vSphere Kubernetes Service com o storage NetApp

NetApp container solutions

NetApp
September 01, 2026

Índice

vSphere Kubernetes Service com o storage NetApp

Saiba mais sobre o VMware vSphere Kubernetes Service com o NetApp ONTAP

O VMware vSphere Kubernetes Service (VKS) é uma oferta de Kubernetes gerenciado da VMware que permite que você implante, gerencie e dimensione aplicativos em contêineres usando o Kubernetes. Combinado com o storage NetApp ONTAP, o VKS oferece persistência, desempenho e gerenciamento de dados de nível empresarial para cargas de trabalho nativas da nuvem.

O VKS está integrado ao VMware Cloud Foundation (VCF) e está em conformidade com o CNCF Kubernetes upstream, garantindo compatibilidade com o ecossistema Kubernetes e as ferramentas em geral.

Principais características do vSphere Kubernetes Service

1. **Clusters Kubernetes gerenciados:** vSphere Kubernetes Service simplifica a implantação e o gerenciamento de clusters Kubernetes. Ele automatiza tarefas como criação, atualização, escalonamento e monitoramento de clusters.
2. **Integração com a infraestrutura VMware:** O VKS integra-se perfeitamente com o VMware vSphere, NSX-T e outras soluções VMware, permitindo que as organizações aproveitem sua infraestrutura VMware existente para cargas de trabalho do Kubernetes.
3. **Suporte a várias nuvens e nuvem híbrida:** vSphere Kubernetes Service oferece suporte à implantação em nuvens privadas, nuvens públicas (por exemplo, AWS, Azure, Google Cloud) e ambientes de nuvem híbrida, proporcionando às organizações flexibilidade no gerenciamento de seus aplicativos.
4. **Segurança integrada:** O VKS inclui recursos de segurança como controle de acesso baseado em funções (RBAC), aplicação de políticas e integração com o VMware NSX para segurança de rede.
5. **Suporte ao ecossistema Kubernetes:** O VKS oferece suporte à API completa do Kubernetes, garantindo portabilidade entre ambientes e compatibilidade com o ecossistema Kubernetes, incluindo ferramentas como Istio para service mesh, Prometheus para monitoramento e outras ferramentas certificadas pela CNCF. As organizações podem aplicar habilidades e integrações padrão do Kubernetes diretamente aos clusters VKS.
6. **Ferramentas amigáveis para desenvolvedores:** O VKS oferece suporte a fluxos de trabalho padrão de desenvolvedores Kubernetes, incluindo pipelines de CI/CD, GitOps práticas e ferramentas como kubectl e Helm, permitindo que as equipes de desenvolvimento criem e implantem aplicativos em contêineres sem precisar aprender interfaces proprietárias.
7. **Escalabilidade e alta disponibilidade:** O VKS oferece recursos como escalonamento automático e tolerância de falhas para garantir que os aplicativos permaneçam altamente disponíveis e com bom desempenho.
8. **Observabilidade e monitoramento:** O VKS integra-se com ferramentas de monitoramento padrão compatíveis com Kubernetes para fornecer informações em tempo real sobre o desempenho do cluster e do aplicativo.

Casos de uso para o vSphere Kubernetes Service

1. **Modernização de aplicações:** As organizações podem modernizar aplicações legadas, conteinerizando-as e implantando-as em clusters Kubernetes gerenciados pelo VKS.
2. **DevOps Enablement:** O VKS oferece suporte a fluxos de trabalho DevOps, fornecendo automação, integração de CI/CD e ferramentas amigáveis para desenvolvedores.
3. **Implantações em nuvem híbrida:** As empresas podem usar o VKS para implantar clusters Kubernetes em ambientes locais e na nuvem, permitindo operações consistentes.
4. **Arquitetura de microsserviços:** O VKS oferece suporte ao desenvolvimento de aplicativos baseados em microsserviços, permitindo que as equipes criem e dimensionem sistemas distribuídos.

Introdução ao storage e ao serviço Kubernetes do vSphere

Instale um cluster do vSphere Kubernetes Service

Instale um cluster do vSphere Kubernetes Service (VKS) em seu ambiente VCF 9.1 e configure os nós de trabalho com os utilitários de storage apropriados para suas cargas de trabalho.

Sobre esta tarefa

Os utilitários NFS são instalados automaticamente nos nós de trabalho do cluster que você cria. Se você quiser criar nós de trabalho com utilitários iSCSI ou NVMe instalados, é necessário criar uma imagem OVA personalizada e usá-la para os nós de trabalho. A imagem personalizada pode ser criada usando o Docker, e a ferramenta de linha de comando `vcf` pode ser usada para criar um arquivo OVA a partir dessa imagem. Esse arquivo OVA pode ser carregado na Content Library no vSphere. Ao criar o cluster VKS, essa imagem da Content Library pode ser selecionada para os pools de nós.

Passos

1. Crie o cluster VKS:

   Você pode criar um cluster VKS usando a imagem de nó de trabalho padrão ou uma imagem personalizada que você cria. A imagem de nó de trabalho padrão inclui utilitários NFS pré-instalados, enquanto a imagem personalizada pode incluir utilitários iSCSI e NVMe. As seguintes opções estão disponíveis:

   - **Somente NAS (padrão):** Crie um cluster VKS usando a imagem de nó de trabalho padrão, que inclui utilitários NFS pré-instalados.
   - **SAN habilitado (imagem personalizada):** Crie uma imagem Docker personalizada que inclua utilitários iSCSI e NVMe, gere um arquivo OVA usando a ferramenta de linha de comando `vcf`, carregue o OVA na vSphere Content Library e, em seguida, crie o cluster VKS selecionando essa imagem personalizada para os pools de nós.

   ▼ Mostrar instruções para criar um cluster do vSphere Kubernetes Service somente NAS com a imagem padrão do nó de trabalho

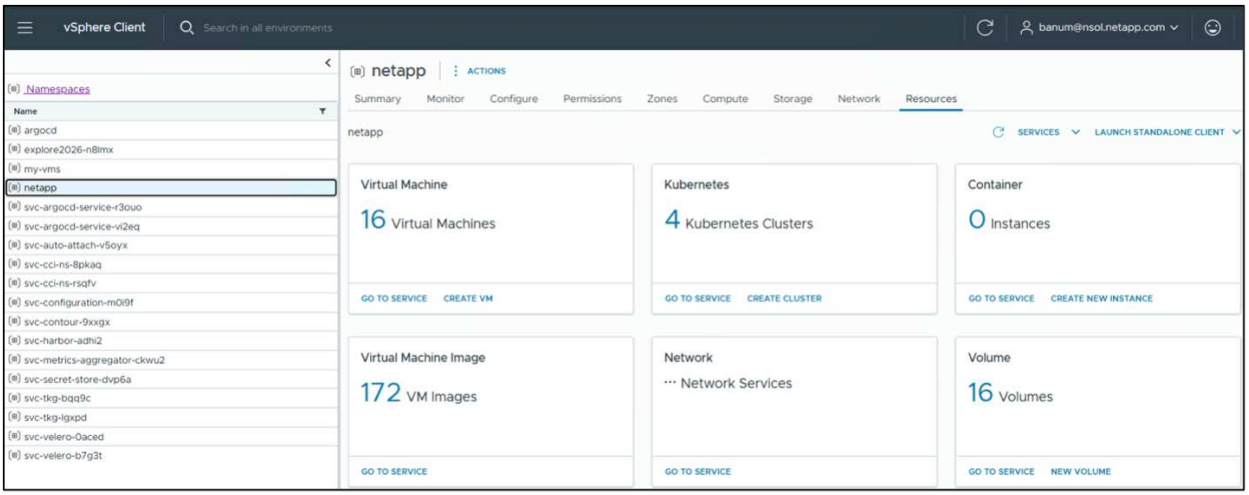
   Crie o cluster VKS usando a imagem de nó de trabalho padrão. Siga as instruções para especificar o nome do cluster, a configuração do pool de nós e outras configurações.

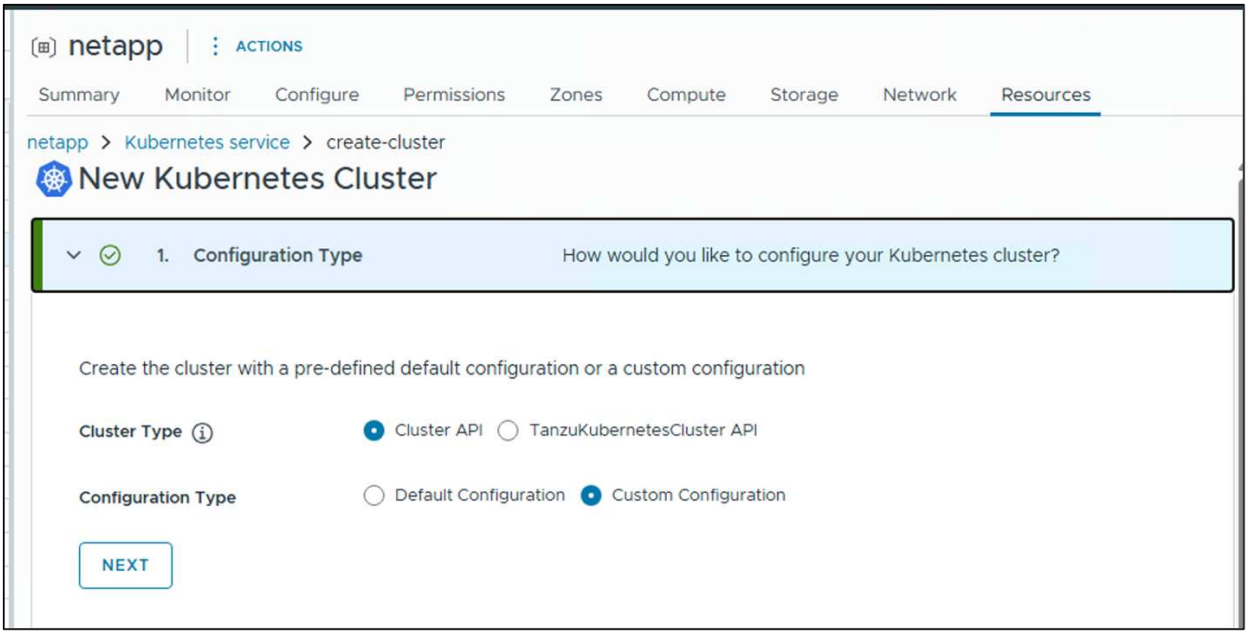
   No vSphere Client, acesse o namespace onde você deseja criar o cluster Kubernetes do vSphere. Na guia **Recursos** desse namespace, clique em **Criar cluster** na seção Kubernetes ou clique em **Ir para o serviço** e, em seguida, clique em **Criar cluster**.

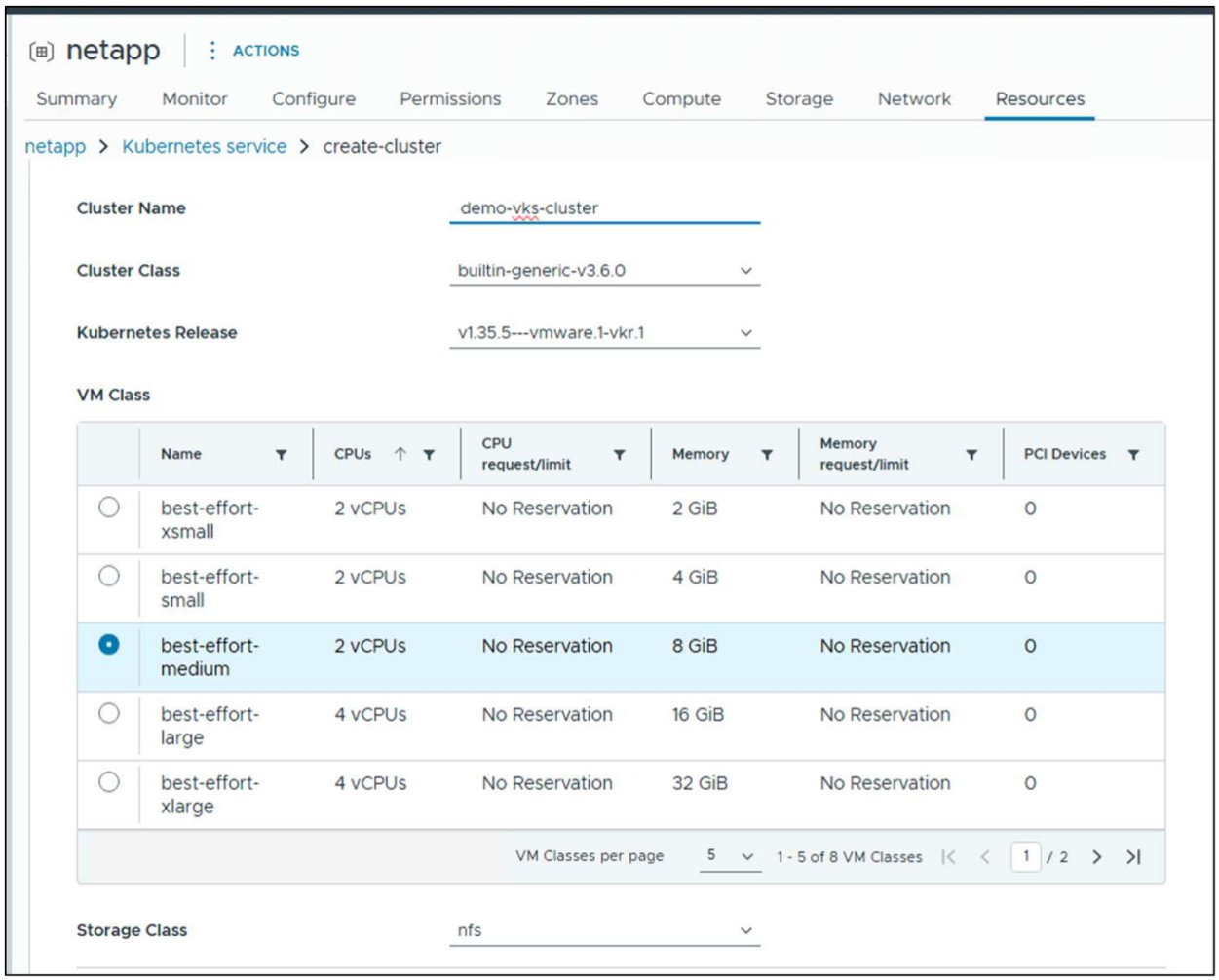
Preencha o formulário com os detalhes do cluster:

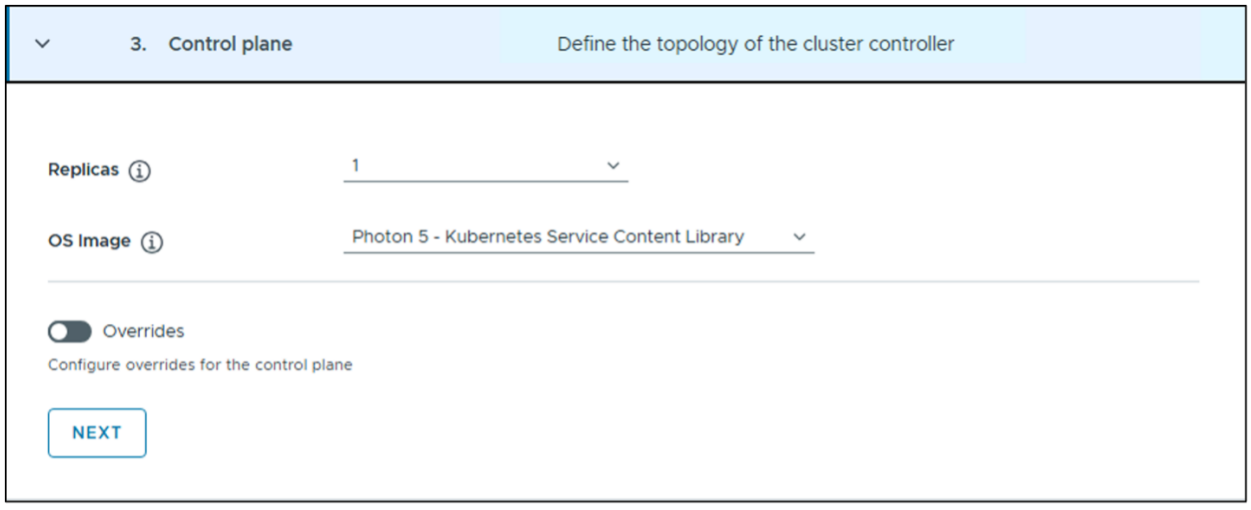
- Na seção 1, **Tipo de Configuração**, selecione **Custom**.
- Na seção 2, **Configurações Gerais**, forneça um nome para o cluster e deixe todas as outras configurações com seus valores padrão.
- Aceite todas as opções padrão para a seção 3.
- Na seção 4, **Pools de Nós**, clique nas reticências (…) ao lado do pool de nós e clique em **Editar**. Aumente o número de réplicas para 3 e selecione a versão mais recente do Ubuntu para a imagem do sistema operacional. Clique em **Avançar**, depois em **Revisar e Confirmar**.

