vSphere Kubernetes Service with NetApp 스토리지

NetApp container solutions

NetApp
August 25, 2026

목차

vSphere Kubernetes Service with NetApp 스토리지

vSphere Kubernetes Service와 NetApp ONTAP에 대해 알아보십시오

vSphere Kubernetes Service(VKS)는 VMware에서 제공하는 관리형 Kubernetes 솔루션으로, 기업에서 Kubernetes를 사용하여 컨테이너화된 애플리케이션을 배포, 관리 및 확장할 수 있도록 지원합니다. NetApp ONTAP 스토리지와 함께 사용하면 VKS는 클라우드 네이티브 워크로드에 엔터프라이즈급 영구 스토리지, 성능 및 데이터 관리 기능을 제공합니다.

vSphere Kubernetes Service는 일반적으로 VMware Tanzu 등 VMware의 광범위한 에코시스템에 통합되어 있으며, VMware Tanzu는 Kubernetes 관리, 애플리케이션 현대화 및 DevOps 실무를 위한 포괄적인 툴 모음을 제공합니다.

vSphere Kubernetes 서비스의 주요 기능

1. 관리형 **Kubernetes** 클러스터**:** vSphere Kubernetes Service는 Kubernetes 클러스터의 배포 및 관리를 간소화합니다. 클러스터 생성, 업그레이드, 확장 및 모니터링과 같은 작업을 자동화합니다.
2. **VMware** 인프라와의 통합**:** VKS는 VMware vSphere, NSX-T 및 기타 VMware 솔루션과 원활하게 통합되어 조직이 기존 VMware 인프라를 Kubernetes 워크로드에 활용할 수 있도록 지원합니다.
3. 멀티 클라우드 및 하이브리드 클라우드 지원**:** vSphere Kubernetes Service는 프라이빗 클라우드, 퍼블릭 클라우드(예: AWS, Azure, Google Cloud) 및 하이브리드 클라우드 환경에 배포를 지원하여 조직이 애플리케이션을 유연하게 관리할 수 있도록 합니다.
4. 내장 보안**:** VKS에는 역할 기반 액세스 제어(RBAC), 정책 시행, 네트워크 보안을 위한 VMware NSX와의 통합 등 보안 기능이 포함되어 있습니다.
5. 개발자 친화적인 툴**:** vSphere Kubernetes Service는 Tanzu Application Platform 등 개발자 툴과 통합되어 컨테이너화된 애플리케이션의 개발 및 배포를 간소화합니다.
6. 확장성 및 고가용성**:** VKS는 자동 확장 및 내결함성과 같은 기능을 제공하여 애플리케이션의 높은 가용성과 성능을 보장합니다.
7. 관찰 가능성 및 모니터링**:** vSphere Kubernetes Service는 VMware Tanzu Observability(이전 Wavefront) 및 기타 모니터링 툴과 통합되어 클러스터 및 애플리케이션 성능에 대한 실시간 인사이트를 제공합니다.
8. **Kubernetes** 생태계 지원**:** VKS는 업스트림 Kubernetes와 완벽하게 호환되므로 Kubernetes API를 지원하고 Helm, Istio, Prometheus 등 Kubernetes 네이티브 툴과 함께 작동합니다.

vSphere Kubernetes Service의 사용 사례

1. 애플리케이션 현대화**:** 조직은 기존 애플리케이션을 컨테이너화하고 VKS에서 관리하는 Kubernetes 클러스터에 배포함으로써 현대화할 수 있습니다.
2. **DevOps** 활성화**:** VKS는 자동화, CI/CD 통합 및 개발자 친화적인 툴을 제공하여 DevOps 워크플로우를 지원합니다.
3. 하이브리드 클라우드 배포**:** 기업은 VKS를 사용하여 온프레미스 및 클라우드 환경 전반에 걸쳐 Kubernetes 클러스터를 배포하고 일관된 운영을 구현할 수 있습니다.
4. 마이크로서비스 아키텍처**:** VKS는 마이크로서비스 기반 애플리케이션 개발을 지원하여 팀이 분산 시스템을 구축하고 확장할 수 있도록 합니다.

스토리지 및 **vSphere Kubernetes** 서비스 시작하기

vSphere Kubernetes Service 클러스터를 설치합니다**.**

VCF 9.1 환경에 vSphere Kubernetes Service(VKS) 클러스터를 설치하고 워크로드에 적합한 스토리지 유틸리티를 사용하여 워커 노드를 구성하십시오.

이 작업 정보

NFS 유틸리티는 클러스터를 생성할 때 워커 노드에 자동으로 설치됩니다. iSCSI 또는 NVMe 유틸리티가 설치된 워커 노드를 생성하려면 사용자 지정 OVA 이미지를 생성하고 해당 이미지를 워커 노드에 사용해야 합니다. 사용자 지정 이미지는 Docker를 사용하여 생성할 수 있으며, `vcf` 명령줄 툴을 사용하여 이 이미지에서 OVA 파일을 생성할 수 있습니다. 이 OVA 파일은 vSphere의 콘텐츠 라이브러리에 업로드할 수 있습니다. VKS 클러스터를 생성할 때 콘텐츠 라이브러리에서 이 이미지를 노드 풀에 선택할 수 있습니다.

단계

1. VKS 클러스터를 생성합니다.

   기본 워커 노드 이미지 또는 사용자가 직접 생성한 사용자 지정 이미지를 사용하여 VKS 클러스터를 생성할 수 있습니다. 기본 워커 노드 이미지에는 NFS 유틸리티가 사전 설치되어 있으며, 사용자 지정 이미지에는 iSCSI 및 NVMe 유틸리티를 포함할 수 있습니다. 다음 옵션을 사용할 수 있습니다:

   - **NAS** 전용**(**기본값**):** NFS 유틸리티가 사전 설치된 기본 워커 노드 이미지를 사용하여 VKS 클러스터를 생성합니다.
   - **SAN** 활성화**(**사용자 지정 이미지**):** iSCSI 및 NVMe 유틸리티가 포함된 사용자 지정 Docker 이미지를 생성하고, `vcf` 명령줄 툴을 사용하여 OVA 파일을 생성하고, OVA를 vSphere 콘텐츠 라이브러리에 업로드한 다음, 노드 풀에 대해 해당 사용자 지정 이미지를 선택하여 VKS 클러스터를 생성합니다.

   ▼ 기본 워커 노드 이미지를 사용하여 **NAS** 전용 **vSphere Kubernetes Service** 클러스터를 생성하는 지침 표시

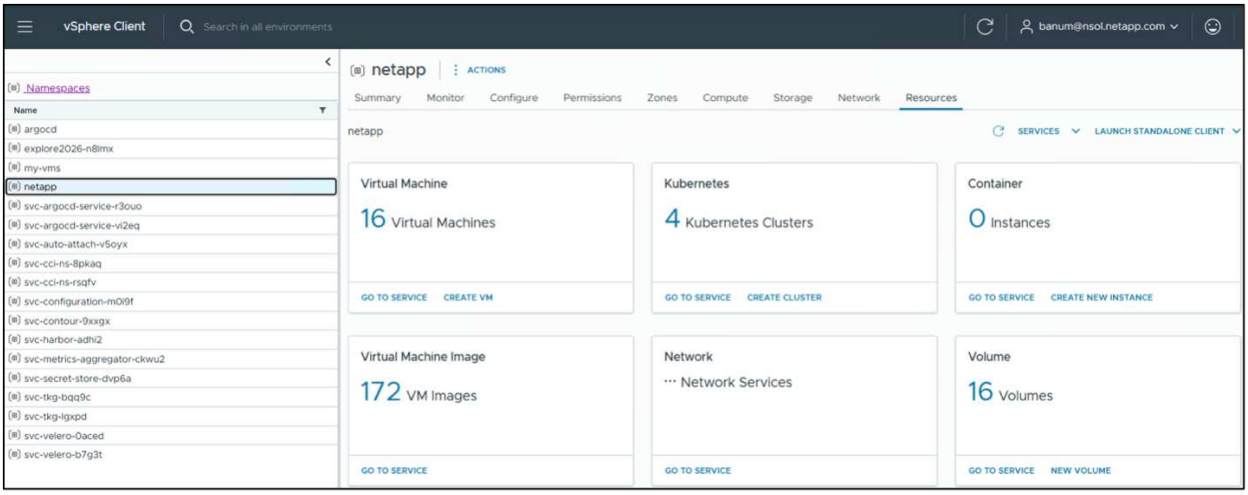
   기본 워커 노드 이미지를 사용하여 VKS 클러스터를 생성합니다. 안내에 따라 클러스터 이름, 노드 풀 구성 및 기타 설정을 지정합니다.

