Red Hat OpenShift 与NetApp

NetApp container solutions

NetApp
January 21, 2026

目录

Red Hat OpenShift 与NetApp

NVA-1160：Red Hat OpenShift 与NetApp

NetApp 的Alan Cowles 和 Nikhil M Kulkarni

本参考文档提供了 Red Hat OpenShift 解决方案的部署验证，该解决方案通过安装程序配置基础架构 (IPI) 在多个不同的数据中心环境中部署，并由NetApp验证。它还详细介绍了如何利用Trident存储编排器管理持久存储，从而实现与NetApp存储系统的存储集成。最后，我们探索并记录了大量解决方案验证和实际用例。

使用情形

Red Hat OpenShift 与NetApp解决方案的架构旨在通过以下用例为客户提供卓越的价值：

- 易于部署和管理使用 IPI（安装程序配置基础架构）在裸机、Red Hat OpenStack 平台、Red Hat 虚拟化和 VMware vSphere 上部署的 Red Hat OpenShift。
- 将企业容器和虚拟化工作负载的功能与 Red Hat OpenShift 结合起来，在 OSP、RHV 或 vSphere 上虚拟部署，或者在带有 OpenShift Virtualization 的裸机上部署。
- 真实世界的配置和用例突出显示了 Red Hat OpenShift 与NetApp存储和Trident（ Kubernetes 的开源存储编排器）一起使用时的功能。

商业价值

企业越来越多地采用 DevOps 实践来创建新产品、缩短发布周期并快速添加新功能。由于其固有的敏捷特性，容器和微服务在支持 DevOps 实践方面发挥着至关重要的作用。然而，在企业环境中以生产规模实践 DevOps 也面临着自身的挑战，并对底层基础设施提出了一定的要求，例如：

- 堆栈中所有层的高可用性
- 简化部署程序
- 无中断运行和升级
- API 驱动和可编程的基础设施，以跟上微服务的敏捷性
- 具有性能保证的多租户
- 能够同时运行虚拟化和容器化工作负载
- 能够根据工作负载需求独立扩展基础设施

NetApp的 Red Hat OpenShift 承认这些挑战，并提出了一种解决方案，通过在客户选择的数据中心环境中实施 RedHat OpenShift IPI 的全自动部署来帮助解决每个问题。

技术概述

Red Hat OpenShift 与NetApp解决方案由以下主要组件组成：

红帽 **OpenShift** 容器平台

Red Hat OpenShift Container Platform 是一个完全支持的企业级 Kubernetes 平台。 Red Hat 对开源 Kubernetes 进行了多项改进，以提供一个完全集成所有组件的应用程序平台，用于构建、部署和管理容器化应用程序。

欲了解更多信息，请访问 OpenShift 网站 "此处"。

NetApp存储系统

NetApp拥有多种非常适合企业数据中心和混合云部署的存储系统。 NetApp产品组合包括NetApp ONTAP、NetApp Element和NetApp e-Series 存储系统，所有这些系统都可以为容器化应用程序提供持久存储。

欲了解更多信息，请访问NetApp网站 "此处"。

NetApp存储集成

Trident是一个开源且完全支持的容器和 Kubernetes 发行版（包括 Red Hat OpenShift）存储编排器。

欲了解更多信息，请访问Trident网站 "此处"。

高级配置选项

本节专门介绍实际用户在将此解决方案部署到生产中时可能需要执行的自定义，例如创建专用私有映像注册表或部署自定义负载均衡器实例。

已验证版本的当前支持矩阵

技术	目的	软件版本
NetApp ONTAP	存储	9.8、9.9.1、9.12.1
NetApp Element	存储	12.3
NetApp Trident	存储编排	22.01.0、23.04、23.07、23.10、24.02
红帽 OpenShift	容器编排	4.6 EUS，4.7，4.8，4.10，4.11，4.12，4.13，4.14
VMware vSphere	数据中心虚拟化	7.0、8.0.2

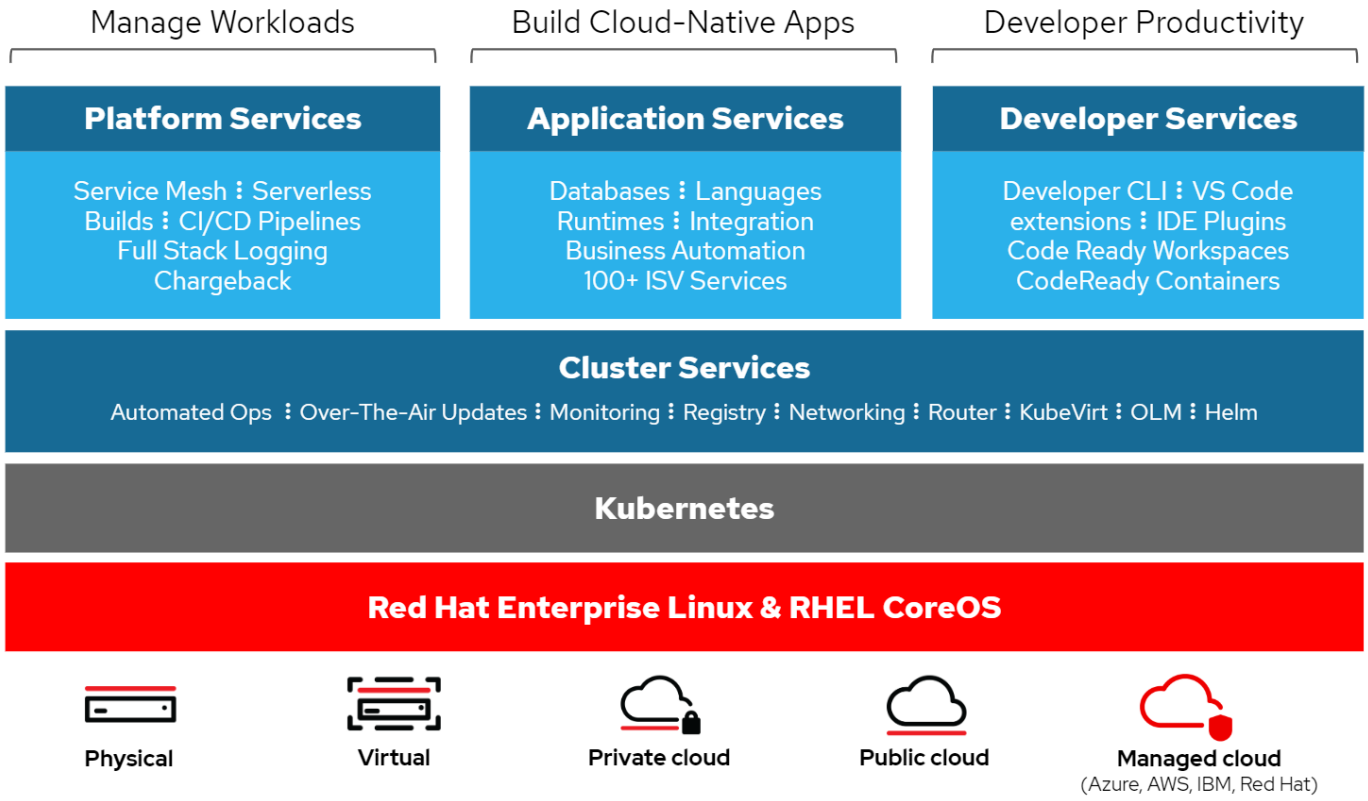
红帽 **OpenShift**

OpenShift 概述

Red Hat OpenShift 容器平台将开发和 IT 运营整合在一个平台上，以便在本地和混合云基础架构上一致地构建、部署和管理应用程序。 Red Hat OpenShift 基于开源创新和行业标准构建，包括 Kubernetes 和 Red Hat Enterprise Linux CoreOS，后者是专为基于容器的工作负载而设计的全球领先的企业 Linux 发行版。 OpenShift 是云原生计算基金会 (CNCF) 认证 Kubernetes 计划的一部分，提供容器工作负载的可移植性和互操作性。

Red Hat OpenShift 提供以下功能：

- 自助服务配置 开发人员可以使用他们最常用的工具快速轻松地按需创建应用程序，同时操作保留对整个环境的完全控制。
- 持久存储 通过提供对持久存储的支持，OpenShift Container Platform 允许您同时运行有状态应用程序和云原生无状态应用程序。
- 持续集成和持续开发 **(CI/CD)** 该源代码平台可大规模管理构建和部署映像。
- 开源标准 这些标准除了其他开源技术外，还结合了开放容器计划 (OCI) 和 Kubernetes 用于容器编排。您不会受到特定供应商的技术或业务路线图的限制。
- **CI/CD** 管道 OpenShift 为 CI/CD 管道提供开箱即用的支持，以便开发团队可以自动化应用程序交付过程的每个步骤，并确保在对应用程序的代码或配置所做的每个更改上执行它。
- 基于角色的访问控制 **(RBAC)** 此功能提供团队和用户跟踪，以帮助组织大型开发人员群体。
- 自动构建和部署 OpenShift 为开发人员提供了构建容器化应用程序的选项，或者让平台从应用程序源代码甚至二进制文件构建容器。然后，该平台根据为应用程序定义的特性自动在整个基础设施中部署这些应用程序。例如，为了符合第三方许可证，应该分配多少资源以及在基础设施的什么位置部署它们。
- 一致的环境 OpenShift 确保为开发人员提供的环境以及应用程序整个生命周期的环境都是一致的，从操作系统到库、运行时版本（例如 Java 运行时），甚至正在使用的应用程序运行时（例如 tomcat），以消除源自不一致环境的风险。
- 配置管理 平台内置配置和敏感数据管理，以确保无论使用哪种技术构建应用程序或部署在哪种环境中，都为应用程序提供一致且与环境无关的应用程序配置。
- *应用程序日志和指标。*快速反馈是应用程序开发的一个重要方面。 OpenShift 集成监控和日志管理向开发人员提供即时指标，以便他们研究应用程序在变化中的行为，并能够在应用程序生命周期中尽早解决问题。
- 安全和容器目录 OpenShift 提供多租户功能，并使用安全增强型 Linux (SELinux)、CGroups 和安全计算模式 (seccomp) 等已建立的安全性来隔离和保护容器，从而保护用户免受有害代码执行的侵害。它还通过 TLS 证书为各个子系统提供加密，并提供对 Red Hat 认证容器（access.redhat.com/containers）的访问，这些容器经过扫描和分级，特别强调安全性，以便为最终用户提供经过认证、值得信赖和安全的应用程序容器。



Red Hat OpenShift 的部署方法

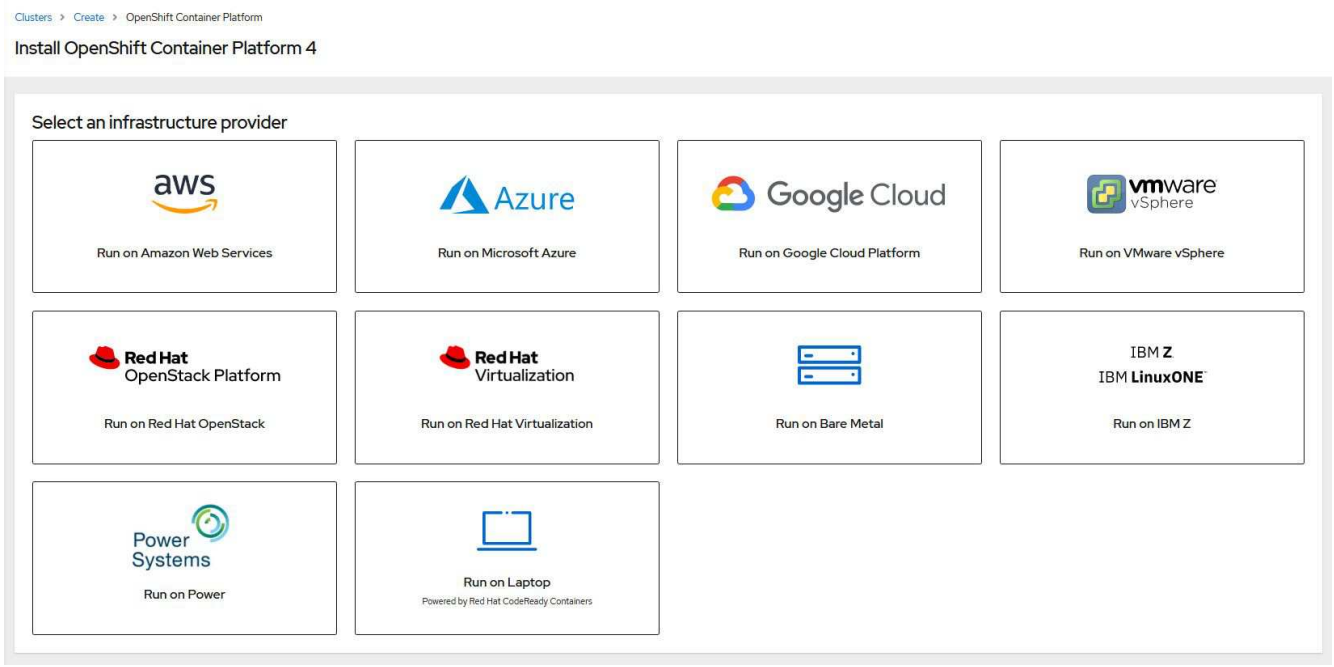
从 Red Hat OpenShift 4 开始，OpenShift 的部署方法包括使用用户配置基础设施 (UPI) 进行高度定制的部署的手动部署或使用安装程序配置基础设施 (IPI) 进行全自动部署。

在大多数情况下，IPI 安装方法是首选方法，因为它允许为开发、测试和生产环境快速部署 OpenShift 集群。

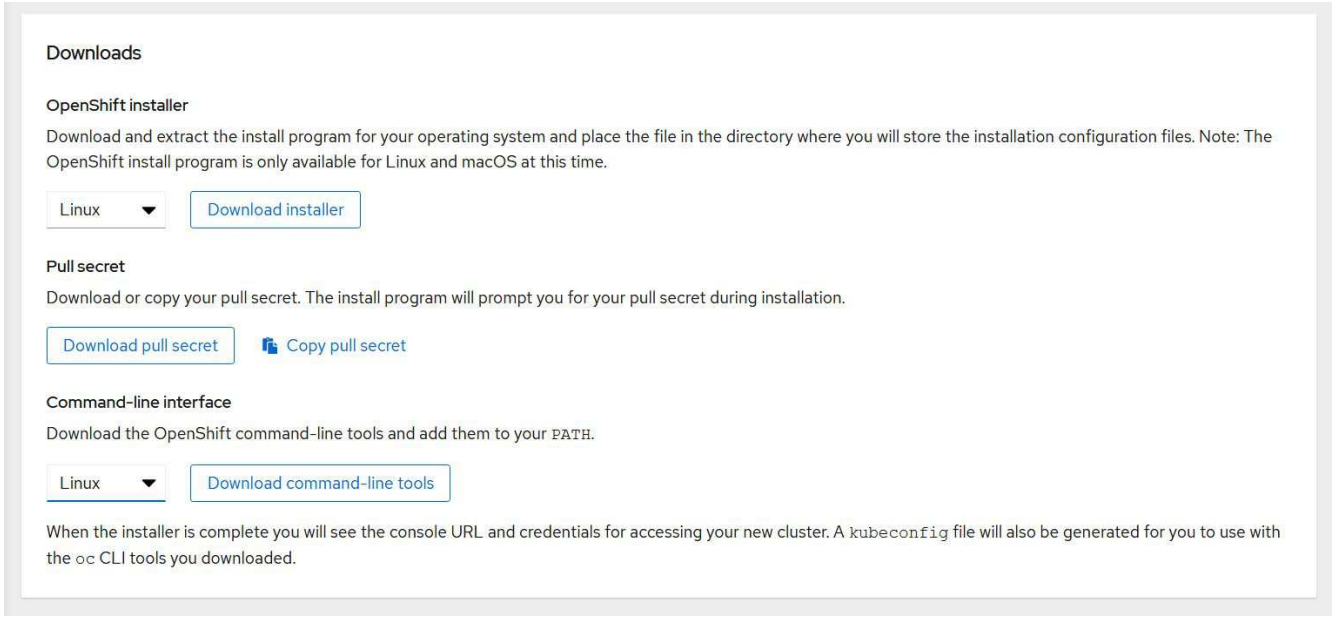
Red Hat OpenShift 的 IPI 安装

OpenShift 的安装程序配置基础设施 (IPI) 部署涉及以下高级步骤：

1. 访问 Red Hat OpenShift"网站"并使用您的 SSO 凭据登录。
2. 选择您想要部署 Red Hat OpenShift 的环境。



3. 在下一个屏幕上下载安装程序、唯一的拉取密钥和用于管理的 CLI 工具。



4. 关注"安装说明"由 Red Hat 提供，可部署到您选择的环境中。

NetApp验证的 **OpenShift** 部署

NetApp已在其实验室中使用安装程序配置基础架构 (IPI) 部署方法在以下每个数据中心环境中测试并验证了 Red Hat OpenShift 的部署：

- "裸机上的 OpenShift"
- "Red Hat OpenStack 平台上的 OpenShift"
- "Red Hat 虚拟化上的 OpenShift"
- "VMware vSphere 上的 OpenShift"

裸机上的 **OpenShift**

OpenShift on Bare Metal 可在商用服务器上自动部署 OpenShift 容器平台。

OpenShift on Bare Metal 类似于 OpenShift 的虚拟部署，它可以轻松部署、快速配置和扩展 OpenShift 集群，同时支持尚未准备好容器化的应用程序的虚拟化工作负载。通过在裸机上部署，您不需要除了 OpenShift 环境之外管理主机虚拟机管理程序环境所需的额外开销。通过直接在裸机服务器上部署，您还可以减少主机和 OpenShift 环境之间共享资源的物理开销限制。

OpenShift on Bare Metal 提供以下功能：

- **IPI** 或辅助安装程序部署 通过在裸机服务器上由安装程序配置基础设施 (IPI) 部署的 OpenShift 集群，客户可以直接在商用服务器上部署高度通用、易于扩展的 OpenShift 环境，而无需管理虚拟机管理程序层。
- 紧凑的集群设计 为了最大限度地减少硬件要求，裸机上的 OpenShift 允许用户部署仅 3 个节点的集群，通过使 OpenShift 控制平面节点也可以充当工作节点和主机容器。
- **OpenShift** 虚拟化 OpenShift 可以使用 OpenShift 虚拟化在容器内运行虚拟机。这种容器原生虚拟化在容器内运行 KVM 虚拟机管理程序，并为 VM 存储附加持久卷。
- **AI/ML** 优化基础设施 通过将基于 GPU 的工作节点合并到您的 OpenShift 环境并利用 OpenShift Advanced Scheduling，为机器学习应用程序部署 Kubeflow 等应用程序。

网络设计

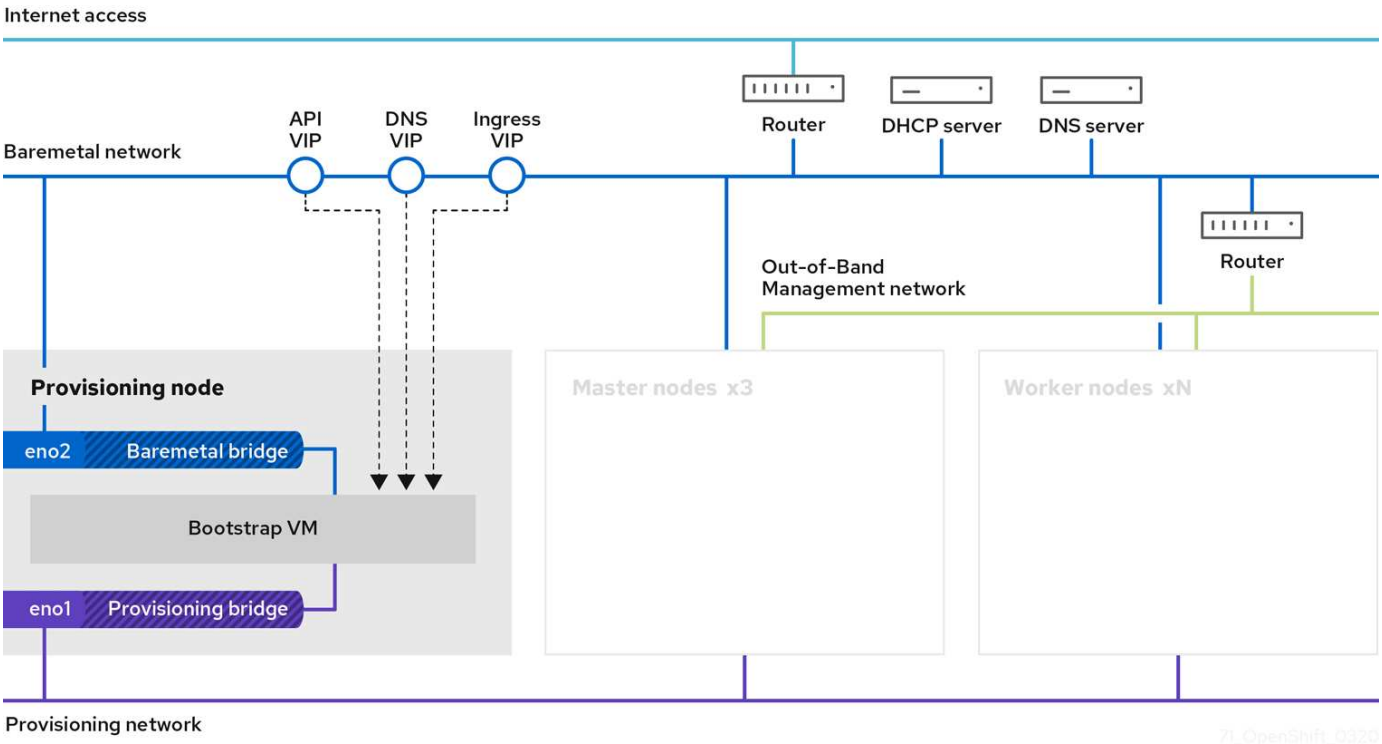
NetApp解决方案上的 Red Hat OpenShift 使用两个数据交换机以 25Gbps 提供主要数据连接。它还使用两个管理交换机，以 1Gbps 的速度提供存储节点的带内管理连接以及 IPMI 功能的带外管理连接。

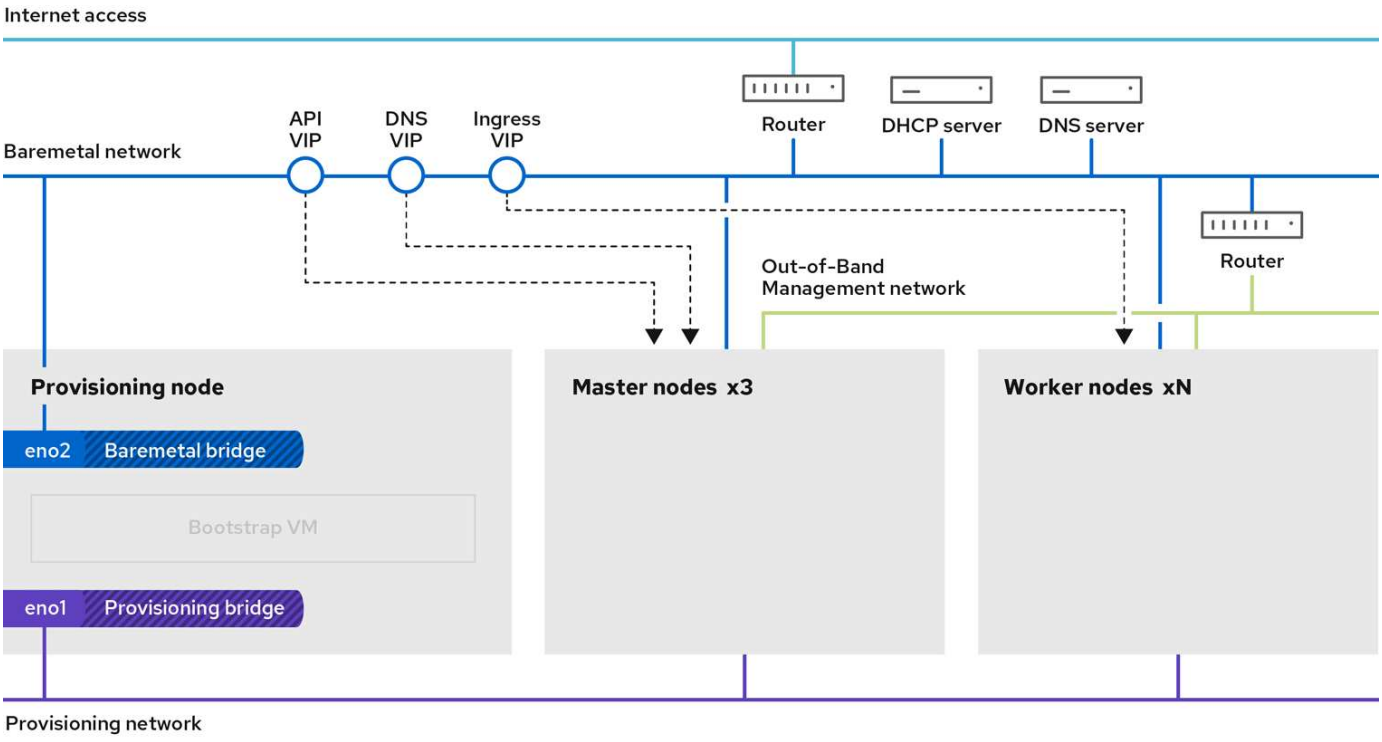
对于 OpenShift 裸机 IPI 部署，您必须创建一个配置节点，即一台必须具有连接到单独网络的网络接口的 Red Hat Enterprise Linux 8 机器。

- 配置网络 该网络用于启动裸机节点并安装部署 OpenShift 集群所需的映像和包。
- 裸机网络 该网络用于集群部署后面向公众的通信。

为了设置配置节点，客户创建桥接接口，允许流量在节点本身和为部署目的而配置的 Bootstrap VM 上正确路由。集群部署完成后，API 和 ingress VIP 地址从 bootstrap 节点迁移到新部署的集群。

下图描述了 IPI 部署期间和部署完成后的环境。





VLAN 要求

Red Hat OpenShift 与NetApp解决方案旨在通过使用虚拟局域网 (VLAN) 在逻辑上分离用于不同用途的网络流量。

VLAN	目的	VLAN ID
带外管理网络	裸机节点和 IPMI 管理	16
裸机网络	集群可用时，OpenShift 服务的网络	181
配置网络	通过 IPI 进行 PXE 启动和安装裸机节点的网络	3485

尽管这些网络实际上是由 VLAN 分隔的，但每个物理端口都必须设置为访问模式并分配主 VLAN ，因为在 PXE 启动序列期间无法传递 VLAN 标签。

网络基础设施支持资源

在部署 OpenShift 容器平台之前，应具备以下基础设施：

- 至少有一个 DNS 服务器提供可从带内管理网络和 VM 网络访问的完整主机名解析。
- 至少一个可从带内管理网络和 VM 网络访问的 NTP 服务器。
- （可选）带内管理网络和 VM 网络的出站互联网连接。

Red Hat OpenStack 平台上的 OpenShift

Red Hat OpenStack 平台提供了一个集成的基础来创建、部署和扩展安全可靠的私有 OpenStack 云。

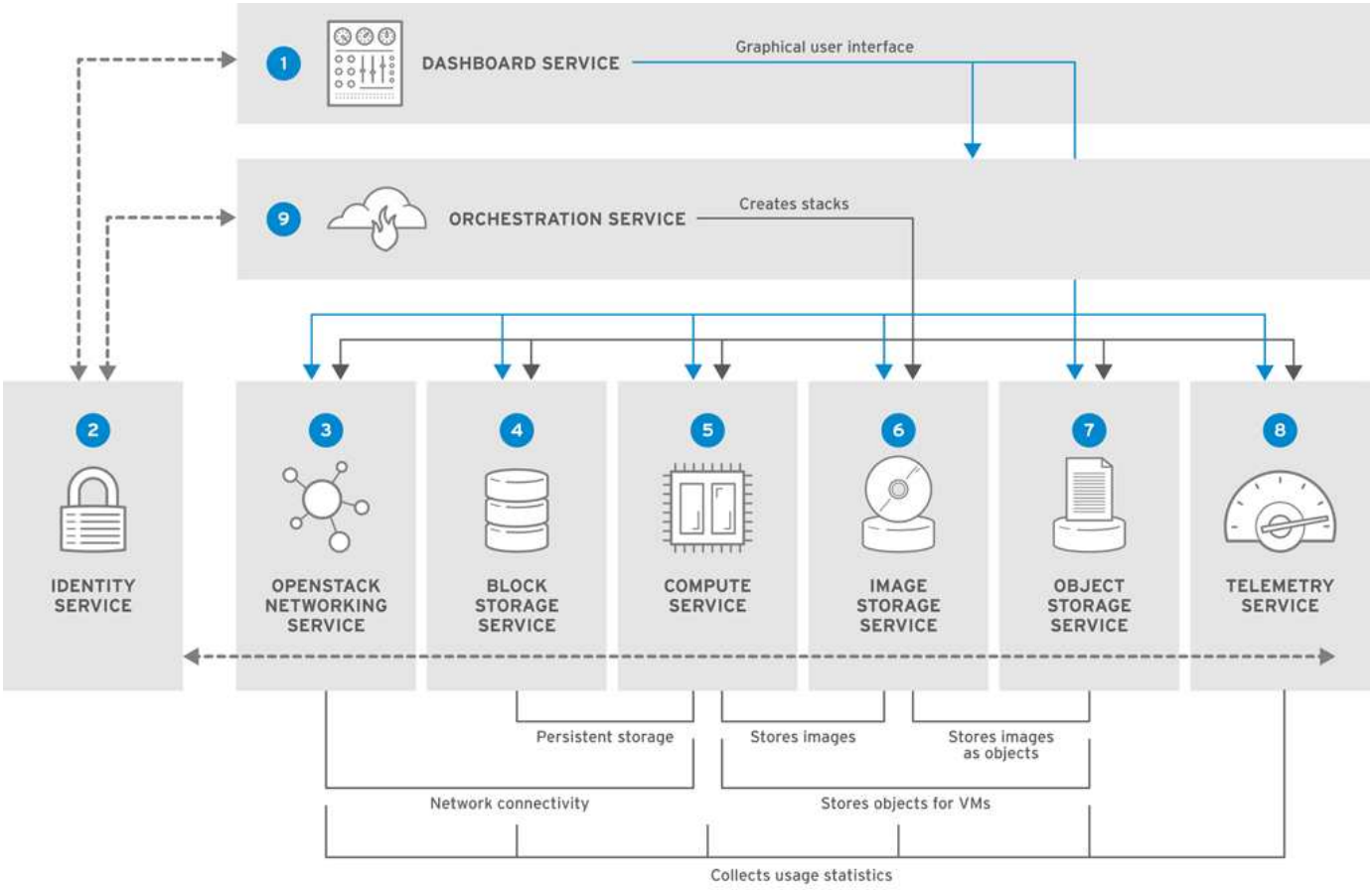
OSP 是一种基础设施即服务 (IaaS) 云，由管理计算、存储和网络资源的一系列控制服务实现。该环境使用基于 Web 的界面进行管理，允许管理员和用户控制、配置和自动化 OpenStack 资源。此外，OpenStack 基础设施通过广泛的命令行界面和 API 实现，为管理员和最终用户提供完全自动化功能。

OpenStack项目是一个快速发展的社区项目，每六个月提供一次更新版本。最初，Red Hat OpenStack Platform 通过随每个上游版本发布新版本并为每三个版本提供长期支持来跟上此发布周期。最近，随着 OSP 16.0 版本（基于 OpenStack Train）的发布，Red Hat 选择不跟上版本号，而是将新功能反向移植到子版本中。最新版本是 Red Hat OpenStack Platform 16.1，其中包括从 Ussuri 和 Victoria 上游版本移植的高级功能。

有关 OSP 的更多信息，请参阅"红帽 OpenStack 平台网站"。

OpenStack服务

OpenStack 平台服务以容器形式部署，从而将服务彼此隔离并实现轻松升级。 OpenStack 平台使用一组通过 Kolla 构建和管理的容器。服务的部署是通过从 Red Hat Custom Portal 提取容器镜像来执行的。这些服务容器使用 Podman 命令进行管理，并使用 Red Hat OpenStack Director 进行部署、配置和维护。



服务	项目名称	描述
信息板	Horizon	用于管理 OpenStack 服务的基于 Web 浏览器的仪表板。
身份	Keystone	用于对 OpenStack 服务进行身份验证和授权以及管理用户、项目和角色的集中服务。
OpenStack 网络连接	中子	提供 OpenStack 服务接口之间的连接。
块存储	Cinder	管理虚拟机 (VM) 的持久块存储卷。
计算	Nova	管理和配置在计算节点上运行的虚拟机。
映像	Glance	用于存储虚拟机镜像、卷快照等资源的注册服务。
对象存储	Swift	允许用户存储和检索文件和任意数据。
遥测	云高仪	提供云资源使用情况的测量。
编排	热	基于模板的编排引擎，支持自动创建资源栈。

网络设计

Red Hat OpenShift 与NetApp解决方案使用两个数据交换机以 25Gbps 提供主要数据连接。它还使用两个额外的管理交换机，以 1Gbps 的速度提供存储节点的带内管理连接以及 IPMI 功能的带外管理连接。

Red Hat OpenStack Director 需要 IPMI 功能来使用 Ironic 裸机配置服务部署 Red Hat OpenStack Platform。

VLAN 要求

Red Hat OpenShift 与NetApp旨在通过使用虚拟局域网 (VLAN) 在逻辑上分离用于不同用途的网络流量。此配置可以扩展以满足客户需求或为特定网络服务提供进一步的隔离。下表列出了在NetApp验证解决方案时实施该解决方案所需的 VLAN。

VLAN	目的	VLAN ID
带外管理网络	用于管理 Ironic 的物理节点和 IPMI 服务的网络。	16
存储基础设施	用于控制器节点直接映射卷以支持 Swift 等基础设施服务的网络。	201
储存煤渣	用于将块卷直接映射并附加到环境中部署的虚拟实例的网络。	202
内部 API	用于使用 API 通信、RPC 消息和数据库通信的 OpenStack 服务之间的通信的网络。	301
Tenant（租户）	Neutron 通过 VXLAN 隧道为每个租户提供自己的网络。网络流量在每个租户网络内是隔离的。每个租户网络都有一个与之关联的 IP 子网，网络命名空间意味着多个租户网络可以使用相同的地址范围而不会引起冲突。	302
存储管理	OpenStack 对象存储（Swift）使用此网络在参与的副本节点之间同步数据对象。代理服务充当用户请求和底层存储层之间的中介接口。代理接收传入的请求并找到必要的副本以检索请求的数据。	303
PXE	OpenStack Director 提供 PXE 启动作为 Ironic 裸机配置服务的一部分，以协调 OSP Overcloud 的安装。	3484
外部	公共可用网络，托管用于图形管理的 OpenStack 仪表板（Horizon），并允许公共 API 调用来管理 OpenStack 服务。	3485
带内管理网络	提供对系统管理功能（例如 SSH 访问、DNS 流量和网络时间协议 (NTP) 流量）的访问。该网络还充当非控制器节点的网关。	3486

网络基础设施支持资源

在部署 OpenShift 容器平台之前，应具备以下基础设施：

- 至少一个提供完整主机名解析的 DNS 服务器。
- 至少三个NTP服务器可以为解决方案中的服务器保持时间同步。
- （可选）OpenShift 环境的出站互联网连接。

生产部署的最佳实践

本节列出了组织在将此解决方案部署到生产中之前应考虑的几种最佳实践。

将 OpenShift 部署到具有至少三个计算节点的 OSP 私有云

本文档中描述的经过验证的架构通过部署三个 OSP 控制节点和两个 OSP 计算节点，提供了适合 HA 操作的最小硬件部署。该架构确保了容错配置，其中两个计算节点都可以启动虚拟实例，并且部署的虚拟机可以在两个虚拟机管理程序之间迁移。

