vSphere Kubernetes Service 与 NetApp 存储

NetApp container solutions

NetApp
August 25, 2026

目录

vSphere Kubernetes Service 与 NetApp 存储 . . . 1
- 了解如何将 vSphere Kubernetes 服务与 NetApp ONTAP 结合使用 . . . 1
  - vSphere Kubernetes Service 的主要功能 . . . 1
  - vSphere Kubernetes Service 的使用案例 . . . 1
- 存储和 vSphere Kubernetes Service 入门 . . . 1
  - 安装 vSphere Kubernetes Service 集群 . . . 2
  - 了解适用于 vSphere Kubernetes Service 的 NetApp 存储 . . . 10
  - 在 VKS 集群中安装 NetApp Trident . . . 13
  - 部署具有 NetApp 永久存储的容器应用程序 . . . 24

vSphere Kubernetes Service 与 NetApp 存储

了解如何将 vSphere Kubernetes 服务与 NetApp ONTAP 结合使用

vSphere Kubernetes 服务 (VKS) 是 VMware 提供的托管 Kubernetes 产品，使组织能够使用 Kubernetes 部署、管理和扩展容器化应用程序。与 NetApp ONTAP 存储相结合，VKS 为云原生工作负载提供企业级持久性、性能和数据管理。

vSphere Kubernetes Service 通常集成到 VMware 更广泛的生态系统中，例如 VMware Tanzu，它为 Kubernetes 管理、应用程序现代化和 DevOps 实践提供了一套全面的工具。

vSphere Kubernetes Service 的主要功能

1. 托管 **Kubernetes** 集群： vSphere Kubernetes Service 简化了 Kubernetes 集群的部署和管理。它可以自动执行集群创建、升级、扩展和监控等任务。
2. 与 **VMware** 基础设施集成： VKS 与 VMware vSphere、NSX-T 和其他 VMware 解决方案无缝集成，使组织能够利用其现有的 VMware 基础设施来处理 Kubernetes 工作负载。
3. 多云和混合云支持： vSphere Kubernetes Service 支持在私有云、公共云（例如 AWS、Azure、Google Cloud）和混合云环境中进行部署，为组织提供管理应用程序的灵活性。
4. 内置安全性： VKS 包括基于角色的访问控制 (RBAC)、策略实施以及与 VMware NSX 集成以实现网络安全等安全功能。
5. 开发人员友好型工具： vSphere Kubernetes Service 与 Tanzu Application Platform 等开发人员工具集成，以简化容器化应用程序的开发和部署。
6. 可扩展性和高可用性： VKS 提供自动扩展和容错等功能，以确保应用程序保持高可用性和高性能。
7. 可观测性和监控： vSphere Kubernetes Service 与 VMware Tanzu Observability（以前称为 Wavefront）和其他监控工具集成，可实时洞察集群和应用程序性能。
8. 支持 **Kubernetes** 生态系统： VKS 完全符合上游 Kubernetes，这意味着它支持 Kubernetes API，并与 Helm、Istio 和 Prometheus 等 Kubernetes 原生工具配合使用。

vSphere Kubernetes Service 的使用案例

1. 应用程序现代化： 组织可以通过容器化旧应用程序并将其部署在由 VKS 管理的 Kubernetes 集群上来实现旧应用程序的现代化。
2. **DevOps** 支持： VKS 通过提供自动化、CI/CD 集成和便于开发人员使用的工具来支持 DevOps 工作流程。
3. 混合云部署： 企业可以使用 VKS 跨本地和云环境部署 Kubernetes 集群，实现一致的操作。
4. 微服务架构： VKS 支持基于微服务的应用程序开发，允许团队构建和扩展分布式系统。

存储和 vSphere Kubernetes Service 入门

安装 **vSphere Kubernetes Service** 集群

在 VCF 9.1 环境中安装 vSphere Kubernetes Service (VKS) 集群，并为工作负载配置具有适当存储实用程序的工作节点。

关于此任务

NFS 实用程序会自动安装在您创建的集群的工作节点中。如果要创建安装了 iSCSI 或 NVMe 实用程序的工作节点，则必须创建自定义 OVA 映像，并将该映像用于工作节点。可以使用 Docker 创建自定义映像，然后可以使用 `vcf`命令行工具从此映像创建 OVA 文件。此 OVA 文件可以上传到 vSphere 中的内容库。创建 VKS 集群时，可以为节点池选择内容库中的此映像。

步骤

1. 创建 VKS 集群：

   您可以使用默认工作节点映像或自定义映像创建 VKS 集群。默认工作节点映像包含预安装的 NFS 实用程序，而自定义映像可以包含 iSCSI 和 NVMe 实用程序。以下选项可用：

   - 仅 **NAS**（默认）： 使用默认工作节点映像创建 VKS 集群，其中包括预安装的 NFS 实用程序。
   - 启用 **SAN**（自定义映像）： 创建包含 iSCSI 和 NVMe 实用程序的自定义 Docker 映像，使用 `vcf`命令行工具生成 OVA 文件，将 OVA 上传到 vSphere 内容库，然后创建 VKS 集群，为节点池选择该自定义映像。

   ▼ 显示使用默认工作节点映像创建仅限 **NAS** 的 **vSphere Kubernetes Service** 集群的说明

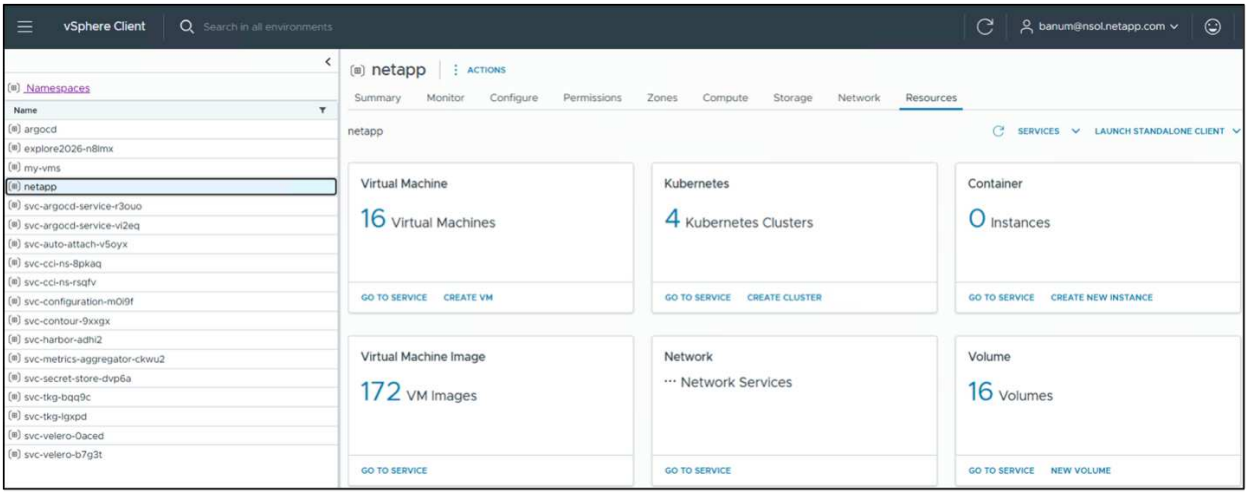
   使用默认工作节点映像创建 VKS 集群。按照提示指定集群名称、节点池配置和其他设置。

