



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
跨命名空間共用NFS磁碟區	21
功能	22
快速入門	22
設定來源和目的地命名空間	23
刪除共享Volume	24
使用 <code>tridentctl get</code> 查詢從屬Volume	24
限制	25
以取得更多資訊	25
使用 SnapMirror 複寫磁碟區	25
複寫先決條件	25
建立鏡射 PVC	26
Volume 複寫狀態	29
在非計畫性容錯移轉期間升級次要 PVC	29
在規劃的容錯移轉期間升級次要 PVC	29
在容錯移轉後還原鏡射關係	30
其他作業	30
當 ONTAP 連線時、請更新鏡射關係	31
當 ONTAP 離線時更新鏡射關係	31
啟用 Astra Control Provisioner	31
使用「csi拓撲」	40
總覽	40
步驟1：建立可感知拓撲的後端	42
步驟2：定義可感知拓撲的StorageClass	44
步驟3：建立並使用PVC	45
更新後端以納入 supportedTopologies	48
如需詳細資訊、請參閱	48
使用快照	48

總覽	48
建立磁碟區快照	49
從磁碟區快照建立 PVC	50
匯入 Volume 快照	51
使用快照恢復 Volume 資料	53
從快照進行原位磁碟區還原	53
刪除含有相關快照的 PV	55
部署 Volume Snapshot 控制器	55
相關連結	56

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

答 "*PersistentVolume*" (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、與命名的 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、並提供與命名 StorageClass 相關的 `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. 確認磁碟區已掛載到上 `/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
```

請參閱 "[Kubernetes和Trident物件](#)" 如需儲存類別如何與互動的詳細資訊、請參閱 `PersistentVolumeClaim` 以及用於控制 Astra Trident 如何配置容量的參數。

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)。



支援iSCSI Volume擴充 `ontap-san`、`ontap-san-economy`、`solidfire-san` 並需要Kubernetes 1.16及更新版本。

步驟1：設定**StorageClass**以支援**Volume**擴充

編輯 `StorageClass` 定義以設定 `allowVolumeExpansion` 欄位至 `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、請編輯此類以納入 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 以反映新的所需大小、此大小必須大於原始大小。

```
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。

```

kubect1 get pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound     pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi
RWO          ontap-san    8s

kubect1 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。

```

kubect1 get pod
NAME          READY    STATUS    RESTARTS  AGE
ubuntu-pod   1/1     Running   0          65s

kubect1 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建立至2Gi的PV大小、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：驗證擴充

您可以檢查Pvc、PV和Astra Trident Volume的大小、以正確驗證擴充作業：

```

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

若要調整NFS PV的大小、管理員必須先設定儲存類別、以允許透過設定來擴充磁碟區 allowVolumeExpansion 欄位至 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

```

如果您已建立不含此選項的儲存類別、則只要使用編輯現有的儲存類別即可 kubect1 edit storageclass 以允許磁碟區擴充。

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調整至1GiB、請編輯該PVC並設定組合 `spec.resources.requests.storage` 至 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。

- 不過 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。
 - 回收原則一開始設定為 retain 在 PV 中。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 delete、儲存磁碟區會在 PV 刪除時刪除。
- 根據預設、Astra Trident 會管理 PVC、並重新命名後端上的 FlexVol 和 LUN。您可以通過 `--no-manage` 用於匯入非託管磁碟區的旗標。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`。您也可以在此 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` 用於指定 Astra Trident 後端名稱的命令、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 支援使用匯入 Volume `ontap-nas` 和 `ontap-nas-flexgroup` 驅動程式：



- `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。

使用建立的每個 Volume `ontap-nas` 驅動程式 `FlexVol` 是 ONTAP 指在整個叢集上執行的功能。使用匯入 `FlexVols` `ontap-nas` 驅動程式的運作方式相同。可將已存在於某個叢集上的一個功能、匯入為 `FlexVol` ONTAP `ontap-nas` PVC。同樣地 `FlexGroup`、也可以將此資訊匯入為 `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

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

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

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

非託管 Volume

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

以下範例匯入 `unmanaged_volume` 在上 `ontap_nas` 後端：

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

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

SAN ONTAP

Astra Trident 支援使用匯入 Volume `ontap-san` 驅動程式：不支援使用匯入 Volume `ontap-san-economy` 驅動程式：

Astra Trident 可以匯入包含單一 LUN 的 ONTAP SAN FlexVols。這與一致 `ontap-san` 驅動程式、為 FlexVol 每個實體磁碟和 FlexVol 一個 LUN 建立一個實體。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

對於託管的 Volume、Astra Trident 會將 FlexVol 重新命名為 pvc-<uuid> 格式化及 FlexVol LUN 在功能區內 lun0。

