Trident 25.06 文檔

Trident

NetApp
January 15, 2026

目錄

Trident 25.06 文檔

發行說明

什麼是新的

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

這 `tridentctl`安裝程式 zip 檔案中提供的 Linux 二進位檔案是經過測試和支援的版本。請注意，`macos`提供的二進位文件 `/extras`壓縮文件中的部分內容未經測試或支援。

25.06.2 中的新增功能

「新增功能」摘要提供了有關Trident和Trident Protect 版本增強功能、修復程序和棄用項目的詳細資訊。

Trident

修復

- **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

對於使用SolidFire的客戶，請不要升級到 25.06.1，因為取消發布捲時有已知問題。25.06.2 即將發布以解決此問題。

修復

- **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（"第1008期"）。

25.06 中的變化

Trident

增強功能

- **Kubernetes**：
  - 新增對 CSI 磁碟區組快照的支持 `v1beta1`適用於ONTAP的磁碟區組快照 Kubernetes API - SAN iSCSI 驅動程式。看"使用磁碟區組快照"。

VolumeGroupSnapshot 是 Kubernetes 中的測試版功能，其 API 也處於測試版階段。VolumeGroupSnapshot 所需的最低 Kubernetes 版本為 1.32。

- 除了 iSCSI 之外，還增加了對ONTAP ASA r2 的 NVMe/TCP 支援。看link:"ONTAP SAN 配置選項和範例"。
- 為ONTAP-NAS 和ONTAP-NAS-Economy 卷添加了安全的 SMB 支援。現在可以將 Active Directory 使用者和群組與 SMB 磁碟區一起使用，以增強安全性。看"啟用安全的 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，允許執行 discard/TRIM 命令以回收空間。
- 提高格式化 LUKS 加密磁碟區時的效能。
- 增強了對失敗但部分格式化的 LUKS 設備的 LUKS 清理功能。
- 增強了Trident節點對NVMe卷掛載和分離的冪等性。
- 額外 `internalID`欄位到ONTAP -SAN-Economy 驅動程式的Trident磁碟區配置。
- 為 NVMe 後端添加了對SnapMirror磁碟區複製的支援。看"使用SnapMirror複製卷"。

實驗性增強

不可用於生產環境。

- [技術預覽] 透過以下方式啟用並發的Trident控制器操作 `--enable-concurrency`功能標誌。這樣一來，控制器操作就可以並行運行，從而提高繁忙或大型環境的效能。

  此功能為實驗性功能，目前僅支援使用ONTAP-SAN 驅動程式（iSCSI 和 FCP 協定）的有限平行工作流程。

- 【技術預覽】為 ANF 驅動程式新增了手動 QOS 支援。

修復

- **Kubernetes**：
  - 修正了 CSI NodeExpandVolume 的一個問題，即當底層 SCSI 磁碟不可用時，多路徑設備的大小可能會不一致。
  - 修正了ONTAP-NAS 和ONTAP-NAS-Economy 驅動程式的重複匯出策略清理失敗的問題。
  - 修正了 GCNV 磁碟區預設使用 NFSv3 的問題 `nfsMountOptions`未設定；現在同時支援 NFSv3 和 NFSv4 協定。如果 `nfsMountOptions`如果未提供，則將使用主機的預設 NFS 版本（NFSv3 或 NFSv4）。
  - 修正了使用 Kustomize 安裝Trident時出現的部署問題（"第831期"）。
  - 修正了從快照建立的 PVC 缺少匯出策略的問題（"第1016期"）。
  - 修正了 ANF 磁碟區大小無法自動按 1 GiB 增量對齊的問題。
  - 修正了使用 Bottlerocket 時 NFSv3 出現的問題。
- 修正了使用SolidFire後端克隆卷時出現的逾時問題（"第1008期"）。
- 修正了ONTAP-NAS-Economy 磁碟區在調整大小失敗的情況下仍可擴充至 300 TB 的問題。

- 修正了使用ONTAP REST API 時克隆分割操作同步執行的問題。

棄用：

- **Kubernetes**：已將最低支援的 Kubernetes 版本更新至 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

- 提高了復原速度，並提供了更頻繁執行完整備份的選項。
- 改進了應用程式定義的粒度，並可透過群組版本類型 (GVK) 篩選進行選擇性復原。
- 使用 AppMirrorRelationship (AMR) 和NetApp SnapMirror時，可實現高效的重新同步和反向複製，從而避免完全 PVC 複製。
- 新增了使用 EKS Pod Identity 建立 AppVault 儲存桶的功能，無需為 EKS 叢集指定儲存桶憑證的金鑰。
- 增加了在恢復命名空間時跳過恢復標籤和註釋的功能（如果需要）。
- AppMirrorRelationship (AMR) 現在將檢查來源 PVC 是否擴展，並根據需要對目標 PVC 執行相應的擴展。

修復

- 修正了先前快照的快照註解值被套用到新快照的錯誤。所有快照註解現已正確應用。
- 預設情況下，為資料移動器加密（Kopia / Restic）定義了一個金鑰，如果未定義則不會定義。
- 為 S3 應用程式庫建立新增了改進的驗證和錯誤訊息。
- AppMirrorRelationship (AMR) 現在只複製處於綁定狀態的 PV，以避免複製失敗。
- 修正了在具有大量備份的 AppVault 上取得 AppVaultContent 時顯示錯誤的問題。
- 為避免故障，KubeVirt VMSnapshots 被排除在恢復和故障轉移操作之外。
- 修正了 Kopia 的一個問題，即由於 Kopia 的預設保留計劃覆蓋了使用者在計劃中設定的內容，因此導致快照過早刪除。

25.02.1 中的變更

Trident

修復

- **Kubernetes**：
  - 修正了 trident-operator 在使用非預設鏡像倉庫時 sidecar 鏡像名稱和版本填充不正確的問題 ("第983期" ）。
  - 修正了ONTAP故障轉移復原期間多路徑會話無法復原的問題（"第961期" ）。

25.02 中的變化

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**：
  - 增加了對ONTAP ASA r2 的 iSCSI 支援。
  - 增加了在非正常節點關閉情況下強制分離ONTAP-NAS 卷的支援。新的ONTAP-NAS 磁碟區現在將使用Trident管理的按磁碟區匯出策略。為現有磁碟區提供升級路徑，使其在取消發佈時過渡到新的匯出策略模型，而不會影響正在運行的工作負載。
  - 新增了 cloneFromSnapshot 註解。
  - 增加了對跨命名空間卷克隆的支援。
  - 增強 iSCSI 自癒掃描修復功能，可透過精確的主機、通道、目標和 LUN ID 啟動重新掃描。
  - 增加了對 Kubernetes 1.32 的支援。
- **OpenShift**：
  - 為 ROSA 叢集上的 RHCOS 新增了自動 iSCSI 節點準備支援。
  - 為ONTAP驅動程式新增了對 OpenShift 虛擬化的支援。
- ONTAP-SAN驅動程式新增了光纖通道支援。
- 新增NVMe LUKS支援。
- 所有基礎影像均已切換為臨時影像。
- 新增了 iSCSI 連線狀態發現和記錄功能，用於記錄 iSCSI 會話應該登入但實際未登入的情況（"第961期"）。
- 新增了對使用 google-cloud-netapp-volumes 驅動程式的 SMB 磁碟區的支援。
- 增加了對ONTAP磁碟區在刪除時跳過復原佇列的支援。
- 新增了使用 SHA 值而非標籤覆寫預設影像的功能。
- 為 tridentctl 安裝程式新增了 image-pull-secrets 標誌。

修復

- **Kubernetes**：
  - 修正了自動匯出策略中缺少的節點 IP 位址（"第965期"）。
  - 修正了ONTAP-NAS-Economy 的自動匯出策略過早切換到按磁碟區策略的問題。
  - 修復後端配置憑證，以支援所有可用的 AWS ARN 分割區（"第913期"）。
  - 在Trident運算子中新增了停用自動配置器協調的選項（"第924期"）。
  - 為 csi-resizer 容器新增了 securityContext（"第976期"）。

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。
- 為 Kubevirt 環境增加了在應用程式層級控製凍結行為的功能。
- 增加了對配置AutoSupport代理連線的支援。
- 增加了為資料移動加密（Kopia / Restic）定義金鑰的功能。
- 增加了手動運行執行鉤子的功能。
- 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。
- 為 AppVault 物件新增了對 HTTP / HTTPS 出口代理的支援。
- 擴展資源過濾器，以排除叢集範圍的資源。
- 為 S3 AppVault 憑證新增了對 AWS 會話令牌的支援。
- 增加了對快照執行前鉤子之後資源收集的支援。

修復

- 改進了臨時磁碟區的管理，使其跳過ONTAP磁碟區復原佇列。
- SCC 註解現已恢復為原始值。
- 支援並行操作，提高了恢復效率。
- 增強了對大型應用程式執行鉤子超時的支援。

24.10.1 中的變更

增強功能

- **Kubernetes**：新增對 Kubernetes 1.32 的支援。
- 新增了 iSCSI 連線狀態發現和記錄功能，用於記錄 iSCSI 會話應該登入但實際未登入的情況（"第961期"）。

修復

- 修正了自動匯出策略中缺少的節點 IP 位址（"第965期"）。
- 修正了ONTAP-NAS-Economy 的自動匯出策略過早切換到按磁碟區策略的問題。
- 更新了Trident和 Trident-ASUP 依賴項，以解決 CVE-2024-45337 和 CVE-2024-45310。
- 在 iSCSI 自癒期間，移除間歇性不健康的非 CHAP 入口網站的註銷操作 ("第961期")。

24.10 中的變化

增強功能

- Google Cloud NetApp Volumes驅動程式現已正式推出，適用於 NFS 卷，並支援區域感知配置。
- GCP Workload Identity 將用作Google Cloud NetApp Volumes與 GKE 的雲端身分。
- 額外 `formatOptions`為ONTAP-SAN 和ONTAP-SAN-Economy 驅動程式新增配置參數，允許使用者指定 LUN 格式選項。
- 將Azure NetApp Files區的最小大小減少到 50 GiB。 Azure 新的最小系統規模預計將於 11 月正式推出。
- 額外 `denyNewVolumePools`配置參數，用於將ONTAP-NAS-Economy 和ONTAP-SAN-Economy 驅動程式限制為預先存在的 Flexvol 池。
- 增加了對所有ONTAP驅動程式中 SVM 聚合的新增、刪除或重新命名的偵測。
- 在 LUKS LUN 中添加了 18 MiB 開銷，以確保報告的 PVC 大小可用。
- 改進了ONTAP-SAN 和ONTAP-SAN-Economy 節點階段和取消階段錯誤處理，允許取消階段操作在階段失敗後移除設備。
- 新增了自訂角色產生器，讓客戶在ONTAP中為Trident建立極簡角色。
- 增加了額外的故障排除日誌記錄 `lsscsi`（"第792期"）。

Kubernetes

- 為 Kubernetes 原生工作流程新增了新的Trident功能：
  - 資料保護
  - 資料遷移
  - 災難復原
  - 應用行動性

    "了解更多關於Trident Protect的信息"。
- 新增標誌 `--k8s-api-qps`指示安裝程式設定Trident用於與 Kubernetes API 伺服器通訊的 QPS 值。
- 額外 `--node-prep`向安裝程式發出訊號，以便自動管理 Kubernetes 叢集節點上的儲存協定相依性。已測試並驗證與 Amazon Linux 2023 iSCSI 儲存協定的兼容性
- 增加了對ONTAP-NAS-Economy 磁碟區在非正常節點關閉場景下強制分離的支援。
- 新的ONTAP-NAS-Economy NFS 磁碟區在使用時將採用基於 qtree 的匯出策略 `autoExportPolicy`後端選項。 Qtree 僅在發佈時才會對應到節點限制性匯出策略，以提高存取控制和安全性。當Trident從所有節點取消發布磁碟區時，現有的 qtree 將切換到新的匯出策略模型，這樣就不會影響正在進行的工作負載。
- 增加了對 Kubernetes 1.31 的支援。

實驗性增強

- ONTAP-SAN驅動程式新增了光纖通道支援的技術預覽。

修復

- **Kubernetes：**
  - 修正了 Rancher 進入 webhook 封鎖Trident Helm 安裝的問題（"第839期"）。
  - 固定 Helm Chart 值中的親和力鍵（"第898期"）。
  - 已修正 tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector 無法處理「true」值（"第899期"）。
  - 克隆過程中建立的已刪除臨時快照（"第901期"）。
- 新增對 Windows Server 2019 的支援。
- 修復了Trident倉庫中的 `go mod tidy`（"第767期"）。

棄用

- **Kubernetes:**
  - 已將支援的最低 Kubernetes 版本更新至 1.25。
  - 已移除對 POD 安全策略的支援。

產品品牌重塑

從 24.10 版本開始， Astra Trident更名為Trident （Netapp Trident）。此次品牌重塑不會影響Trident的任何功能、支援的平台或互通性。

24.06 中的變化

增強功能

- 重要提示： `limitVolumeSize`此參數現在限制ONTAP經濟型驅動程式中的 qtree/LUN 大小。使用新的 `limitVolumePoolSize`用於控制這些驅動程式中 Flexvol 大小的參數。 （"第341期"）。
- 增加了 iSCSI 自癒功能，以便在使用已棄用的 igroup 時透過精確的 LUN ID 啟動 SCSI 掃描 ("第883期"）。
- 增加了對磁碟區克隆和調整大小操作的支持，即使後端處於掛起模式也允許執行這些操作。
- 增加了將Trident控制器的使用者設定日誌設定傳播到Trident節點 pod 的功能。
- Trident增加了對ONTAP 9.15.1 及更高版本預設使用 REST 而不是 ONTAPI (ZAPI) 的支援。
- ONTAP儲存後端新增了對自訂磁碟區名稱和元資料的支持，用於建立新的持久性磁碟區。
- 增強了 `azure-netapp-files`（ANF）驅動程序，當 NFS 掛載選項設定為使用 NFS 版本 4.x 時，預設會自動啟用快照目錄。
- 增加了對 NFS 磁碟區的 Bottlerocket 支援。
- 新增對Google Cloud NetApp Volumes的技術預覽支援。

Kubernetes

- 增加了對 Kubernetes 1.30 的支援。
- 為Trident DaemonSet 新增了在啟動時清理殭屍掛載點和殘留追蹤檔案的功能（"第883期"）。

- 新增了PVC註釋 `trident.netapp.io/luksEncryption`用於動態導入 LUKS 磁碟區（"第849期"）。
- 為 ANF 驅動程式添加了拓撲感知功能。
- 新增對 Windows Server 2022 節點的支援。

修復

- 修正了過期交易導致的Trident安裝失敗問題。
- 修正了 tridentctl 忽略來自 Kubernetes 的警告訊息的問題（"第892期"）。
- 更換了Trident控制器 `SecurityContextConstraint`優先 `0（"第887期"）。
- ONTAP驅動程式現在接受小於 20 MiB 的磁碟區大小（"問題[#885"）。
- 修正了Trident ，以防止在ONTAP -SAN 驅動程式調整大小操作期間FlexVol磁碟區縮小。
- 修正了 NFS v4.1 中 ANF 磁碟區匯入失敗的問題。

24.02 中的變化

增強功能

- 新增對雲端身分的支援。
  - 將使用 AKS 和 ANF - Azure 工作負載標識作為雲端標識。
  - EKS 與 FSxN - 將使用 AWS IAM 角色作為雲端身分。
- 增加了從 EKS 控制台將Trident作為外掛程式安裝到 EKS 叢集的支援。
- 增加了配置和停用 iSCSI 自癒功能 ("第864期"）。
- 為ONTAP驅動程式添加了Amazon FSx特性，以啟用與 AWS IAM 和 SecretsManager 的集成，並使Trident能夠刪除帶有備份的 FSx 磁碟區 ("第453期"）。

Kubernetes

- 新增對 Kubernetes 1.29 的支援。

修復

- 修正了未啟用 ACP 時的 ACP 警告訊息（"第866期"）。
- 在ONTAP驅動程式中，當克隆與快速關聯時，在刪除快照期間執行克隆分割之前，增加了 10 秒的延遲。

棄用

- 從多平台鏡像清單中移除了 in-toto 認證框架。

23.10 中的變化

修復

- 對於 ontap-nas 和 ontap-nas-flexgroup 儲存驅動程序，如果新請求的大小小於總卷大小，則固定卷擴展（"第834期"）。

- 固定磁碟區大小，以便在匯入 ontap-nas 和 ontap-nas-flexgroup 儲存驅動程式時僅顯示磁碟區的可用大小 ("第722期" ）。
- 修正了ONTAP -NAS-Economy 的FlexVol名稱轉換問題。
- 修正了 Windows 節點重新啟動後Trident初始化出現的問題。

增強功能

Kubernetes

增加了對 Kubernetes 1.28 的支援。

Trident

- 增加了對使用 Azure 託管識別 (AMI) 和 azure-netapp-files 儲存驅動程式的支援。
- 為ONTAP-SAN 驅動程式新增了對 NVMe over TCP 的支援。
- 增加了當後端被使用者設定為暫停狀態時暫停磁碟區配置的功能（"第558期" ）。

23.07.1 中的變更

*Kubernetes：*修正了守護程式集刪除問題，以支援零停機升級（"第740期" ）。

23.07 中的變化

修復

Kubernetes

- 修正了Trident升級，使其忽略處於終止狀態的舊 pod（"第740期" ）。
- 為「transient-trident-version-pod」定義添加了容忍度（"第795期" ）。

Trident

- 修正了 ONTAPI (ZAPI) 請求，以確保在取得 LUN 屬性時查詢 LUN 序號，從而在節點暫存作業期間識別和修復幽靈 iSCSI 裝置。
- 修復了儲存驅動程式程式碼中的錯誤處理（"第816期" ）。
- 修正了使用 use-rest=true 的ONTAP驅動程式時配額調整的問題。
- 修復了 ontap-san-economy 中的 LUN 克隆創建問題。
- 恢復發布資訊字段 `rawDevicePath`到 `devicePath`新增了用於填充和恢復（在某些情況下）的邏輯 `devicePath`場地。

增強功能

Kubernetes

- 新增對匯入預配置快照的支援。
- 最小化部署和守護程式集 Linux 權限（"第817期" ）。

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，則更新後端狀態（"第 801 期" ，"#543" ）。
- LUN 序號始終在 ControllerVolumePublish 工作流程中檢索和發布。
- 增加了額外的邏輯來驗證 iSCSI 多路徑設備的序號和大小。
- 對 iSCSI 磁碟區進行額外驗證，以確保正確的多路徑裝置已取消暫存。

實驗增強

為ONTAP-SAN 驅動程式新增了對 NVMe over TCP 的技術預覽支援。

文件

在組織結構和格式方面都進行了許多改進。

棄用

Kubernetes

- 已移除對 v1beta1 快照的支援。
- 移除對 CSI 之前的磁碟區和儲存類別的支援。
- 已將支援的最低 Kubernetes 版本更新至 1.22。

23.04 中的變化

只有當 Kubernetes 版本啟用了非優雅節點關閉功能門時，才支援對ONTAP-SAN-* 磁碟區強制分離。必須在安裝時使用以下方式啟用強制分離： `--enable-force-detach` Trident安裝程式標誌。

修復

- 修正了Trident Operator 在規範中指定時使用 IPv6 localhost 進行安裝的問題。
- 修正了Trident Operator 叢集角色權限，使其與捆綁包權限保持同步（"第799期" ）。
- 修正了在 RWX 模式下將原始區塊卷附加到多個節點的問題。
- 修正了FlexGroup克隆支援和 SMB 磁碟區的磁碟區匯入問題。
- 修正了Trident控制器無法立即關閉的問題（"第811期" ）。
- 新增了修復程序，用於列出與使用 ontap-san-* 驅動程式配置的指定 LUN 關聯的所有 igroup 名稱。
- 新增了允許外部進程運行完成的修復。
- 修正了 s390 架構的編譯錯誤（"第537期" ）。
- 修正了卷宗掛載操作期間日誌等級不正確的問題"第781期"）。
- 修復了潛在的類型斷言錯誤（"第802期" ）。

增強功能

- Kubernetes：
  - 新增對 Kubernetes 1.27 的支援。
  - 新增對導入 LUKS 磁碟區的支援。
  - 增加了對 ReadWriteOncePod PVC 存取模式的支援。
  - 增加了在非正常節點關閉場景下強制分離ONTAP-SAN-* 磁碟區的支援。
  - 所有ONTAP-SAN-* 磁碟區現在都將使用按節點 igroup。只有在 LUN 主動發佈到這些節點時，才會將其對應到 igroup，以提高我們的安全態勢。當Trident認為在不影響活動工作負載的情況下安全地將現有捲切換到新的 igroup 方案時，將會擇機進行切換（"第758期"）。
  - 透過從ONTAP -SAN-* 後端清理未使用的 Trident 管理的 igroup，提高了Trident 的安全性。
- 為 ontap-nas-economy 和 ontap-nas-flexgroup 儲存驅動程式新增了對Amazon FSx SMB 磁碟區的支援。
- 增加了對 ontap-nas、ontap-nas-economy 和 ontap-nas-flexgroup 儲存驅動程式的 SMB 共享的支援。
- 增加了對 arm64 節點的支援（"第732期"）。
- 改進了Trident的關閉流程，先停用API伺服器（"第811期"）。
- 在 Makefile 中新增了對 Windows 和 arm64 主機的跨平台建置支援；請參閱 BUILD.md。

棄用

Kubernetes：設定 ontap-san 和 ontap-san-economy 驅動程式時，將不再建立後端範圍的 igroups（"第758期"）。

23.01.1 中的變更

修復

- 修正了Trident Operator 在規範中指定時使用 IPv6 localhost 進行安裝的問題。
- 修正了Trident Operator 叢集角色權限，使其與捆綁包權限保持同步。"第799期"。
- 新增了允許外部進程運行完成的修復。
- 修正了在 RWX 模式下將原始區塊卷附加到多個節點的問題。
- 修正了FlexGroup克隆支援和 SMB 磁碟區的磁碟區匯入問題。

23.01 中的變化

Trident現已支援 Kubernetes 1.27。請先升級Trident ，再升級 Kubernetes。

修復

- Kubernetes：新增了排除 Pod 安全性原則建立的選項，以修復透過 Helm 安裝Trident 的問題（"第 783 期、第 794 期"）。

增強功能

Kubernetes

- 增加了對 Kubernetes 1.26 的支援。
- 提高了Trident RBAC資源的整體利用率（"第757期"）。
- 增加了自動化功能，用於偵測和修復主機節點上損壞或過期的 iSCSI 會話。
- 增加了對擴展 LUKS 加密卷的支援。
- Kubernetes：為 LUKS 加密磁碟區新增了憑證輪替支援。

Trident

- 為 ontap-nas 儲存驅動程式新增了對Amazon FSx for NetApp ONTAP 的SMB 磁碟區的支援。
- 增加了在使用 SMB 磁碟區時對 NTFS 權限的支援。
- 為具有 CVS 服務等級的 GCP 磁碟區的儲存池新增了支援。
- 在使用 ontap-nas-flexgroup 儲存驅動程式建立 FlexGroups 時，增加了對可選使用 flexgroupAggregateList 的支援。
- 改進了 ontap-nas-economy 儲存驅動程式在管理多個FlexVol磁碟區時的效能
- 已為所有ONTAP NAS 儲存驅動程式啟用 dataLIF 更新。
- 更新了Trident Deployment 和 DaemonSet 的命名約定，以反映主機節點作業系統。

棄用

- Kubernetes：已將支援的最低 Kubernetes 版本更新為 1.21。
- 配置時不應再指定 DataLIF。 `ontap-san`或者 `ontap-san-economy`司機。

22.10 中的變化

*升級到Trident 22.10之前，請務必閱讀以下重要資訊。 *

關於Trident 22.10 的關鍵資訊

- Trident現已支援 Kubernetes 1.25。在升級到 Kubernetes 1.25 之前，必須先將Trident升級到 22.10。
- Trident現在嚴格強制要求在 SAN 環境中使用多路徑配置，建議值為 `find_multipaths: no`在 multipath.conf 檔案中。

  使用非多路徑配置或使用 `find_multipaths: yes`或者 `find_multipaths: smart`multipath.conf 檔案中的值會導致掛載失敗。 Trident建議使用 `find_multipaths: no`自 21.07 版本發布以來。

修復

- 修正了使用ONTAP後端建立的特定問題 `credentials`22.07.0 升級期間欄位無法上線 ("第759期"）。
- **Docker**：修正了導致 Docker 磁碟區外掛程式在某些環境下無法啟動的問題（"第548期"和"第760期"）。
- 修正了ONTAP SAN 後端特有的 SLM 問題，以確保僅發布屬於報告節點的 dataLIF 子集。

- 修正了附加磁碟區時發生不必要的 iSCSI LUN 掃描的效能問題。
- 移除了Trident iSCSI 工作流程中的細微重試，以便快速失敗並減少外部重試間隔。
- 修正了當對應的多路徑裝置已重新整理時，重新整理 iSCSI 裝置會傳回錯誤的問題。

增強功能

- Kubernetes：
  - 新增對 Kubernetes 1.25 的支援。在升級到 Kubernetes 1.25 之前，必須先將Trident升級到 22.10。
  - 為Trident部署和 DaemonSet 新增了單獨的 ServiceAccount、ClusterRole 和 ClusterRoleBinding，以便將來進行權限增強。
  - 增加了對"跨命名空間卷共享"。
- 所有Trident `ontap-*`儲存驅動程式現在可以與ONTAP REST API 配合使用。
- 新增運算子 yaml(`bundle_post_1_25.yaml`) 沒有 `PodSecurityPolicy`支援 Kubernetes 1.25。
- 額外"支援 LUKS 加密卷"為了 `ontap-san`和 `ontap-san-economy`儲存驅動程式。
- 新增對 Windows Server 2019 節點的支援。
- 額外"支援 Windows 節點上的 SMB 卷"透過 `azure-netapp-files`儲存驅動程式。
- ONTAP驅動程式的自動MetroCluster切換偵測功能現已普遍可用。

棄用

- **Kubernetes**： 已將支援的最低 Kubernetes 版本更新為 1.20。
- 已移除Astra資料儲存 (ADS) 驅動程式。
- 已移除對以下功能的支持 `yes`和 `smart`選項 `find_multipaths`配置 iSCSI 工作節點多路徑時。

22.07 中的變化

修復

Kubernetes

- 修正了使用 Helm 或Trident Operator 設定Trident時處理節點選擇器的布林值和數值的問題。 （"GitHub 問題 #700" ）
- 修正了處理非 CHAP 路徑錯誤的問題，以便 kubelet 在失敗時重試。"GitHub 問題 #736" ）

增強功能

- 將 CSI 映像檔的預設鏡像倉庫從 k8s.gcr.io 過渡到 registry.k8s.io
- ONTAP-SAN 磁碟區現在將使用每個節點的 igroup，並且僅在主動發佈到這些節點時才將 LUN 對應到 igroup，以提高我們的安全態勢。當Trident認為在不影響目前工作負載的情況下安全地將現有磁碟區切換到新的 igroup 方案時，將會擇機進行。
- 在Trident安裝中包含 ResourceQuota，以確保在 PriorityClass 消耗預設受到限制時， Trident DaemonSet 能夠被調度。
- 為Azure NetApp Files驅動程式新增了對網路功能的支援。 （"GitHub 問題 #717" ）

- ONTAP驅動程式新增了技術預覽版自動MetroCluster切換偵測功能。 （"GitHub 問題 #228" ）

棄用

- **Kubernetes**： 已將支援的最低 Kubernetes 版本更新為 1.19。
- 後端配置不再允許在單一配置中使用多種身份驗證類型。

搬家

- AWS CVS 驅動程式（自 22.04 版本起已棄用）已移除。
- Kubernetes
  - 從節點 pod 移除不必要的 SYS_ADMIN 功能。
  - 將 nodeprep 簡化為簡單的主機資訊和主動服務發現，以盡力確認 NFS/iSCSI 服務在工作節點上可用。

文件

一個新的"艙體安全標準"（PSS）部分已添加，詳細說明了Trident在安裝時啟用的權限。

22.04 中的變化

NetApp不斷改進和增強其產品和服務。以下是Trident的一些最新功能。有關先前版本，請參閱 "早期版本的文檔"。

> 如果您是從任何先前的Trident版本升級，並且使用Azure NetApp Files，則`location`配置參數現在是必填的單例欄位。

修復

- 改進了 iSCSI 發起程序名稱的解析。 （"GitHub 問題 #681" ）
- 修復了不允許使用 CSI 儲存類別參數的問題。 （"GitHub 問題 #598" ）
- 修正了Trident CRD 中重複的按鍵聲明。 （"GitHub 問題 #671" ）
- 修復了不準確的 CSI 快照日誌。 （"GitHub 問題 #629" ））
- 修正了在已刪除節點上取消發布捲的問題。 （"GitHub 問題 #691" ）
- 增加了對區塊設備上檔案系統不一致性的處理。 （"GitHub 問題 #656" ）
- 修正了設定時自動拉取支撐影像的問題 `imageRegistry`安裝過程中標記。 （"GitHub 問題 #715" ）
- 修正了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

- 現在，與 Trident 安全端點的入站連線至少需要 TLS 1.3。 （"GitHub 問題 #698" ）
- Trident現在會在其安全端點的回應中加入 HSTS 標頭。
- Trident現在會嘗試自動啟用Azure NetApp FilesUnix 權限功能。
- **Kubernetes**： Trident daemonset 現在以 system-node-critical 優先權運行。 （"GitHub 問題 #694" ）

搬家

E系列驅動程式（自20.07版本起已停用）已移除。

22.01.1 中的變更

修復

- 修正了在已刪除節點上取消發布捲的問題。 （"GitHub 問題 #691" ）
- 修正了在ONTAP API 回應中存取聚合空間的 nil 欄位時發生的 panic 問題。

22.01.0 中的變更

修復

- **Kubernetes:** 增加大型叢集的節點註冊退避重試時間。
- 修正了 azure-netapp-files 驅動程式可能被多個同名資源混淆的問題。
- 如果用方括號指定， ONTAP SAN IPv6 DataLIF 現在可以正常運作。
- 修正了嘗試匯入已匯入磁碟區時傳回 EOF 導致 PVC 處於待處理狀態的問題。 （"GitHub 問題 #489" ）
- 修正了在SolidFire磁碟區上建立超過 32 個快照時Trident效能變慢的問題。
- 在建立 SSL 憑證時，將 SHA-1 替換為 SHA-256。
- 修正了Azure NetApp Files驅動程序，使其允許重複的資源名稱，並將操作限制在單一位置。
- 修正了Azure NetApp Files驅動程序，使其允許重複的資源名稱，並將操作限制在單一位置。

增強功能

- Kubernetes 增強功能：
  - 增加了對 Kubernetes 1.23 的支援。
  - 透過Trident Operator 或 Helm 安裝Trident pod 時，新增排程選項。 （"GitHub 問題 #651" ）
- 允許 GCP 驅動程式中的跨區域磁碟區。 （"GitHub 問題 #633" ）
- 為Azure NetApp Files區新增了對「unixPermissions」選項的支援。 （"GitHub 問題 #666" ）

棄用

Trident REST 介面只能監聽並提供服務於 127.0.0.1 或 [::1] 位址

21.10.1 中的變更

> v21.10.0 版本有一個問題，當節點被移除然後又被加入到 Kubernetes 叢集時， Trident控制器可能會進入 CrashLoopBackOff 狀態。此問題已在 v21.10.1 中修復（GitHub 問題 669）。

修復

- 修正了在 GCP CVS 後端匯入磁碟區時可能出現的競爭條件，該條件會導致匯入失敗。

- 修正了當節點被移除然後又被加入到 Kubernetes 叢集時， Trident控制器可能進入 CrashLoopBackOff 狀態的問題（GitHub 問題 669）。
- 修正了未指定 SVM 名稱時無法發現 SVM 的問題（GitHub 問題 612）。

21.10.0 中的變更

修復

- 修正了 XFS 磁碟區的克隆無法掛載到與來源磁碟區相同的節點上的問題（GitHub 問題 514）。
- 修正了Trident在關閉時記錄致命錯誤的問題（GitHub 問題 597）。
- 與 Kubernetes 相關的修復：
  - 使用下列命令建立快照時，將磁碟區的已使用空間作為最小還原大小傳回： `ontap-nas`和 `ontap-nas-flexgroup`驅動程式（GitHub 問題 645）。
  - 修復了以下問題 `Failed to expand filesystem`卷大小調整後記錄了錯誤（GitHub 問題 560）。
  - 修復了艙體可能卡住的問題 `Terminating`狀態（GitHub 問題 572）。
  - 修復了以下情況： `ontap-san-economy` FlexVol可能充滿了快照 LUN（GitHub 問題 533）。
  - 修正了使用不同鏡像時自訂 YAML 安裝程式的問題（GitHub 問題 613）。
  - 修復了快照大小計算（GitHub 問題 611）。
  - 修正了所有Trident安裝程式都能將普通 Kubernetes 識別為 OpenShift 的問題（GitHub 問題 639）。
  - 修正了Trident操作符，使其在 Kubernetes API 伺服器無法存取時停止協調（GitHub 問題 599）。

增強功能

- 增加了對 `unixPermissions`可選擇 GCP-CVS 性能卷。
- 為 GCP 中 600 GiB 到 1 TiB 範圍內的規模優化 CVS 磁碟區新增了支援。
- Kubernetes 相關增強功能：
  - 新增對 Kubernetes 1.22 的支援。
  - 使Trident operator 和 Helm chart 能夠與 Kubernetes 1.22 一起使用（GitHub 問題 628）。
  - 已新增操作員圖像 `tridentctl`images 指令（GitHub 問題 570）。

實驗性改進

- 增加了對卷複製的支持 `ontap-san`司機。
- 新增了對 REST 的*技術預覽*支持 `ontap-nas-flexgroup`， `ontap-san` ， 和 `ontap-nas-economy`司機。

已知問題

已知問題是指可能妨礙您成功使用產品的問題。

- 當已安裝Trident 的Kubernetes 叢集從 1.24 版本升級到 1.25 或更高版本時，必須更新 values.yaml 檔案進行設定。 `excludePodSecurityPolicy`到 `true`或添加 `--set excludePodSecurityPolicy=true`到 `helm

upgrade`升級叢集前需要執行此命令。

- Trident現在強制執行空白 `fsType` (`fsType=""`對於沒有捲的 `fsType`在其儲存類別中指定。使用 Kubernetes 1.17 或更高版本時， Trident支援提供一個空白的 `fsType`適用於 NFS 磁碟區。對於 iSCSI 卷，您需要設定 `fsType`在強制執行儲存類別時 `fsGroup`使用安全上下文。
- 當在多個Trident實例中使用後端時，每個後端設定檔都應該有不同的設定。 `storagePrefix` ONTAP後端的值，或使用其他值 `TenantName`適用於SolidFire後端。 Trident無法偵測到其他Trident實例所建立的磁碟區。嘗試在ONTAP或SolidFire後端建立現有磁碟區都會成功，因為Trident將磁碟區建立視為冪等操作。如果 `storagePrefix`或者 `TenantName`即使名稱相同，在相同後端建立的磁碟區也可能出現名稱衝突。
- 安裝Trident時（使用 `tridentctl`或Trident運算子）並使用 `tridentctl`要管理Trident，您應該確保 `KUBECONFIG`環境變數已設定。這是為了指明 Kubernetes 叢集。 `tridentctl`應該起到反作用。當使用多個 Kubernetes 環境時，您應該確保： `KUBECONFIG`文件來源準確無誤。
- 要對 iSCSI PV 執行線上空間回收，工作節點上的底層作業系統可能需要將掛載選項傳遞給磁碟區。對於 RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) 實例來說，情況確實如此，這些實例需要 `discard` "掛載選項"確保你的配置中包含 discard mountOption。[`StorageClass` ^] 支援在線區塊丟棄。
- 如果每個 Kubernetes 叢集中有多個Trident實例， Trident將無法與其他實例通信，也無法發現它們建立的其他卷，如果叢集中運行多個實例，則會導致意外和不正確的行為。每個 Kubernetes 叢集應該只有一個Trident實例。
- 如果基於三叉戟 `StorageClass`當Trident離線時，Kubernetes 中的物件會被刪除；當Trident重新上線時，它不會從資料庫中刪除對應的儲存類別。您應該使用以下命令刪除這些存儲類 `tridentctl`或使用 REST API。
- 如果使用者在刪除對應的 PVC 之前刪除了Trident提供的 PV， Trident不會自動刪除支援卷。您應該透過以下方式移除音量： `tridentctl`或使用 REST API。
- 除非每個配置請求的聚合集都是唯一的，否則ONTAP不能同時配置多個FlexGroup 。
- 使用 IPv6 上的Trident時，您應該指定 `managementLIF`和 `dataLIF`在方括號內的後端定義中。例如，`[fd20:8b1e:b258:2000:f816:3eff:feec:0]` 。

  您無法指定 `dataLIF`基於ONTAP SAN 後端。 Trident會發現所有可用的 iSCSI LIF，並使用它們來建立多路徑會話。

- 如果使用 `solidfire-san`使用 OpenShift 4.5 的驅動程式時，請確保底層工作節點使用 MD5 作為 CHAP 認證演算法。 Element 12.7 提供符合 FIPS 標準的 CHAP 安全演算法 SHA1、SHA-256 和 SHA3-256。

查找更多信息

- "Trident GitHub"
- "Trident部落格"

早期版本的文檔

如果您使用的不是Trident 25.06 版本，則可以根據以下資訊取得先前版本的文件："Trident支援生命週期" 。

- "Trident25.02"
- "Trident24.10"
- "Trident24.06"

- "Trident24.02"
- "Trident23.10"
- "Trident23.07"
- "Trident23.04"
- "Trident23.01"
- "Trident22.10"

已知問題

已知問題指明了可能會阻止您成功使用此版本產品的問題。

以下已知問題會影響目前版本：

恢復 **Restic** 備份的大檔案可能會失敗

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

了解Trident

了解Trident

Trident是由NetApp維護的完全支援的開源專案。它旨在幫助您使用業界標準介面（例如容器儲存介面 (CSI)）來滿足容器化應用程式的持久性需求。

什麼是Trident？

NetApp Trident支援在所有流行的NetApp儲存平台（包括公有雲和本機）上使用和管理儲存資源，例如本機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 。

Trident是符合容器儲存介面 (CSI) 標準的動態儲存編排器，可與下列系統原生整合："Kubernetes" 。 Trident以單一 Controller Pod 和叢集中每個工作節點上的 Node Pod 的形式運作。參考 "Trident架構" 了解詳情。

Trident也為NetApp儲存平台提供與 Docker 生態系統的直接整合。 NetApp Docker Volume Plugin (nDVP) 支援從儲存平台到 Docker 主機的儲存資源配置和管理。參考 "部署適用於 Docker 的Trident" 了解詳情。

> 如果您是第一次使用 Kubernetes，您應該先熟悉它。"Kubernetes 概念與工具" 。

Kubernetes 與NetApp產品集成

NetApp的儲存產品組合與 Kubernetes 叢集的許多面向集成，提供進階資料管理功能，從而增強 Kubernetes 部署的功能、能力、效能和可用性。

Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP"是一項完全託管的 AWS 服務，可讓您啟動並執行由NetApp ONTAP儲存作業系統支援的檔案系統。

Azure NetApp Files

"Azure NetApp Files"是由NetApp支援的企業級 Azure 檔案共用服務。您可以在 Azure 中原生運行要求最高的基於文件的工作負載，並獲得您期望從NetApp獲得的高效能和豐富的資料管理功能。

Cloud Volumes ONTAP

"Cloud Volumes ONTAP"是一款純軟體儲存設備，可在雲端運行ONTAP資料管理軟體。

Google Cloud NetApp Volumes

"Google Cloud NetApp Volumes"是 Google Cloud 中完全託管的文件儲存服務，提供高效能、企業級的文件儲存。

Element軟體

"元素"透過確保效能並實現簡化和精簡的儲存佈局，使儲存管理員能夠整合工作負載。

NetApp HCI

"NetApp HCI"透過自動化日常任務，簡化資料中心的管理和擴展，使基礎設施管理員能夠專注於更重要的功能。

Trident可以直接針對底層NetApp HCI儲存平台，為容器化應用程式設定和管理儲存設備。

NetApp ONTAP

"NetApp ONTAP"NetApp是 NetApp 的多協定統一儲存作業系統，可為任何應用程式提供進階資料管理功能。

ONTAP系統具有全快閃、混合或全 HDD 配置，並提供多種不同的部署模型：本機FAS、AFA 和ASA叢集、 ONTAP Select和Cloud Volumes ONTAP。 Trident支援這些ONTAP部署模型。

Trident架構

Trident以單一 Controller Pod 和叢集中每個工作節點上的 Node 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。

了解控制器 **pod** 和節點 **pod**

Trident以單一部署。Trident控制器艙以及一個或多個Trident節點 Pod在 Kubernetes 叢集上，並使用標準的 Kubernetes *CSI Sidecar Containers* 來簡化 CSI 插件的部署。"Kubernetes CSI Sidecar 容器"由 Kubernetes 儲存社群維護。

Kubernetes"節點選擇器"和"容忍與污點"用於限制 pod 在特定或首選節點上運行。在Trident安裝過程中，您可以為控制器和節點 pod 設定節點選擇器和容差。

- 控制器插件負責磁碟區的配置和管理，例如快照和調整大小。
- 節點插件負責將儲存連接到節點。

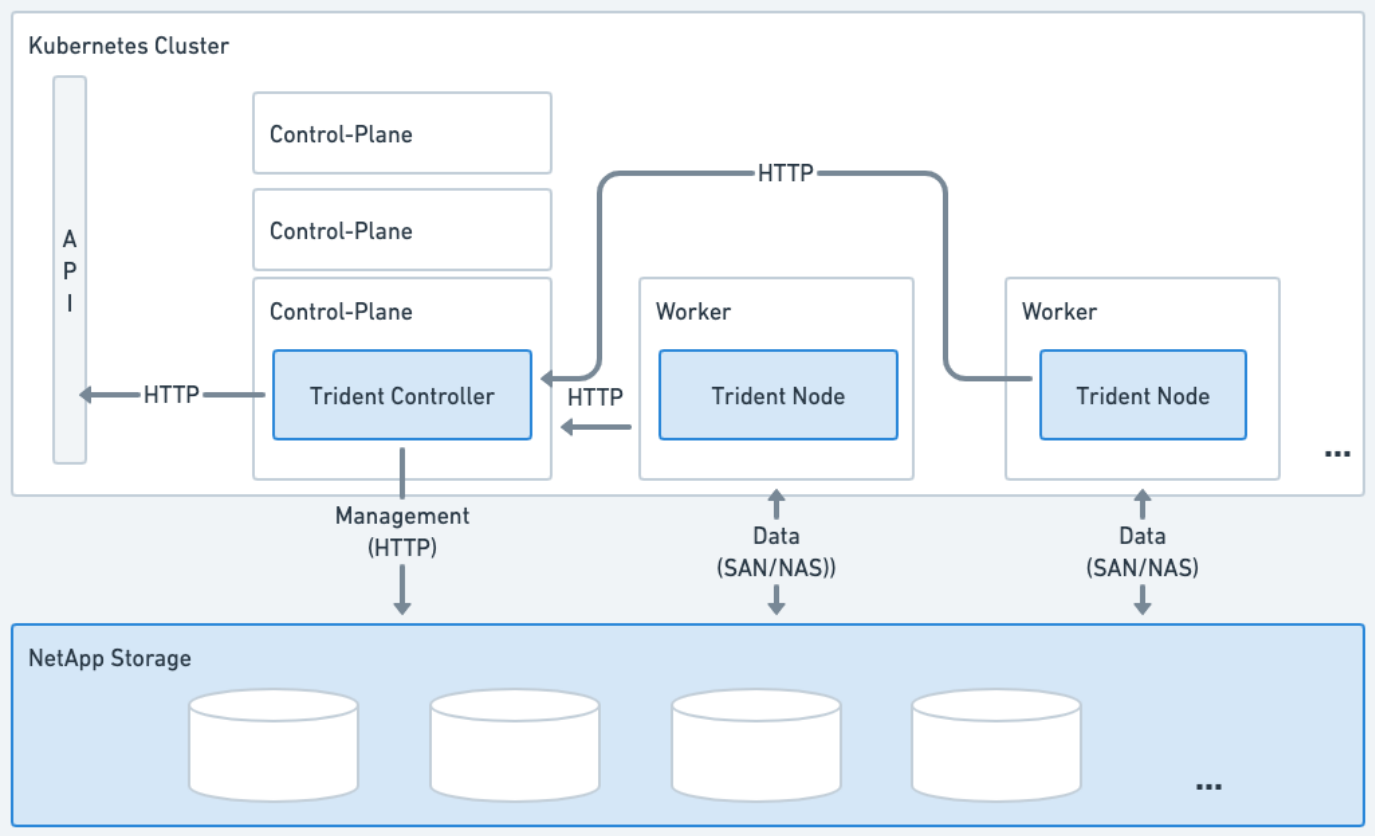


圖 **1. Trident**已部署在 **Kubernetes** 叢集上

Trident控制器艙

Trident Controller Pod 是一個運行 CSI Controller 插件的單一 Pod。

- 負責在NetApp儲存中配置和管理卷
- 由 Kubernetes 部署管理
- 根據安裝參數的不同，可以在控制平面節點或工作節點上運作。

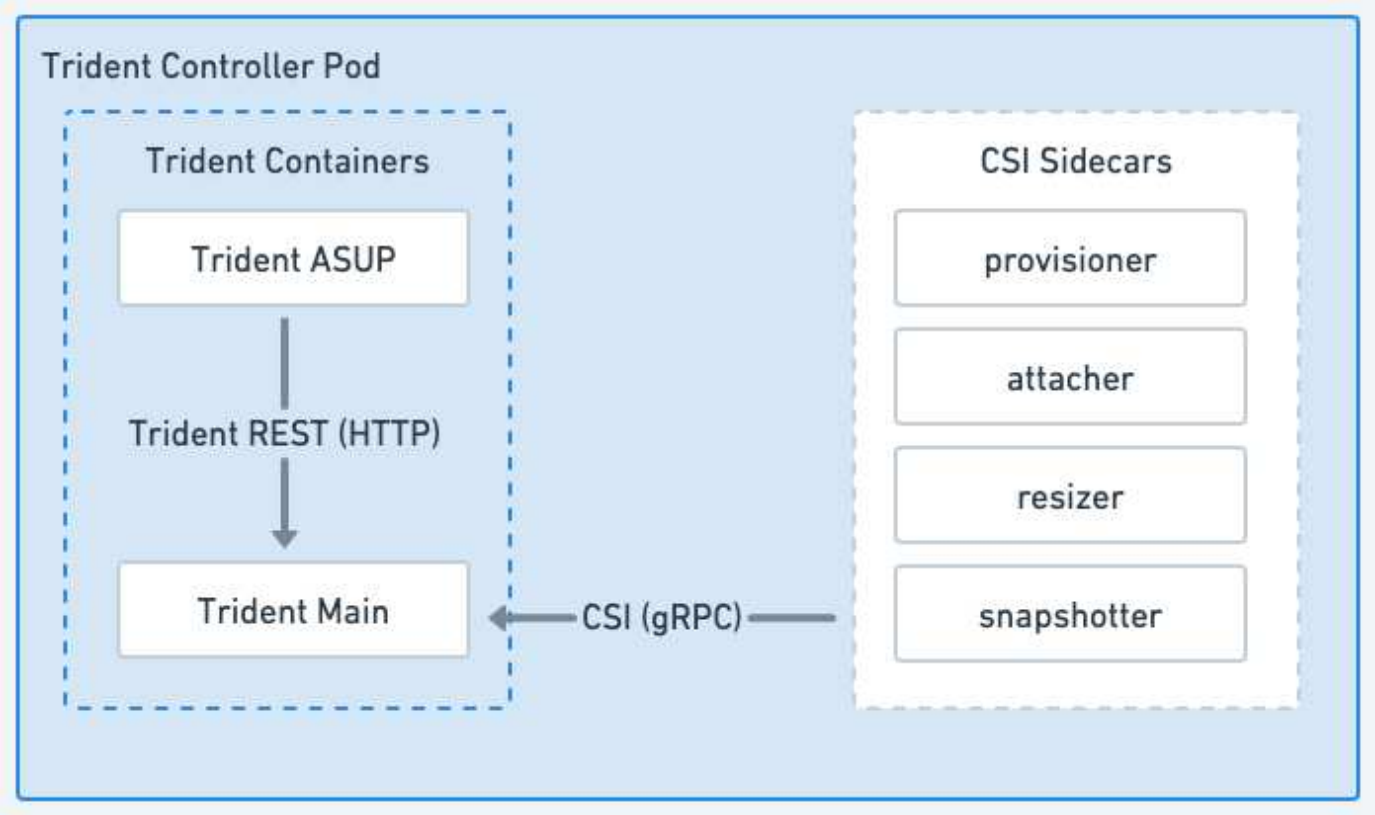


圖 **2. Trident**控制器艙示意圖

Trident節點 **Pod**

Trident Node Pod 是運行 CSI Node 外掛的特權 Pod。

- 負責掛載和卸載主機上運行的 Pod 的儲存設備
- 由 Kubernetes DaemonSet 管理
- 必須在任何能夠掛載NetApp儲存的節點上執行

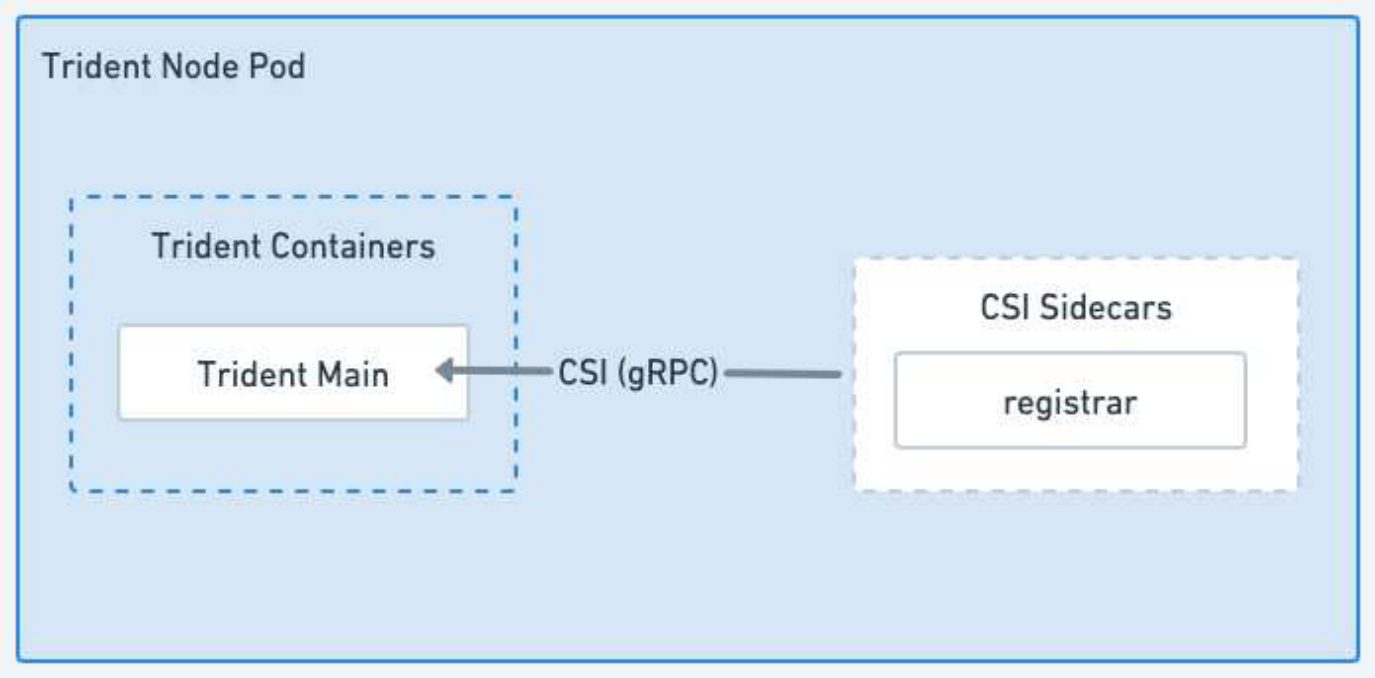


圖 **3. Trident**節點艙示意圖

支援的 **Kubernetes** 叢集架構

Trident支援以下 Kubernetes 架構：

Kubernetes叢集架構	支援	預設安裝
單主控計算	是的	是的
多主計算	是的	是的
掌握， `etcd`計算	是的	是的
主控、基礎設施、計算	是的	是的

概念

供應

Trident的配置流程分為兩個主要階段。第一階段將儲存類別與一組適當的後端儲存池關聯起來，這是在進行設定之前必須進行的準備工作。第二階段包含磁碟區的建立本身，需要從與待建立磁碟區的儲存類別關聯的儲存池中選擇儲存池。

儲存類別關聯

將後端儲存池與儲存類別關聯起來，取決於儲存類別的請求屬性及其 `storagePools`， `additionalStoragePools` ， 和 `excludeStoragePools`列表。建立儲存類別時， Trident會將其每個後端提供的屬性和池與儲存類別請求的屬性和池進行比較。如果儲存池的屬性和名稱與所有要求的屬性和池名稱相符，Trident會將該儲存池新增至該儲存類別的適用儲存池集合中。此外， Trident還新增了清單中列出的所有儲存

池。 `additionalStoragePools`即使它們的屬性不滿足儲存類別的所有或任何請求屬性，也要將其新增至該集合。你應該使用 `excludeStoragePools`列出要覆蓋和移除儲存類別所使用的儲存池。每次新增的後端時，Trident都會執行類似的流程，檢查其儲存池是否符合現有儲存類別的要求，並刪除任何被標記為排除的儲存池。

銷售創造

Trident然後利用儲存類別和儲存池之間的關聯來確定在哪裡配置磁碟區。建立磁碟區時， Trident首先取得該磁碟區儲存類別的儲存池集合，如果您為該磁碟區指定了協議， Trident會刪除那些無法提供所要求協定的儲存池（例如， NetApp HCI/ SolidFire後端無法提供基於檔案的磁碟區，而ONTAP NAS 後端無法提供基於區塊的磁碟區）。 Trident會隨機化所得集合的順序，以方便磁碟區的均勻分佈，然後遍歷該集合，依序嘗試在每個儲存池上配置磁碟區。如果一次成功，則傳回成功結果，並將流程中遇到的任何失敗記錄下來。 Trident僅在無法為要求的儲存類別和協定配置所有可用的儲存池時才會傳回失敗。

卷快照

了解更多關於Trident如何處理其驅動程式的磁碟區快照所建立的資訊。

了解如何建立磁碟區快照

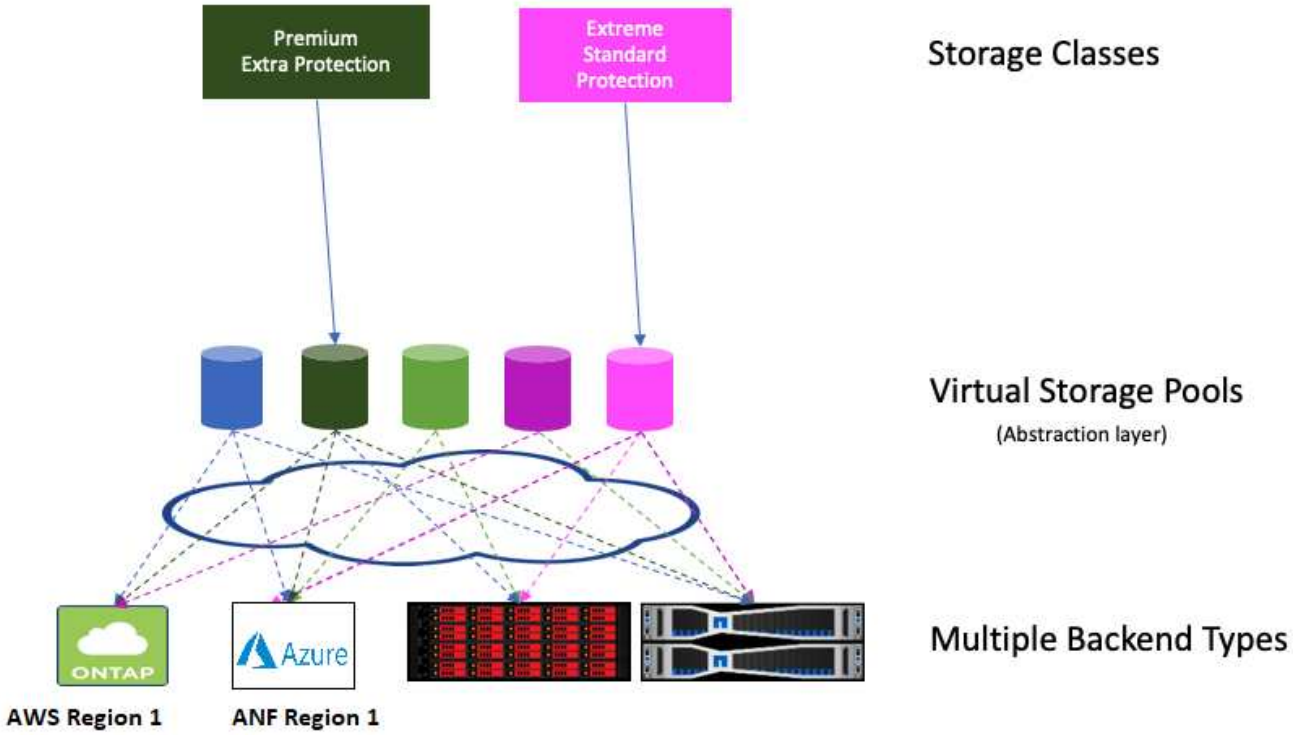
- 對於 `ontap-nas`， `ontap-san` ， `gcp-cvs` ， 和 `azure-netapp-files`在驅動程式中，每個持久性磁碟區 (PV) 都會對應到一個FlexVol volume。因此，磁碟區快照被創建為NetApp快照。 NetApp快照技術比同類快照技術具有更高的穩定性、可擴展性、可恢復性和效能。這些快照副本在創建所需時間和儲存空間方面都非常有效率。
- 對於 `ontap-nas-flexgroup`在驅動程式中，每個持久性磁碟區 (PV) 都會對應到一個FlexGroup。因此，磁碟區快照將建立為NetApp FlexGroup快照。 NetApp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。
- 對於 `ontap-san-economy`驅動程序，PV 對應到在共享FlexVol磁碟區上建立的 LUN。透過對關聯的 LUN 執行 FlexClone 操作，即可獲得 PV 的 VolumeSnapshot。 ONTAP FlexClone技術幾乎可以瞬間建立即使是最大資料集的副本。副本與其父級共享資料塊，除了元資料所需的空間外，不佔用任何儲存空間。
- 對於 `solidfire-san`驅動程式中，每個 PV 會對應到在NetApp Element軟體/ NetApp HCI叢集上建立的 LUN。 VolumeSnapshots 由底層 LUN 的 Element 快照表示。這些快照是特定時間點的副本，僅佔用少量系統資源和空間。
- 當與 `ontap-nas`和 `ontap-san`在驅動程式中， ONTAP快照是FlexVol的某個時間點的副本，會佔用FlexVol本身的空間。隨著快照的建立/計劃，這會導致磁碟區中的可寫入空間隨時間減少。解決此問題的一個簡單方法是透過 Kubernetes 調整大小來增加磁碟區。另一種方法是刪除不再需要的快照。當透過 Kubernetes 建立的 VolumeSnapshot 被刪除時， Trident將刪除關聯的ONTAP快照。並非透過 Kubernetes 建立的ONTAP快照也可以刪除。

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，適用於支援的ONTAP和 CVS 後端。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。

虛擬池

虛擬池在Trident儲存後端和 Kubernetes 之間提供了一個抽象層。 `StorageClasses` 。它們允許管理員以通用的、與後端無關的方式定義每個後端的各個方面，例如位置、效能和保護，而無需建立單獨的配置。 `StorageClass`指定要使用的實體後端、後端池或後端類型，以滿足所需條件。

儲存管理員可以在任何Trident後端上，透過 JSON 或 YAML 定義檔定義虛擬池。



在虛擬池列表之外指定的任何方面對於後端都是全域的，並將應用於所有虛擬池；而每個虛擬池可以單獨指定一個或多個方面（覆蓋任何後端全域方面）。

- 定義虛擬池時，請勿嘗試重新排列後端定義中現有虛擬池的順序。
- 我們不建議修改現有虛擬池的屬性。您需要定義一個新的虛擬池來進行變更。

大多數方面都是用後端特有的術語來描述的。至關重要的是，這些方面值不會暴露在後端驅動程式之外，也無法用於匹配。 `StorageClasses`相反，管理員可以為每個虛擬池定義一個或多個標籤。每個標籤都是一個鍵值對，標籤可能在不同的後端中是相同的。與方麵類似，標籤可以針對每個池進行指定，也可以全域指定到後端。與具有預先定義名稱和值的方面不同，管理員可以完全自主地根據需要定義標籤鍵和值。為了方便起見，儲存管理員可以為每個虛擬池定義標籤，並按標籤將磁碟區分組。

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 `-`

一個 `StorageClass`透過引用選擇器參數中的標籤來確定要使用的虛擬池。虛擬池選擇器支援以下運算符：

操作員	例子	池的標籤值必須：
=	性能=優質	匹配
!=	性能！ =極限	不匹配
in	位置在（東，西）	屬於價值集合。
notin	表演獎（銀獎、銅獎）	不在值集中
<key>	保護	存在且具有任意值
!<key>	！保護	不存在

卷訪問群組

了解更多關於Trident如何使用 "卷訪問群組"。

> 如果您使用的是 CHAP，請忽略此部分。建議使用 CHAP 以簡化管理並避免下文所述的擴展限制。此外，如果您在 CSI 模式下使用Trident ，則可以忽略此部分。 Trident在作為增強型 CSI 設定器安裝時使用 CHAP。

了解卷訪問群組

Trident可以使用磁碟區存取群組來控制對其所配置磁碟區的存取。如果 CHAP 被停用，它會尋找一個名為「存取群組」的存取群組。 `trident`除非您在設定中指定一個或多個存取群組 ID。

Trident會將新磁碟區與已設定的存取群組關聯起來，但它本身並不會建立或以其他方式管理存取群組。在將儲存後端新增至Trident之前，存取群組必須存在，且它們需要包含 Kubernetes 叢集中每個節點的 iSCSI IQN，這些節點可能會掛載由該後端配置的磁碟區。在大多數安裝中，這包括叢集中的每個工作節點。

對於節點數超過 64 個的 Kubernetes 集群，您應該使用多個存取組。每個訪問組最多可包含 64 個 IQN，每個卷可屬於四個訪問組。配置最多四個存取群組後，叢集中最多 256 個節點中的任何節點都將能夠存取任何磁碟區。有關磁碟區存取群組的最新限制，請參閱 "這裡"。

如果您要修改的配置是基於預設配置的，那麼請注意以下事項。 `trident`訪問群組必須與其他群組一起使用，並且必須包含該群組的 ID。 `trident`存取清單中的群組。

Trident快速入門

只需幾個步驟，即可安裝Trident並開始管理儲存資源。開始之前，請先回顧一下"Trident的要求"。

> 有關 Docker 的信息，請參閱"Trident for Docker"。

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為 Pod 設定的磁碟區。

"準備工作節點"

2 安裝**Trident**

Trident提供多種安裝方法和模式，並針對各種環境和組織進行了最佳化。

"安裝Trident"

3 創建後端

後端定義了Trident與儲存系統之間的關係。它告訴Trident如何與該儲存系統通信，以及Trident應該如何從中設定磁碟區。

"配置後端"適用於您的儲存系統

4 建立 **Kubernetes** 儲存類

Kubernetes StorageClass 物件指定Trident作為配置器，並允許您建立儲存類別以配置具有可自訂屬性的磁碟區。 Trident為指定Trident設定器的 Kubernetes 物件建立相符的儲存類別。

"建立儲存類別"

5 提供一定量

持久性磁碟區 (PV) 是 Kubernetes 叢集上由叢集管理員配置的實體儲存資源。 *PersistentVolumeClaim* (PVC) 是對叢集上持久卷的存取請求。

建立一個持久性磁碟區 (PV) 和一個持久性磁碟區宣告 (PVC)，使用設定的 Kubernetes StorageClass 請求存取 PV。然後您可以將光伏組件安裝到支架上。

"提供一定量"

下一步是什麼？

現在您可以新增其他後端、管理儲存類別、管理後端以及執行磁碟區操作。

要求

在安裝Trident之前，您應該查看這些通用系統需求。特定後端可能還有其他要求。

關於**Trident**的關鍵訊息

*您必須閱讀以下關於Trident的重要資訊。 *

- Trident現已支援 Kubernetes 1.34。在升級 Kubernetes 之前先升級Trident 。
- Trident嚴格強制要求在 SAN 環境中使用多路徑配置，建議值為 `find_multipaths: no`在 multipath.conf 檔案中。

  使用非多路徑配置或使用 `find_multipaths: yes`或者 `find_multipaths: smart`multipath.conf 檔案中的值會導致掛載失敗。 Trident建議使用 `find_multipaths: no`自 21.07 版本發布以來。

支援的前端（編排器）

Trident支援多種容器引擎和編排器，包括以下幾種：

- Anthos On-Prem (VMware) 和 Anthos on Bare Metal 1.16
- Kubernetes 1.27 - 1.34
- OpenShift 4.12、4.14 - 4.19（如果您打算使用 OpenShift 4.19 進行 iSCSI 節點準備，則支援的最低Trident版本為 25.06.1。）

  Trident繼續支援舊版的 OpenShift，以符合…"Red Hat 擴充更新支援 (EUS) 發布生命週期"即使他們依賴上游不再官方支援的 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）v1.27.x - 1.34.x

  雖然Trident在 Rancher Kubernetes Engine 2 (RKE2) 版本 1.27.x - 1.34.x 上受支持，但Trident目前僅在 RKE2 v1.28.5+rke2r1 上獲得認證。