由于 Red Hat OpenShift 最初部署了三个主节点，因此双节点配置可能会导致至少两个主节点占用同一个节点，如果该特定节点不可用，则可能导致 OpenShift 中断。因此，Red Hat 的最佳实践是部署至少三个 OSP 计算节点，以便 OpenShift 主机可以均匀分布，并且解决方案可以获得额外的容错程度。

配置虚拟机/主机亲和性

通过启用 VM/主机亲和性，可以将 OpenShift 主机分布在多个虚拟机管理程序节点上。

亲和性是一种为虚拟机和/或主机集合定义规则的方法，用于确定虚拟机是否在同一主机或组中的主机上一起运行，还是在不同的主机上运行。它通过创建由具有一组相同参数和条件的虚拟机和/或主机组成的亲和性组应用于虚拟机。根据亲和性组中的虚拟机是运行在组内的同一主机上还是运行在组内的多个主机上，还是分别运行在不同的主机上，亲和性组的参数可以定义正亲和性或负亲和性。在 Red Hat OpenStack Platform 中，可以通过创建服务器组和配置过滤器来创建和强制执行主机亲和性和反亲和性规则，以便 Nova 在服务器组中部署的实例部署在不同的计算节点上。

一个服务器组最多可以管理 10 个虚拟实例。可以通过更新 Nova 的默认配额来修改这一点。

> OSP 服务器组存在特定的硬亲和性/反亲和性限制；如果没有足够的资源部署在单独的节点上或没有足够的资源允许共享节点，则虚拟机将无法启动。

要配置关联组，请参阅"如何为 OpenStack 实例配置亲和性和反亲和性？"。

使用自定义安装文件进行 **OpenShift** 部署

IPI 通过本文档前面讨论的交互式向导使 OpenShift 集群的部署变得简单。但是，您可能需要在集群部署过程中更改一些默认值。

在这种情况下，您可以运行向导并执行任务，而无需立即部署集群；它会创建一个配置文件，稍后可以从中部署集群。如果您需要更改任何 IPI 默认值，或者如果您想在环境中部署多个相同的集群以用于多租户等其他用途，这将非常有用。有关创建 OpenShift 自定义安装配置的更多信息，请参阅 "Red Hat OpenShift 在 OpenStack 上安装自定义集群"。

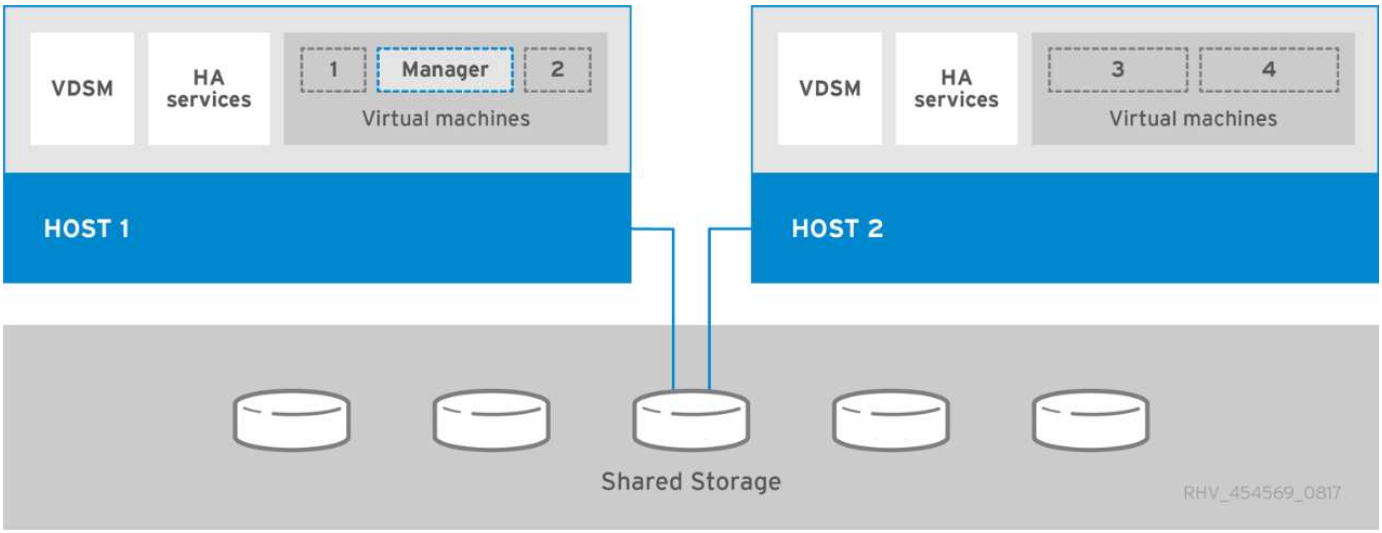
Red Hat 虚拟化上的 OpenShift

Red Hat Virtualization (RHV) 是一个企业虚拟数据中心平台，运行在 Red Hat Enterprise Linux (RHEL) 上并使用 KVM 虚拟机管理程序。

有关 RHV 的更多信息，请参阅"Red Hat 虚拟化网站"。

RHV 提供以下功能：

- 虚拟机和主机的集中管理 RHV 管理器在部署中作为物理机或虚拟机 (VM) 运行，并提供基于 Web 的 GUI，以便从中央界面管理解决方案。
- 自托管引擎 为了最大限度地减少硬件要求，RHV 允许将 RHV 管理器 (RHV-M) 作为 VM 部署在运行来宾 VM 的同一主机上。
- 高可用性 为避免主机故障时造成中断，RHV 允许对虚拟机进行高可用性配置。高可用性虚拟机在集群级别使用弹性策略进行控制。
- 高可扩展性 单个 RHV 集群最多可拥有 200 个虚拟机管理程序主机，使其能够支持大量虚拟机托管资源密集型企业级工作负载的需求。
- 增强的安全性 从 RHV 继承而来，RHV 采用安全虚拟化 (sVirt) 和安全增强 Linux (SELinux) 技术来提高主机和虚拟机的安全性和强化程度。这些功能的主要优势是虚拟机及其相关资源的逻辑隔离。



网络设计

NetApp解决方案上的 Red Hat OpenShift 使用两个数据交换机以 25Gbps 提供主要数据连接。它还使用两个额外的管理交换机，以 1Gbps 的速度提供存储节点的带内管理连接以及 IPMI 功能的带外管理。 OCP使用RHV上的虚拟机逻辑网络进行集群管理。本节介绍解决方案中使用的每个虚拟网段的安排和用途，并概述部署解决方案的先决条件。

VLAN 要求

RHV 上的 Red Hat OpenShift 旨在通过使用虚拟局域网 (VLAN) 在逻辑上分离用于不同目的的网络流量。此配置可以扩展以满足客户需求或为特定网络服务提供进一步的隔离。下表列出了在NetApp验证解决方案时实施该解决方案所需的 VLAN。

VLAN	目的	VLAN ID
带外管理网络	物理节点和IPMI的管理	16
虚拟机网络	虚拟访客网络访问	1172
带内管理网络	RHV-H 节点、RHV-Manager 和 ovirtmgmt 网络的管理	3343
存储网络	NetApp Element iSCSI 的存储网络	3344
移民网络	虚拟客户迁移网络	3345

网络基础设施支持资源

在部署 OpenShift 容器平台之前，应具备以下基础设施：

- 至少有一个 DNS 服务器提供完整的主机名解析，可从带内管理网络和 VM 网络访问。
- 至少一个可从带内管理网络和 VM 网络访问的 NTP 服务器。
- （可选）带内管理网络和 VM 网络的出站互联网连接。

生产部署的最佳实践

本节列出了组织在将此解决方案部署到生产中之前应考虑的几种最佳实践。

将 **OpenShift** 部署到至少三个节点的 **RHV** 集群

本文档中描述的经过验证的架构通过部署两个 RHV-H 虚拟机管理程序节点并确保容错配置（其中两个主机都可以管理托管引擎并且部署的虚拟机可以在两个虚拟机管理程序之间迁移）提供了适合 HA 操作的最小硬件部署。

由于 Red Hat OpenShift 最初部署有三个主节点，因此在双节点配置中可以确保至少两个主节点占用同一个节点，如果该特定节点不可用，则可能导致 OpenShift 中断。因此，Red Hat 的最佳实践是将至少三个 RHV-H 虚拟机管理程序节点作为解决方案的一部分进行部署，以便 OpenShift 主机可以均匀分布，并且解决方案可以获得额外的容错程度。

配置虚拟机/主机亲和性

您可以通过启用 VM/主机亲和性将 OpenShift 主机分布在多个虚拟机管理程序节点上。

亲和性是一种为虚拟机和/或主机集合定义规则的方法，用于确定虚拟机是否在同一主机或组中的主机上一起运行，还是在不同的主机上运行。它通过创建由具有一组相同参数和条件的虚拟机和/或主机组成的亲和性组应用于虚拟机。根据亲和性组中的虚拟机是运行在组内的同一主机上还是运行在组内的多个主机上，还是分别运行在不同的主机上，亲和性组的参数可以定义正亲和性或负亲和性。

参数定义的条件可以是硬执行，也可以是软执行。硬执行确保亲和性组中的虚拟机始终严格遵循正亲和性或负亲和性，而不考虑任何外部条件。软执行确保为亲和性组中的虚拟机设置更高的优先级，以便在可行的情况下遵循正或负亲和性。在本文档描述的两个或三个虚拟机管理程序配置中，软亲和性是推荐的设置。在较大的集群中，硬亲和性可以正确分配 OpenShift 节点。

要配置关联组，请参阅"红帽 6.11。亲和性群组文档"。

使用自定义安装文件进行 **OpenShift** 部署

IPI 通过本文档前面讨论的交互式向导使 OpenShift 集群的部署变得简单。但是，作为集群部署的一部分，可能需要更改一些默认值。

在这些情况下，您可以运行并执行向导，而无需立即部署集群。而是创建一个配置文件，以便以后可以部署集群。如果您想要更改任何 IPI 默认值，或者想要在您的环境中部署多个相同的集群用于其他用途（例如多租户），这将非常有用。有关为 OpenShift 创建自定义安装配置的更多信息，请参阅"Red Hat OpenShift 在 RHV 上安装具有自定义的集群"。

VMware vSphere 上的 OpenShift

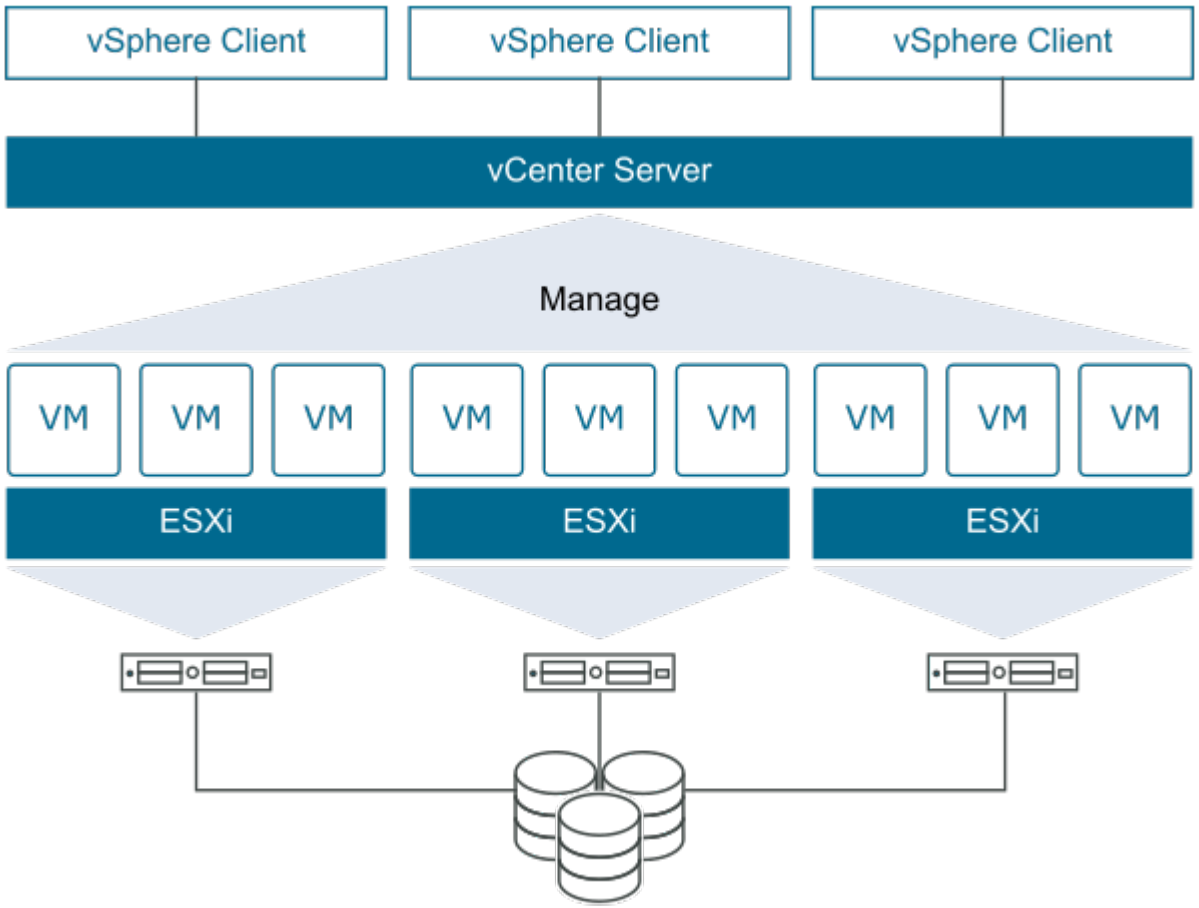
VMware vSphere 是一个虚拟化平台，用于集中管理在 ESXi 虚拟机管理程序上运行的大量虚拟化服务器和网络。

有关 VMware vSphere 的更多信息，请参见"VMware vSphere 网站"。

VMware vSphere 提供以下功能：

- **VMware vCenter Server** VMware vCenter Server 从单个控制台统一管理所有主机和虚拟机，并汇总集群、主机和虚拟机的性能监控。
- **VMware vSphere vMotion** VMware vCenter 允许您以无中断的方式根据请求在集群中的节点之间热迁移虚拟机。
- **vSphere** 高可用性 为避免主机发生故障时造成中断，VMware vSphere 允许对主机进行集群化并配置为高可用性。因主机故障而中断的虚拟机将在集群中的其他主机上短暂重新启动，从而恢复服务。

- 分布式资源调度程序 **(DRS)** 可以配置 VMware vSphere 集群以平衡其托管的虚拟机的资源需求。存在资源争用的虚拟机可以热迁移到集群中的其他节点，以确保有足够的资源可用。



网络设计

NetApp解决方案上的 Red Hat OpenShift 使用两个数据交换机以 25Gbps 提供主要数据连接。它还使用两个额外的管理交换机，以 1Gbps 的速度提供存储节点的带内管理连接以及 IPMI 功能的带外管理连接。 OCP 使用 VMware vSphere 上的 VM 逻辑网络进行集群管理。本节描述了解决方案中使用的每个虚拟网络段的安排和用途，并概述了部署解决方案的先决条件。

VLAN 要求

VMware vSphere 上的 Red Hat OpenShift 旨在通过使用虚拟局域网 (VLAN) 在逻辑上分离用于不同目的的网络流量。此配置可以扩展以满足客户需求或为特定网络服务提供进一步的隔离。下表列出了在NetApp验证解决方案时实施该解决方案所需的 VLAN。

VLAN	目的	**VLAN ID**
带外管理网络	物理节点和IPMI的管理	16
虚拟机网络	虚拟访客网络访问	181
存储网络	ONTAP NFS 的存储网络	184
存储网络	ONTAP iSCSI 的存储网络	185
带内管理网络	ESXi 节点、VCenter Server、 ONTAP Select的管理	3480

VLAN	目的	VLAN ID
存储网络	NetApp Element iSCSI 的存储网络	3481
移民网络	虚拟客户迁移网络	3482

网络基础设施支持资源

在部署 OpenShift 容器平台之前，应具备以下基础设施：

- 至少有一个 DNS 服务器提供完整的主机名解析，可从带内管理网络和 VM 网络访问。
- 至少一个可从带内管理网络和 VM 网络访问的 NTP 服务器。
- （可选）带内管理网络和 VM 网络的出站互联网连接。

生产部署的最佳实践

本节列出了组织在将此解决方案部署到生产中之前应考虑的几种最佳实践。

将 **OpenShift** 部署到至少三个节点的 **ESXi** 集群

本文档中描述的经过验证的架构通过部署两个 ESXi 虚拟机管理程序节点并通过启用 VMware vSphere HA 和 VMware vMotion 确保容错配置，提供了适合 HA 操作的最低硬件部署。此配置允许已部署的虚拟机在两个虚拟机管理程序之间迁移，并在一个主机不可用时重新启动。

由于 Red Hat OpenShift 最初部署有三个主节点，因此在某些情况下，双节点配置中至少有两个主节点可以占用同一个节点，如果该特定节点不可用，则可能导致 OpenShift 中断。因此，Red Hat 的最佳实践是必须部署至少三个 ESXi 虚拟机管理程序节点，以便 OpenShift 主机可以均匀分布，从而提供额外的容错程度。

配置虚拟机和主机关联性

通过启用 VM 和主机亲和性，可以确保 OpenShift 主机分布在多个虚拟机管理程序节点上。

亲和性或反亲和性是一种为虚拟机和/或主机集定义规则的方法，用于确定虚拟机是否在同一主机或组中的主机上一起运行，还是在不同的主机上运行。它通过创建由具有一组相同参数和条件的虚拟机和/或主机组成的亲和性组应用于虚拟机。根据亲和性组中的虚拟机是运行在组内的同一主机上还是运行在组内的多个主机上，还是分别运行在不同的主机上，亲和性组的参数可以定义正亲和性或负亲和性。

要配置亲缘组，请参阅 "vSphere 9.0 文档：使用 DRS 关联性规则"。

使用自定义安装文件进行 **OpenShift** 部署

IPI 通过本文档前面讨论的交互式向导使 OpenShift 集群的部署变得简单。但是，您可能需要在集群部署过程中更改一些默认值。

在这些情况下，您可以运行向导并执行任务而不立即部署集群，而是向导会创建一个配置文件，稍后可以从中部署集群。如果您需要更改任何 IPI 默认值，或者想要在您的环境中部署多个相同的集群用于其他用途（例如多租户），这将非常有用。有关为 OpenShift 创建自定义安装配置的更多信息，请参阅"Red Hat OpenShift 在 vSphere 上安装具有自定义设置的集群"。

AWS 上的 Red Hat OpenShift 服务

AWS 上的 Red Hat OpenShift 服务 (ROSA) 是一项托管服务，您可以使用它在 AWS 上通

过 Red Hat OpenShift 企业 Kubernetes 平台构建、扩展和部署容器化应用程序。 ROSA 简化了将本地 Red Hat OpenShift 工作负载迁移到 AWS 的过程，并提供了与其他 AWS 服务的紧密集成。

有关 ROSA 的更多信息，请参阅此处的文档："AWS 上的 Red Hat OpenShift 服务（AWS 文档）"。"AWS 上的 Red Hat OpenShift 服务（Red Hat 文档）"。

NetApp存储系统

NetApp ONTAP

NetApp ONTAP是一款功能强大的存储软件工具，具有直观的 GUI、具有自动化集成的 REST API、基于 AI 的预测分析和纠正措施、无中断硬件升级以及跨存储导入等功能。

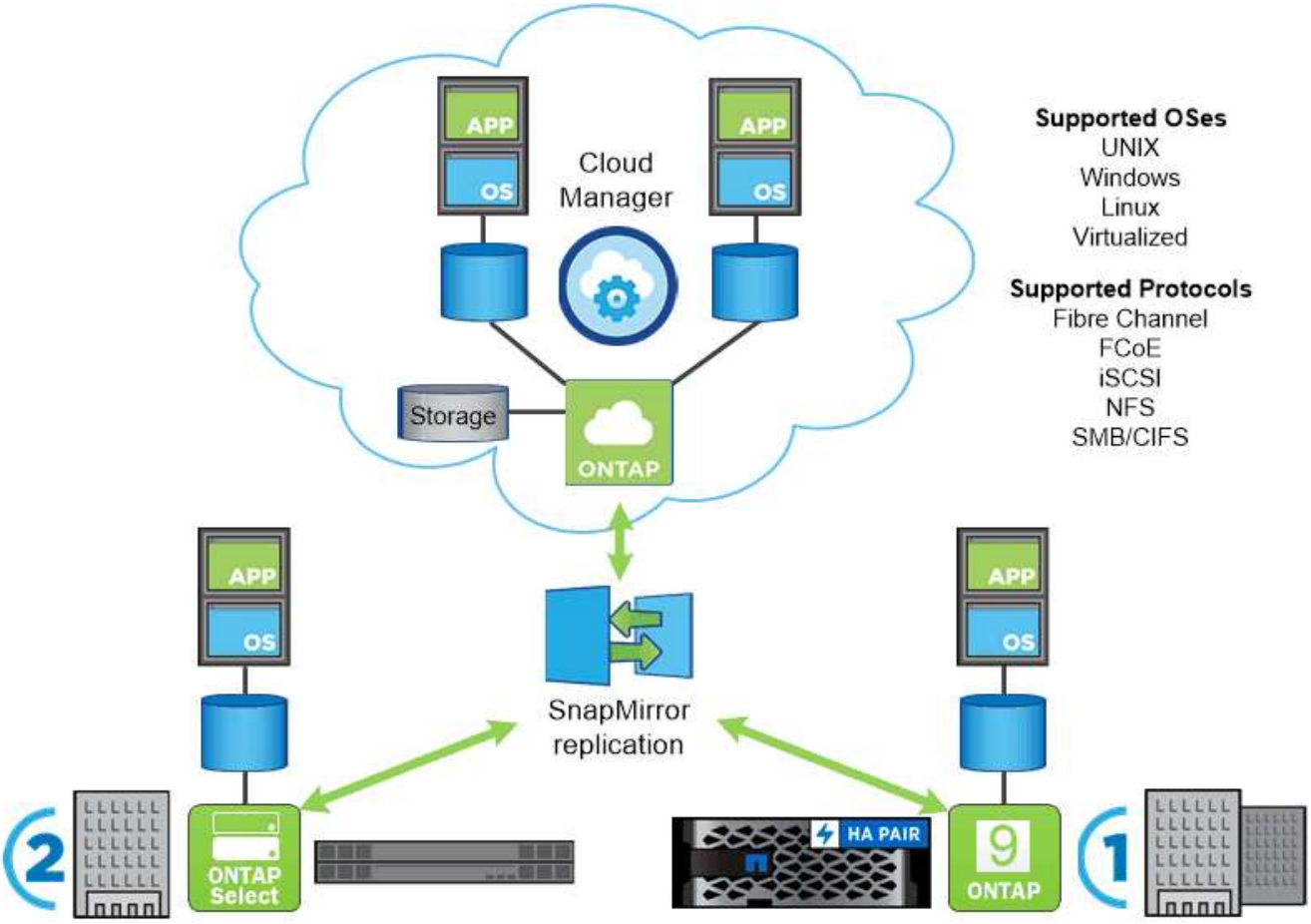
有关NetApp ONTAP存储系统的更多信息，请访问 "NetApp ONTAP网站"。

ONTAP提供以下功能：

- 一个统一的存储系统，可同时访问和管理 NFS、CIFS、iSCSI、FC、FCoE 和 FC-NVMe 协议的数据。
- 不同的部署模型包括全闪存、混合和全 HDD 硬件配置上的本地部署；受支持的虚拟机管理程序（如ONTAP Select）上的基于 VM 的存储平台；以及在云中的Cloud Volumes ONTAP。
- 通过支持自动数据分层、内联数据压缩、重复数据删除和压缩，提高了ONTAP系统上的数据存储效率。
- 基于工作负载、QoS 控制的存储。
- 与公共云无缝集成，实现数据分层和保护。 ONTAP还提供强大的数据保护功能，使其在任何环境中都脱颖而出：
  - * NetApp Snapshot 副本。*使用最少的磁盘空间快速进行时间点数据备份，且不产生额外的性能开销。
  - * NetApp SnapMirror。*将数据的 Snapshot 副本从一个存储系统镜像到另一个存储系统。 ONTAP还支持将数据镜像到其他物理平台和云原生服务。
  - * NetApp SnapLock。*通过将不可重写数据写入在指定时间内无法覆盖或擦除的特殊卷，可以有效地管理这些数据。
  - * NetApp SnapVault。*将来自多个存储系统的数据备份到中央 Snapshot 副本，作为所有指定系统的备份。
  - * NetAppSyncMirror。*为物理连接到同一控制器的两个不同磁盘丛提供实时 RAID 级数据镜像。
  - * NetApp SnapRestore。*提供从 Snapshot 副本按需快速恢复备份数据的功能。
  - * NetApp FlexClone。*根据 Snapshot 副本提供NetApp卷的完全可读、可写副本的即时配置。

有关ONTAP的更多信息，请参阅 "ONTAP 9 文档中心"。

NetApp ONTAP可在本地、虚拟化或云中使用。



NetApp平台

NetApp AFF/ FAS

NetApp提供强大的全闪存 (AFF) 和横向扩展混合 (FAS) 存储平台，这些平台具有低延迟性能、集成数据保护和多协议支持等特点。

这两种系统均由NetApp ONTAP数据管理软件提供支持，这是业界最先进的数据管理软件，具有高可用性、云集成、简化的存储管理功能，可提供您的数据结构所需的企业级速度、效率和安全性。

有关 NETAPP AFF/ FAS平台的更多信息，请单击 "此处"。

ONTAP Select

ONTAP Select是NetApp ONTAP的软件定义部署，可以部署到您环境中的虚拟机管理程序上。它可以安装在 VMware vSphere 或 KVM 上，并提供基于硬件的ONTAP系统的全部功能和体验。

有关ONTAP Select的更多信息，请单击 "此处"。

Cloud Volumes ONTAP

NetApp Cloud Volumes ONTAP是NetApp ONTAP的云部署版本，可部署在许多公共云中，包括：Amazon AWS、Microsoft Azure 和 Google Cloud。

有关Cloud Volumes ONTAP 的更多信息，请单击 "此处"。

Amazon FSx ONTAP

Amazon FSx ONTAP通过ONTAP流行的数据访问和管理功能在 AWS 云中提供完全托管的共享存储。有关Amazon FSx ONTAP 的更多信息，请单击 "此处"。

Azure NetApp Files

Azure NetApp Files是 Azure 原生的、第一方的、企业级的高性能文件存储服务。它提供卷即服务，您可以为其创建NetApp帐户、容量池和卷。您还可以选择服务和性能级别并管理数据保护。您可以使用熟悉并依赖的相同协议和工具来创建和管理高性能、高可用性和可扩展的文件共享。有关Azure NetApp Files 的更多信息，请单击 "此处"。

Google Cloud NetApp Volumes

Google Cloud NetApp Volumes是一种完全托管的基于云的数据存储服务，可提供高级数据管理功能和高度可扩展的性能。它允许您将基于文件的应用程序移动到 Google Cloud。它内置支持网络文件系统（NFSv3 和 NFSv4.1）和服务器消息块（SMB）协议，因此您无需重新构建应用程序，并可以继续为您的应用程序获取持久存储。有关 Google Cloud NetApp VolumesP 的更多信息，请点击 "此处"。

NetApp Element：Red Hat OpenShift 与NetApp

NetApp Element软件提供模块化、可扩展的性能，每个存储节点都为环境提供有保证的容量和吞吐量。 NetApp Element系统可以在单个集群中从 4 个节点扩展到 100 个节点，并提供许多高级存储管理功能。

有关NetApp Element存储系统的更多信息，请访问 "NetApp Solidfire 网站"。

iSCSI 登录重定向和自我修复功能

NetApp Element软件利用 iSCSI 存储协议，这是在传统 TCP/IP 网络上封装 SCSI 命令的标准方法。当 SCSI 标准发生变化或以太网性能提高时，iSCSI 存储协议将受益，而无需进行任何更改。

尽管所有存储节点都有一个管理 IP 和一个存储 IP，但NetApp Element软件会为集群中的所有存储流量公布一个存储虚拟 IP 地址（SVIP 地址）。作为 iSCSI 登录过程的一部分，存储可以响应目标卷已移动到不同的地址，因此无法继续协商过程。然后，主机会向新地址重新发出登录请求，整个过程不需要主机端重新配置。此过程称为 iSCSI 登录重定向。

iSCSI 登录重定向是NetApp Element软件集群的关键部分。当收到主机登录请求时，节点根据 IOPS 和卷的容量要求决定集群中的哪个成员应该处理流量。卷分布在NetApp Element软件集群中，如果单个节点处理的卷流量过多或添加了新节点，则会重新分配。在整个阵列中分配给定卷的多个副本。

通过这种方式，如果节点故障后进行卷重新分配，则除了注销并登录并重定向到新位置之外，对主机连接没有影响。借助 iSCSI 登录重定向， NetApp Element软件集群是一种可自我修复、横向扩展的架构，能够实现无中断升级和运行。

NetApp Element软件集群 QoS

NetApp Element软件集群允许根据每个卷动态配置 QoS。您可以使用每个卷的 QoS 设置根据您定义的 SLA 来控制存储性能。以下三个可配置参数定义 QoS：

- 最低 **IOPS**。 NetApp Element软件集群为卷提供的最小持续 IOPS 数。为卷配置的最小 IOPS 是卷的保证性能级别。每卷的性能不会低于这个水平。
- 最大 **IOPS**。 NetApp Element软件集群为特定卷提供的最大持续 IOPS 数。
- *突发 IOPS。*短时间突发场景下允许的最大 IOPS 数。突发持续时间设置是可配置的，默认为 1 分钟。如果卷的运行速度低于最大 IOPS 水平，则会累积突发积分。当性能水平变得非常高并被推动时，卷上允许出现超出最大 IOPS 的短时间突发 IOPS。

多租户

安全多租户通过以下功能实现：

- *安全认证。*质询握手身份验证协议 (CHAP) 用于安全卷访问。轻量级目录访问协议 (LDAP) 用于安全访问集群以进行管理和报告。
- 卷访问组 (VAG)。或者，可以使用 VAG 代替身份验证，将任意数量的 iSCSI 启动器特定的 iSCSI 限定名称 (IQN) 映射到一个或多个卷。要访问 VAG 中的卷，启动器的 IQN 必须位于该卷组允许的 IQN 列表中。
- 租户虚拟局域网 (VLAN)。在网络级别，通过使用 VLAN 实现 iSCSI 启动器和NetApp Element软件集群之间的端到端网络安全。对于为隔离工作负载或租户而创建的任何 VLAN， NetApp Element Software 都会创建一个单独的 iSCSI 目标 SVIP 地址，该地址只能通过特定的 VLAN 访问。
- *启用 VRF 的 VLAN。*为了进一步支持数据中心的安全性和可扩展性， NetApp Element软件允许您启用任何租户 VLAN 以实现类似 VRF 的功能。此功能增加了以下两个关键功能：
  - *L3 路由到租户 SVIP 地址。*此功能允许您将 iSCSI 启动器放置在与NetApp Element软件集群不同的网络或 VLAN 上。
  - *重叠或重复的 IP 子网。*此功能使您能够向租户环境添加模板，从而允许每个相应的租户 VLAN 分配来自同一 IP 子网的 IP 地址。此功能对于服务提供商环境中非常有用，因为 IP 空间的规模和保存非常重要。

企业存储效率

NetApp Element软件集群提高了整体存储效率和性能。以下功能以内联方式执行，始终处于开启状态，并且不需要用户手动配置：

- *重复数据删除*系统仅存储唯一的 4K 块。任何重复的 4K 块都会自动与已存储的数据版本相关联。数据位于块驱动器上，并使用NetApp Element软件 Helix 数据保护进行镜像。该系统大大减少了容量消耗和系统内的写入操作。
- *压缩。*在数据写入NVRAM之前，压缩是内联执行的。数据被压缩，存储在 4K 块中，并在系统中保持压缩状态。这种压缩显著减少了整个集群的容量消耗、写入操作和带宽消耗。
- *精简配置。*此功能可在您需要时提供适量的存储，从而消除因卷过度配置或卷利用不足而导致的容量消耗。
- *螺旋。*单个卷的元数据存储在元数据驱动器上，并复制到辅助元数据驱动器以实现冗余。

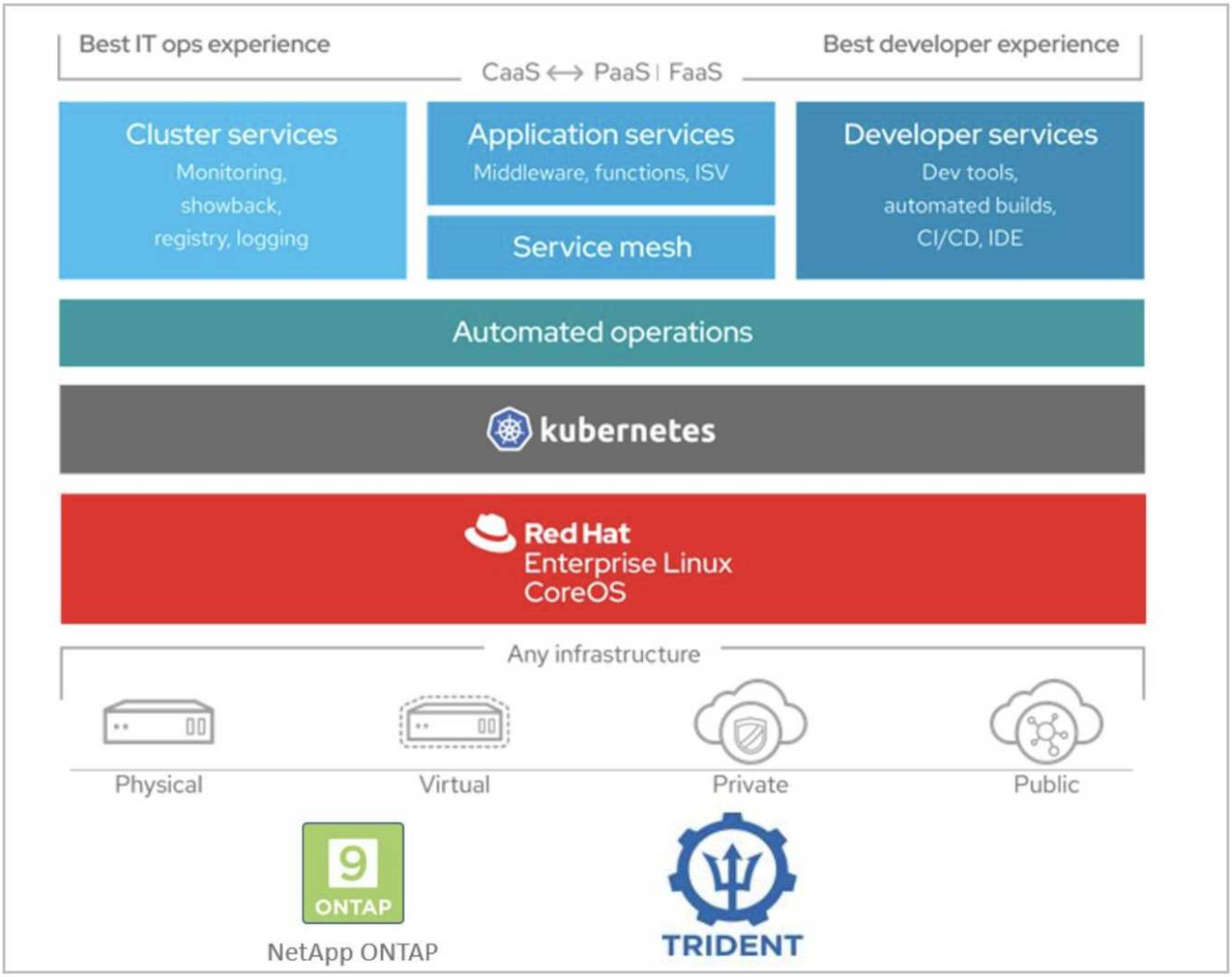
Element 是为自动化而设计的。所有存储功能均可通过 API 获得。这些 API 是 UI 用来控制系统的唯一方法。

