参考
Astra Trident

NetApp
January 14, 2026

This PDF was generated from https://docs.netapp.com/zh-cn/trident-2310/trident-reference/ports.html on January 14, 2026. Always check docs.netapp.com for the latest.

目录

参考 . . . 1
Astra Trident 端口 . . . 1
Astra Trident 端口 . . . 1
Astra 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
Kubernetes `PersistentVolumeClaim` 对象 . . . 4
Kubernetes `PersistentVolume` 对象 . . . 5
Kubernetes `StorageClass` 对象 . . . 5
Kubernetes `VolumeSnapshotClass` 对象 . . . 9
Kubernetes `VolumeSnapshot` 对象 . . . 9
Kubernetes `VolumeSnapshotContent` 对象 . . . 10
Kubernetes `CustomResourceDefinition` 对象 . . . 10
Astra Trident `StorageClass` 对象 . . . 10
Asta三端对象 . . . 11
Astra Trident `StoragePool` 对象 . . . 11
Astra Trident `Volume` 对象 . . . 11
Astra Trident `Snapshot` 对象 . . . 12
Astra Trident `ResourceQuota` 对象 . . . 13
POD安全标准(PSS)和安全上下文限制(SCC) . . . 14
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) . . . 15
安全上下文限制(SCC) . . . 16

参考

Astra Trident 端口

详细了解Astra Trident用于通信的端口。

Astra Trident 端口

Astra Trident 通过以下端口进行通信：

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。

Astra Trident REST API

同时 "tridentctl 命令和选项" 是与Astra Trident REST API交互的最简单方式、您可以根据需要直接使用REST端点。

何时使用REST API

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

为了提高安全性、我们使用了Astra Trident `REST API` 默认情况下、在Pod内部运行时、仅限于localhost。要更改此行为、您需要设置Astra Trident `-address` 参数。

使用REST API

有关如何调用这些API的示例、请通过调试 (`-d`)标志。有关详细信息，请参见 "使用tridentct利 管理Astra三端"。

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tridentct利 管理Astra三端"。
- 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，请参见 "使用tridentct利 管理Astra三端"。

命令行选项

Astra 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 `-k8s_pod` 以启用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。这样可以在 Pod 之外部署 Trident ；但是，它仅支持与 API 服务器的不安全连接。要安全连接、请使用在Pod中部署Trident `-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>

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 `PersistentVolumeClaim` 请求新的 `PersistentVolume` 的大小 `StorageClass` 之前由管理员配置的。
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独立查看 `StorageClass` 名称相同、用于标识匹配项 `Backends` 和 `StoragePools` 可用于为类配置卷。
4. Trident在匹配的后端配置存储并创建两个对象：A `PersistentVolume` 在Kubernetes中、此命令告诉Kubernetes如何查找、挂载和处理卷、以及在Trident中保留两个卷之间关系的卷 `PersistentVolume` 和实际存储。
5. Kubernetes绑定 `PersistentVolumeClaim` 到新的 `PersistentVolume`。包含的Pod `PersistentVolumeClaim` 将此PersistentVolume挂载到其运行所在的任何主机上。
6. 用户创建 `VolumeSnapshot` 现有PVC、使用 `VolumeSnapshotClass` 这就是Trident。
7. Trident 标识与 PVC 关联的卷，并在其后端创建卷的快照。它还会创建 `VolumeSnapshotContent` 这将指

示Kubernetes如何识别快照。

8. 用户可以创建 `PersistentVolumeClaim` 使用 `VolumeSnapshot` 作为源。
9. Trident可确定所需的快照、并执行与创建相同的一组步骤 `PersistentVolume` 和 A `Volume`。

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

一个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、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经济型、ONTAP—NAS—Flexgroup
trident.netapp.io/snapshotPolicy	snapshotPolicy	ONTAP NAS、ONTAP—NAS经济型、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经济型、ONTAP—NAS—Flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ONTAP NAS、ONTAP—NAS经济型、ONTAP—NAS—Flexgroup
trident.netapp.io/blockSize	块大小	solidfire-san