Trident也與許多其他完全託管和自架的 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可用作儲存提供者"KubeVirt"。

在將已安裝Trident的 Kubernetes 叢集從 1.25 版本升級至 1.26 或更高版本之前，請參閱下列內容："升級 Helm 安裝" 。

支援的後端（儲存）

要使用Trident，您需要以下一個或多個支援的後端：

- Amazon FSx for NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp Volumes

- NetApp全 SAN 陣列 (ASA)
- NetApp 有限支援下的本機FAS、 AFF、Select 或ASA r2（iSCSI 和 NVMe/TCP）叢集版本。看"軟體版本支持"。
- NetApp HCI/Element 軟體 11 或更高版本

Trident對 KubeVirt 和 OpenShift 虛擬化的支持

支援的儲存驅動程式：

Trident支援以下適用於 KubeVirt 和 OpenShift 虛擬化的ONTAP驅動程式：

- ontap-nas
- ontap-nas-economy
- ontap-san（基於 TCP 的 iSCSI、FCP、NVMe）
- ontap-san-economy（僅限 iSCSI）

需要考慮的要點：

- 更新存儲類別以使其具有 `fsType`參數（例如： `fsType: "ext4"`在 OpenShift 虛擬化環境中。如有需要，請使用以下方式明確設定音量模式為阻止模式： `volumeMode=Block`參數 `dataVolumeTemplates`通知 CDI 建立區塊資料卷。
- 區塊儲存驅動程式的 *RWX* 存取模式: ontap-san (iSCSI、NVMe/TCP、FC) 和 ontap-san-economy (iSCSI) 驅動程式僅支援「volumeMode: Block」（原始裝置）。對於這些司機來說， `fstype`由於磁碟區是以原始設備模式提供的，因此無法使用該參數。
- 對於需要 RWX 存取模式的即時遷移工作流程，支援以下組合：
  - NFS+ `volumeMode=Filesystem`
  - iSCSI+ `volumeMode=Block` （原始設備）
  - NVMe/TCP + `volumeMode=Block` （原始設備）
  - FC+ `volumeMode=Block` （原始設備）

功能需求

下表總結了此版本Trident提供的功能及其支援的 Kubernetes 版本。

特徵	**Kubernetes** 版本	需要設置特色門嗎？
Trident	1.27 - 1.34	不
卷快照	1.27 - 1.34	不
PVC 來自磁碟區快照	1.27 - 1.34	不
iSCSI PV 調整大小	1.27 - 1.34	不
ONTAP雙向 CHAP	1.27 - 1.34	不

特徵	**Kubernetes** 版本	需要設置特色門嗎？
動態出口政策	1.27 - 1.34	不
Trident操作員	1.27 - 1.34	不
CSI拓撲	1.27 - 1.34	不

測試過的主機作業系統

雖然Trident官方並未正式支援特定作業系統，但已知以下作業系統可以正常運作：

- 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 或更高版本。

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

預設情況下， Trident在容器中運行，因此可以在任何 Linux 工作節點上運行。但是，這些工作人員需要能夠使用標準 NFS 用戶端或 iSCSI 發起程序掛載Trident提供的磁碟區，具體取決於您使用的後端。

這 `tridentctl`該實用程式也可在上述任何 Linux 發行版上運行。

主機配置

Kubernetes 叢集中的所有工作節點都必須能夠掛載您為 Pod 設定的磁碟區。若要準備工作節點，您必須根據所選驅動程式安裝 NFS、iSCSI 或 NVMe 工具。

"準備工作節點"

儲存系統配置

Trident可能需要對儲存系統進行更改，後端配置才能使用它。

"配置後端"

Trident港口

Trident需要存取特定連接埠才能進行通訊。

"Trident港口"

容器鏡像和相應的 **Kubernetes** 版本

對於實體隔離安裝，以下清單是安裝Trident所需的容器鏡像參考。使用 `tridentctl images`用於驗證所需容器鏡像清單的命令。

Trident 25.06.2 所需的容器鏡像

Kubernetes 版本	容器影像
v1.27.0、v1.28.0、v1.29.0、v1.30.0、v1.31.0、v1.32.0、v1.33.0、v1.34.0	• docker.io/netapp/trident:25.06.2 • docker.io/netapp/trident-autosupport:25.06 • registry.k8s.io/sig-storage/csi-provisioner:v5.2.0 • registry.k8s.io/sig-storage/csi-attacher:v4.8.1 • registry.k8s.io/sig-storage/csi-resizer:v1.13.2 • registry.k8s.io/sig-storage/csi-snapshotter:v8.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.13.0 • docker.io/netapp/trident-operator:25.06.2（選購）

Trident 25.06 所需的容器鏡像

Kubernetes 版本	容器影像
v1.27.0、v1.28.0、v1.29.0、v1.30.0、v1.31.0、v1.32.0、v1.33.0、v1.34.0	• docker.io/netapp/trident:25.06.0 • docker.io/netapp/trident-autosupport:25.06 • registry.k8s.io/sig-storage/csi-provisioner:v5.2.0 • registry.k8s.io/sig-storage/csi-attacher:v4.8.1 • registry.k8s.io/sig-storage/csi-resizer:v1.13.2 • registry.k8s.io/sig-storage/csi-snapshotter:v8.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.13.0 • docker.io/netapp/trident-operator:25.06.0（選購）

安裝Trident

使用Trident操作員進行安裝

使用 tridentctl 安裝

使用 OpenShift 認證操作員進行安裝

使用Trident

準備工作節點

Kubernetes 叢集中的所有工作節點都必須能夠掛載您為 Pod 設定的磁碟區。若要準備工作節點，您必須根據所選驅動程式安裝 NFS、iSCSI、NVMe/TCP 或 FC 工具。

選擇合適的工具

如果您使用的是多種驅動程序，則應安裝所有驅動程式所需的工具。最新版本的 Red Hat Enterprise Linux CoreOS (RHCOS) 預設安裝了這些工具。

NFS 工具

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

iSCSI 工具

"安裝 iSCSI 工具"如果您正在使用： `ontap-san` ， `ontap-san-economy` ， `solidfire-san` 。

NVMe 工具

"安裝 NVMe 工具"如果你正在使用 `ontap-san`用於基於 TCP 的非揮發性記憶體高速介面 (NVMe) (NVMe/TCP) 協定。

NetApp建議 NVMe/TCP 使用ONTAP 9.12 或更高版本。

透過光纖通道 **(FC)** 進行 **SCSI** 的工具

請參閱"配置 FC 和 FC-NVMe SAN 主機的方法"有關配置 FC 和 FC-NVMe SAN 主機的詳細資訊。

"安裝 FC 工具"如果你正在使用 `ontap-san`使用 `sanType` `fcp` （透過光纖通道進行 SCSI 通訊）。

需要考慮的要點：* OpenShift 和 KubeVirt 環境支援透過 FC 進行 SCSI 通訊。 * Docker 不支援透過 FC 傳輸 SCSI。 * iSCSI 自癒功能不適用於基於 FC 的 SCSI。

節點服務發現

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

節點服務發現功能可以辨識已發現的服務，但無法保證服務配置正確。反之，未發現服務並不保證卷掛載一定會失敗。

回顧事件

Trident會為節點建立事件，以識別已發現的服務。要查看這些事件，請運行：

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

查看已發現的服務

Trident識別Trident節點 CR 上每個節點啟用的服務。若要查看已發現的服務，請執行：

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

NFS卷

使用適用於您作業系統的命令安裝 NFS 工具。確保NFS服務在啟動時啟動。

RHEL 8+

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

Ubuntu

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

安裝 NFS 工具後重新啟動工作節點，以防止將磁碟區附加到容器時發生故障。

iSCSI 卷

Trident可以自動建立 iSCSI 會話、掃描 LUN、發現多路徑裝置、格式化裝置並將其掛載到 pod 中。

iSCSI自癒能力

對於ONTAP系統， Trident每五分鐘運行一次 iSCSI 自癒程序，以：

1. *確定*所需的 iSCSI 會話狀態和目前的 iSCSI 會話狀態。
2. 將理想狀態與目前狀態進行比較，以確定需要進行的維修。 Trident確定維修優先順序以及何時進行搶修。
3. 執行必要的修復，使目前的 iSCSI 會話狀態恢復到所需的 iSCSI 會話狀態。

自癒活動的日誌位於… `trident-main`對應 Daemonset pod 上的容器。要查看日誌，您必須已設定 `debug`在Trident安裝過程中設定為「true」。

Trident iSCSI 的自癒功能可以幫助預防：

- 網路連線問題後可能出現過期或不健康的 iSCSI 會話。如果會話已過期， Trident會等待七分鐘，然後登出並重新與入口網站建立連線。

例如，如果儲存控制器上輪換了 CHAP 金鑰，並且網路失去連接，則舊的（過時的）CHAP 金鑰可能會保留下來。自癒功能可以識別這一點，並自動重新建立會話以套用更新後的 CHAP 金鑰。

- 缺少 iSCSI 會話
- 缺少 LUN

升級**Trident**之前需要考慮的要點

- 如果僅使用每個節點的 igroup（在 23.04+ 中引入），則 iSCSI 自癒功能將啟動 SCSI 總線上所有裝置的 SCSI 重新掃描。
- 如果僅使用後端範圍的 igroup（自 23.04 版本起已棄用），則 iSCSI 自癒功能將啟動 SCSI 重新掃描，以尋找 SCSI 總線上的確切 LUN ID。
- 如果同時使用節點級 igroup 和後端級 igroup，iSCSI 自癒功能將啟動 SCSI 重新掃描，以尋找 SCSI 總線上的精確 LUN ID。

安裝 **iSCSI** 工具

使用適用於您作業系統的指令安裝 iSCSI 工具。

開始之前

- Kubernetes 叢集中的每個節點都必須有一個唯一的 IQN。這是必要的前提條件。
- 如果使用 RHCOS 4.5 或更高版本，或其他與 RHEL 相容的 Linux 發行版，則需要執行以下操作：`solidfire-san`對於驅動程式和Element OS 12.5 或更早版本，請確保將 CHAP 驗證演算法設定為 MD5。 `/etc/iscsi/iscsid.conf` Element 12.7 提供符合 FIPS 標準的 CHAP 安全演算法 SHA1 、SHA-256 和 SHA3-256。

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

- 當使用執行 RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) 且具有 iSCSI PV 的工作節點時，請指定 `discard`StorageClass 中的 mountOption 用於執行內嵌空間回收。參考 "紅帽文檔"。
- 請確保您已升級至最新版本。 `multipath-tools` 。

RHEL 8+

1. 安裝以下系統軟體包：

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

2. 請檢查 iscsi-initiator-utils 版本是否為 6.2.0.874-2.el7 或更高版本：

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

3. 將掃描方式設定為手動：

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

4. 啟用多路徑：

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

> 確保 `/etc/multipath.conf`包含 `find_multipaths no`在下面 `defaults`。

5. 確保 `iscsid`和 `multipathd`正在運行：

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

6. 啟用並啟動 `iscsi`：

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

Ubuntu

1. 安裝以下系統軟體包：

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

2. 請檢查 open-iscsi 版本是否為 2.0.874-5ubuntu2.10 或更高版本（適用於 bionic）或 2.0.874-7.1ubuntu6.1 或更高版本（適用於 focal）：

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

3. 將掃描方式設定為手動：

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

4. 啟用多路徑：

```
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`在下面 `defaults。

5. 確保 `open-iscsi`和 `multipath-tools`已啟用並正在運行：

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

> 對於 Ubuntu 18.04，您必須使用以下命令發現目標連接埠： `iscsiadm`開始之前 `open-iscsi`啟動 iSCSI 守護程式。您也可以修改 `iscsi`服務啟動 `iscsid`自動地。

配置或停用 **iSCSI** 自愈

您可以設定以下Trident iSCSI 自癒設定來修復過期的會話：

- **iSCSI** 自癒間隔：決定 iSCSI 自癒的呼叫頻率（預設值：5 分鐘）。你可以透過設定較小的數字來增加運作頻率，或是設定較大的數字來降低運作頻率。

> 將 iSCSI 自癒間隔設為 0 將完全停止 iSCSI 自癒功能。我們不建議停用 iSCSI 自癒功能；只有在 iSCSI 自癒功能無法如預期運作或出於除錯目的時，才應停用該功能。

- **iSCSI** 自癒等待時間：確定 iSCSI 自癒在登出不健康的會話並嘗試再次登入之前等待的時間（預設值：7 分

鐘）。您可以將其配置為更大的數字，以便被識別為不健康的會話必須等待更長時間才能登出，然後再嘗試重新登入；或配置為較小的數字，以便更快地登出和登入。

舵

若要配置或變更 iSCSI 自癒設置，請傳遞以下參數： `iscsiSelfHealingInterval`和 `iscsiSelfHealingWaitTime`Helm 安裝或 Helm 更新期間的參數。

以下範例將 iSCSI 自癒間隔設定為 3 分鐘，自癒等待時間設定為 6 分鐘：

```
helm install trident trident-operator-100.2506.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

三叉戟

若要配置或變更 iSCSI 自癒設置，請傳遞以下參數： `iscsi-self-healing-interval`和 `iscsi-self-healing-wait-time`tridentctl 安裝或更新期間的參數。

以下範例將 iSCSI 自癒間隔設定為 3 分鐘，自癒等待時間設定為 6 分鐘：

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

NVMe/TCP 卷

使用適用於您作業系統的指令安裝 NVMe 工具。

- NVMe 需要 RHEL 9 或更高版本。
- 如果您的 Kubernetes 節點的核心版本太舊，或者您的核心版本沒有 NVMe 軟體包，則您可能需要將節點的核心版本更新為包含 NVMe 軟體包的版本。

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

安裝完成後，使用以下命令驗證 Kubernetes 叢集中的每個節點是否具有唯一的 NQN：

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

Trident修改了 `ctrl_device_tmo`確保 NVMe 在路徑中斷時不會放棄路徑的值。請勿更改此設定。

SCSI over FC 卷

現在您可以使用光纖通道 (FC) 協定和Trident在ONTAP系統上設定和管理儲存資源。

先決條件

配置 FC 所需的網路和節點設定。

網路設定

1. 取得目標介面的 WWPN。請參閱 "網路介面顯示"了解更多。
2. 取得發起方（主機）上介面的 WWPN。

   請參考對應的主機作業系統實用程式。

3. 使用主機和目標的 WWPN 在 FC 交換器上設定區域。

   有關信息，請參閱相應交換機供應商的文檔。

   詳情請參閱以下ONTAP文件：

   - "光纖通道和FCoE分區概述"
   - "配置 FC 和 FC-NVMe SAN 主機的方法"

安裝 **FC** 工具

使用適用於您作業系統的命令安裝 FC 工具。

- 當使用執行 RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) 且具有 FC PV 的工作節點時，請指定 `discard`StorageClass 中的 mountOption 用於執行內嵌空間回收。參考 "紅帽文檔"。

RHEL 8+

1. 安裝以下系統軟體包：

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

2. 啟用多路徑：

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

> 確保 /etc/multipath.conf`包含 `find_multipaths no`在下面 `defaults。

3. 確保 `multipathd`正在運行：

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

Ubuntu

1. 安裝以下系統軟體包：

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

2. 啟用多路徑：

```
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`在下面 `defaults。

3. 確保 `multipath-tools`已啟用並正在運行：

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

設定和管理後端

配置後端

後端定義了Trident與儲存系統之間的關係。它告訴Trident如何與該儲存系統通信，以及Trident應該如何從中設定磁碟區。

Trident會自動提供符合儲存類別定義要求的後端儲存池。了解如何配置儲存系統的後端。

- "設定Azure NetApp Files後端"
- "設定Google Cloud NetApp Volumes後端"
- "為 Google Cloud Platform 後端設定Cloud Volumes Service"
- "配置NetApp HCI或SolidFire後端"
- "使用ONTAP或Cloud Volumes ONTAP NAS 驅動程式設定後端"
- "使用ONTAP或Cloud Volumes ONTAP SAN 驅動程式設定後端"
- "將Trident與Amazon FSx for NetApp ONTAP"

Azure NetApp Files

設定**Azure NetApp Files**後端

您可以將Azure NetApp Files設定為Trident的後端。您可以使用Azure NetApp Files後端附加 NFS 和 SMB 磁碟區。 Trident也支援使用託管識別碼對 Azure Kubernetes 服務 (AKS) 叢集進行憑證管理。

Azure NetApp Files驅動程式詳細信息

Trident提供以下Azure NetApp Files儲存驅動程序，以便與叢集通訊。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

司機	協定	音量模式	支援的存取模式	支援的檔案系統
`azure-netapp-files`	NFS SMB	檔案系統	RWO、ROX、RWX、RWOP	`nfs`， `smb`

注意事項

- Azure NetApp Files服務不支援小於 50 GiB 的磁碟區。如果要求的磁碟區較小， Trident會自動建立 50GiB 的磁碟區。
- Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。

為 **AKS** 管理身份

Trident支持"託管身分"適用於 Azure Kubernetes 服務叢集。若要利用託管身分提供的簡化憑證管理功能，您必須具備以下條件：

- 使用 AKS 部署的 Kubernetes 集群
- 在 AKS Kubernetes 叢集上配置的託管身份
- 已安裝的Trident包括 `cloudProvider`指定 `"Azure"`。

Trident操作員

若要使用Trident操作員安裝Trident ，請編輯 `tridentorchestrator_cr.yaml`設定 `cloudProvider`到 `"Azure"`。例如：

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

舵

以下範例安裝Trident套裝 `cloudProvider`使用環境變數連接到 `Azure` `$CP`：

```
helm install trident trident-operator-100.2506.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

以下範例安裝Trident並進行設置 `cloudProvider`標記 `Azure`：

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

AKS 的雲端身份

雲端身分使 Kubernetes pod 能夠透過作為工作負載身分進行驗證來存取 Azure 資源，而無需提供明確的 Azure 憑證。

若要在 Azure 中利用雲端身分功能，您必須具備以下條件：

- 使用 AKS 部署的 Kubernetes 集群
- 在 AKS Kubernetes 叢集上設定工作負載身分和 oidc-issuer
- 已安裝的Trident包括 `cloudProvider`指定 `"Azure"`和 `cloudIdentity`指定工作負載標識

Trident操作員

若要使用Trident操作員安裝Trident ，請編輯 `tridentorchestrator_cr.yaml`設定 `cloudProvider`到 `"Azure"`並設定 `cloudIdentity`到 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`。

例如：

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

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-
xxxxxxxxxx'"
```

以下範例安裝Trident並進行設置 `cloudProvider`使用環境變數連接到 `Azure` `$CP`並設定 `cloudIdentity`使用環境變數 `$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 雲端提供者 和 雲端身分 標誌的值：

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

以下範例安裝Trident並進行設置 `cloud-provider`標記 `$CP`，和 `cloud-identity`到 `$CI`：

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

在設定Azure NetApp Files後端之前，需要確保滿足以下要求。

NFS 和 SMB 磁碟區的先決條件

如果您是第一次使用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檔案並建立 NFS 磁碟區。參考 "Azure：設定Azure NetApp Files並建立 NFS 卷"。

配置和使用 "Azure NetApp Files"後端需要以下組件：

> - `subscriptionID`, `tenantID`, `clientID`, `location`, 和 `clientSecret`在 AKS 叢集上使用託管身分時，這些是可選的。
> - `tenantID`, `clientID`, 和 `clientSecret`在 AKS 叢集上使用雲端身分時，這些是可選的。

- 容量池。參考"微軟：為Azure NetApp Files建立容量池"。
- 委派給Azure NetApp Files 的子網路。參考"Microsoft：將子網路委派給Azure NetApp Files"。
- `subscriptionID`來自已啟用Azure NetApp Files功能的 Azure 訂閱。
- `tenantID`, `clientID`, 和 `clientSecret`來自"應用程式註冊"在 Azure Active Directory 中擁有對Azure NetApp Files服務的足夠權限。應用程式註冊應使用以下任一方式：
  - 所有者或貢獻者角色"Azure 預定義"。
  - 一個"自訂貢獻者角色"訂閱等級(`assignableScopes`）但權限僅限於Trident所需的權限。建立自訂角色後，"使用 Azure 入口網站指派角色" 。

自訂貢獻者角色

```
{
  "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/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

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

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
```

```
ions/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": []
      }
    ]
  }
}
```

- 蔚藍 `location`包含至少一個 "委託子網"。截至Trident 22.01版本， `location`參數是後端設定檔頂層的一個必填欄位。虛擬池中指定的位置值將被忽略。
- 使用 `Cloud Identity`得到 `client ID`從一個 "使用者指派的託管身份"並指定該 `ID azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx` 。

中小企業卷的附加要求

若要建立 SMB 卷，您必須具備以下條件：

- Active Directory 已設定並連線至Azure NetApp Files。參考"Microsoft：建立和管理Azure NetApp Files管理器的 Active Directory 連接"。
- 一個 Kubernetes 叢集，包含一個 Linux 控制器節點和至少一個執行 Windows Server 2022 的 Windows 工作節點。 Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。
- 至少需要一個包含 Active Directory 憑證的Trident金鑰，以便Azure NetApp Files可以向 Active Directory 進行驗證。生成秘密 `smbcreds`：

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

- 配置為 Windows 服務的 CSI 代理程式。要配置 `csi-proxy`，請參閱"GitHub：CSI代理"或者"GitHub：適用於 Windows 的 CSI 代理"適用於在 Windows 上執行的 Kubernetes 節點。

Azure NetApp Files後端設定選項和範例

了解Azure NetApp Files的 NFS 和 SMB 後端設定選項，並查看設定範例。

Trident使用您的後端設定（子網路、虛擬網路、服務等級和位置），在要求的位置中可用的容量池上建立Azure NetApp Files卷，並與要求的服務等級和子網路相符。

* 從NetApp Trident 25.06 版本開始，手動 QoS 容量池作為技術預覽版受到支援。 *

Azure NetApp Files後端提供以下設定選項。

範圍	描述	預設
`version`		始終為 1
`storageDriverName`	儲存驅動程式的名稱	"azure-netapp-files"
`backendName`	自訂名稱或儲存後端	司機姓名 + "_" + 隨機字符
`subscriptionID`	Azure 訂閱的訂閱 ID（在 AKS 叢集上啟用託管識別時為可選）。	
`tenantID`	在 AKS 叢集上使用託管身分或雲端身分時，應用程式註冊中的租用戶 ID 是可選的。	
`clientID`	在 AKS 叢集上使用託管身分或雲端身分時，應用程式註冊中的用戶端 ID 是可選的。	
`clientSecret`	在 AKS 叢集上使用託管身分或雲端身分時，應用程式註冊中的用戶端金鑰是可選的。	
`serviceLevel`	之一 `Standard`， `Premium` ，或者 `Ultra`	「「（隨機的）
`location`	將在 Azure 中建立新磁碟區的位置名稱（在 AKS 叢集上啟用託管識別碼時為可選）。	
`resourceGroups`	用於篩選已發現資源的資源群組列表	（無濾鏡）
`netappAccounts`	用於篩選已發現資源的NetApp帳戶列表	（無濾鏡）
`capacityPools`	用於篩選已發現資源的容量池列表	（無過濾，隨機）
`virtualNetwork`	具有委派子網路的虛擬網路的名稱	“”
`subnet`	委派給的子網路名稱 `Microsoft.Netapp/volumes`	“”
`networkFeatures`	卷的 VNet 功能集，可能是 `Basic`或者 `Standard`。網路功能並非在所有地區都可用，可能需要透過訂閱才能啟用。指定`networkFeatures`未啟用該功能會導致磁碟區配置失敗。	“”

範圍	描述	預設
`nfsMountOptions`	對 NFS 掛載選項進行精細控制。對於 SMB 卷，此設定將被忽略。若要使用 NFS 版本 4.1 掛載卷，請包含以下內容 `nfsvers=4`在以逗號分隔的掛載選項清單中選擇 NFS v4.1。儲存類別定義中設定的掛載選項會覆蓋後端配置中設定的掛載選項。	"nfsvers=3"
`limitVolumeSize`	如果請求的磁碟區大小大於此值，則配置失敗。	（預設不強制執行）
`debugTraceFlags`	故障排除時要使用的調試標誌。例子， `\{"api": false, "method": true, "discovery": true}` 。除非您正在進行故障排除並需要詳細的日誌轉儲，否則請勿使用此功能。	無效的
`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。	`nfs`
`supportedTopologies`	表示從後端支援的區域和區域列表。更多信息，請參閱"使用 CSI 拓撲"。	
`qosType`	表示 QoS 類型：自動或手動。* Trident 25.06 技術預覽版*	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。* Trident 25.06 技術預覽版*	`4 MiB/sec`

有關網絡功能的更多信息，請參閱"為Azure NetApp Files磁碟區設定網路功能"。

所需權限和資源

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沒有關聯的必要權限和資源（子網路、虛擬網路、容量池）。如果啟用偵錯功能， Trident將記錄在建立後端時發現的 Azure 資源。請確認是否使用了適當的角色。

值 `resourceGroups`， `netappAccounts` ， `capacityPools` ， `virtualNetwork` ， 和 `subnet`可以使用簡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。

這 `resourceGroups`， `netappAccounts` ， 和 `capacityPools`值是過濾器，用於將發現的資源集限制為該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
資源組	<資源組>
NetApp帳戶	<資源組>/<NetApp 帳號>
容量池	<資源組>/<NetApp 帳號>/<容量池>

類型	格式
虛擬網路	<資源組>/<虛擬網路>
子網	<資源群組>/<虛擬網路>/<子網路>

卷配置

您可以透過在設定檔的特定部分中指定以下選項來控制預設磁碟區配置。參考 [範例配置] 了解詳情。

範圍	描述	預設
`exportRule`	新卷的出口規則。 `exportRule`必須是以逗號分隔的 IPv4 位址或 IPv4 子網路的任意組合列表，採用 CIDR 表示法。對於 SMB 卷，此設定將被忽略。	“0.0.0.0/0”
`snapshotDir`	控制 .snapshot 目錄的可見性	NFSv4 為“true”，NFSv3 為“false”。
`size`	新磁碟區的預設大小	100G
`unixPermissions`	新磁碟區的 Unix 權限（4 位八進位數字）。對於 SMB 卷，此設定將被忽略。	（預覽功能，需訂閱並加入白名單）

範例配置

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。

最小配置

這是最基本的後端配置。透過此配置， Trident會發現配置位置中所有委派給Azure NetApp Files儲存的NetApp帳戶、容量池和子網，並將新磁碟區隨機放置在其中一個池和子網路上。因為 `nasType`省略了 `nfs`預設設定生效，後端將為 NFS 磁碟區進行設定。

如果您剛開始使用Azure NetApp Files並進行嘗試，此配置是理想的選擇，但在實踐中，您需要為預配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
```

為 **AKS** 管理身份

此後端配置省略了 `subscriptionID`， `tenantID` ， `clientID` ， 和 `clientSecret`在使用託管身分時，這些是可選的。

```
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
```

AKS 的雲端身份

此後端配置省略了 `tenantID`， `clientID` ， 和 `clientSecret`在使用雲端身分時，這些是可選的。

```
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
```

具有容量池過濾器的特定服務等級配置

此後端配置將磁碟區放置在 Azure 中 `eastus`位置 `Ultra`容量池。 Trident會自動發現該位置中委派給Azure NetApp Files的所有子網，並隨機在其中一個子網路上放置一個新磁碟區。

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

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。* NetApp Trident 25.06 中的技術預覽*。

```
---
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
```

進階配置

此後端配置進一步縮小了磁碟區放置範圍到單一子網，並且還修改了一些磁碟區配置預設值。

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

虛擬池配置

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，並且想要在 Kubernetes 中建立代表這些級別的儲存類別時，這將非常有用。虛擬池標籤用於根據以下因素區分池子：`performance` 。

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

支援的拓撲配置

Trident可根據區域和可用區為工作負載提供磁碟區。這 `supportedTopologies`此後端配置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與每個 Kubernetes 叢集節點上的標籤中的區域和區域值相符。這些區域和分區代表儲存類別中可以提供的允許值的清單。對於包含後端提供的區域和可用區子集的儲存類， Trident會在所述區域和可用區中建立磁碟區。更多信息，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

下列 `StorageClass`上述定義指的是儲存池。

使用範例定義 **`parameter.selector`**場地

使用 `parameter.selector`你可以為每個物件指定。 `StorageClass`用於託管磁碟區的虛擬池。該磁碟區將具有所選池中定義的方面。

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

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

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

SMB 磁碟區的範例定義

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。

預設命名空間上的基本配置

```
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密鑰

```
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密鑰

```
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`或者 `nasType: null`NFS池過濾器。

創建後端

建立後端設定檔後，執行以下命令：

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

如果後端建立失敗，則後端配置存在問題。您可以透過執行以下命令查看日誌以確定原因：

```
tridentctl logs
```

在您發現並修正設定檔中的問題後，您可以再次執行建立命令。

Google Cloud NetApp Volumes

設定Google Cloud NetApp Volumes後端

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

Google Cloud NetApp Volumes驅動程式詳情

Trident提供 `google-cloud-netapp-volumes`驅動程式與集群通訊。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

司機	協定	音量模式	支援的存取模式	支援的檔案系統
`google-cloud-netapp-volumes`	NFS SMB	檔案系統	RWO、ROX、RWX、RWOP	`nfs， smb`

GKE 的雲端身份

雲端身分使 Kubernetes pod 能夠透過作為工作負載身分進行驗證來存取 Google Cloud 資源，而無需提供明確的 Google Cloud 憑證。

若要在 Google Cloud 中利用雲端身分功能，您必須具備以下條件：

- 使用 GKE 部署的 Kubernetes 叢集。
- 在 GKE 叢集上配置工作負載標識，並在節點池上配置 GKE 元資料伺服器。
- 具有Google Cloud NetApp Volumes管理員 (roles/netapp.admin) 角色或自訂角色的 GCP 服務帳戶。
- Trident已安裝，其中包括 cloudProvider（指定「GCP」）和 cloudIdentity（指定新的 GCP 服務帳戶）。下面給出一個例子。

Trident操作員

若要使用Trident操作員安裝Trident ，請編輯 `tridentorchestrator_cr.yaml`設定 `cloudProvider`到 `"GCP"`並設定 `cloudIdentity`到 `iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com`。

例如：

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

舵

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

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

以下範例安裝Trident並進行設置 `cloudProvider`使用環境變數連接到 `GCP` `$CP`並設定 `cloudIdentity`使用環境變數 `$ANNOTATION`：

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

<code>tridentctl</code>

請使用以下環境變數設定 雲端提供者 和 雲端身分 標誌的值：

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

以下範例安裝Trident並進行設置 `cloud-provider`標記 `$CP`，和 `cloud-identity`到 `$ANNOTATION`：

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

準備設定**Google Cloud NetApp Volumes**後端

在設定Google Cloud NetApp Volumes後端之前，您需要確保符合下列要求。

NFS 磁碟區的先決條件

如果您是第一次使用Google Cloud NetApp Volumes或在新的位置使用，則需要進行一些初始設定來設定Google Cloud NetApp Volumes並建立 NFS 磁碟區。參考"開始之前"。

在設定Google Cloud NetApp Volumes後端之前，請確保您已具備以下條件：

- 已設定Google Cloud NetApp Volumes服務的 Google Cloud 帳戶。參考"Google Cloud NetApp Volumes"。
- 您的 Google Cloud 帳戶項目編號。參考"確定項目"。
- 擁有NetApp Volumes 管理員權限的 Google Cloud 服務帳戶(`roles/netapp.admin`） 角色。參考"身分和存取管理角色和權限"。
- GCNV帳戶的API金鑰檔案。請參閱"建立服務帳戶金鑰"
- 儲水池。參考"儲存池概覽"。

有關如何設定對Google Cloud NetApp Volumes 的存取權限的更多信息，請參閱："設定對Google Cloud NetApp Volumes 的存取權限" 。

Google Cloud NetApp Volumes後端設定選項和範例

了解Google Cloud NetApp Volumes的後端設定選項並查看設定範例。

後端配置選項

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

範圍	描述	預設
`version`		始終為 1
`storageDriverName`	儲存驅動程式的名稱	價值 `storageDriverName` 必須指定為“google-cloud-netapp-volumes”。
`backendName`	（可選）儲存後端自訂名稱	驅動程式名稱 + "_" + API 金鑰的一部分
`storagePools`	用於指定磁碟區建立儲存池的可選參數。	
`projectNumber`	Google Cloud 帳戶項目編號。該值可在 Google Cloud 入口網站首頁找到。	
`location`	Trident建立 GCNV 磁碟區的 Google Cloud 位置。建立跨區域 Kubernetes 叢集時，在下列位置建立的磁碟區： `location`可用於跨多個 Google Cloud 區域的節點上調度的工作負載。跨區域運輸會產生額外費用。	

範圍	描述	預設
apiKey	用於 Google Cloud 服務帳戶的 API 金鑰 netapp.admin`角色。它包含 Google Cloud 服務帳戶私鑰檔案的 JSON 格式內容（原封不動地複製到後端設定檔中）。這 `apiKey`必須包含以下鍵的鍵值對： `type ， project_id ， client_email ， client_id ， auth_uri ， token_uri ， auth_provider_x509_cert_url ，和 client_x509_cert_url。	
nfsMountOptions	對 NFS 掛載選項進行精細控制。	"nfsvers=3"
limitVolumeSize	如果請求的磁碟區大小大於此值，則配置失敗。	（預設不強制執行）
serviceLevel	儲存池的服務等級及其容量。這些值是 flex，standard ， premium ， 或者 extreme。	
labels	若要套用於磁碟區的任意 JSON 格式標籤集	“”
network	Google Cloud 網路用於 GCNV 磁碟區。	
debugTraceFlags	故障排除時要使用的調試標誌。例子，{"api":false, "method":true} 。除非您正在進行故障排除並需要詳細的日誌轉儲，否則請勿使用此功能。	無效的
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。	nfs
supportedTopologies	表示從後端支援的區域和區域列表。更多信息，請參閱"使用 CSI 拓撲"。例如：supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a	

卷配置選項

您可以控制預設卷配置 `defaults`設定檔部分。

範圍	描述	預設
exportRule	新卷的出口規則。必須是以逗號分隔的 IPv4 位址列表，位址可以任意組合。	“0.0.0.0/0”
snapshotDir	訪問 `.snapshot`目錄	NFSv4 為“true”，NFSv3 為“false”。
snapshotReserve	快照預留的磁碟區百分比	（接受預設值 0）
unixPermissions	新磁碟區的 Unix 權限（4 位八進位數字）。	“”

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。

最小配置

這是最基本的後端配置。透過此配置， Trident會發現您已委派給配置位置中Google Cloud NetApp Volumes的所有儲存池，並隨機將新磁碟區放置在其中一個儲存池上。因為 `nasType`省略了 `nfs`預設設定生效，後端將為 NFS 磁碟區進行設定。

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

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    XsYg6gyxy4zq7OlwWgLwGa==\n
    -----END PRIVATE KEY-----\n

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

SMB磁碟區配置

```
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
```

使用儲存池過濾器進行配置

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  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
```

虛擬池配置

此後端配置在一個檔案中定義了多個虛擬池。虛擬池在以下位置定義： `storage`部分。當您有多個支援不同服務等級的儲存池，並且想要在 Kubernetes 中建立代表這些儲存層級的儲存類別時，它們非常有用。虛擬池標籤用於區分不同的池子。例如，在下面的例子中 `performance`標籤和 `serviceLevel`類型用於區分虛擬池。

您也可以設定一些適用於所有虛擬池的預設值，並覆寫各個虛擬池的預設值。在下面的例子中，`snapshotReserve`和 `exportRule`作為所有虛擬池的預設值。

更多信息，請參閱"虛擬池"。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
```

```
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

GKE 的雲端身份

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

支援的拓撲配置

Trident可根據區域和可用區為工作負載提供磁碟區。這 `supportedTopologies`此後端配置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與每個 Kubernetes 叢集節點上的標籤中的區域和區域值相符。這些區域和分區代表儲存類別中可以提供的允許值的清單。對於包含後端提供的區域和可用區子集的儲存類， Trident會在所述區域和可用區中建立磁碟區。更多信息，請參閱"使用 CSI 拓撲"。

```
---
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
```

下一步是什麼？

建立後端設定檔後，執行以下命令：

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

若要驗證後端是否已建立成功，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，則後端配置存在問題。您可以使用以下方式描述後端： `kubectl get tridentbackendconfig <backend-name>`執行以下命令查看日誌以確定原因：

```
tridentctl logs
```

在您發現並修正設定檔中的問題後，您可以刪除後端並再次執行建立命令。

儲存類別定義

以下是一個基本內容 `StorageClass`上述後端所指的定義。

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

使用以下範例定義 **`parameter.selector`**場地：

使用 `parameter.selector`你可以為每個物件指定。 `StorageClass`這"虛擬池"用於託管卷。該磁碟區將具有所選池中定義的方面。

```
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
```

有關存儲類別的更多詳細信息，請參閱"建立儲存類別"。

SMB 磁碟區的範例定義

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。任何 Active Directory 使用者/密碼，無論擁有任何權限或沒有權限，都可以用作節點階段金鑰。

預設命名空間上的基本配置

```
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密鑰

```
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密鑰

```
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`篩選支援 SMB 磁碟區的儲存池。 `nasType: nfs`或者 `nasType: null`NFS池過濾器。

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 是否已綁定，請執行以下命令：

```
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
```

為 Google Cloud 後端設定Cloud Volumes Service

了解如何使用提供的範例配置，將NetApp Cloud Volumes Service for Google Cloud 配置為Trident安裝的後端。

Google Cloud 驅動程式詳情

Trident提供 `gcp-cvs`驅動程式與集群通訊。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

司機	協定	音量模式	支援的存取模式	支援的檔案系統
`gcp-cvs`	NFS	檔案系統	RWO、ROX、RWX、RWOP	`nfs`

了解Trident對 Google Cloud Cloud Volumes Service 的支持

Trident可以在以下兩種格式之一建立Cloud Volumes Service磁碟區："服務類型" ：

- **CVS-Performance**： Trident 的預設服務類型。這種效能優化型服務類型最適合重視效能的生產工作負載。CVS-Performance 服務類型是一種硬體選項，支援最小 100 GiB 大小的磁碟區。您可以選擇其中之一"三個服務級別"：
  - `standard`
  - `premium`
  - `extreme`
- **CVS**：CVS 服務類型提供較高的區域可用性，但效能水準有限或中等。 CVS 服務類型是一種軟體選項，它使用儲存池來支援小至 1 GiB 的磁碟區。儲存池最多可包含 50 個磁碟區，所有磁碟區共用池的容量和效能。您可以選擇其中之一"兩種服務級別"：
  - `standardsw`
  - `zoneredundantstandardsw`

你需要什麼

配置和使用 "適用於 Google Cloud 的Cloud Volumes Service"後端需要以下組件：

- 已配置NetApp Cloud Volumes Service的 Google Cloud 帳戶
- 您的 Google Cloud 帳戶的項目編號
- 擁有 Google Cloud 服務帳戶 `netappcloudvolumes.admin`角色
- 您的Cloud Volumes Service帳戶的 API 金鑰文件

後端配置選項

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

範圍	描述	預設
`version`		始終為 1
`storageDriverName`	儲存驅動程式的名稱	"gcp-cvs"
`backendName`	自訂名稱或儲存後端	驅動程式名稱 + "_" + API 金鑰的一部分
`storageClass`	用於指定 CVS 服務類型的可選參數。使用 `software`選擇 `CVS` 服務類型。否則， `Trident` 會假定為 `CVS-Performance` 服務類型(`hardware`) 。	
`storagePools`	僅限CVS服務類型。用於指定磁碟區建立儲存池的可選參數。	
`projectNumber`	Google Cloud 帳戶項目編號。該值可在 Google Cloud 入口網站首頁找到。	
`hostProjectNumber`	如果使用共用 VPC 網絡，則必須執行此操作。在這種情況下， `projectNumber`這是一個服務項目，而且 `hostProjectNumber`是宿主項目。	

範圍	描述	預設
`apiRegion`	Trident建立Cloud Volumes Service區的 Google Cloud 區域。建立跨區域 Kubernetes 叢集時，在下列位置建立的磁碟區： `apiRegion`可用於跨多個 Google Cloud 區域的節點上調度的工作負載。跨區域運輸會產生額外費用。	
`apiKey`	用於 Google Cloud 服務帳戶的 API 金鑰 `netappcloudvolumes.admin`角色。它包含 Google Cloud 服務帳戶私鑰檔案的 JSON 格式內容（原封不動地複製到後端設定檔中）。	
`proxyURL`	如果需要代理伺服器才能連接到 CVS 帳戶，請提供代理 URL。代理伺服器可以是HTTP代理，也可以是HTTPS代理。對於 HTTPS 代理，會跳過憑證驗證，以允許在代理伺服器中使用自簽名憑證。不支援啟用身份驗證的代理伺服器。	
`nfsMountOptions`	對 NFS 掛載選項進行精細控制。	"nfsvers=3"
`limitVolumeSize`	如果請求的磁碟區大小大於此值，則配置失敗。	（預設不強制執行）
`serviceLevel`	CVS-Performance 或 CVS 服務等級（適用於新磁碟區）。 CVS-Performance 值是 `standard`， `premium` ， 或者 `extreme`。 CVS 值是 `standardsw`或者 `zoneredundantstandardsw`。	CVS-Performance 預設值為「標準」。 CVS 預設值為“standardsw”。
`network`	Google Cloud 網路用於Cloud Volumes Service磁碟區。	"預設"
`debugTraceFlags`	故障排除時要使用的調試標誌。例子， `\{"api":false, "method":true}` 。除非您正在進行故障排除並需要詳細的日誌轉儲，否則請勿使用此功能。	無效的
`allowedTopologies`	若要啟用跨區域訪問，您的 StorageClass 定義如下： `allowedTopologies`必須包含所有地區。例如： `- key: topology.kubernetes.io/region values: - us-east1 - europe-west1`	

卷配置選項

您可以控制預設卷配置 `defaults`設定檔部分。

範圍	描述	預設
`exportRule`	新卷的出口規則。必須是以逗號分隔的 IPv4 位址或 IPv4 子網路的列表，採用 CIDR 表示法。	“0.0.0.0/0”
`snapshotDir`	訪問 `.snapshot`目錄	“錯誤的”
`snapshotReserve`	快照預留的磁碟區百分比	（接受 CVS 預設值 0）

範圍	描述	預設
`size`	新卷的規模。 CVS-Performance 最低要求為 100 GiB。 CVS 最小容量為 1 GiB。	CVS-Performance 服務類型預設為「100GiB」。 CVS 服務類型不設定預設值，但要求至少 1 GiB。

CVS-Performance 服務類型範例

以下範例提供了 CVS-Performance 服務類型的範例設定。

範例 **1**：最小配置

這是使用預設 CVS-Performance 服務類型和預設「標準」服務等級的最小後端配置。

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

範例 2：服務等級配置

此範例展示了後端配置選項，包括服務等級和磁碟區預設值。

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

範例 3：虛擬池配置

此範例使用 `storage`配置虛擬池和 `StorageClasses`指的是他們。請參閱[儲存類別定義]查看儲存類別的定義方式。

這裡為所有虛擬池設定了特定的預設值，這些預設值決定了： `snapshotReserve` 5%和 `exportRule` 至 `0.0.0.0/0`。虛擬池在以下位置定義： `storage`部分。每個虛擬池都定義了自己的規則。 `serviceLevel`並且有些池會覆蓋預設值。虛擬池標籤用於根據以下因素區分池子： `performance` 和 `protection`。

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

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

儲存類別定義

以下 StorageClass 定義適用於虛擬池設定範例。使用 `parameters.selector`您可以為每個 StorageClass 指定用於託管磁碟區的虛擬池。該磁碟區將具有所選池中定義的方面。

儲存類別範例

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

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

- 第一個儲存類(`cvs-extreme-extra-protection`) 映射到第一個虛擬池。這是唯一提供極致效能且快照儲備為 10% 的儲存池。
- 最後一個儲存類別(`cvs-extra-protection`) 呼叫任何提供 10% 快照保留的儲存池。 Trident決定選擇哪個虛擬池，並確保滿足快照儲備要求。

CVS 服務類型範例

以下範例提供了 CVS 服務類型的範例配置。

範例 **1**：最小配置

這是使用最簡後端配置 `storageClass`指定 CVS 服務類型和預設值 `standardsw`服務水平。

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

範例 **2**：儲存池配置

此範例後端配置使用 `storagePools`配置儲存池。

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

下一步是什麼？

建立後端設定檔後，執行以下命令：

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

如果後端建立失敗，則後端配置存在問題。您可以透過執行以下命令查看日誌以確定原因：

```
tridentctl logs
```

在您發現並修正設定檔中的問題後，您可以再次執行建立命令。

配置NetApp HCI或SolidFire後端

了解如何在Trident安裝中建立和使用 Element 後端。

元素驅動程式詳情

Trident提供 `solidfire-san`用於與叢集通訊的儲存驅動程式。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

這 `solidfire-san`儲存驅動程式支援_檔案_和_區塊_磁碟區模式。對於 `Filesystem`volumeMode， Trident建立一個磁碟區並建立一個檔案系統。檔案系統類型由 StorageClass 指定。

司機	協定	音量模式	支援的存取模式	支援的檔案系統
`solidfire-san`	iSCSI	堵塞	RWO、ROX、RWX、RWOP	沒有檔案系統。原始塊設備。
`solidfire-san`	iSCSI	檔案系統	RWO，RWOP	`xfs`， `ext3` ， `ext4`

開始之前

在建立 Element 後端之前，您需要以下內容。

- 一個支援運行 Element 軟體的儲存系統。
- 擁有NetApp HCI/ SolidFire叢集管理員或租用戶用戶權限，可管理磁碟區。
- 所有 Kubernetes 工作節點都應該安裝對應的 iSCSI 工具。參考"工作節點準備訊息"。

後端配置選項

請參閱下表以了解後端配置選項：

範圍	描述	預設
`version`		始終為 1
`storageDriverName`	儲存驅動程式的名稱	始終是“solidfire-san”
`backendName`	自訂名稱或儲存後端	"solidfire_" + 儲存體 (iSCSI) IP 位址
`Endpoint`	針對SolidFire叢集的 MVIP，包含租戶憑證	
`SVIP`	儲存（iSCSI）IP 位址和連接埠	

範圍	描述	預設
`labels`	若要套用於磁碟區的任意 JSON 格式標籤集。	“”
`TenantName`	要使用的租用戶名稱（如果找不到則建立）	
`InitiatorIFace`	將 iSCSI 流量限製到特定主機介面	"預設"
`UseCHAP`	使用 CHAP 對 iSCSI 進行身份驗證。 Trident使用 CHAP。	真的
`AccessGroups`	要使用的存取群組 ID 列表	尋找名為「trident」的訪問群組的 ID
`Types`	QoS規範	
`limitVolumeSize`	如果請求的磁碟區大小超過此值，則配置失敗。	（預設不強制執行）
`debugTraceFlags`	故障排除時要使用的調試標誌。例如，{"api":false, "method":true}	無效的

請勿使用 `debugTraceFlags`除非您正在進行故障排除並且需要詳細的日誌轉儲。

範例 1：後端配置 `solidfire-san`具有三種音量類型的驅動器

本範例展示了一個使用 CHAP 認證的後端文件，並對三種具有特定 QoS 保證的捲類型進行了建模。最有可能的情況是，您會定義儲存類別來使用它們。 `IOPS`儲存類別參數。

```
---
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：後端和儲存類別配置 **`solidfire-san`**帶有虛擬池的驅動程式

此範例顯示了配置了虛擬池的後端定義檔以及引用這些虛擬池的儲存類別。

Trident在設定時將儲存池中的標籤複製到後端儲存 LUN。為了方便起見，儲存管理員可以為每個虛擬池定義標籤，並按標籤將磁碟區分組。

在下方所示的範例後端定義檔中，所有儲存池都設定了特定的預設值，這些預設值決定了： `type`銀級。虛擬池在以下位置定義： `storage`部分。在這個例子中，一些儲存池設定了自己的類型，而一些儲存池則覆蓋了上面設定的預設值。

```
---
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
```

以下 StorageClass 定義與上述虛擬池相關。使用 `parameters.selector`在欄位中，每個 StorageClass 都會指定哪些虛擬池可用於託管磁碟區。卷將具有所選虛擬池中定義的各個方面。

第一個儲存類(`solidfire-gold-four`將映射到第一個虛擬池。這是唯一提供黃金級性能的泳池。

`Volume Type QoS`黃金。最後一個儲存類別（`solidfire-silver`）指出任何提供銀級性能的儲存池。Trident將決定選擇哪個虛擬池，並確保滿足儲存需求。

```
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
```

查找更多信息

- "卷訪問群組"

ONTAP SAN 驅動程式

ONTAP SAN 驅動程式概述

了解如何使用ONTAP和Cloud Volumes ONTAP SAN 驅動程式設定ONTAP後端。

ONTAP SAN 驅動程式詳情

Trident提供以下 SAN 儲存驅動程序，用於與ONTAP叢集通訊。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

司機	協定	音量模式	支援的存取模式	支援的檔案系統
`ontap-san`	iSCSI 透過光纖通道提供 SCSI 服務	堵塞	RWO、ROX、RWX、RWOP	無檔案系統；原始區塊設備
`ontap-san`	iSCSI 透過光纖通道提供 SCSI 服務	檔案系統	RWO，RWOP ROX 和 RWX 在檔案系統磁碟區模式下不可用。	`xfs`，`ext3`，`ext4`
`ontap-san`	NVMe/TCP 參考NVMe/TCP 的其他注意事項。	堵塞	RWO、ROX、RWX、RWOP	無檔案系統；原始區塊設備
`ontap-san`	NVMe/TCP 參考NVMe/TCP 的其他注意事項。	檔案系統	RWO，RWOP ROX 和 RWX 在檔案系統磁碟區模式下不可用。	`xfs`，`ext3`，`ext4`

司機	協定	音量模式	支援的存取模式	支援的檔案系統
`ontap-san-economy`	iSCSI	堵塞	RWO、ROX、RWX、RWOP	無檔案系統；原始區塊設備
`ontap-san-economy`	iSCSI	檔案系統	RWO，RWOP ROX 和 RWX 在檔案系統磁碟區模式下不可用。	`xfs`，`ext3`，`ext4`

- 使用 `ontap-san-economy`僅當預計持續卷使用量高於...時"支援的ONTAP容量限制"。
- 使用 `ontap-nas-economy`僅當預計持續卷使用量高於...時"支援的ONTAP容量限制"以及 `ontap-san-economy`驅動程式無法使用。
- 請勿使用 `ontap-nas-economy`如果您預計需要資料保護、災難復原或行動辦公室。
- NetApp不建議在所有ONTAP驅動程式中使用 Flexvol 自動成長，ontap-san 除外。作為變通方法， Trident支援使用快照儲備，並相應地調整 Flexvol 容量。

使用者權限

Trident預期以ONTAP或 SVM 管理員身分執行，通常使用下列方式： `admin`集群用戶或 `vsadmin`SVM 用戶，或具有相同角色但名稱不同的用戶。對於Amazon FSx for NetApp ONTAP部署， Trident需要以ONTAP或 SVM 管理員身分執行，並使用叢集。 `fsxadmin`用戶或 `vsadmin`SVM 用戶，或具有相同角色但名稱不同的用戶。這 `fsxadmin`用戶是集群管理員用戶的有限替代品。

如果你使用 `limitAggregateUsage`需要參數和叢集管理員權限。當使用Amazon FSx for NetApp ONTAP和Trident時， `limitAggregateUsage`參數將無法與 `vsadmin`和 `fsxadmin`用戶帳戶。如果指定此參數，配置操作將失敗。

雖然可以在ONTAP中建立一個限制性更強的角色供Trident驅動程式使用，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，這些 API 必須加以考慮，這使得升級變得困難且容易出錯。

NVMe/TCP 的其他注意事項

Trident支援使用非揮發性記憶體高速介面 (NVMe) 協定 `ontap-san`驅動程式包括：

- IPv6
- NVMe磁碟區的快照和克隆
- 調整 NVMe 卷的大小
- 導入在Trident外部建立的 NVMe 卷，以便Trident可以管理其生命週期。
- NVMe原生多路徑
- K8s節點的優雅關閉或非優雅關閉（24.06）

Trident不支援：

- NVMe 原生支援的 DH-HMAC-CHAP
- 設備映射器 (DM) 多路徑

- LUKS 加密

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

準備配置後端**ONTAP SAN** 驅動程式

了解配置ONTAP後端和ONTAP SAN 驅動程式的要求和驗證選項。

要求

對於所有ONTAP後端， Trident要求至少將一個聚合分配給 SVM。

"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，您還可以運行多個驅動程序，並建立指向其中一個或另一個驅動程式的儲存類別。例如，您可以設定一個 `san-dev`使用類別 `ontap-san`司機和 `san-default`使用類別 `ontap-san-economy`一。

所有 Kubernetes 工作節點都必須安裝對應的 iSCSI 工具。參考 "準備工作節點" 了解詳情。

對**ONTAP**後端進行身份驗證

Trident提供兩種ONTAP後端身分驗證方式。

- 基於憑證：具有所需權限的ONTAP使用者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`或者 `vsadmin`確保與ONTAP版本最大程度相容。
- 基於憑證： Trident也可以使用安裝在後端的憑證與ONTAP叢集通訊。此處，後端定義必須包含用戶端憑證、金鑰和受信任 CA 憑證（如果使用，建議使用）的 Base64 編碼值。

您可以更新現有後端，以在基於憑證的方法和基於憑證的方法之間進行切換。但是，一次只能支援一種身份驗證方法。要切換到不同的身份驗證方法，必須從後端配置中刪除現有方法。

如果您嘗試同時提供憑證和證書，則後端建立將會失敗，並出現錯誤，提示設定檔中提供了多個驗證方法。

啟用基於憑證的身份驗證

Trident需要 SVM 範圍/叢集範圍管理員的憑證才能與ONTAP後端通訊。建議使用標準的、預先定義的角色，例如： `admin`或者 `vsadmin`。這樣可以確保與未來ONTAP版本向前相容，這些版本可能會公開一些功能 API，供未來的Trident版本使用。雖然可以建立自訂安全登入角色並將其與Trident一起使用，但不建議這樣做。

後端定義範例如下圖所示：

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"
}
```

請注意，後端定義是唯一以純文字形式儲存憑證的地方。後端建立完成後，使用者名稱/密碼將使用 Base64 進行編碼，並儲存為 Kubernetes 金鑰。只有在建立或更新後端時才需要了解憑證。因此，這是一項僅限管理員執行的操作，由 Kubernetes/儲存管理員執行。

啟用基於憑證的身份驗證

新的和現有的後端都可以使用憑證與ONTAP後端通訊。後端定義需要三個參數。

- clientCertificate：客戶端憑證的 Base64 編碼值。
- clientPrivateKey：關聯私鑰的 Base64 編碼值。
- trustedCACertificate：受信任 CA 憑證的 Base64 編碼值。如果使用受信任的 CA，則必須提供此參數。如果沒有使用受信任的憑證授權機構，則可以忽略此步驟。

典型的工作流程包括以下步驟。

步驟

1. 產生客戶端憑證和金鑰。產生時，將通用名稱 (CN) 設定為要進行驗證的ONTAP使用者。

```
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. 在ONTAP叢集中新增受信任的 CA 憑證。這可能已經由儲存管理員處理了。如果沒有使用受信任的憑證授權機構，則忽略此操作。

```
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叢集上安裝客戶端憑證和金鑰（來自步驟 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安全登入角色支持 `cert`身份驗證方法。

```
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. 使用產生的憑證進行身份驗證。請將 < ONTAP管理 LIF> 和 <vserver 名稱> 替換為管理 LIF IP 位址和 SVM 名稱。

```
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 對憑證、金鑰和受信任的 CA 憑證進行編碼。

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

7. 使用上一步獲得的值建立後端。

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

更新身份驗證方法或輪換憑證

您可以更新現有後端，以使用不同的身份驗證方法或輪換其憑證。這種方法是雙向的：使用使用者名稱/密碼的後端可以更新為使用憑證；使用憑證的後端可以更新為基於使用者名稱/密碼的後端。為此，您必須刪除現有的身份驗證方法並新增新的身份驗證方法。然後使用包含所需參數的更新後的 backend.json 檔案來執行 `tridentctl backend update`。

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

輪換密碼時，儲存管理員必須先更新ONTAP上使用者的密碼。接下來將進行後端更新。輪換證書時，可以為使用者新增多個證書。然後更新後端以使用新證書，之後即可從ONTAP叢集中刪除舊證書。

更新後端不會中斷對已建立磁碟區的訪問，也不會影響之後建立的磁碟區連線。後端更新成功表明Trident可以與ONTAP後端通訊並處理未來的磁碟區操作。

為**Trident**建立自訂**ONTAP**角色

您可以建立一個具有最低權限的ONTAP叢集角色，這樣您就不必使用ONTAP管理員角色在Trident中執行操作。在Trident後端設定中包含使用者名稱時， Trident將使用您建立的ONTAP叢集角色來執行操作。

請參閱"Trident自訂角色產生器"有關建立Trident自訂角色的詳細資訊。

使用**ONTAP CLI**

1. 使用以下命令建立新角色：

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

2. 為Trident用戶建立使用者名稱：

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

3. 將角色映射到使用者：

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

使用系統管理員

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，請選擇「叢集 > 設定」。

      （或）若要在 SVM 層級建立自訂角色，請選擇「儲存」>「儲存虛擬機器」> `required SVM` > 設定 > 使用者和角色*。

   b. 選擇“使用者和角色”旁邊的箭頭圖示（→）。
   c. 在“角色”下選擇“+添加”。
   d. 定義角色規則，然後點選「儲存」。
2. 將角色對應到Trident使用者： + 在「使用者和角色」頁面上執行下列步驟：
   a. 在「使用者」下方選擇「新增」圖示 **+**。
   b. 選擇所需的使用者名，然後在「角色」下拉式選單中選擇角色。
   c. 點選“儲存”。

更多資訊請參閱以下頁面：

- "用於管理ONTAP的自訂角色"或者"定義自訂角色"
- "與角色和使用者協作"

使用雙向 **CHAP** 驗證連接

Trident可以使用雙向 CHAP 對 iSCSI 會話進行驗證。 `ontap-san`和 `ontap-san-economy`司機。這需要啟用 `useCHAP`在後端定義中新增選項。設定為 `true`Trident將 SVM 的預設發起程序安全地配置為雙向 CHAP，並從後端檔案設定使用者名稱和金鑰。 NetApp建議使用雙向 CHAP 協定對連線進行身份驗證。請參閱以下範例配

置：

```
---
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
```

這 `useCHAP`此參數是一個布林選項，只能配置一次。預設值為 false。一旦將其設為 true，就無法再將其設為 false。

另外 `useCHAP=true`，這 `chapInitiatorSecret`， `chapTargetInitiatorSecret` ， `chapTargetUsername` ， 和 `chapUsername`欄位必須包含在後端定義中。創建後端後，可以透過執行以下命令來更改密鑰： `tridentctl update` 。

工作原理

透過設定 `useCHAP`如果設定為 true，則儲存管理員指示Trident在儲存裝置後端設定 CHAP。其中包括以下內容：

- 在SVM上設定CHAP：
  - 如果 SVM 的預設啟動器安全類型為「無」（預設）*且*磁碟區中不存在任何預先存在的 LUN，Trident會將預設安全類型設為「無」。 `CHAP`然後繼續配置 CHAP 發起程序和目標使用者名稱及金鑰。
  - 如果 SVM 包含 LUN， Trident將不會在 SVM 上啟用 CHAP。這樣可以確保對 SVM 上已存在的 LUN 的存取不受限制。
- 配置 CHAP 發起程序和目標使用者名稱和金鑰；這些選項必須在後端配置中指定（如上所示）。

後端建立完成後， Trident會建立一個對應的 `tridentbackend`CRD 並將 CHAP 金鑰和使用者名稱儲存為 Kubernetes 金鑰。 Trident在此後端建立的所有 PV 都將透過 CHAP 進行安裝和連線。

輪換憑證並更新後端

您可以透過更新 CHAP 參數來更新 CHAP 憑證。 `backend.json`文件。這將需要更新 CHAP 金鑰並使用 `tridentctl update`命令反映這些變更。

更新後端 CHAP 金鑰時，必須使用 `tridentctl`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憑證，因為Trident將無法偵測到這些變更。

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

現有連線將不受影響；如果Trident在 SVM 上更新憑證，則這些連線將繼續保持活動狀態。新連線使用更新後的憑證，現有連線將繼續保持活動狀態。斷開並重新連接舊的PV將使它們使用更新的憑證。

ONTAP SAN 配置選項和範例

了解如何在Trident安裝中建立和使用ONTAP SAN 驅動程式。本節提供後端設定範例以及將後端對應到 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。這些變化會影響某些參數的使用，如註釋中所述。"了解更多關於ASA r2 系統與其他ONTAP系統之間的差異"。

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

在Trident後端設定中，無需指定您的系統是ASA r2。當您選擇 `ontap-san`作為 `storageDriverName`Trident可自動偵測ASA r2 或傳統的ONTAP系統。如下表所示，某些後端設定參數不適用於ASA r2 系統。

請參閱下表以了解後端配置選項：

範圍	描述	預設
`version`		始終為 1
`storageDriverName`	儲存驅動程式的名稱	`ontap-san`或者 `ontap-san-economy`
`backendName`	自訂名稱或儲存後端	驅動程式名稱 + "_" + dataLIF
`managementLIF`	叢集或 SVM 管理 LIF 的 IP 位址。 可以指定一個完全限定網域名稱（FQDN）。 如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。 IPv6 位址必須用方括號定義，例如：`[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 。 為了實現MetroCluster 的無縫切換，請參閱MetroCluster範例。 ⓘ 如果您使用的是「vsadmin」憑證，`managementLIF`必須是SVM的憑證；如果使用「admin」憑證，`managementLIF`必須是集群的那個。	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	LIF協定的IP位址。如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。 IPv6 位址必須用方括號定義，例如：`[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 。 *請勿指定使用 iSCSI。 *Trident的使用"ONTAP選擇性 LUN 地圖"發現建立多路徑會話所需的 iSCSI LIF。如果出現以下情況，則會產生警告： `dataLIF`已明確定義。 *Metrocluster 除外。 *查看MetroCluster範例。	由支援向量機導出
`svm`	要使用的儲存虛擬機器 *Metrocluster 除外。 *查看MetroCluster範例。	如果是 SVM 則推導而來 `managementLIF`已指定
`useCHAP`	使用 CHAP 對ONTAP SAN 驅動程式的 iSCSI 進行驗證 [布林值]。設定為 `true`讓Trident配置並使用雙向 CHAP 作為後端給定 SVM 的預設身份驗證。參考 "準備配置後端ONTAP SAN 驅動程式" 了解詳情。 *不支援FCP或NVMe/TCP協定。 *	`false`
`chapInitiatorSecret`	CHAP 發起者金鑰。如果是必填項 `useCHAP=true`	“”
`labels`	若要套用於磁碟區的任意 JSON 格式標籤集	“”

範圍	描述	預設
`chapTargetInitiatorSecret`	CHAP 目標發起者金鑰。如果是必填項 `useCHAP=true`	“”
`chapUsername`	入站用戶名。如果是必填項 `useCHAP=true`	“”
`chapTargetUsername`	目標用戶名。如果是必填項 `useCHAP=true`	“”
`clientCertificate`	用戶端憑證的 Base64 編碼值。用於基於憑證的身份驗證	“”
`clientPrivateKey`	客戶端私鑰的 Base64 編碼值。用於基於憑證的身份驗證	“”
`trustedCACertificate`	受信任 CA 憑證的 Base64 編碼值。選修的。用於基於憑證的身份驗證。	“”
`username`	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"。	“”
`password`	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"。	“”
`svm`	使用的儲存虛擬機	如果是 SVM 則推導而來 `managementLIF`已指定
`storagePrefix`	在 SVM 中配置新磁碟區時所使用的前綴。之後無法修改。要更新此參數，您需要建立一個新的後端。	`trident`
`aggregate`	用於配置的聚合（可選；如果設置，則必須指派給 SVM）。對於 `ontap-nas-flexgroup`驅動程序，此選項將被忽略。如果未分配，則可以使用任何可用的聚合來配置FlexGroup磁碟區。 當 SVM 中的聚合資料更新時， Trident 會自動輪詢 SVM 進行更新，而無需重新啟動Trident控制器。當您在Trident中配置特定聚合以配置磁碟區時，如果該聚合被重新命名或移出 SVM，則在輪詢 SVM 聚合時， Trident中的後端將變為失敗狀態。您必須將聚合變更為 SVM 上存在的聚合，或將其完全刪除，才能使後端恢復連線。 請勿指定用於**ASA r2** 系統。	“”

範圍	描述	預設
`limitAggregateUsage`	如果使用率超過此百分比，則配置失敗。如果您使用的是Amazon FSx for NetApp ONTAP後端，請勿指定 `limitAggregateUsage`。提供的 `fsxadmin`和 `vsadmin`不包含檢索匯總使用情況和使用Trident限制它所需的權限。 請勿指定用於**ASA r2** 系統。	（預設不強制執行）
`limitVolumeSize`	如果請求的磁碟區大小大於此值，則配置失敗。同時限制其管理的 LUN 卷的最大大小。	（預設不強制執行）
`lunsPerFlexvol`	每個 Flexvol 的最大 LUN 數量必須在 [50, 200] 範圍內	`100`
`debugTraceFlags`	故障排除時要使用的調試標誌。例如，{"api":false, "method":true} 除非您正在進行故障排除並且需要詳細的日誌轉儲，否則請勿使用此方法。	`null`
`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。 `useREST`完全符合 NVMe/TCP 標準。 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`適用於ONTAP 9.15.1 或更高版本，否則 `false`。
`sanType`	用於選擇 `iscsi`對於 iSCSI， `nvme`適用於 NVMe/TCP 或 `fcp`用於光纖通道 (FC) 上的 SCSI。	`iscsi`如果為空

範圍	描述	預設
`formatOptions`	使用 `formatOptions`為以下情況指定命令列參數 `mkfs`該命令將在每次格式化磁碟區時套用。這樣您就可以根據自己的喜好格式化音量。請確保指定與 mkfs 指令選項類似的 formatOptions，但不包含裝置路徑。例如：“-E nodiscard” *支援 `ontap-san`和 `ontap-san-economy`支援 iSCSI 協定的驅動程式。 * *此外，在使用 iSCSI 和 NVMe/TCP 協定時， ASA r2 系統也受支援。 *	
`limitVolumePoolSize`	在 ontap-san-economy 後端使用 LUN 時可請求的最大FlexVol大小。	（預設不強制執行）
`denyNewVolumePools`	限制 `ontap-san-economy`後端建立新的FlexVol區來包含它們的 LUN。只有預先存在的 Flexvol 才能用於配置新的 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:

- 保留，不要在執行 mkfs 時嘗試丟棄區塊（最初丟棄區塊對固態設備和稀疏/精簡配置儲存很有用）。這取代了已棄用的選項“-K”，並且適用於所有檔案系統（xfs、ext3 和 ext4）。

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。

您可以使用下列選項控制預設配置。 `defaults`配置部分。例如，請參閱下面的設定範例。

範圍	描述	預設
`spaceAllocation`	LUN 的空間分配	“true” 如果指定，則設定為 **`true`**適用於**ASA r2** 系統。
`spaceReserve`	空間預留模式；「無」（細）或「大量」（粗）。 設定為 **`none`**適用於**ASA r2** 系統。	“沒有任何”
`snapshotPolicy`	要使用的快照策略。 設定為 **`none`**適用於**ASA r2** 系統。	“沒有任何”
`qosPolicy`	若要為建立的磁碟區指派的 QoS 策略群組。每個儲存池/後端選擇 qosPolicy 或 adaptiveQosPolicy 之一。將 QoS 策略群組與Trident結合使用需要ONTAP 9.8 或更高版本。您應該使用非共享的 QoS 策略群組，並確保該策略群組單獨套用至每個成員。共享的 QoS 策略群組強制規定所有工作負載的總吞吐量上限。	“”
`adaptiveQosPolicy`	若要為建立的磁碟區指派的自適應 QoS 策略群組。每個儲存池/後端選擇 qosPolicy 或 adaptiveQosPolicy 之一	“”
`snapshotReserve`	為快照預留的磁碟區百分比。 請勿指定用於**ASA r2** 系統。	如果為“0”， `snapshotPolicy`為“無”，否則為“
`splitOnClone`	創建時將克隆體從其母體中分離出來	“錯誤的”
`encryption`	在新磁碟區啟用NetApp磁碟區加密 (NVE)；預設為 `false`。若要使用此選項，必須在叢集上取得 NVE 許可並啟用 NVE。如果後端啟用了 NAE，則在Trident中配置的任何磁碟區都會啟用 NAE。更多信息，請參閱：“Trident如何與 NVE 和 NAE 協同工作” 。	“false” 如果指定，則設定為 **`true`**適用於**ASA r2** 系統。
`luksEncryption`	啟用LUKS加密。參考“使用 Linux 統一金鑰設定 (LUKS)”。	設定為 **`false`**適用於**ASA r2** 系統。
`tieringPolicy`	分層策略使用「無」 請勿為**ASA r2** 系統指定 。	
`nameTemplate`	用於建立自訂磁碟區名稱的範本。	“”

卷配置範例

以下是一個定義了預設值的範例：

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

對於使用以下方式建立的所有捲 `ontap-san`驅動程式Trident為FlexVol增加了 10% 的額外容量，以容納 LUN 元資料。 LUN 將依照使用者在 PVC 中要求的確切大小進行設定。 Trident使FlexVol增加 10%（在ONTAP中顯示為可用尺寸）。用戶現在將獲得他們所申請的可用容量。此項變更還可以防止 LUN 在可用空間未完全利用之前變為唯讀。這不適用於 ontap-san-economy。

對於定義後端 `snapshotReserve`Trident計算體積大小的方法如下：