下列範例會匯入 ontap-san-managed 上的顯示 FlexVol ontap_san_default 後端：

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

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

非託管 Volume

以下範例匯入 unmanaged_example_volume 在上 ontap_san 後端：

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
block	pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228	e3275890-7d80-4af6-90cc-c7a0759f555a	1.0 GiB	online	san-blog	false

如果您將 LUN 對應至與 Kubernetes 節點 IQN 共用 IQN 的 igroup、如下列範例所示、您將會收到錯誤訊息：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 NetApp Element 軟體和 NetApp HCI Volume 匯入 solidfire-san 驅動程式：



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。

下列範例會匯入 element-managed 後端上的 Volume element_default。

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

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

Google Cloud Platform

Astra Trident 支援使用匯入 Volume gcp-cvs 驅動程式：



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 路徑是之後 Volume 匯出路徑的一部分 :/。例如、如果匯出路徑為 10.0.0.1:/adroit-jolly-swift、磁碟區路徑為 adroit-jolly-swift。

Google Cloud Platform 範例

下列範例會匯入 gcp-cvs 後端上的 Volume gpcvs_YEppr 的磁碟區路徑 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

Astra Trident 支援使用匯入 Volume azure-netapp-files 驅動程式：



若要匯入 Azure NetApp Files Volume、請依磁碟區路徑識別該磁碟區。Volume 路徑是之後 Volume 匯出路徑的一部分 `:/`。例如、如果掛載路徑為 `10.0.0.2:/importvol1`、磁碟區路徑為 `importvol1`。

下列範例會匯入 azure-netapp-files 後端上的 Volume azurenetappfiles_40517 磁碟區路徑 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      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

跨命名空間共用NFS磁碟區

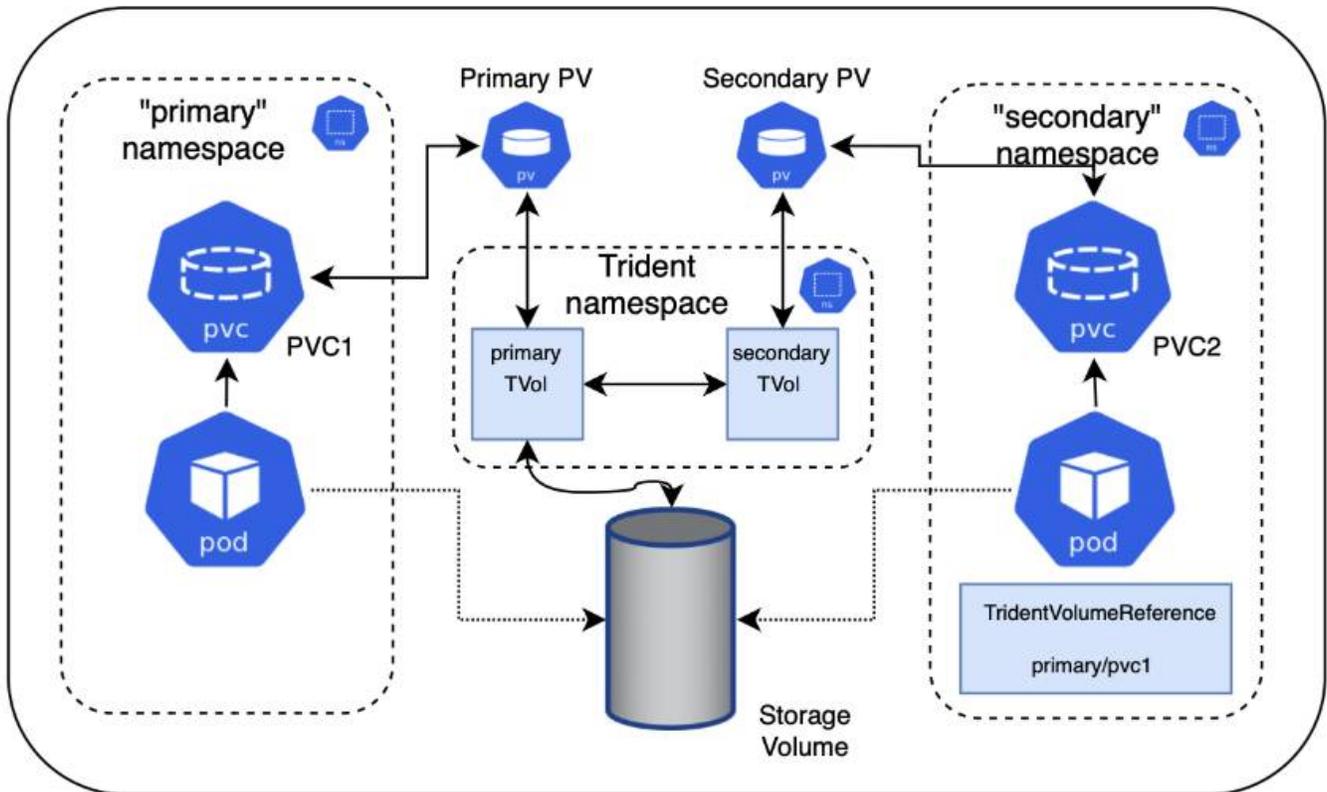
使用Astra Trident、您可以在主要命名空間中建立磁碟區、並將其共用於一或多個次要命名空間中。

