# 使用Trident

Trident

NetApp
January 15, 2026

This PDF was generated from https://docs.netapp.com/zh-tw/trident-2506/trident-use/worker-node-prep.html on January 15, 2026. Always check docs.netapp.com for the latest.

# 目錄

# 使用Trident

## 準備工作節點

Kubernetes 叢集中的所有工作節點都必須能夠掛載您為 Pod 設定的磁碟區。若要準備工作節點，您必須根據所選驅動程式安裝 NFS、iSCSI、NVMe/TCP 或 FC 工具。

### 選擇合適的工具

如果您使用的是多種驅動程序，則應安裝所有驅動程式所需的工具。最新版本的 Red Hat Enterprise Linux CoreOS (RHCOS) 預設安裝了這些工具。

**NFS** 工具

"安裝 NFS 工具"如果您正在使用： `ontap-nas` ， `ontap-nas-economy` ， `ontap-nas-flexgroup` ， `azure-netapp-files` ， `gcp-cvs` 。

**iSCSI** 工具

"安裝 iSCSI 工具"如果您正在使用： `ontap-san` ， `ontap-san-economy` ， `solidfire-san` 。

**NVMe** 工具

"安裝 NVMe 工具"如果你正在使用 `ontap-san`用於基於 TCP 的非揮發性記憶體高速介面 (NVMe) (NVMe/TCP) 協定。

NetApp建議 NVMe/TCP 使用ONTAP 9.12 或更高版本。

透過光纖通道 **(FC)** 進行 **SCSI** 的工具

請參閱"配置 FC 和 FC-NVMe SAN 主機的方法"有關配置 FC 和 FC-NVMe SAN 主機的詳細資訊。

"安裝 FC 工具"如果你正在使用 `ontap-san`使用 `sanType` `fcp` （透過光纖通道進行 SCSI 通訊）。

需要考慮的要點：* OpenShift 和 KubeVirt 環境支援透過 FC 進行 SCSI 通訊。 * Docker 不支援透過 FC 傳輸 SCSI。 * iSCSI 自癒功能不適用於基於 FC 的 SCSI。

### 節點服務發現

Trident會嘗試自動偵測節點是否可以執行 iSCSI 或 NFS 服務。

節點服務發現功能可以辨識已發現的服務，但無法保證服務配置正確。反之，未發現服務並不保證卷掛載一定會失敗。

回顧事件

Trident會為節點建立事件，以識別已發現的服務。要查看這些事件，請運行：

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

查看已發現的服務

Trident識別Trident節點 CR 上每個節點啟用的服務。若要查看已發現的服務，請執行：

```
tridentctl get node -o wide -n <Trident namespace>
```

## NFS卷

使用適用於您作業系統的命令安裝 NFS 工具。確保NFS服務在啟動時啟動。

**RHEL 8+**

```
sudo yum install -y nfs-utils
```

**Ubuntu**

```
sudo apt-get install -y nfs-common
```

安裝 NFS 工具後重新啟動工作節點，以防止將磁碟區附加到容器時發生故障。

## iSCSI 卷

Trident可以自動建立 iSCSI 會話、掃描 LUN、發現多路徑裝置、格式化裝置並將其掛載到 pod 中。

### iSCSI自癒能力

對於ONTAP系統， Trident每五分鐘運行一次 iSCSI 自癒程序，以：

1. *確定*所需的 iSCSI 會話狀態和目前的 iSCSI 會話狀態。
2. 將理想狀態與目前狀態進行比較，以確定需要進行的維修。 Trident確定維修優先順序以及何時進行搶修。
3. 執行必要的修復，使目前的 iSCSI 會話狀態恢復到所需的 iSCSI 會話狀態。

自癒活動的日誌位於… `trident-main`對應 Daemonset pod 上的容器。要查看日誌，您必須已設定 `debug`在Trident安裝過程中設定為「true」。

Trident iSCSI 的自癒功能可以幫助預防：

- 網路連線問題後可能出現過期或不健康的 iSCSI 會話。如果會話已過期， Trident會等待七分鐘，然後登出並重新與入口網站建立連線。

例如，如果儲存控制器上輪換了 CHAP 金鑰，並且網路失去連接，則舊的（過時的）CHAP 金鑰可能會保留下來。自癒功能可以識別這一點，並自動重新建立會話以套用更新後的 CHAP 金鑰。

- 缺少 iSCSI 會話
- 缺少 LUN

升級**Trident**之前需要考慮的要點

- 如果僅使用每個節點的 igroup（在 23.04+ 中引入），則 iSCSI 自癒功能將啟動 SCSI 總線上所有裝置的 SCSI 重新掃描。
- 如果僅使用後端範圍的 igroup（自 23.04 版本起已棄用），則 iSCSI 自癒功能將啟動 SCSI 重新掃描，以尋找 SCSI 總線上的確切 LUN ID。
- 如果同時使用節點級 igroup 和後端級 igroup，iSCSI 自癒功能將啟動 SCSI 重新掃描，以尋找 SCSI 總線上的精確 LUN ID。

安裝 **iSCSI** 工具

使用適用於您作業系統的指令安裝 iSCSI 工具。

開始之前

- Kubernetes 叢集中的每個節點都必須有一個唯一的 IQN。這是必要的前提條件。
- 如果使用 RHCOS 4.5 或更高版本，或其他與 RHEL 相容的 Linux 發行版，則需要執行以下操作：`solidfire-san`對於驅動程式和`Element OS 12.5` 或更早版本，請確保將 `CHAP` 驗證演算法設定為 `MD5`。 `/etc/iscsi/iscsid.conf` Element 12.7 提供符合 FIPS 標準的 CHAP 安全演算法 SHA1 、SHA-256 和 SHA3-256。

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- 當使用執行 RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) 且具有 iSCSI PV 的工作節點時，請指定 `discard`StorageClass 中的 mountOption 用於執行內嵌空間回收。參考 "紅帽文檔"。
- 請確保您已升級至最新版本。 `multipath-tools` 。

**RHEL 8+**

1. 安裝以下系統軟體包：

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-
multipath
```

2. 請檢查 iscsi-initiator-utils 版本是否為 6.2.0.874-2.el7 或更高版本：

```
rpm -q iscsi-initiator-utils
```

3. 將掃描方式設定為手動：

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. 啟用多路徑：

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> 確保 `/etc/multipath.conf`包含 `find_multipaths no`在下面 `defaults`。

5. 確保 `iscsid`和 `multipathd`正在運行：

```
sudo systemctl enable --now iscsid multipathd
```

6. 啟用並啟動 `iscsi`：

```
sudo systemctl enable --now iscsi
```

**Ubuntu**

1. 安裝以下系統軟體包：

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. 請檢查 open-iscsi 版本是否為 2.0.874-5ubuntu2.10 或更高版本（適用於 bionic）或 2.0.874-7.1ubuntu6.1 或更高版本（適用於 focal）：

```
dpkg -l open-iscsi
```

3. 將掃描方式設定為手動：

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. 啟用多路徑：

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> 確保 /etc/multipath.conf`包含 `find_multipaths no`在下面 `defaults。

5. 確保 `open-iscsi`和 `multipath-tools`已啟用並正在運行：

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

> 對於 Ubuntu 18.04，您必須使用以下命令發現目標連接埠： `iscsiadm`開始之前 `open-iscsi`啟動 iSCSI 守護程式。您也可以修改 `iscsi`服務啟動 `iscsid`自動地。

配置或停用 **iSCSI** 自愈

您可以設定以下Trident iSCSI 自癒設定來修復過期的會話：

- **iSCSI** 自癒間隔：決定 iSCSI 自癒的呼叫頻率（預設值：5 分鐘）。你可以透過設定較小的數字來增加運作頻率，或是設定較大的數字來降低運作頻率。

> 將 iSCSI 自癒間隔設為 0 將完全停止 iSCSI 自癒功能。我們不建議停用 iSCSI 自癒功能；只有在 iSCSI 自癒功能無法如預期運作或出於除錯目的時，才應停用該功能。

- **iSCSI** 自癒等待時間：確定 iSCSI 自癒在登出不健康的會話並嘗試再次登入之前等待的時間（預設值：7 分

鐘）。您可以將其配置為更大的數字，以便被識別為不健康的會話必須等待更長時間才能登出，然後再嘗試重新登入；或配置為較小的數字，以便更快地登出和登入。

舵

若要配置或變更 iSCSI 自癒設置，請傳遞以下參數： `iscsiSelfHealingInterval`和 `iscsiSelfHealingWaitTime`Helm 安裝或 Helm 更新期間的參數。

以下範例將 iSCSI 自癒間隔設定為 3 分鐘，自癒等待時間設定為 6 分鐘：

```
helm install trident trident-operator-100.2506.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

三叉戟

若要配置或變更 iSCSI 自癒設置，請傳遞以下參數： `iscsi-self-healing-interval`和 `iscsi-self-healing-wait-time`tridentctl 安裝或更新期間的參數。

以下範例將 iSCSI 自癒間隔設定為 3 分鐘，自癒等待時間設定為 6 分鐘：

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

## NVMe/TCP 卷

使用適用於您作業系統的指令安裝 NVMe 工具。

- NVMe 需要 RHEL 9 或更高版本。
- 如果您的 Kubernetes 節點的核心版本太舊，或者您的核心版本沒有 NVMe 軟體包，則您可能需要將節點的核心版本更新為包含 NVMe 軟體包的版本。

**RHEL 9**

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

**Ubuntu**

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

驗證安裝

安裝完成後，使用以下命令驗證 Kubernetes 叢集中的每個節點是否具有唯一的 NQN：

```
cat /etc/nvme/hostnqn
```

Trident修改了 `ctrl_device_tmo`確保 NVMe 在路徑中斷時不會放棄路徑的值。請勿更改此設定。

## SCSI over FC 卷

現在您可以使用光纖通道 (FC) 協定和Trident在ONTAP系統上設定和管理儲存資源。

先決條件

配置 FC 所需的網路和節點設定。

網路設定

1. 取得目標介面的 WWPN。請參閱 "網路介面顯示"了解更多。
2. 取得發起方（主機）上介面的 WWPN。

   請參考對應的主機作業系統實用程式。

3. 使用主機和目標的 WWPN 在 FC 交換器上設定區域。

   有關信息，請參閱相應交換機供應商的文檔。

   詳情請參閱以下ONTAP文件：

   - "光纖通道和FCoE分區概述"
   - "配置 FC 和 FC-NVMe SAN 主機的方法"

安裝 **FC** 工具

使用適用於您作業系統的命令安裝 FC 工具。

- 當使用執行 RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) 且具有 FC PV 的工作節點時，請指定 `discard`StorageClass 中的 mountOption 用於執行內嵌空間回收。參考 "紅帽文檔"。

**RHEL 8+**

1. 安裝以下系統軟體包：

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. 啟用多路徑：

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> 確保 /etc/multipath.conf`包含 `find_multipaths no`在下面 `defaults。

3. 確保 `multipathd`正在運行：

```
sudo systemctl enable --now multipathd
```

**Ubuntu**

1. 安裝以下系統軟體包：

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. 啟用多路徑：

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> 確保 /etc/multipath.conf`包含 `find_multipaths no`在下面 `defaults。

3. 確保 `multipath-tools`已啟用並正在運行：

```
sudo systemctl status multipath-tools
```

# 設定和管理後端

## 配置後端

後端定義了Trident與儲存系統之間的關係。它告訴Trident如何與該儲存系統通信，以及Trident應該如何從中設定磁碟區。

Trident會自動提供符合儲存類別定義要求的後端儲存池。了解如何配置儲存系統的後端。

- "設定Azure NetApp Files後端"
- "設定Google Cloud NetApp Volumes後端"
- "為 Google Cloud Platform 後端設定Cloud Volumes Service"
- "配置NetApp HCI或SolidFire後端"
- "使用ONTAP或Cloud Volumes ONTAP NAS 驅動程式設定後端"
- "使用ONTAP或Cloud Volumes ONTAP SAN 驅動程式設定後端"
- "將Trident與Amazon FSx for NetApp ONTAP"

### Azure NetApp Files

#### 設定**Azure NetApp Files**後端

您可以將Azure NetApp Files設定為Trident的後端。您可以使用Azure NetApp Files後端附加 NFS 和 SMB 磁碟區。 Trident也支援使用託管識別碼對 Azure Kubernetes 服務 (AKS) 叢集進行憑證管理。

##### **Azure NetApp Files**驅動程式詳細信息

Trident提供以下Azure NetApp Files儲存驅動程序，以便與叢集通訊。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

| 司機 | 協定 | 音量模式 | 支援的存取模式 | 支援的檔案系統 |
|---|---|---|---|---|
| `azure-netapp-files` | NFS SMB | 檔案系統 | RWO、ROX、RWX、RWOP | `nfs`， `smb` |

注意事項

- Azure NetApp Files服務不支援小於 50 GiB 的磁碟區。如果要求的磁碟區較小， Trident會自動建立 50GiB 的磁碟區。
- Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。

##### 為 **AKS** 管理身份

Trident支持"託管身分"適用於 Azure Kubernetes 服務叢集。若要利用託管身分提供的簡化憑證管理功能，您必須具備以下條件：

- 使用 AKS 部署的 Kubernetes 集群
- 在 AKS Kubernetes 叢集上配置的託管身份
- 已安裝的Trident包括 `cloudProvider`指定 `"Azure"`。

**Trident操作員**

若要使用Trident操作員安裝Trident ，請編輯 `tridentorchestrator_cr.yaml`設定 `cloudProvider`到 `"Azure"`。例如：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

舵

以下範例安裝Trident套裝 `cloudProvider`使用環境變數連接到 `Azure` `$CP`：

```
helm install trident trident-operator-100.2506.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

**<code>tridentctl</code>**

以下範例安裝Trident並進行設置 `cloudProvider`標記 `Azure`：

```
tridentctl install --cloud-provider="Azure" -n trident
```

#### AKS 的雲端身份

雲端身分使 Kubernetes pod 能夠透過作為工作負載身分進行驗證來存取 Azure 資源，而無需提供明確的 Azure 憑證。

若要在 Azure 中利用雲端身分功能，您必須具備以下條件：

- 使用 AKS 部署的 Kubernetes 集群
- 在 AKS Kubernetes 叢集上設定工作負載身分和 oidc-issuer
- 已安裝的Trident包括 `cloudProvider`指定 `"Azure"`和 `cloudIdentity`指定工作負載標識

**Trident操作員**

若要使用Trident操作員安裝Trident ，請編輯 `tridentorchestrator_cr.yaml`設定 `cloudProvider`到 `"Azure"`並設定 `cloudIdentity`到 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`。

例如：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx' # Edit
```

舵

使用下列環境變數設定 **cloud-provider (CP)** 和 **cloud-identity (CI)** 標誌的值：

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx'"
```

以下範例安裝Trident並進行設置 `cloudProvider`使用環境變數連接到 `Azure` `$CP`並設定 `cloudIdentity`使用環境變數 `$CI`：

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

**<code>tridentctl</code>**

請使用以下環境變數設定 雲端提供者 和 雲端身分 標誌的值：

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx"
```

以下範例安裝Trident並進行設置 `cloud-provider`標記 `$CP`，和 `cloud-identity`到 `$CI`：

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

# 在設定Azure NetApp Files後端之前，需要確保滿足以下要求。

### NFS 和 SMB 磁碟區的先決條件

如果您是第一次使用Azure NetApp Files或在新的位置使用，則需要進行一些初始設定來設定 Azure NetApp檔案並建立 NFS 磁碟區。參考 "Azure：設定Azure NetApp Files並建立 NFS 卷"。

配置和使用 "Azure NetApp Files"後端需要以下組件：

> - `subscriptionID`, `tenantID`, `clientID`, `location`, 和 `clientSecret`在 AKS 叢集上使用託管身分時，這些是可選的。
> - `tenantID`, `clientID`, 和 `clientSecret`在 AKS 叢集上使用雲端身分時，這些是可選的。

- 容量池。參考"微軟：為Azure NetApp Files建立容量池"。
- 委派給Azure NetApp Files 的子網路。參考"Microsoft：將子網路委派給Azure NetApp Files"。
- `subscriptionID`來自已啟用Azure NetApp Files功能的 Azure 訂閱。
- `tenantID`, `clientID`, 和 `clientSecret`來自"應用程式註冊"在 Azure Active Directory 中擁有對Azure NetApp Files服務的足夠權限。應用程式註冊應使用以下任一方式：
  - 所有者或貢獻者角色"Azure 預定義"。
  - 一個"自訂貢獻者角色"訂閱等級(`assignableScopes`）但權限僅限於Trident所需的權限。建立自訂角色後，"使用 Azure 入口網站指派角色" 。

自訂貢獻者角色

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
```

```
ions/delete",
          "Microsoft.Features/features/read",
          "Microsoft.Features/operations/read",
          "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
        ],
        "notActions": [],
        "dataActions": [],
        "notDataActions": []
      }
    ]
  }
}
```

- 蔚藍 `location`包含至少一個 "委託子網"。截至Trident 22.01版本， `location`參數是後端設定檔頂層的一個必填欄位。虛擬池中指定的位置值將被忽略。
- 使用 `Cloud Identity`得到 `client ID`從一個 "使用者指派的託管身份"並指定該 `ID azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx` 。

中小企業卷的附加要求

若要建立 SMB 卷，您必須具備以下條件：

- Active Directory 已設定並連線至Azure NetApp Files。參考"Microsoft：建立和管理Azure NetApp Files管理器的 Active Directory 連接"。
- 一個 Kubernetes 叢集，包含一個 Linux 控制器節點和至少一個執行 Windows Server 2022 的 Windows 工作節點。 Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。
- 至少需要一個包含 Active Directory 憑證的Trident金鑰，以便Azure NetApp Files可以向 Active Directory 進行驗證。生成秘密 `smbcreds`：

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- 配置為 Windows 服務的 CSI 代理程式。要配置 `csi-proxy`，請參閱"GitHub：CSI代理"或者"GitHub：適用於 Windows 的 CSI 代理"適用於在 Windows 上執行的 Kubernetes 節點。

**Azure NetApp Files**後端設定選項和範例

了解Azure NetApp Files的 NFS 和 SMB 後端設定選項，並查看設定範例。

Trident使用您的後端設定（子網路、虛擬網路、服務等級和位置），在要求的位置中可用的容量池上建立Azure NetApp Files卷，並與要求的服務等級和子網路相符。

* 從NetApp Trident 25.06 版本開始，手動 QoS 容量池作為技術預覽版受到支援。 *

Azure NetApp Files後端提供以下設定選項。

| 範圍 | 描述 | 預設 |
|---|---|---|
| `version` | | 始終為 1 |
| `storageDriverName` | 儲存驅動程式的名稱 | "azure-netapp-files" |
| `backendName` | 自訂名稱或儲存後端 | 司機姓名 + "_" + 隨機字符 |
| `subscriptionID` | Azure 訂閱的訂閱 ID（在 AKS 叢集上啟用託管識別時為可選）。 | |
| `tenantID` | 在 AKS 叢集上使用託管身分或雲端身分時，應用程式註冊中的租用戶 ID 是可選的。 | |
| `clientID` | 在 AKS 叢集上使用託管身分或雲端身分時，應用程式註冊中的用戶端 ID 是可選的。 | |
| `clientSecret` | 在 AKS 叢集上使用託管身分或雲端身分時，應用程式註冊中的用戶端金鑰是可選的。 | |
| `serviceLevel` | 之一 `Standard`， `Premium` ，或者 `Ultra` | 「「（隨機的） |
| `location` | 將在 Azure 中建立新磁碟區的位置名稱（在 AKS 叢集上啟用託管識別碼時為可選）。 | |
| `resourceGroups` | 用於篩選已發現資源的資源群組列表 | （無濾鏡） |
| `netappAccounts` | 用於篩選已發現資源的NetApp帳戶列表 | （無濾鏡） |
| `capacityPools` | 用於篩選已發現資源的容量池列表 | （無過濾，隨機） |
| `virtualNetwork` | 具有委派子網路的虛擬網路的名稱 | “” |
| `subnet` | 委派給的子網路名稱 `Microsoft.Netapp/volumes` | “” |
| `networkFeatures` | 卷的 VNet 功能集，可能是 `Basic`或者 `Standard`。網路功能並非在所有地區都可用，可能需要透過訂閱才能啟用。指定`networkFeatures`未啟用該功能會導致磁碟區配置失敗。 | “” |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| `nfsMountOptions` | 對 NFS 掛載選項進行精細控制。對於 SMB 卷，此設定將被忽略。若要使用 NFS 版本 4.1 掛載卷，請包含以下內容 `nfsvers=4`在以逗號分隔的掛載選項清單中選擇 NFS v4.1。儲存類別定義中設定的掛載選項會覆蓋後端配置中設定的掛載選項。 | "nfsvers=3" |
| `limitVolumeSize` | 如果請求的磁碟區大小大於此值，則配置失敗。 | （預設不強制執行） |
| `debugTraceFlags` | 故障排除時要使用的調試標誌。例子， `\{"api": false, "method": true, "discovery": true}` 。除非您正在進行故障排除並需要詳細的日誌轉儲，否則請勿使用此功能。 | 無效的 |
| `nasType` | 配置 NFS 或 SMB 磁碟區的建立。選項有 `nfs`， `smb`或空值。設定為 null 則預設使用 NFS 磁碟區。 | `nfs` |
| `supportedTopologies` | 表示從後端支援的區域和區域列表。更多信息，請參閱"使用 CSI 拓撲"。 | |
| `qosType` | 表示 QoS 類型：自動或手動。* Trident 25.06 技術預覽版* | 汽車 |
| `maxThroughput` | 設定允許的最大吞吐量，單位為 MiB/秒。僅支援手動 QoS 容量池。* Trident 25.06 技術預覽版* | `4 MiB/sec` |

有關網絡功能的更多信息，請參閱"為Azure NetApp Files磁碟區設定網路功能"。

所需權限和資源

如果在建立 PVC 時收到「未找到容量池」錯誤，則可能是您的應用程式註冊沒有關聯的必要權限和資源（子網路、虛擬網路、容量池）。如果啟用偵錯功能， Trident將記錄在建立後端時發現的 Azure 資源。請確認是否使用了適當的角色。

值 `resourceGroups`， `netappAccounts` ， `capacityPools` ， `virtualNetwork` ， 和 `subnet`可以使用簡稱或完全限定名稱來指定。大多數情況下建議使用完全限定名稱，因為短名稱可能會符合多個同名資源。

這 `resourceGroups`， `netappAccounts` ， 和 `capacityPools`值是過濾器，用於將發現的資源集限制為該儲存後端可用的資源，並且可以以任意組合指定。完全限定名稱遵循以下格式：

| 類型 | 格式 |
| --- | --- |
| 資源組 | <資源組> |
| NetApp帳戶 | <資源組>/<NetApp 帳號> |
| 容量池 | <資源組>/<NetApp 帳號>/<容量池> |

| 類型 | 格式 |
| --- | --- |
| 虛擬網路 | <資源組>/<虛擬網路> |
| 子網 | <資源群組>/<虛擬網路>/<子網路> |

卷配置

您可以透過在設定檔的特定部分中指定以下選項來控制預設磁碟區配置。參考 [範例配置] 了解詳情。

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| `exportRule` | 新卷的出口規則。 `exportRule`必須是以逗號分隔的 IPv4 位址或 IPv4 子網路的任意組合列表，採用 CIDR 表示法。對於 SMB 卷，此設定將被忽略。 | “0.0.0.0/0” |
| `snapshotDir` | 控制 .snapshot 目錄的可見性 | NFSv4 為“true”，NFSv3 為“false”。 |
| `size` | 新磁碟區的預設大小 | 100G |
| `unixPermissions` | 新磁碟區的 Unix 權限（4 位八進位數字）。對於 SMB 卷，此設定將被忽略。 | （預覽功能，需訂閱並加入白名單） |

範例配置

以下範例展示了基本配置，其中大多數參數都保留預設值。這是定義後端最簡單的方法。

最小配置

這是最基本的後端配置。透過此配置， Trident會發現配置位置中所有委派給Azure NetApp Files儲存的NetApp帳戶、容量池和子網，並將新磁碟區隨機放置在其中一個池和子網路上。因為 `nasType`省略了 `nfs`預設設定生效，後端將為 NFS 磁碟區進行設定。

如果您剛開始使用Azure NetApp Files並進行嘗試，此配置是理想的選擇，但在實踐中，您需要為預配的磁碟區提供額外的範圍。

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
  tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
  clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
  clientSecret: SECRET
  location: eastus
```

為 **AKS** 管理身份

此後端配置省略了 `subscriptionID`， `tenantID` ， `clientID` ， 和 `clientSecret`在使用託管身分時，這些是可選的。

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
```

**AKS 的雲端身份**

此後端配置省略了 `tenantID`， `clientID` ， 和 `clientSecret`在使用雲端身分時，這些是可選的。

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

具有容量池過濾器的特定服務等級配置

此後端配置將磁碟區放置在 Azure 中 `eastus`位置 `Ultra`容量池。 Trident會自動發現該位置中委派給Azure NetApp Files的所有子網，並隨機在其中一個子網路上放置一個新磁碟區。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
```

具有手動 **QoS** 容量池的後端範例

此後端配置將磁碟區放置在 Azure 中 `eastus`具有手動 QoS 容量池的位置。* NetApp Trident 25.06 中的技術預覽*。

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

此後端配置進一步縮小了磁碟區放置範圍到單一子網，並且還修改了一些磁碟區配置預設值。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
virtualNetwork: my-virtual-network
subnet: my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: "true"
  size: 200Gi
  unixPermissions: "0777"
```

虛擬池配置

此後端配置在單一檔案中定義了多個儲存池。當您有多個容量池支援不同的服務級別，並且想要在 Kubernetes 中建立代表這些級別的儲存類別時，這將非常有用。虛擬池標籤用於根據以下因素區分池子： `performance` 。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
  - application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
  - labels:
      performance: gold
    serviceLevel: Ultra
    capacityPools:
      - ultra-1
      - ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - standard-1
      - standard-2
```