NetApp存储集成

了解NetApp Trident与 Red Hat OpenShift 的集成

了解NetApp Trident保护，该保护已通过 OpenShift 虚拟化解决方案的应用程序和持久存储管理验证。

Trident是由NetApp维护的开源存储配置器和编排器， NetApp Trident保护可帮助您在基于容器的环境（例如 Red Hat OpenShift）中编排和管理持久数据。



以下页面包含有关NetApp产品的更多信息，这些产品已在 Red Hat OpenShift with NetApp解决方案中经过应用程序和持久存储管理验证：

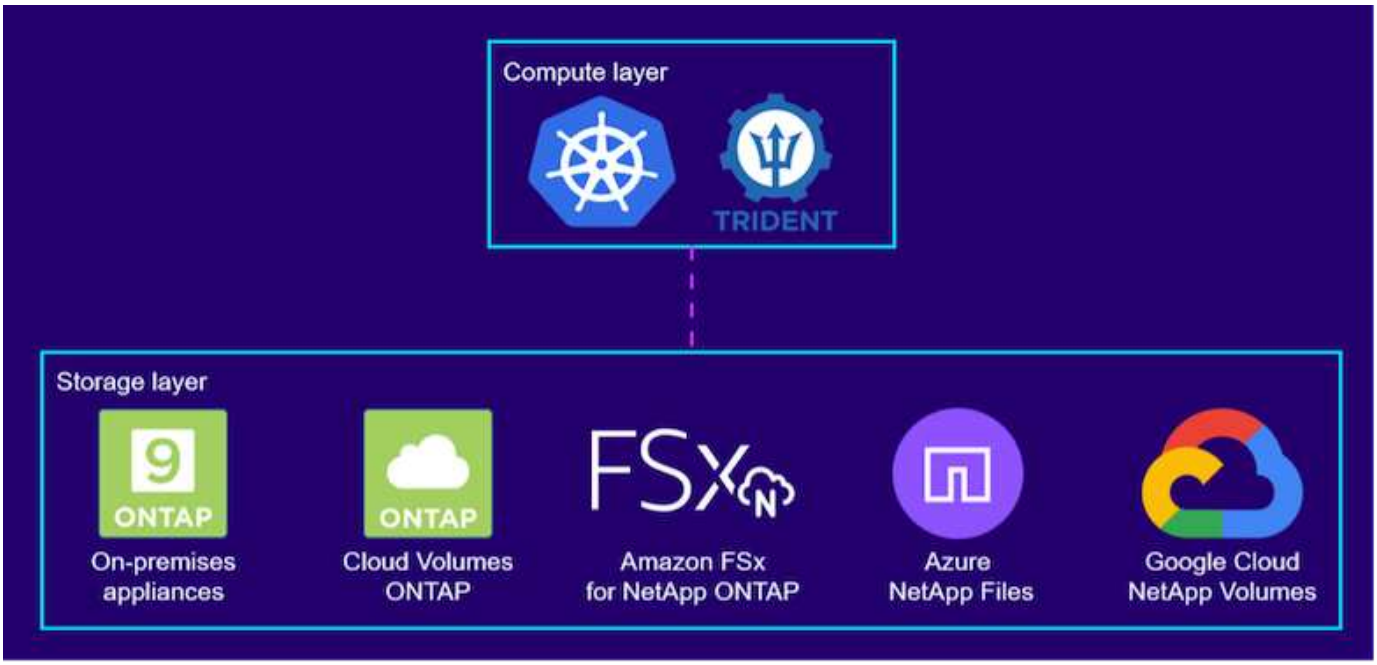
- "Trident文档"
- "Trident保护文档"

NetApp Trident

Trident概述

Trident是一个开源且完全受支持的容器和 Kubernetes 发行版（包括 Red Hat OpenShift）存储编排器。 Trident可与整个NetApp存储产品组合配合使用，包括NetApp ONTAP和 Element 存储系统，并且还支持 NFS 和 iSCSI 连接。 Trident允许最终用户从其NetApp存储系统配置和管理存储，而无需存储管理员的干预，从而加速 DevOps 工作流程。

管理员可以根据项目需求和存储系统模型配置多个存储后端，以实现高级存储功能，包括压缩、特定磁盘类型或保证一定性能水平的 QoS 级别。定义完成后，开发人员可以在他们的项目中使用这些后端来创建持久卷声明 (PVC) 并根据需要将持久存储附加到他们的容器。



Trident的开发周期很快，和 Kubernetes 一样，每年发布四次。

已测试过哪个版本的Trident以及可以找到哪个 Kubernetes 发行版的支持矩阵 "此处"。

请参阅"Trident产品文档"有关安装和配置的详细信息。

下载Trident

要在已部署的用户集群上安装Trident并配置持久卷，请完成以下步骤：

1. 将安装档案下载到管理工作站并提取内容。当前版本的Trident可以下载 "此处"。
2. 从下载的软件包中提取Trident安装。

```
[netapp-user@rhel7 ~]$ tar -xzf trident-installer-22.01.0.tar.gz
[netapp-user@rhel7 ~]$ cd trident-installer/
[netapp-user@rhel7 trident-installer]$
```

使用 **Helm** 安装**Trident Operator**

1. 首先设置用户集群的 `kubeconfig`文件作为环境变量，这样您就不必引用它，因为Trident没有选项来传递此文件。

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. 运行 Helm 命令从 helm 目录中的 tarball 安装Trident操作符，同时在用户集群中创建 trident 命名空间。

```
[netapp-user@rhel7 trident-installer]$ helm install trident
helm/trident-operator-22.01.0.tgz --create-namespace --namespace trident
NAME: trident
LAST DEPLOYED: Fri May  7 12:54:25 2021
NAMESPACE: trident
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing trident-operator, which will deploy and manage
NetApp's Trident CSI
storage provisioner for Kubernetes.

Your release is named 'trident' and is installed into the 'trident'
namespace.
Please note that there must be only one instance of Trident (and
trident-operator) in a Kubernetes cluster.

To configure Trident to manage storage resources, you will need a copy
of tridentctl, which is
available in pre-packaged Trident releases.  You may find all Trident
releases and source code
online at https://github.com/NetApp/trident.

To learn more about the release, try:

  $ helm status trident
  $ helm get all trident
```

3. 您可以通过检查命名空间中运行的 pod 或使用 tridentctl 二进制文件检查已安装的版本来验证Trident是否已成功安装。

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-5z45l                   1/2     Running   2          30s
trident-csi-696b685cf8-htdb2        6/6     Running   0          30s
trident-csi-b74p2                   2/2     Running   0          30s
trident-csi-lrw4n                   2/2     Running   0          30s
trident-operator-7c748d957-gr2gw    1/1     Running   0          36s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0        | 22.01.0        |
+----------------+----------------+
```

在某些情况下，客户环境可能需要定制Trident部署。在这些情况下，还可以手动安装Trident操作符并更新包含的清单以定制部署。

手动安装**Trident Operator**

1. 首先，设置用户集群的 `kubeconfig`文件作为环境变量，这样您就不必引用它，因为Trident没有选项来传递此文件。

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. 这 `trident-installer`目录包含定义所有必需资源的清单。使用适当的清单，创建 `TridentOrchestrator`自定义资源定义。

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
customresourcedefinition.apiextensions.k8s.io/tridentorchestrators.tride
nt.netapp.io created
```

3. 如果不存在，请使用提供的清单在集群中创建一个Trident命名空间。

```
[netapp-user@rhel7 trident-installer]$ oc apply -f deploy/namespace.yaml
namespace/trident created
```

4. 创建Trident操作员部署所需的资源，例如 `ServiceAccount`对于操作员来说， `ClusterRole`和 `ClusterRoleBinding`到 `ServiceAccount`，一个专门的 `PodSecurityPolicy`或操作员本身。

```
[netapp-user@rhel7 trident-installer]$ oc create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created
```

5. 部署操作员后，您可以使用以下命令检查操作员的状态：

```
[netapp-user@rhel7 trident-installer]$ oc get deployment -n trident
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator   1/1     1            1           23s
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-operator-66f48895cc-lzczk   1/1     Running   0          41s
```

6. 部署操作员后，我们现在可以使用它来安装Trident。这需要创建一个 `TridentOrchestrator`。

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
[netapp-user@rhel7 trident-installer]$ oc describe torc trident
Name:         trident
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
Metadata:
  Creation Timestamp:  2021-05-07T17:00:28Z
  Generation:          1
  Managed Fields:
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:spec:
        .:
        f:debug:
        f:namespace:
    Manager:      kubectl-create
    Operation:    Update
    Time:         2021-05-07T17:00:28Z
    API Version:  trident.netapp.io/v1
```

```
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        .:
        f:currentInstallationParams:
          .:
          f:IPv6:
          f:autosupportHostname:
          f:autosupportimage:
          f:autosupportProxy:
          f:autosupportSerialNumber:
          f:debug:
          f:enableNodePrep:
          f:imagePullSecrets:
          f:imageRegistry:
          f:k8sTimeout:
          f:kubeletDir:
          f:logFormat:
          f:silenceAutosupport:
          f:tridentimage:
        f:message:
        f:namespace:
        f:status:
        f:version:
    Manager:         trident-operator
    Operation:       Update
    Time:            2021-05-07T17:00:28Z
  Resource Version:  931421
  Self Link:
/apis/trident.netapp.io/v1/tridentorchestrators/trident
  UID:               8a26a7a6-dde8-4d55-9b66-a7126754d81f
Spec:
  Debug:      true
  Namespace:  trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport image:         netapp/trident-autosupport:21.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
```

```
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident image:        netapp/trident:22.01.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v22.01.0
Events:
  Type    Reason      Age   From                           Message
  ----    ------      ----  ----                           -------
  Normal  Installing  80s   trident-operator.netapp.io  Installing
Trident
  Normal  Installed   68s   trident-operator.netapp.io  Trident
installed
```

7. 您可以通过检查命名空间中运行的 pod 或使用 tridentctl 二进制文件检查已安装的版本来验证Trident是否已成功安装。

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-bb64c6cb4-lmd6h         6/6     Running   0          82s
trident-csi-gn59q                   2/2     Running   0          82s
trident-csi-m4szj                   2/2     Running   0          82s
trident-csi-sb9k9                   2/2     Running   0          82s
trident-operator-66f48895cc-lzczk   1/1     Running   0          2m39s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0          | 22.01.0          |
+----------------+----------------+
```

准备工作节点以进行存储

NFS

大多数 Kubernetes 发行版都附带默认安装的用于挂载 NFS 后端的软件包和实用程序，包括 Red Hat OpenShift。

但是，对于 NFSv3，没有在客户端和服务器之间协商并发的机制。因此，必须手动将客户端 sunrpc 插槽表条目的最大数量与服务器上支持的值同步，以确保 NFS 连接的最佳性能，而无需服务器减小连接的窗口大小。

对于ONTAP，支持的 sunrpc 插槽表条目的最大数量为 128，即ONTAP一次可以处理 128 个并发 NFS 请求。但是，默认情况下，Red Hat CoreOS/Red Hat Enterprise Linux 每个连接最多有 65,536 个 sunrpc 插槽表条目。

我们需要将此值设置为 128，这可以使用 OpenShift 中的 Machine Config Operator (MCO) 来完成。

要修改 OpenShift 工作节点中的最大 sunrpc 插槽表条目数，请完成以下步骤：

1. 登录 OCP 网络控制台并导航至 Compute > Machine Configs。单击创建机器配置。复制并粘贴 YAML 文件，然后单击“创建”。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 98-worker-nfs-rpc-slot-tables
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - contents:
            source: data:text/plain;charset=utf-
8;base64,b3B0aW9ucyBzdW5ycGMgdGNwX21heF9zbG90X3RhYmxlX2VudHJpZXM9MTI4Cg=
=
          filesystem: root
          mode: 420
          path: /etc/modprobe.d/sunrpc.conf
```

2. 创建 MCO 后，需要在所有工作节点上应用配置并逐个重新启动。整个过程大约需要20到30分钟。使用以下命令验证机器配置是否已应用 `oc get mcp`并确保工人的机器配置池已更新。

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                             UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False
```

iSCSI

要准备工作节点以允许通过 iSCSI 协议映射块存储卷，您必须安装必要的软件包来支持该功能。

在 Red Hat OpenShift 中，这是通过在部署集群后将 MCO（机器配置操作员）应用于集群来处理的。

要配置工作节点以运行 iSCSI 服务，请完成以下步骤：

1. 登录 OCP 网络控制台并导航至 Compute > Machine Configs。单击创建机器配置。复制并粘贴 YAML 文件，然后单击“创建”。

   不使用多路径时：

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 99-worker-element-iscsi
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
  osImageURL: ""
```

   使用多路径时：

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 99-worker-ontap-iscsi
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgbm8KICAgICA
gICBmaW5kX211bHRpcGF0aHMgbm8KfQoKYmxhY2tsaXN0X2V4Y2VwdGlvbnMgewogICAgICA
gIHByb3BlcnR5ICIoU0NTSV9JREVOVF98SURfV1dOKSIKfQoKYmxhY2tsaXN0IHsKfQoK
          verification: {}
        filesystem: root
        mode: 400
        path: /etc/multipath.conf
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
        - name: multipathd.service
          enabled: true
          state: started
  osImageURL: ""
```

2. 配置创建后，大约需要 20 到 30 分钟将配置应用到工作节点并重新加载它们。使用以下命令验证机器配置是否已应用 `oc get mcp`并确保工人的机器配置池已更新。您还可以登录工作节点来确认 iscsid 服务正在运行（如果使用多路径，则 multipathd 服务正在运行）。

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                         UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status iscsid
● iscsid.service - Open-iSCSI
   Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled;
vendor preset: disabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
     Docs: man:iscsid(8)
           man:iscsiadm(8)
 Main PID: 1242 (iscsid)
   Status: "Ready to process requests"
    Tasks: 1
   Memory: 4.9M
      CPU: 9ms
   CGroup: /system.slice/iscsid.service
           └─1242 /usr/sbin/iscsid -f

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status multipathd
 ● multipathd.service - Device-Mapper Multipath Device Controller
   Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled;
vendor preset: enabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
  Main PID: 918 (multipathd)
    Status: "up"
    Tasks: 7
    Memory: 13.7M
    CPU: 57ms
    CGroup: /system.slice/multipathd.service
            └─918 /sbin/multipathd -d -s
```

还可以通过运行以下命令来确认 MachineConfig 已成功应用且服务已按预期启动 `oc debug` 带有适当标志的命令。

创建存储系统后端

完成Trident Operator 安装后，您必须为正在使用的特定NetApp存储平台配置后端。按照下面的链接继续设置和配置Trident。

- "NetApp ONTAP NFS"
- "NetApp ONTAP iSCSI"
- "NetApp Element iSCSI"

NetApp ONTAP NFS 配置

要实现Trident与NetApp ONTAP存储系统的集成，您必须创建一个能够与存储系统通信的后端。

1. 下载的安装档案中提供了示例后端文件 `sample-input`文件夹层次结构。对于服务于 NFS 的NetApp ONTAP系统，复制 `backend-ontap-nas.json`文件到您的工作目录并编辑该文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-nas/backend-ontap-nas.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-nas.json
```

2. 编辑此文件中的 backendName、managementLIF、dataLIF、svm、username 和 password 值。

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap-nas+10.61.181.221",
    "managementLIF": "172.21.224.201",
    "dataLIF": "10.61.181.221",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "password"
}
```

最佳做法是将自定义 backendName 值定义为 storageDriverName 和为 NFS 提供服务的 dataLIF 的组合，以便于识别。

3. 有了这个后端文件，运行以下命令来创建您的第一个后端。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-nas.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas+10.61.181.221 | ontap-nas      | be7a619d-c81d-445c-b80c-
5c87a73c5b1e | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. 创建后端后，接下来必须创建存储类。与后端一样，有一个示例存储类文件，可以根据 sample-inputs 文件夹中提供的环境进行编辑。将其复制到工作目录并进行必要的编辑以反映创建的后端。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 对此文件唯一需要做的编辑是定义 `backendType`将值设置为新创建的后端的存储驱动程序的名称。还要注意名称字段值，该值必须在后续步骤中引用。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
```

有一个可选字段称为 `fsType`这是在这个文件中定义的。可以在 NFS 后端删除此行。

6. 运行 `oc`命令来创建存储类。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. 创建存储类后，您必须创建第一个持久卷声明 (PVC)。有一个示例 `pvc-basic.yaml`也可用于执行位于 sample-inputs 中的此操作的文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. 对此文件唯一需要做的编辑是确保 `storageClassName`字段与刚刚创建的字段匹配。 PVC 定义可以根据要配置的工作负载的需要进一步定制。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. 通过发出 `oc`命令。创建可能需要一些时间，具体取决于所创建的备份卷的大小，因此您可以在创建完成时观察该过程。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-b4370d37-0fa4-4c17-bd86-94f96c94b42d   1Gi
RWO            basic-csi      7s
```

NetApp ONTAP iSCSI 配置

要实现Trident与NetApp ONTAP存储系统的集成，您必须创建一个能够与存储系统通信的后端。

1. 下载的安装档案中提供了示例后端文件 `sample-input`文件夹层次结构。对于服务于 iSCSI 的NetApp ONTAP系统，复制 `backend-ontap-san.json`文件到您的工作目录并编辑该文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-san/backend-ontap-san.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-san.json
```

2. 编辑此文件中的 managementLIF、dataLIF、svm、用户名和密码值。

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "172.21.224.201",
  "dataLIF": "10.61.181.240",
  "svm": "trident_svm",
  "username": "admin",
  "password": "password"
}
```

3. 有了这个后端文件，运行以下命令来创建您的第一个后端。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-san.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontapsan_10.61.181.241 | ontap-san      | 6788533c-7fea-4a35-b797-
fb9bb3322b91 | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. 创建后端后，接下来必须创建存储类。与后端一样，有一个示例存储类文件，可以根据 sample-inputs 文件夹中提供的环境进行编辑。将其复制到工作目录并进行必要的编辑以反映创建的后端。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 对此文件唯一需要做的编辑是定义 `backendType`将值设置为新创建的后端的存储驱动程序的名称。还要注意名称字段值，该值必须在后续步骤中引用。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
```

有一个可选字段称为 `fsType`这是在这个文件中定义的。在 iSCSI 后端，可以将此值设置为特定的 Linux 文件系统类型（XFS、ext4 等）或删除以允许 OpenShift 决定使用哪种文件系统。

6. 运行 `oc`命令来创建存储类。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. 创建存储类后，您必须创建第一个持久卷声明 (PVC)。有一个示例 `pvc-basic.yaml`也可用于执行位于 sample-inputs 中的此操作的文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. 对此文件唯一需要做的编辑是确保 `storageClassName`字段与刚刚创建的字段匹配。 PVC 定义可以根据要配置的工作负载的需要进一步定制。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. 通过发出 `oc`命令。创建可能需要一些时间，具体取决于所创建的备份卷的大小，因此您可以在创建完成时观察该过程。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-7ceac1ba-0189-43c7-8f98-094719f7956c   1Gi
RWO            basic-csi      3s
```

NetApp Element iSCSI 配置

要实现Trident与NetApp Element存储系统的集成，您必须创建一个后端，以便使用 iSCSI 协议与存储系统进行通信。

1. 下载的安装档案中提供了示例后端文件 `sample-input`文件夹层次结构。对于服务于 iSCSI 的NetApp Element系统，复制 `backend-solidfire.json`文件到您的工作目录并编辑该文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/solidfire/backend-solidfire.json ./
[netapp-user@rhel7 trident-installer]$ vi ./backend-solidfire.json
```

   a. 编辑用户、密码和 MVIP 值 `EndPoint`线。

   b. 编辑 `SVIP`价值。

```
 {
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://trident:password@172.21.224.150/json-
rpc/8.0",
    "SVIP": "10.61.180.200:3260",
    "TenantName": "trident",
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS":
2000, "burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS":
6000, "burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS":
8000, "burstIOPS": 10000}}]
}
```

2. 有了这个后端文件，运行以下命令来创建您的第一个后端。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-solidfire.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| solidfire_10.61.180.200 | solidfire-san  | b90783ee-e0c9-49af-8d26-
3ea87ce2efdf | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

3. 创建后端后，接下来必须创建存储类。与后端一样，有一个示例存储类文件，可以根据 sample-inputs 文件夹中提供的环境进行编辑。将其复制到工作目录并进行必要的编辑以反映创建的后端。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

4. 对此文件唯一需要做的编辑是定义 `backendType`将值设置为新创建的后端的存储驱动程序的名称。还要注意名称字段值，该值必须在后续步骤中引用。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "solidfire-san"
```

> 有一个可选字段称为 `fsType`这是在这个文件中定义的。在 iSCSI 后端，可以将此值设置为特定的 Linux 文件系统类型（XFS、ext4 等），或者可以删除它以允许 OpenShift 决定使用哪种文件系统。

5. 运行 `oc`命令来创建存储类。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

6. 创建存储类后，您必须创建第一个持久卷声明 (PVC)。有一个示例 `pvc-basic.yaml`也可用于执行位于

sample-inputs 中的此操作的文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

7. 对此文件唯一需要做的编辑是确保 `storageClassName`字段与刚刚创建的字段匹配。 PVC 定义可以根据要配置的工作负载的需要进一步定制。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

8. 通过发出 `oc`命令。创建可能需要一些时间，具体取决于所创建的备份卷的大小，因此您可以在创建完成时观察该过程。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-3445b5cc-df24-453d-a1e6-b484e874349d   1Gi
RWO            basic-csi      5s
```

高级配置选项

探索负载均衡器选项

探索负载均衡器选项：**Red Hat OpenShift** 与**NetApp**

大多数情况下，Red Hat OpenShift 通过路由让应用程序可供外界使用。通过为服务提供外部可访问的主机名来公开服务。 OpenShift 路由器可以使用定义的路由及其服务标识的端点来向外部客户端提供这种命名连接。

然而在某些情况下，应用程序需要部署和配置定制的负载均衡器来公开适当的服务。 NetApp Trident Protect 就是一个例子。为了满足这一需求，我们评估了许多自定义负载均衡器选项。本节介绍它们的安装和配置。

以下页面包含有关 Red Hat OpenShift 与NetApp解决方案中验证的负载均衡器选项的更多信息：

- "MetalLB"
- "F5 BIG-IP"

安装 **MetalLB** 负载均衡器：**Red Hat OpenShift** 与**NetApp**

本页列出了 MetalLB 负载均衡器的安装和配置说明。

MetalLB 是安装在 OpenShift 集群上的自托管网络负载均衡器，允许在不在云提供商上运行的集群中创建类型负载均衡器的 OpenShift 服务。 MetalLB 共同支撑 LoadBalancer 服务的两个主要特性是地址分配和对外通告。

MetalLB 配置选项

根据 MetalLB 如何宣布分配给 OpenShift 集群之外的 LoadBalancer 服务的 IP 地址，它以两种模式运行：

- *第 2 层模式。*在此模式下，OpenShift 集群中的一个节点拥有该服务的所有权，并响应该 IP 的 ARP 请求，以使其在 OpenShift 集群外部可访问。由于只有节点通告 IP，因此存在带宽瓶颈和故障转移速度慢的限制。有关详细信息，请参阅文档"此处"。
- *BGP 模式。*在此模式下，OpenShift 集群中的所有节点都与路由器建立 BGP 对等会话，并通告路由以将流量转发到服务 IP。实现此目的的先决条件是将 MetalLB 与该网络中的路由器集成。由于BGP中的哈希机制，当服务的IP到节点映射发生变化时，会受到一定的限制。欲了解更多信息，请参阅文档"此处"。

> 为了本文档的目的，我们在第 2 层模式下配置 MetalLB。

安装 **MetalLB** 负载均衡器

1. 下载 MetalLB 资源。

```
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/name
space.yaml
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/meta
llb.yaml
```

2. 编辑文件 `metallb.yaml`并删除 `spec.template.spec.securityContext`来自控制器部署和扬声器 DaemonSet。

   要删除的行：

```
securityContext:
  runAsNonRoot: true
  runAsUser: 65534
```

3. 创建 `metallb-system`命名空间。

```
[netapp-user@rhel7 ~]$ oc create -f namespace.yaml
namespace/metallb-system created
```

4. 创建 MetalLB CR。

```
[netapp-user@rhel7 ~]$ oc create -f metallb.yaml
podsecuritypolicy.policy/controller created
podsecuritypolicy.policy/speaker created
serviceaccount/controller created
serviceaccount/speaker created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
role.rbac.authorization.k8s.io/config-watcher created
role.rbac.authorization.k8s.io/pod-lister created
role.rbac.authorization.k8s.io/controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller
created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker
created
rolebinding.rbac.authorization.k8s.io/config-watcher created
rolebinding.rbac.authorization.k8s.io/pod-lister created
rolebinding.rbac.authorization.k8s.io/controller created
daemonset.apps/speaker created
deployment.apps/controller created
```

5. 在配置 MetalLB 扬声器之前，请授予扬声器 DaemonSet 提升的权限，以便它可以执行使负载均衡器工作所需的网络配置。

```
[netapp-user@rhel7 ~]$ oc adm policy add-scc-to-user privileged -n
metallb-system -z speaker
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:privileged
added: "speaker"
```

6. 通过创建配置 MetalLB `ConfigMap`在 `metallb-system`命名空间。

```
[netapp-user@rhel7 ~]$ vim metallb-config.yaml

apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 10.63.17.10-10.63.17.200

[netapp-user@rhel7 ~]$ oc create -f metallb-config.yaml
configmap/config created
```

7. 现在，当创建负载均衡器服务时，MetalLB 会为服务分配一个 externalIP，并通过响应 ARP 请求来通告该 IP 地址。

> 如果您希望在 BGP 模式下配置 MetalLB，请跳过上面的第 6 步并按照 MetalLB 文档中的步骤进行操作"此处"。

安装 **F5 BIG-IP** 负载均衡器

F5 BIG-IP 是一种应用交付控制器 (ADC)，它提供广泛的高级生产级流量管理和安全服务，如 L4-L7 负载平衡、SSL/TLS 卸载、DNS、防火墙等。这些服务大大提高了应用程序的可用性、安全性和性能。

F5 BIG-IP 可以以多种方式部署和使用，可以在专用硬件上、在云端或作为本地虚拟设备。请参阅此处的文档，根据要求探索和部署 F5 BIG-IP。

为了将 F5 BIG-IP 服务与 Red Hat OpenShift 有效集成，F5 提供了 BIG-IP 容器入口服务 (CIS)。 CIS 作为控制器容器安装，用于监视 OpenShift API 中的某些自定义资源定义 (CRD) 并管理 F5 BIG-IP 系统配置。 F5 BIG-IP CIS 可以配置为控制 OpenShift 中的服务类型 LoadBalancers 和 Routes。

此外，为了自动分配 IP 地址来为 LoadBalancer 类型提供服务，您可以利用 F5 IPAM 控制器。 F5 IPAM 控制器作为控制器 pod 安装，它使用 ipamLabel 注释监视 LoadBalancer 服务的 OpenShift API，以从预配置池中分配 IP 地址。

本页列出了 F5 BIG-IP CIS 和 IPAM 控制器的安装和配置说明。作为先决条件，您必须部署并获得许可的 F5 BIG-IP 系统。它还必须获得 SDN 服务的许可，该服务默认包含在 BIG-IP VE 基本许可证中。

> F5 BIG-IP 可以以独立或集群模式部署。为了进行此验证，F5 BIG-IP 以独立模式部署，但出于生产目的，最好使用 BIG-IP 集群以避免单点故障。

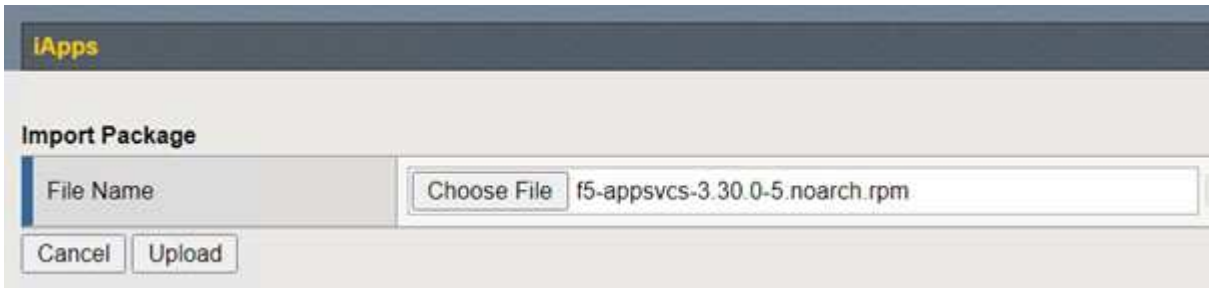
F5 BIG-IP 系统可以部署在专用硬件上、云端或作为本地虚拟设备部署，版本高于 12.x，以便与 F5 CIS 集成。为了本文档的目的，F5 BIG-IP 系统被验证为虚拟设备，例如使用 BIG-IP VE 版本。

已验证版本

技术	软件版本
红帽 OpenShift	4.6 EUS，4.7
F5 BIG-IP VE 版本	16.1.0
F5 容器入口服务	2.5.1
F5 IPAM 控制器	0.1.4
F5 AS3	3.30.0

安装

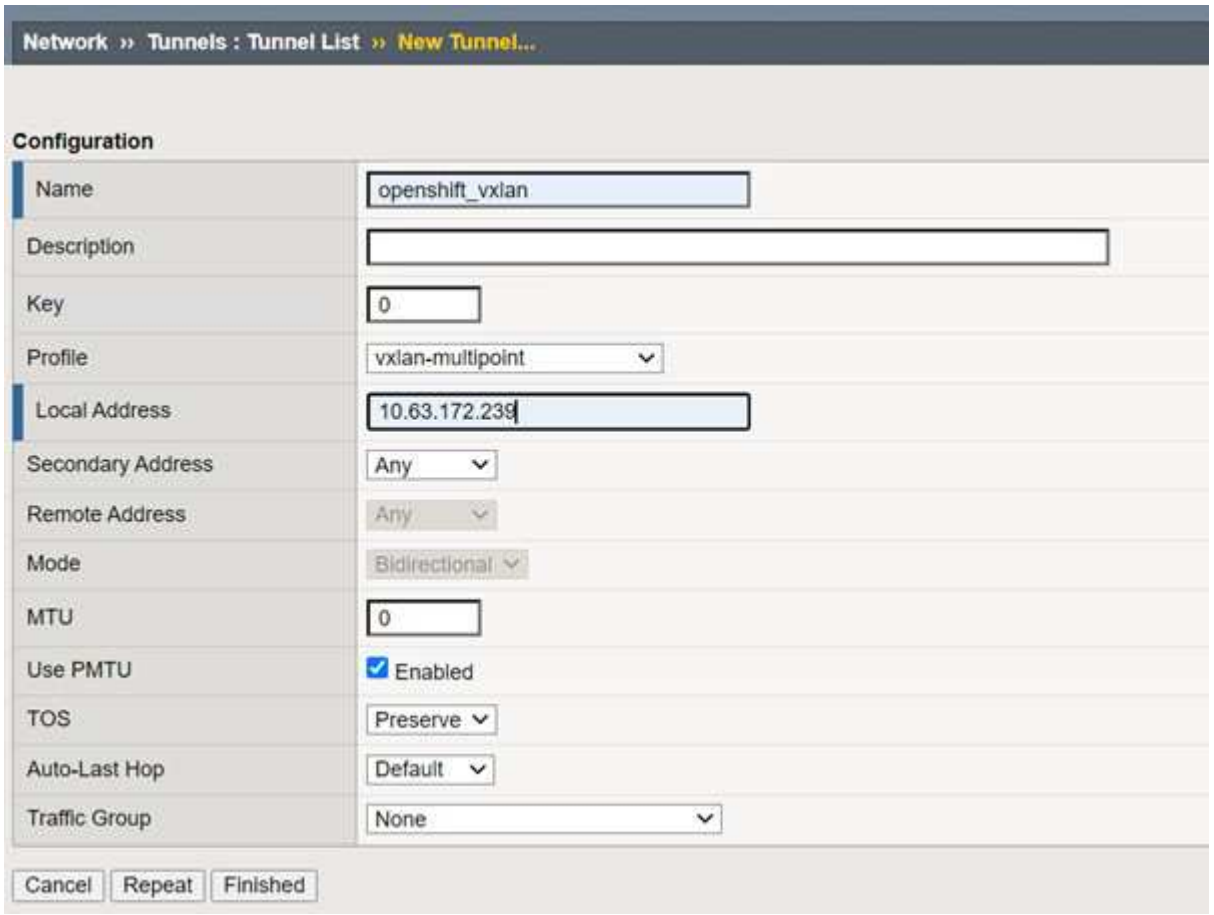
1. 安装 F5 应用服务 3 扩展以允许 BIG-IP 系统接受 JSON 中的配置而不是命令。前往 "F5 AS3 GitHub 存储库"，并下载最新的RPM文件。
2. 登录 F5 BIG-IP 系统，导航到 iApps > Package Management LX 并单击 Import。
3. 单击“选择文件”并选择下载的 AS3 RPM 文件，单击“确定”，然后单击“上传”。



4. 确认 AS3 扩展已成功安装。

5. 接下来配置OpenShift和BIG-IP系统之间通信所需的资源。首先通过在 BIG-IP 系统上为 OpenShift SDN 创建 VXLAN 隧道接口来在 OpenShift 和 BIG-IP 服务器之间创建隧道。导航到网络 > 隧道 > 配置文件，单击创建，然后将父配置文件设置为 vxlan，将泛洪类型设置为多播。输入配置文件的名称并单击“完成”。

6. 导航到网络 > 隧道 > 隧道列表，单击创建，然后输入隧道的名称和本地 IP 地址。选择上一步创建的隧道配置文件，然后单击“完成”。



7. 使用集群管理员权限登录 Red Hat OpenShift 集群。
8. 在 OpenShift 上为 F5 BIG-IP 服务器创建一个 hostsubnet，将子网从 OpenShift 集群扩展到 F5 BIG-IP 服务器。下载主机子网 YAML 定义。

```
wget https://github.com/F5Networks/k8s-bigip-
ctlr/blob/master/docs/config_examples/openshift/f5-kctlr-openshift-
hostsubnet.yaml
```

9. 编辑主机子网文件并为 OpenShift SDN 添加 BIG-IP VTEP（VXLAN 隧道）IP。

```
apiVersion: v1
kind: HostSubnet
metadata:
  name: f5-server
  annotations:
    pod.network.openshift.io/fixed-vnid-host: "0"
    pod.network.openshift.io/assign-subnet: "true"
# provide a name for the node that will serve as BIG-IP's entry into the
cluster
host: f5-server
# The hostIP address will be the BIG-IP interface address routable to
the
# OpenShift Origin nodes.
# This address is the BIG-IP VTEP in the SDN's VXLAN.
hostIP: 10.63.172.239
```