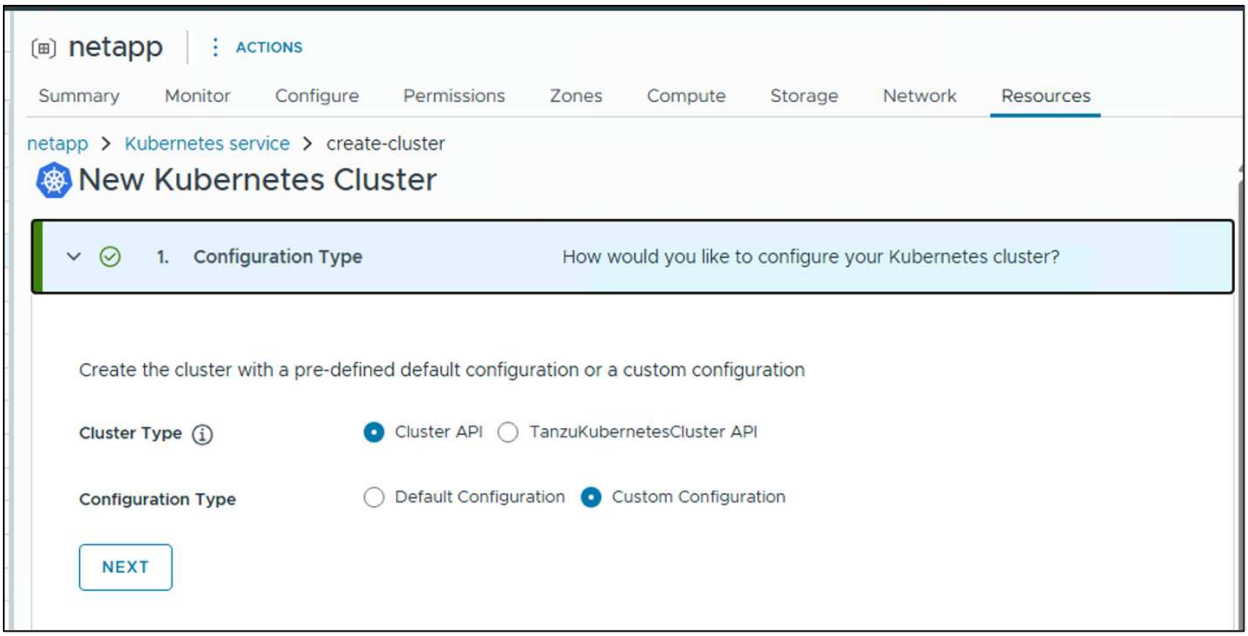
   vSphere Client에서 vSphere Kubernetes 클러스터를 생성할 네임스페이스로 이동합니다. 해당 네임스페이스의 리소스 탭에서 Kubernetes 섹션의 *클러스터 생성*을 클릭하거나, *서비스로 이동*을 클릭한 다음 *클러스터 생성*을 클릭합니다.

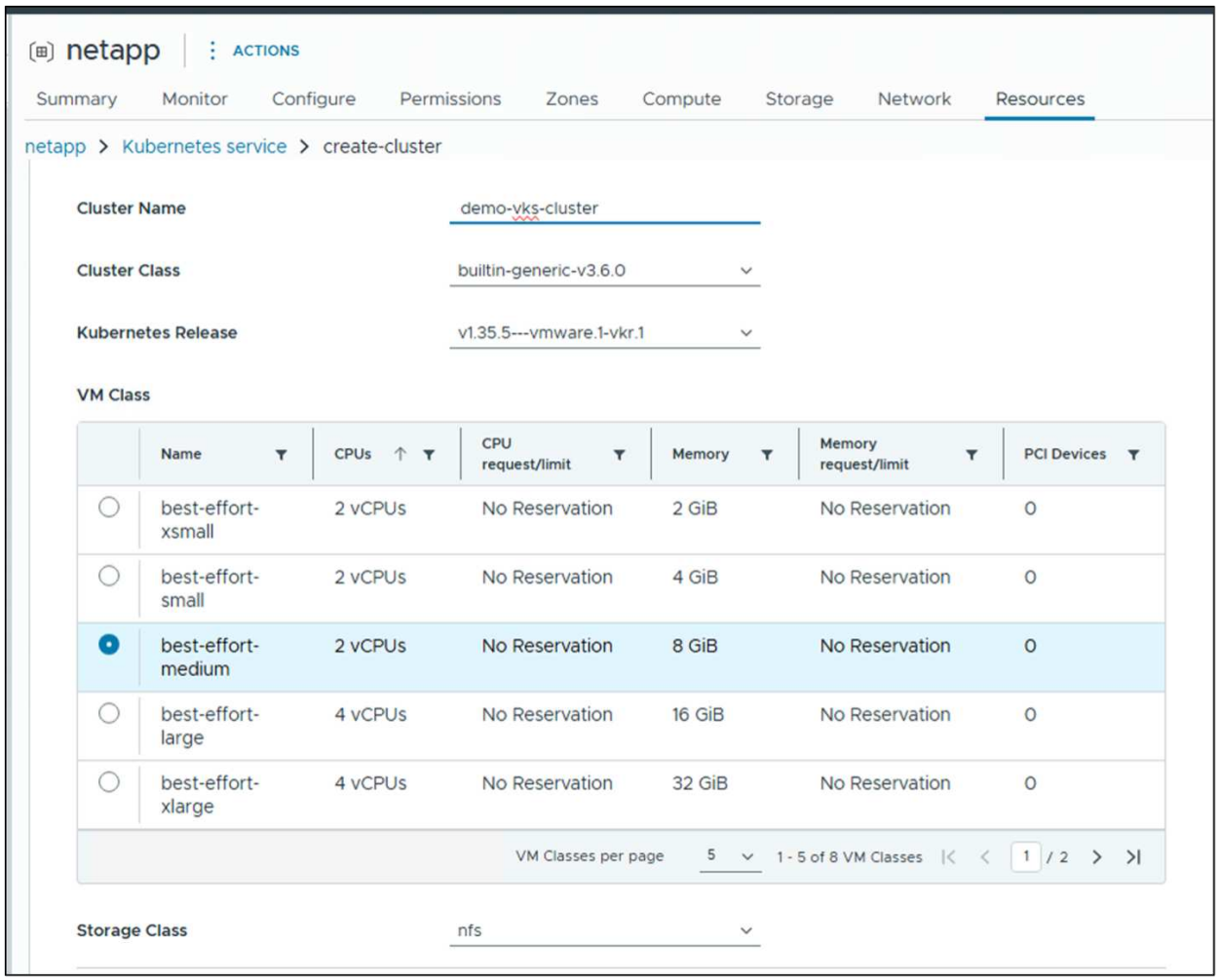
   클러스터 세부 정보를 양식에 입력합니다.

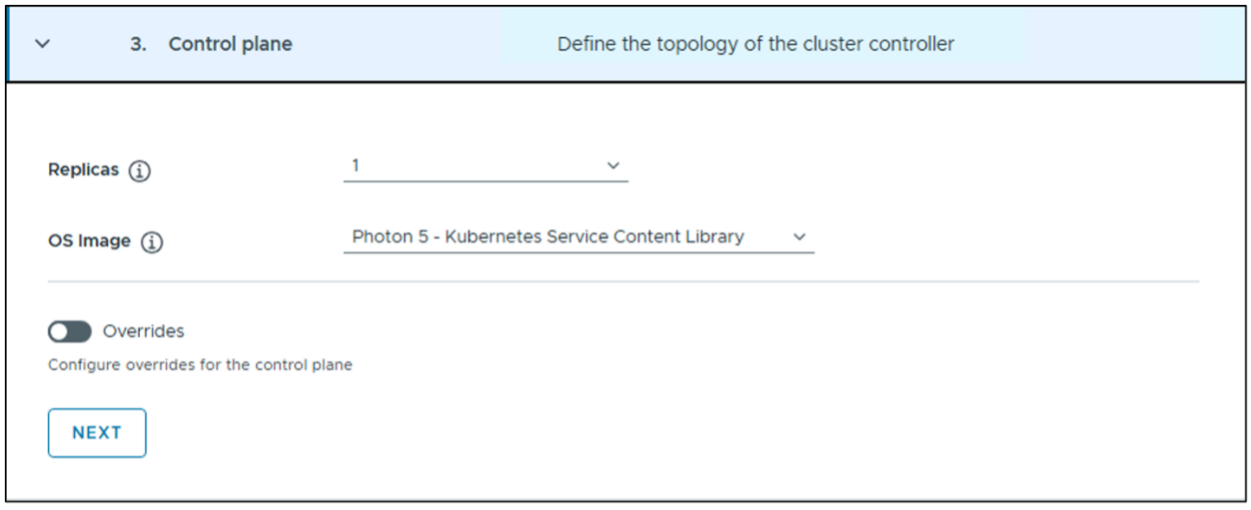
   - 섹션 1, *구성 유형*에서 *사용자 지정*을 선택합니다.
   - 2번 항목인 *일반 설정*에서 클러스터 이름을 지정하고 나머지 설정은 모두 기본값으로 유지하십시오.
   - 3번 항목의 모든 기본값을 그대로 사용합니다.
   - 4번 섹션 *노드 풀*에서 노드 풀 옆에 있는 점 세 개(…)를 클릭하고 *편집*을 클릭합니다. 복제본 수를 3으로 늘리고 OS 이미지로 최신 Ubuntu 버전을 선택합니다. *다음*을 클릭한 다음 *검토 및 확인*을 클릭합니다.

