



Trident 24.10 文件

Trident

NetApp
November 22, 2024

目錄

Trident 24.10 文件	1
版本資訊	2
新功能	2
較早版本的文件	14
開始使用	15
瞭解 Trident	15
Trident 快速入門	22
需求	23
安裝Trident	27
瞭解 Trident 安裝	27
使用Trident操作員安裝	31
使用tridentctl安裝	57
使用 Trident	63
準備工作節點	63
設定及管理後端	73
建立及管理儲存類別	217
資源配置與管理磁碟區	221
管理及監控 Trident	259
升級 Trident	259
使用 tridentctl 管理 Trident	265
監控 Trident	272
解除安裝Trident	275
Trident for Docker	278
部署的先決條件	278
部署 Trident	281
升級或解除安裝Trident	285
使用Volume	287
收集記錄	295
管理多個 Trident 執行個體	296
儲存組態選項	297
已知問題與限制	306
最佳實務做法與建議	308
部署	308
儲存組態	308
整合 Trident	314
資料保護與災難恢復	322
安全性	324
使用 Trident Protect 保護應用程式	331
瞭解 Trident Protect	331

安裝 Trident Protect	331
管理 Trident Protect	342
管理及保護應用程式	350
解除安裝 Trident Protect	391
知識與支援	392
常見問題集	392
疑難排解	398
支援	403
參考資料	405
Trident 連接埠	405
Trident REST API	405
命令列選項	406
Kubernetes和Trident物件	407
Pod安全標準（PSS）與安全內容限制（SCC）	417
法律聲明	421
版權	421
商標	421
專利	421
隱私權政策	421
開放原始碼	421

Trident 24.10 文件

版本資訊

新功能

版本資訊提供最新版 Trident 的新功能、增強功能和錯誤修正等相關資訊。



安裝程式壓縮檔中提供的Linux試用版 (tridentctl) 二進位檔是經過測試且受支援的版本。請注意、不會測試或支援壓縮檔中「/Extras」部分提供的「macos」二進位檔。

24.10 的新功能

增強功能

- Google Cloud NetApp Volumes 驅動程式現在通常可用於 NFS 磁碟區、並支援區域感知資源配置。
- GCP 工作負載身分識別將用作 Google Cloud NetApp Volumes 與 GKE 的雲端身分識別。
- 新增 `formatOptions` 組態參數至 ONTAP SAN 和 ONTAP SAN 經濟型驅動程式、可讓使用者指定 LUN 格式選項。
- 將 Azure NetApp Files 最小磁碟區大小減至 50 GiB。Azure 預計將於 11 月推出全新的最小尺寸。
- 新增 `denyNewVolumePools` 組態參數、將 ONTAP NAS 經濟型和 ONTAP SAN 經濟型驅動程式限制在現有的 FlexVol 集區。
- 新增偵測功能、可在所有 ONTAP 驅動程式中新增、移除或重新命名 SVM 的集合體。
- 新增 18MiB 額外負荷至 LUKS LUN、以確保報告的 PVC 大小可供使用。
- 改善的 ONTAP SAN 和 ONTAP SAN 經濟型節點階段和非階段錯誤處理、可在發生故障階段後進行取消階段移除裝置。
- 新增自訂角色產生器、可讓客戶在 ONTAP 中為 Trident 建立極簡角色。
- 新增其他記錄以進行疑難排解 `lsscsi` ("問題 #792")。

Kubernetes

- 為 Kubernetes 原生工作流程新增 Trident 功能：
 - 資料保護
 - 資料移轉
 - 災難恢復
 - 應用程式行動力

["深入瞭解 Trident Protect"](#)。
- 新增旗標 `--k8s_api_qps` 至安裝程式、以設定 Trident 用來與 Kubernetes API 伺服器通訊的 QPS 值。
- 新增 `--node-prep` 旗標至安裝程式、以自動管理 Kubernetes 叢集節點上的儲存傳輸協定相依性。已測試並驗證與 Amazon Linux 2023 iSCSI 儲存傳輸協定的相容性
- 在非正常節點關機案例中、新增對強制分離 ONTAP NAS 經濟型磁碟區的支援。

- 使用後端選項時、全新的 ONTAP NAS 經濟型 NFS 磁碟區將使用每 qtree 匯出原則 `autoExportPolicy`。qtree 只會在發佈時對應至節點限制的匯出原則、以改善存取控制和安全性。當 Trident 從所有節點取消發佈磁碟區時、現有的 qtree 將切換至新的匯出原則模型、而不會影響作用中的工作負載。
- 增加了對 Kubernetes 1.31 的支援。

實驗性增強功能

- 在 ONTAP SAN 驅動程式上新增光纖通道支援的技術預覽。請參閱 ["Fibre Channel 支援"](#)。

修正

- * Kubernetes* :
 - 固定的 Rancher 接入 Webhook 可防止安裝 Trident Helm (["問題 #839"](#))。
 - 船舵圖表值中的固定關聯鍵 (["問題 #898"](#))。
 - 固定 `TRIDENTControllerPluginNodeSeler/tridentNodePluginNodeSelector` 無法與 `"true"` 值一起使用 (["問題 #899"](#))。
 - 已刪除在複製期間建立的暫時性快照 (["問題 #901"](#))。
- 新增 Windows Server 2019 支援。
- 修正了 "Go mod 整齊的 Trident repo" (["問題 #767"](#))。

棄用

- * Kubernetes : *
 - 已將支援的 Kubernetes 最小值更新為 1.25。
 - 移除 Pod 安全性原則的支援。

產品重新品牌化

從 24.10 版本開始、Astra Trident 將改為 Trident (NetApp Trident) 品牌。這項品牌重塑不會影響 Trident 的任何功能，支援的平台或互通性。

24.06 的變更

增強功能

- **重要** : 此 `limitVolumeSize` 參數現在限制了 ONTAP 經濟驅動程式中的 qtree /LUN 大小。使用新 `limitVolumePoolSize` 參數來控制這些驅動程式中的 FlexVol 大小。 (["問題 #341"](#))。
- 增加了 iSCSI 自我修復功能，可在使用過時的 igroup 時，以確切的 LUN ID 啟動 SCSI 掃描 (["問題 #883"](#))。
- 新增對 Volume Clone 的支援、即使後端處於暫停模式、也能調整作業大小。
- 新增功能、可讓使用者為 Trident 控制器設定記錄檔設定、以傳播至 Trident 節點 Pod。
- 在 Trident 中新增支援、預設使用 REST、而非 ONTAP 9.15.1 版及更新版本的 ZAPI。
- 新增對 ONTAP 儲存設備後端上的自訂磁碟區名稱和中繼資料的支援、以供新的持續磁碟區使用。

- 增強 `azure-netapp-files` (`anf`) 驅動程式、可在 NFS 裝載選項設定為使用 NFS 版本 4.x 時、依預設自動啟用快照目錄
- 新增對 NFS 磁碟區的 Bottlerocket 支援。
- 新增 Google Cloud NetApp Volumes 的技術預覽支援。

Kubernetes

- 增加了對 Kubernetes 1.30 的支援。
- Trident 演示集可在啓動時清理殭屍掛載和剩餘追蹤檔案 ("問題 #883")。
- 新增 PVC 註解 `trident.netapp.io/luksEncryption` 以動態匯入 LUKS Volume ("問題 #849")。
- 新增拓撲感知功能至 `anf` 驅動程式。
- 新增對 Windows Server 2022 節點的支援。

修正

- 修正因過時交易而導致的 Trident 安裝失敗。
- 修正 `tridentctl` 以忽略 Kubernetes () 的警告訊息 "問題 #892"。
- 已將 Trident 控制器優先級更改 `SecurityContextConstraint` 為 `\0` ("問題 #887")。
- ONTAP 驅動程式現在接受低於 20MiB 的磁碟區大小 ("問題 #885")。
- 固定式 Trident 可防止 ONTAP SAN 驅動程式在調整大小作業期間縮小 FlexVols。
- 修正 NFS v4.1 的磁碟區匯入失敗。

棄用

- 移除對 EOL Windows Server 2019 的支援。

24.02 的變更

增強功能

- 新增對 Cloud Identity 的支援。
 - `Anf` 的 AKS - Azure 工作負載身分識別將用作雲端身分識別。
 - 具有 FSxN 的 EKS - AWS IAM 角色將用作雲端身分識別。
- 新增支援、可從 EKS 主控台將 Trident 安裝為 EKS 叢集的附加元件。
- 新增設定及停用 iSCSI 自我修復的功能 ("問題 #864")。
- 為 ONTAP 驅動程式新增了 FSX 特性設定、以啟用與 AWS IAM 和 SecretsManager 的整合、並讓 Trident 能夠刪除具有備份功能的 FSX 磁碟區 ("問題 #453")。

Kubernetes

- 增加了對 Kubernetes 1.29 的支援。

修正

- 當未啟用 ACP 時、會出現固定的 ACP 警告訊息 ("問題 #866") 。
- 當複本與快照相關聯時、在 ONTAP 驅動程式的快照刪除期間執行複本分割前、新增了 10 秒延遲。

棄用

- 已從多平台映像清單移除 TOATteStation 內部架構。

23.10 的變更

修正

- 如果新要求的大小小於 ONTAP NAS 和 ONTAP NAS 的總磁碟區大小、則為固定磁碟區擴充 ("問題 #834") 。
- 固定磁碟區大小、可在匯入 ONTAP NAS 和 ONTAP NAS 時僅顯示磁碟區的可用大小 (.."問題 722") 。
- ONTAP NAS 經濟的固定 FlexVol 名稱轉換。
- 修正重新開機時 Windows 節點上的 Trident 初始化問題。

增強功能

Kubernetes

增加了對 Kubernetes 1.28 的支援。

Trident

- 新增支援搭配 azure-NetApp-Files 儲存驅動程式使用 Azure 託管身分識別 (AMI) 。
- 增加了 ONTAP SAN 驅動程式對 NVMe over TCP 的支援。
- 新增功能、可在使用者將後端設定為暫停狀態時暫停磁碟區的資源配置 ("第 5558 期") 。

23.07.1 的變更

- Kubernetes : * 修正刪除程式集的問題、以支援零停機升級 ("問題 #740") 。

2007 年 23 月 23 日的變更

修正

Kubernetes

- 修正 Trident 升級、以忽略卡在終止狀態 ("問題 #740") 。
- 新增公差至「暫態 - 三叉 - 版本 - pod」定義 ("問題 #795") 。

Trident

- 修正 ONTAP ZAPI 要求、確保在節點暫存作業期間取得 LUN 屬性以識別和修正軌跡 iSCSI 裝置時、會查詢 LUN 序號。

- 已修正儲存驅動程式碼 ("問題 #816") 。
- 使用 ONTAP 驅動程式搭配 `use-rest = true` 時、可調整固定配額大小。
- 在 ONTAP SAN 經濟環境中建立固定 LUN 複製。
- 從還原發佈資訊欄位 `rawDevicePath` 至 `devicePath`；新增邏輯以填入及恢復 (在某些情況下) `devicePath` 欄位。

增強功能

Kubernetes

- 新增匯入預先配置快照的支援。
- 最小化部署和取消 Linux 權限設定 ("問題 #817") 。

Trident

- 不再報告「線上」磁碟區和快照的狀態欄位。
- 如果 ONTAP 後端離線 ("問題 #801"、"#543") 。
- LUN 序號一律會在 ControllerVolume Publish 工作流程中擷取及發佈。
- 新增其他邏輯來驗證 iSCSI 多重路徑裝置序號和大小。
- iSCSI 磁碟區的額外驗證、確保未分段正確的多重路徑裝置。

實驗性增強

新增 ONTAP SAN 驅動程式的 NVMe over TCP 技術預覽支援。

文件

許多組織和格式化的改善都已完成。

棄用

Kubernetes

- 移除對 v1beta1 快照的支援。
- 移除對 CSI 前磁碟區和儲存類別的支援。
- 已將支援的 Kubernetes 最小值更新為 1.22 。

23.04 年的變更



僅當 Kubernetes 版本啟用非正常節點關機功能閘道時、才支援 ONTAP - SAN* 磁碟區的強制磁碟區分離。必須在安裝時使用啟用強制分離 `--enable-force-detach` Trident 安裝程式旗標。

修正

- 固定 Trident 運算子在 SPEC 中指定安裝時使用 IPv6 localhost 。

- 固定的 Trident 運算子叢集角色權限、可與套件權限 ("問題#799") 。
- 已解決在rwx模式下、在多個節點上附加原始區塊Volume的問題。
- 針對FlexGroup SMB Volume提供固定的實體複製支援和Volume匯入。
- 修正 Trident 控制器無法立即關機的問題 ("問題 #811.") 。
- 新增修正程式、列出與指定 LUN 相關的所有 igroup 名稱、並以 ontap - san 驅動程式進行佈建。
- 新增修正程式、允許外部程序執行至完成。
- 修正 s390 架構的編譯錯誤 ("問題 #537") 。
- 修正磁碟區裝載作業期間的記錄層級不正確 ("問題 781") 。
- 修正潛在類型聲明錯誤 ("問題 #802") 。

增強功能

- Kubernetes :
 - 增加了對 Kubernetes 1.27 的支援。
 - 新增匯入 LUKS Volume 的支援。
 - 新增支援 ReadWriteOncePod PVC 存取模式。
 - 新增在非正常節點關機案例中強制卸除 ONTAP SAN* 磁碟區的支援。
 - 所有 ONTAP SAN * 磁碟區現在都會使用每個節點的 igroup 。LUN 只會對應到 igroup 、而會主動發佈到這些節點、以改善我們的安全狀態。當 Trident 判斷在不影響作用中工作負載的情況下、現有磁碟區將會切換至新的 igroup 配置 ("問題 758") 。
 - 透過清理 ONTAP SAN* 後端未使用的 Trident 管理的 igroup 、改善 Trident 的安全性。
- 將 Amazon FSX 對 SMB Volume 的支援新增至 ONTAP NAS 經濟型和 ONTAP NAS Flexgroup 儲存驅動程式。
- 新增了 ONTAP NAS 、 ONTAP NAS 經濟型和 ONTAP NAS Flexgroup 儲存驅動程式的 SMB 共享支援。
- 新增對 arm64 節點的支援 ("問題 #732") 。
- 透過先停用 API 伺服器來改善 Trident 關機程序 ("問題 #811.") 。
- 新增 Windows 和 arm64 主機的跨平台建置支援至 Makefile ；請參閱 build .md 。

棄用

Kubernetes: 設定 ONTAP - SAN 和 ONTAP - SAN 經濟型驅動程式時、將不再建立後端範圍的 igroup ("問題 758") 。

23.01.1 的變更

修正

- 固定Trident運算子在SPEC中指定安裝時使用IPv6 localhost。
- 固定的Trident運算子叢集角色權限、可與套件組合權限同步 "問題#799"。
- 新增修正程式、允許外部程序執行至完成。

- 已解決在rwx模式下、在多個節點上附加原始區塊Volume的問題。
- 針對FlexGroup SMB Volume提供固定的實體複製支援和Volume匯入。

23.01年的變更



Kubernetes 1.27 現在支援 Trident 。請先升級Trident、再升級Kubernetes 。

修正

- Kubernetes：新增選項以排除建立Pod安全性原則、以修正透過Helm ("問題#783、#794") 。

增強功能

Kubernetes

- 新增對Kubernetes 1.26的支援。
- 改善整體Trident RBAC資源使用率 ("問題#757") 。
- 新增自動化功能、可偵測並修正主機節點上的中斷或過時iSCSI工作階段。
- 新增對擴充LUKS加密磁碟區的支援。
- Kubernetes：新增了對LUKS加密磁碟區的認證旋轉支援。

Trident

- 新增支援SMB Volume搭配Amazon FSX ONTAP for Sfor Sfor ONTAP - NAS儲存驅動程式。
- 新增使用SMB磁碟區時對NTFS權限的支援。
- 新增對採用CVS服務層級之GCP磁碟區的儲存資源池支援。
- 新增對使用ONTAP-NAS-Flexgroup儲存驅動程式建立FlexGroups時、FlexGroupAggregateList的選用使用支援。
- 在管理多個FlexVols時、為ONTAP-NAS經濟型儲存驅動程式提升效能。
- 已啟用所有ONTAP 的支援不支援NAS儲存驅動程式的資料LIF更新。
- 更新Trident部署和示範設定命名慣例、以反映主機節點作業系統。

棄用

- Kubernetes：將支援的Kubernetes最低更新為1.21。
- 在設定時、不應再指定資料生命期 `ontap-san` 或 `ontap-san-economy` 驅動程式：

22.10的變更

- 升級至 Trident 22.10.* 之前、您必須先閱讀下列重要資訊

Trident 22.10 的相關資訊

- Kubernetes 1.25 現在支援 Trident 。升級至 Kubernetes 1.25 之前、您必須將 Trident 升級至 22.10 。
- Trident 現在嚴格強制執行 SAN 環境中的多重路徑組態、建議在 multipath.conf 檔案中使用的值為 `find_multipaths: no` 。



使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

修正

- 已修正ONTAP 特定於使用建立的靜止後端的問題 `credentials` 在22.07.0升級期間、現場無法上線 ("問題#759") 。
- 修正導致Docker Volume外掛程式無法在某些環境中啟動的問題 ("問題#548" 和 "問題#760") 。
- 修正ONTAP 了特定於SAN後端的SLE問題、以確保僅發佈屬於報告節點的資料生命期子集。
- 修正連接磁碟區時發生不必要的iSCSI LUN掃描的效能問題。
- 移除 Trident iSCSI 工作流程中的精細重試、以快速失敗並縮短外部重試時間間隔。
- 修正當對應的多重路徑裝置已排清時、在排清iSCSI裝置時傳回錯誤的問題。

增強功能

- Kubernetes :
 - 增加了對 Kubernetes 1.25 的支援。升級至 Kubernetes 1.25 之前、您必須將 Trident 升級至 22.10 。
 - 針對Trident部署和示範集新增了另一個ServiceAccount、ClusterRole和ClusterRoleBinding功能、以允許未來的權限增強功能。
 - 新增支援 "跨命名空間磁碟區共用" 。
- 所有Trident `ontap-*` 儲存驅動程式現在可搭配ONTAP 使用靜態API 。
- 新增運算子yml (`bundle_post_1_25.yml`) 沒有 `PodSecurityPolicy` 支援Kubernetes 1.25 。
- 新增 "支援LUKS加密磁碟區" 適用於 `ontap-san` 和 `ontap-san-economy` 儲存驅動程式：
- 新增對Windows Server 2019節點的支援。
- 新增 "支援Windows節點上的SMB Volume" 透過 `azure-netapp-files` 儲存驅動程式：
- 目前市面上已普遍提供適用於整個過程的自動功能、例如針對不適用的驅動程式進行交換偵測。MetroCluster ONTAP

棄用

- ** Kubernetes : *將支援的Kubernetes最低更新為1.20 。
- 移除Astra Data Store (廣告) 驅動程式。
- 已移除的支援 `yes` 和 `smart` 選項 `find_multipaths` 在設定iSCSI的工作節點多重路徑時。

22.07年的變動

修正

- Kubernetes*
 - 修正使用Helm或Trident運算子設定Trident時、處理節點選取器的布林值和數字值的問題。 (["GitHub問題#700"](#))
 - 修正非CHAP路徑處理錯誤的問題、以便Kubelet在失敗時重試。 (["GitHub問題#736"](#))

增強功能

- 將k8s.gcr.io轉換為登錄.k8s.io、做為SCSI映像的預設登錄
- ONTAP-SAN磁碟區現在會使用每節點igroup、只將LUN對應至igroup、同時主動發佈至這些節點、以改善我們的安全狀態。當Trident判斷在不影響作用中工作負載的情況下、現有的磁碟區將會在適當時機切換至新的igroup方案。
- 隨附資源配額與Trident安裝、可確保在優先級類別使用量預設受限時、排定Trident示範集。
- 新增對 Azure NetApp Files 驅動程式網路功能的支援。 (["GitHub問題#717"](#))
- 新增技術預覽功能可自動MetroCluster 切換偵測ONTAP 到不完整的驅動程式。 (["GitHub問題#228"](#))

棄用

- ** Kubernetes：*將支援的Kubernetes最低更新為1.19。
- 後端組態不再允許在單一組態中使用多種驗證類型。

移除

- AWS CVS驅動程式（自22.04年起已過時）已移除。
- Kubernetes
 - 已從節點Pod移除不必要的SYS_ADMIN功能。
 - 將節點準備工作減至簡單的主機資訊和主動服務探索、以盡力確認工作節點上是否有NFS/iSCSI服務可用。

文件

新增了一個新的"[Pod安全標準](#)"（PSS）區段、詳述 Trident 在安裝時啟用的權限。

22.04年的變化

NetApp持續改善及強化其產品與服務。以下是 Trident 的一些最新功能。如需先前版本的資訊、請參閱 "[較早版本的文件](#)"。



如果您要從任何先前的Trident版本升級並使用Azure NetApp Files 更新版本、則「位置」組態參數現在是必填的單一欄位。

修正

- 改善iSCSI啟動器名稱的剖析。 (["GitHub問題#681"](#))
- 修正不允許使用csi儲存類別參數的問題。 (["GitHub問題#598"](#))
- 修復Trident CRD中的重複金鑰宣告。 (["GitHub問題#671"](#))
- 修正不正確的「csi Snapshot記錄」。 (["GitHub問題#629"](#))
- 已修正在刪除節點上解除發佈磁碟區的問題。 (["GitHub問題#691"](#))
- 新增區塊裝置上檔案系統不一致的處理方式。 (["GitHub問題#656"](#))
- 修正在安裝期間設定「imageRegistry (影像登錄)」旗標時拉出自動支援映像的問題。 (["GitHub問題#715"](#))
- 修正 Azure NetApp Files 驅動程式無法複製具有多個匯出規則的磁碟區的問題。

增強功能

- 若要連入Trident的安全端點、現在至少需要TLS 1.3。 (["GitHub問題#698"](#))
- Trident現在將HSTC標頭新增至其安全端點的回應。
- Trident現在會自動嘗試啟用Azure NetApp Files 「UNIX權限」功能。
- * Kubernetes*：Trident取消程式集現在以系統節點關鍵優先順序類別執行。 (["GitHub問題#694"](#))

移除

E系列驅動程式 (自20.07起停用) 已移除。

22.01.1中的變更

修正

- 已修正在刪除節點上解除發佈磁碟區的問題。 (["GitHub問題#691"](#))
- 存取零欄位以取得ONTAP 靜止API回應中的集合空間時、會出現固定的恐慌。

22.01.0版的變更

修正

- * Kubernetes*：*增加大型叢集的節點登錄回退重試時間。
- 已解決以下問題：azure-NetApp-Files驅動程式可能會被同名的多個資源混淆。
- 如果使用方括弧指定SAN IPv6資料生命量、現在就能正常運作。ONTAP
- 修正嘗試匯入已匯入磁碟區傳回EOF、使PVC處於擱置狀態的問題。 (["GitHub問題#489"](#))
- 解決了在 SolidFire 磁碟區上建立超過 32 個快照時、Trident 效能降低的問題。
- 在建立SSL憑證時、以SHA-256取代SHA-1。
- 固定式 Azure NetApp Files 驅動程式可允許重複的資源名稱、並將作業限制在單一位置。
- 固定式 Azure NetApp Files 驅動程式可允許重複的資源名稱、並將作業限制在單一位置。

增強功能

- Kubernetes增強功能：
 - 新增對Kubernetes 1.23的支援。
 - 透過Trident運算子或Helm安裝Trident Pod時、請新增排程選項。 ("[GitHub問題#65](#)")
- 允許GCP驅動程式中的跨區域磁碟區。 ("[GitHub問題#633](#)")
- 新增對 Azure NetApp Files Volume 的「unixPermissions」選項支援。 ("[GitHub問題#6666](#)")

棄用

Trident REST介面只能以127.0.0.1或[:1]位址接聽和使用

210.1的變更



v21.10.0版本發生問題、可在移除節點後將Trident控制器重新新增回Kubernetes叢集時、將其置於CrashLoopBackOff狀態。此問題已在版本210.1中修正 ([GitHub問題669](#))。

修正

- 修正在GCP CVS後端匯入磁碟區時可能發生的競爭狀況、導致無法匯入。
- 修正刪除節點後、將Trident控制器重新加入Kubernetes叢集 ([GitHub問題669](#)) 時、使Trident控制器進入CrashLoopBackOff狀態的問題。
- 修正未指定SVM名稱時不再探索SVM的問題 ([GitHub問題612](#))。

21.0

修正

- 修正XFS磁碟區的複本無法與來源磁碟區掛載在同一個節點上的問題 ([GitHub問題514](#))。
- 修正 Trident 關機時發生嚴重錯誤的問題 ([GitHub 問題 597](#))。
- Kubernetes相關修正：
 - 使用「ONTAP-NAS」和「ONTAP-NAS-flexgroup」驅動程式建立快照時、傳回磁碟區的已用空間作為最小重新設定大小 ([GitHub問題645](#))。
 - 修正磁碟區調整大小後記錄「無法擴充檔案系統」錯誤的問題 ([GitHub問題560](#))。
 - 已解決Pod可能陷入「終止」狀態的問題 ([GitHub問題572](#))。
 - 修正「ONTAP-san經濟」FlexVol 的情況、即快照LUN可能已滿 ([GitHub問題533](#))。
 - 修正不同映像的自訂Yaml安裝程式問題 ([GitHub問題613](#))。
 - 修正快照大小計算 ([GitHub問題611](#))。
 - 解決了所有 Trident 安裝程式都能將純 Kubernetes 識別為 OpenShift 的問題 ([GitHub 問題 639](#))。
 - 修正Trident運算子、在Kubernetes API伺服器無法連線時停止協調 ([GitHub問題599](#))。

增強功能

- 新增了對GCP-CVS Performance Volume的「unixPermissions」選項支援。
- 在GCP中新增對大規模最佳化的CVS磁碟區的支援、範圍介於600 GiB到1 TiB之間。
- Kubernetes相關增強功能：
 - 新增對Kubernetes 1.22的支援。
 - 讓Trident運算子和Helm圖表能與Kubernetes 1.22搭配使用（GitHub問題628）。
 - 將操作員映像新增至「tridentctl」映像命令（GitHub Issue 570）。

實驗性增強功能

- 在「ONTAP-san」驅動程式中新增了對Volume複寫的支援。
- 新增*技術預覽* REST支援功能、支援「ONTAP-NAA-flexgroup」、「ONTAP-SAN」和「ONTAP-NAS-P節約」驅動程式。

已知問題

已知問題可識別可能導致您無法成功使用產品的問題。

- 將已安裝 Trident 的 Kubernetes 叢集從 1.24 升級至 1.25 或更新版本時、您必須 `true` 先更新 values.yaml 以設定 `excludePodSecurityPolicy` 或新增 `--set excludePodSecurityPolicy=true` 至 `helm upgrade` 命令、才能升級叢集。
- Trident 現在 (fsType=""`對未在其 StorageClass 中指定的卷強制執行空白 `fsType) fsType。使用 Kubernetes 1.17 或更新版本時、Trident 支援為 NFS 磁碟區提供空白 fsType`資料。對於 iSCSI 磁碟區、您必須在使用安全性內容強制執行時、在 StorageClass `fsGroup 上設定 fsType。
- 在多個 Trident 執行個體之間使用後端時、每個後端組態檔案的 ONTAP 後端應具有不同的 storagePrefix`值、或在 SolidFire 後端使用不同的值 `TenantName。Trident 無法偵測其他 Trident 執行個體所建立的磁碟區。嘗試在 ONTAP 或 SolidFire 後端上建立現有的磁碟區成功、因為 Trident 將磁碟區建立視為冪等操作。如果或 `TenantName` 不同、則 `storagePrefix` 在同一個後端上建立的磁碟區可能會發生名稱衝突。
- 安裝 Trident (使用或 Trident 運算子) 並使用來 tridentctl`管理 Trident 時 `tridentctl、您應該確定 `KUBERNETES` 已設定環境變數。這是表示 Kubernetes 叢集應可處理的必要 `tridentctl` 動作。在使用多個 Kubernetes 環境時、您應該確保 `KUBERNETES` 檔案的來源正確無誤。
- 若要執行iSCSI PV的線上空間回收、工作節點上的基礎作業系統可能需要將掛載選項傳遞至磁碟區。對於需要「discard」的RHEL/RedHat CoreOS執行個體來說、這是正確的做法 "[掛載選項](#)"; 請確定您的隨附了捨棄掛載選項 "[d4b9b9554fd820f43eae492d33e41167](#)" 支援線上區塊捨棄。
- 如果每個 Kubernetes 叢集有多個 Trident 執行個體、則 Trident 無法與其他執行個體通訊、也無法探索它們所建立的其他磁碟區、如果叢集內有多個執行個體執行、就會導致非預期和不正確的行為。每個 Kubernetes 叢集應該只有一個 Trident 執行個體。
- 如果在 Trident 離線時從 Kubernetes 刪除 Trident 型物件、則 StorageClass Trident 在重新連線時、不會從其資料庫中移除對應的儲存類別。您應該使用或 REST API 刪除這些儲存類別 tridentctl。
- 如果使用者在刪除對應的 PVC 之前刪除由 Trident 提供的 PV、Trident 不會自動刪除備份磁碟區。您應該透過或 REST API 移除 Volume tridentctl。
- 除非集合體是每個資源配置要求的唯一集合體、否則無法同時配置多個支援區。ONTAP FlexGroup

- 在使用 Trident over IPv6 時、您應該在方括號內指定 `managementLIF` 和 `dataLIF` 在後端定義中。例如 `[fd20:8b1e:b258:2000:f816:3eff:feec:0]` 。



您無法在 ONTAP SAN 後端上指定 `dataLIF`。Trident 會探索所有可用的 iSCSI 生命期、並使用它們來建立多重路徑工作階段。

- 如果使用 `solidfire-san` 使用 OpenShift 4.5 的驅動程式、請確保基礎工作者節點使用 MD5 做為 CHAP 驗證演算法。元素 12.7 提供安全的 FIPS 相容 CHAP 演算法 SHA1、SHA-256 和 SHA3-256。

如需詳細資訊、請參閱

- ["Trident GitHub"](#)
- ["Trident 部落格"](#)

較早版本的文件

如果您未執行 Trident 24.10，則可根據 ["Trident 支援生命週期"](#) 取得舊版的文件。

- ["Trident 24.06"](#)
- ["Trident 24.02"](#)
- ["Trident 23.10"](#)
- ["Trident 23.07"](#)
- ["Trident 23.04"](#)
- ["Trident 23.01"](#)
- ["Trident 22.10"](#)
- ["Trident 22.07"](#)
- ["Trident 22.04"](#)
- ["Trident 22.01"](#)

開始使用

瞭解 Trident

瞭解 Trident

Trident 是 NetApp 所維護的完全支援的開放原始碼專案。其設計旨在協助您使用業界標準介面（例如 Container Storage Interface（CSI））來滿足容器化應用程式的持續需求。

什麼是 **Trident** ？

NetApp Trident 可在所有熱門的 NetApp 儲存平台、公有雲或內部部署中、使用和管理儲存資源、包括 ONTAP（AFF、FAS、Select、Cloud、Amazon FSX for NetApp ONTAP）、Element 軟體（NetApp HCI、SolidFire）、Azure NetApp Files 服務和 Cloud Volumes Service on Google Cloud。

Trident 是符合 Container Storage Interface（CSI）規範的動態儲存協調器"[Kubernetes](#)"、可與原生整合。Trident 會在叢集中的每個工作節點上、以單一控制器 Pod 加上節點 Pod 的形式執行。如 "[Trident 架構](#)" 需詳細資訊、請參閱。

Trident 也可直接整合 NetApp 儲存平台的 Docker 生態系統。NetApp Docker Volume 外掛程式（nDVP）支援從儲存平台到 Docker 主機的儲存資源配置與管理。如 "[部署 Trident for Docker](#)" 需詳細資訊、請參閱。



如果這是您第一次使用 Kubernetes、您應該熟悉 "[Kubernetes 概念與工具](#)"。

參加 Trident 試用

若要試用、請使用現成的實驗室映像、要求存取「輕鬆部署及複製容器化工作負載的持續儲存設備」"[NetApp 試用](#)"。此測試磁碟機提供一個沙箱環境、其中安裝並設定了三節點 Kubernetes 叢集和 Trident。這是熟悉 Trident 並探索其功能的好方法。

另一個選項是 "[Kubeadm 安裝指南](#)" 由 Kubernetes 提供。



請勿在正式作業環境中使用這些指示來建置 Kubernetes 叢集。請使用經銷商所提供的正式作業部署指南、以用於正式作業就緒的叢集。

Kubernetes 與 NetApp 產品整合

NetApp 儲存產品組合可與 Kubernetes 叢集的許多層面整合、提供進階的資料管理功能、強化 Kubernetes 部署的功能、功能、效能和可用度。

Amazon FSX for NetApp ONTAP 產品

"[Amazon FSX for NetApp ONTAP 產品](#)" 是一項完全託管的 AWS 服務、可讓您啟動及執行 NetApp ONTAP 儲存作業系統所支援的檔案系統。

Azure NetApp Files

"[Azure NetApp Files](#)" 是採用NetApp技術的企業級Azure檔案共享服務。您可以在Azure原生環境中執行最嚴苛的檔案型工作負載、並享有NetApp所提供的效能與豐富資料管理功能。

Cloud Volumes ONTAP

"[Cloud Volumes ONTAP](#)" 是一款純軟體的儲存應用裝置、可在ONTAP 雲端上執行功能完善的資料管理軟體。

Google Cloud NetApp Volumes

"[Google Cloud NetApp Volumes](#)" 是 Google Cloud 中的完全託管檔案儲存服務，可提供高效能的企業級檔案儲存。

Element軟體

"[元素](#)" 儲存管理員可藉由保證效能、並簡化及簡化儲存設備佔用空間、來整合工作負載。

NetApp HCI

"[NetApp HCI](#)" 將例行工作自動化、讓基礎架構管理員能夠專注於更重要的功能、進而簡化資料中心的管理與規模。

Trident可直接針對底層NetApp HCI 的資訊儲存平台、為容器化應用程式配置及管理儲存設備。

NetApp ONTAP

"[NetApp ONTAP](#)" 是 NetApp 多重傳輸協定、統一化的儲存作業系統、可為任何應用程式提供進階的資料管理功能。

支援所有Flash、混合式或全硬碟組態的系統、可提供多種不同的部署模式、包括工程設計硬體（英文版）、白箱（英文版）和僅雲端（英文版）ONTAP FAS AFF ONTAP Select Cloud Volumes ONTAP
◦ Trident 支援這些 ONTAP 部署模式。

Trident 架構

Trident 會在叢集中的每個工作節點上、以單一控制器 Pod 加上節點 Pod 的形式執行。節點 Pod 必須在任何想要掛載 Trident Volume 的主機上執行。

瞭解控制器 Pod 和節點 Pod

Trident 在 Kubernetes 叢集上部署為單一 [Trident 控制器 Pod](#) 和多個 [Trident 節點 Pod](#) 並使用標準 Kubernetes [CSI Sidecar Containers](#) 來簡化 CSI 外掛程式的部署。["Kubernetes CSI Sidecar Container"](#) 由 Kubernetes 儲存社群維護。

Kubernetes "節點選取器" 和"容忍和污染"用於限制 Pod 在特定或偏好的節點上執行。您可以在 Trident 安裝期間、為控制器和節點 Pod 設定節點選取器和公差。

- 控制器外掛程式可處理磁碟區資源配置與管理、例如快照和調整大小。
- 節點外掛程式會處理將儲存設備附加至節點的問題。

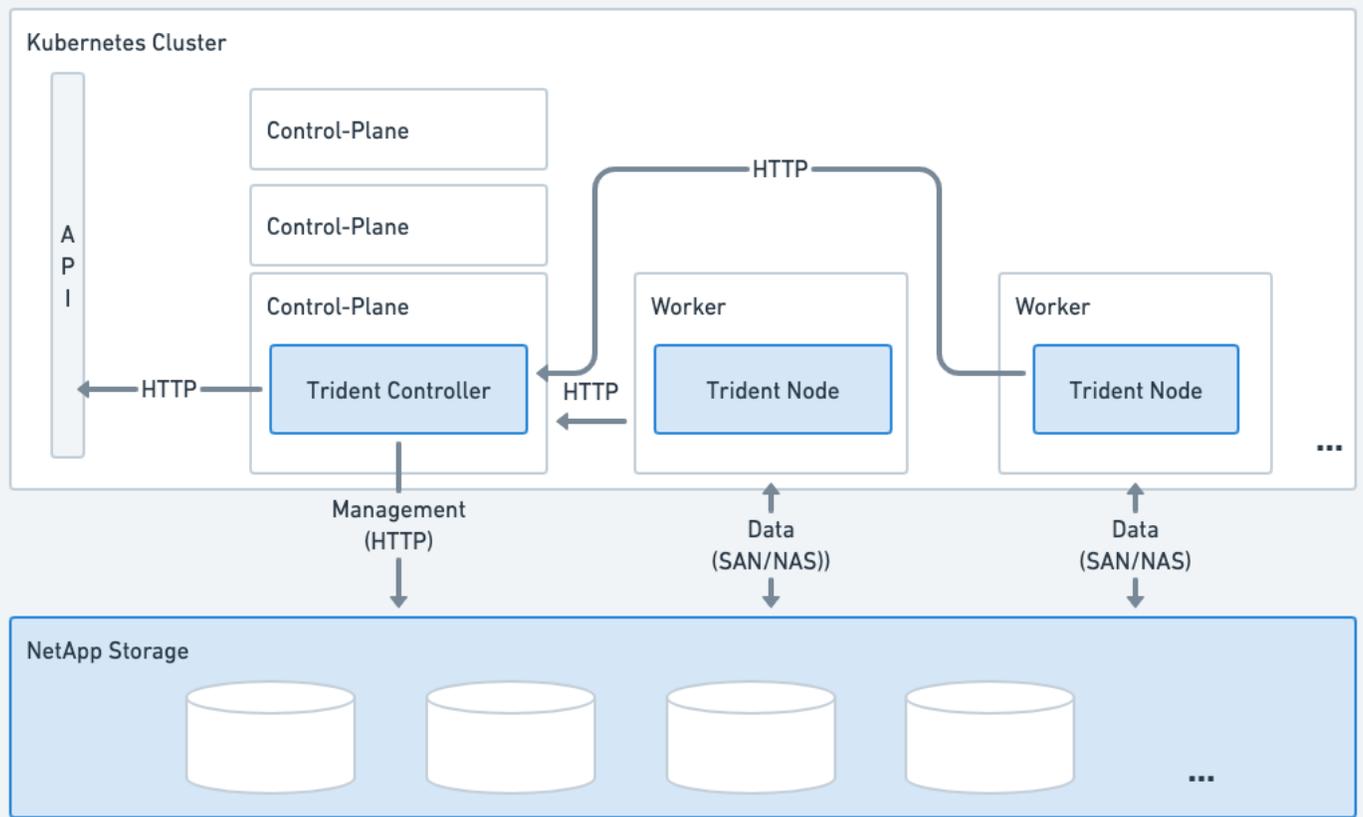


圖 1. Trident 部署在 Kubernetes 叢集上

Trident 控制器 Pod

Trident 控制器 Pod 是執行 CSI 控制器外掛程式的單一 Pod。

- 負責在 NetApp 儲存設備中佈建及管理磁碟區
- 由 Kubernetes 部署管理
- 可在控制面或工作節點上執行、視安裝參數而定。

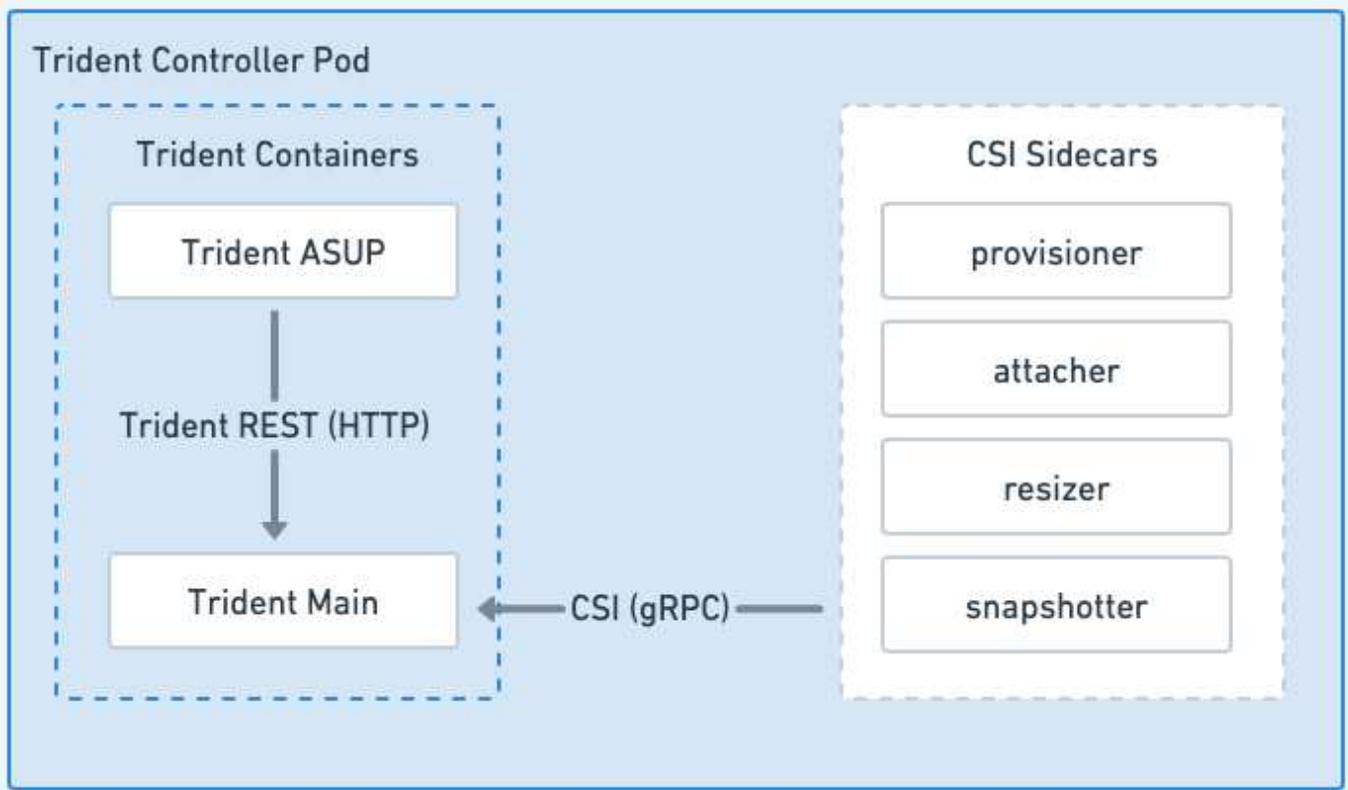


圖 2. Trident 控制器 Pod 圖表

Trident 節點 Pod

Trident Node Pod 是執行 CSI Node 外掛程式的特殊權限 Pod。

- 負責裝載和卸載主機上執行的 Pod 儲存設備
- 由 Kubernetes 示範集管理
- 必須在將裝載 NetApp 儲存設備的任何節點上執行

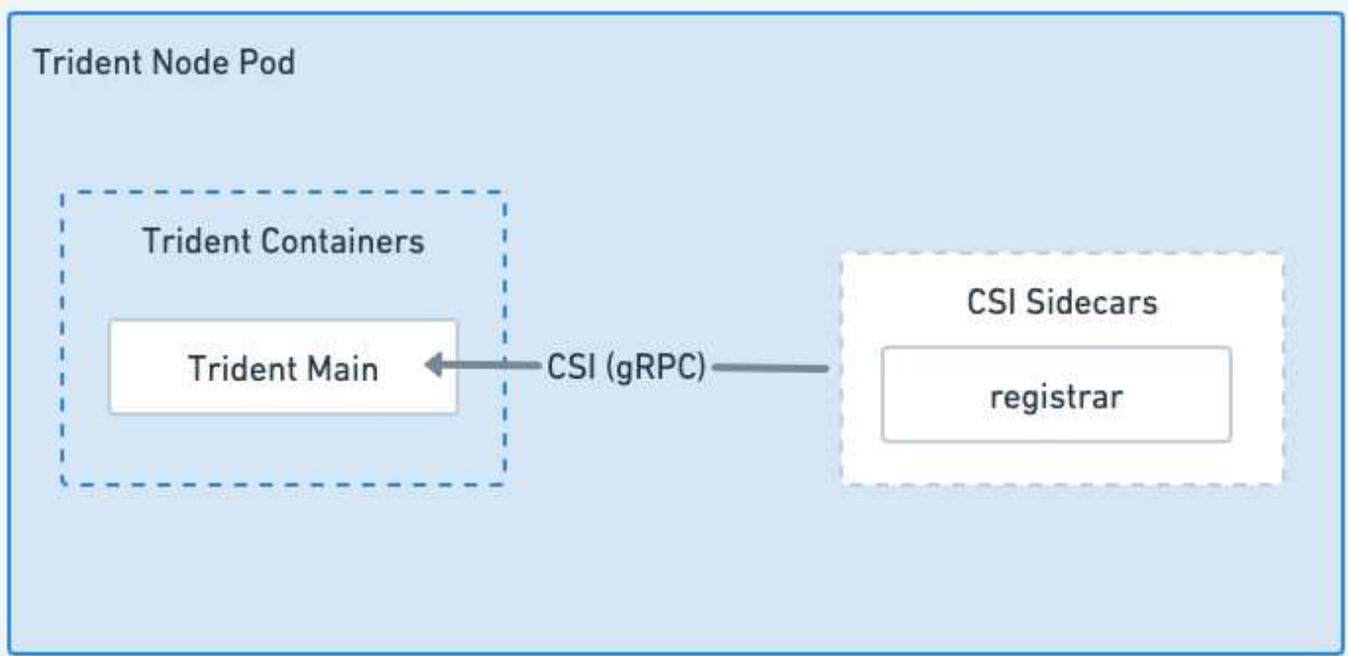


圖 3. Trident Node Pod 圖表

支援的Kubernetes叢集架構

Trident 支援下列 Kubernetes 架構：

Kubernetes叢集架構	支援	預設安裝
單一主機、運算	是的	是的
多重主機、運算	是的	是的
Master、「etcd」、運算	是的	是的
主要、基礎架構、運算	是的	是的

概念

資源配置

Trident 中的資源配置有兩個主要階段。第一階段會將儲存類別與一組適當的後端儲存資源池建立關聯、並在進行資源配置之前做好必要準備。第二階段包括磁碟區建立本身、需要從與擱置磁碟區的儲存類別相關的儲存池中選擇儲存池。

儲存類別關聯

將後端儲存集區與儲存類別建立關聯、需要同時仰賴儲存類別所要求的屬性及其 `storagePools`、`additionalStoragePools` 和 `excludeStoragePools` 清單。當您建立儲存類別時、Trident 會比較每個後端所提供的屬性和集區、以及儲存類別所要求的屬性和集區。如果儲存池的屬性和名稱符合所有要求的屬性和集區名稱、Trident 會將該儲存集區新增至該儲存類別的適當儲存集區集。此外、Trident 也會將清

單中列出的所有儲存資源池新增至該集區 `additionalStoragePools`、即使其屬性無法滿足所有或任何儲存類別的要求屬性。您應該使用此 `excludeStoragePools` 清單來覆寫及移除儲存資源池、以供儲存類別使用。每次新增後端時、Trident 都會執行類似的程序、檢查其儲存資源池是否符合現有儲存類別的要求、並移除任何標記為排除的儲存資源池。

Volume建立

然後，Trident 會使用儲存類別與儲存資源池之間的關聯來決定資源配置磁碟區的位置。當您建立磁碟區時、Trident 會先取得該磁碟區儲存類別的儲存資源池集區、如果您為該磁碟區指定傳輸協定、Trident 會移除無法提供所要求傳輸協定的儲存資源池（例如、NetApp HCI / SolidFire 後端無法提供檔案型磁碟區、而 ONTAP NAS 後端無法提供區塊型磁碟區）。Trident 會隨機排列此結果集的順序、以利平均分配磁碟區、然後逐一重複執行、然後嘗試在每個儲存池上佈建磁碟區。如果某個項目成功、則會成功傳回、並記錄程序中發生的任何故障。Trident 只有在 * 無法在 * 所有 * 上配置可用於所要求儲存類別和傳輸協定的儲存集區時、才會傳回故障 *。

Volume快照

深入瞭解 Trident 如何處理其驅動程式的磁碟區快照建立。

深入瞭解Volume Snapshot建立

- 適用於 `ontap-nas`、`ontap-san`、`gcp-cvs` 和 `azure-netapp-files` 驅動程式、每個持續Volume (PV) 都會對應FlexVol 至一個功能區。因此、磁碟區快照會建立為NetApp快照。相較於競爭的快照技術、NetApp Snapshot 快照技術可提供更高的穩定性、擴充性、可恢復性和效能。這些Snapshot複本無論在建立所需的時間、還是在儲存空間中、都能發揮極高的效率。
- 適用於 `ontap-nas-flexgroup` 驅動程式、每個持續Volume (PV) 都會對應FlexGroup 至一個功能區。因此、磁碟區快照會建立為NetApp FlexGroup 的「資訊快照」。相較於競爭的快照技術、NetApp Snapshot 快照技術可提供更高的穩定性、擴充性、可恢復性和效能。這些Snapshot複本無論在建立所需的時間、還是在儲存空間中、都能發揮極高的效率。
- 適用於 `ontap-san-economy` 驅動程式、PV對應至在共享FlexVols上建立的LUN。PV的Volume Snapshot 是透過執行相關LUN的FlexClones來達成的。ONTAP FlexClone 技術讓您幾乎可以立即建立最大資料集的複本。複本會與其父實體共用資料區塊、除了中繼資料所需的儲存空間外、不需要使用任何儲存設備。
- 對於「Poolidfire - san」驅動程式、每個PV對應至NetApp Element 在該軟體/NetApp HCI叢集上建立的LUN。Volume Snapshot以基礎LUN的元素快照來表示。這些快照是時間點複本、只佔用少量系統資源和空間。
- 使用 `ontap-nas` 和 `ontap-san` 驅動程式時、ONTAP 快照是 FlexVol 的時間點複本、會佔用 FlexVol 本身的空間。這可能會產生磁碟區中的可寫入空間量、以便在建立/排程快照時縮短時間。解決此問題的一種簡單方法、就是透過Kubernetes調整大小來擴充磁碟區。另一個選項是刪除不再需要的快照。刪除透過Kubernetes 建立的 Volume Snapshot 時、Trident 會刪除相關的 ONTAP 快照。不透過Kubernetes建立的支援快照也可以刪除。ONTAP

有了 Trident、您可以使用 Volume Snapshots 來建立新的 PV。使用FlexClone技術建立這些快照的PV、以支援ONTAP 支援的支援的支援功能和CVS後端。從快照建立 PV 時、備份磁碟區是快照父磁碟區的 FlexClone。此 `solidfire-san` 驅動程式使用元素軟體磁碟區複本、從快照建立 PV。在此、它會從元素快照建立複本。

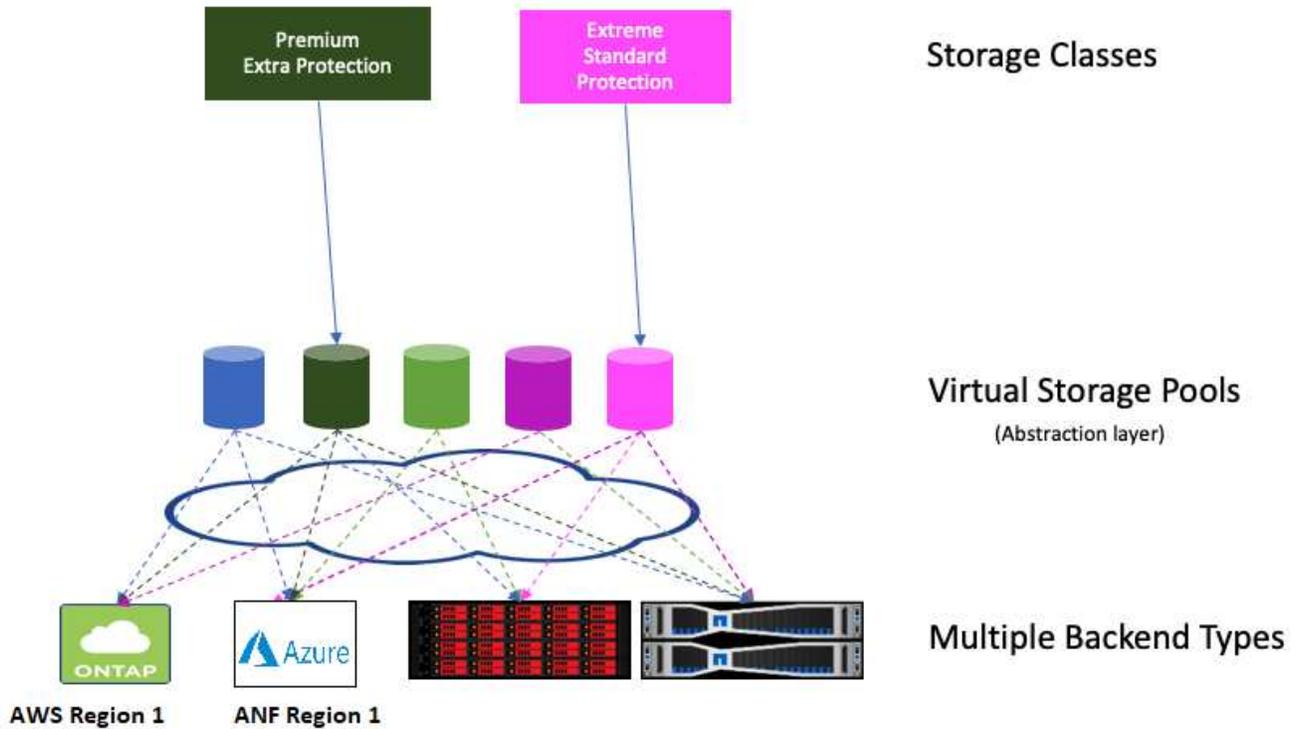
虛擬資源池

虛擬池在 Trident 儲存後端和 Kubernetes 之間提供抽象層 `StorageClasses`。這些功能可讓系統管理員以通用、不受後端限制的方式、定義各個後端的位置、效能和保護等層面、而無需 `StorageClass` 指定要使用哪種實體後端、後端集區或後端類型來符合所需的

條件。

瞭解虛擬資源池

儲存管理員可以在 JSON 或 YAML 定義檔案的任何 Trident 後端上定義虛擬集區。



在虛擬資源池清單之外指定的任何層面、都會對後端進行全域設定、並套用至所有虛擬資源池、而每個虛擬資源池則可個別指定一個或多個層面（覆寫任何後端全域層面）。



- 定義虛擬資源池時、請勿嘗試重新排列後端定義中現有虛擬資源池的順序。
- 我們建議您不要修改現有虛擬資源池的屬性。您應該定義新的虛擬資源池以進行變更。

大部分方面都是以後端特定的詞彙來指定。最重要的是、在後端驅動程式之外、不會顯示高寬比值、也無法在中進行比對 StorageClasses。而是由系統管理員為每個虛擬資源池定義一或多個標籤。每個標籤都是「金鑰：值配對」、而且標籤可能在獨特的後端之間通用。如同個別層面、標籤可依資源池指定、也可全域指定至後端。不同於具有預先定義名稱和值的各個層面、系統管理員有充分的判斷權、可視需要定義標籤金鑰和值。為了方便起見、儲存管理員可以針對每個虛擬資源池定義標籤、並依標籤將磁碟區分組。

答 StorageClass 透過參照選取元參數中的標籤來識別要使用的虛擬資源池。虛擬資源池選取器支援下列運算子：

營運者	範例	集區的標籤值必須：
'='	效能=優異	相符
!!=	效能!=極致	不相符
《in》	位置（東部、西部）	加入一組值

營運者	範例	集區的標籤值必須：
《不》	效能附註（銀、銅）	不在一組值中
<key>	保護	存在於任何值
!<key>	!保護	不存在

Volume存取群組

深入瞭解 Trident 的使用 "[Volume存取群組](#)"方式。



如果您使用的是CHAP、建議您略過本節、以簡化管理並避免以下所述的擴充限制。此外、如果您在 CSI 模式中使用 Trident、則可以忽略此部分。當 Trident 安裝為增強式 CSI 資源配置程式時、會使用 CHAP。

深入瞭解Volume存取群組

Trident 可以使用 Volume 存取群組來控制對其所配置之磁碟區的存取。如果停用 CHAP、除非您在組態中指定一或多個存取群組 ID、否則它會預期找到一個稱為的存取群 `trident` 組。

雖然 Trident 會將新磁碟區與設定的存取群組建立關聯、但不會自行建立或管理存取群組。存取群組必須先存在、才能將儲存後端新增至 Trident、而且必須包含 Kubernetes 叢集中每個節點的 iSCSI IQN、這些節點可能會裝載該後端所佈建的磁碟區。在大多數安裝中、這包括叢集中的每個工作節點。

對於具有超過64個節點的Kubernetes叢集、您應該使用多個存取群組。每個存取群組最多可包含64個IQN、每個磁碟區可屬於四個存取群組。在設定最多四個存取群組的情況下、叢集中最多256個節點的任何節點都能存取任何磁碟區。如需 Volume 存取群組的最新限制、請參閱 "[請按這裡](#)"。

如果您是從使用預設值的組態修改組態 trident 存取群組也會使用其他群組、包括的ID trident 清單中的存取群組。

Trident 快速入門

您可以在幾個步驟內安裝 Trident 並開始管理儲存資源。開始使用之前，請查看"[Trident 需求](#)"。



若為 Docker "[Trident for Docker](#)"、請參閱。



1 安裝 Trident

Trident 提供多種安裝方法和模式、針對各種環境和組織進行最佳化。

"安裝Trident"



2 準備工作節點

Kubernetes叢集中的所有工作節點都必須能夠掛載您已為Pod配置的磁碟區。

"準備工作節點"

3

建立後端

後端定義 Trident 與儲存系統之間的關係。它告訴Trident如何與該儲存系統通訊、以及Trident如何從該儲存系統配置磁碟區。

"設定後端" 適用於您的儲存系統

4

建立 Kubernetes StorageClass

Kubernetes StorageClass 物件會將 Trident 指定為資源配置程式、並可讓您建立儲存類別、以使用可自訂的屬性來資源配置磁碟區。Trident 會為指定 Trident 資源配置程式的 Kubernetes 物件建立相符的儲存類別。

"建立儲存類別"

5

配置 Volume

PersistentVolume (PV) 是叢集管理員在 Kubernetes 叢集上配置的實體儲存資源。_PersistentVolume Claim (PVC) 是存取叢集上 PersistentVolume 的要求。

建立 PersistentVolume (PV) 和 PersistentVolume Claim (PVC) 、使用設定的 Kubernetes StorageClass 來要求存取 PV 。然後、您可以將 PV 掛載至 Pod 。

"配置 Volume"

接下來呢？

您現在可以新增其他後端、管理儲存類別、管理後端、以及執行 Volume 作業。

需求

安裝 Trident 之前、您應該先檢閱這些一般系統需求。特定後端可能有其他需求。

Trident 的重要資訊

- 您必須閱讀下列有關 Trident 的重要資訊。 *

 的 Trident 相關資訊

- Kubernetes 1.31 現在支援 Trident 。升級Kubernetes之前先升級Trident 。
- Trident 嚴格強制在 SAN 環境中使用多重路徑組態、建議在 multipath.conf 檔案中使用值 `find_multipaths: no` 。

使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

支援的前端（協調器）

Trident 支援多個容器引擎和協調器、包括：

- Antos on – Prem （VMware）和 Antos on bare metal 金片 1.16
- Kubernetes 1.25 - 1.31
- OpenShift 4.10 - 4.17
- Rancher Kubernetes Engine 2 （RKE2） v1.28.5+rke2r1

這些版本支援Trident運算子：

- Antos on – Prem （VMware）和 Antos on bare metal 金片 1.16
- Kubernetes 1.25 - 1.31
- OpenShift 4.10 - 4.17
- Rancher Kubernetes Engine 2 （RKE2） v1.28.5+rke2r1

Trident 也與其他完全託管且自行管理的 Kubernetes 產品合作、包括 Google Kubernetes Engine （GKE） 、 Amazon Elastic Kubernetes Services （EKS） 、 Azure Kubernetes Service （aks） 、 Mirantis Kubernetes Engine （MKE） 和 VMware Tanzu Portfolio 。

Trident 和 ONTAP 可作為的儲存供應商"KubeVirt" 。



在將安裝了 Trident 的 Kubernetes 叢集從 1.24 升級至 1.25 或更新版本之前"升級 Helm 安裝"、請參閱。

支援的後端（儲存）

若要使用 Trident 、您需要下列一或多個支援的後端：

- Amazon FSX for NetApp ONTAP 產品
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp Volumes

- FAS/AFF/選取9.5或更新版本
- NetApp All SAN Array ASA (ESAN)
- NetApp HCI / Element軟體11或更新版本

功能需求

下表摘要說明此 Trident 版本的可用功能、以及其支援的 Kubernetes 版本。

功能	Kubernetes版本	需要功能閘道？
Trident	1.25 - 1.31	否
Volume Snapshot	1.25 - 1.31	否
來自Volume Snapshot的PVc	1.25 - 1.31	否
iSCSI PV調整大小	1.25 - 1.31	否
資訊雙向CHAP ONTAP	1.25 - 1.31	否
動態匯出原則	1.25 - 1.31	否
Trident運算子	1.25 - 1.31	否
csi拓撲	1.25 - 1.31	否

已測試的主機作業系統

雖然 Trident 並未正式支援特定作業系統、但已知下列項目可以正常運作：

- 受 OpenShift Container Platform (AMD64 和 ARM64) 支援的 RedHat CoreOS (RHCOS) 版本
- RHEL 8+ (AMD64 和 ARM64)



NVMe / TCP 需要 RHEL 9 或更新版本。

- Ubuntu 22.04 或更新版本 (AMD64 和 ARM64)
- Windows Server 2022

根據預設、Trident 會在容器中執行、因此會在任何 Linux 工作者上執行。不過、這些工作者必須能夠使用標準的 NFS 用戶端或 iSCSI 啟動器來裝載 Trident 所提供的磁碟區、視您使用的後端而定。

「tridentctl」公用程式也可在任何這些Linux版本上執行。

主機組態

Kubernetes叢集中的所有工作節點都必須能夠掛載您已為Pod配置的磁碟區。若要準備工作節點、您必須根據您選擇的驅動程式來安裝 NFS 、 iSCSI 或 NVMe 工具。

["準備工作節點"](#)

儲存系統組態

Trident 可能需要變更儲存系統、後端組態才能使用。

["設定後端"](#)

Trident 連接埠

Trident 需要存取特定連接埠才能進行通訊。

["Trident 連接埠"](#)

Container映像和對應的Kubernetes版本

對於無線安裝、下列清單是安裝 Trident 所需的容器映像參考資料。使用 `tridentctl images` 命令來驗證所需的容器映像清單。

Kubernetes 版本	Container映像
v1.25.0 、 v1.26.0 、 v1.27.0 、 v1.28.0 、 v1.29.0 、 v1.30.0 、 v1.31.0	<ul style="list-style-type: none">• Docker ◦ IO/NetApp/Trident : 24.10.0• Docker ◦ IO/NetApp/Trident 自動支援 : 24.10• registry · k8s.io/SIG-storage / csi 置備程式 : v5.1.0• 登錄 .k8s.io/SIG-storage / csi 附加程式 : v4.7.0• 登錄 .k8s.io/SIG-storage / csi 大小調整 : v1.12.0.• 登錄 .k8s.io/SIG-storage / csi 快照機 : v8.1.0• 登錄 .k8s.io/SIG-storage / csi 節點驅動程式登錄器 : v2.12.0• Docker ◦ IO/NetApp/Trident : 24.10.0 (選用)

安裝 Trident

瞭解 Trident 安裝

為了確保 Trident 能夠安裝在各種環境和組織中、NetApp 提供多種安裝選項。您可以使用 Trident 運算子（手動或使用 Helm）或搭配來安裝 Trident `tridentctl`。本主題提供重要資訊、協助您選擇正確的安裝程序。

Trident 24.06 的重要資訊

- 您必須閱讀下列有關 Trident 的重要資訊。*

的 Trident 相關資訊

- Kubernetes 1.31 現在支援 Trident。升級Kubernetes之前先升級Trident。
- Trident 嚴格強制在 SAN 環境中使用多重路徑組態、建議在 `multipath.conf` 檔案中使用值 `find_multipaths: no`。

使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

開始之前

無論安裝路徑為何、您都必須具備：

- 對執行支援版本Kubernetes及啟用功能需求的Kubernetes叢集擁有完整權限。檢閱 ["需求"](#) 以取得詳細資料。
- 存取支援的NetApp儲存系統。
- 能夠從所有Kubernetes工作節點掛載磁碟區。
- Linux主機 `kubectl`（或 `oc`（如果您使用OpenShift）已安裝並設定為管理您要使用的Kubernetes叢集。
- `KUBECONFIG` 環境變數設定為指向Kubernetes叢集組態。
- 如果您使用Kubernetes搭配Docker Enterprise、["請依照他們的步驟啟用CLI存取"](#)。



如果您尚未熟悉 ["基本概念"](#) 現在正是這麼做的好時機。

選擇您的安裝方法

選取最適合您的安裝方法。您也應該檢閱的考量事項 ["在方法之間移動"](#) 做出決定之前。

使用Trident運算子

無論是手動部署或使用 Helm 、 Trident 營運者都是簡化安裝並動態管理 Trident 資源的絕佳方式。您甚至可以[自訂您的Trident營運者部署](#)使用自訂資源（ CR ）中的屬性 TridentOrchestrator 。

使用Trident營運者的好處包括：

Trident 物件建立

Trident運算子會自動為Kubernetes版本建立下列物件。

- 營運者服務帳戶
- 叢集角色和叢集角色繫結至服務帳戶
- 專屬的PodSecurity原則（適用於Kubernetes 1.25及更早版本）
- 營運者本身

問責表

叢集範圍的 Trident 操作員可在叢集層級管理與 Trident 安裝相關的資源。這可減輕使用命名空間範圍運算子來維護叢集範圍資源時可能造成的錯誤。這對於自我修復和修補至關重要。

還原功能的功能

操作人員會監控 Trident 安裝、並主動採取措施來解決問題、例如刪除部署或意外修改部署的時間。trident-operator-`<generated-id>`系統會建立一個 Pod 、將 CR 與 Trident 安裝建立關聯 TridentOrchestrator。這可確保叢集中只有一個 Trident 執行個體、並控制其設定、確保安裝具有冪等功能。當對安裝進行變更（例如刪除部署或節點取消設定）時、操作員會分別識別並修正這些變更。

更新至現有安裝的更新功能

您可以輕鬆地與營運者一起更新現有的部署。您只需要編輯 TridentOrchestrator 以更新安裝。

例如、請考慮需要啟用 Trident 來產生偵錯記錄的案例。若要執行此操作、請將您的 TridentOrchestrator 設 `spec.debug` 為 `true`：

```
kubectl patch torc <trident-orchestrator-name> -n trident --type=merge -p '{"spec":{"debug":true}}'
```

之後 TridentOrchestrator 更新後、營運者會處理更新並修補現有安裝。這可能會觸發建立新的 Pod 、以據此修改安裝。

 的重新安裝過程

叢集範圍的 Trident 運算子可清除移除叢集範圍的資源。使用者可以完全解除安裝 Trident、並輕鬆重新安裝。

支援升級功能的不一樣之處

當叢集的 Kubernetes 版本升級為支援的版本時、操作員會自動更新現有的 Trident 安裝、並加以變更、以確保它符合 Kubernetes 版本的要求。



如果叢集升級為不受支援的版本、則操作員會阻止安裝 Trident。如果已與操作員一起安裝 Trident、則會顯示警告訊息、指出 Trident 安裝在不受支援的 Kubernetes 版本上。

使用 tridentctl

如果您現有的部署必須升級、或是想要高度自訂部署、您應該考慮。這是部署 Trident 的傳統方法。

您可以產生 Trident 資源的資訊清單。這包括 Trident 在安裝時所建立的部署、取消設定程式集、服務帳戶和叢集角色。



從 22.04 版本開始、每次安裝 Trident 時、就不會再重新產生 AES 金鑰。在此版本中、Trident 會安裝一個新的秘密物件、在整個安裝過程中都會持續存在。這表示 `tridentctl` 在 22.04 中可以解除安裝舊版 Trident、但舊版無法解除安裝 22.04 安裝。選取適當的安裝方法。

選擇安裝模式

根據組織所需的安裝模式_ (標準、離線或遠端) 來判斷您的部署程序。

標準安裝

這是安裝 Trident 最簡單的方法、適用於大多數不設有網路限制的環境。標準安裝模式使用默認註冊表來存儲所需的 Trident (registry.k8s.io) (docker.io 和 CSI (CSI) 映像。

使用標準模式時、Trident 安裝程式會：

- 透過網際網路擷取容器映像
- 建立部署或節點取消設定集、可在 Kubernetes 叢集中的所有合格節點上啟動 Trident Pod

離線安裝

在無線或安全的位置可能需要離線安裝模式。在此案例中、您可以建立單一私有、鏡射的登錄或兩個鏡射登錄、以儲存所需的Trident和csi映像。



無論您的登錄組態為何、都必須將csi映像存放在單一登錄中。

遠端安裝

以下是遠端安裝程序的高階概觀：

- 在您要部署 Trident 的遠端機器上部署適當版本的 kubectl。
- 從Kubernetes叢集複製組態檔、然後在遠端機器上設定「KUBECCONFIG」環境變數。
- 啟動「kubectl Get nodes」命令、確認您可以連線至所需的Kubernetes叢集。
- 使用標準安裝步驟、從遠端機器完成部署。

根據您的方法和模式選取程序

做出決策後、請選擇適當的程序。

方法	安裝模式
Trident運算子 (手動)	"標準安裝" "離線安裝"
Trident運算子 (Helm)	"標準安裝" "離線安裝"
tridentctl	"標準或離線安裝"

在安裝方法之間移動

您可以決定變更安裝方法。在執行此操作之前、請先考慮下列事項：

- 請務必使用相同的方法來安裝和解除安裝 Trident。如果您已與一起部署 tridentctl、則應該使用適當的

二進位版本 `tridentctl` 來解除安裝 Trident 。同樣地、如果您是與運算子一起部署、則應編輯 `TridentOrchestrator` CR 並設定 `spec.uninstall=true` 為解除安裝 Trident 。

- 如果您想要移除並改用以營運者為基礎的部署 `tridentctl` 來部署 Trident 、則應先編輯 `TridentOrchestrator` 並設定 `spec.uninstall=true` 為解除安裝 Trident 。然後刪除 `TridentOrchestrator` 和操作員部署。然後您可以使用安裝 `tridentctl` 。
- 如果您有手動的操作員型部署、而且想要使用以Helm為基礎的Trident操作員部署、您應該先手動解除安裝操作員、然後再執行Helm安裝。如此一來、Helm就能部署具有所需標籤和註釋的Trident運算子。如果您不這麼做、則Helm型Trident營運者部署將會失敗、並顯示標籤驗證錯誤和註釋驗證錯誤。如果您有 `tridentctl` 根據部署、您可以使用以Helm為基礎的部署、而不會發生問題。

其他已知組態選項

在 VMware Tanzu Portfolio 產品上安裝 Trident 時：

- 叢集必須支援特殊權限的工作負載。
- 「-kubelet-dir」旗標應設定為kubelet目錄的位置。依預設、這是「/var/vcap/data/kubelet」。

使用「-kubelet-dir」指定kubelet位置、已知適用於Trident運算子、Helm及「tridentctl」部署。

使用Trident操作員安裝

手動部署Trident運算子（標準模式）

您可以手動部署 Trident 操作員來安裝 Trident 。此程序適用於 Trident 所需的容器映像未儲存在私有登錄中的安裝。如果您有私有映像登錄，請使用["離線部署程序"](#)。

Trident 24.10 的重要資訊

- 您必須閱讀下列有關 Trident 的重要資訊。*

<的 Trident **>** 相關資訊

- Kubernetes 1.31 現在支援 Trident 。升級Kubernetes之前先升級Trident 。
- Trident 嚴格強制在 SAN 環境中使用多重路徑組態、建議在 `multipath.conf` 檔案中使用值 `find_multipaths: no` 。

使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

手動部署Trident運算子並安裝Trident

檢閱 ["安裝總覽"](#) 為了確保您符合安裝先決條件、並為您的環境選擇正確的安裝選項。

開始之前

在開始安裝之前、請先登入Linux主機、然後確認它正在管理正常運作的 "支援的Kubernetes叢集" 而且您擁有必要的權限。



使用OpenShift時、請在所有範例中使用「oc"而非「kubectll」、然後先執行「ocLogin -u system:admin」或「oc login-u kube-admin」、以*系統：admin*登入。

1. 驗證Kubernetes版本：

```
kubectl version
```

2. 驗證叢集管理員權限：

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. 確認您可以啟動使用Docker Hub映像的Pod、並透過Pod網路連線至儲存系統：

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
ping <management IP>
```

步驟1：下載Trident安裝程式套件

Trident 安裝程式套件包含部署 Trident 營運商和安裝 Trident 所需的一切。從下載並解壓縮最新版本"[GitHub的_Assets區段](#)"的Trident 安裝程式。

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

步驟2：建立 TridentOrchestrator 客戶需求日

建立 TridentOrchestrator`自訂資源定義 (CRD)。您稍後會建立 `TridentOrchestrator`自訂資源。使用中適當的 CRD YAML 版本 `deploy/crds`來建立 `TridentOrchestrator CRD`。

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

步驟3：部署Trident運算子

Trident 安裝程式提供套件檔案、可用於安裝操作員及建立相關物件。套件檔案是部署操作員及使用預設組態安裝 Trident 的簡單方法。

- 對於運行 Kubernetes 1.24 的羣集，請使用 `bundle_pre_1_25.yaml`。
- 對於執行 Kubernetes 1.25 或更新版本的叢集、請使用 `bundle_post_1_25.yaml`。

開始之前

- 根據預設、Trident 安裝程式會在中部署運算子 `trident` 命名空間。如果是 `trident` 命名空間不存在、請使用以下方式建立：

```
kubectl apply -f deploy/namespace.yaml
```

- 可在非的命名空間中部署運算子 `trident` 命名空間、更新 `serviceaccount.yaml`、`clusterrolebinding.yaml` 和 `operator.yaml` 並使用產生套裝組合檔案 `kustomization.yaml`。
 - a. 建立 `kustomization.yaml` 使用下列命令、其中包含 `<bundle.yaml>` `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 以 Kubernetes 版本為基礎。

```
cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
```

- b. 使用以下命令編譯套件（其中的 `<bundle.yaml>` 是） `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 以 Kubernetes 版本為基礎。

