執行Volume作業

Astra Trident

NetApp
November 14, 2025

This PDF was generated from https://docs.netapp.com/zh-tw/trident-2301/trident-use/csi-topology.html on November 14, 2025. Always check docs.netapp.com for the latest.

目錄

執行Volume作業 1
使用「csi拓撲」 1
步驟1：建立可感知拓撲的後端 2
步驟2：定義可感知拓撲的StorageClass 4
步驟3：建立並使用PVC 5
更新後端以納入 `supportedTopologies` 8
如需詳細資訊、請參閱 8
使用快照 8
步驟1：建立 `VolumeSnapshotClass` 8
步驟2：建立現有PVc的快照 9
步驟3：從Volume Snapshot建立PVCS 10
部署Volume Snapshot控制器 11
相關連結 11
展開Volume 12
展開iSCSI Volume 12
展開NFS Volume 16
匯入磁碟區 19
支援Volume匯入的驅動程式 19
為什麼要匯入磁碟區？ 20
匯入如何運作？ 20
`ontap-nas` 和 `ontap-nas-flexgroup` 匯入 21
`ontap-san` 匯入 22
`element` 匯入 23
`gcp-cvs` 匯入 23
`azure-netapp-files` 匯入 24

執行Volume作業

深入瞭解Astra Trident提供的功能、協助您管理磁碟區。

- "使用「csi拓撲」"
- "使用快照"
- "展開Volume"
- "匯入磁碟區"

使用「csi拓撲」

Astra Trident可以利用、選擇性地建立磁碟區、並將磁碟區附加至Kubernetes叢集中的節點 "「csi拓撲」功能"。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。

> 深入瞭解「csi拓撲」功能 "請按這裡"。

Kubernetes提供兩種獨特的Volume繫結模式：

- 與 `VolumeBindingMode` 設定為 `Immediate`Astra Trident在沒有任何拓撲感知的情況下建立磁碟區。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` 適用於未強制拓撲限制的叢集。建立永續性磁碟區時、不會對要求的Pod排程需求有任何相依性。
- 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","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-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
  region: Iowa-DC
  zone: Iowa-DC-A
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
  region: Iowa-DC
  zone: Iowa-DC-B
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
```

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

```
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

您可以建立持續磁碟區（PV）的Kubernetes Volume Snapshot（Volume Snapshot）、以維護Astra Trident磁碟區的時間點複本。此外、您也可以從現有的Volume Snapshot建立新的Volume、也稱為_clon__。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）。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。

步驟1：建立 `VolumeSnapshotClass`

此範例會建立Volume Snapshot類別。

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

。 `driver` 指向Astra Trident的SCSI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。

如需詳細資訊、請參閱連結：../ Trident參考/ objects.html#Kubernetes-volumesnapshotclass-objects[`VolumeSnapshotClass`]。

步驟2：建立現有PVc的快照

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
```

在此範例中、快照是針對名為的PVc建立 `pvc1` 快照名稱設為 `pvc1-snap`。

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

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

這會建立一個 `VolumeSnapshot` 物件：Volume Snapshot類似於PVC、並與相關聯 `VolumeSnapshotContent` 代表實際快照的物件。

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

```
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
.
.
```

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

步驟3：從Volume Snapshot建立PVCS

此範例使用快照建立一個PVC..

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

`dataSource` 顯示必須使用名為的Volume Snapshot建立PVc `pvc1-snap` 做為資料來源。這會指示Astra Trident從快照建立一個永久虛擬資料。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。

> 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」。若要刪除Astra Trident磁碟區、則應移除該磁碟區的快照。

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. 在所需的命名空間中建立Snapshot控制器。編輯下方的Yaml清單以修改命名空間。

```
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
```

相關連結

- "Volume快照"
- "Volume SnapshotClass"

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。

總覽

擴充iSCSI PV包括下列步驟：

- 編輯StorageClass定義以設定 `allowVolumeExpansion` 欄位至 `true`。
- 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。
- 必須將PV附加至Pod、才能調整其大小。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。

以下範例顯示擴充iSCSI PV的運作方式。

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

```
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

```
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

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：