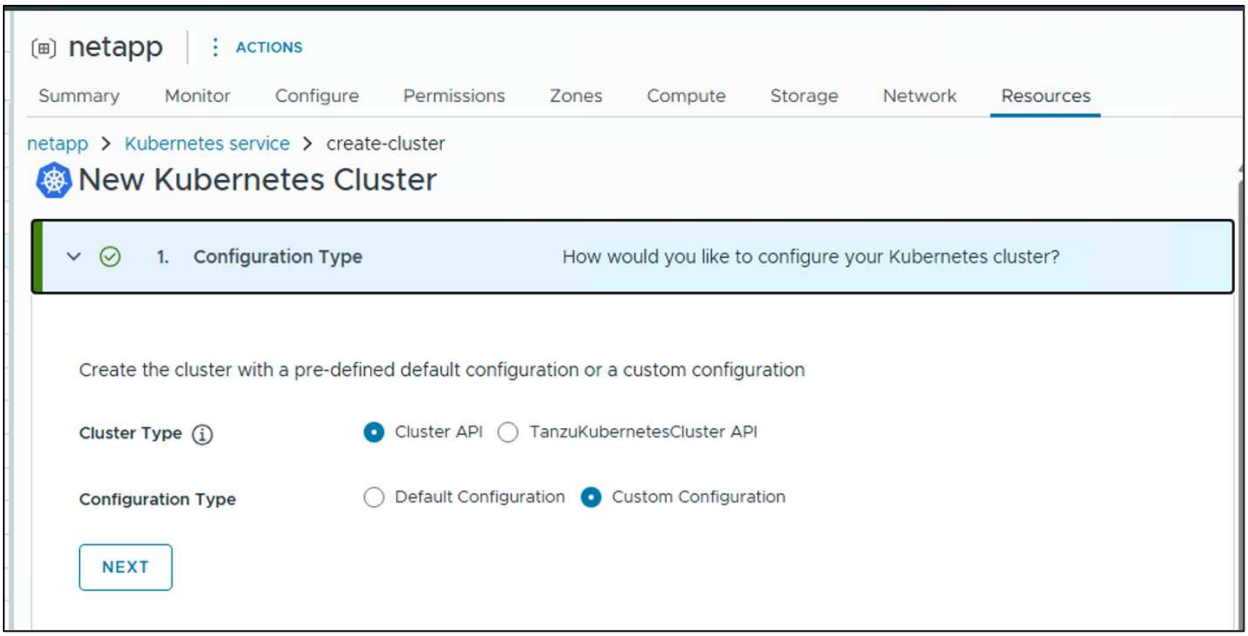
   从 vSphere 客户端转到要创建 vSphere Kubernetes 集群的命名空间。在该命名空间的*资源*选项卡上，单击 Kubernetes 部分中的*创建集群*，或单击*转到服务*，然后单击*创建集群*。

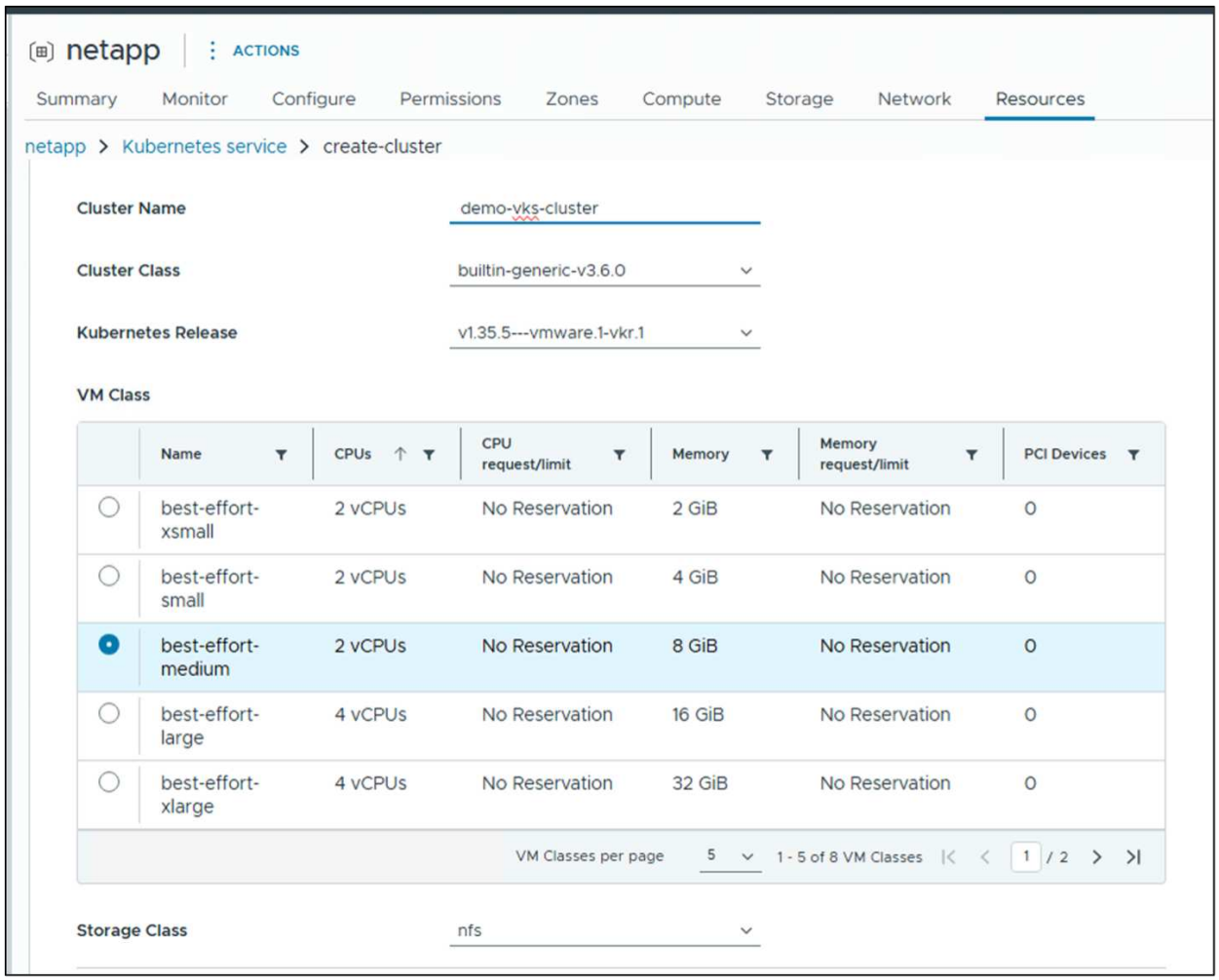
   使用集群详细信息填写表格：

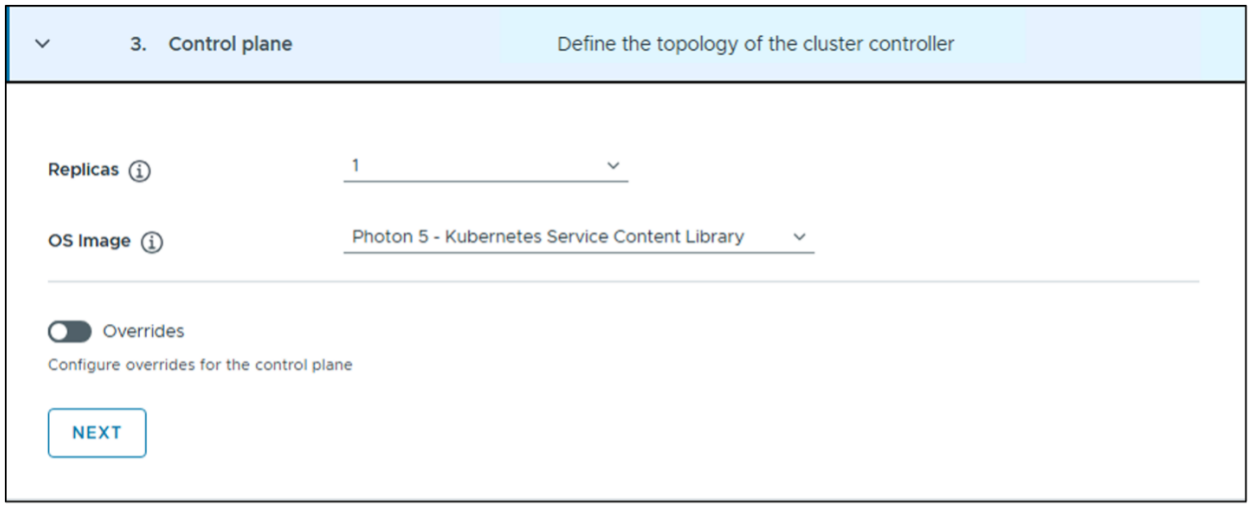
   - 在第 1 部分的*配置类型*中，选择*自定义*。
   - 在第 2 部分的 **General Settings** 中，提供集群的名称，并将所有其他设置保留为默认值。
   - 接受第 3 节的所有默认值。
   - 在第 4 部分，节点池*中，单击节点池旁边的省略号（**.....**），然后单击*编辑。将副本增加到 3 个，并为操作系统映像选择最新的 Ubuntu 版本。单击*下一步*，然后单击*查看并确认*。