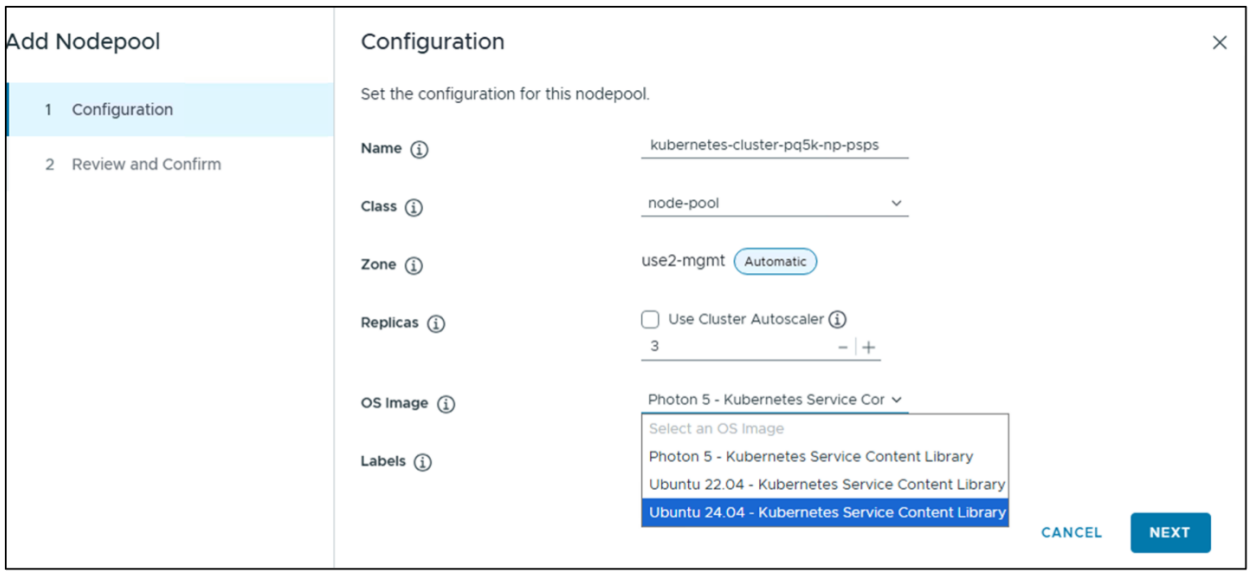
   클러스터 세부 정보를 검토한 후 *완료*를 클릭하십시오. vSphere Kubernetes 클러스터가 몇 분 안에 생성됩니다.

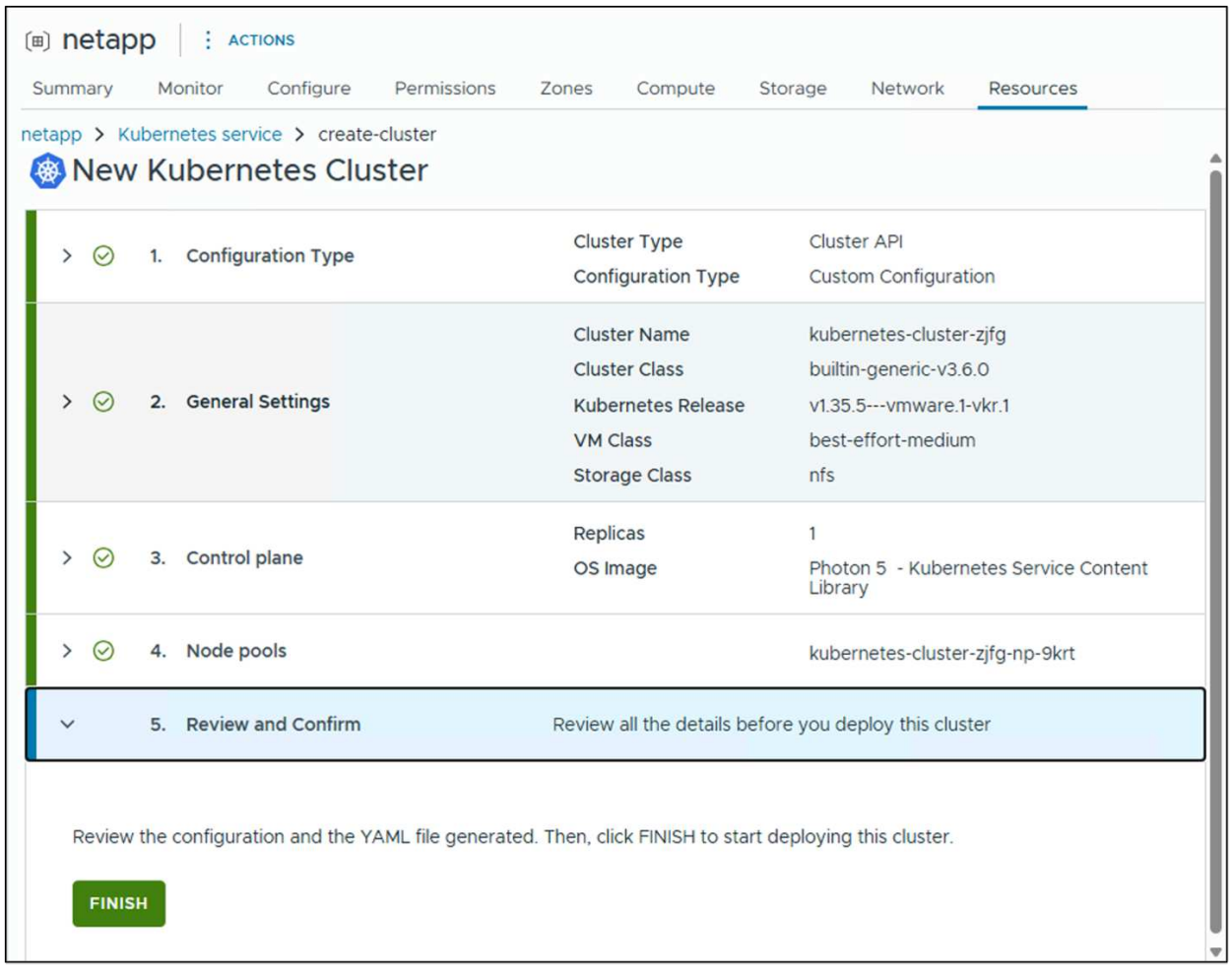

vSphere Client
Search in all environments
banum@nsol.netapp.com
Namespaces
Name
argocd
explore2026-n8lmx
my-vms
netapp
svc-argocd-service-r3ouo
svc-argocd-service-vi2eq
svc-auto-attach-v5oyx
svc-cci-ns-8pkaq
svc-cci-ns-rsqfv
svc-configuration-m0i9f
svc-contour-9xxgx
svc-harbor-adhi2
svc-metrics-aggregator-ckwu2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

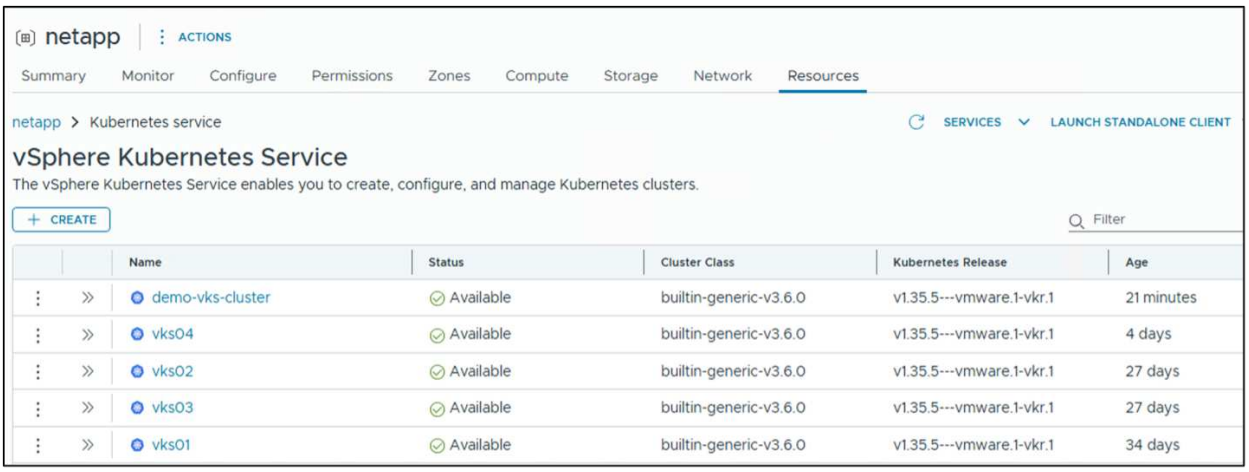

netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp > Kubernetes service > create-cluster
New Kubernetes Cluster
1. Configuration Type
How would you like to configure your Kubernetes cluster?
Create the cluster with a pre-defined default configuration or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


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


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


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


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



▼ **SAN** 지원 워커 노드용 사용자 지정 **OVA** 이미지 생성 지침 표시

필요한 유틸리티가 설치된 Docker 이미지를 생성합니다. 다음 예는 iSCSI 및 멀티패싱 유틸리티가 설치된 Ubuntu 24.04 기반 Docker 이미지를 생성하는 방법을 보여줍니다.

```
#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"]
```

다음을 사용하여 Docker 이미지를 빌드합니다.

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

Docker 레지스트리를 사용할 수 없는 경우, 다음 명령을 실행하여 로컬 레지스트리를 시작하고 이미지를 해당 레지스트리에 푸시하십시오.

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

이미지에 태그를 지정하고 푸시합니다.

