參考
Trident

NetApp
July 01, 2026

目錄

參考

Trident 連接埠

深入瞭解 Trident 用於通訊的連接埠。

概況

Trident 使用各種連接埠與 Kubernetes 叢集內部以及儲存後端進行通訊。以下概述了關鍵連接埠、其用途和安全注意事項。

- 出站流量控制重點：Kubernetes 節點（控制器和工作節點）主要啟動對儲存 LIF/IP 的流量，因此 iptables 規則應允許節點 IP 透過這些連接埠向特定儲存 IP 發起出站流量。避免使用過於寬泛的「任意到任意」規則。
- 入站限制：將內部 Trident 連接埠限制為叢集內部流量（例如，使用 Calico 等 CNI）。主機防火牆上無不必要的入站暴露。
- 協定安全性：
  - 盡可能使用 TCP （更可靠）。
  - 如果敏感，請為 iSCSI 啟用 CHAP/IPsec；為管理啟用 TLS/HTTPS（連接埠 443/8443）。
  - 對於 NFSv4（Trident 中的預設值），如果不需要，請剪除 UDP/ 較舊的 NFSv3 連接埠（例如 4045-4049）。
  - 限制在受信任的子網路內；使用 Prometheus 等工具進行監控（選用連接埠 8001）。

控制器節點的連接埠

這些連接埠主要用於 Trident 操作員（後端管理）。所有內部連接埠均為 Pod 層級；僅當主機防火牆干擾 CNI 時才允許在節點上使用。

連接埠 / 協定	方向	目的	驅動程式 / 通訊協定	安全注意事項
TCP 8000	入站 / 出站（叢集內部）	Trident REST 伺服器（operator-controller 通訊）	全部	僅限使用 pod CIDR；不得暴露於外部環境。
TCP 8443	入站 / 出站（叢集內部）	反向通道 HTTPS （安全內部 API）	全部	採用 TLS 加密；若使用，則僅限於 Kubernetes 服務網格。
TCP 8001	入站（叢集內部、選用）	Prometheus 指標	全部	僅對監控工具開放（例如，使用 RBAC）；如果未使用則停用。
TCP 443	傳出	HTTPS 至 ONTAP SVM/ 叢集管理 LIF	ONTAP（全部）、ANF	需要進行 TLS 憑證驗證；僅限管理 LIF IP 位址。
TCP 8443	傳出	HTTPS 至 E 系列 Web Services Proxy	E 系列（iSCSI）	預設 REST API；使用憑證；可在後端 YAML 中設定。

工作節點的連接埠

這些連接埠用於 CSI 節點守護程序集和 pod 掛載。資料連接埠用於出站連接到儲存資料 LIF；如果使用 NFSv3 ，則包含 NFSv3 附加資訊（NFSv4 為選用）。

連接埠 / 協定	方向	目的	驅動程式 / 通訊協定	安全注意事項
TCP 17546	傳入（本機至 Pod）	CSI 節點存活 / 就緒偵測	全部	可設定（--probe-port）；確保無主機衝突；僅限本機。
TCP 8000	入站 / 出站（叢集內部）	Trident REST 伺服器	全部	如上所述；Pod 內部。
TCP 8443	入站 / 出站（叢集內部）	反向通道 HTTPS	全部	如上所述。
TCP 8001	入站（叢集內部、選用）	Prometheus 指標	全部	如上所述。
TCP 443	傳出	HTTPS 至 ONTAP SVM/ 叢集管理 LIF	ONTAP（全部）、ANF	如上所述；用於探索。
TCP 8443	傳出	HTTPS 至 E 系列 Web Services Proxy	E 系列（iSCSI）	如上所述。
TCP/UDP 111	傳出	RPCBIND/portmapper	ONTAP-NAS（NFSv3/v4）、ANF（NFS）	v3 版本必需；v4 版本可選（防火牆卸載）；如果僅使用 NFSv4 ，則限制使用。
TCP/UDP 2049	傳出	NFS 精靈程式	ONTAP-NAS（NFSv3/v4）、ANF（NFS）	核心資料；眾所周知；使用 TCP 以確保可靠性。
TCP/UDP 635	傳出	掛載精靈程式	ONTAP-NAS（NFSv3/v4）、ANF（NFS）	掛載；可進行雙向回呼（如有需要，允許傳入臨時連線）。
UDP 4045	傳出	NFS 鎖定管理器（nlockmgr）	ONTAP-NAS（NFSv3）	檔案鎖定；跳過 v4 （ pNFS 處理）；僅限 UDP 。
UDP 4046	傳出	NFS 狀態監視器（statd）	ONTAP-NAS（NFSv3）	通知；可能需要入站臨時連接埠 (1024-65535) 進行回調。
UDP 4049	傳出	NFS 配額守護程式（rquotad）	ONTAP-NAS（NFSv3）	配額；v4 版本跳過。
TCP 3260	傳出	iSCSI 目標（探索 / 資料 / CHAP）	ONTAP-SAN（iSCSI）、E-Series（iSCSI）	眾所周知；透過此連接埠進行 CHAP 驗證；啟用雙向 CHAP 以確保安全。
TCP 445	傳出	SMB/CIFS	ONTAP-NAS (SMB)、ANF (SMB)	眾所周知；使用具有加密功能的 SMB3（Trident 註釋 netapp.io/smb-encryption=true）。

