采用 NetApp 技术的 Red Hat OpenShift

NetApp Solutions

NetApp
April 12, 2024

目录

NVA-1160： 采用 NetApp 技术的 Red Hat OpenShift

NetApp 公司 Alan Cowles 和 Nikhil M Kulkarni

本参考文档对通过安装程序配置的基础架构（ IPI ）在 NetApp 验证的多种不同数据中心环境中部署的 Red Hat OpenShift 解决方案进行了部署验证。同时，还详细介绍了如何利用 Astra Trident 存储编排程序来管理永久性存储，从而与 NetApp 存储系统实现存储集成。最后，我们还探讨并记录了许多解决方案验证和实际使用情形。

用例

采用 NetApp 解决方案的 Red Hat OpenShift 旨在为客户提供卓越的价值，其使用情形如下：

- 使用 IPI （安装程序配置的基础架构）在裸机上， Red Hat OpenStack Platform ， Red Hat 虚拟化和 VMware vSphere 上轻松部署和管理 Red Hat OpenShift 。
- 将企业级容器和虚拟化工作负载的强大功能与 Red Hat OpenShift 相结合， Red Hat OpenShift 可虚拟部署在 OSP ， RHV 或 vSphere 上，也可通过 OpenShift 虚拟化部署在裸机上。
- 与 NetApp 存储和适用于 Kubernetes 的开源存储编排程序 Astra Trident 结合使用时，重点介绍 Red Hat OpenShift 的功能的实际配置和使用情形。

业务价值

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。由于容器和微服务本身的灵活性，它们在支持 DevOps 实践方面发挥着至关重要的作用。但是，在企业环境中以生产规模实施 DevOps 会带来自身的挑战，并对底层基础架构提出一些要求，例如：

- 堆栈中所有层的高可用性
- 易于部署过程
- 无中断运行和升级
- API 驱动的可编程基础架构，可跟上微服务灵活性的步伐
- 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架构

Red Hat OpenShift with NetApp 认可这些挑战，并提供了一个解决方案，通过在客户选择的数据中心环境中实施完全自动化的 RedHat OpenShift IPI 部署，帮助解决每个问题。

技术概述

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

Red Hat OpenShift 容器平台

Red Hat OpenShift 容器平台是一个完全受支持的企业 Kubernetes 平台。Red Hat 对开源 Kubernetes 进行了多

项增强，可提供一个应用程序平台，其中包含所有组件，这些组件均已完全集成，可用于构建，部署和管理容器化应用程序。

有关详细信息，请访问 OpenShift 网站 "此处"。

NetApp 存储系统

NetApp 拥有多个存储系统，非常适合企业数据中心和混合云部署。NetApp 产品组合包括 NetApp ONTAP ，NetApp Element 和 NetApp E 系列存储系统，所有这些系统均可为容器化应用程序提供永久性存储。

有关详细信息，请访问 NetApp 网站 "此处"。

NetApp 存储集成

NetApp Astra 控制中心为有状态 Kubernetes 工作负载提供了一组丰富的存储和应用程序感知型数据管理服务，这些服务部署在内部环境中，并采用值得信赖的 NetApp 数据保护技术。

有关详细信息，请访问 NetApp Astra 网站 "此处"。

Astra Trident 是一款开源且完全受支持的存储编排程序，适用于容器和 Kubernetes 分发版，包括 Red Hat OpenShift 。

有关详细信息，请访问 Astra 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环境中时可能需要执行的自定义设置，例如创建专用私有映像注册表或部署自定义负载平衡器实例。

已验证版本的当前支持列表

技术	目的	软件版本
NetApp ONTAP	存储	9.8 ， 9.9.1
NetApp Element	存储	12.3
NetApp Astra 控制中心	应用程序感知型数据管理	21.12.60
NetApp Astra Trident	存储编排	22.01.0
Red Hat OpenShift	容器编排	4.6 EUS ， 4.7 ， 4.8
Red Hat OpenStack 平台	私有云基础架构	16.1
Red Hat 虚拟化	数据中心虚拟化	4.4
VMware vSphere	数据中心虚拟化	6.7U3.

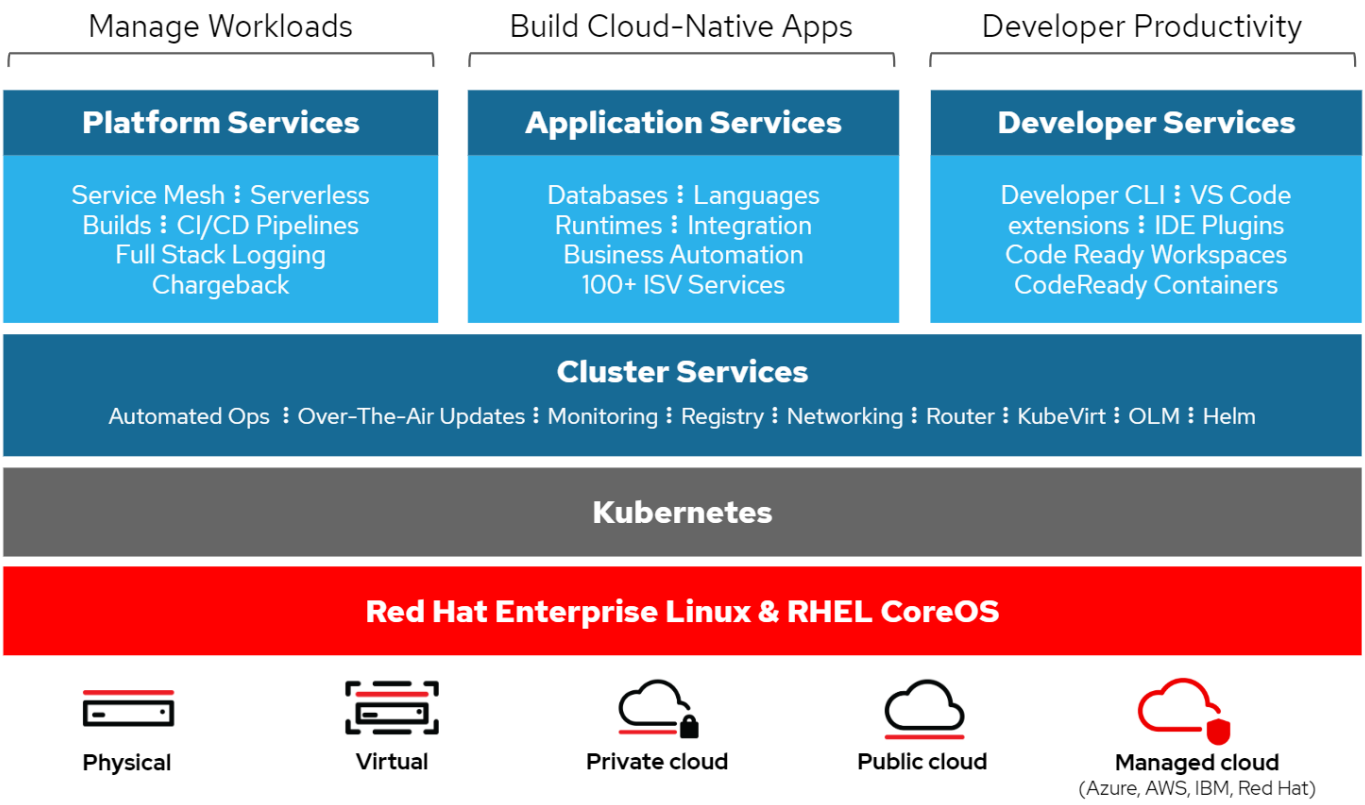
OpenShift 概述

Red Hat OpenShift 容器平台将开发和 IT 运营统一到一个平台上，以便在内部和混合云基础架构中一致地构建，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，其中包括 Kubernetes 和 Red Hat Enterprise Linux CoreOS ，这是全球领先的企业级 Linux 版本，专为基于容器的工作负载而设计。OpenShift 是 Cloud 原生计算基金会（ CNCF ）认证 Kubernetes 计划的一部分，可为容器工作负载提供可移植性和互操作性。