支援的拓撲配置

Trident可根據區域和可用區為工作負載提供磁碟區。這 `supportedTopologies`此後端配置中的區塊用於提供每個後端的區域和區域清單。此處指定的區域和區域值必須與每個 Kubernetes 叢集節點上的標籤中的區域和區域值相符。這些區域和分區代表儲存類別中可以提供的允許值的清單。對於包含後端提供的區域和可用區子集的儲存類， Trident會在所述區域和可用區中建立磁碟區。更多信息，請參閱"使用 CSI 拓撲"。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

儲存類別定義

下列 `StorageClass`上述定義指的是儲存池。

使用範例定義 **`parameter.selector`**場地

使用 `parameter.selector`你可以為每個物件指定。 `StorageClass`用於託管磁碟區的虛擬池。該磁碟區將具有所選池中定義的方面。

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze
allowVolumeExpansion: true
```

**SMB** 磁碟區的範例定義

使用 `nasType` ， `node-stage-secret-name` ， 和 `node-stage-secret-namespace`您可以指定 SMB 磁碟區並提供所需的 Active Directory 憑證。

預設命名空間上的基本配置

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

每個命名空間使用不同的密鑰

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

每個磁碟區使用不同的密鑰

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`篩選支援 SMB 磁碟區的儲存池。 `nasType: nfs`或者 `nasType: null`NFS池過濾器。

創建後端

建立後端設定檔後，執行以下命令：

```
tridentctl create backend -f <backend-file>
```

如果後端建立失敗，則後端配置存在問題。您可以透過執行以下命令查看日誌以確定原因：

```
tridentctl logs
```

在您發現並修正設定檔中的問題後，您可以再次執行建立命令。

## Google Cloud NetApp Volumes

### 設定Google Cloud NetApp Volumes後端

現在您可以將Google Cloud NetApp Volumes設定為Trident的後端。您可以使用Google Cloud NetApp Volumes後端來附加 NFS 和 SMB 磁碟區。

#### Google Cloud NetApp Volumes驅動程式詳情

Trident提供 `google-cloud-netapp-volumes`驅動程式與集群通訊。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

| 司機 | 協定 | 音量模式 | 支援的存取模式 | 支援的檔案系統 |
|---|---|---|---|---|
| `google-cloud-netapp-volumes` | NFS SMB | 檔案系統 | RWO、ROX、RWX、RWOP | `nfs， smb` |

#### GKE 的雲端身份

雲端身分使 Kubernetes pod 能夠透過作為工作負載身分進行驗證來存取 Google Cloud 資源，而無需提供明確的 Google Cloud 憑證。

若要在 Google Cloud 中利用雲端身分功能，您必須具備以下條件：

- 使用 GKE 部署的 Kubernetes 叢集。
- 在 GKE 叢集上配置工作負載標識，並在節點池上配置 GKE 元資料伺服器。
- 具有Google Cloud NetApp Volumes管理員 (roles/netapp.admin) 角色或自訂角色的 GCP 服務帳戶。
- Trident已安裝，其中包括 cloudProvider（指定「GCP」）和 cloudIdentity（指定新的 GCP 服務帳戶）。下面給出一個例子。

**Trident操作員**

若要使用Trident操作員安裝Trident ，請編輯 `tridentorchestrator_cr.yaml`設定 `cloudProvider`到 `"GCP"`並設定 `cloudIdentity`到 `iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com`。

例如：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "GCP"
  cloudIdentity: 'iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com'
```

舵

使用下列環境變數設定 **cloud-provider (CP)** 和 **cloud-identity (CI)** 標誌的值：

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com'"
```

以下範例安裝Trident並進行設置 `cloudProvider`使用環境變數連接到 `GCP` `$CP`並設定 `cloudIdentity`使用環境變數 `$ANNOTATION`：

```
helm install trident trident-operator-100.6.0.tgz --set cloudProvider=$CP --set cloudIdentity="$ANNOTATION"
```

**<code>tridentctl</code>**

請使用以下環境變數設定 雲端提供者 和 雲端身分 標誌的值：

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com'"
```

以下範例安裝Trident並進行設置 `cloud-provider`標記 `$CP`，和 `cloud-identity`到 `$ANNOTATION`：

```
tridentctl install --cloud-provider=$CP --cloud-identity="$ANNOTATION" -n trident
```

準備設定**Google Cloud NetApp Volumes**後端

# 在設定Google Cloud NetApp Volumes後端之前，您需要確保符合下列要求。

**NFS** 磁碟區的先決條件

如果您是第一次使用Google Cloud NetApp Volumes或在新的位置使用，則需要進行一些初始設定來設定Google Cloud NetApp Volumes並建立 NFS 磁碟區。參考"開始之前"。

在設定Google Cloud NetApp Volumes後端之前，請確保您已具備以下條件：

- 已設定Google Cloud NetApp Volumes服務的 Google Cloud 帳戶。參考"Google Cloud NetApp Volumes"。
- 您的 Google Cloud 帳戶項目編號。參考"確定項目"。
- 擁有NetApp Volumes 管理員權限的 Google Cloud 服務帳戶(`roles/netapp.admin`） 角色。參考"身分和存取管理角色和權限"。
- GCNV帳戶的API金鑰檔案。請參閱"建立服務帳戶金鑰"
- 儲水池。參考"儲存池概覽"。

有關如何設定對Google Cloud NetApp Volumes 的存取權限的更多信息，請參閱："設定對Google Cloud NetApp Volumes 的存取權限" 。

**Google Cloud NetApp Volumes**後端設定選項和範例

# 了解Google Cloud NetApp Volumes的後端設定選項並查看設定範例。

後端配置選項

每個後端都在單一 Google Cloud 區域中配置磁碟區。若要在其他區域建立卷，您可以定義其他後端。

| 範圍 | 描述 | 預設 |
|---|---|---|
| `version` | | 始終為 1 |
| `storageDriverName` | 儲存驅動程式的名稱 | 價值 `storageDriverName` 必須指定為“google-cloud-netapp-volumes”。 |
| `backendName` | （可選）儲存後端自訂名稱 | 驅動程式名稱 + "_" + API 金鑰的一部分 |
| `storagePools` | 用於指定磁碟區建立儲存池的可選參數。 | |
| `projectNumber` | Google Cloud 帳戶項目編號。該值可在 Google Cloud 入口網站首頁找到。 | |
| `location` | Trident建立 GCNV 磁碟區的 Google Cloud 位置。建立跨區域 Kubernetes 叢集時，在下列位置建立的磁碟區： `location`可用於跨多個 Google Cloud 區域的節點上調度的工作負載。跨區域運輸會產生額外費用。 | |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| apiKey | 用於 Google Cloud 服務帳戶的 API 金鑰 netapp.admin`角色。它包含 Google Cloud 服務帳戶私鑰檔案的 JSON 格式內容（原封不動地複製到後端設定檔中）。這 `apiKey`必須包含以下鍵的鍵值對： `type ， project_id ， client_email ， client_id ， auth_uri ， token_uri ， auth_provider_x509_cert_url ，和 client_x509_cert_url。 | |
| nfsMountOptions | 對 NFS 掛載選項進行精細控制。 | "nfsvers=3" |
| limitVolumeSize | 如果請求的磁碟區大小大於此值，則配置失敗。 | （預設不強制執行） |
| serviceLevel | 儲存池的服務等級及其容量。這些值是 flex，standard ， premium ， 或者 extreme。 | |
| labels | 若要套用於磁碟區的任意 JSON 格式標籤集 | “” |
| network | Google Cloud 網路用於 GCNV 磁碟區。 | |
| debugTraceFlags | 故障排除時要使用的調試標誌。例子，{"api":false, "method":true} 。除非您正在進行故障排除並需要詳細的日誌轉儲，否則請勿使用此功能。 | 無效的 |
| nasType | 配置 NFS 或 SMB 磁碟區的建立。選項有 nfs，`smb`或空值。設定為 null 則預設使用 NFS 磁碟區。 | nfs |
| supportedTopologies | 表示從後端支援的區域和區域列表。更多信息，請參閱"使用 CSI 拓撲"。例如：supportedTopologies:<br>- topology.kubernetes.io/region: asia-east1<br>topology.kubernetes.io/zone: asia-east1-a | |

卷配置選項

您可以控制預設卷配置 `defaults`設定檔部分。

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| exportRule | 新卷的出口規則。必須是以逗號分隔的 IPv4 位址列表，位址可以任意組合。 | “0.0.0.0/0” |
| snapshotDir | 訪問 `.snapshot`目錄 | NFSv4 為“true”，NFSv3 為“false”。 |
| snapshotReserve | 快照預留的磁碟區百分比 | （接受預設值 0） |
| unixPermissions | 新磁碟區的 Unix 權限（4 位八進位數字）。 | “” |

以下範例展示了基本配置，其中大多數參數都保留預設值。這是定義後端最簡單的方法。

最小配置

這是最基本的後端配置。透過此配置， Trident會發現您已委派給配置位置中Google Cloud NetApp Volumes的所有儲存池，並隨機將新磁碟區放置在其中一個儲存池上。因為 `nasType`省略了 `nfs`預設設定生效，後端將為 NFS 磁碟區進行設定。

如果您剛開始使用Google Cloud NetApp Volumes並進行嘗試，這種配置是理想的，但在實踐中，您很可能需要為配置的磁碟區提供額外的範圍。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    XsYg6gyxy4zq7OlwWgLwGa==\n
    -----END PRIVATE KEY-----\n

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

**SMB磁碟區配置**

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

使用儲存池過濾器進行配置

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

虛擬池配置

此後端配置在一個檔案中定義了多個虛擬池。虛擬池在以下位置定義： `storage`部分。當您有多個支援不同服務等級的儲存池，並且想要在 Kubernetes 中建立代表這些儲存層級的儲存類別時，它們非常有用。虛擬池標籤用於區分不同的池子。例如，在下面的例子中 `performance`標籤和 `serviceLevel`類型用於區分虛擬池。

您也可以設定一些適用於所有虛擬池的預設值，並覆寫各個虛擬池的預設值。在下面的例子中，`snapshotReserve`和 `exportRule`作為所有虛擬池的預設值。

更多信息，請參閱"虛擬池"。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
```

```
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

**GKE 的雲端身份**

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

支援的拓撲配置

Trident可根據區域和可用區為工作負載提供磁碟區。這 `supportedTopologies`此後端配置中的區塊用於提供每個後端的區域和區域清單。此處指定的區域和區域值必須與每個 Kubernetes 叢集節點上的標籤中的區域和區域值相符。這些區域和分區代表儲存類別中可以提供的允許值的清單。對於包含後端提供的區域和可用區子集的儲存類， Trident會在所述區域和可用區中建立磁碟區。更多信息，請參閱"使用 CSI 拓撲"。

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

下一步是什麼？

建立後端設定檔後，執行以下命令：

```
kubectl create -f <backend-file>
```

若要驗證後端是否已建立成功，請執行下列命令：

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

如果後端建立失敗，則後端配置存在問題。您可以使用以下方式描述後端： `kubectl get tridentbackendconfig <backend-name>`執行以下命令查看日誌以確定原因：

```
tridentctl logs
```

在您發現並修正設定檔中的問題後，您可以刪除後端並再次執行建立命令。

#### 儲存類別定義

以下是一個基本內容 `StorageClass`上述後端所指的定義。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

使用以下範例定義 **`parameter.selector`**場地：

使用 `parameter.selector`你可以為每個物件指定。 `StorageClass`這"虛擬池"用於託管卷。該磁碟區將具有所選池中定義的方面。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

有關存儲類別的更多詳細信息，請參閱"建立儲存類別"。

**SMB** 磁碟區的範例定義

使用 `nasType`， `node-stage-secret-name` ， 和 `node-stage-secret-namespace`您可以指定 SMB 磁碟區並提供所需的 Active Directory 憑證。任何 Active Directory 使用者/密碼，無論擁有任何權限或沒有權限，都可以用作節點階段金鑰。

預設命名空間上的基本配置

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

每個命名空間使用不同的密鑰

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

每個磁碟區使用不同的密鑰

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`篩選支援 SMB 磁碟區的儲存池。 `nasType: nfs`或者 `nasType: null`NFS池過濾器。

**PVC定義範例**

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

若要驗證 PVC 是否已綁定，請執行以下命令：

```
kubectl get pvc gcnv-nfs-pvc

NAME           STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc   Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213   100Gi
RWX            gcnv-nfs-sc  1m
```

## 為 Google Cloud 後端設定Cloud Volumes Service

了解如何使用提供的範例配置，將NetApp Cloud Volumes Service for Google Cloud 配置為Trident安裝的後端。

### Google Cloud 驅動程式詳情

Trident提供 `gcp-cvs`驅動程式與集群通訊。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

| 司機 | 協定 | 音量模式 | 支援的存取模式 | 支援的檔案系統 |
|---|---|---|---|---|
| `gcp-cvs` | NFS | 檔案系統 | RWO、ROX、RWX、RWOP | `nfs` |

### 了解Trident對 Google Cloud Cloud Volumes Service 的支持

Trident可以在以下兩種格式之一建立Cloud Volumes Service磁碟區："服務類型" ：

- **CVS-Performance**： Trident 的預設服務類型。這種效能優化型服務類型最適合重視效能的生產工作負載。CVS-Performance 服務類型是一種硬體選項，支援最小 100 GiB 大小的磁碟區。您可以選擇其中之一"三個服務級別"：
  - `standard`
  - `premium`
  - `extreme`
- **CVS**：CVS 服務類型提供較高的區域可用性，但效能水準有限或中等。 CVS 服務類型是一種軟體選項，它使用儲存池來支援小至 1 GiB 的磁碟區。儲存池最多可包含 50 個磁碟區，所有磁碟區共用池的容量和效能。您可以選擇其中之一"兩種服務級別"：
  - `standardsw`
  - `zoneredundantstandardsw`

你需要什麼

配置和使用 "適用於 Google Cloud 的Cloud Volumes Service"後端需要以下組件：

- 已配置NetApp Cloud Volumes Service的 Google Cloud 帳戶
- 您的 Google Cloud 帳戶的項目編號
- 擁有 Google Cloud 服務帳戶 `netappcloudvolumes.admin`角色
- 您的Cloud Volumes Service帳戶的 API 金鑰文件

後端配置選項

每個後端都在單一 Google Cloud 區域中配置磁碟區。若要在其他區域建立卷，您可以定義其他後端。

| 範圍 | 描述 | 預設 |
|---|---|---|
| `version` | | 始終為 1 |
| `storageDriverName` | 儲存驅動程式的名稱 | "gcp-cvs" |
| `backendName` | 自訂名稱或儲存後端 | 驅動程式名稱 + "_" + API 金鑰的一部分 |
| `storageClass` | 用於指定 CVS 服務類型的可選參數。使用 `software`選擇 `CVS` 服務類型。否則， `Trident` 會假定為 `CVS-Performance` 服務類型(`hardware`) 。 | |
| `storagePools` | 僅限CVS服務類型。用於指定磁碟區建立儲存池的可選參數。 | |
| `projectNumber` | Google Cloud 帳戶項目編號。該值可在 Google Cloud 入口網站首頁找到。 | |
| `hostProjectNumber` | 如果使用共用 VPC 網絡，則必須執行此操作。在這種情況下， `projectNumber`這是一個服務項目，而且 `hostProjectNumber`是宿主項目。 | |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| apiRegion | Trident建立Cloud Volumes Service區的 Google Cloud 區域。建立跨區域 Kubernetes 叢集時，在下列位置建立的磁碟區： `apiRegion`可用於跨多個 Google Cloud 區域的節點上調度的工作負載。跨區域運輸會產生額外費用。 | |
| apiKey | 用於 Google Cloud 服務帳戶的 API 金鑰 `netappcloudvolumes.admin`角色。它包含 Google Cloud 服務帳戶私鑰檔案的 JSON 格式內容（原封不動地複製到後端設定檔中）。 | |
| proxyURL | 如果需要代理伺服器才能連接到 CVS 帳戶，請提供代理 URL。代理伺服器可以是HTTP代理，也可以是HTTPS代理。對於 HTTPS 代理，會跳過憑證驗證，以允許在代理伺服器中使用自簽名憑證。不支援啟用身份驗證的代理伺服器。 | |
| nfsMountOptions | 對 NFS 掛載選項進行精細控制。 | "nfsvers=3" |
| limitVolumeSize | 如果請求的磁碟區大小大於此值，則配置失敗。 | （預設不強制執行） |
| serviceLevel | CVS-Performance 或 CVS 服務等級（適用於新磁碟區）。 CVS-Performance 值是 standard， premium ， 或者 extreme。 CVS 值是 standardsw`或者 `zoneredundantstandardsw。 | CVS-Performance 預設值為「標準」。 CVS 預設值為“standardsw”。 |
| network | Google Cloud 網路用於Cloud Volumes Service磁碟區。 | "預設" |
| debugTraceFlags | 故障排除時要使用的調試標誌。例子， \{"api":false, "method":true} 。除非您正在進行故障排除並需要詳細的日誌轉儲，否則請勿使用此功能。 | 無效的 |
| allowedTopologies | 若要啟用跨區域訪問，您的 StorageClass 定義如下： allowedTopologies`必須包含所有地區。例如： `- key: topology.kubernetes.io/region<br>values:<br>- us-east1<br>- europe-west1 | |

卷配置選項

您可以控制預設卷配置 `defaults`設定檔部分。

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| exportRule | 新卷的出口規則。必須是以逗號分隔的 IPv4 位址或 IPv4 子網路的列表，採用 CIDR 表示法。 | “0.0.0.0/0” |
| snapshotDir | 訪問 `.snapshot`目錄 | “錯誤的” |
| snapshotReserve | 快照預留的磁碟區百分比 | （接受 CVS 預設值 0） |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| `size` | 新卷的規模。 CVS-Performance 最低要求為 100 GiB。 CVS 最小容量為 1 GiB。 | CVS-Performance 服務類型預設為「100GiB」。 CVS 服務類型不設定預設值，但要求至少 1 GiB。 |

#### CVS-Performance 服務類型範例

以下範例提供了 CVS-Performance 服務類型的範例設定。

範例 **1**：最小配置

這是使用預設 CVS-Performance 服務類型和預設「標準」服務等級的最小後端配置。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: "012345678901"
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: <id_value>
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: "123456789012345678901"
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
```

範例 2：服務等級配置

此範例展示了後端配置選項，包括服務等級和磁碟區預設值。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
proxyURL: http://proxy-server-hostname/
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 10Ti
serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '5'
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  size: 5Ti
```

範例 3：虛擬池配置

此範例使用 `storage`配置虛擬池和 `StorageClasses`指的是他們。請參閱[儲存類別定義]查看儲存類別的定義方式。

這裡為所有虛擬池設定了特定的預設值，這些預設值決定了： `snapshotReserve` 5%和 `exportRule` 至 `0.0.0.0/0`。虛擬池在以下位置定義： `storage`部分。每個虛擬池都定義了自己的規則。 `serviceLevel`並且有些池會覆蓋預設值。虛擬池標籤用於根據以下因素區分池子： `performance` 和 `protection`。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
nfsMountOptions: vers=3,proto=tcp,timeo=600
defaults:
  snapshotReserve: '5'
  exportRule: 0.0.0.0/0
labels:
  cloud: gcp
region: us-west2
storage:
- labels:
    performance: extreme
    protection: extra
  serviceLevel: extreme
  defaults:
```

```
    snapshotDir: 'true'
    snapshotReserve: '10'
    exportRule: 10.0.0.0/24
- labels:
    performance: extreme
    protection: standard
  serviceLevel: extreme
- labels:
    performance: premium
    protection: extra
  serviceLevel: premium
  defaults:
    snapshotDir: 'true'
    snapshotReserve: '10'
- labels:
    performance: premium
    protection: standard
  serviceLevel: premium
- labels:
    performance: standard
  serviceLevel: standard
```

儲存類別定義

以下 StorageClass 定義適用於虛擬池設定範例。使用 `parameters.selector`您可以為每個 StorageClass 指定用於託管磁碟區的虛擬池。該磁碟區將具有所選池中定義的方面。

儲存類別範例

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme; protection=extra
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium; protection=standard
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium; protection=extra
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium; protection=standard
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: csi.trident.netapp.io
parameters:
```

```
  selector: performance=standard
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: protection=extra
allowVolumeExpansion: true
```

- 第一個儲存類(`cvs-extreme-extra-protection`) 映射到第一個虛擬池。這是唯一提供極致效能且快照儲備為 10% 的儲存池。
- 最後一個儲存類別(`cvs-extra-protection`) 呼叫任何提供 10% 快照保留的儲存池。 Trident決定選擇哪個虛擬池，並確保滿足快照儲備要求。

**CVS** 服務類型範例

以下範例提供了 CVS 服務類型的範例配置。

範例 **1**：最小配置

這是使用最簡後端配置 `storageClass`指定 CVS 服務類型和預設值 `standardsw`服務水平。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
storageClass: software
apiRegion: us-east4
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
serviceLevel: standardsw
```

範例 **2**：儲存池配置

此範例後端配置使用 `storagePools`配置儲存池。

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
data.iam.gserviceaccount.com
  client_id: '107071413297115343396'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40cloud-native-data.iam.gserviceaccount.com
storageClass: software
zone: europe-west1-b
network: default
storagePools:
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509
serviceLevel: Standardsw
```

下一步是什麼？

建立後端設定檔後，執行以下命令：

```
tridentctl create backend -f <backend-file>
```

如果後端建立失敗，則後端配置存在問題。您可以透過執行以下命令查看日誌以確定原因：

```
tridentctl logs
```

在您發現並修正設定檔中的問題後，您可以再次執行建立命令。

## 配置NetApp HCI或SolidFire後端

### 了解如何在Trident安裝中建立和使用 Element 後端。

#### 元素驅動程式詳情

Trident提供 `solidfire-san`用於與叢集通訊的儲存驅動程式。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

這 `solidfire-san`儲存驅動程式支援_檔案_和_區塊_磁碟區模式。對於 `Filesystem`volumeMode， Trident建立一個磁碟區並建立一個檔案系統。檔案系統類型由 StorageClass 指定。

| 司機 | 協定 | 音量模式 | 支援的存取模式 | 支援的檔案系統 |
|---|---|---|---|---|
| `solidfire-san` | iSCSI | 堵塞 | RWO、ROX、RWX、RWOP | 沒有檔案系統。原始塊設備。 |
| `solidfire-san` | iSCSI | 檔案系統 | RWO，RWOP | `xfs`， `ext3` ， `ext4` |

#### 開始之前

在建立 Element 後端之前，您需要以下內容。

- 一個支援運行 Element 軟體的儲存系統。
- 擁有NetApp HCI/ SolidFire叢集管理員或租用戶用戶權限，可管理磁碟區。
- 所有 Kubernetes 工作節點都應該安裝對應的 iSCSI 工具。參考"工作節點準備訊息"。

#### 後端配置選項

請參閱下表以了解後端配置選項：

| 範圍 | 描述 | 預設 |
|---|---|---|
| `version` | | 始終為 1 |
| `storageDriverName` | 儲存驅動程式的名稱 | 始終是“solidfire-san” |
| `backendName` | 自訂名稱或儲存後端 | "solidfire_" + 儲存體 (iSCSI) IP 位址 |
| `Endpoint` | 針對SolidFire叢集的 MVIP，包含租戶憑證 | |
| `SVIP` | 儲存（iSCSI）IP 位址和連接埠 | |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| `labels` | 若要套用於磁碟區的任意 JSON 格式標籤集。 | “” |
| `TenantName` | 要使用的租用戶名稱（如果找不到則建立） | |
| `InitiatorIFace` | 將 iSCSI 流量限製到特定主機介面 | "預設" |
| `UseCHAP` | 使用 CHAP 對 iSCSI 進行身份驗證。 Trident使用 CHAP。 | 真的 |
| `AccessGroups` | 要使用的存取群組 ID 列表 | 尋找名為「trident」的訪問群組的 ID |
| `Types` | QoS規範 | |
| `limitVolumeSize` | 如果請求的磁碟區大小超過此值，則配置失敗。 | （預設不強制執行） |
| `debugTraceFlags` | 故障排除時要使用的調試標誌。例如，{"api":false, "method":true} | 無效的 |

請勿使用 `debugTraceFlags`除非您正在進行故障排除並且需要詳細的日誌轉儲。

**範例 1：後端配置 `solidfire-san`具有三種音量類型的驅動器**

本範例展示了一個使用 CHAP 認證的後端文件，並對三種具有特定 QoS 保證的捲類型進行了建模。最有可能的情況是，您會定義儲存類別來使用它們。 `IOPS`儲存類別參數。

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
```

範例 2：後端和儲存類別配置 **`solidfire-san`**帶有虛擬池的驅動程式

此範例顯示了配置了虛擬池的後端定義檔以及引用這些虛擬池的儲存類別。

Trident在設定時將儲存池中的標籤複製到後端儲存 LUN。為了方便起見，儲存管理員可以為每個虛擬池定義標籤，並按標籤將磁碟區分組。

在下方所示的範例後端定義檔中，所有儲存池都設定了特定的預設值，這些預設值決定了： `type`銀級。虛擬池在以下位置定義： `storage`部分。在這個例子中，一些儲存池設定了自己的類型，而一些儲存池則覆蓋了上面設定的預設值。

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
UseCHAP: true
```

```
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
  - labels:
      performance: gold
      cost: "4"
    zone: us-east-1a
    type: Gold
  - labels:
      performance: silver
      cost: "3"
    zone: us-east-1b
    type: Silver
  - labels:
      performance: bronze
      cost: "2"
    zone: us-east-1c
    type: Bronze
  - labels:
      performance: silver
      cost: "1"
    zone: us-east-1d
```

以下 StorageClass 定義與上述虛擬池相關。使用 `parameters.selector`在欄位中，每個 StorageClass 都會指定哪些虛擬池可用於託管磁碟區。卷將具有所選虛擬池中定義的各個方面。

第一個儲存類(`solidfire-gold-four`將映射到第一個虛擬池。這是唯一提供黃金級性能的泳池。

`Volume Type QoS`黃金。最後一個儲存類別（`solidfire-silver`）指出任何提供銀級性能的儲存池。Trident將決定選擇哪個虛擬池，並確保滿足儲存需求。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold; cost=4
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=3
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze; cost=2
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=1
  fsType: ext4