根据您环境的需要更改 hostIP 和其他详细信息。

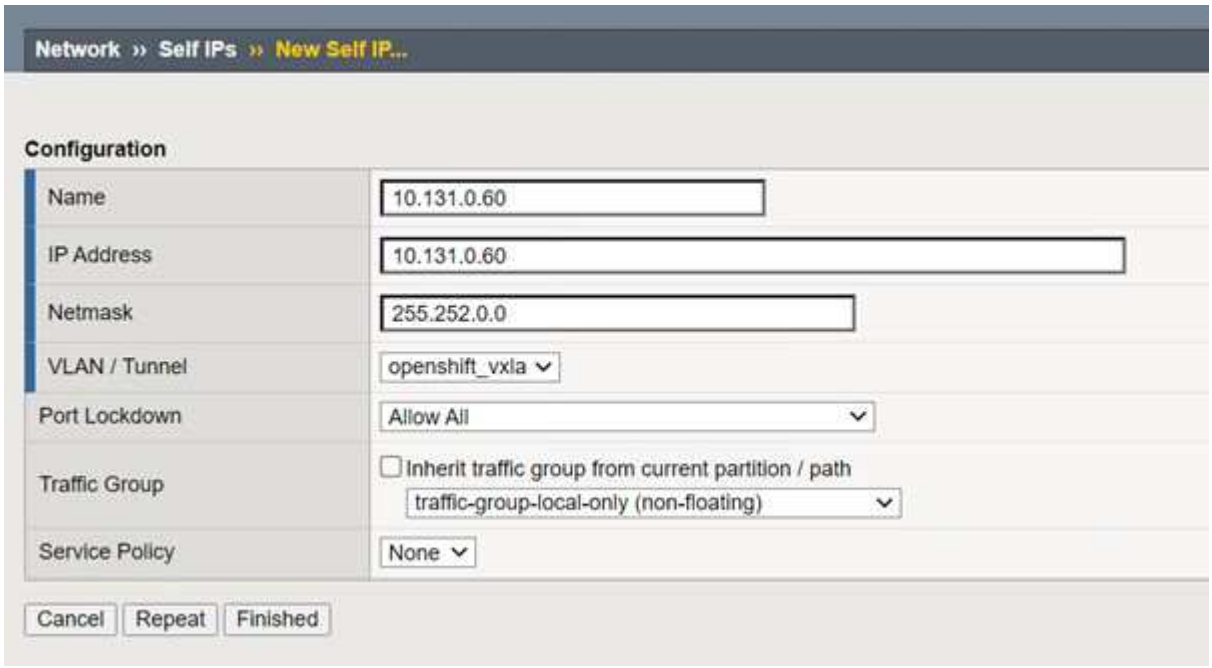
10. 创建 HostSubnet 资源。

```
[admin@rhel-7 ~]$ oc create -f f5-kctlr-openshift-hostsubnet.yaml

hostsubnet.network.openshift.io/f5-server created
```

11. 获取为 F5 BIG-IP 服务器创建的主机子网的集群 IP 子网范围。

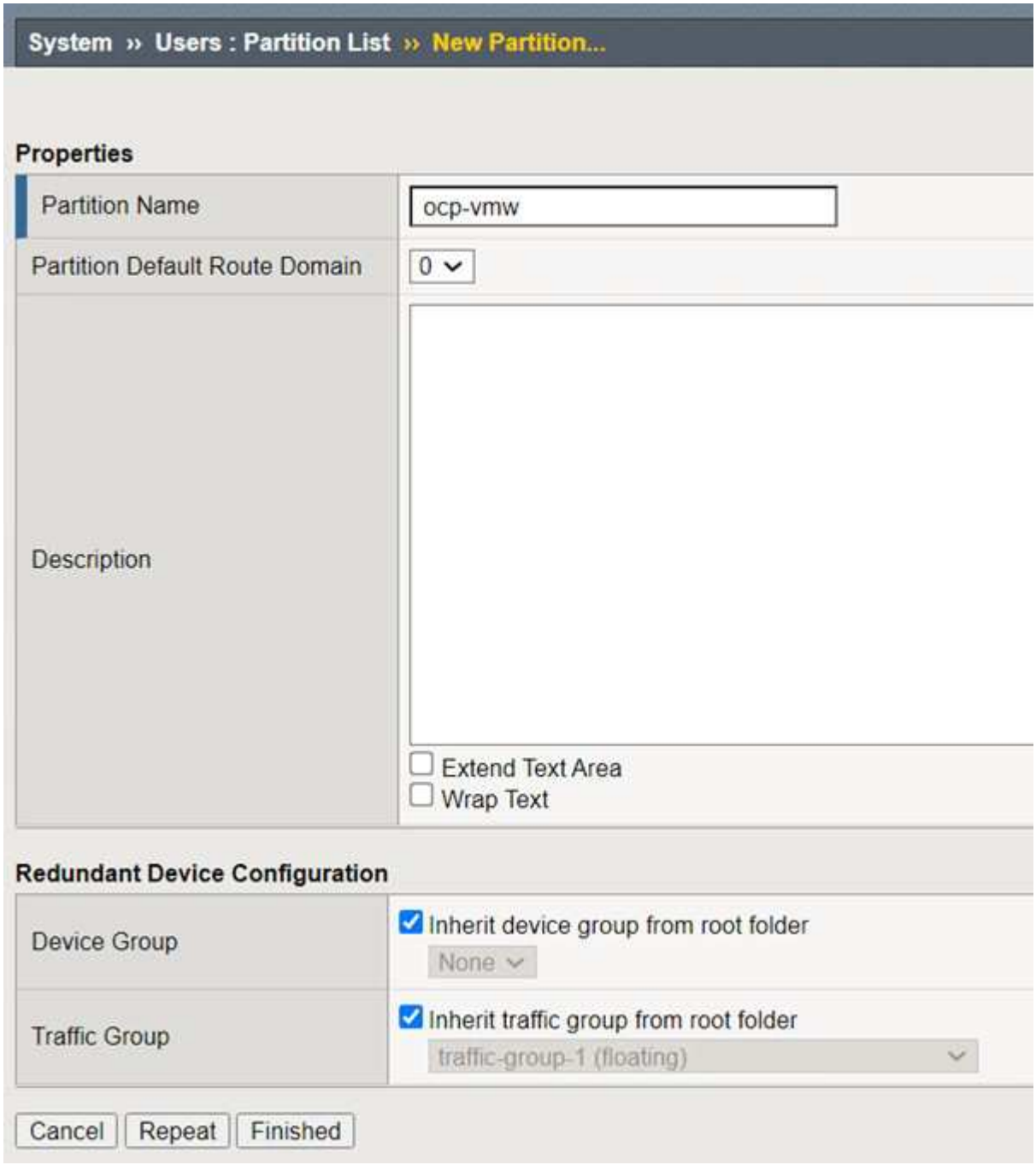
```
[admin@rhel-7 ~]$ oc get hostsubnet

NAME                           HOST                           HOST IP
SUBNET          EGRESS CIDRS   EGRESS IPS
f5-server                      f5-server                      10.63.172.239
10.131.0.0/23
ocp-vmw-nszws-master-0         ocp-vmw-nszws-master-0         10.63.172.44
10.128.0.0/23
ocp-vmw-nszws-master-1         ocp-vmw-nszws-master-1         10.63.172.47
10.130.0.0/23
ocp-vmw-nszws-master-2         ocp-vmw-nszws-master-2         10.63.172.48
10.129.0.0/23
ocp-vmw-nszws-worker-r8fh4     ocp-vmw-nszws-worker-r8fh4     10.63.172.7
10.130.2.0/23
ocp-vmw-nszws-worker-tvr46     ocp-vmw-nszws-worker-tvr46     10.63.172.11
10.129.2.0/23
ocp-vmw-nszws-worker-wdxhg     ocp-vmw-nszws-worker-wdxhg     10.63.172.24
10.128.2.0/23
ocp-vmw-nszws-worker-wg8r4     ocp-vmw-nszws-worker-wg8r4     10.63.172.15
10.131.2.0/23
ocp-vmw-nszws-worker-wtgfw     ocp-vmw-nszws-worker-wtgfw     10.63.172.17
10.128.4.0/23
```

12. 在 OpenShift VXLAN 上创建一个自身 IP，其 IP 位于与 F5 BIG-IP 服务器对应的 OpenShift 主机子网范围内。登录 F5 BIG-IP 系统，导航至网络 > 自有 IP，然后单击创建。输入为 F5 BIG-IP 主机子网创建的集群 IP 子网中的 IP，选择 VXLAN 隧道，然后输入其他详细信息。然后单击“完成”。



13. 在 F5 BIG-IP 系统中创建一个分区，以便与 CIS 一起配置和使用。导航到系统 > 用户 > 分区列表，单击创

建，然后输入详细信息。然后单击“完成”。



F5 建议不要对 CIS 管理的分区进行手动配置。

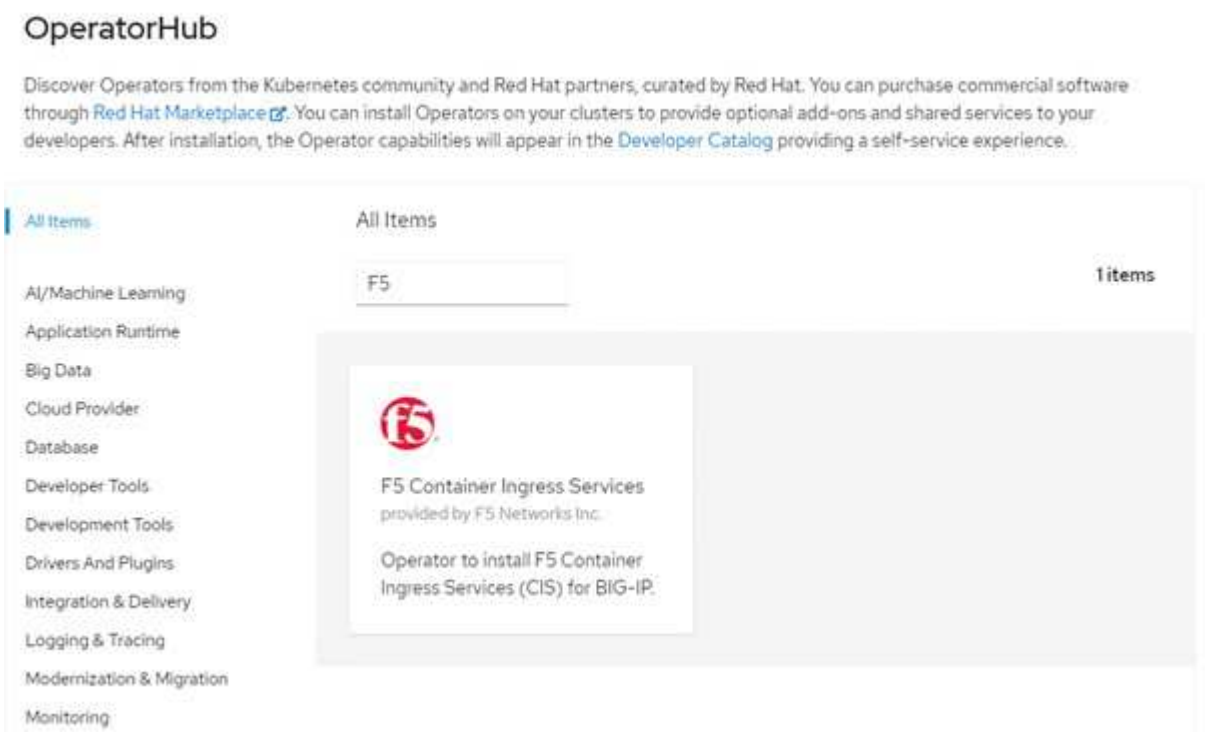
14. 使用来自 OperatorHub 的操作员安装 F5 BIG-IP CIS。使用 cluster-admin 权限登录 Red Hat OpenShift 集群，并使用 F5 BIG-IP 系统登录凭证创建一个 secret，这是操作员的先决条件。

```
[admin@rhel-7 ~]$ oc create secret generic bigip-login -n kube-system
--from-literal=username=admin --from-literal=password=admin

secret/bigip-login created
```

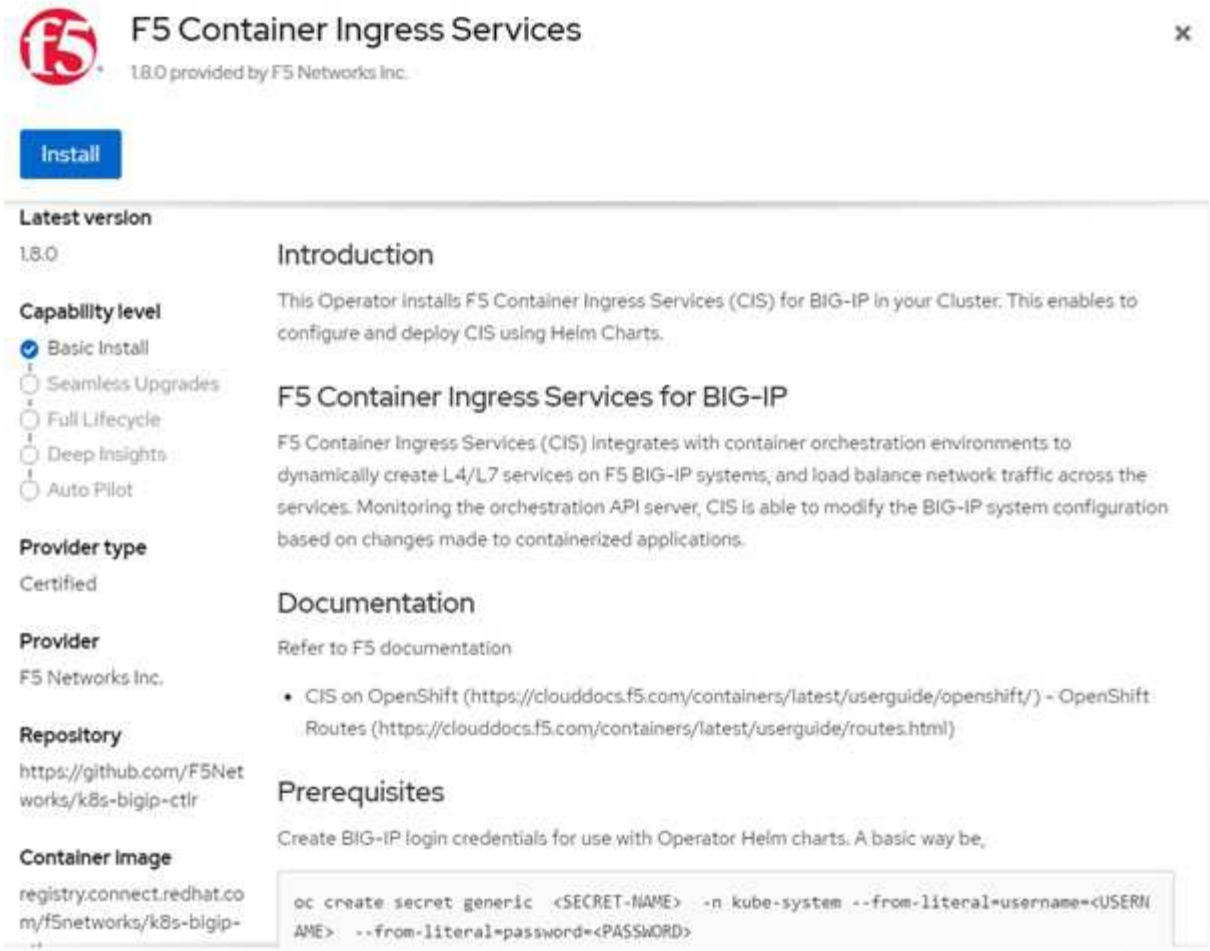
15. 安装 F5 CIS CRD。

```
[admin@rhel-7 ~]$ oc apply -f
https://raw.githubusercontent.com/F5Networks/k8s-bigip-
ctlr/master/docs/config_examples/crd/Install/customresourcedefinitions.y
ml

customresourcedefinition.apiextensions.k8s.io/virtualservers.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/tlsprofiles.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/transportservers.cis.f5.co
m created
customresourcedefinition.apiextensions.k8s.io/externaldnss.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/ingresslinks.cis.f5.com
created
```

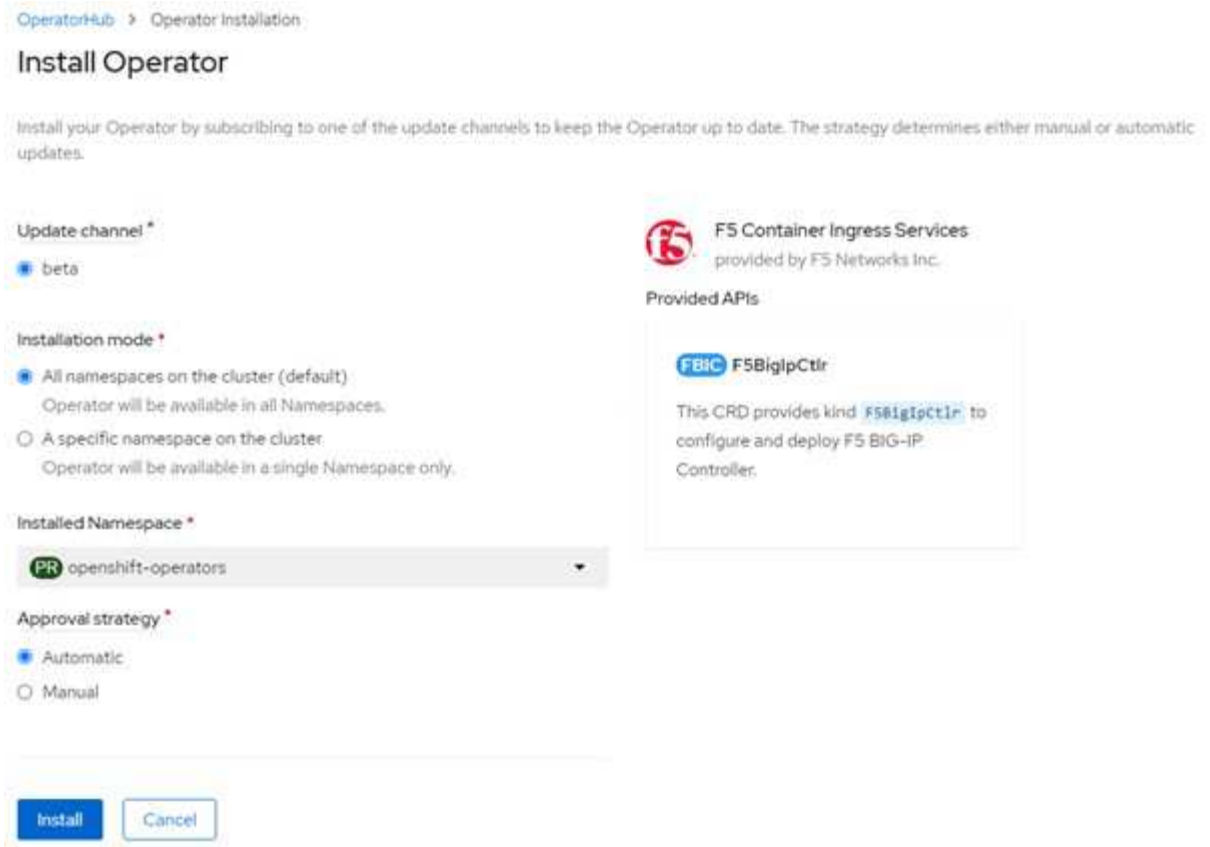
16. 导航到 Operators > OperatorHub，搜索关键字 F5，然后单击 F5 Container Ingress Service 磁贴。



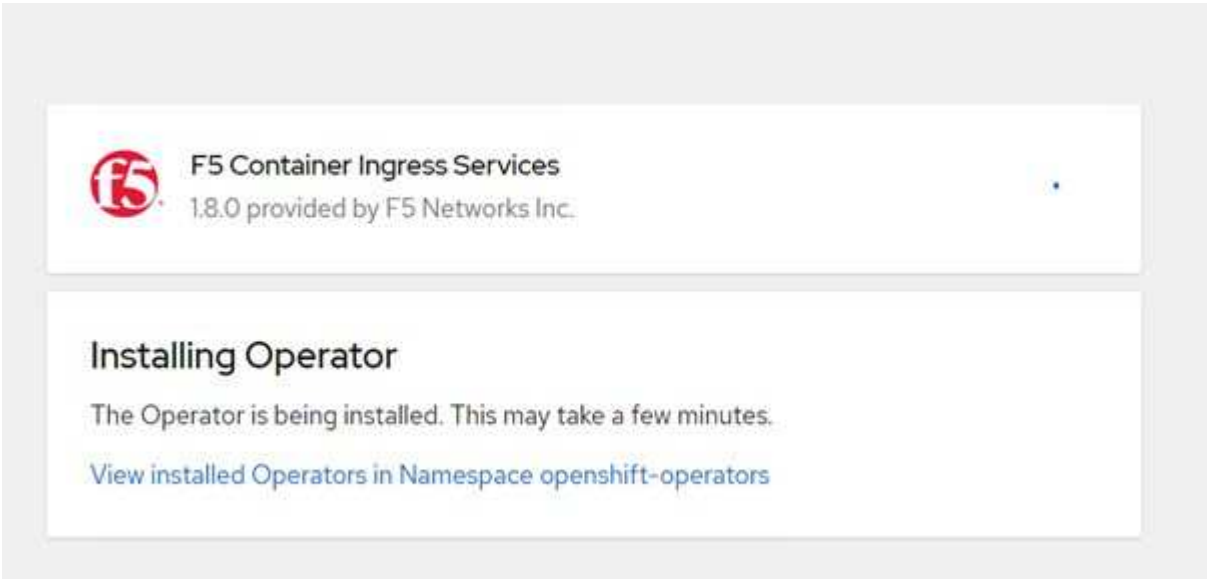
17. 阅读操作员信息并单击安装。



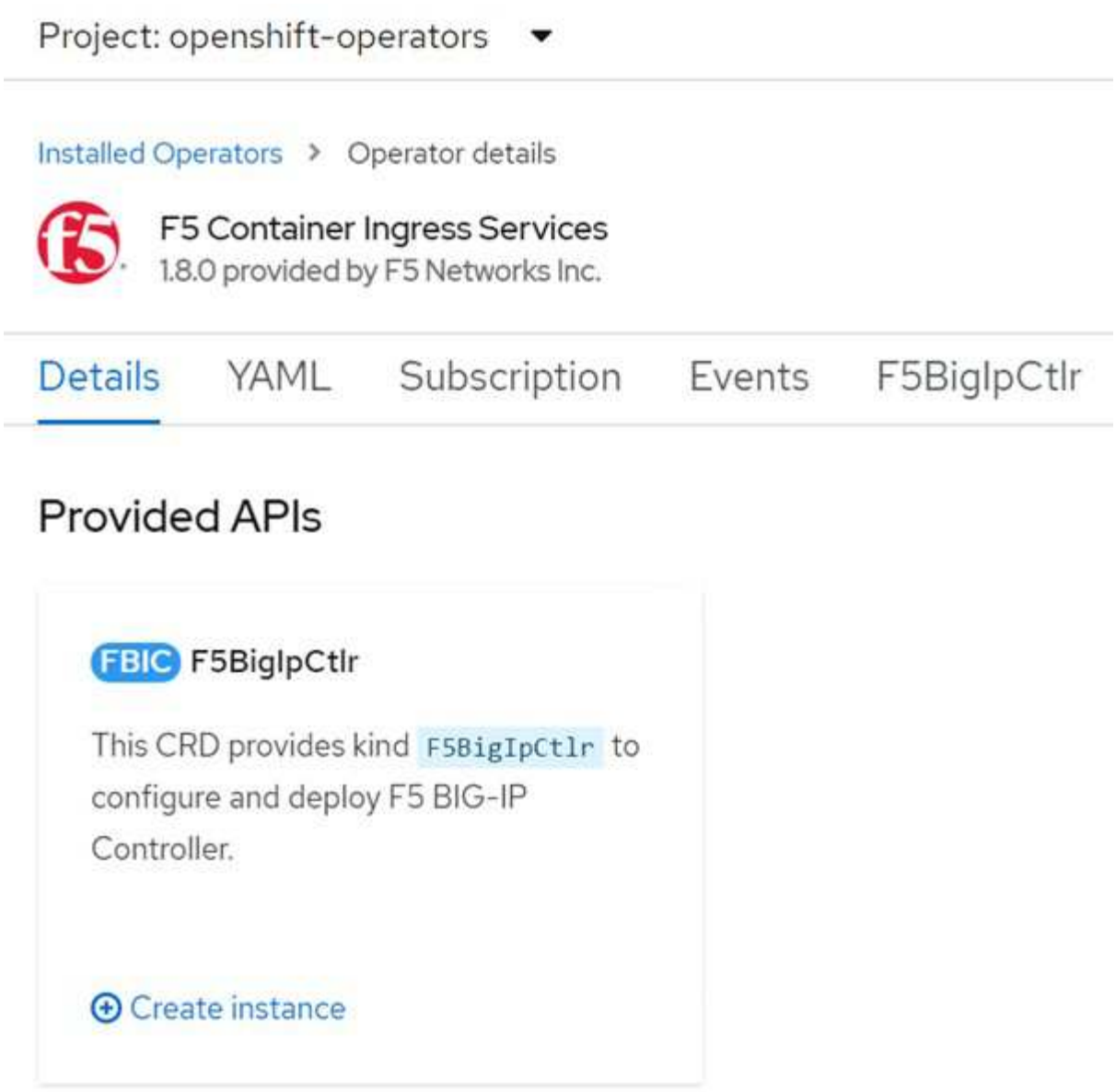
18. 在安装操作员屏幕上，保留所有默认参数，然后单击安装。



19. 安装操作员需要一段时间。



20. 操作员安装完成后，将显示“安装成功”消息。
21. 导航到 Operators > Installed Operators，单击 F5 Container Ingress Service，然后单击 F5BigIpCtlr 图块下的 Create Instance。



22. 单击 YAML View，更新必要的参数后粘贴以下内容。

更新参数 `bigip_partition`，` openshift_sdn_name`， `bigip_url`和 `bigip_login_secret` 在复制内容之前，请先查看以下设置的值。

```
apiVersion: cis.f5.com/v1
kind: F5BigIpCtlr
metadata:
  name: f5-server
  namespace: openshift-operators
spec:
  args:
    log_as3_response: true
    agent: as3
    log_level: DEBUG
    bigip_partition: ocp-vmw
    openshift_sdn_name: /Common/openshift_vxlan
    bigip_url: 10.61.181.19
    insecure: true
    pool-member-type: cluster
    custom_resource_mode: true
    as3_validation: true
    ipam: true
    manage_configmaps: true
  bigip_login_secret: bigip-login
  image:
    pullPolicy: Always
    repo: f5networks/cntr-ingress-svcs
    user: registry.connect.redhat.com
  namespace: kube-system
  rbac:
    create: true
  resources: {}
  serviceAccount:
    create: true
  version: latest
```

23. 粘贴此内容后，单击“创建”。这会在 kube-system 命名空间中安装 CIS pod。

默认情况下，Red Hat OpenShift 提供了一种通过路由公开服务以实现 L7 负载平衡的方法。内置的 OpenShift 路由器负责宣传和处理这些路由的流量。但是，您也可以配置 F5 CIS 以通过外部 F5 BIG-IP 系统支持路由，该系统可以作为辅助路由器运行，也可以作为自托管 OpenShift 路由器的替代品运行。 CIS 在 BIG-IP 系统中创建一个虚拟服务器，充当 OpenShift 路由的路由器，而 BIG-IP 负责处理广告和流量路由。有关启用此功能的参数的信息，请参阅此处的文档。请注意，这些参数是在 apps/v1 API 中为 OpenShift Deployment 资源定义的。因此，当将这些与 F5BigIpCtlr 资源 cis.f5.com/v1 API 一起使用时，请将参数名称中的连字符 (-) 替换为下划线 (_)。

24. 传递给 CIS 资源创建的参数包括 `ipam: true`和 `custom_resource_mode: true`。这些参数是启用 CIS 与 IPAM 控制器集成所必需的。通过创建 F5 IPAM 资源验证 CIS 是否已启用 IPAM 集成。

```
[admin@rhel-7 ~]$ oc get f5ipam -n kube-system

NAMESPACE   NAME                                AGE
kube-system   ipam.10.61.181.19.ocp-vmw       43s
```

25. 创建 F5 IPAM 控制器所需的服务帐户、角色和角色绑定。创建一个 YAML 文件并粘贴以下内容。

```
[admin@rhel-7 ~]$ vi f5-ipam-rbac.yaml

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole
rules:
  - apiGroups: ["fic.f5.com"]
    resources: ["ipams","ipams/status"]
    verbs: ["get", "list", "watch", "update", "patch"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole-binding
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ipam-ctlr-clusterrole
subjects:
  - apiGroup: ""
    kind: ServiceAccount
    name: ipam-ctlr
    namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ipam-ctlr
  namespace: kube-system
```

26. 创建资源。

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-rbac.yaml

clusterrole.rbac.authorization.k8s.io/ipam-ctlr-clusterrole created
clusterrolebinding.rbac.authorization.k8s.io/ipam-ctlr-clusterrole-
binding created
serviceaccount/ipam-ctlr created
```

27. 创建一个 YAML 文件并粘贴下面提供的 F5 IPAM 部署定义。

更新下面的 spec.template.spec.containers[0].args 中的 ip-range 参数以反映与您的设置相对应的 ipamLabels 和 IP 地址范围。

ipam标签[`range1`和 `range2`在下面的示例中] 需要为 LoadBalancer 类型的服务进行注释，以便 IPAM 控制器从定义的范围内检测并分配 IP 地址。

```
[admin@rhel-7 ~]$ vi f5-ipam-deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    name: f5-ipam-controller
  name: f5-ipam-controller
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: f5-ipam-controller
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: f5-ipam-controller
    spec:
      containers:
      - args:
        - --orchestration=openshift
        - --ip-range='{"range1":"10.63.172.242-10.63.172.249",
"range2":"10.63.170.111-10.63.170.129"}'
        - --log-level=DEBUG
        command:
        - /app/bin/f5-ipam-controller
        image: registry.connect.redhat.com/f5networks/f5-ipam-
controller:latest
        imagePullPolicy: IfNotPresent
        name: f5-ipam-controller
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: ipam-ctlr
      serviceAccountName: ipam-ctlr
```

28. 创建 F5 IPAM 控制器部署。

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-deployment.yaml

deployment/f5-ipam-controller created
```

29. 验证 F5 IPAM 控制器 pod 是否正在运行。

```
[admin@rhel-7 ~]$ oc get pods -n kube-system

NAME                                      READY   STATUS    RESTARTS
AGE
f5-ipam-controller-5986cff5bd-2bvn6       1/1     Running   0
30s
f5-server-f5-bigip-ctlr-5d7578667d-qxdgj  1/1     Running   0
14m
```

30. 创建 F5 IPAM 模式。

```
[admin@rhel-7 ~]$ oc create -f
https://raw.githubusercontent.com/F5Networks/f5-ipam-
controller/main/docs/_static/schemas/ipam_schema.yaml

customresourcedefinition.apiextensions.k8s.io/ipams.fic.f5.com
```

验证

1. 创建 LoadBalancer 类型的服务

```
[admin@rhel-7 ~]$ vi example_svc.yaml

apiVersion: v1
kind: Service
metadata:
  annotations:
    cis.f5.com/ipamLabel: range1
  labels:
    app: f5-demo-test
  name: f5-demo-test
  namespace: default
spec:
  ports:
  - name: f5-demo-test
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: f5-demo-test
  sessionAffinity: None
  type: LoadBalancer
```

```
[admin@rhel-7 ~]$ oc create -f example_svc.yaml

service/f5-demo-test created
```

2. 检查 IPAM 控制器是否为其分配了外部 IP。

```
[admin@rhel-7 ~]$ oc get svc

NAME           TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
f5-demo-test   LoadBalancer   172.30.210.108   10.63.172.242
80:32605/TCP   27s
```

3. 创建部署并使用创建的 LoadBalancer 服务。

```
[admin@rhel-7 ~]$ vi example_deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: f5-demo-test
  name: f5-demo-test
spec:
  replicas: 2
  selector:
    matchLabels:
      app: f5-demo-test
  template:
    metadata:
      labels:
        app: f5-demo-test
    spec:
      containers:
      - env:
        - name: service_name
          value: f5-demo-test
        image: nginx
        imagePullPolicy: Always
        name: f5-demo-test
        ports:
        - containerPort: 80
          protocol: TCP
```

```
[admin@rhel-7 ~]$ oc create -f example_deployment.yaml

deployment/f5-demo-test created
```

4. 检查 pod 是否正在运行。

```
[admin@rhel-7 ~]$ oc get pods

NAME                           READY   STATUS    RESTARTS   AGE
f5-demo-test-57c46f6f98-47wwp   1/1     Running   0          27s
f5-demo-test-57c46f6f98-cl2m8   1/1     Running   0          27s
```

5. 检查BIG-IP系统中是否为OpenShift中LoadBalancer类型的服务创建了对应的虚拟服务器。导航到本地流量 > 虚拟服务器 > 虚拟服务器列表。

创建私有镜像仓库

对于大多数 Red Hat OpenShift 部署，使用公共注册表 "Quay.io"或者 "DockerHub"满足大多数客户的需求。然而，有时客户可能希望托管他们自己的私人或定制图像。

此过程记录了如何创建由Trident和NetApp ONTAP提供的持久卷支持的私有映像注册表。

Trident Protect 需要一个注册表来托管Astra容器所需的图像。以下部分介绍在 Red Hat OpenShift 集群上设置私有注册表以及推送支持安装Trident Protect 所需的映像的步骤。

创建私有镜像仓库

1. 从当前默认存储类中删除默认注释，并将 Trident 支持的存储类注释为 OpenShift 集群的默认存储类。

```
[netapp-user@rhel7 ~]$ oc patch storageclass thin -p '{"metadata":
{"annotations": {"storageclass.kubernetes.io/is-default-class":
"false"}}}'
storageclass.storage.k8s.io/thin patched

[netapp-user@rhel7 ~]$ oc patch storageclass ocp-trident -p
'{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-
class": "true"}}}'
storageclass.storage.k8s.io/ocp-trident patched
```

2. 通过在以下位置输入以下存储参数来编辑 imageregistry 操作符 `spec`部分。

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

storage:
  pvc:
    claim:
```

3. 在 `spec`用于创建具有自定义主机名的 OpenShift 路由的部分。保存并退出。

```
routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
```

当您想要为路由指定自定义主机名时，可以使用上述路由配置。如果您希望 OpenShift 创建具有默认主机名的路由，则可以将以下参数添加到 `spec`部分： `defaultRoute: true`。

自定义 TLS 证书

当您使用自定义主机名进行路由时，默认情况下，它会使用 OpenShift Ingress 操作员的默认 TLS 配置。但是，您可以向路由添加自定义 TLS 配置。为此，请完成以下步骤。

a. 使用路由的 TLS 证书和密钥创建一个秘密。

```
[netapp-user@rhel7 ~]$ oc create secret tls astra-route-tls -n
openshift-image-registry –cert/home/admin/netapp-astra/tls.crt
--key=/home/admin/netapp-astra/tls.key
```

b. 编辑 imageregistry 操作符，添加以下参数 `spec`部分。

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
    secretName: astra-route-tls
```

4. 再次编辑imageregistry Operator，将Operator的管理状态改为 `Managed`状态。保存并退出。

```
oc edit configs.imageregistry/cluster

managementState: Managed
```