Red Hat OpenShift 提供以下功能：

- *自助式配置*开发人员可以使用最常用的工具快速轻松地按需创建应用程序，同时操作人员仍可完全控制整个环境。
- 永久性存储 OpenShift容器平台支持永久性存储、可让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)*此源代码平台可大规模管理构建和部署映像。
- *开源标准*除了其他开源技术之外、这些标准还包括用于容器流程编排的开放式容器计划(OCI)和Kubernetes。您不受限于特定供应商的技术或业务路线图。
- **CI/CD**管道 OpenShift为CI/CD管道提供开箱即用的支持，使开发团队可以自动执行应用程序交付流程的每个步骤，并确保对应用程序代码或配置进行的每次更改都能执行该步骤。
- *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组。
- 自动化构建和部署 OpenShift允许开发人员选择构建容器化应用程序、或者让平台基于应用程序源代码甚至二进制文件构建容器。然后，该平台会根据为这些应用程序定义的特征在整个基础架构中自动部署这些应用程序。例如，为了使资源符合第三方许可证的要求，应分配的资源数量以及应部署在基础架构上的什么位置。
- 一致的环境 OpenShift可确保为开发人员配置的环境在应用程序的整个生命周期内从操作系统、库、运行时版本(例如Java运行时)、 甚至包括正在使用的应用程序运行时(例如Tomcat)、以消除因环境不一致而产生的风险。
- *配置管理*配置和敏感数据管理功能内置在平台中，以确保无论使用哪种技术构建应用程序或环境如何，都能为应用程序提供一致且不受环境限制的应用程序配置
已部署。
- *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 集成式监控和日志管理可为开发人员提供即时指标，使他们能够研究应用程序在发生变更时的行为方式，并能够在应用程序生命周期中尽早修复问题。
- 安全性和容器目录 OpenShift提供多租户功能、并通过使用安全增强型Linux (SELinux)、CGroups和安全计算模式(seccomp)建立的安全性隔离和保护容器、保护用户免受有害代码执行的影响。此外，它还通过 TLS 证书为各种子系统提供加密，并可访问经过 Red Hat 认证的容器（ access.redhat.com/containers ），这些容器经过扫描和评级，并特别强调安全性，以便为最终用户提供经过认证的，可信的安全应用程序容器。



Red Hat OpenShift 的部署方法

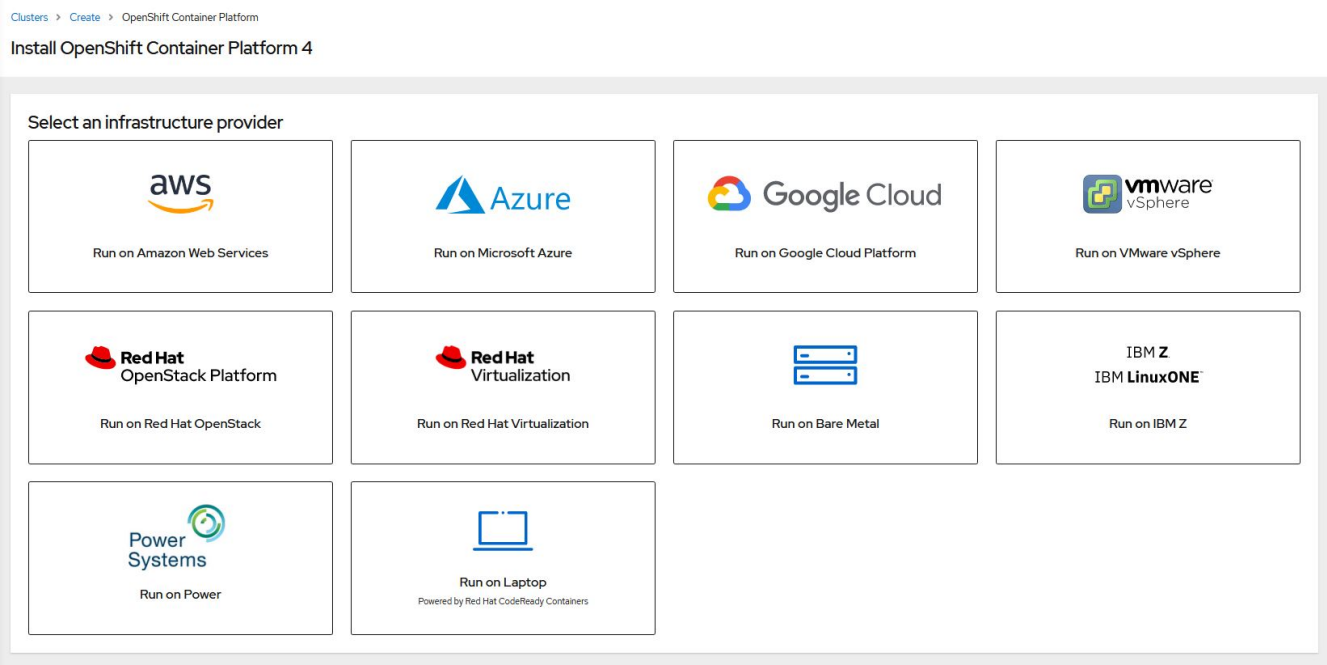
从 Red Hat OpenShift 4 开始， OpenShift 的部署方法包括使用用户配置基础架构（User Provisioned Infrastructure， UPI）手动部署高度自定义的部署，或者使用安装程序配置的基础架构（IPI）完全自动化部署。

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

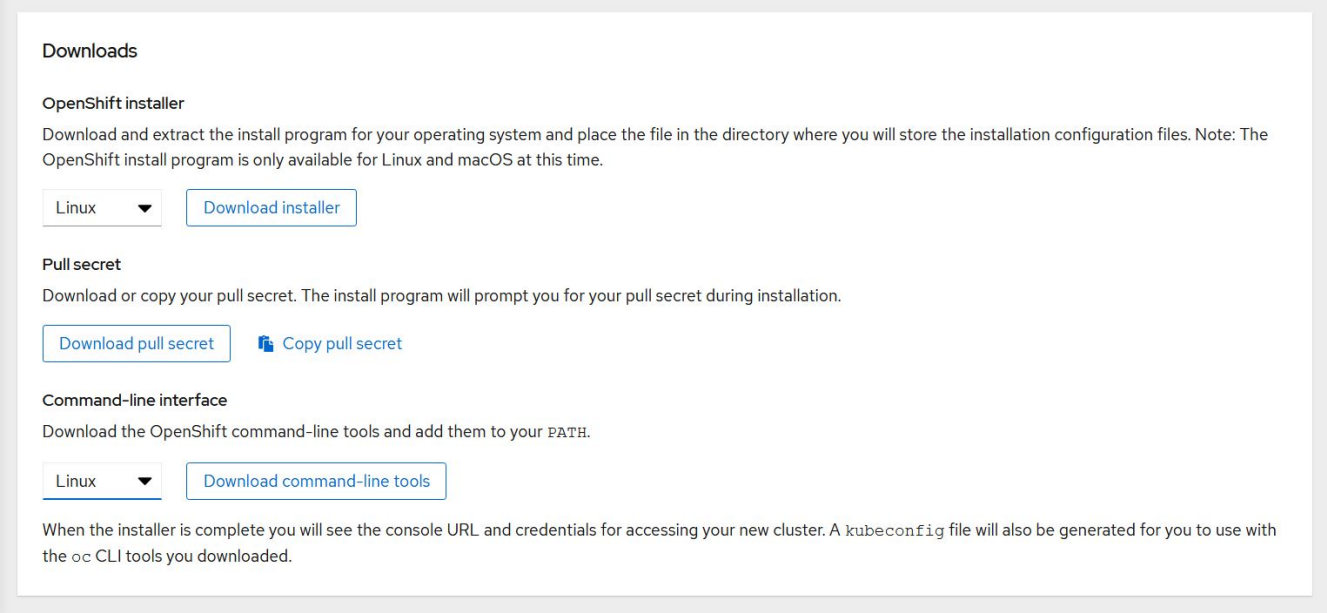
Red Hat OpenShift 的 IPI 安装

OpenShift 的安装程序配置基础架构（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. 选择要将 Red Hat OpenShift 部署到的环境。



3. 在下一屏幕中，下载安装程序，唯一的拉取密钥以及用于管理的 CLI 工具。



4. 按照 "安装说明" 由 Red Hat 提供，用于部署到您选择的环境。

经过 **NetApp** 验证的 **OpenShift** 部署

NetApp 已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和验证了 Red Hat OpenShift 在其实验室中的部署：

