資源配置與管理磁碟區

Astra Trident

NetApp
March 05, 2026

目錄

資源配置與管理磁碟區 1
配置 Volume 1
總覽 1
建立 PV 和 PVC 4
展開Volume 5
展開iSCSI Volume 5
展開NFS Volume 9
匯入磁碟區 12
總覽與考量 12
匯入 Volume 13
範例 15
自訂磁碟區名稱和標籤 21
開始之前 22
限制 22
可自訂 Volume 名稱的主要行為 22
名稱範本和標籤的後端組態範例 22
名稱範本範例 23
需要考量的重點 24
跨命名空間共用NFS磁碟區 24
功能 24
快速入門 25
設定來源和目的地命名空間 25
刪除共享Volume 27
用於 `tridentctl get`查詢從屬磁碟區 27
限制 27
以取得更多資訊 28
使用 SnapMirror 複寫磁碟區 28
複寫先決條件 28
建立鏡射 PVC 29
Volume 複寫狀態 32
在非計畫性容錯移轉期間升級次要 PVC 32
在規劃的容錯移轉期間升級次要 PVC 32
在容錯移轉後還原鏡射關係 33
其他作業 33
當 ONTAP 連線時、請更新鏡射關係 34
當 ONTAP 離線時更新鏡射關係 34
啟用 Astra Control Provisioner 34
使用「csi拓撲」 43
總覽 43

步驟1：建立可感知拓撲的後端 45
步驟2：定義可感知拓撲的StorageClass 47
步驟3：建立並使用PVC 48
更新後端以納入 `supportedTopologies` 51
如需詳細資訊、請參閱 51
使用快照 51
總覽 51
建立磁碟區快照 52
從磁碟區快照建立 PVC 53
匯入 Volume 快照 54
使用快照恢復 Volume 資料 56
從快照進行原位磁碟區還原 56
刪除含有相關快照的 PV 58
部署 Volume Snapshot 控制器 58
相關連結 59

資源配置與管理磁碟區

配置 Volume

建立 PersistentVolume （ PV ）和 PersistentVolume Claim （ PVC ）、使用設定的 Kubernetes StorageClass 來要求存取 PV 。然後、您可以將 PV 掛載至 Pod 。

總覽

"*PersistentVolumer*"（ PV ）是叢集管理員在 Kubernetes 叢集上配置的實體儲存資源。 "_PersistentVolume Claim"（ PVC ）是存取叢集上 PersistentVolume 的要求。

可將 PVC 設定為要求儲存特定大小或存取模式。叢集管理員可以使用相關的 StorageClass 來控制超過 PersistentVolume 大小和存取模式的權限、例如效能或服務層級。

建立 PV 和 PVC 之後、您可以將磁碟區裝入 Pod 。

範例資訊清單

PersistentVolume 範例資訊清單

此範例資訊清單顯示與 StorageClass 相關的 10Gi 基本 PV `basic-csi` 。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/my/host/path"
```

PersistentVolume Claim 範例資訊清單

這些範例顯示基本的 PVC 組態選項。

可存取 **RWO** 的 **PVC**

此範例顯示具有 rwo 存取權的基本 PVC 、與名稱為的 StorageClass 相關聯 `basic-csi`。

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

採用 **NVMe / TCP** 的 **PVC**

此範例顯示 NVMe / TCP 的基本 PVC 、並提供與名稱為的 StorageClass 相關聯的 rwo 存取 `protection-gold`。

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

Pod 資訊清單範例

這些範例顯示將 PVC 連接至 Pod 的基本組態。

基本組態

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

基本 **NVMe / TCP** 組態

```
---
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
    - image: nginx
      name: nginx
      resources: {}
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
      claimName: pvc-san-nvme
```

建立 **PV** 和 **PVC**

步驟

1. 建立 PV 。

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

2. 確認 PV 狀態。

```
kubectl get pv
NAME        CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM
STORAGECLASS  REASON  AGE
pv-storage  4Gi       RWO           Retain          Available
7s
```

3. 建立 PVC 。

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

4. 確認 PVC 狀態。

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

5. 將磁碟區裝入 Pod 。

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

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

6. 驗證是否已將磁碟區掛載到上 `/my/mount/path`。

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

7. 您現在可以刪除 Pod 。Pod 應用程式將不再存在、但該磁碟區仍會保留。

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

如需儲存類別如何與互動的詳細資訊 `PersistentVolumeClaim`、以及控制 Astra Trident 配置磁碟區的參數、請參閱"Kubernetes和Trident物件"。

展開Volume

Astra Trident可讓Kubernetes使用者在建立磁碟區之後擴充磁碟區。尋找擴充iSCSI和NFS磁碟區所需組態的相關資訊。

展開iSCSI Volume

您可以使用「SCSI資源配置程式」來擴充iSCSI持續磁碟區（PV）。

> iSCSI Volume 擴充受、、 `solidfire-san`驅動程式支援 `ontap-san`、`ontap-san-economy`需要 Kubernetes 1.16 及更新版本。

步驟**1**：設定**StorageClass**以支援**Volume**擴充

編輯 StorageClass 定義以將欄位設定 `allowVolumeExpansion`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 、請編輯以加入 `allowVolumeExpansion`參數。

步驟**2**：使用您建立的**StorageClass**建立一個永久虛擬儲存設備

編輯 PVC 定義並更新、 `spec.resources.requests.storage`以反映新的所需尺寸、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
```

Astra 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**：定義一個連接至**PVc**的**Pod**

將 PV 附加至 Pod 、以便調整大小。調整iSCSI PV的大小有兩種情況：

- 如果PV附加至Pod、Astra Trident會在儲存後端擴充磁碟區、重新掃描裝置、並重新調整檔案系統的大小。
- 嘗試調整未附加PV的大小時、Astra Trident會在儲存後端上擴充磁碟區。在將永久虛擬磁碟綁定至Pod之後、Trident會重新掃描裝置並重新調整檔案系統的大小。然後、Kubernetes會在擴充作業成功完成後、更新PVc大小。

在此範例中，會建立使用的 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**

若要調整從 1Gi 建立到 2Gi 的 PV 大小、請編輯 PVC 定義並將更新 `spec.resources.requests.storage`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Astra Trident Volume的大小、以正確驗證擴充作業：

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

展開NFS Volume

Astra Trident 支援在、、 `ontap-nas-economy`、 `ontap-nas-flexgroup` `gcp-cvs`和 `azure-netapp-files`後端上佈建的 `NFS PV` 的 `Volume` 擴充 `ontap-nas`。

