



## 参考 Astra Trident

NetApp  
April 16, 2024

# 目录

- 参考 ..... 1
  - Astra Trident 端口 ..... 1
  - Astra Trident REST API ..... 1
  - 命令行选项 ..... 2
  - 与 Kubernetes 集成的 NetApp 产品 ..... 3
  - Kubernetes 和 Trident 对象 ..... 4
  - tridentctl 命令和选项 ..... 13

# 参考

## Astra Trident 端口

详细了解 Astra Trident 所通过的端口。

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



可以在安装期间使用 `-probe-port` 标志更改活跃度 / 就绪性探测端口。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 的 REST API 进行交互的最简单方式，您可以根据需要直接使用 REST 端点。

这对于在非 Kubernetes 部署中使用 Astra Trident 作为独立二进制文件的高级安装非常有用。

为了提高安全性，在 Pod 内部运行时，Astra Trident 的 REST API 默认限制为 localhost。要更改此行为，您需要在其 Pod 配置中设置 Astra Trident 的 `地址` 参数。

API 的工作原理如下：

### GET

- `get <trident 地址>/trident /v1/< 对象类型 >`：列出此类型的所有对象。
- `get <trident 地址>/trident /v1/< 对象类型 >/< 对象名称 >`：获取命名对象的详细信息。

### POST

`post <trident 地址>/trident /v1/< 对象类型 >`：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 ["ce87eb03803d5633c163541464e9e7f2"](#)。
- 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。

## DELETE

`delete <trident 地址 >/trident /v1/< 对象类型 >/< 对象名称 >`：删除指定的资源。



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 ["ce87eb03803d5633c163541464e9e7f2"](#)。

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，请参见 ["ce87eb03803d5633c163541464e9e7f2"](#)。

## 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`` 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 < 不安全地址：不安全端口 >``：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 `-K8s_pod`` 选项将 Trident 部署在 Pod 中。
- `-K8s_config_path <file>``：必需；必须指定 KubeConfig 文件的路径。

### Docker

- `-volume_driver < 名称 >``：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，而不是 UNIX 域套接字。
- `配置 < 文件 >``：必需；您必须指定后端配置文件的路径。

### REST

- `地址 <ip-or-host>``：指定 Trident 的 REST 服务器应侦听的地址。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 `地址 ""`` 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 Trident 的 REST 服务器应侦听的端口。默认为 `8000`。

- `-REST`：启用 REST 接口。默认为 true。

## 与 Kubernetes 集成的 NetApp 产品

NetApp 存储产品组合可与 Kubernetes 集群的许多不同方面相集成，从而提供高级数据管理功能，从而增强 Kubernetes 部署的功能，功能，性能和可用性。

### Astra

"Astra" 使企业能够更轻松的管理，保护和移动在公有云内部和内部环境中以及之间的 Kubernetes 上运行的数据丰富的容器化工作负载。Astra 使用 NetApp 在公有云和内部环境中成熟而广泛的存储产品组合中的 Trident 配置和提供永久性容器存储。此外，它还提供了一组丰富的高级应用程序感知型数据管理功能，例如快照，备份和还原，活动日志和主动克隆，用于数据保护，灾难 / 数据恢复，数据审核以及 Kubernetes 工作负载的迁移用例。

### ONTAP

ONTAP 是 NetApp 的多协议统一存储操作系统，可为任何应用程序提供高级数据管理功能。ONTAP 系统采用全闪存，混合或全 HDD 配置，并提供多种不同的部署模式，包括专门设计的硬件（FAS 和 AFF），白盒（ONTAP Select）和纯云（Cloud Volumes ONTAP）。



Trident 支持上述所有 ONTAP 部署模式。

### Cloud Volumes ONTAP

"Cloud Volumes ONTAP" 是一款纯软件存储设备，可在云中运行 ONTAP 数据管理软件。您可以将 Cloud Volumes ONTAP 用于生产工作负载、灾难恢复、DevOps、文件共享和数据库管理。它通过提供存储效率，高可用性，数据复制，数据分层和应用程序一致性，将企业级存储扩展到云。