- "裸机上的 OpenShift"
- "基于 Red Hat OpenStack 平台的 OpenShift"
- "基于 Red Hat 虚拟化的 OpenShift"
- "VMware vSphere 上的 OpenShift"

裸机上的 **OpenShift**

基于裸机的 OpenShift 可在商用服务器上自动部署 OpenShift 容器平台。

基于裸机的 OpenShift 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，同时还可以为尚未准备好进行容器化的应用程序提供虚拟化工作负载支持。通过在裸机上部署，除了 OpenShift 环境之外，您无需额外的开销即可管理主机虚拟机管理程序环境。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。

裸机上的 **OpenShift** 可提供以下功能：

- *IPI或辅助安装程序部署*借助由安装程序配置的基础架构(IPI)在裸机服务器上部署的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。
- *紧凑型集群设计*为了最大限度地降低硬件要求、裸机上的OpenShift允许用户部署仅包含3个节点的集群、方法是使OpenShift控制平台节点也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。此容器本机虚拟化可在容器内运行 KVM 虚拟机管理程序，并为 VM 存储附加永久性卷。
- *人工智能/机器学习优化的基础架构*通过将基于GPU的工作节点整合到您的OpenShift环境中并利用OpenShift高级计划、为机器学习应用程序部署Kubeflow等应用程序。

网络设计

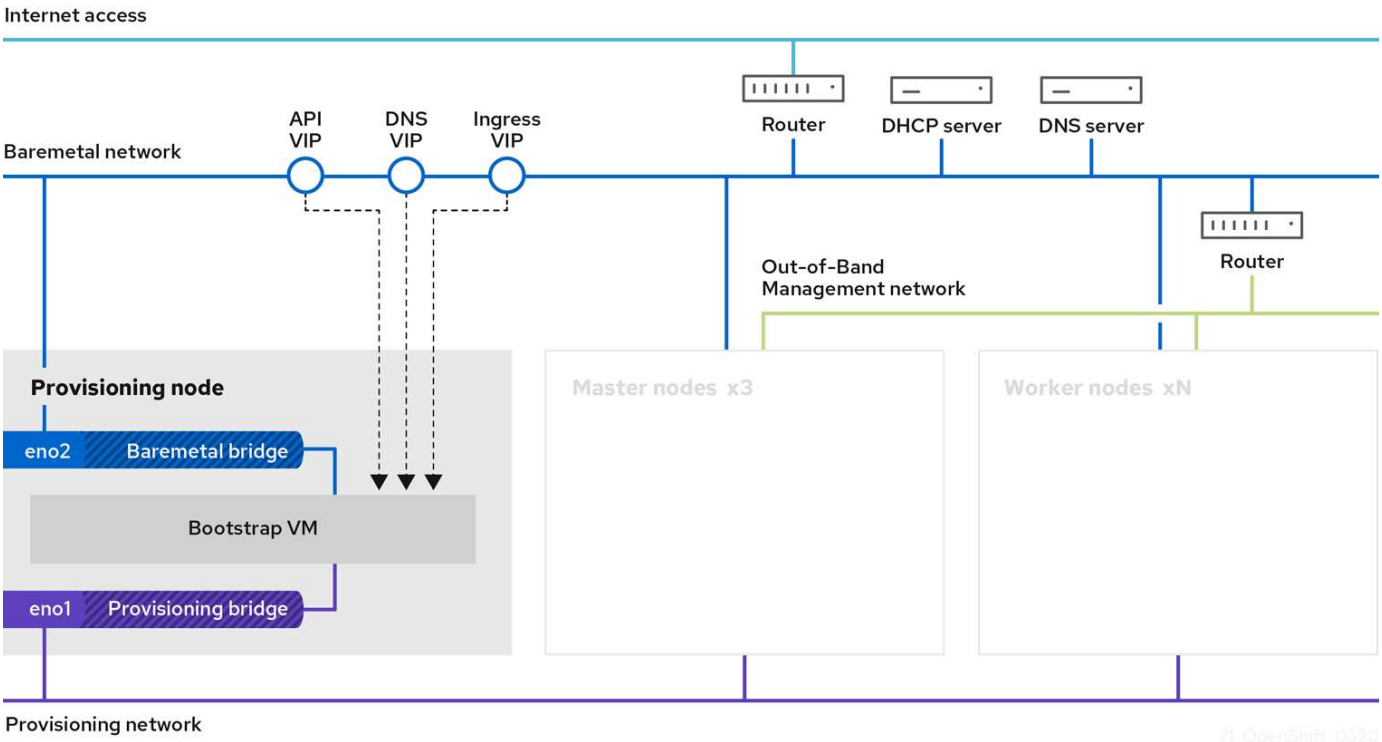
NetApp 解决方案上的 Red Hat OpenShift 使用两个数据交换机提供 25 Gbps 的主数据连接。它还使用两个管理交换机，这些交换机以 1 Gbps 的速度提供连接，用于存储节点的带内管理以及 IPMI 功能的带外管理。

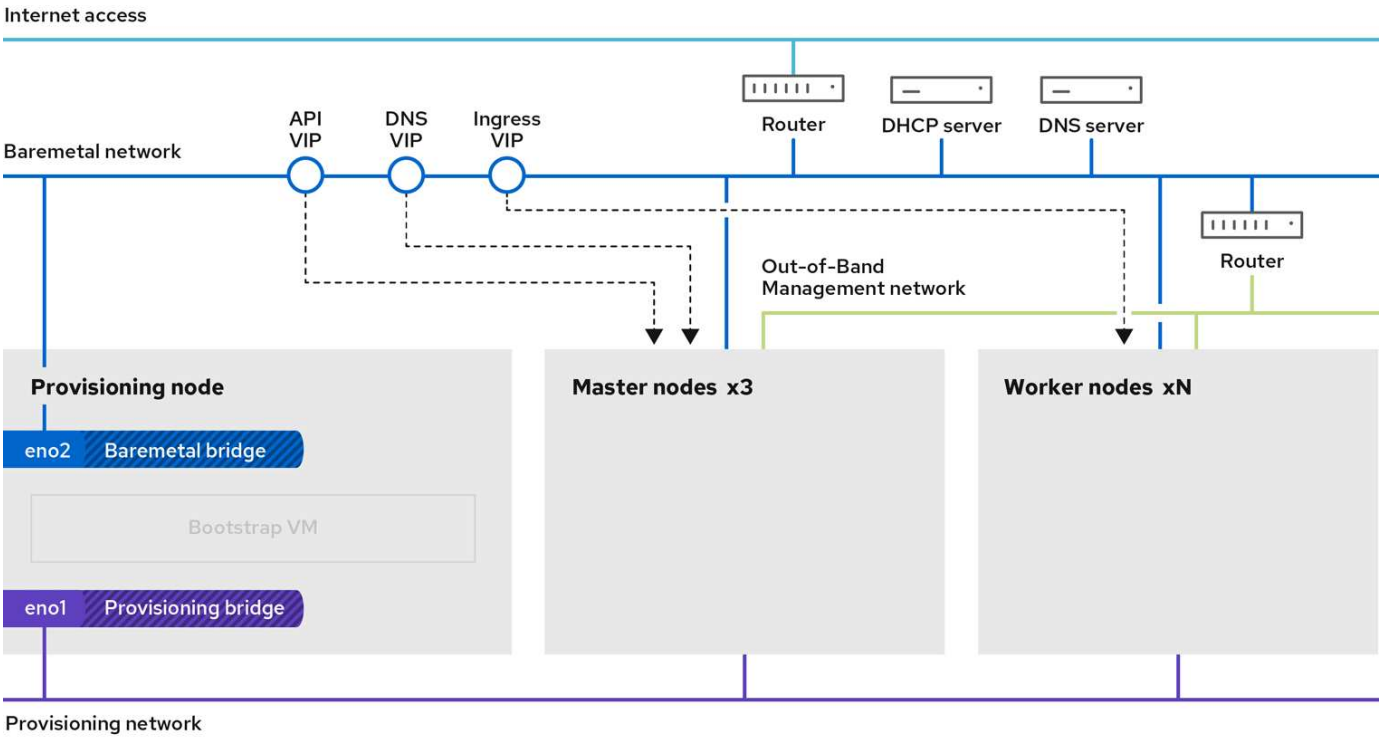
对于 OpenShift 裸机 IPI 部署，您必须创建一个配置程序节点，即必须将网络接口连接到不同网络的 Red Hat Enterprise Linux 8 计算机。