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

```
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
  fsType: ext4
```

查找更多信息

- "卷訪問群組"

## ONTAP SAN 驅動程式

### ONTAP SAN 驅動程式概述

了解如何使用ONTAP和Cloud Volumes ONTAP SAN 驅動程式設定ONTAP後端。

#### ONTAP SAN 驅動程式詳情

Trident提供以下 SAN 儲存驅動程序，用於與ONTAP叢集通訊。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

| 司機 | 協定 | 音量模式 | 支援的存取模式 | 支援的檔案系統 |
|---|---|---|---|---|
| `ontap-san` | iSCSI 透過光纖通道提供 SCSI 服務 | 堵塞 | RWO、ROX、RWX、RWOP | 無檔案系統；原始區塊設備 |
| `ontap-san` | iSCSI 透過光纖通道提供 SCSI 服務 | 檔案系統 | RWO，RWOP<br><br>ROX 和 RWX 在檔案系統磁碟區模式下不可用。 | `xfs`， `ext3` ， `ext4` |
| `ontap-san` | NVMe/TCP<br><br>參考NVMe/TCP 的其他注意事項。 | 堵塞 | RWO、ROX、RWX、RWOP | 無檔案系統；原始區塊設備 |
| `ontap-san` | NVMe/TCP<br><br>參考NVMe/TCP 的其他注意事項。 | 檔案系統 | RWO，RWOP<br><br>ROX 和 RWX 在檔案系統磁碟區模式下不可用。 | `xfs`， `ext3` ， `ext4` |

| 司機 | 協定 | 音量模式 | 支援的存取模式 | 支援的檔案系統 |
| --- | --- | --- | --- | --- |
| `ontap-san-economy` | iSCSI | 堵塞 | RWO、ROX、RWX、RWOP | 無檔案系統；原始區塊設備 |
| `ontap-san-economy` | iSCSI | 檔案系統 | RWO，RWOP<br><br>ROX 和 RWX 在檔案系統磁碟區模式下不可用。 | `xfs`，`ext3`，`ext4` |

- 使用 `ontap-san-economy`僅當預計持續卷使用量高於...時"支援的ONTAP容量限制"。
- 使用 `ontap-nas-economy`僅當預計持續卷使用量高於...時"支援的ONTAP容量限制"以及 `ontap-san-economy`驅動程式無法使用。
- 請勿使用 `ontap-nas-economy`如果您預計需要資料保護、災難復原或行動辦公室。
- NetApp不建議在所有ONTAP驅動程式中使用 Flexvol 自動成長，ontap-san 除外。作為變通方法， Trident支援使用快照儲備，並相應地調整 Flexvol 容量。

使用者權限

Trident預期以ONTAP或 SVM 管理員身分執行，通常使用下列方式： `admin`集群用戶或 `vsadmin`SVM 用戶，或具有相同角色但名稱不同的用戶。對於Amazon FSx for NetApp ONTAP部署， Trident需要以ONTAP或 SVM 管理員身分執行，並使用叢集。 `fsxadmin`用戶或 `vsadmin`SVM 用戶，或具有相同角色但名稱不同的用戶。這 `fsxadmin`用戶是集群管理員用戶的有限替代品。

如果你使用 `limitAggregateUsage`需要參數和叢集管理員權限。當使用Amazon FSx for NetApp ONTAP和Trident時， `limitAggregateUsage`參數將無法與 `vsadmin`和 `fsxadmin`用戶帳戶。如果指定此參數，配置操作將失敗。

雖然可以在ONTAP中建立一個限制性更強的角色供Trident驅動程式使用，但我們不建議這樣做。 Trident的大多數新版本都會呼叫額外的 API，這些 API 必須加以考慮，這使得升級變得困難且容易出錯。

**NVMe/TCP** 的其他注意事項

Trident支援使用非揮發性記憶體高速介面 (NVMe) 協定 `ontap-san`驅動程式包括：

- IPv6
- NVMe磁碟區的快照和克隆
- 調整 NVMe 卷的大小
- 導入在Trident外部建立的 NVMe 卷，以便Trident可以管理其生命週期。
- NVMe原生多路徑
- K8s節點的優雅關閉或非優雅關閉（24.06）

Trident不支援：

- NVMe 原生支援的 DH-HMAC-CHAP
- 設備映射器 (DM) 多路徑

- LUKS 加密

NVMe 僅支援ONTAP REST API，不支援 ONTAPI (ZAPI)。

準備配置後端**ONTAP SAN** 驅動程式

## 了解配置ONTAP後端和ONTAP SAN 驅動程式的要求和驗證選項。

### 要求

對於所有ONTAP後端， Trident要求至少將一個聚合分配給 SVM。

"ASA r2 系統"與其他ONTAP系統（ASA、 AFF和FAS）在儲存層的實作上有所不同。在ASA r2 系統中，使用儲存可用區而不是聚合。請參閱"這"知識庫文章，介紹如何在ASA r2 系統中將聚合指派給 SVM。

請記住，您還可以運行多個驅動程序，並建立指向其中一個或另一個驅動程式的儲存類別。例如，您可以設定一個 `san-dev`使用類別 `ontap-san`司機和 `san-default`使用類別 `ontap-san-economy`一。

所有 Kubernetes 工作節點都必須安裝對應的 iSCSI 工具。參考 "準備工作節點" 了解詳情。

#### 對**ONTAP**後端進行身份驗證

Trident提供兩種ONTAP後端身分驗證方式。

- 基於憑證：具有所需權限的ONTAP使用者的使用者名稱和密碼。建議使用預先定義的安全登入角色，例如：`admin`或者 `vsadmin`確保與ONTAP版本最大程度相容。
- 基於憑證： Trident也可以使用安裝在後端的憑證與ONTAP叢集通訊。此處，後端定義必須包含用戶端憑證、金鑰和受信任 CA 憑證（如果使用，建議使用）的 Base64 編碼值。

您可以更新現有後端，以在基於憑證的方法和基於憑證的方法之間進行切換。但是，一次只能支援一種身份驗證方法。要切換到不同的身份驗證方法，必須從後端配置中刪除現有方法。

如果您嘗試同時提供憑證和證書，則後端建立將會失敗，並出現錯誤，提示設定檔中提供了多個驗證方法。

#### 啟用基於憑證的身份驗證

Trident需要 SVM 範圍/叢集範圍管理員的憑證才能與ONTAP後端通訊。建議使用標準的、預先定義的角色，例如： `admin`或者 `vsadmin`。這樣可以確保與未來ONTAP版本向前相容，這些版本可能會公開一些功能 API，供未來的Trident版本使用。雖然可以建立自訂安全登入角色並將其與Trident一起使用，但不建議這樣做。

後端定義範例如下圖所示：

**YAML**

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

**JSON**

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

請注意，後端定義是唯一以純文字形式儲存憑證的地方。後端建立完成後，使用者名稱/密碼將使用 Base64 進行編碼，並儲存為 Kubernetes 金鑰。只有在建立或更新後端時才需要了解憑證。因此，這是一項僅限管理員執行的操作，由 Kubernetes/儲存管理員執行。

啟用基於憑證的身份驗證

新的和現有的後端都可以使用憑證與ONTAP後端通訊。後端定義需要三個參數。

- clientCertificate：客戶端憑證的 Base64 編碼值。
- clientPrivateKey：關聯私鑰的 Base64 編碼值。
- trustedCACertificate：受信任 CA 憑證的 Base64 編碼值。如果使用受信任的 CA，則必須提供此參數。如果沒有使用受信任的憑證授權機構，則可以忽略此步驟。

典型的工作流程包括以下步驟。

步驟

1. 產生客戶端憑證和金鑰。產生時，將通用名稱 (CN) 設定為要進行驗證的ONTAP使用者。

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. 在ONTAP叢集中新增受信任的 CA 憑證。這可能已經由儲存管理員處理了。如果沒有使用受信任的憑證授權機構，則忽略此操作。

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. 在ONTAP叢集上安裝客戶端憑證和金鑰（來自步驟 1）。

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. 確認ONTAP安全登入角色支持 `cert`身份驗證方法。

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. 使用產生的憑證進行身份驗證。請將 < ONTAP管理 LIF> 和 <vserver 名稱> 替換為管理 LIF IP 位址和 SVM 名稱。

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. 使用 Base64 對憑證、金鑰和受信任的 CA 憑證進行編碼。

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 使用上一步獲得的值建立後端。

```
cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

更新身份驗證方法或輪換憑證

您可以更新現有後端，以使用不同的身份驗證方法或輪換其憑證。這種方法是雙向的：使用使用者名稱/密碼的後端可以更新為使用憑證；使用憑證的後端可以更新為基於使用者名稱/密碼的後端。為此，您必須刪除現有的身份驗證方法並新增新的身份驗證方法。然後使用包含所需參數的更新後的 backend.json 檔案來執行 `tridentctl backend update`。

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

輪換密碼時，儲存管理員必須先更新ONTAP上使用者的密碼。接下來將進行後端更新。輪換證書時，可以為使用者新增多個證書。然後更新後端以使用新證書，之後即可從ONTAP叢集中刪除舊證書。

更新後端不會中斷對已建立磁碟區的訪問，也不會影響之後建立的磁碟區連線。後端更新成功表明Trident可以與ONTAP後端通訊並處理未來的磁碟區操作。

### 為**Trident**建立自訂**ONTAP**角色

您可以建立一個具有最低權限的ONTAP叢集角色，這樣您就不必使用ONTAP管理員角色在Trident中執行操作。在Trident後端設定中包含使用者名稱時， Trident將使用您建立的ONTAP叢集角色來執行操作。

請參閱"Trident自訂角色產生器"有關建立Trident自訂角色的詳細資訊。

使用**ONTAP CLI**

1. 使用以下命令建立新角色：

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. 為Trident用戶建立使用者名稱：

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver
<svm_name\> -comment "user_description"
```

3. 將角色映射到使用者：

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

使用系統管理員

在ONTAP系統管理員中執行下列步驟：

1. 建立自訂角色：
   a. 若要在叢集層級建立自訂角色，請選擇「叢集 > 設定」。

      （或）若要在 SVM 層級建立自訂角色，請選擇「儲存」>「儲存虛擬機器」> `required SVM` > 設定 > 使用者和角色*。

   b. 選擇“使用者和角色”旁邊的箭頭圖示（→）。
   c. 在“角色”下選擇“+添加”。
   d. 定義角色規則，然後點選「儲存」。
2. 將角色對應到Trident使用者： + 在「使用者和角色」頁面上執行下列步驟：
   a. 在「使用者」下方選擇「新增」圖示 **+**。
   b. 選擇所需的使用者名，然後在「角色」下拉式選單中選擇角色。
   c. 點選“儲存”。

更多資訊請參閱以下頁面：

- "用於管理ONTAP的自訂角色"或者"定義自訂角色"
- "與角色和使用者協作"

使用雙向 **CHAP** 驗證連接

Trident可以使用雙向 CHAP 對 iSCSI 會話進行驗證。 `ontap-san`和 `ontap-san-economy`司機。這需要啟用 `useCHAP`在後端定義中新增選項。設定為 `true`Trident將 SVM 的預設發起程序安全地配置為雙向 CHAP，並從後端檔案設定使用者名稱和金鑰。 NetApp建議使用雙向 CHAP 協定對連線進行身份驗證。請參閱以下範例配

置：

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

這 `useCHAP`此參數是一個布林選項，只能配置一次。預設值為 false。一旦將其設為 true，就無法再將其設為 false。

另外 `useCHAP=true`，這 `chapInitiatorSecret`， `chapTargetInitiatorSecret` ， `chapTargetUsername` ， 和 `chapUsername`欄位必須包含在後端定義中。創建後端後，可以透過執行以下命令來更改密鑰： `tridentctl update` 。

工作原理

透過設定 `useCHAP`如果設定為 true，則儲存管理員指示Trident在儲存裝置後端設定 CHAP。其中包括以下內容：

- 在SVM上設定CHAP：
  - 如果 SVM 的預設啟動器安全類型為「無」（預設）*且*磁碟區中不存在任何預先存在的 LUN，Trident會將預設安全類型設為「無」。 `CHAP`然後繼續配置 CHAP 發起程序和目標使用者名稱及金鑰。
  - 如果 SVM 包含 LUN， Trident將不會在 SVM 上啟用 CHAP。這樣可以確保對 SVM 上已存在的 LUN 的存取不受限制。
- 配置 CHAP 發起程序和目標使用者名稱和金鑰；這些選項必須在後端配置中指定（如上所示）。

後端建立完成後， Trident會建立一個對應的 `tridentbackend`CRD 並將 CHAP 金鑰和使用者名稱儲存為 Kubernetes 金鑰。 Trident在此後端建立的所有 PV 都將透過 CHAP 進行安裝和連線。

輪換憑證並更新後端

您可以透過更新 CHAP 參數來更新 CHAP 憑證。 `backend.json`文件。這將需要更新 CHAP 金鑰並使用 `tridentctl update`命令反映這些變更。

更新後端 CHAP 金鑰時，必須使用 `tridentctl`更新後端。請勿使用ONTAP CLI 或ONTAP系統管理員更新儲存叢集上的憑證，因為Trident將無法偵測到這些變更。

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

現有連線將不受影響；如果Trident在 SVM 上更新憑證，則這些連線將繼續保持活動狀態。新連線使用更新後的憑證，現有連線將繼續保持活動狀態。斷開並重新連接舊的PV將使它們使用更新的憑證。

### ONTAP SAN 配置選項和範例

了解如何在Trident安裝中建立和使用ONTAP SAN 驅動程式。本節提供後端設定範例以及將後端對應到 StorageClasses 的詳細資訊。

"ASA r2 系統"與其他ONTAP系統（ASA、 AFF和FAS）在儲存層的實作上有所不同。這些變化會影響某些參數的使用，如註釋中所述。"了解更多關於ASA r2 系統與其他ONTAP系統之間的差異"。

只有 `ontap-san`ASA r2 系統支援驅動程式（支援 iSCSI 和 NVMe/TCP 協定）。

在Trident後端設定中，無需指定您的系統是ASA r2。當您選擇 `ontap-san`作為 `storageDriverName`Trident可自動偵測ASA r2 或傳統的ONTAP系統。如下表所示，某些後端設定參數不適用於ASA r2 系統。

請參閱下表以了解後端配置選項：

| 範圍 | 描述 | 預設 |
|---|---|---|
| `version` | | 始終為 1 |
| `storageDriverName` | 儲存驅動程式的名稱 | `ontap-san`或者 `ontap-san-economy` |
| `backendName` | 自訂名稱或儲存後端 | 驅動程式名稱 + "_" + dataLIF |
| `managementLIF` | 叢集或 SVM 管理 LIF 的 IP 位址。<br><br>可以指定一個完全限定網域名稱（FQDN）。<br><br>如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。 IPv6 位址必須用方括號定義，例如：`[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 。<br><br>為了實現MetroCluster 的無縫切換，請參閱MetroCluster範例。<br><br>ⓘ 如果您使用的是「vsadmin」憑證，`managementLIF`必須是SVM的憑證；如果使用「admin」憑證，`managementLIF`必須是集群的那個。 | "10.0.0.1", "[2001:1234:abcd::fefe]" |
| `dataLIF` | LIF協定的IP位址。如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。 IPv6 位址必須用方括號定義，例如：`[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 。 *請勿指定使用 iSCSI。 *Trident的使用"ONTAP選擇性 LUN 地圖"發現建立多路徑會話所需的 iSCSI LIF。如果出現以下情況，則會產生警告： `dataLIF`已明確定義。 *Metrocluster 除外。 *查看MetroCluster範例。 | 由支援向量機導出 |
| `svm` | 要使用的儲存虛擬機器 *Metrocluster 除外。 *查看MetroCluster範例。 | 如果是 SVM 則推導而來 `managementLIF`已指定 |
| `useCHAP` | 使用 CHAP 對ONTAP SAN 驅動程式的 iSCSI 進行驗證 [布林值]。設定為 `true`讓Trident配置並使用雙向 CHAP 作為後端給定 SVM 的預設身份驗證。參考 "準備配置後端ONTAP SAN 驅動程式" 了解詳情。 *不支援FCP或NVMe/TCP協定。 * | `false` |
| `chapInitiatorSecret` | CHAP 發起者金鑰。如果是必填項 `useCHAP=true` | “” |
| `labels` | 若要套用於磁碟區的任意 JSON 格式標籤集 | “” |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| `chapTargetInitiatorSecret` | CHAP 目標發起者金鑰。如果是必填項 `useCHAP=true` | “” |
| `chapUsername` | 入站用戶名。如果是必填項 `useCHAP=true` | “” |
| `chapTargetUsername` | 目標用戶名。如果是必填項 `useCHAP=true` | “” |
| `clientCertificate` | 用戶端憑證的 Base64 編碼值。用於基於憑證的身份驗證 | “” |
| `clientPrivateKey` | 客戶端私鑰的 Base64 編碼值。用於基於憑證的身份驗證 | “” |
| `trustedCACertificate` | 受信任 CA 憑證的 Base64 編碼值。選修的。用於基於憑證的身份驗證。 | “” |
| `username` | 與ONTAP叢集通訊所需的使用者名稱。用於基於憑證的身份驗證。有關 Active Directory 驗證，請參閱 "使用 Active Directory 憑證向後端 SVM 驗證Trident 的身份"。 | “” |
| `password` | 與ONTAP集群通訊所需的密碼。用於基於憑證的身份驗證。有關 Active Directory 驗證，請參閱 "使用 Active Directory 憑證向後端 SVM 驗證Trident 的身份"。 | “” |
| `svm` | 使用的儲存虛擬機 | 如果是 SVM 則推導而來 `managementLIF`已指定 |
| `storagePrefix` | 在 SVM 中配置新磁碟區時所使用的前綴。之後無法修改。要更新此參數，您需要建立一個新的後端。 | `trident` |
| `aggregate` | 用於配置的聚合（可選；如果設置，則必須指派給 SVM）。對於 `ontap-nas-flexgroup`驅動程序，此選項將被忽略。如果未分配，則可以使用任何可用的聚合來配置FlexGroup磁碟區。<br><br>ⓘ 當 SVM 中的聚合資料更新時， Trident 會自動輪詢 SVM 進行更新，而無需重新啟動Trident控制器。當您在Trident中配置特定聚合以配置磁碟區時，如果該聚合被重新命名或移出 SVM，則在輪詢 SVM 聚合時， Trident中的後端將變為失敗狀態。您必須將聚合變更為 SVM 上存在的聚合，或將其完全刪除，才能使後端恢復連線。<br><br>請勿指定用於**ASA r2** 系統。 | “” |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| `limitAggregateUsage` | 如果使用率超過此百分比，則配置失敗。如果您使用的是Amazon FSx for NetApp ONTAP後端，請勿指定 `limitAggregateUsage`。提供的 `fsxadmin`和 `vsadmin`不包含檢索匯總使用情況和使用Trident限制它所需的權限。 請勿指定用於**ASA r2** 系統。 | （預設不強制執行） |
| `limitVolumeSize` | 如果請求的磁碟區大小大於此值，則配置失敗。同時限制其管理的 LUN 卷的最大大小。 | （預設不強制執行） |
| `lunsPerFlexvol` | 每個 Flexvol 的最大 LUN 數量必須在 [50, 200] 範圍內 | `100` |
| `debugTraceFlags` | 故障排除時要使用的調試標誌。例如，{"api":false, "method":true} 除非您正在進行故障排除並且需要詳細的日誌轉儲，否則請勿使用此方法。 | `null` |
| `useREST` | 使用ONTAP REST API 的布林參數。<br><br>`useREST`設定為 `true`Trident使用ONTAP REST API與後端通訊；當設定為 `false`Trident使用 ONTAPI (ZAPI) 呼叫與後端通訊。此功能需要ONTAP 9.11.1及更高版本。此外，所使用的ONTAP登入角色必須具有存取權限。`ontapi`應用。預定義項滿足了這一點。`vsadmin`和 `cluster-admin`角色。從Trident 24.06 版本和ONTAP 9.15.1 或更高版本開始，`useREST`設定為 `true`預設；更改 `useREST`到 `false`使用 ONTAPI (ZAPI) 呼叫。<br><br>`useREST`完全符合 NVMe/TCP 標準。<br><br>NVMe 僅支援ONTAP REST API，不支援 ONTAPI (ZAPI)。<br><br>如果指定，則始終設定為 **`true`**適用於**ASA r2** 系統。 | `true`適用於ONTAP 9.15.1 或更高版本，否則 `false`。 |
| `sanType` | 用於選擇 `iscsi`對於 iSCSI， `nvme`適用於 NVMe/TCP 或 `fcp`用於光纖通道 (FC) 上的 SCSI。 | `iscsi`如果為空 |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| `formatOptions` | 使用 `formatOptions`為以下情況指定命令列參數 `mkfs`該命令將在每次格式化磁碟區時套用。這樣您就可以根據自己的喜好格式化音量。請確保指定與 mkfs 指令選項類似的 formatOptions，但不包含裝置路徑。例如：“-E nodiscard”<br><br>*支援 `ontap-san`和 `ontap-san-economy`支援 iSCSI 協定的驅動程式。 * *此外，在使用 iSCSI 和 NVMe/TCP 協定時， ASA r2 系統也受支援。 * | |
| `limitVolumePoolSize` | 在 ontap-san-economy 後端使用 LUN 時可請求的最大FlexVol大小。 | （預設不強制執行） |
| `denyNewVolumePools` | 限制 `ontap-san-economy`後端建立新的FlexVol區來包含它們的 LUN。只有預先存在的 Flexvol 才能用於配置新的 PV。 | |

使用 **formatOptions** 的建議

Trident建議採用以下選項來加速格式化流程：

**-E nodiscard:**

- 保留，不要在執行 mkfs 時嘗試丟棄區塊（最初丟棄區塊對固態設備和稀疏/精簡配置儲存很有用）。這取代了已棄用的選項“-K”，並且適用於所有檔案系統（xfs、ext3 和 ext4）。

使用 **Active Directory** 憑證向後端 **SVM** 驗證**Trident** 的身份

您可以設定Trident以使用 Active Directory (AD) 憑證對後端 SVM 進行驗證。在 AD 帳戶可以存取 SVM 之前，您必須設定 AD 網域控制站對叢集或 SVM 的存取權限。對於使用 AD 帳戶進行叢集管理，您必須建立網域隧道。參考 "在ONTAP中設定 Active Directory 網域控制站存取" 了解詳情。

步驟

1. 為後端 SVM 配置網域名稱系統 (DNS) 設定：

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. 執行下列命令在 Active Directory 中為 SVM 建立電腦帳戶：

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. 使用此命令建立 AD 使用者或群組來管理叢集或 SVM

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. 在Trident後端設定檔中，設定 `username` 和 `password` 參數分別為 AD 使用者或群組名稱和密碼。

用於配置磁碟區的後端配置選項

您可以使用下列選項控制預設配置。 `defaults`配置部分。例如，請參閱下面的設定範例。

| 範圍 | 描述 | 預設 |
|---|---|---|
| `spaceAllocation` | LUN 的空間分配 | “true” 如果指定，則設定為 **`true`**適用於**ASA r2** 系統。 |
| `spaceReserve` | 空間預留模式；「無」（細）或「大量」（粗）。 設定為 **`none`**適用於**ASA r2** 系統。 | “沒有任何” |
| `snapshotPolicy` | 要使用的快照策略。 設定為 **`none`**適用於**ASA r2** 系統。 | “沒有任何” |
| `qosPolicy` | 若要為建立的磁碟區指派的 QoS 策略群組。每個儲存池/後端選擇 qosPolicy 或 adaptiveQosPolicy 之一。將 QoS 策略群組與Trident結合使用需要ONTAP 9.8 或更高版本。您應該使用非共享的 QoS 策略群組，並確保該策略群組單獨套用至每個成員。共享的 QoS 策略群組強制規定所有工作負載的總吞吐量上限。 | “” |
| `adaptiveQosPolicy` | 若要為建立的磁碟區指派的自適應 QoS 策略群組。每個儲存池/後端選擇 qosPolicy 或 adaptiveQosPolicy 之一 | “” |
| `snapshotReserve` | 為快照預留的磁碟區百分比。 請勿指定用於**ASA r2** 系統。 | 如果為“0”， `snapshotPolicy`為“無”，否則為“ |
| `splitOnClone` | 創建時將克隆體從其母體中分離出來 | “錯誤的” |
| `encryption` | 在新磁碟區啟用NetApp磁碟區加密 (NVE)；預設為 `false`。若要使用此選項，必須在叢集上取得 NVE 許可並啟用 NVE。如果後端啟用了 NAE，則在Trident中配置的任何磁碟區都會啟用 NAE。更多信息，請參閱：“Trident如何與 NVE 和 NAE 協同工作” 。 | “false” 如果指定，則設定為 **`true`**適用於**ASA r2** 系統。 |
| `luksEncryption` | 啟用LUKS加密。參考“使用 Linux 統一金鑰設定 (LUKS)”。 | 設定為 **`false`**適用於**ASA r2** 系統。 |
| `tieringPolicy` | 分層策略使用「無」 請勿為**ASA r2** 系統指定 。 | |
| `nameTemplate` | 用於建立自訂磁碟區名稱的範本。 | “” |

卷配置範例

以下是一個定義了預設值的範例：

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

對於使用以下方式建立的所有捲 `ontap-san`驅動程式Trident為FlexVol增加了 10% 的額外容量，以容納 LUN 元資料。 LUN 將依照使用者在 PVC 中要求的確切大小進行設定。 Trident使FlexVol增加 10%（在ONTAP中顯示為可用尺寸）。用戶現在將獲得他們所申請的可用容量。此項變更還可以防止 LUN 在可用空間未完全利用之前變為唯讀。這不適用於 ontap-san-economy。

