



配置和管理磁碟區 Trident

NetApp
October 30, 2026

目錄

配置和管理磁碟區	1
配置 Trident Volume	1
概況	1
建立 PVC	1
移動磁碟區	4
需求	4
運作方式	4
移動磁碟區	4
擴充 Trident Volume	7
擴充 iSCSI Volume	7
擴充 FC Volume	10
擴充 NFS Volume	14
了解 RWX NVMe 子系統限制	16
了解 64 節點限制	17
了解 NVMe 子系統模型	17
識別錯誤症狀	17
解決子系統限制錯誤	17
升級 Trident 以套用超子系統模型	17
控制器擴充性	18
關鍵概念和定義	18
預設行為（串列模式）	18
控制器擴充性支援	18
enableConcurrency 的運作方式	19
啟用之前	19
啟用控制器擴充性	19
並行行為	21
注意事項和限制	22
建議	22
自動磁碟區擴充	22
需求	23
限制	23
使用自動增長策略配置磁碟區	23
建立 Autogrow 策略	23
建立 Autogrow 策略	24
政策狀態	25
將政策與 StorageClass 建立關聯	25
政策優先順序	26
組態範例	26
管理 Autogrow 策略	29

檢視自動成長原則	29
更新 Autogrow 策略	29
刪除 Autogrow 策略	30
監控 Autogrow Policy 使用情況	31
支援的通訊協定	31
支援的後端	32
已知限制	32
常見問題	32
匯入 Trident Volume	37
概述和注意事項	37
匯入磁碟區	38
範例	40
ONTAP SAN-economy 範例	45
自訂 Trident Volume 名稱和標籤	48
開始之前	48
限制	49
可自訂磁碟區名稱的關鍵行為	49
後端組態範例、名稱範本和標籤	49
名稱範本範例	50
需要考慮的要點	51
跨命名空間共享 Trident NFS Volume	51
功能	51
快速入門	52
設定來源和目的地命名空間	52
刪除共享磁碟區	54
使用 `tridentctl get` 查詢從屬磁碟區	54
限制	54
如需更多資訊	55
跨命名空間複製 Trident Volume	55
先決條件	55
快速入門	55
設定來源和目的地命名空間	55
限制	57
使用 SnapMirror 複製磁碟區	57
複寫的前提條件	57
建立鏡射 PVC	58
Volume 複寫狀態	61
在非計劃性容錯移轉期間提升次要 PVC	61
在計劃故障切換期間推廣次要 PVC	61
故障轉移後還原鏡像關係	62
其他作業	62

ONTAP 上線時更新鏡像關係	63
ONTAP 離線時更新鏡像關係	63
使用 CSI 拓撲	63
概況	63
步驟 1：建立拓撲感知後端	65
步驟 2：定義可感知拓撲的 StorageClasses	67
步驟 3：建立並使用 PVC	67
更新後端以包含 supportedTopologies	70
尋找更多資訊	70
使用 Trident 快照	70
概況	70
建立 Volume Snapshot	70
從磁碟區快照建立 PVC	72
匯入磁碟區快照	73
使用快照恢復磁碟區資料	74
從快照進行就地 Volume 還原	75
刪除具有相關快照的 PV	77
部署 Volume Snapshot Controller	77
相關連結	78
使用 Trident Volume 群組快照	78
建立 Volume Group 快照	78
使用群組快照恢復磁碟區資料	80
從快照進行就地 Volume 還原	80
刪除具有相關群組快照的 PV	80
部署 Volume Snapshot Controller	80
相關連結	81

配置和管理磁碟區

配置 Trident Volume

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。

移動磁碟區

從 Trident 26.06 開始，儲存管理員可以在工作負載持續運行的情況下，將 iSCSI 磁碟區無中斷地移動至相同 ONTAP 叢集內的不同 ONTAP 節點和聚合。您可以透過 `TridentVolumeMove` 自訂資源或 `tridentctl` 等效方式以宣告式方式要求移動，Trident 會協調 ONTAP 和主機，確保 I/O 不會中斷。

需求

磁碟區移動功能適用於 iSCSI 磁碟區的 `ontap-san` 驅動程式。不支援 NVMe 和 NAS 協定。

在移動磁碟區之前，請確保符合下列需求：

- 後端使用 `ontap-san` 驅動程式搭配 ONTAP REST API。
- ONTAP 叢集至少有兩個節點。
- 您擁有以下其中一組憑證：
 - 如果使用專用自訂 ONTAP 角色，則需要 ONTAP 9.19.1 或更高版本。
 - 早期 ONTAP 版本中的叢集管理員憑證。

運作方式

Trident 會在同一 ONTAP 叢集內的兩個位置之間移動磁碟區（LUN）。在資料移動之前，Trident 會在目標位置新增新的 iSCSI 路徑，與現有路徑並存，以確保主機多路徑裝置始終保持存取。ONTAP 完成移動後，Trident 會刪除過時的來源路徑。

Trident 透過 `TridentVolumeMove` 資源中可見的一組有序狀態來驅動移動，以便您可以追蹤移動進度。

移動磁碟區

識別磁碟區和位置

1. 找到要移動的 PV 的名稱。PV 名稱也是移動資源的名稱。
2. 從 ONTAP 叢集中，記下該磁碟區目前的（來源）節點和 Aggregate，然後選擇目的地（目標）節點和 Aggregate。

開始移動

兩種方法都可以。兩種方法都會建立相同的 `TridentVolumeMove` 資源，資源名稱即為 PV 名稱。

使用 `tridentctl`

```
tridentctl create volume-move \  
  --volume      pvc-8a814d62-bd58-4253-b0d1-82f2885db671 \  
  --source-node ontap-node-01 \  
  --source-pool aggr1_node01 \  
  --target-node ontap-node-02 \  
  --target-pool aggr2_node02
```

```
--target-pool aggr1_node02 \  
-n trident
```

`tvm`是`volume-move`的別名。新增`--delete-after-success 10m`可在移動成功後自動刪除資源。

使用 **TridentVolumeMove** 資源

```
apiVersion: trident.netapp.io/v1  
kind: TridentVolumeMove  
metadata:  
  name: pvc-8a814d62-bd58-4253-b0d1-82f2885db671    # the PV to move  
  namespace: trident  
spec:  
  sourceNode: ontap-node-01  
  sourcePool: aggr1_node01  
  targetNode: ontap-node-02  
  targetPool: aggr1_node02  
  # deleteAfterSuccess: 10m    # optional
```