```
kubectl kustomize deploy/ > deploy/<bundle.yaml>
```

步驟

1. 建立資源並部署營運者：

```
kubectl create -f deploy/<bundle.yaml>
```

2. 確認已建立運算子、部署和複本集。

```
kubectl get all -n <operator-namespace>
```



Kubernetes叢集中只應有*一個運算子執行個體*。請勿建立Trident營運者的多個部署。

步驟4：建立 `TridentOrchestrator` 並安裝 `Trident`

您現在可以建立 `TridentOrchestrator` 並安裝 `Trident`。您也可以選擇"自訂您的Trident安裝"使用規格中的屬性 `TridentOrchestrator`。

```

kubect1 create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubect1 describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Debug:       true
  Namespace:   trident
  nodePrep:
    - iscsi
Status:
  Current Installation Params:
    IPv6:                false
    Autosupport Hostname:
    Autosupport Image:   netapp/trident-autosupport:24.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:               true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:          30
    Kubelet Dir:         /var/lib/kubelet
    Log Format:           text
    Silence Autosupport: false
    Trident Image:       netapp/trident:24.10.0
  Message:              Trident installed Namespace:
trident
  Status:                Installed
  Version:               v24.10.0
Events:
  Type Reason Age From Message ---- -
  Installing 74s trident-operator.netapp.io Installing Trident Normal
  Installed 67s trident-operator.netapp.io Trident installed

```

驗證安裝

驗證安裝的方法有多種。

使用 TridentOrchestrator 狀態

狀態 `TridentOrchestrator` 指出安裝是否成功、並顯示安裝的 Trident 版本。安裝期間的狀態 `TridentOrchestrator` 變更來源 `Installing` 至 `Installed`。如果您觀察到 `Failed` 狀態、而且營運者無法自行恢復、"檢查記錄"。

狀態	說明
安裝	操作人員正在使用此 CR 安裝 Trident <code>TridentOrchestrator</code> 。
已安裝	Trident 已成功安裝。
正在解除安裝	操作員正在解除安裝 Trident、因為 <code>spec.uninstall=true</code> 。
已解除安裝	Trident 已解除安裝。
失敗	操作員無法安裝、修補、更新或解除安裝 Trident；操作員將自動嘗試從此狀態恢復。如果此狀態持續存在、您將需要疑難排解。
正在更新	營運者正在更新現有的安裝。
錯誤	不使用「 <code>TridentOrchestrator</code> 」。另一個已經存在。

使用 Pod 建立狀態

您可以檢閱建立的 Pod 狀態、確認 Trident 安裝是否已完成：

```
kubectl get pods -n trident
```

NAME	READY	STATUS	RESTARTS
AGE			
trident-controller-7d466bf5c7-v4cpw 1m	6/6	Running	0
trident-node-linux-mr6zc 1m	2/2	Running	0
trident-node-linux-xrp7w 1m	2/2	Running	0
trident-node-linux-zh2jt 1m	2/2	Running	0
trident-operator-766f7b8658-ldzsv 3m	1/1	Running	0

使用 `tridentctl`

您可以使用 `tridentctl` 檢查已安裝的 Trident 版本。

```
./tridentctl -n trident version
```

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 24.10.0        | 24.10.0        |
+-----+-----+
```

手動部署Trident運算子（離線模式）

您可以手動部署 Trident 操作員來安裝 Trident 。此程序適用於 Trident 所需的容器映像儲存在私有登錄中的安裝。如果您沒有私有映像登錄，請使用"[標準部署程序](#)"。

Trident 24.10 的重要資訊

- 您必須閱讀下列有關 Trident 的重要資訊。*

<的 Trident **>** 相關資訊

- Kubernetes 1.31 現在支援 Trident 。升級Kubernetes之前先升級Trident。
- Trident 嚴格強制在 SAN 環境中使用多重路徑組態、建議在 multipath.conf 檔案中使用值 `find_multipaths: no`。

使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

手動部署Trident運算子並安裝Trident

檢閱 "[安裝總覽](#)" 為了確保您符合安裝先決條件、並為您的環境選擇正確的安裝選項。

開始之前

登入Linux主機、驗證其是否正在管理正常運作的和 "[支援的Kubernetes叢集](#)" 而且您擁有必要的權限。



使用OpenShift時、請在所有範例中使用「oc」而非「kubectl」、然後先執行「oc login -u system:admin」或「oc login -u kube-admin」、以*系統：admin*登入。

1. 驗證Kubernetes版本：

```
kubectl version
```

2. 驗證叢集管理員權限：

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. 確認您可以啟動使用Docker Hub映像的Pod、並透過Pod網路連線至儲存系統：

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
ping <management IP>
```

步驟1：下載Trident安裝程式套件

Trident 安裝程式套件包含部署 Trident 營運商和安裝 Trident 所需的一切。從下載並解壓縮最新版本["GitHub的_Assets區段"](#)的Trident 安裝程式。

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

步驟2：建立 TridentOrchestrator 客戶需求日

建立 TridentOrchestrator`自訂資源定義 (CRD)。您稍後會建立 `TridentOrchestrator`自訂資源。使用中適當的 CRD YAML 版本 `deploy/crds`建立 `TridentOrchestrator CRD：

```
kubectl create -f deploy/crds/<VERSION>.yaml
```

步驟3：更新操作員中的登錄位置

在 /deploy/operator.yaml`中，更新 `image: docker.io/netapp/trident-operator:24.10.0`以反映映像登錄的位置。您 "[Trident和csi影像](#)" 可以位於一個登錄或不同的登錄中、但所有 CSI 映像都必須位於同一個登錄中。例如：

- image: <your-registry>/trident-operator:24.10.0 如果您的映像全部位於一個登錄中。
- image: <your-registry>/netapp/trident-operator:24.10.0 如果您的 Trident 映像位於與 CSI 映像不同的登錄中。

步驟4：部署Trident運算子

Trident 安裝程式提供套件檔案、可用於安裝操作員及建立相關物件。套件檔案是部署操作員及使用預設組態安裝 Trident 的簡單方法。

- 對於運行 Kubernetes 1.24 的羣集，請使用 `bundle_pre_1_25.yaml`。
- 對於執行 Kubernetes 1.25 或更新版本的叢集、請使用 `bundle_post_1_25.yaml`。

開始之前

- 根據預設、Trident 安裝程式會在中部署運算子 `trident` 命名空間。如果是 `trident` 命名空間不存在、請使用以下方式建立：

```
kubectl apply -f deploy/namespace.yaml
```

- 可在非的命名空間中部署運算子 `trident` 命名空間、更新 `serviceaccount.yaml`、`clusterrolebinding.yaml` 和 `operator.yaml` 並使用產生套裝組合檔案 `kustomization.yaml`。
 - a. 建立 `kustomization.yaml` 使用下列命令、其中包含 `<bundle.yaml>` `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 以 Kubernetes 版本為基礎。

```
cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
```

- b. 使用以下命令編譯套件（其中的 `<bundle.yaml>` 是） `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 以 Kubernetes 版本為基礎。

```
kubectl kustomize deploy/ > deploy/<bundle.yaml>
```

步驟

1. 建立資源並部署營運者：

```
kubectl create -f deploy/<bundle.yaml>
```

2. 確認已建立運算子、部署和複本集。

```
kubectl get all -n <operator-namespace>
```



Kubernetes叢集中只應有*一個運算子執行個體*。請勿建立Trident營運者的多個部署。

步驟5：更新中的映像登錄位置 `TridentOrchestrator`

您的 "[Trident和csi影像](#)" 可以位於一個登錄或不同的登錄中、但所有的SCSI映像都必須位於同一個登錄中。更新 `deploy/crds/tridentorchestrator_cr.yaml` 根據登錄組態新增額外的位置規格。

一個登錄中的映像

```
imageRegistry: "<your-registry>"  
autosupportImage: "<your-registry>/trident-autosupport:24.10"  
tridentImage: "<your-registry>/trident:24.10.0"
```

不同登錄中的映像

```
imageRegistry: "<your-registry>"  
autosupportImage: "<your-registry>/trident-autosupport:24.10"  
tridentImage: "<your-registry>/trident:24.10.0"
```

步驟6：建立 TridentOrchestrator 並安裝Trident

您現在可以建立 TridentOrchestrator 並安裝 Trident 。或者、您也可以進一步["自訂您的Trident安裝"](#)使用規格中的屬性 `TridentOrchestrator`。下列範例顯示Trident與csi映像位於不同登錄中的安裝。

```

kubect1 create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubect1 describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Autosupport Image: <your-registry>/trident-autosupport:24.10
  Debug:             true
  Image Registry:    <your-registry>
  Namespace:        trident
  Trident Image:     <your-registry>/trident:24.10.0
Status:
  Current Installation Params:
    IPv6:           false
    Autosupport Hostname:
    Autosupport Image: <your-registry>/trident-autosupport:24.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:          true
    Http Request Timeout: 90s
    Image Pull Secrets:
    Image Registry:  <your-registry>
    k8sTimeout:     30
    Kubelet Dir:    /var/lib/kubelet
    Log Format:     text
    Probe Port:    17546
    Silence Autosupport: false
    Trident Image: <your-registry>/trident:24.10.0
  Message:          Trident installed
  Namespace:        trident
  Status:           Installed
  Version:          v24.10.0
Events:
  Type Reason Age From Message -----Normal
  Installing 74s trident-operator.netapp.io Installing Trident Normal
  Installed 67s trident-operator.netapp.io Trident installed

```

驗證安裝

驗證安裝的方法有多種。

使用 `TridentOrchestrator` 狀態

狀態 `TridentOrchestrator` 指出安裝是否成功、並顯示安裝的Trident版本。安裝期間的狀態 `TridentOrchestrator` 變更來源 `Installing` 至 `Installed`。如果您觀察到 `Failed` 狀態、而且營運者無法自行恢復、"檢查記錄"。

狀態	說明
安裝	操作人員正在使用此 CR 安裝 Trident <code>TridentOrchestrator</code> 。
已安裝	Trident 已成功安裝。
正在解除安裝	操作員正在解除安裝 Trident、因為 <code>spec.uninstall=true</code> 。
已解除安裝	Trident 已解除安裝。
失敗	操作員無法安裝、修補、更新或解除安裝 Trident；操作員將自動嘗試從此狀態恢復。如果此狀態持續存在、您將需要疑難排解。
正在更新	營運者正在更新現有的安裝。
錯誤	不使用「 <code>TridentOrchestrator</code> 」。另一個已經存在。

使用 `Pod` 建立狀態

您可以檢閱建立的 `Pod` 狀態、確認 Trident 安裝是否已完成：

```
kubectl get pods -n trident
```

NAME	READY	STATUS	RESTARTS
trident-controller-7d466bf5c7-v4cpw 1m	6/6	Running	0
trident-node-linux-mr6zc 1m	2/2	Running	0
trident-node-linux-xrp7w 1m	2/2	Running	0
trident-node-linux-zh2jt 1m	2/2	Running	0
trident-operator-766f7b8658-ldzsv 3m	1/1	Running	0

使用 `tridentctl`

您可以使用 `tridentctl` 檢查已安裝的 Trident 版本。

```
./tridentctl -n trident version
```

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 24.10.0        | 24.10.0        |
+-----+-----+
```

使用Helm部署Trident運算子（標準模式）

您可以部署 Trident 運算子、並使用 Helm 安裝 Trident 。此程序適用於 Trident 所需的容器映像未儲存在私有登錄中的安裝。如果您有私有映像登錄，請使用["離線部署程序"](#)。

Trident 24.10 的重要資訊

- 您必須閱讀下列有關 Trident 的重要資訊。 *

的 Trident 相關資訊

- Kubernetes 1.31 現在支援 Trident 。升級Kubernetes之前先升級Trident 。
- Trident 嚴格強制在 SAN 環境中使用多重路徑組態、建議在 multipath.conf 檔案中使用值 `find_multipaths: no` 。

使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

部署 Trident 操作員、並使用 Helm 安裝 Trident

使用Trident ["掌舵表"](#) 您可以部署Trident運算子、並在單一步驟中安裝Trident 。

檢閱 ["安裝總覽"](#) 為了確保您符合安裝先決條件、並為您的環境選擇正確的安裝選項。

開始之前

此外 ["部署先決條件"](#) 您的需求 ["Helm版本3"](#) 。

步驟

1. 新增Trident Helm儲存庫：

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. 使用 `helm install` 並指定部署名稱、如下例所示、其中 `100.2404.0` 是您要安裝的 Trident 版本。

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0
--create-namespace --namespace <trident-namespace>
```



如果您已經為Trident建立命名空間、則「-cree-namespace」參數不會建立其他命名空間。

您可以使用 `helm list` 若要檢閱安裝詳細資料、例如名稱、命名空間、圖表、狀態、應用程式版本、和修訂編號。

在安裝期間傳遞組態資料

安裝期間有兩種傳遞組態資料的方法：

選項	說明
<code>--values</code> (或 <code>-f</code>)	指定具有覆寫的Yaml檔案。這可以多次指定、最右邊的檔案會優先。
<code>--set</code>	在命令列上指定置換。

例如，若要變更的預設值 `debug`，請執行下列命令，其中 `100.2410.0` 是您要安裝的 Trident 版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0
--create-namespace --namespace trident --set tridentDebug=true
```

組態選項

此表格和 `values.yaml` 檔案是 Helm 圖表的一部分、提供按鍵清單及其預設值。

選項	說明	預設
<code>nodeSelector</code>	Pod 指派的節點標籤	
<code>podAnnotations</code>	Pod 註釋	
<code>deploymentAnnotations</code>	部署註釋	
<code>tolerations</code>	Pod 指派的容錯功能	

選項	說明	預設
affinity	Pod 指派的關聯性	<pre> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDur ingExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">  <p>請勿移除 values.yaml 檔案的預設關聯性。當您想要提供自訂關聯性時、請擴充預設關聯性。</p> </div>
tridentContr ollerPluginN odeSelector	用於 Pod 的其他節點選取器。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
tridentContr ollerPluginT olerations	覆寫 Pod 的 Kubernetes 公差。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
tridentNodeP luginNodeSel ector	用於 Pod 的其他節點選取器。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
tridentNodeP luginTolerat ions	覆寫 Pod 的 Kubernetes 公差。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
「影像登錄」	識別、trident`和其他影像的登錄 `trident-operator。保留空白以接受預設值。重要事項：在私有儲存庫中安裝 Trident 時、如果您使用 imageRegistry` 交換器來指定儲存庫位置、請勿在儲存庫路徑中使用 `/netapp/`。	""

選項	說明	預設
imagePullPolicy	設定的映像拉出原則 trident-operator。	IfNotPresent
「imagePullSecrets」	設定的影像拉出秘密 trident-operator、`trident` 和其他影像。	
《kubeletDir	允許覆寫 kubelet 內部狀態的主機位置。	"/var/lib/kubelet"
operatorLogLevel	允許 Trident 運算子的記錄層級設定為：trace、debug、info、warn、error 或 fatal。	"info"
operatorDebug	允許將 Trident 運算子的記錄層級設定為偵錯。	"真的"
operatorImage	允許完全置換的映像 trident-operator。	""
operatorImageTag	允許覆寫的標記 trident-operator 映像。	""
tridentIPv6	允許 Trident 在 IPv6 叢集中運作。	「假」
tridentK8sTimeout	覆寫大部分 Kubernetes API 作業的預設 30 秒逾時（如果非零、則以秒為單位）。	0
tridentHttpRequestTimeout	以取代 HTTP 要求的預設 90 秒逾時 0s 是超時的無限持續時間。不允許使用負值。	"90s"
tridentSilenceAutosupport	允許停用 Trident 定期 AutoSupport 報告。	「假」
tridentAutosupportImageTag	允許覆寫 Trident AutoSupport 容器的映像標記。	<version>
tridentAutosupportProxy	可讓 Trident AutoSupport Container 透過 HTTP Proxy 撥打電話回家。	""
tridentLogFormat	設定 Trident 記錄格式(text 或 json)。	"text"
tridentDisableAuditLog	停用 Trident 稽核記錄程式。	"真的"
tridentLogLevel	允許將 Trident 的日誌級別設置為：trace、debug、info、warn error 或 fatal。	"info"
tridentDebug	允許將 Trident 的記錄層級設定為 debug。	「假」
tridentLogWorkflows	允許啟用特定的 Trident 工作流程、以進行追蹤記錄或記錄抑制。	""

選項	說明	預設
tridentLogLayers	允許啟用特定 Trident 層以進行追蹤記錄或記錄抑制。	""
「TridentImage」	允許完全置換 Trident 的映像。	""
tridentImageTag	可覆寫 Trident 的映像標記。	""
tridentProbePort	允許覆寫 Kubernetes 活性 / 整備性探查所使用的預設連接埠。	""
windows	可在 Windows 工作節點上安裝 Trident。	「假」
enableForceDetach	允許啟用強制分離功能。	「假」
excludePodSecurityPolicy	不建立營運商 Pod 安全性原則。	「假」
cloudProvider	設定為 "Azure" 在 AKS 叢集上使用託管身分識別或雲端身分識別時。在 EKS 叢集上使用雲端身分識別時、請設定為「AWS」。	""
cloudIdentity	在 AKS 叢集上使用雲端身分識別時、請設定為工作負載身分識別（「azure.Workload .idental/client-id : XXXXXXXX-xxxx-xxxx-xxxx-xxxx-xxxx-xxxxxxx」）。在 EKS 叢集上使用雲端身分識別時、請設定為 AWS IAM 角色（「eks.amazonaws.com/role-arn:arn:AWS:iam::123456 :角色 Trident 角色」）。	""
iscsiSelfHealingInterval	啟動 iSCSI 自我修復的時間間隔。	5m0s
iscsiSelfHealingWaitTime	iSCSI 自我修復透過執行登出和後續登入來嘗試解決過時工作階段的持續時間。	7m0s
nodePrep	可讓 Trident 準備 Kubernetes 叢集的節點、以使用指定的資料儲存傳輸協定來管理磁碟區。* 目前 `iscsi` 是唯一支援的值。*	

瞭解控制器 Pod 和節點 Pod

Trident 以單一控制器 Pod 的形式執行、並在叢集中的每個工作節點上執行節點 Pod。節點 Pod 必須在任何想要掛載 Trident Volume 的主機上執行。

Kubernetes "節點選取器" 和 "容忍和污染" 用於限制 Pod 在特定或偏好的節點上執行。使用「ControllerPlugin」和 NodePlugin，您可以指定限制和置換。

- 控制器外掛程式可處理磁碟區資源配置與管理、例如快照和調整大小。
- 節點外掛程式會處理將儲存設備附加至節點的問題。

使用Helm部署Trident運算子（離線模式）

您可以部署 Trident 運算子、並使用 Helm 安裝 Trident 。此程序適用於 Trident 所需的容器映像儲存在私有登錄中的安裝。如果您沒有私有映像登錄，請使用["標準部署程序"](#)。

Trident 24.10 的重要資訊

- 您必須閱讀下列有關 Trident 的重要資訊。*

 的 Trident 相關資訊

- Kubernetes 1.31 現在支援 Trident 。升級Kubernetes之前先升級Trident 。
 - Trident 嚴格強制在 SAN 環境中使用多重路徑組態、建議在 multipath.conf 檔案中使用值 `find_multipaths: no` 。
- 使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

部署 Trident 操作員、並使用 Helm 安裝 Trident

使用Trident ["掌舵表"](#) 您可以部署Trident運算子、並在單一步驟中安裝Trident 。

檢閱 ["安裝總覽"](#) 為了確保您符合安裝先決條件、並為您的環境選擇正確的安裝選項。

開始之前

此外 ["部署先決條件"](#) 您的需求 ["Helm版本3"](#) 。



在私有儲存庫中安裝 Trident 時、如果您使用 imageRegistry` 交換器來指定儲存庫位置、請勿在儲存庫路徑中使用 ``/netapp/` 。

步驟

1. 新增Trident Helm儲存庫：

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. 使用 `helm install` 並指定部署和映像登錄位置的名稱。您 ["Trident和csi影像"](#) 可以位於一個登錄或不同的登錄中、但所有 CSI 映像都必須位於同一個登錄中。在範例中 100.2410.0、是您要安裝的 Trident 版本。

一個登錄中的映像

```
helm install <name> netapp-trident/trident-operator --version
100.2410.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

不同登錄中的映像

```
helm install <name> netapp-trident/trident-operator --version
100.2410.0 --set imageRegistry=<your-registry> --set
operatorImage=<your-registry>/trident-operator:24.10.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:24.06
--set tridentImage=<your-registry>/trident:24.10.0 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```



如果您已經為Trident建立命名空間、則「-cree-namespace」參數不會建立其他命名空間。

您可以使用 `helm list` 若要檢閱安裝詳細資料、例如名稱、命名空間、圖表、狀態、應用程式版本、和修訂編號。

在安裝期間傳遞組態資料

安裝期間有兩種傳遞組態資料的方法：

選項	說明
<code>--values</code> (或 <code>-f</code>)	指定具有覆寫的Yaml檔案。這可以多次指定、最右邊的檔案會優先。
<code>--set</code>	在命令列上指定置換。

例如，若要變更的預設值 `debug`，請執行下列命令，其中 `100.2410.0` 是您要安裝的 Trident 版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0
--create-namespace --namespace trident --set tridentDebug=true
```

若要新增 `nodePrep` 值、請執行下列命令：

```
helm install <name> netapp-trident/trident-operator --version 100.2406.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

組態選項

此表格和 `values.yaml` 檔案是 Helm 圖表的一部分、提供按鍵清單及其預設值。



請勿移除 `values.yaml` 檔案的預設關聯性。當您想要提供自訂關聯性時、請擴充預設關聯性。

選項	說明	預設
<code>nodeSelector</code>	Pod 指派的節點標籤	
<code>podAnnotations</code>	Pod 註釋	
<code>deploymentAnnotations</code>	部署註釋	
<code>tolerations</code>	Pod 指派的容錯功能	
<code>affinity</code>	Pod 指派的關聯性	<pre>affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux</pre> <p> 請勿移除 <code>values.yaml</code> 檔案的預設關聯性。當您想要提供自訂關聯性時、請擴充預設關聯性。</p>

選項	說明	預設
tridentControllerPluginNodeSelector	用於 Pod 的其他節點選取器。請參閱 "瞭解控制器 Pod 和節點 Pod" 以取得詳細資料。	
tridentControllerPluginTolerations	覆寫 Pod 的 Kubernetes 公差。請參閱 "瞭解控制器 Pod 和節點 Pod" 以取得詳細資料。	
tridentNodePluginNodeSelector	用於 Pod 的其他節點選取器。請參閱 "瞭解控制器 Pod 和節點 Pod" 以取得詳細資料。	
tridentNodePluginTolerations	覆寫 Pod 的 Kubernetes 公差。請參閱 "瞭解控制器 Pod 和節點 Pod" 以取得詳細資料。	
「影像登錄」	識別、trident`和其他影像的登錄 `trident-operator。保留空白以接受預設值。重要事項：在私有儲存庫中安裝 Trident 時、如果您使用 imageRegistry`交換器來指定儲存庫位置、請勿在儲存庫路徑中使用 `/netapp/`。	"
imagePullPolicy	設定的映像拉出原則 trident-operator。	IfNotPresent
「imagePullSecrets」	設定的影像拉出秘密 trident-operator、`trident`和其他影像。	
《kubeletDir	允許覆寫 kubelet 內部狀態的主機位置。	"/var/lib/kubelet"
operatorLogLevel	允許 Trident 運算子的記錄層級設定為：trace、debug、info、warn、error`或 `fatal。	"info"
operatorDebug	允許將 Trident 運算子的記錄層級設定為偵錯。	"真的"
operatorImage	允許完全置換的映像 trident-operator。	"
operatorImageTag	允許覆寫的標記 trident-operator 映像。	"
tridentIPv6	允許 Trident 在 IPv6 叢集中運作。	「假」
tridentK8sTimeout	覆寫大部分 Kubernetes API 作業的預設 30 秒逾時（如果非零、則以秒為單位）。	0
tridentHttpRequestTimeout	以取代 HTTP 要求的預設 90 秒逾時 0s 是超時的無限持續時間。不允許使用負值。	"90s"
tridentSilenceAutosupport	允許停用 Trident 定期 AutoSupport 報告。	「假」

選項	說明	預設
tridentAutosupportImageTag	允許覆寫 Trident AutoSupport 容器的映像標記。	<version>
tridentAutosupportProxy	可讓 Trident AutoSupport Container 透過 HTTP Proxy 撥打電話回家。	"
tridentLogFormat	設定 Trident 記錄格式(text、或 `json)。	"text"
tridentDisableAuditLog	停用 Trident 稽核記錄程式。	"真的"
tridentLogLevel	允許將 Trident 的日誌級別設置為：trace、debug、info、warn、error、或 `fatal)。	"info"
tridentDebug	允許將 Trident 的記錄層級設定為 debug。	「假」
tridentLogWorkflows	允許啟用特定的 Trident 工作流程、以進行追蹤記錄或記錄抑制。	"
tridentLogLayers	允許啟用特定 Trident 層以進行追蹤記錄或記錄抑制。	"
「TridentImage」	允許完全置換 Trident 的映像。	"
tridentImageTag	可覆寫 Trident 的映像標記。	"
tridentProbePort	允許覆寫 Kubernetes 活性 / 整備性探查所使用的預設連接埠。	"
windows	可在 Windows 工作節點上安裝 Trident。	「假」
enableForceDetach	允許啟用強制分離功能。	「假」
excludePodSecurityPolicy	不建立營運商 Pod 安全性原則。	「假」
nodePrep	可讓 Trident 準備 Kubernetes 叢集的節點、以使用指定的資料儲存傳輸協定來管理磁碟區。* 目前 `iscsi` 是唯一支援的值。*	

自訂Trident操作員安裝

Trident 運算子可讓您使用規格中的屬性來自訂 Trident 安裝 TridentOrchestrator。如果您想自訂超出引數允許範圍的安裝 TridentOrchestrator、請考慮使用 `tridentctl` 產生自訂的 YAML 資訊清單、以視需要進行修改。

瞭解控制器 Pod 和節點 Pod

Trident 以單一控制器 Pod 的形式執行、並在叢集中的每個工作節點上執行節點 Pod。節點 Pod 必須在任何想要掛載 Trident Volume 的主機上執行。

Kubernetes "節點選取器" 和 "容忍和污染" 用於限制 Pod 在特定或偏好的節點上執行。使用「ControllerPlugin」和 NodePlugin，您可以指定限制和置換。

- 控制器外掛程式可處理磁碟區資源配置與管理、例如快照和調整大小。
- 節點外掛程式會處理將儲存設備附加至節點的問題。

組態選項



`spec.namespace`` 在中指定 ``TridentOrchestrator``、表示安裝 Trident 的命名空間。此參數 * 安裝 Trident 後無法更新 *。嘗試這樣做會導致 `TridentOrchestrator`` 狀態變更為 ``Failed``。Trident 不打算跨命名空間移轉。

本表詳細說明 `TridentOrchestrator`` 屬性。

參數	說明	預設
<code>namespace`</code>	安裝 Trident 的命名空間	"default"
「Debug」	啟用 Trident 的偵錯功能	「假」
<code>enableForceDetach`</code>	僅限 <code>`ontap-san`</code> 、 <code>ontap-san-economy`</code> 和 <code>`ontap-nas-economy`</code> 與 Kubernetes Non-Graceful Node Shutdown (NGNS) 一起運作、讓叢集管理員能夠在節點發生問題時、將已掛載磁碟區的工作負載安全移轉至新節點。	「假」
<code>windows`</code>	設定為 <code>true`</code> 可在 Windows 工作節點上安裝。	「假」
<code>cloudProvider`</code>	設定為 "Azure" 在 AKS 叢集上使用託管身分識別或雲端身分識別時。在 EKS 叢集上使用雲端身分識別時、請設定為「AWS」。	""
<code>cloudIdentity`</code>	在 AKS 叢集上使用雲端身分識別時、請設定為工作負載身分識別 (「azure.Workload .idental/client-id : XXXXXXXX-xxxx-xxxx-xxxx-xxxx-xxxx-xxxxxxx」)。在 EKS 叢集上使用雲端身分識別時、請設定為 AWS IAM 角色 (「eks.amazonaws.com/role-arn: arn:AWS:iam::123456 :角色 Trident 角色」)。	""
<code>IPv6`</code>	透過 IPv6 安裝 Trident	錯
<code>k8sTimeout`</code>	Kubernetes 作業逾時	30sec
<code>silenceAutosupport`</code>	請勿將 AutoSupport 套裝組合傳送至 NetApp 自動	「假」
「autosupportImage」	遙測的容器影像 AutoSupport	"netapp/trident-autosupport:24.10"
「autosupportProxy」	用於傳送 AutoSupport 的 Proxy 位址 / 連接埠遙測	"http://proxy.example.com:8888"
解除安裝	用於解除安裝 Trident 的旗標	「假」
<code>logFormat`</code>	要使用的 Trident 記錄格式 [text,json]	"text"
「TridentImage」	要安裝的 Trident 映像	"netapp/trident:24.10"

參數	說明	預設
「影像登錄」	內部登錄的路徑、格式 <registry FQDN>[:port][/subpath]	"k8s.gcr.io" (Kubernetes 1.19+) 或 "quay.io/k8s/scsi"
《kubeletDir	主機上的kubelet目錄路徑	"/var/lib/kubelet"
wipeout	要刪除以執行 Trident 完整移除的資源清單	
「imagePullSecrets」	從內部登錄擷取映像的機密	
imagePullPolicy	設定Trident運算子的影像提取原則。有效值包括： Always 永遠拉出映像。 IfNotPresent 僅當節點上尚未存在映像時才提取映像。 Never 永遠不要拉動映像。	IfNotPresent
「controllerPluginNodeSelector」	用於 Pod 的其他節點選取器。格式與相同 pod.spec.nodeSelector。	無預設值；選用
「控制器插件」	覆寫 Pod 的 Kubernetes 公差。遵循與相同的格式 pod.spec.Tolerations。	無預設值；選用
「nodePluginNodeSelector」	用於 Pod 的其他節點選取器。格式與相同 pod.spec.nodeSelector。	無預設值；選用
「nodePluginTolerations」	覆寫 Pod 的 Kubernetes 公差。遵循與相同的格式 pod.spec.Tolerations。	無預設值；選用
nodePrep	可讓 Trident 準備 Kubernetes 叢集的節點、以使用指定的資料儲存傳輸協定來管理磁碟區。* 目前 `iscsi` 是唯一支援的值。*	



如需格式化 Pod 參數的詳細資訊、請參閱 ["將Pod指派給節點"](#)。

強制分離的詳細資料

「強制分離」僅適用於 `ontap-san`、`ontap-san-economy` 和 `onatp-nas-economy`。啟用強制分離之前、必須先在 Kubernetes 叢集上啟用非正常節點關機 (NGNS)。如需詳細資訊、請 ["Kubernetes：非正常節點關機"](#) 參閱。



使用驅動程式時 `ontap-nas-economy`、您需要將後端組態中的參數設定 `autoExportPolicy` 為 `true`、以便 Trident 可以使用受管理的匯出原則套用的污染來限制從 Kubernetes 節點的存取。



由於 Trident 仰賴 Kubernetes NGNS、因此在重新排程所有不可容忍的工作負載之前、請勿移除 `out-of-service` 不良節點的污點。如果不考慮套用或移除污染、可能會危及後端資料保護。

當 Kubernetes 叢集管理員已將 `Tintt` 套用 `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` 至節點、並 `enableForceDetach` 設定為 `true` 時、Trident 會判斷節點狀態、並：

1. 停止掛載到該節點之磁碟區的後端 I/O 存取。
2. 將 Trident 節點物件標記為 `dirty` (不適用於新出版物)。



Trident 控制器將拒絕新的發佈 Volume 要求、直到 Trident 節點 Pod 重新驗證節點（標記為之後）為止 dirty。除非 Trident 能夠驗證節點（新出版品安全）、否則任何排程使用已掛載 PVC 的工作負載（即使在叢集節點健全且準備就緒之後）都不會被接受 clean。

還原節點健全狀況並移除污染時、Trident 將：

1. 識別並清除節點上過時的已發佈路徑。
2. 如果節點處於某個狀態（已移除服務外污染、且節點處於 Ready 狀態）、且所有過時的已發佈路徑均為乾淨、則 cleanable Trident 會將節點重新接收為 clean、並允許新的已發佈磁碟區至節點。

組態範例

您可以在中使用屬性 [\[組態選項\]](#) 定義時 TridentOrchestrator 以自訂安裝。

基本自訂組態

這是基本自訂安裝的範例。

```
cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

節點選取器

此範例會安裝 Trident 搭配節點選取器。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

Windows 工作者節點

此範例會在 Windows 工作者節點上安裝 Trident。

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

在 AKS 叢集上的託管身分識別

此範例會安裝 Trident 、以在 AKS 叢集上啟用託管身分識別。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
```

在 AKS 叢集上的雲端身分識別

此範例會安裝 Trident 、以搭配使用於 AKS 叢集上的雲端身分識別。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxxx'
```

EKS 叢集上的雲端身分識別

此範例會安裝 Trident 、以搭配使用於 AKS 叢集上的雲端身分識別。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

GKE 的雲端身分識別

此範例會安裝 Trident 、以搭配 GKE 叢集上的雲端身分識別使用。

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

使用tridentctl安裝

使用tridentctl安裝

您可以使用安裝 Trident `tridentctl` 。此程序適用於 Trident 所需的容器映像儲存在私有登錄中或不儲存在私有登錄中的安裝。若要自訂您的 `tridentctl` 部署、請參閱["自訂試用部署"](#)。

Trident 24.10 的重要資訊

- 您必須閱讀下列有關 Trident 的重要資訊。 *

 的 Trident 相關資訊

- Kubernetes 1.27 現在支援 Trident。升級Kubernetes之前先升級Trident。
 - Trident 嚴格強制在 SAN 環境中使用多重路徑組態、建議在 multipath.conf 檔案中使用值 `find_multipaths: no`。
- 使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

使用安裝 Trident tridentctl

檢閱 "[安裝總覽](#)" 為了確保您符合安裝先決條件、並為您的環境選擇正確的安裝選項。

開始之前

在開始安裝之前、請先登入Linux主機、然後確認它正在管理正常運作的 "[支援的Kubernetes叢集](#)" 而且您擁有必要的權限。



使用OpenShift時、請在所有範例中使用「oc」而非「kubectl」、然後先執行「oc login -u system:admin」或「occ login-u kube-admin」、以*系統：admin*登入。

1. 驗證Kubernetes版本：

```
kubectl version
```

2. 驗證叢集管理員權限：

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. 確認您可以啟動使用Docker Hub映像的Pod、並透過Pod網路連線至儲存系統：

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
ping <management IP>
```

步驟1：下載Trident安裝程式套件

Trident 安裝程式套件會建立 Trident Pod、設定用於維持其狀態的 CRD 物件、並初始化 CSI 側板、以執行資源配置及將磁碟區附加至叢集主機等動作。從下載並解壓縮最新版本"[GitHub的_Assets區段](#)"的Trident 安裝程式。在範例中、使用您所選的 Trident 版本更新 <XX.XX.x.tar.gz> Trident。

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

步驟 2：安裝 Trident

執行命令、在所需的命名空間中安裝 Trident `tridentctl install`。您可以新增其他引數來指定映像登錄位置。

標準模式

```
./tridentctl install -n trident
```

一個登錄中的映像

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:24.10 --trident
-image <your-registry>/trident:24.10.0
```

不同登錄中的映像

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:24.10 --trident
-image <your-registry>/trident:24.10.0
```

您的安裝狀態應該類似這樣。

```

.....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                        namespace=trident
pod=trident-controller-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                version=24.10.0
INFO Trident installation succeeded.
.....

```

驗證安裝

您可以使用Pod建立狀態或來驗證安裝 `tridentctl`。

使用Pod建立狀態

您可以檢閱建立的 Pod 狀態、確認 Trident 安裝是否已完成：

```
kubectl get pods -n trident
```

NAME	READY	STATUS	RESTARTS	AGE
trident-controller-679648bd45-cv2mx	6/6	Running	0	5m29s
trident-node-linux-vgc8n	2/2	Running	0	5m29s



如果安裝程式未成功完成或 `trident-controller-<generated id>` (`trident-csi-<generated id>` 在23.01之前的版本中) 沒有*執行中*的狀態、表示平台尚未安裝。使用 `-d` 至 "[開啟偵錯模式](#)" 並疑難排解問題。

使用 `tridentctl`

您可以使用 `tridentctl` 檢查已安裝的 Trident 版本。

```
./tridentctl -n trident version
```

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 24.10.0        | 24.10.0        |
+-----+-----+
```

組態範例

以下範例提供使用安裝 Trident 的範例組態 `tridentctl`。

Windows 節點

若要啟用 Trident 在 Windows 節點上執行：

```
tridentctl install --windows -n trident
```

強制分離

如需強制分離的詳細資訊、請參閱 ["自訂Trident操作員安裝"](#)。

```
tridentctl install --enable-force-detach=true -n trident
```

自訂tridentctl安裝

您可以使用 Trident 安裝程式自訂安裝。

深入瞭解安裝程式

Trident 安裝程式可讓您自訂屬性。例如，如果您已將 Trident 映像複製到私有儲存庫，則可以使用指定映像名稱 `--trident-image`。如果您已將 Trident 映像以及所需的 CSI sidecar 映像複製到私有儲存庫，最好使用採用格式的交流器 `<registry FQDN>[:port]` 來指定該儲存庫的位置 `--image-registry`。



在私有儲存庫中安裝 Trident 時、如果您使用 `--image-registry` 交換器來指定儲存庫位置、請勿在儲存庫路徑中使用 `/netapp/`。例如：`./tridentctl install --image-registry <image-registry> -n <namespace>`

如果您使用 Kubernetes 的發佈方式、而其中的「kubelet」將資料保留在通常「`/var/lib/kubelet`」以外的路徑上、您可以使用「`-kubelet-dir`」來指定替代路徑。

如果您需要自訂安裝、而不需要安裝程式的引數允許、也可以自訂部署檔案。使用「`-generate-custom-yaml`」參數、可在安裝程式的「設置」目錄中建立下列 Yaml 檔案：

使用 Trident

準備工作節點

Kubernetes叢集中的所有工作節點都必須能夠掛載您已為Pod配置的磁碟區。若要準備工作節點，您必須根據您選擇的驅動程式來安裝 NFS，iSCSI，NVMe / TCP 或 FC 工具。

選擇適當的工具

如果您使用的是驅動程式組合、則應該安裝所有必要的驅動程式工具。最新版本的 RedHat CoreOS 預設會安裝這些工具。

NFS工具

"[安裝 NFS 工具](#)" 如果您使用的是：ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、azure-netapp-files、gcp-cvs。

iSCSI工具

"[安裝iSCSI工具](#)" 如果您使用的是：ontap-san、ontap-san-economy、solidfire-san。

NVMe 工具

"[安裝 NVMe 工具](#)" 如果您正在使用 ontap-san 適用於透過 TCP（NVMe / TCP）傳輸協定的非揮發性記憶體高速（NVMe）。



我們建議使用適用於 NVMe / TCP 的 ONTAP 9.12 或更新版本。

SCSI over FC 工具

- SCSI over Fibre Channel（FC）是 Trident 24.10 版本的技術預覽功能。*

"[安裝iSCSI工具](#)"如果您使用 ontap-san sanType fcp（SCSI over FC）。

如需詳細資訊、請參閱 "[設定 FC 擴大機、FC-NVMe SAN 主機的方法](#)"。

節點服務探索

Trident 會嘗試自動偵測節點是否可以執行 iSCSI 或 NFS 服務。



節點服務探索可識別探索到的服務、但無法保證服務已正確設定。相反地、沒有探索到的服務並不保證磁碟區掛載會失敗。

檢閱事件

Trident 會為節點建立事件、以識別探索到的服務。若要檢閱這些事件、請執行：

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

檢閱探索到的服務

Trident 會識別 Trident 節點 CR 上每個節點啟用的服務。若要檢視探索到的服務、請執行：

```
tridentctl get node -o wide -n <Trident namespace>
```

NFS磁碟區

使用作業系統的命令來安裝NFS工具。確保NFS服務在開機期間啟動。

RHEL 8以上

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```



安裝NFS工具之後、請重新啟動工作節點、以避免將磁碟區附加至容器時發生故障。

iSCSI磁碟區

Trident 可以自動建立 iSCSI 工作階段、掃描 LUN 、探索多重路徑裝置、將其格式化、並將其裝載至 Pod 。

iSCSI自我修復功能

對於 ONTAP 系統、Trident 每五分鐘執行一次 iSCSI 自我修復、以：

1. *識別*所需的iSCSI工作階段狀態和目前的iSCSI工作階段狀態。
2. *比較*所需狀態與目前狀態、以識別所需的維修。Trident 會決定維修優先順序、以及何時優先執行維修。
3. 執行必要的修復、以將目前的iSCSI工作階段狀態恢復至所需的iSCSI工作階段狀態。



自我修復活動記錄位於個別 Dem隨 選裝置上的容器中 `trident-main`。若要檢視記錄、您必須在 Trident 安裝期間將其設 `debug` 為「true」。

Trident iSCSI 自我修復功能有助於防止：

- 發生網路連線問題後、可能會發生過時或不正常的iSCSI工作階段。如果工作階段過時、Trident 會在登出前等待七分鐘、以重新建立與入口網站的連線。



例如、如果在儲存控制器上旋轉CHAP機密、而網路失去連線、則舊的 (`stal_`) CHAP機密可能會持續存在。自我修復可辨識此情況、並自動重新建立工作階段、以套用更新的CHAP機密。

- 遺失iSCSI工作階段
- 遺失LUN
- 升級 Trident 之前應考慮的要點 *
- 如果僅使用每個節點的 igroup （於 23.04+ 推出）、iSCSI 自我修復將會為 SCSI 匯流排中的所有裝置啟動 SCSI 重新掃描。
- 如果僅使用後端範圍的 igroup （自 2004 年 23 日起已過時）、iSCSI 自我修復將會針對 SCSI 匯流排中的確切 LUN ID 啟動 SCSI 重新掃描。
- 如果混合使用每個節點的 igroup 和後端範圍的 igroup 、iSCSI 自我修復將會啟動 SCSI 重新掃描、以取得 SCSI 匯流排中的確切 LUN ID 。

安裝iSCSI工具

使用適用於您作業系統的命令來安裝iSCSI工具。

開始之前

- Kubernetes叢集中的每個節點都必須具有唯一的IQN。這是必要的先決條件。
- 若搭配使用RMCOS 4.5或更新版本、或其他與RHEL相容的Linux套裝作業系統 solidfire-san 驅動程式和元素OS 12.5或更早版本、請確定CHAP驗證演算法已在中設定為MD5 /etc/iscsi/iscsid.conf。元素12.7提供安全的FIPS相容CHAP演算法SHA1、SHA-256和SHA3-256。

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*\/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- 使用執行RHEL/RedHat CoreOS搭配iSCSI PV的工作節點時、請指定 discard StorageClass中的掛載選項、以執行即時空間回收。請參閱 "[RedHat文件](#)"。

RHEL 8以上

1. 安裝下列系統套件：

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-  
multipath
```

2. 檢查iscsite-initier-utils版本是否為6.6.0.874-2.el7或更新版本：

```
rpm -q iscsi-initiator-utils
```

3. 啟用多重路徑：

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```



確保在"default"（錯誤）下"etc/multipath.conf"包含"fappe_multipaths no"。

4. 確保運行的是"iscsid"和"multipathd"：

```
sudo systemctl enable --now iscsid multipathd
```

5. 啟用並啟動「iSCSI」：

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. 安裝下列系統套件：

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools  
scsitools
```

2. 檢查開放式iSCSI版本是否為2.0.874-5ubuntu2.10或更新版本（適用於雙聲網路）或2.0.874-7.1ubuntu6.1或更新版本（適用於焦點）：

```
dpkg -l open-iscsi
```

3. 將掃描設為手動：

```
sudo sed -i 's/^\(node.session.scan\).*\/\1 = manual/'  
/etc/iscsi/iscsid.conf
```

4. 啟用多重路徑：

```
sudo tee /etc/multipath.conf <<-EOF  
defaults {  
    user_friendly_names yes  
    find_multipaths no  
}  
EOF  
sudo systemctl enable --now multipath-tools.service  
sudo service multipath-tools restart
```



確保在"default"（錯誤）下"etc/multipath.conf"包含"fappp_multipaths no"。

5. 確保已啟用並執行「open-iscsi」和「多路徑工具」：

```
sudo systemctl status multipath-tools  
sudo systemctl enable --now open-iscsi.service  
sudo systemctl status open-iscsi
```



對於Ubuntu 18.04、您必須先使用「iscsiadm」探索目標連接埠、然後再啟動「open-iscsi」、iSCSI精靈才能啟動。您也可以修改「iSCSI」服務、以自動啟動「iscsid」。

設定或停用 iSCSI 自我修復

您可以設定下列 Trident iSCSI 自我修復設定、以修復過時的工作階段：

- ***iSCSI 自我修復時間間隔***：決定啟動 iSCSI 自我修復的頻率（預設值：5 分鐘）。您可以設定較小的數字、或設定較大的數字、將其設定為較常執行。



將 iSCSI 自我修復時間間隔設為 0 會完全停止 iSCSI 自我修復。我們不建議停用 iSCSI 自我修復功能；只有在 iSCSI 自我修復功能未如預期運作或無法進行偵錯時、才應停用 iSCSI 自我修復功能。

- ***iSCSI 自我修復等待時間***：決定 iSCSI 自我修復等待的時間、再登出不正常的工作階段並再次嘗試登入（預設值：7 分鐘）。您可以將其設定為較大的數目、以便識別為不正常的工作階段必須等待較長時間才能登出、然後再嘗試重新登入、或是較小的數目來登出和較早登入。

掌舵

若要設定或變更 iSCSI 自我修復設定、請通過 `iscsiSelfHealingInterval` 和 `iscsiSelfHealingWaitTime` 在 helm 安裝或 helm 更新期間的參數。

以下範例將 iSCSI 自我修復間隔設為 3 分鐘、而自我修復等候時間設為 6 分鐘：

```
helm install trident trident-operator-100.2410.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

試用

若要設定或變更 iSCSI 自我修復設定、請通過 `iscsi-self-healing-interval` 和 `iscsi-self-healing-wait-time` 在 Tridentctl 安裝或更新期間的參數。

以下範例將 iSCSI 自我修復間隔設為 3 分鐘、而自我修復等候時間設為 6 分鐘：

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

NVMe / TCP 磁碟區

使用適用於您作業系統的命令來安裝 NVMe 工具。



- NVMe 需要 RHEL 9 或更新版本。
- 如果 Kubernetes 節點的核心版本太舊、或 NVMe 套件無法用於您的核心版本、您可能必須使用 NVMe 套件將節點的核心版本更新為一個。

RHEL 9.

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

驗證安裝

安裝後、請使用命令確認 Kubernetes 叢集中的每個節點都有唯一的 NQN ：

```
cat /etc/nvme/hostnqn
```



Trident 會修改此 `ctrl_device_tmo` 值、確保 NVMe 在故障時不會放棄路徑。請勿變更此設定。

光纖通道（FC）支援

您現在可以搭配 Trident 使用光纖通道（FC）傳輸協定，在 ONTAP 系統上配置及管理儲存資源。

- SCSI over Fibre Channel（FC）是 Trident 24.10 版本的技術預覽功能。*

Fibre Channel 是企業儲存環境中廣泛採用的一種傳輸協定，因為它具有高效能，可靠性和擴充性。它為儲存裝置提供強大且有效率的通訊通道，可快速且安全地傳輸資料。透過光纖通道使用 SCSI，您可以運用現有的 SCSI 儲存基礎架構，同時享有光纖通道的高效能和長距離功能。它可整合儲存資源，並建立可擴充且有效率的儲存區域網路（SAN），以低延遲處理大量資料。

使用 FC 功能搭配 Trident，您可以執行下列動作：

- 使用部署規格動態配置 PVC。
- 擷取 Volume 快照，並從快照建立新的 Volume。
- 複製現有的 FC-PVC。
- 調整已部署磁碟區的大小。

先決條件

設定 FC 所需的網路和節點設定。

網路設定

1. 取得目標介面的 WWPN。如需詳細資訊、請參閱 ["網路介面顯示"](#)。
2. 取得啟動器（主機）介面的 WWPN。

請參閱對應的主機作業系統公用程式。

3. 使用主機和目標的 WWPN 在 FC 交換器上設定分區。

如需詳細資訊，請參閱重新輸入交換器廠商文件。

如需詳細資訊，請參閱下列 ONTAP 文件：

- ["Fibre Channel和FCoE分區總覽"](#)
- ["設定 FC 擴大機、FC-NVMe SAN 主機的方法"](#)

準備工作節點

Kubernetes叢集中的所有工作節點都必須能夠掛載您已為Pod配置的磁碟區。若要為 FC 準備工作節點，您必須安裝必要的工具。

安裝 FC 工具

使用作業系統的命令來安裝FC工具。

- 使用執行RHEL/RedHat CoreOS搭配iSCSI PV的工作節點時、請指定 `discard` StorageClass中的掛載選項、以執行即時空間回收。請參閱 "[RedHat文件](#)"。

RHEL 8以上

1. 安裝下列系統套件：

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-  
multipath
```

2. 檢查iscsite-initier-utils版本是否為6.6.0.874-2.el7或更新版本：

```
rpm -q iscsi-initiator-utils
```

3. 啟用多重路徑：

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```



確保在"default"（錯誤）下"etc/multipath.conf"包含"fappe_multipaths no"。

4. 確保運行的是"iscsid"和"multipathd"：

```
sudo systemctl enable --now iscsid multipathd
```

5. 啟用並啟動「iSCSI」：

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. 安裝下列系統套件：

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools  
scsitools
```

2. 檢查開放式iSCSI版本是否為2.0.874-5ubuntu2.10或更新版本（適用於雙聲網路）或2.0.874-7.1ubuntu6.1或更新版本（適用於焦點）：

```
dpkg -l open-iscsi
```

3. 將掃描設為手動：

```
sudo sed -i 's/^\(node.session.scan\).*\/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. 啟用多重路徑：

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```



確保在"default"（錯誤）下"etc/multipath.conf"包含"fappp_multipaths no"。

5. 確保已啟用並執行「open-iscsi」和「多路徑工具」：

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```



對於Ubuntu 18.04、您必須先使用「iscsiadm」探索目標連接埠、然後再啟動「open-iscsi」、iSCSI精靈才能啟動。您也可以修改「iSCSI」服務、以自動啟動「iscsid」。

建立後端組態

為驅動程式和 fcp sanType 建立 Trident 後端 ontap-san。

請參閱：

- ["準備使用ONTAP 支援的SAN驅動程式來設定後端"](#)
- ["SAN組態選項與範例ONTAP"](#)

FC 的後端組態範例

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  sanType: fcp
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

建立儲存類別

如需詳細資訊、請參閱：

- ["儲存組態選項"](#)

儲存類別範例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fcp-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  protocol: "fcp"
  storagePool: "aggr1"
allowVolumeExpansion: True
```

設定及管理後端

設定後端

後端定義 Trident 與儲存系統之間的關係。它告訴Trident如何與該儲存系統通訊、以及Trident如何從該儲存系統配置磁碟區。

Trident 會自動從後端提供符合儲存類別所定義需求的儲存資源池。瞭解如何設定儲存系統的後端。

- ["設定Azure NetApp Files 一個靜態後端"](#)

- "設定Cloud Volumes Service 適用於Google Cloud Platform後端的功能"
- "設定NetApp HCI 一個不只是功能的SolidFire 後端"
- "使用ONTAP 功能不一的Cloud Volumes ONTAP NAS驅動程式來設定後端"
- "使用ONTAP 不支援的Cloud Volumes ONTAP SAN驅動程式來設定後端"
- "搭配 Amazon FSX for NetApp ONTAP 使用 Trident"

Azure NetApp Files

設定Azure NetApp Files 一個靜態後端

您可以將 Azure NetApp Files 設定為 Trident 的後端。您可以使用 Azure NetApp Files 後端連接 NFS 和 SMB 磁碟區。Trident 也支援使用 Azure Kubernetes Services (aks) 叢集的託管身分識別來進行認證管理。

Azure NetApp Files 驅動程式詳細資料

Trident 提供下列 Azure NetApp Files 儲存驅動程式、可與叢集進行通訊。支援的存取模式包括：*ReadWriteOnce* (rwo)、*ReadOnlyMany* (ROX)、*_ReadWriteMany* (rwx)、*_ReadWriteOncePod* (RWOP)。

驅動程式	傳輸協定	Volume模式	支援的存取模式	支援的檔案系統
《azure-NetApp-fil形》	NFS 中小企業	檔案系統	Rwo、ROX、rwx、RWOP	nfs、smb

考量

- Azure NetApp Files 服務不支援小於 50 GiB 的磁碟區。如果要求較小的磁碟區、Trident 會自動建立 50-GiB 磁碟區。
- Trident 僅支援掛載至 Windows 節點上執行的 Pod 的 SMB 磁碟區。

管理的身分識別

Trident 支援 "託管身分識別" Azure Kubernetes 服務叢集。若要善用託管身分識別所提供的簡化認證管理功能、您必須具備：

- 使用 aks 部署的 Kubernetes 叢集
- 在 aks Kubernetes 叢集上設定的託管身分識別
- 安裝的 Trident，其中包括 `cloudProvider` 要指定 `Azure` 的。

Trident 運算子

若要使用 Trident 運算子安裝 Trident、請編輯 `tridentorchestrator_cr.yaml` 以設定為 `cloudProvider "Azure"`。例如：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

掌舵

以下範例使用環境變數將 Trident Set 安裝 `cloudProvider` 至 `Azure` `$CP`：

```
helm install trident trident-operator-100.2410.0.tgz --create
--namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

`tridentctl`

以下範例安裝 Trident 並將旗標設定 `cloudProvider` 為 `Azure`：

```
tridentctl install --cloud-provider="Azure" -n trident
```

雲端身分識別

雲端身分識別可讓 Kubernetes Pod 以工作負載身分驗證來存取 Azure 資源、而非提供明確的 Azure 認證。

若要在 Azure 中使用雲端身分識別、您必須具備：

- 使用 aks 部署的 Kubernetes 叢集
- 在 OKS Kubernetes 叢集上設定的工作負載識別和 oidc-c 發行者
- 安裝的 Trident、其中包含 `cloudProvider` 指定 `"Azure"` 及 `cloudIdentity` 指定工作負載身分識別的

Trident 運算子

若要使用 Trident 運算子安裝 Trident、請編輯 `tridentorchestrator_cr.yaml` 以設定為 `cloudProvider "Azure"`、並設定 `cloudIdentity` 為 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx`。

例如：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  *cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
  xxxxx-xxxx-xxxxxxxxxxxxx'*
```

掌舵

使用下列環境變數設定 * 雲端供應商 (CP) * 和 * 雲端身分識別 (CI) * 旗標的值：

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxxx'"
```

下列範例使用環境變數安裝 Trident 並將設定 `cloudProvider` 為 `Azure $CP`、並使用環境變數 `$CI` 設定 `cloudIdentity`：

```
helm install trident trident-operator-100.2410.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

`tridentctl`

使用下列環境變數設定 * 雲端供應商 * 和 * 雲端 IDENTITY * 旗標的值：

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxxx"
```

以下範例會安裝 Trident 並將旗標設定 `cloud-provider` 為 `$CP`、和 `cloud-identity $CI`：

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

準備設定 **Azure NetApp Files** 一個功能完善的後端

在您設定 Azure NetApp Files 完後端功能之前、您必須確保符合下列要求。

NFS 和 SMB 磁碟區的必要條件

如果您是第一次使用 Azure NetApp Files、或是在新位置使用、則必須先進行一些初始設定、才能設定 Azure NetApp Files 並建立 NFS Volume。請參閱 ["Azure：設定 Azure NetApp Files 功能以建立 NFS Volume"](#)。

若要設定及使用 ["Azure NetApp Files"](#) 後端、您需要下列項目：



- subscriptionID、tenantID、clientID、location 和 `clientSecret` 在 AKS 叢集上使用託管身分識別時為選用項目。
- tenantID、clientID 和 `clientSecret` 在 AKS 叢集上使用雲端身分識別時為選用項目。

- 容量集區。請參閱 ["Microsoft：為 Azure NetApp Files 建立容量集區"](#)。
- 委派給 Azure NetApp Files 的子網路。請參閱 ["Microsoft：將子網路委派給 Azure NetApp Files"](#)。
- Azure 訂閱提供的「SubscriptionID」Azure NetApp Files（含功能不支援的功能）。
- tenantID、clientID 和 `clientSecret` 從 ["應用程式註冊"](#) 在 Azure Active Directory 中、具備 Azure NetApp Files 充分的權限執行此功能。應用程式登錄應使用下列其中一項：
 - 擁有者或貢獻者角色 ["由 Azure 預先定義"](#)。
 - ["自訂貢獻者角色"](#)(assignableScopes (在訂閱級別))，具有以下權限，僅限於 Trident 所需的權限。建立自訂角色之後 ["使用 Azure 入口網站指派角色"](#)，。

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited
permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",
          "Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",
```

```

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations/delete",

    "Microsoft.Features/features/read",
    "Microsoft.Features/operations/read",
    "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
    ],
    "notActions": [],
    "dataActions": [],
    "notDataActions": []
    }
  ]
}
}

```

- Azure location 至少包含一個 ["委派的子網路"](#)。從 Trident 22.01 起 location 參數是後端組態檔最上層的必填欄位。會忽略虛擬資源池中指定的位置值。
- 以供使用 Cloud Identity 請取得 `client ID` 從 ["使用者指派的託管身分識別"](#) 並在中指定該 ID
 azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx。

SMB 磁碟區的其他需求

若要建立 SMB Volume、您必須具備：

- Active Directory 已設定並連線至 Azure NetApp Files。請參閱 ["Microsoft：建立及管理 Azure NetApp Files 的 Active Directory 連線"](#)。
- Kubernetes 叢集具備 Linux 控制器節點、以及至少一個執行 Windows Server 2022 的 Windows 工作節點。Trident 僅支援掛載至 Windows 節點上執行的 Pod 的 SMB 磁碟區。
- 至少有一個 Trident 機密包含您的 Active Directory 認證、以便 Azure NetApp Files 能夠驗證至 Active Directory。產生機密 smbcreds：

```

kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'

```

- 設定為 Windows 服務的 SCSI Proxy。若要設定 csi-proxy、請參閱 ["GitHub：csi Proxy"](#) 或 ["GitHub：適用於 Windows 的 SCSI Proxy"](#) 適用於 Windows 上執行的 Kubernetes 節點。

列舉後端組態選項與範例 Azure NetApp Files

瞭解 Azure NetApp Files 的 NFS 和 SMB 後端組態選項、並檢閱組態範例。

後端組態選項

Trident 使用後端組態（子網路、虛擬網路、服務層級和位置）、在所要求位置的可用容量集區上建立 Azure NetApp Files Volume、並符合所要求的服務層級和子網路。



Trident 不支援手動 QoS 容量集區。

Azure NetApp Files 後端提供這些組態選項。

參數	說明	預設
「分度」		永遠為1
「storageDriverName」	儲存驅動程式名稱	「Azure - NetApp-Files」
「後端名稱」	自訂名稱或儲存後端	驅動程式名稱+「_」+隨機字元
《訂閱ID》	Azure訂閱的訂閱ID 在 AKS 叢集上啟用受管理的身分識別時為選用項目。	
「TenantId」	應用程式註冊的租戶ID 在 AKS 叢集上使用託管身分識別或雲端身分識別時、為選用項目。	
"clientId"	應用程式註冊的用戶端ID 在 AKS 叢集上使用託管身分識別或雲端身分識別時、為選用項目。	
「客戶機密」	應用程式註冊的用戶端機密 在 AKS 叢集上使用託管身分識別或雲端身分識別時、為選用項目。	
《服務層級》	其中一種是「標準」、「高級」或「超高」	"" (隨機)
位置	要建立新磁碟區的Azure位置名稱 在 AKS 叢集上啟用受管理的身分識別時為選用項目。	
"來源群組"	用於篩選已探索資源的資源群組清單	「[]」 (無篩選器)
《netappAccounts》	篩選探索資源的NetApp帳戶清單	「[]」 (無篩選器)
《容量Pools》	用於篩選已探索資源的容量集區清單	「[]」 (無篩選器、隨機)
「虛擬化網路」	具有委派子網路的虛擬網路名稱	"

參數	說明	預設
《Subnet》	委派給「microsoft.Netapp/volumes`」的子網路名稱	"
《網路功能》	Volume的vnet功能集可能是 Basic 或 Standard。並非所有地區都提供網路功能、可能必須在訂閱中啟用。指定 networkFeatures 如果未啟用此功能、則會導致磁碟區資源配置失敗。	"
「nfsMountOptions」	精細控制NFS掛載選項。SMB磁碟區已忽略。若要使用NFS 4.1版掛載磁碟區、請包含 nfsvers=4 在以逗號分隔的掛載選項清單中、選擇NFS v4.1。儲存類別定義中設定的掛載選項會覆寫在後端組態中設定的掛載選項。	"nfsves=3"
《限制Volume大小》	如果要求的磁碟區大小高於此值、則資源配置失敗	"" (預設不強制執行)
「DebugTraceFlags」	疑難排解時要使用的偵錯旗標。範例：「{"API":假、「方法」:真、「探索」:true}。除非您正在進行疑難排解並需要詳細的記錄傾印、否則請勿使用此功能。	null
nasType	設定NFS或SMB磁碟區建立。選項包括 nfs、smb 或 null。NFS磁碟區的預設值設為null。	nfs
supportedTopologies	代表此後端所支援的區域和區域清單。如需詳細資訊、請"使用「csi拓撲」"參閱。	



如需網路功能的詳細資訊、請參閱 ["設定Azure NetApp Files 適用於某個聲音量的網路功能"](#)。

必要的權限與資源

如果您在建立 PVC 時收到「找不到容量集區」錯誤、您的應用程式註冊可能沒有相關的必要權限和資源（子網路、虛擬網路、容量集區）。如果啟用除錯功能、Trident 會記錄建立後端時所探索到的 Azure 資源。確認使用的角色是否適當。

的值 resourceGroups、netappAccounts、capacityPools、virtualNetwork`和 `subnet 可以使用簡短或完整名稱來指定。在大多數情況下、建議使用完整名稱、因為短名稱可以符合多個名稱相同的資源。

。resourceGroups、netappAccounts`和 `capacityPools 值是篩選器、可將探索到的資源集合限制在此儲存後端可用的資源、並可任意組合指定。完整名稱格式如下：

類型	格式
資源群組	<資源群組>

類型	格式
NetApp帳戶	資源群組//<NetApp帳戶>
容量資源池	資源群組//<NetApp帳戶>/<容量資源池>
虛擬網路	資源群組//<虛擬網路>
子網路	資源群組//<虛擬網路>/<子網路>

Volume資源配置

您可以在組態檔的特殊區段中指定下列選項、以控制預設的Volume資源配置。請參閱 [\[組態範例\]](#) 以取得詳細資料。

參數	說明	預設
「匯出規則」	匯出新磁碟區的規則。 <code>exportRule</code> 必須是以逗號分隔的清單、以CIDR表示法列出所有的IPv4位址或IPv4子網路組合。SMB磁碟區已忽略。	「0.0.0.0/0」
「napshotDir	控制.snapshot目錄的可見度	針對 NFSv3 的 NFSv4 "false" 為 "true"
《大小》	新磁碟區的預設大小	100公克
「unixPermissions」	新磁碟區的UNIX權限（4個八進位數字）。SMB磁碟區已忽略。	""（預覽功能、訂閱時需要白名單）

組態範例

下列範例顯示基本組態、讓大部分參數保留預設值。這是定義後端最簡單的方法。

最小組態

這是絕對最低的后端組態。使用此組態、Trident 會探索您在設定位置中委派給 Azure NetApp Files 的所有 NetApp 帳戶、容量集區和子網路、並隨機將新磁碟區放在其中一個集區和子網路上。由於省略、因此 `nasType nfs` 會套用預設值、而后端會為 NFS 磁碟區進行資源配置。

當您剛開始使用 Azure NetApp Files 並試用時、這項組態是理想的選擇、但實際上您會想要為您所配置的磁碟區提供額外的範圍。

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
  tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
  clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
  clientSecret: SECRET
  location: eastus
```

管理的身分識別

此后端組態已不再如此 `subscriptionID`、`tenantID`、`clientID` 和 `clientSecret`，使用託管身分識別時為選用功能。

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools: ["ultra-pool"]
  resourceGroups: ["aks-ami-eastus-rg"]
  netappAccounts: ["smb-na"]
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
```

雲端身分識別

此後端組態已不再如此 tenantID、clientID 和 clientSecret (使用雲端身分識別時為選用功能)。

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools: ["ultra-pool"]
  resourceGroups: ["aks-ami-eastus-rg"]
  netappAccounts: ["smb-na"]
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

具有容量集區篩選器的特定服務層級組態

此後端組態會將磁碟區放置在 Azure 的位置、並置於 eastus 容量集區中 Ultra。Trident 會自動探索該位置中委派給 Azure NetApp Files 的所有子網路、並隨機在其中一個磁碟區上放置新的磁碟區。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
```

此後端組態可進一步將磁碟區放置範圍縮小至單一子網路、並修改部分Volume資源配置預設值。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
virtualNetwork: my-virtual-network
subnet: my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: 'true'
  size: 200Gi
  unixPermissions: '0777'
```

此後端組態可在單一檔案中定義多個儲存集區。當您有多個容量集區支援不同的服務層級、而且想要在Kubernetes中建立代表這些層級的儲存類別時、這很有用。虛擬資源池標籤是用來區分資源池的依據

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
- application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
- labels:
  performance: gold
  serviceLevel: Ultra
  capacityPools:
  - ultra-1
  - ultra-2
  networkFeatures: Standard
- labels:
  performance: silver
  serviceLevel: Premium
  capacityPools:
  - premium-1
- labels:
  performance: bronze
  serviceLevel: Standard
  capacityPools:
  - standard-1
  - standard-2
```

Trident 可根據地區和可用性區域、為工作負載提供更多資源。`supportedTopologies` 此後端組態中的區塊用於提供每個後端的區域和區域清單。此處指定的區域和區域值必須符合每個 Kubernetes 叢集節點上標籤的區域和區域值。這些區域和區域代表可在儲存類別中提供的允許值清單。對於包含後端所提供區域和區域子集的儲存類別、Trident 會在所述區域和區域中建立磁碟區。如需詳細資訊、請["使用「csi拓撲」"](#)參閱。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
supportedTopologies:
- topology.kubernetes.io/region: eastus
  topology.kubernetes.io/zone: eastus-1
- topology.kubernetes.io/region: eastus
  topology.kubernetes.io/zone: eastus-2
```

儲存類別定義

以下內容 StorageClass 定義請參閱上述儲存資源池。

使用的範例定義 `parameter.selector` 欄位

使用 `parameter.selector` 您可以為每個項目指定 StorageClass 用於裝載磁碟區的虛擬集區。該磁碟區會在所選的資源池中定義各個層面。

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze"
allowVolumeExpansion: true

```

SMB磁碟區的定義範例

使用 `nasType`、`node-stage-secret-name` 和 `node-stage-secret-namespace`、您可以指定SMB磁碟區、並提供所需的Active Directory認證資料。

預設命名空間的基本組態

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

每個命名空間使用不同的機密

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

每個磁碟區使用不同的機密

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```



`nasType: smb` 支援SMB磁碟區的集區篩選器。`nasType: nfs` 或 `nasType: null` NFS集區的篩選器。

建立後端

建立後端組態檔之後、請執行下列命令：

```
tridentctl create backend -f <backend-file>
```

如果後端建立失敗、表示後端組態有問題。您可以執行下列命令來檢視記錄、以判斷原因：

```
tridentctl logs
```

識別並修正組態檔的問題之後、您可以再次執行create命令。

Google Cloud NetApp Volumes

設定 **Google Cloud NetApp Volumes** 後端

您現在可以將 Google Cloud NetApp Volumes 設定為 Trident 的後端。您可以使用 Google Cloud NetApp Volumes 後端來附加 NFS 磁碟區。

Google Cloud NetApp Volumes 驅動程式詳細資料

Trident 提供 `google-cloud-netapp-volumes` 與叢集通訊的驅動程式。支援的存取模式包括：*ReadWriteOnce* (*rwo*)、*ReadOnlyMany* (*ROX*)、*_ReadWriteMany* (*rwX*)、*_ReadWriteOncePod* (*RWOP*)。

驅動程式	傳輸協定	Volume模式	支援的存取模式	支援的檔案系統
google-cloud-netapp-volumes	NFS	檔案系統	Rwo、ROX、rwX、RWOP	nfs

GKE 的雲端身分識別

雲端身分識別可讓 Kubernetes Pod 以工作負載身分驗證來存取 Google Cloud 資源、而非提供明確的 Google Cloud 身分證明。

若要在 Google Cloud 中善用雲端身分識別、您必須具備：

- 使用 GKE 部署的 Kubernetes 叢集。
- 在 GKE 叢集上設定的工作負載身分識別和 `oidc-c` 發行者。
- 安裝的 Trident、其中包含 `cloudProvider` 指定 "GCP" 及 `cloudIdentity` 指定工作負載身分識別的。

Trident 運算子

若要使用 Trident 運算子安裝 Trident、請編輯 `tridentorchestrator_cr.yaml` 以設定為 `cloudProvider "GCP"`、並設定 `cloudIdentity` 為 `iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com`。

例如：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "GCP"
  cloudIdentity: 'iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com'
```

掌舵

使用下列環境變數設定 * 雲端供應商 (CP) * 和 * 雲端身分識別 (CI) * 旗標的值：

```
export CP="GCP"
export ANNOTATION="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

以下範例使用環境變數安裝 Trident 並設定 `cloudProvider` 為 `GCP`、`$CP`、並使用環境變數 `$ANNOTATION` 設定 `cloudIdentity`：

```
helm install trident trident-operator-100.2406.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$ANNOTATION"
```

`tridentctl`

使用下列環境變數設定 * 雲端供應商 * 和 * 雲端 IDENTITY * 旗標的值：

```
export CP="GCP"
export ANNOTATION="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

以下範例會安裝 Trident 並將旗標設定 `cloud-provider` 為 `$CP`、和 `cloud-identity` `$ANNOTATION`：

```
tridentctl install --cloud-provider=$CP --cloud
-identity="$ANNOTATION" -n trident
```

準備設定 Google Cloud NetApp Volumes 後端

在您設定 Google Cloud NetApp Volumes 後端之前、您必須確保符合下列需求。

NFS Volume 的必要條件

如果您是第一次使用 Google Cloud NetApp Volumes、或是在新位置使用、則需要進行一些初始設定、才能設定 Google Cloud NetApp Volumes 並建立 NFS Volume。請參閱 ["開始之前"](#)。

在設定 Google Cloud NetApp Volumes 後端之前、請先確認您擁有下列項目：

- 使用 Google Cloud NetApp Volumes 服務設定的 Google Cloud 帳戶。請參閱 ["Google Cloud NetApp Volumes"](#)。
- Google Cloud 帳戶的專案編號。請參閱 ["識別專案"](#)。
- 具有 Volumes Admin (NetApp Volume 管理) 角色的 Google Cloud 服務帳戶 (netappcloudvolumes.admin。請參閱 ["身分識別與存取管理角色與權限"](#))。
- 您的 GCNV 帳戶的 API 金鑰檔案。請參閱 ["建立服務帳戶金鑰"](#)
- 儲存池。請參閱 ["儲存資源池總覽"](#)。

如需如何設定 Google Cloud NetApp Volumes 存取權限的詳細資訊、請 ["設定 Google Cloud NetApp Volumes 的存取權"](#)參閱。

Google Cloud NetApp Volumes 後端組態選項和範例

瞭解 Google Cloud NetApp Volumes 的 NFS 後端組態選項、並檢閱組態範例。

後端組態選項

每個後端都會在單一 Google Cloud 區域中配置磁碟區。若要在其他區域建立磁碟區、您可以定義其他後端。

參數	說明	預設
「分度」		永遠為1
「storageDriverName」	儲存驅動程式名稱	的值 storageDriverName 必須指定為「googoogle 雲端 -NetApp-Volumes」。
「後端名稱」	(選用) 儲存後端的自訂名稱	驅動程式名稱+「_」+ API 金鑰的一部分
storagePools	選用參數、用於指定用於建立磁碟區的儲存資源池。	
「ProjectNumber」	Google Cloud 帳戶專案編號。此值可在 Google Cloud 入口網站首頁找到。	
位置	Trident 建立 GCNV Volume 的 Google Cloud 位置。建立跨區域 Kubernetes 叢集時、在中建立的磁碟區 location 可用於跨多個 Google Cloud 區域的節點上排程的工作負載。跨區域流量會產生額外成本。	

參數	說明	預設
「apiKey」	具有此角色的 Google Cloud 服務帳戶的 API 金鑰 netappcloudvolumes.admin。其中包含Google Cloud服務帳戶私密金鑰檔案（逐字複製到後端組態檔）的JSON-格式內容。apiKey`必須包含下列金鑰的金鑰值配對：`type project_id`、`client_email`、`client_id`、`auth_uri`、`token_uri`、`auth_provider_x509_cert_url`、和`client_x509_cert_url`。	
「nfsMountOptions」	精細控制NFS掛載選項。	"nfsves=3"
《限制Volume大小》	如果要求的磁碟區大小高於此值、則資源配置失敗。	""（預設不強制執行）
《服務層級》	儲存池及其磁碟區的服務層級。這些值包括 flex、standard、premium`或`extreme。	
網路	用於 GCNV Volume 的 Google Cloud 網路。	
「DebugTraceFlags」	疑難排解時要使用的偵錯旗標。範例： { "api":false, "method":true}。除非您正在進行疑難排解並需要詳細的記錄傾印、否則請勿使用此功能。	null
supportedTopologies	代表此後端所支援的區域和區域清單。如需詳細資訊、請"使用「csi拓撲」"參閱。例如： supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a	

Volume資源配置選項

您可以在中控制預設的Volume資源配置 defaults 組態檔的一節。

參數	說明	預設
「匯出規則」	新磁碟區的匯出規則。必須是以逗號分隔的任何 IPv4 位址組合清單。	「0.0.0.0/0」
「snapshotDir	存取「.snapshot」目錄	針對 NFSv3 的 NFSv4 "false" 為 "true"
「快照保留區」	保留給快照的磁碟區百分比	"（接受預設值 0）
「unixPermissions」	新磁碟區的UNIX權限（4個八進位數字）。	"

組態範例

下列範例顯示基本組態、讓大部分參數保留預設值。這是定義後端最簡單的方法。

-----END PRIVATE KEY-----\n

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '123455380079'
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: '103346282737811234567'
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```



```
version: 1
storageDriverName: google-cloud-netapp-volumes
projectNumber: '123455380079'
location: europe-west6
serviceLevel: premium
storagePools:
- premium-pool1-europe-west6
- premium-pool2-europe-west6
apiKey:
  type: service_account
  project_id: my-gcnv-project
  client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
  client_id: '103346282737811234567'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```



```
znHczZsrtrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
XsYg6gyxy4zq7OlwWgLwGa==
-----END PRIVATE KEY-----
```

