配置和管理卷
Trident

NetApp
January 15, 2026

目錄

配置和管理卷

提供一定量

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

概況

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

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

製作好PVC管後，就可以將管體安裝到管座中。

製作**PVC**管

步驟

1. 建立 PVC。

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

2. 核實PVC狀態。

```
kubectl get pvc
```

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

1. 將音量旋鈕安裝在一個音控器中。

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

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

2. 確認卷已掛載到 `/my/mount/path`。

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

3. 現在您可以刪除該 Pod 了。 Pod 應用將不復存在，但卷將保留。

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

樣品清單

PersistentVolumeClaim 樣本清單

這些範例展示了PVC的基本配置選項。

PVC管材，附RWO通道

此範例展示了一個具有 RWO 存取權限的基本 PVC，它與一個名為 StorageClass 的 StorageClass 相關聯。 `basic-csi` 。

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

PVC 和 NVMe/TCP

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

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

Pod清單範例

這些範例展示了將 PVC 連接到艙體的基本配置。

基本配置

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

基本 **NVMe/TCP** 配置

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

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

擴大銷量

Trident為 Kubernetes 使用者提供了在建立磁碟區後擴充磁碟區的能力。尋找有關擴展 iSCSI、NFS、SMB、NVMe/TCP 和 FC 磁碟區所需的配置的資訊。

擴充 iSCSI 卷

您可以使用 CSI 設定程式擴充 iSCSI 持久性磁碟區 (PV)。

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

步驟 1：設定 StorageClass 以支援磁碟區擴展

編輯 StorageClass 定義以進行設置 `allowVolumeExpansion`田野 `true`。

```
cat storageclass-ontapsan.yaml
```

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

對於已存在的 StorageClass，對其進行編輯以包含以下內容： `allowVolumeExpansion`範圍。

步驟 2：使用您建立的 StorageClass 建立 PVC。

編輯PVC定義並更新 `spec.resources.requests.storage`為了反映新的所需尺寸，該尺寸必須大於原始尺寸。

```
cat pvc-ontapsan.yaml
```

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

Trident創建一個持久銷售 (PV) 並將其與此持久銷售聲明 (PVC) 關聯起來。

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

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

步驟 **3**：定義一個連接 **PVC** 的艙體

將光電模組連接到艙體上，以便調整其尺寸。調整 iSCSI PV 大小時有兩種情況：

- 如果 PV 連接到 pod， Trident會擴展儲存後端上的捲，重新掃描設備，並調整檔案系統的大小。
- 嘗試調整未連接的 PV 的大小時， Trident會在儲存後端擴充磁碟區。 PVC 與 pod 綁定後， Trident會重新掃描裝置並調整檔案系統的大小。 Kubernetes 會在擴充操作成功完成後更新 PVC 大小。

在這個例子中，建立了一個使用下列功能的 pod： `san-pvc` 。

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

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

步驟 **4**：展開 **PV**

若要將已建立的 PV 從 1Gi 調整為 2Gi，請編輯 PVC 定義並更新。 `spec.resources.requests.storage`至 2Gi。

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

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

步驟 **5**：驗證擴展

您可以檢查PVC、PV和Trident的體積來驗證擴容是否正確：

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

擴大 FC 體積

您可以使用 CSI 設定程式擴充 FC 持久性磁碟區 (PV)。

FC體積擴張得到了以下的支持： `ontap-san`驅動程式需要 Kubernetes 1.16 及更高版本。

步驟 1：設定 StorageClass 以支援磁碟區擴展

編輯 StorageClass 定義以進行設置 `allowVolumeExpansion`田野 `true`。

```
cat storageclass-ontapsan.yaml
```

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

對於已存在的 StorageClass，對其進行編輯以包含以下內容： `allowVolumeExpansion`範圍。

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

編輯PVC定義並更新 `spec.resources.requests.storage`為了反映新的所需尺寸，該尺寸必須大於原始尺寸。

```
cat pvc-ontapsan.yaml
```

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

Trident創建一個持久銷售 (PV) 並將其與此持久銷售聲明 (PVC) 關聯起來。

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

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

步驟 **3**：定義一個連接 **PVC** 的艙體

將光電模組連接到艙體上，以便調整其尺寸。調整燃料電池光電模組容量時有兩種情況：

- 如果 PV 連接到 pod， Trident會擴展儲存後端上的捲，重新掃描設備，並調整檔案系統的大小。
- 嘗試調整未連接的 PV 的大小時， Trident會在儲存後端擴充磁碟區。 PVC 與 pod 綁定後， Trident會重新掃描裝置並調整檔案系統的大小。 Kubernetes 會在擴充操作成功完成後更新 PVC 大小。

在這個例子中，建立了一個使用下列功能的 pod： `san-pvc` 。

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

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

步驟 **4**：展開 **PV**

若要將已建立的 PV 從 1Gi 調整為 2Gi，請編輯 PVC 定義並更新。 `spec.resources.requests.storage`至 2Gi。

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

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

步驟 **5**：驗證擴展

您可以檢查PVC、PV和Trident的體積來驗證擴容是否正確：

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

擴展 NFS 卷

Trident支援為已設定的 NFS PV 擴充容量。 `ontap-nas` ， `ontap-nas-economy` ， `ontap-nas-flexgroup` ， `gcp-cvs` ，和 `azure-netapp-files`後端。

步驟 1：設定 StorageClass 以支援磁碟區擴展

要調整 NFS PV 的大小，管理員首先需要配置儲存類別以允許磁碟區擴展，方法是設定以下參數： `allowVolumeExpansion`田野 `true`：

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

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

如果您已經建立了一個沒有此選項的儲存類，則可以透過使用下列命令直接編輯現有儲存類別： `kubectl edit storageclass`允許體積膨脹。

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

```
cat pvc-ontapnas.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident應該為該 PVC 建立一個 20 MiB 的 NFS PV：

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

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

步驟 **3**：擴充 **PV**

若要將新建的 20 MiB PV 調整為 1 GiB，請編輯 PVC 並進行設置 `spec.resources.requests.storage`到 1 GiB：

```
kubectl edit pvc ontapnas20mb
```

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

步驟 **4**：驗證擴展

您可以檢查 PVC、PV 和Trident體積的大小來驗證調整大小是否正確：