應用該資源：

```
kubectl apply -f tridentvolumemove.yaml
```

規格欄位

sourceNode / targetNode

ONTAP 節點名稱（不是 UUID）。

sourcePool / targetPool

ONTAP aggregate 名稱。

deleteAfterSuccess

可選持續時間，例如 30s 或 10m。設定為 0 可立即刪除資源，省略則可保留資源。

監控移動

使用 `tvm` 簡稱或 `tridentvolumemove` 追蹤進度：

```
kubectl get tridentvolumemove -n trident
```

若要查看目前狀態和每個節點的進度，請描述資源：

```
kubectl describe tridentvolumemove <pv-name> -n trident
```

在 `Moving` 狀態期間，磁碟區可以同時存在多個附件，因為 ONTAP 可能需要一些時間才能完成大型磁碟區的遷移。為了確保無縫遷移，Trident 會在 Trident 控制器和 Trident 節點之間進行協調。每個狀態依序執行以下步驟，最終以 `Succeeded` 或 `Failed` 結束。

狀態	怎麼了
Pending	Trident 會接受並驗證移動請求，並在開始任何儲存變更之前收集所需的資訊。
ControllerStaging	Trident 會準備目標位置。它會將目標節點的儲存介面對應到 LUN，並更新 Kubernetes，以便將來節點重新啟動時能夠重新連線至正確的位置。
NodeStaging	在每個掛載了該磁碟區的主機上，Trident 會將目標節點的新儲存路徑新增至現有路徑旁邊，以便 pod 的 I/O 可以在兩組路徑上繼續進行。
Moving	ONTAP 將磁碟區資料複製並切換到目標 Aggregate。這是耗時最長的階段，整個過程由 ONTAP 獨立完成，Trident 會輪詢直至完成。在此狀態下允許新的發布。
ControllerUnstaging	資料移動已完成。Trident 更新 Kubernetes，從所需狀態中刪除來源路徑，然後從 LUN 取消映射來源節點的儲存介面。
NodeUnstaging	在每台主機上，Trident 會移除過時的來源路徑。主機上只保留目標路徑，多路徑裝置可繼續不間斷地提供 I/O 服務。
Succeeded	磁碟區已完全移動至目標位置並上線。Trident 會保留 <code>TridentVolumeMove</code> 資源，或者如果您設定了 <code>deleteAfterSuccess</code> ，則會自動將其刪除。
Failed	不可恢復的錯誤導致遷移操作中止或無法啟動。Trident 重設磁碟區狀態以便恢復正常運作，並保留資源以及錯誤訊息。請查看錯誤訊息，修復根本原因，刪除該資源，然後再試一次。

此 `.status.attachments` 清單會報告每個附加節點在其 `state` 欄位中的進度：

依附狀態	意義
Pending	節點已識別為已連線；工作尚未開始。
Bridged	在現有路徑之外新增目標路徑。
Migrated	流量現在已透過新路徑提供服務。
Cleaned	從節點中移除過時的來源路徑。
Detached	節點在移動過程中已分離，並排除在進一步考慮之外，而 Trident 正在評估連接是否已準備好進行下一步。
Failed	此節點出現終端錯誤。請查看訊息。

清理移動資源

如果設定了 `deleteAfterSuccess`，Trident 會自動移除資源。否則，您可以手動刪除。在要求對同一磁碟區進行另一次移動之前，請確保該磁碟區不存在任何處於 `TridentVolumeMove` 狀態的資源，無論是 `Succeeded` 還是 `Failed` 狀態。



Trident 不會自動刪除失敗的移動。Trident 將 `Failed` 資源保留為稽核記錄。修復根本原因後，請在重試之前手動刪除該資源。

擴充 Trident Volume

Trident 為 Kubernetes 使用者提供了在建立磁碟區後擴充磁碟區的功能。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 附加至 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：

```

kubect1 get pvc san-pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound      pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO           ontap-san    11m
kubect1 get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY STATUS    CLAIM                                STORAGECLASS  REASON  AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi      RWO
Delete              Bound      default/san-pvc  ontap-san    12m
tridentctl get volumes -n trident
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE  | STORAGE CLASS |
+-----+-----+-----+-----+
|          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
```

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

```

了解 RWX NVMe 子系統限制

使用 NVMe 協定的 ReadWriteMany (RWX) 磁碟區的擴充性限制為每個磁碟區 64 個節點。以下內容包括這些限制、解釋了所涉及的 NVMe 子系統架構、並概述了所需的解決步驟。

了解 64 節點限制

如果您打算將 ReadWriteMany (RWX) 磁碟區與 NVMe 協定搭配使用，則單一 RWX NVMe 磁碟區在 Kubernetes 叢集中最多只能由 64 個節點掛載。

不要在超過 64 個節點上排程掛載相同 RWX NVMe PersistentVolumeClaim 的工作負載。

此限制僅適用於使用 NVMe 傳輸協定的 RWX Volume。

了解 NVMe 子系統模型

每個磁碟區子系統模型（**Trident 26.02** 之前的版本）

在 Trident 26.02 之前的版本中，RWX NVMe 磁碟區採用每個磁碟區子系統模型進行配置。每個 RWX NVMe 磁碟區都對應到 ONTAP 上專屬的 NVMe 子系統。

此模型雖然簡單，但可擴展性限制較低。在大型 Kubernetes 叢集中，子系統控制器限制會很快達到，因為每個 RWX 磁碟區都會佔用一個專用子系統。

超子系統模型（在 **Trident 26.02** 中引入）

從 Trident 26.02 開始，RWX NVMe 磁碟區使用共享的超級子系統模型。多個 RWX NVMe 磁碟區共用同一個 NVMe 子系統。

每個超級子系統最多支援 1024 個命名空間（磁碟區）。此模型顯著提高了 RWX 工作負載的可擴充性，並降低了達到 ONTAP 子系統限制的可能性。

每個 RWX NVMe 磁碟區最多支援 64 個節點。

識別錯誤症狀

如果大規模建立或附加 RWX NVMe Volume，可能會遇到類似以下的錯誤：

