Trident 26.10 文件

Trident

NetApp
November 10, 2026

This PDF was generated from https://docs.netapp.com/zh-tw/trident/index.html on November 10, 2026.
Always check docs.netapp.com for the latest.

目錄

Trident 26.10 文件 1
版本資訊 2
新功能 2
26.10.1 版本新增內容 2
26.10 版本新增內容 2
26.06.2 版本新增內容 5
26.06.1 版本新增內容 6
26.06 版本新增內容 6
26.02.1 版本新增內容 9
26.02 版本新增內容 10
25.10 版本新增內容 12
25.06.2 中的變更 14
25.06.1 中的變更 14
25.06 中的變化 15
25.02.1 的變更 16
25.02 的變更 17
24.10.1 的變更 18
24.10 的變更 19
24.06 的變更 20
24.02 的變更 21
23.10 的變更 22
23.07.1 的變更 22
2007 年 23 月 23 日的變更 22
23.04 年的變更 23
23.01.1 的變更 24
23.01年的變更 24
22.10的變更 25
22.07年的變動 26
22.04年的變化 27
22.01.1中的變更 28
22.01.0版的變更 28
210.1的變更 29
21.0 29
已知問題 30
如需詳細資訊、請參閱 31
較早版本的文件 31
NetApp Trident 對 ONTAP ASA r2 儲存系統的支援 31
支援的作業 32
不支援的作業 32

已知問題 32
從 Trident 22.10 或更早版本升級後，Volume 仍保持附加狀態 32
VolumeSnapshots 未達 ReadyToUse 狀態 32
還原大型檔案的還原還原備份可能會失敗 33
開始使用 34
瞭解 Trident 34
瞭解 Trident 34
Trident 架構 35
概念 38
Trident 快速入門 41
接下來呢？ 42
Trident 需求 42
Trident 的重要資訊 42
支援的前端（協調器） 43
支援的後端（儲存） 43
Trident 支援 KubeVirt 和 OpenShift 虛擬化 44
Trident 對 Kubernetes 上的 Kubernetes 和 OpenShift 託管控制平面的支援 44
功能需求 45
已測試的主機作業系統 45
主機組態 46
儲存系統組態 46
Trident 連接埠 46
Container映像和對應的Kubernetes版本 46
安裝Trident 48
瞭解 Trident 安裝 48
關於 Trident 26.10 的關鍵資訊 48
開始之前 48
選擇您的安裝方法 48
選擇安裝模式 50
根據您的方法和模式選取程序 50
在安裝方法之間移動 51
其他已知組態選項 51
使用Trident操作員安裝 51
手動部署Trident運算子（標準模式） 51
手動部署Trident運算子（離線模式） 56
使用Helm部署Trident運算子（標準模式） 61
使用Helm部署Trident運算子（離線模式） 68
自訂Trident操作員安裝 75
使用tridentctl安裝 85
使用 tridentctl 安裝 Trident 85
自訂 Trident tridentctl 安裝 89

使用 OpenShift 認證營運商進行安裝 89
使用 OpenShift OperatorHub 安裝 Trident 89
切換到 OpenShift 認證的 Trident 操作員 93
使用 Trident 97
準備 Trident 工作節點 97
選擇適當的工具 97
節點服務探索 97
NFS磁碟區 98
iSCSI磁碟區 98
NVMe / TCP 磁碟區 102
FC 磁碟區上的 SCSI 102
準備配置SMB磁碟區 104
使用 COSI 佈建物件儲存 105
總覽 106
需求與限制 106
啟用 COSI 106
設定 StorageGRID 物件存放區 107
設定 ONTAP S3 物件存放區 108
建立 BucketClass 並申請儲存桶 108
授予工作負載對儲存桶的存取 109
設定及管理後端 110
設定 Trident 後端 110
Azure NetApp Files 110
Google Cloud NetApp Volumes 126
設定NetApp HCI 一個不只是功能的SolidFire 後端 162
支援SAN驅動程式ONTAP 167
ASNAS驅動程式ONTAP 193
使用 ONTAP 後端進行均衡放置 223
Amazon FSX for NetApp ONTAP 產品 226
使用kubecl建立後端 257
管理後端 262
建立及管理儲存類別 271
建立 Trident 儲存設備類別 271
管理 Trident 儲存設備類別 274
資源配置與管理磁碟區 275
配置 Trident Volume 275
移動磁碟區 278
擴充 Trident Volume 281
了解 RWX NVMe 子系統限制 291
控制器擴充性 292
自動磁碟區擴充 297

設定 ONTAP Volume 自動調整規模 303
管理 Autogrow 策略 307
匯入 Trident Volume 316
自訂 Trident Volume 名稱和標籤 327
跨命名空間共享 Trident NFS Volume 329
跨命名空間複製 Trident Volume 333
使用 SnapMirror 複寫磁碟區 335
使用「csi拓撲」 341
使用 Trident 快照 348
使用 Trident Volume 群組快照 355
管理及監控 Trident 360
升級 Trident 360
升級 Trident 360
與營運者一起升級 361
使用 tridentctl 升級 Trident 365
使用 tridentctl 管理 Trident 366
命令和全域旗標 366
命令選項和旗標 367
外掛程式支援 372
監控 Trident 儲存設備作業 372
總覽 372
步驟1：定義Prometheus目標 372
步驟2：建立Prometheus ServiceMonitor 373
步驟3：使用PromQL查詢Trident度量 374
瞭解 Trident AutoSupport 遙測 375
停用 Trident 計量 376
解除安裝Trident 376
確定原始安裝方法 377
解除安裝 Trident 運算子安裝 377
解除安裝 `tridentctl` 安裝 378
Trident for Docker 379
部署的先決條件 379
驗證需求 379
NVMe 工具 381
FC工具 381
部署 Trident 383
Docker託管外掛程式方法（1.1/17.03版及更新版本） 383
傳統方法（1.12版或更早版本） 385
升級或解除安裝Trident 388
升級 388
解除安裝 389

使用Volume . . . 390
建立Volume . . . 390
移除Volume . . . 390
複製磁碟區 . . . 391
存取外部建立的磁碟區 . . . 392
驅動程式專屬的Volume選項 . . . 392
收集記錄 . . . 396
收集記錄以進行疑難排解 . . . 396
一般疑難排解秘訣 . . . 397
管理多個 Trident 執行個體 . . . 397
Docker託管外掛程式（1.3/17.03版或更新版本）的步驟 . . . 397
傳統的步驟（1.12版或更早版本） . . . 398
儲存組態選項 . . . 398
全域組態選項 . . . 398
組態ONTAP . . . 399
元件軟體組態 . . . 406
已知問題與限制 . . . 407
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。 . . . 407
Volume名稱長度必須至少2個字元。 . . . 407
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。 . . . 408
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 . . . 408
最佳實務做法與建議 . . . 409
Trident 部署建議 . . . 409
部署至專屬命名空間 . . . 409
使用配額和範圍限制來控制儲存使用量 . . . 409
Trident 儲存組態最佳實務 . . . 409
平台總覽 . . . 409
最佳實務做法ONTAP Cloud Volumes ONTAP . . . 409
最佳實務做法SolidFire . . . 413
哪裡可以找到更多資訊？ . . . 414
將 Trident 與 Kubernetes 整合 . . . 415
驅動程式選擇與部署 . . . 415
儲存層級設計 . . . 418
虛擬資源池設計 . . . 418
Volume作業 . . . 419
度量服務 . . . 422
Trident 資料保護與災難恢復 . . . 423
Trident 複寫與還原 . . . 423
SVM 複寫與還原 . . . 424

Volume 複寫與還原 425
Snapshot 資料保護 425
使用Trident實現有狀態應用程式的故障轉移自動化 425
強制分離的詳細資料 425
有關自動故障轉移的詳細信息 426
安全性 430
Trident 安全性建議 430
Trident Linux 統一金鑰設定 （LUKS） 431
修正 RWX Linux 統一金鑰設定（LUKS）Volume 卡住的租約 437
Trident Kerberos 傳輸中加密 439
使用Trident Protect 保護應用程式 447
了解Trident Protect 447
接下來呢？ 447
安裝Trident Protect 447
Trident保護要求 447
安裝並設定Trident Protect 450
安裝Trident Protect CLI 插件 453
自訂Trident Protect 安裝 456
管理Trident Protect 461
管理Trident Protect 授權和存取控制 461
監控Trident保護資源 467
產生Trident Protect 支援包 471
升級Trident保護 472
管理及保護應用程式 474
使用Trident Protect AppVault 物件來管理儲存桶。 474
加密 Trident Protect 快照與備份的 AppVault 中繼資料 485
使用Trident Protect 定義管理應用程式 491
使用Trident Protect 保護應用程式 496
搭配 Trident Protect 使用 Volume 群組快照 505
還原應用程式 509
使用 NetApp SnapMirror 複寫應用程式 529
使用Trident Protect 遷移應用程式 545
使用 Trident Protect 保護 KubeVirt 虛擬機器 548
管理Trident Protect 執行鉤子 559
解除安裝Trident Protect 566
Trident和Trident Protect 博客 567
Trident 部落格 567
Trident Protect博客 567
知識與支援 569
常見問題集 569
一般問題 569

在 Kubernetes 叢集上安裝及使用 Trident 569
疑難排解與支援 570
升級 Trident 571
管理後端和磁碟區 571
疑難排解 575
一般疑難排解 575
使用運算子的 Trident 部署不成功 576
使用不成功的 Trident 部署 `tridentctl` 578
完全移除 Trident 和客戶需求日 578
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 579
當預期啟用“v4.2-xattrs”時，NFSv4.2 用戶端在升級ONTAP後報告“無效參數” 580
支援 580
Trident 支援生命週期 580
自我支援 581
社群支援 581
NetApp 技術支援 581
以取得更多資訊 581
參考資料 582
Trident 連接埠 582
總覽 582
Trident REST API 584
何時使用REST API 584
使用REST API 584
命令列選項 585
記錄 585
Kubernetes 585
Docker 585
休息 585
Kubernetes和Trident物件 586
物件如何彼此互動？ 586
Kubernetes `PersistentVolumeClaim` 物件 586
Kubernetes `PersistentVolume` 物件 588
Kubernetes `StorageClass` 物件 588
Kubernetes `VolumeSnapshotClass` 物件 591
Kubernetes `VolumeSnapshot` 物件 592
Kubernetes `VolumeSnapshotContent` 物件 592
Kubernetes `VolumeGroupSnapshotClass` 物件 592
Kubernetes `VolumeGroupSnapshot` 物件 593
Kubernetes `VolumeGroupSnapshotContent` 物件 593
Kubernetes `CustomResourceDefinition` 物件 593

Trident 物件 StorageClass . . . 594
Trident後端物件 . . . 594
Trident 物件 StoragePool . . . 594
Trident 物件 Volume . . . 594
Trident 物件 Snapshot . . . 595
Trident 物件 ResourceQuota . . . 596
Pod安全標準（PSS）與安全內容限制（SCC） . . . 597
必要的Kubernetes安全內容和相關欄位 . . . 597
Pod安全標準（PSS） . . . 598
Pod安全原則（PSP） . . . 598
安全內容限制（SCC） . . . 599
法律聲明 . . . 601
版權 . . . 601
商標 . . . 601
專利 . . . 601
隱私權政策 . . . 601
開放原始碼 . . . 601

Trident 26.10 文件

版本資訊

新功能

發行說明提供有關 NetApp Trident 最新版本中的新功能、增強功能和錯誤修復的資訊。

> 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。

26.10.1 版本新增內容

了解 NetApp Trident 和 Trident Protect 的新特性，包括增強功能、修復和棄用。

Trident

修正

- * Kubernetes* ：
  - 修正了從 Trident 25.10 或更早版本升級後，Trident 節點 pod 在啟動時強制分離使用中磁碟區的問題，此問題會導致 pod 失敗並顯示 `volume … is not staged`。

26.10 版本新增內容

了解 NetApp Trident 和 Trident Protect 的新特性，包括增強功能、修復和棄用。

> 請勿升級到 Trident 26.10.0。在啟動過程中，Trident 節點 Pod 可能會分離仍在使用的磁碟區。受影響的 Pod 可能會失敗並顯示錯誤 `volume … is not staged`。此問題已在 Trident 26.10.1 中修復。請改為升級到 Trident 26.10.1 或更新版本。

Trident

增強功能

- * Kubernetes* ：
  - 新增支援從 StorageGRID 進行以 COSI 為基礎的物件儲存桶動態配置。如需詳細資訊，請參閱 "使用 COSI 佈建物件儲存"。
  - 新增了對定期重新整理 NVMe 資料 LIF 的支援，以偵測並適應組態變更。如需詳細資訊，請參閱 "重新整理 NVMe 資料 LIFs"。
  - 新增了對 ONTAP-SAN 和 ONTAP-SAN-Economy 驅動程式配置的新 Volume 的 ONTAP 自動調整規模功能的支援。如需詳細資訊，請參閱 "設定 ONTAP Volume 自動調整規模"。
  - 新增對 Kubernetes 1.37 的支援。如需支援的版本與所需的映像，請參閱"Trident 需求"。
  - 新增了對 CommVault 和 Rubrik 備份的備份 Clone PVC 識別，以跳過 ONTAP Volume 恢復佇列（問題 "#1182"）。
  - 為新的 NVMe RWX Volume 新增了 `maxNamespacesPerSuperSubsystem` 後端選項。如需詳細資訊，請參閱 "NVMe RWX 子系統限制"。

- 透過 `TridentNode`規格與狀態新增了以 CRD 為基礎的 CSI 節點註冊，取代了 Kubernetes 部署的 HTTPS 控制器後端通道。
- 新增了安裝程式支援，以便在 OpenShift 上為 Google Cloud Workload Identity Federation （WIF） 投影具有可設定對象的服務帳戶權杖。如需安裝與後端組態的詳細資訊，請參閱 "設定 Google Cloud NetApp Volumes 與工作負載身分聯盟"。
- 已將 `backend_state`標籤新增至 `trident_backend_info`計量，以識別 Prometheus 中的個別後端健全狀況（問題 "#1088"）。如需 Trident 計量的相關資訊，請參閱 "監控 Trident"。

- 新增對由 ONTAP-SAN 驅動程式透過 iSCSI 配置的 RWX Volume 的 LUKS 加密支援。如需詳細資訊，請參閱 "使用Linux統一金鑰設定（LUKS）"。
- 新增對 ONTAP-NAS-FlexGroup 驅動程式和 Docker 的 GA 並行支援；ASA r2 並行功能現已 GA，從技術預覽版升級。如需支援的驅動程式和行為，請參閱 "控制器擴充性"。
- 新增了對 CSI Volume 擴充的支援，當並非所有 iSCSI 路徑都可用時，可以使用路徑移除和 iSCSI 自我修復。如需詳細資訊，請參閱 "展開Volume"。
- 新增了 `trident.netapp.io/unixGroupID` PVC 註釋，以便在建立具有 root squash 的 ONTAP-NAS 和 ONTAP-NAS-FlexGroup Volume 時，設定 ONTAP Volume UNIX 群組 ID（問題 "#1079"）。
- 新增 Talos OS 對 NVMe/TCP、NFSv4 和 iSCSI 的支援（問題 "#806"）。如需支援的平台與工作節點要求，請參閱 "Trident 需求" 和 "準備工作節點"。
- 新增了可選的 `storageAPITimeout`後端參數，用於設定每個後端的 ONTAP REST 和 ZAPI 用戶端逾時。
- 新增了 `useRESTSQLBypass` ONTAP 後端選項，以新增 `SQL=false`至 ONTAP REST 名稱查詢。
- 已將 ONTAP-NAS-Economy qtree 匯入功能從技術預覽版升級到正式版。如需匯入說明，請參閱 "匯入 Trident Volume"。

修正

- * Kubernetes* ：
  - 修正當 `enableConcurrency` 為 `true` 時，在後端更新或控制器重新啟動期間， ONTAP-NAS 匯出原則規則被錯誤刪除的問題（問題 "#1166" 、 "#1179" ）。如需匯出原則組態，請參閱 "列舉NAS組態選項與範例ONTAP" 。
  - 修正了非實質後端更新（例如認證資料或 QoS 變更）導致的每個 Volume 孤立掃描問題，這些掃描可能會區塊大型後端上的 CSI 操作。現在，掃描僅在 SVM 或儲存設備前綴發生變更時執行。
  - 修正了偵錯日誌中出現的認證資料，包括 CSI gRPC 請求傾印、啟動環境輸出和已記錄的 `TridentBackendConfig`規格。啟動輸出現在只記錄 `Trident` `TRIDENT_*` 變數的值，並按名稱列出其他變數。
  - 修正了 CSI 發布和 Volume 移動之間的競爭條件，該條件可能導致附加具有過期存取資訊的 Volume。如需 Volume 移動的詳細資訊，請參閱 "移動磁碟區"。
  - 修正了 `NodeGetInfo`在節點註冊完成之前傳回空拓撲標籤的問題，這可能導致拓撲感知資源配置失敗（問題 "#1131"）。關於拓撲感知資源配置，請參閱 "使用 CSI 拓撲資源配置儲存設備"。
  - 修復了罕見的 NVMe 中斷連線競爭問題，該問題可能導致在另一個 pod 附加時，共享的子系統被拆除。
  - 修正了在並行情況下 Clone 掛載 Volume 時，NVMe PVC Clone 發布失敗和父命名空間對應錯誤的問題。
  - 修正了在設定 `nameTemplate` 的 ONTAP-SAN-Economy 後端上 PVC 快照建立失敗的問題。如需後端組態範例，請參閱 "SAN組態選項與範例ONTAP"。
  - 修正 ONTAP REST 後端在 REST 名稱索引更新之前刪除新建立的 FlexVol 時，會使其變成孤立的問題

；Trident 現在依 UUID 刪除 Volume。

- 修正後端 LIF 無法連線時控制器啟動停滯的問題。受影響的後端會被標記 `Failed`，並繼續啟動。
- 修正了設定多鍵節點選擇器時，Operator 代管的控制器和節點 Pod 在每次協調時都會重新啟動的問題。有關節點選擇器組態，請參閱 "自訂Trident操作員安裝"。
- 修正了 TridentConfigurator 建立的 Azure NetApp Files `TridentBackendConfig` 資源缺少 `trident.netapp.io/configurator` 建立後驗證所需標籤的問題。
- 修正當 Trident CSI 服務使用 IPv6 ClusterIP 時節點註冊失敗的問題（問題 "#1164"）。
- 修正了當過期 Volume 發布仍在取消發佈時，刪除並重新加入工作節點會導致 Trident 節點永久移除的問題。
- 修正了當 Trident 快取的匯出原則過期時， ONTAP-NAS Volume 在已附加的節點上遺失 NFS 匯出存取的問題。如需匯出原則組態，請參閱 "列舉NAS組態選項與範例ONTAP"。
- 修正了 `tridentctl`安裝清理過程中刪除節點修復範本 CRD 的問題，這可能會導致啟用強制分離的後續安裝失敗。如需安裝詳細資料，請參閱 "使用 tridentctl 安裝 Trident"。
- 透過新增重試延遲，修復了快照刪除呼叫導致 ONTAP API 過載的問題。有關快照操作，請參閱 "管理快照"。
- 修復了 Trident 營運者 Helm chart 的 AutoSupport 映像輔助程式註解與 cloudProvider 值引用。
- 修正了針對繁忙或現有工作的 ONTAP ZAPI Volume 建立處理，以免 CSI 重試過早毀損 FlexVols 或導致建立/毀損循環。
- 修正了重試已不存在的一致性群組的 Volume 群組快照時發生的 panic 問題。如需 Volume 群組快照的詳細資訊，請參閱 "建立卷宗組快照"。
- 根據支援的最低和最高 Kubernetes 版本，移除了 Trident 支援警告。

- 修正了共享的命名鎖定中可能允許對相同資源進行重疊操作的競爭條件。
- 修正了找不到快照時 ASA 驅動程式初始化失敗的問題。
- 修正了當叢集節點不在主要 HA 區域時，針對區域 Flex Unified 資源池的 GCNV 資源配置問題；Trident 不再將該 HA 區域視為必要的 CSI 拓撲。分區資源池保持不變。如需 Flex Unified 組態的詳細資訊，請參閱 "設定區塊儲存設備（iSCSI）"。
- 修正了當規範後端識別與 Trident Volume 名稱不同時，GCNV Flex Unified NAS 和 SAN Volume 在後端協調期間被錯誤標記為孤立的問題（問題 "#1189"）。有關 GCNV 後端組態，請參閱 "配置 Google Cloud NetApp Volumes"。
- 修正了在傳輸協定內建於核心的主機上偵測 NVMe/TCP 的問題（問題 "#1019"）。有關節點要求，請參閱 "準備工作節點"。
- 修正了當 ONTAP Volume 已經夠大以滿足要求（包括快照保留）時， ONTAP-NAS 和 ONTAP-NAS-FlexGroup 擴充無限期重試的問題。如需 Volume 擴充，請參閱 "展開Volume"。
- 修正了由於缺少交會路徑而導致的 Volume 掛載失敗問題。有關 ONTAP NAS 組態，請參閱 "設定 ONTAP NAS 驅動程式"。
- 修正當 `storagePrefix` 包含破折號時， ONTAP-SAN-Economy Volume 建立失敗的問題。如需後端組態選項，請參閱 "SAN組態選項與範例ONTAP"。
- 修正了使用自動匯出原則發佈唯讀 ONTAP-NAS Clone 的問題，以便保留來源 FlexVol 匯出規則。如需 ONTAP NAS 組態，請參閱 "列舉NAS組態選項與範例ONTAP"。
- 修正了 HTTP 傳輸重試行為（問題 "#1139"）。

- 修正了跳過 SCSI 裝置移除的幽靈裝置偵測問題（問題"#1161"）。
- 修正啟用並行的資源配置會選擇所選節點偏好區域以外之拓撲區域的問題。如需並行與拓撲行為的相關資訊，請參閱 "控制器擴充性"。
- 修正啟用並行時，等效後端之間 Volume 放置位置不均的問題；現在後端會像非並行資源配置一樣，隨機均勻地選擇。如需詳細資訊，請參閱 "控制器擴充性"。
- 修正了在 Trident 節點 pod 啟動期間主機網路位址表變更時，註冊節點缺少 IP 位址，從而可能移除 NFS 匯出原則規則的問題。
- 修正了在 REST 輪詢之後繼續建立 Clone 並將 PVC 綁定為未分割的 FlexClone 的問題；Trident 現在可以觀察到稍後的建立作業失敗。
- 修正了 ONTAP-SAN Volume 擴充在節點上回報錯誤的調整大小失敗的問題。如需 Volume 擴充的相關資訊，請參閱 "展開Volume"。

實驗性增強功能

> 不適用於生產環境。

- **[技術預覽] Kubernetes**：新增支援從 ONTAP S3 進行以 COSI 為基礎的物件儲存桶動態資源配置。如需詳細資訊，請參閱 "使用 COSI 佈建物件儲存"。

Trident保護

增強功能

- 新增對加密 AppVault 快照和備份中繼資料的支援。為每個 AppVault 使用自訂的保存庫密碼。輪換保存庫密碼會重新加密中繼資料加密金鑰，並更新 AppVault 中的每個 Kopia 和 Restic 儲存庫。若要深入瞭解，請參閱 "加密 Trident Protect 快照與備份的 AppVault 中繼資料"。
- 新增了在還原期間保留 KubeVirt 虛擬機器 MAC 位址的支援，這在應用程式授權與 MAC 位址綁定時非常有用。若要啟用此功能，請將 protect.trident.netapp.io/retain-mac-address 註解設為 "true"。若要深入瞭解，請參閱 "使用進階Trident Protect 恢復設定"。
- 新增對 Volume 群組快照的支援，該快照可擷取應用程式在同一時間點的相關持續性 Volume 宣告。在應用程式自訂資源的 `spec.volumeGroups`中，透過標籤、KubeVirt 虛擬機器或兩者來定義 Volume 群組。若要深入瞭解，請參閱 "搭配 Trident Protect 使用 Volume 群組快照"。
- 新增支援容錯移轉至複寫應用程式的較早快照，允許從資料損壞或受損前的時間點進行恢復。如需詳細資訊，請參閱 "使用 Trident Protect 容錯移轉到較早的快照"。

修正

- 修正了 Trident Protect 安裝在 `trident-protect-pre-upgrade-hook` 作業失敗的問題，因為遺失 `trident-protect-controller-manager` ServiceAccount（"問題 #1154"）。
- 修正僅限 PVC 的還原，使其僅在擁有者存在時才包含 `ownerReference`。

26.06.2 版本新增內容

了解 NetApp Trident 和 Trident Protect 的新特性，包括增強功能、修復和棄用。

Trident

增強功能

修正

- * Kubernetes* ：
  - 修正了當 Trident 控制器重新啟動或更新 `TridentBackendConfig`，且 `enableConcurrency`設定為 `true`時，ONTAP-NAS 匯出原則規則會被靜默刪除的問題。這可能會使已掛載的 Volume 處於停滯狀態（"問題 #1166"，"問題 #1179"）。
  - 修正了後端更新針對非實質變更（例如憑證和 QoS）執行每個 Volume 孤立掃描的問題，這可能會在大型後端上區塊 CSI 操作數小時。現在，該掃描僅在 SVM 或儲存設備前綴發生變更時執行。
  - 修正了 Trident 偵錯日誌中出現的憑證問題，包括 CSI gRPC 請求轉儲、啟動環境轉儲以及 `TridentBackendConfig` CRD 控制器記錄的規格。啟動環境轉儲現在只記錄 Trident 自身 `TRIDENT_*` 變數的值，並按名稱列出其他變數。

26.06.1 版本新增內容

了解 NetApp Trident 和 Trident Protect 的新特性，包括增強功能、修復和棄用。

Trident

修正

- * Kubernetes* ：
  - 修正了 CSI 發布和 Volume 移動之間的競爭條件，該條件可能導致使用過時的存取資訊附加 Volume。
  - 修復了 NodeGetInfo 在節點註冊完成之前可能返回空拓撲標籤的競爭條件，進而導致拓撲感知佈建失敗。
  - 修復了一個罕見的 NVMe 斷開連接競爭問題，該問題可能在另一個 pod 仍在連接時拆除共享的子系統。
  - 修正了在並發條件下建立掛載 Volume 的 Clone 時，NVMe PVC Clone 發布失敗（以及父命名空間映射錯誤）的問題。
  - 修正了使用 nameTemplate 的 ONTAP-SAN-Economy 後端上 PVC 快照建立失敗的問題。

增強功能

- * Kubernetes* ：
  - 在 ONTAP 後端設定中新增選用的 `storageAPITimeout`（秒），用於設定每個後端的 REST 和 ZAPI API 用戶端逾時時間。

26.06 版本新增內容

了解 NetApp Trident 和 Trident Protect 的新特性，包括增強功能、修復和棄用。

Trident

增強功能

- * Kubernetes* ：
  - 為 ONTAP-NAS-Economy 和 ONTAP-SAN-Economy 驅動程式新增並行處理的正式版 （GA） 支援，使此功能從技術預覽版升級，並為 Azure NetApp Files （ANF） 和 SolidFire 驅動程式新增 GA 並行處理支援。
  - 新增對 Kubernetes 1.36 的支援。
  - 為 Google Cloud NetApp Volumes (GCNV) NAS 和 SAN 驅動程式新增了磁碟區自動成長的正式版 (GA) 支援。
  - 新增統一的 ONTAP 平衡放置功能，以便在佈建新磁碟區時實現最佳集合體選擇（"問題 #64"）。
  - 為 ONTAP-NAS-Flexgroup 驅動程式新增了 ONTAP AFX 支援。
  - 新增與 NetApp Shift Toolkit 的整合，以在使用 Red Hat OpenShift Migration Toolkit for Virtualization 時簡化虛擬機器移轉。
  - 為 GCNV SAN 驅動程式新增了 Cloud Identity 支援。
  - 已為 Trident Volume Publication （TVP） 傳播和自動增長工作流程新增從屬磁碟區支援。
  - 新增在啟用並行功能的情況下執行 Trident 時，對磁碟區更新和磁碟區群組快照的支援。
  - 為 GCNV 新增動態 gRPC 速率限制。
  - 新增了控制器就緒執行探測，並調整了控制器和邊車的預設 Kubernetes API QPS 和 Burst 設定。
  - 改進節點操作順序，將附加、加密設定和檔案系統格式化步驟解耦。
  - 將 Trident 容器根檔案系統掛載為唯讀，以提高安全性。
  - 在 ONTAP 後端中，當複製磁碟區的來源快照被刪除時，會盡力刪除複製磁碟區快照。
  - 新增備份複製 PVC 識別，以防止由 Veeam Kasten、Trilio 和 Cohesity 建立的臨時 PVC 卡在 ONTAP 磁碟區復原佇列中。
  - 增強了 AWS ARN 處理能力，以支援特殊區域。
  - 防止 ONTAP-SAN-ECONOMY 驅動程式在調整大小操作期間 Flexvols 縮小。

實驗性增強功能

不適用於生產環境。

- **[Tech Preview]**：新增對 ONTAP-NAS-Economy qtree 匯入的支援。
- **[Tech Preview]**：為 ASA-R2 （SAN 和 NVMe）驅動程式新增並行支援。
- **[Tech Preview]**：為 ONTAP-SAN 和 ONTAP-NAS 驅動程式新增了 GCNV ONTAP 模式 HTTP 傳輸。

修正

- * Kubernetes* ：
  - 修復了並行發布/取消發布、複製、快取和後端更新作業的競爭條件和狀態處理。
  - 修正了 ONTAP-NAS 和 ONTAP-NAS-Economy 驅動程式中從屬磁碟區和唯讀複製的匯出原則競爭條件及並行發布/取消發布處理問題。
  - 已修正當快照原則不同時，在 MetroCluster 容錯移轉後的 ONTAP-NAS-Economy FlexVol 探索 （"問題

#1082"）。

- 修正了 ONTAP-NAS-Economy 磁碟區刪除重試中斷長時間執行的 qtree 刪除的問題（"問題 #1121"）。
- 修正了 ONTAP-SAN 和 ONTAP-SAN-Economy 的匯入和調整大小行為，包括 fsType 驗證、磁碟區元資料處理和自動成長模式行為。
- 修正了並行建立磁碟區期間的 NVMe/TCP 命名空間競爭條件（"問題 #1089"）。
- 修正了當可用路徑和入口網站不符時發生的 iSCSI 調整大小失敗問題，並改善了重複調整大小的行為。
- 修正了由磁碟區繁忙狀態引起的 ONTAP-SAN 磁碟區刪除失敗問題。
- 修正了 ONTAP REST 錯誤處理問題，並改進了 SVM 狀態逾時行為。
- 修正了 GCNV SAN 區域集區和主機群組處理、標準化磁碟區識別碼轉換，以及標準化低於最小大小的建立要求。
- 修正了原生 GCNV 容量池選擇問題，以排除 ONTAP 模式集區。
- 修正了使用 Trident 自動工作負載故障轉移的虛擬機器的 KubeVirt 故障轉移問題，以加快虛擬機器故障轉移速度。
- 修正了 Operator 管理的安裝中 RBAC 問題，包括節點缺失 ClusterRoles 以及升級期間 ClusterRole 和 ClusterRoleBinding 的錯誤協調問題。此修復適用於使用 Trident Operator chart 的 Operator 和 Helm 型安裝。
- 修正了 GitOps 在 TridentConfigurator 和相關 CRD 上的持久差異，方法是省略預設優先值並修正 YAML 範本縮排（"問題 #915"、"問題 #1120"）。
- 修正了在 Helm 升級期間修補 TridentVolumePublication CRD 的問題（"問題 #1151"）。
- 透過在節點註冊重試之前啟動 CSI gRPC 服務來修復 CSI 啟動順序問題。
- 修正了並行作業期間安全 SMB 複製組態覆蓋的問題。
- 修正了升級後 Trident 控制器 pod 上缺少的註解（"問題 #1004"）。
- 改進大型 SVM 上的慢速 qtree 配置（"問題 #1142"）。
- 已修復 ControllerPublish，現在使用磁碟區組態檔案系統類型，而不是錯誤的預設檔案系統類型。
- 修復了孤立的 dm 設備可能導致 CSI 重複 NodeUnstageVolume 故障的問題。
- 改進了過時節點和多路徑清理，在軟刪除節點清理後移除幽靈多路徑成品和過時的 TridentNode 資源。

Trident保護

增強功能

- 新增了 OpenShift Service CA Operator 整合，用於自動化 Trident Protect 准入 Webhook 的 TLS 憑證管理。可透過在 Helm 值中將 `useOpenshiftServiceCA`設定為 `true`來啟用。若要深入瞭解，請參閱"自訂Trident Protect 安裝"。
- 新增在 OpenShift Virtualization 上備份與還原 KubeVirt VM 的支援。保護命名空間中的所有 VM，或依名稱指定特定的 VM，並從備份或快照中選擇性地還原個別 VM。若要深入瞭解，請參閱 "使用 Trident Protect 保護 KubeVirt 虛擬機器"。
- 新增了對資源轉換的支援，讓您在還原過程中新增、刪除和修改 Kubernetes 資源屬性。若要深入瞭解，請參閱 "使用自訂資源還原資源時修改資源"。
- 新增了對 AppMirrorRelationship （AMR）中 `storageClassMapping`的支援，可在單一 AMR 定義中為各個 PVC 進行儲存類別對應。若要深入瞭解，請參閱 "使用 Trident Protect 設定和管理應用程式複寫"。

修正

- 修正了當來源應用程式命名空間在目標叢集上不存在時，跨叢集虛擬機器還原失敗並出現准入 webhook 錯誤的問題。
- 修正了由於與現有應用程式自訂資源的規格不符，導致選擇性虛擬機器就地還原在應用程式建立步驟失敗的問題；現在可重複使用現有應用程式。
- 修正了虛擬機器就地復原作業在虛擬機器及其 PersistentVolumeClaim 均被刪除時停滯於 `Running`狀態的問題；缺少 PersistentVolumeClaim 現在將被視為嚴重錯誤。
- 修正了編輯現有保護排程時錯誤地觸發立即備份或快照執行的問題。
- 修正了快照終結器在應用程式自訂資源被刪除後嘗試獲取該資源的問題；現在，如果找不到應用程式，終結器會跳過此步驟。
- 修正在 PersistentVolumeClaim 複製步驟中使用稀疏複製進行虛擬機器就地還原後，固定保護儲存容量大幅增加的問題。
- 修正了 Trident Protect 在 ONTAP 中確認刪除後端快照之前刪除 Kubernetes 快照物件的問題；現在終結器會等待 `VolumeSnapshot`刪除。
- 修正了虛擬機器凍結操作失敗時快照無法繼續執行的問題；現在凍結和解凍操作在失敗時仍會繼續進行。
- 已修正 KubeVirt 虛擬機器應用程式在 AppMirrorRelationship 反向、反向重新同步和重新同步作業期間，錯誤地重新建立 PersistentVolumeClaims 的問題。

26.02.1 版本新增內容

了解 NetApp Trident 和 Trident Protect 的新特性，包括增強功能、修復和棄用。

Trident

修正

- * Kubernetes* ：
- 已修正 Google Cloud NetApp Volumes（GCNV）SAN 行為，以支援區域性 Flex Unified ZONAL 資源池用於區塊（iSCSI）工作負載。現在同時支援區域性與區域間的組態。
- 修正 Google Cloud NetApp Volumes（GCNV）NAS UNIFIED 資源池的處理方式，透過改進磁碟區建立、驗證、命名以及在 `InternalID`不存在時的備援磁碟區查找操作。
- 修正了當可用路徑和 portal 不符時，iSCSI 裝置調整大小作業可能會失敗的問題。
- 修正了 `blkid` 錯誤識別 LUKS 裝置的檔案系統類型、導致無法建立新磁碟區的間歇性問題。
- 修正了大規模 LUKS 發布和暫存操作可能導致 Trident 控制器阻塞的問題。現在使用者需要追蹤與磁碟區關聯的 LUKS 密碼短語。
- 修正了 iSCSI 多路徑分割區可能顯示為幽靈裝置，導致 CSI `NodeUnstage` 操作無法成功的問題。
- 修正了並行建立磁碟區期間的 NVMe/TCP 命名空間競爭條件（Issue #1089）。
- 修正了在同一秒內執行快照時並發克隆快照操作的問題。
- 修正了阻止使用 `dataVolumes`的 KubeVirt VM 透過 Trident Automated Workload Failover 進行故障轉移的問題。
- 修正了在使用 Trident 並發核心時刪除無效 `TridentMirrorRelationship`自訂資源時發生的錯誤。

- 修正了 `tridentactionmirrorupdates` Kubernetes CRD YAML 定義中的縮排問題（問題 #1120）。
- 修正了 REST qtree 非同步刪除 API 呼叫（Issue #1121）。
- 已修復 `ControllerPublish`，現在使用磁碟區組態檔案系統類型，而不是錯誤的預設檔案系統類型。
- 修正了控制器啟動日誌記錄，使其能夠正確報告並行驅動程式狀態。
- 修正了並發 Trident 中對已暫停後端的處理。
- 增強了 AWS ARN 處理能力，以支援特殊區域。

Trident保護

修正

更新了 Trident Protect 依賴項以解決安全漏洞：

- gRPC-Go："CVE-2026-33186"
- Go x/crypto SSH："CVE-2024-45337"
- OpenTelemetry-Go："CVE-2026-24051"
- Go 執行階段："CVE-2025-22871"
- 其他相依性："CVE-2025-23109"

26.02 版本新增內容

了解 NetApp Trident 和 Trident Protect 的新特性，包括增強功能、修復和棄用。

Trident

增強功能

- * Kubernetes* ：
  - 新增 ONTAP-NAS（僅限 NFS）、ONTAP-SAN（iSCSI、FCP、NVMe）和 Google Cloud NetApp Volumes（GCNV）驅動程式並發支援的正式版（GA），此功能已從技術預覽版升級。如需詳細資訊，請參閱 "控制器擴充性文件"。
  - 新增了對 Trident 基於自訂使用者定義 Trident AutoGrow 策略的磁碟區自動成長支援。如需詳細資訊，請參閱 "自動磁碟區擴充文件"。
  - 增強了 Trident 節點的並發性，以提高 NVMe 磁碟區節點操作的可擴展性。如需詳細資訊，請參閱 "NVMe Volume 說明文件"。
  - 新增對 Google Cloud NetApp Volumes NAS 磁碟區自動分層的支援（透過 PVC 註解 (`tieringPolicy`、 `tieringMinimumCoolingDays`），包括池選擇和克隆繼承。如需詳細資訊，請參閱 "設定 Google Cloud NetApp Volumes 文件的自動分層"。
  - 新增對使用 `google-cloud-netapp-volumes-san` 驅動程式的 GCNV SAN 區塊（iSCSI）磁碟區的支援，包括資源配置、用於 LUN 存取的每節點主機群組對應，以及從磁碟區複製作業。如需詳細資訊，請參閱 "Google Cloud NetApp Volumes 區塊組態文件"。
  - 新增對 Amazon FSx for NetApp ONTAP 自動後端組態的支援。當您建立包含所需參數的 StorageClass 時、 Trident 會自動建立對應的後端和 VolumeSnapshotClass （如有需要）。如需詳細資訊、請參閱 "Amazon FSx for NetApp ONTAP 後端組態文件"。

- 增加了對不同 Microsoft Azure 雲端的支援，例如 Azure Government 和 Azure China，以及 Azure NetApp Files 後端的自訂雲端組態。如需詳細資訊，請參閱 "Azure NetApp Files 後端組態文件"。
- 新增對 Kubernetes 1.35 的支援。如需詳細資訊，請參閱 "需求文件"。

實驗性增強功能

不適用於生產環境。

- **[Tech Preview]**：為 ONTAP-NAS-Economy 和 ONTAP-SAN-Economy 驅動程式新增了並發支援。

已知問題

`csi-snapshotter` sidecar 存在已知問題。在所有 Kubernetes 版本中、VolumeGroupSnapshots `v1beta1` 會阻止 VolumeSnapshots 達到 `ReadyToUse`狀態。

有兩種因應措施：

1. 刪除 VolumeGroupSnapshots CRD 以停用 VolumeGroupSnapshots，然後重新安裝 Trident。
2. 安裝 VolumeGroupSnapshots `v1beta2` 和 `snapshot-controller` 版本 8.4.0 或更高版本，然後重新安裝 Trident。VolumeGroupSnapshots 無法在低於 v1.34 的 Kubernetes 版本上運行。

修正

- * Kubernetes* ：
  - 修正了在 ONTAP-NAS、ONTAP-NAS-Economy 和 Google Cloud NetApp Volumes 驅動程式中，取消發布只讀複製會從來源磁碟區中刪除匯出策略規則的問題"問題 #1086"。
  - 將 kubectl 映像切換為基於 Alpine 的輕量級變體，以防止 Bitnami 公開映像棄用後出現拉取失敗 "問題 #1080"。
  - 修正了在 Trident 升級期間保留現有部署註解的問題"問題 #1004"。
  - 如果兩個儲存類別都指向同一個後端 "問題 #1104"，則允許跨不同儲存類別進行複製。
  - 修正了雲端環境中網路延遲導致的節點準備逾時問題。提高了雲端部署的逾時值。
  - 修正了 LUN 建立過程中的一個問題，該問題會導致程序進入重試狀態時檔案系統類型屬性保持未設定狀態。
  - 修正了 REST API Volume 查詢功能，使其忽略 Volume 狀態，防止在 Volume 查詢期間出現誤報。
  - 提高 Trident 控制器在大規模使用 ontap-nas-economy 驅動程式時的效率。
  - 在 ontap-san-economy 驅動程式中，於 LUN 匯入期間設定 internalID。
  - 提高了 Azure Resource Graph 查詢限制，以處理更多子網路。
  - 改進了 CSI 和 ONTAP 克隆分割逾時，以避免與某些備份應用程式出現競爭條件 "問題 #1098" "問題 #1100"。
  - 修正了 LUKS 錯誤訊息的抑制問題"問題 #1069"。

- 修正了 iSCSI 和 NVMe 協定中過期 LUKS 映射器的處理問題。增強的清理邏輯可防止因孤立的裝置映射器而導致的掛載失敗。

- 修正了 RWX NVMe Volume 的規模限制。
- 更新了 OpenTelemetry-Go 套件以修復 "CVE-2026-24051"。

Trident保護

增強功能

- Trident Protect 現在會在執行就地還原之前自動停用保護排程並取消正在進行的作業，並在還原完成後重新啟用它們。如需深入瞭解，請參閱 "使用Trident Protect 恢復應用程式"。
- 已將 `runImmediately`欄位新增至排程 `CR` 和 `--run-immediately` CLI 標誌中，以便在建立排程時立即觸發備份或快照。如需深入瞭解，請參閱"建立資料保護排程"。
- 新增使用 restore CR 中的 `destinationApplicationName`欄位或 `--destination-app-name` CLI 旗標為還原的應用程式指定自訂名稱的支援。如需深入瞭解、請參閱"使用Trident Protect 恢復應用程式"。

修正

- 修正了由於在所需服務帳戶可用之前建立 Pod 而導致的還原失敗問題。
- 已修復在應用程式還原期間跳過 `Roles` 和 `RoleBindings` 的問題。
- 修正了儘管配置正確，但來源叢集名稱未顯示在 `tridentctl-protect get appvaultcontent` 輸出中的問題。
- 修正了由於缺少 `pipefail` 處理而被忽略的 Kopia 還原錯誤。
- 修正了因資源篩選器排除持久性磁碟區而導致的快照和備份失敗問題。
- 修正了在跨命名空間具有相同名稱的 PVC 的多命名空間應用程式中不正確的 PVC 還原問題，這可能會導致資料遺失。

25.10 版本新增內容

了解Trident和Trident Protect 的新特性，包括增強功能、修復和棄用。

Trident

增強功能

- * Kubernetes* ：
  - 除了ONTAP -SAN（iSCSI 和 FC）驅動程式外，還為ONTAP-NAS NFS 和ONTAP -SAN-Economy 驅動程式新增了對 CSI 磁碟區組快照的支持，v1beta1 磁碟區組快照 Kubernetes API。看"使用磁碟區組快照"。
  - 為ONTAP-NAS 和ONTAP-NAS-Economy（NAS 驅動程式中不包括 SMB）以及ONTAP-SAN 和ONTAP-SAN-Economy 驅動程式新增了強製磁碟區分離的自動工作負載故障轉移支援。看"使用Trident實現有狀態應用程式的故障轉移自動化"。
  - 增強Trident節點並發性，以提高 FCP 卷節點操作的可擴展性。
  - 為ONTAP NAS 驅動程式新增了ONTAP AFX 支援。看"列舉NAS組態選項與範例ONTAP"。
  - 增加了透過 TridentOrchestrator CR 和 Helm chart 值配置Trident容器的 CPU 和記憶體資源請求和限制

的支援。（"第1000期" ，"第927期" ，"第853期" ，"第592期" ，"第110期" ）。

- 為 ASAr2 人格添加了 FC 支援。看"SAN組態選項與範例ONTAP"。
- 新增了一個選項，可以使用 HTTPS 而不是 HTTP 來提供 Prometheus 指標。看"監控 Trident"。
- 新增了一個選項 `--no-rename`導入磁碟區時，保留原始名稱，但讓Trident管理磁碟區的生命週期。看"匯入磁碟區"。
- Trident部署現在以系統叢集關鍵優先權運作。

- 為Trident控制器新增了透過 helm、operator 和 tridentctl 使用主機網路的選項（"第858期" ）。
- 在 Trident 25.10 中為 ANF 驅動程式新增了手動 QoS 支援，使其可用於正式作業；此實驗性增強功能已於 Trident 25.06 中推出。

實驗性增強功能

不適用於生產環境。

- **[技術預覽]**：除了現有的ONTAP-SAN 驅動程式技術預覽版（統一ONTAP 9 中的 iSCSI 和 FCP 協定）之外，還增加了對ONTAP-NAS（僅限 NFS）和ONTAP-SAN（統一ONTAP 9 的 NVMe）的並發支援。

修正

- **Kubernetes**：
  - 透過將 Linux DaemonSet 標準化為 node-driver-registrar 來修復 CSI node-driver-registrar 容器名稱不一致的問題，使其與 Windows DaemonSet 和容器鏡像命名保持一致。
  - 修正了舊版 qtree 的匯出策略未正確升級的問題。
- **Openshift**：
  - 修正了由於 SCC 將 allowHostDirVolumePlugin 設為 false 而導致的 Openshift 中 Windows 節點上的Trident節點 pod 無法啟動的問題（"第950期" ）。
- 修正了 Kubernetes API QPS 無法通過 Helm 設定的問題（"第975期" ）。
- 修正了無法在同一個 Kubernetes 節點上基於 NVMe XFS 檔案系統 PVC 的快照掛載持久性磁碟區宣告 (PVC) 的問題。
- 修正了在 NDVP 模式下主機/Docker 重新啟動後 UUID 變更的問題，方法是為每個後端新增唯一/共用的子系統名稱（例如 netappdvp_subsystem）。
- 修正了Trident從 23.10 之前的版本升級到 24.10 及以上版本期間 iSCSI 磁碟區的掛載錯誤，解決了「無效的 SANType」問題。
- 修正了Trident後端狀態在不重新啟動Trident控制器的情況下無法轉換到線上/離線狀態的問題。
- 修正了導致PVC尺寸調整緩慢的間歇性競爭條件。
- 修正了卷宗克隆失敗時快照未被清理的問題。
- 修正了核心更改磁碟區的裝置路徑時無法取消暫存磁碟區的問題。
- 修正了由於 LUKS 裝置已關閉而導致無法取消暫存磁碟區的問題。
- 修復了儲存操作緩慢導致 ContextDeadline 錯誤的問題。
- Trident Operator 將等待可設定的 k8s-timeout 時間來檢查Trident版本。

Trident保護

NetApp Trident Protect 提供進階應用程式資料管理功能，增強了由NetApp ONTAP儲存系統和NetApp Trident CSI 儲存供應器支援的有狀態 Kubernetes 應用程式的功能和可用性。

增強功能

- 新增了用於控制計劃和備份 CR 的快照 CR 逾時的註釋：
  - `protect.trident.netapp.io/snapshot-completion-timeout`
  - `protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout`
  - `protect.trident.netapp.io/volume-snapshots-created-timeout`

    看"支援的備份和計劃註釋"。
- 在計劃 CR 中新增了一個註釋，用於配置 PVC 綁定逾時，該逾時將由備份 CR 使用：`protect.trident.netapp.io/pvc-bind-timeout-sec` 。看"支援的備份和計劃註釋"。
- 改進 `tridentctl-protect`備份和快照清單新增一個字段，用於指示執行鉤子失敗。

25.06.2 中的變更

Trident

修正

- **Kubernetes**：修正了從 Kubernetes 節點分離磁碟區時發現不正確的 iSCSI 裝置的嚴重問題。

25.06.1 中的變更

Trident

> 對於正在使用 SolidFire 的客戶，請勿升級至 25.06.1，因為在取消發佈磁碟區時存在已知問題。25.06.2 即將發佈以解決此問題。

修正

- * Kubernetes* ：
  - 修正了從子系統取消映射之前未檢查 NQN 的問題。
  - 修正了多次嘗試關閉 LUKS 裝置導致無法分離磁碟區的問題。
  - 修正了當裝置路徑自建立以來發生變化時 iSCSI 磁碟區取消暫存的問題。
  - 阻止跨儲存類別的磁碟區克隆。
- **OpenShift**：修正了 OCP 4.19 中 iSCSI 節點準備失敗的問題。
- 增加了使用SolidFire後端克隆卷時的超時時間（"問題 #1008" ）。

25.06 中的變化

Trident

增強功能

- * Kubernetes* ：
  - 增加了對 CSI 卷組快照的支持 `v1beta1`適用於 ONTAP-SAN iSCSI 驅動程式的磁碟區組快照 Kubernetes API。請參閱。 "使用磁碟區組快照"

    VolumeGroupSnapshot 是 Kubernetes 中的一個 Beta 功能，包含 Beta 版 API。VolumeGroupSnapshot所需的最低版本為 Kubernetes 1.32。

  - 除了 iSCSI 之外，還增加了對ONTAP ASA r2 的 NVMe/TCP 支援。看"SAN組態選項與範例ONTAP"。
  - 為 ONTAP-NAS 和 ONTAP-NAS-Economy 磁碟區新增了安全的 SMB 支援。 ActiveDirectory 使用者和群組現在可以與 SMB 磁碟區一起使用，以增強安全性。請參閱。 "啟用安全 SMB"
  - 增強 Trident 節點並發性，以提高 iSCSI 磁碟區節點作業的可擴充性。
  - 額外 `--allow-discards`打開 LUKS 磁碟區時允許丟棄/TRIM 命令以回收空間。
  - 格式化 LUKS 加密磁碟區時的效能增強。
  - 增強了對失敗但部分格式化的 LUKS 設備的 LUKS 清理。
  - 增強了 Trident 節點冪等性，用於 NVMe 卷的連接和分離。
  - 額外 `internalID`欄位到 ONTAP-SAN-Economy 驅動程式的 Trident 磁碟區配置。
  - 增加了對使用 SnapMirror 對 NVMe 後端進行磁碟區複製的支援。請參閱。 "使用 SnapMirror 複寫磁碟區"

實驗性增強功能

不適用於生產環境。

- [技術預覽] 透過以下方式啟用並發 Trident 控制器操作 `--enable-concurrency`功能標誌。這允許控制器操作並行運行，從而提高繁忙或大型環境中的效能。

  此功能尚處於實驗階段，目前支援使用 ONTAP-SAN 驅動程式（iSCSI 和 FCP 協定）的有限平行工作流程。

- [技術預覽] 使用 ANF 驅動程式新增了手動 QOS 支援。

修正

- * Kubernetes* ：
  - 修正了 CSI NodeExpandVolume 的問題，當底層 SCSI 磁碟不可用時，多路徑裝置可能會出現大小不一致的情況。
  - 修正了無法清理 ONTAP-NAS 和 ONTAP-NAS-Economy 驅動程式的重複匯出策略的問題。
  - 修正了 GCNV 磁碟區在 `nfsMountOptions`未設定時預設使用 `NFSv3` 的問題；現在同時支援

NFSv3 和 NFSv4 協定。如果未提供 `nfsMountOptions，則使用主機的預設 NFS 版本（NFSv3 或 NFSv4）。

- 修正了使用 Kustomize 安裝 Trident 時出現的部署問題（"問題 #831"）。
- 修正了從快照建立的 PVC 缺少匯出策略的問題（"問題 #1016"）。
- 修正了 ANF 磁碟區大小未自動與 1 GiB 增量對齊的問題。
- 修正了將 NFSv3 與 Bottlerocket 結合使用時的問題。

- 修正了 ONTAP-NAS-Economy 磁碟區儘管調整大小失敗但仍可擴展至 300 TB 的問題。
- 修正了使用 ONTAP REST API 時克隆分割操作同步完成的問題。

棄用：

- **Kubernetes**：將最低支援的 Kubernetes 更新至 v1.27。

Trident保護

NetApp Trident Protect 提供進階應用程式資料管理功能，增強了由NetApp ONTAP儲存系統和NetApp Trident CSI 儲存供應器支援的有狀態 Kubernetes 應用程式的功能和可用性。

增強功能

- 改善了復原時間，提供了更頻繁的完整備份的選項。
- 透過 Group-Version-Kind (GVK) 過濾提高了應用程式定義和選擇性恢復的粒度。
- 將 AppMirrorRelationship (AMR) 與 NetApp SnapMirror 結合使用時可實現高效的重新同步和反向複製，以避免完整的 PVC 複製。
- 增加了使用 EKS Pod Identity 建立 AppVault 儲存桶的功能，無需使用 EKS 叢集的儲存桶憑證指定機密。
- 如果需要，新增了在復原命名空間中跳過復原標籤和註解的功能。
- AppMirrorRelationship (AMR) 現在將檢查來源 PVC 擴展並根據需要對目標 PVC 執行適當的擴展。

修正

- 修正了先前快照的快照註解值被套用到新快照的錯誤。現在所有快照註解均已正確套用。
- 如果未定義，則預設定義資料移動器加密（Kopia / Restic）的秘密。
- 為 S3 appvault 建立新增了改進的驗證和錯誤訊息。
- AppMirrorRelationship (AMR) 現在只複製處於 Bound 狀態的 PV，以避免嘗試失敗。
- 修正了在具有大量備份的 AppVault 上取得 AppVaultContent 時顯示錯誤的問題。
- KubeVirt VMSnapshots 被排除在復原和故障轉移操作之外，以避免故障。
- 修正了 Kopia 的問題：由於 Kopia 預設保留計劃覆蓋了用戶在計劃中設定的計劃，導致快照過早刪除。

25.02.1 的變更

Trident

修正

- * Kubernetes* ：
  - 解決了 Trident 運算符中使用非默認映像註冊表時 sidecar 映像名稱和版本錯誤填充的問題（"問題 #983"）。
  - 解決了在 ONTAP 容錯移轉恢復期間多重路徑工作階段無法恢復的問題（"問題 #961"）。

25.02 的變更

從Trident 25.02 開始，「新增功能」摘要提供了Trident和Trident Protect 版本的增強功能、修復和棄用詳情。

Trident

增強功能

- * Kubernetes* ：
  - 新增對 iSCSI 的 ONTAP ASA R2 支援。
  - 新增在非正常節點關機案例中強制分離 ONTAP NAS 磁碟區的支援。新的 ONTAP NAS 磁碟區現在將使用由 Trident 管理的每個磁碟區匯出原則。提供升級路徑，讓現有的磁碟區在解除發佈時轉換至新的匯出原則模型，而不會影響作用中的工作負載。
  - 新增 cloneFromSnapshot 註釋。
  - 新增跨命名空間磁碟區複製支援。
  - 增強的 iSCSI 自我修復掃描修正功能，可透過精確的主機，通道，目標和 LUN ID 來初始化重新掃描。
  - 增加了對 Kubernetes 1.32 的支援。
- * OpenShift* ：
  - 新增對 ROSA 叢集上的 RHCOS 自動 iSCSI 節點準備的支援。
  - 新增對 ONTAP 驅動程式 OpenShift 虛擬化的支援。
- 在 ONTAP SAN 驅動程式上新增光纖通道支援。
- 新增 NVMe LUKS 支援。
- 已切換至所有基礎映像的暫存映像。
- 已新增 iSCSI 連線狀態探索和記錄功能，可在 iSCSI 工作階段應登入時進行，但不會（"問題 #961"）。
- 使用 googoogle 雲端 NetApp 磁碟區驅動程式新增對 SMB 磁碟區的支援。
- 新增支援，允許 ONTAP 磁碟區在刪除時略過恢復佇列。
- 新增支援以取代標籤，取代預設影像。
- 新增映像拉取秘密旗標至 Tridentctl 安裝程式。

修正

- * Kubernetes* ：
  - 修復自動匯出原則中遺失的節點 IP 位址（"問題 #965"）。
  - 為節省 ONTAP NAS 成本，提早將自動匯出原則切換至每個 Volume 原則。

- 固定後端組態認證，可支援所有可用的 AWS ARN 分割區（"問題 #913"）。
- 新增選項可在 Trident 運算子（）中停用自動組態設定器協調"問題 #924"。
- 增加安全性 CSI 調整容器的 Context （"問題 #976"）。

Trident保護

NetApp Trident Protect 提供進階應用程式資料管理功能，增強了由NetApp ONTAP儲存系統和NetApp Trident CSI 儲存供應器支援的有狀態 Kubernetes 應用程式的功能和可用性。

增強功能

- 為 KubeVirt / OpenShift 虛擬化虛擬機器添加了備份和復原支持，支援 volumeMode: File 和 volumeMode: Block（原始設備）儲存。此支援與所有Trident驅動程式相容，並在使用NetApp SnapMirror和Trident Protect 複製儲存時增強了現有的保護功能。
- 新增在 Kubevirt 環境的應用程式層級控制凍結行為的功能。
- 新增了設定 AutoSupport Proxy 連線的支援。
- 新增定義資料移動器加密機密的功能（ Kopia / Restic ）。
- 新增了手動執行掛鉤的功能。
- 增加了在Trident Protect 安裝過程中設定安全上下文約束 (SCC) 的功能。
- 增加了在Trident Protect 安裝過程中設定 nodeSelector 的支援。
- 新增 AppVault 物件的 HTTP / HTTPS 外傳 Proxy 支援。
- 延伸資源篩選器可啟用叢集範圍資源的排除。
- 在 S3 AppVault 認證中新增對 AWS 工作階段權杖的支援。
- 在快照前執行攔截之後新增資源集合支援。

修正

- 改善暫存磁碟區的管理，以略過 ONTAP 磁碟區恢復佇列。
- SCC 註釋現在會還原為原始值。
- 支援平行作業，提升還原效率。
- 強化支援大型應用程式的執行掛機逾時。

24.10.1 的變更

增強功能

- * Kubernetes* ：增加了對 Kubernetes 1.32 的支援。
- 已新增 iSCSI 連線狀態探索和記錄功能，可在 iSCSI 工作階段應登入時進行，但不會（"問題 #961"）。

修正

- 修復自動匯出原則中遺失的節點 IP 位址（"問題 #965"）。
- 為節省 ONTAP NAS 成本，提早將自動匯出原則切換至每個 Volume 原則。

- 已更新 Trident 和 Trident ASUP 相依性，以解決 CVE-2024-45337 和 CVE-2024-45310 的問題。
- 在 iSCSI 自我修復期間，移除間歇性不佳的非 CHAP 入口網站登出 ("問題 #961") 。

24.10 的變更

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。
- 在 LUKS LUN 中添加了 18 MiB 開銷，以確保報告的 PVC 大小可用。
- 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

    "了解更多關於Trident Protect的信息"。
- 新增了新標誌 `--k8s-api-qps`安裝程式設定Trident與 Kubernetes API 伺服器通訊所使用的 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。

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 mody 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 ONTAPI （ 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`爲 `0（"問題 #887"）。
- ONTAP驅動程式現在接受低於 20 MiB 的磁碟區大小（ "問題 [#885" ）。
- 固定式 Trident ，可在 ONTAP SAN 驅動程式調整大小的作業期間，防止 FlexVol 磁碟區縮小。
- 修正 NFS v4.1 的磁碟區匯入失敗。

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"）。
- 新增 Amazon FSX 特性至 ONTAP 驅動程式，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

- 修正了 ONTAPI （ 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 一 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 。升級Kubernetes之前先升級Trident。

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

- 將 Amazon FSX for NetApp ONTAP 的 SMB Volume 支援新增至 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 FlexVol 磁碟區時，改善 ONTAP NAS 經濟型儲存驅動程式的效能
- 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。
- 設定或 `ontap-san-economy`驅動程式時，不應再指定 `DataLIFs `ontap-san`。

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 SAN 後端的特定 SLM 問題，以確保僅發佈屬於報告節點的 datahLIFs 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、ClusterRO容 和ClusterROlexBinding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yaml (`bundle_post_1_25.yaml`）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。
- ONTAP SAN IPv6 DataLIFs 現在可以在使用方括號指定的情況下運作。
- 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CrashLooper BackOff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CrashLooper BackOff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 `KUBECONFIG`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、您應確保 `KUBECONFIG`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。對於需要的 RHEL/Red Hat Enterprise Linux CoreOS （ RHCOS ）執行個體而言，這是正確的做法 `discard` "掛載選項"；請確保在您的] 中包含「丟棄掛載選項」，以支援線上區塊捨棄[`StorageClass`。
- 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

如果您使用的不是 Trident 26.10 版本，則可以根據 "Trident 支援生命週期" 取得先前版本的文件。

- "Trident 26.06"
- "Trident 26.02"
- "Trident 25.10"
- "Trident25.06"
- "Trident 25.02"
- "Trident 24.10"
- "Trident 24.06"
- "Trident 24.02"
- "Trident 23.10"
- "Trident 23.07"
- "Trident 23.04"
- "Trident 23.01"

NetApp Trident 對 ONTAP ASA r2 儲存系統的支援

NetApp Trident 25.02 及更高版本支援 NetApp ASA r2 系統作為儲存後端。如需更多資訊，請參閱 "ASA r2 系統"。

ASA r2 系統需要 `ontap-san` 驅動程式。Trident 不支援 ASA r2 系統的 `ontap-san-economy` 驅動程式。

當您在後端設定中將 `ontap-san`指定為 `storageDriverName`時，Trident 會自動偵測 ASA r2 儲存系統。

Trident 為 ASA r2 系統提供有限的資料保護，並配備 Trident protect。

支援的 SAN 傳輸協定取決於您的 Trident 版本：

- Trident 25.02 及更高版本支援 iSCSI 。
- Trident 25.06 及更新版本除了支援 iSCSI 外，還支援 NVMe/TCP。
- Trident 25.10 及更高版本除了支援 iSCSI 和 NVMe/TCP 外，還支援 FC。

您必須為 ONTAP 後端儲存的儲存虛擬機器（SVM）指派至少一個 Aggregate。請參閱 "在 ASA r2 系統中為 SVM 指派 Aggregate" 以取得相關說明。

支援的作業

- 配置持久磁碟區（PV）
- 動態磁碟區配置
- 建立和刪除磁碟區
- 複製磁碟區
- 擴充磁碟區
- 管理儲存類別

不支援的作業

- Linux 統一金鑰設定（LUKS）加密
- SnapMirror Volume 複寫
- 限制 Aggregate 使用量
- 空間保留模式
- 快照
- 分層

如需更多資訊，請參閱 "SAN組態選項與範例ONTAP"。

已知問題

已知問題可識別可能導致您無法成功使用本產品版本的問題。

下列已知問題會影響目前的版本：

從 **Trident 22.10** 或更早版本升級後，**Volume** 仍保持附加狀態

此問題會影響從 Trident 22.10 或更早版本直接升級至 Trident 26.10 且 Volume 已附加的客戶。從 Trident 23.01 或更高版本升級不受影響。

升級後，受影響的 Volume 可能無法分離。使用 ReadWriteOnce（RWO）Volume 的 Pod 無法移動到另一個節點，並可能報告多重掛載錯誤。刪除 PersistentVolume（PV）也可能停滯。

若要解決此問題，請在擁有過時 Volume 附件的節點上，從該 Volume 的 `globalmount`目錄中刪除殘留的 `volumePublishInfo.json`檔案。

VolumeSnapshots 未達 **ReadyToUse** 狀態

> `csi-snapshotter` sidecar 存在已知問題。在所有 Kubernetes 版本中、VolumeGroupSnapshots `v1beta1` 會阻止 VolumeSnapshots 達到 `ReadyToUse`狀態。

有兩種因應措施：

1. 刪除 VolumeGroupSnapshots CRD 以停用 VolumeGroupSnapshots，然後重新安裝 Trident。
2. 安裝 VolumeGroupSnapshots `v1beta2` 和 `snapshot-controller` 版本 8.4.0 或更高版本，然後重新安裝 Trident。VolumeGroupSnapshots 無法在低於 v1.34 的 Kubernetes 版本上運行。

還原大型檔案的還原還原備份可能會失敗

從使用 Restic 建立的 Amazon S3 備份中還原 30GB 或更大的檔案時，復原作業可能會失敗。作為變通方法，可以使用 Kopia 作為資料移動工具備份資料（Kopia 是備份的預設資料移動工具）。請參閱 "使用Trident Protect保護應用程式"以取得說明。

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 可在公有雲或本地端，跨所有主流 NetApp 儲存平台實現儲存資源的使用與管理，包括本地端 ONTAP 叢集（AFF、FAS 和 ASA）、ONTAP Select、Cloud Volumes ONTAP、Element 軟體（NetApp HCI、SolidFire）、Azure NetApp Files、Amazon FSx for NetApp ONTAP，以及 Google Cloud NetApp Volumes。

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"。

支援的 Kubernetes 平台

Trident 支援多種 Kubernetes 發行版和平台。

支援的平台包括：* Upstream Kubernetes * Red Hat OpenShift * SUSE Harvester 1.7.0 (ONTAP iSCSI)

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。

ONTAP 系統具有 All Flash ，混合式或全硬碟組態，並提供許多不同的部署模式：內部部署 FAS ， AFF 和 ASA 叢集， ONTAP Select 和 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。