- *配置网络*此网络用于启动裸机节点并安装部署OpenShift集群所需的映像和软件包。
- *裸机网络*部署集群后、此网络用于集群面向公共的通信。

在设置配置程序节点时，客户会创建网桥接口，以便在节点本身以及为部署目的配置的 Bootstrap 虚拟机上正确路由流量。部署集群后， API 和传入 VIP 地址将从启动节点迁移到新部署的集群。

以下图像显示了 IPI 部署期间和部署完成后的环境。





VLAN 要求

采用 NetApp 解决方案的 Red Hat OpenShift 可通过使用虚拟局域网（VLAN）在逻辑上隔离不同用途的网络流量。

VLAN	目的	VLAN ID
带外管理网络	管理裸机节点和 IPMI	16.
裸机网络	集群可用后用于 OpenShift 服务的网络	181
配置网络	通过 IPI 以 PXE 启动和安装裸机节点的网络	3485

> 尽管这些网络中的每个网络实际上都由 VLAN 分隔，但必须在访问模式下设置每个物理端口并分配主 VLAN ，因为在 PXE 启动序列期间无法传递 VLAN 标记。

网络基础架构支持资源

在部署 OpenShift 容器平台之前，应具备以下基础架构：

- 至少一个 DNS 服务器，该服务器可提供可从带内管理网络和 VM 网络访问的完整主机名解析。
- 至少可从带内管理网络和 VM 网络访问一个 NTP 服务器。
- （可选）带内管理网络和 VM 网络的出站 Internet 连接。

基于 **Red Hat OpenStack** 平台的 **OpenShift**

Red Hat OpenStack 平台为创建，部署和扩展安全可靠的私有 OpenStack 云提供了一个集成基础。

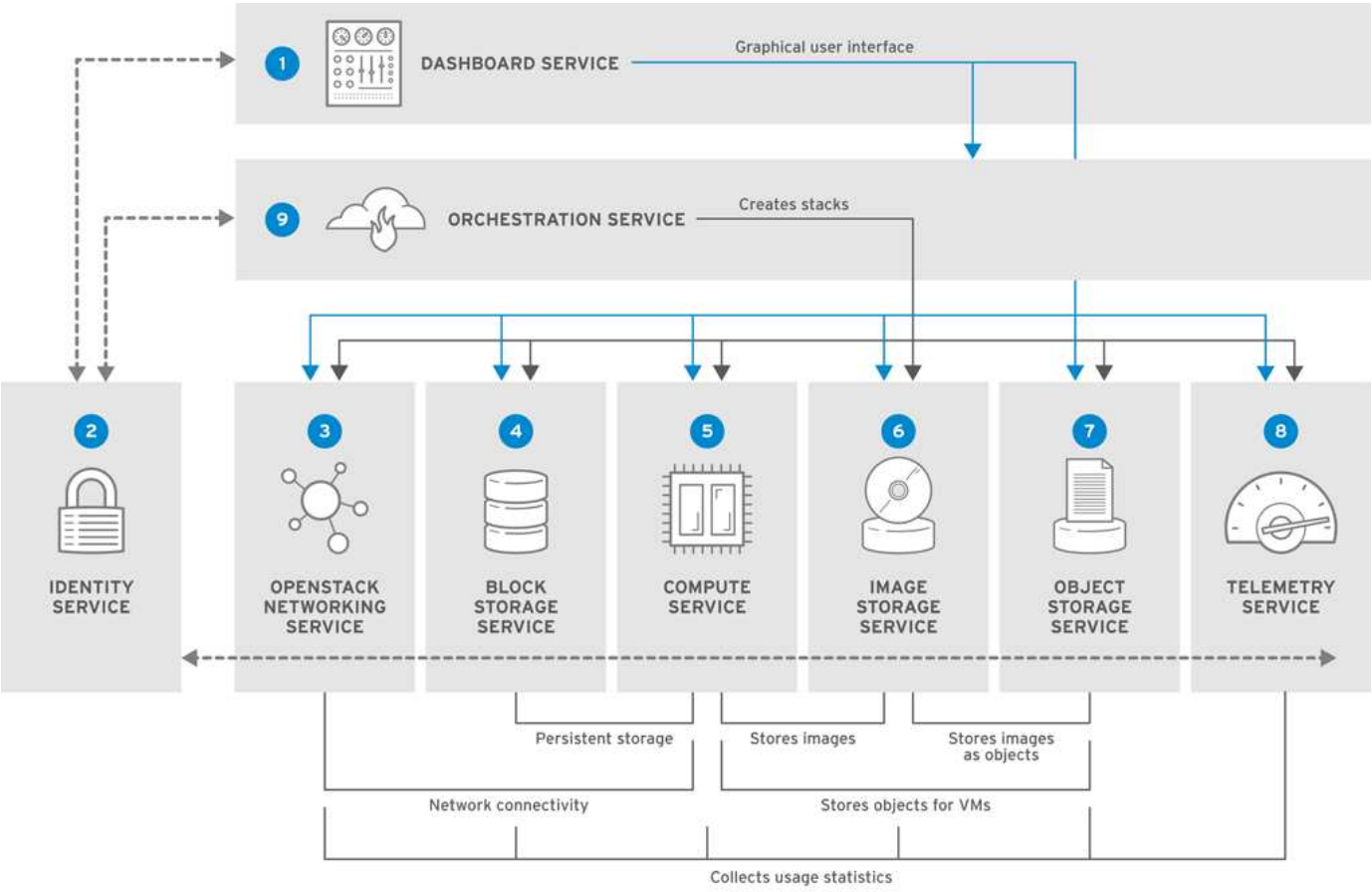
OSP 是一种基础架构即服务（ Infrastructure-as-a-Service ， IaaS ）云，由一组控制服务实施，用于管理计算，存储和网络资源。环境可通过基于 Web 的界面进行管理，管理员和用户可以控制，配置和自动化 OpenStack 资源。此外，还通过广泛的命令行界面和 API 为 OpenStack 基础架构提供便利，为管理员和最终用户提供全面的自动化功能。

OpenStack 项目是一个快速开发的社区项目，每六个月提供更新版本。最初， Red Hat OpenStack Platform 通过发布新版本以及每个上游版本并为每个第三个版本提供长期支持，跟上了此版本周期的步伐。最近，在 OSP 16.0 版（基于 OpenStack 训练）中， Red Hat 选择不跟上版本号的步伐，而是将新功能支持到子版本中。最新版本是 Red Hat OpenStack Platform 16.1 ，其中包括上游 Ussuri 和维多利亚版本的后台高级功能。

有关 OSP 的详细信息，请参见 "Red Hat OpenStack Platform 网站"。

OpenStack 服务

OpenStack 平台服务以容器的形式部署，可将服务彼此隔离，并可轻松进行升级。OpenStack 平台使用一组使用 Kolla 构建和管理的容器。可通过从 Red Hat 自定义门户中提取容器映像来部署服务。这些服务容器可使用 Podman 命令进行管理，并可使用 Red Hat OpenStack Director 进行部署，配置和维护。



服务	项目名称	**Description**
信息板	Horizon	基于 Web 浏览器的信息板，用于管理 OpenStack 服务。
身份	Keystone	用于身份验证和授权 OpenStack 服务以及管理用户，项目和角色的集中式服务。
OpenStack 网络	中子	在 OpenStack 服务的接口之间提供连接。
块存储	Cinder	管理虚拟机（ VM ）的永久性块存储卷。
计算	Nova	管理和配置计算节点上运行的 VM 。
图像	概览	用于存储 VM 映像和卷快照等资源的注册表服务。
对象存储	Swift	允许用户存储和检索文件和任意数据。
遥测	Ceilmeter	提供对云资源使用情况的衡量指标。
流程编排	热	基于模板的流程编排引擎，支持自动创建资源堆栈。

