Trident 25.10 文件

Trident

NetApp
February 25, 2026

目錄

Trident 25.10 文件

版本資訊

新增功能

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。

> 安裝程式 zip 檔案中提供的 Linux `tridentctl` 執行檔是經過測試且受支援的版本。請注意，zip 檔案 `/extras` 部分中提供的 `macos` 執行檔未經測試且不受支援。

25.10 版本新增內容

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

Trident

增強功能

- **Kubernetes**：
  - 除了 ONTAP-SAN（iSCSI 和 FC）驅動程式外，還為 ONTAP-NAS NFS 和 ONTAP-SAN-Economy 驅動程式新增了對 CSI Volume Group Snapshots 的支援，並支援 v1beta1 Volume Group Snapshot Kubernetes API。請參閱"使用磁碟區群組快照"。
  - 新增了 ONTAP-NAS 和 ONTAP-NAS-Economy（兩個 NAS 驅動程式中均不包括 SMB）以及 ONTAP-SAN 和 ONTAP-SAN-Economy 驅動程式的自動工作負載容錯移轉支援，並強制分離磁碟區。請參閱 "使用 Trident 自動化具狀態應用程式的容錯移轉"。
  - 增強 Trident 節點並發性，以提高 FCP 磁碟區節點作業的擴充性。
  - 為 ONTAP NAS 驅動程式新增了 ONTAP AFX 支援。請參閱 "ONTAP NAS 設定選項和範例"。
  - 增加了透過 TridentOrchestrator CR 和 Helm chart 值配置 Trident 容器的 CPU 和記憶體資源請求和限制的支援。（"問題 #1000"、"問題 #927"、"問題 #853"、"問題 #592"、"第 110 期"）。
  - 為 ASAr2 人格新增了 FC 支援。請參閱 "ONTAP SAN 配置選項和範例"。
  - 新增了使用 HTTPS 而非 HTTP 提供 Prometheus 指標的選項。請參閱 "監控 Trident"。
  - 匯入磁碟區時新增了一個選項 `--no-rename`，可以保留磁碟區的原始名稱，但讓 Trident 管理磁碟區的生命週期。參見 "匯入磁碟區"。
  - Trident 部署現在以系統叢集關鍵優先權運作。
- 為 Trident 控制器新增了透過 helm、operator 和 tridentctl（"問題 #858"）使用主機網路的選項。
- 在 Trident 25.10 中為 ANF 驅動程式新增了手動 QoS 支援，使其可用於正式作業；此實驗性增強功能已於 Trident 25.06 中推出。

實驗性增強功能

> 不可用於正式作業環境。

- **[技術預覽]**：新增對 ONTAP-NAS（僅限 NFS）和 ONTAP-SAN（統一 ONTAP 9 的 NVMe）並行的支援，此外還有現有的 ONTAP-SAN 驅動程式技術預覽（統一 ONTAP 9 中的 iSCSI 和 FCP 協定）。

修復

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

Trident Protect

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

增強功能

- 新增了用於控制排程和備份 CR 的 Snapshot CR 逾時的註釋：
  - `protect.trident.netapp.io/snapshot-completion-timeout`
  - `protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout`
  - `protect.trident.netapp.io/volume-snapshots-created-timeout`

    參見 "支援的備份和排程註釋"。
- 在計劃 CR 中新增了一個註釋，用於配置 PVC 綁定逾時，該逾時將由備份 CR 使用：`protect.trident.netapp.io/pvc-bind-timeout-sec`。請參閱 "支援的備份和排程註釋"。
- 改進了 `tridentctl-protect` 備份和快照列表，新增了一個欄位來指示執行掛鉤失敗。

25.06.2 的變更

Trident

修復

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

25.06.1 中的變更

Trident

> 對於正在使用 SolidFire 的客戶，請勿升級到 25.06.1 版本，因為在取消發布磁碟區時存在已知問題。25.06.2 版本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 裝置導致分離磁碟區失敗的問題。
  - 修正了當裝置路徑自建立以來發生變更時、iSCSI 磁碟區取消暫存的問題。
  - 跨儲存類別進行磁碟區區塊複製。
- **OpenShift**：修正了 OCP 4.19 中 iSCSI 節點準備失敗的問題。
- 增加了使用 SolidFire 後端複製磁碟區時的逾時時間（"問題 #1008"）。

25.06 的變更

Trident

增強功能

- **Kubernetes**：
  - 新增對 CSI Volume Group Snapshots 的支援，並提供 `v1beta1` Volume Group Snapshot Kubernetes API 供 ONTAP-SAN iSCSI 驅動程式使用。請參閱"使用磁碟區群組快照"。

    > VolumeGroupSnapshot 是 Kubernetes 中的一項測試版功能，使用測試版 API。Kubernetes 1.32 是 VolumeGroupSnapshot 的最低版本要求。

  - 除了 iSCSI 之外，還新增對 NVMe/TCP 的 ONTAP ASA r2 支援。請參閱 "ONTAP SAN 配置選項和範例"。
  - 為 ONTAP-NAS 和 ONTAP-NAS-Economy 磁碟區新增了安全的 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。請參閱 "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。
  - 在打開 LUKS 磁碟區時新增 `--allow-discards`，以允許執行 discard/TRIM 命令來回收空間。
  - 格式化 LUKS 加密磁碟區時的效能已增強。

- 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 磁碟區掛載和分離的冪等性。
- 為 ONTAP-SAN-Economy 驅動程式的 Trident 磁碟區組態新增了 `internalID` 欄位。
- 新增 SnapMirror 對 NVMe 後端磁碟區複寫的支援。請參閱 "使用 SnapMirror 複製磁碟區"。

實驗性增強功能

> 不可用於正式作業環境。

- 【技術預覽】透過 `--enable-concurrency`功能標誌啟用了並發的 Trident 控制器操作。這使得控制器操作可以並行運行，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。

- [Tech Preview] 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，多路徑裝置的大小可能不一致。
  - 修正了 ONTAP-NAS 和 ONTAP-NAS-Economy 驅動程式無法清理重複匯出原則的問題。
  - 修正了 `nfsMountOptions` 未設定時 GCNV 磁碟區預設使用 NFSv3 的問題；現在同時支援 NFSv3 和 NFSv4 協定。如果 `nfsMountOptions` 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（"問題 #831"）。
  - 修正了從快照建立的 PVC 缺少匯出策略的問題（"問題 #1016"）。
  - 修正了 ANF 磁碟區大小無法自動按 1 GiB 增量對齊的問題。
  - 修正了使用 NFSv3 與 Bottlerocket 時出現的問題。
- 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、複製分割作業同步執行的問題。

棄用：

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

Trident Protect

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

增強功能

- 改善還原時間，提供更頻繁執行完整備份的選項。

- 改善應用程式定義的精細度，並透過 Group-Version-Kind（GVK）篩選進行選擇性還原。
- 使用 AppMirrorRelationship（AMR）搭配 NetApp SnapMirror 時，可實現高效的重新同步和反向複製，以避免完全 PVC 複製。
- 新增了使用 EKS Pod Identity 建立 AppVault 儲存桶的功能，無需為 EKS 叢集的儲存桶憑證指定金鑰。
- 新增在還原命名空間中略過還原標籤和註釋的功能（如有需要）。
- AppMirrorRelationship（AMR）現在將檢查來源 PVC 擴充、並視需要在目的地 PVC 上執行適當的擴充。

修復

- 修正了先前快照的快照註解值被錯誤地套用到新快照的錯誤。現在所有快照註解都能正確套用。
- 預設情況下，如果未定義，則為資料移動器加密（Kopia / Restic）定義一個金鑰。
- 為 S3 appvault 建立新增了改進的驗證和錯誤訊息。
- AppMirrorRelationship （AMR）現在只複製處於 Bound 狀態的 PV，以避免複製失敗。
- 已修正當在具有大量備份的 AppVault 上取得 AppVaultContent 時顯示錯誤的問題。
- KubeVirt VMSnapshots 被排除在還原和容錯移轉作業之外，以避免故障。
- 修正了 Kopia 的一個問題，即由於 Kopia 的預設保留排程覆蓋了使用者在排程中設定的內容，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，邊車鏡像名稱和版本填入不正確的問題（"問題 #983"）。
  - 修正了 ONTAP 故障轉移復原期間多路徑工作階段無法復原的問題（"問題 #961"）。

25.02 的變更

從 Trident 25.02 開始、「新增功能」摘要提供 Trident 和 Trident Protect 版本的增強功能、修正和淘汰詳細資料。

Trident

增強功能

- **Kubernetes**：
  - 新增對 ONTAP ASA r2 for iSCSI 的支援。
  - 新增了 ONTAP-NAS 磁碟區在非正常節點關機情況下強制分離的支援。新的 ONTAP-NAS 磁碟區現在將使用由 Trident 管理的每個磁碟區匯出原則。為現有磁碟區提供了升級路徑，使其在取消發佈時能夠過渡到新的匯出原則模型，而不會影響正在進行的工作負載。
  - 新增了 cloneFromSnapshot 註釋。

- 新增對跨命名空間 Volume 複製的支援。
- 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。
- 新增對 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。
  - 新增 OpenShift Virtualization 對 ONTAP 驅動程式的支援。
- 在 ONTAP-SAN 驅動程式上新增了光纖通道支援。
- 新增 NVMe LUKS 支援。
- 所有基礎映像均已切換為 scratch 映像。
- 新增了 iSCSI 連線狀態發現和記錄功能，當 iSCSI 工作階段應該登入但實際上沒有登入時（"問題 #961"）。
- 新增了對使用 google-cloud-netapp-volumes 驅動程式的 SMB 磁碟區的支援。
- 新增支援功能，允許 ONTAP 磁碟區在刪除時跳過復原佇列。
- 新增使用 SHA 而非標籤覆寫預設映像的支援。
- 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（"問題 #965"）。
  - 修正了 ONTAP-NAS-Economy 的自動匯出原則過早切換到每個磁碟區原則的問題。
  - 修復後端配置憑證以支援所有可用的 AWS ARN 分區（"問題 #913"）。
  - 在 Trident 運算子中新增了停用自動配置器協調的選項（"問題 #924"）。
  - 為 csi-resizer 容器新增 securityContext（"問題 #976"）。

Trident Protect

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

增強功能

- 新增了 KubeVirt / OpenShift Virtualization VM 的備份和還原支援，適用於 volumeMode: File 和 volumeMode: Block （原始裝置）儲存設備。此支援與所有 Trident 驅動程式相容，並增強了使用 NetApp SnapMirror 搭配 Trident Protect 複寫儲存設備時的現有保護功能。
- 為 Kubevirt 環境增加了在應用程式層級控制凍結行為的功能。
- 增加了對配置 AutoSupport 代理連接的支援。
- 增加了為資料移動加密（Kopia / Restic）定義密鑰的功能。
- 新增手動執行 execution hook 的功能。
- 增加了在 Trident Protect 安裝過程中設定安全上下文約束（SCC）的功能。

- 增加了在 Trident Protect 安裝過程中設定 nodeSelector 的支援。
- 新增對 AppVault 物件的 HTTP / HTTPS 輸出 Proxy 支援。
- 擴充 ResourceFilter 功能，可排除叢集範圍的資源。
- 增加了對 S3 AppVault 憑證中 AWS 會話令牌的支援。
- 增加了對快照前執行鉤子之後資源收集的支援。

修復

- 改善臨時磁碟區的管理，以略過 ONTAP 磁碟區還原佇列。
- SCC 註解現已恢復為原始值。
- 透過支援平行作業來提升還原效率。
- 增強對大型應用程式執行掛鉤逾時的支援。

24.10.1 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，當 iSCSI 工作階段應該登入但實際上沒有登入時（"問題 #961"）。

修復

- 修正了自動匯出策略中缺少的節點 IP 位址（"問題 #965"）。
- 修正了 ONTAP-NAS-Economy 的自動匯出原則過早切換到每個磁碟區原則的問題。
- 更新了 Trident 和 Trident-ASUP 相依性，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（"問題 #961"）。

24.10 的變更

增強功能

- Google Cloud NetApp Volumes 驅動程式現已正式推出，適用於 NFS 磁碟區，並支援區域感知資源配置。
- GCP Workload Identity 將用作 Google Cloud NetApp Volumes 與 GKE 的雲端身分。
- 已新增 `formatOptions`組態參數至 ONTAP-SAN 和 ONTAP-SAN-Economy 驅動程式，以允許使用者指定 LUN 格式選項。
- Azure NetApp Files 的最小磁碟區大小已降至 50 GiB。Azure 新的最小磁碟區大小預計將於 11 月正式推出。
- 新增了 `denyNewVolumePools` 配置參數，以將 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 aggregate 的新增、刪除或重新命名的偵測。
- 為 LUKS LUN 增加了 18 MiB 的額外負荷，以確保報告的 PVC 大小可用。

- 改善 ONTAP-SAN 和 ONTAP-SAN-Economy 節點階段和取消階段錯誤處理，以允許在階段失敗後取消階段以移除裝置。
- 新增了自訂角色產生器，可讓客戶在 ONTAP 中為 Trident 建立最小化角色。
- 新增了用於故障排除的額外日誌記錄 `lsscsi`（"問題 #792"）。

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
  - 災難恢復
  - 應用程式行動性

    "深入瞭解 Trident Protect".
- 在安裝程式中新增了一個新標誌 `--k8s-api-qps`，用於設定 Trident 與 Kubernetes API 伺服器通訊時所使用的 QPS 值。
- 已新增 `--node-prep`標誌至安裝程式，用於自動管理 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 儲存協定的相容性。
- 增加了 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 `autoExportPolicy`後端選項時將採用每個 qtree 的匯出原則。為了改善存取控制和安全性，qtree 只會在發佈時對應到節點限制性匯出原則。當 Trident 從所有節點取消發佈磁碟區時，現有 qtree 將切換到新的匯出原則模型，以避免影響作用中的工作負載。
- 新增對 Kubernetes 1.31 的支援。

實驗性增強功能

- 在 ONTAP-SAN 驅動程式上新增光纖通道支援的技術預覽。

修復

- **Kubernetes**：
  - 修正了 Rancher 准入 webhook 防止 Trident Helm 安裝的問題（"問題 #839"）。
  - 固定 Helm Chart 值中的 Affinity 鍵（"問題 #898"）。
  - 已修復 tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector 不會與 "true" 值一起使用（"問題 #899"）。
  - 已刪除克隆過程中建立的臨時快照（"問題 #901"）。
- 新增對 Windows Server 2019 的支援。
- 修復了 Trident 儲存庫中的 `go mod tidy`（"問題 #767"）。

棄用

- **Kubernetes** ：

- 已將支援的最低 Kubernetes 版本更新至 1.25 。
- 已移除對 POD Security Policy 的支援。

產品品牌重塑

從 24.10 版本開始，Astra Trident 更名為 Trident（NetApp Trident）。此次更名不會影響 Trident 的任何功能、支援的平台或互通性。

24.06 的變更

增強功能

- 重要提示： `limitVolumeSize` 參數現在限制 ONTAP 經濟型驅動程式中的 qtree/LUN 大小。請使用新的 `limitVolumePoolSize` 參數控制這些驅動程式中的 Flexvol 大小。（"問題 #341"）。
- 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("問題 #883")。
- 新增對磁碟區複製和調整大小作業的支援，即使後端處於暫停模式也允許執行這些作業。
- 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。
- 在 Trident 中新增支援、針對 ONTAP 9.15.1 及更新版本預設使用 REST 而非 ONTAPI（ZAPI）。
- 在 ONTAP 儲存後端上新增對新持久性磁碟區的自訂磁碟區名稱和中繼資料的支援。
- 增強了 `azure-netapp-files`（ANF）驅動程式，以便在 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。
- 新增對 NFS 磁碟區的 Bottlerocket 支援。
- 新增對 Google Cloud NetApp Volumes 的技術預覽支援。

Kubernetes

- 新增對 Kubernetes 1.30 的支援。
- 新增 Trident DaemonSet 在啟動時清理殭屍掛載和殘留追蹤檔案的功能（"問題 #883"）。
- 新增了 PVC 註解 `trident.netapp.io/luksEncryption`用於動態導入 LUKS 磁碟區（"問題 #849"）。
- 為 ANF driver 新增拓撲感知功能。
- 新增對 Windows Server 2022 節點的支援。

修復

- 修正了因過時交易導致的 Trident 安裝失敗問題。
- 修正了 tridentctl 忽略來自 Kubernetes 的警告訊息（"問題 #892"）。
- 將 Trident 控制器 `SecurityContextConstraint` 優先權變更為 `0`（"問題 #887"）。
- 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 驅動程式的調整大小作業期間 FlexVol 磁碟區縮小。
- 修正了 NFS v4.1 的 ANF Volume 匯入失敗問題。

24.02 的變更

增強功能

- 新增對 Cloud Identity 的支援。
  - AKS 與 ANF - Azure Workload Identity 將用作雲端身分。
  - EKS 與 FSxN - 將使用 AWS IAM 角色作為雲端身分。
- 新增從 EKS 主控台將 Trident 作為附加元件安裝在 EKS 叢集上的支援。
- 增加了配置和停用 iSCSI 自癒功能（"問題 #864"）。
- 為 ONTAP 驅動程式新增了 Amazon FSx 特性，以啟用與 AWS IAM 和 SecretsManager 的整合，並使 Trident 能夠刪除帶有備份的 FSx 磁碟區（"問題 #453"）。

Kubernetes

- 新增對 Kubernetes 1.29 的支援。

修復

- 修正了未啟用 ACP 時出現的 ACP 警告訊息（"問題 #866"）。
- 在 ONTAP 驅動程式中，當複本與快照相關聯時,在快照刪除期間執行複本分割之前增加了 10 秒的延遲。

棄用

- 從多平台映像清單中移除了 in-toto 認證架構。

23.10 的變更

修復

- 如果新請求的大小小於 ontap-nas 和 ontap-nas-flexgroup 儲存驅動程式的總磁碟區大小，則固定磁碟區擴充（"問題 #834"）。
- 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（"問題 #722"）。
- 修正了 FlexVol 針對 ONTAP-NAS-Economy 的名稱轉換問題。
- 修正了在 Windows 節點重新啟動時 Trident 初始化的問題。

增強功能

Kubernetes

新增對 Kubernetes 1.28 的支援。

Trident

- 增加了對使用 Azure Managed Identities （AMI） 和 azure-netapp-files 儲存驅動程式的支援。
- 為 ONTAP-SAN 驅動程式新增透過 TCP 的 NVMe 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（"問題 #558"）。

23.07.1 的變更

*Kubernetes：*修正了守護程式集刪除問題，以支援零停機時間升級（"問題 #740"）。

23.07 的變更

修復

Kubernetes

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

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 ("問題 #816")。
- 修正了使用具有 use-rest=true 的 ONTAP 驅動程式時配額調整的問題。
- 已修正 ontap-san-economy 中的 LUN 複製建立問題。
- 將發布資訊欄位從 `rawDevicePath` 恢復為 `devicePath`；新增了用於填充和恢復（在某些情況下）`devicePath` 欄位的邏輯。

增強功能

Kubernetes

- 新增對匯入預先配置快照的支援。
- 最小化部署和 daemonset Linux 權限（"問題 #817"）。

Trident

- 不再報告「線上」磁碟區和快照的狀態欄位。
- 如果 ONTAP 後端離線，則更新後端狀態（"問題 #801"、"#543"）。
- LUN 序號始終在 ControllerVolumePublish 工作流程中檢索和發布。
- 新增額外的邏輯來驗證 iSCSI 多路徑裝置序號和大小。
- 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。

實驗性增強功能

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

文件

已進行許多組織和格式改進。

棄用

Kubernetes

- 已移除對 v1beta1 快照的支援。
- 移除對 CSI 之前的 Volume 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 的變更

> 只有當 Kubernetes 版本啟用了 Non-Graceful Node Shutdown 功能閘道時，才支援對 ONTAP-SAN-* 磁碟區進行強制磁碟區分離。必須在安裝時使用 `--enable-force-detach` Trident 安裝程式標誌啟用強制分離。

修復

- 修正了 Trident Operator 在規格中指定時使用 IPv6 localhost 進行安裝的問題。
- 修正了 Trident Operator 叢集角色權限，使其與捆綁包權限同步（"問題 #799"）。
- 修正了在 RWX 模式下將原始區塊磁碟區附加到多個節點的問題。
- 修正了 FlexGroup 對 SMB 磁碟區的複製支援和磁碟區匯入問題。
- 修正了 Trident 控制器無法立即關閉的問題（"問題 #811"）。
- 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 相關聯的所有 igroup 名稱。
- 新增了允許外部程序執行完成的修復。
- 修正了 s390 架構的編譯錯誤（"問題 #537"）。
- 修正了磁碟區掛載作業期間不正確的記錄層級（"問題 #781"）。
- 修復了潛在的類型斷言錯誤（"問題 #802"）。

增強功能

- Kubernetes ：
  - 新增對 Kubernetes 1.27 的支援。
  - 新增對匯入 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-* Volume 的支援。
  - 所有 ONTAP-SAN-* 磁碟區現在都將使用基於節點的 igroup。LUN 僅在主動發佈到對應節點時才會對應到 igroup，以提升安全性。當 Trident 認為在不影響現有工作負載的情況下可以安全切換時，現有磁碟區會根據情況切換到新的 igroup 方案（"問題 #758"）。
  - 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。

- 新增對 Amazon FSx SMB 磁碟區的支援至 ontap-nas-economy 和 ontap-nas-flexgroup 儲存驅動程式。
- 新增對使用 ontap-nas、ontap-nas-economy 和 ontap-nas-flexgroup 儲存驅動程式的 SMB 共用的支援。
- 增加了對 arm64 節點的支援（"問題 #732"）。
- 改進了 Trident 關閉程序，首先停用 API 伺服器（"問題 #811"）。
- 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（"問題 #758"）。

23.01.1 的變更

修復

- 修正了 Trident Operator 在規格中指定時使用 IPv6 localhost 進行安裝的問題。
- 修正了 Trident Operator 叢集角色權限，使其與捆綁包權限同步"問題 #799"。
- 新增了允許外部程序執行完成的修復。
- 修正了在 RWX 模式下將原始區塊磁碟區附加到多個節點的問題。
- 修正了 FlexGroup 對 SMB 磁碟區的複製支援和磁碟區匯入問題。

23.01 的變更

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

修復

- Kubernetes：新增了排除 Pod Security Policy 建立的選項，以修復透過 Helm 安裝 Trident 的問題（"問題 #783 、 #794"）。

增強功能

Kubernetes

- 新增對 Kubernetes 1.26 的支援。
- 提高了 Trident RBAC 的整體資源利用率（"問題 #757"）。
- 新增自動化功能，可偵測並修復主機節點上損壞或過時的 iSCSI 工作階段。
- 新增對擴充 LUKS 加密磁碟區的支援。
- Kubernetes：為 LUKS 加密磁碟區新增了認證輪替支援。

Trident

- 在 ontap-nas 儲存驅動程式中新增對 Amazon FSx for NetApp ONTAP 的 SMB 磁碟區的支援。
- 新增在使用 SMB 磁碟區時對 NTFS 權限的支援。
- 為具有 CVS 服務等級的 GCP 磁碟區新增儲存資源池支援。

- 新增支援在使用 ontap-nas-flexgroup 儲存驅動程式建立 FlexGroups 時，可選擇性使用 flexgroupAggregateList。
- 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 部署和 DaemonSet 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 `ontap-san`或 `ontap-san-economy`驅動程式時，不應再指定 DataLIF。

22.10 的變更

升級到 **Trident 22.10** 之前、您必須閱讀下列重要資訊。

關於 Trident 22.10 的關鍵資訊

- Trident 現在支援 Kubernetes 1.25。您必須先將 Trident 升級至 22.10，然後才能升級至 Kubernetes 1.25。
- Trident 現在嚴格強制要求在 SAN 環境中使用多路徑配置，建議在 multipath.conf 檔案中使用 `find_multipaths: no`值。

  使用非多路徑配置或在 multipath.conf 檔案中使用 `find_multipaths: yes`或 `find_multipaths: smart`值會導致掛載失敗。`Trident 自 21.07 版本起就建議使用 `find_multipaths: no`。

修復

- 修正了使用 `credentials`欄位建立的 ONTAP 後端在 22.07.0 升級期間無法上線的特定問題（"問題 #759" ）。
- **Docker**：修正了導致 Docker 磁碟區外掛程式在某些環境（"問題 #548" 和 "問題 #760"）中無法啟動的問題。
- 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 工作流程中的精細重試，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。您必須先將 Trident 升級至 22.10 ，然後才能升級至 Kubernetes 1.25 。
  - 為 Trident Deployment 和 DaemonSet 新增了獨立的 ServiceAccount、ClusterRole 和 ClusterRoleBinding，以便未來能增強權限。
  - 新增對 "跨命名空間磁碟區共享" 的支援。

- 所有 Trident `ontap-*` 儲存驅動程式現在都可與 ONTAP REST API 搭配使用。
- 新增了新的 operator yaml(`bundle_post_1_25.yaml`），沒有 `PodSecurityPolicy`以支援 Kubernetes 1.25。
- 新增 "支援 LUKS 加密磁碟區" 了 `ontap-san` 和 `ontap-san-economy` 儲存驅動程式。
- 新增對 Windows Server 2019 節點的支援。
- 已透過"支援 Windows 節點上的 SMB Volume" storage driver `azure-netapp-files`新增。
- 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 Data Store （ ADS ）驅動程式。
- 已移除在為 iSCSI 配置工作節點多路徑時，對 `yes` 和 `smart` 選項於 `find_multipaths` 的支援。

22.07 的變更

修復

Kubernetes

- 修正了使用 Helm 或 Trident Operator 設定 Trident 時、節點選擇器處理布林值和數值的問題。（"GitHub 問題 #700"）
- 修正了處理非 CHAP 路徑錯誤時出現的問題，現在 kubelet 會在失敗時重試。"GitHub 問題 #736")

增強功能

- 將 CSI 映像的預設登錄從 k8s.gcr.io 過渡到 registry.k8s.io
- 為了提升安全性、ONTAP-SAN 磁碟區現在將使用基於節點的 igroup、並且僅在 LUN 被主動發佈到對應節點時才將其對應到 igroup。當 Trident 認為在不影響目前工作負載的情況下可以安全切換時、現有磁碟區會根據情況切換到新的 igroup 方案。
- 在 Trident 安裝中加入了 ResourceQuota，以確保當預設限制 PriorityClass 使用量時，Trident DaemonSet 能夠被排程。
- 為 Azure NetApp Files 驅動程式新增了對網路功能的支援。("GitHub 問題 #717")
- 已將技術預覽版自動 MetroCluster 切換偵測功能新增至 ONTAP 驅動程式。("GitHub 問題 #228")

棄用

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

移除

- AWS CVS 驅動程式（自 22.04 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探索，以盡力確認 NFS/iSCSI 服務在工作節點上可用。

文件

新增了"Pod 安全標準"（PSS）部分，詳細說明 Trident 在安裝時啟用的權限。

22.04 的變更

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。有關先前的版本、請參閱 "早期版本的文件"。

> 如果您是從先前的 Trident 版本升級而來，並且使用 Azure NetApp Files ，`location` 配置參數現在是必需的單例欄位。

修復

- 改進了 iSCSI 啟動器名稱的剖析。("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 中重複的金鑰宣告。("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磁碟區的問題。("GitHub 問題 #691")
- 增加了對塊設備上文件系統不一致情況的處理。("GitHub 問題 #656")
- 修正了在安裝過程中設定 `imageRegistry` 標誌時無法拉取自動支援鏡像的問題。("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

- 與 Trident 安全端點的入站連線現在至少需要 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 Files Unix 權限功能。
- **Kubernetes**：Trident 守護程序集現在以系統節點關鍵優先權運行。（"GitHub 問題 #694"）

移除

E-Series 驅動程式（自 20.07 版本起已停用）已移除。

22.01.1 的變更

修復

- 修正了在已刪除節點上取消發布磁碟區的問題。("GitHub 問題 #691")
- 修正了在 ONTAP API 回應中存取彙總空間的 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。
- 如果使用方括號指定，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。
- 修復了 Azure NetApp Files 驅動程式，使其允許重複的資源名稱，並將操作限制在單一位置。
- 修復了 Azure NetApp Files 驅動程式，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：
  - 新增對 Kubernetes 1.23 的支援。
  - 新增透過 Trident Operator 或 Helm 安裝 Trident Pod 時的排程選項。（"GitHub 問題 #651"）
- 允許在 GCP 驅動程式中使用跨區域磁碟區。("GitHub 問題 #633")
- 新增對 Azure NetApp Files 磁碟區的「unixPermissions」選項支援。（"GitHub 問題 #666"）

棄用

Trident REST 介面只能在 127.0.0.1 或 [::1] 位址上進行監聽和服務

21.10.1 的變更

> v21.10.0 版本有一個問題，當節點從 Kubernetes 叢集移除後再重新新增時，可能會使 Trident 控制器進入 CrashLoopBackOff 狀態。此問題已在 v21.10.1 版本中修復（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，可能會使 Trident 控制器進入 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 磁碟區的複本無法掛載到與來源磁碟區相同節點上的問題（GitHub 問題 514）。
- 修正了 Trident 關機時記錄致命錯誤的問題（GitHub 問題 597）。

- 與 Kubernetes 相關的修復：
  - 使用 `ontap-nas` 和 `ontap-nas-flexgroup` 驅動程式建立快照時，傳回磁碟區的已使用空間作為最小 restoreSize（GitHub 問題 645）。
  - 修正了磁碟區調整大小後記錄 `Failed to expand filesystem` 錯誤的問題（GitHub 問題 560）。
  - 修正了 pod 可能卡在 `Terminating`狀態的問題（GitHub 問題 572）。
  - 修正了 `ontap-san-economy` FlexVol 可能充滿快照 LUN 的情況（GitHub 問題 533）。
  - 修正了使用不同映像時自訂 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（GitHub 問題 639）的問題。
  - 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 GCP-CVS Performance 磁碟區 `unixPermissions` 選項的支援。
- 為 GCP 中 600 GiB 到 1 TiB 範圍內的規模最佳化 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 配合使用（GitHub 問題 628）。
  - 向 `tridentctl` images 指令新增了操作員影像（GitHub 問題 570）。

實驗性增強功能

- 在 `ontap-san` 驅動程式中新增了對磁碟區複寫的支援。
- 為 `ontap-nas-flexgroup`、 `ontap-san` 和 `ontap-nas-economy` 驅動程式新增了 **tech preview** REST 支援。

已知問題

已知問題是指可能妨礙您成功使用產品的問題。

- 當將安裝了 Trident 的 Kubernetes 叢集從 1.24 升級到 1.25 或更高版本時，必須先更新 values.yaml 將 `excludePodSecurityPolicy`設定為 `true`或新增 `--set excludePodSecurityPolicy=true`至 `helm upgrade`命令中，然後才能升級叢集。
- Trident 現在會對於在其 StorageClass 中未指定 `fsType` 的磁碟區強制執行空白 `fsType` (`fsType=""`。當使用 Kubernetes 1.17 或更新版本時，Trident 支援為 NFS 磁碟區提供空白 `fsType`。對於 iSCSI 磁碟區，當使用 Security Context 強制執行 `fsGroup` 時，您必須在 StorageClass 上設定 `fsType`。
- 當在多個 Trident 實例中使用相同後端時，每個後端設定檔應該為 ONTAP 後端設定不同的 `storagePrefix`值，或為 `SolidFire` 後端使用不同的 `TenantName`。Trident 無法偵測到其他 Trident 實例建立的磁碟區。嘗試在 ONTAP 或 SolidFire 後端上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 `TenantName`配置值相同，則在相同後端上建立的磁碟區可能會出現名稱衝突。
- 安裝 Trident（使用 `tridentctl`或 Trident Operator）以及使用 `tridentctl`管理 Trident 時、應確保已設定 `KUBECONFIG`環境變數。這是指定 `tridentctl`應使用的 Kubernetes 叢集所必需的。使用多個 Kubernetes

環境時、應確保 `KUBECONFIG`檔案已正確載入。

- 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) 實例來說尤其如此，它們需要 `discard` "掛載選項"；確保 discard mountOption 包含在您的[`StorageClass`中，以支援線上區塊丟棄。
- 如果每個 Kubernetes 叢集中存在多個 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磁碟區。如果在叢集中執行多個 Trident 實例，則會導致意外的錯誤行為。每個 Kubernetes 叢集應該只有一個 Trident 實例。
- 如果在 Trident 離線期間從 Kubernetes 中刪除了基於 Trident 的 `StorageClass`物件，Trident 在重新上線後不會從其資料庫中刪除相應的儲存類別。您應該使用 `tridentctl`或 REST API 刪除這些儲存類別。
- 如果使用者在刪除相應的 PVC 之前刪除了 Trident 配置的 PV，Trident 不會自動刪除其支援磁碟區。您應該透過 `tridentctl`或 REST API 刪除該磁碟區。
- 除非每個配置請求的聚合集都是唯一的，否則 ONTAP 一次不能同時配置多個 FlexGroup。
- 使用 Trident 透過 IPv6 時，您應在後端定義中以方括號指定 `managementLIF`和 `dataLIF`。例如，`[fd20:8b1e:b258:2000:f816:3eff:feec:0]`。

> ⓘ 您無法在 ONTAP SAN 後端上指定 `dataLIF`。Trident 會發現所有可用的 iSCSI LIF 並使用它們來建立多路徑工作階段。

- 如果使用 `solidfire-san`驅動程式搭配 OpenShift 4.5，請確保底層工作節點使用 MD5 作為 CHAP 驗證演算法。Element 12.7 提供安全的 FIPS 相容 CHAP 演算法 SHA1、SHA-256 和 SHA3-256。

尋找更多資訊

- "Trident GitHub"
- "Trident 部落格"

早期版本的文件

如果您使用的不是 Trident 25.10 版本，則可以根據 "Trident 支援生命週期" 取得先前版本的文件。

- "Trident 25.06"
- "Trident 25.02"
- "Trident 24.10"
- "Trident 24.06"
- "Trident 24.02"
- "Trident 23.10"
- "Trident 23.07"
- "Trident 23.04"
- "Trident 23.01"

NetApp Trident 對 ONTAP ASA r2 儲存系統的支援

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

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

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

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

支援的 SAN 傳輸協定取決於您的 Trident 版本：

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

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

支援的作業

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

不支援的作業

- LUKS 加密
- SnapMirror Volume 複寫
- 限制 Aggregate 使用量
- 空間保留模式
- 快照
- 分層

如需更多資訊，請參閱 "ONTAP SAN 配置選項和範例"。

已知問題

已知問題會識別可能妨礙您成功使用此產品版本的問題。

以下已知問題會影響目前版本：

還原大型檔案的 **Restic** 備份可能會失敗

從使用 Restic 建立的 Amazon S3 備份中還原 30GB 或更大的檔案時,還原作業可能會失敗。作為因應措施,請使用 Kopia 作為資料移動工具來備份資料（Kopia 是備份的預設資料移動工具）。如需相關說明,請參閱 "使用 Trident Protect 保護應用程式"。

開始使用

了解 Trident

了解 Trident

Trident 是一個由 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 Container Storage Interface（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 儲存平台上使用和管理儲存資源，無論是在公有雲還是在本地，包括本地 ONTAP 叢集（AFF、FAS 和 ASA）、ONTAP Select、Cloud Volumes ONTAP、Element 軟體（NetApp HCI、SolidFire）、Azure NetApp Files 和 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 "Kubernetes" 原生整合。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。詳情請參閱 "Trident 架構"。

Trident 還提供與 Docker 生態系統的直接整合，適用於 NetApp 儲存平台。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。詳情請參閱 "部署適用於 Docker 的 Trident"。

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

Kubernetes 與 NetApp 產品的整合

NetApp 儲存產品組合與 Kubernetes 叢集的許多面向整合，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 提供支援的企業級 Azure 檔案共用服務。您可以在 Azure 中原生運作要求最嚴苛的基於檔案的工作負載，並獲得您期望從 NetApp 獲得的卓越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，可在雲端執行 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佔用空間，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，使基礎架構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配置和管理儲存裝置。

NetApp ONTAP

"NetApp ONTAP" 是 NetApp 多協定、統一的儲存作業系統，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：內部部署 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一個 Node Pod 的形式運作。Node 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 在 Kubernetes 叢集上部署為單一 Trident Controller Pod 和一個或多個 Trident 節點 Pod，並使用標準 Kubernetes *CSI Sidecar Containers* 來簡化 CSI 外掛程式的部署。"Kubernetes CSI Sidecar 容器" 由 Kubernetes Storage 社群維護。

Kubernetes "節點選取器" 和 "容忍度與污點" 用於將 Pod 限制在特定或首選節點上運行。您可以在 Trident 安裝期間為控制器 Pod 和節點 Pod 設定節點選擇器和容錯範圍。

- 控制器外掛程式負責處理磁碟區配置和管理，例如快照和調整大小。
- 節點外掛程式負責將儲存設備連接到節點。

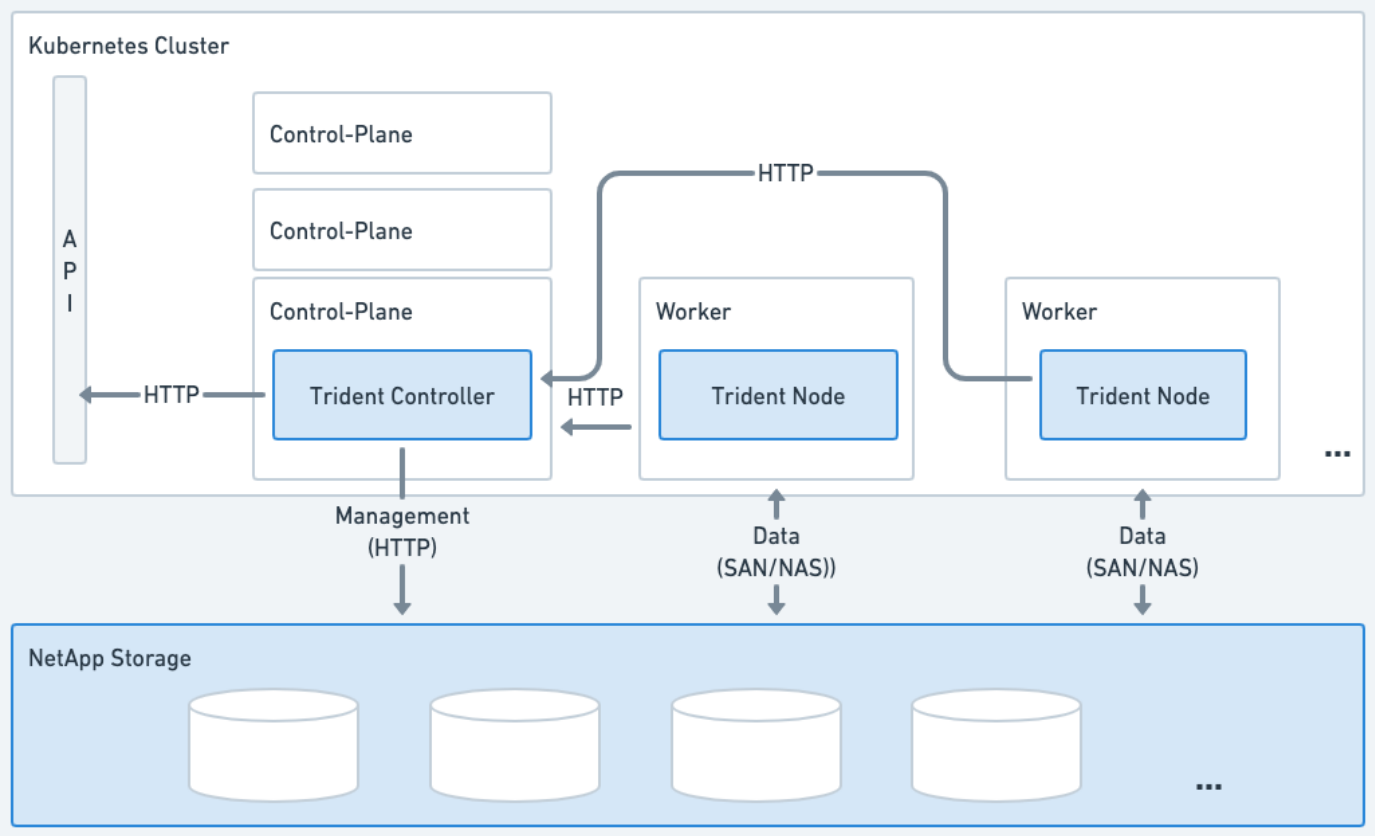


圖 **1. Trident** 已部署在 **Kubernetes** 叢集上

Trident Controller Pod

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

- 負責在 NetApp 儲存設備中配置和管理磁碟區
- 由 Kubernetes Deployment 管理
- 可以在控制平面或工作節點上執行，具體取決於安裝參數。

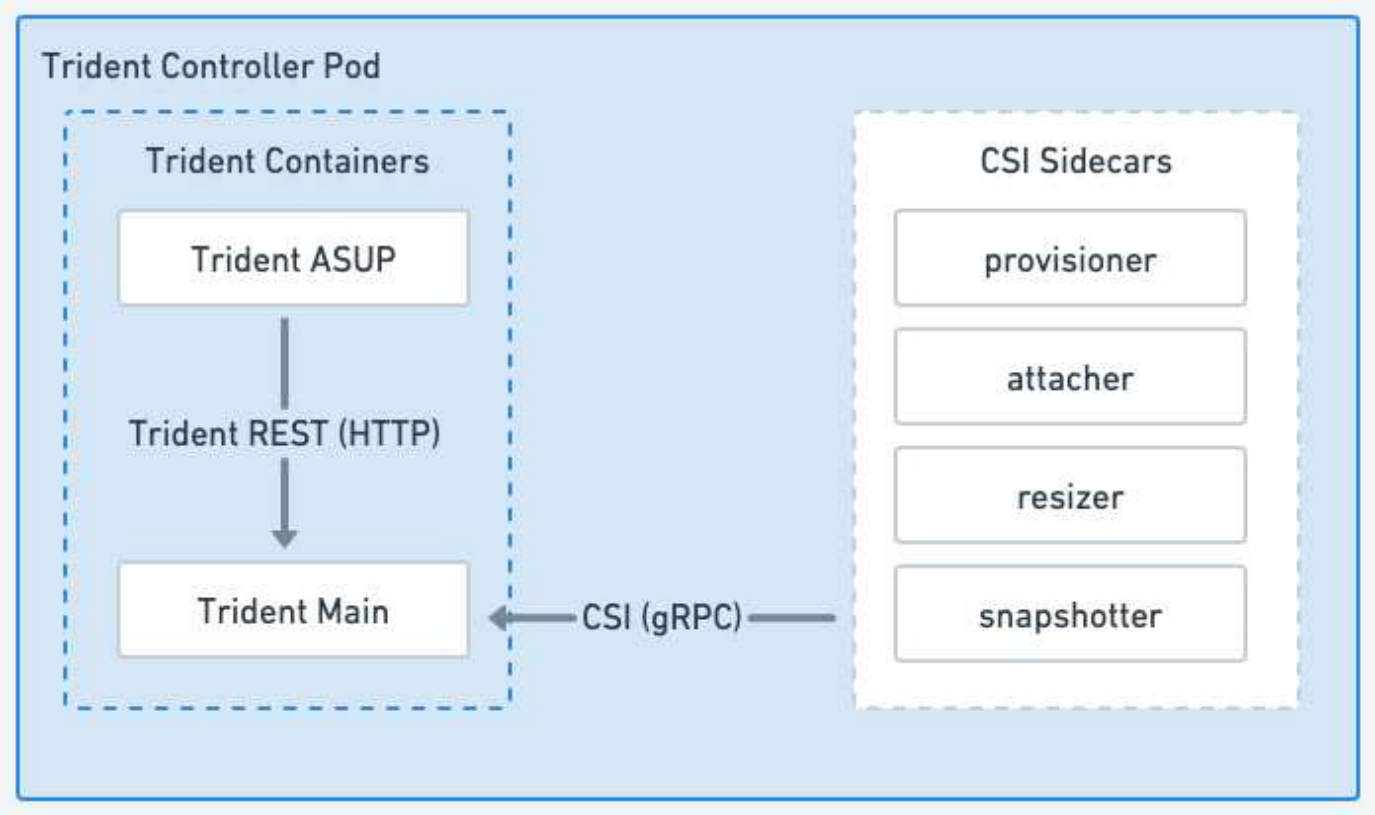


圖 **2. Trident Controller Pod** 示意圖

Trident 節點 Pod

Trident Node Pod 是執行 CSI Node 外掛程式的特權 Pod。

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

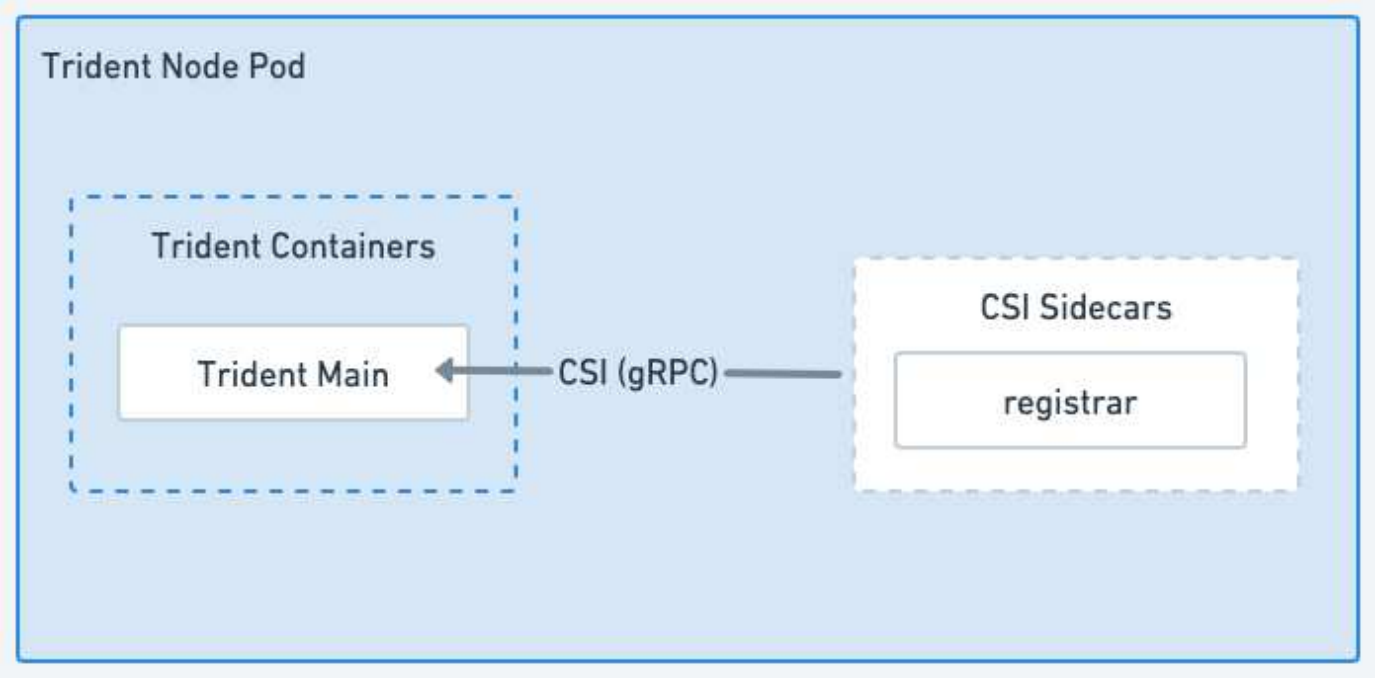


圖 **3. Trident** 節點 **Pod** 示意圖

支援的 **Kubernetes** 叢集架構

Trident 支援下列 Kubernetes 架構：

Kubernetes 叢集架構	支援	預設安裝
單一主節點、運算節點	是的	是的
多主機、運算	是的	是的
主節點、 `etcd`運算	是的	是的
主節點、基礎架構、運算	是的	是的

概念

資源配置

在 Trident 中進行資源配置分為兩個主要階段。第一階段是將儲存類別與一組適當的後端儲存資源池關聯起來，這是資源配置之前必要的準備工作。第二階段包含磁碟區的建立，需要從與待處理磁碟區的儲存類別關聯的儲存資源池中選擇一個。

儲存類別關聯

將後端儲存池與儲存類別關聯起來，取決於儲存類別請求的屬性以及其 `storagePools`、`additionalStoragePools` 和 `excludeStoragePools` 清單。建立儲存類別時、Trident 會將每個後端提供的屬性和儲存池與儲存類別請求的屬性和儲存池進行比較。如果某個儲存池的屬性和名稱與所有要求的屬性和儲存池名稱都匹配、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 也會將

additionalStoragePools 清單中列出的所有儲存池新增至該集合中，即使它們的屬性不符合儲存類別請求的全部或任何屬性。您應該使用 excludeStoragePools 清單來覆寫和移除儲存類別所使用的儲存池。每次新增後端時、Trident 都會執行類似的過程，檢查其儲存池是否符合現有儲存類別的儲存池，並移除任何標記為已排除的儲存池。

Volume 建立

Trident 會利用儲存類別和儲存池之間的關聯來決定磁碟區的配置位置。在建立磁碟區時，Trident 首先會取得該磁碟區所屬儲存類別的儲存池集合，如果您為磁碟區指定了協議，Trident 會移除那些無法提供所需協定的儲存池（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 會將結果集合的順序隨機化，以確保磁碟區的均勻分佈，然後遍歷該集合，依序嘗試在每個儲存池上配置磁碟區。如果在一個儲存池上配置成功，則傳回成功，並記錄過程中遇到的任何失敗。Trident 僅當無法在所有可用於所要求的儲存類別和協定的儲存池上設定磁碟區時才會傳回失敗。

Volume 快照

深入瞭解 Trident 如何為其驅動程式處理磁碟區快照的建立。

了解磁碟區快照建立

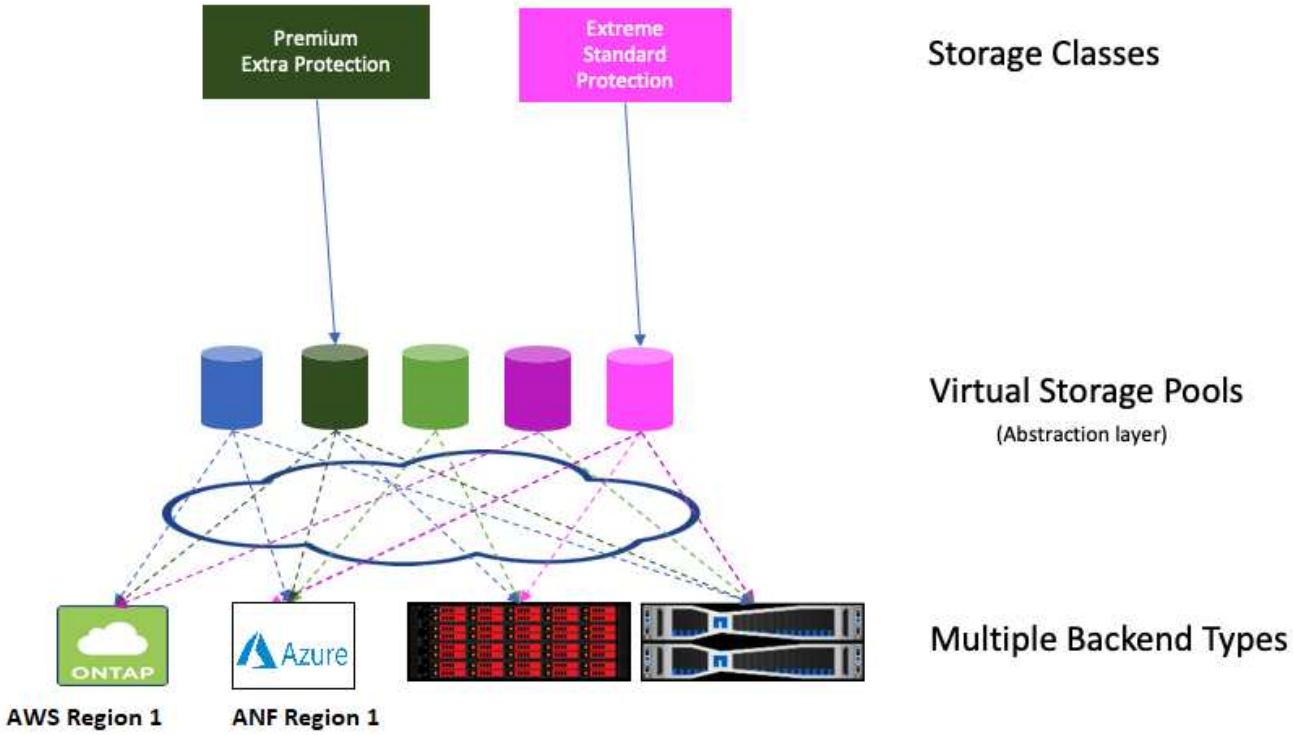
- 對於 ontap-nas、 ontap-san 和 azure-netapp-files 驅動程式、每個持續磁碟區（ PV ）都會對應至 FlexVol 磁碟區。因此、磁碟區快照會建立為 NetApp 快照。NetApp 快照技術比競爭快照技術提供更高的穩定性、擴充性、恢復能力和效能。這些 Snapshot 複本在建立所需的時間和儲存空間方面都非常有效率。
- 對於 ontap-nas-flexgroup 驅動程式而言，每個持久性磁碟區（PV）都會對應到一個 FlexGroup。因此，磁碟區快照以 NetApp FlexGroup 快照的形式建立。NetApp 快照技術相比其他快照技術，具有更高的穩定性、可擴展性、可恢復性和效能。這些快照副本在創建時間和儲存空間方面都非常有效率。
- 對於 ontap-san-economy 驅動程式，PV 會對應到在共用 FlexVol volume 上建立的 LUN，PV 的 VolumeSnapshots 是透過對相關 LUN 執行 FlexClones 來實現的。ONTAP FlexClone 技術可讓您幾乎瞬間建立即使是最大資料集的複本。複本會與其父項共用資料區塊，除了中繼資料所需的儲存空間外，不會佔用其他儲存空間。
- 對於 solidfire-san 驅動程式而言，每個 PV 都會對應到 NetApp Element 軟體/NetApp HCI 叢集上建立的一個 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。
- 使用 `ontap-nas`和 `ontap-san`驅動程式時，ONTAP 快照是 FlexVol 在特定時間點的副本，會佔用 FlexVol 本身的空間。隨著快照的建立/排程，磁碟區中的可寫入空間可能會隨時間減少。一個簡單的解決方法是透過 Kubernetes 調整大小來擴充磁碟區。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 會刪除關聯的 ONTAP 快照。並非透過 Kubernetes 建立的 ONTAP 快照也可以被刪除。

使用 Trident ，您可以使用 VolumeSnapshots 從快照建立新的 PV 。從這些快照建立 PV 是透過使用 FlexClone 技術來支援 ONTAP 後端。從快照建立 PV 時，備份磁碟區是快照父磁碟區的 FlexClone 。 solidfire-san 驅動程式使用 Element 軟體磁碟區複本從快照建立 PV 。在此會從 Element 快照建立複本。

虛擬資源池

虛擬資源池在 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、與後端無關的方式為每個後端定義位置、效能和保護等方面，而無需 `StorageClass`指定要使用哪個實體後端、後端資源池或後端類型來滿足所需標準。

儲存管理員可以在任何 Trident 後端的 JSON 或 YAML 定義檔中定義虛擬資源池。



在虛擬資源池清單之外指定的任何層面對於後端都是全域的，並將套用於所有虛擬資源池，而每個虛擬資源池可能會個別指定一個或多個層面（覆寫任何後端全域層面）。

- 定義虛擬資源池時、請勿嘗試重新排列後端定義中現有虛擬資源池的順序。
- 我們不建議修改現有虛擬資源池的屬性。您應該定義新的虛擬資源池來進行變更。

大多數方面都以後端特定的術語進行指定。關鍵在於，這些方面值不會暴露在後端驅動程式之外，也無法在 `StorageClasses`中進行匹配。相反，管理員會為每個虛擬資源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之間通用。與方面類似，標籤可以按資源池指定，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，管理員可以根據需要完全自主地定義標籤的鍵和值。為了方便起見，儲存管理員可以為每個虛擬資源池定義標籤，並按標籤將磁碟區分組。

可以使用以下字元定義虛擬資源池標籤：

- 大寫字母 `A-Z`
- 小寫字母 `a-z`
- 數字 `0-9`
- 底線 `_`
- 連字號 `-`

A `StorageClass` 透過引用選擇器參數中的標籤來識別要使用的虛擬池。虛擬池選擇器支援以下運算子：

操作員	範例	資源池的標籤值必須：
=	效能＝高級	符合
!=	效能!=極致	不符合
in	位置在（東、西）	屬於價值集合
notin	performance notin（silver、bronze）	不在值集中
<key>	保護	存在且具有任意值
!<key>	！保護	不存在

Volume 存取群組

深入瞭解 Trident 如何使用 "Volume 存取群組"。

> 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，也可以忽略此部分。Trident 在作為增強型 CSI 佈建程式安裝時使用 CHAP。

了解 **Volume Access Group**

Trident 可以使用磁碟區存取群組來控制對其配置的磁碟區的存取。如果 CHAP 被停用，它會尋找名為 `trident` 的存取群組，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建立關聯，但它本身並不會建立或管理存取群組。存取群組必須在將儲存後端新增至 Trident 之前就已存在，並且需要包含 Kubernetes 叢集中所有可能掛載該後端所配置磁碟區的節點的 iSCSI IQN。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 叢集，您應該使用多個存取群組。每個存取群組最多可以包含 64 個 IQN，每個磁碟區可以屬於四個存取群組。配置最多四個存取群組後，規模不超過 256 個節點的叢集中的任何節點都可以存取任何磁碟區。有關磁碟區存取群組的最新限制，請參閱 "這裡"。

如果您要將組態從使用預設 `trident` 存取群組的組態修改為同時使用其他存取群組的組態，請在清單中包含 `trident` 存取群組的 ID。

Trident 快速入門

您可以在幾個步驟內安裝 Trident 並開始管理儲存資源。開始之前，請先檢閱"Trident 需求"。

> 有關 Docker 的資訊，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 配置的磁碟區。

"準備工作節點"

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"

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訊，以及 Trident 應該如何從中配置磁碟區。

"配置後端" 適用於您的儲存系統

建立 **Kubernetes StorageClass**

Kubernetes StorageClass 物件指定 Trident 作為儲存配置器，並允許您建立儲存類別來配置具有可自訂屬性的磁碟區。Trident 會為指定 Trident 儲存配置器的 Kubernetes 物件建立相符的儲存類別。

"建立儲存類別"

配置磁碟區

PersistentVolume (PV) 是叢集管理員在 Kubernetes 叢集上配置的實體儲存資源。*PersistentVolumeClaim* (PVC) 是對叢集上 PersistentVolume 的存取請求。

建立一個 PersistentVolume（PV）和一個 PersistentVolumeClaim（PVC），使用已設定的 Kubernetes StorageClass 來請求存取 PV。然後，您可以將 PV 掛載到 Pod 中。

"配置磁碟區"

接下來呢？

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 Volume 操作。

需求

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

關於 **Trident** 的重要資訊

您必須閱讀以下關於 **Trident** 的重要資訊。

關於 Trident 的重要資訊

- Trident 現在支援 Kubernetes 1.34。請先升級 Trident，再升級 Kubernetes。
- Trident 嚴格強制要求在 SAN 環境中使用多路徑配置，建議在 multipath.conf 檔案中設定 `find_multipaths: no` 值。

