vSphere Kubernetes Service 搭配 NetApp 儲存設備

NetApp container solutions

NetApp
September 01, 2026

目錄

vSphere Kubernetes Service 搭配 NetApp 儲存設備

了解 VMware vSphere Kubernetes Service with NetApp ONTAP

VMware vSphere Kubernetes Service （VKS） 是 VMware 的代管 Kubernetes 產品，使企業能夠使用 Kubernetes 部署、管理和擴充容器化應用程式。與 NetApp ONTAP 儲存設備結合，VKS 可為雲端原生工作負載提供企業級持續性、效能和資料管理。

VKS 嵌入在 VMware Cloud Foundation (VCF) 中，並符合上游 CNCF Kubernetes 標準，確保與更廣泛的 Kubernetes 生態系統和工具相容。

vSphere Kubernetes Service 的主要特性

1. 代管 **Kubernetes** 叢集： vSphere Kubernetes Service 簡化了 Kubernetes 叢集的部署和管理。它可自動執行叢集建立、升級、擴充和監控等任務。
2. 與 **VMware** 基礎架構整合： VKS 可與 VMware vSphere、NSX-T 及其他 VMware 解決方案無縫地整合，使組織能夠運用其現有的 VMware 基礎架構來執行 Kubernetes 工作負載。
3. 多雲和混合雲支援： vSphere Kubernetes 服務支援在私有雲、公有雲（例如 AWS、Azure、Google Cloud ）和混合雲環境中部署，讓組織在管理其應用程式時擁有更大的靈活度。
4. 內建安全功能： VKS 包含角色型存取控制 （RBAC）、原則執行以及與 VMware NSX 整合以實現網路安全性等安全功能。
5. 對 **Kubernetes** 生態系統的支援： VKS 支援完整的上游 Kubernetes API，確保跨環境的可移植性，並與更廣泛的 Kubernetes 生態系統相容，包括用於服務網格的 Istio、用於監控的 Prometheus 以及其他 CNCF 認證的工具。組織可以直接將標準的 Kubernetes 技能和整合應用於 VKS 叢集。
6. 對開發者友善的工具： VKS 支援標準的 Kubernetes 開發者工作流程，包括 CI/CD 管線、GitOps 實踐以及 kubectl 和 Helm 等工具，使開發團隊能夠建置和部署容器化應用程式，而無需學習專屬介面。
7. 擴充性與高可用度： VKS 提供自動擴充和容錯能力等功能，確保應用程式保持高可用度的且高效能。
8. 可觀測性和監控： VKS 與標準的 Kubernetes 相容監控工具整合，以提供有關叢集和應用程式效能的即時深入解析。

vSphere Kubernetes Service 的使用案例

1. 應用程式現代化： 組織可以透過將傳統應用程式容器化，並將其部署到由 VKS 代管的 Kubernetes 叢集上，實現應用程式現代化。
2. **DevOps** 開發維運賦能： VKS 透過提供自動化、CI/CD 整合和對開發者友善的工具來支援 DevOps 開發維運工作流程。
3. *混合雲部署：*企業可以使用 VKS 在本地和雲端環境中部署 Kubernetes 叢集，實現一致的操作。
4. 微服務架構： VKS 支援基於微服務的應用程式開發，使團隊能夠建置和擴充分散式系統。

儲存設備與 **vSphere Kubernetes** 服務入門

安裝 **vSphere Kubernetes Service** 叢集

在 VCF 9.1 環境中安裝 vSphere Kubernetes Service (VKS) 叢集，並為工作負載設定具有適當儲存設備公用程式的工作節點。

關於此任務

您建立的叢集的工作節點會自動安裝 NFS 工具。如果您希望建立安裝了 iSCSI 或 NVMe 工具的工作節點，則必須建立自訂 OVA 映像並將其用於工作節點。可以使用 Docker 建立自訂映像，然後使用 `vcf`命令列工具從該映像建立 OVA 檔案。此 OVA 檔案可以上傳到 vSphere 的內容庫。建立 VKS 叢集時，可以從內容庫中選擇此映像用於節點池。

步驟

1. 建立 VKS 叢集：

   您可以使用預設工作節點映像或自訂映像建立 VKS 叢集。預設工作節點映像預先安裝了 NFS 工具，而自訂映像可以包含 iSCSI 和 NVMe 工具。以下選項可供使用：

   - 僅限 **NAS**（預設）： 使用預設工作節點映像建立 VKS 叢集，該映像包含預先安裝的 NFS 公用程式。
   - 啟用 **SAN**（自訂映像）： 建立一個包含 iSCSI 和 NVMe 公用程式的自訂 Docker 映像，使用 `vcf`命令列工具產生 OVA 檔案，將 OVA 上傳到 vSphere 內容庫，然後建立 VKS 叢集，為節點資源池選擇該自訂映像。

   ▼ 顯示使用預設工作節點映像建立僅包含 **NAS** 的 **vSphere Kubernetes Service** 叢集的說明

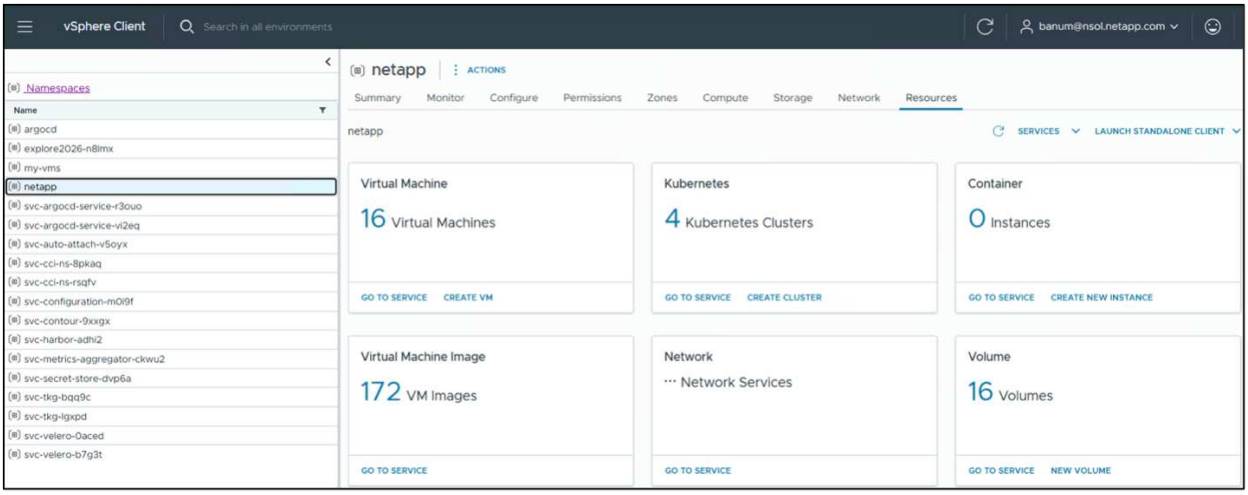
   使用預設工作節點映像建立 VKS 叢集。依照提示指定叢集名稱、節點資源池組態和其他設定。

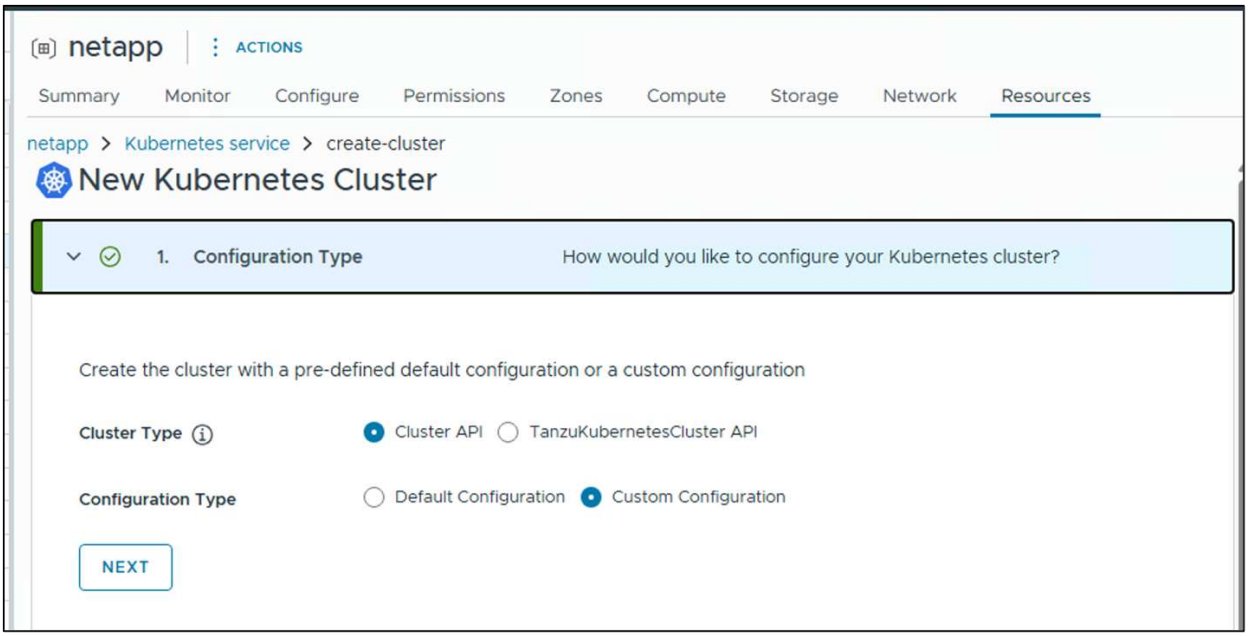
   在 vSphere Client 中，前往要建立 vSphere Kubernetes 叢集的命名空間。在該命名空間的 **Resources** 索引標籤上，按一下 Kubernetes 區段中的 **Create Cluster**，或按一下 **Go to Service**，然後按一下 **Create Cluster**。

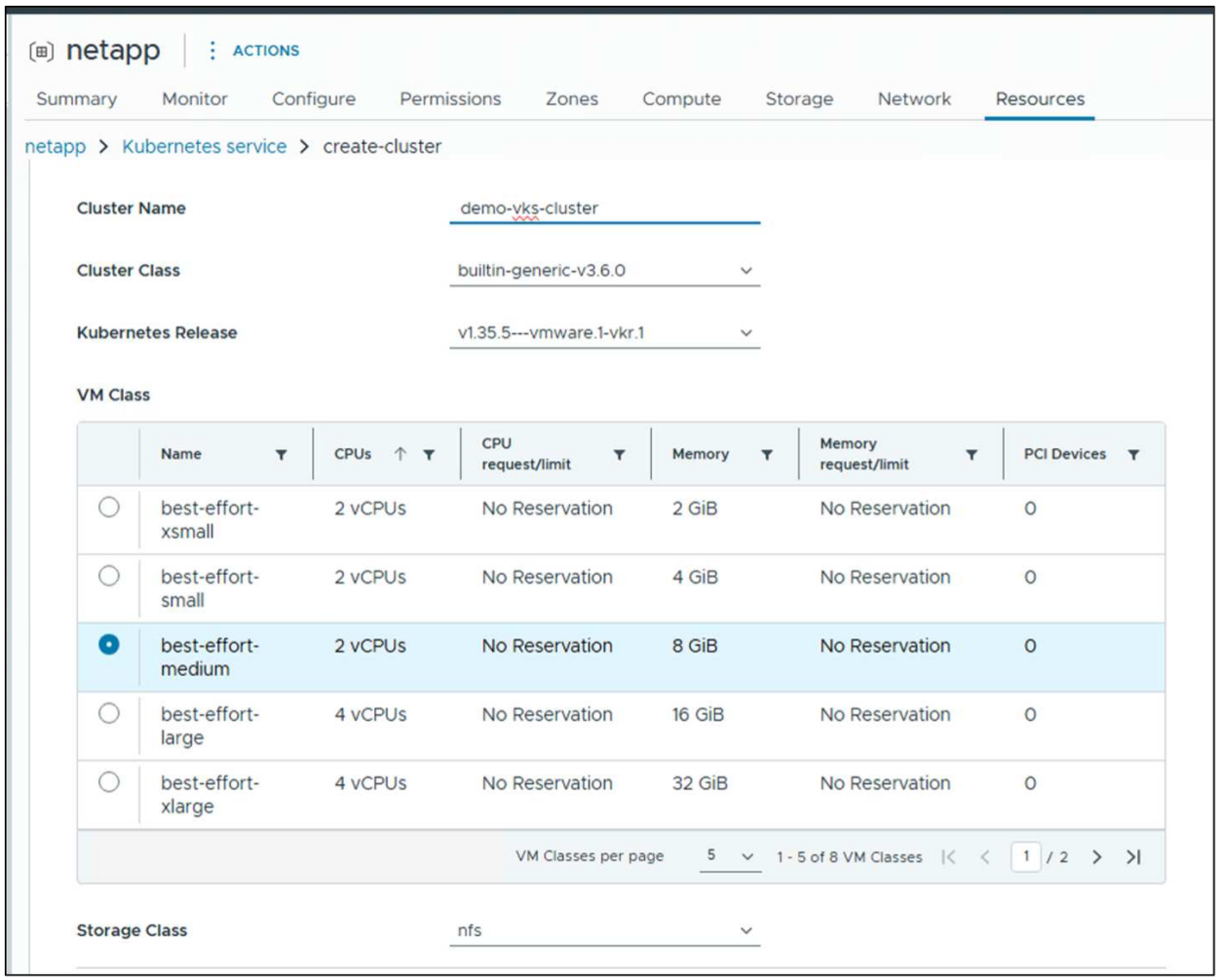
   請填寫表格，提供集群詳細資料：

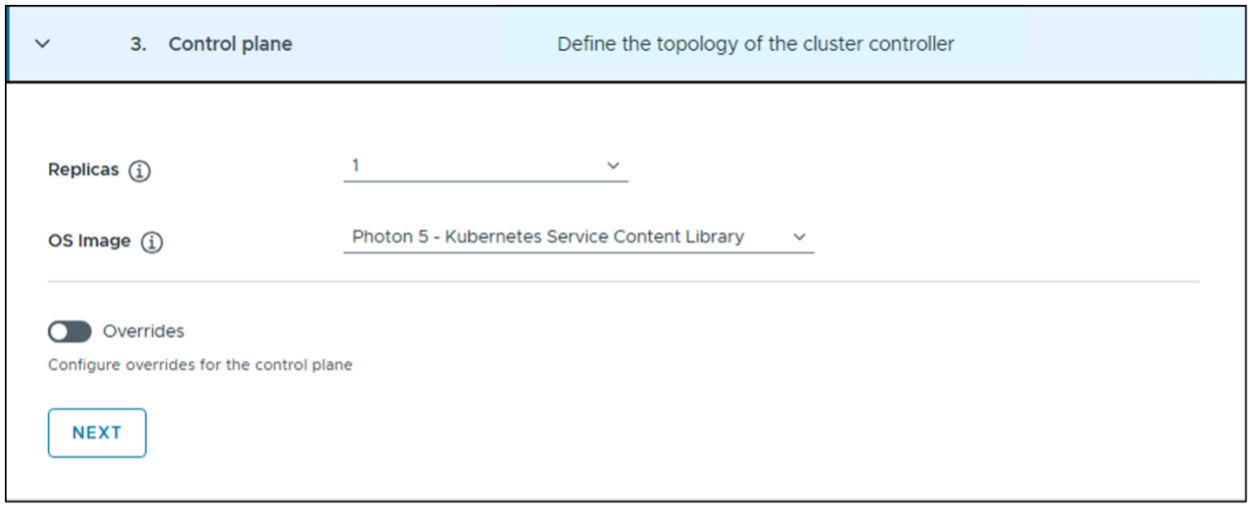
   - 在第 1 節「組態類型」中，選取「自訂」。
   - 在第 2 部分「一般設定」中，為叢集提供名稱，並將所有其他設定保留為預設值。
   - 第 3 部分的所有設定均接受預設值。
   - 在第 4 部分「節點資源池」中，按一下節點資源池旁的省略號（……），然後按一下「編輯」。將副本數增加到 3，並選擇最新的 Ubuntu 作業系統映像版本。按一下「下一步」，然後按一下「檢視並確認」。

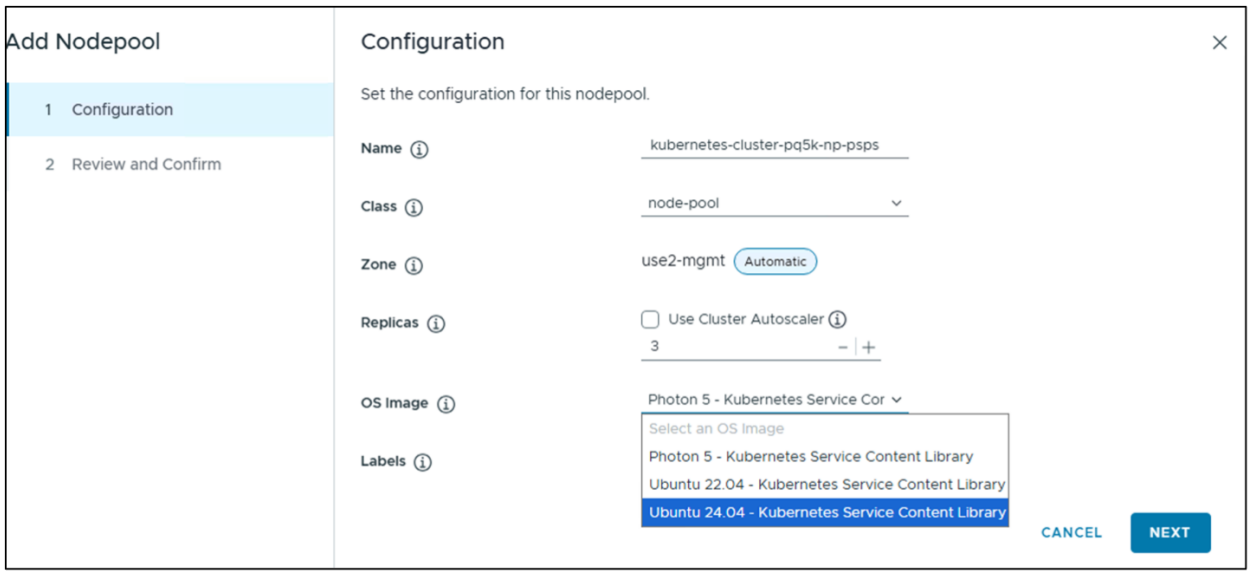
   查看叢集詳細資訊後，按一下 **Finish**。vSphere Kubernetes 叢集將在幾分鐘內建立完成。

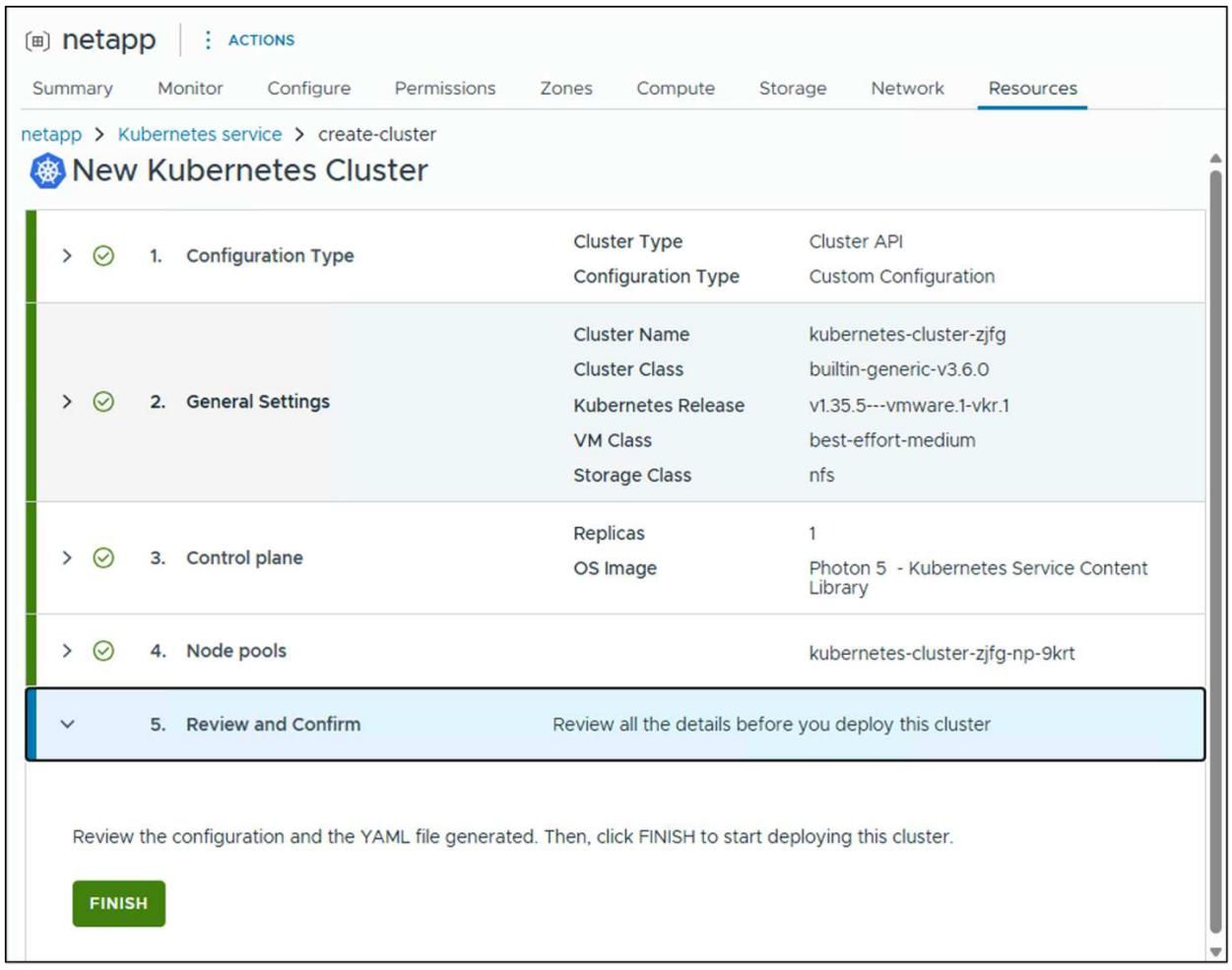

vSphere Client
Search in all environments
banum@nsol.netapp.com
Namespaces
Name
argocd
explore2026-n8lmx
my-vms
netapp
svc-argocd-service-r3ouo
svc-argocd-service-vi2eq
svc-auto-attach-v5oyx
svc-cci-ns-8pkaq
svc-cci-ns-rsqfv
svc-configuration-m0i9f
svc-contour-9xxgx
svc-harbor-adhi2
svc-metrics-aggregator-ckwu2
svc-secret-store-dvp6a
svc-tkg-bqq9c
svc-tkg-lgxpd
svc-velero-0aced
svc-velero-b7g3t
netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp
SERVICES
LAUNCH STANDALONE CLIENT
Virtual Machine
16 Virtual Machines
GO TO SERVICE
CREATE VM
Kubernetes
4 Kubernetes Clusters
GO TO SERVICE
CREATE CLUSTER
Container
0 Instances
GO TO SERVICE
CREATE NEW INSTANCE
Virtual Machine Image
172 VM Images
GO TO SERVICE
Network
Network Services
GO TO SERVICE
Volume
16 Volumes
GO TO SERVICE
NEW VOLUME