5. 如果满足所有先决条件，则会为私有镜像注册表创建 PVC、pod 和服务。几分钟后，注册表就会启动。

```
[netapp-user@rhel7 ~]$oc get all -n openshift-image-registry

NAME                                                    READY   STATUS
```

```
RESTARTS   AGE
pod/cluster-image-registry-operator-74f6d954b6-rb7zr   1/1     Running
3          90d
pod/image-pruner-1627257600-f5cpj                      0/1     Completed
0          2d9h
pod/image-pruner-1627344000-swqx9                      0/1     Completed
0          33h
pod/image-pruner-1627430400-rv5nt                      0/1     Completed
0          9h
pod/image-registry-6758b547f-6pnj8                     1/1     Running
0          76m
pod/node-ca-bwb5r                                      1/1     Running
0          90d
pod/node-ca-f8w54                                      1/1     Running
0          90d
pod/node-ca-gjx7h                                      1/1     Running
0          90d
pod/node-ca-lcx4k                                      1/1     Running
0          33d
pod/node-ca-v7zmx                                      1/1     Running
0          7d21h
pod/node-ca-xpppp                                      1/1     Running
0          89d

NAME                              TYPE        CLUSTER-IP       EXTERNAL-
IP   PORT(S)     AGE
service/image-registry            ClusterIP   172.30.196.167   <none>
5000/TCP    15h
service/image-registry-operator   ClusterIP   None             <none>
60000/TCP   90d

NAME                     DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/node-ca   6         6         6       6            6
kubernetes.io/os=linux   90d

NAME                                           READY   UP-TO-DATE
AVAILABLE   AGE
deployment.apps/cluster-image-registry-operator   1/1     1            1
90d
deployment.apps/image-registry                    1/1     1            1
15h

NAME                                                      DESIRED
CURRENT   READY   AGE
replicaset.apps/cluster-image-registry-operator-74f6d954b6   1         1
```

```
1       90d
replicaset.apps/image-registry-6758b547f                    1         1
1       76m
replicaset.apps/image-registry-78bfbd7f59                   0         0
0       15h
replicaset.apps/image-registry-7fcc8d6cc8                   0         0
0       80m
replicaset.apps/image-registry-864f88f5b                    0         0
0       15h
replicaset.apps/image-registry-cb47fffb                     0         0
0       10h

NAME                                COMPLETIONS   DURATION   AGE
job.batch/image-pruner-1627257600   1/1           10s        2d9h
job.batch/image-pruner-1627344000   1/1           6s         33h
job.batch/image-pruner-1627430400   1/1           5s         9h

NAME                           SCHEDULE    SUSPEND   ACTIVE   LAST
SCHEDULE   AGE
cronjob.batch/image-pruner   0 0 * * *   False     0        9h
90d

NAME                                     HOST/PORT
PATH   SERVICES          PORT    TERMINATION   WILDCARD
route.route.openshift.io/public-routes   astra-registry.apps.ocp-
vmw.cie.netapp.com          image-registry   <all>   reencrypt     None
```

6. 如果您正在为入口操作员 OpenShift 注册表路由使用默认 TLS 证书，则可以使用以下命令获取 TLS 证书。

```
[netapp-user@rhel7 ~]$ oc extract secret/router-ca --keys=tls.crt -n
openshift-ingress-operator
```

7. 为了允许 OpenShift 节点访问并从注册表中提取图像，请将证书添加到 OpenShift 节点上的 docker 客户端。在 `openshift-config`命名空间使用 TLS 证书并将其修补到集群映像配置以使证书受信任。

```
[netapp-user@rhel7 ~]$ oc create configmap astra-ca -n openshift-config
--from-file=astra-registry.apps.ocp-vmw.cie.netapp.com=tls.crt

[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster
--patch '{"spec":{"additionalTrustedCA":{"name":"astra-ca"}}}'
--type=merge
```

8. OpenShift 内部注册表由身份验证控制。所有OpenShift用户都可以访问OpenShift注册表，但登录用户可以执行的操作取决于用户权限。

a. 要允许用户或一组用户从注册表中提取图像，必须为用户分配 registry-viewer 角色。

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-viewer
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-viewer
ocp-user-group
```

b. 要允许用户或用户组写入或推送图像，必须为用户分配注册表编辑器角色。

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-editor
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-editor
ocp-user-group
```

9. 为了让 OpenShift 节点访问注册表并推送或拉取图像，您需要配置一个拉取密钥。

```
[netapp-user@rhel7 ~]$ oc create secret docker-registry astra-registry-
credentials --docker-server=astra-registry.apps.ocp-vmw.cie.netapp.com
--docker-username=ocp-user --docker-password=password
```

10. 然后可以将这个拉取机密修补到服务帐户或在相应的 pod 定义中引用。

a. 要将其修补到服务帐户，请运行以下命令。

```
[netapp-user@rhel7 ~]$ oc secrets link <service_account_name> astra-
registry-credentials --for=pull
```

b. 要在 pod 定义中引用 pull secret，请将以下参数添加到 `spec`部分。

```
imagePullSecrets:
  - name: astra-registry-credentials
```

11. 要从 OpenShift 节点以外的工作站推送或拉取图像，请完成以下步骤。

a. 将 TLS 证书添加到 docker 客户端。

```
[netapp-user@rhel7 ~]$ sudo mkdir /etc/docker/certs.d/astra-
registry.apps.ocp-vmw.cie.netapp.com

[netapp-user@rhel7 ~]$ sudo cp /path/to/tls.crt
/etc/docker/certs.d/astra-registry.apps.ocp-vmw.cie.netapp.com
```

b. 使用 oc login 命令登录 OpenShift。

```
[netapp-user@rhel7 ~]$ oc login --token=sha256~D49SpB_lesSrJYwrM0LIO
-VRcjWHu0a27vKa0 --server=https://api.ocp-vmw.cie.netapp.com:6443
```

c. 使用 OpenShift 用户凭据通过 podman/docker 命令登录注册表。

podman

```
[netapp-user@rhel7 ~]$ podman login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t) --tls
-verify=false
```

+ 注意：如果您使用 `kubeadmin`用户登录私有注册中心，然后使用令牌而不是密码。

码头工人

```
[netapp-user@rhel7 ~]$ docker login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t)
```

+ 注意：如果您使用 `kubeadmin`用户登录私有注册中心，然后使用令牌而不是密码。

d. 推送或拉取图像。

podman

```
[netapp-user@rhel7 ~]$ podman push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ podman pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

码头工人

```
[netapp-user@rhel7 ~]$ docker push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ docker pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

解决方案验证和用例

解决方案验证和用例：**Red Hat OpenShift** 与**NetApp**

本页提供的示例是 Red Hat OpenShift 与NetApp的解决方案验证和用例。

- "使用持久存储部署 Jenkins CI/CD 管道"
- "使用NetApp在 Red Hat OpenShift 上配置多租户"
- "搭载NetApp ONTAP 的Red Hat OpenShift 虚拟化"
- "NetApp助力 Red Hat OpenShift 上 Kubernetes 的高级集群管理"

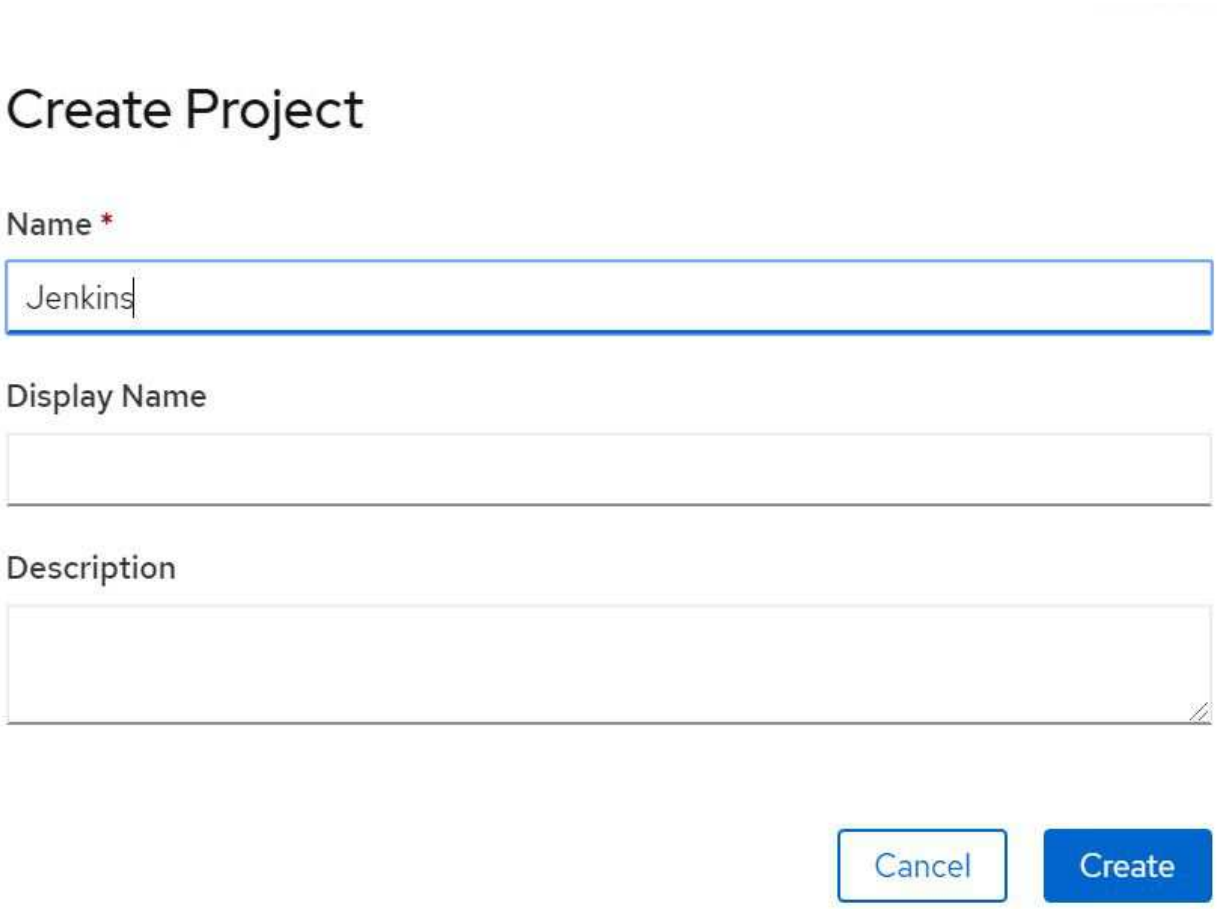
使用持久存储部署 **Jenkins CI/CD** 管道：**Red Hat OpenShift** 与**NetApp**

本节提供使用 Jenkins 部署持续集成/持续交付或部署 (CI/CD) 管道以验证解决方案运行的步骤。

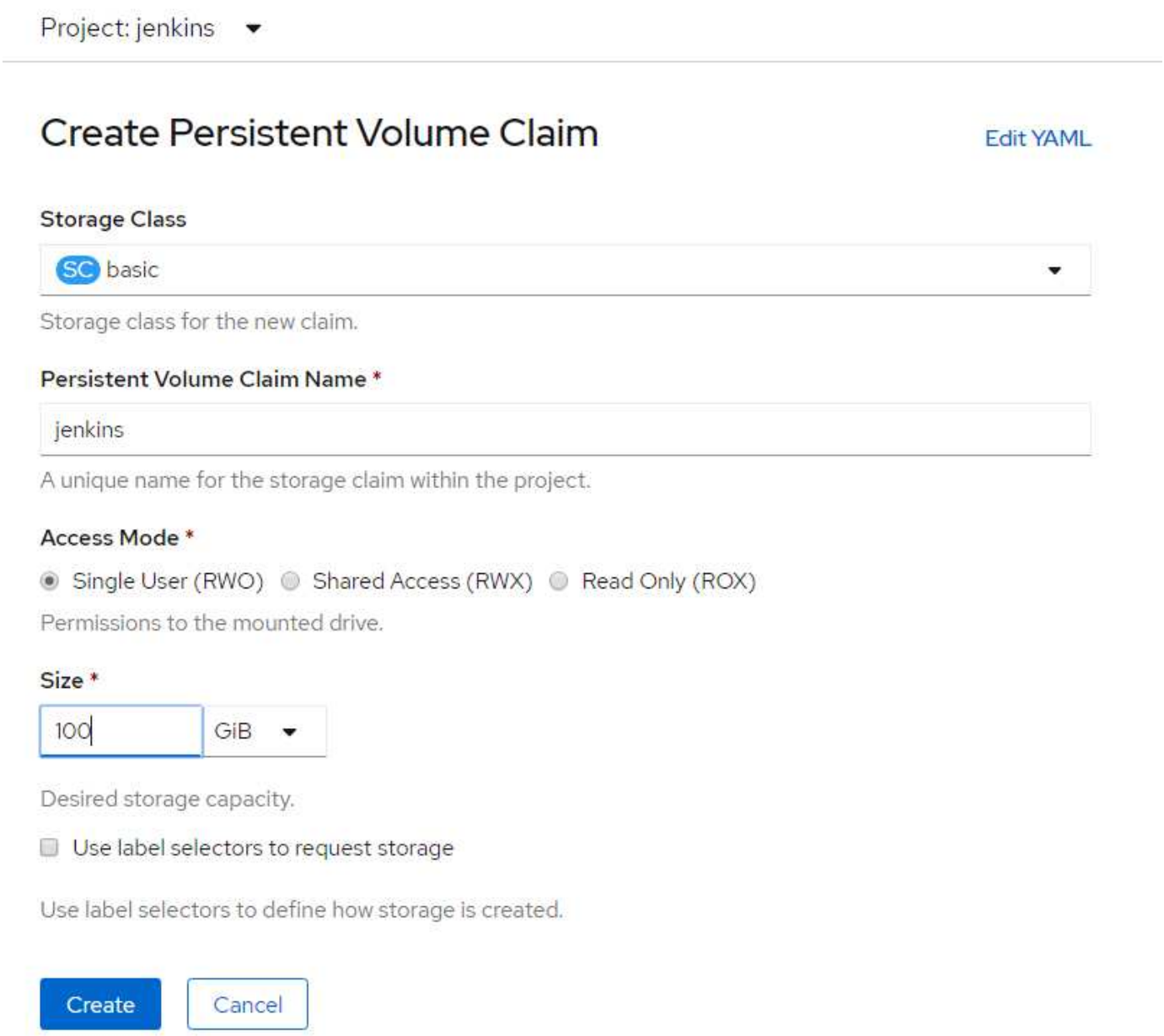
创建**Jenkins**部署所需的资源

要创建部署 Jenkins 应用程序所需的资源，请完成以下步骤：

1. 创建一个名为 Jenkins 的新项目。



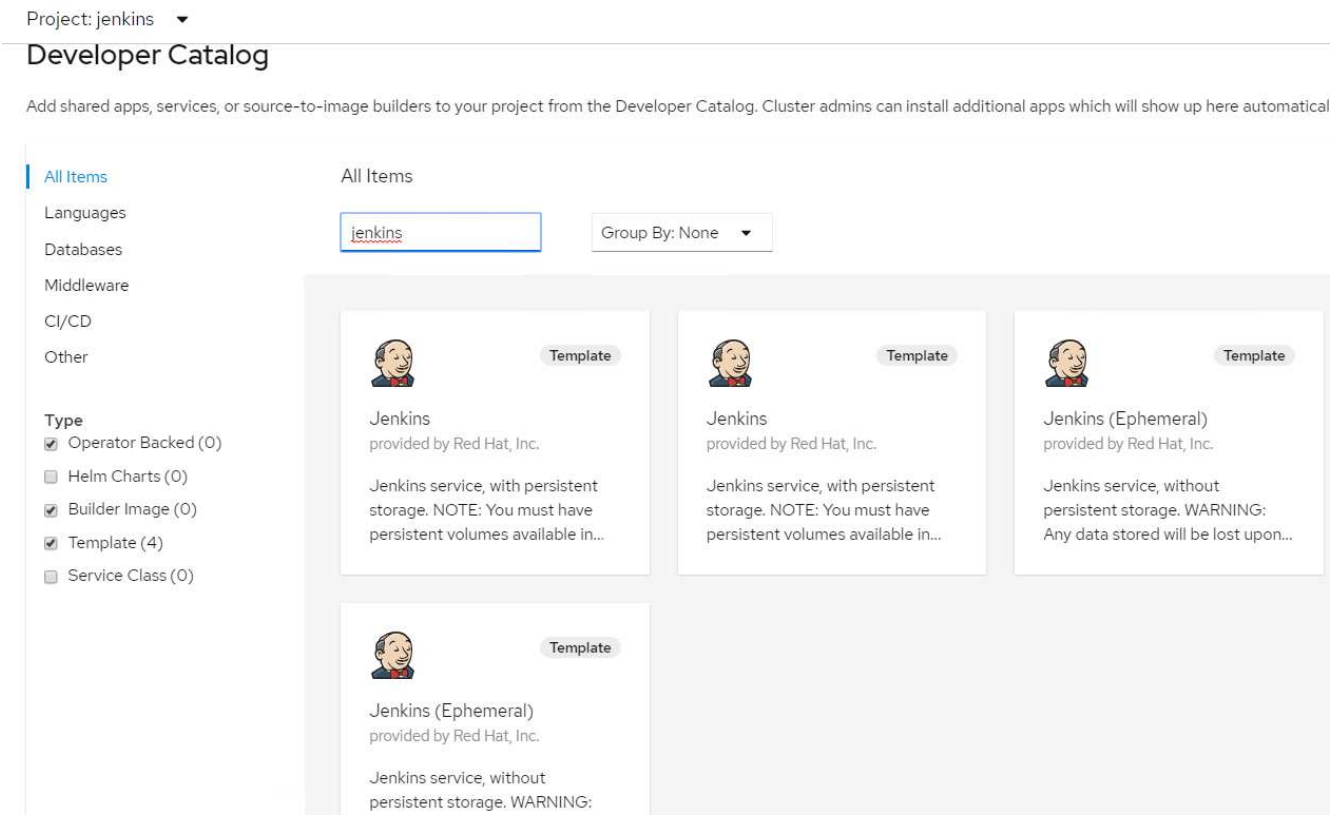
2. 在这个例子中，我们部署了具有持久存储的 Jenkins。为了支持 Jenkins 构建，请创建 PVC。导航到存储 > 持久卷声明，然后单击创建持久卷声明。选择刚刚创建的存储类，确保持久卷声明名称为jenkins，选择合适的大小和访问模式，然后单击创建。



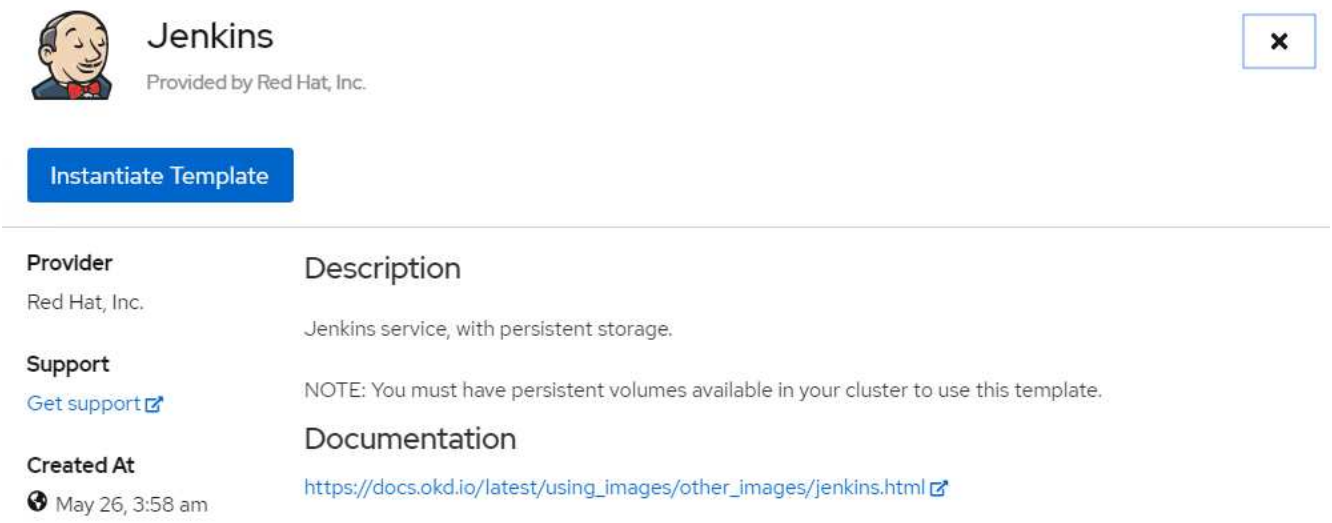
使用持久存储部署 **Jenkins**

要使用持久存储部署 Jenkins，请完成以下步骤：

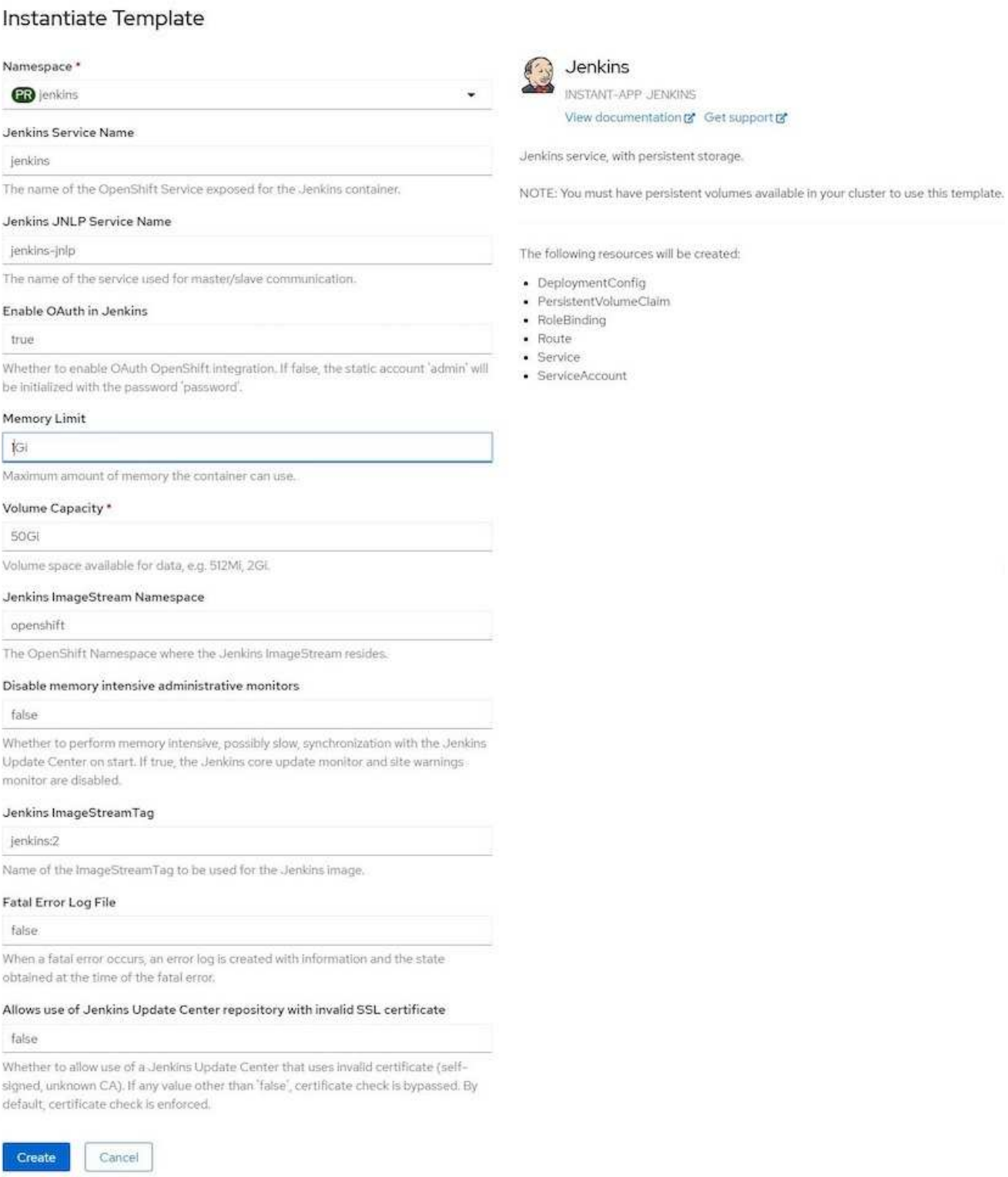
1. 在左上角，将角色从管理员更改为开发人员。单击 +添加并选择来自目录。在按关键字过滤栏中，搜索 jenkins。选择具有持久存储的 Jenkins 服务。



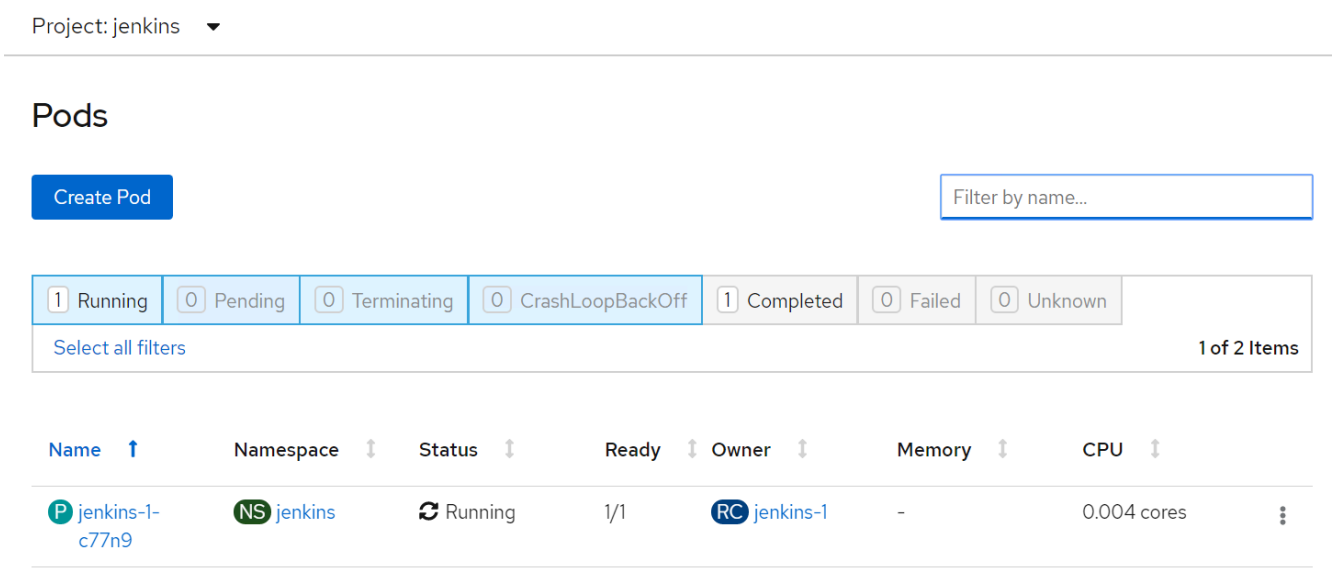
2. 点击 `Instantiate Template`。



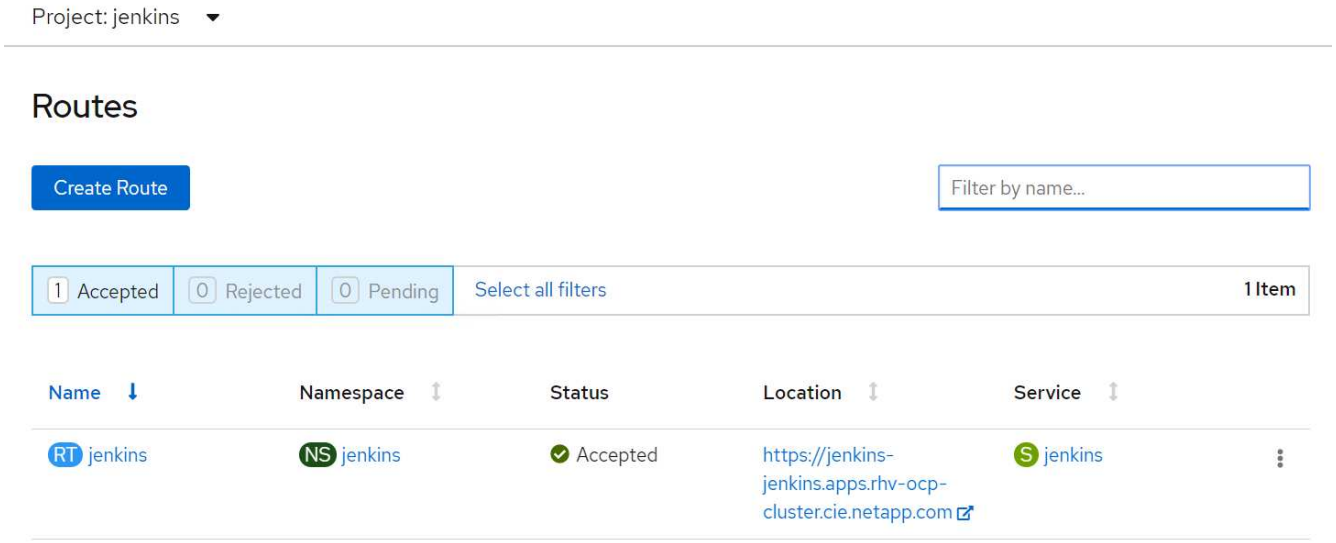
3. 默认情况下，会填充 Jenkins 应用程序的详细信息。根据您的需求，修改参数并单击“创建”。此过程创建了在 OpenShift 上支持 Jenkins 所需的所有资源。



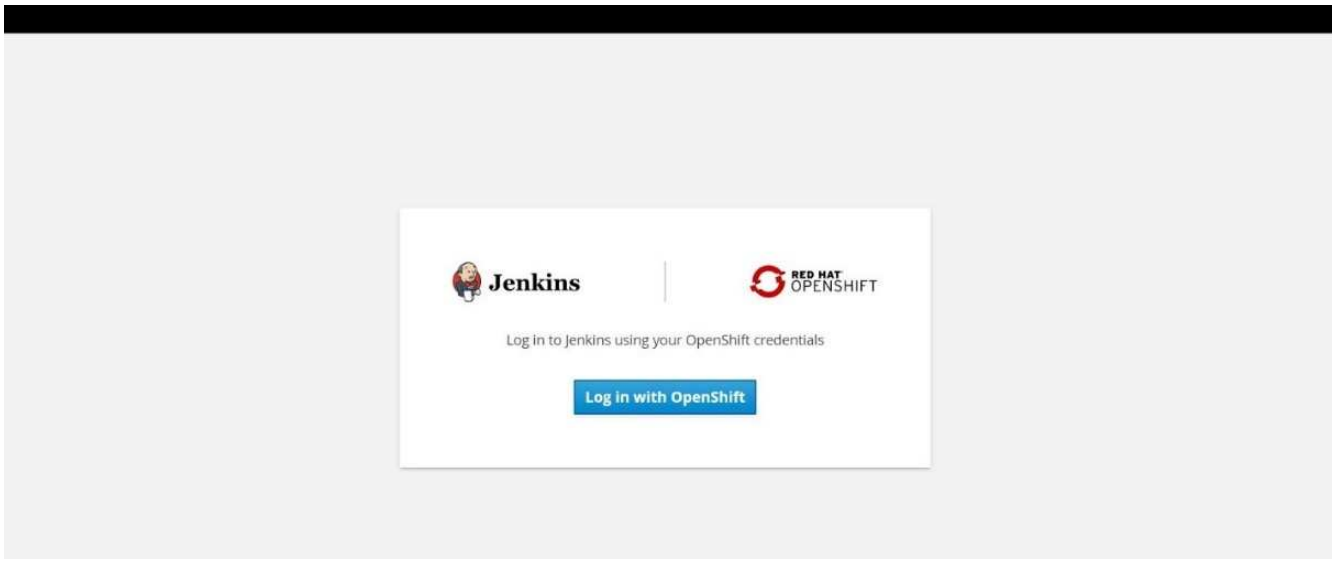
4. Jenkins pod 大约需要 10 到 12 分钟才能进入就绪状态。



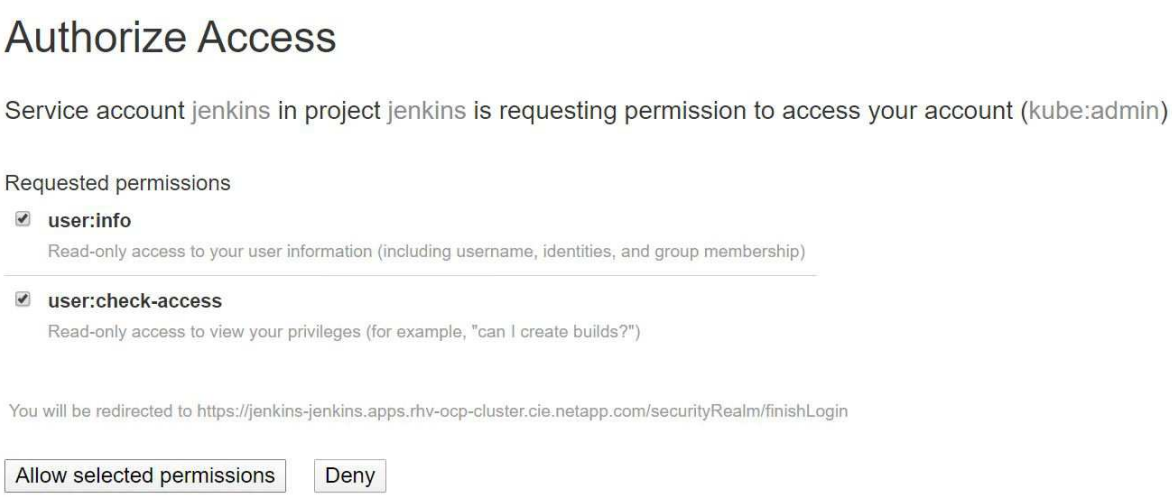
5. 实例化 Pod 后，导航至“网络”>“路由”。要打开 Jenkins 网页，请单击为 jenkins 路由提供的 URL。



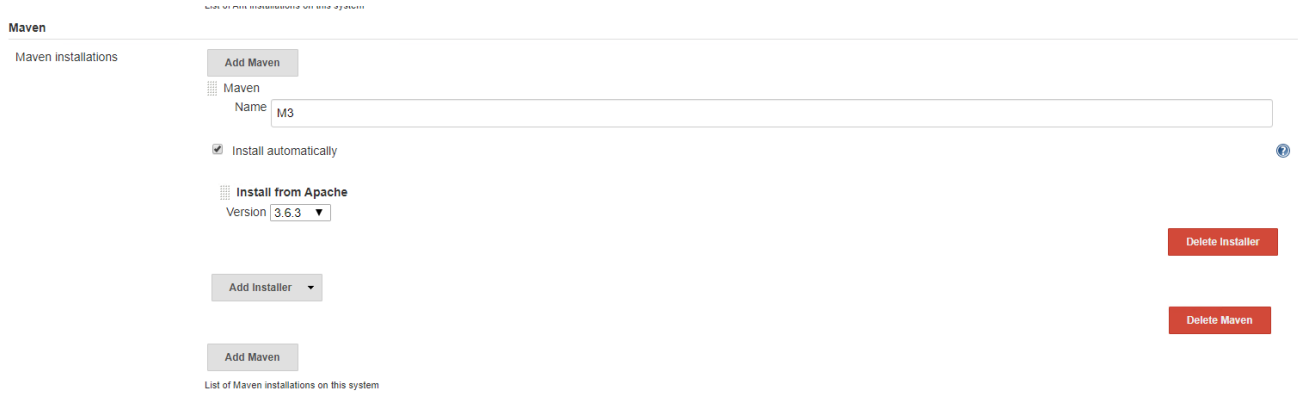
6. 由于在创建 Jenkins 应用程序时使用了 OpenShift OAuth，因此请单击使用 OpenShift 登录。