連接埠 / 協定	方向	目的	驅動程式 / 通訊協定	安全注意事項
TCP/UDP 88（選用）	傳出	Kerberos 驗證	ONTAP （NFS/SMB/iSCSI with Kerb）	如果使用 Kerberos（非預設）；連接至 AD 伺服器，而非儲存設備。
TCP/UDP 389（選用）	傳出	LDAP	ONTAP （NFS/SMB 搭配 LDAP ）	類似；用於名稱解析 / 驗證；限制為 AD。

安裝過程中可以使用 `--probe-port` 標誌變更存活 / 就緒偵測連接埠。務必確保工作節點上的其他進程沒有佔用此連接埠。

Trident REST API

雖然 "Trident 指令和選項" 是與 Trident REST API 互動的最簡單方法，但如果您願意，也可以直接使用 REST 端點。

何時使用 REST API

REST API 適用於在非 Kubernetes 部署中使用 Trident 作為獨立二進位檔的進階安裝。

為了提高安全性，Trident `REST API` 在 Pod 內運行時預設僅限於本機。若要變更此行為，您需要在 Trident 的 Pod 設定中設定 Trident 的 `-address` 參數。

使用 REST API

若要查看這些 API 的呼叫範例，請傳遞 debug (`-d` 標誌。如需更多資訊，請參閱 "使用 tridentctl 管理 Trident"。

API 的運作方式如下：

取得

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

列出該類型的所有物件。

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

取得指定物件的詳細資訊。

POST

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

建立指定類型的物件。

- 需要為要建立的物件提供 JSON 組態。有關每種物件類型的規格，請參閱 "使用 tridentctl 管理 Trident"。
- 如果物件已存在，則行為會有所不同：後端會更新現有物件，而所有其他物件類型都會使操作失敗。

刪除

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

刪除指定的資源。

> 與後端或儲存類別關聯的磁碟區將繼續存在；這些磁碟區必須單獨刪除。如需更多資訊，請參閱 "使用 tridentctl 管理 Trident"。

命令列選項

Trident 為 Trident 協調器公開了多個命令列選項。您可以使用這些選項來修改部署。

日誌記錄

-debug

啟用偵錯輸出。

-loglevel <level>

設定日誌等級（debug、info、warn、error、fatal）。預設為 info。

Kubernetes

-k8s_pod

使用此選項或 `-k8s_api_server`啟用 Kubernetes 支援。設定此選項會導致 Trident 使用其所在 Pod 的 Kubernetes 服務帳戶認證來聯絡 API 伺服器。此功能僅在 Trident 作為 Pod 在已啟用服務帳戶的 Kubernetes 叢集中執行時才有效。

-k8s_api_server <insecure-address:insecure-port>

使用此選項或 `-k8s_pod`啟用 Kubernetes 支援。指定後，Trident 會使用提供的不安全位址和連接埠連線至 Kubernetes API 伺服器。這使得 Trident 可以部署在 Pod 之外；但是，它僅支援與 API 伺服器的不安全連線。若要安全連線，請使用 `-k8s_pod`選項在 Pod 中部署 Trident。

Docker

-volume_driver <name>

註冊 Docker 外掛程式時使用的驅動程式名稱。預設為 `netapp`。

-driver_port <port-number>

在此連接埠上進行監聽，而非 UNIX 網域通訊端。

-config <file>

必填項；您必須指定後端組態檔的路徑。

REST

-address <ip-or-host>

指定 Trident REST 伺服器監聽的位址。預設為 localhost。當監聽 localhost 且運作在 Kubernetes Pod 內時

，REST 介面無法從 Pod 外部直接存取。使用 `-address ""`可使 REST 介面可透過 Pod IP 位址存取。

> Trident REST 介面可以設定為僅監聽和提供服務於 127.0.0.1 （適用於 IPv4）或 [::1] （適用於 IPv6）。

-port <port-number>

指定 Trident REST 伺服器監聽的連接埠。預設為 8000。

-rest

啟用 REST 介面。預設為 true。

Kubernetes 和 Trident 物件

您可以使用 REST API 透過讀取和寫入資源物件與 Kubernetes 和 Trident 進行互動。有多個資源物件定義了 Kubernetes 與 Trident、Trident 與儲存以及 Kubernetes 與儲存之間的關係。其中一些物件由 Kubernetes 管理，而另一些則由 Trident 管理。

這些物件之間是如何相互作用的？

要了解這些物件、它們的用途以及它們如何互動，最簡單的方法或許是追蹤 Kubernetes 使用者發出的單一儲存請求：

1. 使用者建立一個 `PersistentVolumeClaim`，請求從管理員先前已設定的 Kubernetes `StorageClass`中獲取特定大小的新 `PersistentVolume`。
2. Kubernetes `StorageClass` 將 Trident 識別為其配置器，並包含參數，告訴 Trident 如何為請求的類別配置磁碟區。
3. Trident 會尋找自身 `StorageClass`同名的、用於識別符合項目的 `Backends`和 `StoragePools`，並可利用該實例為該類別配置磁碟區。
4. Trident 在匹配的後端上配置儲存，並建立兩個物件：一個 `PersistentVolume`在 Kubernetes 中，它告訴 Kubernetes 如何尋找、掛載和處理磁碟區，以及 Trident 中的一個磁碟區，它保留了 `PersistentVolume`與實際儲存之間的關係。
5. Kubernetes 將 `PersistentVolumeClaim`綁定到新的 `PersistentVolume`。包含 `PersistentVolumeClaim`的 Pod 會在其運行的任何主機上掛載該 PersistentVolume。
6. 使用者建立一個 `VolumeSnapshot`現有 `PVC` 的 `VolumeSnapshotClass`，該指向 Trident。
7. Trident 會辨識與 PVC 關聯的磁碟區，並在其後端建立該磁碟區的快照。它還會建立一個 `VolumeSnapshotContent`，指示 Kubernetes 如何識別該快照。
8. 使用者可以建立 `PersistentVolumeClaim`，並以 `VolumeSnapshot`作為來源。
9. Trident 識別所需的快照，並執行與建立 `PersistentVolume`和 `Volume`相同的步驟集。