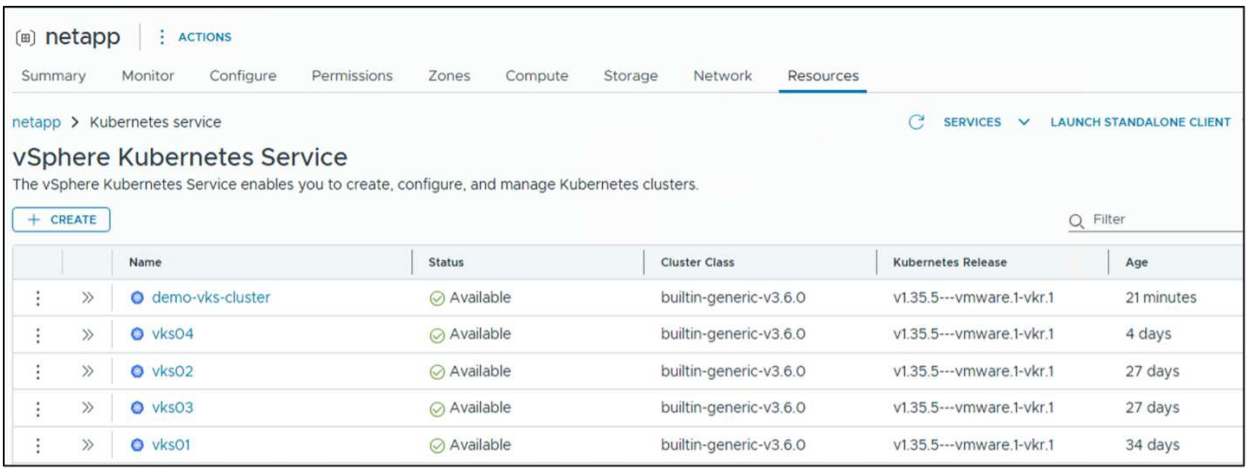

netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp > Kubernetes service > create-cluster
New Kubernetes Cluster
1. Configuration Type
How would you like to configure your Kubernetes cluster?
Create the cluster with a pre-defined default configuration or a custom configuration
Cluster Type
Cluster API
TanzuKubernetesCluster API
Configuration Type
Default Configuration
Custom Configuration
NEXT


netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp > Kubernetes service > create-cluster
Cluster Name
demo-vks-cluster
Cluster Class
builtin-generic-v3.6.0
Kubernetes Release
v1.35.5---vmware.1-vkr.1
VM Class
Name
CPUs
CPU request/limit
Memory
Memory request/limit
PCI Devices
best-effort-xsmall
2 vCPUs
No Reservation
2 GiB
No Reservation
0
best-effort-small
2 vCPUs
No Reservation
4 GiB
No Reservation
0
best-effort-medium
2 vCPUs
No Reservation
8 GiB
No Reservation
0
best-effort-large
4 vCPUs
No Reservation
16 GiB
No Reservation
0
best-effort-xlarge
4 vCPUs
No Reservation
32 GiB
No Reservation
0
VM Classes per page
5
1 - 5 of 8 VM Classes
1 / 2
Storage Class
nfs


3. Control plane
Define the topology of the cluster controller
Replicas
1
OS Image
Photon 5 - Kubernetes Service Content Library
Overrides
Configure overrides for the control plane
NEXT


Add Nodepool
1 Configuration
2 Review and Confirm
Configuration
Set the configuration for this nodepool.
Name
kubernetes-cluster-pq5k-np-psps
Class
node-pool
Zone
use2-mgmt
Automatic
Replicas
Use Cluster Autoscaler
3
OS Image
Photon 5 - Kubernetes Service Cor
Select an OS Image
Photon 5 - Kubernetes Service Content Library
Ubuntu 22.04 - Kubernetes Service Content Library
Ubuntu 24.04 - Kubernetes Service Content Library
Labels
CANCEL
NEXT


netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp > Kubernetes service > create-cluster
New Kubernetes Cluster
1. Configuration Type
Cluster Type
Cluster API
Configuration Type
Custom Configuration
2. General Settings
Cluster Name
kubernetes-cluster-zjfg
Cluster Class
builtin-generic-v3.6.0
Kubernetes Release
v1.35.5---vmware.1-vkr.1
VM Class
best-effort-medium
Storage Class
nfs
3. Control plane
Replicas
1
OS Image
Photon 5 - Kubernetes Service Content Library
4. Node pools
kubernetes-cluster-zjfg-np-9krt
5. Review and Confirm
Review all the details before you deploy this cluster
Review the configuration and the YAML file generated. Then, click FINISH to start deploying this cluster.
FINISH



▼ 顯示為啟用 **SAN** 的工作節點建立自訂 **OVA** 映像的說明

建立已安裝所需工具的 Docker 映像。以下範例示範如何建立基於 Ubuntu 24.04 並安裝了 iSCSI 和多路徑工具的 Docker 映像。

```
#san-dockerfile.txt
FROM ubuntu:24.04
# Prevent interactive prompts during package installation
ENV DEBIAN_FRONTEND=noninteractive
# Install required system packages for iSCSI and Multipathing
RUN apt-get update && \
    apt-get install -y --no-install-recommends \
    nvme-cli \
    kmod \
    linux-tools-common \
    open-iscsi \
    lsscsi \
    sg3-utils \
    multipath-tools \
    scsitools  \
    chrony \
    networkd-dispatcher &&\
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
# Set iSCSI session scanning to manual as required by Trident
RUN sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
# Configure device-mapper-multipath with Trident recommendations
RUN printf "defaults {\n    user_friendly_names yes\n
find_multipaths no\n}\n" > /etc/multipath.conf
RUN mkdir -p /etc/networkd-dispatcher/off.d /etc/networkd-
dispatcher/routable.d/
CMD ["bash"]
```

使用以下命令建置 Docker 映像：

```
docker build -f san-dockerfile.txt -t ubuntu-2404:san .
```

如果 Docker 映像倉庫不可用，請執行下列命令啟動本機映像倉庫並將映像推送到該倉庫：

```
docker run -d -p 5000:5000 --restart=always --name registry
registry:2
```

為映像添加標籤並推送。

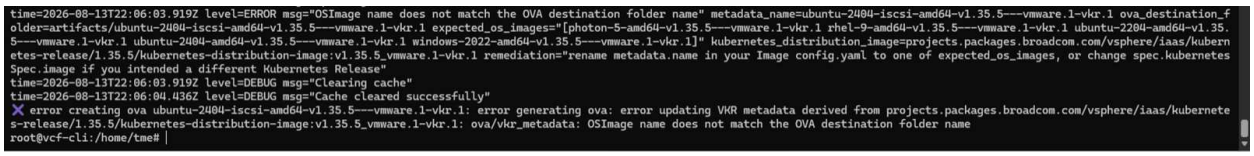
```
docker tag ubuntu-2404:san localhost:5000/ubuntu-2404:san
docker push localhost:5000/ubuntu-2404:san
```

建立映像組態檔案（另存為 `ubuntu2404vkr135-san.yml`），並引用該映像：

```
#ubuntu2404vkr135-san.yml
apiVersion: imageconfiguration.vmware.com/v1alpha1
kind: Image
metadata:
    name: ubuntu-2404-amd64-v1.35.5---vmware.1-vkr.1
spec:
    osSpec:
        name: ubuntu
        version: "24.04"
        image: "localhost:5000/ubuntu-2404:san"
    kubernetesSpec:
        image:
projects.packages.broadcom.com/vsphere/iaas/kubernetes-
release/1.35.5/kubernetes-distribution-image:v1.35.5_vmware.1-vkr.1
    diskSize: 20Gi
    repositorySpec:
        debs:
            - name: noble
              url: http://archive.ubuntu.com/ubuntu
              suites:
                  - noble
              components:
                  - main
                  - restricted
                  - universe
                  - multiverse
            - name: noble-security
```

```
            url: http://security.ubuntu.com/ubuntu
            suites:
                - noble-security
            components:
                - main
                - restricted
                - universe
                - multiverse
    packages:
      present:                    # Packages to install
        - name: open-iscsi
        - name: lsscsi
        - name: sg3-utils
        - name: multipath-tools
        - name: scsitools
        - name: nvme-cli
    services:
      - name: multipath-tools.service
        status: enabled
      - name: open-iscsi.service
        status: enabled
```

中繼資料名稱必須來自預先核准的清單。否則，嘗試建立 OVA 檔案時會收到如下所示的錯誤。預先核准的清單可在 VCF CLI 說明的 `kr bake`命令中找到。

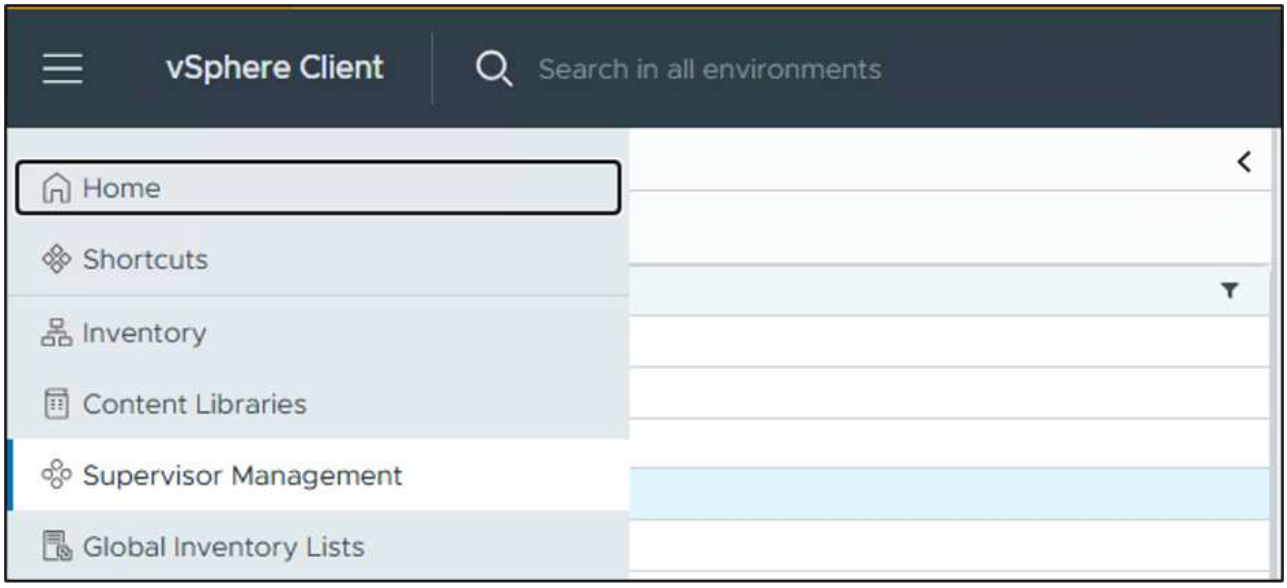


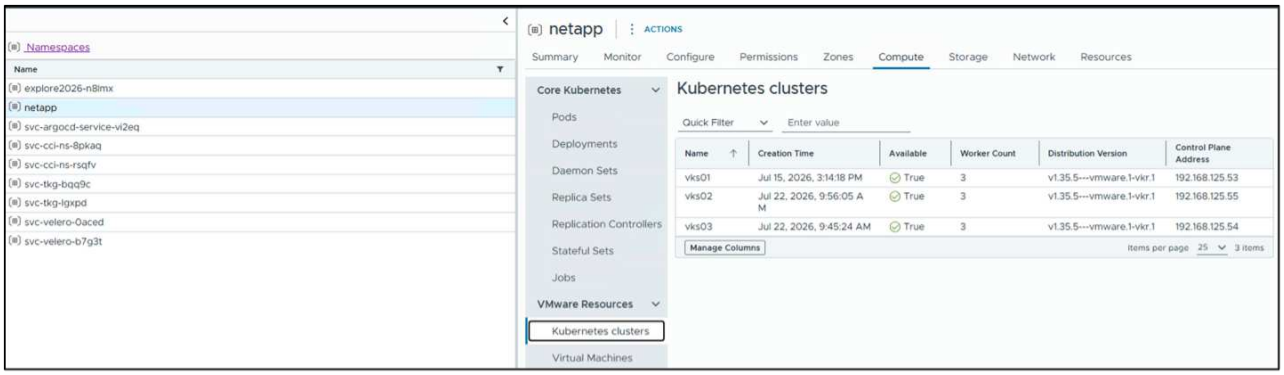
使用 VCF CLI 建立 OVA 檔案：

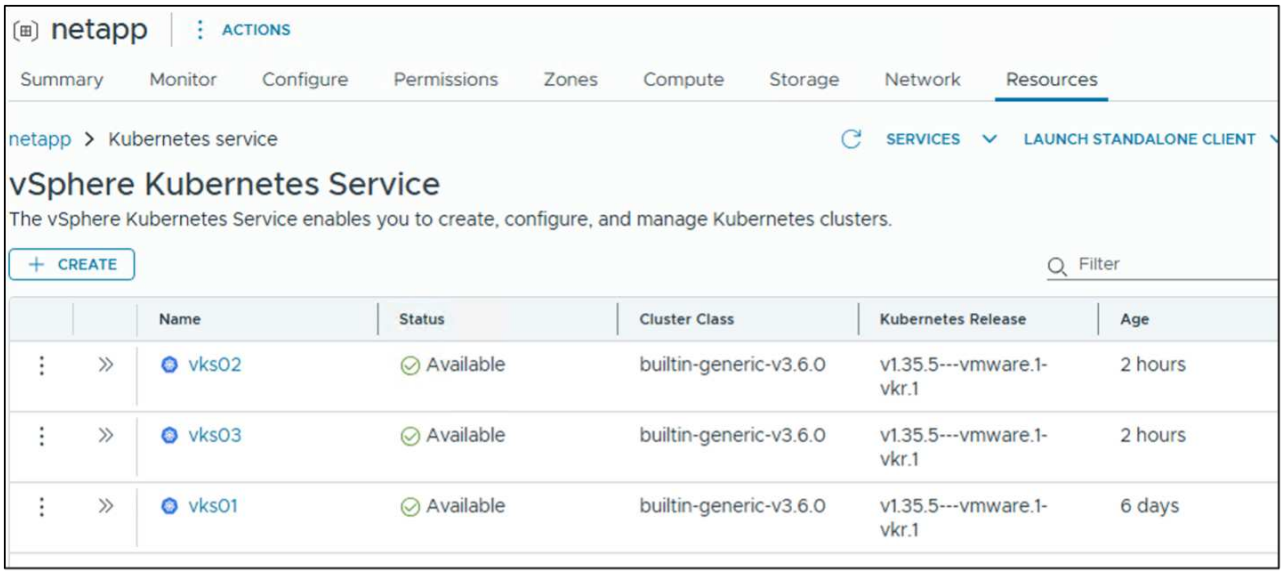
```
vcf kr bake -f <path/to/ubuntu2404vkr135-san.yml> --log-level DEBUG
--log-format plain
```

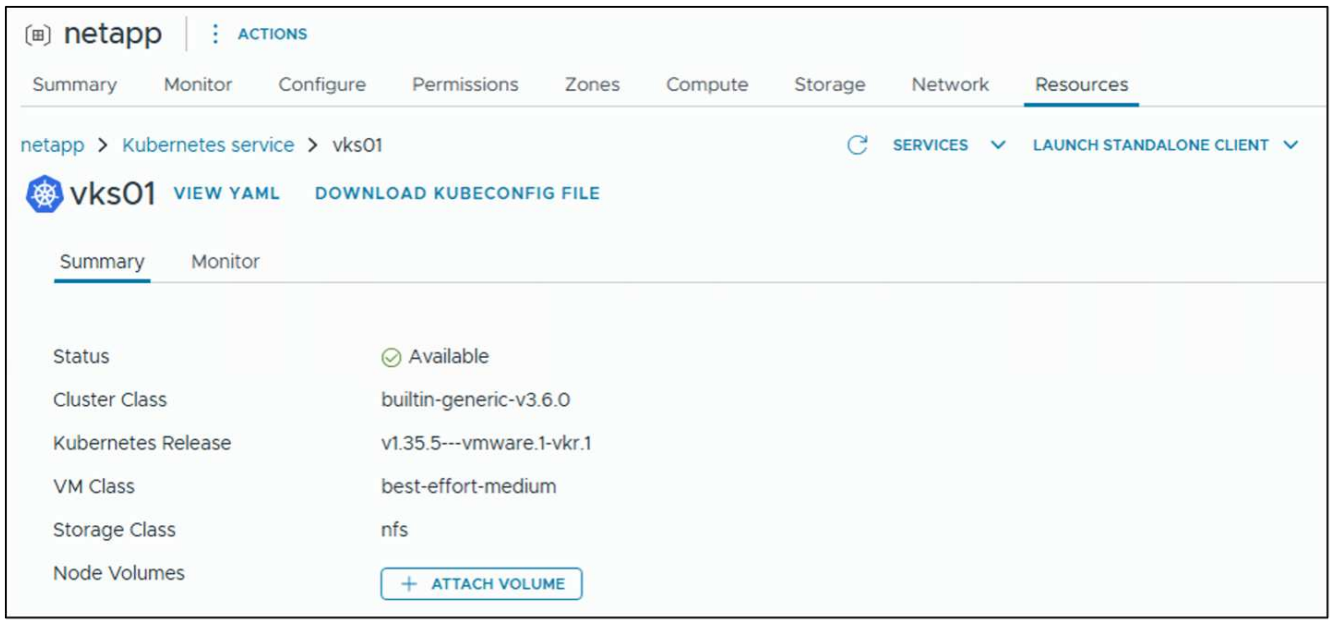
使用 SCP 將 OVA 檔案傳輸到可以將其上傳至 vSphere 內容庫的機器。

2. 叢集安裝完成後，在 vCenter 中找到它並下載 kubeconfig 檔案。
    a. 從主選單中點選 **Supervisor Management**，然後選擇建立叢集的命名空間。
    b. 選取 **Compute** 索引標籤，然後選取 **Kubernetes Clusters**。命名空間中的所有叢集都會顯示。
    c. 前往「資源」索引標籤，選取所需的 Kubernetes 叢集，並下載其 kubeconfig 檔案。


vSphere Client
Search in all environments
Home
Shortcuts
Inventory
Content Libraries
Supervisor Management
Global Inventory Lists


Namespaces
Name
explore2026-n8lmx
netapp
svc-argocd-service-vi2eq
svc-cci-ns-8pkaq
svc-cci-ns-rsqfv
svc-tkg-bqq9c
svc-tkg-lgxpd
svc-velero-0aced
svc-velero-b7g3t
netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
Core Kubernetes
Pods
Deployments
Daemon Sets
Replica Sets
Replication Controllers
Stateful Sets
Jobs
VMware Resources
Kubernetes clusters
Virtual Machines
Kubernetes clusters
Quick Filter
Enter value
Name
Creation Time
Available
Worker Count
Distribution Version
Control Plane Address
vks01
Jul 15, 2026, 3:14:18 PM
True
3
v1.35.5---vmware.1-vkr.1
192.168.125.53
vks02
Jul 22, 2026, 9:56:05 AM
True
3
v1.35.5---vmware.1-vkr.1
192.168.125.55
vks03
Jul 22, 2026, 9:45:24 AM
True
3
v1.35.5---vmware.1-vkr.1
192.168.125.54
Manage Columns
Items per page
25
3 items


netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp > Kubernetes service
SERVICES
LAUNCH STANDALONE CLIENT
vSphere Kubernetes Service
The vSphere Kubernetes Service enables you to create, configure, and manage Kubernetes clusters.
CREATE
Filter
Name
Status
Cluster Class
Kubernetes Release
Age
vks02
Available
builtin-generic-v3.6.0
v1.35.5---vmware.1-vkr.1
2 hours
vks03
Available
builtin-generic-v3.6.0
v1.35.5---vmware.1-vkr.1
2 hours
vks01
Available
builtin-generic-v3.6.0
v1.35.5---vmware.1-vkr.1
6 days



3. 從堡壘主機，使用 kubeconfig 檔案連線至叢集，以便透過 CLI 進行管理。

```
vksbastion@vks:~$ kubectl get nodes
NAME                                    STATUS   ROLES           AGE    VERSION
vks01-dhwbg-rpxst                       Ready    control-plane   6d20h  v1.35.5+vmware.1
vks01-vks01-np-pt27-764sx-cdd5q-g5qnw   Ready    <none>          3h45m  v1.35.5+vmware.1
vks01-vks01-np-pt27-764sx-cdd5q-ltj5s   Ready    <none>          6d20h  v1.35.5+vmware.1
vks01-vks01-np-pt27-764sx-cdd5q-npxmj   Ready    <none>          6d20h  v1.35.5+vmware.1
```

影片示範

以下影片示範了建立 vSphere Kubernetes 叢集的兩種方法。

在 Supervisor 叢集上建立 vSphere Kubernetes 叢集

使用 VCF 自動化建立 vSphere Kubernetes 集群

瞭解適用於 **vSphere Kubernetes** 服務的 **NetApp** 儲存設備

使用 NetApp 作為 vSphere Kubernetes Service （VKS） 的儲存設備，是一種強大的組合，能夠運用 NetApp 強大的儲存設備解決方案，提升 Kubernetes 工作負載的效能、擴充性和可靠性。NetApp Trident 是一款開源的 Kubernetes 動態儲存設備配置器，用於將儲存設備與 vSphere Kubernetes Service 上的工作負載整合。

搭配 **VKS** 使用 **NetApp** 儲存設備的主要效益

1. 動態儲存配置： NetApp Trident 允許動態配置儲存設備 Volume，使 Kubernetes Pod 能夠根據自身需求即時請求儲存設備。
2. 持久性儲存： NetApp 解決方案提供持續儲存功能，確保在 pod 重新啟動和故障期間資料的持久性和可用度。
3. 高效能： NetApp 的儲存設備解決方案，例如 ONTAP，提供低延遲的高效能儲存設備，非常適合在 Kubernetes 上執行的 I/O 密集型應用程式。

4. 擴充性： NetApp 儲存設備系統可擴充以滿足不斷成長的 Kubernetes 工作負載需求，支援大規模部署。
5. 資料管理： NetApp 提供進階資料管理員功能，包括 Snapshot 快照技術、複製和資料複寫，這對於備份、災難恢復和開發工作流程都很有幫助。
6. 多雲和混合雲支援： NetApp 的儲存設備解決方案可部署於本地、公有雲和混合雲環境中，在儲存管理方面提供靈活度與一致性。

NetApp ONTAP 整合總覽

NetApp ONTAP 提供企業級儲存設備，具備資料去重、壓縮和加密等功能。您可以使用 NetApp Trident 將 NetApp 儲存設備與 Kubernetes 整合。NetApp Trident 支援多種 NetApp 儲存設備平台，包括本地部署的 ONTAP、AWS 中的 FSx for NetApp ONTAP、Azure 中的 Azure NetApp Files，以及 Google Cloud 中的 Google Cloud NetApp Volumes。