```
kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

進口量

您可以使用以下方式將現有儲存磁碟區匯入為 Kubernetes PV： `tridentctl import` 。

概述和注意事項

您可以將磁碟區匯入Trident以執行下列操作：

- 將應用程式容器化並重複使用其現有資料集
- 為臨時應用程式使用資料集的克隆版本
- 重建失敗的 Kubernetes 集群
- 在災難復原期間遷移應用程式數據

注意事項

在匯入磁碟區之前，請考慮以下事項。

- Trident只能匯入 RW（讀寫）類型的ONTAP區。 DP（資料保護）類型磁碟區是SnapMirror目標磁碟區。在

將磁碟區導入Trident之前，應該先斷開鏡像關係。

- 我們建議匯入沒有活動連結的磁碟區。若要匯入正在使用的捲，請複製該卷，然後執行匯入操作。

這對於塊卷來說尤其重要，因為 Kubernetes 不知道先前的連接，很容易將活動卷附加到 pod 上。這可能導致資料損壞。

- 儘管 `StorageClass`必須在 PVC 上指定， Trident在導入過程中不使用此參數。在建立磁碟區時，會使用儲存類別根據儲存特性從可用儲存池中進行選擇。由於磁碟區已存在，因此匯入時無需選擇儲存池。因此，即使磁碟區存在於與 PVC 中指定的儲存類別不符的後端或池中，匯入也不會失敗。
- 現有容積尺寸在PVC中確定並設定。磁碟區被儲存驅動程式匯入後，PV 會創建，並帶有指向 PVC 的 ClaimRef。
  - 回收策略初始設定為 `retain`在PV中。 Kubernetes 成功綁定 PVC 和 PV 後，回收策略會更新為與儲存類別的回收策略相符。
  - 如果儲存類別的回收策略是 `delete`當 PV 被刪除時，儲存磁碟區也會被刪除。
- 預設情況下， Trident管理 PVC，並在後端重新命名FlexVol volume和 LUN。你可以透過 `--no-manage`導入非託管磁碟區的標誌。如果你使用 `--no-manage`在物件的生命週期內， Trident不會對 PVC 或 PV 執行任何額外的操作。刪除 PV 時，儲存磁碟區不會被刪除，其他操作（如磁碟區複製和磁碟區調整大小）也會被忽略。

如果您想使用 Kubernetes 來管理容器化工作負載，但又想在 Kubernetes 之外管理儲存磁碟區的生命週期，則此選項非常有用。

- 在 PVC 和 PV 中加入註釋，其作用有兩個：一是指示卷已導入，二是指示 PVC 和 PV 是否已管理。此註釋不應修改或刪除。

導入磁碟區

您可以使用 `tridentctl import`導入卷。

步驟

1. 建立持久性磁碟區聲明 (PVC) 檔案（例如， `pvc.yaml`) 將用於製造PVC。 PVC檔案應包含 `name` ， `namespace` ， `accessModes` ， 和 `storageClassName` 。 （可選）您可以指定 `unixPermissions`在你的PVC定義中。

   以下是一個最低規格範例：

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

不要包含其他參數，例如 PV 名稱或體積大小。這可能會導致導入命令失敗。

2. 使用 `tridentctl import`用於指定包含磁碟區的Trident後端名稱以及唯一標識儲存上磁碟區的名稱的命令（例如： ONTAP FlexVol、Element Volume、 Cloud Volumes Service路徑）。這 `-f`需要提供參數來指定PVC檔案的路徑。

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

範例

請查看以下卷導入範例，以了解支援的驅動程式。

ONTAP NAS 和ONTAP NAS FlexGroup

Trident支援使用以下方式導入磁碟區： `ontap-nas`和 `ontap-nas-flexgroup`司機。

- Trident不支援使用 `ontap-nas-economy` 司機。
- 這 `ontap-nas`和 `ontap-nas-flexgroup`驅動程式不允許重複的捲名稱。

每卷都是用以下方式創建的 `ontap-nas`driver 是ONTAP叢集上的FlexVol volume。使用以下方式導入FlexVol卷 `ontap-nas`驅動程式的工作原理相同。已存在於ONTAP叢集上的FlexVol磁碟區可以作為下列方式匯入： `ontap-nas` PVC。同樣， FlexGroup磁碟區也可以匯入為 `ontap-nas-flexgroup`PVC。

ONTAP NAS 範例

下面展示了託管磁碟區和非託管磁碟區匯入的範例。

託管磁碟區

以下範例導入一個名為“ `managed_volume`在名為 `ontap_nas`：

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
```

未託管卷

使用時 `--no-manage`理由是， Trident不會重新命名磁碟區。

以下範例導入 `unmanaged_volume`在 `ontap_nas`後端：

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

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

ONTAP SAN

Trident支援使用卷宗導入 `ontap-san` （iSCSI、NVMe/TCP 和 FC）和 `ontap-san-economy` 司機。

Trident可以匯入包含單一 LUN 的ONTAP SAN FlexVol磁碟區。這與 `ontap-san` 驅動程序，它為每個 PVC 建立一個FlexVol volume，並在FlexVol volume內建立一個 LUN。Trident導入FlexVol volume並將其與 PVC 定義關聯。Trident可以導入 `ontap-san-economy` 包含多個 LUN 的磁碟區。

ONTAP SAN 範例

下面展示了託管磁碟區和非託管磁碟區匯入的範例。

託管磁碟區

對於託管卷， Trident會將FlexVol volume重新命名為 `pvc-<uuid>`FlexVol volume中的 LUN 格式 `lun0`。

以下範例導入了 `ontap-san-managed`FlexVol volume存在於 `ontap_san_default`後端：

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

未託管卷

以下範例導入 `unmanaged_example_volume`在 `ontap_san`後端：

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

如果將 LUN 對應到與 Kubernetes 節點 IQN 共用相同 IQN 的 igroup，如下例所示，則會收到下列錯誤： `LUN already mapped to initiator(s) in this group`。您需要移除啟動器或取消映射 LUN 才能匯入磁碟區。

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

元素

Trident支援使用NetApp Element軟體和NetApp HCI卷導入 `solidfire-san`司機。

> Element驅動程式支援重複的磁碟區名稱。但是，如果磁碟區名稱重複， Trident會傳回錯誤。作為一種變通方法，克隆卷，提供一個唯一的磁碟區名稱，然後匯入克隆的磁碟區。

元素範例

以下範例導入一個 `element-managed`後端容量 `element_default`。