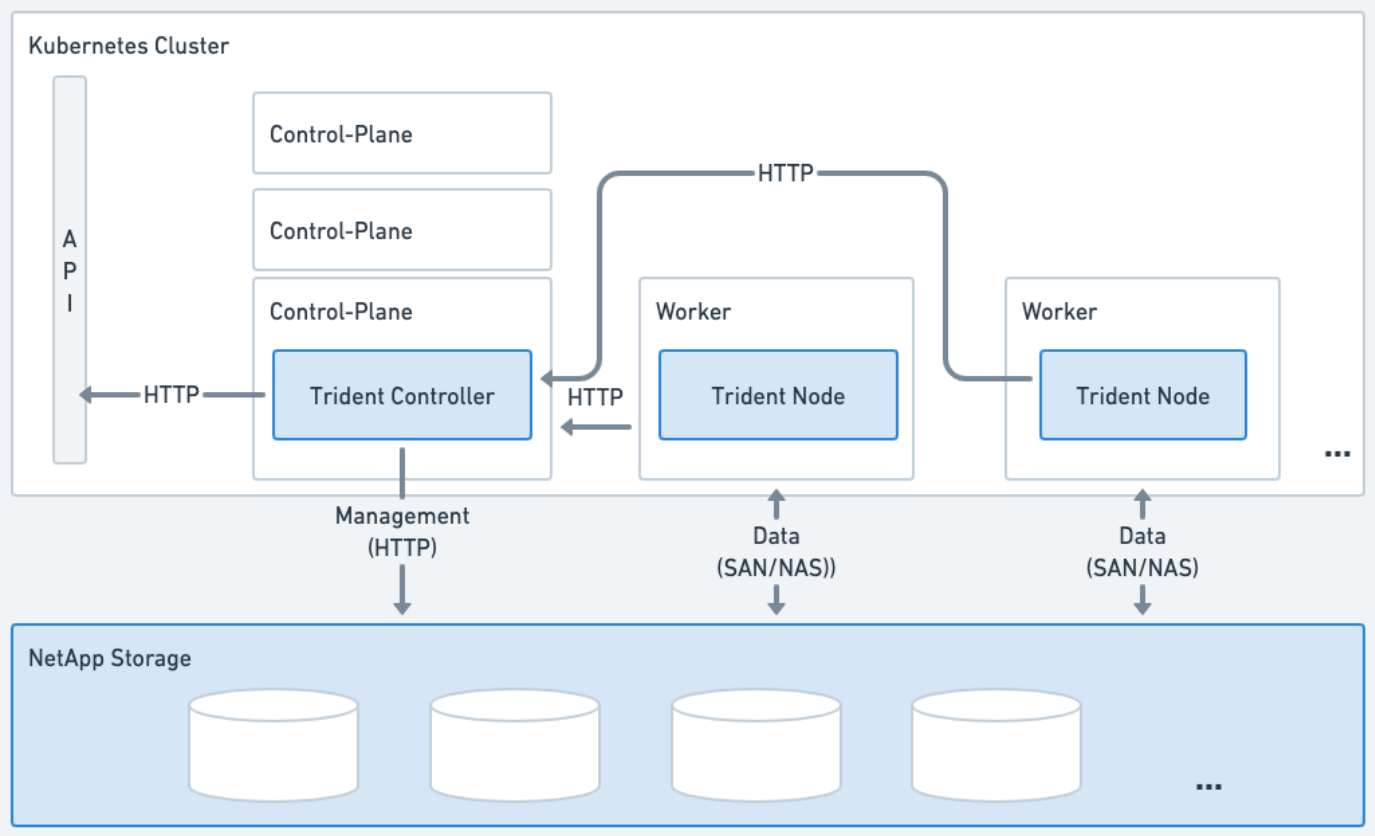


圖 **1. Trident** 部署在 **Kubernetes** 叢集上

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。

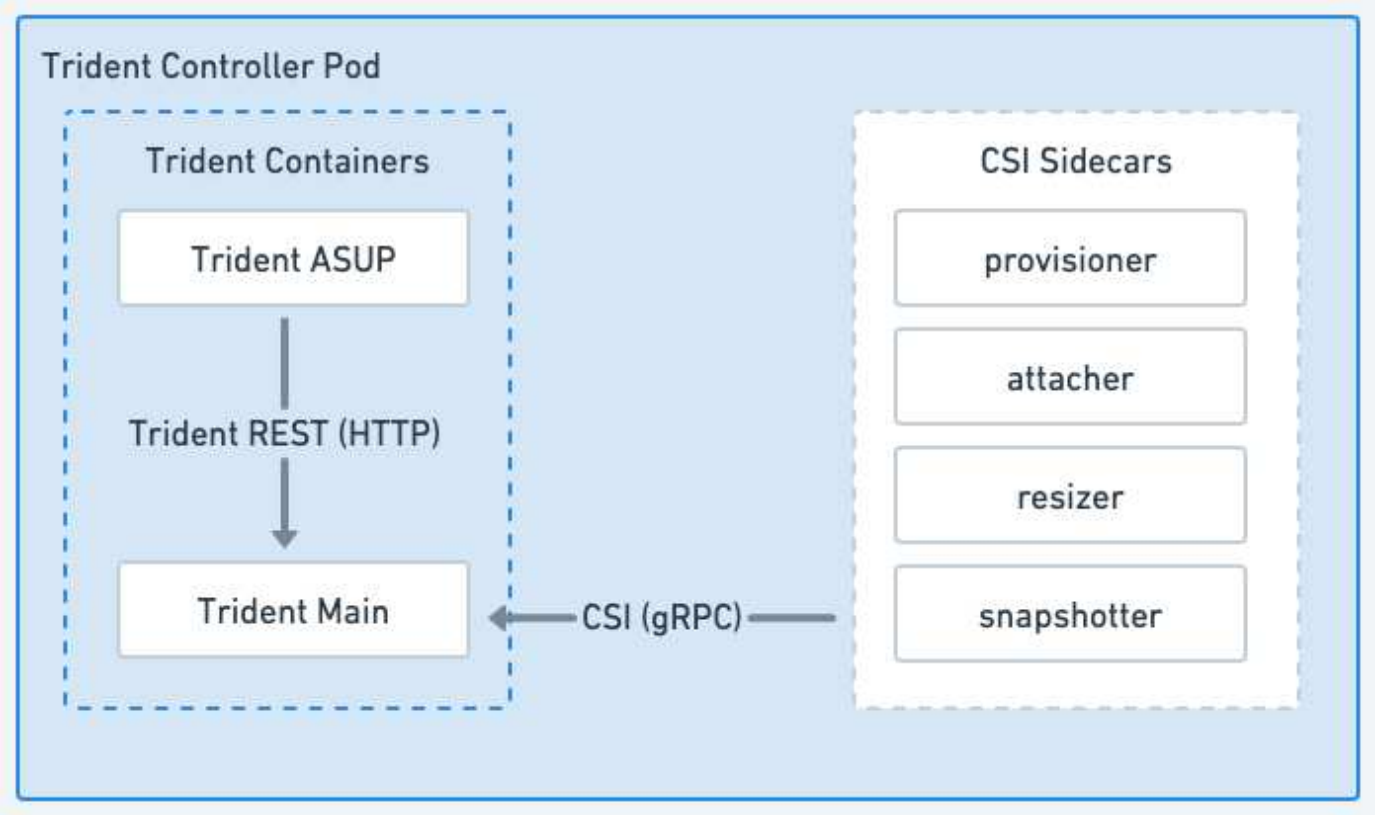


圖 **2. Trident** 控制器 **Pod** 圖表

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

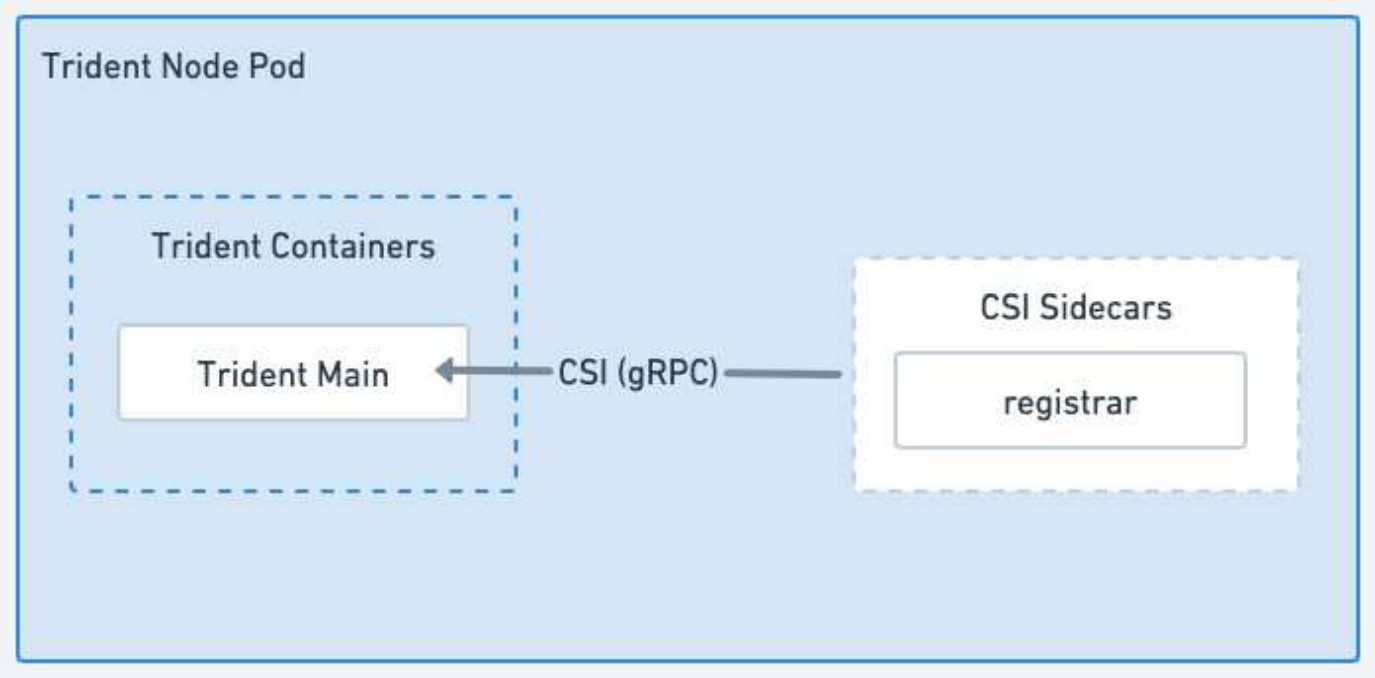


圖 **3. Trident Node Pod** 圖表

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

Trident Volume 配置流程

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 * 。

Trident Volume 快照建立

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

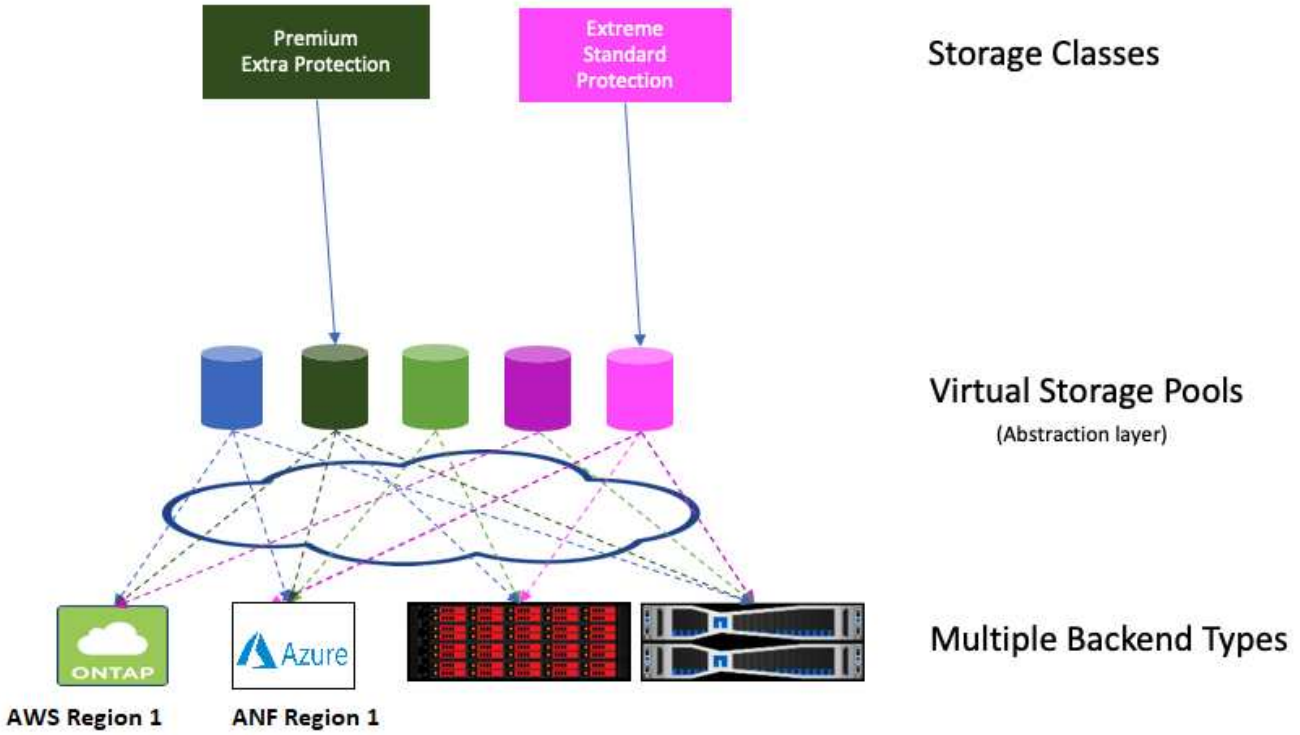
- 對於 `ontap-nas`， `ontap-san` ， 和 `azure-netapp-files`在驅動程式中，每個持久性磁碟區 (PV) 都會對應到一個FlexVol volume。因此，磁碟區快照被創建為NetApp快照。NetApp快照技術比同類快照技術具有更高的穩定性、可擴展性、可恢復性和效能。這些快照副本在創建所需時間和儲存空間方面都非常有效率。
- 適用於 `ontap-nas-flexgroup` 驅動程式、每個持續Volume（PV）都會對應FlexGroup 至一個功能區。因此、磁碟區快照會建立為NetApp FlexGroup 的「資訊快照」。相較於競爭的快照技術、 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。
- 對於 `ontap-san-economy`驅動程式， PV 會對應至在共享 FlexVol Volume Volume Volume ， PV 的 Volume Snapshot 上建立的 LUN ，這是透過執行相關 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

使用Trident，您可以利用 VolumeSnapshots 從中建立新的 PV。使用FlexClone技術從這些快照建立 PV（適用於受支援的ONTAP後端）執行。從快照建立 PV 時，後備磁碟區是快照父磁碟區的FlexClone 。這 `solidfire-san`驅動程式使用 Element 軟體磁碟區克隆從快照建立 PV。它在這裡從 Element 快照創建一個克隆。

Trident 虛擬儲存設備資源池組態

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。

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。

可以使用以下字元來定義虛擬池標籤：

- 大寫字母 `A-Z`
- 小寫字母 `a-z`
- 數位 `0-9`
- 底線 `_`
- 連字符 `-`

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
《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

Trident Volume 存取群組管理

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 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"

2 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"

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**

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。

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。

Trident 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

- Trident 26.10 支援 Kubernetes 1.27 至 1.37 版本。升級 Kubernetes 前，請先查看下方特定版本的

映像清單。

- 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 met金 片 1.16
- Kubernetes 1.27 - 1.37
- OpenShift 4.12、4.14 - 4.22（如果您打算在 OpenShift 4.19 版本中使用 iSCSI 節點準備功能，則支援的最低 Trident 版本為 25.06.1。）

> Trident繼續支援舊版 OpenShift 版本，與"Red Hat 擴充更新支援 (EUS) 發布生命週期"，即使它們依賴上游不再受官方支援的 Kubernetes 版本。在這種情況下安裝Trident時，您可以放心地忽略有關 Kubernetes 版本的任何警告訊息。

- Rancher Kubernetes Engine 2（RKE2）v1.28.x - 1.37.x
- 適用於 NFSv4、iSCSI 和 NVMe/TCP 的 Talos OS。每種協定都需要下列 Talos 系統擴充；請與 Sidero Labs 確認這些要求。
  - NFSv4： `siderolabs/nfs-utils`
  - iSCSI： `siderolabs/iscsi-tools`、 `siderolabs/multipath-tools`和 `siderolabs/trident-iscsi-tools`
  - NVMe/TCP： `siderolabs/nvme-cli`、 `siderolabs/util-linux-tools`和 `siderolabs/trident-iscsi-tools`

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 1.25 升級至 1.26 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
- NetApp All SAN Array ASA （ESAN）
- 在NetApp完全或有限支援下執行ONTAP版本的本機FAS、 AFF或ASA r2（iSCSI、NVMe/TCP 和 FC）。看 "軟體版本支援"。
- NetApp HCI / Element軟體11或更新版本

Trident 支援 KubeVirt 和 OpenShift 虛擬化

支援的儲存驅動程式：

Trident 支援下列適用於 KubeVirt 和 OpenShift 虛擬化的 ONTAP 驅動程式：

- ONTAP-NAS
- ONTAP SAN （ iSCSI ， FCP ， NVMe over TCP ）
- ONTAP SAN 經濟型（僅限 iSCSI ）

要考量的重點：

- 將儲存類別更新為在 OpenShift 虛擬化環境中使用 `fsType`參數（例如： `fsType: "ext4"`）。如有需要，請將 Volume 模式設定為使用中的參數 `dataVolumeTemplates`明確封鎖 `volumeMode=Block`，以通知 CDI 建立區塊資料磁碟區。
- 區塊儲存驅動程式的 _rwx 存取模式 _ ： ONTAP SAN （ iSCSI ， NVMe / TCP ， FC ）和 ONTAP SAN 經濟（ iSCSI ）驅動程式僅支援「 volumemode ：區塊」（原始裝置）。對於這些驅動程式，無法使用此參數，因為這些 `fstype`磁碟區是以原始裝置模式提供。
- 對於需要 rwx 存取模式的即時移轉工作流程，支援下列組合：
  - NFS + `volumeMode=Filesystem`
  - iSCSI + `volumeMode=Block` （原始裝置）
  - NVMe / TCP + `volumeMode=Block` （原始裝置）
  - FC + `volumeMode=Block` （原始裝置）

Trident 對 Kubernetes 上的 Kubernetes 和 OpenShift 託管控制平面的支援

Trident 完全符合 Kubernetes on Kubernetes 拓撲的資格。其節點作為 KubeVirt 虛擬機器在基礎架構（主機）叢集上執行的 Kubernetes 叢集。此拓撲是託管控制平面產品的基礎，例如 OpenShift 虛擬化上的 OpenShift 託管控制平面（HCP）。

此組態在早期版本中有效，但完整資格判斷從 26.06 版本開始。

客戶（租戶）叢集使用 KubeVirt CSI 驅動程式(`kubevirt-csi`作為其儲存提供者。當客戶叢集中的工作負載請求 `PersistentVolumeClaim`時，KubeVirt CSI 驅動程式會在基礎架構叢集中建立對應的磁碟區。Trident 從 ONTAP 佈建磁碟區，並將該磁碟區熱插拔至客戶節點虛擬機器中。Trident 僅在基礎架構叢集中運作。客戶叢集透過 KubeVirt CSI 驅動程式間接使用 ONTAP 儲存設備。

要求：

- Trident 安裝在具有支援的 ONTAP 後端的基礎架構叢集上。
- 基礎架構叢集中存在一個 Trident 儲存類別，供 KubeVirt CSI 驅動程式將客體叢集磁碟區對應至該儲存類

別。

- KubeVirt 或 OpenShift 虛擬化部署在基礎架構叢集上，且 KubeVirt CSI 驅動程式安裝在每個客體叢集中。

元件	版本
Trident	26.10
Kubernetes（基礎架構和訪客叢集）	1.37
KubeVirt	1.9.0
容器化資料導入器 (CDI)	1.65

要考量的重點：

- 在客體叢集中，磁碟區以熱插拔區塊裝置的形式顯示在客體節點虛擬機器上。
- KubeVirt CSI 驅動程式工作流程不需要在客體叢集中使用 Trident。您需要在基礎架構叢集中設定 ONTAP 後端和儲存類別。

直接在訪客叢集中安裝 **Trident**：

您也可以直接在客戶（租用戶）叢集中安裝 Trident。當您需要 Trident 的完整功能集（例如快照、複製、磁碟區調整大小和匯入，而 `kubevirt-csi`本身不提供這些功能）時，請將 Trident 安裝在客戶叢集中。

在客體叢集中，請使用基於 IP 的協定：NFS、iSCSI 和 NVMe/TCP。客體叢集不支援光纖通道 (FC)，因為客體節點是虛擬機器，沒有直接的 FC 連線。

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	1.27 - 1.37	否
Volume Snapshot	1.27 - 1.37	否
來自Volume Snapshot的PVc	1.27 - 1.37	否
iSCSI PV調整大小	1.27 - 1.37	否
資訊雙向CHAP ONTAP	1.27 - 1.37	否
動態匯出原則	1.27 - 1.37	否
Trident運算子	1.27 - 1.37	否
csi拓撲	1.27 - 1.37	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：

- OpenShift 容器平台在 AMD64 和 ARM64 架構上支援的 Red Hat Enterprise Linux CoreOS (RHCOS) 版本
- Red Hat Enterprise Linux (RHEL) 8 或更高版本，支援 AMD64 和 ARM64 架構

NVMe / TCP 需要 RHEL 9 或更新版本。

- Ubuntu 22.04 LTS 或更高版本，支援 AMD64 和 ARM64 架構
- Windows Server 2022
- SUSE Linux Enterprise Server (SLES) 15 或更高版本

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。

以下列出的 Trident 26.10 映像清單包含 Kubernetes 1.37 的映像。

Trident 26.10.1 所需的容器映像

Kubernetes 版本	Container映像
v1.27.0、v1.28.0、v1.29.0、v1.30.0、v1.31.0、v1.32.0、v1.33.0、v1.34.0、v1.35.0、v1.36.0、v1.37.0	• docker.io/netapp/trident:26.10.1 • docker.io/netapp/trident-autosupport:26.10 • docker.io/netapp/trident-operator:26.10.1 （選用） • registry.k8s.io/sig-storage/csi-attacher:v4.11.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.17.0 • registry.k8s.io/sig-storage/csi-provisioner:v6.3.0 • registry.k8s.io/sig-storage/csi-resizer:v2.1.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.6.0

Trident 26.06.2 所需的容器映像

Kubernetes 版本	Container映像
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0, v1.35.0, v1.36.0	• docker.io/netapp/trident:26.06.2 • docker.io/netapp/trident-autosupport:26.06 • docker.io/netapp/trident-operator:26.06.2 （選用） • registry.k8s.io/sig-storage/csi-attacher:v4.11.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.18.0 • registry.k8s.io/sig-storage/csi-provisioner:v6.3.0 • registry.k8s.io/sig-storage/csi-resizer:v2.1.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.6.0

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。

關於 **Trident 26.10** 的關鍵資訊

- 您必須閱讀下列有關 Trident 的重要資訊。 *

▼ **** 的 **Trident ** 相關資訊

- Trident 現在支援 Kubernetes 1.37。請先升級 Trident，再升級 Kubernetes。
- 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：

- 對執行受支援 Kubernetes 版本且已啟用相關功能需求的受支援 Kubernetes 叢集擁有完整權限。詳情請查看"需求"。
- 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"。
- 叢集必須支援特殊權限的工作負載。

如果您尚未熟悉"基本概念"、現在正是您熟悉的好時機。

選擇您的安裝方法

選擇適合您的安裝方式。在做出決定之前，您還應該考慮 "在方法之間移動"。

使用**Trident**運算子

無論手動部署或使用 Helm，Trident operator 都是簡化安裝和動態管理 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 `TridentOrchestrator`自訂資源（CR）中的屬性。

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`。
- 如果您有一個基於手動操作員的部署，並且想要使用基於 Helm 的Trident操作員部署，則應先手動卸載操作員，然後執行 Helm 安裝。這使得 Helm 能夠使用所需的標籤和註解來部署Trident運算子。如果不這樣做，基於 Helm 的Trident操作員部署將因標籤驗證錯誤和註解驗證錯誤而失敗。
- 如果你有一個 `tridentctl`基於部署，無需卸載Trident即可進行基於 Helm 或基於 Operator 的部署。

其他已知組態選項

在 VMware Tanzu Portfolio 產品上安裝 Trident 時：

- 「-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"。

關於 **Trident 26.10** 的關鍵資訊

- 您必須閱讀下列有關 Trident 的重要資訊。 *

▼ **** 的 **Trident ** 相關資訊

- Trident 26.10 支援 Kubernetes 1.27 至 1.37 版本。升級 Kubernetes 之前，請檢查 "Trident 需求" 中

的特定版本映像清單。

- 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

檢查 "安裝總覽" 以確保滿足安裝先決條件並為您的環境選擇了正確的安裝選項。

開始之前

在開始安裝之前，請登入 Linux 主機並驗證它是否正在管理運作中的 "支援的Kubernetes叢集"，以及您是否擁有必要的權限。

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

▼ **Details**

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/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

步驟**2**：建立 **`TridentOrchestrator`** 客戶需求日

創建 `TridentOrchestrator`自訂資源定義（CRD）。您將創建一個 `TridentOrchestrator`稍後自訂資源。使用適當的 CRD YAML 版本 `deploy/crds`創建 `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`。

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

kubectl 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:26.10
    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:26.10.1
  Message:                   Trident installed  Namespace:
trident
  Status:                    Installed
```

```
  Version:                 v26.10.0
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 `TridentOrchestrator` 。
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 `spec.uninstall=true`。
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「TridentOrchestrator」。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        6/6     Running   0
1m
trident-node-linux-mr6zc                   2/2     Running   0
1m
trident-node-linux-xrp7w                   2/2     Running   0
1m
trident-node-linux-zh2jt                   2/2     Running   0
```

```
1m
trident-operator-766f7b8658-ldzsv           1/1     Running   0
3m
```

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 |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

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 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

- Trident 26.10 支援 Kubernetes 1.27 至 1.37 版本。升級 Kubernetes 之前，請檢查 "Trident 需求" 中的特定版本映像清單。
- 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

檢查 "安裝總覽" 以確保滿足安裝先決條件並為您的環境選擇了正確的安裝選項。

開始之前

登入 Linux 主機，並驗證它是否正在管理一個正常運作的 "支援的Kubernetes叢集"，以及您是否擁有必要的權限。

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

▼ **Details**

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/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

步驟**2**：建立 **`TridentOrchestrator`** 客戶需求日

創建 `TridentOrchestrator`自訂資源定義（CRD）。您將創建一個 `TridentOrchestrator`稍後自訂資源。使用適當的 CRD YAML 版本 `deploy/crds`創建 `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:26.10.1`以反映您的映像登錄的位置。您的 "Trident和csi影像" 可以位於一個登錄中，也可以位於不同的登錄中，但所有 CSI 映像必須位於同一個登錄中。例如：

- `image: <your-registry>/trident-operator:26.10.1` 如果您的所有映像都位於同一個登錄中。
- `image: <your-registry>/netapp/trident-operator:26.10.1` 如果您的 Trident 映像與 CSI 映像位於不同的登錄檔。

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:26.10"
tridentImage: "<your-registry>/trident:26.10.1"
```

不同登錄中的映像

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.10"
tridentImage: "<your-registry>/trident:26.10.1"
```

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。

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

kubectl 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:26.10
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.10.1
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.10
    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:26.10.1
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.10.0
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 `TridentOrchestrator` 。
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 `spec.uninstall=true`。
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「TridentOrchestrator」。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   6/6     Running   0
1m
trident-node-linux-mr6zc              2/2     Running   0
1m
trident-node-linux-xrp7w              2/2     Running   0
1m
trident-node-linux-zh2jt              2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv     1/1     Running   0
3m
```

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 |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

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"。

關於 **Trident 26.10** 的關鍵資訊

- 您必須閱讀下列有關 Trident 的重要資訊。 *

▼ 的 Trident 相關資訊

- Trident 26.10 支援 Kubernetes 1.27 至 1.37 版本。升級 Kubernetes 之前，請檢查 "Trident 需求" 中的特定版本映像清單。
- 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。

檢查 "安裝總覽" 以確保滿足安裝先決條件並為您的環境選擇了正確的安裝選項。

開始之前

除了 "部署先決條件" 之外，您還需要 "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` 並為您的部署指定一個名稱，如下例所示，其中 `100.2610.0` 是您要安裝的 Trident 版本。

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace <trident-namespace>
```

> 如果您已經為Trident建立命名空間、則「-cree-namespacre」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
`--values` （或 `-f`）	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。
`--set`	在命令列上指定置換。

例如，若要變更 `debug` 的預設值，請執行下列命令，其中 `100.2610.0` 是您要安裝的 Trident 版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--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
nodeSelector	Pod 指派的節點標籤	
podAnnotations	Pod 註釋	
deploymentAnnotations	部署註釋	
tolerations	Pod 指派的容錯功能	

affinity	Pod 指派的關聯性	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		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。

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。	

選項	說明	預設
「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」
`tridentExcludeAutosupport`	當設定為 `true`時，Trident 在安裝過程中不會設定 AutoSupport Container。	「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"`

選項	說明	預設
tridentDisableAuditLog	停用 Trident 稽核記錄程式。	"真的"
tridentLogLevel	允許將 Trident 的日誌級別設置爲：trace、debug、info、warn error`或 `fatal。	"info"
tridentDebug	允許將Trident的日誌等級設定為debug。您可以透過與節點健康檢查 (NHC) 操作符整合來自動執行強制分離程序。有關信息，請參閱"使用Trident實現有狀態應用程式的故障轉移自動化"。	「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」
enableDataLIFRefresh	當 LIF 新增、移除或指派新位址時，重新整理已發佈 Volume 的 NVMe 資料 LIF。適用於統一 ONTAP 上的 ontap-san`搭配 `sanType: nvme，以及 ASA r2 上的 NVMe/TCP。如需詳細資訊，請參閱 "重新整理 NVMe 資料 LIFs"。	「假」
excludePodSecurityPolicy	不建立營運商 Pod 安全性原則。	「假」
cloudProvider	在 AKS 叢集上使用代管身分或雲端身分時設定為 "Azure"，在 EKS 叢集上使用雲端身分時設定為 "AWS"，或在 GKE 上使用雲端身分或在 OpenShift 上使用工作負載身分聯合時設定為 "GCP"。	""

選項	說明	預設
`cloudIdentity`	在 AKS 叢集上使用雲端身分時，請設定為工作負載身分（"azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx"）。在 EKS 叢集上使用雲端身分時，請設定為 AWS IAM 角色（'"eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role"'）。在 GKE 上使用雲端身分時，請設定為 GKE 雲端身分註解。請勿為 OpenShift 工作負載身分聯合進行設定。	""
`gcpWIFAudience`	在 OpenShift 上與 Google Cloud Workload Identity Federation 搭配使用的預計服務帳戶權杖受眾。請使用供應商的規範資源名稱。如需詳細資訊，請參閱 "設定 Google Cloud NetApp Volumes 與工作負載身分聯盟"。	""
`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`	使Trident能夠準備 Kubernetes 叢集的節點，以使用指定的資料儲存協定管理磁碟區。 *現在，`iscsi`是唯一支援的值。 *注意：從 OpenShift 4.19 開始，此功能支援的最低Trident版本為 25.06.1。	
`enableConcurrency`	啟用並行 Trident 控制器作業，以提高處理量。如需支援驅動程式的詳細資訊，請參閱連結：https://docs.netapp.com/us-en/trident/trident-use/controller-scalability.html#controller-scalability[控制器擴充性文件]。	錯
`k8sAPIQPS`	控制器與 Kubernetes API 伺服器通訊時所使用的每秒查詢數 (QPS) 限制。 Burst 值會根據 QPS 值自動設定。	`100`; 選修的

選項	說明	預設

resources	設定 Kubernetes 對Trident控制器、節點和操作員 pod 的資源限制和請求。您可以為每個容器和邊車配置 CPU 和內存，以管理 Kubernetes 中的資源分配。
	
	有關配置資源請求和限制的更多信息，請參閱"Pod 和容器的資源管理"。
	
	• 請勿變更任何容器或欄位的名稱。
	• 請勿更改縮排 - YAML 縮排對於正確解析至關重要。
	
	• 預設不套用任何限制－只有請求才有預設值，如果未指定則會自動套用。
	• 容器名稱依其在 pod 規格中的顯示方式列出。
	• 邊車列在每個主貨櫃下方。
	• 查看TORC的 `status.CurrentInstallationParams `用於查看目前應用值的欄位。
		resources:
		  controller:
		  trident-main:
		  requests:
		  cpu: 10m
		  memory: 80Mi
		  limits:
		  cpu:
		  memory:
		  csi-provisioner:
		  requests:
		  cpu: 2m
		  memory: 20Mi
		  limits:
		  cpu:
		  memory:
		  csi-attacher:
		  requests:
		  cpu: 2m
		  memory: 20Mi
		  limits:
		  cpu:
		  memory:
		  csi-resizer:
		  requests:
		  cpu: 3m
		  memory: 20Mi
		  limits:
		  cpu:
		  memory:
		  csi-snapshotter:
		  requests:
		  cpu: 2m
		  memory: 20Mi
		  limits:
		  cpu:
		  memory:
		  trident-autosupport:
		  requests:
		  cpu: 1m
		  memory: 30Mi
		  limits:
		  cpu:
		  memory:
		  node:

選項	說明	預設
`httpsMetrics`	為 Prometheus 指標端點啟用 HTTPS。	錯
「主機網路」	為Trident控制器啟用主機聯網功能。在多宿主網路中，當需要分離前端和後端流量時，這非常有用。	錯

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"。

關於 **Trident 26.10** 的關鍵資訊

- 您必須閱讀下列有關 Trident 的重要資訊。 *

▼ **** 的 **Trident ** 相關資訊

- Trident 26.10 支援 Kubernetes 1.27 至 1.37 版本。升級 Kubernetes 之前，請檢查 "Trident 需求" 中的特定版本映像清單。
- 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。

檢查 "安裝總覽" 以確保滿足安裝先決條件並為您的環境選擇了正確的安裝選項。

開始之前

除了 "部署先決條件" 之外，您還需要 "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`並為您的部署和映像登錄位置指定名稱。您的 "Trident和csi影像" 可以位於一個登錄中，也可以位於不同的登錄中，但所有 CSI 映像必須位於同一個登錄中。在範例中，`100.2610.0`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
100.2610.0 --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
100.2610.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.10.1 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.10
--set tridentImage=<your-registry>/trident:26.10.1 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

如果您已經為Trident建立命名空間、則「-cree-namespacre」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
`--values` （或 `-f`）	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。
`--set`	在命令列上指定置換。

例如，若要變更 `debug` 的預設值，請執行下列命令，其中 `100.2610.0` 是您要安裝的 Trident 版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
```

```
--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 100.2610.0
--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
`nodeSelector`	Pod 指派的節點標籤	
`podAnnotations`	Pod 註釋	
`deploymentAnnotations`	部署註釋	
`tolerations`	Pod 指派的容錯功能	

選項	說明	預設

`affinity`	Pod 指派的關聯性	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		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。

`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。	

選項	說明	預設
「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`	覆寫大多數 Kubernetes API 操作的預設 180 秒逾時（如果非零，則以秒為單位）。 這 `tridentK8sTimeout` 參數僅適用於Trident 安裝。	`180`
`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」
`tridentExcludeAutosupport`	當設定為 `true`時，Trident 在安裝過程中不會設定 AutoSupport Container。	「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。	"

選項	說明	預設
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 的日誌級別設置爲：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。您可以透過與節點健康檢查 (NHC) 操作符整合來自動執行強制分離程序。有關信息，請參閱"使用Trident實現有狀態應用程式的故障轉移自動化"。	「假」
enableDataLIFRefresh	當 LIF 新增、移除或指派新位址時，重新整理已發佈 Volume 的 NVMe 資料 LIF。適用於統一 ONTAP 上的 ontap-san`搭配 `sanType: nvme，以及 ASA r2 上的 NVMe/TCP。如需詳細資訊，請參閱 "重新整理 NVMe 資料 LIFs"。	「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。 * 從 OpenShift 4.19 開始，此功能支援的最低Trident版本為 25.06.1。	

選項	說明	預設

`resources`	設定 Kubernetes 對Trident控制器、節點和操作員 pod 的資源限制和請求。您可以為每個容器和邊車配置 CPU 和內存，以管理 Kubernetes 中的資源分配。
	
	有關配置資源請求和限制的更多信息，請參閱"Pod 和容器的資源管理"。
	
	• 請勿變更任何容器或欄位的名稱。
	• 請勿更改縮排 - YAML 縮排對於正確解析至關重要。
	
	• 預設不套用任何限制－只有請求才有預設值。
	• 容器名稱依其在 pod 規格中的顯示方式列出。
	• 邊車列在每個主貨櫃下方。
	• 查看TORC的 `status.CurrentInstallationParams `用於查看目前應用值的欄位。
		`resources:`
		`  controller:`
		`    trident-main:`
		`      requests:`
		`        cpu: 10m`
		`        memory: 80Mi`
		`      limits:`
		`          cpu:`
		`          memory:`
		`    csi-provisioner:`
		`      requests:`
		`        cpu: 2m`
		`        memory: 20Mi`
		`      limits:`
		`        cpu:`
		`        memory:`
		`    csi-attacher:`
		`      requests:`
		`        cpu: 2m`
		`        memory: 20Mi`
		`      limits:`
		`        cpu:`
		`        memory:`
		`    csi-resizer:`
		`      requests:`
		`        cpu: 3m`
		`        memory: 20Mi`
		`      limits:`
		`        cpu:`
		`        memory:`
		`    csi-snapshotter:`
		`      requests:`
		`        cpu: 2m`
		`        memory: 20Mi`
		`      limits:`
		`        cpu:`
		`        memory:`
		`    trident-autosupport:`
		`      requests:`
		`        cpu: 1m`
		`        memory: 30Mi`
		`      limits:`
		`        cpu:`
		`        memory:`

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在叢集中的每個工作節點上以單一控制器 pod 和節點 pod 的形式運作。節點 pod 必須運行在您希望掛載Trident磁碟區的任何主機上。

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
`namespace`	安裝 Trident 的命名空間	`"default"`
「Debug」	啟用 Trident 的偵錯功能	「假」
`enableForceDetach`	`ontap-san`， `ontap-san-economy` ， `ontap-nas` ， 和 `ontap-nas-economy`僅有的。與 Kubernetes 非優雅節點關閉 (NGNS) 配合使用，使叢集管理員能夠在節點發生故障時安全地將掛載磁碟區的工作負載遷移到新節點。有關信息，請參閱"使用Trident實現有狀態應用程式的故障轉移自動化"。	「假」
`enableDataLIFRefresh`	當 LIF 被新增、移除或指派新位址時，可為已發佈的 Volume 重新整理 NVMe 資料 LIF。適用於統一 ONTAP 上的 `ontap-san`搭配 `sanType: nvme`，以及 ASA r2 上的 NVMe/TCP。如需詳細資訊，請參閱"重新整理 NVMe 資料 LIFs"。	「假」
`windows`	設定為 `true` 可在Windows工作節點上安裝。	「假」
`cloudProvider`	在 AKS 叢集上使用託管身分識別或雲端身分識別時、請設定為 `"Azure"`。在 EKS 叢集上使用雲端身分識別時設為 `"AWS"`。在 GKE 叢集上使用雲端身分識別時設為 `"GCP"`。	`""`

參數	說明	預設
`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-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」）。在 GKE 叢集上使用雲端身分識別時，請設定為雲端身分識別（「iam.GKE 。 IO/GCP 服務帳戶： xxxx@mygcpproject.iam.gserviceaccount.com 」）。	""
`gcpWIFAudience`	在 OpenShift 上與 Google Cloud Workload Identity Federation 搭配使用的預計服務帳戶權杖受眾。請使用供應商的規範資源名稱。如需詳細資訊，請參閱 "設定 Google Cloud NetApp Volumes 與工作負載身分聯盟"。	""
`IPv6`	透過 IPv6 安裝 Trident	錯
`k8sTimeout`	Kubernetes 操作逾時。 這 `k8sTimeout`參數僅適用於Trident安裝。	`180sec`
`silenceAutosupport`	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:26.10"`
「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」
`logFormat`	要使用的 Trident 記錄格式 [text,json]	`"text"`
「TridentImage」	要安裝的 Trident 映像	`"netapp/trident:26.10.1"`
「影像登錄」	內部登錄的路徑、格式 `<registry FQDN>[:port][/subpath]`	`"registry.k8s.io"`
《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

參數	說明	預設
「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
「nodePluginBolerations」	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。 * 從 OpenShift 4.19 開始，此功能支援的最低Trident版本為 25.06.1。	
`k8sAPIQPS`	控制器與 Kubernetes API 伺服器通訊時所使用的每秒查詢數 (QPS) 限制。 Burst 值會根據 QPS 值自動設定。	`100`; 選修的
`enableConcurrency`	啟用並行 Trident 控制器作業，以提高處理量。如需支援驅動程式的詳細資訊，請參閱連結：https://docs.netapp.com/us-en/trident/trident-use/controller-scalability.html#controller-scalability[控制器擴充性文件]。	`enableCOSI`
啟用 Trident Container 物件儲存介面（COSI）前端。需要將 `enableConcurrency` 設定為 `true`。Trident COSI 無法與社群 COSI 控制器同時運作。如需詳細資訊，請參閱"使用 COSI 佈建物件儲存"。	「假」	`resources`

參數	說明	預設

設定Trident控制器和節點pod 的 Kubernetes 資源限制和請求。您可以為每個容器和邊車配置 CPU 和內存，以管理 Kubernetes 中的資源分配。
 	
有關配置資源請求和限制的更多信息，請參閱"Pod 和容器的資源管理"。
 	
• 請勿變更任何容器或欄位的名稱。
• 請勿更改縮排 - YAML 縮排對於正確解析至關重要。
 	
• 預設不套用任何限制－只有請求才有預設值，如果未指定則會自動套用。
• 容器名稱依其在 pod 規格中的顯示方式列出。
• 邊車列在每個主貨櫃下方。
• 查看TORC的
	``
	resources:
	  controller:
	    trident-main:
	      requests:
	        cpu: 10m
	        memory: 80Mi
	      limits:
	          cpu:
	          memory:
	    csi-provisioner:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-attacher:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-resizer:
	      requests:
	        cpu: 3m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-snapshotter:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    trident-autosupport:
	      requests:
	        cpu: 1m
	        memory: 30Mi
	      limits:
	        cpu:
	        memory:
	  node:
	``
		httpsMetrics

參數	說明	預設
為 Prometheus 指標端點啟用 HTTPS。	錯	「主機網路」

如需格式化 Pod 參數的詳細資訊、請參閱 "將Pod指派給節點"。

組態範例

您可以使用下列屬性[組態選項]在定義時 `TridentOrchestrator`自訂您的安裝。

▼ 基本自訂配置

這個範例是在運行以下命令後創建的： `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml`該命令代表一個基本的自訂安裝：

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

▼ 節點選擇器

此範例使用節點選擇器安裝Trident 。

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

▼ **Windows** 工作節點

這個範例是在運行以下命令後創建的： `cat deploy/crds/tridentorchestrator_cr.yaml`該指令在 Windows 工作節點上安裝Trident 。

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

▼ **AKS 叢集上的託管身份**

本範例安裝Trident以在 AKS 叢集上啟用託管身分。

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

▼ **AKS 叢集上的雲端身份**

本範例在 AKS 叢集上安裝Trident以用於雲端身分。

```
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-xxxxxxxxxxx'
```

▼ **EKS 叢集上的雲端身份**

本範例在 AKS 叢集上安裝Trident以用於雲端身分。

```
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

本範例在 GKE 叢集上安裝Trident以用於雲端身分。

```
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
```

▼ **Trident**控制器和**Trident Linux** 節點 **Pod** 的 **Kubernetes** 資源請求和限製配置

本範例為Trident控制器和Trident Linux 節點 pod 設定 Kubernetes 資源請求和限制。



免責聲明：本範例中提供的請求和限制值僅用於演示目的。請根據您的環境和工作負載需求調整這些值。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
```

```
        limits:
          cpu: 200m
          memory: 256Mi
      # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
```

```
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
```

▼ **Kubernetes資源請求和限製配置，適用於Trident控制器以及Trident Windows和Linux節點Pod**

本範例為Trident控制器和Trident Windows 和 Linux 節點 pod 設定 Kubernetes 資源請求和限制。



免責聲明：本範例中提供的請求和限制值僅用於演示目的。請根據您的環境和工作負載需求調整這些值。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  windows: true
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
        # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
```

```
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
      windows:
        trident-main:
```

```
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 200m
            memory: 128Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 100m
            memory: 128Mi
        liveness-probe:
          requests:
            cpu: 2m
            memory: 40Mi
          limits:
            cpu: 50m
            memory: 64Mi
```

使用tridentctl安裝

使用 tridentctl 安裝 Trident

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"。

關於 Trident 26.10 的關鍵資訊

- 您必須閱讀下列有關 Trident 的重要資訊。 *

▼ 的 Trident 相關資訊

- Trident 26.10 支援 Kubernetes 1.37。請先升級 Trident，再升級 Kubernetes。
- 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`

檢查 "安裝總覽" 以確保滿足安裝先決條件並為您的環境選擇了正確的安裝選項。

開始之前

在開始安裝之前，請登入 Linux 主機並驗證它是否正在管理運作中的 "支援的Kubernetes叢集"，以及您是否擁有必要的權限。

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

▼ **Details**

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/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.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:26.10 --trident
-image <your-registry>/trident:26.10.1
```

不同登錄中的映像

```
./tridentctl install -n trident --image-registry <csi-registry>
--autosupport-image <autosupport-registry>/trident-autosupport:26.10
--trident-image <trident-registry>/trident:26.10.1
```

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=26.10.1
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>`沒有 **Running** 狀態，則平台未安裝。請使用 `-d`來"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 |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

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

有關信息，請參閱"使用Trident實現有狀態應用程式的故障轉移自動化"。

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

▼ 啟用並發**Trident**控制器操作

若要啟用並行 Trident 控制器作業以提高處理量，請在安裝期間新增 `--enable-concurrency` 選項，如本範例所示。如需支援驅動程式的詳細資訊，請參閱連結： 控制器擴充性文件。

```
tridentctl install --enable-concurrency -n trident
```

▼ 重新整理 **NVMe** 資料 **LIFs**

當 LIF 組態變更時，若要重新整理已發佈 Volume 的 NVMe 資料 LIF，請使用 `--enable-datalif-refresh` 安裝 Trident。如需支援的驅動程式與組態詳細資料，請參閱 "重新整理 NVMe 資料 LIFs"。

```
tridentctl install --enable-datalif-refresh -n trident
```

自訂 **Trident 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換器 `<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「-generame-custom-yaml」參數、可在安裝程式的「設置」目錄中建立下列Yaml檔案：

- 《三叉叢集繫結.yaml》
- 《三叉部署：yaml》
- 《三叉怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪怪
- 《Trident - clusterroy.yaml》
- 《三叉魔魔怪、yaml》
- 《Trident服務：yaml》
- 《Trident命名空間.yaml》
- 「Trident服務帳戶」。yaml
- `trident-resourcequota.yaml*`

產生這些檔案之後、您可以根據自己的需求加以修改、然後使用「-use自訂-yaml」來安裝自訂部署。

```
./tridentctl install -n trident --use-custom-yaml
```

使用 **OpenShift** 認證營運商進行安裝

使用 **OpenShift OperatorHub** 安裝 **Trident**

如果您使用 Red Hat OpenShift ，則可以使用 Red Hat 認證的營運者來安裝 NetApp Trident 。請使用此程序，從 Red Hat OpenShift Container Platform 安裝 Trident 。

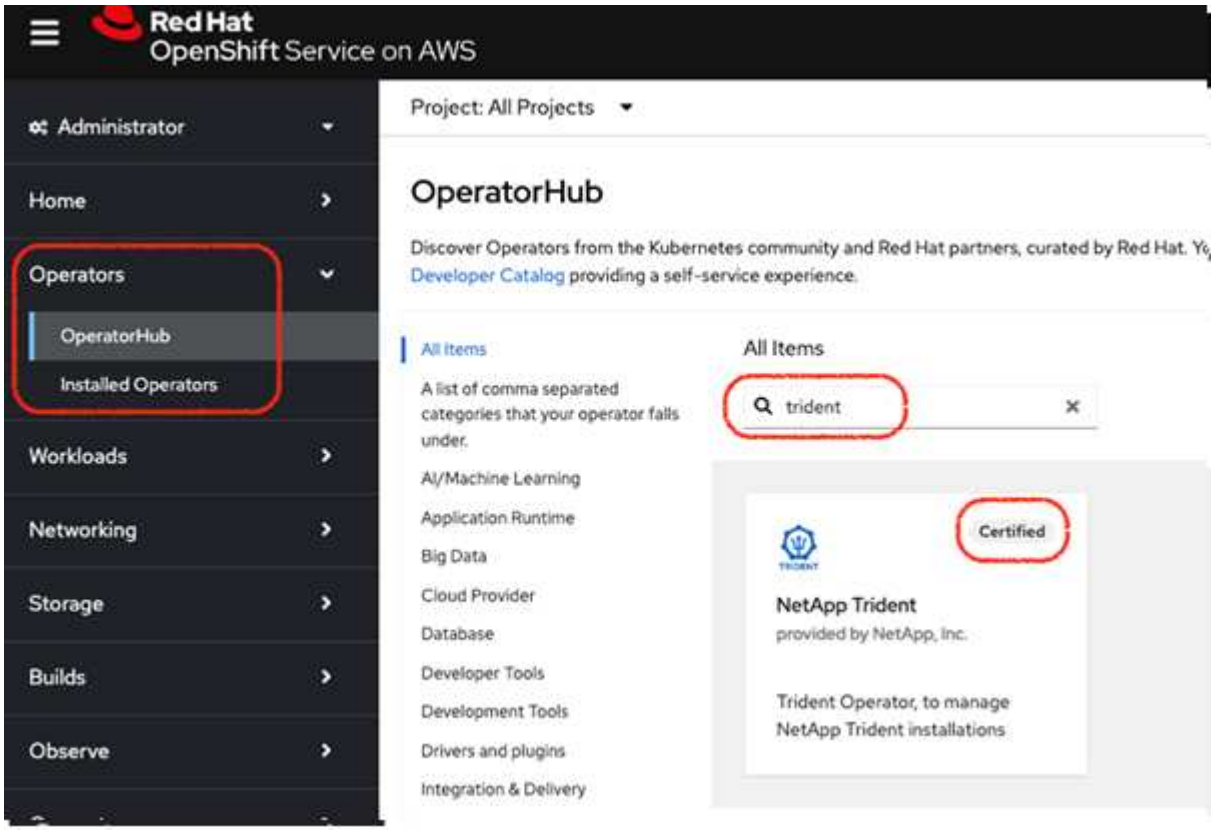
開始之前

開始安裝之前"準備好您的環境以進行 Trident 安裝"，。

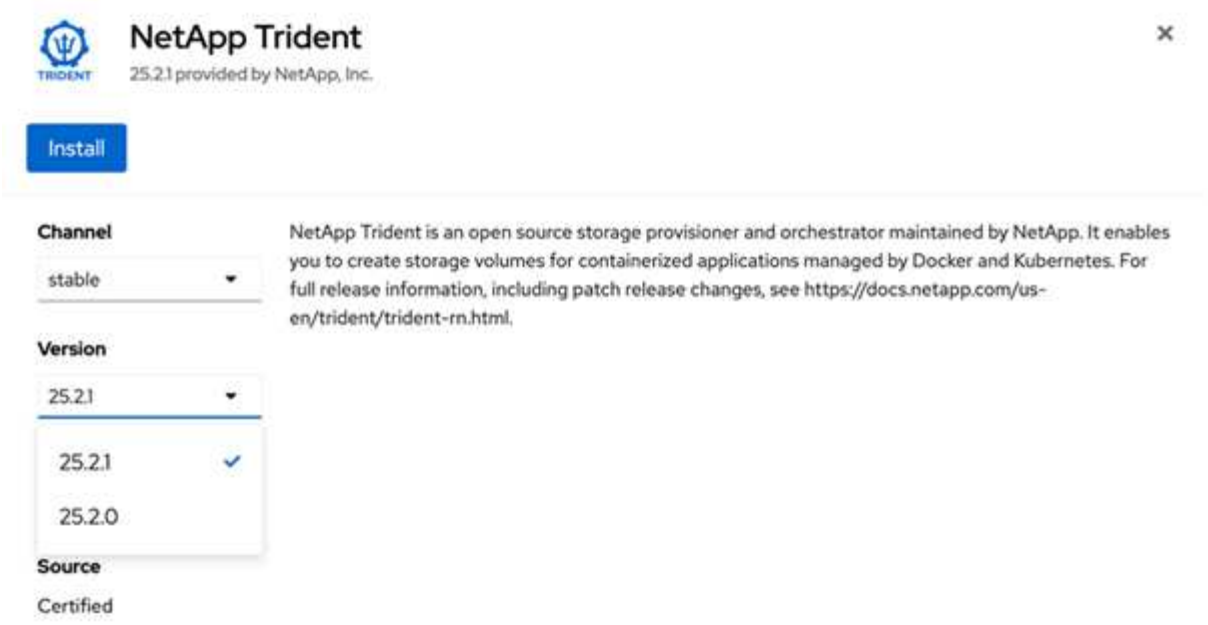
尋找並安裝 **Trident** 營運商

步驟

1. 瀏覽至 OpenShift OperatorHub 並搜尋 NetApp Trident 。

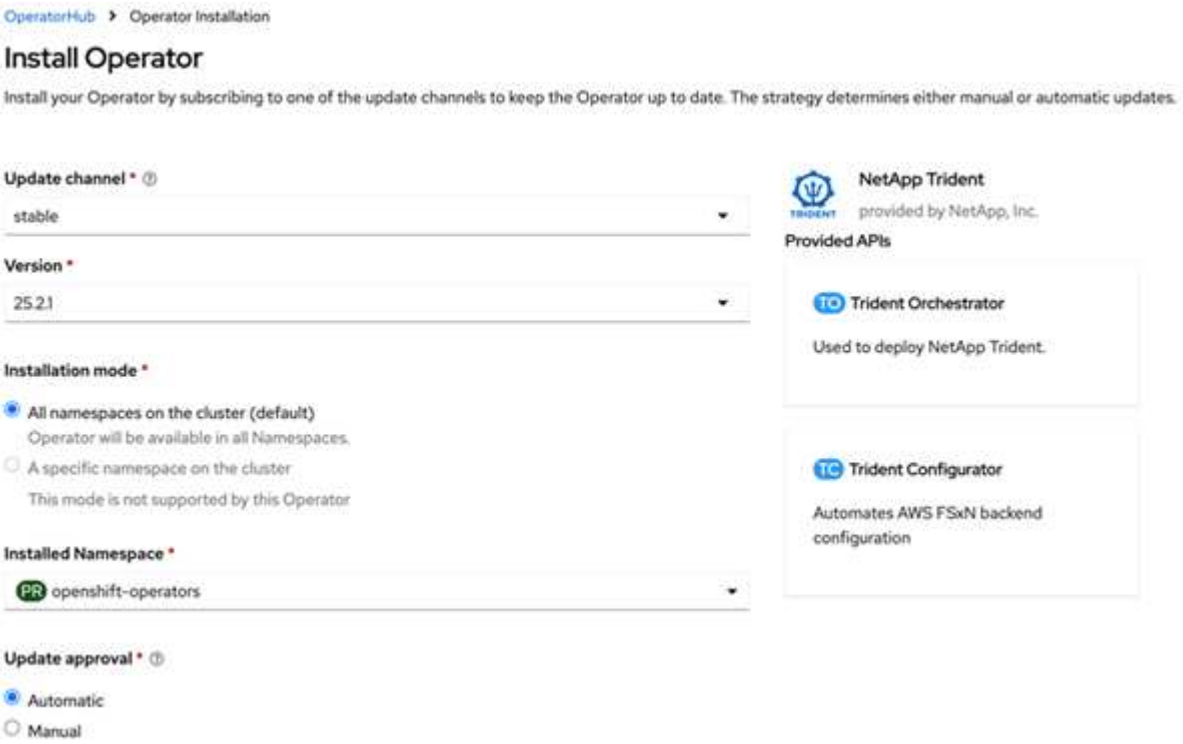


2. 點選* NetApp Trident* 開啟安裝設定。
3. 選擇所需的選項，然後按一下「安裝」以開啟操作員配置。

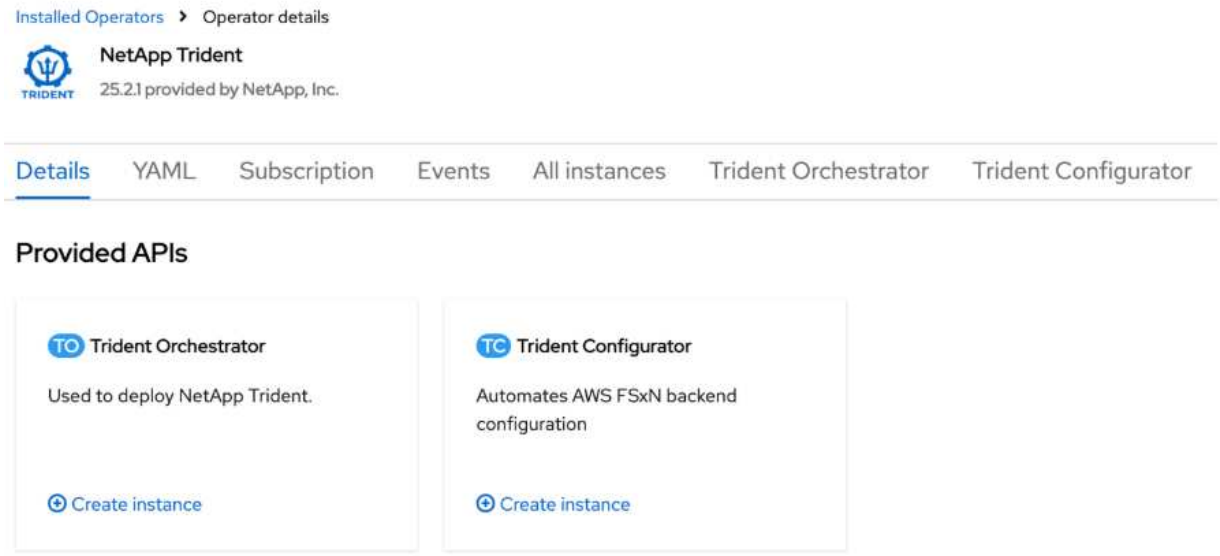


請確保選擇最新的 Operator 版本。

4. 保留所有參數，然後按一下“安裝”。



5. 按一下「查看操作員」以查看操作員的詳細資料。



6. 按一下「YAML 檢視」並將以下內容貼上到表單中：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
  debug: false
  nodePrep:
  - iscsi
  imageRegistry: ''
  k8sTimeout: 180s
  namespace: trident
  silenceAutosupport: false
```

UI 提供預設範例。您可以直接編輯，而無需複製完整組態。

選用：啟用並行

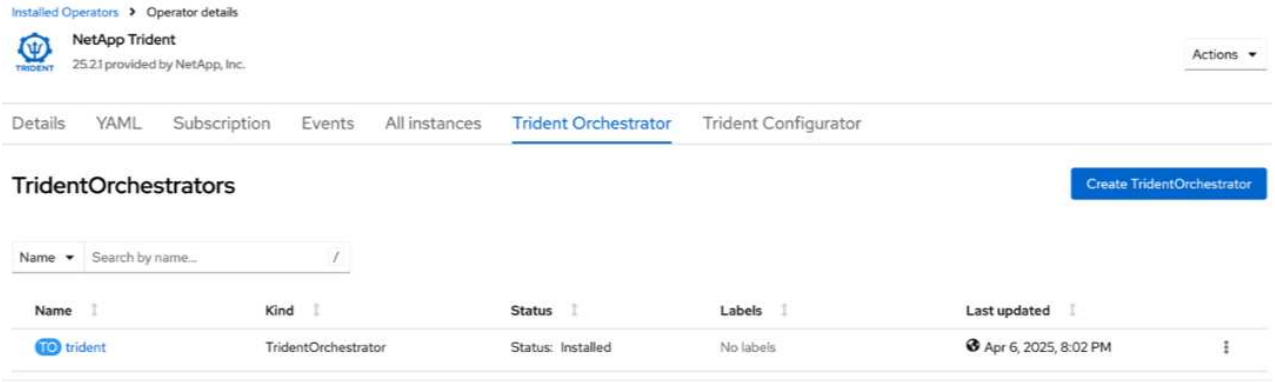
若要啟用並行性，請在規格中新增下列欄位：

```
enableConcurrency: true
```

- Red Hat Enterprise Linux CoreOS （RHCOS） 未啟用和設定 iSCSI。
- 您可以新增 `nodePrep`在所有 OpenShift 工作節點上配置和啟用 iSCSI 和 Multipath 服務的參數。

- 從 OpenShift 4.19 開始，此功能支援的最低Trident版本為 25.06.1。

1. 按一下「建立」；Trident Orchestrator 將會完全安裝。



卸載**Trident**操作員

步驟

1. 從已安裝操作員清單中選取 Trident 操作員。
2. 如果您要從運算子刪除所有的運算元執行個體，請選取此選項。

> 如果未選中 * 刪除此運算符的所有操作數實例 * 複選框，則不會卸載 Trident 。

3. 按一下*解除安裝*。

切換到 **OpenShift** 認證的 **Trident** 操作員

您可以從社群版 Operator、基於 Helm 的安裝或手動部署的 Operator 切換到 Red Hat OpenShift 認證的 Trident Operator。每種方法的過程都包括卸載現有 Operator，然後使用 OperatorHub 安裝認證的 Operator。

開始之前

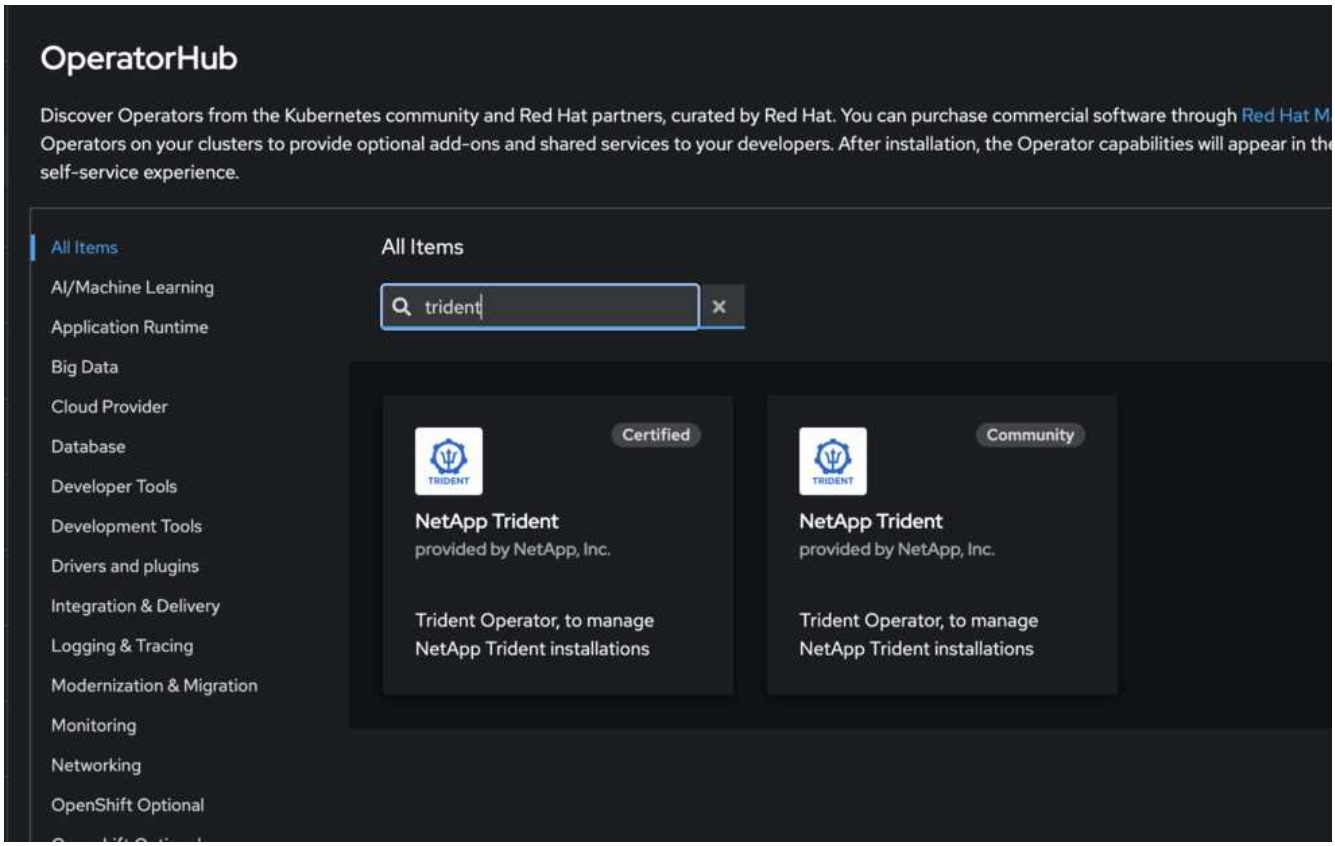
開始安裝之前"準備好您的環境以進行 Trident 安裝"，。

> 在卸載過程中請勿刪除 `TridentOrchestrator`自訂資源（CR）。 `TridentOrchestrator` CR 會保留您的後端和儲存類別配置，這在安裝認證操作員後是必要的。