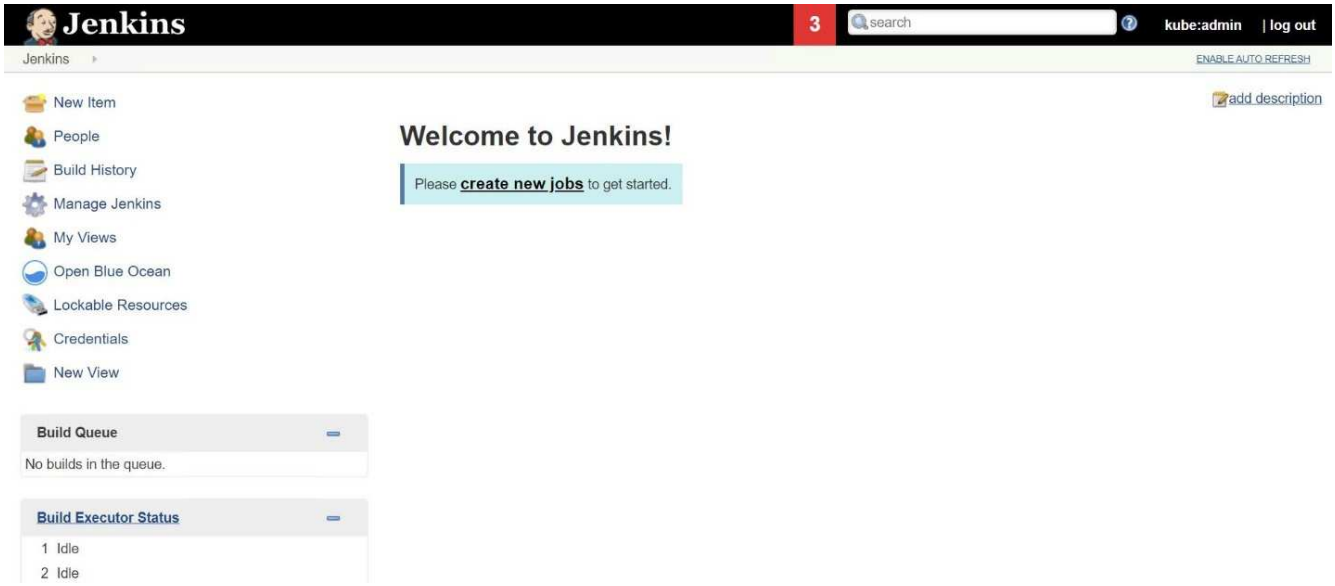
7. 授权 Jenkins 服务帐户访问 OpenShift 用户。



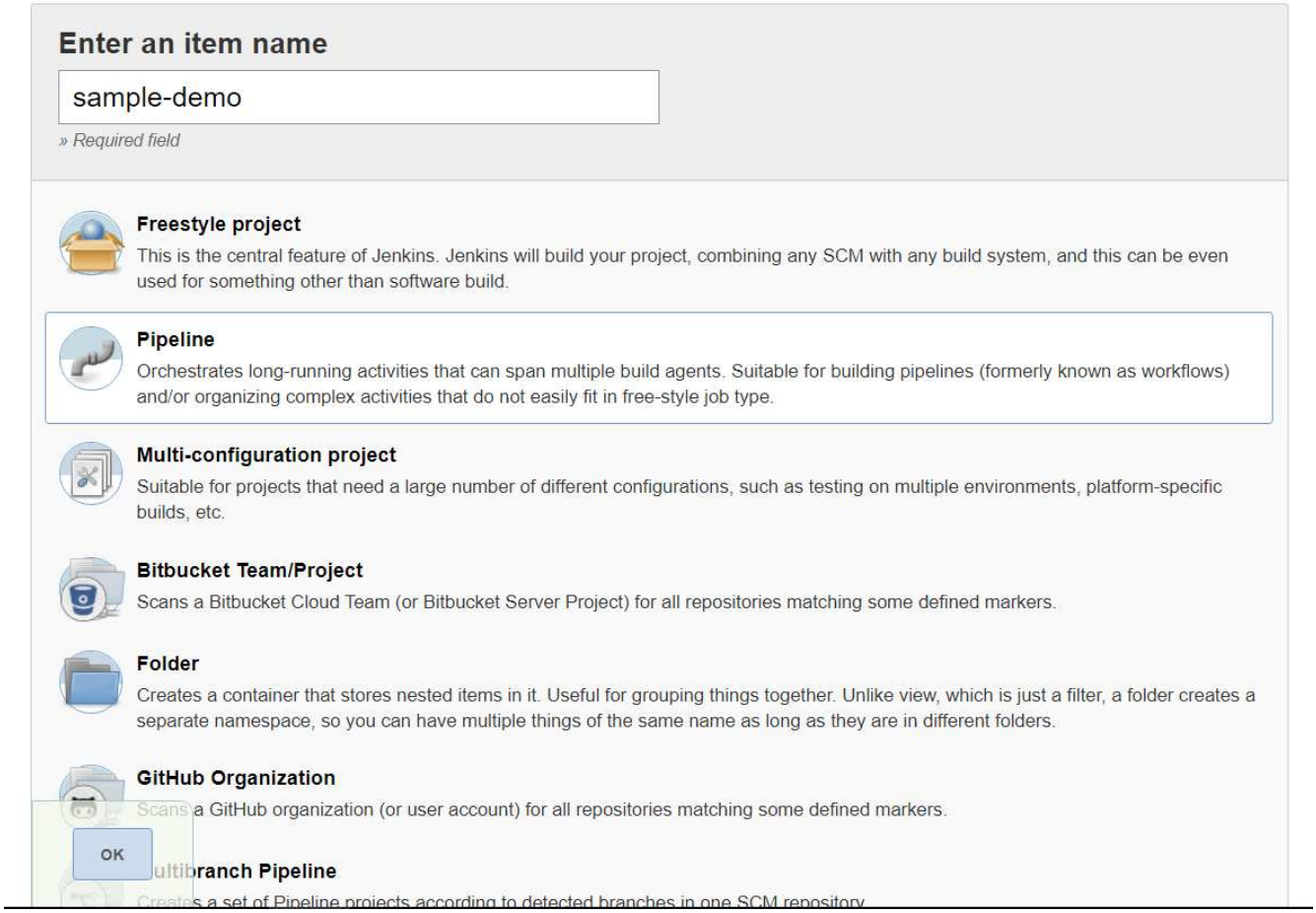
8. 将显示 Jenkins 欢迎页面。因为我们使用的是 Maven 构建，所以请先完成 Maven 安装。导航到“管理 Jenkins”>“全局工具配置”，然后在“Maven”子标题中单击“添加 Maven”。输入您选择的名称并确保选择了“自动安装”选项。单击“Save”。



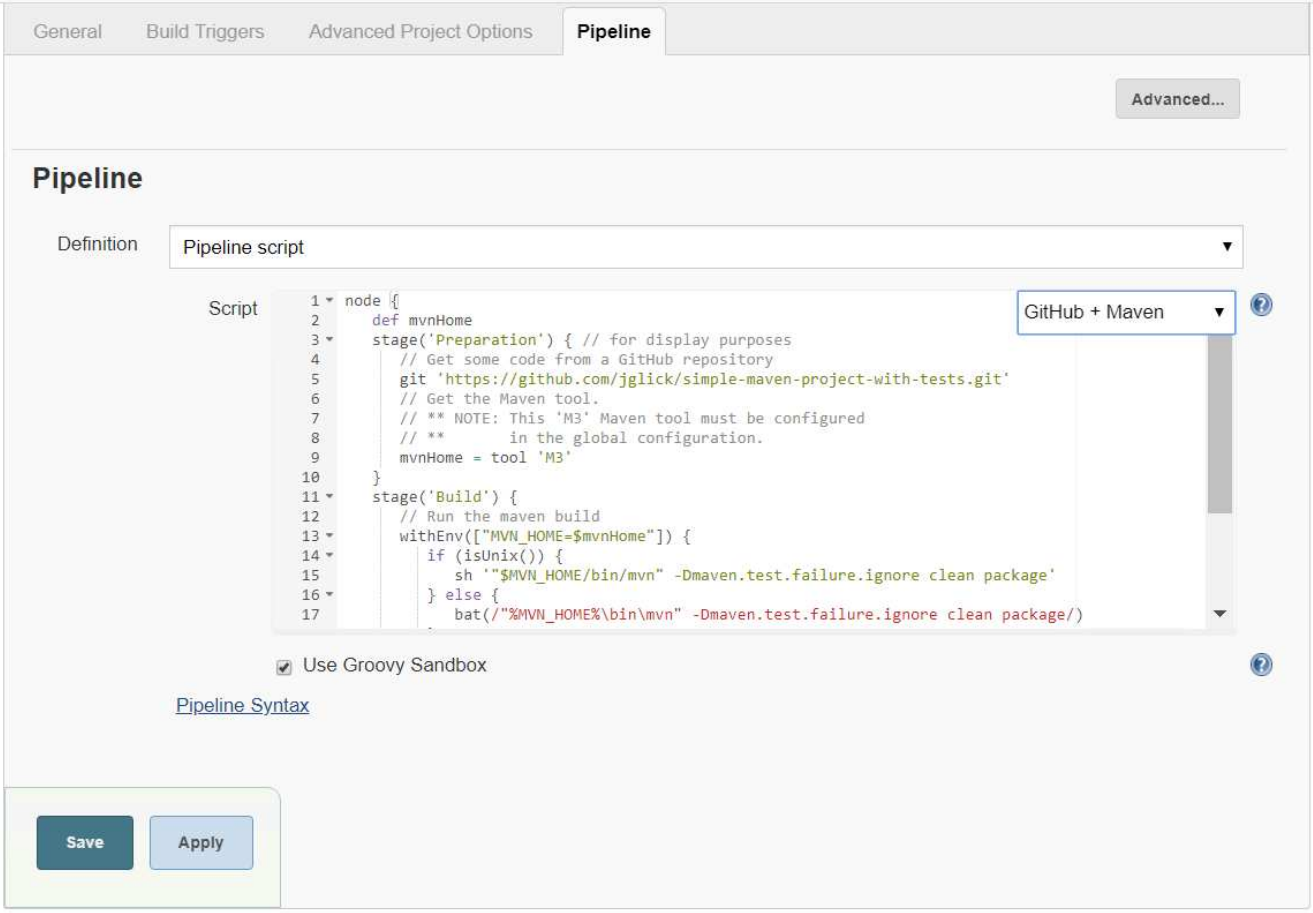
9. 您现在可以创建一个管道来演示 CI/CD 工作流程。在主页上，单击左侧菜单中的“创建新作业”或“新项目”。



10. 在创建项目页面上，输入您选择的名称，选择管道，然后单击确定。



11. 选择管道选项卡。从尝试示例管道下拉菜单中，选择 Github + Maven。代码会自动填充。单击“Save”。



12. 单击“立即构建”以通过准备、构建和测试阶段触发开发。完成整个构建过程并显示构建结果可能需要几分钟。

13. 每当代码发生任何变化时，都可以重建管道来修补新版本的软件，从而实现持续集成和持续交付。单击“最近更改”可跟踪自上一版本以来的更改。

配置多租户

使用**NetApp**在 **Red Hat OpenShift** 上配置多租户

许多在容器上运行多个应用程序或工作负载的组织倾向于为每个应用程序或工作负载部署一个 Red Hat OpenShift 集群。这使得他们能够对应用程序或工作负载实施严格隔离，优化性能并减少安全漏洞。但是，为每个应用程序部署单独的 Red Hat OpenShift 集群会带来一系列问题。它增加了必须自行监控和管理每个集群的运营开销，由于不同应用程序的专用资源而增加了成本，并阻碍了有效的可扩展性。

为了克服这些问题，可以考虑在单个 Red Hat OpenShift 集群中运行所有应用程序或工作负载。但在这样的架构下，资源隔离和应用安全漏洞是主要挑战之一。一个工作负载中的任何安全漏洞都可能自然蔓延到另一个工作负载，从而扩大影响范围。此外，由于默认情况下没有资源分配策略，因此一个应用程序突然不受控制地利用资源都会影响另一个应用程序的性能。

因此，组织寻求能够兼顾两方面优势的解决方案，例如，允许他们在单个集群中运行所有工作负载，同时又为每个工作负载提供专用集群的优势。

一个有效的解决方案是在 Red Hat OpenShift 上配置多租户。多租户是一种允许多个租户在同一个集群上共存的架构，对资源、安全性等进行适当的隔离。在此上下文中，租户可被视为集群资源的子集，配置为由特定用户组

用于专用目的。在 Red Hat OpenShift 集群上配置多租户具有以下优势：

- 通过共享集群资源来减少资本支出和运营支出
- 降低运营和管理开销
- 保护工作负载免受安全漏洞的交叉污染
- 保护工作负载免受资源争用导致的意外性能下降

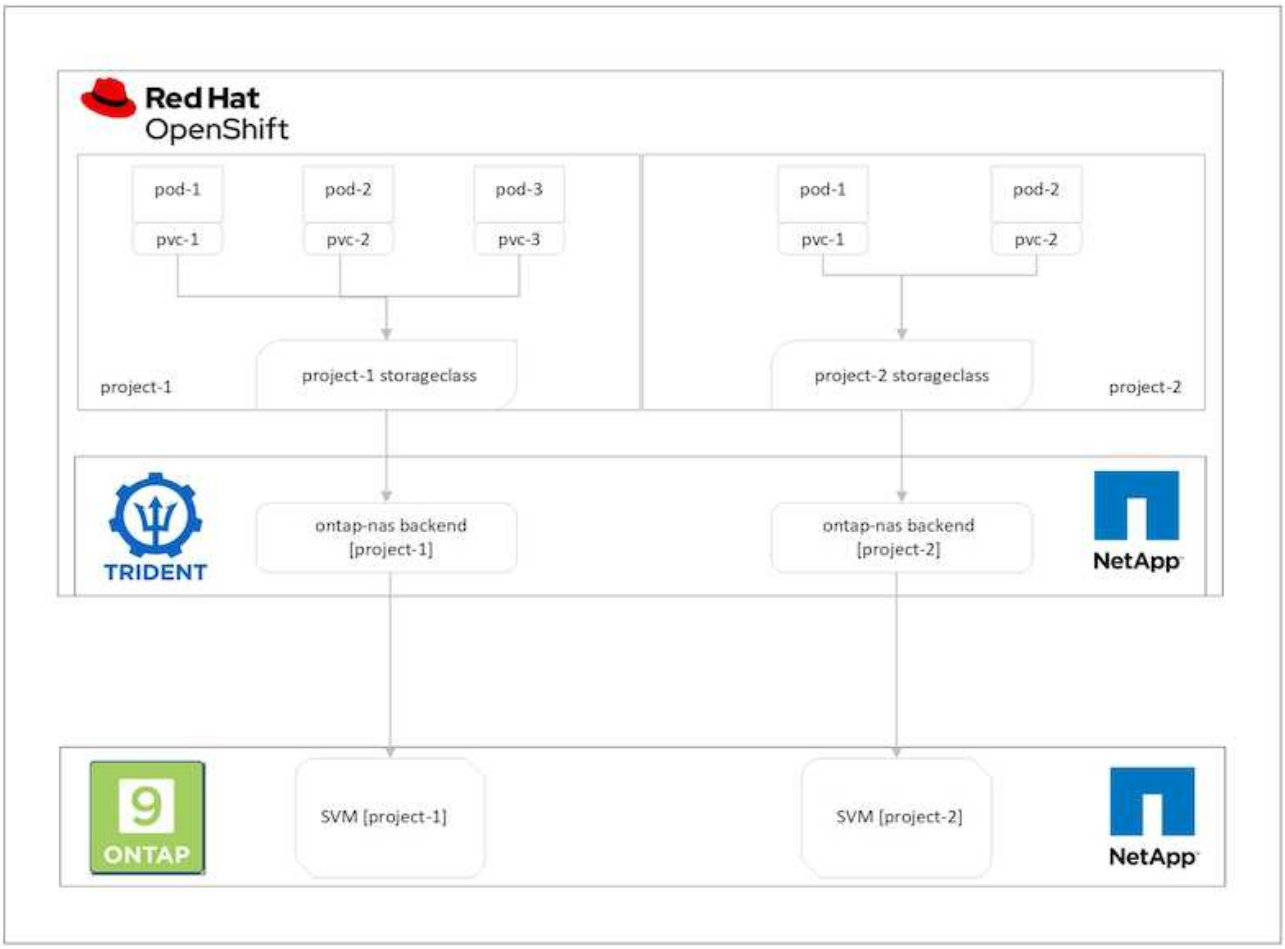
对于完全实现的多租户 OpenShift 集群，必须为属于不同资源桶的集群资源配置配额和限制：计算、存储、网络、安全等。虽然我们在此解决方案中涵盖了所有资源桶的某些方面，但我们专注于通过在由 NetApp ONTAP 支持的TridentNetApp分配的存储资源上配置多租户来隔离和保护同一 Red Hat OpenShift 集群上多个工作负载所服务或使用的数据的ONTAP实践。

架构

尽管NetApp ONTAP支持的 Red Hat OpenShift 和Trident默认不提供工作负载之间的隔离，但它们提供了可用于配置多租户的广泛功能。为了更好地理解在由NetApp ONTAP支持的Trident的 Red Hat OpenShift 集群上设计多租户解决方案，让我们考虑一个具有一组要求的示例并概述围绕它的配置。

假设一个组织在 Red Hat OpenShift 集群上运行两个工作负载，这是两个不同团队正在开展的两个项目的一部分。这些工作负载的数据驻留在由Trident在NetApp ONTAP NAS 后端动态配置的 PVC 上。该组织需要为这两个工作负载设计一个多租户解决方案，并隔离用于这些项目的资源，以确保维护安全性和性能，主要关注为这些应用程序提供服务的数据。

下图描述了由NetApp ONTAP支持的带有Trident的 Red Hat OpenShift 集群上的多租户解决方案。



技术要求

1. NetApp ONTAP存储集群
2. Red Hat OpenShift 集群
3. Trident

Red Hat OpenShift – 集群资源

从 Red Hat OpenShift 集群的角度来看，首先要启动的顶级资源是项目。 OpenShift 项目可以看作是一个集群资源，它将整个 OpenShift 集群划分为多个虚拟集群。因此，项目级别的隔离为配置多租户提供了基础。

接下来是在集群中配置 RBAC。最佳做法是让所有开发人员在身份提供者 (IdP) 中将单个项目或工作负载配置到单个用户组中。 Red Hat OpenShift 允许 IdP 集成和用户组同步，从而允许将来自 IdP 的用户和组导入到集群中。这有助于集群管理员将专用于某个项目的集群资源的访问隔离给从事该项目的一个或多个用户组，从而限制对任何集群资源的未经授权的访问。要了解有关 IdP 与 Red Hat OpenShift 集成的更多信息，请参阅文档 "此处"。

NetApp ONTAP

隔离作为 Red Hat OpenShift 集群的持久存储提供程序的共享存储非常重要，以确保在每个项目的存储上创建的卷对于主机来说就像是在单独的存储上创建的一样。为此，请在NetApp ONTAP上创建与项目或工作负载数量相同的 SVM（存储虚拟机），并将每个 SVM 专用于一个工作负载。

Trident

在NetApp ONTAP上为不同的项目创建不同的 SVM 后，必须将每个 SVM 映射到不同的Trident后端。 Trident上的后端配置驱动持久存储到 OpenShift 集群资源的分配，并且需要映射到 SVM 的详细信息。这至少应该是后端的协议驱动程序。或者，它允许您定义如何在存储上配置卷，并为卷的大小或聚合的使用等设置限制。关于Trident后端定义的详细信息可以参见 "此处"。

Red Hat OpenShift – 存储资源

配置Trident后端后，下一步是配置 StorageClasses。配置与后端数量相同的存储类，为每个存储类提供仅在一个后端上启动卷的访问权限。我们可以在定义存储类时使用 storagePools 参数将 StorageClass 映射到特定的Trident后端。定义存储类的详细信息可以在这里找到 "此处"。因此，从 StorageClass 到Trident后端存在一对一映射，该映射指向一个 SVM。这可确保通过分配给该项目的 StorageClass 提出的所有存储声明均由专用于该项目的 SVM 提供服务。

由于存储类不是命名空间资源，我们如何确保另一个命名空间或项目中的 pod 对一个项目的存储类的存储声明被拒绝？答案是使用 ResourceQuotas。 ResourceQuotas 是控制每个项目资源总使用量的对象。它可以限制项目中对象可消耗的资源数量和总量。几乎所有项目的资源都可以使用 ResourceQuotas 进行限制，有效使用 ResourceQuotas 可以帮助组织削减因资源过度配置或过度消耗而导致的成本和中断。请参阅文档 "此处"了解更多信息。

对于这种用例，我们需要限制特定项目中的 pod 从非专用于其项目的存储类中声明存储。为此，我们需要通过设置来限制其他存储类的持久卷声明 `<storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims`为 0。此外，集群管理员必须确保项目中的开发人员无权修改资源配额。

配置

对于任何多租户解决方案，任何用户都不能访问超出所需的集群资源。因此，作为多租户配置的一部分而要配置的整个资源集被划分到集群管理员、存储管理员和每个项目的开发人员之间。

下表概述了不同用户需要执行的不同任务：

角色	**Tasks**
集群管理员	为不同的应用程序或工作负载创建项目
	为 storage-admin 创建 ClusterRoles 和 RoleBindings
	为开发人员创建角色和角色绑定，分配对特定项目的访问权限
	[可选] 配置项目以在特定节点上调度 Pod
存储管理	在NetApp ONTAP上创建 SVM
	创建Trident后端
	创建存储类
	创建存储资源配额

角色	Tasks
开发人员	验证在指定项目中创建或修补 PVC 或 Pod 的权限
	验证在另一个项目中创建或修补 PVC 或 Pod 的权限
	验证查看或编辑项目、资源配额和存储类的权限

配置

以下是使用NetApp在 Red Hat OpenShift 上配置多租户的先决条件。

前提条件

- NetApp ONTAP集群
- Red Hat OpenShift 集群
- 集群上安装的Trident
- 安装了 tridentctl 和 oc 工具并添加到 $PATH 的管理员工作站
- ONTAP管理员访问权限
- 集群管理员访问 OpenShift 集群
- 集群与身份提供者集成
- 配置身份提供者以有效区分不同团队中的用户

配置：集群管理任务

Red Hat OpenShift 集群管理员执行以下任务：

1. 以集群管理员身份登录 Red Hat OpenShift 集群。
2. 创建两个项目，分别对应不同的工程。

```
oc create namespace project-1
oc create namespace project-2
```

3. 为 project-1 创建开发人员角色。

```
cat << EOF | oc create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: project-1
  name: developer-project-1
rules:
  - verbs:
      - '*'
```

```
    apiGroups:
      - apps
      - batch
      - autoscaling
      - extensions
      - networking.k8s.io
      - policy
      - apps.openshift.io
      - build.openshift.io
      - image.openshift.io
      - ingress.operator.openshift.io
      - route.openshift.io
      - snapshot.storage.k8s.io
      - template.openshift.io
    resources:
      - '*'
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - bindings
      - configmaps
      - endpoints
      - events
      - persistentvolumeclaims
      - pods
      - pods/log
      - pods/attach
      - podtemplates
      - replicationcontrollers
      - services
      - limitranges
      - namespaces
      - componentstatuses
      - nodes
  - verbs:
      - '*'
    apiGroups:
      - trident.netapp.io
    resources:
      - tridentsnapshots
EOF
```

本节提供的角色定义仅仅是一个示例。必须根据最终用户的要求来定义开发人员的角色。

1. 同样，为 project-2 创建开发人员角色。
2. 所有 OpenShift 和NetApp存储资源通常由存储管理员管理。存储管理员的访问权限由安装Trident时创建的 trident 操作员角色控制。除此之外，存储管理员还需要访问 ResourceQuotas 来控制存储的消耗方式。
3. 创建一个用于管理集群中所有项目的 ResourceQuotas 的角色，并将其附加到存储管理员。

```
cat << EOF | oc create -f -
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: resource-quotas-role
rules:
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - resourcequotas
  - verbs:
      - '*'
    apiGroups:
      - quota.openshift.io
    resources:
      - '*'
EOF
```

4. 确保集群与组织的身份提供者集成，并且用户组与集群组同步。以下示例显示身份提供者已与集群集成并与用户组同步。

```
$ oc get groups
NAME                          USERS
ocp-netapp-storage-admins     ocp-netapp-storage-admin
ocp-project-1                 ocp-project-1-user
ocp-project-2                 ocp-project-2-user
```

1. 为存储管理员配置 ClusterRoleBindings。

```
cat << EOF | oc create -f -
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: netapp-storage-admin-trident-operator
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-netapp-storage-admins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-operator
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: netapp-storage-admin-resource-quotas-cr
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-netapp-storage-admins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: resource-quotas-role
EOF
```

对于存储管理员，必须绑定两个角色：trident-operator 和 resource-quotas。

1. 为开发者创建 RoleBindings，将developer-project-1 角色绑定到 project-1 中对应的组（ocp-project-1）。

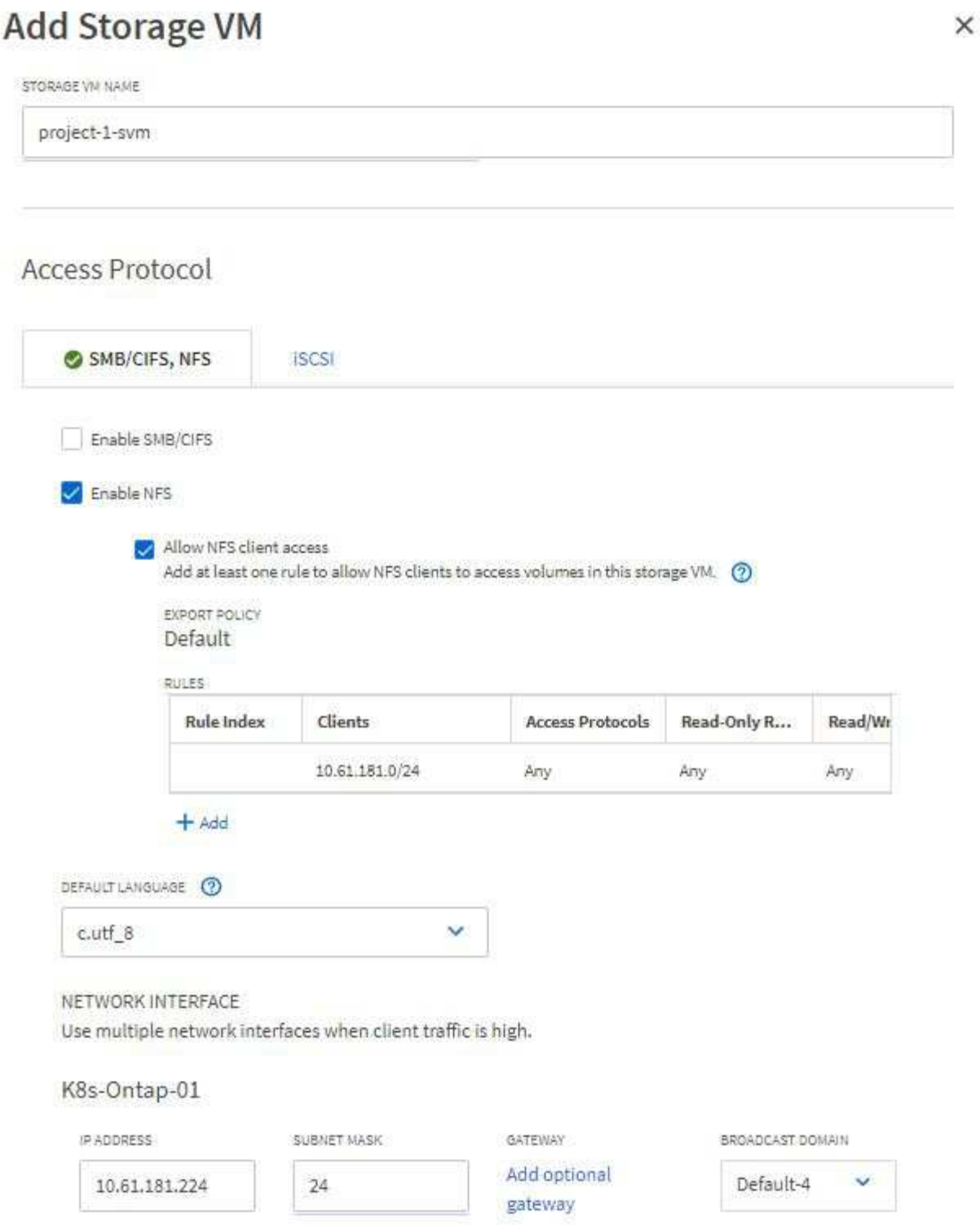
```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-1-developer
  namespace: project-1
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-project-1
EOF
```

2. 同样的，在project-2中为开发者创建RoleBindings，将开发者角色绑定到对应的用户组。

配置：存储管理任务

存储管理员必须配置以下资源：

1. 以管理员身份登录NetApp ONTAP集群。
2. 导航到存储>存储虚拟机，然后单击添加。通过提供所需的详细信息，创建两个 SVM，一个用于项目 1，另一个用于项目 2。还要创建一个 vsadmin 帐户来管理 SVM 及其资源。



1. 以存储管理员身份登录 Red Hat OpenShift 集群。
2. 为 project-1 创建后端并将其映射到专用于该项目的 SVM。 NetApp建议使用 SVM 的 vsadmin 帐户将后端连接到 SVM，而不是使用ONTAP集群管理员。

```
cat << EOF | tridentctl -n trident create backend -f
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "nfs_project_1",
    "managementLIF": "172.21.224.210",
    "dataLIF": "10.61.181.224",
    "svm": "project-1-svm",
    "username": "vsadmin",
    "password": "NetApp123"
}
EOF
```

我们在此示例中使用 ontap-nas 驱动程序。根据用例创建后端时使用适当的驱动程序。

我们假设Trident已安装在 trident 项目中。

1. 类似地为 project-2 创建Trident后端并将其映射到专用于 project-2 的 SVM。
2. 接下来，创建存储类。为 project-1 创建存储类，并通过设置 storagePools 参数将其配置为使用专用于 project-1 的后端存储池。

```
cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: project-1-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
  storagePools: "nfs_project_1:.*"
EOF
```

3. 同样，为 project-2 创建一个存储类并将其配置为使用专用于 project-2 的后端存储池。
4. 创建 ResourceQuota 来限制 project-1 中的资源从专用于其他项目的存储类中请求存储。

```
cat << EOF | oc create -f -
kind: ResourceQuota
apiVersion: v1
metadata:
  name: project-1-sc-rq
  namespace: project-1
spec:
  hard:
    project-2-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
EOF
```

5. 类似地，创建一个 ResourceQuota 来限制 project-2 中的资源从专用于其他项目的存储类中请求存储。

验证

要验证前面步骤中配置的多租户架构，请完成以下步骤：

验证在指定项目中创建 **PVC** 或 **Pod** 的权限

1. 以 ocp-project-1-user、project-1 中的开发人员身份登录。
2. 检查创建新项目的权限。

```
oc create ns sub-project-1
```

3. 使用分配给 project-1 的存储类在 project-1 中创建 PVC。

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-1
  namespace: project-1
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-1-sc
EOF
```

4. 检查与 PVC 关联的 PV。

```
oc get pv
```

5. 验证 PV 及其卷是否在NetApp ONTAP上专用于 project-1 的 SVM 中创建。

```
volume show -vserver project-1-svm
```

6. 在 project-1 中创建一个 pod，并挂载上一步创建的 PVC。

```
cat << EOF | oc create -f -
kind: Pod
apiVersion: v1
metadata:
  name: test-pvc-pod
  namespace: project-1
spec:
  volumes:
    - name: test-pvc-project-1
      persistentVolumeClaim:
       claimName: test-pvc-project-1
  containers:
    - name: test-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: test-pvc-project-1
EOF
```

7. 检查 pod 是否正在运行以及是否安装了卷。

```
oc describe pods test-pvc-pod -n project-1
```

验证在另一个项目中创建 **PVC** 或 **Pod** 或使用专用于另一个项目的资源的访问权限

1. 以 ocp-project-1-user、project-1 中的开发人员身份登录。
2. 使用分配给 project-2 的存储类在 project-1 中创建 PVC。

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-1-sc-2
  namespace: project-1
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-2-sc
EOF
```

3. 在 project-2 中创建 PVC。

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-2-sc-1
  namespace: project-2
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-1-sc
EOF
```

4. 确保 PVC `test-pvc-project-1-sc-2`和 `test-pvc-project-2-sc-1`沒有創造。

```
oc get pvc -n project-1
oc get pvc -n project-2
```

5. 在 project-2 中创建一个 pod。

```
cat << EOF | oc create -f -
kind: Pod
apiVersion: v1
metadata:
  name: test-pvc-pod
  namespace: project-1
spec:
  containers:
    - name: test-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
EOF
```

验证查看和编辑项目、资源配额和存储类的权限

1. 以 ocp-project-1-user、project-1 中的开发人员身份登录。
2. 检查创建新项目的权限。

```
oc create ns sub-project-1
```

3. 验证查看项目的访问权限。

```
oc get ns
```

4. 检查用户是否可以查看或编辑 project-1 中的 ResourceQuotas。

```
oc get resourcequotas -n project-1
oc edit resourcequotas project-1-sc-rq -n project-1
```

5. 验证用户是否有权查看存储类别。

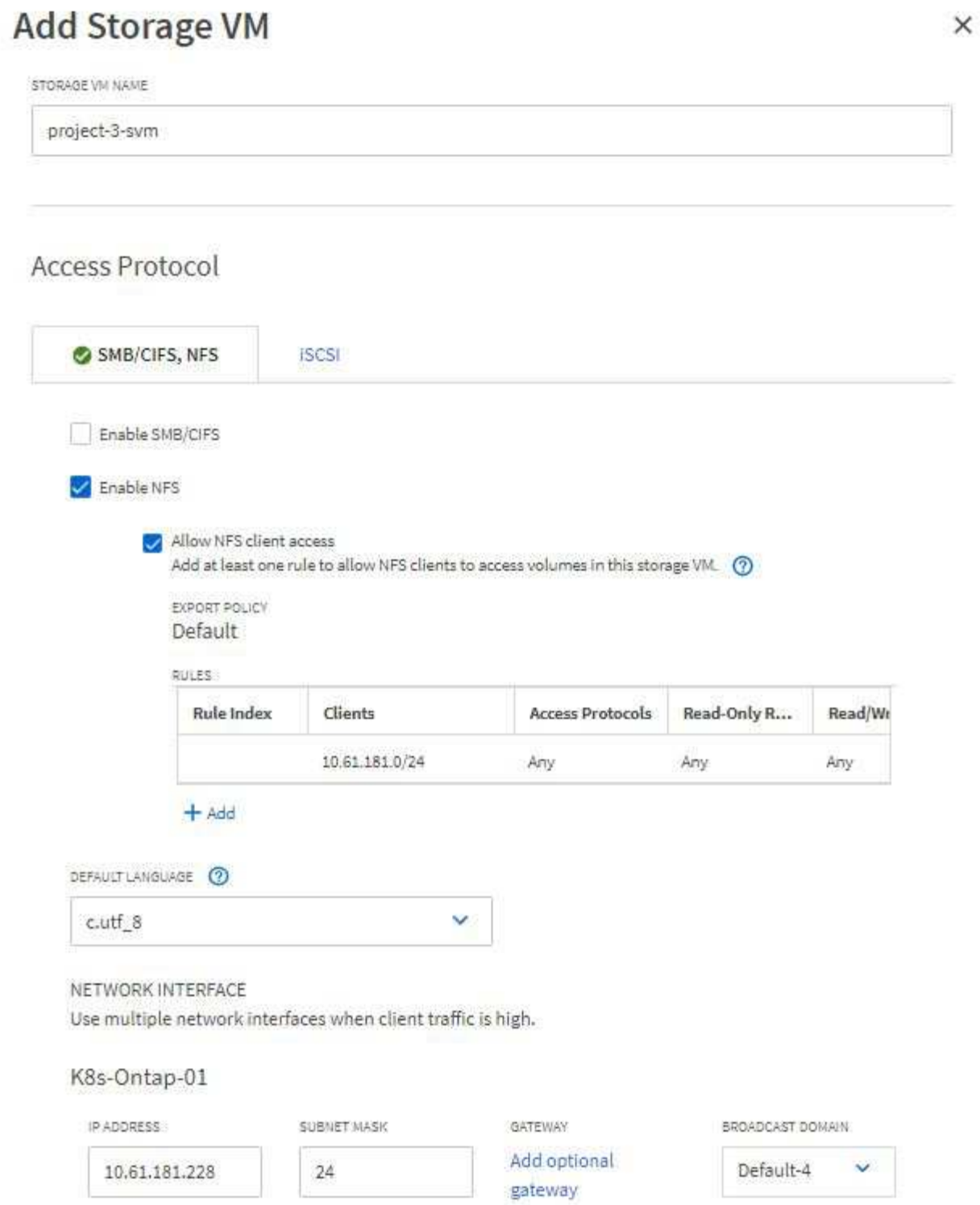
```
oc get sc
```

6. 检查访问权限以描述存储类别。
7. 验证用户编辑存储类的权限。

