubbernetes `VolumeSnapshot`对象

Kubnetes `VolumeSnapshot`对象是指创建卷快照的请求。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、并通过创建唯一对象来公开快照 `VolumeSnapshotContent`。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：即使删除了源 PVC ，快照也会持续存在。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。

Kubbernetes `VolumeSnapshotContent`对象

Kubbernetes `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。

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

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、从而将快照公开给Kubnetes 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 `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。您可以通过运行来获取这些资源定义 `kubectl get crds`。

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。简单地说，使用创建后端时 `tridentctl`，会创建一个相应的 `tridentbackends`CRD对象供Kubbernetes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。
- 使用命令卸载Trident时 `tridentctl uninstall`、Trident Pod将被删除、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会为在其配置程序字段中指定的Kubbernetes对象 `csi.trident.netapp.io`创建匹配的存储类 `StorageClass`。存储类名称与它所代表的Kubbernetes对象的名称匹配 `StorageClass`。

> 使用Kubnetes时、将在注册使用Trident作为配置程序的Kubnetes时自动创建这些对象 `StorageClass`。

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KubeNet部署、我们希望在注册新的KubeNet对象时创建这些 `StorageClass`对象。

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Kubbernetes `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

存储池表示可在每个后端配置的不同位置。对于 ONTAP ，这些聚合对应于 SVM 中的聚合。对于 NetApp HCI/SolidFire ，这些 QoS 分段对应于管理员指定的 QoS 分段。对于 Cloud Volumes Service ，这些区域对应于云提供商区域。每个存储池都有一组不同的存储属性，用于定义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，由后端端点组成，例如 NFS 共享和 iSCSI LUN 。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	键入	必填	说明
version	string	否	Trident API 版本（ "1" ）
name	string	是	要创建的卷的名称
存储类	string	是	配置卷时要使用的存储类
大小	string	是	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

属性	键入	必填	说明
块大小	string	否	SolidFire — *： 块 / 扇区大小
文件系统	string	否	文件系统类型

Trident会在创建卷时生成 `internalName`。这包括两个步骤。首先，它会在卷名称前面附加存储前缀(默认 `trident`前缀或后端配置中的前缀)，从而生成格式为的名称 `<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。

您可以使用卷配置直接使用REST API配置卷、但在Kubbernetes部署中、我们希望大多数用户使用标准Kubbernetes `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Kubnetes中、这些直接对应于 `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`对象都包括下列属性：

属性	键入	必填	说明
version	字符串	是	Trident API 版本（"1"）
name	字符串	是	Trident Snapshot 对象的名称
内部名称	字符串	是	存储系统上 Trident Snapshot 对象的名称
volumeName	字符串	是	为其创建快照的永久性卷的名称
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Kubnetes `VolumeSnapshot`对象请求后、`Trident`会通过在后备存储系统上创建`Snapshot`对象来工作。此快照对象的是通过将前缀与 `UID`该对象的 `VolumeSnapshot`组合来生成 `snapshot-`的 `internalName`(例如 `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`)。 `volumeName`和 `volumeInternalName`将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别进行更多控制、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"。

如果安装失败、请进行清理 `ResourceQuota`

在创建对象后安装失败的极少数情况下 `ResourceQuota`、请先尝试、"正在卸载"然后再重新安装。

如果不起作用、请手动删除该 `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，请参阅"Kubernetes：安全性"。

所需的Kubernetes安全上下文和相关字段

权限	说明
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，请参阅"Kubernetes：挂载传播"。
主机网络连接	对于iSCSI守护进程为必需项。 `iscsiadm`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	此 `SYS_ADMIN`功能是作为有权限的容器的默认功能的一部分提供的。例如、`Docker`可为有权限的容器设置以下功能：
	`CapPrm: 0000003ffffffffff`
	`CapEff: 0000003ffffffffff`

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)

标签	说明	默认
`pod-security.kubernetes.io/enforce` `pod-security.kubernetes.io/enforce-version`	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` `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、请检查的命名空间标签是否 `privileged`已更改。如果是、请重新安装Trident。

POD安全策略(PSP)

字段	说明	默认
`allowPrivilegeEscalation`	有权限的容器必须允许权限升级。	`true`

字段	说明	默认
allowedCSIDrivers	Trident不使用实时CSI临时卷。	空
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
allowedFlexVolumes	Trident不使用"FlexVolume驱动程序"，因此它们不包括在允许的卷列表中。	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
allowedProcMountTypes	Trident不使用任何 ProcMountTypes。	空
allowedUnsafeSysctls	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	`sysctls`不允许。	空
fsGroup	Trident容器以root身份运行。	RunAsAny
hostIPC	挂载NFS卷需要与主机IPC进行通信 nfsd	true
hostNetwork	iscsiadm要求主机网络与iSCSI守护进程进行通信。	true
hostPID	需要主机PID来检查节点上是否`rpc-statd`正在运行。	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运行有权限的容器、无法删除功能。	none
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	未设置Trident seLinuxOptions、因为容器运行时和Kubnetes分发版处理SELinux的方式目前存在差异。	空
supplementalGroups	Trident容器以root身份运行。	RunAsAny

字段	说明	默认
volumes	Trident Pod需要这些卷插件。	hostPath, projected, emptyDir

安全上下文限制(SCC)

标签	说明	默认
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与进行通信nfsd。	true
allowHostNetwork	iscsiadm要求主机网络与iSCSI守护进程进行通信。	true
allowHostPID	需要主机PID来检查节点上是否`rpc-statd`正在运行。	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
allowedUnsafeSysctls	Trident不需要任何不安全的sysctls。	none
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
groups	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。	none
runAsUser	Trident容器以root身份运行。	RunAsAny
seLinuxContext	未设置Trident seLinuxOptions、因为容器运行时和Kubnetes分发版处理SELinux的方式目前存在差异。	空
seccompProfiles	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	说明	默认
`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 Pod需要这些卷插件。	`hostPath, downwardAPI, projected, emptyDir`

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，本文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。