如果创建的PV具有 `Delete` reclaiming policy、Trident会在PV释放时(即用户删除PVC时)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。PV使用时 `Retain` 策略中、Trident会忽略该策略、并假定管理员将从Kubernetes和后端清理该卷、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 。请查看 `On-Demand Volume Snapshots` 以了解其工作原理。

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。

以下是一个示例：如果用户已有一个名为的PVC `mysql`、用户可以创建一个名为的新PVC `mysqlclone` 使用标注、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。
- 使用 `ontap-nas` 和 `ontap-san` 驱动程序、可能需要设置PVC标注 `trident.netapp.io/splitOnClone` 与结合使用 `trident.netapp.io/cloneFromPVC`。使用 `trident.netapp.io/splitOnClone` 设置为 `true`、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、从而降低了存储效率。未设置 `trident.netapp.io/splitOnClone` 或将其设置为 `false` 这样可以减少后端的空间消耗、而不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

一个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创建 `PersistentVolume` 根据Kubernetes集群所配置的卷、自动将这些对象注册到Kubernetes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一个新卷、并为该卷注册一个新的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` 对象在中按名称指定 `PersistentVolumeClaims` 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。

一个Kubernetes `StorageClass` 使用Trident的对象如下所示：

```
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	将后端名称映射到列表中的存储池数
附加 StoragePools	map[string]StringList	否	后端名称映射中的存储池列表
排除 StoragePools	map[string]StringList	否	后端名称映射到列出中的存储池

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 、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、您可以使用 `attributes` 您需要单独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的_all_匹配的存储池 `attributes` 您指定的。

如果您发现自己无法使用 `attributes` 要自动为某个类选择合适的池、您可以使用 `storagePools` 和 `additionalStoragePools` 用于进一步细化池甚至选择一组特定池的参数。

您可以使用 `storagePools` 参数以进一步限制与指定的任何池匹配的池集 `attributes`。换言之、Trident使用由标识的池的交叉点 `attributes` 和 `storagePools` 用于配置的参数。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` 参数以扩展Trident用于配置的一组池、而不管选择的任何池如何 `attributes` 和 `storagePools` parameters

您可以使用 `excludeStoragePools` 用于筛选Trident用于配置的一组池的参数。使用此参数将删除任何匹配的池。

在中 `storagePools` 和 `additionalStoragePools` 参数、每个条目采用的形式 `<backend>:<storagePoolList>`、其中 `<storagePoolList>` 是指定后端的存储池列表、以英文逗号分隔。例如、的值 `additionalStoragePools` 可能如下所示 `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` 以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 version
FSType	string	ext4 ， ext3 ， xfs 等	块的文件系统类型 volumes	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 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：

  - `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。以便使用 `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、用于中的Trident `sample-input/storage-class-*.yaml`。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。

答 `VolumeSnapshotClass` 要创建快照、应由管理员定义。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` 指定请求的卷快照的Kubernetes `csi-snapclass` 类由Trident处理。。 `deletionPolicy` 指定必须删除快照时要执行的操作。时间 `deletionPolicy` 设置为 `Delete`、卷快照对象以及存储集群上的底层快照会在删除快照时被删除。或者、也可以将其设置为 `Retain` 这意味着 `VolumeSnapshotContent` 并保留物理快照。

Kubernetes `VolumeSnapshot` 对象

一个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、并通过创建唯一快照来公开快照 `VolumeSnapshotContent` 对象。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。

Kubernetes `VolumeSnapshotContent` 对象

一个Kubernetes `VolumeSnapshotContent` 对象表示从已配置的卷创建的快照。它类似于 `PersistentVolume` 和表示存储集群上配置的快照。类似于 `PersistentVolumeClaim` 和 `PersistentVolume` 对象、创建快照时、 `VolumeSnapshotContent` 对象保持与的一对一映射 `VolumeSnapshot` 对象、该对象已请求创建快照。

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

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 `VolumeSnapshotContent` 对象、从而将快照公开到Kubernetes API。

> 通常、您不需要管理 `VolumeSnapshotContent` 对象。但如果需要、则例外 "导入卷快照" 在Asta Trdent外部创建。

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。

Astra Trident使用 `CustomResourceDefinition` 用于保留Trident对象身份的对象、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、使用创建后端时 `tridentctl`、对应的 `tridentbackends` 创建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 。

Astra 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、这些对象会在Kubernetes时自动创建 `StorageClass` 使用Trident作为配置程序进行注册。

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Kubernetes时创建这些部署 `StorageClass` 对象。

Asta三端对象

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"。

Astra 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。

Astra 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Kubernetes中、这些关系直接对应于 `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

属性	Type	Required	Description
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

生成Trident `internalName` 创建卷时。这包括两个步骤。首先、它会预先添加存储前缀(默认值 `trident` 或后端配置中的前缀)添加到卷名称、从而生成表单的名称 `<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。

您可以使用卷配置使用REST API直接配置卷、但在Kubernetes部署中、我们希望大多数用户都使用标准Kubernetes `PersistentVolumeClaim` 方法在配置过程中、 {\f429 trident} 会自动创建此卷对象流程。

Astra 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、这些关系直接对应于 `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。

当Kubernetes时 `VolumeSnapshot` 对象请求已创建、Trident可通过在备用存储系统上创建快照对象来工作。。 `internalName` 的快照对象是通过合并前缀来生成的 `snapshot-` 使用 `UID` 的 `VolumeSnapshot` 对

象(例如、 `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`）。 `volumeName` 和 `volumeInternalName` 将通过获取后备详细信息来填充
卷。

Astra Trident `ResourceQuota` 对象

Trident的降级使用 `system-node-critical` 优先级类—Kubernetes中可用的最高优先级类—用于确保Astra Trident能够在正常节点关闭期间识别和清理卷、并允许Trident demonset 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、Astra Trident采用了 `ResourceQuota` 用于确保满足Trident子集上的"系统节点关键"优先级类的对象。在部署和创建emonset之前、Astra Trident会查找 `ResourceQuota` 对象、如果未发现、则应用此对象。

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

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、安装会在之后失败 `ResourceQuota` 对象已创建、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、则可以删除Astra Trident `ResourceQuota` 使用命令的对象：