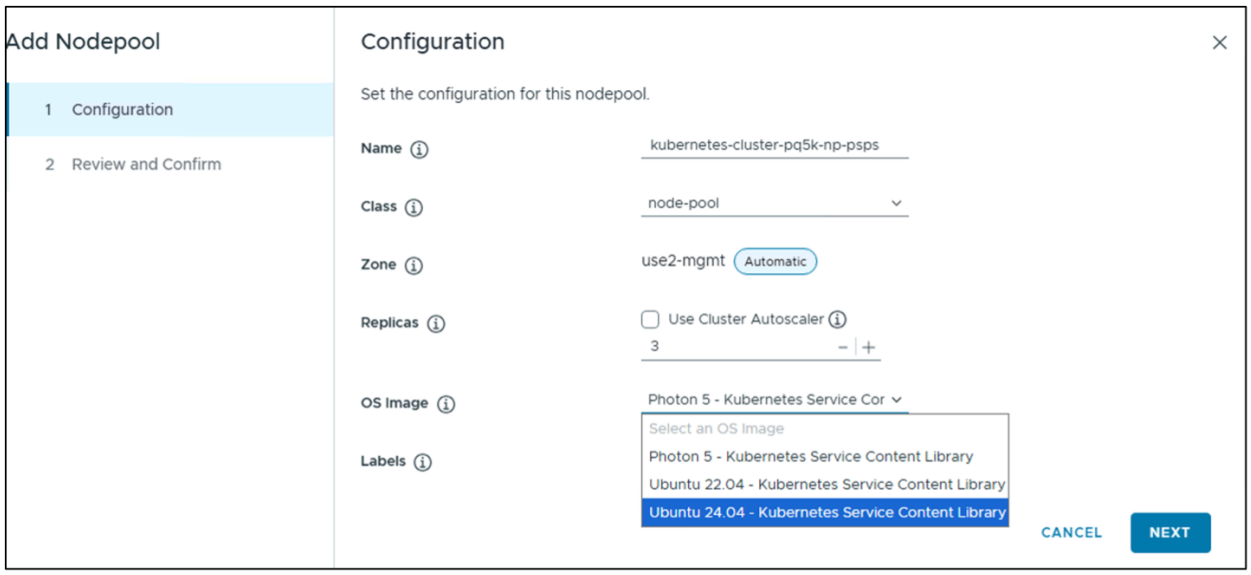
   查看集群详细信息后，单击 完成。vSphere Kubernetes 集群将在几分钟后创建。

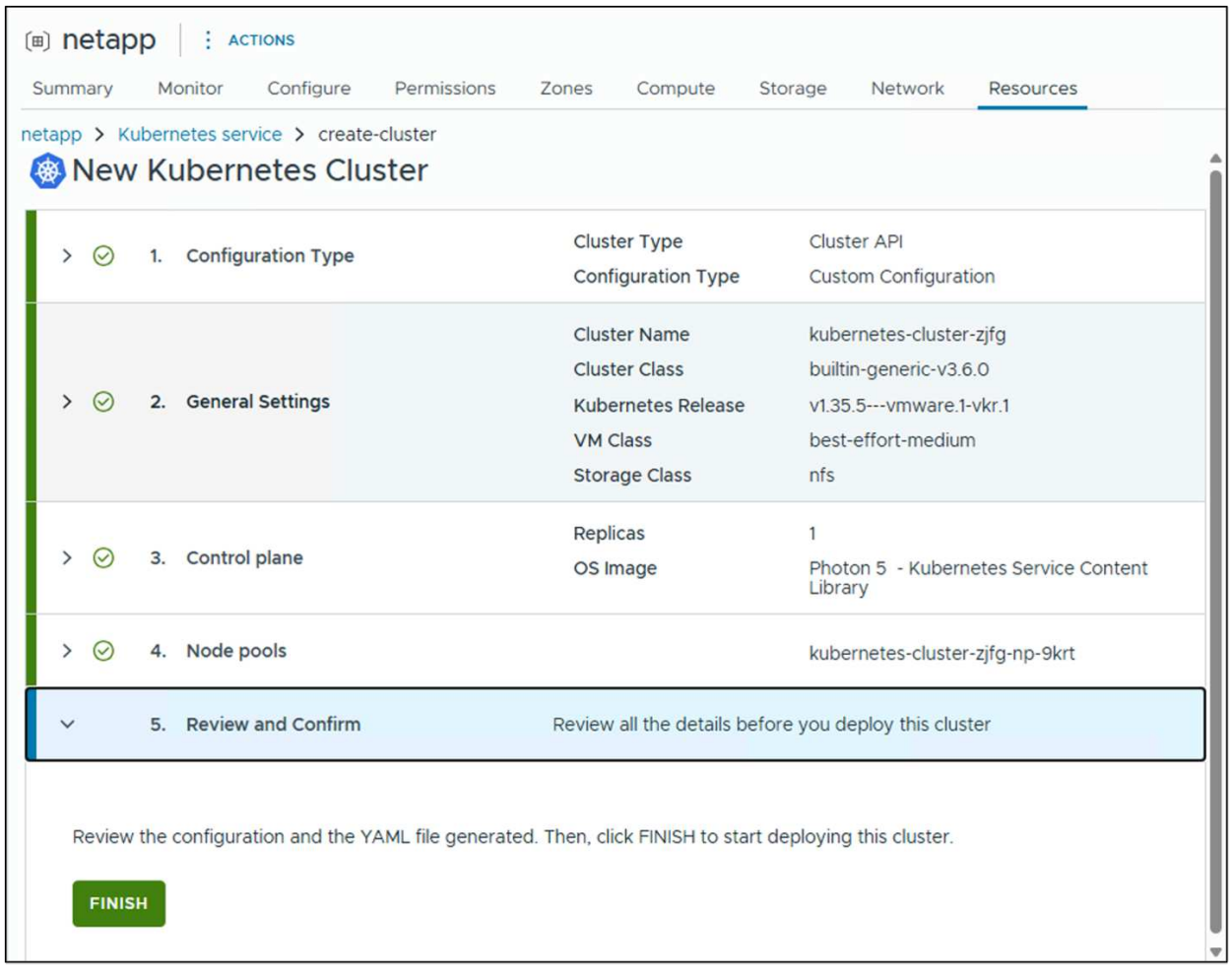

vSphere Client
Search in all environments
banum@nsol.netapp.com
Namespaces
Name
argocd
explore2026-n8lmx
my-vms
netapp
svc-argocd-service-r3ouo
svc-argocd-service-vi2eq
svc-auto-attach-v5oyx
svc-cci-ns-8pkaq
svc-cci-ns-rsqfv
svc-configuration-m0i9f
svc-contour-9xxgx
svc-harbor-adhi2
svc-metrics-aggregator-ckwu2
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