您可以使用 Trident Protect 來管理 Kubernetes 工作負載的資料保護和災難恢復。Trident Protect 可讓您建立 Snapshot 快照技術、複製及跨不同儲存設備後端複寫資料，確保您的資料安全無虞，並在發生故障時可以恢復。

Trident 的主要功能

CSI 合規性 確保與最新的 Kubernetes 儲存設備功能和標準相容。聲明式設定 允許使用者在 YAML 檔案中定義儲存設備需求，Trident 將自動管理這些需求。驅動程式 Trident 支援 ONTAP 所支援的 NFS、CIFS/SMB 、iSCSI、FC 和 NVMe/TCP 協定的驅動程式。混合雲和多雲支援 Trident 支援 ONTAP 本機儲存設備以及超大規模雲端服務商中代管儲存設備服務的驅動程式，包括 AWS 中的 FSx for NetApp ONTAP、Azure 中的 Azure NetApp Files，以及 Google Cloud 中的 Google Cloud NetApp Volumes。進階資料管理 運用 ONTAP 的 Snapshot 快照技術和複製功能，實現高效的資料保護並快速部署測試與開發環境。

如需 Trident 的完整詳細資訊，請參閱 "Trident 文件"。

Trident Protect

Trident Protect 與 Trident 是分開安裝的。部署前，請驗證您的 Kubernetes 平台、儲存設備後端、Snapshot 快照技術元件和物件儲存是否符合"Trident Protect 要求"。

Trident Protect 的主要功能

自動備份 Trident Protect 可實現 Kubernetes 應用程式的自動、原則驅動型備份，確保定期備份而無需手動介入。

Snapshot 快照技術管理 它運用 ONTAP 的 Snapshot 快照技術功能，為 Container 應用程式和虛擬機器建立頻繁的時間點副本，提供可靠的恢復機制。

備份儲存庫加密 Trident Protect 支援對 Restic 和 Kopia 備份儲存庫進行密碼型加密。可在儲存設備層和網路層分別設定持續性 Volume 加密和傳輸中加密。

合規與審計 提供工具和功能，幫助組織滿足合規要求並定期稽核其資料保護實務。

災難復原 與 ONTAP 的複寫功能整合，提供強大的災難復原解決方案，即使面對災難性事件也能確保業務連續性。

有關 Trident Protect 的完整詳細資訊，請參閱"Trident Protect 說明文件"。

NetApp ONTAP 支援的硬體型號與通訊協定

NetApp ONTAP 軟體可在多種硬體型號上運行，每種型號都旨在滿足不同的效能和容量需求。Trident 擴展了其強大的功能，以支援各種 NetApp ONTAP 系統，包括全快閃 FAS （AFF）、全 SAN 陣列 （ASA） 和 ASA R2 系統。這種支援確保企業能夠在容器化環境中充分發揮其高效能儲存解決方案的潛力。

全快閃 **FAS** （**AFF**）系統 AFF 系統具有卓越的效能和低延遲，使其成為高需求應用程式的理想選擇。

All SAN Array (ASA) 系統 ASA 系統專為專用 SAN 環境而設計，提供強大的效能和可靠性。

ASA R2 Systems ASA R2 系統為 SAN 環境提供增強的功能和效能。

AFX NetApp AFX 是一種專為企業級 AI 工作負載而設計的解耦式全快閃儲存架構。它提供高效能 NAS，支援運算能力和容量的獨立擴充。NetApp AFX 儲存系統與其他基於 ONTAP 的系統（ASA、AFF 和 FAS）在儲存層的實作上有所不同。AFX 使用分散式檔案系統，從而實現高效能和可擴充性，而其他基於 ONTAP 的系統則採用傳統的儲存架構。

1. AFF 支援的協定：NFS、CIFS/SMB、iSCSI、FC、NVMe/TCP
2. ASA 支援的協定：iSCSI、FC、NVMe/TCP
3. ASA R2 支援的協定：iSCSI、FC、NVMe/TCP
4. AFX 支援的協定：從 Trident 25.10 開始，僅支援 NFS 協定；不支援 SMB 協定。

用於 **ONTAP** 儲存的 **Trident** 驅動程式

Trident 包含以下 CSI 驅動程式，每個驅動程式都能夠在後端儲存設備中佈建 Volume。

對於受支援硬體型號上的 **NAS** 通訊協定

1. ontap-nas：一個 PVC 對應至一個 PV，即一個專用 FlexVol Volume。
2. ontap-nas-economy：每個配置的 PV 都是一個 qtree，每個 FlexVol Volume 的 qtree 數量可配置（預設值為 200，可在 50 到 300 之間配置）。

   一個 PVC 映射到一個 PV，即共享的 FlexVol Volume 上的一個 qtree。

   > 您可以將所有虛擬機器的磁碟作為 qtree 放置在單一 Volume 中。但是，請注意，Trident 不支援 Snapshot 快照技術、Clone 及複寫功能。當這些功能由儲存管理員管理時，它們並非針對單一 PV 進行細粒度控制。

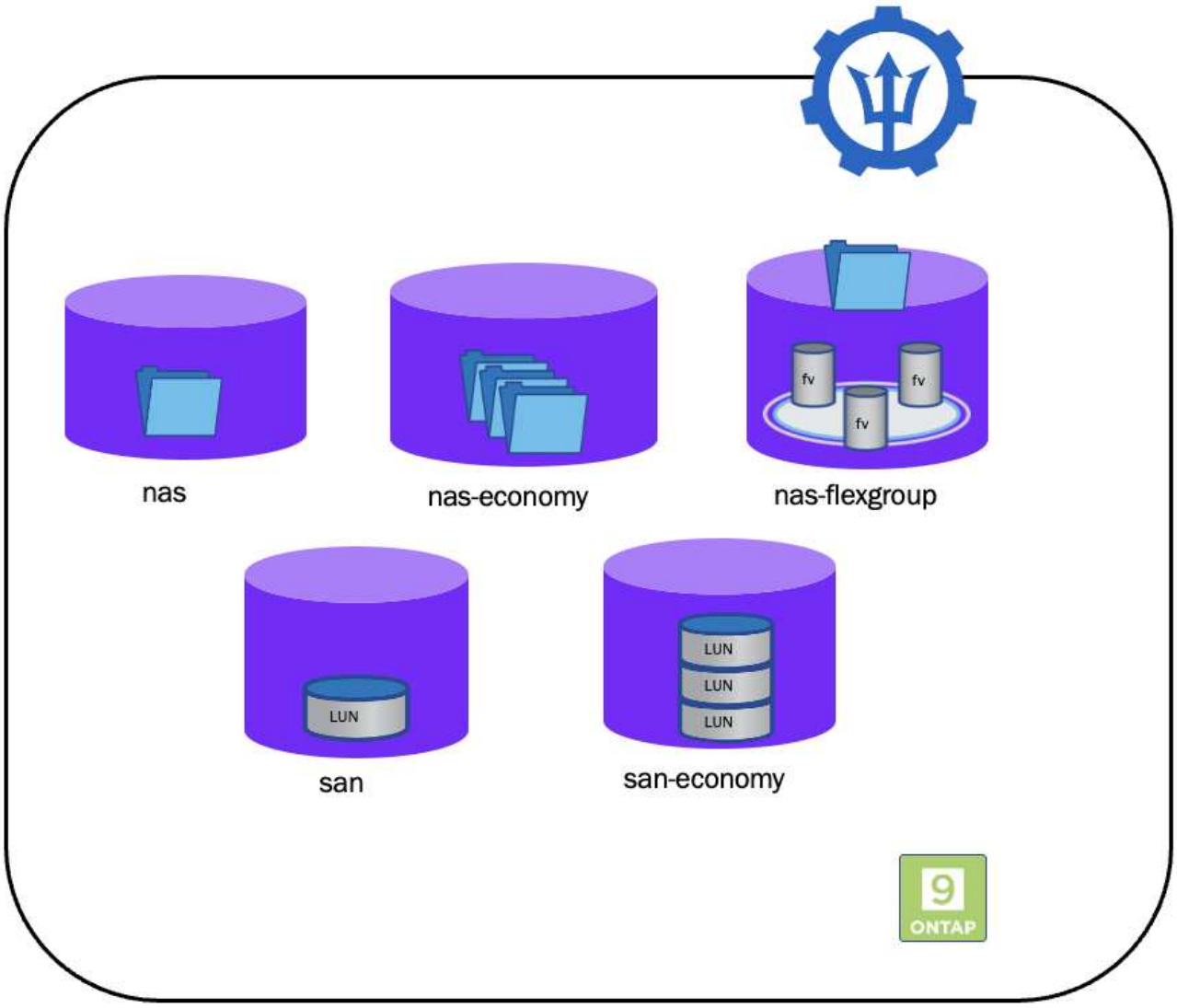
1. ontap-nas-flexgroup：每個配置的 PV 都是一個完整的 ONTAP FlexGroup，並且會使用分配給 SVM 的所有集合體。

對於受支援硬體型號上的 **SAN** 通訊協定

1. ontap-san：一個 PVC 對應到一個 PV，即專用 FlexVol Volume 上的一個 LUN。
2. ontap-san-economy：一個 PVC 對應到一個 PV，PV 是共享的 FlexVol Volume 上的一個 LUN。共享的 FlexVol Volume 中可以有多個 LUN（預設值為 100，每個 FlexVol Volume 可配置為 50 到 200 個 LUN/PV）。

   > 您可以將所有虛擬機器的磁碟作為 LUN 放置在單一 Volume 中。但是，此驅動程式支援 Snapshot 快照技術和 Clone，但不支援 Replication。當 Replication 由儲存管理員管理時，它不

是針對 PV 層級的。



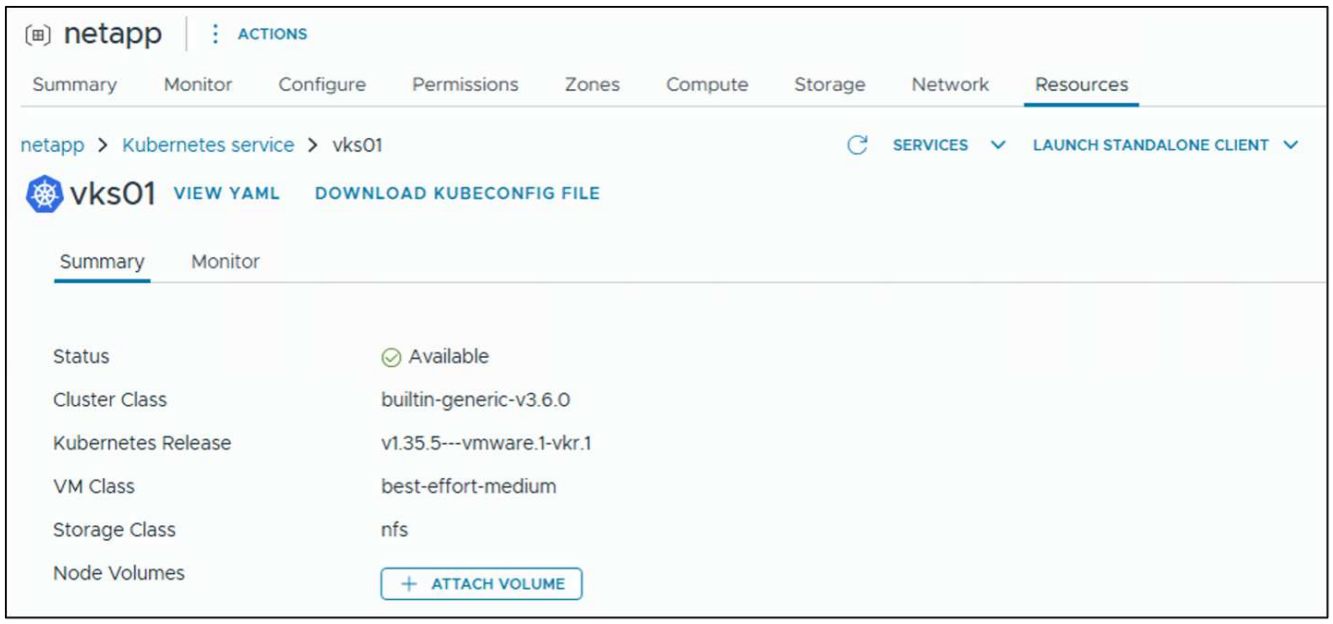
有關透過 Trident 驅動程式公開的功能以及必須由儲存管理員套用的功能的完整清單，請參閱 Trident 文件中的 "ONTAP 後端驅動程式" 章節。

在 **VKS** 叢集中安裝 **NetApp Trident**

在您的 vSphere Kubernetes Service 叢集中安裝 NetApp Trident 作為動態儲存資源配置程式，以將 NetApp ONTAP 儲存設備與您的 Kubernetes 工作負載整合。

開始之前

- 請確保您的 ONTAP 叢集已設定並連線至您的 VMware Kubernetes 環境。
- 如果您的工作負載將使用這些協定進行儲存設備，請確保您的 VKS 叢集已安裝 iSCSI 或 NVMe 工具。
- 請確保已在可以使用 kubeconfig 檔案連接到 VKS 叢集的機器上安裝了 `kubectl`。



步驟

1. 依照 Trident 文件中的說明，使用 Helm chart 安裝 Trident Operator：

   "https://docs.netapp.com/us-en/trident/trident-install/kubernetes-deploy-helm.html"

   在執行 Helm 安裝之前，請建立並標記 `trident` 命名空間。此 `enforce=privileged` 標籤是必要的，因為 Trident 執行的是特權工作負載。如果沒有此標籤，Kubernetes Pod 安全性許可控制器將封鎖 Trident Pod 啟動。

   ```
   kubectl create namespace trident
   kubectl label namespace trident pod-
   security.kubernetes.io/enforce=privileged
   ```

   ▼ 顯示範例

   ```
   helm install trident netapp-trident/trident-operator --version
   100.2606.0 --namespace trident
   ```

   若要啟用除錯記錄，請新增 `--set tridentDebug=true`：

   ```
   helm install trident netapp-trident/trident-operator --version
   100.2606.0 --namespace trident --set tridentDebug=true
   ```

   您也可以使用 Trident Operator 手動安裝 Trident。請參閱 Trident 文件中的步驟："手動部署 Trident Operator"

在手動安裝 Trident 之前，請確保已使用所需的 pod 安全性標籤來標記 `trident`命名空間。

2. 確認叢集中所有 Trident Pod 都正在運作。

▼ 顯示範例

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

```
vksbastion@vks:~/demo-vks$ kubectl get pods -n trident
NAME                                READY   STATUS    RESTARTS        AGE
trident-controller-67f7cd9bf8-4b289   6/6     Running   0               3d21h
trident-node-linux-46c86              2/2     Running   0               3d21h
trident-node-linux-bh29q              2/2     Running   0               11d
trident-node-linux-fcvjw              2/2     Running   1 (3d20h ago)   3d20h
trident-node-linux-gkhls              2/2     Running   0               3d20h
trident-operator-5cbf7f857-h5hx8      1/1     Running   0               3d20h
vksbastion@vks:~/demo-vks$
```

如果 Trident Pod 啟動失敗並出現 `PodSecurity`存取錯誤，則可能是安裝前未將 Pod 安全性標籤套用到 `trident`命名空間。請使用 `--overwrite`重新套用這些標籤，然後驗證 Pod 是否啟動。

```
kubectl label namespace trident pod-
security.kubernetes.io/enforce=privileged --overwrite
```

為您的環境準備儲存設備後端和 **StorageClass** 組態檔案

1. 透過定義必要的連線詳細資訊和儲存設備參數，為 ONTAP 設定 Trident 後端。
2. 在 Kubernetes 中定義對應至 NetApp 儲存設備後端的儲存設備類別。這些類別指定效能特性和複寫原則等參數。

   根據工作負載的要求，為下列協定定義 Trident 後端和儲存設備類別：

   - NAS（ontap-nas 驅動程式）
   - NAS Economy（ontap-nas-economy 驅動程式）
   - SAN（ontap-san 驅動程式）支援 iSCSI、FC 或 NVMe 協議
   - SAN Economy（ontap-san-economy 驅動程式）適用於 iSCSI

   網路儲存

   為 ONTAP NAS 建立 TridentBackendConfig 和 StorageClass，以啟用基於 NFS 的持久性儲存配置。後端組態包含儲存在 Kubernetes Secret 中的認證，並參照您的 ONTAP SVM 和管理 LIF。

   使用使用者名稱和密碼認證的後端金鑰和後端設定檔案範例（另存為 `tbc-nas.yaml`）：

   ```
   # tbc-nas.yaml
   apiVersion: v1
   kind: Secret
   ```

```
metadata:
  name: tbc-nas-secret
  namespace: trident
type: Opaque
data:
  username: "<base64-encoded cluster admin username>"
  password: "<base64-encoded cluster admin password>"
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: <ONTAP management LIF>
  backendName: tbc-nas
  svm: <your SVM name>
  storagePrefix: <your prefix for all volume names created using
this backend configuration>
  defaults:
    nameTemplate: "{{ .config.StoragePrefix }}_{{
.volume.Namespace }}_{{ .volume.RequestName }}"
  credentials:
    name: tbc-nas-secret
```

使用用戶端憑證認證的後端金鑰和後端設定檔案範例（另存為 `tbc-nas.yaml`）：

```
# tbc-nas.yaml
apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-secret
  namespace: trident
type: Opaque
stringData:
  clientPrivateKey: <client private key value>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-nas
  namespace: trident
spec:
```

```
  version: 1
  storageDriverName: ontap-nas
  managementLIF: <ONTAP management LIF>
  backendName: tbc-nas
  svm: <your SVM name>
  storagePrefix: <your prefix for all volume names created using
this backend configuration>
  credentials:
    name: ontap-nas-secret
  clientCertificate: <client certificate>
```

StorageClass 定義（另存為 `sc-nas.yaml`）：

`mountOptions` 是一個可選參數，用於指定 NFS Volume 的附加裝載選項。此 `nconnect` 選項允許為單一 NFS 裝載建立多個並行 TCP 連線，從而提升高吞吐量工作負載的效能。預設值為 1，但可以增加至 ONTAP 系統允許的最大值。

ONTAP 支援在支援 nconnect 的用戶端上，單一 NFS 裝載最多可建立 16 個連線。Trident 會將裝載選項直接傳遞給 Kubernetes 工作節點，因此請選擇工作節點 NFS 用戶端和 ONTAP 都支援的值；以下範例使用四個連線。

```
# sc-nas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
allowVolumeExpansion: true
mountOptions:
- nconnect=4
```

為 ONTAP NAS Economy 驅動程式建立 TridentBackendConfig 和 StorageClass，以使用 qtree 啟用基於 NFS 的持續儲存資源配置。後端組態包含儲存在 Kubernetes Secret 中的憑證，並參照您的 ONTAP SVM 和管理 LIF。

使用使用者名稱和密碼認證的後端金鑰和後端設定檔案範例（另存為 `tbc-nas-economy.yaml`）：

```
# tbc-nas-economy.yaml
apiVersion: v1
```

```
kind: Secret
metadata:
  name: tbc-nas-economy-secret
  namespace: trident
type: Opaque
data:
  username: "<base64-encoded cluster admin username>"
  password: "<base64-encoded cluster admin password>"
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-nas-economy
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: <ONTAP management LIF>
  backendName: tbc-nas-eco
  svm: <your SVM name>
  storagePrefix: <your prefix for all volume names created using this
backend configuration>
  defaults:
    nameTemplate: "{{ .config.StoragePrefix }}_{{ .volume.Namespace
}}_{{ .volume.RequestName }}"
  credentials:
    name: tbc-nas-economy-secret
```

StorageClass 定義（另存為 `sc-nas-economy.yaml`）：

```
# sc-nas-economy.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nas-economy
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas-economy"
  media: "ssd"
  provisioningType: "thin"
allowVolumeExpansion: true
mountOptions:
- nconnect=4
```

建立適用於 ONTAP SAN 與 iSCSI 的 TridentBackendConfig 和 StorageClass，以啟用基於 iSCSI 的區塊

儲存資源配置。後端組態使用 ontap-san 驅動程式，並包含儲存在 Kubernetes Secret 中的憑證。若省略 `sanType` 參數，則預設使用 iSCSI 通訊協定。

使用使用者名稱和密碼驗證的後端金鑰和後端組態檔案（另存為 `tbc-iscsi.yaml`）：

```
# tbc-iscsi.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-iscsi-secret
  namespace: trident
type: Opaque
data:
  username: "<base64-encoded cluster admin username>"
  password: "<base64-encoded cluster admin password>"
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: ontap-iscsi
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <ONTAP management LIF>
  backendName: ontap-iscsi
  svm: <your SVM name>
  credentials:
    name: backend-tbc-ontap-iscsi-secret
```

使用用戶端憑證認證的後端金鑰和後端設定檔案（另存為 `tbc-iscsi.yaml`）：

```
# tbc-iscsi.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-iscsi-secret
  namespace: trident
type: Opaque
stringData:
  clientPrivateKey: <client private key>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
```

```
  name: ontap-iscsi
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <ONTAP management LIF>
  backendName: ontap-iscsi
  svm: <your SVM name>
  credentials:
    name: backend-tbc-ontap-iscsi-secret
  clientCertificate: <client certificate>
```

StorageClass 定義（另存為 `sc-iscsi.yaml`）：

```
# sc-iscsi.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-iscsi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  media: "ssd"
  provisioningType: "thin"
  fsType: ext4
  snapshots: "true"
allowVolumeExpansion: true
```

為 ONTAP SAN 搭配 NVMe 建立 TridentBackendConfig 和 StorageClass，以啟用基於 NVMe 的區塊儲存資源配置。後端組態使用 ontap-san 驅動程式，並包含儲存在 Kubernetes Secret 中的憑證。 `sanType` 參數必須設定為 `nvme`。

使用使用者名稱和密碼驗證的後端金鑰和後端組態檔案（另存為 `tbc-nvme.yaml`）：

```
# tbc-nvme.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-nvme-secret
  namespace: trident
type: Opaque
data:
  username: "<base64-encoded cluster admin username>"
  password: "<base64-encoded cluster admin password>"
```

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: ontap-nvme
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  sanType: nvme
  managementLIF: <ONTAP management LIF>
  backendName: ontap-nvme
  svm: <your SVM name>
  credentials:
    name: backend-tbc-ontap-nvme-secret
```

使用用戶端憑證認證的後端金鑰和後端設定檔案（另存為 `tbc-nvme.yaml`）：

```
# tbc-nvme.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-nvme-secret
  namespace: trident
type: Opaque
stringData:
  clientPrivateKey: <client private key>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: ontap-nvme
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  sanType: nvme
  managementLIF: <ONTAP management LIF>
  backendName: ontap-nvme
  svm: <your SVM name>
  credentials:
    name: backend-tbc-ontap-nvme-secret
  clientCertificate: <client certificate>
```

StorageClass 定義（另存為 `sc-nvme.yaml`）：

