vSphere Kubernetes Service con almacenamiento NetApp

NetApp container solutions

NetApp
September 01, 2026

Tabla de contenidos

vSphere Kubernetes Service con almacenamiento NetApp

Conoce el servicio Kubernetes de VMware vSphere con NetApp ONTAP

El servicio Kubernetes de VMware vSphere (VKS) es una solución gestionada de Kubernetes de VMware que permite a las organizaciones poner en marcha, gestionar y escalar aplicaciones en contenedores mediante Kubernetes. En combinación con el almacenamiento ONTAP de NetApp, VKS ofrece persistencia, rendimiento y gestión de datos de clase empresarial para cargas de trabajo nativas de la nube.

VKS está integrado en VMware Cloud Foundation (VCF) y cumple con Kubernetes de la CNCF upstream, lo que garantiza la compatibilidad con el ecosistema y las herramientas de Kubernetes en general.

Funciones clave de vSphere Kubernetes Service

1. **Clústeres de Kubernetes gestionados:** vSphere Kubernetes Service simplifica la implementación y la gestión de clústeres de Kubernetes. Automatiza tareas como la creación de clústeres, las actualizaciones, el escalado y la supervisión.
2. **Integración con la infraestructura de VMware:** VKS se integra a la perfección con VMware vSphere, NSX-T y otras soluciones de VMware, lo que permite a las organizaciones aprovechar su infraestructura existente de VMware para las cargas de trabajo de Kubernetes.
3. **Compatibilidad con entornos multinube e híbridos:** El servicio de Kubernetes de vSphere permite el despliegue en nubes privadas, nubes públicas (por ejemplo, AWS, Azure, Google Cloud) y entornos híbridos, lo que ofrece a las organizaciones flexibilidad para gestionar sus aplicaciones.
4. **Seguridad integrada:** VKS incluye funciones de seguridad como el control de acceso basado en roles (RBAC), la aplicación de políticas y la integración con VMware NSX para la seguridad de la red.
5. **Compatibilidad con el ecosistema de Kubernetes:** VKS es compatible con toda la API de Kubernetes upstream, lo que garantiza la portabilidad entre entornos y la compatibilidad con el ecosistema más amplio de Kubernetes, incluidas herramientas como Istio para la malla de servicios, Prometheus para la monitorización y otras herramientas certificadas por la CNCF. Las organizaciones pueden aplicar los conocimientos e integraciones estándar de Kubernetes directamente a los clústeres de VKS.
6. **Herramientas pensadas para desarrolladores:** VKS es compatible con los flujos de trabajo estándar de Kubernetes para desarrolladores, incluidos los procesos de CI/CD, las prácticas de GitOps y herramientas como kubectl y Helm, lo que permite a los equipos de desarrollo crear e implementar aplicaciones en contenedores sin necesidad de aprender interfaces propias.
7. **Escalabilidad y alta disponibilidad:** VKS ofrece funciones como el escalado automático y la tolerancia a fallos para garantizar que las aplicaciones mantengan un alto nivel de disponibilidad y rendimiento.
8. **Observabilidad y supervisión:** VKS se integra con herramientas de supervisión estándar compatibles con Kubernetes para ofrecer información en tiempo real sobre el rendimiento del clúster y de las aplicaciones.

Casos de uso para vSphere Kubernetes Service

1. **Modernización de aplicaciones:** Las organizaciones pueden modernizar sus aplicaciones heredadas mediante su contenedorización y su puesta en marcha en clústeres de Kubernetes gestionados por VKS.

2. **DevOps Enablement:** VKS es compatible con los flujos de trabajo de DevOps al ofrecer automatización, integración de CI/CD y herramientas amigables para los desarrolladores.
3. **Implementaciones en la nube híbrida:** Las empresas pueden usar VKS para implementar clústeres de Kubernetes tanto en entornos locales como en la nube, lo que permite operaciones coherentes.
4. **Arquitectura de microservicios:** VKS es compatible con el desarrollo de aplicaciones basadas en microservicios, lo que permite a los equipos crear y escalar sistemas distribuidos.

Introducción al almacenamiento y al servicio de Kubernetes de vSphere

Instala un clúster de Kubernetes Service de vSphere

Instala un clúster de vSphere Kubernetes Service (VKS) en tu entorno VCF 9.1 y configura los nodos de trabajo con las utilidades de almacenamiento adecuadas para tus cargas de trabajo.

Acerca de esta tarea

Las utilidades de NFS se instalan automáticamente en los nodos de trabajo del clúster que creas. Si quieres crear nodos de trabajo con utilidades iSCSI o NVMe instaladas, debes crear una imagen OVA personalizada y usar esa imagen para los nodos de trabajo. La imagen personalizada se puede crear usando Docker, y la herramienta de línea de comandos `vcf` se puede usar para crear un archivo OVA a partir de esta imagen. Este archivo OVA se puede subir a la Content Library en vSphere. Al crear el clúster VKS, puedes seleccionar esta imagen de la Content Library para los grupos de nodos.

Pasos

1. Crea el clúster VKS:

   Puedes crear un clúster VKS utilizando la imagen predeterminada del nodo de trabajo o una imagen personalizada que tú crees. La imagen predeterminada del nodo de trabajo incluye utilidades NFS preinstaladas, mientras que la imagen personalizada puede incluir utilidades iSCSI y NVMe. Las siguientes opciones están disponibles:

   - **Solo NAS (predeterminado):** Crea un clúster VKS usando la imagen predeterminada del nodo de trabajo, que incluye utilidades NFS preinstaladas.
   - **SAN habilitada (imagen personalizada):** Crea una imagen personalizada de Docker que incluya utilidades iSCSI y NVMe, genera un archivo OVA usando la herramienta de línea de comandos `vcf`, sube el OVA a la vSphere Content Library y luego crea el clúster VKS seleccionando esa imagen personalizada para los grupos de nodos.

     ▼ Mostrar instrucciones para crear un clúster NAS-only de vSphere Kubernetes Service con la imagen predeterminada del nodo de trabajo

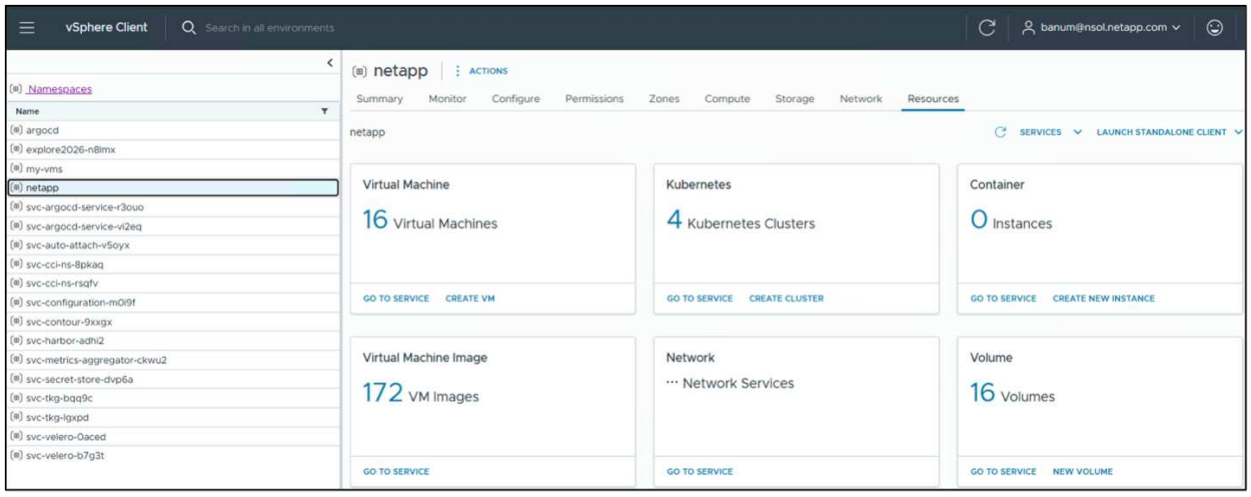
     Crea el clúster VKS utilizando la imagen predeterminada del nodo de trabajo. Sigue las instrucciones que aparecen en pantalla para especificar el nombre del clúster, la configuración del grupo de nodos y otros parámetros.

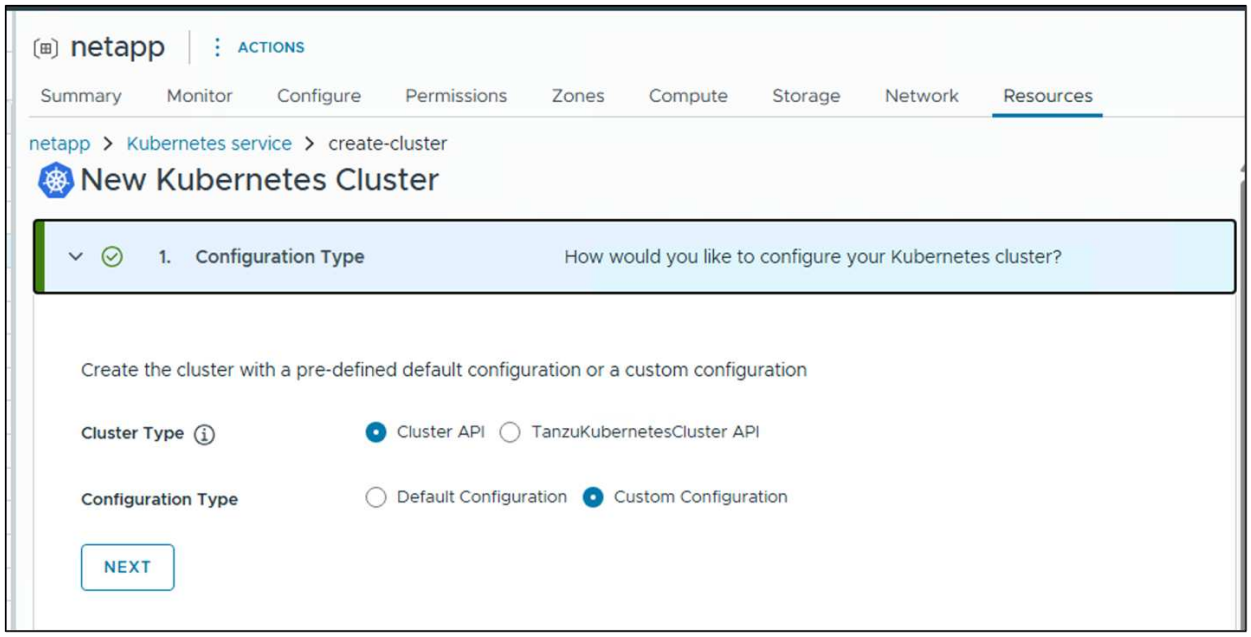
     Desde el vSphere Client, ve al espacio de nombres donde quieres crear el clúster de Kubernetes de vSphere. En la pestaña **Recursos** de ese espacio de nombres, haz clic en **Crear clúster** en la sección de Kubernetes, o haz clic en **Ir al servicio** y luego en **Crear clúster**.

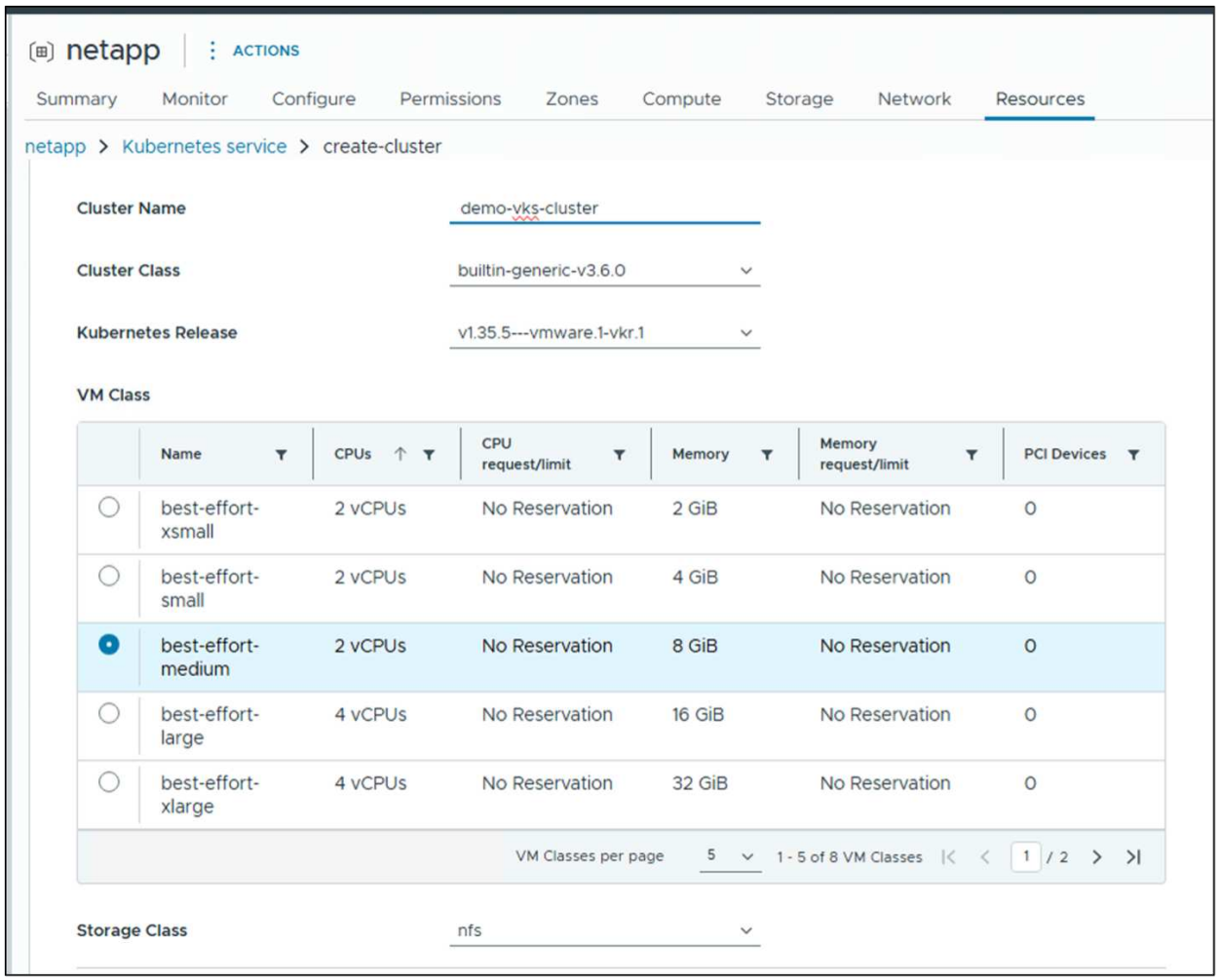
     Rellena el formulario con los datos del clúster:

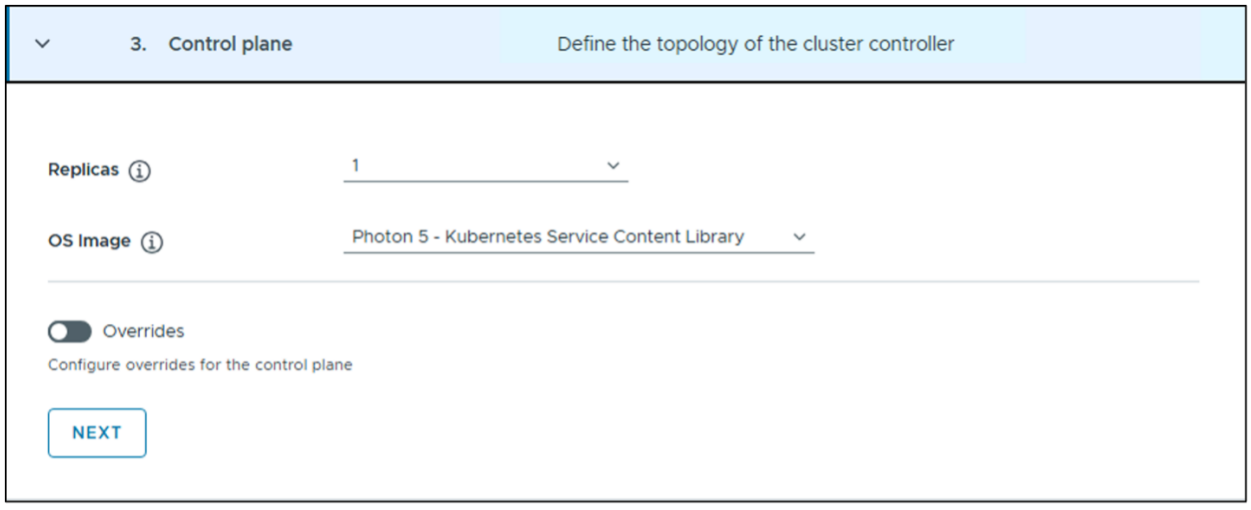
- En la sección 1, **Tipo de configuración**, selecciona **Custom**.
- En la sección 2, **Configuración general**, asigna un nombre al clúster y deja el resto de ajustes con sus valores predeterminados.
- Acepta todos los valores predeterminados de la sección 3.
- En la sección 4, **Grupos de nodos**, haz clic en los tres puntos (…) junto al grupo de nodos y haz clic en **Editar**. Aumenta las réplicas a 3 y selecciona la versión más reciente de Ubuntu para la imagen del sistema operativo. Haz clic en **Siguiente** y luego en **Revisar y confirmar**.

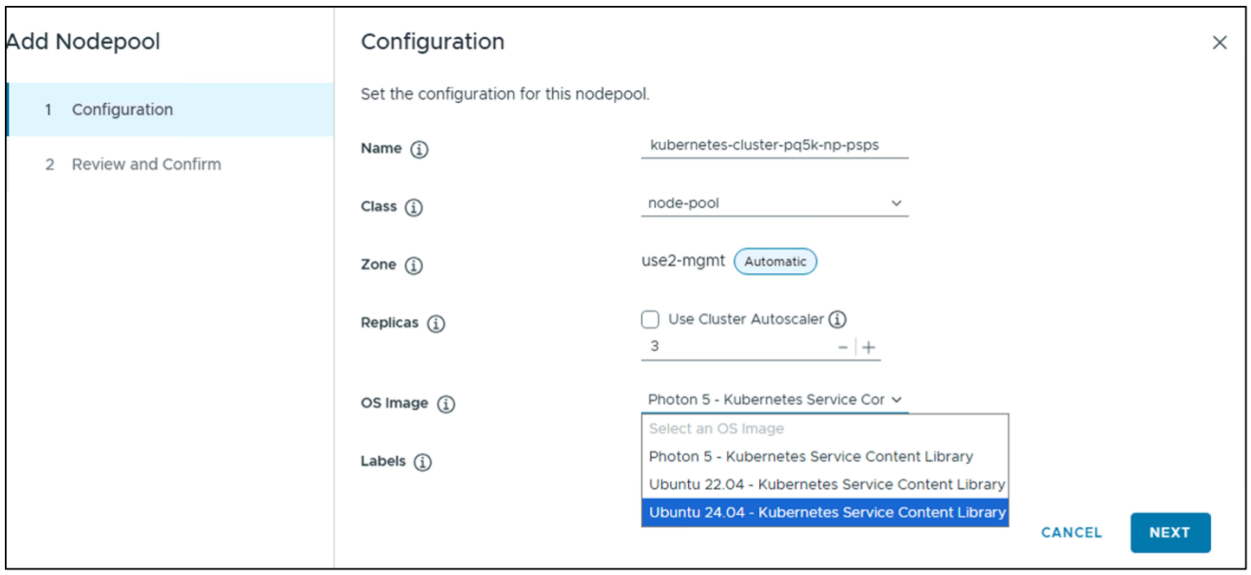
Tras revisar los detalles del clúster, haz clic en **Finalizar**. El clúster de Kubernetes vSphere se creará en unos minutos.