  使用非多路徑配置或在 multipath.conf 檔案中使用 `find_multipaths: yes`或 `find_multipaths: smart`值會導致掛載失敗。`Trident 自 21.07 版本起就建議使用 `find_multipaths: no`。

支援的前端（協調器）

Trident 支援多種容器引擎和協調器，包括以下項目：

- Anthos On-Prem (VMware) 和 Anthos on bare metal 1.16
- Kubernetes 1.27 - 1.34
- OpenShift 4.12、4.14 - 4.20（如果您打算在 OpenShift 4.19 版本中使用 iSCSI 節點準備功能，則支援的最低 Trident 版本為 25.06.1。）

  Trident 會繼續支援舊版 OpenShift 版本"Red Hat Extended Update Support (EUS) 版本生命週期"，即使這些舊版本依賴上游已不再官方支援的 Kubernetes 版本。在這種情況下安裝 Trident 時，您可以放心忽略任何關於 Kubernetes 版本的警告訊息。

- Rancher Kubernetes Engine 2（RKE2）v1.28.x - 1.34.x

  雖然 *Trident* 支援 *Rancher Kubernetes Engine 2* （*RKE2*） *1.27.x - 1.34.x* 版本，但 *Trident* 目前僅在 *RKE2 v1.28.5+rke2r1* 上通過認證。

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 "KubeVirt" 的儲存提供者。

在將已安裝 Trident 的 Kubernetes 叢集從 1.25 升級到 1.26 或更高版本之前,請參閱 "升級 Helm 安裝"。

支援的後端 （儲存）

若要使用 Trident、您需要下列一或多個支援的後端：

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

- Google Cloud NetApp Volumes
- NetApp All SAN Array（ASA）
- 本地部署的 FAS、AFF 或 ASA r2（iSCSI、NVMe/TCP 和 FC）運行 ONTAP 版本，這些版本在 NetApp 完全支援或有限支援下運行。請參閱 "軟體版本支援"。
- NetApp HCI/Element 軟體 11 或更高版本

Trident 對 KubeVirt 和 OpenShift Virtualization 的支援

支援的儲存驅動程式：

Trident 支援以下 ONTAP 驅動程式，適用於 KubeVirt 和 OpenShift Virtualization：

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

需要考慮的要點：

- 更新儲存類別，使其在 OpenShift Virtualization 環境中包含 `fsType` 參數（例如： `fsType: "ext4"`）。如有必要，請使用 `volumeMode=Block` 中的 `dataVolumeTemplates` 參數明確將磁碟區模式設定為區塊，以通知 CDI 建立區塊資料磁碟區。
- 區塊儲存驅動程式的 *RWX* 存取模式：ontap-san（iSCSI、NVMe/TCP、FC）和 ontap-san-economy（iSCSI）驅動程式僅支援「volumeMode: Block」（原始裝置）。對於這些驅動程式，`fstype`參數無法使用，因為磁碟區以原始裝置模式提供。
- 對於需要 RWX 存取模式的即時移轉工作流程，支援以下組合：
  - NFS + `volumeMode=Filesystem`
  - iSCSI + `volumeMode=Block`（原始裝置）
  - NVMe/TCP + `volumeMode=Block`（原始裝置）
  - FC + `volumeMode=Block`（原始設備）

功能需求

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

功能	**Kubernetes** 版本	需要功能閘道嗎？
Trident	1.27 - 1.34	否
Volume Snapshot	1.27 - 1.34	否
來自 Volume Snapshot 的 PVC	1.27 - 1.34	否
iSCSI PV 調整大小	1.27 - 1.34	否

功能	**Kubernetes** 版本	需要功能閘道嗎？
ONTAP 雙向 CHAP	1.27 - 1.34	否
動態匯出原則	1.27 - 1.34	否
Trident Operator	1.27 - 1.34	否
CSI 拓撲	1.27 - 1.34	否

測試過的主機作業系統

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

- OpenShift Container Platform 在 AMD64 和 ARM64 架構上支援的 Red Hat Enterprise Linux CoreOS（RHCOS）版本
- Red Hat Enterprise Linux (RHEL) 8 或更新版本（AMD64 和 ARM64）

  NVMe/TCP 需要 RHEL 9 或更新版本。

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

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

`tridentctl` 實用程式也可在上述任何 `Linux` 發行版上執行。

主機組態

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

儲存系統組態

Trident 可能需要對儲存系統進行更改，後端組態才能使用它。

"配置後端"

Trident 連接埠

Trident 需要存取特定連接埠才能進行通訊。

"Trident 連接埠"

容器映像和相應的 **Kubernetes** 版本

對於實體隔離安裝，以下清單是安裝 Trident 所需的容器映像參考。使用 `tridentctl images`命令驗證所需容器映像的清單。

Trident 25.10 所需的容器鏡像

Kubernetes 版本	容器映像
v1.27.0 、 v1.28.0 、 v1.29.0 、 v1.30.0 、 v1.31.0 、 v1.32.0 、 v1.33.0 、 v1.34.0	• docker.io/netapp/trident:25.10.0 • docker.io/netapp/trident-autosupport:25.10 • registry.k8s.io/sig-storage/csi-provisioner:v5.3.0 • registry.k8s.io/sig-storage/csi-attacher:v4.10.0 • registry.k8s.io/sig-storage/csi-resizer:v1.14.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.3.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.15.0 • docker.io/netapp/trident-operator:25.10.0 （選用）

安裝 Trident

使用 Trident 操作員進行安裝

使用 tridentctl 安裝

使用 OpenShift 認證的操作人員進行安裝

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 配置的磁碟區。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

如果您使用多種驅動程式，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）預設已安裝這些工具。

NFS 工具

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

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" 如果您使用 `ontap-san`非揮發性記憶體高速介面（NVMe）over TCP（NVMe/TCP）協定。

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

透過 **FC** 進行 **SCSI** 的工具

如需有關設定 FC 和 FC-NVMe SAN 主機的詳細資訊，請參閱 "配置 FC 和 FC-NVMe SAN 主機的方法"。

"安裝 FC 工具" 如果您使用 `ontap-san` 搭配 sanType `fcp`（透過 FC 連線的 SCSI）。

注意事項：* SCSI over FC 在 OpenShift 和 KubeVirt 環境中受到支援。* SCSI over FC 不支援 Docker。* iSCSI 自癒功能不適用於 SCSI over FC。

SMB 工具

"準備配置 SMB Volume" 如果您正在使用： `ontap-nas` 來配置 SMB Volume 。

節點服務探索

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

節點服務發現功能可以辨識已發現的服務，但並不能保證這些服務配置正確。反之，未發現服務也不代表磁碟區掛載一定會失敗。

檢閱事件

Trident 會為節點建立事件，以識別已發現的服務。若要檢視這些事件，請執行：

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

檢閱探索到的服務

Trident 會辨識 Trident 節點 CR 上每個節點啟用的服務。若要檢視已探索的服務、請執行：

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

NFS 磁碟區

使用適用於您作業系統的命令安裝 NFS 工具。確保 NFS 服務在開機時啟動。

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 磁碟區

Trident 可以自動建立 iSCSI 工作階段、掃描 LUN 、探索多重路徑裝置、格式化裝置並將其掛載到 Pod 。

iSCSI 自癒能力

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

1. *識別*所需的 iSCSI 工作階段狀態和目前的 iSCSI 工作階段狀態。
2. *比較*所需狀態與目前狀態，以識別所需的修復。Trident 會決定修復優先順序以及何時優先進行修復。
3. 執行必要的修復，使目前的 iSCSI 工作階段狀態恢復到所需的 iSCSI 工作階段狀態。

自癒活動的日誌位於對應 Daemonset pod 上的 `trident-main`容器中。若要查看日誌，您必須在 Trident 安裝過程中將 `debug`設為「true」。

Trident iSCSI 自我修復功能可協助防止：

- 網路連線問題後可能會出現過期或不健康的 iSCSI 工作階段。如果工作階段過期，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，如果儲存控制器上的 CHAP 金鑰進行了輪換，而網路連線中斷，則舊的（過時的）CHAP 金鑰可能會繼續存在。自癒機制可以識別這種情況，並自動重新建立工作階段以套用更新後的 CHAP 金鑰。

- 缺少 iSCSI 工作階段
- 缺少 LUN

升級 **Trident** 前需考量的要點

- 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 匯流排上所有裝置的 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 匯流排中的確切 LUN ID。
- 如果同時使用每節點 igroup 和後端範圍 igroup，iSCSI 自我修復將針對 SCSI 匯流排中的確切 LUN ID 啟動 SCSI 重新掃描。

安裝 **iSCSI** 工具

使用適用於您作業系統的命令安裝 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 發行版本，並搭配 `solidfire-san`驅動程式和 Element OS 12.5 或更早版本，請確保在 `/etc/iscsi/iscsid.conf`中將 CHAP 驗證演算法設定為 MD5。Element 12.7 提供了符合 FIPS 標準的安全 CHAP 演算法 SHA1、SHA-256 和 SHA3-256。

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

- 當使用執行 RHEL/Red Hat Enterprise Linux CoreOS（RHCOS）的工作節點並搭配 iSCSI PV 時，請在StorageClass 中指定 `discard` mountOption 以執行即時空間回收。請參閱 "Red Hat 說明文件"。
- 請確保您已升級至最新版本的 `multipath-tools`。

RHEL 8+

1. 安裝下列系統套件：

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

2. 檢查 iscsi-initiator-utils 版本是否為 6.2.0.874-2.el7 或更新版本：

```
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. 啟用多重路徑：

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

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

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. 安裝下列系統套件：

```
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. 啟用多重路徑：

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

> 請確保 `/etc/multipath.conf`包含 `find_multipaths no`在 `defaults`之下。

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 `iscsiadm` 探索目標連接埠，然後再啟動 `open-iscsi`，iSCSI 精靈才能啟動。或者，您也可以修改 `iscsi` 服務，使其自動啟動 `iscsid`。

設定或停用 **iSCSI** 自我修復

您可以設定以下 Trident iSCSI 自我修復設定來修復過時的工作階段：

- **iSCSI** 自癒間隔：決定 iSCSI 自癒功能的呼叫頻率（預設值：5 分鐘）。您可以設定較小的數值來提高呼叫頻率，或設定較大的數值來降低呼叫頻率。

> 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；僅當 iSCSI 自癒功能無法按預期工作或出於偵錯目的時才應停用它。

- **iSCSI** 自癒等待時間：決定 iSCSI 自癒在登出不健康的工作階段並嘗試重新登入之前等待的時間（預設值

：7 分鐘）。您可以將其設定為較大的數字，以便識別為不健康的工作階段必須等待較長時間才能登出，然後嘗試重新登入；或設定為較小的數字，以便更早登出並重新登入。

Helm

若要配置或變更 iSCSI 自癒設置，請在 helm 安裝或 helm 更新期間傳遞 `iscsiSelfHealingInterval`和 `iscsiSelfHealingWaitTime`參數。

以下範例將 iSCSI 自我修復間隔設定為 3 分鐘，自我修復等待時間設定為 6 分鐘：

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

tridentctl

若要配置或變更 iSCSI 自癒設置，請在 tridentctl 安裝或更新期間傳遞 `iscsi-self-healing-interval`和 `iscsi-self-healing-wait-time`參數。

以下範例將 iSCSI 自我修復間隔設定為 3 分鐘，自我修復等待時間設定為 6 分鐘：

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

NVMe/TCP Volume

使用適用於您作業系統的命令安裝 NVMe 工具。

- NVMe 需要 RHEL 9 或更新版本。
- 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 套件，則您可能需要將節點的核心版本更新為包含 NVMe 套件的版本。

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 會修改 `ctrl_device_tmo` 值，以確保 NVMe 在路徑中斷時不會放棄連線。請勿更改此設定。

SCSI over FC 磁碟區

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

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。如需詳細資訊，請參閱 "network interface show"。
2. 取得啟動器（主機）上介面的 WWPN。

   請參閱對應的主機作業系統公用程式。
3. 使用主機和目標的 WWPN 在 FC 交換器上設定分區。

   如需相關資訊，請參閱各交換器廠商文件。

   如需詳細資訊，請參閱下列 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）的工作節點並搭配 FC PVs 時，請在 `discard` StorageClass 中指定 mountOption 以執行即時空間回收。請參閱 "Red Hat 說明文件"。

RHEL 8+

1. 安裝下列系統套件：

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

2. 啟用多重路徑：

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

> 請確保 `/etc/multipath.conf`包含 `find_multipaths no`在 `defaults`之下。

3. 確保 `multipathd` 正在執行：

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

Ubuntu

1. 安裝下列系統套件：

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

2. 啟用多重路徑：

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

> 請確保 `/etc/multipath.conf`包含 `find_multipaths no`在 `defaults`之下。

3. 確保 `multipath-tools` 已啟用並正在運行：

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

準備配置 SMB Volume

您可以使用 `ontap-nas` 驅動程式配置 SMB 磁碟區。

> 您必須在 SVM 上同時設定 NFS 和 SMB/CIFS 通訊協定，才能為 ONTAP 內部部署叢集建立 `ontap-nas-economy` SMB Volume。若未設定其中任一通訊協定，將導致 SMB Volume 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。

- 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 僅支援掛載到在 Windows 節點上執行的 pod 的 SMB 磁碟區。
- 至少需要一個包含您的 Active Directory 憑證的 Trident 金鑰。要產生金鑰 `smbcreds`：

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

- CSI Proxy 設定為 Windows 服務。若要設定 `csi-proxy`，請參閱"GitHub：CSI Proxy"或"GitHub：適用於 Windows 的 CSI Proxy"以瞭解在 Windows 上執行的 Kubernetes 節點。

步驟

1. 對於內部部署 ONTAP、您可以選擇性地建立 SMB 共用區、或 Trident 可以為您建立一個。

   > Amazon FSx for ONTAP 需要 SMB 共用。

   您可以透過兩種方式建立 SMB 管理共用：使用 "Microsoft Management Console" 共用資料夾嵌入式管理單元或使用 ONTAP CLI。若要使用 ONTAP CLI 建立 SMB 共用：

   a. 如有必要、請建立共用區的目錄路徑結構。

      此 `vserver cifs share create`指令會檢查在建立共用時透過 -path 選項指定的路徑。如果指定的路徑不存在，則命令執行失敗。

   b. 建立與指定 SVM 相關聯的 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. 確認共用已建立：

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

詳情請參閱 "建立 SMB 共用區"。

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"。

參數	說明	範例
smbShare	您可以指定以下選項之一：使用 Microsoft Management Console 或 ONTAP CLI 建立的 SMB 共用名稱；允許 Trident 建立 SMB 共用的名稱；或者您可以將此參數留空以封鎖對磁碟區的公共共用存取。對於內部部署 ONTAP，此參數為選用項目。對於 Amazon FSx for ONTAP 後端，此參數為必要項目且不能為空白。	smb-share
nasType	* 必須設為 smb。*如果為 null ，則預設為 nfs。	smb
securityStyle	新磁碟區的安全樣式。對於 **SMB** 磁碟區，必須設定為 **ntfs**`或 `**mixed**。	ntfs 或 mixed 適用於 SMB 磁碟區
unixPermissions	新磁碟區的模式。**SMB** 磁碟區必須保留空白。	""

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訊，以及 Trident 應該如何從中配置磁碟區。

Trident 會自動從後端提供符合儲存類別定義要求的儲存資源池。瞭解如何為您的儲存系統設定後端。

- "配置 Azure NetApp Files 後端"
- "配置 Google Cloud NetApp Volumes 後端"
- "設定 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 搭配使用"

Azure NetApp Files

配置 **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

Services (AKS) 叢集進行憑證管理。

Azure NetApp Files 驅動程式詳細資料

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

驅動程式	傳輸協定	**volumeMode**	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`

考量事項

- Azure NetApp Files 服務不支援小於 50 GiB 的磁碟區。如果請求的磁碟區較小，Trident 會自動建立 50 GiB 的磁碟區。
- Trident 僅支援掛載到在 Windows 節點上執行的 Pod 的 SMB 磁碟區。

AKS 的受管理身分識別

Trident 支援 "託管身分識別" Azure Kubernetes Services 叢集。若要利用託管識別碼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 叢集
- 在 AKS Kubernetes 叢集上設定的託管身分
- 已安裝的 Trident 包括 `cloudProvider`以指定 `"Azure"`。

Trident 操作程式

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

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

Helm

以下範例會使用環境變數 `$CP`將 Trident sets `cloudProvider`安裝到 Azure：

```
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 並將 `cloudProvider` 標誌設為 `Azure`：

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

AKS 的雲端身分

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 認證。

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 叢集
- 在 AKS Kubernetes 叢集上設定 Workload identity 和 oidc-issuer
- 已安裝的 Trident 包括 `cloudProvider`以指定 `"Azure"`和 `cloudIdentity`指定工作負載身分

Trident 操作程式

若要使用 Trident 操作員安裝 Trident，請編輯 `tridentorchestrator_cr.yaml`以將 `cloudProvider`設定為 `"Azure"`，並將 `cloudIdentity`設定為 `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
```

Helm

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'"
```

以下範例會安裝 Trident，並使用環境變數 `cloudProvider`將 `$CP`設為 `Azure`，同時使用環境變數 `cloudIdentity`設定 `$CI`：

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

<code>tridentctl</code>

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

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

以下範例安裝 Trident 並將 `cloud-provider` 標誌設為 `$CP`，並將 `cloud-identity` 設定為 `$CI`：

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

在配置 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

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

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

- 在 AKS 叢集上使用託管識別時，`subscriptionID`、 `tenantID`、 `clientID`、 `location` 和 `clientSecret` 是可選的。
- 在 AKS 叢集上使用雲端身分時，`tenantID`、 `clientID` 和 `clientSecret` 是可選的。

- 容量池。請參閱 "Microsoft：為 Azure NetApp Files 建立容量池"。
- 委派給 Azure NetApp Files 的子網路。請參閱 "Microsoft ：將子網路委派給 Azure NetApp Files"。
- `subscriptionID` 來自已啟用 Azure NetApp Files 的 Azure 訂閱。
- `tenantID`、 `clientID` 和 `clientSecret` 來自 "應用程式註冊" Azure Active Directory 中具有足夠權限存取 Azure NetApp Files 服務的帳戶。應用程式註冊應使用以下任一方式：
  - 擁有者或貢獻者角色"由 Azure 預先定義"。
  - "自訂 Contributor 角色" 在訂閱層級 (`assignableScopes` 建立自訂角色，該角色擁有下列權限，這些權限僅限於 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": []
      }
    ]
  }
}
```

- 包含至少一個 `location`的 Azure "委派子網路"。自 Trident 22.01 起， `location 參數是後端組態檔頂層的必填欄位。在虛擬資源池中指定的位置值將被忽略。
- 若要使用 Cloud Identity，請從 "使用者指派的受控身分"取得 client ID，並在 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`中指定該 ID。

SMB 磁碟區的其他需求

若要建立 SMB Volume ，您必須具備以下條件：

- Active Directory 已設定並連線至 Azure NetApp Files 。請參閱 "Microsoft：建立和管理 Azure NetApp Files 的 Active Directory 連線" 。
- 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 僅支援掛載到在 Windows 節點上執行的 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。要產生金鑰 smbcreds：

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

- CSI Proxy 設定為 Windows 服務。若要設定 csi-proxy，請參閱"GitHub：CSI Proxy"或"GitHub：適用於 Windows 的 CSI Proxy"以瞭解在 Windows 上執行的 Kubernetes 節點。

Azure NetApp Files 後端組態選項和範例

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 磁碟區，並與要求的服務等級和子網路相符。

Azure NetApp Files 後端提供以下設定選項。

參數	說明	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	驅動程式名稱 + "_" + 隨機字元
`subscriptionID`	Azure 訂閱的訂閱 ID（在 AKS 叢集上啟用託管識別時為選用）。	
`tenantID`	在 AKS 叢集上使用託管身分或雲端身分時，來自應用程式註冊的租用戶 ID 為選用項目。	
`clientID`	在 AKS 叢集上使用託管身分或雲端身分時，來自 App Registration 的用戶端 ID 為選用項目。	
`clientSecret`	在 AKS 叢集上使用託管身分或雲端身分時，來自應用程式註冊的用戶端密碼為選用項目。	
`serviceLevel`	其中之一 `Standard`、 `Premium` 或 `Ultra`	"（隨機）
`location`	將在其中建立新磁碟區的 Azure 位置名稱在 AKS 叢集上啟用託管身分識別時為選用。	
`resourceGroups`	用於篩選已發現資源的資源群組清單	"[]"（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集區清單	"[]"（無過濾，隨機）
`virtualNetwork`	具有委派子網路的虛擬網路名稱	""
`subnet`	委派給 `Microsoft.Netapp/volumes` 的子網路名稱	""
`networkFeatures`	磁碟區的 VNet 功能集，可以是 `Basic`或 `Standard`。網路功能並非在所有區域都可用，可能需要在訂閱中啟用。指定 `networkFeatures`時，如果未啟用此功能，會導致磁碟區佈建失敗。	""

參數	說明	預設
`nfsMountOptions`	對 NFS 掛載選項進行精細控制。SMB 卷將忽略此設定。若要使用 NFS 版本 4.1 掛載卷，請在以逗號分隔的掛載選項清單中新增 `nfsvers=4` 以選擇 NFS v4.1。在儲存類別定義中設定的掛載選項會覆蓋後端配置中設定的掛載選項。	"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`	疑難排解時使用的偵錯旗標。例如 `\{"api": false, "method": true, "discovery": true}`。除非您正在進行疑難排解並需要詳細的記錄傾印，否則請勿使用此功能。	null
`nasType`	配置 NFS 或 SMB 磁碟區的建立。選項為 `nfs`、`smb`或 null。設定為 null 則預設建立 NFS 磁碟區。	`nfs`
`supportedTopologies`	表示此後端支援的區域和區域清單。如需詳細資訊，請參閱 "使用 CSI 拓撲"。	
`qosType`	表示 QoS 類型：自動或手動。	自動
`maxThroughput`	設定允許的最大處理量（單位：MiB/ 秒）。僅支援手動 QoS 容量集區。	`4 MiB/sec`

如需網路功能的詳細資訊，請參閱 "設定 Azure NetApp Files 磁碟區的網路功能"。

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缺少所需的權限和資源（子網路、虛擬網路、容量池）。如果啟用了偵錯模式，Trident 會在建立後端時記錄發現的 Azure 資源。請確認是否使用了適當的角色。

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

如果 vNet 位於與 Azure NetApp Files（ANF）儲存帳戶不同的資源群組中，則在設定後端的 resourceGroups 清單時，請為虛擬網路指定資源群組。

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

類型	格式
資源群組	<resource group>
NetApp 帳戶	<resource group>/<netapp account>
容量池	<resource group>/<netapp account>/<capacity pool>
虛擬網路	<resource group>/<virtual network>
子網路	<resource group>/<virtual network>/<subnet>

Volume 資源配置

您可以透過在設定檔特定部分中指定以下選項來控制預設磁碟區配置。詳情請參閱 [範例組態]。

參數	說明	預設
`exportRule`	新磁碟區的匯出規則。 `exportRule` 必須是以逗號分隔的 IPv4 位址或 CIDR 表示法的 IPv4 子網路的任意組合清單。SMB 磁碟區會忽略此規則。	"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）。SMB 磁碟區忽略此設定。	" （預覽功能，需在訂閱中加入白名單）

範例組態

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

最小組態

這是最基本的後端配置。使用此配置， Trident 會發現您所有的 NetApp 帳戶、容量集區和子網路（已委派給位於已設定位置的 Azure NetApp Files ），並隨機將新磁碟區放置在其中一個集區和子網路上。由於 `nasType`省略，因此 `nfs`預設值將套用，後端將為 NFS 磁碟區進行佈建。

如果您剛開始使用 Azure NetApp Files 並進行嘗試，這種設定是理想的，但在實務中，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
```

AKS 的受管理身分識別

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

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    -  resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/eastus-anf-subnet
```

AKS 的雲端身分

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

```
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
```

具有容量資源池篩選器的特定服務等級組態

此後端組態會將磁碟區放置在 Azure 的 `eastus`位置中的 `Ultra`容量集區。Trident 會自動探索該位置中委派給 Azure NetApp Files 的所有子網路，並隨機將新磁碟區放置在其中一個子網路上。

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

後端範例，包含手動 **QoS** 容量集區

此後端配置將磁碟區放置在 Azure 的 `eastus`位置，並具有手動 QoS 容量集區。

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

進階組態

此後端組態進一步將磁碟區放置範圍縮小至單一子網路，同時也會修改部分磁碟區資源配置預設值。

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

虛擬資源池組態

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

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

支援的拓撲組態

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

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

儲存類別定義

以下 `StorageClass`定義是指上述儲存資源池。

使用 **`parameter.selector`**欄位的範例定義

使用 `parameter.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屬性。

```
---
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組態

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

每個命名空間使用不同的機密

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

每個磁碟區使用不同的機密

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

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

建立後端

建立後端組態檔後，執行以下命令：

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

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

```
tridentctl logs
```

在您識別並修正組態檔的問題後、您可以再次執行 create 命令。

Google Cloud NetApp Volumes

配置 **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`驅動程式來與叢集通訊。支援的存取模式包括：*ReadWriteOnce*（RWO）、*ReadOnlyMany*（ROX）、*ReadWriteMany*（RWX）、*ReadWriteOncePod*（RWOP）。

驅動程式	傳輸協定	**volumeMode**	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 的雲端身分

雲端身分可讓 Kubernetes Pod 透過以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 認證。

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身分，並在節點池上配置 GKE MetaData Server。
- 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」並 cloudIdentity 指定新的 GCP 服務帳戶。以下提供範例。

Trident 操作程式

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

例如：

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

Helm

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 並使用環境變數 `$CP`將 `cloudProvider`設為 `GCP`，並使用環境變數 `$ANNOTATION`設置 `cloudIdentity`：

```
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 **cloud provider** 和 **cloud identity** 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 並將 `cloud-provider` 標誌設為 `$CP`，並將 `cloud-identity` 設定為 `$ANNOTATION`：

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

準備設定 **Google Cloud NetApp Volumes** 後端

在配置 Google Cloud NetApp Volumes 後端之前，您需要確保滿足以下要求。

NFS 磁碟區的先決條件

如果您是首次使用 Google Cloud NetApp Volumes 或在新的位置使用，則需要進行一些初始配置來設定 Google Cloud NetApp Volumes 並建立 NFS 磁碟區。請參閱 "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。請參閱 "Google Cloud NetApp Volumes"。
- 您的 Google Cloud 帳戶的專案編號。請參閱 "識別專案"。
- 具有 NetApp Volumes Admin (`roles/netapp.admin` 角色的 Google Cloud 服務帳號。請參閱 "身分與存取管理角色和權限"。
- 您的 GCNV 帳戶的 API 金鑰檔案。請參閱 "建立服務帳戶金鑰"
- 儲存池。請參閱 "儲存池概覽"。

有關如何設定對 Google Cloud NetApp Volumes 存取權限的詳細資訊、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 的後端組態選項並檢閱組態範例。

後端組態選項

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

參數	說明	預設
`version`		始終為 1
`storageDriverName`	儲存驅動程式的名稱	`storageDriverName` 的值必須指定為「google-cloud-netapp-volumes」。
`backendName`	（選用）儲存後端的自訂名稱	驅動程式名稱 "_" API 金鑰的一部分
`storagePools`	用於指定磁碟區建立之儲存池的選用參數。	
`projectNumber`	Google Cloud 帳戶專案編號。該值可在 Google Cloud 入口網站首頁找到。	

參數	說明	預設
location	Trident 建立 GCNV 磁碟區的 Google Cloud 位置。建立跨區域 Kubernetes 叢集時，在 `location`中建立的磁碟區可用於調度到多個 Google Cloud 區域節點上的工作負載。跨區域流量會產生額外費用。	
apiKey	用於具有 netapp.admin`角色的 Google Cloud 服務帳號的 API 金鑰。它包含 Google Cloud 服務帳號私鑰檔案的 JSON 格式內容（原封不動地複製到後端設定檔中）。 `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超過此值，則配置失敗。	"（預設不強制執行）
serviceLevel	儲存池及其磁碟區的服務等級。取值為 flex、standard、premium`或 `extreme。	
labels	要套用於磁碟區的任意 JSON 格式標籤集	""
network	用於 Google Cloud NetApp Volumes 磁碟區的 Google Cloud 網路。	
debugTraceFlags	疑難排解時使用的偵錯旗標。例如 {"api":false, "method":true}。除非您正在進行疑難排解並需要詳細的記錄傾印，否則請勿使用此功能。	null
nasType	配置 NFS 或 SMB 磁碟區的建立。選項為 nfs、`smb`或 null。設定為 null 則預設建立 NFS 磁碟區。	nfs
supportedTopologies	表示此後端支援的區域和可用區列表。如需更多資訊，請參閱"使用 CSI 拓撲"。例如： supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a	

Volume 配置選項

您可以在設定檔的 defaults 區段中控制預設磁碟區配置。

參數	說明	預設
exportRule	新磁碟區的匯出規則。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	存取 .snapshot 目錄	NFSv4 為 "true" ， NFSv3 為 "false"
snapshotReserve	為快照保留的磁碟區百分比	""（接受預設值 0）

參數	說明	預設
`unixPermissions`	新磁碟區的 unix 權限（4 位八進位數字）。	""

範例組態

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

最小組態

這是絕對最小的後端組態。使用此組態，Trident 會在已設定的位置中，發現所有委派給 Google Cloud NetApp Volumes 的儲存資源池，並隨機將新磁碟區放置在其中一個資源池上。由於 `nasType`被省略，`nfs`預設值會套用，後端將會佈建 NFS 磁碟區。

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-----
    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
  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 **StoragePools** 篩選器進行組態設定

```
---
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
```

虛擬資源池組態

此後端組態在單一檔案中定義了多個虛擬資源池。虛擬資源池在 `storage`區段中定義。當您有多個支援不同服務層級的儲存資源池、並想在 Kubernetes 中建立代表這些資源池的儲存類別時、虛擬資源池非常實用。虛擬資源池標籤用於區分資源池。例如、在以下範例中、 `performance`標籤和 `serviceLevel`類型用於區分虛擬資源池。

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。在下列範例中，`snapshotReserve` 和 `exportRule` 做為所有虛擬池的預設值。

如需更多資訊，請參閱 "虛擬資源池"。

```
---
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 的雲端身分

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

支援的拓撲組態

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

接下來呢？

建立後端組態檔後，執行以下命令：

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

若要驗證後端是否已成功建立，請執行下列命令：

```
kubectl get tridentbackendconfig

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

如果後端建立失敗，則表示後端配置存在問題。您可以使用 `kubectl get tridentbackendconfig <backend-name>` 命令描述後端，或執行以下命令查看日誌以確定原因：

```
tridentctl logs
```

在您識別並修正組態檔的問題後、您可以刪除後端並再次執行 create 命令。

儲存類別定義

以下是參照上述後端的基本 `StorageClass`定義。

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

使用 **`parameter.selector`** 欄位的範例定義：

使用 `parameter.selector` 您可以為每個 `StorageClass` 指定用於託管磁碟區的 "虛擬資源池"。此磁碟區將具有所選儲存池中定義的設定項。

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

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

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

如需儲存類別的詳細資訊，請參閱 "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。任何具有任意權限或無權限的 Active Directory 使用者名稱/密碼均可用作節點階段金鑰。

預設命名空間上的基本組態

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

每個命名空間使用不同的機密

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

每個磁碟區使用不同的機密

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

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

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 是否已繫結，請執行下列命令：

```
kubectl get pvc gcnv-nfs-pvc

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

設定 NetApp HCI 或 SolidFire 後端

了解如何在 Trident 安裝中建立和使用 Element 後端。

Element 驅動程式詳細資料

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

`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 會建立磁碟區並建立檔案系統。檔案系統類型由 StorageClass 指定。

驅動程式	傳輸協定	VolumeMode	支援的存取模式	支援的檔案系統
solidfire-san	iSCSI	區塊	RWO 、 ROX 、 RWX 、 RWOP	無檔案系統。原始區塊裝置。

驅動程式	傳輸協定	VolumeMode	支援的存取模式	支援的檔案系統
`solidfire-san`	iSCSI	檔案系統	RWO、RWOP	`xfs`, `ext3`, `ext4`

開始之前

在建立 Element 後端之前、您需要下列項目。

- 執行 Element 軟體的受支援儲存系統。
- NetApp HCI/SolidFire 叢集管理員或租用戶使用者的憑證，可管理磁碟區。
- 所有 Kubernetes 工作節點都應安裝對應的 iSCSI 工具。請參閱"工作節點準備資訊"。

後端組態選項

請參閱下表以了解後端組態選項：

參數	說明	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_" + 儲存設備（iSCSI）IP 位址
`Endpoint`	具有租戶認證資料的 SolidFire 叢集的 MVIP	
`SVIP`	儲存設備（iSCSI）IP 位址和連接埠	
`labels`	要套用於磁碟區的任意 JSON 格式標籤集。	""
`TenantName`	要使用的租戶名稱（如果找不到則建立）	
`InitiatorIFace`	將 iSCSI 流量限制到特定主機介面	"default"
`UseCHAP`	使用 CHAP 協定對 iSCSI 進行身份驗證。Trident 就使用 CHAP 協定。	true
`AccessGroups`	要使用的存取群組 ID 清單	尋找名為「trident」的存取群組 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`	疑難排解時使用的偵錯旗標。例如、{"api":false, "method":true}	null

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

範例 **1**： `solidfire-san` 驅動程式具有三種磁碟區類型的後端組態

此範例展示了一個使用 CHAP 驗證的後端文件，並對三種具有特定 QoS 保證的磁碟區類型進行了建模。您很可能需要使用 `IOPS`儲存類別參數來定義儲存類別，以便使用每種儲存類別。

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

範例 **2**： `solidfire-san` 驅動程式搭配虛擬資源池的後端和儲存類別組態

此範例顯示了配置了虛擬池的後端定義檔以及參考這些虛擬池的 StorageClasses。

Trident 會在配置時將儲存資源池上的標籤複製到後端儲存 LUN。為了方便起見，儲存管理員可以為每個虛擬資源池定義標籤，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，這些值將 `type`設定為 Silver。虛擬池在 `storage`區段中定義。在此範例中，部分儲存池設定了自己的類型，而部分儲存池則覆寫了上述預設值。

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
```

```
SVIP: <svip>:3260
TenantName: <tenant>
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
  - labels:
      performance: gold
      cost: "4"
    zone: us-east-1a
    type: Gold
  - labels:
      performance: silver
      cost: "3"
    zone: us-east-1b
    type: Silver
  - labels:
      performance: bronze
      cost: "2"
    zone: us-east-1c
    type: Bronze
  - labels:
      performance: silver
      cost: "1"
    zone: us-east-1d
```

以下 StorageClass 定義均與上述虛擬池相關。使用 `parameters.selector` 欄位，每個 StorageClass 都會指

定可用於託管磁碟區的虛擬池。磁碟區將具有所選虛擬池中定義的屬性。

第一個 StorageClass (`solidfire-gold-four` 將對應到第一個虛擬儲存池。這是唯一提供黃金級效能且 `Volume Type QoS` 為 Gold 的儲存池。最後一個 StorageClass (`solidfire-silver` 會指定任何提供白銀級效能的儲存池。Trident 將決定選擇哪個虛擬儲存池，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
```

尋找更多資訊

- "Volume 存取群組"

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 驅動程式詳細資料

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

驅動程式	傳輸協定	**volumeMode**	支援的存取模式	支援的檔案系統
`ontap-san`	iSCSI SCSI over FC	區塊	RWO 、 ROX 、 RWX 、 RWOP	無檔案系統；原始區塊裝置
`ontap-san`	iSCSI SCSI over FC	檔案系統	RWO、RWOP ROX 和 RWX 在 Filesystem 磁碟區模式下不可用。	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP 請參閱 NVMe/TCP 的其他考量事項。	區塊	RWO 、 ROX 、 RWX 、 RWOP	無檔案系統；原始區塊裝置
`ontap-san`	NVMe/TCP 請參閱 NVMe/TCP 的其他考量事項。	檔案系統	RWO、RWOP ROX 和 RWX 在 Filesystem 磁碟區模式下不可用。	`xfs`, `ext3`, `ext4`

驅動程式	傳輸協定	volumeMode	支援的存取模式	支援的檔案系統
ontap-san-economy	iSCSI	區塊	RWO 、 ROX 、 RWX 、 RWOP	無檔案系統；原始區塊裝置
ontap-san-economy	iSCSI	檔案系統	RWO、RWOP ROX 和 RWX 在 Filesystem 磁碟區模式下不可用。	xfs, ext3, ext4

- 僅當預期持久性磁碟區使用次數高於"支援的 ONTAP Volume 限制"時，才使用 ontap-san-economy。
- 僅當預期持久性磁碟區使用次數高於"支援的 ONTAP Volume 限制" 且無法使用 ontap-san-economy 驅動程式時，才使用 ontap-nas-economy。
- 如果您預計需要資料保護、災難復原或行動性，請勿使用 ontap-nas-economy。
- NetApp 除 ontap-san 外，不建議在所有 ONTAP 驅動程式中使用 Flexvol 自動增長功能。作為變通方案，Trident 支援使用快照預留，並相應地擴展 Flexvol 磁碟區。

使用者權限

Trident 需要以 ONTAP 或 SVM 管理員身分執行，通常使用 `admin`叢集使用者或 `vsadmin`SVM 使用者，或使用具有相同角色但名稱不同的使用者。對於 Amazon FSx for NetApp ONTAP 部署、Trident 需要以 ONTAP 或 SVM 管理員身分執行，使用叢集 `fsxadmin`使用者或 `vsadmin`SVM 使用者，或使用具有相同角色但名稱不同的使用者。 `fsxadmin`使用者是叢集管理使用者的有限替代方案。

如果使用 `limitAggregateUsage`參數，則需要叢集管理員權限。將 Amazon FSx for NetApp ONTAP 與 Trident 搭配使用時， `limitAggregateUsage`參數與 `vsadmin`和 `fsxadmin`使用者帳戶不相容。如果指定此參數，組態作業將失敗。

雖然可以在 ONTAP 中建立更嚴格的角色供 Trident 驅動程式使用，但我們不建議這樣做。大多數新版本的 Trident 都會呼叫額外的 API，這些 API 需要考慮，這會讓升級變得困難且容易出錯。

NVMe/TCP 的其他考量事項

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

- IPv6
- NVMe 磁碟區的快照和複本
- 調整 NVMe Volume 的大小
- 導入在 Trident 外部建立的 NVMe 磁碟區，以便 Trident 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
- 裝置對應程式（DM）多重路徑
- LUKS 加密

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

準備使用 **ONTAP SAN** 驅動程式配置後端

了解使用 ONTAP SAN 驅動程式配置 ONTAP 後端的要求和驗證選項。

需求

對於所有 ONTAP 後端、 Trident 要求至少將一個 Aggregate 指派給 SVM 。

"ASA r2 系統" 與其他 ONTAP 系統（ASA、AFF 和 FAS）在儲存層的實作方式上有所不同。在 ASA r2 系統中，使用儲存可用區而非 Aggregate。請參閱 "這" 知識庫文章，瞭解如何在 ASA r2 系統中將 Aggregate 指派給 SVM。

請記住，您還可以執行多個驅動程式，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`類別，使用 `ontap-san`驅動程式，以及一個 `san-default`類別，使用 `ontap-san-economy`驅動程式。

所有 Kubernetes 工作節點都必須安裝適當的 iSCSI 工具。詳情請參閱 "準備工作節點"。

驗證 **ONTAP** 後端

Trident 提供兩種 ONTAP 後端驗證模式。

- 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 `vsadmin`，以確保與 ONTAP 版本的最大相容性。
- 基於憑證：Trident 也可以使用安裝在後端的憑證與 ONTAP 叢集通訊。在這種情況下，後端定義必須包含客戶端憑證、金鑰以及受信任 CA 憑證（如果使用，建議）的 Base64 編碼值。

您可以更新現有後端，以在基於認證和基於憑證的方法之間移動。但是，一次只支援一種驗證方法。若要切換到不同的驗證方法，您必須從後端組態中移除現有方法。

如果您嘗試同時提供憑證和憑證，則後端建立將會失敗，並出現錯誤，提示組態檔中提供了多個驗證方法。

啟用基於認證的驗證

Trident 需要 SVM 範圍 / 叢集範圍的管理員憑證才能與 ONTAP 後端通訊。建議使用標準預先定義的角色，例如 `admin`或 `vsadmin`。這可確保與未來 ONTAP 版本向前相容,因為未來版本可能會公開供 Trident 版本使用的功能 API。雖然可以建立自訂安全登入角色並將其與 Trident 搭配使用,但不建議這樣做。

後端定義範例如下所示：

YAML

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

JSON

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

請注意，後端定義是唯一以純文字形式儲存認證資料的地方。建立後端之後，使用者名稱 / 密碼會使用 Base64 編碼，並儲存為 Kubernetes 機密。建立或更新後端是唯一需要瞭解認證資料的步驟。因此，這是僅限管理員執行的作業，由 Kubernetes/ 儲存管理員執行。

啟用基於憑證的驗證

新建和現有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：用戶端憑證的 Base64 編碼值。
- clientPrivateKey: 關聯私密金鑰的 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。如果未使用受信任的 CA，則可以忽略此參數。

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 Common Name（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. 將信任的 CA 憑證新增至 ONTAP 叢集。儲存管理員可能已經處理此作業。如果未使用信任的 CA ，請忽略。

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

3. 在 ONTAP 叢集上安裝用戶端憑證和金鑰（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
```

> 執行此命令後，ONTAP 會提示輸入憑證。貼上步驟 1 中產生的 `k8senv.pem`檔案內容，然後輸入 `END`以完成安裝。

4. 確認 ONTAP 安全登入角色支援 `cert` 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. 使用產生的憑證測試驗證。將 <ONTAP Management LIF> 和 <vserver name> 替換為 Management LIF IP 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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

更新驗證方法或輪換認證資料

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。此操作雙向有效：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 / 密碼的身份驗證。為此，您必須移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。之後，後端需要進行更新。輪換證書時，可以為該使用者新增多個證書。後端隨後會更新以使用新證書，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存取，也不會影響之後建立的磁碟區連線。後端更新成功表示 Trident 可以與 ONTAP 後端通訊並處理未來的磁碟區作業。

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。當您在 Trident 後端組態中包含使用者名稱時，Trident 會使用您建立的 ONTAP 叢集角色來執行操作。

如需建立 Trident 自訂角色的詳細資訊，請參閱 "Trident 自訂角色產生器"。

使用 **ONTAP CLI**

1. 使用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 使用者建立使用者名稱：

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

3. 將角色對應至使用者：

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

使用 **System Manager**

在 ONTAP System Manager 中執行下列步驟：

1. 建立自訂角色 ：
   a. 若要在叢集層級建立自訂角色，請選取 **Cluster > Settings** 。

      （或）若要在 SVM 層級建立自訂角色、請選取 **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**。

   b. 選擇 **Users and Roles** 旁邊的箭頭圖示（→） 。
   c. 在 **Roles** 下選擇 **+Add**。
   d. 定義角色規則，然後點選 **Save** 。
2. 將角色對應到 **Trident** 使用者： + 在 **Users and Roles** 頁面上執行下列步驟：
   a. 在 **Users** 下方選擇 Add 圖示 **+**。
   b. 選擇所需的使用者名稱，然後在 **Role** 下拉式選單中選擇角色。
   c. 按一下 **Save** 。

如需更多資訊、請參閱下列頁面：

- "用於管理 ONTAP 的自訂角色" 或 "定義自訂角色"
- "使用角色和使用者"

使用雙向 **CHAP** 驗證連線

Trident 可以使用雙向 CHAP 對 iSCSI 工作階段進行驗證，適用於 `ontap-san`和 `ontap-san-economy`驅動程式。這需要在後端定義中啟用 `useCHAP`選項。當設定為 `true`時，Trident 會將 SVM 的預設啟動器安全性設定為雙向 CHAP，並從後端檔案設定使用者名稱和密碼。NetApp 建議使用雙向 CHAP 來驗證連線。請參閱以下範

例組態：

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

useCHAP 參數為布林值選項，只能配置一次。預設值為 false。將其設為 true 後，無法再將其設為 false。

除了 `useCHAP=true`之外， `chapInitiatorSecret`、 `chapTargetInitiatorSecret`、 `chapTargetUsername`和 `chapUsername`欄位也必須包含在後端定義中。後端建立完成後，可以透過執行 `tridentctl update`來變更金鑰。

運作方式

將 `useCHAP`設為 true 後，儲存管理員會指示 Trident 在儲存後端設定 CHAP。這包括以下內容：

- 在 SVM 上設定 CHAP：
  - 如果 SVM 的預設發起程序安全類型為 none（預設設定）*且*磁碟區中沒有預先存在的 LUN，則 Trident 會將預設安全類型設為 `CHAP`，並繼續設定 CHAP 發起程式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。這確保對 SVM 上已存在的 LUN 的存取不受限制。
- 配置 CHAP 啟動器和目標使用者名稱和密碼；這些選項必須在後端組態中指定（如上所示）。

後端建立完成後， Trident 會建立對應的 `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 協定進行掛載和連線。

輪換認證資料並更新後端

您可以透過更新 `backend.json` 檔案中的 CHAP 參數來更新 CHAP 憑證。這需要更新 CHAP 金鑰，並使用 `tridentctl update` 命令來反映這些變更。

更新後端的 CHAP 密碼時，您必須使用 `tridentctl`來更新後端。請勿使用 ONTAP CLI 或 ONTAP System Manager 更新儲存叢集上的認證，因為 Trident 將無法識別這些變更。

```
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 上更新了認證資料，這些連線將繼續保持作用中狀態。新連線使用更新的認證資料，現有連線則繼續保持作用中狀態。中斷連線並重新連線舊的 PV 將導致其使用更新的認證資料。

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。本節提供後端組態範例以及將後端對應至 StorageClasses 的詳細資訊。

"ASA r2 系統" 與其他 ONTAP 系統（ASA、AFF 和 FAS）在儲存層的實作方面有所不同。這些差異會影響某些參數的使用方式（如註明）。"深入瞭解 ASA r2 系統與其他 ONTAP 系統之間的差異"。

ASA r2 系統僅支援 `ontap-san`驅動程式（使用 iSCSI、NVMe/TCP 和 FC 協定）。

在 Trident 後端組態中、您不需要指定系統為 ASA r2。當您選擇 `ontap-san`作為 `storageDriverName` 時、Trident 會自動偵測 ASA r2 或其他 ONTAP 系統。如下表所示、某些後端組態參數不適用於 ASA r2 系統。

請參閱下表以了解後端組態選項：

參數	說明	預設
`version`		始終為 1
`storageDriverName`	儲存驅動程式的名稱	`ontap-san` 或 `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]`。 如需無縫 MetroCluster 切換、請參閱 MetroCluster 範例。 ⓘ 如果使用「 vsadmin 」認證、`managementLIF`則必須是 SVM 的認證；如果使用「 admin 」認證、`managementLIF`則必須是叢集的認證。	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	協定 LIF 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]`。*對於 iSCSI，請勿指定此參數。*Trident 使用"ONTAP Selective LUN Map"來發現建立多路徑會話所需的 iSCSI LIF。如果 `dataLIF`明確定義了此參數，則會產生警告。*對於 MetroCluster，請省略此參數。*請參閱MetroCluster 範例。	由 SVM 導出
`svm`	要使用的儲存虛擬機器 *MetroCluster 除外。*請參閱 MetroCluster 範例。	如果指定了 `managementLIF` SVM ，則衍生
`useCHAP`	使用 CHAP 對 ONTAP SAN 驅動程式的 iSCSI 進行驗證 [布林值]。設定為 `true`時、Trident 會將雙向 CHAP 設定並用作後端中指定 SVM 的預設驗證。如需詳細資訊、請參閱"準備使用 ONTAP SAN 驅動程式配置後端"。不支援 **FCP** 或 **NVMe/TCP**。	`false`
`chapInitiatorSecret`	CHAP 啟動器密碼。如果 `useCHAP=true`則為必填項	""
`labels`	要套用於磁碟區的任意 JSON 格式標籤集	""

參數	說明	預設
`chapTargetInitiatorSecret`	CHAP 目標啟動器密碼。如果需要，則為必填項 `useCHAP=true`	""
`chapUsername`	入站使用者名稱。如果 `useCHAP=true`則為必填項	""
`chapTargetUsername`	目標使用者名稱。如果 `useCHAP=true`則為必填項目	""
`clientCertificate`	用戶端憑證的 Base64 編碼值。用於基於憑證的驗證	""
`clientPrivateKey`	用戶端私密金鑰的 Base64 編碼值。用於憑證型驗證	""
`trustedCACertificate`	受信任 CA 憑證的 Base64 編碼值。選用。用於憑證型驗證。	""
`username`	與 ONTAP 叢集通訊所需的使用者名稱。用於基於認證的驗證。有關 Active Directory 驗證，請參閱 "使用 Active Directory 憑證對後端 SVM 進行 Trident 驗證"。	""
`password`	與 ONTAP 叢集通訊所需的密碼。用於基於認證的驗證。有關 Active Directory 驗證，請參閱 "使用 Active Directory 憑證對後端 SVM 進行 Trident 驗證"。	""
`svm`	要使用的儲存虛擬機器	如果指定了 `managementLIF` SVM，則衍生
`storagePrefix`	在 SVM 中配置新磁碟區時所使用的前綴。無法修改。要更新此參數、您需要建立新的後端。	`trident`
`aggregate`	用於配置的 Aggregate （選用；如果設定，則必須指派給 SVM）。對於 `ontap-nas-flexgroup` 驅動程式，此選項將被忽略。如果未指派，則可以使用任何可用的 Aggregate 來配置 FlexGroup Volume。 ⓘ 當 SVM 中的 Aggregate 更新時，Trident 會自動輪詢 SVM 並更新，無需重新啟動 Trident Controller。如果您已在 Trident 中設定用於配置 Volume 的特定 Aggregate，則如果該 Aggregate 被重新命名或從 SVM 中移出，後端將在輪詢 SVM Aggregate 時進入故障狀態。您必須將該 Aggregate 變更為 SVM 中已存在的 Aggregate，或將其完全移除，才能使後端恢復連線狀態。 *請勿指定用於 ASA r2 系統 *。	""

參數	說明	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`不包含使用 Trident 擷取 Aggregate 使用量並加以限制所需的權限。請勿為 **ASA r2** 系統指定。	"（預設不強制執行）
limitVolumeSize	如果要求的磁碟區大小超過此值，則資源配置失敗。此外，也會限制其管理的 LUN 磁碟區大小上限。	"（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	用於疑難排解的偵錯旗標。例如、{"api":false, "method":true} 除非您正在進行疑難排解並需要詳細的記錄傾印、否則請勿使用。	null
useREST	使用 ONTAP REST API 的布林參數。 `useREST`當設定為 `true`時，Trident 使用 ONTAP REST API 與後端通訊；當設定為 `false` 時，Trident 使用 ONTAPI（ZAPI）呼叫與後端通訊。此功能需要 ONTAP 9.11.1 或更新版本。此外，使用的 ONTAP 登入角色必須具有 `ontapi`應用程式的存取權限。預先定義的 `vsadmin`和 `cluster-admin`角色可滿足此要求。從 Trident 24.06 版本和 ONTAP 9.15.1 或更新版本開始， `useREST`預設設定為 `true`；若要使用 ONTAPI（ZAPI）呼叫，請將 `useREST`變更為 `false`。 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、則一律設為 **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 適用於 iSCSI、 nvme 適用於 NVMe/TCP 或 fcp 適用於 SCSI over Fibre Channel （FC）。	iscsi 如果為空

參數	說明	預設
`formatOptions`	使用 `formatOptions`為 `mkfs`命令指定命令列引數，這些引數將在每次格式化磁碟區時套用。這樣可讓您根據自己的偏好格式化磁碟區。請確保指定的 formatOptions 與 mkfs 命令選項類似，但不包含裝置路徑。範例："-E nodiscard" 支援使用 **iSCSI** 傳輸協定的 **`ontap-san`**和 **`ontap-san-economy`**驅動程式。*此外，使用 iSCSI 和 NVMe/TCP 傳輸協定時，也支援 ASA r2 系統。*	
`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 磁碟區以包含其 LUN 。僅使用預先存在的 Flexvol 來配置新的 PV 。	

使用 **formatOptions** 的建議

Trident 建議採用以下選項來加速格式化程序：

- **-E nodiscard (ext3, ext4):** 在執行 mkfs 時不要嘗試丟棄資料區塊（在固態裝置和稀疏 / 精簡配置儲存上、初始丟棄資料區塊很有用）。此選項取代了已棄用的「 -K 」選項、適用於 ext3 和 ext4 檔案系統。
- **-K (xfs):** 執行 mkfs 時不要嘗試丟棄資料區塊。此選項適用於 xfs 檔案系統。

使用 **Active Directory** 憑證對後端 **SVM** 進行 **Trident** 驗證

您可以設定 Trident 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。若要使用 AD 帳戶進行叢集管理、您必須建立網域通道。如需詳細資訊、請參閱 "在 ONTAP 中設定 Active Directory 網域控制器存取"。

步驟

1. 為後端 SVM 設定網域名稱系統（Domain Name System、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 後端組態檔中，將 `username` 和 `password` 參數分別設定為 AD 使用者或群組名稱和密碼。

您可以使用 `defaults` 配置部分中的這些選項來控制預設配置。例如、請參閱下面的組態範例。

參數	說明	預設
`spaceAllocation`	LUN 的空間分配	"true" 如果指定、請針對 **ASA r2** 系統設定為 **`true`**。
`spaceReserve`	空間保留模式；「 none 」（精簡）或「 volume 」（完整）。針對 **ASA r2** 系統設為 `none`。	"none"
`snapshotPolicy`	要使用的快照原則。針對 **ASA r2** 系統設定為 **`none`**。	"none"
`qosPolicy`	要為建立的磁碟區指派的 QoS 原則群組。每個儲存資源池/後端可選擇 qosPolicy 或 adaptiveQosPolicy 其中之一。搭配 Trident 使用 QoS 原則群組需要 ONTAP 9.8 或更新版本。您應該使用非共享的 QoS 原則群組，並確保該原則群組個別套用至每個成員。共享的 QoS 原則群組會對所有工作負載的總處理量強制執行上限。	""
`adaptiveQosPolicy`	為建立的磁碟區指派的自適應 QoS 原則群組。為每個儲存資源池 / 後端選擇 qosPolicy 或 adaptiveQosPolicy 其中之一	""
`snapshotReserve`	為快照預留的磁碟區百分比。請勿為 **ASA r2** 系統指定。	若 `snapshotPolicy`為「none」，則為「0」，否則為「」
`splitOnClone`	建立時將複本從其父項分割	"false"
`encryption`	在新磁碟區上啟用 NetApp Volume Encryption（NVE）；預設值為 `false`。要使用此選項，叢集必須已獲得 NVE 許可並啟用 NVE。如果後端啟用了 NAE，則在 Trident 中佈建的任何磁碟區都會啟用 NAE。如需詳細資訊，請參閱："Trident 與 NVE 和 NAE 的運作方式"。	"false" 如果指定、請針對 **ASA r2** 系統設定為 **`true`**。
`luksEncryption`	啟用 LUKS 加密。請參閱 "使用 Linux Unified Key Setup （ 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。	""

Volume 配置範例

以下是定義預設值的範例：

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

對於使用 `ontap-san`驅動程式建立的所有磁碟區、Trident 會額外增加 10% 的容量至 FlexVol 以容納 LUN 中繼資料。LUN 將根據使用者在 PVC 中請求的確切大小進行佈建。Trident 會額外增加 10% 的容量至 FlexVol（在 ONTAP 中顯示為可用大小）。使用者現在將獲得他們請求的可用容量。此變更還可防止 LUN 在可用空間未完全使用之前變為唯讀。此變更不適用於 ontap-san-economy。

對於定義了 `snapshotReserve`的後端， Trident 會依照下列方式計算磁碟區的大小：

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

1.1 是 Trident 為 FlexVol 額外增加的 10% ，以容納 LUN 元資料。對於 `snapshotReserve` = 5% ，且 PVC 請求 = 5 GiB ，總磁碟區大小為 5.79 GiB ，可用大小為 5.5 GiB 。 `volume show`命令應顯示與此範例類似的結果：

目前，調整大小是將新計算方法應用於現有磁碟區的唯一方法。

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 for NetApp ONTAP 搭配 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 `ontap-san` 驅動程式的基本配置。

```
---
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 "SVM 複製與復原" 期間進行切換和切換後手動更新後端定義。

為了實現無縫切換和回退，請使用 `managementLIF` 指定 SVM 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>
```

基於憑證的驗證範例

在這個基本設定範例中 `clientCertificate`、 `clientPrivateKey` 和 `trustedCACertificate`（選用，如果使用受信任的 CA）會填入 `backend.json` 中，並分別採用用戶端憑證、私密金鑰和受信任的 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

這些範例建立了一個後端，並將 `useCHAP`設為 `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 economy 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 後端必須設定一個支援 NVMe 的 SVM。這是 NVMe/TCP 的基本後端組態。

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

SCSI over FC （ FCP ） 範例

您的 ONTAP 後端必須設定一個支援 FC 的 SVM。這是 FC 的基本後端組態。

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

使用 **nameTemplate** 的後端組態範例

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

formatOptions ontap-san-economy 驅動程式範例

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

具有虛擬資源池的後端範例

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`為 none、`spaceAllocation`為 false 和 `encryption`為 false。虛擬資源池在儲存區段中定義。

Trident 在「備註」欄位中設定配置標籤。備註設置在 FlexVol volume 上。Trident 在配置時將虛擬資源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資源池定義標籤，並按標籤將磁碟區分組。

在這些範例中，部分儲存資源池設定了自己的 `spaceReserve`、 `spaceAllocation` 和 `encryption` 值，而部分儲存資源池則覆寫了預設值。

ONTAP SAN 範例

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

ONTAP SAN 經濟範例

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
labels:
  store: san_economy_store
region: us_east_1
storage:
  - labels:
      app: oracledb
      cost: "30"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
  - labels:
      app: postgresdb
      cost: "20"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
  - labels:
      app: mysqldb
      cost: "10"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
  - labels:
      department: legal
      creditpoints: "5000"
```

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

NVMe/TCP 範例

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

將後端對應至 StorageClasses

以下 StorageClass 定義均與此相關 [具有虛擬資源池的後端範例]。透過該 `parameters.selector` 字段，每個 StorageClass 定義都會指定哪些虛擬池可用於託管磁碟區。磁碟區將具有所選虛擬池中定義的方面。

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

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

- 該 `protection-not-gold` StorageClass 將對應至 `ontap-san` 後端中的第二個和第三個虛擬資源池。這些是唯一提供金級以外保護層級的資源池。

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

- `app-mysqldb` StorageClass 將對應至 `ontap-san-economy`後端中的第三個虛擬資源池。這是唯一為 mysqldb 類型應用程式提供儲存資源池組態的資源池。

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

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

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

- `creditpoints-5k` StorageClass 將對應至 `ontap-san` 後端的第三個虛擬資源池和 `ontap-san-economy` 後端的第四個虛擬資源池。這些是唯一提供 5000 creditpoints 的資源池。

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

- 此 `my-test-app-sc` StorageClass 會對應到 `testAPP` 虛擬資源池，在 `ontap-san` 驅動程式中使用 `sanType: nvme`。這是唯一提供 `testApp` 的資源池。