步驟1：設定StorageClass以支援Volume擴充

若要調整 NFS PV 的大小、管理員必須先將欄位設定為 `true`、以設定儲存類別以允許擴充磁碟區 `allowVolumeExpansion`：

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

如果您已經建立了沒有此選項的儲存類別、只要使用編輯現有的儲存類別、即可進行 `kubectl edit storageclass` Volume 擴充。

步驟**2**：使用您建立的**StorageClass**建立一個永久虛擬儲存設備

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

Astra Trident應為此PVC建立20MiB NFS PV：

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

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

步驟 **3** ：展開 **PV**

若要將新建立的 20MiB PV 調整為 1GiB 、請編輯 PVC 並設定 `spec.resources.requests.storage`為 1GiB ：

```
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Astra Trident Volume的大小、以正確驗證調整大小：

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

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

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

匯入磁碟區

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

總覽與考量

您可以將磁碟區匯入 Astra Trident 、以便：

- 將應用程式容器化、並重新使用其現有的資料集
- 針對臨時應用程式使用資料集的複本
- 重建故障的 Kubernetes 叢集
- 在災難恢復期間移轉應用程式資料

考量

匯入 Volume 之前、請先檢閱下列考量事項。

- Astra Trident 只能匯入 RW （讀寫）類型的 ONTAP Volume 。DP （資料保護）類型磁碟區是 SnapMirror

目的地磁碟區。您應該先中斷鏡射關係、再將 Volume 匯入 Astra Trident 。

- 我們建議您在沒有作用中連線的情況下匯入磁碟區。若要匯入使用中的 Volume 、請複製該 Volume 、然後執行匯入。

這對區塊磁碟區特別重要、因為 Kubernetes 不會知道先前的連線、而且很容易將作用中的磁碟區附加到 Pod 。這可能導致資料毀損。

- 雖然 `StorageClass`必須在 PVC 上指定、 Astra Trident 在匯入期間不會使用此參數。建立磁碟區時會使用儲存類別、根據儲存特性從可用的集區中選取。由於該磁碟區已經存在、因此在匯入期間不需要選取任何集區。因此、即使磁碟區存在於與 PVC 中指定的儲存類別不相符的後端或集區、匯入也不會失敗。
- 現有的 Volume 大小是在 PVC 中決定和設定的。儲存驅動程式匯入磁碟區之後、PV會以PVc的ClaimRef建立。
  - 回收原則一開始會在 PV 中設定為 `retain`。Kubernetes成功繫結了PVc和PV之後、系統會更新回收原則以符合儲存類別的回收原則。
  - 如果儲存類別的回收原則為、則 `delete`刪除 PV 時、儲存磁碟區將會刪除。
- 根據預設、 Astra Trident 會管理 PVC 、並重新命名後端上的 FlexVol 和 LUN 。您可以傳遞 `--no-manage`旗標來匯入未受管理的磁碟區。如果您使用 `--no-manage`、 Astra Trident 在物件生命週期內不會在 PVC 或 PV 上執行任何其他作業。刪除 PV 時不會刪除儲存磁碟區、也會忽略其他操作、例如 Volume Clone 和 Volume resize 。

如果您想要將Kubernetes用於容器化工作負載、但想要管理Kubernetes以外儲存磁碟區的生命週期、則此選項非常實用。

- 將註釋新增至PVc和PV、這有兩種用途、表示已匯入磁碟區、以及是否管理了PVc和PV。不應修改或移除此附註。

匯入 **Volume**

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

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

使用 **tridentctl**

步驟

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

   以下是最低規格的範例：

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

   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`命令指定 Astra Trident 後端的名稱、該後端包含磁碟區、以及唯一識別儲存區中磁碟區的名稱（例如： ONTAP FlexVol 、 Element Volume 、 Cloud Volumes Service 路徑）。 `-f`需要引數來指定 PVC 檔案的路徑。

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

使用 **PVC** 註釋

步驟

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

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

       以下是匯入託管磁碟區的範例規格：

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

2. 將 PVC YAML 檔案套用到您的 Kubernetes 叢集：

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

Trident 會自動匯入磁碟區並將其繫結至 PVC。

範例

請參閱下列 Volume 匯入範例、瞭解支援的驅動程式。

ONTAP NAS 和 ONTAP NAS FlexGroup

Astra Trident 支援使用和 `ontap-nas-flexgroup`驅動程式進行 `Volume` 匯入 `ontap-nas`。

- `ontap-nas-economy`驅動程式無法匯入及管理 qtree 。
- `ontap-nas`和 `ontap-nas-flexgroup`驅動程式不允許重複的磁碟區名稱。

使用驅動程式建立的每個磁碟區都 `ontap-nas`是 `ONTAP` 叢集上的 `FlexVol` 。使用驅動程式匯入 `FlexVols` 的 `ontap-nas`運作方式相同。`ONTAP` 叢集上已存在的 `FlexVol` 可以匯入為 `ontap-nas` PVC 。同樣地、 FlexGroup Vols 也可以匯入為 `ontap-nas-flexgroup` PVCS 。

使用 **tridentctl** 的 **ONTAP NAS** 範例

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

託管 Volume

以下範例會匯入在名為的後端上 `ontap_nas`命名的 `Volume `managed_volume`：

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

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

非託管 Volume

使用 `--no-manage`引數時、 Astra Trident 不會重新命名 Volume 。

下列範例會匯入 `unmanaged_volume` `ontap_nas`後端：

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

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

使用 PVC 註解的 ONTAP NAS 範例

以下範例展示如何使用 PVC 註解匯入託管和非託管磁碟區。

託管 Volume

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

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

非託管 Volume

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