```
tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
```

谷歌雲端平台

Trident支援使用以下方式導入磁碟區： `gcp-cvs`司機。

> 若要將由NetApp Cloud Volumes Service支援的磁碟區匯入 Google Cloud Platform，請透過磁碟區路徑識別該磁碟區。磁碟區是磁碟區匯出路徑中位於下列位置之後的部分： `:/` 。例如，如果匯出路徑是 `10.0.0.1:/adroit-jolly-swift`體積路徑為 `adroit-jolly-swift`。

Google Cloud Platform 範例

以下範例導入一個 `gcp-cvs`後端容量 `gcpcvs_YEppr`具有體積路徑 `adroit-jolly-swift`。

```
tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-pvc-
file> -n trident

+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | gcp-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Azure NetApp Files

Trident支援使用以下方式導入磁碟區：`azure-netapp-files`司機。

> 若要匯入Azure NetApp Files，請透過磁碟區路徑識別該磁碟區。磁碟區是磁碟區匯出路徑中位於下列位置之後的部分：`:/`。例如，如果掛載路徑是`10.0.0.2:/importvol1`，**體積路徑**為`importvol1`。

Azure NetApp Files範例

以下範例導入一個`azure-netapp-files`**後端容量**`azurenetappfiles_40517`**體積路徑**`importvol1`。

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Google Cloud NetApp Volumes

Trident支援使用以下方式導入磁碟區：`google-cloud-netapp-volumes`司機。

Google Cloud NetApp Volumes範例

以下範例導入一個 `google-cloud-netapp-volumes`後端容量 `backend-tbc-gcnv1`音量 `testvoleasiaeast1`。

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

以下範例導入一個 `google-cloud-netapp-volumes`當兩個體積存在於同一區域時，體積為：

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

自訂磁碟區名稱和標籤

使用Trident，您可以為建立的磁碟區指派有意義的名稱和標籤。這可以幫助您識別磁碟區並將其輕鬆對應到各自的 Kubernetes 資源（PVC）。您也可以在後端層級定義模板，以建立自訂磁碟區名稱和自訂標籤；您建立、匯入或複製的任何磁碟區都會遵循這些模板。

開始之前

支援自訂磁碟區名稱和標籤：

1. 磁碟區建立、匯入和克隆操作。
2. 對於 ontap-nas-economy 驅動程序，只有 Qtree 磁碟區的名稱符合名稱範本。
3. 對於 ontap-san-economy 驅動程序，只有 LUN 名稱符合名稱範本。

限制

1. 可自訂磁碟區名稱僅與ONTAP本機驅動程式相容。
2. 可自訂的磁碟區名稱不適用於現有磁碟區。

可自訂磁碟區名稱的關鍵行為

1. 如果由於名稱範本中的語法無效而導致失敗，則後端建立將失敗。但是，如果範本套用失敗，則磁碟區將按照現有的命名約定命名。
2. 當使用後端配置中的名稱模板命名磁碟區時，儲存前綴不適用。可以直接在模板中加入任何所需的前綴值。

後端設定範例，包含名稱範本和標籤

可以在根層級和/或池層級定義自訂名稱範本。

根級別範例

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

池級範例

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

名稱範本範例

範例 **1**：

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

範例 **2**：

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

需要考慮的幾點

1. 對於磁碟區匯入，只有當現有磁碟區具有特定格式的標籤時，才會更新標籤。例如：`{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}` 。
2. 對於託管磁碟區匯入，磁碟區名稱遵循後端定義根層級定義的名稱範本。
3. Trident不支援將切片運算子與儲存前綴一起使用。
4. 如果模板無法產生唯一的磁碟區名稱， Trident將添加一些隨機字元來建立唯一的磁碟區名稱。
5. 如果 NAS 經濟型磁碟區的自訂名稱長度超過 64 個字符， Trident將依照現有的命名約定為磁碟區命名。對於所有其他ONTAP驅動程序，如果磁碟區名稱超過名稱限制，則磁碟區建立程序將會失敗。

跨命名空間分享 NFS 卷

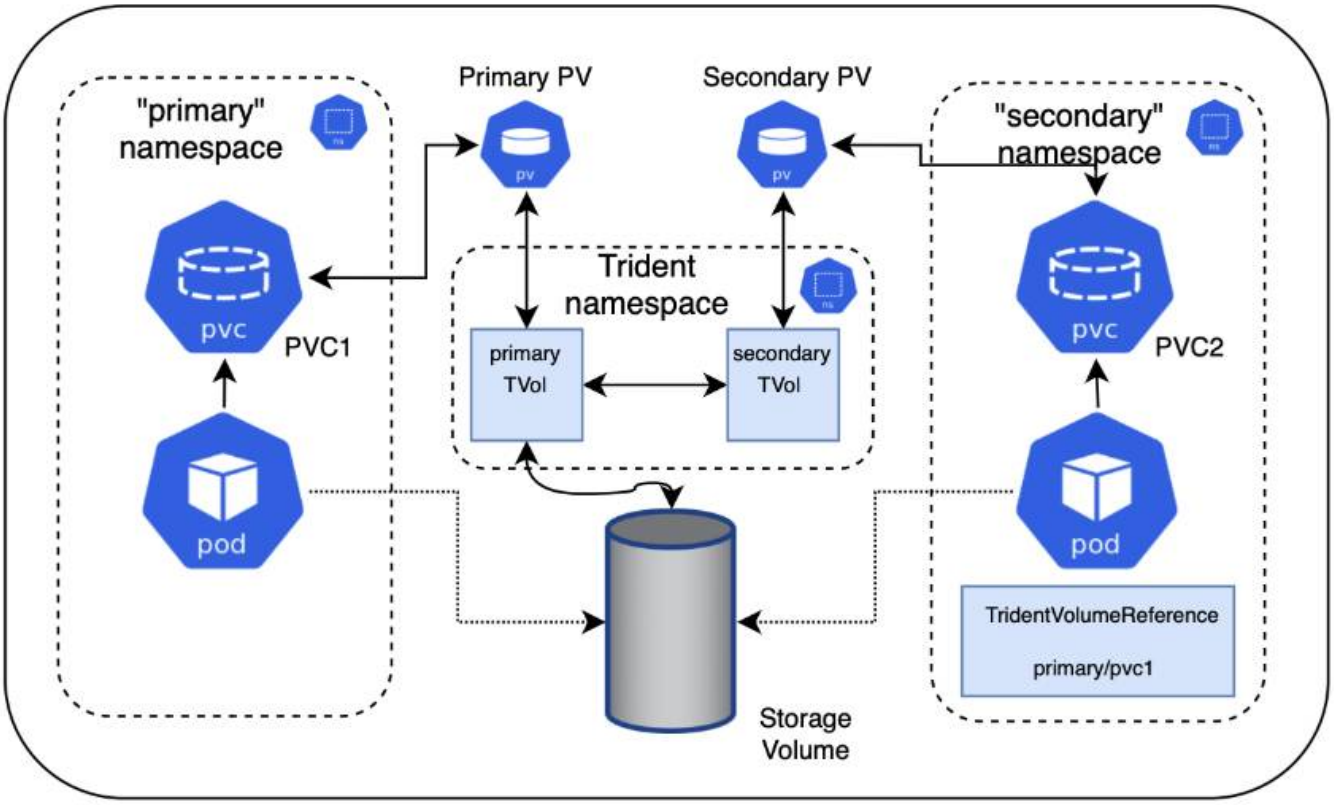
使用Trident，您可以在主命名空間中建立磁碟區，並在一個或多個輔助命名空間中共用該磁碟區。