對於定義後端 `snapshotReserve`Trident計算體積大小的方法如下：

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

1.1 是Trident為容納 LUN 元資料而額外添加到FlexVol 的10%。為了 `snapshotReserve`= 5%，PVC 請求 = 5 GiB，總體積大小為 5.79 GiB，可用大小為 5.5 GiB。這 `volume show`該命令應顯示與此範例類似的結果：

```
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
          _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4
                       online     RW         10GB     5.00GB    0%
          _pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d
                       online     RW       5.79GB     5.50GB    0%
          _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba
                       online     RW          1GB    511.8MB    0%
3 entries were displayed.
```

目前，調整大小是將新計算方法應用於現有體積的唯一方法。

以下範例展示了基本配置，其中大多數參數都保留預設值。這是定義後端最簡單的方法。

> 如果您在NetApp ONTAP上使用Amazon FSx和Trident， NetApp建議您為 LIF 指定 DNS 名稱而非 IP 位址。

**ONTAP SAN 範例**

這是使用以下方法的基本配置： `ontap-san`司機。

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

**MetroCluster範例**

您可以設定後端，以避免在切換和切換回後端後手動更新後端定義。"SVM複製與復原" 。

為了實現無縫切換和切換回，請指定 SVM `managementLIF`並省略 `svm`參數。例如：

```
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

**ONTAP SAN 經濟範例**

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

基於憑證的身份驗證範例

在這個基本設定範例中 `clientCertificate`， `clientPrivateKey` ， 和 `trustedCACertificate`（如果使用受信任的 CA，則為可選） `backend.json`分別取客戶端憑證、私鑰和受信任 CA 憑證的 base64 編碼值。

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

### 雙向 CHAP 範例

這些範例創建了一個後端。 `useCHAP`設定為 `true`。

#### ONTAP SAN CHAP 範例

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

#### ONTAP SAN 經濟 CHAP 範例

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

**NVMe/TCP 範例**

您的ONTAP後端必須設定使用 NVMe 的 SVM。這是 NVMe/TCP 的基本後端配置。

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

**SCSI over FC (FCP) 範例**

您的ONTAP後端必須配置 FC 的 SVM。這是 FC 的基本後端配置。

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

使用 **nameTemplate** 的後端設定範例

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

**ontap-san-economy** 驅動程式的 **formatOptions** 範例

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

具有虛擬池的後端範例

在這些範例後端定義檔中，所有儲存池都設定了特定的預設值，例如： `spaceReserve`沒有，`spaceAllocation`為假，並且 `encryption`錯誤。虛擬池在儲存部分中定義。

Trident在「備註」欄位中設定配置標籤。在配置時， Trident會將虛擬池上的所有標籤複製到儲存卷，並在FlexVol volume上設定註解。為了方便起見，儲存管理員可以為每個虛擬池定義標籤，並按標籤將磁碟區分組。

在這些範例中，一些儲存池會設定自己的參數。 `spaceReserve` ， `spaceAllocation` ， 和 `encryption` 有些值會覆蓋預設值，有些池會覆蓋預設值。

## ONTAP SAN 範例

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

**ONTAP SAN 經濟範例**

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
labels:
  store: san_economy_store
region: us_east_1
storage:
  - labels:
      app: oracledb
      cost: "30"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
  - labels:
      app: postgresdb
      cost: "20"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
  - labels:
      app: mysqldb
      cost: "10"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
  - labels:
      department: legal
      creditpoints: "5000"
```

```
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

**NVMe/TCP** 範例

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: "false"
  encryption: "true"
storage:
  - labels:
      app: testApp
      cost: "20"
    defaults:
      spaceAllocation: "false"
      encryption: "false"
```

將後端映射到儲存類

以下 StorageClass 定義指的是[具有虛擬池的後端範例]。使用 `parameters.selector`在欄位中，每個 StorageClass 都會指出哪些虛擬池可用於託管磁碟區。卷將具有所選虛擬池中定義的各個方面。

- 這 `protection-gold`StorageClass 將會對應到第一個虛擬池。 `ontap-san`後端。這是唯一提供黃金級保護的泳池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- 這 `protection-not-gold`StorageClass 將會對應到第二個和第三個虛擬池。 `ontap-san`後端。除了黃金等級之外，只有這些金池提供其他等級的保護。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- 這 `app-mysqldb`StorageClass 將會對應到第三個虛擬池 `ontap-san-economy`後端。這是唯一一個為mysqldb類型應用程式提供儲存池配置的儲存池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- 這 `protection-silver-creditpoints-20k`StorageClass 將會對應到第二個虛擬池 `ontap-san`後端。這是唯一提供銀級保護和 20000 積分的獎金池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- 這 `creditpoints-5k`StorageClass 將會對應到第三個虛擬池 `ontap-san`後端與第四個虛擬池 `ontap-san-economy`後端。這是唯一提供 5000 點的彩池產品。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- 這 `my-test-app-sc`StorageClass 將會對應到 `testAPP`虛擬池 `ontap-san`司機 `sanType: nvme`。這是唯一一家提供泳池的公司 `testApp`。

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

Trident將決定選擇哪個虛擬池，並確保滿足儲存需求。

## ONTAP NAS 驅動程式

### ONTAP NAS 驅動程式概述

了解如何使用ONTAP和Cloud Volumes ONTAP NAS 驅動程式設定ONTAP後端。

**ONTAP NAS** 驅動程式詳情

Trident提供以下 NAS 儲存驅動程序，用於與ONTAP叢集通訊。支援的存取模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

| 司機 | 協定 | 音量模式 | 支援的存取模式 | 支援的檔案系統 |
|---|---|---|---|---|
| `ontap-nas` | NFS SMB | 檔案系統 | RWO、ROX、RWX、RWOP | "， `nfs` ， `smb` |
| `ontap-nas-economy` | NFS SMB | 檔案系統 | RWO、ROX、RWX、RWOP | "， `nfs` ， `smb` |
| `ontap-nas-flexgroup` | NFS SMB | 檔案系統 | RWO、ROX、RWX、RWOP | "， `nfs` ， `smb` |

- 使用 `ontap-san-economy`僅當預計持續卷使用量高於…時"支援的ONTAP容量限制"。
- 使用 `ontap-nas-economy`僅當預計持續卷使用量高於…時"支援的ONTAP容量限制"以及 `ontap-san-economy`驅動程式無法使用。
- 請勿使用 `ontap-nas-economy`如果您預計需要資料保護、災難復原或行動辦公室。
- NetApp不建議在所有ONTAP驅動程式中使用 Flexvol 自動成長，ontap-san 除外。作為變通方法， Trident支援使用快照儲備，並相應地調整 Flexvol 容量。

使用者權限

Trident預期以ONTAP或 SVM 管理員身分執行，通常使用下列方式： `admin`集群用戶或 `vsadmin`SVM 用戶，或具有相同角色但名稱不同的用戶。

對於Amazon FSx for NetApp ONTAP部署， Trident需要以ONTAP或 SVM 管理員身分執行，並使用叢集。`fsxadmin`用戶或 `vsadmin`SVM 用戶，或具有相同角色但名稱不同的用戶。這 `fsxadmin`用戶是集群管理員用戶的有限替代品。

如果你使用 `limitAggregateUsage`需要參數和叢集管理員權限。當使用Amazon FSx for NetApp ONTAP和Trident時， `limitAggregateUsage`參數將無法與 `vsadmin`和 `fsxadmin`用戶帳戶。如果指定此參數，配置操作將失敗。

雖然可以在ONTAP中建立一個限制性更強的角色供Trident驅動程式使用，但我們不建議這樣做。 Trident的大多數新版本都會呼叫額外的 API，這些 API 必須加以考慮，這使得升級變得困難且容易出錯。

準備配置帶有**ONTAP NAS** 驅動程式的後端

## 了解配置具有ONTAP NAS 驅動程式的ONTAP後端的要求、驗證選項和匯出策略。

要求

- 對於所有ONTAP後端， Trident要求至少將一個聚合分配給 SVM。
- 您可以執行多個驅動程序，並建立指向其中一個或另一個驅動程式的儲存類別。例如，您可以設定一個使用以下方式的 Gold 類別： `ontap-nas`駕駛員和使用青銅級的駕駛員 `ontap-nas-economy`一。

- 所有 Kubernetes 工作節點都必須安裝對應的 NFS 工具。請參閱"這裡"更多詳情請見下文。
- Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。參考 準備配置SMB卷 了解詳情。

#### 對ONTAP後端進行身份驗證

Trident提供兩種ONTAP後端身分驗證方式。

- 基於憑證：此模式需要對ONTAP後端擁有足夠的權限。建議使用與預先定義的安全登入角色關聯的帳戶，例如： `admin`或者 `vsadmin`確保與ONTAP版本最大程度相容。
- 基於證書：此模式要求後端安裝證書，以便Trident才能與ONTAP叢集通訊。此處，後端定義必須包含用戶端憑證、金鑰和受信任 CA 憑證（如果使用，建議使用）的 Base64 編碼值。

您可以更新現有後端，以在基於憑證的方法和基於憑證的方法之間進行切換。但是，一次只能支援一種身份驗證方法。要切換到不同的身份驗證方法，必須從後端配置中刪除現有方法。

> 如果您嘗試同時提供憑證和證書，則後端建立將會失敗，並出現錯誤，提示設定檔中提供了多個驗證方法。

##### 啟用基於憑證的身份驗證

Trident需要 SVM 範圍/叢集範圍管理員的憑證才能與ONTAP後端通訊。建議使用標準的、預先定義的角色，例如： `admin`或者 `vsadmin`。這樣可以確保與未來ONTAP版本向前相容，這些版本可能會公開一些功能 API，供未來的Trident版本使用。雖然可以建立自訂安全登入角色並將其與Trident一起使用，但不建議這樣做。

後端定義範例如下圖所示：

**YAML**

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

**JSON**

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

請注意，後端定義是唯一以純文字形式儲存憑證的地方。後端建立完成後，使用者名稱/密碼將使用 Base64 進行編碼，並儲存為 Kubernetes 金鑰。後端建立/更新是唯一需要了解憑證的步驟。因此，這是一項僅限管理員執行的操作，由 Kubernetes/儲存管理員執行。

啟用基於憑證的身份驗證

新的和現有的後端都可以使用憑證與ONTAP後端通訊。後端定義需要三個參數。

- clientCertificate：客戶端憑證的 Base64 編碼值。
- clientPrivateKey：關聯私鑰的 Base64 編碼值。
- trustedCACertificate：受信任 CA 憑證的 Base64 編碼值。如果使用受信任的 CA，則必須提供此參數。如果沒有使用受信任的憑證授權機構，則可以忽略此步驟。

典型的工作流程包括以下步驟。

步驟

1. 產生客戶端憑證和金鑰。產生時，將通用名稱 (CN) 設定為要進行驗證的ONTAP使用者。

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. 在ONTAP叢集中新增受信任的 CA 憑證。這可能已經由儲存管理員處理了。如果沒有使用受信任的憑證授權機構，則忽略此操作。

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. 在ONTAP叢集上安裝客戶端憑證和金鑰（來自步驟 1）。

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. 確認ONTAP安全登入角色支持 `cert`身份驗證方法。

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. 使用產生的憑證進行身份驗證。請將 < ONTAP管理 LIF> 和 <vserver 名稱> 替換為管理 LIF IP 位址和 SVM 名稱。您必須確保 LIF 的服務策略已設定為 `default-data-management`。

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. 使用 Base64 對憑證、金鑰和受信任的 CA 憑證進行編碼。

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 使用上一步獲得的值建立後端。

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

更新身份驗證方法或輪換憑證

您可以更新現有後端，以使用不同的身份驗證方法或輪換其憑證。這種方法是雙向的：使用使用者名稱/密碼的後端可以更新為使用憑證；使用憑證的後端可以更新為基於使用者名稱/密碼的後端。為此，您必須刪除現有的身份驗證方法並新增新的身份驗證方法。然後使用包含所需參數的更新後的 backend.json 檔案來執行 `tridentctl update backend`。

```
cat cert-backend-updated.json
```

```
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}
```

```
#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

輪換密碼時，儲存管理員必須先更新ONTAP上使用者的密碼。接下來將進行後端更新。輪換證書時，可以為使用者新增多個證書。然後更新後端以使用新證書，之後即可從ONTAP叢集中刪除舊證書。

更新後端不會中斷對已建立磁碟區的訪問，也不會影響之後建立的磁碟區連線。後端更新成功表明Trident可以與ONTAP後端通訊並處理未來的磁碟區操作。

### 為Trident建立自訂ONTAP角色

您可以建立一個具有最低權限的ONTAP叢集角色，這樣您就不必使用ONTAP管理員角色在Trident中執行操作。在Trident後端設定中包含使用者名稱時， Trident將使用您建立的ONTAP叢集角色來執行操作。

請參閱"Trident自訂角色產生器"有關建立Trident自訂角色的詳細資訊。

使用**ONTAP CLI**

1. 使用以下命令建立新角色：

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. 為Trident用戶建立使用者名稱：

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver
<svm_name\> -comment "user_description"
```

3. 將角色映射到使用者：

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

使用系統管理員

在ONTAP系統管理員中執行下列步驟：

1. 建立自訂角色：
   a. 若要在叢集層級建立自訂角色，請選擇「叢集 > 設定」。

      （或）若要在 SVM 層級建立自訂角色，請選擇「儲存」>「儲存虛擬機器」> `required SVM` > 設定 > 使用者和角色*。

   b. 選擇“使用者和角色”旁邊的箭頭圖示（→）。
   c. 在“角色”下選擇“+添加”。
   d. 定義角色規則，然後點選「儲存」。
2. 將角色對應到Trident使用者： + 在「使用者和角色」頁面上執行下列步驟：
   a. 在「使用者」下方選擇「新增」圖示 **+**。
   b. 選擇所需的使用者名，然後在「角色」下拉式選單中選擇角色。
   c. 點選“儲存”。

更多資訊請參閱以下頁面：

- "用於管理ONTAP的自訂角色"或者"定義自訂角色"
- "與角色和使用者協作"

#### 管理 NFS 導出策略

Trident使用 NFS 匯出策略來控制對其所配置磁碟區的存取。

Trident在處理出口策略時提供兩種選擇：

- Trident可以動態管理匯出策略本身；在這種操作模式下，儲存管理員指定 CIDR 區塊列表，這些 CIDR 區塊代表可接受的 IP 位址。 Trident會在發佈時自動將屬於這些範圍的適用節點 IP 新增至匯出原則。或者，如果沒有指定 CIDR，則會將發布捲的節點上找到的所有全域範圍的單播 IP 新增至匯出策略。
- 儲存管理員可以建立匯出策略並手動新增規則。除非在組態中指定了不同的導出策略名稱，否則Trident將使用預設導出策略。

動態管理出口策略

Trident提供了動態管理ONTAP後端匯出策略的功能。這樣，儲存管理員就可以指定工作節點 IP 的允許位址空間，而無需手動定義明確的規則。它大大簡化了導出策略管理；對導出策略的修改不再需要對儲存叢集進行人工幹預。此外，這有助於將對儲存叢集的存取限制在僅允許掛載磁碟區且 IP 位址在指定範圍內的節點上，從而支援細粒度和自動化管理。

使用動態匯出策略時，請勿使用網路位址轉換（NAT）。使用 NAT 時，儲存控制器看到的是前端 NAT 位址而不是實際的 IP 主機位址，因此在匯出規則中找不到匹配項時，存取將被拒絕。

例子

必須使用兩種配置選項。以下是一個後端定義範例：

```
---
version: 1
storageDriverName: ontap-nas-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
  - 192.168.0.0/24
autoExportPolicy: true
```

使用此功能時，必須確保 SVM 中的根連接點具有先前建立的匯出策略，該策略的匯出規則允許節點 CIDR 區塊（例如預設匯出策略）。始終遵循NetApp推薦的最佳實踐，為Trident專用一個 SVM。

以下以上述範例為例，說明此功能的工作原理：

- `autoExportPolicy`設定為 `true`。這表示Trident會為使用從後端配置的每個磁碟區建立一個匯出策略。`svm1`使用 SVM 處理規則的新增和刪除 `autoexportCIDRs`地址塊。在磁碟區連接到節點之前，該磁碟區使用空的匯出策略，沒有任何規則來防止對該磁碟區的未經授權的存取。當磁碟區發佈到節點時，Trident會建立一個匯出策略，該策略的名稱與包含指定 CIDR 區塊內節點 IP 的底層 qtree 的名稱相同。這些 IP 位址也會加入到父FlexVol volume所使用的匯出策略中。
  - 例如：
    - 後端 UUID 403b5326-8482-40db-96d0-d83fb3f4daec
    - `autoExportPolicy`設定為 `true`

- 儲存前綴 `trident`
- PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
- 名為 trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c 的 qtree 為名為FlexVol的 FlexVol 建立了一個匯出策略。 `trident-403b5326-8482-40db96d0-d83fb3f4daec` ，名為 qtree 的匯出策略 `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`以及一個名為「空的匯出策略」的 `trident_empty`在支援向量機上。 FlexVol導出策略的規則將是 qtree 導出策略中包含的任何規則的超集。任何未附加的磁碟區都將重複使用空匯出策略。

- `autoExportCIDRs`包含地址塊列表。此字段為可選字段，預設值為 ["0.0.0.0/0", "::/0"]。如果未定義，Trident會新增在工作節點上找到的所有具有發佈的全域作用域的單播位址。

在這個例子中， `192.168.0.0/24`已提供地址空間。這意味著，位於此位址範圍內且已發佈 Kubernetes 節點 IP 的節點將被加入到Trident建立的匯出策略中。當Trident註冊它運行所在的節點時，它會檢索該節點的 IP 位址，並將其與提供的位址區塊進行比對。 `autoExportCIDRs`在發佈時， Trident在過濾 IP 位址後，會為它要發佈到的節點的用戶端 IP 位址建立匯出政策規則。

您可以更新 `autoExportPolicy`和 `autoExportCIDRs`建立後端之後，需要進行以下操作。您可以為自動管理的後端新增新的 CIDR，或刪除現有的 CIDR。刪除 CIDR 時要格外小心，確保現有連線不會中斷。您也可以選擇停用 `autoExportPolicy`對於後端，如果出現問題，則回退到手動建立的匯出策略。這將需要進行設置 `exportPolicy`後端配置中的參數。

Trident建立或更新後端後，您可以使用下列指令檢查後端： `tridentctl`或相應的 `tridentbackend`CRD：

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

當一個節點被移除時， Trident會檢查所有匯出策略，以移除與該節點對應的存取規則。透過從受管後端匯出政策移除此節點 IP， Trident可以防止惡意掛載，除非叢集中的新節點重新使用此 IP。

對於先前存在的後端，使用以下方式更新後端： `tridentctl update backend`確保Trident自動管理出口策略。這樣，當需要時，就會建立兩個以以後端 UUID 和 qtree 名稱命名的新匯出策略。後端存在的磁碟區在卸載並重新掛載後，將使用新建立的匯出策略。

> ⓘ 刪除具有自動管理匯出策略的後端將刪除動態建立的匯出策略。如果後端被重新創建，它將被視為一個新的後端，並將導致創建一個新的匯出策略。

如果正在執行中的節點的 IP 位址更新，則必須重新啟動該節點上的Trident pod。 Trident隨後將更新其管理的後端的匯出策略，以反映此 IP 變更。

#### 準備配置SMB卷

稍作準備，您就可以使用以下方式設定 SMB 卷 `ontap-nas`司機。

> ⓘ 您必須在 SVM 上同時設定 NFS 和 SMB/CIFS 協定才能建立 `ontap-nas-economy`適用於ONTAP本地叢集的 SMB 磁碟區。如果未能配置這些協定中的任何一個，都會導致 SMB 磁碟區建立失敗。

> ⓘ `autoExportPolicy`不支援 SMB 磁碟區。

開始之前

在配置 SMB 磁碟區之前，您必須具備以下條件。

- 一個 Kubernetes 叢集，包含一個 Linux 控制器節點和至少一個執行 Windows Server 2022 的 Windows 工作節點。 Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。
- 至少有一個包含您的 Active Directory 憑證的Trident金鑰。生成秘密 `smbcreds`：

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- 配置為 Windows 服務的 CSI 代理程式。要配置 `csi-proxy`，請參閱"GitHub：CSI代理"或者"GitHub：適用於 Windows 的 CSI 代理"適用於在 Windows 上執行的 Kubernetes 節點。

步驟

1. 對於本地部署的ONTAP，您可以選擇建立 SMB 共享，或者Trident可以為您建立一個。

   > ⓘ Amazon FSx for ONTAP需要 SMB 共用。

   您可以透過以下兩種方式之一建立 SMB 管理共用："Microsoft 管理控制台"共用資料夾管理單元或使用ONTAP CLI。使用ONTAP CLI 建立 SMB 共享：

   a. 如有必要，請建立共用的目錄路徑結構。

      這 `vserver cifs share create`此指令檢查在建立共用時 -path 選項中指定的路徑。如果指定的路徑不存在，則命令執行失敗。

   b. 建立與指定 SVM 關聯的 SMB 共用：

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

c. 確認共享已建立：

```
vserver cifs share show -share-name share_name
```

請參閱"建立 SMB 共享"了解詳細資訊。

2. 建立後端時，必須配置以下內容以指定 SMB 磁碟區。有關所有 FSx for ONTAP後端設定選項，請參閱"FSx for ONTAP設定選項和範例"。

| 範圍 | 描述 | 例子 |
|---|---|---|
| `smbShare` | 您可以指定以下一項：使用 Microsoft 管理控制台或ONTAP CLI 建立的 SMB 共用的名稱；允許Trident建立 SMB 共用的名稱；或者您可以將參數留空以防止對磁碟區的公共共用存取。對於本地部署的ONTAP，此參數是可選的。此參數是Amazon FSx for ONTAP後端所必需的，不能為空。 | `smb-share` |
| `nasType` | *必須設定為 `smb`.*如果為空，則預設為空。 `nfs` 。 | `smb` |
| `securityStyle` | 新磁碟區的安全樣式。 *必須設定為 `ntfs`或者 `mixed`適用於 SMB 卷。 * | `ntfs`或者 `mixed`適用於 SMB 卷 |
| `unixPermissions` | 新卷模式。 *對於 SMB 卷，此項必須留空。 * | “” |

啟用安全的 **SMB**

從 25.06 版本開始， NetApp Trident支援使用以下方式安全配置建立的 SMB 磁碟區： `ontap-nas`和 `ontap-nas-economy`後端。啟用安全 SMB 後，您可以使用存取控制清單 (ACL) 為 Active Directory (AD) 使用者和使用者群組提供對 SMB 共用的受控存取。

需要記住的要點

- 輸入 `ontap-nas-economy`不支援卷。
- 僅支援唯讀克隆 `ontap-nas-economy`卷。
- 如果啟用了安全 SMB， Trident將忽略後端提到的 SMB 共用。
- 更新 PVC 註解、儲存類別註解和後端欄位不會更新 SMB 共用 ACL。
- 克隆 PVC 註釋中指定的 SMB 共用 ACL 將優先於來源 PVC 中的 ACL。
- 啟用安全 SMB 時，請確保提供有效的 AD 使用者。無效使用者將不會被加入到存取控制清單 (ACL) 中。
- 如果在後端、儲存類別和 PVC 中為同一個 AD 使用者提供不同的權限，則權限優先權為：PVC、儲存類，然後是後端。
- 支援安全 SMB `ontap-nas`僅適用於託管磁碟區匯入，不適用於非託管磁碟區匯入。

步驟

1. 在 TridentBackendConfig 中指定 adAdminUser，如下例所示：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. 在儲存類別中加入註解。

   添加 `trident.netapp.io/smbShareAdUser`對儲存類別進行註解，以啟用安全可靠的 SMB 連線。使用者為註釋指定的值 `trident.netapp.io/smbShareAdUser`應該與指定的使用者名稱相同 `smbcreds`秘密。您可以從以下選項中選擇一項 `smbShareAdUserPermission：full_control ，change ， 或者 read。預設權限是 full_control。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. 製作PVC管。