```
# sc-nvme.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nvme
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  media: "ssd"
  provisioningType: "thin"
  fsType: ext4
  snapshots: "true"
allowVolumeExpansion: true
```

請參閱 Trident 文件，以取得更多 YAML 檔案範例，以及 SAN 驅動程式和 NAS 驅動程式支援的存取模式和 Volume 模式的相關資訊。

- "使用 NAS 驅動程式的範例"
- "使用 SAN 驅動程式的範例"

建立 **VolumeSnapshotClass** 組態檔案

建立 VolumeSnapshotClass 定義。此組態可為持續性 Volume 啟用快照型作業。

VolumeSnapshotClass 定義（另存為 `snapshot-class.yaml`）：

```
# snapshot-class.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: trident-snapshotclass
driver: csi.trident.netapp.io
deletionPolicy: Retain
```

將組態檔套用至叢集

使用 `kubectl`將您在前面步驟中建立的組態檔套用至 vSphere Kubernetes 服務叢集。這將在您的叢集中建立必要的機密、後端組態、儲存設備類別和快照類別。

步驟

1. 為您配置的每個協定套用 TridentBackendConfig 和 StorageClass 檔案。

```
kubectl apply -f tbc-nas.yaml -n trident
kubectl apply -f sc-nas.yaml

kubectl apply -f tbc-nas-economy.yaml -n trident
kubectl apply -f sc-nas-economy.yaml

kubectl apply -f tbc-iscsi.yaml -n trident
kubectl apply -f sc-iscsi.yaml

kubectl apply -f tbc-nvme.yaml -n trident
kubectl apply -f sc-nvme.yaml

kubectl apply -f snapshot-class.yaml
```

2. 確認資源已成功建立。

   檢查 TridentBackendConfig 物件：

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

   檢查 StorageClass 物件：

   ```
   kubectl get storageclass
   ```

   檢查 VolumeSnapshotClass：

   ```
   kubectl get volumesnapshotclass
   ```

設定預設的 **Trident** 儲存設備和快照類別

將 Trident StorageClass 和 VolumeSnapshotClass 設為 vSphere Kubernetes Service 叢集中的預設值。

步驟

1. 設定預設的 Trident StorageClass。

   將 Trident 支援的 StorageClass 設定為叢集預設值，以便在未指定儲存類別時，PersistentVolumeClaims 自動使用它。

   確保只將一個 StorageClass 設定為預設值。如果另一個 StorageClass 已設定為預設值，請將其註解設為 `false`。

   從 CLI：

```
kubectl patch storageclass <storage class name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

2. 設定預設的 Trident VolumeSnapshotClass。

   將 Trident 支援的 VolumeSnapshotClass 設定為叢集預設值，以啟用持續性 Volume 的快照型作業。這可確保 VolumeSnapshots 在未指定快照類別時，自動使用 Trident CSI 驅動程式。

   確保只將一個 VolumeSnapshotClass 設定為預設值。如果另一個 VolumeSnapshotClass 已設定為預設值，請將其註解設為 `false`。

   從 CLI：

```
kubectl patch volumesnapshotclass <snapshot class name> --type=merge -p
'{"metadata":{"annotations":{"snapshot.storage.kubernetes.io/is-default-
class":"true"}}}'
```

使用 **Kubernetes Persistent Volume Claims** 為工作負載請求儲存設備

Trident 會自動從後端 NetApp 儲存設備配置必要的 Volume。有關在 VKS 叢集中使用 PVC 部署工作負載的範例，請參閱 "使用持續儲存部署工作負載"。

部署具有 **NetApp** 持續儲存的 **Container** 應用程式

使用 NetApp ONTAP 持續儲存，在您的 vSphere Kubernetes Service 叢集中部署 Container 應用程式。以下範例說明如何使用 NAS （ontap-nas）或 SAN iSCSI （ontap-san）儲存設備來部署 PostgreSQL 資料庫。

以下 YAML 組態直接在清單中設定 POSTGRES_USER / POSTGRES_PASSWORD。在正式作業環境中，建議使用 Kubernetes Secrets 來管理資料庫憑證等敏感資訊。

使用 **NAS** 儲存設備部署 **PostgreSQL**（**ontap-nas** 驅動程式）

您可以部署由 ontap-nas 驅動程式佈建的 NFS 持續性 Volume 支援的 PostgreSQL Container 應用程式。以下範例示範如何為 PostgreSQL 建立 PersistentVolumeClaim （PVC） 和部署。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-pvc
spec:
  storageClassName: sc-nas # Specify the StorageClass for dynamic
provisioning
  accessModes:
    - ReadWriteOnce
```

```
  resources:
    requests:
      storage: 10Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      securityContext:
        fsGroup: 999  # Ensures proper permissions for the mounted
volume (PostgreSQL default user is 999)
      containers:
        - name: postgres
          image: postgres:15
          securityContext:
            allowPrivilegeEscalation: false
            runAsNonRoot: true
            runAsUser: 999  # PostgreSQL default user ID
            capabilities:
              drop:
                - ALL
            seccompProfile:
              type: RuntimeDefault
          env:
            - name: POSTGRES_DB
              value: "postgres"
            - name: POSTGRES_USER
              value: "<username>"
            - name: POSTGRES_PASSWORD
              value: "<password>"
            - name: PGDATA
              value: "/var/lib/postgresql/data/pgdata"
          ports:
            - containerPort: 5432
          volumeMounts:
            - mountPath: /var/lib/postgresql/data
```

```
              name: postgres-storage
              subPath: postgres-data  # Ensures data is stored in a
subdirectory to avoid permission issues
          resources:
            requests:
              memory: "512Mi"
              cpu: "250m"
            limits:
              memory: "1Gi"
              cpu: "500m"
      volumes:
        - name: postgres-storage
          persistentVolumeClaim:
            claimName: postgres-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
spec:
  type: ClusterIP
  ports:
    - port: 5432
      targetPort: 5432
  selector:
    app: postgres
```

使用 **SAN** 儲存設備部署 **PostgreSQL**（**ontap-san iSCSI** 驅動程式）

使用下列 YAML 部署由 ontap-san 驅動程式佈建的 iSCSI 持續性 Volume 所支援的 PostgreSQL Container 應用程式。此 `Recreate`部署策略確保在啟動新 Pod 之前，舊 Pod 已完全終止，這對於 ReadWriteOnce iSCSI Volume 而言是必要的，並可防止 Pod 重新啟動時發生資料毀損。此組態支援 Pod 刪除和重建後的資料持續性，並且與 Trident Volume 快照及備份與恢復作業相容。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-block-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: sc-iscsi # Must map to an ontap-san driver
---
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres-block-app
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      # 1. POD LEVEL SECURITY CONTEXT
      securityContext:
        runAsNonRoot: true
        runAsUser: 999       # Official Postgres image default user ID
        runAsGroup: 999      # Official Postgres image default group ID
        fsGroup: 999
        seccompProfile:
          type: RuntimeDefault
      containers:
        - name: postgres
          image: postgres:15
          # 2. CONTAINER LEVEL SECURITY CONTEXT
          securityContext:
            allowPrivilegeEscalation: false
            capabilities:
              drop:
                - ALL
          env:
            - name: POSTGRES_DB
              value: "postgres"
            - name: POSTGRES_USER
              value: "<username>"
            - name: POSTGRES_PASSWORD
              value: "<password>"
            - name: PGDATA
              value: /var/lib/postgresql/data/pgdata
          ports:
            - containerPort: 5432
              name: postgres
          volumeMounts:
```

```
            - mountPath: /var/lib/postgresql/data
              name: postgres-block-storage
              subPath: postgres-data
          resources:
            requests:
              memory: "512Mi"
              cpu: "250m"
            limits:
              memory: "1Gi"
              cpu: "500m"
      volumes:
        - name: postgres-block-storage
          persistentVolumeClaim:
            claimName: postgres-block-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
spec:
  type: ClusterIP
  ports:
    - port: 5432
      targetPort: 5432
  selector:
    app: postgres
```

驗證資料庫部署

應用程式部署完成後，請驗證 Pod 是否處於運作狀態以及資料庫是否正常運作。使用以下命令檢查 Pod、PVC 和服務的狀態。確保 Pod 正在執行，且 PVC 已繫結至適當的 Volume。

```
kubectl get all -n <namespace>
kubectl get pvc -n <namespace>
```

```
vksbastion@vks:~$ kubectl get all -n postgres
NAME                                    READY   STATUS    RESTARTS   AGE
pod/postgres-block-app-7c9859c558-bxhbg   1/1     Running   0          20h

NAME                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.100.154.170   <none>        5432/TCP   25h

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres-block-app   1/1     1            1           25h

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-block-app-7c9859c558   1         1         1       25h
vksbastion@vks:~$ kubectl get pvc -n postgres
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
postgres-block-pvc   Bound    pvc-b992a5cf-1127-4b6c-a775-80ab64e4b536   20Gi       RWO            sc-iscsi       <unset>                 25h
vksbastion@vks:~$
```

根據所用協定的不同，後端儲存設備可以是 Volume、qtree 或 LUN。您可以透過描述 PersistentVolume （PV）來擷取內部 Volume 名稱，然後在 ONTAP CLI 命令中使用該名稱來取得儲存設備詳細資訊，藉此驗證 ONTAP 系統中已佈建的儲存設備。使用以下命令描述 PV 並擷取內部 Volume 名稱：

```
kubectl describe pv/<pv-name>
```

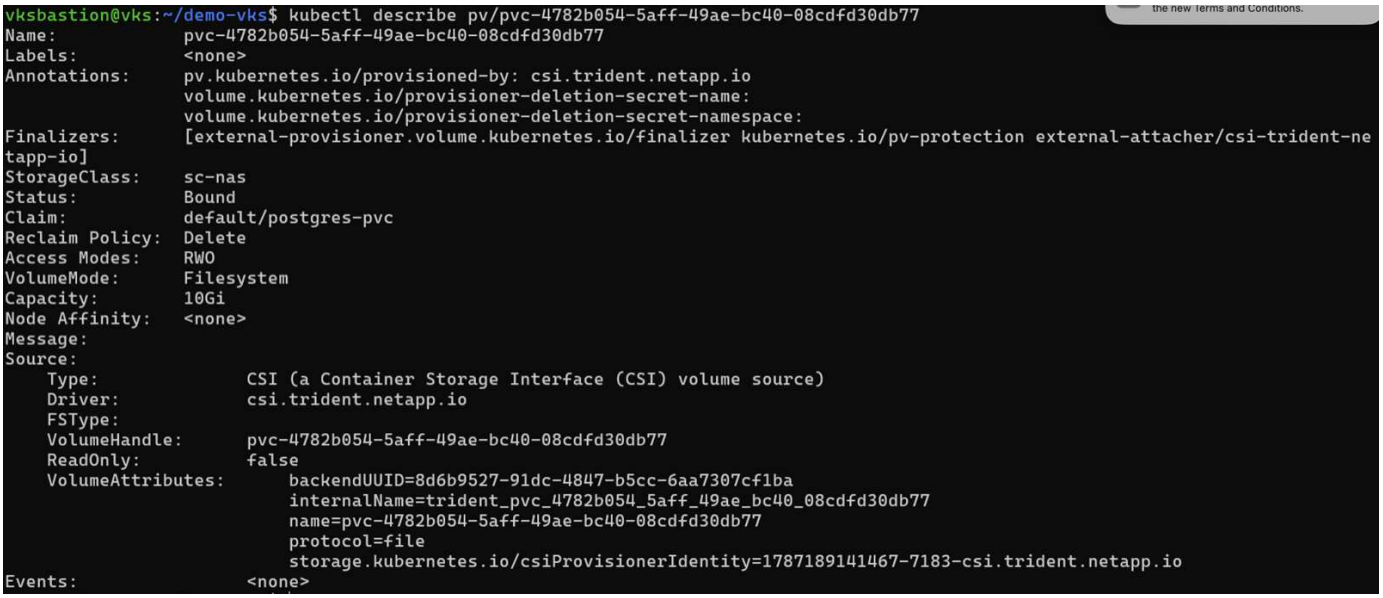
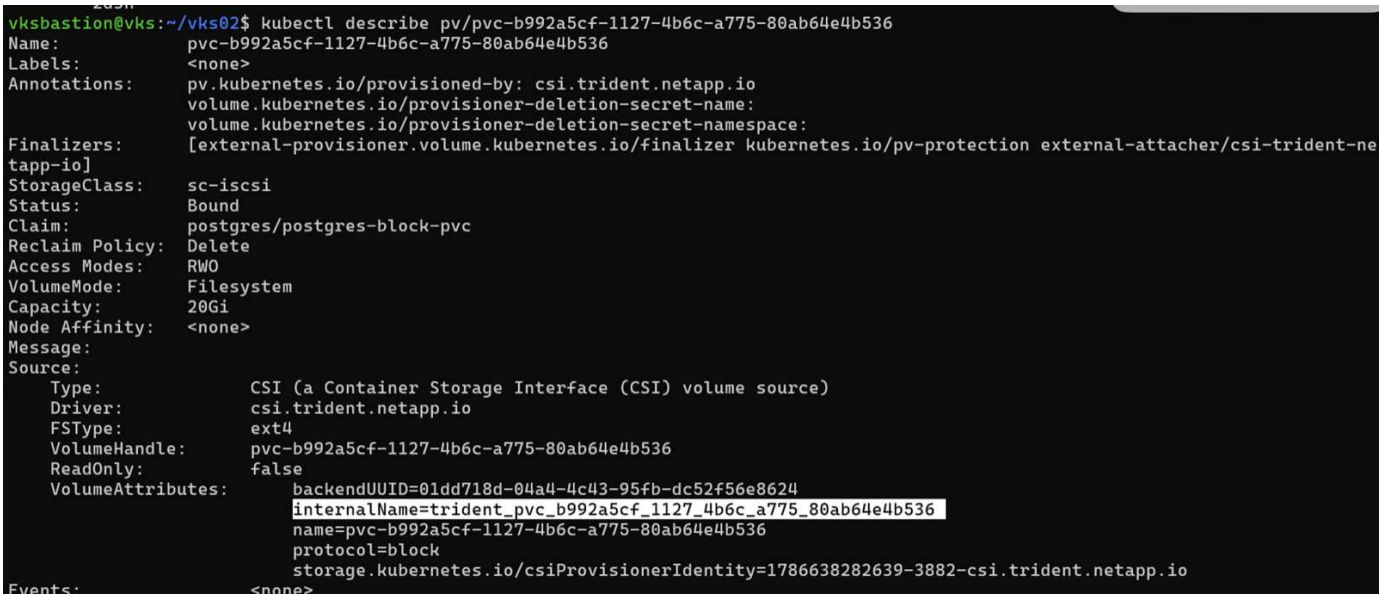


圖 **1.** 顯示範例



從輸出複製內部 Volume 名稱，然後執行以下命令，從 ONTAP CLI 取得儲存設備詳細資訊。

對於 NAS Volume：

```
volume show -vserver <vserver> -volume <internal-volume-name>
```

```
NSOL-NetApp-A70-T19U05::> volume show -vserver vks -volume trident_pvc_4782b054_5aff_49ae_bc40_08cdfd30db77

                                    Vserver Name: vks
                                     Volume Name: trident_pvc_4782b054_5aff_49ae_bc40_08cdfd30db77
                                  Aggregate Name: NSOL_NetApp_A70_T19U05a_NVME_SSD_1
    List of Aggregates for FlexGroup Constituents: NSOL_NetApp_A70_T19U05a_
                                                   NVME_SSD_1
                                Encryption Type: volume
               List of Nodes Hosting the Volume: NSOL-NetApp-A70-T19U05a
                                    Volume Size: 10GB
                             Volume Data Set ID: 2179
                      Volume Master Data Set ID: 2155359667
                                    Volume State: online
                                    Volume Style: flex
                           Extended Volume Style: flexvol
                         FlexCache Endpoint Type: none
                          Is Cluster-Mode Volume: true
                           Is Constituent Volume: false
                   Number of Constituent Volumes: -
                                  Export Policy: default
                                        User ID: 0
                                       Group ID: 0
                                 Security Style: unix
                                UNIX Permissions: ---rwxrwxrwx
                                   Junction Path: /trident_pvc_4782b054_5aff_49ae_bc40_08cdfd30db77
                            Junction Path Source: RW_volume
```

對於 SAN LUN，請執行以下命令從 ONTAP CLI 取得 LUN 詳細資訊：

```
lun show -vserver <vserver> -volume <internal-volume-name>
```

```
NSOL-NetApp-A70-T19U05::> lun show -vserver vks -volume trident_pvc_b992a5cf_1127_4b
6c_a775_80ab64e4b536
Vserver   Path                            State   Mapped   Type        Size
--------- ------------------------------ ------- -------- -------- --------
vks       /vol/trident_pvc_b992a5cf_1127_4b6c_a775_80ab64e4b536/lun0
                                          online  mapped   linux        20GB

NSOL-NetApp-A70-T19U05::>
```

如果您在 NAS 或 NAS Economy 儲存設備類別中使用了 `nconnect`裝載選項，則可以驗證從工作節點到儲存設備伺服器的作用中 TCP 連線數。

首先，列出 Trident 後端組態以尋找後端名稱，然後對其進行描述以擷取 vserver 名稱和資料 LIF：

```
kubectl get tbc -n trident
kubectl describe tbc <backend-name> -n trident
```

然後登入 ONTAP 叢集並執行以下命令，將上述輸出中的資料 LIF 替換為對應的 LIF：

```
network connections active show -local-address <data-lif> -local-port 2049
```

```
NSOL-NetApp-A70-T19U05::> network connections active show -local-address 192.168.164.214 -local-port 2049
Vserver     Interface              Remote
Name        Name:Local Port        Host:Port                    Protocol/Service
----------- ---------------------- ---------------------------- ----------------
Node: NSOL-NetApp-A70-T19U05b
vks         lif_VKS_4608:2049      192.168.178.41:821           TCP/nfs
vks         lif_VKS_4608:2049      192.168.178.54:843           TCP/nfs
vks         lif_VKS_4608:2049      192.168.178.54:745           TCP/nfs
vks         lif_VKS_4608:2049      192.168.178.54:866           TCP/nfs
vks         lif_VKS_4608:2049      192.168.178.54:759           TCP/nfs
5 entries were displayed.
```

圖 **2.** 顯示範例

Pod 處於運作狀態後，連線至資料庫並驗證資料操作。

步驟

1. 將 PostgreSQL 服務連接埠轉送到本機：

```
kubectl port-forward svc/postgres-service 5432:5432 -n <namespace>
```

2. 在另一個終端機中，使用 psql 連接到資料庫：

```
psql "postgresql://<username>:<password>@127.0.0.1:5432/postgres"
```

3. 建立資料庫和資料表，然後將資料填入該資料表：

```
CREATE DATABASE employee;
```

切換到新資料庫（psql 元命令）：

```
\c employee
```

建立表格、插入一列資料，並查詢結果：

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    department VARCHAR(50),
    hire_date DATE DEFAULT CURRENT_DATE,
    salary NUMERIC(10, 2)
);

INSERT INTO employees (first_name, last_name, email, department, salary)
VALUES ('John', 'Doe', 'john.doe@example.com', 'Engineering', 85000.00);
```