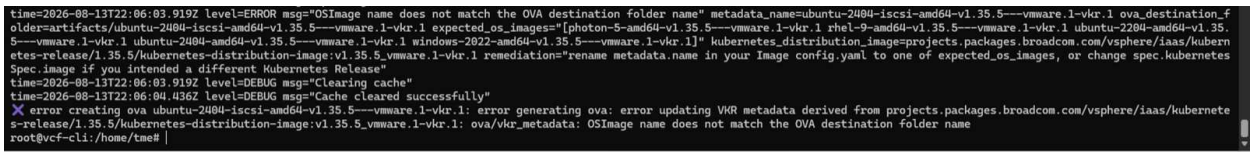
```
docker tag ubuntu-2404:san localhost:5000/ubuntu-2404:san
docker push localhost:5000/ubuntu-2404:san
```

이미지 구성 파일을 생성하고(`ubuntu2404vkr135-san.yml`로 저장) 이미지를 참조하십시오.

```
#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
```

메타데이터 이름은 사전 승인된 목록에 있는 이름이어야 합니다. 그렇지 않으면 OVA 파일을 생성하려고 할 때 아래와 같은 오류가 발생합니다. 사전 승인된 목록은 VCF CLI 도움말의 `kr bake` 명령에서 확인할 수 있습니다.



VCF CLI를 사용하여 OVA 파일을 생성합니다.

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

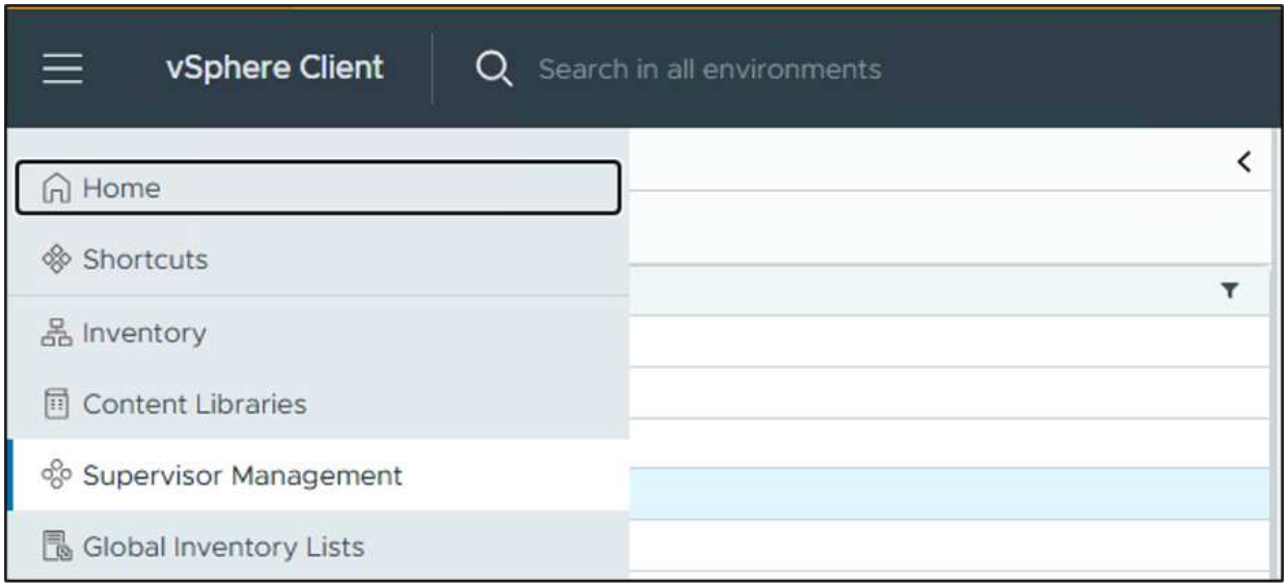
SCP를 사용하여 OVA 파일을 vSphere 콘텐츠 라이브러리에 업로드할 수 있는 컴퓨터로 전송합니다.

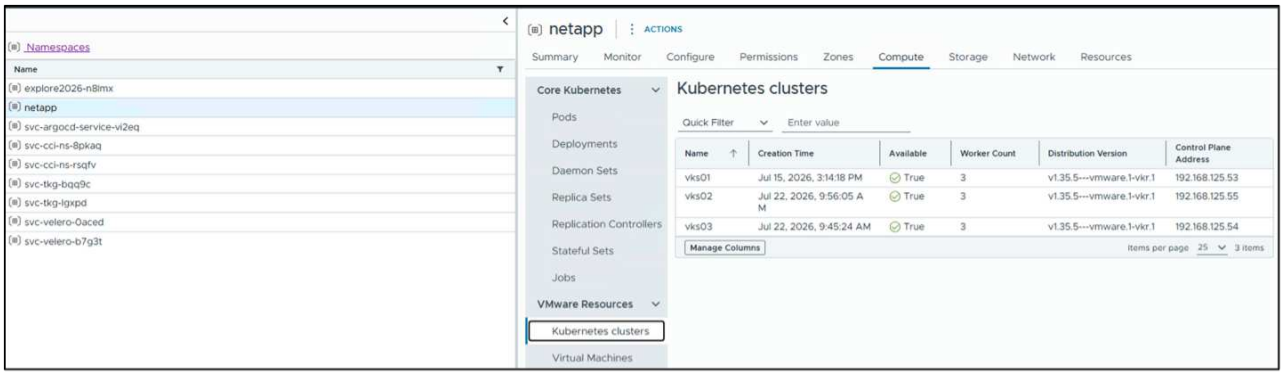
2. 클러스터 설치가 완료되면 vCenter에서 클러스터를 찾아 kubeconfig 파일을 다운로드하십시오.

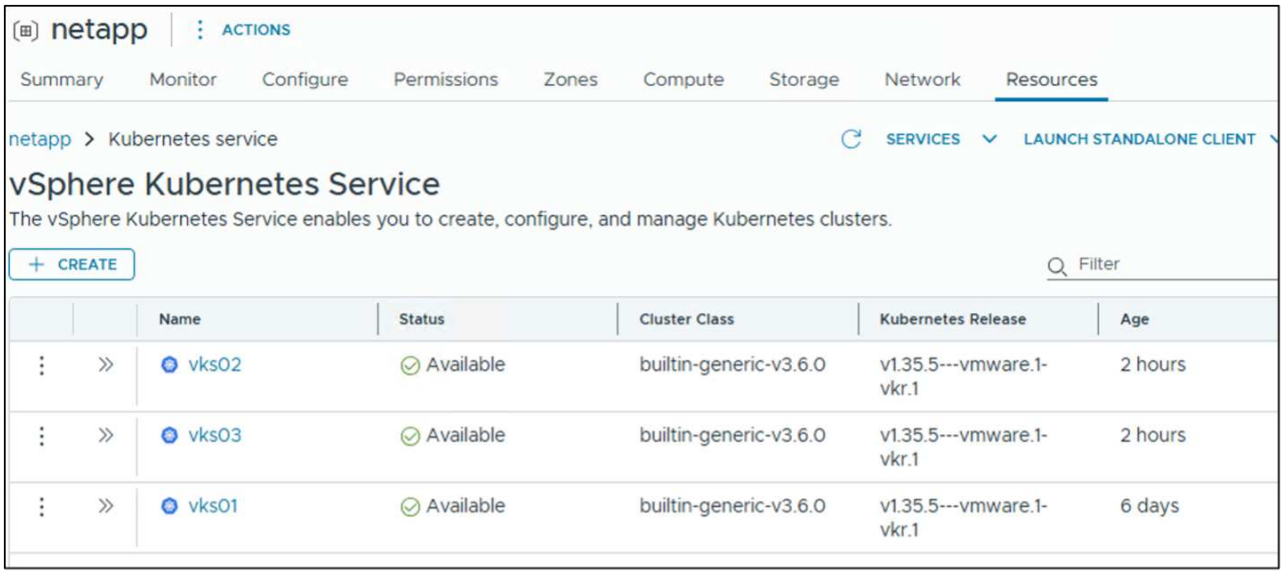
   a. 메인 메뉴에서 *Supervisor Management*를 클릭하고 클러스터가 생성된 네임스페이스를 선택하십시오.