```
---
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '123455380079'
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: '103346282737811234567'
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: '10'
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
        serviceLevel: extreme
      defaults:
        snapshotReserve: '5'
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
        serviceLevel: premium
    - labels:
        performance: standard
        serviceLevel: standard
```

GKE 的雲端身分識別

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

支援的拓撲組態

Trident 可根據地區和可用性區域、為工作負載提供更多資源。`supportedTopologies` 此後端組態中的區塊用於提供每個後端的區域和區域清單。此處指定的區域和區域值必須符合每個 Kubernetes 叢集節點上標籤的區域和區域值。這些區域和區域代表可在儲存類別中提供的允許值清單。對於包含後端所提供區域和區域子集的儲存類別、Trident 會在所述區域和區域中建立磁碟區。如需詳細資訊、請["使用「csi拓撲」"](#)參閱。

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
- topology.kubernetes.io/region: asia-east1
  topology.kubernetes.io/zone: asia-east1-a
- topology.kubernetes.io/region: asia-east1
  topology.kubernetes.io/zone: asia-east1-b
```

接下來呢？

建立後端組態檔之後、請執行下列命令：

```
kubectl create -f <backend-file>
```

若要確認後端已成功建立、請執行下列命令：

```
kubectl get tridentbackendconfig
```

NAME	BACKEND NAME	BACKEND UUID
PHASE	STATUS	
backend-tbc-gcnv	backend-tbc-gcnv	b2fd1ff9-b234-477e-88fd-713913294f65
Bound	Success	

如果後端建立失敗、表示後端組態有問題。您可以使用命令來描述後端 `kubectl get tridentbackendconfig <backend-name>`、或是執行下列命令來檢視記錄以判斷原因：

```
tridentctl logs
```

識別並修正組態檔的問題之後、您可以刪除後端、然後再次執行 `create` 命令。

更多範例

儲存類別定義範例

以下是上述後端的基本 `StorageClass` 定義。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

- 使用欄位的範例定義 `parameter.selector`：`*`

使用 `parameter.selector` 您可以為用於裝載 `Volume` 的每個指定 `StorageClass` ["虛擬集區"](#)。該磁碟區會在所選的資源池中定義各個層面。

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=extreme"
  backendType: "google-cloud-netapp-volumes"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium"
  backendType: "google-cloud-netapp-volumes"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=standard"
  backendType: "google-cloud-netapp-volumes"

```

如需儲存類別的詳細資訊、請 ["建立儲存類別"](#) 參閱。

PVC 定義範例

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc

```

若要驗證 PVC 是否受限、請執行下列命令：

```
kubectl get pvc gcnv-nfs-pvc
```

NAME	STATUS	VOLUME	CAPACITY
gcnv-nfs-pvc	Bound	pvc-b00f2414-e229-40e6-9b16-ee03eb79a213	100Gi
ACCESS MODES	STORAGECLASS	AGE	
RWX	gcnv-nfs-sc	1m	

設定 Cloud Volumes Service 適用於 Google Cloud 後端的功能

瞭解如何使用提供的範例組態、將 NetApp Cloud Volumes Service for Google Cloud 設定為 Trident 安裝的後端。

Google Cloud 驅動程式詳細資料

Trident 提供 `gcp-cvs` 與叢集通訊的驅動程式。支援的存取模式包括：*ReadWriteOnce*（*rwo*）、*ReadOnlyMany*（*ROX*）、*_ReadWriteMany*（*rwx*）、*_ReadWriteOncePod*（*RWOP*）。

驅動程式	傳輸協定	Volume 模式	支援的存取模式	支援的檔案系統
《GCP—CVS》	NFS	檔案系統	Rwo、ROX、rwx、RWOP	nfs

瞭解 Trident 支援 Cloud Volumes Service for Google Cloud

Trident 可以在"服務類型"以下兩種中的其中一種中建立 Cloud Volumes Service Volume：

- ***CVS-Performance***：預設的 Trident 服務類型。這種效能最佳化的服務類型最適合重視效能的正式作業工作負載。CVS效能服務類型是一種硬體選項、可支援最小100 GiB大小的磁碟區。您可以選擇"三種服務層級"下列其中一項：
 - standard
 - premium
 - extreme
- ***CVS***：CVS服務類型提供高分區可用度、但效能等級僅限於中度。CVS服務類型是一種軟體選項、使用儲存資源池來支援小至1 GiB的磁碟區。儲存資源池最多可包含50個磁碟區、其中所有磁碟區都會共用資源池的容量和效能。您可以選擇其中一項"兩種服務層級"：
 - standardsw
 - zoneredundantstandardsw

您需要的產品

以設定及使用"適用於 Google Cloud Cloud Volumes Service"後端、您需要下列項目：

- Google Cloud帳戶已設定NetApp Cloud Volumes Service 功能
- Google Cloud帳戶的專案編號

- Google Cloud服務帳戶的角色為「netappcloudvolumes.admin」
- API金鑰檔案、供Cloud Volumes Service 您的I方面 帳戶使用

後端組態選項

每個後端都會在單一Google Cloud區域中配置磁碟區。若要在其他區域建立磁碟區、您可以定義其他後端。

參數	說明	預設
「分度」		永遠為1
「storageDriverName」	儲存驅動程式名稱	「GCP-CVS」
「後端名稱」	自訂名稱或儲存後端	驅動程式名稱+「_」+ API 金鑰的一部分
「storageClass」	用於指定CVS服務類型的選用參數。用於 software、選擇 CVS 服務類型。否則，Trident 將採用 CVS-Performance 服務類型(`hardware`)。	
storagePools	僅限CVS服務類型。選用參數、用於指定用於建立磁碟區的儲存資源池。	
「ProjectNumber」	Google Cloud帳戶專案編號。此值可在Google Cloud 入口網站首頁找到。	
「hostProjectNumber」	如果使用共享VPC網路、則為必要項目。在此案例中、projectNumber 是服務專案、以及 hostProjectNumber 是主機專案。	
《apiRegion》	Trident 建立 Cloud Volumes Service Volume 的 Google Cloud 區域。建立跨區域 Kubernetes 叢集時、在中建立的磁碟區`apiRegion`可用於跨多個 Google Cloud 區域的節點上排程的工作負載。跨區域流量會產生額外成本。	
「apiKey」	的Google Cloud服務帳戶API金鑰 netappcloudvolumes.admin 角色：其中包含Google Cloud服務帳戶私密金鑰檔案（逐字複製到後端組態檔）的JSON-格式內容。	
"proxyurl"	Proxy URL（如果需要Proxy伺服器才能連線至CVS帳戶）。Proxy伺服器可以是HTTP Proxy或HTTPS Proxy。對於HTTPS Proxy、會跳過憑證驗證、以允許在Proxy伺服器中使用自我簽署的憑證。不支援已啟用驗證的Proxy伺服器。	
「nfsMountOptions」	精細控制NFS掛載選項。	"nfsves=3"
《限制Volume大小》	如果要求的磁碟區大小高於此值、則資源配置失敗。	""（預設不強制執行）
《服務層級》	適用於新磁碟區的CVS效能或CVS服務層級。CVS的效能值為 standard、premium、或 `extreme`。CVS值包括 standardsw 或 zoneredundantstandardsw。	CVS效能預設為「標準」。CVS預設為「標準」。
網路	Google Cloud網路用於Cloud Volumes Service 解決資料不整的問題。	"預設"

參數	說明	預設
「DebugTraceFlags」	疑難排解時要使用的偵錯旗標。範例： \{"api":false, "method":true}。除非您正在進行疑難排解並需要詳細的記錄傾印、否則請勿使用此功能。	null
allowedTopologies	若要啟用跨區域存取、您的StorageClass定義適用於allowedTopologies 必須包含所有區域。例如： - key: topology.kubernetes.io/region values: - us-east1 - europe-west1	

Volume資源配置選項

您可以在中控制預設的Volume資源配置 defaults 組態檔的一節。

參數	說明	預設
「匯出規則」	新磁碟區的匯出規則。必須是以逗號分隔的清單、以CIDR表示法列出所有的IPv4位址或IPv4子網路組合。	「0.0.0.0/0」
「snapshotDir	存取「.snapshot」目錄	"假"
「快照保留區」	保留給快照的磁碟區百分比	""（接受CVS預設值為0）
《大小》	新磁碟區的大小。CVS效能最低為100 GiB。CVS最低為1 GiB。	CVS效能服務類型預設為「100GiB」。CVS服務類型並未設定預設值、但至少需要1 GiB。

CVS效能服務類型範例

下列範例提供CVS效能服務類型的範例組態。

範例1：最低組態

這是使用預設「標準」服務層級的預設CVS效能服務類型的最低後端組態。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
```

範例2：服務層級組態

本範例說明後端組態選項、包括服務層級和Volume預設值。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
proxyURL: http://proxy-server-hostname/
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 10Ti
serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '5'
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  size: 5Ti
```

範例3：虛擬資源池組態

此範例使用 `storage` 來設定虛擬集區和 `StorageClasses` 請回頭參考。請參閱 [\[儲存類別定義\]](#) 以瞭解如何定義儲存類別。

此處會針對所有設定的虛擬資源池設定特定的預設值 `snapshotReserve 5%`和 `exportRule 至 0.00.0/0`。虛擬資源池是在中定義的 `storage` 區段。每個個別虛擬集區都會定義自己的虛擬集區 `serviceLevel``和某些資源池會覆寫預設值。虛擬資源池標籤是用來區分資源池的依據 `performance` 和 `protection``。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
nfsMountOptions: vers=3,proto=tcp,timeo=600
defaults:
  snapshotReserve: '5'
  exportRule: 0.0.0.0/0
labels:
  cloud: gcp
region: us-west2
storage:
- labels:
  performance: extreme
  protection: extra
  serviceLevel: extreme
defaults:
```

```
    snapshotDir: 'true'
    snapshotReserve: '10'
    exportRule: 10.0.0.0/24
- labels:
  performance: extreme
  protection: standard
  serviceLevel: extreme
- labels:
  performance: premium
  protection: extra
  serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '10'
- labels:
  performance: premium
  protection: standard
  serviceLevel: premium
- labels:
  performance: standard
  serviceLevel: standard
```

儲存類別定義

下列StorageClass定義適用於虛擬集區組態範例。使用 `parameters.selector`、您可以為每個StorageClass指定用於裝載磁碟區的虛擬集區。該磁碟區會在所選的資源池中定義各個層面。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=standard"
```

```
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=extra"
allowVolumeExpansion: true
```

- 第一個StorageClass (cvs-extreme-extra-protection) 對應至第一個虛擬資源池。這是唯一提供極致效能、快照保留率為10%的資源池。
- Last StorageClass (cvs-extra-protection (最後一個 StorageClass) 調用任何提供 10% 快照保留的儲存池。Trident 會決定要選取哪個虛擬集區、並確保符合快照保留要求。

CVS服務類型範例

下列範例提供CVS服務類型的範例組態。

範例1：最低組態

這是使用的最低後端組態 storageClass 指定CVS服務類型和預設值 standardsw 服務層級：

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
storageClass: software
apiRegion: us-east4
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
serviceLevel: standardsw
```

範例2：儲存資源池組態

此範例後端組態使用 `storagePools` 以設定儲存資源池。

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
data.iam.gserviceaccount.com
  client_id: '107071413297115343396'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40cloud-native-data.iam.gserviceaccount.com
storageClass: software
zone: europe-west1-b
network: default
storagePools:
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509
serviceLevel: Standardsw
```

接下來呢？

建立後端組態檔之後、請執行下列命令：

```
tridentctl create backend -f <backend-file>
```

如果後端建立失敗、表示後端組態有問題。您可以執行下列命令來檢視記錄、以判斷原因：

```
tridentctl logs
```

識別並修正組態檔的問題之後、您可以再次執行create命令。

設定NetApp HCI 一個不只是功能的SolidFire 後端

瞭解如何在 Trident 安裝中建立和使用元素後端。

元素驅動程式詳細資料

Trident 提供 `solidfire-san` 儲存驅動程式以與叢集通訊。支援的存取模式包括：`ReadWriteOnce` (rwo)、`ReadOnlyMany` (ROX)、`_ReadWriteMany` (rwx)、`_ReadWriteOncePod` (RWOP)。

`solidfire-san` 儲存驅動程式支援 `_file_` 和 `_block_` 磁碟區模式。對於 `Filesystem` 的 `volemode`、Trident 會建立一個 Volume 並建立檔案系統。檔案系統類型由 `StorageClass` 指定。

驅動程式	傳輸協定	Volume 模式	支援的存取模式	支援的檔案系統
「olidfire - san」	iSCSI	區塊	Rwo、ROX、rwx、RWOP	無檔案系統。原始區塊裝置。
「olidfire - san」	iSCSI	檔案系統	RWO、RWOP	《xfs》、《ext3》、《ext4》

開始之前

在建立元素後端之前、您需要下列項目。

- 支援的儲存系統、可執行Element軟體。
- 提供給NetApp HCI / SolidFire叢集管理員或租戶使用者的認證、以管理磁碟區。
- 您所有的Kubernetes工作節點都應該安裝適當的iSCSI工具。請參閱 "[工作節點準備資訊](#)"。

後端組態選項

如需後端組態選項、請參閱下表：

參數	說明	預設
「分度」		永遠為1
「storageDriverName」	儲存驅動程式名稱	永遠是「solidfire-san」
「後端名稱」	自訂名稱或儲存後端	「S指_」+儲存設備 (iSCSI) IP位址SolidFire

參數	說明	預設
端點	MVIP、適用於SolidFire 採用租戶認證的不含用戶身分證明的叢集	
《VIP》	儲存設備 (iSCSI) IP位址和連接埠	
《標籤》	套用到磁碟區的任意JSON-格式化標籤集。	「」
《天王名稱》	要使用的租戶名稱 (如果找不到、請建立)	
《初始器IFACE》	將iSCSI流量限制在特定的主機介面	「預設」
《UseCHAP》	使用 CHAP 驗證 iSCSI。Trident 使用 CHAP。	是的
《存取群組》	要使用的存取群組ID清單	尋找名為「Trident」的存取群組ID
《類型》	QoS規格	
《限制Volume大小》	如果要求的磁碟區大小高於此值、則資源配置失敗	「」 (預設不強制執行)
「DebugTraceFlags」	疑難排解時要使用的偵錯旗標。範例： {"API":假、「方法」:true }	null



除非您正在進行疑難排解並需要詳細的記錄傾印、否則請勿使用「debugTraceFlags」。

範例1：的後端組態 `solidfire-san` 三種磁碟區類型的驅動程式

此範例顯示使用CHAP驗證的後端檔案、並建立具有特定QoS保證的三種Volume類型模型。您很可能會使用「IOPS」儲存類別參數來定義儲存類別、以使用每個類別。

```

---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: "<svip>:3260"
TenantName: "<tenant>"
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000

```

範例2：的後端與儲存類別組態 solidfire-san 驅動程式與虛擬資源池

此範例顯示使用虛擬資源池設定的後端定義檔、以及參照這些資源池的StorageClass。

Trident 會在資源配置時、將儲存池上的標籤複製到後端儲存 LUN。為了方便起見、儲存管理員可以針對每個虛擬資源池定義標籤、並依標籤將磁碟區分組。

在下圖所示的範例後端定義檔中、會針對所有設定的儲存資源池設定特定的預設值 type 銀級。虛擬資源池是在中定義的 storage 區段。在此範例中、有些儲存資源池會自行設定類型、有些資源池則會覆寫上述預設值。

```

---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: "<svip>:3260"
TenantName: "<tenant>"
UseCHAP: true
Types:

```

```

- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
- labels:
  performance: gold
  cost: '4'
  zone: us-east-1a
  type: Gold
- labels:
  performance: silver
  cost: '3'
  zone: us-east-1b
  type: Silver
- labels:
  performance: bronze
  cost: '2'
  zone: us-east-1c
  type: Bronze
- labels:
  performance: silver
  cost: '1'
  zone: us-east-1d

```

下列StorageClass定義是指上述虛擬資源池。使用 `parameters.selector` 欄位中、每個StorageClass會呼叫哪些虛擬資源池可用於裝載Volume。磁碟區將會在所選的虛擬資源池中定義各個層面。

第一個 StorageClass (`solidfire-gold-four`) 將映射到第一個虛擬池。這是唯一提供黃金級效能的集區 Volume Type QoS。Last StorageClass (`solidfire-silver` (最後一個 StorageClass) 調用任何提供銀牌

性能的存儲池。Trident 會決定要選取哪個虛擬集區、並確保符合儲存需求。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold; cost=4"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=3"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze; cost=2"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=1"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
  fsType: "ext4"
```

如需詳細資訊、請參閱

- ["Volume存取群組"](#)

支援SAN驅動程式ONTAP

ONTAP SAN 驅動程式概觀

深入瞭解如何使用ONTAP 支援功能的功能和功能性SAN驅動程式來設定功能性的後端。ONTAP Cloud Volumes ONTAP

ONTAP SAN 驅動程式詳細資料

Trident 提供下列 SAN 儲存驅動程式、可與 ONTAP 叢集進行通訊。支援的存取模式包括：*ReadWriteOnce*（*rwo*）、*ReadOnlyMany*（*ROX*）、*_ReadWriteMany*（*rwX*）、*_ReadWriteOncePod*（*RWOP*）。

驅動程式	傳輸協定	Volume模式	支援的存取模式	支援的檔案系統
「ONTAP-SAN」	iSCSI	區塊	Rwo、ROX、rwX、RWOP	無檔案系統；原始區塊裝置
「ONTAP-SAN」	iSCSI	檔案系統	RWO、RWOP 檔案系統磁碟區模式中無法使用 Rox 和 rwX。	《xfs》、《ext3》、《ext4》
「ONTAP-SAN」	NVMe / TCP 請參閱 NVMe / TCP 的其他考量事項 。	區塊	Rwo、ROX、rwX、RWOP	無檔案系統；原始區塊裝置
「ONTAP-SAN」	NVMe / TCP 請參閱 NVMe / TCP 的其他考量事項 。	檔案系統	RWO、RWOP 檔案系統磁碟區模式中無法使用 Rox 和 rwX。	《xfs》、《ext3》、《ext4》
《ONTAP-san經濟》	iSCSI	區塊	Rwo、ROX、rwX、RWOP	無檔案系統；原始區塊裝置
《ONTAP-san經濟》	iSCSI	檔案系統	RWO、RWOP 檔案系統磁碟區模式中無法使用 Rox 和 rwX。	《xfs》、《ext3》、《ext4》



- 使用 `ontap-san-economy` 只有持續磁碟區使用量計數預期會高於 "支援的 ONTAP Volume 限制"。
- 使用 `ontap-nas-economy` 只有持續磁碟區使用量計數預期會高於 "支援的 ONTAP Volume 限制" 和 `ontap-san-economy` 無法使用驅動程式。
- 請勿使用 `ontap-nas-economy` 如果您預期需要資料保護、災難恢復或行動性、

使用者權限

Trident 預期會以 ONTAP 或 SVM 管理員的身分執行、通常使用叢集使用者或 `vsadmin` SVM 使用者、或是使用 `admin`、具有相同角色的不同名稱的使用者。對於用於 NetApp ONTAP 部署的 Amazon FSX、Trident 預期會以 ONTAP 或 SVM 管理員的身分、使用叢集使用者或 `vsadmin` SVM 使用者、或是具有相同角色的不同名稱的使用者來執行 `fsxadmin`。`fsxadmin` 使用者只能有限地取代叢集管理使用者。



如果您使用此 `limitAggregateUsage` 參數、則需要叢集管理權限。將 Amazon FSX for NetApp ONTAP 搭配 Trident 使用時、此 `limitAggregateUsage` 參數將無法與和 `fsxadmin` 使用者帳戶搭配 `vsadmin` 使用。如果您指定此參數、組態作業將會失敗。

雖然可以在 ONTAP 中建立更具限制性的角色、讓 Trident 驅動程式可以使用、但我們不建議這樣做。Trident 的大多數新版本都會呼叫額外的 API、而這些 API 必須納入考量、使升級變得困難且容易出錯。

NVMe / TCP 的其他考量事項

Trident 支援使用驅動程式的非揮發性記憶體高速 (NVMe) 傳輸協定 `ontap-san`、包括：

- IPv6
- NVMe 磁碟區的快照和複本
- 調整 NVMe 磁碟區大小
- 匯入在 Trident 之外建立的 NVMe Volume、以便 Trident 管理其生命週期
- NVMe 原生多重路徑
- K8s 節點正常或不正常關機 (24.06)

Trident 不支援：

- NVMe 原生支援的 DH-HMAC-CHAP
- 裝置對應工具 (DM) 多重路徑
- LUKS 加密

準備使用 ONTAP 支援的 SAN 驅動程式來設定後端

瞭解使用 ONTAP SAN 驅動程式設定 ONTAP 後端的需求和驗證選項。

需求

對於所有 ONTAP 後端、Trident 至少需要指派一個 Aggregate 給 SVM。

請記住、您也可以執行多個驅動程式、並建立指向一個或多個驅動程式的儲存類別。例如、您可以設定使

用「ONTAP-SAN」驅動程式的「SAN開發」類別、以及使用「ONTAP-SAN經濟」類別的「SAN預設」類別。

您所有的Kubernetes工作節點都必須安裝適當的iSCSI工具。請參閱 "[準備工作節點](#)" 以取得詳細資料。

驗證 ONTAP 後端

Trident 提供兩種驗證 ONTAP 後端的模式。

- 認證型：ONTAP 對具備所需權限的使用者名稱和密碼。建議使用預先定義的安全登入角色、例如「admin」或「vsadmin」、以確保與ONTAP 各種版本的最大相容性。
- 憑證型：Trident 也可以使用安裝在後端的憑證與 ONTAP 叢集通訊。在此處、後端定義必須包含用戶端憑證、金鑰及信任的CA憑證（建議使用）的Base64編碼值。

您可以更新現有的後端、以便在認證型和憑證型方法之間移動。不過、一次只支援一種驗證方法。若要切換至不同的驗證方法、您必須從後端組態中移除現有方法。



如果您嘗試同時提供*認證與憑證*、後端建立將會失敗、並在組態檔中提供多種驗證方法。

啟用認證型驗證

Trident 需要 SVM 範圍 / 叢集範圍管理員的認證、才能與 ONTAP 後端通訊。建議您使用標準的預先定義角色、例如 admin、或 vsadmin。如此可確保與未來 ONTAP 版本的前移相容性、這些版本可能會公開未來 Trident 版本所使用的功能 API。自訂安全登入角色可建立並搭配 Trident 使用、但不建議使用。

後端定義範例如下所示：

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

請記住、後端定義是唯一以純文字儲存認證的位置。建立後端之後、使用者名稱/密碼會以Base64編碼、並儲存為Kubernetes機密。建立或更新後端是唯一需要具備認證知識的步驟。因此、這是一項純管理員操作、由Kubernetes /儲存管理員執行。

啟用憑證型驗證

新的和現有的後端可以使用憑證、並與ONTAP 該後端通訊。後端定義需要三個參數。

- 用戶端憑證：用戶端憑證的Base64編碼值。
- 用戶端私密金鑰：關聯私密金鑰的Base64編碼值。
- 信任的CACertificate：受信任CA憑證的Base64編碼值。如果使用信任的CA、則必須提供此參數。如果未使用信任的CA、則可忽略此問題。

典型的工作流程包括下列步驟。

步驟

1. 產生用戶端憑證和金鑰。產生時、請將Common Name (CN) (一般名稱 (CN)) 設定為ONTAP 驗證身分。

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. 將信任的CA憑證新增ONTAP 至整個叢集。這可能已由儲存管理員處理。如果未使用信任的CA、請忽略。

```
security certificate install -type server -cert-name <trusted-ca-cert-name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca <cert-authority>
```

3. 在ONTAP 支援叢集上安裝用戶端憑證和金鑰（步驟1）。

```
security certificate install -type client-ca -cert-name <certificate-name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. 確認ONTAP 支援「cert」驗證方法的支援功能。

```
security login create -user-or-group-name admin -application ontapi -authentication-method cert
security login create -user-or-group-name admin -application http -authentication-method cert
```

5. 使用產生的憑證測試驗證。以ONTAP Management LIF IP和SVM名稱取代<SfManagement LIF>和<vserver name>。

```
curl -X POST -Lk https://<ONTAP-Management-LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key --cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp xmlns="http://www.netapp.com/filer/admin" version="1.21" vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. 使用Base64編碼憑證、金鑰和信任的CA憑證。

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 使用從上一步取得的值建立後端。

```

cat cert-backend.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "SanBackend",
  "managementLIF": "1.2.3.4",
  "svm": "vserver_test",
  "clientCertificate": "Faaaakkkkeeee...Vaaalllluuueeee",
  "clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
  "trustedCACertificate": "QNFinfO...SiqOyN",
  "storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+-----+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |         0 |
+-----+-----+-----+-----+
+-----+-----+

```

更新驗證方法或旋轉認證資料

您可以更新現有的後端、以使用不同的驗證方法或旋轉其認證資料。這兩種方法都可行：使用使用者名稱/密碼的後端可更新以使用憑證；使用憑證的後端可更新為使用者名稱/密碼。若要這麼做、您必須移除現有的驗證方法、然後新增驗證方法。然後使用更新的backend.json檔案、其中包含執行「tridentctl後端更新」所需的參數。

```

cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |          UUID          |
STATE | VOLUMES |
+-----+-----+-----+
+-----+-----+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |      9 |
+-----+-----+-----+
+-----+-----+

```



當您旋轉密碼時、儲存管理員必須先更新ONTAP 使用者的密碼（位於BIOS）。接著是後端更新。在循環憑證時、可將多個憑證新增至使用者。然後更新後端以使用新的憑證、之後可從ONTAP 該叢集刪除舊的憑證。

更新後端不會中斷對已建立之磁碟區的存取、也不會影響之後建立的磁碟區連線。成功的後端更新表示 Trident 可以與 ONTAP 後端通訊、並處理未來的 Volume 作業。

為 Trident 建立自訂 ONTAP 角色

您可以使用最低 Privileges 來建立 ONTAP 叢集角色、這樣就不需要使用 ONTAP 管理員角色來執行 Trident 中的作業。當您在 Trident 後端組態中包含使用者名稱時、Trident 會使用您建立的 ONTAP 叢集角色來執行作業。

如需建立 Trident 自訂角色的詳細資訊、請參閱["Trident 自訂角色產生器"](#)。

使用 ONTAP CLI

1. 使用下列命令建立新角色：

```
security login role create <role_name\> -cmddirname "command" -access all  
-vserver <svm_name\>
```

2. 為 Trident 使用者建立使用者名稱：

```
security login create -username <user_name\> -application ontapi  
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver  
<svm_name\> -comment "user_description"
```

3. 將角色對應至使用者：

```
security login modify username <user_name\> -vserver <svm_name\> -role  
<role_name\> -application ontapi -application console -authmethod  
<password\>
```

使用 System Manager

在 ONTAP 系統管理員中執行下列步驟：

1. * 建立自訂角色 *：

- a. 若要在叢集層級建立自訂角色、請選取 * 叢集 > 設定 *。

(或) 若要在 SVM 層級建立自訂角色、請選取 * 儲存設備 > 儲存 VM >> required SVM 設定 > 使用者與角色 *。

- b. 選取 * 使用者和角色 * 旁的箭頭圖示 (* → *)。
- c. 在 * 角色 * 下選擇 **+Add**。
- d. 定義角色的規則、然後按一下 * 儲存 *。

2. * 將角色對應至 Trident 使用者 *：+ 在「* 使用者與角色 *」頁面上執行下列步驟：

- a. 在 * 使用者 * 下選取新增圖示 +。
- b. 選取所需的使用者名稱、然後在 * 角色 * 的下拉式功能表中選取角色。
- c. 按一下「* 儲存 *」。

如需詳細資訊、請參閱下列頁面：

- ["用於管理 ONTAP 的自訂角色"或"定義自訂角色"](#)
- ["與角色和使用者合作"](#)

使用雙向 CHAP 驗證連線

Trident 可以使用和 `ontap-san-economy` 驅動程式的雙向 CHAP 驗證 iSCSI 工作階段 `ontap-san`。這需要在後端定義中啟用 `useCHAP` 選項。設為 `true` 時、Trident 會將 SVM 的預設啟動器安全性設定為雙向 CHAP、並從後端檔案設定使用者名稱和密碼。NetApp 建議使用雙向 CHAP 來驗證連線。請參閱下列組態範例：

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLsd6cNwxyz
```



「useCHAP」參數是布林選項、只能設定一次。預設值設為假。將其設為true之後、您就無法將其設為假。

除了"useCHAP=true"之外、"chapInitiator Secret (chapInitiator機密)"、"chaptarketatorSecret (chaptarketusername)"、"chaptarketusername" (chaptargetuseamuse) 和"chapusername" (chamus在建立後端後端之後、可以執行「tridentctl update」來變更機密。

運作方式

儲存管理員會將設定 `useCHAP` 為 true 、指示 Trident 在儲存後端上設定 CHAP 。這包括下列項目：

- 在SVM上設定CHAP：
 - 如果 SVM 的預設啟動器安全性類型為無（預設為「無」） * 且 * 磁碟區中沒有預先存在的 LUN 、則 Trident 會將預設安全性類型設為 CHAP 、並繼續設定 CHAP 啟動器和目標使用者名稱和機密。
 - 如果 SVM 包含 LUN 、 Trident 將不會在 SVM 上啟用 CHAP 。這可確保不限制對 SVM 上已存在的 LUN 的存取。
- 設定CHAP啟動器和目標使用者名稱和機密；這些選項必須在後端組態中指定（如上所示）。

建立後端之後、Trident 會建立對應的 tridentbackend CRD 、並將 CHAP 機密和使用者名稱儲存為 Kubernetes 機密。Trident 在此後端建立的所有 PV 都會透過 CHAP 掛載及附加。

旋轉認證資料並更新後端

您可以更新「backend.json」檔案中的CHAP參數、以更新CHAP認證。這需要更新CHAP機密、並使用「tridentctl update」命令來反映這些變更。



更新後端的 CHAP 機密時、您必須使用 `tridentctl` 來更新後端。請勿透過 ONTAP UI 更新儲存叢集上的認證、因為 Trident 將無法取得這些變更。

```

cat backend-san.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "ontap_san_chap",
  "managementLIF": "192.168.0.135",
  "svm": "ontap_iscsi_svm",
  "useCHAP": true,
  "username": "vsadmin",
  "password": "password",
  "chapInitiatorSecret": "cl9qxUpDaTeD",
  "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLSD6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+-----+-----+-----+-----+
+-----+-----+
| NAME          | STORAGE DRIVER |          UUID          |
STATE | VOLUMES |
+-----+-----+-----+-----+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |          7 |
+-----+-----+-----+-----+
+-----+-----+

```

現有連線不會受到影響；如果 Trident 在 SVM 上更新認證、則這些連線將繼續保持作用中狀態。新的連線使用更新的認證資料、而現有的連線會繼續保持作用中。中斷舊PV的連線並重新連線、將會使用更新的認證資料。

SAN組態選項與範例ONTAP

瞭解如何在 Trident 安裝中建立及使用 ONTAP SAN 驅動程式。本節提供後端組態範例及將後端對應至 StorageClasses 的詳細資料。

後端組態選項

如需後端組態選項、請參閱下表：

參數	說明	預設
「分度」		永遠為1

參數	說明	預設
「storageDriverName」	儲存驅動程式名稱	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy
「後端名稱」	自訂名稱或儲存後端	驅動程式名稱 + "_" + dataLIF
《馬納格門達利》	叢集或 SVM 管理 LIF 的 IP 位址。您可以指定完整網域名稱 (FQDN)。如果使用 IPv6 旗標安裝 Trident、則可設定為使用 IPv6 位址。IPv6 位址必須以方括弧定義，例如 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。如需無縫 MetroCluster 之間的互通性[mcc-best]、請參閱。	「10.0.0.1」、 「[2001:1234:abcd:::fefo]」
「DataLIF」	傳輸協定LIF的IP位址。* 請勿指定 iSCSI。Trident 使用"可選擇的LUN對應ONTAP"來探索建立多重路徑工作階段所需的 iSCI 生命。如果明確定義、就會產生警告 dataLIF。MetroCluster 省略。*請參閱[mcc-best]。	源自SVM
《虛擬機器》	要使用的儲存虛擬機器 * MetroCluster 請省略。* 請參閱 [mcc-best]。	如果指定SVM "managementLIF"則衍生
《使用CHAP》	使用CHAP驗證iSCSI以供ONTAP 支援不支援的SAN驅動程式使用[布林值]。設為 true、讓 Trident 設定並使用雙向 CHAP 做為後端所指定 SVM 的預設驗證。如 "準備使用ONTAP 支援的SAN驅動程式來設定後端" 需詳細資訊、請參閱。	「假」
《chapInitiator機密》	CHAP啟動器密碼。如果是"useCHAP=true"、則為必要項目	"
《標籤》	套用到磁碟區的任意JSON-格式化標籤集	"
《chapTargetInitiator機密》	CHAP目標啟動器機密。如果是"useCHAP=true"、則為必要項目	"
「chapUsername」	傳入使用者名稱。如果是"useCHAP=true"、則為必要項目	"
《chapTargetUsername》	目標使用者名稱。如果是"useCHAP=true"、則為必要項目	"
「用戶端憑證」	用戶端憑證的Base64編碼值。用於憑證型驗證	"
「clientPrivate Key」	用戶端私密金鑰的Base64編碼值。用於憑證型驗證	"
「可信賴的CACertificate」	受信任CA憑證的Base64編碼值。選用。用於憑證型驗證。	"
《使用者名稱》	與ONTAP 該叢集通訊所需的使用者名稱。用於認證型驗證。	"
密碼	與ONTAP 該叢集通訊所需的密碼。用於認證型驗證。	"

參數	說明	預設
《虛擬機器》	要使用的儲存虛擬機器	如果指定SVM "managementLIF"則衍生
「storagePrefix」	在SVM中配置新磁碟區時所使用的前置碼。稍後無法修改。若要更新此參數、您需要建立新的後端。	trident
《Aggregate》	<p>用於資源配置的Aggregate（選用；如果已設定、則必須指派給SVM）。對於`ontap-nas-flexgroup`驅動程式、此選項會被忽略。如果未指派、任何可用的集合體都可用於佈建 FlexGroup Volume。</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;">  <p>在 SVM 中更新 Aggregate 時、它會透過輪詢 SVM 而無需重新啟動 Trident 控制器、在 Trident 中自動更新。當您在 Trident 中設定特定的 Aggregate 以配置 Volume 時、如果將 Aggregate 重新命名或移出 SVM、則在輪詢 SVM Aggregate 時、後端將會移至 Trident 中的失敗狀態。您必須將 Aggregate 變更為 SVM 上的 Aggregate、或是將其全部移除、才能使後端重新上線。</p> </div>	"
「限制Aggregateusage」	如果使用率高於此百分比、則無法進行資源配置。如果您使用 Amazon FSX for NetApp ONTAP 後端、請勿指定 limitAggregateUsage。提供的`fsxadmin`和`vsadmin`不包含使用 Trident 擷取彙總使用量並加以限制所需的權限。	""（預設不強制執行）
《限制Volume大小》	如果要求的磁碟區大小高於此值、則資源配置失敗。也會限制其管理 LUN 的最大磁碟區大小。	""（預設不會強制執行）
《lunsPerFlexvol》	每FlexVol 個LUN的最大LUN數量、範圍必須在[50、200]	100
「DebugTraceFlags」	<p>疑難排解時要使用的偵錯旗標。例如、 { "api" : false、 "method" : true}</p> <p>除非您正在進行疑難排解並需要詳細的記錄傾印、否則請勿使用。</p>	null
《useREST》	<p>使用ONTAP Isrest API的布林參數。useREST`設為`true`時、Trident 會使用 ONTAP REST API 與後端通訊；設為`false`時、Trident 會使用 ONTAP ZAPI 呼叫與後端通訊。此功能需要ONTAP 使用更新版本的版本。此外、使用的ONTAP 登入角色必須具有應用程式存取權`ontap`。這是預先定義的和角色所滿足 vsadmin cluster-admin 的。從 Trident 24.06 版本和 ONTAP 9.15.1 或更新版本開始、userREST`依預設會設定為`true；變更 useREST`為`false`使用 ONTAP ZAPI 呼叫。`useREST 完全符合 NVMe / TCP 的資格。</p>	true 對於 ONTAP 9.15.1 或更高版本，否則 false。

參數	說明	預設
sanType	用於選擇 iscsi iSCSI、nvme NVMe / TCP 或 fcp SCSI over Fibre Channel (FC)。*FCP (SCSI over FC) 是 Trident 24.10 版本的技術預覽功能。*	iscsi 如果空白
formatOptions	用於 formatOptions、指定命令的命令列引數、每當格式化磁碟區時都會套用這些引數、mkfs。這可讓您根據偏好設定來格式化 Volume。請務必指定與 mkfs 命令選項類似的格式選項、但不包括裝置路徑。範例： 「-E nobard」 • ontap-san ontap-san-economy 僅支援和驅動程式。*	
limitVolumePoolSize	在 ONTAP SAN 經濟型後端中使用 LUN 時、可要求的最大 FlexVol 大小。	"" (預設不強制執行)
denyNewVolumePools	限制 `ontap-san-economy` 後端建立新的 FlexVol 磁碟區以包含其 LUN。只有預先存在的 FlexVols 可用於佈建新的 PV。	

使用 formatOptions 的建議

Trident 建議使用下列選項來加速格式化程序：

*-E nobard : *

- 保留、請勿嘗試在 mkfs 時間捨棄區塊 (丟棄區塊一開始在固態裝置和稀疏 / 精簡配置儲存設備上很有用)。這會取代已過時的選項「-K」、而且適用於所有檔案系統 (xfs、ext3 和 ext4)。

用於資源配置磁碟區的后端組態選項

您可以使用中的這些選項來控制預設資源配置 defaults 組態區段。如需範例、請參閱下列組態範例。

參數	說明	預設
"paceAllocate (配置)"	LUN的空間分配	"對"
《保護區》	空間保留模式；「無」(精簡)或「Volume」(粗)	"無"
「快照原則」	要使用的Snapshot原則	"無"
「qosPolicy」	要指派給所建立磁碟區的QoS原則群組。選擇每個儲存集區/後端的其中一個qosPolicy或adaptiveQosPolicy。搭配 Trident 使用 QoS 原則群組需要 ONTAP 9.8 或更新版本。您應該使用非共用的 QoS 原則群組、並確保個別將原則群組套用至每個成員。共享 QoS 原則群組會強制執行所有工作負載總處理量的上限。	"
《adaptiveQosPolicy》	要指派給所建立磁碟區的調適性QoS原則群組。選擇每個儲存集區/後端的其中一個qosPolicy或adaptiveQosPolicy	"

參數	說明	預設
「快照保留區」	保留給快照的磁碟區百分比	「0」如果 snapshotPolicy 為「無」、否則為「」
「PlitOnClone」	建立複本時、從其父複本分割複本	"假"
加密	在新磁碟區上啟用 NetApp Volume Encryption (NVE) ；預設為 false 。必須在叢集上授權並啟用NVE 、才能使用此選項。如果在後端啟用 NAE 、則 Trident 中配置的任何 Volume 都將啟用 NAE 。如需更多資訊、請參閱" Trident 如何與 NVE 和 NAE 搭配運作 " ；。	"假"
luksEncryption	啟用LUKS加密。請參閱 " 使用Linux統一金鑰設定 (LUKS) "。 NVMe / TCP 不支援 LUKS 加密。	"
《生態樣式》	新磁碟區的安全樣式	unix
「分層政策」	分層原則以使用「無」	「僅限快照」適用於 ONTAP 9.5 之前的 SVM-DR 組態
nameTemplate	建立自訂磁碟區名稱的範本。	"

Volume資源配置範例

以下是定義預設值的範例：

```

---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'

```



對於使用驅動程式建立的所有磁碟區 `ontap-san`、Trident 會為 FlexVol 額外增加 10% 的容量、以容納 LUN 中繼資料。LUN 的配置大小與使用者在 PVC 中要求的大小完全相同。Trident 將 10% 新增至 FlexVol（在 ONTAP 中顯示為可用大小）。使用者現在可以取得所要求的可用容量。此變更也可防止 LUN 成為唯讀、除非可用空間已充分利用。這不適用於 ONTAP-san 經濟型。

對於定義的後端 `snapshotReserve`，Trident 將按以下方式計算卷的大小：

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

1.1 是額外 10% 的 Trident 新增至 FlexVol、以容納 LUN 中繼資料。若 `snapshotReserve = 5%`、且 PVC 要求 = 5GiB、則總 Volume 大小為 5.79GiB、可用大小為 5.5GiB。`volume show` 命令應顯示類似於此範例的結果：

Vserver	Volume	Aggregate	State	Type	Size	Available	Used%
		_pvc_89f1c156_3801_4de4_9f9d_034d54c395f4	online	RW	10GB	5.00GB	0%
		_pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d	online	RW	5.79GB	5.50GB	0%
		_pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba	online	RW	1GB	511.8MB	0%

3 entries were displayed.

目前、只有調整大小、才能將新計算用於現有的 Volume。

最低組態範例

下列範例顯示基本組態、讓大部分參數保留預設值。這是定義後端最簡單的方法。



如果您在 NetApp ONTAP 上搭配 Trident 使用 Amazon FSX、建議您指定生命週轉的 DNS 名稱、而非 IP 位址。

ONTAP SAN 範例

這是使用的基本組態 `ontap-san` 驅動程式：

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

ONTAP SAN 經濟效益範例

```
---  
version: 1  
storageDriverName: ontap-san-economy  
managementLIF: 10.0.0.1  
svm: svm_iscsi_eco  
username: vsadmin  
password: <password>
```

1. 範例

您可以設定後端、避免在切換和切換期間手動更新後端定義 "SVM 複寫與還原"。

若要無縫切換和切換、請使用指定 SVM managementLIF 並省略 dataLIF 和 svm 參數。例如：

```
---  
version: 1  
storageDriverName: ontap-san  
managementLIF: 192.168.1.66  
username: vsadmin  
password: password
```

憑證型驗證範例

在此基本組態範例中 `clientCertificate`、`clientPrivateKey` 和 `trustedCACertificate` (選用、如果使用信任的CA) 會填入 `backend.json` 並分別取得用戶端憑證、私密金鑰及信任CA憑證的基礎64編碼值。

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

雙向 CHAP 範例

這些範例使用建立後端 useCHAP 設定為 true。

ONTAP SAN CHAP 範例

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
```

ONTAP SAN 經濟 CHAP 範例

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
```

NVMe / TCP 範例

您必須在 ONTAP 後端上設定 NVMe 的 SVM 。這是適用於 NVMe / TCP 的基本後端組態。

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

名稱範本的后端組態範例

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults: {
  "nameTemplate":
    "{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.R
    equestName}}"
  },
  "labels": {"cluster": "ClusterA", "PVC":
    "{{.volume.Namespace}}_{{.volume.RequestName}}"}
}
```

formatOptions 範例： `ONTAP – San 經濟型` 驅動程式

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: ''
svm: svm1
username: ''
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: "-E nodiscard"
```

虛擬集區的后端範例

在這些后端定義檔案範例中、會針對所有儲存池設定特定的預設值、例如 `spaceReserve` 無、`spaceAllocation` 假、和 `encryption` 錯。虛擬資源池是在儲存區段中定義的。

Trident 會在「意見」欄位中設定資源配置標籤。請在 FlexVol The 過程中提出意見。Trident 會在資源配置時、將虛擬集區上的所有標籤複製到儲存磁碟區。為了方便起見、儲存管理員可以針對每個虛擬資源池定義標籤、並依標籤將磁碟區分組。

在這些範例中、有些儲存池是自行設定的 `spaceReserve`、`spaceAllocation` 和 `encryption` 值、而某些資源池會覆寫預設值。



```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
- labels:
  protection: gold
  creditpoints: '40000'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
    adaptiveQosPolicy: adaptive-extreme
- labels:
  protection: silver
  creditpoints: '20000'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
    qosPolicy: premium
- labels:
  protection: bronze
  creditpoints: '5000'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
```

```

---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
labels:
  store: san_economy_store
region: us_east_1
storage:
- labels:
  app: oracledb
  cost: '30'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
- labels:
  app: postgresdb
  cost: '20'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
- labels:
  app: mysqldb
  cost: '10'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
- labels:
  department: legal
  creditpoints: '5000'

```

```
zone: us_east_1c
defaults:
  spaceAllocation: 'true'
  encryption: 'false'
```

NVMe / TCP 範例

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: 'false'
  encryption: 'true'
storage:
- labels:
  app: testApp
  cost: '20'
  defaults:
    spaceAllocation: 'false'
    encryption: 'false'
```

將後端對應至StorageClass

下列 StorageClass 定義請參閱 [\[虛擬集區的後端範例\]](#)。使用 `parameters.selector` 欄位中、每個 StorageClass 都會呼叫哪些虛擬集區可用於主控磁碟區。磁碟區將會在所選的虛擬資源池中定義各個層面。

- `protection-gold` StorageClass 會對應至中的第一個虛擬集區 `ontap-san` 後端：這是唯一提供金級保護的集區。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- ◦ protection-not-gold StorageClass 會對應至中的第二個和第三個虛擬集區 ontap-san 後端：這是唯一提供金級以外保護層級的集區。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- ◦ app-mysqldb StorageClass 會對應至中的第三個虛擬集區 ontap-san-economy 後端：這是唯一為mysqldb 類型應用程式提供儲存池組態的集區。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- ◦ protection-silver-creditpoints-20k StorageClass 會對應至中的第二個虛擬集區 ontap-san 後端：這是唯一提供銀級保護和 20000 個信用點數的資源池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- creditpoints-5k StorageClass 會對應至中的第三個虛擬集區 ontap-san 中的後端和第四個虛擬集區 ontap-san-economy 後端：這是唯一擁有 5000 個信用點數的集區方案。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- my-test-app-sc StorageClass 會對應至 testAPP 中的虛擬集區 ontap-san 驅動程式搭配 sanType: nvme。這是唯一的集區服務項目 testApp。

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

Trident 會決定要選取哪個虛擬集區、並確保符合儲存需求。

ASNAS 驅動程式 ONTAP

ONTAP NAS 驅動程式概述

深入瞭解如何使用 ONTAP 功能性和功能性 NAS 驅動程式來設定功能性的後端。ONTAP Cloud Volumes ONTAP

Trident 提供下列 NAS 儲存驅動程式、可與 ONTAP 叢集進行通訊。支援的存取模式包括：*ReadWriteOnce*（*rwo*）、*ReadOnlyMany*（*ROX*）、*_ReadWriteMany*（*rwx*）、*_ReadWriteOncePod*（*RWOP*）。

驅動程式	傳輸協定	Volume模式	支援的存取模式	支援的檔案系統
「ONTAP-NAS」	NFS 中小企業	檔案系統	Rwo、ROX、rwx、RWOP	"、nfs、smb
《ONTAP-NANAS經濟》	NFS 中小企業	檔案系統	Rwo、ROX、rwx、RWOP	"、nfs、smb
「ONTAP-NAA-flexgroup」	NFS 中小企業	檔案系統	Rwo、ROX、rwx、RWOP	"、nfs、smb



- 使用 `ontap-san-economy` 只有持續磁碟區使用量計數預期會高於 "支援的 ONTAP Volume 限制"。
- 使用 `ontap-nas-economy` 只有持續磁碟區使用量計數預期會高於 "支援的 ONTAP Volume 限制" 和 `ontap-san-economy` 無法使用驅動程式。
- 請勿使用 `ontap-nas-economy` 如果您預期需要資料保護、災難恢復或行動性、

使用者權限

Trident 預期會以 ONTAP 或 SVM 管理員的身分執行、通常使用叢集使用者或 `vsadmin` SVM 使用者、或是使用 ``admin`` 具有相同角色的不同名稱的使用者。

對於用於 NetApp ONTAP 部署的 Amazon FSX、Trident 預期會以 ONTAP 或 SVM 管理員的身分、使用叢集使用者或 `vsadmin` SVM 使用者、或是具有相同角色的不同名稱的使用者來執行 `fsxadmin`。``fsxadmin`` 使用者只能有限地取代叢集管理使用者。



如果您使用此 ``limitAggregateUsage`` 參數、則需要叢集管理權限。將 Amazon FSX for NetApp ONTAP 搭配 Trident 使用時、此 ``limitAggregateUsage`` 參數將無法與 ``fsxadmin`` 使用者帳戶搭配 ``vsadmin`` 使用。如果您指定此參數、組態作業將會失敗。

雖然可以在 ONTAP 中建立更具限制性的角色、讓 Trident 驅動程式可以使用、但我們不建議這樣做。Trident 的大多數新版本都會呼叫額外的 API、而這些 API 必須納入考量、使升級變得困難且容易出錯。

準備使用 ONTAP 不含 NAS 的驅動程式來設定後端

瞭解使用 ONTAP NAS 驅動程式設定 ONTAP 後端的需求、驗證選項和匯出原則。

需求

- 對於所有 ONTAP 後端、Trident 至少需要指派一個 Aggregate 給 SVM。
- 您可以執行多個驅動程式、並建立指向其中一個或另一個的儲存類別。例如、您可以設定使用的 Gold 類別 `ontap-nas` 驅動程式和銅級、使用 `ontap-nas-economy` 一、

- 您所有的Kubernetes工作節點都必須安裝適當的NFS工具。請參閱 ["請按這裡"](#) 以取得更多詳細資料。
- Trident 僅支援掛載至 Windows 節點上執行的 Pod 的 SMB 磁碟區。如 [準備配置SMB磁碟區](#) 需詳細資訊、請參閱。

驗證 ONTAP 後端

Trident 提供兩種驗證 ONTAP 後端的模式。

- 認證型：此模式需要對 ONTAP 後端擁有足夠的權限。建議您使用與預先定義的安全登入角色相關聯的帳戶、例如 `admin` 或 `vsadmin` 以確保與ONTAP 更新版本的最大相容性。
- 憑證型：此模式需要在後端安裝憑證、Trident 才能與 ONTAP 叢集通訊。在此處、後端定義必須包含用戶端憑證、金鑰及信任的CA憑證（建議使用）的Base64編碼值。

您可以更新現有的後端、以便在認證型和憑證型方法之間移動。不過、一次只支援一種驗證方法。若要切換至不同的驗證方法、您必須從後端組態中移除現有方法。



如果您嘗試同時提供*認證與憑證*、後端建立將會失敗、並在組態檔中提供多種驗證方法。

啟用認證型驗證

Trident 需要 SVM 範圍 / 叢集範圍管理員的認證、才能與 ONTAP 後端通訊。建議您使用標準的預先定義角色、例如 `admin`` 或 ``vsadmin`。如此可確保與未來 ONTAP 版本的前移相容性、這些版本可能會公開未來 Trident 版本所使用的功能 API。自訂安全登入角色可建立並搭配 Trident 使用、但不建議使用。

後端定義範例如下所示：

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

請記住、後端定義是唯一以純文字儲存認證的位置。建立後端之後、使用者名稱/密碼會以Base64編碼、並儲存為Kubernetes機密。建立/更新後端是唯一需要知道認證資料的步驟。因此、這是一項純管理員操作、由Kubernetes /儲存管理員執行。

啟用憑證型驗證

新的和現有的後端可以使用憑證、並與ONTAP 該後端通訊。後端定義需要三個參數。

- 用戶端憑證：用戶端憑證的Base64編碼值。
- 用戶端私密金鑰：關聯私密金鑰的Base64編碼值。
- 信任的CACertificate：受信任CA憑證的Base64編碼值。如果使用信任的CA、則必須提供此參數。如果未使用信任的CA、則可忽略此問題。

典型的工作流程包括下列步驟。

步驟

1. 產生用戶端憑證和金鑰。產生時、請將Common Name (CN) (一般名稱 (CN)) 設定為ONTAP 驗證身分。

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. 將信任的CA憑證新增ONTAP 至整個叢集。這可能已由儲存管理員處理。如果未使用信任的CA、請忽略。

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. 在ONTAP 支援叢集上安裝用戶端憑證和金鑰（步驟1）。

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. 確認ONTAP 支援「cert」驗證方法的支援功能。

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. 使用產生的憑證測試驗證。以ONTAP Management LIF IP和SVM名稱取代<SfManagement LIF>和<vserver name>。您必須確保LIF的服務原則設定為「預設資料管理」。

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. 使用Base64編碼憑證、金鑰和信任的CA憑證。

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 使用從上一步取得的值建立後端。

```

cat cert-backend-updated.json
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "NasBackend",
  "managementLIF": "1.2.3.4",
  "dataLIF": "1.2.3.8",
  "svm": "vserver_test",
  "clientCertificate": "Faaaakkkkeeee...Vaaalllluuueeee",
  "clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
  "storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+
+-----+-----+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |         9 |
+-----+-----+-----+
+-----+-----+

```

更新驗證方法或旋轉認證資料

您可以更新現有的後端、以使用不同的驗證方法或旋轉其認證資料。這兩種方法都可行：使用使用者名稱/密碼的後端可更新以使用憑證；使用憑證的後端可更新為使用者名稱/密碼。若要這麼做、您必須移除現有的驗證方法、然後新增驗證方法。然後使用更新的backend.json檔案、其中包含要執行的必要參數 `tridentctl update backend`。

```

cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+-----+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |         9 |
+-----+-----+-----+-----+
+-----+-----+

```



當您旋轉密碼時、儲存管理員必須先更新ONTAP 使用者的密碼（位於BIOS）。接著是後端更新。在循環憑證時、可將多個憑證新增至使用者。然後更新後端以使用新的憑證、之後可從ONTAP 該叢集刪除舊的憑證。

更新後端不會中斷對已建立之磁碟區的存取、也不會影響之後建立的磁碟區連線。成功的後端更新表示 Trident 可以與 ONTAP 後端通訊、並處理未來的 Volume 作業。

為 Trident 建立自訂 ONTAP 角色

您可以使用最低 Privileges 來建立 ONTAP 叢集角色、這樣就不需要使用 ONTAP 管理員角色來執行 Trident 中的作業。當您在 Trident 後端組態中包含使用者名稱時、Trident 會使用您建立的 ONTAP 叢集角色來執行作業。

如需建立 Trident 自訂角色的詳細資訊、請參閱["Trident 自訂角色產生器"](#)。

使用 ONTAP CLI

1. 使用下列命令建立新角色：

```
security login role create <role_name\> -cmddirname "command" -access all  
-vserver <svm_name\>
```

2. 為 Trident 使用者建立使用者名稱：

```
security login create -username <user_name\> -application ontapi  
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver  
<svm_name\> -comment "user_description"
```

3. 將角色對應至使用者：

```
security login modify username <user_name\> -vserver <svm_name\> -role  
<role_name\> -application ontapi -application console -authmethod  
<password\>
```

使用 System Manager

在 ONTAP 系統管理員中執行下列步驟：

1. * 建立自訂角色 *：

- a. 若要在叢集層級建立自訂角色、請選取 * 叢集 > 設定 *。

(或) 若要在 SVM 層級建立自訂角色、請選取 * 儲存設備 > 儲存 VM >> required SVM 設定 > 使用者與角色 *。

- b. 選取 * 使用者和角色 * 旁的箭頭圖示 (* → *)。

- c. 在 * 角色 * 下選擇 **+Add**。

- d. 定義角色的規則、然後按一下 * 儲存 *。

2. * 將角色對應至 Trident 使用者 *：+ 在「* 使用者與角色 *」頁面上執行下列步驟：

- a. 在 * 使用者 * 下選取新增圖示 +。

- b. 選取所需的使用者名稱、然後在 * 角色 * 的下拉式功能表中選取角色。

- c. 按一下「* 儲存 *」。

如需詳細資訊、請參閱下列頁面：

- ["用於管理 ONTAP 的自訂角色"或"定義自訂角色"](#)
- ["與角色和使用者合作"](#)

管理 NFS 匯出原則

Trident 使用 NFS 匯出原則來控制對其所配置之磁碟區的存取。

Trident 在使用匯出原則時提供兩個選項：

- Trident 可以動態管理匯出原則本身；在此作業模式中、儲存管理員會指定代表可接受 IP 位址的 CIDR 區塊清單。Trident 會在發佈時自動將屬於這些範圍的適用節點 IP 新增至匯出原則。或者、如果未指定 CIDR、則在要發佈的磁碟區所在節點上找到的所有全域範圍單點傳播 IP 都會新增至匯出原則。
- 儲存管理員可以建立匯出原則、並手動新增規則。除非在組態中指定不同的匯出原則名稱、否則 Trident 會使用預設匯出原則。

動態管理匯出原則

Trident 提供動態管理 ONTAP 後端匯出原則的功能。這可讓儲存管理員為工作節點 IP 指定允許的位址空間、而非手動定義明確的規則。它可大幅簡化匯出原則管理；修改匯出原則不再需要在儲存叢集上進行手動介入。此外、這有助於將儲存叢集的存取限制在裝載磁碟區且指定範圍內有 IP 的工作節點、以支援精細且自動化的管理。



使用動態匯出原則時、請勿使用網路位址轉譯（NAT）。使用 NAT 時、儲存控制器會看到前端 NAT 位址、而非實際 IP 主機位址、因此在匯出規則中找不到相符項目時、就會拒絕存取。



在 Trident 24.10 中、`ontap-nas` 儲存驅動程式將繼續如舊版一樣運作；ONTAP NAS 驅動程式並未進行任何變更。只有儲存驅動程式才能 `ontap-nas-economy` 在 Trident 24.10 中進行 Volume 型精細存取控制。

範例

必須使用兩種組態選項。以下是後端定義範例：

```
---
version: 1
storageDriverName: ontap-nas-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
- 192.168.0.0/24
autoExportPolicy: true
```



使用此功能時、您必須確保 SVM 中的根連接點具有先前建立的匯出原則、並具有允許節點 CIDR 區塊（例如預設匯出原則）的匯出規則。請務必遵循 NetApp 建議的最佳實務做法、將 SVM 專用於 Trident。

以下是使用上述範例說明此功能的運作方式：

- `autoExportPolicy` 設定為 `true`。這表示 Trident 會為使用此後端為 SVM 佈建的每個 Volume 建立匯出原則 `svm1`、並使用位址區塊來處理規則的新增和刪除 `autoexportCIDRs`。在磁碟區附加至節點之前、該磁碟區會使用沒有規則的空匯出原則、以防止不必要的存取該磁碟區。當磁碟區發佈至節點 Trident 時、會建立一個匯出原則、其名稱與包含指定 CIDR 區塊內節點 IP 的基礎 `qtree` 相同。這些 IP 也會新增至父 FlexVol 所使用的匯出原則。

◦ 例如：

- 後端 UUID 403b5326-8482-40der-96d0-d83fb3f4daec
- autoExportPolicy 設定為 true
- 儲存字首 trident
- PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
- qtree 名稱為 Trident_PVC_a79bcf5f_7b6d_4a40_9876_e2551f159c1c FlexVol、會為命名的 qtree 建立匯出原則、為命名的 qtree 建立匯 trident-403b5326-8482-40db96d0-d83fb3f4daec 出原則、
trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c 以及在 SVM 上命名的空白匯出原則 trident_empty。FlexVol 匯出原則的規則將是 qtree 匯出原則所包含的任何規則的超集。未附加的任何磁碟區都會重複使用空的匯出原則。

- autoExportCIDRs 包含位址區塊清單。此欄位為選用欄位、預設為「0.00.0.0/0」、「:/0」。如果未定義、Trident 會新增所有在工作節點上找到的全域範圍單點傳播位址、並提供出版物。

在此範例中 192.168.0.0/24、會提供位址空間。這表示位於此位址範圍內的 Kubernetes 節點 IP 與出版物將會新增至 Trident 所建立的匯出原則。當 Trident 登錄其執行的節點時，它會擷取節點的 IP 位址，並對照中提供的位址區塊進行檢查 autoExportCIDRs。在發佈時，在篩選 IP 之後，Trident 會為其所發佈節點的用戶端 IP 建立匯出原則規則。

您可以在建立後端後、更新「AutoExportPolicy」和「AutoExportCTR」。您可以為自動管理或刪除現有CIDR的後端附加新的CIDR。刪除CIDR時請務必謹慎、以確保不會中斷現有的連線。您也可以選擇停用後端的「autodportPolicy」、然後回到手動建立的匯出原則。這需要在後端組態中設定「exportPolicy」參數。

Trident 建立或更新後端之後、您可以使用或對應的 tridentbackend CRD 來檢查後端 tridentctl：

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileType: ext4
```

移除節點時、Trident 會檢查所有匯出原則、以移除對應於節點的存取規則。透過從受管理後端的匯出原則中移除此節點 IP、Trident 可防止惡意掛載、除非叢集中的新節點重複使用此 IP。

對於先前存在的後端、使用更新後端 `tridentctl update backend` 可確保 Trident 自動管理匯出原則。這會在需要時建立兩個以後端 UUID 和 qtree 名稱命名的新匯出原則。後端上的磁碟區會在新建立的匯出原則卸載並重新掛載之後、使用這些原則。



刪除具有自動管理匯出原則的後端、將會刪除動態建立的匯出原則。如果重新建立後端、則會將其視為新的後端、並導致建立新的匯出原則。

如果即時節點的 IP 位址已更新、您必須在節點上重新啟動 Trident Pod。然後 Trident 會更新匯出原則、以反映其所管理的 IP 變更。

準備配置SMB磁碟區

只需稍加準備、您就可以使用來配置 SMB 磁碟區 `ontap-nas` 驅動程式：



您必須在 SVM 上同時設定 NFS 和 SMB/CIFS 通訊協定、才能建立 `ontap-nas-economy` 適用於內部部署 ONTAP 的 SMB Volume。若未設定上述任一種通訊協定、將導致 SMB 磁碟區建立失敗。



`autoExportPolicy` 不支援 SMB Volume。

開始之前

在配置 SMB 磁碟區之前、您必須具備下列項目。

- Kubernetes叢集具備Linux控制器節點、以及至少一個執行Windows Server 2022的Windows工作節點。Trident 僅支援掛載至 Windows 節點上執行的 Pod 的 SMB 磁碟區。
- 至少有一個 Trident 機密包含您的 Active Directory 認證。產生機密 `smbcreds`：

```
kubectl create secret generic smbcreds --from-literal username=user  
--from-literal password='password'
```

- 設定為Windows服務的SCSI Proxy。若要設定 `csi-proxy`、請參閱 "[GitHub：csi Proxy](#)" 或 "[GitHub：適用於Windows的SCSI Proxy](#)" 適用於Windows上執行的Kubernetes節點。

步驟

1. 對於內部部署 ONTAP、您可以選擇性地建立 SMB 共用、或 Trident 可以為您建立 SMB 共用。



Amazon FSX for ONTAP 需要 SMB 共享。

您可以使用兩種方式之一來建立SMB管理共用區 "[Microsoft管理主控台](#)" 共享資料夾嵌入式管理單元或使用ONTAP CLI。若要使用ONTAP CLI建立SMB共用：

- a. 如有必要、請建立共用的目錄路徑結構。

◦ `vserver cifs share create` 命令會在共用建立期間檢查 `-path` 選項中指定的路徑。如果指定的

路徑不存在、則命令會失敗。

b. 建立與指定SVM相關的SMB共用區：

```
vserver cifs share create -vserver vserver_name -share-name  
share_name -path path [-share-properties share_properties,...]  
[other_attributes] [-comment text]
```

c. 確認共用區已建立：

```
vserver cifs share show -share-name share_name
```



請參閱 "建立SMB共用區" 以取得完整詳細資料。

2. 建立後端時、您必須設定下列項目以指定SMB Volume。如需ONTAP 所有的FSXfor Sendbackend組態選項、請參閱 "FSX提供ONTAP 各種組態選項和範例"。

參數	說明	範例
smbShare	您可以指定下列其中一項：使用 Microsoft 管理主控台或 ONTAP CLI 建立的 SMB 共用名稱；允許 Trident 建立 SMB 共用的名稱；或將參數保留空白以防止共用磁碟區。對於內部部署 ONTAP、此參數為選用項目。Amazon FSX 需要此參數才能支援 ONTAP 後端、且不可為空白。	smb-share
nasType	*必須設定為 smb.*如果為null、則預設為 nfs。	smb
《生態樣式》	新磁碟區的安全樣式。必須設定為 ntfs 或 mixed 適用於 SMB 磁碟區。	ntfs 或 mixed 適用於SMB磁碟區
「unixPermissions」	新磁碟區的模式。SMB磁碟區*必須保留為空白。*	"

列舉NAS組態選項與範例ONTAP

瞭解如何在 Trident 安裝中建立及使用 ONTAP NAS 驅動程式。本節提供後端組態範例及將後端對應至 StorageClasses 的詳細資料。

後端組態選項

如需後端組態選項、請參閱下表：

參數	說明	預設
「分度」		永遠為1
「storageDriverName」	儲存驅動程式名稱	「ONTAP - NAS」、 「ONTAP - NAS - 經濟」、 「ONTAP - NAS - Flexgroup」、 「ONTAP - SAN」、 「ONTAP - SAN 經濟」

參數	說明	預設
「後端名稱」	自訂名稱或儲存後端	驅動程式名稱 + "_" + dataLIF
《馬納格門達利》	叢集或 SVM 管理 LIF 的 IP 位址可以指定完整網域名稱 (FQDN)。如果使用 IPv6 旗標安裝 Trident、則可設定為使用 IPv6 位址。IPv6 位址必須以方括弧定義，例如 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。如需無縫 MetroCluster 之間的互通性[mcc-best]、請參閱。	「10.0.0.1」、"[2001:1234:abcd:::fefo]"
「DataLIF」	傳輸協定LIF的IP位址。建議您指定 dataLIF。如果未提供、Trident 會從 SVM 擷取資料生命。您可以指定要用於NFS掛載作業的完整網域名稱 (FQDN)、讓您建立循環配置資源DNS、以便在多個資料生命期之間達到負載平衡。可在初始設定之後變更。請參閱。如果使用 IPv6 旗標安裝 Trident、則可設定為使用 IPv6 位址。IPv6 位址必須以方括弧定義，例如 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。* MetroCluster 省略。*請參閱[mcc-best]。	指定位址或從SVM衍生 (若未指定) (不建議使用)
《虛擬機器》	要使用的儲存虛擬機器 * MetroCluster 請省略。* 請參閱 [mcc-best]。	如果指定SVM "managementLIF"則衍生
「AutoExpportPolicy」	啟用自動匯出原則建立及更新[布林值]。使用 `autoExportPolicy` 和 `autoExportCIDRs` 選項、Trident 可以自動管理匯出原則。	錯
《AutoExpportCIDR》(自動匯出CTR)	將 Kubernetes 節點 IP 篩選在啟用時的 CIDR 清單 autoExportPolicy。使用 `autoExportPolicy` 和 `autoExportCIDRs` 選項、Trident 可以自動管理匯出原則。	["0.0.0/0"、":/0"]
《標籤》	套用到磁碟區的任意JSON-格式化標籤集	"
「用戶端憑證」	用戶端憑證的Base64編碼值。用於憑證型驗證	"
「clientPrivate Key」	用戶端私密金鑰的Base64編碼值。用於憑證型驗證	"
「可信賴的CACertificate」	受信任CA憑證的Base64編碼值。選用。用於憑證型驗證	"
《使用者名稱》	連線至叢集/ SVM的使用者名稱。用於認證型驗證	
密碼	連線至叢集/ SVM的密碼。用於認證型驗證	
「storagePrefix」	在SVM中配置新磁碟區時所使用的前置碼。設定後無法更新  使用 ONTAP NAS 經濟型和 24 個以上字元的 storagePrefix 時，qtree 將不會內嵌儲存前置字元，不過它會位於磁碟區名稱中。	" Trident "

參數	說明	預設
《Aggregate》	<p>用於資源配置的Aggregate（選用；如果已設定、則必須指派給SVM）。對於`ontap-nas-flexgroup`驅動程式、此選項會被忽略。如果未指派、任何可用的集合體都可用於佈建 FlexGroup Volume。</p> <p> 在 SVM 中更新 Aggregate 時、它會透過輪詢 SVM 而無需重新啟動 Trident 控制器、在 Trident 中自動更新。當您在 Trident 中設定特定的 Aggregate 以配置 Volume 時、如果將 Aggregate 重新命名或移出 SVM、則在輪詢 SVM Aggregate 時、後端將會移至 Trident 中的失敗狀態。您必須將 Aggregate 變更為 SVM 上的 Aggregate、或是將其全部移除、才能使後端重新上線。</p>	"
「限制Aggregateusage」	<p>如果使用率高於此百分比、則無法進行資源配置。*不適用於Amazon FSX for ONTAP Sfor Sfor *</p>	""（預設不強制執行）
FlexgroupAggregateList	<p>用於資源配置的集合體清單（選用；如果已設定、則必須指派給 SVM）。指派給 SVM 的所有集合體都會用於佈建 FlexGroup Volume。支援 * ONTAP NAS FlexGroup * 儲存驅動程式。</p> <p> 在 SVM 中更新 Aggregate 清單時、會透過輪詢 SVM 而無需重新啟動 Trident 控制器、自動在 Trident 中更新清單。當您在 Trident 中設定特定的 Aggregate 清單來配置 Volume 時、如果將 Aggregate 清單重新命名或移出 SVM、則在輪詢 SVM Aggregate 時、後端將會移至 Trident 中的失敗狀態。您必須將 Aggregate 清單變更為 SVM 上的集合清單、或是將其全部移除以使後端重新上線。</p>	"
《限制Volume大小》	<p>如果要求的磁碟區大小高於此值、則資源配置失敗。也會限制其管理 qtree 的最大磁碟區大小、且此`qtreesPerFlexvol`選項可讓您自訂每個 FlexVol 的最大 qtree 數量。</p>	""（預設不會強制執行）
「DebugTraceFlags」	<p>疑難排解時要使用的偵錯旗標。例如、 { "api" : false、 "method" : true }</p> <p>請勿使用 debugTraceFlags 除非您正在疑難排解並需要詳細的記錄傾印。</p>	null
nasType	<p>設定NFS或SMB磁碟區建立。選項包括 nfs、 smb 或null。NFS磁碟區的預設值設為null。</p>	nfs

參數	說明	預設
「nfsMountOptions」	以逗號分隔的NFS掛載選項清單。Kubernetes-Persistent Volume 的掛載選項通常是在儲存類別中指定、但如果儲存類別中未指定掛載選項、則 Trident 會回復為使用儲存後端組態檔案中指定的掛載選項。如果儲存類別或組態檔案中未指定任何掛載選項、Trident 將不會在關聯的持續磁碟區上設定任何掛載選項。	"
"qtreesPerFlexvol"	每FlexVol 個邊的最大qtree數、必須在範圍內[50、300]	"200"
smbShare	您可以指定下列其中一項：使用 Microsoft 管理主控台或 ONTAP CLI 建立的 SMB 共用名稱；允許 Trident 建立 SMB 共用的名稱；或將參數保留空白以防止共用磁碟區。對於內部部署 ONTAP、此參數為選用項目。Amazon FSX 需要此參數才能支援 ONTAP 後端、且不可為空白。	smb-share
《useREST》	使用ONTAP Isrest API的布林參數。useREST` 設為 `true` 時、Trident 會使用 ONTAP REST API 與後端通訊；設為 `false` 時、Trident 會使用 ONTAP ZAPI 呼叫與後端通訊。此功能需要ONTAP 使用更新版本的版本。此外、使用的 ONTAP 登入角色必須具有應用程式存取權 `ontap`。這是預先定義的和角色所滿足 vsadmin cluster-admin 的。從 Trident 24.06 版本和 ONTAP 9.15.1 或更新版本開始、userREST` 依預設會設定為 `true`；變更 `useREST` 為 `false` 使用 ONTAP ZAPI 呼叫。	true 對於 ONTAP 9.15.1 或更高版本，否則 false。
limitVolumePoolSize	在 ONTAP NAS 經濟型後端使用 qtree 時、可要求的 FlexVol 大小上限。	"" (預設不強制執行)
denyNewVolumePools	限制 `ontap-nas-economy` 後端建立新的 FlexVol 磁碟區以包含其 qtree。只有預先存在的 FlexVols 可用於佈建新的 PV。	

用於資源配置磁碟區的後端組態選項

您可以使用中的這些選項來控制預設資源配置 defaults 組態區段。如需範例、請參閱下列組態範例。

參數	說明	預設
"paceAllocate (配置)"	qtree 的空間分配	"對"
《保護區》	空間保留模式；「無」（精簡）或「Volume」（粗）	"無"
「快照原則」	要使用的Snapshot原則	"無"
「qosPolicy」	要指派給所建立磁碟區的QoS原則群組。選擇每個儲存集區/後端的其中一個qosPolicy或adaptiveQosPolicy	"
《adaptiveQosPolicy》	要指派給所建立磁碟區的調適性QoS原則群組。選擇每個儲存集區/後端的其中一個qosPolicy或adaptiveQosPolicy。不受ONTAP-NAS-經濟支援。	"

參數	說明	預設
「快照保留區」	保留給快照的磁碟區百分比	「0」如果 snapshotPolicy 為「無」、否則為「」
「PlitOnClone」	建立複本時、從其父複本分割複本	"假"
加密	在新磁碟區上啟用 NetApp Volume Encryption (NVE) ；預設為 false 。必須在叢集上授權並啟用NVE、才能使用此選項。如果在後端啟用 NAE、則 Trident 中配置的任何 Volume 都將啟用 NAE 。如需更多資訊、請參閱" Trident 如何與 NVE 和 NAE 搭配運作 "： ：	"假"
「分層政策」	分層原則以使用「無」	「僅限快照」適用於 ONTAP 9.5 之前的 SVM-DR 組態
「unixPermissions」	新磁碟區的模式	"777" 表示 NFS 磁碟區；SMB 磁碟區為空的（不適用）
「snapshotDir」	控制對的存取 .snapshot 目錄	針對 NFSv3 的 NFSv4 "false" 為 "true"
「匯出政策」	要使用的匯出原則	"預設"
《生態樣式》	新磁碟區的安全樣式。NFS支援 mixed 和 unix 安全樣式；SMB支援 mixed 和 ntfs 安全樣式：	NFS預設為 unix 。SMB預設為 ntfs 。
nameTemplate	建立自訂磁碟區名稱的範本。	"



搭配 Trident 使用 QoS 原則群組需要 ONTAP 9.8 或更新版本。您應該使用非共用的 QoS 原則群組、並確保個別將原則群組套用至每個成員。共享 QoS 原則群組會強制執行所有工作負載總處理量的上限。

Volume 資源配置範例

以下是定義預設值的範例：

```

---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: <password>
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: '10'

```

對於 `ontap-nas` 和 `ontap-nas-flexgroups`、Trident 現在使用新的計算方式、確保 FlexVol 的大小正確、並使用 `snapshotReserve` 百分比和 PVC。當使用者要求使用 PVC 時、Trident 會使用新計算來建立具有更多空間的原始 FlexVol。此計算可確保使用者在永久虛擬磁碟中獲得所要求的可寫入空間、且空間不得小於所要求的空間。在 v21.07 之前、當使用者要求使用 PVC（例如 5GiB）、快照保留區達到 50% 時、他們只能獲得 2.5GiB 的可寫入空間。這是因為使用者所要求的是整個 Volume、而且 `snapshotReserve` 是其中的百分比。使用 Trident 21.07 時、使用者要求的是可寫入空間、而 Trident 則將該數量定義 `snapshotReserve` 為整個 Volume 的百分比。這不適用於 `ontap-nas-economy`。請參閱下列範例以瞭解此功能的運作方式：

計算方式如下：

```

Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)

```

對於 `snapshotReserve = 50%`、而 PVC 要求 = 5GiB、磁碟區總大小為 $2/0.5 = 10\text{GiB}$ 、可用大小為 5GiB、這是使用者在 PVC 要求中要求的大小。「`volume show (Volume show)`」命令應顯示類似以下範例的結果：

Vserver	Volume	Aggregate	State	Type	Size	Available	Used%
	_pvc_89f1c156_3801_4de4_9f9d_034d54c395f4		online	RW	10GB	5.00GB	0%
	_pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba		online	RW	1GB	511.8MB	0%

2 entries were displayed.