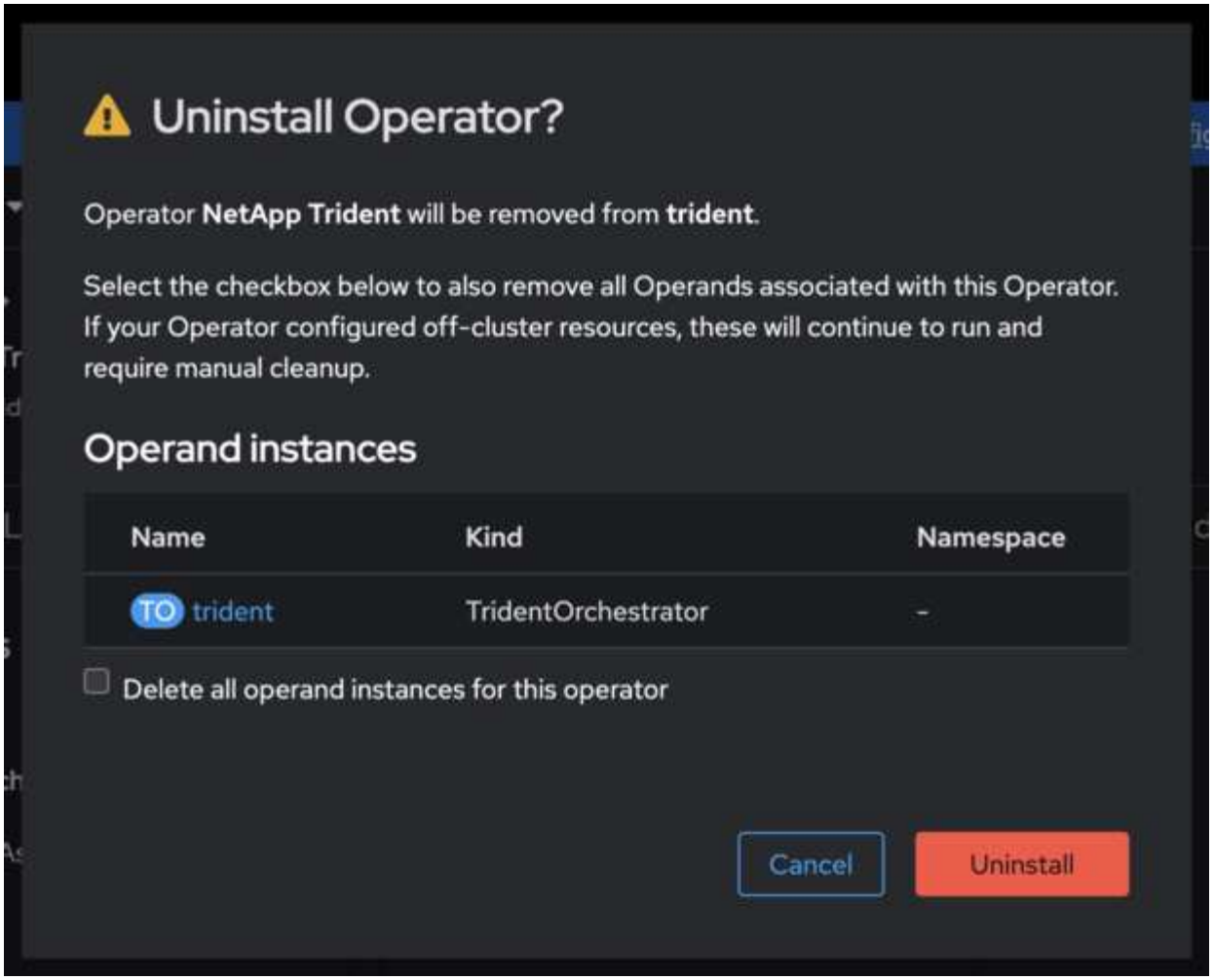
從社群營運商切換

步驟

1. 使用 OpenShift 主控台瀏覽至 OperatorHub 。



2. 尋找 NetApp Trident 社群業者。



請勿選取 * 刪除此運算子的所有運算元執行個體 * 。

3. 按一下*解除安裝*。
4. 卸載完成後，繼續執行 安裝 OpenShift 認證操作員。

從基於 **Helm** 的操作員安裝切換

步驟

1. 列出您的 Trident 安裝所使用的 Helm 版本：

```
helm ls -n trident
```

2. 解除安裝 Helm 版本：

```
helm uninstall <release-name> -n trident
```

3. 卸載完成後，繼續執行 安裝 OpenShift 認證操作員。

從手動部署的 **operator** 切換

如果您是透過 `bundle.yaml`安裝程式套件中的檔案手動部署操作員來安裝 Trident，請透過刪除相同的資訊清單來移除。

步驟

1. 使用套件組合資訊清單刪除操作員部署：

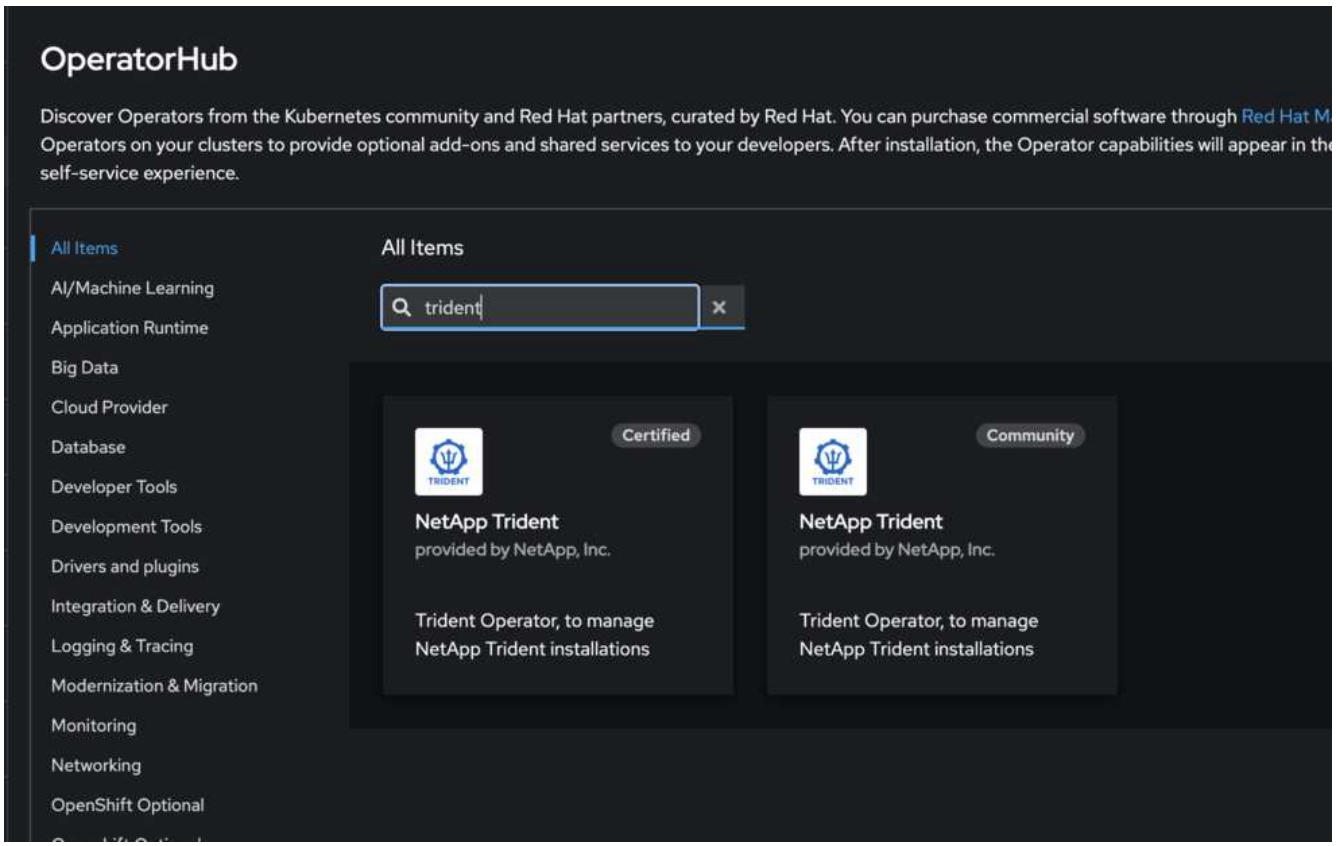
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. 卸載完成後，繼續執行 安裝 OpenShift 認證操作員。

安裝 **OpenShift** 認證操作員

步驟

1. 瀏覽至 Red Hat OperatorHub 。
2. 搜尋並選擇 NetApp Trident 運算子。



3. 依照畫面上的指示安裝操作人員。

驗證

- 檢查主控台的 OperatorHub ，確保已成功安裝新的認證營運者。

使用 Trident

準備 Trident 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。Red Hat Enterprise Linux CoreOS （ RHCOS ）的最新版本預設會安裝工具。

NFS工具

"安裝 NFS 工具"如果您正在使用： `ontap-nas` ， `ontap-nas-economy` ， `ontap-nas-flexgroup` ， 或者 `azure-netapp-files`。

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 ）。

NetApp 建議使用 ONTAP 9.12 或更新版本來處理 NVMe / TCP 。

SCSI over FC 工具

請參閱"設定 FC 擴大機、 FC-NVMe SAN 主機的方法"如需設定 FC 和 FC-NVMe SAN 主機的詳細資訊，。

"安裝 FC 工具"如果您使用 `ontap-san` sanType `fcp` （ SCSI over FC ）。

- 要考慮的要點 * ： * OpenShift 和 KubeVirt 環境支援 SCSI over FC 。* Docker 不支援 SCSI over FC 。* iSCSI 自我修復不適用於 FC 上的 SCSI 。

SMB工具

"準備配置SMB磁碟區" 如果您正在使用： `ontap-nas` 為 SMB 提供卷。

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
```

- 當使用搭配 iSCSI PV 執行 RHEL/Red Hat Enterprise Linux CoreOS （ RHCOS ）的工作節點時，請在 StorageClass 中指定 `discard` mountOption 以執行內嵌空間回收。請參閱 "Red Hat 說明文件"。
- 確保您已升級至最新版本 `multipath-tools` 。

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. 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 `/etc/multipath.conf`包含 `find_multipaths no`在 `defaults`中。

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
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 `/etc/multipath.conf`包含 `find_multipaths no`在 `defaults`中。

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「iscsiadmd」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-100.2610.0.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。

FC 磁碟區上的 SCSI

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。

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"

安裝 **FC** 工具

使用作業系統的命令來安裝FC工具。

- 當使用搭配 FC PV 執行 RHEL/Red Hat Enterprise Linux CoreOS （ RHCOS ）的工作節點時，請在 StorageClass 中指定 `discard` mountOption 以執行內嵌空間回收。請參閱 "Red Hat 說明文件"。

RHEL 8以上

1. 安裝下列系統套件：

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. 啟用多重路徑：

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

   確保 `/etc/multipath.conf`包含 `find_multipaths no`在 `defaults`中。

3. 確定 `multipathd`執行中：

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

Ubuntu

1. 安裝下列系統套件：

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

2. 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 `/etc/multipath.conf`包含 `find_multipaths no`在 `defaults`中。

3. 確定 `multipath-tools`已啟用並正在執行：

```
sudo systemctl status multipath-tools
```

準備配置SMB磁碟區

您可以使用以下方式設定 SMB 卷 `ontap-nas` 司機。

您必須在 SVM 上同時設定 NFS 和 SMB/CIFS 通訊協定，才能為 ONTAP 內部部署叢集建立 `ontap-nas-economy` 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。*	"

使用 COSI 佈建物件儲存

使用容器物件儲存介面（COSI）透過 Trident 代管的 Kubernetes 資源為 Kubernetes 工作負載資源配置 S3 儲存桶。

總覽

COSI 將物件儲存提供者連接至 Kubernetes 工作負載。Trident 提供下列資源：

- `TridentObjectStore`：將 Trident 連接到物件儲存系統及其認證資料。在 Trident 命名空間中建立此資源。
- `BucketClass`：定義物件存放區選擇和儲存桶刪除原則。
- `BucketClaim`：請求一個儲存桶。
- `BucketAccessClass`：定義如何資源配置儲存桶的存取。
- `BucketAccess`：要求一或多個儲存桶的認證資料。

Trident 透過下列物件儲存驅動程式支援 COSI：

- StorageGRID：已正式發布。
- ONTAP S3：技術預覽。

Trident 支援 S3 協定。在初始版本中，僅支援路徑式 S3 定址。

需求與限制

- 在 `TridentOrchestrator`自訂資源 （`TORC`） 中同時啟用 `enableConcurrency`和 `enableCOSI`。COSI 預設不會啟用。
- 在啟用控制器並行之前，請先驗證所有已設定的 Trident 後端是否都使用支援的驅動程式。不受支援的後端驅動程式會導致 Trident 無法啟動；請參閱 "在啟用控制器擴充性之前"。
- 請勿將 Trident COSI 與社群 COSI 控制器同時執行。
- 初始版本使用 `objectstorage.k8s.io/v1alpha2` COSI API。
- StorageGRID `TridentObjectStore` 資源使用長期有效的租戶根目錄認證資料。每個 `TridentObjectStore` 連接到一個 StorageGRID 租戶；多個資源可以連接到同一個租戶。將認證資料儲存在 Trident 命名空間的 Kubernetes Secret 中，並根據您的安全性要求加以保護。
- StorageGRID `BucketAccess` 支援最多八個儲存桶。

如需並行控制器作業的一般資訊，請參閱 "控制器擴充性"。

啟用 COSI

在 `TridentOrchestrator` 規範中同時設定這兩個選項：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
```

```
  enableCOSI: true
```

套用 TORC 或使用一般的營運者工作流程更新現有資源。如需設定 TORC 的詳細資訊，請參閱 "自訂Trident操作員安裝"。

設定 StorageGRID 物件存放區

在 Trident 命名空間中，使用 StorageGRID 租戶認證資料建立 Kubernetes Secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: tenant3
  namespace: trident
type: Opaque
stringData:
  username: <tenant-root-username>
  password: <tenant-root-password>
```

建立參照 Secret 的 `TridentObjectStore`。將範例端點和帳戶 ID 替換為您 StorageGRID 環境的值：

```
apiVersion: trident.netapp.io/v1
kind: TridentObjectStore
metadata:
  name: trident-tenant3
  namespace: trident
spec:
  accountId: "<account-id>"
  credentials:
    secretName: tenant3
  labels:
    tenantInstance: three
  managementEndpoint: https://<storagegrid-management-endpoint>
  objectStoreDriver: storage-grid
  s3Endpoint: https://<storagegrid-s3-endpoint>
```

套用這兩個資源：

```
kubectl apply -f tenant-secret.yaml
kubectl apply -f trident-object-store.yaml
```

設定 ONTAP S3 物件存放區

透過 COSI 提供的 ONTAP S3 支援在 Trident 26.10 中仍處於技術預覽階段，暫不支援在生產環境中使用。組態欄位和行為在正式可用度前可能會有所更改。

在 Trident 命名空間中使用 ONTAP S3 使用者認證資料建立 Kubernetes Secret，然後建立參照該 Secret 的 `TridentObjectStore`。將預留位置替換為您 ONTAP S3 環境的值：

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-s3-credentials
  namespace: trident
type: Opaque
stringData:
  username: "<user>"
  password: "<password>"
---
apiVersion: trident.netapp.io/v1
kind: TridentObjectStore
metadata:
  name: ontap-s3
  namespace: trident
spec:
  objectStoreDriver: ontap-s3
  objectStoreName: ontap-object-store
  managementEndpoint: "<management-endpoint>"
  svmName: "<svm-name>"
  storagePrefix: trident
  credentials:
    secretName: ontap-s3-credentials
```

此 `storagePrefix` 是選用的，省略時預設為 `trident`。套用 Secret 和 `TridentObjectStore`：

```
kubectl apply -f ontap-s3-object-store.yaml
```

建立 BucketClass 並申請儲存桶

`BucketClass`
可以使用標籤選擇器、物件存放區類型或兩者來選擇物件存放區。如果沒有選擇參數， `Trident` 可以從所有已設定的物件存放區中進行選擇。

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketClass
metadata:
  name: sg-delete
spec:
  driverName: cosi.trident.netapp.io
  deletionPolicy: Delete
  parameters:
    selector: tenantInstance=three
    objectStoreType: storage-grid
```

建立一個使用該類別的 `BucketClaim`：

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketClaim
metadata:
  name: claim-a1
  namespace: default
spec:
  bucketClassName: sg-delete
  protocols:
    - S3
```

當 Trident 處理要求時，它會選擇一個相符的物件存放區。如果有多個存放區相符， Trident 會隨機選擇一個，將儲存體分配到相符的存放區中。

授予工作負載對儲存桶的存取

建立 `BucketAccessClass`以定義存取資源配置：

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketAccessClass
metadata:
  name: sg-access
spec:
  authenticationType: Key
  driverName: cosi.trident.netapp.io
  multiBucketAccess: MultipleBuckets
```

建立參照 bucket claim 的 `BucketAccess`：

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketAccess
```

```
metadata:
  name: sg-claim-a1
  namespace: default
spec:
  bucketAccessClassName: sg-access
  bucketClaims:
    - accessMode: ReadWrite
      accessSecretName: sg-claim-a1
      bucketClaimName: claim-a1
  protocol: S3
```

Trident 會建立一個包含儲存桶存取資訊的 Kubernetes Secret。工作負載可以使用標準的 Kubernetes Secret 機制來使用此 Secret。產生的 Secret 包含 S3 連線詳細資訊，例如儲存桶識別碼、端點、區域和存取金鑰。請勿將產生的 Secret 暴露給未經授權的使用者或工作負載。

產生的 Secret 包含 `COSI_S3_ACCESS_KEY_ID`、`COSI_S3_ACCESS_SECRET_KEY`、`COSI_S3_ADDRESSING_STYLE`、`COSI_S3_BUCKET_ID`、`COSI_S3_ENDPOINT`和 `COSI_S3_REGION` 等金鑰。

設定及管理後端

設定 Trident 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 Google Cloud NetApp Volumes 後端"
- "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

使用 Azure NetApp Files 作為 Trident 的後端。此後端支援 NFS 和 SMB 磁碟區。Trident 支援 Azure Kubernetes Service（AKS）叢集的託管身分識別和工作負載身分識別。

支援的 **Azure** 雲端環境

Trident 支援多種 Azure 雲端環境中的 Azure NetApp Files 後端。

支援的 Azure 雲端包括：

- Azure 商業版
- Azure Government（Azure Government / MAG）
- Azure 中國

部署 Trident 或設定 Azure NetApp Files 後端時、請確保 Azure Resource Manager 和驗證端點與您的 Azure 雲端環境相符。

查看 **Azure NetApp Files** 驅動程式支援

Trident 提供以下 Azure NetApp Files 儲存驅動程式。

支援的存取模式包括 *ReadWriteOnce* （RWO）、*ReadOnlyMany* （ROX）、*ReadWriteMany* （RWX）和 *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`

審查注意事項

- Azure NetApp Files 不支援小於 50 GiB 的磁碟區。當請求較小的磁碟區時，Trident 會建立一個 50 GiB 的磁碟區。
- Trident 僅支援掛載至 Windows 節點上執行的 Pod 的 SMB 磁碟區。
- Azure NetApp Files 在非商業 Azure 雲端環境中部署需要使用特定於雲端的 Azure Resource Manager 和驗證終端點。請確保 Trident 和任何後端組態都使用適合您 Azure 雲端環境的終端點。

使用 **AKS** 的託管身分

Trident 支援 "託管身分識別" AKS 叢集。

如果您使用 `tridentctl`建立或管理 Azure NetApp Files 後端，請確保已將其配置為正確的 Azure 雲端環境。

若要使用託管身分，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
- Trident 已安裝並將 `cloudProvider` 設定為 `"Azure"`

Trident運算子

編輯 `tridentorchestrator_cr.yaml` 並設定 `cloudProvider` 為 `"Azure"`。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
```

```
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

掌舵

以下範例會安裝 Trident 並設定 `cloudProvider`，使用環境變數 `$CP`：

```
helm install trident trident-operator-100.2610.0.tgz --create-namespace
--namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

以下範例安裝 Trident 並將 `cloud-provider` 標誌設為 `Azure`：

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

使用 **AKS** 的工作負載身分識別

工作負載身分可讓 Kubernetes Pod 透過以工作負載身分進行驗證來存取 Azure 資源。

如果您使用 `tridentctl`建立或管理 Azure NetApp Files 後端，請確保已將其配置為正確的 Azure 雲端環境。

若要使用工作負載身分，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
- Trident 已安裝， `cloudProvider` 設定為 `"Azure"`， `cloudIdentity` 設定為工作負載識別值

Trident運算子

編輯 `tridentorchestrator_cr.yaml`並設定 `cloudProvider`為 `"Azure"`。設定 `cloudIdentity`為 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`。

```
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-
xxxx-xxxx-xxxxxxxxxxx' # Edit
```

掌舵

使用下列環境變數設定 **cloud-provider** （ **CP** ） 和 **cloud-identity** （ **CI** ） 標誌的值：

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

以下範例安裝 Trident 並使用 `cloudProvider` 設定 `$CP`，並使用 `cloudIdentity` 設定 `$CI`：

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>tridentctl</code>

請使用以下環境變數設定 **cloud provider** 和 **cloud identity** 標誌的值：

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

以下範例安裝 Trident 並將 `cloud-provider`設定為 `$CP`、將 `cloud-identity`設定為 `$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。

支援的 **Azure** 雲端環境

Trident 支援多種 Azure 雲端環境中的 Azure NetApp Files 後端。

支援的 Azure 雲端包括：

- Azure 商業版
- Azure Government（Azure Government / MAG）
- Azure 中國

在準備環境時，請確保在對應的 Azure 雲端環境中建立 Azure 訂閱、身分配置和 Azure NetApp Files 資源。

如果您是首次使用 Azure NetApp Files 或在新的位置使用，則需要進行一些初始設定來設定 Azure NetApp Files 並建立 NFS 磁碟區。請參閱 "Azure：設定Azure NetApp Files 功能以建立NFS Volume"。

要設定和使用 "Azure NetApp Files" 後端、您需要以下元件：

- `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。
- 在非商業 Azure 雲端環境中部署 Azure NetApp Files 需要使用特定於雲端的 Azure Resource Manager 和驗證終端點。請確保 Trident 和任何後端組態都使用適合您 Azure 雲端環境的終端點。

- 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"，。

▼ 自訂參與者角色

```
{
  "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/re
ad",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/wr
ite",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/de
lete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTargets
/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/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": []
      }
    ]
  }
}
```

- 包含至少一個 "委派的子網路"的 Azure `location`。 `location`參數必須在後端組態檔案的最上層指定。在虛擬集區中指定的位置值將被忽略。
- 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-xxxxxxxxxxx`。

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 使用您的後端設定（子網路、虛擬網路、服務等級和位置）在要求的位置可用的容量池上建立 Azure NetApp Files Volume，並符合要求的服務等級和子網路。

Azure NetApp Files 後端提供這些組態選項。

參數	說明	預設
「分度」	後端配置版本。	永遠為1
「torageDriverName」	儲存驅動程式名稱	「Azure - NetApp-Files」
「後端名稱」	儲存後端的自訂名稱	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。	

參數	說明	預設
「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。	
`cloudConfiguration.cloudName`	識別後端的 Azure 雲端環境。使用 Azure Government 時設定為 `AzureGovernment`。	
"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	"
《Subnet》	委派給「icrosoft.Netapp/volumes`」的子網路名稱	"
《網路功能》	磁碟區的 VNet 功能集，可以是 `Basic`或 `Standard`。網路功能並非在所有區域都可用，可能需要在訂閱中啟用。在未啟用功能時指定 `networkFeatures`會導致磁碟區佈建失敗。	"
「nfsMountOptions」	對 NFS 掛載選項進行精細控制。SMB 卷將忽略此設定。若要使用 NFS 版本 4.1 掛載卷，請在以逗號分隔的掛載選項清單中新增 `nfsvers=4` 以選擇 NFS v4.1。在儲存類別定義中設定的掛載選項會覆蓋後端配置中設定的掛載選項。	"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`

參數	說明	預設
`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。	
`qosType`	表示 QoS 類型：自動或手動。	汽車
`maxThroughput`	設定允許的最大吞吐量，單位為 MiB/秒。僅支援手動 QoS 容量池。	`4 MiB/sec`

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"。

考慮 **Azure** 雲端環境（**26.06**）

從 26.06 版本開始， Trident 支援在多個 Azure 雲端環境中建立和管理 Azure NetApp Files 後端。

支援的 Azure 雲端包括：

- Azure 商業版
- Azure Government（Azure Government / MAG）
- Azure 中國

部署 Trident 或建立 Azure NetApp Files 後端時、請確保 Azure Resource Manager 和驗證端點與您的 Azure 雲端環境相符。如果端點不相符、 `tridentctl`就無法驗證、後端建立也會失敗。

對於 Azure Government 後端，請在後端組態中設定 `cloudConfiguration.cloudName`：

```
cloudConfiguration:
  cloudName: AzureGovernment
```

必要的權限與資源

如果在建立 PVC 時收到「未找到容量池」錯誤，則可能是您的應用程式註冊缺少所需的權限和資源（子網路、虛擬網路、容量池）。如果啟用了偵錯模式，Trident 會記錄建立後端時發現的 Azure 資源。請確認是否使用了適當的角色。

`resourceGroups`、 `netappAccounts`、 `capacityPools`、 `virtualNetwork` 和 `subnet`
的值可以使用短名稱或完全限定名稱來指定。大多數情況下建議使用完全限定名稱，因為短名稱可能會符合多個同名資源。

如果虛擬網路與Azure NetApp Files(ANF) 儲存帳戶位於不同的資源群組中，則在設定後端資源群組清單時，請為虛擬網路指定資源群組。

`resourceGroups`、 `netappAccounts` 和 `capacityPools`
值是過濾器，用於將發現的資源集限制為此儲存後端可用的資源，並且可以以任意組合指定。完全限

定名稱遵循以下格式：

類型	格式
資源群組	<資源群組>
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

您可以透過在設定檔特定部分中指定以下選項來控制預設磁碟區配置。如需詳細資訊，請參閱 [組態範例]。

參數	說明	預設
「匯出規則」	匯出新磁碟區的規則。 `exportRule` 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。	「0.00.0.0/0」
「napshotDir	存取「.snapshot」目錄	`true`, `false`（明確設定）。
《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

以下範例展示了基本配置，其中大多數參數都保留預設值。這是定義後端最簡單的方法。

▼ 最小組態

這是絕對最小的後端組態。使用此配置， Trident 會發現您所有的 NetApp 帳戶、容量集區和子網路（已委派給位於已設定位置的 Azure NetApp Files ），並隨機將新磁碟區放置在其中一個集區和子網路上。由於 `nasType`被省略， `nfs`預設值會套用，後端將會佈建 NFS 磁碟區。

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

此後端組態已不再如此 `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:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    - resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/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 `eastus`位置中的 `Ultra`容量集區。Trident 會自動探索該位置中委派給 Azure NetApp Files 的所有子網路，並隨機將新磁碟區放置在其中一個子網路上。

```
---
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
```

▼ 具有手動 **QoS** 容量池的後端範例

此後端配置將磁碟區放置在 Azure 中 `eastus`具有手動 QoS 容量池的位置。

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
```

```
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

▼ 進階組態

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: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/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"
```

▼ 虛擬集區組態

此後端配置在單一檔案中定義了多個儲存池。當您有多個容量池支援不同的服務級別，並且希望在 Kubernetes 中建立代表這些容量池的儲存類別時，此配置非常有用。虛擬池標籤用於根據 `performance`來區分這些池。

```
---
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:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

▼ 支援的拓撲組態

Trident 能夠根據區域和可用區為工作負載配置磁碟區。 `supportedTopologies`此後端配置中的程式碼區塊用於為每個後端提供區域和可用區清單。此處指定的區域和可用區值必須與每個 Kubernetes 叢集節點標籤中的區域和可用區值相符。這些區域和可用區代表儲存類別中可以提供的允許值清單。對於包含後端提供的區域和可用區子集的儲存類， Trident 會在指定的區域和可用區中建立磁碟區。如需更多資訊，請參閱"使用「csi 拓撲」"。

```
---
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 `StorageClass`指定用於託管磁碟區的虛擬池。此磁碟區將具有所選儲存池中定義的設定項。

```
---
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 憑證。

▼ 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` 篩選支援 SMB 磁碟區的儲存池。
`nasType: nfs` 或 `nasType: null` 篩選 NFS 儲存池。

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>
```

如果您使用的是非商業版 Azure 雲端，請確保 `tridentctl`已將其設定為使用 Azure Resource Manager 和 Azure 雲端環境的驗證端點。如果後端建立失敗，請檢查後端組態並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

配置 **Google Cloud NetApp Volumes**

您可以將 Google Cloud NetApp Volumes 設定為 Trident 的後端，以便為 Kubernetes 工作負載配置儲存設備。

總覽

Trident 支援 Google Cloud NetApp Volumes，可用於 NAS（NFS 和 SMB）和區塊（iSCSI）工作負載。

- NAS 工作負載使用 `google-cloud-netapp-volumes` 後端
- 區塊 （iSCSI） 工作負載使用 `google-cloud-netapp-volumes-san` 後端

NAS Volume 提供檔案型儲存設備，並使用 NFS 或 SMB 傳輸協定進行存取。這些 Volume 支援跨多個 Pod 或節點的共享存取。

區塊磁碟區提供原始區塊儲存設備，並以連接至 Kubernetes 節點的 iSCSI 裝置形式進行存取。當應用程式需要區塊層級存取時，就會使用這些磁碟區。

這適用於以下環境：

- Trident 26.02 及更新版本
- Google Kubernetes Engine (GKE) 或在 GCP 上執行的 Red Hat OpenShift 叢集
- OpenShift 工作負載身分聯盟（WIF）需要 Trident 26.10 或更新版本

- Google Cloud NetApp Volumes 儲存池

若要設定區塊（iSCSI）儲存設備、請參閱 "設定區塊儲存設備（iSCSI）"。

準備進行設定

雲端身分可讓 Kubernetes 工作負載透過以工作負載身分進行驗證來存取 Google Cloud 資源，而非使用靜態認證。

在設定工作負載身分之前，請使用獲授權建立所需身分資源的帳戶。對於 GKE，這需要 Google Cloud 權限來建立和設定 Google Cloud 服務帳戶。對於 OpenShift，這需要建立或選擇叢集的 WIF 組態，以及設定 Google Cloud 工作負載身分資源池和供應商的權限。

使用 **GKE** 工作負載身分

若要將 GKE 工作負載身分與 Google Cloud NetApp Volumes 搭配使用，您必須具備：

- 使用 Google Kubernetes Engine （ GKE ）部署的 Kubernetes 叢集
- GKE 叢集上已啟用工作負載身分識別，節點集區上已啟用中繼資料伺服器
- 具有 Google Cloud NetApp Volumes 管理員角色的 Google Cloud 服務帳號(`roles/netapp.admin`）或等效的自訂角色
- Trident 已安裝，雲端提供者設定為 `GCP` 並已設定雲端身分註解

Trident運算子

若要使用 Trident 運算子安裝 Trident，請編輯 `tridentorchestrator_cr.yaml`：

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

掌舵

使用 Helm 安裝 Trident 時、請設定雲端供應商和雲端身分：

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

試用

透過指定雲端供應商和雲端身分來安裝 Trident：

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
  -n trident
```

在 **OpenShift** 上使用工作負載身分聯合

從 Trident 26.10 開始，您可以使用 Google Cloud Workload Identity Federation （WIF） 在 GCP 上執行的 OpenShift 叢集上驗證 Google Cloud NetApp Volumes 驅動程式。這適用於自管理的 OpenShift Container Platform （OCP，包括 IPI 部署） 和 OpenShift Dedicated （OSD），並同時支援 NAS (`google-cloud-netapp-volumes`） 和 SAN (`google-cloud-netapp-volumes-san`） 驅動程式。

WIF 將投射的 Kubernetes 服務帳戶 Token 交換為短期的 Google Cloud 認證資料。Trident 使用這些認證資料來模擬 Google Cloud 服務帳戶；該服務帳戶的 IAM 角色會授權其 Google Cloud NetApp Volumes API 呼叫。這可避免在後端組態中儲存長期的服務帳戶金鑰。

> ⓘ 在 OpenShift Dedicated 上，OIDC 供應商是可以與其他 OpenShift 元件共享的。如果供應商的 `allowedAudiences`清單為空，Google 會接受供應商的規範資源名稱（有或沒有 `https://`前綴），因此 Trident 對象無需更改供應商。請勿僅為 Trident 設定明確的對象清單。如果供應商已包含明確的對象，請在保留所有現有值的情況下附加 Trident 對象。替換清單可能會阻止其他 OpenShift 元件交換權杖。

準備 **OpenShift** 叢集

對於 OSD，請在建立叢集時建立或選擇一個 GCP WIF 組態。例如，OpenShift 叢集管理器 CLI 可以建立一個：

```
ocm gcp create wif-config --name <wif-name> --project <gcp-project-id>
```

選擇工作負載身分聯合（Workload Identity Federation）作為叢集驗證類型，並將 WIF 組態與叢集關聯。對於自我管理的 OCP，設定一個信任叢集服務帳戶權杖發行者的 Google Cloud OIDC 供應商。OSD 和自我管理 OCP 的叢集資源配置和權杖投影步驟有所不同；請遵循適用的 Red Hat 指南，並將投影的權杖提供給 Trident 控制器 `/var/run/secrets/wif-token/token`。

OSD 的 WIF 驗證在建立叢集時設定。Trident 的 WIF 對象是分隔的安裝設定；您無需重新建立叢集即可為 Trident 啟用 WIF。

在安裝或設定 Trident 之前，請確認用於安裝的 OpenShift 身分擁有所選安裝方法所需的叢集權限。安裝後，請確認控制器和節點 Pod 運作狀況良好：

```
oc whoami
oc auth can-i create namespaces
oc auth can-i create csidrivers.storage.k8s.io
```

```
oc auth can-i create customresourcedefinitions.apiextensions.k8s.io
oc auth can-i create securitycontextconstraints.security.openshift.io

oc -n trident get pods
oc -n trident rollout status deploy/trident-controller --timeout=300s
oc -n trident rollout status ds/trident-node-linux --timeout=300s
```

使用 **WIF** 對象安裝 **Trident**

Trident 僅在安裝期間設定對象時，才會將權杖投射到 WIF。請在 Google Cloud 供應商和後端的 `wipCredentialConfig.audience`中使用相同的對象：

```
//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>
```

將 `cloudProvider`設為 `GCP`，並使用您選擇的安裝方法來設定 `gcpWIFAudience`：

試用

```
tridentctl install \
  --cloud-provider GCP \
  --gcp-wif-audience "<audience>" \
  --namespace trident
```

Trident運算子

在 `TridentOrchestrator`規格中設定這些欄位：

```
spec:
  cloudProvider: GCP
  gcpWIFAudience: "<audience>"
```

掌舵

安裝圖表時請設定以下值：

```
helm install trident netapp-trident/trident-operator \
  --set cloudProvider=GCP \
  --set-string gcpWIFAudience="<audience>"
```

請勿同時為 OpenShift WIF 設定 GKE `cloudIdentity` / `--cloud-identity` 選項。安裝後請勿修補 Trident 控制器部署；營運者會還原手冊變更。後端可在 `/var/run/secrets/wif-token/token`取得投影權杖。

`trident-main` Container 不包含 shell，因此請檢查控制器部署的 Volume 組態，而不是使用 `oc exec`來檢查檔案：

```
oc -n trident get deploy trident-controller -o yaml
```

設定 **Google Cloud** 信任與權限

為叢集設定 Google Cloud Workload Identity Pool 和 OIDC 供應商。供應商發行者必須與權杖的 `iss`宣告相符，且其屬性對應必須將 `assertion.sub`對應至 `google.subject`。供應商的允許對象必須與投射權杖的 `aud`宣告相符。

僅在使用 OIDC 探索時，才需要公開 `.well-known/openid-configuration`和 JWKS 端點。Google 也支援使用 `jwksJson`或 `--jwk-json-path`直接上傳 JWKS。

在 Trident 將要模擬的 Google Cloud 服務帳戶上，授予 WIF 主體 `roles/iam.workloadIdentityUser` 角色。對於權杖主旨 `system:serviceaccount:trident:trident-controller`，主體具有以下形式：

```
principal://iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/subject/system:serviceaccount:trident:trident-controller
```

在包含 Google Cloud NetApp Volumes 資源的專案上，授予被模擬的服務帳戶 `roles/netapp.admin`，或具有所需權限的等效自訂角色。這些是分隔的授權：第一個允許模擬，第二個授權儲存設備 API 操作。

如果權杖對象和供應商組態不相符，Google Security Token Service 可能會因 `invalid_grant`對象錯誤而拒絕交換。請使用以下命令檢查供應商組態：

```
gcloud iam workload-identity-pools providers describe <provider-id> \
  --project=<wif-pool-project-id> \
  --location=global \
  --workload-identity-pool=<pool-id> \
  --format="yaml(oidc.allowedAudiences,oidc.issuerUri,attributeMapping)"
```

診斷權杖組態時，請將權杖的 `iss`、`aud`和 `sub`宣告與供應商和 IAM 組態進行比較。請勿共享或記錄完整的投影權杖。

準備 **Google Cloud NetApp Volumes** 儲存設備

在後端使用的區域中建立或識別儲存資源池。NAS 後端需要 FILE 資源池；SAN 後端使用 "設定區塊儲存設備（iSCSI）" 中所述的 Flex Unified 資源池。確保資源池的 VPC 和專用服務存取（PSA）組態適用於叢集網路。

例如，在為資源池選擇 PSA 範圍之前，列出網路現有的全域 VPC 對等關係範圍：

```
gcloud compute addresses list \
  --project <gcnv-project-id> \
```

```
  --global \
  --filter="purpose=VPC_PEERING AND network:<network-name>" \
  --format="table(name,address,prefixLength,status,network,purpose)"
```

在 `location`中使用資源池的區域，並在需要時，在 `storagePools`中使用其完整資源路徑明確選擇資源池。區域和資源池位置不符、網路不正確或資源池無法使用，即使驗證成功，也可能無法進行探索或資源配置。

設定 Trident 後端

在供應商組態和 `wipCredentialConfig.audience`中使用預期的權杖對象。 `projectNumber` 欄位可識別 Google Cloud NetApp Volumes 專案；對象使用包含工作負載身分資源池的專案編號。請將此範例中的預留位置替換為您環境的值：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas-wif
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
    serviceLevel: Premium
  wipCredentialConfig:
    type: external_account
    audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
    subject_token_type: urn:ietf:params:oauth:token-type:jwt
    token_url: https://sts.googleapis.com/v1/token
    credential_source:
      file: /var/run/secrets/wif-token/token
    service_account_impersonation_url:
https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<service-account-email>:generateAccessToken
```

對於 SAN 後端，請使用 `google-cloud-netapp-volumes-san` 作為 `storageDriverName`，以及適用的 Flex Unified 資源池設定。除此之外，兩個驅動程式的 WIF 認證資料組態是相同的。

套用後端並檢查其觀測狀態：

```
oc apply -f <backend-file>.yaml
oc -n trident get tridentbackendconfig <backend-name> -o yaml
```

在測試配置之前，請確認後端為 `Bound`且其上次操作已成功。檢查即時的 `spec`和 `status`；先前套用的組態註解可能無法反映後續的變更。使用 WIF 時，請勿包含靜態 `apiKey`認證資料。

驗證端對端配置

建立一個針對後端的 StorageClass，然後建立一個小型測試 PVC。例如，NAS StorageClass 可以選擇一個後端標籤：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-wif-smoke
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  selector: "auth=wif"
allowVolumeExpansion: true
```

將符合的 `auth: wif`標籤新增至 NAS 後端的儲存設備條目，使用 `gcnv-wif-smoke`建立 PVC，並驗證其是否到達 `Bound`。對於 SAN 冒煙測試，請使用 `google-cloud-netapp-volumes-san`、SAN 驅動程式設定和適當的 Flex Unified 資源池。完成後，請刪除測試 PVC 和 StorageClass。

排除常見故障

症狀	需要檢查什麼
`invalid_grant` 或對象不匹配	將投射的 token 的 `aud`與供應商允許的對象和 `wipCredentialConfig.audience`進行比較。
`iam.serviceAccounts.getAccessToken` 權限遭拒	檢查 Trident 控制器 `sub`的確切 `WIF` 主體在模擬的服務帳戶上是否具有 `roles/iam.workloadIdentityUser`。
`netapp.storagePools.list` 權限遭拒	檢查被模擬的服務帳戶是否在正確的專案上擁有 Google Cloud NetApp Volumes 權限，例如 `roles/netapp.admin`。
未找到資源池或資源池未找到	檢查後端 `location`、資源池區域和類型、網路、PSA 組態，以及任何明確設定的儲存資源池資源路徑。
後端仍處於未綁定狀態	檢查 Trident 控制器日誌和 TridentBackendConfig 狀態，尋找底層驗證、權限或資源池發現錯誤。

若要檢查控制器記錄以尋找後端協調錯誤：

```
oc -n trident logs deploy/trident-controller -c trident-main --tail=400
```

對於 Google Cloud NetApp Volumes UNIFIED 儲存池，Trident 在執行磁碟區操作時會套用 UNIFIED 專屬的命名與驗證規則。

在尋找磁碟區時，Trident 可以評估多個相容的磁碟區名稱變體（例如連字號和底線格式），以提高匯入和探索的可靠性。

驅動程式詳細資料

Trident 提供 `google-cloud-netapp-volumes` 驅動程式，以從 Google Cloud NetApp Volumes 配置 NAS 儲存空間。

此驅動程式支援以下存取模式：

- 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 中小企業	檔案系統	Rwo 、 ROX 、 rwx 、 RWOP	`nfs`、 `smb`

設定 Trident NAS 後端

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
```

配置 NAS Volume

NAS 磁碟區透過 `google-cloud-netapp-volumes` 後端配置，並支援 NFS 和 SMB 協定。

NFS Volume 的 StorageClass

若要佈建 NFS 磁碟區，請將 `nasType` 設為 `nfs`。

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

適用於 SMB 磁碟區的 StorageClass

若要配置 SMB Volume ，請將 `nasType` 設為 `smb` 並提供認證資料。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
allowVolumeExpansion: true
```

PersistentVolumeClaim 範例（RWX）

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwx
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

PersistentVolumeClaim 範例（RWO）

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

NAS 磁碟區使用 `volumeMode: Filesystem`。

設定 **Google Cloud NetApp Volumes** 以用於 **SAN** 工作負載

使用 `google-cloud-netapp-volumes-san` 驅動程式從 Google Cloud NetApp Volumes Flex Unified 儲存設備池資源配置 iSCSI 區塊 Volume。

SAN 驅動程式僅支援區塊工作負載。若要資源配置 NAS Volume，請使用 `google-cloud-netapp-volumes` 驅動程式。

總覽

此驅動程式從 Flex Unified 儲存設備池資源配置 Volume，並將其做為 iSCSI 區塊裝置呈現給 Kubernetes 節點。

此驅動程式支援以下環境：

- Trident 26.02 及更新版本
- Google Kubernetes Engine (GKE) 或在 GCP 上執行的 Red Hat OpenShift 叢集
- Google Cloud NetApp Volumes Flex 統一儲存池
- 基於 iSCSI 的工作負載

從 Trident 26.10 開始，SAN 驅動程式支援在 GCP 上執行的自我管理 OpenShift Container Platform（OCP，包括 IPI 部署）和 OpenShift Dedicated（OSD）叢集上的 Google Cloud Workload Identity Federation（WIF）。如需 WIF 先決條件與後端組態，請參閱 "在 OpenShift 上使用工作負載身分聯合"。請使用此頁面上的 SAN 驅動程式名稱與 Flex Unified 儲存設備設定。

設定 **SAN** 的 **WIF**

WIF 將投射的 Kubernetes 服務帳戶權杖交換為短期的 Google Cloud 認證資料。Trident 在呼叫 Google Cloud NetApp Volumes API 時，會使用這些認證資料來模擬 Google Cloud 服務帳戶。

為 OpenShift 叢集設定 Google Cloud Workload Identity Pool 和 OIDC 供應商。供應商發行者必須與權杖的 `iss`宣告相符，且其屬性對應必須將 `assertion.sub`對應至 `google.subject`。

安裝 Trident 時，將供應商資源名稱設定為 WIF 對象。這會使 Trident 專案控制器服務帳戶權杖於 `/var/run/secrets/wif-token/token`。

Trident運算子

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  cloudProvider: GCP
  gcpWIFAudience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

掌舵

```
helm install trident <trident-operator-package> \
  --namespace trident \
  --create-namespace \
  --set cloudProvider=GCP \
  --set-string gcpWIFAudience="//iam.googleapis.com/projects/<wif-pool-
project-number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

試用

```
tridentctl install \
  --namespace trident \
  --cloud-provider GCP \
  --gcp-wif-audience "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

請為 Trident 安裝和 `wipCredentialConfig.audience`使用相同的對象。請勿同時設定 `GKE cloudIdentity` 或 `--cloud-identity`。請勿修補控制器 Deployment 來新增 projected token；Trident 營運者可以覆寫該變更。

在 OpenShift Dedicated 上，OIDC 供應商可以是與其他 OpenShift 元件共享的。如果供應商的 `allowedAudiences`清單為空，Google 會接受供應商的規範資源名稱（有或沒有 `https://`前綴），因此上述對象無需更改供應商。請勿僅為 Trident 設定明確的對象清單。如果供應商已有明確的對象，請在保留所有現有值的情況下附加 Trident 對象。替換清單可能會阻止其他 OpenShift

元件交換權杖。

在 Trident 將要模擬的 Google Cloud 服務帳戶上，授予 Trident 控制器服務帳戶的 WIF 主體 `roles/iam.workloadIdentityUser`角色。對於預設的 `Trident` 控制器服務帳戶，權杖主旨為 `system:serviceaccount:trident:trident-controller`；請使用為您的安裝所設定的主旨。在包含 Google Cloud NetApp Volumes 資源的專案上，授予被模擬的服務帳戶 `roles/netapp.admin`或等效權限。

將以下程式碼段新增至 SAN 後端的 `spec`：

```
wipCredentialConfig:
  type: external_account
  audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
  subject_token_type: urn:ietf:params:oauth:token-type:jwt
  token_url: https://sts.googleapis.com/v1/token
  credential_source:
    file: /var/run/secrets/wif-token/token
  service_account_impersonation_url:
https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<service-account-email>:generateAccessToken
```

兩種 GCNV 驅動程式的 WIF 組態相同。SAN 後端仍必須使用 `google-cloud-netapp-volumes-san` 和受支援的 Flex Unified 儲存資源池。

Flex Unified 儲存池

Flex 統一儲存資源池使用 iSCSI 協定提供區塊儲存設備，是 SAN 資源配置所必需的：

- 支援 Flex Unified REGIONAL 資源池。
- 從 Trident 26.02.1 開始支援 Flex Unified ZONAL 池。
- SAN 工作負載僅支援 **Flex** 服務等級。

後端 `location`必須與 Flex Unified 資源池的區域相符。如果 Trident 找不到該資源池，請驗證區域、網路和資源池資源路徑。

設定 Trident SAN 後端

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
```

```
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
      performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex
```

建立 StorageClass

配置完 SAN 後端後，建立一個 StorageClass 引用 `google-cloud-netapp-volumes-san` 驅動程式。

請在 StorageClass 中指定檔案系統類型，而不是在後端。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

此驅動程式支援以下檔案系統類型：

- `ext4` (預設)
- `ext3`
- `xfs`

> SAN 驅動程式僅支援 Flex 服務級別，不使用 NAS 特定的後端參數，例如 `exportRule`、`unixPermissions`、`nasType`、`snapshotDir`、`nfsMountOptions`或與分層相關的設定。

驗證 WIF 配置

套用後端並驗證 TridentBackendConfig 是否已綁定且其最後一次操作成功：

```
oc apply -f <backend-file>.yaml
oc -n trident get tridentbackendconfig <backend-name> -o yaml
```

使用此頁面上的 `gcnv-san` StorageClass 建立 PVC，並驗證其是否達到 `Bound`。這將同時驗證 WIF 交換、服務帳戶模擬、Google Cloud NetApp Volumes 權限和 SAN 資源池選擇。完成後，請刪除測試 PVC。

如果後端資料核對失敗，請檢查 Trident 控制器日誌：

```
oc -n trident logs deploy/trident-controller -c trident-main --tail=400
```

請檢查以下常見錯誤：

- 對於 `invalid_grant`對象錯誤，請將權杖的 `aud`聲明與 OIDC 供應商允許的對象和 `wipCredentialConfig.audience`進行比較。
- 對於 `iam.serviceAccounts.getAccessToken`權限錯誤，請驗證 Trident 控制器的 WIF 主體是否對被模擬的服務帳戶具有 `roles/iam.workloadIdentityUser`。
- 如果發生 `netapp.storagePools.list`權限錯誤，請驗證被模擬的服務帳戶是否對正確的專案擁有所需的 Google Cloud NetApp Volumes 權限。
- 如果找不到資源池，請驗證其位置和類型是否受支援，後端區域是否與資源池區域相符，以及後端是否參照正確的網路和資源池。

配置區塊磁碟區

ReadWriteOnce（RWO）

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadWriteOncePod（RWOP）

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
```

```
  storageClassName: gcnv-san
```

ReadOnlyMany（ROX）

ROX 的一個常見模式是複製現有 ReadWriteOnce 磁碟區並將複製磁碟區掛載為唯讀。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo
```

ReadWriteMany (RWX) — 僅原始區塊

僅在 `volumeMode: Block`時才支援 ReadWriteMany。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

區塊磁碟區行為

區塊磁碟區以 iSCSI LUN 的形式配置，並以區塊裝置的形式呈現給 Kubernetes 節點。

區塊磁碟區：

- 使用 iSCSI 協定
- 支援檔案系統和原始區塊呈現
- 由 Trident 附加和管理
- 支援多種 Kubernetes 存取模式

存取模式

Trident 配置的區塊磁碟區支援下列存取模式：

- `ReadWriteOnce`（RWO）
- `ReadOnlyMany`（ROX）
- `ReadWriteOncePod`（RWOP）
- `ReadWriteMany` (RWX)，僅在以下情況下支持 `volumeMode: Block`

volumeMode 行為

`volumeMode`欄位控制區塊磁碟區的公開方式：

- `Filesystem` Trident 格式化並掛載磁碟區。
- `Block` Trident 會連接該裝置並將其作為原始區塊裝置公開。

支援的作業

使用 `google-cloud-netapp-volumes-san` 驅動程式配置的區塊磁碟區支援：

- 建立
- 刪除
- 複製
- Snapshot
- 調整大小
- 匯入

Volume 大小例行成本

Google Cloud NetApp Volumes 區塊 Volume 包含內部中繼資料例行成本。因此，核心可見的裝置大小小於已配置的容量。

觀察到的例行成本：

- 建立 Volume 時大約 300 KiB
- 調整大小後最多約 107 MiB

Google Cloud NetApp Volumes 接受以整數 GiB 為單位的容量分配。為確保可用裝置大小符合或超過 PVC 要求，Trident：

- 將要求的大小無條件進位至下一個整數 GiB。
- 新增 1 GiB 緩衝區。

例如：

- PVC 請求：100 GiB。
- 已配置容量：101 GiB。
- 應用程式可見的可用容量：至少 100 GiB。

Pod 範例

檔案系統掛載區塊磁碟區（**RWO**）

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
    - name: data
      mountPath: /mnt/data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-rwo
```

原始區塊設備（**RWX**）

```
apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
```

```
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx
```

附加和掛載行為

對於從 Google Cloud NetApp Volumes 配置的 SAN 磁碟區：

- Trident 在 Flex Unified 儲存池中建立邏輯單元號碼（LUN）。
- 發布期間，Trident 將 LUN 對應到每個節點的主機群組。
- 在節點暫存期間、 Trident ：
  - 登入 iSCSI 目標
  - 探索 LUN
  - 配置多路徑
- 如果 `volumeMode: Filesystem`，Trident 會視需要格式化裝置並將其掛載。
- 如果 `volumeMode: Block`，Trident 會將裝置連接並直接暴露給 pod，而無需格式化或掛載。

> SAN 區塊磁碟區不提供分散式鎖定或寫入協調功能。當多個節點存取區塊磁碟區時（ReadWriteMany `volumeMode: Block`），應用程式或檔案系統必須管理並發性。

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 或 **SMB** 磁碟區的先決條件

如果您是首次使用 Google Cloud NetApp Volumes 或在新的位置使用，則需要進行一些初始配置才能設定 Google Cloud NetApp Volumes 並建立 NFS 或 SMB 磁碟區。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 (`roles/netapp.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
「torageDriverName」	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。	
「apiKey」	具有此角色的 Google Cloud 服務帳戶的 API 金鑰 `netapp.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`。	
《標籤》	套用到磁碟區的任意JSON-格式化標籤集	"
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
`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`

參數	說明	預設
`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。	「0.00.0.0/0」
「napshotDir	存取「.snapshot」目錄	`true`, `false`（預設行為可能有所不同。請明確設定）NFSv3 設定為 "false"
「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。

▼ 最小組態

這是絕對最低的後端組態。有了這項組態、 Trident 會探索您在設定位置中委派給 Google Cloud NetApp Volumes 的所有儲存池、並隨機將新磁碟區放在其中一個集區上。由於省略、因此 `nasType` `nfs` 會套用預設值、而後端會為 NFS 磁碟區進行資源配置。

如果您剛開始使用 Google Cloud NetApp Volumes 並進行嘗試，此配置是理想的選擇，但在實踐中，您可能需要為配置的磁碟區提供額外的範圍。

請將 `<id_value>`和 `<key_value>`替換為您的服務帳戶憑證。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
```

```
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----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
  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
```

▼ **SMB** 磁碟區的組態

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
```

```
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    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/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

▼ 使用 **StoragePools** 篩選器進行組態

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----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
  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
```

▼ 虛擬集區組態

此後端組態會在單一檔案中定義多個虛擬集區。虛擬集區是在一節中定義 `storage` 。當您有多個儲存集區支援不同的服務層級、而且您想要在 Kubernetes 中建立代表這些層級的儲存類別時、這些功能就很有用。虛擬集區標籤用於區分集區。例如、在下面的範例中、 `performance` 標籤和 `serviceLevel` 類型是用來區分虛擬集區。

您也可以將某些預設值設定為適用於所有虛擬集區、並覆寫個別虛擬集區的預設值。在下列範例中 `snapshotReserve` 、並 `exportRule` 做為所有虛擬集區的預設值。

如需詳細資訊、請 "虛擬資源池"參閱。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----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

當您使用 GKE 雲端身分安裝 Trident 時，後端會以工作負載身分進行驗證，並省略 `apiKey`區塊。`Google Cloud NetApp Volumes` 驅動程式的雲端身分配置方式相同。唯一的差異在於 `storageDriverName` 值：`NAS`（`NFS` 或 `SMB`）磁碟區使用 `google-cloud-netapp-volumes`，SAN（iSCSI 區塊）磁碟區使用 `google-cloud-netapp-volumes-san`。

> ⓘ 驅動程式對雲端身分的支援需要 Trident 26.06 或更高版本。`google-cloud-netapp-volumes-san`SAN 磁碟區是從 `Flex Unified` 儲存池中佈建的，因此請將 `serviceLevel`設定為 `flex`。

NAS 驅動程式（**<code>google-cloud-netapp-volumes</code>**）

```
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
```

SAN 驅動程式（<code>google-cloud-netapp-volumes-san</code>）

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv-san
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: flex
  storagePool: pool-flex1
```

▼ 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。

儲存類別定義

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。

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。任何具有任何 / 無權限的 Active Directory 使用者 / 密碼都可用於節點階段密碼。

▼ 預設命名空間的基本組態

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

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

配置 **Google Cloud NetApp Volumes** 的自動分層

自動分層是透過 Trident 後端參數和 PersistentVolumeClaim 註解在磁碟區配置期間進行設定的。您可以使用 Trident 為 Google Cloud NetApp Volumes 設定自動分層。

總覽

自動分層儲存功能可讓 Trident 自動配置磁碟區，將不活躍資料從效能層移至容量層。這樣既能降低儲存成本，又能確保頻繁存取資料的效能。

Trident 僅在建立磁碟區時套用自動分層設定。Trident 26.10 不支援配置後變更。

概念

自動分層

自動分層儲存會根據存取模式，將存取頻率較低的資料從效能層移至容量層。資料移動是非同步進行的，並非即時生效。

分層原則

分層原則決定是否為磁碟區啟用自動分層。

支援以下策略：* `auto`：啟用基於存取模式的自動分層 * `none`：停用自動分層

冷卻天數

冷卻天數規定了資料區塊在符合分層儲存條件之前必須保持非活動狀態的最短天數。冷卻天數僅在分層儲存策略設定為 `auto`時適用。

組態模型

組態範圍

自動分層可在多個範圍內進行設定：

- 儲存池範圍 適用於從該池配置的所有磁碟區。

- **Volume** 範圍 透過 PersistentVolumeClaim 註解應用於單一磁碟區。

Trident 會根據每個設定的定義位置來決定有效組態。

組態優先順序

當相同設定被定義在多個作用域時，Trident 會套用下列優先順序：

1. PersistentVolumeClaim 註解
2. Trident 後端組態
3. 儲存池預設值

在較高優先順序定義的設定會覆蓋較低層級的值。

Trident 26.10 支援的功能

Trident 26.10 支援以下 Google Cloud NetApp Volumes 自動分層功能：

- 在磁碟區配置期間啟用或停用自動分層
- 在 Trident 後端組態中定義分層原則
- 使用 PVC 註解覆蓋分層原則和每個磁碟區的冷卻天數
- 為啟用自動分層的磁碟區配置冷卻日

Trident 26.10 中不支援的功能

不支援以下操作：

- 在建立磁碟區後修改自動分層設定
- 使用 Kubernetes 更新變更現有磁碟區的分層原則
- 在 Trident 管理的佈建工作流程之外套用自動分層設定

後端配置參數

以下參數可在 Trident 後端組態中定義時控制自動分層行為：

參數	必要	說明
tieringPolicy	否	磁碟區的分層原則(`auto`或 `none`)
tieringMinimumCoolingDays	否	資料分層前的非活躍天數（範圍：2–183、預設值：31）

使用 PersistentVolumeClaim 註解進行磁碟區層級覆寫

支持的註釋

PersistentVolumeClaim 註解允許對每個磁碟區的自動分層設定進行覆蓋。

註釋	說明
`trident.netapp.io/tieringPolicy`	覆寫磁碟區的分層原則
`trident.netapp.io/tieringMinimumCoolingDays`	覆寫磁碟區的冷卻天數值

範例：**PersistentVolumeClaim** 與自動分層覆寫

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

行為和限制

資源配置行為

- 自動分層設定僅在建立磁碟區時進行評估和應用。
- Trident 在配置完成後不會協調分層配置。
- 當分層策略設定為 `none`時，冷卻日將被忽略。

平台限制

- 自動分層僅支援 NAS 磁碟區（ NFS 和 SMB ）。
- 區塊磁碟區（iSCSI）不支援自動分層。
- Google Cloud NetApp Volumes 儲存池必須在 Google Cloud 中啟用自動分層儲存。

支援的值

- `tieringMinimumCoolingDays` 的有效範圍：2 至 183
- 預設值：31

設定 **Google Cloud NetApp Volumes ONTAP** 模式儲存池

從 Trident 26.06 開始，您可以設定現有 `ontap-san`和 `ontap-nas`後端，以便在 Google

Cloud NetApp Volumes （GCNV） ONTAP 模式儲存池上佈建儲存。此功能為技術預覽。

註	此功能在 Trident 26.10 中為技術預覽版。技術預覽功能不支援在生產環境中使用。在正式上市（GA） 之前，其功能、組態欄位和支援的矩陣可能會有所變更。

工作原理

Trident 重複使用現有的 `ontap-san` 和 `ontap-nas` 驅動程式邏輯，並透過 GCNV 代理端點路由 ONTAP REST 呼叫。此方法可讓您在裝載於 GCNV 的 ONTAP 叢集上佈建和管理儲存設備，同時保持與直接 ONTAP 後端相同的作業工作流程。ONTAP 模式不會引入新的儲存驅動程式。您可以透過將 `gcnv` 組態區塊新增至 `ontap-san` 或 `ontap-nas` 後端，來針對每個後端啟用它。

配置路徑如下：

`PersistentVolumeClaim` → Trident (`ontap-san` 或 `ontap-nas`) → ONTAP REST 用戶端 → GCNV 代理程式 → GCNV 集區中的 ONTAP 叢集

受支援和不受支援的範圍

此技術預覽版支援以下功能：

- 驅動程式： `ontap-san`（iSCSI）和 `ontap-nas`（NFS 或 SMB）。
- 透過代理執行生命週期操作：後端初始化、磁碟區建立和刪除以及相關的 ONTAP REST 流程。
- 驗證模型：工作負載身分集區 (WIP)、服務帳戶金鑰和應用程式預設憑證 (ADC) 備援。

以下內容不在本次技術預覽的範圍內：

- `ontap-san-economy`、 `ontap-nas-economy` 以及 `ontap-nas-flexgroup` 驅動程式。
- ASA r2 特性路徑。
- ZAPI 回退。ONTAP 模式僅使用 ONTAP REST。

先決條件

在設定 ONTAP 模式後端之前，請確保您已具備以下條件：

- Trident 26.06 或更高版本。
- 在目標專案和位置中有一個 GCNV ONTAP 模式儲存池。
- GCP IAM 代理程式操作權限，並已根據您的環境進行適當範圍設定。
- 後端配置包括 `proxyURL`、 `projectNumber`、 `location`和 `poolID`。
- 有關 iSCSI SAN 工作流程，請參閱平台所需的節點端 iSCSI 和多路徑先決條件。請參閱 "準備工作節點"。

配置後端

A `TridentBackendConfig` 需要 `credentials` 參照 Kubernetes Secret。對於 ONTAP 模式，此 Secret 提供 GCP Proxy 存取的認證資料，而非 ONTAP `managementLIF` / `username` / `password`。根據身分驗證方法的不同，它可以包含服務帳戶金鑰欄位，也可以是空白。 `svm` 是選用的：每個 GCNV ONTAP 模式資源池目前都有一個 SVM，如果省略，Trident 會從資源池衍生出該 SVM。

對於服務帳戶金鑰驗證，請使用與原生 GCNV 相同的模式：非敏感欄位位於 `gcnv.apiKey`中， `private_key`和 `private_key_id`位於 secret 中。請參閱 "Google Cloud NetApp Volumes 設定範例"。對於工作負載身分資源池或 GKE 雲端身分驗證，請參閱 驗證 中的範例。

ontap-san 後端

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-san
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  backendName: gcnv-ontap-san
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
  svm: "<svm-name>"
```

ontap-nas 後端

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
    apiKey:
      type: service_account
```

```
      project_id: "<project-id>"
      client_email: "<service-account-email>"
      client_id: "<client-id>"
      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: "<client-x509-cert-url>"
  svm: "<svm-name>"
```

秘密

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-sa-secret
  namespace: trident
type: Opaque
stringData:
  private_key_id: "<private-key-id>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <private-key>
    -----END PRIVATE KEY-----
```

註	請勿將 `private_key`或 `private_key_id`放入 `TridentBackendConfig`規格中——CRD 會拒絕它們。對於區塊儲存，請使用 `ontap-san`搭配相同的 `gcnv`和憑證模式。

驗證

Trident 會依照下列順序解析 Proxy 存取認證資料：

1. 工作負載身分資源池（WIP）
2. 服務帳戶金鑰
3. 應用程式預設認證資料（ADC）

對於本次技術預覽，請遵循以下做法：

- 優先選擇工作負載身分資源池（如果可用）。
- 不要將原始私鑰嵌入到版本控制的後端檔案中。將認證資料儲存在 Kubernetes Secret 中。
- 對服務帳戶套用最小權限 IAM 範圍。

`location`中的 `gcnv`區塊必須與資源池的位置相符：區域性資源池使用

zone，地區性資源池使用 region。

工作負載身分資源池

將 gcnv.wipCredential 用於在 TridentBackendConfig 上設定的明確外部帳戶聯盟。在 OpenShift 上，Trident 會將權杖投射到 /var/run/secrets/wif-token/token；這是 Trident 為 OpenShift WIF 投射的唯一權杖路徑，因此請將其用於 credential_source.file。如 "設定 Google Cloud NetApp Volumes 與工作負載身分聯盟" 所述，使用 WIF 對象安裝 Trident。 credentials 欄位仍參照 Secret，但 Secret 不得包含 private_key。

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-ontap-empty
  namespace: trident
type: Opaque
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas-wip
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas-wip
  useREST: true
  nasType: nfs
  nfsMountOptions: nfsvers=4.1
  credentials:
    name: gcnv-ontap-empty
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
    wipCredential:
      type: external_account
      audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<wif-pool-
id>/providers/<provider-id>"
      subject_token_type: "urn:ietf:params:oauth:token-type:jwt"
      token_url: "https://sts.googleapis.com/v1/token"
      service_account_impersonation_url:
"https://iamcredentials.googleapis.com/v1/projects/-
```

```
/serviceAccounts/<service-account-email>:generateAccessToken"
      credential_source:
        file: "/var/run/secrets/wif-token/token"
```

GKE 雲端身分

對於 GKE，請在安裝時依照 "使用 GKE 工作負載身分" 中的說明設定 Trident 的雲端身分。後端使用應用程式預設認證資料（ADC）；請省略 `gcnv.apiKey`和 `gcnv.wipCredential`。`credentials` 欄位仍然參照 Secret，但請勿在其中放置 `private_key`。

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-ontap-empty
  namespace: trident
type: Opaque
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas-wi
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas-wi
  useREST: true
  nasType: nfs
  nfsMountOptions: nfsvers=4.1
  credentials:
    name: gcnv-ontap-empty
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
```

驗證和初始化行為

Trident 在初始化期間驗證 ONTAP 模式後端，並快速失敗而不是部分初始化。

如果缺少必要 `gcnv`欄位、驅動程式不支援 ONTAP 模式或認證資料解析失敗，則後端初始化失敗。

已知限制

- 此功能目前處於技術預覽階段。在 GA 前，其行為和支援的對照表可能會有所更改。
- ONTAP 模式下不使用 ZAPI 回退機制。
- 協定和節點就緒要求仍然適用。例如，iSCSI 工作流程需要 "準備工作節點" 中所述的節點端先決條件。
- 現有的 ONTAP 和 GCNV 環境限制仍然適用。

升級和相容性

ONTAP 模式可透過 `gcnv`組態區塊按後端選擇啟用：

- 現有的非 GCNV ONTAP 後端不受影響。
- 透過後端層級的組態，支援結合直接 ONTAP 後端和 ONTAP 模式 Proxy 後端的混合部署。每個後端都是獨立設定的。
- 若要停止使用 ONTAP 模式，請移除或取代受影響的後端。不需要全域交換器。

接下來呢？

在 TBC 之前套用機密；使用 `kubectl apply` 和 `-n trident` 來符合範例。

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` 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
「torageDriverName」	儲存驅動程式名稱	永遠為「 SolidFire - SAN 」
「後端名稱」	自訂名稱或儲存後端	「 SolidFire _ 」 + 儲存（ iSCSI ）IP 位址
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。例如、 {"api"：false、"method"：true}	null

警告 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 SCSI over FC	區塊	Rwo 、 ROX 、 rwx 、 RWOP	無檔案系統；原始區塊裝置
「ONTAP-SAN」	iSCSI SCSI over FC	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》

警告

- 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、
- ONTAP Volume 自動調整規模支援 `ontap-san` 和 `ontap-san-economy`。ONTAP NAS 驅動程式不支援此功能。如需詳細資料和限制，請參閱 "設定 ONTAP Volume 自動調整規模"。

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。

> 註 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
- Linux 統一金鑰設定（LUKS）加密

> 註 NVMe 僅支援ONTAP REST API，不支援 ONTAPI (ZAPI)。

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 要求至少將一個聚合分配給 SVM。

> 註 "ASA r2 系統"與其他ONTAP系統（ASA、 AFF和FAS）在儲存層的實作上有所不同。在ASA r2

> 系統中，使用儲存可用區而不是聚合。請參閱"這"知識庫文章，介紹如何在ASA r2 系統中將聚合指派給 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。

> 警告 如果您嘗試同時提供*認證與認證*、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

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

JSON

```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。

啟用基於憑證的身份驗證

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CACertifate：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
```

> 註 執行此命令後，ONTAP 會提示輸入憑證。貼上步驟 1 中產生的 `k8senv.pem`檔案內容，然

後輸入 `END`以完成安裝。

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...Vaaalllluuuueeee",
"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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

註	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** 驗證連接

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
```

警告	「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。

工作原理

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。

輪換憑證並更新後端

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。

警告	更新後端的 CHAP 機密時、您必須使用 `tridentctl`來更新後端。請勿使用 ONTAP CLI 或 ONTAP 系統管理員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

在 Trident 安裝中建立和使用 ONTAP SAN 驅動程式。本節提供後端組態範例，以及將後端映射至 StorageClasses 的詳細資訊。"ASA r2 系統"與其他 ONTAP 系統（ASA、AFF 和 FAS）在儲存層的實作上有所不同。這些差異會影響某些參數的使用，如註釋所示。"詳細了解 ASA r2 系統與其他 ONTAP 系統之間的差異"。在 Trident 後端組態中，您無需指定系統為 ASA r2。當您選擇 `ontap-san`作為 `storageDriverName`時，Trident 會自動偵測

ASA r2 或其他 ONTAP 系統。如下表所示，某些後端組態參數不適用於 ASA r2 系統。

註	只有 `ontap-san`ASA r2 系統支援驅動程式（支援 iSCSI、NVMe/TCP 和 FC 協定）。

後端組態選項

從 26.06 版本開始，您可以啟用 ONTAP 均衡放置，以便 ONTAP 為新 Volume 選擇彙總。請參閱 "使用 ONTAP 後端進行均衡放置" 以瞭解需求和範例。

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
「torageDriverName」	儲存設備驅動程式的名稱。建立後無法修改。若要更新此參數，必須建立新的後端。	`ontap-san`或 `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MetroCluster 範例、請參閱。 註 如果您使用的是「 vsadmin 」認證，則必須是 SVM 的認 `managementLIF`證；如果使用的是「 `admin` 」認證，則必須是叢集的認證 `managementLIF`。	"10.0.0.1 " ，"[2001:1234:abcd::fefe]
「DataLIF」	傳輸協定LIF的IP位址。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]`。* 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 **iSCSI** 生命。如果明確定義、就會產生警告 **`dataLIF`**。MetroCluster 省略。 *請參閱MetroCluster 範例。	源自SVM
《虛擬機器》	要使用的儲存虛擬機器 * MetroCluster 請省略。 * 請參閱 MetroCluster 範例。	如果指定SVM "managementLIF"則衍生

參數	說明	預設
《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。*不支援 FCP 或 NVMe/TCP。 *	「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	"
「clientPrivateKey」	用戶端私密金鑰的Base64編碼值。用於憑證型驗證	"
「可信賴的CACertifcate」	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叢集通訊所需的使用者名稱。用於基於憑證的身份驗證。有關 Active Directory 驗證，請參閱 "使用 Active Directory 憑證向後端 SVM 驗證Trident 的身份"。	"
密碼	與ONTAP集群通訊所需的密碼。用於基於憑證的身份驗證。有關 Active Directory 驗證，請參閱 "使用 Active Directory 憑證向後端 SVM 驗證Trident 的身份"。	"
《虛擬機器》	要使用的儲存虛擬機器	如果指定SVM "managementLIF"則衍生
「torагePrefix」	在 SVM 中為新 Volume 進行資源配置時所使用的前置詞。之後無法修改。若要更新此參數，您需要建立新的後端。	`trident`