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

SAN ONTAP

Astra Trident 支援使用驅動程式進行 Volume 匯入 `ontap-san`。磁碟區匯入不支援使用 `ontap-san-economy` 驅動程式。

Astra Trident 可以匯入包含單一 LUN 的 ONTAP SAN FlexVols 。這與驅動程式一致 `ontap-san`、可為 FlexVol 中的每個 PVC 和 LUN 建立 FlexVol 。Astra Trident 會匯入 FlexVol 、並將其與 PVC 定義相關聯。

ONTAP SAN 範例

以下範例展示如何匯入託管和非託管磁碟區：

託管 Volume

對於託管卷， Astra Trident 將 FlexVol 重命名爲格式，並將 FlexVol 中的 LUN `lun0` 重命名爲 `pvc-<uuid>`。

下列範例會匯入 `ontap-san-managed`**後端上的** `FlexVol `ontap_san_default`：

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

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

非託管 Volume

下列範例會匯入 `unmanaged_example_volume` `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 LUN 對應至與 Kubernetes 節點 IQN 共用 IQN 的 igroup 、則會收到錯誤訊息： `LUN already mapped to initiator(s) in this group`。您需要移除啟動器或取消對應 LUN 、才能匯入磁碟區。

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

元素

Astra Trident 支援使用驅動程式的 NetApp Element 軟體和 NetApp HCI Volume 匯入 `solidfire-san`。

> Element驅動程式支援重複的Volume名稱。不過、如果有重複的磁碟區名稱、 Astra 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。

以下示例將在後端導入 `element-managed`卷 `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    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Google Cloud Platform

Astra Trident 支援使用驅動程式進行 Volume 匯入 `gcp-cvs`。

> 若要在 Google Cloud Platform 中匯入以 NetApp Cloud Volumes Service 為後盾的 Volume 、請依其 Volume 路徑識別該 Volume 。Volume 路徑是磁碟區匯出路徑的一部分、位於之後 `:/`。例如，如果匯出路徑為，則 `10.0.0.1:/adroit-jolly-swift`磁碟區路徑為 `adroit-jolly-swift`。

Google Cloud Platform 範例

以下範例會在後端 `gcpcvs_YEppr`以的 `Volume` 路徑匯入 `gcp-cvs` Volume `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

Astra Trident 支援使用驅動程式進行 Volume 匯入 `azure-netapp-files`。

> 若要匯入 Azure NetApp Files Volume 、請依磁碟區路徑識別該磁碟區。Volume 路徑是磁碟區匯出路徑的一部分、位於之後 `:/`。例如，如果掛載路徑爲，則 `10.0.0.2:/importvol1`卷路徑爲 `importvol1`。

以下示例使用卷路徑 `importvol1`導入 `azure-netapp-files`後端上的卷 `azurenetappfiles_40517`。

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

自訂磁碟區名稱和標籤

使用 Astra Trident 、您可以為您建立的 Volume 指派有意義的名稱和標籤。這有助於您識別並輕鬆地將磁碟區對應至各自的 Kubernetes 資源（ PVCS ）。您也可以在後端層級定義範本、以建立自訂磁碟區名稱和自訂標籤；您建立、匯入或複製的任何磁碟區都會遵守

這些範本。

開始之前

可自訂的 Volume 名稱和標籤支援：

1. Volume 建立、匯入及複製作業。
2. 在 ONTAP NAS 經濟驅動程式的情況下、只有 Qtree Volume 的名稱符合名稱範本。
3. 在 ONTAP SAN 經濟型驅動程式的情況下、只有 LUN 名稱符合名稱範本。

限制

1. 可自訂的磁碟區名稱僅與 ONTAP 內部部署驅動程式相容。
2. 可自訂的 Volume 名稱不適用於現有的 Volume 。

可自訂 **Volume** 名稱的主要行為

1. 如果名稱範本中的語法無效而導致失敗、則後端建立會失敗。但是、如果範本應用程式失敗、則會根據現有的命名慣例來命名磁碟區。
2. 如果使用後端組態的名稱範本命名磁碟區、則不適用儲存前置詞。任何所需的前置字元值都可以直接新增至範本。

名稱範本和標籤的後端組態範例

自訂名稱範本可在根和 / 或集區層級定義。

根層級範例

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

集區層級範例

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

名稱範本範例

- 範例 1* ：

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

- 範例 2 * ：

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

需要考量的重點

1. 在 Volume 匯入的情況下、只有現有的 Volume 具有特定格式的標籤時、標籤才會更新。例如 `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`：。
2. 在託管 Volume 匯入的情況下、 Volume 名稱會遵循在後端定義的根層級所定義的名稱範本。
3. Astra Trident 不支援使用含有儲存前置碼的 Slice 運算子。
4. 如果範本不會產生唯一的 Volume 名稱、 Astra Trident 會附加幾個隨機字元、以建立唯一的 Volume 名稱。
5. 如果 NAS 經濟 Volume 的自訂名稱長度超過 64 個字元、 Astra Trident 會根據現有的命名慣例來命名磁碟區。對於所有其他 ONTAP 驅動程式、如果磁碟區名稱超過名稱限制、磁碟區建立程序就會失敗。

跨命名空間共用**NFS**磁碟區

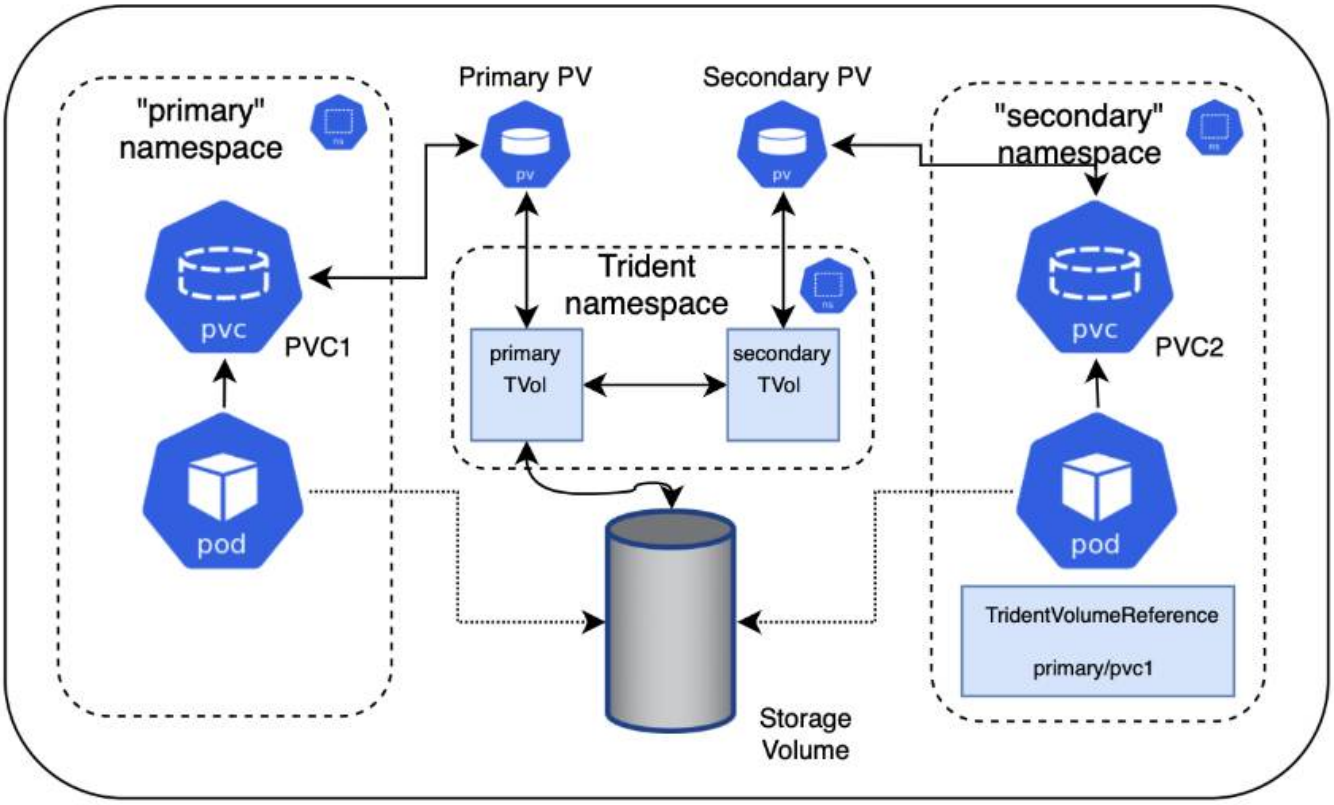
使用Astra Trident、您可以在主要命名空間中建立磁碟區、並將其共用於一或多個次要命名空間中。

功能

Astra TridentVolume Reference CR可讓您在一或多個Kubernetes命名空間中安全地共用ReadWriteMany（rwx）NFS磁碟區。此Kubernetes原生解決方案具有下列優點：

- 多層存取控制、確保安全性
- 可搭配所有Trident NFS Volume驅動程式使用
- 不依賴tridentctl或任何其他非原生Kubernetes功能

此圖說明兩個Kubernetes命名空間之間的NFS Volume共用。



快速入門

您只需幾個步驟就能設定NFS Volume共享。

1 設定來源 **PVC** 以共用磁碟區

來源命名空間擁有者授予存取來源PVc中資料的權限。

2 授予在目的地命名空間中建立 **CR** 的權限

叢集管理員授予目的地命名空間擁有者建立TridentVolume Reference CR的權限。

3 在目的地命名空間中建立 **TridentVolume Reference**

目的地命名空間的擁有者會建立TridentVolume Reference CR來參照來源PVC。

4 在目的地命名空間中建立次級 **PVC**

目的地命名空間的擁有者會建立從屬的PVc、以使用來源PVc的資料來源。

設定來源和目的地命名空間

為了確保安全性、跨命名空間共用需要來源命名空間擁有者、叢集管理員和目的地命名空間擁有者的協同作業與行動。使用者角色會在每個步驟中指定。

步驟

1. * 來源命名空間擁有者： * (pvc1`在來源命名空間中建立 PVC （`namespace2（ PVC ）、以授予與目的地命名空間共用的權限）、 shareToNamespace