```
SELECT * FROM employees;
```

影片示範

以下影片示範如何在 vSphere Kubernetes 叢集上安裝 Trident，以及如何使用 Trident 部署具有持續儲存的 PostgreSQL 資料庫。

使用 Trident 在由 ONTAP 持續儲存支援的 vSphere Kubernetes Service 叢集上部署工作負載

資料保護

使用 **NetApp Console** 備份與恢復 **vSphere Kubernetes Service** 叢集中的 **Container** 應用程式

使用 NetApp Console 備份與恢復 vSphere Kubernetes Service (VKS) 叢集中的 Container 應用程式。本程序涵蓋透過 NetApp Console 的 Backup and Recovery 服務建立與管理備份與恢復作業，並使用 ONTAP S3 物件儲存作為備份目標。

NetApp Console 提供對本地和雲端環境中儲存設備和資料服務的集中管理。它提供統一的控制、簡易的操作和安全的管理。使用者可以透過使用者介面存取多種資料服務和管理功能，網址為 "console.netapp.com"。有關 NetApp Console 的完整詳細資訊，請參閱 "NetApp Console 文件"。

本節重點介紹 NetApp Backup and Recovery 針對 Kubernetes 工作負載的資料服務，特別是 vSphere Kubernetes Service 叢集上的 Container 應用程式。NetApp Backup and Recovery Service 可讓您透過單一介面探索、管理、建立保護原則，並將其與 Kubernetes 應用程式建立關聯。如需完整指南，請參閱 "NetApp Backup and Recovery 文件"。

備份與恢復工作流程

1. 請確保您已安裝 **Console** 代理程式

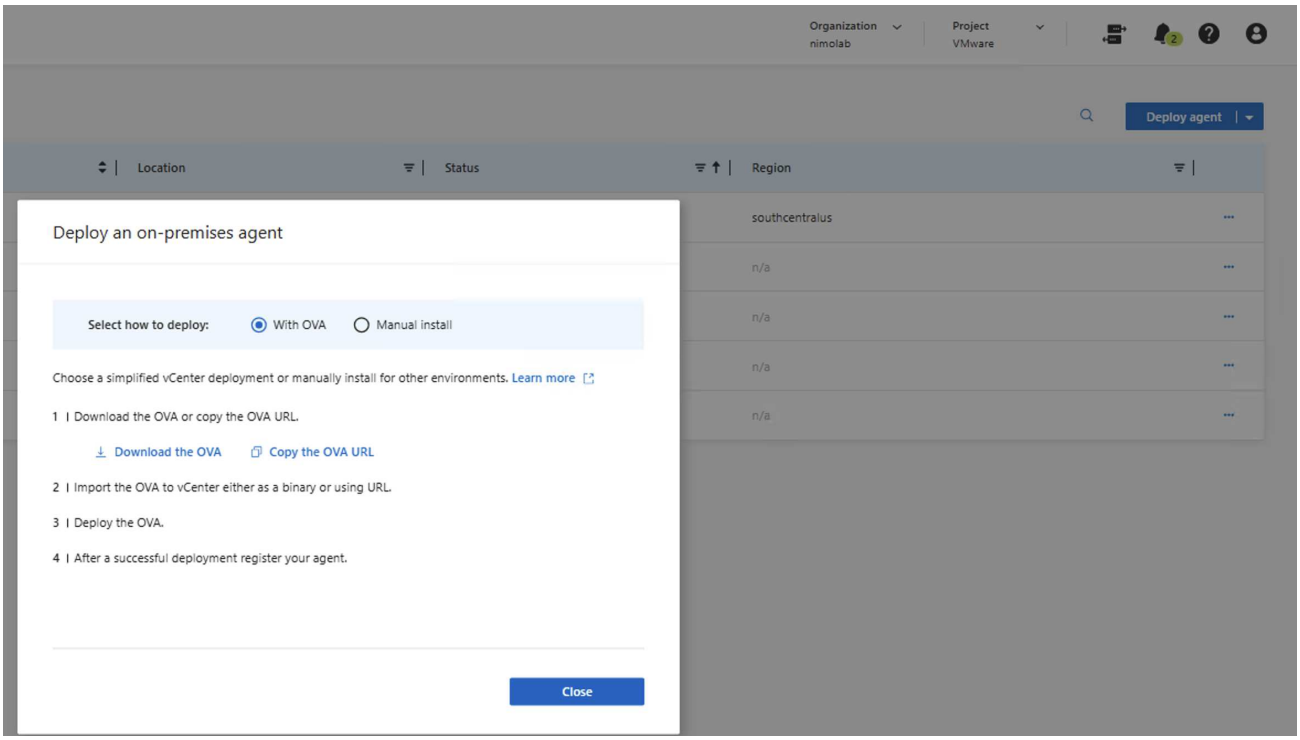
   請確保在 VKS 叢集執行的環境中已部署 Console 代理程式。有關安裝 Console 代理程式的詳細資訊，請參閱 "NetApp Console Setup 文件"。

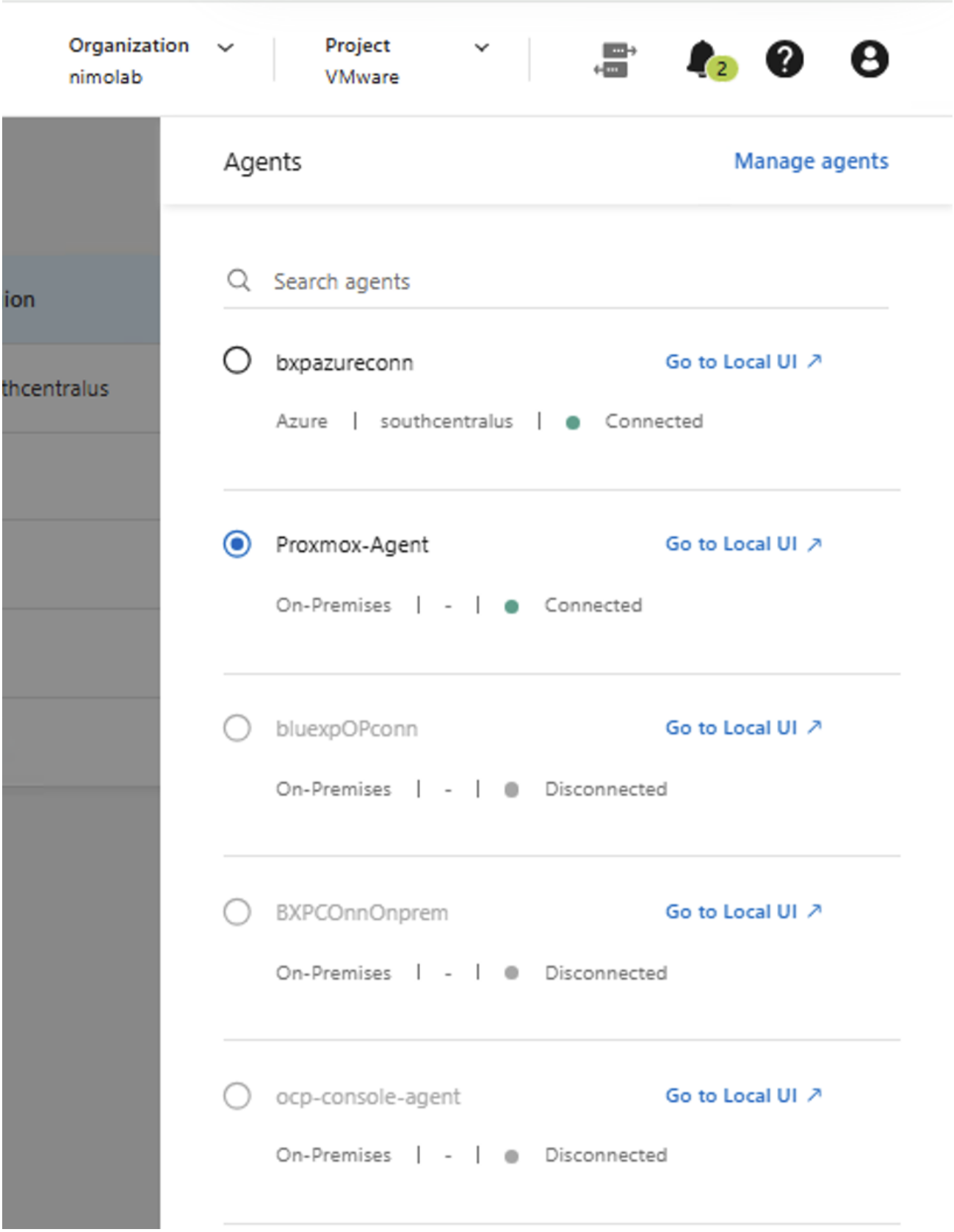
   ▼ 顯示範例

   在 NetApp Console 中，按一下快照所示的「Deploy Agent」，並將代理程式部署到 VKS 叢集執行的環境中。

   在本例中，選擇「On-Premises > With OVA」，複製 OVA URL，並在 vCenter 中使用該 URL 部署 Console 代理程式。

   使用代理程式的 IP 位址在 URL 中進行註冊。如需詳細指示，請參閱 "文件"。代理程式註冊完成後，您可以在 NetApp Console 中看到它，如下快照所示。


Organization
nimolab
Project
VMware
Deploy agent
Location
Status
Region
southcentralus
n/a
n/a
n/a
n/a
Deploy an on-premises agent
Select how to deploy:
With OVA
Manual install
Choose a simplified vCenter deployment or manually install for other environments. Learn more
1 I Download the OVA or copy the OVA URL.
Download the OVA
Copy the OVA URL
2 I Import the OVA to vCenter either as a binary or using URL.
3 I Deploy the OVA.
4 I After a successful deployment register your agent.
Close



2. 將 **ONTAP** 叢集新增至 **NetApp Console** 並啟用 **Backup and Recovery**

   必須先將主要 ONTAP 叢集新增至 NetApp Console，並在其上啟用 Backup and Recovery 服務，才能保護工作負載。如需相關指示，請參閱 "設定 NetApp Backup and Recovery"。

3. 在 **NetApp Console** 中，探索 **vSphere Kubernetes** 叢集

▼ 顯示範例

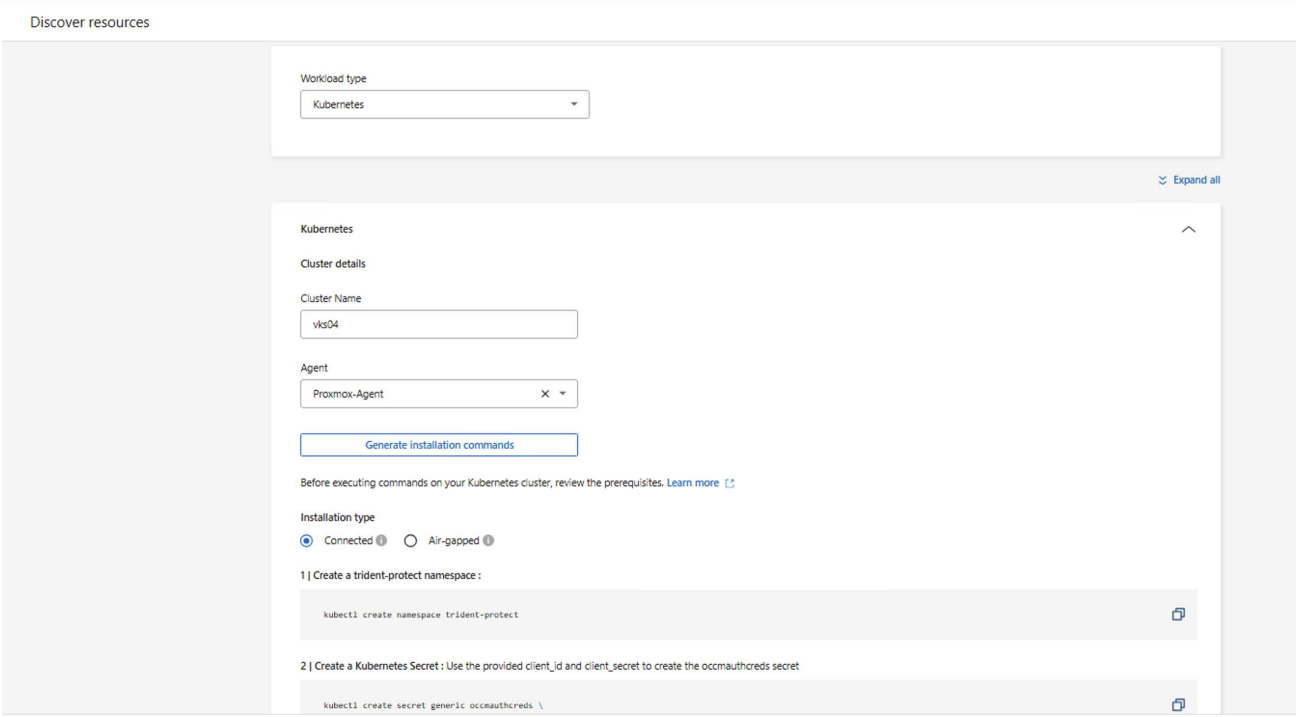
此步驟需要 vSphere Kubernetes 叢集正在執行，且 Console 代理程式已部署在相同的環境中。請遵循以下步驟，在 NetApp Console 中探索 vSphere Kubernetes 叢集。

a. 在 NetApp Console 中，選取 **Inventory > Kubernetes**，並提供 vSphere Kubernetes 叢集的詳細資訊。

b. 選擇部署在與 vSphere Kubernetes 叢集相同環境中的代理程式。

c. 複製提供的命令來安裝 Trident Protect，並從連接到 vSphere Kubernetes 叢集的機器上執行這些命令。

d. Trident Protect 安裝成功後，按一下 **Discover**。NetApp Console 透過代理程式探索 vSphere Kubernetes 叢集及其所有資源，並在使用者介面中填入這些資源。



1 | Create a trident-protect namespace :

```
kubectl create namespace trident-protect
```

2 | Create a Kubernetes Secret : Use the provided client_id and client_secret to create the occmauthcreds secret

```
kubectl create secret generic occmauthcreds \
--namespace=trident-protect \
--from-literal=client_id=PZBjqkcEVZeeg-fJECfCRm42rO1EOKrq \
--from-literal=client_secret=5-XbW8MoCnQKb7ImJSIdTC0pef5W4klpmTqeLTeDU5s8ArZWYJuup7OuNdgyBJbz
```

3 | Add/update helm repo :

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

4 | Install or Upgrade Trident Protect and Trident Protect Connector

```
helm upgrade --install trident-protect \
netapp-trident-protect/trident-protect-console \
--version 100.2608.0-console \
--namespace trident-protect \
--set clusterName=vks04 \
--set clusterID=966e65b1-c36a-441b-b23b-99db28c33151 \
--set trident-protect.cbs.accountID=account-B6oEAubL \
--set trident-protect.cbs.agentID=kgl3b3JQQR7e0NxidFg1wH3XockAb8p7clients \
--set trident-protect.cbs.proxySecretName=occmauthcreds \
--set trident-protect.cbs.proxyHostIP=https://10.192.113.198
```

```
vksbastion@vks:~/vks04$ kubectl get pods -n trident-protect
NAME                                                READY   STATUS    RESTARTS   AGE
trident-protect-connector-86bf87f7d7-76llf          1/1     Running   0          29s
trident-protect-controller-manager-8567cf76b9-lxxqz   1/1     Running   0          29s
vksbastion@vks:~/vks04$
```

Applications | Clusters | Namespaces | Virtual machines

Clusters (4)

Name	Kubernetes Version	Kubernetes distribution	Trident protect version	Applications	Namespaces
demo-vks Pending				0 Applications (0 out of 0 protected)	0
vks-demo01 Pending				0 Applications (0 out of 0 protected)	0
vks03-old Deleting		kubernetes	26.08.0-console	0 Applications (0 out of 0 protected)	17
vks04 Connected		kubernetes	26.08.0-console	0 Applications (0 out of 0 protected)	16

4. 將 **ONTAP S3** 設定為備份目標

   在本範例中，我們使用 ONTAP S3 物件儲存作為備份目標。您也可以使用 AWS S3、Azure Blob、Google Cloud Storage 或 StorageGRID 作為備份目標。

   如需更多詳細資訊，請審查 "NetApp Console 文件"。

   若要在 NetApp Console 中新增 ONTAP S3 作為備份目標，您需要從 ONTAP 叢集取得下列資訊。

   ```
   S3 Endpoint:
   ```