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從屬的 **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：*建立 PVC (pvc1) (namespace2) 使用 shareToNamespace 註釋：

```
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 ◦
- 您可以使用共用至所有命名空間 *。例如、
trident.netapp.io/shareToNamespace: *
- 您可以更新 PVC, 以納入 shareToNamespace 隨時註釋。

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. *目的地命名空間擁有者：*在參照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：*建立一個PVC (pvc2) (namespace2) 使用 shareFromPVC 註釋以指定來源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
```



目的地PVC的大小必須小於或等於來源PVC。

結果

Astra Trident讀取 shareFromPVC 在目的地永久虛擬磁碟上註釋、並將目的地PV建立為從屬磁碟區、而其本身並無儲存資源指向來源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。

使用 tridentctl get 查詢從屬Volume

使用[tridentctl 公用程式、您可以執行 get 取得從屬磁碟區的命令。如需詳細資訊、請參閱連結：[../ Trident 參考/ tridentctl.html](#)[tridentctl 命令與選項]。

Usage:

```
tridentctl get [option]
```

旗標：

- `-h, --help`：Volume的說明。
- `--parentOfSubordinate string`：將查詢限制在從屬來源Volume。
- `--subordinateOf string`：將查詢限制在Volume的下屬。

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。
- 您無法藉由移除來撤銷對來源PVC的存取權 `shareToNamespace` 或 `shareFromNamespace` 註釋或刪除 `TridentVolumeReference` CR.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"](#)。

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 *TridentMirrorRelationship* _ 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 flexvols。

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。

The screenshot shows the 'Add cluster' wizard in the Astra Control console, specifically the 'STEP 2/4: CLUSTER' screen. The interface is dark-themed. At the top, there's a header 'Add cluster' and 'STEP 2/4: CLUSTER'. Below the header, there's a instruction: 'Choose an Azure Kubernetes Service cluster to enable application data management and the Astra Control storage operator.' A warning banner indicates 'Some Kubernetes cluster(s) below have private networking.' Below this is a table of clusters with columns for 'Cluster', 'Location', and 'Eligibility'. The table lists three clusters: 'sandbox-ragnarok-aks-02' (Eligible), 'sandbox-ragnarok-aks-03' (Partially eligible), and 'sandbox-rstphe2-aks-01' (Eligible). A tooltip for the 'Partially eligible' cluster shows 'Configuration required' and 'Failed to detect Astra Control Provisioner on cluster'. At the bottom, there are 'Back' and 'Next' buttons.

Cluster	Location	Eligibility
<input checked="" type="radio"/> sandbox-ragnarok-aks-02	centraluseuap	Eligible
<input type="radio"/> sandbox-ragnarok-aks-03	centraluseuap	Partially eligible
<input type="radio"/> sandbox-rstphe2-aks-01	centraluseuap	Eligible

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 24.02.0：



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:24.02.0)。
- ii. 啟用 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:24.02.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.02.0
  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 狀態為 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

- "下載 [Astra Trident 安裝程式](#) 並將其解壓縮"。
- "如果您有現有的 [Astra Trident](#) 、請將其從裝載它的叢集上解除安裝"。
- 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
```

- 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

- 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。
- 如果 [Kubernetes](#) 叢集執行 1.24 或更早版本、請刪除 [PSP](#) ：

```
kubectl delete psp tridentoperatorpod
```

- 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:24.02.0|
| | docker.io/netapp/trident-
autosupport:24.02|
| | 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:24.02.0 (optional)
```

f. 確保 Trident 操作員 24.02.0 可用：

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

回應：

NAME	CHART VERSION	APP VERSION	
DESCRIPTION			
netapp-trident/trident-operator	100.2402.0	24.02.0	A

9. 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:24.02.0 --set enableACP=true --set operatorImage=netapp/trident-operator:24.02.0 --set tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02 --set tridentImage=netapp/trident:24.02.0 --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:24.02.0 --set tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02 --set tridentImage=netapp/trident:24.02.0 --namespace trident
```

您可以使用 `helm list` 若要檢閱安裝詳細資料、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：

- 與 `VolumeBindingMode` 設定為 `WaitForFirstConsumer`、永久磁碟區的建立與繫結會延遲、直到排程並建立使用該永久磁碟的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` 繫結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

若使用「csi拓撲」、您需要下列項目：

- 執行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"}
```

- 叢集中的節點應該有標籤來介紹拓撲認知 (`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","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/os":"linux","node-role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