> 如需進一步了解有關 Kubernetes 物件的資訊，我們強烈建議您閱讀 Kubernetes 文件的 "持續磁碟區"章節。

Kubernetes PersistentVolumeClaim 物件

Kubernetes PersistentVolumeClaim 物件是 Kubernetes 叢集使用者發出的儲存請求。

除了標準規格之外、 Trident 還允許使用者指定以下磁碟區專屬註釋、以便在需要時覆寫您在後端組態中設定的預設值：

註解	Volume 選項	支援的驅動程式
trident.netapp.io/fileSystem	fileSystem	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	通訊協定	任何
trident.netapp.io/exportPolicy	exportPolicy	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	blockSize	solidfire-san
trident.netapp.io/skipRecoveryQueue	skipRecoveryQueue	ontap-nas 、 ontap-nas-economy 、 ontap-nas-flexgroup 、 ontap-san 、 ontap-san-economy

如果建立的 PV 啟用了 Delete`回收策略，Trident 會在 PV 釋放時（即使用者刪除 PVC 時）同時刪除 PV 及其後端磁碟區。如果刪除操作失敗，Trident 會將該 PV 標記為失敗，並定期重試該操作，直到成功或 PV 手動刪除。如果 PV 使用了 `Retain`回收策略，Trident 會忽略該策略，並假定管理員會將其從 Kubernetes 和後端清除，從而允許在刪除磁碟區之前對其進行備份或檢查。請注意，刪除 PV 不會導致 Trident 刪除後端磁碟區。您應該使用 REST API 來刪除後端磁碟區（(`tridentctl）。

Trident 支援使用 CSI 規格建立 Volume Snapshot ：您可以建立 Volume Snapshot 並將其做為資料來源來複製現有的 PVC 。如此一來、 PV 的時間點複本就能以快照的形式公開給 Kubernetes 。然後可以使用快照來建立新的 PV 。請參閱 `On-Demand Volume Snapshots`以瞭解其運作方式。

Trident 也提供了 `cloneFromPVC`和 `splitOnClone`註解來建立複本。您可以使用這些註解來複製 PVC，而無需使用 CSI 實作。

例如：如果使用者已有一個名為 mysql`的 PVC，則可以使用註解（例如 `mysqlclone）建立名為 `trident.netapp.io/cloneFromPVC: mysql`的新 PVC。設定此註解後，Trident 會複製與 mysql PVC 對應的磁碟

區，而非從頭配置磁碟區。

請考慮以下幾點：

- NetApp 建議複製閒置的磁碟區。
- PVC 及其複本應位於相同的 Kubernetes 命名空間中，並具有相同的儲存類別。
- 使用 `ontap-nas`和 `ontap-san`驅動程式時，可能需要將 PVC 註解 `trident.netapp.io/splitOnClone`與 `trident.netapp.io/cloneFromPVC`結合使用。將 `trident.netapp.io/splitOnClone`設為 `true`後，Trident 會將複製的磁碟區與父磁碟區分離，從而完全解耦複製磁碟區與其父磁碟區的生命週期，但代價是損失一些儲存效率。不設定 `trident.netapp.io/splitOnClone`或將其設為 `false`會導致後端空間佔用減少，但代價是在父磁碟區和複製磁碟區之間建立依賴關係，使得必須先刪除複製磁碟區才能刪除父磁碟區。在複製空資料庫磁碟區的情況下，分離複製磁碟區是合理的，因為預計該磁碟區及其複製磁碟區會有很大差異，並且無法從 ONTAP 提供的儲存效率中受益。

  `sample-input` 目錄包含可與 `Trident` 搭配使用的 `PVC` 定義範例。如需與 `Trident` 磁碟區相關的參數和設定的完整說明，請參閱 。

Kubernetes `PersistentVolume` 物件

Kubernetes `PersistentVolume` 物件代表一塊可供 Kubernetes 叢集使用的儲存資源。它的生命週期獨立於使用它的 Pod。

Trident 會根據其配置的磁碟區自動建立 `PersistentVolume`物件並將其註冊到 Kubernetes 叢集中。您無需自行管理這些物件。

建立引用基於 Trident 的 `StorageClass`儲存單元（PVC）時，Trident 會使用對應的儲存類別來設定新磁碟區，並為該磁碟區註冊一個新的實體磁碟區（PV）。在配置已設定的磁碟區和對應的 PV 時，Trident 遵循以下規則：

- Trident 會為 Kubernetes 產生一個 PV 名稱，並產生一個用於配置儲存設備的內部名稱。在這兩種情況下，它都會確保名稱在其範圍內是唯一的。
- 磁碟區大小與 PVC 中要求的大小盡可能接近，但可能會向上取整到最接近的可分配數量，具體取決於平台。

Kubernetes `StorageClass` 物件

Kubernetes `StorageClass` 物件透過名稱在 `PersistentVolumeClaims`中指定，以便使用一組屬性來配置儲存設備。儲存類別本身會識別要使用的資源配置程式，並以資源配置程式能夠理解的方式定義該組屬性。