```
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
```

Astra Trident會建立PV及其後端NFS儲存磁碟區。

- 您可以使用以逗號分隔的清單、將永久虛擬儲存設備共用至多個命名空間。例如 trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4：。
- 您可以使用共用所有命名空間 *。例如、 trident.netapp.io/shareToNamespace: *
- 您可以隨時更新 PVC 以納入 `shareToNamespace`附註。

2. *叢集管理：*建立自訂角色和KUbeconfig、以授予目的地命名空間擁有者權限、以便在目的地命名空間中建立TridentVolume Reference CR。
3. *Destination 命名空間擁有者： * 在指向來源命名空間的目的地命名空間中建立 TridentVolume Reference CR pvc1 。

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

4. * 目的地命名空間擁有者： * (pvc2`在目的地命名空間中建立 PVC （`namespace2）使用 `shareFromPVC`註釋來指定來源 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

Astra Trident 會讀取 `shareFromPVC`目的地 PVC 上的註釋、並將目的地 PV 建立為次級磁碟區、而不會有本身的儲存資源指向來源 PV 、並共用來源 PV 儲存資源。目的地的PVc和PV似乎正常連結。

刪除共享Volume

您可以刪除跨多個命名空間共用的磁碟區。Astra Trident會移除對來源命名空間上磁碟區的存取權、並維持對其他共用該磁碟區的命名空間的存取權。當所有參照磁碟區的命名空間都移除時、Astra Trident會刪除該磁碟區。

用於 `tridentctl get`查詢從屬磁碟區

您可以使用[`tridentctl`公用程式執行 `get`命令來取得附屬磁碟區。如需更多資訊、請參閱連結：`../../lce-reference Trident / tridentctl.html[`tridentctl 命令和選項]。

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

旗標：

- `-h, --help`：有關 Volume 的幫助。
- `--parentOfSubordinate string`：將查詢限制在從屬來源 Volume 。
- `--subordinateOf string`：將查詢限制在 Volume 的從屬。

限制

- Astra Trident無法防止目的地命名空間寫入共用磁碟區。您應該使用檔案鎖定或其他程序來防止覆寫共

用Volume資料。

- 您無法移除或 `shareFromNamespace`註釋、或刪除 `CR` 、 `TridentVolumeReference`以撤銷對來源 `PVC` 的存取權 `shareToNamespace`。若要撤銷存取權、您必須刪除從屬的PVc。
- 在從屬磁碟區上無法執行快照、複製和鏡射。

以取得更多資訊

若要深入瞭解跨命名空間Volume存取：

- 請造訪。"在命名空間之間共用磁碟區：歡迎使用跨命名空間磁碟區存取"
- 觀看上的示範 "NetAppTV"。

使用 SnapMirror 複寫磁碟區

使用 Astra Control Provisioner 、您可以在一個叢集上的來源磁碟區和對等叢集上的目的地磁碟區之間建立鏡射關係、以便複寫資料以進行災難恢復。您可以使用命名的自訂資源定義（ CRD ）來執行下列作業：

- 建立磁碟區之間的鏡射關係（ PVCS ）
- 移除磁碟區之間的鏡射關係
- 中斷鏡射關係
- 在災難情況（容錯移轉）期間提升次要 Volume
- 在計畫性容錯移轉或移轉期間、將應用程式從叢集無損移轉至叢集

複寫先決條件

在您開始之前、請確定符合下列先決條件：

叢集 ONTAP

- * Astra Control Provisioner* ： Astra Control Provisioner 版本 23.10 或更新版本必須同時存在於使用 ONTAP 作為後端的來源叢集和目的地 Kubernetes 叢集上。
- * 授權 * ：使用資料保護套件的 ONTAP SnapMirror 非同步授權必須同時在來源和目的地 ONTAP 叢集上啟用。如需詳細資訊、請參閱 "SnapMirror授權概述ONTAP" 。

對等關係

- * 叢集與 SVM* ：必須對 ONTAP 儲存設備的後端進行對等處理。如需詳細資訊、請參閱 "叢集與SVM對等概觀" 。

  確保兩個 ONTAP 叢集之間複寫關係中使用的 SVM 名稱是唯一的。

- **Astra Control Provisioner** 和 **SVM** ：對等的遠端 SVM 必須可供目的地叢集上的 Astra Control Provisioner 使用。

支援的驅動程式

- ONTAP NAS 和 ONTAP SAN 驅動程式支援 Volume 複寫。

建立鏡射 PVC

請遵循下列步驟、並使用 CRD 範例在主要和次要磁碟區之間建立鏡射關係。

步驟

1. 在主 Kubernetes 叢集上執行下列步驟：
   a. 使用參數建立 StorageClass 物件 `trident.netapp.io/replication: true` 。

      範例

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

   b. 使用先前建立的 StorageClass 建立 PVC 。

      範例

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

   c. 使用本機資訊建立 MirrorRelationship CR 。

範例

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

Astra Control Provisioner 會擷取磁碟區的內部資訊和磁碟區目前的資料保護（ DP ）狀態、然後填入 MirrorRelationship 的狀態欄位。

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

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

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

2. 在次 Kubernetes 叢集上執行下列步驟：

a. 使用 trident.netapp.io/replication: true 參數建立 StorageClass 。

範例

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

b. 使用目的地和來源資訊建立 MirrorRelationship CR 。

範例

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

Astra Control Provisioner 將使用設定的關係原則名稱（或 ONTAP 的預設名稱）建立 SnapMirror 關係、並將其初始化。

c. 使用先前建立的 StorageClass 建立 PVC 、作為次要（ SnapMirror 目的地）。

範例

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

Astra Control Provisioner 會檢查 TridentMirrorRelationship CRD 、如果關係不存在、則無法建立 Volume 。如果存在這種關係、 Astra Control Provisioner 將確保新的 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 Mirror Relationship （ TMR ）是一種 CRD 、代表 PVC 之間複寫關係的一端。目的地 TMR 具有狀態、可告知 Astra Control Provisioner 所需的狀態。目的地 TMR 有下列狀態：

- * 建立 * ：本機 PVC 是鏡射關係的目的地 Volume 、這是新的關係。
- * 升級 * ：本機 PVC 為可讀寫且可掛載、目前無鏡射關係。
- * 重新建立 * ：本機 PVC 是鏡射關係的目的地 Volume 、先前也屬於該鏡射關係。