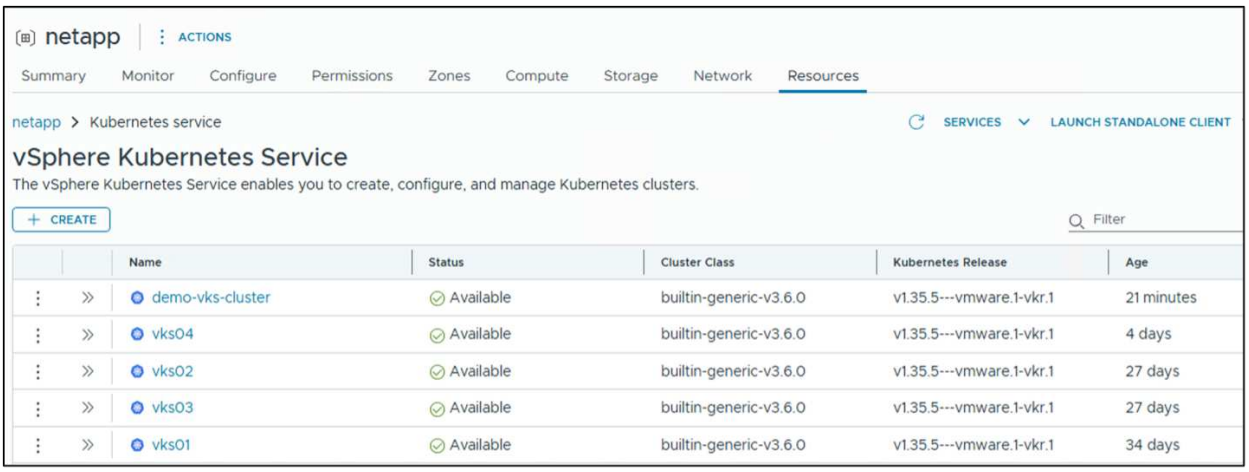

netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp > Kubernetes service > create-cluster
New Kubernetes Cluster
1. Configuration Type
How would you like to configure your Kubernetes cluster?
Create the cluster with a pre-defined default configuration or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



▼ 显示为启用 **SAN** 的工作节点创建自定义 **OVA** 映像的说明

创建一个安装了所需实用程序的 Docker 镜像。以下示例演示了如何基于 Ubuntu 24.04 创建安装了 iSCSI 和多路径实用程序的 Docker 镜像。

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

使用以下命令构建 Docker 镜像：

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

如果 Docker 注册表不可用，请运行此命令以启动本地注册表并将镜像推送到本地注册表：

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

标记并推送图片。

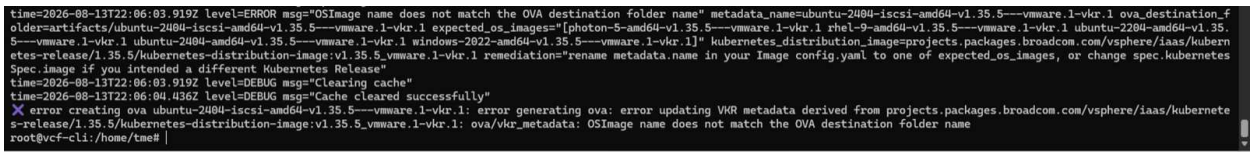
```
docker tag ubuntu-2404:san localhost:5000/ubuntu-2404:san
docker push localhost:5000/ubuntu-2404:san
```

创建映像配置文件（另存为 `ubuntu2404vkr135-san.yml`）并引用映像：

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

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

元数据名称应为预先批准列表中的名称。如果不是，则在尝试创建 OVA 文件时会出现如下所示的错误。预先批准的列表可在 `kr bake`命令的 VCF CLI 帮助中找到。



使用 VCF CLI 创建 OVA 文件：

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

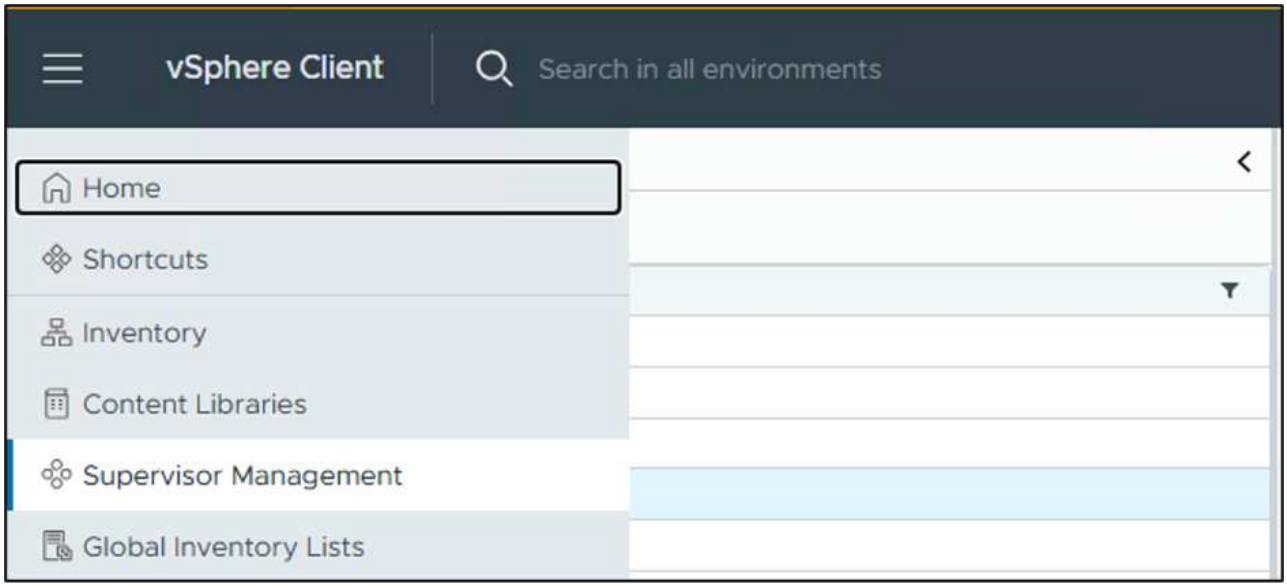
使用 SCP 将 OVA 文件传输到计算机，您可以从该计算机将其上传到 vSphere 内容库。

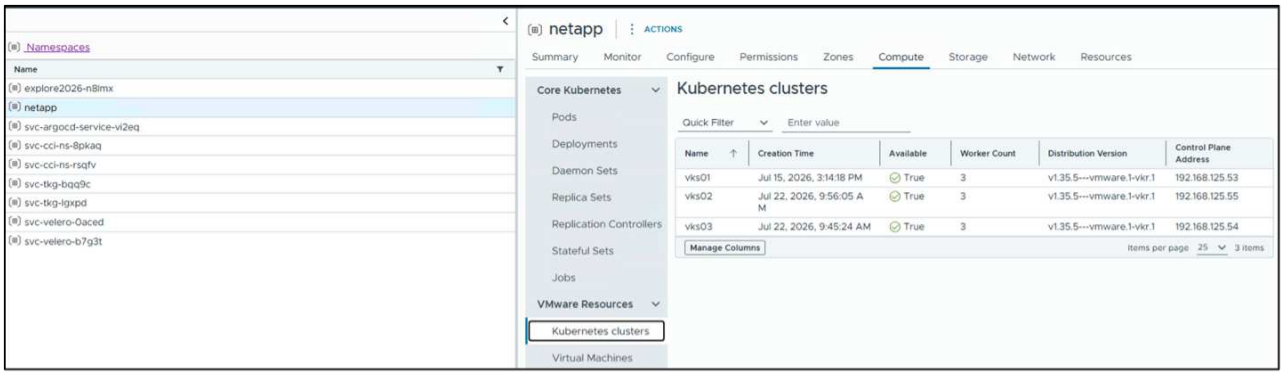
2. 安装集群后，在 vCenter 中找到它并下载 kubeconfig 文件。