网络设计

采用 NetApp 解决方案的 Red Hat OpenShift 使用两个数据交换机以 25 Gbps 的速度提供主数据连接。它还使用两个额外的管理交换机，这些交换机以 1 Gbps 的速度提供连接，用于存储节点的带内管理以及 IPMI 功能的带外管理。

Red Hat OpenStack Director 需要 IPMI 功能才能使用具有讽刺意味的裸机配置服务部署 Red Hat OpenStack

Platform 。

VLAN 要求

采用 NetApp 的 Red Hat OpenShift 旨在通过使用虚拟局域网（ VLAN ）在逻辑上隔离不同用途的网络流量。此配置可以进行扩展，以满足客户需求，或者为特定网络服务提供进一步隔离。下表列出了在 NetApp 验证解决方案时实施解决方案所需的 VLAN 。

VLAN	目的	VLAN ID
带外管理网络	用于管理物理节点和 IPMI 服务的网络具有讽刺意味。	16.
存储基础架构	用于控制器节点直接映射卷以支持 Swift 等基础架构服务的网络。	201
存储 Cinder	用于将块卷直接映射和附加到环境中部署的虚拟实例的网络。	202
内部 API	用于通过 API 通信， RPC 消息和数据库通信在 OpenStack 服务之间进行通信的网络。	301.
租户	中子通过 VXLAN 的通道为每个租户提供自己的网络。网络流量在每个租户网络中隔离。每个租户网络都有一个关联的 IP 子网，而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导致冲突	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与底层存储层之间的中间接口。代理接收传入请求并找到所需的副本以检索请求的数据。	303.
PXE	OpenStack Director 在具有讽刺意味的裸机配置服务中提供 PXE 启动，用于编排 OSP Overcloud 的安装。	3484
外部	公共网络，用于托管用于图形管理的 OpenStack 信息板（ Horizon ），并允许公有 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的访问，例如 SSH 访问， DNS 流量和网络时间协议（ NTP ）流量。此网络还充当非控制器节点的网关。	3486

网络基础架构支持资源

在部署 OpenShift 容器平台之前，应具备以下基础架构：

- 至少一个 DNS 服务器，可提供完整的主机名解析。
- 至少三个 NTP 服务器，这些服务器可以使解决方案中的服务器保持时间同步。
- （可选） OpenShift 环境的出站 Internet 连接。

生产部署的最佳实践

本节列出了企业在将此解决方案部署到生产环境之前应考虑的几个最佳实践。

将 OpenShift 部署到至少包含三个计算节点的 OSP 私有云

本文档中介绍的经验证的架构通过部署三个 OSP 控制器节点和两个 OSP 计算节点提供了最适合 HA 操作的硬件部署。此架构可确保容错配置，其中两个计算节点均可启动虚拟实例，而已部署的 VM 则可在两个虚拟机管理程序之间迁移。

由于 Red Hat OpenShift 最初使用三个主节点进行部署，因此双节点配置可能会发生原因至少两个主节点占用同一节点，从而可能导致 OpenShift 在特定节点不可用时发生中断。因此， Red Hat 的最佳实践是至少部署三个

OSP 计算节点，以便 OpenShift 主节点可以均匀分布，并且解决方案可以获得更多的容错能力。

配置虚拟机 / 主机关联性

通过启用虚拟机 / 主机关联性，可以在多个虚拟机管理程序节点之间分布 OpenShift 主节点。

关联性是一种为一组 VM 和 / 或主机定义规则的方法，用于确定这些 VM 是在组中的同一主机上运行还是在不同主机上运行。它通过创建由具有一组相同参数和条件的 VM 和 / 或主机组成的关联组来应用于 VM 。根据关联组中的 VM 是在组中的同一主机上运行，还是在不同主机上单独运行，此关联组的参数可以定义正关联性或负关联性。在 Red Hat OpenStack 平台中，可以通过创建服务器组和配置筛选器来创建和实施主机关联性和反关联性规则，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。

默认情况下，服务器组最多可管理 10 个虚拟实例的放置。可以通过更新 Nova 的默认配额来修改此设置。

> OSP 服务器组具有特定的硬关联性 / 反关联性限制；如果没有足够的资源可在不同的节点上部署，或者没有足够的资源可用于共享节点，则 VM 将无法启动。

要配置相关性组，请参见 "如何为 OpenStack 实例配置关联性和反关联性？"。

使用自定义安装文件进行 **OpenShift** 部署

IPI 可通过本文档前面讨论的交互式向导轻松部署 OpenShift 集群。但是，在集群部署过程中，您可能需要更改某些默认值。

在这些情况下，无需立即部署集群，即可运行并执行向导任务；而是创建一个配置文件，以便稍后可以从中部署集群。如果您需要更改任何 IPI 默认值，或者要在环境中部署多个相同的集群以用于多租户等其他用途，则此功能非常有用。有关为 OpenShift 创建自定义安装配置的详细信息，请参见 "Red Hat OpenShift 通过自定义在 OpenStack 上安装集群"。

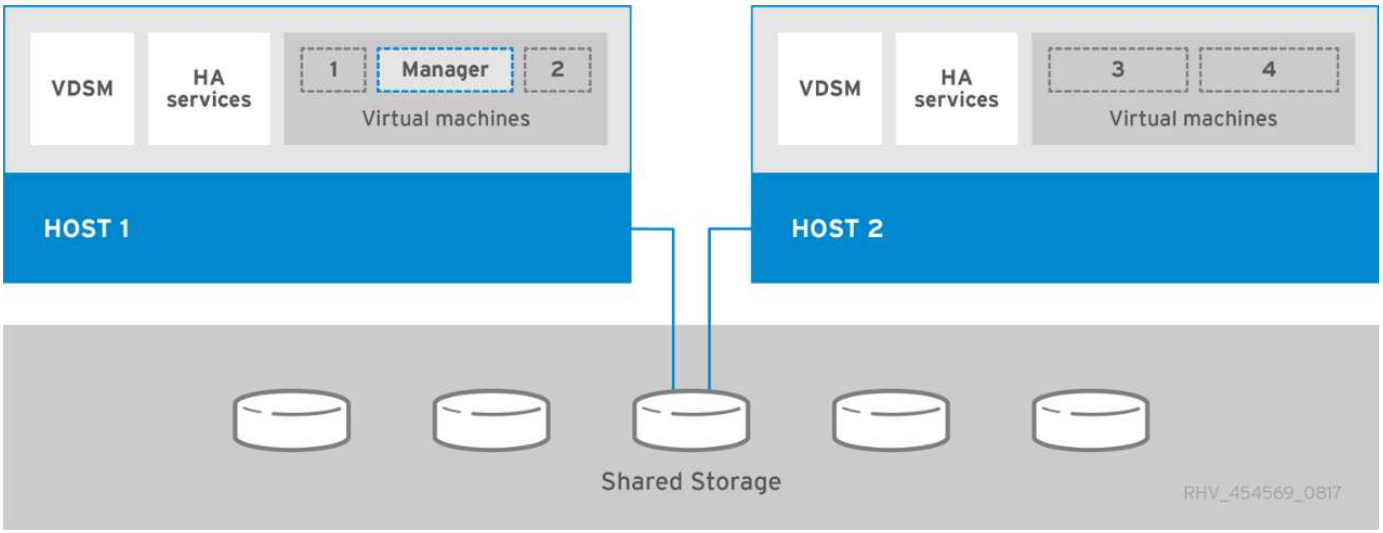
基于 **Red Hat** 虚拟化的 **OpenShift**

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

有关 RHV 的详细信息，请参见 "Red Hat 虚拟化网站"。

RHV 具有以下功能：

- 集中管理虚拟机和主机 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的图形用户界面、用于从中央界面管理解决方案。
- *自托管引擎*为了最大限度地降低硬件要求，RHV允许将RHV Manager (RHV-M)部署为运行子VM的同一主机上的VM。
- *高可用性*为了避免主机发生故障时发生中断，RHV允许将虚拟机配置为高可用性。高可用性 VM 可通过故障恢复策略在集群级别进行控制。
- *高可扩展性*一个RHV集群最多可包含200台虚拟机管理程序主机、使IT能够满足大型VM的需求、从而托管资源要求较高的企业级工作负载。
- *从RHV、安全虚拟化(sVirt)和安全增强Linux (SELinux)技术继承的增强安全性*被RHV用于提高主机和VM的安全性和强化。这些功能的主要优势是对虚拟机及其相关资源进行逻辑隔离。