```
Maximum number of controllers reached. No more controllers can be created.
```

此錯誤表示已達到 ONTAP NVMe 子系統控制器限制。

解決子系統限制錯誤

若要突破每個磁碟區子系統的限制，並利用超級子系統模型，請升級至 Trident 26.02 或更新版本。

升級 Trident 以套用超子系統模型

若要將超子系統模型套用於 RWX NVMe 磁碟區：

1. 將 Trident 升級到 26.02 或更高版本。
2. 將所有使用 RWX NVMe 磁碟區的 Pod 縮減為零個複本。
3. 確認沒有工作負載正在使用 RWX NVMe 磁碟區。

4. 將 Pod 重新擴展。

此重新啟動順序可確保使用超級子系統模型附加 RWX NVMe 磁碟區。

- 此限制僅適用於使用 NVMe 傳輸協定的 RWX Volume。
- 每個 RWX NVMe 磁碟區的 64 節點限制適用。
- 其他存取模式和其他傳輸協定不受影響。

控制器擴充性

Trident 透過提高多個儲存驅動程式之間的並行性來增強控制器的可擴充性。啟用控制器可擴充性後，Trident 控制器會並行處理儲存作業，而非序列化處理，從而提高具有大量並行作業的 Kubernetes 環境中的處理量。

部署前，請確定哪些 Trident 驅動程式在正式發佈時支援控制器可擴充性，哪些驅動程式在 Trident 26.06 中以技術預覽版的形式提供。這有助於您做出明智的部署決策並管理風險。控制器可擴充性預設為停用狀態。

關鍵概念和定義

控制器擴充性

控制器可擴展性是指 Trident 控制器能夠並行處理多個儲存操作，而不是將它們串行化並置於單一鎖定之後。這些操作包括磁碟區的建立、刪除和調整大小；快照的建立和刪除；磁碟區的發布和取消發布；以及後端管理。

啟用控制器可擴充性後，對不同磁碟區和後端執行的操作將同時進行。這可以提高吞吐量，並縮短並發 PersistentVolumeClaim 和 VolumeSnapshot 操作數量較多環境下的端對端操作時間。

預設行為（串列模式）

預設情況下，Trident 控制器一次處理一個作業。每個建立、刪除、調整大小或快照請求完成後，Trident 才會開始處理下一個請求。串行模式是所有安裝的預設支援模式，無需任何組態設定。

對於大多數工作負載而言，串列模式已足夠。僅當串列處理下的操作量造成積壓時，才啟用控制器可擴展性。

控制器擴充性支援

Trident 支援不同成熟度等級的控制器可擴充性，以適應不同的儲存驅動程式。

正式發行

以下驅動程式在 Trident 26.06 正式版中支援控制器擴充性：

- `ontap-san`
- `ontap-nas`
- `ontap-nas-economy`
- `ontap-san-economy`

- google-cloud-netapp-volumes
- azure-netapp-files
- solidfire-san



The google-cloud-netapp-volumes 和 google-cloud-netapp-volumes-san 驅動程式不同。Trident 僅支援 google-cloud-netapp-volumes。請勿在後端配置或範例中使用 google-cloud-netapp-volumes-san。

技術預覽

以下驅動程式在 Trident 26.06 中以技術預覽功能的形式支援控制器可擴充性：

- asa-r2 (SAN 和 NVMe)

該驅動程式存在以下限制：

- 控制器並發功能僅用於評估和測試。
- 行為可能會因版本而異。
- NetApp 不建議在生產環境中使用。

enableConcurrency 的運作方式

當您將 enableConcurrency 設定為 true 時，Trident 會在控制器管理的所有後端套用並行處理。此設定會同時套用於所有後端。無法針對個別後端或個別驅動程式啟用此功能。

每個已配置的後端都必須使用正式版或技術預覽版清單中的驅動程式。如果任何後端使用了不支援的驅動程式，Trident 將無法啟動，且 Trident 不會新增使用不支援驅動程式的後端。

若要恢復啟動，請刪除或重新配置任何使用不支援的驅動程式的後端，或停用控制器可擴充性。

啟用之前



啟用控制器可擴充性之前，請確認所有已設定的後端都使用正式版或技術預覽版清單中的驅動程式。如果任何後端使用了不支援的驅動程式，啟用該功能後 Trident 將無法啟動。

請使用下表決定是否啟用控制器可擴充性。

如果您的環境	然後
僅使用受支援的驅動程式，且控制器作業存在積壓。	啟用控制器可擴充性。
使用任何不受支援的驅動程式	保持預設序列模式。請勿啟用控制器可擴充性。
處理低作業量且無積壓	保持預設序列模式。

啟用控制器擴充性

此 enableConcurrency 組態選項可控制控制器的可擴充性。您必須在 Trident 安裝期間或更新現有部署時明確啟用此選項。

Trident 操作程式部署

若要使用 Trident operator 啟用控制器擴充性，請在 TridentOrchestrator 自訂資源 (CR) 中將 enableConcurrency 設定為 true。

新安裝

建立或編輯 TridentOrchestrator CR 並將 enableConcurrency 設定為 true：

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

套用 CR：

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

現有安裝

對現有 TridentOrchestrator CR 進行修補，以啟用控制器可擴充性：

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

確認 Trident 是否套用了此設定：

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

Helm 部署

若要使用 Helm 啟用控制器可擴充性，請將 enableConcurrency 值設為 true。

新安裝

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

現有安裝

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

或者，在自訂的 `values.yaml` 檔案中，將 `enableConcurrency` 設定為 `true`：

```
# values.yaml
enableConcurrency: true
```

然後使用值檔案進行安裝或升級：

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace -f values.yaml
```

Tridentctl 部署

若要啟用控制器可擴充性與 `tridentctl`，請在安裝期間傳遞 `--enable-concurrency` 標誌。

新安裝

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

現有安裝

若要在現有 `tridentctl` 部署中啟用控制器可擴充性，請解除安裝 Trident 並使用下列標誌重新安裝：

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

確認控制器可擴充性已啟用

啟用控制器可擴充性後，請驗證 Trident 控制器是否以並發模式運作。檢查控制器 Pod 日誌：

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

輸出結果包含一個日誌項目，確認並行功能已啟用。

並行行為

啟用控制器可擴充性後，Trident 控制器將採用下列行為：

- Trident 將單一全域鎖替換為細粒度的個別資源鎖定。
- Trident 對修改相同資源的作業進行序列化，以維持資料一致性。
- 僅從資源讀取資料的操作與其他對該資源的讀取操作同時進行。
- Trident 將每個管理 LIF 的並行 ONTAP API 請求限制為 20 個，以防止後端儲存系統過載。
- 如果多個後端共用同一個管理 LIF，則它們共用這 20 次請求的限制。