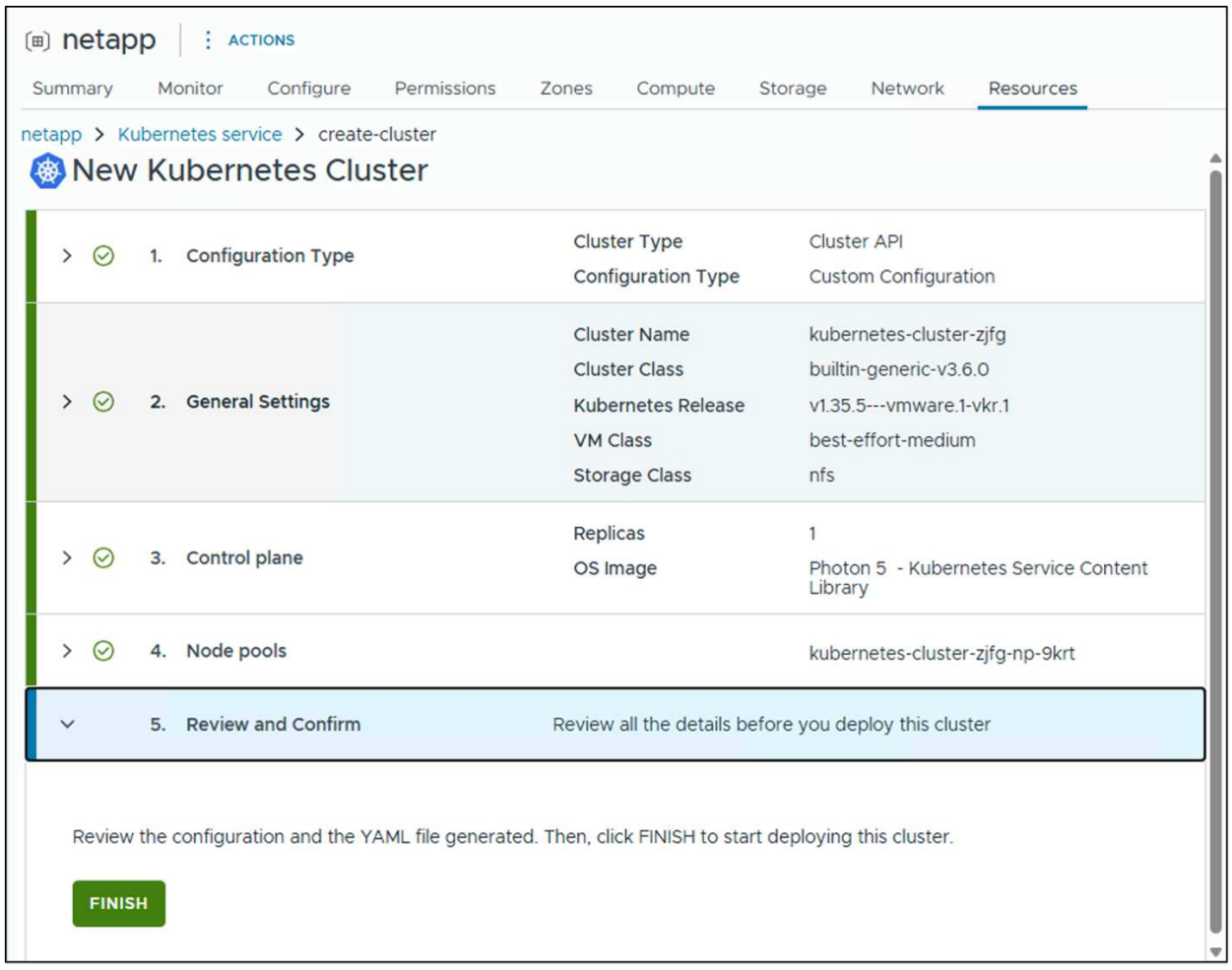


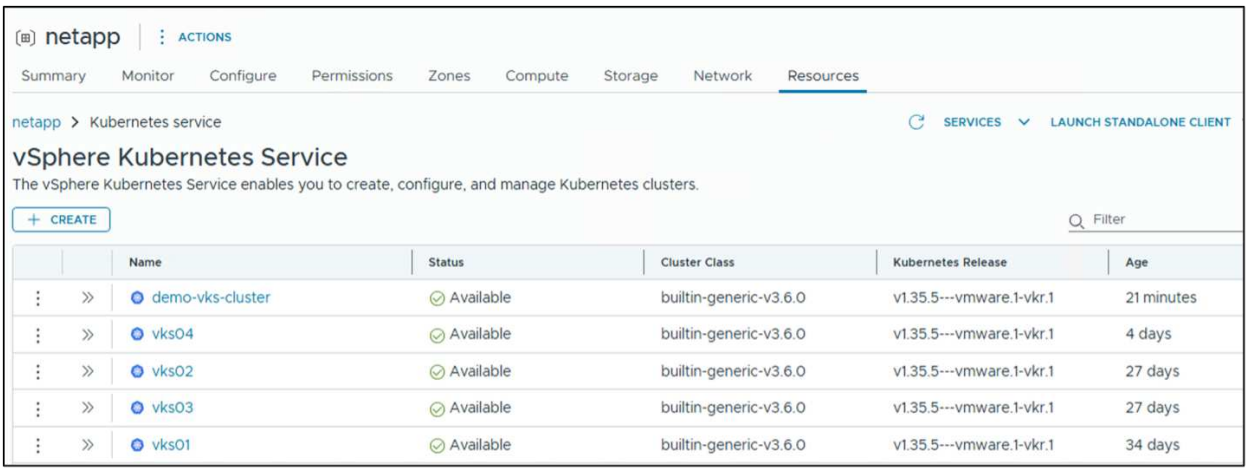



netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp > Kubernetes service > create-cluster
Cluster Name
demo-vks-cluster
Cluster Class
builtin-generic-v3.6.0
Kubernetes Release
v1.35.5---vmware.1-vkr.1
VM Class
Name
CPUs
CPU request/limit
Memory
Memory request/limit
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



▼ Mostrar instrucciones para crear una imagen OVA personalizada para nodos de trabajo habilitados para SAN

Crea una imagen de Docker con las utilidades necesarias instaladas. El siguiente ejemplo muestra cómo crear una imagen de Docker basada en Ubuntu 24.04 con las utilidades de iSCSI y Multipathing instaladas.

```
#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"]
```

Crea la imagen de Docker utilizando:

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

Si el registro de Docker no está disponible, ejecuta este comando para iniciar un registro local y subir la imagen a él:

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

Etiqueta la imagen y haz push.

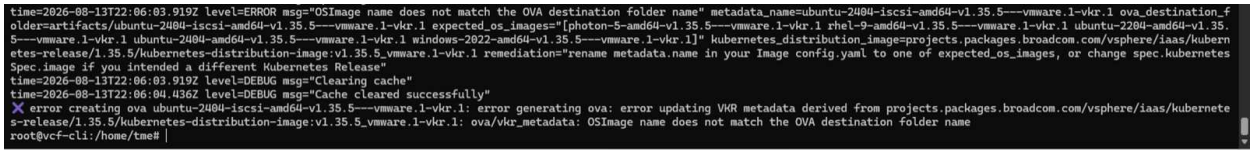
```
docker tag ubuntu-2404:san localhost:5000/ubuntu-2404:san
docker push localhost:5000/ubuntu-2404:san
```

Crea el archivo de configuración de la imagen (guárdalo como `ubuntu2404vkr135-san.yml`) e incluye una referencia a la imagen:

```
#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
```

El nombre de los metadatos debe ser uno de la lista preaprobada. Si no, recibirás un error como el que se muestra abajo cuando intentes crear el archivo OVA. La lista preaprobada se puede encontrar en la ayuda de la CLI de VCF para el comando `kr bake`.



Crea el archivo OVA usando la CLI de VCF:

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

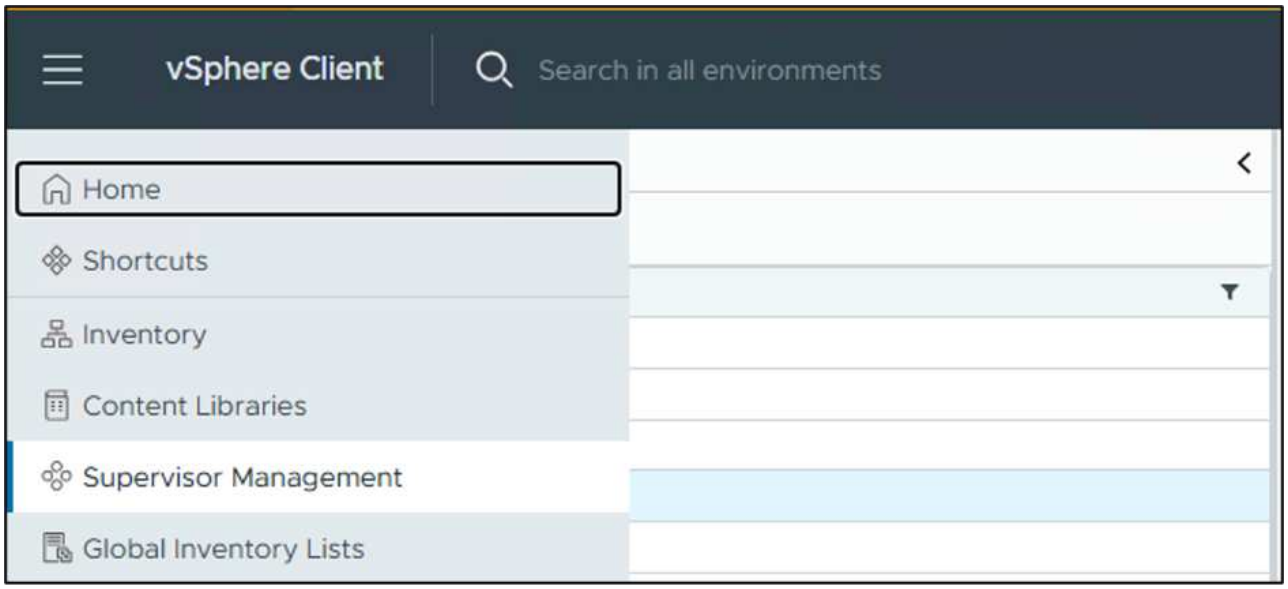
Transfiere el archivo OVA mediante SCP a un equipo desde el que puedas subirlo a la vSphere Content Library.

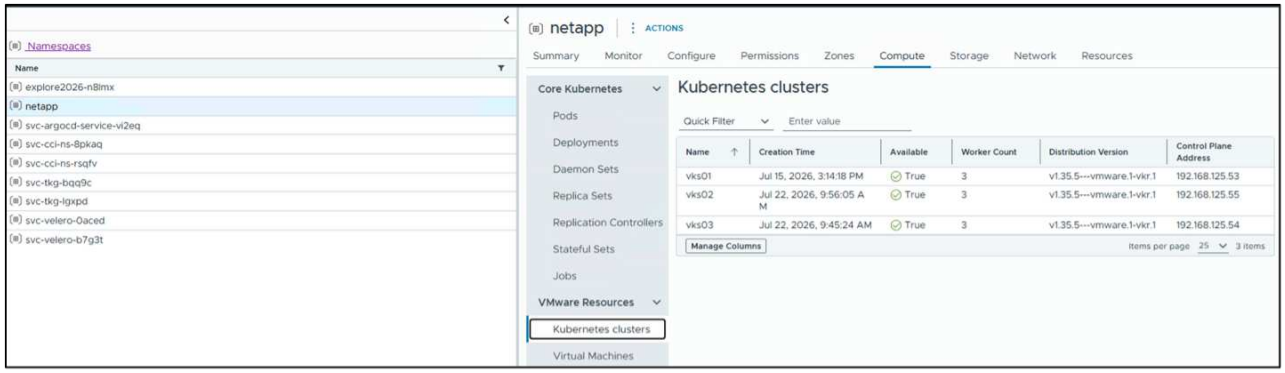
2. Una vez instalado el clúster, búscalo en vCenter y descarga el archivo kubeconfig.

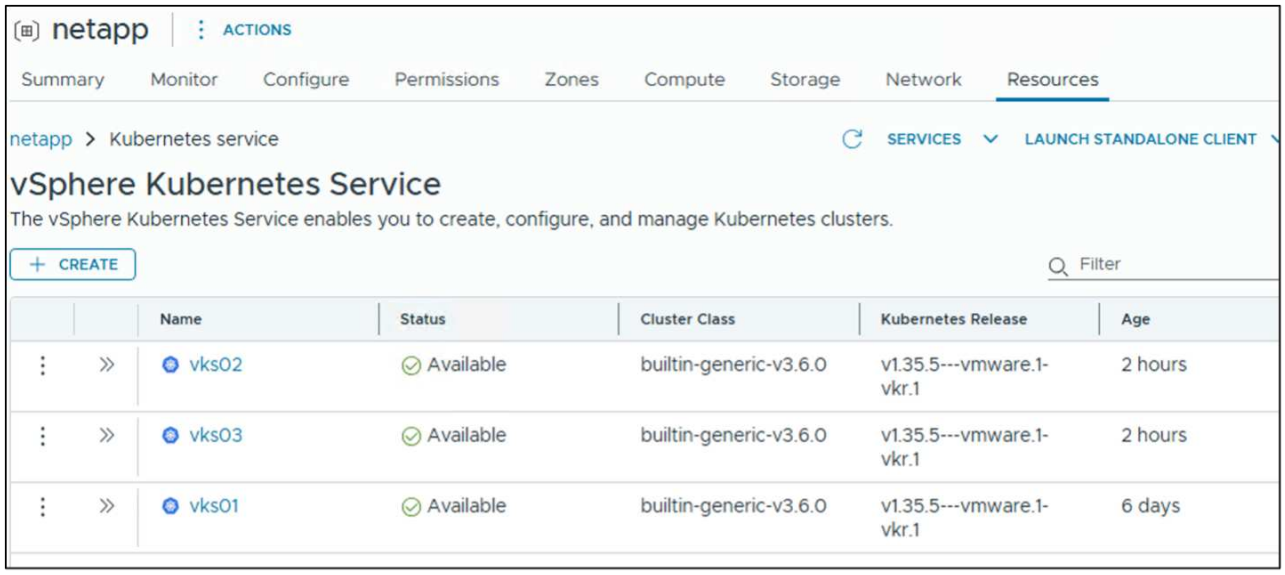
a. Haz clic en **Gestión de supervisores** en el menú principal y selecciona el espacio de nombres en el que se creó tu clúster.

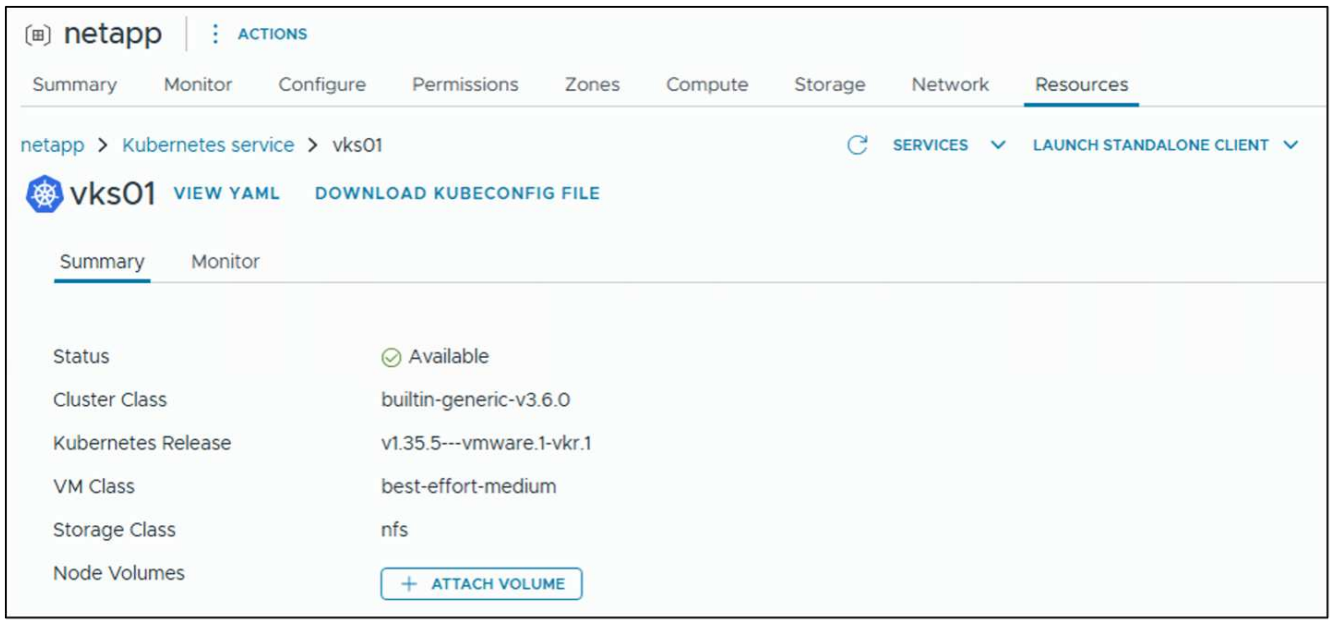
b. Selecciona la pestaña **Compute** y luego **Kubernetes Clusters**. Se muestran todos los clústeres del espacio de nombres.

c. Ve a la pestaña **Recursos**, selecciona el clúster de Kubernetes que necesites y descarga su archivo kubeconfig.