### 适用于 NetApp ONTAP 的 Amazon FSX

"适用于 NetApp ONTAP 的 Amazon FSX" 是一种完全托管的 AWS 服务，可使客户启动和运行由 NetApp ONTAP 存储操作系统提供支持的文件系统。借助适用于 ONTAP 的 FSX，客户可以利用他们熟悉的 NetApp 功能，性能和管理功能，同时利用在 AWS 上存储数据的简便性，灵活性，安全性和可扩展性。FSX for ONTAP 支持 ONTAP 的许多文件系统功能和管理 API。

### Element 软件

"Element" 通过保证性能并简化存储占用空间，使存储管理员能够整合工作负载。Element 与 API 相结合，可实现存储管理各个方面的自动化，可帮助存储管理员事半功倍。

### NetApp HCI

"NetApp HCI" 通过自动化执行日常任务并使基础架构管理员能够专注于更重要的功能，简化数据中心的管理和扩展。

Trident 完全支持 NetApp HCI。Trident 可以直接在底层 NetApp HCI 存储平台上为容器化应用程序配置和管理存储设备。

## Azure NetApp Files

"[Azure NetApp Files](#)" 是一种企业级 Azure 文件共享服务，由 NetApp 提供支持。您可以在 Azure 中以本机方式运行要求最苛刻的基于文件的工作负载，同时享受 NetApp 应有的性能和丰富的数据管理功能。

## 适用于 Google Cloud 的 Cloud Volumes Service

"[适用于 Google Cloud 的 NetApp Cloud Volumes Service](#)" 是一种云原生文件服务，可通过 NFS 和 SMB 提供具有全闪存性能的 NAS 卷。此服务支持在 GCP 云中运行任何工作负载，包括原有应用程序。它提供了一种完全托管的服务，可提供稳定一致的高性能，即时克隆，数据保护以及对 Google Compute Engine（GCE）实例的安全访问。

## 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 , E 系列-iscsi , 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 reclaim 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 还提供了用于创建克隆的 cloneFromPVC 和 splitOnClone 标注。您可以使用这些标注克隆 PVC，而无需使用 CSI 实施（在 Kubernetes 1.13 及更早版本上），或者您的 Kubernetes 版本不支持测试版卷快照（Kubernetes 1.16 及更早版本）。请注意，Trident 19.10 支持从 PVC 克隆的 CSI 工作流。



您可以将 cloneFromPVC 和 splitOnClone 标注与 CSI Trident 以及传统的非 CSI 前端结合使用。

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：

- 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。有关与 Trident 卷关联的参数和设置的完整问题描述，请参见 Trident 卷对象。

## 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/v1beta1
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 <sup>1</sup>	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	厚：全 ONTAP 和 E 系列 iSCSI ； 精简：全 ONTAP 和 solidfire-san
后端类型	string	ontap-nas ， ontap-nas-economy. ontap-nas-flexgroup ， ontap-san ， solidfire-san ， E 系列 iSCSI ， 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

属性	Type	值	优惠	请求	支持
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

<sup>1</sup> : 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 `additionalStoragePools` 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 `additionalStoragePools` 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 `additionalStoragePools` 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，`additionalStoragePools` 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 等	块卷的文件系统类型	solidfire-san , ontap-nas , ontap-nas-economy. ontap-nas-flexgroup , ontap-san , ontap-san-economy-、 E 系列 -iscsi	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	1.17 及更高版本



- FSType 参数用于控制 SAN LUN 所需的文件系统类型。此外，Kubernetes 还会使用存储类中存在的 FSType 来指示文件系统已存在。只有在设置了 FSType 的情况下，才能使用 Pod 的 fsGroup 安全上下文来控制卷所有权。请参见 ["Kubernetes：为 Pod 或容器配置安全上下文"](#) 有关使用 fsGroup 上下文设置卷所有权的概述。只有在以下情况下，Kubernetes 才会应用 fsGroup 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/v1beta1
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。deletionPolicy 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