它是管理員需要建立和管理的兩個基本物件之一。另一個是 Trident 後端物件。

使用 Trident 的 Kubernetes `StorageClass` 物件如下所示：

```
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
```

這些參數是 Trident 特有的，它們告訴 Trident 如何為該類別配置磁碟區。

儲存類別參數如下：

屬性	類型	必填	說明
屬性	map[string]string	否	請參閱以下屬性部分
storagePools	map[string]StringList	否	後端名稱到其中儲存池清單的映射
additionalStoragePools	map[string]StringList	否	後端名稱到其中儲存池清單的映射
excludeStoragePools	map[string]StringList	否	後端名稱到其中儲存池清單的映射

儲存屬性及其可能的值可以分為儲存資源池選擇屬性和 Kubernetes 屬性。

儲存資源池選擇屬性

這些參數決定了應使用哪些 Trident 管理的儲存資源池來配置給定類型的磁碟區。

屬性	類型	價值觀	優惠	要求	支援者
媒體[1]	字串	HDD、混合式、SSD	Pool 包含此類型的媒體；混合型表示兩者兼具	指定的媒體類型	ontap-nas 、ontap-nas-economy 、ontap-nas-flexgroup 、ontap-san 、solidfire-san
provisioningType	字串	薄、厚	資源池支援此佈建方法	已指定佈建方法	thick ：所有 ONTAP ；thin ：所有 ONTAP 和 SolidFire-SAN

屬性	類型	價值觀	優惠	要求	支援者
backendType	字串	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、solidfire-san 、azure-netapp-files、ontap-san-economy	Pool 屬於這種類型的後端	指定後端	所有驅動程式
快照	布林值	true、false	Pool 支援帶快照的磁碟區	已啟用快照的磁碟區	ontap-nas 、ontap-san 、solidfire-san
複製	布林值	true、false	儲存池支援複製磁碟區	已啟用複本的磁碟區	ontap-nas 、ontap-san 、solidfire-san
加密	布林值	true、false	儲存池支援加密磁碟區	已啟用加密的磁碟區	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroups、ontap-san
IOPS	int	正整數	Pool 能夠保證此範圍內的 IOPS	Volume 保證了這些 IOPS	solidfire-san

[1]：ONTAP Select 系統不支援

在大多數情況下，請求的值會直接影響資源配置；例如，請求厚資源配置會導致磁碟區採用厚資源配置方式。但是，Element 儲存資源池使用其提供的最小和最大 IOPS 來設定 QoS 值，而不是使用請求的值。在這種情況下，請求的值僅用於選擇儲存資源池。

理想情況下，您可以單獨使用 `attributes`來模擬滿足特定類別需求的儲存特性。Trident 會自動發現並選擇與您指定的 `attributes`所有特性都相符的儲存池。

如果您發現無法使用 `attributes` 自動選擇類別的正確池，則可以使用 `storagePools` 和 `additionalStoragePools` 參數進一步細化池，甚至選擇一組特定的池。

您可以使用 `storagePools`參數進一步限制與任何指定 `attributes`相符的資源池集合。換句話說、Trident 使用由 `attributes`和 `storagePools`參數所識別的資源池交集進行資源配置。您可以單獨使用其中一個參數、也可以同時使用兩個參數。

您可以使用 `additionalStoragePools`參數擴展 Trident 用於配置的池集，而無需考慮透過 `attributes`和 `storagePools`參數選擇的任何池。

您可以使用 `excludeStoragePools` 參數篩選 Trident 用於資源配置的池集合。使用此參數會移除所有符合的池。

在 `storagePools`和 `additionalStoragePools`參數中，每個條目都採用 `<backend>:<storagePoolList>`的形式，其中 `<storagePoolList>`是指定後端的儲存池清單（以逗號分隔）。例如， `additionalStoragePools`的值可能類似於

`ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`。這些清單接受後端和清單值的正規表示式值。您可以使用 `tridentctl get backend`取得後端及其儲存池的清單。

Kubernetes 屬性

這些屬性不會影響 Trident 在動態資源配置期間選取儲存資源池 / 後端。相反地，這些屬性只是提供 Kubernetes Persistent Volume 支援的參數。工作節點負責檔案系統建立作業，可能需要檔案系統公用程式，例如 xfsprogs。

屬性	類型	價值觀	說明	相關驅動程式	**Kubernetes** 版本
fsType	字串	ext4、ext3、xfs	區塊磁碟區的檔案系統類型	solidfire-san、ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy	全部
allowVolumeExpansion	布林值	true、false	啟用或停用對增大 PVC 大小的支援	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy、solidfire-san、azure-netapp-files	1.11+
volumeBindingMode	字串	即時、WaitForFirstConsumer	選擇何時進行磁碟區綁定和動態資源配置	全部	1.19 - 1.26

- `fsType` 參數用於控制 SAN LUNs 所需的檔案系統類型。此外，Kubernetes 也會在儲存類別中使用 `fsType` 來表示檔案系統已存在。只有在設定 `fsType` 時，才能使用 pod 的 `fsGroup` 安全性內容來控制磁碟區擁有權。請參閱 "Kubernetes：為 Pod 或 Container 設定 Security Context" 以取得使用 `fsGroup` 內容設定磁碟區擁有權的概述。Kubernetes 僅會在下列情況套用 `fsGroup` 值：
  - `fsType` 設定在儲存類別中。
  - PVC 存取模式為 RWO。