```
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。为了提供此自定义设置、Astra Trident会在安装期间启用某些权限。以下各节详细介绍了Astra 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` 管理iSCSI挂载并使用主机网络连接与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	需要启动 `rpc-statd` 对于NFS。Astra Trident会查询主机进程以确定是否 `rpc-statd` 正在挂载NFS卷之前运行。

功能	。 `SYS_ADMIN` 此功能是特权容器的默认功能的一部分。例如、Docker为有权限的容器设置了以下功能：
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccomp	Seccomp配置文件在特权容器中始终处于"非受限"状态；因此、无法在Astra Trident中启用它。
SELinux	在OpenShift上、特权容器在中运行 `spc_t` ("超级特权容器")域和无特权容器在中运行 `container_t` 域。开启 `containerd`、使用 `container-selinux` 安装后、所有容器都会在中运行 `spc_t` 域、这会有效禁用SELinux。因此、Astra Trident不会添加 `seLinuxOptions` 到容器。
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

`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 `privileged` 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`
`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
`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`	否 `sysctls` 允许。	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`	需要使用主机PID检查是否存在 `rpc-statd` 正在节点上运行。	`true`
`hostPorts`	Trident不使用任何主机端口。	空

字段	Description	Default
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
volumes	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	需要使用主机PID检查是否存在 rpc-statd 正在节点上运行。	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

标签	Description	Default
`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` 因为容器运行时间和Kubernetes分发程序处理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`
`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 © 2026 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。