网络设计

NetApp 解决方案上的 Red Hat OpenShift 使用两个数据交换机提供 25 Gbps 的主数据连接。它还使用两个额外的管理交换机，这些交换机以 1 Gbps 的速度提供连接，用于存储节点的带内管理以及 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布局和用途，并概述部署解决方案的前提条件。

VLAN 要求

RHV 上的 Red Hat OpenShift 旨在通过使用虚拟局域网（VLAN）在逻辑上隔离不同用途的网络流量。此配置可以进行扩展，以满足客户需求，或者为特定网络服务提供进一步隔离。下表列出了在 NetApp 验证解决方案时实施解决方案所需的 VLAN 。

VLAN	目的	VLAN ID
带外管理网络	管理物理节点和 IPMI	16.
VM 网络	虚拟子系统网络访问	1172.
带内管理网络	对 RHP-H 节点， RHP-Manager 和 ovirtmgmt 网络进行管理	3343
存储网络	适用于 NetApp Element iSCSI 的存储网络	3344
迁移网络	用于虚拟子系统迁移的网络	3345

网络基础架构支持资源

在部署 OpenShift 容器平台之前，应具备以下基础架构：

- 至少一个 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。
- 至少可从带内管理网络和 VM 网络访问一个 NTP 服务器。
- （可选）带内管理网络和 VM 网络的出站 Internet 连接。

生产部署的最佳实践

本节列出了企业在将此解决方案部署到生产环境之前应考虑的几个最佳实践。

将 OpenShift 部署到至少包含三个节点的 RHV 集群

本文档中介绍的经验证的架构介绍了适用于 HA 操作的最低硬件部署，具体方法是部署两个 RHV-H 虚拟机管理程序节点，并确保采用容错配置，两个主机均可管理托管引擎，而已部署的虚拟机可在两个虚拟机管理程序之间迁移。

由于 Red Hat OpenShift 最初使用三个主节点进行部署，因此在双节点配置中，可以确保至少有两个主节点将占用同一节点，如果特定节点不可用，可能会导致 OpenShift 中断。因此， Red Hat 的最佳实践是，在解决方案中至少部署三个 RHV-H 虚拟机管理程序节点，以便 OpenShift 主节点可以均匀分布，并且解决方案可以获得更多的容错能力。

配置虚拟机 / 主机关联性

您可以通过启用虚拟机 / 主机关联性在多个虚拟机管理程序节点之间分布 OpenShift 主节点。

关联性是一种为一组 VM 和 / 或主机定义规则的方法，用于确定这些 VM 是在组中的同一主机上运行还是在不同主机上运行。它通过创建由具有一组相同参数和条件的 VM 和 / 或主机组成的关联组来应用于 VM 。根据关联组中的 VM 是在组中的同一主机上运行，还是在不同主机上单独运行，此关联组的参数可以定义正关联性或负关联性。

为参数定义的条件可以是强制实施，也可以是软强制实施。强制实施可确保关联组中的 VM 始终严格遵循正负关联性，而不考虑外部条件。软强制实施可确保在可行的情况下为关联组中的 VM 设置更高的首选项，以遵循正或负关联性。在本文档所述的两个或三个虚拟机管理程序配置中，建议使用软关联性设置。在大型集群中，硬关联可以正确分布 OpenShift 节点。

要配置相关性组，请参见 "Red Hat 6.11 。关联性组文档"。

使用自定义安装文件进行 OpenShift 部署

IPI 可通过本文档前面讨论的交互式向导轻松部署 OpenShift 集群。但是，在集群部署过程中，可能需要更改某些默认值。

在这些情况下，您无需立即部署集群即可运行和执行向导任务。而是会创建一个配置文件，以便稍后从该文件部署集群。如果要更改任何 IPI 默认值，或者要在环境中部署多个相同的集群以用于多租户等其他用途，则此功能非常有用。有关为 OpenShift 创建自定义安装配置的详细信息，请参见 "Red Hat OpenShift 通过自定义在 RHV 上安装集群"。

VMware vSphere 上的 OpenShift

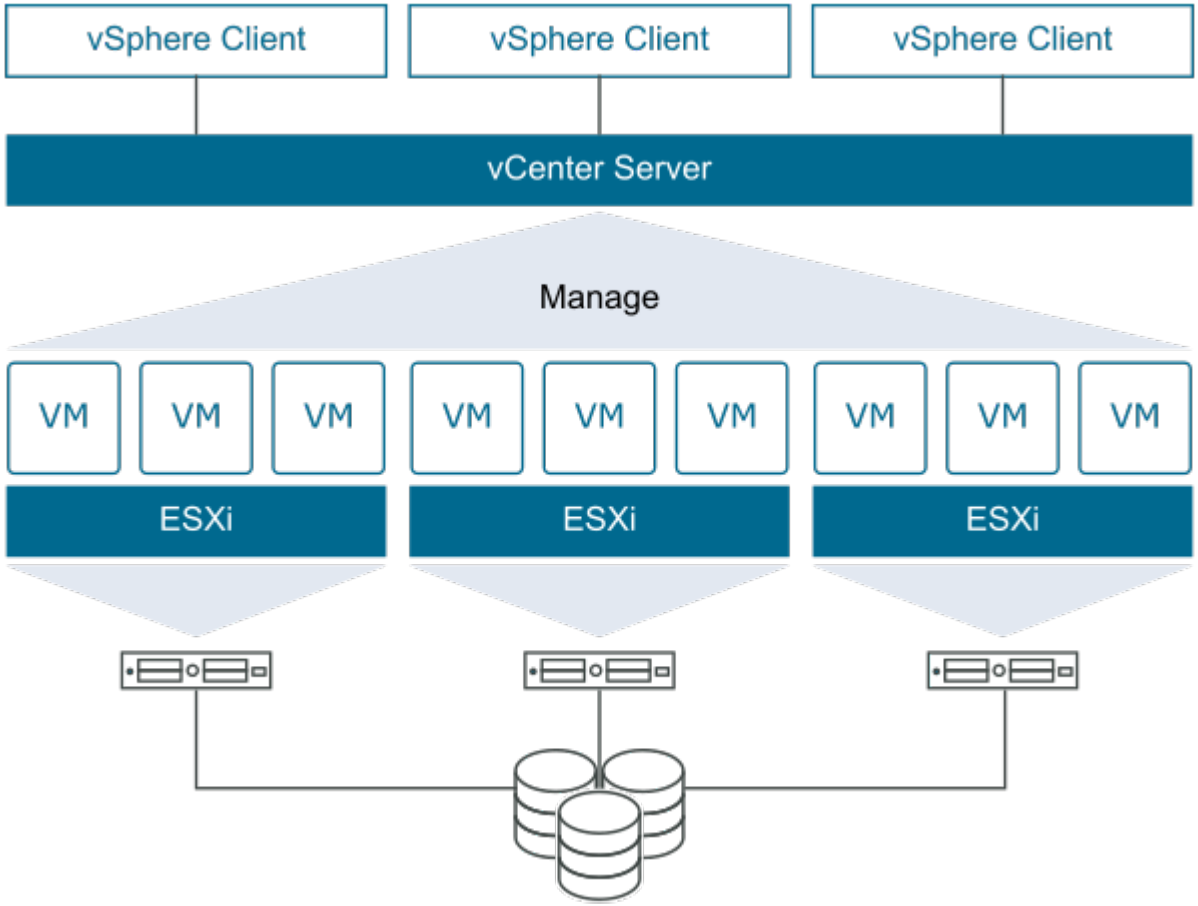
VMware vSphere 是一个虚拟化平台，用于集中管理 ESXi 虚拟机管理程序上运行的大量虚拟化服务器和网络。

有关 VMware vSphere 的详细信息，请参见 "VMware vSphere 网站"。

VMware vSphere 可提供以下功能：

- * VMware vCenter Server* VMware vCenter Server 可通过一个控制台统一管理所有主机和 VM ，并聚合对集群，主机和 VM 的性能监控。
- **VMware vSphere vMotion** VMware vCenter允许您根据请求无中断地在集群中的节点之间热迁移VM。
- *vSphere高可用性*为了避免主机发生故障时发生中断，VMware vSphere允许将主机集群化并配置为高可用性。由于主机故障而中断的 VM 不久将在集群中的其他主机上重新启动，从而还原服务。