```
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%。為了 `snapshotReserve`= 5%，PVC 請求 = 5 GiB，總體積大小為 5.79 GiB，可用大小為 5.5 GiB。這 `volume show`該命令應顯示與此範例類似的結果：

```
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
          _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4
                       online     RW         10GB     5.00GB    0%
          _pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d
                       online     RW       5.79GB     5.50GB    0%
          _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba
                       online     RW          1GB    511.8MB    0%
3 entries were displayed.
```

目前，調整大小是將新計算方法應用於現有體積的唯一方法。

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。

> 如果您在NetApp ONTAP上使用Amazon FSx和Trident， NetApp建議您為 LIF 指定 DNS 名稱而非 IP 位址。

ONTAP SAN 範例

這是使用以下方法的基本配置： `ontap-san`司機。

```
---
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，以避免在切換和切換回後端後手動更新後端定義。"SVM複製與復原" 。

為了實現無縫切換和切換回，請指定 SVM `managementLIF`並省略 `svm`參數。例如：

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

ONTAP SAN 經濟範例

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

基於憑證的身份驗證範例

在這個基本設定範例中 `clientCertificate`， `clientPrivateKey` ， 和 `trustedCACertificate`（如果使用受信任的 CA，則為可選） `backend.json`分別取客戶端憑證、私鑰和受信任 CA 憑證的 base64 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

這些範例創建了一個後端。 `useCHAP`設定為 `true`。

ONTAP SAN CHAP 範例

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

ONTAP SAN 經濟 CHAP 範例

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

NVMe/TCP 範例

您的ONTAP後端必須設定使用 NVMe 的 SVM。這是 NVMe/TCP 的基本後端配置。

```
---
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

您的ONTAP後端必須配置 FC 的 SVM。這是 FC 的基本後端配置。

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

使用 **nameTemplate** 的後端設定範例

```
---
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}}"
```

ontap-san-economy 驅動程式的 **formatOptions** 範例

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

具有虛擬池的後端範例

在這些範例後端定義檔中，所有儲存池都設定了特定的預設值，例如： `spaceReserve`沒有，`spaceAllocation`為假，並且 `encryption`錯誤。虛擬池在儲存部分中定義。

Trident在「備註」欄位中設定配置標籤。在配置時， Trident會將虛擬池上的所有標籤複製到儲存卷，並在FlexVol volume上設定註解。為了方便起見，儲存管理員可以為每個虛擬池定義標籤，並按標籤將磁碟區分組。

在這些範例中，一些儲存池會設定自己的參數。 `spaceReserve` ， `spaceAllocation` ， 和 `encryption` 有些值會覆蓋預設值，有些池會覆蓋預設值。

ONTAP SAN 範例

```yaml
---
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"
```

將後端映射到儲存類

以下 StorageClass 定義指的是[具有虛擬池的後端範例]。使用 `parameters.selector`在欄位中，每個 StorageClass 都會指出哪些虛擬池可用於託管磁碟區。卷將具有所選虛擬池中定義的各個方面。

- 這 `protection-gold`StorageClass 將會對應到第一個虛擬池。 `ontap-san`後端。這是唯一提供黃金級保護的泳池。

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

- 這 `protection-not-gold`StorageClass 將會對應到第二個和第三個虛擬池。 `ontap-san`後端。除了黃金等級之外，只有這些金池提供其他等級的保護。

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

- 這 `app-mysqldb`StorageClass 將會對應到第三個虛擬池 `ontap-san-economy`後端。這是唯一一個為mysqldb類型應用程式提供儲存池配置的儲存池。

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

- 這 `protection-silver-creditpoints-20k`StorageClass 將會對應到第二個虛擬池 `ontap-san`後端。這是唯一提供銀級保護和 20000 積分的獎金池。

```
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"
```

- 這 `creditpoints-5k`StorageClass 將會對應到第三個虛擬池 `ontap-san`後端與第四個虛擬池 `ontap-san-economy`後端。這是唯一提供 5000 點的彩池產品。

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

- 這 `my-test-app-sc`StorageClass 將會對應到 `testAPP`虛擬池 `ontap-san`司機 `sanType: nvme`。這是唯一一家提供泳池的公司 `testApp`。

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

Trident將決定選擇哪個虛擬池，並確保滿足儲存需求。

ONTAP NAS 驅動程式

ONTAP NAS 驅動程式概述

了解如何使用ONTAP和Cloud Volumes ONTAP NAS 驅動程式設定ONTAP後端。

ONTAP NAS 驅動程式詳情

Trident提供以下 NAS 儲存驅動程序，用於與ONTAP叢集通訊。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

司機	協定	音量模式	支援的存取模式	支援的檔案系統
`ontap-nas`	NFS SMB	檔案系統	RWO、ROX、RWX、RWOP	"， `nfs` ， `smb`
`ontap-nas-economy`	NFS SMB	檔案系統	RWO、ROX、RWX、RWOP	"， `nfs` ， `smb`
`ontap-nas-flexgroup`	NFS SMB	檔案系統	RWO、ROX、RWX、RWOP	"， `nfs` ， `smb`

- 使用 `ontap-san-economy`僅當預計持續卷使用量高於…時"支援的ONTAP容量限制"。
- 使用 `ontap-nas-economy`僅當預計持續卷使用量高於…時"支援的ONTAP容量限制"以及 `ontap-san-economy`驅動程式無法使用。
- 請勿使用 `ontap-nas-economy`如果您預計需要資料保護、災難復原或行動辦公室。
- NetApp不建議在所有ONTAP驅動程式中使用 Flexvol 自動成長，ontap-san 除外。作為變通方法， Trident支援使用快照儲備，並相應地調整 Flexvol 容量。

使用者權限

Trident預期以ONTAP或 SVM 管理員身分執行，通常使用下列方式： `admin`集群用戶或 `vsadmin`SVM 用戶，或具有相同角色但名稱不同的用戶。

對於Amazon FSx for NetApp ONTAP部署， Trident需要以ONTAP或 SVM 管理員身分執行，並使用叢集。`fsxadmin`用戶或 `vsadmin`SVM 用戶，或具有相同角色但名稱不同的用戶。這 `fsxadmin`用戶是集群管理員用戶的有限替代品。

如果你使用 `limitAggregateUsage`需要參數和叢集管理員權限。當使用Amazon FSx for NetApp ONTAP和Trident時， `limitAggregateUsage`參數將無法與 `vsadmin`和 `fsxadmin`用戶帳戶。如果指定此參數，配置操作將失敗。

雖然可以在ONTAP中建立一個限制性更強的角色供Trident驅動程式使用，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，這些 API 必須加以考慮，這使得升級變得困難且容易出錯。

準備配置帶有**ONTAP NAS** 驅動程式的後端

了解配置具有ONTAP NAS 驅動程式的ONTAP後端的要求、驗證選項和匯出策略。

要求

- 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序，並建立指向其中一個或另一個驅動程式的儲存類別。例如，您可以設定一個使用以下方式的 Gold 類別： `ontap-nas`駕駛員和使用青銅級的駕駛員 `ontap-nas-economy`一。

- 所有 Kubernetes 工作節點都必須安裝對應的 NFS 工具。請參閱"這裡"更多詳情請見下文。
- Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。參考 準備配置SMB卷 了解詳情。

對ONTAP後端進行身份驗證

Trident提供兩種ONTAP後端身分驗證方式。

- 基於憑證：此模式需要對ONTAP後端擁有足夠的權限。建議使用與預先定義的安全登入角色關聯的帳戶，例如： `admin`或者 `vsadmin`確保與ONTAP版本最大程度相容。
- 基於證書：此模式要求後端安裝證書，以便Trident才能與ONTAP叢集通訊。此處，後端定義必須包含用戶端憑證、金鑰和受信任 CA 憑證（如果使用，建議使用）的 Base64 編碼值。

您可以更新現有後端，以在基於憑證的方法和基於憑證的方法之間進行切換。但是，一次只能支援一種身份驗證方法。要切換到不同的身份驗證方法，必須從後端配置中刪除現有方法。

> 如果您嘗試同時提供憑證和證書，則後端建立將會失敗，並出現錯誤，提示設定檔中提供了多個驗證方法。

啟用基於憑證的身份驗證

Trident需要 SVM 範圍/叢集範圍管理員的憑證才能與ONTAP後端通訊。建議使用標準的、預先定義的角色，例如： `admin`或者 `vsadmin`。這樣可以確保與未來ONTAP版本向前相容，這些版本可能會公開一些功能 API，供未來的Trident版本使用。雖然可以建立自訂安全登入角色並將其與Trident一起使用，但不建議這樣做。

後端定義範例如下圖所示：

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

```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"
    }
}
```

請注意，後端定義是唯一以純文字形式儲存憑證的地方。後端建立完成後，使用者名稱/密碼將使用 Base64 進行編碼，並儲存為 Kubernetes 金鑰。後端建立/更新是唯一需要了解憑證的步驟。因此，這是一項僅限管理員執行的操作，由 Kubernetes/儲存管理員執行。

啟用基於憑證的身份驗證

新的和現有的後端都可以使用憑證與ONTAP後端通訊。後端定義需要三個參數。

- clientCertificate：客戶端憑證的 Base64 編碼值。
- clientPrivateKey：關聯私鑰的 Base64 編碼值。
- trustedCACertificate：受信任 CA 憑證的 Base64 編碼值。如果使用受信任的 CA，則必須提供此參數。如果沒有使用受信任的憑證授權機構，則可以忽略此步驟。

典型的工作流程包括以下步驟。

步驟

1. 產生客戶端憑證和金鑰。產生時，將通用名稱 (CN) 設定為要進行驗證的ONTAP使用者。

```
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. 在ONTAP叢集中新增受信任的 CA 憑證。這可能已經由儲存管理員處理了。如果沒有使用受信任的憑證授權機構，則忽略此操作。

```
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叢集上安裝客戶端憑證和金鑰（來自步驟 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安全登入角色支持 `cert`身份驗證方法。

```
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. 使用產生的憑證進行身份驗證。請將 < ONTAP管理 LIF> 和 <vserver 名稱> 替換為管理 LIF IP 位址和 SVM 名稱。您必須確保 LIF 的服務策略已設定為 `default-data-management`。

```
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 對憑證、金鑰和受信任的 CA 憑證進行編碼。

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

7. 使用上一步獲得的值建立後端。

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

更新身份驗證方法或輪換憑證

您可以更新現有後端，以使用不同的身份驗證方法或輪換其憑證。這種方法是雙向的：使用使用者名稱/密碼的後端可以更新為使用憑證；使用憑證的後端可以更新為基於使用者名稱/密碼的後端。為此，您必須刪除現有的身份驗證方法並新增新的身份驗證方法。然後使用包含所需參數的更新後的 backend.json 檔案來執行 `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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

輪換密碼時，儲存管理員必須先更新ONTAP上使用者的密碼。接下來將進行後端更新。輪換證書時，可以為使用者新增多個證書。然後更新後端以使用新證書，之後即可從ONTAP叢集中刪除舊證書。

更新後端不會中斷對已建立磁碟區的訪問，也不會影響之後建立的磁碟區連線。後端更新成功表明Trident可以與ONTAP後端通訊並處理未來的磁碟區操作。

為Trident建立自訂ONTAP角色

您可以建立一個具有最低權限的ONTAP叢集角色，這樣您就不必使用ONTAP管理員角色在Trident中執行操作。在Trident後端設定中包含使用者名稱時， Trident將使用您建立的ONTAP叢集角色來執行操作。

請參閱"Trident自訂角色產生器"有關建立Trident自訂角色的詳細資訊。

使用**ONTAP CLI**

1. 使用以下命令建立新角色：

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

2. 為Trident用戶建立使用者名稱：

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

3. 將角色映射到使用者：

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

使用系統管理員

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，請選擇「叢集 > 設定」。

      （或）若要在 SVM 層級建立自訂角色，請選擇「儲存」>「儲存虛擬機器」> `required SVM` > 設定 > 使用者和角色*。

   b. 選擇“使用者和角色”旁邊的箭頭圖示（→）。
   c. 在“角色”下選擇“+添加”。
   d. 定義角色規則，然後點選「儲存」。
2. 將角色對應到Trident使用者： + 在「使用者和角色」頁面上執行下列步驟：
   a. 在「使用者」下方選擇「新增」圖示 **+**。
   b. 選擇所需的使用者名，然後在「角色」下拉式選單中選擇角色。
   c. 點選“儲存”。

更多資訊請參閱以下頁面：

- "用於管理ONTAP的自訂角色"或者"定義自訂角色"
- "與角色和使用者協作"

管理 NFS 導出策略

Trident使用 NFS 匯出策略來控制對其所配置磁碟區的存取。

Trident在處理出口策略時提供兩種選擇：

- Trident可以動態管理匯出策略本身；在這種操作模式下，儲存管理員指定 CIDR 區塊列表，這些 CIDR 區塊代表可接受的 IP 位址。 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，如果沒有指定 CIDR，則會將發布捲的節點上找到的所有全域範圍的單播 IP 新增至匯出策略。
- 儲存管理員可以建立匯出策略並手動新增規則。除非在組態中指定了不同的導出策略名稱，否則Trident將使用預設導出策略。

動態管理出口策略

Trident提供了動態管理ONTAP後端匯出策略的功能。這樣，儲存管理員就可以指定工作節點 IP 的允許位址空間，而無需手動定義明確的規則。它大大簡化了導出策略管理；對導出策略的修改不再需要對儲存叢集進行人工幹預。此外，這有助於將對儲存叢集的存取限制在僅允許掛載磁碟區且 IP 位址在指定範圍內的節點上，從而支援細粒度和自動化管理。

使用動態匯出策略時，請勿使用網路位址轉換（NAT）。使用 NAT 時，儲存控制器看到的是前端 NAT 位址而不是實際的 IP 主機位址，因此在匯出規則中找不到匹配項時，存取將被拒絕。

例子

必須使用兩種配置選項。以下是一個後端定義範例：

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

使用此功能時，必須確保 SVM 中的根連接點具有先前建立的匯出策略，該策略的匯出規則允許節點 CIDR 區塊（例如預設匯出策略）。始終遵循NetApp推薦的最佳實踐，為Trident專用一個 SVM。

以下以上述範例為例，說明此功能的工作原理：

- `autoExportPolicy`設定為 `true`。這表示Trident會為使用從後端配置的每個磁碟區建立一個匯出策略。`svm1`使用 SVM 處理規則的新增和刪除 `autoexportCIDRs`地址塊。在磁碟區連接到節點之前，該磁碟區使用空的匯出策略，沒有任何規則來防止對該磁碟區的未經授權的存取。當磁碟區發佈到節點時，Trident會建立一個匯出策略，該策略的名稱與包含指定 CIDR 區塊內節點 IP 的底層 qtree 的名稱相同。這些 IP 位址也會加入到父FlexVol volume所使用的匯出策略中。
  - 例如：
    - 後端 UUID 403b5326-8482-40db-96d0-d83fb3f4daec
    - `autoExportPolicy`設定為 `true`

- 儲存前綴 `trident`
- PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
- 名為 trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c 的 qtree 為名為FlexVol的 FlexVol 建立了一個匯出策略。 `trident-403b5326-8482-40db96d0-d83fb3f4daec` ，名為 qtree 的匯出策略 `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`以及一個名為「空的匯出策略」的 `trident_empty`在支援向量機上。 FlexVol導出策略的規則將是 qtree 導出策略中包含的任何規則的超集。任何未附加的磁碟區都將重複使用空匯出策略。

- `autoExportCIDRs`包含地址塊列表。此字段為可選字段，預設值為 ["0.0.0.0/0", "::/0"]。如果未定義，Trident會新增在工作節點上找到的所有具有發佈的全域作用域的單播位址。

在這個例子中， `192.168.0.0/24`已提供地址空間。這意味著，位於此位址範圍內且已發佈 Kubernetes 節點 IP 的節點將被加入到Trident建立的匯出策略中。當Trident註冊它運行所在的節點時，它會檢索該節點的 IP 位址，並將其與提供的位址區塊進行比對。 `autoExportCIDRs`在發佈時， Trident在過濾 IP 位址後，會為它要發佈到的節點的用戶端 IP 位址建立匯出政策規則。

您可以更新 `autoExportPolicy`和 `autoExportCIDRs`建立後端之後，需要進行以下操作。您可以為自動管理的後端新增新的 CIDR，或刪除現有的 CIDR。刪除 CIDR 時要格外小心，確保現有連線不會中斷。您也可以選擇停用 `autoExportPolicy`對於後端，如果出現問題，則回退到手動建立的匯出策略。這將需要進行設置 `exportPolicy`後端配置中的參數。

Trident建立或更新後端後，您可以使用下列指令檢查後端： `tridentctl`或相應的 `tridentbackend`CRD：

```
./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
```

當一個節點被移除時， Trident會檢查所有匯出策略，以移除與該節點對應的存取規則。透過從受管後端匯出政策移除此節點 IP， Trident可以防止惡意掛載，除非叢集中的新節點重新使用此 IP。

對於先前存在的後端，使用以下方式更新後端： `tridentctl update backend`確保Trident自動管理出口策略。這樣，當需要時，就會建立兩個以以後端 UUID 和 qtree 名稱命名的新匯出策略。後端存在的磁碟區在卸載並重新掛載後，將使用新建立的匯出策略。

> ⓘ 刪除具有自動管理匯出策略的後端將刪除動態建立的匯出策略。如果後端被重新創建，它將被視為一個新的後端，並將導致創建一個新的匯出策略。

如果正在執行中的節點的 IP 位址更新，則必須重新啟動該節點上的Trident pod。 Trident隨後將更新其管理的後端的匯出策略，以反映此 IP 變更。

準備配置SMB卷

稍作準備，您就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 協定才能建立 `ontap-nas-economy`適用於ONTAP本地叢集的 SMB 磁碟區。如果未能配置這些協定中的任何一個，都會導致 SMB 磁碟區建立失敗。

> ⓘ `autoExportPolicy`不支援 SMB 磁碟區。

開始之前

在配置 SMB 磁碟區之前，您必須具備以下條件。

- 一個 Kubernetes 叢集，包含一個 Linux 控制器節點和至少一個執行 Windows Server 2022 的 Windows 工作節點。 Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。
- 至少有一個包含您的 Active Directory 憑證的Trident金鑰。生成秘密 `smbcreds`：

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

- 配置為 Windows 服務的 CSI 代理程式。要配置 `csi-proxy`，請參閱"GitHub：CSI代理"或者"GitHub：適用於 Windows 的 CSI 代理"適用於在 Windows 上執行的 Kubernetes 節點。

步驟

1. 對於本地部署的ONTAP，您可以選擇建立 SMB 共享，或者Trident可以為您建立一個。

   > ⓘ Amazon FSx for ONTAP需要 SMB 共用。

   您可以透過以下兩種方式之一建立 SMB 管理共用："Microsoft 管理控制台"共用資料夾管理單元或使用ONTAP CLI。使用ONTAP CLI 建立 SMB 共享：

   a. 如有必要，請建立共用的目錄路徑結構。

      這 `vserver cifs share create`此指令檢查在建立共用時 -path 選項中指定的路徑。如果指定的路徑不存在，則命令執行失敗。

   b. 建立與指定 SVM 關聯的 SMB 共用：

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

c. 確認共享已建立：

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

請參閱"建立 SMB 共享"了解詳細資訊。

2. 建立後端時，必須配置以下內容以指定 SMB 磁碟區。有關所有 FSx for ONTAP後端設定選項，請參閱"FSx for ONTAP設定選項和範例"。

範圍	描述	例子
`smbShare`	您可以指定以下一項：使用 Microsoft 管理控制台或ONTAP CLI 建立的 SMB 共用的名稱；允許Trident建立 SMB 共用的名稱；或者您可以將參數留空以防止對磁碟區的公共共用存取。對於本地部署的ONTAP，此參數是可選的。此參數是Amazon FSx for ONTAP後端所必需的，不能為空。	`smb-share`
`nasType`	*必須設定為 `smb`.*如果為空，則預設為空。 `nfs` 。	`smb`
`securityStyle`	新磁碟區的安全樣式。 *必須設定為 `ntfs`或者 `mixed`適用於 SMB 卷。 *	`ntfs`或者 `mixed`適用於 SMB 卷
`unixPermissions`	新卷模式。 *對於 SMB 卷，此項必須留空。 *	“”

啟用安全的 **SMB**

從 25.06 版本開始， NetApp Trident支援使用以下方式安全配置建立的 SMB 磁碟區： `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。

需要記住的要點

- 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存取控制清單 (ACL) 中。
- 如果在後端、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，然後是後端。
- 支援安全 SMB `ontap-nas`僅適用於託管磁碟區匯入，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`對儲存類別進行註解，以啟用安全可靠的 SMB 連線。使用者為註釋指定的值 `trident.netapp.io/smbShareAdUser`應該與指定的使用者名稱相同 `smbcreds`秘密。您可以從以下選項中選擇一項 `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. 製作PVC管。

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
```

ONTAP NAS 設定選項和範例

學習如何在Trident系統中建立和使用ONTAP NAS 驅動程式。本節提供後端設定範例以及將後端對應到 StorageClasses 的詳細資訊。

後端配置選項

請參閱下表以了解後端配置選項：

範圍	描述	預設
`version`		始終為 1
`storageDriverName`	儲存驅動程式的名稱	`ontap-nas`， `ontap-nas-economy` ， 或者 `ontap-nas-flexgroup`
`backendName`	自訂名稱或儲存後端	驅動程式名稱 + "_" + dataLIF
`managementLIF`	叢集或 SVM 管理 LIF 的 IP 位址可以指定完全限定網域名稱 (FQDN)。如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。 IPv6 位址必須用方括號定義，例如： `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 。為了實現MetroCluster 的無縫切換，請參閱MetroCluster範例。	"10.0.0.1", "[2001:1234:abcd::fefe]"

範圍	描述	預設
`dataLIF`	LIF協定的IP位址。 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。參考。如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。 IPv6 位址必須用方括號定義，例如： `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 。 *Metrocluster 除外。 *查看MetroCluster範例。	指定位址或從 SVM 派生，如果未指定（不建議）
`svm`	要使用的儲存虛擬機器 *Metrocluster 除外。 *查看MetroCluster範例。	如果是 SVM 則推導而來 `managementLIF`已指定
`autoExportPolicy`	啟用自動匯出策略建立和更新 [布林值]。使用 `autoExportPolicy`和 `autoExportCIDRs`選用功能包括： Trident可以自動管理出口策略。	錯誤的
`autoExportCIDRs`	用於過濾 Kubernetes 節點 IP 的 CIDR 列表 `autoExportPolicy`已啟用。使用 `autoExportPolicy`和 `autoExportCIDRs`選用功能包括： Trident可以自動管理出口策略。	["0.0.0.0/0", "::/0"]`
`labels`	若要套用於磁碟區的任意 JSON 格式標籤集	“”
`clientCertificate`	用戶端憑證的 Base64 編碼值。用於基於憑證的身份驗證	“”
`clientPrivateKey`	客戶端私鑰的 Base64 編碼值。用於基於憑證的身份驗證	“”
`trustedCACertificate`	受信任 CA 憑證的 Base64 編碼值。選修的。用於基於憑證的身份驗證	“”
`username`	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"。	
`password`	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"。	
`storagePrefix`	在 SVM 中配置新磁碟區時所使用的前綴。設定後無法更新 當使用 ontap-nas-economy 和 24 個或更多字元的 storagePrefix 時，qtree 將不會嵌入儲存前綴，儘管它會包含在磁碟區名稱中。	“三叉戟”

範圍	描述	預設
`aggregate`	用於配置的聚合（可選；如果設置，則必須指派給SVM）。對於 `ontap-nas-flexgroup`驅動程序，此選項將被忽略。如果未分配，則可以使用任何可用的聚合來配置FlexGroup磁碟區。 當 SVM 中的聚合資料更新時， Trident會自動輪詢 SVM 進行更新，而無需重新啟動Trident控制器。當您在Trident中配置特定聚合以配置磁碟區時，如果該聚合被重新命名或移出 SVM，則在輪詢 SVM 聚合時， Trident中的後端將變為失敗狀態。您必須將聚合變更為 SVM 上存在的聚合，或將其完全刪除，才能使後端恢復連線。	“”
`limitAggregateUsage`	如果使用率超過此百分比，則配置失敗。 不適用於**Amazon FSx for ONTAP**。	（預設不強制執行）
flexgroupAggregateList	用於配置的聚合清單（可選；如果設置，則必須指派給 SVM）。指派給 SVM 的所有聚合都用於配置FlexGroup磁碟區。支援 **ontap-nas-flexgroup** 儲存驅動程式。 當 SVM 中的聚合列表更新時， Trident會透過輪詢 SVM 自動更新該列表，而無需重新啟動Trident控制器。當您在Trident中設定了特定的聚合清單來設定磁碟區時，如果聚合清單被重新命名或移出 SVM，則在輪詢 SVM 聚合時，Trident中的後端將進入失敗狀態。您必須將聚合列表變更為 SVM 上存在的列表，或將其完全刪除，才能使後端恢復連線。	“”
`limitVolumeSize`	如果請求的磁碟區大小大於此值，則配置失敗。此外，它還限制了 qtree 管理的捲的最大大小，以及`qtreesPerFlexvol`此選項允許自訂每個FlexVol volume的最大 qtree 數量。	（預設不強制執行）
`debugTraceFlags`	故障排除時要使用的調試標誌。例如，{"api":false, "method":true} 請勿使用 `debugTraceFlags`除非您正在進行故障排除並且需要詳細的日誌轉儲。	無效的
`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。	`nfs`

範圍	描述	預設
nfsMountOptions	以逗號分隔的 NFS 掛載選項清單。 Kubernetes 持久性磁碟區的掛載選項通常在儲存類別中指定，但如果儲存類別中沒有指定掛載選項， Trident將回退到使用儲存後端設定檔中指定的掛載選項。如果在儲存類別或設定檔中未指定任何掛載選項， Trident將不會在關聯的持久性磁碟區上設定任何掛載選項。	“”
qtreesPerFlexvol	每個FlexVol 的最大 Qtree 數量必須在 [50, 300] 範圍內	“200”
smbShare	您可以指定以下一項：使用 Microsoft 管理控制台或ONTAP CLI 建立的 SMB 共用的名稱；允許Trident建立 SMB 共用的名稱；或者您可以將參數留空以防止對磁碟區的公共共用存取。對於本地部署的ONTAP，此參數是可選的。此參數是Amazon FSx for ONTAP後端所必需的，不能為空。	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。	true`適用於ONTAP 9.15.1 或更高版本，否則 `false。
limitVolumePoolSize	在 ontap-nas-economy 後端使用 Qtrees 時可請求的最大FlexVol大小。	（預設不強制執行）
denyNewVolumePools	限制 `ontap-nas-economy`後端創建新的FlexVol磁碟區來包含它們的 Qtree。只有預先存在的 Flexvol 才能用於配置新的 PV。	
adAdminUser	具有對 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 共用提供管理員權限，並賦予其完全控制權限。	

用於配置磁碟區的後端配置選項

您可以使用下列選項控制預設配置。 `defaults`配置部分。例如，請參閱下面的設定範例。

範圍	描述	預設
spaceAllocation	Q樹的空間分配	“真的”
spaceReserve	空間預留模式；「無」（細）或「大量」（粗）	“沒有任何”
snapshotPolicy	要使用的快照策略	“沒有任何”
qosPolicy	若要為建立的磁碟區指派的 QoS 策略群組。每個儲存池/後端選擇 qosPolicy 或 adaptiveQosPolicy 之一	“”

範圍	描述	預設
`adaptiveQosPolicy`	若要為建立的磁碟區指派的自適應 QoS 策略群組。每個儲存池/後端選擇 qosPolicy 或 adaptiveQosPolicy 之一。 ontap-nas-economy 不支援此功能。	“”
`snapshotReserve`	快照預留的磁碟區百分比	如果為“0”， `snapshotPolicy`為“無”，否則為“
`splitOnClone`	創建時將克隆體從其母體中分離出來	“錯誤的”
`encryption`	在新磁碟區啟用NetApp磁碟區加密 (NVE)；預設為 `false`。若要使用此選項，必須在叢集上取得 NVE 許可並啟用 NVE。如果後端啟用了 NAE，則在Trident中配置的任何磁碟區都會啟用 NAE。更多信息，請參閱："Trident如何與 NVE 和 NAE 協同工作" 。	“錯誤的”
`tieringPolicy`	分層策略使用“無”	
`unixPermissions`	新卷模式	NFS 磁碟區的連接埠號碼為「777」；SMB 磁碟區的連接埠號碼為空（不適用）。
`snapshotDir`	控制對以下方面的訪問 `.snapshot`目錄	NFSv4 為“true”，NFSv3 為“false”。
`exportPolicy`	出口政策的使用	"預設"
`securityStyle`	新磁碟區的安全樣式。 NFS 支持 `mixed`和 `unix`安全措施。中小企業支持 `mixed`和 `ntfs`安全措施。	NFS 預設值為 `unix`。 SMB 預設值為 `ntfs`。
`nameTemplate`	用於建立自訂磁碟區名稱的範本。	“”

將 QoS 策略群組與Trident結合使用需要ONTAP 9.8 或更高版本。您應該使用非共享的 QoS 策略群組，並確保該策略群組單獨套用至每個成員。共享的 QoS 策略群組強制規定所有工作負載的總吞吐量上限。

卷配置範例

以下是一個定義了預設值的範例：

```
---
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 。此計算方法可確保使用者在 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：

計算方法如下：

```
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 請求中請求的大小這 `volume show`該命令應顯示與此範例類似的結果：

升級Trident時，先前安裝的現有後端將以上述方式設定磁碟區。對於升級前建立的捲，您應該調整其大小以觀察到變更。例如，一個 2 GiB 的 PVC `snapshotReserve=50`先前的結果是一個提供 1 GiB 可寫空間的磁碟區。例如，將磁碟區大小調整為 3 GiB，將在 6 GiB 的磁碟區上為應用程式提供 3 GiB 的可寫入空間。

最小配置範例

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。

> 如果您在NetApp ONTAP上使用Amazon FSx和Trident，建議為 LIF 指定 DNS 名稱而非 IP 位址。

ONTAP NAS 經濟範例

```
---
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，以避免在切換和切換回後端後手動更新後端定義。"SVM複製與復原" 。

為了實現無縫切換和切換回，請指定 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 磁碟區範例

```
---
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
```

基於憑證的身份驗證範例

這是一個最小後端設定範例。 `clientCertificate` ， `clientPrivateKey` ， 和 `trustedCACertificate`（如果使用受信任的 CA，則為可選） `backend.json`分別取客戶端憑證、私鑰和受信任 CA 憑證的 base64 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_
```

自動出口政策範例

本範例向您展示如何指示Trident使用動態匯出策略來自動建立和管理匯出策略。這對於以下情況也適用：`ontap-nas-economy`和 `ontap-nas-flexgroup`司機。

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

IPv6 位址範例

這個例子表明 `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
```

使用 **SMB** 磁碟區的**Amazon FSx for ONTAP**範例

這 `smbShare`使用 SMB 磁碟區的 FSx for ONTAP需要此參數。

```
---
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_
```

使用 **nameTemplate** 的後端設定範例

```
---
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}}"
```

具有虛擬池的後端範例

在下方所示的範例後端定義檔中，所有儲存池都設定了特定的預設值，例如： `spaceReserve`沒有，`spaceAllocation`為假，並且 `encryption`錯誤。虛擬池在儲存部分中定義。

Trident在「備註」欄位中設定配置標籤。 FlexVol上已設定評論 `ontap-nas`或`FlexGroup `ontap-nas-flexgroup`。 Trident在設定時會將虛擬池上的所有標籤複製到儲存磁碟區。為了方便起見，儲存管理員可以為每個虛擬池定義標籤，並按標籤將磁碟區分組。

在這些範例中，一些儲存池會設定自己的參數。 `spaceReserve`， `spaceAllocation`， 和 `encryption`有些值會覆蓋預設值，有些池會覆蓋預設值。

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"
```

將後端映射到儲存類

以下 StorageClass 定義指的是[具有虛擬池的後端範例]。使用 `parameters.selector`在欄位中，每個 StorageClass 都會指出哪些虛擬池可用於託管磁碟區。卷將具有所選虛擬池中定義的各個方面。

- 這 `protection-gold`StorageClass 將會對應到第一個和第二個虛擬池。 `ontap-nas-flexgroup`後端。只有這些泳池提供黃金級保護。

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

- 這 `protection-not-gold`StorageClass 將會對應到第三個和第四個虛擬池。 `ontap-nas-flexgroup`後端。除了黃金之外，只有這些金池提供其他等級的保護。

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

- 這 `app-mysqldb`StorageClass 將會對應到第四個虛擬池。 `ontap-nas`後端。這是唯一一個為mysqldb類型應用程式提供儲存池配置的儲存池。

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

- 該 `protection-silver-creditpoints-20k`StorageClass 將會對應到第三個虛擬池。 `ontap-nas-flexgroup`後端。這是唯一提供銀級保護和 20000 積分的彩池。

```
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"
```

- 這 `creditpoints-5k`StorageClass 將會對應到第三個虛擬池。 `ontap-nas`後端和第二個虛擬池 `ontap-nas-economy`後端。這是唯一提供 5000 點的彩池產品。

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

Trident將決定選擇哪個虛擬池，並確保滿足儲存需求。

更新 `dataLIF`初始配置後

初始設定完成後，您可以透過執行以下命令來變更 dataLIF，從而為新的後端 JSON 檔案提供更新的 dataLIF。

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

如果 PVC 連接到一個或多個艙體，則必須將所有相應的艙體放下，然後再將它們重新升起，以便新的 dataLIF 生效。

安全的中小企業範例

使用 **ontap-nas** 驅動程式進行後端配置

```
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** 驅動程式進行後端配置

```
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
```

後端配置及儲存池

```
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
```

使用 **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
```

請確保添加 `annotations`實作安全的SMB。如果沒有註釋，無論後端或 PVC 中設定了什麼配置，安全 SMB 都無法運作。

使用 **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
```

包含單一 **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
```

包含多個 **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
```

Amazon FSx for NetApp ONTAP

將Trident與Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP"是一項完全託管的 AWS 服務，可讓客戶啟動和執行由NetApp ONTAP儲存作業系統支援的檔案系統。 FSx for ONTAP可讓您利用您熟悉的NetApp功能、效能和管理能力，同時享受在 AWS 上儲存資料的簡單性、敏捷性、安全性和可擴充性。 FSx for ONTAP支援ONTAP檔案系統功能和管理 API。

您可以將Amazon FSx for NetApp ONTAP檔案系統與Trident集成，以確保在 Amazon Elastic Kubernetes Service (EKS) 中執行的 Kubernetes 叢集可以設定由ONTAP支援的區塊和檔案持久磁碟區。

檔案系統是Amazon FSx中的主要資源，類似本地的ONTAP叢集。在每個 SVM 中，您可以建立一個或多個卷，這些卷是用於儲存檔案系統中的檔案和資料夾的資料容器。 Amazon FSx for NetApp ONTAP將作為雲端託管檔案系統提供。新的檔案系統類型稱為 * NetApp ONTAP*。

透過將Trident與Amazon FSx for NetApp ONTAP結合使用，您可以確保在 Amazon Elastic Kubernetes Service (EKS) 中執行的 Kubernetes 叢集可以設定由ONTAP支援的區塊和檔案持久性磁碟區。

要求

另外"Trident的要求"要將 FSx for ONTAP與Trident集成，您需要：

- 現有的 Amazon EKS 叢集或自管理 Kubernetes 叢集 `kubectl`已安裝。
- 叢集工作節點可存取的現有Amazon FSx for NetApp ONTAP檔案系統和儲存虛擬機器 (SVM)。
- 已準備好的工作節點"NFS 或 iSCSI"。

> 請務必按照 Amazon Linux 和 Ubuntu 所需的節點準備步驟進行操作。 "亞馬遜機器圖像"（AMI）取決於您的 EKS AMI 類型。

注意事項

- SMB 卷：
  - 使用以下方式支援 SMB 卷 `ontap-nas`僅限司機。
  - Trident EKS 外掛程式不支援 SMB 磁碟區。
  - Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。參考 "準備配置SMB卷" 了解詳情。
- 在Trident 24.02 之前，在Amazon FSx檔案系統上建立的、啟用了自動備份的磁碟區無法被Trident刪除。為防止在Trident 24.02 或更高版本中出現此問題，請指定 `fsxFilesystemID`，AWS `apiRegion` ，AWS `apikey`以及 AWS `secretKey`在 AWS FSx for ONTAP的後端設定檔中。

> 如果您要為Trident指定 IAM 角色，則可以省略指定以下內容： `apiRegion` ， `apiKey` ， 和 `secretKey`明確地將欄位傳遞給Trident 。更多信息，請參閱"FSx for ONTAP設定選項和範例"。

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設定但排除 EBS 磁碟進行多路徑設定的檔案：

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

驗證

Trident提供兩種身分驗證方式。

- 基於憑證（建議）：將憑證安全地儲存在 AWS Secrets Manager 中。您可以使用 `fsxadmin`檔案系統的使用者或 `vsadmin`使用者已配置到您的 SVM。

> Trident預計將以...的形式運營 `vsadmin`SVM 用戶，或使用具有相同角色但名稱不同的用戶。 Amazon FSx for NetApp ONTAP具有 `fsxadmin`該用戶是ONTAP的有限替代品 `admin`集群用戶。我們強烈建議使用 `vsadmin`用Trident。

- 基於憑證： Trident將使用安裝在 SVM 上的憑證與 FSx 檔案系統上的 SVM 進行通訊。

有關啟用身份驗證的詳細信息，請參閱您的驅動程式類型的身份驗證說明：

- "ONTAP NAS 認證"
- "ONTAP SAN 驗證"

已測試的亞馬遜機器映像 (AMI)

EKS 叢集支援各種作業系統，但 AWS 針對容器和 EKS 優化了某些 Amazon Machine Image (AMI)。以下 AMI 已使用NetApp Trident 25.02 進行測試。

急性心肌梗塞	網路儲存	NAS經濟	iSCSI	iSCSI經濟型
AL2023_x86_64_STANDARD	是的	是的	是的	是的
AL2_x86_64	是的	是的	是的*	是的*
BOTTLEROCKET_x86_64	是的**	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	是的*	是的*
BOTTLEROCKET_ARM_64	是的**	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。

> 如果您所需的 AMI 未在此處列出，並不意味著它不受支援；這僅僅意味著它尚未經過測試。此清單可作為 AMI 已知可用系統的指南。

使用以下工具進行測試：

- 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 都進行了測試。

- SAN 僅測試了 iSCSI，未測試 NVMe-oF。

已執行測試：

- 建立：儲存類別、PVC、Pod
- 刪除：pod、pvc（常規、qtree/lun – 經濟型、有 AWS 備份的 NAS）

查找更多信息

- "Amazon FSx for NetApp ONTAP文檔"
- "關於Amazon FSx for NetApp ONTAP的部落格文章"

建立 **IAM** 角色和 **AWS Secret**

您可以設定 Kubernetes pod 以透過 AWS IAM 角色進行驗證來存取 AWS 資源，而不是提供明確的 AWS 憑證。

若要使用 AWS IAM 角色進行驗證，您必須擁有一個使用 EKS 部署的 Kubernetes 叢集。

建立 **AWS Secrets Manager** 金鑰

由於Trident將向 FSx 虛擬伺服器發出 API 來為您管理存儲，因此它需要憑證才能執行此操作。傳遞這些憑證的安全方法是透過 AWS Secrets Manager 金鑰。因此，如果您還沒有 AWS Secrets Manager 金鑰，則需要建立一個包含 vsadmin 帳戶憑證的金鑰。

此範例建立一個 AWS Secrets Manager 金鑰來儲存Trident CSI 憑證：

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

建立 **IAM** 策略

Trident也需要 AWS 權限才能正常運作。因此，您需要建立一個策略，賦予Trident所需的權限。

以下範例使用 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"
```

策略 **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"
}
```

建立 **Pod** 身分或 **IAM** 角色以關聯服務帳戶 **(IRSA)**

您可以設定 Kubernetes 服務帳戶，使其承擔 AWS Identity and Access Management (IAM) 角色（使用 EKS Pod Identity 或 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 身分關聯可讓您管理應用程式的憑證，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 **EKS** 叢集上安裝 **Pod Identity**：

您可以透過 AWS 控制台建立 Pod 標識，也可以使用下列 AWS CLI 命令：

```
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 身分代理"。

建立 **trust-relationship.json** 檔案：

建立 trust-relationship.json 文件，使 EKS 服務主體能夠承擔 Pod 身分的此角色。然後使用以下信任策略建立角色：

```
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：

將上一個步驟中建立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
```

trust-relationship.json 文件：

```
{
  "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"
        }
      }
    }
  ]
}
```

更新以下值 `trust-relationship.json`文件：

- **<account_id>** - 您的 AWS 帳戶 ID
- **<oidc_provider>** - 您的 EKS 叢集的 OIDC。您可以透過執行以下命令來取得oidc_provider：

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

將 **IAM** 角色與 **IAM** 原則關聯：

角色創建完成後，使用以下命令將策略（在上一個步驟中建立的策略）附加到該角色：

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

請核實 **OICD** 提供者是否已關聯：

請確認您的 OIDC 提供者已與您的叢集關聯。您可以使用以下命令進行驗證：

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

如果輸出為空，請使用以下命令將 IAM OIDC 關聯到您的叢集：

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

如果您使用的是 **eksctl**，請使用以下範例在 EKS 中為服務帳戶建立 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 ：

- 舵
- EKS 附加元件

如果要使用快照功能，請安裝 CSI 快照控制器外掛程式。請參閱"為CSI磁碟區啟用快照功能"了解更多。

透過 **Helm** 安裝**Trident**。

Pod 身份

1. 新增Trident Helm 倉庫：

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

2. 請依照以下範例安裝Trident ：

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

您可以使用 `helm list`查看安裝詳細資訊的命令，例如名稱、命名空間、圖表、狀態、應用程式版本和修訂號。

```
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 倉庫：

```
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`查看安裝詳細資訊的命令，例如名稱、命名空間、圖表、狀態、應用程式版本和修訂號。

```
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-
100.2506.0    25.06.0
```

如果您打算使用 iSCSI，請確保您的用戶端電腦上已啟用 iSCSI。如果您使用的是 AL2023 工作節點作業系統，可以透過在 Helm 安裝中新增節點準備參數來自動安裝 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** 外掛程式安裝**Trident**

Trident EKS 外掛程式包含最新的安全性修補程式和錯誤修復，並經過 AWS 驗證，可與 Amazon EKS 搭配使用。 EKS 外掛程式可讓您持續確保您的 Amazon EKS 叢集安全穩定，並減少安裝、設定和更新外掛程式所需的工作量。

先決條件

在為 AWS EKS 設定Trident外掛程式之前，請確保您已具備以下條件：

- 附加訂閱的 Amazon EKS 叢集帳戶
- AWS 對 AWS Marketplace 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

為 **AWS** 啟用**Trident**插件

管理控制台

1. 開啟 Amazon EKS 主控台 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左側導覽窗格中，選擇「集群」。
3. 選擇要為其配置NetApp Trident CSI 外掛程式的叢集名稱。
4. 選擇“附加元件”，然後選擇“取得更多附加元件”。
5. 請依照以下步驟選擇外掛：
   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「Trident by NetApp」方塊右上角的複選框。
   c. 選擇“下一步”。
6. 在「配置所選外掛」設定頁面上，執行以下操作：

   *如果您使用 Pod Identity 關聯，請跳過這些步驟。 *

   a. 請選擇您要使用的*版本*。
   b. 如果您使用IRSA身份驗證，請確保設定可選配置設定中提供的配置值：
      - 請選擇您要使用的*版本*。
      - 依照*外掛程式設定方案*進行操作，並將*配置值*部分中的*configurationValues*參數設定為您在上一個步驟建立的角色 ARN（值應採用下列格式）：

```
{

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

}
```

+
如果選擇「覆蓋」作為衝突解決方法，則現有外掛程式的一個或多個設定可能會被 Amazon EKS 外掛程式設定覆蓋。如果您不啟用此選項，並且與您現有的設定有衝突，則操作將會失敗。您可以使用產生的錯誤訊息來排查衝突問題。在選擇此選項之前，請確保 Amazon EKS 外掛程式不會管理您需要自行管理的設定。

7. 選擇“下一步”。
8. 在「審核和新增」頁面上，選擇「建立」。

   插件安裝完成後，您將看到已安裝的插件。

AWS CLI

1.創建 **`add-on.json`**文件：

對於 **Pod** 標識，請使用以下格式：

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

對於**IRSA**認證，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"
  }
}
```

代替 `<role ARN>`使用上一步創建的角色的 ARN。

*2.安裝Trident EKS 外掛程式。 *

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

eksctl

以下範例指令安裝Trident EKS 外掛：

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

更新**Trident EKS** 外掛

管理控制台

1. 開啟 Amazon EKS 主控台 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左側導覽窗格中，選擇「集群」。
3. 選擇要更新NetApp Trident CSI 外掛程式的叢集名稱。
4. 選擇“附加元件”標籤。
5. 選擇“ NetApp Trident ”，然後選擇“編輯”。
6. 在「配置NetApp Trident 」頁面上，執行以下操作：
   a. 請選擇您要使用的*版本*。
   b. 展開“可選配置設定”，並根據需要進行修改。
   c. 選擇“儲存變更”。

AWS CLI

以下範例更新 EKS 外掛：

```
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

- 請檢查您的 FSxN Trident CSI 外掛程式的目前版本。代替 `my-cluster`使用您的叢集名稱。

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

  範例輸出：

```
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'"}
```

- 將插件更新到上一步輸出中「UPDATE AVAILABLE」下傳回的版本。

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

如果你移除 `--force`如果選項和任何 Amazon EKS 外掛程式設定與您現有的設定衝突，則更新 Amazon EKS 外掛程式將失敗；您將收到錯誤訊息，以協助您解決衝突。在指定此選項之前，請確保 Amazon EKS 外掛程式不會管理您需要管理的設置，因為此選項會覆寫這些設定。有關此設定的其他選項的更多信息，請參閱"外掛"。有關 Amazon EKS Kubernetes 欄位管理的更多信息，請參閱"Kubernetes 欄位管理"。

解除安裝/移除**Trident EKS** 插件

您可以透過兩種方式移除 Amazon EKS 外掛：

- 保留叢集上的附加軟體 – 此選項移除 Amazon EKS 對所有設定的管理。它還取消了 Amazon EKS 通知您更新以及在您啟動更新後自動更新 Amazon EKS 外掛程式的功能。但是，它可以保留叢集上的附加軟體。此選項使外掛程式成為自管理安裝，而不是 Amazon EKS 外掛程式。選擇此選項，插件不會出現停機時間。保留 `--preserve`命令中的選項用於保留插件。
- 從叢集中完全移除附加軟體 – NetApp建議，僅當叢集中沒有任何資源依賴 Amazon EKS 附加元件時，才會從叢集中移除該附加元件。移除 `--preserve`選項 `delete`移除插件的命令。

如果外掛程式關聯了 IAM 帳戶，則不會刪除該 IAM 帳戶。

管理控制台

1. 開啟 Amazon EKS 主控台 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左側導覽窗格中，選擇「集群」。
3. 選擇要從中移除NetApp Trident CSI 外掛程式的叢集名稱。
4. 選擇“附加元件”選項卡，然後選擇“ NetApp Trident ”。
5. 選擇*刪除*。
6. 在「移除 netapp_trident-operator 確認」對話方塊中，執行下列操作：
   a. 如果您希望 Amazon EKS 停止管理外掛程式的設置，請選擇「在叢集上保留」。如果您希望在叢集上保留附加軟體，以便您可以自行管理附加軟體的所有設置，請執行此操作。
   b. 輸入 **netapp_trident-operator**。
   c. 選擇*刪除*。

AWS CLI

代替 `my-cluster`輸入叢集名稱，然後執行以下命令。

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

eksctl

以下指令卸載Trident EKS 外掛程式：

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

ONTAP SAN 和 NAS 驅動程式集成

若要建立儲存後端，您需要建立 JSON 或 YAML 格式的設定檔。該文件需要指定您想要的儲存類型（NAS 或 SAN）、檔案系統、要從中取得儲存的 SVM 以及如何對其進行驗證。以下範例展示如何定義基於 NAS 的存儲，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"
    }
  }
}
```

執行下列命令以建立和驗證Trident後端設定 (TBC)：

- 從 yaml 檔案建立 Trident 後端設定 (TBC)，並執行下列命令：

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

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

- 驗證 Trident 後端設定 (TBC) 是否已成功建立：

```
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
```

FSx 適用於ONTAP驅動程式的詳細信息

您可以使用下列驅動程式將Trident與Amazon FSx for NetApp ONTAP整合：

- `ontap-san`每個已設定的 PV 都是其自身Amazon FSx for NetApp ONTAP磁碟區中的一個 LUN。推薦用於塊存儲。
- `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-economy`每個已設定的 PV 都是一個 LUN，每個Amazon FSx for NetApp ONTAP磁碟區可設定 LUN 的數量。
- `ontap-nas-economy`每個已配置的 PV 都是一個 qtree，每個Amazon FSx for NetApp ONTAP磁碟區可以配置 qtree 的數量。
- `ontap-nas-flexgroup`每個已配置的 PV 都是一個完整的Amazon FSx for NetApp ONTAP FlexGroup磁碟區。

有關司機詳情，請參閱"NAS驅動程式"和"SAN驅動程式"。

設定檔建立完成後，執行以下命令將其建立在 EKS 中：

```
kubectl create -f configuration_file
```

若要驗證狀態，請執行以下命令：

```
kubectl get tbc -n trident
```

```
NAME                      BACKEND NAME            BACKEND UUID
PHASE   STATUS
backend-fsx-ontap-nas   backend-fsx-ontap-nas   7a551921-997c-4c37-a1d1-
f2f4c87fa629   Bound   Success
```

後端高級配置和範例

請參閱下表以了解後端配置選項：

範圍	描述	例子
`version`		始終為 1
`storageDriverName`	儲存驅動程式的名稱	`ontap-nas`，`ontap-nas-economy`，`ontap-nas-flexgroup`，`ontap-san`，`ontap-san-economy`
`backendName`	自訂名稱或儲存後端	驅動程式名稱 + "_" + dataLIF
`managementLIF`	叢集或 SVM 管理 LIF 的 IP 位址可以指定完全限定網域名稱 (FQDN)。如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。IPv6 位址必須用方括號定義，例如 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。如果您提供 `fsxFilesystemID`在 `aws`在欄位中，您無需提供 `managementLIF`因為Trident可以檢索 SVM `managementLIF`來自 AWS 的資訊。因此，您必須提供 SVM 下使用者的憑證（例如：vsadmin），且該使用者必須擁有下列權限：`vsadmin`角色。	"10.0.0.1", "[2001:1234:abcd::fefe]"

範圍	描述	例子
`dataLIF`	LIF協定的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。參考。 * ONTAP SAN 驅動程式*：不要為 iSCSI 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 iSCI LIF。如果明確定義了 dataLIF，則會產生警告。如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。 IPv6 位址必須用方括號定義，例如 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。	
`autoExportPolicy`	啟用自動匯出策略建立和更新 [布林值]。使用 `autoExportPolicy`和 `autoExportCIDRs`選用功能包括：Trident可以自動管理出口策略。	`false`
`autoExportCIDRs`	用於過濾 Kubernetes 節點 IP 的 CIDR 列表 `autoExportPolicy`已啟用。使用 `autoExportPolicy`和 `autoExportCIDRs`選用功能包括：Trident可以自動管理出口策略。	"["0.0.0.0/0", "::/0"]"
`labels`	若要套用於磁碟區的任意 JSON 格式標籤集	“”
`clientCertificate`	用戶端憑證的 Base64 編碼值。用於基於憑證的身份驗證	“”
`clientPrivateKey`	客戶端私鑰的 Base64 編碼值。用於基於憑證的身份驗證	“”
`trustedCACertificate`	受信任 CA 憑證的 Base64 編碼值。選修的。用於基於憑證的身份驗證。	“”
`username`	連接到叢集或 SVM 的使用者名稱。用於基於憑證的身份驗證。例如，vsadmin。	
`password`	連接叢集或SVM的密碼。用於基於憑證的身份驗證。	
`svm`	使用的儲存虛擬機	如果指定了 SVM 管理 LIF，則派生出該 LIF。
`storagePrefix`	在 SVM 中配置新磁碟區時所使用的前綴。創建後無法修改。要更新此參數，您需要建立一個新的後端。	`trident`

範圍	描述	例子
`limitAggregateUsage`	*請勿指定用於Amazon FSx for NetApp ONTAP。 *提供的 `fsxadmin`和 `vsadmin`不包含檢索匯總使用情況和使用Trident限制它所需的權限。	請勿使用。
`limitVolumeSize`	如果請求的磁碟區大小大於此值，則配置失敗。此外，它還限制了其管理的 qtree 和 LUN 卷的最大大小，以及 `qtreesPerFlexvol`此選項允許自訂每個FlexVol volume的最大 qtree 數量。	（預設不強制執行）
`lunsPerFlexvol`	每個 Flexvol 卷的最大 LUN 數必須在 [50, 200] 範圍內。僅限SAN。	“100”
`debugTraceFlags`	故障排除時要使用的調試標誌。例如，{"api":false, "method":true} 請勿使用 `debugTraceFlags`除非您正在進行故障排除並且需要詳細的日誌轉儲。	無效的
`nfsMountOptions`	以逗號分隔的 NFS 掛載選項清單。Kubernetes 持久性磁碟區的掛載選項通常在儲存類別中指定，但如果儲存類別中沒有指定掛載選項，Trident將回退到使用儲存後端設定檔中指定的掛載選項。如果在儲存類別或設定檔中未指定任何掛載選項， Trident將不會在關聯的持久性磁碟區上設定任何掛載選項。	“”
`nasType`	配置 NFS 或 SMB 磁碟區的建立。選項有 `nfs`， `smb`或空值。 *必須設定為 `smb`適用於 SMB 卷。 *設定為 null 則預設使用 NFS 磁碟區。	`nfs`
`qtreesPerFlexvol`	每個FlexVol volume的最大 Qtree 數量必須在 [50, 300] 範圍內	`"200"`
`smbShare`	您可以指定以下名稱之一：使用 Microsoft 管理主控台或ONTAP CLI 所建立的 SMB 共用的名稱，或允許Trident建立 SMB 共用的名稱。此參數是Amazon FSx for ONTAP後端所必需的。	`smb-share`
`useREST`	使用ONTAP REST API 的布林參數。設定為 `true`Trident將使用ONTAP REST API 與後端進行通訊。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。 `ontap`應用。預定義項滿足了這一點。 `vsadmin`和 `cluster-admin`角色。	`false`

範圍	描述	例子
aws	您可以在 AWS FSx for ONTAP的設定檔中指定以下內容： - 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 金鑰。	"" "" ""
credentials	指定要儲存在 AWS Secrets Manager 中的 FSx SVM 憑證。 - name ：包含 SVM 憑證的金鑰的 Amazon 資源名稱 (ARN)。 - type` 設定為 `awsarn。請參閱"建立 AWS Secrets Manager 金鑰"了解更多。	

用於配置磁碟區的後端配置選項

您可以使用下列選項控制預設配置。 `defaults`配置部分。例如，請參閱下面的設定範例。

範圍	描述	預設
spaceAllocation	LUN 的空間分配	true
spaceReserve	空間預留模式；「無」（細）或「大量」（粗）	none
snapshotPolicy	要使用的快照策略	none
qosPolicy	若要為建立的磁碟區指派的 QoS 策略群組。每個儲存池或後端選擇 qosPolicy 或 adaptiveQosPolicy 之一。將 QoS 策略群組與Trident結合使用需要ONTAP 9.8 或更高版本。您應該使用非共享的 QoS 策略群組，並確保該策略群組單獨套用至每個成員。共享的 QoS 策略群組強制規定所有工作負載的總吞吐量上限。	“”
adaptiveQosPolicy	若要為建立的磁碟區指派的自適應 QoS 策略群組。每個儲存池或後端選擇 qosPolicy 或 adaptiveQosPolicy 之一。 ontap-nas-economy 不支援此功能。	“”
snapshotReserve	為快照“0”預留的磁碟區百分比	如果 snapshotPolicy`是 `none ，else “”
splitOnClone	創建時將克隆體從其母體中分離出來	false

範圍	描述	預設
`encryption`	在新磁碟區啟用NetApp磁碟區加密(NVE)；預設為 `false`。若要使用此選項，必須在叢集上取得 NVE 許可並啟用 NVE。如果後端啟用了NAE，則在Trident中配置的任何磁碟區都會啟用 NAE。更多信息，請參閱："Trident如何與 NVE 和 NAE 協同工作" 。	`false`
`luksEncryption`	啟用LUKS加密。參考"使用 Linux 統一金鑰設定 (LUKS)"。僅限 SAN。	“”
`tieringPolicy`	分層策略的使用 `none`	
`unixPermissions`	新卷模式。 *SMB卷請留空。 *	“”
`securityStyle`	新磁碟區的安全樣式。 NFS 支持 `mixed`和 `unix`安全措施。中小企業支持 `mixed`和 `ntfs`安全措施。	NFS 預設值為 `unix`。 SMB 預設值為 `ntfs`。

準備配置SMB卷

您可以使用以下方式設定 SMB 磁碟區： `ontap-nas`司機。在你完成之前ONTAP SAN 和 NAS 驅動程式集成請完成以下步驟。

開始之前

在使用以下方式設定 SMB 磁碟區之前： `ontap-nas`駕駛員，您必須具備以下條件。

- 一個 Kubernetes 叢集，包含一個 Linux 控制器節點和至少一個執行 Windows Server 2019 的 Windows 工作節點。 Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。
- 至少有一個包含您的 Active Directory 憑證的Trident金鑰。生成秘密 `smbcreds`：

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

- 配置為 Windows 服務的 CSI 代理程式。要配置 `csi-proxy`，請參閱"GitHub：CSI代理"或者"GitHub：適用於 Windows 的 CSI 代理"適用於在 Windows 上執行的 Kubernetes 節點。

步驟

1. 建立SMB共享。您可以透過以下兩種方式之一建立 SMB 管理共用："Microsoft 管理控制台"共用資料夾管理單元或使用ONTAP CLI。使用ONTAP CLI 建立 SMB 共享：
   a. 如有必要，請建立共用的目錄路徑結構。

      這 `vserver cifs share create`此指令檢查在建立共用時 -path 選項中指定的路徑。如果指定的路徑不存在，則命令執行失敗。

   b. 建立與指定 SVM 關聯的 SMB 共用：

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

c. 確認共享已建立：

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

請參閱"建立 SMB 共享"了解詳細資訊。

2. 建立後端時，必須配置以下內容以指定 SMB 磁碟區。有關所有 FSx for ONTAP後端設定選項，請參閱"FSx for ONTAP設定選項和範例"。

範圍	描述	例子
smbShare	您可以指定以下名稱之一：使用 Microsoft 管理主控台或ONTAP CLI 所建立的 SMB 共用的名稱，或允許Trident建立 SMB 共用的名稱。此參數是Amazon FSx for ONTAP後端所必需的。	smb-share
nasType	*必須設定為 smb.*如果為空，則預設為空。 nfs 。	smb
securityStyle	新磁碟區的安全樣式。 *必須設定為 `ntfs`或者 `mixed`適用於 SMB 卷。 *	`ntfs`或者 `mixed`適用於 SMB 卷
unixPermissions	新卷模式。 *對於 SMB 卷，此項必須留空。 *	“”

配置儲存等級和**PVC**

配置 Kubernetes StorageClass 物件並建立儲存類，以指示Trident如何設定磁碟區。建立一個使用已設定的 Kubernetes StorageClass 的 PersistentVolumeClaim (PVC) 來要求存取 PV。然後您可以將光伏組件安裝到支架上。

建立儲存類別

設定 **Kubernetes StorageClass** 對象

這 "Kubernetes StorageClass 對象"該物件將Trident標識為該類別使用的配置器，並指示Trident如何配置磁碟區。使用此範例為使用 NFS 的磁碟區設定 Storageclass（有關屬性的完整列表，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"
```

使用此範例為使用 iSCSI 的磁碟區設定 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"
```

若要在 AWS Bottlerocket 上配置 NFSv3 卷，請新增所需內容。 `mountOptions`到儲存類別：

```
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
```

請參閱"Kubernetes 和Trident對象"有關存儲類如何與…交互的詳細信息 `PersistentVolumeClaim`以及控制Trident如何分配容量的參數。

建立儲存類別

步驟

1. 這是一個 Kubernetes 對象，所以請使用 `kubectl`在 Kubernetes 中創建它。

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

2. 現在您應該在 Kubernetes 和Trident中都看到 **basic-csi** 儲存類，並且Trident應該已經發現了後端上的儲存池。

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

製作**PVC**管

一個 "*PersistentVolumeClaim*"（PVC）是對叢集上持久卷的存取請求。

PVC 可以配置為請求儲存特定尺寸或存取模式。使用關聯的 StorageClass，叢集管理員不僅可以控制持久磁碟區的大小和存取模式，還可以控制效能或服務等級。

製作好 PVC 後，就可以將容積安裝到艙體中。

樣品清單

PersistentVolumeClaim 樣本清單

這些範例展示了PVC的基本配置選項。

附**RWX**接口的**PVC**

此範例展示了一個具有 RWX 存取權限的基本 PVC，它與一個名為 StorageClass 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**

此範例展示了一個與名為 StorageClass 的儲存類別關聯的、具有 RWO 存取權限的 iSCSI 基本 PVC。 `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. 核實PVC狀態。

```
kubectl get pvc
```

```
NAME          STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi       RWO                          5m
```

請參閱"Kubernetes 和Trident對象"有關存儲類如何與…交互的詳細信息 `PersistentVolumeClaim`以及控制Trident如何分配容量的參數。

Trident屬性

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

屬性	類型	價值觀	提供	要求	由...支持
媒體[1]	細繩	機械式硬碟、混合式硬碟、固態硬碟	Pool 包含此類媒體；混合型媒體是指兩者兼具。	指定的媒體類型	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、solidfire-san
供應類型	細繩	薄的，厚的	池支援這種配置方法	指定的配置方法	厚：全部 ontap；薄：全部 ontap 和 solidfire-san
後端類型	細繩	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、solidfire-san、gcp-cvs、azure-netapp-files、ontap-san-economy	池屬於這種類型的後端	指定的後端	所有司機
快照	布林值	真，假	儲存池支援帶快照的磁碟區	已啟用快照的磁碟區	ontap-nas、ontap-san、solidfire-san、gcp-cvs
複製	布林值	真，假	儲存池支援磁碟區克隆	已啟用克隆的磁碟區	ontap-nas、ontap-san、solidfire-san、gcp-cvs