3. Desde un host de bastion, conéctate al clúster utilizando el archivo kubeconfig para gestionarlo desde la CLI.

```
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
```

Demostración en vídeo

Los siguientes vídeos muestran dos formas de crear un clúster de Kubernetes con vSphere.

Crea un clúster de Kubernetes vSphere en un clúster Supervisor

Crea un clúster de Kubernetes de vSphere con la automatización de VCF

Conoce el almacenamiento de NetApp para vSphere Kubernetes Service

El uso de NetApp como almacenamiento para vSphere Kubernetes Service (VKS) es una combinación potente que aprovecha las sólidas soluciones de almacenamiento de NetApp para mejorar el rendimiento, la escalabilidad y la fiabilidad de las cargas de trabajo de Kubernetes. NetApp Trident, un proveedor de almacenamiento dinámico de código abierto para Kubernetes, se utiliza para integrar el almacenamiento con las cargas de trabajo en vSphere Kubernetes Service.

Ventajas principales de usar el almacenamiento NetApp con VKS

1. **Aprovisionamiento dinámico de almacenamiento:** NetApp Trident permite el aprovisionamiento dinámico de volúmenes de almacenamiento, lo que permite a los pods de Kubernetes solicitar almacenamiento sobre la marcha según sus necesidades.
2. **Almacenamiento persistente:** Las soluciones de NetApp ofrecen capacidades de almacenamiento persistente, asegurando la durabilidad y la disponibilidad de los datos incluso cuando se reinician los pods

o hay fallos.

3. **Alto rendimiento:** Las soluciones de almacenamiento de NetApp, como ONTAP, ofrecen un almacenamiento de alto rendimiento con baja latencia, lo que resulta ideal para aplicaciones con un uso intensivo de I/O que se ejecutan en Kubernetes.
4. **Escalabilidad:** Los sistemas de almacenamiento NetApp pueden escalar para satisfacer las demandas de las cargas de trabajo de Kubernetes en crecimiento, admitiendo implementaciones a gran escala.
5. **Gestión de datos:** NetApp ofrece funciones avanzadas de gestión de datos, como instantáneas, clonación y replicación de datos, que pueden ser útiles para la copia de seguridad, la recuperación ante desastres y los flujos de trabajo de desarrollo.
6. **Compatibilidad con entornos multicloud e híbridos:** Las soluciones de almacenamiento de NetApp pueden implementarse en entornos locales, cloud pública y cloud híbrida, lo que ofrece flexibilidad y coherencia en la gestión del almacenamiento.

NetApp descripción general de la integración con ONTAP

NetApp ONTAP ofrece almacenamiento de clase empresarial con funciones como deduplicación de datos, compresión y cifrado. Usas NetApp Trident para integrar el almacenamiento de NetApp con Kubernetes. NetApp Trident es compatible con varias plataformas de almacenamiento de NetApp, incluyendo ONTAP en las instalaciones, FSx para NetApp ONTAP en AWS, Azure NetApp Files en Azure y Google Cloud NetApp Volumes en Google Cloud.

Usas Trident Protect para gestionar la protección de datos y la recuperación ante desastres de tus cargas de trabajo de Kubernetes. Trident Protect te permite crear instantáneas, clones y replicar datos en diferentes backends de almacenamiento, asegurando que tus datos estén seguros y sean recuperables en caso de fallos.

Funciones clave de Trident

Cumplimiento de CSI garantiza la compatibilidad con las últimas funciones y estándares de almacenamiento de Kubernetes. **Configuración declarativa** permite a los usuarios definir los requisitos de almacenamiento en archivos YAML, que luego son gestionados automáticamente por Trident. **Controladores** Trident admite controladores para los protocolos NFS, CIFS/SMB, iSCSI, FC y NVMe/TCP compatibles con ONTAP. **Compatibilidad híbrida y multicloud** Trident admite controladores para almacenamiento ONTAP en las instalaciones y servicios de almacenamiento gestionados en los hyperscalers, concretamente FSx for NetApp ONTAP en AWS, Azure NetApp Files en Azure y Google Cloud NetApp Volumes en Google Cloud. **Gestión avanzada de datos** aprovechando las capacidades de instantáneas y clonación de ONTAP para una protección de datos eficiente y un aprovisionamiento rápido de entornos de prueba y desarrollo.

Para obtener información detallada sobre Trident, consulta la página "Documentación de Trident".

Trident Protect

Trident Protect se instala por separado de Trident. Antes de la implementación, comprueba que tu plataforma de Kubernetes, el backend de almacenamiento, los componentes de instantáneas y el almacenamiento de objetos cumplan con los "Requisitos de Trident Protect".

Funciones clave de Trident Protect

Copias de seguridad automáticas Trident Protect permite realizar copias de seguridad automáticas de aplicaciones de Kubernetes condicionadas por políticas, lo que garantiza que se realicen copias de seguridad periódicas sin necesidad de intervención manual.

Gestión de instantáneas aprovecha las capacidades de instantáneas de ONTAP para crear copias

frecuentes y de un momento específico de aplicaciones en contenedores y máquinas virtuales, proporcionando un mecanismo fiable para la recuperación.

Cifrado de repositorios de copias de seguridad Trident Protect admite el cifrado mediante contraseña para los repositorios de copias de seguridad de Restic y Kopia. Configura por separado el cifrado de volúmenes persistentes y el cifrado en tránsito en las capas de almacenamiento y red.

Cumplimiento normativo y auditorías Ofrece herramientas y funciones que ayudan a las organizaciones a cumplir los requisitos de cumplimiento y a realizar auditorías periódicas de sus prácticas de protección de datos.

Recuperación ante desastres se integra con las funciones de replicación de ONTAP para ofrecer soluciones sólidas de recuperación ante desastres, garantizando la continuidad del negocio incluso ante eventos catastróficos.

Para obtener información detallada sobre Trident Protect, consulta la "Documentación de Trident Protect".

NetApp modelos de hardware y protocolos compatibles con ONTAP

El software NetApp ONTAP se ejecuta en diversos modelos de hardware, cada uno diseñado para satisfacer distintos requisitos de rendimiento y capacidad. Trident amplía sus sólidas capacidades para dar soporte a varios sistemas NetApp ONTAP, incluidos All Flash FAS (AFF), All SAN Array (ASA) y ASA R2. Este soporte garantiza que las organizaciones puedan aprovechar todo el potencial de sus soluciones de almacenamiento de alto rendimiento en entornos contenedorizados.

Sistemas All Flash FAS (AFF) Los sistemas AFF ofrecen un rendimiento excepcional y baja latencia, lo que los hace ideales para aplicaciones de demanda elevada.

Todos los sistemas de matriz SAN (ASA) Los sistemas ASA están diseñados para entornos SAN dedicados, ofreciendo un rendimiento y una fiabilidad robustos.

Sistemas ASA R2 Los sistemas ASA R2 ofrecen funciones y prestaciones mejoradas para entornos SAN.

AFX NetApp AFX es una arquitectura de almacenamiento all-flash desagregada y diseñada específicamente para cargas de trabajo de IA empresarial. Ofrece un NAS de alto rendimiento, lo que permite escalar de forma independiente la capacidad de cálculo y el almacenamiento. NetApp AFX los sistemas de almacenamiento se diferencian de otros sistemas basados en ONTAP (ASA, AFF y FAS) en la implementación de su capa de almacenamiento. AFX utiliza un sistema de archivos distribuido que permite un alto rendimiento y escalabilidad, mientras que otros sistemas basados en ONTAP utilizan una arquitectura de almacenamiento tradicional.

1. Protocolos compatibles con AFF: NFS, CIFS/SMB, iSCSI, FC, NVMe/TCP
2. Protocolos compatibles con ASA: iSCSI, FC, NVMe/TCP
3. Protocolos compatibles con ASA R2: iSCSI, FC, NVMe/TCP
4. Protocolos compatibles con AFX: a partir de Trident 25.10, solo se admite el protocolo NFS; el protocolo SMB no es compatible.

Controladores Trident para almacenamiento ONTAP

Trident incluye los siguientes controladores CSI, cada uno de los cuales es capaz de aprovisionar un volumen en el almacenamiento backend.

Para los protocolos NAS en los modelos de hardware compatibles

1. ontap-nas: Un PVC se asigna a un PV, que es un volumen dedicado de FlexVol.
2. ontap-nas-economy: Cada PV aprovisionado es un qtree, con un número configurable de qtrees por volumen FlexVol (el valor por defecto es 200, configurable entre 50 y 300).

   Cada PVC se asigna a un PV, que es un qtree en un volumen compartido de FlexVol.

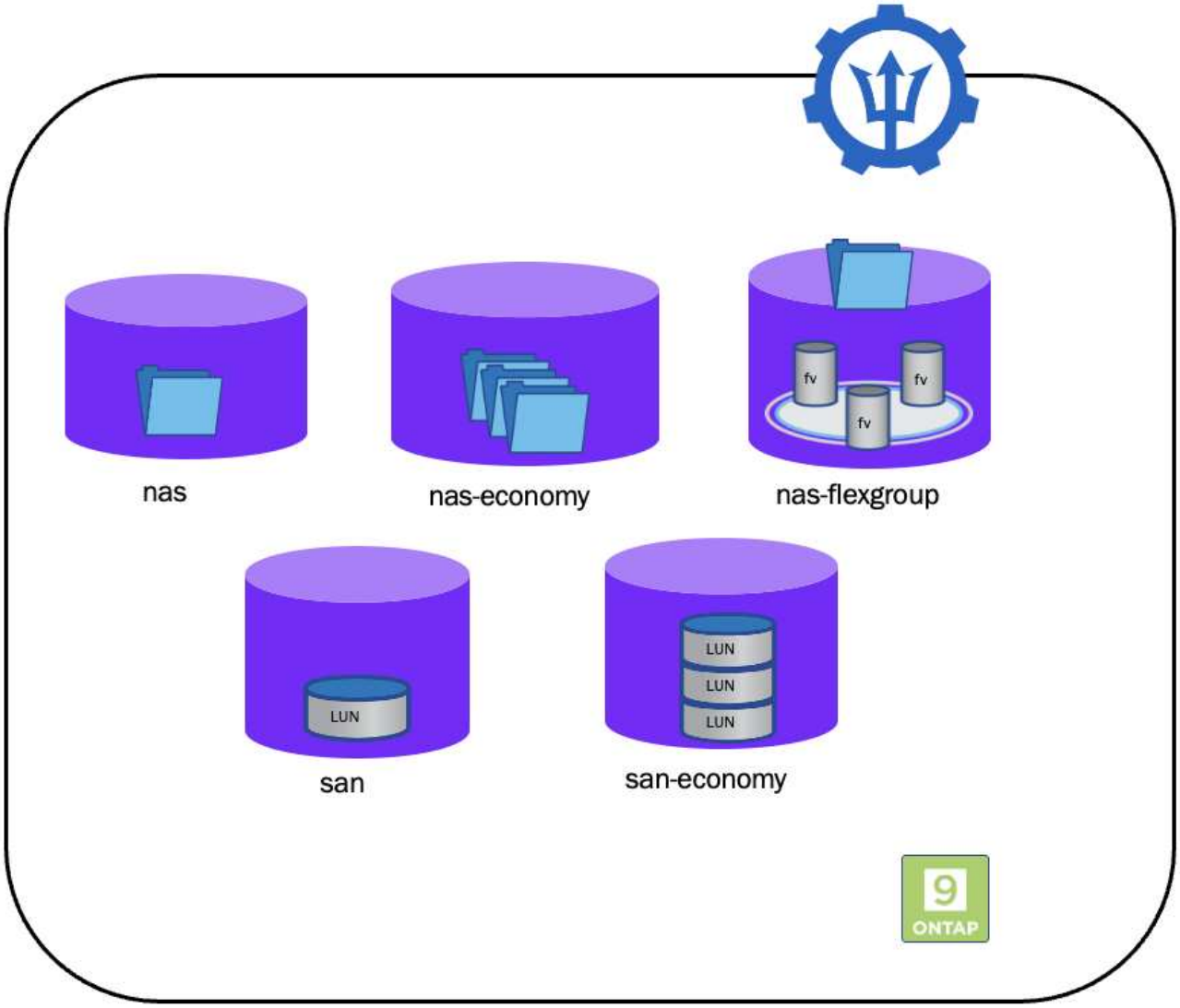
> Puedes colocar todos los discos de las máquinas virtuales como qtrees en un único volumen. Sin embargo, ten en cuenta que las funciones de Snapshots, Clones y Replication no son compatibles con Trident. Cuando esas funciones son gestionadas por el administrador de almacenamiento, no son granulares a nivel de PV.

1. ontap-nas-flexgroup: Cada PV aprovisionado es un ONTAP completo FlexGroup, y se utilizan todos los agregados asignados a un SVM.

Para los protocolos SAN en los modelos de hardware compatibles

1. ontap-san: Un PVC se asigna a un PV, que es un LUN en un volumen dedicado de FlexVol.
2. ontap-san-economy: Cada PVC se asigna a un PV, que es un LUN en un volumen compartido FlexVol. Puedes tener varios LUN en el volumen compartido FlexVol (el valor predeterminado es 100, configurable entre 50 y 200 LUN/PV por volumen FlexVol).

> Puedes colocar todos los discos de las máquinas virtuales como LUN en un único volumen. Sin embargo, este controlador admite Snapshots y Clones, pero no admite Replication. Cuando Replication la gestiona el administrador de almacenamiento, no es granular a nivel de PV.



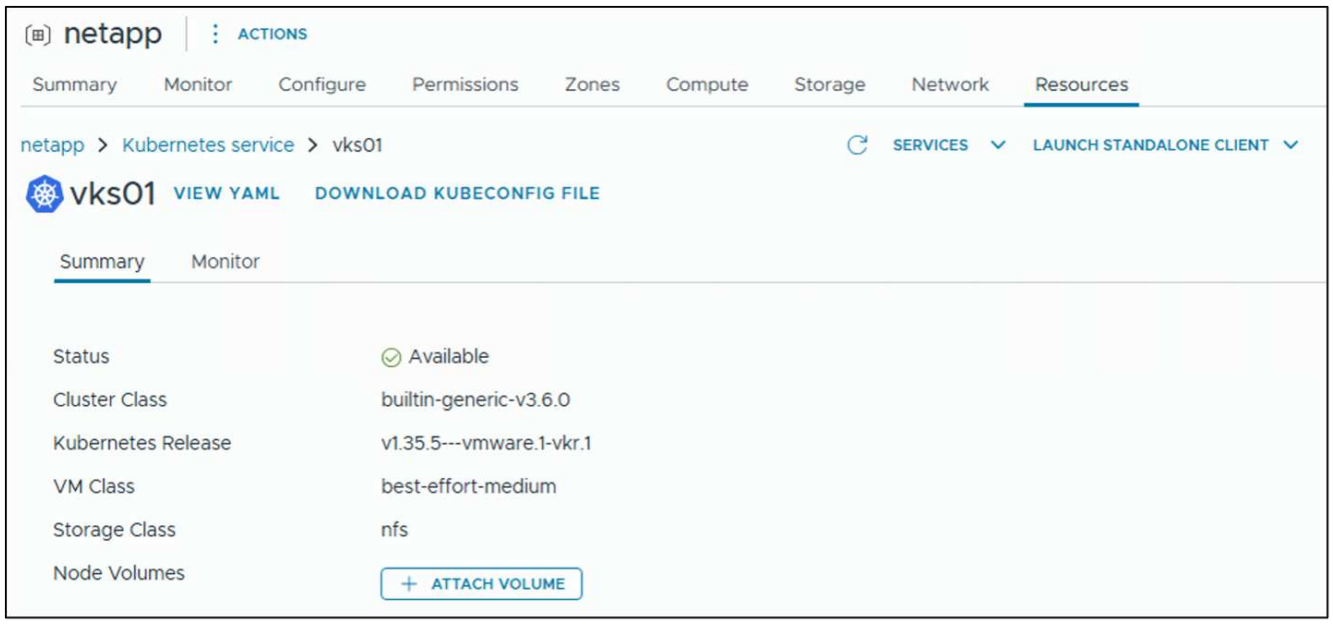
Para consultar la lista completa de funciones disponibles a través de los controladores de Trident y aquellas que debe configurar el administrador de almacenamiento, consulta la sección "Controladores de backend de ONTAP" en la documentación de Trident.

Instala Trident de NetApp en un clúster de VKS

Instala NetApp Trident como el aprovisionador de almacenamiento dinámico en tu clúster de Kubernetes Service de vSphere para integrar el almacenamiento NetApp ONTAP con tus cargas de trabajo de Kubernetes.

Antes de empezar

- Asegúrate de que tu clúster ONTAP esté configurado y conectado a tu entorno de VMware Kubernetes.
- Asegúrate de que tu clúster VKS tenga instaladas las herramientas iSCSI o NVMe si tus cargas de trabajo van a utilizar esos protocolos para almacenamiento.
- Asegúrate de tener `kubectl` instalado en un equipo desde el que puedas conectarte al clúster VKS utilizando el archivo kubeconfig.



Pasos

1. Instala el operador de Trident usando un chart de Helm siguiendo las instrucciones de la documentación de Trident:

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

   Antes de ejecutar la instalación de Helm, crea y etiqueta el `trident` espacio de nombres. La `enforce=privileged` etiqueta es obligatoria porque Trident ejecuta cargas de trabajo con privilegios. Sin ella, el controlador de admisión de seguridad de pods de Kubernetes impedirá que se inicien los pods de Trident.

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

   ▼ **Mostrar ejemplo**

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

   Para habilitar el registro de depuración, añade `--set tridentDebug=true`:

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

   ⓘ También puedes instalar Trident manualmente mediante el Trident Operator. Consulta los procedimientos que se indican en la documentación de Trident: "Implementa manualmente el operador Trident"

Asegúrate de haber etiquetado el `trident` espacio de nombres con las etiquetas de seguridad de pod necesarias antes de instalar Trident manualmente.

2. Confirma que todos los pods Trident están en ejecución en el clúster.

▼ **Mostrar ejemplo**

```
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$
```

Si los pods de Trident no se inician con un error de admisión de `PodSecurity`, es posible que las etiquetas de seguridad de los pods no se hayan aplicado al `trident` espacio de nombres antes de la instalación. Usa `--overwrite` para volver a aplicarlas y luego verifica que los pods se inicien.

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

Prepara el backend de almacenamiento y los archivos de configuración de StorageClass para tu entorno

1. Configura un backend de Trident para ONTAP definiendo los datos de conexión necesarios y los parámetros de almacenamiento.
2. Define clases de almacenamiento en Kubernetes que se corresponden con los backends de almacenamiento NetApp. Estas clases especifican parámetros como las características de rendimiento y las políticas de replicación.