- *分布式资源计划程序(DRS)*可以配置VMware vSphere集群，以便对其托管的VM的资源需求进行负载平衡。具有资源管理的 VM 可以热迁移到集群中的其他节点，以确保有足够的可用资源。



网络设计

NetApp 解决方案上的 Red Hat OpenShift 使用两个数据交换机提供 25 Gbps 的主数据连接。它还使用两个额外的管理交换机，这些交换机以 1 Gbps 的速度提供连接，用于存储节点的带内管理以及 IPMI 功能的带外管理。OCP 使用 VMware vSphere 上的 VM 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布局和用途，并概述了部署解决方案的前提条件。

VLAN 要求

VMware vSphere 上的 Red Hat OpenShift 旨在通过使用虚拟局域网（VLAN）在逻辑上隔离不同用途的网络流量。此配置可以进行扩展，以满足客户需求，或者为特定网络服务提供进一步隔离。下表列出了在 NetApp 验证解决方案时实施解决方案所需的 VLAN 。

VLAN	目的	VLAN ID
带外管理网络	管理物理节点和 IPMI	16.
VM 网络	虚拟子系统网络访问	181
存储网络	ONTAP NFS 的存储网络	184
存储网络	适用于 ONTAP iSCSI 的存储网络	185.

VLAN	目的	VLAN ID
带内管理网络	管理 ESXi 节点， vCenter Server ，ONTAP Select	3480
存储网络	适用于 NetApp Element iSCSI 的存储网络	3481
迁移网络	用于虚拟子系统迁移的网络	3482

网络基础架构支持资源

在部署 OpenShift 容器平台之前，应具备以下基础架构：

- 至少一个 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。
- 至少可从带内管理网络和 VM 网络访问一个 NTP 服务器。
- （可选）带内管理网络和 VM 网络的出站 Internet 连接。

生产部署的最佳实践

本节列出了企业在将此解决方案部署到生产环境之前应考虑的几个最佳实践。

将 **OpenShift** 部署到至少包含三个节点的 **ESXi** 集群

本文档中介绍的经验证的架构介绍了适用于 HA 操作的最低硬件部署，具体方法是部署两个 ESXi 虚拟机管理程序节点，并通过启用 VMware vSphere HA 和 VMware vMotion 来确保容错配置。此配置允许部署的 VM 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，可能会导致 OpenShift 中断。因此， Red Hat 的最佳实践是，必须至少部署三个 ESXi 虚拟机管理程序节点，以便可以均匀分布 OpenShift 主节点，从而提高容错能力。

配置虚拟机和主机关联性

通过启用 VM 和主机关联性，可确保在多个虚拟机管理程序节点之间分布 OpenShift 主节点。

关联性或反关联性是一种为一组 VM 和 / 或主机定义规则的方法，用于确定这些 VM 是在同一主机上运行还是在组中的主机上运行，还是在不同主机上运行。它通过创建由具有一组相同参数和条件的 VM 和 / 或主机组成的关联组来应用于 VM 。根据关联组中的 VM 是在组中的同一主机上运行，还是在不同主机上单独运行，此关联组的参数可以定义正关联性或负关联性。

要配置相关性组，请参见 "vSphere 6.7 文档：使用 DRS 关联性规则"。

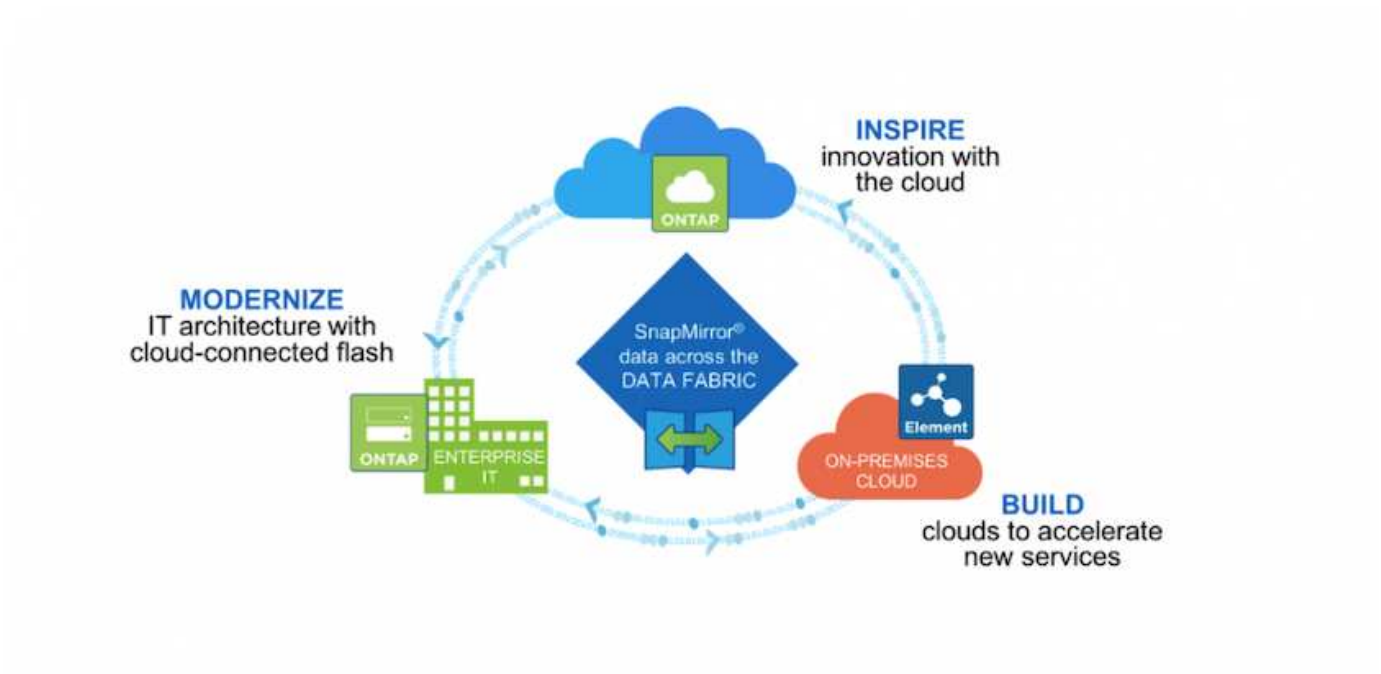
使用自定义安装文件进行 **OpenShift** 部署

IPI 可通过本文档前面讨论的交互式向导轻松部署 OpenShift 集群。但是，在集群部署过程中，您可能需要更改某些默认值。

在这些情况下，您无需立即部署集群即可运行和执行向导任务，但向导会创建一个配置文件，以便稍后可以从中部署集群。如果您需要更改任何 IPI 默认值，或者要在环境中部署多个相同的集群以用于多租户等其他用途，则此功能非常有用。有关为 OpenShift 创建自定义安装配置的详细信息，请参见 "Red Hat OpenShift 通过自定义在 vSphere 上安装集群"。

NetApp 存储概述

NetApp 拥有多个存储平台，这些平台符合我们的 Astra Trident Storage Orchestrator 标准，可为在 Red Hat OpenShift 上部署的应用程序配置存储。



- AFF 和 FAS 系统运行 NetApp ONTAP ，并为基于文件（ NFS ）和基于块（ iSCSI ）的使用情形提供存储。
- Cloud Volumes ONTAP 和 ONTAP Select 在云和虚拟空间方面的优势各不相同。
- NetApp Cloud Volumes Service （ AWS/GCP ）和 Azure NetApp Files 可在云中提供基于文件的存储。
- NetApp Element 存储系统可在高度可扩展的环境中提供基于块的（ iSCSI ）用例。

> NetApp 产品组合中的每个存储系统都可以简化内部站点和云之间的数据管理和移动，从而确保数据位于应用程序所在位置。

以下页面介绍了有关已在 Red Hat OpenShift with NetApp 解决方案中验证的 NetApp 存储系统的追加信息：

- "NetApp ONTAP"
- "NetApp Element"