參數	說明	預設
《Aggregate》	用於資源配置的彙總（可選；如果設定，則必須指派給 SVM）。對於 `ontap-nas-flexgroup`驅動程式，此選項將被忽略。如果未指派，則可以使用任何可用的彙總來資源配置 FlexGroup Volume。 註：當 SVM 中的彙總更新時，Trident 會自動輪詢 SVM 並更新，無需重新啟動 Trident 控制器。如果您已在 Trident 中設定了特定的彙總來資源配置 Volume，並且該彙總被重新命名或從 SVM 移出，則在輪詢 SVM 彙總期間，Trident 中的後端將進入故障狀態。您必須將該彙總變更為 SVM 中存在的彙總，或將其完全移除，才能使後端恢復線上。 不要指定 **ASA r2** 系統。 請勿指定何時 **`useBalancedPlacement`**設定為 **`true`**。	"
`useBalancedPlacement`	將新 Volume 的彙總選擇委託給 ONTAP [布林值]。需要 ONTAP 9.19.1 或更新版本，且 `useREST`設定為 `true`。不能與 `aggregate`組合使用。 請參考 "使用 ONTAP 後端進行均衡放置" 以了解需求和範例。	「假」
「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。不要指定 **ASA r2** 系統。	""（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」	疑難排解時要使用的偵錯旗標。例如、 {"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。	`null`

參數	說明	預設
《useREST》	使用ONTAP REST API 的布林參數。 `useREST`當設定為 `true`，Trident使用ONTAP REST API與後端通訊；當設定為 `false`，Trident使用 ONTAPI（ZAPI）呼叫與後端進行通訊。此功能需要ONTAP 9.11.1 及更高版本。此外，使用的ONTAP登入角色必須具有訪問 `ontapi`應用。這是透過預定義的 `vsadmin`和 `cluster-admin`角色。從Trident 24.06 版本和ONTAP 9.15.1 或更高版本開始，`useREST`設定為 `true`預設；改變 `useREST`到 `false`使用 ONTAPI (ZAPI) 呼叫。 `useREST`完全符合 NVMe/TCP 的要求。 註 NVMe 僅支援ONTAP REST API，不支援 ONTAPI (ZAPI)。 如果指定，則始終設定為 **`true`**適用於**ASA r2** 系統。	true 對於 ONTAP 9.15.1 或更高版本，否則 false。
sanType	用於選擇 iscsi iSCSI 、 nvme NVMe / TCP 或 fcp SCSI over Fibre Channel （ FC ）。	iscsi 如果空白
formatOptions	使用 `formatOptions`為 `mkfs`命令指定命令列參數，此命令會在每次格式化 Volume 時套用。這可讓您根據自己的偏好格式化 Volume。請確保指定的 formatOptions 與 mkfs 命令選項類似，但不包含裝置路徑。例如："-E nodiscard" *支援 `ontap-san`和 `ontap-san-economy`帶有 iSCSI 協定的驅動程式。 **此外，在使用 iSCSI 和 NVMe/TCP 協定時，支援 ASA r2 系統。 *	
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建議採用以下選項來加速格式化流程：

- **-E nodiscard (ext3, ext4):** 不要嘗試在 mkfs 時丟棄區塊（最初丟棄區塊對固態設備和稀疏/精簡配置儲存很有用）。這將取代已棄用的選項"-K"，並且適用於 ext3、ext4 檔案系統。
- **-K (xfs):** 不要在執行 mkfs 時嘗試丟棄區塊。此選項適用於 xfs 檔案系統。

使用 **Active Directory** 憑證向後端 **SVM** 驗證**Trident** 的身份

您可以設定Trident以使用 Active Directory (AD) 憑證對後端 SVM 進行驗證。在 AD 帳戶可以存取 SVM 之前，您必須設定 AD 網域控制站對叢集或 SVM 的存取權限。對於使用 AD 帳戶進行叢集管理，您必須建立網域隧道。參考 "在ONTAP中設定 Active Directory 網域控制站存取" 了解詳情。

步驟

1. 為後端 SVM 配置網域名稱系統 (DNS) 設定：

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. 執行下列命令在 Active Directory 中為 SVM 建立電腦帳戶：

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. 使用此命令建立 AD 使用者或群組來管理叢集或 SVM

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. 在Trident後端設定檔中，設定 `username` 和 `password` 參數分別為 AD 使用者或群組名稱和密碼。

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）"	LUN的空間分配	"true" 如果指定，則設定為 **`true`**適用於 **ASA r2** 系統。
《保護區》	空間保留模式；「無」（精簡）或「 Volume （大量）」（粗）。設定為 **`none`**適用於 **ASA r2** 系統。	" 無 "
「快照原則」	要使用的 Snapshot 原則。設定為 **`none`**適用於 **ASA r2** 系統。	" 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。不要為 **ASA r2** 系統指定。	「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 Volume 上啟用 NetApp Volume Encryption (NVE)；預設為 `false`。必須在叢集上取得 NVE 授權並啟用，才能使用此選項。如果後端已啟用 NAE，則在 Trident 中資源配置的任何 Volume 都會啟用 NAE。如需詳細資訊，請參閱："Trident 如何與 NVE 和 NAE 搭配運作"。	“false” 如果指定，則設定為 **`true`**適用於 **ASA r2** 系統。
`luksEncryption`	啟用LUKS加密。請參閱 "使用Linux統一金鑰設定（LUKS）"。	"" 設定為 **`false`**適用於 **ASA r2** 系統。
「分層政策」	分層策略使用「無」 不要為 **ASA r2** 系統指定。	
`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'
```

註

對於使用此 **`ontap-san`**驅動程式建立的所有 Volume，Trident 會額外增加 10% 的容量至 FlexVol，以容納 LUN 中繼資料。LUN 的資源配置大小與使用者在 PVC 中要求的大小完全一致。Trident 額外增加 10% 至 FlexVol（在 ONTAP 中顯示為可用大小）。使用者現在可獲得所要求的可用容量。此變更也可防止 LUN 在可用空間未完全使用之前變為唯讀狀態。此變更不適用於 ontap-san-

economy。

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Trident為容納 LUN 元資料而額外添加到FlexVol 的10%。對於 `snapshotReserve` = 5%，PVC 請求 = 5 GiB，則總磁碟區大小為 5.79 GiB，可用大小為 5.5 GiB。 `volume show`命令應顯示與此範例類似的結果：



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。

> 註 如果您在 NetApp ONTAP 上搭配 Trident 使用 Amazon FSX ， NetApp 建議您指定生命體的 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>
```

▼ **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，請使用並省略 `svm`參數來指定 `SVM `managementLIF`。例如：

```
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
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>
```

▼ 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>
```

▼ 均衡佈局範例

此範例啟用 ONTAP 平衡放置，以便 ONTAP 為新 Volume 選擇彙總。平衡放置需要 ONTAP 9.19.1 或更新版本。如需詳細資訊，請參閱 "使用 ONTAP 後端進行均衡放置"。

```
---
version: 1
storageDriverName: ontap-san
backendName: san-balanced-placement
managementLIF: 10.0.0.1
svm: svm_iscsi
username: vsadmin
password: <password>
useBalancedPlacement: true
```

▼ **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
```

▼ **SCSI over FC （ FCP ）範例**

您必須在 ONTAP 後端設定具有 FC 的 SVM 。這是 FC 的基本後端組態。

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

▼ 名稱範本的後端組態範例

```
---
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}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

▼ **formatOptions ONTAP - SAN 經濟型驅動程式範例**

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

虛擬集區的後端範例

在這些後端定義檔案範例中、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。在 FlexVol volume Trident 上設定的註解會將虛擬集區上的所有標籤複製到資源配置時的儲存磁碟區。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。

▼ **ONTAP SAN 範例**

```
---
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"
```

▼ **ONTAP SAN 經濟效益範例**

```
---
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>
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

以下 StorageClass 定義均參照 [虛擬集區的後端範例]。使用 `parameters.selector` 欄位，每個 StorageClass 都會指出哪些虛擬資源池可用於裝載 Volume。Volume 現在具有所選虛擬資源池中定義的屬性。

- The `protection-gold` StorageClass 映射到 `ontap-san` 後端的第一個虛擬資源池。這是唯一提供黃金級保護的資源池。

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

- The `protection-not-gold` StorageClass 映射至 `ontap-san`後端中的第二和第三個虛擬資源池。這些是唯一提供金級以外其他保護層級的資源池。

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

- The `app-mysqldb` StorageClass 映射到 `ontap-san-economy`後端中的第三個虛擬儲存資源池。這是唯一一個為 mysqldb 類型應用程式提供儲存資源池組態的儲存資源池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
```

```
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- The `protection-silver-creditpoints-20k` StorageClass 對應至 `ontap-san` 後端中的第二個虛擬資源池。這是唯一提供白銀級保護和 20000 信用點的資源池。

```
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"
```

- The `creditpoints-5k` StorageClass 分別對應 `ontap-san`後端中的第三個虛擬資源池和 `ontap-san-economy`後端中的第四個虛擬資源池。只有這兩個資源池產品提供 5000 creditpoints。

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

- StorageClass `my-test-app-sc`映射到 `testAPP`驅動程式中的虛擬資源池 `ontap-san`，並搭配 `sanType: nvme`。這是唯一提供 `testApp`的資源池。

```
---
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 決定選擇哪個虛擬資源池，並確保符合儲存需求。

啟用 NVMe 資料 LIF 重新整理，讓已發佈的 NVMe Volume 能夠使用已新增、移除或指派新位址的 LIF，而無需重新建立後端或重新啟動 Trident。

總覽

當您新增 ONTAP SAN 後端時，Trident 會發現 SVM 的 NVMe 資料 LIF。預設情況下，Trident 會保留這組 LIF，直到您再次新增後端或重新啟動 Trident。

將 `enableDataLIFRefresh`設定為 `true`，以便在 Trident 執行時重新整理 NVMe 資料 LIF。已發佈的 NVMe Volume 接著會使用已新增、已移除或已指派新位址的 LIF。

此選項為安裝參數，而非後端組態參數。預設為停用。

從 Trident 26.10 開始，可使用 NVMe 資料 LIF 重新整理。

此功能適用於：

- `ontap-san` 搭配 `sanType: nvme` 於統一的 ONTAP 上。
- `ontap-san`在 ASA r2 上使用 NVMe/TCP。

警告	同時移除或取代所有資料 LIF 會造成中斷。如果您打算移除或取代所有資料 LIF，請確保先部署一個替代 LIF，以避免資料存取中斷。

啟用 **NVMe** 資料 **LIF** 重新整理

Trident運算子

若要進行新的安裝，請在 `TridentOrchestrator` spec 中設定 `enableDataLIFRefresh`：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableDataLIFRefresh: true
```

套用清單：

```
kubectl apply -f tridentorchestrator_cr.yaml
```

對於現有安裝，請修補 `TridentOrchestrator`：

```
kubectl patch torc trident --type=merge -p
```

```
'{"spec":{"enableDataLIFRefresh":true}}'
```

驗證所應用的值：

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableDataLIFRefresh}'
```

掌舵

對於全新安裝，請設定 Helm 值：

```
helm install trident netapp-trident/trident-operator \
  --namespace trident --create-namespace \
  --set enableDataLIFRefresh=true
```

對於現有的 Helm 安裝，請使用啟用的值升級版本：

```
helm upgrade trident netapp-trident/trident-operator \
  --namespace trident \
  --reuse-values \
  --set enableDataLIFRefresh=true
```

您也可以在 Helm 值檔案中設定 `enableDataLIFRefresh: true`，並透過 `-f <values-file>`將其傳遞給 `helm upgrade`。

試用

對於全新安裝，請使用 `--enable-datalif-refresh` 旗標：

```
tridentctl install --enable-datalif-refresh -n trident
```

若要在現有 `tridentctl`安裝中啟用此選項，請解除安裝 Trident 並使用該旗標重新安裝。解除安裝前，請遵循 "解除安裝Trident" 中的注意事項。

```
tridentctl uninstall -n trident
tridentctl install --enable-datalif-refresh -n trident
```

註	此 `--enable-datalif-refresh`標誌是安裝選項，不會顯示在 `tridentctl install --help`中。

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

ONTAP NAS 驅動程式詳細資料

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`

警告

- 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、
- ONTAP NAS 驅動程式不支援 ONTAP Volume 自動調整規模。如需支援的 SAN 驅動程式與組態詳細資訊，請參閱 "設定 ONTAP Volume 自動調整規模"。

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。

註

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。

準備使用ONTAP 不含NAS的驅動程式來設定後端

了解使用 ONTAP NAS 驅動程式配置 ONTAP 後端的要求、驗證選項和匯出原則。從

25.10 版本開始，NetApp Trident 支援 "NetApp AFX儲存系統"。NetApp AFX 儲存系統與其他 ONTAP 系統（ASA、AFF 和 FAS）在儲存層的實作方式上有所不同。在 Trident 後端組態中、您無需指定系統為 AFX。當您選擇 `ontap-nas`作為 `storageDriverName` 時、Trident 會自動偵測 AFX 系統。

註	只有 `ontap-nas`AFX 系統支援 NFS 協定驅動程式；不支援 SMB 協定。

需求

- 對於所有 ONTAP 後端，Trident 要求至少將一個聚合分配給 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。

警告	如果您嘗試同時提供*認證與認證*、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

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
```

```
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

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CACertifate：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...Vaaalllluuuueeee",
"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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

註	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。

> 註 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。

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
```

註	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 volume 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「AutoExpportPolicy」和「AutoExpportCTR」。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>
      fileSystemType: 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。

註	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：

警告	您必須在 SVM 上同時設定 NFS 和 SMB/CIFS 通訊協定，才能為 ONTAP 內部部署叢集建立 `ontap-nas-economy` 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。
註	`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。

   註 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
```

   註 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。*	"

啟用安全 **SMB**

從 25.06 版本開始，NetApp Trident 支援使用以下方式建立的 SMB 磁碟區的安全性配置 `ontap-nas`和 `ontap-nas-economy`後端。啟用安全 SMB 後，您可以使用存取控制清單 (ACL) 為 Active Directory (AD) 使用者和使用者群組提供對 SMB 共用的受控存取。

值得記住的重點

- 輸入 `ontap-nas-economy`不支援卷。
- 僅支援唯讀克隆 `ontap-nas-economy`卷。
- 如果啟用了安全 SMB，Trident 將忽略後端提到的 SMB 共用。
- 更新 PVC 註解、儲存類別註解和後端欄位不會更新 SMB 共用 ACL。
- 克隆 PVC 註釋中指定的 SMB 共用 ACL 將優先於來源 PVC 中的 ACL。
- 啟用安全 SMB 時，請確保提供有效的 AD 使用者。無效使用者將不會被加入到 ACL。
- 如果您在後端、儲存類別和 PVC 中為同一個 AD 使用者提供不同的權限，則權限優先權為：PVC、儲存類別、後端。
- 安全 SMB 支持 `ontap-nas`託管磁碟區匯入，不適用於非託管磁碟區匯入。

步驟

1. 在 TridentBackendConfig 中指定 adAdminUser，如下例所示：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
```

```
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. 在儲存類別中加入註解。

   添加 `trident.netapp.io/smbShareAdUser`註解到儲存類，以啟用安全 SMB 而不會失敗。為註釋指定的使用者值 `trident.netapp.io/smbShareAdUser`應該與 `smbcreds`秘密。您可以選擇以下其中之一 `smbShareAdUserPermission ： full_control ， change ， 或者 read 。預設權限是 full_control 。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. 建立PVC。

   以下範例建立 PVC：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
```

```
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

列舉**NAS**組態選項與範例**ONTAP**

在 Trident 安裝中建立和使用 ONTAP NAS 驅動程式。本節提供後端組態範例，以及將後端映射至 StorageClasses 的詳細資訊。從 25.10 版本開始，NetApp Trident 支援"NetApp AFX 儲存系統"。NetApp AFX 儲存系統與其他基於 ONTAP 的系統（ASA、AFF 和 FAS）在儲存層的實作上有所不同。

> 註 只有 `ontap-nas`NetApp AFX 系統支援 NFS 協定驅動程式；不支援 SMB 協定。

後端組態選項

在 Trident 後端組態中、您不需要指定系統是 NetApp AFX 儲存系統。當您選取 `ontap-nas`作為 `storageDriverName`時、 Trident 會自動偵測 AFX 儲存系統。部分後端組態參數不適用於 AFX 儲存系統。

從 26.06 版本開始，您可以啟用 ONTAP 均衡放置功能，以便 ONTAP 為新 Volume 選擇彙總。ONTAP 9.19.1 或更新版本上的 AFX 儲存系統需要啟用均衡放置功能。有關需求和範例，請參閱 "使用 ONTAP 後端進行均衡放置"。

下表顯示後端組態選項：

參數	說明	預設
「分度」		永遠為1
「torageDriverName」	儲存設備驅動程式的名稱。 建立後無法修改。若要更新此參數，必須建立新的後端。 註 僅適用於NetApp AFX 系統 `ontap-nas` 已支援。	`ontap-nas`、 `ontap-nas-economy`或 `ontap-nas-flexgroup`
「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MetroCluster 範例、請參閱。	"10.0.0.1 "，"[2001:1234:abcd::fefe]

參數	說明	預設
「DataLIF」	傳輸協定LIF的IP位址。NetApp 建議指定 `dataLIF`。如果未提供， Trident 會從 SVM 擷取 dataLIFs 。您可以指定完整網域名稱（ FQDN ），以用於 NFS 裝載作業，讓您建立循環 DNS ，以便在多個 dataLIFs 之間進行負載平衡。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MetroCluster 範例。	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 MetroCluster 範例。	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	"
「clientPrivateKey」	用戶端私密金鑰的Base64編碼值。用於憑證型驗證	"
「可信賴的CACertifcate」	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》	用於連接到叢集/SVM 的使用者名稱。用於基於憑證的身份驗證。有關 Active Directory 驗證，請參閱 "使用 Active Directory 憑證向後端 SVM 驗證Trident 的身份"。	
密碼	連接到叢集/SVM 的密碼。用於基於憑證的身份驗證。有關 Active Directory 驗證，請參閱 "使用 Active Directory 憑證向後端 SVM 驗證Trident 的身份"。	
「toragePrefix」	在SVM中配置新磁碟區時所使用的前置碼。設定後無法更新 註：當使用 ontap-nas-economy 且 storagePrefix 為 24 個或更多字元時，qtree 中不會嵌入儲存設備前綴，但它存在於 Volume 名稱中。	" Trident "

參數	說明	預設
《Aggregate》	用於資源配置的彙總（可選；如果設定，則必須指派給 SVM）。對於 `ontap-nas-flexgroup`驅動程式，此選項將被忽略。如果未指派，則可以使用任何可用的彙總來資源配置 FlexGroup Volume。 註：當 SVM 中的彙總更新時，Trident 會自動輪詢 SVM 並更新，無需重新啟動 Trident 控制器。如果您已在 Trident 中設定了特定的彙總來資源配置 Volume，且該彙總被重新命名或從 SVM 移出，則在輪詢 SVM 彙總時，Trident 中的後端將進入故障狀態。您必須將該彙總變更為 SVM 中存在的彙總，或將其完全移除，才能使後端恢復線上。 *請勿指定用於 AFX 儲存系統。 * 請勿指定何時 **`useBalancedPlacement`**設定為 **`true`**。	"
`useBalancedPlacement`	將新 Volume 的彙總選擇委託給 ONTAP [布林值]。需要 ONTAP 9.19.1 或更新版本，且 `useREST`**設定為** `true`。不能與 `aggregate`組合使用。 請參考 "使用 ONTAP 後端進行均衡放置" 以了解需求和範例。 如果指定，對於 **ONTAP 9.19.1** 或更新版本的 **AFF** 儲存系統，一律設定為 **`true`**。	`false`; `true` 適用於 ONTAP 9.19.1 或更新版本的 AFX 儲存系統
「限制Aggregateusage」	如果使用率超過此百分比，則配置失敗。不適用於**Amazon FSx for ONTAP**。*請勿指定用於 AFX 儲存系統。 *	""（預設不強制執行）
FlexgroupAggregateList	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。 註：當 SVM 中的彙總列表更新時，Trident 會自動輪詢 SVM 來更新該列表，無需重新啟動 Trident 控制器。如果您已在 Trident 中設定了特定的彙總清單來資源配置 Volume，則如果該彙總清單已重新命名或從 SVM 中移出，Trident 在輪詢 SVM 彙總時會將後端置於故障狀態。您必須將該彙總列表變更為 SVM 中存在的列表，或將其完全移除，才能使後端恢復線上。	"
《限制Volume大小》	如果請求的磁碟區大小大於此值，則配置失敗。	""（預設不會強制執行）

參數	說明	預設
「DebugTraceFlags」	疑難排解時要使用的偵錯旗標。例如、 {"api" ： false 、 "method" ： true} 請勿使用 `debugTraceFlags` 除非您正在疑難排解並需要詳細的記錄傾印。	null
`nasType`	配置 NFS 或 SMB 磁碟區的建立。選項有 `nfs`，`smb`或空值。設定為 null 則預設使用 NFS 磁碟區。如果指定，則始終設定為 **`nfs`**適用於 **AFX** 儲存系統。	`nfs`
「nfsMountOptions」	以逗號分隔的 NFS 裝載選項清單。Kubernetes 持久性磁碟區的裝載選項通常在儲存設備類別中指定，但如果儲存設備類別中未指定任何裝載選項，Trident 將回退到使用儲存設備後端組態檔案中指定的裝載選項。如果儲存設備類別和組態檔案中均未指定裝載選項，Trident 將不會在關聯的持久性 Volume 上設定任何裝載選項。	"
"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 REST API 的布林參數。 `useREST`設定為 `true`Trident使用ONTAP REST API 與後端通訊；當設定為 `false`Trident使用 ONTAPI (ZAPI) 呼叫與後端通訊。此功能需要ONTAP 9.11.1 及更高版本。此外，所使用的ONTAP登入角色必須具有存取權限。 `ontapi`應用。預定義項滿足了這一點。 `vsadmin`和 `cluster-admin`角色。從Trident 24.06 版本和ONTAP 9.15.1 或更高版本開始， `useREST`設定為 `true`預設；更改 `useREST`到 `false`使用 ONTAPI (ZAPI) 呼叫。如果指定，則始終設定為 **`true`**適用於 **AFX** 儲存系統。	`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 。	
`adAdminUser`	具有 SMB 共用完全存取權限的 Active Directory 管理員使用者或使用者群組。使用此參數可為 SMB 共用提供具有完全控制權的管理員權限。	

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。	"
「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 Volume 上啟用 NetApp Volume Encryption (NVE)；預設為 `false`。必須在叢集上取得 NVE 授權並啟用，才能使用此選項。如果後端已啟用 NAE，則在 Trident 中資源配置的任何 Volume 都會啟用 NAE。如需詳細資訊，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」	
「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）
「napshotDir	控制對的存取 `.snapshot` 目錄	`true`, `false`（明確設定）。
「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。	"

註	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"
```

為了 `ontap-nas`和 `ontap-nas-flexgroups`Trident現在使用新的計算方法，以確保FlexVol 的尺寸與 snapshotReserve 百分比和 PVC 正確匹配。當使用者要求 PVC 時， Trident會使用新的計算方法建立具有更多空間的原始FlexVol 。此計算可確保使用者在 PVC 中獲得其請求的可寫入空間，而不是少於其請求的空間。在 v21.07 之前，當使用者要求 PVC（例如 5 GiB）時，如果快照預留百分比為 50%，則只能獲得 2.5 GiB 的可寫入空間。這是因為使用者要求的是整個磁碟區。 `snapshotReserve`是其中的百分比。在Trident 21.07 中，使用者要求的是可寫入空間，而Trident定義了該空間。 `snapshotReserve`將數值表示為佔總體積的百分比。這不適用於 `ontap-nas-economy`。請參閱以下範例以了解其工作原理：

計算方式如下：

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

對於快照預留 = 50% 且 PVC 請求 = 5 GiB 的情況，總磁碟區大小為 5/.5 = 10 GiB，可用大小為 5 GiB，這正是使用者在 PVC 請求中請求的大小。 `volume show`命令應顯示與此範例類似的結果：



升級 Trident 時，對於先前安裝的現有後端，請依照上述說明進行 Volume 資源配置。對於升級前建立的 Volume，您需要調整其大小才能使變更生效。例如，2 GiB 的 PVC 搭配 `snapshotReserve=50`先前產生的 Volume 僅提供 1 GiB 的可寫入空間。例如，將 Volume 調整為 3 GiB，即可為應用程式在 6 GiB 的 Volume 上提供 3 GiB 的可寫入空間。

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。

> 註 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
```

▼ 均衡佈局範例

此範例啟用 ONTAP 平衡放置，以便 ONTAP 為新 Volume 選擇彙總。平衡放置需要 ONTAP 9.19.1 或更新版本。如需詳細資訊，請參閱 "使用 ONTAP 後端進行均衡放置"。

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-balanced-placement
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
useBalancedPlacement: true
```

▼ **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_
```

▼ 名稱範本的後端組態範例

```
---
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}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

虛擬集區的後端範例

在下面顯示的後端定義檔案範例中、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。

▼ **ONTAP NAS** 範例

```
---
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"
```

▼ **ONTAP NAS 經濟效益範例**

```
---
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

以下 StorageClass 定義均參照 [虛擬集區的後端範例]。使用 `parameters.selector` 欄位，每個 StorageClass 都會指出哪些虛擬資源池可用於裝載 Volume。Volume 具有所選虛擬資源池中定義的屬性。

- The `protection-gold` StorageClass 對應至 `ontap-nas-flexgroup`後端中的第一個和第二個虛擬資源池。這些是唯一提供黃金層級保護的資源池。

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

- The `protection-not-gold` StorageClass 對應至 `ontap-nas-flexgroup`後端中的第三和第四個虛擬資源池。這些是唯一提供黃金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"
```

- The `app-mysqldb` StorageClass 映射到 `ontap-nas`後端中的第四個虛擬儲存資源池。這是唯一一個為 mysqldb 類型應用程式提供儲存資源池組態的儲存資源池。

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

- The `protection-silver-creditpoints-20k` StorageClass 對應至 `ontap-nas-flexgroup` 後端中的第三個虛擬資源池。這是唯一提供白銀級保護和 20000 信用點的資源池。

```
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"
```

- The `creditpoints-5k` StorageClass 分別對應 `ontap-nas`後端中的第三個虛擬資源池和 `ontap-nas-economy`後端中的第二個虛擬資源池。只有這兩個資源池產品提供 5000 creditpoints。

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

Trident 決定選擇哪個虛擬資源池，並確保符合儲存需求。

更新 `dataLIF` 初始組態之後

您可以在初始設定後變更 dataLIF ，方法是執行下列命令，以更新的 dataLIF 提供新的後端 JSON 檔案。

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

註	如果 PVCS 連接到一個或多個 Pod ，您必須關閉所有對應的 Pod ，然後重新啟動，新的 dataLIF 才會生效。

安全 **SMB** 範例

使用 **ontap-nas** 驅動程式的後端配置

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

使用 **ontap-nas-economy** 驅動程式的後端配置

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

具有儲存池的後端配置

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

採用 **ontap-nas** 驅動程式的儲存類別範例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

註	確保添加 `annotations`啟用安全 SMB。如果沒有註釋，安全 SMB 就無法運作，無論後端或 PVC 中設定了什麼配置。

採用 **ontap-nas-economy** 驅動程式的儲存類別範例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

具有單一 **AD** 使用者的 **PVC** 範例

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

具有多個 **AD** 使用者的 **PVC** 範例

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```

```
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

使用 ONTAP 後端進行均衡放置

您可以在 ONTAP 後端啟用 ONTAP 均衡放置。啟用均衡放置後，ONTAP 會為每個新 Volume 選擇最佳彙總，因此您無需在 Trident 中管理彙總分配。

均衡配置的運作方式

預設情況下，Trident 會探索指派給 SVM 的彙總，將每個彙總視為一個儲存資源池，並在資源配置 Volume 時選擇一個彙總。

啟用均衡佈局後：

- Trident 將整個 SVM 報告為單一儲存資源池，而非報告個別彙總。
- ONTAP 為每個新 Volume 選擇彙總，並在可用的彙總之間平衡 Volume。
- Trident 使用 ONTAP `/application/containers` REST API 建立 Volume。

統一 ONTAP 系統（AFF、FAS 和 ASA）和 NetApp AFX 儲存系統均可使用平衡放置。

需求

- 後端必須使用 ONTAP 9.19.1 或更新版本。

- Trident 必須使用 REST 與 ONTAP 通訊。ZAPI 不支援均衡部署。
- 後端組態不得指定彙總。均衡放置與明確的彙總選擇不能同時使用。

預設值

預設值 `useBalancedPlacement`取決於儲存系統：

儲存系統	預設	附註
統一 ONTAP （AFF、FAS、ASA）	「假」	除非啟用均衡放置，否則 Trident 會選擇彙總。
NetApp AFX （ONTAP 9.19.1 或更新版本）	"真的"	需要平衡配置。將 `useBalancedPlacement` 設定為 `false` 會導致後端驗證失敗。
NetApp AFX （早於 ONTAP 9.19.1）	「假」	無法進行均衡配置。

組態選項

參數	說明	預設
`useBalancedPlacement`	將新 Volume 的彙總選擇委託給 ONTAP。可以在後端組態的根目錄或各個虛擬資源池中設定此選項。	`false`； `true`適用於 ONTAP 9.19.1 或更新版本的 AFX

平衡放置與虛擬資源池

均衡放置策略無論是否使用虛擬池都適用：

- 如果沒有虛擬資源池，Trident 會將 SVM 報告為單一儲存資源池，並使用均衡放置來建立所有新 Volume。
- 對於虛擬資源池，Trident 會像往常一樣報告虛擬資源池並將 Volume 與之相符。如果符合的虛擬資源池啟用了均衡放置，ONTAP 會為該 Volume 選擇彙總。

在後端組態的根目錄下設定 `useBalancedPlacement`，可將其套用至所有虛擬資源池；也可以在單一虛擬資源池上設定，以選擇性地啟用該功能。

範例

以下範例可在統一的 ONTAP 後端上實現均衡放置：

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-balanced-placement
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: cluster-admin
password: <password>
```

```
useBalancedPlacement: true
```

以下範例使用 `TridentBackendConfig`啟用均衡佈局：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: nas-balanced-placement
  managementLIF: 10.0.0.1
  svm: trident_svm
  useBalancedPlacement: true
  credentials:
    name: backend-tbc-ontap-nas-secret
```

以下範例僅在 `gold`虛擬資源池上啟用均衡放置。Trident 繼續為與 `bronze`虛擬資源池相符的 Volume 選擇彙總：

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-virtual-pools
managementLIF: 10.0.0.1
svm: trident_svm
username: cluster-admin
password: <password>
storage:
- labels:
    performance: gold
  useBalancedPlacement: true
- labels:
    performance: bronze
```

後端驗證錯誤

在以下情況下，後端建立或更新會失敗：

組態	結果
`useBalancedPlacement: true` 在 ONTAP 9.19.1 之前的版本中	驗證失敗。請升級至 ONTAP 9.19.1 或更新版本。

組態	結果
`useBalancedPlacement: true`在後端使用 ZAPI	驗證失敗。將後端設定為使用 REST。
`useBalancedPlacement: true`與特定彙總結合	驗證失敗。請移除彙總組態或停用均衡放置。
`useBalancedPlacement: false` 在搭載 ONTAP 9.19.1 或更新版本的 AFX 上	驗證失敗。ONTAP 9.19.1 或更高版本的 AFX 需要均衡放置。

Amazon FSX for NetApp ONTAP 產品

搭配 **Amazon FSX for NetApp ONTAP** 使用 **Trident**

"Amazon FSX for NetApp ONTAP 產品" 是一項完全託管的 AWS 服務，可執行由 NetApp ONTAP 儲存作業系統提供支援的檔案系統。它提供 ONTAP 的功能、效能和管理能力，同時兼具 AWS 的可擴充性和營運簡易性。檔案系統是 Amazon FSx 中的主要資源，類似於內部部署的 ONTAP 叢集。每個檔案系統包含一個或多個儲存虛擬機器（SVM），每個 SVM 包含一個或多個用於儲存檔案和目錄的磁碟區。這種整合使得在 Amazon Elastic Kubernetes Service（EKS）中執行的 Kubernetes 叢集能夠為區塊和檔案工作負載佈建由 ONTAP 支援的持續磁碟區。

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「kubecll」。
- 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"。

> 註 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 僅支援掛載到在 Windows 節點上執行的 Pod 的 SMB 磁碟區。詳情請參閱 "準備配置SMB磁碟區"。
- 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` 。

> 註 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。

同時使用**Trident SAN/iSCSI** 和 **EBS-CSI** 驅動程式

如果您打算將 ontap-san 驅動程式（例如 iSCSI）與 AWS（EKS、ROSA、EC2 或任何其他執行個體）一起使用，則節點上所需的多路徑配置可能會與 Amazon Elastic Block Store (EBS) CSI 驅動程式衝突。為了確保多路徑功能不會幹擾同一節點上的 EBS 磁碟，您需要在多路徑設定中排除 EBS。這個例子展示了 `multipath.conf`包含所需Trident設定的文件，同時將 EBS 磁碟排除在多路徑之外：

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

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。

  警告 | 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"

已測試的 **Amazon Machine** 映像（ **Amis** ）

EKS 叢集支援各種作業系統，但 AWS 已針對容器和 EKS 最佳化某些 Amazon Machine 映像（ Amis ）。以下 AMI 已通過 NetApp Trident 25.02 測試。

Ami	NAS	NAS 經濟效益	iSCSI	iSCSI經濟型
AL2023_x86_64_STANDARD	是的	是的	是的	是的
AL2_x86_64	是的	是的	是*	是*
BOTTLEROCKET_x86_64	是 **	是的	不適用	不適用

AL2023_ARM_64_STANDARD	是的	是的	是的	是的
AL2_ARM_64	是的	是的	是*	是*
BOTTLEROCKET_ARM_64	是 **	是的	不適用	不適用

- * 如果不重新啟動節點，則無法刪除 PV
- ** 不適用於Trident版本 25.02 的 NFSv3。

> 註 如果此處未列出您想要的 AMI ，並不表示不支援，只是表示尚未測試。此清單可作為已知可運行的 AMI 的指南。

- 使用 * 執行的測試：
- EKS版本：1.32
- 安裝方法：Helm 25.06 和 AWS 附加元件 25.06
- 對於 NAS ， NFSv3 和 NFSv4.1 都已經過測試。
- 僅針對 SAN 進行 iSCSI 測試，非 NVMe 型。
- 已執行的測試 * ：
- 建立：儲存類別， PVC ， Pod
- 刪除： Pod ， PVC （一般， qtree /LUN –經濟， NAS 搭配 AWS 備份）

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。

> 註 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 Secrets Manager** 密碼

由於 Trident 將針對 FSX Vserver 發行 API ，以便為您管理儲存設備，因此需要認證才能這麼做。傳遞這些認證的安全方法是透過 AWS Secrets Manager 密碼。因此，如果您還沒有，就必須建立 AWS Secrets Manager 密碼，其中包含 vsadmin 帳戶的認證。

此範例建立 AWS Secrets Manager 密碼來儲存 Trident CSI 認證：

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
```

```
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

建立 IAM 原則

Trident 也需要 AWS 權限才能正確執行。因此，您需要建立一個原則，讓 Trident 擁有所需的權限。

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
Secrets manager"
```

- 政策 JSON 範例 * ：

```
{
  "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"
}
```

為服務帳戶關聯 **(IRSA)** 建立 **Pod Identity** 或 **IAM** 角色

您可以使用 EKS Pod Identity 設定 Kubernetes 服務帳戶，使其代入 AWS Identity and Access Management (IAM) 角色，或使用 IAM 角色進行服務帳戶關聯 (IRSA)。任何已配置為使用該服務帳戶的 Pod 都可以存取該角色有權存取的任何 AWS 服務。

Pod 身份

Amazon EKS Pod Identity 關聯提供了管理應用程式憑證的能力，類似於 Amazon EC2 執行個體設定檔向 Amazon EC2 執行個體提供憑證的方式。

在您的 **EKS** 叢集上安裝 **Pod Identity**：

您可以透過 AWS 主控台或使用下列 AWS CLI 指令建立 Pod 身分：

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

更多資訊請參閱"設定 Amazon EKS Pod Identity Agent" 。

創建 **trust-relationship.json**：

建立 trust-relationship.json 文件，使 EKS 服務主體能夠承擔 Pod Identity 的此角色。然後建立一個具有以下信任策略的角色：

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

trust-relationship.json 文件：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
```

```
}
```

將角色策略附加到 **IAM** 角色：

將上一個步驟中的角色策略附加到已建立的 IAM 角色：

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

建立 **pod** 身分關聯：

在 IAM 角色和 Trident 服務帳戶（trident-controller）之間建立 pod 身分關聯

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

服務帳戶關聯 **(IRSA)** 的 **IAM** 角色
使用 **AWS CLI**：

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

- 信任關係 .json 檔案： *

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
```

```
          }
        }
      }
    ]
}
```

更新檔案中的下列值 `trust-relationship.json`：

- * <account_id> * - 您的 AWS 帳戶 ID
- * <oidc_provider> * - EKS 叢集的 OIDC 。您可以執行下列項目來取得 oidc_provider ：

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

- 使用 IAM 原則附加 IAM 角色 * ：

建立角色後，請使用以下命令將原則（在上述步驟中建立）附加至角色：

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

- 驗證 OICD 提供者是否已關聯 * ：

確認您的 OIDC 供應商與您的叢集相關聯。您可以使用下列命令來驗證：

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

如果輸出為空，請使用下列命令將 IAM OIDC 與叢集建立關聯：

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

如果您使用 **eksctl**，請使用下列範例為 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 <IAM-Policy ARN> --approve
```

安裝Trident

Trident 簡化了 Kubernetes 中 Amazon FSx for NetApp ONTAP 儲存管理，讓您的開發人員和管理員能夠專注於應用程式部署。您可以使用下列方法之一安裝 Trident：

- 掌舵
- EKS 附加元件

如果您想要使用快照功能，請安裝 CSI Snapshot 控制器附加元件。如需詳細資訊、請參閱 "啟用 CSI Volume 的快照功能" 。

透過 helm 安裝 Trident

Pod 身份

1. 新增Trident Helm儲存庫：

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

2. 使用下列範例安裝 Trident：

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

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-
100.2502.0    25.02.0
```

服務帳戶協會 (IRSA)

1. 新增Trident Helm儲存庫：

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

2. 設定「雲端提供者」和「雲端身分」的值：

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

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-
100.2510.0    25.10.0
```

註

如果您打算使用 iSCSI，請確保在用戶端電腦上啟用了 iSCSI。如果您使用的是 AL2023 工作節點作業系統，則可以透過在 helm 安裝中新增 node prep 參數來自動安裝 iSCSI 用戶端：

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace --
set nodePrep={iscsi}
```

透過 **EKS** 附加元件安裝 **Trident**

Trident EKS 附加元件包含最新的安全性修補程式、錯誤修正、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 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 **Trident** 附加元件

管理主控台

1. 開啟 Amazon EKS 主控台： https://console.aws.amazon.com/eks/home#/clusters。
2. 在左側導航窗格中，選擇 **Clusters** 。
3. 選取您要設定 NetApp Trident CSI 附加元件的叢集名稱。
4. 選取 * 附加元件 * ，然後選取 * 取得更多附加元件 * 。
5. 請依照以下步驟選擇附加元件：
   a. 向下捲動至 **AWS Marketplace** 附加元件 部分，然後在搜尋框中輸入 「**Trident**」 。
   b. 選取 NetApp 的 Trident 方塊右上角的核取方塊。
   c. 選擇*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：

   註 *如果您使用 Pod Identity 關聯，請跳過這些步驟。 *

   a. 選擇您要使用的 * 版本 * 。
   b. 如果您使用 IRSA 驗證，請確保設定可選組態設定中可用的設定值：
      - 選擇您要使用的 * 版本 * 。
      - 依照 附加元件配置模式，將 配置值 部分中的 **configurationValues** 參數設定為您在上一個步驟中建立的角色 ARN（值應採用以下格式）：

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

+
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。

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.創建 `**add-on.json**`文件：

對於 **Pod Identity**，請使用以下格式：

註	使用

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

對於 **IRSA** 驗證，請使用以下格式：

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

註	取代 `<role ARN>`前一步驟所建立角色的 ARN 。

*2.安裝 Trident EKS 外掛程式。 *

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

下列範例命令會安裝 Trident EKS 附加元件：

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

更新 **Trident EKS** 附加元件

管理主控台

1. 打開 Amazon EKS 控制檯 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左側導航窗格中，選擇 **Clusters** 。
3. 選取您要更新 NetApp Trident CSI 附加元件的叢集名稱。
4. 選取 * 附加元件 * 索引標籤。
5. 選取 * Trident by NetApp * ，然後選取 * 編輯 * 。
6. 在 * Configure Trident by NetApp * 頁面上、執行下列步驟：
   a. 選擇您要使用的 * 版本 * 。
   b. 展開 * 選用組態設定 * ，並視需要修改。
   c. 選取*儲存變更*。

AWS CLI

下列範例更新 EKS 附加元件：

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl

- 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    v25.6.0-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
v25.6.0-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"。

解除安裝 / 移除 **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 * – NetApp 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。

註	如果附加元件有相關的 IAM 帳戶、則不會移除 IAM 帳戶。

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，選取 * 叢集 * 。
3. 選取您要移除 NetApp Trident CSI 附加元件的叢集名稱。
4. 選擇 **Add-ons** 標籤，然後選擇 Trident by NetApp * 。 *
5. 選擇*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. 選擇*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
```

eksctl

下列命令會解除安裝 Trident EKS 附加元件：

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

此 "Kubernetes StorageClass 物件"可識別資源配置程式，並指示資源配置程式如何配置磁碟區。本節說明如何設定 Kubernetes StorageClass 物件，以指定 Trident 作為資源配置程式。

建立 **StorageClass** 物件

當您為 FSx for ONTAP 建立 StorageClass 時，Trident 將自動建立後端組態。

> 註 如果您想要手動設定儲存後端，請參閱建立一個不帶自動後端配置的 Kubernetes StorageClass 章節，瞭解如何分別建立 Trident 後端和儲存類別。

指定所需的 **StorageClass** 參數

建立 StorageClass 時需要定義以下三個參數：

參數	必要	類型	說明
`fsxFilesystemID`	是的	字串	FSx for NetApp ONTAP 檔案系統 ID
「torageDriverName」	是的	字串	Trident 儲存驅動程式（例如、 `ontap-nas` 或 `ontap-san`）
`credentialsName`	是的	字串	包含 FSx for ONTAP 憑證的 Kubernetes Secret 名稱

指定選用參數

您可以透過 StorageClass 傳遞可選的後端參數。在 StorageClass `parameters` 部分中將所有可選值定義為字串。有關後端參數的完整列表、請參閱："FSx for NetApp ONTAP 後端組態"。

範例 **StorageClass** 設定檔。

以下範例展示 StorageClass 如何觸發自動後端配置。

YAML

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-fsx-demo
  annotations:
    description: "Demo StorageClass for FSx for NetApp ONTAP"
provisioner: csi.trident.netapp.io
parameters:
  fsxFilesystemID: "fs-0abc123"
  storageDriverName: "ontap-nas"
  credentialsName: trident-fsx-credentials
allowVolumeExpansion: true
```

```
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

JSON

```
{
  "apiVersion": "storage.k8s.io/v1",
  "kind": "StorageClass",
  "metadata": {
    "name": "ontap-fsx-demo",
    "annotations": {
      "description": "Demo StorageClass for FSx for NetApp ONTAP"
    }
  },
  "provisioner": "csi.trident.netapp.io",
  "parameters": {
    "fsxFilesystemID": "fs-0abc123",
    "storageDriverName": "ontap-nas",
    "credentialsName": "trident-fsx-credentials"
  },
  "allowVolumeExpansion": true,
  "reclaimPolicy": "Delete",
  "volumeBindingMode": "Immediate"
}
```

建立 **StorageClass**

建立設定檔案後，執行以下命令以建立儲存類別。

```
kubectl create -f storage-class-ontapnas.yaml
```

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
```

套用 StorageClass 後，Trident 會自動建立後端。然後您可以建立 PersistentVolumeClaims 來參照此 StorageClass。

驗證後端組態狀態

Trident 會將後端建立的結果記錄在 StorageClass 註解中。

註釋	說明
`trident.netapp.io/configuratorStatus`	組態結果 (`Success` 或 `Failure`)
`trident.netapp.io/configuratorMessage`	詳細狀態或錯誤訊息
`trident.netapp.io/configuratorName`	內部配置器資源的名稱
`trident.netapp.io/managed`	表示 StorageClass 由 Trident 管理
`trident.netapp.io/additionalStoragePools`	為此後端建立的儲存資源池

若要驗證狀態、請執行：

```
kubectl get storageclass ontap-fsx-demo -o yaml
```

確認 `trident.netapp.io/configuratorStatus` 是否已設定為 `Success`。如果值為 `Failure`，請檢查 `trident.netapp.io/configuratorMessage` 以瞭解錯誤。

新增其他 **FSxN** 檔案系統

如果您在繼續使用相同 StorageClass 的情況下需要額外的儲存容量，請新增額外的 FSxN 檔案系統 ID。

編輯 StorageClass 並新增以下註釋：

```
metadata:
  annotations:
    trident.netapp.io/additionalFsxNFileSystemID: '["fs-
xxxxxxxxxxxxxxxxxx"]'
```

套用變更後， Trident 會更新後端組態並更新 StorageClass 註釋。

操作考量與限制

- 刪除具有自動後端組態的 StorageClass 通常會刪除相關的 Trident 後端。這可能會中斷儲存連線並導致執行中的工作負載中斷。在刪除託管的 StorageClass 之前，請先驗證其影響。
- 僅 AWS FSx for NetApp ONTAP 支援自動後端組態。

建立一個不帶自動後端配置的 **Kubernetes StorageClass**

如果您想單獨建立 Trident 後端和 StorageClass，請依照下列步驟操作。

了解自動後端組態的運作方式

Trident 從 StorageClass 定義中派生後端配置。當您套用 StorageClass 時，Trident 會驗證必要的參數、建立後端，並使用狀態標記 StorageClass。

Trident 只會建立一次 VolumeSnapshotClass。Trident 會為後續的 StorageClasses 重複使用相同的 VolumeSnapshotClass。

建立 **Trident** 後端

要建立 Trident 後端，您需要建立一個 JSON 或 YAML 格式的設定檔案。該檔案需要指定所需的儲存類型（NAS 或 SAN）、檔案系統、要從中取得資料的 SVM 以及如何進行驗證。以下範例展示如何定義基於 NAS 的儲存，以及如何使用 AWS Secret 來儲存要使用的 SVM 的憑證：

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
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:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "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:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

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：

驅動程式名稱	說明
「ONTAP-SAN」	每個已配置的 PV 都是其自身 Amazon FSx for NetApp ONTAP 磁碟區中的一個 LUN。建議用於區塊儲存設備。
「ONTAP-NAS」	每個已配置的 PV 都是一個完整的 Amazon FSx for NetApp ONTAP 磁碟區。推薦用於 NFS 和 SMB。
《ONTAP-san經濟》	每個已配置的 PV 都是一個 LUN，每個 Amazon FSx for NetApp ONTAP 磁碟區可設定 LUN 的數量。
《ONTAP-NANAS經濟》	每個已配置的 PV 都是一個 qtree，每個 Amazon FSx for NetApp ONTAP 磁碟區可配置的 qtree 數量。
「ONTAP-NAA-flexgroup」	每個已配置的 PV 都是一個完整的 Amazon FSx for NetApp ONTAP FlexGroup 磁碟區。

如需驅動程式詳細資料、請參閱 "NAS 驅動程式" 和 "SAN 驅動程式"。

建立後端

建立設定檔案後，執行以下命令來建立和驗證 Trident 後端設定（TBC）：

- 從 yaml 檔案建立 Trident 後端組態（ TBC ），然後執行下列命令：

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- 驗證已成功建立 Trident 後端組態（ TBC ）：

```
Kubectl get tbc -n trident
```

```
NAME                          BACKEND NAME          BACKEND UUID
```

```
PHASE   STATUS

backend-tbc-ontap-nas        tbc-ontap-nas        933e0071-66ce-4324-
b9ff-f96d916ac5e9   Bound   Success
```

如需其他組態選項的詳細資訊，請參閱下方的後端進階組態及範例區段。

在*不使用*自動後端配置的情況下配置儲存類別

以下是與 Trident 和 FSx for ONTAP 一起使用的儲存類別設定範例。

NFS 儲存類別

您可以參考此範例來設定使用 NFS 的磁碟區 StorageClass（有關完整的屬性列表，請參閱下面的 Trident 屬性部分）：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

iSCSI 儲存類別

請使用以下範例設定使用 iSCSI 的磁碟區的 StorageClass：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

使用 **NFSv3** 和 **AWS Bottlerocket** 的儲存類別

若要在 AWS Bottlerocket 上佈建 NFSv3 磁碟區，請將必要的新增 `mountOptions`至儲存類別：

```
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"
mountOptions:
  - nfsvers=3
  - nolock
```

Trident StorageClass 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
後端類型	字串	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、solidfire-san、azure-netapp-files、ontap-san-economy	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
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

屬性	類型	價值	優惠	申請	支援者
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、ONTAP-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 或 FSx for ONTAP 系統不支援此功能

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

配置完成 StorageClass 後，即可在 Kubernetes 中建立它。

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
```

提供 **SMB** 卷

您可以使用 `ontap-nas` 驅動程式來配置 SMB 磁碟區。但是，若要執行此操作，您必須完成以下步驟："準備配置SMB磁碟區"。

後端高階組態和範例

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

參數	說明	範例
「torageDriverName」	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`aws`提供、則`fsxFilesystemID`不需要提供、 `managementLIF`因為 `Trident` 會從 `AWS` 擷取 `SVM` `managementLIF` 資訊。因此、您必須在 SVM 下提供使用者的認證（例如： vsadmin ）、且使用者必須具有該 `vsadmin` 角色。	"10.0.0.1 "，"[2001:1234:abcd::fefe]
「DataLIF」	協定 LIF 的 IP 位址。**ONTAP NAS** 驅動程式：NetApp 建議指定 dataLIF。如果未提供，Trident 將從 SVM 取得 dataLIF。您可以指定一個完全限定網域名稱（FQDN）用於 NFS 掛載操作，從而建立輪詢 DNS 以在多個 dataLIF 之間進行負載平衡。初始設定後可以更改。**ONTAP SAN** 驅動程式：iSCSI 無需指定。Trident 使用 ONTAP 選擇性 LUN 對應來發現建立多路徑會話所需的 iSCSI LIF。如果明確定義了 dataLIF，則會產生警告。如果 Trident 是使用 IPv6 旗標安裝的，可以設定為使用 IPv6 位址。IPv6 位址必須用中括號定義，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。	"["0.0.0/0" ， ":/0"]"

參數	說明	範例
《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CACertifcate」	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。
「toragePrefix」	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`
「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 qtree 和 LUN 的最大磁碟區大小，而且此 `qtreesPerFlexvol`選項可讓您自訂每個 FlexVol volume 的最大 qtree 數量	""（預設不強制執行）
《lunsPerFlexvol》	每個 FlexVol volume 的最大 LUN 數必須在 [50 ， 200] 範圍內。僅限 SAN 。	“100”
「DebugTraceFlags」	疑難排解時要使用的偵錯旗標。例如、 {"api" ： false 、 "method" ： true} 請勿使用 `debugTraceFlags` 除非您正在疑難排解並需要詳細的記錄傾印。	null

參數	說明	範例
「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"	每個 FlexVol volume 的最大 qtree 數必須在範圍 [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
《useREST》	使用ONTAP Isrest API的布林參數。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。	「假」
aws	您可以在 AWS FSX for ONTAP 的組態檔中指定下列項目： - 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	指定要儲存在 AWS Secrets Manager 中的 FSX SVM 認證。- name：機密的 Amazon 資源名稱（ ARN ）、其中包含 SVM 的認證。- type：設為 awsarn。如需詳細資訊、請參閱 "建立 AWS Secrets Manager 密碼" 。	

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 」（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」	為快照保留的磁碟區百分比「 0 」	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`	
「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`。

配置 **PVC**

本節包含如何建立 PersistentVolumeClaim（PVC）使用已設定 Kubernetes StorageClass 來請求 PV 的說明。如果成功，則可以將 PV 掛載到 Pod 中。

建立 **PVC**

`https://kubernetes.io/docs/concepts/storage/persistent-volumes["_PersistentVolume Claim"^]（PVC）是對叢集上 PersistentVolume 的存取請求。PVC 可以配置為請求特定大小的儲存空間或存取模式。透過關聯的 StorageClass，叢集管理員不僅可以控制 PersistentVolume 大小和存取模式，還可以控制效能或服務等級等更多參數。`

建立 Trident 後端和 StorageClass 後，即可建立 PVC。建立 PVC 後，即可將該磁碟區掛載到 pod 中。

範例資訊清單

以下範例展示了 PVC 的基本組態選項。

可存取 **RWX** 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: ontap-gold
```

使用 **iSCSI** 範例的 **PVC**

此範例展示了具有 RWO 存取權限的 iSCSI 基本 PVC，它與名為 `protection-gold`。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
```

```
  resources:
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

建立PVC

步驟

1. 建立 PVC 。

```
kubectl create -f pvc.yaml
```

2. 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"。

部署應用程式

建立儲存類別和 PVC 時，您可以將 PV 掛載到 Pod 。本節列出範例命令和組態，以將 PV 附加至 Pod 。

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
```

註 | 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 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
```

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 pv-pod
```

在 EKS 叢集上設定 Trident EKS 附加元件

NetApp Trident 簡化了 Kubernetes 中適用於 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。NetApp Trident EKS 附加元件包含最新的安全性修補程式，錯誤修正，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

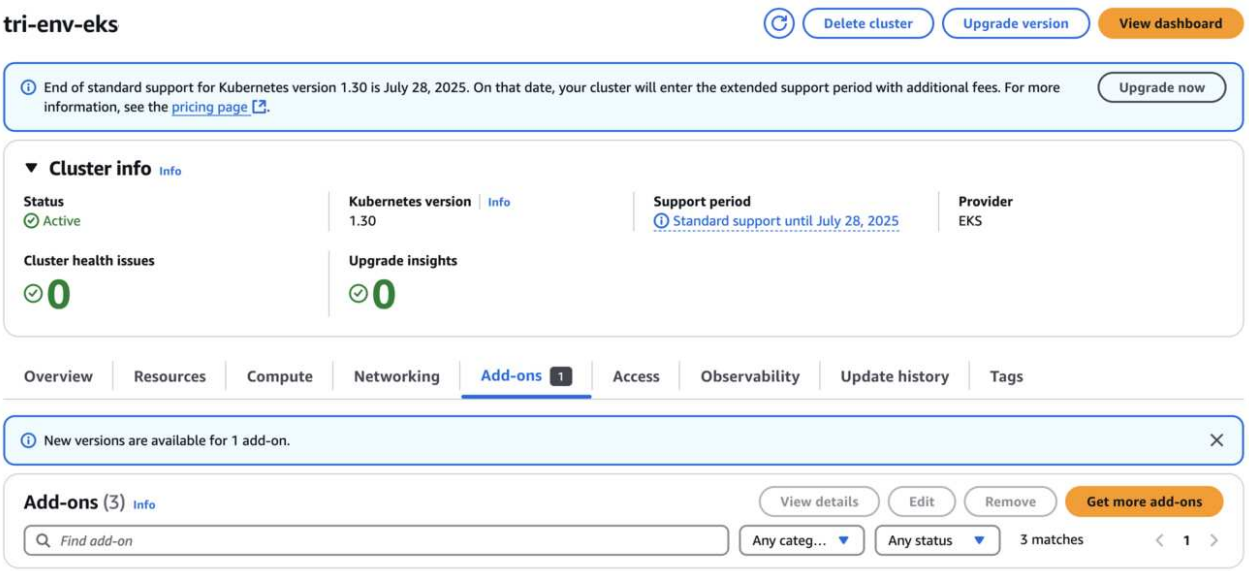
在設定 AWS EKS 的 Trident 附加元件之前、請確定您具有下列項目：

- 具有附加元件使用權限的 Amazon EKS 叢集帳戶。請參閱 "Amazon EKS 附加元件"。

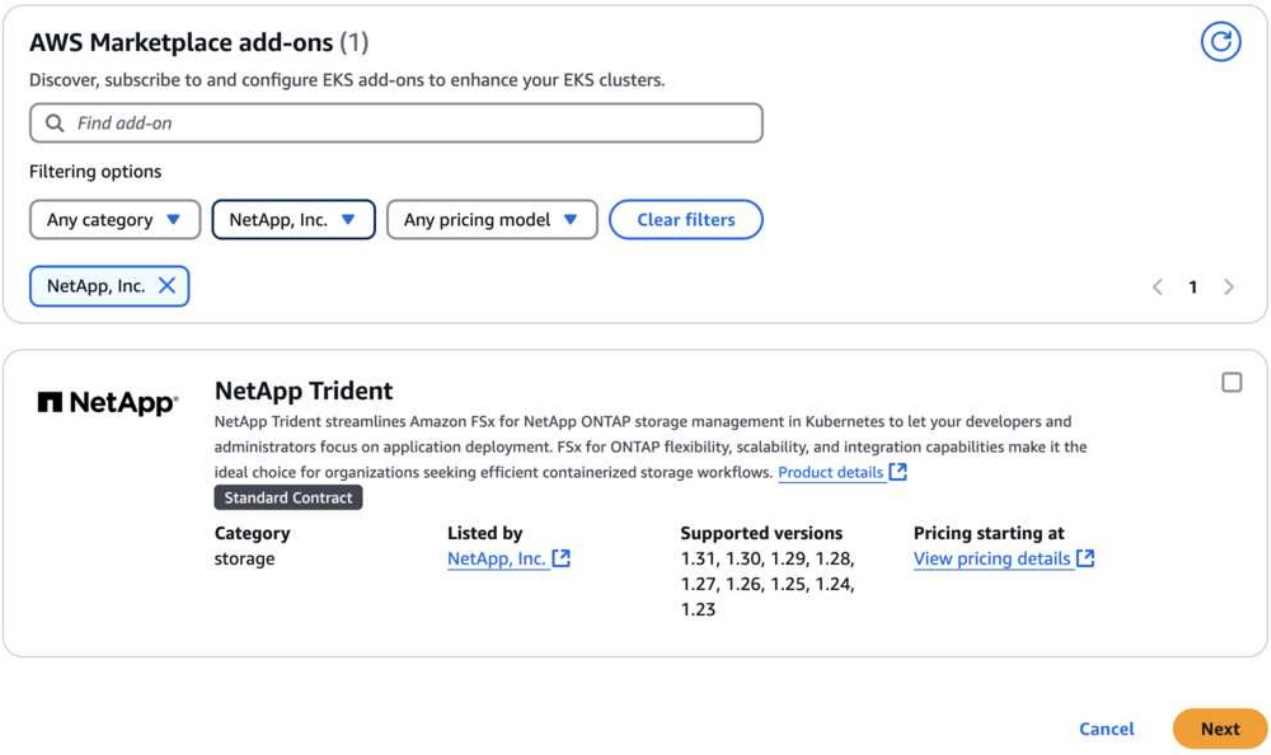
- AWS 對 AWS 市場的權限：
  "aws-marketplace:ViewSubscriptions",
  "aws-marketplace:Subscribe",
  "aws-marketplace:Unsubscribe
- AMI 類型： Amazon Linux 2 （AL2_x86_64 ）或 Amazon Linux 2 ARM （AL2_ARM_64 ）
- 節點類型： AMD 或 ARM
- 現有的 Amazon FSX for NetApp ONTAP 檔案系統

步驟

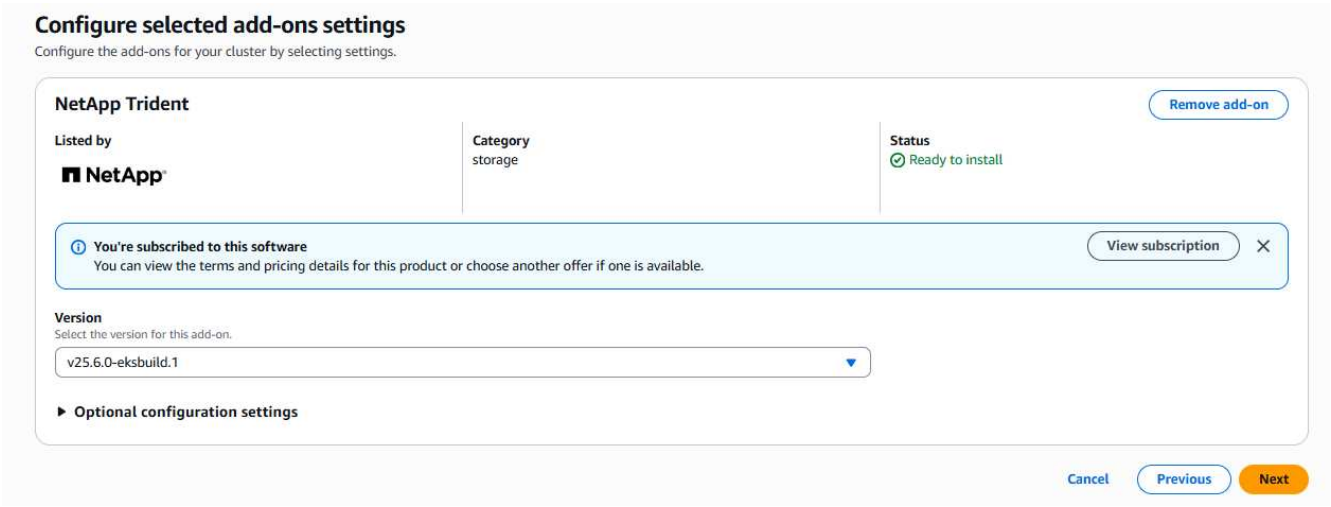
1. 請務必建立 IAM 角色和 AWS 密碼，讓 EKS Pod 能夠存取 AWS 資源。有關說明，請參閱"建立 IAM 角色和 AWS 密碼"。
2. 在 EKS Kubernetes 叢集上，瀏覽至 * 附加元件 * 索引標籤。



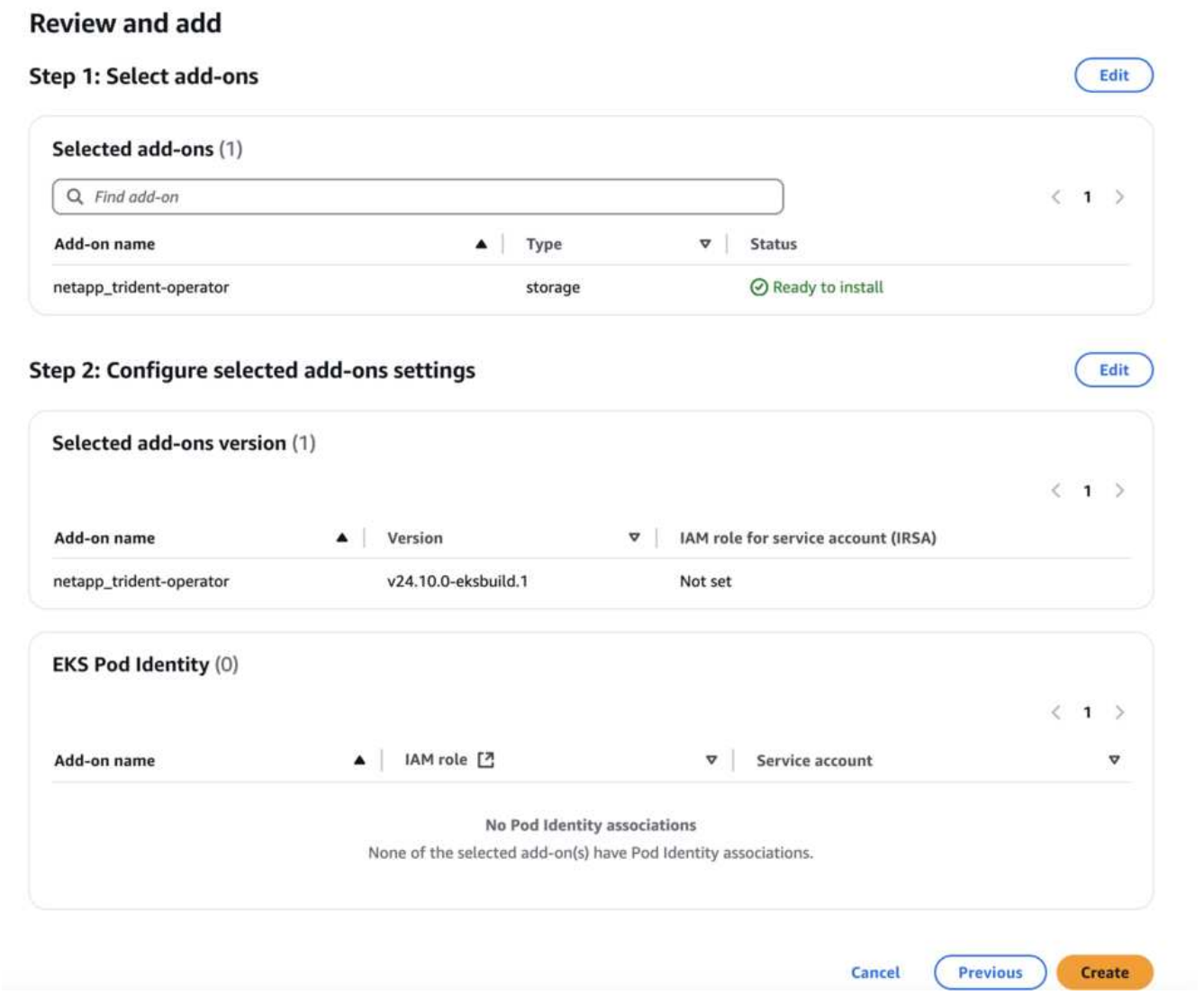
3. 前往 * AWS Marketplace 附加元件 * 並選擇 _storage 類別。



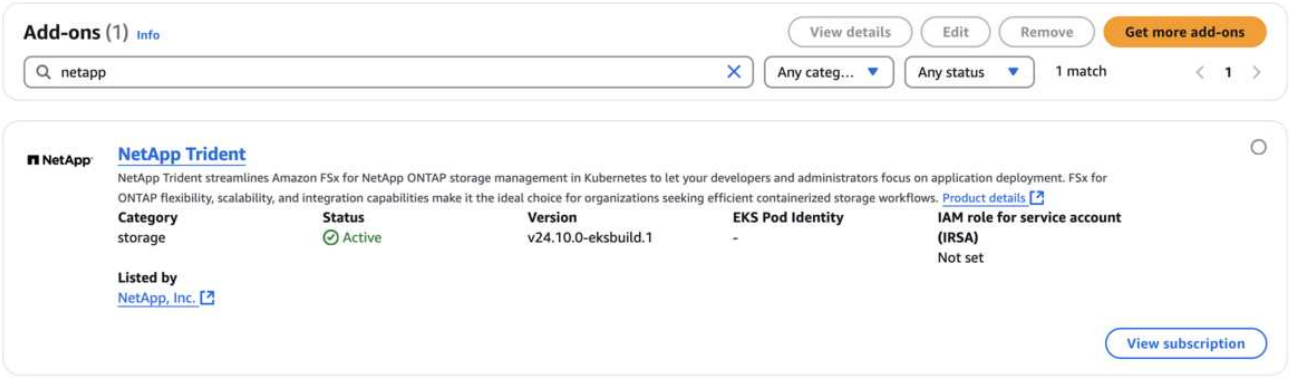
4. 找到 * NetApp Trident * 並選取 Trident 附加元件的核取方塊，然後按一下 * 下一步 * 。
5. 選擇所需版本的附加元件。