以下範例建立PVC：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

**ONTAP NAS** 設定選項和範例

學習如何在Trident系統中建立和使用ONTAP NAS 驅動程式。本節提供後端設定範例以及將後端對應到 StorageClasses 的詳細資訊。

後端配置選項

請參閱下表以了解後端配置選項：

| 範圍 | 描述 | 預設 |
|---|---|---|
| `version` | | 始終為 1 |
| `storageDriverName` | 儲存驅動程式的名稱 | `ontap-nas`， `ontap-nas-economy` ， 或者 `ontap-nas-flexgroup` |
| `backendName` | 自訂名稱或儲存後端 | 驅動程式名稱 + "_" + dataLIF |
| `managementLIF` | 叢集或 SVM 管理 LIF 的 IP 位址可以指定完全限定網域名稱 (FQDN)。如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。 IPv6 位址必須用方括號定義，例如： `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 。為了實現MetroCluster 的無縫切換，請參閱MetroCluster範例。 | "10.0.0.1", "[2001:1234:abcd::fefe]" |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| `dataLIF` | LIF協定的IP位址。 NetApp建議指定 `dataLIF`。如果未提供， Trident將從 SVM 取得 dataLIF。您可以指定一個完全限定網域名稱 (FQDN) 用於 NFS 掛載操作，從而建立輪詢 DNS 以在多個 dataLIF 之間進行負載平衡。初始設定後可以更改。參考。如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。 IPv6 位址必須用方括號定義，例如： `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 。 *Metrocluster 除外。 *查看MetroCluster範例。 | 指定位址或從 SVM 派生，如果未指定（不建議） |
| `svm` | 要使用的儲存虛擬機器 *Metrocluster 除外。 *查看MetroCluster範例。 | 如果是 SVM 則推導而來 `managementLIF`已指定 |
| `autoExportPolicy` | 啟用自動匯出策略建立和更新 [布林值]。使用 `autoExportPolicy`和 `autoExportCIDRs`選用功能包括： Trident可以自動管理出口策略。 | 錯誤的 |
| `autoExportCIDRs` | 用於過濾 Kubernetes 節點 IP 的 CIDR 列表 `autoExportPolicy`已啟用。使用 `autoExportPolicy`和 `autoExportCIDRs`選用功能包括： Trident可以自動管理出口策略。 | ["0.0.0.0/0", "::/0"]` |
| `labels` | 若要套用於磁碟區的任意 JSON 格式標籤集 | “” |
| `clientCertificate` | 用戶端憑證的 Base64 編碼值。用於基於憑證的身份驗證 | “” |
| `clientPrivateKey` | 客戶端私鑰的 Base64 編碼值。用於基於憑證的身份驗證 | “” |
| `trustedCACertificate` | 受信任 CA 憑證的 Base64 編碼值。選修的。用於基於憑證的身份驗證 | “” |
| `username` | 用於連接到叢集/SVM 的使用者名稱。用於基於憑證的身份驗證。有關 Active Directory 驗證，請參閱 "使用 Active Directory 憑證向後端 SVM 驗證Trident 的身份"。 | |
| `password` | 連接到叢集/SVM 的密碼。用於基於憑證的身份驗證。有關 Active Directory 驗證，請參閱 "使用 Active Directory 憑證向後端 SVM 驗證Trident 的身份"。 | |
| `storagePrefix` | 在 SVM 中配置新磁碟區時所使用的前綴。設定後無法更新<br>當使用 ontap-nas-economy 和 24 個或更多字元的 storagePrefix 時，qtree 將不會嵌入儲存前綴，儘管它會包含在磁碟區名稱中。 | “三叉戟” |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| `aggregate` | 用於配置的聚合（可選；如果設置，則必須指派給SVM）。對於 `ontap-nas-flexgroup`驅動程序，此選項將被忽略。如果未分配，則可以使用任何可用的聚合來配置FlexGroup磁碟區。<br><br>當 SVM 中的聚合資料更新時， Trident會自動輪詢 SVM 進行更新，而無需重新啟動Trident控制器。當您在Trident中配置特定聚合以配置磁碟區時，如果該聚合被重新命名或移出 SVM，則在輪詢 SVM 聚合時， Trident中的後端將變為失敗狀態。您必須將聚合變更為 SVM 上存在的聚合，或將其完全刪除，才能使後端恢復連線。 | “” |
| `limitAggregateUsage` | 如果使用率超過此百分比，則配置失敗。 不適用於**Amazon FSx for ONTAP**。 | （預設不強制執行） |
| flexgroupAggregateList | 用於配置的聚合清單（可選；如果設置，則必須指派給 SVM）。指派給 SVM 的所有聚合都用於配置FlexGroup磁碟區。支援 **ontap-nas-flexgroup** 儲存驅動程式。<br><br>當 SVM 中的聚合列表更新時， Trident會透過輪詢 SVM 自動更新該列表，而無需重新啟動Trident控制器。當您在Trident中設定了特定的聚合清單來設定磁碟區時，如果聚合清單被重新命名或移出 SVM，則在輪詢 SVM 聚合時，Trident中的後端將進入失敗狀態。您必須將聚合列表變更為 SVM 上存在的列表，或將其完全刪除，才能使後端恢復連線。 | “” |
| `limitVolumeSize` | 如果請求的磁碟區大小大於此值，則配置失敗。此外，它還限制了 qtree 管理的捲的最大大小，以及`qtreesPerFlexvol`此選項允許自訂每個FlexVol volume的最大 qtree 數量。 | （預設不強制執行） |
| `debugTraceFlags` | 故障排除時要使用的調試標誌。例如，{"api":false, "method":true} 請勿使用 `debugTraceFlags`除非您正在進行故障排除並且需要詳細的日誌轉儲。 | 無效的 |
| `nasType` | 配置 NFS 或 SMB 磁碟區的建立。選項有 `nfs`，`smb`或空值。設定為 null 則預設使用 NFS 磁碟區。 | `nfs` |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| nfsMountOptions | 以逗號分隔的 NFS 掛載選項清單。 Kubernetes 持久性磁碟區的掛載選項通常在儲存類別中指定，但如果儲存類別中沒有指定掛載選項， Trident將回退到使用儲存後端設定檔中指定的掛載選項。如果在儲存類別或設定檔中未指定任何掛載選項， Trident將不會在關聯的持久性磁碟區上設定任何掛載選項。 | “” |
| qtreesPerFlexvol | 每個FlexVol 的最大 Qtree 數量必須在 [50, 300] 範圍內 | “200” |
| smbShare | 您可以指定以下一項：使用 Microsoft 管理控制台或ONTAP CLI 建立的 SMB 共用的名稱；允許Trident建立 SMB 共用的名稱；或者您可以將參數留空以防止對磁碟區的公共共用存取。對於本地部署的ONTAP，此參數是可選的。此參數是Amazon FSx for ONTAP後端所必需的，不能為空。 | smb-share |
| useREST | 使用ONTAP REST API 的布林參數。 `useREST`設定為 `true`Trident使用ONTAP REST API 與後端通訊；當設定為 `false`Trident使用 ONTAPI (ZAPI) 呼叫與後端通訊。此功能需要ONTAP 9.11.1 及更高版本。此外，所使用的ONTAP登入角色必須具有存取權限。 `ontapi`應用。預定義項滿足了這一點。 `vsadmin`和 `cluster-admin`角色。從Trident 24.06 版本和ONTAP 9.15.1 或更高版本開始， `useREST`設定為 `true`預設；更改 `useREST`到 `false`使用 ONTAPI (ZAPI) 呼叫。 | true`適用於ONTAP 9.15.1 或更高版本，否則 `false。 |
| limitVolumePoolSize | 在 ontap-nas-economy 後端使用 Qtrees 時可請求的最大FlexVol大小。 | （預設不強制執行） |
| denyNewVolumePools | 限制 `ontap-nas-economy`後端創建新的FlexVol磁碟區來包含它們的 Qtree。只有預先存在的 Flexvol 才能用於配置新的 PV。 | |
| adAdminUser | 具有對 SMB 共用完全存取權限的 Active Directory 管理員使用者或使用者群組。使用此參數可為 SMB 共用提供管理員權限，並賦予其完全控制權限。 | |

用於配置磁碟區的後端配置選項

您可以使用下列選項控制預設配置。 `defaults`配置部分。例如，請參閱下面的設定範例。

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| spaceAllocation | Q樹的空間分配 | “真的” |
| spaceReserve | 空間預留模式；「無」（細）或「大量」（粗） | “沒有任何” |
| snapshotPolicy | 要使用的快照策略 | “沒有任何” |
| qosPolicy | 若要為建立的磁碟區指派的 QoS 策略群組。每個儲存池/後端選擇 qosPolicy 或 adaptiveQosPolicy 之一 | “” |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| `adaptiveQosPolicy` | 若要為建立的磁碟區指派的自適應 QoS 策略群組。每個儲存池/後端選擇 qosPolicy 或 adaptiveQosPolicy 之一。 ontap-nas-economy 不支援此功能。 | “” |
| `snapshotReserve` | 快照預留的磁碟區百分比 | 如果為“0”， `snapshotPolicy`為“無”，否則為“ |
| `splitOnClone` | 創建時將克隆體從其母體中分離出來 | “錯誤的” |
| `encryption` | 在新磁碟區啟用NetApp磁碟區加密 (NVE)；預設為 `false`。若要使用此選項，必須在叢集上取得 NVE 許可並啟用 NVE。如果後端啟用了 NAE，則在Trident中配置的任何磁碟區都會啟用 NAE。更多信息，請參閱："Trident如何與 NVE 和 NAE 協同工作" 。 | “錯誤的” |
| `tieringPolicy` | 分層策略使用“無” | |
| `unixPermissions` | 新卷模式 | NFS 磁碟區的連接埠號碼為「777」；SMB 磁碟區的連接埠號碼為空（不適用）。 |
| `snapshotDir` | 控制對以下方面的訪問 `.snapshot`目錄 | NFSv4 為“true”，NFSv3 為“false”。 |
| `exportPolicy` | 出口政策的使用 | "預設" |
| `securityStyle` | 新磁碟區的安全樣式。 NFS 支持 `mixed`和 `unix`安全措施。中小企業支持 `mixed`和 `ntfs`安全措施。 | NFS 預設值為 `unix`。 SMB 預設值為 `ntfs`。 |
| `nameTemplate` | 用於建立自訂磁碟區名稱的範本。 | “” |

將 QoS 策略群組與Trident結合使用需要ONTAP 9.8 或更高版本。您應該使用非共享的 QoS 策略群組，並確保該策略群組單獨套用至每個成員。共享的 QoS 策略群組強制規定所有工作負載的總吞吐量上限。

卷配置範例

以下是一個定義了預設值的範例：

```
---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: <password>
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: "10"
```

為了 ontap-nas`和 `ontap-nas-flexgroups`Trident現在使用新的計算方法，以確保FlexVol 的尺寸與 snapshotReserve 百分比和 PVC 正確匹配。當使用者要求 PVC 時， Trident會使用新的計算方法建立具有更多空間的原始FlexVol 。此計算方法可確保使用者在 PVC 中獲得其請求的可寫入空間，而不是少於其請求的空間。在 v21.07 之前，當使用者要求 PVC（例如 5 GiB）時，如果快照預留百分比為 50%，則只能獲得 2.5 GiB 的可寫入空間。這是因為使用者要求的是整個磁碟區。 `snapshotReserve`是其中的百分比。在Trident 21.07 中，使用者要求的是可寫入空間，而Trident定義了該空間。 `snapshotReserve`將數值表示為佔總體積的百分比。這不適用於 `ontap-nas-economy。請參閱以下範例以了解其工作原理：

計算方法如下：

```
Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)
```

對於快照預留 = 50% 且 PVC 請求 = 5 GiB 的情況，總磁碟區大小為 5/.5 = 10 GiB，可用大小為 5 GiB，這正是使用者在 PVC 請求中請求的大小這 `volume show`該命令應顯示與此範例類似的結果：


升級Trident時，先前安裝的現有後端將以上述方式設定磁碟區。對於升級前建立的捲，您應該調整其大小以觀察到變更。例如，一個 2 GiB 的 PVC `snapshotReserve=50`先前的結果是一個提供 1 GiB 可寫空間的磁碟區。例如，將磁碟區大小調整為 3 GiB，將在 6 GiB 的磁碟區上為應用程式提供 3 GiB 的可寫入空間。

最小配置範例

以下範例展示了基本配置，其中大多數參數都保留預設值。這是定義後端最簡單的方法。

> 如果您在NetApp ONTAP上使用Amazon FSx和Trident，建議為 LIF 指定 DNS 名稱而非 IP 位址。

**ONTAP NAS 經濟範例**

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

**ONTAP NAS Flexgroup 範例**

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

**MetroCluster範例**

您可以設定後端，以避免在切換和切換回後端後手動更新後端定義。"SVM複製與復原" 。

為了實現無縫切換和切換回，請指定 SVM `managementLIF`並省略 `dataLIF`和 `svm`參數。例如：

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

**SMB 磁碟區範例**

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

基於憑證的身份驗證範例

這是一個最小後端設定範例。 `clientCertificate` ， `clientPrivateKey` ， 和 `trustedCACertificate`（如果使用受信任的 CA，則為可選） `backend.json`分別取客戶端憑證、私鑰和受信任 CA 憑證的 base64 編碼值。

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

自動出口政策範例

本範例向您展示如何指示Trident使用動態匯出策略來自動建立和管理匯出策略。這對於以下情況也適用：`ontap-nas-economy`和 `ontap-nas-flexgroup`司機。

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

**IPv6** 位址範例

這個例子表明 `managementLIF`使用 IPv6 位址。

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

使用 **SMB** 磁碟區的**Amazon FSx for ONTAP**範例

這 `smbShare`使用 SMB 磁碟區的 FSx for ONTAP需要此參數。

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

使用 **nameTemplate** 的後端設定範例

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

具有虛擬池的後端範例

在下方所示的範例後端定義檔中，所有儲存池都設定了特定的預設值，例如： `spaceReserve`沒有，`spaceAllocation`為假，並且 `encryption`錯誤。虛擬池在儲存部分中定義。

Trident在「備註」欄位中設定配置標籤。 FlexVol上已設定評論 `ontap-nas`或`FlexGroup `ontap-nas-flexgroup`。 Trident在設定時會將虛擬池上的所有標籤複製到儲存磁碟區。為了方便起見，儲存管理員可以為每個虛擬池定義標籤，並按標籤將磁碟區分組。

在這些範例中，一些儲存池會設定自己的參數。 `spaceReserve`， `spaceAllocation`， 和 `encryption`有些值會覆蓋預設值，有些池會覆蓋預設值。

**ONTAP NAS 範例**

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: "false"
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      app: msoffice
      cost: "100"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
      adaptiveQosPolicy: adaptive-premium
  - labels:
      app: slack
      cost: "75"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
```

```
      app: wordpress
      cost: "50"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      app: mysqldb
      cost: "25"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

**ONTAP NAS FlexGroup範例**

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "50000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: gold
      creditpoints: "30000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      protection: bronze
      creditpoints: "10000"
    zone: us_east_1d
```

```
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

**ONTAP NAS 經濟範例**

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: nas_economy_store
region: us_east_1
storage:
  - labels:
      department: finance
      creditpoints: "6000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: engineering
      creditpoints: "3000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      department: humanresource
      creditpoints: "2000"
    zone: us_east_1d
    defaults:
```

```
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

將後端映射到儲存類

以下 StorageClass 定義指的是[具有虛擬池的後端範例]。使用 `parameters.selector`在欄位中，每個 StorageClass 都會指出哪些虛擬池可用於託管磁碟區。卷將具有所選虛擬池中定義的各個方面。

- 這 `protection-gold`StorageClass 將會對應到第一個和第二個虛擬池。 `ontap-nas-flexgroup`後端。只有這些泳池提供黃金級保護。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- 這 `protection-not-gold`StorageClass 將會對應到第三個和第四個虛擬池。 `ontap-nas-flexgroup`後端。除了黃金之外，只有這些金池提供其他等級的保護。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- 這 `app-mysqldb`StorageClass 將會對應到第四個虛擬池。 `ontap-nas`後端。這是唯一一個為mysqldb類型應用程式提供儲存池配置的儲存池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- 該 `protection-silver-creditpoints-20k`StorageClass 將會對應到第三個虛擬池。 `ontap-nas-flexgroup`後端。這是唯一提供銀級保護和 20000 積分的彩池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- 這 `creditpoints-5k`StorageClass 將會對應到第三個虛擬池。 `ontap-nas`後端和第二個虛擬池 `ontap-nas-economy`後端。這是唯一提供 5000 點的彩池產品。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Trident將決定選擇哪個虛擬池，並確保滿足儲存需求。

#### 更新 `dataLIF`初始配置後

初始設定完成後，您可以透過執行以下命令來變更 dataLIF，從而為新的後端 JSON 檔案提供更新的 dataLIF。

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

如果 PVC 連接到一個或多個艙體，則必須將所有相應的艙體放下，然後再將它們重新升起，以便新的 dataLIF 生效。

安全的中小企業範例

使用 **ontap-nas** 驅動程式進行後端配置

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

使用 **ontap-nas-economy** 驅動程式進行後端配置

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

後端配置及儲存池

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

使用 **ontap-nas** 驅動程式的儲存類別範例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

請確保添加 `annotations`實作安全的SMB。如果沒有註釋，無論後端或 PVC 中設定了什麼配置，安全 SMB 都無法運作。

使用 **ontap-nas-economy** 驅動程式的儲存類別範例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

包含單一 **AD** 使用者的 **PVC** 範例

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

包含多個 **AD** 使用者的 **PVC** 範例

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

## Amazon FSx for NetApp ONTAP

### 將Trident與Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP"是一項完全託管的 AWS 服務，可讓客戶啟動和執行由NetApp ONTAP儲存作業系統支援的檔案系統。 FSx for ONTAP可讓您利用您熟悉的NetApp功能、效能和管理能力，同時享受在 AWS 上儲存資料的簡單性、敏捷性、安全性和可擴充性。 FSx for ONTAP支援ONTAP檔案系統功能和管理 API。

您可以將Amazon FSx for NetApp ONTAP檔案系統與Trident集成，以確保在 Amazon Elastic Kubernetes Service (EKS) 中執行的 Kubernetes 叢集可以設定由ONTAP支援的區塊和檔案持久磁碟區。

檔案系統是Amazon FSx中的主要資源，類似本地的ONTAP叢集。在每個 SVM 中，您可以建立一個或多個卷，這些卷是用於儲存檔案系統中的檔案和資料夾的資料容器。 Amazon FSx for NetApp ONTAP將作為雲端託管檔案系統提供。新的檔案系統類型稱為 * NetApp ONTAP*。

透過將Trident與Amazon FSx for NetApp ONTAP結合使用，您可以確保在 Amazon Elastic Kubernetes Service (EKS) 中執行的 Kubernetes 叢集可以設定由ONTAP支援的區塊和檔案持久性磁碟區。

要求

另外"Trident的要求"要將 FSx for ONTAP與Trident集成，您需要：

- 現有的 Amazon EKS 叢集或自管理 Kubernetes 叢集 `kubectl`已安裝。
- 叢集工作節點可存取的現有Amazon FSx for NetApp ONTAP檔案系統和儲存虛擬機器 (SVM)。
- 已準備好的工作節點"NFS 或 iSCSI"。

> 請務必按照 Amazon Linux 和 Ubuntu 所需的節點準備步驟進行操作。 "亞馬遜機器圖像"（AMI）取決於您的 EKS AMI 類型。

注意事項

- SMB 卷：
  - 使用以下方式支援 SMB 卷 `ontap-nas`僅限司機。
  - Trident EKS 外掛程式不支援 SMB 磁碟區。
  - Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。參考 "準備配置SMB卷" 了解詳情。
- 在Trident 24.02 之前，在Amazon FSx檔案系統上建立的、啟用了自動備份的磁碟區無法被Trident刪除。為防止在Trident 24.02 或更高版本中出現此問題，請指定 `fsxFilesystemID`，AWS `apiRegion` ，AWS `apikey`以及 AWS `secretKey`在 AWS FSx for ONTAP的後端設定檔中。

> 如果您要為Trident指定 IAM 角色，則可以省略指定以下內容： `apiRegion` ， `apiKey` ， 和 `secretKey`明確地將欄位傳遞給Trident 。更多信息，請參閱"FSx for ONTAP設定選項和範例"。

同時使用**Trident SAN/iSCSI** 和 **EBS-CSI** 驅動程式

如果您打算將 ontap-san 驅動程式（例如 iSCSI）與 AWS（EKS、ROSA、EC2 或任何其他執行個體）一起使用，則節點上所需的多路徑配置可能會與 Amazon Elastic Block Store (EBS) CSI 驅動程式衝突。為了確保多路徑功能不會幹擾同一節點上的 EBS 磁碟，您需要在多路徑設定中排除 EBS。這個例子展示了 `multipath.conf`包含所需Trident設定但排除 EBS 磁碟進行多路徑設定的檔案：

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

### 驗證

Trident提供兩種身分驗證方式。

- 基於憑證（建議）：將憑證安全地儲存在 AWS Secrets Manager 中。您可以使用 `fsxadmin`檔案系統的使用者或 `vsadmin`使用者已配置到您的 SVM。

> Trident預計將以...的形式運營 `vsadmin`SVM 用戶，或使用具有相同角色但名稱不同的用戶。 Amazon FSx for NetApp ONTAP具有 `fsxadmin`該用戶是ONTAP的有限替代品 `admin`集群用戶。我們強烈建議使用 `vsadmin`用Trident。

- 基於憑證： Trident將使用安裝在 SVM 上的憑證與 FSx 檔案系統上的 SVM 進行通訊。

有關啟用身份驗證的詳細信息，請參閱您的驅動程式類型的身份驗證說明：

- "ONTAP NAS 認證"
- "ONTAP SAN 驗證"

#### 已測試的亞馬遜機器映像 **(AMI)**

EKS 叢集支援各種作業系統，但 AWS 針對容器和 EKS 優化了某些 Amazon Machine Image (AMI)。以下 AMI 已使用NetApp Trident 25.02 進行測試。

| 急性心肌梗塞 | 網路儲存 | NAS經濟 | iSCSI | iSCSI經濟型 |
|---|---|---|---|---|
| AL2023_x86_64_STANDARD | 是的 | 是的 | 是的 | 是的 |
| AL2_x86_64 | 是的 | 是的 | 是的* | 是的* |
| BOTTLEROCKET_x86_64 | 是的** | 是的 | 不適用 | 不適用 |
| AL2023_ARM_64_STANDARD | 是的 | 是的 | 是的 | 是的 |
| AL2_ARM_64 | 是的 | 是的 | 是的* | 是的* |
| BOTTLEROCKET_ARM_64 | 是的** | 是的 | 不適用 | 不適用 |

- * 如果不重新啟動節點，則無法刪除 PV
- ** 不適用於Trident版本 25.02 的 NFSv3。

> 如果您所需的 AMI 未在此處列出，並不意味著它不受支援；這僅僅意味著它尚未經過測試。此清單可作為 AMI 已知可用系統的指南。

使用以下工具進行測試：

- EKS版本：1.32
- 安裝方法：Helm 25.06 和 AWS 附加元件 25.06
- 對於 NAS，NFSv3 和 NFSv4.1 都進行了測試。

- SAN 僅測試了 iSCSI，未測試 NVMe-oF。

已執行測試：

- 建立：儲存類別、PVC、Pod
- 刪除：pod、pvc（常規、qtree/lun – 經濟型、有 AWS 備份的 NAS）

查找更多信息

- "Amazon FSx for NetApp ONTAP文檔"
- "關於Amazon FSx for NetApp ONTAP的部落格文章"

## 建立 **IAM** 角色和 **AWS Secret**

您可以設定 Kubernetes pod 以透過 AWS IAM 角色進行驗證來存取 AWS 資源，而不是提供明確的 AWS 憑證。

若要使用 AWS IAM 角色進行驗證，您必須擁有一個使用 EKS 部署的 Kubernetes 叢集。

### 建立 **AWS Secrets Manager** 金鑰

由於Trident將向 FSx 虛擬伺服器發出 API 來為您管理存儲，因此它需要憑證才能執行此操作。傳遞這些憑證的安全方法是透過 AWS Secrets Manager 金鑰。因此，如果您還沒有 AWS Secrets Manager 金鑰，則需要建立一個包含 vsadmin 帳戶憑證的金鑰。

此範例建立一個 AWS Secrets Manager 金鑰來儲存Trident CSI 憑證：

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

### 建立 **IAM** 策略

Trident也需要 AWS 權限才能正常運作。因此，您需要建立一個策略，賦予Trident所需的權限。

以下範例使用 AWS CLI 建立 IAM 原則：

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

策略 **JSON** 範例：

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
  "Version": "2012-10-17"
}
```

建立 **Pod** 身分或 **IAM** 角色以關聯服務帳戶 **(IRSA)**

您可以設定 Kubernetes 服務帳戶，使其承擔 AWS Identity and Access Management (IAM) 角色（使用 EKS Pod Identity 或 IAM 角色進行服務帳戶關聯）(IRSA)。任何已配置為使用該服務帳戶的 Pod 都可以存取該角色有權存取的任何 AWS 服務。

**Pod 身份**

Amazon EKS Pod 身分關聯可讓您管理應用程式的憑證，類似於 Amazon EC2 執行個體設定檔向 Amazon EC2 執行個體提供憑證的方式。

在 **EKS** 叢集上安裝 **Pod Identity**：

您可以透過 AWS 控制台建立 Pod 標識，也可以使用下列 AWS CLI 命令：

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

更多資訊請參閱"設定 Amazon EKS Pod 身分代理"。

建立 **trust-relationship.json** 檔案：

建立 trust-relationship.json 文件，使 EKS 服務主體能夠承擔 Pod 身分的此角色。然後使用以下信任策略建立角色：

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

**trust-relationship.json** 文件：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

將角色策略附加到 **IAM** 角色：

將上一個步驟中建立的角色策略附加到已建立的 IAM 角色：

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

建立 **Pod** 身分關聯：

在 IAM 角色和Trident服務帳戶 (trident-controller) 之間建立 Pod 身分關聯

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

服務帳戶關聯 **(IRSA)** 的 **IAM** 角色

使用 **AWS CLI**：

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

**trust-relationship.json** 文件：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

更新以下值 `trust-relationship.json`文件：

- **<account_id>** - 您的 AWS 帳戶 ID
- **<oidc_provider>** - 您的 EKS 叢集的 OIDC。您可以透過執行以下命令來取得oidc_provider：

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

將 **IAM** 角色與 **IAM** 原則關聯：

角色創建完成後，使用以下命令將策略（在上一個步驟中建立的策略）附加到該角色：

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

請核實 **OICD** 提供者是否已關聯：

請確認您的 OIDC 提供者已與您的叢集關聯。您可以使用以下命令進行驗證：

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

如果輸出為空，請使用以下命令將 IAM OIDC 關聯到您的叢集：

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

如果您使用的是 **eksctl**，請使用以下範例在 EKS 中為服務帳戶建立 IAM 角色：

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

#### 安裝Trident

## Trident簡化了 Kubernetes 中Amazon FSx for NetApp ONTAP儲存管理，讓您的開發人員和管理員能夠專注於應用程式部署。

您可以使用下列方法之一安裝Trident ：

- 舵
- EKS 附加元件

如果要使用快照功能，請安裝 CSI 快照控制器外掛程式。請參閱"為CSI磁碟區啟用快照功能"了解更多。

透過 **Helm** 安裝**Trident**。

**Pod 身份**

1. 新增Trident Helm 倉庫：

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

2. 請依照以下範例安裝Trident ：

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace
```

您可以使用 `helm list`查看安裝詳細資訊的命令，例如名稱、命名空間、圖表、狀態、應用程式版本和修訂號。

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

服務帳戶協會（**IRSA**）

1. 新增Trident Helm 倉庫：

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

2. 設定*雲端提供者*和*雲端身分*的值：

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

您可以使用 `helm list`查看安裝詳細資訊的命令，例如名稱、命名空間、圖表、狀態、應用程式版本和修訂號。

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2506.0    25.06.0
```

如果您打算使用 iSCSI，請確保您的用戶端電腦上已啟用 iSCSI。如果您使用的是 AL2023 工作節點作業系統，可以透過在 Helm 安裝中新增節點準備參數來自動安裝 iSCSI 用戶端：

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace --
set nodePrep={iscsi}
```