   Define los backends y las clases de almacenamiento de Trident para los siguientes protocolos según lo requieran tus cargas de trabajo:

   - NAS (controlador ontap-nas)
   - NAS Economy (controlador ontap-nas-economy)
   - SAN (controlador ontap-san) para los protocolos iSCSI, FC o NVMe
   - SAN Economy (ontap-san-economy driver) para iSCSI

   NAS

   Crea un TridentBackendConfig y un StorageClass para ONTAP NAS para habilitar el aprovisionamiento de almacenamiento persistente basado en NFS. La configuración del backend incluye credenciales almacenadas en un Kubernetes Secret y hace referencia a tu ONTAP SVM y a tu LIF de gestión.

Ejemplo de secreto de backend y archivo de configuración de backend con autenticación mediante nombre de usuario y contraseña (guardar como `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
```

Ejemplo de secreto de backend y archivo de configuración de backend con autenticación mediante certificado de cliente (guardar como `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 definición (guarda como `sc-nas.yaml`):

`mountOptions` es un parámetro opcional que especifica opciones de montaje adicionales para volúmenes NFS. La opción `nconnect` permite múltiples conexiones TCP en paralelo para un único montaje NFS, lo que puede mejorar el rendimiento en cargas de trabajo de alto rendimiento. El valor por defecto es 1, pero se puede aumentar hasta el máximo permitido por el sistema ONTAP.

ONTAP admite hasta 16 conexiones para un único montaje NFS en un cliente compatible con nconnect. Trident transmite las opciones de montaje directamente a los nodos de trabajo de Kubernetes, así que elige un valor que sea compatible tanto con el cliente de NFS del nodo de trabajo como con ONTAP; en el siguiente ejemplo se utilizan cuatro conexiones.

```
# 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
```

Crea un TridentBackendConfig y un StorageClass para el controlador ONTAP NAS Economy para habilitar el aprovisionamiento de almacenamiento persistente basado en NFS usando qtrees. La configuración del backend incluye credenciales almacenadas en un Secret de Kubernetes y hace referencia a tu SVM de ONTAP y a tu LIF de gestión.

Ejemplo de secreto de backend y archivo de configuración de backend con autenticación mediante nombre de usuario y contraseña (guardar como `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 definición (guarda como `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
```

Crea un TridentBackendConfig y un StorageClass para ONTAP SAN con iSCSI para habilitar el aprovisionamiento de almacenamiento en bloques basado en iSCSI. La configuración del backend utiliza el controlador ontap-san e incluye credenciales almacenadas en un secreto de Kubernetes. El `sanType` parámetro adopta por defecto el protocolo iSCSI si se omite.

Secreto del backend y archivo de configuración del backend con autenticación mediante nombre de usuario y contraseña (guardar como `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
```

Secreto del backend y archivo de configuración del backend con autenticación mediante certificado de cliente (guardar como `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 definición (guarda como `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
```

Crea un TridentBackendConfig y un StorageClass para ONTAP SAN con NVMe para habilitar el aprovisionamiento de almacenamiento en bloques basado en NVMe. La configuración del backend utiliza el controlador ontap-san e incluye credenciales almacenadas en un Secret de Kubernetes. El parámetro `sanType` debe establecerse en `nvme`.

Secreto del backend y archivo de configuración del backend con autenticación mediante nombre de usuario y contraseña (guardar como `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
```

Secreto del backend y archivo de configuración del backend con autenticación mediante certificado de cliente (guardar como `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 definición (guarda como `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
```

Consulta la documentación de Trident para ver más ejemplos de archivos YAML e información sobre los modos de acceso y los modos de volumen compatibles con los controladores SAN y NAS.

- "Ejemplos con controladores NAS"
- "Ejemplos con controladores SAN"

Crea un archivo de configuración de VolumeSnapshotClass

Crea una definición de VolumeSnapshotClass. Esta configuración habilita las operaciones basadas en instantáneas para los volúmenes persistentes.

Definición de VolumeSnapshotClass (guarda como `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
```

Aplica los archivos de configuración a tu clúster

Aplica los archivos de configuración que creaste en los pasos anteriores al clúster de vSphere Kubernetes Service usando `kubectl`. Esto crea los secretos, las configuraciones de backend, las clases de almacenamiento y la clase de instantáneas necesarios en tu clúster.

Pasos

1. Aplica los archivos TridentBackendConfig y StorageClass para cada protocolo que configuraste.

```
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. Verifica que los recursos se hayan creado correctamente.

   Revisa los objetos TridentBackendConfig:

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

   Comprueba los objetos de StorageClass:

```
kubectl get storageclass
```

   Revisa VolumeSnapshotClass:

```
kubectl get volumesnapshotclass
```

Configura las clases predeterminadas de almacenamiento y de instantáneas de Trident

Configura la StorageClass y la VolumeSnapshotClass de Trident como valores predeterminados en el clúster del servicio de Kubernetes de vSphere.

Pasos

1. Configura el StorageClass predeterminado de Trident.

   Configura un StorageClass respaldado por Trident como predeterminado del clúster para que las PersistentVolumeClaims lo usen automáticamente cuando no se especifique ninguna clase de almacenamiento.

   Asegúrate de que solo una StorageClass esté configurada como predeterminada. Si otra StorageClass ya está configurada como predeterminada, establece su anotación en `false`.

   Desde la CLI:

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

2. Establece el Trident VolumeSnapshotClass predeterminado.

   Configura un VolumeSnapshotClass compatible con Trident como valor predeterminado del clúster para habilitar las operaciones basadas en instantáneas para los volúmenes persistentes. Esto asegura que VolumeSnapshots use automáticamente el controlador CSI de Trident cuando no se especifique ninguna clase de instantánea.

   Asegúrate de que solo haya un VolumeSnapshotClass configurado como predeterminado. Si ya hay otro VolumeSnapshotClass configurado como predeterminado, establece su anotación en `false`.

   Desde la CLI:

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

Utiliza las reclamaciones de volumen persistente de Kubernetes para solicitar almacenamiento para las cargas de trabajo

Trident aprovisiona automáticamente los volúmenes necesarios desde el almacenamiento backend NetApp. Consulta "Implementa cargas de trabajo con almacenamiento persistente" para ver ejemplos de cómo implementar una carga de trabajo con PVCs en un clúster de VKS.

Implementa una aplicación en contenedores con almacenamiento persistente de NetApp

Implementa una aplicación en contenedores en tu clúster del servicio Kubernetes de vSphere utilizando almacenamiento persistente NetApp ONTAP. Los siguientes ejemplos muestran cómo implementar una base de datos PostgreSQL utilizando almacenamiento

NAS (ontap-nas) o SAN iSCSI (ontap-san).

La siguiente configuración YAML establece POSTGRES_USER y POSTGRES_PASSWORD directamente en el manifiesto. En entorno de producción, se recomienda utilizar los Secrets de Kubernetes para gestionar información confidencial, como las credenciales de la base de datos.

Implementa PostgreSQL con almacenamiento NAS (controlador ontap-nas)

Puedes implementar una aplicación en contenedor de PostgreSQL respaldada por un volumen persistente NFS aprovisionado por el controlador ontap-nas. El siguiente ejemplo muestra cómo crear un PersistentVolumeClaim (PVC) y una Deployment para PostgreSQL.

```
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
```

Implementa PostgreSQL con almacenamiento SAN (controlador ontap-san iSCSI)

Utiliza el siguiente YAML para implementar una aplicación en contenedor de PostgreSQL respaldada por un volumen persistente iSCSI aprovisionado por el controlador ontap-san. La `Recreate` estrategia de implementación garantiza que el pod se termine por completo antes de que se inicie uno nuevo, lo cual es necesario para los volúmenes iSCSI ReadWriteOnce y evita la corrupción de datos durante los reinicios de los pods. Esta configuración admite la persistencia de datos tras la eliminación y recreación de pods, y es compatible con las instantáneas de volumen de Trident y con las operaciones de backup y restauración.

```
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
---
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
```

Valida la implementación de la base de datos

Tras implementar la aplicación, comprueba que los pods se encuentren en estado de ejecución y que la base de datos esté operativa. Utiliza los siguientes comandos para comprobar el estado de los pods, los PVC y los servicios. Asegúrate de que los pods estén en ejecución y de que los PVC estén vinculados a los volúmenes correspondientes.

```
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:~$
```

Dependiendo del protocolo utilizado, el almacenamiento backend es un volumen, qtree o LUN. Puedes verificar el almacenamiento aprovisionado en el sistema ONTAP describiendo el PersistentVolume (PV) para obtener el nombre interno del volumen y luego usarlo en los comandos de la CLI de ONTAP para obtener los detalles del almacenamiento. Usa el siguiente comando para describir el PV y obtener el nombre interno del volumen:

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

```
vksbastion@vks:~/demo-vks$ kubectl describe pv/pvc-4782b054-5aff-49ae-bc40-08cdfd30db77
Name:            pvc-4782b054-5aff-49ae-bc40-08cdfd30db77
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                 volume.kubernetes.io/provisioner-deletion-secret-name:
                 volume.kubernetes.io/provisioner-deletion-secret-namespace:
Finalizers:      [external-provisioner.volume.kubernetes.io/finalizer kubernetes.io/pv-protection external-attacher/csi-trident-ne
tapp-io]
StorageClass:    sc-nas
Status:          Bound
Claim:           default/postgres-pvc
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        10Gi
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            csi.trident.netapp.io
    FSType:
    VolumeHandle:      pvc-4782b054-5aff-49ae-bc40-08cdfd30db77
    ReadOnly:          false
    VolumeAttributes:      backendUUID=8d6b9527-91dc-4847-b5cc-6aa7307cf1ba
                           internalName=trident_pvc_4782b054_5aff_49ae_bc40_08cdfd30db77
                           name=pvc-4782b054-5aff-49ae-bc40-08cdfd30db77
                           protocol=file
                           storage.kubernetes.io/csiProvisionerIdentity=1787189141467-7183-csi.trident.netapp.io
Events:                <none>
```

Figura 1. Mostrar ejemplo

```
vksbastion@vks:~/vks02$ kubectl describe pv/pvc-b992a5cf-1127-4b6c-a775-80ab64e4b536
Name:            pvc-b992a5cf-1127-4b6c-a775-80ab64e4b536
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                 volume.kubernetes.io/provisioner-deletion-secret-name:
                 volume.kubernetes.io/provisioner-deletion-secret-namespace:
Finalizers:      [external-provisioner.volume.kubernetes.io/finalizer kubernetes.io/pv-protection external-attacher/csi-trident-ne
tapp-io]
StorageClass:    sc-iscsi
Status:          Bound
Claim:           postgres/postgres-block-pvc
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        20Gi
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            csi.trident.netapp.io
    FSType:            ext4
    VolumeHandle:      pvc-b992a5cf-1127-4b6c-a775-80ab64e4b536
    ReadOnly:          false
    VolumeAttributes:      backendUUID=01dd718d-04a4-4c43-95fb-dc52f56e8624
                           internalName=trident_pvc_b992a5cf_1127_4b6c_a775_80ab64e4b536
                           name=pvc-b992a5cf-1127-4b6c-a775-80ab64e4b536
                           protocol=block
                           storage.kubernetes.io/csiProvisionerIdentity=1786638282639-3882-csi.trident.netapp.io
Events:                <none>
```

Copia el nombre del volumen interno de la salida y ejecuta el siguiente comando para obtener los detalles de almacenamiento desde la CLI de ONTAP.

Para volúmenes NAS:

```
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
```

En el caso de los LUN de SAN, ejecuta el siguiente comando para obtener los detalles del LUN desde la CLI de ONTAP:

```
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::>
```

Si has utilizado las opciones de montaje `nconnect` en la clase de almacenamiento NAS o NAS Economy, puedes comprobar el número de conexiones TCP activas desde el nodo de trabajo al servidor de almacenamiento.

En primer lugar, enumera las configuraciones del backend de Trident para encontrar el nombre del backend, luego descríbelo para obtener el nombre del vserver y el LIF de datos:

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

A continuación, inicia sesión en el clúster ONTAP y ejecuta el siguiente comando, sustituyendo el LIF de datos que aparece en la salida anterior:

```
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.
```

Figura 2. Mostrar ejemplo

Después de que los pods estén en estado de ejecución, conéctate a la base de datos y verifica las operaciones de datos.

Pasos

1. Reenvía el puerto del servicio PostgreSQL a tu ordenador local:

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

2. Desde otro terminal, conéctate a la base de datos mediante psql:

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

3. Crea una base de datos y una tabla, luego llena la tabla con datos:

```
CREATE DATABASE employee;
```

Cambia a la nueva base de datos (metacomando de psql):

```
\c employee
```

Crea la tabla, inserta una fila y consulta los resultados:

```
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;
```

Demostración en vídeo

El siguiente vídeo muestra cómo instalar Trident en un clúster de Kubernetes de vSphere y cómo implementar una base de datos PostgreSQL con almacenamiento persistente usando Trident.

Poner en marcha cargas de trabajo en un clúster de vSphere Kubernetes Service respaldado por almacenamiento persistente ONTAP usando Trident

Protección de datos

Haz copias de seguridad y restaura aplicaciones en contenedores en un clúster de vSphere Kubernetes Service usando NetApp Console

Haz copias de seguridad y restaura aplicaciones en contenedores en un clúster de vSphere Kubernetes Service (VKS) usando NetApp Console. Este procedimiento cubre cómo crear y gestionar operaciones de backup y restauración a través de Backup and Recovery Service mediante NetApp Console, usando almacenamiento de objetos ONTAP S3 como destino de la copia de seguridad.

NetApp Console ofrece gestión centralizada de servicios de almacenamiento y datos tanto en entornos locales como en la nube. Ofrece control unificado, simplicidad operativa y gestión segura. Hay varios servicios de datos y funciones de gestión a los que puedes acceder desde la interfaz de usuario en "console.netapp.com". Para obtener todos los detalles sobre NetApp Console, consulta "NetApp documentación de la consola".

Esta sección se centra en el servicio de datos NetApp Backup and Recovery para cargas de trabajo de Kubernetes, específicamente aplicaciones en contenedores en clústeres de Kubernetes Service de vSphere. El servicio NetApp Backup and Recovery te permite descubrir, gestionar, crear y asociar políticas de protección con tus aplicaciones de Kubernetes usando una sola interfaz. Para una guía completa, consulta "NetApp Backup and Recovery documentación".

Flujo de trabajo de backup y recuperación

1. **Asegúrate de tener un agente de Console**

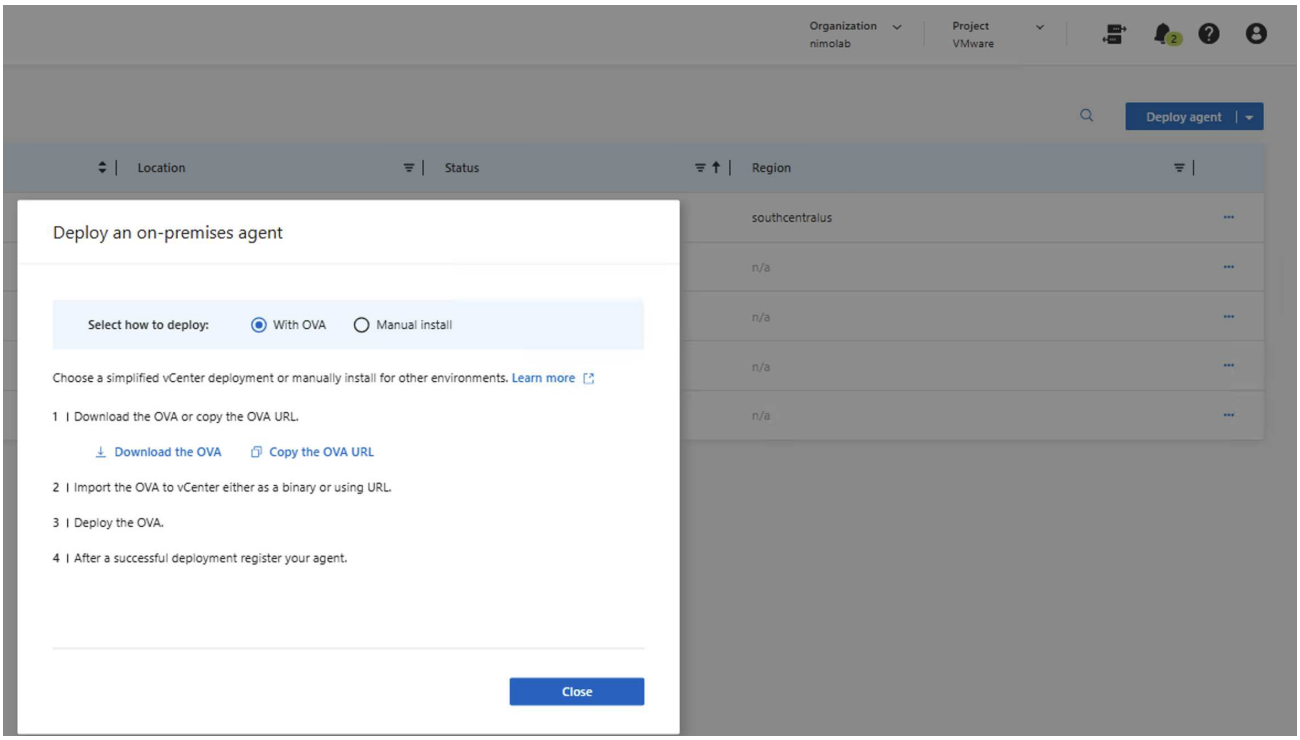
   Asegúrate de que tienes un agente de Console implementado en el entorno en el que se ejecuta tu clúster VKS. Para obtener más información sobre cómo instalar un agente de Console, consulta "NetApp Documentación de configuración de la consola".