```
Access Key:
Secret Key:
Bucket Name:
```

若要取得這些值，請在 ONTAP 叢集中使用 System Manager 執行下列操作。

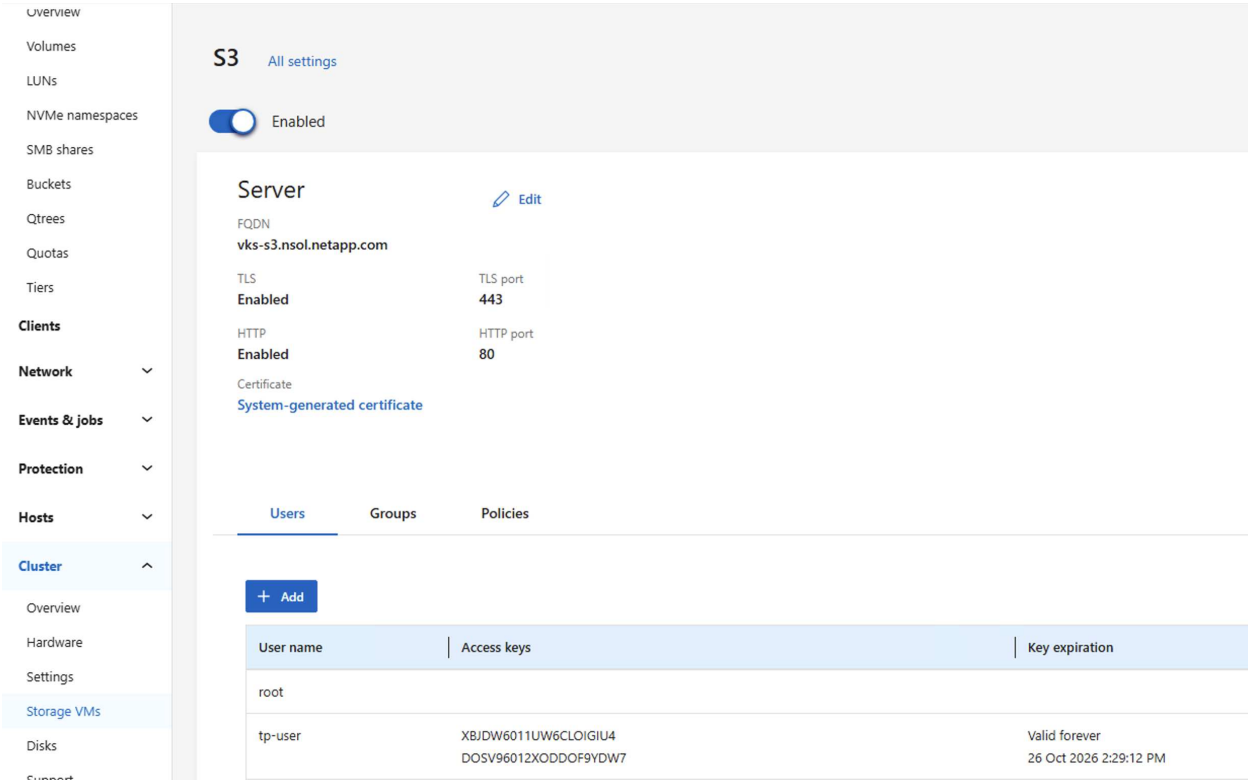
a. 在 ONTAP 叢集中建立一個啟用 S3 的 SVM 來儲存備份資料。記下此 SVM 的 S3 端點 URL。

b. 在 SVM 的 S3 設定中，建立使用者並指派所需的存取權限。記下此使用者的存取金鑰和私密金鑰。

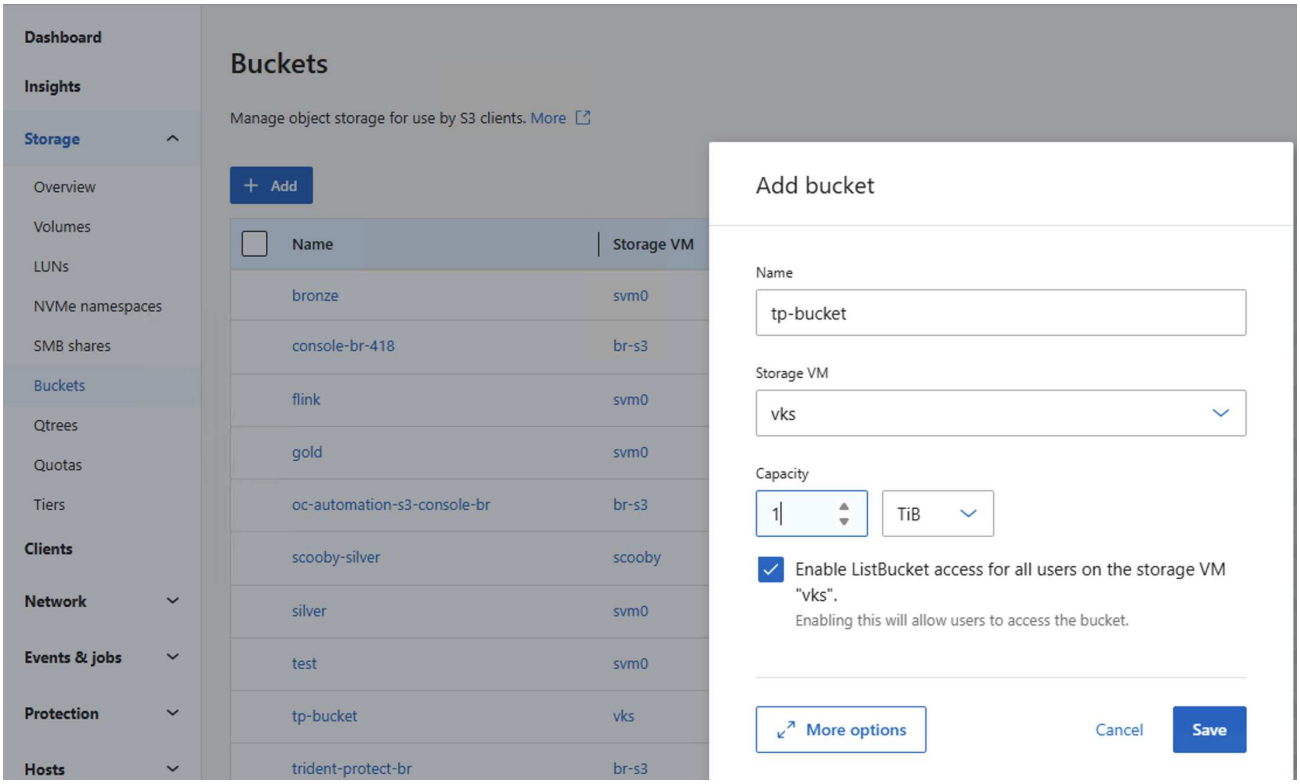
> Trident Protect 要求 S3 使用者至少擁有 `PutObject`、`GetObject`、`ListBucket` 和 `DeleteObject` 權限。如果沒有這些權限，AppVault 備份與恢復作業將失敗並出現存取被拒絕的錯誤。如需詳細資訊，請參閱"Trident Protect AppVault 文件"。

▼ 顯示範例

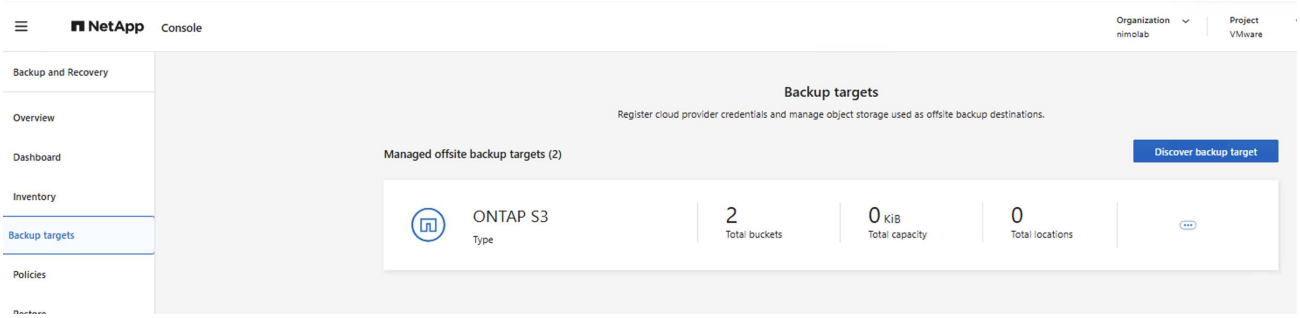
i. 建立 S3 使用者並下載存取金鑰和私密金鑰

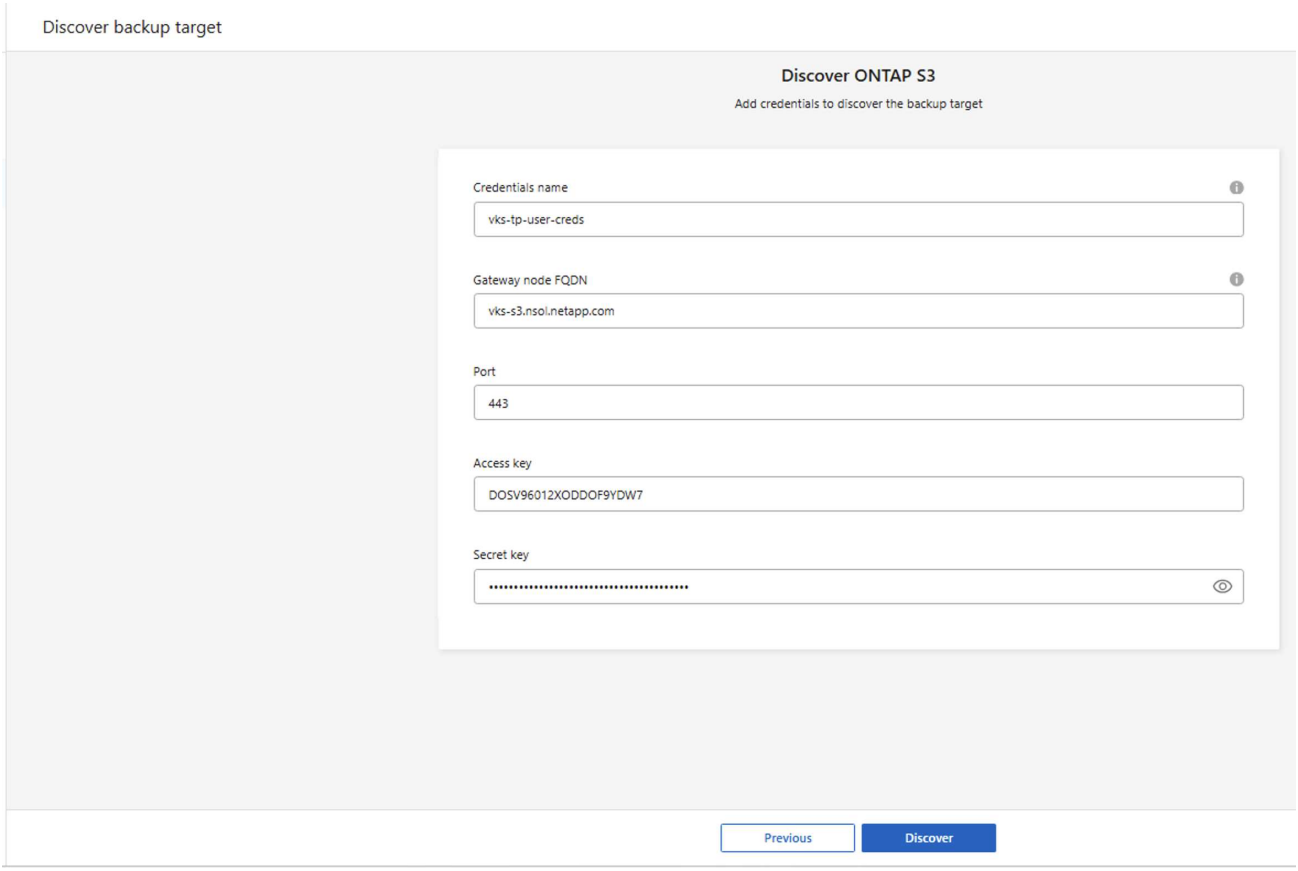


A. 在啟用 S3 的 SVM 中建立一個儲存桶。記下儲存桶名稱，以便在 NetApp Console 中將 ONTAP S3 物件儲存新增為備份目標時使用。



使用 S3 端點、存取金鑰、私密金鑰和儲存桶名稱，在 NetApp Console 中新增 ONTAP S3 物件儲存作為備份目標。





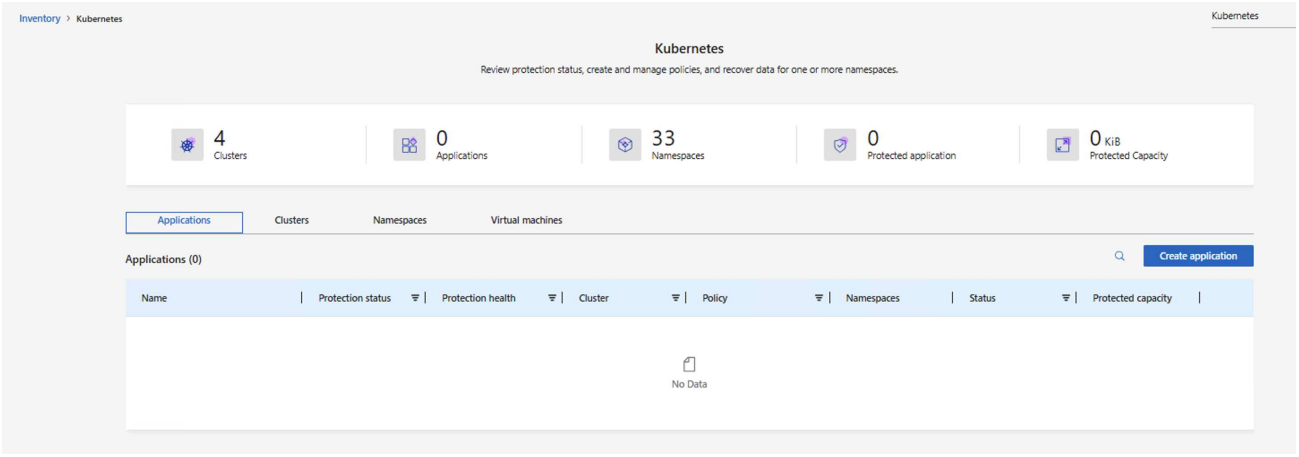
5. 在 **NetApp Console** 中，為 **Container** 工作負載建立應用程式並使用備份加以保護

> 您需要擁有 Backup and Recovery 超級管理員或 Backup and Recovery 備份管理員角色，才能建立和保護 Kubernetes 應用程式。若要還原 Kubernetes 應用程式，您需要擁有 Backup and Recovery 超級管理員或 Backup and Recovery 還原管理員角色。

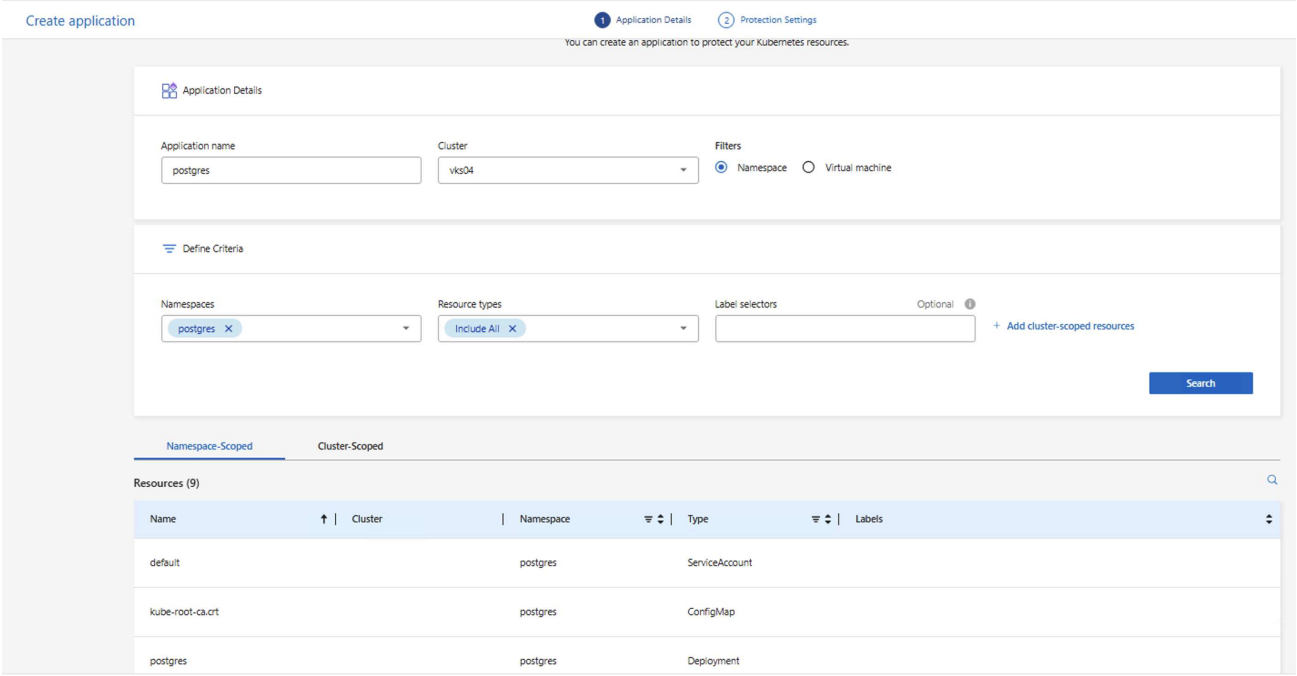
▼ 顯示範例

在此步驟中，請在 NetApp Console 中為 vSphere Kubernetes 叢集中需要保護的 Container 應用程式建立應用程式。您可以使用命名空間來建立應用程式，以保護 vSphere Kubernetes 叢集中某個命名空間內的所有 Container 應用程式。

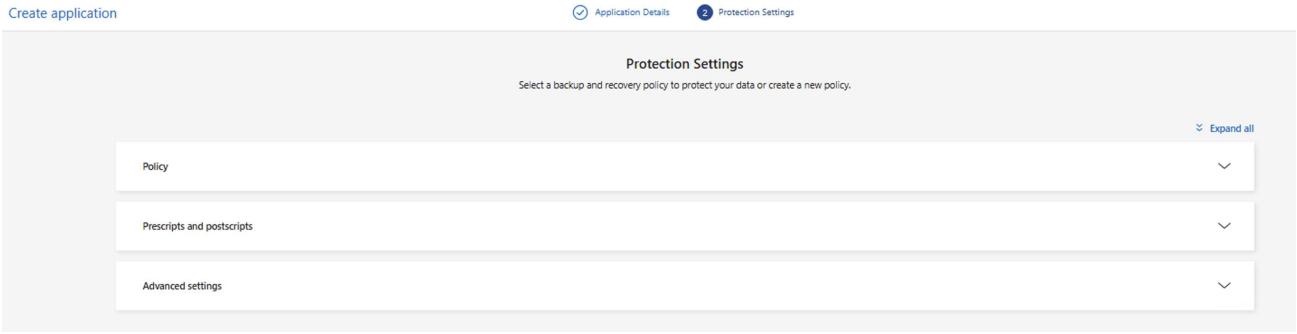
您還需要一個與應用程式關聯的保護原則。您可以建立新的保護原則，也可以使用現有的保護原則。



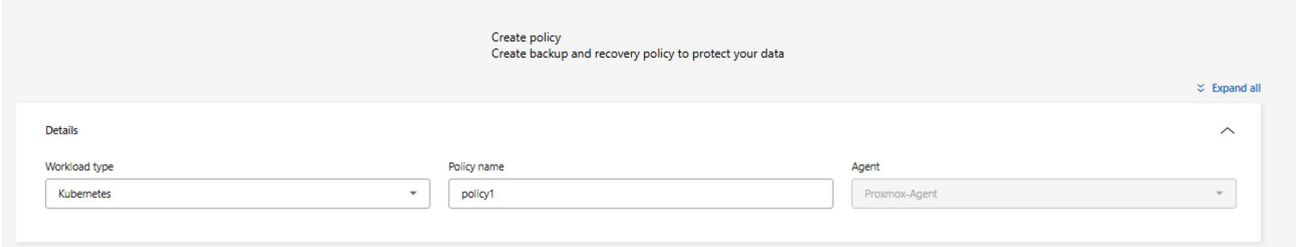
提供應用程式詳細資訊並定義應用程式的適用條件。在本例中，應用程式將會針對 vSphere Kubernetes 叢集中某個命名空間內的所有 Container 應用程式建立。按一下「搜尋」以查看屬於該應用程式的命名空間範圍資源清單。按一下「下一步」繼續。

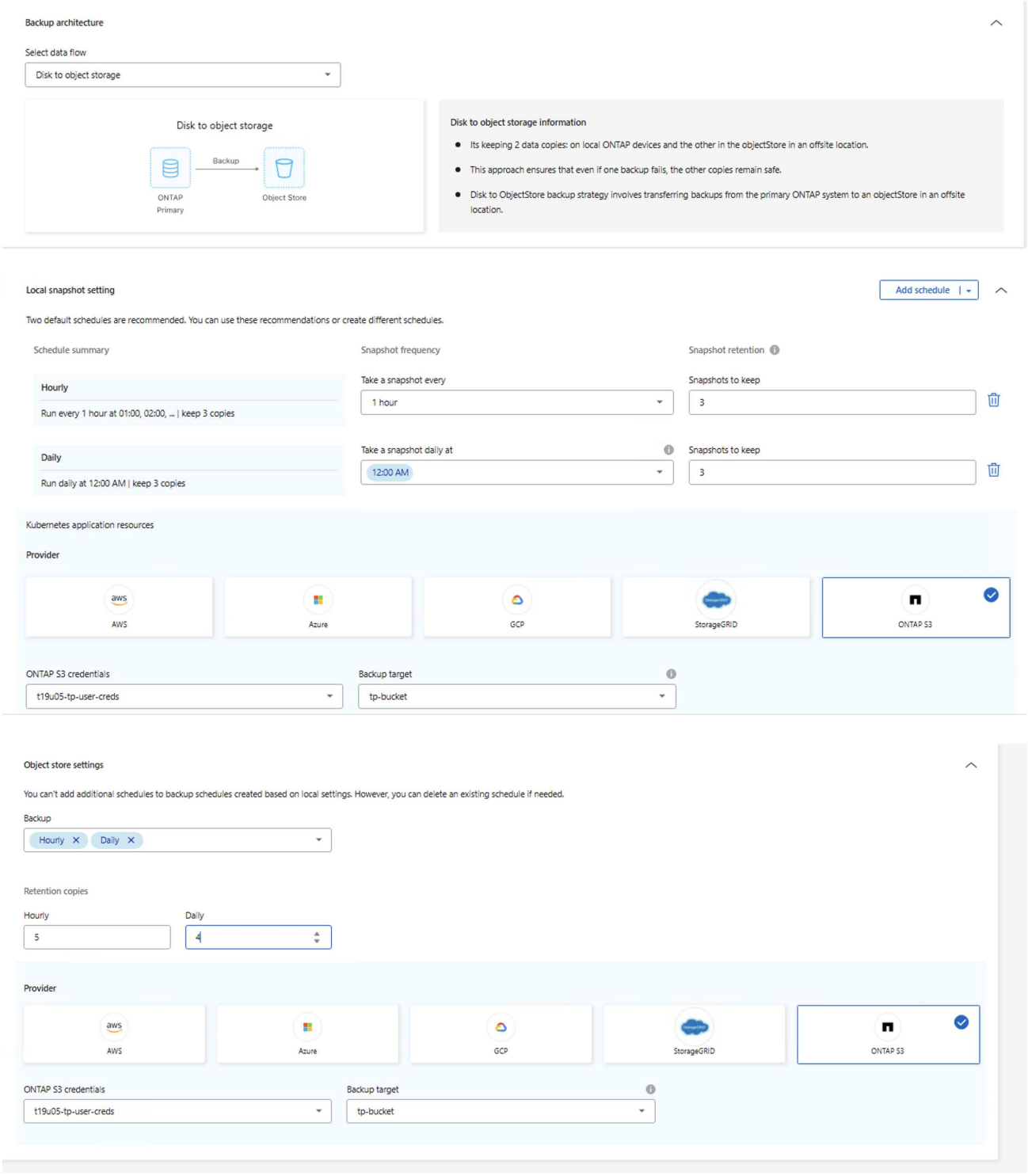


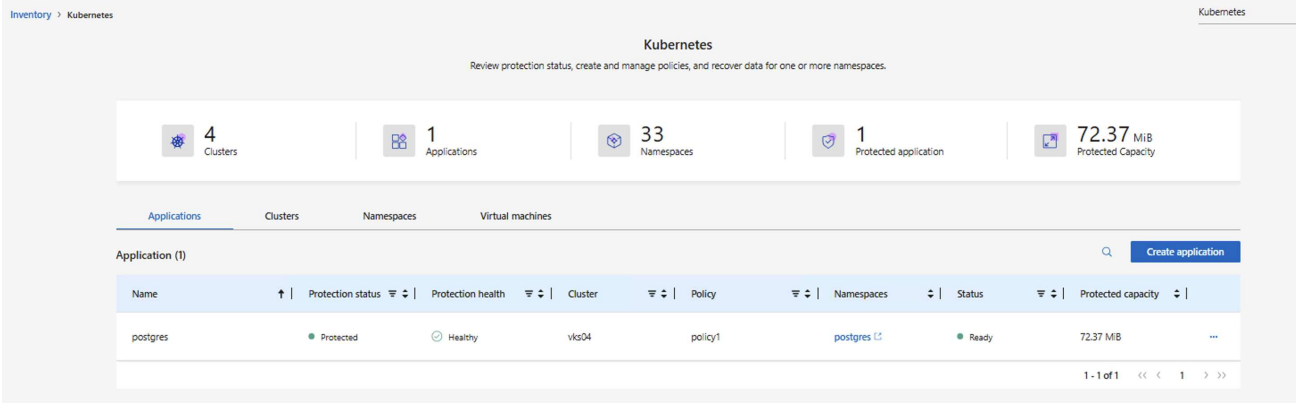
接下來，您需要為應用程式提供保護設定。您可以建立新的保護原則，也可以使用現有的原則。在本範例中，我們將為應用程式建立新的保護原則並將其關聯。展開 **Policy** ，然後按一下 **Create new policy** 。



為原則命名並選擇備份架構。在本例中，選擇磁碟到物件儲存設備。定義本機快照排程，該排程控制在主儲存設備上建立快照的頻率。然後單獨設定物件儲存設備備份設定：選擇備份目標（在本例中為 ONTAP S3），設定傳輸排程（控制快照資料何時複製到物件儲存設備），並指定保留複本的數量。傳輸排程獨立於快照排程，因此資源會根據傳輸排程複製到物件儲存設備，而不是在每次建立快照時立即複製。您也可以根據需要修改最大重試次數和重試間隔。按一下 **Create** 繼續。系統將探索該應用程式，並將保護原則與其建立關聯。




Backup architecture
Select data flow
Disk to object storage
Disk to object storage
Backup
ONTAP Primary
Object Store
Disk to object storage information
Its keeping 2 data copies: on local ONTAP devices and the other in the objectStore in an offsite location.
This approach ensures that even if one backup fails, the other copies remain safe.
Disk to ObjectStore backup strategy involves transferring backups from the primary ONTAP system to an objectStore in an offsite location.
Local snapshot setting
Add schedule
Two default schedules are recommended. You can use these recommendations or create different schedules.
Schedule summary
Snapshot frequency
Snapshot retention
Hourly
Run every 1 hour at 01:00, 02:00, ... | keep 3 copies
Take a snapshot every
1 hour
Snapshots to keep
3
Daily
Run daily at 12:00 AM | keep 3 copies
Take a snapshot daily at
12:00 AM
Snapshots to keep
3
Kubernetes application resources
Provider
AWS
Azure
GCP
StorageGRID
ONTAP S3
ONTAP S3 credentials
t19u05-tp-user-creds
Backup target
tp-bucket
Object store settings
You can't add additional schedules to backup schedules created based on local settings. However, you can delete an existing schedule if needed.
Backup
Hourly
Daily
Retention copies
Hourly
5
Daily
4
Provider
AWS
Azure
GCP
StorageGRID
ONTAP S3
ONTAP S3 credentials
t19u05-tp-user-creds
Backup target
tp-bucket

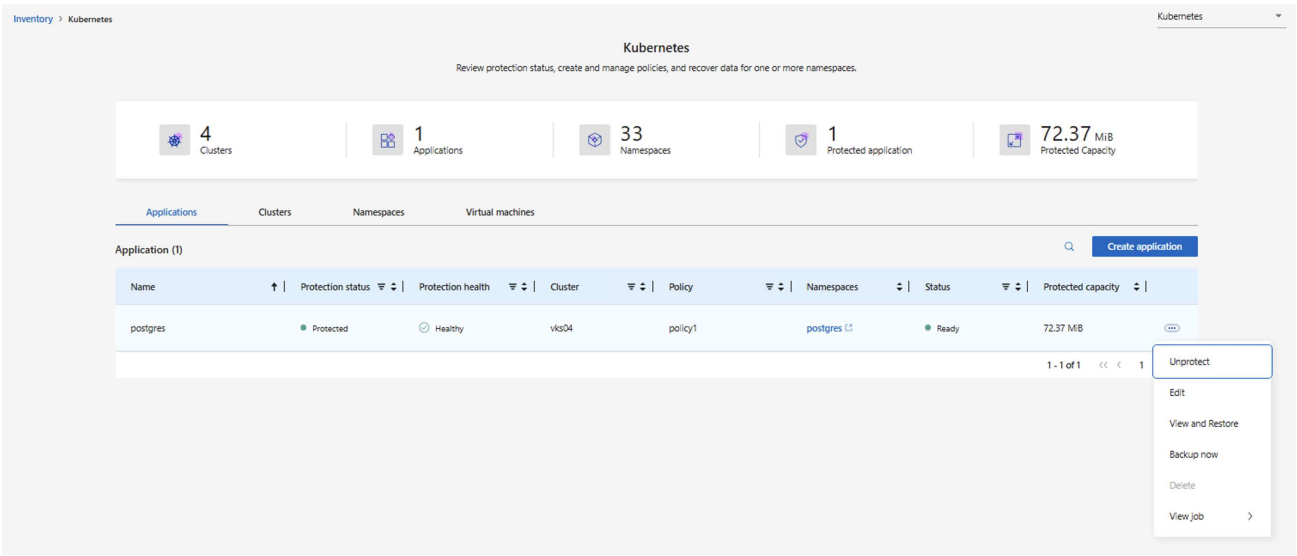


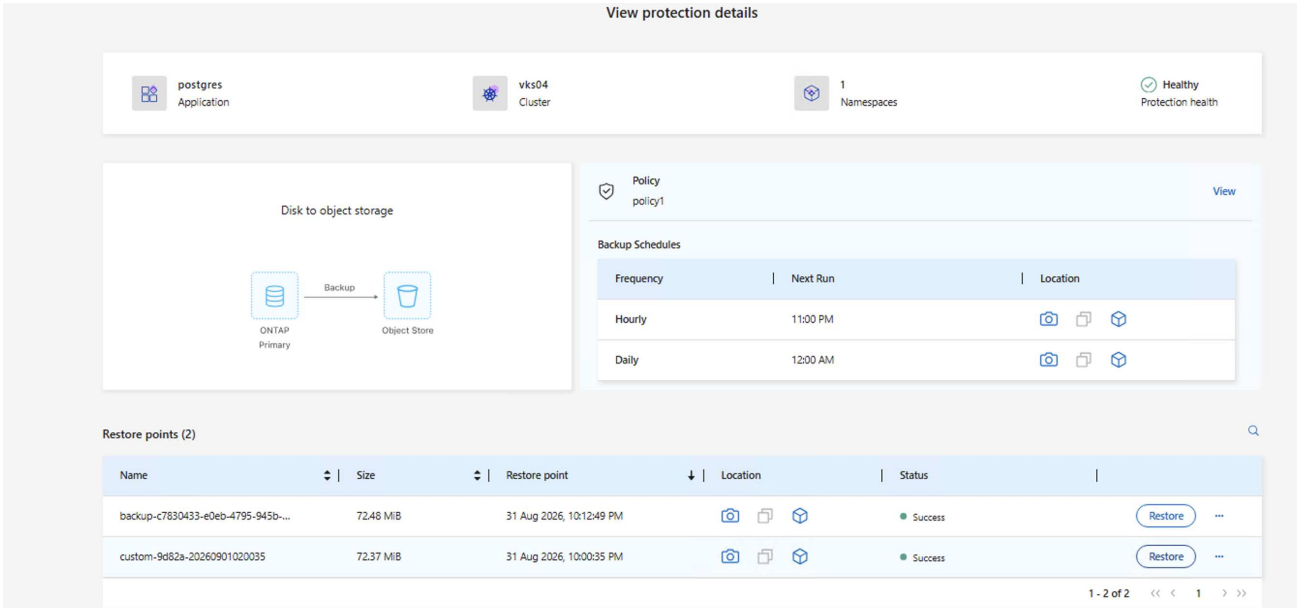
6. 從備份還原應用程式

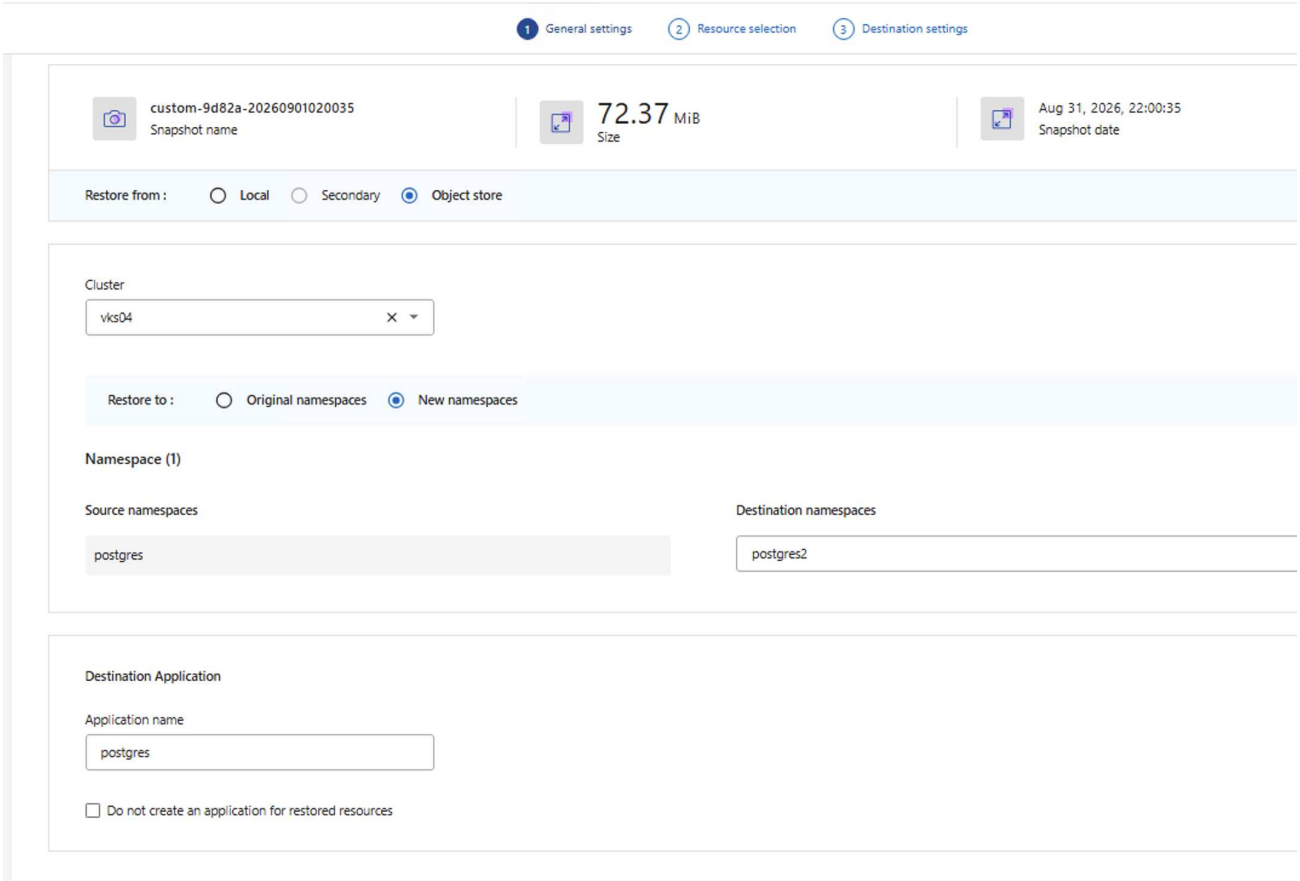
> 您需要 Backup and Recovery 超級管理員或 Backup and Recovery 還原管理員角色才能還原 Kubernetes 應用程式。

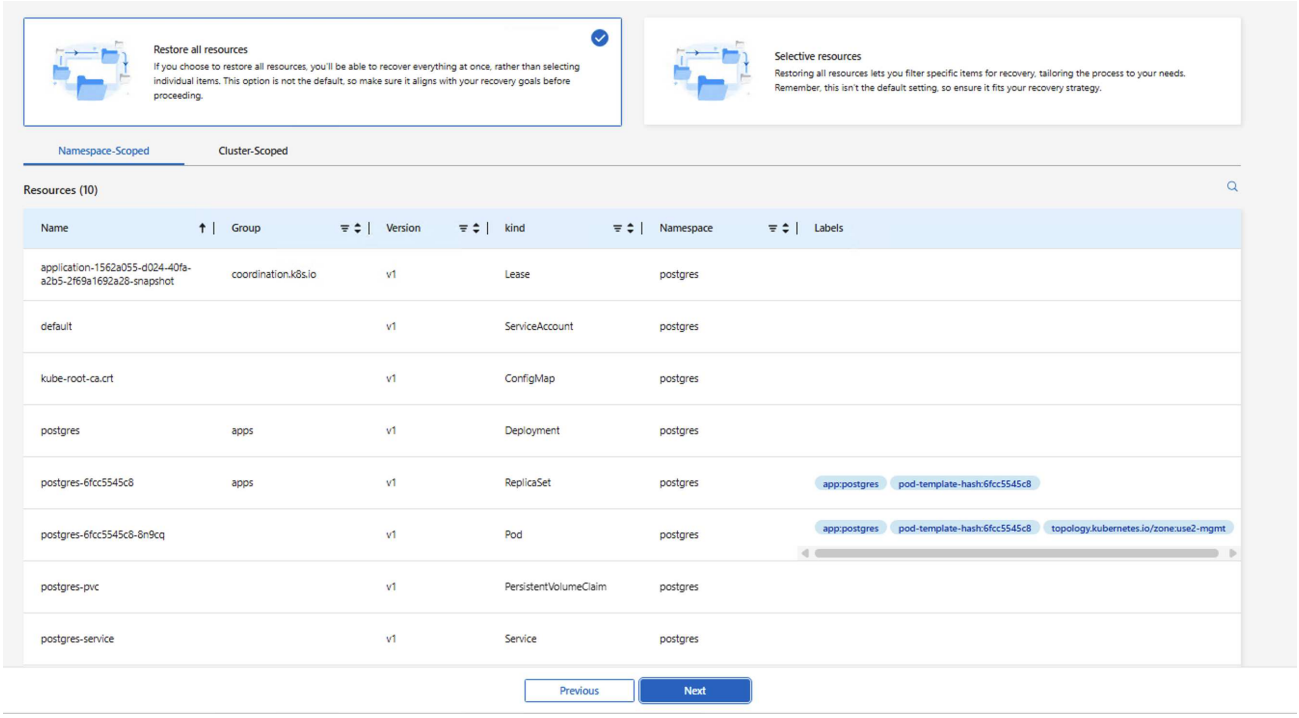
▼ 顯示範例

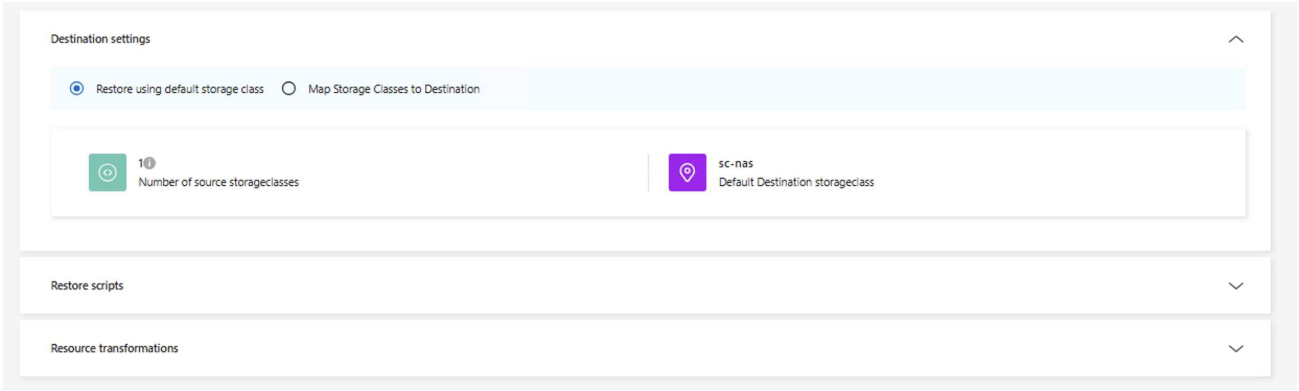
- 您可以選擇任何可用的還原點（本機快照、物件儲存備份或來自次要目標的備份）來還原應用程式。
- 您可以將應用程式還原到其原始命名空間或不同的命名空間。
- 您可以選擇要包含在還原中的資源。
- 您可以選擇用於還原應用程式資源的儲存設備類別。



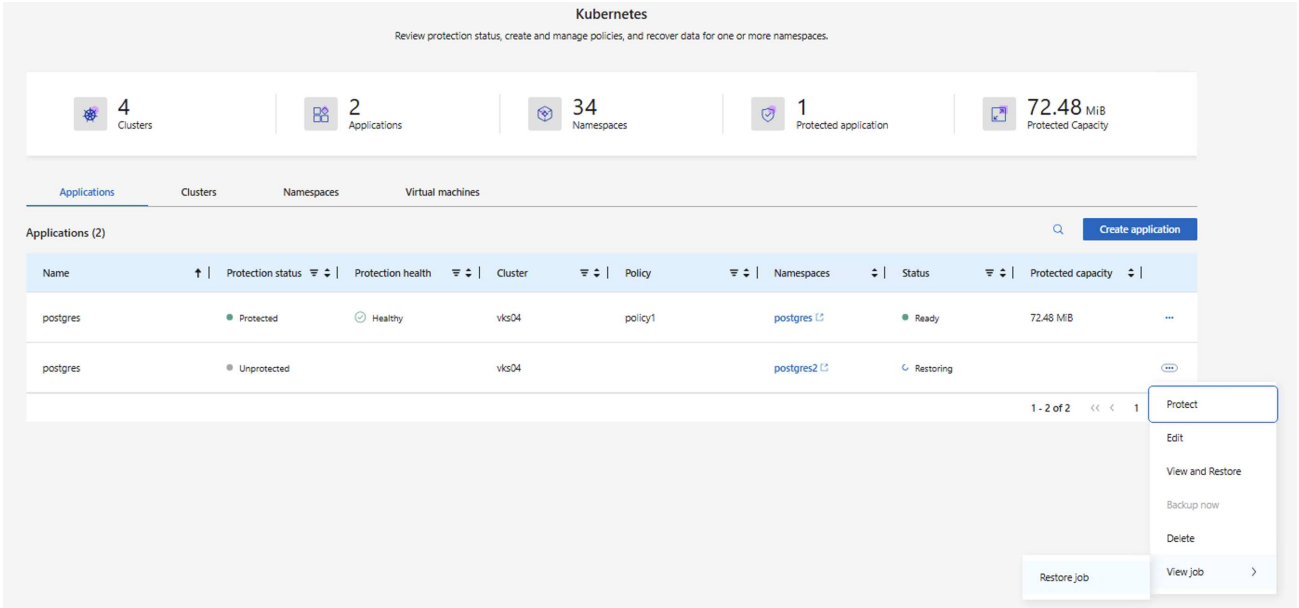

View protection details
postgres
Application
vks04
Cluster
1
Namespaces
Healthy
Protection health
Disk to object storage
Backup
ONTAP
Primary
Object Store
Policy
policy1
View
Backup Schedules
Frequency
Next Run
Location
Hourly
11:00 PM
Daily
12:00 AM
Restore points (2)
Name
Size
Restore point
Location
Status
backup-c7830433-e0eb-4795-945b-...
72.48 MiB
31 Aug 2026, 10:12:49 PM
Success
Restore
custom-9d82a-20260901020035
72.37 MiB
31 Aug 2026, 10:00:35 PM
Success
Restore
1 - 2 of 2
nimolab

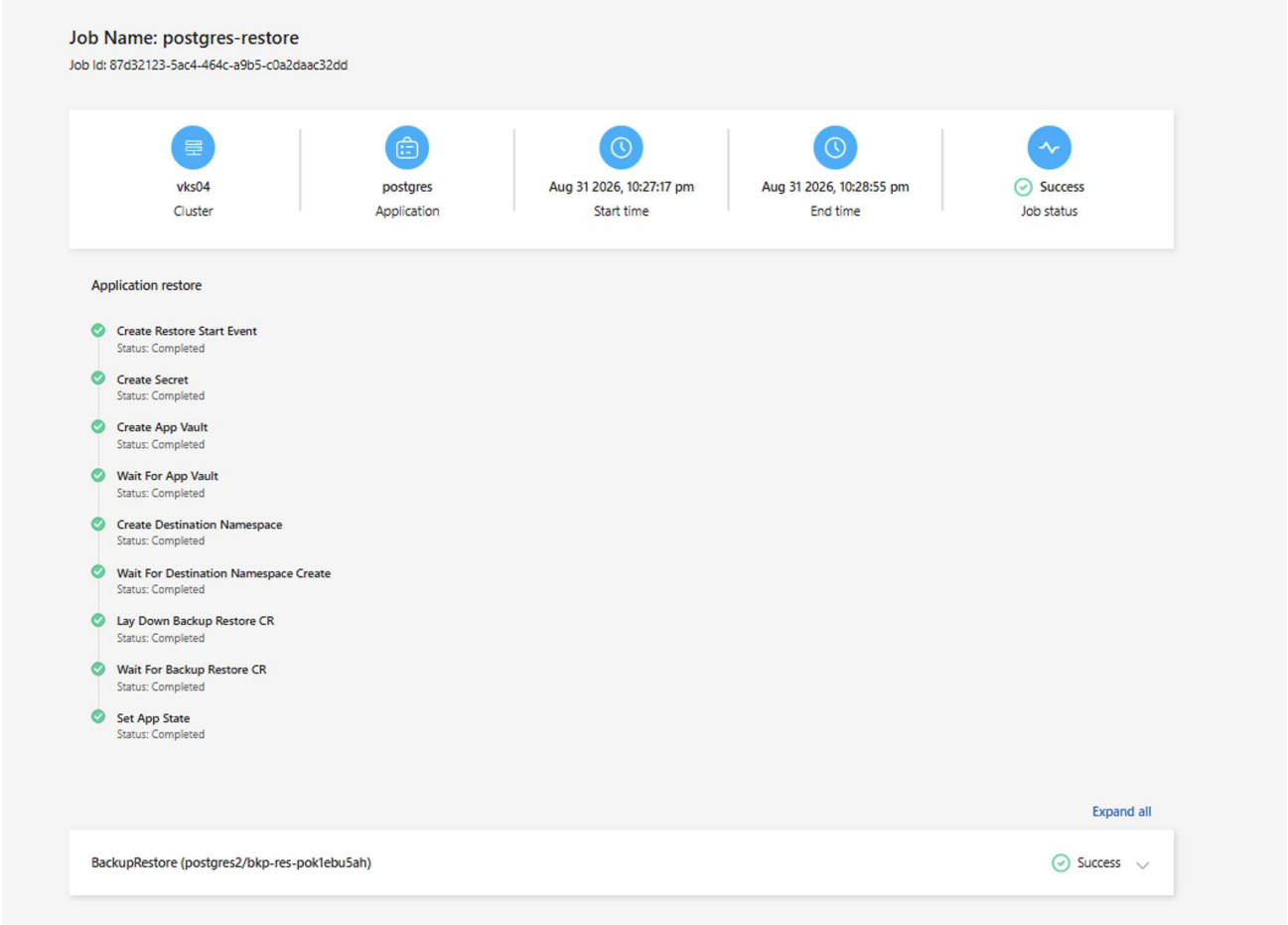

1 General settings
2 Resource selection
3 Destination settings
custom-9d82a-20260901020035
Snapshot name
72.37 MiB
Size
Aug 31, 2026, 22:00:35
Snapshot date
Restore from :
Local
Secondary
Object store
Cluster
vks04
Restore to :
Original namespaces
New namespaces
Namespace (1)
Source namespaces
postgres
Destination namespaces
postgres2
Destination Application
Application name
postgres
Do not create an application for restored resources

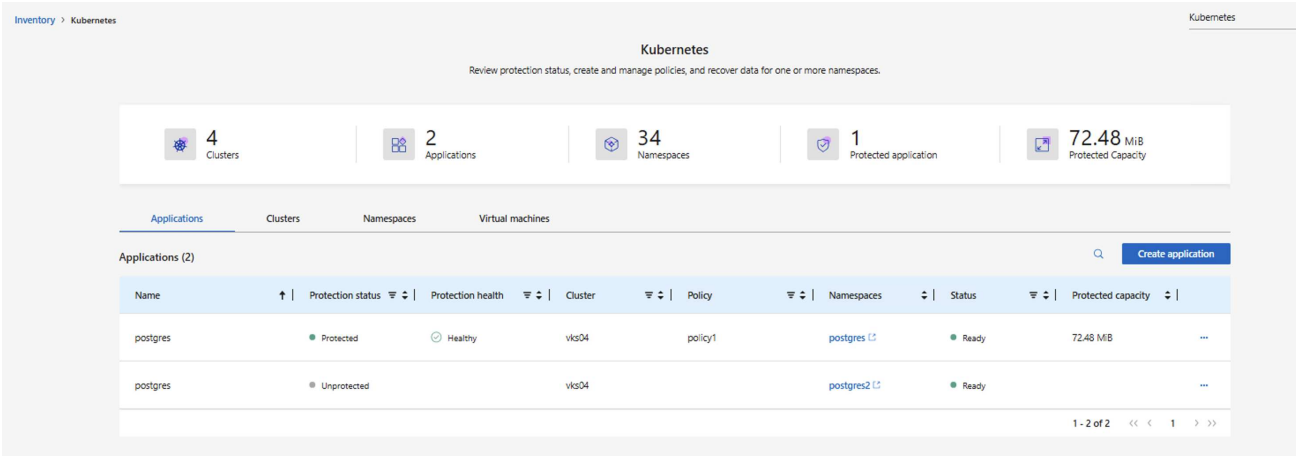




您可以在 NetApp Console 的「監控」部分查看還原作業的進度。還原作業完成後，您可以在 vSphere Kubernetes 叢集中看到已還原的資源。







使用 **Trident Protect** 在 **vSphere Kubernetes Service** 叢集中備份及還原 **Container** 應用程式

使用 Trident Protect 快照和備份功能，在 vSphere Kubernetes Service （VKS） 叢集中備份和還原 Container 應用程式。此程序包括使用 ONTAP S3 物件儲存建立 AppVault、設定 Trident Protect 以擷取應用程式資料（包括 Kubernetes 資源物件和持續儲存 Volume），以及在必要時還原資料。

在 VKS 叢集中執行的 Container 應用程式會作為 Kubernetes 工作負載在工作節點命名空間中進行代管。保護應用程式中繼資料和持續儲存 Volume 至關重要，以便在遺失或損毀時能夠加以恢復。

VKS 應用程式的持續性 Volume 可由與 VKS 叢集整合的 ONTAP 儲存設備提供支援，使用 "Trident CSI"。此程序使用 "Trident Protect" 建立應用程式快照和備份。應用程式快照的中繼資料儲存在 ONTAP 物件儲存中，而 Volume 快照資料則保留在儲存設備後端；備份資料則複製到 ONTAP 物件儲存。您可以在需要時從快照或備份還原。

Trident Protect 支援對 Kubernetes 叢集上的應用程式進行快照、備份、還原和災難恢復。Trident Protect 可保護的資料包括與應用程式及其持續性 Volume 相關聯的 Kubernetes 資源物件。

以下是本節範例中使用的各種元件版本

- VMware Cloud Foundation (VCF) 9.1 與 vSphere Kubernetes 服務
- "Trident 26.06"
- "Trident Protect 26.06"
- "ONTAP 9.16"

在 **vSphere Kubernetes** 叢集上安裝 **Trident Protect**

▼ 安裝 **Trident Protect**

使用 Helm 在 vSphere Kubernetes Service 叢集上安裝 Trident Protect。本節中的範例使用 `tridentctl-protect`，即 Trident Protect CLI。請依照"Trident Protect CLI 文件"中的說明進行安裝。其他 Trident Protect 安裝方法已記錄於"Trident Protect 說明文件"中。

首先，建立並標記 `trident-protect`命名空間。 `enforce=privileged`標籤為必要項目，因為 Trident Protect 執行特權工作負載。若無此標籤，Kubernetes Pod 安全性許可控制器將封鎖 Trident Protect Pod 的啟動。