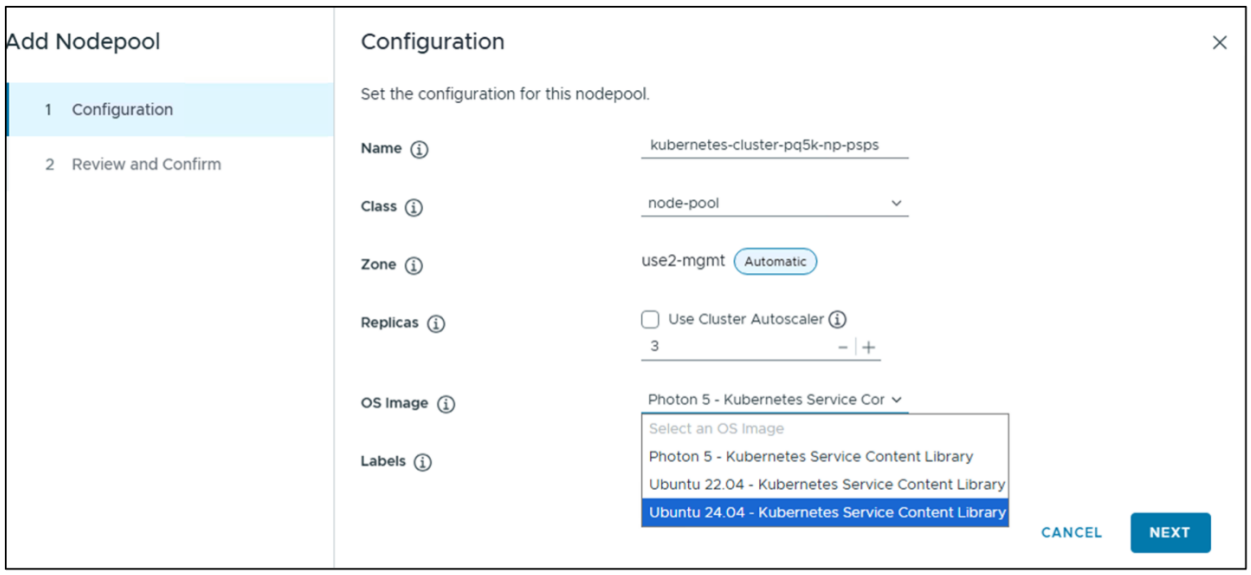
Após revisar os detalhes do cluster, clique em **Concluir**. O vSphere Kubernetes cluster será criado em alguns minutos.

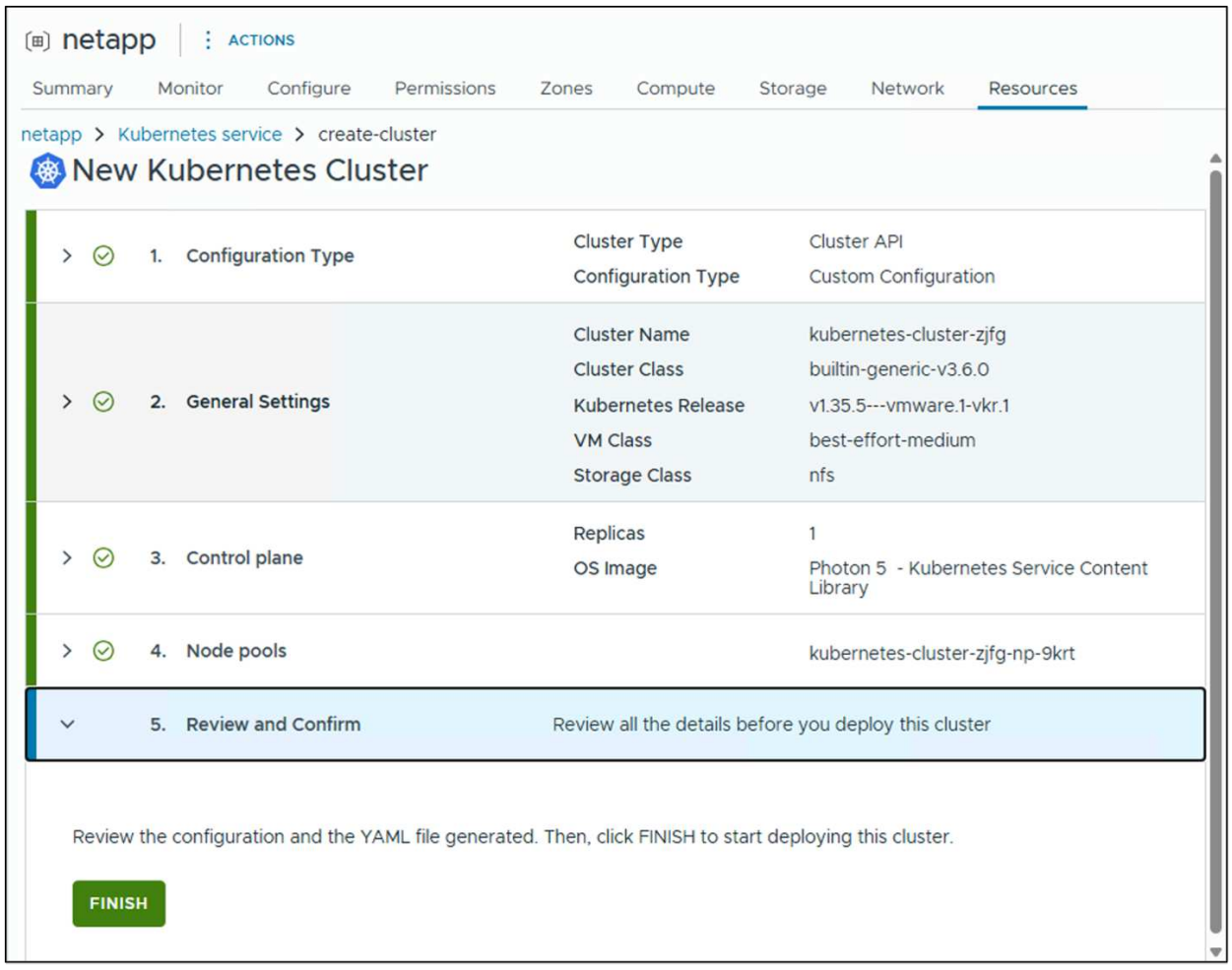


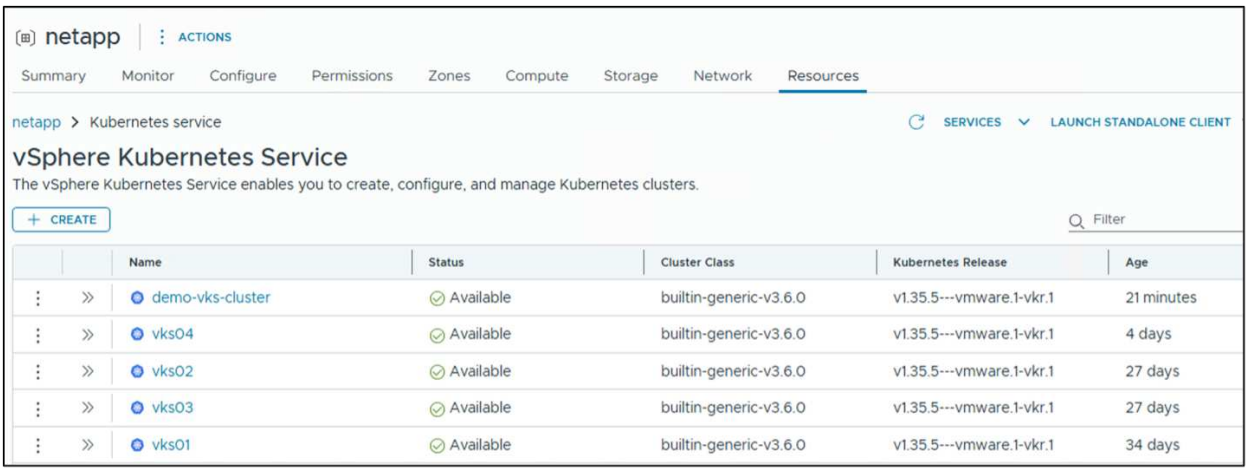



netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp > Kubernetes service > create-cluster
Cluster Name
demo-vks-cluster
Cluster Class
builtin-generic-v3.6.0
Kubernetes Release
v1.35.5---vmware.1-vkr.1
VM Class
Name
CPUs
CPU request/limit
Memory
Memory request/limit
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 instruções para criar uma imagem OVA personalizada para nós de trabalho habilitados para SAN

Crie uma imagem Docker com os utilitários necessários instalados. O exemplo a seguir demonstra a criação de uma imagem Docker baseada no Ubuntu 24.04 com os utilitários iSCSI e Multipathing instalados.

```
#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"]
```

Crie a imagem Docker usando:

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

Se o registro Docker não estiver disponível, execute este comando para iniciar um registro local e enviar a imagem para ele:

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

Marque a imagem e envie-a.

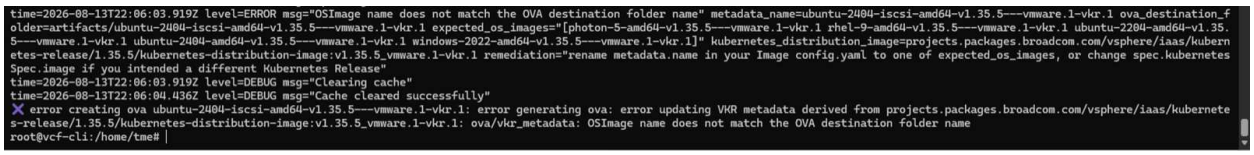
```
docker tag ubuntu-2404:san localhost:5000/ubuntu-2404:san
docker push localhost:5000/ubuntu-2404:san
```

Crie o arquivo de configuração da imagem (salve como `ubuntu2404vkr135-san.yml`) e faça referência à imagem:

```
#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
```

ⓘ O nome dos metadados deve estar na lista pré-aprovada. Caso contrário, você receberá um erro como o mostrado abaixo ao tentar criar o arquivo OVA. A lista pré-aprovada pode ser encontrada na ajuda da CLI do VCF para o `kr bake` comando.



Crie o arquivo OVA usando a CLI do VCF:

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

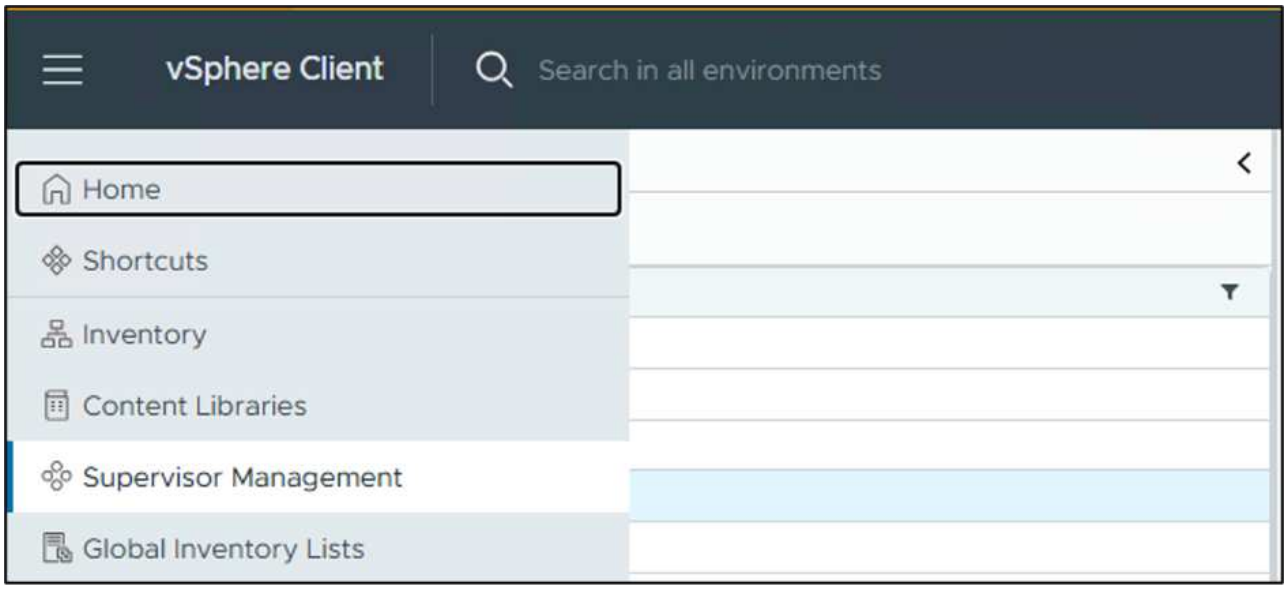
Transfira o arquivo OVA usando SCP para um computador a partir do qual você possa carregá-lo para a vSphere Content Library.

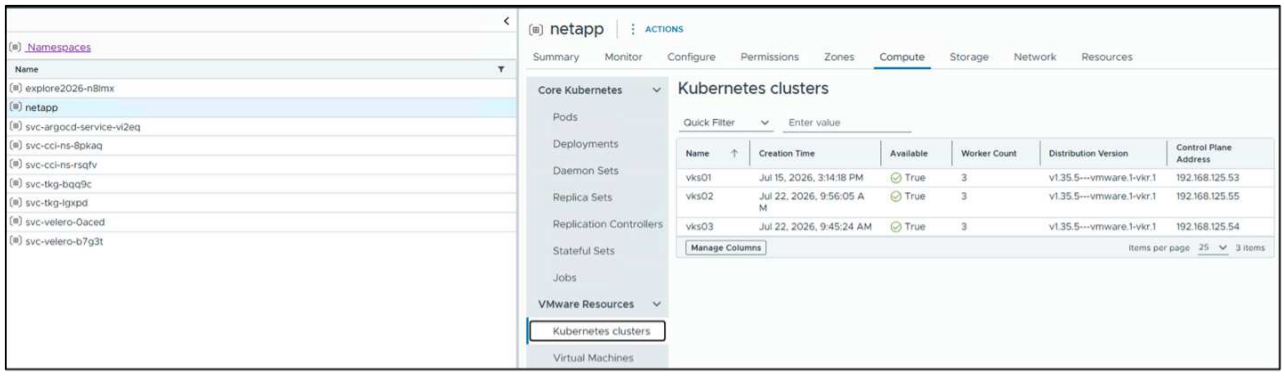
2. Após a instalação do cluster, localize-o no vCenter e faça o download do arquivo kubeconfig.

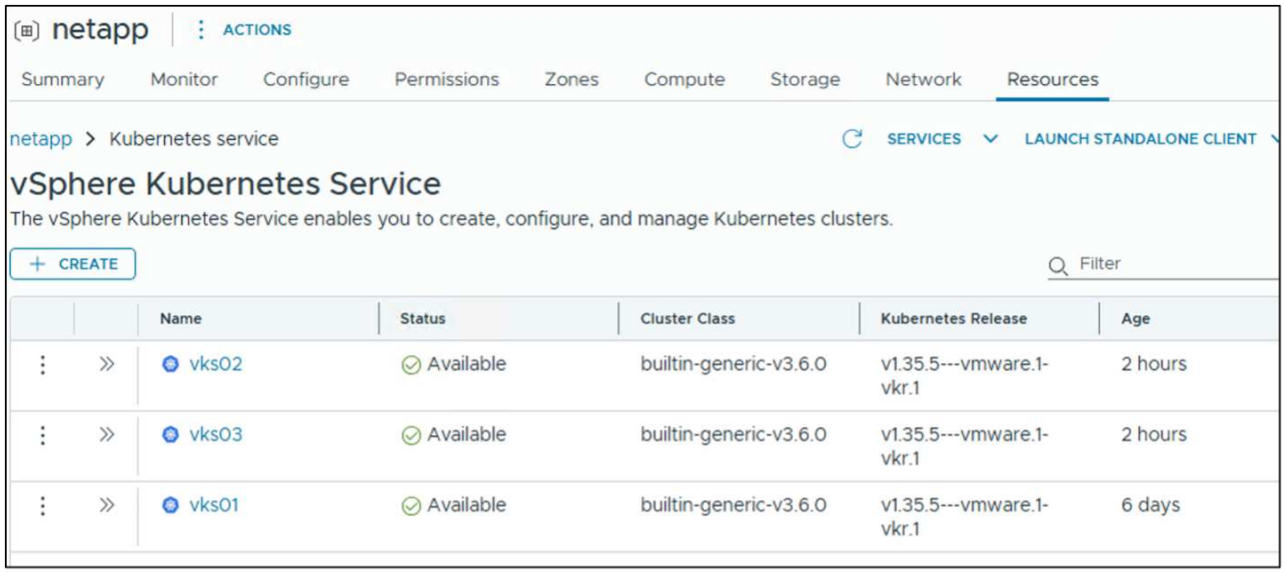
    a. Clique em **Gerenciamento de supervisores** no menu principal e selecione o namespace onde o seu cluster foi criado.