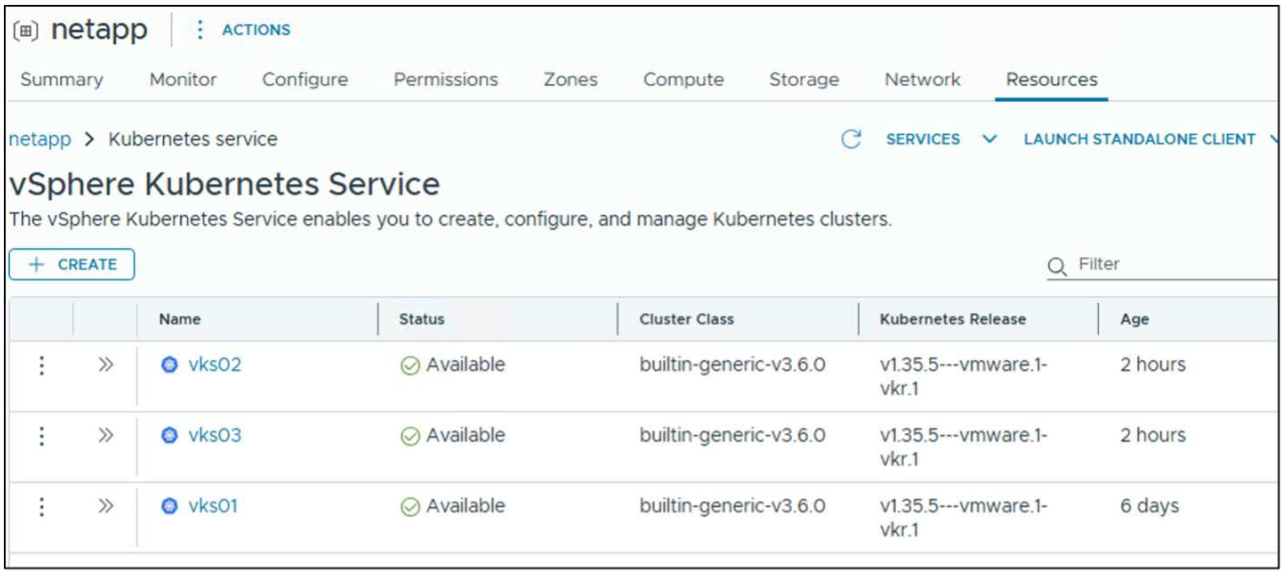
   a. 单击主菜单中的 **Supervisor Management**，然后选择创建集群的命名空间。

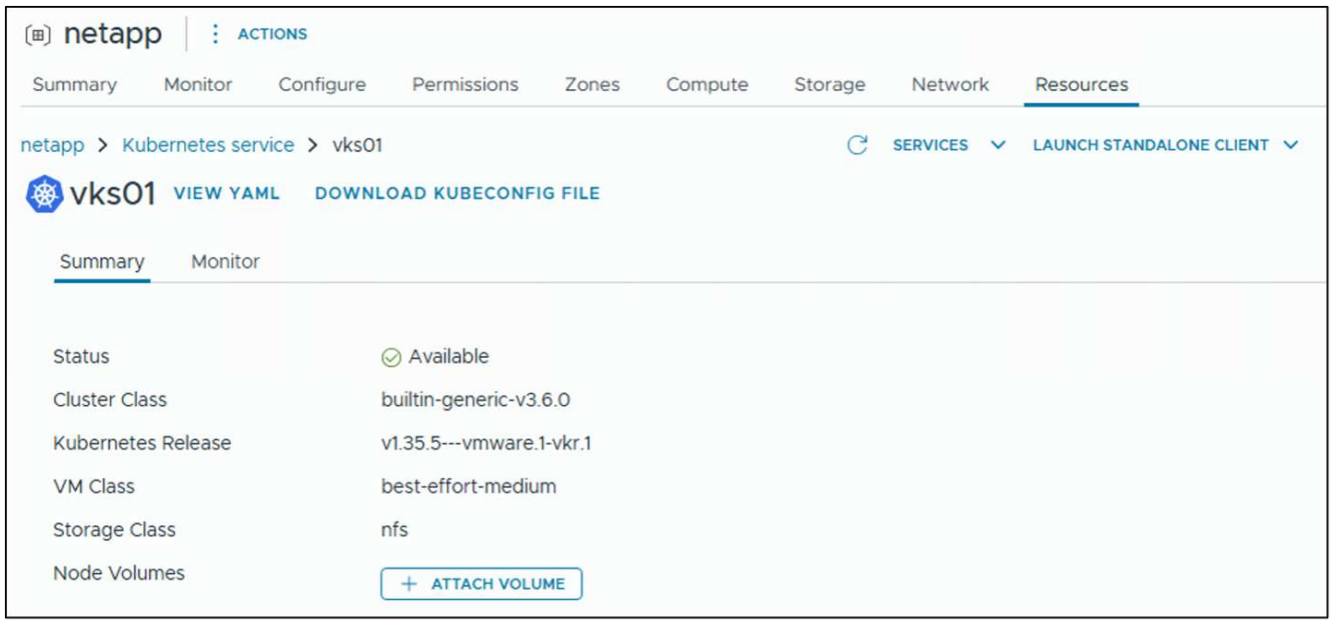
   b. 选择 **Compute** 选项卡，然后选择 **Kubernetes Clusters**。将显示命名空间中的所有集群。

   c. 转到 **Resources** 选项卡，选择所需的 Kubernetes 集群，然后下载其 kubeconfig 文件。


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. 从 bastion 主机，使用 kubeconfig 文件连接到集群，以便从 CLI 对其进行管理。

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

视频演示

以下视频演示如何创建 vSphere Kubernetes 集群。

在 Supervisor 集群上创建 vSphere Kubernetes 集群

了解适用于 **vSphere Kubernetes Service** 的 **NetApp** 存储

使用 NetApp 作为 vSphere Kubernetes 服务 (VKS) 的存储是一种强大的组合，它利用 NetApp 强大的存储解决方案来增强 Kubernetes 工作负载的性能、可扩展性和可靠性。NetApp Trident 是 Kubernetes 的开源动态存储配置程序，用于将存储与 vSphere Kubernetes 服务上的工作负载集成。

将 **NetApp** 存储与 **VKS** 配合使用的主要优势

1. 动态存储配置： NetApp Trident 允许动态配置存储卷，使 Kubernetes Pod 能够根据其需求即时请求存储。
2. 永久存储： NetApp 解决方案提供永久存储功能，确保 Pod 重启和故障期间的数据持久性和可用性。
3. 高性能： NetApp 的存储解决方案，如 ONTAP，提供低延迟的高性能存储，非常适合在 Kubernetes 上运行的 I/O 密集型应用程序。
4. 可扩展性： NetApp 存储系统可以扩展以满足不断增长的 Kubernetes 工作负载的需求，支持大规模部署。
5. 数据管理： NetApp 提供高级数据管理功能，包括快照、克隆和数据复制，这有助于备份、灾难恢复和开发工作流程。

6. 多云和混合云支持： NetApp 的存储解决方案可以部署到本地、公共云和混合云环境中，提供存储管理的灵活性和一致性。

NetApp ONTAP 集成概述

NetApp ONTAP 提供具有重复数据删除、数据压缩和加密等功能的企业级存储。您可以使用 NetApp Trident 将 NetApp 存储与 Kubernetes 集成。NetApp Trident 支持各种 NetApp 存储平台，包括本地 ONTAP、AWS 中的 FSx for NetApp ONTAP、Azure 中的 Azure NetApp Files 以及 Google Cloud 中的 Google Cloud NetApp Volumes。

您可以使用 Trident Protect 处理 Kubernetes 工作负载的数据保护和灾难恢复。Trident Protect 允许您在不同的存储后端创建快照、克隆和复制数据，以确保您的数据在出现故障时是安全且可恢复的。

Trident 的主要功能

CSI 合规性 确保与最新的 Kubernetes 存储功能和标准兼容。声明性配置 允许用户在 YAML 文件中定义存储要求，然后由 Trident 自动管理。驱动程序 Trident 支持 ONTAP 所支持的 NFS、CIFS/SMB、iSCSI、FC 和 NVMe/TCP 协议的驱动程序。混合云和多云支持 Trident 支持本地 ONTAP 存储及超大规模云服务商中托管存储服务的驱动程序，即 AWS 中的 FSx for NetApp ONTAP、Azure 中的 Azure NetApp Files 以及 Google Cloud 中的 Google Cloud NetApp Volumes。高级数据管理 利用 ONTAP 的快照和克隆功能，实现高效的数据保护以及测试和开发环境的快速配置。