注意事項和限制

以下幾點適用於 Trident 26.06 中的控制器可擴充性：

- 控制器可擴充性僅支援正式發行清單和技術預覽清單中的驅動程式。如需詳細資訊，請參閱 [啟用之前](#)。
- Trident 控制器在內部管理並行作業。此版本不提供使用者可設定的並行限制。
- 整體吞吐量取決於所使用的儲存驅動程式、後端回應速度和 Kubernetes API 伺服器效能。
- 高並發性會增加後端儲存系統的負載。
- 不同驅動程式的控制器可擴展性行為並不相同。
- 技術預覽版驅動程式在高負載下可能表現不穩定，且不同版本之間的行為可能會有所變化。
- 由於並行執行，偵錯並發操作可能更加複雜。指標和日誌可能會顯示交錯的操作輸出。

建議

啟用控制器可擴充性之前，請完成以下步驟：

1. 確認每個已設定的後端都使用正式版或技術預覽版清單中的驅動程式。
2. 在將變更應用到生產環境之前，先在非生產叢集中進行測試。
3. 套用變更後，請確認控制器可擴充性已啟用。

在使用控制器可擴展性時，請遵循以下一般建議：

- 對於需要高可擴展性的生產環境，請使用通用可用性驅動程式。
- 僅在非生產環境中評估技術預覽版驅動程式。
- 大規模運作時，監控後端和控制器的效能。
- 不要在自動化腳本中假定操作順序。

自動磁碟區擴充

自動磁碟區擴充功能可讓 Trident 配置的持續磁碟區能夠在已使用容量達到定義的閾值時自動擴展。此功能可降低營運開銷，並有助於防止因容量耗盡而導致的應用程式中斷。自動磁碟區擴充是使用 Autogrow Policies 實作的。Autogrow Policy 定義：

- 觸發擴充的使用率臨界值
- 磁碟區增長的量

- 磁碟區可達到的最大大小



當超過定義的使用率臨界值時，磁碟區大小會自動增加。磁碟區大小永遠不會自動減少。

需求

在配置自動磁碟區擴充之前、請確保滿足以下要求：

- Trident 26.02 或更新版本
- 基於角色的存取控制權限，用於建立 `TridentAutogrowPolicy` 自訂資源
- `StorageClasses` 配置 `allowVolumeExpansion: true`
- 支援的 ONTAP 通訊協定：
 - 網路檔案系統 (NFS)
 - 網際網路小型電腦系統介面 (iSCSI)
 - 非揮發性記憶體高速介面 (NVMe)
 - 光纖通道協定 (FCP)

限制

- ONTAP 9.16.1 之前的 ONTAP Non-Volatile Memory Express 原始區塊磁碟區不支援自動擴充。
- 對於儲存區域網路磁碟區，如果 `growthAmount` 小於或等於 50 mebibytes，Trident 會在調整大小之前自動將該值增加到 51 mebibytes，前提是產生的大小不超過 `maxSize`。
- 在棕地環境中，由於磁碟區發佈遷移行為，自動擴充功能可能無法對某些現有磁碟區生效。
- 當磁碟區達到 `maxSize` 時，就不會再發生擴充。
- 支援自動磁碟區擴充的通訊協定：
 - 網路檔案系統 (NFS)
 - 網際網路小型電腦系統介面 (iSCSI)
 - 非揮發性記憶體高速介面 (NVMe)
 - 光纖通道協定 (FCP)

使用自動增長策略配置磁碟區

Autogrow Policy 可以在兩個層級進行設定：

- 儲存類別層級：設定所有磁碟區的預設值（使用註釋）
- PVC 等級：覆蓋儲存類別預設值（使用註解）

建立 Autogrow 策略

自動增長策略可在磁碟區達到定義的容量臨界值時自動擴展磁碟區。

請確保您已具備：

- 已安裝 Trident 26.02 或更新版本
- 基於角色的存取控制權限，用於建立 TridentAutogrowPolicy 資源
- 了解工作負載成長需求

Autogrow Policy 定義了當磁碟區達到定義的容量臨界值時如何自動擴充。

您可以在工作流程中的任何階段建立 Autogrow 策略：

- 在創建 StorageClasses 和磁碟區之前
- StorageClasses 存在之後
- 配置磁碟區之後

這種靈活性使您能夠在不重新建立現有資源的情況下引入自動擴充。

自動增長策略規格

Autogrow Policies 是 Kubernetes 自訂資源，定義如下：

欄位	說明	格式	必填	範例	預設
姓名	唯一原則識別碼	字串	是的	production-db-policy	無
usedThresh hold	觸發擴充的容量百分比	百分比字 串	是的	"80%"	無
growthAm ount	達到臨界值時的增長量	百分比或 大小	否	"10%" 或 "5Gi"	"10%"
maxSize	最大磁碟區大小限制	Kubernete s 數量	否	"500Gi"	無限

建立 Autogrow 策略

步驟

1. 建立定義 Autogrow Policy 的 YAML 檔案：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. 將原則套用至您的叢集：

```
kubectl apply -f autogrow-policy.yaml
```

3. 確認策略已建立：

```
kubectl get tridentautogrowpolicy standard-autogrow
```

預期輸出

NAME	USED THRESHOLD	GROWTH AMOUNT	STATE
standard-autogrow	80%	10%	Success

政策狀態

建立原則後、Trident 會驗證規格並指派下列其中一種狀態：

狀態	說明	需要採取行動
成功	策略已驗證，可以投入使用。	無。
失敗	偵測到驗證錯誤。	檢閱並修正規格。
刪除	刪除作業正在進行中。	等待完成。

將政策與 **StorageClass** 建立關聯

您可以使用 `trident.netapp.io/autogrowPolicy` 註解將 Autogrow Policy 與 StorageClass 關聯起來。所有從該 StorageClass 配置的磁碟區都將繼承該策略。



StorageClass 必須具有 `allowVolumeExpansion: true`。

步驟

1. 建立或修改帶有 Autogrow Policy 註解的 StorageClass：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. 應用 StorageClass：

```
kubectl apply -f storageclass.yaml
```

3. 驗證註釋：

```
kubectl get storageclass ontap-gold -o  
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

預期輸出