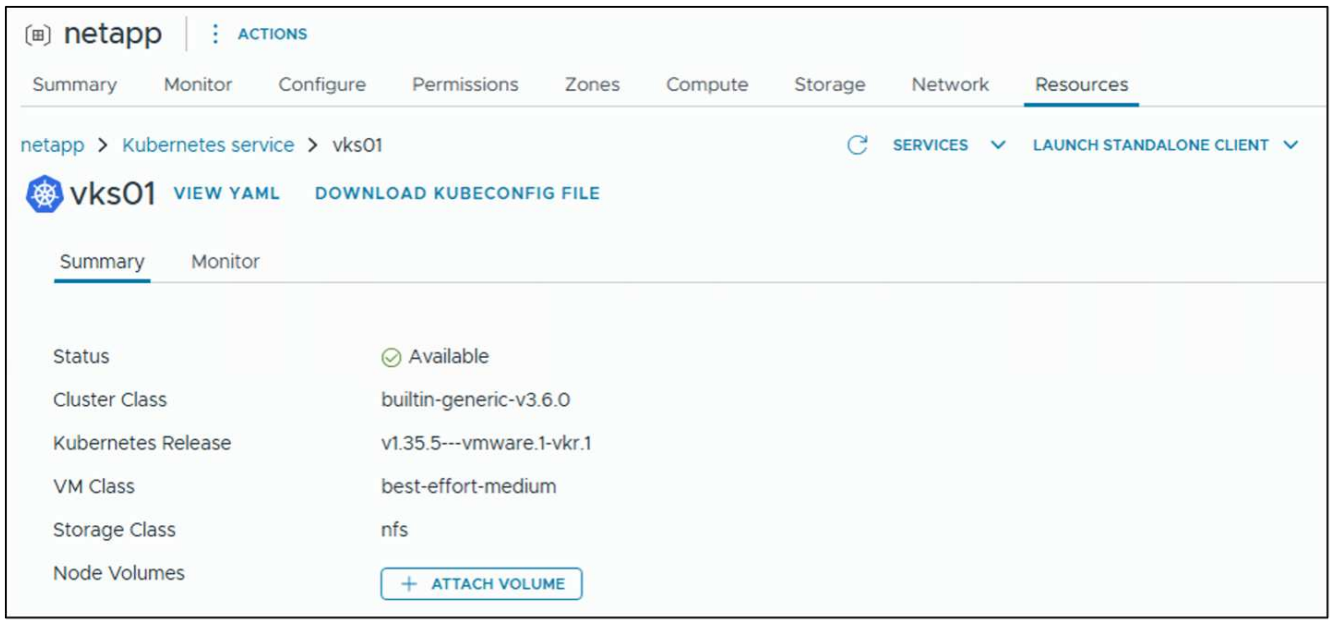
b. Selecione a guia **Computação** e, em seguida, **Clusters do Kubernetes**. Todos os clusters no namespace são exibidos.

c. Vá até a aba **Recursos**, selecione o cluster Kubernetes que você precisa e faça o download do seu arquivo kubeconfig.









3. A partir de um host bastion, conecte-se ao cluster usando o arquivo kubeconfig para gerenciá-lo pela linha de comando.

```
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
```

Demonstração em vídeo

Os vídeos a seguir demonstram duas maneiras de criar um cluster Kubernetes vSphere.

Crie um cluster Kubernetes do vSphere em um Supervisor Cluster

Crie um cluster Kubernetes do vSphere com a automação do VCF

Saiba mais sobre o storage da NetApp para o vSphere Kubernetes Service

Usar o NetApp como storage para o vSphere Kubernetes Service (VKS) é uma combinação poderosa que aproveita as robustas soluções de storage do NetApp para aprimorar o desempenho, a escalabilidade e a confiabilidade das cargas de trabalho do Kubernetes. O NetApp Trident, um provisionador de storage dinâmico de código aberto para Kubernetes, é usado para integrar o storage às cargas de trabalho no vSphere Kubernetes Service.

Principais benefícios de usar o armazenamento NetApp com VKS

1. **Provisionamento dinâmico de armazenamento:** NetApp Trident permite o provisionamento dinâmico de volumes de armazenamento, possibilitando que os pods do Kubernetes solicitem storage em tempo real, com base em suas necessidades.
2. **Storage persistente**: As soluções NetApp oferecem recursos de storage persistente, garantindo a durabilidade e a disponibilidade dos dados mesmo após reinicializações e falhas de pods.

3. **Alto desempenho:** As soluções de storage da NetApp, como o ONTAP, oferecem storage de alto desempenho com baixa latência, o que é ideal para aplicações com uso intensivo de E/S executadas no Kubernetes.
4. **Escalabilidade:** Os sistemas de armazenamento NetApp podem ser dimensionados para atender às demandas de cargas de trabalho crescentes do Kubernetes, suportando implantações em larga escala.
5. **Gerenciamento de dados:** NetApp oferece recursos avançados de gerenciamento de dados, incluindo snapshots, clonagem e replicação de dados, que podem ser benéficos para backup, recuperação de desastres e fluxos de trabalho de desenvolvimento.
6. **Suporte a várias nuvens e nuvem híbrida:** NetApp pode ter suas soluções de armazenamento implementadas em ambientes locais, de nuvem pública e de nuvem híbrida, proporcionando flexibilidade e consistência no gerenciamento de storage.

NetApp ONTAP visão geral da integração

NetApp ONTAP oferece armazenamento de nível empresarial com recursos como deduplicação de dados, compactação e criptografia. Você usa NetApp Trident para integrar o armazenamento NetApp ao Kubernetes. NetApp Trident é compatível com várias plataformas de armazenamento NetApp, incluindo o ONTAP local, o FSx para NetApp ONTAP na AWS, o Azure NetApp Files no Azure e o Google Cloud NetApp Volumes no Google Cloud.

Você utiliza o Trident Protect para gerenciar a proteção de dados e a recuperação de desastres para suas cargas de trabalho do Kubernetes. O Trident Protect permite que você crie Snapshots, clones e replique dados em diferentes backends de armazenamento, garantindo que seus dados estejam seguros e possam ser recuperados em caso de falhas.

Principais características do Trident

Conformidade com CSI Garantindo compatibilidade com os recursos e padrões de armazenamento mais recentes do Kubernetes. **Configuração declarativa** Permitindo que você defina os requisitos de armazenamento em arquivos YAML, que são então gerenciados automaticamente pelo Trident. **Drivers** O Trident oferece suporte a drivers para os protocolos NFS, CIFS/SMB, iSCSI, FC e NVMe/TCP suportados pelo ONTAP. **Suporte a ambientes híbridos e várias nuvens** O Trident oferece suporte a drivers para storage ONTAP on-premises e serviço de storage gerenciado nos hiperescaladores, como FSx para NetApp ONTAP na AWS, Azure NetApp Files no Azure e Google Cloud NetApp Volumes no Google Cloud. **Gerenciamento avançado de dados** Aproveitando os recursos de Snapshot e clonagem do ONTAP para proteção eficiente de dados e provisionamento rápido de ambientes de teste e desenvolvimento.

Para obter detalhes completos sobre o Trident, consulte o "Documentação do Trident".

Trident Protect

Trident Protect é instalado separadamente do Trident. Antes da implantação, verifique se sua plataforma Kubernetes, backend de armazenamento, componentes de Snapshot e storage de objetos atendem aos "Requisitos do Trident Protect".

Principais características do Trident Protect

Backups automatizados O Trident Protect permite backups automatizados e voltados a políticas de aplicativos Kubernetes, garantindo que eles sejam copiados regularmente sem intervenção manual.

Gerenciamento de Snapshots Ele aproveita os recursos de snapshot do ONTAP para criar cópias frequentes de ponto no tempo de aplicativos em contêineres e VMs, fornecendo um mecanismo confiável de recuperação.

Criptografia do repositório de backup O Trident Protect oferece suporte à criptografia baseada em senha para os repositórios de backup Restic e Kopia. Configure a criptografia de volume persistente e em trânsito separadamente nas camadas de storage e rede.

Conformidade e Auditoria Oferece ferramentas e recursos que ajudam as organizações a atender aos requisitos de conformidade e realizar auditorias regulares de suas práticas de proteção de dados.

Recuperação de Desastres integra-se aos recursos de replicação do ONTAP para fornecer soluções robustas de recuperação de desastres, garantindo a continuidade dos negócios mesmo diante de eventos catastróficos.

Para obter detalhes completos sobre o Trident Protect, consulte o "Documentação do Trident Protect".

NetApp Modelos de hardware e protocolos ONTAP suportados

NetApp ONTAP software é executado em diversos modelos de hardware, cada um projetado para atender a diferentes requisitos de desempenho e capacidade. Trident amplia seus recursos robustos para oferecer suporte a vários sistemas NetApp ONTAP, incluindo All Flash FAS (AFF), All SAN Array (ASA) e ASA R2. Esse suporte garante que as organizações possam aproveitar todo o potencial de suas soluções de storage de alto desempenho em ambientes conteinerizados.

Sistemas All Flash FAS (AFF) Os sistemas AFF oferecem desempenho excepcional e baixa latência, tornando-os ideais para aplicações de alta demanda.

All SAN Array (ASA) Systems Os sistemas ASA são projetados para ambientes SAN dedicados, oferecendo desempenho robusto e confiabilidade.

ASA R2 Systems Os sistemas ASA R2 oferecem recursos e capacidades aprimorados para ambientes SAN.

AFX NetApp AFX é uma arquitetura de storage all-flash desagregada, desenvolvida especificamente para cargas de trabalho de IA corporativas. Oferece NAS de alto desempenho, permitindo o escalonamento independente de computação e capacidade. Os sistemas de armazenamento AFX da NetApp diferem de outros sistemas baseados em ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage. AFX utiliza um sistema de arquivos distribuído que permite alto desempenho e escalabilidade, enquanto outros sistemas baseados em ONTAP utilizam uma arquitetura de storage tradicional.

1. Protocolos suportados para AFF: NFS, CIFS/SMB, iSCSI, FC, NVMe/TCP
2. Protocolos suportados para ASA: iSCSI, FC, NVMe/TCP
3. Protocolos suportados para ASA R2: iSCSI, FC, NVMe/TCP
4. Protocolos suportados para AFX: a partir do Trident 25.10, apenas o protocolo NFS é suportado; o protocolo SMB não é suportado.

Drivers Trident para storage ONTAP

O Trident inclui os seguintes drivers CSI, cada um capaz de provisionar um volume no storage de backend.

Para protocolos NAS em modelos de hardware compatíveis

1. ontap-nas: Um PVC corresponde a um PV, que é um FlexVol volume dedicado.
2. ontap-nas-economy: Cada PV provisionado é uma qtree, com um número configurável de qtrees por FlexVol volume (o padrão é 200, configurável entre 50 e 300).

   Um PVC corresponde a um PV, que é uma qtree em um volume FlexVol compartilhado.

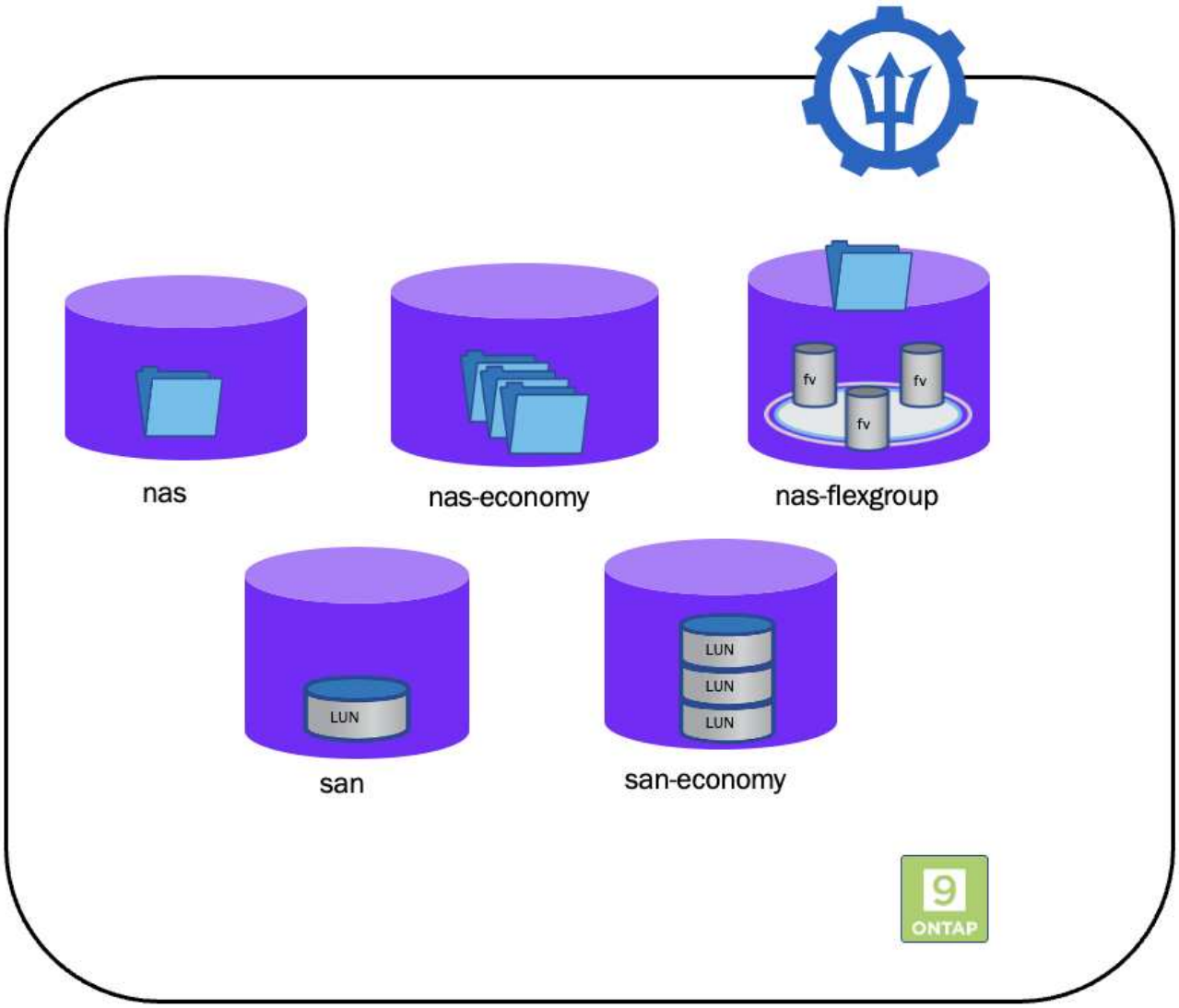
Você pode colocar todos os discos das VMs como qtrees em um único volume. No entanto, esteja ciente de que os recursos de Snapshot, clonagem e replicação não são suportados pelo Trident. Quando esses recursos são gerenciados pelo administrador de storage, eles não são granulares em nível de PV.

1. ontap-nas-flexgroup: Cada PV provisionado é um ONTAP FlexGroup, e todos os agregados atribuídos a um SVM são utilizados.

Para protocolos SAN em modelos de hardware compatíveis

1. ontap-san: Um PVC corresponde a um PV, que é um LUN em um FlexVol volume dedicado.
2. ontap-san-economy: Um PVC corresponde a um PV, que é um LUN em um volume FlexVol compartilhado. Você pode ter vários LUNs no volume FlexVol compartilhado (o padrão é 100, configurável entre 50 e 200 LUNs/PVs por volume FlexVol).

Você pode colocar todos os discos das VMs como LUNs em um único volume. No entanto, este driver suporta Snapshots e Clones, mas não suporta replicação. Quando a replicação é gerenciada pelo administrador de storage, ela não é granular por PV.



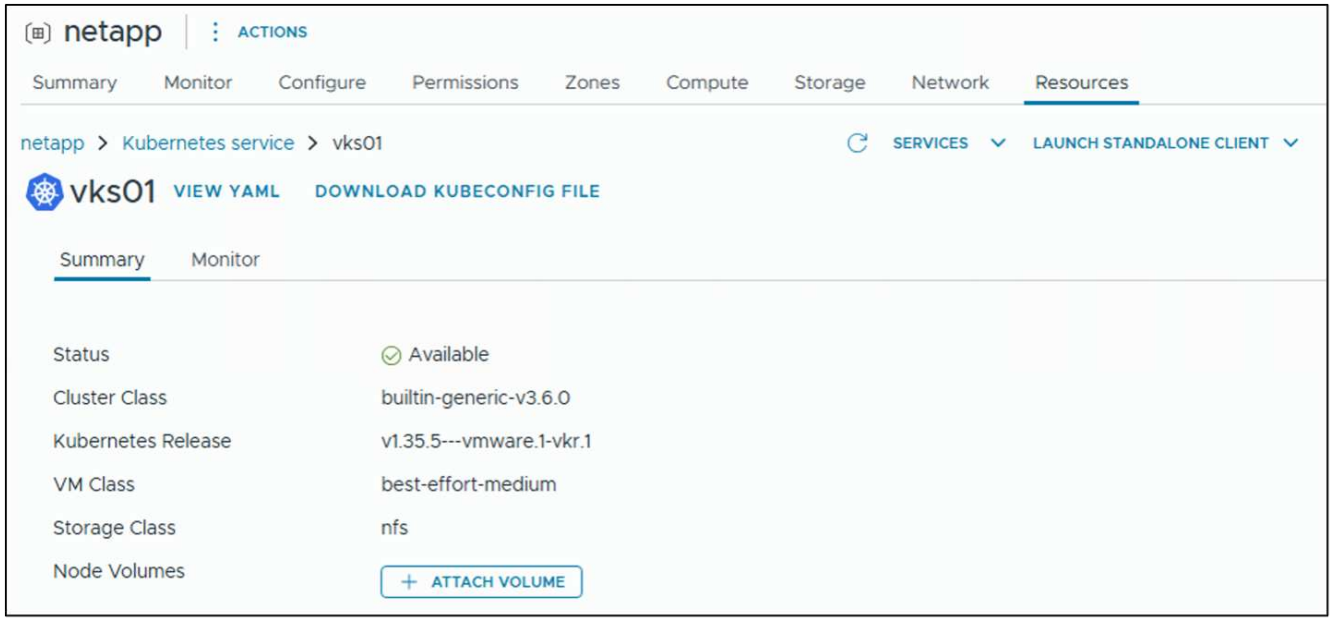
Para obter a lista completa de recursos expostos pelos drivers do Trident e aqueles que devem ser aplicados pelo administrador de storage, consulte a seção "Drivers de backend ONTAP" na documentação do Trident.

Instale o NetApp Trident em um cluster VKS

Instale o NetApp Trident como o provisionador de storage dinâmico no seu cluster do vSphere Kubernetes Service para integrar o storage NetApp ONTAP às suas cargas de trabalho do Kubernetes.