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

Trident 將決定選擇哪個虛擬資源池，並確保符合儲存需求。

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 驅動程式詳細資料

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

驅動程式	傳輸協定	**volumeMode**	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`

- 僅當預期持久性磁碟區使用次數高於"支援的 ONTAP Volume 限制"時，才使用 `ontap-san-economy`。
- 僅當預期持久性磁碟區使用次數高於"支援的 ONTAP Volume 限制" 且無法使用 `ontap-san-economy` 驅動程式時，才使用 `ontap-nas-economy`。
- 如果您預計需要資料保護、災難復原或行動性，請勿使用 `ontap-nas-economy`。
- NetApp 除 ontap-san 外，不建議在所有 ONTAP 驅動程式中使用 Flexvol 自動增長功能。作為變通方案，Trident 支援使用快照預留，並相應地擴展 Flexvol 磁碟區。

使用者權限

Trident 希望以 ONTAP 或 SVM 管理員身分執行，通常使用 `admin` 叢集使用者或 `vsadmin` SVM 使用者，或使用具有相同角色但名稱不同的使用者。

對於 Amazon FSx for NetApp ONTAP 部署、 Trident 需要以 ONTAP 或 SVM 管理員身分執行，可以使用叢集 `fsxadmin`使用者、 `vsadmin` SVM 使用者、或使用具有相同角色但名稱不同的使用者。 `fsxadmin`使用者是叢集管理員使用者的有限替代方案。

如果使用 `limitAggregateUsage`參數，則需要叢集管理員權限。將 Amazon FSx for NetApp ONTAP 與 Trident 搭配使用時， `limitAggregateUsage`參數與 `vsadmin`和 `fsxadmin`使用者帳戶不相容。如果指定此參數，組態作業將失敗。

雖然可以在 ONTAP 中建立更嚴格的角色供 Trident 驅動程式使用，但我們不建議這樣做。大多數新版本的 Trident 都會呼叫額外的 API，這些 API 需要考慮，這會讓升級變得困難且容易出錯。

準備使用 **ONTAP NAS** 驅動程式設定後端

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

從 25.10 版本開始，NetApp Trident 支援 "NetApp AFX 儲存系統"。NetApp AFX 儲存系統與其他 ONTAP 系統（ASA、AFF 和 FAS）在儲存層的實作方式上有所不同。

AFX 系統僅支援 `ontap-nas` 驅動程式（使用 NFS 協定）；不支援 SMB 協定。

在 Trident 後端組態中、您無需指定系統為 AFX。當您選擇 `ontap-nas`作為 `storageDriverName`時、Trident 會自動偵測 AFX 系統。

需求

- 對於所有 ONTAP 後端、 Trident 要求至少將一個 Aggregate 指派給 SVM 。
- 您可以執行多個驅動程式，並建立指向其中一個或另一個驅動程式的儲存類別。例如，您可以設定使用 `ontap-nas` 驅動程式的 Gold 類別和使用 `ontap-nas-economy` 驅動程式的 Bronze 類別。
- 所有 Kubernetes 工作節點都必須安裝適當的 NFS 工具。如需更多詳細資料，請參閱 "這裡"。
- Trident 僅支援掛載到在 Windows 節點上執行的 Pod 的 SMB 磁碟區。如需詳細資訊，請參閱 準備配置 SMB Volume。

驗證 **ONTAP** 後端

Trident 提供兩種 ONTAP 後端驗證模式。

- 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安全登入角色關聯的帳戶，例如 `admin`或 `vsadmin`，以確保與 ONTAP 版本的最大相容性。
- 基於憑證：此模式要求在後端安裝憑證，以便 Trident 與 ONTAP 叢集通訊。在這種情況下，後端定義必須包含用戶端憑證、金鑰以及受信任 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，以在基於認證和基於憑證的方法之間移動。但是，一次只支援一種驗證方法。若要切換到不同的驗證方法，您必須從後端組態中移除現有方法。

如果您嘗試同時提供憑證和憑證，則後端建立將會失敗，並出現錯誤，提示組態檔中提供了多個驗證方法。

啟用基於認證的驗證

Trident 需要 SVM 範圍 / 叢集範圍的管理員憑證才能與 ONTAP 後端通訊。建議使用標準預先定義的角色，例如 `admin`或 `vsadmin`。這可確保與未來 ONTAP 版本向前相容,因為未來版本可能會公開供 Trident 版本使用的功能 API。雖然可以建立自訂安全登入角色並將其與 Trident 搭配使用,但不建議這樣做。

後端定義範例如下所示：

YAML

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

JSON

```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認證資料的地方。建立後端之後，使用者名稱/密碼會使用 Base64 編碼並儲存為 Kubernetes 機密。建立/更新後端是唯一需要知道認證資料的步驟。因此，這是僅限管理員的作業，由 Kubernetes/儲存管理員執行。

啟用基於憑證的驗證

新建和現有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：用戶端憑證的 Base64 編碼值。
- clientPrivateKey: 關聯私密金鑰的 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。如果未使用受信任的 CA，則可以忽略此參數。

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 Common Name（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. 將信任的 CA 憑證新增至 ONTAP 叢集。儲存管理員可能已經處理此作業。如果未使用信任的 CA ，請忽略。

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

3. 在 ONTAP 叢集上安裝用戶端憑證和金鑰（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 安全登入角色支援 `cert` 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. 使用產生的憑證測試驗證。將 <ONTAP Management LIF> 和 <vserver name> 替換為管理 LIF IP 和 SVM 名稱。您必須確保 LIF 的服務原則已設為 `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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

更新驗證方法或輪換認證資料

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。此操作雙向有效：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/密碼的身份驗證。為此、您必須移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。之後，後端需要進行更新。輪換證書時，可以為該使用者新增多個證書。後端隨後會更新以使用新證書，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存取，也不會影響之後建立的磁碟區連線。後端更新成功表示 Trident 可以與 ONTAP 後端通訊並處理未來的磁碟區作業。

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。當您在 Trident 後端組態中包含使用者名稱時，Trident 會使用您建立的 ONTAP 叢集角色來執行操作。

如需建立 Trident 自訂角色的詳細資訊，請參閱 "Trident 自訂角色產生器"。

使用 **ONTAP CLI**

1. 使用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 使用者建立使用者名稱：

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

3. 將角色對應至使用者：

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

使用 **System Manager**

在 ONTAP System Manager 中執行下列步驟：

1. 建立自訂角色 ：
   a. 若要在叢集層級建立自訂角色，請選取 **Cluster > Settings** 。

      （或）若要在 SVM 層級建立自訂角色、請選取 **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**。

   b. 選擇 **Users and Roles** 旁邊的箭頭圖示（→） 。
   c. 在 **Roles** 下選擇 **+Add**。
   d. 定義角色規則，然後點選 **Save** 。
2. 將角色對應到 **Trident** 使用者： + 在 **Users and Roles** 頁面上執行下列步驟：
   a. 在 **Users** 下方選擇 Add 圖示 **+**。
   b. 選擇所需的使用者名稱，然後在 **Role** 下拉式選單中選擇角色。
   c. 按一下 **Save** 。

如需更多資訊、請參閱下列頁面：

- "用於管理 ONTAP 的自訂角色" 或 "定義自訂角色"
- "使用角色和使用者"

管理 **NFS** 匯出原則

Trident 使用 NFS 匯出原則來控制對其所配置之磁碟區的存取。

Trident 在處理匯出原則時提供兩種選項：

- Trident 可以動態管理匯出策略；在這種模式下，儲存管理員指定 CIDR 區塊列表，這些 CIDR 區塊代表允許的 IP 位址。Trident 會在發佈時自動將落入這些範圍內的適用節點 IP 位址新增至匯出策略。或者，如果沒有指定 CIDR，則會將磁碟區發佈到的節點上找到的所有全域作用域單播 IP 位址新增至匯出策略。
- 儲存管理員可以建立匯出原則並手動新增規則。除非在組態中指定不同的匯出原則名稱、否則 Trident 會使用預設的匯出原則。

動態管理匯出原則

Trident 提供了動態管理 ONTAP 後端匯出原則的功能。這使得儲存管理員能夠為工作節點 IP 指定允許的位址空間，而無需手動定義明確規則。這大大簡化了匯出原則的管理；對匯出原則的修改不再需要在儲存叢集上進行手動干預。此外，這還有助於將對儲存叢集的存取限制在僅掛載磁碟區且 IP 位址在指定範圍內的工作節點上，從而支援精細和自動化管理。

使用動態匯出策略時，請勿使用網路位址轉換（NAT）。啟用 NAT 後，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，如果在匯出規則中找不到符合項，則會拒絕存取。

範例

必須使用兩種組態選項。以下是後端定義範例：

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

使用此功能時，必須確保 SVM 中的根接合點已預先建立匯出原則，且該原則包含允許節點 CIDR 區塊的匯出規則（例如預設匯出原則）。請務必遵循 NetApp 建議的最佳實務做法，為 Trident 專用 SVM。

以下以上述範例為例、說明此功能的工作原理：

- `autoExportPolicy` 已設定為 `true`。這表示 Trident 會為使用此後端為 `svm1` SVM 配置的每個磁碟區建立一個匯出策略，並使用 `autoexportCIDRs` 位址區塊處理規則的新增和刪除。在磁碟區連接到節點之前，該磁碟區使用一個空的匯出策略，其中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 會建立一個與底層 qtree 同名的匯出策略，該 qtree 包含指定 CIDR 區塊內的節點 IP 位址。這些 IP 位址也會加入到父 FlexVol 磁碟區使用的匯出策略中。
  - 例如：
    - 後端 UUID 403b5326-8482-40db-96d0-d83fb3f4daec

- `autoExportPolicy` 設定為 `true`
- 儲存前置詞 `trident`
- PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
- 名為 trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c 的 qtree 會為名為 `trident-403b5326-8482-40db96d0-d83fb3f4daec`的 FlexVol 建立匯出原則、為名為 `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`的 qtree 建立匯出原則，並在 SVM 上建立名為 `trident_empty`的空白匯出原則。FlexVol 匯出原則的規則將是 qtree 匯出原則中所含任何規則的超集。未附加的磁碟區將重複使用空白匯出原則。

- `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 會新增在已發佈訊息的工作節點上找到的所有全域作用域單播位址。

在本例中，提供了 `192.168.0.0/24` 位址空間。這表示位於此位址範圍內且具有發佈的 Kubernetes 節點 IP 將新增至 Trident 建立的匯出原則。當 Trident 註冊其執行所在的節點時，會擷取該節點的 IP 位址，並根據 `autoExportCIDRs` 中提供的位址區塊進行檢查。在發佈時，篩選 IP 後，Trident 會為其發佈目標節點的用戶端 IP 建立匯出原則規則。

建立後端後，您可以更新其 `autoExportPolicy`和 `autoExportCIDRs`。您可以為自動管理的後端附加新的 CIDR 或刪除現有的 CIDR。刪除 CIDR 時請務必小心，以確保現有連線不會中斷。您也可以選擇停用後端的 `autoExportPolicy`，並回退到手動建立的匯出原則。這需要在後端組態中設定 `exportPolicy`參數。

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

```
./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
```

當節點被移除時、 Trident 會檢查所有匯出原則、以移除與該節點對應的存取規則。透過從受管理後端的匯出原則中移除此節點 IP 、 Trident 可防止惡意掛載、除非叢集中的新節點重複使用此 IP 。

對於先前存在的後端，使用 `tridentctl update backend`更新後端可確保 Trident 自動管理匯出原則。這會在需要時建立兩個新的匯出原則，分別以後端的 UUID 和 qtree 名稱命名。後端上的磁碟區在卸載並重新掛載後，將使用新建立的匯出原則。

> 刪除具有自動管理匯出原則的後端將刪除動態建立的匯出原則。如果重新建立該後端，則會將其視為新後端，並建立新的匯出原則。

如果正在執行中的節點的 IP 位址更新，您必須重新啟動該節點上的 Trident pod。Trident 隨後會更新其管理的後端匯出原則，以反映此 IP 位址變更。

準備配置 SMB Volume

稍加準備，即可使用 `ontap-nas` 驅動程式配置 SMB Volume 。

> 您必須在 SVM 上同時設定 NFS 和 SMB/CIFS 通訊協定，才能為 ONTAP 內部部署叢集建立 `ontap-nas-economy` SMB Volume。若未設定其中任一通訊協定，將導致 SMB Volume 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。

- 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 僅支援掛載到在 Windows 節點上執行的 pod 的 SMB 磁碟區。
- 至少需要一個包含您的 Active Directory 憑證的 Trident 金鑰。要產生金鑰 `smbcreds`：

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

- CSI Proxy 設定為 Windows 服務。若要設定 `csi-proxy`，請參閱"GitHub：CSI Proxy"或"GitHub：適用於 Windows 的 CSI Proxy"以瞭解在 Windows 上執行的 Kubernetes 節點。

步驟

1. 對於內部部署 ONTAP、您可以選擇性地建立 SMB 共用區、或 Trident 可以為您建立一個。

   > Amazon FSx for ONTAP 需要 SMB 共用。

   您可以透過兩種方式建立 SMB 管理共用：使用 "Microsoft Management Console" 共用資料夾嵌入式管理單元或使用 ONTAP CLI。若要使用 ONTAP CLI 建立 SMB 共用：

   a. 如有必要、請建立共用區的目錄路徑結構。

      此 `vserver cifs share create`指令會檢查在建立共用時透過 -path 選項指定的路徑。如果指定的路徑不存在，則命令執行失敗。

   b. 建立與指定 SVM 相關聯的 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. 確認共用已建立：

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

詳情請參閱 "建立 SMB 共用區"。

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"。

參數	說明	範例
`smbShare`	您可以指定以下選項之一：使用 Microsoft Management Console 或 ONTAP CLI 建立的 SMB 共用名稱；允許 Trident 建立 SMB 共用的名稱；或者您可以將此參數留空以封鎖對磁碟區的公共共用存取。對於內部部署 ONTAP，此參數為選用項目。對於 Amazon FSx for ONTAP 後端，此參數為必要項目且不能為空白。	`smb-share`
`nasType`	* 必須設為 `smb`。*如果為 null ，則預設為 `nfs`。	`smb`
`securityStyle`	新磁碟區的安全樣式。對於 **SMB** 磁碟區，必須設定為 **`ntfs`或 `mixed`**。	`ntfs` 或 `mixed` 適用於 SMB 磁碟區
`unixPermissions`	新磁碟區的模式。**SMB** 磁碟區必須保留空白。	""

啟用安全的 **SMB**

從 25.06 版本開始，NetApp Trident 支援使用 `ontap-nas`和 `ontap-nas-economy`後端建立的 SMB 磁碟區的安全性資源配置。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 共用的受控存取權限。

要記住的要點

- 不支援匯入 `ontap-nas-economy`磁碟區。
- `ontap-nas-economy` 磁碟區僅支援唯讀複本。
- 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。
- 複製 PVC 註釋中指定的 SMB 共用 ACL 將優先於來源 PVC 中的 ACL 。
- 啟用安全 SMB 時，請確保提供有效的 AD 使用者。無效使用者將不會被加入到 ACL 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，則權限優先順序為：PVC、儲存類別，然後是後端。

- 安全 SMB 支援 `ontap-nas`託管磁碟區匯入，但不適用於非託管磁碟區匯入。

步驟

1. 請在 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. 在儲存類別中新增註解。

   將 `trident.netapp.io/smbShareAdUser` 註解新增至儲存類別，以啟用安全 SMB 而不會失敗。為註解 `trident.netapp.io/smbShareAdUser` 指定的使用者值應與 `smbcreds` 密鑰中指定的使用者名稱相同。您可以為 `smbShareAdUserPermission` 選擇下列其中一項： `full_control`、 `change` 或 `read`。預設權限為 `full_control`。

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

1. 建立 PVC 。

   以下範例建立 PVC：

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

ONTAP NAS 設定選項和範例

學習如何在 Trident 安裝中建立和使用 ONTAP NAS 驅動程式。本節提供後端組態範例以及將後端對應至 StorageClasses 的詳細資訊。

從 25.10 版本開始，NetApp Trident 支援 "NetApp AFX 儲存系統"。NetApp AFX 儲存系統與其他基於 ONTAP 的系統（ASA、AFF 和 FAS）在儲存層的實作方式上有所不同。

僅支援 `ontap-nas` 驅動程式（使用 NFS 協定）用於 NetApp AFX 系統；不支援 SMB 協定。

在 Trident 後端設定中，您無需指定係統為 NetApp AFX 儲存系統。當您選擇 `ontap-nas`作為 `storageDriverName`時，Trident 會自動偵測 AFX 儲存系統。如下表所示，某些後端組態參數不適用於 AFX 儲存系統。

後端組態選項

請參閱下表以了解後端組態選項：

參數	說明	預設
`version`		始終為 1

參數	說明	預設
`storageDriverName`	儲存驅動程式的名稱 對於 NetApp AFX 系統，僅支援 `ontap-nas`。	`ontap-nas`、 `ontap-nas-economy` 或 `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。可以指定完全限定域名（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]`。有關無縫 MetroCluster 切換，請參閱MetroCluster 範例。	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	協定 LIF 的 IP 位址。NetApp 建議指定 `dataLIF`。如果未提供、Trident 會從 SVM 擷取 dataLIF。您可以指定完整網域名稱（FQDN）以用於 NFS 掛載作業、讓您建立循環配置資源 DNS、以便在多個 dataLIF 之間進行負載平衡。可在初始設定後變更。請參閱。如果 Trident 是使用 IPv6 旗標安裝、則可設定為使用 IPv6 位址。IPv6 位址必須以方括弧定義、例如 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`。* 省略 MetroCluster 。*請參閱MetroCluster 範例。	指定的位址或從 SVM 衍生（如果未指定）（不建議）
`svm`	要使用的儲存虛擬機器 *MetroCluster 除外。*請參閱 MetroCluster 範例。	如果指定了 `managementLIF` SVM，則衍生
`autoExportPolicy`	啟用自動匯出原則建立和更新 [布林值]。使用 `autoExportPolicy`和 `autoExportCIDRs`選項、Trident 可以自動管理匯出原則。	錯誤
`autoExportCIDRs`	啟用 `autoExportPolicy`時用於篩選 Kubernetes 節點 IP 的 CIDR 清單。使用 `autoExportPolicy`和 `autoExportCIDRs`選項，Trident 可以自動管理匯出原則。	["0.0.0.0/0", "::/0"]`
`labels`	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。用於基於憑證的驗證	""
`clientPrivateKey`	用戶端私密金鑰的 Base64 編碼值。用於憑證型驗證	""
`trustedCACertificate`	受信任 CA 憑證的 Base64 編碼值。選用。用於憑證型驗證	""
`username`	用於連接叢集 / SVM 的使用者名稱。用於基於憑證的身份驗證。有關 Active Directory 身份驗證，請參閱 "使用 Active Directory 憑證對後端 SVM 進行 Trident 驗證"。	

參數	說明	預設
`password`	連接到叢集 / SVM 的密碼。用於基於憑證的驗證。有關 Active Directory 驗證、請參閱 "使用 Active Directory 憑證對後端 SVM 進行 Trident 驗證"。	
`storagePrefix`	在 SVM 中配置新磁碟區時所使用的前置字元。設定後無法更新 當使用 ontap-nas-economy 並且 storagePrefix 為 24 個字元或更長時，qtree 將不會嵌入儲存前綴，儘管它會包含在磁碟區名稱中。	"Trident"
`aggregate`	用於配置的 Aggregate （選用；如果設定，則必須指派給 SVM）。對於 `ontap-nas-flexgroup` 驅動程式，此選項將被忽略。如果未指派，則可以使用任何可用的 Aggregate 來配置 FlexGroup Volume。 當 SVM 中的 Aggregate 更新時，Trident 會自動輪詢 SVM 並更新，無需重新啟動 Trident Controller。如果您已在 Trident 中設定用於配置 Volume 的特定 Aggregate，則如果該 Aggregate 被重新命名或從 SVM 中移出，後端將在輪詢 SVM Aggregate 時進入故障狀態。您必須將該 Aggregate 變更為 SVM 中已存在的 Aggregate，或將其完全移除，才能使後端恢復連線狀態。 請勿指定用於 **AFX** 儲存系統。	""
`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**。請勿為 **AFX** 儲存系統指定此規則。	" （預設不強制執行）

參數	說明	預設
flexgroupAggregateList	用於配置的 Aggregate 清單（選用；如果設定，則必須指派給 SVM）。指派給 SVM 的所有 Aggregate 都將用於配置 FlexGroup Volume。**ontap-nas-flexgroup** 儲存驅動程式支援此功能。 當 SVM 中的 Aggregate 清單更新時，Trident 會自動輪詢 SVM 來更新該清單，無需重新啟動 Trident Controller。如果您已在 Trident 中設定特定的 Aggregate 清單來配置磁碟區，若該 Aggregate 清單已重新命名或從 SVM 中移出，Trident 在輪詢 SVM Aggregate 時後端將進入故障狀態。您必須將該 Aggregate 清單變更為 SVM 中存在的清單，或將其完全移除，才能使後端恢復連線狀態。	""
`limitVolumeSize`	如果要求的磁碟區大小超過此值，則資源配置會失敗。	"（預設不強制執行）
`debugTraceFlags`	用於疑難排解的偵錯旗標。例如、{"api":false, "method":true}除非您正在進行疑難排解並需要詳細的記錄傾印、否則請勿使用 `debugTraceFlags`。	null
`nasType`	配置 NFS 或 SMB 磁碟區的建立。選項為 `nfs`、 `smb` 或 `null`。設定為 `null` 時，預設使用 `NFS` 磁碟區。如果指定，則對於 **`AFX`** 儲存系統始終設定為 **`nfs`**。	`nfs`
`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，但如果儲存類別中未指定任何掛載選項， Trident 將回退到使用儲存後端設定檔中指定的掛載選項。如果儲存類別和設定檔中均未指定掛載選項， 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`	您可以指定以下選項之一：使用 Microsoft Management Console 或 ONTAP CLI 建立的 SMB 共用名稱；允許 Trident 建立 SMB 共用的名稱；或者您可以將此參數留空以封鎖對磁碟區的公共共用存取。對於內部部署 ONTAP，此參數為選用項目。對於 Amazon FSx for ONTAP 後端，此參數為必要項目且不能為空白。	`smb-share`

參數	說明	預設
useREST	用於使用 ONTAP REST API 的布林參數。 useREST` 當設定為 `true`時，Trident 使用 ONTAP REST API 與後端通訊；當設定為 `false`時，Trident 使用 ONTAPI（ZAPI）呼叫與後端通訊。此功能需要 ONTAP 9.11.1 及更新版本。此外，使用的 ONTAP 登入角色必須具有 `ontapi`應用程式的存取權限。預先定義的 `vsadmin`和 `cluster-admin`角色可滿足此要求。從 Trident 24.06 版本和 ONTAP 9.15.1 或更新版本開始， `useREST`預設設定為 `true；將 useREST`變更為 `false`以使用 ONTAPI（ZAPI）呼叫。如果指定，對於 **AFX** 儲存系統，請始終設定為 `**true**。	true 適用於 ONTAP 9.15.1 或更高版本，否則 false。
limitVolumePoolSize	在 ontap-nas-economy 後端中使用 Qtree 時可請求的最大 FlexVol 大小。	"（預設不強制執行）
denyNewVolumePools	限制 `ontap-nas-economy`後端建立新 FlexVol 磁碟區來存放其 Qtree 。僅使用預先存在的 Flexvol 來配置新的 PV 。	
adAdminUser	擁有對 SMB 共用完全存取權限的 Active Directory 管理員使用者或使用者群組。使用此參數可授予對 SMB 共用的完全控制權限。	

磁碟區配置的後端組態選項

您可以使用 defaults 配置部分中的這些選項來控制預設配置。例如、請參閱下面的組態範例。

參數	說明	預設
spaceAllocation	Qtree 的空間分配	"true"
spaceReserve	空間保留模式；「 none 」（精簡）或「 volume 」（完整）	"none"
snapshotPolicy	要使用的 Snapshot 原則	"none"
qosPolicy	為建立的磁碟區指派的 QoS 策略群組。為每個儲存池 / 後端選擇 qosPolicy 或 adaptiveQosPolicy 其中之一	""
adaptiveQosPolicy	為建立的磁碟區指派的自適應 QoS 原則群組。每個儲存資源池 / 後端可選擇 qosPolicy 或 adaptiveQosPolicy 其中之一。ontap-nas-economy 不支援。	""
snapshotReserve	為快照保留的磁碟區百分比	若 `snapshotPolicy`為「none」，則為「0」，否則為「」
splitOnClone	建立時將複本從其父項分割	"false"

參數	說明	預設
`encryption`	在新磁碟區上啟用 NetApp Volume Encryption（NVE）；預設值為 `false`。要使用此選項，叢集必須已獲得 NVE 許可並啟用 NVE。如果後端啟用了 NAE，則在 Trident 中佈建的任何磁碟區都會啟用 NAE。如需詳細資訊，請參閱："Trident 與 NVE 和 NAE 的運作方式"。	"false"
`tieringPolicy`	分層策略使用 "none"	
`unixPermissions`	新磁碟區模式	NFS 磁碟區為「777」；SMB 磁碟區為空（不適用）
`snapshotDir`	控制對 `.snapshot` 目錄的存取	NFSv4 為 "true" ， NFSv3 為 "false"
`exportPolicy`	要使用的匯出原則	"default"
`securityStyle`	新磁碟區的安全樣式。NFS 支援 `mixed`和 `unix`安全樣式。SMB 支援 `mixed`和 `ntfs`安全樣式。	NFS 預設值為 `unix`。SMB 預設值為 `ntfs`。
`nameTemplate`	用於建立自訂磁碟區名稱的範本。	""

使用 QoS 原則群組搭配 Trident 需要 ONTAP 9.8 或更新版本。您應該使用非共享的 QoS 原則群組，並確保該原則群組分別套用於每個成員。共享的 QoS 原則群組會強制限制所有工作負載的總處理量上限。

Volume 配置範例

以下是定義預設值的範例：

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

對於 `ontap-nas`和 `ontap-nas-flexgroups`，Trident 現在使用新的計算方法，以確保 FlexVol 的大小與 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更大的原始 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，當使用者請求 PVC（例如 5 GiB）且 snapshotReserve 百分比為 50% 時，他們只能獲得 2.5 GiB 的可寫空間。這是因為使用者要求的是整個磁碟區，而 `snapshotReserve`是其百分比。在 `Trident 21.07` 中，使用者要求的是可寫入空間，`Trident` 將該 `snapshotReserve`數值定義為整個磁碟區的百分比。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 snapshotReserve = 50% 以及 PVC 請求 = 5 GiB，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 請求中所要求的大小。 `volume show` 命令應顯示類似於此範例的結果：

升級 Trident 時，先前安裝的現有後端將按照上述說明配置磁碟區。對於升級前建立的磁碟區，您需要調整其大小才能使變更生效。例如，先前使用 `snapshotReserve=50` 建立的 2 GiB PVC 會產生提供 1 GiB 可寫入空間的磁碟區。將磁碟區大小調整為 3 GiB 後，應用程式將在 6 GiB 磁碟區上獲得 3 GiB 的可寫入空間。

最小組態範例

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

> 如果您在 NetApp ONTAP 上使用 Amazon FSx 搭配 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 "SVM 複製與復原" 期間進行切換和切換後手動更新後端定義。

為了實現無縫切換和切換回，請使用 `managementLIF`指定 SVM 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
```

基於憑證的驗證範例

這是一個最小的後端設定範例。 `clientCertificate`、 `clientPrivateKey` 和 `trustedCACertificate`（如果使用受信任的 CA，則為可選）分別填充在 `backend.json` 中，並分別接受客戶端憑證、私鑰和受信任的 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_
```

自動匯出原則範例

本範例示範如何指示 Trident 使用動態匯出原則來自動建立和管理匯出原則。這對於 `ontap-nas-economy` 和 `ontap-nas-flexgroup`驅動程式的運作方式相同。

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

IPv6 位址範例

此範例展示 `managementLIF`使用 IPv6 位址。

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

使用 SMB 磁碟區的 Amazon FSx for ONTAP 範例

使用 SMB 磁碟區的 FSx for ONTAP 需要 `smbShare` 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** 的後端組態範例

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

具有虛擬資源池的後端範例

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`為 none、`spaceAllocation`為 false 和 `encryption`為 false。虛擬池在儲存部分中定義。

Trident 在「備註」欄位中設定配置標籤。備註可以針對 `ontap-nas`在 FlexVol 上設定，或針對 `ontap-nas-flexgroup`在 FlexGroup 上設定。Trident 在配置時會將虛擬資源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資源池定義標籤，並按標籤將磁碟區分組。

在這些範例中，部分儲存資源池設定了自己的 `spaceReserve`、 `spaceAllocation` 和 `encryption` 值，而部分儲存資源池則覆寫了預設值。

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"
```

將後端對應至 StorageClasses

以下 StorageClass 定義均指涉[具有虛擬資源池的後端範例]。透過 `parameters.selector` 欄位，每個 StorageClass 都會指定哪些虛擬資源池可用於託管磁碟區。磁碟區將具有所選虛擬資源池中定義的各個層面。

- `protection-gold` StorageClass 將對應至 `ontap-nas-flexgroup`後端的第一個和第二個虛擬資源池。這些是唯一提供金級保護的資源池。

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

- 該 `protection-not-gold` StorageClass 將映射到 `ontap-nas-flexgroup`後端的第三和第四個虛擬資源池。這些是唯一提供 gold 以外保護等級的資源池。

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

- `app-mysqldb`StorageClass 將對應至 `ontap-nas 後端的第四個虛擬資源池。這是唯一為 mysqldb 類型應用程式提供儲存資源池組態的資源池。

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

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

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

- `creditpoints-5k` StorageClass 將對應至 `ontap-nas` 後端的第三個虛擬資源池和 `ontap-nas-economy` 後端的第二個虛擬資源池。這些是唯一提供 5000 creditpoints 的資源池。

```
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資源池，並確保符合儲存需求。

初始配置後更新 `dataLIF`

初始設定完成後，您可以執行以下命令來變更 dataLIF，以提供包含更新 dataLIF 的新後端 JSON 檔案。

```
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 Pod、則必須關閉所有對應的 Pod、然後再將其重新啟動、以便讓新的 dataLIF 生效。

安全 **SMB** 範例

使用 **ontap-nas** 驅動程式進行後端組態

```
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** 驅動程式進行後端組態

```
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
```

後端配置與儲存資源池

```
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
```

請務必新增 `annotations`以啟用安全 SMB 。無論後端或 PVC 中如何配置、如果沒有這些註釋、安全 SMB 都無法正常運作。

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** 搭配使用

"Amazon FSx for NetApp ONTAP" 是一項完全託管的 AWS 服務，可讓客戶啟動並執行由 NetApp ONTAP 儲存作業系統提供支援的檔案系統。FSx for ONTAP 讓您能夠利用熟悉的 NetApp 特性、效能和管理功能，同時享受在 AWS 上儲存資料的簡易性、敏捷性、安全性和可擴充性。FSx for ONTAP 支援 ONTAP 檔案系統特性和管理 API。

您可以將 Amazon FSx for NetApp ONTAP 檔案系統與 Trident 整合，以確保在 Amazon Elastic Kubernetes Service （EKS） 中執行的 Kubernetes 叢集可以配置由 ONTAP 支援的區塊和檔案持續磁碟區。

檔案系統是 Amazon FSx 中的主要資源，類似於內部部署的 ONTAP 叢集。在每個 SVM 中，您可以建立一個或多個磁碟區，這些磁碟區是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。這種新的檔案系統類型稱為 **NetApp ONTAP**。

將 Trident 與 Amazon FSx for NetApp ONTAP 搭配使用，可確保在 Amazon Elastic Kubernetes Service（EKS ）中執行的 Kubernetes 叢集能夠配置由 ONTAP 支援的區塊和檔案持續磁碟區。

需求

此外 "Trident 需求"，要將 FSx for ONTAP 與 Trident 整合，您還需要：

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

> 請確保根據您的 EKS AMI 類型，按照 Amazon Linux 和 Ubuntu "Amazon Machine Images"（AMI）所需的節點準備步驟進行操作。

考量事項

- SMB 磁碟區：
  - 僅透過 `ontap-nas`驅動程式支援 SMB 磁碟區。
  - Trident EKS 外掛程式不支援 SMB 磁碟區。
  - Trident 僅支援掛載到在 Windows 節點上執行的 Pod 的 SMB 磁碟區。如需詳細資訊，請參閱 "準備配置 SMB Volume"。
- 在 Trident 24.02 之前，在啟用自動備份的 Amazon FSx 檔案系統上建立的磁碟區無法由 Trident 刪除。為了避免在 Trident 24.02 或更高版本中出現此問題，請在 AWS FSx for ONTAP 的後端組態檔中指定 `fsxFilesystemID`、AWS `apiRegion`、AWS `apikey`和 `AWS secretKey`。

> 如果您要為 Trident 指定 IAM 角色，則可以省略明確指定 `apiRegion`、`apiKey` 和 `secretKey` 欄位給 Trident。如需更多資訊，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）搭配使用，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。為確保多路徑功能正常運作而不干擾同一節點上的 EBS 磁碟，您需要在多路徑設定中排除 EBS。以下範例展示了一個 `multipath.conf`檔案，其中包含所需的 Trident 設定，同時將 EBS 磁碟從多路徑中排除：

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

驗證

Trident 提供兩種驗證模式。

- 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`使用者作為您的檔案系統，或使用為您的 SVM 設定的 `vsadmin`使用者。

> Trident 需要以 `vsadmin` SVM 使用者身分執行，或以具有相同角色但名稱不同的使用者身分執行。Amazon FSx for NetApp ONTAP 提供了一個 `fsxadmin`使用者，可作為 ONTAP `admin`叢集使用者的有限替代方案。我們強烈建議將 `vsadmin`與 Trident 搭配使用。

- 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。

有關啟用身分驗證的詳細資訊，請參閱您的驅動程式類型的身分驗證說明：

- "ONTAP NAS 認證"
- "ONTAP SAN 驗證"

已測試的 **Amazon Machine Images**（**AMI**）

EKS 叢集支援多種作業系統，但 AWS 已針對容器和 EKS 優化了某些 Amazon Machine Images（AMIs）。以下 AMIs 已使用 NetApp Trident 25.02 進行測試。

AMI	NAS	NAS 經濟	iSCSI	iSCSI 經濟性
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	N/A	N/A
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	N/A	N/A

- * 無法在不重新啟動節點的情況下刪除 PV
- ** 無法與 Trident 版本 25.02 的 NFSv3 搭配使用。

> 如果您所需的 AMI 未在此列出，並不意味著它不受支援；這僅僅意味著它尚未經過測試。此列表僅供參考，列出了已知可正常運作的 AMI。

使用以下工具進行測試：

- EKS 版本：1.32
- 安裝方法：Helm 25.06 和做為 AWS 附加元件 25.06
- 對於 NAS ， NFSv3 和 NFSv4.1 都進行了測試。

- 對於 SAN，僅測試了 iSCSI，未測試 NVMe-oF。

已執行測試：

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

尋找更多資訊

- "Amazon FSx for NetApp ONTAP 文件"
- "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、您必須擁有使用 EKS 部署的 Kubernetes 叢集。

建立 **AWS Secrets Manager** 密碼

由於 Trident 將針對 FSx vserver 發出 API 來為您管理儲存設備，因此需要相應的認證資料。傳遞這些認證資料的安全方法是透過 AWS Secrets Manager 密碼。因此，如果您還沒有密碼，則需要建立一個包含 vsadmin 帳戶認證資料的 AWS Secrets Manager 密碼。

此範例建立一個 AWS Secrets Manager 密碼來儲存 Trident CSI 認證：

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

建立 **IAM** 政策

Trident 也需要 AWS 權限才能正常運作。因此、您需要建立一個原則、授予 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"
```

Policy 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 Identity** 或 **IAM** 角色以關聯服務帳戶（**IRSA**）

您可以使用 EKS Pod Identity 或 IAM role for Service account association (IRSA) 設定 Kubernetes 服務帳戶來承擔 AWS Identity and Access Management (IAM) 角色。任何已設定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 Identity

Amazon EKS Pod Identity 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 身分，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
```

如需詳細資訊，請參閱 "設定 Amazon EKS Pod Identity Agent"。

建立 **trust-relationship.json**：

建立 trust-relationship.json 檔案，使 EKS Service Principal 能夠承擔 Pod Identity 的此角色。然後使用此信任政策建立角色：

```
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 檔案：

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

將角色原則附加到 **IAM** 角色：

將上一個步驟中的角色原則附加到已建立的 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 檔案：

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

請更新 `trust-relationship.json` 文件中的以下值：

- **<account_id>** - 您的 AWS 帳號 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：

建立角色後，使用此命令將原則（在上述步驟中建立）附加至角色：

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

驗證 **OICD** 提供者是否已關聯：

請確認您的 OIDC 提供者已關聯到您的叢集。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：

- Helm
- EKS 附加元件

如果您想使用快照功能，請安裝 CSI 快照控制器外掛程式。如需詳細資訊，請參閱 "為 CSI 磁碟區啟用快照功能"。

透過 **Helm** 安裝 **Trident**

Pod Identity

1. 新增 Trident Helm 儲存庫：

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

2. 請依照下列範例安裝 Trident：

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

您可以使用 `helm list` 命令來檢視安裝詳細資訊，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
```

服務帳戶關聯（**IRSA**）

1. 新增 Trident Helm 儲存庫：

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

2. 設定 **cloud provider** 和 **cloud identity** 的值：

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

您可以使用 `helm list` 命令來檢視安裝詳細資訊，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-
100.2510.0    25.10.0
```

如果您打算使用 iSCSI ，請確保用戶端電腦上已啟用 iSCSI 。如果您使用的是 AL2023 Worker node OS ，可以透過在 helm 安裝過程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 附加元件包含最新的安全性修補程式、錯誤修正，並經 AWS 驗證可與 Amazon EKS 搭配使用。EKS 附加元件可讓您持續確保 Amazon EKS 叢集的安全性和穩定性，並減少安裝、設定和更新附加元件所需的工作量。

先決條件

在為 AWS EKS 設定 Trident 附加元件之前，請確保您已具備以下條件：

- 具有附加訂閱的 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

啟用適用於 **AWS** 的 **Trident** 附加元件

管理主控台

1. 在 https://console.aws.amazon.com/eks/home#/clusters開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中、選取 **Clusters** 。
3. 選擇要為其配置 NetApp Trident CSI 附加元件的叢集名稱。
4. 選擇 **Add-ons** ，然後選擇 **Get more add-ons** 。
5. 請依照以下步驟選擇附加元件：
   a. 向下捲動至 **AWS Marketplace** 附加元件 部分，然後在搜尋框中輸入 **"Trident"** 。
   b. 選取 Trident by NetApp 方塊右上角的核取方塊。
   c. 選擇 **Next** 。
6. 在 **Configure selected add-ons** 設定頁面上，執行以下操作：

   如果您使用的是 **Pod Identity association**，請跳過這些步驟。

   a. 選取您要使用的 **Version** 。
   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：
      - 選取您要使用的 **Version** 。
      - 依照 **Add-on configuration schema** 進行操作，並將 **configurationValues** 參數在 **Configuration values** 部分設定為您在上一個步驟建立的 role-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」作為衝突解決方式，則現有外掛程式的一個或多個設定可能會被 Amazon EKS 外掛程式的設定覆蓋。如果未啟用此選項且與現有設定有衝突，則操作將會失敗。您可以利用產生的錯誤訊息來排查衝突。選擇此選項之前，請確保 Amazon EKS 外掛程式沒有管理您需要自行管理的設定。

7. 選擇 **Next** 。
8. 在 **Review and add** 頁面上，選擇 **Create** 。

   附加元件安裝完成後，您會看到已安裝的附加元件。

AWS CLI

*1.建立 `add-on.json` 檔案 *：

對於 **Pod Identity** ，請使用以下格式：

使用

```
{
  "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"
  }
}
```

將 `<role ARN>`替換為上一步驟中建立的角色 ARN。

2.安裝 **Trident EKS** 附加元件。

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

eksctl

以下範例命令安裝 Trident EKS 附加元件：

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

更新 **Trident EKS** 附加元件

管理主控台

1. 開啟 Amazon EKS 主控台 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左側導覽窗格中、選取 **Clusters** 。
3. 選擇要為其更新 NetApp Trident CSI 外掛程式的叢集名稱。
4. 選擇 **Add-ons** 標籤。
5. 選取 **Trident by NetApp**，然後選取 **Edit**。
6. 在 **Configure Trident by NetApp** 頁面上、執行以下操作：
   a. 選取您要使用的 **Version** 。
   b. 展開 **Optional configuration settings** 並根據需要進行修改。
   c. 選擇 **Save changes** 。

AWS CLI

以下範例更新 EKS 附加元件：

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

eksctl

- 檢查您的 FSxN Trident CSI 外掛程式的目前版本。將 `my-cluster`替換為您的叢集名稱。

  ```
  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'"}
```

- 將附加元件更新至上一步驟輸出中 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
  ```

如果您移除 `--force`選項，且任何 Amazon EKS 外掛程式設定與您現有的設定衝突，則更新 Amazon EKS 外掛程式將會失敗；您會收到錯誤訊息，以協助您解決衝突。在指定此選項之前，請確保 Amazon EKS 外掛程式不管理您需要管理的設定，因為這些設定會被此選項覆寫。有關此設定的其他選項的更多資訊，請參閱"外掛程式"。有關 Amazon EKS Kubernetes 欄位管理的更多資訊，請參閱"Kubernetes 欄位管理"。

解除安裝 / 移除 **Trident EKS** 附加元件

您有兩種選項可移除 Amazon EKS 附加元件：

- 保留叢集上的附加軟體 – 此選項將移除 Amazon EKS 對所有設定的管理。它還會移除 Amazon EKS 通知您更新以及在您啟動更新後自動更新 Amazon EKS 附加軟體的功能。但是，它會保留叢集上的附加軟體。此選項使附加軟體成為自管理安裝，而不是 Amazon EKS 附加軟體。使用此選項，附加軟體不會出現停機時間。在命令中保留 `--preserve`選項以保留附加軟體。
- 從叢集完全移除附加元件軟體 — NetApp 建議僅在您的叢集上沒有任何依賴該附加元件的資源時，才從叢集移除 Amazon EKS 附加元件。從 `--preserve`指令中移除 `delete`選項以移除附加元件。

如果附加元件關聯了 IAM 帳戶，則不會移除該 IAM 帳戶。

管理主控台

1. 在 https://console.aws.amazon.com/eks/home#/clusters開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中、選取 **Clusters** 。
3. 選擇要移除 NetApp Trident CSI 外掛程式的叢集名稱。
4. 選取 **Add-ons** 標籤，然後選取 **Trident by NetApp**。
5. 選擇 **Remove** 。
6. 在 **Remove netapp_trident-operator confirmation** 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 **Preserve on cluster**。如果您希望將外掛程式軟體保留在叢集上，以便您可以自行管理外掛程式的所有設置，請執行此操作。
   b. 輸入 **netapp_trident-operator** 。
   c. 選擇 **Remove** 。

AWS CLI

將 `my-cluster`替換為您的叢集名稱，然後執行以下命令。

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

eksctl

以下命令會解除安裝 Trident EKS 附加元件：

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

ONTAP SAN 和 NAS 驅動程式整合

若要建立儲存後端，您需要建立 JSON 或 YAML 格式的組態檔。該檔案需要指定所需的儲存類型（ NAS 或 SAN ）、檔案系統、要從中取得資料的 SVM 以及如何進行驗證。以下範例展示如何定義基於 NAS 的儲存設備，以及如何使用 AWS 密碼來儲存要使用的 SVM 認證資料：

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 Backend Configuration（TBC）：

- 從 yaml 檔案建立 Trident 後端組態（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 後端組態（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 for ONTAP 驅動程式詳細資料

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

- `ontap-san`：每個已配置的 PV 都是其自身 Amazon FSx for NetApp ONTAP 磁碟區中的一個 LUN 。推薦用於區塊儲存。
- `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`：每個已配置的 PV 都是一個 LUN，每個 Amazon FSx for NetApp ONTAP 磁碟區可設定 LUN 的數量。
- `ontap-nas-economy`：每個已配置的 PV 都是一個 qtree，每個 Amazon FSx for NetApp ONTAP 磁碟區可配置的 qtree 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。

有關驅動程式詳細資料、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、請執行此命令：

```
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
```

後端進階組態和範例

請參閱下表以了解後端組態選項：

參數	說明	範例
`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，可以指定完整網域名稱（FQDN）。如果 Trident 是使用 IPv6 旗標安裝的，可以設定為使用 IPv6 位址。IPv6 位址必須用中括號定義，例如［28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555］。如果您在 `fsxFilesystemID` 下的 `aws` 欄位提供了 `managementLIF`，則不需要再提供 `managementLIF`，因為 Trident 會從 AWS 擷取 SVM 資訊。因此，您必須為 SVM 下的使用者（例如：vsadmin）提供認證，且該使用者必須具有 `vsadmin` 角色。	"10.0.0.1", "[2001:1234:abcd::fefe]"

參數	說明	範例
dataLIF	協定 LIF 的 IP 位址。**ONTAP NAS** 驅動程式：NetApp 建議指定 dataLIF。如果未提供，Trident 將從 SVM 取得 dataLIF。您可以指定一個完全限定網域名稱（FQDN）用於 NFS 掛載操作，從而建立輪詢 DNS 以在多個 dataLIF 之間進行負載平衡。初始設定後可以更改。請參閱。**ONTAP SAN** 驅動程式：iSCSI 無需指定。Trident 使用 ONTAP 選擇性 LUN 對應來發現建立多路徑會話所需的 iSCSI LIF。如果明確定義了 dataLIF，則會產生警告。如果 Trident 是使用 IPv6 標誌安裝的，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	啟用自動匯出原則建立和更新 [布林值]。使用 `autoExportPolicy`和 `autoExportCIDRs`選項、 Trident 可以自動管理匯出原則。	false
autoExportCIDRs	啟用 `autoExportPolicy`時用於篩選 Kubernetes 節點 IP 的 CIDR 清單。使用 `autoExportPolicy`和 `autoExportCIDRs`選項，Trident 可以自動管理匯出原則。	"["0.0.0.0/0", "::/0"]"
labels	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。用於基於憑證的驗證	""
clientPrivateKey	用戶端私密金鑰的 Base64 編碼值。用於憑證型驗證	""
trustedCACertificate	受信任 CA 憑證的 Base64 編碼值。選用。用於憑證型驗證。	""
username	用於連接叢集或 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	要使用的儲存虛擬機器	如果指定了 SVM 管理 LIF，則會衍生。
storagePrefix	在 SVM 中配置新磁碟區時所使用的前綴。建立後無法修改。若要更新此參數、您需要建立新的後端。	trident

參數	說明	範例
`limitAggregateUsage`	*請勿為 Amazon FSx for NetApp ONTAP 指定。*提供的 `fsxadmin`和 `vsadmin`不包含使用 Trident 檢索 Aggregate 使用情況並加以限制所需的權限。	請勿使用。
`limitVolumeSize`	如果請求的磁碟區大小超過此值，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，並且該 `qtreesPerFlexvol`選項允許自訂每個 FlexVol 磁碟區的最大 qtree 數量	"（預設不強制執行）
`lunsPerFlexvol`	每個 FlexVol volume 的最大 LUN 數量必須在 [50, 200] 範圍內。僅限 SAN。	“100”
`debugTraceFlags`	用於疑難排解的偵錯旗標。例如、{"api":false, "method":true}除非您正在進行疑難排解並需要詳細的記錄傾印、否則請勿使用 `debugTraceFlags`。	null
`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，但如果儲存類別中未指定任何掛載選項， Trident 將回退到使用儲存後端設定檔中指定的掛載選項。如果儲存類別和設定檔中均未指定掛載選項， 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。選項為 `nfs`、 `smb`或 null。*必須設定為 `smb`才能建立 SMB 磁碟區。*設定為 null 則預設建立 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`	您可以指定下列名稱之一：使用 Microsoft Management Console 或 ONTAP CLI 建立的 SMB 共用名稱，或允許 Trident 建立 SMB 共用的名稱。此參數對於 Amazon FSx for ONTAP 後端是必要的。	`smb-share`

參數	說明	範例
useREST	布林參數，用於使用 ONTAP REST API。當設定為 `true`時，Trident 將使用 ONTAP REST API 與後端通訊。此功能需要 ONTAP 9.11.1 或更新版本。此外，使用的 ONTAP 登入角色必須具有 `ontap`應用程式的存取權限。預先定義的 `vsadmin`和 `cluster-admin`角色可滿足此要求。	false
aws	您可以在 AWS FSx for ONTAP 的組態檔中指定下列項目：- fsxFilesystemID：指定 AWS FSx 檔案系統的 ID。- apiRegion：AWS API 區域名稱。- apikey：AWS API 金鑰。- secretKey：AWS 秘密金鑰。	"" "" ""
credentials	指定要儲存在 AWS Secrets Manager 中的 FSx SVM 認證。- name：包含 SVM 認證的密碼的 Amazon Resource Name (ARN)。- type：設定為 awsarn。如需詳細資訊，請參閱"建立 AWS Secrets Manager 密碼"。	

磁碟區配置的後端組態選項

您可以使用 defaults 配置部分中的這些選項來控制預設配置。例如、請參閱下面的組態範例。

參數	說明	預設
spaceAllocation	LUN 的空間分配	true
spaceReserve	空間保留模式；「 none 」（精簡）或「 volume 」（完整）	none
snapshotPolicy	要使用的 Snapshot 原則	none
qosPolicy	要為建立的磁碟區指派 QoS 策略群組。每個儲存池或後端選擇 qosPolicy 或 adaptiveQosPolicy 其中之一。搭配 Trident 使用 QoS 策略群組需要 ONTAP 9.8 或更新版本。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 策略群組會強制限制所有工作負載的總吞吐量上限。	""

參數	說明	預設
adaptiveQosPolicy	為建立的磁碟區指派的自適應 QoS 原則群組。每個儲存資源池或後端可選擇 qosPolicy 或 adaptiveQosPolicy 其中之一。ontap-nas-economy 不支援。	""
snapshotReserve	為快照保留的磁碟區百分比 "0"	如果 snapshotPolicy`是 `none`，`else`""
splitOnClone	建立時將複本從其父項分割	false
encryption	在新磁碟區上啟用 NetApp Volume Encryption（NVE）；預設值為 false。要使用此選項，叢集必須已獲得 NVE 許可並啟用 NVE。如果後端啟用了 NAE，則在 Trident 中佈建的任何磁碟區都會啟用 NAE。如需詳細資訊，請參閱："Trident 與 NVE 和 NAE 的運作方式"。	false
luksEncryption	啟用 LUKS 加密。請參閱 "使用 Linux Unified Key Setup （ LUKS ）"。僅限 SAN。	""
tieringPolicy	要使用的分層原則 none	
unixPermissions	新磁碟區的模式。**SMB** 磁碟區請保留空白。	""
securityStyle	新磁碟區的安全樣式。NFS 支援 `mixed`和 `unix`安全樣式。SMB 支援 `mixed`和 `ntfs`安全樣式。	NFS 預設值為 unix。SMB 預設值為 ntfs。

配置 **SMB Volume**

您可以使用 ontap-nas 驅動程式來配置 SMB 磁碟區。在完成 ONTAP SAN 和 NAS 驅動程式整合 之前，請先完成以下步驟："準備配置 SMB Volume"

配置儲存等級和 **PVC**

配置 Kubernetes StorageClass 物件並建立儲存類別，以指示 Trident 如何配置磁碟區。建立一個 PersistentVolumeClaim（PVC）來使用已配置的 Kubernetes StorageClass 請求存取 PV。然後，您可以將 PV 掛載到 Pod。

建立儲存類別

配置 **Kubernetes StorageClass** 物件

該 "Kubernetes StorageClass 對象"物件將 Trident 識別為該類別使用的儲存配置器，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（有關完整的屬性列表，請參閱下方的 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"
```

請使用以下範例為使用 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 磁碟區，請將所需內容新增 `mountOptions`至儲存類別：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
  - nfsvers=3
  - nolock
```

請參閱"Kubernetes 和 Trident 物件"以瞭解儲存類別如何與 `PersistentVolumeClaim`互動，以及控制 Trident 配置磁碟區的參數詳細資訊。

建立儲存類別

步驟

1. 這是一個 Kubernetes 物件，因此請使用 `kubectl`在 Kubernetes 中建立它。

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

2. 現在您應該在 Kubernetes 和 Trident 中看到 **basic-csi** 儲存類別，而 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
```

建立 **PVC**

```
https://kubernetes.io/docs/concepts/storage/persistent-
volumes["_PersistentVolumeClaim_"^]（PVC）是對叢集上 PersistentVolume
的存取請求。
```

PVC 可以配置為請求特定大小的儲存空間或存取模式。透過關聯的 StorageClass，叢集管理員不僅可以控制 PersistentVolume 大小和存取模式，還可以控制效能或服務等級等更多參數。

建立 PVC 後，您可以在 pod 中掛載 Volume 。

範例資訊清單

PersistentVolumeClaim 樣本清單

這些範例展示了 PVC 的基本配置選項。

具有 **RWX** 存取權限的 **PVC**

此範例顯示了一個與名為 `basic-csi`的 StorageClass 關聯的具有 RWX 存取權限的基本 PVC。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