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。



Trident 会创建 VolumeSnapshotContent 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。

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。

## 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。

从 19.07 版开始，Trident 会使用许多 CustomResourceDefinition 对象来保留 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。
- 从先前版本的 Trident（使用 etcd 保持状态的版本）升级时，Trident 安装程序会从 etcd key-value 数据存储库迁移数据并创建相应的 CRD 对象。
- 使用 `tridentctl uninstall` 命令卸载 Trident 时，Trident Pod 会被删除，但不会清理创建的 CRD。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 StorageClass 在其配置程序字段中指定 `csi.trident.netapp.io/netapp.io/trident` 的对象创建匹配的存储类。存储类名称与 Kubernetes StorageClass 其所代表的对象的名称匹配。



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，请参见后端配置。

## 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 Kubernetes 中，这些卷直接对应于

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

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。

## tridentctl 命令和选项

。"Trident 安装程序包" 包括一个命令行实用程序 tridentctl，可用于轻松访问 Astra Trident。具有足够权限的 Kubernetes 用户可以使用它来安装 Astra Trident，并直接与其交互以管理包含 Astra Trident Pod 的命名空间。

有关使用情况信息，请运行 tridentctl -help。

可用的命令和全局选项包括：

```
Usage:
tridentctl [command]
```

可用命令：

- create：将资源添加到 Astra Trident。

- `d`删除：从 Astra Trident 中删除一个或多个资源。
- 获取：从 Astra Trident 获取一个或多个资源。
- 帮助：有关任何命令的帮助。
- `images`：打印一个包含 Astra Trident 所需容器映像的表。
- `import`：将现有资源导入到 Astra Trident。
- 安装：安装 Astra Trident。
- `logs`：从 Astra Trident 打印日志。
- `sEnd`：从 Astra Trident 发送资源。
- 卸载：卸载 Astra Trident。
- `update`：修改 Astra Trident 中的资源。
- 升级：升级 Astra Trident 中的资源。
- `version`：打印 Astra Trident 的版本。

## flags

- `-d` , `-debug`：调试输出。
- `-h` , `-help`：帮助 `tridentctl`。
- `-n` , `-namespace string`：Astra 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`：Astra Trident REST 接口的地址 / 端口。



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。



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。

## create

您可以使用 `run the create` 命令向 Astra Trident 添加资源。

```
Usage:
tridentctl create [option]
```

可用选项： `backend`：将后端添加到 Astra Trident。

## delete

您可以运行 `delete` 命令从 Astra Trident 中删除一个或多个资源。



```
Usage:
  tridentctl delete [option]
```

可用选项：

- 后端：从 Astra Trident 中删除一个或多个存储后端。
- node：从 Astra Trident 中删除一个或多个 CSI 节点。
- snapshot：从 Astra Trident 中删除一个或多个卷快照。
- storageclass：从 Astra Trident 中删除一个或多个存储类。
- volume：从 Astra Trident 中删除一个或多个存储卷。

## get

您可以运行 `get` 命令从 Astra Trident 获取一个或多个资源。

```
Usage:
  tridentctl get [option]
```

可用选项：

- 后端：从 Astra Trident 获取一个或多个存储后端。
- snapshot：从 Astra Trident 获取一个或多个快照。
- storageclass：从 Astra Trident 获取一个或多个存储类。
- volume：从 Astra Trident 获取一个或多个卷。

## images

您可以运行 `images` 标志来打印包含 Astra Trident 所需容器映像的表。

```
Usage:
  tridentctl images [flags]
```

标志：\* `-h` , `-help`` : 图像帮助。\* `-v` , `-k8s-version` 字符串` : Kubernetes 集群的语义版本。

## import volume

您可以运行 `import volume` 命令将现有卷导入到 Astra Trident 。

Usage:

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

别名: volume , v

flags

- ` -f , -filename string : YAML 或 JSON PVC 文件的路径。
- ` -h , -help : 卷帮助。
- ` -no-manage : 仅创建 PV/PVC 。不要假定卷生命周期管理。

install

您可以运行 install 标志来安装 Astra Trident 。

Usage:

```
tridentctl install [flags]
```

flags

- ` -autosupport-image string : AutoSupport 遥测的容器映像（默认为 "netapp/trident autosupport : 20.07.0" ）。
- ` -autosupport-proxy string : 用于发送 AutoSupport 遥测的代理的地址 / 端口。
- ` -CSI : 安装 CSI Trident （仅对 Kubernetes 1.13 进行覆盖，需要功能安全门）。
- ` -enable-node-prep : 尝试在节点上安装所需的软件包。
- ` -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（默认值为 1m30s ）。
- ` -image-regRegistry string : 内部映像注册表的地址 / 端口。
- ` -K8s-timeout duration : 所有 Kubernetes 操作的超时（默认值为 3 毫秒）。
- ` -kubelet-dir string : kubelet 内部状态的主机位置（默认为 "/var/lib/kubelet" ）。
- ` -log-format string : Astra Trident 日志记录格式（文本， json ）（默认为 "text" ）。
- ` -pv 字符串 : Astra Trident 使用的原有 PV 名称，用于确保此名称不存在（默认为 "trident" ）。
- ` -PVC 字符串 : Astra Trident 使用的原有 PVC 的名称，用于确保此名称不存在（默认为 "trident" ）。
- ` -silencing-autosupport : 不要自动向 NetApp 发送 AutoSupport 捆绑包（默认为 true ）。
- ` -静默 : 在安装期间禁用大多数输出。
- ` -trident 映像字符串 : 要安装的 Astra Trident 映像。

- ` -use-custom-yaml`：使用设置目录中现有的任何 YAML 文件。
- ` -use-ipv6`：使用 IPv6 进行 Astra Trident 的通信。

## logs

您可以运行 logs 标志从 Astra Trident 打印日志。

```
Usage:
  tridentctl logs [flags]
```

## flags

- ` A` , ` -archive`：使用所有日志创建支持归档，除非另有说明。
- ` -h` , ` -help`：日志帮助。
- ` -l` , ` -log` 字符串：要显示的 Astra Trident 日志。Trident 中的一个 "auto"|trident 操作符 "All"（默认为 "auto"）。
- ` -node` string：用于收集节点 Pod 日志的 Kubernetes 节点名称。
- ` -p` , ` -previous`：获取先前容器实例（如果存在）的日志。
- ` -sidecar`：获取 sidecar 容器的日志。

## send

您可以运行 send 命令从 Astra Trident 发送资源。

```
Usage:
  tridentctl send [option]
```

可用选项：AutoSupport：将 AutoSupport 归档发送到 NetApp。

## 卸载

您可以运行 uninstall 标志来卸载 Astra Trident。

```
Usage:
  tridentctl uninstall [flags]
```

标志：` -h` , ` -help`：卸载帮助。` -quiet`：在卸载期间禁用大多数输出。

## update

您可以运行 update 命令来修改 Astra Trident 中的资源。

```
Usage:
  tridentctl update [option]
```

可用选项： `backend`：在 Astra Trident 中更新后端。

## upgrade

您可以运行 `upgrade` 命令来升级 Astra Trident 中的资源。

```
Usage:
  tridentctl upgrade [option]
```

可用选项： `volume`：将一个或多个永久性卷从 NFS/iSCSI 升级到 CSI。

## version

您可以运行 `version` 标志来打印 `tridentctl` 的版本以及正在运行的 Trident 服务。

```
Usage:
  tridentctl version [flags]
```

标志： `* -client`：仅限客户端版本（不需要服务器）。 `* -h`，`-help`：版本帮助。

## 版权信息

版权所有 © 2024 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。