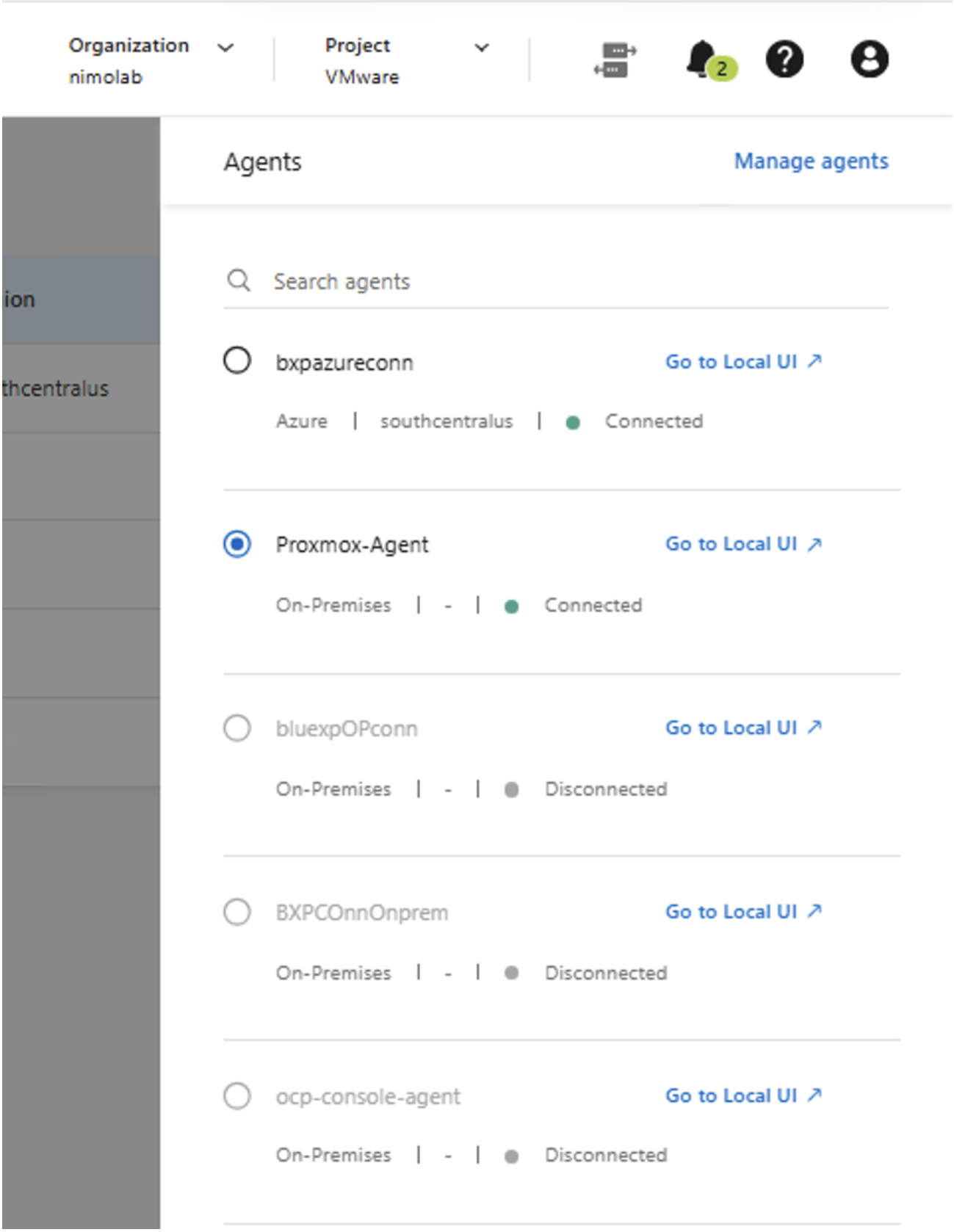
   ▼ **Mostrar ejemplo**

   En NetApp Console, haz clic en Deploy Agent como se muestra en la captura de pantalla y despliega el agente en el entorno donde se está ejecutando el clúster de VKS.

   En este ejemplo, selecciona **On-Premises > With OVA**, copia la URL del archivo OVA y utiliza esa URL en vCenter para implementar el agente de Console.

   Regístralo utilizando la dirección IP del agente en la URL. Consulta "la documentación" para ver instrucciones detalladas. Una vez registrado el agente, podrás verlo en NetApp Console, como se muestra en la captura de pantalla siguiente.


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. **Añade el clúster ONTAP a NetApp Console y activa Backup and Recovery**

   El clúster ONTAP principal debe añadirse a NetApp Console y habilitar el servicio de Backup and Recovery en él antes de que se puedan proteger las cargas de trabajo. Para ver las instrucciones, consulta "Configura NetApp Backup and Recovery".

3. **En NetApp Console, descubre el clúster de Kubernetes de vSphere**

▼ **Mostrar ejemplo**

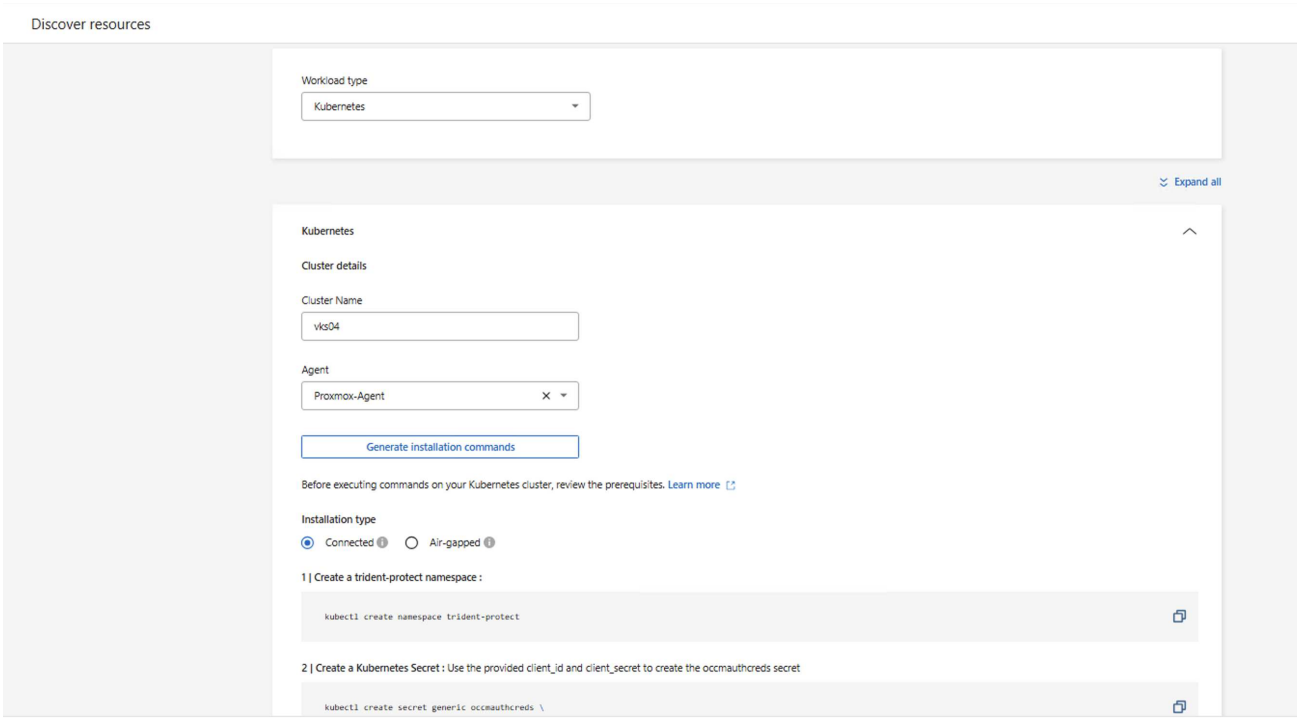
Este paso requiere que el clúster de Kubernetes de vSphere esté en funcionamiento y que el agente de Console esté implementado en el mismo entorno. Sigue estos pasos para descubrir el clúster de Kubernetes de vSphere en NetApp Console.

a. En NetApp Console, selecciona **Inventory > Kubernetes** e introduce los detalles del clúster de Kubernetes de vSphere.

b. Selecciona el agente implementado en el mismo entorno donde se encuentra el clúster de Kubernetes de vSphere.

c. Copia los comandos proporcionados para instalar Trident Protect y ejecútalos desde un equipo que esté conectado a tu clúster de Kubernetes de vSphere.

d. Una vez instalado correctamente Trident Protect, haz clic en **Descubrir**. NetApp Console detecta el clúster de Kubernetes de vSphere y todos sus recursos a través del agente, y los muestra en la interfaz de usuario.



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
```

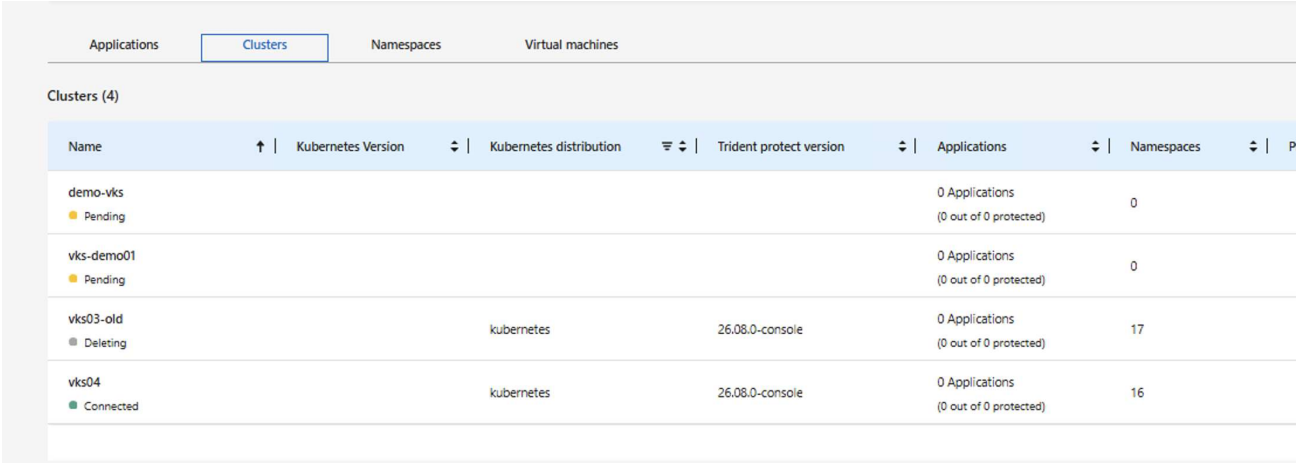
3 | Add/update helm repo :

```
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$
```



4. **Configurar ONTAP S3 como destino de copia de seguridad**

   En este ejemplo, utilizamos el almacenamiento de objetos ONTAP S3 como destino de copia de seguridad. También puedes usar AWS S3, Azure Blob, Google Cloud Storage o StorageGRID como destinos de copia de seguridad.

   Para obtener más información, consulta "NetApp documentación de la consola".

   Para añadir ONTAP S3 como destino de copia de seguridad en NetApp Console, necesitarás la siguiente información de tu clúster ONTAP.