```
production-db-policy
```

政策優先順序

當 StorageClass 和 PVC 上都設定了自動成長策略註解時，Trident 應用以下優先規則：

1. *PVC 註釋優先。*如果 PVC 設定 `trident.netapp.io/autogrowPolicy`，則始終使用該值。
2. **StorageClass** 註解僅適用於 **PVC** 沒有註解的情況。
3. 如果兩者都沒有註解、則不會套用 Autogrow Policy。

StorageClass 註解	PVC 註釋	有效行為
<code>trident.netapp.io/autogrowPolicy: standard-agp</code>	未設定	用途 <code>standard-agp</code> 。
<code>trident.netapp.io/autogrowPolicy: standard-agp</code>	<code>trident.netapp.io/autogrowPolicy: logs-policy</code>	使用 <code>logs-policy</code> (PVC 覆寫 StorageClass)。
<code>trident.netapp.io/autogrowPolicy: standard-agp</code>	<code>trident.netapp.io/autogrowPolicy: "none"</code>	無自動增長策略 (PVC 停用自動增長)。
未設定	<code>trident.netapp.io/autogrowPolicy: dev-policy</code>	用途 <code>dev-policy</code> 。
未設定	未設定	無自動成長政策。

組態範例

以下範例展示了不同使用情境下的常見 Autogrow Policy 配置。

生產資料庫的保守原則

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

具有固定增長增量的日誌儲存

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

包含預設設定的最小原則

當您省略 `growthAmount` 和 `maxSize` 時，Trident 使用預設的(`10%`成長、無限大小)：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

具有自訂 **maxSize** 和預設 **growthAmount** 的策略

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

無限制的積極成長 **maxSize**

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

包含小數百分比的原則

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```



支援小數百分比。若您指定超過三位小數，Trident 會將數值四捨五入至小數點後三位。

具有自動增長功能的 **NAS StorageClass**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

具有自動增長功能的 **SAN StorageClass**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
name: database-storage
annotations:
  trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

管理 Autogrow 策略

建立 Autogrow Policies 後，您可以檢視、更新和刪除它們。您還可以監控哪些磁碟區使用了特定的原則。

檢視自動成長原則

列出所有原則

使用 `kubectl` 列出叢集中的所有 Autogrow 原則：

```
kubectl get tridentautogrowpolicy
```

或者，使用 `tridentctl`：

```
tridentctl get autogrowpolicy
```

檢視原則詳細資料

若要檢視原則的完整規格和狀態：

```
kubectl describe tridentautogrowpolicy production-db-policy
```

若要以 YAML 格式檢視策略及其相關磁碟區：

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

更新 Autogrow 策略

您可以修改現有原則，以變更其臨界值、成長量或大小上限。變更會立即對所有使用該原則的磁碟區生效。



變更會影響所有使用該策略的磁碟區。如果可能，請先在非生產環境中測試變更。

步驟

1. 編輯原則：

```
kubectl edit tridentautogrowpolicy production-db-policy
```

2. 根據需要修改 `spec` 欄位：

```
spec:
  usedThreshold: "75%"      # Changed from 80%
  growthAmount: "20%"       # Changed from 10%
  maxSize: "1Ti"            # Changed from 500Gi
```

3. 儲存並退出。變更會立即生效。

更新考量事項

- ***立即生效：***所有使用此原則的磁碟區將在下次成長評估時採用新參數。
- ***無需重新啟動磁碟區：***變更將套用於下一次成長作業。
- **先測試：**盡可能在非生產環境中驗證變更。
- **溝通變更：**當您修改共享策略時，請通知團隊。

刪除 Autogrow 策略

自動增長原則使用終結器保護，以防止磁碟區在積極使用時意外刪除。

步驟

1. 刪除原則：

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. 如果磁碟區仍然使用該原則，則刪除作業將進入 `Deleting` 狀態。請檢查哪些磁碟區受到影響：

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. 從每個受影響的磁碟區中移除該原則。請選擇下列其中一個選項：

- **選項 A：**明確停用自動增長，方法是將註解設定為 "none"：

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

- 選項 **B**：完全移除註解：

```
kubectl annotate pvc <pvc-name> \
    trident.netapp.io/autogrowPolicy-
```

刪除行為

情境	行為
沒有磁碟區使用此原則	原則會立即刪除。
磁碟區使用此原則	策略進入 Deleting 狀態。終結器會阻止完成，直到所有磁碟區都被移除。
所有磁碟區均已從該原則中移除	最終處理程序被移除，該政策被刪除。

監控 Autogrow Policy 使用情況

使用原則檢查磁碟區

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

尋找磁碟區使用的原則

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

監控原則事件

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

支援的通訊協定

Autogrow 支援以下儲存傳輸協定：

- NFS
- iSCSI
- FCP
- NVMe



對於 SAN 磁碟區，如果設定的 `growthAmount` 為 50 MiB 或更小，Trident 會自動將調整大小作業的成長量增加到 51 MB，只要產生的大小不超過 `maxSize`。

支援的後端

Autogrow 支援 ONTAP 後端。Autogrow 也支援 Google Cloud NetApp Volumes (GCNV) 的 NAS 和 SAN 磁碟區。

已知限制

- *ONTAP NVMe 原始區塊磁碟區：*使用 ONTAP 9.16.1 之前版本建立的磁碟區不支援自動成長。
- *現有磁碟區（棕地部署）：*即使應用了有效的自動成長策略，自動成長功能也可能無法對現有磁碟區生效。此限制是由於磁碟區發布正在進行遷移所致。若要確認遷移已完成，請檢查 Trident 控制器日誌中的 "Migration completed" 訊息。

常見問題

Trident 何時評估閾值？

Trident 持續監控磁碟區使用量。當已使用容量超過 `usedThreshold` 時，Trident 會建立內部調整大小請求，並依據設定的 `growthAmount` 擴充磁碟區。

例如，此原則會在容量達到 80% 時觸發擴充，每次將磁碟區增加 10%，最多可達 500 GiB：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

磁碟區已配置完畢後，我還能套用原則嗎？

是的。您可以隨時建立 Autogrow Policy，並透過新增或更新 `trident.netapp.io/autogrowPolicy` 註解將其應用於現有的 PVC。您無需重新建立 PVC 或 StorageClass。

將原則套用至現有的 PVC：

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

將原則套用至現有的 StorageClass：

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
```

```
--overwrite
```

如果我在 **StorageClass** 和 **PVC** 上都設定了自動成長策略會發生什麼事？

PVC 註解始終優先。如果 PVC 具有 `trident.netapp.io/autogrowPolicy` 註解，Trident 將使用該值，而不管 StorageClass 中指定了什麼值。如需詳細資訊，請參閱 ["政策優先順序"](#)。

例如，假設有以下 StorageClass：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

而這個 PVC 覆寫了 StorageClass 原則：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident 使用 logs-policy 為 database-pvc，不是 standard-agp。

如何停用特定磁碟區的自動增長？

將 PVC 註釋設定為 "none"。此設定會覆蓋該磁碟區的任何 StorageClass 層級原則：

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

您可以確認自動增長功能已停用：

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

預期輸出

```
none
```

當磁碟區達到 **maxSize** 時會發生什麼事？

Trident 停止擴充磁碟區。即使使用量持續增加超過 `usedThreshold`，它也不會再為該磁碟區建立任何調整大小的請求。

例如，根據此策略，Trident 在容量達到 100 GiB 後將停止成長：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