使用 **iSCSI** 的 **PVC** 範例

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

```
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 物件"以瞭解儲存類別如何與 `PersistentVolumeClaim`互動，以及控制 Trident 配置磁碟區的參數詳細資訊。

Trident 屬性

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

屬性	類型	價值觀	優惠	要求	支援者
媒體[1]	字串	HDD、混合式、SSD	Pool 包含此類型的媒體；混合型表示兩者兼具	指定的媒體類型	ontap-nas 、ontap-nas-economy 、ontap-nas-flexgroup 、ontap-san 、solidfire-san
provisioningType	字串	薄、厚	資源池支援此佈建方法	已指定佈建方法	thick ：所有 ONTAP ；thin ：所有 ONTAP 和 SolidFire-SAN
backendType	字串	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、solidfire-san 、azure-netapp-files、ontap-san-economy	Pool 屬於這種類型的後端	指定後端	所有驅動程式
快照	布林值	true、false	Pool 支援帶快照的磁碟區	已啟用快照的磁碟區	ontap-nas 、ontap-san 、solidfire-san
複製	布林值	true、false	儲存池支援複製磁碟區	已啟用複本的磁碟區	ontap-nas 、ontap-san 、solidfire-san

屬性	類型	價值觀	優惠	要求	支援者
加密	布林值	true、false	儲存池支援加密磁碟區	已啟用加密的磁碟區	ontap-nas、ontap-nas-economy、ontap-nas-flexgroups、ontap-san
IOPS	int	正整數	Pool 能夠保證此範圍內的 IOPS	Volume 保證了這些 IOPS	solidfire-san

[1]：ONTAP Select 系統不支援

部署範例應用程式

建立儲存類別和 PVC 後，即可將 PV 掛載到 Pod 上。本節列出將 PV 連接到 Pod 的範例命令和組態。

步驟

1. 在 pod 中掛載 volume 。

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

以下範例展示了將 PVC 連接到 pod 的基本組態：基本組態：

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

您可以使用 `kubectl get pod --watch` 監控進度。

2. 確認磁碟區已掛載到 `/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 Pod 了。Pod 應用程式將不復存在，但卷將保留。

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

在 **EKS** 叢集上設定 **Trident EKS** 附加元件

NetApp Trident 簡化了 Kubernetes 中 Amazon FSx for NetApp ONTAP 儲存的管理，讓您的開發人員和管理員能夠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。此 EKS 外掛程式可協助您持續確保 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。

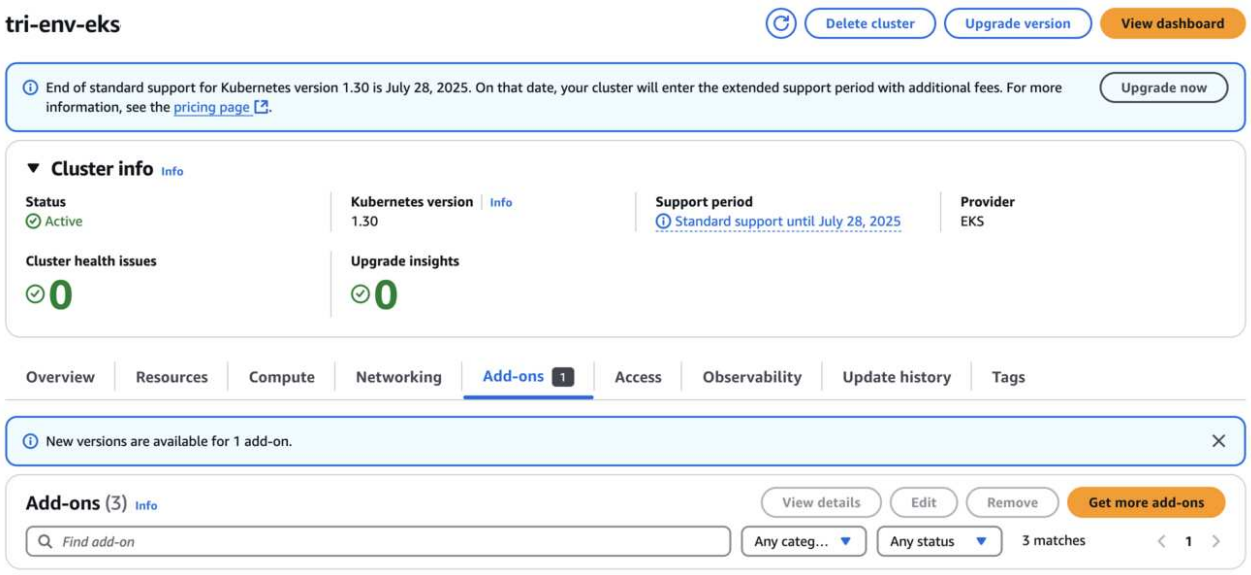
先決條件

在為 AWS EKS 設定 Trident 附加元件之前，請確保您已具備以下條件：

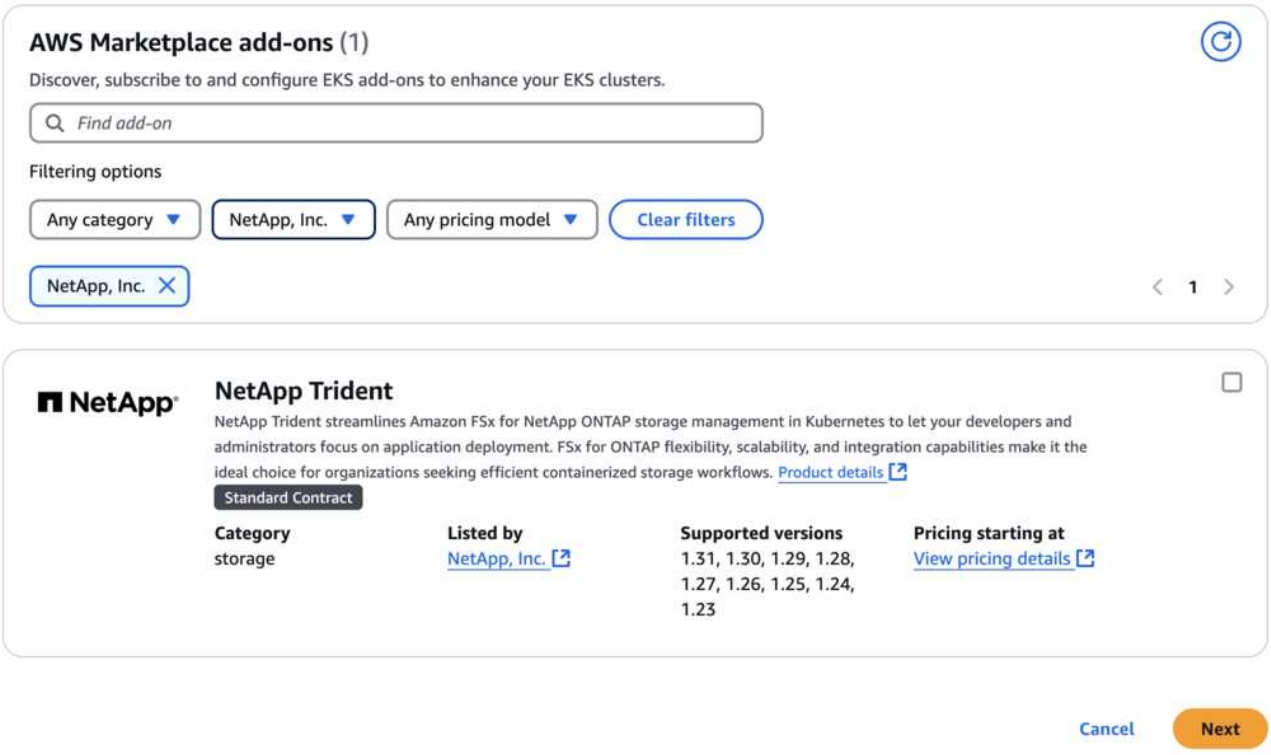
- 具有使用附加元件權限的 Amazon EKS 叢集帳戶。請參閱 "Amazon EKS 附加元件"。
- AWS 對 AWS Marketplace 的權限：
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 類型：Amazon Linux 2（AL2_x86_64）或 Amazon Linux 2 Arm（AL2_ARM_64）
- 節點類型：AMD 或 ARM
- 現有的 Amazon FSx for NetApp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

步驟

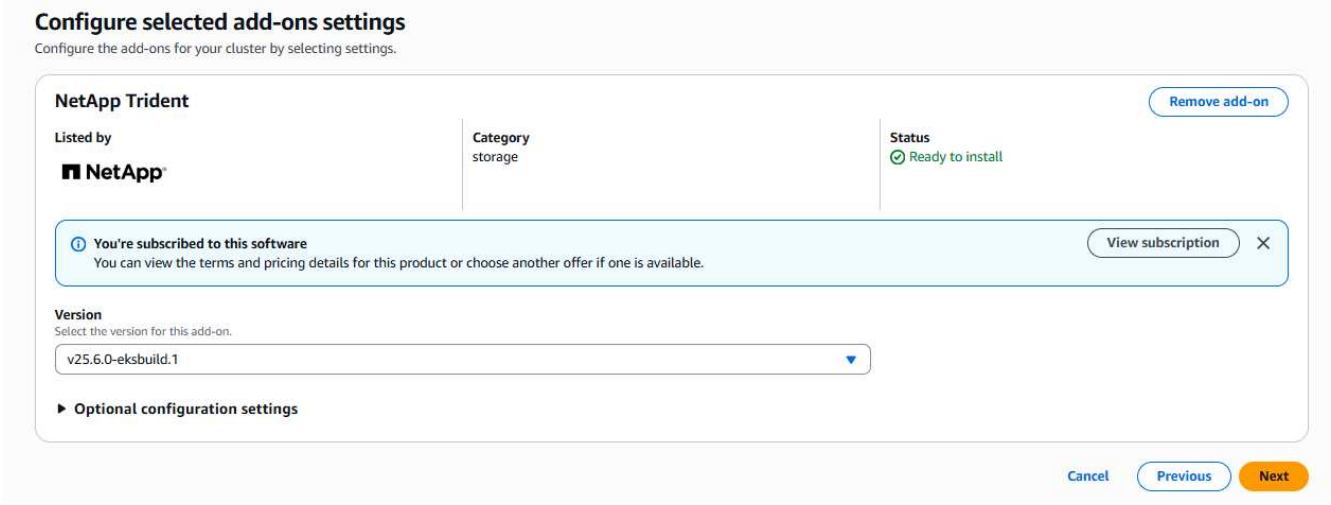
1. 請務必建立 IAM 角色和 AWS 密鑰，以使 EKS Pod 能夠存取 AWS 資源。有關說明，請參閱 "建立 IAM 角色和 AWS Secret"。
2. 在 EKS Kubernetes 叢集上，導覽至 **Add-ons** 標籤。



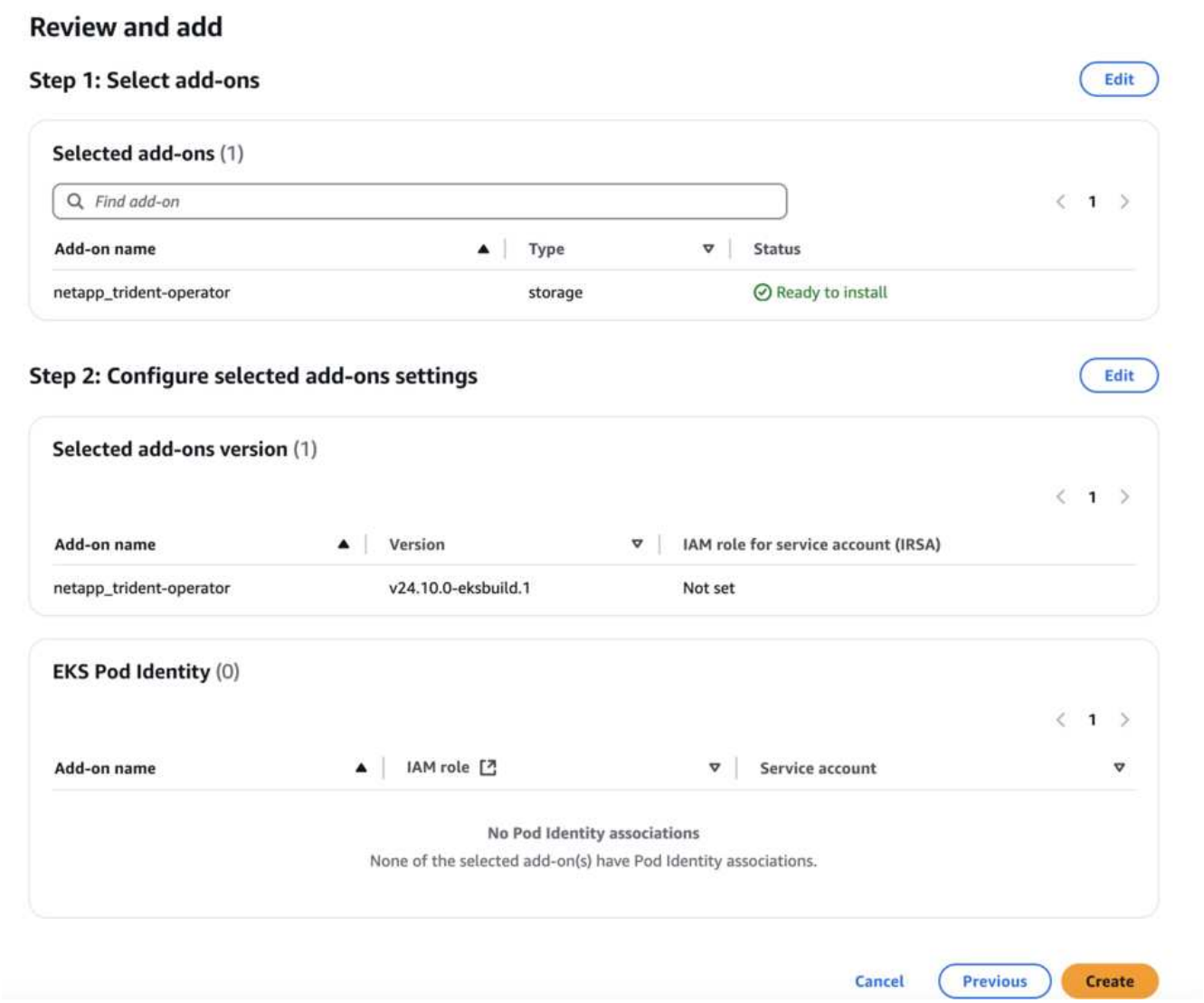
3. 前往 **AWS Marketplace** 外掛程式，然後選擇 *storage* 類別。



4. 找到 **NetApp Trident**，選取 Trident 外掛程式的複選框，然後按一下 **Next**。
5. 選擇所需的附加元件版本。

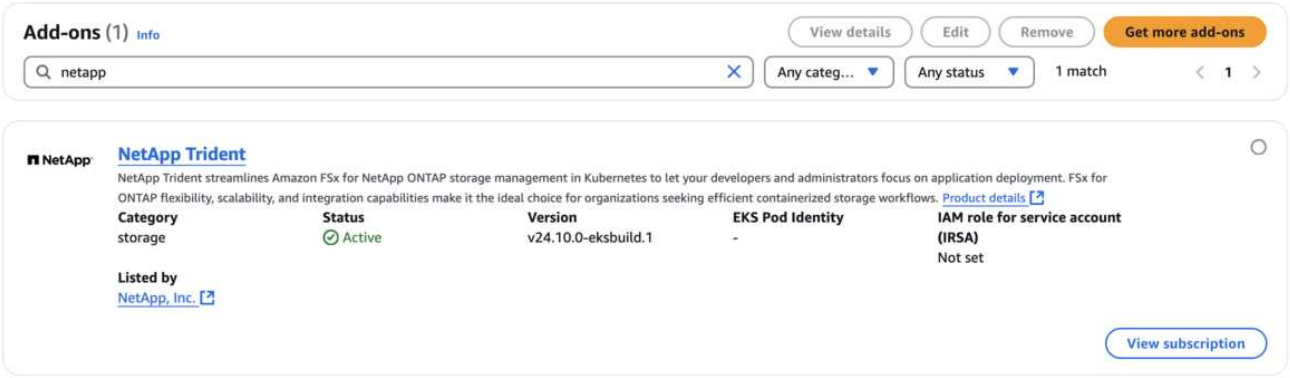


6. 配置所需的附加元件設定。



7. 如果您正在使用 IRSA （服務帳戶的 IAM 角色），請參閱其他組態步驟"這裡"。
8. 選擇 **Create** 。

9. 確認附加元件的狀態為 *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。如需相關資訊、請參閱 "配置儲存後端"。

使用 **CLI** 安裝/解除安裝 **Trident EKS** 附加元件

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

以下範例指令安裝 Trident EKS 外掛程式：

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

以下範例指令安裝 Trident EKS 外掛程式版本 25.6.1：

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.6.1-eksbuild.1（使用專用版本）
```

以下範例指令安裝 Trident EKS 外掛程式版本 25.6.2：

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.6.2-eksbuild.1（使用專用版本）
```

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

以下命令會解除安裝 Trident EKS 附加元件：

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

使用 **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 如何與儲存系統通信，以及 Trident 應該如何從中配置磁碟區。安裝 Trident 後，下一步是建立後端。 `TridentBackendConfig`自訂資源定義（CRD）可讓您直接透過 Kubernetes 介面建立和管理 Trident 後端。您可以使用 `kubectl`或適用於您的 Kubernetes 發行版的等效 CLI 工具來完成此操作。

`TridentBackendConfig`

`TridentBackendConfig` (`tbc tbconfig tbackendconfig`) 是一個前端命名空間 CRD，可讓您使用 `kubectl` 管理 Trident 後端。Kubernetes 和儲存管理員現在可以直接透過 Kubernetes CLI 建立和管理後端，而無需專用的命令列實用程式（(`tridentctl`）。

建立 `TridentBackendConfig` 物件時，會發生以下情況：

- Trident 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。
- `TridentBackendConfig` 唯一綁定到由 Trident 建立的 `TridentBackend`。

每個 `TridentBackendConfig`都與 `TridentBackend`保持一對一的映射關係。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。您*不應該*修改它們。如果您想要更新後端、請透過修改 `TridentBackendConfig`物件來執行此作業。

請參閱以下 `TridentBackendConfig` CR 格式範例：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

您也可以查看 "trident-installer" 目錄中的範例，以了解所需儲存平台/服務的範例配置。

`spec`接受後端特定的組態參數。在本例中、後端使用 `ontap-san`儲存驅動程式、並使用此處表格中列出的組態參數。如需所需儲存驅動程式的組態選項清單、請參閱link:backends.html["儲存驅動程式的後端組態資訊"^]。

`spec` 區段也包含 `credentials` 和 `deletionPolicy` 欄位，這些欄位是在 `TridentBackendConfig` CR 中新引入的：

- `credentials`：此參數為必填欄位，包含用於向儲存系統/服務進行驗證的認證資料。此參數設定為使用者

建立的 Kubernetes Secret。認證資料不能以純文字形式傳遞，否則將導致錯誤。

- `deletionPolicy`：此欄位定義 `TridentBackendConfig` 刪除時應執行的操作。它可以取以下兩個值之一：
  - `delete`：這將導致 `TridentBackendConfig` CR 及其關聯的後端都被刪除。這是預設值。
  - `retain`：當 `TridentBackendConfig` CR 被刪除時，後端定義仍會存在，並可透過 `tridentctl`進行管理。將刪除原則設為 `retain`可讓使用者降級至較早版本（21.04 之前版本）並保留已建立的後端。建立 `TridentBackendConfig`之後，可以更新此欄位的值。

> 後端名稱透過 `spec.backendName` 設定。如果未指定，後端名稱將設定為 `TridentBackendConfig` 物件的名稱（metadata.name）。建議使用 `spec.backendName` 明確設定後端名稱。

> 使用 `tridentctl`建立的後端沒有關聯的 `TridentBackendConfig`物件。您可以選擇透過建立 `TridentBackendConfig` CR，使用 `kubectl`來管理這些後端。必須注意指定相同的組態參數（例如 `spec.backendName`、`spec.storagePrefix`、`spec.storageDriverName`等）。Trident 會自動將新建立的 `TridentBackendConfig`與現有的後端綁定。

步驟概述

要使用 `kubectl`建立新的後端，您應該執行以下操作：

1. 建立 "Kubernetes Secret"。此密鑰包含 Trident 與儲存叢集/服務通訊所需的認證資料。
2. 建立 `TridentBackendConfig`物件。其中包含有關儲存叢集 / 服務的詳細資訊、以及在上一步中建立的機密參照。

建立後端後，您可以使用 `kubectl get tbc <tbc-name> -n <trident-namespace>` 來觀察其狀態並收集更多詳細資訊。

步驟 **1**：建立 **Kubernetes Secret**

建立一個包含後端存取認證的 Secret 。每個儲存服務 / 平台都是唯一的。以下是一個範例：

```
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 Secret 中必須包含的欄位：

儲存平台 **Secret Fields** 描述	機密	欄位說明
Azure NetApp Files	clientID	應用程式註冊中的用戶端 ID
Element（NetApp HCI／SolidFire ）	端點	具有租戶認證資料的 SolidFire 叢集的 MVIP
ONTAP	使用者名稱	用於連接叢集 / SVM 的使用者名稱。用於基於認證的身份驗證
ONTAP	密碼	連接叢集 /SVM 的密碼。用於基於憑證的身份驗證
ONTAP	clientPrivateKey	客戶端私密金鑰的 Base64 編碼值。用於基於憑證的驗證。
ONTAP	chapUsername	入站使用者名稱。如果 useCHAP=true 則為必填項。適用於 `ontap-san`和 `ontap-san-economy`
ONTAP	chapInitiatorSecret	CHAP 發起方金鑰。如果 useCHAP=true ，則為必填項。適用於 `ontap-san`和 `ontap-san-economy`
ONTAP	chapTargetUsername	目標使用者名稱。如果 useCHAP=true，則為必填項。適用於 `ontap-san`和 `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	CHAP 目標啟動器密碼。如果 useCHAP=true 則為必填項。適用於 `ontap-san`和 `ontap-san-economy`

在此步驟中建立的 Secret，將會在下一步建立的 `TridentBackendConfig` 物件的 `spec.credentials` 欄位中被引用。

步驟 **2**：建立 `TridentBackendConfig` **CR**

您現在可以建立 `TridentBackendConfig` CR 了。在本例中，使用 `ontap-san` 驅動程式的後端是使用以下所示的 `TridentBackendConfig` 物件建立的：

```
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 `TridentBackendConfig` CR 的狀態

建立 `TridentBackendConfig` CR 後、您可以驗證其狀態。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。

階段可以採用下列其中一個值：

- `Bound`： `TridentBackendConfig` CR 與後端關聯，且該後端包含 `configRef`設定為 `TridentBackendConfig` CR 的 uid 的值。
- `Unbound`：使用 `""`表示。 `TridentBackendConfig`物件未綁定到後端。所有新建立的 `TridentBackendConfig` CR 預設都處於此階段。階段變更後，無法再恢復為 Unbound 狀態。
- `Deleting`： `TridentBackendConfig` CR 的 `deletionPolicy`被設定為刪除。當 `TridentBackendConfig` CR 被刪除時，它會進入「正在刪除」狀態。
  - 如果後端不存在持久性磁碟區宣告（PVC），則刪除 `TridentBackendConfig`將導致 Trident 刪除後端以及 `TridentBackendConfig`CR。
  - 如果後端存在一個或多個 PVC，則後端進入刪除狀態。 `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 `TridentBackendConfig` CR 關聯的後端被意外或故意刪除，而 `TridentBackendConfig` CR 仍然保留對已刪除後端的參考。無論 `TridentBackendConfig`值為何， `deletionPolicy` CR 仍然可以被刪除。
- `Unknown`:Trident 無法確定與 `TridentBackendConfig` CR 關聯的後端的狀態或是否存在。例如,如果 API 伺服器沒有回應或 `tridentbackends.trident.netapp.io` CRD 缺失。則可能需要介入。

至此，後端已成功創建！還可以處理一些其他操作，例如 "後端更新和後端刪除"。

（選用）步驟 **4**：取得更多詳細資料

您可以執行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 `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`和 `backendUUID`回應 `TridentBackendConfig` CR 而建立的後端。 `lastOperationStatus` 欄位表示 `TridentBackendConfig` CR 最後一次操作的狀態，該操作可以由使用者觸發（例如，使用者變更了 `spec` 中的某些內容）或由 Trident 觸發（例如，在 Trident 重新啟動期間）。其值可以是 Success 或 Failed。 `phase` 表示 `TridentBackendConfig` CR 與後端之間關係的狀態。在上面的範例中， `phase` 的值為 Bound，這意味著 `TridentBackendConfig` CR 已與後端關聯。

您可以執行 `kubectl -n trident describe tbc <tbc-cr-name>` 命令來取得事件日誌的詳細資訊。

> 您無法使用 `TridentBackendConfig`物件透過 `tridentctl`來更新或刪除包含關聯的後端。若要了解在 `tridentctl`與 `TridentBackendConfig`之間切換所需的步驟，請"請看這裡"。

管理後端

使用 **kubectl** 執行後端管理

了解如何使用 `kubectl`執行後端管理操作。

刪除後端

刪除 `TridentBackendConfig`會指示 `Trident` 刪除/保留後端（取決於 `deletionPolicy`）。若要刪除後端，請確保 `deletionPolicy`設定為刪除。若要僅刪除 `TridentBackendConfig`，請確保 `deletionPolicy`設定為保留。這樣可以確保後端仍然存在，並且可以使用 `tridentctl`進行管理。

執行下列命令：

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

Trident 不會刪除 Kubernetes 正在使用的 Secret `TridentBackendConfig`。Kubernetes 用戶負責清理 Secret。刪除 Secret 時必須格外小心。只有當 Secret 不再被後端使用時，才應刪除。

檢視現有的後端

執行下列命令：

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

您也可以執行 `tridentctl get backend -n trident`或 `tridentctl get backend -o yaml -n trident`來取得所有現有後端的清單。此清單還將包括使用 `tridentctl`建立的後端。

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。若要更新憑證,必須更新 `TridentBackendConfig` 物件中使用的 Kubernetes Secret。Trident 將使用提供的最新憑證自動更新後端。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）。
  - 您可以使用以下命令直接透過 Kubernetes 更新 `TridentBackendConfig`物件：

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

  - 或者，您可以使用以下命令對現有 `TridentBackendConfig` CR 進行更改：

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

- 如果後端更新失敗，後端將繼續保持其上次已知的配置。您可以透過執行 `kubectl get tbc <tbc-name> -o yaml -n trident`或 `kubectl describe tbc <tbc-name> -n trident`來查看日誌以確定原因。
- 在您識別並修正組態檔的問題後，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 `tridentctl`執行後端管理操作。

建立後端

建立 "後端組態檔" 後，執行以下命令：

```
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。您可以執行以下命令來檢視記錄以判斷原因：

```
tridentctl logs -n trident
```

在您發現並修正設定檔中的問題後，您可以再次執行 `create`命令。

刪除後端

若要從 Trident 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 已從此後端配置了仍然存在的磁碟區和快照，則刪除該後端將阻止其配置新的磁碟區。該後端將繼續以「正在刪除」狀態存在。

檢視現有的後端

若要檢視 Trident 已知的後端、請執行下列步驟：

- 若要取得摘要、請執行下列命令：

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

- 若要取得所有詳細資料、請執行下列命令：

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

更新後端

建立新的後端組態檔後，執行以下命令：

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

如果後端更新失敗，則表示後端配置存在問題，或者您嘗試了無效的更新操作。您可以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，您可以再次執行 `update` 命令。

識別使用後端的儲存類別

這是一個可以使用 `tridentctl` 為後端物件輸出的 JSON 資料來回答的問題範例。這需要用到 `jq` 實用程式，您需要先安裝該程式。

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

這也適用於使用 `TridentBackendConfig` 建立的後端。

在後端管理選項之間移動

了解在 Trident 中管理後端的不同方式。

管理後端的選項

隨著 `TridentBackendConfig`的推出，管理員現在有兩種獨特的後端管理方式。這引發了以下問題：

- 使用 `tridentctl`建立的後端可以用 `TridentBackendConfig`來管理嗎？
- 使用 `TridentBackendConfig`建立的後端可以使用 `tridentctl`來管理嗎？

使用 `tridentctl`後端管理 `TridentBackendConfig`

本節說明如何通過 Kubernetes 介面直接建立 `tridentctl` 並建立 `TridentBackendConfig` 物件來管理後端的步驟。

這適用於以下情況：

- 已存在的後端，沒有 `TridentBackendConfig`，因為它們是用 `tridentctl`建立的。
- 使用 `tridentctl`建立的新後端，同時存在其他 `TridentBackendConfig`物件。

無論哪種情況，後端都將繼續存在，Trident 負責排程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`來管理使用它建立的後端。
- 將使用 `tridentctl` 建立的後端繫結至新的 `TridentBackendConfig` 物件。這樣做意味著後端將使用 `kubectl` 而非 `tridentctl` 進行管理。

若要使用 `kubectl`管理現有後端，您需要建立一個 `TridentBackendConfig`綁定到現有後端。以下概述了其工作原理：

1. 建立 Kubernetes Secret。此 Secret 包含 Trident 與儲存叢集/服務通訊所需的認證資料。
2. 建立 `TridentBackendConfig`物件。其中包含儲存叢集 / 服務的詳細資訊、以及在上一步中建立的機密參照。必須小心指定相同的組態參數（例如 `spec.backendName`、`spec.storagePrefix`、`spec.storageDriverName`等）。`spec.backendName`必須設定為現有後端的名稱。

步驟 **0**：識別後端

若要建立 `TridentBackendConfig`綁定到現有後端，您需要取得後端配置。在本例中，我們假設後端是使用以下 JSON 定義建立的：

```
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**

下一步是建立 `TridentBackendConfig` CR，使其自動綁定到現有 `ontap-nas-backend`（如本例所示）。請確保滿足以下要求：

- 相同的後端名稱已在 `spec.backendName`中定義。
- 配置參數與原始後端相同。
- 虛擬資源池（如果存在）必須保持與原始後端相同的順序。
- 認證資訊透過 Kubernetes Secret 提供，而非以純文字形式提供。

在這種情況下， `TridentBackendConfig`看起來會像這樣：

```
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 `TridentBackendConfig` **CR** 的狀態

在 `TridentBackendConfig`建立之後，其階段必須為 `Bound`。它還應反映與現有後端相同的後端名稱和 UUID。

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                    BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

現在後端將完全使用 tbc-ontap-nas-backend TridentBackendConfig 物件進行管理。

使用 TridentBackendConfig`後端管理 `tridentctl

`tridentctl` 可用於列出使用 `TridentBackendConfig`
建立的後端。此外，管理員也可以選擇透過 `tridentctl` 完全管理這些後端，方法是刪除
`TridentBackendConfig` 並確保 `spec.deletionPolicy` 設定為 `retain`。

步驟 **0**：識別後端

例如，假設使用 TridentBackendConfig 建立了以下後端：

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

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

從輸出結果可以看出，`TridentBackendConfig`已成功創建，並綁定到後端（觀察後端的 UUID）。

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

讓我們來看一下 `deletionPolicy` 的值。這需要設定為 `retain`。這可確保刪除 `TridentBackendConfig` CR 時，後端定義仍會存在，並可使用 `tridentctl` 進行管理。

```
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 `deletionPolicy`**設定為** `retain`，否則不要進行下一步。

步驟 2：刪除 `TridentBackendConfig` CR

最後一步是刪除 `TridentBackendConfig` CR。確認 `deletionPolicy`設定為 `retain`後，即可執行刪除操作：

```
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 `TridentBackendConfig` 物件時，Trident 只是將其移除，而不會實際刪除後端本身。

建立和管理儲存類別

建立儲存類別

配置 Kubernetes StorageClass 物件並建立儲存類別，以指示 Trident 如何配置磁碟區。

配置 **Kubernetes StorageClass** 物件

此 "Kubernetes StorageClass 對象"識別 Trident 為該類別使用的佈建程式，並指示 Trident 如何佈建磁碟區。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 物件"以瞭解儲存類別如何與 `PersistentVolumeClaim`互動，以及控制 Trident 配置磁碟區的參數詳細資訊。

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] 提供了一些可供使用或修改的基本範例。

步驟

1. 這是一個 Kubernetes 物件，因此請使用 `kubectl`在 Kubernetes 中建立它。

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

2. 現在您應該在 Kubernetes 和 Trident 中看到 **basic-csi** 儲存類別，而 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 "特定後端的簡單儲存類別定義"。

或者、您可以編輯安裝程式隨附的 `sample-input/storage-class-csi.yaml.templ` 檔案、並將 *`BACKEND_TYPE`* 替換為儲存驅動程式名稱。

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

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

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

管理儲存類別

您可以檢視現有儲存類別、設定預設儲存類別、識別儲存類別後端，以及刪除儲存類別。

檢視現有的儲存類別

- 若要檢視現有的 Kubernetes 儲存類別，請執行下列命令：

```
kubectl get storageclass
```

- 若要檢視 Kubernetes 儲存類別詳細資訊，請執行下列命令：

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

- 若要檢視 Trident 的同步儲存類別，請執行下列命令：

```
tridentctl get storageclass
```

- 若要檢視 Trident 的同步儲存類別詳細資料、請執行下列命令：

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

設定預設儲存類別

Kubernetes 1.6 新增了設定預設儲存類別的功能。如果使用者未在持久性磁碟區宣告（PVC）中指定儲存類別，則將使用此儲存類別來配置持久性磁碟區。

- 透過在儲存類別定義中將註解 `storageclass.kubernetes.io/is-default-class` 設為 true 來定義預設儲存類別。根據規範，任何其他值或註解的缺失都將被解釋為 false。
- 您可以使用下列命令將現有儲存類別設定為預設儲存類別：

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

- 同樣地，您可以使用下列命令移除預設儲存類別註釋：

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

Trident 安裝程式套件中也有包含此註解的範例。

叢集中一次只能存在一個預設儲存類別。Kubernetes 技術上並不會禁止存在多個預設儲存類別，但它的行為會如同根本沒有預設儲存類別一樣。

識別儲存類別的後端

這是一個可以使用 `tridentctl` 為 Trident 後端物件輸出的 JSON 資料來回答的問題範例。此範例使用了 `jq` 實用程式，您可能需要先安裝該程式。

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

刪除儲存類別

若要從 Kubernetes 中刪除儲存類別，請執行以下命令：

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

`<storage-class>` 應該替換成您的儲存類別。

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 會為其建立的磁碟區強制執行空白 `fsType`。對於 iSCSI 後端、建議在 StorageClass 中強制執行 `parameters.fsType`。您應該刪除現有的 StorageClasses 並使用指定的 `parameters.fsType`重新建立它們。

配置和管理磁碟區

配置磁碟區

建立一個 PersistentVolumeClaim（PVC），使用已配置的 Kubernetes StorageClass 來請求存取 PV。然後，您可以將 PV 掛載到 Pod 中。

概況

`https://kubernetes.io/docs/concepts/storage/persistent-volumes["_PersistentVolumeClaim_"^]（PVC）是對叢集上 PersistentVolume 的存取請求。`

PVC 可以配置為請求特定大小的儲存空間或存取模式。透過關聯的 StorageClass，叢集管理員不僅可以控制 PersistentVolume 大小和存取模式，還可以控制效能或服務等級等更多參數。

建立 PVC 之後，您可以在 Pod 中掛載磁碟區。

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    1Gi        RWO                   5m
```

1. 在 pod 中掛載 volume 。

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

> 您可以使用 `kubectl get pod --watch` 監控進度。

2. 確認磁碟區已掛載到 `/my/mount/path`。

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

3. 現在您可以刪除 Pod 了。Pod 應用程式將不復存在，但卷將保留。

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

範例資訊清單

PersistentVolumeClaim 樣本清單

這些範例展示了 PVC 的基本配置選項。

具有 RWO 存取權限的 PVC

此範例顯示了一個與名為 `basic-csi` 的 StorageClass 關聯的具有 RWO 存取權限的基本 PVC。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC 與 NVMe/TCP

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

```
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 pod 的基本組態。

基本組態

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

基本 NVMe/TCP 組態

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

請參閱"Kubernetes 和 Trident 物件"以瞭解儲存類別如何與 `PersistentVolumeClaim`互動，以及控制 Trident 配置磁碟區的參數詳細資訊。

擴充磁碟區

Trident 為 Kubernetes 使用者提供了在建立磁碟區後擴充磁碟區的功能。尋找有關擴充 iSCSI、NFS、SMB、NVMe/TCP 和 FC 磁碟區所需的組態資訊。

擴充 **iSCSI Volume**

您可以使用 CSI 配置程式擴充 iSCSI 持續性磁碟區（PV）。

> ⓘ iSCSI 磁碟區擴充受 `ontap-san`、 `ontap-san-economy`、 `solidfire-san` 驅動程式支援，並且需要 Kubernetes 1.16 及更高版本。

步驟 **1**：設定 **StorageClass** 以支援磁碟區擴充

編輯 StorageClass 定義，將 `allowVolumeExpansion` 欄位設為 `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 `allowVolumeExpansion` 參數。

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` 以反映新的所需尺寸，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 會建立持續磁碟區（PV），並將其與此持續磁碟區宣告（PVC）建立關聯。

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

步驟 3：定義連接 PVC 的 pod

將 PV 連接到 Pod 以調整其大小。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 Volume、重新掃描裝置並調整檔案系統大小。
- 嘗試調整未掛載的 PV 的大小時，Trident 會在儲存後端擴充該磁碟區。PVC 綁定到 Pod 後，Trident 會重新掃描裝置並調整檔案系統的大小。擴充操作成功完成後，Kubernetes 會更新 PVC 的大小。

在此範例中、會建立使用 `san-pvc` 的 Pod 。

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

步驟 4：展開 PV

若要將已建立的 PV 的大小從 1Gi 調整為 2Gi，請編輯 PVC 定義並將 `spec.resources.requests.storage` 更新為 2Gi。

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

步驟 **5**：驗證擴充

您可以檢查 PVC、PV 和 Trident Volume 的大小，以驗證擴充是否正常運作：

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

擴充 FC Volume

您可以使用 CSI 設定程式擴充 FC Persistent Volume（PV）。

FC Volume 擴充受 `ontap-san` 驅動程式支援，需要 Kubernetes 1.16 及更高版本。

步驟 1：設定 StorageClass 以支援磁碟區擴充

編輯 StorageClass 定義，將 `allowVolumeExpansion` 欄位設為 `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 `allowVolumeExpansion` 參數。

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` 以反映新的所需尺寸，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 會建立持續磁碟區（PV），並將其與此持續磁碟區宣告（PVC）建立關聯。

```
kubectl get pvc
NAME       STATUS    VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc    Bound     pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

步驟 3：定義連接 PVC 的 pod

將 PV 附加至 Pod 以調整其大小。調整 FC 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 Volume、重新掃描裝置並調整檔案系統大小。
- 嘗試調整未掛載的 PV 的大小時，Trident 會在儲存後端擴充該磁碟區。PVC 綁定到 Pod 後，Trident 會重新掃描裝置並調整檔案系統的大小。擴充操作成功完成後，Kubernetes 會更新 PVC 的大小。

在此範例中、會建立使用 `san-pvc` 的 Pod 。

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

步驟 4：展開 PV

若要將已建立的 PV 的大小從 1Gi 調整為 2Gi，請編輯 PVC 定義並將 `spec.resources.requests.storage` 更新為 2Gi。

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

步驟 **5**：驗證擴充

您可以檢查 PVC、PV 和 Trident Volume 的大小，以驗證擴充是否正常運作：

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

擴充 NFS Volume

Trident 支援在 `ontap-nas`、 `ontap-nas-economy`、 `ontap-nas-flexgroup`和 `azure-netapp-files`後端上配置的 NFS PV 的磁碟區擴充。

步驟 1：設定 StorageClass 以支援磁碟區擴充

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，方法是將 `allowVolumeExpansion`欄位設為 `true`：

```
cat storageclass-ontapnas.yaml
```

```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別，您可以簡單地使用 `kubectl edit storageclass` 編輯現有儲存類別以允許磁碟區擴充。

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 應該為此 PVC 建立 20 MiB NFS PV：

```
kubectl get pvc
NAME           STATUS    VOLUME
CAPACITY     ACCESS MODES    STORAGECLASS    AGE
ontapnas20mb   Bound     pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

步驟 3：展開 PV

若要將新建的 20 MiB PV 調整為 1 GiB，請編輯 PVC 並將 `spec.resources.requests.storage` 設為 1 GiB：

```
kubectl edit pvc ontapnas20mb
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
# ...
```

步驟 **4**：驗證擴充

您可以檢查 PVC、PV 和 Trident Volume 的大小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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
```

匯入磁碟區

您可以使用 `tridentctl import`或透過使用 Trident 匯入註解建立持久性磁碟區宣告（PVC），將現有儲存磁碟區匯入為 Kubernetes PV。

概述和注意事項

您可以將磁碟區匯入 Trident 以：

- 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複本
- 重建失敗的 Kubernetes 叢集
- 在災難復原期間遷移應用程式資料

考量事項

在匯入磁碟區之前，請檢閱下列考量事項。

- Trident 只能匯入 RW（讀寫）類型的 ONTAP 磁碟區。DP（資料保護）類型的磁碟區是 SnapMirror 目標磁碟區。在將磁碟區匯入 Trident 之前，您應該中斷鏡射關係。

- 我們建議匯入沒有作用中連線的磁碟區。若要匯入正在使用的磁碟區，請複製該磁碟區，然後執行匯入。

  這一點對於區塊磁碟區尤其重要，因為 Kubernetes 無法感知先前的連線，很容易將作用中磁碟區附加到 Pod 上。這可能導致資料毀損。

- 雖然 `StorageClass`必須在 PVC 上指定，但 Trident 在匯入期間不會使用此參數。儲存類別用於在建立磁碟區時根據儲存特性從可用的儲存池中進行選擇。由於磁碟區已存在，因此在匯入期間不需要選擇儲存池。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儲存池上，匯入也不會失敗。
- 現有磁碟區的大小在 PVC 中決定和設定。儲存驅動程式匯入磁碟區後，會建立 PV 並將其 ClaimRef 與 PVC 關聯起來。
  - 回收原則最初在 PV 中設定為 `retain`。在 Kubernetes 成功繫結 PVC 和 PV 後，回收原則會更新為與 Storage Class 的回收原則相符。
  - 如果 Storage Class 的回收策略為 `delete`，則當 PV 被刪除時，儲存磁碟區也會被刪除。
- 預設情況下， Trident 會管理 PVC ，並在後端重新命名 FlexVol Volume 和 LUN 。您可以傳遞 `--no-manage`旗標來匯入非託管 Volume ，以及 `--no-rename`旗標來保留 Volume 名稱。
  - `--no-manage`* - 如果使用 `--no-manage`標誌，Trident 在物件生命週期內不會對 PVC 或 PV 執行任何其他操作。刪除 PV 時，儲存磁碟區不會被刪除，其他操作（例如磁碟區複製和磁碟區調整大小）也會被忽略。
  - `--no-rename`* - 如果使用 `--no-rename` 標誌、Trident 會在匯入磁碟區時保留現有磁碟區名稱、並管理磁碟區的生命週期。此選項僅支援 `ontap-nas`、 `ontap-san`（包括 ASA r2 系統）和 `ontap-san-economy` 驅動程式。

    如果您想使用 Kubernetes 進行容器化工作負載，但又想在 Kubernetes 之外管理儲存磁碟區的生命週期，那麼這些選項非常有用。

- PVC 和 PV 會新增一個註釋，其作用有兩個：一是指示該磁碟區已匯入，二是指示 PVC 和 PV 是否受管理。此註釋不應修改或刪除。

匯入磁碟區

您可以使用 `tridentctl import` 或透過建立具有 Trident 匯入註解的 PVC 來匯入磁碟區。

如果使用 PVC 註釋，則無需下載或使用 `tridentctl`匯入磁碟區。

使用 **tridentctl**

步驟

1. 建立 PVC 檔案（例如 `pvc.yaml`），用於建立 PVC。PVC 檔案應包含 `name`、 `namespace`、 `accessModes`和 `storageClassName`。您也可以在 PVC 定義中指定 `unixPermissions`。

   以下是最低規格範例：

   ```
   kind: PersistentVolumeClaim
   apiVersion: v1
   metadata:
     name: my_claim
     namespace: my_namespace
   spec:
     accessModes:
       - ReadWriteOnce
     storageClassName: my_storage_class
   ```

   僅包含必需參數。其他參數（例如 PV 名稱或磁碟區大小）可能會導致匯入命令失敗。

2. 使用 `tridentctl import` 指令指定包含磁碟區的 Trident 後端名稱以及儲存上唯一標識該磁碟區的名稱（例如：ONTAP FlexVol、Element Volume）。 `-f` 參數是必需的，用於指定 PVC 檔案的路徑。

   ```
   tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
   ```

使用 **PVC** 註釋

步驟

1. 建立一個 PVC YAML 檔案（例如， `pvc.yaml`），其中包含所需的 Trident 匯入註解。PVC 檔案應包含：

   - `name` 和 `namespace` 在中繼資料中
   - `accessModes`、 `resources.requests.storage` 和 `storageClassName` 在規格中
   - 註釋：
     - `trident.netapp.io/importOriginalName`：後端的磁碟區名稱
     - `trident.netapp.io/importBackendUUID`：磁碟區所在的後端 UUID
     - `trident.netapp.io/notManaged`（可選）：設定為 `"true"`表示非託管磁碟區。預設值為 `"false"`。

       以下是匯入託管磁碟區的範例規格：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. 將 PVC YAML 檔案套用到您的 Kubernetes 叢集：

```
kubectl apply -f <pvc-file>.yaml
```

Trident 會自動匯入磁碟區並將其繫結至 PVC。

範例

請查看以下磁碟區匯入範例，以了解支援的驅動程式。

ONTAP NAS 和 ONTAP NAS FlexGroup

Trident 支援使用 `ontap-nas` 和 `ontap-nas-flexgroup` 驅動程式進行磁碟區匯入。

- Trident 不支援使用 `ontap-nas-economy` 驅動程式進行磁碟區匯入。
- `ontap-nas` 和 `ontap-nas-flexgroup` 驅動程式不允許重複的磁碟區名稱。

使用 `ontap-nas`驅動程式建立的每個磁碟區都是 `ONTAP` 叢集上的 `FlexVol` 磁碟區。使用 `ontap-nas`驅動程式匯入 `FlexVol` 磁碟區的操作方式相同。`ONTAP` 叢集上已存在的 `FlexVol` 磁碟區可以作為 `ontap-nas` PVC 匯入。同樣、FlexGroup 磁碟區也可以作為 `ontap-nas-flexgroup` PVC 匯入。

使用 **tridentctl** 的 **ONTAP NAS** 範例

以下範例展示如何使用 `tridentctl`匯入託管磁碟區和非託管磁碟區。

託管磁碟區

以下範例匯入名為 `managed_volume`的 Volume，該 Volume 位於名為 `ontap_nas`的後端：

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

未託管磁碟區

使用 `--no-manage` 參數時、Trident 不會重新命名磁碟區。

以下範例會在 `ontap_nas` 後端匯入 `unmanaged_volume`：

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-
file> --no-manage

+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
```

使用 **PVC** 註解的 **ONTAP NAS** 範例

以下範例展示如何使用 PVC 註解匯入託管和非託管磁碟區。

託管磁碟區

以下範例從後端 `81abcb27-ea63-49bb-b606-0a5315ac5f21`匯入一個名為 `ontap_volume1`的 1GiB `ontap-nas` 磁碟區，並使用 PVC 註解設定了 RWO 存取模式：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

未託管磁碟區

以下範例從後端 `34abcb27-ea63-49bb-b606-0a5315ac5f34`匯入名為 `ontap-volume2`的 1Gi `ontap-nas` 磁碟區，並使用 PVC 註解設定 RWO 存取模式：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

ONTAP SAN

Trident 支援使用 `ontap-san`（iSCSI、NVMe/TCP 和 FC）及 `ontap-san-economy` 驅動程式進行磁碟區匯入。

Trident 可以匯入包含單一 LUN 的 ONTAP SAN FlexVol Volume 。這與 `ontap-san`驅動程式一致、此驅動程式會為每個 PVC 建立一個 FlexVol Volume 、並在 FlexVol Volume 內建立一個 LUN 。Trident 會匯入 FlexVol Volume 並將其與 PVC 定義建立關聯。Trident 可以匯入 `ontap-san-economy`包含多個 LUN 的 Volume 。

以下範例展示如何匯入託管和非託管磁碟區：

託管磁碟區

對於託管磁碟區、 Trident 會將 FlexVol 磁碟區重新命名為 `pvc-<uuid>`格式、並將 `FlexVol` 磁碟區內的 `LUN` 重新命名為 `lun0。

以下範例匯入 `ontap-san-managed` FlexVol volume，該磁碟區存在於 `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    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

未託管磁碟區

以下範例會在 `unmanaged_example_volume` `ontap_san` 後端匯入：

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

如果您將 LUN 對應到與 Kubernetes 節點 IQN 共用相同 IQN 的 igroup，如下例所示，您將收到以下錯誤：`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 `solidfire-san`驅動程式匯入 NetApp Element 軟體和 NetApp HCI 磁碟區。

Element 驅動程式支援重複的磁碟區名稱。但是，如果存在重複的磁碟區名稱，Trident 會傳回錯誤。因應措施是複製磁碟區、提供唯一的磁碟區名稱，然後匯入複製的磁碟區。

以下範例會在後端 `element_default` 匯入一個 `element-managed` volume。

```
tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
```

Azure NetApp Files

Trident 支援使用 `azure-netapp-files` 驅動程式導入磁碟區。

若要匯入 Azure NetApp Files 磁碟區，請透過磁碟區路徑識別該磁碟區。磁碟區路徑是磁碟區匯出路徑中 `:/`後的部分。例如，如果掛載路徑為 `10.0.0.2:/importvol1`，則磁碟區路徑為 `importvol1`。

以下範例會在後端 `azurenetappfiles_40517` 匯入一個 `azure-netapp-files` volume，並使用 volume 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 `google-cloud-netapp-volumes` 驅動程式導入磁碟區。

以下範例使用磁碟區 `testvoleasiaeast1` 從後端 `backend-tbc-gcnv1` 匯入磁碟區。

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

以下範例示範如何在同一區域存在兩個磁碟區時匯入 `google-cloud-netapp-volumes`磁碟區：

```
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。這有助於您識別磁碟區並將其輕鬆對應到相應的 Kubernetes 資源（ PVC ）。您也可以在後端定義模板，用於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：

- Volume 建立、匯入和複製作業。
- 對於 `ontap-nas-economy` 驅動程式而言，只有 Qtree 磁碟區的名稱符合名稱範本。
- 對於 `ontap-san-economy` 驅動程式而言，只有 LUN 名稱符合名稱範本。

限制

- 自訂磁碟區名稱僅與 ONTAP 內部部署驅動程式相容。
- 僅 `ontap-san`、 `ontap-nas` 和 `ontap-nas-flexgroup` 驅動程式支援自訂標籤。
- 自訂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

- 如果由於名稱範本中的語法無效而導致故障，則後端建立將失敗。但是，如果範本應用程式失敗，則磁碟區將根據現有的命名慣例命名。
- 當使用後端組態中的名稱範本命名磁碟區時，儲存前置字元不適用。任何所需的前置字元值都可以直接新增

至範本。

後端組態範例、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。

根層級範例

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

資源池層級範例

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

名稱範本範例

範例 **1**：

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

範例 **2**：

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

需要考慮的要點

1. 對於磁碟區匯入，僅當現有磁碟區的標籤採用特定格式時，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中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置字元一起使用。
4. 如果範本無法產生唯一的磁碟區名稱，Trident 將附加一些隨機字元以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 個字元，Trident 將依照現有的命名規則命名磁碟區。對於所有其他 ONTAP 驅動程式，如果磁碟區名稱超過名稱限制，則磁碟區建立程序將會失敗。

跨命名空間共享 **NFS Volume**

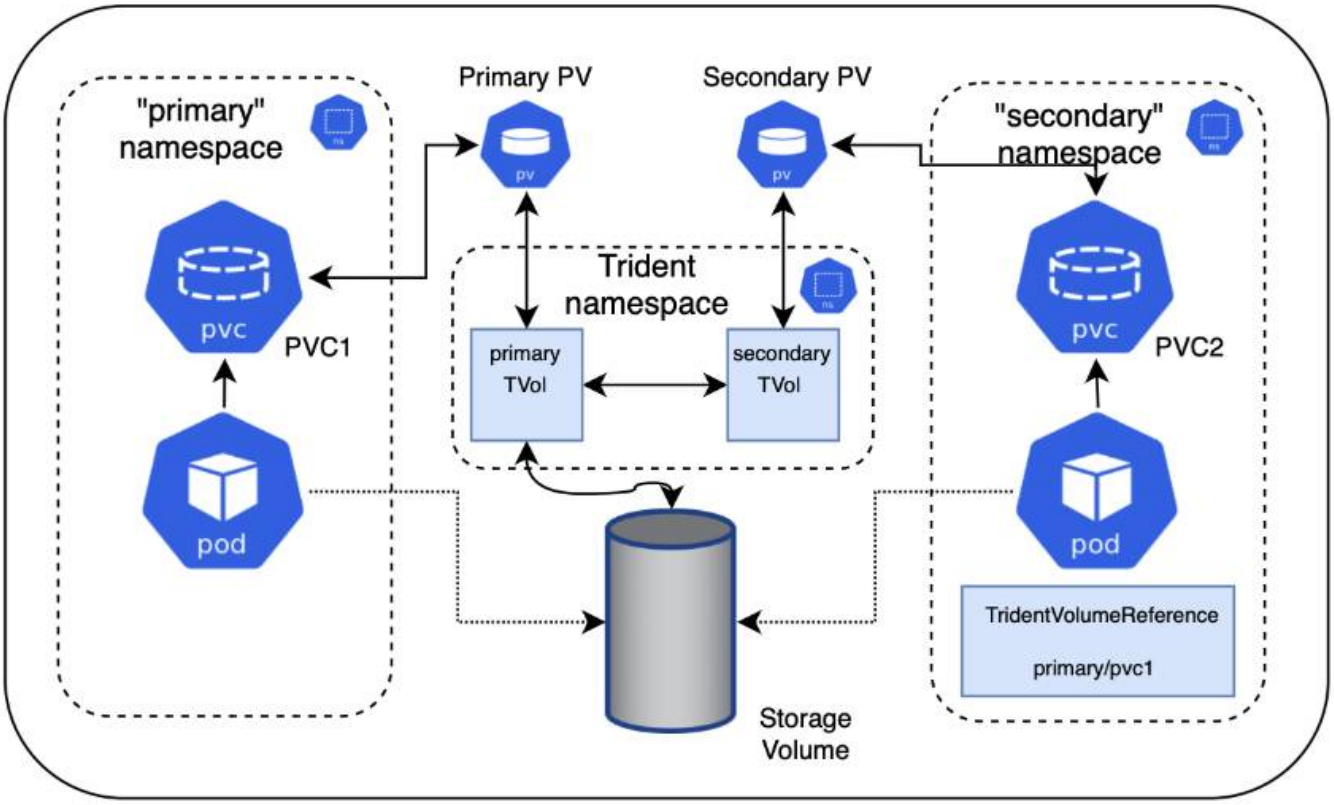
使用 Trident、您可以在主命名空間中建立磁碟區、並在一個或多個次要命名空間中共用該磁碟區。

功能

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勢：

- 多層級存取控制以確保安全性
- 適用於所有 Trident NFS Volume 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。



快速入門

只需幾個步驟即可設定 NFS 磁碟區共用。

1 配置來源 **PVC** 以共用磁碟區

來源命名空間擁有者授予存取來源 PVC 中資料的權限。

2 授予在目標命名空間中建立 **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。

設定來源和目的地命名空間

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操作。每個步驟都會指定使用者角色。

步驟

1. 來源命名空間擁有者： 在來源命名空間中建立 PVC(`pvc1`，該 PVC 授予與目標命名空間(`namespace2`共享的權限，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 會建立 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 `*` 共用到所有命名空間。例如，`trident.netapp.io/shareToNamespace: *`
- 您可以隨時更新 PVC 以包含 `shareToNamespace`註釋。

2. 叢集管理員： 確保已部署適當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。
3. 目標命名空間擁有者： 在目標命名空間中建立一個 TridentVolumeReference CR，引用來源命名空間 `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： 在目標命名空間 (`namespace2`）中建立一個 PVC (`pvc2`），並使用 `shareFromPVC` 註解來指定來源 PVC。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

目的地 PVC 的大小必須小於或等於來源 PVC。

結果

Trident 讀取 `shareFromPVC`目標 PVC 上的註解，並將目標 PV 建立為從屬磁碟區，該磁碟區本身沒有儲存資源，指向來源 PV 並共用來源 PV 的儲存資源。目標 PVC 和 PV 看起來就像正常綁定一樣。

刪除共享磁碟區

您可以刪除跨多個命名空間共享的磁碟區。Trident 將移除來源命名空間對該磁碟區的存取權限，並保留其他共用該磁碟區的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移除後，Trident 才會刪除該磁碟區。

使用 **`tridentctl get`**查詢從屬磁碟區

使用 [`tridentctl` 公用程式、您可以執行 `get` 命令來取得從屬磁碟區。如需詳細資訊、請參閱 `tridentctl` 命令和選項。

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

標記：

- `-h, --help`：有關磁碟區的說明。
- `--parentOfSubordinate string`：將查詢限制在從屬來源磁碟區。
- `--subordinateOf string`：將查詢限制在 Volume 的下級。

限制

- Trident 無法阻止目的地命名空間寫入共用磁碟區。您應該使用檔案鎖定或其他程序來防止覆寫共用磁碟區資料。

- 您無法透過移除 `shareToNamespace` 或 `shareFromNamespace` 註解或刪除 `TridentVolumeReference` CR 來撤銷對來源 PVC 的存取權限。若要撤銷存取權限，您必須刪除從屬 PVC。
- 從屬磁碟區無法進行 Snapshot、複本和鏡射操作。

如需更多資訊

若要深入瞭解跨命名空間磁碟區存取：

- 訪問 "在命名空間之間共享磁碟區：迎接跨命名空間磁碟區存取"。
- 觀看 "NetAppTV" 上的示範影片。

跨命名空間複製磁碟區

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

在複製磁碟區之前、請確保來源和目的地後端的類型相同、且具有相同的儲存類別。

跨命名空間複製僅支援 `ontap-san` 和 `ontap-nas` 儲存驅動程式。不支援唯讀複製。

快速入門

只需幾個步驟即可設定磁碟區複製。

1. 配置來源 **PVC** 以複製磁碟區

來源命名空間擁有者授予存取來源 PVC 中資料的權限。

2. 授予在目標命名空間中建立 **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複製 **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 。

設定來源和目的地命名空間

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。每個步驟都會指定使用者角色。

步驟

1. 來源命名空間擁有者： 在來源命名空間(`namespace1`)中建立 PVC(`pvc1`)，並使用 `cloneToNamespace`` 註解授權與目標命名空間 (`namespace2`)共享。

```
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 `*` 共用到所有命名空間。例如，`trident.netapp.io/cloneToNamespace: *`
- 您可以隨時更新 PVC 以包含 `cloneToNamespace`註釋。

2. 叢集管理員： 確保已配置正確的 RBAC，以授予目標命名空間擁有者在目標命名空間中建立 TridentVolumeReference CR 的權限(`namespace2`）。
3. 目標命名空間擁有者： 在目標命名空間中建立一個 TridentVolumeReference CR，引用來源命名空間 `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： 在目標命名空間 (`namespace2`）中建立一個 PVC (`pvc2`），使用 `cloneFromPVC` 或 `cloneFromSnapshot`，以及 `cloneFromNamespace` 註解來指定來源 PVC。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

限制

- 對於使用 ontap-nas-economy 驅動程式配置的 PVC、不支援唯讀複本。

使用 SnapMirror 複製磁碟區

Trident 支援在叢集上的來源磁碟區和對等叢集上的目標磁碟區之間建立鏡像關係，以複製資料進行災難復原。 您可以使用名為 Trident Mirror Relationship（TMR）的命名空間自訂資源定義（CRD）來執行下列操作：

- 在磁碟區（PVC）之間建立鏡像關係
- 移除磁碟區之間的鏡射關係
- 打破鏡像關係
- 在災難情況下（容錯移轉）提升次要磁碟區
- 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

複寫的前提條件

在開始之前、請確保符合下列先決條件：

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 或更新版本。
- 授權：必須在來源和目的地 ONTAP 叢集上啟用使用資料保護套件的 ONTAP SnapMirror 非同步授權。如需詳細資訊，請參閱 "SnapMirror 授權總覽（ONTAP）"。

從 ONTAP 9.10.1 開始，所有授權均以 NetApp 授權檔案（NLF）的形式提供，這是一個可啟用多項功能的單一檔案。如需詳細資訊，請參閱 "ONTAP One 隨附的授權"。

僅支援 SnapMirror 非同步保護。

對等

- 叢集和 **SVM**：ONTAP 儲存後端必須建立對等連線。如需詳細資訊、請參閱 "叢集和 SVM 對等連接概述"。

確保兩個 ONTAP 叢集之間複寫關係中使用的 SVM 名稱是唯一的。

- **Trident** 和 **SVM**：對等遠端 SVM 必須可供目的地叢集上的 Trident 使用。

支援的驅動程式

NetApp Trident 支援使用下列驅動程式支援的儲存類別、透過 NetApp SnapMirror 技術進行磁碟區複寫：**`ontap-nas`**：**NFS** `ontap-san`：iSCSI **`ontap-san`**：**FC** `ontap-san`：NVMe/TCP（需要最低 ONTAP 版本 9.15.1）

ASA r2 系統不支援使用 SnapMirror 的磁碟區複寫。如需 ASA r2 系統的相關資訊、請參閱"了解 ASA r2 儲存系統"。

建立鏡射 **PVC**

請依照這些步驟並使用 CRD 範例，在主磁碟區和次要磁碟區之間建立鏡像關係。

步驟

1. 在主要 Kubernetes 叢集上執行下列步驟：
   a. 建立一個帶有 `trident.netapp.io/replication: true` 參數的 StorageClass 物件。

      範例

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. 使用先前建立的 StorageClass 建立 PVC。

範例

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

c. 建立包含本機資訊的 MirrorRelationship CR。

範例

```
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. 在次要 Kubernetes 叢集上執行下列步驟：
   a. 建立 StorageClass 並設定 trident.netapp.io/replication: true 參數。

      範例

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        trident.netapp.io/replication: true
      ```

   b. 建立包含目標和來源資訊的 MirrorRelationship CR。

      範例

      ```
      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 將使用已設定的關係原則名稱（或 ONTAP 的預設值）建立 SnapMirror 關係並將其初始化。

c. 建立一個 PVC，將先前建立的 StorageClass 作為輔助（SnapMirror 目標）。

範例

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Trident 將檢查 TridentMirrorRelationship CRD，如果關係不存在，則建立磁碟區失敗。如果關係存在，Trident 將確保將新 FlexVol 磁碟區放置在與 MirrorRelationship 中定義的遠端 SVM 對等的 SVM 上。

Volume 複寫狀態

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，該狀態告知 Trident 所需的狀態。目標 TMR 具有以下狀態：

- 已建立：本地 PVC 是鏡像關係的目標磁碟區，這是一個新的關係。
- **Promoted**：本機 PVC 為 ReadWrite 且可掛載，目前沒有鏡像關係生效。
- 重新建立：本機 PVC 是鏡像關係的目的地 Volume，且先前也處於該鏡像關係中。
  - 如果目標 volume 曾經與來源 volume 有關聯，則必須使用重新建立的狀態，因為它會覆寫目標 volume 的內容。
  - 如果磁碟區之前未與來源建立關係，則重新建立的狀態將會失敗。

在非計劃性容錯移轉期間提升次要 **PVC**

在次要 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次要 **PVC**

在計劃性容錯移轉（移轉）期間，執行以下步驟以提升次要 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。