```
S3 Endpoint:
Access Key:
Secret Key:
Bucket Name:
```

Para obtener estos valores, haz lo siguiente en tu clúster ONTAP usando System Manager.

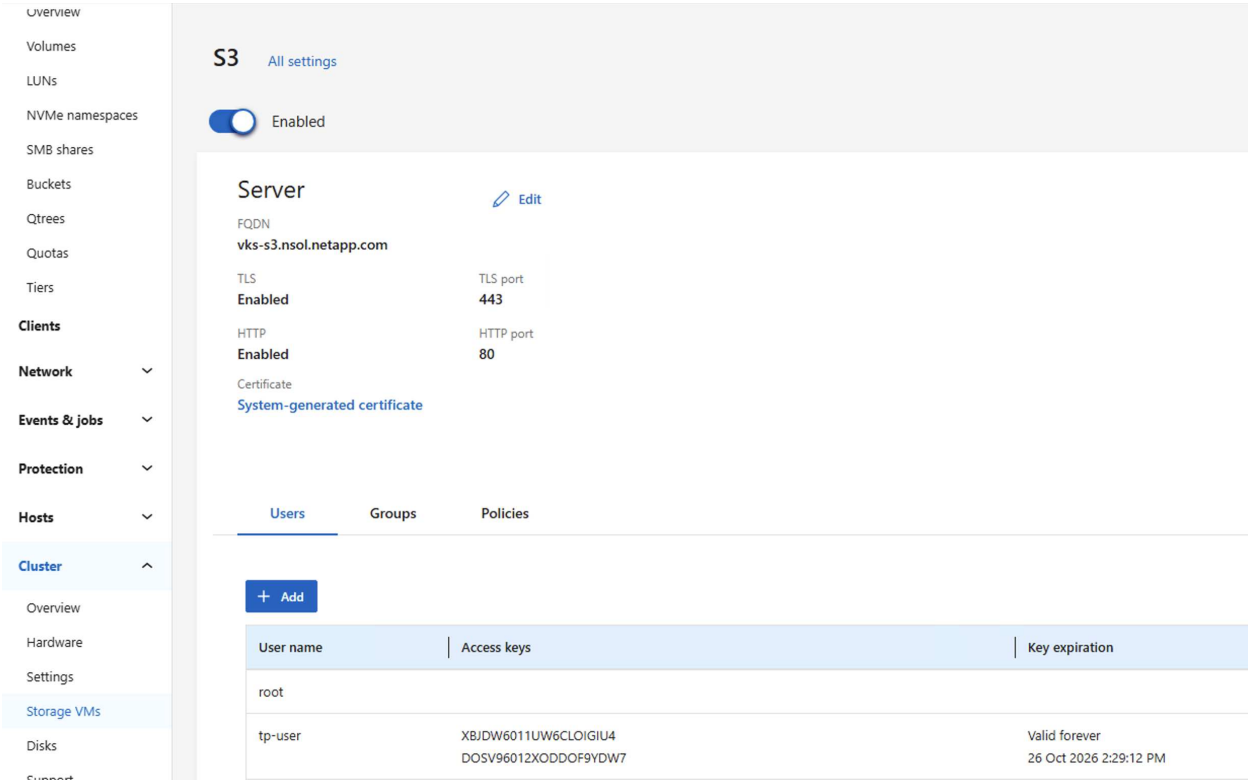
a. Crea una SVM compatible con S3 en el clúster ONTAP para almacenar los datos de copia de seguridad. Anota la URL del endpoint S3 de esta SVM.

b. En la configuración de S3 de la SVM, crea un usuario y asígnale el acceso requerido. Anota la clave de acceso y la clave secreta de este usuario.

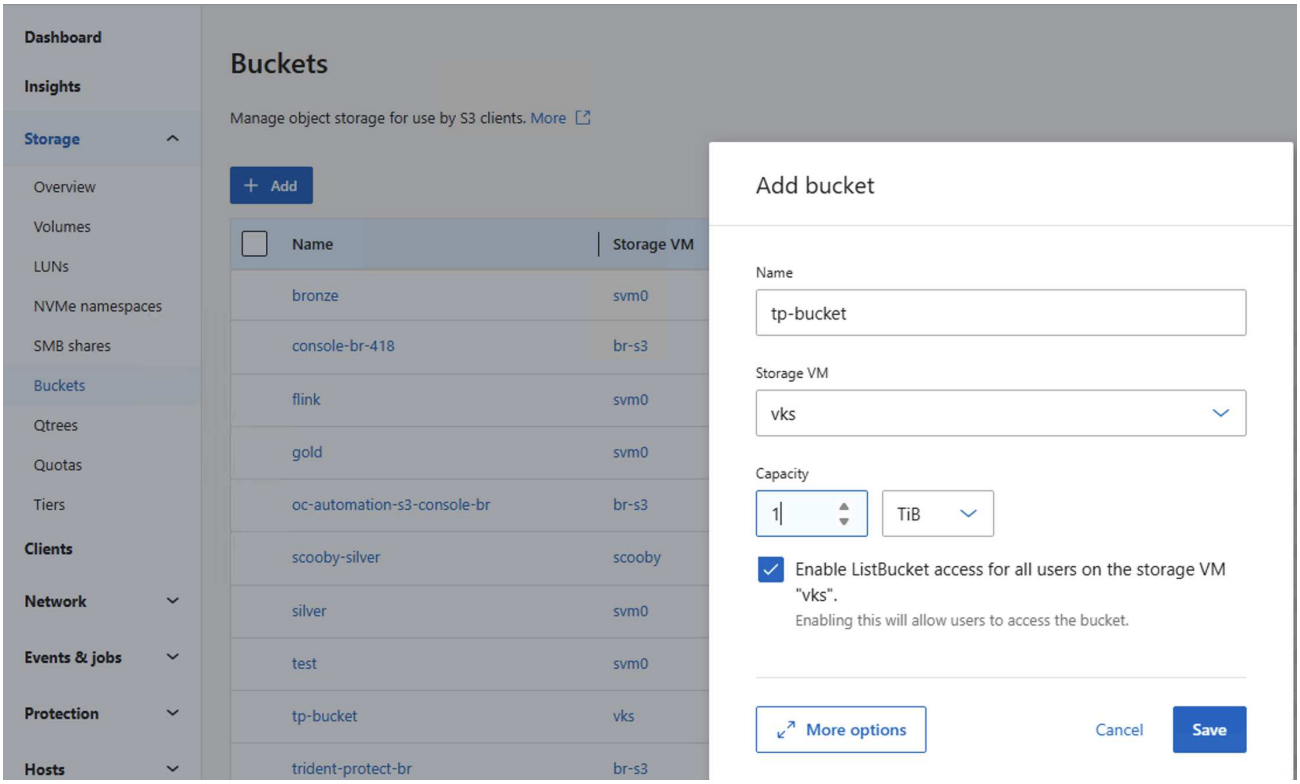
> Trident Protect requiere que el usuario de S3 tenga al menos `PutObject`, `GetObject`, `ListBucket` y `DeleteObject` permisos. Sin estos permisos, las operaciones de backup y restauración de AppVault fallan con errores de acceso denegado. Para más detalles, consulta "Documentación de Trident Protect AppVault".

▼ **Mostrar ejemplo**

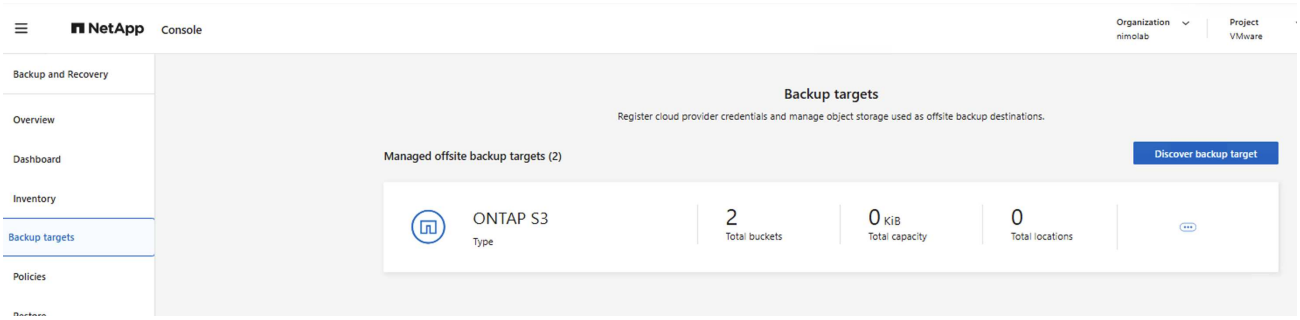
i. **Crea un usuario de S3 y descarga la clave de acceso y la clave secreta**

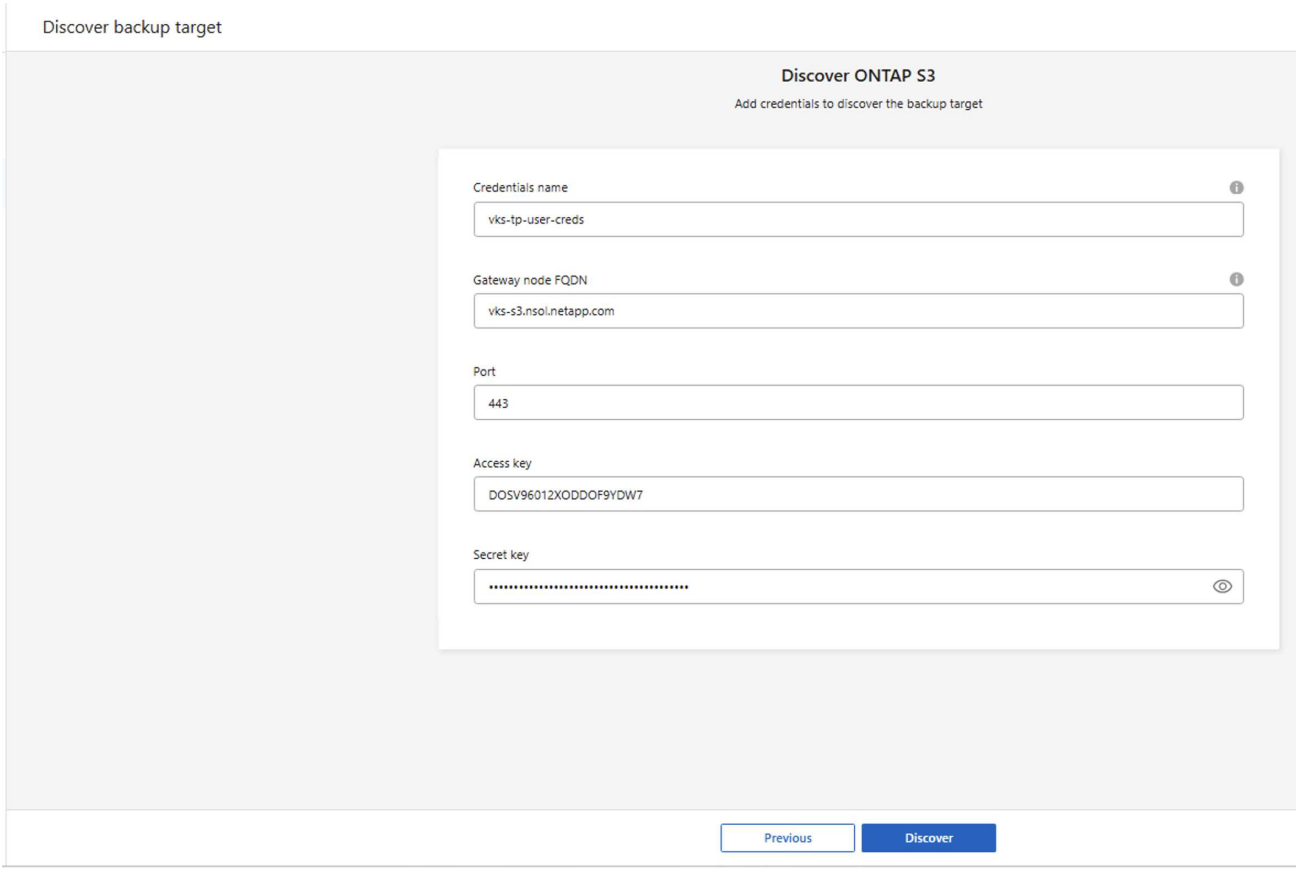


A. Crea un depósito en el SVM con S3 habilitado. Anota el nombre del depósito para utilizarlo al añadir el almacenamiento de objetos ONTAP S3 como destino de copia de seguridad en NetApp Console.



Añade el almacenamiento de objetos ONTAP S3 como destino de copia de seguridad en la NetApp Console usando el endpoint S3, la clave de acceso, la clave secreta y el nombre del bucket.





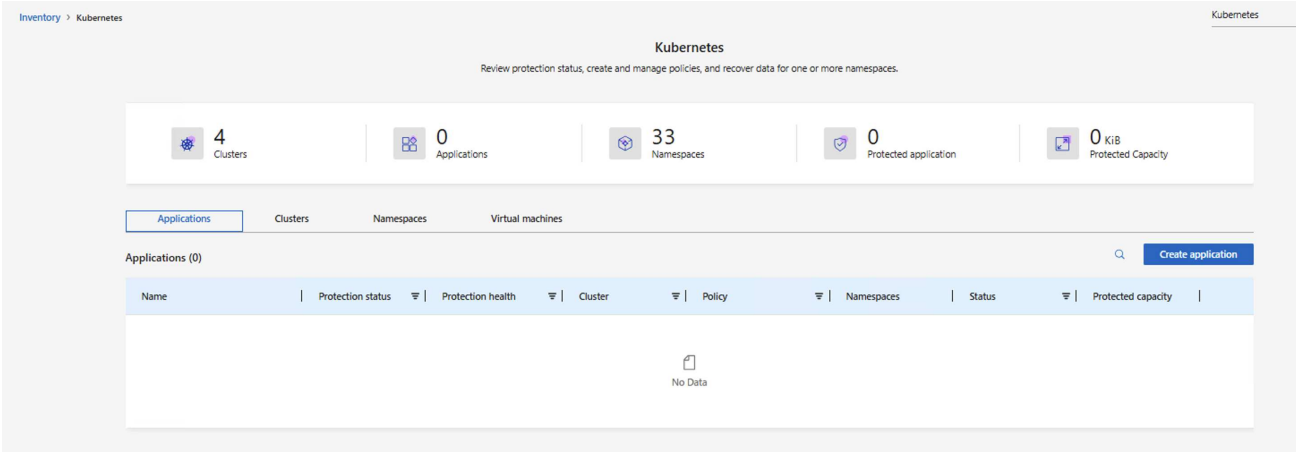
5. **En NetApp Console, crea una aplicación para las cargas de trabajo de contenedores y protégela mediante copias de seguridad**

> Para crear y proteger una aplicación de Kubernetes, necesitas el rol de super admin de Backup and Recovery o el rol de backup admin de Backup and Recovery. Para restaurar una aplicación de Kubernetes, necesitas el rol de super admin de Backup and Recovery o el rol de restore admin de Backup and Recovery.

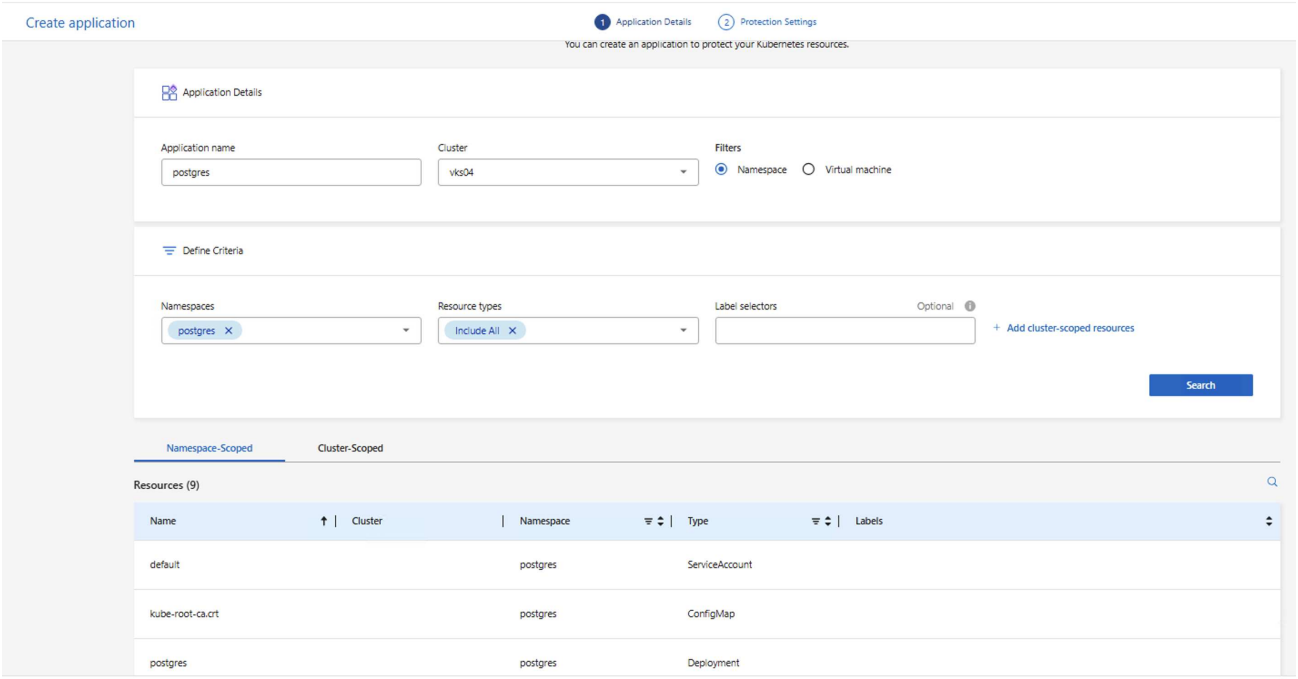
▼ **Mostrar ejemplo**

En este paso, crea una aplicación en NetApp Console para las aplicaciones en contenedores de tu clúster de Kubernetes de vSphere que necesitan ser protegidas. Puedes usar espacios de nombres para crear una aplicación que proteja todas las aplicaciones en contenedores de un espacio de nombres en el clúster de Kubernetes de vSphere.

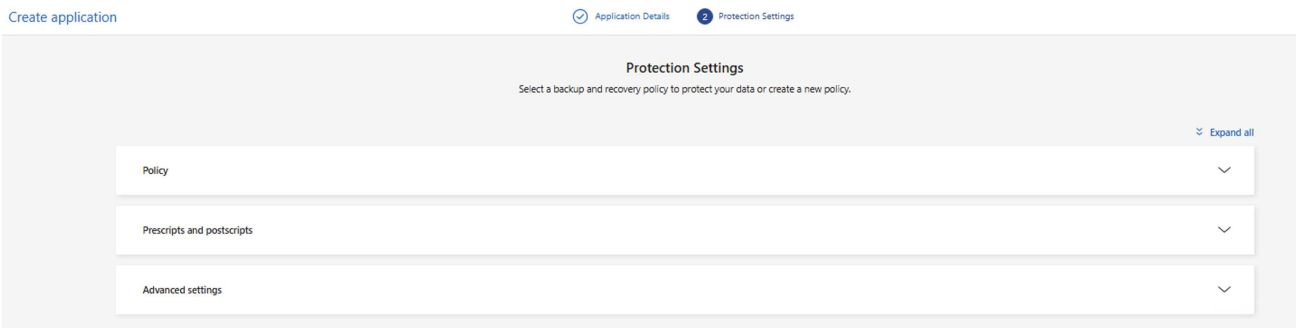
También necesitarás una política de protección para asociar con la aplicación. Puedes crear una nueva política de protección o usar una existente.



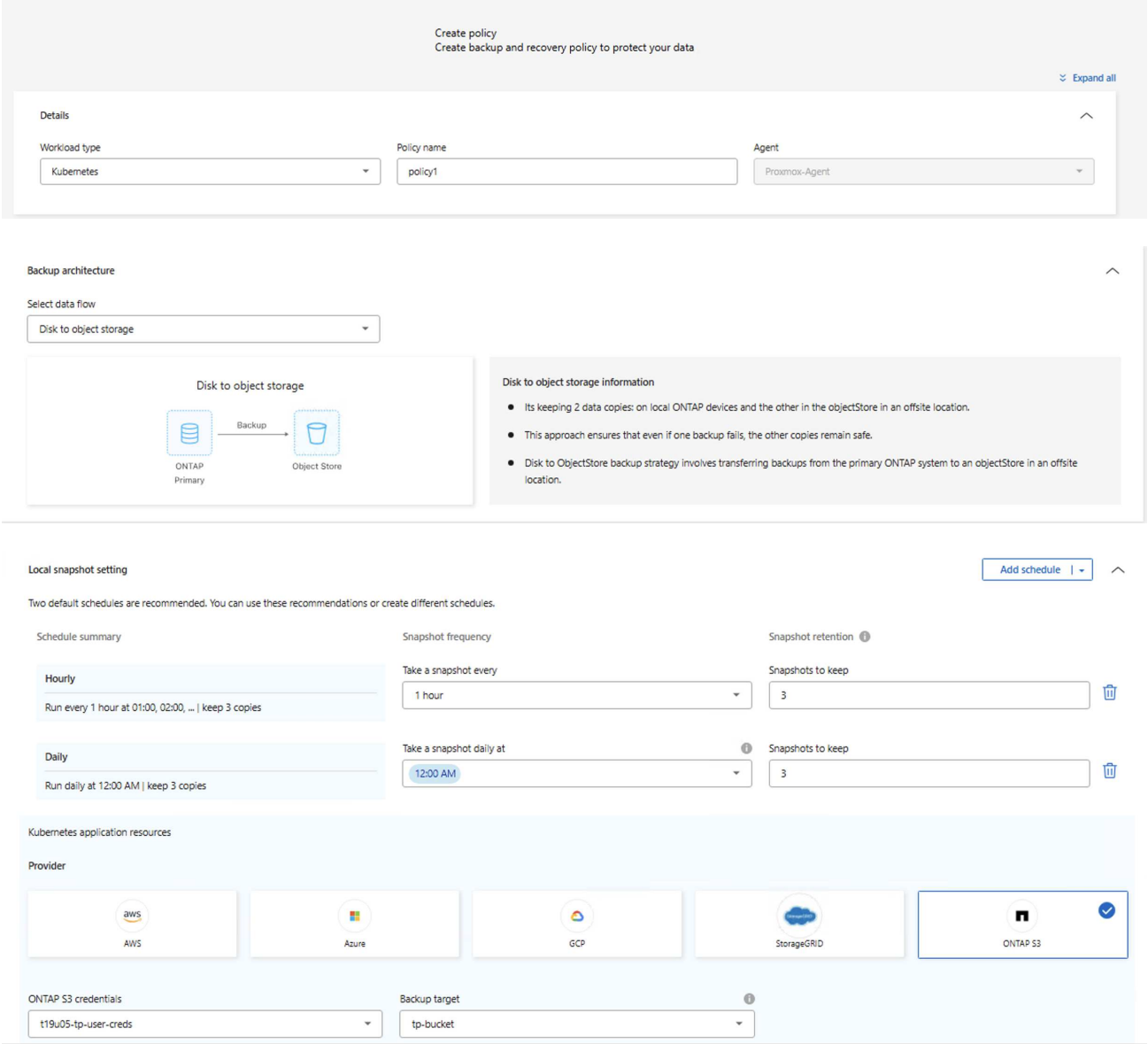
Proporciona los detalles de la aplicación y define los criterios para la aplicación. En este ejemplo, la aplicación se crea para todas las aplicaciones de contenedor en un espacio de nombres en el clúster de Kubernetes vSphere. Haz clic en **Buscar** para ver la lista de recursos con ámbito de espacio de nombres que forman parte de la aplicación. Haz clic en **Siguiente** para continuar.

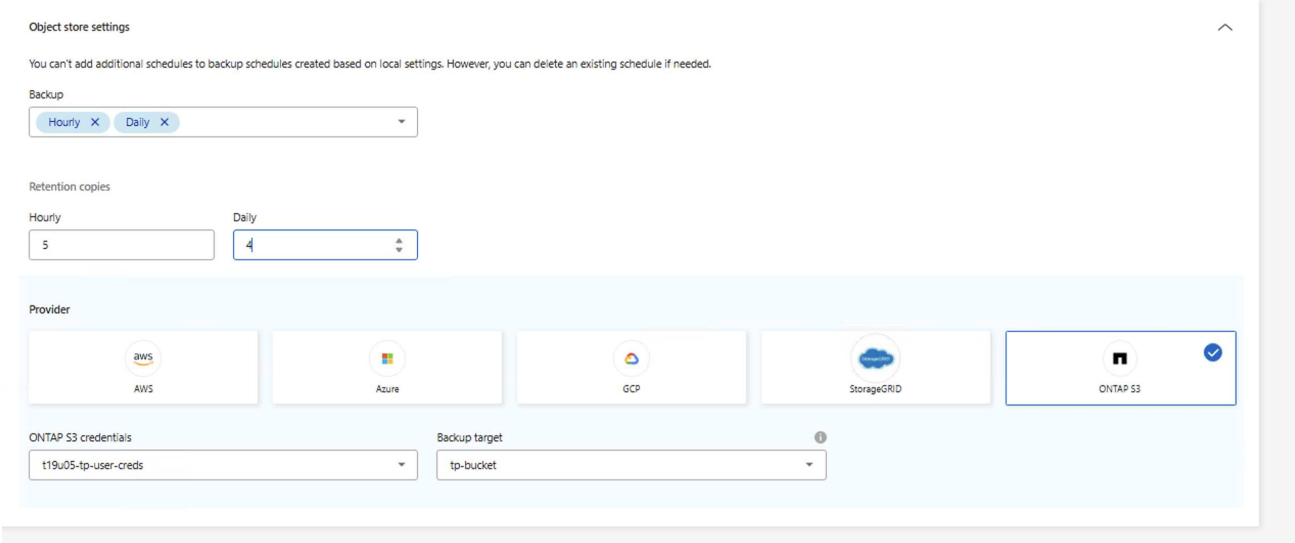


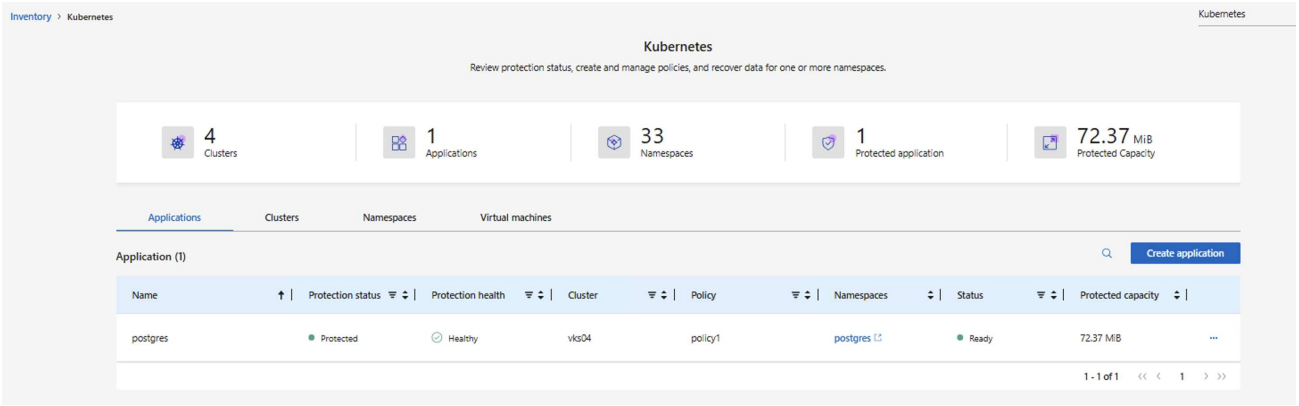
A continuación, debes configurar los parámetros de protección de la aplicación. Puedes crear una nueva política de protección o usar una ya existente. En este ejemplo, se crea una nueva política de protección para la aplicación y se asocia a ella. Despliega **Política** y haz clic en **Crear nueva política**.



Introduce un nombre para la política y elige la arquitectura de backup. En este ejemplo, se elige de disco a almacén de objetos. Define la programación de instantáneas locales, que controla con qué frecuencia se toman instantáneas en el almacenamiento primario. Luego configura por separado los ajustes de backup en el almacén de objetos: elige el destino de backup (en este ejemplo, ONTAP S3), establece la programación de transferencia que controla cuándo se copian los datos de las instantáneas al almacén de objetos y especifica el número de copias de retención. La programación de transferencia es independiente de la programación de instantáneas, así que los recursos se copian al almacén de objetos según la programación de transferencia, no inmediatamente cuando se toma cada instantánea. También puedes modificar el número máximo de reintentos y el intervalo de reintento si lo necesitas. Haz clic en **Create** para continuar. La aplicación se detectará y se le asociará la política de protección.





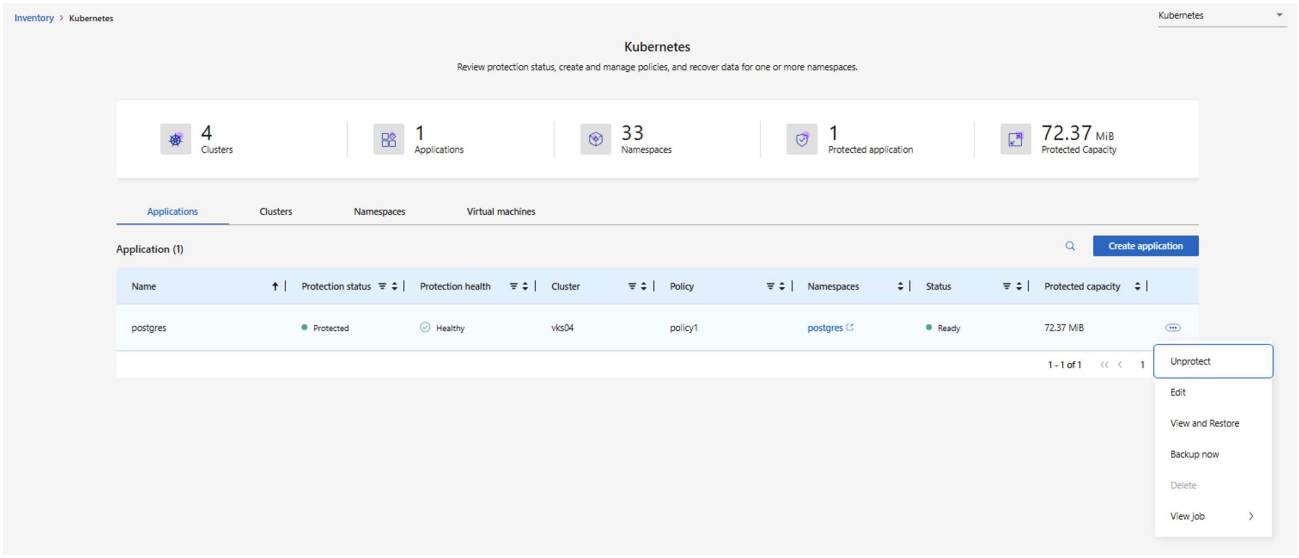


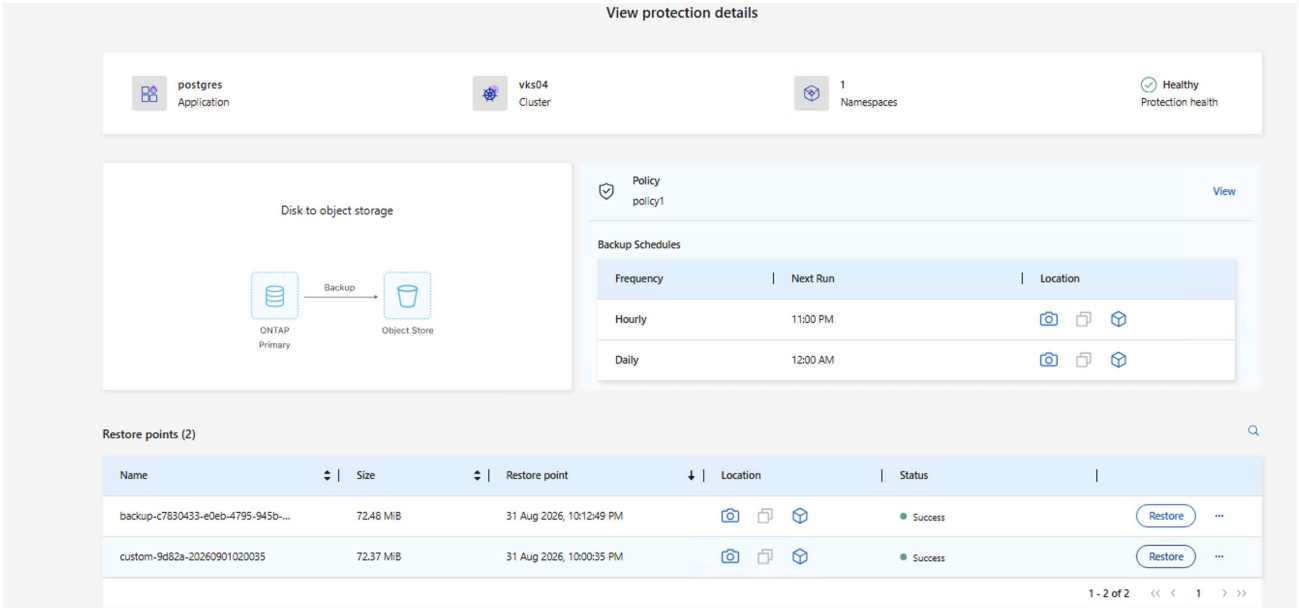
6. **Restaurar la aplicación desde una copia de seguridad**

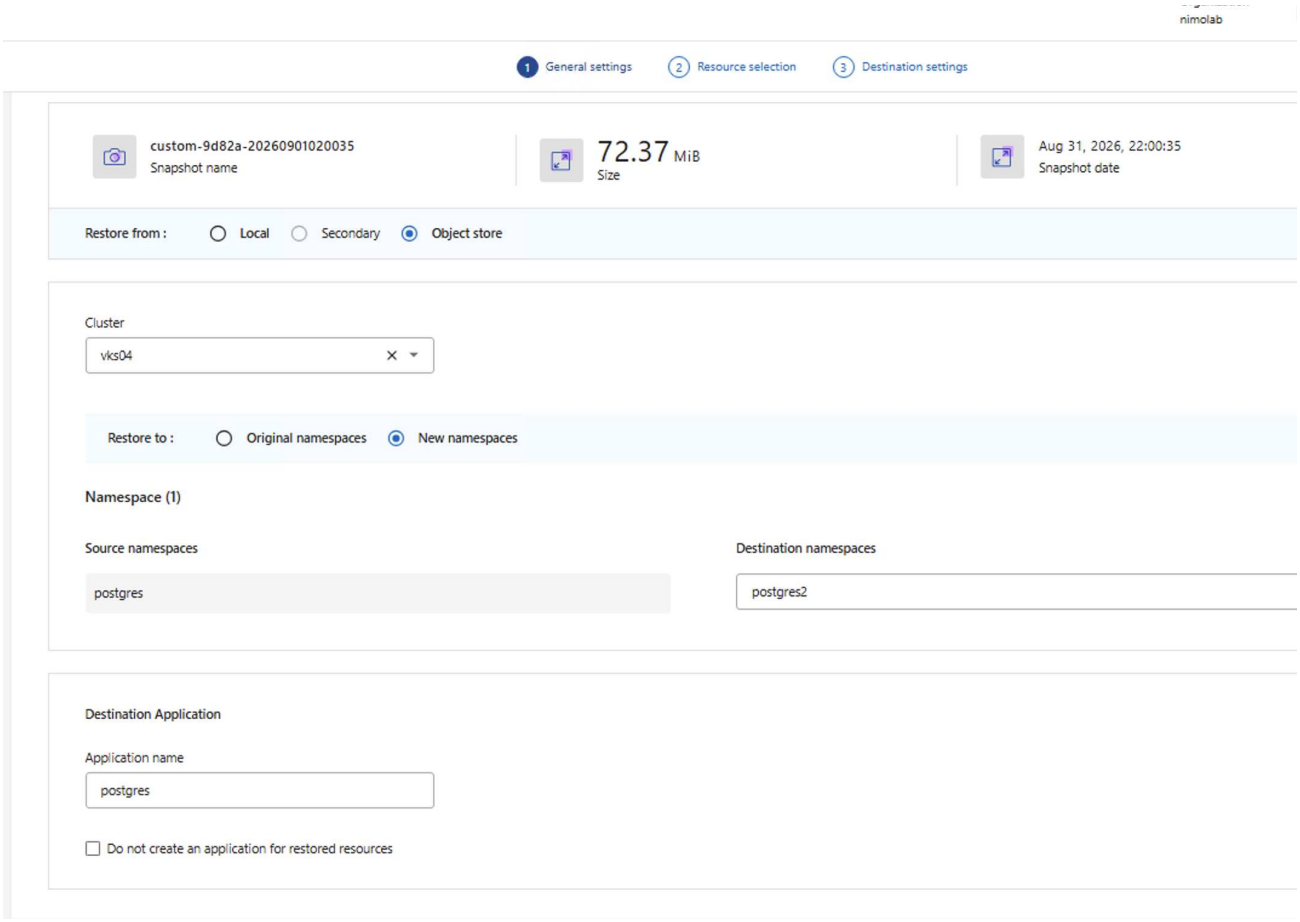
> Necesitas el rol de superadministrador de NetApp Backup and Recovery o el de administrador de restauración de NetApp Backup and Recovery para restaurar una aplicación de Kubernetes.

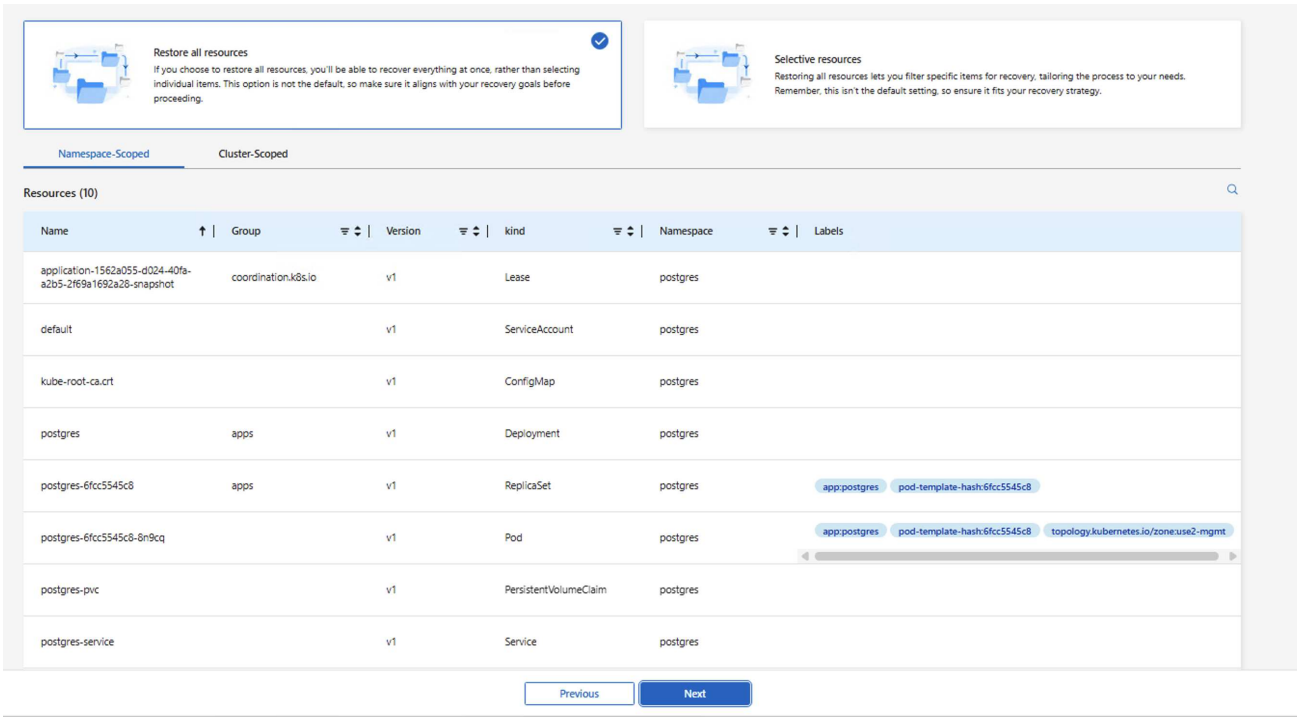
▼ **Mostrar ejemplo**

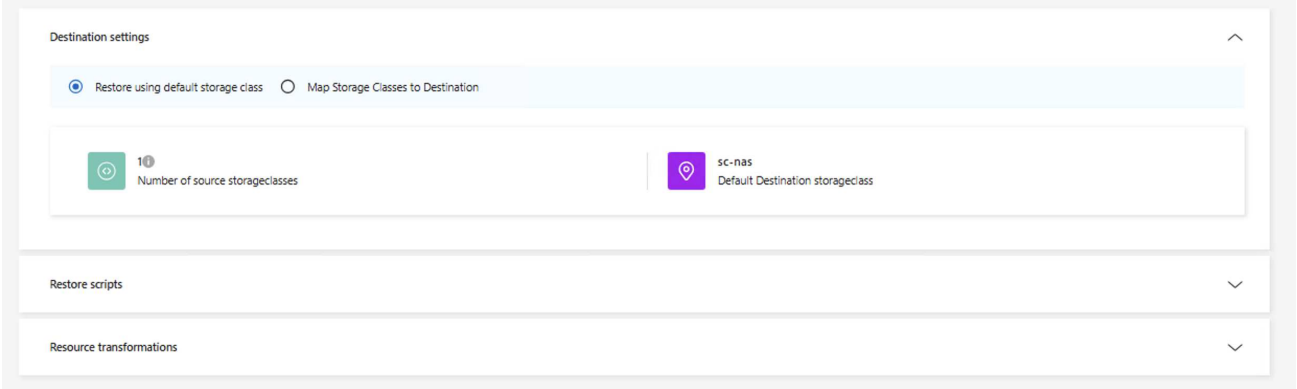
- Puedes elegir cualquier punto de restauración disponible —una instantánea local, una copia de seguridad en un almacén de objetos o una copia de seguridad de un destino secundario— para restaurar la aplicación.
- Puedes restaurar la aplicación a su espacio de nombres original o a otro espacio de nombres.
- Puedes seleccionar qué recursos quieres incluir en la restauración.
- Puedes elegir la clase de almacenamiento que se usará para los recursos de la aplicación restaurados.


Inventory › Kubernetes
Kubernetes
Kubernetes
Review protection status, create and manage policies, and recover data for one or more namespaces.
4 Clusters
1 Applications
33 Namespaces
1 Protected application