Antes de começar

- Certifique-se de que o seu cluster ONTAP esteja configurado e conectado ao seu ambiente VMware Kubernetes.
- Certifique-se de que seu cluster VKS tenha as ferramentas iSCSI ou NVMe instaladas caso suas cargas de trabalho utilizem esses protocolos para storage.
- Certifique-se de ter `kubectl` instalado em uma máquina a partir da qual você possa se conectar ao cluster VKS usando o arquivo kubeconfig.



Passos

1. Instale o Trident Operator usando um Helm chart, seguindo as instruções na documentação do Trident:

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

   Antes de executar a instalação do Helm, crie e rotule o `trident` namespace. O `enforce=privileged` rótulo é necessário porque o Trident executa cargas de trabalho privilegiadas. Sem ele, o controlador de admissão de segurança de pods do Kubernetes impedirá que os pods do Trident sejam iniciados.

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

   ▼ Mostrar exemplo

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

Para ativar o registro de depuração, adicione `--set tridentDebug=true`:

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

Você também pode instalar o Trident manualmente usando o Trident Operator. Revise os procedimentos na documentação do Trident: "Implante manualmente o Trident Operator"

Certifique-se de ter atribuído os rótulos de segurança de pod necessários ao `trident`namespace antes de instalar o Trident manualmente.

2. Confirme que todos os pods do Trident estão em execução no cluster.

▼ **Mostrar exemplo**

```
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$
```

Se os pods do Trident não iniciarem com um erro de admissão `PodSecurity`, os rótulos de segurança do pod podem não ter sido aplicados ao namespace `trident` antes da instalação. Use `--overwrite` para reaplicá-los e, em seguida, verifique se os pods iniciam.

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

Prepare o backend de armazenamento e os arquivos de configuração StorageClass para o seu ambiente

1. Configure um backend Trident para o ONTAP definindo os detalhes de conexão e os parâmetros de storage necessários.
2. Defina classes de armazenamento no Kubernetes que correspondam aos backends de armazenamento NetApp. Essas classes especificam parâmetros como características de desempenho e políticas de replicação.

Defina os backends e as classes de armazenamento do Trident para os seguintes protocolos, conforme exigido pelas suas cargas de trabalho:

- NAS (driver ontap-nas)
- NAS Economy (driver ontap-nas-economy)
- SAN (driver ontap-san) para os protocolos iSCSI, FC ou NVMe
- SAN Economy (driver ontap-san-economy) para iSCSI

NAS

Crie um TridentBackendConfig e StorageClass para ONTAP NAS para habilitar o provisionamento de storage persistente baseado em NFS. A configuração de backend inclui credenciais armazenadas em um Secret do Kubernetes e faz referência à sua ONTAP SVM e à LIF de gerenciamento.

Exemplo de segredo do backend e arquivo de configuração do backend usando autenticação por nome de usuário e senha (salve 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
```

Exemplo de segredo do backend e arquivo de configuração do backend usando autenticação por certificado de cliente (salve 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 definição (salvar como `sc-nas.yaml`):

`mountOptions` é um parâmetro opcional que especifica opções de montagem adicionais para volumes NFS. A opção `nconnect` permite múltiplas conexões TCP paralelas para uma única montagem NFS, o que pode melhorar o desempenho em cargas de trabalho de alta taxa de transferência. O valor padrão é 1, mas pode ser aumentado até o máximo permitido pelo sistema ONTAP.

O ONTAP suporta até 16 conexões para uma única montagem NFS em um cliente compatível com nconnect. O Trident passa as opções de montagem diretamente para os nós de trabalho do Kubernetes, então escolha um valor compatível tanto com o cliente NFS do nó de trabalho quanto com o ONTAP; o exemplo a seguir usa quatro conexões.

```
# 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
```

Crie um TridentBackendConfig e um StorageClass para o driver ONTAP NAS Economy do ONTAP para habilitar o provisionamento de storage persistente baseado em NFS usando qtrees. A configuração de backend inclui credenciais armazenadas em um Kubernetes Secret e faz referência à sua ONTAP SVM e à LIF de gerenciamento.

Exemplo de segredo do backend e arquivo de configuração do backend usando autenticação por nome de usuário e senha (salve 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 definição (salvar 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
```

Crie um TridentBackendConfig e um StorageClass para o ONTAP SAN com iSCSI para habilitar o provisionamento de storage em bloco baseado em iSCSI. A configuração do backend usa o driver ontap-san e inclui credenciais armazenadas em um segredo do Kubernetes. O parâmetro `sanType` assume o protocolo iSCSI por padrão se omitido.

Segredo do backend e arquivo de configuração do backend usando autenticação por nome de usuário e senha (salve 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
```

Segredo do backend e arquivo de configuração do backend usando autenticação por certificado de cliente (salve 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 definição (salvar 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
```

Crie um TridentBackendConfig e um StorageClass para o ONTAP SAN com NVMe para habilitar o provisionamento de storage em bloco baseado em NVMe. A configuração do backend usa o driver ontap-san e inclui credenciais armazenadas em um Kubernetes Secret. O `sanType` parâmetro deve ser definido como `nvme`.

Segredo do backend e arquivo de configuração do backend usando autenticação por nome de usuário e senha (salve 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
```

Segredo do backend e arquivo de configuração do backend usando autenticação por certificado de cliente (salve 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 definição (salvar 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
```

Consulte a documentação do Trident para obter exemplos adicionais de arquivos YAML e informações sobre os modos de acesso e os modos de volume suportados pelos drivers SAN e NAS.

- "Exemplos usando drivers NAS"
- "Exemplos usando drivers SAN"

Crie um arquivo de configuração VolumeSnapshotClass

Crie uma definição de VolumeSnapshotClass. Essa configuração habilita operações baseadas em snapshots para volumes persistentes.

VolumeSnapshotClass definição (salvar 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
```

Aplique os arquivos de configuração ao seu cluster

Aplique os arquivos de configuração que você criou nas etapas anteriores ao cluster do serviço Kubernetes vSphere usando `kubectl`. Isso cria os segredos necessários, as configurações de backend, as classes de armazenamento e a classe de snapshot em seu cluster.

Passos

1. Aplique os arquivos TridentBackendConfig e StorageClass para cada protocolo que você configurou.

```
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. Verifique se os recursos foram criados com sucesso.

   Verifique os objetos TridentBackendConfig:

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

Verifique os objetos StorageClass:

```
kubectl get storageclass
```

Verifique VolumeSnapshotClass:

```
kubectl get volumesnapshotclass
```

Defina as classes padrão de armazenamento e snapshot do Trident

Defina o Trident StorageClass e VolumeSnapshotClass como padrão no cluster do serviço Kubernetes do vSphere.

Passos

1. Defina o StorageClass padrão do Trident.

   Defina um StorageClass com suporte Trident como padrão do cluster para que PersistentVolumeClaims o utilizem automaticamente quando nenhuma classe de armazenamento for especificada.

   Certifique-se de que apenas um StorageClass esteja definido como padrão. Se outro StorageClass já estiver definido como padrão, defina sua anotação como `false`.

   A partir da linha de comando:

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

2. Defina o Trident VolumeSnapshotClass padrão.

   Defina um VolumeSnapshotClass com suporte do Trident como padrão do cluster para habilitar operações baseadas em snapshots para volumes persistentes. Isso garante que VolumeSnapshots usem automaticamente o driver CSI do Trident quando nenhuma classe de snapshot for especificada.

   Certifique-se de que apenas um VolumeSnapshotClass esteja definido como padrão. Se outro VolumeSnapshotClass já estiver definido como padrão, defina sua anotação como `false`.

   A partir da linha de comando:

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

Use as Persistent Volume Claims do Kubernetes para solicitar storage para cargas de trabalho

O Trident provisiona automaticamente os volumes necessários a partir do storage de backend NetApp. Consulte "Implante cargas de trabalho com storage persistente" para exemplos de implantação de uma carga de trabalho com PVCs em um cluster VKS.

Implante um aplicativo em contêiner com o NetApp storage persistente

Implante um aplicativo em contêiner no seu cluster do vSphere Kubernetes Service usando o NetApp ONTAP storage persistente. Os exemplos a seguir demonstram como implantar um banco de dados PostgreSQL usando storage NAS (ontap-nas) ou storage SAN iSCSI (ontap-san).

A seguinte configuração YAML define POSTGRES_USER / POSTGRES_PASSWORD diretamente no manifesto. Em produção, recomenda-se usar os Segredos do Kubernetes para gerenciar informações sensíveis, como credenciais de banco de dados.

Implantar o PostgreSQL com storage NAS (driver ontap-nas)

Você pode implantar um aplicativo de contêiner PostgreSQL com suporte em um volume persistente NFS provisionado pelo driver ontap-nas. O exemplo a seguir demonstra como criar um PersistentVolumeClaim (PVC) e uma implantação para o 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
```

Implante o PostgreSQL com storage SAN (driver iSCSI ontap-san)

Use o seguinte arquivo YAML para implantar um aplicativo de contêiner PostgreSQL com suporte em um volume persistente iSCSI provisionado pelo driver ontap-san. A estratégia de implantação `Recreate` garante que o pod seja completamente encerrado antes que um novo seja iniciado, o que é necessário para volumes iSCSI ReadWriteOnce e evita corrupção de dados durante reinicializações do pod. Essa configuração oferece suporte à persistência de dados em casos de exclusão e recriação de pods e é compatível com snapshots de volume do Trident e operações de backup e restauração.

```
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
---
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
spec:
  type: ClusterIP
  ports:
    - port: 5432
      targetPort: 5432
  selector:
    app: postgres
```

Valide a implantação do banco de dados

Após implantar a aplicação, verifique se os pods estão em execução e se o banco de dados está operacional. Use os seguintes comandos para verificar o status dos pods, PVCs e serviços. Certifique-se de que os pods estão em execução e que os PVCs estão vinculados aos volumes apropriados.

```
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:~$
```

Dependendo do protocolo utilizado, o storage de back-end é um volume, qtree ou LUN. Você pode verificar o storage provisionado no sistema ONTAP descrevendo o PersistentVolume (PV) para recuperar o nome interno do volume e, em seguida, usá-lo nos comandos da CLI do ONTAP para obter os detalhes do storage. Use o seguinte comando para descrever o PV e recuperar o nome interno do volume:

```
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 exemplo

```
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>
```

Copie o nome do volume interno da saída e execute o seguinte comando para obter os detalhes do storage a partir da CLI do ONTAP.

Para volumes 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
```

Para LUNs SAN, execute o seguinte comando para obter os detalhes da LUN a partir da CLI do 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::>
```

Se você utilizou `nconnect`opções de montagem na classe de storage NAS ou NAS Economy, você pode verificar o número de conexões TCP ativas do nó de trabalho para o servidor de storage.

Primeiro, liste as configurações do back-end do Trident para encontrar o nome do back-end e, em seguida, descreva-o para recuperar o nome do vserver e o LIF de dados:

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

Em seguida, faça login no cluster ONTAP e execute o seguinte comando, substituindo o LIF de dados da saída acima:

```
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 exemplo

Após os pods estarem em execução, conecte-se ao banco de dados e verifique as operações de dados.

Passos

1. Encaminhe a porta do serviço PostgreSQL para sua máquina local:

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

2. A partir de um terminal diferente, conecte-se ao banco de dados usando psql:

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

3. Crie um banco de dados e uma tabela e, em seguida, preencha a tabela com dados:

```
CREATE DATABASE employee;
```

Alterne para o novo banco de dados (metacomando psql):

```
\c employee
```

Crie a tabela, insira uma linha e consulte os 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;
```

Demonstração em vídeo

O vídeo a seguir demonstra a instalação do Trident em um cluster Kubernetes vSphere e a implantação de um banco de dados PostgreSQL com storage persistente usando o Trident.

Implantando cargas de trabalho em um cluster do serviço Kubernetes do vSphere com o ONTAP storage persistente usando o Trident

Proteção de dados

Faça backup e restaure aplicativos em contêineres em um cluster do vSphere Kubernetes Service usando o NetApp Console

Faça backup e restaure aplicativos em contêineres em um cluster vSphere Kubernetes Service (VKS) usando o NetApp Console. Este procedimento abrange a criação e o gerenciamento de operações de backup e restauração por meio do Backup and Recovery Service via o NetApp Console, usando o storage de objetos ONTAP S3 como destino do backup.

NetApp Console oferece gerenciamento centralizado de armazenamento e serviços de dados em ambientes locais e na nuvem. Ele proporciona controle unificado, simplicidade operacional e gerenciamento seguro. Diversos serviços de dados e recursos de gerenciamento podem ser acessados pela interface de usuário em "console.netapp.com". Para obter detalhes completos sobre NetApp Console, consulte "NetApp Console documentação".

Esta seção se concentra no serviço de dados NetApp Backup and Recovery para workloads do Kubernetes, especificamente aplicações de contêiner em clusters do vSphere Kubernetes Service. O NetApp Backup and Recovery Service permite que você descubra, gerencie, crie e associe políticas de proteção às suas aplicações do Kubernetes usando uma única interface. Para obter um guia completo, consulte "NetApp Backup and Recovery documentação".

Fluxo de trabalho de backup e recuperação

1. **Certifique-se de ter um agente do Console**

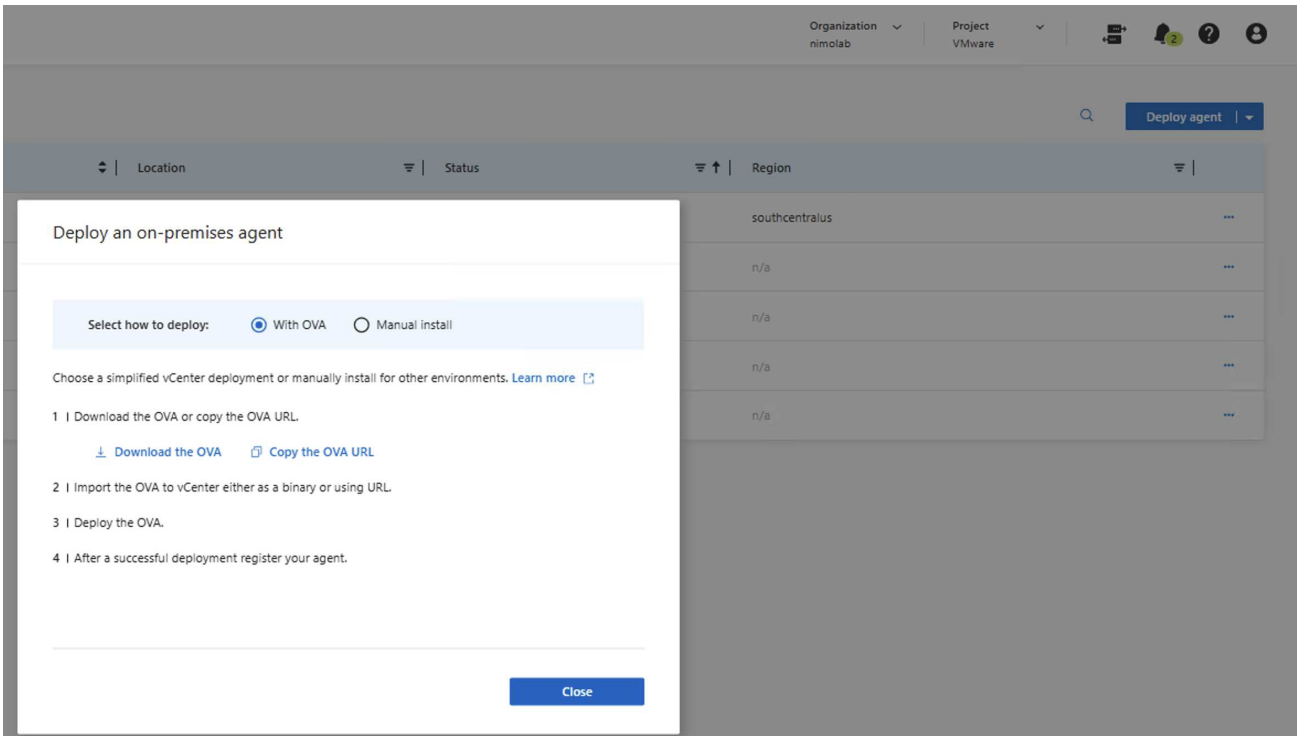
   Certifique-se de ter um agente de Console implantado no ambiente onde seu cluster VKS está em execução. Para obter detalhes sobre como instalar um agente de Console, consulte o "NetApp Console Documentação de configuração".