```
kubectl create namespace trident-protect
kubectl label namespace trident-protect pod-
security.kubernetes.io/enforce=privileged
```

然後新增 Helm 儲存庫並將 Trident Protect 安裝到命名空間。

```
helm repo add netapp-trident-protect https://netapp.github.io/trident-
protect-helm-chart
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2606.0 --namespace
trident-protect
```

如果 Trident Protect Pod 啟動失敗並出現 `PodSecurity`許可錯誤，則可能是安裝前未將 Pod 安全性標籤套用到 `trident-protect`命名空間。請使用 `--overwrite`重新套用這些標籤，然後驗證 Pod 是否啟動。

```
kubectl label namespace trident-protect pod-
security.kubernetes.io/enforce=privileged --overwrite
```

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

```
vksbastion@vks:~/demo-vks$ kubectl get pods -n trident-protect
NAME                                                READY   STATUS    RESTARTS   AGE
autosupportbundle-e4cb1693-6640-4fcd-bddf-b413ff41f37c-znvsw   1/1     Running   0          16h
trident-protect-controller-manager-5f64ff594f-cl8cp            1/1     Running   0          3h50m
vksbastion@vks:~/demo-vks$
```

```
vksbastion@vks:~/demo-vks$ tridentctl-protect version -n trident-protect
26.06.0
vksbastion@vks:~/demo-vks$
```

建立用於物件儲存的 **App Vault**

▼ 建立 **AppVault**

在為應用程式建立快照和備份之前，必須先在 Trident Protect 中設定物件儲存設備。這可以透過建立 AppVault CR 來完成。只有管理員才能建立和設定 AppVault CR。

AppVault 物件是儲存設備貯體的 Kubernetes 自訂資源表示。AppVault CR 包含儲存設備貯體在保護作業（例如備份、快照、還原作業和 SnapMirror 複寫）中使用所需的組態。

以下步驟建立針對 ONTAP S3 設定的 AppVault CR：.在 ONTAP 叢集的 SVM 中建立 S3 物件存放區伺服器。.在物件存放區伺服器中建立儲存桶。.在 SVM 中建立 S3 使用者。請將存取金鑰和私密金鑰保存在安全的位置。

+ 注意：Trident Protect 要求 S3 使用者至少擁有 `PutObject`、 `GetObject`、 `ListBucket` 和 `DeleteObject` 權限。如果沒有這些權限，AppVault 備份與恢復作業將會失敗並出現存取被拒絕的錯誤。如需詳細資訊，請參閱"Trident Protect AppVault 文件"。。在 VKS 叢集中，建立一個金鑰來儲存 ONTAP S3 憑證。。為 ONTAP S3 建立 AppVault 物件。

為 **ONTAP S3** 設定 **Trident Protect AppVault**

以下清單使用啟用了憑證驗證的 HTTPS，這是生產環境的必要條件。如果您的 ONTAP S3 端點使用叢集已信任的公共 CA 所簽署的憑證，則無需進行額外的憑證組態。如果使用自簽名或內部 CA 憑證，請使用 `rootCA`欄位提供自訂根目錄 CA 憑證，如清單所示。如果您需要用於隔離的非生產環境測試的 HTTP 組態，請參閱本節末尾的實驗室專用版本。