特徵

TridentVolumeReference CR 可讓您在一個或多個 Kubernetes 命名空間之間安全地共用 ReadWriteMany (RWX) NFS 磁碟區。這種 Kubernetes 原生解決方案具有以下優點：

- 多層級存取控制以確保安全性
- 適用於所有Trident NFS 磁碟區驅動程式
- 不依賴 tridentctl 或任何其他非原生 Kubernetes 功能

此圖展示了跨兩個 Kubernetes 命名空間的 NFS 磁碟區共用。



快速啟動

只需幾個步驟即可設定NFS卷共享。

1 配置源**PVC**以共享體積

來源命名空間擁有者授予存取來源 PVC 中資料的權限。

2 授予在目標命名空間中建立 **CR** 的權限

叢集管理員授予目標命名空間的擁有者建立 TridentVolumeReference CR 的權限。

3 在目標命名空間中建立 **TridentVolumeReference**

目標命名空間的擁有者建立 TridentVolumeReference CR 以引用來源 PVC。

4 在目標命名空間中建立從屬 **PVC**

目標命名空間的擁有者建立從屬 PVC，以使用來源 PVC 中的資料來源。

配置來源命名空間和目標命名空間

為確保安全，跨命名空間共用需要來源命名空間擁有者、叢集管理員和目標命名空間擁有者的協作和行動。使用者角色在每個步驟中都會被指定。

步驟

1. 來源命名空間擁有者： 建立PVC(`pvc1`）在來源命名空間中，授予與目標命名空間共享的權限(`namespace2`) 使用 `shareToNamespace`註解。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident建立 PV 及其後端 NFS 儲存磁碟區。

- 您可以使用逗號分隔的清單將 PVC 共用給多個命名空間。例如，`trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4` 。
- 您可以使用以下方式共用到所有命名空間 `*`。例如，`trident.netapp.io/shareToNamespace: *`
- 您可以更新PVC以包含 `shareToNamespace`隨時可以添加註釋。

2. *叢集管理員：*確保已建立適當的 RBAC，以授予目標命名空間擁有者在目標命名空間中建立 TridentVolumeReference CR 的權限。
3. 目標命名空間擁有者： 在目標命名空間中建立一個指向來源命名空間的 TridentVolumeReference CR。`pvc1` 。

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

4. 目標命名空間擁有者： 建立PVC(`pvc2`）在目標命名空間(`namespace2`) 使用 `shareFromPVC`註記以指定來源 PVC。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

目標PVC管的尺寸必須小於或等於源PVC管的尺寸。

結果

Trident讀取 `shareFromPVC`在目標 PVC 上新增註釋，並將目標 PV 建立為沒有自身儲存資源的從屬卷，該磁碟區指向來源 PV 並共用來源 PV 儲存資源。目的地 PVC 和 PV 看起來已正常綁定。

刪除共享卷

您可以刪除跨多個命名空間共享的磁碟區。 Trident將移除對來源命名空間中該磁碟區的存取權限，並保留對共用該磁碟區的其他命名空間的存取權限。當所有引用該磁碟區的命名空間都被刪除後， Trident會刪除該磁碟區。

使用 `tridentctl get`查詢子卷

使用[`tridentctl`實用程序，您可以運行 `get`取得子卷的指令。更多信息，請參閱連結：../trident-reference/tridentctl.html[`tridentctl`命令和選項]。

```
Usage:
  tridentctl get [option]
```

標誌：

- `-h, --help`：有關卷的幫助。
- `--parentOfSubordinate string`：將查詢限制在子來源磁碟區上。
- `--subordinateOf string`：將查詢限制在磁碟區的下級。

限制

- Trident無法阻止目標命名空間寫入共享磁碟區。您應該使用文件鎖定或其他方法來防止覆蓋共享卷資料。
- 您無法透過移除來源PVC來撤銷對來源PVC的存取權限。 `shareToNamespace`或者 `shareFromNamespace`註釋或刪除 `TridentVolumeReference`CR。若要撤銷存取權限，必須刪除從屬PVC。
- 從屬磁碟區無法進行快照、複製和鏡像操作。

更多資訊

要了解有關跨命名空間卷訪問的更多資訊：

- 訪問"在命名空間之間共享卷：迎接跨命名空間卷訪問"。
- 觀看示範"NetAppTV"。

跨命名空間克隆卷

使用Trident，您可以利用同一 Kubernetes 叢集中不同命名空間內的現有磁碟區或磁碟區快照建立新磁碟區。

先決條件

在克隆卷之前，請確保來源後端和目標後端是同一類型，並且具有相同的儲存類別。

跨命名空間克隆僅支援以下情況： `ontap-san`和 `ontap-nas`儲存驅動程式。不支援只讀克隆。

快速啟動

只需幾個步驟即可設定卷克隆。

1 配置來源 **PVC** 以克隆卷

來源命名空間擁有者授予存取來源 PVC 中資料的權限。

2 授予在目標命名空間中建立 **CR** 的權限

叢集管理員授予目標命名空間的擁有者建立 TridentVolumeReference CR 的權限。

3 在目標命名空間中建立 **TridentVolumeReference**

目標命名空間的擁有者建立 TridentVolumeReference CR 以引用來源 PVC。

4 在目標命名空間中建立克隆 **PVC**

目標命名空間的擁有者建立 PVC 以複製來源命名空間中的 PVC。

配置來源命名空間和目標命名空間

為確保安全，跨命名空間複製磁碟區需要來源命名空間擁有者、叢集管理員和目標命名空間擁有者的協作和操作。使用者角色在每個步驟中都會被指定。

步驟

1. 來源命名空間擁有者： 建立PVC(`pvc1`）在來源命名空間中(`namespace1`)授予與目標命名空間共享的權限(`namespace2`)使用 `cloneToNamespace`註解。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident建立 PV 及其後端儲存磁碟區。