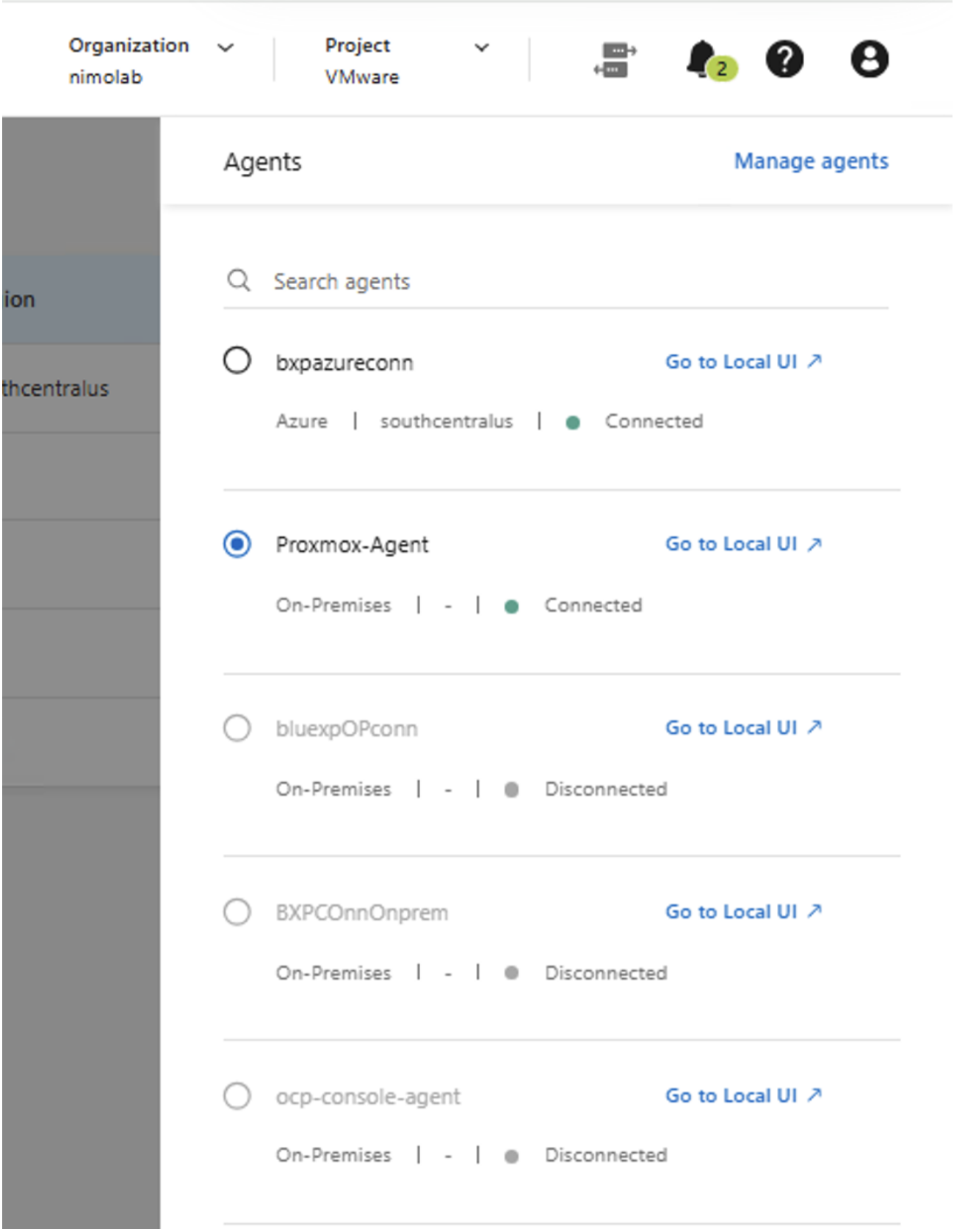
   ▼ **Mostrar exemplo**

   No NetApp Console, clique em Implantar Agente, conforme mostrado na captura de tela, e implante o agente no ambiente onde o cluster VKS está em execução.

   Neste exemplo, selecione **Local > Com OVA**, copie a URL do OVA e use essa URL no vCenter para implantar o agente do Console.

   Registre-o usando o endereço IP do agente na URL. Veja "a documentação" para instruções detalhadas. Após o agente ser registrado, você pode vê-lo no NetApp Console, conforme mostrado na captura de tela abaixo.


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. **Adicione o cluster ONTAP ao NetApp Console e habilite o NetApp Backup and Recovery**

   O cluster ONTAP primário deve ser adicionado ao NetApp Console e o serviço de backup e restauração deve ser ativado nele antes que as cargas de trabalho possam ser protegidas. Para obter instruções, consulte "Configurar o NetApp Backup and Recovery".

3. **No NetApp Console, descubra o cluster Kubernetes do vSphere**

▼ **Mostrar exemplo**

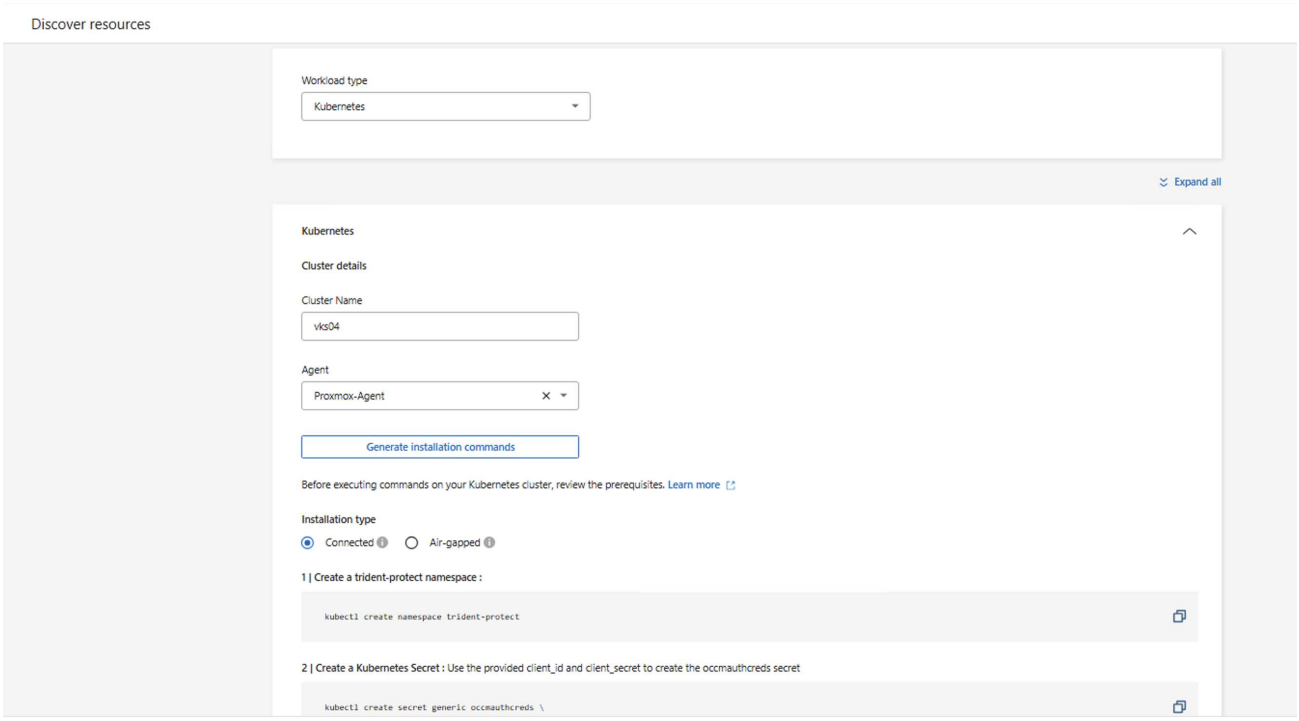
Esta etapa requer que o cluster Kubernetes do vSphere esteja em execução e que o agente do Console esteja implantado no mesmo ambiente. Siga estas etapas para descobrir o cluster Kubernetes do vSphere no NetApp Console.

a. No NetApp Console, selecione **Inventário > Kubernetes** e forneça os detalhes do cluster Kubernetes do vSphere.

b. Selecione o agente implantado no mesmo ambiente em que o vSphere cluster Kubernetes está localizado.

c. Copie os comandos fornecidos para instalar o Trident Protect e execute-os a partir de uma máquina conectada ao seu vSphere cluster Kubernetes.

d. Após a instalação bem-sucedida do Trident Protect, clique em **Descobrir**. O NetApp Console descobre o cluster Kubernetes do vSphere e todos os seus recursos por meio do agente e os exibe na interface do usuário.



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
```

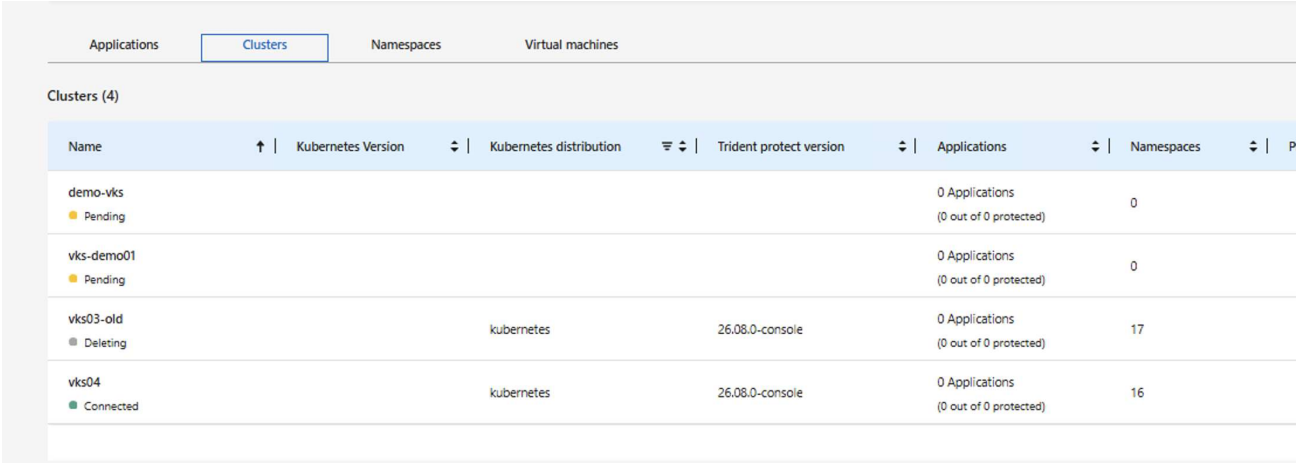
3 | Add/update helm repo :

```
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 o ONTAP S3 como destino de backup**

   Neste exemplo, usamos o ONTAP S3 storage de objetos como o destino de backup. Você também pode usar o AWS S3, o Azure Blob, o Google Cloud Storage ou o StorageGRID como destinos de backup.

   Para obter mais detalhes, consulte "NetApp Console documentação".

   Para adicionar o ONTAP S3 como destino de backup no NetApp Console, você precisará das seguintes informações do seu cluster ONTAP.

```
S3 Endpoint:
```

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

Para obter esses valores, faça o seguinte no seu ONTAP cluster usando o System Manager.

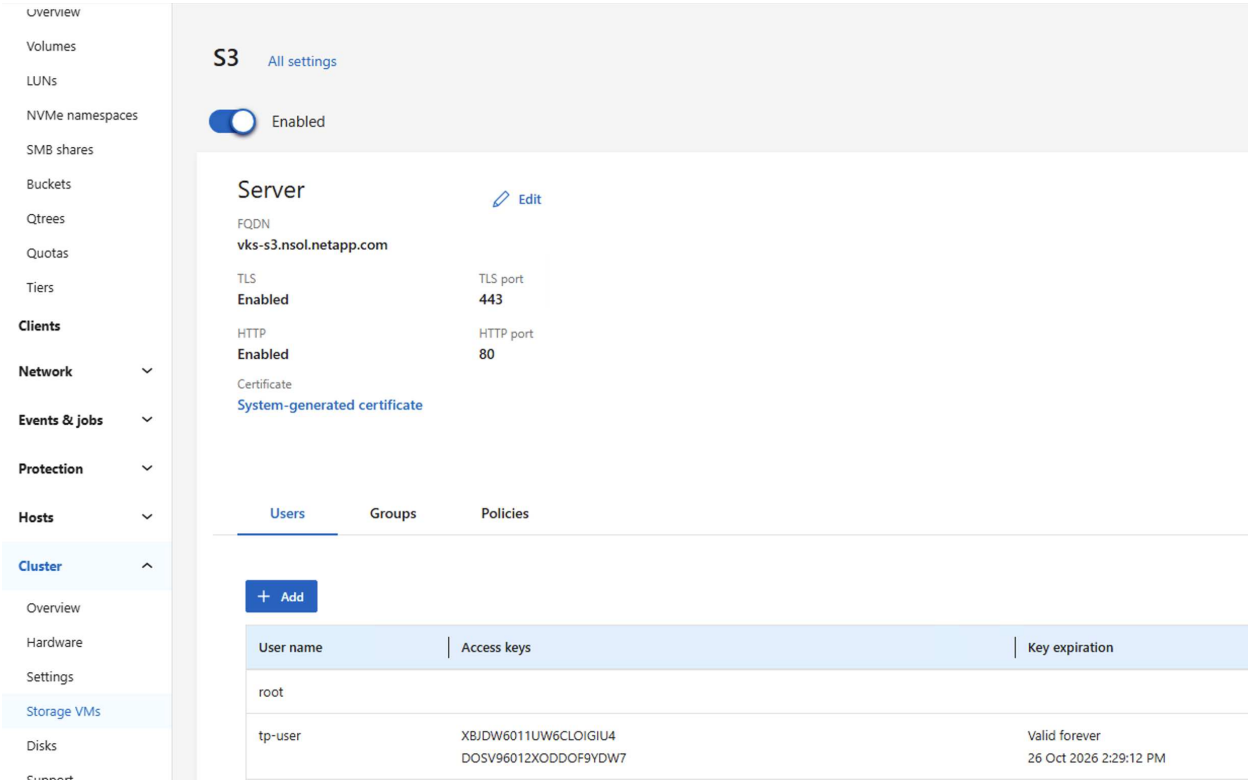
a. Crie uma SVM habilitada para S3 no cluster ONTAP para armazenar os dados de backup. Anote a URL do endpoint S3 para esta SVM.

b. Nas configurações do S3 da SVM, crie um usuário e atribua o acesso necessário. Anote a chave de acesso e a chave secreta desse usuário.

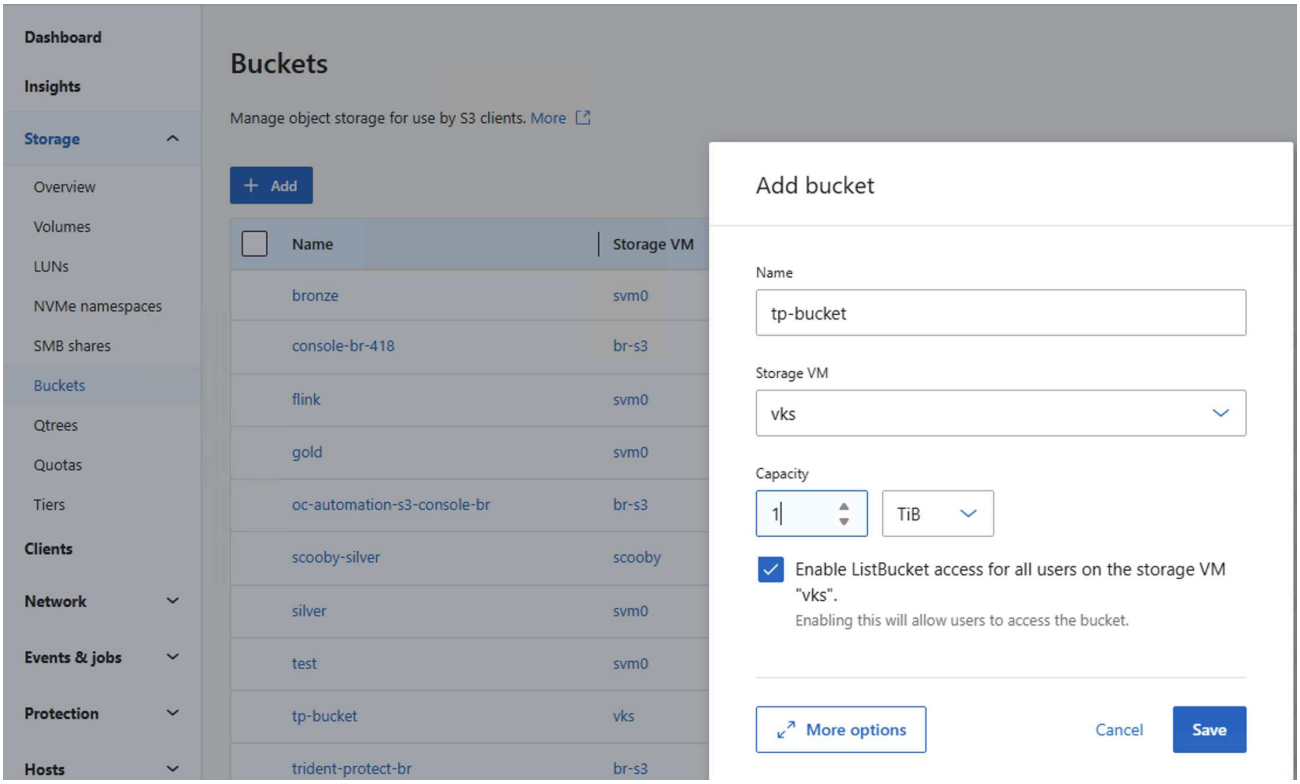
> O Trident Protect exige que o usuário do S3 tenha pelo menos `PutObject`, `GetObject`, `ListBucket` e `DeleteObject` permissões. Sem essas permissões, as operações de backup e restauração do AppVault falham com erros de acesso negado. Para obter detalhes, consulte "Documentação do Trident Protect AppVault".