6. 配置所需的附加元件設定。



7. 如果您使用 IRSA（服務帳戶的 IAM 角色），請參閱其他設定步驟"請按這裡"。
8. 選擇* Create （建立）。
9. 確認附加元件的狀態為 *Active* 。



10. 執行下列命令，確認叢集上已正確安裝 Trident ：

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

11. 繼續進行設定並配置儲存後端。如需相關資訊、請參閱 "設定儲存後端"。

使用 **CLI** 安裝 / 解除安裝 **Trident EKS** 附加元件

使用 **CLI** 安裝 **NetApp Trident EKS** 附加元件：

以下範例指令安裝Trident EKS 附加元件：

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.6.0-eksbuild.1 （附專用版本）
```

以下範例指令安裝 Trident EKS 外掛程式版本 25.6.1：

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.6.1-eksbuild.1（使用專用版本）
```

以下範例指令安裝 Trident EKS 外掛程式版本 25.6.2：

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.6.2-eksbuild.1（使用專用版本）
```

使用 **CLI** 解除安裝 **NetApp Trident EKS** 附加元件：

下列命令會解除安裝 Trident EKS 附加元件：

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

使用kubecl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。這在內部表示爲 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。

警告

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。

註

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。

提示

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「kubecll」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「kubecl 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: password
```

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
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「」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「kubeclt -n triident描述tbc <tbc-cr-name>」命令、以取得事件記錄的詳細資料。

警告	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

使用 kubectl 管理 Trident 後端

瞭解如何使用「kubecll」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
    ```

註

- 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「kubecl Get tbc <tbc-name>-o yaml -n trident」或「kubecl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** 管理 **Trident** 後端

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
   ```

   註 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。後端將繼續存在於「刪除」狀態。

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。

選擇 **Trident** 後端管理方法

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？

管理 **`tridentctl`** 後端使用 **`TridentBackendConfig`**

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「kubecll」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。

```
kubectl 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
```

註	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 **Trident** 儲存設備類別

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: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
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 Trident 儲存設備類別

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「shorageclass.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。

註 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。

註 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 **Trident Volume**

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

總覽

```
https://kubernetes.io/docs/concepts/storage/persistent-
volumes["_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 PVC 之後，您可以將磁碟區裝入 Pod 。

建立 PVC

步驟

1. 建立 PVC 。

```
kubectl create -f pvc.yaml
```

2. 確認 PVC 狀態。

```
kubectl get pvc
```

```
NAME          STATUS   VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage   Bound    pv-name     1Gi        RWO                    5m
```

1. 將磁碟區裝入 Pod 。

```
kubectl create -f pv-pod.yaml
```

> 註 | 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
```

3. 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 pv-pod
```

範例資訊清單

▼ **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: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
```

```
        - mountPath: "/my/mount/path"
          name: storage
```

基本 **NVMe / TCP** 組態

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

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"。

移動磁碟區

從 Trident 26.06 開始，儲存管理員可以在工作負載持續運行的情況下，將 iSCSI 磁碟區無中斷地移動至相同 ONTAP 叢集內的不同 ONTAP 節點和聚合。您可以透過 `TridentVolumeMove`自訂資源或 `tridentctl`等效方式以宣告式方式要求移動，Trident 會協調 ONTAP 和主機，確保 I/O 不會中斷。

需求

磁碟區移動功能適用於 iSCSI 磁碟區的 `ontap-san`驅動程式。不支援 NVMe 和 NAS 協定。

在移動磁碟區之前，請確保符合下列需求：

- 後端使用 `ontap-san`驅動程式搭配 ONTAP REST API。
- ONTAP 叢集至少有兩個節點。
- 您擁有以下其中一組憑證：
  - 如果使用專用自訂 ONTAP 角色，則需要 ONTAP 9.19.1 或更高版本。
  - 早期 ONTAP 版本中的叢集管理員憑證。

工作原理

Trident 會在同一 ONTAP 叢集內的兩個位置之間移動磁碟區（LUN）。在資料移動之前，Trident 會在目標位置新增新的 iSCSI 路徑，與現有路徑並存，以確保主機多路徑裝置始終保持存取。ONTAP 完成移動後，Trident 會刪除過時的來源路徑。

Trident 透過 `TridentVolumeMove` 資源中可見的一組有序狀態來驅動移動，以便您可以追蹤移動進度。

移動磁碟區

識別磁碟區和位置

1. 找到要移動的 PV 的名稱。PV 名稱也是移動資源的名稱。
2. 從 ONTAP 叢集中，記下該磁碟區目前的（來源）節點和 Aggregate，然後選擇目的地（目標）節點和 Aggregate。

開始移動

兩種方法都可以。兩種方法都會建立相同的 `TridentVolumeMove`資源，資源名稱即為 PV 名稱。

使用 tridentctl

```
tridentctl create volume-move \
  --volume      pvc-8a814d62-bd58-4253-b0d1-82f2885db671 \
  --source-node ontap-node-01 \
  --source-pool aggr1_node01 \
  --target-node ontap-node-02 \
  --target-pool aggr1_node02 \
  -n trident
```

`tvm`是 `volume-move`的別名。新增 `--delete-after-success 10m`可在移動成功後自動刪除資源。

使用 TridentVolumeMove 資源

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeMove
metadata:
  name: pvc-8a814d62-bd58-4253-b0d1-82f2885db671    # the PV to move
  namespace: trident
spec:
  sourceNode: ontap-node-01
  sourcePool: aggr1_node01
  targetNode: ontap-node-02
  targetPool: aggr1_node02
  # deleteAfterSuccess: 10m    # optional
```

應用該資源：

```
kubectl apply -f tridentvolumemove.yaml
```

規格欄位

sourceNode / targetNode

ONTAP 節點名稱（不是 UUID）。

sourcePool / targetPool

ONTAP aggregate 名稱。

deleteAfterSuccess

可選持續時間，例如 `30s`或 `10m`。設定為 `0`可立即刪除資源，省略則可保留資源。

監控移動

使用 `tvm`簡稱或 `tridentvolumemove`追蹤進度：

```
kubectl get tridentvolumemove -n trident
```

若要查看目前狀態和每個節點的進度，請描述資源：

```
kubectl describe tridentvolumemove <pv-name> -n trident
```

在 `Moving`狀態期間，磁碟區可以同時存在多個附件，因為 ONTAP 可能需要一些時間才能完成大型磁碟區的遷移。為了確保無縫遷移，Trident 會在 Trident 控制器和 Trident 節點之間進行協調。每個狀態依序執行以下步驟，最終以 `Succeeded`或 `Failed`結束。

狀態	怎麼了
`Pending`	Trident 會接受並驗證移動請求，並在開始任何儲存變更之前收集所需的資訊。
`ControllerStaging`	Trident 會準備目標位置。它會將目標節點的儲存介面對應到 LUN，並更新 Kubernetes，以便將來節點重新啟動時能夠重新連線至正確的位置。
`NodeStaging`	在每個掛載了該磁碟區的主機上，Trident 會將目標節點的新儲存路徑新增至現有路徑旁邊，以便 pod 的 I/O 可以在兩組路徑上繼續進行。
`Moving`	ONTAP 將磁碟區資料複製並切換到目標 Aggregate。這是耗時最長的階段，整個過程由 ONTAP 獨立完成，Trident 會輪詢直至完成。在此狀態下允許新的發布。
`ControllerUnstaging`	資料移動已完成。Trident 更新 Kubernetes，從所需狀態中刪除來源路徑，然後從 LUN 取消映射來源節點的儲存介面。
`NodeUnstaging`	在每台主機上，Trident 會移除過時的來源路徑。主機上只保留目標路徑，多路徑裝置可繼續不間斷地提供 I/O 服務。

狀態	怎麼了
`Succeeded`	磁碟區已完全移動至目標位置並上線。Trident 會保留 `TridentVolumeMove`資源，或者如果您設定了 `deleteAfterSuccess`，則會自動將其刪除。
`Failed`	不可恢復的錯誤導致遷移操作中止或無法啟動。Trident 重設磁碟區狀態以便恢復正常運作，並保留資源以及錯誤訊息。請查看錯誤訊息，修復根本原因，刪除該資源，然後再試一次。

此 `.status.attachments`清單會報告每個附加節點在其 `state`欄位中的進度：

依附狀態	意義
`Pending`	節點已識別為已連線；工作尚未開始。
`Bridged`	在現有路徑之外新增目標路徑。
`Migrated`	流量現在已透過新路徑提供服務。
`Cleaned`	從節點中移除過時的來源路徑。
`Detached`	節點在移動過程中已分離，並排除在進一步考慮之外，而 Trident 正在評估連接是否已準備好進行下一步。
`Failed`	此節點出現終端錯誤。請查看訊息。

清理移動資源

如果設定了 `deleteAfterSuccess`，Trident 會自動移除資源。否則，您可以手動刪除。在要求對同一磁碟區進行另一次移動之前，請確保該磁碟區不存在任何處於 `TridentVolumeMove`狀態的資源，無論是 `Succeeded`還是 `Failed`狀態。

> 註 Trident 不會自動刪除失敗的移動。Trident 將 `Failed`資源保留為稽核記錄。修復根本原因後，請在重試之前手動刪除該資源。

擴充 Trident Volume

Trident為 Kubernetes 使用者提供了在建立磁碟區後擴充磁碟區的能力。尋找有關擴展 iSCSI、NFS、SMB、NVMe/TCP 和 FC 磁碟區所需的配置的資訊。

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）。

> 註 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「owalumVolume 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。

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl 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。

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl 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：

```
kubectl 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
kubectl 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FC Volume

您可以使用 CSI 資源配置程式來擴充 FC 持續 Volume （ PV ）。

註	驅動程式支援 FC Volume 擴充 `ontap-san`，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「owalumVolume 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。

```
kubectl get pvc
NAME       STATUS    VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc    Bound     pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl 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 FC 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。

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl 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：

```
kubectl 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
kubectl 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支援為已設定的 NFS PV 擴充容量。 `ontap-nas` ， `ontap-nas-economy` ， `ontap-nas-flexgroup` ， 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「ow淺Volume Expansion」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「kubecl 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應該為該 PVC 建立一個 20 MiB 的 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 20 MiB PV 調整為 1 GiB，請編輯 PVC 並設定 `spec.resources.requests.storage`至 1 GiB：

```
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：

```
kubectl get pvc ontapnas20mb
NAME           STATUS    VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

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   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    |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
```

了解 **RWX NVMe** 子系統限制

使用 NVMe 協定的 ReadWriteMany （RWX）磁碟區的擴充性限制為每個磁碟區 64 個節點。以下內容包括這些限制、解釋了所涉及的 NVMe 子系統架構、並概述了所需的解決步驟。

了解 **64** 節點限制

如果您打算將 ReadWriteMany（RWX）磁碟區與 NVMe 協定一起使用，則單一 RWX NVMe 磁碟區不能在 Kubernetes 叢集中被超過 64 個節點掛載。

不要在超過 64 個節點上排程掛載相同 RWX NVMe PersistentVolumeClaim 的工作負載。

此限制僅適用於使用 NVMe 傳輸協定的 RWX Volume。

了解 **NVMe** 子系統模型

每個磁碟區子系統模型（**Trident 26.02** 之前的版本）

在 Trident 26.02 之前的版本中，RWX NVMe 磁碟區採用每個磁碟區子系統模型進行配置。每個 RWX NVMe 磁碟區都對應到 ONTAP 上專屬的 NVMe 子系統。

此模型雖然簡單，但可擴展性限制較低。在大型 Kubernetes 叢集中，子系統控制器限制會很快達到，因為每個 RWX 磁碟區都會佔用一個專用子系統。

超子系統模型（在 **Trident 26.02** 中引入）

從 Trident 26.02 開始，RWX NVMe 磁碟區使用共享的超級子系統模型。多個 RWX NVMe 磁碟區共用同一個 NVMe 子系統。

每個超級子系統最多支援 1024 個命名空間（磁碟區）。此模型顯著提高了 RWX 工作負載的可擴充性，並降低了達到 ONTAP 子系統限制的可能性。

每個 RWX NVMe 磁碟區最多支援 64 個節點。

識別錯誤症狀

如果大規模建立或附加 RWX NVMe Volume，可能會遇到類似以下的錯誤：

```
Maximum number of controllers reached. No more controllers can be created.
```

此錯誤表示已達到 ONTAP NVMe 子系統控制器限制。

解決子系統限制錯誤

若要突破每個磁碟區子系統的限制，並利用超級子系統模型，請升級至 Trident 26.02 或更新版本。

升級 **Trident** 以套用超子系統模型

若要將超子系統模型套用於 RWX NVMe 磁碟區：

1. 將 Trident 升級到 26.02 或更高版本。
2. 將所有使用 RWX NVMe 磁碟區的 Pod 縮減為零個複本。
3. 確認沒有工作負載正在使用 RWX NVMe 磁碟區。
4. 將 Pod 重新擴展。

此重新啟動順序可確保使用超級子系統模型附加 RWX NVMe 磁碟區。

- 此限制僅適用於使用 NVMe 傳輸協定的 RWX Volume。
- 每個 RWX NVMe 磁碟區的 64 節點限制適用。
- 其他存取模式和其他傳輸協定不受影響。

控制器擴充性

Trident 透過提高多個儲存驅動程式之間的並行性來增強控制器的可擴充性。啟用控制器可擴充性後，Trident 控制器會並行處理儲存作業，而非序列化處理，從而提高具有大量並行作業的 Kubernetes 環境中的處理量。

部署前，請確定哪些 Trident 驅動程式在正式發佈時支援控制器可擴充性，哪些驅動程式在 Trident 26.10 中以技術預覽版的形式提供。這有助於您做出明智的部署決策並管理風險。控制器可擴充性預設為停用狀態。

關鍵概念和定義

控制器擴充性

控制器可擴展性是指 Trident 控制器能夠並行處理多個儲存操作，而不是將它們串行化並置於單一鎖定之後。這些操作包括磁碟區的建立、刪除和調整大小；快照的建立和刪除；磁碟區的發布和取消發布；以及後端管理。

啟用控制器可擴充性後，對不同磁碟區和後端執行的操作將同時進行。這可以提高吞吐量，並縮短並發 PersistentVolumeClaim 和 VolumeSnapshot 操作數量較多環境下的端對端操作時間。

預設行為（串列模式）

預設情況下，Trident 控制器一次處理一個作業。每個建立、刪除、調整大小或快照請求完成後，Trident 才會開始處理下一個請求。串行模式是所有安裝的預設支援模式，無需任何組態設定。

對於大多數工作負載而言，串列模式已足夠。僅當串列處理下的操作量造成積壓時，才啟用控制器可擴展性。

控制器擴充性支援

Trident 支援不同成熟度等級的控制器可擴充性，以適應不同的儲存驅動程式。

正式發行

以下驅動程式在 Trident 26.10 正式版中支援控制器擴充性：

- 「ONTAP-SAN」
- 「ONTAP-NAS」
- 《ONTAP-NANAS經濟》
- 「ONTAP-NAA-flexgroup」
- 《ONTAP-san經濟》
- `google-cloud-netapp-volumes`
- 《azure-NetApp-fil形》
- 「olidfire - san」

> 註 The `google-cloud-netapp-volumes`和 `google-cloud-netapp-volumes-san`驅動程式不同。`Trident` 僅支援 `google-cloud-netapp-volumes`。請勿在後端配置或範例中使用 `google-cloud-netapp-volumes-san`。

技術預覽

以下驅動程式在 Trident 26.10 中以技術預覽功能的形式支援控制器可擴充性：

- `asa-r2`（SAN 和 NVMe）

該驅動程式存在以下限制：

- 控制器並發功能僅用於評估和測試。
- 行為可能會因版本而異。
- NetApp 不建議在生產環境中使用。

enableConcurrency 的運作方式

當您將 `enableConcurrency`設定為 `true`時，Trident 會在控制器管理的所有後端套用並行處理。此設定會同時套用於所有後端。無法針對個別後端或個別驅動程式啟用此功能。

每個已配置的後端都必須使用正式版或技術預覽版清單中的驅動程式。如果任何後端使用了不支援的驅動程式，Trident 將無法啟動，且 Trident 不會新增使用不支援驅動程式的後端。

若要恢復啟動，請刪除或重新配置任何使用不支援的驅動程式的後端，或停用控制器可擴充性。

啟用之前

重要	啟用控制器可擴充性之前，請確認所有已設定的後端都使用正式版或技術預覽版清單中的驅動程式。如果任何後端使用了不支援的驅動程式，啟用該功能後 Trident 將無法啟動。

請使用下表決定是否啟用控制器可擴充性。

如果您的環境	然後
僅使用受支援的驅動程式，且控制器作業存在積壓。	啟用控制器可擴充性。
使用任何不受支援的驅動程式	保持預設串列模式。請勿啟用控制器可擴充性。
處理低作業量且無積壓	保持預設序列模式。

啟用控制器擴充性

此 `enableConcurrency`組態選項可控制控制器的可擴充性。您必須在 Trident 安裝期間或更新現有部署時明確啟用此選項。

Trident 操作程式部署

若要使用 Trident operator 啟用控制器擴充性，請在 `TridentOrchestrator` 自訂資源（CR）中將 `enableConcurrency` 設定為 `true`。

新安裝

建立或編輯 `TridentOrchestrator` CR 並將 `enableConcurrency`設定為 `true`：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
```

套用 CR ：

```
kubectl apply -f tridentorchestrator_cr.yaml
```

現有安裝

對現有 `TridentOrchestrator` CR 進行修補，以啟用控制器可擴充性：

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

確認 Trident 是否套用了此設定：

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Helm 部署

若要使用 Helm 啟用控制器可擴充性，請將 `enableConcurrency` 值設為 `true`。

新安裝

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

現有安裝

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

或者，在自訂的 `values.yaml` 檔案中，將 `enableConcurrency` 設定為 `true`：

```
# values.yaml
enableConcurrency: true
```

然後使用值檔案進行安裝或升級：

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace -f values.yaml
```

Tridentctl 部署

若要啟用控制器可擴充性與 `tridentctl`，請在安裝期間傳遞 `--enable-concurrency`標誌。

新安裝

```
tridentctl install -n trident --enable-concurrency
```

現有安裝

若要在現有 `tridentctl`部署中啟用控制器可擴充性，請解除安裝 Trident 並使用下列標誌重新安裝：

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

確認控制器可擴充性已啟用

啟用控制器可擴充性後，請驗證 Trident 控制器是否以並發模式運作。檢查控制器 Pod 日誌：

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

輸出結果包含一個日誌項目，確認並行功能已啟用。

並行行為

啟用控制器可擴充性後，Trident 控制器將採用下列行為：

- Trident 將單一全域鎖替換為細粒度的個別資源鎖定。
- Trident 對修改相同資源的作業進行序列化，以維持資料一致性。
- 僅從資源讀取資料的操作與其他對該資源的讀取操作同時進行。
- Trident 將每個管理 LIF 的並行 ONTAP API 請求限制為 20 個，以防止後端儲存系統過載。
- 如果多個後端共用同一個管理 LIF，則它們共用這 20 次請求的限制。

注意事項和限制

以下幾點適用於 Trident 26.10 中的控制器可擴充性：

- 控制器可擴充性僅支援正式發行清單和技術預覽清單中的驅動程式。如需詳細資訊，請參閱 啟用之前。
- Trident 控制器在內部管理並行作業。此版本不提供使用者可設定的並行限制。
- 整體吞吐量取決於所使用的儲存驅動程式、後端回應速度和 Kubernetes API 伺服器效能。
- 高並發性會增加後端儲存系統的負載。
- 不同驅動程式的控制器可擴展性行為並不相同。
- 技術預覽版驅動程式在高負載下可能表現不穩定，且不同版本之間的行為可能會有所變化。
- 由於並行執行，偵錯並發操作可能更加複雜。指標和日誌可能會顯示交錯的操作輸出。

建議

啟用控制器可擴充性之前，請完成以下步驟：

1. 確認每個已設定的後端都使用正式版或技術預覽版清單中的驅動程式。
2. 在將變更應用到生產環境之前，先在非生產叢集中進行測試。

3. 套用變更後，請確認控制器可擴充性已啟用。

在使用控制器可擴展性時，請遵循以下一般建議：

- 對於需要高可擴展性的生產環境，請使用通用可用性驅動程式。
- 僅在非生產環境中評估技術預覽版驅動程式。
- 大規模運作時，監控後端和控制器的效能。
- 不要在自動化腳本中假定操作順序。

自動磁碟區擴充

自動磁碟區擴充功能可讓 Trident 配置的持續磁碟區能夠在已使用容量達到定義的閾值時自動擴展。此功能可降低營運開銷，並有助於防止因容量耗盡而導致的應用程式中斷。自動磁碟區擴充是使用 Autogrow Policies 實作的。Autogrow Policy 定義：

- 觸發擴充的使用率臨界值
- 磁碟區增長的量
- 磁碟區可達到的最大大小

Trident Autogrow 原則會擴充 PVC。這與 "ONTAP Volume 自動調整規模" 不同，後者會擴充基礎 FlexVol 而不改變 PVC 大小。

註	當超過定義的使用率臨界值時，磁碟區大小會自動增加。磁碟區大小永遠不會自動減少。

需求

在配置自動磁碟區擴充之前、請確保滿足以下要求：

- Trident 26.02 或更新版本
- 基於角色的存取控制權限，用於建立 `TridentAutogrowPolicy` 自訂資源
- StorageClasses 配置 `allowVolumeExpansion: true`
- 支援的 ONTAP 通訊協定：
  - 網路檔案系統 (NFS)
  - 網際網路小型電腦系統介面（iSCSI）
  - 非揮發性記憶體高速介面 （NVMe）
  - 光纖通道協定（FCP）

限制

- ONTAP 9.16.1 之前的 ONTAP Non-Volatile Memory Express 原始區塊磁碟區不支援自動擴充。
- 對於儲存區域網路磁碟區，如果 `growthAmount`小於或等於 `50 mebibytes`，`Trident` 會在調整大小之前自動將該值增加到 `51 mebibytes`，前提是產生的大小不超過 `maxSize`。
- 在棕地環境中，由於磁碟區發佈遷移行為，自動擴充功能可能無法對某些現有磁碟區生效。

- 當磁碟區達到 `maxSize`時，就不會再發生擴充。
- 支援自動磁碟區擴充的通訊協定：
  - 網路檔案系統 (NFS)
  - 網際網路小型電腦系統介面（iSCSI）
  - 非揮發性記憶體高速介面 （NVMe）
  - 光纖通道協定（FCP）

使用自動增長策略配置磁碟區

Autogrow Policy 可以在兩個層級進行設定：

- 儲存類別層級：設定所有磁碟區的預設值（使用註釋）
- PVC 等級：覆蓋儲存類別預設值（使用註解）

建立 **Autogrow** 策略

自動增長策略可在磁碟區達到定義的容量臨界值時自動擴展磁碟區。

請確保您已具備：

- 已安裝 Trident 26.02 或更新版本
- 基於角色的存取控制權限，用於建立 `TridentAutogrowPolicy` 資源
- 了解工作負載成長需求

Autogrow Policy 定義了當磁碟區達到定義的容量臨界值時如何自動擴充。

您可以在工作流程中的任何階段建立 Autogrow 策略：

- 在創建 StorageClasses 和磁碟區之前
- StorageClasses 存在之後
- 配置磁碟區之後

這種靈活性使您能夠在不重新建立現有資源的情況下引入自動擴充。

自動增長策略規格

Autogrow Policies 是 Kubernetes 自訂資源，定義如下：

欄位	說明	格式	必要	範例	預設
名稱	唯一原則識別碼	字串	是的	production-db-policy	沒有任何
usedThreshold	觸發擴充的容量百分比	百分比字串	是的	"80%"	沒有任何
growthAmount	達到臨界值時的增長量	百分比或大小	否	"10%" 或 "5Gi"	"10%"

欄位	說明	格式	必要	範例	預設
maxSize	最大磁碟區大小限制	Kubernetes 數量	否	"500Gi"	無限

建立 **Autogrow** 策略

步驟

1. 建立定義 Autogrow Policy 的 YAML 檔案：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. 將原則套用至您的叢集：

```
kubectl apply -f autogrow-policy.yaml
```

3. 確認策略已建立：

```
kubectl get tridentautogrowpolicy standard-autogrow
```

預期輸出

```
NAME                 USED THRESHOLD   GROWTH AMOUNT   STATE
standard-autogrow    80%              10%             Success
```

政策狀態

建立原則後、Trident 會驗證規格並指派下列其中一種狀態：

狀態	說明	需要採取行動
成功	策略已驗證，可以投入使用。	無。
失敗	偵測到驗證錯誤。	檢閱並修正規格。
刪除	刪除作業正在進行中。	等待完成。

將政策與 StorageClass 建立關聯

您可以使用 `trident.netapp.io/autogrowPolicy` 註解將 Autogrow Policy 與 StorageClass 關聯起來。所有從該 StorageClass 配置的磁碟區都將繼承該策略。

> 註 StorageClass 必須具有 `allowVolumeExpansion: true`。

步驟

1. 建立或修改帶有 Autogrow Policy 註解的 StorageClass：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. 應用 StorageClass：

```
kubectl apply -f storageclass.yaml
```

3. 驗證註釋：

```
kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

預期輸出

```
production-db-policy
```

政策優先順序

當 StorageClass 和 PVC 上都設定了自動成長策略註解時，Trident 應用以下優先規則：

1. *PVC 註釋優先。*如果 PVC 設定 `trident.netapp.io/autogrowPolicy`，則始終使用該值。
2. **StorageClass** 註解僅適用於 **PVC** 沒有註解的情況。
3. 如果兩者都沒有註解、 則不會套用 Autogrow Policy 。

StorageClass 註解	PVC 註釋	有效行為
`trident.netapp.io/autogrowPolicy: standard-agp`	未設定	用途 `standard-agp`。
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: logs-policy`	使用 `logs-policy`（PVC 覆寫 StorageClass）。
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: "none"`	無自動增長策略（PVC 停用自動增長）。
未設定	`trident.netapp.io/autogrowPolicy: dev-policy`	用途 `dev-policy`。
未設定	未設定	無自動成長政策。

組態範例

以下範例展示了不同使用情境下的常見 Autogrow Policy 配置。

生產資料庫的保守原則

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

具有固定增長增量的日誌儲存

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

包含預設設定的最小原則

當您省略 `growthAmount`和 `maxSize`時，Trident 使用預設的(`10%`成長、無限大小）：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

具有自訂 **maxSize** 和預設 **growthAmount** 的策略

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

無限制的積極成長 **maxSize**

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

包含小數百分比的原則

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```

註 | 支援小數百分比。若您指定超過三位小數，Trident 會將數值四捨五入至小數點後三位。

NAS StorageClass 具備自動成長功能

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

具有自動增長功能的 SAN StorageClass

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

設定 ONTAP Volume 自動調整規模

設定 Trident 以在支援持續性 Volume 的 FlexVol Volume 上啟用 ONTAP Volume 自動調整規模。當已用空間達到設定的臨界值時， ONTAP 會自動將 FlexVol 擴充至最大大小。這可為快照和儲存負荷提供額外的容量，而無需手動調整 FlexVol 的大小。

ONTAP Volume 自動調整規模僅會增加後端 FlexVol；它不會變更 PersistentVolumeClaim（PVC）大小。這與 "Trident Autogrow 原則" 不同，後者會擴充 PVC。

Trident 在建立、複製或調整 Volume 以及匯入其管理的 Volume 時，會套用後端自動調整規模設定。

需求

- Trident 26.10 或更高版本。

- 下列驅動程式之一：
  - `ontap-san` 可使用 iSCSI、NVMe 或 Fibre Channel 傳輸協定（FCP）。
  - `ontap-san-economy` 使用 iSCSI。
- 本地部署 ONTAP：
  - 對於使用 ONTAP REST API 的後端，需要 ONTAP 9.12.1 或更新版本。NVMe 後端必須使用 ONTAP REST API。
  - 對於使用 ONTAPI (ZAPI) 的後端，任何 Trident 支援的 ONTAP 版本均可。請參閱"Trident 需求"。

限制

- ONTAP Volume 自動調整規模不會縮小 FlexVols。
- 自動調整規模是按後端設定的。建立分隔的後端，以便針對不同的工作負載使用不同的設定。
- ONTAP Volume 自動調整規模在 ONTAP NAS 驅動程式、ASA r2 系統、NetApp AFX 或其他解耦式 ONTAP 系統，以及 Google Cloud NetApp Volumes 上不受支援。這些組態的後端建立會失敗。
- ONTAP Volume 自動調整規模不支援 TridentMirrorRelationship 複寫與 MetroCluster 組態。Trident 不會阻止這些組態。
- 對於 `ontap-san-economy`， `maximumSize`必須是百分比。
- 對於 `ontap-san`，絕對的 `maximumSize`可作為大小限制。如果後端 FlexVol 必須超過該限制，則建立或擴充 PVC 將會失敗。
- 一個 `ontap-san-economy` FlexVol 可以包含多個 PVC 的 LUN。每次 Volume 操作都會將目前後端的自動調整規模設定套用至整個 FlexVol。如果多個後端共享 FlexVols，則最近一次操作會決定其設定。對於 `ontap-san-economy`，只需設定 `mode: grow`並省略 `growThresholdPercent`和 `maximumSize`，以便所有後端使用相同的預設值。

ONTAP Volume 自動調整規模選項

在後端組態的頂層 `ontapAutosize`區塊中設定這些選項。

選項	說明	格式	必要	當 `ontapAutosize` 存在時的預設值	範例
`mode`	啟用或停用自動調整規模。 `grow`啟用自動調整規模； `off`停用自動調整規模。	字串	否	`off`	`grow` 或 `off`
`growThresholdPercent`	ONTAP 擴充 FlexVol 的已使用空間百分比。有效值為 2 到 99。	整數	否	`85`	`90`
`maximumSize`	最大 FlexVol 大小，以其大小的百分比或絕對大小表示。僅針對 `ontap-san`支援絕對大小。	百分比或 Kubernetes 數量	否	`"20%"`	`"20%"` 或 `"2Ti"`

當 `ontapAutosize` 區塊存在時，會套用預設值。如果省略該區塊，Trident 不會變更 FlexVols 上的自動調整規模設定。

Trident 會將百分比 `maximumSize`轉換為絕對大小，並在每次調整 `Volume` 大小時重新計算。例如，使用 `maximumSize: "20%"`，100 GiB 的 FlexVol 可以成長到 120 GiB。

設定 **ONTAP Volume** 自動調整規模

1. 將 `ontapAutosize` 區塊新增至您的後端組態：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  backendName: ontap-san-autosize
  managementLIF: 10.0.0.1
  svm: svm_san
  sanType: iscsi
  ontapAutosize:
    mode: grow
    growThresholdPercent: 85
    maximumSize: "20%"
  credentials:
    name: backend-tbc-ontap-san-secret
```

2. 建立後端：

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

3. 從此後端配置 Volume 後，驗證套用於其 Trident Volume 的自動調整規模設定：

```
kubectl get tridentvolume <pv-name> -n trident \
  -o jsonpath='{.config.ontapAutosizeMode}
{.config.ontapAutosizeGrowThresholdPercent}
{.config.ontapAutosizeMaximumSize}'
```

對於上述組態，預期輸出為：

```
grow 85 20%
```

在後端更新自動調整規模設定

您可以透過更新後端來新增、變更或移除 `ontapAutosize`區塊。變更不會立即更新現有的 FlexVols：

- 對於 `ontap-san`，新的 Volume、複製和代管匯入會使用新設定。現有 Volume 會保留其目前的設定，包括在調整大小時。
- 對於 `ontap-san-economy`，共享的 FlexVol 會在下次 Trident 於其中建立、調整大小、複製或匯入 Volume 時接收新設定。

組態範例

`ontap-san`使用 FCP 和絕對最大大小

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_san",
  "username": "vsadmin",
  "password": "<password>",
  "sanType": "fcp",
  "ontapAutosize": {
    "mode": "grow",
    "growThresholdPercent": 90,
    "maximumSize": "2Ti"
  }
}
```

《ONTAP-san經濟》

此範例僅設定 `mode`，因此 Trident 會針對 `growThresholdPercent`和 `maximumSize`使用預設值：

```
{
  "version": 1,
  "storageDriverName": "ontap-san-economy",
  "managementLIF": "10.0.0.1",
  "svm": "svm_san",
  "username": "vsadmin",
  "password": "<password>",
  "ontapAutosize": {
    "mode": "grow"
  }
}
```

常見問題集

我能否將 **ONTAP Volume** 自動調整規模與 **Trident Autogrow** 原則結合使用？

是的。Trident Autogrow 原則會擴充 PVC，且 Trident 會在每次調整大小期間重新計算 ONTAP 自動調整規模最大值。如果您將絕對 `maximumSize`與 `ontap-san`搭配使用，請將原則的 `maxSize`設定得夠低，以將後端 FlexVol 保持在該限制內。FlexVol 大於 PVC，因為 Trident 會增加快照保留和中繼資料的空間。

省略 `**ontapAutosize**`和設定 `**mode: off**`之間有什麼差異？

如果省略 `ontapAutosize`，Trident 不會變更 ONTAP FlexVols 上的自動調整規模設定。如果設定 `mode: off`，Trident 會在它建立或更新的 FlexVols 上停用自動調整規模，包括從其來源繼承自動調整規模設定的複本。

為什麼我的 **PVC** 擴充失敗並出現 `**ontapAutosize maximum … is below required FlexVol size**`錯誤？

該 Volume 使用絕對 `maximumSize`，而擴充的 PVC 需要更大的 FlexVol。 `ontap-san` Volume 會保留其建立時的最大大小，因此更新後端不會變更現有 Volume 的設定。對於新 Volume，請增加 `maximumSize`或使用百分比。

為什麼建立 **Volume** 失敗並出現自動調整規模錯誤？

ONTAP 拒絕了自動調整規模設定，PVC 事件中包含來自 ONTAP 的原因。例如，最大大小可能超過 ONTAP 系統支援的最大 FlexVol 大小。降低後端的 `maximumSize`；Trident 下次嘗試建立 Volume 時會使用更新後的值。

相關資訊

- "使用 Trident Autogrow 原則自動擴充 Volume"
- "ONTAP 命令參考：Volume 自動調整規模"

管理 **Autogrow** 策略

建立 Autogrow Policies 後，您可以檢視、更新和刪除它們。您還可以監控哪些磁碟區使用了特定的原則。

檢視自動成長原則

列出所有原則

使用 `kubectl`列出叢集中的所有 Autogrow 原則：

```
kubectl get tridentautogrowpolicy
```

或者，使用 `tridentctl`：

```
tridentctl get autogrowpolicy
```

檢視原則詳細資料

若要檢視原則的完整規格和狀態：

```
kubectl describe tridentautogrowpolicy production-db-policy
```

若要以 YAML 格式檢視策略及其相關磁碟區：

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

更新 **Autogrow** 策略

您可以修改現有原則，以變更其臨界值、成長量或大小上限。變更會立即對所有使用該原則的磁碟區生效。

> 重要　變更會影響所有使用該策略的磁碟區。如果可能，請先在非生產環境中測試變更。

步驟

1. 編輯原則：

   ```
   kubectl edit tridentautogrowpolicy production-db-policy
   ```

2. 根據需要修改 `spec`欄位：

   ```
   spec:
     usedThreshold: "75%"     # Changed from 80%
     growthAmount: "20%"      # Changed from 10%
     maxSize: "1Ti"           # Changed from 500Gi
   ```

3. 儲存並退出。變更會立即生效。

更新考量事項

- *立即生效：*所有使用此原則的磁碟區將在下次成長評估時採用新參數。
- *無需重新啟動磁碟區：*變更將套用於下一次成長作業。
- 先測試： 盡可能在非生產環境中驗證變更。
- 溝通變更： 當您修改共享策略時，請通知團隊。

刪除 **Autogrow** 策略

自動增長原則使用終結器保護，以防止磁碟區在積極使用時意外刪除。

步驟

1. 刪除原則：

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. 如果磁碟區仍然使用該原則，則刪除作業將進入 `Deleting`狀態。請檢查哪些磁碟區受到影響：

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. 從每個受影響的磁碟區中移除該原則。請選擇下列其中一個選項：

   - 選項 **A**：明確停用自動增長，方法是將註解設定為 `"none"`：

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

   - 選項 **B**：完全移除註解：

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

刪除行為

情境	行為
沒有磁碟區使用此原則	原則會立即刪除。
磁碟區使用此原則	策略進入 `Deleting` 狀態。終結器會阻止完成，直到所有磁碟區都被移除。
所有磁碟區均已從該原則中移除	最終處理程序被移除，該政策被刪除。

監控 **Autogrow Policy** 使用情況

使用原則檢查磁碟區

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

尋找磁碟區使用的原則

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

監控原則事件

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

支援的通訊協定

Autogrow 支援以下儲存傳輸協定：

- NFS
- iSCSI
- FCP
- NVMe

> 註 對於 SAN 磁碟區，如果設定的 `growthAmount`為 `50 MiB` 或更小，`Trident` 會自動將調整大小作業的成長量增加到 `51 MB`，只要產生的大小不超過 `maxSize`。

支援的後端

Autogrow 支援 ONTAP 後端。Autogrow 也支援 Google Cloud NetApp Volumes （GCNV） 的 NAS 和 SAN 磁碟區。

已知限制

- *ONTAP NVMe 原始區塊磁碟區：*使用 ONTAP 9.16.1 之前版本建立的磁碟區不支援自動成長。
- *現有磁碟區（棕地部署）：*即使應用了有效的自動成長策略，自動成長功能也可能無法對現有磁碟區生效。此限制是由於磁碟區發布正在進行遷移所致。若要確認遷移已完成，請檢查 Trident 控制器日誌中的 `"Migration completed"`訊息。

常見問題集

Trident 何時評估閾值？

Trident 持續監控磁碟區使用量。當已使用容量超過 `usedThreshold`時，Trident 會建立內部調整大小請求，並依據設定的 `growthAmount`擴充磁碟區。

例如，此原則會在容量達到 80% 時觸發擴充，每次將磁碟區增加 10%，最多可達 500 GiB：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

磁碟區已配置完畢後，我還能套用原則嗎？

是的。您可以隨時建立 Autogrow Policy，並透過新增或更新 `trident.netapp.io/autogrowPolicy`註解將其應用於現有的 PVC。您無需重新建立 PVC 或 StorageClass。

將原則套用至現有的 PVC：

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

將原則套用至現有的 StorageClass：

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

如果我在 **StorageClass** 和 **PVC** 上都設定了自動成長策略會發生什麼事？

PVC 註解始終優先。如果 PVC 具有 `trident.netapp.io/autogrowPolicy`註解，Trident 將使用該值，而不管 StorageClass 中指定了什麼值。詳情請參閱 "政策優先順序"。

例如，假設有以下 StorageClass：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

而這個 PVC 覆寫了 StorageClass 原則：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
```

```
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident 使用 `logs-policy`for `database-pvc，不是 standard-agp。

如何停用特定磁碟區的自動增長？

將 PVC 註釋設定為 "none"。此設定會覆蓋該磁碟區的任何 StorageClass 層級原則：

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

您可以確認自動增長功能已停用：

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

預期輸出

```
none
```

當磁碟區達到 **maxSize** 時會發生什麼事？

Trident 停止擴充磁碟區。即使使用量持續增加超過 usedThreshold，它也不會再為該磁碟區建立任何調整大小的請求。

例如，根據此策略，Trident 在容量達到 100 GiB 後將停止成長：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

若要允許無限增長，請省略 maxSize 或將其設為 0：

```
apiVersion: trident.netapp.io/v1
```

```
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

我可以在不重新啟動磁碟區的情況下變更原則嗎？

是的。更新原則後，所有使用此原則的磁碟區都會在下次成長評估時採用新參數。無需重新啟動磁碟區。

若要就地更新原則：

```
kubectl edit tridentautogrowpolicy production-db-policy
```

根據需要修改欄位：

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

儲存並退出。驗證更新的原則：

```
kubectl get tridentautogrowpolicy production-db-policy
```

預期輸出

```
NAME                   USED THRESHOLD   GROWTH AMOUNT   STATE
production-db-policy   75%              20%             Success
```

為什麼我的原則處於失敗狀態？

`Failed` 狀態表示原則規格包含驗證錯誤。執行下列命令以檢視錯誤詳細資料：

```
kubectl describe tridentautogrowpolicy <policy-name>
```

常見原因包括無效的 `usedThreshold`（必須為 1-99%）、 `growthAmount`超過 `maxSize`或無效的 Kubernetes 數量格式。請更正規格並重新應用：

```
kubectl apply -f autogrow-policy.yaml
```

為什麼我無法刪除原則？

策略使用終結器保護。如果磁碟區仍然使用該策略，刪除操作將進入 `Deleting`狀態，並等待您將所有磁碟區從該策略中移除。

識別受影響的磁碟區：

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

然後從每個 PVC 中移除註解：

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

移除所有磁碟區後，Trident 會釋放終結器並刪除政策。

自動增長功能支援哪些後端？

Autogrow 支援 ONTAP 後端。也支援 Google Cloud NetApp Volumes （GCNV） 的 NAS 和 SAN 磁碟區。

自動成長功能支援 NFS、iSCSI、FCP 和 NVMe 協定。NVMe 原始區塊磁碟區需要 ONTAP 9.16.1 或更新版本。

對於現有部署環境中的磁碟區，可能需要先完成磁碟區發布移轉，然後自動增長功能才能生效。若要驗證移轉狀態，請檢查 Trident 控制器日誌：

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

以下 StorageClass 範例展示了為 NAS 和 SAN 後端配置的自動成長功能：

NAS 後端

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN 後端

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN 磁碟區的最小成長量是多少？

對於 SAN 磁碟區，有效最小增長量為 51 MB。如果您配置的 `growthAmount`為 50 MiB 或更少，Trident 會自動將調整大小作業的增長量增加到 51 MB。

例如，此原則設定了 `growthAmount`為 `"40Mi"`，但 Trident 會對使用該原則的任何 SAN 磁碟區套用 51 MB 的成長：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"
```

為避免自動調整，請將 `growthAmount` 設定為大於 50 MiB 的值：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
```

```
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"
```

匯入 **Trident Volume**

您可以使用 `tridentctl import`或透過使用 Trident 匯入註解建立持久性磁碟區宣告（PVC），將現有儲存磁碟區匯入為 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。

警告	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策略為 `delete`，則當 PV 被刪除時，Trident 會刪除儲存磁碟區。
- 預設情況下，Trident 會管理 PVC，並在後端重新命名 FlexVol volume 和 LUN 或 qtree。您可以傳遞 `--no-manage`旗標來匯入非託管磁碟區，以及傳遞 `--no-rename`旗標來保留磁碟區名稱。
  - `--no-manage`* - 如果您使用 `--no-manage`標誌表明， Trident在物件的生命週期內不會對 PVC 或 PV 執行任何額外的操作。刪除 PV 時，儲存磁碟區不會被刪除，其他操作（如磁碟區複製和磁碟區調整大小）也會被忽略。

- `--no-rename*` - 如果使用 `--no-rename`標誌，`Trident` 將在匯入磁碟區時保留現有磁碟區名稱，並管理磁碟區的生命週期。此選項僅支援 `ontap-nas`、 `ontap-san`（包括 ASA r2 系統）、 `ontap-san-economy`和 `ontap-nas-economy`（qtree 匯入）驅動程式。

  提示 | 如果您想使用 Kubernetes 進行容器化工作負載，但又想在 Kubernetes 之外管理儲存磁碟區的生命週期，那麼這些選項非常有用。

- 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` 或透過建立具有 Trident 匯入註解的 PVC 來匯入磁碟區。

註 | 如果使用 PVC 註釋，則無需下載或使用 `tridentctl`匯入磁碟區。

使用 **tridentctl**

步驟

1. 建立 PVC 檔案（例如 `pvc.yaml`），用於建立 PVC。PVC 檔案應包含 `name`、 `namespace`、 `accessModes`和 `storageClassName`。您也可以在 PVC 定義中指定 `unixPermissions`。

   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
   ```

   註 | 僅包含必需參數。其他參數（例如 PV 名稱或磁碟區大小）可能會導致匯入命令失敗。

2. 使用 `tridentctl import`用於指定包含磁碟區的Trident後端名稱以及唯一標識儲存上磁碟區的名稱的命令（例如： ONTAP FlexVol、Element Volume）。這 `-f`需要提供參數來指定PVC檔案的路徑。

   ```
   tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
   ```

使用 **PVC** 註釋

步驟

1. 建立一個 PVC YAML 檔案（例如， `pvc.yaml`），其中包含所需的 Trident 匯入註解。PVC 檔案應包

含：

- `name` 和 `namespace` 在中繼資料中
- `accessModes`、 `resources.requests.storage` 和 `storageClassName` 在規格中
- 註釋：
  - `trident.netapp.io/importOriginalName`：後端的磁碟區名稱
  - `trident.netapp.io/importBackendUUID`：磁碟區所在的後端 UUID
  - `trident.netapp.io/notManaged`（可選）：設定為 `"true"`表示非託管磁碟區。預設值為 `"false"`。

  以下是匯入託管磁碟區的範例規格：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. 將 PVC YAML 檔案套用到您的 Kubernetes 叢集：

```
kubectl apply -f <pvc-file>.yaml
```

Trident 會自動匯入磁碟區並將其繫結至 PVC。

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`。

註 • 若要使用 `ontap-nas-economy`驅動程式匯入 qtree，請參閱匯入 NFS qtree。當 Trident 匯入

qtree 時，也會將父 FlexVol 作為包含該 qtree 的儲存桶匯入。qtree 必須單獨匯入。

- 。 `ontap-nas` 和 `ontap-nas-flexgroup` 驅動程式不允許重複的磁碟區名稱。

使用驅動程式建立的每個磁碟區都 `ontap-nas`是 `ONTAP` 叢集上的 `FlexVol volume` 。使用驅動程式匯入 `FlexVol` 磁碟區 `ontap-nas`的運作方式相同。`ONTAP` 叢集上已存在的 `FlexVol Volume` 可匯入為 `ontap-nas` PVC 。同樣地、 FlexGroup Vols 也可以匯入為 `ontap-nas-flexgroup` PVCS 。

使用 **tridentctl** 的 **ONTAP NAS** 範例

以下範例展示如何使用 `tridentctl`匯入託管磁碟區和非託管磁碟區。

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>

+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | 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

+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+------------------------------------------+---------+---------------
```

```
+----------+----------------------------------------+--------+---------+
```

使用 **PVC** 註解的 **ONTAP NAS** 範例

以下範例展示如何使用 PVC 註解匯入託管和非託管磁碟區。

託管 **Volume**

以下範例從後端 `81abcb27-ea63-49bb-b606-0a5315ac5f21`匯入一個名為 `ontap_volume1`的 1GiB `ontap-nas` 磁碟區，並使用 PVC 註解設定了 RWO 存取模式：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

非託管 **Volume**

以下範例從後端 `34abcb27-ea63-49bb-b606-0a5315ac5f34`匯入名為 `ontap-volume2`的 1Gi `ontap-nas` 磁碟區，並使用 PVC 註解設定 RWO 存取模式：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
```

```
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

導入 qtrees

註	Qtree 導入是 Trident 26.10 中的技術預覽功能。技術預覽功能不支援在生產環境中使用。

您可以使用 `ontap-nas-economy` 驅動程式將現有的 ONTAP qtree 匯入 Trident。請以 `flexvol/qtree` 格式的 FlexVol Volume 和 qtree 名稱，以及包含該 qtree 的後端 UUID 來識別 qtree。[總覽與考量] 中的一般匯入考量也適用於 qtree 匯入。

使用 tridentctl 導入 qtree

以 `flexvol/qtree` 格式將 qtree 指定為磁碟區名稱。以下範例會從後端 `ontap_nas_economy` 上的 FlexVol volume `flexvol1` 匯入 qtree `qtree1`：

```
tridentctl import volume ontap_nas_economy flexvol1/qtree1 -f <path-to-pvc-file>
```

若要匯入 qtree 而不重新命名，請新增 `--no-rename`標誌。若要匯入 qtree 而不讓 Trident 管理其生命週期，請新增 `--no-manage`標誌。

使用 PVC 註解匯入 qtree

建立包含以下註解的 PVC：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "flexvol1/qtree1"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

若要保留現有 FlexVol volume 和 qtree 名稱，請新增 `trident.netapp.io/importNoRename: "true"`註解。若要將

qtree 以非託管方式匯入，請新增 `trident.netapp.io/notManaged: "true"`註解。

命名和管理行為

託管導入（預設）

Trident 會將 qtree 重新命名為與 PVC 名稱相符的名稱，並將 FlexVol volume 重新命名為符合 `ontap-nas-economy`命名規則的名稱。連接路徑保持不變。

`--no-rename`

Trident 會保留現有的 FlexVol volume 和 qtree 名稱。這些名稱必須已遵循 Trident 的命名規則，否則 Trident 將無法管理 qtree 的刪除或掛載作業。

`--no-manage`

Trident 會引用 qtree，但不管理其生命週期。名稱必須遵循 Trident 的命名約定，且 Trident 不使用或管理 qtree 的自動匯出原則。

匯出原則行為

如果後端使用自動匯出原則，則匯入的 qtree 將採用基於 qtree 的匯出原則模型，其中每個 qtree 都有自己的匯出原則。Trident 在匯入時會為 qtree 指派一個空的匯出原則。發佈時，如果啟用了自動匯出且現有 FlexVol 匯出原則為 `default`或空白，Trident 會將 FlexVol 磁碟區更新為基於後端的匯出原則。

警告	更新 FlexVol volume 匯出原則可能會中斷同一 FlexVol volume 中目前已掛載的其他 qtree。僅在 FlexVol volume 中的 qtree 全部卸載後，才匯入 qtree。
警告	不要將兩個 Kubernetes 叢集指向同一個活動 SVM，也不要匯入另一個叢集仍在管理的 qtree。如果兩個叢集管理同一個磁碟區，衝突的操作可能會導致資料損壞或遺失。

SAN ONTAP

Trident支援使用卷宗導入 `ontap-san` （iSCSI、NVMe/TCP 和 FC）和 `ontap-san-economy` 司機。

Trident可以匯入包含單一 LUN 的ONTAP SAN FlexVol磁碟區。這與 `ontap-san` 驅動程序，它為每個 PVC 建立一個FlexVol volume，並在FlexVol volume內建立一個 LUN。Trident導入FlexVol volume並將其與 PVC 定義關聯。Trident可以導入 `ontap-san-economy` 包含多個 LUN 的磁碟區。

以下範例展示如何匯入託管和非託管磁碟區：

託管 **Volume**

對於託管卷， Trident 將 FlexVol volume 重命名爲格式，並將 FlexVol volume 中的 LUN `lun0` 重命名爲 `pvc-<uuid>`。

下列範例會匯入 `ontap-san-managed`**後端上的** `FlexVol volume `ontap_san_default`：

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+-----------------------------------------+--------+---------------
+----------+-----------------------------------------+--------+---------+
```

```
|                    NAME                    |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             |  STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | 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
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
|                    NAME                    |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             |  STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | 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
```

ONTAP SAN-economy 範例

以下範例展示如何為 `ontap-san-economy` 後端匯入託管磁碟區和非託管磁碟區。

託管 **Volume**

當您匯入託管磁碟區時、Trident 會取得 FlexVol 的所有權並將其重新命名。當您從相同的 FlexVol 匯入多個 LUN 時、您必須考慮此重新命名。

以下範例將 `lun1`從FlexVol `toimport`匯入為名為 `vol-managed-saneco`的受管理磁碟區：

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

匯入後 `lun1`， Trident 會重新命名 FlexVol （例如、重新命名為 `trident_lun_pool_xyz`）。若要從相同的 FlexVol 匯入其他 LUN 、請使用新的 FlexVol 名稱：

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```

註	`ontap-san-economy` 後端一次匯入一個 LUN 。您可以使用指令碼自動執行多個匯入作業。

非託管 **Volume**

匯入非託管磁碟區時、 Trident 不會取得 FlexVol 的所有權。但是、 FlexVol 和 LUN 必須遵循 Trident 命名慣例。

FlexVol 命名格式

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX` 是您後端組態中 `storagePrefix` 的值。預設值為 `trident`。
- `RANDOMSTRING` 可以是您選擇的任何字串。

LUN 命名要求

LUN 必須命名為 `lun0`。

範例

如果您的 `storagePrefix`是 `xyz`，則 LUN 的完整路徑為：

```
trident_lun_pool_xyz_randomstring/lun0
```

元素

Trident 支援使用驅動程式的 NetApp Element 軟體和 NetApp HCI Volume 匯入 `solidfire-san`。

註	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。

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

+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Azure NetApp Files

Trident 支援使用驅動程式進行 Volume 匯入 `azure-netapp-files`。

註	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`。

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    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Google Cloud NetApp Volumes

Trident 支援使用驅動程式進行 Volume 匯入 `google-cloud-netapp-volumes`。

以下範例使用磁碟區 `testvoleasiaeast1` 從後端 `backend-tbc-gcnv1` 匯入磁碟區。

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

下列範例會在兩個磁碟區位於同一個區域時匯入 `google-cloud-netapp-volumes` Volume ：

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

自訂 **Trident Volume** 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 ）。您也可以在後端層級定義範本、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：

- Volume 建立、匯入及複製作業。
- 就此而言 `ontap-nas-economy` 驅動程式中，只有 Qtree 磁碟區的名稱符合名稱範本。
- 就此而言 `ontap-san-economy` 驅動程式中，只有 LUN 名稱符合名稱範本。

限制

- 自訂磁碟區名稱僅與ONTAP本機驅動程式相容。
- 僅對以下情況支援自訂標籤： `ontap-san` ， `ontap-nas` ， 和 `ontap-nas-flexgroup` 司機。
- 自訂磁碟區名稱不適用於現有磁碟區。

可自訂 **Volume** 名稱的主要行為

- 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。
- 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.Requ
estName}}"
```

```
  },
  "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。

跨命名空間共享 **Trident NFS Volume**

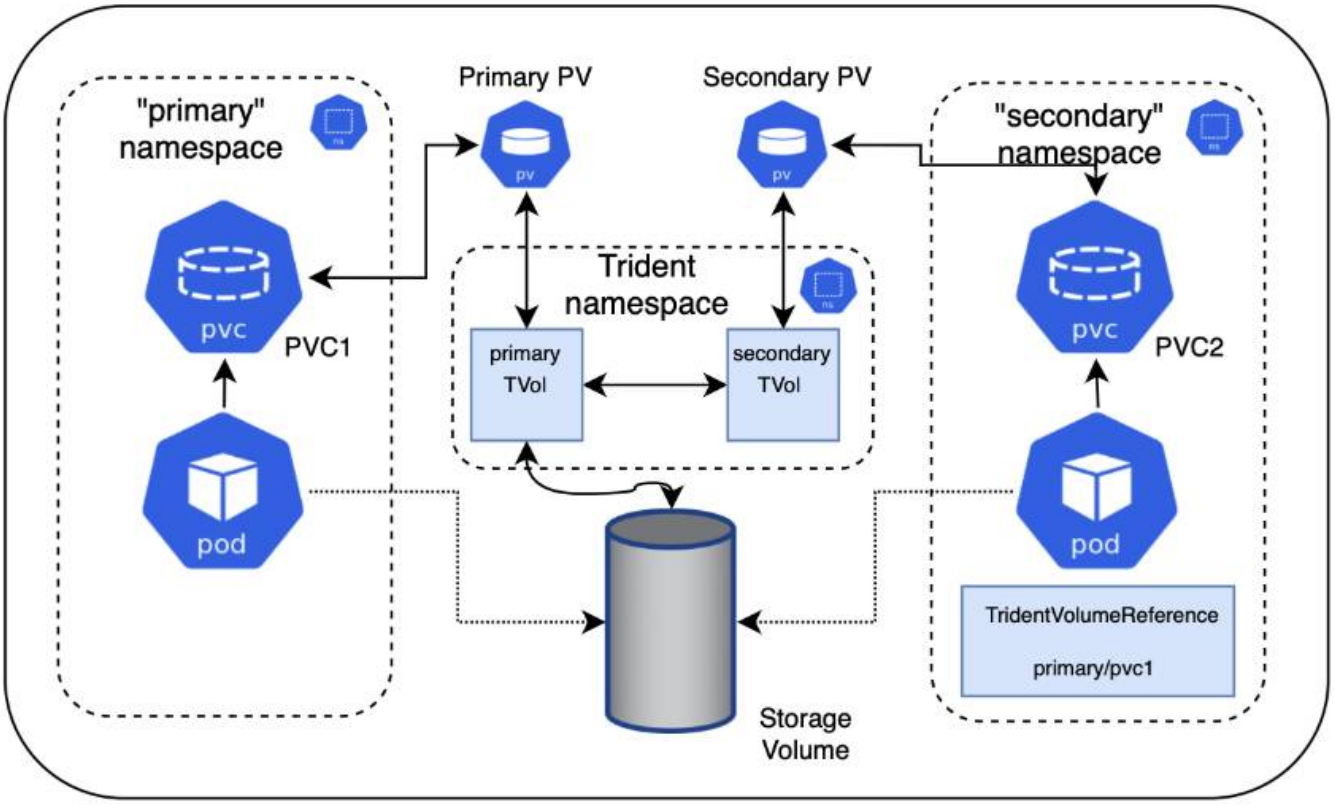
使用 Trident 、您可以在主要命名空間中建立磁碟區、並在一或多個次要命名空間中共用該磁碟區。

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

來源命名空間擁有者授予存取來源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者建立TridentVolume Reference CR的權限。

3 在目的地命名空間中建立**TridentVolume Reference**

目的地命名空間的擁有者會建立TridentVolume 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者會建立從屬的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者、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。

註

- 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. *叢集管理員：*確保已建立適當的 RBAC，以授予目標命名空間擁有者在目標命名空間中建立 TridentVolumeReference CR 的權限。
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
```

註	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 [`tridentctl` 公用程式、您可以執行 `get` 命令來取得從屬磁碟區。如需詳細資訊、請參閱 `tridentctl` 命令和選項。

```
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：

- 觀看上的示範 "NetAppTV"。

跨命名空間複製 **Trident Volume**

使用 Trident ，您可以使用同一個 Kubernetes 叢集中不同命名空間中的現有磁碟區或磁碟區快照來建立新的磁碟區。

先決條件

在複製磁碟區之前，請確定來源和目的地後端的類型相同，而且具有相同的儲存類別。

註	跨命名空間克隆僅支援 `ontap-san`和 `ontap-nas`儲存驅動程式。不支援只讀克隆。

快速入門

只需幾個步驟即可設定磁碟區複製。

1. 設定來源 **PVC** 來複製磁碟區

來源命名空間擁有者授予存取來源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者建立TridentVolume Reference CR的權限。

3. 在目的地命名空間中建立**TridentVolume Reference**

目的地命名空間的擁有者會建立TridentVolume Reference CR來參照來源PVC。

4. 在目的地命名空間中建立複製 **PVC**

目的地命名空間的擁有者會建立 PVC ，從來源命名空間複製 PVC 。

設定來源和目的地命名空間

為了確保安全性，跨命名空間複製磁碟區需要來源命名空間擁有者，叢集管理員和目的地命名空間擁有者共同作業和採取行動。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： * (pvc1`在來源命名空間中建立 PVC （`namespace1），可授予與目的地命名空間共用的權限(namespace2 cloneToNamespace。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident 會建立 PV 及其後端儲存磁碟區。

註

- 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/cloneToNamespace: namespace2,namespace3,namespace4：。
- 您可以使用共用所有命名空間 *。例如、 trident.netapp.io/cloneToNamespace: *
- 您可以隨時更新 PVC 以納入 `cloneToNamespace`附註。

2. 叢集管理員：確保已建立適當的 RBAC，以授予目標命名空間擁有者在目標命名空間中建立 TridentVolumeReference CR 的權限(namespace2）。

3. *目的地命名空間擁有者：*在目的地命名空間中建立一個參照來源 PVC 的 TridentVolumeReference 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： * (pvc2`在目的地命名空間中建立 PVC （`namespace2）使用 `cloneFromPVC`或 `cloneFromSnapshot`和 `cloneFromNamespace`註釋來指定來源 PVC 。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

限制

- 對於使用 ONTAP NAS 經濟型驅動程式配置的 PVC ，不支援唯讀複本。

使用 **SnapMirror** 複寫磁碟區

Trident 支援一個叢集上的來源磁碟區與對等叢集上的目的地磁碟區之間的鏡射關係，以便複寫資料以進行災難恢復。 您可以使用名為 Trident 鏡像關係 (TMR) 的命名空間自訂資源定義 (CRD) 來執行下列操作：

- 建立磁碟區之間的鏡射關係（ PVCS ）
- 移除磁碟區之間的鏡射關係
- 中斷鏡射關係
- 在災難情況（容錯移轉）期間提升次要 Volume
- 在計畫性容錯移轉或移轉期間、將應用程式從叢集無損移轉至叢集

複寫先決條件

在您開始之前、請確定符合下列先決條件：

叢集 **ONTAP**

- * Trident * ： Trident 22.10 版或更新版本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。
- * 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" 。

  從 ONTAP 9.10.1 開始、所有授權都會以 NetApp 授權檔案（ NLF ）的形式交付、這是一個可啟用多項功能的單一檔案。如需詳細資訊、請參閱 "ONTAP One 隨附授權" 。

註	僅支援 SnapMirror 非同步保護。

對等關係

- * 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" 。

重要	確保兩個 ONTAP 叢集之間複寫關係中使用的 SVM 名稱是唯一的。

- * Trident 和 SVM* ：對等的遠端 SVM 必須可用於目的地叢集上的 Trident 。

支援的驅動程式

NetApp Trident 支援使用 NetApp SnapMirror 技術進行磁碟區複製，使用由下列驅動程式支援的儲存類別： **`ontap-nas`** ：**NFS** `ontap-san` ：iSCSI **`ontap-san`** ： **FC** `ontap-san` ：NVMe/TCP（要求最低 ONTAP 版本 9.15.1）

註	ASA r2 系統不支援使用 SnapMirror 進行磁碟區複製。有關 ASA r2 系統的信息，請參閱"瞭解 ASA R2 儲存系統" 。

建立鏡射 **PVC**

請遵循下列步驟、並使用 CRD 範例在主要和次要磁碟區之間建立鏡射關係。

步驟

1. 在主 Kubernetes 叢集上執行下列步驟：
   a. 使用參數建立 StorageClass 物件 `trident.netapp.io/replication: true` 。

      範例

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. 使用先前建立的 StorageClass 建立 PVC 。

      範例

      ```
      kind: PersistentVolumeClaim
      apiVersion: v1
      metadata:
        name: csi-nas
      ```

```
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

c. 使用本機資訊建立 MirrorRelationship CR 。

範例

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident 會擷取磁碟區的內部資訊和磁碟區目前的資料保護（ DP ）狀態，然後填入 MirrorRelationship 的狀態欄位。

d. 取得 TridentMirrorRelationship CR 以取得 PVC 的內部名稱和 SVM 。

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
```

```
    observedGeneration: 1
```

2. 在次 Kubernetes 叢集上執行下列步驟：

   a. 使用 trident.netapp.io/replication: true 參數建立 StorageClass 。

      範例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

   b. 使用目的地和來源資訊建立 MirrorRelationship CR 。

      範例

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  replicationPolicy: MirrorAllSnapshots
  replicationSchedule: 5min
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

      TMR `spec` 有四個頂層欄位： `state` ，用於設定所需的關係狀態（請參閱 Volume 複寫狀態）； `volumeMappings` ，用於將本機 PVC 與其遠端 Volume 控點建立關聯；以及下述選用的 `replicationPolicy` 和 `replicationSchedule` 欄位。

      `replicationPolicy` 和 `replicationSchedule` 欄位為選填。請將 `replicationPolicy` 設定為 `ONTAP` 上的非同步 `SnapMirror` 原則名稱，並將 `replicationSchedule` 設定為 `ONTAP` 排程的名稱。如果省略任一欄位，`Trident` 將使用後端設定的對應複寫原則或排程。請在建立關係時設定這些欄位；關係建立後，您將無法變更原則或排程。若要使用不同的設定，請刪除並重新建立 `TMR`。

Trident 使用設定的複寫原則和排程建立並初始化 SnapMirror 關係。

1. 使用先前建立的 StorageClass 建立 PVC 、作為次要（ SnapMirror 目的地）。

   範例

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

   Trident 會檢查 TridentMirrorRelationship CRD ，如果關係不存在，則無法建立 Volume 。如果存在這種關係， Trident 將確保新的 FlexVol volume 放置在與 MirrorRelationship 中定義的遠端 SVM 對等的 SVM 上。

Volume 複寫狀態

Trident Mirror Relationship （ TMR ）是一種 CRD 、代表 PVC 之間複寫關係的一端。目的地 TMR 具有狀態，可告知 Trident 所需的狀態。目的地 TMR 有下列狀態：

- * 建立 * ：本機 PVC 是鏡射關係的目的地 Volume 、這是新的關係。
- * 升級 * ：本機 PVC 為可讀寫且可掛載、目前無鏡射關係。
- * 重新建立 * ：本機 PVC 是鏡射關係的目的地 Volume 、先前也屬於該鏡射關係。
  - 如果目的地磁碟區與來源磁碟區有任何關係、則必須使用重新建立的狀態、因為它會覆寫目的地磁碟區內容。
  - 如果磁碟區先前未與來源建立關係、則重新建立的狀態將會失敗。

在非計畫性容錯移轉期間升級次要 **PVC**

在次 Kubernetes 叢集上執行下列步驟：

- 將 TridentMirrorRelationship 的 *spec.state* 欄位更新為 `promoted`。

在規劃的容錯移轉期間升級次要 **PVC**

在計畫性容錯移轉（移轉）期間、請執行下列步驟來升級次要 PVC ：

步驟

1. 在主要 Kubernetes 叢集上、建立 PVC 的快照、並等待快照建立完成。
2. 在主要 Kubernetes 叢集上，建立 TridentTridentSnapshotInfoCR 以取得內部詳細資訊。

範例