   範例

   ```
   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 internalName。
4. 在輔助 Kubernetes 叢集上，確認 TridentMirrorRelationship 的狀態（status.state 欄位）是否已提升。

故障轉移後還原鏡像關係

在還原鏡射關係之前、請選擇您要做為新主要端的一端。

步驟

1. 在輔助 Kubernetes 叢集上，確保 TridentMirrorRelationship 上 *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`。

其他作業

Trident 支援對主要磁碟區和次要磁碟區執行下列操作：

將主要 PVC 複寫到新的次要 PVC

請確保您已備有主要 PVC 和次要 PVC。

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、主要或次要 PVC 的大小

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 FlexVol。

從 PVC 移除複寫

若要移除複寫，請對目前的次要磁碟區執行下列其中一項作業：

- 刪除輔助 PVC 上的 MirrorRelationship。這將破壞複寫關係。

- 或者，將 spec.state 欄位更新為 *promoted* 。

刪除一個 **PVC**（之前已鏡像）

Trident 會檢查是否有複寫的 PVC，並在嘗試刪除磁碟區之前釋放複寫關係。

刪除 **TMR**

刪除鏡像關係一側的 TMR 會導致剩餘的 TMR 在 Trident 完成刪除作業之前轉換為 *promoted* 狀態。如果選擇刪除的 TMR 已處於 *promoted* 狀態，表示不存在鏡像關係，此時 TMR 將被移除，Trident 會將本機 PVC 提升為 *ReadWrite* 狀態。此刪除操作會釋放 ONTAP 中本機磁碟區的 SnapMirror 中繼資料。如果將來在鏡像關係中使用此磁碟區，則在建立新鏡像關係時必須使用狀態為 *established* 的磁碟區複寫新 TMR。

ONTAP 上線時更新鏡像關係

鏡像關係建立後可隨時更新。您可以使用 `state: promoted`或 `state: reestablished`欄位來更新關係。將目標磁碟區提升為常規 ReadWrite 磁碟區時、您可以使用 *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 範例格式：

範例

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` 反映 TridentActionMirrorUpdate CRD 的狀態。它可以取值於 *Succeeded*、*In Progress* 或 *Failed*。

使用 **CSI** 拓撲

Trident 可以利用 "CSI Topology 功能"選擇性地建立磁碟區並將其附加到 Kubernetes 叢集中的節點。

概況

使用 CSI Topology 功能，可以根據區域和可用區將磁碟區的存取權限限制在部分節點上。如今，雲端供應商允許 Kubernetes 管理員建立基於可用區的節點。節點可以位於同一區域內的不同可用區，也可以跨越多個區域。為了方便在多可用區架構中為工作負載配置磁碟區，Trident 使用 CSI Topology。

深入瞭解 CSI Topology 功能 "這裡"。

Kubernetes 提供兩種獨特的磁碟區繫結模式：

- 將 `VolumeBindingMode`設為 `Immediate`時、Trident 會在不感知拓撲的情況下建立磁碟區。磁碟區繫結和動態資源配置會在建立 PVC 時處理。這是預設 `VolumeBindingMode`、適用於不強制執行拓撲限制的叢集。持續磁碟區的建立不會依賴要求 Pod 的排程需求。
- 將 `VolumeBindingMode`設為 `WaitForFirstConsumer`時、系統會延遲建立 PVC 的持續磁碟區並將其繫結、直到排程並建立使用該 PVC 的 Pod 為止。如此一來、建立的磁碟區就能符合拓撲需求所強制執行的排程限制。

> `WaitForFirstConsumer` 綁定模式不需要拓樸標籤。它可以獨立於 CSI 拓樸功能使用。

您需要準備的項目

若要使用 CSI Topology，您需要下列項目：

- 執行 "支援的 Kubernetes 版本" 的 Kubernetes 叢集

```
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"}
```

- 叢集中的節點應該帶有用於引入拓撲感知的標籤(`topology.kubernetes.io/region`和 `topology.kubernetes.io/zone`）。這些標籤*應該在安裝 Trident 之前就存在於叢集中的節點上*，以便 Trident 能夠感知拓撲。

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","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。每個後端都可以包含一個可選的 `supportedTopologies` 資料區塊，用於指定支援的區域和可用區清單。對於使用此類後端的 StorageClasses ，僅當調度到受支援區域/可用區的應用程式請求時，才會建立相應的磁碟區。

以下是後端定義範例：

YAML

```yaml
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

JSON

```json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-a"
    },
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-b"
    }
  ]
}
```

`supportedTopologies` 用於為每個後端提供區域和可用區清單。這些區域和可用區代表可以在 StorageClass 中提供的允許值清單。對於包含後端提供的區域和可用區子集的 StorageClasses，Trident 會在後端建立一個磁碟區。

您也可以按儲存池定義 `supportedTopologies`。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
```

在此範例中、 `region` 和 `zone` 標籤代表儲存資源池的位置。 `topology.kubernetes.io/region` 和 `topology.kubernetes.io/zone` 則指定可從何處使用儲存資源池。

步驟 **2**：定義可感知拓撲的 **StorageClasses**

根據提供給叢集中節點的拓樸標籤、StorageClasses 可定義為包含拓樸資訊。這將決定哪些儲存資源池可作為 PVC 要求的候選對象、以及哪些節點子集可以使用 Trident 所配置的磁碟區。

請參閱下列範例：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions:
    - key: topology.kubernetes.io/zone
      values:
        - us-east1-a
        - us-east1-b
    - key: topology.kubernetes.io/region
      values:
        - us-east1
parameters:
  fsType: ext4
```

在上述提供的 StorageClass 定義中， `volumeBindingMode` 被設定為 `WaitForFirstConsumer`。使用此 StorageClass 請求的 PVC 只有在 Pod 中被引用後才會生效。而且， `allowedTopologies` 提供要使用的 zone 和 region。 `netapp-san-us-east1` StorageClass 會在上述定義的 `san-backend-us-east1` 後端上建立 PVC。

步驟 **3**：建立並使用 **PVC**

建立 StorageClass 並映射到後端後，您現在可以建立 PVC。

請看下面的例子 `spec`：

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

使用此資訊清單建立 PVC 將產生下列結果：

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

若要讓 Trident 建立磁碟區並將其繫結至 PVC，請在 Pod 中使用 PVC。請參閱以下範例：

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

這個 podSpec 指示 Kubernetes 將 pod 排程到 `us-east1` 區域中存在的節點上，並從 `us-east1-a` 或 `us-east1-b` 區域中存在的任何節點中進行選擇。

請參閱下列輸出：

```
kubectl get pods -o wide
NAME         READY    STATUS     RESTARTS    AGE    IP                NODE
NOMINATED NODE   READINESS GATES
app-pod-1    1/1      Running    0           19s    192.168.25.131    node2
<none>           <none>
kubectl get pvc -o wide
NAME       STATUS    VOLUME                                     CAPACITY
ACCESS MODES    STORAGECLASS          AGE    VOLUMEMODE
pvc-san    Bound     pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO             netapp-san-us-east1   48s    Filesystem
```

更新後端以包含 `supportedTopologies`

可以更新現有後端，使其包含 `supportedTopologies`清單，使用 `tridentctl backend update`。這不會影響已配置的磁碟區，並且僅用於後續的 PVC。

尋找更多資訊

- "管理容器資源"
- "nodeSelector"
- "親和性與反親和性"
- "污點與容忍度"

使用快照

Kubernetes 持久磁碟區（PV）的磁碟區快照功能可以建立磁碟區的特定時間點副本。您可以建立使用 Trident 建立的磁碟區快照、匯入在 Trident 外部建立的快照、從現有快照建立新磁碟區，以及從快照還原磁碟區資料。

概況

Volume snapshot 受 `ontap-nas`、 `ontap-nas-flexgroup`、 `ontap-san`、 `ontap-san-economy`、 `solidfire-san`、 `azure-netapp-files`和 `google-cloud-netapp-volumes`驅動程式支援。

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 Volume Snapshot Controller。

> 如果要在 GKE 環境中建立按需磁碟區快照、請勿建立快照控制器。GKE 使用內建的隱藏快照控制器。

建立 **Volume Snapshot**

步驟

1. 建立 `VolumeSnapshotClass`。如需詳細資訊、請參閱 "VolumeSnapshotClass"。
   - `driver` 指向 Trident CSI 驅動程式。
   - `deletionPolicy` 可以是 `Delete` 或 `Retain`。設定為 `Retain` 時，即使 `VolumeSnapshot` 物件被刪除，儲存叢集上的底層實體快照也會被保留。

   範例

   ```
   cat snap-sc.yaml
   ```

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

2. 建立現有 PVC 的快照。

   範例
   - 此範例會建立現有 PVC 的快照。

   ```
   cat snap.yaml
   ```

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   kind: VolumeSnapshot
   metadata:
     name: pvc1-snap
   spec:
     volumeSnapshotClassName: csi-snapclass
     source:
       persistentVolumeClaimName: pvc1
   ```

   - 此範例為名為 `pvc1`的 PVC 建立磁碟區快照物件，並將快照名稱設為 `pvc1-snap`。VolumeSnapshot 類似於 PVC，並與 `VolumeSnapshotContent`物件相關聯，該物件代表實際快照。

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                    AGE
pvc1-snap               50s
```

- 您可以透過描述該 `VolumeSnapshotContent` 物件來識別 `pvc1-snap` VolumeSnapshot。該 `Snapshot Content Name` 會識別提供此快照的 VolumeSnapshotContent 物件。該 `Ready To Use` 參數表示此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`來建立 PVC，使用名為 `<pvc-name>`的 VolumeSnapshot 作為資料來源。建立 PVC 後，可以將其附加到 pod 上，並像使用其他 PVC 一樣使用它。

> PVC 將在與來源磁碟區相同的後端建立。請參閱 "知識庫：無法在備用後端從 Trident PVC 快照建立 PVC"。

以下範例使用 `pvc1-snap` 作為資料來源建立 PVC。

```
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 支援"Kubernetes 預先配置快照流程"讓叢集管理員建立 `VolumeSnapshotContent`物件並匯入在 Trident 外部建立的快照。

開始之前

Trident 必須已建立或匯入快照的父 Volume 。

步驟

1. 叢集管理員： 建立一個 `VolumeSnapshotContent`物件，參照後端快照。這會在 Trident 中啟動快照工作流程。
   - 在 `annotations`中將後端快照的名稱指定為 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`。
   - 請在 `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>`中指定 `snapshotHandle`。這是外部快照工具在 `ListSnapshots`呼叫中提供給 Trident 的唯一資訊。

     <volumeSnapshotContentName> 由於 CR 命名限制，無法始終與後端快照名稱相符。

     範例

     以下範例建立了一個 `VolumeSnapshotContent`物件，該物件引用後端快照 `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. **Cluster admin**： 建立 `VolumeSnapshot` CR，參照 `VolumeSnapshotContent`物件。這會要求存取權限，以便在指定的命名空間中使用 `VolumeSnapshot`。

   範例

   以下範例會建立一個 `VolumeSnapshot` CR，名為 `import-snap`，該 CR 會參照 `VolumeSnapshotContent`，名為 `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 `VolumeSnapshotContent`並執行 `ListSnapshots`呼叫。`Trident` 建立 `TridentSnapshot`。
   - 外部 snapshotter 會將 `VolumeSnapshotContent` 設為 `readyToUse`，並將 `VolumeSnapshot` 設為 `true`。
   - Trident 返回 `readyToUse=true`。
4. 任何使用者： 建立一個 `PersistentVolumeClaim` 以參照新的 `VolumeSnapshot`，其中 `spec.dataSource`（或 `spec.dataSourceRef`）名稱是 `VolumeSnapshot` 名稱。

範例

以下範例建立了一個 PVC，參考名為 `VolumeSnapshot`的 `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資料

預設情況下，快照目錄處於隱藏狀態，以確保使用 `ontap-nas` 和 `ontap-nas-economy` 驅動程式配置的磁碟區的最大相容性。啟用 `.snapshot` 目錄即可直接從快照還原資料。

使用 volume snapshot restore 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 Snapshot 複本時、現有的 Volume 組態會被覆寫。建立 Snapshot 複本後對 Volume 資料所做的變更將會遺失。

從快照進行就地 **Volume** 還原

Trident 使用 `TridentActionSnapshotRestore` (TASR) CR 從快照快速進行原廠磁碟區復原。此 CR 作為一項命令式 Kubernetes 操作運行，操作完成後不會持久保存。

Trident 支援在 `ontap-san`、 `ontap-san-economy`、 `ontap-nas`、 `ontap-nas-flexgroup`、 `azure-netapp-files`、 `google-cloud-netapp-volumes`和 `solidfire-san`驅動程式上進行快照還原。

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 狀態為已綁定。

```
kubectl get pvc
```

- 確認磁碟區快照已準備就緒可供使用。

```
kubectl get vs
```

步驟

1. 建立 TASR CR。此範例為 PVC `pvc1` 和磁碟區快照 `pvc1-snapshot`建立 CR。

   TASR CR 必須位於 PVC 和 VS 存在的命名空間中。

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. 套用 CR 從快照還原。此範例從快照還原 `pvc1`。

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

結果

Trident 會從快照還原資料。您可以驗證快照還原狀態：

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

- 在大多數情況下，Trident 不會在操作失敗後自動重試。您需要再次執行該操作。
- 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。

刪除具有相關快照的 PV

刪除包含關聯快照的持久性磁碟區時，對應的 Trident 磁碟區會更新為「正在刪除」狀態。刪除磁碟區快照即可刪除 Trident 磁碟區。

部署 Volume Snapshot Controller

如果您的 Kubernetes 發行版不包含快照控制器和 CRD、您可以依照下列方式部署它們。

步驟

1. 建立 Volume Snapshot 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 Snapshot Controller。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

如有必要，請打開 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` 並更新 `namespace` 到您的命名空間。

相關連結

- "Volume 快照"
- "VolumeSnapshotClass"

使用磁碟區群組快照

Kubernetes 持久性磁碟區（PV）的磁碟區群組快照 NetApp Trident 提供建立多個磁碟區快照（磁碟區快照群組）的功能。此磁碟區群組快照代表在同一時間點從多個磁碟區取得的複本。

VolumeGroupSnapshot 是 Kubernetes 中的一項測試版功能，使用測試版 API。Kubernetes 1.32 是 VolumeGroupSnapshot 的最低版本要求。

建立 **Volume Group** 快照

以下儲存驅動程式支援 Volume group snapshot：

- `ontap-san` driver - 僅適用於 iSCSI 和 FC 協定，不適用於 NVMe/TCP 協定。
- `ontap-san-economy` - 僅適用於 iSCSI 協定。
- `ontap-nas`

> NetApp ASA r2 或 AFX 儲存系統不支援 Volume group snapshot。

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 Volume Snapshot Controller 。

  > 如果要在 GKE 環境中建立按需磁碟區組快照，請勿建立快照控制器。GKE 使用內建的隱藏快照控制器。

- 在快照控制器 YAML 中，將 `CSIVolumeGroupSnapshot` 功能閘設定為 'true'，以確保啟用磁碟區群組快照。
- 在建立磁碟區群組快照之前，請先建立所需的磁碟區群組快照類別。
- 確保所有 PVC/ 磁碟區都在同一 SVM 上、才能建立 VolumeGroupSnapshot。

步驟

- 在建立 VolumeGroupSnapshot 之前，請先建立 VolumeGroupSnapshotClass。如需更多資訊，請參閱"VolumeGroupSnapshotClass"。

```
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 `pvc1-group-snap` 作為資料來源和標籤 `consistentGroupSnapshot: groupA` 建立 PVC。請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
```

- 建立具有與 PVC 中指定的標籤(`consistentGroupSnapshot: groupA`相同的 VolumeGroupSnapshot。

  此範例會建立磁碟區群組快照：

```
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資料

您可以使用作為 Volume Group Snapshot 一部分建立的個別快照來還原個別 Persistent Volume 。您無法將 Volume Group Snapshot 作為一個單元進行還原。

使用 volume snapshot restore 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 Snapshot 複本時、現有的 Volume 組態會被覆寫。建立 Snapshot 複本後對 Volume 資料所做的變更將會遺失。

從快照進行就地 **Volume** 還原

Trident 使用 `TridentActionSnapshotRestore` (TASR) CR 從快照快速進行原廠磁碟區復原。此 CR 作為一項命令式 Kubernetes 操作運行，操作完成後不會持久保存。

如需詳細資訊，請參閱 "從快照進行就地 Volume 還原"。

刪除具有相關群組快照的 **PV**

刪除群組 Volume 快照時：

- 您可以整體刪除 VolumeGroupSnapshots，而不是刪除群組中的單一快照。
- 如果刪除 PersistentVolumes 時該 PersistentVolume 已存在快照，Trident 會將該磁碟區移至「刪除中」狀態，因為必須先移除快照才能安全地移除該磁碟區。
- 如果使用分組快照建立了複本，然後要刪除該群組，則會開始複本分割作業，並且在分割完成之前無法刪除該群組。

部署 **Volume Snapshot Controller**

如果您的 Kubernetes 發行版不包含快照控制器和 CRD、您可以依照下列方式部署它們。

步驟

1. 建立 Volume Snapshot 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 Snapshot Controller。

```
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，請打開 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` 並更新 `namespace` 到您的命名空間。

相關連結

- "VolumeGroupSnapshotClass"
- "Volume 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。每個新版本都基於先前的版本，並提供新功能、效能提升、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 叢集中，所有命名空間應該只安裝一個 Trident 執行個體。
- Trident 23.07 及更高版本需要 v1 Volume Snapshot，不再支援 alpha 或 beta Snapshot。
- 升級時，請務必提供 `parameter.fsType` 在 `StorageClasses` 中由 Trident 使用。您可以刪除並重新建立 `StorageClasses` 而不會影響現有磁碟區。
  - 這是對 SAN 磁碟區進行 "安全情境"強制執行的要求。
  - 範例輸入 目錄包含範例，例如 `storage-class-basic.yaml.templ` 和 `storage-class-bronze-default.yaml`。
  - 如需更多資訊，請參閱 "已知問題"。

步驟 1：選擇版本

Trident 版本遵循基於日期的 `YY.MM`命名規則，其中「YY」代表年份的後兩位數字，「MM」代表月份。點號版本遵循 `YY.MM.X`規則，其中「X」代表補丁級別。您將根據要升級的版本選擇要升級到的版本。

- 您可以將目前版本直接升級到與其相差不超過四個版本號的目標版本。例如，您可以直接從 24.06（或任何 24.06 的小版本）升級到 25.06。
- 如果您要從超出四版本視窗期的版本升級，請執行多步驟升級。使用您要升級的 "早期版本" 版本對應的升級說明，升級到符合四版本視窗期的最新版本。例如，如果您目前運行的是 23.07 版本，並且想要升級到 25.06 版本：
  a. 首次從 23.07 升級到 24.06。
  b. 然後從 24.06 升級到 25.06 。

> 在 OpenShift Container Platform 上使用 Trident Operator 進行升級時,應升級至 Trident 21.01.1 或更高版本。21.01.0 版本發布的 Trident Operator 包含一個已知問題,該問題已在 21.01.1 版本中修復。如需更多詳細資訊,請參閱 "GitHub 上的問題詳情"。

步驟 2：確定原始安裝方法

若要確定您最初用於安裝 Trident 的版本：

1. 使用 `kubectl get pods -n trident`檢查 Pod。
   - 如果沒有操作員 pod，Trident 是使用 `tridentctl`安裝的。
   - 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 pod，請使用 `kubectl describe torc`來判斷 Trident 是否使用 Helm 安裝。
   - 如果有 Helm 標籤，Trident 是使用 Helm 安裝的。
   - 如果沒有 Helm 標籤、則 Trident 是使用 Trident 操作員手動安裝。

步驟 **3**：選擇升級方法

通常情況下，您應該使用與初始安裝相同的方法進行升級，但您也可以"在安裝方法之間移動"。有兩種升級 Trident 的選項。

- "使用 Trident 操作員進行升級"

  我們建議您在使用 operator 升級之前先檢閱"了解 operator 升級工作流程"。

*

使用 operator 進行升級

了解 operator 升級工作流程

在使用 Trident operator 升級 Trident 之前，您應該先了解升級過程中發生的背景程序。這包括對 Trident 控制器、控制器 Pod 和節點 Pod 的變更，以及啟用滾動更新的節點 DaemonSet。

Trident 操作員升級處理

安裝和升級 Trident 的眾多"使用 Trident 運算子的好處"之一是能夠自動處理 Trident 和 Kubernetes 物件，而不會中斷現有掛載的磁碟區。這樣，Trident 可以支援零停機升級，或"滾動更新"。具體來說，Trident Operator 與 Kubernetes 叢集通訊以：

- 刪除並重新建立 Trident Controller 部署和節點 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 架構的更多資訊，請參閱 "Trident 架構"。

Operator 升級工作流程

當您使用 Trident 運算子啟動升級時：

1. **Trident** 運算子：
   a. 偵測目前安裝的 Trident 版本（版本 *n*）。

b. 更新所有 Kubernetes 物件，包括 CRD、RBAC 和 Trident SVC。

c. 刪除版本 *n* 的 Trident 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 Node DaemonSet。該操作符不會等待 Node Pod 終止。

b. 為 *n+1* 建立 Trident 節點守護程序集。

4. **Kubernetes** 會在未執行 Trident Node Pod *n* 的節點上建立 Trident Node Pod。這可確保每個節點上永遠不會存在多個 Trident Node Pod，無論版本為何。

使用 **Trident Operator** 或 **Helm** 升級 **Trident** 安裝

您可以使用 Trident Operator 手動或透過 Helm 升級 Trident。您可以從一個 Trident Operator 安裝升級到另一個 Trident Operator 安裝，也可以從 `tridentctl`安裝升級到 Trident Operator 版本。在升級 Trident Operator 安裝之前，請先檢閱"選擇升級方法"。

升級手動安裝

您可以將叢集範圍的 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操作員。

> 若要從使用命名空間範圍運算子安裝的 Trident （版本 20.07 至 20.10）進行升級，請使用 "您已安裝的版本" 的 Trident 升級說明。

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 "支援的 Kubernetes 版本" 的 Kubernetes 叢集。

步驟

1. 驗證您的 Trident 版本：

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

2. 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更新 `operator.yaml`、`tridentorchestrator_cr.yaml`和 `post_1_25_bundle.yaml`。

3. 刪除用於安裝目前 Trident 實例的 Trident 操作符。例如，如果您是從 25.02 版本升級，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 `TridentOrchestrator`屬性自訂了初始安裝，則可以編輯該 `TridentOrchestrator`物件來修改安裝參數。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等變更。
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 `bundle_post_1_25.yaml`，取決於您的 Kubernetes 版本。例如，如果您要安裝 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 Trident torc，使其包含映像 25.06.0。

升級 Helm 安裝

您可以升級 Trident Helm 安裝。

> 當將安裝了 Trident 的 Kubernetes 叢集從 1.24 升級到 1.25 或更高版本時，必須先更新 values.yaml 將 `excludePodSecurityPolicy`設定為 `true`或新增 `--set excludePodSecurityPolicy=true`至 `helm upgrade`命令中，然後才能升級叢集。

如果您已將 Kubernetes 叢集從 1.24 升級到 1.25，但未升級 Trident helm，則 helm 升級將會失敗。若要成功完成 helm 升級，請先執行以下步驟：

1. 從 https://github.com/helm/helm-mapkubeapis安裝 helm-mapkubeapis 外掛程式。
2. 在安裝 Trident 的命名空間中對 Trident 版本執行預運行。這將列出要清理的資源。

```
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. 如果您"使用 Helm 安裝了 Trident"，可以使用 `helm upgrade trident netapp-trident/trident-operator --version 100.2506.0`一步升級。如果您沒有新增 Helm 倉庫或無法使用它進行升級：
   a. 從 "GitHub 上的_資產_部分" 下載最新版的 Trident。
   b. 使用 `helm upgrade` 命令，其中 `trident-operator-25.10.0.tgz` 反映您要升級到的版本。

```
helm upgrade <name> trident-operator-25.10.0.tgz
```

如果您在初始安裝期間設定了自訂選項（例如為 Trident 和 CSI 映像指定私有、鏡像註冊表），請在 `helm upgrade` 指令後附加 `--set`，以確保這些選項包含在升級指令中，否則這些值將會重設為預設值。

2. 運行 `helm list` 以驗證圖表和應用程式版本是否均已升級。運行 `tridentctl logs` 以查看任何調試訊息。

從 `tridentctl` 安裝程式升級到 **Trident operator**

您可以從 `tridentctl`安裝升級到最新版本的 Trident 操作器。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 "在安裝方法之間移動"。

步驟

1. 下載最新的 Trident 版本。

```
# Download the release required [25.10.0]
mkdir 25.10.0
cd 25.10.0
wget
https://github.com/NetApp/trident/releases/download/v25.10.0/trident-
installer-25.10.0.tar.gz
tar -xf trident-installer-25.10.0.tar.gz
cd trident-installer
```

2. 根據清單檔案建立 `tridentorchestrator` 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 operator。

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

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                  READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. 建立 `TridentOrchestrator` CR 以安裝 Trident。

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

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s
```

5. 確認 Trident 已升級至預期版本。

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v25.10.0
```

使用 **tridentctl** 進行升級

您可以使用 `tridentctl`輕鬆升級現有的 Trident 安裝。

關於此任務

解除安裝並重新安裝 Trident 相當於一次升級。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。已配置的 PV 在 Trident 離線期間仍可用，Trident 將在恢復在線後為在此期間創建的任何 PVC 配置卷。

開始之前

在使用 `tridentctl`進行升級之前，請先檢查"選擇升級方法"。

步驟

1. 在 `tridentctl`中執行卸載指令，以移除與 Trident 相關的所有資源，但 CRD 和相關物件除外。

```
./tridentctl uninstall -n <namespace>
```

2. 重新安裝 Trident 。請參閱 "使用 tridentctl 安裝 Trident" 。

> ⓘ 請勿中斷升級程序。確保安裝程式執行完成。

使用 **tridentctl** 管理 **Trident**

該軟體包 "Trident 安裝程式套件"包含 `tridentctl`命令列實用程序，方便用戶輕鬆存取 Trident。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

您可以執行 `tridentctl help`以取得 `tridentctl`的可用命令清單，或將 `--help`旗標附加至任何命令，以取得該特定命令的選項和旗標清單。

```
tridentctl [command] [--optional-flag]
```

Trident `tridentctl` 工具支援以下指令和全域標誌。

命令

create
將資源新增至 Trident。

delete
從 Trident 移除一個或多個資源。

get
從 Trident 取得一或多個資源。

help
關於任何命令的說明。

images
列印一份 Trident 所需容器映像的表格。

import
將現有資源匯入 Trident。

install
安裝 Trident。

logs
列印 Trident 的日誌。

send
從 Trident 傳送資源。

uninstall
卸載 Trident。

update
在 Trident 中修改資源。

update backend state
暫時停止後端作業。

upgrade
在 Trident 中升級資源。

version
列印 Trident 的版本。

全域旗標

-d, --debug

偵錯輸出。

-h, --help

的說明 `tridentctl`。

-k, --kubeconfig string

指定 `KUBECONFIG` 路徑，以便在本機或從一個 Kubernetes 叢集到另一個 Kubernetes 叢集運行命令。

或者，您可以匯出 `KUBECONFIG` 變數以指向特定的 Kubernetes 叢集，並向該叢集發出 `tridentctl` 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）其中之一。

-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 （適用於 IPv4）或 [::1] （適用於 IPv6）。

命令選項和標誌

建立

使用 `create` 指令將資源新增到 Trident。

```
tridentctl create [option]
```

選項

`backend`：將後端新增至 Trident。

刪除

使用 `delete` 指令從 Trident 移除一個或多個資源。

```
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 Volume 快照。

`storageclass`：從 Trident 刪除一個或多個儲存類別。
`volume`：從 Trident 刪除一個或多個儲存 Volume 。

取得

使用 `get` 指令從 Trident 取得一個或多個資源。

```
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 取得一個或多個磁碟區。

旗標

`-h, --help`：磁碟區的說明。
`--parentOfSubordinate string`：將查詢限制為從屬來源磁碟區。
`--subordinateOf string`：將查詢限制為磁碟區的從屬磁碟區。

映像

使用 `images` 標誌列印 Trident 所需的容器映像表格。

```
tridentctl images [flags]
```

旗標

`-h, --help`:圖像幫助。
`-v, --k8s-version string`:Kubernetes 叢集的語意版本。

匯入磁碟區

使用 `import volume` 指令將現有磁碟區導入 Trident。

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

別名

`volume, v`

旗標

`-f, --filename string`: YAML 或 JSON PVC 檔案的路徑。
`-h, --help`: 卷的幫助資訊。
`--no-manage`: 僅建立 PV/PVC。不進行磁碟區生命週期管理。

安裝

使用 `install` 標誌安裝 Trident。

```
tridentctl install [flags]
```

旗標

--autosupport-image string：自動支援遙測的容器映像（預設值為「netapp/trident autosupport:<current-version>」）。
--autosupport-proxy string：用於發送自動支援遙測的代理程式的位址/連接埠。
--enable-node-prep：嘗試在節點上安裝所需的軟體包。
--generate-custom-yaml：產生 YAML 檔案而不進行任何安裝。
-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（預設值為 1m30s）。
--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（預設值為 3m0s）。
--kubelet-dir string：kubelet 內部狀態的主機位置（預設值為「/var/lib/kubelet」）。
--log-format string：Trident 日誌格式（text、json）（預設值為「text」）。
--node-prep：啟用 Trident 使 Kubernetes 叢集的節點能夠使用指定的資料儲存協定管理磁碟區。目前，**`iscsi`是唯一支援的值。從 `OpenShift 4.19` 版本開始，此功能支援的最低 `Trident` 版本為 `25.06.1`。**
`--pv string`：Trident 使用的舊版 PV 名稱，確保該 PV 不存在（預設值為「trident」）。
--pvc string：Trident 使用的舊版 PVC 名稱，確保該 PVC 不存在（預設值為「trident」）。
--silence-autosupport：不自動發送自動支援包至 NetApp（預設值為 true）。
--silent：安裝期間停用大部分輸出。
--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（預設值為 100；可選）。
--use-custom-yaml：使用 setup 目錄中存在的任何 YAML 檔案。
--use-ipv6：Trident 通訊使用 IPv6。

日誌

使用 logs 標誌列印 Trident 的日誌。

```
tridentctl logs [flags]
```

旗標

-a, --archive：除非另有指定，否則建立包含所有日誌的支援存檔。
-h, --help：日誌說明。
-l, --log string：要顯示的 Trident 日誌。可選值包含 trident|auto|trident-operator|all（預設值為 "auto"）。
--node string：要從中收集節點 Pod 日誌的 Kubernetes 節點名稱。
-p, --previous：如果存在，則取得上一個容器執行個體的日誌。
--sidecars：取得 Sidecar 容器的日誌。

傳送

使用 send 指令從 Trident 傳送資源。

```
tridentctl send [option]
```

選項

autosupport：將 Autosupport 歸檔傳送至 NetApp。

解除安裝

使用 `uninstall` 標誌卸載 Trident 。

```
tridentctl uninstall [flags]
```

旗標

`-h, --help`：卸載幫助。
`--silent`：卸載過程中停用大部分輸出。

更新

使用 `update` 指令修改 Trident 中的資源。

```
tridentctl update [option]
```

選項

`backend`：在 Trident 中更新後端。

更新後端狀態

使用 `update backend state` 命令可以暫停或恢復後端操作。

```
tridentctl update backend state <backend-name> [flag]
```

需要考慮的要點

- 如果後端是使用 TridentBackendConfig（tbc）建立的，則無法使用 `backend.json`檔案更新後端。
- 如果 `userState`已在 tbc 中設置，則無法使用 `tridentctl update backend state <backend-name> --user-state suspended/normal`命令對其進行修改。
- 若要恢復透過 tridentctl 設定 `userState` 的功能（此欄位已透過 tbc 設定），必須先從 tbc 移除 `userState` 欄位。這可以透過 `kubectl edit tbc` 命令完成。移除 `userState` 欄位後，您可以使用 `tridentctl update backend state` 命令變更後端的 `userState`。
- 使用 `tridentctl update backend state`來更改 `userState`。你也可以使用 `TridentBackendConfig`或 `backend.json`檔案來更新 `userState`；這會觸發後端的完整重新初始化，並且可能會花費較多時間。

  旗標

  `-h, --help`:後端狀態說明。
  `--user-state`:設定為 `suspended`可暫停後端操作。設定為 `normal`可恢復後端操作。設定為 `suspended`時:

- `AddVolume` 和 `Import Volume` 已暫停。
- `CloneVolume`, `ResizeVolume`, `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`, `ReconcileNodeAccess` 仍然可用。

您也可以使用 `userState`欄位在後端組態檔 `TridentBackendConfig`或 `backend.json`中更新後端狀態。如需更多資訊，請參閱"管理後端的選項"和"使用 kubectl 執行後端管理"。

- 範例： *

JSON

請按照以下步驟，使用 `userState`檔案來更新 `backend.json`：

1. 編輯 `backend.json` 檔案以包含 `userState` 欄位，並將其值設為「suspended」。
2. 使用 `tridentctl update backend` 命令和更新的 `backend.json` 檔案路徑來更新後端。

   範例：`tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

YAML

您可以使用 `kubectl edit <tbc-name> -n <namespace>` 指令在套用 tbc 後對其進行編輯。以下範例使用 `userState: suspended` 選項將後端狀態更新為暫停：

```
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`標誌列印 `tridentctl`和正在執行的 Trident 服務的版本。

```
tridentctl version [flags]
```

旗標

`--client`：僅限客戶端版本（無需伺服器）。
`-h, --help`：版本說明。

外掛程式支援

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 環境變數指定的資料夾中，Tridentctl 就會偵測到該外掛程式。所有偵測到的外掛程式都會列在 tridentctl help 的插件部分。此外，您也可以透過在環境變數 TRIDENTCTL_PLUGIN_PATH 中指定插件資料夾來縮小搜尋範圍（例如： `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`）。如果使用此變數，tridentctl 將僅在指定的資料夾中搜尋。

監控 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 效能。

概況

Trident 提供的指標可讓您執行以下操作：

- 密切注意 Trident 的運作狀況和組態。您可以檢查作業的成功程度，以及它是否能如預期般與後端通訊。
- 檢查後端使用資訊，了解後端配置了多少磁碟區、消耗的空間量等等。
- 維護可用後端已配置磁碟區數量的映射表。
- 追蹤效能。您可以查看 Trident 與後端通訊以及執行作業所需的時間。

> 預設情況下， Trident 的指標會在目標連接埠 `8001`的 `/metrics`端點上公開。安裝 Trident 時，這些指標*預設為啟用*。您也可以設定為透過 HTTPS 在連接埠 `8444`上使用 Trident 指標。

您需要準備的項目

- 已安裝 Trident 的 Kubernetes 叢集。
- 一個 Prometheus 執行個體。這可以是 "容器化 Prometheus 部署"，或者您可以選擇將 Prometheus 作為 "原生應用程式"執行。

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、它創建的磁碟區等資訊。請參閱 "Prometheus Operator 說明文件"。

步驟 2 ：建立 Prometheus ServiceMonitor

要使用 Trident 指標，您應該建立一個 Prometheus ServiceMonitor 來監視 `trident-csi` 服務並監聽

metrics 連接埠。範例 ServiceMonitor 如下：

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

此 ServiceMonitor 定義會擷取 trident-csi 服務傳回的指標，並專門尋找服務的 metrics 端點。因此，Prometheus 現在已設定為瞭解 Trident 的指標。

除了直接從 Trident 取得的指標之外、 kubelet 還透過自己的指標端點公開了許多 `kubelet_volume_*`指標。Kubelet 可以提供有關已掛載磁碟區、 Pod 以及它處理的其他內部操作的資訊。請參閱 "這裡"。

透過 **HTTPS** 使用 **Trident** 指標

若要透過 HTTPS（連接埠 8444）使用 Trident 指標，您必須修改 ServiceMonitor 定義以包含 TLS 組態。您還需要將 trident-csi 密碼從 trident 命名空間複製到 Prometheus 執行的命名空間。您可以使用以下命令執行此操作：

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

HTTPS 指標的 ServiceMonitor 範例如下所示：

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

Trident 支援所有安裝方法中的 HTTPS 指標：tridentctl、Helm chart 和 Operator：

- 如果您使用 `tridentctl install` 指令，可以傳遞 `--https-metrics` 標誌來啟用 HTTPS 指標。
- 如果您使用的是 Helm chart，則可以設定 `httpsMetrics` 參數以啟用 HTTPS 指標。
- 如果您正在使用 YAML 檔案，您可以在 `trident-deployment.yaml` 檔案中的 `trident-main` container 新增 `--https_metrics` flag。

步驟 3：使用 PromQL 查詢 Trident 指標

PromQL 非常適合建立傳回時間序列或表格資料的運算式。

以下是您可以使用的 PromQL 查詢：

取得 **Trident** 健全狀況資訊

- **Trident 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
```

- 透過狀態碼從 **Trident** 取得的 **REST** 回應百分比

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Trident** 執行作業的平均持續時間（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** 使用資訊

- 平均磁碟區大小

```
trident_volume_allocated_bytes/trident_volume_count
```

- 各後端配置的總磁碟區空間

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

取得個別磁碟區使用量

只有同時收集 kubelet 指標時，此功能才會啟用。

- 每個磁碟區已使用空間的百分比

```
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，請在 Trident 安裝期間傳遞 `--silence-autosupport` 標誌。
- Trident 也可以透過 `tridentctl send autosupport`按需向 NetApp 支援部門發送容器日誌。您需要觸發 Trident 上傳日誌。提交日誌之前，您應該接受 NetApp 的https://www.netapp.com/company/legal/privacy-policy/["隱私權政策"^]。
- 除非另有說明，Trident 會取得過去 24 小時的日誌。
- 您可以使用 `--since`旗標指定日誌保留時間範圍。例如： `tridentctl send autosupport --since=1h`。此資訊會透過與 Trident 一起安裝的 `trident-autosupport`容器收集並傳送。您可以在"Trident AutoSupport"取得容器映像。
- Trident AutoSupport 不會收集或傳輸個人識別資訊（PII）或個人資訊。它附帶 "最終用戶授權協議"不適用於 Trident 容器映像本身。您可以深入瞭解 NetApp 對資料安全性和信任的承諾 "這裡"。

Trident 傳送的承載範例如下所示：

```
---
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。如果您使用私有登錄來儲存容器映像，可以使用 `--image-registry` 旗標。
- 您也可以透過產生安裝 YAML 檔案來設定 proxy URL。這可以使用 `tridentctl install --generate-custom-yaml`來建立 `YAML` 檔案，並為 `trident-autosupport` container 在 `trident-deployment.yaml`中新增 `--proxy-url`參數來完成。

停用 Trident 指標

若要停用指標回報，您應該產生自訂 YAML（使用 `--generate-custom-yaml` 旗標），並編輯它們以移除 `--metrics` 旗標，使其不會在 `trident-main` 容器中被呼叫。

解除安裝 Trident

您應該使用與安裝 Trident 相同的方法來卸載 Trident。

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 不會移除 Trident 所建立的 CRD 或相關物件。如果您需要徹底移除 Trident 及其所有資料，請參閱 "徹底移除 Trident 和 CRDs"。

開始之前

如果您要停用 Kubernetes 集群，則必須在卸載之前刪除所有使用 Trident 建立的磁碟區的應用程式。這可以確保在刪除 Kubernetes 節點之前，PVC 已從這些節點上取消發布。

確定原始安裝方法

您應該使用與安裝 Trident 時相同的方法來卸載 Trident。解除安裝前，請確認您最初用於安裝 Trident 的版本。

1. 使用 `kubectl get pods -n trident`檢查 Pod。
   - 如果沒有操作員 pod，Trident 是使用 `tridentctl`安裝的。
   - 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 pod，請使用 `kubectl describe tproc trident`來確定 Trident 是否使用 Helm 安裝。
   - 如果有 Helm 標籤，Trident 是使用 Helm 安裝的。
   - 如果沒有 Helm 標籤、則 Trident 是使用 Trident 操作員手動安裝。

卸載 Trident Operator 安裝

您可以手動解除安裝 Trident Operator 安裝，也可以使用 Helm 解除安裝。

解除安裝手動安裝

如果您使用操作員安裝了 Trident，則可以透過執行下列其中一項操作來解除安裝它：

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`標誌設為 `true`時，Trident 運算元會卸載 Trident，但不會刪除 TridentOrchestrator 本身。如果您想再次安裝 Trident，則需要清理 TridentOrchestrator 並建立新的設定檔。

2. 刪除 **`TridentOrchestrator`**：透過移除 `TridentOrchestrator` 用於部署 Trident 的 CR，您可以指示操作員卸載 Trident。操作員將處理 `TridentOrchestrator` 的移除，並繼續移除 Trident 部署和守護程序集，同時刪除其在安裝過程中建立的 Trident Pod。

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

解除安裝 **Helm**

如果您使用 Helm 安裝了 Trident，則可以使用 `helm uninstall`卸載它。

```
#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`

使用 `uninstall` 命令在 `tridentctl` 中刪除與 Trident 相關的所有資源，但 CRD 和相關物件除外：

```
./tridentctl uninstall -n <namespace>
```

Trident for Docker

部署的先決條件

在部署 Trident 之前，您必須在主機上安裝並設定必要的傳輸協定先決條件。

驗證需求

- 確認您的部署符合所有 "要求"。
- 請確認您已安裝受支援的 Docker 版本。如果您的 Docker 版本過舊 "安裝或更新它"。

```
docker --version
```

- 請確認您的主機上已安裝並設定傳輸協定必要條件。

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命令安裝 iSCSI 工具。

RHEL 8+

1. 安裝下列系統套件：

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-
mapper-multipath
```

2. 檢查 iscsi-initiator-utils 版本是否為 6.2.0.874-2.el7 或更新版本：

```
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. 啟用多重路徑：

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

> 請確保 `etc/multipath.conf` 包含 `find_multipaths no` 並位於 `defaults` 之下。

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. 安裝下列系統套件：

```
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. 啟用多重路徑：

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

> 請確保 `etc/multipath.conf` 包含 `find_multipaths no` 並位於 `defaults` 之下。

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命令安裝 NVMe 工具。

- NVMe 需要 RHEL 9 或更新版本。
- 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 套件，則您可能需要將節點的核心版本更新為包含 NVMe 套件的版本。

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）的工作節點並搭配 FC PVs 時，請在 `discard` StorageClass 中指定 mountOption 以執行即時空間回收。請參閱 "Red Hat 說明文件"。

RHEL 8+

1. 安裝下列系統套件：

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

2. 啟用多重路徑：

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

> 請確保 `etc/multipath.conf` 包含 `find_multipaths no` 並位於 `defaults` 之下。

3. 確保 `multipathd` 正在執行：

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

Ubuntu

1. 安裝下列系統套件：

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

2. 啟用多重路徑：

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

> 請確保 `etc/multipath.conf` 包含 `find_multipaths no` 並位於 `defaults` 之下。

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。

Trident 的多個執行個體可以在同一主機上同時執行。這允許同時連線至多個儲存系統和儲存類型，並且能夠自訂用於 Docker 磁碟區的儲存。

您需要準備的項目

請參閱 "部署的先決條件"。確保滿足先決條件後，即可部署 Trident。

Docker 管理的外掛程式方法（版本 1.13/17.03 及更新版本）

> 開始之前
>
> 如果您在 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. 停止所有執行中的執行個體：

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. 重新啟動 Docker 。

```
systemctl restart docker
```

3. 請確保您已安裝 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"。

步驟

1. 建立組態檔並依照下列方式指定選項：
   - `config`：預設檔案名稱是 `config.json`，但您可以透過指定 `config`選項與檔案名稱來使用您選擇的任何名稱。設定檔必須位於主機系統的 `/etc/netappdvp`目錄中。
   - `log-level`：指定日誌等級（(`debug`、 `info`、 `warn`、 `error`、 `fatal`）。預設值為 `info`。
   - `debug`：指定是否啟用偵錯記錄。預設值為 false。如果為 true，則會覆寫 log-level。
     i. 建立組態檔的位置：

```
sudo mkdir -p /etc/netappdvp
```

ii. 建立組態檔：

```
cat << EOF > /etc/netappdvp/config.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. 使用託管外掛程式系統啟動 Trident。將 `<version>`替換為您正在使用的外掛程式版本（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. 移除磁碟區「firstVolume」：

```
docker volume rm firstVolume
```

傳統方法（1.12 版或更早版本）

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， "請依照您的發行版本說明進行操作"。

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 Volume 插件：

a. 下載並解壓縮應用程式：

```
wget
https://github.com/NetApp/trident/releases/download/10.0/trident-
installer-25.10.0.tar.gz
tar zxf trident-installer-25.10.0.tar.gz
```

b. 移至 bin 路徑中的位置：

```
sudo mv trident-installer/extras/bin/trident /usr/local/bin/
sudo chown root:root /usr/local/bin/trident
sudo chmod 755 /usr/local/bin/trident
```

c. 建立組態檔的位置：

```
sudo mkdir -p /etc/netappdvp
```

d. 建立組態檔：

```
cat << EOF > /etc/netappdvp/ontap-nas.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. 放置好二進位檔案並建立設定檔後，使用所需的設定檔啟動 Trident 守護程序。

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```

> 除非另有指定，Volume 驅動程式的預設名稱為「netapp」。

啟動精靈程式後，您可以使用 Docker CLI 介面來建立和管理磁碟區。

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 Volume ：

```
docker volume rm trident_1
```

```
docker volume rm trident_2
```

在系統啟動時啟動 **Trident**

systemd 型系統的範例單元檔案可在 Git 儲存庫的 `contrib/trident.service.example`中找到。若要在 RHEL 中使用該檔案，請執行下列操作：

1. 將檔案複製到正確位置。

   如果您執行多個執行個體，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. 編輯該檔案，將描述（第 2 行）變更為與驅動程式名稱相符，並將組態檔路徑（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。

   此名稱會根據您在 `/usr/lib/systemd/system` 目錄中為檔案命名的內容而有所不同。

   ```
   systemctl enable trident
   ```

5. 啟動服務。

   ```
   systemctl start trident
   ```

6. 檢視狀態。

   ```
   systemctl status trident
   ```

每次修改單元檔案時，都要執行 `systemctl daemon-reload` 命令使其識別變更。

升級或解除安裝 Trident

您可以安全地升級 Trident for Docker，不會對正在使用的磁碟區造成任何影響。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無法執行，應用程式將無法掛載磁碟區，直到該插件重新啟動。大多數情況下，這只需幾秒鐘。

升級

請依照下列步驟升級 Trident for Docker 。

步驟

1. 列出現有磁碟區：

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. 停用外掛程式：

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. 升級外掛程式：

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

> Trident 的 18.01 版本取代了 nDVP。您應該直接從 `netapp/ndvp-plugin`映像升級到 `netapp/trident-plugin`映像。

4. 啟用外掛程式：

```
docker plugin enable netapp:latest
```

5. 確認外掛程式已啟用：

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. 確認磁碟區可見：

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

如果您是從舊版 Trident （ pre-20.10 ）升級到 Trident 20.10 或更高版本，可能會遇到錯誤。如需更多資訊，請參閱"已知問題"。如果遇到錯誤，您應該先停用該外掛程式，然後移除該外掛程式，最後透過傳遞額外的組態參數來安裝所需的 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. 移除外掛程式所建立的任何磁碟區。
2. 停用外掛程式：

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. 移除外掛程式：

```
docker plugin rm netapp:latest
```

使用磁碟區

您可以使用標準 `docker volume` 指令輕鬆建立、複製和刪除磁碟區，並在需要時指定 Trident 驅動程式名稱。

建立磁碟區

- 使用預設名稱的驅動程式建立磁碟區：

```
docker volume create -d netapp --name firstVolume
```

- 使用特定的 Trident 執行個體建立磁碟區：

```
docker volume create -d ntap_bronze --name bronzeVolume
```

如果您不指定任何 "選項"，則使用驅動程式的預設值。

- 覆寫預設磁碟區大小。請參閱下列範例、使用驅動程式建立 20 GiB 磁碟區：

```
docker volume create -d netapp --name my_vol --opt size=20G
```

Volume 大小以字串形式表示，包含一個整數值，單位可選（例如：10G、20GB、3TiB）。如果未指定單位，則預設為 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、...）。

移除磁碟區

- 刪除該磁碟區的方式與其他 Docker 磁碟區一樣：

```
docker volume rm firstVolume
```

使用 `solidfire-san` 驅動程式時，上述範例會刪除並清除磁碟區。

請依照下列步驟升級 Trident for Docker 。

複製磁碟區

使用 `ontap-nas`、 `ontap-san`和 `solidfire-san`儲存驅動程式時、Trident 可以複製磁碟區。使用 `ontap-nas-flexgroup`或 `ontap-nas-economy`驅動程式時、不支援複製。從現有磁碟區建立新磁碟區將建立一個新的快照。

- 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。這將建立新的 Snapshot ：

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume>
```

- 從磁碟區上的現有快照建立新磁碟區。這不會建立新快照：

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

範例

```
docker volume inspect firstVolume

[
  {
    "Driver": "ontap-nas",
    "Labels": null,
    "Mountpoint": "/var/lib/docker-volumes/ontap-
nas/netappdvp_firstVolume",
    "Name": "firstVolume",
    "Options": {},
    "Scope": "global",
    "Status": {
      "Snapshots": [
        {
          "Created": "2017-02-10T19:05:00Z",
          "Name": "hourly.2017-02-10_1505"
        }
      ]
    }
  }
]

docker volume create -d ontap-nas --name clonedVolume -o from=firstVolume
clonedVolume

docker volume rm clonedVolume
docker volume create -d ontap-nas --name volFromSnap -o from=firstVolume
-o fromSnapshot=hourly.2017-02-10_1505
volFromSnap

docker volume rm volFromSnap
```

存取外部建立的磁碟區

容器可以使用 Trident 存取外部建立的區塊裝置（或其複製），*僅限於*這些裝置沒有分割區且其檔案系統受 Trident 支援的情況下（例如：`ext4`格式化的`/dev/sdc1`將無法透過 Trident 存取）。

驅動程式特定的磁碟區選項

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。請參閱下文，了解適用於您所配置儲存系統的選項。

在建立磁碟區操作期間使用這些選項非常簡單。在 CLI 操作期間使用 `-o` 運算子提供選項和值。這些會覆寫 JSON 組態檔中的任何等效值。

ONTAP Volume 選項

NFS 、 iSCSI 和 FC 的磁碟區建立選項包括以下幾種：

選項	說明
`size`	磁碟區大小、預設為 1 GiB。
`spaceReserve`	磁碟區的精簡或完整配置，預設為精簡配置。有效值為 `none`（精簡配置）和 `volume`（完整配置）。
`snapshotPolicy`	這將把快照原則設定為所需值。預設值為 `none`，表示不會為該磁碟區自動建立快照。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」的原則，該原則會建立並保留六個每小時快照、兩個每日快照和兩個每週快照。可以透過瀏覽磁碟區中任意目錄下的 `.snapshot`目錄來還原快照中已儲存的資料。
`snapshotReserve`	這將把快照保留設定為所需的百分比。預設值為無，這表示如果您選擇了 snapshotReserve，ONTAP 會選擇 snapshotReserve（通常為 5%）；如果 snapshotPolicy 為 none，則為 0%。您可以在設定檔中為所有 ONTAP 後端設定預設的 snapshotReserve 值，並且除了 ontap-nas-economy 之外，您可以將其作為所有 ONTAP 後端的磁碟區建立選項。
`splitOnClone`	複製磁碟區時，這會導致 ONTAP 立即將複製磁碟區與其父磁碟區分離。預設值為 `false`。某些磁碟區複製使用案例最好在建立後立即將複製磁碟區與其父磁碟區分離，因為不太可能有任何儲存效率的機會。例如，複製空資料庫可以節省大量時間，但節省的儲存空間很少，因此最好立即分離複製磁碟區。
`encryption`	在新磁碟區上啟用 NetApp Volume Encryption（NVE）；預設為 `false`。若要使用此選項，必須在叢集上取得 NVE 授權並啟用 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。 如需更多資訊，請參閱："Trident 與 NVE 和 NAE 的運作方式"。
`tieringPolicy`	設定磁碟區要使用的分層策略。這決定了當資料變為非活動狀態 （ 冷儲存 ） 時，是否將其移至雲端層。

以下附加選項僅適用於 NFS ：

選項	說明
unixPermissions	此設定控制磁碟區本身的權限集。預設情況下，權限將設定為 `---rwxr-xr-x，或數字表示法 0755，`root`將是擁有者。文字或數字格式均可使用。
snapshotDir	將此設定為 true`將使 `.snapshot`目錄對存取磁碟區的用戶端可見。預設值為 `false，表示`.snapshot`目錄的可見性預設為停用狀態。某些映像（例如官方 MySQL 映像）在 `.snapshot`目錄可見的情況下無法正常運作。
exportPolicy	設定磁碟區要使用的匯出原則。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	說明
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	將此設定為 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 無法接受寫入操作時，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

請參閱以下範例：

- 建立 10 GiB 磁碟區：

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- 建立一個包含快照的 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、且快照原則為 `none`，Trident 會使用 0% 的快照保留。

- 建立沒有 Snapshot 原則和 Snapshot 保留的磁碟區：

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- 建立一個沒有快照原則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
```

- 建立一個具有快照原則和 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
```

- 建立具有快照原則的磁碟區，並接受 ONTAP 的預設快照保留空間（通常為 5% ）：

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Element 軟體 Volume 選項

Element 軟體選項公開了與磁碟區關聯的大小和服務品質（QoS）策略。建立磁碟區時，會使用 `-o type=service_level` 命名規則指定與其關聯的 QoS 策略。

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，並在組態檔中指定與名稱相關聯的最小、最大和突發 IOPS。

其他 Element 軟體磁碟區建立選項包括以下幾種：

選項	說明
`size`	磁碟區的大小、預設為 1 GiB 或組態項目 …"defaults": {"size": "5G"}。

選項	說明
`blocksize`	可使用 512 或 4096 、預設為 512 或配置條目 DefaultBlockSize 。

範例

請參閱以下包含 QoS 定義的範例組態檔：

```
{
  "Types": [
    {
      "Type": "Bronze",
      "Qos": {
        "minIOPS": 1000,
        "maxIOPS": 2000,
        "burstIOPS": 4000
      }
    },
    {
      "Type": "Silver",
      "Qos": {
        "minIOPS": 4000,
        "maxIOPS": 6000,
        "burstIOPS": 8000
      }
    },
    {
      "Type": "Gold",
      "Qos": {
        "minIOPS": 6000,
        "maxIOPS": 8000,
        "burstIOPS": 10000
      }
    }
  ]
}
```

在上述組態中、我們有三個原則定義： Bronze 、 Silver 和 Gold 。這些名稱是任意的。

- 建立 10 GiB Gold Volume ：

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- 建立 100 GiB Bronze Volume ：

```
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疑難排解。您用來收集日誌的方法會因您執行 Docker 外掛程式的方式而異。

收集記錄以進行疑難排解

步驟

1. 如果您使用建議的託管外掛程式方法執行 Trident（即使用 `docker plugin`指令），請按如下方式檢視：

```
docker plugin ls
```

```
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin    false
journalctl -u docker | grep 4fb97d2b956b
```

標準日誌等級應該足以幫助您診斷大多數問題。如果發現這還不夠，您可以啟用偵錯日誌記錄。

2. 若要啟用偵錯日誌記錄、請安裝啟用偵錯日誌記錄的外掛程式：

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

或者，在外掛程式已安裝的情況下啟用偵錯記錄：

```
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檔案，則日誌位於主機的 `/var/log/netappdvp` 目錄中。若要啟用偵錯日誌記錄，請在執行外掛程式時指定 `-debug`。

一般疑難排解秘訣

- 新用戶最常遇到的問題是配置錯誤，導致插件無法初始化。發生這種情況時，您在嘗試安裝或啟用插件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絕大多數問題。

- 如果將 PV 掛載到容器時出現問題，請確保 `rpcbind`已安裝並正在運作。使用主機作業系統所需的軟體套件管理器，檢查 `rpcbind`是否正在運作。您可以透過執行 `systemctl status rpcbind`或其等效指令來檢查 rpcbind 服務的狀態。

管理多個 **Trident** 執行個體

當您需要同時使用多個儲存配置時，需要執行多個 Trident 實例。實現多重執行個體運作的關鍵在於，使用 `--alias`選項搭配容器化外掛程式，或使用 `--volume-driver`選項在主機上實例化 Trident 時，為它們指定不同的名稱。

Docker 管理外掛程式（版本 **1.13/17.03** 或更新版本）的步驟

1. 啟動第一個執行個體，指定別名和組態檔。

   ```
   docker plugin install --grant-all-permissions --alias silver
   netapp/trident-plugin:21.07 config=silver.json
   ```

2. 啟動第二個執行個體，指定不同的別名和組態檔。

   ```
   docker plugin install --grant-all-permissions --alias gold
   netapp/trident-plugin:21.07 config=gold.json
   ```

3. 建立磁碟區時，將別名指定為驅動程式名稱。

   例如，對於黃金磁碟區：

   ```
   docker volume create -d gold --name ntapGold
   ```

   例如，對於白銀交易量：

```
docker volume create -d silver --name ntapSilver
```

傳統方法的步驟（版本 **1.12** 或更早版本）

1. 使用自訂驅動程式 ID 透過 NFS 組態啟動外掛程式：

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. 使用自訂驅動程式 ID 和 iSCSI 組態啟動外掛程式：

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. 為每個驅動程式執行個體配置 Docker 磁碟區：

   例如，對於 NFS ：