  對於 NFS 儲存驅動程式，檔案系統已作為 NFS 匯出的一部分存在。若要使用 `fsGroup`，儲存類別仍需指定 `fsType`。您可以將其設定為 `nfs`或任何非空值。
- 如需磁碟區擴充的詳細資訊，請參閱 "擴充磁碟區"。
- Trident 安裝程式套件提供了幾個範例儲存類別定義，可與 Trident 搭配使用於`sample-input/storage-class-*.yaml`。刪除 Kubernetes 儲存類別也會導致對應的 Trident 儲存類別被刪除。

Kubernetes VolumeSnapshotClass 物件

Kubernetes VolumeSnapshotClass 物件類似於 StorageClasses。它們用於定義多種儲存類別，並由磁碟區快照引用，以便將快照與所需的快照類別關聯起來。每個磁碟區快照都與一個磁碟區快照類別相關聯。

若要建立快照，必須由管理員定義 VolumeSnapshotClass。Volume Snapshot Class 按以下定義建立：

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

`driver`向 Kubernetes 指定 `csi-snapclass`類別的磁碟區快照請求由 Trident 處理。`deletionPolicy`指定必須刪除快照時要執行的動作。當 `deletionPolicy`設定為 `Delete`時，刪除快照時會同時移除磁碟區快照物件以及儲存叢集上的底層快照。或者，將其設定為 `Retain`表示 `VolumeSnapshotContent`和實體快照將被保留。

Kubernetes VolumeSnapshot 物件

Kubernetes VolumeSnapshot 物件是建立磁碟區快照的請求。正如 PVC 代表使用者對磁碟區的請求一樣，磁碟區快照是使用者對現有 PVC 建立快照的請求。

當收到磁碟區快照請求時，Trident 會自動在後端建立磁碟區快照，並透過建立一個唯一 `VolumeSnapshotContent`物件來公開該快照。您可以從現有 PVC 建立快照，並在建立新 PVC 時使用這些快照作為 DataSource。

VolumeSnapshot 的生命週期與來源 PVC 無關：即使來源 PVC 被刪除，快照仍然存在。刪除具有相關快照的 PVC 時， Trident 會將此 PVC 的備份磁碟區標記為 **Deleting** 狀態，但不會完全移除。當所有關聯的快照都被刪除後，該磁碟區才會被移除。

Kubernetes VolumeSnapshotContent 物件

Kubernetes VolumeSnapshotContent 物件表示從已配置的磁碟區中取得的快照。它類似於 PersistentVolume，表示儲存叢集上已配置的快照。與 `PersistentVolumeClaim`和 `PersistentVolume`物件類似，建立快照時， `VolumeSnapshotContent`物件會維護與 `VolumeSnapshot`物件一一對應的關係，該物件要求建立快照。

該 VolumeSnapshotContent`物件包含唯一標識快照的詳細資訊，例如 `snapshotHandle。這 `snapshotHandle`是 PV 名稱和 `VolumeSnapshotContent`物件名稱的唯一組合。

當收到快照請求時， Trident 會在後端建立快照。快照建立完成後， Trident 會配置一個 `VolumeSnapshotContent`物件，從而將快照公開給 Kubernetes API。

通常情況下，您無需管理該 `VolumeSnapshotContent` 物件。但如果您想 "匯入 Volume Snapshot" 在 Trident 之外建立。

Kubernetes `VolumeGroupSnapshotClass` 物件

Kubernetes `VolumeGroupSnapshotClass` 物件類似於 `VolumeSnapshotClass`。它們有助於定義多個儲存類別，並由磁碟區群組快照參照，以便將快照與所需的快照類別建立關聯。每個磁碟區群組快照都與單一磁碟區群組快照類別建立關聯。

管理員應定義 `VolumeGroupSnapshotClass`，以便建立快照群組。磁碟區群組快照類別的建立定義如下：

```
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
```

`driver`指定 Kubernetes 將 `csi-group-snap-class`類別的磁碟區群組快照要求交由 Trident 處理。 `deletionPolicy`指定必須刪除群組快照時要執行的動作。當 `deletionPolicy`設定為 `Delete`時，刪除快照時會同時移除磁碟區群組快照物件以及儲存叢集上的底層快照。或者，將其設定為 `Retain`表示 `VolumeGroupSnapshotContent`和實體快照將被保留。

Kubernetes `VolumeGroupSnapshot` 物件

Kubernetes `VolumeGroupSnapshot` 物件是建立多個磁碟區快照的請求。正如 PVC 代表使用者對磁碟區提出的請求一樣，磁碟區群組快照是使用者對建立現有 PVC 快照提出的請求。

當收到磁碟區組快照請求時， Trident 會自動在後端建立磁碟區的群組快照，並透過建立唯一的 `VolumeGroupSnapshotContent`物件來公開該快照。您可以從現有 PVC 建立快照，並在建立新 PVC 時使用這些快照作為 DataSource。

VolumeGroupSnapshot 的生命週期與來源 PVC 無關：即使刪除來源 PVC ，快照仍會保留。刪除具有相關快照的 PVC 時， Trident 會將此 PVC 的備份磁碟區標記為 **Deleting** 狀態，但不會完全移除。刪除所有相關快照時，就會移除磁碟區群組快照。

Kubernetes `VolumeGroupSnapshotContent` 物件

Kubernetes `VolumeGroupSnapshotContent` 物件表示從已配置的磁碟區中取得的群組快照。它類似於 `PersistentVolume`，表示儲存叢集上已配置的快照。與 `PersistentVolumeClaim`和 `PersistentVolume`物件類似，建立快照時， `VolumeSnapshotContent`物件會維護與 `VolumeSnapshot`物件一一對應的關係，該物件要求建立快照。