- 您可以使用逗號分隔的清單將 PVC 共用給多個命名空間。例如，`trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4` 。
- 您可以使用以下方式共用到所有命名空間 `*`。例如，`trident.netapp.io/cloneToNamespace: *`
- 您可以更新PVC以包含 `cloneToNamespace`隨時可以添加註釋。

2. 叢集管理員： 確保已配置正確的基於角色的存取控制 (RBAC)，以授予目標命名空間擁有者在目標命名空間中建立 TridentVolumeReference CR 的權限。(`namespace2`） 。
3. 目標命名空間擁有者： 在目標命名空間中建立一個指向來源命名空間的 TridentVolumeReference CR。`pvc1` 。

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

4. 目標命名空間擁有者： 建立PVC(`pvc2`）在目標命名空間(`namespace2`) 使用 `cloneFromPVC`或者 `cloneFromSnapshot`， 和 `cloneFromNamespace`用於指定來源PVC的註解。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

限制

- 對於使用 ontap-nas-economy 驅動程式配置的 PVC，不支援唯讀克隆。

使用SnapMirror複製卷

Trident支援一個叢集上的來源磁碟區與對等叢集上的目標磁碟區之間的鏡像關係，用於複製資料以實現災難復原。 您可以使用名為Trident鏡像關係 (TMR) 的命名空間自訂資源定義 (CRD) 來執行下列操作：

- 在體積（PVC）之間建立鏡像關係
- 移除磁碟區之間的鏡像關係
- 打破鏡像關係
- 在災難情況下（故障轉移）提升備用磁碟區的可用性

- 在計劃內故障轉移或遷移期間，實現應用程式在叢集間的無損遷移。

複製的前提條件

在開始之前，請確保滿足以下先決條件：

ONTAP叢集

- * Trident *：使用ONTAP作為後端的來源 Kubernetes 叢集和目標 Kubernetes 叢集上必須存在Trident版本 22.10 或更高版本。
- 許可證：使用資料保護包的ONTAP SnapMirror非同步許可證必須在來源 ONTAP 叢集和目標ONTAP叢集上啟用。請參閱 "ONTAP中的SnapMirror許可概述"了解更多。

  從ONTAP 9.10.1 開始，所有許可證均以NetApp許可證文件 (NLF) 的形式交付，這是一個可以啟用多種功能的單一文件。請參閱"ONTAP One 隨附的許可證"了解更多。

  僅支援SnapMirror非同步保護。

對等互連

- 叢集和 **SVM**： ONTAP儲存後端必須相互連接。請參閱 "叢集和SVM對等連接概述"了解更多。

  確保兩個ONTAP叢集之間複製關係中使用的 SVM 名稱是唯一的。

- * Trident和 SVM*：對等遠端 SVM 必須可供目標叢集上的Trident使用。

支援的驅動程式

NetApp Trident支援使用NetApp SnapMirror技術進行磁碟區複製，該技術使用以下驅動程式支援的儲存類別：**`ontap-nas`** ：**NFS** `ontap-san` ：iSCSI **`ontap-san`** ：**FC** `ontap-san` ：NVMe/TCP（最低要求ONTAP版本 9.15.1）

ASA r2 系統不支援使用SnapMirror進行磁碟區複製。有關ASA r2 系統的信息，請參閱"了解ASA r2 儲存系統"。

製作鏡面**PVC**

請依照這些步驟，並使用 CRD 範例，在主磁碟區和輔助磁碟區之間建立鏡像關係。

步驟

1. 在主 Kubernetes 叢集上執行下列步驟：
   a. 使用下列方式建立一個 StorageClass 對象 `trident.netapp.io/replication: true`範圍。