有关 Trident 的完整详细信息，请参阅 "Trident 文档"。

Trident Protect

Trident Protect 与 Trident 分开安装。在部署之前，请验证您的 Kubernetes 平台、存储后端、快照组件和对象存储是否满足 "Trident Protect 要求"。

Trident Protect 的主要功能

自动备份 Trident Protect 支持 Kubernetes 应用程序的自动化、策略驱动备份，确保在无需手动干预的情况下定期备份。

快照管理 它利用 ONTAP 的快照功能来创建容器应用程序和虚拟机的频繁时间点副本，提供可靠的恢复机制。

备份存储库加密 Trident Protect 支持 Restic 和 Kopia 备份存储库的基于密码的加密。在存储和网络层中分别配置持久卷和传输中加密。

合规性和审计 提供工具和功能，帮助组织满足合规性要求并对其数据保护实践进行定期审计。

灾难恢复 与 ONTAP 的复制功能集成，提供强大的灾难恢复解决方案，即使在发生灾难性事件时也能确保业务连续性。

有关 Trident Protect 的完整详细信息，请参阅 "Trident Protect 文档"。

支持的 NetApp ONTAP 硬件型号和协议

NetApp ONTAP 软件运行在各种硬件型号上，每种型号都旨在满足不同的性能和容量要求。Trident 扩展了其强大的功能，以支持各种 NetApp ONTAP 系统，包括全闪存 FAS (AFF)、全 SAN 阵列 (ASA) 和 ASA R2 系统。这种支持可确保组织能够在容器化环境中充分利用其高性能存储解决方案的潜力。

全闪存 **FAS (AFF)** 系统 AFF 系统提供卓越的性能和低延迟，使其成为高需求应用的理想选择。

All SAN Array (ASA) Systems ASA 系统专为专用 SAN 环境而设计，可提供强大的性能和可靠性。

ASA R2 Systems ASA R2 系统为 SAN 环境提供了增强的特性和功能。

AFX NetApp AFX 是一种专为企业 AI 工作负载设计的专用、分解式全闪存存储架构。它提供高性能 NAS，允许独立扩展计算和容量。NetApp AFX 存储系统在存储层的实现方面与其他基于 ONTAP 的系统（ASA、AFF 和 FAS）有所不同。AFX 使用分布式文件系统，可实现高性能和可扩展性，而其他基于 ONTAP 的系统则使用传统的存储架构。

1. AFF 支持的协议：NFS、CIFS/SMB、iSCSI、FC、NVMe/TCP
2. ASA 支持的协议：iSCSI、FC、NVMe/TCP
3. ASA R2 支持的协议：iSCSI、FC、NVMe/TCP
4. AFX 支持的协议：从 Trident 25.10 开始，仅支持 NFS 协议；不支持 SMB 协议。

适用于 **ONTAP** 存储的 **Trident** 驱动程序

Trident 包括以下 CSI 驱动程序，每个驱动程序都能够在后端存储中配置卷。

对于支持的硬件型号上的 **NAS** 协议

1. ontap-nas：一个 PVC 映射到一个 PV，这是一个专用 FlexVol volume。
2. ontap-nas-economy：配置的每个 PV 都是一个 qtree，每个 FlexVol volume 具有可配置的 qtree 数量（默认值为 200，可在 50 到 300 之间配置）。

   一个 PVC 映射到一个 PV，即共享 FlexVol 卷上的 qtree。

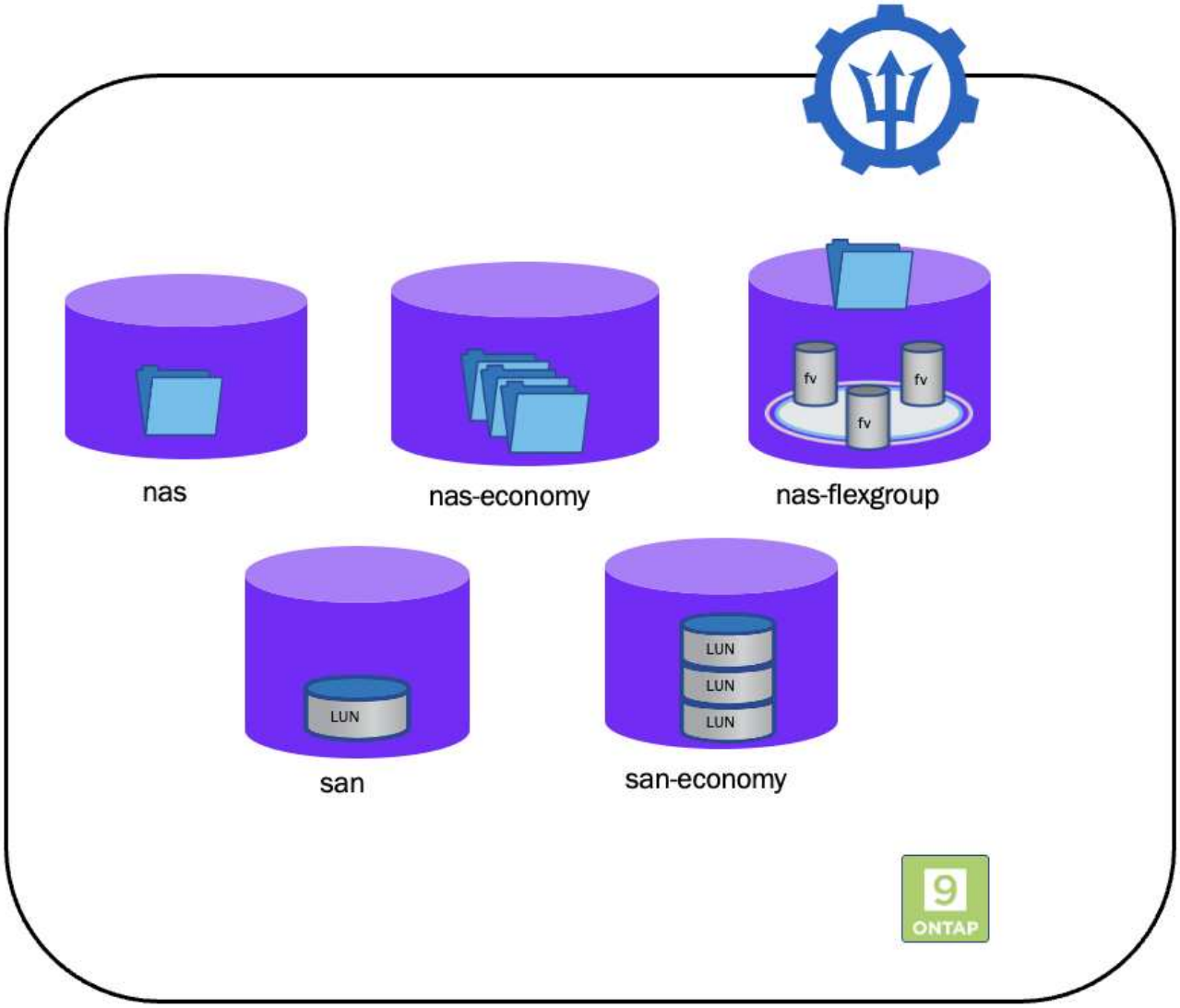
   > 您可以将虚拟机的所有磁盘作为 qtree 放在单个卷中。但是，请注意，Snapshots、Clones 和 Replication 功能不受 Trident 支持。当这些功能由存储管理员管理时，它们不具有 PV 粒度。

1. ontap-nas-flexgroup：配置的每个 PV 都是一个完整的 ONTAP FlexGroup，并使用分配给 SVM 的所有聚合。

对于支持的硬件型号上的 **SAN** 协议

1. ontap-san：一个 PVC 映射到一个 PV，即专用 FlexVol volume 上的 LUN。
2. ontap-san-economy：一个 PVC 映射到一个 PV，这是共享 FlexVol volume 上的 LUN。共享 FlexVol volume 中可以有多个 LUN（默认值为 100，每个 FlexVol volume 可配置 50 到 200 个 LUN/PV）。