屬性	類型	價值觀	提供	要求	由...支持
加密	布林值	真，假	儲存池支援加密磁碟區	已啟用加密的磁碟區	ontap-nas、ontap-nas-economy、ontap-nas-flexgroups、ontap-san
每秒輸入/輸出次數	整數	正整數	Pool 能夠保證在此範圍內的 IOPS	容量保證了這些IOPS	solidfire-san

[1]： ONTAP Select系統不支援此系統

部署範例應用程式

建立儲存等級和 PVC 後，即可將光電模組安裝到艙體上。本節列出了將 PV 連接到 pod 的範例命令和設定。

步驟

1. 將音量旋鈕安裝在一個音控器中。

```
kubectl create -f pv-pod.yaml
```

以下範例展示了將PVC連接到艙體的基本配置：基本配置：

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

您可以使用以下方式監控進度 `kubectl get pod --watch`。

2. 確認卷已掛載到 `/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
```

現在您可以刪除該 Pod 了。 Pod 應用將不復存在，但卷將保留。

```
kubectl delete pod pv-pod
```

在 **EKS** 叢集上設定**Trident EKS** 插件

NetApp 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專注於應用程式部署。 NetApp Trident EKS 外掛程式包含最新的安全性修補程式和錯誤修復，並經過 AWS 驗證，可與 Amazon EKS 搭配使用。 EKS 外掛程式可讓您持續確保 Amazon EKS 叢集的安全性和穩定性，並減少安裝、設定和更新外掛程式所需的工作量。

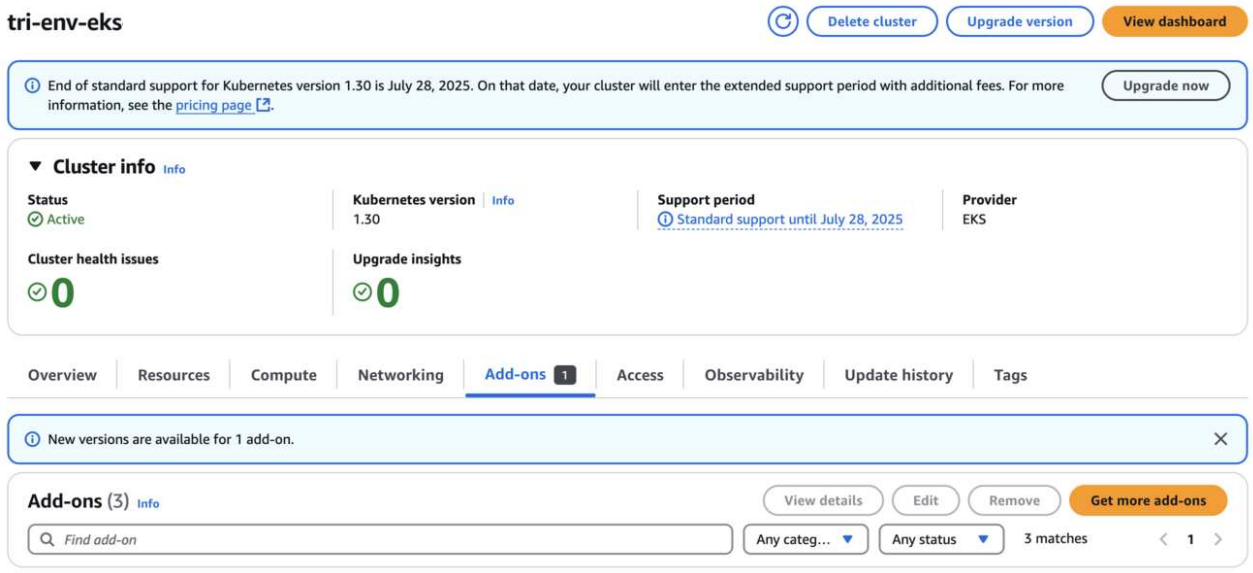
先決條件

在為 AWS EKS 設定Trident外掛程式之前，請確保您已具備以下條件：

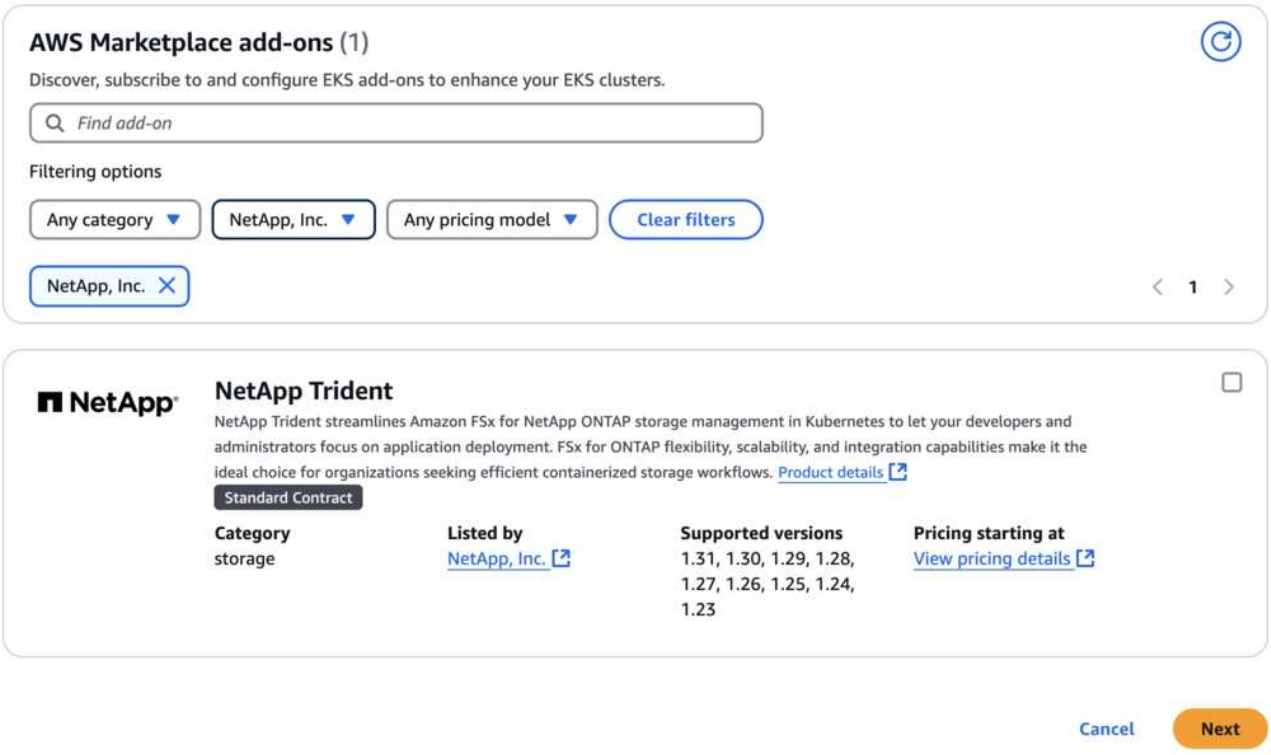
- 具有使用插件權限的 Amazon EKS 叢集帳戶。參考"Amazon EKS 附加元件"。
- AWS 對 AWS Marketplace 的權限：
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 類型：Amazon Linux 2 (AL2_x86_64) 或 Amazon Linux 2 Arm (AL2_ARM_64)
- 節點類型：AMD 或ARM
- 現有的Amazon FSx for NetApp ONTAP檔案系統

步驟

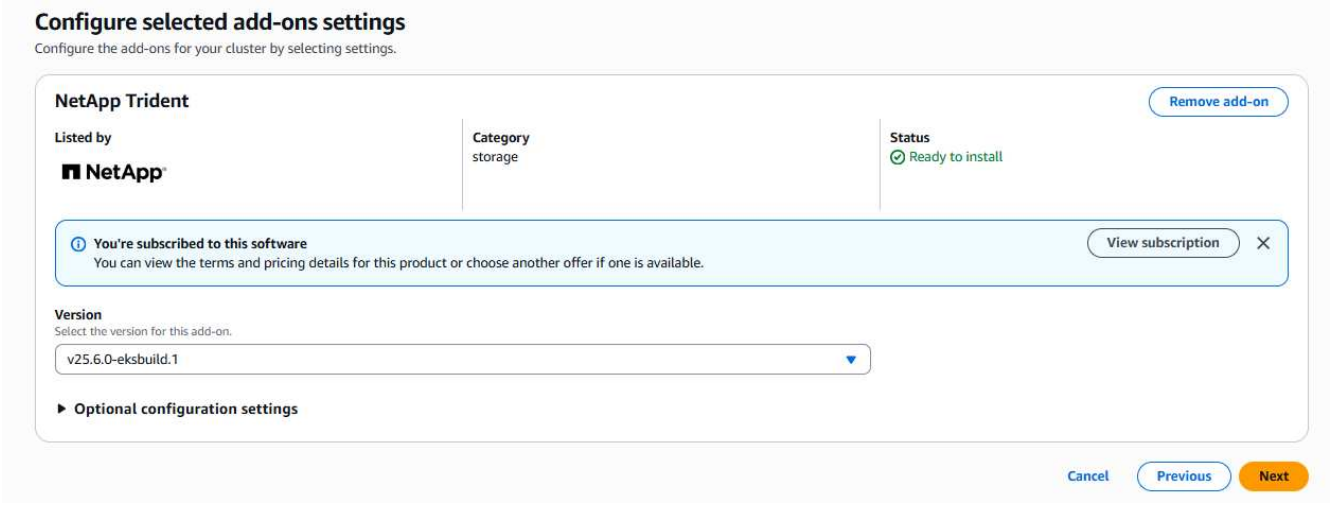
1. 請務必建立 IAM 角色和 AWS 金鑰，以使 EKS pod 能夠存取 AWS 資源。有關說明，請參閱"建立 IAM 角色和 AWS Secret"。
2. 在 EKS Kubernetes 叢集上，導覽至「附加元件」標籤。



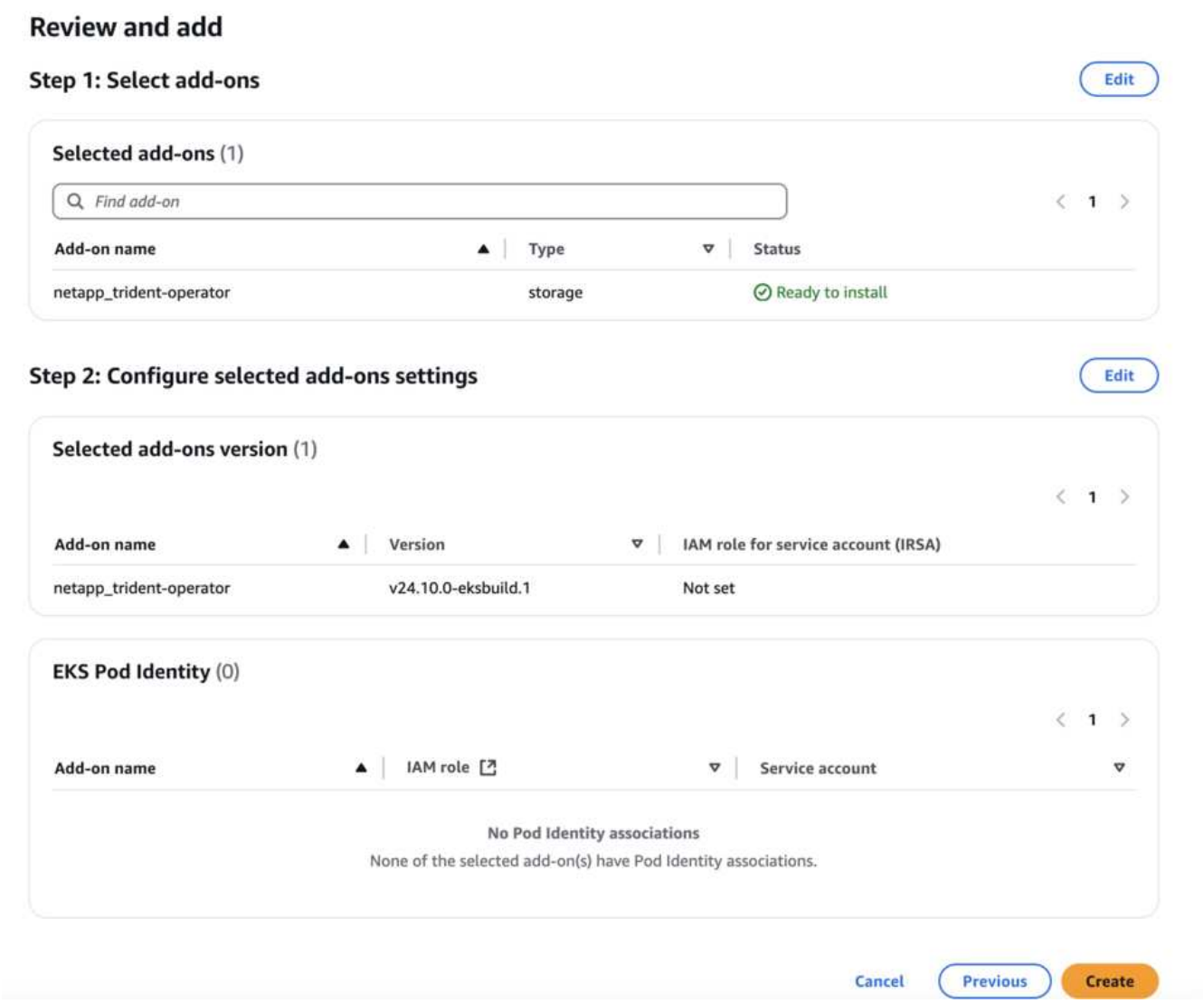
3. 前往 **AWS Marketplace** 外掛程式，然後選擇 *storage* 類別。



4. 找到 * NetApp Trident*，選取Trident程式的複選框，然後按一下 下一步。
5. 選擇所需的插件版本。

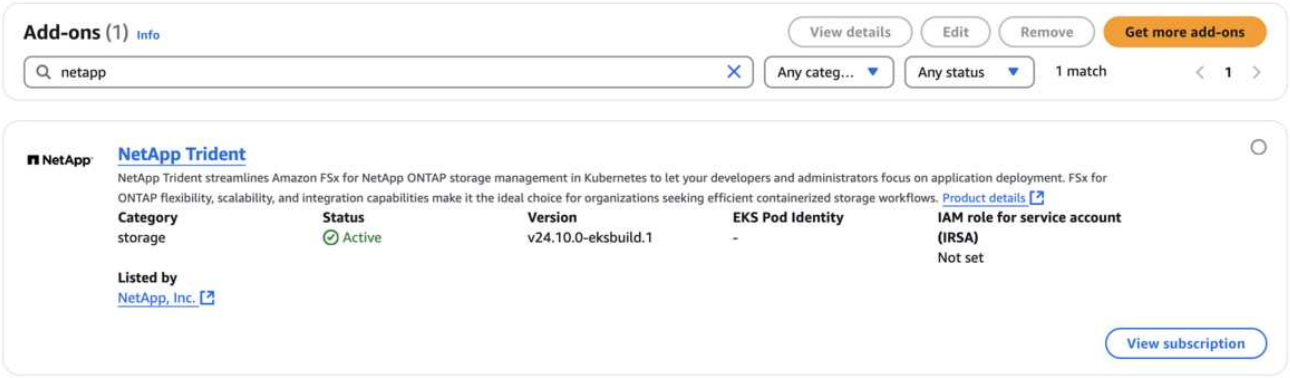


6. 配置所需的附加元件設定。



7. 如果您使用的是 IRSA（服務帳戶的 IAM 角色），請參閱其他設定步驟。"這裡" 。
8. 選擇“創建”。

9. 確認插件狀態為_Active_。



10. 執行以下命令以驗證Trident是否已正確安裝在叢集上：

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

11. 繼續進行設定並配置儲存後端。有關信息，請參閱"配置儲存後端"。

使用 **CLI** 安裝/解除安裝**Trident EKS** 插件

使用 **CLI** 安裝**NetApp Trident EKS** 外掛：

以下範例指令安裝Trident EKS 外掛：

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.6.0-eksbuild.1 （另有專用版本）
```

使用 **CLI** 卸載**NetApp Trident EKS** 外掛程式：

以下指令卸載Trident EKS 外掛程式：

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

使用 **kubectl** 建立後端

後端定義了Trident與儲存系統之間的關係。它告訴Trident如何與該儲存系統通信，以及Trident應該如何從中設定磁碟區。 Trident安裝完成後，下一步是建立後端。這 `TridentBackendConfig`自訂資源定義 (CRD) 可讓您透過 Kubernetes 介面直接建立和管理Trident後端。您可以使用 `kubectl`或適用於您的 Kubernetes 發行版的等效 CLI 工具。

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`， `tbconfig` ， `tbackendconfig`) 是一個前端、命名空間 CRD，讓您能夠使用 來管理Trident後端 `kubectl`。現在，Kubernetes 和儲存管理員可以直接透過 Kubernetes CLI 建立和管理後端，而無需使用專門的命令列實用程式。(`tridentctl` ） 。

在創建之後 `TridentBackendConfig`對於該對象，會發生以下情況：

- Trident會根據您提供的設定自動建立後端。這在內部表示為 `TridentBackend` (`tbe`， `tridentbackend`) CR。
- 這 `TridentBackendConfig`與…有著獨特的聯繫 `TridentBackend`這是由Trident生產的。

每個 `TridentBackendConfig`與…保持一對一映射關係 `TridentBackend`前者是提供給使用者設計和配置後端的介面；後者是Trident表示實際後端物件的方式。

> `TridentBackend`CR 由Trident自動建立。你*不應該*修改它們。如果您想對後端進行更新，請透過修改以下檔案來實現： `TridentBackendConfig`目的。

以下範例展示了格式： `TridentBackendConfig` CR：

```
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
```

您也可以查看以下範例： "trident-installer"包含所需儲存平台/服務的範例配置的目錄。

這 `spec`接受後端特定的配置參數。在這個例子中，後端使用了 `ontap-san`儲存驅動程序，並使用此處表格中列出的配置參數。有關所需儲存驅動程式的配置選項列表，請參閱"儲存驅動程式的後端配置訊息"。

這 `spec`本節還包括 `credentials`和 `deletionPolicy`這些字段是新引入的 `TridentBackendConfig`CR：

- `credentials`此參數為必填字段，包含用於向儲存系統/服務進行身份驗證的憑證。這設定為用戶創建的 Kubernetes Secret。憑證不能以明文形式傳遞，否則將導致錯誤。
- `deletionPolicy`此欄位定義了當…時應該發生的情況 `TridentBackendConfig`被刪除。它可以取以下兩個值之一：
  - `delete`這會導致兩者被刪除。 `TridentBackendConfig` CR及其相關後端。這是預設值。
  - `retain`當 `TridentBackendConfig`CR 刪除後，後端定義仍然存在，並且可以透過以下方式進行管理： `tridentctl` 。將刪除策略設為 `retain`允許使用者降級到早期版本（21.04 之前），並保留已建立的後端。該欄位的值可以在之後更新。 `TridentBackendConfig`已創建。

> 後端名稱是透過以下方式設定的： `spec.backendName` 。如果未指定，則後端名稱設定為…的名稱 `TridentBackendConfig`對象（`metadata.name`）。建議使用顯式方式設定後端名稱。 `spec.backendName` 。

用以下方式建立的後端 `tridentctl`沒有關聯的 `TridentBackendConfig`目的。您可以選擇使用以下方式管理此類後端 `kubectl`透過創建一個 `TridentBackendConfig`CR。必須注意指定完全相同的配置參數（例如： `spec.backendName ， spec.storagePrefix ， spec.storageDriverName ， 等等）。 Trident將自動綁定新建立的`TridentBackendConfig`利用現有的後端。

步驟概述

使用以下方式建立新的後端 `kubectl`你應該這樣做：

1. 創建一個 "Kubernetes Secret"此金鑰包含Trident與儲存叢集/服務通訊所需的憑證。
2. 創建一個 `TridentBackendConfig`目的。這包含有關儲存叢集/服務的具體信息，並引用上一步中建立的密鑰。

建立後端後，您可以使用以下方式觀察其狀態 `kubectl get tbc <tbc-name> -n <trident-namespace>`並收集更多細節資訊。

步驟 1：建立 Kubernetes Secret

建立一個包含後端存取憑證的金鑰。這是每個儲存服務/平台獨有的。以下是一個例子：

```
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
```

下表總結了每個儲存平台金鑰中必須包含的欄位：

儲存平台密鑰字段描述	秘密	字段描述
Azure NetApp Files	客戶端ID	應用程式註冊中的客戶端 ID
適用於 GCP 的 Cloud Volumes Service	私鑰 ID	私鑰的ID。具有 CVS 管理員角色的 GCP 服務帳戶的部分 API 金鑰
適用於 GCP 的 Cloud Volumes Service	私鑰	私鑰。具有 CVS 管理員角色的 GCP 服務帳戶的部分 API 金鑰

儲存平台密鑰字段描述	秘密	字段描述
元素（NetApp HCI/ SolidFire）	端點	針對SolidFire叢集的 MVIP，包含租戶憑證
ONTAP	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
ONTAP	密碼	連接到叢集/SVM 的密碼。用於基於憑證的身份驗證
ONTAP	客戶端私鑰	客戶端私鑰的 Base64 編碼值。用於基於憑證的身份驗證
ONTAP	chap用戶名	入站用戶名。如果 useCHAP=true，則為必填項。為了 `ontap-san`和 `ontap-san-economy`
ONTAP	chapInitiatorSecret	CHAP 發起者金鑰。如果 useCHAP=true，則此項目為必填項。為了 `ontap-san`和 `ontap-san-economy`
ONTAP	chapTargetUsername	目標用戶名。如果 useCHAP=true，則此項目為必填項。為了 `ontap-san`和 `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	CHAP 目標發起者金鑰。如果 useCHAP=true，則此項目為必填項。為了 `ontap-san`和 `ontap-san-economy`

此步驟中建立的秘密將在以下位置引用： `spec.credentials`領域的 `TridentBackendConfig`在下一步建立的物件。

步驟 2：建立 `TridentBackendConfig`CR

現在您可以開始創建您的 `TridentBackendConfig`CR。在這個例子中，後端使用了 `ontap-san`驅動程式是透過使用以下方式建立的： `TridentBackendConfig`下圖所示物體：

```
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：驗證狀態 `TridentBackendConfig`CR

現在你已經創建了 `TridentBackendConfig`CR，您可以核實狀態。請參閱以下範例：

```
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
```

後端已成功建立並綁定到 `TridentBackendConfig`CR。

相位可以取以下值之一：

- `Bound`： 這 `TridentBackendConfig`CR 與後端關聯，此後端包含 `configRef`設定為 `TridentBackendConfig`CR 的 uid。
- `Unbound`：用以下方式表示 ""。這 `TridentBackendConfig`該物件未綁定到後端。所有新創建 `TridentBackendConfig`CR 預設處於此階段。階段變更後，它無法再恢復到未綁定狀態。
- `Deleting`： 這 `TridentBackendConfig`CR的 `deletionPolicy`已設定為刪除。當 `TridentBackendConfig`CR 刪除後，狀態變成正在刪除。
  - 如果後端不存在持久性磁碟區宣告 (PVC)，則刪除 `TridentBackendConfig`這將導致Trident刪除後端以及 `TridentBackendConfig`CR。
  - 如果後端存在一個或多個 PVC，則進入刪除狀態。這 `TridentBackendConfig`CR隨後也進入刪除階段。後端和 `TridentBackendConfig`只有在所有 PVC 都被刪除後才會刪除。
- `Lost`與後端相關的 `TridentBackendConfig`CR 被意外或故意刪除，並且 `TridentBackendConfig`CR 仍然保留著對已刪除後端的引用。這 `TridentBackendConfig`無論如何，CR 仍然可以刪除。 `deletionPolicy`價值。
- `Unknown`Trident無法確定與下列系統相關的後端的狀態或是否存在： `TridentBackendConfig` CR。例如，如果 API 伺服器沒有回應，或者如果 `tridentbackends.trident.netapp.io`CRD 缺失。這可能需要幹預。

至此，後端已成功創建！此外，還可以處理以下幾種操作："後端更新和後端刪除" 。

（可選）步驟 **4**：了解更多詳情

您可以執行以下命令來獲取有關後端的更多資訊：

```
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，您還可以獲得 YAML/JSON 轉儲檔案。 `TridentBackendConfig` 。

```
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 `backendName`以及 `backendUUID`後端是根據以下情況創建的：`TridentBackendConfig CR。這 lastOperationStatus`此欄位表示上次操作的狀態`TridentBackendConfig`CR（變更要求）可以由使用者觸發（例如，使用者變更了某些內容）。 `spec）或由Trident觸發（例如，在Trident重啟期間）。結果要么是成功，要么是失敗。 phase`代表了以下關係的狀態： `TridentBackendConfig CR 和後端。在上面的例子中，`phase`具有 Bound 值，這意味著`TridentBackendConfig`CR與後端相關。

您可以運行 `kubectl -n trident describe tbc <tbc-cr-name>`取得事件日誌詳細資訊的命令。

> 您無法更新或刪除包含關聯後端的後端。 TridentBackendConfig`使用物件`tridentctl。要了解在以下兩者之間切換所涉及的步驟： tridentctl`和`TridentBackendConfig，"請參閱此處" 。

管理後端

使用 **kubectl** 執行後端管理

了解如何使用以下方式執行後端管理操作 kubectl。

刪除後端

透過刪除一個 TridentBackendConfig`您指示Trident刪除/保留後端（基於 `deletionPolicy）。若要刪除後端，請確保 deletionPolicy`已設定為刪除。僅刪除 `TridentBackendConfig`確保 `deletionPolicy`設定為保留。這樣可以確保後端仍然存在，並且可以透過以下方式進行管理：`tridentctl 。

運行以下命令：

```
kubectl delete tbc <tbc-name> -n trident
```

Trident不會刪除正在使用的 Kubernetes Secret。 TridentBackendConfig 。 Kubernetes 用戶負責清理金鑰。刪除機密資訊時務必謹慎。只有當後端不再使用密鑰時，才應該刪除密鑰。

查看現有後端

運行以下命令：

```
kubectl get tbc -n trident
```

你也可以運行 tridentctl get backend -n trident`或者 `tridentctl get backend -o yaml -n trident`取得所有現有後端的清單。此清單還將包括使用以下方式建立的後端： `tridentctl 。

更新後端

更新後端的原因可能有很多：

- 儲存系統的憑證已更改。要更新憑證，需要更新 Kubernetes Secret，該 Secret 用於：`TridentBackendConfig`對象必須更新。 Trident會自動使用提供的最新憑證更新後端。執行以下命令更新 Kubernetes Secret：

  ```
  kubectl apply -f <updated-secret-file.yaml> -n trident
  ```

- 需要更新參數（例如正在使用的ONTAP SVM 的名稱）。
  - 您可以更新 `TridentBackendConfig`使用以下命令直接透過 Kubernetes 存取物件：

    ```
    kubectl apply -f <updated-backend-file.yaml>
    ```

  - 或者，您可以對現有內容進行變更。 `TridentBackendConfig`使用以下命令進行回車：

```
kubectl edit tbc <tbc-name> -n trident
```

- 如果後端更新失敗，後端將繼續保持其最後一次已知的配置。您可以透過執行以下命令查看日誌以確定原因。 `kubectl get tbc <tbc-name> -o yaml -n trident`或者 `kubectl describe tbc <tbc-name> -n trident`。
- 在您發現並修正設定檔中的問題後，您可以重新執行更新命令。

使用 **tridentctl** 執行後端管理

了解如何使用以下方式執行後端管理操作 `tridentctl`。

創建後端

創建之後"後端設定檔"運行以下命令：

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

如果後端建立失敗，則表示後端配置存在問題。您可以透過執行以下命令查看日誌以確定原因：

```
tridentctl logs -n trident
```

在您發現並修正設定檔中的問題後，您只需執行以下命令即可。 `create`再次發出命令。

刪除後端

若要從Trident中刪除後端，請執行下列操作：

1. 取得後端名稱：

```
tridentctl get backend -n trident
```

2. 刪除後端：

```
tridentctl delete backend <backend-name> -n trident
```

如果Trident已從從後端配置了仍然存在的磁碟區和快照，則刪除後端將阻止從該後端設定新磁碟區。後端將繼續處於「刪除」狀態。

查看現有後端

若要查看Trident已知的後端，請執行下列操作：

- 若要取得摘要，請執行以下命令：

```
tridentctl get backend -n trident
```

- 要獲取所有詳細信息，請運行以下命令：

```
tridentctl get backend -o json -n trident
```

更新後端

建立新的後端設定檔後，執行以下命令：

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

如果後端更新失敗，則表示後端配置有問題，或者您嘗試了無效的更新。您可以透過執行以下命令查看日誌以確定原因：

```
tridentctl logs -n trident
```

在您發現並修正設定檔中的問題後，您只需執行以下命令即可。 `update`再次發出命令。

確定使用後端儲存的儲存類

這是一個您可以使用 JSON 回答的問題範例： `tridentctl`後端物件的輸出。這使用 `jq`您需要安裝該實用程式。

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

這也適用於使用以下方式建立的後端： `TridentBackendConfig` 。

在後端管理選項之間切換

了解Trident中管理後端的不同方法。

後端管理選項

隨著 `TridentBackendConfig`現在，管理員有兩種獨特的後端管理方式。這就引出了以下問題：

- 可以使用以下方式建立後端 `tridentctl`**以……進行管理** `TridentBackendConfig`？
- 可以使用以下方式建立後端 `TridentBackendConfig`**可透過以下方式進行管理** `tridentctl`？

管理 tridentctl`使用後端 `TridentBackendConfig

本節介紹管理使用以下方式建立的後端所需的步驟： `tridentctl`直接透過 Kubernetes 介面創建 `TridentBackendConfig`物體。

這適用於以下情況：

- 預先存在的後端，沒有 TridentBackendConfig`因為它們是用…創造的 `tridentctl。
- 使用以下方式建立的新後端 `tridentctl`而其他 `TridentBackendConfig`物體是存在的。

在這兩種情況下，後端都將繼續存在， Trident將調度磁碟區並對其進行操作。管理員此時有兩種選擇：

- 繼續使用 `tridentctl`管理使用它創建的後端。
- 使用以下方式建立的綁定後端 tridentctl`到一個新的 `TridentBackendConfig`目的。這樣做意味著後端將使用以下方式進行管理： `kubectl`而不是 `tridentctl。

使用以下方式管理預先存在的後端 `kubectl`您需要建立一個 `TridentBackendConfig`它與現有後端綁定。以下是其工作原理概述：

1. 創建 Kubernetes Secret。此金鑰包含Trident與儲存叢集/服務通訊所需的憑證。
2. 創建一個 TridentBackendConfig`目的。這包含有關儲存叢集/服務的具體信息，並引用上一步中建立的密鑰。必須注意指定完全相同的配置參數（例如： `spec.backendName ， spec.storagePrefix ， spec.storageDriverName ， 等等）。 `spec.backendName`必須設定為現有後端的名稱。

步驟 0：確定後端

創建一個 `TridentBackendConfig`如果要綁定到現有後端，則需要取得後端配置。在這個例子中，我們假設使用以下 JSON 定義建立了一個後端：

```
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 Secret**

建立一個包含後端憑證的 Secret，如下例所示：

```
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**

下一步是創建一個 `TridentBackendConfig`CR` 將自動綁定到預先存在的 `ontap-nas-backend（如本例所示）。請確保滿足以下要求：

- 後端名稱在以下位置定義： `spec.backendName` 。
- 配置參數與原後端相同。
- 虛擬池（如果存在）必須保持與原始後端相同的順序。
- 憑證透過 Kubernetes Secret 提供，而不是以明文形式提供。

在這種情況下， `TridentBackendConfig`將會像這樣：

```
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：驗證狀態 `TridentBackendConfig`CR

之後 `TridentBackendConfig`已經創建，它的階段必須是 `Bound`。它也應該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 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

後端現在將完全使用以下方式進行管理： `tbc-ontap-nas-backend` `TridentBackendConfig`目的。

管理 `TridentBackendConfig`使用後端 `tridentctl`

`tridentctl`可用於列出使用下列方式建立的後端： `TridentBackendConfig` 。此外，管理員還可以選擇透過以下方式完全管理此類後端： `tridentctl`透過刪除 `TridentBackendConfig`並確保 `spec.deletionPolicy`設定為 `retain`。

步驟 **0**：確定後端

例如，假設我們使用以下方式建立了以下後端 `TridentBackendConfig`：

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

從輸出結果可以看出：`TridentBackendConfig`已成功建立並綁定到後端[觀察後端的 UUID]。

步驟 1：確認 `deletionPolicy`**設定為** `retain`

讓我們來看看它的價值 `deletionPolicy`。需要將其設定為 `retain`。這確保了當 `TridentBackendConfig`CR 刪除後，後端定義仍然存在，並且可以透過以下方式進行管理：`tridentctl`。

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

除非另有說明，否則請勿進行下一步。 `deletionPolicy`設定為 `retain`。

步驟二：刪除 `TridentBackendConfig`CR

最後一步是刪除 `TridentBackendConfig`CR。確認後 `deletionPolicy`設定為 `retain`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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

刪除後 `TridentBackendConfig`Trident只是移除該對象，而不會實際刪除後端本身。

建立和管理儲存類

建立儲存類別

配置 Kubernetes StorageClass 物件並建立儲存類，以指示Trident如何設定磁碟區。

設定 **Kubernetes StorageClass** 對象

這 "Kubernetes StorageClass 對象"識別Trident作為該類別使用的配置器，並指示Trident如何配置磁碟區。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
```

請參閱"Kubernetes 和Trident對象"有關存儲類如何與...交互的詳細信息 `PersistentVolumeClaim`以及控制Trident如何分配容量的參數。

建立儲存類別

建立 StorageClass 物件後，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. 這是一個 Kubernetes 對象，所以請使用 `kubectl`在 Kubernetes 中創建它。

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. 現在您應該在 Kubernetes 和Trident中都看到 **basic-csi** 儲存類，並且Trident應該已經發現了後端上的儲存池。

```
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 "針對特定後端的簡單儲存類別定義"。

或者，您可以編輯 `sample-input/storage-class-csi.yaml.templ`安裝程式隨附的檔案並替換 `*BACKEND_TYPE*` 使用儲存驅動程式名稱。

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

管理儲存類別

您可以查看現有儲存類別、設定預設儲存類別、識別儲存類別後端以及刪除儲存類別。

查看現有儲存類

- 若要查看現有的 Kubernetes 儲存類，請執行下列命令：

```
kubectl get storageclass
```

- 要查看 Kubernetes 儲存類別詳細信息，請執行以下命令：

```
kubectl get storageclass <storage-class> -o json
```

- 若要查看 Trident 的同步儲存類，請執行下列命令：

```
tridentctl get storageclass
```

- 要查看 Trident 的同步儲存類別詳細信息，請執行以下命令：

```
tridentctl get storageclass <storage-class> -o json
```

設定預設儲存類

Kubernetes 1.6 增加了設定預設儲存類別的功能。如果使用者未在持久性磁碟區宣告 (PVC) 中指定持久卷，則將使用此儲存類別來設定持久性磁碟區。

- 透過設定註解來定義預設儲存類 `storageclass.kubernetes.io/is-default-class`在儲存類別定義中設定為 true。根據規範，任何其他值或缺少註釋均被解釋為錯誤。
- 您可以使用下列命令將現有儲存類別配置為預設儲存類別：

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- 同樣，您可以使用以下命令移除預設的儲存類別註解：

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Trident安裝程式包中也有一些包含此註解的範例。

叢集中一次只能存在一個預設儲存類別。從技術上講，Kubernetes 不會阻止你擁有多個預設儲存類，但它的行為就好像根本沒有預設儲存類別一樣。

確定儲存類別的後端

這是一個您可以使用 JSON 回答的問題範例： `tridentctl` Trident後端物件的輸出。這使用 `jq`您可能需要先安裝該實用程式。

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

刪除儲存類

若要從 Kubernetes 中刪除儲存類，請執行以下命令：

```
kubectl delete storageclass <storage-class>
```

`<storage-class>`應該替換成你的儲存類別。

透過此儲存類別建立的任何持久性磁碟區都將保持不變， Trident將繼續管理它們。

Trident強制執行空白 `fsType`因為它創造了大量的銷售。對於 iSCSI 後端，建議強制執行 `parameters.fsType`在儲存類別中。您應該刪除現有的 StorageClasses 並重新建立它們。 `parameters.fsType`指定的。

配置和管理卷

提供一定量

建立一個使用已設定的 Kubernetes StorageClass 的 PersistentVolumeClaim (PVC) 來要求存取 PV。然後您可以將光伏組件安裝到支架上。

概況

一個 "*PersistentVolumeClaim*"（PVC）是對叢集上持久卷的存取請求。

PVC 可以配置為請求儲存特定尺寸或存取模式。使用關聯的 StorageClass，叢集管理員不僅可以控制持久磁碟區的大小和存取模式，還可以控制效能或服務等級。

製作好PVC管後，就可以將管體安裝到管座中。

製作**PVC**管

步驟

1. 建立 PVC。

```
kubectl create -f pvc.yaml
```

2. 核實PVC狀態。

```
kubectl get pvc
```

```
NAME          STATUS   VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage Bound    pv-name     1Gi        RWO                    5m
```

1. 將音量旋鈕安裝在一個音控器中。

```
kubectl create -f pv-pod.yaml
```

> 您可以使用以下方式監控進度 `kubectl get pod --watch`。

2. 確認卷已掛載到 `/my/mount/path`。

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

3. 現在您可以刪除該 Pod 了。 Pod 應用將不復存在，但卷將保留。

```
kubectl delete pod pv-pod
```

樣品清單

PersistentVolumeClaim 樣本清單

這些範例展示了PVC的基本配置選項。

PVC管材，附RWO通道

此範例展示了一個具有 RWO 存取權限的基本 PVC，它與一個名為 StorageClass 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
```

PVC 和 NVMe/TCP

此範例展示了一個與名為 StorageClass 的 StorageClass 關聯的、具有 RWO 存取權限的 NVMe/TCP 基本 PVC。 `protection-gold` 。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

Pod清單範例

這些範例展示了將 PVC 連接到艙體的基本配置。

基本配置

```
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** 配置

```
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
```

請參閱"Kubernetes 和Trident對象"有關存儲類如何與…交互的詳細信息 `PersistentVolumeClaim`以及控制Trident如何分配容量的參數。

擴大銷量

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。

擴充 **iSCSI** 卷

您可以使用 CSI 設定程式擴充 iSCSI 持久性磁碟區 (PV)。

> ⓘ iSCSI 磁碟區擴充受以下方式支援： `ontap-san` ， `ontap-san-economy` ， `solidfire-san`驅動程式需要 Kubernetes 1.16 及更高版本。

步驟 **1**：設定 **StorageClass** 以支援磁碟區擴展

編輯 StorageClass 定義以進行設置 `allowVolumeExpansion`田野 `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，對其進行編輯以包含以下內容： `allowVolumeExpansion`範圍。

步驟 **2**：使用您建立的 **StorageClass** 建立 **PVC**。

編輯PVC定義並更新 `spec.resources.requests.storage`為了反映新的所需尺寸，該尺寸必須大於原始尺寸。

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident創建一個持久銷售 (PV) 並將其與此持久銷售聲明 (PVC) 關聯起來。

```
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 **PVC** 的艙體

將光電模組連接到艙體上，以便調整其尺寸。調整 iSCSI PV 大小時有兩種情況：

- 如果 PV 連接到 pod， Trident會擴展儲存後端上的捲，重新掃描設備，並調整檔案系統的大小。
- 嘗試調整未連接的 PV 的大小時， Trident會在儲存後端擴充磁碟區。 PVC 與 pod 綁定後， Trident會重新掃描裝置並調整檔案系統的大小。 Kubernetes 會在擴充操作成功完成後更新 PVC 大小。

在這個例子中，建立了一個使用下列功能的 pod： `san-pvc` 。

```
 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**

若要將已建立的 PV 從 1Gi 調整為 2Gi，請編輯 PVC 定義並更新。 `spec.resources.requests.storage`至 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**：驗證擴展

您可以檢查PVC、PV和Trident的體積來驗證擴容是否正確：

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

擴大 **FC** 體積

您可以使用 CSI 設定程式擴充 FC 持久性磁碟區 (PV)。

FC體積擴張得到了以下的支持： `ontap-san`驅動程式需要 Kubernetes 1.16 及更高版本。

步驟 **1**：設定 **StorageClass** 以支援磁碟區擴展

編輯 StorageClass 定義以進行設置 `allowVolumeExpansion`田野 `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，對其進行編輯以包含以下內容： `allowVolumeExpansion`範圍。

步驟 2：使用您建立的 **StorageClass** 建立 **PVC**。

編輯PVC定義並更新 `spec.resources.requests.storage`為了反映新的所需尺寸，該尺寸必須大於原始尺寸。

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident創建一個持久銷售 (PV) 並將其與此持久銷售聲明 (PVC) 關聯起來。

```
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 **PVC** 的艙體

將光電模組連接到艙體上，以便調整其尺寸。調整燃料電池光電模組容量時有兩種情況：

- 如果 PV 連接到 pod， Trident會擴展儲存後端上的捲，重新掃描設備，並調整檔案系統的大小。
- 嘗試調整未連接的 PV 的大小時， Trident會在儲存後端擴充磁碟區。 PVC 與 pod 綁定後， Trident會重新掃描裝置並調整檔案系統的大小。 Kubernetes 會在擴充操作成功完成後更新 PVC 大小。

在這個例子中，建立了一個使用下列功能的 pod： `san-pvc` 。

```
 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**

若要將已建立的 PV 從 1Gi 調整為 2Gi，請編輯 PVC 定義並更新。 `spec.resources.requests.storage`至 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**：驗證擴展

您可以檢查PVC、PV和Trident的體積來驗證擴容是否正確：

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

擴展 NFS 卷

Trident支援為已設定的 NFS PV 擴充容量。 `ontap-nas` ， `ontap-nas-economy` ， `ontap-nas-flexgroup` ， `gcp-cvs` ， 和 `azure-netapp-files`後端。

步驟 1：設定 StorageClass 以支援磁碟區擴展

要調整 NFS PV 的大小，管理員首先需要配置儲存類別以允許磁碟區擴展，方法是設定以下參數： `allowVolumeExpansion`田野 `true`：

```
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
```

如果您已經建立了一個沒有此選項的儲存類，則可以透過使用下列命令直接編輯現有儲存類別： `kubectl edit storageclass`允許體積膨脹。

步驟 2：使用您建立的 **StorageClass** 建立 **PVC**。

```
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：擴充 **PV**

若要將新建的 20 MiB PV 調整為 1 GiB，請編輯 PVC 並進行設置 `spec.resources.requests.storage`到 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**：驗證擴展

您可以檢查 PVC、PV 和Trident體積的大小來驗證調整大小是否正確：

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

進口量

您可以使用以下方式將現有儲存磁碟區匯入為 Kubernetes PV： `tridentctl import` 。

概述和注意事項

您可以將磁碟區匯入Trident以執行下列操作：

- 將應用程式容器化並重複使用其現有資料集
- 為臨時應用程式使用資料集的克隆版本
- 重建失敗的 Kubernetes 集群
- 在災難復原期間遷移應用程式數據

注意事項

在匯入磁碟區之前，請考慮以下事項。

- Trident只能匯入 RW（讀寫）類型的ONTAP區。 DP（資料保護）類型磁碟區是SnapMirror目標磁碟區。在將磁碟區導入Trident之前，應該先斷開鏡像關係。

- 我們建議匯入沒有活動連結的磁碟區。若要匯入正在使用的捲，請複製該卷，然後執行匯入操作。

  這對於塊卷來說尤其重要，因為 Kubernetes 不知道先前的連接，很容易將活動卷附加到 pod 上。這可能導致資料損壞。

- 儘管 `StorageClass`必須在 PVC 上指定， Trident在導入過程中不使用此參數。在建立磁碟區時，會使用儲存類別根據儲存特性從可用儲存池中進行選擇。由於磁碟區已存在，因此匯入時無需選擇儲存池。因此，即使磁碟區存在於與 PVC 中指定的儲存類別不符的後端或池中，匯入也不會失敗。
- 現有容積尺寸在PVC中確定並設定。磁碟區被儲存驅動程式匯入後，PV 會創建，並帶有指向 PVC 的 ClaimRef。
  - 回收策略初始設定為 `retain`在PV中。 Kubernetes 成功綁定 PVC 和 PV 後，回收策略會更新為與儲存類別的回收策略相符。
  - 如果儲存類別的回收策略是 `delete`當 PV 被刪除時，儲存磁碟區也會被刪除。
- 預設情況下， Trident管理 PVC，並在後端重新命名FlexVol volume和 LUN。你可以透過 `--no-manage`導入非託管磁碟區的標誌。如果你使用 `--no-manage`在物件的生命週期內， Trident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。

  如果您想使用 Kubernetes 來管理容器化工作負載，但又想在 Kubernetes 之外管理儲存磁碟區的生命週期，則此選項非常有用。

- 在 PVC 和 PV 中加入註釋，其作用有兩個：一是指示卷已導入，二是指示 PVC 和 PV 是否已管理。此註釋不應修改或刪除。

導入磁碟區

您可以使用 `tridentctl import`導入卷。

步驟

1. 建立持久性磁碟區聲明 (PVC) 檔案（例如， `pvc.yaml`) 將用於製造PVC。 PVC檔案應包含 `name`， `namespace` ， `accessModes` ， 和 `storageClassName`。 （可選）您可以指定 `unixPermissions`在你的PVC定義中。

   以下是一個最低規格範例：

   ```
   kind: PersistentVolumeClaim
   apiVersion: v1
   metadata:
     name: my_claim
     namespace: my_namespace
   spec:
     accessModes:
       - ReadWriteOnce
     storageClassName: my_storage_class
   ```

   不要包含其他參數，例如 PV 名稱或體積大小。這可能會導致導入命令失敗。

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、 Cloud Volumes Service路徑）。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>
```

範例

請查看以下卷導入範例，以了解支援的驅動程式。

ONTAP NAS 和ONTAP NAS FlexGroup

Trident支援使用以下方式導入磁碟區： `ontap-nas`和 `ontap-nas-flexgroup`司機。

- Trident不支援使用 `ontap-nas-economy` 司機。
- 這 `ontap-nas`和 `ontap-nas-flexgroup`驅動程式不允許重複的捲名稱。

每卷都是用以下方式創建的 `ontap-nas`driver 是ONTAP叢集上的FlexVol volume。使用以下方式導入FlexVol卷 `ontap-nas`驅動程式的工作原理相同。已存在於ONTAP叢集上的FlexVol磁碟區可以作為下列方式匯入： `ontap-nas` PVC。同樣， FlexGroup磁碟區也可以匯入為 `ontap-nas-flexgroup`PVC。

ONTAP NAS 範例

下面展示了託管磁碟區和非託管磁碟區匯入的範例。

託管磁碟區

以下範例導入一個名為“ `managed_volume`在名為 `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    |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
```

未託管卷

使用時 `--no-manage`理由是， Trident不會重新命名磁碟區。

以下範例導入 `unmanaged_volume`在 `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   |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
```

ONTAP SAN

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。

ONTAP SAN 範例

下面展示了託管磁碟區和非託管磁碟區匯入的範例。

託管磁碟區

對於託管卷， Trident會將FlexVol volume重新命名為 `pvc-<uuid>`FlexVol volume中的 LUN 格式 `lun0`。

以下範例導入了 `ontap-san-managed`FlexVol volume存在於 `ontap_san_default`後端：

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

未託管卷

以下範例導入 `unmanaged_example_volume`在 `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   |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

如果將 LUN 對應到與 Kubernetes 節點 IQN 共用相同 IQN 的 igroup，如下例所示，則會收到下列錯誤： `LUN already mapped to initiator(s) in this group`。您需要移除啟動器或取消映射 LUN 才能匯入磁碟區。

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

元素

Trident支援使用NetApp Element軟體和NetApp HCI卷導入 `solidfire-san`司機。

> Element驅動程式支援重複的磁碟區名稱。但是，如果磁碟區名稱重複， Trident會傳回錯誤。作為一種變通方法，克隆卷，提供一個唯一的磁碟區名稱，然後匯入克隆的磁碟區。

元素範例

以下範例導入一個 `element-managed`後端容量 `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    |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
```

谷歌雲端平台

Trident支援使用以下方式導入磁碟區： `gcp-cvs`司機。

> 若要將由NetApp Cloud Volumes Service支援的磁碟區匯入 Google Cloud Platform，請透過磁碟區路徑識別該磁碟區。磁碟區是磁碟區匯出路徑中位於下列位置之後的部分： `:/` 。例如，如果匯出路徑是 `10.0.0.1:/adroit-jolly-swift`體積路徑為 `adroit-jolly-swift`。

Google Cloud Platform 範例

以下範例導入一個 `gcp-cvs`後端容量 `gcpcvs_YEppr`具有體積路徑 `adroit-jolly-swift`。

```
tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-pvc-
file> -n trident

+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | gcp-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Azure NetApp Files

Trident支援使用以下方式導入磁碟區：`azure-netapp-files`司機。

> 若要匯入Azure NetApp Files，請透過磁碟區路徑識別該磁碟區。磁碟區是磁碟區匯出路徑中位於下列位置之後的部分：`:/`。例如，如果掛載路徑是`10.0.0.2:/importvol1`，**體積路徑**為`importvol1`。

Azure NetApp Files範例

以下範例導入一個`azure-netapp-files`**後端容量**`azurenetappfiles_40517`**體積路徑**`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支援使用以下方式導入磁碟區：`google-cloud-netapp-volumes`司機。

Google Cloud NetApp Volumes範例

以下範例導入一個 google-cloud-netapp-volumes`後端容量 `backend-tbc-gcnv1`音量 `testvoleasiaeast1。

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

以下範例導入一個 `google-cloud-netapp-volumes`當兩個體積存在於同一區域時，體積為：

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

自訂磁碟區名稱和標籤

使用Trident，您可以為建立的磁碟區指派有意義的名稱和標籤。這可以幫助您識別磁碟區並將其輕鬆對應到各自的 Kubernetes 資源（PVC）。您也可以在後端層級定義模板，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循這些模板。

開始之前

支援自訂磁碟區名稱和標籤：

1. 磁碟區建立、匯入和克隆操作。
2. 對於 ontap-nas-economy 驅動程序，只有 Qtree 磁碟區的名稱符合名稱範本。
3. 對於 ontap-san-economy 驅動程序，只有 LUN 名稱符合名稱範本。

限制

1. 可自訂磁碟區名稱僅與ONTAP本機驅動程式相容。
2. 可自訂的磁碟區名稱不適用於現有磁碟區。

可自訂磁碟區名稱的關鍵行為

1. 如果由於名稱範本中的語法無效而導致失敗，則後端建立將失敗。但是，如果範本套用失敗，則磁碟區將按照現有的命名約定命名。
2. 當使用後端配置中的名稱模板命名磁碟區時，儲存前綴不適用。可以直接在模板中加入任何所需的前綴值。

後端設定範例，包含名稱範本和標籤

可以在根層級和/或池層級定義自訂名稱範本。

根級別範例

```
{
  "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}}"
  }
}
```

池級範例

```
{
  "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 }}""
```

需要考慮的幾點

1. 對於磁碟區匯入，只有當現有磁碟區具有特定格式的標籤時，才會更新標籤。例如：`{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}` 。
2. 對於託管磁碟區匯入，磁碟區名稱遵循後端定義根層級定義的名稱範本。
3. Trident不支援將切片運算子與儲存前綴一起使用。
4. 如果模板無法產生唯一的磁碟區名稱， Trident將添加一些隨機字元來建立唯一的磁碟區名稱。
5. 如果 NAS 經濟型磁碟區的自訂名稱長度超過 64 個字符， Trident將依照現有的命名約定為磁碟區命名。對於所有其他ONTAP驅動程序，如果磁碟區名稱超過名稱限制，則磁碟區建立程序將會失敗。

跨命名空間分享 **NFS** 卷

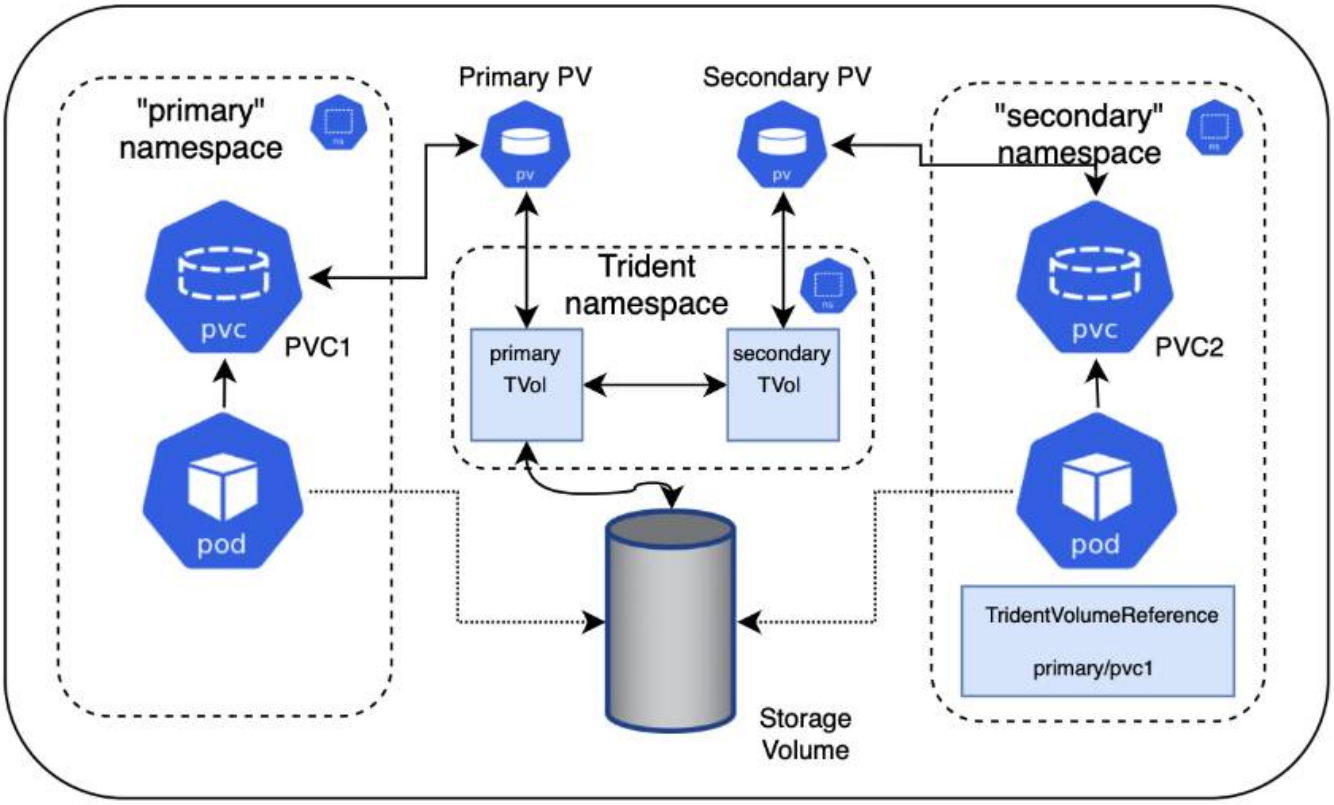
使用Trident，您可以在主命名空間中建立磁碟區，並在一個或多個輔助命名空間中共用該磁碟區。

特徵

TridentVolumeReference CR 可讓您在一個或多個 Kubernetes 命名空間之間安全地共用 ReadWriteMany (RWX) NFS 磁碟區。這種 Kubernetes 原生解決方案具有以下優點：

- 多層級存取控制以確保安全性
- 適用於所有Trident NFS 磁碟區驅動程式
- 不依賴 tridentctl 或任何其他非原生 Kubernetes 功能

此圖展示了跨兩個 Kubernetes 命名空間的 NFS 磁碟區共用。



快速啟動

只需幾個步驟即可設定NFS卷共享。

1 配置源**PVC**以共享體積

來源命名空間擁有者授予存取來源 PVC 中資料的權限。

2 授予在目標命名空間中建立 **CR** 的權限

叢集管理員授予目標命名空間的擁有者建立 TridentVolumeReference CR 的權限。

3 在目標命名空間中建立 **TridentVolumeReference**

目標命名空間的擁有者建立 TridentVolumeReference CR 以引用來源 PVC。

4 在目標命名空間中建立從屬 **PVC**

目標命名空間的擁有者建立從屬 PVC，以使用來源 PVC 中的資料來源。

配置來源命名空間和目標命名空間

為確保安全，跨命名空間共用需要來源命名空間擁有者、叢集管理員和目標命名空間擁有者的協作和行動。使用者角色在每個步驟中都會被指定。

步驟

1. 來源命名空間擁有者： 建立PVC(`pvc1`）在來源命名空間中，授予與目標命名空間共享的權限(`namespace2`) 使用 `shareToNamespace`註解。

```
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建立 PV 及其後端 NFS 儲存磁碟區。

- 您可以使用逗號分隔的清單將 PVC 共用給多個命名空間。例如，`trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4` 。
- 您可以使用以下方式共用到所有命名空間 `*`。例如，`trident.netapp.io/shareToNamespace: *`
- 您可以更新PVC以包含 `shareToNamespace`隨時可以添加註釋。

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. 目標命名空間擁有者： 在目標命名空間中建立一個指向來源命名空間的 TridentVolumeReference CR。`pvc1` 。

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. 目標命名空間擁有者： 建立PVC(`pvc2`）在目標命名空間(`namespace2`) 使用 `shareFromPVC`註記以指定來源 PVC。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

目標PVC管的尺寸必須小於或等於源PVC管的尺寸。

結果

Trident讀取 `shareFromPVC`在目標 PVC 上新增註釋，並將目標 PV 建立為沒有自身儲存資源的從屬卷，該磁碟區指向來源 PV 並共用來源 PV 儲存資源。目的地 PVC 和 PV 看起來已正常綁定。

刪除共享卷

您可以刪除跨多個命名空間共享的磁碟區。 Trident將移除對來源命名空間中該磁碟區的存取權限，並保留對共用該磁碟區的其他命名空間的存取權限。當所有引用該磁碟區的命名空間都被刪除後， Trident會刪除該磁碟區。

使用 **`tridentctl get`**查詢子卷

使用[`tridentctl`實用程序，您可以運行 `get`取得子卷的指令。更多信息，請參閱連結：../trident-reference/tridentctl.html[`tridentctl`命令和選項]。

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

標誌：

- `-h, --help`：有關卷的幫助。
- `--parentOfSubordinate string`：將查詢限制在子來源磁碟區上。
- `--subordinateOf string`：將查詢限制在磁碟區的下級。

限制

- Trident無法阻止目標命名空間寫入共享磁碟區。您應該使用文件鎖定或其他方法來防止覆蓋共享卷資料。

- 您無法透過移除來源PVC來撤銷對來源PVC的存取權限。 `shareToNamespace`或者 `shareFromNamespace`註釋或刪除 `TridentVolumeReference`CR。若要撤銷存取權限，必須刪除從屬PVC。
- 從屬磁碟區無法進行快照、複製和鏡像操作。

更多資訊

要了解有關跨命名空間卷訪問的更多資訊：

- 訪問"在命名空間之間共享卷：迎接跨命名空間卷訪問"。
- 觀看示範"NetAppTV"。

跨命名空間克隆卷

使用Trident，您可以利用同一 Kubernetes 叢集中不同命名空間內的現有磁碟區或磁碟區快照建立新磁碟區。

先決條件

在克隆卷之前，請確保來源後端和目標後端是同一類型，並且具有相同的儲存類別。

跨命名空間克隆僅支援以下情況： `ontap-san`和 `ontap-nas`儲存驅動程式。不支援只讀克隆。

快速啟動

只需幾個步驟即可設定卷克隆。

1 配置來源 **PVC** 以克隆卷

來源命名空間擁有者授予存取來源 PVC 中資料的權限。

2 授予在目標命名空間中建立 **CR** 的權限

叢集管理員授予目標命名空間的擁有者建立 TridentVolumeReference CR 的權限。

3 在目標命名空間中建立 **TridentVolumeReference**

目標命名空間的擁有者建立 TridentVolumeReference CR 以引用來源 PVC。

4 在目標命名空間中建立克隆 **PVC**

目標命名空間的擁有者建立 PVC 以複製來源命名空間中的 PVC。

配置來源命名空間和目標命名空間

為確保安全，跨命名空間複製磁碟區需要來源命名空間擁有者、叢集管理員和目標命名空間擁有者的協作和操

作。使用者角色在每個步驟中都會被指定。

步驟

1. 來源命名空間擁有者： 建立PVC(`pvc1`）在來源命名空間中(`namespace1`)授予與目標命名空間共享的權限(`namespace2`)使用 `cloneToNamespace`註解。

```
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建立 PV 及其後端儲存磁碟區。

- 您可以使用逗號分隔的清單將 PVC 共用給多個命名空間。例如，`trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4` 。
- 您可以使用以下方式共用到所有命名空間 `*`。例如，`trident.netapp.io/cloneToNamespace: *`
- 您可以更新PVC以包含 `cloneToNamespace`隨時可以添加註釋。

2. 叢集管理員： 確保已配置正確的基於角色的存取控制 (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. 目標命名空間擁有者： 在目標命名空間中建立一個指向來源命名空間的 TridentVolumeReference CR。`pvc1` 。

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. 目標命名空間擁有者： 建立PVC(`pvc2` ）在目標命名空間(`namespace2`) 使用 `cloneFromPVC`或者 `cloneFromSnapshot`， 和 `cloneFromNamespace`用於指定來源PVC的註解。

```
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-economy 驅動程式配置的 PVC，不支援唯讀克隆。

使用SnapMirror複製卷

Trident支援一個叢集上的來源磁碟區與對等叢集上的目標磁碟區之間的鏡像關係，用於複製資料以實現災難復原。 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：

- 在體積（PVC）之間建立鏡像關係
- 移除磁碟區之間的鏡像關係
- 打破鏡像關係
- 在災難情況下（故障轉移）提升備用磁碟區的可用性
- 在計劃內故障轉移或遷移期間，實現應用程式在叢集間的無損遷移。

複製的前提條件

在開始之前，請確保滿足以下先決條件：

ONTAP叢集

- * Trident *：使用ONTAP作為後端的來源 Kubernetes 叢集和目標 Kubernetes 叢集上必須存在Trident版本 22.10 或更高版本。
- 許可證：使用資料保護包的ONTAP SnapMirror非同步許可證必須在來源 ONTAP 叢集和目標ONTAP叢集上啟用。請參閱 "ONTAP中的SnapMirror許可概述"了解更多。

從ONTAP 9.10.1 開始，所有許可證均以NetApp許可證文件 (NLF) 的形式交付，這是一個可以啟用多種功能的單一文件。請參閱"ONTAP One 隨附的許可證"了解更多。

僅支援SnapMirror非同步保護。

對等互連

- 叢集和 **SVM**： ONTAP儲存後端必須相互連接。請參閱 "叢集和SVM對等連接概述"了解更多。

確保兩個ONTAP叢集之間複製關係中使用的 SVM 名稱是唯一的。

- * Trident和 SVM*：對等遠端 SVM 必須可供目標叢集上的Trident使用。

支援的驅動程式

NetApp Trident支援使用NetApp SnapMirror技術進行磁碟區複製，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`** ：**FC** `ontap-san` ：NVMe/TCP（最低要求ONTAP版本 9.15.1）

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"了解ASA r2 儲存系統"。

製作鏡面**PVC**

請依照這些步驟，並使用 CRD 範例，在主磁碟區和輔助磁碟區之間建立鏡像關係。

步驟

1. 在主 Kubernetes 叢集上執行下列步驟：
   a. 使用下列方式建立一個 StorageClass 對象 `trident.netapp.io/replication: true`範圍。

      例子

      ```
      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。

例子

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

c. 使用本機資訊建立鏡像關係變更要求。

例子

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident取得磁碟區的內部資訊和磁碟區的目前資料保護 (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. 在輔助 Kubernetes 叢集上執行下列步驟：
   a. 建立一個 StorageClass，並設定 trident.netapp.io/replication: true 參數。

      例子

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        trident.netapp.io/replication: true
      ```

   b. 建立包含目標和來源資訊的鏡像關係 CR。

      例子

      ```
      kind: TridentMirrorRelationship
      apiVersion: trident.netapp.io/v1
      metadata:
        name: csi-nas
      spec:
        state: established
        volumeMappings:
        - localPVCName: csi-nas
          remoteVolumeHandle:
      "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
      ```

Trident將使用配置的關係原則名稱（或ONTAP的預設值）建立SnapMirror關係並對其進行初始化。

c. 建立一個 PVC，使用先前建立的 StorageClass 作為輔助儲存類別（SnapMirror目標）。

例子

```
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將檢查 TridentMirrorRelationship CRD，如果該關係不存在，則建立磁碟區失敗。如果存在此關係， Trident將確保將新的FlexVol volume放置在與 MirrorRelationship 中定義的遠端 SVM 對等的 SVM 上。

卷複製狀態

Trident鏡像關係 (TMR) 是一種 CRD，它表示 PVC 之間複製關係的一端。目標 TMR 具有一個狀態，該狀態告訴Trident期望的狀態是什麼。目的地 TMR 具有以下狀態：

- 已建立：本地 PVC 是鏡像關係的目標量，這是一個新的關係。
- 已推廣：本地 PVC 可讀寫且可安裝，目前沒有鏡像關係。
- 重新建立：本地 PVC 是鏡像關係的目標量，之前也處於該鏡像關係中。
  - 如果目標磁碟區曾經與來源磁碟區有關聯，則必須使用重新建立的狀態，因為它會覆寫目標磁碟區的內容。
  - 如果磁碟區之前未與來源建立關係，則重新建立狀態將會失敗。