若要允許無限增長，請省略 `maxSize` 或將其設為 0：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

我可以在不重新啟動磁碟區的情況下變更原則嗎？

是的。更新原則後，所有使用此原則的磁碟區都會在下次成長評估時採用新參數。無需重新啟動磁碟區。

若要就地更新原則：

```
kubectl edit tridentautogrowpolicy production-db-policy
```

根據需要修改欄位：

```
spec:
  usedThreshold: "75%"      # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

儲存並退出。驗證更新的原則：

```
kubectl get tridentautogrowpolicy production-db-policy
```

預期輸出

NAME	USED THRESHOLD	GROWTH AMOUNT	STATE
production-db-policy	75%	20%	Success

為什麼我的原則處於失敗狀態？

‘Failed’ 狀態表示原則規格包含驗證錯誤。執行下列命令以檢視錯誤詳細資料：

```
kubectl describe tridentautogrowpolicy <policy-name>
```

常見原因包括無效的 `usedThreshold`（必須為 1-99%）、`growthAmount` 超過 `maxSize` 或無效的 Kubernetes 數量格式。請更正規格並重新應用：

```
kubectl apply -f autogrow-policy.yaml
```

為什麼我無法刪除原則？

策略使用終結器保護。如果磁碟區仍然使用該策略，刪除操作將進入 ‘Deleting’ 狀態，並等待您將所有磁碟區從該策略中移除。

識別受影響的磁碟區：

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

然後從每個 PVC 中移除註解：

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
```

```
trident.netapp.io/autogrowPolicy="none" \
--overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
trident.netapp.io/autogrowPolicy-
```

移除所有磁碟區後，Trident 會釋放終結器並刪除政策。

自動增長功能支援哪些後端？

Autogrow 支援 ONTAP 後端。也支援 Google Cloud NetApp Volumes (GCNV) 的 NAS 和 SAN 磁碟區。

自動成長功能支援 NFS、iSCSI、FCP 和 NVMe 協定。NVMe 原始區塊磁碟區需要 ONTAP 9.16.1 或更新版本。

對於現有部署環境中的磁碟區，可能需要先完成磁碟區發布移轉，然後自動增長功能才能生效。若要驗證移轉狀態，請檢查 Trident 控制器日誌：

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

以下 StorageClass 範例展示了為 NAS 和 SAN 後端配置的自動成長功能：

NAS 後端

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN 後端

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
```

```
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN 磁碟區的最小成長量是多少？

對於 SAN 磁碟區，有效最小增長量為 51 MB。如果您配置的 `growthAmount` 為 50 MiB 或更少，Trident 會自動將調整大小作業的增長量增加到 51 MB。

例如，此原則設定了 `growthAmount` 為 `"40Mi"`，但 Trident 會對使用該原則的任何 SAN 磁碟區套用 51 MB 的成長：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"
```

為避免自動調整，請將 `growthAmount` 設定為大於 50 MiB 的值：

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"
```

匯入 Trident Volume

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

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。
 - 如果儲存類別的回收策略為 `delete`，則當 PV 被刪除時，Trident 會刪除儲存磁碟區。
- 預設情況下，Trident 會管理 PVC，並在後端重新命名 FlexVol volume 和 LUN 或 `qtree`。您可以傳遞 `--no-manage` 旗標來匯入非託管磁碟區，以及傳遞 `--no-rename` 旗標來保留磁碟區名稱。
 - `--no-manage` - 如果使用 `--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 將在匯入磁碟區時保留現有磁碟區名稱，並管理磁碟區的生命週期。此選項僅支援 `ontap-nas`、`ontap-san`（包括 ASA r2 系統）、`ontap-san-economy` 和 `ontap-nas-economy`（`qtree` 匯入）驅動程式。



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。



- 該 `ontap-nas-economy` 驅動程式在 `qtree` 層級管理儲存，因此本身不支援匯入 FlexVol volume。若要使用該 `ontap-nas-economy` 驅動程式匯入 `qtree`，請參閱[導入 qtrees](#)（技術預覽版）。當 Trident 匯入 `qtree` 時，也會將包含該 `qtree` 的父 FlexVol 匯入為儲存貯體。
- `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>
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STORAGE CLASS	STATE	MANAGED
file	pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	standard	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 `unmanaged_volume ontap_nas` 後端匯入：

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

PROTOCOL	NAME	BACKEND UUID	SIZE	STORAGE CLASS	STATE	MANAGED
file	pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	standard	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: <unmanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>

```

導入 qtrees



Qtree 導入是 Trident 26.06 中的技術預覽功能。技術預覽功能不支援在生產環境中使用。

從 Trident 26.06 開始，您可以使用 `ontap-nas-economy` 驅動程式將現有的 ONTAP qtree 匯入 Trident。請以 FlexVol volume 和 qtree 名稱（格式為 `flexvol/mtree`）以及包含該 qtree 的後端 UUID 來識別 qtree。[概述和注意事項] 中的一般匯入注意事項也適用於 qtree 匯入。

使用 tridentctl 導入 qtree

以 flexvol/mtree 格式將 mtree 指定為磁碟區名稱。以下範例會從後端 ontap_nas_economy 上的 FlexVol volume flexvol1 匯入 mtree mtree1：