```
kind: TridentSnapshotInfo
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  snapshotName csi-nas-snapshot
```

a. 在輔助 Kubernetes 叢集上，將 *TridentMirrorRelationship* CR 的 *spec.state* 欄位更新為 *promoted*，並將 *spec.volumeMappings[].promotedSnapshotHandle* 更新為快照的 internalName。

3. 在次要 Kubernetes 叢集上、確認要升級的 TridentMirrorRelationship 狀態（ STATUS.STATUS 欄位）。

在容錯移轉後還原鏡射關係

還原鏡射關係之前、請先選擇要設為新主要的一面。

步驟

1. 在次要 Kubernetes 叢集上、確保已更新 TridentMirrorRelationship 上 *spec.remoteVolumeHandle* 欄位的值。
2. 在次要 Kubernetes 叢集上，將 TridentMirrorRelationship 的 *spec.state* 欄位更新為 `reestablished`。

其他作業

Trident 支援在主要和次要磁碟區上執行下列作業：

將主要 **PVC** 複製到新的次要 **PVC**

請確定您已擁有主要 PVC 和次要 PVC 。

步驟

1. 從已建立的次要（目的地）叢集刪除 PersistentVolume Claim 和 TridentMirrorRelationship CRD 。
2. 從主（來源）叢集刪除 TridentMirrorRelationship CRD 。
3. 在主要（來源）叢集上建立新的 TridentMirrorRelationship CRD 、以用於您要建立的新次要（目的地）PVC 。

調整鏡射、主要或次要 **PVC** 的大小

PVC 可以正常調整大小、如果資料量超過目前大小、 ONTAP 會自動擴充任何目的地 flevxols 。

從 **PVC** 移除複寫

若要移除複寫、請在目前的次要磁碟區上執行下列其中一項作業：

- 刪除次要 PVC 上的 MirrorRelationship 。這會中斷複寫關係。
- 或者、將 spec.state 欄位更新為 *updated* 。

刪除 **PVC** （先前已鏡射）

Trident 會檢查複寫的 PVCS ，並在嘗試刪除磁碟區之前先釋放複寫關係。

刪除 **TMR**

刪除鏡射關係一側的 TMR 會導致其餘 TMR 在 Trident 完成刪除之前轉換至 升 遷狀態。如果選取要刪除的 *TMR* 已處於 *_Promive* 狀態，則沒有現有的鏡射關係， TMR 將會移除，而 Trident 會將本機 PVC 升級為 *ReadWrite* 。此刪除作業會在 ONTAP 中針對本機磁碟區釋出 SnapMirror 中繼資料。如果此磁碟區在未來的鏡射關係中使用、則在建立新的鏡射關係時、它必須使用具有 _ 建立 _ 磁碟區複寫狀態的新 TMR 。

當 **ONTAP** 連線時、請更新鏡射關係

建立鏡射關係之後、可以隨時更新它們。您可以使用 `state: promoted` 或 `state: reestablished` 欄位來更新關聯。將目的地 Volume 升級為一般 ReadWrite Volume 時、您可以使用 *promotedSnapshotHandle* 來指定特定快照、將目前的 Volume 還原至。

當 **ONTAP** 離線時更新鏡射關係

您可以使用 CRD 來執行 SnapMirror 更新，而無需 Trident 直接連線至 ONTAP 叢集。請參閱下列 TridentActionMirrorUpdate 範例格式：

範例

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` 反映 TridentActionMirrorUpdate CRD 的狀態。它可以取自 *sued* 、 *in progress* 或 *Failed* 的值。

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。

提示	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`設置爲 `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。

註 「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:"1e11e4a2108024935ecfcb2912226cedeafd99df",
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:"1e11e4a2108024935ecfcb2912226cedeafd99df",
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引進拓撲感知的標籤 `topology.kubernetes.io/region` 和 `topology.kubernetes.io/zone`）。 `topology.kubernetes.io/region` 標籤對於 Trident 中的拓撲感知資源配置是必要的。在安裝 Trident 之前，這些標籤應存在於節點上。如果在安裝後新增或變更拓撲標籤，只需重新啟動 Trident 節點 Pod，Trident 即可套用這些變更。

```
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","kube
rnetes.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","kube
rnetes.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","kube
rnetes.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

```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

```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"
      }
    ]
}
```

註	`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`，設置爲 `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: null
name: pvc-san
spec: null
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：

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl 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                         Message
  ----    ------                ----  ----                         -------
  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: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      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 **Trident** 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

卷快照受支援 `ontap-nas` ， `ontap-nas-flexgroup` ， `ontap-san` ， `ontap-san-economy` ， `solidfire-san` ， `azure-netapp-files` ，和 `google-cloud-netapp-volumes`司機。

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。

> 註 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 pvc1-snap
Name:         pvc1-snap
Namespace:    default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
```

```
    Name:        pvc1
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。

警告	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 `pvc1-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: pvc1-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

1. * 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`。

註	。<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>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

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
```

> 註 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。

從快照進行原位磁碟區還原

Trident 使用（ TASR ） CR 從快照提供快速的原位磁碟區還原 `TridentActionSnapshotRestore`。此 CR 是 Kubernetes 的必要行動、在作業完成後不會持續存在。

Trident支援快照恢復 `ontap-san`， `ontap-san-economy` ， `ontap-nas` ， `ontap-nas-flexgroup` ， `azure-netapp-files` ， `google-cloud-netapp-volumes` ， 和 `solidfire-san`司機。

開始之前

您必須擁有受約束的 PVC 和可用的 Volume 快照。

- 確認 PVC 狀態為「已連結」。

```
kubectl get pvc
```

- 驗證 Volume 快照是否已準備就緒可供使用。

```
kubectl get vs
```

步驟

1. 建立 TASR CR 。本示例爲 PVC 和 Volume Snapshot 創建 CR `pvc1` `pvc1-snapshot`。

   註 | TASR CR 必須位於 PVC 與 VS 所在的命名空間中。

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. 套用 CR 以從快照還原。此示例從 Snapshot 恢復 `pvc1`。

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

結果

Trident 會從快照還原資料。您可以驗證快照還原狀態：

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

註

- 在大多數情況下， Trident 不會在發生故障時自動重試作業。您需要再次執行此作業。
- 不具備管理員存取權限的 Kubernetes 使用者可能必須獲得管理員的權限、才能在其應用程式命名空間中建立 TASR CR 。

刪除含有相關快照的 PV

刪除含相關快照的持續 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.yaml
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
```

> 註 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"

使用 **Trident Volume** 群組快照

Kubernetes 持久性磁碟區 (PV) 磁碟區組快照 NetApp Trident 提供了建立多個磁碟區（一組磁碟區快照）的功能。此磁碟區組快照代表在同一時間點建立的多個磁碟區的副本。

> 註 VolumeGroupSnapshot 是 Kubernetes 中的一個 Beta 功能，包含 Beta 版 API。VolumeGroupSnapshot所需的最低版本為 Kubernetes 1.32。

建立卷宗組快照

以下儲存驅動程式支援磁碟區組快照：

- `ontap-san`驅動程式 - 僅適用於 iSCSI 和 FC 協議，不適用於 NVMe/TCP 協定。
- `ontap-san-economy` - 僅適用於 iSCSI 協定。
- 「ONTAP-NAS」

> 註 NetApp ASA r2 或 AFX 儲存系統不支援磁碟區組快照。

開始之前

- 確保您的 Kubernetes 版本是 K8s 1.32 或更高版本。
- 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 發行版不包含外部快照控制器和 CRD，請參閱部署 Volume Snapshot 控制器 。

  > 註 如果在 GKE 環境中建立按需磁碟區組快照，請不要建立快照控制器。GKE使用內建的隱藏式快照控制器。

- 在快照控制器 YAML 中，設定 `CSIVolumeGroupSnapshot`功能門控設定為“true”，以確保磁碟區組快照已啟用。
- 在建立磁碟區組快照之前，建立所需的磁碟區組快照類別。
- 確保所有 PVC/磁碟區都在同一個 SVM 上，以便能夠建立 VolumeGroupSnapshot。

步驟

- 在創建 VolumeGroupSnapshot 之前建立一個 VolumeGroupSnapshotClass。如需詳細資訊、請 "卷冊組快照類"參閱。

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- 使用現有儲存類別建立具有所需標籤的 PVC，或將這些標籤新增至現有 PVC。

  以下範例使用以下方式建立 PVC `pvc1-group-snap`作為資料來源和標籤 `consistentGroupSnapshot: groupA`.根據您的要求定義標籤鍵和值。

```
kind: PersistentVolumeClaim
apiVersion: v1
```

```
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- 建立具有相同標籤的 VolumeGroupSnapshot (`consistentGroupSnapshot: groupA`) 在 PVC 中指定。

  此範例建立磁碟區組快照：

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

使用群組快照恢復磁碟區數據

您可以使用作為磁碟區組快照的一部分所建立的各個快照來還原各個持久性磁碟區。您無法將磁碟區組快照作為一個整體進行還原。

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
```

> 註 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。

從快照進行原位磁碟區還原

Trident 使用（ TASR） CR 從快照提供快速的原位磁碟區還原 `TridentActionSnapshotRestore`。此 CR 是 Kubernetes 的必要行動、在作業完成後不會持續存在。

如需更多資訊、請參閱 "從快照進行原位磁碟區還原"。

刪除與群組快照關聯的 PV

刪除群組磁碟區快照時：

- 您可以刪除整個 VolumeGroupSnapshots，而不是刪除群組中的單一快照。
- 如果在持久卷存在快照的情況下刪除了該持久卷，Trident 會將該卷移至「刪除」狀態，因為必須先刪除快照，然後才能安全刪除該卷。
- 如果已使用分組快照建立了克隆，然後要刪除該群組，則會開始克隆拆分操作，並且在拆分完成之前無法刪除該群組。

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-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. 建立Snapshot控制器。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
```

```
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

註	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

- "卷冊組快照類"
- "Volume快照"

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。
- 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 24.06 （或任何 24.06 點版本）升級至 25.06 。
- 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 23.07 、而且想要升級至 25.06 ：
  a. 第一次從 23.07 升級至 24.06 。
  b. 然後從 24.06 升級至 25.06 。

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

通常情況下，您應該使用與初始安裝相同的方法進行升級，但您也可以"在安裝方法之間移動"。有兩種升級 Trident 的選項。

- "使用Trident營運者進行升級"

> 提示　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

了解 **Trident** 營運者升級工作流程

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 的眾多"使用 Trident 運算子的優點"之一是能夠自動處理 Trident 和 Kubernetes 物件，而不會中斷現有掛載的磁碟區。這樣，Trident 可以支援零停機升級，或"滾動更新 _"。具體來說，Trident Operator 與 Kubernetes 叢集通訊以：

- 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。

> 提示　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

您可以使用 Trident Operator 手動或透過 Helm 升級 Trident。您可以從一個 Trident Operator 安裝升級到另一個 Trident Operator 安裝，也可以從 `tridentctl`安裝升級到 Trident Operator 版本。在升級 Trident Operator 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操作員安裝升級到另一個叢集範圍的Trident操作員安裝。所有Trident版本都使用叢集範圍的運算元。

註	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 1.25 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. 更新 `operator.yaml`， `tridentorchestrator_cr.yaml` ， 和 `post_1_25_bundle.yaml`使用您要升級到的版本（例如 25.06）的註冊表和圖像路徑以及正確的金鑰。
3. 刪除用於安裝目前Trident實例的Trident操作員。例如，如果您從 25.02 升級，請執行以下命令：

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. 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。
5. 使用適合您環境的正確 bundle YAML 檔案安裝Trident ，其中 *<bundle.yaml>* 是

`bundle_pre_1_25.yaml`或者 `bundle_post_1_25.yaml`根據您的 Kubernetes 版本。例如，如果您正在安裝Trident 25.06.0，請執行下列命令：

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. 編輯三叉戟項圈以包含圖像 25.06.0。

升級 **Helm** 安裝

您可以升級 Trident Helm 安裝。

警告	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-mapkubeaps 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"，可以使用 `helm upgrade trident netapp-trident/trident-operator --version 100.2610.0`一步升級。如果您沒有新增 Helm 倉庫或無法使用它進行升級：
   a. 從下載最新的 Trident 版本"GitHub的_Assets區段"。
   b. 使用 `helm upgrade` 命令，其中 `trident-operator-26.10.1.tgz` 反映您要升級到的版本。

```
helm upgrade <name> trident-operator-26.10.1.tgz
```

註	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。

> 註　在切換安裝方法之前，請先查看 "在安裝方法之間移動"。

步驟

1. 下載最新的 Trident 版本。

```
# Download the release required [26.10.1]
mkdir 26.10.1
cd 26.10.1
wget
https://github.com/NetApp/trident/releases/download/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.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:                v26.10.0
```

使用 **tridentctl** 升級 **Trident**

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

在使用 `tridentctl`進行升級之前，請先檢查"選擇升級方法"。

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"。

重要	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。

▼ 命令

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。

註	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| wl| ps（預設）。

-s、 --server string

Trident REST 介面的位址 / 連接埠。

警告	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 所需的容器影像表格。

```
tridentctl images [flags]
```

旗標

- `-h、 --help`：影像說明。
- `-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：自動支援遙測的容器映像（預設為「netapp/trident autosupport:<current-version>」）。
--autosupport-proxy string ：用於發送自動支援遙測的代理程式的位址/連接埠。
--enable-node-prep ：嘗試在節點上安裝所需的軟體包。
--generate-custom-yaml ：無需安裝任何東西即可產生 YAML 檔案。
-h ， --help ：安裝幫助。
--http-request-timeout ：覆蓋Trident控制器的 REST API 的 HTTP 請求逾時（預設為 1 分 30 秒）。
--image-registry string ：內部影像註冊表的位址/連接埠。
--k8s-timeout duration ：所有 Kubernetes 操作的逾時時間（預設為 3m0s）。
--kubelet-dir string ：kubelet 內部狀態的主機位置（預設為「/var/lib/kubelet」）。
--log-format string ： Trident日誌格式（文字、json）（預設「文字」）。
--node-prep ：使Trident能夠準備 Kubernetes 叢集的節點，以使用指定的資料儲存協定管理磁碟區。*現在， iscsi`是唯一支援的值。從 OpenShift 4.19 開始，此功能支援的最低Trident版本為 25.06.1。 *
`--pv string ： Trident使用的舊版 PV 的名稱，確保其不存在（預設為「trident」）。
--pvc string ： Trident使用的舊式 PVC 的名稱，確保其不存在（預設為「trident」）。
--silence-autosupport ：不要自動將自動支援包傳送到NetApp （預設為 true）。
--silent ：安裝期間停用大部分輸出。
--trident-image string ：要安裝的Trident映像。
--k8s-api-qps ：Kubernetes API 請求的每秒查詢數 (QPS) 限制（預設 100；可選）。
--use-custom-yaml ：使用安裝目錄中存在的任何現有 YAML 檔案。
--use-ipv6 ：使用 IPv6 進行 Trident 的通訊。

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。設定為時 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 `tridentctl update backend`命令和更新的路徑 `backend.json`文件。

   例子： tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident

```
{
  "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、您也可以在 enviornment 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 tridenctl 只會在指定的資料夾中搜尋。

監控 Trident 儲存設備作業

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。

註	預設情況下，Trident 的指標會暴露在目標連接埠上。 `8001`在 `/metrics`端點。安裝Trident時，這些指標*預設為啟用*。您可以設定透過 HTTPS 在連接埠上消費Trident指標。 `8444`也一樣。

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

您應該定義一個 Prometheus 目標來收集指標並獲取有關Trident管理的後端、它創建的捲等的資訊。

看"Prometheus Operator 文檔"。

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 定義擷取由下列方式傳回的指標： `trident-csi`服務，特別是尋找 `metrics`服務的端點。因此，Prometheus 現在已配置為能夠理解 Trident 的指標。

除了可直接從 Trident 取得的指標之外、 kibelet 還會透過自己的指標端點來公開許多 `kubelet_volume_*`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"。

透過 **HTTPS** 使用**Trident**指標

若要透過 HTTPS（連接埠 8444）使用Trident指標，您必須修改 ServiceMonitor 定義以包含 TLS 設定。你還需要複製 `trident-csi`秘密來自 `trident`將命名空間指向 Prometheus 運行所在的命名空間。您可以使用以下命令執行此操作：

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

HTTPS 指標的範例 ServiceMonitor 如下圖所示：

```
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:
      - interval: 15s
        path: /metrics
        port: https-metrics
        scheme: https
        tlsConfig:
          ca:
            secret:
              key: caCert
              name: trident-csi
          cert:
            secret:
              key: clientCert
              name: trident-csi
          keySecret:
            key: clientKey
            name: trident-csi
          serverName: trident-csi
```

Trident支援所有安裝方法中的 HTTPS 指標：tridentctl、Helm chart 和 Operator：

- 如果您正在使用 `tridentctl install`命令，你可以傳遞 `--https-metrics`啟用 HTTPS 指標的標誌。
- 如果您使用的是 Helm Chart，則可以進行設定。 `httpsMetrics`用於啟用 HTTPS 指標的參數。
- 如果您使用的是 YAML 文件，則可以新增以下內容： `--https_metrics`向… `trident-main`容器中的 `trident-deployment.yaml`文件。

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

註	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 不會收集或傳輸個人識別資訊（PII）或個人資訊。它附帶 "EULA"不適用於 Trident 容器映像本身。您可以深入瞭解 NetApp 對資料安全性和信任的承諾 "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: null
      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 `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 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
```

警告 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
```

註	確保在"default"（錯誤）下"etc/multipath.conf"包含"fappe_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
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
```

註	確保在"default"（錯誤）下"etc/multipath.conf"包含"fappe_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。

註	• 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
```

FC工具

使用作業系統的命令來安裝FC工具。

- 當使用搭配 FC PV 執行 RHEL/Red Hat Enterprise Linux CoreOS （ RHCOS ）的工作節點時，請在 StorageClass 中指定 `discard` mountOption 以執行內嵌空間回收。請參閱 "Red Hat 說明文件"。

RHEL 8以上

1. 安裝下列系統套件：

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. 啟用多重路徑：

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

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

3. 確定 `multipathd`執行中：

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

Ubuntu

1. 安裝下列系統套件：

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

2. 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
```

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

3. 確定 `multipath-tools`已啟用並正在執行：

```
sudo systemctl status multipath-tools
```

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）

註	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：

```
pkill /usr/local/bin/netappdvp
pkill /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」、「f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：

      ```
      ----trident-installer-26.10.1.tar.gz
      wget https://github.com/NetApp/trident/releases/download/10.0/
      tar zxf trident-installer-26.10.1.tar.gz
      ```

```
.. Move to a location in the bin path:
+
[source,console]
```

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

```
.. Create a location for the configuration file:
+
[source,console]
```

sudo mkdir -p /etc/netappdvp

```
.. Create the configuration file:
+
[source,console]
```

cat << EOF > /etc/netappdvp/ontap-nas.json

```
+
[source,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

```
. After placing the binary and creating the configuration file, start the
Trident daemon using the desired configuration file.
+
[source,console]
```

sudo trident --config=/etc/netappdvp/ontap-nas.json

```
+
NOTE: Unless specified, the default name for the volume driver is
"netapp".
+
After the daemon is started, you can create and manage volumes by using
the Docker CLI interface.

. Create a volume:
+
[source,console]
```

docker volume create -d netapp --name trident_1

```
. Provision a Docker volume when starting a container:
+
[source,console]
```

docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol alpine ash

```
. Remove a Docker volume:
+
[source,console]
```

docker volume rm trident_1

```
+
```

```
[source,console]
```

docker volume rm trident_2

```
== Start Trident at system startup

A sample unit file for systemd based systems can be found at
`contrib/trident.service.example` in the Git repo. To use the file with
RHEL, do the following:

. Copy the file to the correct location.
+
You should use unique names for the unit files if you have more than one
instance running.
+
[source,console]
```

cp contrib/trident.service.example /usr/lib/systemd/system/trident.service

```
. Edit the file, change the description (line 2) to match the driver name
and the configuration file path (line 9) to reflect your environment.
. Reload systemd for it to ingest changes:
+
[source,console]
```

systemctl daemon-reload

```
. Enable the service.
+
This name varies depending on what you named the file in the
`/usr/lib/systemd/system` directory.
+
[source,console]
```

systemctl enable trident

```
. Start the service.
+
[source,console]
```

systemctl start trident

```
. View the status.
+
[source,console]
```

systemctl status trident

```
NOTE: Any time you modify the unit file, run the `systemctl daemon-reload`
command for it to be aware of the changes.
```

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
```

註	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
```

重要	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
```

> 註 | 如果您未指定任何 "選項"，將使用驅動程式的預設值。

- 覆蓋預設卷大小。請參閱以下範例，使用驅動程式建立 20 GiB 的磁碟區：

```
docker volume create -d netapp --name my_vol --opt size=20G
```

> 提示 | 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
```

> 重要 | 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

使用時 `ontap-nas`， `ontap-san` ， 和 `solidfire-san`儲存驅動程序， Trident可以克隆磁碟區。使用時 `ontap-nas-flexgroup`或者 `ontap-nas-economy`驅動程式不支援克隆。從現有磁碟區建立新磁碟區將建立一個新的快照。

- 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 ， iSCSI 和 FC 的 Volume Create 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 `none`、表示不會自動為磁碟區建立任何快照。除非您的儲存管理員修改，否則所有 ONTAP 系統上都會有一個名為「預設」的原則，該原則會建立並保留六個小時，兩個每日快照和兩個每週快照。瀏覽到磁碟區任何目錄中的目錄、即可恢復快照中保留的資料 `.snapshot`。
「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。

選項	說明
「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。
加密	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。 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"：。
「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。
「napshotDir	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：

- 建立一個 10 GiB 卷：

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- 建立一個帶有快照的 100 GiB 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
```

最小磁碟區大小為 20 MiB。

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》	卷的大小，預設為 1 GiB 或配置條目…“defaults” ：{“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。

- 創建 10 GiB 黃金卷：

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- 創建 100 GiB 青銅卷：

```
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「stystemctl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`
「torageDriverName」	儲存驅動程式名稱	`ontap-nas`、 `ontap-san`、 `ontap-nas-economy`、 `ontap-nas-flexgroup`、 `solidfire-san`
「toragePrefix」	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`

提示	請勿將（包括預設值）用於 `storagePrefix`元素後端。根據預設、 `solidfire-san`驅動程式會忽略此設定、而不使用前置碼。NetApp 建議您使用特定的 TenantId 來進行 Docker Volume 對應，或是使用包含 Docker 版本，驅動程式資訊和 Docker 原始名稱的屬性資料，以供任何名稱佔用。

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」	傳輸協定LIF的IP位址。 • ONTAP NAS 驅動程序 * ： NetApp 建議指定 `dataLIF`。如果未提供， Trident 會從 SVM 擷取 dataLIFs 。您可以指定完整網域名稱（ FQDN ），以用於 NFS 裝載作業，讓您建立循環 DNS ，以便在多個 dataLIFs 之間進行負載平衡。 • ONTAP SAN 驅動程式 * ：請勿指定 iSCSI 或 FC 。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 或 FC 生命負載。如果明確定義、就會產生警告 `dataLIF`。	`10.0.0.2`

選項	說明	範例
《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%`
「nfsMountOptions」	精細控制 NFS 裝載選項；預設為「-o nfsver=3 」。* 僅適用於 `ontap-nas`和 `ontap-nas-economy`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 ）。	`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
「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	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。 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"：。	是的
「unixPermissions」	NAS 選項適用於已佈建的 NFS 磁碟區、預設為 `777`	`777`
「napshotDir	用於存取目錄的 NAS 選項 `.snapshot`。	啟用或停用對 `.snapshot` 目錄的存取。 有效值： `true`、 `false`。預設行為可能因 ONTAP 和 Trident 版本而異。請明確設定以確保行為一致。
「匯出政策」	NFS 匯出原則使用的 NAS 選項、預設為 `default`	`default`
《生態樣式》	NAS選項、可存取已配置的NFS Volume。 NFS支援 `mixed` 和 `unix` 安全樣式：預設值為 `unix`。	`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`。	無
`skipRecoveryQueue`	刪除磁碟區時，繞過儲存中的復原佇列，立即刪除磁碟區。	``

擴充選項

`ontap-nas`和 `ontap-san`驅動程式會為每個 Docker Volume 建立 ONTAP FlexVol 。ONTAP 每個叢集節點最多可支援 1000 個 FlexVols ，叢集最多 12 ， 000 個 FlexVol Volume 。如果您的 Docker Volume 需求符合這項限制，則 `ontap-nas`由於 FlexVols 提供的額外功能（例如 Docker Volume 精細快照和複製），因此驅動程式是首選的 NAS 解決方案。

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。

此 `ontap-nas-economy`驅動程式會在自動管理的 FlexVol Volume 集區內，以 ONTAP qtree 的形式建立 Docker Volume 。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。此 `ontap-nas-economy`驅動程式不支援 Docker Volume 精細快照或複製。

註	Docker swarm 目前不支援此 `ontap-nas-economy`驅動程式，因為 Docker swarm 不會在多個節點之間協調磁碟區建立。

此 `ontap-san-economy`驅動程式會在自動管理的 FlexVol 磁碟區的共用集區中，將 Docker 磁碟區建立為 ONTAP 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。
- 建議的最小FlexGroup成員/磁碟區大小為 100 GiB。
- FlexGroup 磁碟區不支援複製。

有關 FlexGroups 及適用於 FlexGroups 的工作負載的資訊，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取得進階功能和大規模功能，您可以使用執行多個 Docker Volume 外掛程式執行個體，其中一個使用，另 `ontap-nas-economy`一個使用 `ontap-nas`。

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"
security login create -username <user_name\> -application http -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
}
```

▼ **SCSI over FC 範例，適用於 <code> ONTAP — </code> 驅動程式**

```
{
  "version": 1,
  "backendName": "ontap-san-backend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "sanType": "fcp",
  "svm": "trident_svm",
  "username": "vsadmin",
  "password": "password",
```

```
    "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	https://<login>:<password>@<mvip>/json-rpc/<element-version>	https://admin:admin@192.168.160.3/json-rpc/8.0
《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。

註	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遇到競爭條件等問題，導致在沒有明確「贏家」的情況下，為單一要求建立大量磁碟區；例如， Element 具有允許磁碟區擁有相同名稱但不同 ID 的功能。

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

Trident 部署建議

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"。

Trident 儲存組態最佳實務

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、以達到最大的靈活度 "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。

例如，雙節點 ONTAP 叢集最多可裝載 2000 個 FlexVol 磁碟區。將最大Volume數設為1250似乎非常合理。不過、如果只有 "集合體"一個節點指派給 SVM 、或是無法針對一個節點指派的集合體進行資源配置（例如、由於容量）、則另一個節點會成為所有 Trident 資源配置 Volume 的目標。這表示在達到該值之前、可能會達到該節點的磁碟區限制 `max-volumes`、進而影響使用該節點的 Trident 和其他磁碟區作業。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。

複製磁碟區

NetApp Trident在使用時支援克隆卷 `ontap-nas`， `ontap-san` ， 和 `solidfire-san`儲存驅動程式。使用時`ontap-nas-flexgroup`或者 `ontap-nas-economy`驅動程式不支援克隆。從現有磁碟區建立新磁碟區將建立一個新的快照。

警告	避免複製與其他 StorageClass 關聯的 PVC。請在同一 StorageClass 內執行複製操作，以確保相容性並防止意外行為。

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。

重要	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 "ONTAP 命令參照"參閱。

限制Kubernetes叢集成員存取儲存資源

限制對 Trident 所建立的 NFS 磁碟區， iSCSI LUN 和 FC LUN 的存取，是 Kubernetes 部署安全性態勢的重要元件。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的儲存設備、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**

此 `showmount`功能可讓 NFS 用戶端查詢 SVM 以取得可用 NFS 匯出清單。部署至 Kubernetes 叢集的 Pod 可針對發出 `showmount -e`命令，並接收可用掛載清單，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

> 註 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。

區塊大小和頻寬對 IOPS 數量有直接影響。隨著區塊大小增加，系統會將頻寬提高到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 數量會減少。如需 QoS 和效能的詳細資訊，請參閱 "服務品質SolidFire"。

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。

> 註 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"（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。

將 Trident 與 Kubernetes 整合

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》	NO腳 註：3[]	NO腳 註：3[]	是註腳：5[]	是的	NO腳 註：3[]	是的	NO腳 註：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	NO腳 註：3[]	是的	NO腳 註：3[]

> 上表的註腳： Yesporte:1[] ：非由 Trident 管理 Yesporte:2[] ：由 Trident 管理，但非 PV 精細 NO腳 註 :3[] ：非由 Trident 管理，非 PV 精細腳註： 4[] ：支援原始區塊磁碟區 Yesport: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。

註	「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
「olidfire - san」	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
《azure-NetApp-fil形》	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

Google Cloud NetApp Volumes 後端驅動程式

Trident 支援適用於 NAS 工作負載的 `google-cloud-netapp-volumes` 驅動程式，以及適用於 SAN（iSCSI）工作負載的 `google-cloud-netapp-volumes-san`。在 GCP 上執行的 OpenShift 叢集上，從 Trident 26.10 開始，這兩個驅動程式都支援 Google Cloud Workload Identity Federation。這包括自行管理的 OpenShift Container Platform（OCP，包括 IPI 部署）和 OpenShift Dedicated（OSD）。

有關後端組態和 WIF 先決條件，請參閱"配置 Google Cloud NetApp Volumes"和"設定 Google Cloud NetApp Volumes 以用於 SAN 工作負載"。

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，通常可以指定一組參數。管理員無法建立具有相同儲存憑證和不同參數集的另一個後端。隨著虛擬池的引入，這個問題得到了緩解。虛擬池是在後端和 Kubernetes 儲存類別之間引入的層級抽象，以便管理員可以定義參數以及可以透過 Kubernetes 儲存類別作為選擇器引用的標籤，以與後端無關的方式。可以使用Trident為所有支援的NetApp後端定義虛擬池。此清單包括SolidFire/ NetApp HCI、 ONTAP以及Azure NetApp Files。

註	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。

> 註 Kubernetes 樹內置備程式目前不支援 NFS ， iSCSI 或 FC PV 的 Volume resize 作業。Trident 支援擴充 NFS ， iSCSI 和 FC 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、iSCSI 和 FC PV 的大小。這樣使用者就可以直接透過 Kubernetes 圖層調整磁碟區的大小。所有主流NetApp儲存平台（包括ONTAP）及SolidFire/ NetApp HCI後端均可進行磁碟區擴充。為了方便日後擴展，請設定 `allowVolumeExpansion`到 `true`在與該磁碟區關聯的 StorageClass 中。每當需要調整持久卷的大小時，請編輯以下內容： `spec.resources.requests.storage`在持久卷聲明中加入所需卷大小的註解。Trident會自動處理儲存叢集上磁碟區的大小調整。

將現有磁碟區匯入**Kubernetes**

磁碟區匯入功能允許將現有儲存磁碟區匯入 Kubernetes 環境。目前這得到了以下方面的支持： `ontap-nas` ， `ontap-nas-flexgroup` ， `solidfire-san` ， 和 `azure-netapp-files`司機。將現有應用程式移植到 Kubernetes 或災難復原場景中，此功能非常有用。

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`使用了驅動程序，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。這樣可以確保卷號的唯一性。

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。

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。

警告	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。

註	下表僅包含與日誌服務相關的儲存配置變數。您可以在 "Red Hat OpenShift 記錄文件" 中找到其他選項，這些選項應根據您的部署情況進行檢視、配置和使用。

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。
"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 dataLIF 。
"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。

變動	詳細資料
「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。

但是，如果您在初始安裝後進行部署，則需要使用 Ansible 元件 playbook。此過程可能會因 OpenShift 版本而略有不同，因此請務必閱讀並遵循"Red Hat OpenShift Container Platform 3.11 文件"適用於您版本的相關說明。

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。

註	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 dataLIF 。
"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。

上述變數和部署流程可能會隨 OpenShift 的每個版本而變化。請務必查看並遵循"Red Hat 的 OpenShift 部署指南"您所用版本的相關說明，以便根據您的環境進行配置。

Trident 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 volume 上

> 註 NetApp 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 。

註	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"。

註	此 `ontap-san-economy` 驅動程式支援磁碟區匯入。使用此 `ontap-nas-economy` 驅動程式匯入 Qtree 是一項技術預覽功能，不支援在生產環境中使用。此 `ontap-flexgroup-economy` 驅動程式不支援匯入。如需詳細資訊，請參閱 "匯入 Trident Volume"。

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"

使用Trident實現有狀態應用程式的故障轉移自動化

Trident 的強制分離功能可讓您自動將磁碟區從 Kubernetes 叢集中不健康的節點分離，從而防止資料損壞並確保應用程式的可用性。此功能在節點無響應或因維護而離線的情況下特別有用。

強制分離的詳細資料

強制分離適用於 `ontap-san`， `ontap-san-economy` ， `ontap-nas` ， 和 `ontap-nas-economy`僅有的。在啟用強制分離之前，必須在 Kubernetes 叢集上啟用非正常節點關閉 (NGNS)。Kubernetes 1.28 以上版本預設啟用 NGNS。有關詳細信息，請參閱"Kubernetes ：非正常節點關機" 。

註	使用 `ontap-nas`或 `ontap-nas-economy`驅動程式時，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。
警告	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. 停止對掛載到該節點的磁碟區的後端 I/O 存取。
2. 將 Trident 節點物件標記為 `dirty`（不適用於新出版物）。

> 註 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。

有關自動故障轉移的詳細信息

您可以透過與下列系統的整合來自動執行強制分離程序："節點健康檢查 (NHC) 操作符" 。當節點發生故障時，NHC 會透過在 Trident 的命名空間中建立 TridentNodeRemediation CR 來定義故障節點，從而觸發Trident節點修復 (TNR) 並自動強制分離。TNR 僅在節點發生故障時創建，並在節點恢復上線或節點被刪除後由 NHC 刪除。

節點 **Pod** 移除過程失敗

自動故障轉移會選擇要從故障節點移除的工作負載。建立 TNR 時，TNR 控制器會將節點標記為髒節點，阻止任何新的捲發布，並開始移除強制分離支援的 pod 及其磁碟區附件。

所有受強制分離支援的磁碟區/PVC均受自動故障轉移支援：

- NAS 和使用自動匯出策略的 NAS 經濟型磁碟區（尚未支援 SMB）。
- SAN 和 SAN 經濟型磁碟區。

參考[強制分離的詳細資料]。

預設行為：

- 使用 force-detach 支援的磁碟區的 Pod 將從故障節點中移除。Kubernetes 會將這些任務重新調度到健康的節點上。
- 使用不支援強制分離的磁碟區（包括非 Trident 磁碟區）的 Pod 不會從故障節點中移除。
- 無狀態 Pod（非 PVC）不會從故障節點中移除，除非 Pod 註解另有規定。 `trident.netapp.io/podRemediationPolicy: delete`已設定。

覆蓋 **pod** 移除行為：

可以使用 Pod 註解來自訂 Pod 移除行為： `trident.netapp.io/podRemediationPolicy[retain, delete]` 。發生故障轉移時，會檢查並使用這些註解。在 Kubernetes 部署/副本集 Pod 規格中加入註解，以防止故障轉移後註解消失：

- `retain`- 在自動故障轉移期間，Pod 不會從故障節點中移除。
- `delete`- 在自動故障轉移期間，Pod 將從故障節點中移除。

這些註解可以應用於任何 pod。

警告	• 只有當發生故障的節點支援強制分離時，I/O 操作才會被阻塞。 • 對於不支援強制分離的捲，存在資料損壞和多重附加問題的風險。

TridentNode修復CR

TridentNodeRemediation (TNR) CR 定義了一個故障節點。TNR 的名稱是故障節點的名稱。

TNR 範例：

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

TNR狀態：使用以下指令查看TNR狀態：
`kubectl get tnr <name> -n <trident-namespace>`

TNR（誘捕、絕育、放歸）可能處於下列幾種狀態之一：

- 修復中：
  - 停止對強制分離掛載到該節點的磁碟區的後端 I/O 存取。
  - Trident節點物件被標記為髒（不適合發布新內容）。
  - 從節點中移除 Pod 和磁碟區附件
- *NodeRecoveryPending*:
  - 控制器正在等待節點重新上線。
  - 一旦節點上線，發布強制執行將確保節點乾淨且已準備好發布新磁碟區。
- 如果節點從 K8s 中刪除，TNR 控制器將移除 TNR 並停止協調。
- 成功：
  - 所有修復和節點恢復步驟均已成功完成。該節點已清理完畢，可以發布新的捲。
- 失敗的：
  - 無法恢復的錯誤。錯誤原因設定在 CR 的狀態訊息欄位中。

啟用自動故障轉移

先決條件：

- 請確保在啟用自動故障轉移之前已啟用強制分離。更多信息，請參閱[強制分離的詳細資料]。
- 在 Kubernetes 叢集中安裝節點健康檢查 (NHC)。
  - "安裝 operator-sdk"。

- 如果叢集中尚未安裝 Operator Lifecycle Manager (OLM)，請安裝它： `operator-sdk olm install` 。
- 安裝節點健康檢查運算子： `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml` 。

註 | 您也可以使用以下[整合自訂節點健康檢查解決方案]章節中指定的其他方法來偵測節點故障。

看"節點健康檢查操作符"了解更多。

步驟

1. 在Trident命名空間中建立 NodeHealthCheck (NHC) CR，以監控叢集中的工作節點。範例：

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: <CR name>
spec:
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
        operator: DoesNotExist
  remediationTemplate:
    apiVersion: trident.netapp.io/v1
    kind: TridentNodeRemediationTemplate
    namespace: <Trident installation namespace>
    name: trident-node-remediation-template
  minHealthy: 0 # Trigger force-detach upon one or more node failures
  unhealthyConditions:
    - type: Ready
      status: "False"
      duration: 0s
    - type: Ready
      status: Unknown
      duration: 0s
```

2. 在節點健康檢查 CR 中應用 `trident`命名空間。

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

上述 CR 配置為監控 K8s 工作節點，以偵測節點狀態 Ready: false 和 Unknown。當節點進入 Ready: false 或 Ready: Unknown 狀態時，將觸發自動故障轉移。

這 `unhealthyConditions`CR 中使用了 0 秒寬限期。這樣一來，一旦 K8s 將節點狀態 Ready: false 設定為 false（在 K8s 失去節點的心跳訊號後設定），就會立即觸發自動故障轉移。K8s 在最後一次心跳後預設等待 40 秒，

然後才將 Ready: false 設為 false。此寬限期可在 K8s 部署選項中進行自訂。

有關其他配置選項，請參閱"節點健康檢查操作符文檔"。

其他設定訊息

當Trident安裝時啟用了強制分離功能， Trident命名空間中會自動建立兩個額外的資源，以方便與 NHC 整合：TridentNodeRemediationTemplate (TNRT) 和 ClusterRole。

TridentNodeRemediationTemplate (TNRT)：

TNRT 可作為 NHC 控制器的模板，NHC 控制器可依需求使用 TNRT 產生 TNR 資源。

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
    spec: {}
```

集群角色：

啟用強制分離功能時，安裝過程中也會新增叢集角色。這使得 NHC 能夠對Trident命名空間中的 TNR 進行授權。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac.ext-remediation/aggregate-to-ext-remediation: "true"
  name: tridentnoderemediation-access
rules:
- apiGroups:
  - trident.netapp.io
  resources:
  - tridentnoderemediationtemplates
  - tridentnoderemediations
  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
```

```
  - delete
```

K8s叢集升級與維護

為防止故障轉移，在 K8s 維護或升級期間暫停自動故障轉移，因為預計節點會停止運作或重新啟動。您可以透過修改 NHC CR 的補丁來暫停該 CR（如上所述）：

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

這將暫停自動故障轉移。若要重新啟用自動故障轉移，維護完成後，請從規格中刪除 pauseRequests。

限制

- 僅對受強制分離支援的捲，在發生故障的節點上阻止 I/O 操作。只有使用強制分離支援的磁碟區/PVC的pod才會自動移除。
- 自動故障轉移和強制分離在三叉戟控制器艙內運作。如果託管 trident-controller 的節點發生故障，自動故障轉移將會延遲，直到 K8s 將 pod 遷移到健康的節點。

整合自訂節點健康檢查解決方案

您可以將節點健康檢查操作器替換為其他節點故障偵測工具，以觸發自動故障轉移。為確保與自動故障轉移機制相容，您的自訂解決方案應：

- 當偵測到節點故障時，建立 TNR，使用故障節點的名稱作為 TNR CR 名稱。
- 當節點恢復且 TNR 處於成功狀態時，刪除 TNR。

安全性

Trident 安全性建議

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。

> 註 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 。
  - 。您可以將 NVE 加密旗標設定為 `""`建立啟用 NAE 的磁碟區。
- 如果後端未啟用 NAE ，則除非在後端組態中將 NVE 加密旗標設為（預設值），否則在 Trident 中配置的任何 Volume 都將啟用 NVE `false` 。

> 註
>
> 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"

Trident Linux 統一金鑰設定 （LUKS）

Linux 統一金鑰設定 (LUKS) 是一種磁碟加密規範。Trident 使用 LUKS 為 ONTAP SAN 和 ONTAP SAN ECONOMY Volume 提供每個 Volume 的主機端加密，並支援密碼片語輪替與 Volume 擴充。

在 Trident 中， LUKS 加密的磁碟區使用 AES-XTS-plain64 cypher 和模式"NIST"，如所建議。

> 註 ASA r2 系統不支援 LUKS 加密。有關 ASA r2 系統的信息，請參閱"瞭解 ASA R2 儲存系統" 。

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"。
- 基於效能理由， NetApp 建議工作節點支援進階加密標準新指令（ 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
```

將 **LUKS** 與 **RWX** 原始區塊 **Volume** 一起使用

從 Trident 26.10 開始，您可以將 LUKS 加密與 iSCSI 支援的原始區塊 Volume 搭配使用，這些 Volume 使用 `ReadWriteMany` (RWX) 存取模式連接到多個 Kubernetes 節點。這對於需要在即時移轉期間將相同加密 Volume 連接到目的地節點，同時仍連接到來源節點的工作負載（例如 KubeVirt 虛擬機器）非常有用。

RWX LUKS 支援僅限於原始區塊 Volume。不支援檔案系統模式的 RWX LUKS Volume。應用程式負責協調對解密資料的並行存取；Trident 僅負責協調 LUKS 作業。

如 啟用LUKS加密 所述，在後端、StorageClass 和密碼 Secret 上設定 LUKS。然後使用 `volumeMode: Block` 和 `ReadWriteMany` 建立 PVC：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-rwx-pvc
  namespace: my-app
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  storageClassName: san-luks-rwx
  resources:
    requests:
      storage: 10Gi
```

多個節點上的 Pod 可以將此 PVC 作為原始區塊裝置附加。Trident 在裝置首次附件時使用 LUKS2 進行格式化，並在後續節點上開啟現有的加密 Volume。支援現有的 LUKS 操作，包括密碼輪換和 Volume 擴充。

為什麼 **Trident** 會對 **LUKS** 作業進行序列化

LUKS2 在區塊裝置的開頭儲存其加密標頭的備援副本。 `cryptsetup` 當偵測到副本存在差異時，即使在看似唯讀的命令執行期間，也能修理標頭。它的鎖定機制僅限於節點，無法協調其他節點上的作業。如果一個節點正在更新標頭，而另一個節點執行 `cryptsetup`，則第二個節點可能會觸發標頭修理並中斷更新，這可能會導致標頭毀損。

為了防止同時存取標頭， Trident 為每個 RWX LUKS Volume 使用 Kubernetes 租約。在節點對 Volume 執行任何 `cryptsetup`命令之前，必須持有該租約。Trident 在操作成功後會釋放租約。如果在持有租約期間有其他節點要求操作， Trident 將傳回可重試的錯誤， Kubernetes 稍後會重試。對不同 Volume 的操作是獨立的。

如果 `cryptsetup` 作業失敗，Trident 會故意保留租約，以便只有同一節點才能重試。帶有原因 `LeaseRetained` 的警告事件會記錄此故障。租約不會自動過期；系統管理員只有在確認其持有節點上沒有執行 `cryptsetup` 作業後，才應移除租約。如需恢復步驟，請參閱 "修正 RWX Linux 統一金鑰設定（LUKS）Volume 卡住的租約"。

由於節點輪流開啟 Volume，因此在許多不同節點上啟動工作負載所需的時間可能比使用未加密 Volume 更長。Kubernetes 會在目前作業完成後重試暫時性鎖定錯誤。在大規模橫向擴充期間，Pod 可能會停留在 `ContainerCreating`更長時間；同一個節點上的工作負載可以共享單一開啟的裝置對應。

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。

警告	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。

> 註 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。

修正 RWX Linux 統一金鑰設定（LUKS）Volume 卡住的租約

如果節點在持有其 Kubernetes 租約時發生故障，則 ReadWriteMany (RWX) Linux 統一金鑰設定 (LUKS) Volume 可能會被封鎖。刪除租約前，請確認沒有 `cryptsetup`作業仍在執行；在作業期間將其刪除可能會使 LUKS 標頭面臨風險。

了解租約

Trident 使用 Kubernetes 租約來確保同一時間只有一個節點可以 `cryptsetup`對 RWX LUKS Volume 執行命令。租約在 Trident 命名空間中建立，租約名稱即為 PersistentVolume（PV）名稱。持有者身分用於識別執行作業的節點。

Trident 會在作業成功後刪除租約。如果作業失敗， Trident 會保留租約，以便只有同一個節點可以重試。租約不會自動過期或續約。帶有原因的警告事件 `LeaseRetained`包含有關失敗作業的詳細資訊。

識別可能卡住的租約

症狀可能包括 pod 卡在 `ContainerCreating`或 `Terminating`、Volume 擴充失敗，或暫時性鎖定錯誤。

1. 找到受影響的 pod 所使用的 PVC：

```
kubectl get pod -n <app-namespace> <pod-name> \
  -o jsonpath='{.spec.volumes[*].persistentVolumeClaim.claimName}'
```

2. 找出繫結至 PVC 的 PV：

```
kubectl get pvc -n <app-namespace> <pvc-name> \
  -o jsonpath='{.spec.volumeName}'
```

3. 列出 Trident 命名空間中的租約，並識別具有 PV 名稱的租約：

```
kubectl get lease -n <trident-namespace>
```

4. 檢查相符的租約，包括其持有人、存在時間和事件：

```
kubectl describe lease -n <trident-namespace> <pv-name>
```

`cryptsetup` 操作通常會在幾秒鐘內完成。超過 5 到 10 分鐘的租約可能表示發生死結。檢查 `LeaseRetained` 事件是否有重複的錯誤，識別持有者節點，並檢查節點健全狀況：

```
kubectl get nodes
kubectl get pods -A -o wide --field-selector spec.nodeName=<lease-holder-
node>
```

如果節點運作正常且租約沒有警告事件，請在斷定租約卡住之前，檢查持有節點上的 Trident 節點 pod 記錄：

```
kubectl get pods -n <trident-namespace> -o wide \
  --field-selector spec.nodeName=<lease-holder-node>
kubectl logs -n <trident-namespace> <trident-node-pod> -c trident-main
```

如果底層節點或網路問題可以修復，請先進行修復。當 Trident 成功完成作業時，Lease 即可解除。

安全解決卡住的租約

重要	請勿在租約持有者可能仍在執行 `cryptsetup`操作時刪除租約。來自不同節點且未由租約協調的並行 `cryptsetup`命令可能會毀損 LUKS 標頭。

使用與節點狀況相符的修復方案：

- 持有者節點已發生故障並已斷電： 識別該節點持有的租約，並刪除受影響的租約。如果節點在其工作負載重新排程之前恢復，請允許 Trident 完成其作業並版本租約。如果工作負載已移動，請等待其在已復原節點上的 Pod 完全終止，確認死結仍然存在，然後移除過期的租約。
- 持有者節點已失去控制平面連線：該節點可能仍在執行工作負載。建議在刪除租約之前先關閉該節點。如果該節點仍然可以存取，請停止該節點上的 Trident 節點 Container ，或從其記錄中確認它無法取得或刪除租約且未執行 `cryptsetup`。Trident 需要存取 Kubernetes API 才能在每次操作前取得租約；連線遺失後，它將無法啟動其他操作。在考慮刪除之前，請從建立租約起至少等待五分鐘，以便任何正在進行的命令都能完成。
- 持有節點和網路運作正常： 檢查租約警告事件和 Trident 節點日誌，以確定故障操作。如果需要將受影響的工作負載移動到另一個節點，請隔離持有節點，刪除受影響的應用程式 Pod （不要強制刪除），等待它們終止，然後解除隔離節點。只有在確認原始操作不再執行後，才能刪除租約。

只有在確認安全無虞後，才能刪除租約：

```
kubectl delete lease -n <trident-namespace> <pv-name>
```

修復後，監控受影響的 Pod 和 Volume 操作。如果租約再次保留，請調查租約事件中報告的 cryptsetup 錯誤並解決根本問題。

Trident Kerberos 傳輸中加密

使用 Kerberos 在線上加密，您可以針對託管叢集與儲存後端之間的流量啟用加密，藉此改善資料存取安全性。

Trident 支援 ONTAP 的 Kerberos 加密作為儲存後端：

- 內部部署 **ONTAP** - Trident 支援透過 NFSv3 和 NFSv4 連線，從 Red Hat OpenShift 和上游 Kubernetes 叢集到內部部署 ONTAP 磁碟區的 Kerberos 加密。

您可以建立、刪除、調整大小、快照、複製、 唯讀複製及匯入使用 NFS 加密的磁碟區。

使用內部部署的 **ONTAP** 磁碟區來設定在線上 **Kerberos** 加密

您可以在託管叢集與內部部署 ONTAP 儲存後端之間的儲存流量上啟用 Kerberos 加密。

註	內部部署 ONTAP 儲存後端的 NFS 流量 Kerberos 加密僅支援使用 `ontap-nas`儲存驅動程式。

開始之前

- 請確定您可以存取 `tridentctl` 公用程式。
- 確保您具有 ONTAP 儲存後端的管理員存取權。

- 確保您知道將從 ONTAP 儲存後端共用的磁碟區名稱。
- 請確定您已準備好 ONTAP 儲存 VM 、以支援 NFS 磁碟區的 Kerberos 加密。請參閱 "在 dataLIF 上啟用 Kerberos" 以取得指示。
- 請確保所有與 Kerberos 加密一起使用的 NFSv4 磁碟區都已正確設定。請參閱 NetApp NFSv4 域配置部分（第 13 頁） "NetApp NFSv4 增強功能與最佳實務指南"。

新增或修改 **ONTAP** 匯出原則

您需要將規則新增至現有的 ONTAP 匯出原則、或建立新的匯出原則、以支援 ONTAP 儲存 VM 根磁碟區的 Kerberos 加密、以及與上游 Kubernetes 叢集共用的任何 ONTAP 磁碟區。您新增的匯出原則規則或您建立的新匯出原則需要支援下列存取通訊協定和存取權限：

存取傳輸協定

使用 NFS、NFSv3 和 NFSv4 存取協定設定匯出原則。

存取詳細資料

您可以根據對磁碟區的需求、設定 Kerberos 加密的三個不同版本之一：

- * Kerberos 5* - （驗證與加密）
- * Kerberos 5i* - （身分識別保護的驗證與加密）
- * Kerberos 5p* - （身分識別與隱私保護的驗證與加密）

使用適當的存取權限來設定 ONTAP 匯出原則規則。例如、如果叢集將使用 Kerberos 5i 和 Kerberos 5p 加密混合安裝 NFS 磁碟區、請使用下列存取設定：

類型	唯讀存取	讀取 / 寫入存取權	超級使用者存取權
UNIX	已啟用	已啟用	已啟用
Kerberos 5i	已啟用	已啟用	已啟用
Kerberos 5p	已啟用	已啟用	已啟用

請參閱下列文件、瞭解如何建立 ONTAP 匯出原則和匯出原則規則：

- "建立匯出原則"
- "新增規則至匯出原則"

建立儲存後端

您可以建立內含 Kerberos 加密功能的 Trident 儲存後端組態。

關於這項工作

當您建立設定 Kerberos 加密的儲存後端組態檔時、可以使用參數指定 Kerberos 加密的三個不同版本之一 `spec.nfsMountOptions` ：

- `spec.nfsMountOptions: sec=krb5` （驗證與加密）
- `spec.nfsMountOptions: sec=krb5i` （身分識別保護的驗證與加密）
- `spec.nfsMountOptions: sec=krb5p` （身分識別與隱私保護的驗證與加密）

只指定一個 Kerberos 層級。如果您在參數清單中指定多個 Kerberos 加密層級、則只會使用第一個選項。

步驟

1. 在託管叢集上、使用下列範例建立儲存後端組態檔案。以您環境的資訊取代括弧 <> 中的值：

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. 使用您在上一個步驟中建立的組態檔來建立後端：

```
tridentctl create backend -f <backend-configuration-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。

您可以建立儲存類別、以使用 Kerberos 加密來配置磁碟區。

關於這項工作

當您建立儲存類別物件時、可以使用下列參數、指定 Kerberos 加密的三個不同版本之一 `mountOptions` ：

- `mountOptions: sec=krb5` （驗證與加密）
- `mountOptions: sec=krb5i` （身分識別保護的驗證與加密）
- `mountOptions: sec=krb5p` （身分識別與隱私保護的驗證與加密）

只指定一個 Kerberos 層級。如果您在參數清單中指定多個 Kerberos 加密層級、則只會使用第一個選項。如果您在儲存後端組態中指定的加密層級與您在儲存類別物件中指定的層級不同、則儲存類別物件會優先。

步驟

1. 使用以下範例建立 StorageClass Kubernetes 物件：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions:
  - sec=krb5i #can be krb5, krb5i, or krb5p
parameters:
  backendType: ontap-nas
  storagePools: ontapnas_pool
  trident.netapp.io/nasType: nfs
allowVolumeExpansion: true
```

2. 建立儲存類別：

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. 確定已建立儲存類別：

```
kubectl get sc ontap-nas-sc
```

您應該會看到類似下列的輸出：

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

配置 Volume

建立儲存後端和儲存類別之後、您現在可以配置 Volume 。有關說明，請參閱 "配置 Volume"。

使用 Azure NetApp Files 磁碟區設定在線上 Kerberos 加密

您可以在託管叢集與單一 Azure NetApp Files 儲存後端或 Azure NetApp Files 儲存後端的虛擬集區之間的儲存流量上啟用 Kerberos 加密。

開始之前

- 確保您已在託管的 Red Hat OpenShift 叢集上啟用 Trident 。
- 請確定您可以存取 `tridentctl` 公用程式。
- 請注意中的要求並遵循中的指示、以確保您已準備好 Azure NetApp Files 儲存後端進行 Kerberos 加密 "本文檔 Azure NetApp Files"。
- 請確保所有與 Kerberos 加密一起使用的 NFSv4 磁碟區都已正確設定。請參閱 NetApp NFSv4 域配置部分（第 13 頁） "NetApp NFSv4 增強功能與最佳實務指南"。

建立儲存後端

您可以建立包含 Kerberos 加密功能的 Azure NetApp Files 儲存後端組態。

關於這項工作

當您建立儲存後端組態檔案來設定 Kerberos 加密時、您可以加以定義、以便將其套用至下列兩種可能的層級之一：

- 使用欄位的 * 儲存後端層級 * `spec.kerberos`
- 使用欄位的 * 虛擬集區層級 * `spec.storage.kerberos`

當您在虛擬集區層級定義組態時、會使用儲存類別中的標籤來選取集區。

在任一層級、您都可以指定 Kerberos 加密的三個不同版本之一：

- `kerberos: sec=krb5` （驗證與加密）
- `kerberos: sec=krb5i` （身分識別保護的驗證與加密）
- `kerberos: sec=krb5p` （身分識別與隱私保護的驗證與加密）

步驟

1. 在託管叢集上、根據您需要定義儲存後端（儲存後端層級或虛擬集區層級）的位置、使用下列其中一個範例建立儲存後端組態檔案。以您環境的資訊取代括弧 <> 中的值：

   儲存後端層級範例

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: backend-tbc-secret
   type: Opaque
   ```

```
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

虛擬集區層級範例

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
```

```
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  storage:
    - labels:
        type: encryption
      kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

2. 使用您在上一個步驟中建立的組態檔來建立後端：

```
tridentctl create backend -f <backend-configuration-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。

建立儲存類別

您可以建立儲存類別、以使用 Kerberos 加密來配置磁碟區。

步驟

1. 使用以下範例建立 StorageClass Kubernetes 物件：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: azure-netapp-files
```

```
  trident.netapp.io/nasType: nfs
  selector: type=encryption
```

2. 建立儲存類別：

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. 確定已建立儲存類別：

```
kubectl get sc -sc-nfs
```

您應該會看到類似下列的輸出：

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

配置 Volume

建立儲存後端和儲存類別之後、您現在可以配置 Volume 。有關說明，請參閱 "配置 Volume"。

使用Trident Protect 保護應用程式

了解Trident Protect

NetApp Trident Protect 提供進階應用程式資料管理功能，增強了由NetApp ONTAP儲存系統和NetApp Trident CSI 儲存供應器支援的有狀態 Kubernetes 應用程式的功能和可用性。Trident Protect 簡化了跨公有雲和本地環境的容器化工作負載的管理、保護和遷移。它還透過其 API 和 CLI 提供自動化功能。

您可以透過建立自訂資源 (CR) 或使用Trident Protect CLI 來使用Trident Protect 保護應用程式。

接下來呢？

您可以先了解Trident Protect 的相關需求，然後再進行安裝：

- "Trident保護要求"

安裝Trident Protect

Trident保護要求

首先，請驗證您的運行環境、應用程式叢集、應用程式和授權是否已準備就緒。確保您的環境符合部署和執行Trident Protect 的這些要求。

Trident Protect Kubernetes 叢集相容性

Trident Protect 與各種完全託管和自架的 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 Harvester 1.7.0 （ONTAP iSCSI）
- SUSE Rancher
- VMware Tanzu產品組合
- 上游Kubernetes

註

- Trident Protect備份僅支援Linux運算節點。Windows 運算節點不支援備份作業。
- 確保安裝Trident Protect 的叢集已配置正在執行中的快照控制器和相關的 CRD。若要安裝快照控制器，請參閱 "這些指示"。
- 確保至少存在一個 VolumeSnapshotClass。如需詳細資訊、請參閱 "Volume SnapshotClass"。

- 安裝 Trident Protect 需要 Helm 4.x 或更高版本。

Trident Protect 儲存後端相容性

Trident Protect 支援以下儲存後端：

- 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：

- 確保連接到叢集的 NetApp 儲存裝置使用 Trident 24.02 或更新版本（建議使用 Trident 24.10）。
- 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。
- 建立您計劃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了不存在的命名空間，則操作將會失敗。

NAS 經濟容量需求

Trident Protect 支援對 nas-economy 磁碟區進行備份和復原作業。目前不支援將快照、克隆和SnapMirror複製到 nas-economy 磁碟區。您需要為計劃與Trident Protect 一起使用的每個 nas-economy 磁碟區啟用快照目錄。

註

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
```

註

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。

使用 **KubeVirt VM** 保護資料

Trident Protect 在資料保護作業期間為 KubeVirt 虛擬機器提供檔案系統凍結和解凍功能，以確保資料一致性。虛擬機器凍結操作的配置方法和預設行為在Trident Protect 的不同版本中有所不同，較新的版本透過 Helm chart 參數提供了簡化的配置。

註	在恢復操作期間，任何 `VirtualMachineSnapshots`為虛擬機器 (VM) 所建立的資料不會被復原。

▼ **Trident Protect 25.10** 及更新版本

Trident Protect 在資料保護作業期間自動凍結和解凍 KubeVirt 檔案系統，以確保一致性。從Trident Protect 25.10 開始，您可以使用以下方法停用此行為： `vm.freeze` Helm Chart 安裝過程中的參數。此參數預設啟用。

```
helm install ... --set vm.freeze=false ...
```

▼ **Trident Protect 24.10.1** 至 **25.06**

從Trident Protect 24.10.1 開始， Trident Protect 會在資料保護作業期間自動凍結和解凍 KubeVirt 檔案系統。您也可以使用以下命令停用此自動行為：

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

▼ **Trident Protect 24.10**

Trident Protect 24.10 在資料保護作業期間不會自動確保 KubeVirt VM 檔案系統的一致性狀態。如果您想使用Trident Protect 24.10 保護您的 KubeVirt VM 數據，則需要在執行資料保護作業之前手動啟用檔案系統的凍結/解凍功能。這樣可以確保檔案系統處於一致狀態。

您可以設定 Trident Protect 24.10 來管理資料保護作業期間 VM 檔案系統的凍結和解凍，方法是"設定虛擬化"然後使用下列指令：

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

若要了解如何使用 Trident Protect 定義應用程式和還原 VM，請參閱"使用 Trident Protect 保護 KubeVirt 虛擬機器"。

SnapMirror 複寫需求

NetApp SnapMirror複製功能可與Trident Protect 搭配使用，適用於下列ONTAP解決方案：

- 本地端 NetApp FAS、AFF 和 ASA 系統。目前 ASA r2 系統尚不支援使用 Trident protect 的 SnapMirror 複寫。
- 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：

- * NetApp Trident *：使用ONTAP作為後端服務的來源 Kubernetes 叢集和目標 Kubernetes 叢集上都必須存在NetApp Trident 。Trident Protect 支援使用NetApp SnapMirror技術進行複製，該技術使用以下驅動程式支援的儲存類別：
  - `ontap-nas` ：NFS
  - `ontap-san` ：iSCSI
  - `ontap-san` ：足球俱樂部
  - `ontap-san` ：NVMe/TCP（要求最低 ONTAP 版本 9.15.1）
- * 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" 。

  從 ONTAP 9.10.1 開始、所有授權都會以 NetApp 授權檔案（ NLF ）的形式交付、這是一個可啟用多項功能的單一檔案。如需詳細資訊、請參閱 "ONTAP One 隨附授權" 。

註	僅支援 SnapMirror 非同步保護。

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" 。

註	確保兩個 ONTAP 叢集之間複寫關係中使用的 SVM 名稱是唯一的。

- **NetApp Trident 與 SVM**：對等遠端 SVM 必須可供目標叢集上的 NetApp Trident 使用。
- 託管後端：您需要在Trident Protect 中新增和管理ONTAP儲存後端，以建立複製關係。

用於 SnapMirror 複寫的 Trident / ONTAP 組態

Trident Protect 要求您至少設定一個支援來源叢集和目標叢集複製的儲存後端。如果來源叢集和目標叢集相同，為了獲得最佳彈性，目標應用程式應該使用與來源應用程式不同的儲存後端。

SnapMirror複製的 Kubernetes 叢集要求

確保您的 Kubernetes 叢集符合以下要求：

- **AppVault** 可存取性：來源叢集和目標叢集都必須具有網路存取權限，才能從 AppVault 讀取和寫入應用程式物件複製。
- 網路連線：設定防火牆規則、儲存桶權限和 IP 允許列表，以實現跨 WAN 的叢集和 AppVault 之間的通訊。

註	許多企業環境在 WAN 連線中實施嚴格的防火牆策略。在配置複製之前，請與您的基礎設施團隊驗證這些網路需求。

安裝並設定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，或從您自己的私人註冊表中安裝它。如果您的叢集無法存取互聯網，從私有註冊表安裝會很有幫助。

安裝Trident Protect

從NetApp安裝Trident Protect

步驟

1. 新增Trident Helm儲存庫：

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

2. 使用 Helm 安裝Trident Protect。代替 `<name-of-cluster>`集群名稱將分配給集群，並用於標識集群的備份和快照：

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2610.0 --create
-namespace --namespace trident-protect
```

3. （選用）若要啟用偵錯日誌記錄（建議用於故障排除），請使用：

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2610.0 --create-namespace --namespace trident-protect
```

偵錯日誌記錄有助於NetApp支援人員排除故障，而無需變更日誌等級或重現問題。

從私人註冊表安裝Trident Protect

如果您的 Kubernetes 叢集無法存取互聯網，您可以從私人鏡像倉庫安裝Trident Protect。在這些範例中，請將括號中的值替換為您環境中的資訊：

步驟

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：

```
docker.io/netapp/controller:26.10.1
docker.io/netapp/trident-protect-utils:v6.0.0
docker.io/netapp/trident-protect-guestfs:v2.0.0
docker.io/netapp/kopiablockrestore:26.10.1
docker.io/netapp/trident-autosupport:26.10.1
docker.io/netapp/exechook:26.10.1
docker.io/netapp/resourcebackup:26.10.1
docker.io/netapp/resourcerestore:26.10.1
```

```
docker.io/netapp/resourcedelete:26.10.1
docker.io/netapp/filebrowse-transfer:26.10.1
```

例如：

```
docker pull docker.io/netapp/controller:26.10.1
```

```
docker tag docker.io/netapp/controller:26.10.1 <private-registry-url>/controller:26.10.1
```

```
docker push <private-registry-url>/controller:26.10.1
```

> 註 若要取得 Helm chart，請先在可存取互聯網的機器上使用 `helm pull trident-protect --version 100.2610.0 --repo https://netapp.github.io/trident-protect-helm-chart`下載 Helm chart，然後將產生的 `trident-protect-100.2610.0.tgz`檔案複製到您的離線環境中，並在最後一步中使用 `helm install trident-protect ./trident-protect-100.2610.0.tgz`而不是儲存庫引用進行安裝。

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. 建立一個名為的文件 `protectValues.yaml`。請確保其中包含以下Trident Protect 設定：

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

> 註 這 `imageRegistry`和 `imagePullSecrets`這些值適用於所有組件影像，包括 `resourcebackup`和 `resourcerestore`。如果您將鏡像推送到註冊表中的特定儲存庫路徑（例如， `example.com:443/my-repo`），請在登錄欄位中包含完整路徑。這將確保所有圖像都從此處提取。 `<private-registry-url>/<image-name>:<tag>`。

7. 使用 Helm 安裝Trident Protect。代替 `<name_of_cluster>`集群名稱將分配給集群，並用於標識集群的備份和快照：

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2610.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. （選用）若要啟用偵錯日誌記錄（建議用於故障排除），請使用：

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2610.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

偵錯日誌記錄有助於NetApp支援人員排除故障，而無需變更日誌等級或重現問題。

> 註 有關其他 Helm Chart 設定選項，包括AutoSupport設定和命名空間過濾，請參閱 "自訂Trident Protect 安裝"。

安裝Trident Protect CLI 插件

您可以使用Trident Protect 命令列插件，它是Trident的一個擴充。 `tridentctl`用於建立和與Trident Protect 自訂資源 (CR) 互動的實用程式。

安裝Trident Protect CLI 插件

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

步驟

1. 下載Trident Protect CLI 外掛：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-linux-amd64
```

下載適用於 **Linux ARM64 CPU** 的外掛程式

步驟

1. 下載Trident Protect CLI 外掛：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-linux-arm64
```

下載適用於 **Mac AMD64 CPU** 的外掛程式

步驟

1. 下載Trident Protect CLI 外掛：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-macos-amd64
```

下載 **Mac ARM64 CPU** 的外掛程式

步驟

1. 下載Trident Protect CLI 外掛：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-macos-arm64
```

1. 啟用外掛程式二進位檔的執行權限：

```
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. 您也可以選擇將外掛程式二進位檔複製到主目錄中的某個位置。在這種情況下，建議您確保位置是 PATH 變數的一部分：

```
cp ./tridentctl-protect ~/bin/
```

註 將外掛程式複製到 PATH 變數中的某個位置，可讓您輸入或 `tridentctl protect`從任何位置使用外掛程式 `tridentctl-protect`。

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 外掛程式後，您可以為某些指令啟用自動補全功能。

啟用 **Bash Shell** 的自動完成功能

步驟

1. 建立完成腳本：

```
tridentctl-protect completion bash > 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

步驟

1. 建立完成腳本：

```
tridentctl-protect completion zsh > 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，您可以將命令自動完成功能與 tridentctl-Protect 外掛程式搭配使用。

自訂**Trident Protect** 安裝

您可以自訂Trident Protect 的預設配置，以符合您環境的特定要求。

指定**Trident Protect** 容器資源限制

安裝Trident Protect 後，您可以使用設定檔來指定Trident Protect 容器的資源限制。設定資源限制可以控制Trident Protect 作業消耗叢集資源的程度。

步驟

1. 建立名為的檔案 `resourceLimits.yaml`。
2. 根據您的環境需求，在檔案中填入Trident Protect 容器的資源限制選項。

   以下範例組態檔顯示可用的設定，並包含每個資源限制的預設值：

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
```

```
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. 套用檔案中的值 `resourceLimits.yaml`：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

自訂安全性內容限制

安裝Trident Protect 後，您可以使用設定檔來修改Trident Protect 容器的 OpenShift 安全上下文約束 (SCC)。這些約束定義了 Red Hat OpenShift 叢集中 pod 的安全性限制。

步驟

1. 建立名為的檔案 `sccconfig.yaml`。
2. 將 SCC 選項新增至檔案，並根據環境需求修改參數。

   以下範例顯示 SCC 選項參數的預設值：

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

   下表說明 SCC 選項的參數：

參數	說明	預設
建立	決定是否可以建立 SCC 資源。只有在設定為 `true`且 `Helm` 安裝程序識別 `OpenShift` 環境時，才會建立 `SCC` 資源 `scc.create`。如果未在 OpenShift 上操作，或如果設為 `false`，則 `scc.create`不會建立任何 SCC 資源。	是的
名稱	指定SCC的名稱。	Trident 保護工作
優先順序	定義 SCC 的優先順序。優先順序值較高的 SCC 會在值較低之前進行評估。	1

3. 套用檔案中的值 `sccconfig.yaml`：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