72.37 MiB Protected Capacity
Applications
Clusters
Namespaces
Virtual machines
Application (1)
Create application
Name
Protection status
Protection health
Cluster
Policy
Namespaces
Status
Protected capacity
postgres
Protected
Healthy
vks04
policy1
postgres
Ready
72.37 MiB
1 - 1 of 1
Unprotect
Edit
View and Restore
Backup now
Delete
View job

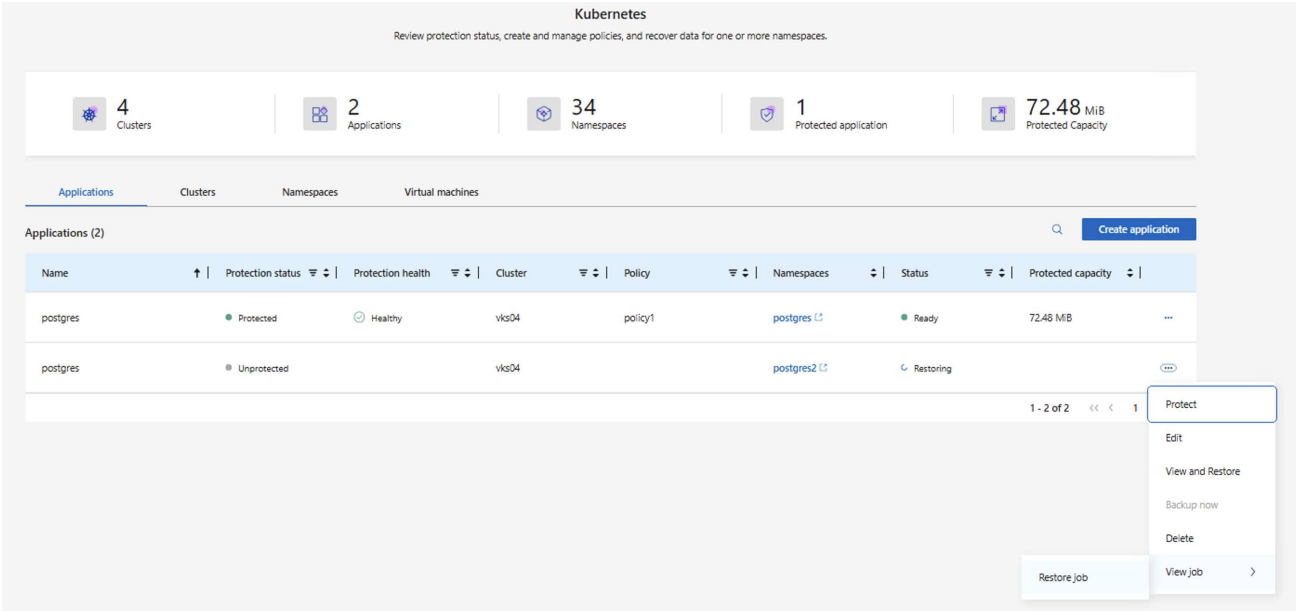

View protection details
postgres
Application
vks04
Cluster
1
Namespaces
Healthy
Protection health
Disk to object storage
Backup
ONTAP Primary
Object Store
Policy
policy1
View
Backup Schedules
Frequency
Next Run
Location
Hourly
11:00 PM
Daily
12:00 AM
Restore points (2)
Name
Size
Restore point
Location
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

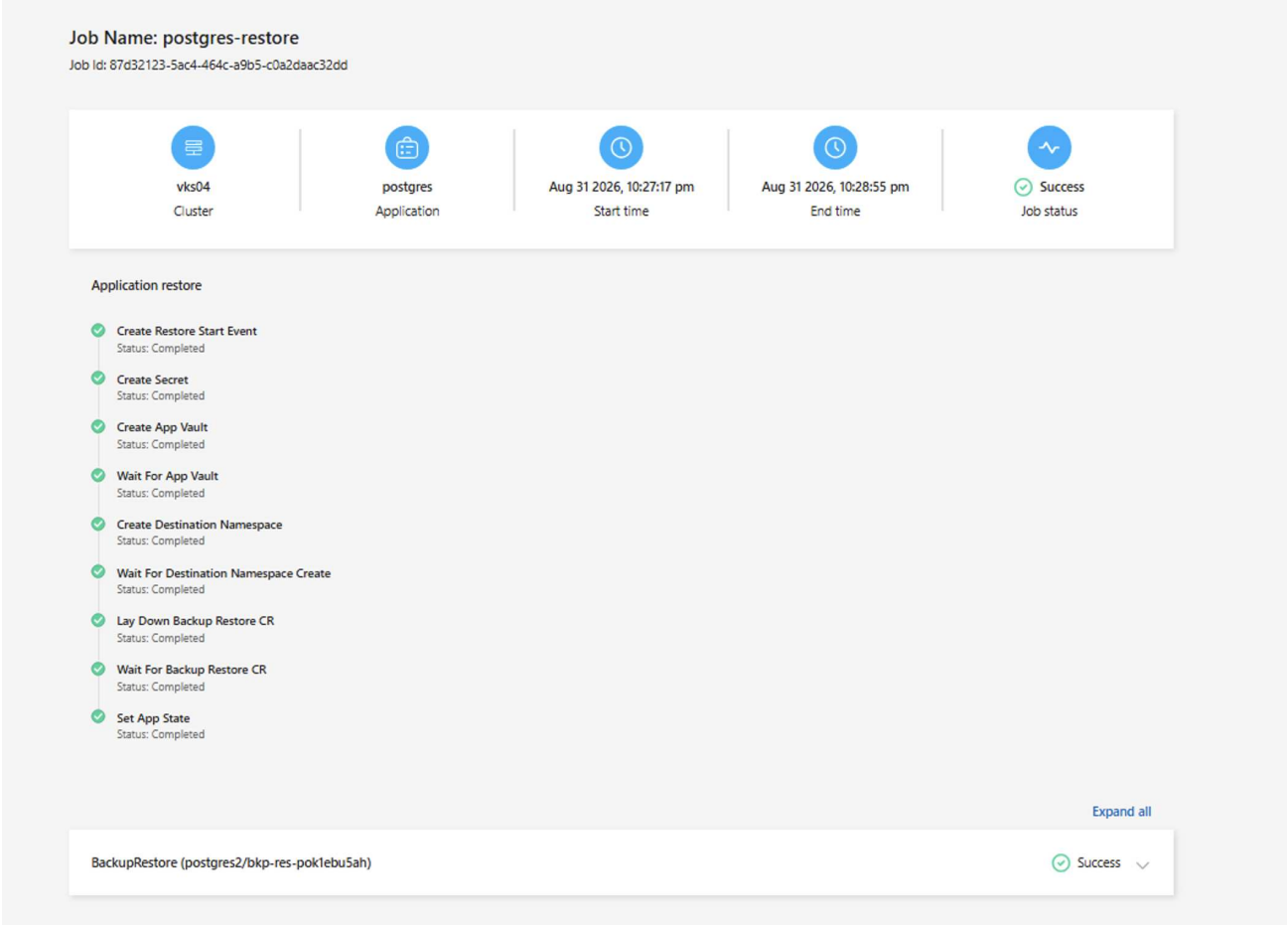

nimolab
1 General settings
2 Resource selection
3 Destination settings
custom-9d82a-20260901020035
Snapshot name
72.37 MiB
Size
Aug 31, 2026, 22:00:35
Snapshot date
Restore from :
Local
Secondary
Object store
Cluster
vks04
Restore to :
Original namespaces
New namespaces
Namespace (1)
Source namespaces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

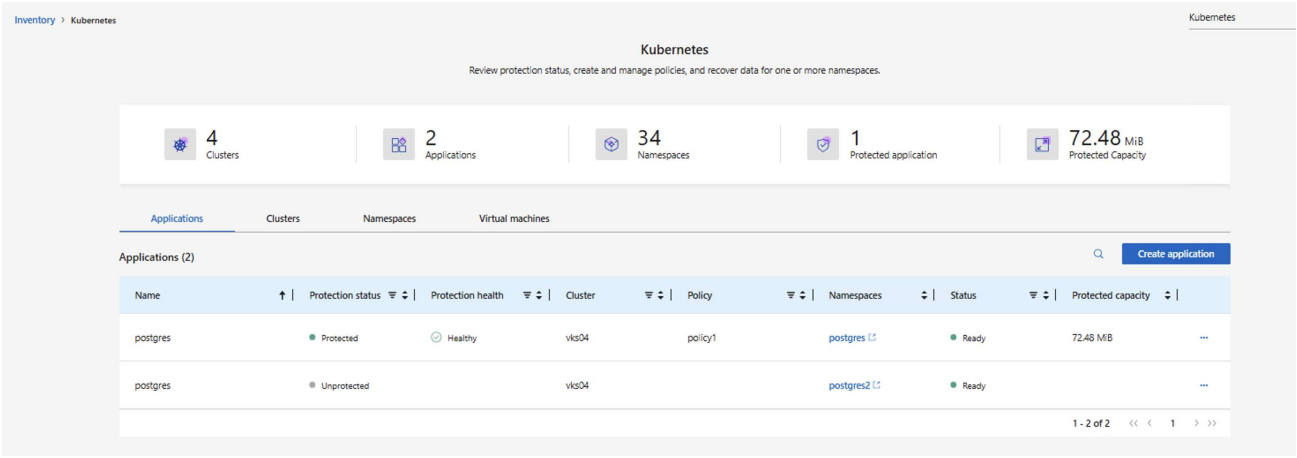

Restore all resources
If you choose to restore all resources, you'll be able to recover everything at once, rather than selecting individual items. This option is not the default, so make sure it aligns with your recovery goals before proceeding.
Selective resources
Restoring all resources lets you filter specific items for recovery, tailoring the process to your needs. Remember, this isn't the default setting, so ensure it fits your recovery strategy.
Namespace-Scoped
Cluster-Scoped
Resources (10)
Name
Group
Version
kind
Namespace
Labels
application-1562a055-d024-40fa-a2b5-2f69a1692a28-snapshot
coordination.k8s.io
v1
Lease
postgres
default
v1
ServiceAccount
postgres
kube-root-ca.crt
v1
ConfigMap
postgres
postgres
apps
v1
Deployment
postgres
postgres-6fcc5545c8
apps
v1
ReplicaSet
postgres
app:postgres
pod-template-hash:6fcc5545c8
postgres-6fcc5545c8-8n9cq
v1
Pod
postgres
app:postgres
pod-template-hash:6fcc5545c8
topology.kubernetes.io/zone:use2-mgmt
postgres-pvc
v1
PersistentVolumeClaim
postgres
postgres-service
v1
Service
postgres
Previous
Next



Puedes ver el progreso de la tarea de restauración en la sección Monitor de NetApp Console. Después de que se complete la tarea de restauración, puedes ver los recursos restaurados en el clúster de Kubernetes de vSphere.







Realiza copias de seguridad y restaura aplicaciones en contenedores en un clúster de Kubernetes Service de vSphere usando Trident Protect

Realiza copias de seguridad y restaura aplicaciones en contenedores en un clúster de vSphere Kubernetes Service (VKS) usando instantáneas y copias de seguridad de Trident Protect. Este procedimiento incluye crear un AppVault usando almacenamiento de objetos S3 de ONTAP, configurar Trident Protect para capturar datos de la aplicación, incluidos los objetos de recursos de Kubernetes y los volúmenes persistentes, y restaurar los datos cuando sea necesario.

Las aplicaciones en contenedores que se ejecutan en un clúster de VKS se gestionan como cargas de trabajo

de Kubernetes en los espacios de nombres de los nodos de trabajo. Es importante proteger tanto los metadatos de la aplicación como los volúmenes persistentes para que, si se pierden o se dañan, puedas recuperarlos.

Los volúmenes persistentes de las aplicaciones VKS pueden respaldarse mediante el almacenamiento ONTAP integrado con el clúster VKS usando "Trident CSI". Este procedimiento usa "Trident Protect" para crear capturas de pantalla de aplicaciones, cuyos metadatos se almacenan en el almacén de objetos ONTAP mientras que los datos de la captura de pantalla de volumen permanecen en el backend de almacenamiento, y copias de seguridad, cuyos datos se copian al almacén de objetos ONTAP. Puedes restaurar desde una captura de pantalla o una copia de seguridad cuando lo necesites.

Trident Protect permite realizar instantáneas, copias de seguridad, restauraciones y recuperación ante desastres de aplicaciones en un clúster de Kubernetes. Los datos que se pueden proteger con Trident Protect incluyen los objetos de recursos de Kubernetes asociados a las aplicaciones y sus volúmenes persistentes.

A continuación se indican las versiones de los distintos componentes utilizados en los ejemplos de esta sección

- VMware Cloud Foundation (VCF) 9.1 con el servicio de Kubernetes de vSphere
- "Trident 26.06"
- "Trident Protect 26.06"
- "ONTAP 9.16"

Instala Trident Protect en el clúster de Kubernetes de vSphere

▼ **Instala Trident Protect**

Utiliza Helm para instalar Trident Protect en el clúster del servicio de Kubernetes de vSphere. Los ejemplos de esta sección utilizan `tridentctl-protect`, la CLI de Trident Protect. Instálalo siguiendo las instrucciones en "Documentación de la interfaz de línea de comandos (CLI) de Trident Protect". Otros métodos para instalar Trident Protect están documentados en "Documentación de Trident Protect".

En primer lugar, crea y etiqueta el `trident-protect` espacio de nombres. La etiqueta `enforce=privileged` es obligatoria porque Trident Protect ejecuta cargas de trabajo con privilegios. Sin ella, el controlador de admisión de seguridad de pods de Kubernetes impedirá que se inicien los pods de Trident Protect.

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

A continuación, añade el repositorio de Helm e instala Trident Protect en el espacio de nombres.