升級 Trident 時、先前安裝的現有後端將會如前文所述配置磁碟區。對於在升級之前建立的磁碟區、您應該調整其磁碟區大小、以便觀察變更。例如、使用較早版本的 2GiB PVC 會產生一個提供 1GiB snapshotReserve=50 可寫入空間的 Volume。例如、將磁碟區大小調整為3GiB、可讓應用程式在6 GiB磁碟區上擁有3GiB的可寫入空間。

最低組態範例

下列範例顯示基本組態、讓大部分參數保留預設值。這是定義後端最簡單的方法。



如果您在NetApp ONTAP 支援Trident的NetApp支援上使用Amazon FSX、建議您指定lifs的DNS名稱、而非IP位址。

ONTAP NAS 經濟效益範例

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

ONTAP NAS FlexGroup 範例

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

MetroCluster 範例

您可以設定後端、避免在切換和切換期間手動更新後端定義 "SVM 複寫與還原"。

若要無縫切換和切換、請使用指定 SVM managementLIF 並省略 dataLIF 和 svm 參數。例如：

```
---  
version: 1  
storageDriverName: ontap-nas  
managementLIF: 192.168.1.66  
username: vsadmin  
password: password
```

SMB Volume 範例

```
---  
version: 1  
backendName: ExampleBackend  
storageDriverName: ontap-nas  
managementLIF: 10.0.0.1  
nasType: smb  
securityStyle: ntfs  
unixPermissions: ""  
dataLIF: 10.0.0.2  
svm: svm_nfs  
username: vsadmin  
password: password
```

憑證型驗證範例

這是最小的後端組態範例。`clientCertificate`、`clientPrivateKey`和`trustedCACertificate`（選用、如果使用信任的CA）會填入`backend.json`並分別取得用戶端憑證、私密金鑰及信任CA憑證的基礎64編碼值。

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

自動匯出原則範例

本範例說明如何指示 Trident 使用動態匯出原則來自動建立及管理匯出原則。和`ontap-nas-flexgroup`驅動程式的運作方式相同`ontap-nas-economy`。

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

IPv6 位址範例

此範例顯示 managementLIF 使用 IPv6 位址。

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

Amazon FSX for ONTAP 使用 SMB Volume 範例

- smbShare 使用 SMB 磁碟區的 ONTAP 需要 FSX 參數。

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

名稱範本的后端組態範例

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults: {
  "nameTemplate":
    "{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.R
    equestName}}"
  },
  "labels": {"cluster": "ClusterA", "PVC":
    "{{.volume.Namespace}}_{{.volume.RequestName}}"}
}
```

虛擬集區的后端範例

在下面顯示的后端定義檔案範例中、會針對所有儲存池設定特定的預設值、例如 `spaceReserve` 無、`spaceAllocation` 假、和 `encryption` 錯。虛擬資源池是在儲存區段中定義的。

Trident 會在「意見」欄位中設定資源配置標籤。註解是在 `FlexVol for` 或 `FlexGroup for ontap-nas-flexgroup` 上設定 `ontap-nas`。Trident 會在資源配置時、將虛擬集區上的所有標籤複製到儲存磁碟區。為了方便起見、儲存管理員可以針對每個虛擬資源池定義標籤、並依標籤將磁碟區分組。

在這些範例中、有些儲存池是自行設定的 `spaceReserve`、`spaceAllocation` 和 `encryption` 值、而某些資源池會覆寫預設值。

```

---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: 'false'
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
  app: msoffice
  cost: '100'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
    adaptiveQosPolicy: adaptive-premium
- labels:
  app: slack
  cost: '75'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  department: legal
  creditpoints: '5000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:

```

```
  app: wordpress
  cost: '50'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
  app: mysqldb
  cost: '25'
  zone: us_east_1d
  defaults:
    spaceReserve: volume
    encryption: 'false'
    unixPermissions: '0775'
```

ONTAP NAS FlexGroup 範例

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
  protection: gold
  creditpoints: '50000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  protection: gold
  creditpoints: '30000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  protection: silver
  creditpoints: '20000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
  protection: bronze
  creditpoints: '10000'
  zone: us_east_1d
```

```
defaults:  
  spaceReserve: volume  
  encryption: 'false'  
  unixPermissions: '0775'
```

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: nas_economy_store
region: us_east_1
storage:
- labels:
  department: finance
  creditpoints: '6000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  protection: bronze
  creditpoints: '5000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  department: engineering
  creditpoints: '3000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
  department: humanresource
  creditpoints: '2000'
  zone: us_east_1d
  defaults:
```

```
spaceReserve: volume
encryption: 'false'
unixPermissions: '0775'
```

將後端對應至StorageClass

請參閱下列 StorageClass 定義 [\[虛擬集區的后端範例\]](#)。使用 `parameters.selector` 欄位中、每個 StorageClass 都會呼叫哪些虛擬集區可用於主控磁碟區。磁碟區將會在所選的虛擬資源池中定義各個層面。

- `protection-gold` StorageClass 會對應至中的第一個和第二個虛擬集區 `ontap-nas-flexgroup` 後端：這是唯一提供金級保護的資源池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- `protection-not-gold` StorageClass 會對應至中的第三和第四個虛擬集區 `ontap-nas-flexgroup` 後端：這是唯一提供金級以外保護層級的資源池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- `app-mysqldb` StorageClass 會對應至中的第四個虛擬集區 `ontap-nas` 後端：這是唯一為 `mysqldb` 類型應用程式提供儲存池組態的集區。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- `tprotection-silver-creditpoints-20k` StorageClass 會對應至中的第三個虛擬集區 `ontap-nas-flexgroup` 後端：這是唯一提供銀級保護和 20000 個信用點數的資源池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- `creditpoints-5k` StorageClass 會對應至中的第三個虛擬集區 `ontap-nas` 後端和中的第二個虛擬集區 `ontap-nas-economy` 後端：這是唯一擁有 5000 個信用點數的集區方案。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Trident 會決定要選取哪個虛擬集區、並確保符合儲存需求。

更新 dataLIF 初始組態之後

您可以在初始組態後變更資料LIF、方法是執行下列命令、以更新資料LIF提供新的後端Json檔案。

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-with-updated-dataLIF>
```



如果將PVCS附加至一或多個Pod、您必須關閉所有對應的Pod、然後將其重新啟動、新的資料LIF才會生效。

Amazon FSX for NetApp ONTAP 產品

搭配 Amazon FSX for NetApp ONTAP 使用 Trident

"Amazon FSX for NetApp ONTAP 產品" 是完全託管的AWS服務、可讓客戶啟動及執行採用NetApp ONTAP 資訊儲存作業系統的檔案系統。FSX for ONTAP VMware可讓您運用熟悉的NetApp功能、效能和管理功能、同時充分發揮儲存AWS資料的簡易性、敏捷度、安全性和擴充性。FSX for ONTAP Sfor支援ONTAP Isf供 檔案系統功能和管理API。

您可以將 Amazon FSX for NetApp ONTAP 檔案系統與 Trident 整合、以確保在 Amazon Elastic Kubernetes Service (EKS) 中執行的 Kubernetes 叢集可配置由 ONTAP 備份的區塊和檔案持續磁碟區。

檔案系統是Amazon FSX的主要資源、類似ONTAP 於內部部署的一個叢集。在每個SVM中、您可以建立一個或多個磁碟區、這些磁碟區是儲存檔案系統中檔案和資料夾的資料容器。有了Amazon FSX for NetApp ONTAP 的功能、Data ONTAP 即可在雲端以託管檔案系統的形式提供支援。新的檔案系統類型稱為* NetApp ONTAP Sing*。

使用 Trident 搭配 Amazon FSX for NetApp ONTAP、您可以確保在 Amazon Elastic Kubernetes Service (EKS) 中執行的 Kubernetes 叢集可以佈建由 ONTAP 支援的區塊和檔案持續性磁碟區。

需求

除了"Trident 需求"、若要將適用於 ONTAP 的 FSX 與 Trident 整合、您還需要：

- 現有的Amazon EKS叢集或自行管理的Kubernetes叢集、已安裝「kubectll」。
- 可從叢集工作節點存取的現有 Amazon FSX for NetApp ONTAP 檔案系統和儲存虛擬機器 (SVM)。
- 已準備好的工作節點 "NFS或iSCSI"。



請務必遵循Amazon Linux和Ubuntu所需的節點準備步驟 "Amazon機器映像" (AMIs)、視您的EKS AMI類型而定。

考量

- SMB Volume：
 - 使用支援SMB磁碟區 `ontap-nas` 僅限驅動程式。
 - Trident EKS 附加元件不支援 SMB Volume。
 - Trident 僅支援掛載至 Windows 節點上執行的 Pod 的 SMB 磁碟區。如 "準備配置SMB磁碟區" 需詳細資訊、請參閱。
- 在 Trident 24.02 之前、在已啟用自動備份的 Amazon FSX 檔案系統上建立的磁碟區、無法由 Trident 刪除。若要在 Trident 24.02 或更新版本中避免此問題、請在 AWS FSX for ONTAP 的後端組態檔案中指定 `fsxFilesystemID`、`AWS apiRegion`、`AWS apikey` 和 `AWS secretKey`。



如果您要將 IAM 角色指定給 Trident、則可以省略將 `apiKey` 和 `secretKey` 欄位明確指定 `apiRegion` 給 Trident。如需詳細資訊、請 ["FSX提供ONTAP 各種組態選項和範例"](#) 參閱。

驗證

Trident 提供兩種驗證模式。

- 認證型（建議）：在 AWS Secrets Manager 中安全地儲存認證。您可以將使用者用於檔案系統、或是使用 `fsxadmin vsadmin` 為 SVM 設定的使用者。



Trident 應以 SVM 使用者或具有相同角色之不同名稱的使用者身分執行 `vsadmin`。Amazon FSX for NetApp ONTAP 的 `fsxadmin` 使用者僅能有限地取代 ONTAP `admin` 叢集使用者。我們強烈建議搭配 Trident 使用 `vsadmin`。

- 憑證型：Trident 將使用 SVM 上安裝的憑證、與 FSX 檔案系統上的 SVM 通訊。

如需啟用驗證的詳細資訊、請參閱您的驅動程式類型驗證：

- ["ASNAS 驗證 ONTAP"](#)
- ["支援 SAN 驗證 ONTAP"](#)

如需詳細資訊、請參閱

- ["Amazon FSX for NetApp ONTAP 的支援文件"](#)
- ["Amazon FSX for NetApp ONTAP 的部落格文章"](#)

建立 IAM 角色和 AWS 密碼

您可以將 Kubernetes Pod 設定為以 AWS IAM 角色進行驗證、而非提供明確的 AWS 認證、以存取 AWS 資源。



若要使用 AWS IAM 角色進行驗證、您必須使用 EKS 部署 Kubernetes 叢集。

建立 AWS Secret Manager 機密

以下範例建立 AWS Secret Manager 密碼來儲存 Trident CSI 認證：

```
aws secretsmanager create-secret --name trident-secret --description "Trident CSI credentials" --secret-string '{"user":"vsadmin","password":"<svmpassword>"}
```

建立 IAM 原則

下列範例使用 AWS CLI 建立 IAM 原則：

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy-document file://policy.json --description "This policy grants access to Trident CSI to FSxN and Secret manager"
```

• 政策 JSON 檔案 * :

```
policy.json:
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx>DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-id>:secret:<aws-secret-manager-name>"
    }
  ],
  "Version": "2012-10-17"
}
```

為服務帳戶建立和 IAM 角色

以下範例為 EKS 中的服務帳戶建立 IAM 角色：

```
eksctl create iamserviceaccount --name trident-controller --namespace trident
--cluster <my-cluster> --role-name <AmazonEKS_FSxN_CSI_DriverRole> --role-only
--attach-policy-arn arn:aws:iam::aws:policy/service-
role/AmazonFSxNCSIDriverPolicy --approve
```

安裝 **Astra Trident**

Astra Trident 簡化了在 Kubernetes 進行 NetApp ONTAP 儲存管理的 Amazon FSX、讓開發人員和管理員能夠專注於應用程式部署。

您可以使用下列其中一種方法來安裝 Astra Trident：

- 掌舵
- EKS 附加元件

If you want to make use of the snapshot functionality, install the CSI snapshot controller add-on. Refer to <https://docs.aws.amazon.com/eks/latest/userguide/csi-snapshot-controller.html>.

透過 helm 安裝 Astra Trident

1. 下載 Astra Trident 安裝程式套件

Astra Trident 安裝程式套件包含部署 Trident 操作員及安裝 Astra Trident 所需的一切。從 GitHub 的 Assets 區段下載並解壓縮 Astra Trident 安裝程式的最新版本。

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

2. 使用下列環境變數設定 * 雲端供應商 * 和 * 雲端 IDENTITY * 旗標的值：

```
export CP="AWS"
export CI="'eks.amazonaws.com/role-arn:arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'"
```

以下範例安裝 Astra Trident 並將旗標設定 cloud-provider 為 `SCP` 和 cloud-identity \$CI：

```
helm install trident trident-operator-100.2410.0.tgz --set
cloudProvider=$CP --set cloudIdentity=$CI --namespace trident
```

您可以使用 `helm list` 命令檢閱安裝詳細資料，例如名稱，命名空間，圖表，狀態，應用程式版本和修訂版編號。

```
helm list -n trident
```

NAME	NAMESPACE	REVISION	UPDATED
STATUS	CHART		APP VERSION
trident-operator	trident	1	2024-10-14 14:31:22.463122
+0300 IDT	deployed	trident-operator-100.2410.0	24.10.0

透過 EKS 附加元件安裝 Astra Trident

Astra Trident EKS 附加元件包含最新的安全性修補程式、錯誤修正程式、並經過 AWS 驗證、可與 Amazon EKS 搭配使用。EKS 附加元件可讓您持續確保 Amazon EKS 叢集安全穩定、並減少安裝、設定及更新附加元件所需的工作量。

先決條件

在設定 AWS EKS 的 Astra Trident 附加元件之前、請確定您具有下列項目：

- 具有附加訂閱的 Amazon EKS 叢集帳戶
- AWS 對 AWS 市場的權限：
"aws-marketplace:ViewSubscriptions",
"aws-marketplace:Subscribe",
"aws-marketplace:Unsubscribe"
- AMI 類型：Amazon Linux 2 (AL2_x86_64) 或 Amazon Linux 2 ARM (AL2_ARM_64)
- 節點類型：AMD 或 ARM
- 現有的 Amazon FSX for NetApp ONTAP 檔案系統

啟用 AWS 的 Astra Trident 附加元件

EKS 叢集

下列命令範例會安裝 Astra Trident EKS 附加元件：

```
eksctl create addon --cluster clusterName --name netapp_trident-operator  
--version v24.6.1-eksbuild  
eksctl create addon --cluster clusterName --name netapp_trident-operator  
--version v24.6.1-eksbuild.1 (使用專用版本)
```



當您設定選用參數 `cloudIdentity` 時，請確保在使用 EKS 附加元件安裝 Trident 時指定 `cloudProvider`。

管理主控台

1. 開啟 Amazon EKS 主控台：<https://console.aws.amazon.com/eks/home#/clusters>。
2. 在左導覽窗格中、按一下 * 叢集 *。
3. 按一下您要設定 NetApp Trident CSI 附加元件的叢集名稱。
4. 按一下 * 附加元件 *、然後按一下 * 取得更多附加元件 *。
5. 在 *S*SELECT 附加元件 * 頁面上、執行下列步驟：
 - a. 在 AWS Marketplace EKS-addons 區段中、選取 *Astra Trident by NetApp* 核取方塊。
 - b. 單擊 * 下一步 *。
6. 在 * 設定選取的附加元件 * 設定頁面上、執行下列步驟：
 - a. 選擇您要使用的 * 版本 *。
 - b. 對於 * 選取 IAM 角色 *、請保留 * 未設定 *。
 - c. 展開 * 選用組態設定 *、遵循 * 附加元件組態架構 *、並將 * 組態值 * 區段上的組態值參數設定為您在上一個步驟中建立的角色參數（值應採用下列格式：`eks.amazonaws.com/role-arn:arn:aws:iam::464262061435:role/AmazonEKS_FSXN_CSI_DriverRole`）。如果您為衝突解決方法選取「覆寫」、則現有附加元件的一或多個設定可以使用 Amazon EKS 附加元件設定覆寫。如果您未啟用此選項、且與現有設定發生衝突、則作業將會失敗。您可以使用產生的錯誤訊息來疑難排解衝突。選取此選項之前、請確定 Amazon EKS 附加元件不會管理您需要自行管理的設定。



當您設定選用參數 `cloudIdentity` 時，請確保在使用 EKS 附加元件安裝 Trident 時指定 `cloudProvider`。

7. 選擇 * 下一步 *。
8. 在 * 檢閱及新增 * 頁面上、選擇 * 建立 *。

附加元件安裝完成後、您會看到已安裝的附加元件。

AWS CLI

1. 建立 `add-on.json` 檔案：

```
add-on.json
{
    "clusterName": "<eks-cluster>",
    "addonName": "netapp_trident-operator",
    "addonVersion": "v24.6.1-eksbuild.1",
    "serviceAccountRoleArn": "arn:aws:iam::123456:role/astratrident-
role",
    "configurationValues": "{\"cloudIdentity\":
'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/astratrident-
role'\",
    \"cloudProvider\": \"AWS\"}"
}
```



當您設定選用參數 `cloudIdentity` 時，請確保在使用 EKS 附加元件安裝 Trident 時指定 `AWS` 為 `cloudProvider`。

2. 安裝 Astra Trident EKS 附加元件 "

```
aws eks create-addon --cli-input-json file://add-on.json
```

更新 Astra Trident EKS 附加元件

EKS 叢集

- 檢查 FSxN Trident CSI 附加元件的目前版本。以叢集名稱取代 `my-cluster`。
`eksctl get addon --name netapp_trident-operator --cluster my-cluster`
- 輸出範例：*

```
NAME                                VERSION                                STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v24.6.1-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}

```

- 將附加元件更新至上一個步驟輸出中可用更新所傳回的版本。
`eksctl update addon --name netapp_trident-operator --version v24.6.1-eksbuild.1 --cluster my-cluster --force`

如果您移除此 `--force` 選項、且任何 Amazon EKS 附加元件設定與您現有的設定發生衝突、則更新 Amazon EKS 附加元件會失敗；您會收到錯誤訊息、協助您解決衝突。在指定此選項之前、請確定 Amazon EKS 附加元件不會管理您需要管理的設定、因為這些設定會以此選項覆寫。如需此設定的其他選項的詳細資訊，請參閱 "[附加元件](#)"。如需 Amazon EKS Kubernetes 現場管理的詳細資訊、請參閱 "[Kubernetes 現場管理](#)"。

管理主控台

1. 打開 Amazon EKS 控制檯 <https://console.aws.amazon.com/eks/home#/clusters>。
2. 在左導覽窗格中、按一下 * 叢集 *。
3. 按一下您要更新 NetApp Trident CSI 附加元件的叢集名稱。
4. 按一下 * 附加元件 * 索引標籤。
5. 按一下 **Astra Trident by NetApp** *，然後按一下 *Edit*。
6. 在 * 設定 Astra Trident by NetApp * 頁面上、執行下列步驟：
 - a. 選擇您要使用的 * 版本 *。
 - b. (可選) 您可以展開 * 可選配置設置 * 並根據需要進行修改。
 - c. 按一下 *儲存變更*。

AWS CLI

下列範例更新 EKS 附加元件：

```
aws eks update-addon --cluster-name my-cluster netapp_trident-operator vpc-cni
--addon-version v24.6.1-eksbuild.1 \
--service-account-role-arn arn:aws:iam::111122223333:role/role-name
--configuration-values '{}' --resolve-conflicts --preserve

```

解除安裝 / 移除 Astra Trident EKS 附加元件

您有兩種移除 Amazon EKS 附加元件的選項：

- * 保留叢集上的附加軟體 * –此選項會移除 Amazon EKS 對任何設定的管理。它也會移除 Amazon EKS 通知您更新的功能、並在您啟動更新後自動更新 Amazon EKS 附加元件。不過、它會保留叢集上的附加軟體。此選項可讓附加元件成為自我管理的安裝、而非 Amazon EKS 附加元件。有了這個選項、附加元件就不會停機。保留 `--preserve` 命令中的選項以保留附加元件。
- * 從叢集完全移除附加軟體 * –我們建議您只有在叢集上沒有任何相關資源的情況下、才從叢集移除 Amazon EKS 附加元件。從命令中移除 `--preserve` 選項 `delete` 以移除附加元件。



如果附加元件有相關的 IAM 帳戶、則不會移除 IAM 帳戶。

EKS 叢集

下列命令會解除安裝 Astra Trident EKS 附加元件：

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

管理主控台

1. 開啟 Amazon EKS 主控台：<https://console.aws.amazon.com/eks/home#/clusters>。
2. 在左導覽窗格中、按一下 * 叢集 *。
3. 按一下您要移除 NetApp Trident CSI 附加元件的叢集名稱。
4. 單擊 **Add-ons** 選項卡，然後單擊 *Astra Trident by NetApp*。
5. 按一下「移除」。
6. 在 * 移除 NetApp_trident 操作員確認 * 對話方塊中、執行下列步驟：
 - a. 如果您想要 Amazon EKS 停止管理附加元件的設定、請選取 * 保留在叢集 * 上。如果您想要保留叢集上的附加軟體、以便自行管理附加元件的所有設定、請執行此動作。
 - b. 輸入 **NetApp_trident -- operer**。
 - c. 按一下「移除」。

AWS CLI

以叢集名稱取代 `my-cluster`、然後執行下列命令。

```
aws eks delete-addon --cluster-name my-cluster --addon-name netapp_trident-operator --preserve
```

設定儲存後端

整合 SAN 和 NAS 驅動程式 ONTAP

您可以使用儲存在 AWS Secret Manager 中的 SVM 認證（使用者名稱和密碼）來建立後端檔案、如以下範例所示：

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFileSystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFileSystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

如需建立後端的相關資訊、請參閱下列頁面：

- ["使用ONTAP NetApp NAS驅動程式設定後端"](#)
- ["使用ONTAP SAN驅動程式設定後端"](#)

適用於 **ONTAP** 驅動程式詳細資料的 **FSX**

您可以使用下列驅動程式、將 Trident 與 Amazon FSX for NetApp ONTAP 整合：

- `ontap-san`：配置的每個 PV 都是其各自 Amazon FSX 中的 LUN（用於 NetApp ONTAP Volume）。建議用於區塊儲存。
- `ontap-nas`：配置的每個 PV 都是 NetApp ONTAP Volume 的完整 Amazon FSX。建議用於 NFS 和 SMB。
- 「ONTAP-san經濟型」：每個配置的PV都是LUN、每個Amazon FSX for NetApp ONTAP 的LUN數量可設定。
- 「ONTAP-NAS-EAS'：每個提供的PV都是qtree、每個Amazon FSX的NetApp ONTAP 功能是可設定的配額樹數。
- 「ONTAP-NAS-Flexgroup」：每個提供的PV都是適用於NetApp ONTAP FlexGroup 的完整Amazon FSX。

如需驅動程式詳細資料、請參閱 ["NAS 驅動程式"](#) 和 ["SAN 驅動程式"](#)。

組態範例

搭配加密管理程式的 **AWS FSX for ONTAP** 組態

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFileSystemID: fs-xxxxxxxxxx
  managementLIF:
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxx:secret:secret-
name"
    type: awsarn
```

SMB 磁碟區的儲存類別組態

使用 `nasType`、`node-stage-secret-name` 和 `node-stage-secret-namespace`、您可以指定 SMB 磁碟區、並提供所需的 Active Directory 認證資料。使用支援 SMB 磁碟區 `ontap-nas` 僅限驅動程式。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: nas-smb-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

後端進階組態和範例

如需後端組態選項、請參閱下表：

參數	說明	範例
「分度」		永遠為1
「storageDriverName」	儲存驅動程式名稱	<code>ontap-nas</code> 、 <code>ontap-nas-economy</code> 、 <code>ontap-nas-flexgroup</code> 、 <code>ontap-san</code> 、 <code>ontap-san-economy</code>
「後端名稱」	自訂名稱或儲存後端	驅動程式名稱+「_」+ dataLIF
《馬納格門達利》	叢集或 SVM 管理 LIF 的 IP 位址可以指定完整網域名稱（FQDN）。如果使用 IPv6 旗標安裝 Trident、則可設定為使用 IPv6 位址。IPv6 位址必須以方括弧來定義、例如[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。如果您在欄位下方 <code>aws</code> 提供、則 <code>fsxFilesystemID</code> 不需要提供、 <code>managementLIF</code> 因為 Trident 會從 AWS 擷取 SVM <code>managementLIF</code> 資訊。因此、您必須在 SVM 下提供使用者的認證（例如： <code>vsadmin</code> ）、且使用者必須具有該 <code>vsadmin</code> 角色。	「10.0.0.1」、 <code>「[2001:1234:abcd:::fefo]」</code>

參數	說明	範例
「DataLIF」	傳輸協定LIF的IP位址。不適用 NAS 驅動程式：建議您指定dataLIF ONTAP。如果未提供、Trident 會從 SVM 擷取資料生命。您可以指定要用於NFS掛載作業的完整網域名稱 (FQDN)、讓您建立循環配置資源DNS、以便在多個資料生命期之間達到負載平衡。可在初始設定之後變更。請參閱。《 SAN 驅動程式：請勿指定用於iSCSI》 ONTAP。Trident 使用 ONTAP 選擇性 LUN 對應來探索建立多重路徑工作階段所需的 iSCSI 生命。如果明確定義dataLIF、就會產生警告。如果使用 IPv6 旗標安裝 Trident、則可設定為使用 IPv6 位址。IPv6位址必須以方括弧來定義、例如[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。	
「AutoExpportPolicy」	啟用自動匯出原則建立及更新[布林值]。使用 `autoExportPolicy` 和 `autoExportCIDRs` 選項、Trident 可以自動管理匯出原則。	「假」
《AutoExpportCIDR》 (自動匯出CTR)	將 Kubernetes 節點 IP 篩選在啟用時的 CIDR 清單 autoExportPolicy。使用 `autoExportPolicy` 和 `autoExportCIDRs` 選項、Trident 可以自動管理匯出原則。	「[「0.00.0.0/0」、 「:/0」]」
《標籤》	套用到磁碟區的任意JSON-格式化標籤集	"
「用戶端憑證」	用戶端憑證的Base64編碼值。用於憑證型驗證	"
「clientPrivate Key」	用戶端私密金鑰的Base64編碼值。用於憑證型驗證	"
「可信賴的CACertificate」	受信任CA憑證的Base64編碼值。選用。用於憑證型驗證。	"
《使用者名稱》	連線至叢集或SVM的使用者名稱。用於認證型驗證。例如、vsadmin。	
密碼	連線至叢集或SVM的密碼。用於認證型驗證。	
《虛擬機器》	要使用的儲存虛擬機器	指定SVM管理LIF時衍生。
「storagePrefix」	在SVM中配置新磁碟區時所使用的前置碼。無法在建立後修改。若要更新此參數、您需要建立新的後端。	trident

參數	說明	範例
「限制Aggregateusage」	* 請勿指定 Amazon FSX for NetApp ONTAP。*提供的 `fsxadmin` 和 `vsadmin` 不包含使用 Trident 擷取彙總使用量並加以限制所需的權限。	請勿使用。
《限制Volume大小》	如果要求的磁碟區大小高於此值、則資源配置失敗。也會限制其管理的qtree和LUN、以及的磁碟區大小上限 qtreesPerFlexvol 選項可自訂每FlexVol 個支援區的配額樹數上限。	「」（預設不強制執行）
《lunsPerFlexvol》	每FlexVol 個LUN的最大LUN數量、範圍必須為[50、200]。僅限SAN。	"100"
「DebugTraceFlags」	疑難排解時要使用的偵錯旗標。範例：{"API":假、「method」:true} 不使用 debugTraceFlags 除非您正在疑難排解並需要詳細的記錄傾印。	null
「nfsMountOptions」	以逗號分隔的NFS掛載選項清單。Kubernetes-Persistent Volume 的掛載選項通常是在儲存類別中指定、但如果儲存類別中未指定掛載選項、則 Trident 會回復為使用儲存後端組態檔案中指定的掛載選項。如果儲存類別或組態檔案中未指定任何掛載選項、Trident 將不會在關聯的持續磁碟區上設定任何掛載選項。	"
nasType	設定NFS或SMB磁碟區建立。選項包括 nfs、smb 或 null。*必須設定為 `smb` 對於SMB Volume。*設定為null、預設為NFS Volume。	nfs
"qtreesPerFlexvol"	每FlexVol 個邊的最大qtree數、必須在範圍內[50、300]	"200"
smbShare	您可以指定下列其中一項：使用 Microsoft 管理主控台或 ONTAP CLI 建立的 SMB 共用名稱、或是允許 Trident 建立 SMB 共用的名稱。ONTAP 後端的 Amazon FSX 需要此參數。	smb-share

參數	說明	範例
《useREST》	<p>使用ONTAP Isrest API的布林參數。<i>* 技術預覽 *</i></p> <p>useREST 是以建議用於測試環境、而非正式作業工作負載的「技術預覽」形式提供。設為 true 時、Trident 將使用 ONTAP REST API 與後端通訊。此功能需要ONTAP 使用更新版本的版本。此外、使用的 ONTAP 登入角色必須具有應用程式存取權 `ontap`。這是預先定義的和角色所滿足 vsadmin cluster-admin 的。</p>	「假」
aws	<p>您可以在 AWS FSX for ONTAP 的組態檔中指定下列項目：</p> <ul style="list-style-type: none"> - fsxFilesystemID：指定 AWS FSX 檔案系統的 ID。 - apiRegion：AWS API 區域名稱。 - apikey：AWS API 金鑰。 - secretKey：AWS 秘密金鑰。 	"" "" ""
credentials	<p>指定要儲存在 AWS Secret Manager 中的 FSX SVM 認證。</p> <ul style="list-style-type: none"> - name：機密的 Amazon 資源名稱（ARN）、其中包含 SVM 的認證。 - type：設為 awsarn。 <p>請參閱 "建立 AWS Secrets Manager 密碼" 以取得更多資訊。</p>	

用於資源配置磁碟區的后端組態選項

您可以使用中的這些選項來控制預設資源配置 defaults 組態區段。如需範例、請參閱下列組態範例。

參數	說明	預設
"paceAllocate (配置) "	LUN的空間分配	"真的"
《保護區》	空間保留模式；「無」（精簡）或「Volume」（完整）	無
「快照原則」	要使用的Snapshot原則	無

參數	說明	預設
「qosPolicy」	要指派給所建立磁碟區的QoS原則群組。選擇每個儲存集區或後端的其中一個qosPolicy或adaptiveQosPolicy。搭配 Trident 使用 QoS 原則群組需要 ONTAP 9.8 或更新版本。您應該使用非共用的 QoS 原則群組、並確保個別將原則群組套用至每個成員。共享 QoS 原則群組會強制執行所有工作負載總處理量的上限。	「」
《adaptiveQosPolicy》	要指派給所建立磁碟區的調適性QoS原則群組。選擇每個儲存集區或後端的其中一個qosPolicy或adaptiveQosPolicy。不受ONTAP-NAS-經濟支援。	「」
「快照保留區」	保留給快照「0」的磁碟區百分比	如果 snapshotPolicy 是 none、else 「」
「PlitOnClone」	建立複本時、從其父複本分割複本	「假」
加密	在新磁碟區上啟用 NetApp Volume Encryption (NVE)；預設為 false。必須在叢集上授權並啟用NVE、才能使用此選項。如果在後端啟用 NAE、則 Trident 中配置的任何 Volume 都將啟用 NAE。如需更多資訊、請參閱" Trident 如何與 NVE 和 NAE 搭配運作 "：。	「假」
luksEncryption	啟用LUKS加密。請參閱 " 使用Linux 統一金鑰設定 (LUKS) "。僅限SAN。	"
「分層政策」	要使用的分層原則 none	snapshot-only 適用於 ONTAP 9.5 之前的 SVM-DR 組態
「unixPermissions」	新磁碟區的模式。如果是SMB磁碟區、請保留空白。	「」
《生態樣式》	新磁碟區的安全樣式。NFS支援 mixed 和 unix 安全樣式；SMB支援 mixed 和 ntfs 安全樣式：	NFS預設為 unix。SMB預設為 ntfs。

準備配置SMB磁碟區

您可以使用來配置SMB磁碟區 `ontap-nas` 驅動程式：完成之前 [整合SAN和NAS驅動程式ONTAP](#) 完成下列步驟。

開始之前

在您使用配置 SMB 磁碟區之前、請先使用 `ontap-nas` 驅動程式、您必須具備下列項目。

- Kubernetes叢集具備Linux控制器節點、以及至少一個執行Windows Server 2019的Windows工作節點。Trident 僅支援掛載至 Windows 節點上執行的 Pod 的 SMB 磁碟區。

- 至少有一個 Trident 機密包含您的 Active Directory 認證。產生機密 `smbcreds`：

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- 設定為Windows服務的SCSI Proxy。若要設定 `csi-proxy`、請參閱 ["GitHub : csi Proxy"](#) 或 ["GitHub : 適用於Windows的SCSI Proxy"](#) 適用於Windows上執行的Kubernetes節點。

步驟

1. 建立SMB共用區。您可以使用兩種方式之一來建立SMB管理共用區 ["Microsoft管理主控台"](#) 共享資料夾嵌入式管理單元或使用ONTAP CLI。若要使用ONTAP CLI建立SMB共用：

- a. 如有必要、請建立共用的目錄路徑結構。

- `vserver cifs share create` 命令會在共用建立期間檢查`-path`選項中指定的路徑。如果指定的路徑不存在、則命令會失敗。

- b. 建立與指定SVM相關的SMB共用區：

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

- c. 確認共用區已建立：

```
vserver cifs share show -share-name share_name
```



請參閱 ["建立SMB共用區"](#) 以取得完整詳細資料。

2. 建立後端時、您必須設定下列項目以指定SMB Volume。如需ONTAP 所有的FSXfor Sendbackend組態選項、請參閱 ["FSX提供ONTAP 各種組態選項和範例"](#)。

參數	說明	範例
<code>smbShare</code>	您可以指定下列其中一項：使用 Microsoft 管理主控台或 ONTAP CLI 建立的 SMB 共用名稱、或是允許 Trident 建立 SMB 共用的名稱。ONTAP 後端的 Amazon FSX 需要此參數。	<code>smb-share</code>
<code>nasType</code>	*必須設定為 <code>smb</code> .*如果為 <code>null</code> 、則預設為 <code>nfs</code> 。	<code>smb</code>
《生態樣式》	新磁碟區的安全樣式。必須設定為 ntfs 或 mixed 適用於SMB磁碟區。	<code>ntfs</code> 或 <code>mixed</code> 適用於SMB磁碟區

參數	說明	範例
「unixPermissions」	新磁碟區的模式。SMB磁碟區*必須保留為空白。*	"

設定儲存類別和 PVC

設定 Kubernetes StorageClass 物件並建立儲存類別、以指示 Trident 如何配置磁碟區。建立 PersistentVolume (PV) 和 PersistentVolume Claim (PVC)、使用設定的 Kubernetes StorageClass 來要求存取 PV。然後、您可以將 PV 掛載至 Pod。

建立儲存類別

設定 Kubernetes StorageClass 物件

將 "Kubernetes StorageClass 物件" Trident 識別為用於該類別的資源配置程式、會指示 Trident 如何資源配置 Volume。例如：

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"

```

如需儲存類別如何與互動的詳細資訊 PersistentVolumeClaim、以及控制 Trident 配置磁碟區的參數、請參閱"[Kubernetes和Trident物件](#)"。

建立儲存類別

步驟

1. 這是 Kubernetes 物件、請使用 kubectl 在Kubernetes中建立。

```
kubectl create -f storage-class-ontapnas.yaml
```

2. 現在您應該會在 Kubernetes 和 Trident 中同時看到 * base-csi* 儲存類別、而 Trident 應該已經在後端上探索到這些集區。

```
kubectl get sc basic-csi
NAME          PROVISIONER          AGE
basic-csi     csi.trident.netapp.io 15h
```

建立 PV 和 PVC

答 "[PersistentVolumer](#)" (PV) 是叢集管理員在 Kubernetes 叢集上配置的實體儲存資源。。
"[_PersistentVolume Claim](#)" (PVC) 是存取叢集上 PersistentVolume 的要求。

可將 PVC 設定為要求儲存特定大小或存取模式。叢集管理員可以使用相關的 StorageClass 來控制超過 PersistentVolume 大小和存取模式的權限、例如效能或服務層級。

建立 PV 和 PVC 之後、您可以將磁碟區裝入 Pod 。

範例資訊清單

PersistentVolume 範例資訊清單

此範例資訊清單顯示與 StorageClass 相關的 10Gi 基本 PV basic-csi 。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/my/host/path"
```

PersistentVolume Claim 範例資訊清單

這些範例顯示基本的 PVC 組態選項。

可存取 **RWO** 的 **PVC**

此範例顯示具有 `rwX` 存取權的基本 PVC、與名稱為的 StorageClass 相關聯 `basic-csi`。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

採用 **NVMe / TCP** 的 **PVC**

此範例顯示 NVMe / TCP 的基本 PVC、並提供與命名 StorageClass 相關的 `rwO` 存取 `protection-gold`。

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san-nvme
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 300Mi
  storageClassName: protection-gold
```

建立 **PV** 和 **PVC**

步驟

1. 建立 PV。

```
kubectl create -f pv.yaml
```

2. 確認 PV 狀態。

```
kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS REASON    AGE
pv-storage   4Gi      RWO           Retain          Available
7s
```

3. 建立 PVC。

```
kubectl create -f pvc.yaml
```

4. 確認 PVC 狀態。

```
kubectl get pvc
NAME          STATUS  VOLUME          CAPACITY  ACCESS MODES  STORAGECLASS  AGE
pvc-storage  Bound  pv-name 2Gi      RWO
5m
```

如需儲存類別如何與互動的詳細資訊 PersistentVolumeClaim、以及控制 Trident 配置磁碟區的參數、請參閱"[Kubernetes和Trident物件](#)"。

Trident 屬性

這些參數決定應使用哪些Trident託管儲存資源池來配置特定類型的磁碟區。

屬性	類型	價值	優惠	申請	支援者
媒體 ^{1^}	字串	HDD、混合式、SSD	資源池包含此類型的媒體、混合式表示兩者	指定的媒體類型	ONTAP-NAS、ONTAP-NAS-經濟型、ONTAP-NAS-flexgroup、ONTAP-SAN、solidfire-san
資源配置類型	字串	纖薄、厚實	Pool支援此資源配置方法	指定的資源配置方法	厚：全ONTAP 是邊、薄：全ONTAP 是邊、邊、邊、邊、邊、邊、邊、邊、邊、邊

屬性	類型	價值	優惠	申請	支援者
後端類型	字串	ONTAP-NAS 、ONTAP-NAS- 經濟 型、ONTAP- NAS-flexgroup 、ONTAP- SAN、solidfire- san、GCP- CVS、azure- NetApp-Files 、ONTAP-san經 濟	集區屬於此類型 的後端	指定後端	所有驅動程式
快照	布爾	對、錯	集區支援具有快 照的磁碟區	已啟用快照 的Volume	ONTAP-NAS 、ONTAP- SAN、Solidfire- SAN、GCP-CVS
複製	布爾	對、錯	資源池支援複製 磁碟區	已啟用複本 的Volume	ONTAP-NAS 、ONTAP- SAN、Solidfire- SAN、GCP-CVS
加密	布爾	對、錯	資源池支援加密 磁碟區	已啟用加密 的Volume	ONTAP-NAS 、ONTAP-NAS- 經濟型、ONTAP- NAS- FlexGroups、ON TAP-SAN
IOPS	內部	正整數	集區能夠保證此 範圍內的IOPS	Volume保證這 些IOPS	solidfire-san

¹：ONTAP Select 不受支援

部署範例應用程式

部署範例應用程式。

步驟

1. 將磁碟區裝入 Pod。

```
kubectl create -f pv-pod.yaml
```

這些範例顯示將 PVC 附加至 Pod 的基本組態：* 基本組態 *：

```

kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
        claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage

```



您可以使用監控進度 `kubectl get pod --watch`。

2. 確認磁碟區已掛載到上 `/my/mount/path`。

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

```

Filesystem                                Size
Used Avail Use% Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06 1.1G
320K 1.0G 1% /my/mount/path

```

1. 您現在可以刪除 Pod。Pod 應用程式將不再存在、但該磁碟區仍會保留。

```
kubectl delete pod task-pv-pod
```

在 **EKS 叢集** 上設定 **Astra Trident EKS 附加元件**

Astra Trident 簡化了在 Kubernetes 進行 NetApp ONTAP 儲存管理的 Amazon FSX、讓開發人員和管理員能夠專注於應用程式部署。Astra Trident EKS 附加元件包含最新的安全性修補程式、錯誤修正程式、並經過 AWS 驗證、可與 Amazon EKS 搭配使用。EKS 附加元件可讓您持續確保 Amazon EKS 叢集安全穩定、並減少安裝、設定及更新附加元件所需的

工作量。

先決條件

在設定 AWS EKS 的 Astra Trident 附加元件之前、請確定您具有下列項目：

- 具有附加訂閱的 Amazon EKS 叢集帳戶
- AWS 對 AWS 市場的權限：
"aws-marketplace:ViewSubscriptions",
"aws-marketplace:Subscribe",
"aws-marketplace:Unsubscribe"
- AMI 類型：Amazon Linux 2 (AL2_x86_64) 或 Amazon Linux 2 ARM (AL2_ARM_64)
- 節點類型：AMD 或 ARM
- 現有的 Amazon FSX for NetApp ONTAP 檔案系統

步驟

1. 在您的 EKS Kubernetes 叢集上、瀏覽至 * 附加元件 * 索引標籤。

The screenshot displays the AWS EKS console interface for a cluster named 'tri-env-eks'. At the top right, there are buttons for 'Delete cluster' and 'Upgrade version'. A notification banner at the top states: 'End of standard support for Kubernetes version 1.30 is July 28, 2025. On that date, your cluster will enter the extended support period with additional fees. For more information, see the pricing page [2].' Below this is a 'Cluster info' section with a table:

Status	Kubernetes version	Support period	Provider
Active	1.30	Standard support until July 28, 2025	EKS

Below the table is a navigation bar with tabs: Overview, Resources, Compute, Networking, Add-ons (1), Access, Observability, Upgrade insights, Update history, and Tags. A notification banner below the navigation bar says: 'New versions are available for 3 add-ons.' The 'Add-ons (3)' section is active, showing a search bar with the text 'Find add-on', filters for 'Any category' and 'Any status', and a '3 matches' indicator. There are buttons for 'View details', 'Edit', 'Remove', and 'Get more add-ons'.

2. 前往 * AWS Marketplace 附加元件 * 並選擇 _storage 類別。

AWS Marketplace add-ons (1) ↻

Discover, subscribe to and configure EKS add-ons to enhance your EKS clusters.

Filtering options

Any category ▾ NetApp, Inc. ▾ Any pricing model ▾ Clear filters

NetApp, Inc. ✕ < 1 >

NetApp **NetApp Trident** □

NetApp Trident streamlines Amazon FSx for NetApp ONTAP storage management in Kubernetes to let your developers and administrators focus on application deployment. FSx for ONTAP flexibility, scalability, and integration capabilities make it the ideal choice for organizations seeking efficient containerized storage workflows. [Product details](#)

Standard Contract

Category storage	Listed by NetApp, Inc.	Supported versions 1.30, 1.29, 1.28, 1.27, 1.26, 1.25, 1.24, 1.23	Pricing starting at View pricing details
---------------------	---	--	---

Cancel **Next**

3. 找到 *Astra NetApp Trident Trident 附加元件* 並選取該核取方塊。
4. 選擇所需版本的附加元件。

NetApp Trident

Remove add-on

Listed by NetApp	Category storage	Status ✔ Ready to install
----------------------------	---------------------	------------------------------

You're subscribed to this software
You can view the terms and pricing details for this product or choose another offer if one is available.

View subscription ×

Version
Select the version for this add-on.

v24.6.1-eksbuild.1 ▼

Select IAM role
Select an IAM role to use with this add-on. To create a new custom role, follow the instructions in the [Amazon EKS User Guide](#).

Not set ▼

▶ Optional configuration settings

Cancel Previous **Next**

5. 選取 IAM 角色選項以從節點繼承。

199

Review and add

Step 1: Select add-ons

Edit

Selected add-ons (1)

Find add-on

< 1 >

Add-on name



Type



Status

netapp_trident-operator

storage

Ready to install

Step 2: Configure selected add-ons settings

Edit

Selected add-ons version (1)

< 1 >

Add-on name



Version



IAM role for service account (IRSA)

netapp_trident-operator

v24.6.1-eksbuild.1

Not set

Cancel

Previous

Create

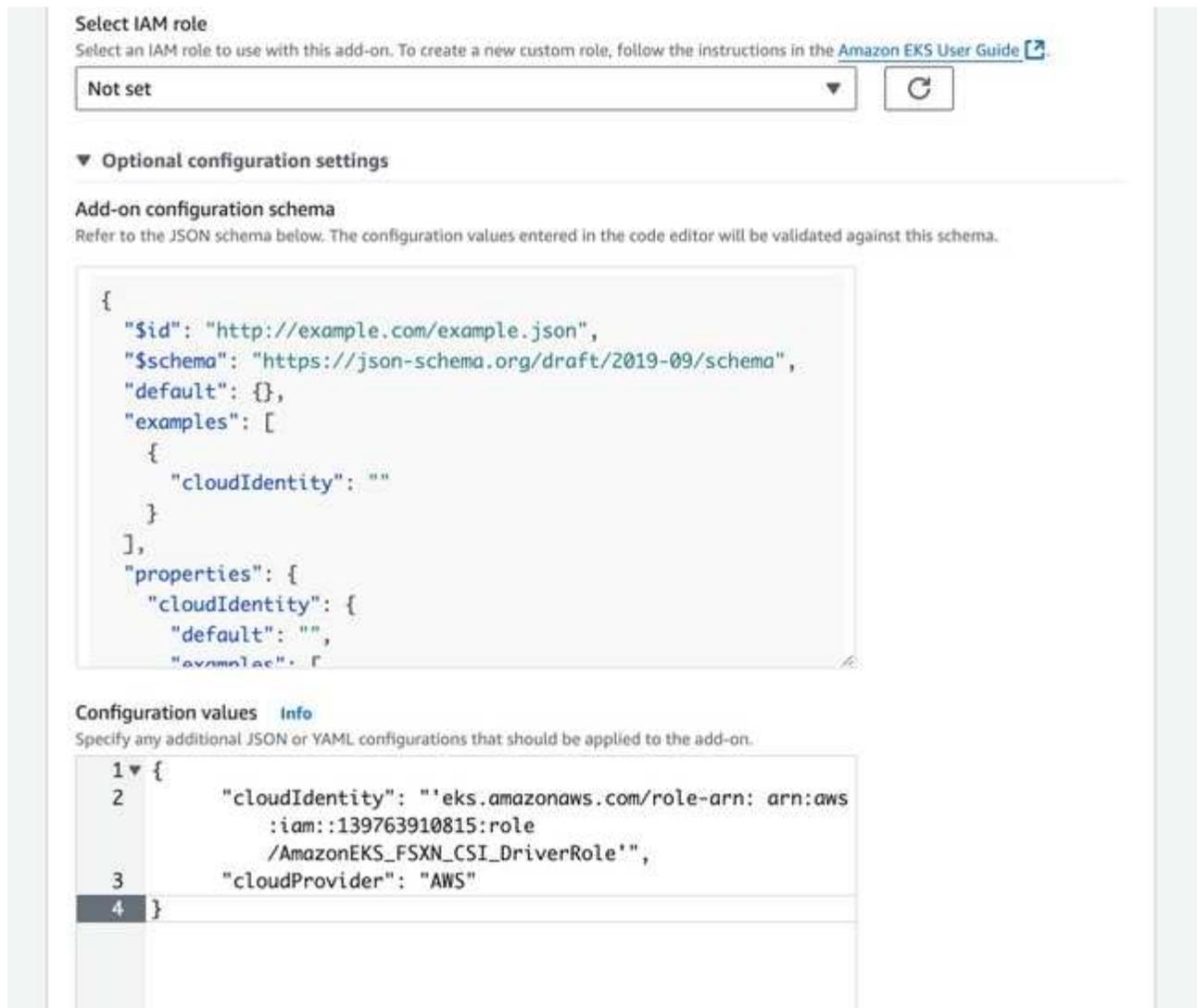
- (可選) 根據需要配置任何可選的配置設置，然後選擇 * 下一步 *。

遵循 *Add-on 組態架構*，並將 *組態值* 區段上的組態值參數設定為您在上一個步驟中建立的角色參數（值應採用下列格式：`eks.amazonaws.com/role-arn:`

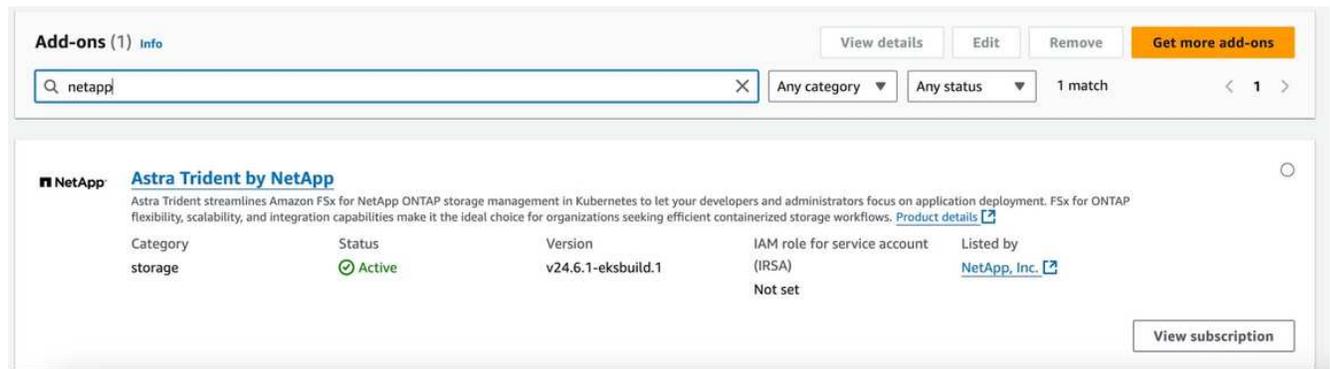
`arn:aws:iam::464262061435:role/AmazonEKS_FSXN_CSI_DriverRole`）。如果您為衝突解決方法選取「覆寫」、則現有附加元件的一或多個設定可以使用 Amazon EKS 附加元件設定覆寫。如果您未啟用此選項、且與現有設定發生衝突、則作業將會失敗。您可以使用產生的錯誤訊息來疑難排解衝突。選取此選項之前、請確定 Amazon EKS 附加元件不會管理您需要自行管理的設定。



當您設定選用參數 `cloudIdentity` 時，請確保在使用 EKS 附加元件安裝 Trident 時指定 `AWS` 為 `cloudProvider`。



7. 選擇* Create (建立)。
8. 確認附加元件的狀態為 *Active*。



使用 CLI 安裝 / 解除安裝 Astra Trident EKS 附加元件

使用 CLI 安裝 Astra Trident EKS 附加元件：

下列範例命令會安裝 Astra Trident EKS 附加元件：

```
eksctl create addon --cluster K8s-arm --name netapp_trident-operator --version
```

```
v24.6.1-eksbuild
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v24.6.1-eksbuild.1 (含專用版本)
```



當您設定選用參數 `cloudIdentity` 時，請確保在使用 EKS 附加元件安裝 Trident 時指定 `cloudProvider`。

使用 CLI 解除安裝 Astra Trident EKS 附加元件：

下列命令會解除安裝 Astra Trident EKS 附加元件：

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

使用 kubectl 建立後端

後端定義 Trident 與儲存系統之間的關係。它告訴 Trident 如何與該儲存系統通訊、以及 Trident 如何從該儲存系統配置磁碟區。安裝 Trident 之後、下一步是建立後端。`TridentBackendConfig` 自訂資源定義 (CRD) 可讓您直接透過 Kubernetes 介面建立及管理 Trident 後端。您可以使用或等效的 CLI 工具來 `kubectl` 進行 Kubernetes 發佈。

`TridentBackendConfig`

`TridentBackendConfig`(`tbc`、`tbconfig`、`tbackendconfig`) 為前端、命名 CRD、可讓您使用管理 Trident 後端 `kubectl`。Kubernetes 和儲存管理員現在可以直接透過 Kubernetes CLI 建立和管理後端 (`tridentctl`、而不需要專用的命令列公用程式)。

建立「`TridentBackendConfig`」物件之後、會發生下列情況：

- Trident 會根據您提供的組態自動建立後端。這在內部表示為 A `TridentBackend` (`tbe`、`tridentbackend`) CR。
- `TridentBackendConfig` 與由 Trident 建立的唯一繫結 `TridentBackend`。

每個「`TridentBackendConfig`」都有一對一的對應、並有「`TridentBackend`」。前者是提供給使用者設計及設定後端的介面、後者是 Trident 代表實際後端物件的方式。



`TridentBackend` CRS 是由 Trident 自動建立。您*不應該*修改這些項目。如果您想要更新後端、請修改物件以進行更新 `TridentBackendConfig`。

請參閱下列範例、以瞭解「`TridentBackendConfig`」CR的格式：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

您也可以查看中的範例 "[Trident安裝程式](#)" 所需儲存平台/服務的範例組態目錄。

◦ spec 採用後端特定的組態參數。在此範例中、後端使用 ontap-san 儲存驅動程式、並使用此處列出的組態參數。如需所需儲存驅動程式的組態選項清單、請參閱 "[儲存驅動程式的後端組態資訊](#)"。

在《TridentBackendConfig》(CRR) 中新推出的「sPEC」一節也包含「認證」和「刪除原則」欄位：

- 「認證資料」：此參數為必填欄位、包含用於驗證儲存系統/服務的認證資料。此設定為使用者建立的 Kubernetes Secret。認證資料無法以純文字格式傳遞、因此會產生錯誤。
- 「刪除原則」：此欄位可定義刪除「TridentBackendConfig」時應發生的情況。可能需要兩種可能的值之一：
 - 「刪除」：這會同時刪除「TridentBackendConfig」和相關後端。這是預設值。
 - 「保留」：刪除「TridentBackendConfig」(TridentBackendConfig) CR時、後端定義仍會存在、並可使用「tridentctl」進行管理。將刪除原則設為「保留」可讓使用者降級至較早版本(21.04之前)、並保留建立的後端。此欄位的值可在建立「TridentBackendConfig」之後更新。



後端名稱是使用「sPEC.backendName」來設定。如果未指定、則會將後端名稱設為「TridentBackendConfig」物件(metadata.name)的名稱。建議使用「sPEC.backendName」明確設定後端名稱。



使用建立的後端 tridentctl 沒有關聯的 `TridentBackendConfig` 物件。您可以建立 CR 來 `TridentBackendConfig` 選擇管理此類後端 `kubectl`。必須注意指定相同的組態參數(例如 spec.backendName、spec.storagePrefix、spec.storageDriverName 等)。Trident 會自動將新建立的後端與先前存在的後端繫結 `TridentBackendConfig`。

步驟總覽

若要使用「kubectl」建立新的後端、您應該執行下列動作：

1. 建立 "[Kubernetes機密](#)"。密碼包含 Trident 與儲存叢集 / 服務通訊所需的認證。
2. 建立「TridentBackendConfig」物件。其中包含有關儲存叢集/服務的詳細資訊、並參考上一步建立的機密。

建立後端之後、您可以使用「`kubectl Get tbc <tbc-name>-n <trident命名空間>`」來觀察其狀態、並收集其他詳細資料。

步驟1：建立Kubernetes機密

建立包含後端存取認證的秘密。這是每個儲存服務/平台所獨有的功能。範例如下：

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: t@Ax@7q(>
```

下表摘要說明每個儲存平台的機密必須包含的欄位：

儲存平台機密欄位說明	秘密	欄位說明
Azure NetApp Files	ClientID	應用程式註冊的用戶端ID
適用於 GCP Cloud Volumes Service	Private金鑰ID	私密金鑰的ID。GCP服務帳戶API金鑰的一部分、具有CVS管理員角色
適用於 GCP Cloud Volumes Service	Private金鑰	私密金鑰：GCP服務帳戶API金鑰的一部分、具有CVS管理員角色
元素 (NetApp HCI / SolidFire)	端點	MVIP、適用於SolidFire 採用租戶認證的不含用戶身分證明的叢集
ONTAP	使用者名稱	連線至叢集/ SVM的使用者名稱。用於認證型驗證
ONTAP	密碼	連線至叢集/ SVM的密碼。用於認證型驗證
ONTAP	用戶端權限金鑰	用戶端私密金鑰的Base64編碼值。用於憑證型驗證
ONTAP	chap使用 者名稱	傳入使用者名稱。如果useCHAP=true則需要。適用於「ONTAP-SAN」和「ONTAP-san經濟」

儲存平台機密欄位說明	秘密	欄位說明
ONTAP	chapInitiator機密	CHAP啟動器密碼。如果useCHAP=true則需要。適用於「ONTAP-SAN」和「ONTAP-san經濟」
ONTAP	chapTargetUsername	目標使用者名稱。如果useCHAP=true則需要。適用於「ONTAP-SAN」和「ONTAP-san經濟」
ONTAP	chapTargetInitiator機密	CHAP目標啟動器機密。如果useCHAP=true則需要。適用於「ONTAP-SAN」和「ONTAP-san經濟」

在此步驟中建立的機密會參照下一步所建立之「TridentBackendConfig」物件的「sapec.ecent」欄位。

步驟2：建立 TridentBackendConfig CR

您現在可以建立「TridentBackendConfig」的CR了。在此範例中、使用「ONTAP-SAN」驅動程式的後端是使用「TridentBackendConfig」物件建立、如下所示：

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

步驟3：確認的狀態 TridentBackendConfig CR

現在您已經建立了「TridentBackendConfig」（TridentBackendConfig）CR、您就可以驗證其狀態。請參閱下列範例：

```
kubectl -n trident get tbc backend-tbc-ontap-san
NAME                                BACKEND NAME                BACKEND UUID
PHASE    STATUS
backend-tbc-ontap-san    ontap-san-backend          8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8    Bound    Success
```

已成功建立後端、並連結至「TridentBackendConfig」CR。

階段可以採用下列其中一個值：

- Bound：TridentBackendConfig CR與後端相關聯、且後端包含 configRef 設定為 TridentBackendConfig CR 的 uid。
- 《Unbound》：使用「U」表示。「TridentBackendConfig」物件不會繫結至後端。根據預設、所有新建立的「TridentBackendConfig」CRS均處於此階段。階段變更之後、就無法再恢復為Unbound（未綁定）。
- Deleting：TridentBackendConfig CR 的 deletionPolicy 已設定為刪除。當 TridentBackendConfig 系統會刪除CR、並轉換為「刪除」狀態。
 - 如果後端不存在持續磁碟區宣告（PVCS）、刪除 TridentBackendConfig、將會導致 Trident 刪除後端和 `TridentBackendConfig` CR。
 - 如果後端上有一個或多個PVCS、則會進入刪除狀態。隨後、「TridentBackendConfig」CR也會進入刪除階段。只有刪除所有的PVCS之後、才會刪除後端和「TridentBackendConfig」。
- 「遺失」：與「TridentBackendConfig」CR相關的後端意外或刻意刪除、而「TridentBackendConfig」CR 仍有刪除後端的參考資料。無論「刪除原則」值為何、「TridentBackendConfig」CR仍可刪除。
- Unknown：Trident 無法確定與 CR 關聯的後端的狀態或存在 TridentBackendConfig。例如、如果 API 伺服器沒有回應、或 tridentbackends.trident.netapp.io CRD 遺失。這可能需要介入。

在此階段、成功建立後端！還有多種作業可以額外處理、例如 ["後端更新和後端刪除"](#)。

(選用) 步驟4：取得更多詳細資料

您可以執行下列命令來取得有關後端的詳細資訊：

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                                BACKEND NAME                BACKEND UUID
PHASE    STATUS    STORAGE DRIVER    DELETION POLICY
backend-tbc-ontap-san    ontap-san-backend          8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8    Bound    Success    ontap-san    delete
```

此外、您也可以取得「TridentBackendConfig」的YAML/Json傾印。

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: "2021-04-21T20:45:11Z"
  finalizers:
  - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound

```

backendInfo 包含回應 CR 所建立後端 TridentBackendConfig 的 backendName 和 backendUUID。此 lastOperationStatus 欄位代表 CR 上次操作的狀態 TridentBackendConfig 可由使用者觸發（例如、使用者在中變更項目 spec）或由 Trident 觸發（例如、在 Trident 重新啟動期間）。可能是「成功」或「失敗」。phase 代表 CR 與後端之間關係的狀態 TridentBackendConfig。在上述範例中、phase 有值界限、表示 TridentBackendConfig CR 與後端相關聯。

您可以執行「`kubectl -n trident describe tbc <tbc-cr-name>`」命令、以取得事件記錄的詳細資料。



您無法使用「tridentctl」來更新或刪除包含相關「TridentBackendConfig」物件的後端。若要瞭解在「tridentctl」和「TridentBackendConfig」之間切換的步驟、["請參閱此處"](#)。

管理後端

以 **KECBEVCL** 執行後端管理

瞭解如何使用「`kubectl`」來執行後端管理作業。

刪除後端

刪除 `TridentBackendConfig` 後、您會指示 Trident 刪除 / 保留後端（根據 `deletionPolicy`）。若要刪除後端、請確定已 `deletionPolicy` 設定為刪除。若要僅刪除 `TridentBackendConfig`、請確定已 `deletionPolicy` 設定為保留。這可確保後端仍存在、並可使用進行管理 `tridentctl`。

執行下列命令：

```
kubectl delete tbc <tbc-name> -n trident
```

Trident 不會刪除使用中的 Kubernetes 機密 `TridentBackendConfig`。Kubernetes 使用者負責清除機密。刪除機密時必須小心。只有在後端未使用機密時、才應刪除這些機密。

檢視現有的後端

執行下列命令：

```
kubectl get tbc -n trident
```

您也可以執行「`tridentctl Get backend -n trident`」或「`tridentctl Get backend -o yaml -n trident`」、以取得所有後端的清單。這份清單也會包含以「`tridentctl`」建立的後端。

更新後端

更新後端可能有多種原因：

- 儲存系統的認證資料已變更。若要更新認證、必須更新物件中使用的 Kubernetes Secret `TridentBackendConfig`。Trident 會使用提供的最新認證、自動更新後端。執行下列命令以更新 Kubernetes Secret：

```
kubectl apply -f <updated-secret-file.yaml> -n trident
```

- 需要 ONTAP 更新參數（例如使用的 SVM 名稱）。
 - 您可以更新 `TridentBackendConfig` 使用下列命令直接透過 Kubernetes 執行物件：

```
kubectl apply -f <updated-backend-file.yaml>
```

- 或者、您也可以變更現有的 `TridentBackendConfig` 使用下列命令的 CR：

```
kubectl edit tbc <tbc-name> -n trident
```



- 如果後端更新失敗、後端仍會繼續維持其最後已知的組態。您可以執行「`kubectl Get tbc <tbc-name>-o yaml -n trident`」或「`kubectl描述tbc <tbc-name>-n trident`」來檢視記錄以判斷原因。
- 識別並修正組態檔的問題之後、即可重新執行`update`命令。

使用`tridentctl`執行後端管理

瞭解如何使用「`tridentctl`」來執行後端管理作業。

建立後端

建立之後 "**後端組態檔**"，執行下列命令：

```
tridentctl create backend -f <backend-file> -n trident
```

如果後端建立失敗、表示後端組態有問題。您可以執行下列命令來檢視記錄、以判斷原因：

```
tridentctl logs -n trident
```

識別並修正組態檔的問題之後、您只需再次執行「`create`」命令即可。

刪除後端

若要從 Trident 刪除後端、請執行下列步驟：

1. 擷取後端名稱：

```
tridentctl get backend -n trident
```

2. 刪除後端：

```
tridentctl delete backend <backend-name> -n trident
```



如果 Trident 已從這個後端佈建磁碟區和快照、但該後端仍存在、則刪除後端將會阻止新磁碟區由其進行佈建。後端將繼續處於「刪除」狀態、而Trident將繼續管理這些磁碟區和快照、直到它們被刪除為止。

檢視現有的後端

若要檢視Trident知道的後端、請執行下列步驟：

- 若要取得摘要、請執行下列命令：

```
tridentctl get backend -n trident
```

- 若要取得所有詳細資料、請執行下列命令：

```
tridentctl get backend -o json -n trident
```

更新後端

建立新的後端組態檔之後、請執行下列命令：

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

如果後端更新失敗、表示後端組態有問題、或是您嘗試了無效的更新。您可以執行下列命令來檢視記錄、以判斷原因：

```
tridentctl logs -n trident
```

識別並修正組態檔的問題之後、您只需再次執行「update」命令即可。

識別使用後端的儲存類別

這是您可以用Json回答的問題類型範例、其中的「tridentctl」會輸出後端物件。這會使用您需要安裝的「jq」公用程式。

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name, storageClasses: [.storage[].storageClasses]|unique}]'
```

這也適用於使用「TridentBackendConfig」建立的後端。

在後端管理選項之間切換

瞭解在 Trident 中管理後端的不同方法。

管理後端的選項

隨之推出「TridentBackendConfig」管理員現在有兩種獨特的後端管理方法。這會提出下列問題：

- 使用「tridentctl」建立的後端、是否能以「TridentBackendConfig」來管理？
- 使用「TridentBackendConfig」建立的後端、是否可以使用「tridentctl」來管理？

本節說明透過Kubernetes介面建立「TridentBackendConfig」物件、直接透過「tridentctl」建立的後端管理所需的步驟。

這將適用於下列案例：

- 沒有的既有後端 TridentBackendConfig 因為它們是使用建立的 tridentctl。
- 使用「tridentctl」建立的新後端、而其他「TridentBackendConfig」物件則存在。

在這兩種情況下、都會繼續出現後端、並在 Trident 排程磁碟區上運作。系統管理員有兩種選擇之一：

- 繼續使用「tridentctl」來管理使用它建立的後端。
- 將使用「tridentctl」建立的後端連結至新的「TridentBackendConfig」物件。這樣做將意味着後端將使用“kubedl”而不是“tridentctl”來管理。

若要使用「kubedl」管理預先存在的後端、您需要建立連結至現有後端的「TridentBackendConfig」。以下是如何運作的總覽：

1. 建立Kubernetes機密。機密包含 Trident 與儲存叢集 / 服務通訊所需的認證。
2. 建立「TridentBackendConfig」物件。其中包含有關儲存叢集/服務的詳細資訊、並參考上一步建立的機密。必須謹慎指定相同的組態參數（例如「s.pec.backendName」、「sec.storagePrefix」、「sPEec.storageDriverName」等）。必須將「Pec.backendName」設定為現有後端的名稱。

步驟0：識別後端

以建立 TridentBackendConfig 若要連結至現有的後端、您必須取得後端組態。在此範例中、假設使用下列Json定義建立後端：

```
tridentctl get backend ontap-nas-backend -n trident
+-----+-----+
+-----+-----+-----+-----+
|          NAME          | STORAGE DRIVER |          UUID          |
| STATE  | VOLUMES |          |          |
+-----+-----+-----+-----+
| ontap-nas-backend    | ontap-nas      | 52f2eb10-e4c6-4160-99fc- |
| 96b3be5ab5d7 | online |          25 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

cat ontap-nas-backend.json

{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.10.10.1",
  "dataLIF": "10.10.10.2",
```

```
"backendName": "ontap-nas-backend",
"svm": "trident_svm",
"username": "cluster-admin",
"password": "admin-password",

"defaults": {
  "spaceReserve": "none",
  "encryption": "false"
},
"labels":{"store":"nas_store"},
"region": "us_east_1",
"storage": [
  {
    "labels":{"app":"msoffice", "cost":"100"},
    "zone":"us_east_1a",
    "defaults": {
      "spaceReserve": "volume",
      "encryption": "true",
      "unixPermissions": "0755"
    }
  },
  {
    "labels":{"app":"mysqldb", "cost":"25"},
    "zone":"us_east_1d",
    "defaults": {
      "spaceReserve": "volume",
      "encryption": "false",
      "unixPermissions": "0775"
    }
  }
]
}
```

步驟1：建立Kubernetes機密

建立包含後端認證的秘密、如以下範例所示：

```
cat tbc-ontap-nas-backend-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password

kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

步驟2：建立 TridentBackendConfig CR

下一步是建立一個「TridentBackendConfig」（TridentBackendConfig）CR、它會自動連結至現有的「ONTAP-NAS-backend」（如本範例所示）。確保符合下列要求：

- 相同的後端名稱是在「s.pec.backendName」中定義。
- 組態參數與原始後端相同。
- 虛擬資源池（若有）必須維持與原始後端相同的順序。
- 認證資料是透過Kubernetes Secret提供、而非以純文字提供。

在這種情況下、「TridentBackendConfig」將會如下所示：

```

cat backend-tbc-ontap-nas.yaml
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
    region: us_east_1
  storage:
  - labels:
    app: msoffice
    cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
    app: mysqldb
    cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'

kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created

```

步驟3：確認的狀態 TridentBackendConfig **CR**

在建立「TridentBackendConfig」之後、其階段必須是「綁定」。它也應反映與現有後端相同的後端名稱和UUID。

```

kubect1 get tbc tbc-ontap-nas-backend -n trident
NAME                                BACKEND NAME                BACKEND UUID
PHASE    STATUS
tbc-ontap-nas-backend  ontap-nas-backend          52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7    Bound    Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+-----+-----+
+-----+-----+-----+-----+
|          NAME          | STORAGE DRIVER |          UUID
| STATE  | VOLUMES |
+-----+-----+-----+-----+
| ontap-nas-backend     | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |          25 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

現在可以使用「tbc-ontap-nas-backend」 「TridentBackendConfig」物件來完全管理後端。

管理 TridentBackendConfig 後端使用 tridentctl

可以使用「tridentctl」來列出使用「TridentBackendConfig」建立的後端。此外、系統管理員也可以刪除「TridentBackendConfig」、並確定「pec.deletionPolicy」設為「效能」、藉此選擇透過「tridentctl」來完全管理此類後端。

步驟0：識別後端

例如、假設使用「TridentBackendConfig」建立下列後端：

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                                BACKEND NAME          BACKEND UUID
PHASE  STATUS    STORAGE DRIVER  DELETION POLICY
backend-tbc-ontap-san  ontap-san-backend    81abcb27-ea63-49bb-b606-
0a5315ac5f82    Bound    Success    ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-----+-----+
+-----+-----+-----+-----+-----+
|          NAME          | STORAGE DRIVER |                               UUID
| STATE  | VOLUMES |
+-----+-----+-----+-----+-----+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |          33 |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
```

從輸出中可以看出這一點 TridentBackendConfig 已成功建立並繫結至後端 [觀察後端的 UUID] 。

步驟1：確認 deletionPolicy 設為 retain

讓我們來看看的價值 deletionPolicy。這需要設為 retain。如此可確保刪除 CR 時 TridentBackendConfig、後端定義仍會存在、並可透過進行管理 tridentctl。

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                                BACKEND NAME          BACKEND UUID
PHASE  STATUS    STORAGE DRIVER  DELETION POLICY
backend-tbc-ontap-san  ontap-san-backend    81abcb27-ea63-49bb-b606-
0a5315ac5f82    Bound    Success    ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                                BACKEND NAME          BACKEND UUID
PHASE  STATUS    STORAGE DRIVER  DELETION POLICY
backend-tbc-ontap-san  ontap-san-backend    81abcb27-ea63-49bb-b606-
0a5315ac5f82    Bound    Success    ontap-san        retain
```



除非將「刪除原則」設定為「需要」、否則請勿繼續下一步。

步驟2：刪除 TridentBackendConfig CR

最後一個步驟是刪除「TridentBackendConfig」（TridentBackendConfig）。確認「刪除原則」設為「保留」之後、您可以繼續刪除：

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-----+-----+
+-----+-----+-----+-----+
|          NAME          | STORAGE DRIVER |          UUID          |
| STATE  | VOLUMES |
+-----+-----+-----+-----+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-0a5315ac5f82 |
| online |      33 |
+-----+-----+-----+-----+
```

刪除物件後 TridentBackendConfig、Trident 只是將其移除、而不會實際刪除後端本身。

建立及管理儲存類別

建立儲存類別

設定 Kubernetes StorageClass 物件並建立儲存類別、以指示 Trident 如何配置磁碟區。

設定 Kubernetes StorageClass 物件

會 "[Kubernetes StorageClass 物件](#)"將 Trident 識別為該類別所使用的資源配置程式、並指示 Trident 如何資源配置 Volume。例如：

```
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
```

如需儲存類別如何與互動的詳細資訊 PersistentVolumeClaim、以及控制 Trident 配置磁碟區的參數、請參

閱"[Kubernetes和Trident物件](#)"。

建立儲存類別

建立 StorageClass 物件之後、即可建立儲存類別。 [\[儲存類別範例\]](#) 提供一些您可以使用或修改的基本範例。

步驟

1. 這是 Kubernetes 物件、請使用 `kubectl` 在Kubernetes中建立。

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. 現在您應該會在 Kubernetes 和 Trident 中同時看到 *base-csi* 儲存類別、而 Trident 應該已經在後端上探索到這些集區。

```
kubectl get sc basic-csi
NAME          PROVISIONER          AGE
basic-csi     csi.trident.netapp.io 15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

儲存類別範例

Trident 提供 "特定後端的簡單儲存類別定義"。

或者、您也可以編輯 `sample-input/storage-class-csi.yaml.templ` 安裝程式隨附並取代的檔案 `BACKEND_TYPE` 儲存驅動程式名稱。

```
./tridentctl -n trident get backend
+-----+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |         0 |
+-----+-----+-----+-----+
+-----+-----+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

管理儲存類別

您可以檢視現有的儲存類別、設定預設的儲存類別、識別儲存類別後端、以及刪除儲存類別。

檢視現有的儲存類別

- 若要檢視現有的Kubernetes儲存類別、請執行下列命令：

```
kubectl get storageclass
```

- 若要檢視Kubernetes儲存類別詳細資料、請執行下列命令：

```
kubectl get storageclass <storage-class> -o json
```