例子

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "nfs"
  trident.netapp.io/replication: "true"
```

b. 使用先前建立的 StorageClass 建立 PVC。

例子

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

c. 使用本機資訊建立鏡像關係變更要求。

例子

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident取得磁碟區的內部資訊和磁碟區的目前資料保護 (DP) 狀態，然後填入 MirrorRelationship 的狀態欄位。

d. 取得 TridentMirrorRelationship CR 以取得 PVC 的內部名稱和 SVM。

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. 在輔助 Kubernetes 叢集上執行下列步驟：
   a. 建立一個 StorageClass，並設定 trident.netapp.io/replication: true 參數。

      例子

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        trident.netapp.io/replication: true
      ```

   b. 建立包含目標和來源資訊的鏡像關係 CR。

例子

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Trident將使用配置的關係原則名稱（或ONTAP的預設值）建立SnapMirror關係並對其進行初始化。

c. 建立一個 PVC，使用先前建立的 StorageClass 作為輔助儲存類別（SnapMirror目標）。

例子

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Trident將檢查 TridentMirrorRelationship CRD，如果該關係不存在，則建立磁碟區失敗。如果存在此關係， Trident將確保將新的FlexVol volume放置在與 MirrorRelationship 中定義的遠端 SVM 對等的 SVM 上。

卷複製狀態

Trident鏡像關係 (TMR) 是一種 CRD，它表示 PVC 之間複製關係的一端。目標 TMR 具有一個狀態，該狀態告訴Trident期望的狀態是什麼。目的地 TMR 具有以下狀態：

- 已建立：本地 PVC 是鏡像關係的目標量，這是一個新的關係。
- 已推廣：本地 PVC 可讀寫且可安裝，目前沒有鏡像關係。
- 重新建立：本地 PVC 是鏡像關係的目標量，之前也處於該鏡像關係中。

- 如果目標磁碟區曾經與來源磁碟區有關聯，則必須使用重新建立的狀態，因為它會覆寫目標磁碟區的內容。
- 如果磁碟區之前未與來源建立關係，則重新建立狀態將會失敗。

在計劃外故障切換期間促進輔助PVC的運行

在輔助 Kubernetes 叢集上執行下列步驟：

- 將 TridentMirrorRelationship 的 *spec.state* 欄位更新為 `promoted`。

在計劃故障切換期間推廣備用PVC

在計劃故障轉移（遷移）期間，執行以下步驟以提升輔助 PVC：

步驟

1. 在主 Kubernetes 叢集上，建立 PVC 的快照，並等待快照建立完成。
2. 在主 Kubernetes 叢集上，建立 SnapshotInfo CR 以取得內部詳細資訊。

   例子

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

3. 在輔助 Kubernetes 叢集上，將 *TridentMirrorRelationship* CR 的 *spec.state* 欄位更新為 *promoted*，並將 *spec.promotedSnapshotHandle* 更新為快照的內部名稱。
4. 在輔助 Kubernetes 叢集上，確認 TridentMirrorRelationship 的狀態（status.state 欄位）是否已提升。

故障轉移後恢復鏡像關係

在恢復鏡像關係之前，選擇你想作為新主要方的那一方。

步驟

1. 在輔助 Kubernetes 叢集上，請確保 TridentMirrorRelationship 上的 *spec.remoteVolumeHandle* 欄位的值已更新。
2. 在輔助 Kubernetes 叢集上，將 TridentMirrorRelationship 的 *spec.mirror* 欄位更新為 `reestablished`。

附加手術

Trident支援對主磁碟區和輔助磁碟區執行下列操作：

將主**PVC**複製到新的輔助**PVC**

請確保您已備有主PVC管及備用PVC管。

步驟

1. 從已建立的輔助（目標）叢集中刪除 PersistentVolumeClaim 和 TridentMirrorRelationship CRD。
2. 從主（來源）叢集中刪除 TridentMirrorRelationship CRD。
3. 在主（來源）叢集上為要建立的新輔助（目標）PVC 建立一個新的 TridentMirrorRelationship CRD。

調整鏡像、主或次級**PVC**的尺寸

PVC 可以像往常一樣調整大小，如果資料量超過目前大小， ONTAP將自動擴展任何目標 flevxols。

從 **PVC** 移除複製

若要移除複製，請對目前輔助磁碟區執行下列其中一項操作：

- 刪除輔助 PVC 上的鏡像關係。這會破壞複製關係。
- 或者，將 spec.state 欄位更新為 *promoted*。

刪除一個**PVC**（之前已鏡像）

Trident會檢查是否有複製的 PVC，並在嘗試刪除磁碟區之前釋放複製關係。

刪除 **TMR**

刪除鏡像關係一側的 TMR 會導致剩餘的 TMR 在Trident完成刪除之前轉換為 *promoted* 狀態。如果選擇刪除的 TMR 已處於 *promoted* 狀態，則不存在鏡像關係，TMR 將被刪除， Trident會將本機 PVC 提升為 *ReadWrite*。此刪除操作會釋放ONTAP中本機磁碟區的SnapMirror元資料。如果將來要將此磁碟區用於鏡像關係，則在建立新的鏡像關係時，必須使用具有 *established* 磁碟區複製狀態的新 TMR。

ONTAP在線時，更新鏡像關係

鏡像關係建立後可以隨時更新。您可以使用 `state: promoted`或者 `state: reestablished`用於更新關係的欄位。將目標磁碟區提升為常規讀寫磁碟區時，可以使用 *promotedSnapshotHandle* 指定要將目前磁碟區還原到的特定快照。

ONTAP離線時更新鏡像關係

您可以使用 CRD 執行SnapMirror更新，而無需Trident與ONTAP叢集直接連線。請參考以下 TridentActionMirrorUpdate 的範例格式：

例子

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state`反映 TridentActionMirrorUpdate CRD 的狀況。它可以取值 成功、進行中_或_失敗。

使用 CSI 拓撲

Trident可以利用以下方式選擇性地建立磁碟區並將其附加到 Kubernetes 叢集中的節點："CSI拓樸功能" 。

概況

使用 CSI 拓撲功能，可以根據區域和可用區域將對磁碟區的存取限製到節點子集。如今，雲端服務供應商允許 Kubernetes 管理員建立基於區域的節點。節點可以位於一個區域內的不同可用區，也可以跨越多個區域。為了方便在多區域架構中為工作負載配置卷， Trident使用了 CSI 拓撲。

了解更多關於 CSI 拓撲功能的信息 "這裡"。

Kubernetes 提供了兩種獨特的磁碟區綁定模式：

- 和 `VolumeBindingMode`設定為 `Immediate`Trident創建卷時沒有任何拓樸感知。磁碟區綁定和動態配置在建立 PVC 時處理。這是預設值。 `VolumeBindingMode`適用於不強制執行拓樸約束的叢集。持久性卷的建立不依賴請求 pod 的調度要求。
- 和 `VolumeBindingMode`設定為 `WaitForFirstConsumer`PVC 的持久性磁碟區的建立和綁定將被延遲，直到使用該 PVC 的 pod 被調度和建立。這樣，就可以建立磁碟區來滿足拓樸要求所強制執行的調度約束。

這 `WaitForFirstConsumer`綁定模式不需要拓樸標籤。這可以獨立於 CSI 拓樸功能使用。

你需要什麼

要使用 CSI 拓撲結構，您需要以下元件：

- 運行中的 Kubernetes 集群"支援的 Kubernetes 版本"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- 叢集中的節點應附有標籤，以體現拓樸感知能力。(`topology.kubernetes.io/region`和`topology.kubernetes.io/zone`）。在安裝Trident之前，叢集中的節點上*應該存在這些標籤*，以便Trident能夠感知拓撲結構。

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

步驟 1：建立拓樸感知後端

Trident儲存後端可設計為根據可用區選擇性地配置磁碟區。每個後端都可以攜帶一個可選組件`supportedTopologies`表示所支援的區域和地區列表的區塊。對於使用此類後端的 StorageClasses，只有在受支援的區域/區域中調度的應用程式請求時才會建立磁碟區。

以下是一個後端定義範例：

YAML

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

JSON

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

`supportedTopologies`用於提供每個後端區域和分區的清單。這些區域和分區代表了 StorageClass 中可以提供的允許值的清單。對於包含後端提供的區域和可用區子集的儲存類，Trident會在後端建立一個磁碟區。

你可以定義 `supportedTopologies`每個儲存池也是如此。請參閱以下範例：

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

在這個例子中， `region`和 `zone`標籤代表儲存池的位置。 `topology.kubernetes.io/region`和 `topology.kubernetes.io/zone`決定儲存池可以從何處使用。

步驟 2：定義拓樸感知的儲存類別。

根據提供給叢集中節點的拓樸標籤，可以定義 StorageClasses 來包含拓樸資訊。這將決定哪些儲存池可作為 PVC 請求的候選對象，以及哪些節點子集可以使用Trident提供的磁碟區。

請參閱以下範例：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata: null
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions: null
  - key: topology.kubernetes.io/zone
    values:
      - us-east1-a
      - us-east1-b
  - key: topology.kubernetes.io/region
    values:
      - us-east1
parameters:
  fsType: ext4
```