▼ **Mostrar exemplo**

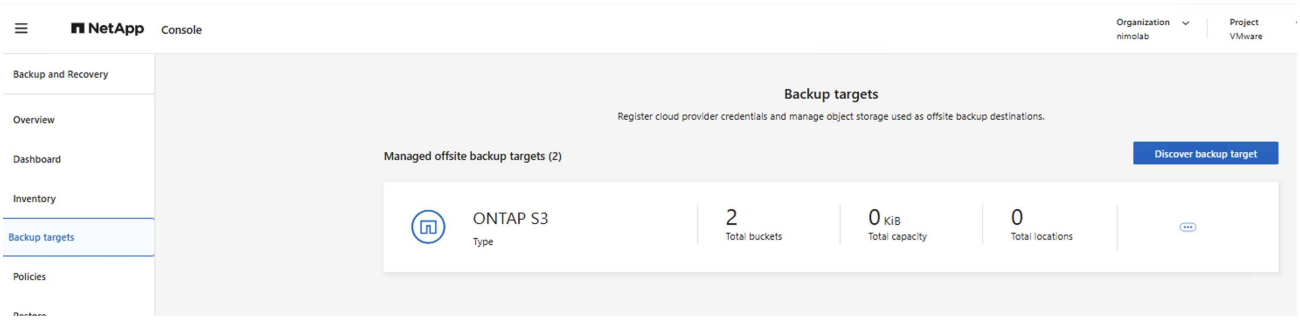
i. **Crie um usuário S3 e baixe a chave de acesso e a chave secreta**

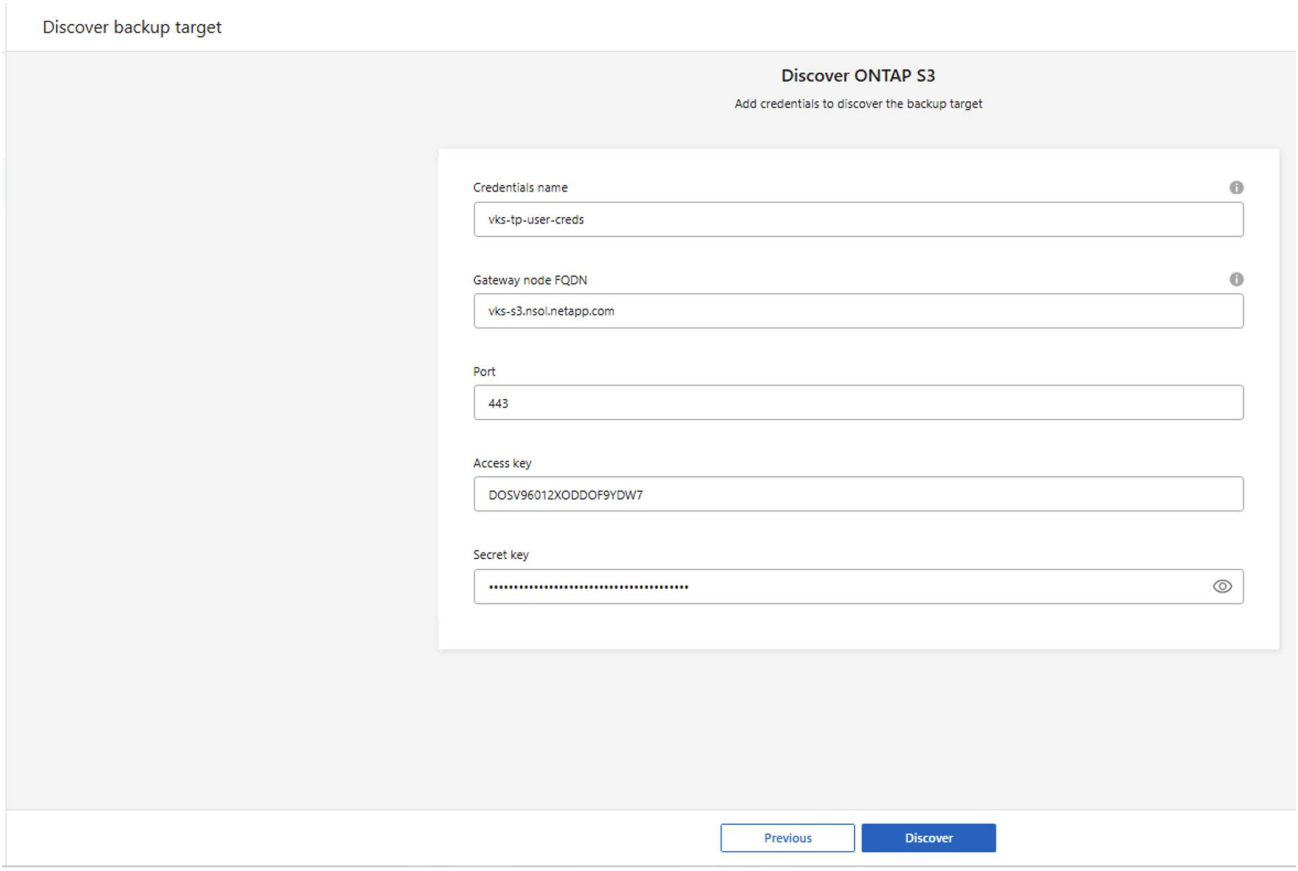


A. Crie um bucket no SVM habilitado para S3. Anote o nome do bucket para usar ao adicionar o storage de objetos ONTAP S3 como um destino de backup no NetApp Console.



Adicione o ONTAP storage de objetos como um destino de backup no NetApp Console usando o endpoint S3, a chave de acesso, a chave secreta e o nome do bucket.





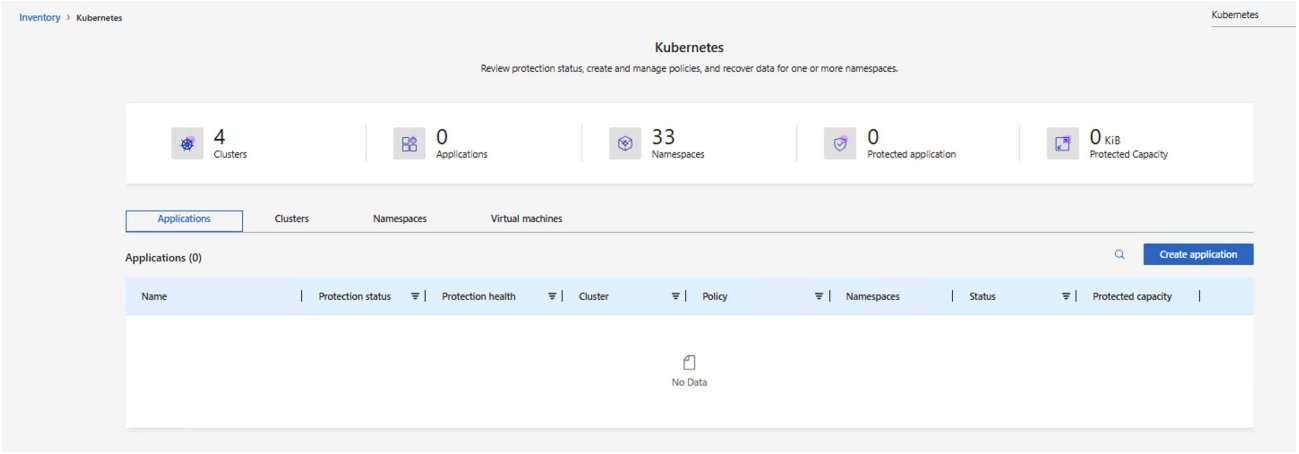
5. **No NetApp Console, crie um aplicativo para as cargas de trabalho do contêiner e proteja-o usando backups.**

> Para criar e proteger uma aplicação Kubernetes, você precisa da função de superadministrador de Backup and Recovery ou de administrador de backup de Backup and Recovery. Para restaurar uma aplicação Kubernetes, você precisa da função de superadministrador de Backup and Recovery ou de administrador de restauração de Backup and Recovery.

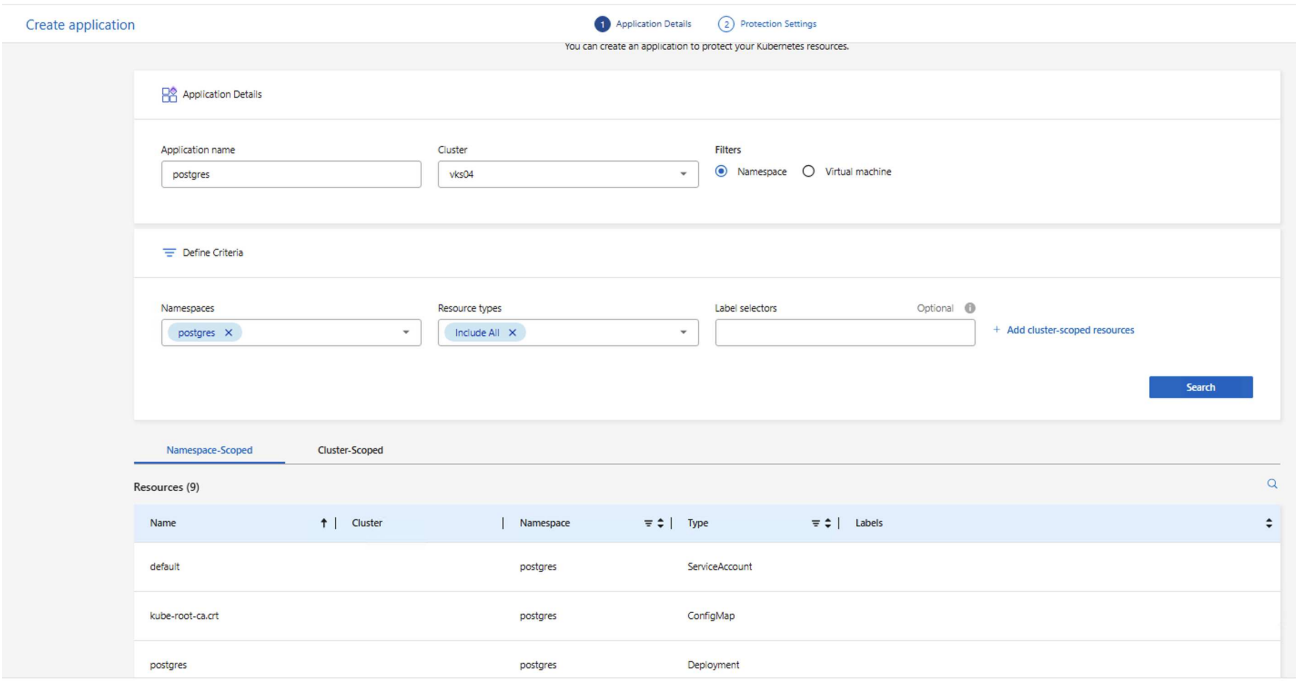
▼ **Mostrar exemplo**

Nesta etapa, crie um aplicativo no NetApp Console para os aplicativos em contêiner no seu cluster Kubernetes vSphere que precisam ser protegidos. Você pode usar namespaces para criar um aplicativo para proteger todos os aplicativos em contêiner em um namespace no cluster Kubernetes vSphere.

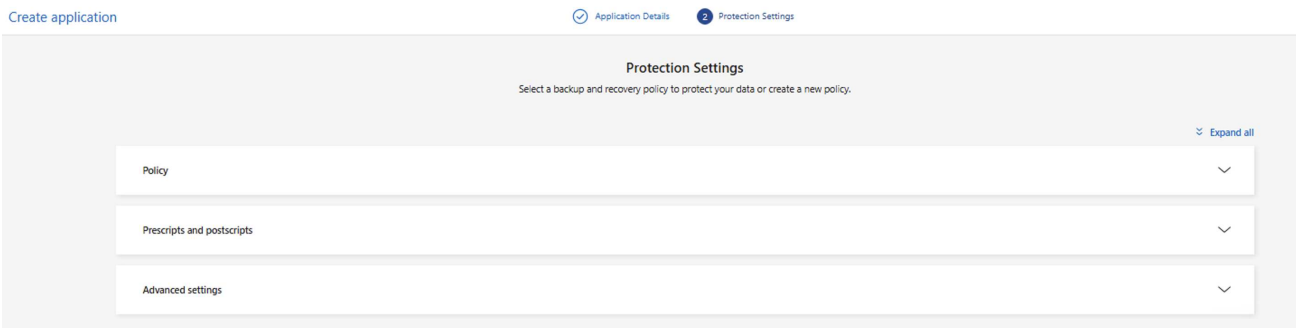
Você também precisará de uma política de proteção para associar ao aplicativo. Você pode criar uma nova política de proteção ou usar uma existente.



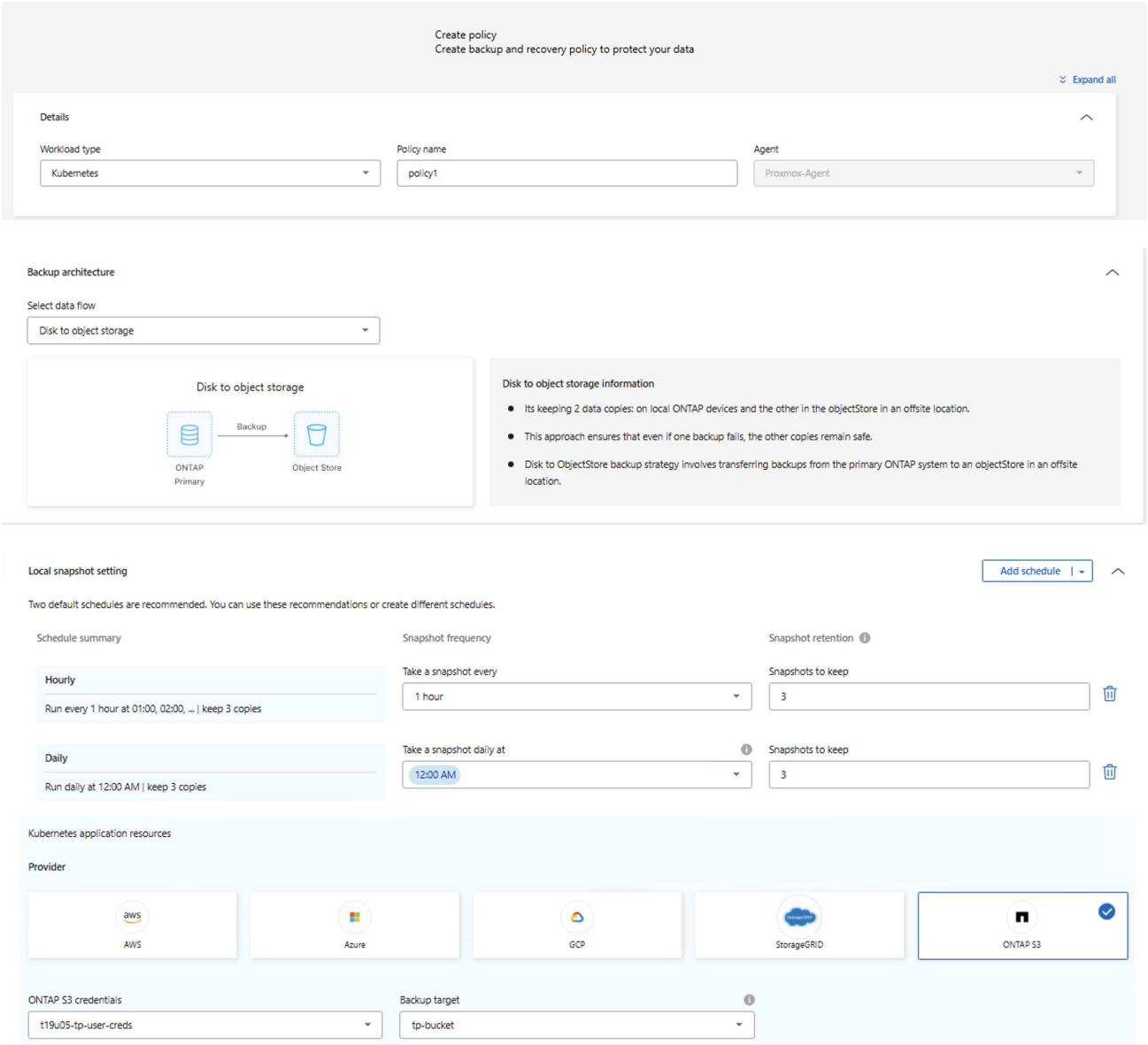
Forneça os detalhes do aplicativo e defina os critérios para o aplicativo. Neste exemplo, o aplicativo é criado para todos os aplicativos de contêiner em um namespace no cluster Kubernetes vSphere. Clique em **Pesquisar** para ver a lista de recursos com escopo de namespace que fazem parte do aplicativo. Clique em **Avançar** para prosseguir.