```
docker volume create -d netapp-nas --name my_nfs_vol
```

   例如，對於 iSCSI ：

```
docker volume create -d netapp-san --name my_iscsi_vol
```

儲存組態選項

查看適用於您的 Trident 組態的組態選項。

全域配置選項

這些組態選項適用於所有 Trident 組態、無論使用何種儲存平台。

選項	說明	範例
`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`

選項	說明	範例
`storagePrefix`	磁碟區名稱的選用前置字元。預設值： `netappdvp_`。	`staging_`
`limitVolumeSize`	對磁碟區大小的限制（可選）。預設值： ""（不強制執行）	`10g`

請勿將 `storagePrefix`（包括預設值）用於 Element 後端。預設情況下，`solidfire-san`驅動程式會忽略此設定，並且不使用前綴。NetApp 建議為 Docker 磁碟區對應使用特定的 tenantID，或使用屬性資料，該資料包含 Docker 版本、驅動程式資訊以及來自 Docker 的原始名稱（以防使用了任何名稱修改）。

系統提供了預設選項，避免您在建立每個磁碟區時都進行指定。此 `size`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 設定部分。

選項	說明	範例
`size`	新磁碟區的可選預設大小。預設值：`1G`	`10G`

ONTAP 組態

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	說明	範例
`managementLIF`	ONTAP 管理 LIF 的 IP 位址。您可以指定完整網域名稱（FQDN）。	`10.0.0.1`
`dataLIF`	協定 LIF 的 IP 位址。 **ONTAP NAS** 驅動程式：NetApp 建議指定 `dataLIF`。如果未提供，Trident 將從 SVM 取得 dataLIF。您可以指定一個完全限定網域名稱（FQDN）用於 NFS 掛載操作，從而建立輪詢 DNS 以在多個 dataLIF 之間進行負載平衡。 **ONTAP SAN** 驅動程式：請勿指定 iSCSI 或 FC。Trident 使用"ONTAP Selective LUN Map"來發現建立多路徑工作階段所需的 iSCSI 或 FC LIF。如果明確定義 `dataLIF`，則會產生警告。	`10.0.0.2`

選項	說明	範例
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稱	vsadmin
password	連接儲存裝置的密碼	secret
aggregate	用於配置的 Aggregate （選用；如果設定，則必須指派給 SVM ）。對於 ontap-nas-flexgroup 驅動程式，此選項將被忽略。指派給 SVM 的所有 Aggregate 都將用於配置 FlexGroup Volume 。	aggr1
limitAggregateUsage	選用，如果使用率超過此百分比，則佈建失敗	75%
nfsMountOptions	對 NFS 掛載選項進行精細控制；預設值為 "-o nfsvers=3"。僅適用於 **`ontap-nas`**和 **`ontap-nas-economy`**驅動程式 "請在此處查看 NFS 主機組態資訊"。	-o nfsvers=4
igroupName	Trident 建立並管理每個節點 igroups 為 netappdvp。 此值不能更改或省略。 僅適用於 **ontap-san driver**。	netappdvp
limitVolumeSize	最大可請求磁碟區大小。	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** 驅動程式，此選項允許自訂每個 **FlexVol** 的 **qtree** 數量上限。	300
sanType	*僅支援 ontap-san`驅動程式。*用於選擇 iSCSI 的 `iscsi 、NVMe/TCP 的 nvme`或透過 Fibre Channel （FC）的 SCSI 的 `fcp。	iscsi 如果為空

選項	說明	範例
limitVolumePoolSize	*僅支援 ontap-san-economy 和 ontap-san-economy 驅動程式。* 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 Volume 上都進行指定：

選項	說明	範例
spaceReserve	空間預留模式； none（精簡配置）或 volume（厚配置）	none
snapshotPolicy	要使用的 Snapshot 原則、預設為 none	none
snapshotReserve	Snapshot 保留百分比，預設為 " 以接受 ONTAP 預設值	10
splitOnClone	建立時將複本與其父項分離，預設為 false	false
encryption	在新磁碟區上啟用 NetApp Volume Encryption (NVE)；預設為 false。若要使用此選項，必須在叢集上取得 NVE 授權並啟用 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。 如需更多資訊，請參閱："Trident 與 NVE 和 NAE 的運作方式"。	true
unixPermissions	已配置 NFS Volume 的 NAS 選項，預設為 777	777
snapshotDir	NAS 選項用於存取 .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，用於存取已配置的 NFS Volume。 NFS 支援 mixed`和 `unix`安全樣式。預設值為 `unix。	unix
fileSystemType	SAN 選項用於選擇檔案系統類型，預設為 ext4	xfs

選項	說明	範例
tieringPolicy	要使用的分層原則、預設為 none。	none
skipRecoveryQueue	刪除磁碟區時、繞過儲存設備中的還原佇列、並立即刪除磁碟區。	``

縮放選項

`ontap-nas`和 `ontap-san`驅動程式會為每個 Docker 磁碟區建立一個 ONTAP FlexVol。ONTAP 每個叢集節點最多支援 1000 個FlexVols，叢集最大支援 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`驅動程式是首選的 NAS 解決方案，因為FlexVols提供了額外的功能，例如 Docker 磁碟區粒度快照和複製。

如果您需要的 Docker 磁碟區數量超過了 FlexVol 限制所能容納的數量，請選擇 `ontap-nas-economy`或 `ontap-san-economy`驅動程式。

此 `ontap-nas-economy`驅動程式會在自動管理的 FlexVol 磁碟區集區中、以 ONTAP Qtree 的形式建立 Docker 磁碟區。Qtree 提供更高的擴充性、每個叢集節點最多可達 100,000 個、每個叢集最多可達 2,400,000 個、但會犧牲部分功能。此 `ontap-nas-economy`驅動程式不支援 Docker 磁碟區精細層級的快照或複製。

Docker Swarm 目前不支援 `ontap-nas-economy`驅動程式，因為 Docker Swarm 無法協調跨多個節點的磁碟區建立。

`ontap-san-economy`驅動程式會在自動管理的 FlexVol 磁碟區共用集區中、將 Docker 磁碟區建立為 ONTAP LUN。如此一來、每個 FlexVol 就不會僅限於一個 LUN、而且能為 SAN 工作負載提供更好的擴充性。視儲存陣列而定、ONTAP 每個叢集最多可支援 16384 個 LUN。由於磁碟區底層為 LUN、因此此驅動程式支援 Docker 磁碟區精細快照和複製。

選擇 `ontap-nas-flexgroup`驅動程式以提高單一磁碟區的平行處理能力，該磁碟區可擴充至 PB 級範圍，包含數十億個檔案。FlexGroups 的一些理想使用案例包括 AI/ML/DL、大數據和分析、軟體建置、串流、檔案儲存庫等。Trident 在配置 FlexGroup 磁碟區時會使用指派給 SVM 的所有 Aggregate。Trident 中的 FlexGroup 支援還需考慮以下幾點：

- 需要 ONTAP 版本 9.2 或更新版本。
- 截至撰寫本文時，FlexGroups 只支援 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 磁碟區不支援複製。

有關 FlexGroups 及適用於 FlexGroups 的工作負載的資訊，請參閱 "NetApp FlexGroup Volume 最佳實務做法與實作指南"。

若要在同一環境中獲得進階功能和大規模擴充，您可以執行多個 Docker Volume Plugin 執行個體，其中一個使

用 `ontap-nas`，另一個使用 `ontap-nas-economy`。

Trident 的自訂 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。當您在 Trident 後端組態中包含使用者名稱時，Trident 會使用您建立的 ONTAP 叢集角色來執行操作。

如需建立 Trident 自訂角色的詳細資訊，請參閱 "Trident 自訂角色產生器"。

使用 ONTAP CLI

1. 使用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 使用者建立使用者名稱：

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

3. 將角色對應至使用者：

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

使用 System Manager

在 ONTAP System Manager 中執行下列步驟：

1. 建立自訂角色 ：
   a. 若要在叢集層級建立自訂角色，請選取 **Cluster > Settings** 。

      （或）若要在 SVM 層級建立自訂角色、請選取 **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**。

   b. 選擇 **Users and Roles** 旁邊的箭頭圖示（→） 。
   c. 在 **Roles** 下選擇 **+Add**。
   d. 定義角色規則，然後點選 **Save** 。
2. 將角色對應到 **Trident** 使用者： + 在 **Users and Roles** 頁面上執行下列步驟：
   a. 在 **Users** 下方選擇 Add 圖示 **+**。
   b. 選擇所需的使用者名稱，然後在 **Role** 下拉式選單中選擇角色。
   c. 按一下 **Save** 。

如需更多資訊、請參閱下列頁面：

- "用於管理 ONTAP 的自訂角色" 或 "定義自訂角色"
- "使用角色和使用者"

ONTAP 組態檔範例

<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 SCSI over FC 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，也可以使用這些選項。

選項	說明	範例
`Endpoint`	https://<login>:<password>@<mvip>/json-rpc/<element-version>	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
`TenantName`	要使用的 SolidFire 租用戶（如果未找到則建立）	`docker`

選項	說明	範例
`InitiatorIFace`	將 iSCSI 流量限制在非預設介面時、請指定介面	`default`
`Types`	QoS 規範	請參閱以下範例
`LegacyNamePrefix`	升級版 Trident 安裝的前綴。如果您使用的 Trident 是 1.3.2 之前的版本，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 volume-name 方法對應的舊磁碟區。	`netappdvp-`

該 `solidfire-san` 驅動程式不支援 Docker Swarm。

Element 軟體組態檔範例

```
{
  "version": 1,
  "storageDriverName": "solidfire-san",
  "Endpoint": "https://admin:admin@192.168.160.3/json-rpc/8.0",
  "SVIP": "10.0.0.7:3260",
  "TenantName": "docker",
  "InitiatorIFace": "default",
  "Types": [
    {
      "Type": "Bronze",
      "Qos": {
        "minIOPS": 1000,
        "maxIOPS": 2000,
        "burstIOPS": 4000
      }
    },
    {
      "Type": "Silver",
      "Qos": {
        "minIOPS": 4000,
        "maxIOPS": 6000,
        "burstIOPS": 8000
      }
    },
    {
      "Type": "Gold",
      "Qos": {
        "minIOPS": 6000,
        "maxIOPS": 8000,
        "burstIOPS": 10000
      }
    }
  ]
}
```

已知問題和限制

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** 從舊版升級至 **20.10** 及更新版本會導致升級失敗，並出現 **no such file or directory** 錯誤。

因應措施

1. 停用外掛程式。

```
docker plugin disable -f netapp:latest
```

2. 移除外掛程式。

```
docker plugin rm -f netapp:latest
```

3. 透過提供額外 `config` 參數重新安裝插件。

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

磁碟區名稱長度至少為 2 個字元。

這是 Docker 用戶端的限制。用戶端會將單一字元的名稱解釋為 Windows 路徑 "請參閱錯誤 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非磁碟區 ID 做為其唯一的磁碟區識別碼。
- 磁碟區要求會同時傳送到 Swarm 叢集中的每個節點。
- Volume 外掛程式（包括 Trident）必須在 Swarm 叢集中的每個節點上獨立執行。由於 ONTAP 的運作方式以及 `ontap-nas` 和 `ontap-san` 驅動程式的功能，它們是唯一能夠在這些限制內運作的外掛程式。

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建立大量磁碟區，而沒有明確的「贏家」；例如、Element 具有允許磁碟區具有相同名稱但 ID 不同的功能。

NetApp 已向 Docker 團隊提供回饋，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 Aggregate，則 ONTAP 不會配置第二個 FlexGroup。

最佳實務做法與建議

部署

部署 Trident 時、請使用此處列出的建議。

部署到專用命名空間

"命名空間" 提供不同應用程式之間的管理隔離，並構成資源共享的障礙。例如，一個命名空間中的 PVC 無法從另一個命名空間使用。Trident 為 Kubernetes 叢集中的所有命名空間提供 PV 資源，因此利用具有提升權限的服務帳戶。

此外，存取 Trident pod 可能使用戶能夠存取儲存系統憑證和其他敏感資訊。請務必確保應用程式使用者和管理應用程式無法存取 Trident 物件定義或 pod 本身。

使用配額和範圍限制來控制儲存使用量

Kubernetes 具有兩項特性，二者結合使用可提供強大的機制來限制應用程式的資源消耗。 "儲存配額機制"可讓管理員以每個命名空間為基礎，實施全域以及儲存類別特定的容量和物件數量消耗限制。此外，使用 "範圍限制" 可確保在將請求轉送給配置器之前，PVC 請求的值均在最小值和最大值範圍內。

這些值是基於命名空間定義的，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。請參閱此處以取得相關資訊 "如何利用配額"。

儲存組態

NetApp 產品組合中的每個儲存平台都具有獨特的功能，無論應用程式是否容器化，都能從中受益。

平台概覽

Trident 可與 ONTAP 和 Element 搭配使用。雖然沒有哪個平台比其他平台更適合所有應用程式和場景，但在選擇平台時，應考慮應用程式的需求以及管理設備的團隊的需求。

您應該遵循所用協定對應的主機作業系統的最佳實務。此外，您還可以考慮在後端、儲存類別和 PVC 設定中融入應用程式最佳實務（如有），以優化特定應用程式的儲存。

ONTAP 和 Cloud Volumes ONTAP 最佳實務做法

了解為 Trident 配置 ONTAP 和 Cloud Volumes ONTAP 的最佳實務做法。

以下建議是為容器化工作負載配置 ONTAP 的指導原則，這些工作負載會使用由 Trident 動態配置的磁碟區。每項建議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 專用於應用程式可實現權限委派，並支援應用限制資源消耗的最佳實務做法。

SVM 的管理有多種選擇：

- 在後端組態中提供叢集管理介面、適當的認證資料、並指定 SVM 名稱。
- 使用 ONTAP System Manager 或 CLI 為 SVM 建立專用管理介面。
- 與 NFS 資料介面共用管理角色。

無論哪種情況，介面都應在 DNS 中配置，並且在配置 Trident 時應使用 DNS 名稱。這有助於簡化某些災難復原場景，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 並無優劣之分，但您應確保網路安全策略與所選方案相符。無論如何，管理 LIF 都應可透過 DNS 訪問，以便在 "SVM-DR"與 Trident 搭配使用時實現最大的靈活性。

限制磁碟區數量上限

ONTAP 儲存系統存在最大磁碟區數限制，此限制因軟體版本和硬體平台而異。請參閱 "NetApp Hardware Universe"以了解您特定平台和 ONTAP 版本的確切限制。當磁碟區數達到上限時，不僅 Trident 的資源配置作業會失敗，所有儲存請求也會失敗。

Trident 的 `ontap-nas`和 `ontap-san`驅動程式會為每個建立的 Kubernetes 持久性磁碟區（PV）配置一個 FlexVolume。 `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）。 `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）。為防止 Trident 佔用儲存系統上所有可用磁碟區，您應該設定 SVM 的限制。您可以透過命令列進行此操作：

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

`max-volumes` 的值會根據您環境的幾個特定條件而有所不同：

- 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磁碟區數量
- Kubernetes 應用程式預計使用的持續磁碟區數量

該 `max-volumes`值是 ONTAP 叢集中所有節點上配置的磁碟區總數、而非單一 ONTAP 節點上的磁碟區數。因此、您可能會遇到某些情況、其中 ONTAP 叢集節點的 Trident 配置磁碟區數可能遠多於或遠少於其他節點。

例如，一個雙節點 ONTAP 叢集最多可以託管 2000 個 FlexVol 磁碟區。將最大磁碟區數設為 1250 看起來非常合理。但是，如果僅 "Aggregate"將來自一個節點的 Aggregate 分配給 SVM，或者來自一個節點的 Aggregate 無法進行配置（例如，由於容量不足），則另一個節點將成為所有 Trident 配置磁碟區的目標。這意味著該節點的磁碟區限制可能在達到 `max-volumes`值之前就已經達到，從而影響 Trident 以及使用該節點的其他磁碟區操作。您可以透過確保將叢集中每個節點的 **Aggregate** 以相等的數量分配給 **Trident** 使用的 **SVM** 來避免這種情況。

複製磁碟區

NetApp Trident 在使用 `ontap-nas`、 `ontap-san`和 `solidfire-san`儲存驅動程式時支援複製磁碟區。使用 `ontap-nas-flexgroup`或 `ontap-nas-economy`驅動程式時，不支援複製。從現有磁碟區建立新磁碟區將建立新的快照。

避免克隆與不同 StorageClass 關聯的 PVC。在同一 StorageClass 內執行克隆操作，以確保相容性並防止意外行為。

限制 **Trident** 建立的磁碟區大小上限

若要設定 Trident 可建立的磁碟區最大大小，請在您的 `limitVolumeSize` 定義中使用 `backend.json` 參數。

除了控制儲存陣列的磁碟區大小之外，還應該利用 Kubernetes 功能。

限制 **Trident** 建立的 **FlexVols** 大小上限

若要為作為 ontap-san-economy 和 ontap-nas-economy 驅動程式資源池所使用的 FlexVols 設定最大大小，請在您的 `limitVolumePoolSize` 定義中使用 `backend.json` 參數。

設定 **Trident** 使用雙向 **CHAP**

您可以在後端定義中指定 CHAP 發起方和目標的使用者名稱和密碼，並讓 Trident 在 SVM 上啟用 CHAP。透過後端設定中的 `useCHAP` 參數，Trident 可使用 CHAP 對 ONTAP 後端的 iSCSI 連線進行驗證。

建立並使用 **SVM QoS** 原則

透過對 SVM 應用 ONTAP QoS 原則，可限制 Trident 已配置磁碟區可使用的 IOPS 數量。這有助於 "阻止霸凌行為"防止失控容器影響 Trident SVM 以外的工作負載。

您只需幾個步驟即可為 SVM 建立 QoS 策略。如需有關 ONTAP 的最準確資訊、請參閱您所用 ONTAP 版本的文件。以下範例建立了一個 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 版本支援，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 層級的原則不相容。

分配給容器化工作負載的 IOPS 數量取決於諸多因素。其中包括：

- 其他使用儲存陣列的工作負載。如果存在與 Kubernetes 部署無關的其他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受到意外的不利影響。
- 預期工作負載將在容器中運作。如果具有高 IOPS 要求的工作負載將在容器中運作，則低 QoS 原則會導致不佳的體驗。

需要注意的是，在 SVM 層級指派的 QoS 策略會導致佈建給該 SVM 的所有磁碟區共用同一個 IOPS 資源池。如果一個或少數幾個容器化應用程式的 IOPS 需求很高，則可能會對其他容器化工作負載造成嚴重影響。在這種情況下，您可能需要考慮使用外部自動化工具來為每個磁碟區指派 QoS 策略。

只有當您的 ONTAP 版本低於 9.8 時，才應將 QoS 原則群組指派給 SVM。

為 **Trident** 建立 **QoS** 原則群組

服務品質（QoS）可確保關鍵工作負載的效能不會因競爭工作負載而降低。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 的更多資訊，請參閱 "透過 QoS 保證處理量"。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儲存池或後端中建立的每個磁碟區。

ONTAP 有兩種 QoS 原則群組：傳統和自適應。傳統原則群組提供固定的最大（或在較新版本中為最小）IOPS 處理量。自適應 QoS 會根據工作負載大小自動調整處理量，在工作負載大小變化時維持 IOPS 與 TB|GB 的比率。當您在大型部署中管理數百或數千個工作負載時，這提供了顯著的優勢。

建立 QoS 策略群組時、請考慮以下事項：

- 您應該在後端設定的 `defaults`區塊中設定 `qosPolicy`鍵。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
```

- 您應該按磁碟區套用原則群組，以便每個磁碟區都能獲得原則群組指定的全部處理量。不支援共享原則群組。

如需 QoS 策略群組的詳細資訊，請參閱 "ONTAP 指令參考"。

限制 **Kubernetes** 叢集成員的儲存資源存取

限制對 Trident 所建立的 NFS 磁碟區、iSCSI LUN 和 FC LUN 的存取，是 Kubernetes 部署安全態勢的關鍵組成部分。這樣做可以防止非 Kubernetes 叢集成員的主機存取這些磁碟區，從而避免資料被意外修改。

理解命名空間是 Kubernetes 中資源的邏輯邊界至關重要。雖然同一命名空間內的資源可以共享，但需要注意的

是，它們之間不支援跨命名空間存取。這意味著，即使 PV 是全域對象，當綁定到 PVC 時，也只有位於相同命名空間的 Pod 才能存取它們。務必確保在適當情況下使用命名空間來實現資源隔離。

在 Kubernetes 環境中，大多數組織對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儲存，但這些儲存並非容器所期望的。 "命名空間" 旨在防止此類安全漏洞。然而，特權容器是個例外。

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。這些權限預設不會被拒絕，因此請務必使用 "Pod 安全性原則"停用此功能。

對於需要同時從 Kubernetes 和外部主機存取的磁碟區，應採用傳統方式管理儲存設備，即由管理員建立 PV，而不是由 Trident 管理。這樣可以確保僅在 Kubernetes 和外部主機都已中斷連線且不再使用該磁碟區時才銷毀儲存磁碟區。此外，還可以套用自訂匯出原則，從而允許從 Kubernetes 叢集節點和 Kubernetes 叢集外部的目標伺服器存取。

對於具有專用基礎架構節點（例如 OpenShift）或其他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架構節點的服務（例如 OpenShift Metrics 和 Logging 服務）以及部署到非基礎架構節點的標準應用程式建立匯出策略。

使用專用的匯出原則

您應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匯出策略。這樣，Trident 將對其配置的磁碟區的存取限制在 Kubernetes 叢集中的節點上，並簡化節點的新增/刪除操作。

或者、您也可以手動建立匯出原則、並在其中填入一或多個匯出規則、以處理每個節點存取要求：

- 使用 `vserver export-policy create` ONTAP CLI 指令建立匯出原則。
- 使用 `vserver export-policy rule create` ONTAP CLI 命令將規則新增至匯出原則。

執行這些命令可以限制哪些 Kubernetes 節點可以存取資料。

停用應用程式 **SVM** 的 `showmount`

`showmount` 功能可讓 `NFS` 用戶端向 `SVM` 查詢可用 `NFS` 匯出的清單。部署至 `Kubernetes` 叢集的 `Pod` 可以針對 `SVM` 發出 `showmount -e` 命令、並接收可用掛載的清單、包括其無權存取的掛載。雖然這本身並不構成安全性危害、但它確實提供了不必要的資訊、可能有助於未獲授權的使用者連線至 `NFS` 匯出。

您應該使用 SVM 層級 ONTAP CLI 命令來停用 `showmount`：

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire 最佳實踐

了解為 Trident 配置 SolidFire 儲存設備的最佳實務做法。

建立 **SolidFire** 帳戶

每個 SolidFire 帳戶代表一個唯一的磁碟區擁有者，並擁有自己的一組 Challenge-Handshake Authentication Protocol (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某個帳戶的磁碟區。一個帳戶最多可以分配兩千個磁碟區，但一個磁碟區只能屬於一個帳戶。

建立 **QoS** 原則

如果您想要建立並儲存可套用於多個磁碟區的標準化服務品質設定，請使用 SolidFire 服務品質（QoS）原則。

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（Min IOPS、Max IOPS 和 Burst IOPS），可以確保每個磁碟區的效能。

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	磁碟區的保證效能層級。	50	50	15000
最大 IOPS	效能不會超過此限制。	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 Max IOPS 和 Burst 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。

區塊大小和頻寬對 IOPS 數量有直接影響。隨著區塊大小增加，系統會將頻寬提高到處理較大區塊大小所需的層級。隨著頻寬增加，系統能夠達到的 IOPS 數量會減少。如需 QoS 和效能的詳細資訊，請參閱 "SolidFire 服務品質"。

SolidFire 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 協定對主機進行身份驗證，以連接到後端伺服器。磁碟區存取群組控制對其配置的磁碟區的存取。NetApp 建議使用 CHAP 進行身份驗證，因為它更簡單且沒有擴展限制。

Trident 配備增強型 CSI 配置程式的 Trident 支援使用 CHAP 驗證。VAG 只能在傳統的非 CSI 作業模式下使用。

CHAP 驗證（驗證啟動器是否為預期的磁碟區使用者）僅支援以帳戶為基礎的存取控制。如果您使用 CHAP 進行驗證，有兩個選項可用：單向 CHAP 和雙向 CHAP。單向 CHAP 使用 SolidFire 帳戶名稱和啟動器密碼來驗證磁碟區存取。雙向 CHAP 選項提供最安全的磁碟區驗證方式，因為磁碟區透過帳戶名稱和啟動器密碼來驗證主機，然後主機透過帳戶名稱和目標密碼來驗證磁碟區。

但是，如果無法啟用 CHAP 且需要 VAG，請建立存取群組並將主機啟動器和磁碟區新增至該存取群組。新增至存取群組的每個 IQN 都可以存取群組中的每個磁碟區，無論是否使用 CHAP 驗證。如果 iSCSI 啟動器設定為使用 CHAP 驗證，則使用基於帳戶的存取控制。如果 iSCSI 啟動器未設定為使用 CHAP 驗證，則使用 Volume

Access Group 存取控制。

哪裡可以找到更多資訊？

以下列出了一些最佳實踐文件。搜尋 "NetApp 資料庫"以取得最新版本。

ONTAP

- "NFS 最佳實務與實施指南"
- "SAN 管理"（適用於 iSCSI ）
- "RHEL 的 iSCSI Express 組態"

Element 軟體

- "配置 SolidFire for Linux"

NetApp HCI

- "NetApp HCI 部署先決條件"
- "存取 NetApp Deployment Engine"

應用程式最佳實務資訊

- "ONTAP 上 MySQL 的最佳實務做法"
- "MySQL 在 SolidFire 上的最佳實踐"
- "NetApp SolidFire 和 Cassandra"
- "Oracle 最佳實務做法 SolidFire"
- "PostgreSQL 在 SolidFire 上的最佳實踐"

並非所有應用程式都有具體指南，重要的是與您的 NetApp 團隊合作，並使用 "NetApp 資料庫"尋找最新文件。

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。

驅動程式選擇與部署

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 後端驅動程式之間的差異在於所使用的傳輸協定以及磁碟區在儲存系統上的配置方式。因此，在決定部署哪個驅動程式時，請務必仔細考慮。

從更高層次來看，如果您的應用程式包含需要共用儲存的元件（多個 Pod 存取同一個 PVC），則基於 NAS 的

驅動程式是預設選擇；而基於區塊的 iSCSI 驅動程式則符合非共用儲存的需求。協議的選擇應基於應用程式的需求以及儲存和基礎架構團隊的熟悉程度。一般來說，對於大多數應用程式而言，兩者之間的差異很小，因此通常取決於是否需要共用儲存（即多個 Pod 需要同時存取）。

可用的 ONTAP 後端驅動程式包括：

- `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 可配置的 qtree 數量（預設值為 200）。
- `ontap-nas-flexgroup`：每個 PV 都配置為完整的 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 內部的一個 LUN。
- `ontap-san-economy`：每個已佈建的 PV 都是一個 LUN，每個 FlexVolume 可配置的 LUN 數量（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，下表中的並非所有功能都透過 Trident 公開。如果需要該功能，則必須由儲存管理員在配置後套用其中一些功能。上標腳註區分了每個功能和驅動程式的功能。

ONTAP NAS 驅動程式	快照	複本	動態匯出原則	多重附加	**QoS**	調整大小	複寫
`ontap-nas`	是的	是的	是	是的	是	是的	是
`ontap-nas-economy`	NO [3]	NO [3]	是	是的	NO [3]	是的	NO [3]
`ontap-nas-flexgroup`	是	否	是	是的	是	是的	是

Trident 為 ONTAP 提供 2 個 SAN 驅動程式，其功能如下所示。

ONTAP SAN 驅動程式	快照	複本	多重附加	雙向 **CHAP**	**QoS**	調整大小	複寫
`ontap-san`	是的	是的	是	是的	是	是的	是
`ontap-san-economy`	是的	是的	是	是的	NO [3]	是的	NO [3]

> 以上表格的註腳：Yes [1]：非 Trident 管理 Yes [2]：由 Trident 管理，但不支援 PV 粒度 NO [3]：非 Trident 管理且不支援 PV 粒度 Yes [4]：支援原始區塊磁碟區 Yes [5]：由 Trident 支援

非 PV 粒度的功能會套用至整個 FlexVolume 和所有 PV（即共享 FlexVols 中的 qtree 或 LUN），並將共用一個共同的排程。

如上表所示、 `ontap-nas`和 `ontap-nas-economy`之間的許多功能都是相同的。但是、由於 `ontap-nas-economy`驅動程式限制了以每個 `PV` 精細度控制排程的能力、這可能會特別影響您的災難恢復和備份規劃。對於希望在 `ONTAP` 儲存設備上利用 `PVC` 複製功能的開發團隊、只有在使用 `ontap-nas`、`ontap-san`或 `ontap-san-economy`驅動程式時才有可能。

該 `solidfire-san` 驅動程式還能夠克隆 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，適用於各種使用案例，包括檔案共用和區塊層級儲存，並支援 NAS 和 SAN 協定（NFS、SMB / CIFS 和 iSCSI）。Cloud Volumes ONTAP 的相容驅動程式為 `ontap-nas`、`ontap-nas-economy`、`ontap-san`和 `ontap-san-economy`。這些適用於 Cloud Volumes ONTAP for Azure、Cloud Volumes ONTAP for GCP。

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 特性、效能和管理功能，同時享受在 AWS 上儲存資料的簡易性、敏捷性、安全性和可擴充性。FSx for ONTAP 支援許多 ONTAP 檔案系統特性和管理 API。Cloud Volume ONTAP 的相容驅動程式包括 `ontap-nas`、`ontap-nas-economy`、`ontap-nas-flexgroup`、`ontap-san` 和 `ontap-san-economy`。

NetApp HCI/SolidFire 後端驅動程式

`solidfire-san`與 `NetApp HCI/SolidFire` 平台配合使用的驅動程式可協助管理員根據 `QoS` 限制為 `Trident` 配置 `Element` 後端。如果您希望設計後端以對 `Trident` 配置的磁碟區設定特定的 `QoS` 限制，請在後端檔案中使用 `type`參數。管理員也可以使用 `limitVolumeSize`參數限制儲存上可建立的磁碟區大小。目前， `solidfire-san`驅動程式不支援 `Element` 儲存功能，例如磁碟區調整大小和磁碟區複製。這些操作需要透過 `Element Software Web UI` 手動完成。

SolidFire 驅動程式	快照	複本	多重附加	CHAP	QoS	調整大小	複寫
solidfire-san	是的	是的	是 [2]	是的	是的	是的	是

> 註腳：Yes [1]：不由 Trident 管理 Yes [2]：支援原始區塊磁碟區

Azure NetApp Files 後端驅動程式

Trident 使用 `azure-netapp-files` 驅動程式來管理 "Azure NetApp Files" 服務。

有關此驅動程式及其配置方法的更多資訊，請參見 "Azure NetApp Files 的 Trident 後端組態"。

Azure NetApp Files 驅動程式	快照	複本	多重附加	QoS	展開	複寫
azure-netapp-files	是的	是的	是的	是的	是的	是

> 註腳：是註腳：1[]：非由 Trident 管理

儲存類別設計

需要配置並套用個別儲存類別，才能建立 Kubernetes Storage Class 物件。本節將討論如何為您的應用程式設計儲存類別。

特定後端使用率

在特定的儲存類別物件中可以使用篩選功能來確定要與該特定儲存類別一起使用的儲存池或儲存池集合。可以在儲存類別中設定三組過濾器： `storagePools`、 `additionalStoragePools` 和/或 `excludeStoragePools`。

`storagePools` 參數有助於將儲存空間限制在符合任何指定屬性的儲存池集合中。 `additionalStoragePools` 參數用於擴展 Trident 用於資源配置的儲存池集合，以及透過屬性和 `storagePools` 參數選擇的儲存池集合。您可以單獨使用任一參數，也可以同時使用兩個參數，以確保選擇合適的儲存池集合。

`excludeStoragePools`參數用於專門排除符合屬性的已列出資源池集合。

模擬 **QoS** 原則

如果您希望設計儲存類別來模擬服務品質政策，請建立一個儲存類別，並將 `media`屬性設為 `hdd`或 `ssd`。根據儲存類別中提及的 `media`屬性，`Trident` 將選擇適當的後端來提供 `hdd`或 `ssd`集合體以符合媒體屬性，然後將磁碟區的配置導向至特定的集合體。因此，我們可以建立一個名為 `PREMIUM` 的儲存類別，其 `media`屬性設為 `ssd`，這可以歸類為 PREMIUM QoS 政策。我們可以建立另一個名為 STANDARD 的儲存類別，其媒體屬性設為 `hdd`，這可以歸類為 STANDARD QoS 政策。我們也可以使用儲存類別中的 `IOPS` 屬性將配置重新導向至 Element 設備，這可以定義為 QoS 政策。

根據特定功能使用後端

儲存類別可以設計用於指導在特定後端上進行磁碟區配置，這些後端啟用了精簡配置、厚配置、快照、複製和加密等功能。若要指定要使用的儲存，請建立儲存類別，並指定啟用了所需功能的相應後端。

虛擬資源池

所有 Trident 後端均可使用虛擬資源池。您可以使用 Trident 提供的任何驅動程式、為任何後端定義虛擬資源池。

虛擬池允許管理員在後端之上建立一層抽象層，並透過 Storage Classes 來引用這些後端，從而提高磁碟區在後端上的部署靈活性和效率。不同的後端可以使用相同的服務類別。此外，可以在同一個後端上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 Storage Class 時，Trident 會選擇一個與所有選擇器標籤都相符的後端來部署磁碟區。如果 Storage Class 選擇器標籤符合多個儲存池，Trident 將從中選擇其中一個來配置磁碟區。

虛擬資源池設計

在建立後端時，您通常可以指定一組參數。管理員無法使用相同的儲存憑證並搭配不同的參數組來建立另一個後端。隨著虛擬資源池的引入，這個問題已經得到緩解。虛擬資源池是在後端與 Kubernetes Storage Class 之間引入的一層抽象，讓管理員可以定義參數及標籤，這些標籤可以透過 Kubernetes Storage Classes 作為選擇器來引用，並且與後端無關。虛擬資源池可以為所有支援的 NetApp 後端與 Trident 一起定義。該清單包括 SolidFire/NetApp HCI、ONTAP，以及 Azure NetApp Files。

定義虛擬資源池時，建議不要嘗試在後端定義中重新排列現有虛擬資源池的順序。此外，也不建議編輯 / 修改現有虛擬資源池的屬性，而是定義一個新的虛擬資源池。

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 Cloud Volume Service for Azure NetApp Files 的虛擬池實作，讓我們來探討如何設定不同的服務類別。使用多個標籤設定 Azure NetApp Files 後端，代表不同的效能等級。將 `servicelevel`方面設定為適當的效能等級，並在每個標籤下新增其他所需的方面。現在建立不同的 Kubernetes Storage Classes，對應到不同的虛擬池。使用 `parameters.selector`欄位，每個 StorageClass 會指定哪些虛擬池可用於託管磁碟區。

指定特定的層面集

可以從單一儲存後端設計多個具有特定屬性的虛擬儲存池。為此、請為後端配置多個標籤、並在每個標籤下設定所需的屬性。現在使用 `parameters.selector` 欄位建立不同的 Kubernetes Storage Classes、以對應到不同的虛擬儲存池。在後端配置的磁碟區將具有所選虛擬儲存池中定義的屬性。

影響儲存資源配置的 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 配置決策過程。

存取模式

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選擇用於託管儲存請求的後端。

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。

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
iSCSI	是的	是的	是（Raw block）
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，則不會建立任何磁碟區。因此，請求者應使用適合其應用程式的存取模式。

磁碟區作業

修改持續磁碟區

持久性磁碟區在 Kubernetes 中除兩種例外情況外，都是不可變物件。創建後，可以修改其回收策略和大小。但是，這並不妨礙在 Kubernetes 外部修改磁碟區的某些方面。這樣做可能有助於為特定應用程式定製磁碟區，確保容量不會被意外耗盡，或者只是為了將磁碟區遷移到不同的儲存控制器。

目前，Kubernetes 內建的配置器不支援對 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 的連線詳細資料在建立後無法修改。

建立隨需磁碟區快照

Trident 支援隨需建立 Volume 快照，並可使用 CSI 架構從快照建立 PVC。快照提供了一種便捷的方法來維護資料的時間點複本，並且其生命週期獨立於 Kubernetes 中的來源 PV。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PersistentVolumes。為此、只需建立PersistentVolumeClaim並提及 `datasource`作為需要從中建立磁碟區的所需快照。Trident 將使用快照中存在的資料建立磁碟區來處理此 PVC。透過此功能、可以跨區域複製資料、建立測試環境、完整替換損壞或毀損的正式作業磁碟區、或擷取特定檔案和目錄並將其傳輸到另一個附加的磁碟區。

在叢集中移動磁碟區

儲存管理員可以在 ONTAP 叢集中無中斷地將磁碟區在聚合和控制器之間移動，而不會對儲存使用者造成任何影響。只要目標聚合是 Trident 使用的 SVM 可以存取的聚合，此操作就不會影響 Trident 或 Kubernetes 叢集。需要注意的是，如果聚合是新添加到 SVM 的，則需要透過將其重新新增至 Trident 來重新整理後端。這將觸發 Trident 重新清點 SVM，以便識別新聚合。

然而，Trident 不支援在後端之間自動移動磁碟區。這包括在同一叢集中的 SVM 之間、叢集之間，或移動到不同的儲存平台（即使該儲存系統已連接到 Trident）。

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 。

擴充磁碟區

Trident 支援調整 NFS、iSCSI 和 FC PV 的大小。這使用戶能夠直接透過 Kubernetes 層調整磁碟區的大小。所有主流 NetApp 儲存平台（包括 ONTAP 和 SolidFire/NetApp HCI 後端）均支援磁碟區擴充。為了允許日後擴展，請在與磁碟區關聯的 StorageClass 中將 `allowVolumeExpansion` 設定為 `true`。每當需要調整 Persistent Volume 的大小時，請將 Persistent Volume Claim 中的 `spec.resources.requests.storage` 註解編輯為所需的磁碟區大小。Trident 將自動處理儲存叢集上的磁碟區大小調整。

將現有磁碟區匯入 **Kubernetes**

磁碟區匯入功能可將現有儲存磁碟區匯入 Kubernetes 環境。目前 `ontap-nas`、 `ontap-nas-flexgroup`、 `solidfire-san`和 `azure-netapp-files`驅動程式支援此功能。此功能在將現有應用程式移植到 Kubernetes 或災難恢復情境中非常實用。

使用 ONTAP 和 `solidfire-san`驅動程式時，請使用指令 `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。匯入磁碟區指令中使用的 PVC YAML 或 JSON 檔案指向一個儲存類別，該儲存類別會將 Trident 識別為佈建程式。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能夠區分它們。

如果使用 `azure-netapp-files`驅動程式，請使用命令 `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml`將磁碟區匯入 Kubernetes 以便由 Trident 管理。這樣可以確保磁碟區參照的唯一性。

執行上述指令後、 Trident 將在後端尋找磁碟區並讀取其大小。它會自動新增（必要時會覆蓋）已配置 PVC 的磁碟區大小。然後、 Trident 會建立新的 PV 、 Kubernetes 會將 PVC 綁定到該 PV 。

如果容器的部署需要特定的匯入 PVC，則該容器將保持待處理狀態，直到透過磁碟區匯入流程綁定 PVC/PV 對。PVC/PV 對綁定後，如果沒有其他問題，容器即可啟動。

登錄服務

註冊表的部署和管理儲存已在 "netapp.io" 的 "部落格" 中記錄。

日誌服務

與其他 OpenShift 服務一樣，日誌服務也使用 Ansible 進行部署，設定參數由提供給 playbook 的清單檔案（也稱為 hosts）提供。將介紹兩種安裝方法：在初始 OpenShift 安裝期間部署日誌服務，以及在 OpenShift 安裝完成後部署日誌服務。

從 Red Hat OpenShift 3.9 版本開始，官方文件建議不要使用 NFS 作為日誌服務，因為有資料損壞的風險。這是基於 Red Hat 對其產品的測試結果。ONTAP NFS 伺服器不存在這些問題，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，但需要注意的是，這兩種協定在使用 NetApp 平台時都能很好地工作，如果您偏好 NFS，則沒有理由避免使用 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`為 `true`以防止安裝程序失敗。

開始使用

日誌服務可以選擇性地部署在應用程式和 OpenShift 叢集的核心操作上。如果您選擇部署操作日誌服務，透過指定變數 `openshift_logging_use_ops` 為 `true`，系統將建立兩個服務實例。控制操作日誌實例的變數包含「ops」參數，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 變數至關重要，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。您可以在 "Red Hat OpenShift 日誌文檔" 中找到其他選項，這些選項應根據您的部署情況進行檢視、配置和使用。

下表中的變數將使 Ansible playbook 使用提供的詳細資訊為日誌服務建立 PV 和 PVC 。與 OpenShift 安裝後使用元件安裝 playbook 相比，此方法彈性要差得多，但如果您已有可用的磁碟區，則此方法也是一種選擇。

變數	詳細資料
`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 值。
`openshift_logging_storage_nfs_directory`	NFS 匯出的掛載路徑。例如，如果磁碟區已連接為 `/openshift_logging`，則此變數應使用該路徑。
`openshift_logging_storage_volume_name`	要建立的 PV 名稱，例如 `pv_ose_logs`。
`openshift_logging_storage_volume_size`	NFS 匯出的大小，例如 `100Gi`。

如果您的 OpenShift 叢集已在運行，並且 Trident 已部署和配置，則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。

變數	詳細資料
`openshift_logging_es_pvc_dynamic`	設定為 true 以使用動態配置磁碟區。

變數	詳細資料
`openshift_logging_es_pvc_storage_class_name`	PVC 中將使用的儲存類別名稱。
`openshift_logging_es_pvc_size`	PVC 中請求的磁碟區大小。
`openshift_logging_es_pvc_prefix`	日誌記錄服務使用的 PVC 前置碼。
`openshift_logging_es_ops_pvc_dynamic`	設定為 `true` 以使用動態配置的磁碟區來執行運維日誌實例。
`openshift_logging_es_ops_pvc_storage_class_name`	運維日誌執行個體的儲存類別名稱。
`openshift_logging_es_ops_pvc_size`	ops 實例的磁碟區請求大小。
`openshift_logging_es_ops_pvc_prefix`	ops 執行個體 PVC 的前置詞。

部署日誌堆疊

如果您在初始 OpenShift 安裝過程中部署日誌記錄，則只需遵循標準部署流程即可。Ansible 將配置並部署所需的服務和 OpenShift 對象，以便在 Ansible 完成後服務即可使用。

但是，如果您在初始安裝後進行部署，則需要使用 Ansible 元件 playbook。此過程可能會因 OpenShift 版本而略有不同，因此請務必閱讀並遵循"Red Hat OpenShift Container Platform 3.11 文件"適用於您版本的相關說明。

指標服務

指標服務為管理員提供有關 OpenShift 叢集狀態、資源利用率和可用性的重要資訊。它對於 Pod 自動擴縮容功能也至關重要，許多組織也利用指標服務中的資料來實施成本分攤和/或成本展示應用。

與日誌服務和整個 OpenShift 類似，指標服務也使用 Ansible 進行部署。此外，與日誌服務類似，指標服務既可以在叢集初始設定期間部署，也可以在叢集運行後使用元件安裝方法進行部署。下表包含配置指標服務持久性儲存時的重要變數。

下表僅包含與指標服務相關的儲存配置變數。文件中還提供了許多其他選項，您應該根據部署情況進行查閱、配置和使用。

變數	詳細資料
`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 SVM 的 dataLIF。
`openshift_metrics_storage_nfs_directory`	NFS 匯出的掛載路徑。例如，如果磁碟區已連接為 `/openshift_metrics`，則此變數應使用該路徑。
`openshift_metrics_storage_volume_name`	要建立的 PV 的名稱，例如 `pv_ose_metrics`。
`openshift_metrics_storage_volume_size`	NFS 匯出的大小，例如 `100Gi`。

如果您的 OpenShift 叢集已在運行，並且 Trident 已部署和配置，則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。

變數	詳細資料
`openshift_metrics_cassandra_pvc_prefix`	用於指標 PVC 的前綴。
`openshift_metrics_cassandra_pvc_size`	要求的磁碟區大小。
`openshift_metrics_cassandra_storage_type`	用於指標的儲存類型，必須設定為 dynamic，以便 Ansible 建立具有適當儲存類別的 PVC。
`openshift_metrics_cassanda_pvc_storage_class_name`	要使用的儲存類別名稱。

部署指標服務

在 hosts/inventory 檔案中定義好對應的 Ansible 變數後，即可使用 Ansible 部署服務。如果在 OpenShift 安裝時部署，PV 將自動建立並使用。如果使用元件 playbook 進行部署，在 OpenShift 安裝完成後，Ansible 會建立所需的 PVC，並在 Trident 為其配置儲存後部署服務。

上述變數和部署流程可能會隨 OpenShift 的每個版本而變化。請務必查看並遵循"Red Hat OpenShift 部署指南"您所用版本的相關說明，以便根據您的環境進行配置。

資料保護與災難恢復

了解使用 Trident 建立的 Trident 和磁碟區的保護和復原選項。您應該為每個具有持久性要求的應用程式製定資料保護和復原策略。

Trident 複寫與還原

您可以建立備份，以便在災難發生時還原 Trident。

Trident 複寫

Trident 使用 Kubernetes CRD 來儲存和管理自己的狀態，並使用 Kubernetes 叢集 etcd 來儲存其中繼資料。

步驟

1. 使用 "Kubernetes：備份 etcd 叢集" 備份 Kubernetes 叢集 etcd。
2. 將備份檔案放置在 FlexVol 磁碟區上

> NetApp 建議您使用 SnapMirror 關係，將 FlexVol 所在的 SVM 保護到另一個 SVM。

Trident 恢復

使用 Kubernetes CRD 和 Kubernetes 叢集 etcd 快照、您可以還原 Trident。

步驟

1. 從目的地 SVM 掛載包含 Kubernetes etcd 資料檔案和憑證的磁碟區到將設定為主節點的主機。
2. 複製 Kubernetes 叢集下的所有必要憑證 `/etc/kubernetes/pki` 以及 etcd 成員檔案 `/var/lib/etcd`。
3. 使用 "Kubernetes：還原 etcd 叢集" 從 etcd 備份還原 Kubernetes 叢集。

4. 執行 `kubectl get crd`以驗證所有 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 無法設定複寫關係、但儲存管理員可以使用 "ONTAP SnapMirror"來複寫 SVM。

發生災難時，您可以啟動 SnapMirror 目標 SVM 開始提供資料服務。系統復原後，您可以切換回主 SVM。

關於此任務

使用 SnapMirror SVM 複製功能時，請考慮以下事項：

- 您應該為每個啟用 SVM-DR 的 SVM 建立一個獨立的後端。
- 配置儲存類別，僅在需要時選擇複寫後端，以避免將不需要複寫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與複寫相關的額外成本和複雜性，並在開始此程序之前仔細考慮其恢復計畫。

SVM 複製

您可以使用 "ONTAP：SnapMirror SVM 複製" 來建立 SVM 複寫關係。

SnapMirror 允許您設定選項來控制要複寫的內容。執行 使用 Trident 進行 SVM 恢復 時，您需要知道選擇了哪些選項。

- "-identity-preserve true" 複製整個 SVM 組態。
- "-discard-configs network" 不包括 LIF 和相關網路設定。
- "-identity-preserve false" 僅複製磁碟區和安全性組態。

使用 Trident 進行 SVM 恢復

Trident 不會自動偵測 SVM 故障。發生災難時、管理員可以手動啟動 Trident 容錯移轉至新的 SVM。

步驟

1. 取消已排程和正在進行的 SnapMirror 傳輸、中斷複寫關係、停止來源 SVM、然後啟動 SnapMirror 目的地 SVM。
2. 如果您在設定 SVM 複寫時指定了 `-identity-preserve false`或 `-discard-config network`，請更新 Trident 後端定義檔中的 `managementLIF`和 `dataLIF`。
3. 確認 `storagePrefix`已存在於 Trident 後端定義檔中。此參數不可更改。省略 `storagePrefix`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，以反映新的目的地 SVM 名稱：

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. 如果您指定了 `-identity-preserve false`或 `discard-config network`，則必須重新啟動所有應用程式 pod。

如果您指定了 `-identity-preserve true`，則當目標 SVM 啟動時，Trident 配置的所有磁碟區都會開始提供資料服務。

Volume 複寫與還原

Trident 無法設定 SnapMirror 複製關係，但儲存管理員可以使用 "ONTAP SnapMirror 複製和恢復" 來複製由 Trident 建立的磁碟區。

然後，您可以使用 "tridentctl volume import" 將復原的磁碟區匯入到 Trident 中。

不支援在 `ontap-nas-economy`、 `ontap-san-economy`或 `ontap-flexgroup-economy`驅動程式上匯入。

快照資料保護

您可以使用以下方法保護和還原資料：

- 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。

  "Volume 快照"

- ONTAP 快照可還原磁碟區的全部內容，或還原個別檔案或 LUN。

  "ONTAP Snapshot"

使用 Trident 自動化具狀態應用程式的容錯移轉

Trident 的強制分離功能可讓您自動將磁碟區從 Kubernetes 叢集中不健康的節點分離，從而防止資料損壞並確保應用程式的可用性。此功能在節點無回應或因維護而離線的情況下尤其有用。

強制分離的詳細資訊

強制分離僅適用於 `ontap-san`、 `ontap-san-economy`、 `ontap-nas` 和 `ontap-nas-economy`。啟用強制分離前，必須在 Kubernetes 叢集上啟用非優雅節點關閉（NGNS）。Kubernetes 1.28 及更高版本預設啟用 NGNS。更多信息，請參閱"Kubernetes：非正常節點關機"。

使用 `ontap-nas`或 `ontap-nas-economy`驅動程式時，需要在後端配置中將 `autoExportPolicy`參數設定為 `true`，以便 Trident 可以使用託管匯出原則限制對應用了污點的 Kubernetes 節點的存取。

由於 Trident 依賴 Kubernetes NGNS，因此在所有不可接受的工作負載重新調度之前，請勿從不健康節點上移除 `out-of-service`污點。貿然應用或移除污點可能會危及後端資料保護。

當 Kubernetes 叢集管理員將 `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute`污點套用到節點並 `enableForceDetach`設定為 `true`時、Trident 將確定節點狀態並：

1. 停止對掛載到該節點的磁碟區的後端 I/O 存取。
2. 將 Trident 節點物件標記為 `dirty`（不適合新發佈）。

> Trident 控制器將拒絕新的發布磁碟區請求，直到 Trident 節點 pod 重新驗證節點是否符合條件（標記為 `dirty`之後）為止。即使叢集節點運作正常且已準備就緒，任何使用已掛載 PVC 調度的工作負載也不會被接受，直到 `Trident` 驗證節點 `clean（可以安全地進行新發布）為止。

當節點健全狀況恢復且污點被移除後、Trident 將：

1. 識別並清理節點上過期的已發布路徑。
2. 如果節點處於 `cleanable`狀態（已移除服務中斷污點，節點處於 `Ready`狀態），並且所有過期的已發布路徑都已清理，`Trident` 將重新接納該節點為 `clean，並允許向該節點發布新的磁碟區。

有關自動容錯移轉的詳細資訊

您可以透過與"節點健全狀況檢查 (NHC) operator"整合來自動執行強制分離程序。當節點發生故障時、NHC 會透過在 Trident 的命名空間中建立定義故障節點的 TridentNodeRemediation CR 來觸發 Trident 節點補救（TNR）並自動執行強制分離。TNR 僅在節點發生故障時建立、並在節點恢復上線或節點被刪除後由 NHC 移除。

故障節點 **Pod** 移除程序

自動故障轉移會選擇要從故障節點移除的工作負載。建立 TNR 時、TNR 控制器會將該節點標記為髒節點、阻止任何新的磁碟區發佈、並開始移除支援強制分離的 Pod 及其磁碟區附件。

所有受強制分離支援的磁碟區 / PVC 均受自動故障轉移支援：

- NAS 和使用自動匯出原則的 NAS 經濟型磁碟區（尚未支援 SMB）。
- SAN 和 SAN-economy 磁碟區。

請參閱 [強制分離的詳細資訊]。

預設行為：

- 使用強制分離（force-detach）支援的磁碟區的 Pod 將從故障節點移除。Kubernetes 會將這些 Pod 重新調度到健康的節點上。
- 使用不支援強制分離的磁碟區（包括非 Trident 磁碟區）的 Pod 不會從故障節點中移除。
- 無狀態 Pod （非 PVC）不會從故障節點中移除，除非設定了 pod 註解 `trident.netapp.io/podRemediationPolicy: delete`。

覆寫 **pod** 移除行為：

可使用 Pod 註解自訂 Pod 移除行為：`trident.netapp.io/podRemediationPolicy[retain, delete]`。發生故障轉移時，系統會檢查並使用這些註解。將註解應用於 Kubernetes 部署 / 複本集 Pod 規格，以防止註解在故障轉移後消失：

- `retain` - 在自動故障轉移期間，Pod 不會從故障節點中移除。
- `delete` - 在自動故障轉移期間，Pod 將從故障節點中移除。

這些註解可以應用於任何 pod 。

- 只有在支援強制分離的磁碟區上，發生故障的節點才會阻塞 I/O 操作。
- 對於不支援強制分離的磁碟區，存在資料損毀和多重附加問題的風險。

TridentNodeRemediation CR

TridentNodeRemediation（TNR）CR 定義了一個故障節點。TNR 的名稱就是故障節點的名稱。

TNR 範例：

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

TNR 狀態：使用以下命令檢視 TNR 的狀態：
`kubectl get tnr <name> -n <trident-namespace>`

TNR 可能處於下列幾種狀態之一：

- 修復中：
  - 停止對強制分離掛載到該節點所支援磁碟區的後端 I/O 存取。
  - Trident 節點物件被標記為髒（不適合新發佈）。
  - 從節點移除 Pod 和磁碟區附件
- *NodeRecoveryPending*：
  - 控制器正在等待節點重新上線。
  - 一旦節點上線，publish-enforcement 將確保節點乾淨且已準備好發布新磁碟區。
- 如果節點從 K8s 中刪除，TNR 控制器將移除 TNR 並停止協調。
- 成功：
  - 所有修復和節點恢復步驟均已成功完成。節點已清理乾淨，可以發布新的磁碟區。
- *Failed*：
  - 無法恢復的錯誤。錯誤原因在 CR 的 status.message 欄位中設定。

啟用自動容錯移轉

先決條件 ：

- 請確保在啟用自動故障轉移之前已啟用強制分離。有關更多資訊，請參閱 [強制分離的詳細資訊]。
- 在 Kubernetes 叢集中安裝節點健康檢查（NHC）。
  - "安裝 operator-sdk".