在上述 StorageClass 定義中， `volumeBindingMode`設定為 `WaitForFirstConsumer`。使用此 StorageClass 請求的 PVC 只有在 pod 中被引用後才會執行。和， `allowedTopologies`提供要使用的區域和範圍。這 `netapp-san-us-east1`StorageClass 在儲存體上建立 PVC。 `san-backend-us-east1`後端定義如上所述。

步驟 3：製作並使用 PVC

建立 StorageClass 並將其對應到後端後，現在可以建立 PVC。

請參閱範例 `spec`以下：

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

使用此清單建立 PVC 將產生以下結果：

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

為了讓Trident形成體積並將其與 PVC 結合，請使用 PVC 管。請參閱以下範例：

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

此 podSpec 指示 Kubernetes 將 pod 調度到存在於下列位置的節點上：`us-east1`在該區域中，選擇任何存在的節點。`us-east1-a`或者 `us-east1-b`區域。

請查看以下輸出：

```
kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

更新後端以包含 supportedTopologies

可以更新現有的後端，使其包含以下清單： `supportedTopologies`使用 `tridentctl backend update`。這不會影響已經分配的容量，只會用於後續的PVC。

查找更多信息

- "管理容器資源"
- "節點選擇器"
- "親和力和反親和力"
- "污點和容忍度"

使用快照

Kubernetes 持久性磁碟區 (PV) 的磁碟區快照可以建立磁碟區的某個時間點副本。您可以建立使用Trident建立的磁碟區的快照，匯入在Trident之外建立的快照，從現有快照建立新磁碟區，以及從快照還原磁碟區資料。

概況

卷快照受支援 `ontap-nas`， `ontap-nas-flexgroup` ， `ontap-san` ， `ontap-san-economy` ， `solidfire-san` ， `gcp-cvs` ， `azure-netapp-files` ， 和 `google-cloud-netapp-volumes`司機。

開始之前

若要使用快照，您必須擁有外部快照控制器和自訂資源定義 (CRD)。這是 Kubernetes 編排器（例如：Kubeadm、GKE、OpenShift）的職責。

如果您的 Kubernetes 發行版不包含快照控制器和 CRD，請參閱[部署磁碟區快照控制器]。

> 如果在 GKE 環境中建立按需磁碟區快照，則不要建立快照控制器。 GKE 使用內建的隱藏快照控制器。

建立磁碟區快照

步驟

1. 創建一個 `VolumeSnapshotClass`更多信息，請參閱...["卷快照類"] 。
   - 這 `driver`指向Trident CSI 驅動程式。
   - `deletionPolicy`可以 `Delete`或者 `Retain`。設定為 `Retain`即使發生以下情況，儲存叢集上的底層實體快照仍會被保留： `VolumeSnapshot`物件已被刪除。

     例子

     ```
     cat snap-sc.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshotClass
     metadata:
       name: csi-snapclass
     driver: csi.trident.netapp.io
     deletionPolicy: Delete
     ```

2. 建立現有PVC的快照。

   範例
   - 此範例建立現有 PVC 的快照。

     ```
     cat snap.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshot
     metadata:
       name: pvc1-snap
     spec:
       volumeSnapshotClassName: csi-snapclass
       source:
         persistentVolumeClaimName: pvc1
     ```

   - 此範例為名為“ `pvc1`快照名稱設定為 `pvc1-snap`。 VolumeSnapshot 類似於 PVC，並且與某個 PVC 相關聯。 `VolumeSnapshotContent`代表實際快照的對象。

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                    AGE
pvc1-snap               50s
```

- 你可以識別出 `VolumeSnapshotContent`對象為 `pvc1-snap`透過描述來取得磁碟區快照。這 `Snapshot Content Name`識別提供此快照的 VolumeSnapshotContent 物件。這 `Ready To Use`此參數表示快照可用於建立新的 PVC。

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
...
```

從磁碟區快照建立PVC

您可以使用 `dataSource`使用名為 VolumeSnapshot 的 VolumeSnapshot 建立 PVC `<pvc-name>`作為數據來源。 PVC管製作完成後，可以將其連接到艙體上，並像其他PVC管一樣使用。

PVC 將在與來源磁碟區相同的後端建立。參考"知識庫：無法在備用後端從Trident PVC 快照建立 PVC。"。

以下範例使用以下方法建立 PVC： `pvc1-snap`作為資料來源。

```
cat pvc-from-snap.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

匯入磁碟區快照

Trident支持"Kubernetes 預先設定快照流程"使集群管理員能夠創建 `VolumeSnapshotContent`在Trident之外建立的物件和匯入快照。

開始之前

Trident肯定建立或匯入了快照的父卷。

步驟

1. 集群管理員： 建立一個 `VolumeSnapshotContent`引用後端快照的物件。這將啟動Trident中的快照工作流程。
   - 指定後端快照的名稱 `annotations`作為 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`。
   - 指定 `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>`在 `snapshotHandle`這是外部快照程式提供給Trident的唯一資訊。 `ListSnapshots`稱呼。

     > ⓘ 這 `<volumeSnapshotContentName>`由於 CR 命名限制，有時無法與後端快照名稱完全匹配。

     例子

     以下範例建立了一個 `VolumeSnapshotContent`引用後端快照的對象 `snap-01`。

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

2. 集群管理員： 建立 `VolumeSnapshot`引用 CR `VolumeSnapshotContent`目的。這是對使用權限的請求 `VolumeSnapshot`在給定的命名空間中。

   例子

   以下範例建立了一個 `VolumeSnapshot`CR命名 `import-snap`指的是 `VolumeSnapshotContent`命名 `import-snap-content`。