  - 如果目的地磁碟區與來源磁碟區有任何關係、則必須使用重新建立的狀態、因為它會覆寫目的地磁碟區內容。
  - 如果磁碟區先前未與來源建立關係、則重新建立的狀態將會失敗。

在非計畫性容錯移轉期間升級次要 PVC

在次 Kubernetes 叢集上執行下列步驟：

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

在規劃的容錯移轉期間升級次要 PVC

在計畫性容錯移轉（移轉）期間、請執行下列步驟來升級次要 PVC ：

步驟

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

   範例

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

3. 在次要 Kubernetes 叢集上、將 *TridentMirrorRelationationship _ CR* 的 *_spec.state* 欄位更新為 *updated* 、 *spec.promotedSnapshotHandle* 更新為快照的內部名稱。
4. 在次要 Kubernetes 叢集上、確認要升級的 TridentMirrorRelationship 狀態（ STATUS.STATUS 欄位）。

在容錯移轉後還原鏡射關係

還原鏡射關係之前、請先選擇要設為新主要的一面。

步驟

1. 在次要 Kubernetes 叢集上、確保已更新 TridentMirrorRelationship 上 *spec.remoteVolumeHandle* 欄位的值。
2. 在次 Kubernetes 叢集上、將 TridentMirrorRelationship 的 _spec.mirror 欄位更新為 `reestablished`。

其他作業

Astra Control Provisioner 支援在主要和次要磁碟區上執行下列作業：

將主要 **PVC** 複製到新的次要 **PVC**

請確定您已擁有主要 PVC 和次要 PVC 。

步驟

1. 從已建立的次要（目的地）叢集刪除 PersistentVolume Claim 和 TridentMirrorRelationship CRD 。
2. 從主（來源）叢集刪除 TridentMirrorRelationship CRD 。
3. 在主要（來源）叢集上建立新的 TridentMirrorRelationship CRD 、以用於您要建立的新次要（目的地）PVC 。

調整鏡射、主要或次要 **PVC** 的大小

PVC 可以正常調整大小、如果資料量超過目前大小、 ONTAP 會自動擴充任何目的地 flevxols 。

從 **PVC** 移除複寫

若要移除複寫、請在目前的次要磁碟區上執行下列其中一項作業：

- 刪除次要 PVC 上的 MirrorRelationship 。這會中斷複寫關係。
- 或者、將 spec.state 欄位更新為 *updated* 。

刪除 **PVC** （先前已鏡射）

Astra Control Provisioner 會檢查複寫的 PVCS 、並在嘗試刪除磁碟區之前先釋放複寫關係。

刪除 **TMR**

在鏡射關係的一側刪除 TMR 會導致其餘 TMR 在 Astra Control Provisioner 完成刪除之前轉換至 升 遷狀態。如果選取要刪除的 *TMR* 已處於 _ 升級 _ 狀態、則沒有現有的鏡射關係、 *TMR* 將會移除、 *Astra Control Provisioner* 會將本機 *PVC* 升級為 *_ReadWrite* 。此刪除作業會在 ONTAP 中針對本機磁碟區釋出 SnapMirror 中繼資料。如果此磁碟區在未來的鏡射關係中使用、則在建立新的鏡射關係時、它必須使用具有 _ 建立 _ 磁碟區複寫狀態的新 TMR 。

當 ONTAP 連線時、請更新鏡射關係

建立鏡射關係之後、可以隨時更新它們。您可以使用 `state: promoted` 或 `state: reestablished` 欄位來更新關聯。將目的地 Volume 升級為一般 ReadWrite Volume 時、您可以使用 *promotedSnapshotHandle* 來指定特定快照、將目前的 Volume 還原至。

當 ONTAP 離線時更新鏡射關係

您可以使用 CRD 來執行 SnapMirror 更新、而無需 Astra Control 直接連線至 ONTAP 叢集。請參閱下列 TridentActionMirrorUpdate 範例格式：

範例

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

`status.state` 反映 TridentActionMirrorUpdate CRD 的狀態。它可以取自 *sued* 、 *in progress* 或 *Failed* 的值。

啟用 Astra Control Provisioner

Trident 版本 23.10 及更新版本包含使用 Astra Control Provisioner 的選項，可讓獲授權的 Astra Control 使用者存取進階儲存資源配置功能。Astra Control Provisioner 除了提供標準 Astra Trident CSI 型功能之外、還提供這項延伸功能。您可以使用此程序來啟用和安裝 Astra Control Provisioner 。

您的 Astra Control Service 訂閱會自動包含 Astra Control Provisioner 使用授權。

Astra Control 的更新將取代 Astra Trident 做為儲存資源配置程式和 Orchestrator 、並成為 Astra Control 使用的必要項目。因此、強烈建議 Astra Control 使用者啟用 Astra Control Provisioner 。Astra Trident 將繼續保持開放原始碼、並以 NetApp 的新 CSI 和其他功能來發行、維護、支援及更新。

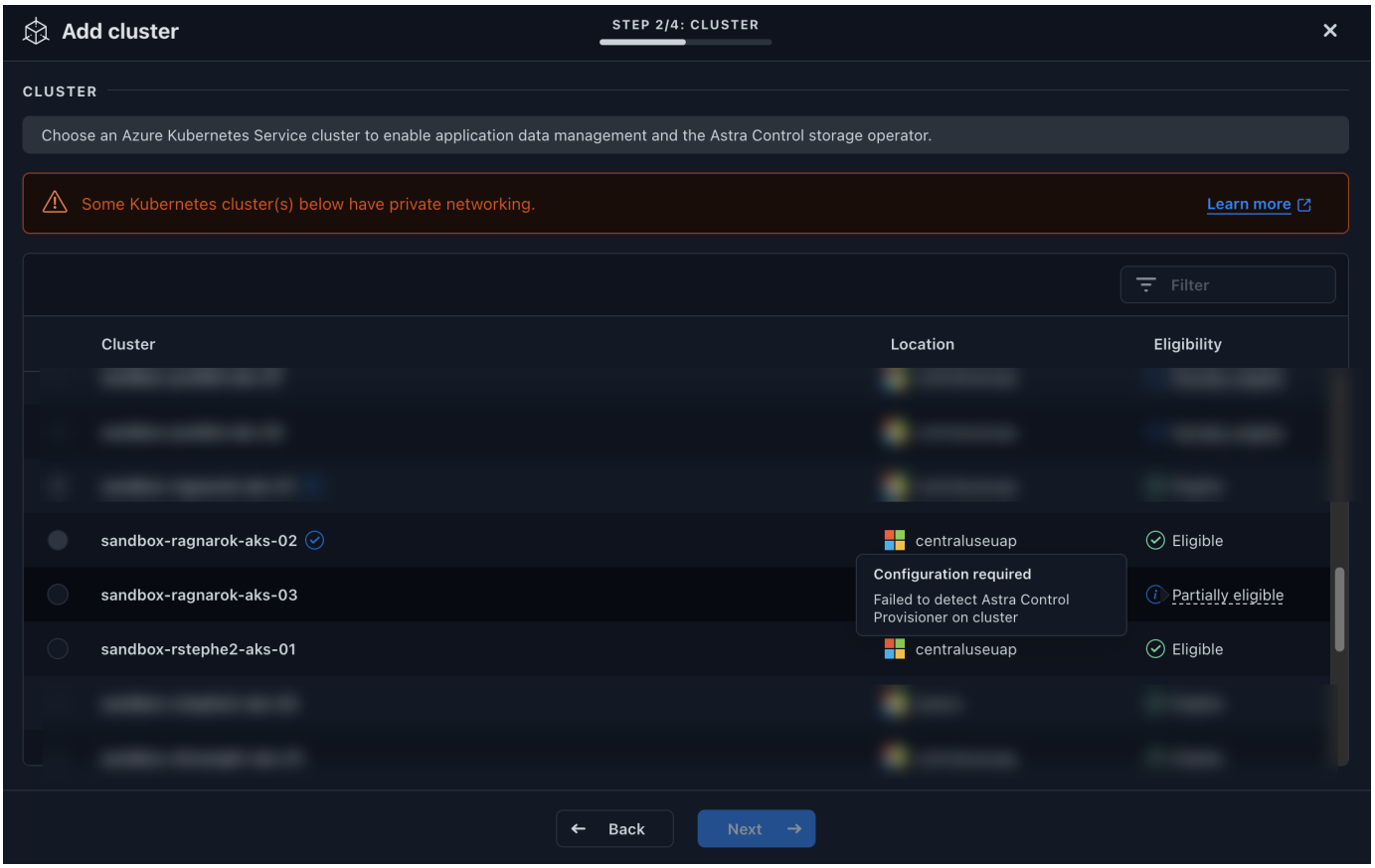
如何知道我是否需要啟用 **Astra Control Provisioner** ？

如果您將叢集新增至先前未安裝 Astra Trident 的 Astra Control Service 、叢集將會標示為 `Eligible`。您之後 "將叢集新增至 Astra Control"、 Astra Control Provisioner 將會自動啟用。

如果您的叢集未標記、則 `Eligible`會標記為 `Partially eligible` 下列其中一項：

- 它使用的是舊版 Astra Trident
- 它使用的 Astra Trident 23.10 尚未啟用置備程式選項
- 這是一種不允許自動啟用的叢集類型

在 `Partially eligible` 案例中、請使用這些指示來手動為叢集啟用 Astra Control Provisioner 。



啟用 **Astra Control Provisioner** 之前

如果您現有的 Astra Trident 不含 Astra Control Provisioner 、而且想要啟用 Astra Control Provisioner 、請先執行下列步驟：

- * 如果您已安裝 Astra Trident 、請確認其版本位於四個版本的視窗 * ：如果 Astra Trident 位於 24.02 版的四個版本視窗內、您可以使用 Astra Control Provisioner 直接升級至 Astra Trident 24.02 。例如、您可以直接從 Astra Trident 23.04 升級至 24.02 。
- * 確認您的叢集具有 AMD64 系統架構 * ： Astra Control Provisioner 映像同時在 AMD64 和 ARM64 CPU 架構中提供、但 Astra Control 僅支援 AMD64 。

步驟

1. 存取 NetApp Astra Control 影像登錄：
   a. 登入 Astra Control Service UI 並記錄 Astra Control 帳戶 ID 。
      i. 選取頁面右上角的圖示。
      ii. 選擇* API存取*。
      iii. 記下您的帳戶 ID 。
   b. 從同一頁面選取 * 產生 API 權杖 * 、然後將 API 權杖字串複製到剪貼簿、並將其儲存在編輯器中。
   c. 使用您偏好的方法登入 Astra Control 登錄：

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