該 `VolumeGroupSnapshotContent`物件包含用於識別快照群組的詳細資訊，例如 `volumeGroupSnapshotHandle`以及儲存系統上現有的個別 volumeSnapshotHandles。

當收到快照請求時、 Trident 會在後端建立磁碟區群組快照。磁碟區群組快照建立完成後、 Trident 會設定 `VolumeGroupSnapshotContent`物件、從而將快照公開給 Kubernetes API 。

Kubernetes `CustomResourceDefinition` 物件

Kubernetes 自訂資源是 Kubernetes API 中的端點，由管理員定義，用於將相似物件分組。Kubernetes 支援建立自訂資源來儲存物件集合。您可以透過執行 `kubectl get crds`來取得這些資源定義。

Kubernetes 將自訂資源定義（CRD）及其關聯的物件中繼資料儲存在其中繼資料存放區中。這樣就不需要為 Trident 設置單獨的存放區。

Trident 使用 `CustomResourceDefinition`物件來維護 Trident 物件的身份，例如 Trident 後端、Trident 儲存類別和 Trident 磁碟區。這些物件由 Trident 管理。此外，CSI Volume Snapshot 架構引入了一些定義 Volume Snapshot 所需的 CRD。

CRD 是 Kubernetes 的一種建構。上述資源的物件由 Trident 建立。例如，當使用 `tridentctl`建立後端時，會建立一個對應的 `tridentbackends` CRD 物件供 Kubernetes 使用。

關於 Trident 的 CRD，需要記住以下幾點：

- 安裝 Trident 時，會建立一組 CRD，可以像使用任何其他資源類型一樣使用這些 CRD。
- 使用 `tridentctl uninstall`命令卸載 Trident 時、Trident Pod 會被刪除、但建立的 CRD 不會被清除。請參閱"解除安裝 Trident"以瞭解如何完全移除 Trident 並從頭開始重新設定。

Trident `StorageClass` 物件

Trident 會為 Kubernetes `StorageClass` 物件建立相符的儲存類別，這些物件在其 provisioner 欄位中指定 `csi.trident.netapp.io`。儲存類別名稱與其所代表的 Kubernetes `StorageClass` 物件名稱相符。

使用 Kubernetes 時，當使用 Trident 作為佈建程式的 Kubernetes `StorageClass`註冊時，這些物件會自動建立。

儲存類別包含一系列磁碟區需求。Trident 會將這些需求與每個儲存池中存在的屬性進行匹配；如果匹配，則該儲存池是使用該儲存類別配置磁碟區的有效目標。

您可以使用 REST API 直接建立儲存類別配置來定義儲存類別。但是，對於 Kubernetes 部署，我們希望在註冊新的 Kubernetes `StorageClass` 物件時建立儲存類別配置。

Trident 後端物件

後端代表 Trident 在其上配置磁碟區的儲存提供者；單一 Trident 執行個體可以管理任意數量的後端。

這是您可以自行建立和管理的兩種物件類型之一。另一種是 Kubernetes `StorageClass` 物件。

如需如何建構這些物件的詳細資訊，請參閱 "配置後端"。

Trident `StoragePool` 物件

儲存池代表每個後端可用於資源配置的不同位置。對於 ONTAP，這些儲存池對應於 SVM 中的聚合。對於 NetApp HCI/SolidFire，這些儲存池對應於管理員指定的 QoS 頻段。每個儲存池都有一組不同的儲存屬性，這些屬性定義了其效能特徵和資料保護特徵。

與此處的其他物件不同、儲存資源池候選對象始終會自動探索和管理。

Trident `Volume` 物件

磁碟區是基本的資源配置單元，包含後端端點（例如 NFS 共用）、iSCSI 和 FC LUN。在 Kubernetes 中，這些端點直接對應至 `PersistentVolumes`。建立磁碟區時，請確保其具有儲存類別（用於確定磁碟區的配置位置）和大小。

- 在 Kubernetes 中，這些物件由系統自動管理。您可以查看這些物件，以了解 Trident 已配置了哪些資源。
- 刪除包含關聯快照的 PV 時，對應的 Trident 磁碟區會更新為 **Deleting** 狀態。若要刪除 Trident 磁碟區，您需要先刪除該磁碟區的快照。

Volume 組態定義了已配置 Volume 應具有的內容。

屬性	類型	必填	說明
版本	字串	否	Trident API 版本（「1」）
姓名	字串	是的	要建立的磁碟區名稱
storageClass	字串	是的	配置磁碟區時要使用的儲存類別
尺寸	字串	是的	要配置的磁碟區大小（以位元組為單位）
通訊協定	字串	否	使用的協定類型；「file」或「block」
internalName	字串	否	儲存系統中物件的名稱；由 Trident 產生
cloneSourceVolume	字串	否	ontap（nas，san）＆solidfire-*：要複製的 Volume 名稱
splitOnClone	字串	否	ONTAP（NAS、SAN）：將複本從其父項分割
snapshotPolicy	字串	否	ontap-*：要使用的 Snapshot 原則
snapshotReserve	字串	否	ontap-*：為 Snapshot 預留的 Volume 百分比
exportPolicy	字串	否	ontap-nas*：要使用的匯出原則
snapshotDirectory	布林值	否	ontap-nas*：Snapshot 目錄是否可見