#### 透過 **EKS** 外掛程式安裝**Trident**

Trident EKS 外掛程式包含最新的安全性修補程式和錯誤修復，並經過 AWS 驗證，可與 Amazon EKS 搭配使用。 EKS 外掛程式可讓您持續確保您的 Amazon EKS 叢集安全穩定，並減少安裝、設定和更新外掛程式所需的工作量。

先決條件

在為 AWS EKS 設定Trident外掛程式之前，請確保您已具備以下條件：

- 附加訂閱的 Amazon EKS 叢集帳戶
- AWS 對 AWS Marketplace 的權限：
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 類型：Amazon Linux 2 (AL2_x86_64) 或 Amazon Linux 2 Arm (AL2_ARM_64)
- 節點類型：AMD 或ARM
- 現有的Amazon FSx for NetApp ONTAP檔案系統

為 **AWS** 啟用**Trident**插件

管理控制台

1. 開啟 Amazon EKS 主控台 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左側導覽窗格中，選擇「集群」。
3. 選擇要為其配置NetApp Trident CSI 外掛程式的叢集名稱。
4. 選擇“附加元件”，然後選擇“取得更多附加元件”。
5. 請依照以下步驟選擇外掛：
   a. 向下捲動至「AWS Marketplace 附加元件」部分，然後在搜尋框中輸入「Trident」。
   b. 選取「Trident by NetApp」方塊右上角的複選框。
   c. 選擇“下一步”。
6. 在「配置所選外掛」設定頁面上，執行以下操作：

   *如果您使用 Pod Identity 關聯，請跳過這些步驟。 *

   a. 請選擇您要使用的*版本*。
   b. 如果您使用IRSA身份驗證，請確保設定可選配置設定中提供的配置值：
      - 請選擇您要使用的*版本*。
      - 依照*外掛程式設定方案*進行操作，並將*配置值*部分中的*configurationValues*參數設定為您在上一個步驟建立的角色 ARN（值應採用下列格式）：

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

+
如果選擇「覆蓋」作為衝突解決方法，則現有外掛程式的一個或多個設定可能會被 Amazon EKS 外掛程式設定覆蓋。如果您不啟用此選項，並且與您現有的設定有衝突，則操作將會失敗。您可以使用產生的錯誤訊息來排查衝突問題。在選擇此選項之前，請確保 Amazon EKS 外掛程式不會管理您需要自行管理的設定。

7. 選擇“下一步”。
8. 在「審核和新增」頁面上，選擇「建立」。

   插件安裝完成後，您將看到已安裝的插件。

**AWS CLI**

**1.**創建 **`add-on.json`**文件：

對於 **Pod** 標識，請使用以下格式：

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

對於**IRSA**認證，請使用以下格式：

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

代替 `<role ARN>`使用上一步創建的角色的 ARN。

*2.安裝Trident EKS 外掛程式。 *

```
aws eks create-addon --cli-input-json file://add-on.json
```

**eksctl**

以下範例指令安裝Trident EKS 外掛：

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

更新**Trident EKS** 外掛

管理控制台

1. 開啟 Amazon EKS 主控台 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左側導覽窗格中，選擇「集群」。
3. 選擇要更新NetApp Trident CSI 外掛程式的叢集名稱。
4. 選擇“附加元件”標籤。
5. 選擇“ NetApp Trident ”，然後選擇“編輯”。
6. 在「配置NetApp Trident 」頁面上，執行以下操作：
   a. 請選擇您要使用的*版本*。
   b. 展開“可選配置設定”，並根據需要進行修改。
   c. 選擇“儲存變更”。

**AWS CLI**

以下範例更新 EKS 外掛：

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

**eksctl**

- 請檢查您的 FSxN Trident CSI 外掛程式的目前版本。代替 `my-cluster`使用您的叢集名稱。

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

  範例輸出：

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- 將插件更新到上一步輸出中「UPDATE AVAILABLE」下傳回的版本。

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

如果你移除 `--force`如果選項和任何 Amazon EKS 外掛程式設定與您現有的設定衝突，則更新 Amazon EKS 外掛程式將失敗；您將收到錯誤訊息，以協助您解決衝突。在指定此選項之前，請確保 Amazon EKS 外掛程式不會管理您需要管理的設置，因為此選項會覆寫這些設定。有關此設定的其他選項的更多信息，請參閱"外掛"。有關 Amazon EKS Kubernetes 欄位管理的更多信息，請參閱"Kubernetes 欄位管理"。

解除安裝/移除**Trident EKS** 插件

您可以透過兩種方式移除 Amazon EKS 外掛：

- 保留叢集上的附加軟體 – 此選項移除 Amazon EKS 對所有設定的管理。它還取消了 Amazon EKS 通知您更新以及在您啟動更新後自動更新 Amazon EKS 外掛程式的功能。但是，它可以保留叢集上的附加軟體。此選項使外掛程式成為自管理安裝，而不是 Amazon EKS 外掛程式。選擇此選項，插件不會出現停機時間。保留 `--preserve`命令中的選項用於保留插件。
- 從叢集中完全移除附加軟體 – NetApp建議，僅當叢集中沒有任何資源依賴 Amazon EKS 附加元件時，才會從叢集中移除該附加元件。移除 `--preserve`選項 `delete`移除插件的命令。

如果外掛程式關聯了 IAM 帳戶，則不會刪除該 IAM 帳戶。

管理控制台

1. 開啟 Amazon EKS 主控台 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左側導覽窗格中，選擇「集群」。
3. 選擇要從中移除NetApp Trident CSI 外掛程式的叢集名稱。
4. 選擇“附加元件”選項卡，然後選擇“ NetApp Trident ”。
5. 選擇*刪除*。
6. 在「移除 netapp_trident-operator 確認」對話方塊中，執行下列操作：
   a. 如果您希望 Amazon EKS 停止管理外掛程式的設置，請選擇「在叢集上保留」。如果您希望在叢集上保留附加軟體，以便您可以自行管理附加軟體的所有設置，請執行此操作。
   b. 輸入 **netapp_trident-operator**。
   c. 選擇*刪除*。

**AWS CLI**

代替 `my-cluster`輸入叢集名稱，然後執行以下命令。

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

**eksctl**

以下指令卸載Trident EKS 外掛程式：

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

### ONTAP SAN 和 NAS 驅動程式集成

若要建立儲存後端，您需要建立 JSON 或 YAML 格式的設定檔。該文件需要指定您想要的儲存類型（NAS 或 SAN）、檔案系統、要從中取得儲存的 SVM 以及如何對其進行驗證。以下範例展示如何定義基於 NAS 的存儲，以及如何使用 AWS Secret 來儲存要使用的 SVM 的憑證：

**YAML**

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

**JSON**

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

執行下列命令以建立和驗證Trident後端設定 (TBC)：

- 從 yaml 檔案建立 Trident 後端設定 (TBC)，並執行下列命令：

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- 驗證 Trident 後端設定 (TBC) 是否已成功建立：

```
Kubectl get tbc -n trident
```

```
NAME                           BACKEND NAME          BACKEND UUID
PHASE    STATUS

backend-tbc-ontap-nas          tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9    Bound    Success
```

#### FSx 適用於ONTAP驅動程式的詳細信息

您可以使用下列驅動程式將Trident與Amazon FSx for NetApp ONTAP整合：

- `ontap-san`每個已設定的 PV 都是其自身Amazon FSx for NetApp ONTAP磁碟區中的一個 LUN。推薦用於塊存儲。
- `ontap-nas`每個已配置的 PV 都是一個完整的Amazon FSx for NetApp ONTAP磁碟區。推薦用於NFS和SMB。
- `ontap-san-economy`每個已設定的 PV 都是一個 LUN，每個Amazon FSx for NetApp ONTAP磁碟區可設定 LUN 的數量。
- `ontap-nas-economy`每個已配置的 PV 都是一個 qtree，每個Amazon FSx for NetApp ONTAP磁碟區可以配置 qtree 的數量。
- `ontap-nas-flexgroup`每個已配置的 PV 都是一個完整的Amazon FSx for NetApp ONTAP FlexGroup磁碟區。

有關司機詳情，請參閱"NAS驅動程式"和"SAN驅動程式"。

設定檔建立完成後，執行以下命令將其建立在 EKS 中：

```
kubectl create -f configuration_file
```

若要驗證狀態，請執行以下命令：

```
kubectl get tbc -n trident
```

```
NAME                      BACKEND NAME            BACKEND UUID
PHASE   STATUS
backend-fsx-ontap-nas   backend-fsx-ontap-nas   7a551921-997c-4c37-a1d1-
f2f4c87fa629   Bound   Success
```

後端高級配置和範例

請參閱下表以了解後端配置選項：

| 範圍 | 描述 | 例子 |
|---|---|---|
| `version` | | 始終為 1 |
| `storageDriverName` | 儲存驅動程式的名稱 | `ontap-nas`，`ontap-nas-economy`，`ontap-nas-flexgroup`，`ontap-san`，`ontap-san-economy` |
| `backendName` | 自訂名稱或儲存後端 | 驅動程式名稱 + "_" + dataLIF |
| `managementLIF` | 叢集或 SVM 管理 LIF 的 IP 位址可以指定完全限定網域名稱 (FQDN)。如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。IPv6 位址必須用方括號定義，例如 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。如果您提供 `fsxFilesystemID`在 `aws`在欄位中，您無需提供 `managementLIF`因為Trident可以檢索 SVM `managementLIF`來自 AWS 的資訊。因此，您必須提供 SVM 下使用者的憑證（例如：vsadmin），且該使用者必須擁有下列權限：`vsadmin`角色。 | "10.0.0.1", "[2001:1234:abcd::fefe]" |

| 範圍 | 描述 | 例子 |
| --- | --- | --- |
| dataLIF | LIF協定的IP位址。 * ONTAP NAS 驅動程式*： NetApp建議指定 dataLIF。如果未提供， Trident將從 SVM 取得 dataLIF。您可以指定一個完全限定網域名稱 (FQDN) 用於 NFS 掛載操作，從而建立輪詢 DNS 以在多個 dataLIF 之間進行負載平衡。初始設定後可以更改。參考。 * ONTAP SAN 驅動程式*：不要為 iSCSI 指定。 Trident使用ONTAP選擇性 LUN 對應來發現建立多路徑會話所需的 iSCI LIF。如果明確定義了 dataLIF，則會產生警告。如果Trident安裝時使用了 IPv6 標誌，則可以設定為使用 IPv6 位址。 IPv6 位址必須用方括號定義，例如 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。 | |
| autoExportPolicy | 啟用自動匯出策略建立和更新 [布林值]。使用 `autoExportPolicy`和 `autoExportCIDRs`選用功能包括：Trident可以自動管理出口策略。 | false |
| autoExportCIDRs | 用於過濾 Kubernetes 節點 IP 的 CIDR 列表 `autoExportPolicy`已啟用。使用 `autoExportPolicy`和 `autoExportCIDRs`選用功能包括：Trident可以自動管理出口策略。 | "["0.0.0.0/0", "::/0"]" |
| labels | 若要套用於磁碟區的任意 JSON 格式標籤集 | “” |
| clientCertificate | 用戶端憑證的 Base64 編碼值。用於基於憑證的身份驗證 | “” |
| clientPrivateKey | 客戶端私鑰的 Base64 編碼值。用於基於憑證的身份驗證 | “” |
| trustedCACertificate | 受信任 CA 憑證的 Base64 編碼值。選修的。用於基於憑證的身份驗證。 | “” |
| username | 連接到叢集或 SVM 的使用者名稱。用於基於憑證的身份驗證。例如，vsadmin。 | |
| password | 連接叢集或SVM的密碼。用於基於憑證的身份驗證。 | |
| svm | 使用的儲存虛擬機 | 如果指定了 SVM 管理 LIF，則派生出該 LIF。 |
| storagePrefix | 在 SVM 中配置新磁碟區時所使用的前綴。創建後無法修改。要更新此參數，您需要建立一個新的後端。 | trident |

| 範圍 | 描述 | 例子 |
| --- | --- | --- |
| `limitAggregateUsage` | *請勿指定用於Amazon FSx for NetApp ONTAP。 *提供的 `fsxadmin`和 `vsadmin`不包含檢索匯總使用情況和使用Trident限制它所需的權限。 | 請勿使用。 |
| `limitVolumeSize` | 如果請求的磁碟區大小大於此值，則配置失敗。此外，它還限制了其管理的 qtree 和 LUN 卷的最大大小，以及 `qtreesPerFlexvol`此選項允許自訂每個FlexVol volume的最大 qtree 數量。 | （預設不強制執行） |
| `lunsPerFlexvol` | 每個 Flexvol 卷的最大 LUN 數必須在 [50, 200] 範圍內。僅限SAN。 | “100” |
| `debugTraceFlags` | 故障排除時要使用的調試標誌。例如，{"api":false, "method":true} 請勿使用 `debugTraceFlags`除非您正在進行故障排除並且需要詳細的日誌轉儲。 | 無效的 |
| `nfsMountOptions` | 以逗號分隔的 NFS 掛載選項清單。Kubernetes 持久性磁碟區的掛載選項通常在儲存類別中指定，但如果儲存類別中沒有指定掛載選項，Trident將回退到使用儲存後端設定檔中指定的掛載選項。如果在儲存類別或設定檔中未指定任何掛載選項， Trident將不會在關聯的持久性磁碟區上設定任何掛載選項。 | “” |
| `nasType` | 配置 NFS 或 SMB 磁碟區的建立。選項有 `nfs`， `smb`或空值。 *必須設定為 `smb`適用於 SMB 卷。 *設定為 null 則預設使用 NFS 磁碟區。 | `nfs` |
| `qtreesPerFlexvol` | 每個FlexVol volume的最大 Qtree 數量必須在 [50, 300] 範圍內 | `"200"` |
| `smbShare` | 您可以指定以下名稱之一：使用 Microsoft 管理主控台或ONTAP CLI 所建立的 SMB 共用的名稱，或允許Trident建立 SMB 共用的名稱。此參數是Amazon FSx for ONTAP後端所必需的。 | `smb-share` |
| `useREST` | 使用ONTAP REST API 的布林參數。設定為 `true`Trident將使用ONTAP REST API 與後端進行通訊。此功能需要ONTAP 9.11.1 及更高版本。此外，所使用的ONTAP登入角色必須具有存取權限。 `ontap`應用。預定義項滿足了這一點。 `vsadmin`和 `cluster-admin`角色。 | `false` |

| 範圍 | 描述 | 例子 |
| --- | --- | --- |
| aws | 您可以在 AWS FSx for ONTAP的設定檔中指定以下內容： - fsxFilesystemID ：指定 AWS FSx 檔案系統的 ID。 - apiRegion AWS API 區域名稱。 - apikey AWS API 金鑰。 - secretKey AWS 金鑰。 | ""<br>""<br>"" |
| credentials | 指定要儲存在 AWS Secrets Manager 中的 FSx SVM 憑證。 - name ：包含 SVM 憑證的金鑰的 Amazon 資源名稱 (ARN)。 - type` 設定為 `awsarn。請參閱"建立 AWS Secrets Manager 金鑰"了解更多。 | |

用於配置磁碟區的後端配置選項

您可以使用下列選項控制預設配置。 `defaults`配置部分。例如，請參閱下面的設定範例。

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| spaceAllocation | LUN 的空間分配 | true |
| spaceReserve | 空間預留模式；「無」（細）或「大量」（粗） | none |
| snapshotPolicy | 要使用的快照策略 | none |
| qosPolicy | 若要為建立的磁碟區指派的 QoS 策略群組。每個儲存池或後端選擇 qosPolicy 或 adaptiveQosPolicy 之一。將 QoS 策略群組與Trident結合使用需要ONTAP 9.8 或更高版本。您應該使用非共享的 QoS 策略群組，並確保該策略群組單獨套用至每個成員。共享的 QoS 策略群組強制規定所有工作負載的總吞吐量上限。 | “” |
| adaptiveQosPolicy | 若要為建立的磁碟區指派的自適應 QoS 策略群組。每個儲存池或後端選擇 qosPolicy 或 adaptiveQosPolicy 之一。 ontap-nas-economy 不支援此功能。 | “” |
| snapshotReserve | 為快照“0”預留的磁碟區百分比 | 如果 snapshotPolicy`是 `none ，else “” |
| splitOnClone | 創建時將克隆體從其母體中分離出來 | false |

| 範圍 | 描述 | 預設 |
| --- | --- | --- |
| `encryption` | 在新磁碟區啟用NetApp磁碟區加密(NVE)；預設為 `false`。若要使用此選項，必須在叢集上取得 NVE 許可並啟用 NVE。如果後端啟用了NAE，則在Trident中配置的任何磁碟區都會啟用 NAE。更多信息，請參閱："Trident如何與 NVE 和 NAE 協同工作" 。 | `false` |
| `luksEncryption` | 啟用LUKS加密。參考"使用 Linux 統一金鑰設定 (LUKS)"。僅限 SAN。 | “” |
| `tieringPolicy` | 分層策略的使用 `none` | |
| `unixPermissions` | 新卷模式。 *SMB卷請留空。 * | “” |
| `securityStyle` | 新磁碟區的安全樣式。 NFS 支持 `mixed`和 `unix`安全措施。中小企業支持 `mixed`和 `ntfs`安全措施。 | NFS 預設值為 `unix`。 SMB 預設值為 `ntfs`。 |

#### 準備配置SMB卷

您可以使用以下方式設定 SMB 磁碟區： `ontap-nas`司機。在你完成之前ONTAP SAN 和 NAS 驅動程式集成請完成以下步驟。

##### 開始之前

在使用以下方式設定 SMB 磁碟區之前： `ontap-nas`駕駛員，您必須具備以下條件。

- 一個 Kubernetes 叢集，包含一個 Linux 控制器節點和至少一個執行 Windows Server 2019 的 Windows 工作節點。 Trident僅支援掛載到執行在 Windows 節點上的 pod 的 SMB 磁碟區。
- 至少有一個包含您的 Active Directory 憑證的Trident金鑰。生成秘密 `smbcreds`：

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- 配置為 Windows 服務的 CSI 代理程式。要配置 `csi-proxy`，請參閱"GitHub：CSI代理"或者"GitHub：適用於 Windows 的 CSI 代理"適用於在 Windows 上執行的 Kubernetes 節點。

##### 步驟

1. 建立SMB共享。您可以透過以下兩種方式之一建立 SMB 管理共用："Microsoft 管理控制台"共用資料夾管理單元或使用ONTAP CLI。使用ONTAP CLI 建立 SMB 共享：
   a. 如有必要，請建立共用的目錄路徑結構。

      這 `vserver cifs share create`此指令檢查在建立共用時 -path 選項中指定的路徑。如果指定的路徑不存在，則命令執行失敗。

   b. 建立與指定 SVM 關聯的 SMB 共用：

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

c. 確認共享已建立：

```
vserver cifs share show -share-name share_name
```

請參閱"建立 SMB 共享"了解詳細資訊。

2. 建立後端時，必須配置以下內容以指定 SMB 磁碟區。有關所有 FSx for ONTAP後端設定選項，請參閱"FSx for ONTAP設定選項和範例"。

| 範圍 | 描述 | 例子 |
|---|---|---|
| `smbShare` | 您可以指定以下名稱之一：使用 Microsoft 管理主控台或ONTAP CLI 所建立的 SMB 共用的名稱，或允許Trident建立 SMB 共用的名稱。此參數是Amazon FSx for ONTAP後端所必需的。 | `smb-share` |
| `nasType` | *必須設定為 `smb`.*如果為空，則預設為空。 `nfs` 。 | `smb` |
| `securityStyle` | 新磁碟區的安全樣式。 *必須設定為 `ntfs`或者 `mixed`適用於 SMB 卷。 * | `ntfs`或者 `mixed`適用於 SMB 卷 |
| `unixPermissions` | 新卷模式。 *對於 SMB 卷，此項必須留空。 * | “” |

配置儲存等級和**PVC**

# 配置 Kubernetes StorageClass 物件並建立儲存類，以指示Trident如何設定磁碟區。建立一個使用已設定的 Kubernetes StorageClass 的 PersistentVolumeClaim (PVC) 來要求存取 PV。然後您可以將光伏組件安裝到支架上。

建立儲存類別

## 設定 **Kubernetes StorageClass** 對象

這 "Kubernetes StorageClass 對象"該物件將Trident標識為該類別使用的配置器，並指示Trident如何配置磁碟區。使用此範例為使用 NFS 的磁碟區設定 Storageclass（有關屬性的完整列表，請參閱下面的Trident屬性部分）：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

使用此範例為使用 iSCSI 的磁碟區設定 Storageclass：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

若要在 AWS Bottlerocket 上配置 NFSv3 卷，請新增所需內容。 `mountOptions`到儲存類別：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
  - nfsvers=3
  - nolock
```

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

建立儲存類別

步驟

1. 這是一個 Kubernetes 對象，所以請使用 `kubectl`在 Kubernetes 中創建它。

```
kubectl create -f storage-class-ontapnas.yaml
```

2. 現在您應該在 Kubernetes 和Trident中都看到 **basic-csi** 儲存類，並且Trident應該已經發現了後端上的儲存池。

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

製作**PVC**管

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

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

製作好 PVC 後，就可以將容積安裝到艙體中。

樣品清單

**PersistentVolumeClaim** 樣本清單

這些範例展示了PVC的基本配置選項。

附**RWX**接口的**PVC**

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

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

使用 **iSCSI** 範例的 **PVC**

此範例展示了一個與名為 StorageClass 的儲存類別關聯的、具有 RWO 存取權限的 iSCSI 基本 PVC。 `protection-gold` 。

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

## 建立**PVC**

步驟

1. 建立 PVC。

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

2. 核實PVC狀態。

```
kubectl get pvc
```

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

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

#### Trident屬性

這些參數決定了應使用哪些 Trident 管理的儲存池來配置給定類型的磁碟區。

| 屬性 | 類型 | 價值觀 | 提供 | 要求 | 由...支持 |
|---|---|---|---|---|---|
| 媒體[1] | 細繩 | 機械式硬碟、混合式硬碟、固態硬碟 | Pool 包含此類媒體；混合型媒體是指兩者兼具。 | 指定的媒體類型 | ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、solidfire-san |
| 供應類型 | 細繩 | 薄的，厚的 | 池支援這種配置方法 | 指定的配置方法 | 厚：全部 ontap；薄：全部 ontap 和 solidfire-san |
| 後端類型 | 細繩 | ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、solidfire-san、gcp-cvs、azure-netapp-files、ontap-san-economy | 池屬於這種類型的後端 | 指定的後端 | 所有司機 |
| 快照 | 布林值 | 真，假 | 儲存池支援帶快照的磁碟區 | 已啟用快照的磁碟區 | ontap-nas、ontap-san、solidfire-san、gcp-cvs |
| 複製 | 布林值 | 真，假 | 儲存池支援磁碟區克隆 | 已啟用克隆的磁碟區 | ontap-nas、ontap-san、solidfire-san、gcp-cvs |

| 屬性 | 類型 | 價值觀 | 提供 | 要求 | 由...支持 |
| --- | --- | --- | --- | --- | --- |
| 加密 | 布林值 | 真，假 | 儲存池支援加密磁碟區 | 已啟用加密的磁碟區 | ontap-nas、ontap-nas-economy、ontap-nas-flexgroups、ontap-san |
| 每秒輸入/輸出次數 | 整數 | 正整數 | Pool 能夠保證在此範圍內的 IOPS | 容量保證了這些IOPS | solidfire-san |

[1]： ONTAP Select系統不支援此系統

部署範例應用程式

## 建立儲存等級和 PVC 後，即可將光電模組安裝到艙體上。本節列出了將 PV 連接到 pod 的範例命令和設定。

步驟

1. 將音量旋鈕安裝在一個音控器中。

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

以下範例展示了將PVC連接到艙體的基本配置：基本配置：

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: 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
```