```
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
```

Si los pods de Trident Protect no se inician con un error de admisión `PodSecurity`, es posible que las etiquetas de seguridad de los pods no se hayan aplicado al `trident-protect` espacio de nombres antes de la instalación. Usa `--overwrite` para volver a aplicarlas y luego verifica que los pods se inicien.

```
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$
```

Crear App Vault para almacenamiento de objetos

▼ **Crear AppVault**

Antes de crear instantáneas y copias de seguridad de una aplicación, debes configurar el almacenamiento de objetos en Trident Protect. Esto se hace creando un CR de AppVault. Solo los administradores pueden crear y configurar un CR de AppVault.

Los objetos AppVault son la representación de un recurso personalizado de Kubernetes de un bucket de almacenamiento. Un AppVault CR contiene las configuraciones necesarias para que un bucket se pueda usar en operaciones de protección, como copias de seguridad, instantáneas, operaciones de restauración y replicación SnapMirror.

Los siguientes pasos permiten crear un CR de AppVault configurado para ONTAP S3: Crea un servidor de almacén de objetos S3 en el SVM del clúster ONTAP. . Crea un bucket en el servidor de almacén de objetos. . Crea un usuario S3 en el SVM. Guarda la clave de acceso y la clave secreta en un lugar seguro.

+ NOTA: Trident Protect requiere que el usuario de S3 tenga al menos `PutObject`, `GetObject`, `ListBucket` y `DeleteObject` permisos. Sin estos permisos, las operaciones de copia de seguridad y restauración de AppVault fallan con errores de acceso denegado. Para más detalles, consulta "Documentación de Trident Protect AppVault". . En el clúster VKS, crea un secreto para guardar las credenciales de ONTAP S3. . Crea un objeto AppVault para ONTAP S3.

Configura Trident Protect AppVault para ONTAP S3

El manifiesto que se muestra a continuación utiliza HTTPS con la validación de certificados activada, lo cual es obligatorio para producción. Si tu endpoint ONTAP S3 utiliza un certificado firmado por una CA pública en la que el clúster ya confía, no necesitas ninguna configuración adicional de certificados. Si utiliza un certificado autofirmado o de una CA interna, proporciona el certificado de CA raíz personalizado usando el campo `rootCA` como se muestra en el manifiesto. Consulta la variante solo para laboratorio al final de esta sección

si necesitas una configuración HTTP para pruebas aisladas que no sean de producción.

```
# 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
```

Variante solo para laboratorio (HTTP, sin validación de certificados)

Utiliza únicamente la siguiente `s3` configuración en un entorno de laboratorio aislado en el que no haya datos confidenciales. No utilices esta configuración en producción.

```
    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$
```

Crear una aplicación de Trident Protect

▼ **Crear una aplicación de Trident Protect**

Este ejemplo trata sobre la protección de datos de la aplicación de ejemplo «postgres», que está instalada en el espacio de nombres «postgres». Para obtener más información sobre la instalación, consulta "Implementa cargas de trabajo de VKS usando el almacenamiento NetApp".

> Los PVC de PostgreSQL de ejemplo solicitan el modo de acceso RWO. El modo de acceso debe especificarse explícitamente en el manifiesto del PVC; Trident no selecciona automáticamente un modo de acceso en función del protocolo de almacenamiento. Se admite RWX para los PVC respaldados por NAS, y los PVC respaldados por SAN pueden admitir RWX con una configuración adicional. Para obtener más información, consulta "Documentación de Trident Protect".

En el ejemplo, el espacio de nombres postgres tiene una aplicación y todos los recursos de ese espacio de nombres se incluyen al crear la Trident Protect Application.

```
# 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
```

Protege la aplicación creando una copia de seguridad

▼ **Crear copias de seguridad**

Crear una copia de seguridad bajo demanda

Crea una copia de seguridad de la aplicación postgres-app creada anteriormente que incluya todos los recursos en el espacio de nombres postgres. Indica el nombre de appvault donde se almacenarán las copias de seguridad.

```
# 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
```

Realizar copias de seguridad de forma programada

Crea una programación para las copias de seguridad especificando la granularidad y el número de copias de seguridad que quieres conservar.

```
# 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
```

Restaurar desde una copia de seguridad

▼ **Restaurar a partir de copias de seguridad**

Restaurar la aplicación en el mismo espacio de nombres

En este ejemplo, la copia de seguridad postgres-backup-on-demand contiene la copia de seguridad para postgres-app.

Antes de eliminar la aplicación, comprueba que la copia de seguridad desde la que pretendes restaurar se encuentre en el estado `Completed`. No puedes usar una copia de seguridad en curso o fallida para restaurar la aplicación.

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

Después de confirmar que el estado de la copia de seguridad es `Completed`, elimina la aplicación postgres y asegúrate de que los PVC y los objetos de pod se eliminen del espacio de nombres "postgres".

```
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.
```

Ahora, crea un objeto de restaurar sin movimiento.

```
# 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
```

Comprueba que se hayan restaurado la puesta en marcha de aplicaciones de postgres, el servicio, el pod y los PVC.

```
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
```

Restaurar la aplicación en un espacio de nombres diferente

En primer lugar, crea un nuevo espacio de nombres en el que quieras restaurar la app; en este ejemplo, postgres2. Ya está disponible una copia de seguridad horaria creada por la programación para postgres-app. Utiliza esta copia de seguridad para restaurar la aplicación en el nuevo espacio de nombres 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
```

Para obtener la ruta de la copia de seguridad, utiliza el comando `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
```

Comprueba que los objetos de la aplicación postgres y los PVC se hayan creado en el nuevo espacio de nombres postgres2.

```
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$
```

Protege la aplicación mediante instantáneas

▼ Crear instantáneas

Crear una instantánea bajo demanda Crea una instantánea para la app y especifica el AppVault donde se almacenarán los metadatos de la instantánea. Los datos de la instantánea del volumen no se copian en el almacenamiento de objetos; permanecen en el backend de almacenamiento.

```
# 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$
```

Crear una programación para las instantáneas Crea una programación para las instantáneas. Especifica la granularidad y el número de instantáneas que se van a conservar.

```
# 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
```

Restaurar desde una instantánea

▼ **Restaurar desde una instantánea**

Restaurar la aplicación desde una instantánea en el mismo espacio de nombres

Antes de eliminar la aplicación, comprueba que la instantánea desde la que pretendes restaurar se encuentre en el estado `Completed`. No se puede utilizar una instantánea en curso o fallida para restaurar la aplicación.

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

Después de confirmar que el estado de la instantánea es `Completed`, elimina los objetos de la aplicación y los PVC de postgres del espacio de nombres de postgres.

```
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$
```

Crea un objeto de restaurar sin movimiento a partir de la instantánea.

```
# 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
```

Comprueba que los objetos y los PVC de la aplicación se creen en el espacio de nombres postgres.

```
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$
```

Restaurar la aplicación desde una instantánea en un espacio de nombres diferente

Elimina la aplicación en el espacio de nombres postgres2 que restauraste previamente desde la copia de seguridad.

```
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$
```

Crea el objeto de restaurar sin movimiento de la instantánea a partir de la instantánea y proporciona la asignación del espacio de nombres.

```
# 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
```

Comprueba que los objetos y los PVC de la aplicación se restauraron en el espacio de nombres postgres2.

```
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$
```