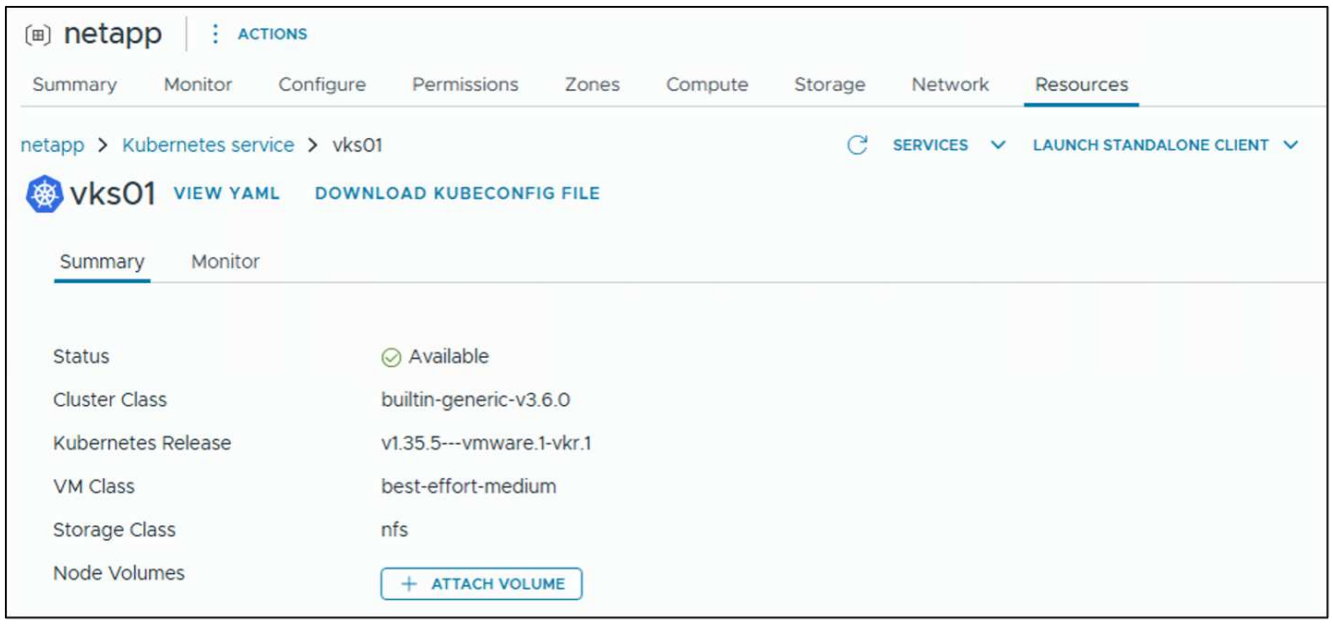
   b. 컴퓨팅 탭을 선택한 다음 *Kubernetes 클러스터*를 선택합니다. 네임스페이스에 있는 모든 클러스터가 표시됩니다.

   c. 리소스 탭으로 이동하여 필요한 Kubernetes 클러스터를 선택하고 해당 kubeconfig 파일을 다운로드하십시오.


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


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


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



3. 배스천 호스트에서 kubeconfig 파일을 사용하여 클러스터에 연결하고 CLI를 통해 클러스터를 관리합니다.

```
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
```

비디오 데모

다음 영상은 vSphere Kubernetes 클러스터를 생성하는 방법을 보여줍니다.

Supervisor 클러스터에 vSphere Kubernetes 클러스터를 생성합니다.

vSphere Kubernetes Service용 NetApp 스토리지에 대해 알아보십시오

NetApp을 vSphere Kubernetes Service(VKS)의 스토리지로 사용하는 것은 NetApp의 강력한 스토리지 솔루션을 활용하여 Kubernetes 워크로드의 성능, 확장성 및 안정성을 향상시키는 강력한 조합입니다. Kubernetes용 오픈 소스 동적 스토리지 프로비저너인 NetApp Trident는 vSphere Kubernetes Service에서 워크로드와 스토리지를 통합하는 데 사용됩니다.

VKS에서 **NetApp** 스토리지를 사용할 때의 주요 이점

1. 동적 스토리지 프로비저닝**:** NetApp Trident는 스토리지 볼륨의 동적 프로비저닝을 지원하여 Kubernetes Pod가 요구 사항에 따라 필요시 즉각적으로 구성 가능한 스토리지를 요청할 수 있도록 합니다.
2. 영구 스토리지**:** NetApp 솔루션은 영구 스토리지 기능을 제공하여 Pod 재시작 및 장애 발생 시에도 데이터의 내구성과 가용성을 보장합니다.
3. 고성능**:** NetApp의 스토리지 솔루션(예: ONTAP)은 낮은 지연 시간으로 고성능 스토리지를 제공하므로 Kubernetes에서 실행되는 I/O 집약적인 애플리케이션에 이상적입니다.
4. 확장성**:** NetApp 스토리지 시스템은 증가하는 Kubernetes 워크로드의 요구 사항을 충족하도록 확장할 수 있으며 대규모 배포를 지원합니다.

5. 데이터 관리**:** NetApp은 스냅샷, 클로닝 및 데이터 복제 등 고급 데이터 관리 기능을 제공하며, 이는 백업, 재해 복구 및 개발 워크플로에 유용할 수 있습니다.
6. 멀티 클라우드 및 하이브리드 클라우드 지원**:** NetApp의 스토리지 솔루션은 온프레미스, 퍼블릭 클라우드 및 하이브리드 클라우드 환경 전반에 걸쳐 배포할 수 있어 스토리지 관리에 유연성과 일관성을 제공합니다.

NetApp ONTAP 통합 개요

NetApp ONTAP는 데이터 중복제거, 압축 및 암호화와 같은 기능을 갖춘 엔터프라이즈급 스토리지를 제공합니다. NetApp Trident를 사용하여 NetApp 스토리지를 Kubernetes와 통합할 수 있습니다. NetApp Trident는 온프레미스 ONTAP, AWS의 FSx for NetApp ONTAP, Azure의 Azure NetApp Files, Google Cloud의 Google Cloud NetApp Volumes 등 다양한 NetApp 스토리지 플랫폼을 지원합니다.

Trident Protect를 사용하여 Kubernetes 워크로드의 데이터 보호 및 재해 복구를 처리합니다. Trident Protect를 사용하면 스냅샷 생성, 클론 생성, 다양한 스토리지 백엔드 간 데이터 복제를 통해 장애 발생 시 데이터의 안전과 복구를 보장할 수 있습니다.

Trident의 주요 기능

CSI 규격 준수 최신 Kubernetes 스토리지 기능 및 표준과의 호환성을 보장합니다. 선언적 구성 사용자가 YAML 파일에 스토리지 요구 사항을 정의할 수 있으며, 이는 Trident에 의해 자동으로 관리됩니다. 드라이버 Trident는 ONTAP에서 지원하는 NFS, CIFS/SMB, iSCSI, FC 및 NVMe/TCP 프로토콜용 드라이버를 지원합니다. 하이브리드 및 멀티 클라우드 지원 Trident는 온프레미스 ONTAP 스토리지와 하이퍼스케일러의 관리형 스토리지 서비스, 즉 AWS의 FSx for NetApp ONTAP, Azure의 Azure NetApp Files, Google Cloud의 Google Cloud NetApp Volumes용 드라이버를 지원합니다. 고급 데이터 관리 ONTAP의 스냅샷 및 클론 기능을 활용하여 효율적인 데이터 보호와 테스트 및 개발 환경의 신속한 프로비저닝을 지원합니다.

Trident에 대한 자세한 내용은 "Trident 문서"을 참조하십시오.

Trident Protect

Trident Protect는 Trident와 별도로 설치됩니다. 배포하기 전에 Kubernetes 플랫폼, 스토리지 백엔드, 스냅샷 구성 요소 및 오브젝트 스토리지가 "Trident Protect 요구 사항"을 충족하는지 확인하십시오.

Trident Protect의 주요 기능

자동 백업 Trident Protect는 Kubernetes 애플리케이션의 자동화된 정책 기반 백업을 지원하여 수동 개입 없이 정기적으로 백업이 이루어지도록 합니다.

Snapshot Management ONTAP의 스냅샷 기능을 활용하여 컨테이너 애플리케이션 및 VM의 특정 시점 복사본을 자주 생성함으로써 안정적인 복구 메커니즘을 제공합니다.

백업 저장소 암호화 Trident Protect는 Restic 및 Kopia 백업 저장소에 대한 암호 기반 암호화를 지원합니다. 영구 볼륨 및 전송 중 암호화는 스토리지 및 네트워크 계층에서 별도로 구성하십시오.

규정 준수 및 감사 조직이 규정 준수 요건을 충족하고 데이터 보호 관행에 대한 정기적인 감사를 수행하는 데 도움이 되는 도구와 기능을 제공합니다.

재해 복구 ONTAP의 복제 기능과 통합되어 강력한 재해 복구 솔루션을 제공함으로써 재앙적인 사태 발생 시에도 비즈니스 연속성을 보장합니다.

Trident Protect에 대한 자세한 내용은 "Trident Protect 문서"을 참조하십시오.

NetApp ONTAP 하드웨어 모델 및 프로토콜 지원

NetApp ONTAP 소프트웨어는 다양한 하드웨어 모델에서 실행되며, 각 모델은 서로 다른 성능 및 용량 요구 사항을 충족하도록 설계되었습니다. Trident는 강력한 기능을 확장하여 All Flash FAS(AFF), All SAN Array(ASA) 및 ASA R2 시스템을 포함한 다양한 NetApp ONTAP 시스템을 지원합니다. 이러한 지원을 통해 기업은 컨테이너화된 환경에서 고성능 스토리지 솔루션의 잠재력을 최대한 활용할 수 있습니다.

All Flash FAS(AFF) 시스템 AFF 시스템은 탁월한 성능과 낮은 지연 시간을 제공하여 고성능 애플리케이션에 이상적입니다.

All SAN Array (ASA) 시스템 ASA 시스템은 전용 SAN 환경을 위해 설계되었으며, 강력한 성능과 안정성을 제공합니다.

ASA R2 시스템 ASA R2 시스템은 SAN 환경을 위한 향상된 기능과 성능을 제공합니다.