```
oc edit sc project-1-sc
```

在多租户配置中，添加具有存储资源的新项目需要进行额外的配置以确保不违反多租户原则。要在多租户集群中添加更多项目，请完成以下步骤：

1. 以存储管理员身份登录NetApp ONTAP集群。
2. 导航至 `Storage → Storage VMs`并点击 `Add。创建一个专用于 project-3 的新 SVM。还要创建一个 vsadmin 帐户来管理 SVM 及其资源。



1. 以集群管理员身份登录 Red Hat OpenShift 集群。
2. 创建新项目。

```
oc create ns project-3
```

3. 确保在 IdP 上创建了 project-3 的用户组并与 OpenShift 集群同步。

```
oc get groups
```

4. 为 project-3 创建开发人员角色。

```
cat << EOF | oc create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: project-3
  name: developer-project-3
rules:
  - verbs:
      - '*'
    apiGroups:
      - apps
      - batch
      - autoscaling
      - extensions
      - networking.k8s.io
      - policy
      - apps.openshift.io
      - build.openshift.io
      - image.openshift.io
      - ingress.operator.openshift.io
      - route.openshift.io
      - snapshot.storage.k8s.io
      - template.openshift.io
    resources:
      - '*'
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - bindings
      - configmaps
      - endpoints
      - events
      - persistentvolumeclaims
      - pods
      - pods/log
      - pods/attach
      - podtemplates
```

```
      - replicationcontrollers
      - services
      - limitranges
      - namespaces
      - componentstatuses
      - nodes
  - verbs:
      - '*'
    apiGroups:
      - trident.netapp.io
    resources:
      - tridentsnapshots
EOF
```

本节提供的角色定义仅仅是一个示例。必须根据最终用户的要求来定义开发人员的角色。

1. 为 project-3 中的开发人员创建 RoleBinding，将 development-project-3 角色绑定到 project-3 中相应的组（ocp-project-3）。

```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-3-developer
  namespace: project-3
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-3
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-project-3
EOF
```

2. 以存储管理员身份登录 Red Hat OpenShift 集群
3. 创建Trident后端并将其映射到专用于 project-3 的 SVM。 NetApp建议使用 SVM 的 vsadmin 帐户将后端连接到 SVM，而不是使用ONTAP集群管理员。

```
cat << EOF | tridentctl -n trident create backend -f
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "nfs_project_3",
    "managementLIF": "172.21.224.210",
    "dataLIF": "10.61.181.228",
    "svm": "project-3-svm",
    "username": "vsadmin",
    "password": "NetApp!23"
}
EOF
```

我们在此示例中使用 ontap-nas 驱动程序。根据用例使用适当的驱动程序创建后端。

我们假设Trident已安装在 trident 项目中。

1. 为 project-3 创建存储类并将其配置为使用专用于 project-3 的后端存储池。

```
cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: project-3-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
  storagePools: "nfs_project_3:.*"
EOF
```

2. 创建 ResourceQuota 来限制 project-3 中的资源从专用于其他项目的存储类中请求存储。

```
cat << EOF | oc create -f -
kind: ResourceQuota
apiVersion: v1
metadata:
  name: project-3-sc-rq
  namespace: project-3
spec:
  hard:
    project-1-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
    project-2-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
EOF
```

3. 修补其他项目中的 ResourceQuotas，以限制这些项目中的资源访问专用于 project-3 的存储类的存储。