```
crane auth login cr.astra.netapp.io -u <account-id> -p <api-token>
```

2. （僅限自訂登錄）請依照下列步驟將映像移至自訂登錄。如果您不使用登錄，請遵循中的 Trident 運算子步驟 下一節。

> 您可以使用 Podman 而非 Docker 來執行下列命令。如果您使用的是 Windows 環境、建議您使用 PowerShell 。

Docker

a. 從登錄中拉出 Astra Control Provisioner 映像：

> 所擷取的映像不支援多個平台、而且僅支援與擷取映像的主機相同的平台、例如 Linux AMD64 。

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0
--platform <cluster platform>
```

範例：

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0
--platform linux/amd64
```

b. 標記影像：

```
docker tag cr.astra.netapp.io/astra/trident-acp:24.02.0
<my_custom_registry>/trident-acp:24.02.0
```

c. 將映像推送至自訂登錄：

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

起重機

a. 將 Astra Control Provisioner 資訊清單複製到您的自訂登錄：

```
crane copy cr.astra.netapp.io/astra/trident-acp:24.02.0
<my_custom_registry>/trident-acp:24.02.0
```

3. 確定原始 Astra Trident 安裝方法是否使用。
4. 使用您最初使用的安裝方法、在 Astra Trident 中啟用 Astra Control Provisioner ：

Astra Trident 運算子

a. "下載 Astra Trident 安裝程式並將其解壓縮"。

b. 如果您尚未安裝 Astra Trident 、或是從原始 Astra Trident 部署中移除運算子、請完成下列步驟：

   i. 建立客戶需求日：

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.1
6.yaml
```

   ii. 創建 trident 命名空間 (`kubectl create namespace trident`）或確認 trident 命名空間仍然存在 (`kubectl get all -n trident`）。如果已移除命名空間、請重新建立。

c. 將 Astra Trident 更新為 24.06.0 ：

> 對於運行 Kubernetes 1.24 或更早版本的羣集，請使用 `bundle_pre_1_25.yaml`。對於運行 Kubernetes 1.25 或更高版本的羣集，請使用 `bundle_post_1_25.yaml`。

```
kubectl -n trident apply -f trident-installer/deploy/<bundle-
name.yaml>
```

d. 確認 Astra Trident 正在執行：

```
kubectl get torc -n trident
```

回應：

```
NAME      AGE
trident   21m
```

e. [[Pull 機密]] 如果您有使用機密的登錄、請建立秘密來拉出 Astra Control Provisioner 映像：

```
kubectl create secret docker-registry <secret_name> -n trident
--docker-server=<my_custom_registry> --docker-username=<username>
--docker-password=<token>
```

f. 編輯 TridentOrchestrator CR 並進行下列編輯：

```
kubectl edit torc trident -n trident
```

i. 設定 Astra Trident 映像的自訂登錄位置、或從 Astra Control 登錄或中拉出 (`tridentImage: <my_custom_registry>/trident:24.02.0 tridentImage: netapp/trident:24.06.0`) 。

ii. 啓用 Astra Control Provisioner (`enableACP: true`) 。

iii. 設定 Astra Control Provisioner 映像的自訂登錄位置、或從 Astra Control 登錄或將其拉出 (`acpImage: <my_custom_registry>/trident-acp:24.02.0 acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0`) 。

iv. 如果您先前在此程序中建立 影像拉出秘密 、您可以在此設定 (`imagePullSecrets: - <secret_name>`) 。請使用您在先前步驟中建立的相同名稱密碼名稱。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: <registry>/trident:24.06.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.06.0
  imagePullSecrets:
  - <secret_name>
```

g. 儲存並結束檔案。部署程序將會自動開始。

h. 確認已建立運算子、部署和複本集。

```
kubectl get all -n trident
```

Kubernetes叢集中只應有*一個運算子執行個體*。請勿建立 Astra Trident 運算子的多個部署。

i. 驗證該 `trident-acp` 容器是否正在運行且 `acpVersion` `24.02.0` 狀態爲 `Installed`：

```
kubectl get torc -o yaml
```

回應：

```
status:
  acpVersion: 24.02.0
  currentInstallationParams:
    ...
    acpImage: <registry>/trident-acp:24.02.0
    enableACP: "true"
    ...
  ...
  status: Installed
```

試用

a. "下載 Astra Trident 安裝程式並將其解壓縮"。

b. "如果您有現有的 Astra Trident 、請將其從裝載它的叢集上解除安裝"。

c. 安裝 Astra Trident 並啟用 Astra Control Provisioner (`--enable-acp=true`) ：

```
./tridentctl -n trident install --enable-acp=true --acp
-image=mycustomregistry/trident-acp:24.02
```

d. 確認 Astra Control Provisioner 已啟用：

```
./tridentctl -n trident version
```

回應：

```
+----------------+----------------+-------------+ | SERVER
VERSION | CLIENT VERSION | ACP VERSION | +----------------
+----------------+-------------+ | 24.02.0 | 24.02.0 | 24.02.0. |
+----------------+----------------+-------------+
```

掌舵

a. 如果您已安裝 Astra Trident 23.07.1 或更早版本、則 "解除安裝" 為操作員和其他元件。

b. 如果 Kubernetes 叢集執行 1.24 或更早版本、請刪除 PSP ：

```
kubectl delete psp tridentoperatorpod
```

c. 新增Astra Trident Helm儲存庫：

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

d. 更新 Helm 圖表：

```
helm repo update netapp-trident
```

回應：

```
Hang tight while we grab the latest from your chart
repositories...
...Successfully got an update from the "netapp-trident" chart
repository
Update Complete. □Happy Helming!□
```

e. 列出影像：

```
./tridentctl images -n trident
```

回應：

```
| v1.28.0            | netapp/trident:24.06.0|
|                    | docker.io/netapp/trident-
autosupport:24.06|
|                    | registry.k8s.io/sig-storage/csi-
provisioner:v4.0.0|
|                    | registry.k8s.io/sig-storage/csi-
attacher:v4.5.0|
|                    | registry.k8s.io/sig-storage/csi-
resizer:v1.9.3|
|                    | registry.k8s.io/sig-storage/csi-
snapshotter:v6.3.3|
|                    | registry.k8s.io/sig-storage/csi-node-
driver-registrar:v2.10.0 |
|                    | netapp/trident-operator:24.06.0 (optional)
```

f. 確保 Trident 操作員 24.06.0 可用：

```
helm search repo netapp-trident/trident-operator --versions
```

回應：

```
NAME                                CHART VERSION   APP VERSION
DESCRIPTION
netapp-trident/trident-operator 100.2406.0      24.06.0         A
```

g. 使用 `helm install` 並執行下列其中一個選項、其中包括這些設定：

- 部署位置的名稱
- Astra Trident 版本
- Astra Control Provisioner 映像的名稱
- 啟用資源配置程式的旗標
- （選用）本機登錄路徑。如果您使用的是本機登錄，則 "Trident 影像" 可以位於一個登錄或不同的登錄中，但所有 CSI 映像都必須位於相同的登錄中。
- Trident 命名空間

選項

。沒有登錄的映像

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=cr.astra.netapp.io/astra/trident-
acp:24.06.0 --set enableACP=true --set operatorImage=netapp/trident-
operator:24.06.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.06
--set tridentImage=netapp/trident:24.06.0 --namespace trident
```