AFX NetApp AFX는 기업용 AI 워크로드를 위해 특별히 설계된 분산형 올플래시 스토리지 아키텍처입니다. 고성능 NAS를 제공하여 컴퓨팅 및 용량을 독립적으로 확장할 수 있습니다. NetApp AFX 스토리지 시스템은 스토리지 계층 구현 방식에서 다른 ONTAP 기반 시스템(ASA, AFF, FAS)과 다릅니다. AFX는 고성능과 확장성을 제공하는 분산 파일 시스템을 사용하는 반면, 다른 ONTAP 기반 시스템은 기존 스토리지 아키텍처를 사용합니다.

1. AFF에 지원되는 프로토콜: NFS, CIFS/SMB, iSCSI, FC, NVMe/TCP
2. ASA에 지원되는 프로토콜: iSCSI, FC, NVMe/TCP
3. ASA R2에서 지원하는 프로토콜: iSCSI, FC, NVMe/TCP
4. AFX에서 지원하는 프로토콜: Trident 25.10부터는 NFS 프로토콜만 지원되며, SMB 프로토콜은 지원되지 않습니다.

ONTAP 스토리지용 Trident 드라이버

Trident에는 각각 백엔드 스토리지에 볼륨을 프로비저닝할 수 있는 다음과 같은 CSI 드라이버가 포함되어 있습니다.

지원되는 하드웨어 모델의 NAS 프로토콜

1. ontap-nas: 하나의 PVC는 하나의 PV에 매핑되며, 이는 전용 FlexVol 볼륨입니다.
2. ontap-nas-economy: 프로비저닝된 각 PV는 qtree이며, FlexVol 볼륨당 구성 가능한 qtree 수를 가집니다 (기본값은 200이며 50~300 사이에서 구성 가능).

   하나의 PVC는 하나의 PV에 매핑되며, 이는 공유 FlexVol 볼륨의 qtree입니다.

> VM의 모든 디스크를 단일 볼륨에 qtree로 구성할 수 있습니다. 그러나 Snapshots, Clones 및 Replication 기능은 Trident에서 지원되지 않습니다. 이러한 기능이 스토리지 관리자에 의해 관리되는 경우, PV 단위로 세부적인 관리가 불가능합니다.

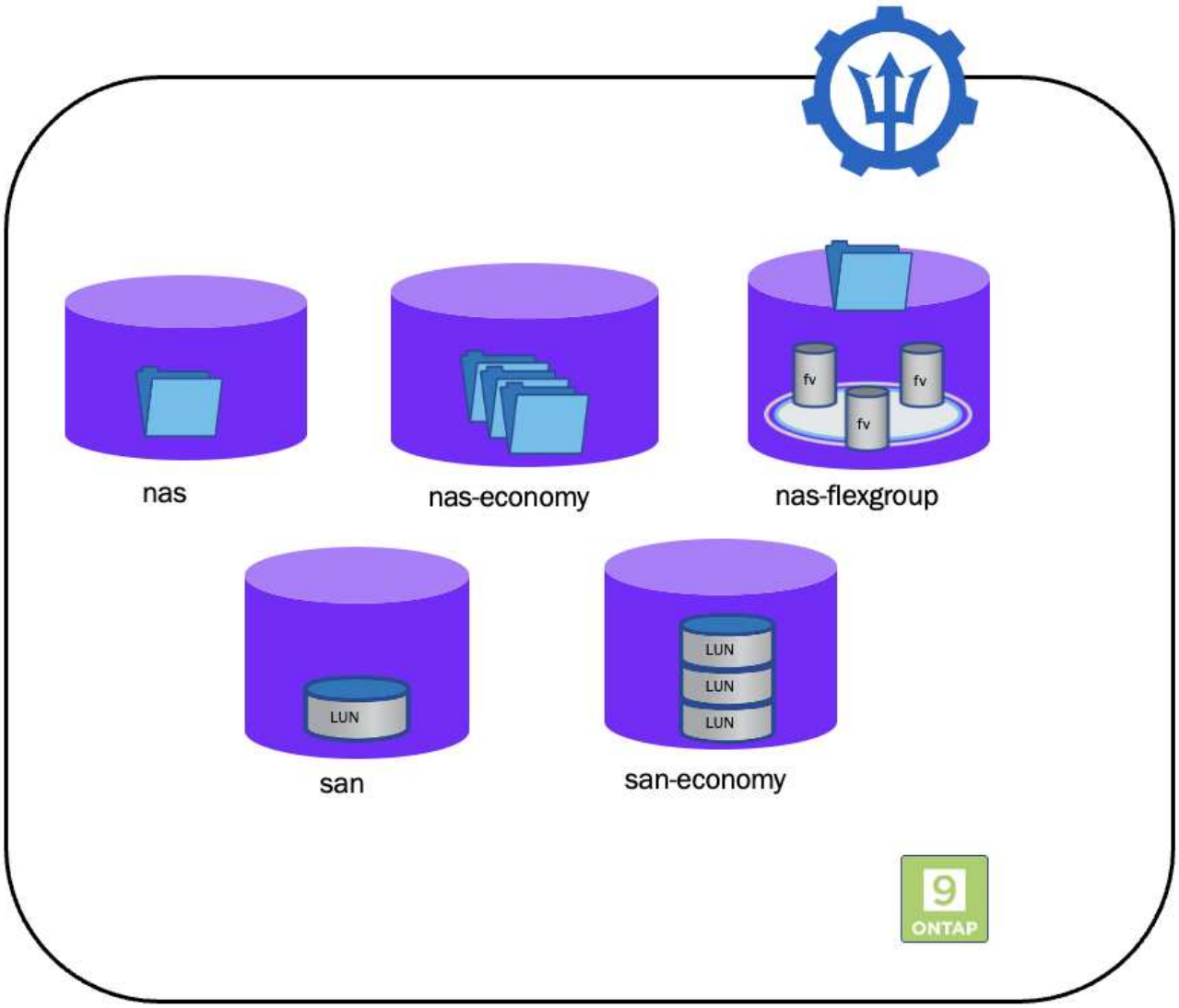
1. ontap-nas-flexgroup: 프로비저닝된 각 PV는 완전한 ONTAP FlexGroup이며, SVM에 할당된 모든 애그리게이트가 사용됩니다.

지원되는 하드웨어 모델의 SAN 프로토콜

1. ontap-san: 하나의 PVC는 하나의 PV에 매핑되며, 이는 전용 FlexVol 볼륨의 LUN입니다.
2. ontap-san-economy: 하나의 PVC는 하나의 PV에 매핑되며, 이는 공유 FlexVol 볼륨의 LUN입니다. 공유 FlexVol 볼륨에는 여러 개의 LUN을 저장할 수 있습니다(기본값은 100개이며, FlexVol 볼륨당 50~200개의 LUN/PV로 구성

가능).

모든 VM 디스크를 단일 볼륨에 LUN으로 배치할 수 있습니다. 하지만 이 드라이버는 스냅샷과 클론은 지원하지만 복제는 지원하지 않습니다. 복제가 스토리지 관리자에 의해 관리되는 경우 PV 단위로 세분화되지 않습니다.



Trident 드라이버를 통해 제공되는 기능과 스토리지 관리자가 적용해야 하는 기능의 전체 목록은 Trident 설명서의 "ONTAP 백엔드 드라이버" 섹션을 참조하십시오.

VKS 클러스터에 NetApp Trident 설치

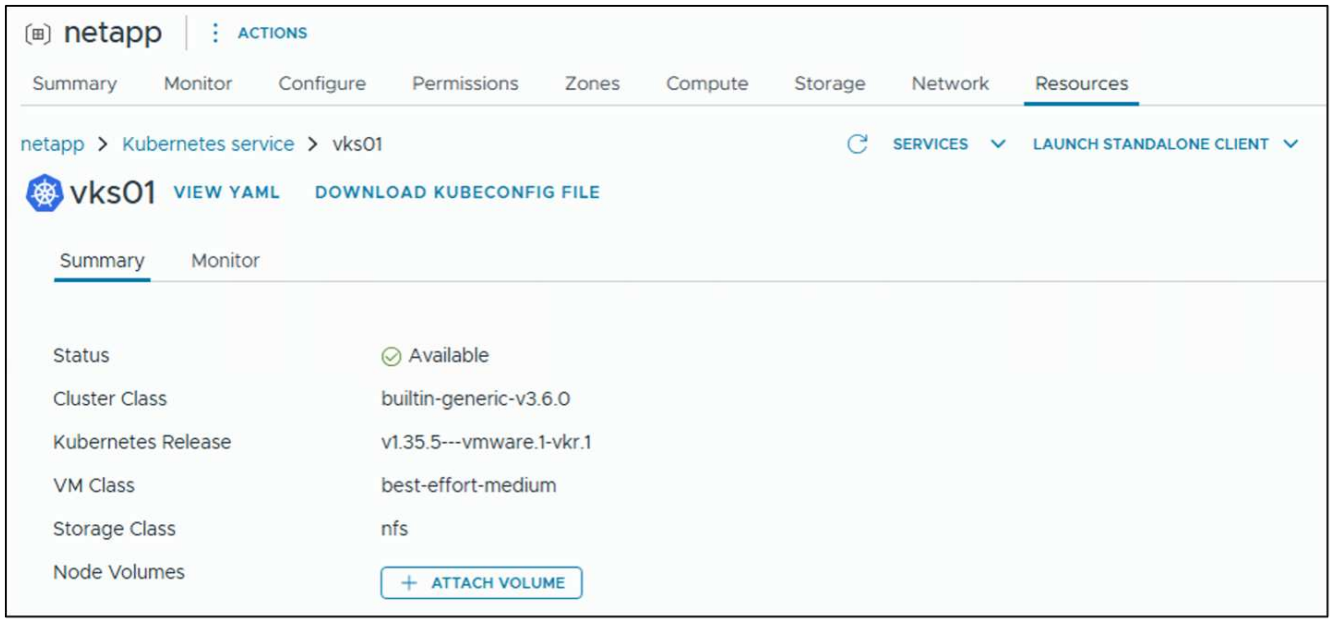
NetApp ONTAP 스토리지를 Kubernetes 워크로드와 통합하려면 vSphere Kubernetes Service 클러스터에 NetApp Trident를 동적 스토리지 프로비저너로 설치하십시오.