- 若要檢視 Trident 的同步儲存類別、請執行下列命令：

```
tridentctl get storageclass
```

- 若要檢視 Trident 的同步儲存類別詳細資料、請執行下列命令：

```
tridentctl get storageclass <storage-class> -o json
```

設定預設儲存類別

Kubernetes 1.6 新增了設定預設儲存類別的功能。如果使用者未在「持續磁碟區宣告」(PVC) 中指定一個、則此儲存類別將用於配置「持續磁碟區」。

- 在儲存類別定義中、將「storageclass.Kubernetes.IO/as-default-Class」註釋設為 true、以定義預設儲存類別。根據規格、任何其他值或不存在的附註都會解譯為假。
- 您可以使用下列命令、將現有的儲存類別設定為預設的儲存類別：

```
kubectl patch storageclass <storage-class-name> -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- 同樣地、您也可以使用下列命令移除預設儲存類別註釋：

```
kubectl patch storageclass <storage-class-name> -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Trident 安裝程式套件中也有包含此附註的範例。



叢集中一次只應有一個預設儲存類別。Kubernetes 在技術上並不妨礙您擁有多個儲存類別、但它的行為方式就如同完全沒有預設的儲存類別一樣。

識別儲存類別的後端

這是您可以使用 JSON 來回答的問題類型範例、此問題 `tridentctl` 會針對 Trident 後端物件輸出。這會使用 `jq` 您可能需要先安裝的公用程式。

```
tridentctl get storageclass -o json | jq '[.items[] | {storageClass: .Config.name, backends: [.storage]|unique}]'
```

刪除儲存類別

若要從 Kubernetes 刪除儲存類別、請執行下列命令：

```
kubectl delete storageclass <storage-class>
```

「<storage-class>」應改用您的儲存類別。

透過此儲存類別建立的任何持續磁碟區都將保持不變、Trident 將繼續管理這些磁碟區。



Trident 會對其建立的磁碟區強制執行空白 fsType。對於 iSCSI 後端、建議在 StorageClass 中強制執行 parameters.fsType。您應該刪除現有的 StorageClasses、然後使用指定的方式重新建立它們 parameters.fsType。

資源配置與管理磁碟區

配置 Volume

建立 PersistentVolume (PV) 和 PersistentVolume Claim (PVC)、使用設定的 Kubernetes StorageClass 來要求存取 PV。然後、您可以將 PV 掛載至 Pod。

總覽

答 "[PersistentVolume](#)" (PV) 是叢集管理員在 Kubernetes 叢集上配置的實體儲存資源。。
"[PersistentVolume Claim](#)" (PVC) 是存取叢集上 PersistentVolume 的要求。

可將 PVC 設定為要求儲存特定大小或存取模式。叢集管理員可以使用相關的 StorageClass 來控制超過 PersistentVolume 大小和存取模式的權限、例如效能或服務層級。

建立 PV 和 PVC 之後、您可以將磁碟區裝入 Pod。

範例資訊清單

PersistentVolume 範例資訊清單

此範例資訊清單顯示與 StorageClass 相關的 10Gi 基本 PV basic-csi。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/my/host/path"
```

PersistentVolume Claim 範例資訊清單

這些範例顯示基本的 PVC 組態選項。

可存取 **RWO** 的 **PVC**

此範例顯示具有 `rwo` 存取權的基本 PVC、與命名的 StorageClass 相關聯 `basic-csi`。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

採用 **NVMe / TCP** 的 **PVC**

此範例顯示 NVMe / TCP 的基本 PVC、並提供與命名 StorageClass 相關的 `rwo` 存取 `protection-gold`。

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san-nvme
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 300Mi
  storageClassName: protection-gold
```

Pod 資訊清單範例

這些範例顯示將 PVC 連接至 Pod 的基本組態。

基本組態

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
        claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

基本 NVMe / TCP 組態

```
---
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
    volumeMounts:
    - mountPath: "/usr/share/nginx/html"
      name: task-pv-storage
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
  - name: task-pv-storage
    persistentVolumeClaim:
      claimName: pvc-san-nvme
```

建立 PV 和 PVC

步驟

1. 建立 PV。

```
kubectl create -f pv.yaml
```

2. 確認 PV 狀態。

```
kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS  REASON    AGE
pv-storage    4Gi       RWO           Retain          Available
7s
```

3. 建立 PVC。

```
kubectl create -f pvc.yaml
```

4. 確認 PVC 狀態。

```
kubectl get pvc
NAME          STATUS VOLUME          CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage  Bound  pv-name 2Gi          RWO          5m
```

5. 將磁碟區裝入 Pod。

```
kubectl create -f pv-pod.yaml
```



您可以使用監控進度 `kubectl get pod --watch`。

6. 確認磁碟區已掛載到上 `/my/mount/path`。

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

7. 您現在可以刪除 Pod。Pod 應用程式將不再存在、但該磁碟區仍會保留。

```
kubectl delete pod task-pv-pod
```

如需儲存類別如何與互動的詳細資訊 `PersistentVolumeClaim`、以及控制 `Trident` 配置磁碟區的參數、請參閱"[Kubernetes和Trident物件](#)"。

展開Volume

`Trident` 可讓 `Kubernetes` 使用者在建立磁碟區之後擴充其容量。尋找擴充 `iSCSI` 和 `NFS` 磁碟區所需組態的相關資訊。

展開 `iSCSI Volume`

您可以使用「`SCSI`資源配置程式」來擴充 `iSCSI` 持續磁碟區 (PV)。



`iSCSI` 磁碟區擴充支援「`ontap-san`」、`「ONTAP-san經濟」`、`「Poolidfire-san`」等驅動程式、需要 `Kubernetes` 1.16 及更新版本。

步驟1：設定 `StorageClass` 以支援 `Volume` 擴充

編輯 `StorageClass` 定義以設定 `allowVolumeExpansion` 欄位至 `true`。

```
cat storageclass-ontapsan.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

對於已存在的StorageClass、請編輯此類以包含「allowVolume Expansion」參數。

步驟2：使用您建立的StorageClass建立一個永久虛擬儲存設備

編輯 PVC 定義並更新 spec.resources.requests.storage 以反映新的所需大小、此大小必須大於原始大小。

```
cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident 會建立持續 Volume (PV)、並將其與此持續 Volume Claim (PVC) 相關聯。

```

kubect1 get pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound     pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi
RWO          ontap-san    8s

kubect1 get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY  STATUS    CLAIM          STORAGECLASS  REASON  AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi      RWO
Delete          Bound     default/san-pvc  ontap-san    10s

```

步驟3：定義一個連接至PVC的Pod

將 PV 附加至 Pod 、以便調整大小。調整iSCSI PV的大小有兩種情況：

- 如果 PV 附加至 Pod 、 Trident 會在儲存後端擴充磁碟區、重新掃描裝置、並調整檔案系統的大小。
- 當嘗試調整未附加 PV 的大小時、 Trident 會在儲存後端擴充磁碟區。在將永久虛擬磁碟綁定至Pod之後、Trident會重新掃描裝置並重新調整檔案系統的大小。然後、Kubernetes會在擴充作業成功完成後、更新PVC大小。

在此範例中、會建立使用「shan -PVC」的Pod。

```

kubect1 get pod
NAME          READY    STATUS    RESTARTS  AGE
ubuntu-pod   1/1     Running   0          65s

kubect1 describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:     1Gi
Access Modes:  RWO
VolumeMode:   Filesystem
Mounted By:    ubuntu-pod

```

步驟4：展開PV

若要調整從1Gi建立至2Gi的PV大小、請編輯PVC定義、並將「sec.resumes.requests.storage」更新為2Gi。

```
kubectl edit pvc san-pvc
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
  ...
```

步驟5：驗證擴充

您可以檢查 PVC、PV 和 Trident Volume 的大小、以驗證擴充是否正常運作：

```

kubect1 get pvc san-pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound      pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO          ontap-san    11m
kubect1 get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY  STATUS    CLAIM          STORAGECLASS  REASON  AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi      RWO
Delete          Bound      default/san-pvc  ontap-san    12m
tridentctl get volumes -n trident
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          |  STATE  | MANAGED |
+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san    |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----+-----+-----+
+-----+-----+-----+-----+

```

展開NFS Volume

Trident 支援 NFS PV 的 Volume 擴充、部署於 ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、gcp-cvs 和 azure-netapp-files 後端。

步驟1：設定StorageClass以支援Volume擴充

若要調整NFS PV的大小、管理員必須先將「allowVolumeExpansion」欄位設定為「true」、以設定儲存類別以允許磁碟區擴充：

```

cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true

```

如果您已經建立了沒有此選項的儲存類別、只要使用「kubect1 Edit storageclass」來編輯現有的儲存類別、即可進行磁碟區擴充。

步驟2：使用您建立的StorageClass建立一個永久虛擬儲存設備

```
cat pvc-ontapnas.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident 應為此 PVC 建立 20MiB NFS PV：

```
kubectl get pvc
NAME                STATUS    VOLUME
CAPACITY            ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb       Bound      pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi
RWO                 ontapnas      9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                CAPACITY  ACCESS MODES
RECLAIM POLICY     STATUS    CLAIM                STORAGECLASS  REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi      RWO
Delete             Bound     default/ontapnas20mb  ontapnas
2m42s
```

步驟3：展開PV

若要將新建立的20MiB PV調整至1GiB、請編輯該PVC並設定組合 `spec.resources.requests.storage` 至 1GiB：

```
kubectl edit pvc ontapnas20mb
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  ...
```

步驟4：驗證擴充

您可以檢查 PVC、PV 和 Trident Volume 的大小、以驗證調整大小是否正常運作：

```

kubect1 get pvc ontapnas20mb
NAME                STATUS      VOLUME
CAPACITY  ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb  Bound        pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi
RWO                ontapnas                4m44s

kubect1 get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                CAPACITY  ACCESS MODES
RECLAIM POLICY     STATUS    CLAIM                STORAGECLASS  REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi      RWO
Delete            Bound     default/ontapnas20mb  ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          |  STATE  |  MANAGED  |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file      | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

匯入磁碟區

您可以使用「tridentctl匯入」將現有的儲存磁碟區匯入為Kubernetes PV。

總覽與考量

您可以將磁碟區匯入 Trident、以便：

- 將應用程式容器化、並重新使用其現有的資料集
- 針對臨時應用程式使用資料集的複本
- 重建故障的 Kubernetes 叢集
- 在災難恢復期間移轉應用程式資料

考量

匯入 Volume 之前、請先檢閱下列考量事項。

- Trident 只能匯入 RW（讀寫）類型的 ONTAP Volume。DP（資料保護）類型磁碟區是 SnapMirror 目的地的磁碟區。您應該先中斷鏡射關係、再將磁碟區匯入 Trident。

- 我們建議您在沒有作用中連線的情況下匯入磁碟區。若要匯入使用中的 Volume、請複製該 Volume、然後執行匯入。



這對區塊磁碟區特別重要、因為 Kubernetes 不會知道先前的連線、而且很容易將作用中的磁碟區附加到 Pod。這可能導致資料毀損。

- 雖然必須在 PVC 上指定、但 StorageClass Trident 在匯入期間不會使用此參數。建立磁碟區時會使用儲存類別、根據儲存特性從可用的集區中選取。由於該磁碟區已經存在、因此在匯入期間不需要選取任何集區。因此、即使磁碟區存在於與 PVC 中指定的儲存類別不相符的後端或集區、匯入也不會失敗。
- 現有的 Volume 大小是在 PVC 中決定和設定的。儲存驅動程式匯入磁碟區之後、PV 會以 PVC 的 ClaimRef 建立。
 - 回收原則一開始設定為 retain 在 PV 中。Kubernetes 成功繫結了 PVC 和 PV 之後、系統會更新回收原則以符合儲存類別的回收原則。
 - 如果儲存類別的回收原則為 delete、儲存磁碟區會在 PV 刪除時刪除。
- 根據預設、Trident 會管理 PVC 並重新命名後端上的 FlexVol 和 LUN。您可以傳遞 `--no-manage` 旗標來匯入未受管理的磁碟區。如果您使用 `--no-manage`、Trident 在物件生命週期內不會在 PVC 或 PV 上執行任何其他作業。刪除 PV 時不會刪除儲存磁碟區、也會忽略其他操作、例如 Volume Clone 和 Volume resize。



如果您想要將 Kubernetes 用於容器化工作負載、但想要管理 Kubernetes 以外儲存磁碟區的生命週期、則此選項非常實用。

- 將註釋新增至 PVC 和 PV、這有兩種用途、表示已匯入磁碟區、以及是否管理了 PVC 和 PV。不應修改或移除此附註。

匯入 Volume

您可以使用 `tridentctl import` 匯入 Volume。

步驟

1. 建立持續 Volume Claim (PVC) 檔案 (例如、`pvc.yaml`) 用於建立 PVC。PVC 檔案應包含在內 `name`、`namespace`、`accessModes` 和 `storageClassName`。您也可以指定 `unixPermissions` 在您的 PVC 定義中。

以下是最低規格的範例：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```



請勿包含其他參數、例如 PV 名稱或 Volume 大小。這可能會導致匯入命令失敗。

2. 使用 `tridentctl import` 命令指定 Trident 後端的名稱、該後端包含磁碟區、以及唯一識別儲存區中磁碟區的名稱（例如：ONTAP FlexVol、Element Volume、Cloud Volumes Service 路徑）。`-f` 需要引數來指定 PVC 檔案的路徑。

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-  
file>
```

範例

請參閱下列 Volume 匯入範例、瞭解支援的驅動程式。

ONTAP NAS 和 ONTAP NAS FlexGroup

Trident 支援使用和 `ontap-nas-flexgroup` 驅動程式進行 Volume 匯入 `ontap-nas`。



- `ontap-nas-economy` 驅動程式無法匯入及管理 `qtree`。
- `ontap-nas` 和 `ontap-nas-flexgroup` 驅動程式不允許重複的磁碟區名稱。

每個使用「ONTAP-NAS」驅動程式建立的 Volume FlexVol、都是 ONTAP 一個在整個叢集上的功能。使用「ONTAP-NAS」驅動程式匯入 FlexVols 的運作方式相同。已存在於某個叢集上的一個功能、可以匯入為「ONTAP-NAS」的 PVC。FlexVol ONTAP 同樣地 FlexGroup、也可以將此數據匯入為「ONTAP-NAS-Flexgroup」PVCS。

ONTAP NAS 範例

以下是託管 Volume 和非託管 Volume 匯入的範例。

託管 Volume

以下範例會匯入名為的 Volume managed_volume 在名為的後端上 ontap_nas :

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
file	pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	online	standard	true

非託管 Volume

使用 `--no-manage` 引數時、Trident 不會重新命名磁碟區。

以下範例匯入 unmanaged_volume 在上 ontap_nas 後端：

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file> --no-manage
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
file	pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	online	standard	false

SAN ONTAP

Trident 支援使用和 ontap-san-economy 驅動程式進行 Volume 匯入 `ontap-san`。

Trident 可以匯入包含單一 LUN 的 ONTAP SAN FlexVols。這與驅動程式一致 ontap-san、可為 FlexVol 中的每個 PVC 和 LUN 建立 FlexVol。Trident 會匯入 FlexVol、並將其與 PVC 定義相關聯。

ONTAP SAN 範例

以下是託管 Volume 和非託管 Volume 匯入的範例。

託管 Volume

對於託管卷，Trident 將 FlexVol 重命名為格式，並將 FlexVol 中的 LUN lun0 重命名為 pvc-<uuid>。

下列範例會匯入 ontap-san-managed 上的顯示 FlexVol ontap_san_default 後端：

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
block	pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a	cd394786-ddd5-4470-adc3-10c5ce4ca757	20 MiB	online	basic	true

非託管 Volume

以下範例匯入 unmanaged_example_volume 在上 ontap_san 後端：

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
block	pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228	e3275890-7d80-4af6-90cc-c7a0759f555a	1.0 GiB	online	san-blog	false

如果您將 LUN 對應至與 Kubernetes 節點 IQN 共用 IQN 的 igroup，如下列範例所示，您將會收到錯誤訊息：LUN already mapped to initiator(s) in this group。您需要移除啟動器或取消對應 LUN，才能匯入磁碟區。

Vserver	Igroup	Protocol	OS Type	Initiators
svm0	k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3	iscsi	linux	iqn.1994-05.com.redhat:4c2e1cf35e0
svm0	unmanaged-example-igroup	mixed	linux	iqn.1994-05.com.redhat:4c2e1cf35e0

元素

Trident 支援使用驅動程式的 NetApp Element 軟體和 NetApp HCI Volume 匯入 solidfire-san。



Element 驅動程式支援重複的 Volume 名稱。不過、如果有重複的磁碟區名稱、Trident 會傳回錯誤。因應措施是複製磁碟區、提供唯一的磁碟區名稱、然後匯入複製的磁碟區。

元素範例

下列範例會匯入 element-managed 後端上的 Volume element_default。

```
tridentctl import volume element_default element-managed -f pvc-basic-import.yaml -n trident -d
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STORAGE CLASS	STATE	MANAGED
block	pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe	d3ba047a-ea0b-43f9-9c42-e38e58301c49	10 GiB	basic-element	online	true

Google Cloud Platform

Trident 支援使用驅動程式進行 Volume 匯入 gcp-cvs。



若要在 Google Cloud Platform 中匯入以 NetApp Cloud Volumes Service 為後盾的 Volume、請依其 Volume 路徑識別該 Volume。Volume 路徑是之後 Volume 匯出路徑的一部分 :/。例如、如果匯出路徑為 10.0.0.1:/adroit-jolly-swift、磁碟區路徑為 adroit-jolly-swift。

Google Cloud Platform 範例

下列範例會匯入 gcp-cvs 後端上的 Volume gcpcvs_YEppr 的磁碟區路徑 adroit-jolly-swift。

```
tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-pvc-
file> -n trident
```

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | gcp-storage | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

Azure NetApp Files

Trident 支援使用驅動程式進行 Volume 匯入 azure-netapp-files。



若要匯入 Azure NetApp Files Volume、請依磁碟區路徑識別該磁碟區。Volume 路徑是之後 Volume 匯出路徑的一部分：/。例如、如果掛載路徑為 10.0.0.2:/importvol1、磁碟區路徑為 importvol1。

Azure NetApp Files 範例

下列範例會匯入 azure-netapp-files 後端上的 Volume azurenetappfiles_40517 磁碟區路徑 importvol1。

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident
```

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage |
file      | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

自訂磁碟區名稱和標籤

有了 Trident、您就可以為您建立的磁碟區指派有意義的名稱和標籤。這有助於您識別並輕鬆地將磁碟區對應至各自的 Kubernetes 資源（PVCS）。您也可以後端層級定義範

本、以建立自訂磁碟區名稱和自訂標籤；您建立、匯入或複製的任何磁碟區都會遵守這些範本。

開始之前

可自訂的 Volume 名稱和標籤支援：

1. Volume 建立、匯入及複製作業。
2. 在 ONTAP NAS 經濟驅動程式的情況下、只有 Qtree Volume 的名稱符合名稱範本。
3. 在 ONTAP SAN 經濟型驅動程式的情況下、只有 LUN 名稱符合名稱範本。

限制

1. 可自訂的磁碟區名稱僅與 ONTAP 內部部署驅動程式相容。
2. 可自訂的 Volume 名稱不適用於現有的 Volume 。

可自訂 **Volume** 名稱的主要行為

1. 如果名稱範本中的語法無效而導致失敗、則後端建立會失敗。但是、如果範本應用程式失敗、則會根據現有的命名慣例來命名磁碟區。
2. 如果使用後端組態的名稱範本命名磁碟區、則不適用儲存前置詞。任何所需的前置字元值都可以直接新增至範本。

名稱範本和標籤的後端組態範例

自訂名稱範本可在根和 / 或集區層級定義。

根層級範例

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
    "{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.RequestName}}"
  },
  "labels": {"cluster": "ClusterA", "PVC":
    "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

集區層級範例

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels":{"labelname":"label1", "name": "{{ .volume.Name }}"},
      "defaults":
      {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
        }}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels":{"cluster":"label2", "name": "{{ .volume.Name }}"},
      "defaults":
      {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
        }}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

名稱範本範例

- 範例 1* :

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

- 範例 2* :

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}"
```

需要考量的重點

1. 在 Volume 匯入的情況下、只有現有的 Volume 具有特定格式的標籤時、標籤才會更新。例如 `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}:`。
2. 在託管 Volume 匯入的情況下、Volume 名稱會遵循在後端定義的根層級所定義的名稱範本。
3. Trident 不支援使用含有儲存前置碼的 Slice 運算子。
4. 如果範本未產生唯一的磁碟區名稱、Trident 會附加幾個隨機字元、以建立唯一的磁碟區名稱。
5. 如果 NAS 經濟 Volume 的自訂名稱長度超過 64 個字元、Trident 會根據現有的命名慣例來命名磁碟區。對於所有其他 ONTAP 驅動程式、如果磁碟區名稱超過名稱限制、磁碟區建立程序就會失敗。

跨命名空間共用NFS磁碟區

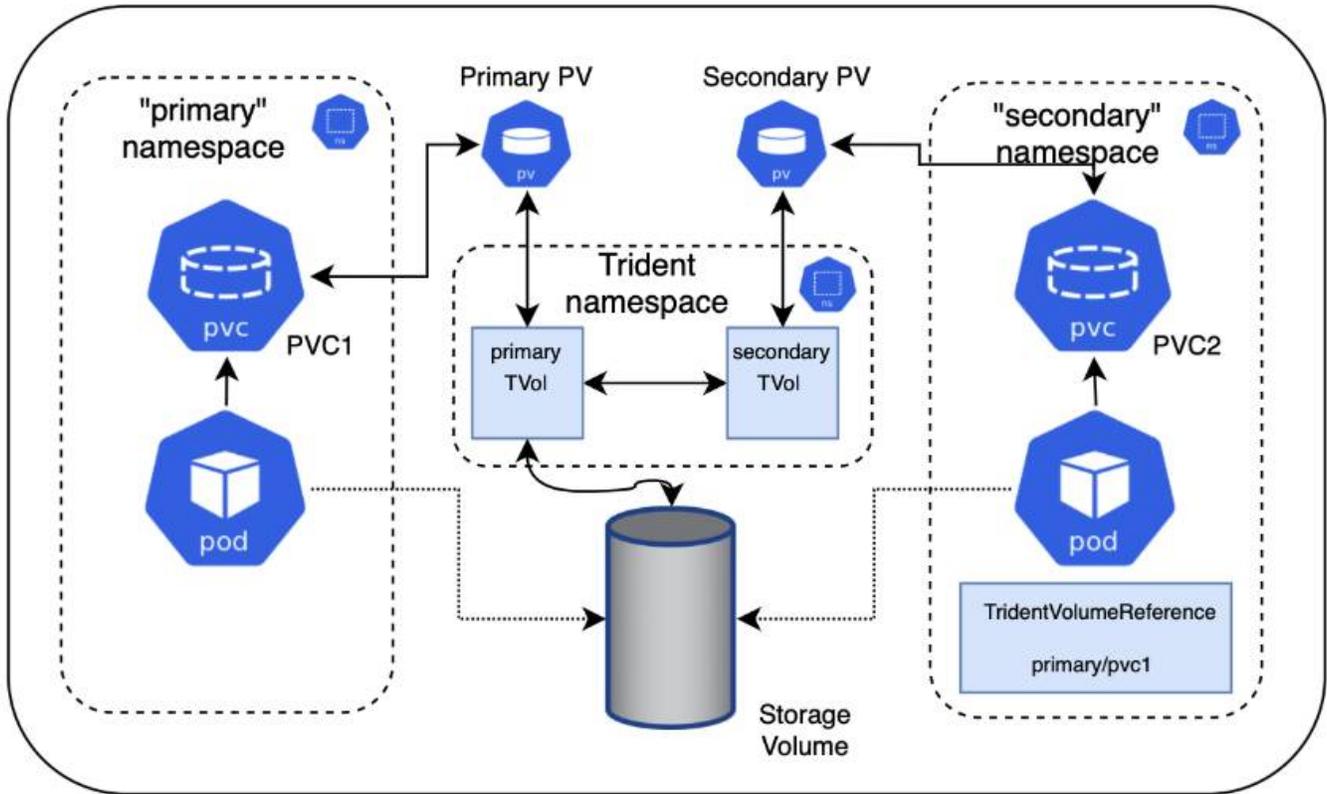
使用 Trident 、您可以在主要命名空間中建立磁碟區、並在一或多個次要命名空間中共用該磁碟區。

功能

TridentVolume Reference CR 可讓您安全地跨一或多個 Kubernetes 命名空間共用 ReadWriteMany (rwx) NFS 磁碟區。此Kubernetes原生解決方案具有下列優點：

- 多層存取控制、確保安全性
- 可搭配所有Trident NFS Volume驅動程式使用
- 不依賴tridentctl或任何其他非原生Kubernetes功能

此圖說明兩個Kubernetes命名空間之間的NFS Volume共用。



快速入門

您只需幾個步驟就能設定 NFS Volume 共享。

1

設定來源 PVC 以共用磁碟區

來源命名空間擁有人授予存取來源 PVC 中資料的權限。

2

授予在目的地命名空間中建立 CR 的權限

叢集管理員授予目的地命名空間擁有人建立 Trident Volume Reference CR 的權限。

3

在目的地命名空間中建立 Trident Volume Reference

目的地命名空間的擁有人會建立 Trident Volume Reference CR 來參照來源 PVC。

4

在目的地命名空間中建立從屬的 PVC

目的地命名空間的擁有人會建立從屬的 PVC、以使用來源 PVC 的資料來源。

設定來源和目的地命名空間

為了確保安全性、跨命名空間共用需要來源命名空間擁有人、叢集管理員和目的地命名空間擁有者的協同作業與行動。使用者角色會在每個步驟中指定。

步驟

1. *來源命名空間擁有者：*建立PVC (pvc1) (namespace2) 使用 shareToNamespace 註釋：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident 會建立 PV 及其後端 NFS 儲存磁碟區。



- 您可以使用以逗號分隔的清單、將永久虛擬儲存設備共用至多個命名空間。例如、
trident.netapp.io/shareToNamespace:
namespace2, namespace3, namespace4 ◦
- 您可以使用共用至所有命名空間 *。例如、
trident.netapp.io/shareToNamespace: *
- 您可以更新PVC,以納入 shareToNamespace 隨時註釋。

2. *叢集管理：*建立自訂角色和Kubeconfig、以授予目的地命名空間擁有者權限、以便在目的地命名空間中建立TridentVolume Reference CR ◦
3. *目的地命名空間擁有者：*在參照來源命名空間的目的地命名空間中建立TridentVolume Reference CR pvc1 ◦

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. *目的地命名空間擁有者：*建立一個PVC (pvc2) (namespace2) 使用 shareFromPVC 註釋以指定來源PVC ◦

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```



目的地PVC的大小必須小於或等於來源PVC。

結果

Trident 會讀取 `shareFromPVC` 目的地 PVC 上的附註、並將目的地 PV 建立為次級磁碟區、而不會有本身的儲存資源指向來源 PV、並共用來源 PV 儲存資源。目的地的PVC和PV似乎正常連結。

刪除共享Volume

您可以刪除跨多個命名空間共用的磁碟區。Trident 將移除來源命名空間上的磁碟區存取權、並維護共用該磁碟區的其他命名空間的存取權。移除所有參照該 Volume 的命名空間時、Trident 會刪除該 Volume。

使用 tridentctl get 查詢從屬Volume

使用../ Trident 參考/ tridentctl.html[tridentctl 命令與選項]。

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

旗標：

- `-h, --help`：Volume的說明。
- `--parentOfSubordinate string`：將查詢限制在從屬來源Volume。
- `--subordinateOf string`：將查詢限制在Volume的下屬。

限制

- Trident 無法防止目的地命名空間寫入共用磁碟區。您應該使用檔案鎖定或其他程序來防止覆寫共用Volume 資料。

- 您無法藉由移除來撤銷對來源PVC的存取權 `shareToNamespace` 或 `shareFromNamespace` 註釋或刪除 `TridentVolumeReference` CR.若要撤銷存取權、您必須刪除從屬的PVC。
- 在從屬磁碟區上無法執行快照、複製和鏡射。

以取得更多資訊

若要深入瞭解跨命名空間Volume存取：

- 請造訪 ["在命名空間之間共用磁碟區：歡迎使用跨命名空間磁碟區存取"](#)。
- 觀看上的示範 ["NetAppTV"](#)。

使用「csi拓撲」

Trident 可以利用來選擇性地建立磁碟區、並將其附加至 Kubernetes 叢集中的節點 "[「csi拓撲」功能](#)"。

總覽

使用「csi拓撲」功能、可根據區域和可用性區域、限制對磁碟區的存取、只能存取一部分節點。如今、雲端供應商可讓Kubernetes管理員建立以區域為基礎的節點。節點可位於某個區域內的不同可用度區域、或位於不同區域之間。為了協助在多區域架構中為工作負載配置磁碟區、Trident 使用 CSI 拓撲。



深入瞭解「csi拓撲」功能 ["請按這裡"](#)。

Kubernetes提供兩種獨特的Volume繫結模式：

- 如果 `VolumeBindingMode` 設置為 `Immediate`，則 Trident 會在沒有任何拓撲感知的情況下創建卷。建立永久虛擬磁碟時、即會處理磁碟區繫結和動態資源配置。這是預設值 `VolumeBindingMode`、適用於不強制執行拓撲限制的叢集。持續磁碟區的建立不需依賴要求的 Pod 排程需求。
- 將「Volume BindingMode」設為「WaitForFirst消費者」時、會延遲建立和繫結永久磁碟區、直到排程並建立使用永久磁碟的Pod為止。如此一來、就能建立磁碟區、以符合拓撲需求所強制執行的排程限制。



「等待使用者」繫結模式不需要拓撲標籤。這可獨立於「csi拓撲」功能使用。

您需要的產品

若要使用「csi拓撲」、您需要下列項目：

- 執行的Kubernetes叢集 ["支援的Kubernetes版本"](#)

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeadfd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeadfd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- 叢集中的節點應具有標籤、以引入拓撲感知(topology.kubernetes.io/region`和`topology.kubernetes.io/zone) 。在安裝 Trident 之前，這些標籤 * 應該存在於叢集 * 的節點上，以便 Trident 能夠感知拓撲。

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kuber-
netes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kuber-
netes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kuber-
netes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

步驟1：建立可感知拓撲的後端

Trident 儲存設備後端可根據可用性區域、選擇性地配置磁碟區。每個後端都可以帶有一個可選的 supportedTopologies 區塊、代表支援的區域和區域清單。對於使用此類後端的StorageClass、只有在受支援地區/區域中排程的應用程式要求時、才會建立Volume。

以下是後端定義範例：

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-a
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-b
```

JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {"topology.kubernetes.io/region": "us-east1",
     "topology.kubernetes.io/zone": "us-east1-a"},
    {"topology.kubernetes.io/region": "us-east1",
     "topology.kubernetes.io/zone": "us-east1-b"}
  ]
}
```



`supportedTopologies` 用於提供每個後端的區域和區域清單。這些區域和區域代表 StorageClass 中可提供的允許值清單。對於包含後端所提供區域和區域子集的 StorageClasses、Trident 會在後端建立磁碟區。

您也可以定義每個儲存資源池的「支援拓撲」。請參閱下列範例：

```

---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-a
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-b
storage:
- labels:
    workload: production
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b

```

在此範例中、「REGion」和「Zone」標籤代表儲存資源池的位置。「topology.kubernetes.io/region」和「topology.kubernetes.io/zone」決定儲存資源池的使用來源。

步驟2：定義可感知拓撲的StorageClass

根據提供給叢集中節點的拓撲標籤、可以定義StorageClass以包含拓撲資訊。這將決定做為所提出之永久虛擬磁碟要求候選的儲存資源池、以及可以使用Trident所提供之磁碟區的節點子集。

請參閱下列範例：

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"

```

在上面提供的 StorageClass 定義中 volumeBindingMode，設置為 WaitForFirstConsumer。在Pod中引用此StorageClass所要求的PVCS之前、系統不會對其採取行動。此外、還 allowedTopologies`提供要使用的區域和區域。StorageClass 會 `netapp-san-us-east1`在上述定義的後端建立 PVC `san-backend-us-east1`。

步驟3：建立並使用PVC

建立StorageClass並對應至後端後端後端之後、您現在就可以建立PVCS。

請參閱以下「sPEC」範例：

```

---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 300Mi
  storageClassName: netapp-san-us-east1

```

使用此資訊清單建立永久虛擬環境可能會產生下列結果：

```

kubect1 create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubect1 get pvc
NAME          STATUS      VOLUME      CAPACITY    ACCESS MODES   STORAGECLASS
AGE
pvc-san      Pending
2s
kubect1 describe pvc
Name:          pvc-san
Namespace:     default
StorageClass: netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type          Reason              Age   From
  ----          -
Normal WaitForFirstConsumer 6s    persistentvolume-controller waiting
for first consumer to be created before binding

```

若要Trident建立磁碟區並將其連結至PVC、請在Pod中使用PVC。請參閱下列範例：

```

apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: topology.kubernetes.io/region
                operator: In
                values:
                  - us-east1
            preferredDuringSchedulingIgnoredDuringExecution:
              - weight: 1
                preference:
                  matchExpressions:
                    - key: topology.kubernetes.io/zone
                      operator: In
                      values:
                        - us-east1-a
                        - us-east1-b
      securityContext:
        runAsUser: 1000
        runAsGroup: 3000
        fsGroup: 2000
    volumes:
      - name: voll
        persistentVolumeClaim:
          claimName: pvc-san
    containers:
      - name: sec-ctx-demo
        image: busybox
        command: [ "sh", "-c", "sleep 1h" ]
        volumeMounts:
          - name: voll
            mountPath: /data/demo
        securityContext:
          allowPrivilegeEscalation: false

```

此pod化 規範會指示Kubernetes在「us-east1」區域的節點上排程pod、並從「us-east1-a」或「us-east1-b」區域中的任何節點中進行選擇。

請參閱下列輸出：

```
kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP              NODE
NOMINATED NODE READINESS GATES
app-pod-1    1/1     Running   0          19s   192.168.25.131  node2
<none>      <none>
kubectl get pvc -o wide
NAME          STATUS   VOLUME                                     CAPACITY
ACCESS MODES STORAGECLASS          AGE   VOLUMEMODE
pvc-san      Bound   pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b 300Mi
RWO          netapp-san-us-east1  48s   Filesystem
```

更新後端以納入 supportedTopologies

您可以使用「tridentctl後端更新」來更新現有的後端、以納入「最上層拓撲」清單。這不會影響已配置的磁碟區、而且只會用於後續的PVCS。

如需詳細資訊、請參閱

- ["管理容器的資源"](#)
- ["節點選取器"](#)
- ["關聯性與反關聯性"](#)
- ["污染與容許"](#)

使用快照

Kubernetes 持續磁碟區（PV）的磁碟區快照可啟用磁碟區的時間點複本。您可以建立使用 Trident 建立的磁碟區快照、匯入在 Trident 外部建立的快照、從現有快照建立新的磁碟區、以及從快照復原磁碟區資料。

總覽

支援Volume Snapshot ontap-nas、ontap-nas-flexgroup、ontap-san、ontap-san-economy、solidfire-san、gcp-cvs`和`azure-netapp-files 驅動程式：

開始之前

您必須擁有外部快照控制器和自訂資源定義（CRD）、才能使用快照。這是Kubernetes Orchestrator的責任（例如：Kubeadm、GKE、OpenShift）。

如果您的Kubernetes發佈版本未包含快照控制器和CRD、請參閱 [部署 Volume Snapshot 控制器](#)。



如果在 GKE 環境中建立隨需磁碟區快照、請勿建立快照控制器。GKE使用內建的隱藏式快照控制器。

建立磁碟區快照

步驟

1. 建立 VolumeSnapshotClass。如需詳細資訊、請參閱 "[Volume SnapshotClass](#)"。
 - `driver` 指向 Trident CSI 驅動程式。
 - deletionPolicy 可以 Delete 或 Retain。設定為時 Retain、儲存叢集上的基礎實體快照、即使在 VolumeSnapshot 物件已刪除。

範例

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

2. 建立現有 PVC 的快照。

範例

- 此範例會建立現有 PVC 的快照。

```
cat snap.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

- 此範例會為名稱為 PVC 的 Volume Snapshot 物件建立一個 pvc1 快照名稱設為 pvc1-snap。Volume Snapshot 類似於 PVC、並與相關聯 VolumeSnapshotContent 代表實際快照的物件。

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                AGE
pvc1-snap           50s
```

- 您可以識別 VolumeSnapshotContent 的物件 pvc1-snap 描述 Volume Snapshot。◦ Snapshot

Content Name 識別提供此快照的 Volume SnapshotContent 物件。Ready To Use 參數表示快照可用於建立新的 PVC。

```
kubectl describe volumesnapshots pvcl-snap
Name:          pvcl-snap
Namespace:    default
.
.
.
Spec:
  Snapshot Class Name:  pvcl-snap
  Snapshot Content Name: snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:      PersistentVolumeClaim
    Name:      pvcl
  Status:
    Creation Time:  2019-06-26T15:27:29Z
    Ready To Use:  true
    Restore Size:  3Gi
.
.
```

從磁碟區快照建立 PVC

您可以使用 dataSource 使用名為的 Volume Snapshot 建立 PVC <pvc-name> 做為資料來源。建立好永久虛擬基礎架構之後、就能將它附加到Pod上、就像使用任何其他永久虛擬基礎架構一樣使用。



將在來源 Volume 所在的同一個後端建立 PVC。請參閱 ["KB：無法在替代後端建立 Trident PVC Snapshot 的 PVC"](#)。

以下範例使用建立 PVC pvcl-snap 做為資料來源。

```

cat pvc-from-snap.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

匯入 **Volume** 快照

Trident 支援、"[Kubernetes 預先配置的快照程序](#)"可讓叢集管理員建立 `VolumeSnapshotContent` 物件、並匯入在 Trident 之外建立的快照。

開始之前

Trident 必須已建立或匯入快照的父磁碟區。

步驟

- * 叢集管理：* 建立 `VolumeSnapshotContent` 參照後端快照的物件。這會在 Trident 中啟動快照工作流程。
 - 在中指定後端快照的名稱 annotations 做為 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`。
 - 請在中 `snapshotHandle`` 指定 `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>`。這是通話中外部快照機提供給 Trident 的唯一資訊 `ListSnapshots`。



◦ `<volumeSnapshotContentName>` 由於 CR 命名限制、無法永遠符合後端快照名稱。

範例

下列範例建立 `VolumeSnapshotContent` 參照後端快照的物件 `snap-01`。

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01" # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>

```

2. * 叢集管理：* 建立 VolumeSnapshot 參照的 CR VolumeSnapshotContent 物件：這會要求存取權以使用 VolumeSnapshot 在指定的命名空間中。

範例

下列範例建立 VolumeSnapshot CR 命名 import-snap 這是參考的 VolumeSnapshotContent 已命名 import-snap-content。

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content

```

3. * 內部處理（不需採取任何行動）：* 外部快照機可辨識新建立的 VolumeSnapshotContent、並執行 ListSnapshots 通話。Trident 會建立 TridentSnapshot。
 - 外部快照器會設定 VolumeSnapshotContent 至 readyToUse 和 VolumeSnapshot 至 true。
 - Trident 退貨 readyToUse=true。
4. * 任何使用者：* 建立 PersistentVolumeClaim 以參考新的 VolumeSnapshot、其中 spec.dataSource（或 spec.dataSourceRef）名稱為 VolumeSnapshot 名稱。

範例

下列範例建立一個 PVC 參照 VolumeSnapshot 已命名 import-snap。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

使用快照恢復 **Volume** 資料

快照目錄預設為隱藏、以協助使用進行資源配置的磁碟區達到最大相容性 `ontap-nas` 和 `ontap-nas-economy` 驅動程式：啟用 `.snapshot` 直接從快照恢復資料的目錄。

使用 `Volume Snapshot Restore ONTAP CLI` 將磁碟區還原至先前快照中記錄的狀態。

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```



當您還原快照複本時、會覆寫現有的 `Volume` 組態。建立快照複本之後對 `Volume` 資料所做的變更將會遺失。

刪除含有相關快照的 **PV**

刪除具有相關快照的持續 `Volume` 時、對應的 `Trident Volume` 會更新為「刪除狀態」。移除磁碟區快照以刪除 `Trident` 磁碟區。

部署 **Volume Snapshot** 控制器

如果您的 `Kubernetes` 發佈版本未包含快照控制器和客戶需求日、您可以依照下列方式進行部署。

步驟

1. 建立 `Volume Snapshot` 客戶需求日。

```
cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. 建立Snapshot控制器。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```



如有必要、請開啟 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` 和更新 namespace 到您的命名空間。

相關連結

- ["Volume快照"](#)
- ["Volume SnapshotClass"](#)

管理及監控 Trident

升級 Trident

升級 Trident

從 24.02 版開始、Trident 遵循四個月的發行步調、每個日曆年度提供三個主要版本。每個新版本均以舊版為基礎、並提供新功能、效能增強、錯誤修正及改善功能。我們建議您每年至少升級一次、以充分利用 Trident 的新功能。

升級前的考量

升級至最新版的 Trident 時、請考慮下列事項：

- 在指定的 Kubernetes 叢集中、所有命名空間都應該只安裝一個 Trident 執行個體。
- Trident 23.07 及更新版本需要 v1 Volume 快照、不再支援 Alpha 或 beta 快照。
- 如果您在中建立了 Cloud Volumes Service for Google Cloud "[CVS服務類型](#)"、則從 Trident 23.01 升級時、必須更新後端組態、才能使用 `standardsw` 或 `zoneredundantstandardsw` 服務層級。如果無法在後端更新 `serviceLevel`、可能會導致磁碟區失敗。如 "[CVS 服務類型範例](#)" 需詳細資訊、請參閱。
- 升級時、Trident 必須提供 `parameter.fsType` 使用中的 `StorageClasses` 資訊。您可以在不中斷現有磁碟區的情況下刪除及重新建立 `StorageClasses` 磁碟區。
 - 這是強制實施的一項**要求 "[安全性內容](#)" 適用於 SAN 磁碟區。
 - `sample INPUT` 目錄包含 <https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-basic.yaml.templ> 等範例[`storage-class-basic.yaml.templ`] 和連結：`storage-class-bronze-default.yaml`。
 - 如需詳細資訊、請參閱 "[已知問題](#)"。

步驟 1：選取版本

Trident 版本遵循日期 `YY.MM` 命名慣例、其中「是」是年份的最後兩位數、「MM」是月份。DOT 版本遵循 `YY.MM.X` 慣例、其中「X」是修補程式層級。您將根據要升級的版本、選擇要升級的版本。

- 您可以直接升級至安裝版本的四個版本範圍內的任何目標版本。例如、您可以直接從 23.04（或任何 23.04 點版本）升級至 24.06。
- 如果您要從四個版本的外部版本升級、請執行多步驟升級。使用升級說明從升級至最新版本、以符合四個版本的 "[舊版](#)" 視窗。例如、如果您執行的是 22.01、而且想要升級至 24.06：
 - a. 第一次從 22.07 升級至 23.04。
 - b. 然後從 23.04 升級至 24.06。



在 OpenShift Container Platform 上使用 Trident 運算子進行升級時、您應升級至 Trident 21.01.1 或更新版本。隨 21.01.0 一起發行的 Trident 運算子包含已在 21.01.1 中修正的已知問題。如需詳細資訊、請參閱 "[GitHub 問題詳細資料](#)"。

步驟 2：確定原始安裝方法

若要判斷您最初安裝 Trident 的版本：

1. 使用 `kubectl get pods -n trident` 檢查 Pod。
 - 如果沒有運算子 Pod、則使用安裝 Trident `tridentctl`。
 - 如果有操作員 Pod、則 Trident 是使用 Trident 操作員手動或使用 Helm 來安裝。
2. 如果有操作員 Pod、請使用 ``kubectl describe torc`` 判斷是否使用 Helm 安裝 Trident。
 - 如果有 Helm 標籤、則使用 Helm 安裝 Trident。
 - 如果沒有 Helm 標籤、則會使用 Trident 操作員手動安裝 Trident。

步驟 3：選擇升級方法

一般而言，您應該使用初始安裝所使用的相同方法進行升級"[在安裝方法之間移動](#)"，不過您可以。升級 Trident 有兩個選項。

- "[使用Trident營運者進行升級](#)"



我們建議您檢閱 "[瞭解營運商升級工作流程](#)" 與操作員一起升級之前。

*

與營運者一起升級

瞭解營運商升級工作流程

在使用 Trident 操作員升級 Trident 之前、您應該先瞭解升級期間所發生的背景程序。其中包括 Trident 控制器、控制器 Pod 和節點 Pod 的變更、以及啟用循環更新的節點示範集。

Trident 營運商升級處理

安裝和升級 Trident 的其中一項"[使用 Trident 運算子的優點](#)"、是在不中斷現有掛載磁碟區的情況下、自動處理 Trident 和 Kubernetes 物件。如此一來、Trident 就能支援零停機的升級、或"[滾動更新](#)"。尤其是 Trident 運算子會與 Kubernetes 叢集通訊、以便：

- 刪除並重新建立 Trident Controller 部署和節點示範集。
- 以新版本更換 Trident 控制器 Pod 和 Trident 節點 Pod。
 - 如果節點未更新、則不會阻止其餘節點更新。
 - 只有執行中 Trident Node Pod 的節點才能裝載磁碟區。



有關 Kubernetes 叢集上 Trident 架構的詳細資訊"[Trident 架構](#)"、請參閱。

營運商升級工作流程

當您使用 Trident 運算子啟動升級時：

1. * Trident 運算子 * :
 - a. 偵測目前安裝的 Trident 版本 (版本 n) 。
 - b. 更新所有 Kubernetes 物件、包括 CRD、RBAC 和 Trident SVC 。
 - c. 刪除版本 n 的 Trident 控制器部署。
 - d. 為版本 $n+1$ 建立 Trident Controller 部署。
 2. * Kubernetes* 為 $n+1$ 建立 Trident 控制器 Pod 。
 3. * Trident 運算子 * :
 - a. 刪除 n 的 Trident 節點示範集。操作人員不會等待節點 Pod 終止。
 - b. 為 $n+1$ 建立 Trident 節點 Demont 。
 4. * Kubernetes* 會在未執行 Trident Node Pod 的節點上建立 Trident Node Pod 。
- 如此可確保節點上的任何版本、都不會有超過一個 Trident Node Pod 。

使用 Trident 營運商或 Helm 升級 Trident 安裝

您可以手動或使用 Helm、使用 Trident 營運商來升級 Trident。您可以從 Trident 營運商安裝升級至其他 Trident 營運商安裝、或從安裝升級 `tridentctl` 至 Trident 營運商版本。在升級 Trident 操作員安裝之前、請先檢閱["選擇升級方法"](#)。

升級手動安裝

您可以從叢集範圍的 Trident 運算子安裝升級到另一個叢集範圍的 Trident 運算子安裝。所有 Trident 版本 21.01 及更新版本均使用叢集範圍的運算子。



若要從使用命名空間範圍運算子 (20.07 至 20.10 版) 安裝的 Trident 升級、請使用 Trident 的升級指示["您已安裝的版本"](#)。

關於這項工作

Trident 提供一個套件檔案、可讓您用來安裝運算子、並為 Kubernetes 版本建立相關的物件。

- 對於運行 Kubernetes 1.24 的羣集，請使用 `"bunder_pre_1_25.yaml"`。
- 對於運行 Kubernetes 1.25 或更高版本的羣集，請使用 `"bunder_POST_1_25.yaml"`。

開始之前

確保您使用的是執行中的 Kubernetes 叢集 ["支援的Kubernetes版本"](#)。

步驟

1. 驗證您的 Trident 版本：

```
./tridentctl -n trident version
```

2. 刪除用於安裝目前 Trident 執行個體的 Trident 運算子。例如、如果您是從 23.07 升級、請執行下列命令：

```
kubectl delete -f 23.07.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

3. 如果您使用自訂初始安裝 `TridentOrchestrator` 屬性、您可以編輯 `TridentOrchestrator` 物件以修改安裝參數。這可能包括針對離線模式指定鏡射Trident和csi映像登錄、啟用偵錯記錄或指定映像提取機密所做的變更。
4. 使用適用於您環境的正確套件 YAML 檔案安裝 Trident 、其中 `_Kubernetes <bundle.yaml>` 版本為 `_bundle_pre_1_25.yaml` 或 `_bundle_post_1_25.yaml` 以 Kubernetes 版本為基礎。例如、如果您要安裝 Trident 24.10 、請執行下列命令：

```
kubectl create -f 24.10.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

升級 Helm 安裝

您可以升級 Trident Helm 安裝。



將已安裝 Trident 的 Kubernetes 叢集從 1.24 升級至 1.25 或更新版本時、您必須 `true` 先更新 `values.yaml` 以設定 `excludePodSecurityPolicy` 或新增 `--set excludePodSecurityPolicy=true` 至 `helm upgrade` 命令、才能升級叢集。

如果您已經將 Kubernetes 叢集從 1.24 升級至 1.25 、而不升級 Trident helm 、則 helm 升級將會失敗。若要順利完成升級、請先執行下列步驟：

1. 從安裝 `helm-mapkubeapis` 外掛程式 <https://github.com/helm/helm-mapkubeapis> 。
2. 在安裝 Trident 的命名空間中、為 Trident 版本執行演習。這會列出將會清除的資源。

```
helm mapkubeapis --dry-run trident --namespace trident
```

3. 以 `helm` 執行完整執行以進行清理。

```
helm mapkubeapis trident --namespace trident
```

步驟

1. 如果您 "已使用 Helm 安裝 Trident" 是、您可以在單一步驟中使用 `helm upgrade trident netapp-trident/trident-operator --version 100.2410.0` 進行升級。如果您未新增 Helm repo 或無法使用它來升級：
 - a. 從下載最新的 Trident 版本 "[GitHub的_Assets區段](#)" 。
 - b. 使用 `helm upgrade` 反映您要升級至的版本的命令 `trident-operator-24.10.0.tgz` 。

```
helm upgrade <name> trident-operator-24.10.0.tgz
```



如果您在初始安裝期間設定自訂選項（例如指定 Trident 和 CSI 映像的私有、鏡射登錄）、請附加 `helm upgrade` 命令使用 `--set` 為了確保升級命令中包含這些選項、否則這些值會重設為預設值。

2. 執行 `helm list` 以確認圖表和應用程式版本均已升級。執行 `tridentctl logs` 以檢閱任何偵錯訊息。

從升級 `tridentctl` 安裝至 **Trident** 操作員

您可以從升級至最新版的 Trident 運算子 `tridentctl` 安裝：現有的後端和 PVC 將會自動提供使用。



在安裝方法之間切換之前、請參閱 "[在安裝方法之間移動](#)"。

步驟

1. 下載最新的 Trident 版本。

```
# Download the release required [24.10.0]
mkdir 24.10.0
cd 24.10.0
wget
https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

2. 從資訊清單建立「TridentOrchestrator」CRD。

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. 在同一個命名空間中部署叢集範圍的運算子。

```
kubectl create -f deploy/<bundle-name.yaml>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
```

NAME	READY	STATUS	RESTARTS	AGE
trident-controller-79df798bdc-m79dc	6/6	Running	0	150d
trident-node-linux-xrst8	2/2	Running	0	150d
trident-operator-5574dbbc68-nthjv	1/1	Running	0	1m30s

4. 建立 TridentOrchestrator CR 以安裝 Trident。

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
```

NAME	READY	STATUS	RESTARTS	AGE
trident-csi-79df798bdc-m79dc	6/6	Running	0	1m
trident-csi-xrst8	2/2	Running	0	1m
trident-operator-5574dbbc68-nthjv	1/1	Running	0	5m41s

5. 確認 Trident 已升級至所需版本。

```
kubectl describe torc trident | grep Message -A 3

Message:          Trident installed
Namespace:        trident
Status:           Installed
Version:          v24.10.0
```

使用tridentctl進行升級

您可以使用輕鬆升級現有的 Trident 安裝 tridentctl。

關於這項工作

解除安裝及重新安裝 Trident 即為升級。當您解除安裝 Trident 時、不會刪除 Trident 部署所使用的持續 Volume Claim (PVC) 和持續 Volume (PV)。Trident 離線時、已佈建的 PV 仍可繼續使用、而 Trident 會在恢復上線後、為在此期間建立的任何 PVC 配置磁碟區。

開始之前

檢閱 "[選擇升級方法](#)" 使用升級之前 tridentctl。

步驟

1. 執行中的解除安裝命令 tridentctl、移除 CRD 和相關物件以外的所有與 Trident 相關的資源。

```
./tridentctl uninstall -n <namespace>
```

2. 重新安裝 Trident。請參閱 "[使用 tridentctl 安裝 Trident](#)"。



請勿中斷升級程序。確保安裝程式執行完成。

使用 tridentctl 管理 Trident

<https://github.com/NetApp/trident/releases> ["Trident安裝程式套裝組合"] 包含 `tridentctl` 命令列公用程式、可讓您輕鬆存取 Trident。擁有足夠 Privileges 的 Kubernetes 使用者可以使用它來安裝 Trident 或管理包含 Trident Pod 的命名空間。

命令和全域旗標

您可以執行 tridentctl help 取得的可用命令清單 tridentctl 或附加 --help 標記至任何命令、以取得該特定命令的選項和旗標清單。

```
tridentctl [command] [--optional-flag]
```

Trident tridentctl 公用程式支援下列命令和全域旗標。

create

將資源新增至 Trident 。

delete

從 Trident 移除一或多個資源。

get

從 Trident 取得一或多個資源。

help

任何命令的相關說明。

images

列印 Trident 所需的容器影像表格。

import

將現有資源匯入 Trident 。

install

安裝Trident 。

logs

從 Trident 列印記錄。

send

從 Trident 傳送資源。

解除安裝

解除安裝 Trident 。

update

在 Trident 中修改資源。

update backend state

暫時暫停後端作業。

upgrade

在 Trident 中升級資源。

「分度」

列印 Trident 版本。

全域旗標

-d、**--debug**

除錯輸出。

-h、**--help**

的說明 `tridentctl`。

-k、**--kubeconfig string**

指定 `KUBECONFIG` 從本機或從一個 Kubernetes 叢集到另一個叢集執行命令的路徑。



或者、您也可以匯出 `KUBECONFIG` 可指向特定 Kubernetes 叢集和問題的變數 `tridentctl` 命令到該叢集。

-n、**--namespace string**

Trident 部署的命名空間。

-o、**--output string**

輸出格式。json之一|yaml|name|w|ps (預設)。

-s、**--server string**

Trident REST 介面的位址 / 連接埠。



Trident REST 介面可設定為偵聽、僅適用於 127.0.0.1 (適用於 IPV4) 或 `[:1]` (適用於 IPV6)。

命令選項和旗標

建立

使用 `create` 命令將資源新增至 Trident。

```
tridentctl create [option]
```

選項

`backend`：將後端新增至 Trident。

刪除

使用 `delete` 命令從 Trident 中移除一或多個資源。

```
tridentctl delete [option]
```

選項

`backend`：從 Trident 刪除一個或多個儲存設備後端。
`snapshot`：從 Trident 刪除一個或多個 Volume 快照。
`storageclass`：從 Trident 刪除一個或多個儲存類別。

volume：從 Trident 刪除一個或多個儲存磁碟區。

取得

使用 `get` 命令從 Trident 取得一或多個資源。

```
tridentctl get [option]
```

選項

backend：從 Trident 獲得一個或多個儲存設備後端。
snapshot：從 Trident 獲取一個或多個快照。
storageclass：從 Trident 獲取一個或多個儲存類。
volume：從 Trident 獲取一個或多個卷。

旗標

-h、--help：Volume 的說明。
--parentOfSubordinate string：將查詢限制在從屬來源 Volume。
--subordinateOf string：將查詢限制在 Volume 的下屬。

映像

使用 `images` 旗標來列印 Trident 所需的容器映像表格。

```
tridentctl images [flags]
```

旗標

-h、--help：映像說明。
-v、--k8s-version string：Kubernetes 叢集的語義版本。

匯入 Volume

使用 `import volume` 命令將現有磁碟區匯入 Trident。

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

別名

volume、v

旗標

-f、--filename string：Yaml 或 Json PVC 檔案的路徑。
-h、--help：Volume 的說明。
--no-manage：僅建立 PV/PVC。不要假設磁碟區生命週期管理。

安裝

使用 `install` 旗標來安裝 Trident。

```
tridentctl install [flags]
```

旗標

- `--autosupport-image string`: AutoSupport 遙測的容器影像 (預設為「NetApp / Trident AutoSupport : <current-version>」)。
- `--autosupport-proxy string`: 用於發送 AutoSupport 遙測的代理的地址 / 端口。
- `--enable-node-prep`: 嘗試在節點上安裝所需的軟件包。
- `--generate-custom-yaml`: 生成 YAML 文件而無需安裝任何內容。
- `-h --help`: 安裝說明。
- `--http-request-timeout`: 覆寫 Trident 控制器 REST API 的 HTTP 要求逾時 (預設值為 1m30s)。
- `--image-registry string`: 內部映像登錄的位址 / 連接埠。
- `--k8s-timeout duration`: 所有 Kubernetes 作業的逾時時間 (預設為 30 個月)。
- `--kubelet-dir string`: kubelet 內部狀態的主機位置 (預設為「/var/lib/kubelet」)。
- `--log-format string`: Trident 記錄格式 (text、json) (預設「text」(文字))。
- `--node-prep`: 可讓 Trident 準備 Kubernetes 叢集的節點、以使用指定的資料儲存傳輸協定來管理磁碟區。*** 目前 iscsi 是唯一支援的值。 ***
- `--pv string`: Trident 使用的舊版 PV 名稱、請確定不存在 (預設為「Trident」)。
- `--pvc string`: Trident 使用的傳統 PVC 名稱、請確定不存在 (預設的「Trident」)。
- `--silence-autosupport`: 不要自動將 AutoSupport 套裝軟體傳送至 NetApp (預設為 true)。
- `--silent`: 在安裝過程中禁用大多數輸出。
- `--trident-image string`: 要安裝的 Trident 映像。
- `--use-custom-yaml`: 使用安裝目錄中存在的任何現有 YAML 文件。
- `--use-ipv6`: 使用 IPv6 進行 Trident 通訊。

記錄

使用 `logs` 旗標從 Trident 列印記錄。

```
tridentctl logs [flags]
```

旗標

- `-a, --archive`: 創建包含所有日誌的支持歸檔文件 (除非另有指定)。
- `-h, --help`: 日誌幫助。
- `-l --log string`: 要顯示的 Trident 日誌。其中一個是 Trident | auto | Trident 運算子 | All (預設為「自動」)。
- `--node string`: 要從中收集節點 Pod 日誌的 Kubernetes 節點名稱。
- `-p --previous`: 獲取以前的 Container 實例的日誌 (如果存在)。
- `--sidecars`: 獲取 sidecar 容器的日誌。

傳送

使用 `send` 命令從 Trident 傳送資源。

```
tridentctl send [option]
```

選項

`autosupport`: 將 AutoSupport 一份不適用的歸檔文件傳送給 NetApp。

解除安裝

使用 `uninstall` 旗標來解除安裝 Trident。

```
tridentctl uninstall [flags]
```

旗標

- h, --help：解除安裝說明。
- silent：卸載期間禁用大多數輸出。

更新

使用 `update` 命令修改 Trident 中的資源。

```
tridentctl update [option]
```

選項

backend：在 Trident 中更新後端。

更新後端狀態

使用 `update backend state` 暫停或恢復後端作業的命令。

```
tridentctl update backend state <backend-name> [flag]
```

需要考量的重點

- 如果使用 TridentBackendConfig (tbc) 建立後端、則無法使用檔案更新後端 `backend.json` 。
- 如果已在 tbc 中設定、則 `userState` 無法使用命令加以修改 `tridentctl update backend state <backend-name> --user-state suspended/normal` 。
- 若要在透過 tbc 設定 Via `tridentctl` 之後重新取得設定 `userState` 功能、`userState` 必須從 tbc 移除該欄位。這可以使用命令來完成 `kubectl edit tbc` 。`userState` 欄位移除後、您可以使用 `tridentctl update backend state` 命令來變更 `userState` 後端的。
- 使用 `tridentctl update backend state` 變更 `userState` 。您也可以更新 `userState` 使用 `TridentBackendConfig` 或 `backend.json` 檔案、這會觸發後端的完整重新初始化、而且可能會耗費時間。

旗標

- h、--help：後端狀態說明。
- user-state：設為 `suspended` 暫停後端作業。設定為 `normal` 以恢復後端作業。設定為 `suspended`：

- `AddVolume` 和 `Import Volume` 已暫停。
- `CloneVolume`、`ResizeVolume` `PublishVolume` `UnPublishVolume` `CreateSnapshot`、`GetSnapshot` `RestoreSnapshot`、`DeleteSnapshot` `RemoveVolume` `GetVolumeExternal`、`ReconcileNodeAccess` 保持可用狀態。

您也可以使用後端組態檔案或中的欄位來更新後端狀態 `userState` `TridentBackendConfig` `backend.json` 。如需詳細資訊、請參閱 ["管理後端的選項"](#) 和 ["以KECBECVL執行後端管理"](#) 。

範例：

JSON

請依照下列步驟使用檔案更新 `userState backend.json`：

1. 編輯 `backend.json` 檔案、`userState` 將欄位的值設為「已待定」。
2. 使用命令和更新檔案的路徑來更新後端 `tridentctl backend update backend.json`。
 - 範例 *：`tridentctl backend update -f /<path to backend JSON file>/backend.json`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended",
}
```

YAML

您可以在使用命令套用 `tbc` 之後編輯它 `kubectl edit <tbc-name> -n <namespace>`。下列範例會使用選項更新後端狀態以暫停 `userState: suspended`：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
userState: suspended
credentials:
  name: backend-tbc-ontap-nas-secret
```

版本

使用 `version` 用於列印版本的旗標 `tridentctl` 以及執行中的 `Trident` 服務。

```
tridentctl version [flags]
```

旗標

- `--client`：僅限用戶端版本（不需要伺服器）。
- `-h, --help`：版本說明。

外掛程式支援

Tridentctl 支援類似 kubectl 的外掛程式。如果外掛程式二進位檔案名稱遵循「<plugin>」配置、則 Tridentctl 會偵測外掛程式、且二進位檔案位於列出 PATH 環境變數的資料夾中。所有偵測到的外掛程式都會列在 tridentctl 說明的外掛程式區段中。或者、您也可以環境變數 TRIDENTCTL_plugin_path 中指定外掛程式資料夾來限制搜尋（例如：`TRIDENTCTL_PLUGIN_PATH=~/.tridentctl-plugins/`）。如果使用此變數、則 tridentctl 只會在指定的資料夾中搜尋。

監控 Trident

Trident 提供一組 Prometheus 指標端點、可用於監控 Trident 效能。

總覽

Trident 提供的計量可讓您執行下列動作：

- 保留 Trident 健全狀況和組態的索引標籤。您可以檢查作業的成功程度、以及是否能如預期般與後端進行通訊。
- 檢查後端使用資訊、並瞭解後端上配置的磁碟區數量、以及所耗用的空間量等。
- 維護可用後端配置的磁碟區數量對應。
- 追蹤效能。您可以查看 Trident 與後端通訊和執行作業所需的時間。



根據預設、Trident的指標會顯示在「/度量」端點的目標連接埠「8001」上。安裝Trident時*預設會啟用這些度量。

您需要的產品

- 安裝 Trident 的 Kubernetes 叢集。
- Prometheus執行個體。這可以是 ["容器化Prometheus部署"](#) 或者、您也可以選擇以執行Prometheus ["原生應用程式"](#)。

步驟1：定義Prometheus目標

您應該定義 Prometheus 目標、以收集指標、並取得 Trident 管理的後端、建立的磁碟區等相關資訊。這 ["部落格"](#)將說明如何搭配 Trident 使用 Prometheus 和 Grafana 來擷取計量。部落格會說明如何在 Kubernetes 叢集中以營運者的身份執行 Prometheus、以及如何建立 ServiceMonitor 來取得 Trident 指標。

步驟2：建立Prometheus ServiceMonitor

若要使用Trident指標、您應該建立Prometheus ServiceMonitor、以監控「Trident - csi」服務、並在「metrics」連接埠上聆聽。ServiceMonitor範例如下所示：

```

apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s

```

此 ServiceMonitor 定義會擷取服務傳回的度量 `trident-csi`、並特別尋找 `metrics` 服務的端點。因此、Prometheus 現在已設定為瞭解 Trident 的指標。

除了可直接從 Trident 取得的指標之外、kubelet 還會透過自己的指標端點來公開許多 ``kubernetes_`` 指標。Kubelet 可提供有關所附加磁碟區、Pod 及其處理的其他內部作業的資訊。請參閱 ["請按這裡"](#)。

步驟3：使用 PromQL 查詢 Trident 度量

PromQL 適用於建立傳回時間序列或表格資料的運算式。

以下是一些您可以使用的 PromQL 查詢：

取得 Trident 健全狀況資訊

- 來自 Trident 的 HTTP 2XX 回應百分比

```

(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100

```

- Trident 透過狀態代碼的 REST 回應百分比

```

(sum (trident_rest_ops_seconds_total_count) by (status_code) / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100

```

- 由 Trident 執行之作業的平均持續時間（以毫秒為單位）

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

取得 Trident 使用資訊

- 平均Volume大小*

```
trident_volume_allocated_bytes/trident_volume_count
```

- 每個後端配置的Volume空間總計*

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

取得個別Volume使用量



只有同時收集kubelet度量時、才會啟用此功能。

- 每個Volume的已用空間百分比*

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

瞭解 Trident AutoSupport 遙測

根據預設、Trident 會在每日步調中、將 Prometheus 指標和基本後端資訊傳送至 NetApp 。

- 若要停止 Trident 傳送 Prometheus 度量和基本後端資訊至 NetApp、請在 Trident 安裝期間傳遞 `--silence-autosupport` 旗標。
- Trident 也可以透過將容器記錄傳送至 NetApp 隨選支援 `tridentctl send autosupport`。您需要觸發 Trident 來上傳其記錄檔。在提交日誌之前，您應該接受 NetApp 的 <https://www.netapp.com/company/legal/privacy-policy/>["隱私權政策"]。
- 除非另有說明、否則 Trident 會從過去 24 小時擷取記錄。
- 您可以使用旗標指定記錄保留時間範圍 `--since`。例如 `tridentctl send autosupport --since=1h`。這些資訊會透過安裝在 Trident 旁邊的容器收集和傳送 `trident-autosupport`。您可以在上取得容器映像 "[Trident AutoSupport 的](#)"。
- Trident AutoSupport 無法收集或傳輸個人識別資訊 (PII) 或個人資訊。隨附 "[EULA](#)" 不適用於 Trident 容器映像本身的。您可以深入瞭解 NetApp 對資料安全與信任的承諾 "[請按這裡](#)"。

Trident 傳送的有效負載範例如下：

```
---
items:
- backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
  protocol: file
  config:
    version: 1
    storageDriverName: ontap-nas
    debug: false
    debugTraceFlags:
    disableDelete: false
    serialNumbers:
    - nwkvzfanek_SN
    limitVolumeSize: ''
  state: online
  online: true
```

- 此資訊將傳送至NetApp的「不只是」端點。AutoSupport AutoSupport如果您使用私有登錄來儲存容器映像、可以使用「-image-registry」旗標。
- 您也可以產生安裝Yaml檔案來設定Proxy URL。您可以使用「tridentctl install -generate-custom-yaml」來建立Yaml檔案、並在「trident部署.yaml」中新增「trident -autodupport」容器的「-proxy-URL」引數。

停用 Trident 計量

要使指標不被報告，您應該生成自定義YAML（使用"-generame-custom-yaml"標誌）並進行編輯，以刪除對"trident-main"容器所調用的"-mication"標誌。

解除安裝Trident

您應該使用與安裝 Trident 相同的方法來解除安裝 Trident 。

關於這項工作

- 如果您需要修正在升級、相依性問題或升級失敗或不完整之後所觀察到的錯誤，您應該解除安裝 Trident ，並使用該的特定指示重新安裝舊版"版本"。這是將 _ 降級 _ 降級至較早版本的唯一建議方法。
- 為了方便升級和重新安裝、解除安裝 Trident 並不會移除 Trident 所建立的 CRD 或相關物件。如果您需要完全移除 Trident 及其所有資料、請參閱["完全移除 Trident 和客戶需求日"](#)。

開始之前

如果您要停用 Kubernetes 叢集、則必須先刪除所有使用 Trident 建立之 Volume 的應用程式、然後再解除安裝。如此可確保在刪除之前、不會在 Kubernetes 節點上發佈 PVC 。

確定原始安裝方法

您應該使用與安裝相同的方法來解除安裝 Trident 。在解除安裝之前、請先確認您原本安裝 Trident 的版本。

1. 使用 `kubect1 get pods -n trident` 檢查 Pod 。

- 如果沒有運算子 Pod 、則使用安裝 Trident `tridentctl` 。
 - 如果有操作員 Pod 、則 Trident 是使用 Trident 操作員手動或使用 Helm 來安裝。
2. 如果有操作員 Pod 、請使用 `kubectl describe tproc trident` 判斷是否使用 Helm 安裝 Trident 。
 - 如果有 Helm 標籤、則使用 Helm 安裝 Trident 。
 - 如果沒有 Helm 標籤、則會使用 Trident 操作員手動安裝 Trident 。

解除安裝 Trident 運算子安裝

您可以手動或使用 Helm 解除安裝 Trident 運算子安裝。

解除安裝手動安裝

如果您使用運算子安裝 Trident 、則可以執行下列其中一項動作來解除安裝：

1. 編輯 TridentOrchestrator CR 並設定解除安裝旗標 **：

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"uninstall":true}}'
```

當 `uninstall` 旗標設定為 `true`、Trident 運算子會卸載 Trident、但不會移除 TridentOrchestrator 本身。如果您想要再次安裝 Trident、請清理 TridentOrchestrator 並建立新的 Trident。

2. 刪除 **TridentOrchestrator**：移除用於部署 Trident 的 CR 後 TridentOrchestrator、您會指示操作員解除安裝 Trident。操作員會處理移除並繼續移除 Trident 部署和取消程式集、刪除其在安裝過程 `TridentOrchestrator` 中所建立的 Trident Pod。

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

解除安裝 Helm 安裝

如果您使用 Helm 安裝 Trident、可以使用解除安裝 `helm uninstall`。

```
#List the Helm release corresponding to the Trident install.
helm ls -n trident
NAME                NAMESPACE          REVISION          UPDATED
STATUS              CHART               APP VERSION
trident             trident             1                 2021-04-20
00:26:42.417764794 +0000 UTC deployed      trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

解除安裝 tridentctl 安裝

使用 `uninstall` 中的命令 `tridentctl` 移除與 Trident 相關的所有資源、但 CRD 和相關物件除外：

```
./tridentctl uninstall -n <namespace>
```

Trident for Docker

部署的先決條件

您必須先在主機上安裝及設定必要的通訊協定先決條件、才能部署 Trident。

驗證需求

- 確認您的部署符合所有的 "需求"。
- 確認您安裝的Docker版本受支援。如果您的Docker版本過時、"安裝或更新"。

```
docker --version
```

- 確認主機上已安裝並設定通訊協定先決條件。

NFS工具

使用作業系統的命令來安裝NFS工具。

RHEL 8以上

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```



安裝NFS工具之後、請重新啟動工作節點、以避免將磁碟區附加至容器時發生故障。

iSCSI工具

使用適用於您作業系統的命令來安裝iSCSI工具。

RHEL 8以上

1. 安裝下列系統套件：

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-  
mapper-multipath
```

2. 檢查iscsite-initier-utils版本是否為6.6.0.874-2.el7或更新版本：

```
rpm -q iscsi-initiator-utils
```

3. 將掃描設為手動：

```
sudo sed -i 's/^\(node.session.scan\) .*/\1 = manual/'  
/etc/iscsi/iscsid.conf
```

4. 啟用多重路徑：

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```



確保在"default" (錯誤) 下"etc/multipath.conf"包含"fapfe_multipaths no"。

5. 確保運行的是"iscsid"和"multipathd"：

```
sudo systemctl enable --now iscsid multipathd
```

6. 啟用並啟動「iSCSI」：

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. 安裝下列系統套件：

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools  
scsistools
```

2. 檢查開放式iSCSI版本是否為2.0.874-5ubuntu2.10或更新版本 (適用於雙聲網路) 或2.0.874-7.1ubuntu6.1或更新版本 (適用於焦點)：

```
dpkg -l open-iscsi
```

3. 將掃描設為手動：

```
sudo sed -i 's/^\(node.session.scan\).*\/\1 = manual/'  
/etc/iscsi/iscsid.conf
```

4. 啟用多重路徑：

```
sudo tee /etc/multipath.conf <<-EOF  
defaults {  
    user_friendly_names yes  
    find_multipaths no  
}  
EOF  
sudo systemctl enable --now multipath-tools.service  
sudo service multipath-tools restart
```



確保在"default"（錯誤）下"etc/multipath.conf"包含"find_multipaths no"。

5. 確保已啟用並執行「open-iscsi」和「多路徑工具」：

```
sudo systemctl status multipath-tools  
sudo systemctl enable --now open-iscsi.service  
sudo systemctl status open-iscsi
```

NVMe 工具

使用適用於您作業系統的命令來安裝 NVMe 工具。



- NVMe 需要 RHEL 9 或更新版本。
- 如果 Kubernetes 節點的核心版本太舊、或 NVMe 套件無法用於您的核心版本、您可能必須使用 NVMe 套件將節點的核心版本更新為一個。

RHEL 9.

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

部署 Trident

Trident for Docker 可直接與適用於 NetApp 儲存平台的 Docker 生態系統整合。它支援從儲存平台到 Docker 主機的儲存資源資源配置與管理、並提供架構、可在未來新增更多平台。

Trident 的多個執行個體可以同時在同一部主機上執行。這可同時連線至多個儲存系統和儲存類型、並可自訂 Docker 磁碟區所使用的儲存設備。

您需要的產品

請參閱["部署的先決條件"](#)。在您確定符合先決條件之後、即可開始部署 Trident。

Docker 託管外掛程式方法（1.1/17.03 版及更新版本）



開始之前

如果您在傳統的精靈方法中使用 Trident pred Docker 1.3/17.03、請務必先停止 Trident 程序、然後重新啟動 Docker 精靈、再使用託管外掛程式方法。

1. 停止所有執行中的執行個體：

```
killall /usr/local/bin/netappdvp
killall /usr/local/bin/trident
```

2. 重新啟動 Docker。

```
systemctl restart docker
```

3. 請確定您已安裝 Docker Engine 17.03（全新 1.13）或更新版本。

```
docker --version
```

如果您的版本過時、["安裝或更新安裝"](#)。

步驟

1. 建立組態檔並指定下列選項：

- 「config」：預設檔案名稱為「config.json」、但您可以使用任何名稱、只要在檔案名稱中指定「config」選項即可。組態檔必須位於主機系統的「/etc/netappdvp」目錄中。
- 記錄層級：指定記錄層級（「debug」、「info」、「warn」、「誤差」、「fatal」）。預設值為「資訊」。
- 「Debug」：指定是否啟用偵錯記錄。預設值為假。如果為true、則會置換記錄層級。

i. 建立組態檔的位置：