- 如果叢集中尚未安裝 Operator Lifecycle Manager（OLM），請安裝它： `operator-sdk olm install`
- 安裝 Node Health check Operator： `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml`。

> 您也可以使用以下[Integrating Custom Node Health Check Solutions]章節中指定的其他方法來偵測節點故障。

如需詳細資訊，請參閱 "節點健康檢查 Operator"。

步驟

1. 在 Trident 命名空間中建立一個 NodeHealthCheck (NHC) CR，用於監控叢集中的工作節點。範例：

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: <CR name>
spec:
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
        operator: DoesNotExist
  remediationTemplate:
    apiVersion: trident.netapp.io/v1
    kind: TridentNodeRemediationTemplate
    namespace: <Trident installation namespace>
    name: trident-node-remediation-template
  minHealthy: 0 # Trigger force-detach upon one or more node failures
  unhealthyConditions:
    - type: Ready
      status: "False"
      duration: 0s
    - type: Ready
      status: Unknown
      duration: 0s
```

2. 在 `trident` 命名空間中應用節點健康檢查 CR。

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

上述 CR 設定用於監控 K8s 工作節點的節點狀況 Ready: false 和 Unknown。當節點進入 Ready: false 或 Ready: Unknown 狀態時，將觸發 Automated-Failover。

`unhealthyConditions` 在 CR 中使用 0 秒寬限期。這會導致自動容錯移轉在 K8s 設定節點條件 Ready: false 時立即觸發，該條件會在 K8s 失去節點的活動訊號後設定。K8s 預設在最後一次活動訊號後等待 40 秒，然後才設定 Ready: false。此寬限期可在 K8s 部署選項中自訂。

如需其他組態選項，請參閱 "Node-Healthcheck-Operator 說明文件"。

其他設定資訊

當 Trident 安裝時啟用了強制分離功能，Trident 命名空間中會自動建立兩個額外的資源，以方便與 NHC 整合：TridentNodeRemediationTemplate（TNRT）和 ClusterRole。

TridentNodeRemediationTemplate (TNRT)：

TNRT 可作為 NHC 控制器的範本，NHC 控制器可依需求使用 TNRT 產生 TNR 資源。

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
    spec: {}
```

ClusterRole：

當啟用 force-detach 時，安裝過程中也會新增一個叢集角色。這會賦予 NHC 在 Trident 命名空間中對 TNRs 的權限。

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac.ext-remediation/aggregate-to-ext-remediation: "true"
  name: tridentnoderemediation-access
rules:
- apiGroups:
  - trident.netapp.io
  resources:
  - tridentnoderemediationtemplates
  - tridentnoderemediations
  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
```

K8s 叢集升級與維護

為防止任何故障轉移，請在 K8s 維護或升級期間暫停自動故障轉移，因為預計節點會當機或重新啟動。您可以透過修補 NHC CR（如上所述）來暫停其 CR：

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

這將暫停自動故障轉移。若要重新啟用自動故障轉移、請在維護完成後從規格中移除 pauseRequests。

限制

- 僅對受強制分離支援的磁碟區，在發生故障的節點上才會阻止 I/O 操作。只有使用受強制分離支援的磁碟區 / PVC 的 Pod 才會被自動移除。
- 自動故障轉移和強制分離作業在 trident-controller pod 內部運作。如果託管 trident-controller 的節點發生故障，自動故障轉移將會延遲，直到 K8s 將 pod 遷移到健康的節點。

整合自訂節點健全狀況檢查解決方案

您可以使用其他節點故障偵測工具取代 Node Healthcheck Operator，以觸發自動故障轉移。為確保與自動故障轉移機制相容，您的自訂解決方案應：

- 當偵測到節點故障時，建立 TNR，使用故障節點的名稱作為 TNR CR 名稱。
- 當節點恢復且 TNR 處於 Succeeded 狀態時，刪除 TNR。

安全性

安全性

請按照此處列出的建議、確保您的 Trident 安裝安全可靠。

在其自己的命名空間中執行 **Trident**

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 分開，請務必將 Trident 安裝在其專屬的 Kubernetes 命名空間中 (`trident`。將 Trident 置於其專屬的命名空間可確保只有 Kubernetes 管理人員才能存取 Trident Pod 以及儲存在命名空間 CRD 物件中的成品（例如後端和 CHAP 密碼，如果適用）。您應該確保僅允許管理員存取 Trident 命名空間，從而存取 `tridentctl` 應用程式。

使用 **CHAP** 驗證與 **ONTAP SAN** 後端

Trident 支援基於 CHAP 的 ONTAP SAN 工作負載驗證（使用 `ontap-san`和 `ontap-san-economy`驅動程式）。NetApp 建議 Trident 在主機和儲存後端之間使用雙向 CHAP 進行身份驗證。

對於使用 SAN 儲存驅動程式的 ONTAP 後端， Trident 可以透過 `tridentctl`設定雙向 CHAP 並管理 CHAP 使用者名稱和金鑰。請參閱"準備使用 ONTAP SAN 驅動程式配置後端"以了解 Trident 在 ONTAP 後端上的 CHAP 設定方式。

在 **NetApp HCI** 和 **SolidFire** 後端使用 **CHAP** 驗證

NetApp 建議部署雙向 CHAP，以確保主機與 NetApp HCI 及 SolidFire 後端之間的驗證。Trident 會使用一個包含每個租戶兩組 CHAP 密碼的 secret 物件。安裝 Trident 時，它會管理 CHAP 機密並將其儲存在對應 PV 的 `tridentvolume` CR 物件中。當你建立 PV 時，Trident 會使用 CHAP 機密來啟動 iSCSI 連線，並透過 CHAP 與 NetApp HCI 及 SolidFire 系統進行通訊。

由 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，以保護磁碟被盜、退回或重新利用時的敏感資料。如需詳細資訊，請參閱 "設定 NetApp Volume Encryption 總覽"。

- 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。
  - 您可以設定 NVE 加密標誌以 `""` 建立啟用 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，除非在後端組態中將 NVE 加密旗標設為 `false`（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。

- 您可以在 Trident 後端設定中將 NVE 加密標誌設定為 `true`，以覆蓋 NAE 加密並按磁碟區使用特定的加密金鑰。
- 在啟用 NAE 的後端上將 NVE 加密標誌設為 `false`會建立一個啟用 NAE 的磁碟區。您無法將 NVE 加密標誌設為 `false`來停用 NAE 加密。

- 您可以透過明確地將 NVE 加密標誌設為 `true` 來在 Trident 中手動建立 NVE 磁碟區。

如需後端組態選項的詳細資訊，請參閱：

- "ONTAP SAN 組態選項"
- "ONTAP NAS 組態選項"

Linux Unified Key Setup（LUKS）

您可以啟用 Linux Unified Key Setup （LUKS） 來加密 Trident 上的 ONTAP SAN 和 ONTAP SAN ECONOMY 磁碟區。Trident 支援對 LUKS 加密磁碟區進行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，如 "NIST" 所建議。

ASA r2 系統不支援 LUKS 加密。有關 ASA r2 系統的資訊，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）。如需更多資訊，請造訪 "Gitlab：cryptsetup"。
- 出於效能考慮，NetApp 建議工作節點支援高級加密標準新指令集（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 的更多資訊，請造訪："Intel：Advanced Encryption Standard Instructions（AES-NI）"。

啟用 **LUKS** 加密

您可以使用 Linux Unified Key Setup (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資訊,請參閱 "ONTAP SAN 組態選項"。

```
{
  "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` 來定義使用 LUKS 加密的儲存池。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 密碼的密鑰。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。

匯入 **LUKS** 磁碟區的後端組態

若要匯入 LUKS 磁碟區、您必須在後端將 `luksEncryption`設為（`true`。 `luksEncryption`選項會告知 Trident 磁碟區是否符合 LUKS 規範（`true`或不符合 LUKS 規範（`false`、如下列範例所示。

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

用於匯入 **LUKS** 磁碟區的 **PVC** 組態

若要動態匯入 LUKS 卷，請將註解 `trident.netapp.io/luksEncryption`設為 `true`並在 PVC 中包含啟用 LUKS 的儲存類，如本範例所示。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

輪換 LUKS 密碼

您可以輪換 LUKS 密碼並確認輪換。

> 請務必在確認所有磁碟區、快照或金鑰均不再引用該密碼短語後再忘記它。如果引用的密碼短語遺失，您可能無法掛載該磁碟區，且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 密碼後建立掛載磁碟區的 Pod 時，就會發生 LUKS 密碼輪替。建立新 Pod 時，Trident 會將磁碟區上的 LUKS 密碼與金鑰中的活動密碼進行比較。

- 如果磁碟區上的密碼與密碼中的作用中密碼不符、就會發生輪替。
- 如果磁碟區上的密碼與密碼中的作用中密碼相符、則會忽略 `previous-luks-passphrase` 參數。

步驟

1. 新增 `node-publish-secret-name`和 `node-publish-secret-namespace` StorageClass 參數。例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. 識別磁碟區或快照上現有的複雜密碼。

磁碟區

```
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`與先前的密碼短語一致。

```
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: B
  luks-passphrase: secretB
  previous-luks-passphrase-name: A
  previous-luks-passphrase: secretA
```

4. 建立新的 pod 並掛載磁碟區。這是啟動輪替所必需的。
5. 確認密碼已輪換。

磁碟區

```
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複雜密碼時，複雜密碼就會輪替。

> 如果傳回兩個密碼短語，例如 `luksPassphraseNames: ["B", "A"]`，則輪換尚未完成。您可以觸發一個新的 pod 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` 功能閘控（beta 1.25+）。詳情請參閱 "Kubernetes 1.25：使用 Secrets 實作 CSI 磁碟區的節點驅動擴展"。
2. 新增 `node-expand-secret-name`和 `node-expand-secret-namespace` StorageClass 參數。例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

結果

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 傳輸中加密

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：

- 內部部署 **ONTAP** - Trident 支援透過 NFSv3 和 NFSv4 連線，從 Red Hat OpenShift 和上游 Kubernetes 叢集到內部部署 ONTAP 磁碟區的 Kerberos 加密。

您可以建立、刪除、調整大小、建立快照、複製、唯讀複製和匯入使用 NFS 加密的磁碟區。

使用內部部署 **ONTAP** 磁碟區設定傳輸中 **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。

> 對於使用本機 ONTAP 儲存後端的 NFS 流量，僅支援使用 `ontap-nas` 儲存驅動程式進行 Kerberos 加密。

開始之前

- 請確保您可以使用 `tridentctl` 公用程式。
- 請確保您擁有 ONTAP 儲存後端的管理員存取權限。
- 請確保您知道要從 ONTAP 儲存後端共用的磁碟區名稱。
- 請確保您已設定 ONTAP 儲存虛擬機器以支援 NFS 磁碟區的 Kerberos 加密。請參閱 "在資料 LIF 上啟用 Kerberos"以取得相關說明。
- 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 域配置部分（第 13 頁） "NetApp NFSv4 增強功能與最佳實務指南"。

新增或修改 **ONTAP** 匯出原則

您需要為現有的 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建立的新匯出策略需要支援以下存取協定和存取權限：

存取通訊協定

使用 NFS、NFSv3 和 NFSv4 存取協定設定匯出原則。

存取詳細資料

您可以根據磁碟區的需求、配置三種不同版本的 Kerberos 加密之一：

- **Kerberos 5** -（驗證和加密）
- **Kerberos 5i** -（驗證和加密，具有身分保護功能）
- **Kerberos 5p** -（驗證和加密，提供身分和隱私保護）

使用適當的存取權限設定 ONTAP 匯出原則規則。例如，如果叢集將使用 Kerberos 5i 和 Kerberos 5p 加密混合掛載 NFS 磁碟區，請使用下列存取設定：

類型	唯讀存取	讀取 / 寫入存取權	超級使用者存取
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 匯出原則和匯出原則規則的資訊，請參閱下列文件：

- "建立匯出原則"
- "將規則新增至匯出原則"

建立儲存後端

您可以建立包含 Kerberos 加密功能的 Trident 儲存後端組態。

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 `spec.nfsMountOptions` 參數指定三種不同版本的 Kerberos 加密之一：

- `spec.nfsMountOptions: sec=krb5` （驗證與加密）
- `spec.nfsMountOptions: sec=krb5i`（驗證與加密及身分保護）
- `spec.nfsMountOptions: sec=krb5p`（驗證與加密，具備身分與隱私保護）

只能指定一個 Kerberos 等級。如果在參數清單中指定多個 Kerberos 加密等級，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組態檔。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. 使用上一步建立的組態檔建立後端：

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

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

```
tridentctl logs
```

在您識別並修正組態檔的問題後、您可以再次執行 create 命令。

建立儲存類別

您可以建立儲存類別來配置具有 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 `mountOptions` 參數指定三種不同版本的 Kerberos 加密之一：

- `mountOptions: sec=krb5`（驗證與加密）
- `mountOptions: sec=krb5i`（驗證與加密及身分保護）
- `mountOptions: sec=krb5p`（驗證與加密，具備身分與隱私保護）

只能指定一個 Kerberos 加密等級。如果在參數清單中指定了多個 Kerberos 加密等級，則僅使用第一個選項。如果在儲存後端組態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 物件，請參考以下範例：

```
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
```

配置磁碟區

建立儲存後端和儲存類別後，即可配置磁碟區。有關說明，請參閱 "配置磁碟區"。

使用 **Azure NetApp Files** 磁碟區設定傳輸中 **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

- 請確保已在受管理的 Red Hat OpenShift 叢集上啟用 Trident。
- 請確保您可以使用 `tridentctl` 公用程式。
- 請確保您已為 Kerberos 加密準備好 Azure NetApp Files 儲存後端，請注意需求並遵循 "Azure NetApp Files 文件"中的說明。
- 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 域配置部分（第 13 頁） "NetApp NFSv4 增強功能與最佳實務指南"。

建立儲存後端

您可以建立包含 Kerberos 加密功能的 Azure NetApp Files 儲存後端組態。

關於此任務

建立配置 Kerberos 加密的儲存後端組態檔時，您可以將其定義為套用於下列兩個層級之一：

- 使用 `spec.kerberos`欄位的 **storage backend level**
- 使用 `spec.storage.kerberos`欄位的*虛擬資源池層級*

在虛擬資源池層級定義組態時，會使用儲存類別中的標籤來選取資源池。

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` （驗證與加密）
- `kerberos: sec=krb5i`（驗證與加密及身分保護）
- `kerberos: sec=krb5p`（驗證與加密，具備身分與隱私保護）

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資源池等級），使用下列其中一個範例建立儲存後端組態檔。將方括號 <> 中的值替換為您環境中的資訊：

儲存後端層級範例

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

虛擬資源池層級範例

```
---
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. 使用上一步建立的組態檔建立後端：

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

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

```
tridentctl logs
```

在您識別並修正組態檔的問題後、您可以再次執行 create 命令。

建立儲存類別

您可以建立儲存類別來配置具有 Kerberos 加密的磁碟區。

步驟

1. 建立一個 StorageClass Kubernetes 物件，請參考以下範例：

```
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
```

配置磁碟區

建立儲存後端和儲存類別後，即可配置磁碟區。有關說明，請參閱 "配置磁碟區"。

使用 Trident Protect 保護應用程式

了解 Trident Protect

NetApp Trident Protect 提供進階應用資料管理功能，可增強由 NetApp ONTAP 儲存系統和 NetApp Trident CSI 儲存設定器支援的有狀態 Kubernetes 應用的功能和可用性。Trident Protect 簡化了跨公有雲和本地環境的容器化工作負載的管理、保護和遷移。它還透過其 API 和 CLI 提供自動化功能。

您可以透過建立自訂資源（CR）或使用 Trident Protect CLI 來使用 Trident Protect 保護應用程式。

接下來呢？

您可以在安裝 Trident Protect 之前先了解其相關需求：

- "Trident Protect 要求"

安裝 Trident Protect

Trident Protect 要求

首先，請驗證您的運作環境、應用程式叢集、應用程式和授權是否已準備就緒。確保您的環境符合這些要求，以部署和執行 Trident Protect。

Trident Protect Kubernetes 叢集相容性

Trident Protect 與各種完全託管和自架的 Kubernetes 產品相容，包括：

- Amazon Elastic Kubernetes Service（EKS）
- Google Kubernetes Engine（GKE）
- Microsoft Azure Kubernetes Service（AKS）
- Red Hat OpenShift
- SUSE Rancher
- VMware Tanzu Portfolio
- 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"。
- 確保至少存在一個 VolumeSnapshotClass。如需更多資訊，請參閱 "VolumeSnapshotClass"。

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 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。
- 請確保您已設定用於儲存備份的物件儲存貯體。
- 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 Volume 的需求

Trident Protect 支援對 nas-economy 磁碟區進行備份和還原作業。目前不支援對 nas-economy 磁碟區進行快照、複製和 SnapMirror 複寫。您需要為計劃與 Trident Protect 搭配使用的每個 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。對於這些應用程式，您需要在 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 `<volume-UUID>`替換為您要變更的磁碟區的 UUID：

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

您可以將 Trident 後端組態選項 `snapshotDir`設為 `true`，為新磁碟區預設啟用快照目錄。現有磁碟區不受影響。

利用 KubeVirt VM 保護資料

Trident Protect 在資料保護作業期間為 KubeVirt 虛擬機器提供檔案系統凍結和解凍功能，以確保資料一致性。虛擬機器凍結操作的配置方法和預設行為因 Trident Protect 版本而異，較新版本透過 Helm Chart 參數提供了更簡化的配置方式。

在復原作業期間，不會復原為虛擬機器 (VM) 建立的任何 `VirtualMachineSnapshots`。

Trident Protect 25.10 及更新版本

Trident Protect 會在資料保護作業期間自動凍結和解凍 KubeVirt 檔案系統，以確保資料一致性。從 Trident Protect 25.10 開始，您可以在 Helm chart 安裝過程中使用 `vm.freeze`參數來停用此功能。此參數預設為啟用。

```
helm install ... --set vm.freeze=false ...
```

Trident Protect 24.10.1 至 **25.06**

從 Trident Protect 24.10.1 版本開始，Trident Protect 會在資料保護作業期間自動凍結和解凍 KubeVirt 檔案系統。您也可以選擇使用以下命令停用此自動行為：

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

Trident Protect 24.10

Trident Protect 24.10 在資料保護作業期間不會自動確保 KubeVirt VM 檔案系統的一致性狀態。如果您想使用 Trident Protect 24.10 保護 KubeVirt VM 資料，則需要在執行資料保護作業之前手動啟用檔案系統的凍結/解凍功能。這樣可以確保檔案系統處於一致狀態。

您可以設定 Trident Protect 24.10 來管理資料保護作業期間 VM 檔案系統的凍結和解凍，方法是"配置虛擬化"然後使用下列指令：

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

SnapMirror 複寫的需求

NetApp SnapMirror 複寫可用於 Trident Protect，支援以下 ONTAP 解決方案：

- 本地端 NetApp FAS、AFF 和 ASA 系統。目前 ASA r2 系統尚不支援使用 Trident protect 的 SnapMirror 複寫。
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

ONTAP 叢集 SnapMirror 複製要求

如果您打算使用 SnapMirror 複製功能，請確保您的 ONTAP 叢集符合下列要求：

- **NetApp Trident**：NetApp Trident 必須同時存在於使用 ONTAP 作為後端的來源和目標 Kubernetes 叢集上。Trident Protect 支援使用 NetApp SnapMirror 技術，透過以下驅動程式支援的儲存類別進行複寫：
  - `ontap-nas`：NFS
  - `ontap-san`：iSCSI
  - `ontap-san`：FC
  - `ontap-san`：NVMe/TCP （需要最低 ONTAP 版本 9.15.1）
- 授權：必須在來源和目的地 ONTAP 叢集上啟用使用資料保護套件的 ONTAP SnapMirror 非同步授權。如需詳細資訊，請參閱 "SnapMirror 授權總覽（ONTAP）"。

  從 ONTAP 9.10.1 開始，所有授權均以 NetApp 授權檔案（NLF）的形式提供，這是一個可啟用多項功能的單一檔案。如需詳細資訊，請參閱 "ONTAP One 隨附的授權"。

  僅支援 SnapMirror 非同步保護。

SnapMirror 複寫的對等考量

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 儲存後端必須建立對等連線。如需詳細資訊、請參閱 "叢集和 SVM 對等連接概述"。

  確保兩個 ONTAP 叢集之間複寫關係中使用的 SVM 名稱是唯一的。

- **NetApp Trident** 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。

Trident / ONTAP 用於 SnapMirror 複製的配置

Trident Protect 要求您至少設定一個支援來源叢集和目標叢集複寫的儲存後端。如果來源叢集和目標叢集相同，為了獲得最佳的恢復能力，目標應用程式應使用與來源應用程式不同的儲存後端。

Kubernetes 叢集 SnapMirror 複製要求

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 讀取和寫入資料以進行應用程式物件複製。
- 網路連線：設定防火牆規則、儲存桶權限和 IP 允許列表，以啟用兩個叢集之間以及 AppVault 跨 WAN 的通訊。

許多企業環境在 WAN 連線上實施嚴格的防火牆原則。在設定複寫之前，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，也可以從您自己的私人登錄安裝。如果您的叢集無法存取網際網路，從私人登錄安裝會很有幫助。

安裝 **Trident Protect**

從 **NetApp** 安裝 **Trident Protect**

步驟

1. 新增 Trident Helm 儲存庫：

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

2. 使用 Helm 安裝 Trident Protect。將 `<name-of-cluster>`替換為叢集名稱，該名稱將分配給叢集並用於識別叢集的備份和快照：

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect
```

3. （選用）若要啟用偵錯日誌記錄（建議用於疑難排解），請使用：

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect
```

偵錯日誌有助於 NetApp 支援人員在無需更改日誌等級或重現問題的情況下排查問題。

從私人註冊表安裝 **Trident Protect**

如果您的 Kubernetes 叢集無法存取網際網路，您可以從私人映像登錄安裝 Trident Protect。在這些範例中，請將方括號中的值替換為您環境中的資訊：

步驟

1. 將以下映像拉取到本機、更新標籤、然後將其推送到您的私有登錄：

```
docker.io/netapp/controller:25.10.0
docker.io/netapp/restic:25.10.0
docker.io/netapp/kopia:25.10.0
docker.io/netapp/kopiablockrestore:25.10.0
docker.io/netapp/trident-autosupport:25.10.0
docker.io/netapp/exechook:25.10.0
docker.io/netapp/resourcebackup:25.10.0
docker.io/netapp/resourcerestore:25.10.0
docker.io/netapp/resourcedelete:25.10.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

例如：

```
docker pull docker.io/netapp/controller:25.10.0
```

```
docker tag docker.io/netapp/controller:25.10.0 <private-registry-url>/controller:25.10.0
```

```
docker push <private-registry-url>/controller:25.10.0
```

若要取得 Helm chart，請先在可存取互聯網的機器上使用 `helm pull trident-protect --version 100.2510.0 --repo https://netapp.github.io/trident-protect-helm-chart`下載 Helm chart，然後將產生的 `trident-protect-100.2510.0.tgz`檔案複製到您的離線環境中，並在最後一步中使用 `helm install trident-protect ./trident-protect-100.2510.0.tgz`而不是儲存庫引用進行安裝。

2. 建立 Trident Protect 系統命名空間：

```
kubectl create ns trident-protect
```

3. 登入登錄：

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. 建立用於私有登錄驗證的 pull secret ：

```
kubectl create secret docker-registry regcred --docker-username=<registry-username> --docker-password=<api-token> -n trident-protect --docker-server=<private-registry-url>
```

5. 新增 Trident Helm 儲存庫：

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

6. 建立一個名為 `protectValues.yaml`的檔案。確保該檔案包含以下 Trident Protect 設定：

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

imageRegistry 和 imagePullSecrets 值適用於所有組件鏡像，包括 resourcebackup 和 resourcerestore。如果您將鏡像推送到登錄中的特定儲存庫路徑（例如， example.com:443/my-repo），請在登錄欄位中包含完整路徑。這將確保所有鏡像都從 <private-registry-url>/<image-name>:<tag> 拉取。

7. 使用 Helm 安裝 Trident Protect。將 `<name_of_cluster>`替換為叢集名稱，該名稱將分配給叢集並用於識別叢集的備份和快照：

```
helm install trident-protect netapp-trident-protect/trident-protect --set clusterName=<name_of_cluster> --version 100.2510.0 --create-namespace --namespace trident-protect -f protectValues.yaml
```

8. （選用）若要啟用偵錯日誌記錄（建議用於疑難排解），請使用：

```
helm install trident-protect netapp-trident-protect/trident-protect --set clusterName=<name-of-cluster> --set logLevel=debug --version 100.2510.0 --create-namespace --namespace trident-protect -f protectValues.yaml
```

偵錯日誌有助於 NetApp 支援人員在無需更改日誌等級或重現問題的情況下排查問題。

有關 Helm chart 配置的其他選項，包括 AutoSupport 設定和命名空間過濾，請參閱 "自訂 Trident Protect 安裝"。

安裝 **Trident Protect CLI** 外掛程式

您可以使用 Trident Protect 命令列外掛程式（ Trident `tridentctl` 實用程式的擴充功能）來建立和與 Trident Protect 自訂資源（ CR ）進行互動。

安裝 **Trident Protect CLI** 外掛程式

在使用命令列實用程式之前，需要將其安裝到用於存取叢集的電腦上。請根據您的電腦使用的是 x64 還是 ARM CPU，依照以下步驟操作。

下載適用於 **Linux AMD64 CPU** 的外掛程式

步驟

1. 下載 Trident Protect CLI 外掛程式：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-amd64
```

下載適用於 **Linux ARM64 CPU** 的外掛程式

步驟

1. 下載 Trident Protect CLI 外掛程式：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-arm64
```

下載適用於 **Mac AMD64 CPU** 的外掛程式

步驟

1. 下載 Trident Protect CLI 外掛程式：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-amd64
```

下載適用於 **Mac ARM64 CPU** 的外掛程式

步驟

1. 下載 Trident Protect CLI 外掛程式：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-arm64
```

1. 啟用外掛程式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 環境變數中定義的某個位置。例如， `/usr/bin` 或 `/usr/local/bin`（您可能需要管理員權限）：

```
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下的某個位置。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/
```

> 將外掛程式複製到 PATH 變數中的某個位置,即可透過鍵入 `tridentctl-protect` 或 `tridentctl protect` 從任何位置使用該外掛程式。

查看 **Trident CLI** 外掛程式說明

您可以使用內建的外掛程式說明功能，以取得外掛程式功能的詳細說明：

步驟

1. 使用說明功能檢視使用指南：

```
tridentctl-protect help
```

啟用命令自動完成

安裝 Trident Protect CLI 外掛程式後、您可以為某些命令啟用自動完成功能。

為 **Bash shell** 啟用自動完成功能

步驟

1. 建立完成指令碼：

```
tridentctl-protect completion bash > 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. 將下載的腳本移至 `~/.bash/completions` 目錄：

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. 將以下行新增至您主目錄中的 `~/.bashrc` 檔案：

```
source ~/.bash/completions/tridentctl-completion.bash
```

為 **Z shell** 啟用自動完成

步驟

1. 建立完成指令碼：

```
tridentctl-protect completion zsh > 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. 將下載的腳本移至 `~/.zsh/completions` 目錄：

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. 將以下行新增至您主目錄中的 `~/.zprofile` 檔案：

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

結果

下次登入 shell 時、您可以使用 tridentctl-protect 外掛程式進行命令自動完成。

自訂 Trident Protect 安裝

您可以自訂 Trident Protect 的預設組態，以符合您環境的特定需求。

指定 Trident Protect 容器資源限制

安裝 Trident Protect 後，您可以使用組態檔為 Trident Protect 容器指定資源限制。設定資源限制可讓您控制 Trident Protect 作業所消耗的叢集資源量。

步驟

1. 建立一個名為 `resourceLimits.yaml` 的檔案。
2. 根據您的環境需求，在檔案中填入 Trident Protect 容器的資源限制選項。

   以下範例組態檔顯示可用的設定，並包含每個資源限制的預設值：

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
```

```
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. 應用 `resourceLimits.yaml` 檔案中的值：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

自訂安全內容限制

安裝 Trident Protect 後，您可以使用組態檔修改 Trident Protect 容器的 OpenShift 安全性內容限制（SCC）。這些限制定義了 Red Hat OpenShift 叢集中 Pod 的安全性限制。

步驟

1. 建立一個名為 `sccconfig.yaml` 的檔案。
2. 在檔案中新增 SCC 選項，並根據您的環境需求修改參數。

   以下範例顯示 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：

參數	說明	預設
建立	確定是否可以建立 SCC 資源。僅當 `scc.create`設定為 `true`且 Helm 安裝程序識別 OpenShift 環境時，才會建立 SCC 資源。如果未在 OpenShift 中執行，或 `scc.create`設定為 `false`，則不會建立 SCC 資源。	true
姓名	指定 SCC 的名稱。	Trident 保護工作
優先順序	定義 SCC 的優先順序。優先順序值較高的 SCC 會在優先順序值較低的 SCC 之前進行評估。	1

3. 應用 `sccconfig.yaml` 檔案中的值：

```
helm upgrade trident-protect -n trident-protect netapp-trident-protect/trident-protect -f sccconfig.yaml --reuse-values
```

這將把預設值替換為 `sccconfig.yaml` 檔案中指定的值。

設定其他 **Trident Protect Helm Chart** 設定

您可以自訂 AutoSupport 設定和命名空間篩選，以滿足您的特定需求。下表說明可用的組態參數：

參數	類型	說明
autoSupport.proxy	字串	設定 NetApp AutoSupport 連線的代理 URL。使用此功能可將支援包上傳路由至代理伺服器。例如：`http://my.proxy.url`。
autoSupport.insecure	布林值	設定為 `true`時，會跳過 AutoSupport Proxy 連線的 TLS 驗證。僅用於不安全的 Proxy 連線。（預設值： `false`）
autoSupport.enabled	布林值	啟用或停用每日 Trident Protect AutoSupport 捆綁包上傳。設定為 `false`時，每日定時上傳將被停用，但您仍然可以手動產生支援捆綁包。（預設值： `true`）
restoreSkipNamespaceAnnotations	字串	以逗號分隔的命名空間註解清單，用於從備份和還原作業中排除。讓您根據註解篩選命名空間。

參數	類型	說明
restoreSkipNamespaceLabels	字串	以逗號分隔的命名空間標籤清單，用於從備份和還原作業中排除。允許您根據標籤篩選命名空間。

您可以使用 YAML 組態檔或命令列旗標來設定這些選項：

使用 **YAML** 檔案

步驟

1. 建立一個設定檔並將其命名為 `values.yaml`。
2. 在您建立的檔案中，新增您想要自訂的組態選項。

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
```

3. 在 `values.yaml` 檔案中填入正確的值後，套用設定檔：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

使用 **CLI** 標誌

步驟

1. 使用以下命令並加上 `--set` 標誌位元來指定各個參數：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --reuse-values
```

將 **Trident Protect Pod** 限制在特定節點上

您可以使用 Kubernetes nodeSelector 節點選擇約束，根據節點標籤來控制哪些節點可以執行 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約束。

步驟

1. 建立一個名為 `nodeSelectorConfig.yaml` 的檔案。
2. 將 nodeSelector 選項新增至檔案中，並修改該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 `nodeSelectorConfig.yaml` 檔案中的值：

```
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 `nodeSelectorConfig.yaml` 文件中指定的限制。

管理 Trident Protect

管理 Trident Protect 授權和存取控制

Trident Protect 使用 Kubernetes 的角色型存取控制（RBAC）模型。預設情況下，Trident Protect 提供單一系統命名空間及其相關的預設服務帳戶。如果您的組織有許多使用者或特定的安全需求，您可以使用 Trident Protect 的 RBAC 功能，更精細地控制對資源和命名空間的存取。

叢集管理員始終擁有對預設 `trident-protect` 命名空間中資源的存取權限，並且還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存取，您需要建立其他命名空間，並將資源和應用程式新增至這些命名空間。

請注意，任何使用者都無法在預設 `trident-protect`命名空間中建立應用程式資料管理 CR。您需要在應用程式命名空間中建立應用程式資料管理 CR（最佳實踐是將應用程式資料管理 CR 建立在與其關聯應用程式相同的命名空間中）。

只有管理員才能存取具有特權的 Trident Protect 自訂資源物件，其中包括：

- **AppVault**：需要儲存桶憑證資料
- **AutoSupportBundle**：收集指標、日誌和其他敏感的 Trident Protect 數據
- **AutoSupportBundleSchedule**：管理日誌收集排程

最佳實踐是使用 RBAC 將對特權物件的存取限制在管理員範圍內。

如需有關 RBAC 如何管理對資源和命名空間的存取的詳細資訊，請參閱 "Kubernetes RBAC 文件"。

如需服務帳戶的相關資訊，請參閱 "Kubernetes 服務帳戶文件"。

範例：管理兩組使用者的存取權限

例如，一個組織有一名 cluster 管理員、一組工程用戶和一組行銷用戶。cluster 管理員需要完成以下任務，以建立一個環境，使工程使用者群組和行銷使用者群組各自只能存取分配給其各自 namespace 的資源。

步驟 **1**：建立命名空間以包含每個群組的資源

建立命名空間可讓您以邏輯方式分隔資源，並更有效地控制誰可以存取這些資源。

步驟

1. 為工程群組建立命名空間：

```
kubectl create ns engineering-ns
```

2. 為行銷組建立命名空間：

```
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. 為工程群組建立服務帳戶：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. 為行銷群組建立服務帳戶：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

步驟 3：為每個新服務帳戶建立一個密碼

服務帳戶密碼用於對服務帳戶進行身分驗證，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密鑰：

```
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密鑰：

```
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：建立一個 **RoleBinding** 對象，並將該 **ClusterRole** 對象綁定到每個新的服務帳戶

當您安裝 Trident Protect 時，會自動建立一個預設的 ClusterRole 物件。您可以透過建立並套用 RoleBinding 物件，將此 ClusterRole 綁定至服務帳戶。

步驟

1. 將 ClusterRole 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 ClusterRole 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. 確認工程使用者無法存取行銷資源：

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

步驟 6：授予 AppVault 物件存取權限

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 和密碼組合 YAML 檔案，授予使用者存取 AppVault 的權限。例如，以下 CR 授予使用者存取 AppVault 的權限 `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 Role CR，以使叢集管理員能夠授予對命名空間中特定資源的存取權。例如：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. 建立並套用 RoleBinding CR，將權限綁定到使用者 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. 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 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以外的任何資源，或建立他們無權存取的新資源。

監控 Trident Protect 資源

您可以使用 kube-state-metrics、Prometheus 和 Alertmanager 這些開源工具來監控由 Trident Protect 保護的資源健康狀況。

kube-state-metrics 服務會從 Kubernetes API 通訊產生指標。將其與 Trident Protect 結合使用，可公開有關環境中資源狀態的實用資訊。

Prometheus 是一個工具包，可以接收由 kube-state-metrics 產生的數據，並將其呈現為易於閱讀的這些物件資訊。kube-state-metrics 和 Prometheus 結合使用，為你提供一種方式來監控你使用 Trident Protect 管理的資源的健康狀態和狀態。

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。

> 這些步驟中包含的組態和指南僅為範例；您需要根據自己的環境進行自訂。有關具體說明和支援，請參閱以下官方文件：
>
> - "kube-state-metrics 說明文件"
> - "Prometheus 說明文件"
> - "Alertmanager 文件"

步驟 1：安裝監控工具

要在 Trident Protect 中啟用資源監控，您需要安裝和設定 kube-state-metrics、Prometheus 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 應用於叢集：

```
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`）。您可以自訂以下範例設定以符合您的環境：

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 "kube-state-metrics 自訂資源文件"中的說明配置 kube-state-metrics，以產生 Trident Protect 使用的自訂資源的指標。

安裝 Prometheus

您可以按照 "Prometheus 說明文件"中的說明安裝 Prometheus。

安裝 Alertmanager

您可以按照 "Alertmanager 文件"中的說明安裝 Alertmanager。

步驟 2：設定監控工具以協同運作

安裝監控工具後，您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 的整合

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. 設定 Prometheus 將警示路由至 Alertmanager。編輯 Prometheus 組態檔 (`prometheus.yaml`，並新增以下區段：

prometheus.yaml：將警報傳送至 **Alertmanager**

```
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**：設定警示和警示目的地

配置好工具協同工作後，您需要配置哪些類型的資訊會觸發警示，以及警示應該發送到哪裡。

警示範例：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 `Error`且持續時間達到 5 秒或更長時，將觸發該警報。您可以根據自己的環境自訂此範例，並將此 YAML 程式碼片段新增至您的 `prometheus.yaml`組態檔：

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** 以將警示傳送至其他管道

您可以設定 Alertmanager，使其會向其他管道（例如電子郵件、PagerDuty、Microsoft Teams 或其他通知服務）發送通知，只需在 `alertmanager.yaml` 文件中指定相應的配置即可。

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，請將 `api_url` 鍵值替換為您環境中使用的 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** 支援套件

Trident Protect 讓管理員產生包含對 NetApp 支援團隊有用的資訊的軟體包，例如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）檔案將支援軟體包上傳至 NetApp 支援網站（NSS）。

使用 **CR** 建立支援套件

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. 設定下列屬性：
   - **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**：（必填）確定支援包是立即產生還是定時產生。定時產生會在 UTC 時間凌晨 12 點進行。可選值：
     - 已排程
     - 手動
   - **spec.uploadEnabled**：（選用）控制產生支援服務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：
     - true
     - false （預設）
   - **spec.dataWindowStart**：（選用）RFC 3339 格式的日期字串，用於指定支援套件中包含的資料視窗的起始日期和時間。如果未指定，則預設為 24 小時前。您可以指定的最早視窗日期為 7 天前。

     YAML 範例：

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. 在 `trident-protect-support-bundle.yaml` 檔案中填入正確的值後，套用 CR：

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-
protect
```

使用 **CLI** 建立支援套件

步驟

1. 建立支援包，將括號中的值替換為您環境中的資訊。 `trigger-type`決定是立即建立支援包還是根據排程建立，其值可以是 `Manual`或 `Scheduled`。預設設定為 `Manual`。

   例如：

```
tridentctl-protect create autosupportbundle <my-bundle-name>
--trigger-type <trigger-type> -n trident-protect
```

監控並擷取支援服務包

使用任一方法建立支援套件後，您可以監控其產生進度並將其擷取到本機系統。

步驟

1. 等待 `status.generationState` 達到 `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. 將支援套件組合擷取至本機系統。從已完成的 AutoSupport 套件組合取得複製命令：

```
kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
```

從輸出中找到 `kubectl cp`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 Protect**

您可以將 Trident Protect 升級至最新版本，以利用新功能或錯誤修正。

- 從 24.10 版本升級時，升級期間執行的快照可能會失敗。此失敗不會阻止未來的快照（無論是手動還是排程）被建立。如果快照在升級期間失敗，您可以手動建立新快照以確保應用程式受到保護。

  為避免潛在故障，您可以在升級前停用所有快照排程，並在升級後重新啟用。但是，這樣做會導致升級期間所有已排程的快照遺失。

- 對於私有鏡像倉庫安裝，請確保目標版本所需的 Helm chart 和鏡像已存在於您的私有鏡像倉庫中，並驗證您的自訂 Helm 值與新 chart 版本相容。如需更多資訊，請參閱"從私人註冊表安裝 Trident Protect"。

若要升級 Trident Protect、請執行下列步驟。

步驟

1. 更新 Trident Helm 儲存庫：

```
helm repo update
```

2. 升級 Trident Protect CRD：

> 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. 執行此命令將 CRD 的管理權限從 `trident-protect-crds`切換到 `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. 執行以下指令刪除 `trident-protect-crds` chart 的 Helm secret：

> 請勿使用 Helm 卸載 `trident-protect-crds`圖表，因為這可能會刪除您的 CRD 和任何相關資料。

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. 升級 Trident Protect：

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2510.0 --namespace trident-protect
```

> 您可以透過在升級命令中新增 `--set logLevel=debug`來配置升級期間的日誌等級。預設日誌等級為 `warn`。建議啟用偵錯日誌以進行故障排除，因為它有助於 NetApp 支援人員診斷問題，而無需更改日誌等級或重現問題。

管理和保護應用程式

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式 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）中所需的設定。只有管理員才能建立 AppVaults。

在對應用程式執行資料保護操作時，您需要手動或透過命令列建立 AppVault CR。AppVault CR 是針對您的環境而設定的，您可以參考此頁面上的範例來建立 AppVault CR。

> 請確保 AppVault CR 位於安裝了 Trident Protect 的叢集上。如果 AppVault CR 不存在或無法訪問，命令列將顯示錯誤。

設定 AppVault 驗證和密碼

在建立 AppVault CR 之前，請確保 AppVault 和您選擇的資料移動程式能夠透過提供者和任何相關資源進行身份驗證。

Data mover 儲存庫密碼

當您使用 CR 或 Trident Protect CLI 外掛程式建立 AppVault 物件時，您可以指定 Kubernetes Secret，其中包含用於 Restic 和 Kopia 加密的自訂密碼。如果您未指定 Secret，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`參數指定金鑰。

建立資料移動器儲存庫密碼密鑰

請使用以下範例建立密碼密鑰。建立 AppVault 物件時，您可以指示 Trident Protect 使用此密鑰向資料移動器儲存庫進行驗證。

- 根據您使用的資料遷移工具，您只需包含該資料遷移工具對應的密碼即可。例如，如果您使用的是 Restic，並且將來不打算使用 Kopia，則在建立 secret 時只需包含 Restic 的密碼即可。
- 請妥善保管密碼。您需要使用此密碼在同一叢集或其他叢集上還原資料。如果叢集或 `trident-protect`命名空間被刪除，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 相容儲存 IAM 權限

當您使用 Trident Protect 存取 S3 相容儲存（例如 Amazon S3、Generic S3 "StorageGrid S3"或 "ONTAP S3"）時，您需要確保所提供的使用者認證具有存取儲存貯體的必要權限。以下是授予使用 Trident Protect 存取所需最低權限的原則範例。您可以將此原則套用至管理 S3 相容儲存貯體原則的使用者。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

有關 Amazon S3 策略的更多資訊，請參閱 "Amazon S3 文件" 中的範例。

用於 Amazon S3（AWS）驗證的 EKS Pod Identity

Trident Protect 支援 EKS Pod Identity 進行 Kopia 資料遷移操作。此功能可安全存取 S3 儲存貯體，而無需在 Kubernetes Secret 中儲存 AWS 認證。

使用 Trident Protect 的 EKS Pod Identity 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 S3 儲存桶權限的 IAM 角色。如需深入瞭解，請參閱 "S3 相容儲存 IAM 權限"。
- IAM 角色與下列 Trident Protect 服務帳戶相關聯：
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

有關啟用 Pod Identity 並將 IAM 角色與服務帳戶關聯的詳細說明，請參閱 "AWS EKS Pod Identity 文件"。

AppVault 設定 使用 EKS Pod Identity 時，請使用 `useIAM: true`標誌設定 AppVault CR，而不是明確憑證：

```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
```

AppVault 雲端服務提供者的金鑰產生範例

定義 AppVault CR 時，除非使用 IAM 驗證，否則您需要包含存取提供者託管資源的憑證。憑證金鑰的產生方式因提供者而異。以下是幾個提供者的命令列金鑰產生範例。您可以使用以下範例為每個雲端提供者的憑證建立金鑰。

Google Cloud

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

Amazon S3（AWS）

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

Microsoft Azure

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

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

以下是各提供者的 AppVault 定義範例。

AppVault CR 範例

您可以使用以下 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。如需更多資訊，請參閱 Data mover 儲存庫密碼。
- 對於 Amazon S3 (AWS) AppVault 物件，您可以選擇性地指定 sessionToken，如果您使用單一登入（SSO）進行身份驗證，這將非常有用。此令牌在您為提供者產生金鑰時建立 AppVault 雲端服務提供者的金鑰產生範例。
- 對於 S3 AppVault 對象，您可以使用 `spec.providerConfig.S3.proxyURL` 鍵選擇性地指定出站 S3 流量的出口代理 URL。

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3（AWS）

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret
```

對於使用 Pod Identity 與 Kopia data mover 的 EKS 環境，您可以移除 `providerCredentials`區段，並將 `useIAM: true`新增到 `s3`設定下方。

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

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。如需更多資訊，請參閱 Data mover 儲存庫密碼。
- 對於 S3 AppVault 對象，您可以使用 `--proxy-url <ip_address:port>`參數選擇性地指定出站 S3 流量的出口代理 URL。

Google Cloud

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Amazon S3（AWS）

```
tridentctl-protect create vault AWS <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Microsoft Azure

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

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
```

支援的 `providerConfig.s3` 配置選項

請參閱下表以了解 S3 提供者組態選項：

參數	說明	預設	範例
`providerConfig.s3.skipCertValidation`	停用 SSL/TLS 憑證驗證。	錯誤	"true"、"false"
`providerConfig.s3.secure`	啟用與 S3 端點的安全 HTTPS 通訊。	true	"true"、"false"
`providerConfig.s3.proxyURL`	指定用於連接到 S3 的 Proxy 伺服器 URL。	無	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	提供用於 SSL/TLS 驗證的自訂根 CA 憑證。	無	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	啟用 IAM 驗證以存取 S3 儲存桶。適用於 EKS Pod Identity。	錯誤	true、false

查看 **AppVault** 資訊

您可以使用 Trident Protect CLI 外掛程式查看有關您在叢集上建立的 AppVault 物件的資訊。

步驟

1. 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，請使用標誌 `--show-paths`。

   只有在 Trident Protect Helm 安裝過程中指定了叢集名稱時，表格第一列中的叢集名稱才可用。例如：`--set clusterName=production1`。

移除 **AppVault**

您可以隨時移除 AppVault 物件。

> 在刪除 AppVault 物件之前，請勿移除 AppVault CR 中的 `finalizers`金鑰。否則，可能會導致 AppVault 儲存桶中殘留資料，並在叢集中產生孤立資源。

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. 刪除 AppVault 物件，將 `appvault-name`替換為要刪除的 AppVault 物件名稱：

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

使用 **Trident Protect CLI** 刪除 **AppVault**

1. 刪除 AppVault 物件，將 `appvault-name`替換為要刪除的 AppVault 物件名稱：

```
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作業時使用，且該 AppVault CR 必須位於安裝了 Trident Protect 的叢集上。AppVault CR 特定於您的環境；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. 建立目的地應用程式 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. 設定下列屬性：
      - **metadata.name**：（必填）應用程式自訂資源的名稱。請記住您選擇的名稱，因為保護作業所需的其他 CR 檔案會參照此值。
      - **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中的資源。
      - **spec.includedClusterScopedResources**：（選用）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、類型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器定義的應用程式，例如在 KubeVirt 環境中，快照之前會發生檔案系統凍結。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，則應用程式會忽略全域設定，並可在快照期間寫入檔案系統。如果設定為 false，則應用程式會忽略全域設定，且檔案系統會在快照期間凍結。如果已指定但應用程式定義中沒有虛擬機器，則會忽略該註解。如果未指定，則應用程式會遵循"全球 Trident Protect 凍結設定"。

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. （選用）新增篩選條件，包含或排除標記有特定標籤的資源：

   - **resourceFilter.resourceSelectionCriteria**：（篩選必需）使用 `Include`或 `Exclude`來包含或排除在 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 物件的陣列。如果在此陣列中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的欄位（group、kind、version）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 metadata.name 欄位中的名稱。

- **resourceMatchers[].namespaces**：（可選）要篩選的資源的 Kubernetes metadata.name 欄位中的命名空間。
- **resourceMatchers[].labelSelectors**：（可選）資源在 Kubernetes metadata.name 欄位中定義的標籤選擇器字串 "Kubernetes 說明文件"。例如：`"trident.netapp.io/os=linux"`。

> 當兩者 `resourceFilter` 和 `labelSelector` 同時使用時，`resourceFilter` 首先運行，然後 `labelSelector` 將應用於生成的資源。

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 CR 後、套用該 CR。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，並將括號中的值替換為您環境中的資訊。您可以使用範例中所示的參數，以逗號分隔的清單在應用程式定義中包含命名空間和資源。

   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器定義的應用程式、例如在 KubeVirt 環境中、快照之前會發生檔案系統凍結。如果將註解設為 `true`、則應用程式會忽略全域設定、並可在快照期間寫入檔案系統。如果將其設為 `false`、則應用程式會忽略全域設定、且檔案系統會在快照期間凍結。如果您使用註解、但應用程式定義中沒有虛擬機器、則會忽略該註解。如果您不使用註解、則應用程式會遵循"全球 Trident Protect 凍結設定"。

   使用 CLI 建立應用程式時，若要指定註解，可以使用 `--annotation` 標誌。

- 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>
```

- 建立應用程式並設定本機應用程式設定以控制檔案系統凍結行為：

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace> --annotation protect.trident.netapp.io/skip-vm-freeze
=<"true"|"false">
```

您可以使用 `--resource-filter-include` 和 `--resource-filter-exclude` 標誌來根據 `resourceSelectionCriteria` 群組、種類、版本、標籤、名稱和命名空間等條件包含或排除資源，如下例所示：

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-namespace>
--resource-filter-include
'[{"Group":"apps","Kind":"Deployment","Version":"v1","Names":["my-
deployment"],"Namespaces":["my-
namespace"],"LabelSelectors":["app=my-app"]}]'
```

使用 **Trident Protect** 保護應用程式

您可以使用自動保護原則或臨機方式拍攝快照和備份，以保護 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及解除凍結檔案系統。"深入瞭解如何使用 Trident Protect 設定檔案系統凍結"。

建立隨需快照

您可以隨時建立隨需快照。

如果叢集範圍的資源在應用程式定義中被明確引用，或引用了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. 在您建立的檔案中、設定以下屬性：
   - **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**：（必填）應儲存快照內容（元資料）的 AppVault 名稱。
   - **spec.reclaimPolicy**：（選用）定義當快照 CR 刪除時，快照的 AppArchive 會發生什麼情況。這意味著即使設為 `Retain`，快照也會被刪除。有效選項：
     - `Retain` (預設)
     - `Delete`

```
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  reclaimPolicy: Delete
```

3. 在 `trident-protect-snapshot-cr.yaml` 檔案中填入正確的值後，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，並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>
```

建立隨需備份

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，或引用了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

請確保 AWS 會話令牌的有效期限足以應付任何長時間運行的 s3 備份作業。如果令牌在備份作業期間過期，則操作可能會失敗。

- 有關檢查當前會話令牌過期時間的更多資訊，請參閱 "AWS API 文件"。
- 如需 AWS 資源憑證的詳細資訊，請參閱 "AWS IAM 文件"。

使用 **CR** 建立備份

步驟

1. 建立自訂資源（CR）檔案並將其命名為 `trident-protect-backup-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：
   - **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**：（必需）要備份的應用程式的 Kubernetes 名稱。
   - **spec.appVaultRef**：（必填）應儲存備份內容的 AppVault 名稱。
   - **spec.dataMover**：（選用）一個字串，指示要用於備份作業的備份工具。可能的值（區分大小寫）：
     - `Restic`
     - `Kopia` (預設)
   - **spec.reclaimPolicy**：（選用）定義備份從其聲明中釋放後會發生什麼。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，則會建立並備份一個臨時快照。

     YAML 範例：

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. 在 `trident-protect-backup-cr.yaml` 檔案中填入正確的值後，套用 CR：

```
kubectl apply -f trident-protect-backup-cr.yaml
```

使用 **CLI** 建立備份

步驟

1. 建立備份，並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>
```

您可以選擇使用 `--full-backup` 標誌來指定備份是否為非增量備份。預設情況下，所有備份均為增量備份。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，並在兩次完整備份之間執行增量備份，以最大程度地降低還原風險。

支援的備份註釋

下表說明建立備份 CR 時可使用的註解：

註解	類型	說明	預設值
protect.trident.netapp.io/full-backup	字串	指定備份是否為非增量備份。設定為 `true`可建立非增量備份。最佳實踐是定期執行完整備份，並在兩次完整備份之間執行增量備份，以最大程度地降低還原相關風險。	"false"
protect.trident.netapp.io/snapshot-completion-timeout	字串	允許完成整個快照作業的最長時間。	"60 分鐘"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	字串	磁碟區快照達到可用狀態所允許的最長時間。	"30 分鐘"
protect.trident.netapp.io/volume-snapshots-created-timeout	字串	建立磁碟區快照所允許的最長時間。	"5 分鐘"
protect.trident.netapp.io/pvc-bind-timeout-sec	字串	等待所有新建立的 PersistentVolumeClaims（PVC）達到 `Bound`階段的最長時間（以秒為單位），超過此時間操作將會失敗。	"1200"（20 分鐘）

建立資料保護排程

保護原則透過按定義的排程建立快照、備份或兩者來保護應用程式。您可以選擇每小時、每天、每週和每月建立快照和備份，並可指定要保留的副本數量。您可以使用 full-backup-rule 註解來排程非增量完整備份。預設情況下，所有備份都是增量備份。定期執行完整備份以及其間的增量備份，有助於降低與還原相關的風險。

- 您可以透過將 `backupRetention`設為零並將 `snapshotRetention`設為大於零的值來建立僅用於快照的排程。將 `snapshotRetention`設為零表示任何排程的備份仍會建立快照，但這些快照是暫時的，並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，或引用了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** 建立排程

步驟

1. 建立自訂資源（CR）檔案並將其命名為 `trident-protect-schedule-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：
   - **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**：（選用）一個字串，指示要用於備份作業的備份工具。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**：（必填）應儲存備份內容的 AppVault 名稱。
   - **spec.backupRetention**：（必填）要保留的備份數量。零表示不應建立任何備份（僅快照）。
   - **spec.backupReclaimPolicy**：（選用）決定如果在保留期間內刪除備份 CR 時，備份會發生什麼情況。保留期間過後，備份一律會被刪除。可能的值（區分大小寫）：
     - `Retain` (預設)
     - `Delete`
   - **spec.snapshotRetention**：（必填）要保留的快照數量。零表示不建立任何快照。
   - **spec.snapshotReclaimPolicy**：（選用）決定在保留期間內刪除快照 CR 時，快照會發生什麼情況。保留期間過後，快照一律會被刪除。可能的值（區分大小寫）：
     - `Retain`
     - `Delete` (預設)
   - **spec.granularity**：排程應執行的頻率。可能的值以及相關的必填欄位：
     - `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 、 spec.dayOfWeek`）
     - `Monthly` （需要您指定 `spec.minute, spec.hour 、 spec.dayOfMonth`）
     - `Custom`
   - **spec.dayOfMonth**：（選用）排程應執行的月份日期（1 - 31）。如果精細度設定為 `Monthly`，則此欄位為必填。此值必須以字串形式提供。
   - **spec.dayOfWeek**：（選用）排程應執行的星期幾（0 - 7）。值為 0 或 7 表示星期日。如果精細度設為 `Weekly`，則此欄位為必填項。該值必須以字串形式提供。
   - **spec.hour**：（選用）排程應執行的一天中的小時（0 - 23）。如果粒度設定為 `Daily、 Weekly`或 `Monthly`，則此欄位為必填項。值必須以字串形式提供。
   - **spec.minute**：（選用）排程應執行的小時分鐘數（0 - 59）。如果粒度設定為 `Hourly、 Daily、 Weekly`或 `Monthly`，則此欄位為必填項。該值必須以字串形式提供。

備份和快照排程的範例 YAML：

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"
```

僅快照排程的範例 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. 在 `trident-protect-schedule-cr.yaml` 檔案中填入正確的值後，套用 CR：

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

使用 **CLI** 建立排程

步驟

1. 建立保護排程，並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`來查看此命令的詳細說明資訊。

```
tridentctl-protect create schedule <my_schedule_name> \
  --appvault <my_appvault_name> \
  --app <name_of_app_to_snapshot> \
  --backup-retention <how_many_backups_to_retain> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_to_run_schedule> \
  --granularity <frequency_to_run> \
  --hour <hour_of_day_to_run> \
  --minute <minute_of_hour_to_run> \
  --recurrence-rule <recurrence> \
  --snapshot-retention <how_many_snapshots_to_retain> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

以下旗標可讓您對排程進行更多控制：

- 完整備份排程：使用 `--full-backup-rule` 標誌來排程非增量完整備份。此標誌僅適用於 `--granularity Daily`。可能的值：
  - `Always`：每天都要建立完整備份。
  - 特定工作日：指定一個或多個以逗號分隔的日期（例如， `"Monday,Thursday"`）。有效值：Monday、Tuesday、Wednesday、Thursday、Friday、Saturday、Sunday。

    `--full-backup-rule` 標誌不適用於每小時、每週或每月粒度。
- 僅快照排程：設定 `--backup-retention 0` 並為 `--snapshot-retention` 指定大於零的值。

支援的排程註釋

下表描述了建立排程 CR 時可以使用的註釋：

註解	類型	說明	預設值
protect.trident.netapp.io/full-backup-rule	字串	指定完整備份的調度規則。您可以將其設定為 `Always`持續完整備份，或根據您的需求進行自訂。例如，如果您選擇每日粒度，則可以指定執行完整備份的星期幾（例如，`"Monday,Thursday"）。有效的星期幾值為：星期一、星期二、星期三、星期四、星期五、星期六、星期日。請注意，此註解只能與已將 `granularity`設定為 `Daily`的計劃一起使用。	未設定 （所有備份均為增量備份）
protect.trident.netapp.io/snapshot-completion-timeout	字串	允許完成整個快照作業的最長時間。	"60 分鐘"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	字串	磁碟區快照達到可用狀態所允許的最長時間。	"30 分鐘"
protect.trident.netapp.io/volume-snapshots-created-timeout	字串	建立磁碟區快照所允許的最長時間。	"5 分鐘"
protect.trident.netapp.io/pvc-bind-timeout-sec	字串	等待所有新建立的 PersistentVolumeClaims （PVC）達到 `Bound`階段的最長時間（以秒為單位），超過此時間操作將會失敗。	"1200"（20 分鐘）

刪除快照

刪除不再需要的已排程或隨需快照。

步驟

1. 刪除與快照相關聯的快照 CR：

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

刪除備份

刪除不再需要的排程備份或隨需備份。

請確保將回收策略設定為 `Delete`，以從物件儲存中刪除所有備份資料。此策略的預設設定為 `Retain`，以避免意外資料遺失。如果未將策略變更為 `Delete`，則備份資料將保留在物件儲存中，需要手動刪除。

步驟

1. 刪除與備份相關聯的備份 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狀態。

步驟

1. 使用以下命令檢索備份作業的狀態，將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-files**（**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 磁碟區，並且不會佔用容量池中的額外空間。

開始之前

請確保以下事項：

- 您已安裝 Trident Protect 。
- 您已在 Trident Protect 中定義了一個應用程式。在您完成此程序之前，該應用程式的保護功能將受到限制。
- 您已選擇 `azure-netapp-files`作為儲存後端的預設儲存類別。

展開以檢視設定步驟

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 且與應用程式相關聯的 PV 啟用 Snapshot 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 Snapshot 目錄：

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   回應：

   ```
   snapshotDirectory: "true"
   ```

   +
   如果未啟用快照目錄，Trident Protect 將選擇常規備份功能，該功能會在備份過程中暫時佔用容量池中的空間。在這種情況下，請確保容量池中有足夠的空間來建立一個與待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。

還原應用程式

使用 **Trident Protect** 恢復應用程式

您可以使用 Trident Protect 從快照或備份還原應用程式。如果要將應用程式還原到同一個叢集，從現有快照還原速度會更快。

- 還原應用程式時，所有為該應用程式配置的執行鉤子都會隨應用程式一起還原。如果存在還原後執行鉤子，它會在還原操作過程中自動執行。
- qtree 磁碟區支援從備份還原到不同的命名空間或原始命名空間。但是，qtree 磁碟區不支援從快照還原到不同的命名空間或原始命名空間。
- 您可以使用進階設定來自訂還原操作。若要深入瞭解、請參閱 "使用進階 Trident Protect 還原設定"。

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

式，並為還原後的應用程式建立一個應用程式 CR。若要保護還原後的應用程式，您可以建立按需備份或快照，或設定保護計畫。

- 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。
- 使用 CR 還原到新命名空間時，必須先手動建立目標命名空間，然後再套用 CR。Trident Protect 僅在使用 CLI 時才會自動建立命名空間。

開始之前

請確保 AWS 工作階段權杖的有效期限足以應付任何長時間執行的 s3 還原作業。如果權杖在還原作業期間過期、作業可能會失敗。

- 有關檢查當前會話令牌過期時間的更多資訊，請參閱 "AWS API 文件"。
- 如需 AWS 資源憑證的詳細資訊，請參閱 "AWS IAM 文件"。

當您使用 Kopia 作為資料移動工具還原備份時，您可以選擇在 CR 中或使用 CLI 指定註釋，以控制 Kopia 使用的暫存的行為。有關可配置選項的更多資訊，請參閱 "Kopia 說明文件"。使用 `tridentctl-protect create --help`命令以取得有關使用 Trident Protect CLI 指定註釋的更多資訊。

使用 **CR**

步驟

1. 建立自訂資源（CR）檔案並將其命名為 `trident-protect-backup-restore-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：
   - **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來尋找此路徑：

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**：（必填）儲存備份內容的 AppVault 名稱。
   - **spec.namespaceMapping**：復原作業的來源命名空間到目標命名空間的對應。請將 `my-source-namespace`和 `my-destination-namespace`替換為您環境中的資訊。

     ```
     ---
     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還原，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，如果您選擇了持久卷聲明資源，並且它關聯了一個 pod，Trident Protect 也會恢復該關聯 pod。

   - **resourceFilter.resourceSelectionCriteria**：（篩選必需）使用 `Include`或 `Exclude`來包含或排除在 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 物件的陣列。如果在此陣列中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的欄位（group、kind、version）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 metadata.name 欄位中的名稱。
- **resourceMatchers[].namespaces**：（可選）要篩選的資源的 Kubernetes metadata.name 欄位中的命名空間。
- **resourceMatchers[].labelSelectors**：（可選）資源在 Kubernetes metadata.name 欄位中定義的標籤選擇器字串 "Kubernetes 說明文件"。例如：`"trident.netapp.io/os=linux"`。

  例如：

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 在 `trident-protect-backup-restore-cr.yaml` 檔案中填入正確的值後，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，並將方括號中的值替換為您環境中的資訊。 `namespace-mapping`引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 <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至原始命名空間。

開始之前

請確保 AWS 工作階段權杖的有效期限足以應付任何長時間執行的 s3 還原作業。如果權杖在還原作業期間過期、作業可能會失敗。

- 有關檢查當前會話令牌過期時間的更多資訊，請參閱 "AWS API 文件"。
- 如需 AWS 資源憑證的詳細資訊，請參閱 "AWS IAM 文件"。

當您使用 Kopia 作為資料移動工具還原備份時，您可以選擇在 CR 中或使用 CLI 指定註釋，以控制 Kopia 使用的暫存的行為。有關可配置選項的更多資訊，請參閱 "Kopia 說明文件"。使用 `tridentctl-protect create --help`命令以取得有關使用 Trident Protect CLI 指定註釋的更多資訊。

使用 **CR**

步驟

1. 建立自訂資源（CR）檔案並將其命名為 `trident-protect-backup-ipr-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：
   - **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來尋找此路徑：

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**：（必填）儲存備份內容的 AppVault 名稱。

     例如：

```
---
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
```

3. （可選）如果您只需要選擇應用程式中的某些資源進行還原，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，如果您選擇了持久卷聲明資源，並且它關聯了一個 pod，Trident Protect 也會恢復該關聯 pod。

   - **resourceFilter.resourceSelectionCriteria**：（篩選必需）使用 `Include`或 `Exclude`來包含或排除在 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 物件的陣列。如果在此陣列中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的欄位（group、kind、version）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 metadata.name 欄位

中的名稱。

- **resourceMatchers[].namespaces**：（可選）要篩選的資源的 Kubernetes metadata.name 欄位中的命名空間。
- **resourceMatchers[].labelSelectors**：（可選）資源在 Kubernetes metadata.name 欄位中定義的標籤選擇器字串 "Kubernetes 說明文件"。例如：`"trident.netapp.io/os=linux"`。

  例如：

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 在 `trident-protect-backup-ipr-cr.yaml` 檔案中填入正確的值後，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，並將方括號中的值替換為您環境中的資訊。 `backup` 參數使用命名空間和備份名稱，格式為 `<namespace>/<name>`。例如：

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

如果原始叢集出現問題、您可以將備份還原至不同的叢集。

- 當您使用 Kopia 作為資料移動工具還原備份時，您可以選擇在 CR 中或使用 CLI 指定註釋，以控制 Kopia 使用的暫存的行為。有關可配置選項的更多資訊，請參閱 "Kopia 說明文件"。使用 `tridentctl-protect create --help`命令以取得有關使用 Trident Protect CLI 指定註釋的更多資訊。
- 使用 CR 還原到新命名空間時，必須先手動建立目標命名空間，然後再套用 CR。Trident Protect 僅在使用 CLI 時才會自動建立命名空間。

開始之前

請確保符合下列先決條件：

- 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 儲存桶路徑，備份檔案就儲存在該路徑中。
- 執行 `tridentctl-protect get appvaultcontent`命令時，請確保本機環境可以連接到 AppVault CR 中定義的物件儲存桶。如果網路限制導致無法存取，請改為在目標叢集的 Pod 內執行 Trident Protect CLI。
- 請確保 AWS 工作階段權杖的有效期限足以應付任何長時間執行的還原作業。如果權杖在還原作業期間過期、作業可能會失敗。
  - 有關檢查當前會話令牌過期時間的更多資訊，請參閱 "AWS API 文件"。
  - 如需 AWS 資源憑證的詳細資訊，請參閱 "AWS 文件"。

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的地叢集上。

2. 從目的地叢集檢視可用 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 中的可用備份，包括其來源叢集、相應的應用程式名稱、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. 在您建立的檔案中、設定以下屬性：
   - **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**：（必填）儲存備份內容的 AppVault 名稱。
   - **spec.appArchivePath**：AppVault 內儲存備份內容的路徑。您可以使用以下命令來尋找此路徑：

     ```
     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**：復原作業的來源命名空間到目標命名空間的對應。請將 `my-source-namespace`和 `my-destination-namespace`替換為您環境中的資訊。

     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. 在 `trident-protect-backup-restore-cr.yaml` 檔案中填入正確的值後，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式，並將方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）檔案從快照還原資料，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式，並為還原的應用程式建立一個應用程式 CR。若要保護還原的應用程式，您可以建立隨需備份或快照，或建立保護排程。

- SnapshotRestore 支援 `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支援。如果嘗試還原到使用不同儲存後端的 `StorageClass`，則還原操作將會失敗。
- 使用 CR 還原到新命名空間時，必須先手動建立目標命名空間，然後再套用 CR。Trident Protect 僅在使用 CLI 時才會自動建立命名空間。

開始之前

請確保 AWS 工作階段權杖的有效期限足以應付任何長時間執行的 s3 還原作業。如果權杖在還原作業期間過期、作業可能會失敗。

- 有關檢查當前會話令牌過期時間的更多資訊，請參閱 "AWS API 文件"。
- 如需 AWS 資源憑證的詳細資訊，請參閱 "AWS IAM 文件"。

使用 **CR**

步驟

1. 建立自訂資源（CR）檔案並將其命名為 `trident-protect-snapshot-restore-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：
   - **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**：（必填）儲存快照內容的 AppVault 名稱。
   - **spec.appArchivePath**：AppVault 內儲存快照內容的路徑。您可以使用以下命令來尋找此路徑：

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.namespaceMapping**：復原作業的來源命名空間到目標命名空間的對應。請將 `my-source-namespace`和 `my-destination-namespace`替換為您環境中的資訊。

     ```
     ---
     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還原，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，如果您選擇了持久卷聲明資源，並且它關聯了一個 pod，Trident Protect 也會恢復該關聯 pod。

   - **resourceFilter.resourceSelectionCriteria**：（篩選必需）使用 `Include`或 `Exclude`來包含或排除在 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 物件的陣列。如果在此陣列中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的欄位（group、kind、version）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 metadata.name 欄位中的名稱。
- **resourceMatchers[].namespaces**：（可選）要篩選的資源的 Kubernetes metadata.name 欄位中的命名空間。
- **resourceMatchers[].labelSelectors**：（可選）資源在 Kubernetes metadata.name 欄位中定義的標籤選擇器字串 "Kubernetes 說明文件"。例如：`"trident.netapp.io/os=linux"`。

  例如：

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 在 `trident-protect-snapshot-restore-cr.yaml` 檔案中填入正確的值後，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。

   - `snapshot`引數使用命名空間和快照名稱，格式為 `<namespace>/<name>`。
   - `namespace-mapping` 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 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至原始命名空間。

開始之前

請確保 AWS 工作階段權杖的有效期限足以應付任何長時間執行的 s3 還原作業。如果權杖在還原作業期間過期、作業可能會失敗。

- 有關檢查當前會話令牌過期時間的更多資訊，請參閱 "AWS API 文件"。
- 如需 AWS 資源憑證的詳細資訊，請參閱 "AWS IAM 文件"。

使用 **CR**

步驟

1. 建立自訂資源（CR）檔案並將其命名為 `trident-protect-snapshot-ipr-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：
   - **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**：（必填）儲存快照內容的 AppVault 名稱。
   - **spec.appArchivePath**：AppVault 內儲存快照內容的路徑。您可以使用以下命令來尋找此路徑：