시작하기 전에

- ONTAP 클러스터가 구성되어 VMware Kubernetes 환경에 연결되었는지 확인하십시오.
- 워크로드가 스토리지에 iSCSI 또는 NVMe 툴을 사용할 경우 VKS 클러스터에 해당 툴이 설치되어 있는지

확인하십시오.

- kubeconfig 파일을 사용하여 VKS 클러스터에 연결할 수 있는 컴퓨터에 `kubectl`이(가) 설치되어 있는지 확인하십시오.



단계

1. Trident 설명서의 지침에 따라 Helm 차트를 사용하여 Trident Operator를 설치하십시오.

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

   ▼ 예제 표시

   ```
   helm install trident netapp-trident/trident-operator --version
   100.2606.0 --create-namespace --namespace trident
   --kubeconfig=/path/to/your-kubeconfig-file
   ```

   디버그 로깅을 활성화하려면 다음을 추가하십시오 `--set tridentDebug=true`:

   ```
   helm install trident netapp-trident/trident-operator --version
   100.2606.0 --create-namespace --namespace trident --set
   tridentDebug=true --kubeconfig=/path/to/your-kubeconfig-file
   ```

   Trident Operator를 사용하여 Trident를 수동으로 설치할 수도 있습니다. Trident 설명서에서 절차를 검토하십시오: "Trident 오퍼레이터를 수동으로 배포합니다"

2. 클러스터에서 모든 Trident Pod가 실행 중인지 확인하십시오.

   ▼ 예제 표시

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

이 시점에서 Trident Pod가 생성되지 않을 수 있습니다. ReplicaSet 오브젝트를 설명하면 아래와 같이 PodSecurity 위반에 대한 오류가 표시될 수 있습니다. 이 오류는 `trident` 설치 중에 생성된 네임스페이스가 Kubernetes의 제한적인 Pod 보안 표준(PSS)을 적용하기 때문에 발생합니다. Trident Operator는 호스트 스토리지를 관리하기 위해 높은 시스템 권한이 필요하며, 제한된 프로필의 엄격한 제약 조건 하에서는 실행될 수 없습니다. 이 문제를 해결하려면 `trident` 네임스페이스에 권한 있는 프로필을 부여하여 ReplicaSet 컨트롤러가 Pod를 성공적으로 생성할 수 있도록 하십시오.

```
Warning  FailedCreate  25s (x5 over 64s)  replicaset-controller
(combined from similar events): Error creating: pods "trident-
operator-5cbf7f857-z7ztd" is forbidden: violates PodSecurity
"restricted:latest": allowPrivilegeEscalation != false (container
"trident-operator" must set
securityContext.allowPrivilegeEscalation=false), unrestricted
capabilities (container "trident-operator" must set
securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod
or container "trident-operator" must set
securityContext.runAsNonRoot=true), seccompProfile (pod or container
"trident-operator" must set securityContext.seccompProfile.type to
"RuntimeDefault" or "Localhost")
```

기본 제공 Kubernetes Pod 보안 승인 레이블을 `trident` 네임스페이스에 적용하여 제한된 프로필 제약 조건에서 제외합니다.

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

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

```
vksbastion@vks:~/vks02$  kubectl get pods -n trident --kubeconfig=vks02-kubeconfig.yaml
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-84576f9bcf-vz622   6/6     Running   0          4d
trident-node-linux-24d47              2/2     Running   0          4d
trident-node-linux-52ccn              2/2     Running   0          4d
trident-node-linux-5bnv9              2/2     Running   0          4d
trident-node-linux-8hgnb              2/2     Running   0          4d
trident-operator-5cbf7f857-kskcx      1/1     Running   0          4d
vksbastion@vks:~/vks02$
```

사용 환경에 맞는 스토리지 백엔드 및 **StorageClass** 구성 파일을 준비합니다

1. 필요한 연결 세부 정보와 스토리지 매개변수를 정의하여 ONTAP용 Trident 백엔드를 구성합니다.
2. Kubernetes에서 NetApp 스토리지 백엔드에 매핑되는 스토리지 클래스를 정의합니다. 이러한 클래스는 성능 특성 및 복제 정책과 같은 매개변수를 지정합니다.

   워크로드 요구 사항에 따라 다음 프로토콜에 대한 Trident 백엔드 및 스토리지 클래스를 정의하십시오.

   - NAS(ontap-nas 드라이버)
   - NAS 이코노미(ontap-nas-economy 드라이버)
   - iSCSI, FC 또는 NVMe 프로토콜용 SAN(ontap-san 드라이버)
   - iSCSI용 SAN Economy(ontap-san-economy 드라이버)

NAS

TridentBackendConfig 및 StorageClass를 생성하여 ONTAP NAS에 대한 NFS 기반 영구 스토리지 프로비저닝을 활성화합니다. 백엔드 구성에는 Kubernetes Secret에 저장된 자격 증명이 포함되며 ONTAP SVM 및 관리 LIF를 참조합니다.

사용자 이름과 비밀번호 인증을 사용하는 백엔드 비밀 키 및 백엔드 구성 파일 예시 (다음으로 저장 `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
```

클라이언트 인증서 인증을 사용하는 백엔드 비밀 키 및 백엔드 구성 파일 예시 (다른 이름으로 저장 `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 정의(다른 이름으로 저장 `sc-nas.yaml`):

`mountOptions`은 NFS 볼륨에 대한 추가 마운트 옵션을 지정하는 선택적 매개변수입니다. `nconnect` 옵션을 사용하면 단일 NFS 마운트에 대해 여러 개의 병렬 TCP 연결을 허용하여 처리량이 높은 워크로드의 성능을 향상시킬 수 있습니다. 기본값은 1이지만 ONTAP 시스템에서 허용하는 최대값까지 늘릴 수 있습니다.

ONTAP는 nconnect를 지원하는 클라이언트에서 단일 NFS 마운트에 대해 최대 16개의 연결을 지원합니다. Trident는 마운트 옵션을 Kubernetes 워커 노드에 직접 전달하므로 워커 노드 NFS 클라이언트와 ONTAP 모두에서 지원되는

값을 선택하십시오. 다음 예에서는 4개의 연결을 사용합니다.

```
# 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
```

TridentBackendConfig 및 StorageClass를 생성하여 ONTAP NAS Economy 드라이버에서 qtree를 사용한 NFS 기반 영구 스토리지 프로비저닝을 활성화합니다. 백엔드 구성에는 Kubernetes Secret에 저장된 자격 증명이 포함되며 ONTAP SVM 및 관리 LIF를 참조합니다.

사용자 이름과 비밀번호 인증을 사용하는 백엔드 비밀 키 및 백엔드 구성 파일 예시 (다음으로 저장 `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 정의(다른 이름으로 저장 `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
```

iSCSI를 사용하는 ONTAP SAN에 대한 TridentBackendConfig 및 StorageClass를 생성하여 iSCSI 기반 블록 스토리지 프로비저닝을 활성화합니다. 백엔드 구성은 ontap-san 드라이버를 사용하며 Kubernetes Secret에 저장된 자격 증명을 포함합니다. 이 `sanType` 매개변수를 생략하면 기본값으로 iSCSI 프로토콜이 사용됩니다.

사용자 이름 및 비밀번호 인증을 사용하는 백엔드 비밀 키 및 백엔드 구성 파일(다른 이름으로 저장 `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
```

클라이언트 인증서 인증을 사용하는 백엔드 비밀 키 및 백엔드 구성 파일(다른 이름으로 저장 `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 정의(다른 이름으로 저장 `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
```

NVMe를 사용하는 ONTAP SAN에 대한 TridentBackendConfig 및 StorageClass를 생성하여 NVMe 기반 블록 스토리지 프로비저닝을 활성화합니다. 백엔드 구성은 ontap-san 드라이버를 사용하며 Kubernetes Secret에 저장된 자격 증명을 포함합니다. 해당 `sanType` 매개변수는 `nvme`로 설정해야 합니다.

사용자 이름 및 비밀번호 인증을 사용하는 백엔드 비밀 키 및 백엔드 구성 파일(다른 이름으로 저장 `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
```

클라이언트 인증서 인증을 사용하는 백엔드 비밀 키 및 백엔드 구성 파일(다른 이름으로 저장 `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 정의(다른 이름으로 저장 `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
```

Trident 설명서를 참조하시면 YAML 파일의 추가 샘플과 SAN 드라이버 및 NAS 드라이버에서 지원하는 액세스 모드 및 볼륨 모드에 대한 정보를 확인할 수 있습니다.