```
tridentctl import volume ontap_nas_economy flexvol1/mtree1 -f <path-to-pvc-file>
```

若要匯入 mtree 而不重新命名，請新增 `--no-rename` 標誌。若要匯入 mtree 而不讓 Trident 管理其生命週期，請新增 `--no-manage` 標誌。

使用 PVC 註解匯入 mtree

建立包含以下註解的 PVC：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "flexvol1/mtree1"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

若要保留現有 FlexVol volume 和 mtree 名稱，請新增 `trident.netapp.io/importNoRename: "true"` 註解。若要將 mtree 以非託管方式匯入，請新增 `trident.netapp.io/notManaged: "true"` 註解。

命名和管理行為

託管導入（預設）

Trident 會將 mtree 重新命名為與 PVC 名稱相符的名稱，並將 FlexVol volume 重新命名為符合 `ontap-nas-economy` 命名規則的名稱。連接路徑保持不變。

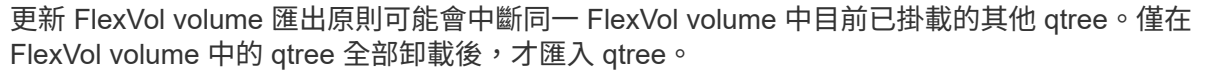
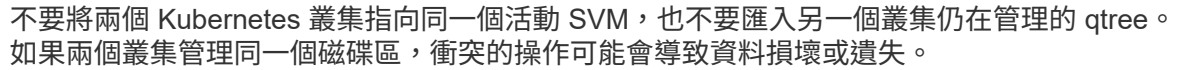
`--no-rename`

Trident 會保留現有的 FlexVol volume 和 mtree 名稱。這些名稱必須已遵循 Trident 的命名規則，否則 Trident 將無法管理 mtree 的刪除或掛載作業。

`--no-manage`

Trident 會引用 mtree，但不管理其生命週期。名稱必須遵循 Trident 的命名約定，且 Trident 不使用或管理 mtree 的自動匯出原則。

如果後端使用自動匯出原則，則匯入的 qtree 將採用基於 qtree 的匯出原則模型，其中每個 qtree 都有自己的匯出原則。Trident 在匯入時會為 qtree 指派一個空的匯出原則。發佈時，如果啟用了自動匯出且現有 FlexVol 匯出原則為 `default` 或空白，Trident 會將 FlexVol 磁碟區更新為基於後端的匯出原則。



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。

以下範例展示如何匯入託管和非託管磁碟區：

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。

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

```
tridentctl import volume -n trident san blog unmanaged example volume
```

```
-f pvc-import.yaml --no-manage
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS |
+-----+-----+-----+-----+
|          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

ONTAP SAN-economy 範例

以下範例展示如何為 ontap-san-economy 後端匯入託管磁碟區和非託管磁碟區。

託管磁碟區

當您匯入託管磁碟區時、Trident 會取得 FlexVol 的所有權並將其重新命名。當您從相同的 FlexVol 匯入多個 LUN 時、您必須考慮此重新命名。

以下範例將 `lun1` 從 FlexVol `toimport` 匯入為名為 `vol-managed-saneco` 的受管理磁碟區：

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

匯入後 lun1，Trident 會重新命名 FlexVol（例如、重新命名為 trident_lun_pool_xyz）。若要從相同的 FlexVol 匯入其他 LUN、請使用新的 FlexVol 名稱：

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```



ontap-san-economy 後端一次匯入一個 LUN。您可以使用指令碼自動執行多個匯入作業。

未託管磁碟區

匯入非託管磁碟區時、Trident 不會取得 FlexVol 的所有權。但是、FlexVol 和 LUN 必須遵循 Trident 命名慣例。

FlexVol 命名格式

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- STORAGEPREFIX 是您後端組態中 storagePrefix 的值。預設值為 trident。
- RANDOMSTRING 可以是您選擇的任何字串。

LUN 命名要求

LUN 必須命名為 lun0。

範例

如果您的 storagePrefix 是 `xyz`，則 LUN 的完整路徑為：

```
trident_lun_pool_xyz_randomstring/lun0
```

元素

Trident 支援使用 `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 `azurenetaappfiles_40517` 匯入一個 `azure-netapp-files` volume，並使用 volume 路徑 `importvol1`。

```
tridentctl import volume azurenetaappfiles_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
pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0	10 GiB	gcnv-nfs-sc-identity
8c18cdf1-0770-4bc0-bcc5-c6295fe6d837	online	true

自訂 Trident Volume 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.RequestName}}"
  },
  "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 Trident NFS Volume

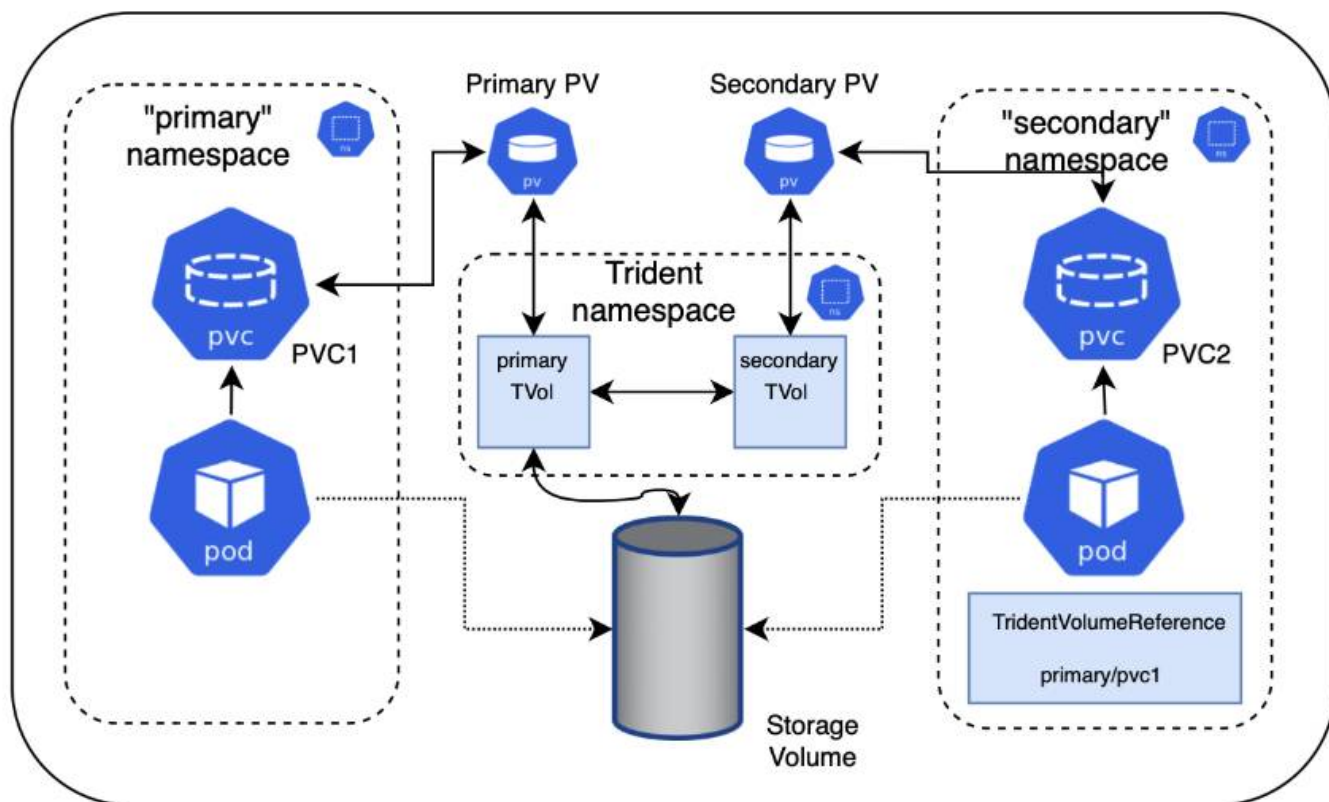
使用 Trident、您可以在主命名空間中建立磁碟區、並在一個或多個次要命名空間中共用該磁碟區。

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：

- 觀看 ["NetAppTV"](#) 上的示範影片。

跨命名空間複製 Trident Volume

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. *目的地命名空間擁有者：*在目的地命名空間中建立一個參照來源 PVC 的 TridentVolumeReference CR pvc1。

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. 目標命名空間擁有者：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。請參閱 ["SnapMirror 授權總覽 \(ONTAP\)"](#) 以了解更多資訊。

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。請參閱 ["叢集和 SVM 對等連接概述"](#)以了解更多資訊。



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
  replicationPolicy: MirrorAllSnapshots
  replicationSchedule: 5min
  volumeMappings:
    - localPVCName: csi-nas
      remoteVolumeHandle:
        "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

TMR spec 有四個頂層欄位：state，用於設定所需的關係狀態（請參閱 [Volume 複寫狀態](#)）；volumeMappings，用於將本機 PVC 與其遠端 Volume 控點建立關聯；以及下述選用的 replicationPolicy 和 replicationSchedule 欄位。

`replicationPolicy` 和 `replicationSchedule` 欄位為選填。請將 `replicationPolicy` 設定為 ONTAP 上的非同步 SnapMirror 原則名稱，並將 `replicationSchedule` 設定為 ONTAP 排程的名稱。如果省略任一欄位，Trident 將使用後端設定的對應複寫原則或排程。請在建立關係時設定這些欄位；關係建立後，您將無法變更原則或排程。若要使用不同的設定，請刪除並重新建立 TMR。

Trident 使用設定的複寫原則和排程建立並初始化 SnapMirror 關係。

1. 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. 在主要 Kubernetes 叢集上，建立 TridentTridentSnapshotInfoCR 以取得內部詳細資訊。

範例