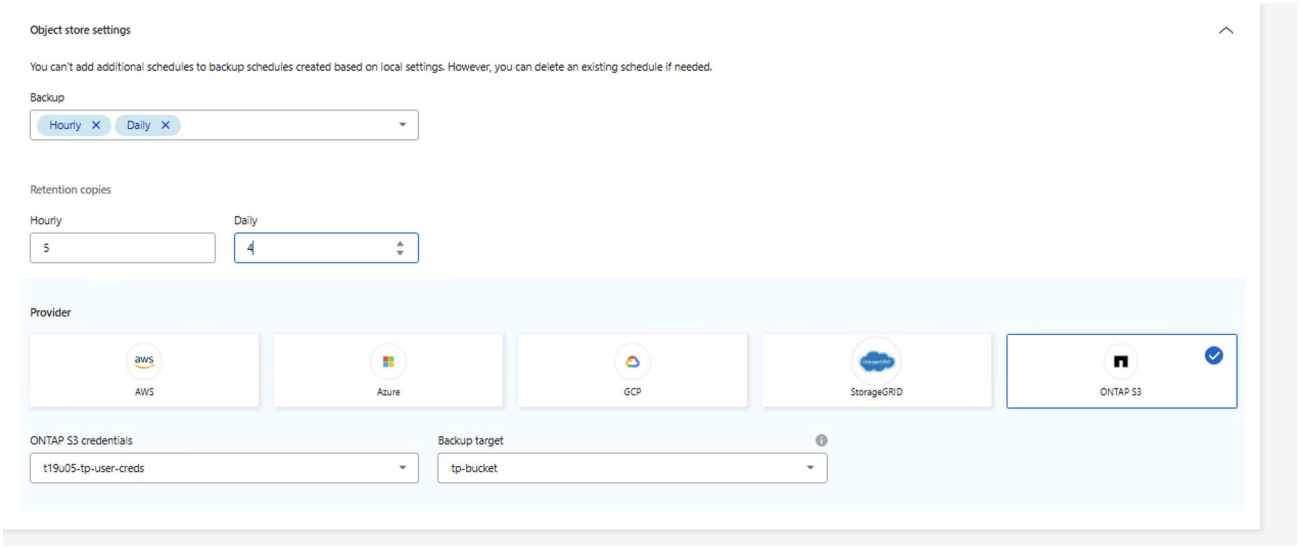


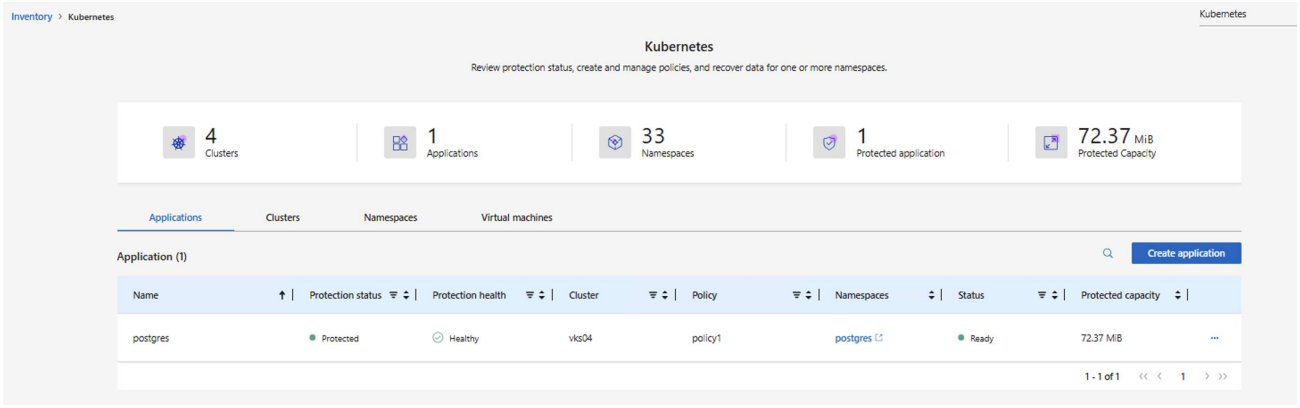
Em seguida, você precisa fornecer as configurações de proteção para o aplicativo. Você pode criar uma nova política de proteção ou usar uma existente. Neste exemplo, uma nova política de proteção é criada para o aplicativo e associada a ele. Expanda **Política** e clique em **Criar nova política**.



Forneça um nome para a política e escolha a arquitetura de backup. Neste exemplo, disco para storage de objetos é escolhido. Defina a programação de snapshots locais, que controla com que frequência os snapshots são criados no storage primário. Em seguida, configure separadamente as configurações de backup do storage de objetos: escolha o destino de backup (neste exemplo, ONTAP S3), defina a programação de transferência que controla quando os dados do snapshot são copiados para o storage de objetos e especifique o número de cópias de retenção. A programação de transferência é independente da programação de snapshots, então os recursos são copiados para o storage de objetos de acordo com a programação de transferência, e não imediatamente quando cada snapshot é criado. Você também pode modificar o número máximo de tentativas e o intervalo de repetição, se necessário. Clique em **Criar** para prosseguir. O aplicativo será detectado e a política de proteção será associada a ele.





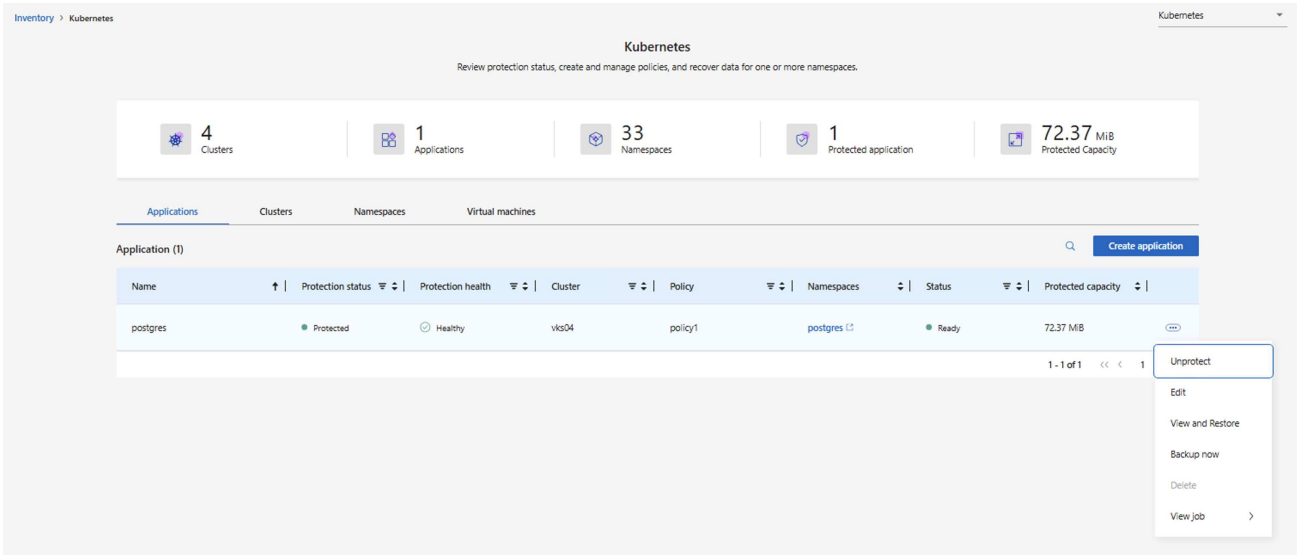


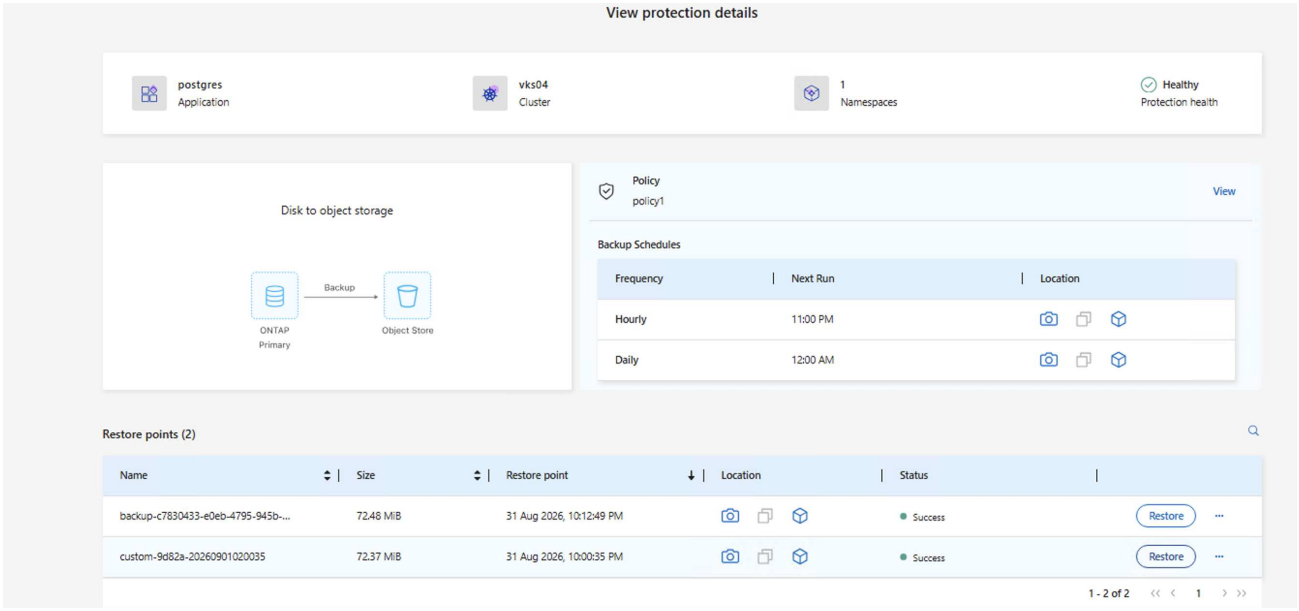
6. **Restaure o aplicativo a partir de um backup**

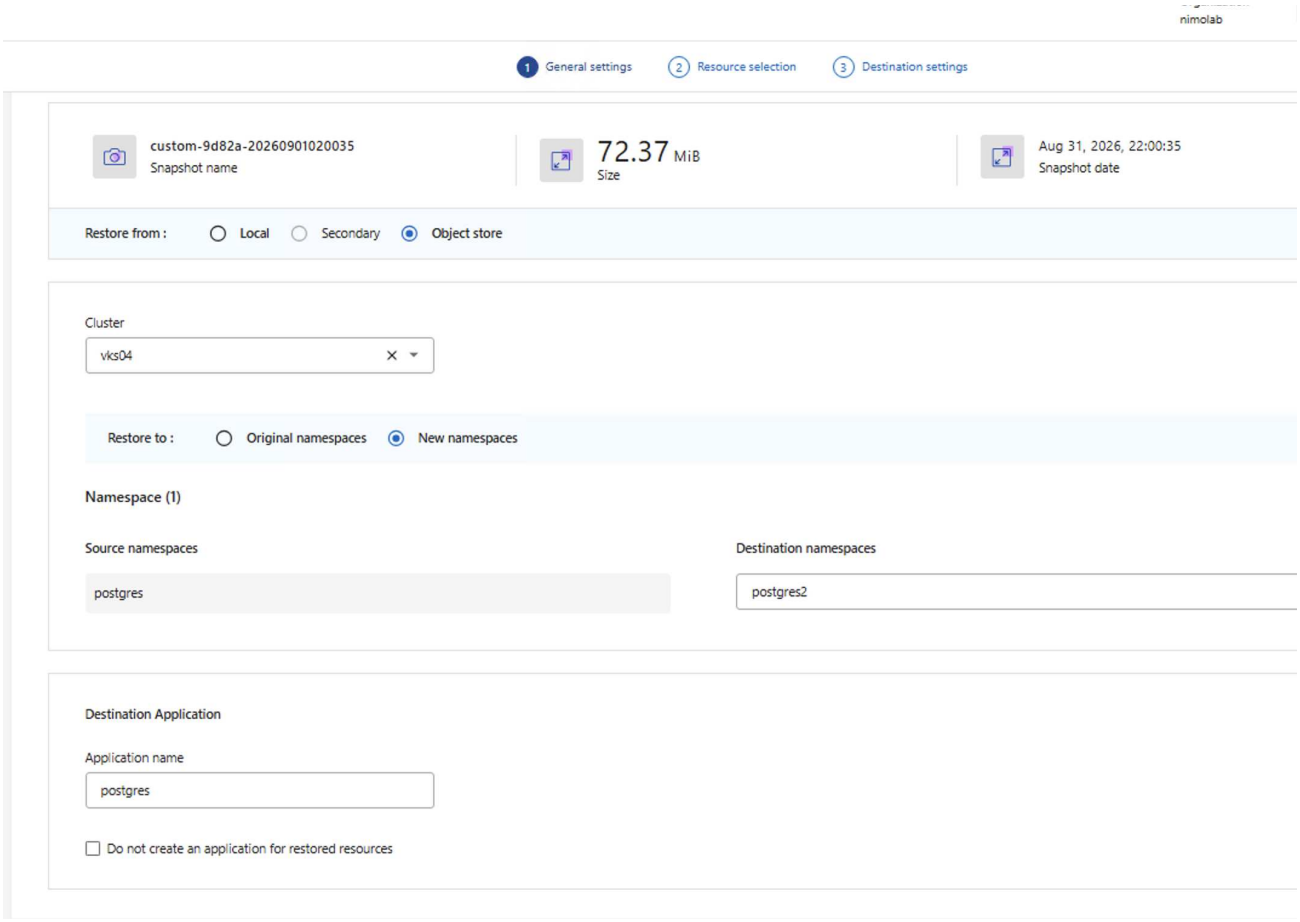
> Você precisa da função de superadministrador de Backup and Recovery ou de administrador de restauração de Backup and Recovery para restaurar um aplicativo Kubernetes.

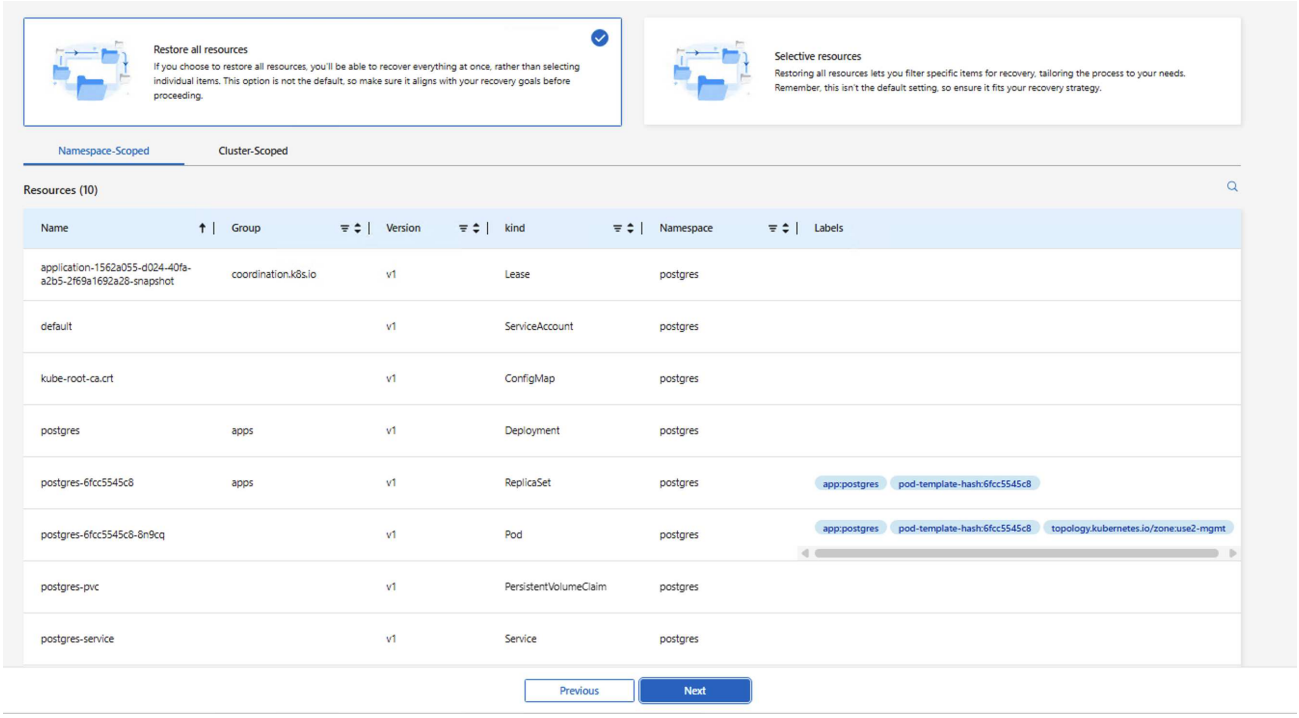
▼ Mostrar exemplo

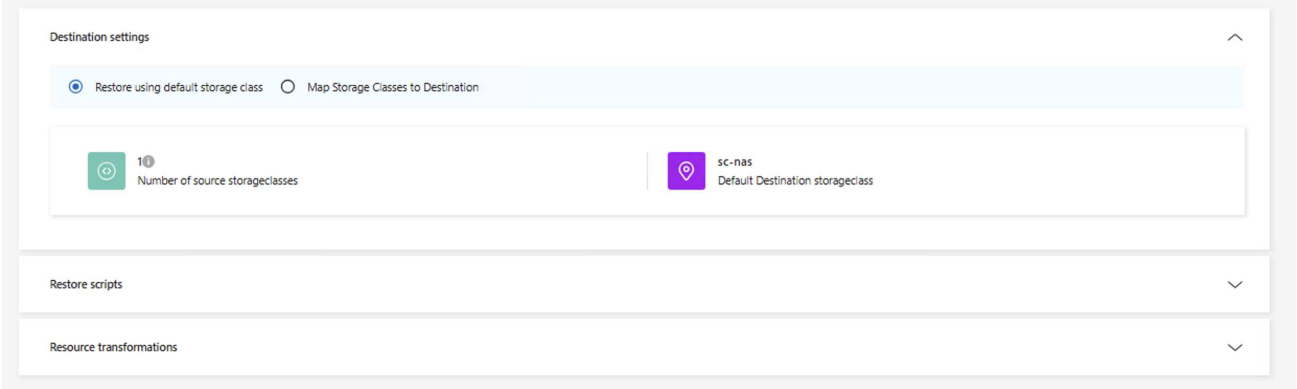
- Você pode escolher qualquer ponto de restauração disponível — um snapshot local, um backup em storage de objetos ou um backup de um destino secundário — para restaurar o aplicativo.
- Você pode restaurar o aplicativo para o seu namespace original ou para um namespace diferente.
- Você pode selecionar quais recursos incluir na restauração.
- Você pode escolher a classe de armazenamento a ser usada para os recursos restaurados do aplicativo.