```
# alias tp='tridentctl-protect'

# cat appvault-secret.yaml
apiVersion: v1
data:
  accessKeyID: <base64 encoded access key>
  secretAccessKey: <base64 encoded secret access key>
# Alternatively, remove the data section above and use:
# stringData:
#   accessKeyID: "<access key of S3>"
#   secretAccessKey: "<secret access key of S3>"
```

```
kind: Secret
metadata:
  name: ontap-s3-appvault-secret1
  namespace: trident-protect
type: Opaque

# cat appvault.yaml
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-appvault1
  namespace: trident-protect
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: trident-protect
      endpoint: <lif for S3 access>
      secure: "true"
      skipCertValidation: "false"
      # If your ONTAP S3 endpoint uses a certificate signed by a public
CA
      # already trusted by the cluster, no additional certificate config
is needed.
      # If it uses a self-signed or internal CA certificate, provide the
      # root CA certificate:
      # rootCA: <root CA certificate>
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: ontap-s3-appvault-secret1
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: ontap-s3-appvault-secret1
  providerType: OntapS3

# kubectl create -f appvault-secret.yaml -n trident-protect
```

```
# kubectl create -f appvault.yaml -n trident-protect
```

僅限實驗室環境（**HTTP**，無憑證驗證）

以下 `s3`設定僅可在不含敏感資料的隔離實驗室環境中使用。請勿在生產環境中使用這些設定。

```
    s3:
      bucketName: trident-protect
      endpoint: <lif for S3 access>
      secure: "false"
      skipCertValidation: "true"
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get appvault -n trident-protect
NAME                STATE       ERROR   MESSAGE   AGE
ontap-s3-appvault1  Available                     3d14h
vks-appvault        Available                     5d15h
vksbastion@vks:~/demo-vks/tp$
```

建立 **Trident Protect** 應用程式

▼ 建立 **Trident Protect** 應用程式

本範例涵蓋了安裝在 postgres 命名空間中的範例 postgres 應用程式的資料保護。如需安裝的詳細資訊，請參閱 "使用 NetApp 儲存設備部署 VKS 工作負載"。

> ⓘ 範例 PostgreSQL PVC 請求 RWO 存取模式。存取模式必須在 PVC 清單中明確指定；Trident 不會根據儲存傳輸協定自動選擇存取模式。NAS 支援的 PVC 支援 RWX 存取模式，SAN 支援的 PVC 可透過額外組態支援 RWX 存取模式。如需詳細資訊，請參閱 "Trident Protect 說明文件"。

在本例中，postgres 命名空間有一個應用程式，在建立 Trident Protect 應用程式時會包含該命名空間的所有資源。

```
# alias tp='tridentctl-protect'
# tp create app postgres-app --namespaces postgres -n postgres --dry-run
> postgres-app.yaml

# cat postgres-app.yaml
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: postgres-app
spec:
  includedNamespaces:
  - namespace: postgres
  resourceFilter: {}
```

```
# kubectl create -f postgres-app.yaml -n postgres
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get application -n postgres
NAME           PROTECTION STATE   AGE
postgres-app   None               3d13h
```

建立備份以保護應用程式

▼ 建立備份

建立隨需備份

為先前建立的 postgres-app 建立備份，備份內容應包含 postgres 命名空間中的所有資源。請提供用於儲存備份的 appvault 名稱。

```
# cat postgres-backup-on-demand.yaml
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  name: postgres-backup-on-demand
spec:
  appVaultRef: ontap-s3-appvault1
  applicationRef: postgres-app
  cleanupSnapshot: true
  replicateSnapshot: false
```

```
kubectl create -f postgres-backup-on-demand.yaml -n postgres
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-backup-on-demand.yaml -n postgres
backup.protect.trident.netapp.io/postgres-backup-on-demand created
vksbastion@vks:~/demo-vks/tp$ kubectl get backups -n postgres
NAME                        APP            RECLAIM POLICY   STATE     ERROR   AGE
postgres-backup-on-demand   postgres-app   Retain           Running           12s
vksbastion@vks:~/demo-vks/tp$ kubectl get backups -n postgres -w
NAME                        APP            RECLAIM POLICY   STATE     ERROR   AGE
postgres-backup-on-demand   postgres-app   Retain           Running           29s
postgres-backup-on-demand   postgres-app   Retain           Running           82s
postgres-backup-on-demand   postgres-app   Retain           Running           82s
postgres-backup-on-demand   postgres-app   Retain           Running           82s
postgres-backup-on-demand   postgres-app   Retain           Running           83s
postgres-backup-on-demand   postgres-app   Retain           Running           83s
postgres-backup-on-demand   postgres-app   Retain           Running           83s
postgres-backup-on-demand   postgres-app   Retain           Completed          83s
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get application -n postgres
NAME           PROTECTION STATE   AGE
postgres-app   Partial            3d13h
```

按排程建立備份

制定備份計劃，指定精細度和要保留的備份數量。

```
# tp create schedule backup-schedule1 --app postgres-app --appvault
ontap-s3-appvault1 --granularity Hourly --minute 45 --backup-retention 1
-n postgres --dry-run>postgres-backup-schedule1.yaml

#cat postgres-backup-schedule1.yaml
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: backup-schedule1
  namespace: postgres
spec:
  appVaultRef: ontap-s3-appvault1
  applicationRef: postgres-app
  backupRetention: "1"
  dayOfMonth: ""
  dayOfWeek: ""
  enabled: true
  granularity: Hourly
  hour: ""
  minute: "45"
  recurrenceRule: ""
  replicationRetention: "0"
  runImmediately: false
  snapshotRetention: "0"
# kubectl create -f postgres-backup-schedule1.yaml -n postgres
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-backup-schedule1.yaml -n postgres
schedule.protect.trident.netapp.io/backup-schedule1 created
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get schedule -n postgres
NAME               ENABLED   GRANULARITY   STATE   ERROR   AGE
backup-schedule1   true      Hourly                        10m
```

從備份還原

▼ 從備份還原

將應用程式還原到相同的命名空間

在這個範例中，備份 postgres-backup-on-demand 包含 postgres-app 的備份。

刪除應用程式之前，請確認您計劃從中還原的備份處於 `Completed`狀態。進行中或失敗的備份無法用於還原應用程式。

```
kubectl get backup -n postgres
```

確認備份狀態為 `Completed`後，刪除 postgres 應用程式，並確保從命名空間「postgres」中刪除 PVC 和 pod 物件。

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all -n postgres
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-6gw9h   1/1     Running   0          3d13h

NAME                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.101.183.142   <none>        5432/TCP   3d13h

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           3d13h

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       3d13h
vksbastion@vks:~/demo-vks/tp$ kubectl delete deployment/postgres -n postgres
deployment.apps "postgres" deleted from postgres namespace
vksbastion@vks:~/demo-vks/tp$ kubectl get all -n postgres
NAME                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.101.183.142   <none>        5432/TCP   3d13h
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get pvc -n postgres
NAME           STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   V
OLUMEATTRIBUTESCLASS   AGE
postgres-pvc   Bound    pvc-43f275d5-6063-4751-98d5-3a36b07d0bf0   10Gi       RWO            sc-nas         <
unset>                 3d14h
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres
No resources found in postgres namespace.
```

現在，建立一個備份原地還原物件。

```
# tp create bir postgres-app-restore --backup postgres/postgres-backup-
on-demand -n postgres --dry-run>postgres-app-bir.yaml

# cat postgres-app-bir.yaml
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  annotations:
    protect.trident.netapp.io/max-parallel-restore-jobs: "25"
  name: postgres-app-restore
  namespace: postgres
spec:
  appArchivePath: postgres-app_314bbaf6-2ce3-4065-b3c1-
ab85c7bb7c7c/backups/postgres-backup-on-demand_63efc9d1-92d7-45ce-84fe-
846ce7db7b29
  appVaultRef: ontap-s3-appvault1
  cleanUpAdditionalExecHooks: true
  cleanUpArchivedExecHooks: false
  resourceFilter: {}
```

```
  runArchivedExecHooks: true

# kubectl create -f postgres-app-bir.yaml -n postgres
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-app-bir.yaml -n postgres
backupinplacerestore.protect.trident.netapp.io/postgres-app-restore created
```

驗證 postgres 應用程式部署、服務、pod 和 PVC 是否已還原。

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-9pv2m   1/1     Running   0          2m1s

NAME                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.107.38.167   <none>        5432/TCP   2m1s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           2m1s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       2m1s

NAME                                 STATUS   VOLUME                                     CAPACITY   ACCESS MO
DES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc   Bound    pvc-191af706-64ac-4f09-921a-ff9bf6d86a68   10Gi       RWO
      sc-nas         <unset>                 2m6s
```

將應用程式還原到不同的命名空間

首先，建立一個新的命名空間，以便將應用程式還原至其中，本範例中為 postgres2。此時，postgres-app 已可使用依排程建立的每小時備份。使用此備份將應用程式還原至新的命名空間 postgres2。

```
vksbastion@vks:~/demo-vks/tp$ kubectl get backup -n postgres
NAME                           APP            RECLAIM POLICY   STATE       ERROR   AGE
hourly-cbd11-20260831104500    postgres-app   Retain           Completed           14m
postgres-backup-on-demand      postgres-app   Retain           Completed           51m
```

```
# tp create backuprestore --appvault ontap-s3-appvault1 --path postgres-
app_314bbaf6-2ce3-4065-b3c1-ab85c7bb7c7c/backups/hourly-cbd11-
20260831124500_2b57bda9-5cb0-4c8f-ae7a-20f94c23c3b9 --namespace-mapping
postgres:postgres2 -n postgres2 --dry-run>postgres-app-postgres2-br.yaml
```

若要取得備份的路徑，請使用命令 `kubectl get backups <backup-name> -n postgres -o jsonpath='{.status.appArchivePath}'`。

```
vksbastion@vks:~/demo-vks/tp$ kubectl get backups -n postgres
NAME                           APP            RECLAIM POLICY   STATE       ERROR   AGE
hourly-cbd11-20260831124500    postgres-app   Retain           Completed           13m
postgres-backup-on-demand      postgres-app   Retain           Completed           170m
vksbastion@vks:~/demo-vks/tp$ tp get backups -n postgres -o yaml | grep hourly-cbd11-20260831124500
    name: hourly-cbd11-20260831124500
    appArchivePath: postgres-app_314bbaf6-2ce3-4065-b3c1-ab85c7bb7c7c/backups/hourly-cbd11-20260831124500_2b5
7bda9-5cb0-4c8f-ae7a-20f94c23c3b9
vksbastion@vks:~/demo-vks/tp$
```

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
```

```
metadata:
  annotations:
    protect.trident.netapp.io/max-parallel-restore-jobs: "25"
  name: postgres-app-z2woek
  namespace: postgres2
spec:
  appArchivePath: postgres-app_314bbaf6-2ce3-4065-b3c1-
ab85c7bb7c7c/backups/hourly-cbd11-20260831124500_2b57bda9-5cb0-4c8f-
ae7a-20f94c23c3b9
  appVaultRef: ontap-s3-appvault1
  cleanUpAdditionalExecHooks: true
  cleanUpArchivedExecHooks: false
  namespaceMapping:
  - destination: postgres2
    source: postgres
  resourceFilter: {}
  runArchivedExecHooks: true
  skipApplicationCreation: false

# kubectl create -f postgres-app-postgres2-br.yaml -n postgres2
```

```
vksbastion@vks:~/demo-vks/tp$ tp create backuprestore --appvault ontap-s3-appvault1 --path postgres-app_314bb
af6-2ce3-4065-b3c1-ab85c7bb7c7c/backups/hourly-cbd11-20260831124500_2b57bda9-5cb0-4c8f-ae7a-20f94c23c3b9 --na
mespace-mapping postgres:postgres2 -n postgres2 --dry-run>postgres-app-postgres2-br.yaml
vksbastion@vks:~/demo-vks/tp$
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-app-postgres2-br.yaml -n postgres2
backuprestore.protect.trident.netapp.io/postgres-app-btguyt created
```

驗證 postgres 應用程式物件和 PVC 是否已在新命名空間 postgres2 中建立。

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres2
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-858dt   1/1     Running   0          72s

NAME                       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.109.5.180   <none>        5432/TCP   72s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           72s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       72s

NAME                                STATUS   VOLUME                                     CAPACITY   ACCESS MO
DES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc   Bound    pvc-5893851c-c152-439f-a147-c4ae3581cc6f   10Gi       RWO
      sc-nas         <unset>                 78s
vksbastion@vks:~/demo-vks/tp$
```

使用 **Snapshot** 快照技術保護應用程式

▼ 建立 **Snapshot** 快照技術

建立隨選 **Snapshot** 快照技術 為應用程式建立 Snapshot 快照技術，並指定 AppVault 以存放 Snapshot 快照技術中繼資料。Volume Snapshot 快照技術資料本身不會複製到物件儲存設備，而是保留在儲存設備後端。

```
# tp create snapshot postgres-app-snapshot-ondemand --app postgres-app
```

```
--appvault ontap-s3-appvault1 -n postgres --dry-run>postgres-app
-snapshot-ondemand.yaml

# cat postgres-app-snapshot-ondemand.yaml
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  name: postgres-app-snapshot-ondemand
  namespace: postgres
spec:
  appVaultRef: ontap-s3-appvault1
  applicationRef: postgres-app
  cleanupSnapshot: false
  completionTimeout: 0s
  volumeSnapshotsCreatedTimeout: 0s
  volumeSnapshotsReadyToUseTimeout: 0s

# kubectl create -f postgres-app-snapshot-ondemand.yaml
snapshot.protect.trident.netapp.io/postgres-app-snapshot-ondemand
created
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-app-snapshot-ondemand.yaml
snapshot.protect.trident.netapp.io/postgres-app-snapshot-ondemand created
vksbastion@vks:~/demo-vks/tp$
vksbastion@vks:~/demo-vks/tp$ kubectl get snapshots -n postgres
NAME                              APP            RECLAIM POLICY   STATE       ERROR   AGE
postgres-app-snapshot-ondemand    postgres-app   Delete           Completed           20s
vksbastion@vks:~/demo-vks/tp$
```

建立 **Snapshot** 快照技術計劃 建立 Snapshot 快照技術計劃。指定 Snapshot 快照技術的精細度以及要保留的 Snapshot 快照技術數量。

```
# tp create schedule snapshot-schedule1 --app postgres-app --appvault
ontap-s3-appvault1 --granularity Hourly --minute 55 --snapshot-retention
1 -n postgres --dry-run>postgres-app-snapshot-schedule1.yaml

# cat postgres-app-snapshot-schedule1.yaml
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: snapshot-schedule1
  namespace: postgres
spec:
  appVaultRef: ontap-s3-appvault1
  applicationRef: postgres-app
  backupRetention: "0"
  dayOfMonth: ""
```

```
    dayOfWeek: ""
    enabled: true
    granularity: Hourly
    hour: ""
    minute: "55"
    recurrenceRule: ""
    replicationRetention: "0"
    runImmediately: false
    snapshotRetention: "1"

# kubectl create -f postgres-app-snapshot-schedule1.yaml
schedule.protect.trident.netapp.io/snapshot-schedule1 created
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-app-snapshot-schedule1.yaml -n postgres
schedule.protect.trident.netapp.io/snapshot-schedule1 created
vksbastion@vks:~/demo-vks/tp$ kubectl get schedule -n postgres
NAME                 ENABLED   GRANULARITY   STATE   ERROR   AGE
backup-schedule1     true      Hourly                        3h37m
snapshot-schedule1   true      Hourly                        10s
vksbastion@vks:~/demo-vks/tp$ kubectl get snapshots -n postgres -w
NAME                              APP            RECLAIM POLICY   STATE       ERROR   AGE
postgres-app-snapshot-ondemand    postgres-app   Delete           Completed           32m
hourly-f1dd9-20260831135500       postgres-app   Delete                               0s
hourly-f1dd9-20260831135500       postgres-app   Delete                               0s
hourly-f1dd9-20260831135500       postgres-app   Delete                               0s
hourly-f1dd9-20260831135500       postgres-app   Delete                               0s
hourly-f1dd9-20260831135500       postgres-app   Delete                               0s
hourly-f1dd9-20260831135500       postgres-app   Delete                               0s
hourly-f1dd9-20260831135500       postgres-app   Delete                               0s
hourly-f1dd9-20260831135500       postgres-app   Delete                               0s
hourly-f1dd9-20260831135500       postgres-app   Delete                               0s
hourly-f1dd9-20260831135500       postgres-app   Delete                               0s
hourly-f1dd9-20260831135500       postgres-app   Delete           Running             0s
hourly-f1dd9-20260831135500       postgres-app   Delete           Running             0s
hourly-f1dd9-20260831135500       postgres-app   Delete           Running             0s
hourly-f1dd9-20260831135500       postgres-app   Delete           Running             0s
^Cvksbastion@vks:~/demo-vks/tp$ kubectl get snapshots -n postgres
NAME                              APP            RECLAIM POLICY   STATE       ERROR   AGE
hourly-f1dd9-20260831135500       postgres-app   Delete           Completed           15s
postgres-app-snapshot-ondemand    postgres-app   Delete           Completed           33m
```

從 **Snapshot** 快照技術還原

▼ 從 **Snapshot** 快照技術還原

將應用程式從 **Snapshot** 快照技術還原至相同命名空間

刪除應用程式之前，請確認您計劃從中還原的 Snapshot 快照技術處於 `Completed`狀態。進行中或失敗的 Snapshot 快照技術無法用於還原應用程式。

```
kubectl get snapshot -n postgres
```

確認 Snapshot 快照技術狀態為 `Completed`後，從 postgres 命名空間中刪除 postgres 的應用程式物件件和 PVC。

```
vksbastion@vks:~/demo-vks/tp$ kubectl get deployment,svc,pvc -n postgres
NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres    1/1     1            1           3h14m

NAME                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.107.38.167   <none>        5432/TCP   3h14m

NAME                                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc    Bound    pvc-191af706-64ac-4f09-921a-ff9bf6d86a68   10Gi       RWO            sc-nas
<unset>                 3h14m
vksbastion@vks:~/demo-vks/tp$ kubectl delete deployment/postgres -n postgres
deployment.apps "postgres" deleted from postgres namespace
vksbastion@vks:~/demo-vks/tp$ kubectl delete service/postgres-service -n postgres
service "postgres-service" deleted from postgres namespace
vksbastion@vks:~/demo-vks/tp$ kubectl delete pvc/postgres-pvc -n postgres
persistentvolumeclaim "postgres-pvc" deleted from postgres namespace
vksbastion@vks:~/demo-vks/tp$ kubectl get deployment,svc,pvc -n postgres
No resources found in postgres namespace.
vksbastion@vks:~/demo-vks/tp$
```

從 Snapshot 快照技術建立原地還原物件。

```
# tp create sir postgres-restore-from-snapshot --snapshot
postgres/hourly-f1dd9-20260831135500 -n postgres --dry-run > postgres-
sir.yaml

# cat postgres-sir.yaml
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: postgres-restore-from-snapshot
  namespace: postgres
spec:
  appArchivePath: postgres-app_314bbaf6-2ce3-4065-b3c1-
ab85c7bb7c7c/snapshots/20260831135500_hourly-f1dd9-
20260831135500_36c2bd2a-9405-4424-88a7-75e6bbc5b315
  appVaultRef: ontap-s3-appvault1
  cleanUpAdditionalExecHooks: true
  cleanUpArchivedExecHooks: false
  resourceFilter: {}
  runArchivedExecHooks: true

# kubectl create -f postgres-sir.yaml
snapshotinplacerestore.protect.trident.netapp.io/postgres-restore-from-
snapshot created
```

確認應用程式的物件和 PVC 是否在 postgres 命名空間中建立。

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-wrppb   1/1     Running   0          43s

NAME                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.101.142.54   <none>        5432/TCP   43s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           43s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       43s

NAME                                 STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc   Bound    pvc-40f55cd3-ba5c-40b6-94ba-52a4e6767b33   10Gi       RWO            sc-nas
<unset>                 49s
vksbastion@vks:~/demo-vks/tp$
```

將應用程式從 **Snapshot** 快照技術還原至不同的命名空間

刪除先前從備份還原的 postgres2 命名空間中的應用程式。

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres2
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-858dt   1/1     Running   0          65m

NAME                       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.109.5.180   <none>        5432/TCP   65m

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           65m

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       65m

NAME                                 STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc   Bound    pvc-5893851c-c152-439f-a147-c4ae3581cc6f   10Gi       RWO            sc-nas
<unset>                 65m
vksbastion@vks:~/demo-vks/tp$ kubectl delete deployment/postgres -n postgres2
deployment.apps "postgres" deleted from postgres2 namespace
vksbastion@vks:~/demo-vks/tp$ kubectl delete service/postgres-service -n postgres2
service "postgres-service" deleted from postgres2 namespace
vksbastion@vks:~/demo-vks/tp$ kubectl delete pvc/postgres-pvc -n postgres2
persistentvolumeclaim "postgres-pvc" deleted from postgres2 namespace
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres2
No resources found in postgres2 namespace.
vksbastion@vks:~/demo-vks/tp$
```

從 Snapshot 快照技術建立 Snapshot 快照技術還原物件，並提供命名空間映射。

```
# tp create sr postgres-sr --snapshot postgres/hourly-f1dd9-
20260831135500 --namespace-mapping postgres:postgres2 -n postgres2 --dry
-run>postgres-sr.yaml

# cat postgres-sr.yaml
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: postgres-sr
  namespace: postgres2
spec:
  appArchivePath: postgres-app_314bbaf6-2ce3-4065-b3c1-
ab85c7bb7c7c/snapshots/20260831135500_hourly-f1dd9-
20260831135500_36c2bd2a-9405-4424-88a7-75e6bbc5b315
  appVaultRef: ontap-s3-appvault1
  cleanUpAdditionalExecHooks: true
```

```
  cleanUpArchivedExecHooks: false
  namespaceMapping:
  - destination: postgres2
    source: postgres
  resourceFilter: {}
  runArchivedExecHooks: true
  skipApplicationCreation: false

# kubectl create -f postgres-sr.yaml
snapshotrestore.protect.trident.netapp.io/postgres-sr created
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get snapshotrestore  -n postgres2
NAME          STATE       ERROR   AGE
postgres-sr   Completed           44s
```

驗證應用程式的物件和 PVC 是否已在 postgres2 命名空間中還原。

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres2
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-ql8h4   1/1     Running   0          3m29s

NAME                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.107.103.215   <none>        5432/TCP   3m29s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           3m29s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       3m29s

NAME                                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc   Bound    pvc-11e41614-4706-4bc7-ab77-da57b3baf754   10Gi       RWO            sc-nas
<unset>                 3m33s
vksbastion@vks:~/demo-vks/tp$
```