```
kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o
jsonpath='{.status.appArchivePath}'
```

```
---
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-snapshot-path
```

3. （可選）如果您只需要選擇應用程式中的某些資源進行還原，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，如果您選擇了持久卷聲明資源，並且它關聯了一個 pod，Trident Protect 也會恢復該關聯 pod。

   - **resourceFilter.resourceSelectionCriteria**：（篩選必需）使用 `Include`或 `Exclude`來包含或排除在 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 物件的陣列。如果在此陣列中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的欄位（group、kind、version）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 metadata.name 欄位中的名稱。
       - **resourceMatchers[].namespaces**：（可選）要篩選的資源的 Kubernetes metadata.name 欄位中的命名空間。

- **resourceMatchers[].labelSelectors**：（可選）資源在 Kubernetes metadata.name 欄位中定義的標籤選擇器字串 "Kubernetes 說明文件"。例如：`"trident.netapp.io/os=linux"`。

  例如：

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 在 `trident-protect-snapshot-ipr-cr.yaml` 檔案中填入正確的值後，套用 CR：

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

使用 CLI

步驟

1. 將快照還原到原始命名空間，並將方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 <namespace/snapshot_to_restore> \
-n <application_namespace>
```

檢查還原作業的狀態

您可以使用命令列來檢查正在進行、已完成或已失敗的還原作業狀態。

步驟

1. 使用以下命令檢索還原作業的狀態，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** 還原設定

您可以利用註釋、命名空間設定和儲存選項等進階設定來自訂還原作業，以滿足您的特定需求。

還原和容錯移轉作業期間的命名空間註釋和標籤

在還原和容錯移轉作業期間，目的地命名空間中的標籤和註釋會與來源命名空間中的標籤和註釋相符。來源命名空間中存在但目的地命名空間中不存在的標籤或註釋會被新增，而任何已存在的標籤或註釋則會被覆寫以符合來源命名空間中的值。僅存在於目的地命名空間中的標籤或註釋則保持不變。

如果您使用 Red Hat OpenShift，請務必注意命名空間註解在 OpenShift 環境中的關鍵作用。命名空間註解可確保復原的 Pod 遵循 OpenShift 安全上下文約束（SCC）定義的相應權限和安全性配置，並能無權限問題地存取磁碟區。如需更多資訊，請參閱"OpenShift 安全上下文約束文檔"。

您可以透過在執行復原或故障轉移操作之前設定 Kubernetes 環境變數 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`，來防止目標命名空間中的特定註解被覆寫。例如：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

執行還原或容錯移轉作業時， `restoreSkipNamespaceAnnotations` 和 `restoreSkipNamespaceLabels` 中指定的任何命名空間註釋和標籤都會從還原或容錯移轉作業中排除。請確保在初始 Helm 安裝期間配置這些設定。若要深入瞭解、請參閱 "設定其他 Trident Protect Helm Chart 設定"。

如果您使用 Helm 並帶有 `--create-namespace` 標誌安裝了來源應用程式，則會對 `name` 標籤鍵進行特殊處理。在還原或容錯移轉過程中，Trident Protect 會將此標籤複製到目的地命名空間，但如果來源的值與來源命名空間相符，則會將值更新為目的地命名空間值。如果此值與來源命名空間不相符，則會將其複製到目的地命名空間而不做任何變更。

範例

以下範例展示了來源命名空間和目標命名空間，它們各自具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，以及註解和標籤在目標命名空間中是如何組合或覆蓋的。

在還原或容錯移轉作業之前

下表說明了復原或容錯移轉作業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」	• environment=production • 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

還原作業後

下表展示了復原或故障轉移作業後範例目標命名空間的狀態。一些鍵已被添加，一些鍵已被覆蓋，並且 `name` 標籤已更新以匹配目標命名空間：

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 • environment=production • 角色=資料庫

支援的欄位

本節說明可用於還原作業的其他欄位。

儲存類別對應

此 `spec.storageClassMapping`屬性定義了從來源應用程式中的儲存類別到目標叢集上新儲存類別的對應。在將應用程式遷移到具有不同儲存類別的叢集之間，或變更 BackupRestore 操作的儲存後端時，可以使用此屬性。

- 範例： *

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

支援的註釋

本節列出系統中用於配置各種行為的支援註解。如果使用者未明確設定註解，系統將使用預設值。

註解	類型	說明	預設值
protect.trident.netapp.io/data-mover-timeout-sec	字串	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	字串	Kopia 內容快取的大小上限（以 MB 為單位）。	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	字串	等待所有新建立的 PersistentVolumeClaims（PVC）達到 `Bound`階段的最長時間（以秒為單位），超過此時間操作將會失敗。適用於所有還原 CR 類型（BackupRestore、BackupInplaceRestore、SnapshotRestore、SnapshotInplaceRestore）。如果您的儲存後端或叢集通常需要更多時間，請使用更高的值。	"1200"（20 分鐘）

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。

還原和容錯移轉作業期間的命名空間註釋和標籤

在還原和容錯移轉作業期間，目的地命名空間中的標籤和註釋會與來源命名空間中的標籤和註釋相符。來源命名空間中存在但目的地命名空間中不存在的標籤或註釋會被新增，而任何已存在的標籤或註釋則會被覆寫以符合來源命名空間中的值。僅存在於目的地命名空間中的標籤或註釋則保持不變。

如果您使用 Red Hat OpenShift，請務必注意命名空間註解在 OpenShift 環境中的關鍵作用。命名空間註解可確保復原的 Pod 遵循 OpenShift 安全上下文約束（SCC）定義的相應權限和安全性配置，並能無權限問題地存取磁碟區。如需更多資訊，請參閱"OpenShift 安全上下文約束文檔"。

您可以透過在執行復原或故障轉移操作之前設定 Kubernetes 環境變數 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`，來防止目標命名空間中的特定註解被覆寫。例如：

```
helm upgrade trident-protect -n trident-protect netapp-trident-protect/trident-protect \
  --set-string restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_key_to_skip_2>}" \
  --reuse-values
```

執行還原或容錯移轉作業時， `restoreSkipNamespaceAnnotations` 和 `restoreSkipNamespaceLabels` 中指定的任何命名空間註釋和標籤都會從還原或容錯移轉作業中排除。請確保在初始 Helm 安裝期間配置這些設定。若要深入瞭解、請參閱 "設定其他 Trident Protect Helm Chart 設定"。

如果您使用 Helm 並帶有 `--create-namespace` 標誌安裝了來源應用程式，則會對 `name` 標籤鍵進行特殊處理。在還原或容錯移轉過程中，Trident Protect 會將此標籤複製到目的地命名空間，但如果來源的值與來源命名空間相符，則會將值更新為目的地命名空間值。如果此值與來源命名空間不相符，則會將其複製到目的地命名空間而不做任何變更。

範例

以下範例展示了來源命名空間和目標命名空間，它們各自具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，以及註解和標籤在目標命名空間中是如何組合或覆蓋的。

在還原或容錯移轉作業之前

下表說明了復原或容錯移轉作業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」	• environment=production • 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

還原作業後

下表展示了復原或故障轉移作業後範例目標命名空間的狀態。一些鍵已被添加，一些鍵已被覆蓋，並且 `name` 標籤已更新以匹配目標命名空間：

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 • environment=production • 角色=資料庫

您可以設定 Trident Protect 在資料保護作業期間凍結及解除凍結檔案系統。"深入瞭解如何使用 Trident Protect 設定檔案系統凍結"。

故障轉移和反向操作期間的執行掛鉤

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。故障轉移後，執行鉤子將存在於應用程式中，並在執行任何相關操作時運行。
- 在反向同步或反向重同步過程中，應用程式上所有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，這些執行鉤子將失效並被刪除以防止其執行。

若要深入瞭解執行掛鉤，請參閱 "管理 Trident Protect 執行掛鉤"。

建立複寫關係

建立複寫關係涉及下列項目：

- 選擇 Trident Protect 拍攝應用程式快照的頻率（包括應用程式的 Kubernetes 資源以及應用程式每個磁碟區的磁碟區快照）
- 選擇複寫排程（包括 Kubernetes 資源以及持續性磁碟區資料）
- 設定拍攝 Snapshot 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 "AppVault 自訂資源" 中的範例以適應您的環境：

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. 設定下列屬性：

- **metadata.name**：（必填）AppVault 自訂資源的名稱。請記下您選擇的名稱，因為複寫關係所需的其他 CR 檔案會引用此值。
- **spec.providerConfig**：（必要）儲存使用指定提供者存取 AppVault 所需的組態。選擇 bucketName 以及提供者所需的任何其他詳細資料。請記下您選擇的值，因為複寫關係所需的其他 CR 檔案會參照這些值。如需其他提供者的 AppVault CR 範例，請參閱 "AppVault 自訂資源"。
- **spec.providerCredentials**：（*Required*）儲存使用指定提供者存取 AppVault 所需的任何憑證參考。
  - **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 secret 名稱。必須位於同一個 namespace 中。
  - **spec.providerCredentials.secretAccessKey**：（必要）用於存取提供者的存取金鑰。**name** 應與 **spec.providerCredentials.valueFromSecret.name** 相符。
- **spec.providerType**：（必填）確定備份服務商；例如，NetApp ONTAP S3、通用 S3、Google Cloud 或 Microsoft Azure。可選值：
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. 在 `trident-protect-appvault-primary-source.yaml` 檔案中填入正確的值後，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. 設定下列屬性：

- **metadata.name**：（必填）應用程式自訂資源的名稱。請記下您選擇的名稱，因為複寫關係所需的其他 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，並可選擇性地使用標籤來縮小命名空間的範圍，以指定此處列出的命名空間中存在的資源。應用程式命名空間必須包含在此陣列中。

  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. 在 `trident-protect-app-source.yaml` 檔案中填入正確的值後，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。若要建立按需快照，請參閱 "建立隨需快照"。

4. 在來源叢集上、建立複寫排程 CR：

除了下方提供的排程之外，建議建立一個單獨的每日快照排程，並將保留期設定為 7 天，以便在對等 ONTAP 叢集之間維護一個通用的快照。這樣可以確保快照最多可用 7 天，但保留期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進行最多 7 天的逆向操作。這種方法可以加快逆向過程並提高效率，因為只會傳輸自上次快照以來所做的變更，而不是所有資料。

如果應用程式的現有排程已符合所需的保留要求，則不需要其他排程。

使用 **CR** 建立複寫排程

a. 為來源應用程式建立複寫排程：

   i. 建立自訂資源（CR）檔案並將其命名為（例如、 `trident-protect-schedule.yaml`）。

   ii. 設定下列屬性：

   - **metadata.name**：（必填）排程自訂資源的名稱。
   - **spec.appVaultRef**：（必填）此值必須與來源應用程式的 AppVault metadata.name 欄位相符。
   - **spec.applicationRef**：（必填）此值必須與來源應用程式 CR 的 metadata.name 欄位相符。
   - **spec.backupRetention**：（必填）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. 在 `trident-protect-schedule.yaml` 檔案中填入正確的值後，套用 CR：

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-
namespace
```

使用 **CLI** 建立複寫排程

a. 建立複寫排程，並將括號中的值替換為您環境中的資訊：

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

- 範例： *

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. 在目標叢集上，建立一個與在來源叢集上應用的 AppVault CR 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. 套用 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 "AppVault 自訂資源" 中的範例以適應您的環境：
   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. 設定下列屬性：
      - **metadata.name**：（必填）AppVault 自訂資源的名稱。請記下您選擇的名稱，因為複寫關係所需的其他 CR 檔案會引用此值。
      - **spec.providerConfig**：（必要）儲存使用指定提供者存取 AppVault 所需的組態。選擇一個 `bucketName`以及提供者所需的任何其他詳細資訊。請記下您選擇的值，因為複寫關係所需的其他 CR 檔案會參照這些值。請參閱"AppVault 自訂資源"以取得其他提供者的 AppVault CR 範例。
      - **spec.providerCredentials**：（*Required*）儲存使用指定提供者存取 AppVault 所需的任何憑證參考。
        - **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 secret 名稱。必須位於同一個 namespace 中。
        - **spec.providerCredentials.secretAccessKey**：（必要）用於存取提供者的存取金鑰。**name** 應與 **spec.providerCredentials.valueFromSecret.name** 相符。

- **spec.providerType**：（必填）確定備份服務商；例如，NetApp ONTAP S3、通用 S3、Google Cloud 或 Microsoft Azure。可選值：
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. 在 `trident-protect-appvault-secondary-destination.yaml` 檔案中填入正確的值後，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 時，請在套用 CR 之前手動建立目的地命名空間。Trident Protect 僅在使用 CLI 時才會自動建立命名空間。

使用 **CR** 建立 **AppMirrorRelationship**

a. 建立自訂資源（CR）檔案並將其命名為（例如、 `trident-protect-relationship.yaml` ）。

b. 設定下列屬性：

- **metadata.name:** （必填）AppMirrorRelationship 自訂資源的名稱。
- **spec.destinationAppVaultRef**：（必填）此值必須與目標叢集上目標應用程式的 AppVault 名稱相符。
- **spec.namespaceMapping**：（必需）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**：（*Required*）此值必須與來源應用程式的 AppVault 名稱相符。
- **spec.sourceApplicationName**：（必需）此值必須與您在來源應用程式 CR 中定義的來源應用程式名稱相符。
- **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**：（選用）選擇叢集上有效儲存類別的名稱。儲存類別必須連結至與來源環境對等的 ONTAP 儲存 VM。如果未提供儲存類別，則預設會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. 在 `trident-protect-relationship.yaml` 檔案中填入正確的值後，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 物件，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>
```

- 範例： *

```
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、檢查複寫關係的狀態和狀態：

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

故障轉移至目的地叢集

使用 Trident Protect，您可以將複製的應用程式容錯移轉至目的地叢集。此程序會停止複寫關係，並使應用程式在目的地叢集上線上。如果來源叢集上的應用程式正在運作，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 檔案。

3. 套用 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排程。
5. （可選）檢查複寫關係的狀態和狀態：

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

重新同步故障轉移的複寫關係

重新同步作業會重新建立複寫關係。執行重新同步作業後、原始來源應用程式會成為執行中的應用程式、而目的地叢集上執行中應用程式所做的任何變更都會被捨棄。

該過程會在重新建立複寫之前停止目的地叢集上的應用程式。

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. 選用：在來源叢集上，建立來源應用程式的快照。這可確保擷取來源叢集的最新變更。
2. 在目標叢集上，編輯 AppMirrorRelationship CR 檔案（例如， `trident-protect-relationship.yaml` ），並將 spec.desiredState 的值變更為 `Established`。
3. 儲存 CR 檔案。
4. 套用 CR：

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. 如果您在目的地叢集上建立了任何保護排程來保護故障轉移的應用程式，請將其移除。任何保留的排程都會導致磁碟區快照失敗。

反向重新同步已容錯移轉的複寫關係

當您對故障轉移的複製關係進行反向同步時，目標應用程式將變為來源應用程式，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。這將使目標叢集變為來源叢集。如果新目標叢集上仍有任何保護排程，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，以建立複寫關係。
3. 確保新目標（原始來源叢集）配置了兩個 AppVault CR。
4. 在相反的叢集上建立複寫關係，並設定反向的值。

反向應用程式複寫方向

當您反轉複製方向時、Trident Protect 會將應用程式移至目的地儲存後端、同時繼續複寫回原始來源儲存後端。Trident Protect 會停止來源應用程式、並在容錯移轉至目的地應用程式之前、將資料複寫至目的地。

在這種情況下，您正在交換來源和目的地。

步驟

1. 在來源叢集上，建立關機快照：

使用 **CR** 建立關機快照

a. 停用來源應用程式的保護原則排程。

b. 建立 ShutdownSnapshot CR 檔案：

i. 建立自訂資源（CR）檔案並將其命名為（例如、 `trident-protect-shutdownsnapshot.yaml`）。

ii. 設定下列屬性：

- **metadata.name**：（必填）自訂資源的名稱。
- **spec.AppVaultRef**：（*Required*）此值必須與來源應用程式的 AppVault metadata.name 欄位相符。
- **spec.ApplicationRef**：（必填）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. 在 `trident-protect-shutdownsnapshot.yaml` 檔案中填入正確的值後，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，並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 Snapshot 完成後，取得關機 Snapshot 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；這是最後一個斜槓之後的所有內容）：

```
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容錯移轉，並進行以下變更：

> 在故障轉移過程的步驟 2 中，將 `spec.promotedSnapshot` 欄位包含在 AppMirrorRelationship CR 檔案中，並將其值設定為您在上面的步驟 3 中記錄的基本名稱。

5. 在 [反向重新同步已容錯移轉的複寫關係] 中執行反向重新同步步驟。
6. 在新來源叢集上啟用保護排程。

結果

由於反向複寫，會發生以下動作：

- 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 Volume 進行快照並進行複寫。
- SnapMirror 關係已中斷，使目的地磁碟區準備好進行讀取 / 寫入。
- 該應用程式的 Kubernetes 資源是從關閉前的快照中還原，使用的是在原始來源應用程式關閉後複寫的磁碟區資料。
- 複寫會以相反的方向重新建立。

將應用程式容錯回復至原始來源叢集

使用 Trident Protect，您可以透過以下步驟在故障轉移作業後實現「故障復原」。在此工作流程中，為了恢復原始複寫方向，Trident Protect 會在反轉複寫方向之前，將所有應用程式變更複寫（重新同步）回原始來源應用程式。

此程序從已完成容錯移轉至目的地的關係開始，並涉及下列步驟：

- 從故障轉移狀態開始。
- 反向重新同步複寫關係。

> 不要執行正常的重新同步作業，因為這會捨棄在容錯移轉程序期間寫入目的地叢集的資料。

- 反轉複寫方向。

步驟

1. 執行[反向重新同步已容錯移轉的複寫關係]步驟。
2. 執行[反向應用程式複寫方向]步驟。

刪除複寫關係

您可以隨時刪除複寫關係。刪除應用程式複寫關係後，將產生兩個彼此獨立的應用程式，它們之間沒有任何關聯。

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，在叢集之間遷移應用程式或將應用程式遷移到不同的儲存類別。

遷移應用程式時，所有為該應用程式配置的執行鉤子都會隨應用程式一起遷移。如果存在復原後執行鉤子，它會在復原操作過程中自動執行。

備份與還原作業

若要針對下列案例執行備份與還原作業，您可以自動執行特定的備份與還原工作。

複製到同一叢集

若要將應用程式複製到同一個叢集、請建立快照或備份、然後將資料還原到同一個叢集。

步驟

1. 請執行下列其中一項操作：
   a. "建立快照".
   b. "建立備份".
2. 在同一叢集上，根據您建立的是快照還是備份，執行下列其中一項：
   a. "從快照還原資料".
   b. "從備份中恢復資料".

複製到不同的叢集

若要將應用程式複製到另一個叢集（執行跨叢集複製），請在來源叢集上建立備份，然後將備份還原到另一個叢集。確保目標叢集上已安裝 Trident Protect。

您可以使用 "SnapMirror 複製" 在不同的叢集之間複製應用程式。

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. 在目標叢集上，"從備份中恢復資料"。

將應用程式從一個儲存類別遷移到另一個儲存類別

您可以透過將備份還原到目標儲存類別，將應用程式從一個儲存類別遷移到另一個儲存類別。

例如（不包括還原 CR 中的機密）：

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    - destination: "${destinationNamespace}"
      source: "${sourceNamespace}"
  storageClassMapping:
    - destination: "${destinationStorageClass}"
      source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

使用 **CR** 還原快照

步驟

1. 建立自訂資源（CR）檔案並將其命名為 `trident-protect-snapshot-restore-cr.yaml`。
2. 在您建立的檔案中、設定以下屬性：
   - **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來尋找此路徑：

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**：（必填）儲存快照內容的 AppVault 名稱。
   - **spec.namespaceMapping**：復原作業的來源命名空間到目標命名空間的對應。請將 `my-source-namespace`和 `my-destination-namespace`替換為您環境中的資訊。

     ```
     ---
     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. （選用）如果您需要僅選擇要還原的應用程式的某些資源，請新增篩選條件，以包含或排除標記有特定標籤的資源：
   - **resourceFilter.resourceSelectionCriteria**：（篩選必需）使用 `include or exclude`包含或排除在 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 物件的陣列。如果在此陣列中定義多個元素，則它們之間按 OR 運算匹配，每個元素內的欄位（group、kind、version）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 metadata.name 欄位中的名稱。
       - **resourceMatchers[].namespaces**：（可選）要篩選的資源的 Kubernetes metadata.name 欄位中的命名空間。

- **resourceMatchers[].labelSelectors**：（可選）資源在 Kubernetes metadata.name 欄位中定義的標籤選擇器字串 "Kubernetes 說明文件"。例如：`"trident.netapp.io/os=linux"`。

  例如：

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 在 `trident-protect-snapshot-restore-cr.yaml` 檔案中填入正確的值後，套用 CR：

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

使用 **CLI** 還原快照

步驟

1. 將快照還原到不同的命名空間，並將括號中的值替換為您環境中的資訊。

   - `snapshot`引數使用命名空間和快照名稱，格式為 `<namespace>/<name>`。
   - `namespace-mapping` 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 snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

管理 Trident Protect 執行掛鉤

執行鉤子是一種自訂操作，您可以將其配置為與託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式，則可以使用執行鉤子在建立快照之前暫停所有資料庫事務，並在快照完成後恢復事務。這可以確保應用程式快照的一致性。

執行掛鉤的類型

Trident Protect 支援以下幾種執行鉤子類型，取決於它們的執行時機：

- 快照前
- 快照後
- 備份前
- 備份後
- 還原後
- 故障轉移後

執行順序

執行資料保護作業時，執行掛鉤事件會依照以下順序發生：

1. 任何適用的自訂預操作執行鉤子都會在相應的容器上運行。您可以建立並運行任意數量的自訂預操作鉤子，但這些鉤子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，檔案系統會凍結。"深入瞭解如何使用 Trident Protect 設定檔案系統凍結"。
3. 執行資料保護作業。
4. 如果適用，則解凍已凍結的檔案系統。
5. 任何適用的自訂後操作執行鉤子都會在相應的容器上運行。您可以建立並運行任意數量的自訂後操作鉤子，但這些鉤子在操作後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鉤子（例如，快照前鉤子），則這些鉤子的執行順序無法保證。但是，不同類型鉤子的執行順序是有保證的。例如，以下是包含所有不同類型鉤子的組態的執行順序：

1. 已執行快照前鉤子
2. 執行快照後掛鉤
3. 執行備份前掛鉤
4. 執行備份後掛鉤

前述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鉤子腳本之前，請務必先對其進行測試。您可以使用 'kubectl exec' 指令方便地測試這些腳本。在生產環境中啟用執行鉤子後，請測試產生的快照和備份，以確保它們的一致性。您可以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。

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還原作業中。

關於自訂執行掛鉤的重要說明

在規劃應用程式的執行鉤子時,請考慮以下幾點。

- 執行掛鉤必須使用指令碼來執行動作。許多執行掛鉤可以參照同一個指令碼。
- Trident Protect 要求執行鉤子使用的腳本以可執行 shell 腳本的格式編寫。
- 指令碼大小限制為 96KB。
- Trident Protect 使用執行掛鉤設定和任何相符條件來判斷哪些掛鉤適用於快照、備份或還原作業。

由於執行鉤子通常會降低甚至完全停用其所針對應用程式的功能，因此您應該始終盡量縮短自訂執行鉤子的執行時間。如果您啟動了一個帶有關聯執行鉤子的備份或快照操作，但隨後取消了該操作，只要備份或快照操作已經開始，這些鉤子仍然可以執行。這意味著備份後執行鉤子中使用的邏輯不能假定備份已經完成。

執行掛鉤篩選器

當您新增或編輯應用程式的執行鉤子時，您可以為該執行鉤子添加過濾器，以管理鉤子將匹配哪些容器。過濾器適用於所有容器都使用相同容器映像，但每個映像可能用於不同用途的應用程式（例如 Elasticsearch）。過濾器可讓您建立執行鉤子在部分（但不一定全部）相同容器上運行的場景。如果為單一執行鉤子建立多個過濾器，它們將透過邏輯 AND 運算子進行組合。每個執行鉤子最多可以有 10 個活動過濾器。

新增到執行鉤子的每個過濾器都使用正規表示式來匹配叢集中的容器。當鉤子匹配到容器時，它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 支援的執行鉤子過濾器正規表示式語法的更多資訊，請參閱 "Regular Expression 2 (RE2) 語法支援"。

如果在還原或複製作業之後執行的執行鉤子中新增命名空間篩選器，且還原或複製來源和目的地位於不同的命名空間中，則命名空間篩選器僅套用於目的地命名空間。

執行掛鉤範例

造訪 "NetApp Verda GitHub 專案"下載適用於 Apache Cassandra 和 Elasticsearch 等熱門應用程式的真實執行鉤子。您也可以查看範例，並從中取得建置自訂執行鉤子的想法。

建立執行掛鉤

您可以使用 Trident Protect 為應用程式建立自訂執行鉤子。您需要擁有 Owner、Admin 或 Member 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 環境和叢集組態：
   - **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**：（必填）要執行執行鉤子的應用程式的 Kubernetes 名稱。
   - **spec.stage**：（必需）一個字串，指示執行鉤子應在操作的哪個階段運行。可能的值：
     - 預先
     - 發佈
   - **spec.action**：（必要）一個字串，指示執行鉤子將執行的操作，假設符合任何指定的執行鉤子篩選器。可能的值：
     - 快照
     - 備份
     - 還原
     - 容錯移轉
   - **spec.enabled**：（可選）指示此執行鉤子是否啟用或停用。如果未指定，則預設值為 true。
   - **spec.hookSource**：（必填）包含 base64 編碼的 hook 腳本的字串。
   - **spec.timeout**：（可選）一個數字，定義執行鉤子允許運行的最長時間（以分鐘為單位）。最小值為 1 分鐘，如果未指定，則預設值為 25 分鐘。
   - **spec.arguments**：（可選）您可以為執行掛鉤指定的 YAML 引數清單。
   - **spec.matchingCriteria**：（可選）條件鍵值對的可選清單，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新增 10 個過濾器。
   - **spec.matchingCriteria.type**：（可選）用於識別執行鉤子過濾器類型的字串。可能的值：
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**：（可選）識別執行鉤子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，並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>
```

手動執行執行掛鉤

您可以手動執行 execution hook 以進行測試，或者在失敗後需要手動重新執行 hook。您需要擁有 Owner 、Admin 或 Member 權限才能手動執行 execution hook。

手動執行執行掛鉤包含兩個基本步驟：

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. 針對備份執行執行掛鉤

步驟 **1**：建立資源備份

使用 **CR**

步驟

1. 建立自訂資源（CR）檔案並將其命名為 `trident-protect-resource-backup.yaml`。
2. 設定以下屬性以符合您的 Trident Protect 環境和叢集組態：
   - **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**：（必填）要為其建立資源備份的應用程式的 Kubernetes 名稱。
   - **spec.appVaultRef**：（必填）儲存備份內容的 AppVault 名稱。
   - **spec.appArchivePath**：AppVault 內儲存備份內容的路徑。您可以使用以下命令來尋找此路徑：

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

     YAML 範例：

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: ResourceBackup
     metadata:
       name: example-resource-backup
     spec:
       applicationRef: my-app-name
       appVaultRef: my-appvault-name
       appArchivePath: example-resource-backup
     ```

3. 在用正確的值填入 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，並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. 檢視備份狀態。您可以重複使用此範例命令，直到作業完成：

```
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**：執行 **execution hook**

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 環境和叢集組態：
   - **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**：（必填）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**：（必填）請確保此值與您在步驟 1 中建立的 ResourceBackup CR 的 appVaultRef 相符。
   - **spec.appArchivePath**：請確保此值與您在步驟 1 中建立的 ResourceBackup CR 內的 appArchivePath 相符。

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.action**：（必要）一個字串，指示執行鉤子將執行的操作，假設符合任何指定的執行鉤子篩選器。可能的值：
     - 快照
     - 備份
     - 還原
     - 容錯移轉
   - **spec.stage**：（必需）一個字串，指示執行鉤子應在操作的哪個階段運行。此鉤子運行不會在任何其他階段運行鉤子。可能的值：
     - 預先
     - 發佈

     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. 建立手動執行掛鉤執行請求：

```
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. 檢查執行掛鉤執行的狀態。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：

   版本 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 Protect：

   ```
   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 部落格

您可以在這裡找到一些很棒的 NetApp Trident 和 Trident Protect 部落格：

Trident 部落格

- 2025 年 10 月 16 日："適用於 Kubernetes 的進階儲存解決方案"
- 2025 年 8 月 19 日："增強資料一致性：OpenShift 虛擬化中使用 Trident 的 Volume Group Snapshot"
- 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 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 Fiber Channel Protocol 在 Kubernetes 上實現持久儲存"
- 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："使用全新認證操作員簡化 Red Hat OpenShift 上的 Trident 安裝"
- 2025 年 3 月 27 日："使用 Google Cloud NetApp Volumes 為 SMB 配置 Trident"
- 2025 年 3 月 5 日："解鎖無縫的 iSCSI 儲存整合：AWS 上 ROSA 叢集的 FSxN 指南"
- 2025 年 2 月 27 日："使用 Trident、GKE 和 Google Cloud NetApp Volumes 部署雲端身分"
- 2024 年 12 月 12 日："在 Trident 中引入光纖通道支援"
- 2024 年 11 月 11 日："NetApp Trident 與 Google Cloud NetApp Volumes"
- 2024 年 10 月 29 日："在 AWS 上使用 Trident 的 Amazon FSx for NetApp ONTAP 搭配 Red Hat OpenShift Service（ROSA）"
- 2024 年 10 月 29 日："使用 ROSA 和 Amazon FSx for NetApp ONTAP 進行 OpenShift Virtualization 的虛擬機即時遷移"
- 2024 年 7 月 8 日："使用 NVMe/TCP 在 Amazon EKS 上使用 ONTAP 儲存來支援您的現代容器化應用程式"
- 2024 年 7 月 1 日："使用 Google Cloud NetApp Volumes Flex 和 Astra Trident 實現無縫 Kubernetes 儲存"
- 2024 年 6 月 11 日："ONTAP 作為 OpenShift 中整合影像註冊表的後端儲存"

Trident Protect 部落格

- 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："OpenShift Virtualization 使用 NetApp Trident Protect 進行災難復原"
- 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 還原後掛勾重新調整 Kubernetes 應用程式"
- 2025 年 4 月 3 日："Trident Protect 功能升級：Kubernetes 複寫以提供保護與災難恢復"
- 2025 年 3 月 13 日："OpenShift Virtualization VM 的當機一致性備份與還原作業"
- 2025 年 3 月 11 日："將 GitOps 模式擴展到使用 NetApp Trident 的應用程式資料保護"
- 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："Kubernetes 驅動的資料管理：Trident Protect 開啟新時代"

知識與支援

常見問題

尋找有關安裝、設定、升級及疑難排解 Trident 的常見問題解答。

一般性問題

Trident 發布頻率如何？

從 24.02 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 版本功能。Trident 可能會在 Kubernetes beta 版本發布後的兩個 Trident 版本中支援 beta 版本功能。

Trident 的運作是否依賴其他 **NetApp** 產品？

Trident 不依賴任何其他 NetApp 軟體產品，可以作為獨立應用程式運作。但是，您需要一個 NetApp 後端儲存設備。

如何取得完整的 **Trident** 組態詳細資料？

使用 `tridentctl get` 命令可以獲得有關 Trident 組態的更多資訊。

我能否取得有關 **Trident** 如何配置儲存設備的指標？

是的。Prometheus 端點可用於收集有關 Trident 運行的信息，例如管理的後端數量、已配置的磁碟區數量、已消耗的位元組數等等。您也可以使用 "Cloud Insights" 進行監控和分析。

使用 **Trident** 作為 **CSI Provisioner** 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`。如果您想使用目前版本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 是否支援從私人登錄進行離線安裝？

是的，Trident 可以離線安裝。請參閱 "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 `kubectl`有權存取叢集的機器進行遠端安裝。 `kubectl`驗證存取權限後（例如， `kubectl get nodes`從遠端機器啟動命令進行驗證），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，僅包含一個實例，因此內建了高可用性（HA）。您不應增加 Deployment 的副本數。如果 Trident 所在的節點遺失或 Pod 無法訪問，Kubernetes 會自動將 Pod 重新部署到叢集中運行正常的節點。Trident 僅支援控制平面，因此 Trident 重新部署時不會影響目前已掛載的 Pod。

Trident 是否需要存取 **kube-system** 命名空間？

Trident 從 Kubernetes API Server 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，該實例擁有對 Kubernetes 叢集的 PersistentVolume、PersistentVolumeClaim、StorageClass 和 Secret 資源的特定存取權。請參閱"自訂 tridentctl 安裝"。

我可以在本機產生 **Trident** 用於安裝的確切資訊清單檔案嗎？

如有需要，您可以在本機產生和修改 Trident 安裝時使用的確切資訊清單檔案。請參閱 "自訂 tridentctl 安裝"。

我可以為兩個獨立的 **Kubernetes** 叢集中的兩個獨立的 **Trident** 執行個體共用同一個 **ONTAP** 後端 **SVM** 嗎？

雖然不建議，但您可以為兩個 Trident 實例使用相同的後端 SVM。在安裝期間為每個實例指定唯一的磁碟區名稱，和／或在 `StoragePrefix`檔案中指定唯一的 `setup/backend.json`參數。這是為了確保同一個FlexVol volume不會同時用於兩個實例。

是否可以在 **ContainerLinux**（以前稱為 **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** 搭配使用嗎？

是的，Trident 在 AWS、Google Cloud 和 Azure 上均受支援。

疑難排解與支援

NetApp 是否支援 **Trident**？

雖然 Trident 是開源且免費提供的，但只要您的 NetApp 後端受支援，NetApp 就能提供完整支援。

我該如何提交支援案例？

若要提出支援案例，請執行下列其中一項操作：

1. 請聯絡您的支援客戶經理，以取得協助提交工單。
2. 請聯絡 "NetApp 支援"提出支援案例。

如何產生支援記錄套件？

您可以透過執行 `tridentctl logs -a`來建立支援套件。除了套件中擷取的日誌之外、還需擷取 kubelet 日誌、以便

診斷 Kubernetes 端的掛載問題。取得 kubelet 日誌的步驟會因 Kubernetes 的安裝方式而有所不同。

如果我需要提出新功能請求，該怎麼辦？

在 "Trident Github"上建立問題，並在問題的主題和描述中提及 **RFE**。

我應該在哪裡提出缺陷？

在 "Trident Github"上建立問題。務必包含所有與問題相關的必要資訊和日誌。

如果我對 **Trident** 有需要釐清的快速問題，會發生什麼情況？是否有社群或論壇？

如果您有任何疑問、問題或請求，請透過我們的 Trident "Discord 頻道"或 GitHub 與我們聯絡。

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 `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`更新後端密碼。將範例中的 `myBackend`替換為您的後端名稱，並將 `/path/to_new_backend.json`替換為正確 `backend.json`檔案的路徑。

Trident 找不到我的 **Kubernetes** 節點。我該如何解決這個問題？

Trident 找不到 Kubernetes 節點可能有兩種情況。可能是 Kubernetes 內部的網路問題或 DNS 問題。在每個 Kubernetes 節點上執行的 Trident 節點 daemonset 必須能夠與 Trident 控制器通訊，才能向 Trident 註冊節點。如果在安裝 Trident 之後發生網路變更，則只有在新增至叢集的新 Kubernetes 節點上才會遇到此問題。

如果 **Trident Pod** 被摧毀、我會遺失資料嗎？

即使 Trident Pod 被銷毀，資料也不會遺失。Trident 元資料儲存在 CRD 物件中。所有由 Trident 配置的 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。建議在正式作業部署之前、先在實驗室環境中測試升級。

是否可以將 **Trident** 降級至先前的版本？

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 "解除安裝 Trident" 並按照相應版本的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。

管理後端和磁碟區

我是否需要在 **ONTAP** 後端定義檔中同時定義 **Management** 和 **DataLIF** ？

管理 LIF 是強制性的。DataLIF 各不相同：

- ONTAP SAN：請勿為 iSCSI 指定。Trident 使用"ONTAP Selective LUN Map"來探索建立多路徑工作階段所需的 iSCSI LIF。如果明確定義 `dataLIF`，則會產生警告。如需詳細資訊，請參閱"ONTAP SAN 配置選項和

範例"。

- ONTAP NAS：NetApp 建議指定 `dataLIF`。如果未提供、Trident 會從 SVM 擷取 dataLIF。您可以指定完整網域名稱（FQDN）以用於 NFS 掛載作業、讓您建立循環 DNS 以在多個 dataLIF 之間進行負載平衡。如需詳細資料、請參閱"ONTAP NAS 設定選項和範例"

Trident 能否為 **ONTAP** 後端設定 **CHAP**？

是的。Trident 支援 ONTAP 後端的雙向 CHAP 協定。這需要在後端配置中設定 `useCHAP=true`。

如何使用 **Trident** 管理匯出原則？

Trident 可從 20.04 版開始動態建立及管理匯出原則。這可讓儲存管理員在其後端組態中提供一或多個 CIDR 區塊，並讓 Trident 將落在這些範圍內的節點 IP 新增至其建立的匯出原則。如此一來，Trident 就能自動管理 IP 位於指定 CIDR 內之節點的規則新增與刪除作業。

IPv6 位址可以用於 **Management** 和 **DataLIF** 嗎？

Trident 支援為以下項目定義 IPv6 位址：

- `managementLIF` 和 `dataLIF` 適用於 ONTAP NAS 後端。
- `managementLIF` 適用於 ONTAP SAN 後端。您無法在 ONTAP SAN 後端上指定 `dataLIF`。

Trident 必須使用標誌 `--use-ipv6`（用於 `tridentctl`安裝）、 `IPv6`（用於 Trident operator）或 `tridentTPv6`（用於 Helm 安裝）進行安裝，才能透過 IPv6 運行。

是否可以在後端更新管理 **LIF** ？

是的，可以使用 `tridentctl update backend` 指令更新後端管理 LIF 。

是否可以在後端更新 **DataLIF** ？

您只能更新 `ontap-nas` 和 `ontap-nas-economy` 上的 DataLIF 。

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式，也可以使用不同的驅動程式。

Trident 如何儲存後端憑證？

Trident 將後端認證資料儲存為 Kubernetes Secrets 。

Trident 如何選擇特定後端？

如果後端屬性不能用於自動為類別選擇正確的池、則會使用 `storagePools` 和 `additionalStoragePools` 參數來選擇一組特定的池。

如何確保 **Trident** 不會從特定的後端進行配置？

`excludeStoragePools` 參數用於篩選 Trident 用於配置的儲存池集，並將移除任何符合的儲存池。

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 `StorageClass`和 `PersistentVolumeClaim`中存在的參數選擇適當的後端。例如、如果存在多個 ontap-nas 驅動程式後端、Trident 會嘗試符合 `StorageClass`和 `PersistentVolumeClaim`中的參數組合、並且符合能夠滿足 `StorageClass`和 `PersistentVolumeClaim`中列出的要求的後端。如果存在多個符合請求的後端、Trident 會從中隨機選擇一個。

Trident 是否支援與 **Element/SolidFire** 的雙向 **CHAP**？

是的。

Trident 如何在 **ONTAP** 磁碟區上部署 **Qtree**？單一磁碟區上可以部署多少個 **Qtree**？

此 `ontap-nas-economy`驅動程式在同一個 FlexVol Volume 中最多可建立 200 個 Qtree（可設定為 50 到 300 個）、每個叢集節點 100,000 個 Qtree、每個叢集 2.4M 個 Qtree。當您輸入由經濟型驅動程式提供服務的新 `PersistentVolumeClaim`時、驅動程式會查看是否已存在可為新 Qtree 提供服務的 FlexVol Volume。如果不存在可為 Qtree 提供服務的 FlexVol Volume、則會建立新的 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 配置的磁碟區設定 Unix 權限。

在配置磁碟區時，如何設定一組明確的 **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 Storage Class 中指定掛載選項、請依照 "這裡" 提供的範例進行操作。

如何將已配置的磁碟區設定為特定的匯出原則？

若要允許對應的主機存取磁碟區，請使用後端定義檔中配置的 `exportPolicy`參數。

如何透過 **Trident** 與 **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 提供的磁碟區進行加密。如需詳細資訊，請參閱：["Trident 與 NVE 和 NAE 的運作方式"]

透過 **Trident** 為 **ONTAP** 實作 **QoS** 的最佳方法為何？

使用 `StorageClasses` 來實作 ONTAP 的 QoS。

如何透過 **Trident** 指定精簡配置或完整配置？

ONTAP 驅動程式支援精簡配置或厚配置。ONTAP 驅動程式預設使用精簡配置。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`則以優先。請為 ONTAP 配置以下項目：

1. 在 `StorageClass`上，將 `provisioningType`屬性設為 thick。

2. 在後端定義檔中，透過設定 `backend spaceReserve parameter` 為 volume 來啟用厚磁碟區。

如何確保即使我不小心刪除了 **PVC**，正在使用的 **Volume** 也不會被刪除？

從 Kubernetes 1.10 版本開始，PVC 保護功能會自動啟用。

我可以擴充由 **Trident** 建立的 **NFS PVC** 嗎？

是的。您可以擴充由 Trident 建立的 PVC。請注意，磁碟區自動成長是 ONTAP 功能，不適用於 Trident。

我可以在 **SnapMirror** 資料保護（**DP**）或離線模式下匯入磁碟區嗎？

如果外部磁碟區處於 DP 模式或離線狀態，則磁碟區匯入失敗。您將收到以下錯誤訊息：

```
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 儲存容量充足，Kubernetes Storage Resource Quota 就應該正常運作。當 NetApp 儲存容量不足，無法滿足 Kubernetes 配額設定時，Trident 會嘗試進行資源配置，但會發生錯誤。

我可以使用 **Trident** 建立 **Volume Snapshot** 嗎？

是的。Trident 支援按需建立磁碟區快照以及從快照建立持久性磁碟區。若要從快照建立 PV ，請確保已啟用 `VolumeSnapshotDataSource` 功能閘道。

支援 **Trident Volume** 快照的驅動程式有哪些？

截至目前，我們的 `ontap-nas`、 `ontap-nas-flexgroup`、 `ontap-san`、 `ontap-san-economy`、 `solidfire-san`和 `azure-netapp-files`後端驅動程式均提供按需快照支援。

如何對 **Trident** 透過 **ONTAP** 配置的磁碟區進行快照備份？

這可用於 `ontap-nas`、 `ontap-san` 和 `ontap-nas-flexgroup` 驅動程式。您也可以在 FlexVol 層級為 `ontap-san-economy` 驅動程式指定 `snapshotPolicy`。

> `ontap-nas-economy` 驅動程式也支援此功能，但 `FlexVol` 磁碟區層級粒度，而非 `qtree` 層級粒度。若要啟用 `Trident` 所佈建磁碟區的快照功能，請將後端參數選項 `snapshotPolicy` 設定為 `ONTAP` 後端上定義的所需快照原則。儲存控制器所建立的任何快照，`Trident` 無法獲知。

我可以為透過 **Trident** 配置的磁碟區設定 **Snapshot** 保留百分比嗎？

是的，您可以透過在後端定義檔中設定 `snapshotReserve` 屬性，為儲存快照副本預留特定百分比的磁碟空間。如果您已在後端定義檔中配置 `snapshotPolicy` 和 `snapshotReserve`，則快照預留百分比將根據後端檔案中指定的 `snapshotReserve` 百分比進行設定。如果未指定 `snapshotReserve` 百分比，ONTAP 預設將快照預留百分比設定為 5。如果 `snapshotPolicy` 選項設為 none，則快照預留百分比設定為 0。

我可以直接存取 **Volume** 快照目錄並複製檔案嗎？

是的，您可以透過在後端定義檔中設定 `snapshotDir` 參數來存取 Trident 提供的磁碟區上的快照目錄。

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 進行外部設定。

如何將 **Persistent Volume** 還原到特定的 **ONTAP Snapshot** ？

若要將磁碟區還原至 ONTAP 快照、請執行下列步驟：

1. 使正在使用 Persistent Volume 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. 重新啟動應用程式 pod。

Trident 是否可以在配置了負載共享鏡像的 **SVM** 上配置磁碟區？

可以為透過 NFS 提供資料的 SVM 根磁碟區建立負載共享鏡像。ONTAP 會自動更新由 Trident 建立的磁碟區的負載共享鏡像。這可能會導致磁碟區掛載延遲。當使用 Trident 建立多個磁碟區時，磁碟區的配置取決於 ONTAP 更新負載共享鏡像。

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 的儲存資源配額來限制每個命名空間中特定儲存類別的使用量、這些配額是按命名空間設定的。若要禁止特定命名空間存取特定儲存、請將該儲存類別的資源配額設為 0。

疑難排解

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 協助，請使用 `tridentctl logs -a -n trident`建立支援套裝組合，並將其傳送給 NetApp 支援部門。

一般疑難排解

- 如果 Trident Pod 無法正常啟動（例如，當 Trident Pod 卡在 `ContainerCreating`階段且就緒容器少於兩個時），執行 `kubectl -n trident describe deployment trident`和 `kubectl -n trident describe pod trident--**`可以提供更多資訊。取得 `kubelet` 日誌（例如，透過 `journalctl -xeu kubelet`）也很有幫助。

- 如果 Trident 日誌中的資訊不足，您可以嘗試根據您的安裝選項，透過傳送 `-d`標誌給安裝參數來啟用 Trident 的偵錯模式。

  然後使用 `./tridentctl logs -n trident`確認已設定偵錯，並在日誌中搜尋 `level=debug msg`。

  已安裝 **Operator**

  ```
  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 `kubectl get pod -n trident` 輸出中的「AGE」欄來檢查這一點。

  對於 Trident 20.07 和 20.10，請使用 `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
  ```

- 您也可以在後端定義中包含 `debugTraceFlags`，以取得每個後端的偵錯日誌。例如，包含 `debugTraceFlags: {"api":true, "method":true,}`，即可在 Trident 日誌中取得 API 呼叫和方法遍歷。現有的後端可以使用 `debugTraceFlags`，並搭配 `tridentctl backend update`來設定。
- 使用 Red Hat Enterprise Linux CoreOS (RHCOS) 時，請確保在工作節點上啟用 `iscsid`並預設為啟動。這可以使用 OpenShift MachineConfigs 或透過修改 ignition 模板來實現。
- 使用 Trident 搭配 "Azure NetApp Files"時，您可能會遇到的常見問題是租用戶金鑰和客戶端金鑰來自權限不足的應用程式註冊。有關 Trident 要求的完整清單，請參閱"Azure NetApp Files"組態。
- 如果將 PV 掛載到容器時出現問題，請確保已安裝並執行 `rpcbind`。請使用主機作業系統所需的套件管理員，並檢查 `rpcbind`是否正在執行。您可以透過執行 `systemctl status rpcbind`或其等效指令，來檢查 `rpcbind`服務的狀態。
- 如果 Trident 後端報告處於 `failed` 狀態，即使先前運作正常，也可能是因為變更了與後端關聯的 SVM/管理員憑證所致。使用 `tridentctl update backend` 更新後端資訊或重新啟動 Trident pod 即可解決此問題。
- 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 `--in cluster=false`旗標安裝 Trident。這將不使用安裝程式 Pod，並避免因 `trident-installer`使用者而導致的權限問題。
- 使用 `uninstall parameter <Uninstalling Trident>`進行失敗執行後的清理。預設情況下，該腳本不會刪除 Trident 所建立的 CRD，因此即使在執行中的部署中，也可以安全地解除安裝並重新安裝。
- 如果您想要降級到早期版本的 Trident，請先執行 `tridentctl uninstall`指令以移除 Trident。下載所需的 "Trident 版本"並使用 `tridentctl install`指令進行安裝。

- 安裝成功後，如果 PVC 卡在 `Pending`階段，執行 `kubectl describe pvc`可以提供有關 Trident 未能為此 PVC 配置 PV 的其他資訊。

使用操作員部署 **Trident** 失敗

如果您使用 Operator 部署 Trident，則 `TridentOrchestrator`的狀態會從 `Installing`變為 `Installed`。如果您觀察到 `Failed`狀態，且 Operator 無法自行恢復，則應執行以下命令檢查 Operator 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，其中一個問題可能是無法在隔離環境中從上游登錄檔拉取所需的容器映像。

要了解為什麼 Trident 安裝失敗，您應該查看 `TridentOrchestrator` 狀態。

```
kubectl describe torc trident-2
Name:         trident-2
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Image Pull Secrets:         <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:                      Trident is bound to another CR 'trident'
  Namespace:                    trident-2
  Status:                       Error
  Version:
Events:
  Type     Reason  Age                From                          Message
  ----     ------  ----               ----                          -------
  Warning  Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
is bound to another CR 'trident'
```

此錯誤表示已存在 `TridentOrchestrator`用於安裝 `Trident`。由於每個 `Kubernetes` 叢集只能有一個 `Trident` 執行個體，因此操作員會確保在任何給定時間都只存在一個可建立的作用中 `TridentOrchestrator`。

此外，觀察 Trident Pod 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映像未擷取，因此 pod 無法完全初始化。

若要解決此問題，您應該編輯 `TridentOrchestrator` CR。或者，您可以刪除 `TridentOrchestrator`，然後使用修改後的準確定義建立新的變更要求。

使用 **Trident** 部署失敗 `tridentctl`

為了幫助您找出出錯的原因，您可以使用 `-d` 參數再次運行安裝程序，這將啟用調試模式，並幫助您了解問題所在：

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

解決問題後，您可以按以下步驟清理安裝，然後再次執行 `tridentctl install` 命令：

```
./tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Removed Trident user from security context constraint.
INFO Trident uninstallation succeeded.
```

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。除非您想要全新安裝 Trident，否則請勿執行此操作。若要解除安裝 Trident 而不移除 CRD，請參閱 "解除安裝 Trident"。

Trident 操作程式

若要解除安裝 Trident 並使用 Trident 運算子徹底移除 CRD：

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

Helm

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>

若要在使用 `tridentctl` 解除安裝 Trident 後完全移除 CRD

```
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

如果您執行的是 Kubernetes 1.26，在使用 NVMe/TCP 搭配 RWX 原始區塊命名空間時，節點取消暫存可能會失敗。下列案例提供失敗的因應措施。或者，您也可以將 Kubernetes 升級至 1.27。

刪除了命名空間和 pod

假設您有一個 Trident 管理的命名空間（NVMe 持久卷）附加到 pod。如果您直接從 ONTAP 後端刪除該命名空間，則在嘗試刪除 pod 後，卸載程序會卡住。這種情況不會影響 Kubernetes 叢集或其他功能。

因應措施

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。

已封鎖的資料 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.
```

.因應措施

啟動 `dataLIF` 以恢復全部功能。

已刪除命名空間對應

If you remove the `hostNQN` of the worker node from the corresponding subsystem, the unstaging process gets stuck when you attempt to delete the pod. In this scenario, you cannot run any NVMe CLI commands on the Kubernetes node.
.因應措施
將 `hostNQN` 後端加入子系統。

升級 ONTAP 後，NFSv4.2 用戶端在預期啟用「v4.2-xattrs」時回報「invalid argument」

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。此問題發生於 SVM 上未啟用 `v4.2-xattrs`選項時。.因應措施：在 SVM 上啟用 `v4.2-xattrs`選項，或升級至 ONTAP 9.12.1 或更新版本，此選項在該版本中預設為啟用。

支援

NetApp 以多種方式提供 Trident 支援。全天候 24x7 提供豐富的免費自助支援選項，例如知識庫（KB）文章和 Discord 頻道。

Trident 支援生命週期

Trident 根據您的版本提供三個等級的支援。請參閱 "NetApp 軟體版本支援定義"。

完整支援
: Trident 自發布之日起提供十二個月的全面支援。

有限支援
: Trident 在發布日期後的第 13 至 24 個月提供有限的支援。

自助支援
: Trident 文件可在發布日期後的第 25 個月至第 36 個月內查閱。

版本	完整支援	有限支援	自助支援
"25.10"	2026 年 10 月	2027 年 10 月	2028 年 10 月
"25.06"	2026 年 6 月	2027 年 6 月	2028 年 6 月
"25.02"	2026 年 2 月	2027 年 2 月	2028 年 2 月
"24.10"	—	2026 年 10 月	2027 年 10 月
"24.06"	—	2026 年 6 月	2027 年 6 月

"24.02"	—	2026 年 2 月	2027 年 2 月
"23.10"	—	—	2026 年 10 月
"23.07"	—	—	2026 年 7 月
"23.04"	—	—	2026 年 4 月
"23.01"	—	—	2026 年 1 月

自助支援

如需完整的疑難排解文章清單、請參閱 "NetApp Knowledgebase （需要登入）"。

社群支援

在我們的 "Discord 頻道" 上有一個活躍的容器使用者公共社群（包括 Trident 開發人員）。這裡是提出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 技術支援

如需 Trident 的協助，請使用 `tridentctl logs -a -n trident`建立支援服務組合，並將其傳送至 `NetApp Support <Getting Help>`。

如需更多資訊

- "Trident 資源"
- "Kubernetes Hub"

參考

Trident 連接埠

深入瞭解 Trident 用於通訊的連接埠。

概況

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

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

控制器節點的連接埠

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

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

工作節點的連接埠

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

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

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

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

Trident REST API

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

何時使用 REST API

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

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

使用 REST API

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

API 的運作方式如下：

取得

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

列出該類型的所有物件。

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

取得指定物件的詳細資訊。

POST

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

建立指定類型的物件。

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

刪除

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

刪除指定的資源。

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

命令列選項

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

日誌記錄

-debug

啟用偵錯輸出。

-loglevel <level>

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

Kubernetes

-k8s_pod

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

-k8s_api_server <insecure-address:insecure-port>

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

Docker

-volume_driver <name>

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

-driver_port <port-number>

在此連接埠上進行監聽，而非 UNIX 網域通訊端。

-config <file>

必填項；您必須指定後端組態檔的路徑。

REST

-address <ip-or-host>

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

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

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

-port <port-number>

指定 Trident REST 伺服器監聽的連接埠。預設為 8000。

-rest

啟用 REST 介面。預設為 true。

Kubernetes 和 Trident 物件

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

這些物件之間是如何相互作用的？

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

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

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

Kubernetes PersistentVolumeClaim 對象

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

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

註解	**Volume** 選項	支援的驅動程式
trident.netapp.io/fileSystem	fileSystem	ontap-san，solidfire-san，ontap-san-economy
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ontap-nas、ontap-san、solidfire-san、azure-netapp-files、ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas、ontap-san
trident.netapp.io/protocol	通訊協定	任何
trident.netapp.io/exportPolicy	exportPolicy	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	snapshotPolicy	ontap-nas 、 ontap-nas-economy 、 ontap-nas-flexgroup 、 ontap-san
trident.netapp.io/snapshotReserve	snapshotReserve	ontap-nas、ontap-nas-flexgroup、ontap-san
trident.netapp.io/snapshotDirectory	snapshotDirectory	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/blockSize	blockSize	solidfire-san
trident.netapp.io/skipRecoveryQueue	skipRecoveryQueue	ontap-nas 、 ontap-nas-economy 、 ontap-nas-flexgroup 、 ontap-san 、 ontap-san-economy

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

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

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

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

區，而非從頭配置磁碟區。

請考慮以下幾點：

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

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

Kubernetes `PersistentVolume` 對象

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

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

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

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

Kubernetes `StorageClass` 對象

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

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

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

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters: <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

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

儲存類別參數如下：

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

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

儲存資源池選擇屬性

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

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

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

[1]：ONTAP Select 系統不支援

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

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

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

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

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

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

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

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

Kubernetes 屬性

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

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

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

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

Kubernetes VolumeSnapshotClass 對象

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

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

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

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

Kubernetes VolumeSnapshot 對象

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

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

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

Kubernetes VolumeSnapshotContent 物件

Kubernetes VolumeSnapshotContent 物件表示從已配置的磁碟區中取得的快照。它類似於 PersistentVolume，表示儲存叢集上已配置的快照。與 PersistentVolumeClaim 和 PersistentVolume 物件類似，在建立快照時， VolumeSnapshotContent 物件會維護與 VolumeSnapshot 物件的一對一對應關係，該物件曾要求建立快照。

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

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

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

Kubernetes `VolumeGroupSnapshotClass` 物件

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

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

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

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

Kubernetes `VolumeGroupSnapshot` 物件

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

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

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

Kubernetes `VolumeGroupSnapshotContent` 物件

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

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

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

Kubernetes `CustomResourceDefinition` 物件

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

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

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

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

關於 Trident 的 CRD，需要記住以下幾點：

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

Trident `StorageClass` 物件

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

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

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

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

Trident 後端物件

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

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

如需如何建構這些物件的詳細資訊，請參閱 "配置後端"。

Trident `StoragePool` 物件

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

與此處的其他物件不同、儲存資源池候選對象始終會自動探索和管理。

Trident `Volume` 物件

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

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

Volume 組態定義了已配置 Volume 應具有的內容。

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

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

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

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

Trident `Snapshot` 物件

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

每個 `Snapshot` 物件都包含以下屬性：

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

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

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

Trident `ResourceQuota` 物件

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

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

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

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

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

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

如果安裝失敗，請進行清理 `ResourceQuota`

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

如果這樣不行，請手動移除 `ResourceQuota` 物件。

移除 `ResourceQuota`

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

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

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

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

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

必要的 Kubernetes 安全性內容和相關欄位

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

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

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

Pod 安全標準（PSS）

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

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

Pod 安全性原則 (PSP)

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

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

安全內容限制 (SCC)

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

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

法律聲明

法律聲明提供版權聲明、商標、專利等資訊的存取權限。

版權

"https://www.netapp.com/company/legal/copyright/"

商標

NETAPP、NETAPP 標誌以及 NetApp 商標頁面上所列的標記均為 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

目前 NetApp 擁有的專利清單可在以下網址找到：

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

隱私權政策

"https://www.netapp.com/company/legal/privacy-policy/"

開放原始碼

您可以在每個版本的通知文件中查看 NetApp 軟體中用於 Trident 的第三方版權和授權 https://github.com/NetApp/trident/。