在計劃外故障切換期間促進輔助**PVC**的運行

在輔助 Kubernetes 叢集上執行下列步驟：

- 將 TridentMirrorRelationship 的 *spec.state* 欄位更新為 `promoted`。

在計劃故障切換期間推廣備用**PVC**

在計劃故障轉移（遷移）期間，執行以下步驟以提升輔助 PVC：

步驟

1. 在主 Kubernetes 叢集上，建立 PVC 的快照，並等待快照建立完成。
2. 在主 Kubernetes 叢集上，建立 SnapshotInfo CR 以取得內部詳細資訊。

   例子

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. 在輔助 Kubernetes 叢集上，將 *TridentMirrorRelationship* CR 的 *spec.state* 欄位更新為 *promoted*，並將 *spec.promotedSnapshotHandle* 更新為快照的內部名稱。
4. 在輔助 Kubernetes 叢集上，確認 TridentMirrorRelationship 的狀態（status.state 欄位）是否已提升。

故障轉移後恢復鏡像關係

在恢復鏡像關係之前，選擇你想作為新主要方的那一方。

步驟

1. 在輔助 Kubernetes 叢集上，請確保 TridentMirrorRelationship 上的 *spec.remoteVolumeHandle* 欄位的值已更新。
2. 在輔助 Kubernetes 叢集上，將 TridentMirrorRelationship 的 *spec.mirror* 欄位更新為 `reestablished`。

附加手術

Trident支援對主磁碟區和輔助磁碟區執行下列操作：

將主**PVC**複製到新的輔助**PVC**

請確保您已備有主PVC管及備用PVC管。

步驟

1. 從已建立的輔助（目標）叢集中刪除 PersistentVolumeClaim 和 TridentMirrorRelationship CRD。
2. 從主（來源）叢集中刪除 TridentMirrorRelationship CRD。
3. 在主（來源）叢集上為要建立的新輔助（目標）PVC 建立一個新的 TridentMirrorRelationship CRD。

調整鏡像、主或次級**PVC**的尺寸

PVC 可以像往常一樣調整大小，如果資料量超過目前大小， ONTAP將自動擴展任何目標 flevxols。

從 **PVC** 移除複製

若要移除複製，請對目前輔助磁碟區執行下列其中一項操作：

- 刪除輔助 PVC 上的鏡像關係。這會破壞複製關係。

- 或者，將 spec.state 欄位更新為 *promoted*。

刪除一個**PVC**（之前已鏡像）

Trident會檢查是否有複製的 PVC，並在嘗試刪除磁碟區之前釋放複製關係。

刪除 **TMR**

刪除鏡像關係一側的 TMR 會導致剩餘的 TMR 在Trident完成刪除之前轉換為 *promoted* 狀態。如果選擇刪除的 TMR 已處於 *promoted* 狀態，則不存在鏡像關係，TMR 將被刪除， Trident會將本機 PVC 提升為 *ReadWrite*。此刪除操作會釋放ONTAP中本機磁碟區的SnapMirror元資料。如果將來要將此磁碟區用於鏡像關係，則在建立新的鏡像關係時，必須使用具有 *established* 磁碟區複製狀態的新 TMR。

ONTAP在線時，更新鏡像關係

鏡像關係建立後可以隨時更新。您可以使用 `state: promoted`或者 `state: reestablished`用於更新關係的欄位。將目標磁碟區提升為常規讀寫磁碟區時，可以使用 *promotedSnapshotHandle* 指定要將目前磁碟區還原到的特定快照。

ONTAP離線時更新鏡像關係

您可以使用 CRD 執行SnapMirror更新，而無需Trident與ONTAP叢集直接連線。請參考以下 TridentActionMirrorUpdate 的範例格式：

例子

```
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 的狀況。它可以取值 成功、進行中_或_失敗。

使用 **CSI** 拓撲

Trident可以利用以下方式選擇性地建立磁碟區並將其附加到 Kubernetes 叢集中的節點： "CSI拓樸功能" 。

概況

使用 CSI 拓撲功能，可以根據區域和可用區域將對磁碟區的存取限製到節點子集。如今，雲端服務供應商允許 Kubernetes 管理員建立基於區域的節點。節點可以位於一個區域內的不同可用區，也可以跨越多個區域。為了方便在多區域架構中為工作負載配置卷， Trident使用了 CSI 拓撲。

了解更多關於 CSI 拓撲功能的信息 "這裡"。

Kubernetes 提供了兩種獨特的磁碟區綁定模式：

- 和 `VolumeBindingMode`設定為 `Immediate`Trident創建卷時沒有任何拓樸感知。磁碟區綁定和動態配置在建立 PVC 時處理。這是預設值。 `VolumeBindingMode`適用於不強制執行拓樸約束的叢集。持久性卷的建立不依賴請求 pod 的調度要求。
- 和 `VolumeBindingMode`設定為 `WaitForFirstConsumer`PVC 的持久性磁碟區的建立和綁定將被延遲，直到使用該 PVC 的 pod 被調度和建立。這樣，就可以建立磁碟區來滿足拓樸要求所強制執行的調度約束。

> ⓘ 這 `WaitForFirstConsumer`綁定模式不需要拓樸標籤。這可以獨立於 CSI 拓樸功能使用。

你需要什麼

要使用 CSI 拓撲結構，您需要以下元件：

- 運行中的 Kubernetes 集群"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"}
```

- 叢集中的節點應附有標籤，以體現拓樸感知能力。(`topology.kubernetes.io/region`和 `topology.kubernetes.io/zone`）。在安裝Trident之前，叢集中的節點上*應該存在這些標籤*，以便Trident能夠感知拓撲結構。

```
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**：建立拓樸感知後端

Trident儲存後端可設計為根據可用區選擇性地配置磁碟區。每個後端都可以攜帶一個可選組件 `supportedTopologies`表示所支援的區域和地區列表的區塊。對於使用此類後端的 StorageClasses，只有在受支援的區域/區域中調度的應用程式請求時才會建立磁碟區。

以下是一個後端定義範例：

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"
    }
  ]
}
```

`supportedTopologies`用於提供每個後端區域和分區的清單。這些區域和分區代表了StorageClass 中可以提供的允許值的清單。對於包含後端提供的區域和可用區子集的儲存類，Trident會在後端建立一個磁碟區。

你可以定義 `supportedTopologies`每個儲存池也是如此。請參閱以下範例：

```
---
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
```

在這個例子中， `region`和 `zone`標籤代表儲存池的位置。 `topology.kubernetes.io/region`和 `topology.kubernetes.io/zone`決定儲存池可以從何處使用。

步驟 **2**：定義拓樸感知的儲存類別。

根據提供給叢集中節點的拓樸標籤，可以定義 StorageClasses 來包含拓樸資訊。這將決定哪些儲存池可作為 PVC 請求的候選對象，以及哪些節點子集可以使用Trident提供的磁碟區。

請參閱以下範例：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata: null
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions: null
  - key: topology.kubernetes.io/zone
    values:
      - us-east1-a
      - us-east1-b
  - key: topology.kubernetes.io/region
    values:
      - us-east1
parameters:
  fsType: ext4
```

在上述 StorageClass 定義中， `volumeBindingMode`設定為 `WaitForFirstConsumer`。使用此 StorageClass 請求的 PVC 只有在 pod 中被引用後才會執行。和， `allowedTopologies`提供要使用的區域和範圍。這 `netapp-san-us-east1`StorageClass 在儲存體上建立 PVC。 `san-backend-us-east1`後端定義如上所述。

步驟 **3**：製作並使用 **PVC**

建立 StorageClass 並將其對應到後端後，現在可以建立 PVC。

請參閱範例 `spec`以下：

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

使用此清單建立 PVC 將產生以下結果：

```
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
```

為了讓Trident形成體積並將其與 PVC 結合，請使用 PVC 管。請參閱以下範例：

```yaml
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 podSpec 指示 Kubernetes 將 pod 調度到存在於下列位置的節點上：`us-east1`在該區域中，選擇任何存在的節點。`us-east1-a`或者 `us-east1-b`區域。

請查看以下輸出：

```
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
```

更新後端以包含 `supportedTopologies`

可以更新現有的後端，使其包含以下清單： `supportedTopologies`使用 `tridentctl backend update`。這不會影響已經分配的容量，只會用於後續的PVC。

查找更多信息

- "管理容器資源"
- "節點選擇器"
- "親和力和反親和力"
- "污點和容忍度"

使用快照

Kubernetes 持久性磁碟區 (PV) 的磁碟區快照可以建立磁碟區的某個時間點副本。您可以建立使用Trident建立的磁碟區的快照，匯入在Trident之外建立的快照，從現有快照建立新磁碟區，以及從快照還原磁碟區資料。

概況

卷快照受支援 `ontap-nas`， `ontap-nas-flexgroup` ， `ontap-san` ， `ontap-san-economy` ， `solidfire-san` ， `gcp-cvs` ， `azure-netapp-files` ， 和 `google-cloud-netapp-volumes`司機。

開始之前

若要使用快照，您必須擁有外部快照控制器和自訂資源定義 (CRD)。這是 Kubernetes 編排器（例如：Kubeadm、GKE、OpenShift）的職責。

如果您的 Kubernetes 發行版不包含快照控制器和 CRD，請參閱[部署磁碟區快照控制器]。

> 如果在 GKE 環境中建立按需磁碟區快照，則不要建立快照控制器。 GKE 使用內建的隱藏快照控制器。

建立磁碟區快照

步驟

1. 創建一個 `VolumeSnapshotClass`更多信息，請參閱...“卷快照類” 。
   - 這 `driver`指向Trident CSI 驅動程式。
   - `deletionPolicy`可以 `Delete`或者 `Retain`。設定為 `Retain`即使發生以下情況，儲存叢集上的底層實體快照仍會被保留： `VolumeSnapshot`物件已被刪除。

     例子

     ```
     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
     ```

   - 此範例為名為“ `pvc1`快照名稱設定為 `pvc1-snap`。 VolumeSnapshot 類似於 PVC，並且與某個 PVC 相關聯。 `VolumeSnapshotContent`代表實際快照的對象。

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                    AGE
pvc1-snap               50s
```

- 你可以識別出 `VolumeSnapshotContent`對象為 `pvc1-snap`透過描述來取得磁碟區快照。這 `Snapshot Content Name`識別提供此快照的 VolumeSnapshotContent 物件。這 `Ready To Use`此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 VolumeSnapshot 的 VolumeSnapshot 建立 PVC `<pvc-name>`作為數據來源。 PVC管製作完成後，可以將其連接到艙體上，並像其他PVC管一樣使用。

> PVC 將在與來源磁碟區相同的後端建立。參考"知識庫：無法在備用後端從Trident PVC 快照建立 PVC。"。

以下範例使用以下方法建立 PVC： `pvc1-snap`作為資料來源。

```
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
```

匯入磁碟區快照

Trident支持"Kubernetes 預先設定快照流程"使集群管理員能夠創建 `VolumeSnapshotContent`在Trident之外建立的物件和匯入快照。

開始之前

Trident肯定建立或匯入了快照的父卷。

步驟

1. 集群管理員： 建立一個 `VolumeSnapshotContent`引用後端快照的物件。這將啟動Trident中的快照工作流程。
   - 指定後端快照的名稱 `annotations`作為 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`。
   - 指定 `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>`在 `snapshotHandle`這是外部快照程式提供給Trident的唯一資訊。 `ListSnapshots`稱呼。

     這 `<volumeSnapshotContentName>`由於 CR 命名限制，有時無法與後端快照名稱完全匹配。

     例子

     以下範例建立了一個 `VolumeSnapshotContent`引用後端快照的對象 `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. 集群管理員： 建立 `VolumeSnapshot`引用 CR `VolumeSnapshotContent`目的。這是對使用權限的請求 `VolumeSnapshot`在給定的命名空間中。

   例子

   以下範例建立了一個 `VolumeSnapshot`CR命名 `import-snap`指的是 `VolumeSnapshotContent`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（無需操作）： 外部快照程式識別新建立的內容 `VolumeSnapshotContent`並運行 `ListSnapshots`稱呼。`Trident`創造了 `TridentSnapshot`。
   - 外部快照程式設定 `VolumeSnapshotContent`到 `readyToUse`以及 `VolumeSnapshot`到 `true`。
   - Trident回歸 `readyToUse=true`。
4. 任何使用者： 建立一個 `PersistentVolumeClaim`參考新的 `VolumeSnapshot`，其中 `spec.dataSource`（或者 `spec.dataSourceRef`名稱是 `VolumeSnapshot`姓名。

   例子

以下範例建立一個引用 PVC 的 `VolumeSnapshot`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
```

使用快照恢復磁碟區數據

預設情況下，快照目錄是隱藏的，以確保使用以下方式配置的磁碟區的最大相容性： `ontap-nas`和 `ontap-nas-economy`司機。啟用 `.snapshot`直接從快照還原資料的目錄。

使用磁碟區快照還原ONTAP CLI 將磁碟區還原到先前快照中記錄的狀態。

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

還原快照副本時，現有磁碟區配置將被覆寫。建立快照副本後對磁碟區資料所做的變更將會遺失。

從快照進行原地體積恢復

Trident利用快照提供快速、原位體積恢復功能 `TridentActionSnapshotRestore`（TASR）CR。此 CR 作為強制性 Kubernetes 操作，在操作完成後不會持久保存。

Trident支援快照恢復 `ontap-san`， `ontap-san-economy` ， `ontap-nas` ， `ontap-nas-flexgroup` ， `azure-netapp-files` ， `gcp-cvs` ， `google-cloud-netapp-volumes` ， 和 `solidfire-san`司機。

開始之前

您必須擁有已綁定的PVC和可用的磁碟區快照。

- 確認PVC狀態是否已綁定。

```
kubectl get pvc
```

- 確認磁碟區快照已準備就緒。

```
kubectl get vs
```

步驟

1. 建立 TASR CR。此範例為PVC建立CR。 `pvc1`和磁碟區快照 `pvc1-snapshot`。

   TASR CR 必須位於 PVC 和 VS 存在的命名空間。

```
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. 應用 CR 從快照恢復。此範例從快照恢復 `pvc1`。

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

結果

Trident從快照中復原資料。您可以驗證快照復原狀態：

```
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: ""
```

- 大多數情況下， Trident不會在操作失敗後自動重試。您需要再次執行此操作。
- 沒有管理員權限的 Kubernetes 使用者可能需要管理員授予權限才能在其應用程式命名空間中建立 TASR CR。

刪除包含關聯快照的 PV

刪除具有關聯快照的持久性磁碟區時，對應的Trident磁碟區將更新為「正在刪除」狀態。刪除磁碟區快照以刪除Trident磁碟區。

部署磁碟區快照控制器

如果您的 Kubernetes 發行版不包含快照控制器和 CRD，您可以如下部署它們。

步驟

1. 建立磁碟區快照 CRD。

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. 建立快照控制器。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

如有必要，打開 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`並更新 `namespace`新增到您的命名空間。

相關連結

- "卷快照"
- "卷快照類"

使用磁碟區組快照

NetApp Trident提供了建立多個磁碟區（一組磁碟區快照）快照的功能，可用於建立持久磁碟區 (PV) 的 Kubernetes 磁碟區組快照。此磁碟區組快照表示在同一時間點從多個磁碟區中取得的副本。

VolumeGroupSnapshot 是 Kubernetes 中的測試版功能，其 API 也處於測試版階段。VolumeGroupSnapshot 所需的最低 Kubernetes 版本為 1.32。

建立卷宗組快照

卷冊組快照受支援 `ontap-san`此驅動程式僅適用於 iSCSI 協議，尚不支援光纖通道 (FCP) 或 NVMe/TCP。開始之前

- 請確保您的 Kubernetes 版本為 K8s 1.32 或更高版本。
- 若要使用快照，您必須擁有外部快照控制器和自訂資源定義 (CRD)。這是 Kubernetes 編排器（例如：Kubeadm、GKE、OpenShift）的職責。

  如果您的 Kubernetes 發行版不包含外部快照控制器和 CRD，請參閱[部署磁碟區快照控制器]。

  > 如果在 GKE 環境中建立按需卷宗組快照，則不要建立快照控制器。 GKE 使用內建的隱藏快照控制器。

- 在快照控制器 YAML 中，設定 `CSIVolumeGroupSnapshot`將功能門設為“true”，以確保啟用磁碟區組快照。
- 在建立磁碟區組快照之前，請先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 SVM 上，以便能夠建立 VolumeGroupSnapshot。

步驟

- 在創建 VolumeGroupSnapshot 之前，請先建立 VolumeGroupSnapshotClass。更多信息，請參閱"卷冊組快照類"。

  ```
  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
  ```

- 使用現有的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的 PVC。

  以下範例使用以下方法建立 PVC： `pvc1-group-snap`作為資料來源和標籤 `consistentGroupSnapshot: groupA`。根據您的需求定義標籤的鍵和值。

```
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磁碟區組快照(`consistentGroupSnapshot: groupA`）在PVC中規定。

  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一部分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還原。

使用磁碟區快照還原ONTAP CLI 將磁碟區還原到先前快照中記錄的狀態。

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

還原快照副本時，現有磁碟區配置將被覆寫。建立快照副本後對磁碟區資料所做的變更將會遺失。

從快照進行原地體積恢復

Trident利用快照提供快速、原位體積恢復功能 `TridentActionSnapshotRestore`（TASR）CR。此 CR 作為強制性 Kubernetes 操作，在操作完成後不會持久保存。

有關詳細信息，請參閱 "從快照進行原地體積恢復"。

刪除包含關聯群組快照的 **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該持久卷， Trident會將該卷移至「正在刪除」狀態，因為必須先刪除快照才能安全地刪除該卷。
- 如果使用分組快照建立了克隆，然後要刪除該群組，則會開始在克隆時進行分割操作，並且在拆分完成之前無法刪除該群組。

部署磁碟區快照控制器

如果您的 Kubernetes 發行版不包含快照控制器和 CRD，您可以如下部署它們。

步驟

1. 建立磁碟區快照 CRD。

```
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. 建立快照控制器。

```
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
```

如有必要，打開 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`並更新 `namespace`新增到您的命名空間。

相關連結

- "卷冊組快照類"
- "卷快照"

管理與監控Trident

升級Trident

升級Trident

從 24.02 版本開始， Trident遵循四個月的發布節奏，每年發布三個主要版本。每個新版本都以先前的版本為基礎，並提供新功能、效能增強、錯誤修復和改進。我們建議您至少每年升級一次，以充分利用Trident的新功能。

升級前的注意事項

升級至最新版Trident時，請注意以下事項：

- 在給定的 Kubernetes 叢集的所有命名空間中，應該只安裝一個Trident實例。
- Trident 23.07 及更高版本需要 v1 卷快照，不再支援 alpha 或 beta 快照。
- 如果您在 Google Cloud 中建立了Cloud Volumes Service，"CVS 服務類型"您必須更新後端配置才能使用 `standardsw`或者 `zoneredundantstandardsw`從Trident 23.01 升級時的服務等級。未能更新 `serviceLevel`後端故障可能導致磁碟區故障。參考 "CVS 服務類型範例" 了解詳情。
- 升級時，請務必提供以下信息 `parameter.fsType`在 `StorageClasses`由Trident使用。您可以刪除並重新創建 `StorageClasses`在不影響原有流量的情況下。
  - 這是強制執行的要求 "安全情境"適用於 SAN 卷。
  - https://github.com/NetApp/trident/tree/master/trident-installer/sample-input [範例輸入^] 目錄包含範例，例如https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-basic.yaml.templ[`storage-class-basic.yaml.templ` ^] 與連結：`storage-class-bronze-default.yaml` 。
  - 更多信息，請參閱"已知問題"。

步驟 **1**：選擇版本

Trident 的版本遵循基於日期的命名規則。 `YY.MM`命名規則中，「YY」是年份的最後兩位數字，「MM」是月份。 Dot 版本遵循 `YY.MM.X`按照慣例，其中“X”是補丁等級。您需要根據目前版本選擇要升級到的版本。

- 您可以直接升級到與已安裝版本相差不超過四個版本視窗內的任何目標版本。例如，您可以直接從 24.06（或任何 24.06 的小版本）升級到 25.06。
- 如果您要從四個版本視窗以外的版本升級，請執行多步驟升級。請依照升級說明進行操作。"早期版本"您正在從目前版本升級到符合四版本發布視窗的最新版本。例如，如果您目前運行的是 23.07 版本，並且想要升級到 25.06 版本：
  a. 首次升級從 23.07 升級到 24.06。
  b. 然後從 24.06 升級到 25.06。

> 在 OpenShift 容器平台上使用Trident運算子進行升級時，應升級至Trident 21.01.1 或更高版本。隨 21.01.0 版本發布的Trident操作符存在一個已知問題，該問題已在 21.01.1 版本中修復。更多詳情請參閱... "GitHub 上的問題詳情" 。

步驟二：確定原始安裝方法

若要確定您最初安裝Trident時所使用的版本：

1. 使用 `kubectl get pods -n trident`檢查豆莢。
   - 如果沒有操作員艙，Trident是透過以下方式安裝的： `tridentctl` 。
   - 如果存在 operator pod，則Trident是透過Trident operator 手動安裝的，或使用 Helm 安裝的。
2. 如果有操作員艙，請使用 `kubectl describe torc`確定Trident是否使用 Helm 安裝。
   - 如果有 Helm 標籤，表示Trident是使用 Helm 安裝的。
   - 如果沒有 Helm 標籤，則表示Trident是使用Trident操作員手動安裝的。

步驟 **3**：選擇升級方式

通常情況下，您應該使用與初始安裝相同的方法進行升級，但是您也可以…"安裝方法之間的轉換" 。升級Trident有兩種選擇。

- "使用Trident操作員進行升級"

  我們建議您查看"了解操作員升級工作流程"在與運營商升級之前。

*

透過營運商進行升級

了解操作員升級工作流程

在使用Trident操作員升級Trident之前，您應該了解升級期間發生的背景處理程序。這包括對Trident控制器、控制器 Pod 和節點 Pod 以及節點 DaemonSet 的更改，以啟用滾動更新。

Trident操作員升級處理

眾多之一"使用Trident運算子的好處"安裝和升級Trident 的過程是自動處理Trident和 Kubernetes 對象，而不會中斷現有的已掛載磁碟區。這樣一來， Trident就能支援零停機時間的升級，或者說，"捲動更新" 。具體來說，Trident運算子與 Kubernetes 叢集通訊以：

- 刪除並重新建立Trident Controller 部署和節點 DaemonSet。
- 將Trident Controller Pod 和Trident Node Pod 替換為新版本。
  - 如果某個節點沒有更新，並不妨礙其他節點的更新。
  - 只有運行了Trident Node Pod 的節點才能掛載磁碟區。

有關 Kubernetes 叢集上Trident架構的更多信息，請參閱："Trident架構" 。

操作員升級工作流程

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 對象，包括 CRD、RBAC 和Trident SVC。
    c. 刪除版本為 *n* 的Trident Controller 部署。
    d. 建立版本 *n+1* 的Trident Controller 部署。
2. **Kubernetes** 為 *n+1* 建立Trident Controller Pod。
3. Trident運算子：
    a. 刪除 *n* 的Trident節點 DaemonSet。操作員不會等待節點 Pod 終止。
    b. 為 *n+1* 建立Trident節點守護程式集。
4. **Kubernetes** 在未執行Trident Node Pod *n* 的節點上建立Trident Node Pod。這樣可以確保一個節點上永遠不會有超過一個Trident Node Pod，無論版本如何。

使用**Trident Operator** 或 **Helm** 升級**Trident**安裝

您可以使用Trident Operator 手動升級Trident ，也可以使用 Helm 進行升級。您可以從一個Trident Operator 安裝升級到另一個Trident Operator 安裝，或從一個 Trident Operator 安裝升級到另一個 Trident Operator 安裝。 `tridentctl`安裝到Trident操作員版本。審查"選擇升級方式"在升級Trident操作程序之前。

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。

> ⓘ 若要從使用命名空間作用域運算子安裝的Trident （版本 20.07 至 20.10）進行升級，請使用下列升級說明："您已安裝的版本"Trident。

關於此任務

Trident提供了一個捆綁文件，您可以使用該文件安裝 Operator 並為您的 Kubernetes 版本建立關聯物件。

- 對於運行 Kubernetes 1.24 的集群，請使用"bundle_pre_1_25.yaml"。
- 對於運行 Kubernetes 1.25 或更高版本的集群，請使用"bundle_post_1_25.yaml"。

開始之前

請確保您使用的是正在運行的 Kubernetes 叢集。"支援的 Kubernetes 版本" 。

步驟

1. 請驗證您的Trident版本：

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

2. 更新 `operator.yaml`， `tridentorchestrator_cr.yaml` ， 和 `post_1_25_bundle.yaml`使用要升級到的版本（例如 25.06）的登錄和映像路徑，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. 如果您使用自訂方式進行了初始安裝 `TridentOrchestrator`屬性，您可以編輯 `TridentOrchestrator`用於修改安裝參數的物件。這可能包括為離線模式指定鏡像Trident和 CSI 映像登錄、啟用偵錯日誌或指定映像拉取金鑰所做的變更。
5. 使用適用於您環境的正確 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。

> 當已安裝Trident 的Kubernetes 叢集從 1.24 版本升級到 1.25 或更高版本時，必須更新 values.yaml 檔案進行設定。 `excludePodSecurityPolicy`到 `true`或添加 `--set excludePodSecurityPolicy=true`到 `helm upgrade`升級叢集前需要執行此命令。

如果您已將 Kubernetes 叢集從 1.24 升級到 1.25，但未升級Trident helm，則 helm 升級將失敗。若要成功升級 Helm，請先執行以下步驟：

1. 從下列位置安裝 helm-mapkubeapis 插件 https://github.com/helm/helm-mapkubeapis。
2. 在Trident安裝所在的命名空間中，對Trident版本執行一次試運行。這裡列出了需要清理的資源。

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. 使用 Helm 執行完整運行以進行清理。

```
helm mapkubeapis trident  --namespace trident
```

步驟

1. 如果你"使用 Helm 安裝了Trident 。"你可以使用 `helm upgrade trident netapp-trident/trident-operator --version 100.2506.0`一步即可升級。如果您沒有新增 Helm 倉庫或無法使用它進行升級：
   a. 從此處下載最新版Trident"GitHub 上的 *Assets* 部分" 。
   b. 使用 `helm upgrade`其中命令 `trident-operator-25.06.0.tgz`反映您想要升級到的版本。

```
helm upgrade <name> trident-operator-25.06.0.tgz
```

如果在初始安裝期間設定了自訂選項（例如，為Trident和 CSI 映像指定私人映像登錄），請附加以下內容： `helm upgrade`命令使用 `--set`確保將這些選項包含在升級命令中，否則這些值將重設為預設值。

2. 跑步 `helm list`確認圖表和應用程式版本均已升級。跑步 `tridentctl logs`查看所有調試資訊。

從 `tridentctl`安裝到Trident操作員

您可以從下列位置升級到最新版本的Trident運算子： `tridentctl`安裝。現有的後端和PVC將自動可用。

在切換安裝方法之前，請先查看"安裝方法之間的轉換"。

步驟

1. 下載最新版的Trident 。

```
# Download the release required [25.06.0]
mkdir 25.06.0
cd 25.06.0
wget
https://github.com/NetApp/trident/releases/download/v25.06.0/trident-
installer-25.06.0.tar.gz
tar -xf trident-installer-25.06.0.tar.gz
cd trident-installer
```

2. 創建 `tridentorchestrator`來自清單文件的 CRD。

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

3. 在相同命名空間中部署叢集範圍的操作符。

```
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. 創建一個 `TridentOrchestrator`安裝Trident的 CR。

```
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已升級至預期版本。

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v25.06.0
```

使用 tridentctl 進行升級

您可以使用以下方法輕鬆升級現有的Trident安裝 `tridentctl`。

關於此任務

解除安裝並重新安裝Trident相當於升級。卸載Trident時， Trident部署使用的持久卷聲明 (PVC) 和持久卷 (PV) 不會被刪除。在Trident離線期間，已設定的 PV 仍將可用； Trident恢復線上後，將為在此期間建立的任何 PVC 設定磁碟區。

開始之前

審查"選擇升級方式"升級前使用 `tridentctl`。

步驟

1. 運行卸載命令 `tridentctl`移除與Trident關聯的所有資源，但保留 CRD 和相關物件。

```
./tridentctl uninstall -n <namespace>
```

2. 重新安裝Trident。參考"使用 tridentctl 安裝Trident"。

> 請勿中斷升級過程。確保安裝程式運行完成。

使用 tridentctl 管理Trident

這 "Trident安裝包"包括 `tridentctl`用於提供對Trident 的簡單存取的命令列實用程式。擁有足夠權限的 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。

命令和全域標誌

你可以運行 `tridentctl help`取得可用命令列表 `tridentctl`或附加 `--help`為任何指令新增標誌，即可取得該特定指令的選項和標誌清單。

```
tridentctl [command] [--optional-flag]
```

Trident `tridentctl`該實用程式支援以下命令和全域標誌。

命令

create

為Trident新增資源。

delete

從Trident移除一個或多個資源。

get

從Trident獲取一項或多項資源。

help

關於任何命令的幫助。

images

列印一份Trident所需容器影像的表格。

import

將現有資源導入Trident。

install

安裝Trident。

logs

列印Trident的日誌。

send

從Trident發送資源。

uninstall

卸載Trident。

update

在Trident中修改資源。

update backend state

暫時停止後端運轉。

upgrade

在Trident中升級資源。

version

列印Trident版本。

全域標誌

-d，--debug

調試輸出。

-h，--help

幫助 `tridentctl`。

-k，--kubeconfig string

指定 `KUBECONFIG`本地運行命令或從一個 Kubernetes 叢集向另一個 Kubernetes 叢集運行命令的路徑。

> 或者，您可以匯出 `KUBECONFIG`變數指向特定的 Kubernetes 叢集並提出問題 `tridentctl`向該集群發出命令。

-n，--namespace string

Trident部署的命名空間。

-o，--output string

輸出格式。 json|yaml|name|wide|ps（預設）之一。

-s，--server string

Trident REST介面的位址/連接埠。

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

命令選項和標誌

創造

使用 `create`向Trident新增資源的指令。

```
tridentctl create [option]
```

選項

`backend`為Trident添加後端。

刪除

使用 `delete`從Trident移除一個或多個資源的指令。

```
tridentctl delete [option]
```

選項

`backend`從Trident刪除一個或多個儲存後端。
`snapshot`從Trident刪除一個或多個磁碟區快照。

`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。

得到

使用 `get`從Trident取得一個或多個資源的指令。

```
tridentctl get [option]
```

選項

`backend`從Trident取得一個或多個儲存後端。
`snapshot`從Trident取得一個或多個快照。
`storageclass`從Trident取得一個或多個儲存類別。
`volume`從Trident取得一卷或多卷。

旗幟

`-h，--help`：有關卷的幫助。
`--parentOfSubordinate string`：將查詢限制在子來源磁碟區上。
`--subordinateOf string`：將查詢限制在磁碟區的下級。

圖片

使用 `images`用於列印Trident所需容器圖像表格的標誌。

```
tridentctl images [flags]
```

旗幟

`-h，--help`圖片幫助。
`-v，--k8s-version string` Kubernetes 叢集的語意版本。

進口量

使用 `import volume`將現有磁碟區導入Trident 的指令。

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

別名

`volume，v`

旗幟

`-f，--filename string`：YAML 或 JSON PVC 檔案的路徑。
`-h，--help`：音量幫助。
`--no-manage`僅建立 PV/PVC。不要想當然地認為有銷量生命週期管理。

安裝

使用 `install`安裝Trident 的標誌。

```
tridentctl install [flags]
```

旗幟

--autosupport-image string：自動支援遙測的容器映像（預設為「netapp/trident autosupport:<目前版本>」）。
--autosupport-proxy string`用於發送 Autosupport 遙測資料的代理位址/連接埠。
`--enable-node-prep`嘗試在節點上安裝所需的軟體包。
`--generate-custom-yaml`無需安裝任何軟體即可產生 YAML 檔案。
`-h ，--help`安裝幫助。
`--http-request-timeout ：覆蓋Trident控制器 REST API 的 HTTP 請求逾時時間（預設為 1 分 30 秒）。
--image-registry string ：內部鏡像倉庫的位址/連接埠。
--k8s-timeout duration ：所有 Kubernetes 操作的逾時時間（預設 3 分鐘）。
--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）（預設為「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來管理磁碟區。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，確保它不存在（預設值為「trident」）。
--pvc string ： Trident使用的舊版 PVC 的名稱，確保它不存在（預設值為「trident」）。
--silence-autosupport ：不要自動向NetApp傳送自動支援包（預設為 true）。
--silent`安裝過程中會停用大部分輸出。
`--trident-image string`要安裝的Trident鏡像。
`--k8s-api-qps Kubernetes API 請求的每秒查詢數 (QPS) 限制（預設為 100；可選）。
--use-custom-yaml`使用安裝目錄中已存在的任何 YAML 檔案。
`--use-ipv6 ：Trident 通訊使用 IPv6。

紀錄

使用 `logs`用於列印Trident日誌的標誌。

```
tridentctl logs [flags]
```

旗幟

-a，--archive`除非另有規定，否則請建立包含所有日誌的支援存檔。
`-h ，--help ：日誌幫助。
-l ，--log string`要顯示的Trident日誌。 trident|auto|trident-operator|all（預設值「auto」）之一。
`--node string ：要從中收集節點 pod 日誌的 Kubernetes 節點名稱。
-p ，--previous ：取得先前容器實例的日誌（如果存在）。
`--sidecars`取得邊車容器的日誌。

傳送

使用 `send`從Trident發送資源的指令。

```
tridentctl send [option]
```

選項

`autosupport`將 Autosupport 歸檔檔案傳送給NetApp。

解除安裝

使用 `uninstall`卸載Trident 的標誌。

```
tridentctl uninstall [flags]
```

旗幟

`-h, --help`：卸載幫助。
`--silent`卸載過程中會停用大部分輸出。

更新

使用 `update`在Trident中修改資源的指令。

```
tridentctl update [option]
```

選項

`backend`：在Trident中更新後端。

更新後端狀態

使用 `update backend state`暫停或恢復後端操作的命令。

```
tridentctl update backend state <backend-name> [flag]
```

需要考慮的幾點

- 如果後端是使用 TridentBackendConfig (tbc) 建立的，則無法使用下列方式更新後端： `backend.json`文件。
- 如果 `userState`該值已在待辦事項清單中設置，無法使用以下方式修改： `tridentctl update backend state <backend-name> --user-state suspended/normal`命令。
- 重新獲得設定能力 `userState`通過 tridentctl，在通過 tbc 設定之後， `userState`必須從待辦事項清單中移除該欄位。這可以透過以下方式實現： `kubectl edit tbc`命令。之後 `userState`欄位已移除，您可以使用 `tridentctl update backend state`更改命令 `userState`後端。
- 使用 `tridentctl update backend state`改變 `userState`。您也可以更新 `userState`使用 `TridentBackendConfig`或者 `backend.json`文件；這將觸發後端完全重新初始化，可能會很耗時。

  旗幟

  `-h， --help`：後端狀態幫助。
  `--user-state`設定為 `suspended`暫停後端操作。設定為 `normal`恢復後端操作。設定為 `suspended`：

- `AddVolume`和 `Import Volume`暫停。
- `CloneVolume， ResizeVolume ， PublishVolume ， UnPublishVolume ， CreateSnapshot ， GetSnapshot ， RestoreSnapshot ， DeleteSnapshot ， RemoveVolume ， GetVolumeExternal ， `ReconcileNodeAccess`仍然可用。

您也可以使用以下方式更新後端狀態 `userState`後端設定檔中的字段 `TridentBackendConfig`或者 `backend.json`。更多信息，請參閱"後端管理選項"和"使用 kubectl 執行後端管理"。

例：

JSON

請依照以下步驟進行更新 `userState`使用 `backend.json`文件：

1. 編輯 `backend.json`要包含的文件 `userState`字段值設定為“已暫停”。
2. 使用以下方式更新後端 `tridentctl update backend`命令和更新後的路徑 `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

套用待辦事項後，您可以使用以下方法進行編輯：`kubectl edit <tbc-name> -n <namespace>`命令。以下範例使用以下方式將後端狀態更新為掛起：`userState: suspended`選項：

```
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`標誌用於列印版本 `tridentctl`以及正在運行的Trident服務。

```
tridentctl version [flags]
```

旗幟

```
--client：僅限客戶端版本（無需伺服器）。
-h, --help ：版本幫助。
```

插件支援

Tridentctl 支援類似 kubectl 的插件。如果外掛程式二進位檔案名稱遵循「tridentctl-<plugin>」格式，且該二進位檔案位於 PATH 環境變數列出的資料夾中，則 Tridentctl 會偵測到該外掛程式。所有偵測到的外掛程式都列在 tridentctl 幫助的插件部分。您也可以透過在環境變數 TRIDENTCTL_PLUGIN_PATH 中指定插件資料夾來限制搜尋範圍（例如： `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/` ）。如果使用該變量，tridenctl 將僅在指定的資料夾中搜尋。

監視Trident

Trident提供了一組 Prometheus 指標端點，您可以使用這些端點來監控Trident 的效能。

概況

Trident提供的指標讓您可以執行以下操作：

- 密切注意 Trident 的健康狀況和配置。您可以檢查操作是否成功，以及是否能如預期與後端通訊。
- 檢查後端使用資訊，了解後端配置了多少磁碟區、消耗了多少空間等等。
- 維護可用後端已配置卷量的映射表。
- 賽道表現。您可以查看Trident與後端通訊和執行作業所需的時間。

> ⓘ 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*。

你需要什麼

- 已安裝Trident的 Kubernetes 叢集。
- 普羅米修斯實例。這可能是一個 "容器化的 Prometheus 部署"或者您可以選擇以某種方式運行 Prometheus "本機應用程式" 。

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。這 "部落格" 解釋如何將 Prometheus 和 Grafana 與Trident結合使用來檢索指標。這篇部落格解釋瞭如何在 Kubernetes 叢集中以 Operator 的形式運行 Prometheus，以及如何建立 ServiceMonitor 來取得Trident指標。

步驟 2：建立 Prometheus 服務監視器

要使用Trident指標，您應該建立一個 Prometheus ServiceMonitor 來監視 `trident-csi`服務和監聽 `metrics`港口。一個範例 ServiceMonitor 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Trident直接提供的指標外，kubelet 還公開了許多其他指標。 `kubelet_volume_*`透過其自身的指標端點獲取指標。 Kubelet 可以提供有關已附加磁碟區、Pod 以及它處理的其他內部操作的資訊。參考 "這裡"。

步驟 3：使用 PromQL 查詢Trident指標

PromQL 非常適合建立返回時間序列或表格資料的表達式。

以下是您可以使用的 PromQL 查詢：

獲取Trident健康資訊

- Trident伺服器 HTTP 2XX 回應的百分比

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Trident REST** 回應狀態碼百分比

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Trident**執行操作的平均持續時間（毫秒）

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

獲取**Trident**使用信息

- 平均體積大小

```
trident_volume_allocated_bytes/trident_volume_count
```

- 各後端分配的總儲存空間

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

取得個人用量

只有同時收集 kubelet 指標時，此功能才會啟用。

- 各卷冊的空間利用率

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

了解**Trident AutoSupport**遙測技術

預設情況下， Trident每天向NetApp發送 Prometheus 指標和基本後端資訊。

- 若要阻止Trident向NetApp發送 Prometheus 指標和基本後端訊息，請傳遞以下參數： `--silence-autosupport`在Trident安裝過程中發出警報。
- Trident也可以按需透過以下方式將容器日誌傳送給NetApp支援： `tridentctl send autosupport` 。您需要觸發Trident上傳其日誌。在提交日誌之前，您應該接受 NetApp 的條款。https://www.netapp.com/company/legal/privacy-policy/["隱私權政策"^] 。
- 除非另有說明， Trident會取得過去 24 小時的日誌。
- 您可以使用以下方式指定日誌保留時間範圍： `--since`旗幟。例如： `tridentctl send autosupport --since=1h` 。這些資訊透過以下方式收集和發送： `trident-autosupport`與Trident一起安裝的容器。您可以從以下位置取得容器鏡像： "Trident AutoSupport" 。
- Trident AutoSupport不會收集或傳輸個人識別資訊 (PII) 或個人資訊。它附帶一個 "最終用戶許可協議"這並不適用於Trident容器鏡像本身。您可以了解更多關於NetApp對資料安全和信任的承諾。 "這裡" 。

Trident發送的有效載荷範例如下所示：

```
---
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
```

- AutoSupport訊息會傳送到 NetApp 的AutoSupport端點。如果您使用私有鏡像倉庫來存放容器鏡像，則可以使用 `--image-registry`旗幟。
- 您也可以透過產生安裝 YAML 檔案來設定代理 URL。這可以透過使用 `tridentctl install --generate-custom-yaml`建立 `YAML` 檔案並新增 `--proxy-url`論證 `trident-autosupport`容器 `trident-deployment.yaml`。

禁用Trident指標

若要停用指標報告，您應該產生自訂 YAML 檔案（使用下列方式）：`--generate-custom-yaml`標記）並編輯它們以刪除 `--metrics`阻止對 `trident-main`容器。

解除安裝Trident

卸載Trident時，應使用與安裝Trident時相同的方法。

關於此任務

- 如果升級後出現錯誤、依賴項問題或升級失敗/不完整，需要修復，則應卸載Trident並按照相應的說明重新安裝早期版本。"版本" 。這是降級到早期版本的唯一推薦方法。
- 為了方便升級和重新安裝，解除安裝Trident不會刪除Trident建立的 CRD 或相關物件。如果您需要徹底刪除Trident及其所有數據，請參閱以下內容："徹底移除Trident和CRDs" 。

開始之前

如果您要停用 Kubernetes 集群，則必須先刪除所有使用Trident建立的磁碟區的應用程序，然後再卸載。這樣可以確保在 Kubernetes 節點上刪除 PVC 之前，PVC 會被取消發布。

確定原始安裝方法

卸載Trident時，應使用與安裝 Trident 時相同的方法。解除安裝前，請確認您最初安裝Trident時所使用的版本。

1. 使用 `kubectl get pods -n trident`檢查豆莢。
   - 如果沒有操作員艙，Trident是透過以下方式安裝的： `tridentctl` 。
   - 如果存在 operator pod，則Trident是透過Trident operator 手動安裝的，或使用 Helm 安裝的。
2. 如果有操作員艙，請使用 `kubectl describe tproc trident`確定Trident是否使用 Helm 安裝。
   - 如果有 Helm 標籤，表示Trident是使用 Helm 安裝的。
   - 如果沒有 Helm 標籤，則表示Trident是使用Trident操作員手動安裝的。

卸載**Trident**操作員安裝程序

您可以手動解除安裝 Trident Operator 安裝，也可以使用 Helm 解除安裝。

手動解除安裝

如果您使用操作員模式安裝了Trident ，則可以透過下列方法之一將其解除安裝：

1. 編輯 **`TridentOrchestrator`CR** 並設定卸載標誌：

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

   當 `uninstall`標誌位已設定為 `true`Trident運算子會卸載Trident，但不會移除 TridentOrchestrator 本身。如果您想再次安裝Trident，則應該清理 TridentOrchestrator 並建立一個新的。

2. 刪除 **`TridentOrchestrator`**：透過移除 `TridentOrchestrator`如果 CR 用於部署Trident，則指示操作員卸載Trident。操作員處理移除操作 `TridentOrchestrator`然後繼續移除Trident部署和守護程式集，並刪除安裝過程中建立的Trident pod。

   ```
   kubectl delete -f deploy/<bundle.yaml> -n <namespace>
   ```

解除安裝 **Helm** 安裝

如果您使用 Helm 安裝了Trident ，則可以使用以下命令卸載它： `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`命令 `tridentctl`移除與Trident關聯的所有資源，但保留 CRD 及其相關物件：

```
./tridentctl uninstall -n <namespace>
```

Trident for Docker

部署先決條件

在部署Trident之前，您必須在主機上安裝和設定必要的協定先決條件。

核實要求

- 確認您的部署滿足所有要求"要求"。
- 請確認您已安裝受支援的 Docker 版本。如果您的 Docker 版本已過時， "安裝或更新它" 。

```
docker --version
```

- 請確認您的主機上已安裝並設定協定所需的先決條件。

NFS 工具

使用適用於您作業系統的命令安裝 NFS 工具。

RHEL 8+

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

Ubuntu

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

安裝 NFS 工具後重新啟動工作節點，以防止將磁碟區附加到容器時發生故障。

iSCSI 工具

使用適用於您作業系統的指令安裝 iSCSI 工具。

RHEL 8+

1. 安裝以下系統軟體包：

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-
mapper-multipath
```

2. 請檢查 iscsi-initiator-utils 版本是否為 6.2.0.874-2.el7 或更高版本：

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

3. 將掃描方式設定為手動：

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

4. 啟用多路徑：

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

確保 `etc/multipath.conf`包含 `find_multipaths no`在下面 `defaults`。

5. 確保 `iscsid`和 `multipathd`正在運行：

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

6. 啟用並啟動 `iscsi`：

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

Ubuntu

1. 安裝以下系統軟體包：

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

2. 請檢查 open-iscsi 版本是否為 2.0.874-5ubuntu2.10 或更高版本（適用於 bionic）或 2.0.874-7.1ubuntu6.1 或更高版本（適用於 focal）：

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

3. 將掃描方式設定為手動：

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

4. 啟用多路徑：

```
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`在下面 `defaults`。

5. 確保 `open-iscsi`和 `multipath-tools`已啟用並正在運行：

```
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指令安裝 NVMe 工具。

- NVMe 需要 RHEL 9 或更高版本。
- 如果您的 Kubernetes 節點的核心版本太舊，或者您的核心版本沒有 NVMe 軟體包，則您可能需要將節點的核心版本更新為包含 NVMe 軟體包的版本。

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

使用適用於您作業系統的命令安裝 FC 工具。

- 當使用執行 RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) 且具有 FC PV 的工作節點時，請指定 `discard`StorageClass 中的 mountOption 用於執行內嵌空間回收。參考 "紅帽文檔"。

RHEL 8+

1. 安裝以下系統軟體包：

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

2. 啟用多路徑：

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

確保 `etc/multipath.conf`包含 `find_multipaths no`在下面 `defaults`。

3. 確保 `multipathd`正在運行：

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

Ubuntu

1. 安裝以下系統軟體包：

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

2. 啟用多路徑：

```
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`在下面 `defaults`。

3. 確保 `multipath-tools`已啟用並正在運行：

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

部署Trident

Trident for Docker 為NetApp儲存平台提供與 Docker 生態系統的直接整合。它支援從儲存平台到 Docker 主機的儲存資源配置和管理，並提供了一個框架，以便將來添加其他平台。

在同一台主機上可以同時執行多個Trident實例。這樣就可以同時連接到多個儲存系統和儲存類型，並且可以自訂 Docker 磁碟區使用的儲存。

你需要什麼

查看"部署的先決條件"。確保滿足先決條件後，即可部署Trident。

Docker 管理的插件方法（版本 1.13/17.03 及更高版本）

開始之前

如果您之前在 Docker 1.13/17.03 之前的版本中使用過Trident 的傳統守護程序方法，請確保在使用託管外掛程式方法之前停止Trident程序並重新啟動 Docker 守護程序。

1. 停止所有正在運行的實例：

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. 重啟 Docker。

```
systemctl restart docker
```

3. 請確保您已安裝 Docker Engine 17.03（新版本 1.13）或更高版本。

```
docker --version
```

如果您的版本已過時， "安裝或更新您的安裝程序" 。

步驟

1. 建立設定檔並如下指定選項：
   - `config`預設檔名是 `config.json`不過，您可以透過指定名稱來使用您選擇的任何名稱。 `config`檔案名稱選項。設定檔必須位於 `/etc/netappdvp`主機系統上的目錄。
   - `log-level`：指定日誌等級(`debug`， `info` ， `warn` ， `error` ， `fatal` ）。預設值為 `info`。
   - `debug`：指定是否啟用調試日誌記錄。預設值為 false。如果為真，則覆蓋日誌等級。
     i. 建立設定檔存放位置：

```
sudo mkdir -p /etc/netappdvp
```

ii. 建立設定檔：

```
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. 使用託管外掛系統啟動Trident 。代替 `<version>`使用您正在使用的插件版本（xxx.xx.x）。

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:<version> config=myConfigFile.json
```

3. 開始使用Trident從已設定的系統中取得儲存空間。
   a. 建立一個名為「firstVolume」的磁碟區：

```
docker volume create -d netapp --name firstVolume
```

   b. 容器啟動時建立預設磁碟區：

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

   c. 移除音量“firstVolume”：

```
docker volume rm firstVolume
```

傳統方法（版本 **1.12** 或更早版本）

開始之前

1. 請確保您使用的是 Docker 版本 1.10 或更高版本。

```
docker --version
```

如果您的版本過舊，請更新您的安裝。

```
curl -fsSL https://get.docker.com/ | sh
```

或者， "請按照您的分發說明進行操作" 。

2. 請確保您的系統已配置 NFS 和/或 iSCSI。

步驟

1. 安裝並配置NetApp Docker 磁碟區外掛程式：

a. 下載並解壓縮應用程式：

```
wget
https://github.com/NetApp/trident/releases/download/v25.06.0/trident-
installer-25.06.0.tar.gz
tar zxf trident-installer-25.06.0.tar.gz
```

b. 移動到回收站路徑中的某個位置：

```
sudo mv trident-installer/extras/bin/trident /usr/local/bin/
sudo chown root:root /usr/local/bin/trident
sudo chmod 755 /usr/local/bin/trident
```

c. 建立設定檔存放位置：

```
sudo mkdir -p /etc/netappdvp
```

d. 建立設定檔：

```
cat << EOF > /etc/netappdvp/ontap-nas.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. 放置好二進位檔案並建立設定檔後，使用所需的設定檔啟動Trident守護程式。

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```

除非另有指定，磁碟區驅動程式的預設名稱為「netapp」。

守護程序啟動後，您可以使用 Docker CLI 介面建立和管理磁碟區。

3. 建立磁碟區：

```
docker volume create -d netapp --name trident_1
```

4. 啟動容器時配置 Docker 磁碟區：

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. 刪除 Docker 磁碟區：

```
docker volume rm trident_1
```

```
docker volume rm trident_2
```

系統啟動時啟動Trident

systemd 系統的範例單元檔案可在以下位置找到： `contrib/trident.service.example`在 Git 倉庫中。若要在 RHEL 系統中使用該文件，請執行下列操作：

1. 將文件複製到正確位置。

   如果運行多個實例，則應為單元檔案使用唯一的名稱。

   ```
   cp contrib/trident.service.example
   /usr/lib/systemd/system/trident.service
   ```

2. 編輯該文件，將描述（第 2 行）變更為與驅動程式名稱匹配，並將設定檔路徑（第 9 行）變更為反映您的環境。
3. 重新載入 systemd 以使其吸收變更：

   ```
   systemctl daemon-reload
   ```

4. 啟用該服務。

   這個名稱會根據你給文件命名的方式而有所不同。 `/usr/lib/systemd/system`目錄。

   ```
   systemctl enable trident
   ```

5. 啟動服務。

   ```
   systemctl start trident
   ```

6. 查看狀態。

   ```
   systemctl status trident
   ```

每次修改單元檔案後，請執行以下命令： `systemctl daemon-reload`命令使其能夠感知這些變化。

升級或解除Trident

您可以安全地升級Trident for Docker，而不會對正在使用的磁碟區產生任何影響。升級過程中會有一段短暫的時間， `docker volume`針對該插件的命令將不會成功，應用程式將無法掛載卷，直到該插件再次運行。大多數情況下，這只需要幾秒鐘。

升級

請依照下列步驟升級Trident for Docker。

步驟

1. 列出現有捲冊：

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. 禁用插件：

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. 升級插件：

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

> Trident 18.01 版本取代了 nDVP。你應該直接從以下位置升級： `netapp/ndvp-plugin`圖片到`netapp/trident-plugin`圖像。

4. 啟用插件：

```
docker plugin enable netapp:latest
```

5. 請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. 確認卷可見：

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

如果您從舊版的Trident （20.10 之前）升級到Trident 20.10 或更高版本，則可能會遇到錯誤。更多信息，請參閱"已知問題"。如果遇到此錯誤，您應該先停用該插件，然後刪除該插件，最後透過傳遞額外的配置參數來安裝所需的Trident版本： `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

解除安裝

請依照下列步驟解除安裝Trident for Docker。

步驟

1. 刪除插件所建立的所有磁碟區。
2. 禁用插件：

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. 移除插件：

```
docker plugin rm netapp:latest
```

使用卷

您可以使用標準方法輕鬆建立、複製和刪除卷 `docker volume`需要時，請使用指定Trident驅動程式名稱的指令。

創建卷

- 使用預設名稱建立帶有驅動程式的磁碟區：

```
docker volume create -d netapp --name firstVolume
```

- 建立包含特定Trident實例的磁碟區：

```
docker volume create -d ntap_bronze --name bronzeVolume
```

如果您沒有指定任何內容"選項"使用的是驅動程式的預設設定。

- 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
```

磁碟區大小以字串形式表示，其中包含一個整數值，單位可選（例如：10G、20GB、3TiB）。如果沒有指定單位，則預設值為 G。大小單位可以表示為 2 的冪（B、KiB、MiB、GiB、TiB）或 10 的冪（B、KB、MB、GB、TB）。簡寫單位使用 2 的冪（G = GiB，T = TiB，...）。

移除音量

- 刪除該磁碟區的方式與其他 Docker 磁碟區一樣：

```
docker volume rm firstVolume
```

使用時 `solidfire-san`上述驅動程式範例刪除並清除磁碟區。

請依照下列步驟升級Trident for Docker。

複製卷

使用時 `ontap-nas`， `ontap-san` ， `solidfire-san` ， 和 `gcp-cvs storage drivers`Trident可以克隆卷。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>
```

- 從現有磁碟區建立新磁碟區。這將導致創建一個新的快照：

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume>
```

- 從現有磁碟區上的快照建立新磁碟區。這不會創建新的快照：

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

例子

```
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
```

存取外部建立的捲

只有當外部建立的區塊裝置（或其複製）沒有分割且其檔案系統受Trident支援時（例如：一個），容器才能使用Trident存取這些區塊裝置（或其複製）。 `ext4`格式化 `/dev/sdc1`無法透過Trident訪問）。

驅動程式特定音量選項

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這些選項以自訂結果。請參閱下方適用於您配置的儲存系統的選項。

在建立磁碟區操作期間使用這些選項非常簡單。提供選項和值 `-o`CLI 操作期間的運算子。這些值會覆寫 JSON 設定檔中的任何等效值。

ONTAP容量選項

NFS、iSCSI 和 FC 的磁碟區建立選項包括以下幾種：

選項	描述
`size`	卷大小預設為 1 GiB。
`spaceReserve`	容量配置方式可以是精簡配置或厚配置，預設為精簡配置。有效值為 `none`（精簡配置）和 `volume`（厚配置）。
`snapshotPolicy`	這將把快照策略設定為所需值。預設值為 `none`這意味著不會自動為該磁碟區建立快照。除非儲存管理員修改，否則所有ONTAP系統上都存在一個名為「default」的策略，該策略會建立並保留 6 個每小時快照、2 個每日快照和 2 個每週快照。可以透過瀏覽來還原快照中保存的資料。 `.snapshot`卷中任意目錄中的目錄。
`snapshotReserve`	這將把快照預留量設定為所需的百分比。預設值為空，這表示如果您選擇了 snapshotPolicy， ONTAP將選擇 snapshotReserve（通常為 5%）；如果 snapshotPolicy 為 none，則 ONTAP 將選擇 snapshotReserve（通常為 5%）。您可以在設定檔中為所有ONTAP後端設定預設 snapshotReserve 值，並且可以將其用作除 ontap-nas-economy 之外的所有ONTAP後端的磁碟區建立選項。
`splitOnClone`	克隆卷時，這將導致ONTAP立即將克隆卷與其父卷分離。預設值為 `false`。有些克隆卷的用例最好在創建後立即將克隆卷與其父卷分離，因為不太可能有機會提高儲存效率。例如，克隆一個空資料庫可以節省大量時間，但節省的儲存空間卻很少，因此最好立即拆分克隆。
`encryption`	在新磁碟區啟用NetApp磁碟區加密 (NVE)；預設為 `false`。若要使用此選項，必須在叢集上取得 NVE 許可並啟用 NVE。 如果後端啟用了 NAE，則在Trident中配置的任何磁碟區都會啟用 NAE。 更多信息，請參閱："Trident如何與 NVE 和 NAE 協同工作" 。
`tieringPolicy`	設定磁碟區要使用的分層策略。這決定了當資料變成不活動狀態（冷資料）時，是否將其遷移到雲層。

以下附加選項僅適用於 NFS：

選項	描述
unixPermissions	這控製卷本身的權限集。預設情況下，權限將設定為``---rwxr-xr-x`或以數字表示法表示為 0755，且 `root`將成為所有者。文字格式或數字格式均可。
snapshotDir	將其設定為 `true`將使 `.snapshot`用戶端存取該磁碟區時可見的目錄。預設值為 `false`這意味著可見性 `.snapshot`目錄預設為禁用狀態。某些鏡像，例如官方的 MySQL 鏡像，在以下情況下無法如預期運作：`.snapshot`目錄可見。
exportPolicy	設定該磁碟區要使用的匯出策略。預設值為 default。
securityStyle	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。

以下附加選項僅適用於 iSCSI：

選項	描述
fileSystemType	設定用於格式化 iSCSI 磁碟區的檔案系統。預設值為 ext4。有效值為 ext3， ext4 ， 和 xfs。
spaceAllocation	將其設定為 `false`將關閉 LUN 的空間分配功能。預設值為 `true`這表示當磁碟區空間不足且磁碟區中的 LUN 無法接受寫入時， ONTAP會通知主機。此選項還允許ONTAP在主機刪除資料時自動回收空間。

範例

請看以下範例：

- 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
```

- 建立一個啟用了 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，且快照策略為 `none`Trident使用 0% 的快照儲備。

- 建立一個沒有快照策略和快照保留的磁碟區：

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- 建立一個沒有快照策略且自訂快照保留比例為 10% 的磁碟區：

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- 建立一個具有快照策略和 10% 自訂快照保留空間的磁碟區：

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- 建立具有快照策略的捲，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 軟體音量選項

Element 軟體選項會顯示與磁碟區相關的大小和服務品質 (QoS) 策略。建立磁碟區時，使用下列方式指定與其關聯的 QoS 策略： `-o type=service_level`命名法。

使用 Element 驅動程式定義 QoS 服務等級的第一步是建立至少一個類型，並在設定檔中指定與名稱關聯的最小、最大和突發 IOPS。

Element軟體的其他磁碟區建立選項包括以下幾種：

選項	描述
`size`	卷的大小，預設為 1 GiB 或配置條目…“defaults” ：{“size”：“5G”}。
`blocksize`	可使用 512 或 4096，預設值為 512 或設定項 DefaultBlockSize。

例子

請參閱以下包含 QoS 定義的範例設定檔：

```
{
  "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
      }
    }
  ]
}
```

在上述配置中，我們有三個策略定義：青銅、白銀和黃金。這些名稱是隨意起的。

- 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
```

收集日誌

您可以收集日誌以協助進行故障排除。收集日誌的方法取決於您執行 Docker 外掛程式的方式。

收集日誌以進行故障排除

步驟

1. 如果您使用建議的託管外掛程式方法執行Trident （即，使用 `docker plugin`命令），可以這樣理解：

```
docker plugin ls
```

```
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin    false
journalctl -u docker | grep 4fb97d2b956b
```

   標準日誌等級應該足以診斷大多數問題。如果這還不夠，您可以啟用偵錯日誌記錄。

2. 若要啟用偵錯日誌記錄，請安裝啟用調試日誌記錄功能的外掛程式：

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

   或者，在插件已安裝的情況下啟用調試日誌記錄：

```
docker plugin disable <plugin>
```

```
docker plugin set <plugin> debug=true
```

```
docker plugin enable <plugin>
```

3. 如果您在主機上執行該二進位文件，則日誌位於主機的目錄中。 `/var/log/netappdvp`目錄。若要啟用偵錯日誌記錄，請指定 `-debug`當你運行插件。

一般故障排除技巧

- 新用戶遇到的最常見問題是配置錯誤，導致插件無法初始化。發生這種情況時，當您嘗試安裝或啟用外掛程式時，可能會看到類似這樣的訊息：

```
Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock:
connect: no such file or directory
```

這意味著插件啟動失敗。幸運的是，該插件內建了全面的日誌記錄功能，應該可以幫助您診斷可能遇到的大多數問題。

- 如果將光電系統安裝到貨櫃上遇到問題，請確保：`rpcbind`已安裝並正在運行。使用主機作業系統所需的軟體包管理器並檢查是否`rpcbind`正在運行。您可以透過執行以下命令來檢查 rpcbind 服務的狀態：`systemctl status rpcbind`或其等效物。

管理多個Trident實例

當您需要同時擁有多個儲存配置時，需要多個Trident實例。實作多個實例的關鍵在於使用不同的名稱來命名它們。`--alias`使用容器化插件的選項，或者`--volume-driver`在主機上實例化Trident時的選擇。

Docker 管理外掛程式（版本 1.13/17.03 或更高版本）的步驟

1. 啟動第一個實例時，指定別名和設定檔。

```
docker plugin install --grant-all-permissions --alias silver
netapp/trident-plugin:21.07 config=silver.json
```

2. 啟動第二個實例，指定不同的別名和設定檔。

```
docker plugin install --grant-all-permissions --alias gold
netapp/trident-plugin:21.07 config=gold.json
```

3. 建立磁碟區時，將別名指定為驅動程式名稱。

   例如，黃金交易量：

```
docker volume create -d gold --name ntapGold
```

   例如，對於白銀交易量而言：

```
docker volume create -d silver --name ntapSilver
```

傳統方法（版本 1.12 或更早版本）的步驟

1. 使用自訂驅動程式 ID 透過 NFS 配置啟動插件：

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. 使用自訂驅動程式 ID 和 iSCSI 設定啟動插件：

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. 為每個驅動程式實例配置 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
```

儲存配置選項

查看適用於您的Trident配置的設定選項。

全域配置選項

這些配置選項適用於所有Trident配置，無論使用何種儲存平台。

選項	描述	例子
`version`	設定檔版本號	`1`
`storageDriverName`	儲存驅動程式名稱	`ontap-nas`，`ontap-san`，`ontap-nas-economy`，`ontap-nas-flexgroup`，`solidfire-san`
`storagePrefix`	卷名稱的可選前綴。預設: `netappdvp_` 。	`staging_`
`limitVolumeSize`	對容量大小有可選限制。預設值：""（不強制執行）	`10g`

請勿使用 `storagePrefix`（包括預設值）適用於 Element 後端。預設情況下， `solidfire-san` 驅動程式將忽略此設置，並且不使用前綴。 NetApp建議使用特定的租用戶 ID 進行 Docker 磁碟區映射，或使用屬性數據，該屬性資料填入了 Docker 版本、驅動程式資訊和來自 Docker 的原始名稱（如果可能使用了任何名稱修改）。

可以使用預設選項，避免在建立的每個磁碟區上都指定這些選項。這 `size`此選項適用於所有控制器類型。有關如何設定預設磁碟區大小的範例，請參閱ONTAP配置部分。

選項	描述	例子
`size`	新磁碟區的可選預設大小。預設: `1G`	`10G`

ONTAP 配置

除了上述全域配置值之外，使用ONTAP時，還可以使用下列頂級選項。

選項	描述	例子
`managementLIF`	ONTAP管理 LIF 的 IP 位址。您可以指定一個完全限定網域名稱（FQDN）。	`10.0.0.1`
`dataLIF`	LIF協定的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。 • ONTAP SAN 驅動程式*：請勿指定 iSCSI 或 FC。Trident的使用"ONTAP選擇性 LUN 地圖"發現建立多路徑會話所需的 iSCSI 或 FC LIF。如果出現以下情況，則會產生警告： `dataLIF`已明確定義。	`10.0.0.2`
`svm`	要使用的儲存虛擬機器（如果管理 LIF 是叢集 LIF，則此項目為必填項）	`svm_nfs`
`username`	連接到儲存裝置的使用者名	`vsadmin`
`password`	連接儲存裝置的密碼	`secret`

選項	描述	例子
`aggregate`	用於配置的聚合（可選；如果設置，則必須指派給 SVM）。對於 `ontap-nas-flexgroup`驅動程序，此選項將被忽略。指派給 SVM 的所有聚合都用於配置FlexGroup磁碟區。	`aggr1`
`limitAggregateUsage`	可選，如果使用率超過此百分比，則配置失敗	`75%`
`nfsMountOptions`	對 NFS 掛載選項進行精細控制；預設值為“-o nfsvers=3”。 僅限以下情況： **`ontap-nas`**和 **`ontap-nas-economy`**司機。 "請在此處查看 NFS 主機設定資訊" 。	`-o nfsvers=4`
`igroupName`	Trident建立和管理每個節點 `igroups`作為 `netappdvp`。 此值不能更改或省略。 僅限以下情況： **`ontap-san`**司機。	`netappdvp`
`limitVolumeSize`	最大可請求容量。	`300g`
`qtreesPerFlexvol`	每個FlexVol 的最大 qtree 數量必須在 [50, 300] 範圍內，預設值為 200。 對於 **`ontap-nas-economy`**驅動程序，此選項允許自訂每個**FlexVol** 的最大 qtree 數量。	`300`
`sanType`	*支持 `ontap-san`僅限司機。 *用於選擇 `iscsi`對於 iSCSI， `nvme`適用於 NVMe/TCP 或 `fcp`用於光纖通道 (FC) 上的 SCSI。	`iscsi`如果為空
`limitVolumePoolSize`	*支持 `ontap-san-economy`和 `ontap-san-economy`僅限司機。 *限制ONTAP ontap-nas-economy 和 ontap-SAN-economy 驅動程式中的FlexVol大小。	`300g`