屬性	類型	必填	說明
unixPermissions	字串	否	ontap-nas*：初始 UNIX 權限
blockSize	字串	否	SolidFire-*：區塊/磁區大小
fileSystem	字串	否	檔案系統類型
skipRecoveryQueue	字串	否	刪除磁碟區時、繞過儲存設備中的還原佇列、並立即刪除磁碟區。

Trident 在建立磁碟區時會產生 `internalName`。此過程分為兩個步驟。首先，它會在磁碟區名稱前面加上儲存前綴（可以是預設 `trident`或後端配置中的前綴），產生 `<prefix>-<volume-name>`格式的名稱。然後，它會對名稱進行清理，取代後端不允許的字元。對於 `ONTAP` 後端，它會將連字號替換為底線（因此，內部名稱變成 `<prefix>_<volume-name>`）。對於 Element 後端，它會將底線替換為連字號。

您可以使用磁碟區配置透過 REST API 直接設定磁碟區，但在 Kubernetes 部署中，我們預期大多數使用者會使用標準的 Kubernetes `PersistentVolumeClaim` 方法。Trident 會在設定過程中自動建立此磁碟區物件。

Trident `Snapshot` 物件

快照是磁碟區在特定時間點的副本，可用於設定新磁碟區或復原狀態。在 Kubernetes 中，快照直接對應於 `VolumeSnapshotContent` 物件。每個快照都與一個磁碟區關聯，該磁碟區是快照資料的來源。

每個 `Snapshot` 物件都包含以下屬性：

屬性	類型	必填	說明
版本	字串	是的	Trident API 版本（「1」）
姓名	字串	是的	Trident 快照物件的名稱
internalName	字串	是的	儲存系統上 Trident 快照物件的名稱
volumeName	字串	是的	建立快照的持久磁碟區名稱
volumeInternalName	字串	是的	儲存系統上關聯的 Trident 磁碟區物件名稱

在 Kubernetes 中，這些物件由系統自動管理。您可以查看這些物件，以了解 Trident 已配置了哪些資源。

當建立 Kubernetes `VolumeSnapshot` 物件請求時、 Trident 會在後端儲存系統上建立快照物件。此快照物件的 `internalName`是由前置碼 `snapshot-`與 `UID`物件的 `VolumeSnapshot`組合而成（例如、 `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`）。 `volumeName`和 `volumeInternalName`則透過取得後端磁碟區的詳細資料來填入。

Trident `ResourceQuota` 物件

Trident 守護程序集使用 `system-node-critical` Priority Class（Kubernetes 中可用的最高 Priority Class），以確保 Trident 能夠在節點優雅關閉期間識別和清理磁碟區，並允許 Trident 守護程序集 Pod 在資源壓力高的叢集中搶佔優先順序較低的工作負載。

為了實現這一點，Trident 使用一個 `ResourceQuota`物件來確保 Trident 常駐程式集符合「系統節點關鍵」優先權類別。在部署和建立常駐程式集之前，Trident 會尋找該 `ResourceQuota`物件，如果找不到，則套用該物件。

如果您需要對預設 Resource Quota 和 Priority Class 進行更多控制，可以產生 `custom.yaml`或使用 Helm chart 配置 `ResourceQuota`物件。

以下是一個 `ResourceQuota` 物件優先考慮 Trident daemonset 的範例。

```
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
```

如需資源配額的詳細資訊，請參閱 "Kubernetes：Resource Quotas"。

如果安裝失敗，請進行清理 `ResourceQuota`

如果在建立 `ResourceQuota` 物件後安裝失敗（這種情況很少見）、請先嘗試 "解除安裝"、然後再重新安裝。

如果這樣不行，請手動移除 `ResourceQuota` 物件。

移除 `ResourceQuota`

如果您希望自行控制資源分配，可以使用下列指令刪除 Trident `ResourceQuota` 物件：

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

Pod Security Standards（PSS）和 Security Context Constraints（SCC）

Kubernetes Pod Security Standards（PSS）和 Pod Security Policies（PSP）定義了權限等級並限制 pod 的行為。OpenShift Security Context Constraints（SCC）同樣針對 OpenShift Kubernetes Engine 定義了 pod 限制。為了提供此自訂功能，Trident 在安裝期間啟用特定權限。以下章節將詳細說明 Trident 設定的權限。

PSS 取代了 Pod Security Policies（PSP）。PSP 已在 Kubernetes v1.21 中棄用，並將於 v1.25 中移除。如需更多資訊，請參閱"Kubernetes：安全性"。

必要的 Kubernetes 安全性內容和相關欄位

權限	說明
特權	CSI 要求掛載點是雙向的，這表示 Trident 節點 pod 必須執行特權容器。如需更多資訊，請參閱 "Kubernetes：掛載傳播"。
主機網路	iSCSI 守護程式需要此元件。 `iscsiadm` 管理 iSCSI 掛載點，並使用主機網路與 iSCSI 守護程序通訊。
主機 IPC	NFS 使用進程間通訊（IPC）與 NFSD 進行通訊。
主機 PID	啟動 `rpc-statd` 以進行 NFS 時需要此項目。Trident 會查詢主機進程以確定 `rpc-statd` 是否正在執行，然後再掛載 NFS 磁碟區。