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content
```

3. 內部處理（無需操作）： 外部快照程式識別新建立的內容 `VolumeSnapshotContent`並運行 `ListSnapshots`稱呼。`Trident`創造了 `TridentSnapshot`。
   - 外部快照程式設定 `VolumeSnapshotContent`到 `readyToUse`以及 `VolumeSnapshot`到 `true`。
   - Trident回歸 `readyToUse=true`。
4. 任何使用者： 建立一個 `PersistentVolumeClaim`參考新的 `VolumeSnapshot`，其中 `spec.dataSource`（或者 `spec.dataSourceRef`名稱是 `VolumeSnapshot`姓名。

   例子

以下範例建立一個引用 PVC 的 `VolumeSnapshot`命名 `import-snap`。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

使用快照恢復磁碟區數據

預設情況下，快照目錄是隱藏的，以確保使用以下方式配置的磁碟區的最大相容性： `ontap-nas`和 `ontap-nas-economy`司機。啟用 `.snapshot`直接從快照還原資料的目錄。

使用磁碟區快照還原ONTAP CLI 將磁碟區還原到先前快照中記錄的狀態。

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

還原快照副本時，現有磁碟區配置將被覆寫。建立快照副本後對磁碟區資料所做的變更將會遺失。

從快照進行原地體積恢復

Trident利用快照提供快速、原位體積恢復功能 `TridentActionSnapshotRestore`（TASR）CR。此 CR 作為強制性 Kubernetes 操作，在操作完成後不會持久保存。

Trident支援快照恢復 `ontap-san`， `ontap-san-economy` ， `ontap-nas` ， `ontap-nas-flexgroup` ， `azure-netapp-files` ， `gcp-cvs` ， `google-cloud-netapp-volumes` ， 和 `solidfire-san`司機。

開始之前

您必須擁有已綁定的PVC和可用的磁碟區快照。

- 確認PVC狀態是否已綁定。

```
kubectl get pvc
```

- 確認磁碟區快照已準備就緒。

```
kubectl get vs
```

步驟

1. 建立 TASR CR。此範例為PVC建立CR。 `pvc1`和磁碟區快照 `pvc1-snapshot`。

   TASR CR 必須位於 PVC 和 VS 存在的命名空間。

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. 應用 CR 從快照恢復。此範例從快照恢復 `pvc1`。

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

結果

Trident從快照中復原資料。您可以驗證快照復原狀態：

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

- 大多數情況下， Trident不會在操作失敗後自動重試。您需要再次執行此操作。
- 沒有管理員權限的 Kubernetes 使用者可能需要管理員授予權限才能在其應用程式命名空間中建立 TASR CR。

刪除包含關聯快照的 PV

刪除具有關聯快照的持久性磁碟區時，對應的Trident磁碟區將更新為「正在刪除」狀態。刪除磁碟區快照以刪除Trident磁碟區。

部署磁碟區快照控制器

如果您的 Kubernetes 發行版不包含快照控制器和 CRD，您可以如下部署它們。

步驟

1. 建立磁碟區快照 CRD。

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. 建立快照控制器。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

如有必要，打開 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`並更新 `namespace`新增到您的命名空間。

相關連結

- "卷快照"
- "卷快照類"

使用磁碟區組快照

NetApp Trident提供了建立多個磁碟區（一組磁碟區快照）快照的功能，可用於建立持久磁碟區 (PV) 的 Kubernetes 磁碟區組快照。此磁碟區組快照表示在同一時間點從多個磁碟區中取得的副本。

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

建立卷宗組快照

卷冊組快照受支援 `ontap-san`此驅動程式僅適用於 iSCSI 協議，尚不支援光纖通道 (FCP) 或 NVMe/TCP。開始之前

- 請確保您的 Kubernetes 版本為 K8s 1.32 或更高版本。
- 若要使用快照，您必須擁有外部快照控制器和自訂資源定義 (CRD)。這是 Kubernetes 編排器（例如：Kubeadm、GKE、OpenShift）的職責。

  如果您的 Kubernetes 發行版不包含外部快照控制器和 CRD，請參閱[部署磁碟區快照控制器]。

  > 如果在 GKE 環境中建立按需卷宗組快照，則不要建立快照控制器。 GKE 使用內建的隱藏快照控制器。

- 在快照控制器 YAML 中，設定 `CSIVolumeGroupSnapshot`將功能門設為“true”，以確保啟用磁碟區組快照。
- 在建立磁碟區組快照之前，請先建立所需的磁碟區組快照類別。
- 確保所有 PVC/磁碟區都在同一 SVM 上，以便能夠建立 VolumeGroupSnapshot。

步驟

- 在創建 VolumeGroupSnapshot 之前，請先建立 VolumeGroupSnapshotClass。更多信息，請參閱"卷冊組快照類"。

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- 使用現有的儲存類別建立具有所需標籤的 PVC，或將這些標籤新增至現有的 PVC。

  以下範例使用以下方法建立 PVC： `pvc1-group-snap`作為資料來源和標籤 `consistentGroupSnapshot: groupA`。根據您的需求定義標籤的鍵和值。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- 建立具有相同標籤的磁碟區組快照(`consistentGroupSnapshot: groupA`）在PVC中規定。

  此範例建立磁碟區組快照：

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

使用群組快照恢復磁碟區數據

您可以使用作為磁碟區組快照一部分建立的各個快照來還原各個持久性磁碟區。您無法將磁碟區組快照作為一個整體還原。

使用磁碟區快照還原ONTAP CLI 將磁碟區還原到先前快照中記錄的狀態。

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

還原快照副本時，現有磁碟區配置將被覆寫。建立快照副本後對磁碟區資料所做的變更將會遺失。

從快照進行原地體積恢復

Trident利用快照提供快速、原位體積恢復功能 `TridentActionSnapshotRestore`（TASR）CR。此 CR 作為強制性 Kubernetes 操作，在操作完成後不會持久保存。

有關詳細信息，請參閱 "從快照進行原地體積恢復"。

刪除包含關聯群組快照的 PV

刪除群組磁碟區快照時：

- 您可以刪除整個 VolumeGroupSnapshots，而不是刪除群組中的單一快照。
- 如果在持久卷存在快照的情況下刪除該持久卷， Trident會將該卷移至「正在刪除」狀態，因為必須先刪除快照才能安全地刪除該卷。
- 如果使用分組快照建立了克隆，然後要刪除該群組，則會開始在克隆時進行分割操作，並且在拆分完成之前無法刪除該群組。

部署磁碟區快照控制器

如果您的 Kubernetes 發行版不包含快照控制器和 CRD，您可以如下部署它們。

步驟

1. 建立磁碟區快照 CRD。

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. 建立快照控制器。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

如有必要，打開 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`並更新 `namespace`新增到您的命名空間。

相關連結

- "卷冊組快照類"
- "卷快照"