```
sudo mkdir -p /etc/netappdvp
```

ii. 建立組態檔：

```
cat << EOF > /etc/netappdvp/config.json
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. 使用託管外掛程式系統啟動 Trident。請以您使用的外掛程式版本（xxx.xxx.x）取代 <version>。

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:<version> config=myConfigFile.json
```

3. 開始使用 Trident 從設定的系統中消耗儲存設備。

a. 建立名為「firstVolume」的Volume：

```
docker volume create -d netapp --name firstVolume
```

- b. 在容器啟動時建立預設Volume：

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

- c. 移除Volume「firstVolume」：

```
docker volume rm firstVolume
```

傳統方法 (1.12版或更早版本)

開始之前

1. 請確定您擁有Docker 1.10版或更新版本。

```
docker --version
```

如果您的版本已過時、請更新安裝。

```
curl -fsSL https://get.docker.com/ | sh
```

或者、"[請依照您的經銷指示進行](#)"。

2. 確保已為您的系統設定NFS和/或iSCSI。

步驟

1. 安裝及設定NetApp Docker Volume外掛程式：

- a. 下載並解壓縮應用程式：

```
wget
https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar xzf trident-installer-24.10.0.tar.gz
```

- b. 移至Bin路徑中的位置：

```
sudo mv trident-installer/extras/bin/trident /usr/local/bin/
sudo chown root:root /usr/local/bin/trident
sudo chmod 755 /usr/local/bin/trident
```

c. 建立組態檔的位置：

```
sudo mkdir -p /etc/netappdvp
```

d. 建立組態檔：

```
cat << EOF > /etc/netappdvp/ontap-nas.json
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. 放置二進位檔案並建立組態檔案之後、請使用所需的組態檔案啟動 Trident 精靈。

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```



除非指定、否則 Volume 驅動程式的預設名稱為「NetApp」。

精靈啟動後、您可以使用 Docker CLI 介面來建立及管理磁碟區

3. 建立 Volume：

```
docker volume create -d netapp --name trident_1
```

4. 在啟動容器時配置 Docker Volume：

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. 移除 Docker Volume：

```
docker volume rm trident_1
docker volume rm trident_2
```

在系統啟動時啟動 Trident

如需系統型系統的單元檔案範例、請參閱 `contrib/trident.service.example` 在Git repo中。若要搭配RHEL使用檔案、請執行下列步驟：

1. 將檔案複製到正確的位置。

如果執行多個執行個體、則應使用單元檔案的唯一名稱。

```
cp contrib/trident.service.example
/usr/lib/systemd/system/trident.service
```

2. 編輯檔案、變更說明（第2行）以符合驅動程式名稱和組態檔案路徑（第9行）、以反映您的環境。
3. 重新載入系統d以擷取變更：

```
systemctl daemon-reload
```

4. 啟用服務。

此名稱會根據您在 `/r/lib/systemd/system` 目錄中命名的檔案而有所不同。

```
systemctl enable trident
```

5. 啟動服務。

```
systemctl start trident
```

6. 檢視狀態。

```
systemctl status trident
```



每當您修改單元檔案時、請執行「`systemctl daem-reload`」命令、以瞭解變更內容。

升級或解除安裝Trident

您可以安全地升級 Trident for Docker、而不會對使用中的磁碟區造成任何影響。在升級過程中、將會有一段短暫的期間 `docker volume`、指向外掛程式的命令將無法成功執行、而應用程式將無法掛載磁碟區、直到外掛程式再次執行為止。在大多數情況下、這是幾秒鐘的事。

升級

請執行下列步驟以升級 Trident for Docker 。

步驟

1. 列出現有的磁碟區：

```
docker volume ls
DRIVER          VOLUME NAME
netapp:latest  my_volume
```

2. 停用外掛程式：

```
docker plugin disable -f netapp:latest
docker plugin ls
ID              NAME          DESCRIPTION
ENABLED
7067f39a5df5   netapp:latest nDVP - NetApp Docker Volume
Plugin  false
```

3. 升級外掛程式：

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```



18.01 版的 Trident 取代了 nDVP 。您應該直接從映像升級 `netapp/ndvp-plugin` 到 `netapp/trident-plugin` 映像。

4. 啟用外掛程式：

```
docker plugin enable netapp:latest
```

5. 確認外掛程式已啟用：

```
docker plugin ls
ID              NAME          DESCRIPTION
ENABLED
7067f39a5df5   netapp:latest Trident - NetApp Docker Volume
Plugin  true
```

6. 確認磁碟區可見：

```
docker volume ls
DRIVER          VOLUME NAME
netapp:latest   my_volume
```



如果您要從舊版 Trident（20.10 之前）升級至 Trident 20.10 或更新版本、可能會發生錯誤。如需詳細資訊、請 "[已知問題](#)" 參閱。如果發生錯誤、您應該先停用外掛程式、然後移除外掛程式、再透過傳遞額外的組態參數來安裝必要的 Trident 版本：`docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

解除安裝

請執行下列步驟、解除安裝 Trident for Docker。

步驟

1. 移除外掛程式所建立的任何磁碟區。
2. 停用外掛程式：

```
docker plugin disable netapp:latest
docker plugin ls
ID                NAME          DESCRIPTION
ENABLED
7067f39a5df5     netapp:latest nDVP - NetApp Docker Volume
Plugin   false
```

3. 移除外掛程式：

```
docker plugin rm netapp:latest
```

使用Volume

您可以使用標準命令、在需要時指定 Trident 驅動程式名稱、輕鬆建立、複製及移除磁碟區 `docker volume`。

建立Volume

- 使用預設名稱建立具有驅動程式的磁碟區：

```
docker volume create -d netapp --name firstVolume
```

- 使用特定的 Trident 執行個體建立 Volume ：

```
docker volume create -d ntap_bronze --name bronzeVolume
```



如果您未指定任何 "選項"，將使用驅動程式的預設值。

- 覆寫預設的Volume大小。請參閱下列範例、以使用驅動程式建立20GiB磁碟區：

```
docker volume create -d netapp --name my_vol --opt size=20G
```



Volume大小以包含整數值的字串表示、並提供選用單位（例如：10g、20GB、3TiB）。如果未指定單位、則預設值為G大小單位可以表示為2（B、KiB、MiB、GiB、TiB）或10（B、KB、MB、GB、TB）的冪。簡寫單元使用2（G = GiB、T = TiB、...）的權力。

移除Volume

- 移除Volume就像移除任何其他Docker Volume一樣：

```
docker volume rm firstVolume
```



使用「Poolidfire - san」驅動程式時、上述範例會刪除及清除磁碟區。

請執行下列步驟以升級 Trident for Docker 。

複製磁碟區

使用 `ontap-nas`、`ontap-san`、`solidfire-san` 和 `gcp-cvs storage drivers` 時，Trident 可以複製卷。使用 `ontap-nas-flexgroup` 或 `ontap-nas-economy` 驅動程式時、不支援複製。從現有磁碟區建立新磁碟區、將會產生新的快照。

- 檢查磁碟區以列舉快照：

```
docker volume inspect <volume_name>
```

- 從現有的Volume建立新的Volume。這將會產生新的快照：

```
docker volume create -d <driver_name> --name <new_name> -o  
from=<source_docker_volume>
```

- 從磁碟區上現有的快照建立新磁碟區。這不會建立新的快照：

```
docker volume create -d <driver_name> --name <new_name> -o
from=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

範例

```
docker volume inspect firstVolume

[
  {
    "Driver": "ontap-nas",
    "Labels": null,
    "Mountpoint": "/var/lib/docker-volumes/ontap-
nas/netappdvp_firstVolume",
    "Name": "firstVolume",
    "Options": {},
    "Scope": "global",
    "Status": {
      "Snapshots": [
        {
          "Created": "2017-02-10T19:05:00Z",
          "Name": "hourly.2017-02-10_1505"
        }
      ]
    }
  }
]

docker volume create -d ontap-nas --name clonedVolume -o from=firstVolume
clonedVolume

docker volume rm clonedVolume
docker volume create -d ontap-nas --name volFromSnap -o from=firstVolume
-o fromSnapshot=hourly.2017-02-10_1505
volFromSnap

docker volume rm volFromSnap
```

存取外部建立的磁碟區

如果容器沒有分割區、且 Trident 支援其檔案系統、您可以使用 Trident *only* 存取外部建立的區塊裝置（或其複本）（例如：無法透過 Trident 存取格式化的 /dev/sdc1 檔案系統 `ext4`）。

驅動程式專屬的Volume選項

每個儲存驅動程式都有一組不同的選項、您可以在建立磁碟區時指定、以自訂結果。請參閱下方、以瞭解適用於您所設定儲存系統的選項。

在磁碟區建立作業期間使用這些選項非常簡單。在CLI操作期間、使用「-o」運算子來提供選項和值。這會覆寫Json組態檔中的任何等效值。

選購的選購配備ONTAP

NFS和iSCSI的Volume建立選項包括：

選項	說明
《大小》	Volume的大小、預設為1 GiB。
《保護區》	精簡或完整配置磁碟區、預設為精簡。有效值為「NONE」（精簡配置）和「Volume」（完整配置）。
「快照原則」	這會將快照原則設定為所需的值。預設值為「無」、表示不會自動建立磁碟區的快照。除非您的儲存管理員加以修改、否則ONTAP 所有的各種系統都會有一個名為「預設」的原則、用來建立並保留六個每小時、每天兩個和每週兩個快照。快照中保留的資料可透過瀏覽至磁碟區任何目錄中的「.snapshot」目錄來還原。
「快照保留區」	這會將快照保留設定為所需的百分比。預設值為無值、表示ONTAP 如果您已選取snapshotPolicy、將選取snapshotReserve（通常為5%）、如果snapshotPolicy為無、則選取0%。您可以在組態檔中為所有ONTAP 的支援項目設定預設的snapshotReserve值、也可以將其用作所有ONTAP 不支援ONTAP-NAS-經濟功能的支援項目、作為所有支援項目的磁碟區建立選項。
「PlitOnClone」	當複製Volume時、ONTAP 這會導致停止實體複本、立即將其從父複本分割開來。預設值為 false。部分複製磁碟區的使用案例、最好是在建立時立即將複本從父複本分割出來、因為不太可能有提高儲存效率的機會。例如、複製空資料庫可節省大量時間、但儲存成本卻很少、因此最好立即分割複本。

選項	說明
加密	<p>在新磁碟區上啟用NetApp Volume Encryption (NVE)；預設為「假」。必須在叢集上授權並啟用NVE、才能使用此選項。</p> <p>如果在後端啟用 NAE、則 Trident 中配置的任何 Volume 都將啟用 NAE。</p> <p>如需更多資訊、請參閱"Trident 如何與 NVE 和 NAE 搭配運作"：</p>
「分層政策」	設定要用於磁碟區的分層原則。這會決定資料在非作用中（冷）時是否移至雲端層。

下列其他選項適用於NFS * Only *：

選項	說明
「unixPermissions」	這會控制Volume本身的權限設定。依預設、權限會設為「-rwxr-x-x」、或是以數字表示法0755、而「root」則是擁有者。文字或數字格式皆可運作。
「snapshotDir	設定為 true 將會製作 .snapshot 存取磁碟區的用戶端可看到的目錄。預設值為 false、表示的可見度 .snapshot 目錄預設為停用。有些影像、例如正式的MySQL 影像、無法如預期般運作 .snapshot 目錄可見。
「匯出政策」	設定要用於磁碟區的匯出原則。預設值為「預設」。
《生態樣式》	設定用於存取磁碟區的安全樣式。預設值為「UNIX」。有效值為「UNIX」和「mixed」。

下列其他選項僅適用於iSCSI *：

選項	說明
「fileSystemType」	設定用於設定iSCSI磁碟區格式的檔案系統。預設值為「ext4」。有效值包括「ext3」、「ext4」和「xfs」。
"paceAllocate (配置) "	設定為 false 將關閉 LUN 的空間分配功能。預設值為 true、表示ONTAP 當磁碟區空間不足、且磁碟區中的LUN無法接受寫入作業時、此功能會通知主機。此選項也可讓ONTAP 支援功能在主機刪除資料時自動回收空間。

範例

請參閱下列範例：

- 建立10GiB Volume：

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- 使用快照建立100GiB磁碟區：

```
docker volume create -d netapp --name demo -o size=100G -o
snapshotPolicy=default -o snapshotReserve=10
```

- 建立已啟用setuid位元的磁碟區：

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

最小Volume大小為20MiB。

如果未指定快照保留、且快照原則為 none、則 Trident 會使用 0% 的快照保留。

- 建立沒有快照原則且無快照保留的磁碟區：

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- 建立不含快照原則的磁碟區、以及自訂快照保留10%的磁碟區：

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- 建立具有快照原則和10%自訂快照保留的磁碟區：

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- 使用快照原則建立磁碟區、並接受ONTAP的預設快照保留（通常為5%）：

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Element軟體Volume選項

元素軟體選項會顯示與磁碟區相關的服務品質 (QoS) 原則大小和品質。建立磁碟區時、會使用「-o type =service_level」命名法來指定與其相關的QoS原則。

使用元素驅動程式定義QoS服務層級的第一步、是建立至少一種類型、並在組態檔中指定與名稱相關的最小、最大和尖峰IOPS。

其他元素軟體磁碟區建立選項包括：

選項	說明
《大小》	磁碟區大小、預設為1GiB或組態項目... 「預設值」： {"Size": "5G"}。
「區塊大小」	使用512或4096、預設為512或組態項目預 設BlockSizes。

範例

請參閱下列QoS定義範例組態檔：

```

{
  "...": "...",
  "Types": [
    {
      "Type": "Bronze",
      "Qos": {
        "minIOPS": 1000,
        "maxIOPS": 2000,
        "burstIOPS": 4000
      }
    },
    {
      "Type": "Silver",
      "Qos": {
        "minIOPS": 4000,
        "maxIOPS": 6000,
        "burstIOPS": 8000
      }
    },
    {
      "Type": "Gold",
      "Qos": {
        "minIOPS": 6000,
        "maxIOPS": 8000,
        "burstIOPS": 10000
      }
    }
  ]
}

```

在上述組態中、我們有三種原則定義：銅級、銀級和金級。這些名稱為任意名稱。

- 建立10GiB Gold Volume：

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- 建立100GiB銅級磁碟區：

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

收集記錄

您可以收集記錄以協助疑難排解。收集記錄的方法會因執行Docker外掛程式的方式而有所不同。

收集記錄以進行疑難排解

步驟

1. 如果您使用建議的託管外掛程式方法（亦即使用命令）來執行 Trident docker plugin、請依下列方式檢視：

```
docker plugin ls
ID                NAME                DESCRIPTION
ENABLED
4fb97d2b956b     netapp:latest      nDVP - NetApp Docker Volume
Plugin           false
journalctl -u docker | grep 4fb97d2b956b
```

標準記錄層級應可讓您診斷大多數問題。如果您覺得還不夠、可以啟用偵錯記錄。

2. 若要啟用除錯記錄、請安裝已啟用除錯記錄的外掛程式：

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

或者、當外掛程式已安裝時、請啟用偵錯記錄功能：

```
docker plugin disable <plugin>
docker plugin set <plugin> debug=true
docker plugin enable <plugin>
```

3. 如果您在主機上執行二進位檔本身、則主機的記錄檔中會有可用的記錄檔 /var/log/netappdvp 目錄。若要啟用偵錯記錄、請指定 -debug 當您執行外掛程式時。

一般疑難排解秘訣

- 新使用者最常遇到的問題是組態錯誤、導致外掛程式無法初始化。發生這種情況時、當您嘗試安裝或啟用外掛程式時、可能會看到如下訊息：

「精靈的錯誤回應：請撥打UNIX /run / dock/plugins/<id>/NetApp.sock : Connect : 沒有這類檔案或目錄」

這表示外掛程式無法啟動。幸運的是、外掛程式是以全方位的記錄功能打造而成、可協助您診斷大部分可能遇到的問題。

- 如果將PV掛載到容器時發生問題、請確定已安裝並執行「rpcbind」。使用主機作業系統所需的套件管理程式、檢查「rpcbind」是否正在執行。您可以執行「systemctl狀態rpcbind」或其等效項目來檢查rpcbind服務的狀態。

管理多個 Trident 執行個體

當您想要同時使用多個儲存組態時、需要多個Trident執行個體。多個執行個體的關鍵是在主機上具現化Trident時、使用容器化外掛程式的「-alias」選項或「-volume驅動程式」選項、為它們指定不同的名稱。

Docker託管外掛程式（1.3/17.03版或更新版本）的步驟

1. 啟動第一個指定別名和組態檔的執行個體。

```
docker plugin install --grant-all-permissions --alias silver
netapp/trident-plugin:21.07 config=silver.json
```

2. 啟動第二個執行個體、指定不同的別名和組態檔。

```
docker plugin install --grant-all-permissions --alias gold
netapp/trident-plugin:21.07 config=gold.json
```

3. 建立磁碟區、將別名指定為驅動程式名稱。

例如、黃金Volume：

```
docker volume create -d gold --name ntapGold
```

例如、對於銀級Volume：

```
docker volume create -d silver --name ntapSilver
```

傳統的步驟（1.12版或更早版本）

1. 使用自訂驅動程式ID以NFS組態啟動外掛程式：

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. 使用自訂驅動程式ID以iSCSI組態啟動外掛程式：

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. 為每個驅動程式執行個體配置Docker磁碟區：

例如、對於NFS：

```
docker volume create -d netapp-nas --name my_nfs_vol
```

例如、對於iSCSI：

```
docker volume create -d netapp-san --name my_iscsi_vol
```

儲存組態選項

請參閱 Trident 組態可用的組態選項。

全域組態選項

無論使用的儲存平台為何、這些組態選項都適用於所有 Trident 組態。

選項	說明	範例
「分度」	組態檔版本編號	1
「storageDriverName」	儲存驅動程式名稱	ontap-nas、ontap-san、 ontap-nas-economy、 ontap-nas-flexgroup、 solidfire-san
「storagePrefix」	Volume名稱的選用首碼。預設： netappdvp_。	staging_
《限制Volume大小》	Volume大小的選擇性限制。預設： ""（未強制執行）	10g



請勿將「儲存前置詞」（包括預設值）用於元素後端。缺省情況下，"Solidfire-san" 驅動程序將忽略此設置而不使用前綴。我們建議使用特定的TenantId進行Docker Volume對應、或是使用Docker版本、驅動程式資訊和原始名稱填入的屬性資料、以便在可能使用任何名稱標示的情況下使用。

您可以使用預設選項、避免在每個建立的Volume上指定這些選項。「最小化」選項適用於所有控制器類型。如ONTAP 需如何設定預設Volume大小的範例、請參閱「功能區組態」一節。

選項	說明	範例
《大小》	新磁碟區的選用預設大小。預設： 1G	10G

組態ONTAP

除了上述全域組態值之外、使用ONTAP 時還提供下列頂層選項。

選項	說明	範例
《馬納格門達利》	IP位址ONTAP：您可以指定完整網域名稱（FQDN）。	10.0.0.1
「DataLIF」	<p>傳輸協定LIF的IP位址。</p> <ul style="list-style-type: none"> • ONTAP NAS 驅動程式*：建議您指定 dataLIF。如果未提供、Trident 會從 SVM 擷取資料生命。您可以指定要用於NFS掛載作業的完整網域名稱（FQDN）、讓您建立循環配置資源DNS、以便在多個資料生命期之間達到負載平衡。 <p>《SAN驅動程式：請勿指定用於iSCSI》ONTAP。Trident 使用"可選擇的LUN對應ONTAP"來探索建立多重路徑工作階段所需的 iSCI 生命。如果明確定義、就會產生警告 dataLIF。</p>	10.0.0.2
《虛擬機器》	要使用的儲存虛擬機器（如果管理LIF是叢集LIF、則為必要）	svm_nfs
《使用者名稱》	連線至儲存設備的使用者名稱	vsadmin
密碼	連線至儲存設備的密碼	secret
《Aggregate》	用於資源配置的Aggregate（選用；如果已設定、則必須指派給SVM）。對於 `ontap-nas-flexgroup` 驅動程式、此選項會被忽略。指派給 SVM 的所有集合體都會用於佈建 FlexGroup Volume。	aggr1
「限制Aggregateusage」	如果使用率高於此百分比、則可選用、失敗的資源配置	75%

選項	說明	範例
「nfsMountOptions」	精細控制NFS掛載選項；預設為「-o nfsver=3」。僅適用於「 ONTAP-NAS 」和「 ONTAP-NAS-經濟 」驅動程式。 "請參閱此處的NFS主機組態資訊" 。	-o nfsvers=4
「igroupName」	Trident 會以 netappdvp 建立及管理每個節點 `igroups`。 此值不可變更或省略。 *僅適用於 ontap-san 驅動程式*。	netappdvp
《限制Volume大小》	可要求的最大磁碟區大小。	300g
"qtreesPerFlexvol"	每FlexVol 個邊區最多qtree數、範圍必須為[50、300]、預設值為200。 適用於 ontap-nas-economy 驅動程式、此選項可自訂每FlexVol 個版本的qtree數量上限。	300
sanType	* 僅支援 ontap-san 驅動程式。用於選擇 `iscsi` iSCSI、nvme NVMe / TCP 或 fcp SCSI over Fibre Channel (FC)。FCP (SCSI over FC) 是 Trident 24.10 版本的技術預覽功能。*	iscsi 如果空白
limitVolumePoolSize	* ontap-san-economy ontap-san-economy 僅支援和驅動程式。 *在 ONTAP ONTAP NAS 經濟型和 ONTAP SAN 經濟型驅動程式中限制 FlexVol 大小。	300g

您可以使用預設選項、避免在您建立的每個Volume上指定這些選項：

選項	說明	範例
《保護區》	空間保留模式；none (精簡配置) 或 volume (粗)	無
「快照原則」	要使用的 Snapshot 原則、預設為 none	無

選項	說明	範例
「快照保留區」	Snapshot 保留百分比、預設為「」接受 ONTAP 預設值	10
「PlitOnClone」	建立複本時、將其父複本分割成預設值 false	「假」
加密	<p>在新磁碟區上啟用NetApp Volume Encryption (NVE)；預設為「假」。必須在叢集上授權並啟用NVE、才能使用此選項。</p> <p>如果在後端啟用 NAE、則 Trident 中配置的任何 Volume 都將啟用 NAE。</p> <p>如需更多資訊、請參閱"Trident 如何與 NVE 和 NAE 搭配運作"：。</p>	是的
「unixPermissions」	NAS 選項適用於已佈建的 NFS 磁碟區、預設為 777	777
「snapshotDir」	用於存取目錄的 NAS 選項 .snapshot。	針對 NFSv3 的 NFSv4 "false" 為 "true"
「匯出政策」	NFS 匯出原則使用的 NAS 選項、預設為 default	default
《生態樣式》	<p>NAS選項、可存取已配置的NFS Volume。</p> <p>NFS支援 mixed 和 unix 安全樣式：預設值為 unix。</p>	unix
「fileSystemType」	SAN 選項可選擇檔案系統類型、預設為 ext4	xfs
「分層政策」	要使用的分層原則、預設為 none；snapshot-only 適用於 ONTAP 9.5 之前的 SVM-DR 組態	無

擴充選項

「ONTAP-NAS」和「ONTAP-SAN」驅動程式可為ONTAP FlexVol 每個Docker Volume建立一個支援功能。支援每個叢集節點最多1000個FlexVols、叢集最多12、000個FlexVols。ONTAP如果您的Docker Volume需求符合上述限制、則「ONTAP-NAS」驅動程式是首選的NAS解決方案、因為FlexVols提供的其他功能、例如Docker Volume精細快照和複製。

如果您需要的Docker磁碟區數量超過FlexVol了《支援》的範圍、請選擇「ONTAP - NAS經濟」或「ONTAP - SAN經濟」驅動程式。

「ONTAP-NAS經濟」驅動程式會在ONTAP 自動管理的FlexVols資源池中、將Docker磁碟區建立為還原樹狀結構。qtree的擴充能力大幅提升、每個叢集節點最多可達100、000個、每個叢集最多可達2、400、000個、而犧牲了部分功能。「ONTAP-NAS-節約」驅動程式不支援Docker Volume精細快照或複製。



Docker swarm目前不支援「ONTAP-NAS-節約」驅動程式、因為swarm不會協調多個節點之間的磁碟區建立。

「ONTAP-san經濟」驅動程式會在ONTAP 自動管理的FlexVols共用集區內、將Docker Volume建立為還原LUN。如此FlexVol 一來、每個支援不只侷限於一個LUN、而且能為SAN工作負載提供更好的擴充性。根據儲存陣列的不同、ONTAP 每個叢集最多可支援16384個LUN。由於磁碟區是下方的LUN、因此此驅動程式支援Docker磁碟區精細快照和複製。

選擇 `ontap-nas-flexgroup` 驅動程式來增加單一磁碟區的平行度、使其可擴充至數十億個檔案的 PB 範圍。FlexGroups的一些理想使用案例包括AI / ML / DL、Big Data和分析、軟體建置、串流、檔案儲存庫等。Trident 會在佈建 FlexGroup Volume 時、使用指派給 SVM 的所有集合體。支援Trident也有下列考量：
FlexGroup

- 需要ONTAP 9.2版或更新版本。
- 截至本文撰寫時、FlexGroups僅支援NFS v3。
- 建議啟用SVM的64位元NFSv3識別碼。
- 建議的 FlexGroup 成員 / 磁碟區大小下限為 100GiB。
- FlexGroup 磁碟區不支援複製。

有關適用於 FlexGroups 的 FlexGroups 和工作負載的資訊、請參閱 "[NetApp FlexGroup Volume 最佳實務做法與實作指南](#)"。

若要在同一個環境中獲得進階功能和龐大規模、您可以執行多個Docker Volume外掛程式執行個體、其中一個使用「ONTAP-NAS」、另一個使用「ONTAP-NAS-經濟」。

Trident 的自訂 ONTAP 角色

您可以使用最低 Privileges 來建立 ONTAP 叢集角色、這樣就不需要使用 ONTAP 管理員角色來執行 Trident 中的作業。當您在 Trident 後端組態中包含使用者名稱時、Trident 會使用您建立的 ONTAP 叢集角色來執行作業。

如需建立 Trident 自訂角色的詳細資訊、請參閱"[Trident 自訂角色產生器](#)"。

使用 ONTAP CLI

1. 使用下列命令建立新角色：

```
security login role create <role_name\> -cmddirname "command" -access all  
-vserver <svm_name\>
```

2. 為 Trident 使用者建立使用者名稱：

```
security login create -username <user_name\> -application ontapi  
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver  
<svm_name\> -comment "user_description"
```

3. 將角色對應至使用者：

```
security login modify username <user_name\> -vserver <svm_name\> -role  
<role_name\> -application ontapi -application console -authmethod  
<password\>
```

使用 System Manager

在 ONTAP 系統管理員中執行下列步驟：

1. * 建立自訂角色 *：

- a. 若要在叢集層級建立自訂角色、請選取 * 叢集 > 設定 *。

(或) 若要在 SVM 層級建立自訂角色、請選取 * 儲存設備 > 儲存 VM >> required SVM 設定 > 使用者與角色 *。

- b. 選取 * 使用者和角色 * 旁的箭頭圖示 (* → *)。

- c. 在 * 角色 * 下選擇 **+Add**。

- d. 定義角色的規則、然後按一下 * 儲存 *。

2. * 將角色對應至 Trident 使用者 *：+ 在「* 使用者與角色 *」頁面上執行下列步驟：

- a. 在 * 使用者 * 下選取新增圖示 +。

- b. 選取所需的使用者名稱、然後在 * 角色 * 的下拉式功能表中選取角色。

- c. 按一下「* 儲存 *」。

如需詳細資訊、請參閱下列頁面：

- ["用於管理 ONTAP 的自訂角色"或"定義自訂角色"](#)
- ["與角色和使用者合作"](#)

範例 **ONTAP**：功能組態檔

<code>ontap-nas</code> 驅動程式的 NFS 範例

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "defaults": {
    "size": "10G",
    "spaceReserve": "none",
    "exportPolicy": "default"
  }
}
```

<code>ontap-nas-flexgroup</code> 驅動程式的 NFS 範例

```
{
  "version": 1,
  "storageDriverName": "ontap-nas-flexgroup",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "defaults": {
    "size": "100G",
    "spaceReserve": "none",
    "exportPolicy": "default"
  }
}
```

<code>ontap-nas-economy</code> 驅動程式的 NFS 範例

```
{
  "version": 1,
  "storageDriverName": "ontap-nas-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
```

<code>ontap-san</code> 驅動程式的 iSCSI 範例

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}
```

<code>ontap-san-economy</code> 驅動程式的 NFS 範例

```
{
  "version": 1,
  "storageDriverName": "ontap-san-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi_eco",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}
```

<code>ontap-san</code> 驅動程式的 NVMe / TCP 範例

```
{
  "version": 1,
  "backendName": "NVMeBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nvme",
  "username": "vsadmin",
  "password": "password",
  "sanType": "nvme",
  "useREST": true
}
```

元件軟體組態

除了全域組態值之外、使用Element軟體 (NetApp HCI / SolidFire) 時、也可使用這些選項。

選項	說明	範例
端點	<code>https://<login>:<password>@<mvip>/json-rpc/<element-version></code>	<code>https://admin:admin@192.168.160.3/json-rpc/8.0</code>
《VIP》	iSCSI IP位址和連接埠	10.0.0.7 : 3260
《天王名稱》	要使用的SolidFireF租戶 (如果找不到、請建立)	docker
《初始器IFACE》	將iSCSI流量限制為非預設介面時、請指定介面	default
《類型》	QoS規格	請參閱以下範例
"LegacyNamePrefix (名前置詞) "	升級版Trident安裝的首碼。如果您使用 1.3.2 之前的 Trident 版本、並使用現有的 Volume 執行升級、則必須設定此值、才能存取透過 Volume 名稱方法對應的舊 Volume。	netappdvp-

「Poolidfire - san」 驅動程式不支援Docker swarm。

元素軟體組態檔範例

```
{
  "version": 1,
  "storageDriverName": "solidfire-san",
  "Endpoint": "https://admin:admin@192.168.160.3/json-rpc/8.0",
  "SVIP": "10.0.0.7:3260",
  "TenantName": "docker",
  "InitiatorIFace": "default",
  "Types": [
    {
      "Type": "Bronze",
      "Qos": {
        "minIOPS": 1000,
        "maxIOPS": 2000,
        "burstIOPS": 4000
      }
    },
    {
      "Type": "Silver",
      "Qos": {
        "minIOPS": 4000,
        "maxIOPS": 6000,
        "burstIOPS": 8000
      }
    },
    {
      "Type": "Gold",
      "Qos": {
        "minIOPS": 6000,
        "maxIOPS": 8000,
        "burstIOPS": 10000
      }
    }
  ]
}
```

已知問題與限制

在搭配 Docker 使用 Trident 時、尋找已知問題和限制的相關資訊。

從舊版升級 **Trident Docker Volume** 外掛程式至 **20.10** 及更新版本、會導致升級失敗、且不會發生此類檔案或目錄錯誤。

因應措施

1. 停用外掛程式。

```
docker plugin disable -f netapp:latest
```

2. 移除外掛程式。

```
docker plugin rm -f netapp:latest
```

3. 提供額外的「config」參數、重新安裝外掛程式。

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant  
-all-permissions config=config.json
```

Volume名稱長度必須至少**2**個字元。



這是Docker用戶端的限制。用戶端會將單一字元名稱解譯為Windows路徑。"請參閱錯誤 25773"。

Docker swarm 具有某些行為、可防止 **Trident** 在每個儲存設備和驅動程式組合中支援它。

- Docker swarm目前使用Volume名稱、而非Volume ID做為其唯一的Volume識別碼。
- Volume要求會同時傳送至swarm叢集中的每個節點。
- Volume 外掛程式（包括 Trident）必須在 swarm 叢集中的每個節點上分別執行。由於 ONTAP 的運作方式、以及和 `ontap-san` 驅動程式的運作方式、`ontap-nas`、這些都是唯一能夠在這些限制範圍內運作的方法。

其餘的驅動程式可能會受到競爭狀況等問題的影響、導致在單一要求中建立大量磁碟區、而不需要明確的「贏家」；例如、Element的功能可讓磁碟區擁有相同名稱但不同ID。

NetApp已向Docker團隊提供意見回饋、但沒有任何未來追索的跡象。

如果配置的是某個功能區、則如果第二個功能區的一個或多個集合體與要配置的功能區相同、則不提供第二個功能區。**FlexGroup ONTAP FlexGroup FlexGroup FlexGroup**

最佳實務做法與建議

部署

部署 Trident 時、請使用此處列出的建議。

部署至專屬命名空間

"命名空間"在不同的應用程式之間提供管理上的分離、是資源共用的障礙。例如、某個命名空間的某個永久虛電路無法從另一個命名空間使用。Trident 為 Kubernetes 叢集中的所有命名空間提供 PV 資源、因此會運用 Privileges 提升的服務帳戶。

此外、存取Trident Pod可能會讓使用者存取儲存系統認證和其他敏感資訊。請務必確保應用程式使用者和管理應用程式無法存取Trident物件定義或Pod本身。

使用配額和範圍限制來控制儲存使用量

Kubernetes有兩項功能、一旦結合、就能提供強大的機制來限制應用程式的資源使用量。◦ "儲存配額機制" 可讓系統管理員針對每個命名空間來實作全域和儲存類別的容量和物件數使用限制。此外、請使用 "範圍限制" 確保在將要求轉送至資源配置程式之前、永久虛擬機器要求的最小值和最大值都在內。

這些值是以每個命名空間為基礎來定義、這表示每個命名空間都應該定義符合其資源需求的值。如需相關資訊、請參閱此處 "如何運用配額"。

儲存組態

NetApp產品組合中的每個儲存平台都有獨特的功能、無論應用程式是否為容器化的應用程式帶來好處。

平台總覽

Trident可搭配ONTAP 使用沒有一個平台比其他平台更適合所有應用程式和案例、不過在選擇平台時、應考慮應用程式和管理裝置團隊的需求。

您應該遵循主機作業系統的基礎最佳實務做法、以及您所使用的傳輸協定。或者、您可能想要考慮在可用的情況下、將應用程式最佳實務做法與後端、儲存類別和永久虛擬基礎架構設定整合、以最佳化特定應用程式的儲存。

最佳實務做法ONTAP Cloud Volumes ONTAP

瞭解設定ONTAP 適用於Cloud Volumes ONTAP Trident的功能性及功能性的最佳實務做法。

以下建議是設定ONTAP 以容器化工作負載為基礎的功能指南、這些功能會消耗Trident動態配置的磁碟區。每個項目都應考量及評估是否適合您的環境。

使用Trident專用的SVM

儲存虛擬機器 (SVM) 可隔離ONTAP 及管理各個客戶在一個系統上的區隔。將SVM專用於應用程式可委派權限、並可套用最佳實務做法來限制資源使用量。

SVM管理有多種選項可供選擇：

- 在後端組態中提供叢集管理介面、以及適當的認證、然後指定SVM名稱。
- 使用ONTAP 支援功能的支援中心或CLI、為SVM建立專屬的管理介面。
- 與NFS資料介面共用管理角色。

在每種情況下、介面都應該位於DNS中、而且在設定Trident時、應該使用DNS名稱。這有助於推動一些DR案例、例如不使用網路身分保留功能的SVM-DR。

不過、您並不偏好為SVM設定專屬或共享的管理LIF、不過您應該確保網路安全性原則符合您選擇的方法。無論如何、管理LIF都應透過DNS存取、以達到最大的靈活性 "SVM-DR" 與Trident搭配使用。

限制最大Volume數

根據軟體版本和硬體平台、系統可提供最大的Volume數。ONTAP請參閱 "[NetApp Hardware Universe](#)" 針對您的特定平台和ONTAP 版本、決定確切的限制。當磁碟區數用盡時、資源配置作業不僅會針對Trident、也會針對所有儲存要求失敗。

Trident的「ONTAP-NAS」和「ONTAP-SAN」驅動程式會為每個建立的Kubernetes持續Volume (PV) 提供FlexVolume。「ONTAP-NAS-P節約」驅動程式可為每200個PV建立約一個FlexVolume (可設定為50到300個)。「ONTAP-san經濟型」驅動程式可為每100個PV建立約一個FlexVolume (可設定為50到200個)。若要避免Trident佔用儲存系統上的所有可用磁碟區、您應該在SVM上設定限制。您可以從命令列執行此動作：

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

「最大用量 (max volume) 的值會根據您環境的幾項特定條件而有所不同：

- 在叢集中現有的Volume數量ONTAP
- 您預期在Trident外部配置其他應用程式的Volume數量
- Kubernetes應用程式預期會使用的持續磁碟區數量

「最大容量」值是ONTAP 指在整個叢集中所有節點上配置的總Volume、而非在個別ONTAP 的節點上配置的總Volume。因此ONTAP、您可能會遇到一些情況、例如、某個叢集節點的資源配置量可能遠高於或低於其他節點。

舉例ONTAP 來說、雙節點的「叢集」最多可裝載2000個FlexVolumes。將最大Volume數設為1250似乎非常合理。不過、如果只有 "集合體" 從某個節點指派給SVM、或是從某個節點指派的集合體無法根據 (例如、由於容量) 進行資源配置、則另一個節點會成為所有Trident資源配置磁碟區的目標。這表示在達到「最大磁碟區」值之前、該節點可能會達到磁碟區限制、進而影響使用該節點的Trident和其他Volume作業。您可以確保叢集中每個節點的集合體都指派給Trident使用的SVM、數量相等、藉此避免這種情況。

限制Trident所建立的Volume大小上限

若要設定Trident可建立的磁碟區大小上限、請在「backend.json」定義中使用「limitVolume Sizes」參數。

除了控制儲存陣列的磁碟區大小、您也應該善用Kubernetes功能。

限制 Trident 所建立的 FlexVols 大小上限

若要將 FlexVols 的最大大小設定為用於 ONTAP SAN 經濟型和 ONTAP NAS 經濟型驅動程式的集區、請使用 `limitVolumePoolSize` `backend.json` 定義中的參數。

設定 Trident 使用雙向 CHAP

您可以在後端定義中指定 CHAP 啟動器和目標使用者名稱和密碼、並在 SVM 上啟用 Trident 啟用 CHAP。使用 `useCHAP` 後端組態中的參數、Trident 會驗證 iSCSI 連線 ONTAP、以 CHAP 作為後端。

建立並使用 SVM QoS 原則

運用 ONTAP 套用至 SVM 的 SVM 的 SQoS 原則、限制 Trident 佈建磁碟區所耗用的 IOPS 數量。這對我們有幫助 "[預防欺凌](#)" 或失控的容器、避免影響 Trident SVM 以外的工作負載。

您可以在幾個步驟中建立 SVM 的 QoS 原則。如 ONTAP 需最準確的資訊、請參閱您的版次更新文件。以下範例建立 QoS 原則、將 SVM 可用的總 IOPS 限制為 5000。

```
# create the policy group for the SVM
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

此外、如果 ONTAP 您的版本支援此功能、您可以考慮使用 QoS 下限來保證容器化工作負載的處理量。調適性 QoS 與 SVM 層級原則不相容。

容器化工作負載專用的 IOPS 數量取決於許多層面。其中包括：

- 使用儲存陣列的其他工作負載。如果有其他工作負載與 Kubernetes 部署無關、請善用儲存資源、確保這些工作負載不會意外受到不良影響。
- 預期的工作負載會在容器中執行。如果將在容器中執行高 IOPS 需求的工作負載、低 QoS 原則會導致不良體驗。

請務必記住、在 SVM 層級指派的 QoS 原則會導致所有已配置給 SVM 的磁碟區共用相同的 IOPS 集區。如果其中一種或少數幾種容器化應用程式的 IOPS 需求較高、可能會成為其他容器化工作負載的一大功臣。如果是這種情況、您可能需要考慮使用外部自動化來指派每個 Volume QoS 原則。



如果您的版本早於 ONTAP 9.8、您應該將 QoS 原則群組指派給 SVM * Only *。

為 Trident 建立 QoS 原則群組

服務品質 (QoS) 可確保關鍵工作負載的效能不會因競爭工作負載而降級。支援 QoS 原則群組的 QoS 選項可用於磁碟區、並可讓使用者定義一或多個工作負載的處理量上限。ONTAP 如需 QoS 的詳細資訊、請參閱 "[保證 QoS 的處理量](#)"。

您可以在後端或儲存資源池中指定 QoS 原則群組、並將其套用至該資源池或後端中建立的每個磁碟區。

包含兩種QoS原則群組：傳統和可調適。ONTAP傳統原則群組可在IOPS中提供最大（或最小）的單位處理量（在較新版本中）。調適性QoS會自動將處理量調整至工作負載大小、並隨著工作負載大小變更、維持IOPS與TBs的比率。當您在大型部署中管理數百個或數千個工作負載時、這項優勢就相當顯著。

建立QoS原則群組時、請考量下列事項：

- 您應該在後端組態的「故障」區塊中設定「qosPolicy」金鑰。請參閱下列後端組態範例：

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 0.0.0.0
dataLIF: 0.0.0.0
svm: svm0
username: user
password: pass
defaults:
  qosPolicy: standard-pg
storage:
- labels:
  performance: extreme
  defaults:
    adaptiveQosPolicy: extremely-adaptive-pg
- labels:
  performance: premium
  defaults:
    qosPolicy: premium-pg
```

- 您應該為每個Volume套用原則群組、以便每個Volume都能獲得原則群組指定的整個處理量。不支援共用原則群組。

如需 QoS 原則群組的詳細資訊、請參閱 "[Sof 9.8 QoS命令ONTAP](#)"。

限制Kubernetes叢集成員存取儲存資源

限制存取Trident所建立的NFS磁碟區和iSCSI LUN、是Kubernetes部署安全態勢的重要元件。這樣做可防止非Kubernetes叢集一部分的主機存取磁碟區、並可能意外修改資料。

請務必瞭解命名空間是Kubernetes中資源的邏輯邊界。假設相同命名空間中的資源可以共用、但重要的是、沒有跨命名空間功能。這表示即使PV是全域物件、但只有在同一個命名空間中的Pod才能存取它們。確保命名空間在適當時用於提供分隔是非常重要的。

大多數組織對於Kubernetes內容中的資料安全性、主要關注的是、容器中的程序可以存取掛載到主機의儲存設備、但不適用於容器。["命名空間"](#)旨在防止這類入侵。不過、有一個例外：特殊權限容器。

與正常情況相比、特權容器的執行主機層級權限大幅增加。依預設不會拒絕這些功能、因此請務必使用停用該功能 "[Pod安全性原則](#)"。

對於需要從Kubernetes和外部主機存取的磁碟區、儲存設備應以傳統方式進行管理、由系統管理員引進PV、而

非由Trident管理。這可確保只有在Kubernetes和外部主機中斷連線且不再使用磁碟區時、才會銷毀儲存磁碟區。此外、也可以套用自訂匯出原則、以便從Kubernetes叢集節點和Kubernetes叢集以外的目標伺服器存取。

對於具有專用基礎架構節點（例如OpenShift）或其他節點無法排程使用者應用程式的部署、應使用個別的匯出原則、進一步限制對儲存資源的存取。這包括為部署至這些基礎架構節點的服務（例如OpenShift Metrics和記錄服務）、以及部署至非基礎架構節點的標準應用程式建立匯出原則。

使用專屬的匯出原則

您應該確保每個後端都有一個匯出原則、只允許存取Kubernetes叢集中的節點。Trident 可以自動建立及管理匯出原則。如此一來、Trident就能限制對Kubernetes叢集中節點所配置之磁碟區的存取、並簡化節點的新增/刪除作業。

或者、您也可以手動建立匯出原則、並以一或多個匯出規則填入、以處理每個節點存取要求：

- 使用「vserver匯出原則建立」ONTAP 的flexcli命令來建立匯出原則。
- 使用「vserver匯出原則規則create」ONTAP 的CLI命令、將規則新增至匯出原則。

執行這些命令可讓您限制哪些Kubernetes節點可以存取資料。

停用 showmount 適用於應用程式SVM

「show mount」功能可讓NFS用戶端查詢SVM、以取得可用的NFS匯出清單。部署到Kubernetes叢集的Pod可針對資料LIF發出「show mount -e」命令、並接收可用掛載的清單、包括無法存取的掛載。雖然這本身並不是安全威脅、但它確實提供不必要的資訊、可能有助於未獲授權的使用者連線至NFS匯出。

您應該使用SVM層級ONTAP 的CLI命令來停用「show mount」：

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

最佳實務做法SolidFire

瞭解設定SolidFire Trident之用的功能完善的功能。

建立SolidFire 支援帳戶

每個SolidFire 個驗證帳戶都代表唯一的磁碟區擁有者、並會收到自己的挑戰握手驗證傳輸協定（CHAP）認證資料。您可以使用帳戶名稱和相對CHAP認證、或是透過Volume存取群組、來存取指派給帳戶的磁碟區。帳戶最多可指派2、000個磁碟區、但一個磁碟區只能屬於一個帳戶。

建立QoS原則

如果您想建立並儲存可套用至許多Volume的標準化服務品質設定、請使用SolidFire 「服務品質（QoS）」原則。

您可以設定每個Volume的QoS參數。設定三個可設定的參數來定義QoS、以確保每個Volume的效能：最小IOPS、最大IOPS和爆發IOPS。

以下是4KB區塊大小的可能最小、最大和尖峰IOPS值。

IOPS參數	定義	最小價值	預設值	最大價值 (4KB)
最小IOPS	保證磁碟區效能等級。	50	50	15000
最大IOPS	效能不會超過此限制。	50	15000	20萬
暴增IOPS	在短時間暴增案例中允許的最大IOPS。	50	15000	20萬



雖然最大IOPS和爆發IOPS可設定為高達20、000、但實際的Volume最大效能卻受到叢集使用量和每節點效能的限制。

區塊大小和頻寬會直接影響IOPS的數量。隨著區塊大小增加、系統會將頻寬增加至處理較大區塊大小所需的層級。隨著頻寬增加、系統能夠達到的IOPS數量也隨之減少。請參閱 "[服務品質SolidFire](#)" 如需QoS和效能的詳細資訊、請參閱。

驗證SolidFire

Element支援兩種驗證方法：CHAP和Volume Access Groups (VAG)。CHAP使用CHAP傳輸協定驗證主機到後端的驗證。Volume存取群組可控制對其所配置之Volume的存取。NetApp建議使用CHAP進行驗證、因為它更簡單、而且沒有擴充限制。



Trident搭配增強的csi佈置程式、可支援使用CHAP驗證。VAG只能在傳統的非csi操作模式下使用。

CHAP驗證（驗證啟動器是否為預定的Volume使用者）僅支援帳戶型存取控制。如果您使用CHAP進行驗證、則有兩個選項可供使用：單向CHAP和雙向CHAP。單向CHAP使用SolidFire 驗證帳戶名稱和啟動器密碼來驗證Volume存取。雙向CHAP選項提供最安全的驗證磁碟區方法、因為磁碟區會透過帳戶名稱和啟動器密碼來驗證主機、然後主機會透過帳戶名稱和目標密碼來驗證磁碟區。

但是、如果無法啟用CHAP且需要VAG、請建立存取群組、然後將主機啟動器和磁碟區新增至存取群組。您新增至存取群組的每個IQN都可以使用或不使用CHAP驗證來存取群組中的每個磁碟區。如果iSCSI啟動器設定為使用CHAP驗證、則會使用帳戶型存取控制。如果iSCSI啟動器未設定為使用CHAP驗證、則會使用Volume Access Group存取控制。

哪裡可以找到更多資訊？

以下列出部分最佳實務做法文件。搜尋 "[NetApp資料庫](#)" 適用於最新版本。

《》 ONTAP

- "[NFS最佳實務與實作指南](#)"
- "[SAN管理指南](#)"（適用於iSCSI）
- "[適用於RHEL的iSCSI Express組態](#)"

元件軟體

- ["設定SolidFire 適用於Linux的功能"](#)

《》 NetApp HCI

- ["部署先決條件NetApp HCI"](#)
- ["存取NetApp部署引擎"](#)

應用程式最佳實務做法資訊

- ["MySQL ONTAP 的最佳實務做法"](#)
- ["MySQL SolidFire 的最佳實務做法"](#)
- ["NetApp SolidFire 的功能與Cassandra"](#)
- ["Oracle SolidFire 的最佳實務做法"](#)
- ["PostgreSQL SolidFire 的最佳實務做法"](#)

並非所有應用程式都有特定的準則、請務必與您的NetApp團隊合作並使用 ["NetApp資料庫"](#) 以尋找最新的文件。

整合 Trident

若要整合 Trident、下列設計和架構元素需要整合：驅動程式選擇和部署、儲存類別設計、虛擬集區設計、持續 Volume Claim（永久 Volume Claim）對使用 Trident 的儲存資源配置、Volume 作業和 OpenShift 服務部署的影響。

驅動程式選擇與部署

為您的儲存系統選取並部署後端驅動程式。

背後驅動程式ONTAP

以使用的傳輸協定和儲存系統上的磁碟區配置方式來區分後端驅動程式ONTAP。因此、在決定要部署的驅動程式時、請謹慎考量。

較高層級的應用程式若有需要共用儲存設備的元件（多個Pod存取相同的PVc）、則以NAS為基礎的驅動程式將是預設選擇、而區塊型iSCSI驅動程式則可滿足非共用儲存設備的需求。根據應用程式的需求、以及儲存設備和基礎架構團隊的舒適度來選擇傳輸協定。一般而言、大多數應用程式的差異不大、因此通常是根據是否需要共用儲存設備（如果有多個Pod需要同時存取）來決定。

可用ONTAP 的支援功能包括：

- 「ONTAP-NAS」：每個提供的PV都是ONTAP 完整的FlexVolume。
- 「ONTAP-NAS-EAS」：每個已配置的PV都是qtree、每個FlexVolume可設定的qtree數量（預設值為200）。
- 「ONTAP-NAS-flexgroup」：每個PV均以ONTAP FlexGroup 完整的形式配置、並使用指派給SVM的所有集合體。
- 「ONTAP-san」：每個已配置的PV都是其FlexVolume內的LUN。
- 「ONTAP-san經濟」：每個配置的PV都是LUN、每個FlexVolume有可設定的LUN數量（預設值為100）。

在這三種NAS驅動程式之間選擇、會對應用程式可用的功能產生一些影響。

請注意、在下表中、並非所有功能都是透過 Trident 公開的。如果需要這些功能、儲存管理員必須在資源配置後套用部分功能。上標註可區分每項功能和驅動程式的功能。

ASNAS驅動程式ONTAP	快照	複製	動態匯出原則	多重附加	QoS	調整大小	複寫
「ONTAP-NAS」	是的	是的	是註腳：5[]	是的	是註腳：1[]	是的	是註腳：1[]
《ONTAP-NANAS經濟》	是註腳：3[]	是註腳：3[]	是註腳：5[]	是的	是註腳：3[]	是的	是註腳：3[]
「ONTAP-NAA-flexgroup」	是註腳：1[]	否	是註腳：5[]	是的	是註腳：1[]	是的	是註腳：1[]

Trident 提供 2 個適用於 ONTAP 的 SAN 驅動程式、其功能如下所示。

支援SAN驅動程式ONTAP	快照	複製	多重附加	雙向CHAP	QoS	調整大小	複寫
「ONTAP-SAN」	是的	是的	是註腳：4[]	是的	是註腳：1[]	是的	是註腳：1[]
《ONTAP-san經濟》	是的	是的	是註腳：4[]	是的	是註腳：3[]	是的	是註腳：3[]

上述表格的註腳：Yes 註：1[]：非由 Trident 管理是腳註：2[]：由 Trident 管理，但非 PV 精細是註腳：3[]：非由 Trident 管理而非 PV 精細是註腳：4[]：支援原始區塊磁碟區是註腳：5[]：由 Trident 支援

非PV精細的功能會套用至整個FlexVolume、而所有PV（即共享FlexVols中的qtree或LUN）都會共用一個共同排程。

如以上表格所示、「ONTAP-NAS」與「ONTAP-NAS經濟」之間的大部分功能都相同。不過、由於「ONTAP-NAS經濟」驅動程式限制了以每個PV精細度控制排程的能力、因此這可能會特別影響您的災難恢復與備份規劃。對於想要在ONTAP 不支援的儲存設備上使用永久虛擬複製功能的開發團隊、只有在使用「ONTAP-NAS」、「ONTAP-SAN」或「ONTAP-SAN經濟」驅動程式時、才有可能做到這一點。



「Poolidfire - san」驅動程式也能複製PVCS。

背後驅動程式Cloud Volumes ONTAP

支援資料控管功能、並提供企業級的儲存功能、適用於各種使用案例、包括檔案共用、區塊層級儲存設備（NFS、SMB / CIFS及iSCSI）Cloud Volumes ONTAP。Cloud Volume ONTAP 的相容驅動程式為「ONTAP-NAS」、「ONTAP-NAS經濟」、「ONTAP-SAN」和「ONTAP-SAN經濟」。適用於ONTAP Azure的Cloud Volume供應、適用於ONTAP GCP的Cloud Volume供應。

Amazon FSXfor ONTAP Sendbackend驅動程式

Amazon FSX for NetApp ONTAP 可讓您運用熟悉的 NetApp 功能、效能和管理功能、同時充分利用在 AWS 上儲存資料的簡易性、敏捷度、安全性和擴充性。適用於 ONTAP 的 FSX 支援許多 ONTAP 檔案系統功能和管理

API。Cloud Volume ONTAP 的相容驅動程式就是 `ontap-nas`、`ontap-nas-economy`、`ontap-nas-flexgroup`、`ontap-san` 和 `ontap-san-economy`。

NetApp HCI / SolidFire 後端驅動程式

NetApp HCI / SolidFire 平台搭配使用的「Poolidfire」驅動程式、可協助管理員根據 QoS 限制、為 Trident 設定元素後端。如果您想要設計後端、以便針對 Trident 提供的磁碟區設定特定的 QoS 限制、請在後端檔案中使用「`type`」參數。管理員也可以使用「`limitVolume Siz`」參數、限制儲存設備上可建立的磁碟區大小。目前、磁碟區大小調整和磁碟區複寫等元素儲存功能不支援使用「`Poolidfire - san`」驅動程式。這些作業應透過 Element Software Web UI 手動完成。

驅動程式 SolidFire	快照	複製	多重附加	CHAP	QoS	調整大小	複寫
「 <code>olidfire - san</code> 」	是的	是的	是註腳： 2[]	是的	是的	是的	是註腳： 1[]

註腳：是註腳：1[]：非由 Trident 管理是註腳：2[]：支援原始區塊磁碟區

背後驅動程式 **Azure NetApp Files**

Trident 使用 `azure-netapp-files` 驅動程式來管理 "Azure NetApp Files" 服務。

有關此驅動程式及其設定方式 "適用於 Azure NetApp Files 的 Trident 後端組態" 的詳細資訊，請參閱。

驅動程式 Azure NetApp Files	快照	複製	多重附加	QoS	展開	複寫
《 <code>azure-NetApp-fil形</code> 》	是的	是的	是的	是的	是的	是註腳：1[]

註腳：Yes 腳註：1[]：非由 Trident 管理

在 Google Cloud 後端驅動程式上執行 **Cloud Volumes Service**

Trident 使用 `gcp-cvs` 驅動程式連結 Google Cloud 上的 Cloud Volumes Service。

驅動程式會 `gcp-cvs` 使用虛擬集區來抽象化後端、並允許 Trident 判斷磁碟區的放置位置。系統管理員會定義檔案中的虛擬集區 `backend.json`。儲存類別會使用選取器來依標籤識別虛擬資源池。

- 如果在後端定義虛擬集區、Trident 將嘗試在 Google Cloud 儲存池中建立一個磁碟區、而這些虛擬集區則受到限制。
- 如果未在後端定義虛擬集區、Trident 會從該區域的可用儲存集區中選取 Google Cloud 儲存集區。

若要在 Trident 上設定 Google Cloud 後端、您必須在後端檔案中指定 `projectNumber`、`apiRegion` 和 `apiKey`。您可以在 Google Cloud 主控台找到專案編號。API 金鑰取自您在 Google Cloud Volumes Service Cloud 上設定 API 存取功能時所建立的服務帳戶私密金鑰檔案。

如需 Cloud Volumes Service on Google Cloud 服務類型和服務層級的詳細資訊 "瞭解 Trident 對 CVS for GCP 的支援"、請參閱。

適用於Google Cloud驅動程式Cloud Volumes Service	快照	複製	多重附加	QoS	展開	複寫
《GCP—CVS》	是的	是的	是的	是的	是的	僅適用於CVS效能服務類型。



複寫附註

- 複寫並非由 Trident 管理。
- 該實體複本會建立在與來源Volume相同的儲存資源池中。

儲存層級設計

需要設定並套用個別的儲存類別、才能建立Kubernetes儲存類別物件。本節將討論如何為應用程式設計儲存類別。

特定後端使用率

篩選功能可在特定的儲存類別物件內使用、以決定要搭配該特定儲存類別使用的儲存資源池或集區集區集區。儲存類別可設定三組篩選器：「儲存設備」、「其他儲存設備」及/或「排除儲存設備」。

此 `storagePools` 參數有助於將儲存限制為符合任何指定屬性的集區集。此 `additionalStoragePools` 參數用於擴充 Trident 用於資源配置的集區集區集、以及由屬性和參數所選取的集區集 `storagePools`。您可以單獨使用參數或同時使用兩者、以確保已選取適當的儲存資源池集區集區。

「exclude StoragePools」參數是用來明確排除列出的符合屬性的集區集區集區集區。

模擬QoS原則

如果您想設計儲存類別來模擬服務品質原則、請建立儲存類別、並將「媒體」屬性設定為「HDD」或「SD」。根據儲存類別中提及的「媒體」屬性、Trident會選擇適當的後端、以提供「HDD」或「sd」集合體、以符合媒體屬性、然後將磁碟區的資源配置導向特定的集合體。因此、我們可以建立儲存等級Premium、將「媒體」屬性設為「sd」、可歸類為優質QoS原則。我們可以建立另一個儲存類別標準、將媒體屬性設為「HDD」、並將其歸類為標準QoS原則。我們也可以使用儲存類別中的「IOPS」屬性、將資源配置重新導向至可定義為QoS原則的元素應用裝置。

根據特定功能使用後端

儲存類別可設計用於將Volume資源配置導向特定後端、啟用精簡與完整資源配置、快照、複製及加密等功能。若要指定要使用的儲存設備、請建立儲存設備類別、以指定啟用所需功能的適當後端。

虛擬資源池

所有 Trident 後端均可使用虛擬集區。您可以使用 Trident 提供的任何驅動程式、為任何後端定義虛擬集區。

虛擬集區可讓系統管理員在後端建立抽象層級、以便透過「儲存類別」加以參考、以提高磁碟區在後端的靈活度與效率。不同的後端可以使用相同的服務類別來定義。此外、您也可以在相同的後端上建立多個儲存資源池、但其特性不同。當儲存類別設定為具有特定標籤的選取器時、Trident 會選擇符合所有選取器標籤的後端來放置磁碟區。如果儲存類別選取器標籤符合多個儲存集區、Trident 將會選擇其中一個標籤來配置磁碟區。

虛擬資源池設計

建立後端時、您通常可以指定一組參數。系統管理員無法以相同的儲存認證和一組不同的參數來建立另一個後端。隨著虛擬資源池的推出、這個問題已經減輕。虛擬集區是後端與Kubernetes儲存類別之間的層級抽象、可讓系統管理員定義參數及標籤、並以不受後端限制的方式透過Kubernetes儲存類別做為選取元來參考。您可以使用 Trident 為所有支援的 NetApp 後端定義虛擬集區。這份清單包括SolidFire/NetApp HCI、ONTAP 《關於Cloud Volumes Service GCP的功能、功能、功能、功能Azure NetApp Files 、功能、以及



定義虛擬資源池時、建議您不要嘗試重新排列後端定義中現有虛擬資源池的順序。此外、建議您不要編輯/修改現有虛擬資源池的屬性、改為定義新的虛擬資源池。

模擬不同的服務層級/QoS

您可以設計虛擬集區來模擬服務類別。使用適用於Azure NetApp Files 支援功能的Cloud Volume Service for效益的虛擬資源池實作、讓我們來看看如何設定不同的服務類別。使用代表不同效能層級的多個標籤來設定 Azure NetApp Files 後端。設定 `servicelevel` 並在每個標籤下新增其他必要的層面。現在請建立不同的Kubernetes儲存類別、以便對應至不同的虛擬資源池。使用 `parameters.selector` 欄位中、每個StorageClass會呼叫哪些虛擬資源池可用於裝載Volume。

指派特定的層面組合

可從單一儲存後端設計多個具有特定層面的虛擬集區。若要這麼做、請使用多個標籤來設定後端、並在每個標籤下設定所需的層面。現在、請使用建立不同的Kubernetes儲存類別 `parameters.selector` 對應至不同虛擬資源池的欄位。在後端上進行資源配置的磁碟區、將會在所選的虛擬資源池中定義各個層面。

會影響儲存資源配置的永久儲存設備特性

建立 PVC 時、超出所要求儲存類別的部分參數可能會影響 Trident 資源配置決策程序。

存取模式

透過永久虛擬網路申請儲存時、其中一個必填欄位是存取模式。所需的模式可能會影響所選的後端、以裝載儲存要求。

Trident 將嘗試將使用的儲存傳輸協定與根據下列對照表所指定的存取方法配對。這與基礎儲存平台無關。

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
iSCSI	是的	是的	是 (原始區塊)
NFS	是的	是的	是的

如果要求將ReadWriteMany永久虛擬磁碟提交至Trident部署、但未設定NFS後端、則不會配置任何磁碟區。因此、申請者應使用適合其應用程式的存取模式。

Volume作業

修改持續磁碟區

持續磁碟區除了兩個例外、都是Kubernetes中不可變的物件。建立後、即可修改回收原則和大小。不過、這並不會妨礙磁碟區的某些層面在 Kubernetes 之外進行修改。這可能是理想的做法、以便針對特定應用程式自訂磁碟區、確保容量不會意外耗用、或是單純地將磁碟區移至不同的儲存控制器。



Kubernetes樹狀目錄內建資源配置程式目前不支援NFS或iSCSI PV的磁碟區大小調整作業。Trident 支援同時擴充 NFS 和 iSCSI 磁碟區。

PV的連線詳細資料無法在建立後修改。

建立隨需磁碟區快照

Trident 支援隨需建立磁碟區快照、以及使用 CSI 架構從快照建立 PVC。Snapshot提供便利的方法來維護資料的時間點複本、並使Kubernetes中的來源PV在生命週期上獨立不受影響。這些快照可用於複製PVCS。

從快照建立磁碟區

Trident 也支援從磁碟區快照建立 PersistentVolumes。若要達成此目標、只要建立 PersistentVolume Claim、並將提及作為建立磁碟區所需的快照即可 `datasource`。Trident 會建立一個含有快照資料的磁碟區來處理此 PVC。有了這項功能、您可以跨區域複製資料、建立測試環境、完整取代毀損或毀損的正式作業磁碟區、或擷取特定檔案和目錄、然後將它們傳輸到其他附加磁碟區。

在叢集中移動磁碟區

儲存管理員能夠在ONTAP 整個叢集中的集合體和控制器之間、不中斷營運地將磁碟區移至儲存使用者。只要目的地 Aggregate 是 Trident 使用的 SVM 具有存取權、此作業就不會影響 Trident 或 Kubernetes 叢集。重要的是、如果新增 Aggregate 至 SVM、則需要重新將後端新增至 Trident 以重新整理。這會觸發 Trident 重新清查 SVM、以便辨識新的 Aggregate。

不過、Trident 並不自動支援在後端之間移動磁碟區。這包括在同一個叢集中的 SVM 之間、叢集之間或不同的儲存平台上（即使該儲存系統是連線至 Trident 的儲存系統）。

如果將磁碟區複製到其他位置、則可使用 Volume 匯入功能將目前的磁碟區匯入 Trident。

展開Volume

Trident 支援調整 NFS 和 iSCSI PV 的大小。這可讓使用者透過Kubernetes層直接調整磁碟區大小。所有主要的NetApp儲存平台皆可進行Volume擴充、包括ONTAP：NetApp、SolidFire/NetApp HCI及Cloud Volumes Service 背後端點。若要稍後允許擴充、請在與該磁碟區相關的 StorageClass 中設定 `allowVolumeExpansion` 為 `true`。每當需要調整「持續 Volume」的大小時、請將「持續 Volume」宣告中的註釋編輯 `spec.resources.requests.storage` 為所需的 Volume 大小。Trident會自動調整儲存叢集上的磁碟區大小。

將現有磁碟區匯入Kubernetes

Volume匯入功能可將現有的儲存磁碟區匯入Kubernetes環境。目前支援的驅動程式有「ONTAP-NAS」、「ONTAP-NAs-flexgroup」、「Poolidfire - san」、「azure-NetApp-fil卻」和「GCP - CVS」。當將現有應用程式移轉至Kubernetes或發生災難恢復時、此功能非常實用。

使用 ONTAP 和 `solidfire-san` 驅動程式時、請使用命令 `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` 將現有的磁碟區匯入 Kubernetes、以便由 Trident 管理。匯入 Volume 命令中使用的 PVC YAML 或 JSON 檔案會指向將 Trident 識別為資源配置程式的儲存類別。使用NetApp HCI / SolidFire後端時、請確定磁碟區名稱是唯一的。如果磁碟區名稱重複、請將磁碟區複製成唯一名稱、以便磁碟區匯入功能能夠區分它們。

如果 `azure-netapp-files` 使用或 `gcp-cvs` 驅動程式、請使用命令 `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` 將磁碟區匯入 Kubernetes、以便由 Trident 管理。如此可確保唯一的Volume

參考。

執行上述命令時、Trident 會在後端找到該 Volume 並讀取其大小。它會自動新增（並在必要時覆寫）已設定的 PVC Volume Size。然後 Trident 建立新的 PV、Kubernetes 會將 PVC 與 PV 連結起來。

如果部署的容器需要特定匯入的PVc、則會保持擱置狀態、直到PVC/PV配對透過Volume匯入程序繫結為止。在PVC/PV配對繫結之後、如果沒有其他問題、則應啟動容器。

部署OpenShift服務

OpenShift 加值叢集服務可為叢集管理員和託管的應用程式提供重要功能。這些服務所使用的儲存設備可以使用節點本機資源進行資源配置、但這通常會限制服務的容量、效能、可恢復性及永續性。運用企業儲存陣列來提供這些服務的容量、可大幅改善服務品質、不過OpenShift和儲存管理員應該密切合作、以決定每個服務的最佳選項。Red Hat文件應充分運用、以判斷需求、並確保符合規模調整與效能需求。

登錄服務

登錄的儲存設備部署與管理已記錄在中 ["NetApp.IO"](#) 在中 ["部落格"](#)。

記錄服務

如同其他OpenShift服務、記錄服務是使用Ansible搭配庫存檔案所提供的組態參數（即k.a.）來部署主機、提供給教戰手冊。其中包括兩種安裝方法：在初始OpenShift安裝期間部署記錄、以及在安裝OpenShift之後部署記錄。



從Red Hat OpenShift版本3.9起、官方文件建議您不要使用NFS來執行記錄服務、因為您擔心資料毀損。這是以Red Hat測試其產品為基礎。ONTAP NFS 伺服器沒有這些問題、而且可以輕鬆地備份記錄部署。最後、記錄服務的通訊協定選擇取決於您、只要知道兩者在使用NetApp平台時都能順利運作、而且如果您偏好NFS、就沒有理由不使用NFS。

如果您選擇使用NFS搭配記錄服務、則必須將Ansible變數「openshift_enable_unsupported_configurations」設為「true」、以避免安裝程式失敗。

開始使用

記錄服務可選擇性地同時部署給應用程式、以及OpenShift叢集本身的核心作業。如果您選擇部署作業記錄、將變數「openshift_logging_use」指定為「true」、就會建立兩個服務執行個體。控制作業記錄執行個體的變數包含「ops」、而應用程式執行個體則不包含。

根據部署方法設定 Ansible 變數非常重要、如此才能確保基礎服務使用正確的儲存設備。讓我們來看看每種部署方法的選項。



下表僅包含與記錄服務相關的儲存組態變數。您可以在中找到其他選項 ["RedHat OpenShift記錄文件"](#) 應根據您的部署情況來審查、設定及使用。

下表中的變數會使用提供的詳細資料、產生Ansible教戰手冊、為記錄服務建立PV和PVc。這種方法的彈性遠低於OpenShift安裝後使用元件安裝方針、不過如果您有現有的磁碟區可用、這是一個選項。

變動	詳細資料
"openshift_logging_storage_gin"	設定為「NFS」、讓安裝程式為記錄服務建立NFS PV。

變動	詳細資料
"openshift_logging_storage主機"	NFS主機的主機名稱或IP位址。這應該設定為虛擬機器的資料LIF。
"openshift_logging_storage、nfs_directory"	NFS匯出的掛載路徑。例如、如果磁碟區已連接為「/openshift_logging」、您就會將該路徑用於此變數。
"openshift_logging_storage磁碟區名稱"	要建立之PV的名稱、例如「PV_ose記錄」。
"openshift_logging_storage磁碟區大小"	NFS匯出的大小、例如「100Gi」。

如果您的OpenShift叢集已在執行中、因此已部署及設定Trident、則安裝程式可以使用動態資源配置來建立磁碟區。需要設定下列變數。

變動	詳細資料
「openshift_logging_es_PVC_Dynamic」	設為true可使用動態資源配置的磁碟區。
「openshift_logging_es_PVC_storage_class_name」	將在PVC中使用的儲存類別名稱。
「openshift_logging_es_PVC_size」	在永久虛擬磁碟中要求的磁碟區大小。
「openshift_logging_es_PVC_prefix」	記錄服務使用的PVCS前置詞。
「openshift_logging_es_ops_PVC_Dynamic」	設為「true」、以動態配置的磁碟區用於作業記錄執行個體。
「openshift_logging_es_ops_PVC_storage儲存設備類別名稱」	作業記錄執行個體的儲存類別名稱。
「openshift_logging_es_ops_PVC_Size」	作業執行個體的Volume要求大小。
「openshift_logging_es_ops_PVC_prefix」	ops執行個體PVCS的前置詞。

部署記錄堆疊

如果您將記錄部署為初始OpenShift安裝程序的一部分、則只需遵循標準部署程序即可。Ansible會設定及部署所需的服務和OpenShift物件、以便在可執行的完成後立即提供服務。

不過、如果您在初始安裝之後進行部署、Ansible將需要使用元件方針。不同版本的OpenShift可能會稍微改變此程序、因此請務必閱讀並遵循 ["RedHat OpenShift Container Platform 3.11文件"](#) 適用於您的版本。

度量服務

度量服務可針對OpenShift叢集的狀態、資源使用率及可用度、提供寶貴的資訊給系統管理員。此外、也需要Pod自動擴充功能、許多組織會使用指標服務的資料來支付費用和/或顯示應用程式。

如同記錄服務和OpenShift整體、Ansible可用於部署度量服務。此外、與記錄服務一樣、度量服務也可以在叢集初始設定期間或使用元件安裝方法在其運作後進行部署。下表包含在設定度量服務的持續儲存時、重要的變數。



下表僅包含與度量服務相關的儲存組態相關變數。文件中還有許多其他選項、您應該根據部署情況來檢閱、設定及使用。

變動	詳細資料
"openshift_imization_storage類型"	設定為「NFS」、讓安裝程式為記錄服務建立NFS PV。
"openshift_imization_storage主機"	NFS主機的主機名稱或IP位址。這應該設定為SVM的資料LIF。
"openshift_imization_storage、nfs_directory"	NFS匯出的掛載路徑。例如、如果磁碟區已連接為「/openshift_度量」、您就會使用該路徑來處理此變數。
"openshift_imization_storage磁碟區名稱"	要建立之PV的名稱、例如「PV_ose度量」。
"openshift_imization_storage磁碟區大小"	NFS匯出的大小、例如「100Gi」。

如果您的OpenShift叢集已在執行中、因此已部署及設定Trident、則安裝程式可以使用動態資源配置來建立磁碟區。需要設定下列變數。

變動	詳細資料
"openshift_imization_cassandra_PVC_prefix"	用於度量PVCS的前置詞。
"openshift_imization_cassandra_PVC_Size"	要要求的磁碟區大小。
"openshift_imensits_cassandra儲存設備類型"	用於度量的儲存類型、必須設定為動態、Ansible才能建立具有適當儲存類別的PVCS。
"openshift_imization_cassanda_PVC_storage_class_name"	要使用的儲存類別名稱。

部署度量服務

在您的主機/庫存檔案中定義適當的可Ansible變數後、使用Ansible部署服務。如果您是在OpenShift安裝時間進行部署、則會自動建立及使用PV。如果您是使用元件教戰手冊進行部署、則在安裝 OpenShift 之後、Ansible 會建立所需的任何 PVCS、並在 Trident 為其提供儲存設備之後、部署服務。

上述變數及部署程序可能會隨OpenShift的每個版本而變更。請務必檢閱並遵循 ["RedHat的OpenShift部署指南"](#) 以供您的環境使用。

資料保護與災難恢復

瞭解使用 Trident 建立的 Trident 和磁碟區的保護與還原選項。對於每個應用程式、您都應該有持續性需求的資料保護與還原策略。

Trident 複寫與還原

您可以建立備份、以便在發生災難時還原 Trident。

Trident 複寫

Trident 使用 Kubernetes CRD 來儲存及管理其本身的狀態、並使用 Kubernetes 叢集 etcd 來儲存其中繼資料。

步驟

1. 使用備份 Kubernetes 叢集 etcd "[Kubernetes : 備份 etcd 叢集](#)"。
2. 將備份產出工件放在 FlexVol 上。



我們建議您保護 FlexVol 所在的 SVM 、並與另一個 SVM 建立 SnapMirror 關係。

Trident 恢復

您可以使用 Kubernetes CRD 和 Kubernetes 叢集 etcd 快照來復原 Trident 。

步驟

1. 從目的地 SVM 、將包含 Kubernetes etcd 資料檔案和憑證的磁碟區掛載到將設定為主要節點的主機上。
2. 複製下 Kubernetes 叢集的所有必要憑證 `/etc/kubernetes/pki` 以及下的 etcd 成員檔案 `/var/lib/etcd`。
3. 使用從 etcd 備份還原 Kubernetes 叢集 "[Kubernetes : 還原 etcd 叢集](#)"。
4. 執行 `kubectl get crd` 若要驗證所有 Trident 自訂資源都已出現、請擷取 Trident 物件、以驗證所有資料是否可用。

SVM 複寫與還原

Trident 無法設定複寫關係、不過儲存管理員可以使用 "[ONTAP SnapMirror](#)"複寫 SVM 。

發生災難時、您可以啟動 SnapMirror 目的地 SVM 、開始提供資料服務。系統還原時、您可以切換回主要系統。

關於這項工作

使用 SnapMirror SVM 複寫功能時、請考量下列事項：

- 您應該為每個啟用 SVM-DR 的 SVM 建立不同的後端。
- 設定儲存類別、僅在需要時才選取複寫的後端、以避免將不需要複寫的磁碟區佈建到支援 SVM-DR 的後端。
- 應用程式管理員應瞭解複寫的額外成本與複雜度、並在開始此程序之前仔細考慮其還原計畫。