步驟1：建立可感知拓撲的後端

Astra Trident儲存後端可根據可用性區域、選擇性地配置磁碟區。每個後端都可隨附選用功能 `supportedTopologies` 代表必須支援之區域和區域清單的區塊。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。

您可以定義 `supportedTopologies` 也可依儲存資源池。請參閱下列範例：

```

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

```

在此範例中 `region` 和 `zone` 標籤代表儲存資源池的位置。 `topology.kubernetes.io/region` 和 `topology.kubernetes.io/zone` 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

```

在上述StorageClass定義中、volumeBindingMode 設為 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。而且、allowedTopologies 提供要使用的區域和區域。netapp-san-us-east1 StorageClass會在上建立PVCS san-backend-us-east1 上述定義的後端。

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。

請參閱範例 spec 以下：

```

---
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：

```

kubect1 create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubect1 get pvc
NAME          STATUS      VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san      Pending
2s
kubect1 describe pvc
Name:          pvc-san
Namespace:     default
StorageClass: netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type      Reason              Age   From
  ----      -
Normal    WaitForFirstConsumer  6s    persistentvolume-controller
waiting
for first consumer to be created before binding

```

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

```

此podSpec會指示Kubernetes在中的節點上排程pod us-east1 區域、並從中的任何節點中進行選擇 us-east1-a 或 us-east1-b 區域。

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

您可以更新現有的後端、以納入清單 supportedTopologies 使用 `tridentctl backend update`。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

支援 Volume Snapshot `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 發佈版本未包含快照控制器和 CRD、請參閱 [部署 Volume Snapshot 控制器](#)。



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、請參閱 "[Volume SnapshotClass](#)"。
 - driver 指向 Astra Trident CSI 驅動程式。
 - deletionPolicy 可以 Delete 或 Retain。設定為時 Retain、儲存叢集上的基礎實體快照、即使在 VolumeSnapshot 物件已刪除。

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 Volume Snapshot 物件建立一個 pvc1 快照名稱設為 pvc1-snap。Volume Snapshot 類似於 PVC、並與相關聯 VolumeSnapshotContent 代表實際快照的物件。

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

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

- 您可以識別 VolumeSnapshotContent 的物件 pvc1-snap 描述 Volume Snapshot。◦ Snapshot

Content Name 識別提供此快照的 Volume SnapshotContent 物件。 Ready To Use 參數表示快照可用於建立新的 PVC。

```
kubectl describe volumesnapshots pvcl-snap
Name:          pvcl-snap
Namespace:    default
.
.
.
Spec:
  Snapshot Class Name:  pvcl-snap
  Snapshot Content Name: snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:      PersistentVolumeClaim
    Name:      pvcl
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` 使用名為的 Volume Snapshot 建立 PVC `<pvc-name>` 做為資料來源。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"](#)。

以下範例使用建立 PVC `pvcl-snap` 做為資料來源。

```
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: pvcl-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 在 Astra Trident 外部建立的物件和匯入快照。

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 做為 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`。
 - 指定 `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` 在中 `snapshotHandle`。這是中外部快照機提供給 Astra Trident 的唯一資訊 `ListSnapshots` 致電：



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

範例

下列範例建立 VolumeSnapshotContent 參照後端快照的物件 `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 參照的 CR VolumeSnapshotContent 物件：這會要求存取權以使用 VolumeSnapshot 在指定的命名空間中。

範例

下列範例建立 VolumeSnapshot CR 命名 import-snap 這是參考的 VolumeSnapshotContent 已命名 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（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 致電：Astra Trident 會建立 TridentSnapshot。
 - 外部快照器會設定 VolumeSnapshotContent 至 readyToUse 和 VolumeSnapshot 至 true。
 - Trident 退貨 readyToUse=true。
4. * 任何使用者：* 建立 PersistentVolumeClaim 以參考新的 VolumeSnapshot、其中 spec.dataSource（或 spec.dataSourceRef）名稱為 VolumeSnapshot 名稱。

範例

下列範例建立一個 PVC 參照 VolumeSnapshot 已命名 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 Volume 資料

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

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、以協助使用進行資源配置的磁碟區達到最大相容性 `ontap-nas` 和 `ontap-nas-economy` 驅動程式：啟用 `.snapshot` 直接從快照恢復資料的目錄。



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。本示例為 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: pvcl
    volumeSnapshotName: pvcl-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。

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
```

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` 和更新 namespace 到您的命名空間。

相關連結

- ["Volume快照"](#)
- ["Volume SnapshotClass"](#)

版權資訊

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

由 NetApp 版權資料衍伸之軟體必須遵守下列授權和免責聲明：

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

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

本手冊所述之產品受到一項（含）以上的美國專利、國外專利或申請中專利所保障。

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

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

商標資訊

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