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
1
Unprotect
Edit
View and Restore
Backup now
Delete
View job

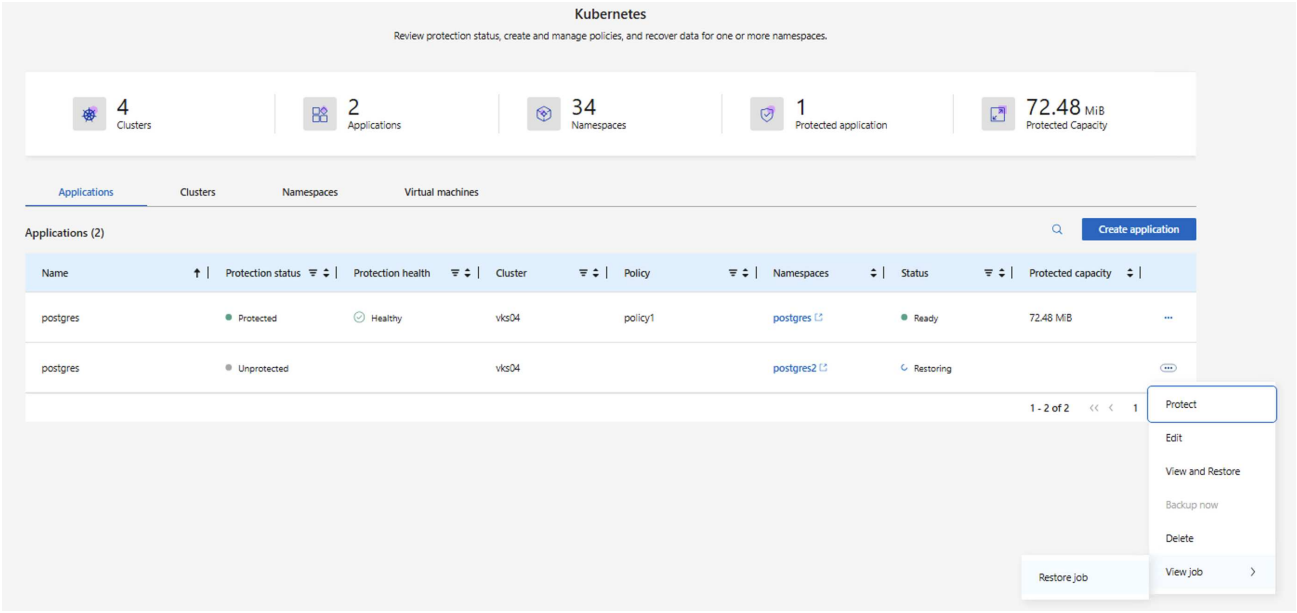

View protection details
postgres
Application
vks04
Cluster
1
Namespaces
Healthy
Protection health
Disk to object storage
Backup
ONTAP
Primary
Object Store
Policy
policy1
View
Backup Schedules
Frequency
Next Run
Location
Hourly
11:00 PM
Daily
12:00 AM
Restore points (2)
Name
Size
Restore point
Location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

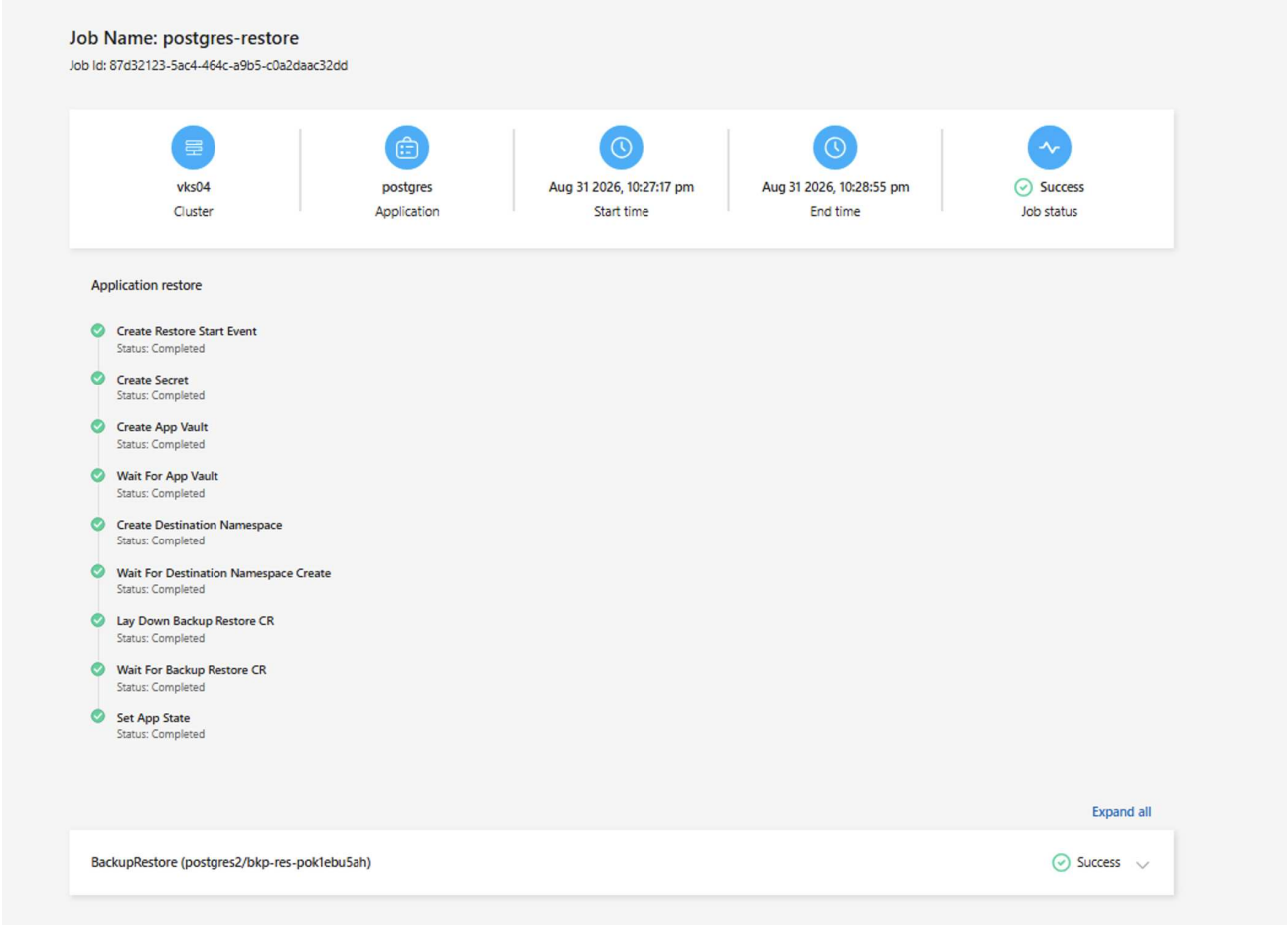

nimolab
1 General settings
2 Resource selection
3 Destination settings
custom-9d82a-20260901020035
Snapshot name
72.37 MiB
Size
Aug 31, 2026, 22:00:35
Snapshot date
Restore from :
Local
Secondary
Object store
Cluster
vks04
Restore to :
Original namespaces
New namespaces
Namespace (1)
Source namespaces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

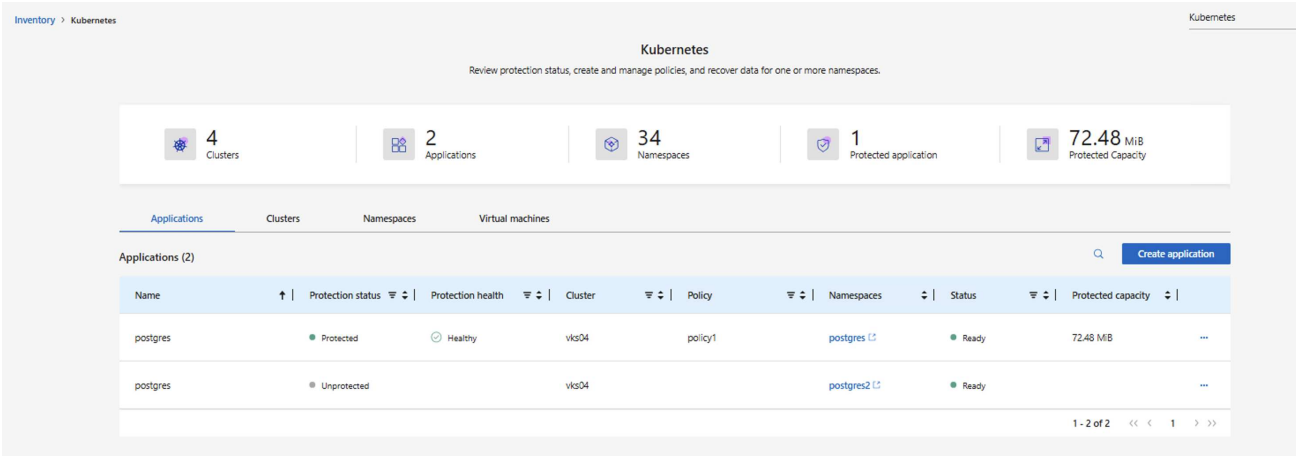

Restore all resources
If you choose to restore all resources, you'll be able to recover everything at once, rather than selecting individual items. This option is not the default, so make sure it aligns with your recovery goals before proceeding.
Selective resources
Restoring all resources lets you filter 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



Você pode acompanhar o progresso da tarefa de restauração na seção Monitor do NetApp Console. Após a conclusão da tarefa de restauração, você pode ver os recursos restaurados no cluster Kubernetes do vSphere.







Faça backup e restaure aplicativos em contêineres em um cluster do vSphere Kubernetes Service usando o Trident Protect

Faça backup e restaure aplicativos em contêineres em um cluster vSphere Kubernetes Service (VKS) usando snapshots e backups do Trident Protect. Este procedimento inclui a criação de um AppVault usando o ONTAP S3 storage de objetos, configurar o Trident Protect para capturar dados do aplicativo, incluindo objetos de recursos do Kubernetes e volumes persistentes, e restaurar os dados quando necessário.

Os aplicativos em contêineres executados em um cluster VKS são gerenciados como cargas de trabalho do Kubernetes em namespaces de nós de trabalho. É importante proteger tanto os metadados do aplicativo

quanto os volumes persistentes para que, caso sejam perdidos ou corrompidos, você possa recuperá-los.

Os volumes persistentes de aplicações VKS podem ser suportados pelo armazenamento ONTAP integrado ao cluster VKS usando "O Trident CSI". Este procedimento utiliza "Trident Protect" para criar snapshots de aplicação, cujos metadados são armazenados no storage de objetos ONTAP enquanto os dados do snapshot do volume permanecem no backend de armazenamento, e backups, cujos dados são copiados para o storage de objetos ONTAP. Você pode restaurar a partir de um snapshot ou de um backup quando necessário.

O Trident Protect permite a criação de snapshots, backups, restauração e recuperação de desastres de aplicações em um cluster Kubernetes. Os dados que podem ser protegidos com o Trident Protect incluem objetos de recursos do Kubernetes associados às aplicações e seus volumes persistentes.

A seguir estão as versões dos diversos componentes utilizados nos exemplos desta seção

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

Instale o Trident Protect no cluster Kubernetes vSphere

▼ Instalar Trident Protect

Use o Helm para instalar o Trident Protect no cluster do vSphere Kubernetes Service. Os exemplos nesta seção usam `tridentctl-protect` o Trident Protect CLI. Instale-o seguindo as instruções no "Documentação da CLI do Trident Protect". Métodos adicionais de instalação do Trident Protect estão documentados no "Documentação do Trident Protect".

Primeiro, crie e nomeie o `trident-protect`namespace. O `enforce=privileged`rótulo é necessário porque o Trident Protect executa cargas de trabalho privilegiadas. Sem ele, o controlador de admissão de segurança de pods do Kubernetes impedirá a inicialização dos pods do Trident Protect.

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

Em seguida, adicione o repositório Helm e instale o Trident Protect no namespace.

```
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
```

Se os pods do Trident Protect não iniciarem com um erro de admissão `PodSecurity`, é possível que os rótulos de segurança do pod não tenham sido aplicados ao namespace `trident-protect` antes da instalação. Use `--overwrite` para reaplicá-los e, em seguida, verifique se os pods iniciam.

```
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$
```

Crie um App Vault para storage de objetos.

▼ **Criar AppVault**

Antes de criar snapshots e backups para um aplicativo, você deve configurar o storage de objetos no Trident Protect. Isso é feito criando um CR AppVault. Somente administradores podem criar e configurar um CR AppVault.

Os objetos AppVault são a representação de recurso personalizado do Kubernetes de um bucket de armazenamento. Um CR de AppVault contém as configurações necessárias para que um bucket seja usado em operações de proteção, como backup, snapshots, operações de restauração e replicação do SnapMirror.

As etapas a seguir criam um AppVault CR configurado para o ONTAP S3: Crie um servidor de armazenamento de objetos S3 na SVM no cluster ONTAP. . Crie um bucket no servidor de armazenamento de objetos. . Crie um usuário S3 na SVM. Guarde a chave de acesso e a chave secreta em um local seguro.

+ NOTA: o Trident Protect exige que o usuário S3 tenha pelo menos `PutObject`, `GetObject`, `ListBucket` e `DeleteObject` permissões. Sem essas permissões, as operações de backup e restauração do AppVault falham com erros de acesso negado. Para obter detalhes, consulte "Documentação do Trident Protect AppVault". . No cluster VKS, crie um segredo para armazenar as credenciais do ONTAP S3. . Crie um objeto AppVault para o ONTAP S3.

Configurar o Trident Protect AppVault para o ONTAP S3

> ⓘ O manifesto abaixo usa HTTPS com validação de certificado habilitada, o que é necessário para produção. Se o endpoint ONTAP S3 usar um certificado assinado por uma CA pública já confiável pelo cluster, nenhuma configuração de certificado adicional é necessária. Se usar um certificado autoassinado ou de uma CA interna, forneça o certificado da CA raiz personalizada usando o campo `rootCA` conforme mostrado no manifesto. Consulte a variante somente para laboratório no final desta seção se você precisar de uma configuração HTTP para testes isolados em ambiente de não produção.

```
# 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 exclusiva para laboratório (HTTP, sem validação de certificado)

Utilize as seguintes `s3` configurações apenas em um ambiente de laboratório isolado, onde não haja dados confidenciais. Não utilize essas configurações em produção.

```
    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$
```

Criar um aplicativo Trident Protect

▼ Criar um aplicativo Trident Protect

Este exemplo aborda a proteção de dados para o aplicativo de exemplo postgres, que está instalado no namespace postgres. Para obter detalhes sobre a instalação, consulte "Implemente cargas de trabalho VKS usando o armazenamento NetApp".

> ⓘ Os PVCs de exemplo do PostgreSQL solicitam o modo de acesso RWO. O modo de acesso deve ser especificado explicitamente no manifesto do PVC; o Trident não seleciona automaticamente um modo de acesso com base no protocolo de storage. O modo RWX é compatível com PVCs baseados em NAS, e os PVCs baseados em SAN podem suportar RWX com configuração adicional. Para mais informações, consulte "Documentação do Trident Protect".

No exemplo, o namespace postgres possui um aplicativo e todos os recursos do namespace são incluídos ao criar o 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
```

Proteja o aplicativo criando um backup

▼ **Criar backups**

Criar um backup sob demanda

Crie um backup do aplicativo postgres-app criado anteriormente, incluindo todos os recursos no namespace postgres. Forneça o nome do appvault onde os backups serão armazenados.

```
# 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
```

Crie backups de acordo com uma programação

Crie um cronograma para os backups especificando a granularidade e o número de backups a serem mantidos.

```
# 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
```

Restauração a partir do backup

▼ **Restaurar a partir de backups**

Restaure o aplicativo no mesmo namespace

Neste exemplo, o backup postgres-backup-on-demand contém o backup para o postgres-app.

Antes de excluir o aplicativo, verifique se o backup que você pretende restaurar está no estado `Completed` correto. Um backup em andamento ou com falha não pode ser usado para restaurar o aplicativo.

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

Após confirmar que o estado do backup é `Completed`, exclua o aplicativo postgres e certifique-se de que os PVCs e os objetos do pod sejam excluídos do namespace "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.
```

Agora, crie um objeto de restauração no local de backup.

```
# 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
```

Verifique se a implantação do aplicativo postgres, o serviço, o pod e os PVCs foram restaurados.

```
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
```

Restaure o aplicativo para um namespace diferente

Primeiro, crie um novo namespace para o qual você deseja restaurar o aplicativo, neste exemplo, postgres2. Um backup criado pelo agendamento agora está disponível para o aplicativo postgres-app. Use esse backup para restaurar o aplicativo para o novo namespace 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 obter o caminho do backup, use o 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
```

Verifique se os objetos da aplicação postgres e os PVCs foram criados no novo namespace 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$
```

Proteja o aplicativo usando Snapshots

▼ Criar Snapshots

Crie um snapshot sob demanda Crie um snapshot para o aplicativo e especifique o AppVault onde os metadados do snapshot serão armazenados. Os dados do snapshot do volume em si não são copiados para o storage de objetos — eles permanecem no backend de storage.

```
# 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$
```

Crie um cronograma para snapshots Crie um cronograma para os snapshots. Especifique a granularidade e o número de snapshots a serem retidos.

```
# 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 a partir de Snapshot

▼ **Restaurar a partir de Snapshot**

Restaure o aplicativo a partir de um Snapshot para o mesmo namespace

Antes de excluir o aplicativo, verifique se o snapshot que você pretende restaurar está no estado `Completed`. Um snapshot em andamento ou com falha não pode ser usado para restaurar o aplicativo.

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

Após confirmar que o estado do snapshot é `Completed`, exclua os objetos do aplicativo e os PVCs para postgres do namespace 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$
```

Crie um objeto de restauração no local a partir do Snapshot.

```
# 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
```

Verifique se os objetos e PVCs da aplicação são criados no namespace 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 a aplicação a partir de um Snapshot para um namespace diferente

Exclua o aplicativo no namespace postgres2 que foi restaurado anteriormente do backup.

```
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$
```

Crie o objeto de restauração do snapshot a partir do snapshot e forneça o mapeamento de namespace.

```
# 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
```

Verifique se os objetos e PVCs da aplicação foram restaurados no namespace 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$
```

Informações sobre direitos autorais

Informações sobre marcas comerciais