```
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 `kubectl 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1GB：

```
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

您可以使用將現有的儲存磁碟區匯入為Kubernetes PV `tridentctl import`。

支援Volume匯入的驅動程式

下表說明支援匯入磁碟區的驅動程式、以及這些磁碟區所引進的版本。

驅動程式	版本
`ontap-nas`	19.04
`ontap-nas-flexgroup`	19.04
`solidfire-san`	19.04
`azure-netapp-files`	19.04

驅動程式	版本
gcp-cvs	19.04
ontap-san	19.04

為什麼要匯入磁碟區？

將Volume匯入Trident的使用案例有多種：

- 容器化應用程式、並重新使用現有的資料集
- 將資料集的複本用於暫時性應用程式
- 重建故障的Kubernetes叢集
- 在災難恢復期間移轉應用程式資料

匯入如何運作？

Volume匯入程序會使用持續磁碟區宣告（PVc）檔案來建立PVc。至少、PVc檔案應包含名稱、命名空間、存取模式及storageClassName欄位、如下例所示。

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

。 `tridentctl` 用戶端用於匯入現有的儲存磁碟區。Trident會持續儲存Volume中繼資料並建立PVc和PV、以匯入Volume。

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

若要匯入儲存磁碟區、請指定包含該磁碟區的Astra Trident後端名稱、以及唯一識別儲存區上該磁碟區的名稱（例如ONTAP FlexVol ：Wsel, Element Volume、CVS Volume path）。儲存磁碟區必須允許讀取/寫入存取、且可由指定的Astra Trident後端存取。。 `-f` 字串引數為必填、並指定Yaml或Json PVc檔案的路徑。

當Astra Trident收到匯入磁碟區要求時、現有的磁碟區大小會在PVc中決定及設定。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。

使用匯入Volume時 `--no-manage` 引數：Trident不會在物件生命週期的PVc或PV上執行任何其他作業。因

為Trident會忽略的PV和PVc事件 `--no-manage` 物件、儲存磁碟區不會在PV刪除時刪除。此外、也會忽略其他作業、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。

Trident 19.07及更新版本可處理PV的附加元件、並在匯入磁碟區時掛載磁碟區。對於使用舊版Astra Trident的匯入、資料路徑不會有任何作業、而且磁碟區匯入不會驗證是否可以掛載磁碟區。如果在匯入磁碟區時發生錯誤（例如、StorageClass不正確）、您可以將PV上的回收原則變更為來恢復 `retain`、刪除PVC和PV、然後重新嘗試Volume匯入命令。

`ontap-nas` 和 `ontap-nas-flexgroup` 匯入

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：

> 若要由Trident匯入某個類型的Rw。ONTAP如果磁碟區是DP類型、則它是SnapMirror目的地磁碟區；在將磁碟區匯入Trident之前、您應該先中斷鏡射關係。

> 。 `ontap-nas` 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。

例如、匯入名為的磁碟區 `managed_volume` 在名為的後端上 `ontap_nas`，請使用下列命令：

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

匯入名為的磁碟區 `unmanaged_volume` （在上 `ontap_nas backend`）（Trident無法管理）、請使用下列命令：

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

使用時 `--no-manage` 引數：Trident不會重新命名磁碟區、也不會驗證磁碟區是否已掛載。如果未手動掛載磁碟區、則磁碟區匯入作業會失敗。

> 已修正先前使用自訂Unix權限 匯入磁碟區的錯誤。您可以在您的PVc定義或後端組態中指定unixPermissions、並指示Astra Trident依此匯入磁碟區。

`ontap-san` 匯入

Astra Trident也能匯入ONTAP 包含單一LUN的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。您可以使用 `tridentctl import` 命令的方式與其他情況相同：

- 包括的名稱 `ontap-san` 後端：
- 請提供FlexVol 需要匯入的名稱。請記住FlexVol 、這個功能只包含一個必須匯入的LUN。
- 提供必須搭配使用的PVc定義路徑 `-f` 旗標。
- 您可以選擇管理或不受管理的永久虛擬網路。根據預設、Trident會管理PVc、並在FlexVol 後端重新命名該LUN。若要匯入為未受管理的Volume、請傳遞 `--no-manage` 旗標。

> 匯入未受管理的時 `ontap-san` Volume中的LUN FlexVol 名稱 `lun0` 並對應至具有所需啟動器的igroup。Astra Trident會自動處理這項作業、以便進行託管匯入。

然後Astra Trident會匯入FlexVol 該等物件、並將其與PVc定義建立關聯。Astra Trident也將FlexVol 該等功能重新命名為 `pvc-<uuid>` 格式化及FlexVol LUN在功能區內 `lun0`。

> 建議匯入沒有現有作用中連線的磁碟區。如果您要匯入使用中的Volume、請先複製該Volume、然後再執行匯入。

範例

以匯入 `ontap-san-managed` 上的顯示FlexVol `ontap_san_default` 後端、執行 `tridentctl import` 命令形式：

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

若要由Astra Trident匯入某個類型的RW磁碟區。ONTAP如果磁碟區為DP類型、則為SnapMirror目的地磁碟區；您應該先中斷鏡射關係、再將磁碟區匯入Astra Trident。

`element` 匯入

您可以使用NetApp Element Trident將支援功能的軟體/NetApp HCI磁碟區匯入Kubernetes叢集。您需要Astra Trident後端的名稱、以及磁碟區的唯一名稱和PVc檔案做為的引數 `tridentctl import` 命令。

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

Element驅動程式支援重複的Volume名稱。如果有重複的Volume名稱、Trident的Volume匯入程序會傳回錯誤。因應措施是複製磁碟區、並提供唯一的磁碟區名稱。然後匯入複製的Volume。

`gcp-cvs` 匯入

若要匯入以NetApp Cloud Volumes Service 支援的GCP磁碟區、請使用磁碟區路徑來識別該磁碟區、而非其名稱。

若要匯入 `gcp-cvs` 後端上的Volume稱為 `gcpcvs_YEppr` 的磁碟區路徑 `adroit-jolly-swift`，請使用下列命令：

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

> Volume路徑是Volume匯出路徑的一部分、位於：/之後。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`。

`azure-netapp-files` 匯入

若要匯入 `azure-netapp-files` 後端上的Volume稱為 `azurenetappfiles_40517` 磁碟區路徑 `importvol1`，執行下列命令：

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

> anf磁碟區的磁碟區路徑會出現在裝載路徑中的：/之後。例如、如果掛載路徑為 `10.0.0.2:/importvol1`、磁碟區路徑為 `importvol1`。

版權資訊

Copyright © 2025 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。