   > 您可以将虚拟机的所有磁盘作为 LUN 放置在单个卷中。但是，此驱动程序支持快照和克隆，但不支持复制。当复制由存储管理员管理时，它不是 PV 粒度的。



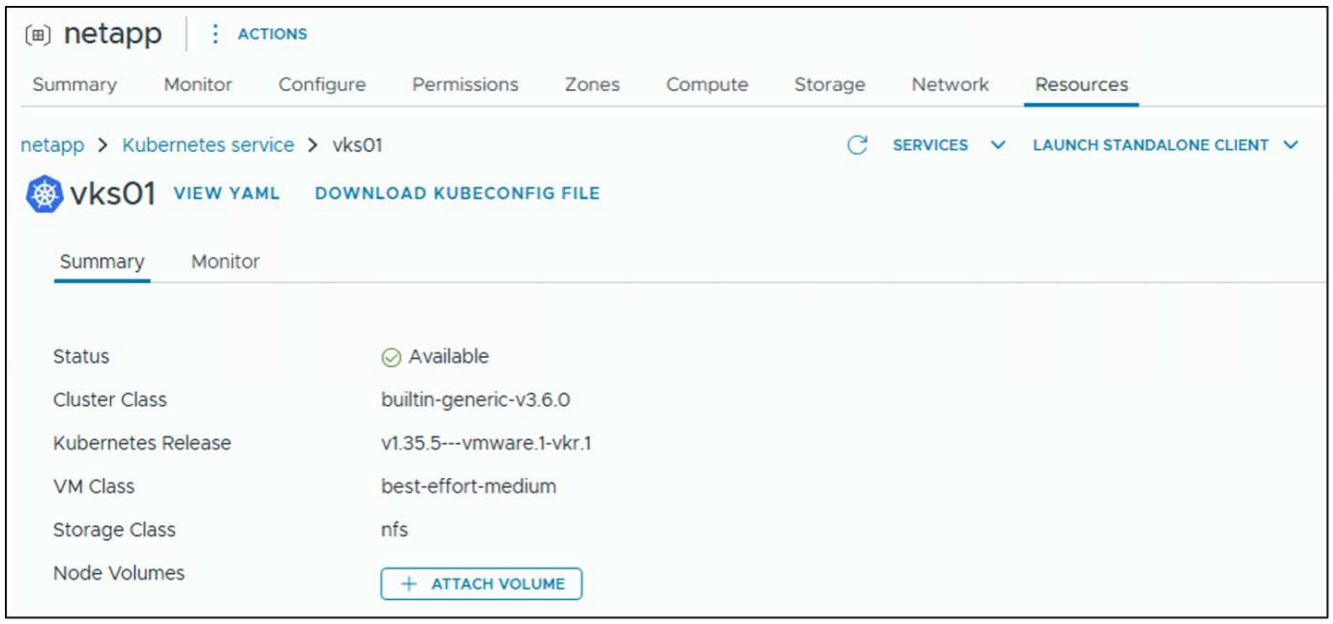
有关通过 Trident 驱动程序公开的功能以及必须由存储管理员应用的功能的完整列表，请参阅 Trident 文档中的"ONTAP 后端驱动程序"部分。

在 **VKS** 集群中安装 **NetApp Trident**

在您的 vSphere Kubernetes Service 集群中安装 NetApp Trident 作为动态存储置备程序，以将 NetApp ONTAP 存储与您的 Kubernetes 工作负载相集成。

开始之前

- 确保您的 ONTAP 集群已配置并连接到 VMware Kubernetes 环境。
- 如果您的工作负载将使用这些协议进行存储，请确保您的 VKS 集群已安装 iSCSI 或 NVMe 工具。
- 请确保您已在可以使用 kubeconfig 文件连接到 VKS 集群的计算机上安装 `kubectl`。



步骤

1. 按照 Trident 文档中的说明，使用 Helm chart 安装 Trident Operator：

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

   ▼ 显示示例

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

   要启用调试日志记录，请添加 `--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
   ```

   ⓘ 您也可以使用 Trident Operator 手动安装 Trident。查看 Trident 文档中的步骤："手动部署 Trident Operator"

2. 确认所有 Trident Pod 都在集群中运行。

   ▼ 显示示例

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

   ⓘ 此时可能无法创建 Trident Pod。如果您描述 ReplicaSet 对象，您可能会看到有关违反 PodSecurity 的错误，如下所示。此错误的原因是，在安装过程中创建的 `trident`命名空间强制执行 Kubernetes 受限 Pod 安全标准 (PSS)。Trident Operator 需要提升的系统权限来

> 管理主机存储，无法在受限配置文件的严格约束下运行。要解决此问题，请为 `trident` 命名空间授予特权配置文件，以便 ReplicaSet 控制器能够成功启动 Pod。

```
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")
```

将内置的 Kubernetes Pod Security Admission 标签应用于 `trident`命名空间，以将其从受限配置文件约束中豁免：

```
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$
```

为您的环境准备存储后端和 **StorageClass** 配置文件

1. 通过定义必要的连接详细信息和存储参数，为 ONTAP 配置 Trident 后端。
2. 在 Kubernetes 中定义映射到 NetApp 存储后端的存储类。这些类指定性能特征和复制策略等参数。

   根据工作负载的要求，为以下协议定义 Trident 后端和存储类：

- NAS（ontap-nas 驱动程序）
- NAS Economy（ontap-nas-economy driver）
- 用于 iSCSI、FC 或 NVMe 协议的 SAN（ontap-san 驱动程序）
- 适用于 iSCSI 的 SAN Economy（ontap-san-economy 驱动程序）

NAS

为 ONTAP NAS 创建 TridentBackendConfig 和 StorageClass，以启用基于 NFS 的持久存储配置。后端配置包括存储在 Kubernetes Secret 中的凭据，并引用您的 ONTAP SVM 和管理 LIF。

使用用户名和密码身份验证（另存为 `tbc-nas.yaml`）的后端密钥和后端配置文件示例：

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

使用客户端证书身份验证的后端密钥和后端配置文件示例（另存为 `tbc-nas.yaml`）：

```
# tbc-nas.yaml
apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-secret
  namespace: trident
type: Opaque
stringData:
  clientPrivateKey: <client private key value>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: <ONTAP management LIF>
  backendName: tbc-nas
  svm: <your SVM name>
  storagePrefix: <your prefix for all volume names created using
this backend configuration>
  credentials:
    name: ontap-nas-secret
  clientCertificate: <client certificate>
```

StorageClass 定义（另存为 `sc-nas.yaml`）：

`mountOptions` 是一个可选参数，用于指定 NFS 卷的其他挂载选项。该 `nconnect` 选项允许单个 NFS 挂载使用多个并行 TCP 连接，从而提高高吞吐量工作负载的性能。默认值为 1，但可以增加到 ONTAP 系统允许的最大值。

ONTAP 支持在支持 nconnect 的客户端上的单个 NFS 挂载最多 16 个连接。Trident 将挂载选项直接传递给 Kubernetes 工作节点，因此请选择工作节点 NFS 客户端和 ONTAP 均支持的值；以下示例使用四个连接。

```
# 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
```

创建 TridentBackendConfig 和 StorageClass，以便为 ONTAP NAS Economy 驱动程序启用使用 qtree 的基于 NFS 的持久存储配置。后端配置包括存储在 Kubernetes Secret 中的凭据，并引用您的 ONTAP SVM 和管理 LIF。

使用用户名和密码身份验证（另存为 `tbc-nas-economy.yaml`）的后端密钥和后端配置文件示例：

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

StorageClass 定义（另存为 `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
```

为使用 iSCSI 的 ONTAP SAN 创建 TridentBackendConfig 和 StorageClass，以启用基于 iSCSI 的块存储配置。后端配置使用 ontap-san 驱动程序，并包含存储在 Kubernetes Secret 中的凭据。如果省略，`sanType` 参数默认为 iSCSI 协议。