系統提供了預設選項，避免在建立的每個磁碟區上都進行指定：

選項	描述	例子
spaceReserve	空間預約模式； none （精簡配置）或 volume（厚的）	none
snapshotPolicy	要使用的快照策略，預設值為 none	none
snapshotReserve	快照預留百分比，預設值為空字串，表示接受ONTAP預設值。	10
splitOnClone	創建時將克隆體與其父級分離，預設為 false	false
encryption	在新磁碟區啟用NetApp磁碟區加密 (NVE)；預設為啟用 false。若要使用此選項，必須在叢集上取得 NVE 許可並啟用 NVE。 如果後端啟用了 NAE，則在Trident中配置的任何磁碟區都會啟用 NAE。 更多信息，請參閱："Trident如何與 NVE 和 NAE 協同工作" 。	真的
unixPermissions	NAS 選項，用於已配置的 NFS 卷，預設值為 777	777
snapshotDir	NAS 存取選項 `.snapshot`目錄。	NFSv4 為“true”，NFSv3 為“false”。
exportPolicy	NFS匯出策略要使用的NAS選項，預設值為 default	default
securityStyle	NAS 選項，用於存取已設定的 NFS 磁碟區。 NFS 支持 mixed`和 `unix`安全措施。預設值為 `unix。	unix
fileSystemType	SAN 選項用於選擇檔案系統類型，預設為 ext4	xfs
tieringPolicy	要使用的分層策略，預設值為 none。	none

縮放選項

這 `ontap-nas`和 `ontap-san`驅動程式為每個 Docker 磁碟區建立一個ONTAP FlexVol 。 ONTAP每個叢集節點最多支援 1000 個 FlexVol，叢集最多可支援 12,000 個FlexVol磁碟區。如果您的 Docker 磁碟區需求符合此限制，則 `ontap-nas`由於 FlexVols 提供了 Docker 磁碟區粒度快照和克隆等附加功能，因此驅動程式是首選的 NAS 解決方案。

如果您需要的 Docker 磁碟區數量超過了FlexVol 的限制，請選擇 `ontap-nas-economy`或者 `ontap-san-economy`司機。

這 `ontap-nas-economy`驅動程式在自動管理的FlexVol磁碟區中建立作為ONTAP Qtree 的 Docker 磁碟區。Qtree 提供了更大的可擴展性，每個叢集節點最多可達 100,000 個節點，每個叢集最多可達 2,400,000 個節點，但代價是犧牲了一些功能。這 `ontap-nas-economy`驅動程式不支援 Docker 磁碟區粒度快照或克隆。

這 `ontap-nas-economy`Docker Swarm 目前不支援該驅動程序，因為 Docker Swarm 無法協調跨多個節點的磁碟區建立。

這 `ontap-san-economy`驅動程式在自動管理的FlexVol磁碟區的共用池中建立 Docker 磁碟區作為ONTAP LUN。這樣一來，每個FlexVol就不限於一個 LUN，並且為 SAN 工作負載提供了更好的可擴充性。根據儲存陣列的不同， ONTAP每個叢集最多支援 16384 個 LUN。由於這些磁碟區底層是 LUN，因此此驅動程式支援 Docker 磁碟區粒度快照和複製。

選擇 `ontap-nas-flexgroup`驅動程式可提高單一磁碟區的平行處理能力，該磁碟區可成長到 PB 級，包含數十億個檔案。 FlexGroups 的一些理想用例包括 AI/ML/DL、大數據和分析、軟體建置、串流媒體、檔案儲存庫等等。 Trident在配置FlexGroup磁碟區時會使用指派給 SVM 的所有聚合。 Trident中的FlexGroup支援還需考慮以下幾點：

- 需要ONTAP版本 9.2 或更高版本。
- 截至撰寫本文時，FlexGroups 僅支援 NFS v3。
- 建議為 SVM 啟用 64 位元 NFSv3 識別碼。
- 建議的最小FlexGroup成員/磁碟區大小為 100 GiB。
- FlexGroup卷不支援克隆。

有關 FlexGroup 以及適合 FlexGroup 的工作負載的信息，請參閱以下內容： "NetApp FlexGroup卷最佳實務與實作指南" 。

為了在同一環境中獲得進階功能和大規模部署，您可以執行多個 Docker Volume Plugin 實例，其中一個使用 `ontap-nas`以及另一個使用 `ontap-nas-economy`。

為Trident訂製ONTAP角色

您可以建立一個具有最低權限的ONTAP叢集角色，這樣您就不必使用ONTAP管理員角色在Trident中執行操作。在Trident後端設定中包含使用者名稱時， Trident將使用您建立的ONTAP叢集角色來執行操作。

請參閱"Trident自訂角色產生器"有關建立Trident自訂角色的詳細資訊。

使用**ONTAP CLI**

1. 使用以下命令建立新角色：

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

2. 為Trident用戶建立使用者名稱：

```
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. 將角色映射到使用者：

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

使用系統管理員

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，請選擇「叢集 > 設定」。

      （或）若要在 SVM 層級建立自訂角色，請選擇「儲存」>「儲存虛擬機器」> `required SVM` > 設定 > 使用者和角色*。

   b. 選擇“使用者和角色”旁邊的箭頭圖示（→）。
   c. 在“角色”下選擇“+添加”。
   d. 定義角色規則，然後點選「儲存」。
2. 將角色對應到Trident使用者： + 在「使用者和角色」頁面上執行下列步驟：
   a. 在「使用者」下方選擇「新增」圖示 **+**。
   b. 選擇所需的使用者名，然後在「角色」下拉式選單中選擇角色。
   c. 點選“儲存”。

更多資訊請參閱以下頁面：

- "用於管理ONTAP的自訂角色"或者"定義自訂角色"
- "與角色和使用者協作"

ONTAP設定檔範例

<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
}
```

<code>ontap-san</code>驅動程式的基於FC的SCSI範例

```
{
  "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
}
```

Element軟體配置

除了全域設定值之外，在使用 Element 軟體（NetApp HCI/ SolidFire）時，還可以使用這些選項。

選項	描述	例子
`Endpoint`	<a href="https://<登入>:<密碼>@<mvip>/json-rpc/<元素版本&gt" class="bare">https://<登入>:<密碼>@<mvip>/json-rpc/<元素版本&gt;	https://admin:admin@192.168.160.3/json-rpc/8.0
`SVIP`	iSCSI IP 位址和連接埠	10.0.0.7:3260

選項	描述	例子
`TenantName`	要使用的 SolidFireF 租用戶（如果未找到則建立）	`docker`
`InitiatorIFace`	將 iSCSI 流量限制在非預設介面時，請指定介面。	`default`
`Types`	QoS規範	請參閱下面的範例
`LegacyNamePrefix`	升級版Trident安裝的前綴。如果您使用的是 1.3.2 之前的Trident版本，並且使用現有磁碟區執行升級，則需要設定此值才能存取透過磁碟區名稱方法對應的舊磁碟區。	`netappdvp-`

這 `solidfire-san`驅動程式不支援 Docker Swarm。

範例元素軟體設定檔

```
{
  "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
      }
    }
  ]
}
```

已知問題和限制

尋找有關將Trident與 Docker 結合使用時已知問題和限制的資訊。

將**Trident Docker Volume Plugin** 從舊版升級到 **20.10** 及更高版本會導致升級失敗，並出現「沒有這樣的檔案或目錄」錯誤。

解決方法

1. 禁用插件。

```
docker plugin disable -f netapp:latest
```

2. 移除插件。

```
docker plugin rm -f netapp:latest
```

3. 重新安裝插件，並提供額外信息 `config`範圍。

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

卷名長度至少為 **2** 個字元。

這是 Docker 客戶端的限制。用戶端會將單一字元名稱解釋為 Windows 路徑。 "參見錯誤 25773" 。

Docker Swarm 的某些行為導致**Trident**無法支援它與所有儲存和驅動程式的組合。

- Docker Swarm 目前使用磁碟區名稱而不是磁碟區 ID 作為其唯一的磁碟區識別碼。
- 卷請求會同時傳送到 Swarm 叢集中的每個節點。
- 磁碟區插件（包括Trident）必須在 Swarm 叢集中的每個節點上獨立運行。由於ONTAP 的工作方式以及 `ontap-nas`和 `ontap-san`駕駛員之所以能發揮作用，是因為他們是唯一能夠在這些限制條件下操作的人。

其餘驅動程式會受到諸如競爭條件之類的問題的影響，這可能會導致單一請求創建大量卷，而沒有明確的「贏家」；例如，Element 具有允許卷具有相同名稱但 ID 不同的功能。

NetApp已向 Docker 團隊提供了回饋，但目前沒有任何跡象表明未來會採取什麼措施。

如果正在配置**FlexGroup** ，而第二個**FlexGroup**與正在配置的**FlexGroup**有一個或多個共同的聚合，則**ONTAP**不會配置第二個**FlexGroup** 。

最佳實踐和建議

部署

部署Trident時，請遵循此處列出的建議。

部署到專用命名空間

"命名空間"造成不同應用程式之間的行政隔離，並阻礙資源共享。例如，一個命名空間中的 PVC 不能在另一個命名空間中使用。 Trident為 Kubernetes 叢集中的所有命名空間提供 PV 資源，因此利用了具有提升權限的服務帳戶。

此外，存取Trident pod 可能會使使用者能夠存取儲存系統憑證和其他敏感資訊。必須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消耗

Kubernetes 具有兩個特性，這兩個特性結合起來，為限制應用程式的資源消耗提供了一種強大的機制。這 "儲存配額機制"管理員可以按命名空間實施全域和儲存類別特定的容量和物件數量消耗限制。此外，使用 "射程限制" 確保 PVC 請求在轉送給配置器之前，其值既在最小值也在最大值範圍內。

這些值是按命名空間定義的，這意味著每個命名空間都應該定義與其資源需求相符的值。請點擊此處查看相關資訊 "如何利用配額"。

儲存配置

NetApp產品組合中的每個儲存平台都具有獨特的功能，無論應用程式是否採用容器化，都能從中受益。

平台概覽

Trident可與ONTAP和 Element 搭配使用。沒有哪個平台比其他平台更適合所有應用和場景，但是，在選擇平台時，應該考慮應用程式的需求以及管理設備的團隊的需求。

您應該遵循您所使用協定的主機作業系統的最佳實務。 （可選）您可以考慮將應用程式最佳實踐（如果可用）與後端、儲存類別和 PVC 設定結合，以優化特定應用程式的儲存。

ONTAP和Cloud Volumes ONTAP最佳實踐

了解配置ONTAP和Cloud Volumes ONTAP for Trident 的最佳實務。

以下建議是為容器化工作負載配置ONTAP的指導原則，這些工作負載使用由Trident動態配置的磁碟區。每項都應根據您的環境進行考慮和評估，以確定其適用性。

使用專用於**Trident** 的**SVM**。

儲存虛擬機器 (SVM) 為ONTAP系統上的租用戶提供隔離和管理分離。將 SVM 專用於應用程式可以實現權限委派，並可以應用限制資源消耗的最佳實踐。

SVM的管理有多種選擇：

- 在後端配置中提供叢集管理接口，以及相應的憑證，並指定 SVM 名稱。
- 使用ONTAP系統管理員或 CLI 為 SVM 建立專用管理介面。
- 與 NFS 資料介面共用管理角色。

在每種情況下，介面都應該在 DNS 中，並且在設定Trident時應該使用 DNS 名稱。這有助於實現某些災難復原場景，例如無需網路身分保留的 SVM-DR。

對於 SVM 而言，採用專用管理 LIF 或共享管理 LIF 沒有偏好，但是，您應該確保您的網路安全策略與您選擇的方法保持一致。無論如何，管理 LIF 應該可以透過 DNS 訪問，以實現最大的靈活性。 "SVM-DR"可與Trident配合使用。

限制最大體積數

ONTAP儲存系統有最大卷數限制，此限制會根據軟體版本和硬體平台而有所不同。請參閱 "NetAppHardware Universe"請根據您的特定平台和ONTAP版本確定確切的限制。當磁碟區計數耗盡時，不僅Trident的設定操作會失敗，而且所有儲存請求的設定操作都會失敗。

三叉戟的 `ontap-nas`和 `ontap-san`驅動程式為每個建立的 Kubernetes 持久性磁碟區 (PV) 提供 FlexVolume。這 `ontap-nas-economy`驅動程式大約每 200 個 PV 建立一個 FlexVolume（可在 50 到 300 之間配置）。這 `ontap-san-economy`驅動程式大約每 100 個 PV 建立一個 FlexVolume（可在 50 到 200 之間配置）。為了防止Trident消耗儲存系統上所有可用的磁碟區，您應該對 SVM 設定限制。您可以從命令列執行此操作：

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

價值 `max-volumes`價格會根據您所在環境的幾個具體因素而有所不同：

- ONTAP叢集中現有磁碟區的數量
- 您預計在Trident之外為其他應用程式配置的捲數
- Kubernetes 應用程式預計使用的持久卷數量

這 `max-volumes`該值是ONTAP叢集中所有節點上配置的總磁碟區數，而不是單一ONTAP節點上的磁碟區數。因此，您可能會遇到某些情況，即ONTAP叢集節點的Trident配置磁碟區可能比其他節點多得多或少得多。

例如，一個雙節點ONTAP叢集最多可以託管 2000 個FlexVol磁碟區。將最大音量計數設為 1250 似乎非常合理。然而，如果僅僅 "聚合體"如果從一個節點指派給 SVM，或無法從一個節點指派的聚合進行配置（例如，由於容量不足），則另一個節點將成為所有Trident組態磁碟區的目標。這意味著該節點的容量限制可能在達到上限之前就已經達到。 `max-volumes`達到值後，將影響Trident和使用該節點的其他磁碟區操作。 *您可以透過確保叢集中每個節點的聚合資料以相等的數量分配給Trident使用的SVM來避免這種情況。 *

複製卷

NetApp Trident在使用時支援克隆卷 `ontap-nas`， `ontap-san` ， `solidfire-san` ， 和 `gcp-cvs`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。

避免複製與不同儲存類別關聯的PVC。在同一個 StorageClass 內執行複製操作，以確保相容性並防止意外行為。

限制**Trident**建立的最大體積。

若要設定Trident可建立的磁碟區的最大大小，請使用下列方法： `limitVolumeSize`參數 `backend.json`定義。

除了控制儲存陣列的磁碟區大小之外，您還應該利用 Kubernetes 功能。

限制**Trident**創建的**FlexVols**的最大尺寸。

若要配置用作 ontap-san-economy 和 ontap-nas-economy 驅動程式集區的 FlexVols 的最大大小，請使用下列方法： `limitVolumePoolSize`參數 `backend.json`定義。

設定**Trident**使用雙向 **CHAP**

您可以在後端定義中指定 CHAP 發起方和目標使用者名稱和密碼，並讓Trident在 SVM 上啟用 CHAP。使用 `useCHAP`在後端設定中設定參數， Trident使用 CHAP 對ONTAP後端的 iSCSI 連線進行驗證。

建立並使用 **SVM QoS** 策略

利用應用於 SVM 的ONTAP QoS 策略，可以限制Trident已配置磁碟區可消耗的 IOPS 數量。這有助於 "阻止霸凌行為"或失控的容器影響Trident SVM 以外的工作負載。

只需幾個步驟即可為 SVM 建立 QoS 策略。請參閱您所使用ONTAP版本的文件以取得最準確的資訊。下面的範例建立了一個 QoS 策略，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，如果您的ONTAP版本支持，您可以考慮使用 QoS 最低要求來確保容器化工作負載的吞吐量。自適應QoS與SVM層策略不相容。

分配給容器化工作負載的 IOPS 數量取決於許多方面。其中包括：

- 使用該儲存陣列的其他工作負載。如果還有其他與 Kubernetes 部署無關的工作負載正在使用儲存資源，則應注意確保這些工作負載不會意外受到不利影響。
- 預期工作負載將在容器中運作。如果對 IOPS 要求高的工作負載將在容器中運行，則低 QoS 策略會導致糟糕的使用者體驗。

需要注意的是，在 SVM 層級指派的 QoS 策略會導致所有指派給 SVM 的磁碟區共用同一個 IOPS 池。如果一個或少數幾個容器化應用程式對 IOPS 有很高的要求，它可能會欺負其他容器化工作負載。如果情況屬實，您可能需要考慮使用外部自動化工具來指派按磁碟區劃分的 QoS 策略。

只有當您的ONTAP版本低於 9.8 時，才應將 QoS 策略群組指派給 SVM。

為**Trident**建立 **QoS** 策略組

服務品質 (QoS) 保證關鍵工作負載的效能不會因競爭工作負載而降低。 ONTAP QoS 策略群組為磁碟區提供 QoS 選項，並允許使用者為一個或多個工作負載定義吞吐量上限。有關 QoS 的更多信息，請參閱 "透過服務品質 (QoS) 保證吞吐量。"。您可以在後端或儲存池中指定 QoS 策略群組，這些策略群組將套用於在該池或後端中建立的每個磁碟區。

ONTAP有兩種類型的 QoS 策略群：傳統型和自適應型。傳統策略群組在 IOPS 中提供固定的最大（或最小，在後續版本中）吞吐量。自適應 QoS 可根據工作負載大小自動調整吞吐量，隨著工作負載大小的變化，保持 IOPS 與 TB|GB 的比率不變。在大型部署中管理數百上千個工作負載時，這提供了顯著的優勢。

建立QoS策略群組時，請考慮以下事項：

- 你應該設定 `qosPolicy`關鍵在於 `defaults`後端配置塊。請參閱以下後端設定範例：

```
---
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
```

- 您應該按卷應用策略群組，以便每個磁碟區都能獲得策略群組指定的全部吞吐量。不支援共享策略組。

有關 QoS 策略組的更多信息，請參閱 "ONTAP指令參考"。

限制對 **Kubernetes** 叢集成員的儲存資源訪問

限制對Trident建立的 NFS 磁碟區、iSCSI LUN 和 FC LUN 的存取是 Kubernetes 部署安全態勢的關鍵組成部分。這樣做可以防止不屬於 Kubernetes 叢集的主機存取磁碟區並可能意外修改資料。

重要的是要理解命名空間是 Kubernetes 中資源的邏輯邊界。假設同一命名空間內的資源可以共享，但是，重要的是，沒有跨命名空間的功能。這意味著，即使 PV 是全域對象，當綁定到 PVC 時，它們也只能被同一命名空

間中的 pod 存取。 *務必確保在適當情況下使用命名空間來實現程式碼分離。 *

對於大多數組織而言，在 Kubernetes 環境中資料安全的主要擔憂是容器中的進程可以存取掛載到主機上的存儲，但該存儲並非為容器所預期的。 "命名空間"旨在防止此類妥協。但是，特權容器是一個例外。

特權容器是指擁有比普通容器高得多的主機級權限的容器。這些功能預設不會被停用，因此請確保使用以下命令停用該功能： "Pod 安全策略" 。

對於需要從 Kubernetes 和外部主機存取的磁碟區，儲存應以傳統方式管理，PV 由管理員引入，而不是由Trident管理。這樣可以確保只有當 Kubernetes 和外部主機都已斷開連接並且不再使用該磁碟區時，才會銷毀儲存磁碟區。此外，還可以應用自訂匯出策略，從而允許從 Kubernetes 叢集節點和 Kubernetes 叢集外部的目標伺服器進行存取。

對於具有專用基礎架構節點（例如 OpenShift）或其他無法調度使用者應用程式的節點的部署，應使用單獨的匯出策略來進一步限制對儲存資源的存取。這包括為部署到這些基礎設施節點的服務（例如，OpenShift Metrics 和 Logging 服務）以及部署到非基礎設施節點的標準應用程式建立匯出策略。

使用專門的出口政策

您應該確保每個後端都存在匯出策略，該策略僅允許存取 Kubernetes 叢集中的節點。Trident可以自動建立和管理出口政策。這樣， Trident將其提供的磁碟區的存取權限限製到 Kubernetes 叢集中的節點，並簡化了節點的新增/刪除。

或者，您也可以手動建立匯出策略，並向其中填入一條或多條匯出規則，以處理每個節點存取請求：

- 使用 `vserver export-policy create`ONTAP CLI 指令建立匯出策略。
- 使用以下方式為出口策略新增規則： `vserver export-policy rule create` ONTAP CLI 指令。

執行這些命令可以限制哪些 Kubernetes 節點可以存取資料。

停用 **`showmount`**對於 **SVM** 應用

這 `showmount`此功能使 NFS 用戶端能夠向 SVM 查詢可用 NFS 匯出清單。部署到 Kubernetes 叢集的 pod 可以發出 `showmount -e`針對該目標執行指令，並接收可用掛載點列表，包括它沒有存取權限的掛載點。雖然這本身並不構成安全隱患，但它確實提供了不必要的信息，可能會幫助未經授權的用戶連接到 NFS 匯出。

您應該禁用 `showmount`透過使用 SVM 等級ONTAP CLI 指令：

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire最佳實踐

了解配置Trident的SolidFire儲存的最佳實務。

建立 **Solidfire** 帳戶

每個SolidFire帳戶代表一個唯一的磁碟區擁有者，並接收自己的一組質詢握手身分驗證協定 (CHAP) 憑證。您可以透過帳戶名稱和對應的 CHAP 憑證或透過磁碟區存取群組來存取指派給帳戶的磁碟區。一個帳戶最多可以分配兩千卷，但一卷只能屬於一個帳戶。

建立 **QoS** 策略

如果您想要建立並儲存可套用於多個磁碟區的標準化服務品質設置，請使用SolidFire服務品質 (QoS) 策略。

您可以按磁碟區設定 QoS 參數。透過設定定義 QoS 的三個可設定參數，可以確保每個磁碟區的效能：最小 IOPS、最大 IOPS 和突發 IOPS。

以下是 4Kb 區塊大小的可能最小、最大和突發 IOPS 值。

IOPS 參數	定義	最小值	預設值	最大值（**4Kb**）
最小 IOPS	保證一定容量的性能水準。	50	50	15000
最大 IOPS	性能不會超過此限制。	50	15000	200,000
突發IOPS	短時突發場景下允許的最大IOPS。	50	15000	200,000

儘管最大 IOPS 和突發 IOPS 可以設定為高達 200,000，但磁碟區的實際最大效能受叢集使用情況和每個節點效能的限制。

資料區塊大小和頻寬對 IOPS 數量有直接影響。隨著資料塊大小的增加，系統會將頻寬增加到足以處理更大資料塊大小的水平。隨著頻寬的增加，系統能夠達到的IOPS數量會減少。請參閱 "SolidFire服務品質"有關服務品質和性能的更多資訊。

SolidFire身份驗證

Element 支援兩種驗證方法：CHAP 和磁碟區存取群組 (VAG)。 CHAP 使用 CHAP 協定對主機進行後端驗證。磁碟區存取群組控制對其所配置磁碟區的存取。 NetApp建議使用 CHAP 進行身份驗證，因為它更簡單且沒有擴充限制。

具有增強型 CSI 設定器的Trident支援使用 CHAP 驗證。 VAG 只能在傳統的非 CSI 模式下使用。

CHAP 驗證（驗證發起者是否為預期的磁碟區使用者）僅在基於帳戶的存取控制中支援。如果您使用 CHAP 進行身份驗證，則有兩種選擇：單向 CHAP 和雙向 CHAP。單向 CHAP 透過使用SolidFire帳戶名稱和發起方金鑰來驗證磁碟區存取。雙向 CHAP 選項提供了最安全的磁碟區身份驗證方式，因為磁碟區透過帳戶名稱和發起方金鑰驗證主機身份，然後主機透過帳戶名稱和目標金鑰驗證磁碟區身份。

但是，如果無法啟用 CHAP 並且需要 VAG，則建立存取群組並將主機啟動器和磁碟區新增至存取群組。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 發起程序配置為使用 CHAP 驗證，則使用基於帳戶的存取控制。如果 iSCSI 發起程序未設定為使用 CHAP 驗證，則使用磁碟區存取群組存取控制。

哪裡可以找到更多資訊？

下面列出了一些最佳實踐文件。搜尋 "NetApp函式庫"適用於最新版本。

ONTAP

- "NFS最佳實務與實施指南"
- "SAN 管理"（適用於 iSCSI）
- "RHEL 的 iSCSI Express 配置"

Element軟體

- "配置適用於 Linux 的SolidFire"

NetApp HCI

- "NetApp HCI部署先決條件"
- "存取NetApp部署引擎"

應用最佳實務資訊

- "ONTAP上 MySQL 的最佳實踐"
- "SolidFire上 MySQL 的最佳實踐"
- "NetApp SolidFire和 Cassandra"
- "Oracle 在SolidFire的最佳實踐"
- "SolidFire上的PostgreSQL最佳實踐"

並非所有應用程式都有具體的指導原則，因此與您的NetApp團隊合作並使用以下方法至關重要： "NetApp函式庫"尋找最新文檔。

整合Trident

要整合Trident，需要整合以下設計和架構元素：驅動程式選擇和部署、儲存類別設計、虛擬池設計、持久性磁碟區聲明 (PVC) 對儲存配置的影響、磁碟區操作以及使用Trident部署 OpenShift 服務。

駕駛員選擇和部署

為您的儲存系統選擇並部署後端驅動程式。

ONTAP後端驅動程式

ONTAP後端驅動程式的差異在於所使用的協定以及在儲存系統上配置磁碟區的方式。因此，在決定部署哪位駕駛員時要仔細考慮。

從更高的層面來看，如果您的應用程式具有需要共用儲存的元件（多個 pod 存取相同 PVC），則基於 NAS 的驅動程式將是預設選擇，而基於區塊的 iSCSI 驅動程式則滿足非共用儲存的需求。根據應用程式的要求以及儲存和基礎設施團隊的接受程度來選擇協議。一般來說，對於大多數應用程式來說，它們之間幾乎沒有區別，因此通常取決於是否需要共用儲存（多個 pod 需要同時存取）。

可用的ONTAP後端驅動程式有：

- `ontap-nas`每個已配置的 PV 都是一個完整的ONTAP FlexVolume。

- `ontap-nas-economy`每個已配置的 PV 都是一個 qtree，每個 FlexVolume 的 qtree 數量可配置（預設值為 200）。
- `ontap-nas-flexgroup`每個 PV 都配置為一個完整的ONTAP FlexGroup，並且分配給 SVM 的所有聚合都將被使用。
- `ontap-san`每個已配置的 PV 都是其自身 FlexVolume 中的一個 LUN。
- `ontap-san-economy`每個已配置的 PV 都是一個 LUN，每個 FlexVolume 的 LUN 數量可配置（預設值為 100）。

在三個 NAS 驅動程式之間進行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，並非所有功能都透過Trident公開。如果需要某些功能，則必須由儲存管理員在配置完成後套用這些功能。上標腳註區分了每個功能和驅動程式的特定功能。

ONTAP NAS 驅動程式	快照	複製	動態出口策略	多重附件	服務品質	調整大小	複製
`ontap-nas`	是的	是的	是的註腳：5[]	是的	是的註腳：1[]	是的	是的註腳：1[]
`ontap-nas-economy`	無註腳：3[]	無註腳：3[]	是的註腳：5[]	是的	無註腳：3[]	是的	無註腳：3[]
`ontap-nas-flexgroup`	是的註腳：1[]	不	是的註腳：5[]	是的	是的註腳：1[]	是的	是的註腳：1[]

Trident為ONTAP提供 2 款 SAN 驅動程序，其功能如下所示。

ONTAP SAN 驅動程式	快照	複製	多重附件	雙向 **CHAP**	服務品質	調整大小	複製
`ontap-san`	是的	是的	是的註腳：4[]	是的	是的註腳：1[]	是的	是的註腳：1[]
`ontap-san-economy`	是的	是的	是的註腳：4[]	是的	無註腳：3[]	是的	無註腳：3[]

> 以上表格的註腳：是註腳1[]：非Trident管理；是註腳2[]：由Trident管理，但不支援PV粒度；否註腳3[]：非Trident管理且不支援PV粒度；是註腳4[]：支援原始區塊磁碟區；是註腳5[]： Trident支援。

非 PV 粒度的功能將應用於整個 FlexVolume，所有 PV（即共享 FlexVol 中的 qtree 或 LUN）將共享一個共同的計劃。

如上表所示，大部分功能都體現在以下方面： `ontap-nas`和 `ontap-nas-economy`是一樣的。然而，因為 `ontap-nas-economy`驅動程式限制了以 `PV` 粒度控制計劃的能力，這可能會特別影響您的災難復原和備份計劃。對於希望在`ONTAP`儲存上利用 `PVC` 克隆功能的開發團隊而言，只有在使用以下配置時才有可能： `ontap-nas` ， `ontap-san`或者 `ontap-san-economy`司機。

這 `solidfire-san`驅動程式也能夠克隆PVC。

Cloud Volumes ONTAP後端驅動程式

Cloud Volumes ONTAP提供資料控制以及企業級儲存功能，適用於各種用例，包括檔案共用和區塊級存儲，支援 NAS 和 SAN 協定（NFS、SMB / CIFS 和 iSCSI）。 Cloud Volume ONTAP的相容驅動程式有： `ontap-nas` ， `ontap-nas-economy` ， `ontap-san`和 `ontap-san-economy`。這些適用於 Azure 版 Cloud Volume ONTAP和 GCP 版 Cloud Volume ONTAP 。

Amazon FSx for ONTAP後端驅動程式

Amazon FSx for NetApp ONTAP讓您能夠利用您熟悉的NetApp功能、效能和管理能力，同時享受在 AWS 上儲存資料的簡單性、敏捷性、安全性和可擴充性。 FSx for ONTAP支援許多ONTAP檔案系統功能和管理 API。 Cloud Volume ONTAP的相容驅動程式有： `ontap-nas` ， `ontap-nas-economy` ， `ontap-nas-flexgroup` ， `ontap-san`和 `ontap-san-economy`。

NetApp HCI/ SolidFire後端驅動程式

這 `solidfire-san`此驅動程式與NetApp HCI/ SolidFire平台搭配使用，可協助管理員根據 QoS 限制為Trident配置 Element 後端。如果您希望設計後端以設定Trident配置磁碟區的特定 QoS 限制，請使用下列方法： `type`後端文件中的參數。管理員還可以使用以下方法限制可在儲存體上建立的磁碟區大小： `limitVolumeSize`範圍。目前，Element 儲存功能（例如磁碟區調整大小和磁碟區複製）尚不支援透過以下方式進行儲存： `solidfire-san`司機。這些操作應該透過 Element Software 的 Web 使用者介面手動完成。

SolidFire驅動程式	快照	複製	多重附件	第章	服務品質	調整大小	複製
`solidfire-san`	是的	是的	是的註腳：2[]	是的	是的	是的	是的註腳：1[]

> 註腳：是註腳1： Trident不管理 是註腳2：支援原始塊卷

Azure NetApp Files後端驅動程式

Trident使用 `azure-netapp-files`司機管理"Azure NetApp Files"服務。

有關此驅動程式及其配置方法的更多信息，請參見此處。"用於Azure NetApp Files的Trident後端配置" 。

Azure NetApp Files驅動程式	快照	複製	多重附件	服務品質	擴張	複製
`azure-netapp-files`	是的	是的	是的	是的	是的	是的註腳：1[]

> 註腳：是註腳：1[]：非由Trident管理

Google Cloud 後端驅動程式上的Cloud Volumes Service

Trident使用 `gcp-cvs`用於連接 Google Cloud 上的Cloud Volumes Service的驅動程式。

這 `gcp-cvs`驅動程式使用虛擬池來抽象化後端，並允許Trident確定磁碟區放置位置。管理員在以下位置定義虛擬池： `backend.json`文件。儲存類別使用選擇器按標籤識別虛擬池。

- 如果後端定義了虛擬池， Trident將嘗試在 Google Cloud 儲存池中建立磁碟區，而這些虛擬池僅限於這些儲存池。

- 如果後端未定義虛擬池， Trident將從該區域的可用儲存池中選擇 Google Cloud 儲存池。

若要在Trident上設定 Google Cloud 後端，您必須指定 `projectNumber`， `apiRegion` ， 和 `apiKey`在後端文件中。您可以在 Google Cloud 控制台中找到項目編號。 API 金鑰取自您在 Google Cloud 上為Cloud Volumes Service設定 API 存取權時所建立的服務帳戶私鑰檔案。

有關 Google Cloud 上的Cloud Volumes Service服務類型和服務等級的詳細信息，請參閱："了解Trident對 GCP 版 CVS 的支持" 。

適用於 **Google Cloud Drive** 的**Cloud Volumes Service**	快照	複製	多重附件	服務品質	擴張	複製
`gcp-cvs`	是的	是的	是的	是的	是的	僅適用於 CVS-Performance 服務類型。

複製說明

- 複製功能並非由Trident管理。
- 克隆卷將建立在與來源磁碟區相同的儲存池中。

儲存類別設計

要建立 Kubernetes 儲存類別對象，需要配置並套用各個儲存類別。本節討論如何為您的應用程式設計儲存類別。

具體後端利用

在特定的儲存類別物件中可以使用篩選來決定要與該特定儲存類別一起使用的儲存池或儲存池集。儲存類別中可以設定三組過濾器： `storagePools` ， `additionalStoragePools`和/或 `excludeStoragePools`。

這 `storagePools`此參數有助於將儲存空間限制在與任何指定屬性相符的儲存池集合中。這 `additionalStoragePools`此參數用於擴展Trident用於配置的池集，以及由屬性選擇的池集。 `storagePools`參數。您可以單獨使用其中一個參數，也可以同時使用這兩個參數，以確保選擇合適的儲存池集。

這 `excludeStoragePools`此參數用於專門排除符合屬性要求的已列出泳池集合。

模擬 **QoS** 策略

如果您希望設計儲存類別來模擬服務品質策略，請建立一個具有以下功能的儲存類別： `media`屬性為 `hdd`或者 `ssd`。基於 `media`根據儲存類別中提到的屬性， Trident將選擇合適的後端來提供服務。 `hdd`或者 `ssd`將聚合與媒體屬性相匹配，然後將磁碟區的配置定向到特定的聚合上。因此，我們可以建立一個 PREMIUM 儲存類，它將具有 `media`屬性集為 `ssd`這可以歸類為高階 QoS 策略。我們可以建立另一個儲存類別 STANDARD，其媒體屬性設定為“hdd”，這可以歸類為 STANDARD QoS 策略。我們還可以使用儲存類別中的「IOPS」屬性將配置重定向到 Element 設備，該設備可以定義為 QoS 策略。

根據特定功能使用後端

儲存類別可以設計為在特定的後端上指導磁碟區配置，其中啟用了精簡配置和厚配置、快照、複製和加密等功

能。若要指定要使用的存儲，請建立存儲類，指定啟用所需功能的相應後端。

虛擬池

所有Trident後端均可使用虛擬池。你可以使用Trident提供的任何驅動程序，為任何後端定義虛擬池。

虛擬池允許管理員在後端之上建立一個抽象層，可以透過儲存類別來引用該抽象層，從而在後端上更靈活、更有效率地放置磁碟區。同一服務類別可以定義不同的後端。此外，可以在同一個後端建立多個具有不同特性的儲存池。當使用具有特定標籤的選擇器配置儲存類別時， Trident會選擇與所有選擇器標籤相符的後端來放置磁碟區。如果儲存類別選擇器標籤與多個儲存池匹配， Trident將從中選擇其中一個來配置磁碟區。

虛擬泳池設計

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。清單包括SolidFire/ NetApp HCI、 ONTAP、GCP 上的Cloud Volumes Service以及Azure NetApp Files。

> 定義虛擬池時，建議不要嘗試在後端定義中重新排列現有虛擬池的順序。此外，建議不要編輯/修改現有虛擬池的屬性，而是定義一個新的虛擬池。

模擬不同的服務等級/**QoS**

可以設計虛擬池來模擬服務類別。使用Azure NetApp Files雲卷服務的虛擬池實現，讓我們來研究如何設定不同的服務類別。設定Azure NetApp Files後端，使用多個標籤來表示不同的效能等級。放 `servicelevel`將各個方面調整到相應的性能水平，並在每個標籤下添加其他所需的方面。現在建立不同的 Kubernetes 儲存類，將它們對應到不同的虛擬池。使用 `parameters.selector`在欄位中，每個 StorageClass 都會指定哪些虛擬池可用於託管磁碟區。

指定一組特定的方面

可以從單一儲存後端設計多個具有特定功能的虛擬池。為此，請在後端配置多個標籤，並在每個標籤下設定所需的方面。現在使用以下方式建立不同的 Kubernetes 儲存類別： `parameters.selector`此欄位將對應到不同的虛擬池。後端配置的磁碟區將具有所選虛擬池中定義的方面。

影響儲存供應的**PVC**特性

在建立 PVC 時，要求的儲存類別以外的某些參數可能會影響Trident 的設定決策過程。

訪問模式

透過 PVC 請求儲存時，必填欄位之一是存取模式。所需的模式可能會影響用於託管儲存請求的後端選擇。

Trident將嘗試根據下列矩陣將所使用的儲存協定與指定的存取方法進行比對。這與底層儲存平台無關。

	讀寫一次	只讀多	讀寫多
iSCSI	是的	是的	是的（原始區塊）
NFS	是的	是的	是的

如果向未配置 NFS 後端的Trident部署提交 ReadWriteMany PVC 請求，則不會配置任何磁碟區。因此，請求者應該使用適合其應用的存取模式。

卷操作

修改持久卷

除兩種例外情況外，持久卷是 Kubernetes 中的不可變物件。創建完成後，可以修改回收策略和大小。然而，這並不能阻止在 Kubernetes 之外修改磁碟區的某些方面。為了針對特定應用定製卷，確保容量不會意外耗盡，或者出於任何原因將卷移動到不同的存儲控制器，這樣做可能是可取的。

目前 Kubernetes 樹內配置器不支援對 NFS、iSCSI 或 FC PV 進行磁碟區大小調整操作。 Trident 支援擴充 NFS、iSCSI 和 FC 磁碟區。

PV的連線詳情已建立後無法修改。

建立按需卷快照

Trident支援按需建立磁碟區快照，並支援使用 CSI 框架從快照建立 PVC。快照提供了一種便捷的方法來維護資料在特定時間點的副本，並且在 Kubernetes 中具有獨立於來源 PV 的生命週期。這些快照可用於複製PVC。

從快照建立磁碟區

Trident也支援從磁碟區快照建立持久性磁碟區。要實現這一點，只需建立一個 PersistentVolumeClaim 並指定 `datasource`作為建立磁碟區所需的快照。 Trident將透過建立一個包含快照中資料的磁碟區來處理此 PVC。借助此功能，可以跨區域複製資料、建立測試環境、完全替換損壞或已損壞的生產卷，或檢索特定檔案和目錄並將其傳輸到另一個附加卷。

在集群中移動卷

儲存管理員能夠在ONTAP叢集中，在聚合和控制器之間移動磁碟區，而不會對儲存使用者造成任何干擾。只要目標聚合是Trident使用的 SVM 可以存取的目標聚合，此操作就不會影響Trident或 Kubernetes 叢集。重要的是，如果聚合體是新添加到 SVM 中的，則需要透過將其重新新增至Trident來刷新後端。這將觸發Trident重新清點 SVM，以便識別新的聚合體。

但是， Trident並不自動支援跨後端移動磁碟區。這包括在同一個叢集中的 SVM 之間、叢集之間，或到不同的儲存平台（即使該儲存系統是連接到Trident 的儲存系統）。

如果將磁碟區複製到另一個位置，則可以使用磁碟區匯入功能將目前磁碟區匯入到Trident。

擴大銷量

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、 SolidFire/ NetApp HCI和Cloud Volumes Service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` ， `azure-netapp-files` ， 和 `gcp-cvs`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時 `solidfire-san`司機們，請使用該命令 `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml`將現有磁碟區匯入 Kubernetes 以便由Trident管理。導入卷宗指令中使用的 PVC YAML 或 JSON 檔案指向一個儲存類，該儲存類別將Trident標識為設定程式。使用NetApp HCI/ SolidFire後端時，請確保磁碟區名稱是唯一的。如果磁碟區名稱重複，請將磁碟區複製為唯一名稱，以便磁碟區匯入功能可以區分它們。

如果 `azure-netapp-files`或者 `gcp-cvs`驅動程式已啟用，請使用該命令 `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。

執行上述命令後， Trident將在後端找到該磁碟區並讀取其大小。它會自動添加（必要時會覆蓋）已配置的PVC的容積大小。然後Trident創建新的 PV，Kubernetes 將 PVC 綁定到 PV。

如果部署的貨櫃需要特定的進口 PVC，則該貨櫃將保持待定狀態，直到透過批量進口流程綁定 PVC/PV 對為止。 PVC/PV管連接好後，如果沒有其他問題，容器應該就能升起來。

註冊服務

註冊表儲存的部署和管理已在文件中記錄。"netapp.io"在"部落格"。

日誌服務

與其他 OpenShift 服務一樣，日誌服務使用 Ansible 進行部署，設定參數由提供給 playbook 的清單檔案（又稱 hosts）提供。本文將介紹兩種安裝方法：在 OpenShift 初始安裝期間部署日誌記錄並在 OpenShift 安裝完成後部署日誌記錄。

從 Red Hat OpenShift 版本 3.9 開始，官方文件建議不要使用 NFS 作為日誌服務，因為有資料損壞的擔憂。這是基於紅帽公司對其產品的測試結果。 ONTAP NFS 伺服器不存在這些問題，並且可以輕鬆支援日誌部署。最終，日誌服務的協定選擇取決於您，但要知道，在使用NetApp平台時，這兩種協定都能很好地工作，如果您更喜歡 NFS，也沒有理由避免使用 NFS。

如果選擇將 NFS 與日誌服務一起使用，則需要設定 Ansible 變數。 `openshift_enable_unsupported_configurations`到 `true`防止安裝程式運作失敗。

開始

日誌服務可以選擇性地部署到應用程式以及 OpenShift 叢集本身的核心操作。如果您選擇部署操作日誌記錄，請指定下列變數 `openshift_logging_use_ops`作為 `true`將建立該服務的兩個實例。控制操作日誌實例的變數包含“ops”，而控制應用程式日誌實例的變數則不包含。

根據部署方法配置 Ansible 變數非常重要，以確保底層服務使用正確的儲存。讓我們來看看每種部署方法的選項。

下表僅包含與日誌服務相關的儲存配置變數。您還可以找到其他選擇"Red Hat OpenShift 日誌記錄文檔"應根據您的部署情況進行審查、配置和使用。

下表中的變數將使 Ansible playbook 使用提供的詳細資訊為日誌服務建立 PV 和 PVC。雖然這種方法不如在 OpenShift 安裝後使用元件安裝 playbook 靈活，但如果您有現有的捲可用，這也不失為一個選擇。

多變的	細節
`openshift_logging_storage_kind`	設定為 `nfs`讓安裝程式為日誌服務建立 NFS PV。
`openshift_logging_storage_host`	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導出掛載路徑。例如，如果體積連接為`/openshift_logging`你會將該路徑用於此變數。
`openshift_logging_storage_volume_name`	名稱，例如 `pv_ose_logs`，用於建立 PV。
`openshift_logging_storage_volume_size`	例如，NFS 導出的大小。 `100Gi` 。

如果您的 OpenShift 叢集已經執行，因此Trident已經部署和配置，安裝程式可以使用動態配置來建立磁碟區。需要配置以下變數。

多變的	細節
`openshift_logging_es_pvc_dynamic`	設定為 true 以使用動態配置磁碟區。
`openshift_logging_es_pvc_storage_class_name`	PVC 中將使用的儲存類別的名稱。
`openshift_logging_es_pvc_size`	PVC中所要求的體積大小。
`openshift_logging_es_pvc_prefix`	日誌記錄服務使用的 PVC 前綴。
`openshift_logging_es_ops_pvc_dynamic`	設定為 `true`為運維日誌實例使用動態配置的磁碟區。
`openshift_logging_es_ops_pvc_storage_class_name`	運維日誌實例的儲存類別名稱。
`openshift_logging_es_ops_pvc_size`	操作實例的磁碟區請求大小。
`openshift_logging_es_ops_pvc_prefix`	操作實例 PVC 的前綴。

部署日誌堆疊

如果您將日誌記錄部署作為 OpenShift 初始安裝過程的一部分，那麼您只需要遵循標準部署程序。 Ansible 將配置和部署所需的服務和 OpenShift 對象，以便在 Ansible 完成後立即提供服務。

但是，如果在初始安裝之後進行部署，則需要使用 Ansible 的元件 playbook。此過程可能因 OpenShift 版本不同而略有差異，因此請務必閱讀並遵循相關說明。"Red Hat OpenShift 容器平台 3.11 文檔"適用於您的版本。

指標服務

指標服務為管理員提供有關 OpenShift 叢集的狀態、資源利用率和可用性的寶貴資訊。此外，它對於 pod 自動擴展功能也是必要的，許多組織使用指標服務中的資料進行費用分攤和/或收益展示應用程式。

與日誌服務以及整個 OpenShift 一樣，Ansible 也用於部署指標服務。此外，與日誌服務一樣，指標服務可以在叢集的初始設定期間或叢集運行後使用元件安裝方法進行部署。以下表格包含了為指標服務配置持久性儲存時重要的變數。

下表僅包含與指標服務相關的儲存配置變數。文件中還有許多其他選項，應根據您的部署情況進行檢視、配置和使用。

多變的	細節
`openshift_metrics_storage_kind`	設定為 `nfs`讓安裝程式為日誌服務建立 NFS PV。
`openshift_metrics_storage_host`	NFS主機的主機名稱或IP位址。這應該設定為你的SVM 的 dataLIF。
`openshift_metrics_storage_nfs_directory`	NFS導出掛載路徑。例如，如果體積連接為`/openshift_metrics`你會將該路徑用於此變數。
`openshift_metrics_storage_volume_name`	名稱，例如 `pv_ose_metrics`，用於建立 PV。
`openshift_metrics_storage_volume_size`	例如，NFS 導出的大小。 `100Gi` 。

如果您的 OpenShift 叢集已經執行，因此Trident已經部署和配置，安裝程式可以使用動態配置來建立磁碟區。需要配置以下變數。

多變的	細節
`openshift_metrics_cassandra_pvc_prefix`	用於指標 PVC 的前綴。
`openshift_metrics_cassandra_pvc_size`	請求的捲冊數量。
`openshift_metrics_cassandra_storage_type`	用於指標的儲存類型，必須設定為動態，以便 Ansible 建立具有適當儲存類別的 PVC。
`openshift_metrics_cassanda_pvc_storage_class_name`	要使用的儲存類別的名稱。

部署指標服務

在 hosts/inventory 檔案中定義了對應的 Ansible 變數後，使用 Ansible 部署服務。如果您在安裝 OpenShift 時進行部署，則 PV 將自動建立和使用。如果您使用元件 playbook 進行部署，則在 OpenShift 安裝之後，Ansible 會建立所需的任何 PVC，並在Trident為其配置儲存之後部署服務。

上述變數以及部署過程可能會隨著 OpenShift 的每個版本而改變。請務必查看並遵循"紅帽 OpenShift 部署指南"請根據您的環境配置您的版本。

資料保護和災難復原

了解Trident的保護和復原選項以及使用Trident建立的磁碟區。對於每個有持久化需求的應用程序，都應該制定資料保護和復原策略。

Trident複製與恢復

您可以建立備份，以便在災難發生時還原Trident 。

Trident複製

Trident使用 Kubernetes CRD 來儲存和管理自己的狀態，並使用 Kubernetes 叢集 etcd 來儲存其元資料。

步驟

1. 使用以下指令備份 Kubernetes 叢集 etcd"Kubernetes：備份 etcd 集群" 。

2. 將備份檔案放置在FlexVol volume上

> NetApp建議您使用SnapMirror關係式將FlexVol所在的 SVM 與其他 SVM 連接起來，從而保護該 SVM。

Trident恢復

使用 Kubernetes CRD 和 Kubernetes 叢集 etcd 快照，您可以還原Trident。

步驟

1. 從目標 SVM 上，將包含 Kubernetes etcd 資料檔案和憑證的磁碟區掛載到將要設定為主節點的主機上。
2. 複製 Kubernetes 叢集所需的所有憑證。 `/etc/kubernetes/pki`以及 `etcd` 成員文件 `/var/lib/etcd`。
3. 使用 etcd 備份還原 Kubernetes 集群"Kubernetes：恢復 etcd 集群"。
4. 跑步 `kubectl get crd`驗證所有Trident自訂資源是否已啟動，並擷取Trident物件以驗證所有資料是否可用。

SVM複製與復原

Trident無法設定複製關係，但儲存管理員可以使用 "ONTAP SnapMirror"複製 SVM。

發生災難時，您可以啟動SnapMirror目標 SVM 開始提供資料服務。系統恢復後，您可以切換回主伺服器。

關於此任務

使用SnapMirror SVM 複製功能時，請考慮以下事項：

- 對於啟用了 SVM-DR 的每個 SVM，您應該建立一個獨立的後端。
- 配置儲存類，僅在需要時選擇複製後端，以避免將不需要複製的磁碟區配置到支援 SVM-DR 的後端上。
- 應用程式管理員應了解複製帶來的額外成本和複雜性，並在開始此過程之前仔細考慮其恢復計劃。

SVM複製

您可以使用"ONTAP： SnapMirror SVM 複製"建立 SVM 複製關係。

SnapMirror可讓您設定選項來控制要複製的內容。您需要知道您在執行操作時選擇了哪些選項。使用Trident進行 SVM 恢復 。

- "-identity-preserve true"複製整個 SVM 配置。
- "-丟棄網路配置"不包括 LIF 和相關網路設定。
- "-identity-preserve false"僅複製捲和安全性配置。

使用Trident進行 SVM 恢復

Trident無法自動偵測 SVM 故障。如果發生災難，管理員可以手動啟動Trident故障轉移到新的 SVM。

步驟

1. 取消已排程和正在進行的SnapMirror傳輸，斷開複製關係，停止來源 SVM，然後啟動SnapMirror目標

SVM。

2. 如果您指定 `-identity-preserve false`或者 `-discard-config network`配置 SVM 複製時，請更新以下內容：`managementLIF`和 `dataLIF`在Trident後端定義檔中。
3. 確認 `storagePrefix`存在於Trident後端定義檔中。此參數無法變更。省略 `storagePrefix`這將導致後端更新失敗。
4. 使用下列命令更新所有必要的後端，以反映新的目標 SVM 名稱：

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. 如果您指定 `-identity-preserve false`或者 `discard-config network`您必須重新啟動所有應用程式 pod。

> 如果您指定 `-identity-preserve true`當目標 SVM 啟動時， Trident提供的所有磁碟區開始提供資料服務。

磁碟區複製和恢復

Trident無法設定SnapMirror複製關係，但儲存管理員可以使用"ONTAP SnapMirror複製與恢復"複製Trident建立的磁碟區。

然後，您可以使用下列方法將復原的磁碟區匯入到Trident ："tridentctl 磁碟區導入" 。

> 不支援導入 `ontap-nas-economy`， `ontap-san-economy` ， 或者 `ontap-flexgroup-economy`司機。

快照資料保護

您可以使用以下方法保護和復原資料：

- 外部快照控制器和 CRD 用於建立持久磁碟區 (PV) 的 Kubernetes 磁碟區快照。

  "卷快照"

- ONTAP快照用於還原磁碟區的全部內容，或還原單一檔案或 LUN。

  "ONTAP快照"

安全

安全

請按照此處列出的建議，確保您的Trident安裝安全可靠。

在Trident自身的命名空間中運行它

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 非常重要，以確保可靠的儲存並封鎖潛在的惡意活動。

為了將其他應用程式和使用者與Trident隔離，請務必將Trident安裝在其自身的 Kubernetes 命名空間中。(`trident`）。將Trident放在自己的命名空間中，可以確保只有 Kubernetes 管理員才能存取Trident pod 和儲存在命名空間 CRD 物件中的工件（例如後端和 CHAP 金鑰，如果適用）。您應該確保只允許管理員存取Trident命名空間，從而獲得存取權限。 `tridentctl`應用。

使用 CHAP 驗證與ONTAP SAN 後端配合使用

Trident支援基於 CHAP 的ONTAP SAN 工作負載驗證（使用 `ontap-san`和 `ontap-san-economy`司機）。NetApp建議使用Trident的雙向 CHAP 進行主機與儲存後端之間的驗證。

對於使用 SAN 儲存驅動程式的ONTAP後端， Trident可以透過以下方式設定雙向 CHAP 並管理 CHAP 使用者名稱和金鑰： `tridentctl` 。請參閱"準備配置後端ONTAP SAN 驅動程式"了解Trident如何在ONTAP後端設定CHAP。

使用 CHAP 驗證連線NetApp HCI和SolidFire後端

NetApp建議部署雙向 CHAP，以確保主機與NetApp HCI和SolidFire後端之間的驗證。 Trident使用一個包含每個租用戶兩個 CHAP 密碼的秘密物件。安裝Trident後，它會管理 CHAP 金鑰並將其儲存在一個位置。`tridentvolume`對應 PV 的 CR 物件。在建立 PV 時， Trident使用 CHAP 金鑰啟動 iSCSI 會話，並透過 CHAP 與NetApp HCI和SolidFire系統通訊。

Trident建立的磁碟區不會與任何磁碟區存取群組關聯。

將Trident與 NVE 和 NAE 結合使用

NetApp ONTAP提供靜態資料加密，以保護敏感數據，防止磁碟被盜、退回或重新利用。有關詳細信息，請參閱"配置NetApp卷加密概述"。

- 如果後端啟用了 NAE，則在Trident中配置的任何磁碟區都會啟用 NAE。
  - 。您可以將 NVE 加密標誌設定為 `""`建立支援 NAE 的磁碟區。
- 如果後端未啟用 NAE，則在Trident中配置的任何磁碟區都會啟用 NVE，除非 NVE 加密標誌設為 。 `false`（後端配置中的預設值）。

在啟用 NAE 的後端上使用Trident建立的磁碟區必須使用 NVE 或 NAE 加密。

- 您可以將 NVE 加密標誌設定為 `true`在Trident後端設定中，可以覆蓋 NAE 加密，並按磁碟區使用特定的加密金鑰。
- 將 NVE 加密標誌設為 `false`在啟用 NAE 的後端上建立啟用 NAE 的磁碟區。您無法透過將 NVE 加密標誌設為 來停用 NAE 加密。 `false` 。

- 您可以透過明確設定 NVE 加密標誌，在Trident中手動建立 NVE 磁碟區。 `true` 。

有關後端配置選項的更多信息，請參閱：

- "ONTAP SAN 配置選項"

- "ONTAP NAS 設定選項"

Linux 統一金鑰設定 (LUKS)

您可以啟用 Linux 統一密鑰設定 (LUKS) 來加密Trident上的ONTAP SAN 和ONTAP SAN ECONOMY 磁碟區。 Trident支援 LUKS 加密磁碟區的密碼短語輪換和磁碟區擴展。

在Trident中，LUKS 加密磁碟區使用 aes-xts-plain64 密碼和模式，這是建議的。"美國國家標準與技術研究院"。

ASA r2 系統不支援 LUKS 加密。有關ASA r2 系統的信息，請參閱"了解ASA r2 儲存系統"。

開始之前

- 工作節點必須安裝 cryptsetup 2.1 或更高版本（但低於 3.0）。欲了解更多信息，請訪問"GitLab：加密設定"。
- 出於效能方面的考慮， NetApp建議工作節點支援進階加密標準新指令 (AES-NI)。若要驗證是否支援 AES-NI，請執行下列命令：

```
grep "aes" /proc/cpuinfo
```

如果沒有回傳任何內容，則表示您的處理器不支援 AES-NI。有關 AES-NI 的更多信息，請訪問："英特爾：高階加密標準指令集（AES-NI）"。

啟用 LUKS 加密

您可以使用 Linux 統一密鑰設定 (LUKS) 為ONTAP SAN 和ONTAP SAN ECONOMY 磁碟區啟用按磁碟區主機端加密。

步驟

1. 在後端配置中定義 LUKS 加密屬性。有關ONTAP SAN 後端配置選項的更多信息，請參閱"ONTAP SAN 配置選項"。

```
{
  "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. 建立一個包含 LUKS 密碼短語的金鑰。例如：

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

限制

LUKS 加密磁碟區無法利用ONTAP重複資料刪除和壓縮功能。

導入 **LUKS** 磁碟區的後端配置

若要匯入 LUKS 卷，您必須設定 `luksEncryption`到(`true`在後端。這 `luksEncryption`此選項用於告知Trident該磁碟區是否符合 LUKS 標準。(`true`或不符合 LUKS 標準(`false`）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'
```

用於導入 **LUKS** 磁碟區的 **PVC** 配置

若要動態匯入 LUKS 卷，請設定註釋 `trident.netapp.io/luksEncryption`到 `true`並在 PVC 中包含一個支援 LUKS 的儲存類，如本例所示。

```
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
```

輪換 **LUKS** 密碼短語

您可以輪換 LUKS 密碼短語並確認輪換。

> 在您確認任何磁碟區、快照或金鑰不再引用該密碼短語之前，請勿忘記該密碼短語。如果引用的密碼短語遺失，您可能無法掛載卷，資料將保持加密狀態且無法存取。

關於此任務

當指定新的 LUKS 密碼短語後建立掛載磁碟區的 pod 時，就會發生 LUKS 密碼短語輪替。當建立新的 pod 時，Trident會將磁碟區上的 LUKS 密碼短語與金鑰中的活動密碼短語進行比較。

- 如果磁碟區上的密碼短語與金鑰中的活動密碼短語不匹配，則會發生輪換。
- 如果磁碟區上的密碼短語與金鑰中的活動密碼短語匹配，則 `previous-luks-passphrase`參數被忽略。

步驟

1. 添加 `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已存在的密碼短語。

   體積

   ```
   tridentctl -d get volume luks-pvc1
   GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

   ...luksPassphraseNames:["A"]
   ```

   快照

   ```
   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 金鑰，以指定新的和先前的密碼短語。確保 `previous-luke-passphrase-name`和 `previous-luks-passphrase`與先前的密碼短語相符。

   ```
   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. 建立一個新的 pod，掛載該磁碟區。這是啟動旋轉所必需的步驟。
5. 確認密碼短語已輪換。

   體積

   ```
   tridentctl -d get volume luks-pvc1
   GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

   ...luksPassphraseNames:["B"]
   ```

快照

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

結果

當磁碟區和快照上僅傳回新密碼短語時，密碼短語就會輪替。

例如，如果傳回兩個密碼短語。 `luksPassphraseNames: ["B", "A"]`旋轉不完整。您可以觸發一個新的機艙來嘗試完成輪換。

啟用磁碟區擴充

您可以對 LUKS 加密磁碟區啟用磁碟區擴充。

步驟

1. 啟用 `CSINodeExpandSecret`功能門（beta 1.25+）。參考 "Kubernetes 1.25：使用金鑰實作 CSI 磁碟區的節點驅動擴展" 了解詳情。
2. 添加 `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

啟動線上儲存擴充功能時，kubelet 會將對應的憑證傳遞給驅動程式。

Kerberos 飛行中加密

使用 Kerberos 傳輸中加密，您可以對託管叢集和儲存後端之間的流量啟用加密，從而提高資料存取安全性。

Trident支援以ONTAP作為儲存後端的 Kerberos 加密：

- 本地**ONTAP** - Trident支援透過 NFSv3 和 NFSv4 連線對來自 Red Hat OpenShift 和上游 Kubernetes 叢集的本機ONTAP磁碟區進行 Kerberos 加密。

您可以建立、刪除、調整大小、建立快照、複製、只讀複製和匯入使用 NFS 加密的磁碟區。

配置本地**ONTAP**磁碟區的飛行中 **Kerberos** 加密

您可以為託管叢集和本機ONTAP儲存後端之間的儲存流量啟用 Kerberos 加密。

僅使用以下方式支援對本地ONTAP儲存後端的 NFS 流量進行 Kerberos 加密： `ontap-nas`儲存驅動程式。

開始之前

- 確保您可以訪問 `tridentctl`公用事業。
- 請確保您擁有ONTAP儲存後端的管理員權限。
- 請確保您知道要從ONTAP儲存後端共用的磁碟區的名稱。
- 請確保您已準備好ONTAP儲存 VM，以支援 NFS 磁碟區的 Kerberos 加密。請參閱 "在資料 LIF 上啟用 Kerberos"以取得說明。
- 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 網域設定部分（第 13 頁）。 "NetApp NFSv4 增強功能與最佳實務指南" 。

新增或修改**ONTAP**匯出策略

您需要為現有的ONTAP匯出策略新增規則，或建立新的匯出策略，以支援對ONTAP儲存 VM 根磁碟區以及與上游 Kubernetes 叢集共用的任何ONTAP磁碟區進行 Kerberos 加密。您新增的匯出策略規則或您建立的新匯出策略需要支援以下存取協定和存取權限：

訪問協定

配置導出策略，支援 NFS、NFSv3 和 NFSv4 存取協定。

訪問詳情

您可以根據磁碟區的需求配置三種不同版本的 Kerberos 加密之一：

- **Kerberos 5** - （驗證和加密）
- **Kerberos 5i** - （身份驗證和加密，具有身份保護功能）
- **Kerberos 5p** - （驗證和加密，提供身分和隱私保護）

配置ONTAP導出策略規則，使其具有適當的存取權限。例如，如果叢集將使用 Kerberos 5i 和 Kerberos 5p 混合加密方式掛載 NFS 卷，請使用下列存取設定：

類型	只讀存取權限	讀/寫權限	超級使用者存取權限
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

有關如何建立ONTAP匯出策略和匯出策略規則，請參閱以下文件：

- "建立導出策略"
- "在匯出策略中新增規則"

建立儲存後端

您可以建立包含 Kerberos 加密功能的Trident儲存後端設定。

關於此任務

建立配置 Kerberos 加密的儲存後端設定檔時，可以使用下列方式指定三種不同版本的 Kerberos 加密之一：`spec.nfsMountOptions`範圍：

- `spec.nfsMountOptions: sec=krb5`（身份驗證和加密）
- `spec.nfsMountOptions: sec=krb5i`（身份驗證和加密，以及身份保護）
- `spec.nfsMountOptions: sec=krb5p`（身份驗證和加密，以及身分和隱私保護）

只能指定一個 Kerberos 等級。如果在參數清單中指定多個 Kerberos 加密級別，則僅使用第一個選項。

步驟

1. 在託管叢集上，使用以下範例建立儲存後端設定檔。將方括號 <> 中的值替換為您環境中的資訊：

```
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. 使用上一步建立的設定檔建立後端：

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

如果後端建立失敗，則後端配置存在問題。您可以透過執行以下命令查看日誌以確定原因：

```
tridentctl logs
```

在您發現並修正設定檔中的問題後，您可以再次執行建立命令。

建立儲存類別

您可以建立儲存類別來配置具有 Kerberos 加密的磁碟區。

關於此任務

建立儲存類別物件時，可以使用下列方式指定三種不同版本的 Kerberos 加密之一：`mountOptions`範圍：

- `mountOptions: sec=krb5`（身份驗證和加密）
- `mountOptions: sec=krb5i`（身份驗證和加密，以及身份保護）
- `mountOptions: sec=krb5p`（身份驗證和加密，以及身分和隱私保護）

只能指定一個 Kerberos 等級。如果在參數清單中指定多個 Kerberos 加密級別，則僅使用第一個選項。如果在儲存後端設定中指定的加密等級與在儲存類別物件中指定的加密等級不同，則以儲存類別物件為準。

步驟

1. 建立一個 StorageClass Kubernetes 對象，請參考下列範例：

```
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. 請確保已建立儲存類別：

```
kubectl get sc ontap-nas-sc
```

您應該會看到類似以下內容的輸出：

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

供應量

建立儲存後端和儲存類別之後，現在可以設定捲了。有關說明，請參閱 "提供一定量"。

使用**Azure NetApp Files**磁碟區設定飛行中 **Kerberos** 加密

您可以為託管叢集與單一Azure NetApp Files儲存後端或Azure NetApp Files儲存後端虛擬池之間的儲存流量啟用 Kerberos 加密。

開始之前

- 請確保已在受管 Red Hat OpenShift 叢集上啟用Trident 。
- 確保您可以訪問 `tridentctl`公用事業。
- 請確保您已按照以下說明準備好Azure NetApp Files儲存後端以進行 Kerberos 加密：注意相關要求並按照說明進行操作。 "Azure NetApp Files文檔" 。
- 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 網域設定部分（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儲存後端設定。

關於此任務

建立配置 Kerberos 加密的儲存後端設定檔時，您可以將其定義為套用於下列兩個層級之一：

- 使用*儲存後端等級* `spec.kerberos`場地
- 使用*虛擬池層級* `spec.storage.kerberos`場地

在虛擬池層級定義配置時，使用儲存類別中的標籤選擇池。

無論在哪個級別，您都可以指定三種不同版本的 Kerberos 加密之一：

- `kerberos: sec=krb5`（身份驗證和加密）
- `kerberos: sec=krb5i`（身份驗證和加密，以及身份保護）
- `kerberos: sec=krb5p`（身份驗證和加密，以及身分和隱私保護）

步驟

1. 在託管叢集上，根據您需要定義儲存後端的位置（儲存後端等級或虛擬池層級），使用下列範例之一建立儲存後端設定檔。將方括號 <> 中的值替換為您環境中的資訊：

儲存後端範例

```
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
```

虛擬池層級範例

```
---
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. 使用上一步建立的設定檔建立後端：

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

如果後端建立失敗，則後端配置存在問題。您可以透過執行以下命令查看日誌以確定原因：

```
tridentctl logs
```

在您發現並修正設定檔中的問題後，您可以再次執行建立命令。

建立儲存類別

您可以建立儲存類別來配置具有 Kerberos 加密的磁碟區。

步驟

1. 建立一個 StorageClass Kubernetes 對象，請參考下列範例：

```
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. 請確保已建立儲存類別：

```
kubectl get sc -sc-nfs
```

您應該會看到類似以下內容的輸出：

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

供應量

建立儲存後端和儲存類別之後，現在可以設定捲了。有關說明，請參閱 "提供一定量"。

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。

下一步是什麼？

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：

- 亞馬遜彈性 Kubernetes 服務 (EKS)
- Google Kubernetes 引擎 (GKE)
- Microsoft Azure Kubernetes 服務 (AKS)
- 紅帽 OpenShift
- SUSE Rancher
- VMware Tanzu 產品組合
- 上游 Kubernetes