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

2. 確認卷已掛載到 `/my/mount/path`。

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

```
Filesystem                                                      Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

現在您可以刪除該 Pod 了。 Pod 應用將不復存在，但卷將保留。

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

在 **EKS** 叢集上設定**Trident EKS** 插件

NetApp Trident簡化了 Kubernetes 中Amazon FSx for NetApp ONTAP儲存管理，讓您的開發人員和管理員專注於應用程式部署。 NetApp Trident EKS 外掛程式包含最新的安全性修補程式和錯誤修復，並經過 AWS 驗證，可與 Amazon EKS 搭配使用。 EKS 外掛程式可讓您持續確保 Amazon EKS 叢集的安全性和穩定性，並減少安裝、設定和更新外掛程式所需的工作量。

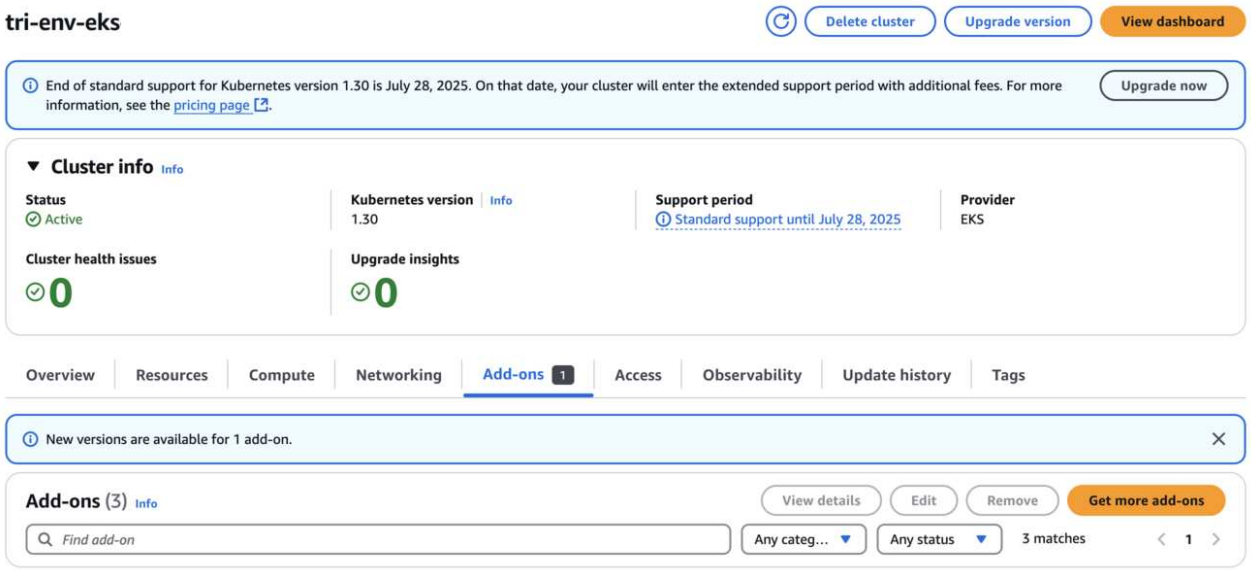
先決條件

在為 AWS EKS 設定Trident外掛程式之前，請確保您已具備以下條件：

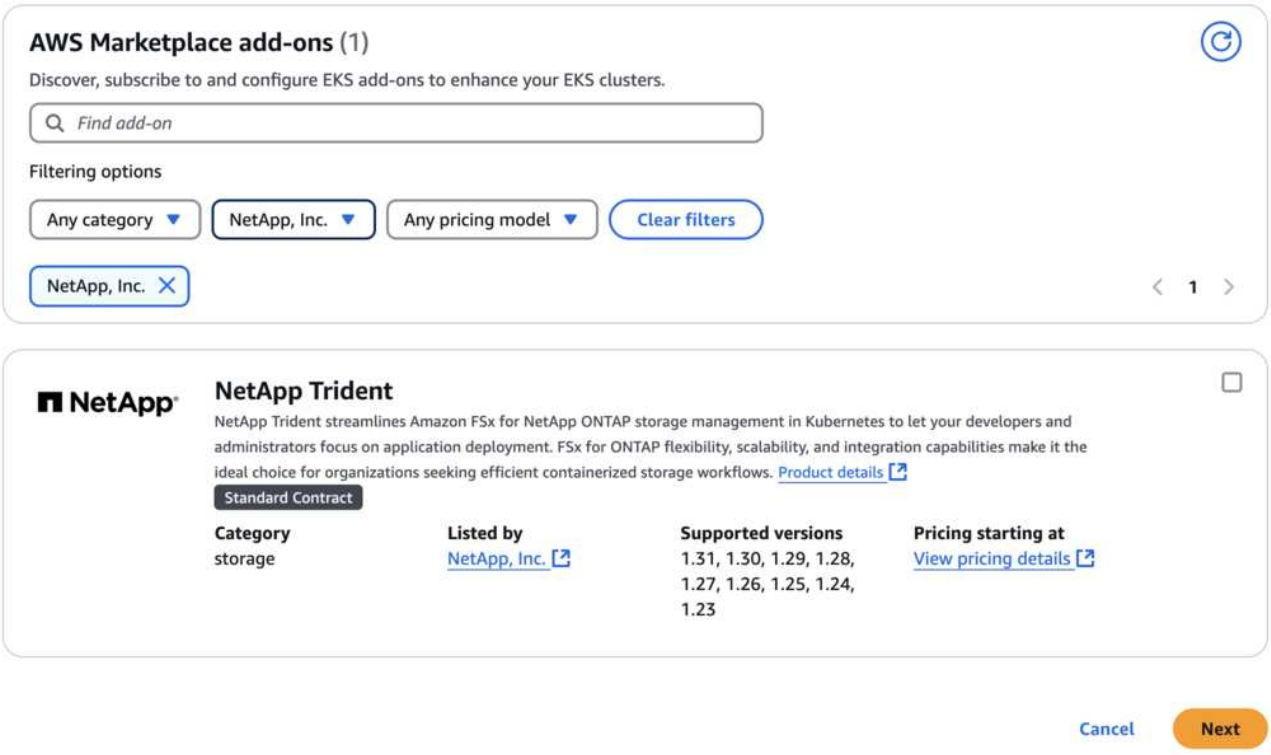
- 具有使用插件權限的 Amazon EKS 叢集帳戶。參考"Amazon EKS 附加元件"。
- AWS 對 AWS Marketplace 的權限：
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 類型：Amazon Linux 2 (AL2_x86_64) 或 Amazon Linux 2 Arm (AL2_ARM_64)
- 節點類型：AMD 或ARM
- 現有的Amazon FSx for NetApp ONTAP檔案系統

步驟

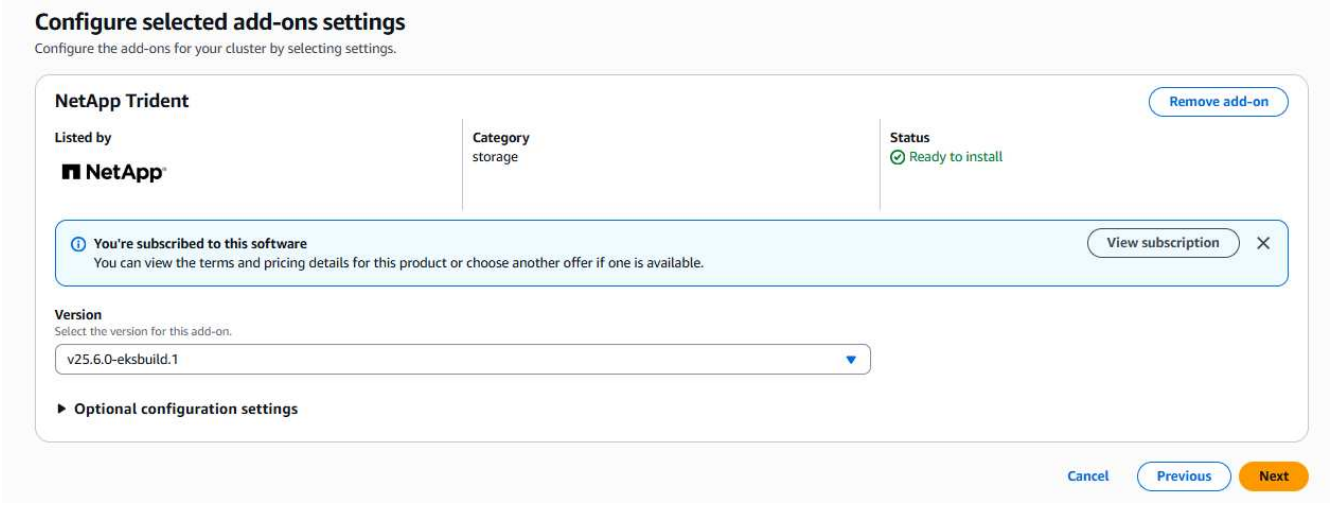
1. 請務必建立 IAM 角色和 AWS 金鑰，以使 EKS pod 能夠存取 AWS 資源。有關說明，請參閱"建立 IAM 角色和 AWS Secret"。
2. 在 EKS Kubernetes 叢集上，導覽至「附加元件」標籤。



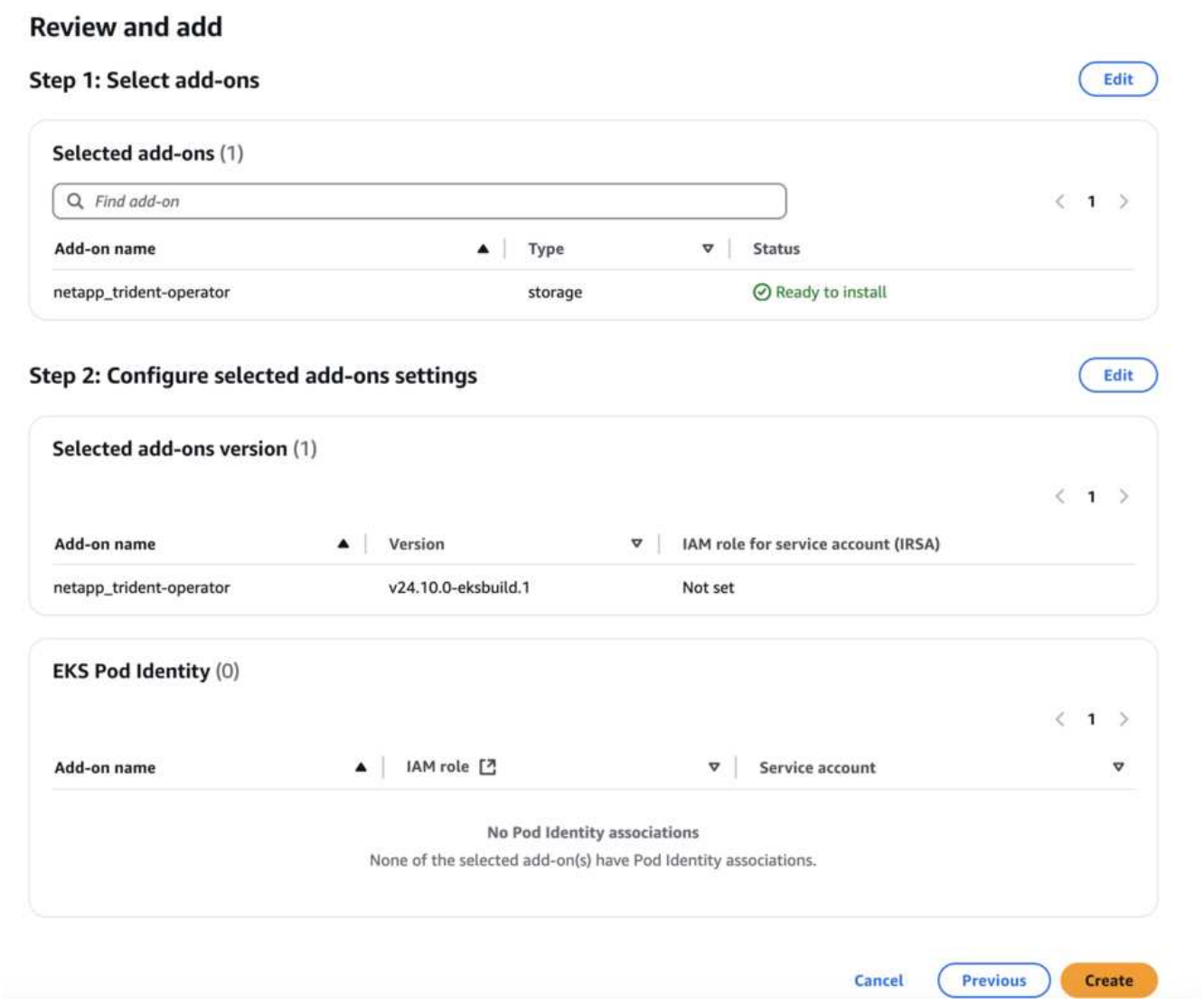

3. 前往 **AWS Marketplace** 外掛程式，然後選擇 *storage* 類別。




4. 找到 * NetApp Trident*，選取Trident程式的複選框，然後按一下 下一步。
5. 選擇所需的插件版本。

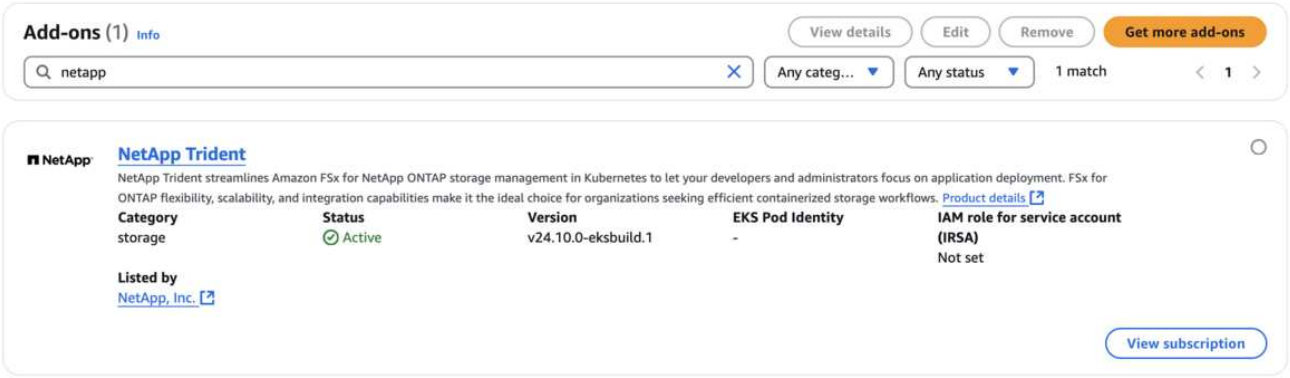



6. 配置所需的附加元件設定。




7. 如果您使用的是 IRSA（服務帳戶的 IAM 角色），請參閱其他設定步驟。"這裡" 。
8. 選擇“創建”。

9. 確認插件狀態為_Active_。




10. 執行以下命令以驗證Trident是否已正確安裝在叢集上：

```
kubectl get pods -n trident
```

11. 繼續進行設定並配置儲存後端。有關信息，請參閱"配置儲存後端"。

使用 **CLI** 安裝/解除安裝**Trident EKS** 插件

使用 **CLI** 安裝**NetApp Trident EKS** 外掛：

以下範例指令安裝Trident EKS 外掛：

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.6.0-eksbuild.1 （另有專用版本）
```

使用 **CLI** 卸載**NetApp Trident EKS** 外掛程式：

以下指令卸載Trident EKS 外掛程式：

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

## 使用 **kubectl** 建立後端

後端定義了Trident與儲存系統之間的關係。它告訴Trident如何與該儲存系統通信，以及Trident應該如何從中設定磁碟區。 Trident安裝完成後，下一步是建立後端。這 `TridentBackendConfig`自訂資源定義 (CRD) 可讓您透過 Kubernetes 介面直接建立和管理Trident後端。您可以使用 `kubectl`或適用於您的 Kubernetes 發行版的等效 CLI 工具。

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`， `tbconfig` ， `tbackendconfig` ) 是一個前端、命名空間 CRD，讓您能夠使用 來管理Trident後端 `kubectl`。現在，Kubernetes 和儲存管理員可以直接透過 Kubernetes CLI 建立和管理後端，而無需使用專門的命令列實用程式。(`tridentctl` ） 。

在創建之後 `TridentBackendConfig`對於該對象，會發生以下情況：

- Trident會根據您提供的設定自動建立後端。這在內部表示為 `TridentBackend`（`tbe`，`tridentbackend`）CR。
- 這 `TridentBackendConfig`與…有著獨特的聯繫 `TridentBackend`這是由Trident生產的。

每個 `TridentBackendConfig`與…保持一對一映射關係 `TridentBackend`前者是提供給使用者設計和配置後端的介面；後者是Trident表示實際後端物件的方式。

> `TridentBackend`CR 由Trident自動建立。你*不應該*修改它們。如果您想對後端進行更新，請透過修改以下檔案來實現： `TridentBackendConfig`目的。

以下範例展示了格式： `TridentBackendConfig` CR：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

您也可以查看以下範例： "trident-installer"包含所需儲存平台/服務的範例配置的目錄。

這 `spec`接受後端特定的配置參數。在這個例子中，後端使用了 `ontap-san`儲存驅動程序，並使用此處表格中列出的配置參數。有關所需儲存驅動程式的配置選項列表，請參閱"儲存驅動程式的後端配置訊息"。

這 `spec`本節還包括 `credentials`和 `deletionPolicy`這些字段是新引入的 `TridentBackendConfig`CR：

- `credentials`此參數為必填字段，包含用於向儲存系統/服務進行身份驗證的憑證。這設定為用戶創建的 Kubernetes Secret。憑證不能以明文形式傳遞，否則將導致錯誤。
- `deletionPolicy`此欄位定義了當…時應該發生的情況 `TridentBackendConfig`被刪除。它可以取以下兩個值之一：
  - `delete`這會導致兩者被刪除。 `TridentBackendConfig` CR及其相關後端。這是預設值。
  - `retain`當 `TridentBackendConfig`CR 刪除後，後端定義仍然存在，並且可以透過以下方式進行管理： `tridentctl` 。將刪除策略設為 `retain`允許使用者降級到早期版本（21.04 之前），並保留已建立的後端。該欄位的值可以在之後更新。 `TridentBackendConfig`已創建。

> 後端名稱是透過以下方式設定的： `spec.backendName` 。如果未指定，則後端名稱設定為…的名稱 `TridentBackendConfig`對象（`metadata.name`）。建議使用顯式方式設定後端名稱。 `spec.backendName` 。

用以下方式建立的後端 `tridentctl`沒有關聯的 `TridentBackendConfig`目的。您可以選擇使用以下方式管理此類後端 `kubectl`透過創建一個 `TridentBackendConfig`CR。必須注意指定完全相同的配置參數（例如： `spec.backendName ， spec.storagePrefix ， spec.storageDriverName ， 等等）。 Trident將自動綁定新建立的 `TridentBackendConfig`利用現有的後端。

步驟概述

使用以下方式建立新的後端 `kubectl`你應該這樣做：

1. 創建一個 "Kubernetes Secret"此金鑰包含Trident與儲存叢集/服務通訊所需的憑證。
2. 創建一個 `TridentBackendConfig`目的。這包含有關儲存叢集/服務的具體信息，並引用上一步中建立的密鑰。

建立後端後，您可以使用以下方式觀察其狀態 `kubectl get tbc <tbc-name> -n <trident-namespace>`並收集更多細節資訊。

### 步驟 1：建立 Kubernetes Secret

建立一個包含後端存取憑證的金鑰。這是每個儲存服務/平台獨有的。以下是一個例子：

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: password
```

下表總結了每個儲存平台金鑰中必須包含的欄位：

| 儲存平台密鑰字段描述 | 秘密 | 字段描述 |
| --- | --- | --- |
| Azure NetApp Files | 客戶端ID | 應用程式註冊中的客戶端 ID |
| 適用於 GCP 的 Cloud Volumes Service | 私鑰 ID | 私鑰的ID。具有 CVS 管理員角色的 GCP 服務帳戶的部分 API 金鑰 |
| 適用於 GCP 的 Cloud Volumes Service | 私鑰 | 私鑰。具有 CVS 管理員角色的 GCP 服務帳戶的部分 API 金鑰 |

| 儲存平台密鑰字段描述 | 秘密 | 字段描述 |
| --- | --- | --- |
| 元素（NetApp HCI/ SolidFire） | 端點 | 針對SolidFire叢集的 MVIP，包含租戶憑證 |
| ONTAP | 使用者名稱 | 用於連接到叢集/SVM 的使用者名稱。用於基於憑證的身份驗證 |
| ONTAP | 密碼 | 連接到叢集/SVM 的密碼。用於基於憑證的身份驗證 |
| ONTAP | 客戶端私鑰 | 客戶端私鑰的 Base64 編碼值。用於基於憑證的身份驗證 |
| ONTAP | chap用戶名 | 入站用戶名。如果 useCHAP=true，則為必填項。為了 `ontap-san`和 `ontap-san-economy` |
| ONTAP | chapInitiatorSecret | CHAP 發起者金鑰。如果 useCHAP=true，則此項目為必填項。為了 `ontap-san`和 `ontap-san-economy` |
| ONTAP | chapTargetUsername | 目標用戶名。如果 useCHAP=true，則此項目為必填項。為了 `ontap-san`和 `ontap-san-economy` |
| ONTAP | chapTargetInitiatorSecret | CHAP 目標發起者金鑰。如果 useCHAP=true，則此項目為必填項。為了 `ontap-san`和 `ontap-san-economy` |

此步驟中建立的秘密將在以下位置引用： `spec.credentials`領域的 `TridentBackendConfig`在下一步建立的物件。

**步驟 2：建立 `TridentBackendConfig`CR**

現在您可以開始創建您的 `TridentBackendConfig`CR。在這個例子中，後端使用了 `ontap-san`驅動程式是透過使用以下方式建立的： `TridentBackendConfig`下圖所示物體：

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

步驟 3：驗證狀態 `TridentBackendConfig`CR

現在你已經創建了 `TridentBackendConfig`CR，您可以核實狀態。請參閱以下範例：

```
kubectl -n trident get tbc backend-tbc-ontap-san
NAME                       BACKEND NAME           BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

後端已成功建立並綁定到 `TridentBackendConfig`CR。

相位可以取以下值之一：

- `Bound`： 這 `TridentBackendConfig`CR 與後端關聯，此後端包含 `configRef`設定為 `TridentBackendConfig`CR 的 uid。
- `Unbound`：用以下方式表示 ""。這 `TridentBackendConfig`該物件未綁定到後端。所有新創建 `TridentBackendConfig`CR 預設處於此階段。階段變更後，它無法再恢復到未綁定狀態。
- `Deleting`： 這 `TridentBackendConfig`CR的 `deletionPolicy`已設定為刪除。當 `TridentBackendConfig`CR 刪除後，狀態變成正在刪除。
  - 如果後端不存在持久性磁碟區宣告 (PVC)，則刪除 `TridentBackendConfig`這將導致Trident刪除後端以及 `TridentBackendConfig`CR。
  - 如果後端存在一個或多個 PVC，則進入刪除狀態。這 `TridentBackendConfig`CR隨後也進入刪除階段。後端和 `TridentBackendConfig`只有在所有 PVC 都被刪除後才會刪除。
- `Lost`與後端相關的 `TridentBackendConfig`CR 被意外或故意刪除，並且 `TridentBackendConfig`CR 仍然保留著對已刪除後端的引用。這 `TridentBackendConfig`無論如何，CR 仍然可以刪除。 `deletionPolicy`價值。
- `Unknown`Trident無法確定與下列系統相關的後端的狀態或是否存在： `TridentBackendConfig` CR。例如，如果 API 伺服器沒有回應，或者如果 `tridentbackends.trident.netapp.io`CRD 缺失。這可能需要幹預。

至此，後端已成功創建！此外，還可以處理以下幾種操作："後端更新和後端刪除" 。

（可選）步驟 **4**：了解更多詳情

您可以執行以下命令來獲取有關後端的更多資訊：

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                      BACKEND NAME         BACKEND UUID
PHASE    STATUS     STORAGE DRIVER    DELETION POLICY
backend-tbc-ontap-san    ontap-san-backend    8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8    Bound    Success    ontap-san         delete
```

此外，您還可以獲得 YAML/JSON 轉儲檔案。 `TridentBackendConfig` 。

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: 2021-04-21T20:45:11Z
  finalizers:
    - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

backendInfo`包含 `backendName`以及 `backendUUID`後端是根據以下情況創建的：`TridentBackendConfig CR。這 lastOperationStatus`此欄位表示上次操作的狀態`TridentBackendConfig`CR（變更要求）可以由使用者觸發（例如，使用者變更了某些內容）。 `spec）或由Trident觸發（例如，在Trident重啟期間）。結果要么是成功，要么是失敗。 phase`代表了以下關係的狀態： `TridentBackendConfig CR 和後端。在上面的例子中，`phase`具有 Bound 值，這意味著`TridentBackendConfig`CR與後端相關。

您可以運行 `kubectl -n trident describe tbc <tbc-cr-name>`取得事件日誌詳細資訊的命令。

> 您無法更新或刪除包含關聯後端的後端。 TridentBackendConfig`使用物件`tridentctl。要了解在以下兩者之間切換所涉及的步驟： tridentctl`和`TridentBackendConfig，"請參閱此處" 。

## 管理後端

使用 **kubectl** 執行後端管理

# 了解如何使用以下方式執行後端管理操作 kubectl。

刪除後端

透過刪除一個 TridentBackendConfig`您指示Trident刪除/保留後端（基於 `deletionPolicy）。若要刪除後端，請確保 deletionPolicy`已設定為刪除。僅刪除 `TridentBackendConfig`確保 `deletionPolicy`設定為保留。這樣可以確保後端仍然存在，並且可以透過以下方式進行管理：`tridentctl 。

運行以下命令：

```
kubectl delete tbc <tbc-name> -n trident
```

Trident不會刪除正在使用的 Kubernetes Secret。 TridentBackendConfig 。 Kubernetes 用戶負責清理金鑰。刪除機密資訊時務必謹慎。只有當後端不再使用密鑰時，才應該刪除密鑰。

查看現有後端

運行以下命令：

```
kubectl get tbc -n trident
```

你也可以運行 tridentctl get backend -n trident`或者 `tridentctl get backend -o yaml -n trident`取得所有現有後端的清單。此清單還將包括使用以下方式建立的後端： `tridentctl 。

更新後端

更新後端的原因可能有很多：

- 儲存系統的憑證已更改。要更新憑證，需要更新 Kubernetes Secret，該 Secret 用於：`TridentBackendConfig`對象必須更新。 Trident會自動使用提供的最新憑證更新後端。執行以下命令更新 Kubernetes Secret：

  ```
  kubectl apply -f <updated-secret-file.yaml> -n trident
  ```

- 需要更新參數（例如正在使用的ONTAP SVM 的名稱）。
  - 您可以更新 `TridentBackendConfig`使用以下命令直接透過 Kubernetes 存取物件：

    ```
    kubectl apply -f <updated-backend-file.yaml>
    ```

  - 或者，您可以對現有內容進行變更。 `TridentBackendConfig`使用以下命令進行回車：

```
kubectl edit tbc <tbc-name> -n trident
```

- 如果後端更新失敗，後端將繼續保持其最後一次已知的配置。您可以透過執行以下命令查看日誌以確定原因。 `kubectl get tbc <tbc-name> -o yaml -n trident`或者 `kubectl describe tbc <tbc-name> -n trident`。
- 在您發現並修正設定檔中的問題後，您可以重新執行更新命令。

使用 **tridentctl** 執行後端管理

## 了解如何使用以下方式執行後端管理操作 `tridentctl`。

創建後端

創建之後"後端設定檔"運行以下命令：

```
tridentctl create backend -f <backend-file> -n trident
```

如果後端建立失敗，則表示後端配置存在問題。您可以透過執行以下命令查看日誌以確定原因：

```
tridentctl logs -n trident
```

在您發現並修正設定檔中的問題後，您只需執行以下命令即可。 `create`再次發出命令。

刪除後端

若要從Trident中刪除後端，請執行下列操作：

1. 取得後端名稱：