```
kind: TridentSnapshotInfo
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  snapshotName csi-nas-snapshot
```

- a. 在輔助 Kubernetes 叢集上，將 *TridentMirrorRelationship* CR 的 *spec.state* 欄位更新為 *promoted*，並將 *spec.volumeMappings[].promotedSnapshotHandle* 更新為快照的 *internalName*。
3. 在輔助 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. 在次要 Kubernetes 叢集上，將 *TridentMirrorRelationship* 的 *spec.state* 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` 綁定模式不需要拓撲標籤。它可以獨立於 CSI 拓撲功能使用。

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:"1e11e4a2108024935ecfcb2912226cedaafd99df",
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:"1e11e4a2108024935ecfcb2912226cedaafd99df",
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","kubernetes.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","kubernetes.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","kubernetes.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

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-a
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-b
```

JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
```

```

    "topology.kubernetes.io/zone": "us-east1-a"
  },
  {
    "topology.kubernetes.io/region": "us-east1",
    "topology.kubernetes.io/zone": "us-east1-b"
  }
]
}

```



supportedTopologies 用於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撲標籤、StorageClasses 可定義為包含拓撲資訊。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。

請參閱以下範例：

```
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
  ----      -
  Normal    WaitForFirstConsumer 6s    persistentvolume-controller
waiting
for first consumer to be created before binding
  Message
```

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: voll
  persistentVolumeClaim:
    claimName: pvc-san
containers:
- name: sec-ctx-demo
  image: busybox
  command: [ "sh", "-c", "sleep 1h" ]
  volumeMounts:
  - name: voll
    mountPath: /data/demo
  securityContext:
    allowPrivilegeEscalation: false

```

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 Trident 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 `` 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.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. 建立 Snapshot 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 Trident Volume 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` 功能開設定為 `'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。如需詳細資訊，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_volumegroupsnapshotcontents.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"](#)

版權資訊

Copyright © 2026 NetApp, Inc. 版權所有。台灣印製。非經版權所有人事先書面同意，不得將本受版權保護文件的任何部分以任何形式或任何方法（圖形、電子或機械）重製，包括影印、錄影、錄音或儲存至電子檢索系統中。

由 NetApp 版權資料衍伸之軟體必須遵守下列授權和免責聲明：

此軟體以 NETAPP「原樣」提供，不含任何明示或暗示的擔保，包括但不限於有關適售性或特定目的適用性之擔保，特此聲明。於任何情況下，就任何已造成或基於任何理論上責任之直接性、間接性、附隨性、特殊性、懲罰性或衍生性損害（包括但不限於替代商品或服務之採購；使用、資料或利潤上的損失；或企業營運中斷），無論是在使用此軟體時以任何方式所產生的契約、嚴格責任或侵權行為（包括疏忽或其他）等方面，NetApp 概不負責，即使已被告知有前述損害存在之可能性亦然。

NetApp 保留隨時變本文所述之任何產品的權利，恕不另行通知。NetApp 不承擔因使用本文所述之產品而產生的責任或義務，除非明確經過 NetApp 書面同意。使用或購買此產品並不會在依據任何專利權、商標權或任何其他 NetApp 智慧財產權的情況下轉讓授權。

本手冊所述之產品受到一項（含）以上的美國專利、國外專利或申請中專利所保障。

有限權利說明：政府機關的使用、複製或公開揭露須受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中的「技術資料權利 - 非商業項目」條款 (b)(3) 小段所述之限制。

此處所含屬於商業產品和 / 或商業服務（如 FAR 2.101 所定義）的資料均為 NetApp, Inc. 所有。根據本協議提供的所有 NetApp 技術資料和電腦軟體皆屬於商業性質，並且完全由私人出資開發。美國政府對於該資料具有非專屬、非轉讓、非轉授權、全球性、有限且不可撤銷的使用權限，僅限於美國政府為傳輸此資料所訂合約所允許之範圍，並基於履行該合約之目的方可使用。除非本文另有規定，否則未經 NetApp Inc. 事前書面許可，不得逕行使用、揭露、重製、修改、履行或展示該資料。美國政府授予國防部之許可權利，僅適用於 DFARS 條款 252.227-7015(b)（2014 年 2 月）所述權利。

商標資訊

NETAPP、NETAPP 標誌及 <http://www.netapp.com/TM> 所列之標章均為 NetApp, Inc. 的商標。文中所涉及的所有其他公司或產品名稱，均為其各自所有者的商標，不得侵犯。