SVM 複寫

您可以使用 "[ONTAP : SnapMirror SVM 複寫](#)" 建立 SVM 複寫關係。

SnapMirror 可讓您設定選項、以控制要複寫的內容。您需要知道您在進行預先設定時所選擇 [使用 Trident 進行 SVM 恢復](#) 的選項。

- "[-identity 保留為真](#)" 複寫整個 SVM 組態。
- "[-discard 配置網路](#)" 不包括生命和相關的網路設定。
- "[-identity 保留錯誤](#)" 僅複寫磁碟區和安全組態。

使用 Trident 進行 SVM 恢復

Trident 不會自動偵測 SVM 故障。發生災難時、管理員可以手動啟動 Trident 容錯移轉至新的 SVM 。

步驟

1. 取消已排程和持續的 SnapMirror 傳輸、中斷複寫關係、停止來源 SVM、然後啟動 SnapMirror 目的地 SVM。
2. 如果您指定 `-identity-preserve false` 或 `-discard-config network` 設定 SVM 複寫時、請更新 `managementLIF` 和 `dataLIF` 在 Trident 後端定義檔案中。
3. 確認 `storagePrefix` 存在於 Trident 後端定義檔案中。此參數無法變更。省略 `storagePrefix` 將導致後端更新失敗。
4. 更新所有必要的後端、以反映新的目的地 SVM 名稱、使用：

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n  
<namespace>
```

5. 如果您指定 `-identity-preserve false` 或 `discard-config network`、您必須退回所有應用程式 Pod。



如果您指定 `-identity-preserve true`、則當目的地 SVM 啟動時、Trident 所佈建的所有磁碟區都會開始提供資料。

Volume 複寫與還原

Trident 無法設定 SnapMirror 複寫關係、不過儲存管理員可以使用"[ONTAP SnapMirror 複寫與還原](#)"複寫 Trident 建立的磁碟區。

然後，您可以使用將恢復的卷導入 Trident "[tridentctl Volume 匯入](#)"。



匯入不受支援 `ontap-nas-economy`、`ontap-san-economy` 或 `ontap-flexgroup-economy` 驅動程式：

Snapshot 資料保護

您可以使用下列項目來保護及還原資料：

- 外部快照控制器和 CRD、用於建立持續磁碟區（PV）的 Kubernetes Volume 快照。

"[Volume 快照](#)"

- ONTAP 快照可還原磁碟區的全部內容、或是還原個別檔案或 LUN。

"[ONTAP 快照](#)"

安全性

安全性

請使用此處列出的建議、確保 Trident 安裝安全無虞。

在自己的命名空間中執行 Trident

請務必防止應用程式、應用程式管理員、使用者和管理應用程式存取 Trident 物件定義或 Pod、以確保可靠的儲存設備、並封鎖潛在的惡意活動。

要將其他應用程序和用戶與 Trident 分開，請始終在其自己的 Kubernetes 命名空間中安裝 Trident (`trident`)。將 Trident 置於自己的命名空間中、可確保只有 Kubernetes 管理人員能夠存取 Trident Pod、以及儲存在命名 CRD 物件中的成品（例如後端和 CHAP 機密、如果適用）。您應確保只允許系統管理員存取 Trident 命名空間、進而存取 `tridentctl` 應用程式。

使用CHAP驗證搭配ONTAP 使用支援SAN的功能

Trident 支援 ONTAP SAN 工作負載的 CHAP 型驗證（使用 `ontap-san` 和 `ontap-san-economy` 驅動程式）。NetApp 建議在主機和儲存後端之間使用 Trident 的雙向 CHAP 進行驗證。

對於使用 SAN 儲存驅動程式的 ONTAP 後端、Trident 可以設定雙向 CHAP、並透過管理 CHAP 使用者名稱和機密 `tridentctl`。請參閱["準備使用ONTAP 支援的SAN驅動程式來設定後端"](#)以瞭解 Trident 如何在 ONTAP 後端上設定 CHAP。

使用CHAP驗證NetApp HCI 搭配不景和SolidFire 不景的後端

NetApp建議部署雙向CHAP、以確保主機與NetApp HCI 支援功能及SolidFire 支援功能之間的驗證。Trident 使用的是每個租戶包含兩個 CHAP 密碼的秘密物件。安裝 Trident 時、它會管理 CHAP 機密、並將其儲存在相關 PV 的 CR 物件中 `tridentvolume`。建立 PV 時、Trident 會使用 CHAP 機密來啟動 iSCSI 工作階段、並透過 CHAP 與 NetApp HCI 和 SolidFire 系統通訊。



由 Trident 建立的磁碟區不會與任何 Volume 存取群組相關聯。

搭配 NVE 和 NAE 使用 Trident

NetApp ONTAP 支援閒置資料加密、可在磁碟遭竊、退回或重新使用時、保護敏感資料。如需詳細資訊、請參閱["設定NetApp Volume Encryption總覽"](#)。

- 如果在後端上啟用 NAE、則 Trident 中配置的任何 Volume 都將啟用 NAE。
- 如果後端未啟用 NAE、除非您在後端組態中將 NVE 加密旗標設定為、否則在 Trident 中配置的任何 Volume 都將啟用 `NVE false`。

在啟用 NAE 的後端 Trident 中建立的磁碟區必須加密 NVE 或 NAE。



- 您可以在 Trident 後端組態中將 NVE 加密旗標設定為 `true`、以覆寫 NAE 加密、並以每個磁碟區為基礎使用特定的加密金鑰。
- 在啟用 NAE 的後端上、將 NVE 加密旗標設定為 `false`、會建立啟用 NAE 的 Volume。您無法透過將 NVE 加密旗標設定為來停用 NAE 加密 `false`。

- 您可以在 Trident 中手動建立 NVE Volume、方法是將 NVE 加密旗標明確設定為 `true`。

如需後端組態選項的詳細資訊、請參閱：

- ["支援SAN組態選項ONTAP"](#)
- ["ASNAS組態選項ONTAP"](#)

Linux統一化金鑰設定 (LUKS)

您可以啟用 Linux 統一金鑰設定 (LUKS) 來加密 Trident 上的 ONTAP SAN 和 ONTAP SAN 經濟磁碟區。Trident 支援使用複雜密碼的旋轉和磁碟區擴充、適用於使用 LUKS 加密的磁碟區。

在 Trident 中，LUKS 加密的磁碟區使用 AES-XTS-plain64 cypher 和模式"NIST"，如所建議。

開始之前

- 工作者節點必須安裝密碼設定2.1或更高版本（但低於3.0）。如需詳細資訊、請造訪 ["Gitlab：密碼設定"](#)。
- 基於效能考量、我們建議工作節點支援進階加密標準新增指令 (AES-NI)。若要驗證AES-NI支援、請執行下列命令：

```
grep "aes" /proc/cpuinfo
```

如果沒有歸還任何內容、您的處理器就不支援AES-NI。如需AES-NI的詳細資訊、請造訪：["Intel：進階加密標準指令 \(AES-NI\)"](#)。

啟用LUKS加密

您可以使用Linux Unified Key Setup (LUKS) 來啟用每個Volume、主機端的加密功能、以利ONTAP 執行SAN 和ONTAP 支援SAN經濟效益的磁碟區。

步驟

1. 在後端組態中定義LUKS加密屬性。如需ONTAP 有關支援不支援SAN的後端組態選項的詳細資訊、請參閱 ["支援SAN組態選項ONTAP"](#)。

```
"storage": [  
  {  
    "labels":{"luks": "true"},  
    "zone":"us_east_1a",  
    "defaults": {  
      "luksEncryption": "true"  
    }  
  },  
  {  
    "labels":{"luks": "false"},  
    "zone":"us_east_1a",  
    "defaults": {  
      "luksEncryption": "false"  
    }  
  },  
]
```

2. 使用 `parameters.selector` 使用LUKS加密定義儲存資源池。例如：

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}

```

3. 建立包含LUKS通關密碼的秘密。例如：

```

kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA

```

限制

LUKS加密磁碟區無法利用ONTAP 重複資料刪除技術與壓縮技術。

用於匯入 **LUKS Volume** 的後端組態

若要匯入 LUKS Volume、您必須在後端將設 `luksEncryption` 為 `'true'`。 `luksEncryption` 選項告訴 Trident 卷是否符合 LUKS (`'false'`) (`'true'` 或不符合 LUKS)，如下例所示。

```

version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'

```

用於匯入 LUKS Volume 的 PVC 組態

若要動態匯入 LUKS Volume、請將註釋設 `trident.netapp.io/luksEncryption` 為 `true`、並在 PVC 中包含啟用 LUKS 的儲存類別、如本範例所示。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

旋轉LUKS複雜密碼

您可以旋轉LUKS複雜密碼並確認輪調。



請勿忘記密碼、除非您已驗證它不再被任何磁碟區、快照或機密所引用。如果參考的通關密碼遺失、您可能無法掛載磁碟區、而且資料將保持加密且無法存取。

關於這項工作

如果在指定新的LUKS通關密碼之後建立裝載磁碟區的Pod、則會發生LUKS通關密碼循環。建立新的 Pod 時、Trident 會將磁碟區上的 LUKS 複雜密碼與機密中的作用中複雜密碼進行比較。

- 如果磁碟區上的通關密碼與機密中的作用中通關密碼不相符、就會發生輪調。
- 如果磁碟區上的通關密碼與機密中的作用中通關密碼相符 `previous-luks-passphrase` 參數被忽略。

步驟

1. 新增 `node-publish-secret-name` 和 `node-publish-secret-namespace` `StorageClass`參數。例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. 識別磁碟區或快照上的現有密碼。

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames: ["A"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames: ["A"]
```

3. 更新磁碟區的LUKS機密、以指定新的和先前的密碼。確保 `previous-luke-passphrase-name` 和 `previous-luks-passphrase` 請與先前的通關密碼相符。

```
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: B
  luks-passphrase: secretB
  previous-luks-passphrase-name: A
  previous-luks-passphrase: secretA
```

4. 建立新的Pod以掛載Volume。這是啟動旋轉所需的。
5. 確認複雜密碼已旋轉。

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames: ["B"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames: ["B"]
```

結果

只有在磁碟區和快照上傳回新的通關密碼時、才會旋轉通關密碼。



例如、如果傳回兩個複雜密碼 `luksPassphraseNames: ["B", "A"]`、旋轉不完整。您可以觸發新的Pod以嘗試完成旋轉。

啟用Volume擴充

您可以在LUKS加密的Volume上啟用Volume擴充。

步驟

1. 啟用 `CSINodeExpandSecret` 功能閘道 (beta 1.25+)。請參閱 ["Kubernetes 1.25：使用Secrets進行節點導向的SCSI Volume擴充"](#) 以取得詳細資料。
2. 新增 `node-expand-secret-name` 和 `node-expand-secret-namespace` `StorageClass`參數。例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

結果

當您啟動線上儲存擴充時、kubelet會將適當的認證資料傳遞給驅動程式。

使用 Trident Protect 保護應用程式

瞭解 Trident Protect

NetApp Trident Protect 提供進階的應用程式資料管理功能，可強化由 NetApp ONTAP 儲存系統和 NetApp Trident CSI 儲存資源配置程式所支援的狀態化 Kubernetes 應用程式的功能與可用性。Trident Protect 可簡化跨公有雲和內部部署環境的容器化工作負載管理，保護和移動。它也透過 API 和 CLI 提供自動化功能。

您可以 Trident 建立自訂資源（CRS），或使用 Trident Protect CLI 來保護應用程式。

接下來呢？

您可以在安裝 Trident Protect 要求之前先瞭解相關資訊：

- ["Trident 保護需求"](#)

安裝 Trident Protect

Trident 保護需求

首先，請確認您的營運環境，應用程式叢集，應用程式和授權是否準備就緒。確保您的環境符合這些需求，以部署及操作 Trident Protect。

Trident 保護 Kubernetes 相容性

Trident Protect 可與多種完全託管且自我管理的 Kubernetes 產品相容，包括：

- Amazon Elastic Kubernetes Service（EKS）
- Google Kubernetes Engine（GKE）
- Microsoft Azure Kubernetes 服務（英文）
- Red Hat OpenShift
- SUSE Rancher
- VMware Tanzu 產品組合
- 上游 Kubernetes

Trident 保護儲存後端相容性

Trident Protect 支援下列儲存設備後端：

- Amazon FSX for NetApp ONTAP 產品
- Cloud Volumes ONTAP
- ONTAP 儲存陣列

- Google Cloud NetApp Volumes
- Azure NetApp Files

確保您的儲存後端符合下列需求：

- 確保連接至叢集的 NetApp 儲存設備使用 Astra Trident 24.02 或更新版本（建議使用 Trident 24.10）。
 - 如果 Astra Trident 早於 24.06.1 版，且您計畫使用 NetApp SnapMirror 災難恢復功能，則需要手動啟用 Astra 控制項資源配置程式。
- 確保您擁有最新的 Astra 控制備份程式（預設為 Astra Trident 24.06.1）。
- 確保您擁有 NetApp ONTAP 儲存後端。
- 請確定您已設定物件儲存貯體以儲存備份。
- 建立您計畫用於應用程式或應用程式資料管理作業的任何應用程式命名空間。Trident Protect 不會為您建立這些命名空間；如果您在自訂資源中指定不存在的命名空間，則作業將會失敗。

NAS 經濟容量需求

Trident Protect 支援 NAS 經濟型磁碟區的備份與還原作業。目前不支援快照，複製和 SnapMirror 複寫至 NAS 經濟型磁碟區。您需要為打算搭配 Trident Protect 使用的每個 NAS 經濟型磁碟區啟用快照目錄。



某些應用程式與使用 Snapshot 目錄的磁碟區不相容。對於這些應用程式，您需要在 ONTAP 儲存系統上執行下列命令，以隱藏快照目錄：

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

您可以針對每個 NAS 經濟型磁碟區執行下列命令，以您要變更的磁碟區 UUID 取代，來啟用 Snapshot 目錄 <volume-UUID>：

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level  
=true -n trident
```



您可以將 Trident 後端組態選項設定為，為 true 新的磁碟區預設啟用快照目錄 `snapshotDir`。現有的磁碟區不受影響。

SnapMirror 複寫需求

NetApp SnapMirror 可與 Trident Protect 搭配使用，適用於下列 ONTAP 解決方案：

- NetApp ASA
- NetApp AFF
- NetApp FAS
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP

- Amazon FSX for NetApp ONTAP 產品

SnapMirror 複寫的 ONTAP 叢集需求

如果您打算使用 SnapMirror 複寫，請確保 ONTAP 叢集符合下列需求：

- * Astra 控制備份程式或 Trident *：Astra 控制備份程式或 Trident 必須同時存在於使用 ONTAP 作為後端的來源叢集和目的地 Kubernetes 叢集上。Trident Protect 使用下列驅動程式所支援的儲存類別，以 NetApp SnapMirror 技術支援複寫：
 - 「ONTAP-NAS」
 - 「ONTAP-SAN」
- * 授權 *：使用資料保護套件的 ONTAP SnapMirror 非同步授權必須同時在來源和目的地 ONTAP 叢集上啟用。如需詳細資訊、請參閱 "[SnapMirror授權概述ONTAP](#)"。

SnapMirror 複寫的對等考量

如果您計畫使用儲存後端對等，請確保您的環境符合下列需求：

- * 叢集與 SVM*：必須對 ONTAP 儲存設備的後端進行對等處理。如需詳細資訊、請參閱 "[叢集與SVM對等概觀](#)"。



確保兩個 ONTAP 叢集之間複寫關係中使用的 SVM 名稱是唯一的。

- **Astra** 控制資源配置程式或 **Trident** 和 **SVM**：對等的遠端 SVM 必須可用於目的地叢集上的 Astra 控制資源配置程式或 Trident。
- * 託管後端 *：您需要在 Trident Protect 中新增及管理 ONTAP 儲存後端，才能建立複寫關係。
- **NVMe over TCP**：Trident Protect 不支援 NetApp SnapMirror 複寫，用於使用 NVMe over TCP 傳輸協定的儲存後端。

用於 SnapMirror 複寫的 Trident / ONTAP 組態

Trident Protect 要求您至少設定一個儲存後端，以支援來源叢集和目的地叢集的複寫。如果來源叢集和目的地叢集相同、則目的地應用程式應使用不同於來源應用程式的儲存後端、以獲得最佳恢復能力。

使用 KubeVirt 時的考量

如果您打算使用 "[KubeVirt](#)" 虛擬機器進行 SnapMirror 複寫，則需要設定虛擬化，以便凍結和取消凍結 SVM。設定虛擬化之後，您部署的 SVM 將包含凍結和取消凍結所需的工具。若要深入瞭解設定虛擬化的相關資訊，請 "[安裝 OpenShift 虛擬化](#)" 參閱。

安裝及設定 Trident Protect

如果您的環境符合 Trident Protect 的要求，您可以依照下列步驟在叢集上安裝 Trident Protect。您可以從 NetApp 取得 Trident Protect，或從您自己的私有登錄安裝。如果您的叢集無法存取網際網路，從私有登錄安裝會很有幫助。



根據預設，Trident Protect 會收集支援資訊，協助處理您可能開啟的任何 NetApp 支援案例，包括叢集和託管應用程式的記錄，度量和拓撲資訊。Trident Protect 會根據每日排程將這些支援套裝組合傳送至 NetApp。您可以選擇在安裝 Trident Protect 時停用此支援套件集合。您可以隨時手動["產生支援服務組合"](#)進行。

安裝 Trident Protect from NetApp

1. 新增Trident Helm儲存庫：

```
helm repo add netapp-trident-protect
https://netapp.github.io/trident-protect-helm-chart
```

2. 安裝 Trident Protect 客戶需求日：

```
helm install trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.0 --create-namespace --namespace
trident-protect
```

3. 使用 Helm 以下列其中一個命令來安裝 Trident Protect 。以叢集名稱取代 <name_of_cluster>，該名稱將指派給叢集，用於識別叢集的備份和快照：

◦ 正常安裝 Trident Protect：

```
helm install trident-protect netapp-trident-protect/trident-
protect --set clusterName=<name_of_cluster> --version 100.2410.0
--create-namespace --namespace trident-protect
```

◦ 安裝 Trident Protect 並停用排定的每日 Trident Protect AutoSupport 支援服務套件上傳：

```
helm install trident-protect netapp-trident-protect/trident-
protect --set autoSupport.enabled=false --set
clusterName=<name_of_cluster> --version 100.2410.0 --create
--namespace --namespace trident-protect
```

4. 您也可以凍結 VM 。如果您使用 KubeVirt 支援 SnapMirror，則凍結虛擬機器可協助您有效管理：

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```



您必須設定虛擬化，凍結功能才能正常運作。在此設定之後部署的 VM 包括凍結和取消凍結所需的二進位檔。若要深入瞭解設定虛擬化的相關["安裝 OpenShift 虛擬化"](#)資訊，請參閱。

從私有登錄安裝 Trident Protect

如果 Kubernetes 叢集無法存取網際網路，您可以從私有映像登錄安裝 Trident Protect。在這些範例中，請將方括號中的值取代之為環境中的資訊：

1. 將下列影像拉到您的本機電腦，更新標記，然後將它們推送到您的私人登錄：

```
netapp/controller:24.10.0
netapp/restic:24.10.0
netapp/kopia:24.10.0
netapp/trident-autosupport:24.10.0
netapp/exechook:24.10.0
netapp/resourcebackup:24.10.0
netapp/resourcerestore:24.10.0
netapp/resourcedelete:24.10.0
bitnami/kubectl:1.30.2
kubebuilder/kube-rbac-proxy:v0.16.0
```

例如：

```
docker pull netapp/controller:24.10.0
```

```
docker tag netapp/controller:24.10.0 <private-registry-
url>/controller:24.10.0
```

```
docker push <private-registry-url>/controller:24.10.0
```

2. 建立 Trident Protect 系統命名空間：

```
kubectl create ns trident-protect
```

3. 登入登錄：

```
helm registry login <private-registry-url> -u <account-id> -p <api-
token>
```

4. 建立用於私人登錄驗證的拉出密碼：

```
kubectl create secret docker-registry regcred --docker
-username=<registry-username> --docker-password=<api-token> -n
trident-protect --docker-server=<private-registry-url>
```

5. 新增 Trident Helm 儲存庫：

```
helm repo add netapp-trident-protect
https://netapp.github.io/trident-protect-helm-chart
```

6. 建立名稱包含下列 Trident Protect 設定的檔案 `protectValues.yaml` :

```
image:
  registry: <private-registry-url>
imagePullSecrets:
- name: regcred
controller:
  image:
    registry: <private-registry-url>
rbacProxy:
  image:
    registry: <private-registry-url>
crCleanup:
  imagePullSecrets:
  - name: regcred
webhooksCleanup:
  imagePullSecrets:
  - name: regcred
```

7. 安裝 Trident Protect 客戶需求日 :

```
helm install trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.0 --create-namespace --namespace
trident-protect
```

8. 使用 Helm 以下列其中一個命令來安裝 Trident Protect 。以叢集名稱取代 `<name_of_cluster>` , 該名稱將指派給叢集, 用於識別叢集的備份和快照 :

- 正常安裝 Trident Protect :

```
helm install trident-protect netapp-trident-protect/trident-
protect --set clusterName=<name_of_cluster> --version 100.2410.0
--create-namespace --namespace trident-protect -f
protectValues.yaml
```

- 安裝 Trident Protect 並停用排定的每日 Trident Protect AutoSupport 支援服務套件上傳 :

```
helm install trident-protect netapp-trident-protect/trident-protect --set autoSupport.enabled=false --set clusterName=<name_of_cluster> --version 100.2410.0 --create --namespace --namespace trident-protect -f protectValues.yaml
```

9. 您也可以凍結 VM。如果您使用 KubeVirt 支援 SnapMirror，則凍結虛擬機器可協助您有效管理：

```
kubectl set env deployment/trident-protect-controller-manager NEPTUNE_VM_FREEZE=true -n trident-protect
```



您必須設定虛擬化，凍結功能才能正常運作。在此設定之後部署的 VM 包括凍結和取消凍結所需的二進位檔。若要深入瞭解設定虛擬化的相關["安裝 OpenShift 虛擬化"](#)資訊，請參閱。

安裝 Trident Protect CLI 外掛程式

您可以使用 Trident Protect 命令列外掛程式（Trident 公用程式的延伸 `tridentctl`）來建立自訂資源（CRS），並與 Trident 互動。

安裝 Trident Protect CLI 外掛程式

在使用命令列公用程式之前，您必須先將其安裝在用來存取叢集的機器上。根據您的機器使用的是 x64 或 ARM CPU，請遵循下列步驟。

下載適用於 **Linux AMD64 CPU** 的外掛程式

1. 下載 Trident Protect CLI 外掛程式：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-protect-linux-amd64
```

下載適用於 **Linux ARM64 CPU** 的外掛程式

1. 下載 Trident Protect CLI 外掛程式：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-protect-linux-arm64
```

下載適用於 **Mac AMD64 CPU** 的外掛程式

1. 下載 Trident Protect CLI 外掛程式：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-protect-macos-amd64
```

下載 **Mac ARM64 CPU** 的外掛程式

1. 下載 Trident Protect CLI 外掛程式：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-protect-macos-arm64
```

1. 啟用二進位檔案的執行權限：

```
chmod +x tridentctl-protect
```

2. 將外掛程式二進位檔複製到路徑變數中定義的位置。例如， `/usr/bin` 或 `/usr/local/bin`（您可能需要提升的 Privileges）：

```
cp ./tridentctl-protect /usr/local/bin/
```

3. 您也可以選擇將二進位檔案複製到主目錄中的某個位置。在這種情況下，您可能需要將位置新增至 PATH 變數：

```
cp ./tridentctl-protect ~/bin/
```

檢視 **Trident CLI** 外掛程式說明

您可以使用內建的外掛程式說明功能，取得外掛程式功能的詳細說明：

步驟

1. 使用說明功能檢視使用指南：

```
tridentctl protect help
```

啟用命令自動完成

安裝 Trident Protect CLI 外掛程式之後，您可以啟用某些命令的自動完成功能。

啟用 **Bash Shell** 的自動完成功能

1. 下載完成指令碼：

```
curl -L -O https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-completion.bash
```

2. 在主目錄中建立新目錄以包含指令碼：

```
mkdir -p ~/.bash/completions
```

3. 將下載的指令碼移至 `~/.bash/completions` 目錄：

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. 將下列行新增至 `~/.bashrc` 主目錄中的檔案：

```
source ~/.bash/completions/tridentctl-completion.bash
```

啟用 **Z Shell** 的自動完成功能

1. 下載完成指令碼：

```
curl -L -O https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-completion.zsh
```

2. 在主目錄中建立新目錄以包含指令碼：

```
mkdir -p ~/.zsh/completions
```

3. 將下載的指令碼移至 `~/.zsh/completions` 目錄：

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. 將下列行新增至 `~/.zprofile` 主目錄中的檔案：

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

結果

下次登入 Shell 時，您可以使用命令自動完成與 `tridentctl Protect` 外掛程式。

管理 Trident Protect

管理授權與存取控制

Trident Protect 採用 Kubernetes 角色型存取控制（RBAC）模式。根據預設，Trident Protect 提供單一系統命名空間及其相關的預設服務帳戶。如果組織有許多使用者或特定的安全需求，您可以使用 Trident Protect 的 RBAC 功能，更精細地控制對資源和命名空間的存取。

叢集管理員一律可以存取預設命名空間中的資源 `trident-protect`，也可以存取所有其他命名空間中的資源。若要控制對資源和應用程式的存取，您需要建立額外的命名空間，並將資源和應用程式新增至這些命名空間。

請注意，沒有使用者可以在預設命名空間中建立應用程式資料管理 CRS `trident-protect`。您需要在應用程式命名空間中建立應用程式資料管理 CRS（最佳做法是在與其相關應用程式相同的命名空間中建立應用程式資料管理 CRS）。

只有系統管理員才能存取授權的 Trident 保護自訂資源物件，包括：



- `* AppVault*`：需要儲存庫認證資料
- `* AutoSupportBundle *`：收集指標，記錄及其他敏感的 Trident Protect 資料
- `* AutoSupportBundleSchedule*`：管理記錄收集排程

最佳做法是使用 RBAC 來限制系統管理員存取權限物件。

如需 RBAC 如何規範資源和命名空間存取的詳細資訊，請參閱 "[Kubernetes RBAC 文件](#)"。

如需服務帳戶的相關資訊，請參閱 "[Kubernetes 服務帳戶文件](#)"。

範例：管理兩組使用者的存取權

例如，組織有叢集管理員，一組工程設計使用者，以及一組行銷使用者。叢集管理員將完成下列工作，以建立一個環境，其中工程群組和行銷群組各自只能存取指派給各自命名空間的資源。

步驟 1：建立命名空間以包含每個群組的資源

建立命名空間可讓您以邏輯方式分隔資源，並更有效地控制誰有權存取這些資源。

步驟

1. 為工程群組建立命名空間：

```
kubectl create ns engineering-ns
```

2. 為行銷群組建立命名空間：

```
kubectl create ns marketing-ns
```

步驟 2： 建立新的服務帳戶，與每個命名空間中的資源互動

您所建立的每個新命名空間都有預設服務帳戶，但您應該為每個使用者群組建立服務帳戶，以便日後在必要時在群組之間進一步分割 Privileges。

步驟

1. 為工程群組建立服務帳戶：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. 為行銷群組建立服務帳戶：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

步驟 3： 為每個新的服務帳戶建立秘密

服務帳戶密碼是用來驗證服務帳戶，如果受到入侵，也可以輕鬆刪除和重新建立。

步驟

1. 為工程服務帳戶建立秘密：

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
type: kubernetes.io/service-account-token
```

2. 為行銷服務帳戶建立秘密：

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
  type: kubernetes.io/service-account-token
```

步驟 4：建立 **RoleBinding** 物件，將 **ClusterRole** 物件繫結至每個新的服務帳戶

安裝 Trident Protect 時會建立預設的 ClusterRole 物件。您可以建立並套用角色繫結物件，將此 ClusterRole 繫結至服務帳戶。

步驟

1. 將 ClusterRole 繫結至工程服務帳戶：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

2. 將 ClusterRole 連結至行銷服務帳戶：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

步驟 5：測試權限

測試權限是否正確。

步驟

1. 確認工程使用者可以存取工程資源：

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. 確認工程使用者無法存取行銷資源：

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

步驟 6：授予對 AppVault 物件的存取權

若要執行資料管理工作，例如備份和快照，叢集管理員必須將 AppVault 物件的存取權授予個別使用者。

步驟

1. 建立並套用 AppVault 和加密組合 YAML 檔案，以授予使用者存取 AppVault 的權限。例如，下列 CR 將 AppVault 的存取權授予使用者 eng-user：

```

apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3

```

2. 建立並套用角色 CR，讓叢集管理員能夠授與對命名空間中特定資源的存取權。例如：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. 建立並套用 RoleBinding CR，將權限繫結至使用者 eng-user。例如：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

4. 確認權限正確。

- a. 嘗試擷取所有命名空間的 AppVault 物件資訊：

```
kubectl get appvaults -n trident-protect
--as=system:serviceaccount:engineering-ns:eng-user
```

您應該會看到類似下列的輸出：

```
Error from server (Forbidden): appvaults.protect.trident.netapp.io is forbidden: User "system:serviceaccount:engineering-ns:eng-user" cannot list resource "appvaults" in API group "protect.trident.netapp.io" in the namespace "trident-protect"
```

- b. 測試以查看使用者是否能取得他們現在有權存取的 AppVault 資訊：

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n trident-protect
```

您應該會看到類似下列的輸出：

```
yes
```

結果

您已授予 AppVault 權限的使用者應該能夠使用授權的 AppVault 物件來執行應用程式資料管理作業，而且不應能夠存取指派命名空間以外的任何資源，或建立他們無法存取的新資源。

產生支援服務組合

Trident Protect 可讓系統管理員產生套件組合，其中包含 NetApp 支援所需的資訊，包括所管理叢集和應用程式的記錄，度量和拓撲資訊。如果您已連線至網際網路，則可以使用自訂資源（CR）檔案，將支援套件上傳至 NetApp 支援網站（NSS）。

使用 CR 建立支援服務組合

1. 建立自訂資源（CR）檔案並命名（例如 `trident-protect-support-bundle.yaml`）。
2. 設定下列屬性：
 - `* metadata.name*`:（`_required`）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
 - `spec.triggerType` :（`_required_`）決定是立即產生支援套件，還是排程產生。排定的套件產生時間為上午 12 點，UTC。可能值：
 - 已排程
 - 手冊
 - `SPEC.uploadEnabled` :（`Optional`）控制是否應在支援服務組合產生後，將其上傳至 NetApp 支援網站。如果未指定，則默認為 `false`。可能值：
 - 是的
 - 否（預設）
 - `spec.daWindowStart` :（`Optional`）RFC 3339 格式的日期字串，指定支援套件中所包含資料的視窗應開始的日期與時間。如果未指定，則預設為 24 小時前。您可以指定的最早時間是 7 天前。

YAML 範例：

```
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. 在您以正確的值填入檔案之後 `astra-support-bundle.yaml`、請套用 CR：

```
kubectl apply -f trident-protect-support-bundle.yaml
```

使用 CLI 建立支援服務包

1. 建立支援服務組合，以環境資訊取代括號中的值。 `trigger-type` 決定套件是立即建立，還是建立時間取決於排程，可以是 `Manual` 或 `Scheduled`。預設設定為 `Manual`。

例如：

```
tridentctl protect create autosupportbundle <my_bundle_name>
--trigger-type <trigger_type>
```

管理及保護應用程式

使用 AppVault 物件來管理貯體

Trident Protect 的貯體自訂資源（CR）稱為 AppVault。AppVault 物件是儲存貯體的宣告性 Kubernetes 工作流程表示。AppVault CR 包含用於保護作業（例如備份，快照，還原作業和 SnapMirror 複寫）的儲存庫所需的組態。只有管理員可以建立 AppVaults。

金鑰產生與 AppVault 定義範例

定義 AppVault CR 時，您需要加入認證，才能存取供應商託管的資源。您產生認證金鑰的方式會因供應商而異。以下是數個提供者的命令列金鑰產生範例，接著是每個提供者的 AppVault 定義範例。

Google Cloud

金鑰產生範例：

```
kubectl create secret generic gcp-creds --from-file=credentials=<mycreds  
-file.json> -n trident-protect
```

以下 AppVault 定義範例是以 CR 的形式提供，您可以使用及修改，或是以 Trident Protect CLI 命令的範例來為您產生 AppVault CR：

範例 AppVault CR

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

使用 Trident Protect CLI 建立 AppVault CR 範例

```
tridentctl protect create vault gcp my-new-vault --bucket mybucket
--project my-gcp-project --secret <gcp-creds>/<credentials>
```

Amazon S3

金鑰產生範例：

```
kubectl create secret generic -n trident-protect s3 --from
-literal=accessKeyID=<secret-name> --from-literal=secretAccessKey
=<generic-s3-trident-protect-src-bucket-secret>
```

以下 AppVault 定義範例是以 CR 的形式提供，您可以使用及修改，或是以 Trident Protect CLI 命令的範例來為您產生 AppVault CR：

範例 AppVault CR

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

使用 CLI 建立 AppVault 範例

```
tridentctl protect create vault GenericS3 s3vault --bucket <bucket-
name> --secret <secret-name> --endpoint <s3-endpoint>
```

Microsoft Azure

金鑰產生範例：

```
kubectl create secret generic <secret-name> --from-literal=accountKey
=<secret-name> -n trident-protect
```

以下 AppVault 定義範例是以 CR 的形式提供，您可以使用及修改，或是以 Trident Protect CLI 命令的範例來為您產生 AppVault CR：

範例 AppVault CR

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

使用 CLI 建立 AppVault 範例

```
tridentctl protect create vault Azure <vault-name> --account <account-
name> --bucket <bucket-name> --secret <secret-name>
```

提供者類型和供應商組態的支援值

`providerType` AppVault CR 中的和
`providerConfig` 金鑰需要特定值。下表列出金鑰支援的值
`providerType`，以及每個值所需使用的相關 `providerConfig` 金鑰 `providerType`。

支援 `providerType` 的值	`providerConfig` 關聯金鑰
AWS	s3
Azure	Azure
GCP	GCP
GenericS3	s3
OntapS3	s3
StorageGridS3	s3

使用 AppVault 瀏覽器檢視 AppVault 資訊

您可以使用 Trident Protect CLI 外掛程式來檢視已在叢集上建立的 AppVault 物件相關資訊。

步驟

1. 檢視 AppVault 物件的內容：

```
tridentctl protect get appvaultcontent gcp-vault --show-resources all
```

◦ 輸出範例 *：

```
+-----+-----+-----+-----+
+-----+
| CLUSTER | APP | TYPE | NAME |
| TIMESTAMP | | | |
+-----+-----+-----+-----+
+-----+
| | mysql | snapshot | mysnap | 2024-
08-09 21:02:11 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815180300 | 2024-
08-15 18:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815200300 | 2024-
08-15 20:03:06 (UTC) |
| production1 | mysql | backup | hourly-e7db6-20240815180300 | 2024-
08-15 18:04:25 (UTC) |
| production1 | mysql | backup | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:30 (UTC) |
| production1 | mysql | backup | hourly-e7db6-20240815200300 | 2024-
08-15 20:04:21 (UTC) |
| production1 | mysql | backup | mybackup5 | 2024-
08-09 22:25:13 (UTC) |
| | mysql | backup | mybackup | 2024-
08-09 21:02:52 (UTC) |
+-----+-----+-----+-----+
+-----+
```

2. (可選) 要查看每個資源的 AppVaultPath，請使用標誌 --show-paths。

只有在 Trident Protect helm 安裝中指定叢集名稱時，表格第一欄中的叢集名稱才能使用。例如 --set clusterName=production1：。

移除 AppVault

您可以隨時移除 AppVault 物件。



刪除 AppVault 物件之前，請勿移除 `finalizers` AppVault CR 中的機碼。如果您這麼做，可能會導致 AppVault 貯體中的剩餘資料，以及叢集中的孤立資源。

開始之前

確保您已刪除儲存在相關儲存庫中的所有快照和備份。

使用 **Kubernetes CLI** 移除 **AppVault**

1. 移除 AppVault 物件，以要移除的 AppVault 物件名稱取代 `appvault_name`：

```
kubectl delete appvault <appvault_name> -n trident-protect
```

使用 **Trident CLI** 移除 **AppVault**

1. 移除 AppVault 物件，以要移除的 AppVault 物件名稱取代 `appvault_name`：

```
tridentctl protect delete appvault <appvault_name> -n trident-protect
```

定義管理應用程式

您可以建立應用程式 CR 和相關的 AppVault CR，以定義您想要使用 Trident Protect 管理的應用程式。

建立 **AppVault CR**

您需要建立 AppVault CR，以便在應用程式上執行資料保護作業時使用，而 AppVault CR 必須位於安裝 Trident Protect 的叢集上。AppVault CR 專屬於您的環境，如需 AppVault CRS 的範例，請參閱"[AppVault 自訂資源](#)。"

建立應用程式 **CR**

您需要為每個想要使用 Trident Protect 管理的應用程式建立應用程式 CR。您可以手動建立應用程式 CR，或使用 Trident Protect CLI 建立 CR 來新增應用程式以進行管理。

使用 CR 新增應用程式

1. 建立目的地應用程式 CR 檔案：

- a. 建立自訂資源（CR）檔案並命名（例如 `maria-app.yaml`）。
- b. 設定下列屬性：
 - `* metadata.name*`:（`_required`）應用程式自訂資源的名稱。請注意您選擇的名稱，因為保護作業所需的其他 CR 檔案都會參照此值。
 - `* spec.includedNamespaces*`:（`_required_`）使用命名空間標籤或命名空間名稱來指定應用程式資源所在的命名空間。應用程式命名空間必須是此清單的一部分。

YAML 範例：

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: maria
  namespace: my-app-namespace
spec:
  includedNamespaces:
    labelSelector: {}
    namespace: my-app-namespace
```

使用 CLI 新增應用程式

1. 建立並套用應用程式定義，以環境資訊取代方括號中的值。您可以使用以逗號分隔的清單，將命名空間和資源包含在應用程式定義中，並附上下列範例所示的引數：

```
tridentctl protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include>
```

保護應用程式

使用自動保護原則或以臨機操作的方式、擷取快照與備份資料、以保護所有應用程式。

建立隨需快照

您可以隨時建立隨需快照。

使用 CR 建立快照

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-snapshot-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
 - `* metadata.name*`: (`_required`) 此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
 - **SPEC.applicationRef** : 要快照的應用程式的 Kubernetes 名稱。
 - **spec.appVaultRef** : (`_required_`) 應儲存快照內容（中繼資料）的 AppVault 名稱。
 - **spec.reclaimersPolicy** : (*Optional*) 定義刪除快照 CR 時，應用程式歸檔會發生什麼情況。這表示即使設定為，快照也 `Retain` 會被刪除。有效選項：
 - Retain (預設)
 - Delete

```
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  reclaimPolicy: Delete
```

3. 在您以正確的值填入檔案之後 `trident-protect-snapshot-cr.yaml`、請套用 CR :

```
kubectl apply -f trident-protect-snapshot-cr.yaml
```

使用 CLI 建立快照

1. 建立快照，以您環境的資訊取代方括號中的值。例如：

```
tridentctl protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot>
```

建立隨選備份

您可以隨時備份應用程式。

使用 CR 建立備份

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-backup-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
 - `* metadata.name*:`（*_required*）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
 - **SPEC.applicationRef**：（*_required*）要備份的應用程式 Kubernetes 名稱。
 - **spec.appVaultRef**：（*_required*）應儲存備份內容的 AppVault 名稱。
 - `*spec.dataMover*`：（*Optional*）字串，指出備份作業所使用的備份工具。可能的值（區分大小寫）：
 - Restic
 - Kopia（預設）
 - **spec.reClaimPolicy**：（*Optional*）定義備份從宣告中釋出時會發生什麼情況。可能值：
 - Delete
 - Retain（預設）
 - **Spec.snapshotRef**：（*Optional*）：用於備份來源的快照名稱。如果未提供，將會建立並備份暫存快照。

```
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. 在您以正確的值填入檔案之後 `trident-protect-backup-cr.yaml`、請套用 CR：

```
kubectl apply -f trident-protect-backup-cr.yaml
```

使用 CLI 建立備份

1. 建立備份，以您環境的資訊取代括號中的值。例如：

```
tridentctl protect create backup <my_backup_name> --appvault <my-  
vault-name> --app <name_of_app_to_back_up>
```

建立資料保護排程

保護原則可在已定義的排程中建立快照、備份或兩者、以保護應用程式。您可以選擇每小時、每天、每週和每月建立快照和備份、也可以指定要保留的複本數量。

使用 CR 建立排程

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-schedule-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
 - `* metadata.name*:`（`_required`）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
 - `*spec.dataMover*`：（*Optional*）字串，指出備份作業所使用的備份工具。可能的值（區分大小寫）：
 - `Restic`
 - `Kopia`（預設）
 - **SPEC.applicationRef**：要備份之應用程式的 Kubernetes 名稱。
 - **spec.appVaultRef**：（`_required`）應儲存備份內容的 AppVault 名稱。
 - `*SPEC.BackupRetention*`：要保留的備份數量。零表示不應建立備份。
 - `*spec.snapshotRetention*`：要保留的快照數。零表示不應建立任何快照。
 - `* spec.granularity*:` 執行排程的頻率。可能的值、以及必要的相關欄位：
 - `hourly`（要求您指定 `spec.minute`）
 - `daily`（要求您指定 `spec.minute` 和 `spec.hour`）
 - `weekly`（要求您指定 `spec.minute`，`spec.hour`，和 `spec.dayOfWeek`）
 - `monthly`（要求您指定 `spec.minute`，`spec.hour`，和 `spec.dayOfMonth`）
 - **spec.dayOfMonth**：（*Optional*）排程應執行的月份日期（1 - 31）。如果精細度設為、則此欄位為必 `monthly` 填。
 - `*spec.dayOfWeek*`：（`_Optional`）排程應執行的一週中的一天（0 - 7）。0 或 7 的值表示星期日。如果精細度設為、則此欄位為必 `weekly` 填。
 - `*spec.hour*`：（`_Optional`）排程應執行的一天中的小時（0 - 23）。如果精細度設置為、或，則此字段為必填字段 `daily weekly monthly`。
 - `* 規格分鐘 *`：（`_選用`）排程應執行的小時（0 - 59）分鐘。如果精細度設置為、或，則此字段為必填字段 `hourly daily weekly monthly`。

```
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: <monthly>
  dayOfMonth: "1"
  dayOfWeek: "0"
  hour: "0"
  minute: "0"
```

3. 在您以正確的值填入檔案之後 trident-protect-schedule-cr.yaml 、請套用 CR ：

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

使用 CLI 建立排程

1. 建立保護排程，以環境資訊取代方括號中的值。例如：



您可以使用 `tridentctl protect create schedule --help` 來檢視此命令的詳細說明資訊。

```
tridentctl protect create schedule <my_schedule_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> --backup
-retention <how_many_backups_to_retain> --data-mover
<kopia_or_restic> --day-of-month <day_of_month_to_run_schedule>
--day-of-week <day_of_month_to_run_schedule> --granularity
<frequency_to_run> --hour <hour_of_day_to_run> --minute
<minute_of_hour_to_run> --recurrence-rule <recurrence> --snapshot
-retention <how_many_snapshots_to_retain>
```

刪除快照

刪除不再需要的排程或隨需快照。

步驟

1. 移除與快照相關的 Snapshot CR ：

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

刪除備份

刪除不再需要的排程或隨需備份。

步驟

1. 移除與備份相關的備份 CR：

```
kubectl delete backup <backup_name> -n my-app-namespace
```

檢查備份作業的狀態

您可以使用命令列來檢查正在進行，已完成或已失敗的備份作業狀態。

步驟

1. 使用下列命令可擷取備份作業的狀態，以環境中的資訊取代方括號中的值：

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath  
='{.status}'
```

啟用 NetApp 檔案（anf）作業的備份與還原

如果您已安裝 Trident Protect，您可以啟用節省空間的備份與還原功能，以供使用 NetApp 檔案儲存類別的儲存後端使用，並在 Trident 24.06 之前建立。此功能可與 NFSv4 磁碟區搭配使用，不會佔用容量集區的額外空間。

開始之前

請確認下列事項：

- 您已安裝 Trident Protect。
- 您已在 Trident Protect 中定義應用程式。在您完成此程序之前、此應用程式的保護功能有限。
- 您已 `azure-netapp-files` 選擇儲存後端的預設儲存類別。

1. 如果 anfv Volume 是在升級至 Trident 24.10 之前建立的，請在 Trident 中執行下列動作：

a. 針對每個以 NetApp 檔案為基礎且與應用程式相關的 PV，啟用快照目錄：

```
tridentctl update volume <pv name> --snapshot-dir=true -n trident
```

b. 確認已為每個相關的 PV 啟用快照目錄：

```
tridentctl get volume <pv name> -n trident -o yaml | grep  
snapshotDir
```

回應：

```
snapshotDirectory: "true"
```

+

未啟用 Snapshot 目錄時，Trident Protect 會選擇一般備份功能，在備份程序期間會暫時佔用容量集區中的空間。在這種情況下，請確保容量集區中有足夠的可用空間，以建立備份磁碟區大小的暫存磁碟區。

結果

應用程式已準備好使用 Trident Protect 進行備份與還原。每個 PVC 也可供其他應用程式用於備份和還原。

還原應用程式

您可以使用 Trident Protect 從快照或備份還原應用程式。將應用程式還原至同一個叢集時、從現有的快照還原速度會更快。



當您還原應用程式時，為應用程式設定的所有執行掛鉤都會隨應用程式一起還原。如果存在還原後執行掛鉤，則會在還原作業中自動執行。

從備份還原至不同的命名空間

當您使用備份還原 CR 將備份還原至不同的命名空間時，Trident Protect 會在新的命名空間中還原應用程式，但還原的應用程式不會自動受到 Trident Protect 的保護。為了保護還原的應用程式，您需要為還原的應用程式建立應用程式 CR，以便受到 Trident Protect 保護。



將備份還原至具有現有資源的不同命名空間，並不會改變任何與備份中共用名稱的資源。若要還原備份中的所有資源，請刪除並重新建立目標命名空間，或將備份還原至新的命名空間。

使用 CR

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-backup-restore-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
 - `* metadata.name*`:（`_required`）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
 - `spec.appArchivePath`：儲存備份內容的 AppVault 內部路徑。您可以使用下列命令來尋找此路徑：

```
kubectl get backups <BACKUP_NAME> -n my-app-namespace -o  
jsonpath='{.status.appArchivePath}'
```

- `spec.appVaultRef`：（`_required`）儲存備份內容的 AppVault 名稱。
- `spec.namespaceMapping`: 將還原作業的來源命名空間對應至目的地命名空間。以環境中的資訊取代 `my-source-namespace` 和 `my-destination-namespace`。
- `spec.storageClassMapping`：將還原作業的來源儲存類別對應至目的地儲存類別。以環境中的資訊取代 `destinationStorageClass` 和 `sourceStorageClass`。

```
apiVersion: protect.trident.netapp.io/v1o  
kind: BackupRestore  
metadata:  
  name: my-cr-name  
  namespace: my-app-namespace  
spec:  
  appArchivePath: my-backup-path  
  appVaultRef: appvault-name  
  namespaceMapping: [{"source": "my-source-namespace",  
"destination": "my-destination-namespace"}]  
  storageClassMapping:  
    destination: "${destinationStorageClass}"  
    source: "${sourceStorageClass}"
```

- 3.（*Optional*）如果您只需要選取應用程式的某些資源來還原，請新增篩選功能，以包含或排除標記有特定標籤的資源：
 - `* resourceFilter.resourceSelectionCriteria`：（篩選所需）用於 `include or exclude` 包含或排除在 `resourceMatchers` 中定義的資源。新增下列資源配置工具參數、以定義要納入或排除的資源：
 - `resourceFilter.resourceMatchers`：resourceMatcher 物件陣列。
 - `resourceMatchers[].group`：（*Optional*）要篩選的資源群組。
 - `resourceMatchers[].cher`：（*Optional*）要篩選的資源種類。
 - `resourceMatchers[].version`：（*Optional*）要篩選的資源版本。
 - 要篩選之資源的 Kubernetes `metadata.name` 欄位中的 `* resourceMatchers[].names*`：（*Optional*）名稱。

- 要篩選之資源的 Kubernetes metadata.name 欄位中的 *resourceMatchers[].names* : (*Optional*) 命名空間。
- 資源的 Kubernetes metadata.name 欄位中的 *resourceMatchers[].labelSelectors* : (*Optional*) Label 選取器字串，如中所定義 "Kubernetes文件"。例如 "trident.netapp.io/os=linux" :。

例如：

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      group: my-resource-group
      kind: my-resource-kind
      version: my-resource-version
      names: ["my-resource-names"]
      namespaces: ["my-resource-namespaces"]
      labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 在您以正確的值填入檔案之後 trident-protect-backup-restore-cr.yaml 、請套用 CR :

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

使用CLI

1. 將備份還原至不同的命名空間，以環境中的資訊取代括弧中的值。此 namespace-mapping` 引數使用以冒號分隔的命名空間，以格式將來源命名空間對應至正確的目的地命名空間 `source1:dest1, source2:dest2`。例如：

```
tridentctl protect create backuprestore <my_restore_name> --backup
<backup_namespace>/<backup_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

從備份還原至原始命名空間

您可以隨時將備份還原至原始命名空間。

使用 CR

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-backup-ipr-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
 - `* metadata.name*`：（`_required`）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
 - `spec.appArchivePath`：儲存備份內容的 AppVault 內部路徑。您可以使用下列命令來尋找此路徑：

```
kubectl get backups <BACKUP_NAME> -n my-app-namespace -o  
jsonpath='{.status.appArchivePath}'
```

- `spec.appVaultRef`：（`_required`）儲存備份內容的 AppVault 名稱。

例如：

```
apiVersion: protect.trident.netapp.io/v1  
kind: BackupInplaceRestore  
metadata:  
  name: my-cr-name  
  namespace: my-app-namespace  
spec:  
  appArchivePath: my-backup-path  
  appVaultRef: appvault-name
```

3. （*Optional*）如果您只需要選取應用程式的某些資源來還原，請新增篩選功能，以包含或排除標記有特定標籤的資源：
 - `*resourceFilter.resourceSelectionCriteria`：（篩選所需）用於 `'include or exclude'` 包含或排除在 `resourceMatchers` 中定義的資源。新增下列資源配置工具參數、以定義要納入或排除的資源：
 - `resourceFilter.resourceMatchers`：resourceMatcher 物件陣列。
 - `resourceMatchers[].group`：（*Optional*）要篩選的資源群組。
 - `resourceMatchers[].cher`：（*Optional*）要篩選的資源種類。
 - `resourceMatchers[].version`：（*Optional*）要篩選的資源版本。
 - 要篩選之資源的 Kubernetes metadata.name 欄位中的 `* resourceMatchers[].names*`：（*Optional*）名稱。
 - 要篩選之資源的 Kubernetes metadata.name 欄位中的 `* resourceMatchers[].names*`：（*Optional*）命名空間。
 - 資源的 Kubernetes metadata.name 欄位中的 `*resourceMatchers[].labelSelectors *`：（*Optional*）Label 選取器字串，如中所定義 "[Kubernetes文件](#)"。例如 `"trident.netapp.io/os=linux"`：◦

例如：

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      group: my-resource-group
      kind: my-resource-kind
      version: my-resource-version
      names: ["my-resource-names"]
      namespaces: ["my-resource-namespaces"]
      labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 在您以正確的值填入檔案之後 `trident-protect-backup-ipr-cr.yaml`、請套用 CR：

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

使用CLI

1. 將備份還原至原始命名空間，以環境中的資訊取代括弧中的值。 `backup`` 引數使用的名稱空間和備份名稱格式為 `<namespace>/<name>`。例如：

```
tridentctl protect create backupinplacerestore <my_restore_name>
--backup <namespace/backup_to_restore>
```

從快照還原至不同的命名空間

您可以使用自訂資源（CR）檔案、將資料從快照還原至不同的命名空間或原始來源命名空間。當您使用 `SnapshotRestore` CR 將快照還原至不同的命名空間時，Trident Protect 會在新的命名空間中還原應用程式，但還原的應用程式並不會受到 Trident Protect 的自動保護。為了保護還原的應用程式，您需要為還原的應用程式建立應用程式 CR，以便受到 Trident Protect 保護。

使用 CR

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-snapshot-restore-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
 - `* metadata.name*`:（`_required`）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
 - `spec.appVaultRef` :（`_required`）儲存快照內容的 AppVault 名稱。
 - `spec.appArchivePath` : 在 AppVault 中儲存快照內容的路徑。您可以使用下列命令來尋找此路徑：

```
kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o  
jsonpath='{.status.appArchivePath}'
```

- `spec.namespaceMapping`: 將還原作業的來源命名空間對應至目的地命名空間。以環境中的資訊取代 `my-source-namespace` 和 `my-destination-namespace`。
- `spec.storageClassMapping` : 將還原作業的來源儲存類別對應至目的地儲存類別。以環境中的資訊取代 `destinationStorageClass` 和 `sourceStorageClass`。

```
apiVersion: protect.trident.netapp.io/v1  
kind: SnapshotRestore  
metadata:  
  name: my-cr-name  
  namespace: my-app-namespace  
spec:  
  appVaultRef: appvault-name  
  appArchivePath: my-snapshot-path  
  namespaceMapping: [{"source": "my-source-namespace",  
"destination": "my-destination-namespace"}]  
  storageClassMapping:  
    destination: "${destinationStorageClass}"  
    source: "${sourceStorageClass}"
```

- 3.（*Optional*）如果您只需要選取應用程式的某些資源來還原，請新增篩選功能，以包含或排除標記有特定標籤的資源：
 - `* resourceFilter.resourceSelectionCriteria` :（篩選所需）用於 `include or exclude` 包含或排除在 `resourceMatchers` 中定義的資源。新增下列資源配置工具參數、以定義要納入或排除的資源：
 - `resourceFilter.resourceMatchers` : `resourceMatcher` 物件陣列。
 - `resourceMatchers[].group` :（*Optional*）要篩選的資源群組。
 - `resourceMatchers[].cher` :（*Optional*）要篩選的資源種類。
 - `resourceMatchers[].version` :（*Optional*）要篩選的資源版本。
 - 要篩選之資源的 Kubernetes `metadata.name` 欄位中的 `* resourceMatchers[].names*` :（*Optional*）名稱。

- 要篩選之資源的 Kubernetes metadata.name 欄位中的 *resourceMatchers[].names* : (*Optional*) 命名空間。
- 資源的 Kubernetes metadata.name 欄位中的 *resourceMatchers[].labelSelectors* : (*Optional*) Label 選取器字串，如中所定義 "Kubernetes文件"。例如 "trident.netapp.io/os=linux" :。

例如：

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      group: my-resource-group
      kind: my-resource-kind
      version: my-resource-version
      names: ["my-resource-names"]
      namespaces: ["my-resource-namespaces"]
      labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 在您以正確的值填入檔案之後 trident-protect-snapshot-restore-cr.yaml、請套用 CR：

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

使用CLI

1. 將快照還原至不同的命名空間，以環境中的資訊取代方括號中的值。
 - snapshot 引數使用格式的命名空間和快照名稱 `<namespace>/<name>`。
 - 此 namespace-mapping 引數使用以冒號分隔的命名空間，以格式將來源命名空間對應至正確的目的地命名空間 `source1:dest1, source2:dest2`。

例如：

```
tridentctl protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

從快照還原至原始命名空間

您可以隨時將快照還原至原始命名空間。

使用 CR

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-snapshot-ipr-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
 - `* metadata.name*`:（`_required`）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
 - `spec.appVaultRef` :（`_required`）儲存快照內容的 AppVault 名稱。
 - `spec.appArchivePath` : 在 AppVault 中儲存快照內容的路徑。您可以使用下列命令來尋找此路徑：

```
kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o  
jsonpath='{.status.appArchivePath}'
```

```
apiVersion: protect.trident.netapp.io/v1  
kind: SnapshotInplaceRestore  
metadata:  
  name: my-cr-name  
  namespace: my-app-namespace  
spec:  
  appVaultRef: appvault-name  
  appArchivePath: my-snapshot-path
```

- 3.（*Optional*）如果您只需要選取應用程式的某些資源來還原，請新增篩選功能，以包含或排除標記有特定標籤的資源：
 - `* resourceFilter.resourceSelectionCriteria` :（篩選所需）用於 `include or exclude` 包含或排除在 `resourceMatchers` 中定義的資源。新增下列資源配置工具參數、以定義要納入或排除的資源：
 - `resourceFilter.resourceMatchers` : `resourceMatcher` 物件陣列。
 - `resourceMatchers[].group` :（*Optional*）要篩選的資源群組。
 - `resourceMatchers[].cher` :（*Optional*）要篩選的資源種類。
 - `resourceMatchers[].version` :（*Optional*）要篩選的資源版本。
 - 要篩選之資源的 Kubernetes `metadata.name` 欄位中的 `* resourceMatchers[].names*` :（*Optional*）名稱。
 - 要篩選之資源的 Kubernetes `metadata.name` 欄位中的 `* resourceMatchers[].names*` :（*Optional*）命名空間。
 - 資源的 Kubernetes `metadata.name` 欄位中的 `*resourceMatchers[].labelSelectors*` :（*Optional*）Label 選取器字串，如中所定義 "[Kubernetes文件](#)"。例如 `"trident.netapp.io/os=linux"` :。

例如：

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      group: my-resource-group
      kind: my-resource-kind
      version: my-resource-version
      names: ["my-resource-names"]
      namespaces: ["my-resource-namespaces"]
      labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 在您以正確的值填入檔案之後 trident-protect-snapshot-ipr-cr.yaml 、請套用 CR ：

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

使用CLI

1. 將快照還原至原始命名空間，以環境中的資訊取代方括號中的值。例如：

```
tridentctl protect create snapshotinplacerestore <my_restore_name>
--snapshot <snapshot_to_restore>
```

檢查還原作業的狀態

您可以使用命令列來檢查進行中，已完成或已失敗的還原作業狀態。

步驟

1. 使用下列命令可擷取還原作業的狀態，以環境中的資訊取代方括號中的值：

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

使用 NetApp SnapMirror 複寫應用程式

使用 Trident Protect，您可以使用 NetApp SnapMirror 技術的非同步複寫功能，將資料和應用程式變更從一個儲存後端複寫到另一個儲存後端，在同一個叢集或不同叢集之間複寫。

設定複寫關係

設定複寫關係涉及下列事項：

- 選擇 Trident Protect 拍攝應用程式快照的頻率（包括應用程式的 Kubernetes 資源，以及每個應用程式磁碟區的磁碟區快照）
- 選擇複寫排程（包括 Kubernetes 資源及持續磁碟區資料）
- 設定拍攝快照的時間

步驟

1. 為來源叢集上的來源應用程式建立 AppVault。視您的儲存供應商而定，請修改中的範例"[AppVault 自訂資源](#)"以符合您的環境：

使用 CR 建立 AppVault

- a. 建立自訂資源 (CR) 檔案並命名 (例如 `trident-protect-appvault-primary-source.yaml`) 。
- b. 設定下列屬性：
 - `* metadata.name*`: (`_required`) AppVault 自訂資源的名稱。請記下您選擇的名稱，因為複寫關係所需的其他 CR 檔案會參照此值。
 - `* spec.providerConfig*`: (`_required`) 儲存使用指定供應商存取 AppVault 所需的組態。請為您的供應商選擇一個「鎖釦名稱」和任何其他必要的詳細資料。請記下您選擇的值，因為複寫關係所需的其他 CR 檔案會參照這些值。如需 AppVault CRS 與其他供應商的範例，請參閱"[AppVault 自訂資源](#)"。
 - `* spec.providerCredentials*`: (`_required`) 會儲存使用指定提供者存取 AppVault 所需之任何認證的參考資料。
 - `* spec.providerCredentials.valueFromSecret*`: (`_required`) 表示認證值應來自機密。
 - `key` : (`_required`) 要從中選擇的密碼的有效金鑰。
 - `* 名稱 *` : (`_必要`) 包含此欄位值的機密名稱。必須位於相同的命名空間中。
 - `* spec.providerCredentials.secretAccessKey*`: (`_required`) 存取提供者所用的存取金鑰。`* 名稱 *` 應與 `* spec.providerCredentials.valueFromSecret.name*` 相符。
 - `* spec.providerType*`: (`_required`) 決定提供備份的內容，例如 NetApp ONTAP S3 ，一般 S3 ， Google Cloud 或 Microsoft Azure 。可能值：
 - AWS
 - Azure
 - GCP
 - generic-S3
 - ONTAP S3
 - StorageGRID S3
- c. 在您以正確的值填入檔案之後 `trident-protect-appvault-primary-source.yaml` 、請套用 CR ：

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n trident-protect
```

使用 CLI 建立 AppVault

- a. 建立 AppVault ，以環境資訊取代方括號中的值：

```
tridentctl protect create vault Azure <vault-name> --account <account-name> --bucket <bucket-name> --secret <secret-name>
```

2. 建立來源應用程式 CR ：

使用 CR 建立來源應用程式

- a. 建立自訂資源（CR）檔案並命名（例如 `trident-protect-app-source.yaml`）。
- b. 設定下列屬性：
 - `* metadata.name*`:（`_required`）應用程式自訂資源的名稱。請記下您選擇的名稱，因為複寫關係所需的其他 CR 檔案會參照此值。
 - `* spec.includedNamespaces*`:（`_required`）一組命名空間和相關標籤。使用命名空間名稱，並選擇性地使用標籤來縮小命名空間的範圍，以指定此處列出的命名空間中存在的資源。應用程式命名空間必須是此陣列的一部分。
 - YAML* 範例：

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: maria
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: maria
    labelSelector: {}
```

- c. 在您以正確的值填入檔案之後 `trident-protect-app-source.yaml`、請套用 CR：

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-namespace
```

使用 CLI 建立來源應用程式

- a. 建立來源應用程式。例如：

```
tridentctl protect create app maria --namespaces maria -n my-app-namespace
```

3. 您也可以選擇拍攝來源應用程式的快照。此快照是作為目的地叢集上應用程式的基礎。如果您略過此步驟，則需要等待下一個排定的快照執行，以便擁有最近的快照。

使用 CR 拍攝快照

a. 建立來源應用程式的複寫排程：

- i. 建立自訂資源（CR）檔案並命名（例如 `trident-protect-schedule.yaml`）。
- ii. 設定下列屬性：
 - `* metadata.name*:`（`_required`）排程自訂資源的名稱。
 - `spec.AppVaultRef`：（`_required`）此值必須符合來源應用程式的 AppVault `metadata.name` 欄位。
 - `spec.ApplicationRef`：（`_required`）此值必須符合來源應用程式 CR 的 `metadata.name` 欄位。
 - `*spec.backupRetention*`：（`_required`）此欄位為必填欄位，且值必須設為 0。
 - `spec.enabled`：必須設置為 `true`。
 - `* spec.granularity*:` 必須設定為 `Custom`。
 - `spec.recurrenceRule`：以 UTC 時間和循環時間間隔定義開始日期。
 - `*spec.snapshotRetention*`：必須設定為 2。

YAML 範例：

```
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule-0e1f88ab-f013-4bce-8ae9-6afed9df59a1
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-46a3-420a-b351-45795e1b5e34
  applicationRef: maria
  backupRetention: "0"
  enabled: true
  granularity: custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

- i. 在您以正確的值填入檔案之後 `trident-protect-schedule.yaml`、請套用 CR：

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-namespace
```

使用 CLI 拍攝快照

- a. 建立快照，以您環境的資訊取代方括號中的值。例如：

```
tridentctl protect create snapshot <my_snapshot_name> --appvault  
<my_appvault_name> --app <name_of_app_to_snapshot>
```

4. 在目的地叢集上建立與您在來源叢集上套用的 AppVault CR 相同的來源應用程式 AppVault CR，並命名該應用程式（例如 `trident-protect-appvault-primary-destination.yaml`）。
5. 套用 CR：

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n  
my-app-namespace
```

6. 為目的地叢集上的目的地應用程式建立 AppVault。視您的儲存供應商而定，請修改中的範例"[AppVault 自訂資源](#)"以符合您的環境：
 - a. 建立自訂資源（CR）檔案並命名（例如 `trident-protect-appvault-secondary-destination.yaml`）。
 - b. 設定下列屬性：
 - `* metadata.name*`: (`_required`) AppVault 自訂資源的名稱。請記下您選擇的名稱，因為複寫關係所需的其他 CR 檔案會參照此值。
 - `* spec.providerConfig*`: (`_required`) 儲存使用指定供應商存取 AppVault 所需的組態。請為您的供應商選擇 `'bucketName'` 和任何其他必要詳細資料。請記下您選擇的值，因為複寫關係所需的其他 CR 檔案會參照這些值。如需 AppVault CRS 與其他供應商的範例，請參閱"[AppVault 自訂資源](#)"。
 - `* spec.providerCredentials*`: (`_required`) 會儲存使用指定提供者存取 AppVault 所需之任何認證的參考資料。
 - `* spec.providerCredentials.valueFromSecret*`: (`_required`) 表示認證值應來自機密。
 - `key` : (`_required`) 要從中選擇的密碼的有效金鑰。
 - `* 名稱 *` : (`_必要`) 包含此欄位值的機密名稱。必須位於相同的命名空間中。
 - `* spec.providerCredentials.secretAccessKey*`: (`_required`) 存取提供者所用的存取金鑰。`* 名稱 *` 應與 `* spec.providerCredentials.valueFromSecret.name*` 相符。
 - `* spec.providerType*`: (`_required`) 決定提供備份的內容，例如 NetApp ONTAP S3，一般 S3，Google Cloud 或 Microsoft Azure。可能值：
 - AWS
 - Azure
 - GCP
 - generic-S3
 - ONTAP S3
 - StorageGRID S3

- c. 在您以正確的值填入檔案之後 `trident-protect-appvault-secondary-destination.yaml`、請套用 CR：

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml  
-n my-app-namespace
```

7. 建立 AppMirrorRelationship CR 檔案：

使用 CR 建立 AppMirrorRelationship

- a. 建立自訂資源 (CR) 檔案並命名 (例如 trident-protect-relationship.yaml) 。
- b. 設定下列屬性：
 - * metadata.name:* (必要) AppMirrorRelationship 自訂資源的名稱。
 - * spec.destinationAppVaultRef*: (_required _) 此值必須符合目的地叢集上目的地應用程式的 AppVault 名稱。
 - * spec.namespaceMapping*: (_required _) 目的地和來源命名空間必須符合各自應用程式 CR 中定義的應用程式命名空間。
 - **spec.sourceAppVaultRef** : (_required _) 此值必須符合來源應用程式的 AppVault 名稱。
 - **spec.sourceApplicationName** : (_required _) 此值必須符合您在來源應用程式 CR 中定義的來源應用程式名稱。
 - **spec.storageClassName** : (_required _) 選擇叢集上有效儲存類別的名稱。儲存類別必須與部署來源應用程式的來源叢集上所使用的儲存類別對等。
 - **spec.recurrenceRule** : 以 UTC 時間和循環時間間隔定義開始日期。

YAML 範例：

```
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
  namespace: my-app-namespace
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: maria
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsims-2
```

- c. 在您以正確的值填入檔案之後 trident-protect-relationship.yaml 、請套用 CR ：

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

使用 CLI 建立 AppMirrorRelationship

- a. 建立並套用 AppMirrorRelationship 物件，以環境資訊取代方括號中的值。例如：

```
tridentctl protect create appmirrorrelationship  
<name_of_appmirrorrelationship> --destination-app-vault  
<my_vault_name> --recurrence-rule <rule> --source-app  
<my_source_app> --source-app-vault <my_source_app_vault>
```

8. (Optional) 檢查複寫關係的狀態和狀態：

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath  
='{.status}' | jq
```

容錯移轉至目的地叢集

使用 Trident Protect，您可以將複寫的應用程式容錯移轉至目的地叢集。此程序會停止複寫關係、並在目的地叢集上使應用程式上線。如果來源叢集上的應用程式正常運作，Trident Protect 不會停止該應用程式。

步驟

1. 開啟 AppMirrorRelationship CR 檔案 (例如 trident-protect-relationship.yaml)，並將 *spec.desiredState* 的值變更為 Promoted。
2. 儲存 CR 檔案。
3. 套用 CR：

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (Optional) 在容錯移轉應用程式上建立所需的任何保護排程。
5. (Optional) 檢查複寫關係的狀態和狀態：

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath  
='{.status}' | jq
```

重新同步容錯移轉複寫關係

重新同步作業會重新建立複寫關係。執行重新同步作業後，原始來源應用程式即成為執行中的應用程式，而對目

的地叢集上執行中的應用程式所做的任何變更都會被捨棄。

此程序會在重新建立複寫之前，停止目的地叢集上的應用程式。



在容錯移轉期間寫入目的地應用程式的任何資料都會遺失。

步驟

1. 建立來源應用程式的快照。
2. 打開 AppMirrorRelationship CR 文件（例如 trident-protect-relationship.yaml），然後將 spec.desiredState 的值更改為 Established。
3. 儲存 CR 檔案。
4. 套用 CR：

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. 如果您在目的地叢集上建立任何保護排程來保護容錯移轉應用程式，請將其移除。任何仍會導致磁碟區快照失敗的排程。

反轉重新同步容錯移轉複寫關係

當您反向重新同步容錯移轉複寫關係時，目的地應用程式會變成來源應用程式，來源會變成目的地。在容錯移轉期間對目的地應用程式所做的變更會保留下來。

步驟

1. 刪除原始目的地叢集上的 AppMirrorRelationship CR。這會導致目的地成為來源。如果新的目的地叢集上還有任何保護排程，請將其移除。
2. 套用原先用來設定與相對叢集關係的 CR 檔案，以設定複寫關係。
3. 確保每個叢集上的 AppVault CRS 均已就緒。
4. 在相對的叢集上設定複寫關係，設定反轉方向的值。

反轉應用程式複寫方向

當您反轉複寫方向時，Trident Protect 會將應用程式移至目的地儲存後端，同時繼續複寫回原始來源儲存後端。Trident Protect 會停止來源應用程式，並在容錯移轉至目的地應用程式之前，將資料複寫到目的地。

在這種情況下、您要交換來源和目的地。

步驟

1. 建立關機快照：

使用 CR 建立關機快照

- a. 停用來源應用程式的保護原則排程。
- b. 建立 ShutdownSnapshot CR 檔案：
 - i. 建立自訂資源（CR）檔案並命名（例如 `trident-protect-shutdownsnapshot.yaml`）。
 - ii. 設定下列屬性：
 - `* metadata.name*`:（`_required`）自訂資源的名稱。
 - `spec.AppVaultRef` :（`_required`）此值必須符合來源應用程式的 AppVault `metadata.name` 欄位。
 - `spec.ApplicationRef` :（`_required`）此值必須符合來源應用程式 CR 檔案的 `metadata.name` 欄位。

YAML 範例：

```
apiVersion: protect.trident.netapp.io/v1
kind: ShutdownSnapshot
metadata:
  name: replication-shutdown-snapshot-afc4c564-e700-4b72-86c3-
c08a5dbe844e
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: maria
```

- c. 在您以正確的值填入檔案之後 `trident-protect-shutdownsnapshot.yaml`、請套用 CR：

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

使用 CLI 建立關機快照

- a. 建立關機快照，以環境資訊取代方括號中的值。例如：

```
tridentctl protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot>
```

2. 快照完成後，取得快照的狀態：

```
kubectl get shutdownsnapshot -n my-app-namespace  
<shutdown_snapshot_name> -o yaml
```

3. 使用下列命令尋找 * shutdownsnapshot .status.appArchivePath* 的值，並記錄檔案路徑的最後一部分（也稱為 `basename`；這將是最後一個斜線之後的所有項目）：

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o  
jsonpath='{.status.appArchivePath}'
```

4. 執行容錯移轉，從目的地叢集移轉至來源叢集，並進行下列變更：



在容錯移轉程序的步驟 2 中，將欄位包含在 `spec.promotedSnapshot` `AppMirrorRelationship` CR 檔案中，並將其值設為您在上述步驟 3 中記錄的基礎名稱。

5. 執行中的反向重新同步步驟[[反轉重新同步容錯移轉複寫關係](#)]。
6. 在新的來源叢集上啟用保護排程。

結果

由於反向複寫，因此會發生下列動作：

- 原始來源應用程式的 Kubernetes 資源會擷取快照。
- 刪除應用程式的 Kubernetes 資源（保留 PVCS 和 PVs）、即可順利停止原始來源應用程式的 Pod。
- 當 Pod 關機之後、應用程式的磁碟區快照就會被擷取和複寫。
- SnapMirror 關係中斷、使目的地磁碟區準備好進行讀寫。
- 應用程式的 Kubernetes 資源會從關機前快照還原、並使用原始來源應用程式關機後複寫的 Volume 資料。
- 複寫會以相反方向重新建立。

將應用程式容錯移轉至原始來源叢集

使用 Trident Protect，您可以使用下列作業順序，在容錯移轉作業之後達成「容錯回復」。在此工作流程中，為了還原原始複寫方向，Trident Protect 會在還原複寫方向之前，將任何應用程式變更複寫回原始來源應用程式。

此程序從已完成容錯移轉至目的地的關係開始、並涉及下列步驟：

- 從容錯移轉狀態開始。
- 反向重新同步複寫關係。



請勿執行正常的重新同步作業，因為這會捨棄在容錯移轉程序期間寫入目的地叢集的資料。

- 反轉複寫方向。

步驟

1. 執行[[反轉重新同步容錯移轉複寫關係](#)]步驟。

2. 執行[\[反轉應用程式複寫方向\]](#)步驟。

刪除複寫關係

您可以隨時刪除複寫關係。當您刪除應用程式複寫關係時，會產生兩個獨立的應用程式，兩者之間沒有任何關係。

步驟

1. 刪除 AppMirrorRelationship CR ：

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

移轉應用程式

您可以將備份或快照資料還原至不同的叢集或儲存類別，在叢集或儲存類別之間移轉應用程式。



當您移轉應用程式時，為應用程式設定的所有執行掛鉤都會隨應用程式一起移轉。如果存在還原後執行掛鉤，則會在還原作業中自動執行。

備份與還原作業

若要針對下列案例執行備份與還原作業，您可以自動化特定的備份與還原工作。

複製到同一個叢集

若要將應用程式複製到同一個叢集，請建立快照或備份，然後將資料還原到同一個叢集。

步驟

1. 執行下列其中一項：
 - a. ["建立快照"](#)。
 - b. ["建立備份"](#)。
2. 在同一個叢集上，視您建立的是快照或備份而定，請執行下列其中一項：
 - a. ["從快照還原資料"](#)。
 - b. ["從備份還原資料"](#)。