后端 secret 和后端配置文件使用用户名和密码身份验证（另存为 `tbc-iscsi.yaml`）：

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

使用客户端证书身份验证的后端 secret 和后端配置文件（另存为 `tbc-iscsi.yaml`）：

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

StorageClass 定义（另存为 `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
```

为使用 NVMe 的 ONTAP SAN 创建 TridentBackendConfig 和 StorageClass，以启用基于 NVMe 的块存储

配置。后端配置使用 ontap-san 驱动程序，并包含存储在 Kubernetes Secret 中的凭据。 `sanType` 参数必须设置为 `nvme`。

后端 secret 和后端配置文件使用用户名和密码身份验证（另存为 `tbc-nvme.yaml`）：

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

使用客户端证书身份验证的后端 secret 和后端配置文件（另存为 `tbc-nvme.yaml`）：

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

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

StorageClass 定义（另存为 `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
```

查看 Trident 文档，了解 YAML 文件的其他示例以及有关 SAN 驱动程序和 NAS 驱动程序支持的访问模式和卷模式的信息。

- "使用 NAS 驱动程序的示例"
- "使用 SAN 驱动程序的示例"

创建 **VolumeSnapshotClass** 配置文件

创建 VolumeSnapshotClass 定义。此配置为永久卷启用基于快照的操作。

VolumeSnapshotClass 定义（另存为 `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
```

将配置文件应用于集群

使用 `kubectl`将前面步骤中创建的配置文件应用到 vSphere Kubernetes Service 集群。这会在集群中创建必要的机密、后端配置、存储类和快照类。

步骤

1. 为您配置的每个协议应用 TridentBackendConfig 和 StorageClass 文件。

```
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. 验证资源是否已成功创建。

   检查 TridentBackendConfig 对象：

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

   检查 StorageClass 对象：

```
kubectl get storageclass
```

   检查 VolumeSnapshotClass：

```
kubectl get volumesnapshotclass
```

设置默认 **Trident** 存储和快照类

将 Trident StorageClass 和 VolumeSnapshotClass 设置为 vSphere Kubernetes Service 集群中的默认值。

步骤

1. 设置默认 Trident StorageClass。

   将 Trident 支持的 StorageClass 设置为集群默认值，以便在未指定存储类时 PersistentVolumeClaims 自动使用它。

   确保仅将一个 StorageClass 设置为默认值。如果另一个 StorageClass 已设置为默认值，则将其注释设置为 `false`。

   从 CLI：

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

2. 设置默认 Trident VolumeSnapshotClass。

   将 Trident 支持的 VolumeSnapshotClass 设置为集群默认值，为持久卷启用基于快照的操作。这可确保在未指定快照类时，VolumeSnapshots 自动使用 Trident CSI 驱动程序。

   确保仅将一个 VolumeSnapshotClass 设置为默认值。如果另一个 VolumeSnapshotClass 已设置为默认值，则将其注释设置为 `false`。

   从 CLI：

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

使用 **Kubernetes** 持久卷声明请求工作负载的存储

Trident 会自动从后端 NetApp 存储配置必要的卷。有关在 VKS 集群中使用 PVC 部署工作负载的示例，请参阅 "使用永久存储部署工作负载"。

部署具有 **NetApp** 永久存储的容器应用程序

使用 NetApp ONTAP 持久化存储在您的 vSphere Kubernetes Service 集群中部署容器化应用程序。以下示例演示了如何使用 NAS (ontap-nas) 或 SAN iSCSI (ontap-san) 存储部署 PostgreSQL 数据库。

以下 YAML 配置直接在清单中设置 POSTGRES_USER / POSTGRES_PASSWORD。在生产环境中，建议使用 Kubernetes Secrets 来管理数据库凭据等敏感信息。

使用 **NAS** 存储部署 **PostgreSQL**（**ontap-nas** 驱动程序）

您可以部署由 ontap-nas 驱动程序配置的 NFS 永久卷支持的 PostgreSQL 容器应用程序。以下示例演示如何为 PostgreSQL 创建 PersistentVolumeClaim（PVC）和 Deployment。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-pvc
spec:
  storageClassName: sc-nas # Specify the StorageClass for dynamic
provisioning
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      securityContext:
        fsGroup: 999  # Ensures proper permissions for the mounted
volume (PostgreSQL default user is 999)
      containers:
        - name: postgres
          image: postgres:15
          securityContext:
            allowPrivilegeEscalation: false
            runAsNonRoot: true
            runAsUser: 999  # PostgreSQL default user ID
            capabilities:
              drop:
                - ALL
            seccompProfile:
```

```
              type: RuntimeDefault
          env:
            - name: POSTGRES_DB
              value: "postgres"
            - name: POSTGRES_USER
              value: "<username>"
            - name: POSTGRES_PASSWORD
              value: "<password>"
            - name: PGDATA
              value: "/var/lib/postgresql/data/pgdata"
          ports:
            - containerPort: 5432
          volumeMounts:
            - mountPath: /var/lib/postgresql/data
              name: postgres-storage
              subPath: postgres-data  # Ensures data is stored in a
subdirectory to avoid permission issues
          resources:
            requests:
              memory: "512Mi"
              cpu: "250m"
            limits:
              memory: "1Gi"
              cpu: "500m"
      volumes:
        - name: postgres-storage
          persistentVolumeClaim:
            claimName: postgres-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
spec:
  type: ClusterIP
  ports:
    - port: 5432
      targetPort: 5432
  selector:
    app: postgres
```

使用 **SAN** 存储部署 **PostgreSQL**（**ontap-san iSCSI** 驱动程序）

使用以下 YAML 部署由 ontap-san 驱动程序配置的 iSCSI 永久卷支持的 PostgreSQL 容器应用程序。The `Recreate` 部署策略确保 Pod 在新 Pod 启动之前完全终止，这是 ReadWriteOnce iSCSI 卷所需的，并防止 Pod 重新启动时出现数据损坏。此配置支持跨 Pod 删除和重新创建的数据持久性，并与 Trident 卷快照以及备份和还原操作兼容。

```yaml
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
```

验证数据库部署

部署应用程序后，请验证 pod 是否处于运行状态以及数据库是否可操作。使用以下命令检查 Pod、PVC 和服务的状态。确保 Pod 正在运行，并且 PVC 已绑定到相应的卷。

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

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

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

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

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

根据使用的协议，后端存储是卷、qtree 或 LUN。您可以通过描述 PersistentVolume（PV）来检索内部卷名称，然后在 ONTAP CLI 命令中使用它来获取存储详细信息，从而验证在 ONTAP 系统中配置的存储。使用以下命令来描述 PV 并检索内部卷名称：

```
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>
```

图 **1.** 显示示例

```
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>
```

从输出中复制内部卷名，并运行以下命令以从 ONTAP CLI 获取存储详细信息。

对于 NAS 卷：

```
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
```

对于 SAN LUN，运行以下命令从 ONTAP CLI 获取 LUN 详细信息：

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

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

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

如果在 NAS 或 NAS Economy 存储类中使用了 `nconnect`挂载选项，则可以验证从工作节点到存储服务器的活动 TCP 连接数。

首先，列出 Trident 后端配置以查找后端名称，然后对其进行描述以检索 vserver 名称和数据 LIF：

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

然后登录到 ONTAP 集群并运行以下命令，替换上面输出的数据 LIF：

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

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

图 **2.** 显示示例

Pod 处于运行状态后，连接到数据库并验证数据操作。

步骤

1. 将 PostgreSQL 服务端口转发到本地计算机：

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

2. 从不同的终端，使用 psql 连接到数据库：

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

3. 创建数据库和表，然后使用数据填充表：

```
CREATE DATABASE employee;
```

切换到新数据库（psql 元命令）：

```
\c employee
```

创建表，插入一行，并查询结果：

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

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

SELECT * FROM employees;
```

视频演示

以下视频演示在 vSphere Kubernetes 集群上安装 Trident，以及使用 Trident 部署具有持久存储的 PostgreSQL 数据库。

使用 Trident 持久存储部署 ONTAP 工作负载

版权信息

商标信息