```
tridentctl get backend -n trident
```

2. 刪除後端：

```
tridentctl delete backend <backend-name> -n trident
```

如果Trident已從從後端配置了仍然存在的磁碟區和快照，則刪除後端將阻止從該後端設定新磁碟區。後端將繼續處於「刪除」狀態。

查看現有後端

若要查看Trident已知的後端，請執行下列操作：

- 若要取得摘要，請執行以下命令：

```
tridentctl get backend -n trident
```

- 要獲取所有詳細信息，請運行以下命令：

```
tridentctl get backend -o json -n trident
```

#### 更新後端

建立新的後端設定檔後，執行以下命令：

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

如果後端更新失敗，則表示後端配置有問題，或者您嘗試了無效的更新。您可以透過執行以下命令查看日誌以確定原因：

```
tridentctl logs -n trident
```

在您發現並修正設定檔中的問題後，您只需執行以下命令即可。 `update`再次發出命令。

#### 確定使用後端儲存的儲存類

這是一個您可以使用 JSON 回答的問題範例： `tridentctl`後端物件的輸出。這使用 `jq`您需要安裝該實用程式。

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

這也適用於使用以下方式建立的後端： `TridentBackendConfig` 。

### 在後端管理選項之間切換

## 了解Trident中管理後端的不同方法。

#### 後端管理選項

隨著 `TridentBackendConfig`現在，管理員有兩種獨特的後端管理方式。這就引出了以下問題：

- 可以使用以下方式建立後端 `tridentctl`**以……進行管理** `TridentBackendConfig`？
- 可以使用以下方式建立後端 `TridentBackendConfig`**可透過以下方式進行管理** `tridentctl`？

## 管理 tridentctl`使用後端 `TridentBackendConfig

本節介紹管理使用以下方式建立的後端所需的步驟： `tridentctl`直接透過 Kubernetes 介面創建 `TridentBackendConfig`物體。

這適用於以下情況：

- 預先存在的後端，沒有 TridentBackendConfig`因為它們是用…創造的 `tridentctl。
- 使用以下方式建立的新後端 `tridentctl`而其他 `TridentBackendConfig`物體是存在的。

在這兩種情況下，後端都將繼續存在， Trident將調度磁碟區並對其進行操作。管理員此時有兩種選擇：

- 繼續使用 `tridentctl`管理使用它創建的後端。
- 使用以下方式建立的綁定後端 tridentctl`到一個新的 `TridentBackendConfig`目的。這樣做意味著後端將使用以下方式進行管理： `kubectl`而不是 `tridentctl。

使用以下方式管理預先存在的後端 `kubectl`您需要建立一個 `TridentBackendConfig`它與現有後端綁定。以下是其工作原理概述：

1. 創建 Kubernetes Secret。此金鑰包含Trident與儲存叢集/服務通訊所需的憑證。
2. 創建一個 TridentBackendConfig`目的。這包含有關儲存叢集/服務的具體信息，並引用上一步中建立的密鑰。必須注意指定完全相同的配置參數（例如： `spec.backendName ， spec.storagePrefix ， spec.storageDriverName ， 等等）。 `spec.backendName`必須設定為現有後端的名稱。

### 步驟 0：確定後端

創建一個 `TridentBackendConfig`如果要綁定到現有後端，則需要取得後端配置。在這個例子中，我們假設使用以下 JSON 定義建立了一個後端：

```
tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

```
cat ontap-nas-backend.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.10.10.1",
  "dataLIF": "10.10.10.2",
  "backendName": "ontap-nas-backend",
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "admin-password",
  "defaults": {
    "spaceReserve": "none",
    "encryption": "false"
  },
  "labels": {
    "store": "nas_store"
  },
  "region": "us_east_1",
  "storage": [
    {
      "labels": {
        "app": "msoffice",
        "cost": "100"
      },
      "zone": "us_east_1a",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "true",
        "unixPermissions": "0755"
      }
    },
    {
      "labels": {
        "app": "mysqldb",
        "cost": "25"
      },
      "zone": "us_east_1d",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "false",
        "unixPermissions": "0775"
      }
    }
  ]
}
```

步驟 1：建立 **Kubernetes Secret**

建立一個包含後端憑證的 Secret，如下例所示：

```
cat tbc-ontap-nas-backend-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password
```

```
kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

步驟 2：建立 **`TridentBackendConfig`CR**

下一步是創建一個 `TridentBackendConfig`CR` 將自動綁定到預先存在的 `ontap-nas-backend（如本例所示）。請確保滿足以下要求：

- 後端名稱在以下位置定義： `spec.backendName` 。
- 配置參數與原後端相同。
- 虛擬池（如果存在）必須保持與原始後端相同的順序。
- 憑證透過 Kubernetes Secret 提供，而不是以明文形式提供。

在這種情況下， `TridentBackendConfig`將會像這樣：

```
cat backend-tbc-ontap-nas.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'
```

```
kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

#### 步驟 3：驗證狀態 `TridentBackendConfig`CR

之後 `TridentBackendConfig`已經創建，它的階段必須是 `Bound`。它也應該反映與現有後端相同的後端名稱和 UUID。

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                    BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

後端現在將完全使用以下方式進行管理： `tbc-ontap-nas-backend` `TridentBackendConfig`目的。

管理 `TridentBackendConfig`使用後端 `tridentctl`

`tridentctl`可用於列出使用下列方式建立的後端： `TridentBackendConfig` 。此外，管理員還可以選擇透過以下方式完全管理此類後端： `tridentctl`透過刪除 `TridentBackendConfig`並確保 `spec.deletionPolicy`設定為 `retain`。

步驟 **0**：確定後端

例如，假設我們使用以下方式建立了以下後端 `TridentBackendConfig`：

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

從輸出結果可以看出：`TridentBackendConfig`已成功建立並綁定到後端[觀察後端的 UUID]。

**步驟 1：確認** `deletionPolicy`**設定為** `retain`

讓我們來看看它的價值 `deletionPolicy`。需要將其設定為 `retain`。這確保了當 `TridentBackendConfig`CR 刪除後，後端定義仍然存在，並且可以透過以下方式進行管理：`tridentctl`。

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

除非另有說明，否則請勿進行下一步。 `deletionPolicy`設定為 `retain`。

步驟二：刪除 `TridentBackendConfig`CR

最後一步是刪除 `TridentBackendConfig`CR。確認後 `deletionPolicy`設定為 `retain`您可以繼續刪除：

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|        NAME         | STORAGE DRIVER |                 UUID
| STATE   | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

刪除後 `TridentBackendConfig`Trident只是移除該對象，而不會實際刪除後端本身。

# 建立和管理儲存類

## 建立儲存類別

配置 Kubernetes StorageClass 物件並建立儲存類，以指示Trident如何設定磁碟區。

### 設定 **Kubernetes StorageClass** 對象

這 "Kubernetes StorageClass 對象"識別Trident作為該類別使用的配置器，並指示Trident如何配置磁碟區。例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

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

### 建立儲存類別

建立 StorageClass 物件後，即可建立儲存類別。[儲存類別範例]提供一些您可以使用或修改的基本範例。

#### 步驟

1. 這是一個 Kubernetes 對象，所以請使用 `kubectl`在 Kubernetes 中創建它。

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. 現在您應該在 Kubernetes 和Trident中都看到 **basic-csi** 儲存類，並且Trident應該已經發現了後端上的儲存池。

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

```
./tridentctl -n trident get storageclass basic-csi -o json
```

```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

儲存類別範例

Trident提供 "針對特定後端的簡單儲存類別定義"。

或者，您可以編輯 `sample-input/storage-class-csi.yaml.templ`安裝程式隨附的檔案並替換 `*BACKEND_TYPE*` 使用儲存驅動程式名稱。

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

## 管理儲存類別

您可以查看現有儲存類別、設定預設儲存類別、識別儲存類別後端以及刪除儲存類別。

### 查看現有儲存類

- 若要查看現有的 Kubernetes 儲存類，請執行下列命令：

```
kubectl get storageclass
```

- 要查看 Kubernetes 儲存類別詳細信息，請執行以下命令：

```
kubectl get storageclass <storage-class> -o json
```

- 若要查看 Trident 的同步儲存類，請執行下列命令：

```
tridentctl get storageclass
```

- 要查看 Trident 的同步儲存類別詳細信息，請執行以下命令：

```
tridentctl get storageclass <storage-class> -o json
```

設定預設儲存類

Kubernetes 1.6 增加了設定預設儲存類別的功能。如果使用者未在持久性磁碟區宣告 (PVC) 中指定持久卷，則將使用此儲存類別來設定持久性磁碟區。

- 透過設定註解來定義預設儲存類 `storageclass.kubernetes.io/is-default-class`在儲存類別定義中設定為 true。根據規範，任何其他值或缺少註釋均被解釋為錯誤。
- 您可以使用下列命令將現有儲存類別配置為預設儲存類別：

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- 同樣，您可以使用以下命令移除預設的儲存類別註解：

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Trident安裝程式包中也有一些包含此註解的範例。

叢集中一次只能存在一個預設儲存類別。從技術上講，Kubernetes 不會阻止你擁有多個預設儲存類，但它的行為就好像根本沒有預設儲存類別一樣。

確定儲存類別的後端

這是一個您可以使用 JSON 回答的問題範例： `tridentctl` Trident後端物件的輸出。這使用 `jq`您可能需要先安裝該實用程式。

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

刪除儲存類

若要從 Kubernetes 中刪除儲存類，請執行以下命令：

```
kubectl delete storageclass <storage-class>
```

`<storage-class>`應該替換成你的儲存類別。

透過此儲存類別建立的任何持久性磁碟區都將保持不變， Trident將繼續管理它們。

Trident強制執行空白 `fsType`因為它創造了大量的銷售。對於 iSCSI 後端，建議強制執行 `parameters.fsType`在儲存類別中。您應該刪除現有的 StorageClasses 並重新建立它們。 `parameters.fsType`指定的。

# 配置和管理卷

## 提供一定量

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

概況

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

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

製作好PVC管後，就可以將管體安裝到管座中。

### 製作**PVC**管

步驟

1. 建立 PVC。

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

2. 核實PVC狀態。

```
kubectl get pvc
```

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

1. 將音量旋鈕安裝在一個音控器中。

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

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

2. 確認卷已掛載到 `/my/mount/path`。

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

3. 現在您可以刪除該 Pod 了。 Pod 應用將不復存在，但卷將保留。

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

樣品清單

**PersistentVolumeClaim** 樣本清單

這些範例展示了PVC的基本配置選項。

**PVC管材，附RWO通道**

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

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

**PVC 和 NVMe/TCP**

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

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

**Pod清單範例**

這些範例展示了將 PVC 連接到艙體的基本配置。

基本配置

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

基本 **NVMe/TCP** 配置

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

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

# 擴大銷量

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

## 擴充 **iSCSI** 卷

您可以使用 CSI 設定程式擴充 iSCSI 持久性磁碟區 (PV)。

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

### 步驟 **1**：設定 **StorageClass** 以支援磁碟區擴展

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

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

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

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

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

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

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

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

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

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

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

步驟 3：定義一個連接 **PVC** 的艙體

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

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

在這個例子中，建立了一個使用下列功能的 pod： `san-pvc` 。

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

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

步驟 **4**：展開 **PV**

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

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

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

步驟 **5**：驗證擴展

您可以檢查PVC、PV和Trident的體積來驗證擴容是否正確：

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

擴大 **FC** 體積

您可以使用 CSI 設定程式擴充 FC 持久性磁碟區 (PV)。

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

步驟 **1**：設定 **StorageClass** 以支援磁碟區擴展

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

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

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

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

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

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

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

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

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

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

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

步驟 3：定義一個連接 **PVC** 的艙體

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

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

在這個例子中，建立了一個使用下列功能的 pod： `san-pvc` 。

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

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

步驟 **4**：展開 **PV**

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

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

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

步驟 **5**：驗證擴展

您可以檢查PVC、PV和Trident的體積來驗證擴容是否正確：

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

### 擴展 NFS 卷

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

#### 步驟 1：設定 StorageClass 以支援磁碟區擴展

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

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

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

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

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

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

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

Trident應該為該 PVC 建立一個 20 MiB 的 NFS PV：

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

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

步驟 3：擴充 **PV**

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

```
kubectl edit pvc ontapnas20mb
```

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

步驟 **4**：驗證擴展

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

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

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

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

## 進口量

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

### 概述和注意事項

您可以將磁碟區匯入Trident以執行下列操作：

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

### 注意事項

在匯入磁碟區之前，請考慮以下事項。

- Trident只能匯入 RW（讀寫）類型的ONTAP區。 DP（資料保護）類型磁碟區是SnapMirror目標磁碟區。在將磁碟區導入Trident之前，應該先斷開鏡像關係。

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

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

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

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

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

導入磁碟區

您可以使用 `tridentctl import`導入卷。

步驟

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

   以下是一個最低規格範例：

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

   不要包含其他參數，例如 PV 名稱或體積大小。這可能會導致導入命令失敗。

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

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

範例

請查看以下卷導入範例，以了解支援的驅動程式。

#### ONTAP NAS 和ONTAP NAS FlexGroup

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

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

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

#### ONTAP NAS 範例

下面展示了託管磁碟區和非託管磁碟區匯入的範例。

託管磁碟區

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

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

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

未託管卷

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

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

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

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

#### ONTAP SAN

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

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

**ONTAP SAN** 範例

下面展示了託管磁碟區和非託管磁碟區匯入的範例。

託管磁碟區

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

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

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

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

未託管卷

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

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

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

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

元素

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

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

元素範例

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

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

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

谷歌雲端平台

Trident支援使用以下方式導入磁碟區： `gcp-cvs`司機。

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

**Google Cloud Platform** 範例

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

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

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

#### Azure NetApp Files

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

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

#### Azure NetApp Files範例

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

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

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

#### Google Cloud NetApp Volumes

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

#### Google Cloud NetApp Volumes範例

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

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

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

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

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

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

# 自訂磁碟區名稱和標籤

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

## 開始之前

支援自訂磁碟區名稱和標籤：

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

## 限制

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

## 可自訂磁碟區名稱的關鍵行為

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

## 後端設定範例，包含名稱範本和標籤

可以在根層級和/或池層級定義自訂名稱範本。

根級別範例

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

池級範例

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

名稱範本範例

範例 **1**：

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

範例 **2**：

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

需要考慮的幾點

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

## 跨命名空間分享 **NFS** 卷

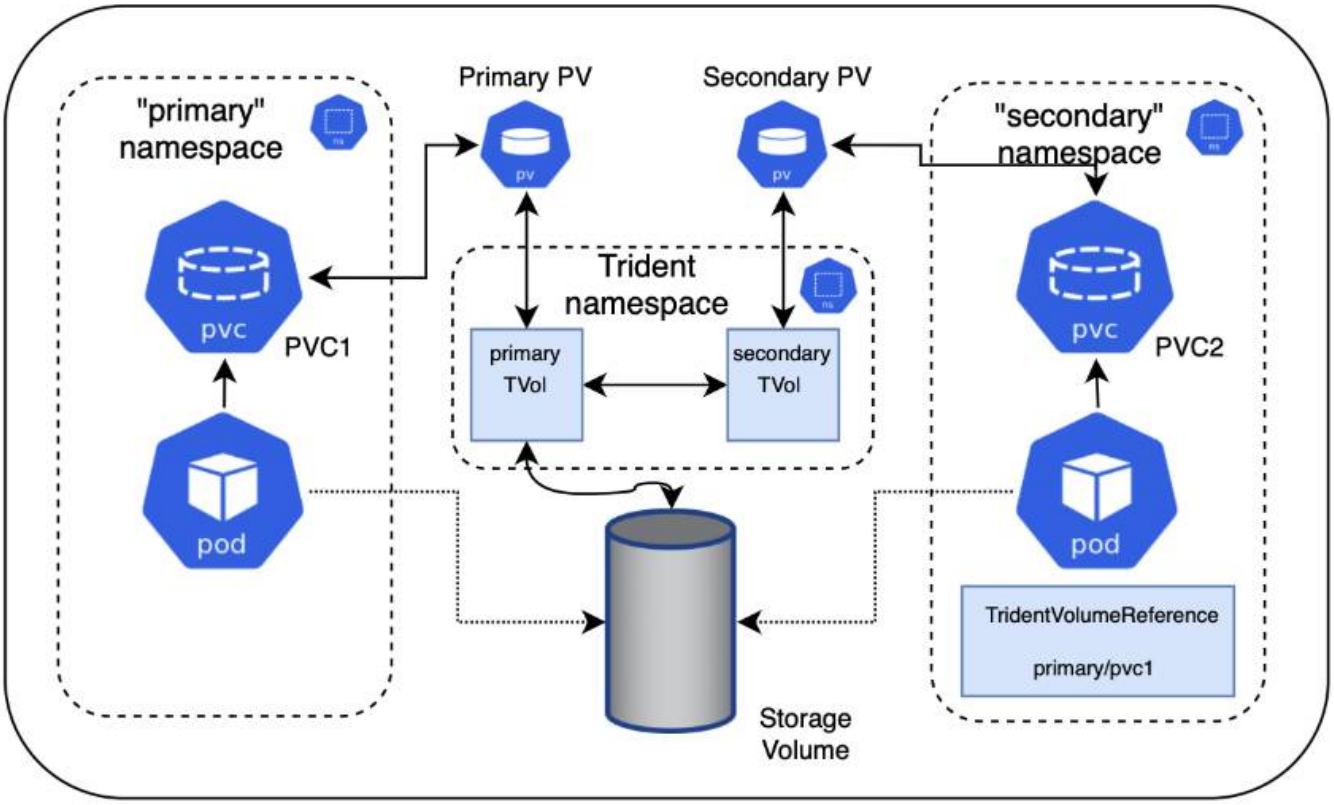
使用Trident，您可以在主命名空間中建立磁碟區，並在一個或多個輔助命名空間中共用該磁碟區。

特徵

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

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

此圖展示了跨兩個 Kubernetes 命名空間的 NFS 磁碟區共用。




快速啟動

只需幾個步驟即可設定NFS卷共享。

1 配置源**PVC**以共享體積

來源命名空間擁有者授予存取來源 PVC 中資料的權限。

2 授予在目標命名空間中建立 **CR** 的權限

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

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

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

4 在目標命名空間中建立從屬 **PVC**

目標命名空間的擁有者建立從屬 PVC，以使用來源 PVC 中的資料來源。

配置來源命名空間和目標命名空間

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

步驟

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

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

Trident建立 PV 及其後端 NFS 儲存磁碟區。

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

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

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

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

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

目標PVC管的尺寸必須小於或等於源PVC管的尺寸。

結果

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

刪除共享卷

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

使用 **`tridentctl get`**查詢子卷

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

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

標誌：

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

限制

- Trident無法阻止目標命名空間寫入共享磁碟區。您應該使用文件鎖定或其他方法來防止覆蓋共享卷資料。

- 您無法透過移除來源PVC來撤銷對來源PVC的存取權限。 `shareToNamespace`或者 `shareFromNamespace`註釋或刪除 `TridentVolumeReference`CR。若要撤銷存取權限，必須刪除從屬PVC。
- 從屬磁碟區無法進行快照、複製和鏡像操作。

更多資訊

要了解有關跨命名空間卷訪問的更多資訊：

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

## 跨命名空間克隆卷

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

先決條件

在克隆卷之前，請確保來源後端和目標後端是同一類型，並且具有相同的儲存類別。

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

快速啟動

只需幾個步驟即可設定卷克隆。

1 配置來源 **PVC** 以克隆卷

來源命名空間擁有者授予存取來源 PVC 中資料的權限。

2 授予在目標命名空間中建立 **CR** 的權限

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

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

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

4 在目標命名空間中建立克隆 **PVC**

目標命名空間的擁有者建立 PVC 以複製來源命名空間中的 PVC。

配置來源命名空間和目標命名空間

為確保安全，跨命名空間複製磁碟區需要來源命名空間擁有者、叢集管理員和目標命名空間擁有者的協作和操

作。使用者角色在每個步驟中都會被指定。

步驟

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

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

Trident建立 PV 及其後端儲存磁碟區。

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

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

3. 目標命名空間擁有者： 在目標命名空間中建立一個指向來源命名空間的 TridentVolumeReference CR。`pvc1` 。

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

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

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

限制

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

## 使用SnapMirror複製卷

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

- 在體積（PVC）之間建立鏡像關係
- 移除磁碟區之間的鏡像關係
- 打破鏡像關係
- 在災難情況下（故障轉移）提升備用磁碟區的可用性
- 在計劃內故障轉移或遷移期間，實現應用程式在叢集間的無損遷移。

複製的前提條件

在開始之前，請確保滿足以下先決條件：

**ONTAP叢集**

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

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

僅支援SnapMirror非同步保護。

對等互連

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

確保兩個ONTAP叢集之間複製關係中使用的 SVM 名稱是唯一的。

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

支援的驅動程式

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

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

製作鏡面**PVC**

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

步驟

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

      例子

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

   b. 使用先前建立的 StorageClass 建立 PVC。

例子

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

c. 使用本機資訊建立鏡像關係變更要求。

例子

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

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

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

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

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

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

      例子

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

   b. 建立包含目標和來源資訊的鏡像關係 CR。

      例子

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

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

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

例子

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

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

卷複製狀態

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

- 已建立：本地 PVC 是鏡像關係的目標量，這是一個新的關係。
- 已推廣：本地 PVC 可讀寫且可安裝，目前沒有鏡像關係。
- 重新建立：本地 PVC 是鏡像關係的目標量，之前也處於該鏡像關係中。
  - 如果目標磁碟區曾經與來源磁碟區有關聯，則必須使用重新建立的狀態，因為它會覆寫目標磁碟區的內容。
  - 如果磁碟區之前未與來源建立關係，則重新建立狀態將會失敗。

在計劃外故障切換期間促進輔助**PVC**的運行

在輔助 Kubernetes 叢集上執行下列步驟：

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

在計劃故障切換期間推廣備用**PVC**

在計劃故障轉移（遷移）期間，執行以下步驟以提升輔助 PVC：

步驟

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

   例子

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

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

故障轉移後恢復鏡像關係

在恢復鏡像關係之前，選擇你想作為新主要方的那一方。

步驟

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

附加手術

Trident支援對主磁碟區和輔助磁碟區執行下列操作：

#### 將主**PVC**複製到新的輔助**PVC**

請確保您已備有主PVC管及備用PVC管。

步驟

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

#### 調整鏡像、主或次級**PVC**的尺寸

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

#### 從 **PVC** 移除複製

若要移除複製，請對目前輔助磁碟區執行下列其中一項操作：

- 刪除輔助 PVC 上的鏡像關係。這會破壞複製關係。

- 或者，將 spec.state 欄位更新為 *promoted*。

#### 刪除一個**PVC**（之前已鏡像）

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

#### 刪除 **TMR**

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

#### **ONTAP**在線時，更新鏡像關係

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

#### **ONTAP**離線時更新鏡像關係

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

例子

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

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

## 使用 **CSI** 拓撲

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

#### 概況

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

了解更多關於 CSI 拓撲功能的信息 "這裡"。

Kubernetes 提供了兩種獨特的磁碟區綁定模式：

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

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

你需要什麼

要使用 CSI 拓撲結構，您需要以下元件：

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

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

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

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

步驟 **1**：建立拓樸感知後端

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

以下是一個後端定義範例：

**YAML**

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

**JSON**

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

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

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

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

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

步驟 **2**：定義拓樸感知的儲存類別。

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

請參閱以下範例：

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

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

步驟 **3**：製作並使用 **PVC**

建立 StorageClass 並將其對應到後端後，現在可以建立 PVC。

請參閱範例 `spec`以下：

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

使用此清單建立 PVC 將產生以下結果：

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

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

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

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

請查看以下輸出：

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

更新後端以包含 `supportedTopologies`

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

查找更多信息

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

## 使用快照

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

### 概況

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

#### 開始之前

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

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

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

建立磁碟區快照

步驟

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

     例子

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

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

2. 建立現有PVC的快照。

   範例

   - 此範例建立現有 PVC 的快照。

     ```
     cat snap.yaml
     ```

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

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

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

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

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

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

從磁碟區快照建立**PVC**

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

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

以下範例使用以下方法建立 PVC： `pvc1-snap`作為資料來源。

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

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

匯入磁碟區快照

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

開始之前

Trident肯定建立或匯入了快照的父卷。

步驟

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

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

     例子

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

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

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

   例子

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

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

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

   例子

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

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

使用快照恢復磁碟區數據

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

使用磁碟區快照還原ONTAP CLI 將磁碟區還原到先前快照中記錄的狀態。

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

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

從快照進行原地體積恢復

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

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

開始之前

您必須擁有已綁定的PVC和可用的磁碟區快照。

- 確認PVC狀態是否已綁定。

```
kubectl get pvc
```

- 確認磁碟區快照已準備就緒。

```
kubectl get vs
```

步驟

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

   TASR CR 必須位於 PVC 和 VS 存在的命名空間。

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

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

2. 應用 CR 從快照恢復。此範例從快照恢復 `pvc1`。

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

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

結果

Trident從快照中復原資料。您可以驗證快照復原狀態：

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

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

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

### 刪除包含關聯快照的 PV

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

### 部署磁碟區快照控制器

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

#### 步驟

1. 建立磁碟區快照 CRD。

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

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

2. 建立快照控制器。

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

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

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

相關連結

- "卷快照"
- "卷快照類"

## 使用磁碟區組快照

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

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

## 建立卷宗組快照

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

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

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

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

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

步驟

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

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

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

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

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

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

  此範例建立磁碟區組快照：

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

使用群組快照恢復磁碟區數據

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

使用磁碟區快照還原ONTAP CLI 將磁碟區還原到先前快照中記錄的狀態。

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

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

從快照進行原地體積恢復

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

有關詳細信息，請參閱 "從快照進行原地體積恢復"。

刪除包含關聯群組快照的 **PV**

刪除群組磁碟區快照時：

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

部署磁碟區快照控制器

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

步驟

1. 建立磁碟區快照 CRD。

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

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

2. 建立快照控制器。

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

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

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

相關連結

- "卷冊組快照類"
- "卷快照"

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。