```
oc patch resourcequotas project-1-sc-rq -n project-1 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
oc patch resourcequotas project-2-sc-rq -n project-2 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
```

Kubernetes 的高级集群管理

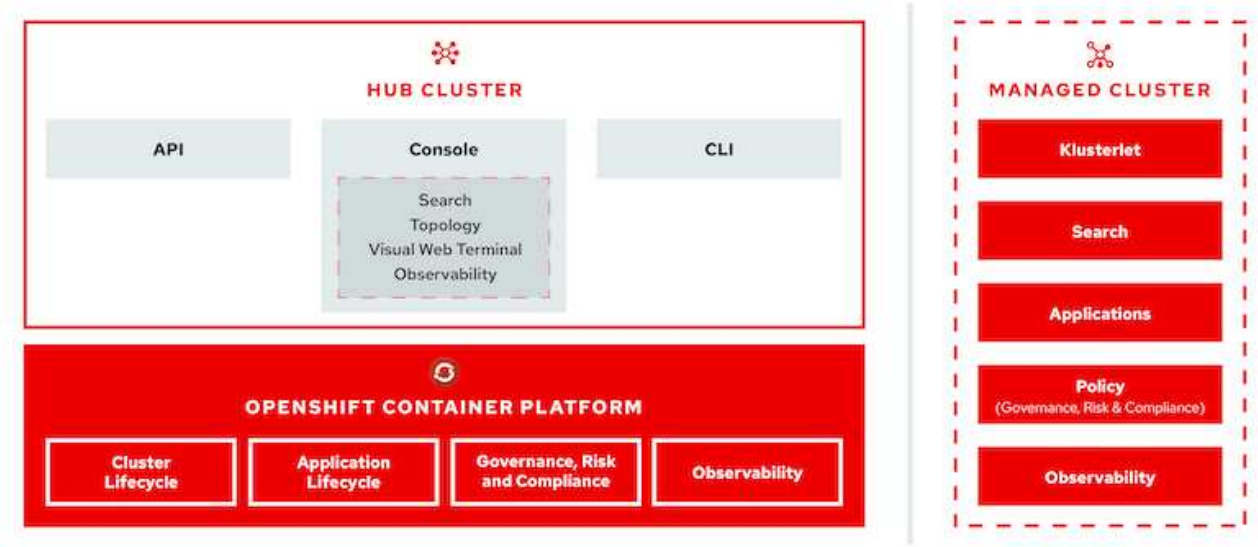
Kubernetes 的高级集群管理：Red Hat OpenShift 与NetApp - 概述

随着容器化应用程序从开发过渡到生产，许多组织需要多个 Red Hat OpenShift 集群来支持该应用程序的测试和部署。与此同时，组织通常在 OpenShift 集群上托管多个应用程序或工作负载。因此，每个组织最终都会管理一组集群，而 OpenShift 管理员必须面对额外的挑战，即管理和维护跨多个本地数据中心和公共云的一系列环境中的多个集群。为了应对这些挑战，Red Hat 推出了针对 Kubernetes 的高级集群管理。

Red Hat Advanced Cluster Management for Kubernetes 使您能够执行以下任务：

1. 跨数据中心和公共云创建、导入和管理多个集群
2. 从单个控制台部署和管理多个集群上的应用程序或工作负载
3. 监控和分析不同集群资源的健康和状态
4. 监控并强制执行跨多个集群的安全合规性

Red Hat Advanced Cluster Management for Kubernetes 作为 Red Hat OpenShift 集群的附加组件安装，并使用该集群作为其所有操作的中央控制器。该集群称为中心集群，它为用户公开了一个管理平面以连接到高级集群管理。通过高级集群管理控制台导入或创建的所有其他 OpenShift 集群均由中心集群管理，并称为托管集群。它在托管集群上安装了一个名为 Klusterlet 的代理，将它们连接到中心集群，并满足与集群生命周期管理、应用程序生命周期管理、可观察性和安全合规性相关的不同活动的请求。



有关详细信息，请参阅文档 "此处"。

为 Kubernetes 部署 ACM

为 Kubernetes 部署高级集群管理

本节介绍使用NetApp在 Red Hat OpenShift 上对 Kubernetes 进行高级集群管理。

前提条件

1. 用于中心集群的 Red Hat OpenShift 集群（高于 4.5 版本）
2. 用于托管集群的 Red Hat OpenShift 集群（高于 4.4.3 版本）
3. 集群管理员访问 Red Hat OpenShift 集群
4. Red Hat 订阅 Kubernetes 高级集群管理

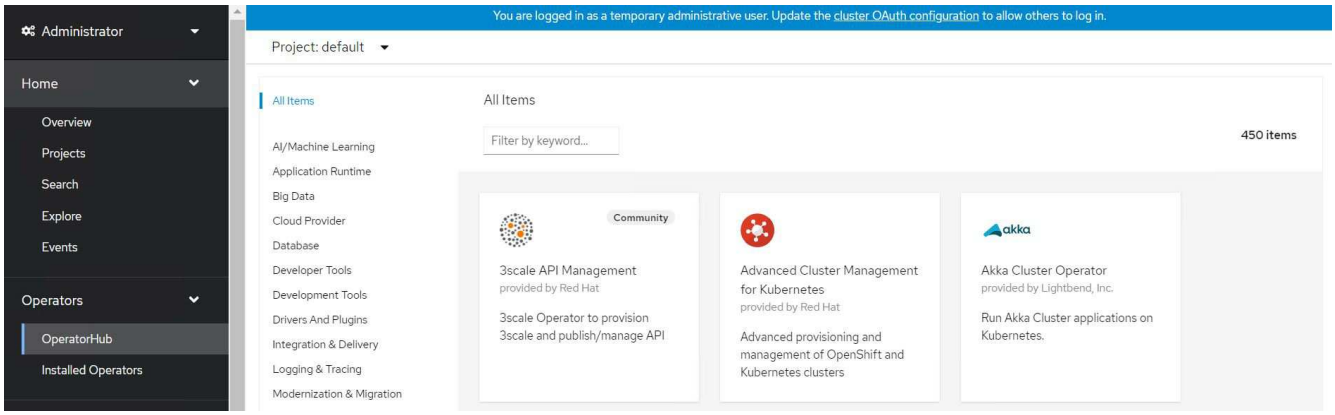
高级集群管理是 OpenShift 集群的一个附加组件，因此根据中心和管理集群中使用的功能，对硬件资源有一定的要求和限制。在确定集群大小时需要考虑这些问题。查看文档 "此处"了解更多详情。

或者，如果中心集群具有用于托管基础设施组件的专用节点，并且您只想在这些节点上安装高级集群管理资源，则需要相应地向这些节点添加容忍度和选择器。更多详细信息请参阅文档 "此处"。

为 Kubernetes 部署高级集群管理

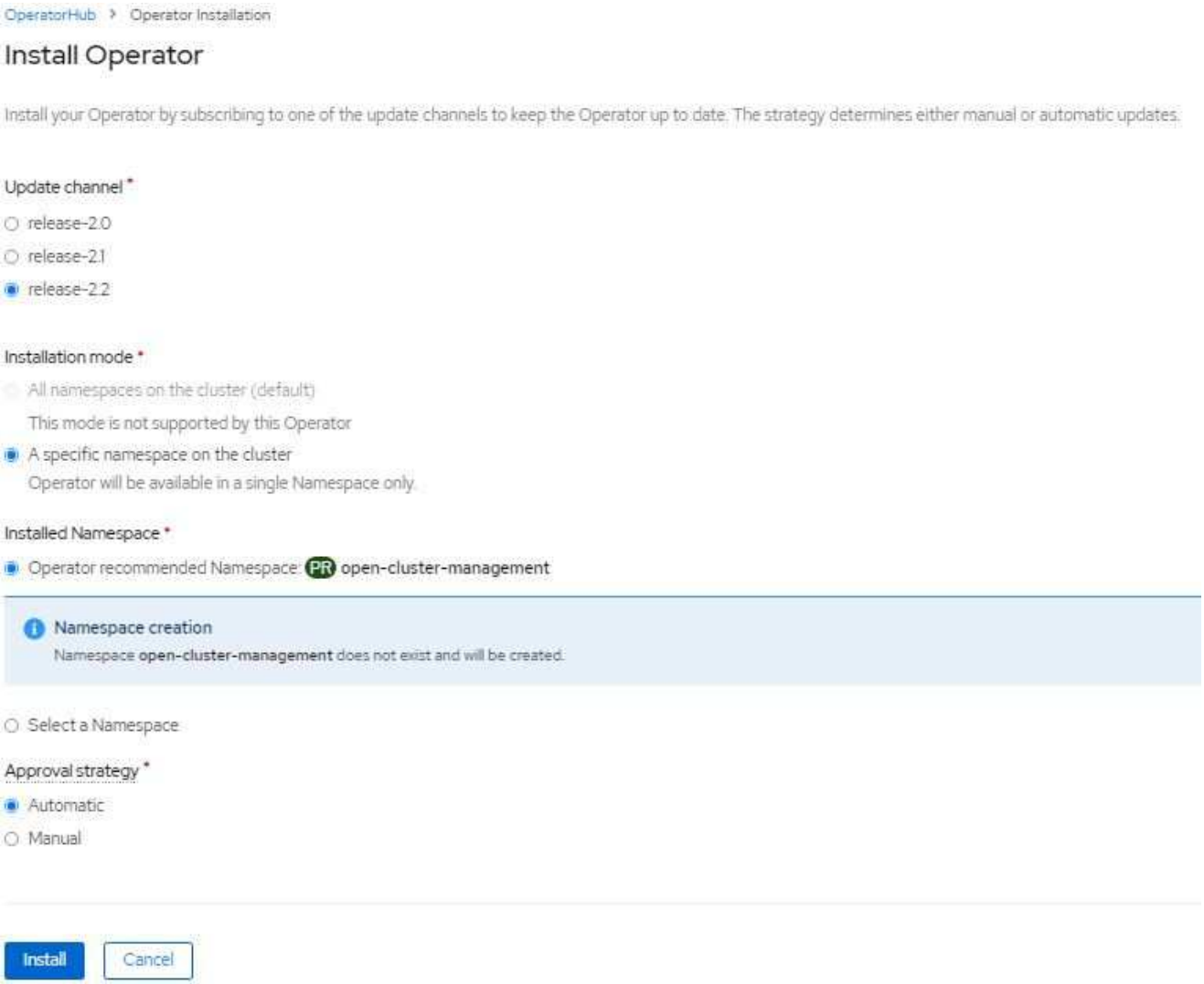
要在 OpenShift 集群上安装 Kubernetes 高级集群管理，请完成以下步骤：

1. 选择一个 OpenShift 集群作为中心集群，并使用集群管理员权限登录。
2. 导航到 Operators > Operators Hub 并搜索 Kubernetes 的高级集群管理。

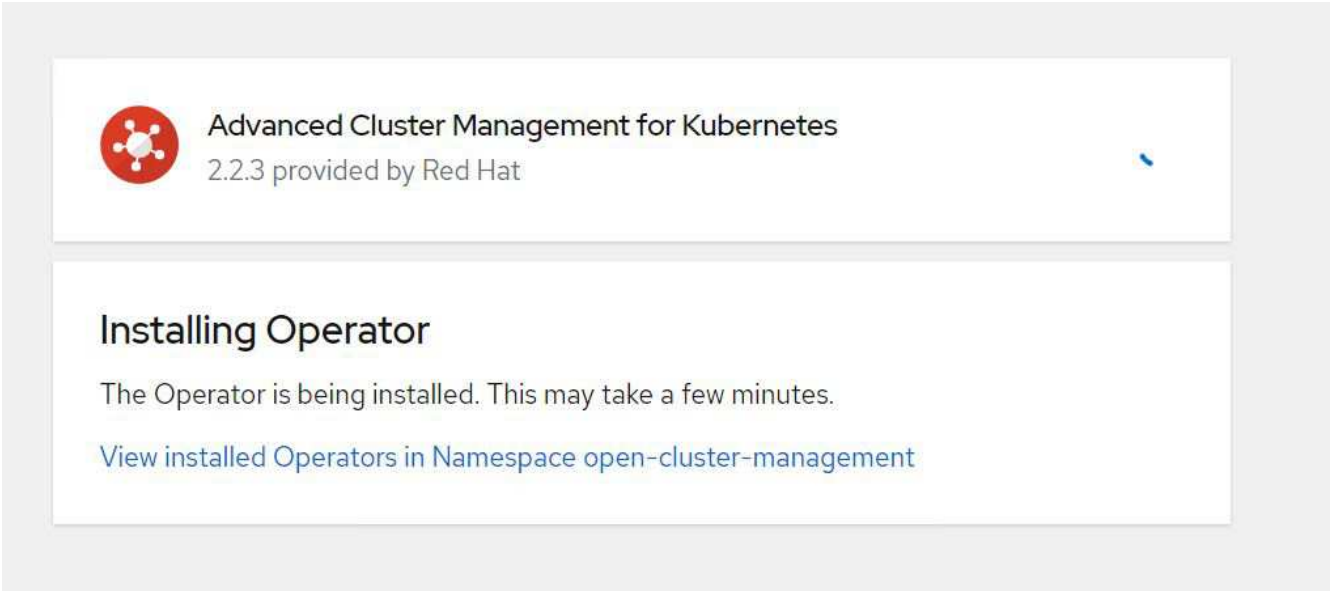


3. 选择 Kubernetes 的高级集群管理，然后单击安装。

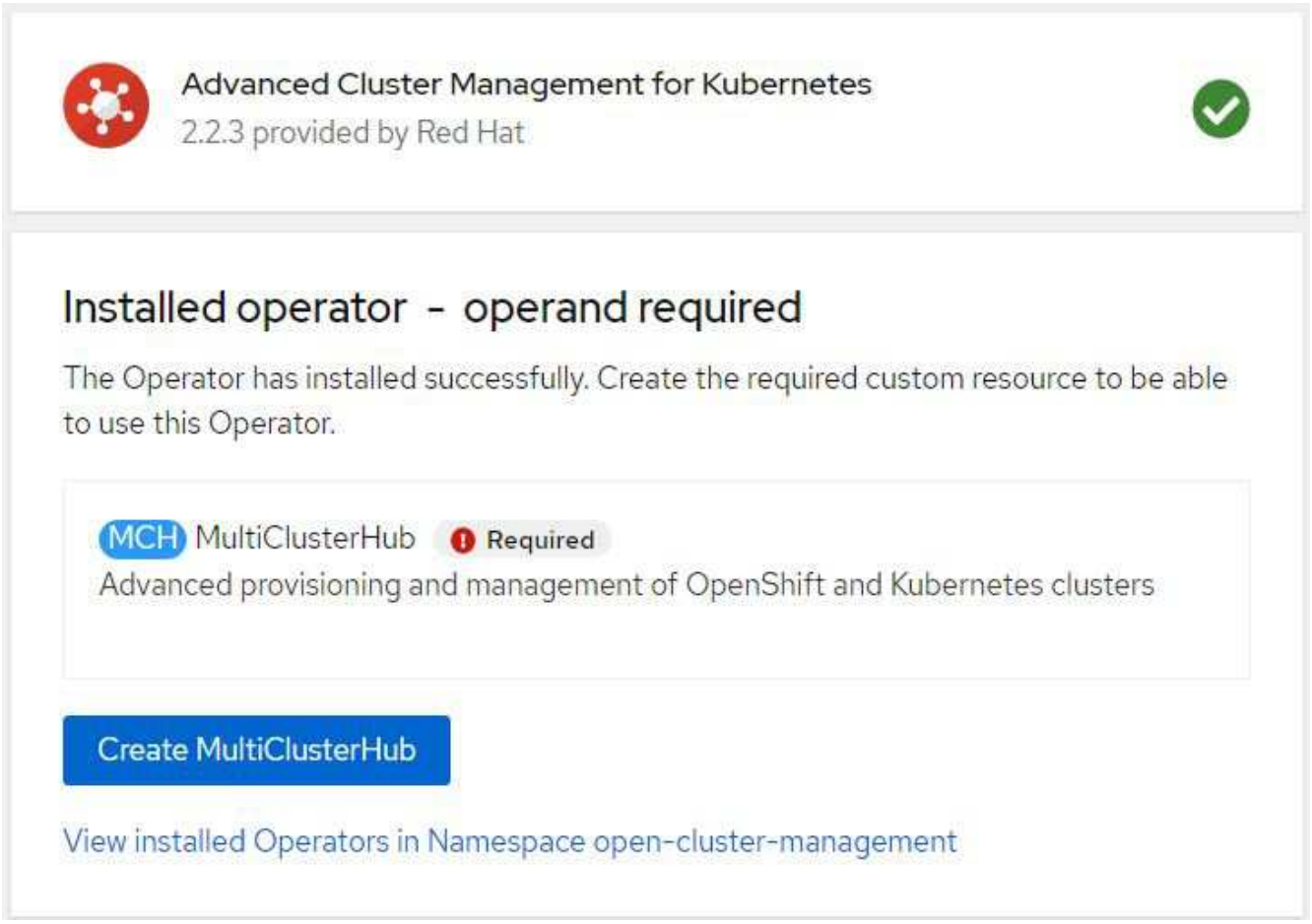
4. 在“安装操作员”屏幕上，提供必要的详细信息（NetApp建议保留默认参数）并单击“安装”。



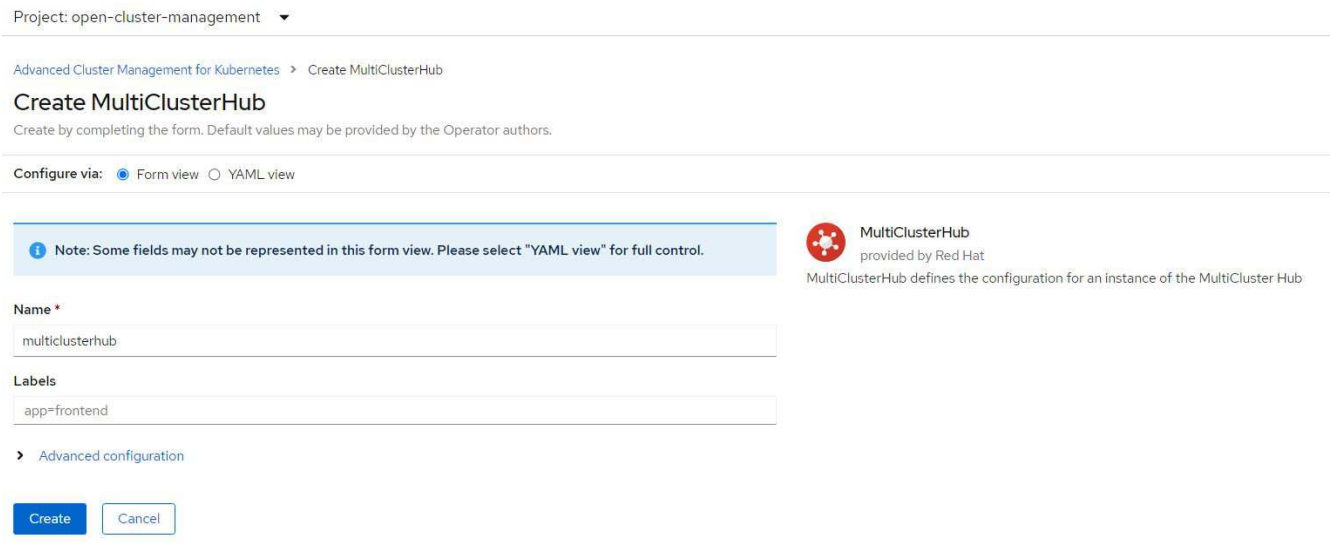
5. 等待操作员安装完成。



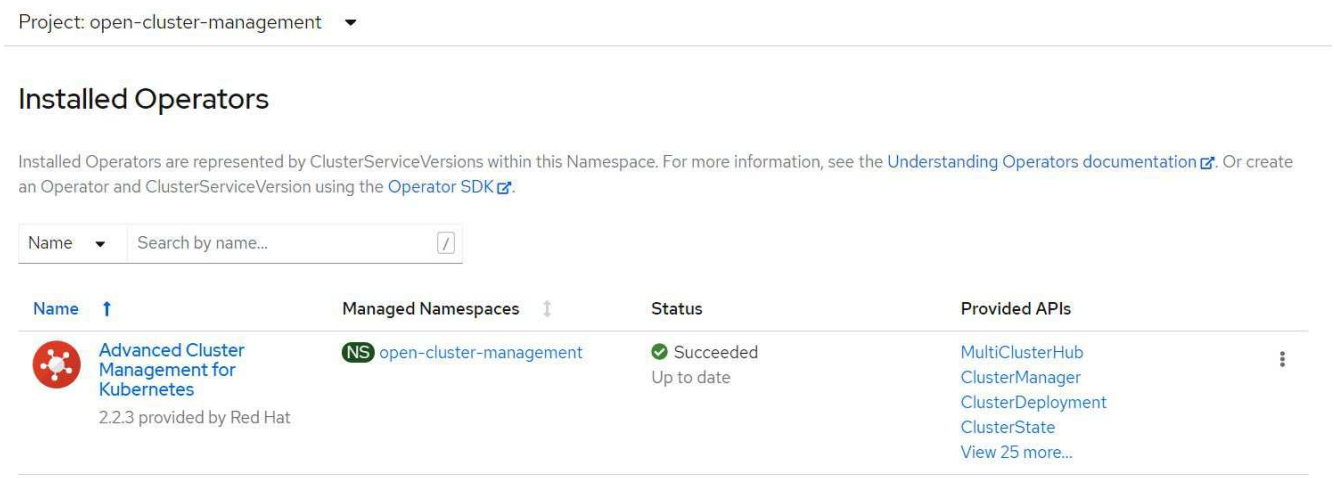
6. 安装操作员后，单击创建 MultiClusterHub。



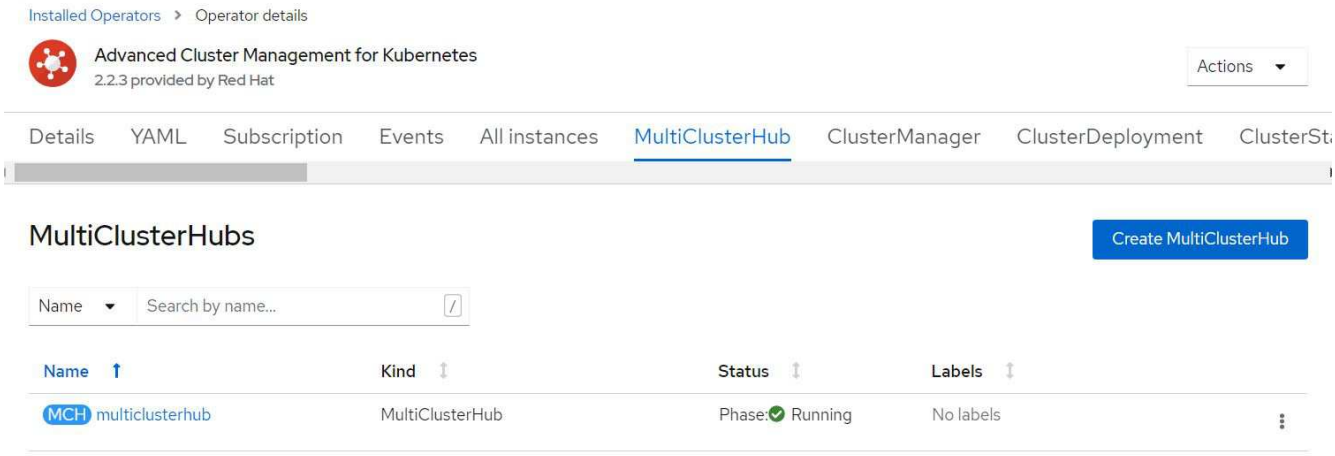
7. 在创建 MultiClusterHub 屏幕上，提供详细信息后单击创建。这将启动多集群集线器的安装。



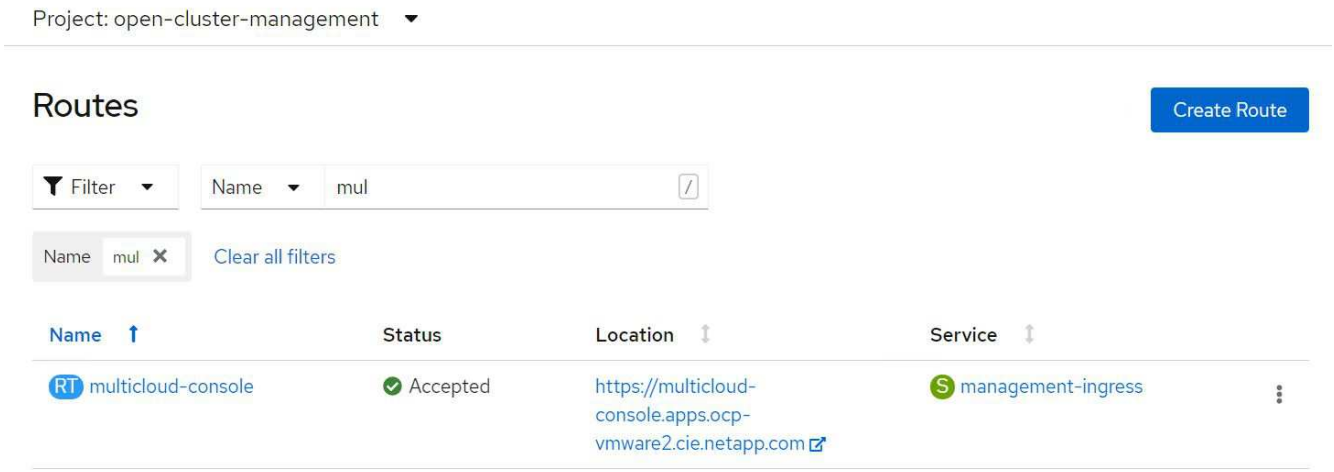
8. 当 open-cluster-management 命名空间中的所有 pod 都转为 Running 状态，并且操作员转为 Succeeded 状态后，Kubernetes 高级集群管理就安装完成了。



9. 完成集线器安装需要一些时间，完成后，多集群集线器将进入运行状态。

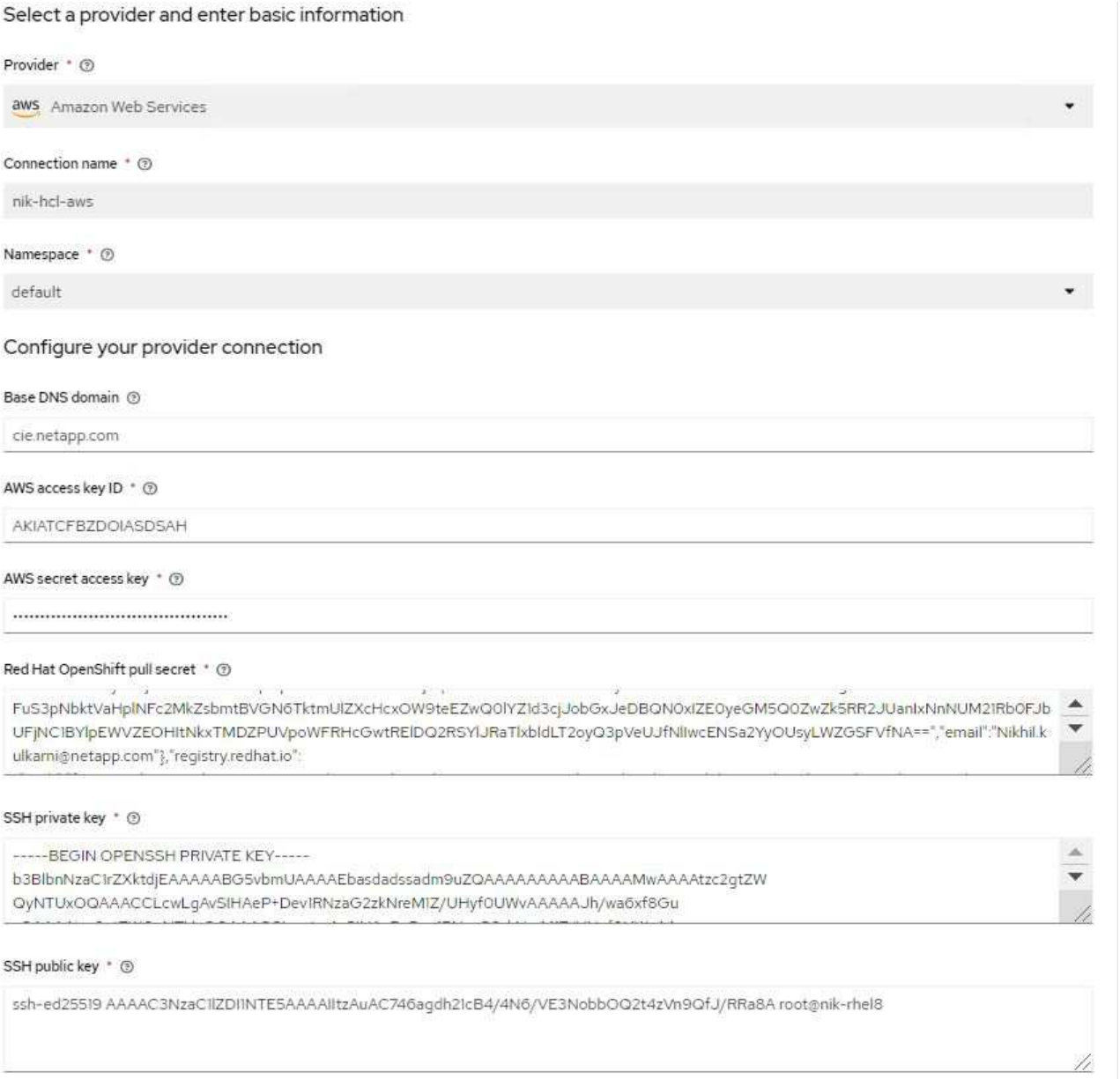


10. 它在 open-cluster-management 命名空间中创建一条路由。连接到路由中的URL，访问高级集群管理控制台。

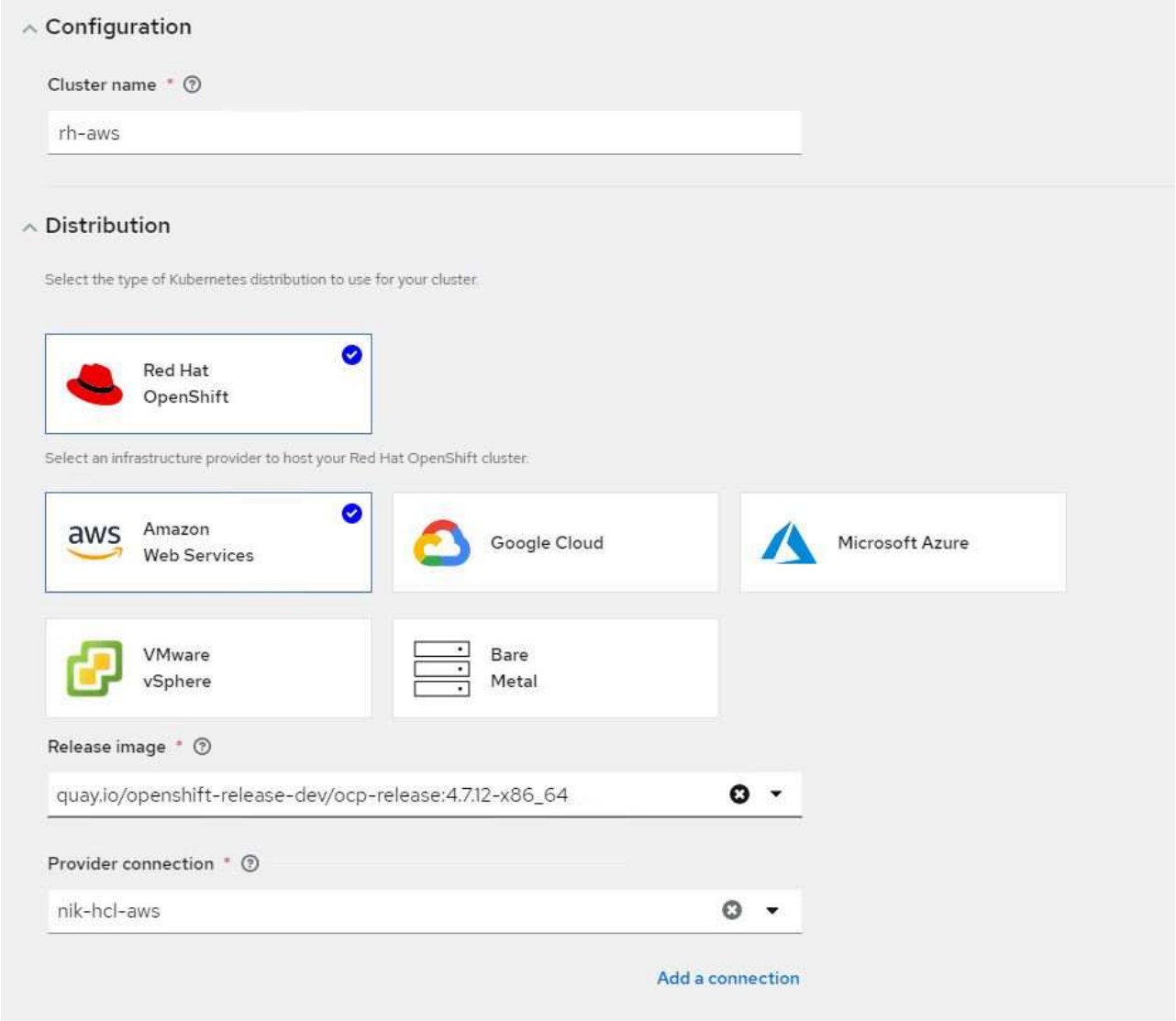


要管理不同的 OpenShift 集群，您可以创建它们或将它们导入高级集群管理。

1. 首先导航到自动化基础设施>集群。
2. 要创建新的 OpenShift 集群，请完成以下步骤：
   a. 创建提供商连接：导航到提供商连接并单击添加连接，提供与所选提供商类型相对应的所有详细信息，然后单击添加。



   b. 要创建新集群，请导航至“集群”，然后单击“添加集群”>“创建集群”。提供集群和相应提供程序的详细信息，然后单击“创建”。



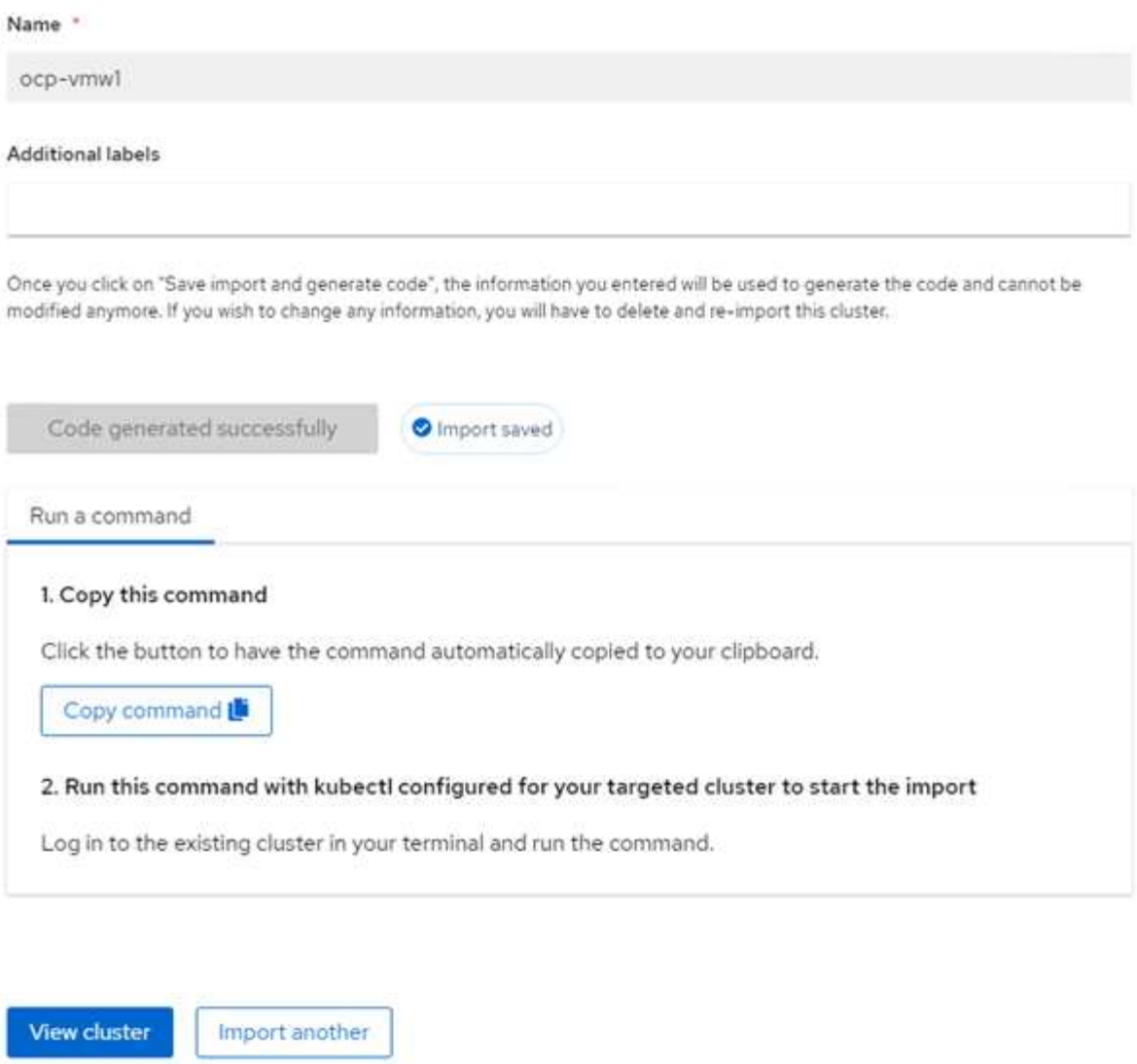
c. 集群创建完成后，出现在集群列表中，状态为Ready。

3. 要导入现有集群，请完成以下步骤：

a. 导航到集群并单击添加集群>导入现有集群。

b. 输入集群名称，点击保存导入并生成代码。显示添加现有集群的命令。

c. 单击“复制命令”，在要添加到中心集群的集群上运行该命令。这将启动集群上必要代理的安装，并且此过程完成后，集群将以“就绪”状态出现在集群列表中。

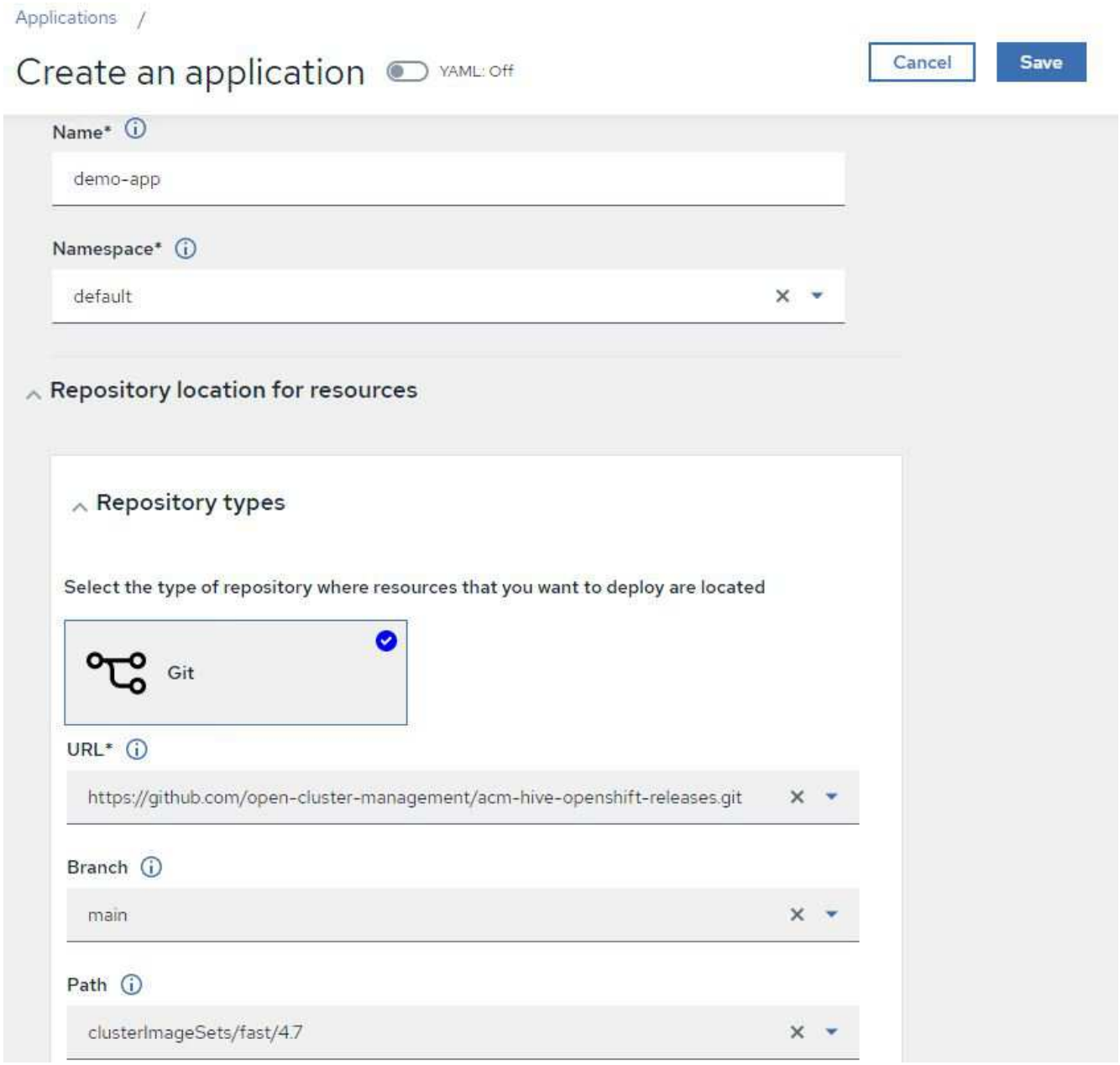


4. 创建并导入多个集群后，您可以从单个控制台监控和管理它们。

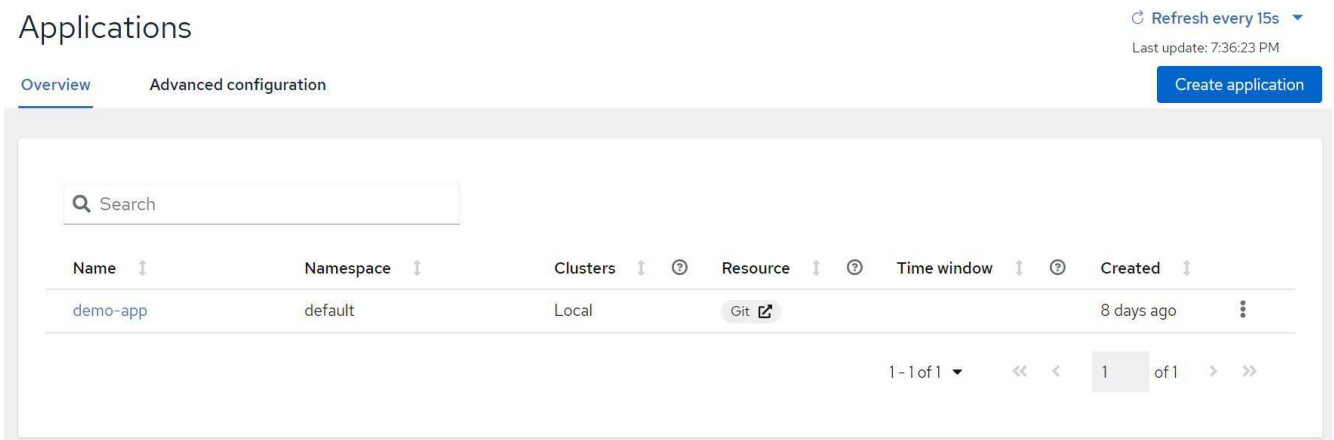
应用程序生命周期管理

要创建一个应用程序并跨一组集群管理它，

1. 从侧边栏导航到“管理应用程序”，然后单击“创建应用程序”。提供您想要创建的应用程序的详细信息，然后单击“保存”。



2. 应用程序组件安装完成后，该应用程序就会出现在列表中。

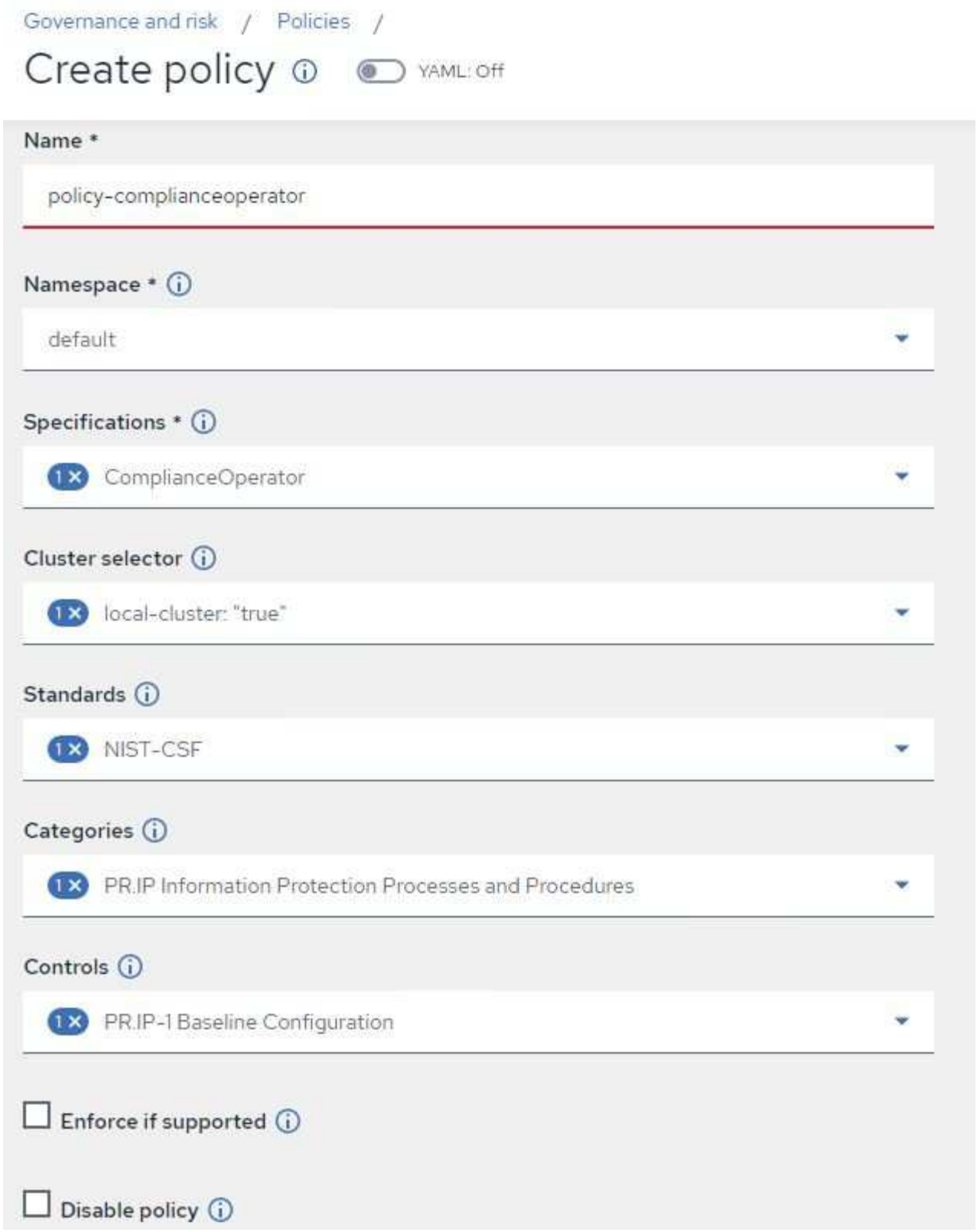


3. 现在可以从控制台监控和管理该应用程序。

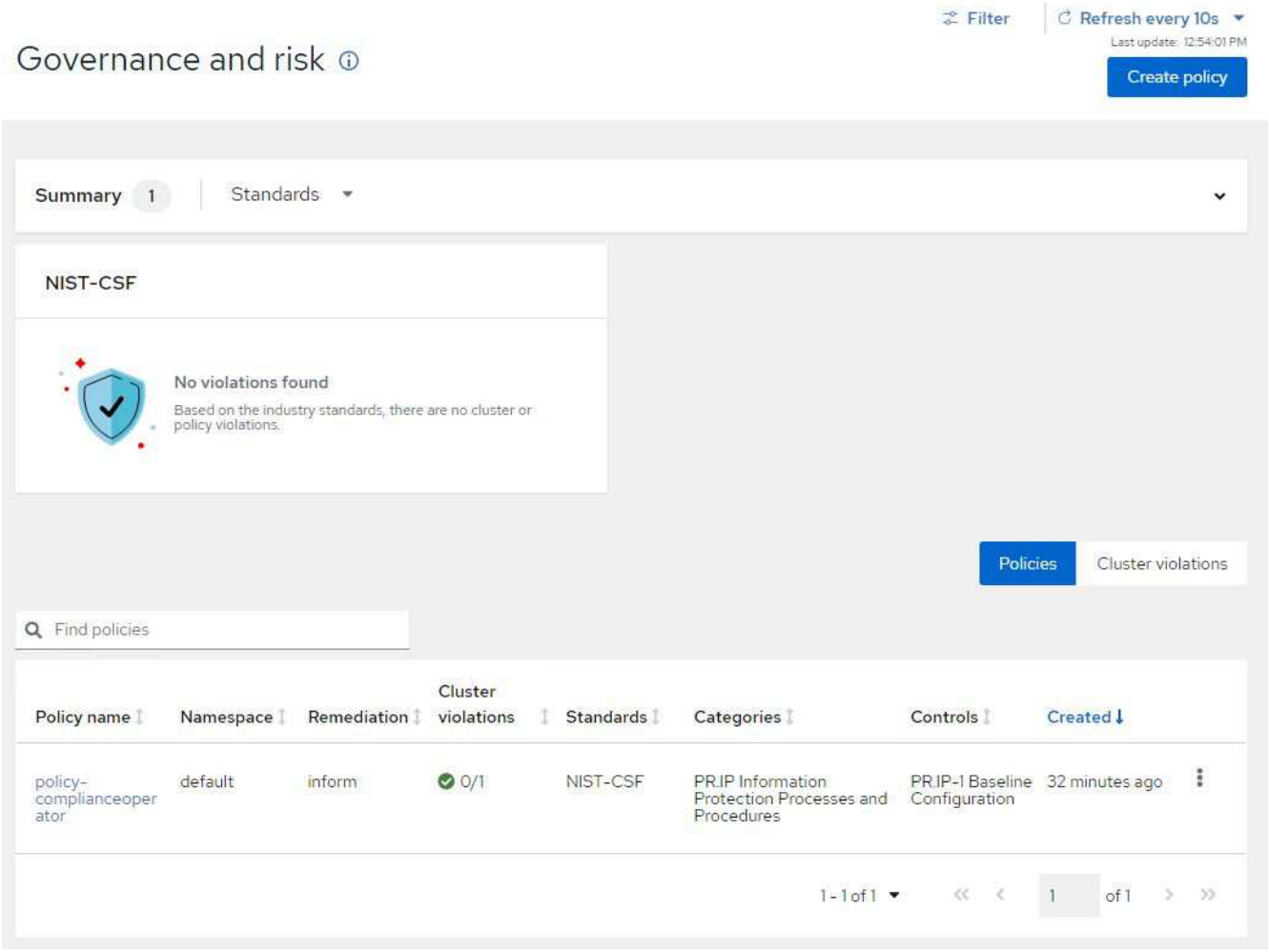
治理与风险

此功能允许您为不同的集群定义合规策略并确保集群遵守该策略。您可以配置策略来通知或补救任何偏离或违反规则的行为。

1. 从侧边栏导航至“治理和风险”。
2. 要创建合规性策略，请单击创建策略，输入策略标准的详细信息，然后选择应遵守此策略的集群。如果您想自动纠正违反此策略的行为，请选中“如果支持则强制执行”复选框，然后单击“创建”。



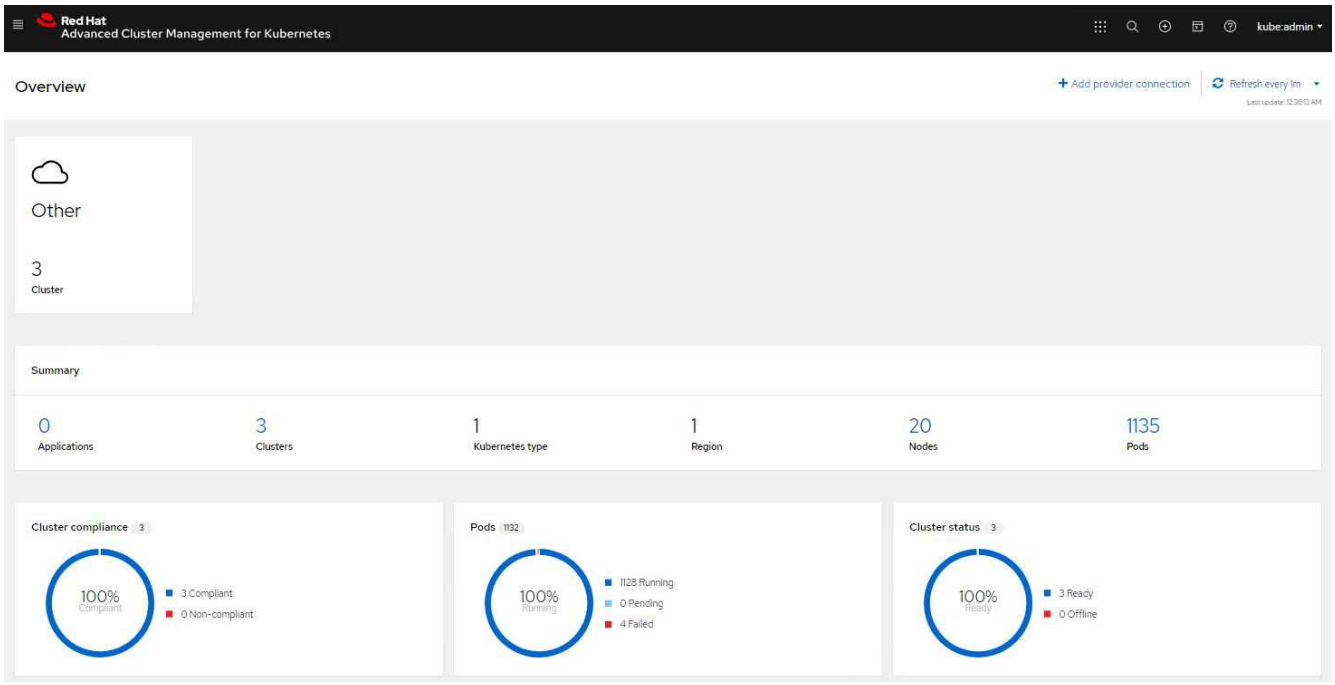
3. 配置所有必需的策略后，可以从高级集群管理中监控和修复任何策略或集群违规行为。



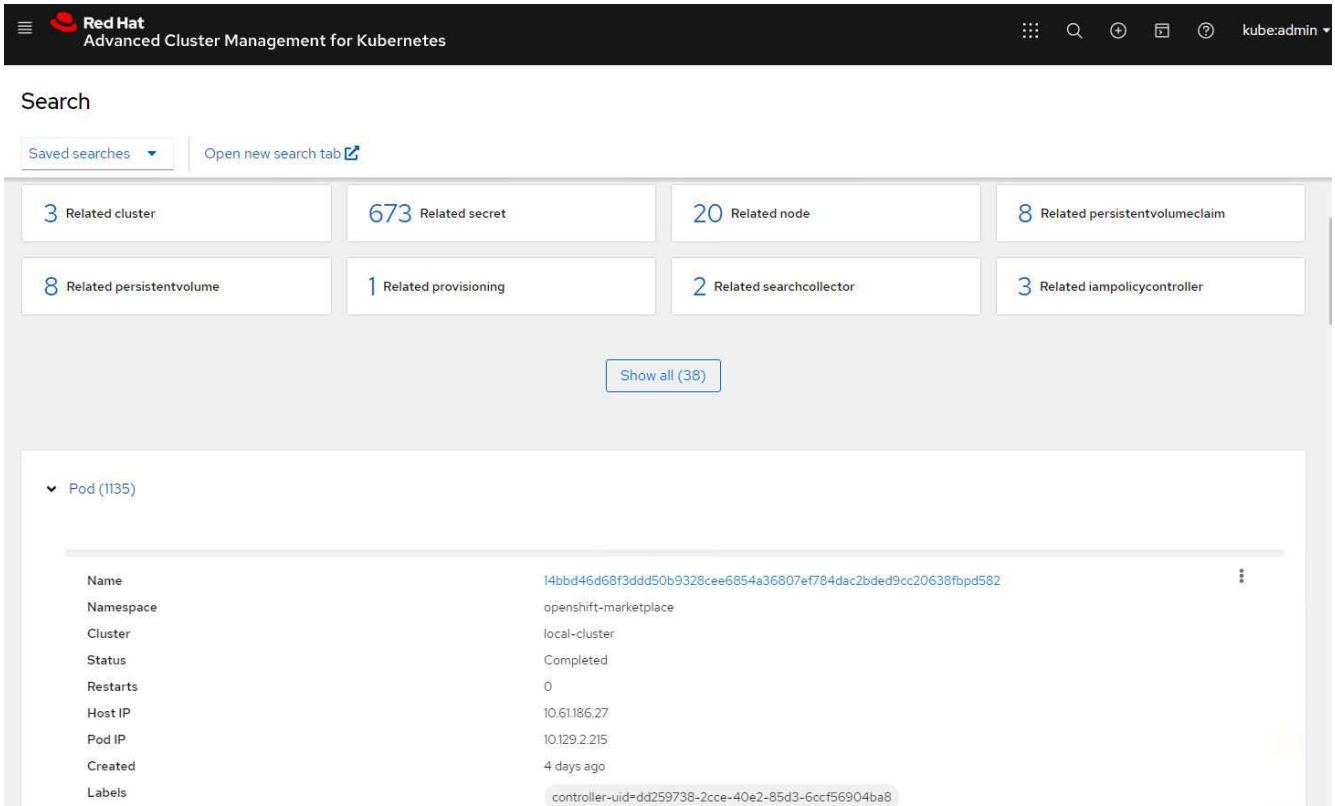
可观察性

Kubernetes 的高级集群管理提供了一种监控所有集群中的节点、pod、应用程序和工作负载的方法。

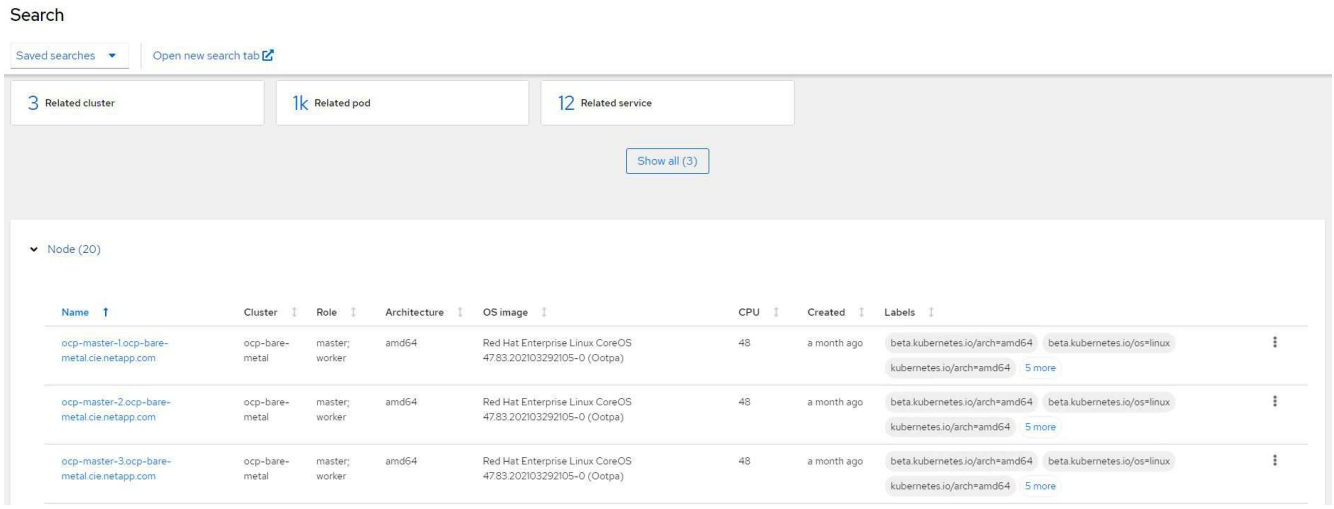
1. 导航至观察环境 > 概览。



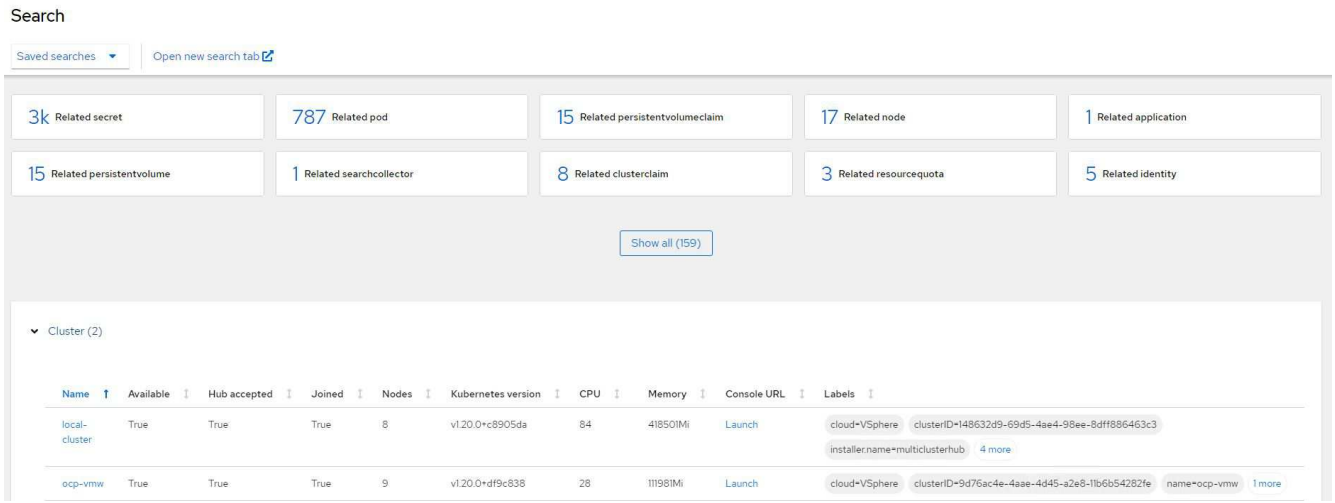
2. 所有集群中的所有 pod 和工作负载都根据各种过滤器进行监控和排序。点击Pods即可查看相应数据。



3. 基于各种数据点对集群中的所有节点进行监控和分析。单击节点可以深入了解相应的详细信息。

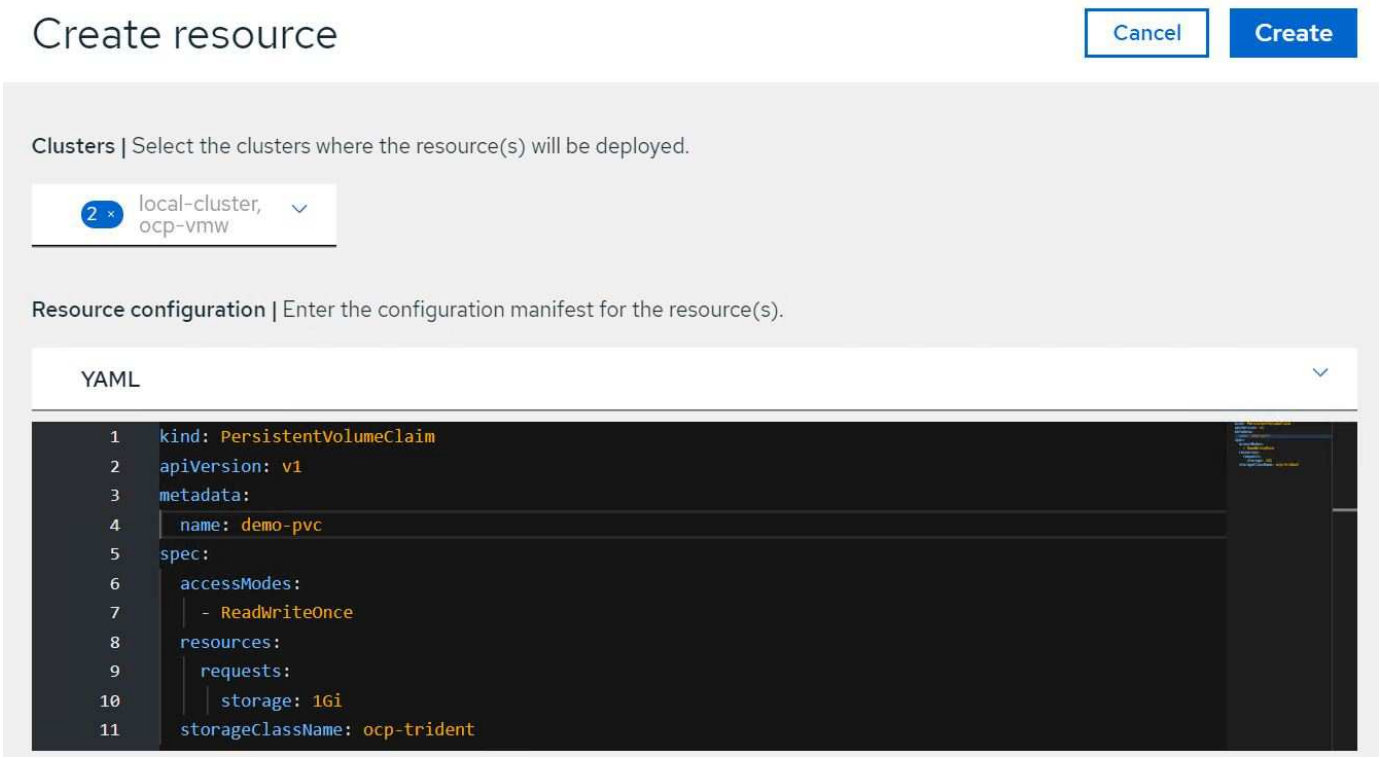


4. 所有集群都根据不同的集群资源和参数进行监控和组织。单击“集群”可查看集群详细信息。



在多个集群上创建资源

Kubernetes 的高级集群管理允许用户从控制台同时在一个或多个托管集群上创建资源。例如，如果您在不同的站点拥有 OpenShift 集群，且这些集群由不同的NetApp ONTAP集群支持，并且想要在两个站点配置 PVC，则可以单击顶部栏上的 (+) 号。然后选择要创建 PVC 的集群，粘贴资源 YAML，然后单击创建。



使用Trident Protect 为容器应用和虚拟机提供数据保护

该解决方案展示了如何使用Trident Protect 对容器和虚拟机执行数据保护操作。

1. 有关在 OpenShift Container 平台中为容器应用程序创建快照和备份以及从中恢复的详细信息，请参阅"此处"。
2. 有关在 OpenShift Container 平台上部署的 OpenShift Virtualization 中创建和恢复虚拟机备份的详细信息，请参阅"此处"。

使用第三方工具对容器应用程序和虚拟机进行数据保护

该解决方案展示了如何使用与 Red Hat OpenShift Container 平台中的 OADP 操作员集成的 Velero 对容器和虚拟机执行数据保护操作。

1. 有关在 OpenShift Container 平台中创建和恢复容器应用程序备份的详细信息，请参阅"此处"。
2. 有关在 OpenShift Container 平台上部署的 OpenShift Virtualization 中创建和恢复虚拟机备份的详细信息，请参阅"此处"。

了解有关 Red Hat OpenShift 虚拟化与NetApp存储集成的其他资源

访问其他资源，这些资源提供有关在各种平台和技术上支持使用ONTAP部署、管理和优化 Red Hat OpenShift Virtualization 的更多信息。

- NetApp 文档

"https://docs.netapp.com/"

- Trident文档

  "https://docs.netapp.com/us-en/trident/index.html"

- Red Hat OpenShift 文档

  "https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/"

- Red Hat OpenStack 平台文档

  "https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/"

- Red Hat 虚拟化文档

  "https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.4/"

- VMware vSphere 文档

  "https://docs.vmware.com/"