功能	`SYS_ADMIN` 功能作為特權容器的預設功能的一部分提供。例如，docker 為特權容器設定了以下功能：
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccomp	在特權容器中，Seccomp 設定檔始終為「Unconfined」；因此，它無法在 Trident 中啟用。
SELinux	在 OpenShift 上，特權容器運行在 `spc_t`（「超級特權容器」）網域中，非特權容器運行在 `container_t` 網域中。在 `containerd` 上，如果安裝了 `container-selinux`，則所有容器都在 `spc_t` 網域中運行，這實際上禁用了 SELinux。因此，Trident 不會將 `seLinuxOptions` 添加到容器中。
DAC	特權容器必須以 root 使用者身分執行。非特權容器以 root 使用者身分執行，以便存取 CSI 所需的 unix 套接字。

Pod 安全標準（PSS）

標籤	說明	預設
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	允許 Trident Controller 和節點新增至安裝命名空間。請勿變更命名空間標籤。	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

更改命名空間標籤可能會導致 Pod 無法調度，並出現「建立錯誤：……」或「警告：trident-csi-……」等錯誤訊息。如果發生這種情況，請檢查 `privileged`命名空間標籤是否已變更。如果是，請重新安裝 Trident。

Pod 安全性原則 (PSP)

欄位	說明	預設
`allowPrivilegeEscalation`	特權容器必須允許權限提升。	`true`
`allowedCSIDrivers`	Trident 不使用內嵌 CSI 臨時性磁碟區。	空白
`allowedCapabilities`	非特權 Trident 容器不需要比預設集更多的功能，而特權容器將被授予所有可能的功能。	空白
`allowedFlexVolumes`	Trident 不使用 "FlexVolume 驅動程式"，因此它們不包含在允許的磁碟區清單中。	空白
`allowedHostPaths`	Trident 節點 pod 會掛載節點的根檔案系統，因此設定此清單沒有任何好處。	空白
`allowedProcMountTypes`	Trident 不使用任何 `ProcMountTypes`。	空白
`allowedUnsafeSysctls`	Trident 不需要任何不安全的 `sysctls`。	空白
`defaultAddCapabilities`	特權容器無需新增任何功能。	空白
`defaultAllowPrivilegeEscalation`	允許權限提升是在每個 Trident pod 中處理的。	`false`
`forbiddenSysctls`	不允許 `sysctls`。	空白
`fsGroup`	Trident 容器以 root 權限運作。	`RunAsAny`
`hostIPC`	掛載 NFS 磁碟區需要主機 IPC 與 `nfsd`通訊	`true`
`hostNetwork`	iscsiadm 需要主機網路才能與 iSCSI 精靈程式通訊。	`true`
`hostPID`	需要主機 PID 來檢查 `rpc-statd` 是否在節點上運行。	`true`
`hostPorts`	Trident 不使用任何主機連接埠。	空白

欄位	說明	預設
privileged	Trident 節點 Pod 必須執行特權容器才能掛載磁碟區。	true
readOnlyRootFilesystem	Trident 節點 Pod 必須寫入節點檔案系統。	false
requiredDropCapabilities	Trident 節點 Pod 運作的是特權容器，無法放棄任何功能。	none
runAsGroup	Trident 容器以 root 權限運作。	RunAsAny
runAsUser	Trident 容器以 root 權限運作。	runAsAny
runtimeClass	Trident 不使用 RuntimeClasses。	空白
seLinux	Trident 未進行設置 seLinuxOptions，因為目前容器執行階段和 Kubernetes 發行版處理 SELinux 的方式有差異。	空白
supplementalGroups	Trident 容器以 root 權限運作。	RunAsAny
volumes	Trident Pod 需要這些磁碟區外掛程式。	hostPath, projected, emptyDir

安全內容限制 (SCC)

標籤	說明	預設
allowHostDirVolumePlugin	Trident 節點 Pod 會掛載節點的根檔案系統。	true
allowHostIPC	掛載 NFS 磁碟區需要主機 IPC 與 `nfsd`通訊。	true
allowHostNetwork	iscsiadm 需要主機網路才能與 iSCSI 精靈程式通訊。	true
allowHostPID	需要主機 PID 來檢查 rpc-statd 是否在節點上運行。	true
allowHostPorts	Trident 不使用任何主機連接埠。	false
allowPrivilegeEscalation	特權容器必須允許權限提升。	true
allowPrivilegedContainer	Trident 節點 Pod 必須執行特權容器才能掛載磁碟區。	true
allowedUnsafeSysctls	Trident 不需要任何不安全的 sysctls。	none
allowedCapabilities	非特權 Trident 容器不需要比預設集更多的功能，而特權容器將被授予所有可能的功能。	空白
defaultAddCapabilities	特權容器無需新增任何功能。	空白
fsGroup	Trident 容器以 root 權限運作。	RunAsAny

標籤	說明	預設
`groups`	此 SCC 專用於 Trident、並與其使用者綁定。	空白
`readOnlyRootFilesystem`	Trident 節點 Pod 必須寫入節點檔案系統。	`false`
`requiredDropCapabilities`	Trident 節點 Pod 運作的是特權容器，無法放棄任何功能。	`none`
`runAsUser`	Trident 容器以 root 權限運作。	`RunAsAny`
`seLinuxContext`	Trident 未進行設置 `seLinuxOptions`，因為目前容器執行階段和 Kubernetes 發行版處理 SELinux 的方式有差異。	空白
`seccompProfiles`	特權容器始終以 "Unconfined" 模式運作。	空白
`supplementalGroups`	Trident 容器以 root 權限運作。	`RunAsAny`
`users`	提供了一個條目，用於將此 SCC 綁定到 Trident 命名空間中的 Trident 使用者。	n/a
`volumes`	Trident Pod 需要這些磁碟區外掛程式。	`hostPath, downwardAPI, projected, emptyDir`

版權資訊

商標資訊