   這會將預設值取代為檔案中指定的值 `sccconfig.yaml`。

設定其他**Trident Protect** 舵圖設置

您可以自訂 AutoSupport 設定、命名空間篩選和 OpenShift Webhook 憑證管理，以滿足您的特定需求。下表說明可用的組態參數：

參數	類型	說明
自動支援代理	字串	為NetApp AutoSupport連線配置代理 URL。使用此功能透過代理伺服器路由支援包上傳。例子：`http://my.proxy.url` 。
自動支援.不安全	布林值	設定為時跳過AutoSupport代理連線的 TLS 驗證 `true`。僅用於不安全的代理連線。 (預設: `false`)
自動支援已啟用	布林值	啟用或停用每日Trident Protect AutoSupport組合包上傳。設定為 `false`每日定時上傳功能已停用，但您仍可以手動產生支援包。（預設： `true` ）
恢復跳過命名空間註釋	字串	若要從備份和復原作業中排除的命名空間註解的逗號分隔清單。允許您根據註釋過濾命名空間。
restoreSkipNamespaceLabels	字串	若要從備份和復原作業中排除的命名空間標籤的逗號分隔清單。允許您根據標籤過濾命名空間。
useOpenshiftServiceCA	布林值	啟用與 OpenShift 原生 Service CA Operator 的整合，以便為 Trident Protect 准入 Webhook 自動建立和注入 TLS 憑證。設為 `true`時，`Service CA Operator` 將自動管理 `Webhook TLS` 憑證。僅在 `OpenShift` 叢集上使用。（預設值： `false`）

若要加密快照和備份中繼資料，請參閱"加密 Trident Protect 快照與備份的 AppVault 中繼資料"。

您可以使用 YAML 設定檔或命令列標誌來設定這些選項：

使用 **YAML** 文件

步驟

1. 建立設定檔並命名 `values.yaml`。
2. 在您建立的文件中，新增您想要自訂的設定選項。

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
```

```
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
useOpenshiftServiceCA: true
```

3. 填充後 `values.yaml`具有正確值的文件，應用設定檔：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

使用 **CLI** 標誌

步驟

1. 使用以下命令 `--set`標誌來指定單一參數：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --set useOpenshiftServiceCA=true \
  --reuse-values
```

將**Trident Protect Pod** 限制在特定節點上

您可以使用 Kubernetes nodeSelector 節點來選擇約束，根據節點標籤來控制哪些節點有資格執行Trident Protect pod。預設情況下， Trident Protect 僅限於執行 Linux 的節點。您可以根據需要進一步自訂這些限制條件。

步驟

1. 建立名為的檔案 `nodeSelectorConfig.yaml`。
2. 將 nodeSelector 選項新增至檔案，並修改檔案以新增或變更節點標籤，以根據環境需求加以限制。例如，下列檔案包含預設的作業系統限制，但也針對特定區域和應用程式名稱：

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. 套用檔案中的值 `nodeSelectorConfig.yaml`：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

這會將預設限制取代為您在檔案中指定的限制 `nodeSelectorConfig.yaml`。

管理Trident Protect

管理Trident Protect 授權和存取控制

Trident Protect 使用 Kubernetes 的角色為基礎的存取控制 (RBAC) 模型。預設情況下，Trident Protect 提供一個系統命名空間及其關聯的預設服務帳戶。如果您的組織擁有眾多使用者或特定的安全需求，則可以使用Trident Protect 的 RBAC 功能來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 ）。

註

只有管理員才能存取具有特權的Trident Protect 自訂資源對象，其中包括：

- * AppVault* ：需要儲存庫認證資料
- **AutoSupportBundle**：收集指標、日誌和其他敏感的Trident Protect數據
- * 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 時會建立一個預設的 ClusterRole 物件。您可以透過建立和套用 RoleBinding 物件將此 ClusterRole 綁定到服務帳戶。

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。

監控Trident保護資源

您可以使用 kube-state-metrics、Prometheus 和 Alertmanager 開源工具來監控Trident Protect 保護的資源的健康狀況。

kube-state-metrics 服務從 Kubernetes API 通訊產生指標。將其與Trident Protect 結合使用，可以顯示有關環境中資源狀態的有用資訊。

Prometheus 是一個工具包，它可以接收 kube-state-metrics 產生的數據，並將其呈現為關於這些物件的易於閱讀的資訊。kube-state-metrics 和 Prometheus 共同提供了一種方法，讓您可以監控使用Trident Protect 管理的資源的健康狀況和狀態。

AlertManager 是一項服務，可擷取 Prometheus 等工具所傳送的警示，並將其路由至您設定的目的地。

註

這些步驟所包含的組態和指南僅為範例，您需要自訂以符合您的環境。請參閱下列正式文件，以取得特定指示與支援：

- "Kube-state指標文件"
- "Prometheus 文件"
- "AlertManager 文件"

步驟 1 ：安裝監控工具

要在Trident Protect 中啟用資源監控，您需要安裝和設定 kube-state-metrics、Promethus 和 Alertmanager。

安裝 kube 狀態度量

您可以使用 Helm 來安裝 kube 狀態度量。

步驟

1. 新增 kube 狀態指標 Helm 圖表。例如：

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. 將 Prometheus ServiceMonitor CRD 應用到叢集：

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. 為 Helm 圖表建立組態檔（例如 `metrics-config.yaml`）。您可以自訂下列範例組態，以符合您的環境：

測量數據 **-config.yaml** ： **Kube-state** 度量 **Helm** 圖表組態

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: protect.trident.netapp.io
          kind: "Backup"
          version: "v1"
        labelsFromPath:
          backup_uid: [metadata, uid]
          backup_name: [metadata, name]
          creation_time: [metadata, creationTimestamp]
        metrics:
        - name: backup_info
          help: "Exposes details about the Backup state"
          each:
            type: Info
            info:
              labelsFromPath:
                appVaultReference: ["spec", "appVaultRef"]
                appReference: ["spec", "applicationRef"]
rbac:
  extraRules:
  - apiGroups: ["protect.trident.netapp.io"]
    resources: ["backups"]
    verbs: ["list", "watch"]

# Collect metrics from all namespaces
namespaces: ""

# Ensure that the metrics are collected by Prometheus
prometheus:
  monitor:
    enabled: true
```

4. 部署 Helm 圖表以安裝 kube 狀態度量。例如：

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. 請依照下列說明配置 kube-state-metrics，以產生Trident Protect 使用的自訂資源的指標： "Kube-state 度量自訂資源文件" 。

安裝Prometheus

您可以依照中的指示來安裝 Prometheus "Prometheus 文件" 。

安裝 AlertManager

您可以依照中的指示安裝 AlertManager "AlertManager 文件" 。

步驟 2 ：設定監控工具以共同作業

安裝監控工具之後，您需要將它們設定為一起運作。

步驟

1. 將 kube 狀態指標與 Prometheus 整合。編輯 Prometheus 配置文件(`prometheus.yaml`）並添加 kube 狀態指標服務信息。例如：

prometheus.yaml：kube-state-metrics 服務與 Prometheus 的集成

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. 設定 Prometheus 將警示路由至 AlertManager 。編輯 Prometheus 配置文件(`prometheus.yaml`）並添加以下部分：

prometheus.yaml：向 Alertmanager 發送警報

```
alerting:
  alertmanagers:
    - static_configs:
```

```
        - targets:
            - alertmanager.trident-protect.svc:9093
```

結果

現在， Prometheus 可以從 kube-state 度量收集度量，並可傳送警示給 Alertmanager 。您現在已準備好設定觸發警示的條件，以及應傳送警示的位置。

步驟 **3** ：設定警示和警示目的地

設定工具以共同作業之後，您需要設定觸發警示的資訊類型，以及應傳送警示的位置。

警示範例：備份失敗

以下範例定義當備份自訂資源的狀態設定為 5 秒或更長時間時觸發的關鍵警示 `Error`。您可以自訂此範例以符合您的環境，並將此 YAML 片段包含在組態檔案中 `prometheus.yaml`：

rules.yaml：定義失敗備份的 **Prometheus** 警報

```
rules.yaml: |
  groups:
    - name: fail-backup
        rules:
          - alert: BackupFailed
            expr: kube_customresource_backup_info{status="Error"}
            for: 5s
            labels:
              severity: critical
            annotations:
              summary: "Backup failed"
              description: "A backup has failed."
```

設定 **AlertManager** 以傳送警示至其他頻道

您可以將 AlertManager 設定為傳送通知給其他通道，例如電子郵件， PagerDuty ， Microsoft 團隊或其他通知服務，方法是在檔案中指定個別的組態 `alertmanager.yaml`。

以下範例將警示管理員設定為傳送通知至 Slack 頻道。若要根據您的環境自訂此範例，請將金鑰的值取代為 `api_url`您環境中使用的 Slack Webhook URL ：

alertmanager.yaml：向 **Slack** 頻道發送警報

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
```

```
receivers:
  - name: 'slack-notifications'
    slack_configs:
      - api_url: '<your-slack-webhook-url>'
        channel: '#failed-backups-channel'
        send_resolved: false
```

產生Trident Protect 支援包

Trident Protect 使管理員能夠產生包含對NetApp支援有用的信息的捆綁包，包括有關受管理叢集和應用程式的日誌、指標和拓撲資訊。如果您已連接到互聯網，則可以使用自訂資源 (CR) 檔案將支援包上傳至NetApp支援網站 (NSS)。

使用 **CR** 建立支援服務組合

步驟

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，則默認爲 `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. 填充後 `trident-protect-support-bundle.yaml`具有正確值的文件，應用 CR：

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-protect
```

使用 **CLI** 建立支援服務包

步驟

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> -n trident-protect
```

監視和檢索支援包

使用任一方法建立支援包後，您可以監視其產生進度並將其檢索到本機系統。

步驟

1. 等待 `status.generationState`到達 `Completed`狀態。您可以使用以下命令監控產生進度：

```
kubectl get autosupportbundle trident-protect-support-bundle -n trident-protect
```

2. 將支援包檢索到您的本機系統。從已完成的AutoSupport套件中取得複製指令：

```
kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
```

   找到 `kubectl cp`從輸出執行命令並運行它，用您喜歡的本機目錄取代目標參數。

升級**Trident**保護

您可以將Trident Protect 升級到最新版本，以享受新功能或修復錯誤。

註

- 從版本 24.10 升級時，升級期間執行的快照可能會失敗。此失敗不會阻止將來建立快照（無論是手動快照還是計劃快照）。如果升級期間快照失敗，您可以手動建立新快照以確保應用程式受到保護。

為避免潛在的故障，您可以在升級前停用所有快照計劃，然後在升級後重新啟用。但是，這會導致升級期間遺失所有計劃的快照。

- 對於私人鏡像倉庫安裝，請確保目標版本所需的 Helm Chart 和鏡像在您的私人鏡像倉庫中可用，並驗證您的自訂 Helm 值與新 Chart 版本相容。更多信息，請參閱"從私人註冊表安裝Trident Protect"。

步驟 **1** ：選取版本

Trident Protect 版本遵循基於日期的 `YY.MM`命名規則，其中「YY」代表年份的後兩位數字，「MM」代表月份。小版本更新遵循 `YY.MM.X`規則，其中「X」代表補丁級別。您將根據要升級的版本選擇要升級到的版本。

- 您可以將目前版本直接升級到與其相差不超過四個版本號的目標版本。例如，您可以直接從 24.10（或任何 24.10 的小版本）升級到 25.10。
- 如果您要從超出四版本週期範圍的版本升級，請執行多步驟升級。使用您要升級的 "舊版" 版本對應的升級說明，升級到符合四版本視窗期的最新版本。例如，如果您目前執行的是 24.10 版本，並且想要升級至 26.10 版本：
  a. 首次從 24.10 升級到 25.06。
  b. 然後從 25.06 升級到 26.10 。

步驟 **2**：升級 **Trident Protect**

若要升級Trident Protect，請執行下列步驟。

步驟

1. 更新 Trident Helm 儲存庫：

```
helm repo update
```

2. 升級Trident Protect CRD：

> 註 如果您是從 25.06 之前的版本升級，則需要執行此步驟，因為 CRD 現在已包含在Trident Protect Helm 圖表中。

a. 運行此命令將 CRD 的管理從 `trident-protect-crds`到 `trident-protect` ：

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

b. 運行此命令刪除 `trident-protect-crds`圖表：

> 註 不要卸載 `trident-protect-crds`圖表使用 Helm，因為這可能會刪除您的 CRD 和任何相關資料。

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. 升級Trident保護：

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2610.0 --namespace trident-protect
```

註	您可以透過新增以下內容來配置升級期間的日誌等級。 `--set logLevel=debug`升級命令。預設日誌等級為 `warn`。建議啟用偵錯日誌記錄進行故障排除，因為它可以幫助NetApp支援人員診斷問題，而無需更改日誌等級或重現問題。

管理及保護應用程式

使用**Trident Protect AppVault** 物件來管理儲存桶。

Trident Protect 的儲存桶自訂資源 (CR) 稱為 AppVault。AppVault 物件是儲存桶的聲明性 Kubernetes 工作流程表示。AppVault CR 包含儲存桶在保護作業（例如備份、快照、復原作業和SnapMirror複製）中所使用的必要配置。只有管理員才能建立應用保險庫。

在應用程式上執行資料保護操作時，您需要手動或從命令列建立 AppVault CR。 AppVaultCR 特定於您的環境，您可以使用本頁上的範例作為建立 AppVault CR 的指南。

註	確保 AppVault CR 位於安裝了Trident Protect 的叢集上。如果 AppVault CR 不存在或您無法訪問，命令列將顯示錯誤。

設定 **AppVault** 驗證和密碼

在建立 AppVault CR 之前，請確保您選擇的 AppVault 和資料移動器可以向提供者和任何相關資源進行驗證。

資料移動器儲存庫密碼

當您使用 CR 或Trident Protect CLI 外掛程式建立 AppVault 物件時，您可以為 Restic 和 Kopia 加密指定帶有自訂密碼的 Kubernetes 金鑰。如果您不指定金鑰， Trident Protect 將使用預設密碼。

- 手動建立 AppVault CR 時，使用 **spec.dataMoverPasswordSecretRef** 欄位指定金鑰。
- 使用Trident Protect CLI 建立 AppVault 物件時，請使用 `--data-mover-password-secret-ref`用於指定密鑰的參數。

註	若要加密快照和備份中繼資料或變更保存庫密碼，請使用 `spec.vaultPasswordSecretRef` 欄位。請參閱 "加密 Trident Protect 快照與備份的 AppVault 中繼資料"。

建立資料移動者儲存庫密碼機密

請參考以下範例建立密碼密鑰。建立 AppVault 物件時，您可以指示Trident Protect 使用此金鑰向資料移動器儲存庫進行驗證。

註

- 視您使用的資料移動器而定，您只需要加入該資料移動器的對應密碼。例如，如果您使用 Restic ，而且不打算在未來使用 Kopia ，則在建立機密時，只能包含 Restic 密碼。
- 請將密碼保存在安全的地方。您將需要它來還原同一叢集或其他叢集上的資料。如果集群或 `trident-protect`命名空間被刪除後，沒有密碼您將無法還原備份或快照。

使用 **CR**

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

使用**CLI**

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

S3 相容於儲存 **IAM** 權限

當您存取與 S3 相容的儲存空間（例如 Amazon S3、通用 S3）時， "StorageGRID S3" ， 或者 "ONTAP S3"使用Trident Protect 時，您需要確保提供的使用者憑證具有存取儲存桶的必要權限。以下是授予使用Trident Protect 進行存取所需的最低權限的政策範例。您可以將此策略套用至管理 S3 相容儲存桶策略的使用者。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
```

```
          "s3:ListBucket",
          "s3:DeleteObject"
        ],
        "Resource": "*"
      }
    ]
}
```

有關 Amazon S3 策略的更多信息，請參閱 "Amazon S3 文檔" 。

用於 **Amazon S3**（**AWS**）驗證的 **EKS Pod Identity**

Trident Protect 支援 Kopia 資料移動器操作的 EKS Pod Identity。此功能可實現對 S3 儲存桶的安全訪問，而無需將 AWS 憑證儲存在 Kubernetes 機密中。

EKS Pod Identity 與**Trident Protect** 的需求

在將 EKS Pod Identity 與Trident Protect 結合使用之前，請確保以下事項：

- 您的 EKS 叢集已啟用 Pod Identity。
- 您已建立具有必要的 S3 儲存桶權限的 IAM 角色。要了解更多信息，請參閱"S3 相容於儲存 IAM 權限"。
- IAM 角色與下列Trident Protect 服務帳號關聯：
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

有關啟用 Pod Identity 以及將 IAM 角色與服務帳戶關聯的詳細說明，請參閱 "AWS EKS Pod Identity 文檔"。

AppVault 設定 使用 EKS Pod Identity 時，請使用下列設定來設定您的 AppVault CR `useIAM: true`標記而不是明確的憑證：

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
      useIAM: true
```

定義 AppVault CR 時，您需要包含憑證以存取提供者託管的資源，除非您使用 IAM 驗證。如何產生憑證金鑰將根據提供者的不同而有所不同。以下是幾個提供者的命令列金鑰產生範例。您可以使用下列範例為每個雲端提供者的憑證建立金鑰。

Google Cloud

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

Amazon S3 （ AWS ）

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

Microsoft Azure

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

一般S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

ONTAP S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<ontap-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

StorageGRID S3

```
kubectl create secret generic <secret-name> \
```

```
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<storagegrid-s3-trident-protect-src
-bucket-secret> \
-n trident-protect
```

AppVault 建立範例

以下是每個提供者的 AppVault 定義範例。

AppVault CR 範例

您可以使用下列 CR 範例，為每個雲端供應商建立 AppVault 物件。

註

- 您可以選擇性地指定 Kubernetes 機密，其中包含 Restic 和 Kopia 儲存庫加密的自訂密碼。如需詳細資訊、請參閱 [資料移動器儲存庫密碼] 。
- 對於 Amazon S3 （ AWS ） AppVault 物件，您可以選擇性地指定一個工作區權杖，如果您使用單一登入（ SSO ）進行驗證，這會很有用。當您在中為提供者產生金鑰時適用於雲端供應商的 AppVault 主要世代範例，就會建立此權杖。
- 對於 S3 AppVault 物件，您可以選擇使用金鑰來指定傳出 S3 流量的外傳 Proxy URL `spec.providerConfig.S3.proxyURL` 。

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3 （ AWS ）

```
---
```

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret
```

註 | 對於使用 Pod Identity 和 Kopia 資料移動器的 EKS 環境，您可以刪除 `providerCredentials` 部分並添加 `useIAM: true`根據 `s3`配置。

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
```

```
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

一般S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
```

```
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

StorageGRID S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

使用**Trident Protect CLI** 建立 **AppVault** 的範例

您可以使用下列 CLI 命令範例，為每個供應商建立 AppVault CRS 。

註

- 您可以選擇性地指定 Kubernetes 機密，其中包含 Restic 和 Kopia 儲存庫加密的自訂密碼。如需詳細資訊、請參閱 [資料移動器儲存庫密碼] 。
- 對於 S3 AppVault 物件，您可以選擇使用引數，為輸出 S3 流量指定外傳 Proxy URL `--proxy-url <ip_address:port>` 。

Google Cloud

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Amazon S3 （AWS）

```
tridentctl-protect create vault AWS <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Microsoft Azure

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

一般S3

```
tridentctl-protect create vault GenericS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
```

```
-n trident-protect
```

StorageGRID S3

```
tridentctl-protect create vault StorageGridS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

支援 `**providerConfig.s3**`配置選項

請參閱下表以了解 S3 提供者設定選項：

參數	說明	預設	範例
`providerConfig.s3.skipCertValidation`	停用 SSL/TLS 憑證驗證。	錯	“真”，“假”
`providerConfig.s3.secure`	啟用與 S3 端點的安全 HTTPS 通訊。	是的	“真”，“假”
`providerConfig.s3.proxyURL`	指定用於連接 S3 的代理伺服器的 URL。	沒有任何	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	提供用於 SSL/TLS 驗證的自訂根 CA 憑證。	沒有任何	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	啟用 IAM 驗證以存取 S3 儲存桶。適用於 EKS Pod 識別。	錯	對、錯

檢視 **AppVault** 資訊

您可以使用Trident Protect CLI 外掛程式查看有關您在叢集上建立的 AppVault 物件的資訊。

步驟

1. 檢視 AppVault 物件的內容：

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

。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 安裝中指定了叢集名稱時，表格第一列中的叢集名稱才可用。例如： `--set clusterName=production1` 。

移除 **AppVault**

您可以隨時移除 AppVault 物件。

註	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

請確定您已刪除要刪除的 AppVault 所使用的所有快照和備份 CRS 。

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 Protect CLI** 刪除 **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
```

加密 **Trident Protect** 快照與備份的 **AppVault** 中繼資料

您可以加密 Trident Protect 儲存在 AppVault 中的快照和備份中繼資料，包括 Kubernetes Secrets 中的敏感資訊。您也可以變更保護加密資料的密碼。單一密碼變更會重新加密中繼資料加密金鑰，並將 AppVault 中的每個 Kopia 儲存庫和每個 Restic 儲存庫移至新密碼。

加密預設為停用。啟用後，Trident Protect 會在將中繼資料儲存至 AppVault 之前，先加密新的快照與備份中繼資料。這可保護您的應用程式資源，包括 Secrets 中的敏感資訊，除非您明確將這些資源從應用程式快照或備份中排除。

保存庫密碼用於保護加密資料的金鑰。請注意以下行為：

- 當您變更保存庫密碼時， Trident Protect 會使用新密碼保護相同的金鑰。它不會重新寫入已加密的資料。
- 啟用加密之前儲存的資料仍然可讀。
- 即使您之後關閉新資料的加密，加密後的資料仍然可以讀取。

關於這項工作

啟用加密後，除非您指定自訂密碼，否則 Trident Protect 會使用不安全的預設密碼。對於每個 AppVault ，請建立自訂保存庫密碼並在 `spec.vaultPasswordSecretRef`中參照它。

如果您使用預設密碼啟用加密，之後又新增了自訂密碼，則必須輪換密碼，以便將現有資料和儲存庫移至自訂密碼。請參閱 [輪換保存庫密碼]。

為每個 AppVault 使用唯一的自訂密碼。一次輪換一個 AppVault 的密碼，並在開始另一次密碼變更之前確認輪換已完成。

開始之前

請確保您擁有下列項目：

- 存取 `trident-protect`命名空間。

- 建立 Kubernetes Secrets 和修補 AppVault 自訂資源（CR）的權限。
- 為每個 AppVault 規劃分隔的密碼 Secret。

啟用中繼資料加密

您可以透過為 Trident Protect 設定 Helm 值來啟用中繼資料加密。

步驟

1. 使用 `values.yaml`檔案將 `resourceBackupEncryptionEnabled`設定為 `true`：
   a. 將以下內容新增至您的 `values.yaml`檔案：

```
resourceBackupEncryptionEnabled: true
```

   b. 套用檔案：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

2. 或者，使用 Helm `--set` 旗標將 `resourceBackupEncryptionEnabled`設定為 `true`：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set resourceBackupEncryptionEnabled=true \
  --reuse-values
```

結果

Trident Protect 會對新的快照和備份中繼資料進行加密。

接下來呢？

為每個 AppVault 建立自訂保存庫密碼。在此之前，Trident Protect 將使用不安全的預設密碼。

建立自訂保存庫密碼

若要使用您自己的密碼，請建立一個包含該密碼的 Kubernetes Secret，然後在 AppVault 中參照該 Secret。

關於這項工作

如果在建立 AppVault 時設定自訂密碼，則在儲存任何資料之前，AppVault 會立即使用該自訂密碼。您無需輪換密碼。

如果 AppVault 已經有 Kopia 或 Restic 儲存庫，新增密碼不會變更這些儲存庫的密碼。請勿使用本節中的步驟。請改為遵循 [輪換保存庫密碼] 中的步驟。

該 `VAULT_PASSWORD`值必須符合以下要求：

- 不得為空
- 必須小於 64 KiB
- 不得包含 NUL 字元
- 不得包含換行符
- 不得以空格或其他空白字元開頭或結尾

密碼可以包含其他字元，包括非 ASCII 字元。

步驟

1. 在與 AppVault 相同的命名空間中建立 Secret。Secret 必須包含名為 `VAULT_PASSWORD`的金鑰：

```
kubectl -n trident-protect create secret generic vault-password \
  --from-literal=VAULT_PASSWORD='<password>'
```

> 註 如果 AppVault 報告密碼無效，請根據上述要求檢查 `VAULT_PASSWORD` 值。常見原因為空值、結尾有換行字元，或是密碼開頭或結尾有空格。

2. 在 AppVault CR 的 `spec.vaultPasswordSecretRef`欄位中參照 Secret：

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: my-vault
  namespace: trident-protect
spec:
  vaultPasswordSecretRef: vault-password
  providerType: <provider-type>
  providerConfig:
    <provider-configuration>
```

如果 `spec.vaultPasswordSecretRef`為空，則 AppVault 使用不安全的預設密碼。

結果

AppVault 使用您的自訂密碼來保護中繼資料加密金鑰。

輪換保存庫密碼

您可以輪換已儲存資料的 AppVault 密碼。一次輪換會重新加密中繼資料加密金鑰，並將 AppVault 中的每個 Kopia 儲存庫和每個 Restic 儲存庫移至新密碼。

關於這項工作

在輪替過程中，Trident Protect 會執行下列操作：

- 等待正在執行的 AppVault 備份與恢復作業完成後再開始。

- 使用新密碼保護中繼資料加密金鑰(`metadata-key.json`）。不會重新寫入歸檔物件。
- 尋找在 AppVault 前置字元下的每個 Kopia 儲存庫和每個 Restic 儲存庫，並將其密碼變更為新密碼。已使用新密碼開啟的儲存庫不會變更。
- 跳過備份或還原作業仍在使用的儲存庫，稍後重試。
- 暫停參照 AppVault 的已啟用排程。對於每個排程， Trident Protect 會將 `spec.enabled` 設為 `false` 並新增下列註解：

```
protect.trident.netapp.io/vault-password-rotation-suspended:
<namespace>/<appvault-name>
```

輪換完成後，Trident Protect 只會重新啟用具有此註解的排程。如果輪換失敗，則會從停止的位置繼續執行。已使用新密碼的儲存庫不會再次變更。

開始之前

請確保您擁有下列項目：

- 符合 保存庫密碼要求 的新密碼。
- 沒有其他 AppVault 正在輪換其密碼。如果多個 AppVaults 共享同一個儲存桶和前綴，則只有其中一個可以輪換密碼。請參閱 管理共享儲存區的 AppVaults 輪換。

步驟

1. 建立一個與目前 Secret 名稱不同的新 Secret：

```
kubectl -n trident-protect create secret generic vault-password-new \
  --from-literal=VAULT_PASSWORD='<new-password>'
```

重要	在輪替完成之前，請勿編輯或刪除目前的 Secret。Trident Protect 需要目前的密碼才能開啟現有資料，直到每個儲存庫都使用新密碼為止。如果您過早刪除目前的 Secret，請使用相同的名稱和原始的 `VAULT_PASSWORD`值重新建立它。

2. 啟用輪換並設定 AppVault 以使用新的機密。將 `my-vault` 替換為您的 AppVault 名稱：

```
kubectl -n trident-protect patch appvault my-vault --type merge --patch
'
spec:
  enableRotation: true
  vaultPasswordSecretRef: vault-password-new
'
```

註	在變更密碼之前，您必須將 `spec.enableRotation`設定為 `true`。如果是 `false`，AppVault 會報告輪替已停用，並且不會變更密碼。

3. 請依照 [監控密碼輪替] 中的步驟來監控輪替。
4. 輪替結束後，如果沒有其他項目使用先前的 Secret，請將其刪除。

結果

中繼資料加密金鑰以及 AppVault 中的每個 Kopia 和 Restic 儲存庫都受到新密碼的保護，而且 Trident Protect 暫停的排程已重新啟用。

監控密碼輪替

您可以查看密碼輪換的進度並確認是否已完成。

步驟

1. 查看 AppVault：

```
kubectl -n trident-protect get appvault my-vault --output yaml
```

2. 在輸出結果中，查看該 `status.vaultPasswordRotation`部分。旋轉過程中，它看起來像這樣：

```
status:
  vaultPasswordRotation:
    state: Rotating
    rotated: 3
    pending: 1
    message: <message>
```

下表說明各欄位：

欄位	說明
status.vaultPasswordRotation.state	目前的旋轉狀態。
status.vaultPasswordRotation.rotated	現在使用新密碼的儲存庫數量。
status.vaultPasswordRotation.pending	因為備份或還原正在使用而跳過的儲存庫數量。
status.vaultPasswordRotation.message	最新進度或故障訊息。

結果

當下列所有條件都成立時，輪替即完成：

- `status.vaultPasswordRotation` 不存在。
- `status.previousRepositoryPasswordSecretRef` 為空。
- `status.repositoryPasswordSecretRef` 符合 `spec.vaultPasswordSecretRef`。

接下來呢？

如果輪替未完成，請檢查下列事項：

- 輪換未開始： `spec.enableRotation` 可能是 `false`。將其設為 `true`，然後將 `spec.vaultPasswordSecretRef` 設定為新的 Secret。
- 儲存庫處於擱置狀態：備份或還原作業仍在使用這些儲存庫。請等待作業完成。Trident Protect 會自動重試擱置中的儲存庫。
- 由於另一個 **AppVault** 擁有鎖定，因此輪換不會開始：請參閱 管理共享儲存區的 AppVaults 輪換。
- 訊息欄位報告故障：解決訊息中所述的問題。輪替會從停止的位置繼續。

從 **dataMoverPasswordSecretRef** 移轉

如果現有的 AppVault 使用較舊的 `spec.dataMoverPasswordSecretRef`欄位，您可以將其移至 `spec.vaultPasswordSecretRef`。

關於這項工作

Trident Protect 保留 `spec.dataMoverPasswordSecretRef`以提供向後相容性。請將 `spec.vaultPasswordSecretRef`用於新的 AppVaults。

步驟

1. 將 `spec.dataMoverPasswordSecretRef`及其 Secret 保留在原處。
2. 使用 `spec.vaultPasswordSecretRef` 將密碼輪替為新的 Secret。請依照 [輪換保存庫密碼] 中的步驟操作。
3. 輪替結束後，請從 AppVault 移除 `spec.dataMoverPasswordSecretRef`。
4. 如果沒有其他項目使用舊的 Secret，請將其刪除。

管理共享儲存區的 **AppVaults** 輪換

如果多個 AppVaults 使用同一個儲存桶和前綴，則只有其中一個可以輪換密碼。

關於這項工作

密碼輪替使用名為 `password-rotation.lock`的鎖定檔案。鎖定會記錄允許輪替儲存區和前置碼密碼之 `AppVault` 的 `UID`。如果第二個 `AppVault` 具有 `spec.enableRotation: true`，且鎖定指定了不同的 AppVault，則第二個 AppVault 無法進行輪替。

範例 **1.** 步驟

將輪替轉移至另一個 **AppVault**

1. 在擁有該鎖的 AppVault 上，清除 `spec.enableRotation`：

```
kubectl -n trident-protect patch appvault <current-owner> --type
merge --patch '
spec:
  enableRotation: false
'
```

2. 等待鎖被移除。

3. 在接下來應該輪替的 AppVault 上，將 `spec.enableRotation`設為 `true`：

```
kubectl -n trident-protect patch appvault <new-owner> --type merge
--patch '
spec:
  enableRotation: true
'
```

移除由已刪除的 **AppVault** 所留下的鎖定

如果擁有該鎖的 AppVault 已不存在，且該鎖未被移除（例如在當機之後），請使用下列步驟。

1. 從您儲存桶的 AppVault 前綴中刪除 `password-rotation.lock`檔案。
2. 在應該輪替的 AppVault 上，將 `spec.enableRotation`設定為 `true`：

```
kubectl -n trident-protect patch appvault <appvault-name> --type
merge --patch '
spec:
  enableRotation: true
'
```

相關資訊

- "使用Trident Protect AppVault 物件來管理儲存桶。"
- "自訂Trident Protect 安裝"
- "保護應用程式"
- "使用Trident Protect 恢復應用程式"

使用**Trident Protect** 定義管理應用程式

您可以透過建立應用程式 CR 和關聯的 AppVault CR 來定義要使用Trident Protect 管理的應用程式。

建立 **AppVault CR**

您需要建立一個 AppVault CR，該 CR 將在對應用程式執行資料保護操作時使用，並且 AppVault CR 需要位於安裝了Trident Protect 的叢集上。AppVault CR 是針對您的特定環境的；有關 AppVault CR 的範例，請參閱："AppVault 自訂資源。"

定義應用程式

您需要定義要使用Trident Protect 管理的每個應用程式。您可以透過手動建立應用程式 CR 或使用Trident Protect CLI 來定義要管理的應用程式。

註	若要定義 VM 範圍的應用程式，以按名稱而非整個命名空間來保護特定的 KubeVirt 虛擬機器（VM），請使用 Application CR 中的 `spec.includedVirtualMachines`欄位或 CLI 中的 `--virtual-machines`旗標。如需詳細資訊、欄位參考和 YAML 範例，請參閱"使用 Trident Protect 保護 KubeVirt 虛擬機器"。

使用 **CR** 新增應用程式

步驟

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 ）使用命名空間和標籤選取器來指定應用程式使用的命名空間和資源。應用程式命名空間必須是此清單的一部分。標籤選取器為選用項目，可用於篩選每個指定命名空間內的資源。
      - * spec.includedClusterScopedResources*: （ *Optional* ）使用此屬性來指定要包含在應用程式定義中的叢集範圍資源。此屬性可讓您根據這些資源的群組，版本，種類和標籤來選取這些資源。
        - **groupVersionKind** ：（ _required _ ）指定叢集範圍資源的 API 群組，版本及種類。
        - *labelSelector * ：（ *Optional* ）根據叢集範圍的資源標籤來篩選資源。
      - **spec.volumeGroups**: （可選）使用此屬性將相關 Volume 一起快照為損毀狀況一致 Volume 群組快照。如需詳細資訊，請參閱 "搭配 Trident Protect 使用 Volume 群組快照"。
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (可選) 此註解僅適用於從虛擬機定義的應用程序，例如 KubeVirt 環境，其中檔案系統凍結發生在快照之前。指定此應用程式在快照期間是否可以寫入檔案系統。如果設定為 true，應用程式將忽略全域設置，並且可以在快照期間寫入檔案系統。如果設定為 false，應用程式將忽略全域設置，並且在快照期間檔案系統將被凍結。如果指定了註解，但應用程式定義中沒有虛擬機，則忽略該註解。如未特別說明，則申請流程如下："全球Trident Protect 冷凍設置" 。

如果您需要在建立應用程式之後套用此註釋，可以使用下列命令：

```
kubectl annotate application -n <application CR namespace> <application CR name> protect.trident.netapp.io/skip-vm-freeze="true"
```

+
YAML 範例：

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
```

```
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: namespace-1
      labelSelector:
        matchLabels:
          app: example-app
    - namespace: namespace-2
      labelSelector:
        matchLabels:
          app: another-example-app
  includedClusterScopedResources:
    - groupVersionKind:
        group: rbac.authorization.k8s.io
        kind: ClusterRole
        version: v1
      labelSelector:
        matchLabels:
          mylabel: test
```

1. （選用）如有需要、您可以將資源篩選新增至相同的 CR、以包含或排除特定資源：

   。通用篩選器範例：

   - **resourceFilter.resourceSelectionCriteria** ：（篩選所需）使用 `Include`或包含或 `Exclude` 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** ：一組 resourceMatcher 物件。如果您在此陣列中定義多個元素，它們會比對為 OR 作業，而每個元素（群組，種類，版本）內的欄位會比對為 AND 作業。
     - **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"`：。

註	當兩者 `resourceFilter`和 `labelSelector`被使用， `resourceFilter`首先運行，然後 `labelSelector`應用於結果資源。

例如：

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

- 僅限 **PVC** 的篩選器範例：

  要定義僅限 PVC 的應用程式、您還必須在資源篩選器中包含 `PersistentVolume`和 `VolumeSnapshotClass`。快照和備份作業取決於 `PersistentVolume`（繫結至每個 PVC 的叢集範圍磁碟區）和 `VolumeSnapshotClass`（快照驅動程式）、若無這些作業、將會失敗。例如：

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-pvc-app
  namespace: my-app-namespace
spec:
  includedNamespaces:
  - namespace: my-app-namespace
  resourceFilter:
    resourceMatchers:
    - kind: PersistentVolumeClaim
      version: v1
    - kind: PersistentVolume
      version: v1
    - kind: VolumeSnapshotClass
      version: v1
```

```
    resourceSelectionCriteria: Include
```

2. 建立應用程式 CR 以符合您的環境之後，請套用 CR 。例如：

```
kubectl apply -f maria-app.yaml
```

步驟

1. 使用下列其中一個範例建立及套用應用程式定義，以環境中的資訊取代方括號中的值。您可以在應用程式定義中加入命名空間和資源，使用以逗號分隔的清單，以及範例中所示的引數。

建立應用程式時，您可以選擇使用註解來指定應用程式在快照期間是否可以寫入檔案系統。這僅適用於從虛擬機定義的應用程序，例如 KubeVirt 環境，其中檔案系統凍結發生在快照之前。如果您將註釋設定為 `true`該應用程式忽略全域設置，可以在快照期間寫入檔案系統。如果你把它設定為 `false`該應用程式忽略全域設置，導致檔案系統在快照期間凍結。如果使用了註解，但應用程式定義中沒有虛擬機，則該註解將被忽略。如果您不使用註解，應用程式將遵循以下規則："全球Trident Protect 冷凍設置" 。

+ 在使用 CLI 建立應用程式時，若要指定註釋，可以使用 `--annotation`標誌。

+ * 建立應用程式並使用全域設定來控制檔案系統凍結行為：

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace>
```

+ * 建立應用程式並配置本機應用程式設定以控制檔案系統凍結行為：

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace> --annotation
protect.trident.netapp.io/skip-vm-freeze=<"true"|"false">
```

+ * 您可以使用 `--resource-filter-include`和 `--resource-filter-exclude`標誌，根據 `resourceSelectionCriteria`（例如群組、種類、版本、標籤、名稱和命名空間）來包含或排除資源，如下例所示：

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace> --resource-filter-include
'[{"Group":"apps","Kind":"Deployment","Version":"v1","Names":["my-
```

```
deployment"],"Namespaces":["my-namespace"],"LabelSelectors":["app=my-
app"]}]'
```

+ * 要定義僅限 PVC 的應用程式，您還必須在資源篩選器中包含 `PersistentVolume`和 `VolumeSnapshotClass`。快照和備份操作依賴 `PersistentVolume`（綁定到每個 PVC 的叢集範圍磁碟區）和 `VolumeSnapshotClass`（快照驅動程式），缺少它們將會失敗。例如：

+

```
tridentctl-protect create app my-pvc-app --namespaces <my-app-
namespace> --resource-filter-include
'[{"Kind":"PersistentVolumeClaim","Version":"v1"},{"Kind":"PersistentVo
lume","Version":"v1"},{"Kind":"VolumeSnapshotClass","Version":"v1"}]'
-n <my-app-namespace>
```

使用Trident Protect 保護應用程式

您可以使用自動保護策略或臨時保護策略，透過拍攝快照和備份來保護Trident Protect 管理的所有應用程式。

註

- 您可以設定Trident Protect 在資料保護作業期間凍結和解凍檔案系統。"了解更多關於使用Trident Protect 設定檔系統凍結的信息"。
- 若要在同一時間點對應用程式的相關 Volume 進行快照，請在應用程式中定義 Volume 群組。請參閱 "搭配 Trident Protect 使用 Volume 群組快照"。

建立隨需快照

您可以隨時建立隨需快照。

註

如果叢集範圍的資源在應用程式定義中明確參照，或是具有任何應用程式命名空間的參照，則這些資源會包含在備份，快照或複製中。

使用 CR 建立快照

步驟

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.relaimersPolicy** ：（ *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

步驟

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> -n
<application_namespace>
```

建立隨選備份

您可以隨時備份應用程式。

註	如果叢集範圍的資源在應用程式定義中明確參照，或是具有任何應用程式命名空間的參照，則這些資源會包含在備份，快照或複製中。

開始之前

確保 AWS 工作階段權杖到期時間足以應付任何長期執行的 S3 備份作業。如果 Token 在備份作業期間過期，作業可能會失敗。

- 如需檢查目前工作階段權杖到期時間的詳細資訊，請參閱 "AWS API 文件"。
- 如需 AWS 資源認證的詳細資訊，請參閱 "AWS IAM 文件"。

使用 **CR** 建立備份

步驟

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**：（可選）：用作備份來源的快照的名稱。如果未提供，將會建立並備份暫存快照。

   YAML 範例：

```
---
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

步驟

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> --data-mover
<Kopia_or_Restic> -n <application_namespace>
```

您可以選擇性地使用 `--full-backup`旗標來指定備份是否應為非遞增備份。依預設，所有備份都是遞增備份。使用此旗標時，備份會變成非遞增備份。最佳做法是定期執行完整備份，然後在完整備份之間執行遞增備份，以將與還原相關的風險降至最低。

支援的備份註釋

下表描述了建立備份 CR 時可以使用的註解：

註釋	類型	說明	預設值
protect.trident.netapp.io/full-backup	字串	指定備份是否應為非增量備份。設定為 `true` 建立非增量備份。最佳實踐是定期執行完整備份，然後在兩次完整備份之間執行增量備份，以最大限度地降低與復原相關的風險。	"假"
protect.trident.netapp.io/snapshot-completion-timeout	字串	完成整個快照操作允許的最長時間。	60米
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	字串	卷快照達到可用狀態所需的最長時間。	30米
protect.trident.netapp.io/volume-snapshots-created-timeout	字串	建立磁碟區快照允許的最長時間。	5米
protect.trident.netapp.io/pvc-bind-timeout-sec	字串	等待新建立的持久卷聲明 (PVC) 到達的最大時間（以秒為單位） `Bound`操作失敗前的階段。	1200（20分鐘）

建立資料保護排程

保護策略透過按照定義的計劃建立快照、備份或兩者來保護應用程式。您可以選擇每小時、每天、每周和每月建立快照和備份，並可以指定要保留的副本數量。您可以使用 full-backup-rule 註解來排程非增量式完整備份。預設情況下，所有備份都是增量的。定期執行完整備份以及其間的增量備份有助於降低與復原相關的風險。

註

- 您可以透過設定 `backupRetention`歸零， `snapshotRetention`為大於零的值。環境 `snapshotRetention`為零意味著任何計劃的備份仍將建立快照，但這些快照是臨時的，並在備份完成後立即刪除。
- 如果叢集範圍的資源在應用程式定義中明確參照，或是具有任何應用程式命名空間的參照，則這些資源會包含在備份，快照或複製中。

使用 **CR** 建立排程

步驟

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**: (必要) 要保留的備份數量。零表示不應建立備份（僅快照）。
- **spec.backupReclaimPolicy**: (可選) 決定如果備份 CR 在其保留期內被刪除，則備份會發生什麼情況。保留期過後，備份檔案總是會被刪除。可能的值（區分大小寫）：
  - `Retain`（預設）
  - `Delete`
- **spec.snapshotRetention**: (必需) 要保留的快照數量。零表示不建立任何快照。
- **spec.snapshotReclaimPolicy**: (可選) 決定如果快照 CR 在其保留期內被刪除，則快照會發生什麼情況。保留期過後，快照總是會被刪除。可能的值（區分大小寫）：
  - `Retain`
  - `Delete`(預設)
- * 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`）
  - `Custom`
- **spec.dayOfMonth**：（可選）計畫應運行的月份日期（1 - 31）。如果粒度設定為 `Monthly`。該值必須以字串形式提供。
- **spec.dayOfWeek**：（可選）計畫應運行的星期幾（0 - 7）。值 0 或 7 表示星期日。如果粒度設定為 `Weekly`。該值必須以字串形式提供。
- **spec.hour**：（可選）計畫應運行的小時數（0 - 23）。如果粒度設定為 `Daily`，`Weekly` ，或者 `Monthly`。該值必須以字串形式提供。
- **spec.minute**：（可選）計畫應運行的小時中的分鐘數（0 - 59）。如果粒度設定為 `Hourly`，`Daily` ，`Weekly` ，或者 `Monthly`。該值必須以字串形式提供。
- **spec.runImmediately**：（選用）設定為 `true`以在建立排程時觸發一次性立即基準執行（根據保留設定進行備份和 / 或快照）。預設為 `false`。這不會修改後續的重複執行。

  備份和快照計劃的範例 YAML：

```
---
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: Daily
  hour: "0"
  minute: "0"
```

僅快照計劃的範例 YAML：

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"
```

範例 YAML 文件，用於實現立即運行的排程任務：

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-daily-schedule-run-immediately
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "7"
  snapshotRetention: "7"
  granularity: Daily
  hour: "3"
  minute: "0"
  runImmediately: true
```

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

步驟

1. 建立保護排程，以環境資訊取代方括號中的值。例如：

> 註 | 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> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_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> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

以下選項可讓您對日程安排進行更多控制：

- 完整備份計畫：使用 `--full-backup-rule` 標記以安排非增量式完整備份。此標誌僅適用於 `--granularity Daily`。可能的值：
  - `Always`每天都要建立完整備份。
  - 具體工作日：指定一個或多個日期，以逗號分隔（例如， `"Monday,Thursday"`）。有效值：星期一、星期二、星期三、星期四、星期五、星期六、星期日。

> 註 | 這 `--full-backup-rule` 此標誌位不適用於按小時、按週或按月劃分的粒度。

- 立即基線保護：使用 `--run-immediately true` 在建立排程時立即建立初始備份或快照，而不是等待首次排程執行時間。預設為 `false`。
- 僅快照計劃：設定 `--backup-retention 0` 並指定一個大於零的值 `--snapshot-retention`。

下表描述了建立計劃變更請求 (CR) 時可以使用的註釋：

註釋	類型	說明	預設值
protect.trident.netapp.io/full-backup-rule	字串	指定安排完整備份的規則。你可以將其設定為 `Always` 您可以根據需要進行持續完整備份或自訂備份。例如，如果您選擇按日粒度進行備份，則可以指定應進行完整備份的星期幾（例如， `"Monday,Thursday"`）。有效的工作日值為：星期一、星期二、星期三、星期四、星期五、星期六、星期日。請注意，此註釋只能用於已包含以下內容的日程表： `granularity` 設定為 `Daily`。	未設定（所有備份均為增量備份）
protect.trident.netapp.io/snapshot-completion-timeout	字串	完成整個快照操作允許的最長時間。	60米
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	字串	卷快照達到可用狀態所需的最長時間。	30米
protect.trident.netapp.io/volume-snapshots-created-timeout	字串	建立磁碟區快照允許的最長時間。	5米
protect.trident.netapp.io/pvc-bind-timeout-sec	字串	等待新建立的持久卷聲明 (PVC) 到達的最大時間（以秒為單位） `Bound`操作失敗前的階段。	1200（20分鐘）

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。

> 註 確保回收策略設定為 `Delete`從物件儲存中刪除所有備份資料。該策略的預設值是 `Retain`以避免意外資料遺失。如果政策沒有改變 `Delete` ，備份資料將保留在物件儲存中，需要手動刪除。

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，則可以為使用 azure-netapp-files 儲存類別且在Trident 24.06 之前建立的儲存後端啟用節省空間的備份和還原功能。此功能適用於 NFSv4 卷，並且不會佔用容量池中的額外空間。

開始之前

請確認下列事項：

- 您已安裝Trident Protect。
- 您已在Trident Protect中定義了一個應用程式。在您完成此步驟之前，此應用程式的保護功能將受到限制。
- 您已 `azure-netapp-files` 選擇儲存後端的預設儲存類別。

▼ 展開以進行組態步驟

1. 如果 anf 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"
   ```

   +
   如果未啟用快照目錄， Trident Protect 將選擇常規備份功能，該功能會在備份過程中暫時佔用容量池中的

空間。在這種情況下，請確保容量池中有足夠的空間來建立與被備份磁碟區大小相同的臨時磁碟區。

結果

該應用程式已準備好使用Trident Protect 進行備份和還原。每個 PVC 也可供其他應用程式用於備份和還原。

搭配 **Trident Protect** 使用 **Volume** 群組快照

使用 Volume 群組快照可以擷取相同時間點的相關持續性 Volume 宣告（PVC）。這對於寫入多個 Volume 的應用程式非常有用，包括具有多個磁碟的虛擬機器（VM）。

關於這項工作

您可以在 `spec.volumeGroups`欄位中定義 Volume 群組，該欄位位於 `Application`自訂資源 （CR） 中。當您建立應用程式的快照時，Trident Protect 會使用 ONTAP 一致性群組，在同一時刻對 Volume 群組中的所有 PVC 建立快照。系統會為 Volume 群組中的每個命名空間建立一個一致性群組。群組中的每個 Volume 也會取得各自的快照，因此您可以個別還原 Volume。

Trident Protect 會為不在 Volume 群組中的 PVC 建立個別的快照。這些快照是依序建立的，因此無法保證它們反映相同的時間點。

Volume 群組快照適用於隨需快照、排程快照，以及 Trident Protect 在備份過程中建立的快照。如果您備份現有快照， Trident Protect 將使用該快照，而不會建立新的群組快照。複寫也不會建立另一個群組快照。它會使用群組快照已建立的每個 Volume 的快照。

Trident Protect 只有在滿足以下所有條件時，才會在同一個時間點擷取 Volume：

- 它們屬於同一 Volume 群組。
- 它們位於同一命名空間中。
- 它們位於同一儲存設備虛擬機器（SVM）上。
- 此命名空間中的 Volume 群組最多包含 80 個 Volume。

如果 Trident Protect 無法建立群組快照（例如，由於未滿足某個要求），它會新增警告至 `status.warnings`並建立該群組各個 PVC 的單獨快照。這些快照是依序建立的，因此它們可能無法反映相同的時間點。應用程式快照仍會完成。

下表說明常見情況下會發生的狀況。

情況	結果
Volume 群組選擇多個命名空間中的 PVC	群組快照屬於一個命名空間，因此 Trident Protect 會在每個命名空間中建立個別的群組快照。這些群組快照不會在同一時刻擷取。其中一個可能成功，而另一個仍在執行中、發生故障，或退回至個別快照。例如，在 `payments-a`和 `payments-b`中標記為 `app: payments-db`的 PVC 會成為兩個個別的群組快照。請勿將來自不同命名空間的群組快照視為同一個還原點。
命名空間中的 PVC 存在於多個 SVM 上	Trident Protect 不會依 SVM 或 Trident 後端分割 Volume。ONTAP 一致性群組不能包含來自多個 SVM 的 Volume，因此該整個命名空間（包括共享 SVM 的 Volume）的群組快照會失敗。Trident Protect 會新增警告，移除群組快照，並為每個 Volume 建立個別的快照。若要一起擷取 Volume，請將它們放在同一個 SVM 上。

情況	結果
一個命名空間擁有超過 80 個匹配的 PVC	Trident Protect 只會跳過該命名空間，並為其 PVC 建立個別的快照。同一 Volume 群組中的其他命名空間仍會獲得群組快照。
一個 PVC 符合多個 `volumeGroups`項目	Trident Protect 會將 PVC 新增至兩個 Volume 群組，其中一個群組快照可能會失敗。失敗的群組快照會回復為個別快照。請確定每個 PVC 僅符合一個項目。
Volume 群組包含使用不支援 Volume 群組快照的驅動程式或協定的 PVC	群組快照失敗，Trident Protect 會為這些 PVC 建立個別的快照。

開始之前

- 您的 PVC 必須使用支援 Volume 群組快照的儲存設備驅動程式和協定。如需支援的驅動程式與協定，請參閱 "使用 Trident Volume 群組快照"。
- 您的叢集必須執行 Kubernetes 1.36 或更高版本。
- Kubernetes 快照控制器必須為 8.5.0 或更新版本，並啟用 `CSIVolumeGroupSnapshot` 功能閘道。
- `v1` 的 `VolumeGroupSnapshot`、 `VolumeGroupSnapshotClass` 和 `VolumeGroupSnapshotContent` CRD 必須已安裝。
- 您必須搭配線上後端使用 Trident 26.10 或更新版本。 `CSIVolumeGroupSnapshot` 功能閘道也必須在 Trident `csi-snapshotter` Container 上啟用。
- 必須定義使用 `csi.trident.netapp.io`驅動程式的 `A `VolumeGroupSnapshotClass`。如果只定義了一個類別，Trident Protect 會使用該類別。如果定義了多個類別，請使用 `snapshot.storage.kubernetes.io/is-default-class: "true"`標註 `Trident Protect` 應使用的類別。否則，`Trident Protect` 會依字母順序使用第一個類別。您無法在 `Application`或 `Snapshot` CR 中選擇類別。如需詳細資訊，請參閱"卷冊組快照類"。
- Trident 後端中的 ONTAP 帳戶必須允許建立、取得和刪除一致性群組，以及為其建立快照。請參閱 "管理應用程式一致性群組" 和 "管理應用程式一致性群組快照"。預設的 SVM 系統管理員角色 (`vsadmin`) 不包含這些權限。

定義 **Volume** 群組

Trident Protect 僅為屬於 Volume 群組的 Volume 建立群組快照。將 `spec.volumeGroups`新增至 `Application` CR 以定義群組。每個項目都是一個分隔的群組。依標籤、VM 或兩者來選擇 Volume。

關於這項工作

- 您只能在 `Application` CR 中定義 Volume 群組。 `tridentctl-protect` CLI 沒有 Volume 群組選項。
- Trident Protect 僅包含 Volume 群組中已綁定的 PVC，並為未綁定的 PVC 建立單獨的快照。如果某個條目未選擇任何已綁定的 PVC，Trident Protect 不會為該條目建立群組快照。
- 確保每個 PVC 僅與一個條目相符。

開始之前

定義您要保護的應用程式。請參閱"使用Trident Protect 定義管理應用程式"。

步驟

1. 如果按標籤選擇 Volume，請為要分組的 PVC 貼上標籤。您可以使用任何標籤鍵和值。本頁範例使用 `volume-group=my-volume-group`。

```
kubectl label pvc <pvc_name_1> <pvc_name_2> volume-group=my-volume-group
-n <application_namespace>
```

2. 將 `spec.volumeGroups` 新增至您現有的 `Application` CR 檔案中。每個項目必須包含 `labelSelector`、`vmSelector` 或兩者。如果項目同時包含這兩個欄位，則 Volume 群組會包含任一欄位所選取的每個 Volume。

   - **spec.volumeGroups[].labelSelector**: Kubernetes 標籤選擇器 (`matchLabels` 或 `matchExpressions`），用於選擇應用程式中的 PVC。空的選擇器無效。
   - **spec.volumeGroups[].vmSelector**：KubeVirt VM 列表，每個 VM 都有一個 `namespace`和 `name`。Trident Protect 包含每個 VM 裝載的 PVC。

   以下範例定義了兩個 Volume 群組。第一組包含標記為 `volume-group: my-volume-group`的 PVC。第二組包含 `my-vm-1`和 `my-vm-2`裝載的 PVC。

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-namespace
spec:
  includedNamespaces:
    - namespace: my-namespace
  volumeGroups:
    - labelSelector:
        matchLabels:
          volume-group: my-volume-group
    - vmSelector:
        - namespace: my-namespace
          name: my-vm-1
        - namespace: my-namespace
          name: my-vm-2
```

+

> 註 保留您的 `Application` CR 中已有的其他欄位，例如 `spec.resourceFilter`。如果您在套用的檔案中遺漏了某個欄位，`kubectl apply`會將其從 CR 中刪除。

3. 套用更新後的 CR：

```
kubectl apply -f <application_cr_file>.yaml
```

結果

`Application` CR 包含 Volume 群組。應用程式的下一個快照會使用它們。

建立快照

為應用程式建立隨選快照、隨選備份或資料保護排程。您不需要新增任何欄位。Trident Protect 使用您在 Application CR 中定義的 Volume 群組。如需相關指示，請參閱"使用Trident Protect 保護應用程式"。

註 預設情況下，Trident Protect 會在凍結虛擬機器檔案系統之前建立群組快照。Volume 群組中的 Volume 會在不凍結的情況下擷取，因此虛擬機器尚未寫入磁碟的資料不會包含在內。不在 Volume 群組中的磁碟會在稍後擷取，屆時 Trident Protect 會在凍結虛擬機器後建立個別的快照。若要同時擷取虛擬機器的所有磁碟，請將所有磁碟都包含在 Volume 群組中。如需凍結虛擬機器檔案系統的詳細資訊，請參閱 "使用 KubeVirt VM 保護資料"。

查看結果

快照完成後，檢查其狀態以確認 Trident Protect 是否已建立群組快照。

註 已完成的應用程式快照並不代表其群組快照已成功。即使每個群組快照都被跳過或失敗，VolumeGroupSnapshotsCreated 條件也會變成 true，而且 Trident Protect 不會建立 Kubernetes 事件或計量。若要偵測此情況，請監控 status.warnings 以及 skipped 和 failed 欄位。

步驟

1. 取得快照名稱。若要列出應用程式的快照，請執行以下命令：

```
kubectl get snapshots -n <application_namespace>
```

2. 檢查快照狀態：

```
kubectl get snapshots <snapshot_name> -n <application_namespace> -o yaml
```

結果

`status.volumeGroupSnapshots` 欄位包含每個群組快照的一個項目。Trident Protect 會為 Volume 群組中的每個命名空間建立一個分隔的群組快照。以下範例為簡化版：

```
status:
  volumeGroupSnapshots:
    - name: vgs-<snapshot_uid_prefix>-<group_index>-<namespace>
      namespace: <namespace>
      ready: true
      memberPVCs:
```

```
    - pvcName: <pvc_name_1>
      pvcNamespace: <namespace>
    - pvcName: <pvc_name_2>
      pvcNamespace: <namespace>
```

群組快照名稱包括 `Snapshot` UID 的前八個字元、 `spec.volumeGroups` 條目的索引（從 `0` 開始），以及命名空間，截斷為 12 個字元。

- `ready: true` 表示群組快照成功。組成該群組的快照也會列在 `status.volumeSnapshots` 之下。這些是群組的成員快照，而不是相同 Volume 的額外快照。
- `skipped: true` 表示 Trident Protect 沒有嘗試建立群組快照，因為該命名空間中的 Volume 群組包含超過 80 個 PVC。Trident Protect 改為建立群組內各個 Volume 的個別快照，並將 `skipReason` 文字新增至 `status.warnings`。
- `failed: true` 表示 Trident Protect 嘗試建立群組快照，但失敗了。Trident Protect 改為建立群組內各 Volume 的個別快照，並將 `error` 文字新增至 `status.warnings`。

臨時 PVC 標籤和清理

Trident Protect 會在群組快照中的每個 PVC 上新增一個標籤，例如 `protect.trident.netapp.io/vgs-group-673c0189-0=true`。您自己的標籤不會更改。該標籤會一直保留在 PVC 上，直到您刪除應用程式快照。刪除應用程式快照時， Trident Protect 也會刪除其 Volume 群組快照和成員快照，並從 PVC 移除標籤。無論快照的 `reclaimPolicy`是 `Delete`還是 `Retain`，都會發生這種情況。

接下來呢？

從快照還原應用程式的方式與還原任何其他快照的方式相同。Volume 群組快照沒有分隔的還原。Trident Protect 從成員快照還原，因此您可以還原個別的 PVC 或 VM，而無需還原群組中的每個 PVC。請參閱 "從快照還原至不同的命名空間" 或 "從快照還原至原始命名空間"。

相關資訊

- "使用Trident Protect 定義管理應用程式"
- "使用Trident Protect 保護應用程式"
- "使用 Trident Protect 保護 KubeVirt 虛擬機器"

還原應用程式

使用Trident Protect 恢復應用程式

您可以使用Trident Protect 從快照或備份中還原您的應用程式。將應用程式還原到同一群集時，從現有快照恢復速度會更快。

註

- 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。
- 對於 qtree 卷，支援從備份還原到其他命名空間或原始命名空間。但是，對於 qtree 卷，不支援從快照還原到其他命名空間或原始命名空間。
- 您可以使用進階設定來自訂恢復操作。欲了解更多信息，請參閱 "使用進階Trident Protect 恢復設定"。

- 若要從歸檔中復原特定的 KubeVirt 虛擬機器 (VM) 而不還原所有命名空間資源，請使用復原 CR 上的 `includedVirtualMachines`欄位。有關支援的 CR、欄位參考和 YAML 範例，請參閱"使用 Trident Protect 保護 KubeVirt 虛擬機器"。

從備份還原至不同的命名空間

當您使用 BackupRestore CR 將備份還原到不同的命名空間時， Trident Protect 會在新的命名空間中還原應用程序，並為還原的應用程式建立一個應用程式 CR。為了保護已還原的應用程序，可以建立按需備份或快照，或製定保護計劃。

註

- 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 還原到新命名空間時，必須先手動建立目標命名空間，然後再套用 CR。Trident Protect 僅在使用 CLI 時才會自動建立命名空間。

開始之前

確保 AWS 工作階段權杖到期時間足以執行任何長時間執行的 S3 還原作業。如果 Token 在還原作業期間過期，作業可能會失敗。

- 如需檢查目前工作階段權杖到期時間的詳細資訊，請參閱 "AWS API 文件"。
- 如需 AWS 資源認證的詳細資訊，請參閱 "AWS IAM 文件"。

註

當您使用 Kopia 作為資料移動器還原備份時，您可以選擇在 CR 中指定註解或使用 CLI 來控制 Kopia 使用的暫存的行為。請參閱 "Kopia 文件"有關您可以配置的選項的詳細資訊。使用 `tridentctl-protect create --help`有關使用Trident Protect CLI 指定註釋的更多信息，請參閱命令。

使用 **CR**

步驟

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.destinationApplicationName**：（選用）還原應用程式的名稱。如果提供、還原的應用程式會使用此名稱。如果未提供、還原的應用程式會使用來源應用程式名稱。
   - **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: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace",
"destination": "my-destination-namespace"}]
```

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：

註	Trident Protect 會自動選擇一些資源，因為它們與您選擇的資源有關聯。例如，如果您選擇持久性磁碟區宣告資源且它有一個關聯的 pod， Trident Protect 也會還原關聯的 pod。

   - **resourceFilter.resourceSelectionCriteria** ：（篩選所需）使用 `Include`或包含或 `Exclude`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** ：一組 resourceMatcher 物件。如果您在此陣列中定義多個元素，它們會比對為 OR 作業，而每個元素（群組，種類，版本）內的欄位會比對為 AND 作業。
       - **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-1
        kind: my-resource-kind-1
```

```
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        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

步驟

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> \
--destination-app-name<custom_app_name>\
-n <application_namespace>
```

從備份還原至原始命名空間

您可以隨時將備份還原到原始命名空間。執行就地還原時，Trident Protect 會自動管理保護排程和進行中的作業，以防止無效的復原點：

- 在還原開始之前，應用程式的所有已啟用保護排程都會停用。這可防止在還原應用程式資源時執行排程的備份或快照。
- 恢復成功完成後，只有恢復前已啟用的排程會重新啟用。先前已停用的排程仍保持停用狀態。
- 任何正在進行的備份或快照操作都會在還原開始前取消。如果操作在 5 分鐘內未取消，則還原將繼續進行，並在還原 CR 狀態中記錄警告。

開始之前

確保 AWS 工作階段權杖到期時間足以執行任何長時間執行的 S3 還原作業。如果 Token 在還原作業期間過期，作業可能會失敗。

- 如需檢查目前工作階段權杖到期時間的詳細資訊，請參閱 "AWS API 文件"。

- 如需 AWS 資源認證的詳細資訊，請參閱 "AWS IAM 文件"。

註	當您使用 Kopia 作為資料移動器還原備份時，您可以選擇在 CR 中指定註解或使用 CLI 來控制 Kopia 使用的暫存的行為。請參閱 "Kopia 文件"有關您可以配置的選項的詳細資訊。使用 `tridentctl-protect create --help`有關使用Trident Protect CLI 指定註釋的更多信息，請參閱命令。

使用 **CR**

步驟

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：

註	Trident Protect 會自動選擇一些資源，因為它們與您選擇的資源有關聯。例如，如果您選擇持久性磁碟區宣告資源且它有一個關聯的 pod， Trident Protect 也會還原關聯的 pod。

   - **resourceFilter.resourceSelectionCriteria** ：（篩選所需）使用 `Include`或包含或 `Exclude`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** ：一組 resourceMatcher 物件。如果您在此陣列中定義多個元素，它們會比對為 OR 作業，而每個元素（群組，種類，版本）內的欄位會比對為 AND 作業。

- **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-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        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

步驟

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> \
-n <application_namespace>
```

如果原始叢集發生問題，您可以將備份還原至不同的叢集。

註

- 當您使用 Kopia 作為資料移動器還原備份時，您可以選擇在 CR 中指定註解或使用 CLI 來控制 Kopia 使用的暫存的行為。請參閱 "Kopia 文件"有關您可以配置的選項的詳細資訊。使用 `tridentctl-protect create --help`有關使用Trident Protect CLI 指定註釋的更多信息，請參閱命令。
- 使用 CR 還原到新命名空間時，必須先手動建立目標命名空間，然後再套用 CR。Trident Protect 僅在使用 CLI 時才會自動建立命名空間。

開始之前

確保符合下列先決條件：

- 目標叢集已安裝Trident Protect。
- 目的地叢集可存取與儲存備份的來源叢集相同 AppVault 的儲存區路徑。
- 執行 AppVault CR 時，請確保本機環境可以連接到 AppVault CR 中定義的物件儲存桶。 `tridentctl-protect get appvaultcontent`命令。如果網路限制阻止訪問，請改為從目標叢集上的 pod 內執行Trident Protect CLI。
- 確保 AWS 工作階段權杖到期時間足以執行任何長時間執行的還原作業。如果 Token 在還原作業期間過期，作業可能會失敗。
  - 如需檢查目前工作階段權杖到期時間的詳細資訊，請參閱 "AWS API 文件"。
  - 如需 AWS 資源認證的詳細資訊，請參閱 "AWS 文件"。

步驟

1. 使用 Trident Protect CLI 外掛程式驗證 AppVault CR 是否存在於目標叢集上：

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

   註 如果目標叢集上不存在 AppVault CR，請按照 "使用Trident Protect AppVault 物件來管理儲存桶。" 中的步驟建立它。

2. 查看目標叢集上可用 AppVault 的備份內容，並記下 `appArchivePath`要還原的備份：

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

   執行此命令會顯示 AppVault 中的可用備份，包括其原始叢集，對應的應用程式名稱，時間戳記和歸檔路徑。

   - 輸出範例： *

```
+------------+----------+--------+-----------------
+--------------------------+-------------+
|   CLUSTER   |    APP    |  TYPE  |      NAME       |         TIMESTAMP
|    PATH     |
+------------+----------+--------+-----------------
+--------------------------+-------------+
| production1 | wordpress | backup | wordpress-bkup-1| 2024-10-30
08:37:40 (UTC)| backuppath1 |
| production1 | wordpress | backup | wordpress-bkup-2| 2024-10-30
08:37:40 (UTC)| backuppath2 |
+------------+----------+--------+-----------------
+--------------------------+-------------+
```

3. 使用 AppVault 名稱和歸檔路徑將應用程式還原至目的地叢集：

註	使用 CR 時，請確保用於應用程式還原的命名空間存在於目的地叢集上。

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.appVaultRef** ：（ _required _ ）儲存備份內容的 AppVault 名稱。
   - **spec.appArchivePath**：（必填）AppVault 內儲存備份內容的路徑。使用步驟 2 中的指令查看備份內容，並找到 `appArchivePath`要還原的備份。
   - **spec.destinationApplicationName**：（選用）還原應用程式的名稱。如果提供、還原的應用程式會使用此名稱。如果未提供、還原的應用程式會使用來源應用程式名稱。
   - **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`。

     例如：

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-backup-path
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
```

```
destination": "my-destination-namespace"}]
```

3. 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. 使用下列命令還原應用程式，將方括號中的值取代為您環境中的資訊。命名空間對應引數使用以冒號分隔的命名空間，將來源命名空間對應到正確的目的地命名空間，格式為 source1:dest1 ，source2:dest2 。例如：

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--destination-app-name <custom_app_name> \
--context <destination_cluster_name> \
-n <application_namespace>
```