。一個或多個登錄中的影像

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=<your-registry>:<acp image> --set
enableACP=true --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=netapp/trident-operator:24.06.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.06
--set tridentImage=netapp/trident:24.06.0 --namespace trident
```

您可以使用 `helm list` 檢閱安裝詳細資料、例如名稱、命名空間、圖表、狀態、應用程式版本、 和修訂版編號。

如果您在使用 Helm 部署 Trident 時遇到任何問題、請執行此命令以完全解除安裝 Astra Trident ：

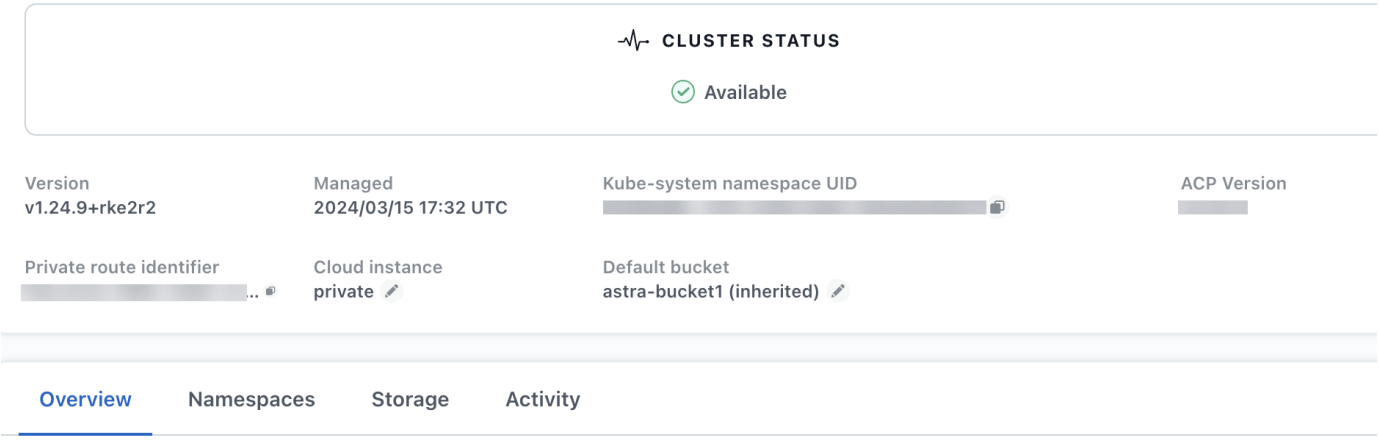
```
./tridentctl uninstall -n trident
```

- 在再次嘗試啟用 Astra Control Provisioner 之前、請勿 * "完全移除 Astra Trident 客戶需求日" 作為解除安裝的一部分。

結果

Astra Control Provisioner 功能已啟用、您可以使用任何適用於所執行版本的功能。

安裝 Astra Control Provisioner 之後、在 Astra Control UI 中裝載置備程式的叢集將會顯示 `ACP version` 而非 `Trident version` 欄位和目前安裝的版本號碼。



以取得更多資訊

- "Astra Trident 升級文件"

使用「csi拓撲」

Astra Trident 可以利用來選擇性地建立磁碟區並將其附加至 Kubernetes 叢集中的節點 "「csi拓撲」功能"。

總覽

使用「csi拓撲」功能、可根據區域和可用性區域、限制對磁碟區的存取、只能存取一部分節點。如今、雲端供應商可讓Kubernetes管理員建立以區域為基礎的節點。節點可位於某個區域內的不同可用度區域、或位於不同區域之間。為了協助在多區域架構中配置工作負載的磁碟區、Astra Trident使用了csi拓撲。

瞭解有關 CSI 拓撲功能的更多 "請按這裡"信息。

Kubernetes提供兩種獨特的Volume繫結模式：

- 如果 `VolumeBindingMode`設置爲 `Immediate`，Astra Trident 將在沒有任何拓撲感知的情況下創建卷。建立永久虛擬磁碟時、即會處理磁碟區繫結和動態資源配置。這是預設值 `VolumeBindingMode`、適用於不強制執行拓撲限制的叢集。持續磁碟區的建立不需依賴要求的 Pod 排程需求。

- `VolumeBindingMode`設為 `WaitForFirstConsumer`時、永久 Volume 的建立與繫結將延遲、直到排程並建立使用 PVC 的 Pod 為止。如此一來、就能建立磁碟區、以符合拓撲需求所強制執行的排程限制。

> `WaitForFirstConsumer`綁定模式不需要拓撲標籤。這可獨立於「csi拓撲」功能使用。

您需要的產品

若要使用「csi拓撲」、您需要下列項目：

- 執行的 Kubernetes 叢集"支援的Kubernetes版本"

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

- 叢集中的節點應具有標籤、以引入拓撲感知(`topology.kubernetes.io/region`和 `topology.kubernetes.io/zone`）。在安裝Astra Trident以識別拓撲之前、這些標籤*應該會出現在叢集*的節點上。

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

步驟**1**：建立可感知拓撲的後端

Astra Trident儲存後端可根據可用性區域、選擇性地配置磁碟區。每個後端都可以帶有一個可選的 `supportedTopologies` 區塊、代表支援的區域和區域清單。對於使用此類後端的StorageClass、只有在受支援地區/區域中排程的應用程式要求時、才會建立Volume。

以下是後端定義範例：

YAML

```
---
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`用於提供每個後端的區域和區域清單。這些區域和區域代表StorageClass中可提供的允許值清單。對於包含後端所提供之區域和區域子集的StorageClass、Astra Trident會在後端建立磁碟區。

您也可以定義 `supportedTopologies`每個儲存池。請參閱下列範例：

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

在此範例中 `region`、和 `zone`標籤代表儲存池的位置。 `topology.kubernetes.io/region`並 `topology.kubernetes.io/zone`指定儲存資源池的使用來源。

步驟2：定義可感知拓撲的StorageClass

根據提供給叢集中節點的拓撲標籤、可以定義StorageClass以包含拓撲資訊。這將決定做為所提出之永久虛擬磁碟要求候選的儲存資源池、以及可以使用Trident所提供之磁碟區的節點子集。

請參閱下列範例：

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

在上面提供的 StorageClass 定義中 `volumeBindingMode`，設置爲 `WaitForFirstConsumer`。在Pod中引用此StorageClass所要求的PVCS之前、系統不會對其採取行動。此外、還 `allowedTopologies`提供要使用的區域和區域。 `netapp-san-us-east1`StorageClass 將在上述定義的後端建立 PVCS `san-backend-us-east1 。

步驟3：建立並使用PVC

建立StorageClass並對應至後端後端後端之後、您現在就可以建立PVCS。

請參閱以下範例 `spec`：

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

使用此資訊清單建立永久虛擬環境可能會產生下列結果：

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

若要Trident建立磁碟區並將其連結至PVc、請在Pod中使用PVc。請參閱下列範例：

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

此 podSpec 會指示 Kubernetes 在區域中的節點上排程 Pod `us-east1` 、並從或 `us-east1-b`區域中的任何節點中進行選擇 `us-east1-a`。

請參閱下列輸出：

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

更新後端以納入 supportedTopologies

可更新現有的後端、以納入使用 `tridentctl backend update`清單 `supportedTopologies`。這不會影響已配置的磁碟區、而且只會用於後續的PVCS。

如需詳細資訊、請參閱

- "管理容器的資源"
- "節點選取器"
- "關聯性與反關聯性"
- "污染與容許"