- 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 "這些說明"。

Trident Protect 儲存後端相容性

Trident Protect 支援以下儲存後端：

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- ONTAP儲存陣列
- Google Cloud NetApp Volumes
- Azure NetApp Files

請確保您的儲存後端符合以下要求：

- 確保連接到叢集的NetApp儲存裝置使用的是Trident 24.02 或更高版本（建議使用Trident 24.10）。
- 請確保您擁有NetApp ONTAP儲存後端。
- 請確保您已配置用於儲存備份的物件儲存桶。
- 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-economy 容量要求

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。

某些應用程式與使用快照目錄的磁碟區不相容。對於這些應用，您需要透過在ONTAP儲存系統上執行以下命令來隱藏快照目錄：

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

您可以透過對每個 nas-economy 磁碟區執行以下命令來啟用快照目錄，並將命令替換為 `<volume-UUID>`使用要變更的磁碟區的 UUID：

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

您可以透過設定Trident後端設定選項，為新磁碟區預設啟用快照目錄。 `snapshotDir`到 `true`。現有捲數不受影響。

使用 **KubeVirt** 虛擬機器保護數據

Trident Protect 24.10 和 24.10.1 及更高版本在保護運行在 KubeVirt VM 上的應用程式時，行為有所不同。對於這兩個版本，您都可以在資料保護操作期間啟用或停用檔案系統凍結和解凍。

在恢復操作期間，任何 `VirtualMachineSnapshots`為虛擬機器 (VM) 建立的設定檔無法復原。

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 檔案系統的凍結與解凍。"配置虛擬化"然後使用以下命令：

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Trident Protect 24.10.1 及更高版本

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
```

SnapMirror複製的要求

NetApp SnapMirror複製功能可與Trident Protect 搭配使用，適用於下列ONTAP解決方案：

- 本地部署的NetApp FAS、 AFF和ASA集群
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

ONTAP叢集對SnapMirror複製的要求

如果您打算使用SnapMirror複製功能，請確保您的ONTAP叢集符合以下要求：

- * 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`：極品飛車
  - `ontap-san`：iSCSI
  - `ontap-san`：FC
  - `ontap-san`：NVMe/TCP（最低要求ONTAP版本 9.15.1）
- 許可證：使用資料保護包的ONTAP SnapMirror非同步許可證必須在來源 ONTAP 叢集和目標ONTAP叢集上啟用。請參閱 "ONTAP中的SnapMirror許可概述"了解更多。

  從ONTAP 9.10.1 開始，所有許可證均以NetApp許可證文件 (NLF) 的形式交付，這是一個可以啟用多種功能的單一文件。請參閱"ONTAP One 隨附的許可證"了解更多。

  僅支援SnapMirror非同步保護。

SnapMirror複製的對等連線注意事項

如果您打算使用儲存後端對等互連，請確保您的環境符合以下要求：

- 叢集和 **SVM**： ONTAP儲存後端必須相互連接。請參閱 "叢集和SVM對等連接概述"了解更多。

  確保兩個ONTAP叢集之間複製關係中使用的 SVM 名稱是唯一的。

- * NetApp Trident和 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複製的Trident / ONTAP配置

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

請確保您的 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 讀取資料和寫入數據，以實現應用程式物件複製。
- 網路連線：設定防火牆規則、儲存桶權限和 IP 允許列表，以啟用兩個叢集和 AppVault 之間透過 WAN 進行通訊。

許多企業環境在廣域網路連線中實施嚴格的防火牆策略。在配置複製之前，請先與您的基礎架構團隊確認這些網路需求。

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 倉庫：

```
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 100.2506.0 --create
-namespace --namespace trident-protect
```

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. 將以下鏡像拉取到本地計算機，更新標籤，然後將其推送到您的私有鏡像倉庫：

```
netapp/controller:25.06.0
netapp/restic:25.06.0
netapp/kopia:25.06.0
netapp/trident-autosupport:25.06.0
netapp/exechook:25.06.0
netapp/resourcebackup:25.06.0
netapp/resourcerestore:25.06.0
netapp/resourcedelete:25.06.0
bitnami/kubectl:1.30.2
kubebuilder/kube-rbac-proxy:v0.16.0
```

例如：

```
docker pull netapp/controller:25.06.0
```

```
docker tag netapp/controller:25.06.0 <private-registry-
url>/controller:25.06.0
```

```
docker push <private-registry-url>/controller:25.06.0
```

2. 建立Trident Protect 系統命名空間：

```
kubectl create ns trident-protect
```

3. 登入註冊表：

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. 建立用於私有註冊表驗證的拉取金鑰：

```
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 倉庫：

```
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：

```
---
image:
  registry: <private-registry-url>
imagePullSecrets:
  - name: regcred
controller:
  image:
    registry: <private-registry-url>
rbacProxy:
  image:
    registry: <private-registry-url>
crCleanup:
  imagePullSecrets:
    - name: regcred
webhooksCleanup:
  imagePullSecrets:
    - name: regcred
```

7. 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 100.2506.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

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

在使用命令列實用程式之前，需要將其安裝在用於存取叢集的電腦上。根據您的機器使用的是 x64 還是ARM CPU，請依照以下步驟操作。

下載適用於 **Linux AMD64 CPU** 的插件

步驟

1. 下載Trident Protect CLI 外掛：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.06.0/tridentctl-protect-linux-amd64
```

下載適用於 **Linux ARM64 CPU** 的插件

步驟

1. 下載Trident Protect CLI 外掛：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.06.0/tridentctl-protect-linux-arm64
```

下載適用於 **Mac AMD64 CPU** 的插件

步驟

1. 下載Trident Protect CLI 外掛：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.06.0/tridentctl-protect-macos-amd64
```

下載適用於 **Mac ARM64 CPU** 的插件

步驟

1. 下載Trident Protect CLI 外掛：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.06.0/tridentctl-protect-macos-arm64
```

1. 啟用插件二進位檔案的執行權限：

```
chmod +x tridentctl-protect
```

2. 將插件二進位檔案複製到 PATH 環境變數中定義的位置。例如， `/usr/bin`或者 `/usr/local/bin`（您可能需要更高的權限）：

```
cp ./tridentctl-protect /usr/local/bin/
```

3. 您也可以選擇將插件二進位檔案複製到您主目錄中的某個位置。在這種情況下，建議確保該位置已新增至您的 PATH 環境變數：

```
cp ./tridentctl-protect ~/bin/
```

將插件複製到 PATH 環境變數中的某個位置，即可透過鍵入以下命令來使用該插件。 `tridentctl-protect`或者 `tridentctl protect`從任何地點。

查看**Trident CLI** 外掛程式幫助

您可以使用插件內建的幫助功能來獲得有關插件功能的詳細幫助：

步驟

1. 使用幫助功能查看使用指南：

```
tridentctl-protect help
```

啟用指令自動補全

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。

為 **Bash shell** 啟用自動補全功能

步驟

1. 下載完成腳本：

```
curl -L -O https://github.com/NetApp/tridentctl-
protect/releases/download/25.06.0/tridentctl-completion.bash
```

2. 在您的使用者目錄下建立一個新目錄，用於存放腳本：

```
mkdir -p ~/.bash/completions
```

3. 將下載的腳本移到 `~/.bash/completions`目錄：

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. 將以下行新增至 `~/.bashrc`您主目錄中的檔案：

```
source ~/.bash/completions/tridentctl-completion.bash
```

為 **Z shell** 啟用自動補全

步驟

1. 下載完成腳本：

```
curl -L -O https://github.com/NetApp/tridentctl-
protect/releases/download/25.06.0/tridentctl-completion.zsh
```

2. 在您的使用者目錄下建立一個新目錄，用於存放腳本：

```
mkdir -p ~/.zsh/completions
```

3. 將下載的腳本移到 `~/.zsh/completions`目錄：

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. 將以下行新增至 `~/.zprofile`您主目錄中的檔案：

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

結果

下次登入 shell 時，您可以使用 tridentctl-protect 外掛程式進行指令自動補全。

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. 建立一個名為的文件 `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。

   以下範例設定檔顯示了可用設置，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. 應用以下值 `resourceLimits.yaml`文件：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

自訂安全上下文約束

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. 建立一個名為的文件 `sccconfig.yaml`。
2. 在文件中新增 SCC 選項，並根據您的環境需求修改參數。

   以下範例顯示了 SCC 選項的參數預設值：

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

下表描述了SCC選項的參數：

範圍	描述	預設
創造	確定是否可以建立 SCC 資源。僅當 `scc.create`設定為 `true`Helm 安裝過程會辨識 OpenShift 環境。如果不是在 OpenShift 上運行，或者如果 `scc.create`設定為 `false` 不會創建 SCC 資源。	真的
姓名	指定SCC的名稱。	三叉戟保護工作
優先事項	確定 SCC 的優先順序。優先順序較高的SCC會優先於優先順序較低的SCC進行評估。	1

3. 應用以下值 `sccconfig.yaml`文件：

```
helm upgrade trident-protect netapp-trident-protect/trident-protect -f
sccconfig.yaml --reuse-values
```

這將用指定的值替換預設值。 `sccconfig.yaml`文件。

設定其他**Trident Protect** 舵圖設置

您可以自訂AutoSupport設定和命名空間過濾以滿足您的特定要求。下表描述了可用的配置參數：

範圍	類型	描述
自動支援代理	細繩	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“跳過AutoSupport代理連接的 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` ）

範圍	類型	描述
恢復跳過命名空間註釋	細繩	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	細繩	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。

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. 建立一個設定檔並將其命名為 `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"
```

3. 填寫完後 `values.yaml`使用正確的值建立配置文件，並套用該設定檔：

```
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`用於指定個別參數的標誌：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set restoreSkipNamespaceAnnotations="annotation1,annotation2" \
  --set restoreSkipNamespaceLabels="label1,label2" \
  --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. 建立一個名為的文件 `nodeSelectorConfig.yaml`。
2. 在檔案中新增 nodeSelector 選項，並修改檔案以新增或變更節點標籤，從而根據您的環境需求進行限制。例如，以下檔案包含預設的作業系統限制，但同時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. 應用以下值 `nodeSelectorConfig.yaml`文件：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

這將預設限制替換為您在設定中指定的限制。 `nodeSelectorConfig.yaml`文件。

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。

叢集管理員始終擁有對預設資源的存取權限。 `trident-protect`命名空間，並且可以存取所有其他命名空間中的資源。要控制對資源和應用程式的訪問，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，預設情況下，任何使用者都無法建立應用程式資料管理變更請求 (CR)。 `trident-protect`命名空間。您需要在應用程式命名空間中建立應用程式資料管理 CR（最佳實踐是在與其關聯的應用程式相同的命名空間中建立應用程式資料管理 CR）。

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**：需要儲存桶憑證數據
- **AutoSupportBundle**：收集指標、日誌和其他敏感的Trident Protect數據
- **AutoSupportBundleSchedule**：管理日誌收集計劃

最佳實踐是使用基於角色的存取控制 (RBAC) 將對特權物件的存取限制在管理員範圍內。

有關基於角色的存取控制 (RBAC) 如何管理對資源和命名空間的存取的更多信息，請參閱... "Kubernetes RBAC 文檔" 。

有關服務帳戶的信息，請參閱 "Kubernetes 服務帳戶文檔"。

範例：管理兩組使用者的存取權限

例如，一個組織有集群管理員、一組工程用戶和一組行銷用戶。叢集管理員將完成以下任務，以建立一個環境，其中工程組和行銷組各自只能存取分配給其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

創建命名空間可以讓你從邏輯上分離資源，並更好地控制誰可以存取這些資源。

步驟

1. 為工程組建立一個命名空間：

```
kubectl create ns engineering-ns
```

2. 為行銷組創造命名空間：

```
kubectl create ns marketing-ns
```

步驟 **2**：建立新的服務帳戶，以便與每個命名空間中的資源互動

您建立的每個新命名空間都附帶一個預設服務帳戶，但您應該為每個使用者群組建立一個服務帳戶，以便將來必要時可以進一步在群組之間劃分權限。

步驟

1. 為工程團隊建立一個服務帳戶：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. 為行銷團隊建立一個服務帳戶：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

步驟 **3**：為每個新服務帳戶建立一個金鑰

服務帳戶金鑰用於對服務帳戶進行身份驗證，如果遭到洩露，可以輕鬆刪除並重新建立。

步驟

1. 為工程服務帳戶建立一個金鑰：

```
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一個金鑰：

```
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** 對象，將 **ClusterRole** 物件綁定到每個新的服務帳戶。

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. 將叢集角色綁定到工程服務帳戶：

```
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. 將群集角色綁定到行銷服務帳戶：

```
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. 確認工程用戶無法存取行銷資源：

```
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 物件的存取權限

若要執行備份和快照等資料管理任務，叢集管理員需要授予個別使用者對 AppVault 物件的存取權限。

步驟

1. 建立並套用 AppVault 和金鑰組合的 YAML 文件，以授予使用者對 AppVault 的存取權限。例如，以下 CR 授予使用者對 AppVault 的存取權限 `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，使叢集管理員能夠授予對命名空間中特定資源的存取權限。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. 建立並套用角色綁定 CR，將權限綁定到使用者 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. 請確認權限是否正確。

   a. 嘗試檢索所有命名空間的 AppVault 物件資訊：

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   您應該會看到類似以下內容的輸出：

```
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. 測試用戶是否可以獲得他們現在有權訪問的 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
```

您應該會看到類似以下內容的輸出：

```
yes
```

結果

您授予 AppVault 權限的使用者應該能夠使用授權的 AppVault 物件進行應用程式資料管理操作，並且不應該能夠存取指派的命名空間之外的任何資源，或建立他們無權存取的新資源。

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，它可以接收 Prometheus 等工具發送的警報，並將它們路由到您配置的目標位置。

> 這些步驟中包含的配置和指導僅供參考；您需要根據自己的環境進行自訂。請參閱以下官方文件以取得具體說明和支援：
>
> - "kube-state-metrics 文檔"
> - "普羅米修斯文檔"
> - "Alertmanager 文檔"

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-state-metrics**

您可以使用 Helm 安裝 kube-state-metrics。

步驟

1. 新增 kube-state-metrics Helm chart。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 chart 建立一個設定檔（例如， `metrics-config.yaml` ）。您可以根據自身環境自訂以下範例配置：

metrics-config.yaml：kube-state-metrics Helm chart 配置

```
---
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. 透過部署 Helm chart 來安裝 kube-state-metrics。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-metrics 自訂資源文檔" 。

安裝 Prometheus

您可以按照以下說明安裝 Prometheus： "普羅米修斯文檔" 。

安裝 Alertmanager

您可以按照以下說明安裝 Alertmanager： "Alertmanager 文檔" 。

步驟 2：配置監控工具以協同工作

安裝監控工具後，需要設定它們以使其協同工作。

步驟

1. 將 kube-state-metrics 與 Prometheus 整合。編輯 Prometheus 配置文件(`prometheus.yaml`）並新增 kube-state-metrics 服務資訊。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. 配置 Prometheus 將警報路由到 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

Prometheus 現在可以從 kube-state-metrics 收集指標，並且可以向 Alertmanager 發送警報。現在您可以設定觸發警報的條件以及警報的發送位置。

步驟 **3**：設定警報和警報目標

配置好工具協同工作後，還需要配置哪些類型的信息會觸發警報，以及警報應該發送到哪裡。

警報範例：備份失敗

以下範例定義了一個關鍵警報，當備份自訂資源的狀態設定為「是」時，警報將會被觸發。 `Error`持續5秒或更長時間。您可以自訂此範例以符合您的環境，並將此 YAML 程式碼片段包含在您的專案中。 `prometheus.yaml`設定檔：

rules.yaml：定義備份失敗的 **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."
```

配置 **Alertmanager** 向其他管道發送警報

您可以設定 Alertmanager，使其將通知傳送到其他管道，例如電子郵件、PagerDuty、Microsoft Teams 或其他通知服務，只需在設定檔中指定相應的配置即可。 `alertmanager.yaml`文件。

以下範例配置 Alertmanager 向 Slack 頻道發送通知。若要根據您的環境自訂此範例，請取代以下值： `api_url`金鑰包含您環境中使用的 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** 建立支援包

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為（例如， `trident-protect-support-bundle.yaml` ）。
2. 配置以下屬性：
   - **metadata.name**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.triggerType**: (*Required*) 確定支援包是立即產生還是按計劃產生。計劃的資料包產生時間為世界協調時凌晨 12 點。可能的值：
     - 已安排
     - 手動的
   - **spec.uploadEnabled**: (可選) 控制產生支援包後是否應將其上傳至NetApp支援網站。如果未指定，則預設為 `false`。可能的值：
     - 真的
     - false（預設值）
   - **spec.dataWindowStart**: (可選) RFC 3339 格式的日期字串，指定支援包中包含的資料視窗應開始的日期和時間。如果未指定，則預設為 24 小時前。您最早可以指定的日期範圍是 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. 填寫完後 `trident-protect-support-bundle.yaml`將檔案的值正確後，套用 CR：

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-
protect
```

使用 **CLI** 建立支援包

步驟

1. 建立支援包，將括號中的值替換為您環境中的資訊。這 `trigger-type`決定捆綁包是立即建立還是由計劃安排決定創建時間，並且可以是 `Manual`或者 `Scheduled`。預設是 `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`從輸出中讀取命令並運行它，將目標參數替換為您首選的本機目錄。

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。

> 從 24.10 版本升級時，升級期間執行的快照可能會失敗。此故障不會阻止將來建立快照，無論是手動建立還是計劃建立。如果在升級過程中快照失敗，您可以手動建立新的快照，以確保您的應用程式受到保護。
>
> 為避免潛在的故障，您可以在升級前停用所有快照計劃，並在升級後重新啟用它們。但是，這會導致在升級期間錯過任何計劃的快照。

若要升級Trident Protect，請執行下列步驟。

步驟

1. 更新Trident Helm 倉庫：

```
helm repo update
```

2. 升級Trident Protect CRD：

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. 執行此命令以將 CRD 的管理權從 `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. 執行此命令以刪除 Helm 金鑰 `trident-protect-crds`圖表：

不要卸載 `trident-protect-crds`使用 Helm 建立圖表可能會刪除您的 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 100.2506.0 --namespace trident-protect
```

管理和保護應用程式

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。

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。

配置 **AppVault** 身份驗證和密碼

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。

資料遷移儲存庫密碼

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，請使用 **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。

建立資料遷移儲存庫密碼金鑰

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。

- 根據您使用的資料傳輸工具，您只需輸入該資料傳輸工具對應的密碼即可。例如，如果您正在使用 Restic 並且將來不打算使用 Kopia，則在建立金鑰時，您可以只包含 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`命名空間已被刪除，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，請參閱以下範例： "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。欲了解更多信息，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 身分並將 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，請設定您的 AppVault CR。 `useIAM: true`使用標誌而不是顯式憑證：

```yaml
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
```

適用於雲端提供者的 **AppVault** 金鑰產生範例

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雲

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

亞馬遜 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
```

微軟 Azure

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

通用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 範例為每個雲端提供者建立 AppVault 物件。

- 您可以選擇指定一個 Kubernetes secret，其中包含 Restic 和 Kopia 儲存庫加密的自訂密碼。請參閱[資料遷移儲存庫密碼]了解更多。
- 對於 Amazon S3 (AWS) AppVault 對象，您可以選擇性地指定 sessionToken，如果您使用單一登入 (SSO) 進行驗證，這將非常有用。當您為提供者產生金鑰時，將建立此令牌。適用於雲端提供者的 AppVault 金鑰產生範例 。
- 對於 S3 AppVault 對象，您可以選擇性地使用下列方式指定出站 S3 流量的出口代理 URL：`spec.providerConfig.S3.proxyURL`鑰匙。

Google雲

```
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
```

亞馬遜 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
```

對於使用 Pod Identity 和 Kopia 資料移動器的 EKS 環境，您可以移除 `providerCredentials` 部分並添加 `useIAM: true`在 `s3`改為配置。

微軟 **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
```

通用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

```yaml
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

您可以使用以下 CLI 命令範例為每個提供者建立 AppVault CR。

- 您可以選擇指定一個 Kubernetes secret，其中包含 Restic 和 Kopia 儲存庫加密的自訂密碼。請參閱[資料遷移儲存庫密碼]了解更多。
- 對於 S3 AppVault 對象，您可以選擇性地使用下列方式指定出站 S3 流量的出口代理 URL：`--proxy-url <ip_address:port>`爭論。

Google雲

```
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
```

亞馬遜 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
```

微軟 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
```

通用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
```

查看 **AppVault** 訊息

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. 查看 AppVault 物件的內容：

   ```
   tridentctl-protect get appvaultcontent gcp-vault \
   --show-resources all \
   -n trident-protect
   ```

   範例輸出：

```
+-------------+-------+----------+-----------------------------
+---------------------------+
|   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）若要查看每個資源的 AppVaultPath，請使用該標誌 `--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**

您可以隨時刪除 AppVault 物件。

> 不要移除 `finalizers`在刪除 AppVault 物件之前，請在 AppVault CR 中輸入金鑰。這樣做可能會導致 AppVault 儲存桶中殘留數據，以及叢集中出現孤立資源。

開始之前

請確保您已刪除要刪除的 AppVault 所使用的所有快照和備份 CR。

使用 **Kubernetes CLI** 刪除 **AppVault**

1. 移除 AppVault 對象，並替換 `appvault-name`要刪除的 AppVault 物件的名稱：

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

使用**Trident Protect CLI** 刪除 **AppVault**

1. 移除 AppVault 對象，並替換 `appvault-name`要刪除的 AppVault 物件的名稱：

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

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。

使用 **CR** 新增應用程式

步驟

1. 建立目標應用程式 CR 檔案：
   a. 建立自訂資源 (CR) 檔案並將其命名為（例如， `maria-app.yaml` ）。
   b. 配置以下屬性：
      - **metadata.name**: (必填) 應用程式自訂資源的名稱。請注意您選擇的名稱，因為保護操作所需的其他 CR 檔案會引用此值。
      - **spec.includedNamespaces**: (必要) 使用命名空間和標籤選擇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包含在此清單中。標籤選擇器是可選的，可用來篩選每個指定命名空間內的資源。
      - **spec.includedClusterScopedResources**: (可選) 使用此屬性指定要包含在應用程式定義中的叢集範圍資源。此屬性可讓您根據資源的群組、版本、種類和標籤來選擇這些資源。
        - **groupVersionKind**: (必要) 指定叢集範圍資源的 API 群組、版本和類型。
        - **labelSelector**: (可選) 依照標籤篩選叢集範圍的資源。
      - **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" 。

如果需要在應用程式建立後套用此註解，可以使用以下命令：

```
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. （可選）新增篩選條件，包含或排除帶有特定標籤的資源：

   - **resourceFilter.resourceSelectionCriteria**：（篩選時必備）使用 `Include`或者 `Exclude`包含或排除 resourceMatchers 中定義的資源。新增以下 resourceMatchers 參數以定義要包含或排除的資源：

     - **resourceFilter.resourceMatchers**：resourceMatcher 物件陣列。如果在該數組中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的字段（組、種類、版本）之間按 AND 運算匹配。

       - **resourceMatchers[].group**: (可選) 要篩選的資源群組。
       - **resourceMatchers[].kind**: (可選) 要篩選的資源類型。
       - **resourceMatchers[].version**: (可選) 要篩選的資源版本。
       - **resourceMatchers[].names**: (可選) 要過濾的資源的 Kubernetes 元資料.name 欄位中的名稱。

- **resourceMatchers[].namespaces**: (可選) 要篩選的資源的 Kubernetes 元資料.name 欄位中的命名空間。
- **resourceMatchers[].labelSelectors**: (可選) 資源的 Kubernetes 元資料.name 欄位中的標籤選擇器字串，如在下列位置定義： "Kubernetes 文檔" 。例如： `"trident.netapp.io/os=linux"` 。

> ⓘ 當兩者 `resourceFilter`和 `labelSelector`被使用， `resourceFilter`先運行，然後 `labelSelector`應用於生成的資源。

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"]
```

2. 建立與您的環境相符的應用程式變更要求後，請套用該變更要求。例如：

```
kubectl apply -f maria-app.yaml
```

步驟

1. 使用以下範例之一建立並應用應用程式定義，將括號中的值替換為您環境中的資訊。您可以使用逗號分隔的列表，並在範例中顯示參數，將命名空間和資源包含在應用程式定義中。

   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，若要指定註解，可以使用以下方法： `--annotation`旗幟。

   - 建立應用程式並使用檔案系統凍結行為的全域設定：

```
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製檔案系統凍結行為：

```
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`用於根據以下條件包含或排除資源的標誌 `resourceSelectionCriteria`例如群組、類型、版本、標籤、名稱和命名空間，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"]}]'
```

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。

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"。

建立按需快照

您可以隨時建立按需快照。

如果叢集範圍的資源在應用程式定義中被明確引用，或被引用到任何應用程式命名空間，則這些資源將包含在備份、快照或複製中。

使用 **CR** 建立快照

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為 `trident-protect-snapshot-cr.yaml`。
2. 在您建立的文件中，配置以下屬性：
   - **metadata.name**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.applicationRef**: 要建立快照的應用程式的 Kubernetes 名稱。
   - **spec.appVaultRef**: (必要) 應儲存快照內容（元資料）的 AppVault 的名稱。
   - **spec.reclaimPolicy**: (可選) 定義當快照 CR 刪除時，快照的 AppArchive 會發生什麼情況。這意味著即使設定為 `Retain`快照將會被刪除。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. 填寫完後 `trident-protect-snapshot-cr.yaml`將檔案的值正確後，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，將括號中的值替換為您環境中的資訊。例如：

```
tridentctl-protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> -n
<application_namespace>
```

建立按需備份

您可以隨時備份應用程式。

如果叢集範圍的資源在應用程式定義中被明確引用，或被引用到任何應用程式命名空間，則這些資源將包含在備份、快照或複製中。

開始之前

確保 AWS 會話令牌過期時間足以滿足任何長時間運行的 S3 備份作業。如果在備份作業期間令牌過期，則操作可能會失敗。

- 請參閱 "AWS API 文件"有關檢查當前會話令牌過期時間的詳細資訊。
- 請參閱 "AWS IAM 文件"有關 AWS 資源憑證的詳細資訊。

使用 **CR** 建立備份

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為 `trident-protect-backup-cr.yaml`。
2. 在您建立的文件中，配置以下屬性：
   - **metadata.name**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.applicationRef**: (必要) 要備份的應用程式的 Kubernetes 名稱。
   - **spec.appVaultRef**: (必要) 應儲存備份內容的 AppVault 的名稱。
   - **spec.dataMover**: (可選) 指示要用於備份作業的備份工具的字串。可能的值（區分大小寫）：
     - `Restic`
     - `Kopia`(預設)
   - **spec.reclaimPolicy**: (可選) 定義從其聲明中釋放備份時會發生什麼。可能的值：
     - `Delete`
     - `Retain`(預設)
   - **spec.snapshotRef**: (可選): 要用作備份來源的快照名稱。如果未提供，則會建立並備份臨時快照。
   - **metadata.annotations.protect.trident.netapp.io/full-backup** : (可選) 此註解用於指定備份是否應為非增量備份。預設情況下，所有備份都是增量的。但是，如果此註釋設定為 `true`這樣，備份就變成了非增量備份。如果未指定，備份將遵循預設的增量備份設定。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。

     YAML 範例：

```yaml
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
  annotations:
    protect.trident.netapp.io/full-backup: "true"
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. 填寫完後 `trident-protect-backup-cr.yaml`將檔案的值正確後，套用 CR：

```
kubectl apply -f trident-protect-backup-cr.yaml
```

使用命令列建立備份

步驟

1. 建立備份，將括號中的值替換為您環境中的資訊。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>
```

您也可以選擇使用 `--full-backup`用於指定備份是否應為非增量備份的標誌。預設情況下，所有備份都是增量的。使用此標誌後，備份將變為非增量備份。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。

制定資料保護計劃

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。

- 您只能透過設定來建立快照計劃。 `backupRetention`歸零和 `snapshotRetention`取大於零的值。環境 `snapshotRetention`將快照值設為零表示任何排程備份仍會建立快照，但這些快照是暫時的，會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被明確引用，或被引用到任何應用程式命名空間，則這些資源將包含在備份、快照或複製中。

使用變更請求 **(CR)** 建立計劃。

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為 `trident-protect-schedule-cr.yaml`。
2. 在您建立的文件中，配置以下屬性：
   - **metadata.name**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.dataMover**: (可選) 指示要用於備份作業的備份工具的字串。可能的值（區分大小寫）：
     - `Restic`
     - `Kopia`(預設)
   - **spec.applicationRef**: 要備份的應用程式的 Kubernetes 名稱。
   - **spec.appVaultRef**: (必要) 應儲存備份內容的 AppVault 的名稱。
   - **spec.backupRetention**：要保留的備份數量。零表示不應建立備份（僅快照）。
   - **spec.snapshotRetention**：要保留的快照數量。零表示不建立任何快照。
   - **spec.granularity**: 計畫運行的頻率。可能的值以及所需的關聯欄位：
     - `Hourly`（需要您指定） `spec.minute` ）
     - `Daily`（需要您指定） `spec.minute`和 `spec.hour`）
     - `Weekly`（需要您指定） `spec.minute, spec.hour` ，和 `spec.dayOfWeek`）
     - `Monthly`（需要您指定） `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，則此欄位為必填項。 `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，則此欄位為必填項。 `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，則此欄位為必填項。 `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，則此欄位為必填項。 `Hourly` ， `Daily` ， `Weekly` ， 或者 `Monthly`。該值必須以字串形式提供。
   - **metadata.annotations.protect.trident.netapp.io/full-backup-rule**: (可選) 此註解用於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。

     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
  annotations:
    protect.trident.netapp.io/full-backup-rule: "Monday,Thursday"
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"
```

3. 填寫完後 `trident-protect-schedule-cr.yaml`將檔案的值正確後，套用 CR：

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

使用 **CLI** 建立計劃任務

步驟

1. 建立保護計劃，將括號中的值替換為您環境中的資訊。例如：

您可以使用 `tridentctl-protect create schedule --help`查看此命令的詳細協助資訊。

```
tridentctl-protect create schedule <my_schedule_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> --backup
-retention <how_many_backups_to_retain> --data-mover
<Kopia_or_Restic> --day-of-month <day_of_month_to_run_schedule>
--day-of-week <day_of_month_to_run_schedule> --granularity
<frequency_to_run> --hour <hour_of_day_to_run> --minute
<minute_of_hour_to_run> --recurrence-rule <recurrence> --snapshot
-retention <how_many_snapshots_to_retain> -n <application_namespace>
--full-backup-rule <string>
```

您可以設定 `--full-backup-rule`標記 `always`您可以根據需要進行持續完整備份或自訂備份。例如，如果您選擇按天粒度進行備份，則可以指定應在哪些工作天進行完整備份。例如，使用 `--full-backup-rule "Monday,Thursday"`安排每週一和週四進行全面備份。

對於僅快照計劃，請設定 `--backup-retention 0`並指定一個大於 `0` 的值 `--snapshot-retention`。

刪除快照

刪除不再需要的已排程或按需快照。

步驟

1. 刪除與快照關聯的快照 CR：

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

刪除備份

刪除不再需要的計劃備份或按需備份。

確保回收策略設定為 `Delete`從物件儲存中刪除所有備份資料。此策略的預設為： `Retain`避免意外資料遺失。如果政策不改變 `Delete`備份資料將保留在物件儲存中，需要手動刪除。

步驟

1. 刪除與備份關聯的備份 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的狀態。

步驟

1. 使用以下命令檢索備份作業的狀態，將方括號中的值替換為您環境中的資訊：

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

啟用 **Azure NetApp** 檔案 **(ANF)** 作業的備份和還原

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

確保以下事項：

- 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。
- 你有 `azure-netapp-files`已選為儲存後端的預設儲存類別。

展開檢視設定步驟

1. 如果 ANF 磁碟區是在升級到Trident 24.10 之前建立的，請在Trident中執行以下操作：

   a. 為每個基於 azure-netapp-files 且與應用程式關聯的 PV 啟用快照目錄：

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. 確認已為每個關聯的 PV 啟用快照目錄：

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   回覆:

   ```
   snapshotDirectory: "true"
   ```

   +
   如果未啟用快照目錄，則選擇常規備份功能，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。

恢復應用程式

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。

- 還原應用程式時，為該應用程式配置的所有執行鉤子都會隨應用程式一起復原。如果存在恢復後執行鉤子，它將作為恢復操作的一部分自動運行。
- 支援從備份還原到不同的命名空間或還原到原始命名空間（適用於 qtree 磁碟區）。但是，對於 qtree 卷，不支援從快照還原到不同的命名空間或還原到原始命名空間。
- 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"。

從備份還原到不同的命名空間

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。

> 將備份還原到具有現有資源的不同命名空間不會變更與備份中資源同名的任何資源。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，或將備份還原到新的命名空間。

開始之前

確保 AWS 會話令牌的過期時間足以滿足任何長時間運行的 S3 復原操作。如果在恢復操作期間令牌過期，則操作可能會失敗。

- 請參閱 "AWS API 文件"有關檢查當前會話令牌過期時間的詳細資訊。
- 請參閱 "AWS IAM 文件"有關 AWS 資源憑證的詳細資訊。

> 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 "科皮亞文檔"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) 檔案並將其命名為 `trident-protect-backup-restore-cr.yaml`。
2. 在您建立的文件中，配置以下屬性：
   - **metadata.name**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.appArchivePath**: AppVault 內儲存備份內容的路徑。您可以使用以下命令尋找此路徑：

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (必要) 儲存備份內容的 AppVault 的名稱。
   - **spec.namespaceMapping**：恢復作業的來源命名空間到目標命名空間的對應。代替 `my-source-namespace`和 `my-destination-namespace`利用來自周圍環境的資訊。

     ```
     ---
     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
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. （可選）如果您只需要選擇應用程式中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：

   > 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**：（篩選時必備）使用 `Include`或者 `Exclude`包含或排除 resourceMatchers 中定義的資源。新增以下 resourceMatchers 參數以定義要包含或排除的資源：
     - **resourceFilter.resourceMatchers**：resourceMatcher 物件陣列。如果在該數組中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的字段（組、種類、版本）之間按 AND 運算匹配。
       - **resourceMatchers[].group**: (可選) 要篩選的資源群組。
       - **resourceMatchers[].kind**: (可選) 要篩選的資源類型。

- **resourceMatchers[].version**: (可選) 要篩選的資源版本。
- **resourceMatchers[].names**: (可選) 要過濾的資源的 Kubernetes 元資料.name 欄位中的名稱。
- **resourceMatchers[].namespaces**: (可選) 要篩選的資源的 Kubernetes 元資料.name 欄位中的命名空間。
- **resourceMatchers[].labelSelectors**: (可選) 資源的 Kubernetes 元資料.name 欄位中的標籤選擇器字串，如在下列位置定義： "Kubernetes 文檔" 。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. 填寫完後 `trident-protect-backup-restore-cr.yaml`將檔案的值正確後，套用 CR：

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

使用 CLI

步驟

1. 將備份還原到不同的命名空間，並將括號中的值替換為您環境中的資訊。這 `namespace-mapping`此參數使用冒號分隔的命名空間，將來源命名空間對應到正確的目標命名空間，格式如下： `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> \
-n <application_namespace>
```

您可以隨時將備份還原到原始命名空間。

開始之前

確保 AWS 會話令牌的過期時間足以滿足任何長時間運行的 S3 復原操作。如果在恢復操作期間令牌過期，則操作可能會失敗。

- 請參閱 "AWS API 文件"有關檢查當前會話令牌過期時間的詳細資訊。
- 請參閱 "AWS IAM 文件"有關 AWS 資源憑證的詳細資訊。

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 "科皮亞文檔"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) 檔案並將其命名為 `trident-protect-backup-ipr-cr.yaml`。
2. 在您建立的文件中，配置以下屬性：
   - **metadata.name**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.appArchivePath**: AppVault 內儲存備份內容的路徑。您可以使用以下命令尋找此路徑：

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (必要) 儲存備份內容的 AppVault 的名稱。

     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. （可選）如果您只需要選擇應用程式中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：

   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**：（篩選時必備）使用 `Include`或者 `Exclude`包含或排除 resourceMatchers 中定義的資源。新增以下 resourceMatchers 參數以定義要包含或排除的資源：
     - **resourceFilter.resourceMatchers**：resourceMatcher 物件陣列。如果在該數組中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的字段（組、種類、版本）之間按 AND 運算匹配。
       - **resourceMatchers[].group**: (可選) 要篩選的資源群組。
       - **resourceMatchers[].kind**: (可選) 要篩選的資源類型。
       - **resourceMatchers[].version**: (可選) 要篩選的資源版本。
       - **resourceMatchers[].names**: (可選) 要過濾的資源的 Kubernetes 元資料.name 欄位中的名

稱。

- **resourceMatchers[].namespaces**: (可選) 要篩選的資源的 Kubernetes 元資料.name 欄位中的命名空間。
- **resourceMatchers[].labelSelectors**: (可選) 資源的 Kubernetes 元資料.name 欄位中的標籤選擇器字串，如在下列位置定義： "Kubernetes 文檔" 。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. 填寫完後 `trident-protect-backup-ipr-cr.yaml`將檔案的值正確後，套用 CR：

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

使用 **CLI**

步驟

1. 將備份還原到原始命名空間，並將括號中的值替換為您環境中的資訊。這 `backup`參數使用命名空間和備份名稱，格式如下 `<namespace>/<name>`。例如：

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

如果原始叢集出現問題，您可以將備份還原到其他叢集。

> 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 "科皮亞文檔"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。

開始之前

確保滿足以下先決條件：

- 目標叢集已安裝Trident Protect。
- 目標叢集可以存取與來源叢集相同的 AppVault 的儲存桶路徑，備份就儲存在該儲存桶中。
- 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 會話令牌的過期時間足以滿足任何長時間運行的復原操作。如果在恢復操作期間令牌過期，則操作可能會失敗。
  - 請參閱 "AWS API 文件"有關檢查當前會話令牌過期時間的詳細資訊。
  - 請參閱 "AWS 文件"有關 AWS 資源憑證的詳細資訊。

步驟

1. 使用Trident Protect CLI 外掛程式檢查目標叢集上 AppVault CR 的可用性：

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

> 確保用於應用程式還原的命名空間存在於目標叢集上。

2. 查看目標叢集中可用 AppVault 的備份內容：

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

執行此命令將顯示 AppVault 中可用的備份，包括其來源叢集、對應的應用程式名稱、時間戳記和歸檔路徑。

範例輸出：

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   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到目標叢集：

使用 **CR**

1. 建立自訂資源 (CR) 檔案並將其命名為 `trident-protect-backup-restore-cr.yaml`。
2. 在您建立的文件中，配置以下屬性：
   - **metadata.name**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.appVaultRef**: (必要) 儲存備份內容的 AppVault 的名稱。
   - **spec.appArchivePath**: AppVault 內儲存備份內容的路徑。您可以使用以下命令尋找此路徑：

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

     如果 BackupRestore CR 不可用，您可以使用步驟 2 中提到的指令來查看備份內容。

   - **spec.namespaceMapping**：恢復作業的來源命名空間到目標命名空間的對應。代替 `my-source-namespace`和 `my-destination-namespace`利用來自周圍環境的資訊。

     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
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. 填寫完後 `trident-protect-backup-restore-cr.yaml`將檔案的值正確後，套用 CR：

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

使用 **CLI**

1. 使用以下命令恢復應用程序，將括號中的值替換為您環境中的資訊。命名空間映射參數使用冒號分隔的命名空間，將來源命名空間對應到正確的目標命名空間，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> \
--context <destination_cluster_name> \
-n <application_namespace>
```

從快照還原到不同的命名空間

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。

SnapshotRestore 支持 `spec.storageClassMapping`屬性，但僅當來源儲存類別和目標儲存類別使用相同的儲存後端時才有效。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。

開始之前

確保 AWS 會話令牌的過期時間足以滿足任何長時間運行的 S3 復原操作。如果在恢復操作期間令牌過期，則操作可能會失敗。

- 請參閱 "AWS API 文件"有關檢查當前會話令牌過期時間的詳細資訊。
- 請參閱 "AWS IAM 文件"有關 AWS 資源憑證的詳細資訊。

使用 **CR**

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為 `trident-protect-snapshot-restore-cr.yaml`。
2. 在您建立的文件中，配置以下屬性：
   - **metadata.name**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.appVaultRef**: (必要) 儲存快照內容的 AppVault 的名稱。
   - **spec.appArchivePath**: AppVault 內儲存快照內容的路徑。您可以使用以下命令尋找此路徑：

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.namespaceMapping**：恢復作業的來源命名空間到目標命名空間的對應。代替 `my-source-namespace`和 `my-destination-namespace`利用來自周圍環境的資訊。

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. （可選）如果您只需要選擇應用程式中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：

   > 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**：（篩選時必備）使用 `Include`或者 `Exclude`包含或排除 resourceMatchers 中定義的資源。新增以下 resourceMatchers 參數以定義要包含或排除的資源：
     - **resourceFilter.resourceMatchers**：resourceMatcher 物件陣列。如果在該數組中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的字段（組、種類、版本）之間按 AND 運算匹配。
       - **resourceMatchers[].group**: (可選) 要篩選的資源群組。
       - **resourceMatchers[].kind**: (可選) 要篩選的資源類型。

- **resourceMatchers[].version**: (可選) 要篩選的資源版本。
- **resourceMatchers[].names**: (可選) 要過濾的資源的 Kubernetes 元資料.name 欄位中的名稱。
- **resourceMatchers[].namespaces**: (可選) 要篩選的資源的 Kubernetes 元資料.name 欄位中的命名空間。
- **resourceMatchers[].labelSelectors**: (可選) 資源的 Kubernetes 元資料.name 欄位中的標籤選擇器字串，如在下列位置定義： "Kubernetes 文檔" 。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. 填寫完後 `trident-protect-snapshot-restore-cr.yaml`將檔案的值正確後，套用 CR：

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

使用 CLI

步驟

1. 將快照還原到不同的命名空間，並將括號中的值替換為您環境中的資訊。

   - 這 `snapshot`參數使用命名空間和快照名稱，格式如下 `<namespace>/<name>`。
   - 這 `namespace-mapping`此參數使用冒號分隔的命名空間，將來源命名空間對應到正確的目標命名空間，格式如下： `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> \
-n <application_namespace>
```

從快照還原到原始命名空間

您可以隨時將快照還原到原始命名空間。

開始之前

確保 AWS 會話令牌的過期時間足以滿足任何長時間運行的 S3 復原操作。如果在恢復操作期間令牌過期，則操作可能會失敗。

- 請參閱 "AWS API 文件"有關檢查當前會話令牌過期時間的詳細資訊。
- 請參閱 "AWS IAM 文件"有關 AWS 資源憑證的詳細資訊。

使用 **CR**

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為 `trident-protect-snapshot-ipr-cr.yaml`。
2. 在您建立的文件中，配置以下屬性：
   - **metadata.name**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.appVaultRef**: (必要) 儲存快照內容的 AppVault 的名稱。
   - **spec.appArchivePath**: AppVault 內儲存快照內容的路徑。您可以使用以下命令尋找此路徑：

```
kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o
jsonpath='{.status.appArchivePath}'
```

```
---
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appVaultRef: appvault-name
    appArchivePath: my-snapshot-path
```

3. （可選）如果您只需要選擇應用程式中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：

> 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**：（篩選時必備）使用 `Include`或者 `Exclude`包含或排除 resourceMatchers 中定義的資源。新增以下 resourceMatchers 參數以定義要包含或排除的資源：
     - **resourceFilter.resourceMatchers**：resourceMatcher 物件陣列。如果在該數組中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的字段（組、種類、版本）之間按 AND 運算匹配。
       - **resourceMatchers[].group**: (可選) 要篩選的資源群組。
       - **resourceMatchers[].kind**: (可選) 要篩選的資源類型。
       - **resourceMatchers[].version**: (可選) 要篩選的資源版本。
       - **resourceMatchers[].names**: (可選) 要過濾的資源的 Kubernetes 元資料.name 欄位中的名稱。
       - **resourceMatchers[].namespaces**: (可選) 要篩選的資源的 Kubernetes 元資料.name 欄位中的命名空間。

- **resourceMatchers[].labelSelectors**: (可選) 資源的 Kubernetes 元資料.name 欄位中的標籤選擇器字串，如在下列位置定義： "Kubernetes 文檔" 。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. 填寫完後 `trident-protect-snapshot-ipr-cr.yaml`將檔案的值正確後，套用 CR：

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

使用 **CLI**

步驟

1. 將快照還原到原始命名空間，並將括號中的值替換為您環境中的資訊。例如：

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <snapshot_to_restore> \
-n <application_namespace>
```

檢查還原操作的狀態

您可以使用命令列來檢查正在進行、已完成或已失敗的還原作業的狀態。

步驟

1. 使用以下命令檢索恢復操作的狀態，將方括號中的值替換為您環境中的資訊：

```
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。

復原和故障轉移操作期間的命名空間註釋和標籤

在復原和故障轉移操作期間，目標命名空間中的標籤和註釋將與來源命名空間中的標籤和註釋相符。來源命名空間中不存在的目標命名空間中的標籤或註釋將被添加，並且任何已存在的標籤或註釋都將被覆蓋以匹配來源命名空間中的值。僅存在於目標命名空間中的標籤或註解保持不變。

> 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 安全上下文約束文檔"。

您可以透過設定 Kubernetes 環境變數來防止目標命名空間中的特定註解被覆蓋。`RESTORE_SKIP_NAMESPACE_ANNOTATIONS`在執行復原或故障轉移操作之前。例如：

```
helm upgrade trident-protect --set
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key
_to_skip_2> --reuse-values
```

> 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 "配置AutoSupport和命名空間過濾選項"。

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。

例子

以下範例展示了來源命名空間和目標命名空間，每個命名空間都有不同的註解和標籤。您可以查看操作前後目標命名空間的狀態，以及目標命名空間中的註解和標籤是如何組合或覆蓋的。

在恢復或故障轉移操作之前

下表說明了復原或故障轉移操作之前範例來源命名空間和目標命名空間的狀態：

命名空間	註解	標籤
命名空間 ns-1（來源）	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• 環境=生產 • 合規性=HIPAA • 名稱=ns-1
命名空間 ns-2（目標）	• annotation.one/key: "true" • annotation.three/key: "false"	• 角色=資料庫

恢復操作後

下表說明了復原或故障轉移作業後範例目標命名空間的狀態。有些按鍵已被添加，有些按鍵已被覆蓋，並且 `name`標籤已更新，以符合目標命名空間：

命名空間	註解	標籤
命名空間 ns-2（目標）	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• 名稱=ns-2 • 合規性=HIPAA • 環境=生產 • 角色=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`屬性定義了從來源應用程式中存在的儲存類別到目標叢集上新儲存類別的對應。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。

例：

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

支持的註釋

本節列出了系統中用於配置各種行為的支援註解。如果使用者沒有明確設定註釋，系統將使用預設值。

註解	類型	描述	預設值
protect.trident.netapp.io/data-mover-timeout-sec	細繩	資料移動器操作允許停止的最長時間（以秒為單位）。	“300”
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	細繩	Kopia 內容快取的最大大小限制（以兆位元組為單位）。	“1000”

使用NetApp SnapMirror和Trident Protect 複製應用程式

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。

復原和故障轉移操作期間的命名空間註釋和標籤

在復原和故障轉移操作期間，目標命名空間中的標籤和註釋將與來源命名空間中的標籤和註釋相符。來源命名空間中不存在的目標命名空間中的標籤或註釋將被添加，並且任何已存在的標籤或註釋都將被覆蓋以匹配來源命名空間中的值。僅存在於目標命名空間中的標籤或註解保持不變。

> ⓘ 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 安全上下文約束文檔"。

您可以透過設定 Kubernetes 環境變數來防止目標命名空間中的特定註解被覆蓋。`RESTORE_SKIP_NAMESPACE_ANNOTATIONS`在執行復原或故障轉移操作之前。例如：

```
helm upgrade trident-protect --set
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key
_to_skip_2> --reuse-values
```

> ⓘ 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 "配置AutoSupport和命名空間過濾選項"。

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。

例子

以下範例展示了來源命名空間和目標命名空間，每個命名空間都有不同的註解和標籤。您可以查看操作前後目標

命名空間的狀態，以及目標命名空間中的註解和標籤是如何組合或覆蓋的。

在恢復或故障轉移操作之前

下表說明了復原或故障轉移操作之前範例來源命名空間和目標命名空間的狀態：

命名空間	註解	標籤
命名空間 ns-1（來源）	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• 環境=生產 • 合規性=HIPAA • 名稱=ns-1
命名空間 ns-2（目標）	• annotation.one/key: "true" • annotation.three/key: "false"	• 角色=資料庫

恢復操作後

下表說明了復原或故障轉移作業後範例目標命名空間的狀態。有些按鍵已被添加，有些按鍵已被覆蓋，並且 `name`標籤已更新，以符合目標命名空間：

命名空間	註解	標籤
命名空間 ns-2（目標）	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• 名稱=ns-2 • 合規性=HIPAA • 環境=生產 • 角色=資料庫

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鉤子

使用 AppMirror 關係保護應用程式時，在故障轉移和反向操作期間，您應該注意與執行鉤子相關的特定行為。

- 故障轉移期間，執行鉤子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，應用程式上會存在執行鉤子，這些鉤子將在任何相關操作期間執行。
- 在反向同步或反向重同步期間，應用程式上任何現有的執行鉤子都會被移除。當來源應用程式變為目標應用程式時，這些執行鉤子將不再有效，並將被刪除以防止其執行。

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"。

建立複製關係

建立複製關係涉及以下步驟：

- 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）。

- 選擇複製計劃（包括 Kubernetes 資源以及持久卷資料）
- 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一個 AppVault。根據您的儲存供應商，修改範例中的內容。"AppVault 自訂資源"為了適應您的環境：

使用 **CR** 建立 **AppVault**

a. 建立自訂資源 (CR) 檔案並將其命名為（例如， `trident-protect-appvault-primary-source.yaml` ）。

b. 配置以下屬性：

- **metadata.name**: (必填) AppVault 自訂資源的名稱。請記下您選擇的名稱，因為複製關係所需的其他 CR 檔案會引用此值。
- **spec.providerConfig**: (必要) 儲存使用指定提供者存取 AppVault 所需的設定。選擇儲存桶名稱以及提供者所需的其他任何詳細資訊。請記下您選擇的值，因為複製關係所需的其他 CR 檔案會引用這些值。請參閱"AppVault 自訂資源"例如，AppVault CR 與其他提供者的合作案例。
- **spec.providerCredentials**: (*Required*) 儲存使用指定提供者存取 AppVault 所需的任何憑證的參考。
  - **spec.providerCredentials.valueFromSecret**: (*Required*) 表示憑證值應來自金鑰。
    - **key**: (必填) 要從中選取的有效金鑰。
    - **name**: (必填) 包含此欄位值的金鑰的名稱。必須位於同一命名空間中。
  - **spec.providerCredentials.secretAccessKey**: (必要) 用於存取提供者的存取金鑰。 名稱應與 **spec.providerCredentials.valueFromSecret.name** 相符。
- **spec.providerType**: (必要) 決定備份的提供者；例如， NetApp ONTAP S3、通用 S3、Google Cloud 或 Microsoft Azure。可能的值：
  - AWS
  - 蔚藍
  - 通用控制協議
  - 通用-s3
  - ontap-s3
  - 儲存網格-s3

c. 填寫完後 `trident-protect-appvault-primary-source.yaml`將檔案的值正確後，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，並將括號中的值替換為您環境中的資訊：

```
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) 檔案並將其命名為（例如， `trident-protect-app-source.yaml` ）。

b. 配置以下屬性：

- **metadata.name**: (必填) 應用程式自訂資源的名稱。請記下您選擇的名稱，因為複製關係所需的其他 CR 檔案會引用此值。
- **spec.includedNamespaces**: (必要) 命名空間及其關聯標籤的陣列。使用命名空間名稱，並可選擇使用標籤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包含在此數組中。

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. 填寫完後 `trident-protect-app-source.yaml`將檔案的值正確後，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 "建立按需快照"。

4. 在來源叢集上，建立複製計劃 CR：

除了下面提供的計劃之外，建議建立一個單獨的每日快照計劃，保留期為 7 天，以在對等ONTAP叢集之間保持共同的快照。這樣可以確保快照最多保留 7 天，但保留期限可以根據使用者需求進行自訂。

如果發生故障轉移，系統可以使用這些快照最多 7 天進行逆向操作。這種方法使得逆向過程更快、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，則無需制定其他計劃。

使用 **CR** 建立複製計劃

a. 為來源應用程式建立複製計劃：

   i. 建立自訂資源 (CR) 檔案並將其命名為（例如， `trident-protect-schedule.yaml` ）。

   ii. 配置以下屬性：

   - **metadata.name**: (必填) 計畫自訂資源的名稱。
   - **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*) 此欄位為必填項，其值必須設為 0。
   - **spec.enabled**：必須設定為 true。
   - **spec.granularity**：必須設定為 `Custom`。
   - **spec.recurrenceRule**: 定義 UTC 時間的開始日期和重複間隔。
   - **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. 填寫完後 `trident-protect-schedule.yaml`將檔案的值正確後，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>
```

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. 在目標叢集上，建立一個與在來源叢集上應用的 AppVault CR 完全相同的來源應用程式 AppVault CR，並將其命名為（例如， `trident-protect-appvault-primary-destination.yaml` ）。

6. 應用 CR：

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
trident-protect
```

7. 在目標叢集上為目標應用程式建立目標 AppVault CR。根據您的儲存供應商，修改範例中的內容。"AppVault 自訂資源"為了適應您的環境：

   a. 建立自訂資源 (CR) 檔案並將其命名為（例如， `trident-protect-appvault-secondary-destination.yaml` ）。

   b. 配置以下屬性：

      - **metadata.name**: (必填) AppVault 自訂資源的名稱。請記下您選擇的名稱，因為複製關係所需的其他 CR 檔案會引用此值。
      - **spec.providerConfig**: (必要) 儲存使用指定提供者存取 AppVault 所需的設定。選擇一個 `bucketName`以及其他任何您需要提供給服務提供者的詳細資訊。請記下您選擇的值，因為複製關係所需的其他 CR 檔案會引用這些值。請參閱"AppVault 自訂資源"例如，AppVault CR 與其他提供者的合作案例。
      - **spec.providerCredentials**: (*Required*) 儲存使用指定提供者存取 AppVault 所需的任何憑證的參考。
        - **spec.providerCredentials.valueFromSecret**: (*Required*) 表示憑證值應來自金鑰。
          - **key**: (必填) 要從中選取的有效金鑰。
          - **name**: (必填) 包含此欄位值的金鑰的名稱。必須位於同一命名空間中。
        - **spec.providerCredentials.secretAccessKey**: (必要) 用於存取提供者的存取金鑰。 名稱 應與 **spec.providerCredentials.valueFromSecret.name** 相符。

- **spec.providerType**: (必要) 決定備份的提供者；例如， NetApp ONTAP S3、通用 S3、Google Cloud 或 Microsoft Azure。可能的值：
  - AWS
  - 蔚藍
  - 通用控制協議
  - 通用-s3
  - ontap-s3
  - 儲存網格-s3

c. 填寫完後 `trident-protect-appvault-secondary-destination.yaml`將檔案的值正確後，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：

使用 **CR** 建立 **AppMirrorRelationship**

a. 建立自訂資源 (CR) 檔案並將其命名為（例如， `trident-protect-relationship.yaml` ）。

b. 配置以下屬性：

- **metadata.name:**（必填）AppMirrorRelationship 自訂資源的名稱。
- **spec.destinationAppVaultRef**: (必要) 此值必須與目標叢集上目標應用程式的 AppVault 名稱相符。
- **spec.namespaceMapping**: (必要) 目標命名空間和來源命名空間必須與對應應用程式 CR 中定義的應用程式命名空間相符。
- **spec.sourceAppVaultRef**: (必要) 此值必須與來源應用程式的 AppVault 名稱相符。
- **spec.sourceApplicationName**: (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名稱相符。
- **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.recurrenceRule**: 定義 UTC 時間的開始日期和重複間隔。

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. 填寫完後 `trident-protect-relationship.yaml`將檔案的值正確後，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> -n <application_namespace>
```

例：

```
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. （可選）在目標叢集上，檢查複製關係的狀態：

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

故障轉移到目標集群

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。

步驟

1. 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`。
2. 儲存CR文件。

3. 應用 CR：

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. （可選）在故障轉移應用程式上建立所需的任何保護計劃。
5. （可選）檢查複製關係的狀態：

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

重新同步失敗的複製關係

重新同步操作會重新建立複製關係。執行重新同步操作後，原始來源應用程式將成為正在運行的應用程序，對目標叢集上正在運行的應用程式所做的任何更改都將被丟棄。

該過程會在重新建立複製之前停止目標叢集上的應用程式。

故障轉移期間寫入目標應用程式的任何資料都會遺失。

步驟

1. （可選）在來源叢集上，建立來源應用程式的快照。這樣可以確保捕獲源集群的最新變更。
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文件。
4. 應用 CR：

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. 如果您在目標叢集上建立了任何保護計劃來保護故障轉移應用程序，請將其刪除。任何殘留的計劃都會導致磁碟區快照失敗。

反向重新同步失敗的複製關係

當您反向同步故障轉移複製關係時，目標應用程式將變為來源應用程序，而來源應用程式將變為目標應用程式。故障轉移期間對目標應用程式所做的變更將被保留。

步驟

1. 在原始目標叢集上，刪除 AppMirrorRelationship CR。這導致目的地變成了出發地。如果新目標叢集上還有任何剩餘的保護計劃，請將其刪除。
2. 透過將最初用於建立關係的 CR 檔案套用到相反的叢集來建立複製關係。
3. 確保新目標（原始來源叢集）配置了兩個 AppVault CR。
4. 在相反的集群上建立複製關係，並配置反向的值。

反向應用程式複製方向

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，你交換了來源位址和目標位址。

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策略計劃。

b. 建立 ShutdownSnapshot CR 檔案：

i. 建立自訂資源 (CR) 檔案並將其命名為（例如，`trident-protect-shutdownsnapshot.yaml`）。

ii. 配置以下屬性：

- **metadata.name**: (必填) 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**: (必要) 此值必須與來源應用程式 CR 檔案的 metadata.name 欄位相符。

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. 填寫完後 `trident-protect-shutdownsnapshot.yaml`將檔案的值正確後，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，將括號中的值替換為您環境中的資訊。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，使用以下命令尋找 **shutdownsnapshot.status.appArchivePath** 的值，並記錄檔案路徑的最後一部分（也稱為基本名稱；這將是最後一個斜線之後的所有內容）：

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. 從新的目標叢集到新的來源叢集執行故障轉移，並進行以下變更：

> 在故障轉移流程的第 2 步驟中，包括：`spec.promotedSnapshot`在 AppMirrorRelationship CR 檔案中，將該欄位的值設定為您在上面的步驟 3 中記錄的基本名稱。

5. 執行反向重新同步步驟[反向重新同步失敗的複製關係]。
6. 在新來源叢集上啟用保護計劃。

結果

由於反向複製，會發生以下操作：

- 對原始來源應用程式的 Kubernetes 資源進行快照。
- 透過刪除應用程式的 Kubernetes 資源（保留 PVC 和 PV），優雅地停止原始來源應用程式的 pod。
- 在 pod 關閉後，會對應用程式的磁碟區進行快照並進行複製。
- SnapMirror關係已斷開，目標磁碟區已準備好進行讀取/寫入操作。
- 該應用程式的 Kubernetes 資源是從關閉前的快照中恢復的，使用的是在原始來源應用程式關閉後複製的捲資料。
- 複製過程以相反的方向重新建立。

將應用程式故障恢復到原始來源集群

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。

流程從已完成故障轉移至目標位置的關係開始，並涉及以下步驟：

- 從故障轉移狀態開始。
- 反向重新同步複製關係。

> 不要執行正常的重新同步操作，因為這將丟棄在故障轉移過程中寫入目標叢集的資料。

- 反轉複製方向。

步驟

1. 執行[反向重新同步失敗的複製關係]步驟。
2. 執行[反向應用程式複製方向]步驟。

刪除複製關係

您可以隨時刪除複製關係。刪除應用程式複製關係後，將產生兩個彼此獨立的應用程序，它們之間沒有任何關係。

步驟

1. 在目前目標叢集上，刪除 AppMirrorRelationship CR：

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

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。

遷移應用程式時，為該應用程式配置的所有執行鉤子都會隨應用程式一起遷移。如果存在恢復後執行鉤子，它將作為恢復操作的一部分自動運行。

備份和復原作業

針對以下場景，您可以自動執行特定的備份和復原任務，以執行備份和復原作業。

克隆到同一群集

若要將應用程式複製到同一個集群，請建立快照或備份，然後將資料還原到同一個集群。

步驟

1. 執行下列操作之一：
   a. "建立快照"。
   b. "建立備份"。
2. 在同一群集上，根據您建立的是快照還是備份，執行下列其中一項：
   a. "從快照恢復數據"。
   b. "從備份中恢復數據"。

克隆到不同的集群

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。

您可以使用以下方法在不同的叢集之間複製應用程式"SnapMirror複製"。

步驟

1. "建立備份"。
2. 確保目標叢集上已配置包含備份的物件儲存桶的 AppVault CR。
3. 在目標集群上，"從備份中恢復數據" 。

將應用程式從一個儲存類別遷移到另一個儲存類

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（不包括恢復 CR 中的金鑰）：

```
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** 恢復快照

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為 `trident-protect-snapshot-restore-cr.yaml`。
2. 在您建立的文件中，配置以下屬性：
   - **metadata.name**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.appArchivePath**: AppVault 內儲存快照內容的路徑。您可以使用以下命令尋找此路徑：

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (必要) 儲存快照內容的 AppVault 的名稱。
   - **spec.namespaceMapping**：恢復作業的來源命名空間到目標命名空間的對應。代替 `my-source-namespace`和 `my-destination-namespace`利用來自周圍環境的資訊。

     ```
     ---
     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. 如果您需要僅選擇要復原的應用程式的某些資源，則可以新增篩選條件，以包含或排除具有特定標籤的資源：
   - **resourceFilter.resourceSelectionCriteria**：（篩選時必備）使用 `include or exclude`包含或排除 resourceMatchers 中定義的資源。新增以下 resourceMatchers 參數以定義要包含或排除的資源：
     - **resourceFilter.resourceMatchers**：resourceMatcher 物件陣列。如果在該數組中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的字段（組、種類、版本）之間按 AND 運算匹配。
       - **resourceMatchers[].group**: (可選) 要篩選的資源群組。
       - **resourceMatchers[].kind**: (可選) 要篩選的資源類型。
       - **resourceMatchers[].version**: (可選) 要篩選的資源版本。
       - **resourceMatchers[].names**: (可選) 要過濾的資源的 Kubernetes 元資料.name 欄位中的名稱。
       - **resourceMatchers[].namespaces**: (可選) 要篩選的資源的 Kubernetes 元資料.name 欄位中的命名空間。

- **resourceMatchers[].labelSelectors**: (可選) 資源的 Kubernetes 元資料.name 欄位中的標籤選擇器字串，如在下列位置定義： "Kubernetes 文檔" 。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. 填寫完後 `trident-protect-snapshot-restore-cr.yaml`將檔案的值正確後，套用 CR：

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

使用 **CLI** 恢復快照

步驟

1. 將快照還原到不同的命名空間，並將括號中的值替換為您環境中的資訊。

   - 這 `snapshot`參數使用命名空間和快照名稱，格式如下 `<namespace>/<name>`。
   - 這 `namespace-mapping`此參數使用冒號分隔的命名空間，將來源命名空間對應到正確的目標命名空間，格式如下： `source1:dest1,source2:dest2` 。

     例如：

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

管理Trident Protect 執行鉤子

執行鉤子是一種自訂操作，您可以將其配置為與受管應用程式的資料保護操作一起運行。例如，如果您有一個資料庫應用程序，則可以使用執行掛鉤在快照之前暫停所有資料庫事務，並在快照完成後恢復事務。這確保了應用程式一致的快照。

執行鉤子的類型

Trident Protect 支援以下幾種執行鉤子類型，取決於它們的運行時機：

- 預快照
- 快照後
- 預備份
- 備份後
- 恢復後
- 故障轉移後

執行順序

當執行資料保護操作時，執行掛鉤事件會依照下列順序發生：

1. 任何適用的自訂預操作執行掛鉤都在適當的容器上運行。您可以根據需要建立和運行任意數量的自訂預操作掛鉤，但這些掛鉤在操作之前的執行順序既無法保證也無法配置。
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. 執行資料保護操作。
4. 如果適用，凍結的檔案系統將被解凍。
5. 任何適用的自訂後操作執行掛鉤都在適當的容器上運行。您可以根據需要建立和運行任意數量的自訂後操作掛鉤，但操作後這些掛鉤的執行順序既無法保證也無法配置。

如果您建立多個相同類型的執行掛鉤（例如，預快照），則無法保證這些掛鉤的執行順序。但是，不同類型的鉤子的執行順序是有保證的。例如，以下是具有所有不同類型鉤子的配置的執行順序：

1. 快照前鉤子執行
2. 快照後鉤子執行
3. 執行備份前掛鉤
4. 執行備份後鉤子

前面的順序範例僅適用於執行不使用現有快照的備份時。

在生產環境中啟用執行掛鉤腳本之前，您應該始終對其進行測試。您可以使用“kubectl exec”命令方便地測試腳本。在生產環境中啟用執行掛鉤後，測試產生的快照和備份以確保它們一致。您可以透過將應用程式複製到臨時命名空間、還原快照或備份，然後測試應用程式來執行此操作。

如果快照前執行鉤子新增、變更或刪除 Kubernetes 資源，則這些變更將包含在快照或備份以及任何後續復原作業中。

關於自訂執行鉤子的重要說明

在為您的應用程式規劃執行掛鉤時，請考慮以下事項。

- 執行鉤子必須使用腳本來執行操作。許多執行鉤子可以引用同一個腳本。
- 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。
- 腳本大小限制為 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。

由於執行鉤子通常會減少或完全停用其所針對的應用程式的功能，因此您應該始終嘗試盡量減少自訂執行鉤子的運行時間。如果您啟動具有相關執行掛鉤的備份或快照操作，但隨後取消它，則如果備份或快照操作已經開始，則仍允許掛鉤運行。這意味著備份後執行掛鉤中使用的邏輯不能假定備份已完成。

執行鉤子過濾器

當您為應用程式新增或編輯執行掛鉤時，您可以向執行掛鉤添加過濾器來管理該掛鉤將匹配哪些容器。過濾器對於在所有容器上使用相同容器鏡像但可能將每個鏡像用於不同目的的應用程式（例如 Elasticsearch）很有用。過濾器可讓您建立執行掛鉤在某些（但不一定是所有）相同的容器上運行的場景。如果為單一執行掛鉤建立多個篩選器，它們將透過邏輯 AND 運算子組合在一起。每個執行掛鉤最多可以有 10 個活動過濾器。

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 "正規表示式 2 (RE2) 語法支持"。

如果將命名空間過濾器新增至在復原或複製作業後執行的執行掛鉤，且復原或複製來源和目標位於不同的命名空間中，則命名空間篩選器僅適用於目標命名空間。

執行鉤子範例

訪問 "NetApp Verda GitHub 項目"下載流行應用程式（如 Apache Cassandra 和 Elasticsearch）的真實執行掛鉤。您還可以查看範例並獲得建立自己的自訂執行掛鉤的想法。

建立執行鉤子

您可以使用以下方法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) 檔案並將其命名為 `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**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.applicationRef**: (必要) 要執行執行鉤子的應用程式的 Kubernetes 名稱。
   - **spec.stage**: (*Required*) 一個字串，指示執行鉤子應該在操作的哪個階段運行。可能的值：
     - 預
     - 郵政
   - **spec.action**: (*Required*) 一個字串，指示執行鉤子將採取什麼操作，假設指定的任何執行鉤子過濾器都匹配。可能的值：
     - 快照
     - 備份
     - 恢復
     - 故障轉移
   - **spec.enabled**: (可選) 指示此執行鉤子是否已啟用或停用。如果未指定，則預設值為 true。
   - **spec.hookSource**: (必要) 包含 base64 編碼的 hook 腳本的字串。
   - **spec.timeout**: (可選) 定義執行鉤子允許運行的分鐘數的數字。最小值為 1 分鐘，如果未指定，則預設值為 25 分鐘。
   - **spec.arguments**: (可選) 一個 YAML 列表，用於指定執行鉤子的參數。
   - **spec.matchingCriteria**: (可選) 一個可選的條件鍵值對列表，每個鍵值對構成一個執行鉤子過濾器。每個執行鉤子最多可以添加 10 個過濾器。
   - **spec.matchingCriteria.type**: (可選) 用於識別執行鉤過濾器類型的字串。可能的值：
     - 容器影像
     - 容器名稱
     - Pod名稱
     - PodLabel
     - 命名空間名稱
   - **spec.matchingCriteria.value**: (可選) 用於識別執行鉤過濾器值的字串或正規表示式。

     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. 在用正確的值填入 CR 檔案後，套用 CR：