- "NAS 드라이버를 사용하는 샘플"
- "SAN 드라이버를 사용하는 샘플"

VolumeSnapshotClass 구성 파일 생성

VolumeSnapshotClass 정의를 생성합니다. 이 구성을 통해 영구 볼륨에 대한 스냅샷 기반 작업을 활성화할 수 있습니다.

VolumeSnapshotClass 정의(다른 이름으로 저장 `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
```

클러스터에 구성 파일을 적용합니다

이전 단계에서 생성한 구성 파일을 `kubectl`을 사용하여 vSphere Kubernetes Service 클러스터에 적용합니다. 이렇게 하면 클러스터에 필요한 시크릿, 백엔드 구성, 스토리지 클래스 및 스냅샷 클래스가 생성됩니다.

단계

1. 구성한 각 프로토콜에 대해 TridentBackendConfig 및 StorageClass 파일을 적용하십시오.

```
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. 리소스가 성공적으로 생성되었는지 확인하십시오.

   TridentBackendConfig 객체를 확인하세요:

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

StorageClass 객체를 확인하세요:

```
kubectl get storageclass
```

VolumeSnapshotClass 확인:

```
kubectl get volumesnapshotclass
```

기본 **Trident** 스토리지 및 스냅샷 클래스 설정

Trident StorageClass 및 VolumeSnapshotClass를 vSphere Kubernetes Service 클러스터의 기본값으로 설정합니다.

단계

1. 기본 Trident StorageClass를 설정합니다.

   Trident 기반 StorageClass를 클러스터 기본값으로 설정하여 스토리지 클래스가 지정되지 않은 경우 PersistentVolumeClaims가 자동으로 사용하도록 합니다.

   StorageClass는 기본값으로 하나만 설정해야 합니다. 다른 StorageClass가 이미 기본값으로 설정되어 있는 경우, 해당 어노테이션을 `false`으로 설정하십시오.

   CLI에서:

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

2. 기본 Trident VolumeSnapshotClass를 설정합니다.

   Trident 기반 VolumeSnapshotClass를 클러스터 기본값으로 설정하면 영구 볼륨에 대한 스냅샷 기반 작업이 활성화됩니다. 이렇게 하면 스냅샷 클래스가 지정되지 않은 경우 VolumeSnapshots이 자동으로 Trident CSI 드라이버를 사용하게 됩니다.

   VolumeSnapshotClass는 기본값으로 하나만 설정해야 합니다. 다른 VolumeSnapshotClass가 이미 기본값으로 설정되어 있는 경우 해당 어노테이션을 `false`로 설정하십시오.

   CLI에서:

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

Kubernetes Persistent Volume Claims를 사용하여 워크로드에 필요한 스토리지를 요청합니다.

Trident는 백엔드 NetApp 스토리지에서 필요한 볼륨을 자동으로 프로비저닝합니다. VKS 클러스터에서 PVC를 사용하여 워크로드를 배포하는 예는 "영구 스토리지로 워크로드 배포"을 참조하십시오.

NetApp 영구 스토리지를 사용하여 컨테이너 애플리케이션 배포

vSphere Kubernetes Service 클러스터에서 NetApp ONTAP 영구 스토리지를 사용하여 컨테이너화된 애플리케이션을 배포합니다. 다음 예제는 NAS(ontap-nas) 또는 SAN iSCSI(ontap-san) 스토리지를 사용하여 PostgreSQL 데이터베이스를 배포하는 방법을 보여줍니다.

다음 YAML 구성은 매니페스트에 POSTGRES_USER/POSTGRES_PASSWORD를 직접 설정합니다. 프로덕션 환경에서는 데이터베이스 자격 증명과 같은 민감한 정보를 관리하기 위해 Kubernetes Secrets를 사용하는 것이 좋습니다.

NAS 스토리지를 사용하여 PostgreSQL 배포(ontap-nas 드라이버)

ontap-nas 드라이버를 사용하여 프로비저닝된 NFS 영구 볼륨을 기반으로 PostgreSQL 컨테이너 애플리케이션을 배포할 수 있습니다. 다음 예제는 PostgreSQL용 PersistentVolumeClaim(PVC) 및 Deployment를 생성하는 방법을 보여줍니다.

```
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
```

SAN 스토리지를 사용하여 PostgreSQL 배포(ontap-san iSCSI 드라이버)

다음 YAML을 사용하여 ontap-san 드라이버로 프로비저닝된 iSCSI 영구 볼륨을 기반으로 하는 PostgreSQL 컨테이너 애플리케이션을 배포합니다. 이 `Recreate`배포 전략은 새 Pod가 시작되기 전에 기존 Pod가 완전히 종료되도록 보장하며, 이는 ReadWriteOnce iSCSI 볼륨에 필수적이며 Pod 재시작 시 데이터 손상을 방지합니다. 이 구성은 Pod 삭제 및 재생성 시에도 데이터 지속성을 지원하며, Trident 볼륨 스냅샷 및 백업과 복원 작업과 호환됩니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-block-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: sc-iscsi # Must map to an ontap-san driver
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres-block-app
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
```

```
      # 1. POD LEVEL SECURITY CONTEXT
      securityContext:
        runAsNonRoot: true
        runAsUser: 999        # Official Postgres image default user ID
        runAsGroup: 999       # Official Postgres image default group ID
        fsGroup: 999
        seccompProfile:
          type: RuntimeDefault
      containers:
        - name: postgres
          image: postgres:15
          # 2. CONTAINER LEVEL SECURITY CONTEXT
          securityContext:
            allowPrivilegeEscalation: false
            capabilities:
              drop:
                - ALL
          env:
            - name: POSTGRES_DB
              value: "postgres"
            - name: POSTGRES_USER
              value: "<username>"
            - name: POSTGRES_PASSWORD
              value: "<password>"
            - name: PGDATA
              value: /var/lib/postgresql/data/pgdata
          ports:
            - containerPort: 5432
              name: postgres
          volumeMounts:
            - mountPath: /var/lib/postgresql/data
              name: postgres-block-storage
              subPath: postgres-data
          resources:
            requests:
              memory: "512Mi"
              cpu: "250m"
            limits:
              memory: "1Gi"
              cpu: "500m"
      volumes:
        - name: postgres-block-storage
          persistentVolumeClaim:
            claimName: postgres-block-pvc
---
apiVersion: v1
```

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

데이터베이스 배포 검증

애플리케이션을 배포한 후, 파드가 실행 중인 상태이고 데이터베이스가 정상적으로 작동하는지 확인하십시오. 다음 명령어를 사용하여 파드, PVC 및 서비스의 상태를 확인하십시오. 파드가 실행 중이고 PVC가 적절한 볼륨에 바인딩되었는지 확인하십시오.

```
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:~$
```

사용하는 프로토콜에 따라 백엔드 스토리지는 볼륨, qtree 또는 LUN입니다. ONTAP 시스템에 프로비저닝된 스토리지는 PersistentVolume(PV)을 설명하여 내부 볼륨 이름을 가져온 다음 ONTAP CLI 명령에서 해당 이름을 사용하여 스토리지 세부 정보를 확인함으로써 검증할 수 있습니다. 다음 명령을 사용하여 PV를 설명하고 내부 볼륨 이름을 가져옵니다.

```
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>
```

그림 **1.** 예시 보기

```
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>
```

출력에서 내부 볼륨 이름을 복사하고 다음 명령을 실행하여 ONTAP CLI에서 스토리지 세부 정보를 가져오십시오.

NAS 볼륨의 경우:

```
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
```

SAN LUN의 경우, ONTAP CLI에서 다음 명령을 실행하여 LUN 세부 정보를 가져오십시오.

```
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::>
```

NAS 또는 NAS Economy 스토리지 클래스에서 `nconnect` 마운트 옵션을 사용한 경우, 워커 노드에서 스토리지 서버로의 활성 TCP 연결 수를 확인할 수 있습니다.

먼저 Trident 백엔드 구성을 나열하여 백엔드 이름을 찾은 다음, 해당 이름을 설명하여 vserver 이름과 데이터 LIF를 가져옵니다.

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

그런 다음 ONTAP 클러스터에 로그인하고 위 출력의 데이터 LIF를 대체하여 다음 명령을 실행하십시오.

```
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.
```

그림 **2.** 예시 보기

Pod가 실행 상태가 되면 데이터베이스에 연결하여 데이터 작업이 제대로 수행되는지 확인합니다.

단계

1. PostgreSQL 서비스 포트를 로컬 시스템으로 포워딩합니다.

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

2. 다른 터미널에서 psql을 사용하여 데이터베이스에 연결합니다.

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

3. 데이터베이스와 테이블을 생성한 다음, 테이블에 데이터를 입력합니다.

```
CREATE DATABASE employee;
```

새 데이터베이스로 전환합니다(psql 메타 명령):

```
\c employee
```

테이블을 생성하고, 행을 삽입하고, 결과를 조회합니다.

```
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;
```

비디오 데모

다음 비디오는 vSphere Kubernetes 클러스터에 Trident를 설치하고 Trident를 사용하여 영구 스토리지로 PostgreSQL 데이터베이스를 배포하는 방법을 보여 줍니다.

Trident를 사용하여 ONTAP 영구 스토리지로 워크로드 배포