複製到不同叢集

若要將應用程式複製到不同的叢集（執行跨叢集複製），請建立快照或備份，並將資料還原到不同的叢集。請確定目的地叢集上已安裝 Trident Protect。

步驟

1. 執行下列其中一項：
 - a. ["建立快照"](#)。

- b. "建立備份"。
2. 請確定已在目的地叢集上設定包含備份或快照之物件儲存貯體的 AppVault CR。
3. 在目的地叢集上，視您建立的是快照或備份而定，請執行下列其中一項：
 - a. "從快照還原資料"。
 - b. "從備份還原資料"。

將應用程式從一個儲存類別移轉至另一個儲存類別

您可以將快照還原至不同的目的地儲存類別，將應用程式從一個儲存類別移轉至不同的儲存類別。

例如（從還原 CR 中排除機密）：

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    destination: "${destinationNamespace}"
    source: "${sourceNamespace}"
  storageClassMapping:
    destination: "${destinationStorageClass}"
    source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

使用 CR 還原快照

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-snapshot-restore-cr.yaml`。
2. 在您建立的檔案中，設定下列屬性：
 - `* metadata.name*`:（`_required`）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
 - `spec.appArchivePath`：在 AppVault 中儲存快照內容的路徑。您可以使用下列命令來尋找此路徑：

```
kubectl get snapshots <my-snapshot-name> -n trident-protect -o jsonpath='{.status.appArchivePath}'
```

- `spec.appVaultRef`：（`_required`）儲存快照內容的 AppVault 名稱。
- `spec.namespaceMapping`: 將還原作業的來源命名空間對應至目的地命名空間。以環境中的資訊取代 `my-source-namespace` 和 `my-destination-namespace`。

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: my-cr-name
  namespace: trident-protect
spec:
  appArchivePath: my-snapshot-path
  appVaultRef: appvault-name
  namespaceMapping: [{"source": "my-source-namespace",
"destination": "my-destination-namespace"}]
```

3. 或者，如果您只需要選取要還原的應用程式特定資源，請新增篩選功能，以包含或排除標記有特定標籤的資源：
 - `*resourceFilter.resourceSelectionCriteria`：（篩選所需）用於 `'include or exclude'` 包含或排除在 `resourceMatchers` 中定義的資源。新增下列資源配置工具參數、以定義要納入或排除的資源：
 - `* 要篩選的資源的 resourceMatchers.group*`:（*Optional*）群組。
 - `resourceMatchers.y`：（*Optional*）要篩選的資源種類。
 - `resourceMatchers.version`：（*Optional*）要篩選的資源版本。
 - `* 要篩選之資源的 Kubernetes metadata.name 欄位中的 resourceMatchers.names*`:（*Optional*）名稱。
 - `* 要篩選之資源的 Kubernetes metadata.name 欄位中的 resourceMatchers.namespaces*`:（*Optional*）命名空間。
 - 資源的 Kubernetes metadata.name 欄位中的 `*resourceMatchers.labelSelectors*`：（*Optional*）Label 選取器字串，如中所定義 ["Kubernetes文件"](#)。例如 `"trident.netapp.io/os=linux"`：

例如：

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      group: my-resource-group
      kind: my-resource-kind
      version: my-resource-version
      names: ["my-resource-names"]
      namespaces: ["my-resource-namespaces"]
      labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 在您以正確的值填入檔案之後 `trident-protect-snapshot-restore-cr.yaml`、請套用 CR：

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

使用 CLI 還原快照

1. 將快照還原至不同的命名空間，以環境中的資訊取代方括號中的值。
 - `snapshot` 引數使用格式的命名空間和快照名稱 `<namespace>/<name>`。
 - 此 `namespace-mapping` 引數使用以冒號分隔的命名空間，以格式將來源命名空間對應至正確的目的地命名空間 `source1:dest1, source2:dest2`。

例如：

```
tridentctl protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

管理執行攔截器

執行攔截是一種自訂動作、可設定搭配託管應用程式的資料保護作業一起執行。例如、如果您有資料庫應用程式、您可以使用執行掛勾來暫停快照之前的所有資料庫交易、並在快照完成後繼續交易。如此可確保應用程式一致的快照。

執行掛勾的類型

Trident Protect 支援下列類型的執行掛鉤，視執行時機而定：

- 快照前

- 快照後
- 預先備份
- 備份後
- 還原後
- 容錯移轉後

執行順序

執行資料保護作業時、執行掛機事件會依照下列順序發生：

1. 任何適用的自訂操作前執行掛勾都會在適當的容器上執行。您可以視需要建立及執行任意數量的自訂操作前掛勾、但在作業之前執行這些掛勾的順序既不保證也無法設定。
2. 執行資料保護作業。
3. 任何適用的自訂操作後執行掛勾都會在適當的容器上執行。您可以視需要建立及執行任意數量的自訂後置作業掛勾、但在作業後執行這些掛勾的順序並不保證也無法設定。

如果您建立同一類型的多個執行掛勾（例如預先快照）、則無法保證這些掛勾的執行順序。不過、不同類型的掛勾的執行順序也有保證。例如，以下是具有所有不同類型勾點的組態執行順序：

1. 執行快照前掛勾
2. 快照後掛勾已執行
3. 執行備份前掛勾
4. 執行備份後掛勾



上述順序範例僅適用於執行不使用現有快照的備份時。



在正式作業環境中啟用執行攔截指令碼之前、請務必先進行測試。您可以使用'`kubect exec`'命令來方便地測試指令碼。在正式作業環境中啟用執行掛勾之後、請測試所產生的快照和備份、以確保它們一致。您可以將應用程式複製到暫用命名空間、還原快照或備份、然後測試應用程式、藉此完成此作業。

關於自訂執行掛勾的重要注意事項

規劃應用程式的執行掛勾時、請考量下列事項。

- 執行攔截必須使用指令碼來執行動作。許多執行掛勾可以參照相同的指令碼。
- Trident Protect 需要以可執行的 Shell 指令碼格式寫入執行攔截程式所使用的指令碼。
- 指令碼大小上限為96KB。
- Trident Protect 使用執行掛鉤設定和任何符合條件，來判斷哪些掛勾適用於快照，備份或還原作業。



由於執行掛勾通常會減少或完全停用執行中應用程式的功能、因此您應該一律盡量縮短自訂執行掛勾執行所需的時間。如果您以相關的執行掛勾開始備份或快照作業、但隨後取消它、則如果備份或快照作業已經開始、仍允許掛勾執行。這表示備份後執行掛勾中使用的邏輯無法假設備份已完成。

執行攔截篩選器

當您新增或編輯應用程式的執行掛鉤時，您可以將篩選器新增至執行掛鉤，以管理掛鉤將符合的容器。篩選器對於在所有容器上使用相同容器映像的應用程式來說非常實用、但可能會將每個映像用於不同的用途（例如Elasticsearch）。篩選器可讓您建立執行攔截器在某些容器上執行的案例、但不一定所有容器都相同。如果您為單一執行掛勾建立多個篩選器、這些篩選器會與邏輯和運算子結合使用。每個執行掛機最多可有10個作用中篩選器。

您新增至執行掛勾的每個篩選器都會使用規則運算式來比對叢集中的容器。當掛機符合容器時、掛機會在該容器上執行其相關的指令碼。篩選器的規則運算式使用規則運算式2（RE2）語法、不支援建立篩選器、將容器從相符項目清單中排除。如需 Trident Protect 支援執行攔截篩選器中規則運算式的語法資訊，請參閱 "[規則運算式2（RE2）語法支援](#)"。



如果您將命名空間篩選器新增至執行掛勾、而執行還原或複製作業之後執行、且還原或複製來源與目的地位於不同的命名空間、則命名空間篩選器只會套用至目的地命名空間。

執行攔截範例

請造訪 "[NetApp Verda GitHub專案](#)" 下載熱門應用程式（例如 Apache Cassandra 和 Elasticsearch）的實際執行連結。您也可以查看範例、瞭解如何建構您自己的自訂執行掛勾。

建立執行掛鉤

您可以使用 Trident Protect 為應用程式建立自訂執行掛鉤。您需要擁有擁有者、管理員或成員權限、才能建立執行掛勾。

使用 CR

1. 建立自訂資源（CR）檔案並命名為 `trident-protect-hook.yaml`。
2. 設定下列屬性以符合 Trident Protect 環境和叢集組態：
 - `* metadata.name*:`（*_required*）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
 - **SPEC.applicationRef**：（*_required*）要執行執行攔截的應用程式 Kubernetes 名稱。
 - `*spec.Stage *`：（*_required*）一個字串，指出執行掛鉤應在動作期間執行的階段。可能值：
 - 準備
 - 貼文
 - **spec.ACTION**：（*_required*）字串，表示執行攔截將採取的行動，前提是指定的任何執行攔截篩選條件都已相符。可能值：
 - Snapshot
 - 備份
 - 還原
 - 容錯移轉
 - **spec.enabled**：（*Optional*）表示此執行掛鉤是否已啟用或停用。如果未指定，則預設值為 `true`。
 - **spec.hookSource**：（*_required*）包含 base64 編碼 hook 指令碼的字串。
 - **spec.timeout**：（*Optional*）一個數字，定義允許執行掛鉤執行的時間（以分鐘為單位）。最小值為 1 分鐘，如果未指定，預設值為 25 分鐘。
 - **spec.arguments**：（*Optional*）YAML 引數清單，您可以為執行攔截器指定。
 - `*spec.mismatchingCriteria`：（*Optional*）選擇性的條件金鑰值配對清單，每個配對組成執行掛鉤篩選器。每個執行掛鉤最多可新增 10 個篩選器。
 - **spec.matchingCriteria.type**：（*Optional*）識別執行掛鉤篩選器類型的字串。可能值：
 - ContainerImage
 - ContainerName
 - PodName
 - PodLabel
 - NamespaceName
 - **spec.matchingCriteria.value**：（*Optional*）識別執行掛鉤篩選值的字串或規則運算式。

YAML 範例：

```
apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production
```

3. 在您以正確的值填入 CR 檔案之後，請套用 CR：

```
kubectl apply -f trident-protect-hook.yaml
```

使用CLI

1. 建立執行掛鉤，以環境資訊取代方括號中的值。例如：

```
tridentctl protect create exehook <my_exec_hook_name> --action  
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>  
--source-file <script-file>
```

解除安裝 Trident Protect

如果您要從試用版升級至完整版產品，則可能需要移除 Trident Protect 元件。

若要移除 Trident Protect，請執行下列步驟。

步驟

1. 移除 Trident Protect CR 檔案：

```
helm uninstall trident-protect-crds
```

2. 移除 Trident Protect：

```
helm uninstall -n trident-protect trident-protect
```

3. 移除 Trident Protect 命名空間：

```
kubectl delete ns trident-protect
```

知識與支援

常見問題集

尋找有關安裝、設定、升級及疑難排解 Trident 的常見問題集解答。

一般問題

Trident 的發行頻率為何？

從 24.02 版開始、Trident 每四個月發佈一次：2 月、6 月和 10 月。

Trident 是否支援特定版本的 **Kubernetes** 所發行的所有功能？

Trident 通常不支援 Kubernetes 中的 Alpha 功能。Trident 可能支援 Kubernetes 試用版之後的兩個 Trident 版本中的試用版功能。

Trident 是否因其運作而對其他 **NetApp** 產品有任何相依性？

Trident 與其他 NetApp 軟體產品沒有任何相依關係、而且可作為獨立應用程式使用。不過、您應該擁有 NetApp 後端儲存設備。

如何取得完整的 **Trident** 組態詳細資料？

使用 `tridentctl get` 命令可取得有關 Trident 組態的更多資訊。

我是否可以取得 **Trident** 如何配置儲存設備的計量標準？

是的。可用來收集 Trident 作業相關資訊的 Prometheus 端點、例如管理的後端數、已配置的磁碟區數量、使用的位元組等。您也可以用於 ["Cloud Insights"](#) 監控和分析。

使用 **Trident** 做為 **CSI** 資源配置程式時、使用者體驗是否會改變？

否。就使用者體驗和功能而言、沒有任何變更。使用的置備程式名稱為 `csi.trident.netapp.io`。如果您想要使用目前和未來版本所提供的所有新功能、建議您使用這種安裝 Trident 的方法。

在 **Kubernetes** 叢集上安裝及使用 **Trident**

Trident 是否支援從私人登錄進行離線安裝？

可以、Trident 可以離線安裝。請參閱 ["瞭解 Trident 安裝"](#)。

我可以遠端安裝 **Trident** 嗎？

是的。Trident 18.10 及更新版本支援從任何可存取叢集的機器進行遠端安裝 `kubectl`。驗證存取後 `kubectl`（例如、從遠端機器啟動 `kubectl get nodes` 命令以驗證）、請遵循安裝指示進行。

我可以使用 **Trident** 設定高可用度嗎？

Trident 是以 Kubernetes 部署（ReplicaSet）的形式安裝、其中包含一個執行個體、因此內建了 HA。您不應該增加部署中的複本數量。如果安裝 Trident 的節點遺失、或 Pod 無法存取、Kubernetes 會自動將 Pod 重新部署至叢集中的正常節點。Trident 僅適用於控制面板、因此如果重新部署 Trident、目前安裝的 Pod 不會受到影響。

Trident 是否需要存取 kube 系統命名空間？

Trident 從 Kubernetes API 伺服器讀取資料、以判斷應用程式何時要求新的 PVC、因此需要存取 kube-system。

Trident 使用哪些角色和 Privileges？

Trident 安裝程式會建立 Kubernetes ClusterRole、該程式可存取叢集的 PersistentVolume、PersistentVolume Claim、StorageClass 和 Kubernetes 叢集的 Secret 資源。請參閱 "[自訂tridentctl安裝](#)"。

我可以本機產生 **Trident** 用於安裝的確切資訊清單檔案嗎？

如有需要、您可以在本機產生及修改 Trident 用於安裝的確切資訊清單檔案。請參閱 "[自訂tridentctl安裝](#)"。

我是否可以針對兩個獨立的 **Kubernetes** 叢集、共用兩個獨立 **Trident** 執行個體的相同 **ONTAP** 後端 **SVM**？

雖然不建議使用相同的後端 SVM 來執行兩個 Trident 執行個體。在安裝期間為每個執行個體指定唯一的磁碟區名稱、及 / 或在檔案中指定唯一的 StoragePrefix 參數 `setup/backend.json`。這是為了確保 FlexVol 兩個執行個體都不會使用相同的功能。

是否能在 **ContainerLinux**（前身為 **CoreOS**）下安裝 **Trident**？

Trident 只是 Kubernetes Pod、可在 Kubernetes 執行的任何地方安裝。

我可以搭配 **NetApp Cloud Volumes ONTAP** 使用 **Trident** 嗎？

是的、AWS、Google Cloud 和 Azure 支援 Trident。

Trident 是否能與雲端 **Volume Services** 搭配運作？

是的、Trident 支援 Azure 中的 Azure NetApp Files 服務、以及 GCP 中的 Cloud Volumes Service。

疑難排解與支援

NetApp 是否支援 **Trident**？

雖然 Trident 是免費提供的開放原始碼、但只要支援您的 NetApp 後端、NetApp 就能完全支援。

如何提出支援案例？

若要提出支援案例、請執行下列其中一項：

1. 請聯絡您的支援客戶經理、以取得索取機票的協助。

2. 請聯絡以提出支援案例 ["NetApp支援"](#)。

如何產生支援記錄套裝組合？

您可以執行「tridentctl logs -A」來建立支援服務組合。除了在套裝組合中擷取的記錄之外、請擷取kubelet記錄、以診斷Kubernetes端的掛載問題。取得Kubernetes記錄的指示會根據Kubernetes的安裝方式而有所不同。

如果我需要提出新功能的要求、該怎麼辦？

在問題的主題和說明中建立問題 ["Trident Github"](#)、並提及 * RFE*。

我該在哪裡提出瑕疵？

在上建立問題 ["Trident Github"](#)。請務必附上與問題相關的所有必要資訊和記錄。

如果我有關於 **Trident** 的快速問題需要澄清、會發生什麼事？是否有社群或論壇？

如果您有任何問題、問題或要求、請透過我們的 Trident 或 GitHub 與我們聯絡["不和通路"](#)。

我的儲存系統密碼已變更、**Trident** 無法再運作、我該如何恢復？

使用更新後端的密碼 `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`。更換 `myBackend` 在範例中、使用您的後端名稱、和 `/path/to_new_backend.json` 並將路徑移至正確位置 `backend.json` 檔案：

Trident 找不到我的 **Kubernetes** 節點。如何修正此問題？

Trident 找不到 Kubernetes 節點的可能情況有兩種。這可能是因為Kubernetes內的網路問題或DNS問題。在每個Kubernetes節點上執行的Trident節點取消影像集、必須能夠與Trident控制器通訊、才能在Trident中登錄節點。如果在安裝 Trident 之後發生網路變更、則只有新增至叢集的 Kubernetes 節點才會發生此問題。

如果**Trident Pod**毀損、我會遺失資料嗎？

如果Trident Pod遭到破壞、資料將不會遺失。Trident 中繼資料儲存在 CRD 物件中。所有由Trident提供的PV均可正常運作。

升級 Trident

我可以直接從舊版本升級至新版本（跳過幾個版本）嗎？

NetApp 支援將 Trident 從一個主要版本升級至下一個立即的主要版本。您可以從11.xx版升級至19.xx、19.xx版升級至20.xx版、依此類推。在正式作業部署之前、您應該先在實驗室中測試升級。

是否能將**Trident**降級至先前的版本？

如果您需要修正在升級、相依性問題或升級失敗或不完整之後所觀察到的錯誤、您應該["解除安裝 Trident"](#)使用該版本的特定指示重新安裝舊版。這是降級至舊版的唯一建議方法。

管理後端和磁碟區

我是否需要在**ONTAP** 一個後端定義檔案中定義管理和資料生命期？

管理LIF為必填項目。資料LIF會有所不同：

- 支援SAN：請勿指定iSCSI ONTAP。Trident 使用"[可選擇的LUN對應ONTAP](#)"來探索建立多重路徑工作階段所需的 iSCSI 生命。如果明確定義、就會產生警告 dataLIF。如 "[SAN組態選項與範例ONTAP](#)" 需詳細資訊、請參閱。
- ONTAP NAS：建議指定 dataLIF。如果未提供、Trident 會從 SVM 擷取資料生命。您可以指定要用於NFS掛載作業的完整網域名稱（FQDN）、讓您建立循環配置資源DNS、以便在多個資料生命期之間達到負載平衡。如"[列舉NAS組態選項與範例ONTAP](#)"需詳細資訊、請參閱

Trident 是否可以為 **ONTAP** 後端設定 **CHAP** ？

是的。Trident 支援 ONTAP 後端的雙向 CHAP。這需要在後端組態中設定 `useCHAP=true`。

如何使用 **Trident** 管理匯出原則？

Trident 可從 20.04 版開始、動態建立及管理匯出原則。如此一來、儲存管理員就能在其後端組態中提供一或多個CIDR區塊、並將位於這些範圍內的Trident新增節點IP、加入其所建立的匯出原則。如此一來、Trident 便會自動管理在指定的 CIDR 內新增和刪除具有 IP 的節點規則。

IPv6位址是否可用於管理和資料生命量？

Trident 支援定義下列項目的 IPv6 位址：

- `managementLIF` 和 `dataLIF` 適用於不支援NAS的後端ONTAP。
- `managementLIF` 適用於SAN後端ONTAP。您無法指定 `dataLIF` 在SAN後端ONTAP。

Trident 必須使用旗標（用於 `tridentctl` 安裝）、（用於 Trident 運算子）或 `\tridentTPv6`（用於 Helm 安裝）來安裝 `--use-ipv6`、``IPv6``才能透過 IPv6 運作。

是否能在後端更新管理**LIF**？

可以、您可以使用「`tridentctl update backend`」命令來更新後端管理LIF。

是否能在後端更新**Data LIF**？

您可以在上更新Data LIF `ontap-nas` 和 `ontap-nas-economy` 僅限。

我可以在 **Kubernetes** 的 **Trident** 中建立多個後端嗎？

Trident 可以同時支援多個後端、無論是使用相同的驅動程式或不同的驅動程式。

Trident 如何儲存後端認證？

Trident 將後端認證儲存為 Kubernetes Secrets。

Trident 如何選擇特定後端？

如果後端屬性無法用於自動選擇某個類的正確池，則可使用"storagePools"和"additionalStoragePools"參數來選擇特定的池集區集區集區集區。

如何確保 Trident 不會從特定後端進行資源配置？

此 `excludeStoragePools` 參數用於篩選 Trident 用於資源配置的資源池集、並移除任何符合的資源池。

如果有相同類型的多個後端、Trident 如何選擇要使用的後端？

如果有多個相同類型的設定後端、Trident 會根據和 PersistentVolumeClaim 中的參數來選取適當的後端 StorageClass。例如，如果有多個 ONTAP — NAS 驅動程序後端，Trident 會嘗試匹配中的參數 StorageClass，並 PersistentVolumeClaim 將後端組合起來，以滿足和 PersistentVolumeClaim 中列出的要求 StorageClass。如果有多個符合要求的後端、Trident 會隨機選取其中一個。

Trident 是否支援元素 / SolidFire 的雙向 CHAP ？

是的。

Trident 如何在 ONTAP 磁碟區上部署 qtree ？單一磁碟區可部署多少 qtree ？

「ONTAP-NAS-節約」驅動程式可在同FlexVol 一個範圍內建立多達200個qtree（可設定為50到300個）、每個叢集節點可建立100、000個qtree、每個叢集可建立240萬個qtree。當您輸入經濟型驅動程式所提供的全新「PersistentVolume Claim」時、駕駛會查看FlexVol 是否已存在可為新Qtree提供服務的功能。如果FlexVol 不存在能夠服務Qtree的功能、FlexVol 就會建立新的功能。

我要如何為ONTAP 以NAS配置的Volume設定Unix權限？

您可以在後端定義檔中設定參數、在 Trident 所佈建的磁碟區上設定 Unix 權限。

如何在ONTAP 配置Volume時、設定一組明確的靜態NFS掛載選項？

根據預設、Trident 不會使用 Kubernetes 將掛載選項設定為任何值。要在 Kubernetes Storage Class 中指定掛載選項，請按照給定的示例[請按這裡](#)操作。

如何將已配置的磁碟區設定為特定的匯出原則？

若要允許適當的主機存取磁碟區、請使用後端定義檔中設定的「exportPolicy」參數。

如何透過 Trident with ONTAP 設定磁碟區加密？

您可以使用後端定義檔中的加密參數、在Trident所提供的磁碟區上設定加密。如需詳細資訊、請參閱：["Trident 如何與 NVE 和 NAE 搭配運作"](#)

透過 Trident 實作 ONTAP QoS 的最佳方法為何？

使用「儲存類」來實作ONTAP QoS以利實現。

如何透過 **Trident** 指定精簡或完整資源配置？

支援精簡或密集資源配置的支援。ONTAP此功能預設為精簡配置。ONTAP如果需要完整資源配置、您應該設定後端定義檔或「儲存類別」。如果兩者都已設定、則「儲存類別」優先。設定ONTAP 下列項目以供參考：

1. 在「儲存類別」上、將「資源配置類型」屬性設為「完整」。
2. 在後端定義檔中、將「backend spaceReserve參數」設為Volume、以啟用厚磁碟區。

如何確保即使意外刪除了**PVC**,也不會刪除使用中的磁碟區？

Kubernetes從1.10版開始自動啟用PVC保護。

我可以擴充 **Trident** 所建立的 **NFS PVCS** 嗎？

是的。您可以擴充 Trident 所建立的 PVC 。請注意、Volume自動擴充ONTAP 是不適用於Trident的功能。

我可以在磁碟區處於**SnapMirror**資料保護 (DP) 或離線模式時匯入該磁碟區嗎？

如果外部磁碟區處於DP模式或離線、則磁碟區匯入會失敗。您會收到下列錯誤訊息：

```
Error: could not import volume: volume import failed to get size of
volume: volume <name> was not found (400 Bad Request) command terminated
with exit code 1.
Make sure to remove the DP mode or put the volume online before importing
the volume.
```

資源配額如何轉譯至**NetApp**叢集？

只要NetApp儲存設備具備容量、Kubernetes儲存資源配額就能運作。當 NetApp 儲存設備因容量不足而無法執行 Kubernetes 配額設定時、Trident 會嘗試進行資源配置、但會排除錯誤。

我可以使用 **Trident** 建立 **Volume Snapshot** 嗎？

是的。Trident 支援從快照建立隨需磁碟區快照和持續磁碟區。若要從快照建立 PV 、請確定 `VolumeSnapshotDataSource` 功能閘道已啟用。

哪些驅動程式支援 **Trident Volume** 快照？

到目前為止、我們的「ONTAP-NAS」、「ONTAP-NAS-flexgroup」、「ONTAP-SAN」、「ONTAP-san經濟型」、「Poolidfire SAN」、「GCP-CVS」、以及「azure-NetApp-fil」後端驅動程式。

我要如何使用 **ONTAP** 對由 **Trident** 所佈建的磁碟區進行快照備份？

這可在「ONTAP-NAS」、「ONTAP-SAN」及「ONTAP-NAA-flexgroup」等驅動程式上使用。您也可以針對FlexVol「ontap-san經濟」驅動程式指定「快照原則」、以利執行此作業。

這也可在驅動程式上使用、但在 FlexVol 層級精細度上使用 `ontap-nas-economy`、而不是在 `qtree` 層級精細度上使用。若要啟用由 Trident 提供快照磁碟區的功能、請將後端參數選項設定為 ONTAP 後端 `snapshotPolicy` 上定義的所需快照原則。Trident 不知道儲存控制器所拍攝的任何快照。

我可以為透過 **Trident** 配置的磁碟區設定快照保留百分比嗎？

是的、您可以在後端定義檔中設定屬性、以保留特定百分比的磁碟空間、以便透過 Trident 儲存快照複本 `snapshotReserve`。如果您已設定 `snapshotPolicy` 後端定義檔中的、`snapshotReserve` 則會根據後端檔案中所述的百分比來設定快照保留 `snapshotReserve` 百分比。如果 `snapshotReserve` 未提及百分比數、則 ONTAP 預設會將快照保留百分比視為 5。如果選項設為「無」、則 `snapshotPolicy` 快照保留百分比會設為 0。

我可以直接存取 **Volume Snapshot** 目錄並複製檔案嗎？

是的、您可以在後端定義檔中設定 `shapshotDir` 參數、以存取 Trident 所佈建之磁碟區上的 Snapshot 目錄。

我可以透過 **Trident** 為磁碟區設定 **SnapMirror** 嗎？

目前、SnapMirror 必須使用 ONTAP CLI 或 OnCommand 《系統管理程式》從外部設定。

如何將持續磁碟區還原至特定 **ONTAP** 的不還原快照？

若要將磁碟區還原 ONTAP 成一個無法修復的快照、請執行下列步驟：

1. 靜止使用持續磁碟區的應用程式 Pod。
2. 透過 ONTAP NetApp CLI 或 OnCommand 《系統管理程式》回復至所需的快照。
3. 重新啟動應用程式 Pod。

是否能在已設定負載共享鏡射的 **SVM** 上、對磁碟區進行 **Trident** 資源配置？

您可以為透過 NFS 提供資料的 SVM 根磁碟區建立負載共享鏡像。針對 Trident 所建立的磁碟區、自動更新負載共享鏡像。ONTAP 這可能會導致掛載磁碟區延遲。使用 Trident 建立多個磁碟區時、資源配置磁碟區會仰賴 ONTAP 於更新負載共享鏡像。

如何區分每位客戶/租戶的儲存類別使用量？

Kubernetes 不允許命名空間中的儲存類別。不過、您可以使用 Kubernetes 來限制每個命名空間的特定儲存類別使用量、方法是使用儲存資源配額（每個命名空間）。若要拒絕特定儲存設備的特定命名空間存取、請將該儲存類別的資源配額設為 0。

疑難排解

請使用此處提供的指標來疑難排解您在安裝和使用 Trident 時可能遇到的問題。

一般疑難排解

- 如果 Trident pod 無法正常啟動（例如、當 Trident pod 卡在「ContainerCreating」階段、且只有兩個可用的容器）、則執行「`kubectrl -n trident 描述部署 Trident`」和「`kubectrl -n trident 描述 pod trident -`」提供更多洞見。取得 kubelet 記錄（例如透過「`journalctl -xeu kubelet`」）也很有幫助。
- 如果 Trident 記錄中的資訊不足、您可以根據安裝選項、將「-d」旗標傳給安裝參數、嘗試啟用 Trident 的偵錯模式。

然後使用「`/tridentctl logs -n trident`」確認偵錯設定、並在記錄中搜尋「level = 偵錯msg」。

與營運者一起安裝

```
kubectl patch torc trident -n <namespace> --type=merge -p
'{"spec":{"debug":true}}'
```

這會重新啟動所有Trident Pod、可能需要數秒鐘的時間。您可以查看輸出「kubectl Get pod -n trident」中的「年齡」欄位來檢查。

對於 Trident 20.07 和 20.10，請使用 tprov 代替 torc。

與Helm一起安裝

```
helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
tridentDebug=true`
```

安裝試用版

```
./tridentctl uninstall -n trident
./tridentctl install -d -n trident
```

- 您也可以在后端定義中加入「debugTraceFlags」、以取得每個後端的偵錯記錄。例如、在Trident記錄中加入「debugTraceFlags: {"API": true、"method": true、}」以取得API呼叫和方法反向。現有的後端可設定「debugTraceFlags」、設定為「tridentctl後端更新」。
- 使用RedHat CoreOS時、請確定已在工作節點上啟用「iscsid」、並預設為啟動。您可以使用OpenShift機器組態或修改點火模板來完成此作業。
- 使用Trident時可能會遇到的常見問題 "[Azure NetApp Files](#)" 當租戶和用戶端機密來自權限不足的應用程式登錄時。如需 Trident 需求的完整清單、請參閱 "[Azure NetApp Files](#)" 組態：
- 如果將PV掛載到容器時發生問題、請確定已安裝並執行「rpcbind」。使用主機作業系統所需的套件管理程式、檢查「rpcbind」是否正在執行。您可以執行「systemctl狀態rpcbind」或其等效項目、來檢查「rpcbind」服務的狀態。
- 如果Trident後端回報雖然曾經工作、但仍處於「失敗」狀態、則可能是因為變更與後端相關的SVM/admin認證資料所致。使用「tridentctl update backend」更新後端資訊、或是退回Trident pod、將可修正此問題。
- 如果在容器執行時間安裝Trident with Docker時遇到權限問題、請嘗試使用「-in cluster =fals」旗標來安裝Trident。這不會使用安裝程式Pod、也不會因為「Trident安裝程式」使用者而造成權限問題。
- 使用「uninstall參數<uninstalling Trident >」來在執行失敗後進行清理。根據預設、指令碼不會移除Trident所建立的客戶需求日、即使在執行中的部署中、也能安全地解除安裝及再次安裝。
- 如果您想要降級至舊版的 Trident、請先執行 tridentctl uninstall 移除Trident的命令。下載所需的 "[Trident版本](#)" 並使用安裝 tridentctl install 命令。
- 成功安裝之後、如果某個永久虛擬磁碟卡在「Pending」（擱置）階段、執行「KECBECTL描述永久虛擬磁碟」可提供有關Trident為何無法為此永久虛擬磁碟配置PV的其他資訊。

使用運算子的 Trident 部署不成功

如果您使用運算子來部署Trident、則「TridentOrchestrator」的狀態會從「安裝」變更為「安裝」。如果您看到「失敗」狀態、而且操作員本身無法恢復、您應該執行下列命令來檢查操作員的記錄：

```
tridentctl logs -l trident-operator
```

追蹤Trident運算子容器的記錄可以指出問題所在。例如、其中一個問題可能是無法從無線環境中的上游登錄擷取所需的容器映像。

若要瞭解Trident安裝失敗的原因、您應該看看「TridentOrchestrator」狀態。

```
kubectl describe torc trident-2
Name:          trident-2
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Image Pull Secrets:      <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:          Trident is bound to another CR 'trident'
  Namespace:       trident-2
  Status:           Error
  Version:
Events:
  Type    Reason  Age                From                    Message
  ----    -
Warning  Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
is bound to another CR 'trident'
```

此錯誤表示已存在用於安裝Trident的「TridentOrchestrator」。由於每個Kubernetes叢集只能有一個Trident執行個體、因此營運者可確保在任何指定時間只存在一個可建立的作用中「TridentOrchestrator」。

此外、觀察Trident Pod的狀態、通常會指出是否有不正確的情況。

```
kubectl get pods -n trident
```

NAME	READY	STATUS	RESTARTS
trident-csi-4p5kq 5m18s	1/2	ImagePullBackOff	0
trident-csi-6f45bfd8b6-vfrkw 5m19s	4/5	ImagePullBackOff	0
trident-csi-9q5xc 5m18s	1/2	ImagePullBackOff	0
trident-csi-9v95z 5m18s	1/2	ImagePullBackOff	0
trident-operator-766f7b8658-ldzsv 8m17s	1/1	Running	0

您可以清楚看到、由於未擷取一或多個容器映像、所以Pod無法完全初始化。

若要解決此問題、您應該編輯「TridentOrchestrator」。或者、您也可以刪除「TridentOrchestrator」、然後使用修改後的準確定義來建立新定義。

使用不成功的 **Trident** 部署 tridentctl

為了協助您找出問題所在、您可以使用「-d」引數再次執行安裝程式、這會開啟偵錯模式、並協助您瞭解問題所在：

```
./tridentctl install -n trident -d
```

在解決此問題之後、您可以依照下列步驟清理安裝、然後再次執行「tridentctl install」命令：

```
./tridentctl uninstall -n trident  
INFO Deleted Trident deployment.  
INFO Deleted cluster role binding.  
INFO Deleted cluster role.  
INFO Deleted service account.  
INFO Removed Trident user from security context constraint.  
INFO Trident uninstallation succeeded.
```

完全移除 **Trident** 和客戶需求日

您可以完全移除 Trident 和所有建立的客戶需求日、以及相關的自訂資源。



此動作無法復原。除非您想要全新安裝 Trident、否則請勿這麼做。若要在不移除客戶需求日的情況下解除安裝 Trident "[解除安裝Trident](#)"、請參閱。

Trident 運算子

若要解除安裝 Trident 、並使用 Trident 操作員完全移除客戶需求日：

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

掌舵

若要解除安裝 Trident 並使用 Helm 完全移除客戶需求日：

```
kubectl patch torc trident --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

<code>tridentctl</code>

若要在使用解除安裝 Trident 後完全移除客戶需求日、請執行以下步驟 tridentctl

```
tridentctl obliviate crd
```

在 Kubernetes 1.26 上使用 **rwX** 原始區塊命名空間時、**NVMe** 節點非分段失敗

如果您執行的是 Kubernetes 1.26 、則當使用含 **rwX** 原始區塊命名空間的 NVMe / TCP 時、節點解除暫存可能會失敗。下列案例提供故障的因應措施。或者、您也可以將 Kubernetes 升級至 1.27 。

已刪除命名空間和 Pod

請考慮將 Trident 託管命名空間（NVMe 持續磁碟區）附加至 Pod 的案例。如果您直接從 ONTAP 後端刪除命名空間、則在嘗試刪除 Pod 之後、取消暫存程序會卡住。此案例不會影響 Kubernetes 叢集或其他功能。

因應措施

從個別節點上卸載持續磁碟區（對應於該命名空間）、然後將其刪除。

封鎖 dataLIFs

```
If you block (or bring down) all the dataLIFs of the NVMe Trident
backend, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
```

. 因應措施

```
開啟 dataLIFs 以還原完整功能。
```

刪除命名空間對應

If you remove the `hostNQN` of the worker node from the corresponding subsystem, the unstaging process gets stuck when you attempt to delete the pod. In this scenario, you cannot run any NVMe CLI commands on the Kubernetes node.

. 因應措施

新增 `hostNQN` 返回子系統。

支援

NetApp以多種方式支援Trident。我們全年無休提供豐富的免費自助支援選項、例如知識庫 (KB) 文章和中和管道。

Trident 支援生命週期

Trident 會根據您的版本提供三個層級的支援。請參閱 "[NetApp 軟體版本支援定義](#)"。

完全支援

Trident 自發行日期起 12 個月內提供完整支援。

有限支援

Trident 自發行日期起 13 至 24 個月內提供有限支援。

自我支援

Trident 文件自發行日期起、提供 25 至 36 個月的版本。

版本	完全支援	有限支援	自我支援
"24.10"	2025 年 10 月	2026 年 10 月	2027 年 10 月
"24.06"	2025 年 6 月	2026 年 6 月	2027 年 6 月
"24.02"	2025 年 2 月	2026 年 2 月	2027 年 2 月
"23.10"	2024 年 10 月	2025 年 10 月	2026 年 10 月
"23.07"	2024 年 7 月	2025 年 7 月	2026 年 7 月
"23.04"	—	2025 年 4 月	2026 年 4 月
"23.01"	—	2025 年 1 月	2026 年 1 月
"22.10."	—	2024 年 10 月	2025 年 10 月

版本	完全支援	有限支援	自我支援
"22.07"	—	2024 年 7 月	2025 年 7 月
"22.04"	—	—	2025 年 4 月
"22.01"	—	—	2025 年 1 月

自我支援

如需疑難排解文章的完整清單，請 ["NetApp知識庫 \(需要登入\)"](#) 參閱。

社群支援

我們的上有一個充滿活力的公共容器使用者社群（包括 Trident 開發人員）["不和通路"](#)。這是您提出專案相關一般問題、並與志同道合的同儕討論相關主題的好地方。

NetApp 技術支援

如需 Trident 的說明、請使用建立支援服務組合 `tridentctl logs -a -n trident`、並將其傳送至 NetApp Support `<Getting Help>`。

以取得更多資訊

- ["Trident 資源"](#)
- ["Kubernetes Hub"](#)

參考資料

Trident 連接埠

深入瞭解 Trident 用於通訊的連接埠。

Trident 連接埠

Trident 透過下列連接埠進行通訊：

連接埠	目的
8443	後端通道HTTPS
8001	Prometheus指標端點
8000	Trident REST伺服器
17546	Trident取消安裝套件所使用的活動/整備度探針連接埠



您可以在安裝期間使用變更活動力/整備度探針連接埠 `--probe-port` 旗標。請務必確認工作節點上的其他程序並未使用此連接埠。

Trident REST API

雖然是與 Trident REST API 互動最簡單的方法、但"[tridentctl命令和選項](#)"您可以視需要直接使用其餘端點。

何時使用REST API

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

為了獲得更好的安全性、在 Pod 內執行時、Trident REST API 預設會限制為 localhost。若要變更此行為、您需要在其 Pod 組態中設定 Trident 的 `-address` 引數。

使用REST API

有關如何調用這些 API 的示例，請傳遞 debug (`-d`) 標誌。如需詳細資訊、請 "[使用 tridentctl 管理 Trident](#)"參閱。

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>
```

取得命名物件的詳細資料。

貼文

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

建立指定類型的物件。

- 需要Json組態才能建立物件。有關每種物件類型的規格、請["使用 tridentctl 管理 Trident"](#)參閱。
- 如果物件已經存在、行為會有所不同：後端會更新現有物件、而其他所有物件類型都會使作業失敗。

刪除

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

刪除命名資源。



與後端或儲存類別相關聯的磁碟區將繼續存在、必須分別刪除。如需詳細資訊、請 ["使用 tridentctl 管理 Trident"](#)參閱。

命令列選項

Trident 為 Trident Orchestrator 提供數個命令列選項。您可以使用這些選項來修改部署。

記錄

-debug

啟用除錯輸出。

-loglevel <level>

設定記錄層級（偵錯、資訊、警告、錯誤、嚴重）。預設為資訊。

Kubernetes

-k8s_pod

使用此選項或 `-k8s_api_server` 以啟用Kubernetes支援。設定此選項會使Trident使用內含Pod的Kubernetes服務帳戶認證、來聯絡API伺服器。這只有當Trident在Kubernetes叢集中以Pod形式執行、且已啟用服務帳戶時才會運作。

-k8s_api_server <insecure-address:insecure-port>

使用此選項或 `-k8s_pod` 啟用 Kubernetes 支援。如果指定、Trident 會使用提供的不安全位址和連接埠、連線至Kubernetes API伺服器。如此一來、Trident 就能部署在 Pod 之外、但它只支援與 API 伺服器的不安全連線。若要安全連線、請在具有選項的 Pod 中部署 Trident `-k8s_pod`。

Docker

-volume_driver <name>

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

-driver_port <port-number>

聆聽此連接埠、而非 UNIX 網域通訊端。

-config <file>

必要；您必須指定後端組態檔案的路徑。

休息

-address <ip-or-host>

指定 Trident 的 REST 伺服器應接聽的位址。預設為localhost。當偵聽localhost並在Kubernetes Pod內部執行時、無法從Pod外部直接存取REST介面。使用 `-address ""` 可讓REST介面從Pod IP位址存取。



Trident REST介面可設定為偵聽、僅適用於127.0.0.1（適用於IPV4）或[:1]（適用於IPV6）。

-port <port-number>

指定 Trident 的 REST 伺服器應接聽的連接埠。預設為8000。

-rest

啟用 REST 介面。預設為true。

Kubernetes和Trident物件

您可以透過讀取和寫入資源物件、使用REST API與Kubernetes和Trident互動。Kubernetes與Trident、Trident與Storage、Kubernetes與儲存設備之間有幾個資源物件、分別是它們之間的關係。其中有些物件是透過Kubernetes進行管理、其他物件則是透過Trident進行管理。

物件如何彼此互動？

瞭解物件、物件的適用範圍及其互動方式、最簡單的方法可能是遵循Kubernetes使用者的單一儲存要求：

1. 使用者會建立一個「PersistentVolume Claim」、要求系統管理員先前設定的Kubernetes「storageClass」中的特定大小的新「PersistentVolume」。
2. Kubernetes「storageClass」可將Trident識別為其資源配置程式、並包含可告知Trident如何為所要求的類別資源配置Volume的參數。
3. Trident查看自己的「儲存類」、其名稱與用來為類別配置磁碟區的「後端」和「儲存類」相符。
4. Trident會在相符的後端上配置儲存設備、並建立兩個物件：Kubernetes的「PersistentVolume」、告訴Kubernetes如何尋找、掛載及處理Volume、以及Trident中保留「PersistentVolume」與實際儲存設備之間關係的Volume。
5. Kubernetes將「PersistentVolume Claim」連結到新的「PersistentVolume」。在執行的任何主機上、包含PersistentVolume的「PersistentVolume Claim」掛載的Pod。
6. 使用者使用指向Trident的「Volume SnapshotClass」建立現有的永久虛擬磁碟的「Volume Snapshot」。
7. Trident會識別與該PVC相關聯的磁碟區、並在其後端建立磁碟區快照。它也會建立「Volume SnapshotContent」、指示Kubernetes如何識別快照。

8. 使用者可以使用「Volume Snapshot」作為來源來建立「PersistentVolume Claim」。
9. Trident會識別所需的快照、並執行建立「PersistentVolume」和「Volume」所需的相同步驟。



如需進一步瞭解Kubernetes物件、我們強烈建議您閱讀 "[持續磁碟區](#)" Kubernetes文件的一節。

Kubernetes PersistentVolumeClaim 物件

Kubernetes 「PersistentVolume Claim」物件是Kubernetes叢集使用者所提出的儲存要求。

除了標準規格之外、Trident還可讓使用者指定下列Volume專屬附註、以覆寫您在後端組態中設定的預設值：

註釋	Volume選項	支援的驅動程式
trident.netapp.io/fileSystem	檔案系統	ONTAP-SAN、solidfire-san 、ONTAP-san經濟型
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ONTAP-NAS、ONTAP-SAN 、solidfire-san、azure-NetApp-Files、GCP-CVS、ONTAP-san經濟型
trident.netapp.io/splitOnClone	分岔OnClone	ONTAP-NAS、ONTAP-SAN
trident.netapp.io/protocol	傳輸協定	任何
trident.netapp.io/exportPolicy	匯出原則	ONTAP-NAS、ONTAP-NAS-經濟型、ONTAP-NAS- Flexgroup
trident.netapp.io/snapshotPolicy	Snapshot原則	ONTAP-NAS、ONTAP-NAS-經濟型、ONTAP-NAS-flexgroup 、ONTAP-SAN
trident.netapp.io/snapshotReserve	Snapshot保留區	ONTAP-NAS、ONTAP-NAS-flexgroup、ONTAP-SAN、GCP-CVS
trident.netapp.io/snapshotDirectory	Snapshot目錄	ONTAP-NAS、ONTAP-NAS-經濟型、ONTAP-NAS- Flexgroup
trident.netapp.io/unixPermissions	unix權限	ONTAP-NAS、ONTAP-NAS-經濟型、ONTAP-NAS- Flexgroup
trident.netapp.io/blockSize	區塊大小	solidfire-san

如果建立的PV具有「刪除」回收原則、則當PV釋出時（亦即使用者刪除PVC時）、Trident會同時刪除PV和備用Volume。如果刪除動作失敗、Trident會將PV標示為這樣、並定期重試該作業、直到成功或手動刪除PV為止。如果PV使用「+Retain +」原則、Trident會忽略它、並假設系統管理員會從Kubernetes和後端進行清理、以便在移除之前備份或檢查磁碟區。請注意、刪除PV並不會導致Trident刪除背板Volume。您應該使用REST API（「tridentctl」）將其移除。

Trident支援使用csi規格建立Volume Snapshot：您可以建立Volume Snapshot、並將其作為資料來源來複製現有的PVCS。如此一來、PV的時間點複本就能以快照形式呈現給Kubernetes。快照可用來建立新的PV。請參閱「[隨需磁碟區快照](#)」、瞭解這項功能的運作方式。

Trident也提供 cloneFromPVC 和 splitOnClone 建立複本的附註。您可以使用這些註釋來複製 PVC、而無需使用 CSI 實作。

以下是一個範例：如果使用者已經有一個名為「mysql」的PVC,則使用者可以使用「trident.netapp.io/cloneFromPVC: mySQL」之類的註解來建立一個名為「mysqlclone」的新PVC.使用此註釋集、Trident會複製對應於mySQL PVC的磁碟區、而非從頭開始配置磁碟區。

請考量以下幾點：

- 我們建議您複製閒置的Volume。
- 一個PVC及其複本應位於相同的Kubernetes命名空間中、且具有相同的儲存類別。
- 有了「ONTAP-NAS」和「ONTAP-SAN」驅動程式、可能需要將「trident.netapp.io/splitOnClone」標註與「trident.netapp.io/cloneFromPVC」一起設定。Trident將trident.netapp.io/splitOnClone設為「true」、將複製的磁碟區從父磁碟區分割出來、因此將複製的磁碟區的生命週期與其父磁碟區完全分離、而犧牲部分儲存效率。如果不將「trident.netapp.io/splitOnClone」設定為「假」、則會減少後端的空間使用量、而犧牲父磁碟區與複製磁碟區之間的相依性、使父磁碟區無法刪除、除非先刪除複本。分割實體複製是合理的做法、是將空的資料庫磁碟區複製到磁碟區及其實體複製環境、以大幅分散差異、而非ONTAP 受益於由NetApp提供的儲存效率。

◦ sample-input 目錄包含用於Trident的PVC定義範例。請參閱 以取得與 Trident Volume 相關的參數和設定的完整說明。

Kubernetes PersistentVolume 物件

Kubernetes 「PersistentVolume」物件代表Kubernetes叢集可用的儲存設備。它的生命週期與使用它的Pod無關。



Trident會建立「PersistentVolume」物件、並根據其所配置的磁碟區、自動在Kubernetes叢集上登錄。您不需要自行管理。

當您建立參照Trident型「StorageClass」的PVC時、Trident會使用對應的儲存類別來配置新的Volume、並針對該Volume登錄新的PV。在設定已配置的Volume和對應的PV時、Trident遵循下列規則：

- Trident會產生Kubernetes的PV名稱、以及用來配置儲存設備的內部名稱。在這兩種情況下、都是確保名稱在其範圍內是唯一的。
- 磁碟區的大小會盡可能接近在室早中所要求的大小、不過視平台而定、磁碟區可能會四捨五入至最接近的可分配數量。

Kubernetes StorageClass 物件

Kubernetes的「StorageClass」物件是以名稱在「PersistentVolume Claims」中指定、以一組內容來配置儲存設備。儲存類別本身會識別要使用的資源配置程式、並根據資源配置程式所瞭解的方式來定義該組內容。

這是需要由系統管理員建立及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」物件看起來像這樣：

```

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專屬的、可告訴Trident如何為類別配置Volume。

儲存類別參數包括：

屬性	類型	必要	說明
屬性	map[string]字串	否	請參閱以下「屬性」一節
storagePools	map[stringList	否	將後端名稱對應至中的儲存資源池清單
其他StoragePools	map[stringList	否	將後端名稱對應至中的儲存資源池清單
排除StoragePools	map[stringList	否	將後端名稱對應至中的儲存資源池清單

儲存屬性及其可能值可分類為儲存資源池選擇屬性和Kubernetes屬性。

儲存資源池選擇屬性

這些參數決定應使用哪些Trident託管儲存資源池來配置特定類型的磁碟區。

屬性	類型	價值	優惠	申請	支援者
媒體 ^{1^}	字串	HDD、混合式、SSD	資源池包含此類型的媒體、混合式表示兩者	指定的媒體類型	ONTAP-NAS、ONTAP-NAS-經濟型、ONTAP-NAS-flexgroup、ONTAP-SAN、solidfire-san
資源配置類型	字串	纖薄、厚實	Pool支援此資源配置方法	指定的資源配置方法	厚：全ONTAP 是邊、薄：全ONTAP 是邊、邊、邊、邊、邊、邊、邊、邊、邊、邊、邊

屬性	類型	價值	優惠	申請	支援者
後端類型	字串	ONTAP-NAS 、ONTAP-NAS- 經濟 型、ONTAP- NAS-flexgroup 、ONTAP- SAN、solidfire- san、GCP- CVS、azure- NetApp-Files 、ONTAP-san經濟	集區屬於此類型 的後端	指定後端	所有驅動程式
快照	布爾	對、錯	集區支援具有快 照的磁碟區	已啟用快照 的Volume	ONTAP-NAS 、ONTAP- SAN、Solidfire- SAN、GCP-CVS
複製	布爾	對、錯	資源池支援複製 磁碟區	已啟用複本 的Volume	ONTAP-NAS 、ONTAP- SAN、Solidfire- SAN、GCP-CVS
加密	布爾	對、錯	資源池支援加密 磁碟區	已啟用加密 的Volume	ONTAP-NAS 、ONTAP-NAS- 經濟型、ONTAP- NAS- FlexGroups、ON TAP-SAN
IOPS	內部	正整數	集區能夠保證此 範圍內的IOPS	Volume保證這 些IOPS	solidfire-san

1：ONTAP Select 不受支援

在大多數情況下、所要求的值會直接影響資源配置、例如、要求完整資源配置會導致資源配置較為密集的Volume。不過、元素儲存資源池會使用其提供的IOPS下限和上限來設定QoS值、而非所要求的值。在此情況下、要求的值僅用於選取儲存資源池。

理想情況下、您可以單獨使用「屬性」來建構儲存設備的品質、以滿足特定類別的需求。Trident會自動探索並選取符合您指定「屬性」的_all_儲存集區。

如果您發現自己無法使用「屬性」來自動選取適合某個類別的資源池、您可以使用「儲存池」和「其他儲存池」參數來進一步精簡資源池、甚至選取特定的資源池集區。

您可以使用「儲存池」參數、進一步限制符合任何指定「屬性」的集區集區集區。換句話說、Trident會使用由「屬性」和「儲存庫」參數所識別的資源池交會來進行資源配置。您可以單獨使用參數、也可以同時使用兩者。

您可以使用「additionalStoragePools」參數來擴充Trident用來資源配置的資源池集區集區集區、而不論「attributes」和「storagePools」參數所選取的任何資源池為何。

您可以使用「排除StoragePools」參數來篩選Trident用於資源配置的資源池集區集區。使用此參數會移除任何相符的集區。

在「儲存池」和「其他儲存池」參數中、每個項目的格式均為「<backender>:<storagePoollist>」、其中「<storagePoollist>」是以逗號分隔的儲存池清單、用於指定的後端。例如、「additionalStoragePools」的值可能會像是「ontapnas_192.168.1.100:solidgr1、aggr2、aggrfire、192.168.1.101:Bronze」。這些清單接受後端值和清單值的regex值。您可以使用「tridentctl Get backend」來取得後端及其資源池的清單。

Kubernetes屬性

這些屬性在動態資源配置期間、不會影響Trident選擇儲存資源池/後端。相反地、這些屬性只會提供Kubernetes持續磁碟區所支援的參數。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	說明	相關驅動因素	Kubernetes版本
FSType	字串	ext4、ext3、xfs	區塊磁碟區的檔案系統類型	solidfire-san、ontap、nap、nap、nas經濟、ontap、nas、flexgroup、ontap、san、ONTAP-san經濟型	全部
owVolume擴充	布林值	對、錯	啟用或停用對增加PVC大小的支援	ONTAP-NAS、ONTAP-NAS-經濟型、ONTAP-NAS-flexgroup、ONTAP-SAN、ONTAP-san經濟型、solidfire-san、gcp-CVS、azure-netapp檔案	1.11+
Volume BindingMode	字串	立即、WaitForFirst消費者	選擇何時進行磁碟區繫結和動態資源配置	全部	1.19 - 1.26

- fsType 參數用於控制SAN LUN所需的檔案系統類型。此外、Kubernetes也會使用的fsType 在儲存類別中、表示檔案系統存在。您可以使用來控制Volume擁有權 fsGroup 只有在下列情況下、Pod的安全內容才會出現 fsType 已設定。請參閱 "[Kubernetes：設定Pod或Container的安全內容](#)" 如需使用設定Volume擁有權的總覽 fsGroup 背景。Kubernetes將套用 fsGroup 只有在下列情況下才会有

- 「FSType」是在儲存類別中設定的。

- PVC存取模式為rwo。

對於NFS儲存驅動程式、檔案系統已存在做為NFS匯出的一部分。為了使用「fsGroup」、儲存類別仍需指定「FSType」。您可以將其設定為「NFS」或任何非null值。

- 請參閱 "[展開Volume](#)" 如需磁碟區擴充的詳細資料、
- Trident安裝程式套裝組合提供多個範例儲存類別定義、可與Trident搭配使用、位於「sham-INPUT /儲存設備類別-* .yaml」。刪除Kubernetes儲存類別也會刪除對應的Trident儲存類別。



Kubernetes VolumeSnapshotClass 物件

Kubernetes的「Volume SnapshotClass」物件類似於「儲存類別」。它們有助於定義多種儲存類別、並由Volume Snapshot參考、以將快照與所需的Snapshot類別建立關聯。每個Volume Snapshot都與單一Volume Snapshot類別相關聯。

系統管理員應定義「Volume SnapshotClass」、以建立快照。建立具有下列定義的Volume Snapshot類別：

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

對Kubernetes而言、「driver」是指Trident處理「Csi-snapClass」類別的Volume快照要求。「刪除原則」指定必須刪除快照時要採取的動作。當「刪除原則」設定為「刪除」時、刪除快照時、就會移除儲存叢集上的Volume Snapshot物件和基礎Snapshot。或者、將其設為「保留」、表示保留「Volume SnapshotContent」和實體快照。

Kubernetes VolumeSnapshot 物件

Kubernetes「Volume Snapshot」物件是建立磁碟區快照的要求。就像使用者針對磁碟區所提出的要求一樣、磁碟區快照是使用者建立現有虛擬磁碟快照的要求。

當磁碟區快照要求出現時、Trident會自動管理後端磁碟區的快照建立、並建立獨特的「Volume SnapshotContent」物件來公開快照。您可以從現有的PVCS建立快照、並在建立新的PVCS時、將快照作為DataSource使用。



Volume Snapshot的生命週期與來源PVCs無關：即使刪除來源PVCs、快照仍會持續存在。刪除具有相關快照的永久虛擬磁碟時、Trident會將此永久虛擬磁碟的備份磁碟區標示為*刪除*狀態、但不會將其完全移除。刪除所有相關的快照時、即會移除該磁碟區。

Kubernetes VolumeSnapshotContent 物件

Kubernetes「Volume SnapshotContent」物件代表從已配置的磁碟區擷取的快照。它類似於「PersistentVolume」、代表儲存叢集上已配置的快照。與「PersistentVolume Claim」和「PersistentVolume」物件類似、建立快照時、「Volume SnapshotContent」物件會維持一對一的對應、以對應「Volume Snapshot」物件、該物件已要求建立快照。

「Volume SnapshotContent」物件包含可唯一識別快照的詳細資料、例如「快照資料」。此「快照處理」是PV名稱與「Volume SnapshotContent」物件名稱的獨特組合。

當快照要求出現時、Trident會在後端建立快照。建立快照之後、Trident會設定「Volume SnapshotContent」物件、並將快照公開給Kubernetes API。



一般而言、您不需要管理「VolumeSnapshotContent」物件。例外情況是您想要[匯入 Volume 快照](#)在 Trident 之外建立。

Kubernetes CustomResourceDefinition 物件

Kubernetes自訂資源是Kubernetes API中由系統管理員定義的端點、用於將類似物件分組。Kubernetes支援建立自訂資源來儲存物件集合。您可以執行「`kubectl Get crds`」來取得這些資源定義。

自訂資源定義 (CRD) 及其相關的物件中繼資料會由Kubernetes儲存在其中繼資料儲存區中。如此一來、您就不需要另外建立Trident的儲存區。

Trident 使用 `CustomResourceDefinition` 物件來保留 Trident 物件的身分識別、例如 Trident 後端、Trident 儲存類別和 Trident Volume。這些物件由Trident管理。此外、「csi Volume Snapshot」架構也引進了定義Volume快照所需的部分CRD。

CRD是Kubernetes建構。上述資源的物件是由Trident所建立。例如、當使用「`tridentctl`」建立後端時、Kubernetes會建立一個對應的「`tridentbackend`」CRD物件供其使用。

以下是Trident客戶需求日的幾點重點：

- 安裝Trident時、會建立一組客戶需求日、並可像使用任何其他資源類型一樣使用。
- 使用解除安裝Trident時 `tridentctl uninstall` 命令、Trident Pod會刪除、但建立的客戶需求日不會清除。請參閱 "[解除安裝Trident](#)" 瞭解如何徹底移除Trident並從頭重新設定。

Trident 物件 StorageClass

Trident為Kubernetes建立相符的儲存類別 `StorageClass` 指定的物件 `csi.trident.netapp.io` 在他們的資源配置工具欄位中。儲存類別名稱與Kubernetes名稱相符 `StorageClass` 所代表的物件。



使用Kubernetes、當Kubernetes「`torageClass`」以Trident做為資源配置程式登錄時、就會自動建立這些物件。

儲存類別包含一組磁碟區需求。Trident會將這些需求與每個儲存資源池中的屬性相符；如果符合、則該儲存資源池是使用該儲存類別來配置磁碟區的有效目標。

您可以使用REST API建立儲存類別組態、以直接定義儲存類別。不過、在Kubernetes部署中、我們預期在登錄新的Kubernetes「`torageClass`」物件時、會建立這些物件。

Trident後端物件

後端代表儲存供應商、其中Trident會配置磁碟區；單一Trident執行個體可管理任何數量的後端。



這是您自己建立和管理的兩種物件類型之一。另一個是Kubernetes的「`torageClass`」物件。

如需如何建構這些物件的詳細資訊、請參閱 "[設定後端](#)"。

Trident 物件 StoragePool

儲存資源池代表可在每個後端上進行資源配置的不同位置。就支援而言ONTAP、這些項目對應於SVM中的集合體。對於NetApp HCI / SolidFire、這些服務會對應到系統管理員指定的QoS頻段。就架構而言、這些項目對應於雲端供應商所在的地區。Cloud Volumes Service每個儲存資源池都有一組獨特的儲存屬性、可定義其效能特性和資料保護特性。

與此處的其他物件不同、儲存資源池候選項目一律會自動探索及管理。

Trident 物件 Volume

Volume是資源配置的基本單位、包含NFS共用和iSCSI LUN等後端端點。在Kubernetes中、這些內容直接對應到「PersistentVolumes」。建立磁碟區時、請確定它有一個儲存類別、決定該磁碟區可以配置的位置及大小。



- 在Kubernetes中、會自動管理這些物件。您可以檢視這些資源、以查看資源配置的Trident內容。
- 刪除具有相關快照的PV時、對應的Trident Volume會更新為*刪除*狀態。若要刪除Trident磁碟區、您應該移除該磁碟區的快照。

Volume組態會定義已配置磁碟區應具備的內容。

屬性	類型	必要	說明
版本	字串	否	Trident API版本（「1」）
名稱	字串	是的	要建立的Volume名稱
storageClass	字串	是的	配置Volume時使用的儲存類別
尺寸	字串	是的	要配置的磁碟區大小（以位元組為單位）
傳輸協定	字串	否	要使用的傳輸協定類型；「檔案」或「區塊」
內部名稱	字串	否	儲存系統上的物件名稱；由Trident產生
cloneSourceVolume	字串	否	Sname (NAS、SAN) & S--*：要複製的磁碟區名稱ONTAP SolidFire
分岔OnClone	字串	否	例 (NAS、SAN)：從父實體分割複本ONTAP
Snapshot原則	字串	否	S--*：快照原則ONTAP
Snapshot保留區	字串	否	Sing-*：保留給快照的磁碟區百分比ONTAP
匯出原則	字串	否	ONTAP-NAS*：要使用的匯出原則
Snapshot目錄	布爾	否	ONTAP-NAS*：快照目錄是否可見
unix權限	字串	否	ONTAP-NAS*：初始UNIX權限
區塊大小	字串	否	S--*：區塊/區段大小SolidFire
檔案系統	字串	否	檔案系統類型

Trident在建立磁碟區時會產生「內部名稱」。這包括兩個步驟。首先、它會將儲存前置詞（預設的「Trident」或後端組態中的前置詞）預先加上磁碟區名稱、以「<prefix>-<volume名稱>」格式命名。然後、它會繼續清理名稱、取代後端不允許的字元。對於後端、它會以底線取代連字號（因此內部名稱會變成「<prefix>_<volume名稱>」）ONTAP。對於元素後端、它會以連字號取代底線。

您可以使用Volume組態、使用REST API直接配置磁碟區、但在Kubernetes部署中、我們預期大多數使用者都會使用標準的Kubernetes「PersistentVolume Claim」方法。Trident會自動建立此Volume物件、做為資源配置程序的一部分。

Trident 物件 Snapshot

快照是磁碟區的時間點複本、可用來配置新的磁碟區或還原狀態。在Kubernetes中、這些物件會直接對應到「Volume SnapshotContent」物件。每個快照都與一個Volume相關聯、該磁碟區是快照資料的來源。

每個「snapshot」物件都包含下列內容：

屬性	類型	必要	說明
版本	字串	是的	Trident API版本（「1」）
名稱	字串	是的	Trident Snapshot物件的名稱
內部名稱	字串	是的	儲存系統上Trident Snapshot物件的名稱
Volume名稱	字串	是的	為其建立快照的持續Volume名稱
Volume內部名稱	字串	是的	儲存系統上相關Trident Volume物件的名稱



在Kubernetes中、會自動管理這些物件。您可以檢視這些資源、以查看資源配置的Trident內容。

當Kubernetes「Volume Snapshot」物件要求建立時、Trident會在備份儲存系統上建立Snapshot物件。此快照物件的「內部名稱」是將前置詞「sfapshot-」與「Volume Snapshot」物件的「UID」（例如、「sfapshot-e8d8a0ca-9826-11e9-9807-525400f3f660」）結合在一起產生的。「Volume Name」（Volume名稱）和「Volume InternalName」（磁碟區內部名稱）會透過取得備用磁碟區的詳細資料來填入資料。

Trident 物件 ResourceQuota

Trident 去除會使用優先順序類別（Kubernetes 中可用的最高優先順序類別）、以確保 Trident 能在正常節點關機期間識別及清理磁碟區、並允許 Trident 去「system-node-critical」除設定群組在資源壓力較大的叢集中、以較低的優先順序來搶佔工作負載。

為達成此目標、Trident 採用「ResourceQuota」物件來確保 Trident 標章集上的「系統節點關鍵」優先順序類別獲得滿足。在建立部署和取消設定集之前、Trident 會先尋找物件、如果未發現、則會套用該「ResourceQuota」物件。

如果您需要對預設資源配額和優先順序類別的更多控制權、可以產生「custustry.yaml」、或使用Helm圖表來設定「資源配額」物件。

以下是「資源配額」物件優先處理Trident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：資源配額"](#)。

清理 ResourceQuota 如果安裝失敗

在極少數情況下、如果在建立「資源配額」物件之後安裝失敗、請先嘗試 ["正在解除安裝"](#) 然後重新安裝。

如果這不管用、請手動移除「資源配額」物件。

移除 ResourceQuota

如果您偏好控制自己的資源配置、可以使用下列命令移除 Trident ResourceQuota 物件：

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

Pod安全標準（PSS）與安全內容限制（SCC）

Kubernetes Pod安全標準（Ps）和Pod安全政策（Ps）定義權限等級、並限制Pod的行為。OpenShift Security內容限制（SCC）同樣定義OpenShift Kubernetes Engine特有的Pod限制。為了提供此自訂功能、Trident 會在安裝期間啟用特定權限。下列各節詳細說明 Trident 所設定的權限。



PSS-取代Pod安全性原則（PSP）。在Kubernetes v1.21中、已不再使用PSP、將在v1.25中移除。如需詳細資訊、請參閱 ["Kubernetes：安全性"](#)。

必要的Kubernetes安全內容和相關欄位

權限	說明
權限	SCSI需要雙向裝載點、這表示Trident節點Pod必須執行特殊權限容器。如需詳細資訊、請參閱 "Kubernetes：掛載傳播" 。

權限	說明
主機網路	iSCSI精靈所需。「iscsiadm」管理iSCSI掛載、並使用主機網路來與iSCSI精靈通訊。
主機iPC	NFS使用程序間通訊 (IPC) 與nfsd通訊。
主機PID	啟動 NFS 所需 rpc-statd。Trident 會查詢主機處理程序、以判斷在掛載 NFS 磁碟區之前是否 `rpc-statd` 正在執行。
功能	「SYS-ADMIN」功能是專為特殊權限容器提供的預設功能之一。例如、Docker為特殊權限容器設定了這些功能：「CapPm:0000003fffffff」、「CapEff:0000003fffffff」
Seccomp	Seccomp 設定檔在特殊權限的容器中一律為「未限制」、因此無法在 Trident 中啟用。
SELinux	在 OpenShift 上、權限容器會在（「超級貴賓 Container」）網域中執行 <code>spc_t</code> 、而非權限容器則會在網域中執行 <code>container_t</code> 。在上 <code>containerd</code> 、安裝後 <code>container-selinux</code> 、所有容器都會在網域中執行 <code>spc_t</code> 、這會有效停用 SELinux。因此、Trident 不會新增 `seLinuxOptions` 至容器。
DAC	權限容器必須以root身分執行。非權限容器會以root身分執行、以存取csi所需的UNIX通訊端。

Pod安全標準 (PSS)

標籤	說明	預設
"pod安全性.Kubernetes.io/enforce (pod安全性) 。Kubernetes.io/enforce版本	允許Trident控制器和節點進入安裝命名空間。請勿變更命名空間標籤。	「enforce：特權」的「enforce version：<目前叢集的版本或通過測試的最高版本的PSS>。」



變更命名空間標籤可能會導致無法排程Pod、「建立錯誤：...」或「警告：Trident：Cig-...」。如果發生這種情況、請檢查「特殊權限」的命名空間標籤是否已變更。如果是、請重新安裝Trident。

Pod安全原則 (PSP)

欄位	說明	預設
「允許升級」	特殊權限容器必須允許權限提高。	"真的"
《分配CSIDriver》	Trident不使用即時的csi暫時性磁碟區。	空白
《分配能力》	非權限Trident容器不需要比預設集更多的功能、而且會將所有的功能授予權限容器。	空白

欄位	說明	預設
《分配FlexVolumes》	Trident並未使用 "FlexVolume驅動程式"因此，它們不會包含在允許的磁碟區清單中。	空白
《主機路徑》	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
《處理器類型》	Trident不使用任何「ProctMountTypes」。	空白
《非安全性系統》	Trident不需要任何不安全的「縮圖」。	空白
'資料錯誤附加功能'	不需要將任何功能新增至權限容器。	空白
「DefaultAllowPrivilegeEscalation」	每個Trident Pod都會處理允許權限提高的問題。	「假」
《ForbiddenSysctls》	不允許使用"sysctls"。	空白
「fsGroup」	Trident容器以root執行。	《RunAsAny》
《hostipc》	掛載NFS磁碟區需要主機IPC與"nfsd"通訊	"真的"
「主機網路」	iscsiadm要求主機網路與iSCSI精靈進行通訊。	"真的"
"hostPID"	需要主機PID來檢查節點上是否正在執行「rps-statd」。	"真的"
"hostPortes"	Trident不使用任何主機連接埠。	空白
"特權"	Trident節點Pod必須執行特殊權限容器、才能掛載磁碟區。	"真的"
《ReadOnlyRootFilesystem》	Trident節點Pod必須寫入節點檔案系統。	「假」
《requiredropCapabilities》	Trident節點Pod執行特殊權限容器、無法丟棄功能。	無
《RunAsGroup》（《RunAsGroup》）	Trident容器以root執行。	《RunAsAny》
「RunAsUser」	Trident容器以root執行。	「RunAsAny」
《RuntimeClass》	Trident不使用「RuntimeClass」。	空白
「eLinux」	Trident並未設定「最新Linux選項」、因為目前容器執行時間與Kubernetes發行版本處理SELinux的方式有所不同。	空白
《支援團體》	Trident容器以root執行。	《RunAsAny》
《Volume》（Volume）	Trident Pod需要這些Volume外掛程式。	《hostPath》、《Project預計》、《emptyDir.》

安全內容限制 (SCC)

標籤	說明	預設
"owHostDirVolume Plugin"	Trident節點Pod會掛載節點的根檔案系統。	"真的"
"owhostipc"	掛載NFS磁碟區需要主機IPC與"nfsd"通訊。	"真的"
「允許主機網路」	iscsiadm要求主機網路與iSCSI精靈進行通訊。	"真的"
"owhostpid"	需要主機PID來檢查節點上是否正在執行「rps-stdt」。	"真的"
"allowHostPort"	Trident不使用任何主機連接埠。	「假」
「允許升級」	特殊權限容器必須允許權限提高。	"真的"
《允許使用容器》	Trident節點Pod必須執行特殊權限容器、才能掛載磁碟區。	"真的"
《非安全性系統》	Trident不需要任何不安全的「縮圖」。	無
《分配能力》	非權限Trident容器不需要比預設集更多的功能、而且會將所有的功能授予權限容器。	空白
'資料錯誤附加功能'	不需要將任何功能新增至權限容器。	空白
「fsGroup」	Trident容器以root執行。	《RunAsAny》
《團體》	此SCC僅適用於Trident、並與其使用者有關。	空白
《ReadOnlyRootFilesystem》	Trident節點Pod必須寫入節點檔案系統。	「假」
《requiredDropCapabilities》	Trident節點Pod執行特殊權限容器、無法丟棄功能。	無
「RunAsUser」	Trident容器以root執行。	《RunAsAny》
「Linux轉換」	Trident並未設定「最新Linux選項」、因為目前容器執行時間與Kubernetes發行版本處理SELinux的方式有所不同。	空白
「eccompProfiles」	特殊權限容器永遠都會執行「未限制」。	空白
《支援團體》	Trident容器以root執行。	《RunAsAny》
《使用者》	提供一個項目來將此SCC繫結至Trident命名空間中的Trident使用者。	不適用
《Volume》 (Volume)	Trident Pod需要這些Volume外掛程式。	《hostPath、DownwardAPI、Project預計、emptyDir》

法律聲明

法律聲明提供版權聲明、商標、專利等存取權限。

版權

["https://www.netapp.com/company/legal/copyright/"](https://www.netapp.com/company/legal/copyright/)

商標

NetApp、NetApp 標誌及 NetApp 商標頁面上列出的標章均為 NetApp、Inc. 的商標。其他公司與產品名稱可能為其各自所有者的商標。

["https://www.netapp.com/company/legal/trademarks/"](https://www.netapp.com/company/legal/trademarks/)

專利

如需最新的 NetApp 擁有專利清單、請參閱：

<https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf>

隱私權政策

["https://www.netapp.com/company/legal/privacy-policy/"](https://www.netapp.com/company/legal/privacy-policy/)

開放原始碼

您可以在每個版本的通知檔中、檢閱 NetApp 軟體 for Trident 中使用的協力廠商版權和授權、網址為：
<https://github.com/NetApp/trident/>。

版權資訊

Copyright © 2024 NetApp, Inc. 版權所有。台灣印製。非經版權所有人事先書面同意，不得將本受版權保護文件的任何部分以任何形式或任何方法（圖形、電子或機械）重製，包括影印、錄影、錄音或儲存至電子檢索系統中。

由 NetApp 版權資料衍伸之軟體必須遵守下列授權和免責聲明：

此軟體以 NETAPP「原樣」提供，不含任何明示或暗示的擔保，包括但不限於有關適售性或特定目的適用性之擔保，特此聲明。於任何情況下，就任何已造成或基於任何理論上責任之直接性、間接性、附隨性、特殊性、懲罰性或衍生性損害（包括但不限於替代商品或服務之採購；使用、資料或利潤上的損失；或企業營運中斷），無論是在使用此軟體時以任何方式所產生的契約、嚴格責任或侵權行為（包括疏忽或其他）等方面，NetApp 概不負責，即使已被告知有前述損害存在之可能性亦然。

NetApp 保留隨時變更本文所述之任何產品的權利，恕不另行通知。NetApp 不承擔因使用本文所述之產品而產生的責任或義務，除非明確經過 NetApp 書面同意。使用或購買此產品並不會在依據任何專利權、商標權或任何其他 NetApp 智慧財產權的情況下轉讓授權。

本手冊所述之產品受到一項（含）以上的美國專利、國外專利或申請中專利所保障。

有限權利說明：政府機關的使用、複製或公開揭露須受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中的「技術資料權利 - 非商業項目」條款 (b)(3) 小段所述之限制。

此處所含屬於商業產品和 / 或商業服務（如 FAR 2.101 所定義）的資料均為 NetApp, Inc. 所有。根據本協議提供的所有 NetApp 技術資料和電腦軟體皆屬於商業性質，並且完全由私人出資開發。美國政府對於該資料具有非專屬、非轉讓、非轉授權、全球性、有限且不可撤銷的使用權限，僅限於美國政府為傳輸此資料所訂合約所允許之範圍，並基於履行該合約之目的方可使用。除非本文另有規定，否則未經 NetApp Inc. 事前書面許可，不得逕行使用、揭露、重製、修改、履行或展示該資料。美國政府授予國防部之許可權利，僅適用於 DFARS 條款 252.227-7015(b)（2014 年 2 月）所述權利。

商標資訊

NETAPP、NETAPP 標誌及 <http://www.netapp.com/TM> 所列之標章均為 NetApp, Inc. 的商標。文中所涉及的所有其他公司或產品名稱，均為其各自所有者的商標，不得侵犯。