```
kubectl apply -f trident-protect-hook.yaml
```

使用 CLI

步驟

1. 建立執行鉤子，將括號中的值替換為您環境中的資訊。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>
```

手動運行執行鉤子

您可以手動執行執行鉤子進行測試，或者在失敗後需要手動重新運行鉤子時也可以這樣做。您需要擁有所有者、管理員或成員權限才能手動執行執行鉤子。

手動運行執行鉤子包含兩個基本步驟：

1. 建立資源備份，該備份會收集資源並建立它們的備份，從而確定鉤子函數的運作位置。
2. 針對備份運行執行鉤子

步驟 **1**：建立資源備份

使用 **CR**

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為 `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**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.applicationRef**: (必要) 要為其建立資源備份的應用程式的 Kubernetes 名稱。
   - **spec.appVaultRef**: (必要) 儲存備份內容的 AppVault 的名稱。
   - **spec.appArchivePath**: AppVault 內儲存備份內容的路徑。您可以使用以下命令尋找此路徑：

     ```
     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. 在用正確的值填入 CR 檔案後，套用 CR：

   ```
   kubectl apply -f trident-protect-resource-backup.yaml
   ```

使用 **CLI**

步驟

1. 建立備份，將括號中的值替換為您環境中的資訊。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. 查看備份狀態。您可以重複使用此範例命令，直到操作完成：

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. 確認備份是否成功：

```
kubectl describe resourcebackup <my_backup_name>
```

步驟 **2**：運行執行鉤子

使用 **CR**

步驟

1. 建立自訂資源 (CR) 檔案並將其命名為 `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**: (必填) 此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.applicationRef**: (必需) 確保此值與您在步驟 1 中建立的 ResourceBackup CR 中的應用程式名稱相符。
   - **spec.appVaultRef**: (必需) 確保此值與您在步驟 1 中建立的 ResourceBackup CR 中的 appVaultRef 相符。
   - **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*) 一個字串，指示執行鉤子將採取什麼操作，假設指定的任何執行鉤子過濾器都匹配。可能的值：
     - 快照
     - 備份
     - 恢復
     - 故障轉移
   - **spec.stage**: (*Required*) 一個字串，指示執行鉤子應該在操作的哪個階段運行。這次鉤子運行不會在任何其他階段運行鉤子。可能的值：
     - 預
     - 郵政

     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. 在用正確的值填入 CR 檔案後，套用 CR：

```
kubectl apply -f trident-protect-hook-run.yaml
```

使用 **CLI**

步驟

1. 建立手動執行鉤子運行請求：

```
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. 檢查執行鉤子運行狀態。您可以重複執行此命令，直到操作完成：

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. 描述 exechooksrun 物件以查看最終詳細資訊和狀態：

```
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：

   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5月9日："使用 Amazon EKS 外掛程式為 FSx for ONTAP自動設定Trident後端"
- 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4月15日："NetApp Trident與Google Cloud NetApp Volumes for 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 系統在 Kubernetes 區塊儲存方面的強大功能"
- 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："解鎖無縫的 iSCSI 儲存整合：AWS ROSA 叢集上的 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引入光纖通道支援"
- 2024年11月26日："Trident 25.01：透過新功能和增強功能增強 Kubernetes 儲存體驗"
- 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："OpenShift虛擬化虛擬機器的崩潰一致性備份與復原作業"
- 2025年3月11日："將 GitOps 模式擴展到使用NetApp Trident進行應用程式資料保護"
- 2025年3月3日："Trident 25.02：透過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： "隆重介紹 tridentctl protect： Trident Protect 的強大命令列介面"
- 2024年11月11日："Kubernetes驅動的資料管理： Trident Protect開啟新時代"

知識和支持

常見問題解答

尋找有關Trident 的安裝、設定、升級和故障排除的常見問題。

一般性問題

Trident 的發布頻率如何？

從 24.02 年版本開始， Trident每四個月發布一次：二月、六月和十月。

Trident是否支援特定版本 **Kubernetes** 中發布的所有功能？

Trident通常不支援 Kubernetes 中的 alpha 功能。 Trident可能會在 Kubernetes beta 版本之後的兩個Trident版本中支援 beta 功能。

Trident 的運作是否依賴其他**NetApp**產品？

Trident不依賴其他NetApp軟體產品，它可以作為獨立應用程式運作。但是，您應該使用NetApp後端儲存設備。

如何取得完整的**Trident**設定詳情？

使用 `tridentctl get`命令以獲取有關您的Trident配置的更多資訊。

我能否取得**Trident**如何設定儲存的指標？

是的。可用於收集有關Trident操作的資訊的 Prometheus 端點，例如管理的後端數量、已配置的磁碟區數量、消耗的位元組數等等。您也可以使用"Cloud Insights"用於監測和分析。

使用**Trident**作為 **CSI** 設定器時，使用者體驗是否會改變？

不，使用者體驗和功能方面沒有任何變化。使用的設定程式名稱是 `csi.trident.netapp.io`。如果您想使用目前和未來版本提供的所有新功能，建議使用此方法安裝Trident 。

在 **Kubernetes** 叢集上安裝和使用**Trident**

Trident是否支援從私人註冊表進行離線安裝？

是的， Trident可以離線安裝。參考"了解Trident安裝"。

我可以遠端安裝**Trident**嗎？

是的。Trident 18.10 及更高版本支援從任何具備以下功能的機器進行遠端安裝： `kubectl`訪問集群。後`kubectl`存取權限已驗證（例如，發起一次存取） `kubectl get nodes`從遠端機器發出命令進行驗證），請按照安裝說明進行操作。

我可以使用**Trident**設定高可用性嗎？

Trident以 Kubernetes Deployment（ReplicaSet）的形式安裝，只有一個實例，因此它內建了高可用性 (HA)。您不應該增加 Deployment 中的副本數量。如果安裝了Trident 的節點遺失或 pod 無法訪問，Kubernetes 會自動將 pod 重新部署到叢集中的健康節點。 Trident僅用於控制平面，因此如果重新部署Trident，目前已安裝的 pod 不會受到影響。

Trident是否需要存取 **kube-system** 命名空間？

Trident從 Kubernetes API 伺服器讀取訊息，以確定應用程式何時要求新的 PVC，因此它需要存取 kube-system。

Trident使用哪些角色和權限？

Trident安裝程式會建立一個 Kubernetes ClusterRole，該角色對 Kubernetes 叢集的 PersistentVolume、PersistentVolumeClaim、StorageClass 和 Secret 資源有特定的存取權。參考"自訂 tridentctl 安裝"。

我可以在本地產生**Trident**安裝時使用的確切清單檔案嗎？

如有需要，您可以在本機產生和修改Trident安裝時使用的確切清單檔案。參考"自訂 tridentctl 安裝"。

我可以為兩個獨立的 **Kubernetes** 叢集中的兩個獨立的**Trident**實例共用同一個**ONTAP**後端 **SVM** 嗎？

雖然不建議這樣做，但您可以為兩個Trident實例使用同一個後端 SVM。安裝期間為每個實例指定唯一的磁碟區名稱和/或指定唯一的 `StoragePrefix`參數 `setup/backend.json`文件。這是為了確保兩個實例不會使用同一個FlexVol volume。

是否可以在 **ContainerLinux**（原 **CoreOS**）下安裝**Trident** ？

Trident其實就是一個 Kubernetes pod，可以安裝在任何執行 Kubernetes 的地方。

我可以將**Trident**與**NetApp Cloud Volumes ONTAP**一起使用嗎？

是的， Trident在 AWS、Google Cloud 和 Azure 上均受支援。

Trident是否與 **Cloud Volumes Services** 相容？

是的， Trident支援 Azure 中的Azure NetApp Files服務以及 GCP 中的Cloud Volumes Service。

故障排除和支持

NetApp是否支援**Trident**？

雖然Trident是開源且免費提供的，但只要您的NetApp後端受支持， NetApp就會完全支援它。

我該如何提交支持申請？

如需提出援助申請，請執行以下操作之一：

1. 請聯絡您的支援客戶經理，以取得協助以提交工單。
2. 請聯絡我們提交支援案例。 "NetApp支援" 。

如何產生支援日誌包？

您可以透過執行以下命令來建立支援包 `tridentctl logs -a`。除了捆綁包中捕獲的日誌外，還要捕獲 kubelet 日誌，以診斷 Kubernetes 端的掛載問題。取得 kubelet 日誌的說明會因 Kubernetes 的安裝方式而異。

如果我需要提出新功能請求，該怎麼辦？

創建問題 "Trident Github"並在問題的主題和描述中註明 **RFE**。

我應該在哪裡提交缺陷報告？

創建問題 "Trident Github"。請務必提供與問題相關的所有必要資訊和日誌。

如果我對**Trident**有個需要澄清的簡短問題，該怎麼辦？這裡有社群或論壇嗎？

如果您有任何疑問、問題或請求，請透過我們的Trident與我們聯絡。"Discord 頻道"或者 GitHub。

我的儲存系統密碼已更改， **Trident**無法再工作，我該如何恢復？

使用以下方式更新後端密碼 `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`。代替 `myBackend`例如，請使用您的後端名稱，並且 `/path/to_new_backend.json`通往正確路徑 `backend.json`文件。

Trident找不到我的**Kubernetes**節點。我該如何解決這個問題？

Trident找不到 Kubernetes 節點可能有以下兩種情況。這可能是由於 Kubernetes 內部的網路問題或 DNS 問題引起的。在每個 Kubernetes 節點上執行的Trident節點守護程序集必須能夠與Trident控制器通信，以便將節點註冊到Trident。如果在Trident安裝後發生了網路變更，則只有在向叢集中新增的 Kubernetes 節點時才會遇到此問題。

如果**Trident**艙被摧毀，我會遺失資料嗎？

即使Trident艙被摧毀，資料也不會遺失。 Trident元資料儲存在 CRD 物件中。所有由Trident提供的PV都將正常運作。

升級**Trident**

我可以直接從舊版本升級到新版本（跳過幾個版本）嗎？

NetApp支援將Trident從一個主要版本升級到下一個緊鄰的主要版本。您可以從 18.xx 版本升級到 19.xx 版本，從 19.xx 版本升級到 20.xx 版本，依此類推。在生產環境部署之前，應該在實驗室環境中測試升級。

是否可以將**Trident**降級到先前的版本？

如果您需要修復升級後發現的錯誤、依賴關係問題或升級失敗/不完整，您應該"解除安裝Trident"然後按照該版本的具體說明重新安裝早期版本。這是降級到早期版本的唯一推薦方法。

管理後端和卷

我是否需要在**ONTAP**後端定義檔中同時定義 **ManagementLIF** 和 **DataLIF**？

管理層 LIF 是強制性的。 DataLIF 值各不相同：

- ONTAP SAN：不要指定用於 iSCSI。Trident的使用"ONTAP選擇性 LUN 地圖"發現建立多路徑會話所需的 iSCI LIF。如果出現以下情況，則會產生警告： `dataLIF`已明確定義。參考 "ONTAP SAN 配置選項和範例"了解詳情。
- ONTAP NAS： 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。請參閱"ONTAP NAS 設定選項和範例"詳情請見

Trident能否為**ONTAP**後端設定**CHAP**？

是的。Trident支援ONTAP後端的雙向 CHAP。這需要設定 `useCHAP=true`在您的後端配置中。

如何使用**Trident**管理匯出策略？

從 20.04 版本開始， Trident可以動態建立和管理匯出策略。這樣，儲存管理員就可以在其後端配置中提供一個或多個 CIDR 區塊，並讓Trident將落入這些範圍內的節點 IP 新增到它所建立的匯出策略中。透過這種方式，Trident可以自動管理給定 CIDR 內 IP 位址的節點的規則新增和刪除。

IPv6 位址可以用於管理和資料 **LIF** 嗎？

Trident支援為下列裝置定義 IPv6 位址：

- `managementLIF`和 `dataLIF`適用於ONTAP NAS 後端。
- `managementLIF`適用於ONTAP SAN 後端。您無法指定 `dataLIF`基於ONTAP SAN 後端。

必須使用標誌安裝Trident 。 `--use-ipv6` （為了 `tridentctl`**安裝**）， `IPv6` （對於Trident操作員），或 `tridentTPv6`（用於 Helm 安裝）使其能夠透過 IPv6 運作。

是否可以在後端更新管理 **LIF**？

是的，可以使用以下方式更新後端管理 LIF： `tridentctl update backend`命令。

是否可以在後端更新 **DataLIF**？

您可以更新 DataLIF `ontap-nas`和 `ontap-nas-economy`僅有的。

我可以在**Trident for Kubernetes** 中建立多個後端嗎？

Trident可以同時支援多個後端，既可以使用相同的驅動程序，也可以使用不同的驅動程式。

Trident是如何儲存後端憑證的？

Trident將後端憑證儲存為 Kubernetes Secrets。

Trident是如何選擇特定後端的？

如果後端屬性無法用於自動為類別選擇適當的連線池，則 `storagePools`和 `additionalStoragePools`參數用於選擇一組特定的池子。

如何確保**Trident**不會從特定的後端進行設定？

這 `excludeStoragePools`此參數用於篩選Trident用於配置的池集，並將刪除任何符合的池。

如果存在多個相同類型的後端， **Trident**如何選擇使用哪個後端？

如果配置了多個相同類型的後端， Trident會根據配置中的參數選擇適當的後端。 `StorageClass`和 `PersistentVolumeClaim`。例如，如果存在多個 ontap-nas 驅動程式後端， Trident會嘗試匹配參數。 `StorageClass`和 `PersistentVolumeClaim`結合併匹配一個能夠滿足所列要求的後端 `StorageClass`和 `PersistentVolumeClaim`。如果存在多個後端與請求匹配， Trident會隨機選擇其中一個。

Trident是否支援與 **Element/ SolidFire** 的雙向 **CHAP** 協定？

是的。

Trident如何在**ONTAP**磁碟區部署 **Qtree**？單一磁碟區上最多可以部署多少個 **Qtree**？

這 `ontap-nas-economy`驅動程式在同一個FlexVol volume中最多可建立 200 個 Qtree（可在 50 到 300 之間配置），每個叢集節點最多可建立 100,000 個 Qtree，每個叢集最多可建立 240 萬個 Qtree。當你進入一個新的 `PersistentVolumeClaim`如果由經濟型驅動程式提供服務，則該驅動程式會檢視是否已存在可為新的 Qtree 提供服務的FlexVol volume。如果不存在可以為 Qtree 提供服務的FlexVol volume，則會建立一個新的FlexVol volume。

如何為**ONTAP NAS** 上配置的磁碟區設定 **Unix** 權限？

您可以透過在後端定義檔中設定參數，為Trident提供的磁碟區設定 Unix 權限。

在配置磁碟區時，如何配置一組明確的**ONTAP NFS** 掛載選項？

預設情況下， Trident不會為 Kubernetes 設定任何掛載選項。若要在 Kubernetes 儲存類別中指定掛載選項，請依照給定的範例進行操作。"這裡" 。

如何將已配置的磁碟區設定為特定的匯出策略？

若要允許對應的主機存取卷，請使用下列方法： `exportPolicy`在後端定義檔中配置的參數。

如何使用**Trident**和**ONTAP**設定磁碟區加密？

您可以使用後端定義檔中的加密參數，對Trident提供的磁碟區設定加密。更多信息，請參閱："Trident如何與 NVE 和 NAE 協同工作"

透過**Trident**為**ONTAP**實現 **QoS** 的最佳方法是什麼？

使用 `StorageClasses`為ONTAP實作 QoS。

如何在**Trident**中指定精簡配置或厚配置？

ONTAP驅動程式支援精簡配置或厚配置。 ONTAP驅動程式預設採用精簡配置。如果需要厚配置，則應配置後端定義檔或 `StorageClass`。如果兩者都已配置， `StorageClass`優先考慮。為ONTAP配置以下內容：

1. 在 `StorageClass`,設定 `provisioningType`屬性為厚。

2. 在後端定義檔中，透過設定啟用厚卷 `backend spaceReserve parameter`作為體積。

如何確保即使我不小心刪除了**PVC**，正在使用的捲也不會被刪除？

從 Kubernetes 1.10 版本開始，PVC 保護功能會自動啟用。

我可以種植**Trident**生產的**NFS PVC**嗎？

是的。您可以膨脹由Trident製造的 PVC。請注意，卷自動增長是ONTAP 的功能，不適用於Trident。

我可以在**SnapMirror**資料保護 **(DP)** 模式或離線模式下匯入磁碟區嗎？

如果外部磁碟區處於 DP 模式或離線，則磁碟區匯入失敗。您收到以下錯誤訊息：

```
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.
```

如何將資源配額轉換為**NetApp**叢集？

只要NetApp儲存有足夠的容量，Kubernetes 儲存資源配額就應該能夠正常運作。當NetApp儲存因容量不足而無法滿足 Kubernetes 配額設定時， Trident會嘗試進行配置，但會出錯。

我可以使用**Trident**建立磁碟區快照嗎？

是的。Trident支援建立按需磁碟區快照和從快照建立持久性磁碟區。若要從快照建立 PV，請確保 `VolumeSnapshotDataSource`功能門已啟用。

支援**Trident**磁碟區快照的驅動程式有哪些？

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` ， `gcp-cvs` ， 和 `azure-netapp-files`後端驅動程式。

如何對透過**Trident ONTAP**配置的磁碟區進行快照備份？

這可以在以下平台找到： `ontap-nas` ， `ontap-san` ， 和 `ontap-nas-flexgroup`司機。您也可以指定一個 `snapshotPolicy`對於 `ontap-san-economy`FlexVol等級的驅動。

這也可以在以下平台找到： `ontap-nas-economy`驅動程序，但粒度是FlexVol volume級別，而不是 qtree 級別。若要啟用對Trident配置的磁碟區進行快照的功能，請設定後端參數選項 `snapshotPolicy`根據ONTAP後端定義的所需快照策略。 Trident無法獲知儲存控制器拍攝的任何快照。

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，您可以透過Trident設定預留特定百分比的磁碟空間來儲存快照副本。 `snapshotReserve`後端定義檔中的屬性。如果您已配置 `snapshotPolicy`和 `snapshotReserve`在後端定義檔中，快照保留百分比是根據以下方式設定的： `snapshotReserve`後端文件中提到的百分比。如果 `snapshotReserve`沒有提及百分比數值， ONTAP

預設將快照保留百分比設為 5%。如果 `snapshotPolicy`如果選項設為“無”，則快照保留百分比設定為 0。

我可以直接存取卷宗快照目錄並複製文件嗎？

是的，您可以透過設定來存取Trident配置的磁碟區上的快照目錄。 `snapshotDir`後端定義檔中的參數。

我可以透過**Trident**為磁碟區設定**SnapMirror**嗎？

目前， SnapMirror必須透過ONTAP CLI 或OnCommand System Manager在外部進行設定。

如何將持久性磁碟區還原到特定的**ONTAP**快照？

若要將磁碟區還原到ONTAP快照，請執行下列步驟：

1. 使正在使用持久卷的應用程式 pod 靜默。
2. 透過ONTAP CLI 或OnCommand System Manager還原到所需的快照。
3. 重啟應用程式 pod。

Trident能否在配置了負載平衡鏡像的 **SVM** 上設定磁碟區？

可以為透過 NFS 提供資料的 SVM 的根磁碟區建立負載平衡鏡像。 ONTAP會自動更新Trident所建立的磁碟區的負載平衡鏡像。這可能會導致卷安裝延遲。使用Trident建立多個磁碟區時，磁碟區的配置取決於ONTAP更新負載平衡鏡像。

如何將每個客戶/租戶的儲存類別使用情況分開統計？

Kubernetes 不允許在命名空間中使用儲存類別。但是，您可以使用 Kubernetes 透過使用儲存資源配額（每個命名空間一個配額）來限制每個命名空間中特定儲存類別的使用量。若要拒絕特定命名空間對特定儲存的訪問，請將該儲存類別的資源配額設為 0。

故障排除

使用此處提供的提示來解決您在安裝和使用Trident時可能遇到的問題。

如需Trident的協助，請使用下列指令建立支援包 `tridentctl logs -a -n trident`然後將其發送給NetApp支援團隊。

常規故障排除

- 如果Trident艙無法正常升空（例如，當Trident艙卡在...中時 `ContainerCreating`階段（準備容器少於兩個）運行 `kubectl -n trident describe deployment trident`和 `kubectl -n trident describe pod trident-**-**`可以提供更多見解。取得 `kubelet` 日誌（例如，透過 `journalctl -xeu kubelet`）也可能有所幫助。
- 如果Trident日誌中的資訊不足，您可以嘗試透過傳遞參數來啟用Trident的偵錯模式。 `-d`根據您的安裝選項，為安裝參數新增標誌。

  然後確認調試模式已設定 `./tridentctl logs -n trident`並正在尋找 `level=debug msg`在日誌中。

已安裝操作員

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

這將重新啟動所有Trident pod，這可能需要幾秒鐘。您可以透過觀察輸出結果中的“AGE”列來驗證這一點。 `kubectl get pod -n trident` 。

適用於Trident 20.07 和 20.10 版本 `tprov`代替 `torc`。

使用 **Helm** 安裝

```
helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
tridentDebug=true`
```

使用 **tridentctl** 安裝

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

- 您也可以透過新增以下命令來取得每個後端的偵錯日誌。 `debugTraceFlags`在您的後端定義中。例如，包括 `debugTraceFlags: {"api":true, "method":true,}`取得Trident日誌中的 `API` 呼叫和方法遍歷。現有後端可以有 `debugTraceFlags`配置為 `tridentctl backend update`。
- 使用 Red Hat Enterprise Linux CoreOS (RHCOS) 時，請確保： `iscsid`在工作節點上已啟用，並且預設已啟動。這可以透過使用 OpenShift MachineConfigs 或修改 Ignition 模板來實現。
- 使用Trident時可能會遇到的常見問題是... "Azure NetApp Files"指的是租用戶和用戶端金鑰來自權限不足的應用程式註冊。有關Trident要求的完整列表，請參閱"Azure NetApp Files"配置。
- 如果將光電系統安裝到貨櫃上遇到問題，請確保： `rpcbind`已安裝並正在運行。使用主機作業系統所需的軟體包管理器並檢查是否 `rpcbind`正在運行。您可以查看以下狀態： `rpcbind`透過運行服務 `systemctl status rpcbind`或其等效物。
- 如果Trident後端報告它處於 `failed`儘管之前運行正常，但當前狀態可能是由於更改了與後端關聯的 SVM/管理員憑證所致。使用以下方式更新後端訊息 `tridentctl update backend`或者晃動Trident煙囪就能解決這個問題。
- 如果在使用 Docker 作為容器執行時安裝Trident時遇到權限問題，請嘗試使用下列方式安裝Trident： `--in cluster=false`旗幟。這樣就不會使用安裝程式 pod，以避免因以下原因導致的權限問題： `trident-installer`用戶。
- 使用 `uninstall parameter <Uninstalling Trident>`用於清理運行失敗後的殘局。預設情況下，該腳本不會刪除Trident建立的 CRD，因此即使在執行的部署中，卸載和重新安裝也是安全的。
- 如果您想要降級到早期版本的Trident，請先執行以下指令： `tridentctl uninstall`移除Trident的指令。下載所需文件 "Trident版"並使用以下方式安裝 `tridentctl install`命令。
- 安裝成功後，如果PVC管卡在... `Pending`階段，運行 `kubectl describe pvc`可以提供更多關於Trident為何未能為此PVC配置PV的資訊。

使用操作員部署Trident失敗

如果您使用 Operator 部署Trident ，則狀態為： `TridentOrchestrator`變化來自 `Installing`到 `Installed`。如果你觀察 `Failed`如果操作員處於異常狀態且無法自行恢復，則應執行以下命令檢查操作員的日誌：

```
tridentctl logs -l trident-operator
```

追蹤 trident-operator 容器的日誌可以指出問題所在。例如，在與外界隔離的環境中，可能無法從上游鏡像倉庫拉取所需的容器鏡像。

要了解為什麼Trident安裝失敗，您應該查看以下內容： `TridentOrchestrator`地位。

```
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 `TridentOrchestrator`用於安裝Trident 的程式。由於每個 Kubernetes 叢集只能有一個Trident 實例，因此 Operator 會確保在任何給定時間都只有一個活躍的 Trident 實例。 `TridentOrchestrator`它能夠創造。

此外，觀察Trident艙的狀態通常可以顯示是否有異常情況。

```
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
```

您可以清楚地看到，由於一個或多個容器鏡像未獲取，因此 pod 無法完全初始化。

要解決這個問題，你應該編輯 `TridentOrchestrator`CR。或者，您可以刪除 `TridentOrchestrator`並根據修改後的準確定義建立一個新的定義。

使用**Trident**部署失敗 `tridentctl`

為了幫助找出出錯的原因，您可以再次執行安裝程式：`-d`使用此參數將開啟調試模式，幫助您了解問題所在：

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

解決問題後，您可以依照下列步驟清理安裝，然後執行： `tridentctl install`再次執行命令：

```
./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.
```

徹底移除Trident和CRDs

您可以完全移除Trident以及所有建立的 CRD 和相關的自訂資源。

> 此操作無法撤銷。除非你想全新安裝Trident，否則不要這樣做。若要在不刪除 CRD 的情況下卸載Trident ，請參閱"解除安裝Trident"。

Trident操作員

若要解除安裝Trident並使用Trident運算子徹底刪除 CRD，請執行下列操作：

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

舵

使用 Helm 卸載Trident並徹底刪除 CRD：

```
kubectl patch torc trident --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

<code>tridentctl</code>

解除Trident後，若要徹底刪除 CRD，請使用 `tridentctl`

```
tridentctl obliviate crd
```

Kubernetes 1.26 版本中，使用 RWX 原始區塊命名空間時，NVMe 節點卸載失敗

如果您使用的是 Kubernetes 1.26，則在使用 NVMe/TCP 和 RWX 原始區塊命名空間時，節點取消暫存可能會失敗。以下方案提供了應對此故障的變通方法。或者，您可以將 Kubernetes 升級到 1.27 版本。

刪除了命名空間和 **pod**。

設想這樣一個場景：你將一個Trident管理的命名空間（NVMe 持久卷）附加到一個 pod 上。如果直接從ONTAP後端刪除命名空間，則在嘗試刪除 pod 後，取消暫存程序會卡住。此情況不會影響 Kubernetes 叢集或其他功能。

解決方法

從對應的節點卸載持久性磁碟區（與該命名空間對應），並將其刪除。

阻塞資料**LIF**

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.
.解決方法
啟動 dataLIFS 以恢復全部功能。

已刪除命名空間映射

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.
.解決方法
添加 `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以多種方式為Trident提供支援。我們提供全天候 (24x7) 的豐富免費自助支援選項，例如知識庫 (KB) 文章和 Discord 頻道。

Trident支持

Trident根據您的版本提供三個等級的支援。參考"NetApp軟體版本對定義的支持"。

全力支持
: Trident自發布之日起提供十二個月的全面支援。

有限的支持
: Trident在發布日期後的第 13 至 24 個月提供有限的支援。

自給自足
: Trident 的文件可在發布日期後的第 25 個月至第 36 個月內查閱。

版本	全力支持	有限的支持	自給自足
"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"	2025年10月	2026年10月	2027年10月
"24.06"	2025年6月	2026年6月	2027年6月
"24.02"	2025年2月	2026年2月	2027年2月
"23.10"	—	2025年10月	2026年10月
"23.07"	—	2025年7月	2026年7月
"23.04"	—	2025年4月	2026年4月
"23.01"	—	—	2026年1月
"22.10"	—	—	2025年10月

自給自足

如需查看完整的故障排除文章列表，請參閱 "NetApp知識庫（需要登入）"。

社區支持

我們的平台上有一個活躍的容器使用者公共社群（包括Trident開發人員）。"Discord 頻道"。這裡是詢問有關專案的一般性問題並與志同道合的同行討論相關主題的好地方。

NetApp技術支持

如需Trident的協助，請使用下列指令建立支援包 `tridentctl logs -a -n trident`並將其發送至 `NetApp Support <Getting Help>`。

更多資訊

- "Trident資源"
- "Kubernetes Hub"

參考

Trident港口

了解更多關於Trident用於通訊的連接埠的資訊。

Trident港口

Trident使用下列連接埠在 Kubernetes 內部進行通訊：

港口	目的
8443	反向通道 HTTPS
8001	Prometheus 指標端點
8000	Trident REST 伺服器
17546	Trident守護程序集 pod 使用的存活/就緒偵測埠

安裝過程中可以使用以下方法變更活性/就緒探針端口： `--probe-port`旗幟。務必確保工作節點上的其他進程沒有使用該連接埠。

Trident REST API

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

何時使用 REST API

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

為了更好的安全性，Trident `REST API`在 pod 內運作時，預設僅限於 localhost。要改變這種行為，你需要設定 Trident 的 `-address`在其 pod 配置中設定參數。

使用 REST API

若要查看這些 API 的呼叫範例，請傳遞偵錯資訊。(`-d`） 旗幟。更多信息，請參閱"使用 tridentctl 管理 Trident"。

API 的工作原理如下：

得到

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

列出該類型的所有物件。

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

取得指定物件的詳細資訊。

郵政

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

建立指定類型的物件。

- 建立物件需要 JSON 配置。有關每種物件類型的詳細說明，請參閱："使用 tridentctl 管理Trident" 。
- 如果對像已存在，則行為會有所不同：後端會更新現有對象，而所有其他對象類型都會使操作失敗。

刪除

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

刪除指定的資源。

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

命令列選項

Trident為Trident編排器提供了幾個命令列選項。您可以使用這些選項來修改您的部署。

日誌記錄

-debug

啟用調試輸出。

-loglevel <level>

設定日誌等級（debug、info、warn、error、fatal）。預設顯示訊息。

Kubernetes

-k8s_pod

使用此選項或 `-k8s_api_server`啟用 Kubernetes 支援。設定此項目後， Trident將使用其所在 pod 的 Kubernetes 服務帳戶憑證來聯絡 API 伺服器。只有當Trident作為 Pod 在啟用了服務帳戶的 Kubernetes 叢集中運行時，此方法才有效。

-k8s_api_server <insecure-address:insecure-port>

使用此選項或 `-k8s_pod`啟用 Kubernetes 支援。指定後， Trident將使用提供的不安全位址和連接埠連接到 Kubernetes API 伺服器。這使得Trident可以部署在 pod 之外；但是，它僅支援與 API 伺服器的不安全連線。為了安全連接，請將Trident部署在具有以下元件的 pod 中： `-k8s_pod`選項。

Docker

-volume_driver <name>

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

-driver_port <port-number>

監聽此端口，而不是 UNIX 域套接字。

-config <file>

必填項；您必須指定後端設定檔的路徑。

休息

-address <ip-or-host>

指定 Trident REST 伺服器應該監聽的位址。預設為本機。當監聽本機主機並在 Kubernetes pod 內運作時，REST 介面無法從 pod 外部直接存取。使用 `-address ""`使 REST 介面可從 pod IP 位址存取。

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

-port <port-number>

指定 Trident REST 伺服器應監聽的連接埠。預設值為 8000。

-rest

啟用 REST 介面。預設為真。

Kubernetes 和Trident對象

您可以使用 REST API 透過讀取和寫入資源物件與 Kubernetes 和Trident進行互動。有幾個資源物件決定了 Kubernetes 與Trident、 Trident與儲存以及 Kubernetes 與儲存之間的關係。其中一些物件透過 Kubernetes 進行管理，有些物件則透過Trident進行管理。

這些物體之間是如何相互作用的？

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

1. 使用者創建 `PersistentVolumeClaim`請求一個新的 `PersistentVolume`來自 Kubernetes 的特定大小 `StorageClass`這是管理員之前配置好的。
2. Kubernetes `StorageClass`將其配置器標識為Trident ，並包含告訴Trident如何為請求的類別配置磁碟區的參數。
3. Trident審視自身 `StorageClass`名稱相同，用於標識匹配項 `Backends`和 `StoragePools`它可以用來為該類別配置磁碟區。
4. Trident在匹配的後端配置儲存並建立兩個物件：a `PersistentVolume`在 Kubernetes 中，它告訴 Kubernetes 如何尋找、掛載和處理磁碟區；在Trident中，它維護著兩者之間的關係。 `PersistentVolume`以及實際儲存。
5. Kubernetes 綁定了 `PersistentVolumeClaim`新的 `PersistentVolume`。包含下列部件的艙體 `PersistentVolumeClaim`將該持久性磁碟區掛載到它執行的任何主機上。

6. 使用者創建 `VolumeSnapshot`利用現有的PVC管材， `VolumeSnapshotClass`這顯示Trident有問題。
7. Trident識別與 PVC 關聯的磁碟區，並在其後端建立該磁碟區的快照。它也創造了一個 `VolumeSnapshotContent`指示 Kubernetes 如何辨識快照。
8. 使用者可以創建 `PersistentVolumeClaim`使用 `VolumeSnapshot`作為來源。
9. Trident會識別所需的快照，並執行與建立快照相同的步驟。 `PersistentVolume`和 `Volume`。

如需進一步了解 Kubernetes 對象，我們強烈建議您閱讀以下內容： "持久卷" Kubernetes 文件的這一部分。

Kubernetes `PersistentVolumeClaim`物件

Kubernetes `PersistentVolumeClaim` object 是 Kubernetes 叢集使用者發出的儲存請求。

除了標準規格之外， Trident還允許使用者指定以下磁碟區特定的註釋，以便覆寫您在後端設定中設定的預設值：

註解	成交量選擇權	支援的驅動程式
trident.netapp.io/fileSystem	檔案系統	ontap-san、solidfire-san、ontap-san-economy
trident.netapp.io/cloneFromPVC	克隆源磁碟區	ontap-nas、ontap-san、solidfire-san、azure-netapp-files、gcp-cvs、ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas，ontap-san
trident.netapp.io/protocol	協定	任何
trident.netapp.io/exportPolicy	出口政策	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	快照策略	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san
trident.netapp.io/snapshotReserve	快照儲備	ontap-nas、ontap-nas-flexgroup、ontap-san、gcp-cvs
trident.netapp.io/snapshotDirectory	快照目錄	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unix權限	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/blockSize	區塊大小	solidfire-san

如果創建的PV具有 `Delete`根據回收策略，當 PV 被釋放時（即用戶刪除 PVC 時）， Trident會同時刪除 PV 和支援卷。如果刪除操作失敗， Trident會將 PV 標記為失敗，並定期重試該操作，直到操作成功或手動刪除 PV 為止。如果光伏發電使用 `Retain`策略方面， Trident會忽略它，並假定管理員會從 Kubernetes 和後端清理它，從而允許在刪除磁碟區之前對其進行備份或檢查。請注意，刪除 PV 不會導致Trident刪除備份磁碟區。您應該使用 REST API 將其刪除。（`tridentctl`）。

Trident支援使用 CSI 規範建立磁碟區快照：您可以建立磁碟區快照並將其用作資料來源來複製現有的 PVC。這樣，就可以將 PV 的特定時間點副本以快照的形式暴露給 Kubernetes。然後可以使用這些快照建立新的PV。看

看 `On-Demand Volume Snapshots`看看這種方法是否可行。

Trident也提供 `cloneFromPVC`和 `splitOnClone`用於建立克隆的註釋。您可以使用這些註解來克隆 PVC，而無需使用 CSI 實作。

例如：如果使用者已經有一個名為 PVC 的 `mysql`用戶可以建立一個名為「新PVC」的 `mysqlclone`透過使用註釋，例如 `trident.netapp.io/cloneFromPVC: mysql`。有了這組註釋， Trident會克隆與 mysql PVC 對應的捲，而不是從頭開始配置卷。

請考慮以下幾點：

- NetApp建議克隆空閒磁碟區。
- PVC 及其克隆應該位於同一個 Kubernetes 命名空間中，並且具有相同的儲存類別。
- 隨著 `ontap-nas`和 `ontap-san`對於驅動程式來說，設定 PVC 註釋可能是有益的。 `trident.netapp.io/splitOnClone`與 `trident.netapp.io/cloneFromPVC`。和 `trident.netapp.io/splitOnClone`設定為 `true`Trident將克隆卷與父卷分離，從而將克隆卷的生命週期與其父卷完全解耦，但代價是損失了一些儲存效率。未設定 `trident.netapp.io/splitOnClone`或將其設為 `false`這樣做可以減少後端空間佔用，但代價是在父捲和克隆卷之間建立依賴關係，使得除非先刪除克隆卷，否則無法刪除父卷。在克隆空資料庫磁碟區時，拆分克隆是有意義的，因為預計該磁碟區及其克隆磁碟區將有很大的不同，並且無法從ONTAP提供的儲存效率中受益。

這 `sample-input`目錄包含可用於Trident的 PVC 定義範例。請參閱有關Trident音量相關參數和設定的完整說明。

Kubernetes `PersistentVolume`物件

Kubernetes `PersistentVolume`此物件代表一塊可供 Kubernetes 叢集使用的儲存空間。它的生命週期與使用它的艙等無關。

> Trident創造 `PersistentVolume`根據其配置的捲，自動將物件註冊到 Kubernetes 叢集。您無需自行管理它們。

當您建立 PVC 時，指的是基於 Trident 的 `StorageClass`Trident使用對應的儲存類別來設定一個新磁碟區，並為該磁碟區註冊一個新的 PV。在配置已設定磁碟區和對應的 PV 時， Trident遵循以下規則：

- Trident會為 Kubernetes 產生一個 PV 名稱，以及一個用於設定儲存的內部名稱。無論哪種情況，名稱在其範圍內都是獨一無二的，這一點都令人放心。
- 容量大小與 PVC 中要求的容量大小盡可能接近，但可能會向上取整到最接近的可分配數量，具體取決於平台。

Kubernetes `StorageClass`物件

Kubernetes `StorageClass`物件透過名稱指定。 `PersistentVolumeClaims`為儲存配置一組屬性。儲存類別本身標識要使用的配置器，並以配置器能夠理解的方式定義該群組屬性。

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

Kubernetes `StorageClass`使用Trident的物件看起來像這樣：

```
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 特有的，用於告訴Trident如何為該類別配置磁碟區。

儲存類別參數如下：

屬性	類型	必需的	描述
屬性	映射[字串]字串	不	請參閱以下屬性部分。
儲存池	map[string]StringList	不	後端名稱到儲存池清單的映射
附加儲存池	map[string]StringList	不	後端名稱到儲存池清單的映射
排除儲存池	map[string]StringList	不	後端名稱到儲存池清單的映射

儲存屬性及其可能的值可以分為儲存池選擇屬性和 Kubernetes 屬性。

儲存池選擇屬性

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

屬性	類型	價值觀	提供	要求	由...支持
媒體[1]	細繩	機械式硬碟、混合式硬碟、固態硬碟	Pool 包含此類媒體；混合型媒體是指兩者兼具。	指定的媒體類型	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、solidfire-san
供應類型	細繩	薄的，厚的	池支援這種配置方法	指定的配置方法	厚：全部 ontap；薄：全部 ontap 和 solidfire-san

屬性	類型	價值觀	提供	要求	由...支持
後端類型	細繩	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、solidfire-san 、gcp-cvs 、azure-netapp-files、ontap-san-economy	池屬於這種類型的後端	指定的後端	所有司機
快照	布林值	真，假	儲存池支援帶快照的磁碟區	已啟用快照的磁碟區	ontap-nas 、ontap-san 、solidfire-san 、gcp-cvs
複製	布林值	真，假	儲存池支援磁碟區克隆	已啟用克隆的磁碟區	ontap-nas 、ontap-san 、solidfire-san 、gcp-cvs
加密	布林值	真，假	儲存池支援加密磁碟區	已啟用加密的磁碟區	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroups、ontap-san
每秒輸入/輸出次數	整數	正整數	Pool 能夠保證在此範圍內的 IOPS	容量保證了這些IOPS	solidfire-san

[1]： ONTAP Select系統不支援此系統

在大多數情況下，所要求的值會直接影響配置；例如，請求厚配置會導致厚配置磁碟區的產生。但是，Element 儲存池使用其提供的最小和最大 IOPS 來設定 QoS 值，而不是使用請求的值。在這種情況下，請求的值僅用於選擇儲存池。

理想情況下，你可以使用 `attributes`單獨建模，以滿足特定類別的需求所需的儲存品質。 Trident會自動發現並選擇符合所有條件的儲存池 `attributes`您指定的。

如果您發現自己無法使用 `attributes`若要自動為類別選擇合適的池，您可以使用 `storagePools`和 `additionalStoragePools`參數用於進一步細化池子，甚至選擇一組特定的池子。

您可以使用 `storagePools`參數用於進一步限制與任何指定參數相符的池集合。 `attributes` 。換句話說， Trident使用了由以下方式識別的池的交集： `attributes`和 `storagePools`配置參數。您可以單獨使用其中一個參數，也可以同時使用兩個參數。

您可以使用 `additionalStoragePools`此參數用於擴展Trident用於配置的池集，而不管 Trident 選擇的任何池。 `attributes`和 `storagePools`參數。

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

在 `storagePools`和 `additionalStoragePools`參數，每個條目都採用以下形式 `<backend>:<storagePoolList>`，在哪裡 `<storagePoolList>`是指定後端儲存池的逗號分隔清單。例如，一個值 `additionalStoragePools`可能看起來像 `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`。這些清單接受後端值和清單值的正規表示式值。您可以使用 `tridentctl get backend`取得後端及其連線池的清單。

Kubernetes屬性

這些屬性對Trident在動態設定期間選擇儲存池/後端沒有任何影響。相反，這些屬性只是提供 Kubernetes 持久卷支援的參數。工作節點負責檔案系統建立操作，可能需要檔案系統實用程序，例如 xfsprogs。

屬性	類型	價值觀	描述	相關驅動因素	**Kubernetes** 版本
檔案系統類型	細繩	ext4、ext3、xfs	區塊卷的檔案系統類型	solidfire-san、ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy	全部
允許卷擴展	布林值	真，假	啟用或停用對增大PVC尺寸的支持	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy、solidfire-san、gcp-cvs、azure-netapp-files	1.11+
容量綁定模式	細繩	立即，等待第一位消費者	選擇何時進行磁碟區綁定和動態配置	全部	1.19 - 1.26

- 這 `fsType`此參數用於控制 SAN LUN 所需的檔案系統類型。此外，Kubernetes 還利用了以下資訊： `fsType`在儲存類別中表示檔案系統存在。可以透過以下方式控製卷所有權： `fsGroup`僅當 `fsType`已設定。請參閱"Kubernetes：為 Pod 或容器設定安全上下文"有關如何使用設定卷所有權的概述 `fsGroup`情境. Kubernetes 將會應用 `fsGroup`僅當滿足以下條件時才有值：

  - `fsType`設定在儲存類別中。
  - PVC接取方式為RWO。

  對於 NFS 儲存驅動程序，檔案系統已作為 NFS 匯出的一部分存在。為了使用 `fsGroup`儲存類別仍然需要指定一個 `fsType`您可以將其設定為 `nfs`或任何非空值。

- 請參閱"擴大銷量"有關擴容的更多詳情。
- Trident安裝程式包提供了幾個範例儲存類別定義，可供Trident使用。`sample-input/storage-class-*.yaml` 。刪除 Kubernetes 儲存類別會導致對應的Trident儲存類別也被刪除。

Kubernetes `VolumeSnapshotClass`物件

Kubernetes `VolumeSnapshotClass`物體類似於 `StorageClasses`。它們有助於定義多種儲存類別，並被磁碟區快照引用，以將快照與所需的快照類別關聯起來。每個磁碟區快照都與一個磁碟區快照類別相關聯。

一個 `VolumeSnapshotClass`應由管理員定義以建立快照。建立卷宗快照類別時，定義如下：

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

這 `driver`向 Kubernetes 指定請求卷快照的 `csi-snapclass`類別由Trident處理。這 `deletionPolicy`指定必須刪除快照時要執行的操作。什麼時候 `deletionPolicy`設定為 `Delete`當刪除快照時，儲存叢集上的磁碟區快照物件以及底層快照也會被刪除。或者，將其設為 `Retain`意味著 `VolumeSnapshotContent`並保留實體快照。

Kubernetes `VolumeSnapshot`物件

Kubernetes `VolumeSnapshot` object 是建立磁碟區快照的請求。 PVC 代表使用者對磁碟區的請求，磁碟區快照則是使用者對現有 PVC 建立快照的請求。

當收到磁碟區快照請求時， Trident會自動在後端建立磁碟區快照，並透過建立唯一識別碼來公開該快照。 `VolumeSnapshotContent`目的。您可以從現有的 PVC 建立快照，並在建立新的 PVC 時將這些快照用作資料來源。

VolumeSnapshot 的生命週期與來源 PVC 無關：即使來源 PVC 被刪除，快照仍然會存在。刪除具有關聯快照的 PVC 時， Trident會將此 PVC 的後備卷標記為「正在刪除」狀態，但不會完全刪除。當所有關聯的快照都被刪除後，該磁碟區也會被移除。

Kubernetes `VolumeSnapshotContent`物件

Kubernetes `VolumeSnapshotContent`該物件表示從已配置的磁碟區中取得的快照。它類似於`PersistentVolume`表示儲存叢集上已配置的快照。類似 `PersistentVolumeClaim`和 `PersistentVolume`當創建快照時，物件會… `VolumeSnapshotContent`物件與…保持一對一的映射關係 `VolumeSnapshot`該物件請求建立快照。

這 `VolumeSnapshotContent`物件包含唯一標識快照的詳細信息，例如： `snapshotHandle` 。這`snapshotHandle`是PV名稱和名稱的獨特組合 `VolumeSnapshotContent`目的。

當收到快照請求時， Trident會在後端建立快照。快照建立完成後， Trident會配置一個`VolumeSnapshotContent`對象，從而將快照暴露給 Kubernetes API。

> 通常情況下，你不需要管理 `VolumeSnapshotContent`目的。但也有例外情況，例如你想…"匯入磁碟區快照"在Trident之外建立。

Kubernetes `VolumeGroupSnapshotClass`物件

Kubernetes `VolumeGroupSnapshotClass`物體類似於 `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起來。每個磁碟區組快照都與一個磁碟區組快照類別相關聯。

一個 `VolumeGroupSnapshotClass`應由管理員定義以建立快照群組。使用下列定義建立磁碟區組快照類別：

```
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
```

這 `driver`向 Kubernetes 指定請求卷宗組快照的權限。 `csi-group-snap-class`類別由Trident處理。這`deletionPolicy`指定必須刪除群組快照時要執行的操作。什麼時候 `deletionPolicy`設定為 `Delete`當刪除快照時，磁碟區組快照物件以及儲存叢集上的基礎快照也會被刪除。或者，將其設為 `Retain`意味著`VolumeGroupSnapshotContent`並保留物理快照。

Kubernetes `VolumeGroupSnapshot`物件

Kubernetes `VolumeGroupSnapshot`該物件是建立多個磁碟區的快照的請求。就像 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對現有 PVC 建立快照的請求。

當收到磁碟區組快照請求時， Trident會自動在後端管理磁碟區的群組快照創建，並透過建立唯一識別碼來公開該快照。 `VolumeGroupSnapshotContent`目的。您可以從現有的 PVC 建立快照，並在建立新的 PVC 時將這些快照用作資料來源。

VolumeGroupSnapshot 的生命週期與來源 PVC 無關：即使來源 PVC 被刪除，快照仍然會保留。刪除具有關聯快照的 PVC 時， Trident會將此 PVC 的後備卷標記為「正在刪除」狀態，但不會完全刪除。當所有關聯的快照都被刪除時，磁碟區組快照也會被刪除。

Kubernetes `VolumeGroupSnapshotContent`物件

Kubernetes `VolumeGroupSnapshotContent`該物件表示從已配置的磁碟區中取得的群組快照。它類似於`PersistentVolume`表示儲存叢集上已配置的快照。類似 `PersistentVolumeClaim`和 `PersistentVolume`當創建快照時，物件會… `VolumeSnapshotContent`物件與…保持一對一的映射關係 `VolumeSnapshot`該物件請求建立快照。

這 `VolumeGroupSnapshotContent`物件包含用於標識快照組的詳細信息，例如：`volumeGroupSnapshotHandle`以及儲存系統上存在的各個 volumeSnapshotHandles。

當收到快照請求時， Trident會在後端建立磁碟區組快照。卷冊組快照建立完成後， Trident會進行設定。`VolumeGroupSnapshotContent`對象，從而將快照暴露給 Kubernetes API。

Kubernetes `CustomResourceDefinition`物件

Kubernetes 自訂資源是 Kubernetes API 中的端點，由管理員定義，用於對類似物件進行分組。 Kubernetes 支援建立自訂資源來儲存物件集合。您可以透過執行以下命令來取得這些資源定義。 `kubectl get crds` 。

Kubernetes 將自訂資源定義 (CRD) 及其關聯的物件元資料儲存在其元資料儲存中。這樣就無需為Trident單獨開設商店了。

Trident的使用 `CustomResourceDefinition`用於保存Trident物件識別的對象，例如Trident後端、 Trident儲存類別和Trident磁碟區。這些物件由Trident管理。此外，CSI 卷快照框架引入了一些定義卷快照所需的 CRD。

CRD 是 Kubernetes 的一種建構。上述資源的物件由Trident建立。舉個簡單的例子，當使用以下方式建立後端時 `tridentctl`相應的 `tridentbackends`建立 CRD 物件供 Kubernetes 使用。

關於Trident的CRD，需要記住以下幾點：

- 安裝Trident時，會建立一組 CRD，可以像使用任何其他資源類型一樣使用這些 CRD。
- 使用以下方式卸載Trident時 `tridentctl uninstall`使用指令後， Trident pod 會被刪除，但已建立的 CRD 不會被清理。請參閱"解除安裝Trident"了解如何將Trident完全移除並從頭開始重新配置。

Trident `StorageClass`物件

Trident為 Kubernetes 建立相符的儲存類 `StorageClass`指定對象 `csi.trident.netapp.io`在他們的供應領域。儲存類別名稱與 Kubernetes 的名稱相符。 `StorageClass`它所代表的對象。

使用 Kubernetes 時，這些物件會在 Kubernetes 叢集啟動時自動建立。 `StorageClass`已註冊使用Trident作為設定器的設定器。

儲存類別包含對磁碟區的一系列要求。 Trident會將這些要求與每個儲存池中存在的屬性進行匹配；如果匹配，則該儲存池是使用該儲存類別配置磁碟區的有效目標。

您可以使用 REST API 建立儲存類別配置，直接定義儲存類別。但是，對於 Kubernetes 部署，我們期望在註冊新的 Kubernetes 執行個體時建立它們。 `StorageClass`物體。

Trident後端對象

後端代表儲存提供者， Trident在其上配置磁碟區；單一Trident實例可以管理任意數量的後端。

這是您可以自行建立和管理的兩種物件類型之一。另一個是 Kubernetes。 `StorageClass`目的。

有關如何建構這些物件的更多信息，請參閱："配置後端" 。

Trident `StoragePool`物件

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。對於Cloud Volumes Service，這些對應於雲端提供者區域。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，儲存池候選對象始終會自動發現和管理。

Trident `Volume`物件

磁碟區是配置的基本單元，包括後端端點（例如 NFS 共用）以及 iSCSI 和 FC LUN。在 Kubernetes 中，這些直接對應於 `PersistentVolumes`。建立磁碟區時，請確保它具有儲存類別（決定磁碟區的部署位置）和大小。

- 在 Kubernetes 中，這些物件是自動管理的。您可以查看這些信息，以了解Trident 的部署情況。
- 刪除帶有關聯快照的 PV 時，對應的Trident磁碟區將更新為 正在刪除 狀態。若要刪除Trident磁碟區，您應該刪除該磁碟區的快照。

卷配置定義了已配置磁碟區應具有的屬性。

屬性	類型	必需的	描述
版本	細繩	不	Trident API 版本（「1」）
姓名	細繩	是的	要建立的磁碟區的名稱
儲存類別	細繩	是的	配置磁碟區時要使用的儲存類
尺寸	細繩	是的	要配置的磁碟區的大小（以位元組為單位）
協定	細繩	不	使用的協定類型：“檔案”或“區塊”
內部名稱	細繩	不	儲存系統中物件的名稱；由Trident產生
克隆源磁碟區	細繩	不	ontap（nas、san）和solidfire-*：要克隆的捲的名稱
splitOnClone	細繩	不	ontap（nas，san）：將克隆體從其父體中分離出來

屬性	類型	必需的	描述
快照策略	細繩	不	ontap-*：要使用的快照策略
快照儲備	細繩	不	ontap-*：為快照預留的磁碟區百分比
出口政策	細繩	不	ontap-nas*：要使用的匯出策略
快照目錄	布林值	不	ontap-nas*：快照目錄是否可見
unix權限	細繩	不	ontap-nas*：初始 UNIX 權限
區塊大小	細繩	不	solidfire-*：塊/扇區大小
檔案系統	細繩	不	檔案系統類型

Trident生成 `internalName`建立卷時。這包括兩個步驟。首先，它會添加儲存前綴（預設值）。`trident`或後端配置中的前綴）加到磁碟區名稱，從而得到如下形式的名稱 `<prefix>-<volume-name>`。然後它會對名稱進行清理，替換後端不允許的字元。對於ONTAP後端，它會將連字號替換為底線（因此，內部名稱變為 <prefix>_<volume-name>）。對於 Element 後端，它會將下劃線替換為連字符。

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

Trident `Snapshot`物件

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

每個 `Snapshot`物件包含以下屬性：

屬性	類型	必需的	描述
版本	細繩	是的	Trident API 版本（「1」）
姓名	細繩	是的	Trident快照物件的名稱
內部名稱	細繩	是的	儲存系統上Trident快照物件的名稱
卷名	細繩	是的	建立快照的持久磁碟區的名稱
volumeInternalName	細繩	是的	儲存系統上關聯的Trident磁碟區物件的名稱

在 Kubernetes 中，這些物件是自動管理的。您可以查看這些信息，以了解Trident 的部署情況。

當 Kubernetes `VolumeSnapshot`建立物件請求後， `Trident` 的工作原理是在後端儲存系統上建立快照物件。這 `internalName`此快照物件是透過組合前綴產生的。 `snapshot-`和 `UID`的

`VolumeSnapshot`物件（例如， `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`）。`volumeName`和`volumeInternalName`透過取得支援卷的詳細資訊來填入。

Trident `ResourceQuota`目的

Trident守護程式消耗一個`system-node-critical`優先等級——Kubernetes 中可用的最高優先等級——以確保Trident能夠在節點優雅關閉期間識別和清理卷，並允許Trident daemonset pod 在資源壓力高的叢集中搶佔優先順序較低的工作負載。

為了實現這一目標， Trident採用了一種`ResourceQuota`確保Trident守護程式集上的「system-node-critical」優先權類別已滿足。在部署和建立守護程序集之前， Trident會查找`ResourceQuota`對象，如果未發現，則應用它。

如果您需要對預設資源配額和優先類別進行更多控制，您可以產生一個`custom.yaml`或配置`ResourceQuota`使用 Helm Chart 的物件。

以下是一個 `ResourceQuota` 物件優先考慮Trident守護程式集的範例。

```
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
```

有關資源配額的更多信息，請參閱"Kubernetes：資源配額"。

清理`ResourceQuota`如果安裝失敗

在極少數情況下，如果安裝失敗，`ResourceQuota`物件已創建，首次嘗試"解除安裝"然後重新安裝。

如果這樣不行，就手動刪除。`ResourceQuota`目的。

消除 `ResourceQuota`

如果您希望自行控制資源分配，您可以移除Trident。`ResourceQuota`使用以下命令物件：

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

Pod 安全標準 (PSS) 和安全情境約束 (SCC)

Kubernetes Pod 安全標準 (PSS) 和 Pod 安全性原則 (PSP) 定義了權限等級並限制了 Pod 的行為。 OpenShift 安全性情境限制 (SCC) 類似地定義了 OpenShift Kubernetes Engine 特有的 pod 限制。為了實現這種自訂功能， Trident會在安裝過程中啟用某些權限。以下各節詳細介紹了Trident設定的權限。

PSS 取代了 Pod 安全性策略 (PSP)。 PSP 在 Kubernetes v1.21 中已被棄用，並將於 v1.25 中移除。更多信息，請參閱"Kubernetes：安全性"。

必需的 Kubernetes 安全上下文和相關字段

允許	描述
特權	CSI 要求掛載點是雙向的，這表示Trident節點 pod 必須執行特權容器。更多信息，請參閱"Kubernetes：掛載傳播"。
主機網路	iSCSI守護程式需要此元件。 `iscsiadm`管理 iSCSI 掛載點，並使用主機網路與 iSCSI 守護程式通訊。
主機 IPC	NFS 使用進程間通訊 (IPC) 與 NFSD 進行通訊。
主機 PID	開始需要 `rpc-statd`適用於 NFS。 Trident會查詢主機進程以確定是否 `rpc-statd`在掛載 NFS 磁碟區之前運行。

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

賽康普	在特權容器中，Seccomp 設定檔始終為「Unconfined」；因此，它無法在Trident中啟用。
SELinux	在 OpenShift 上，特權容器運行在 `spc_t`（「超級特權容器」）域，而非特權容器則是運行在...域中。`container_t`領域。在 `containerd`， 和 `container-selinux`安裝完成後，所有容器都在運作。`spc_t`域，這實際上禁用了 SELinux。因此， Trident不會增加 `seLinuxOptions`到容器中。
DAC	特權容器必須以 root 使用者身分執行。非特權容器以 root 使用者身分執行，以存取 CSI 所需的 Unix 套接字。

艙體安全標準 (PSS)

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

更改命名空間標籤可能會導致 Pod 無法調度，出現「建立時發生錯誤：...」或「警告：trident-csi-...」等錯誤。如果發生這種情況，請檢查命名空間標籤是否為 `privileged`已更改。如果是這樣，請重新安裝Trident。

Pod 安全策略 (PSP)

場地	描述	預設
`allowPrivilegeEscalation`	特權容器必須允許權限提升。	`true`
`allowedCSIDrivers`	Trident不使用內嵌 CSI 臨時磁碟區。	空的
`allowedCapabilities`	非特權Trident容器不需要比預設集更多的功能，而特權容器將被授予所有可能的功能。	空的
`allowedFlexVolumes`	Trident不使用"FlexVolume驅動器"因此，它們不包含在允許的音量列表中。	空的
`allowedHostPaths`	Trident節點 pod 會掛載節點的根檔案系統，因此設定此清單沒有任何好處。	空的
`allowedProcMountTypes`	Trident不使用任何 `ProcMountTypes`。	空的
`allowedUnsafeSysctls`	Trident不需要任何不安全措施 `sysctls`。	空的
`defaultAddCapabilities`	特權容器無需添加任何功能。	空的
`defaultAllowPrivilegeEscalation`	權限提升權限是在每個Trident pod 中處理的。	`false`
`forbiddenSysctls`	不 `sysctls`允許。	空的
`fsGroup`	Trident容器以root權限運作。	`RunAsAny`
`hostIPC`	掛載 NFS 磁碟區需要主機 IPC 與主機通訊 `nfsd`	`true`
`hostNetwork`	iscsiadm 需要主機網路才能與 iSCSI 守護程式通訊。	`true`
`hostPID`	需要主機 PID 來進行檢查 `rpc-statd`正在節點上運行。	`true`
`hostPorts`	Trident不使用任何主機連接埠。	空的

場地	描述	預設
`privileged`	Trident節點 pod 必須運行特權容器才能掛載磁碟區。	`true`
`readOnlyRootFilesystem`	Trident節點 pod 必須寫入節點檔案系統。	`false`
`requiredDropCapabilities`	Trident節點 pod 運作的是特權容器，無法放棄任何功能。	`none`
`runAsGroup`	Trident容器以root權限運作。	`RunAsAny`
`runAsUser`	Trident容器以root權限運作。	`runAsAny`
`runtimeClass`	Trident不使用 `RuntimeClasses`。	空的
`seLinux`	Trident不設定 `seLinuxOptions`因為目前容器運行時和 Kubernetes 發行版在處理 SELinux 方面有差異。	空的
`supplementalGroups`	Trident容器以root權限運作。	`RunAsAny`
`volumes`	Trident pods 需要這些音量外掛。	`hostPath, projected, emptyDir`

安全上下文約束 (SCC)

標籤	描述	預設
`allowHostDirVolumePlugin`	Trident節點 pod 會掛載節點的根檔案系統。	`true`
`allowHostIPC`	掛載 NFS 磁碟區需要主機 IPC 與主機通訊 `nfsd`。	`true`
`allowHostNetwork`	iscsiadm 需要主機網路才能與 iSCSI 守護程式通訊。	`true`
`allowHostPID`	需要主機 PID 來進行檢查 `rpc-statd`正在節點上運行。	`true`
`allowHostPorts`	Trident不使用任何主機連接埠。	`false`
`allowPrivilegeEscalation`	特權容器必須允許權限提升。	`true`
`allowPrivilegedContainer`	Trident節點 pod 必須運行特權容器才能掛載磁碟區。	`true`
`allowedUnsafeSysctls`	Trident不需要任何不安全措施 `sysctls`。	`none`
`allowedCapabilities`	非特權Trident容器不需要比預設集更多的功能，而特權容器將被授予所有可能的功能。	空的
`defaultAddCapabilities`	特權容器無需添加任何功能。	空的
`fsGroup`	Trident容器以root權限運作。	`RunAsAny`

標籤	描述	預設
`groups`	此 SCC 專為Trident而設，並與其使用者綁定。	空的
`readOnlyRootFilesystem`	Trident節點 pod 必須寫入節點檔案系統。	`false`
`requiredDropCapabilities`	Trident節點 pod 運作的是特權容器，無法放棄任何功能。	`none`
`runAsUser`	Trident容器以root權限運作。	`RunAsAny`
`seLinuxContext`	Trident不設定 `seLinuxOptions`因為目前容器運行時和 Kubernetes 發行版在處理 SELinux 方面有差異。	空的
`seccompProfiles`	特權容器始終以“非限制”模式運作。	空的
`supplementalGroups`	Trident容器以root權限運作。	`RunAsAny`
`users`	提供了一個條目，用於將此 SCC 綁定到Trident命名空間中的Trident用戶。	無
`volumes`	Trident pods 需要這些音量外掛。	`hostPath, downwardAPI, projected, emptyDir`

法律聲明

法律聲明提供對版權聲明、商標、專利等的存取。

版權

"https://www.netapp.com/company/legal/copyright/"

商標

NETAPP、NETAPP 標誌和NetApp商標頁面上列出的標誌是NetApp, Inc. 的商標。其他公司和產品名稱可能是其各自所有者的商標。

"https://www.netapp.com/company/legal/trademarks/"

專利

NetApp擁有的專利的最新清單可在以下位置找到：

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

隱私權政策

"https://www.netapp.com/company/legal/privacy-policy/"

開源

您可以在每個版本的通知文件中查看NetApp Trident軟體中使用的第三方版權和授權資訊。 https://github.com/NetApp/trident/ 。