使用快照

Kubernetes 持續磁碟區（ PV ）的磁碟區快照可啟用磁碟區的時間點複本。您可以建立使用 Astra Trident 建立的磁碟區快照、匯入 Astra Trident 外部建立的快照、從現有快照建立新的磁碟區、以及從快照復原磁碟區資料。

總覽

Volume Snapshot 受 `ontap-nas`、 `ontap-nas-flexgroup`、 `ontap-san`、、 `ontap-san-economy` `solidfire-san`、 `gcp-cvs`和 `azure-netapp-files`驅動程式。

開始之前

您必須擁有外部快照控制器和自訂資源定義（ CRD ）、才能使用快照。這是Kubernetes Orchestrator的責任（例如：Kubeadm、GKE、OpenShift）。

如果您的 Kubernetes 發佈不包含快照控制器和 CRD 部署 Volume Snapshot 控制器、請參閱。

> 如果在 GKE 環境中建立隨需磁碟區快照、請勿建立快照控制器。GKE使用內建的隱藏式快照控制器。

建立磁碟區快照

步驟

1. 建立 `VolumeSnapshotClass`。如需詳細資訊，請"Volume SnapshotClass"參閱。
   - `driver`指向 Astra Trident CSI 驅動程式。
   - `deletionPolicy`可以是 `Delete`或 `Retain`。設為 `Retain`時、即使刪除物件、儲存叢集上的基礎實體快照仍會保留 `VolumeSnapshot`。

     範例

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

2. 建立現有 PVC 的快照。

   範例

   - 此範例會建立現有PVc的快照。

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

   - 此範例會為名稱為 PVC 的磁碟區快照物件建立磁碟區快照物件 `pvc1`、並將快照名稱設定為 `pvc1-snap`。Volume Snapshot 類似於 PVC 、與代表實際快照的物件相關聯 `VolumeSnapshotContent`。

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

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

   - 您可以透過說明來識別 `VolumeSnapshotContent` Volume Snapshot 的物件 `pvc1-snap`。

`Snapshot Content Name`識別用於處理此快照的 Volume SnapshotContent 物件。此 `Ready To Use`參數表示快照可用於建立新的 PVC 。

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

從磁碟區快照建立 PVC

您可以使用名為的 Volume Snapshot 作為資料來源、來 `dataSource`建立 PVC `<pvc-name>` 。建立好永久虛擬基礎架構之後、就能將它附加到Pod上、就像使用任何其他永久虛擬基礎架構一樣使用。

> 將在來源 Volume 所在的同一個後端建立 PVC 。請參閱 "KB ：無法在替代後端建立 Trident PVC Snapshot 的 PVC"。

以下範例使用作為資料來源來建立 PVC `pvc1-snap` 。

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

匯入 Volume 快照

Astra Trident 支援"Kubernetes 預先配置的快照程序"、可讓叢集管理員建立 `VolumeSnapshotContent`物件並匯入 Astra Trident 以外建立的快照。

開始之前

Astra Trident 必須已建立或匯入快照的父磁碟區。

步驟

1. * 叢集管理： * 建立 `VolumeSnapshotContent`參照後端快照的物件。這會啟動 Astra Trident 中的快照工作流程。
   - 在中指定後端快照的名稱 `annotations` AS `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`。
   - 請在中 `snapshotHandle`指定 `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>`。這是通話中外部快照機提供給 Astra Trident 的唯一資訊 `ListSnapshots`。

     `<volumeSnapshotContentName>`由於 CR 命名限制、無法永遠符合後端快照名稱。

     範例

     以下示例創建 `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>
```

2. * 叢集管理： * 建立 `VolumeSnapshot`參照物件的 `CR `VolumeSnapshotContent 。這會要求存取以在指定的命名空間中使用 `VolumeSnapshot`。

   範例

   以下範例建立一個 `VolumeSnapshot`參照 `VolumeSnapshotContent`命名的 `import-snap-content` CR `import-snap 。

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

3. * 內部處理（不需採取任何行動）： * 外部快照機可辨識新建立的 `VolumeSnapshotContent`、並執行 `ListSnapshots`通話。`Astra Trident 會建立 `TridentSnapshot`。

   。外部快照器會將設為 `readyToUse` `VolumeSnapshot`、設 `VolumeSnapshotContent`為 `true`。

   。Trident 退貨 `readyToUse=true`。

4. * 任何使用者： * 建立 `PersistentVolumeClaim`以參照新的 `VolumeSnapshot`、其中 `spec.dataSource`（或 `spec.dataSourceRef`）名稱為 `VolumeSnapshot`名稱。

   範例

   以下範例建立一個 PVC ，參照 `VolumeSnapshot`命名的 `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 **Volume** 資料

快照目錄預設為隱藏、以協助使用和 `ontap-nas-economy`驅動程式進行資源配置的磁碟區達到最大相容性 `ontap-nas`。啟用 `.snapshot`目錄、直接從快照中恢復資料。

使用 Volume Snapshot Restore ONTAP CLI 將磁碟區還原至先前快照中記錄的狀態。

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

當您還原快照複本時、會覆寫現有的 Volume 組態。建立快照複本之後對 Volume 資料所做的變更將會遺失。

快照目錄預設為隱藏、以協助使用和 `ontap-nas-economy`驅動程式進行資源配置的磁碟區達到最大相容性 `ontap-nas`。啟用 `.snapshot`目錄、直接從快照中恢復資料。

當您還原快照複本時、會覆寫現有的 Volume 組態。建立快照複本之後對 Volume 資料所做的變更將會遺失。

從快照進行原位磁碟區還原

Astra Control Provisioner 使用（ TASR ） CR 從快照中提供快速的原位磁碟區還原 `TridentActionSnapshotRestore` 功能。此 CR 是 Kubernetes 的必要行動、在作業完成後不會持續存在。

Astra Control Provisioner 支援 `ontap-san`、 `ontap-san-economy`、 `ontap-nas`、 `ontap-nas-flexgroup` `azure-netapp-files`、、 `gcp-cvs` `solidfire-san` 和驅動程式。

開始之前

您必須擁有受約束的 PVC 和可用的 Volume 快照。

- 確認 PVC 狀態為「已連結」。

```
kubectl get pvc
```

- 驗證 Volume 快照是否已準備就緒可供使用。

```
kubectl get vs
```

步驟

1. 建立 TASR CR 。本示例爲 PVC 和 Volume Snapshot 創建 CR `pvc1` `pvc1-snapshot`。

```
cat tasr-pvc1-snapshot.yaml

apiVersion: v1
kind: TridentActionSnapshotRestore
metadata:
  name: this-doesnt-matter
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. 套用 CR 以從快照還原。此示例從 Snapshot 恢復 `pvc1`。

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

tridentactionsnapshotrestore.trident.netapp.io/this-doesnt-matter
created
```

結果

Astra Control Provisioner 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: v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: this-doesnt-matter
    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、 Astra Control Provisioner 不會在發生故障時自動重試作業。您需要再次執行此作業。
- 不具備管理員存取權限的 Kubernetes 使用者可能必須獲得管理員的權限、才能在其應用程式命名空間中建立 TASR CR 。

刪除含有相關快照的 **PV**

刪除具有相關快照的持續Volume時、對應的Trident Volume會更新為「刪除狀態」。移除 Volume 快照以刪除 Astra Trident Volume 。

部署 **Volume Snapshot** 控制器

如果您的Kubernetes發佈版本未包含快照控制器和客戶需求日、您可以依照下列方式進行部署。

步驟

1. 建立Volume Snapshot客戶需求日。

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

2. 建立Snapshot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、請開啟 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`並更新 `namespace`您的命名空間。

相關連結

- "Volume快照"
- "Volume SnapshotClass"