從快照還原至不同的命名空間

您可以使用自訂資源 (CR) 檔案從快照還原數據，還原到不同的命名空間或原始來源命名空間。當您使用 SnapshotRestore CR 將快照還原到不同的命名空間時， Trident Protect 會在新的命名空間中還原應用程序，並為還原的應用程式建立應用程式 CR。為了保護已還原的應用程序，可以建立按需備份或快照，或製定保護計劃。

註

- SnapshotRestore 支持 `spec.storageClassMapping`屬性，但僅當來源和目標儲存類別使用相同的儲存後端。如果您嘗試恢復到 `StorageClass`如果使用不同的儲存後端，則復原操作將會失敗。
- 使用 CR 還原到新命名空間時，必須先手動建立目標命名空間，然後再套用 CR。Trident Protect 僅在使用 CLI 時才會自動建立命名空間。

開始之前

確保 AWS 工作階段權杖到期時間足以執行任何長時間執行的 S3 還原作業。如果 Token 在還原作業期間過期，作業可能會失敗。

- 如需檢查目前工作階段權杖到期時間的詳細資訊，請參閱 "AWS API 文件"。
- 如需 AWS 資源認證的詳細資訊，請參閱 "AWS IAM 文件"。

使用 **CR**

步驟

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 <SNAPHOT_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.destinationApplicationName**：（選用）還原應用程式的名稱。如果提供、還原的應用程式會使用此名稱。如果未提供、還原的應用程式會使用來源應用程式名稱。
   - **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: my-destination-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

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：

註	Trident Protect 會自動選擇一些資源，因為它們與您選擇的資源有關聯。例如，如果您選擇持久性磁碟區宣告資源且它有一個關聯的 pod， Trident Protect 也會還原關聯的 pod。

   - **resourceFilter.resourceSelectionCriteria** ：（篩選所需）使用 `Include`或包含或 `Exclude`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** ：一組 resourceMatcher 物件。如果您在此陣列中定義多個元素，它們會比對為 OR 作業，而每個元素（群組，種類，版本）內的欄位會比對為 AND 作業。
       - **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-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        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

步驟

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> \
--destination-app-name <custom_app_name> \
```

```
-n <application_namespace>
```

從快照還原至原始命名空間

您可以隨時將快照還原到原始命名空間。執行就地還原時，Trident Protect 會自動管理保護排程和進行中的作業，以防止無效的還原點：

- 在還原開始之前，應用程式的所有已啟用保護排程都會停用。這可防止在還原應用程式資源時執行排程的備份或快照。
- 恢復成功完成後，只有恢復前已啟用的排程會重新啟用。先前已停用的排程仍保持停用狀態。
- 任何正在進行的備份或快照操作都會在還原開始前取消。如果操作在 5 分鐘內未取消，則還原將繼續進行，並在還原 CR 狀態中記錄警告。

開始之前

確保 AWS 工作階段權杖到期時間足以執行任何長時間執行的 S3 還原作業。如果 Token 在還原作業期間過期，作業可能會失敗。

- 如需檢查目前工作階段權杖到期時間的詳細資訊，請參閱 "AWS API 文件"。
- 如需 AWS 資源認證的詳細資訊，請參閱 "AWS IAM 文件"。

使用 **CR**

步驟

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
```

+ .（可選）如果您只需要選擇應用程式的某些資源進行恢復，請新增篩選條件，以包含或排除帶有特定標籤的資源：

+ 注意： Trident Protect 會自動選擇某些資源，因為它們與您選擇的資源有關聯。例如，如果您選擇了持續性磁碟區宣告資源，且該資源關聯了一個 Pod，則 Trident Protect 也會還原該關聯 Pod。

+ * **resourceFilter.resourceSelectionCriteria**: （過濾必要）使用 `Include`或 `Exclude`來包含或排除在 `resourceMatchers` 中定義的資源。新增下列 `resourceMatchers` 參數來定義要包含或排除的資源： **`resourceFilter.resourceMatchers`：`resourceMatcher`** 物件的陣列。如果在此陣列中定義多個元素，它們之間將按 **`OR`** 運算匹配，每個元素內的欄位（**`group`**、**`kind`**、**`version`**）之間將按 **`AND`** 運算匹配。* **`resourceMatchers[].group`**: （可選）要篩選的資源群組。* **`resourceMatchers[].kind`**: （可選）要篩選的資源種類。* **`resourceMatchers[].version`**: （可選）要篩選的資源版本。* **`resourceMatchers[].names`**: （可選）要篩選的資源的 **`Kubernetes metadata.name`** 欄位中的名稱。* **`resourceMatchers[].namespaces`**: （可選）要篩選的資源的 `Kubernetes metadata.name` 欄位中的命名空間。* **`resourceMatchers[].labelSelectors`**: （可選）資源的 `Kubernetes metadata.name` 欄位中的標籤選擇器字串，如 "Kubernetes文件"中所定義。例如： `"trident.netapp.io/os=linux"`。

+ 例如：

+

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

+ .在 `trident-protect-snapshot-ipr-cr.yaml` 檔案中填入正確的值後，套用 CR：

+

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

使用CLI

步驟

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 <namespace/snapshot_to_restore> \
-n <application_namespace>
```

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}'
```

使用進階**Trident Protect** 恢復設定

您可以使用進階設定（例如註解、命名空間設定和儲存選項）自訂復原操作，以滿足您的特定要求。

還原和容錯移轉作業期間的命名空間註釋和標籤

在還原和容錯移轉作業期間，目的地命名空間中的標籤和註釋會與來源命名空間中的標籤和註釋相符。會新增來源命名空間中不存在的標籤或註釋，並覆寫已存在的任何標籤或註釋，以符合來源命名空間的值。只存在於目的地命名空間上的標籤或註釋會保持不變。

> 註 如果您使用 Red Hat OpenShift，請務必注意命名空間註解在 OpenShift 環境中的重要角色。命名空間註解可確保復原的 pod 遵守 OpenShift 安全性情境約束 (SCC) 定義的適當權限和安全性配置，並且可以存取磁碟區而不會出現權限問題。欲了解更多信息，請參閱"OpenShift 安全性內容限制文件"。

您可以在執行還原或容錯移轉作業之前，先設定 Kubernetes 環境變數，以避免覆寫目的地命名空間中的特定註釋 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`。例如：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

> 註 執行復原或故障轉移操作時，任何命名空間註解和標籤都將生效。`restoreSkipNamespaceAnnotations` 和 `restoreSkipNamespaceLabels` 不參與恢復或

故障轉移操作。確保在初始 Helm 安裝期間配置這些設定。欲了解更多信息，請參閱 "設定其他Trident Protect 舵圖設置"。

如果您使用 Helm 安裝了來源應用程序， `--create-namespace`國旗，給予特殊待遇 `name`標籤鍵。在復原或故障轉移過程中， Trident Protect 會將此標籤複製到目標命名空間，但如果來源命名空間的值與來源命名空間的值匹配，則會將值更新為目標命名空間的值。如果此值與來源命名空間不匹配，則會將其複製到目標命名空間，而不做任何變更。

範例

以下範例提供來源和目的地命名空間，每個命名空間都有不同的註釋和標籤。您可以查看作業前後目的地命名空間的狀態，以及註釋和標籤在目的地命名空間中的組合或覆寫方式。

還原或容錯移轉作業之前

下表說明還原或容錯移轉作業之前的範例來源和目的地命名空間狀態：

命名空間	註釋	標籤
命名空間 nS-1 （來源）	• annotation.one / 機碼：「 updatedvalue 」 • annotation.b2/key ：「 true 」	• 環境 = 正式作業 • Compliance = HIPAA • NAME=ns-1
命名空間 nS-2 （目的地）	• annotation.one / 機碼：「 true 」 • annotation.the/key ：「 FALSE 」	• role = 資料庫

還原作業之後

下表說明還原或容錯移轉作業之後範例目的地命名空間的狀態。某些金鑰已新增，部分已覆寫， `name`標籤已更新以符合目的地命名空間：

命名空間	註釋	標籤
命名空間 nS-2 （目的地）	• annotation.one / 機碼：「 updatedvalue 」 • annotation.b2/key ：「 true 」 • annotation.the/key ：「 FALSE 」	• NAME=nS-2 • Compliance = HIPAA • 環境 = 正式作業 • role = 資料庫

支援的字段

本節介紹可用於恢復操作的其他欄位。

儲存類別映射

這 `spec.storageClassMapping`屬性定義從來源應用程式中的現有儲存類別到目標叢集上的新儲存類別的對應。您可以在具有不同儲存類別的叢集之間移轉應用程式時或變更 BackupRestore 作業的儲存後端時使用此功能。

範例：

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

支持的註釋

本節列出了系統中支援配置各種行為的註解。如果使用者未明確設定註解，系統將使用預設值。

註釋	類型	說明	預設值
protected.trident.netapp.io/data-mover-timeout-sec	字串	允許資料移動設備操作停止的最長時間（以秒為單位）。	“300”
protected.trident.netapp.io/kopia-content-cache-size-limit-mb	字串	Kopia 內容快取的最大大小限制（以兆位元組為單位）。	“1000”
protect.trident.netapp.io/pvc-bind-timeout-sec	字串	等待新建立的持久卷聲明 (PVC) 到達的最大時間（以秒為單位）`Bound`操作失敗前的階段。適用於所有還原 CR 類型（備份還原、備份就地還原、快照還原、快照就地還原）。如果您的儲存後端或叢集經常需要更多時間，請使用更高的值。	1200（20分鐘）
protect.trident.netapp.io/retain-mac-address	字串	對於 KubeVirt 虛擬機器，請設定為 `"true"`以在還原時保留第一個網路介面的 MAC 位址，例如當應用程式授權綁定到該介面時。適用於所有還原 CR 類型以及 AppMirrorRelationship （請在提升關係之前設定）。預設情況下， Trident Protect 會移除此 MAC 位址以防止衝突。如果來源虛擬機器仍在同一網路上運行，請先將其關閉或還原到隔離的網路，以避免複製的 MAC 位址。	未設定（MAC 位址已移除）

使用自訂資源還原資源時修改資源

資源轉換功能可讓您在復原資源的過程中進行修改。當復原後的版本需要與原始版本有所不同時，此功能非常有用，例如，在將應用程式還原到不同的叢集時，更新 ConfigMap 中的主機名稱。

資源修改的運作方式

The `transformations` 欄位位於 `SnapshotRestore`、 `BackupRestore`、 `AppMirrorRelationship` 及其他復原資源中，可讓您在復原過程中修改 Kubernetes 資源。當復原後的應用程式需要與原始應用程式採用不同的組態時，例如變更主機名稱、登錄 URL、資源限制或環境變數，此功能非常實用。

資源轉換使用 "RFC 6902 JSON Patch"操作和 "RFC 6901 JSON Pointer"路徑來定位和修改 Kubernetes 資源中的特定欄位。

下面展示了包含資源轉換的還原物件的基本結構：

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: my-restore
  namespace: target-namespace
spec:
  appVaultRef: my-vault
  appArchivePath: /path/to/snapshot
  namespaceMapping:
    - source: source-ns
      destination: target-ns
  transformations:
    - resource:
        kind: ConfigMap       # Required: resource kind
        group: ""             # Optional: API group (empty for core
resources)
        version: ""           # Optional: API version
        name: ""              # Optional: specific resource name
      operations:
        - op: replace         # Operation type
          path: "/data/key"   # JSON Pointer path
          value: "new-value"  # New value (for add/replace/test)
```

支援的資源

您可以將資源轉換套用至符合下列條件的資源：

- **kind**（必填）：Kubernetes 資源類型（例如， `ConfigMap` ， `Deployment` ， `Pod`）
- **group**（可選）：API 群組（例如， `apps` ， `route.openshift.io`）— 對於核心資源，請省略
- 版本（可選）：API 版本（例如， `v1` ， `v1beta1`）
- 名稱（可選）：僅套用於指定名稱的資源

> 註　目前不支援修改 PersistentVolumeClaims 和命名空間。

支援的作業

您可以使用以下操作來修改資源： `add` ， `copy` ， `move` ， `remove` ， `replace` ，以及 `test` 。

為資源新增值

使用 `add` 操作可在指定路徑新增欄位或值。您可以為物件或陣列新增資料。

使用 **CR**

以下範例向 `Deployment` 資源新增節點選擇器：

```
transformations:
  - resource:
      kind: Deployment
    operations:
      - op: add
        path: "/spec/template/spec/nodeSelector"
        value:
          "topology.kubernetes.io/zone": "us-east-1a"
          disktype: "ssd"
```

使用**CLI**

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
'apps,v1,Deployment:add{"path":"/spec/template/spec/nodeSelector","valu
e":{"topology.kubernetes.io/zone":"us-east-1a","disktype":"ssd"}}'
```

複製資源中的值

使用 `copy` 操作可將相同資源內的值從一個路徑複製到另一個路徑。來源值不會改變。

使用 **CR**

以下範例複製 `ConfigMap` 物件的資料鍵：

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: copy
        from: "/data/source-key"
        path: "/data/backup-key"
```

使用**CLI**

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation ',v1,ConfigMap:copy{"from":"/data/source-
key","path":"/data/backup-key"}'
```

在資源內移動值

使用 `move`操作可將相同資源中的值從一個路徑移至另一個路徑。來源路徑將被移除，值將被放置在目標路徑中。

使用 **CR**

以下範例重新命名 `ConfigMap` 物件的資料鍵：

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: move
        from: "/data/OLD_KEY"
        path: "/data/NEW_KEY"
```

使用**CLI**

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:move{"from":"/data/OLD_KEY","path":"/data/NEW_KEY"}'
```

從資源中移除值

使用 `remove` 操作可刪除指定路徑中的欄位或值。

使用 **CR**

以下範例從 `ConfigMap`資源中刪除註解：

```
transformations:
  - resource:
```

```
      kind: ConfigMap
    operations:
      - op: remove
        path: "/metadata/annotations/kubectl.kubernetes.io~1last-
applied-configuration"
```

註 在上述範例的路徑中、 `~1` 是 `/` 的 JSON Pointer 轉義序列。

使用CLI

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:remove{"path":"/metadata/annotations/kubectl.kubernetes.
io~1last-applied-configuration"}'
```

取代資源中的值

使用 `replace` 操作可取代指定路徑下資源中的現有值。JSON 路徑必須已存在。

使用 CR

以下範例變更 `Route` 物件的主機名稱：

```
transformations:
  - resource:
      kind: Route
      group: route.openshift.io
    operations:
      - op: replace
        path: "/spec/host"
        value: "prod.example.com"
```

使用CLI

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
```

```
'route.openshift.io,v1,Route:replace{"path":"/spec/host","value":"prod.
example.com"}'
```

測試資源修改

使用 `test` 操作，在執行其他操作之前驗證路徑處的值是否與預期值相符。如果測試失敗，整個修改將會回滾。

使用 **CR**

以下範例僅在 `environment` 設定為 `staging` 時，才會更新 `database-host`：

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: test
        path: "/data/environment"
        value: "staging"
      - op: replace
        path: "/data/database-host"
        value: "prod-db.example.com"
```

使用**CLI**

使用以下命令透過命令列執行此轉換：

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:test{"path":"/data/environment","value":"staging"},repla
ce{"path":"/data/database-host","value":"prod-db.example.com"}'
```

使用 NetApp SnapMirror 複寫應用程式

使用 **Trident Protect** 設定和管理應用程式複寫

使用Trident Protect，您可以利用NetApp SnapMirror技術的非同步複製功能，將資料和應用程式變更從一個儲存後端複製到另一個儲存後端，無論是在同一叢集內還是在不同叢集之間。

在還原和容錯移轉作業期間，目的地命名空間中的標籤和註釋會與來源命名空間中的標籤和註釋相符。會新增來源命名空間中不存在的標籤或註釋，並覆寫已存在的任何標籤或註釋，以符合來源命名空間的值。只存在於目的地命名空間上的標籤或註釋會保持不變。

> 註 如果您使用 Red Hat OpenShift，請務必注意命名空間註解在 OpenShift 環境中的重要角色。命名空間註解可確保復原的 pod 遵守 OpenShift 安全性情境約束 (SCC) 定義的適當權限和安全性配置，並且可以存取磁碟區而不會出現權限問題。欲了解更多信息，請參閱"OpenShift 安全性內容限制文件"。

您可以在執行還原或容錯移轉作業之前，先設定 Kubernetes 環境變數，以避免覆寫目的地命名空間中的特定註釋 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`。例如：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

> 註 執行復原或故障轉移操作時，任何命名空間註解和標籤都將生效。`restoreSkipNamespaceAnnotations` 和 `restoreSkipNamespaceLabels` 不參與恢復或故障轉移操作。確保在初始 Helm 安裝期間配置這些設定。欲了解更多信息，請參閱 "設定其他Trident Protect 舵圖設置"。

如果您使用 Helm 安裝了來源應用程序， `--create-namespace`國旗，給予特殊待遇 `name`標籤鍵。在復原或故障轉移過程中， Trident Protect 會將此標籤複製到目標命名空間，但如果來源命名空間的值與來源命名空間的值匹配，則會將值更新為目標命名空間的值。如果此值與來源命名空間不匹配，則會將其複製到目標命名空間，而不做任何變更。

範例

以下範例提供來源和目的地命名空間，每個命名空間都有不同的註釋和標籤。您可以查看作業前後目的地命名空間的狀態，以及註釋和標籤在目的地命名空間中的組合或覆寫方式。

還原或容錯移轉作業之前

下表說明還原或容錯移轉作業之前的範例來源和目的地命名空間狀態：

命名空間	註釋	標籤
命名空間 nS-1 （來源）	• annotation.one / 機碼：「 updatedvalue 」 • annotation.b2/key ：「 true 」	• 環境 = 正式作業 • Compliance = HIPAA • NAME=ns-1
命名空間 nS-2 （目的地）	• annotation.one / 機碼：「 true 」 • annotation.the/key ：「 FALSE 」	• role = 資料庫

還原作業之後

下表說明還原或容錯移轉作業之後範例目的地命名空間的狀態。某些金鑰已新增，部分已覆寫， `name`標籤已更新以符合目的地命名空間：

命名空間	註釋	標籤
命名空間 nS-2 （目的地）	• annotation.one / 機碼：「 updatedvalue 」 • annotation.b2/key ：「 true 」 • annotation.the/key ：「 FALSE 」	• NAME=nS-2 • Compliance = HIPAA • 環境 = 正式作業 • role = 資料庫

註 | 您可以設定Trident Protect 在資料保護作業期間凍結和解凍檔案系統。"了解更多關於使用Trident Protect 設定檔系統凍結的信息"。

故障轉移和反向操作期間的執行掛鉤

當使用 AppMirror 關係保護您的應用程式時，您應該在故障轉移和反轉操作期間注意與執行掛鉤相關的特定行為。

- 在故障轉移期間，執行掛鉤會自動從來源叢集複製到目標叢集。您無需手動重新建立它們。故障轉移後，執行掛鉤仍存在於應用程式中，並將在任何相關操作期間執行。
- 在反向同步或反向重新同步期間，應用程式上所有現有的執行鉤子都將被移除。當來源應用程式成為目標應用程式時，這些執行鉤子將失效，並將被刪除以阻止其執行。

要了解有關執行鉤子的更多信息，請參閱"管理Trident Protect 執行鉤子"。

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 資源以及應用程式每個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. 在來源叢集上，為來源應用程式建立 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> -n
trident-protect
```

2. 在來源叢集上，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: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
      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 <my-app-name> --namespaces
<namespaces-to-be-included> -n <my-app-namespace>
```

3. （可選）在來源叢集上，對來源應用程式進行快照。此快照將用作目標叢集上應用程式的基礎。如果跳過此步驟，則需要等待下一次排程快照運行，以便取得最新的快照。若要建立隨選快照，請參閱 "建立隨需快照"。

4. 在來源叢集上，建立複製計劃 CR：

註

除了下面提供的計劃外，建議建立一個單獨的每日快照計劃，保留期為 7 天，以便在對等 ONTAP 叢集之間維護通用快照。這可確保快照最多可用 7 天，但保留期可根據使用者需求自訂。

如果發生故障轉移，系統可以使用這些快照最多 7 天進行反向操作。這種方法使反向過程更快、更有效率，因為只會傳輸自上次快照以來所做的更改，而不是所有資料。

如果應用程式的現有計劃已經滿足所需的保留要求，則不需要額外的計劃。

使用 **CR** 建立複製計劃

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**: (必需) 此值必須與來源應用程式的 AppVault 的 metadata.name 欄位相符。
   - **spec.applicationRef**: (必需) 此值必須與來源應用程式 CR 的 metadata.name 欄位相符。
   - *spec.backupRetention * ：（ _required _ ）此欄位為必填欄位，且值必須設為 0 。
   - **spec.enabled** ：必須設置爲 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
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  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** 建立複製計劃

a. 建立複製計劃，並將括號中的值替換為您環境中的資訊：

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

範例：

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. 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`）。
6. 套用 CR ：

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
trident-protect
```

7. 為目的地叢集上的目的地應用程式建立目的地 AppVault CR 。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 trident-protect
```

8. 在目標叢集上，建立 AppMirrorRelationship CR 檔案。

註	使用 CR 時，請在套用 CR 之前手動建立目標命名空間。Trident Protect 僅在使用 CLI 時才會自動建立命名空間。

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。
- **metadata.annotations.protect.trident.netapp.io/retain-mac-address**：（選用）設定為 `"true"` 以保留目的地叢集上 KubeVirt 虛擬機器的 MAC 位址。請參閱 "使用進階Trident Protect 恢復設定"。
- * 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.sourceApplicationUID**: （必要）此值必須與您在來源應用程式 CR 中定義的來源應用程式的 UID 相符。
- **spec.storageClassName**: (可選) 選擇叢集上有效的儲存類別的名稱。儲存類別必須連結到與來源環境建立對等連線的ONTAP儲存 VM。如果未提供儲存類，則預設使用叢集上的預設儲存類別。
- **spec.storageClassMapping**: （可選）將同一 AMR 內各個 PVC 的來源儲存類別對應到目標儲存類別。每個條目都需要一個 `source`和 `destination`儲存類別名稱。設定後，`storageClassMapping`將完全覆蓋 `storageClassName`；沒有符合來源條目的 PVC 將使用叢集預設儲存類別，而非 `storageClassName`。

在 AppMirrorRelationship CR 中使用 `storageClassMapping`時，請注意以下限制：

- `storageClassMapping` 覆蓋 `storageClassName` 的全部內容。當兩個欄位都設定時，`storageClassName` 將被忽略。
- 來源儲存類別在 `storageClassMapping`中沒有相符條目的 `PVC` 將使用叢集預設儲存類別，而非 `storageClassName`。
- 每個條目都需要一個 `source`和 `destination`值。
- 更改目標儲存類別可能會強制刪除並重新建立現有的目標 PVC。

- **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: my-app-name
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsim-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 對象，並將括號中的值替換為您環境中的資訊：

```
tridentctl-protect create appmirrorrelationship
```

```
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
-rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
-class <storage_class_name> --storageclass-mapping
<src_class:dst_class> -n <application_namespace>
```

註	使用 `--storageclass-mapping`將各個來源儲存類別對應到目標儲存類別。指定一個或多個 `source:destination`以逗號分隔的鍵值對（例如，`"class-a:class-b,class-c:class-d"）。使用`*:destination-class`作為萬用字元，可將所有未對應的 PVC 對應到單一目標儲存類別。當`--storageclass-mapping`設定後，`--storage-class`將被忽略。

使用單一目標儲存類別的範例 (**`--storage-class`**）：

```
tridentctl-protect create appmirrorrelationship my-amr
--destination-app-vault appvault2 --source-app-vault appvault1
--recurrence-rule
"DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5"
--source-app my-app --namespace-mapping "my-source-ns1:my-dest-
ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-
404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-
dest-ns1
```

9. （*Optional*）在目的地叢集上，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，您可以將複製的應用程式故障轉移到目標叢集。此過程會停止複製關係，並將應用程式在目標叢集上連線。如果來源叢集上的應用程式正在運行， Trident Protect 不會停止該應用程式。

此程序會容錯移轉至最新的快照。若要容錯移轉至較舊的快照，請參閱 "使用 Trident Protect 容錯移轉到較早的快照"。

步驟

1. （選用）若要保留 KubeVirt 虛擬機器的 MAC 位址，請在提升 AppMirrorRelationship 之前對其進行註解。請參閱 "使用進階Trident Protect 恢復設定"。

```
kubectl annotate appmirrorrelationship <relationship_name> -n
<application_namespace> protect.trident.netapp.io/retain-mac-address
```

```
="true"
```

2. 在目標叢集上，編輯 AppMirrorRelationship CR 檔案（例如 `trident-protect-relationship.yaml`），並將 * spec.desiredState* 的值變更為 `Promoted`。
3. 儲存 CR 檔案。
4. 套用 CR ：

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. （ *Optional* ）在容錯移轉應用程式上建立所需的任何保護排程。
6. （ *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。

重要	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. 選用：在來源叢集上，建立來源應用程式的快照。如此可確保擷取來源叢集的最新變更。
2. 在目標叢集上，編輯 AppMirrorRelationship CR 檔案（例如 `trident-protect-relationship.yaml`），並將 spec.desiredState 的值變更為 `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. 在原始目的地叢集上，刪除 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. 請確定新目的地（原始來源叢集）已同時使用 AppVault CRS 進行設定。
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

當您反轉複製方向時， Trident Protect 會將應用程式移至目標儲存後端，同時繼續複製回原始來源儲存後端。 Trident Protect 會停止來源應用程式並將資料複製到目標位置，然後再故障轉移到目標應用程式。

在這種情況下、您要交換來源和目的地。

步驟

1. 在來源叢集上，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: my-app-name
```

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> -n <application_namespace>
```

2. 在來源叢集上，關機快照完成後，取得關機快照的狀態：

```
kubectl get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o yaml
```

3. 在來源叢集上，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 ；這將是最後一條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，從新的目的地叢集移轉至新的來源叢集，並進行下列變更：

註	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，您可以透過以下步驟序列在故障轉移作業後實現「故障復原」。在此恢復原始複製方向的工作流程中， Trident Protect 會將任何應用程式變更複製（重新同步）回原始來源應用程序，然後再反轉複製方向。

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。

> 警告 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. 在目前的目標叢集上，刪除 AppMirrorRelationship CR ：

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

使用 **Trident Protect** 容錯移轉到較早的快照

使用 Trident Protect 將複寫的應用程式容錯移轉到較早的快照，從而可以從資料損壞或洩漏之前的時間點恢復。預設情況下，容錯移轉使用最近傳輸的快照。

> 警告 容錯移轉至較舊的快照會導致 ONTAP SnapMirror 刪除目的地叢集上比所選快照更新的快照，因為它們已不再有效。只有在可接受此資料遺失的情況下，才在此程序中新增註釋。

開始之前

- 存在 AppMirrorRelationship 且處於 `Established`狀態。若要建立，請參閱 "設定複寫關係"。若要驗證狀態，請執行 `kubectl get amr -n <namespace> <relationship_name> -o=jsonpath='{.status}'`。
- 來源和目的地叢集使用 ONTAP 儲存設備後端搭配 SnapMirror 複寫。不支援 Azure NetApp Files 和其他非 ONTAP 後端。請參閱 "SnapMirror 複寫需求"。
- 來源應用程式上存在快照排程（ `snapshotRetention` 大於 1）。

識別快照

每個快照都有一個完成時間，因此您可以選擇所需的時間點。

步驟

1. 在來源叢集上，列出快照：

```
kubectl get snapshots -n my-app-namespace
```

2. 找到要容錯移轉到的快照。使用 `completionTimestamp` 值選擇時間點。
3. 取得該快照的歸檔路徑：

```
kubectl get snapshots <snapshot_name> -n my-app-namespace -o jsonpath
='{.status.appArchivePath}'
```

4. 記錄路徑的最後一部分，即最後一個斜線之後的所有內容。這部分是 *basename*。例如：

```
20231114173500_schedule-20231114173500_f8df8bd3-a727-4a14-b954-
9666aa618b55
```

> 註 使用基本名稱，而不是快照自訂資源的名稱。

結果

記錄所選快照的基本名稱。

容錯移轉至選定的快照

指定要提升的快照，並透過新增註解確認資料遺失。

步驟

1. 在目的地叢集上，將註解加入 AppMirrorRelationship。如果沒有註解，容錯移轉只能使用最近傳輸的快照。

```
kubectl annotate appmirrorrelationship <relationship_name> -n my-app-
namespace protect.trident.netapp.io/allow-promotion-with-older-
snapshots="true"
```

2. 編輯 AppMirrorRelationship CR 檔案（例如， `trident-protect-relationship.yaml`）：
   - 將 **spec.desiredState** 設定為 `Promoted`。
   - 將 **spec.promotedSnapshot** 設定為您在 [識別快照] 中記錄的基本名稱。

     例如：

```
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: <relationship_name>
  namespace: <namespace>
  annotations:
    protect.trident.netapp.io/allow-promotion-with-older-snapshots:
"true"
spec:
  desiredState: Promoted
  promotedSnapshot: <basename>
```

3. 套用 CR ：

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. 檢查關係狀態：

```
kubectl get amr -n my-app-namespace <relationship_name> -o=jsonpath
='{.status}' | jq
```

結果

關係移至 `Promoted`狀態，且應用程式在目的地叢集上執行，並使用來自所選快照的資料。

如果關係保持在 `Promoting`狀態，且 `status.error`顯示 `The snapshot provided is invalid or older than the most recently transferred snapshot`，則表示缺少註解或未設定為 `true`。請新增註解並重試。

接下來呢？

若要返回原始叢集，請執行 "反向重新同步"，然後 "反轉複寫方向"。

警告	若要回到原始叢集，請勿執行正常重新同步。正常重新同步會捨棄容錯移轉的應用程式，並將關係還原至最新的來源快照。

相關資訊

- "使用 Trident Protect 設定和管理應用程式複寫"
- "容錯移轉至目的地叢集"
- "重新同步容錯移轉複寫關係"
- "將應用程式容錯移轉至原始來源叢集"
- "建立資料保護排程"

使用Trident Protect 遷移應用程式

您可以透過還原備份資料在叢集之間或不同的儲存類別之間遷移您的應用程式。

> 註 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（執行跨叢集克隆），請在來源叢集上建立備份，然後將備份還原到不同的叢集。請確保目標叢集上已安裝Trident Protect。

> 註 您可以使用在不同叢集之間複寫應用程式"SnapMirror 複寫"。

步驟

1. "建立備份"。
2. 請確定已在目的地叢集上設定包含備份之物件儲存貯體的 AppVault CR 。
3. 在目的地叢集上"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

您可以透過將備份還原到目標儲存類，將應用程式從一個儲存類別遷移到另一個儲存類別。

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

步驟

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：
     - **resourceFilter.resourceMatchers** ：一組 resourceMatcher 物件。如果您在此陣列中定義多個元素，它們會比對為 OR 作業，而每個元素（群組，種類，版本）內的欄位會比對為 AND 作業。
       - **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-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        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

步驟

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>
```

使用 **Trident Protect** 保護 **KubeVirt** 虛擬機器

Trident Protect 可讓您備份與還原執行於 OpenShift Virtualization 上的 KubeVirt 虛擬機器（VM）。您可以同時保護一或多個命名空間中的所有 VM，或使用 `includedVirtualMachines` 欄位依名稱指定特定的 VM。Trident Protect 會在備份期間自動收集所有相依的 VM 資源，而在還原期間，您可以復原整個封存或選擇性地還原個別 VM。

KubeVirt 保護模式

Trident Protect 支援兩種定義要保護的 KubeVirt 虛擬機器的方式。定義應用程式時，請選擇其中一種模式：

模式	欄位	說明
基於命名空間	`includedNamespaces`	保護一個或多個命名空間中所有符合的資源。在命名空間條目上使用 `labelSelector`來動態控制包含哪些虛擬機器。
虛擬機器範圍	`includedVirtualMachines`	依命名空間和名稱保護特定 KubeVirt VM。當 VM 分佈在多個命名空間中，或您不想保護所有命名空間資源時，請使用此模式。

您必須在 `ApplicationSpec` 中指定恰好一個欄位（`includedNamespaces` 或 `includedVirtualMachines`）。您無法在同一個應用程式中同時使用兩者。

兩種模式均支援額外的篩選功能：

- 使用 `resourceFilter`來包含或排除特定資源。
- 使用 `includedClusterScopedResources`以包含相容的叢集範圍資源，例如 `StorageClass`物件。

註

- Trident Protect 可在資料保護作業期間凍結和解凍 KubeVirt 虛擬機器檔案系統，以確保資料一致性。如需組態詳細資料，請參閱 "使用 KubeVirt VM 保護資料"。
- 若要在同一時間點為虛擬機器的所有磁碟建立快照，請使用 Volume 群組快照。請參閱 "搭配

Trident Protect 使用 Volume 群組快照"。

備份和快照期間收集的資源

當您為包含 KubeVirt VM 的應用程式建立備份或快照時，Trident Protect 會收集選定的 VM 資源並自動探索其相依資源。

收集行為適用於以下兩種情況：

- 使用 VM 範圍的應用程式 `includedVirtualMachines`。
- 基於命名空間且包含 KubeVirt `VirtualMachine` 資源的應用程式。

VM 範圍的收集

對於 VM 範圍的應用程式，Trident Protect 從 `VirtualMachine`中列出的每個 `includedVirtualMachines`開始，並探索還原該 VM 所需的相依資源。

資源類型	範圍	探索方式
`VirtualMachine`	命名空間	從 `includedVirtualMachines`清單中直接比對。
`VirtualMachineInstance`	命名空間	如果虛擬機器正在執行，則使用與虛擬機器相同的命名空間和名稱。
`VirtualMachineInstancetype`	命名空間	被引用 `spec.instancetype`時 `kind: VirtualMachineInstancetype`。
`VirtualMachineClusterInstancetype`	叢集	由 `spec.instancetype` 參照，當 `kind` 未設定或 `kind: VirtualMachineClusterInstancetype` 時。
`VirtualMachinePreference`	命名空間	被引用 `spec.preference`時 `kind: VirtualMachinePreference`。
`VirtualMachineClusterPreference`	叢集	由 `spec.preference` 參照，當 `kind` 未設定或 `kind: VirtualMachineClusterPreference` 時。
`DataVolume`	命名空間	被 `spec.template.spec.volumes[].dataVolume.name` 引用。
`PersistentVolumeClaim`	命名空間	從 DataVolume 支援的 PVC 中探索，這些 PVC 使用與 DataVolume 相同的名稱；直接 PVC 參照位於 `persistentVolumeClaim.claimName`；記憶體傾印 PVC 位於 `memoryDump.claimName`；以及暫時性 PVC 位於 `ephemeral.persistentVolumeClaim.claimName`。
`PersistentVolume`	叢集	透過使用 `spec.volumeName` 從每個收集的 PVC 的備援 PV 中探索。
`Secret`	命名空間	從磁碟區金鑰 `secret.secretName`中探索到；containerDisk 映像提取金鑰；CloudInitNoCloud 和 CloudInitConfigDrive 使用者資料和網路資料金鑰參考；Sysprep 金鑰；KernelBoot 容器映像提取金鑰；以及 AccessCredentials SSH 或密碼金鑰。

資源類型	範圍	探索方式
ConfigMap	命名空間	從磁碟區 ConfigMaps 中 `configMap.name`和 Sysprep ConfigMaps 中探索到。
ServiceAccount	命名空間	從 `serviceAccount.serviceAccountName`磁碟區參考中探索到。
NetworkAttachmentDefinition	命名空間	當網路名稱不包含 / 時，從 spec.template.spec.networks[].multus.networkName 中的 Multus 網路探索。僅收集相同命名空間的 NetworkAttachmentDefinition 參照。會跳過跨命名空間參照。
Pod	命名空間	透過標籤選擇器為 virt-launcher Pods 探索 kubevirt.io=virt-launcher,vm.kubevirt.io/name=<vmName>。
儲存類別、Trident 磁碟區和 Trident 後端	叢集	收集方式與基於命名空間的應用程式相同，以針對 VM 探索到的 PVC 為基礎。

註 在收集完所有虛擬機器範圍的命名空間資源後，Trident Protect 會套用標準 resourceFilter（如果已設定）。然後，叢集範圍的資源收集將照常進行。

基於命名空間的集合

對於基於命名空間的應用程式，Trident Protect 會掃描選定的命名空間並收集所有相符的資源。如果找到任何 KubeVirt VirtualMachine 物件，Trident Protect 會自動收集其相依資源，包括執行個體類型、偏好設定和 virt-launcher Pod，其方式與處理 VM 範圍應用程式的方式相同。

這意味著即使應用程式已使用 `includedNamespaces`定義，VM 特定資源也會包含在備份或快照中。

使用基於命名空間的應用程式定義保護虛擬機器

在建立基於命名空間的應用程式時，Trident Protect 會掃描選定的命名空間並收集所有資源，包括所有 KubeVirt 虛擬機器。對於找到的每個虛擬機器，Trident Protect 也會收集備份和還原該虛擬機器所需的相依資源。

基於命名空間的應用程式範例 CR：

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
```

建立應用程式後，請使用標準的 Trident Protect 工作流程進行備份和快照：

- 有關快照 YAML 範例，請參閱"建立隨需快照"。
- 有關備份 YAML 範例，請參閱"建立隨選備份"。

由於基於命名空間的應用程式會從選定的命名空間收集資源，因此備份或快照可能包含 KubeVirt 執行時間物件和 KubeVirt 管理的中繼資料。其中某些物件不應直接還原，因為它們代表來源叢集的執行時間狀態。

在復原過程中，Trident Protect 會套用 KubeVirt 感知轉換來移除來源叢集元資料並跳過暫存 KubeVirt 資源。這有助於 KubeVirt 在復原的環境中乾淨地重新建立虛擬機器。

資源	轉換
VirtualMachine	移除以 `kubevirt.io/`開頭的標籤和註釋。
VirtualMachineInstance	移除以 `kubevirt.io/`開頭的標籤和註釋。
VirtualMachineInstanceMigration	已忽略且未恢復。
VirtualMachineSnapshot	已忽略且未恢復。
VirtualMachineSnapshotContent	已忽略且未恢復。
Pod (`virt-launcher`)	當 pod 具有 `kubevirt.io=virt-launcher`標籤時跳過。

註

- Trident Protect 會依照預先定義的順序復原 KubeVirt 資源，以避免相依性問題。CDI DataVolumes (`cdi.kubevirt.io`) 會優先復原，以確保在建立虛擬機器物件之前，虛擬機器磁碟資料已可供使用。之後，VirtualMachines (`kubevirt.io`) 會在 VirtualMachineInstances 之前復原。
- 預設情況下， Trident Protect 會移除每個已還原虛擬機器的第一個網路介面的 MAC 位址。若要保留它，請使用 `protect.trident.netapp.io/retain-mac-address` 註解。請參閱 "使用進階Trident Protect 恢復設定"。

使用標籤選擇器進行動態虛擬機器保護

如果要動態保護 VM，但又不想列出每個 VM 名稱，請使用基於命名空間的應用程式定義和標籤選擇器。

只有 VirtualMachine CR 需要標籤。相依資源（例如 DataVolumes、PVC、Secrets 和 ConfigMaps）會自動收集。

步驟

1. 使用共用標籤標記虛擬機器 CR：

```
kubectl label virtualmachine <vm_name 1> app=<label_value> -n
<vm_namespace>
kubectl label virtualmachine <vm_name_2> app=<label_value> -n
<vm_namespace>
```

2. 建立含有 `includedNamespaces` 和 `labelSelector` 的應用程式 CR：

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
```

```
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
      labelSelector:
        matchLabels:
          app: <label_value>
```

3. 使用標準 Trident Protect 工作流程建立備份和還原。

其他詳細資料：

- 您可以隨時透過變更標籤來新增或移除受保護的虛擬機器：

```
kubectl label virtualmachine <vm_name_to_add> app=<label_value> -n
<vm_namespace>
kubectl label virtualmachine <vm_name_to_remove> app
=<different_label_value> -n <vm_namespace>
```

- 您也可以跨多個命名空間使用標籤選擇器：

```
spec:
  includedNamespaces:
    - namespace: <vm_namespace_1>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
    - namespace: <vm_namespace_2>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
```

使用 **includedVirtualMachines** 保護特定虛擬機器

當您想要保護一個或多個命名空間中的一組指定 VirtualMachines，並從歸檔中選擇性地還原特定虛擬機時，請使用 `includedVirtualMachines`。

應用程式和復原 CR 都使用相同的 `includedVirtualMachines`選擇器格式。

在不同的上下文中使用此選擇器：

- 在應用程式 CR 中，它定義了哪些虛擬機器受到保護。
- 在還原 CR 中，它定義了從備份或快照封存中還原哪些虛擬機器。

註	在還原 CR 時，此欄 `includedVirtualMachines`位為選填欄位。如果不包含此欄位，則會還原整

個封存。如果指定此欄位，則僅還原列出的虛擬機器及其相依資源。

以下 YAML 程式碼展示了共用選擇器結構：

```
includedVirtualMachines:
  - namespace: <vm_namespace_1>
    names:
      - <vm_name_1>
      - <vm_name_2>
  - namespace: <vm_namespace_2>
    names:
      - <vm_name_3>
```

註 多個條目位於 `includedVirtualMachines`中，可以參考不同的命名空間，從而實現多命名空間 VM 感知應用程式定義和還原。

欄位	類型	說明
命名空間	字串	包含目標虛擬機器物件的 Kubernetes 命名空間。
名稱	字串列表	此命名空間中的一個或多個虛擬機器名稱。

基於虛擬機器的應用程式範例：

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace_1>
      names:
        - <vm_name_1>
        - <vm_name_2>
    - namespace: <vm_namespace_2>
      names:
        - <vm_name_3>
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

還原特定 **KubeVirt VM**

您可以使用 `includedVirtualMachines`還原 CR，僅從備份、快照或複製快照存檔中還原選定的 KubeVirt 虛擬機器。

當 includedVirtualMachines`在還原 CR 上指定時，Trident Protect 僅還原列出的 VirtualMachines 及其相依資源，例如 `VirtualMachineInstance、 DataVolume、 PersistentVolumeClaim、 PersistentVolume、 Secret、 ConfigMap、 ServiceAccount，以及其他 VM 相關物件。

如果您未在復原 CR 中指定 includedVirtualMachines，Trident Protect 將根據復原 CR 設定還原完整的歸檔。

支援的還原 CR 和 CLI 命令

下表列出了支援的還原 CR `includedVirtualMachines`及其對應的 CLI 命令。

恢復 CR	CLI 指令	說明
BackupRestore	tridentctl-protect create backuprestore	從備份還原到新的或不同的命名空間映射。支援 namespaceMapping`和 `storageClassMapping。
BackupInplaceRestore	tridentctl-protect create backupinplacerestore	從備份還原到原始命名空間或多個命名空間。
SnapshotRestore	tridentctl-protect create snapshotrestore	從快照還原到新的或不同的命名空間映射。支援 namespaceMapping`和 `storageClassMapping。
SnapshotInplaceRestore	tridentctl-protect create snapshotinplacerestore	從快照還原到原始命名空間或多個命名空間。
ReplicateSnapshotRestore	僅支援 YAML 格式，或如果您的環境中支援，也可以使用受支援的 CLI 命令。	從複製的快照還原。支援 namespaceMapping、 storageClassMapping，以及使用 `inPlaceRestore`旗標進行原地還原。

> 註 tridentctl-protect CLI 在還原命令上沒有適用於 includedVirtualMachines 的旗標。若要還原特定的 VMs，您必須直接使用 kubectl apply 套用還原 CR YAML 檔案。您可以從頭撰寫 YAML，或是搭配 --dry-run --output yaml 使用 CLI 以產生初始檔案，在 spec 下方新增 includedVirtualMachines 欄位，然後套用它。

虛擬機器範圍的復原行為

The includedVirtualMachines 恢復 CR 中的欄位使用與應用程式 CR 相同的選擇器格式：

```
includedVirtualMachines:
  - namespace: <source_vm_namespace>
    names:
      - <vm_name_1>
      - <vm_name_2>
```

對於還原 CR， `includedVirtualMachines`中的命名空間值指的是備份、快照或複製快照封存中存在的來源命名空間。

如果使用 `namespaceMapping`，Trident Protect 會先從來源歸檔中選擇要求的虛擬機，然後在還原過程中套用命名空間對應。如果使用 `storageClassMapping`，Trident Protect 會在還原轉換階段套用儲存類別對應。

使用 CLI 產生還原 YAML 檔案

您可以使用 `tridentctl-protect`來產生基本復原 CR，然後編輯 YAML 以新增 `includedVirtualMachines`。

```
tridentctl-protect create <restore_command> <restore_name> \
  <restore_options> \
  --dry-run --output yaml > <restore_name>.yaml
```

編輯產生的 YAML ，並在 `spec` 下方新增 `includedVirtualMachines` 欄位，然後套用該檔案：

```
kubectl apply -f <restore_name>.yaml
```

範例：定義一個基於虛擬機器的應用程式

以下應用程式保護跨兩個命名空間的三個特定虛擬機器。

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: production-vms
  namespace: prod
spec:
  includedVirtualMachines:
    - namespace: prod
      names:
        - app-server
        - database-server
    - namespace: monitoring
      names:
        - prometheus-vm
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

範例：建立備份

為基於虛擬機器的應用程式建立備份時，請使用標準備份 CR。備份 CR 中不需要任何虛擬機器特定的欄位。

```yaml
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  name: daily-backup
  namespace: prod
spec:
  applicationRef: production-vms
  appVaultRef: s3-vault
```

範例：從備份還原單一虛擬機

以下範例僅從備份存檔中還原 `database-server`。存檔中的其他虛擬機器（例如 `app-server` 和 `prometheus-vm`）將被跳過。

```yaml
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: restore-db-only
  namespace: prod-vms-restore
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  namespaceMapping:
    - source: prod
      destination: prod-dr
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f restore-db-only.yaml
```

僅 `database-server`及其相依資源（例如 DataVolumes、PVCs、PVs、Secrets、ConfigMaps及相關 KubeVirt 資源）會還原。

範例：從備份中就地還原單一虛擬機

以下範例僅將 `database-server`從備份存檔還原到其原始命名空間。

```yaml
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: ipr-db-only
```

```
  namespace: prod
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f ipr-db-only.yaml
```

範例：從快照還原單一虛擬機

以下範例僅從快照存檔還原 `ubuntu-vm-blue-a627be38`。

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: blue-vm-snap-restore
  namespace: blue
spec:
  appVaultRef: my-appvault
  appArchivePath: snapshots/protectctl-blue-vm/snap-1
  includedVirtualMachines:
    - namespace: blue
      names:
        - ubuntu-vm-blue-a627be38
```

```
kubectl apply -f blue-vm-snap-restore.yaml
```

將虛擬機器還原與其他過濾器結合使用

您可以在同一個還原 CR 中同時使用 `includedVirtualMachines`、`resourceFilter`、`namespaceMapping`和 `storageClassMapping`來微調要還原的內容。

新增 `resourceFilter`時，Trident Protect 首先會選擇 `includedVirtualMachines`中列出的 VM 及其相依資源。然後套用資源篩選器，進一步縮小這些結果中包含或排除的項目範圍。

```
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace>
      names:
        - <vm_name>
```

```
  resourceFilter:
    resourceSelectionCriteria: Include
    resourceMatchers:
      - kinds: ["PersistentVolumeClaim"]
        names: ["<pvc_name>"]
```

對於 `BackupRestore`、 `SnapshotRestore` 和 `ReplicateSnapshotRestore`， `namespaceMapping` 和 `storageClassMapping` 協同工作 `includedVirtualMachines`。 `includedVirtualMachines` 中的命名空間參考對應於歸檔中出現的來源命名空間。命名空間和儲存類別映射在過濾完成後，於復原轉換階段套用。

有關還原行為的詳細資訊，請參閱 "使用Trident Protect 恢復應用程式"。

使用 **CLI** 建立虛擬機器範圍的應用程式

使用 `--virtual-machines`標誌可將應用程式的範圍限定於特定的 KubeVirt VM，而不是整個命名空間。 `--virtual-machines`標誌與 `--namespaces`互斥。

步驟

1. 運行以下命令之一以建立應用程式，並將括號中的值替換為您的環境資訊。 `--virtual-machines` 標誌的格式為 `<namespace>(<vm_name>)`。對於多個 VM，請在括號內使用逗號分隔的名稱：

   - 命名空間中的單一虛擬機器：

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace>(<vm_name>)"
     ```

   - 跨命名空間的多台虛擬機器：

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace_1>(<vm_name_1>,<vm_name_2>
     ),<vm_namespace_2>(<vm_name_3>)"
     ```

相關資訊

- "定義管理應用程式"
- "保護應用程式"
- "使用Trident Protect 恢復應用程式"
- "使用進階還原設定"
- "使用NetApp SnapMirror和Trident Protect 複製應用程式"
- "安裝Trident Protect CLI 插件"
- "使用 AppVault 物件來管理貯體"

管理Trident Protect 執行鉤子

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 支援以下幾種執行鉤子類型，取決於它們的運行時機：

- 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. 如果適用，則會發生檔案系統凍結。"了解更多關於使用Trident Protect 設定檔系統凍結的信息"。
3. 執行資料保護作業。
4. 凍結的檔案系統會在適用的情況下解除凍結。
5. 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

註	上述順序範例僅適用於執行不使用現有快照的備份時。
註	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'kubecl 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。

註 如果快照前執行攔截器新增，變更或移除 Kubernetes 資源，則這些變更會包含在快照或備份中，以及任何後續還原作業中。

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 要求執行鉤子使用的腳本以可執行 shell 腳本的格式編寫。
- 指令碼大小上限為96KB。
- Trident Protect 使用執行鉤子設定和任何符合條件來決定哪些鉤子適用於快照、備份或還原作業。

註 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。

新增到執行掛鉤的每個過濾器都使用正規表示式來匹配叢集中的容器。當鉤子與容器匹配時，鉤子將在該容器上運行其關聯的腳本。過濾器的正規表示式使用正規表示式 2 (RE2) 語法，該語法不支援建立從符合清單中排除容器的過濾器。有關Trident Protect 在執行鉤子過濾器中支援的正規表示式語法的詳細信息，請參閱 "規則運算式2（RE2）語法支援"。

註 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。

註 Verda 儲存庫中的腳本按原樣提供，並未獲得 NetApp 的官方支援。NetApp 支援僅涵蓋 Trident Protect 中的執行掛鉤框架。

建立執行掛鉤

您可以使用Trident Protect 為應用程式建立自訂執行鉤子。您需要擁有所有者、管理員或成員權限才能建立執行鉤子。

使用 **CR**

步驟

1. 建立自訂資源（ CR ）檔案並命名為 `trident-protect-hook.yaml`。
2. 設定以下屬性以符合您的Trident Protect 環境和叢集設定：
   - * 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.mmatchingCriteria ：（ *Optional* ）選擇性的條件金鑰值配對清單，每個配對組成執行掛機篩選器。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

步驟

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 exechook <my_exec_hook_name> --action
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>
--source-file <script-file> -n <application_namespace>
```

手動執行掛鉤

您可以手動執行掛鉤以進行測試，或是在故障後需要手動重新執行掛鉤。您需要擁有擁有者，管理員或成員權限，才能手動執行掛鉤。

手動執行掛鉤包含兩個基本步驟：

1. 建立資源備份，收集資源並建立資源備份，以判斷攔截的執行位置
2. 在備份上執行執行掛鉤

▼ 步驟 **1** ：建立資源備份

使用 **CR**

步驟

1. 建立自訂資源（ CR ）檔案並命名為 `trident-protect-resource-backup.yaml` 。
2. 設定以下屬性以符合您的Trident Protect 環境和叢集設定：
   - * 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 _ ）要建立資源備份的應用程式 Kubernetes 名稱。
   - **spec.appVaultRef** ：（ _required _ ）儲存備份內容的 AppVault 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}'
     ```

     YAML 範例：

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: ResourceBackup
     metadata:
       name: example-resource-backup
     spec:
       applicationRef: my-app-name
       appVaultRef: my-appvault-name
       appArchivePath: example-resource-backup
     ```

3. 在您以正確的值填入 CR 檔案之後，請套用 CR ：

   ```
   kubectl apply -f trident-protect-resource-backup.yaml
   ```

使用CLI

步驟

1. 建立備份，以您環境的資訊取代括號中的值。例如：

```
tridentctl protect create resourcebackup <my_backup_name> --app
<my_app_name> --appvault <my_appvault_name> -n <my_app_namespace>
--app-archive-path <app_archive_path>
```

2. 檢視備份狀態。您可以重複使用此範例命令，直到作業完成為止：

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. 確認備份成功：

```
kubectl describe resourcebackup <my_backup_name>
```

▼ 步驟 **2** ：執行掛鉤

使用 **CR**

步驟

1. 建立自訂資源（ CR ）檔案並命名為 `trident-protect-hook-run.yaml`。
2. 設定以下屬性以符合您的Trident Protect 環境和叢集設定：
   - * 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 _ ）請確保此值符合您在步驟 1 中建立的 ResourceBackup CR 應用程式名稱。
   - **spec.appVaultRef** ：（ _required _ ）請確保此值符合您在步驟 1 中建立的 ResourceBackup CR 的 apVaultRef 。
   - **spec.appArchivePath** ：確保此值與您在步驟 1 中建立的 ResourceBackup CR 中的 appArchivePath 相符。

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **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.Stage * ：（ _required _ ）一個字串，指出執行掛鉤應在動作期間執行的階段。此掛鉤掃描不會在任何其他階段執行掛鉤。可能值：

- 準備
- 貼文

YAML 範例：

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ExecHooksRun
metadata:
  name: example-hook-run
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
  stage: Post
  action: Failover
```

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-run.yaml
```

使用CLI

步驟

1. 建立手動執行攔截執行要求：

```
tridentctl protect create exechooksrun <my_exec_hook_run_name> -n
<my_app_namespace> --action snapshot --stage <pre_or_post> --app
<my_app_name> --appvault <my_appvault_name> --path
<my_backup_name>
```

2. 檢查執行攔截執行的狀態。您可以重複執行此命令，直到作業完成為止：

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. 說明 exechooksrun 物件以查看最終詳細資料和狀態：

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

解除安裝Trident Protect

如果您要從試用版升級到完整版產品，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. 刪除Trident Protect CR 檔案：

   註 | 25.06 及更高版本不需要此步驟。

   ```
   helm uninstall -n trident-protect trident-protect-crds
   ```

2. 移除Trident保護：

   ```
   helm uninstall -n trident-protect trident-protect
   ```

3. 移除Trident Protect 命名空間：

   ```
   kubectl delete ns trident-protect
   ```

Trident和Trident Protect 博客

您可以在這裡找到一些很棒的NetApp Trident和Trident Protect 部落格：

Trident 部落格

- 2025年10月16日："面向 Kubernetes 的高階儲存解決方案"
- 2025年8月19日："增強資料一致性：使用Trident在 OpenShift 虛擬化中使用磁碟區組快照"
- 2025年5月9日： "使用 Amazon EKS 外掛程式自動為 FSx for ONTAP 設定 Trident 後端"
- 2025年4月15日： "NetApp Trident 與 Google Cloud NetApp Volumes 的 SMB 協議"
- 2025年4月14日： "利用Trident 25.02 的光纖通道協定實現 Kubernetes 上的持久存儲"
- 2025年4月14日： "釋放 NetApp ASA r2 系統對 Kubernetes 區塊儲存的強大功能"
- 2025年3月31日： "使用新的認證操作員簡化 Red Hat OpenShift 上的Trident安裝"
- 2025年3月27日： "使用Google Cloud NetApp Volumes為 SMB 設定Trident"
- 2025年3月5日： "解除鎖定無縫 iSCSI 儲存整合： ROSA 叢集 for AWS 上的 FSxN 指南"
- 2025年2月27日： "使用Trident、GKE 和Google Cloud NetApp Volumes部署雲端身份"
- 2024年12月12日： "Trident 推出光纖通道支援"
- 2024年11月11日： "NetApp Trident與Google Cloud NetApp Volumes"
- 2024年10月29日： "使用Trident在 AWS 上將Amazon FSx for NetApp ONTAP與 Red Hat OpenShift 服務 (ROSA) 結合使用"
- 2024年10月29日： "使用 ROSA 上的 OpenShift 虛擬化和Amazon FSx for NetApp ONTAP即時遷移虛擬機"
- 2024年7月8日： "使用 NVMe/TCP 在 Amazon EKS 上為現代容器化應用程式使用ONTAP存儲"
- 2024年7月1日： "使用Google Cloud NetApp Volumes Flex 和Astra Trident實現 Kubernetes 無縫存儲"
- 2024年6月11日： "ONTAP作為 OpenShift 中整合映像註冊表的後端存儲"

Trident Protect博客

- 2025年5月16日："利用Trident Protect 的恢復後鉤子實現登錄機碼故障轉移的自動化，以進行災難復原"
- 2025年5月16日： "使用NetApp Trident Protect 進行 OpenShift 虛擬化災難復原"
- 2025年5月13日："使用Trident Protect 備份和還原進行儲存類別遷移"
- 2025年5月9日："使用Trident Protect 恢復後鉤子重新擴展 Kubernetes 應用程式"
- 2025年4月3日： "Trident Protect 升級：Kubernetes 複製保護與災難復原"
- 2025 年 3 月 13 日："適用於 OpenShift 虛擬化 VM 的毀損一致備份與還原作業"
- 2025 年 3 月 11 日："使用 NetApp Trident 將 GitOps 模式延伸至應用程式資料保護"
- 2025 年 3 月 3 日："Trident 25.02 ：以令人興奮的新功能提升 Red Hat OpenShift 體驗"
- 2025年1月15日："隆重介紹Trident Protect 基於角色的存取控制"

- 2024年11月11日："Kubernetes驅動的資料管理： Trident Protect開啟新時代"

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？

是的，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 安裝程式會建立一個 Kubernetes ClusterRole，該實例擁有對 Kubernetes 叢集的 PersistentVolume 、PersistentVolumeClaim、StorageClass 和 Secret 資源的特定存取權。請參閱 "自訂tridentctl安裝"。

我可以在本機產生 **Trident** 用於安裝的確切資訊清單檔案嗎？

如有需要，您可以在本機產生和修改 Trident 安裝時使用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`。這是為了確保兩個執行個體不使用相同的 FlexVol volume 。

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 。

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Kubernetet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** 後端定義檔案中同時定義管理和 **DataLIFs** ？

管理LIF為必填項目。DataLIF 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 iSCI 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 ： NetApp 建議指定 `dataLIF`。如果未提供， Trident 會從 SVM 擷取 dataLIFs 。您可以指定完整網域名稱（ FQDN ），以用於 NFS 裝載作業，讓您建立循環 DNS ，以便在多個 dataLIFs 之間進行負載平衡。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。

管理與 **DataLIFs** 是否可以使用 **IPv6** 位址？

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 **DataLIF** ？

您只能在和 `ontap-nas-economy`上更新 `DataLIF` `ontap-nas` 。

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"scoragePools"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 一 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-economy`驅動程式在同一個 `FlexVol volume` 中最多建立 `200` 個 `qtree`（可設定在 `50` 到 `300` 之間），每個叢集節點建立 `100,000` 個 `qtree` ，每個叢集建立 `2.4M`。當您輸入由經濟駕駛人服務的新 `PersistentVolumeClaim`項目時，駕駛會查看是否已有 `FlexVol volume` 可為新的 `Qtree` 提供服務。如果 `FlexVol volume` 不存在，無法為 `Qtree` 提供服務，則會建立新的 `FlexVol volume` 。

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。您可以擴充由 Trident 建立的 NFS PVC。Trident Autogrow 原則可以根據容量臨界值自動擴充 PVC 。ONTAP Volume 自動調整規模是分隔的，並適用於支援的 ONTAP SAN 驅動程式：它會擴充後端 FlexVol 而不變更 PVC 大小。如需詳細資訊，請參閱 "ONTAP Volume 自動調整規模"。

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？

截至今日，我們的產品已提供按需快照支援。 `ontap-nas` 、 `ontap-nas-flexgroup` 、 `ontap-san` 、 `ontap-san-economy` 、 `solidfire-san` 、和 `azure-netapp-files`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 volume 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。

> 註 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 支援部門。

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「kubechtl -n trident描述部署Trident」和「kubectl -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 triident」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「kubecll Get pod - n triident」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
```

- 您也可以在後端定義中加入、以取得每個後端的偵錯記錄 `debugTraceFlags`。例如、在 Trident 記錄檔中包括 `debugTraceFlags: {"api":true, "method":true,}`以取得 `API` 呼叫和方法傳輸。現有的後端可以 `debugTraceFlags`使用設定 `tridentctl backend update`。
- 使用 Red Hat Enterprise Linux CoreOS （ RHCOS ）時，請確保 `iscsid`已在工作節點上啟用，並依預設啟動。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 搭配 "Azure NetApp Files"時，您可能會遇到的常見問題是租用戶金鑰和客戶端金鑰來自權限不足的應用程式註冊。有關 Trident 要求的完整清單，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「stystemctl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
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

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。

警告	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。**

```
 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。

當預期啟用“v4.2-xattrs”時，NFSv4.2 用戶端在升級ONTAP後報告“無效參數”

升級ONTAP後，NFSv4.2 用戶端在嘗試掛載 NFSv4.2 匯出時可能會報告「無效參數」錯誤。當 `v4.2-xattrs` SVM 上未啟用該選項。.解決方法啟用 `v4.2-xattrs` 選項或升級至ONTAP 9.12.1 或更高版本，預設此選項為啟用。

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）文章和不和管道。

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	全面支援結束	有限支援結束	自助支援終止
"26.10"	2027 年 10 月	2028年10月	2029 年 10 月
"26.06"	2027 年 7 月	2028 年 7 月	2029 年 7 月
"26.02"	2027 年 2 月	2028 年 2 月	2029 年 2 月
"25.10"	2026 年 10 月	2027 年 10 月	2028年10月
"25.06"	2026 年 6 月	2027 年 6 月	2028 年 6 月
"25.02"	2026 年 2 月	2027 年 2 月	2028 年 2 月
"24.10"	—	2026 年 10 月	2027 年 10 月
"24.06"	—	2026 年 6 月	2027 年 6 月

"24.02"	—	2026 年 2 月	2027 年 2 月
"23.10"	—	—	2026 年 10 月
"23.07"	—	—	2026 年 7 月
"23.04"	—	—	2026 年 4 月

自我支援

如需完整的疑難排解文章清單，請參閱 "NetApp知識庫（需要登入）"。

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。

總覽

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

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

控制器節點的連接埠

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

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

工作節點的連接埠

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

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

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

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

何時使用REST API

REST API 適用於在非 Kubernetes 部署中使用 Trident 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。

> 註 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。

警告	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「PeristentVolume Claim」、要求系統管理員先前設定的Kubernetes「torageClass」中的特定大小的新「PersistentVolume」。
2. Kubernetes「torageClass」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「PeristentVolume」、告訴Kubernetes如何尋找、掛載及處理Volume、以及Trident中保留「PeristentVolume」與實際儲存設備之間關係的Volume。
5. Kubernetes將「PeristentVolume Claim」連結到新的「PeristentVolume」。在執行的任何主機上、包含PeristentVolume的「PeristentVolume 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「PeristentVolume Claim」。
9. Trident會識別所需的快照、並執行建立「PeristentVolume」和「Volume」所需的相同步驟。

提示	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「PeristentVolume 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、ontap-san-economy
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
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
trident.netapp.io/skipRecoveryQueue	跳過恢復隊列	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy

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：

- NetApp 建議複製閒置磁碟區。
- 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。

> 註 Trident會建立「PeristentVolume」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「TorageClass」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「torageClass」物件是以名稱在「PeristentVolume 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「torageClass」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[stric]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
後端類型	字串	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、solidfire-san、azure-netapp-files、ontap-san-economy	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
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

屬性	類型	價值	優惠	申請	支援者
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、ONTAP-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「addionalStoragePools」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「scoragePools」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、「addionalStoragePools」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

屬性	類型	價值	說明	相關驅動因素	**Kubernetes**版本
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-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy、solidfire-san、azure-netapp-files	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

提示

- 。 `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會有
  - 「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「Cig-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。

> 註 VolumeSnapshot 的生命週期與來源 PVC 無關：即使來源 PVC 被刪除，快照仍然存在。刪除具有相關快照的永久虛擬磁碟時、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「PeristentVolume Claim」和「PeristentVolume」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。

> 註 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 `VolumeGroupSnapshotClass` 物件

Kubernetes `VolumeGroupSnapshotClass` 物件類似於 `VolumeSnapshotClass`。它們有助於定義多種儲存類別，並被磁碟區組快照引用，以將快照與所需的快照類別關聯。每個磁碟區組快照都與單一磁碟區組快照類別相關聯。

一個 `VolumeGroupSnapshotClass`應由管理員定義，以便建立快照群組。卷冊組快照類別使用以下定義建立：

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
```

```
deletionPolicy: Delete
```

由 Trident 處理。 deletionPolicy`指定必須刪除群組快照時要採取的動作。當 `deletionPolicy`設定為 `Delete ，刪除快照時，磁碟區組快照物件以及儲存叢集上的底層快照也將被刪除。或者、將其設定為`Retain`表示 `VolumeGroupSnapshotContent`保留實體快照。

Kubernetes `VolumeGroupSnapshot` 物件

Kubernetes `VolumeGroupSnapshot`物件是建立多個磁碟區快照的請求。正如 PVC 代表使用者對磁碟區的請求一樣，磁碟區組快照是使用者為現有 PVC 建立快照的請求。

當磁碟區組快照請求到達時，Trident 會自動管理後端磁碟區的群組快照的創建，並透過建立唯一的`VolumeGroupSnapshotContent`目的。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。

> 註 VolumeGroupSnapshot 的生命週期與來源 PVC 無關：即使來源 PVC 被刪除，快照仍然有效。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。當所有關聯的快照都被刪除時，磁碟區組快照也會被移除。

Kubernetes `VolumeGroupSnapshotContent` 物件

Kubernetes VolumeGroupSnapshotContent`物件表示從已配置的磁碟區中取得的群組快照。它類似於`PersistentVolume、表示儲存叢集上的已佈建快照。與和 PersistentVolume`物件類似`PersistentVolumeClaim、建立快照時、 VolumeSnapshotContent`物件會維護對物件的一對一對應`VolumeSnapshot、而物件已要求建立快照。

這 `VolumeGroupSnapshotContent`物件包含識別快照群組的詳細信息，例如 `volumeGroupSnapshotHandle`以及儲存系統上現有的各個volumeSnapshotHandles。

當快照請求到達時，Trident 會在後端建立磁碟區組快照。建立卷宗組快照後，Trident 會配置一個`VolumeGroupSnapshotContent`對象，從而將快照公開給 Kubernetes API。

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「kubecl 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。

> 註 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。

> 註 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`

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

與此處的其他物件不同、儲存資源池候選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，包括 NFS 共用， iSCSI 和 FC LUN 等後端端點。在 Kubernetes 中、這些直接對應到 `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。

> 註
> - 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」）

屬性	類型	必要	說明
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
跳過恢復隊列	字串	否	刪除磁碟區時，繞過儲存中的復原佇列，立即刪除磁碟區。

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「PeristentVolume 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「napshot」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

註 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。

註	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"。
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：「CapPrm：0000003fffffffff'、「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。

權限	說明
SELinux	在 OpenShift 上、權限容器會在（「超級貴賓 Container 」）網域中執行 `spc_t`、而非權限容器則會在網域中執行 `container_t`。在上 `containerd`、安裝後 `container-selinux`、所有容器都會在網域中執行 `spc_t`、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>.」

警告 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、而且會將所有可能的功能授予權限容器。	空白
《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」

欄位	說明	預設
《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」
《requireddropCapabilities》	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》、《empityDir.》

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-statd」。	"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"

標籤	說明	預設
《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、而且會將所有可能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、empyDir'》

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/"

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/"

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/"

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 © 2026 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。