NetApp ONTAP

NetApp ONTAP 是一款功能强大的存储软件工具，具有直观的图形用户界面，具有自动化集成功能的 REST API ，基于 AI 的预测性分析和更正操作，无中断硬件升级和跨存储导入等功能。

有关 NetApp ONTAP 存储系统的详细信息，请访问 "NetApp ONTAP 网站"。

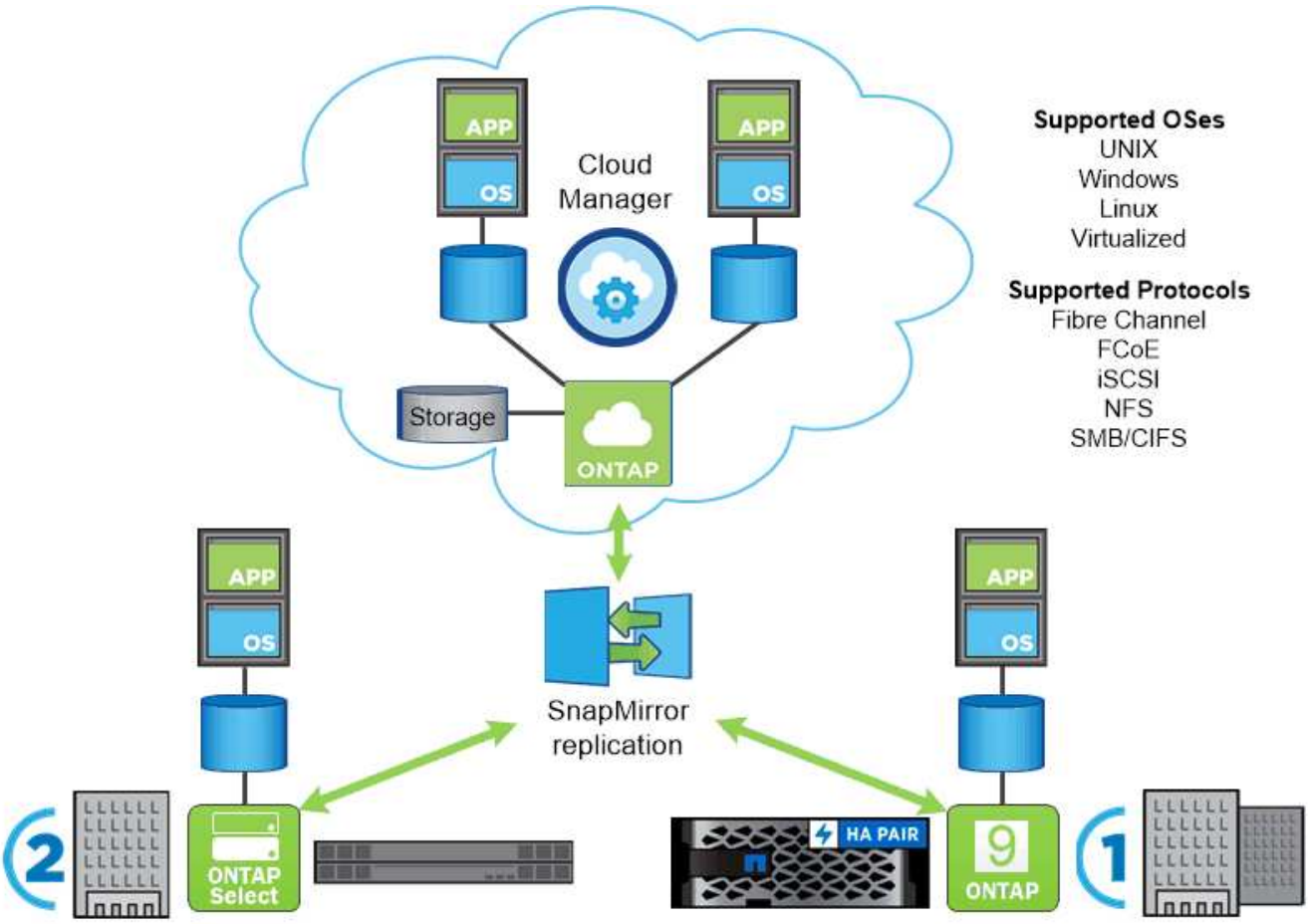
ONTAP 提供以下功能：

- 一个统一存储系统，可同时访问和管理 NFS ， CIFS ， iSCSI ， FC ， FCoE ， 和 FC-NVMe 协议。

- 不同的部署模式包括内部全闪存，混合和全 HDD 硬件配置；基于 VM 的存储平台位于受支持的虚拟机管理程序（如 ONTAP Select ）上；云端为 Cloud Volumes ONTAP 。
- 通过支持自动数据分层，实时数据压缩，重复数据删除和数据缩减，提高 ONTAP 系统的数据存储效率。
- 基于工作负载，由 QoS 控制的存储。
- 与公有云无缝集成，实现数据分层和保护。ONTAP 还提供强大的数据保护功能，使其在任何环境中脱颖而出：
  - * NetApp Snapshot 副本。 * 使用最少的磁盘空间对数据进行快速时间点备份，而不会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 。 * 将不可重写数据写入指定时间段内无法覆盖或擦除的特殊卷，从而高效管理这些数据。
  - * NetApp Snapshot.* 可将多个存储系统中的数据备份到一个中央 SnapVault 副本中，该副本可用作所有指定系统的备份。
  - * NetApp SyncMirror 。 * 可将数据实时镜像到物理连接到同一控制器的两个不同磁盘丛中。
  - * NetApp SnapRestore 。 * 可根据需要从 Snapshot 副本快速还原备份的数据。
  - * NetApp FlexClone 。 * 可根据 Snapshot 副本即时配置 NetApp 卷的完全可读写副本。

有关 ONTAP 的详细信息，请参见 "ONTAP 9 文档中心"。

NetApp ONTAP 可在内部部署，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，这些平台专为低延迟性能，集成数据保护和多协议支持量身定制。

这两个系统均由 NetApp ONTAP 数据管理软件提供支持。 NetApp 数据管理软件是业内最先进的数据管理软件，可提供高度可用的云集成简化存储管理，可提供您的 Data Fabric 所需的企业级速度，效率和安全性。

有关 NetApp AFF/FAS 平台的详细信息，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 系统的完整功能和体验。

有关 ONTAP Select 的详细信息，请单击 "此处"。

Cloud Volumes ONTAP

NetApp Cloud Volumes ONTAP 是 NetApp ONTAP 的云部署版本，可部署在多个公有云中，包括 Amazon AWS， Microsoft Azure 和 Google Cloud 。

有关 Cloud Volumes ONTAP 的详细信息，请单击 "此处"。

NetApp Element： 采用 NetApp 技术的 Red Hat OpenShift

NetApp Element 软件可提供模块化的可扩展性能，每个存储节点均可为环境提供有保障的容量和吞吐量。NetApp Element 系统可以在一个集群中从 4 个节点扩展到 100 个节点，并提供多种高级存储管理功能。

有关 NetApp Element 存储系统的详细信息，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 命令的标准方式。当 SCSI 标准发生变化或以太网网络的性能提高时， iSCSI 存储协议就会受益，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）。在 iSCSI 登录过程中，存储可以响应目标卷已移至其他地址，因此无法继续协商过程。然后，主机将在不需要主机端重新配置的过程中向新地址重新发出登录请求。此过程称为 iSCSI 登录重定向。

iSCSI 登录重定向是 NetApp Element 软件集群的一个关键部分。收到主机登录请求后，节点将根据 IOPS 和卷的容量要求确定集群中应由哪个成员处理流量。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，则会重新分配这些卷。给定卷的多个副本会在阵列中分配。

这样，如果节点发生故障后又发生卷重新分布，则除了注销和登录并重定向到新位置之外，对主机连接不会产生任何影响。通过 iSCSI 登录重定向， NetApp Element 软件集群是一种自我修复型横向扩展架构，能够无中断升级和操作。

NetApp Element 软件集群 QoS

通过 NetApp Element 软件集群，可以按卷动态配置 QoS 。您可以使用每个卷的 QoS 设置根据定义的 SLA 控制存储性能。以下三个可配置参数用于定义 QoS：

- * 最小 IOPS* 。 NetApp Element 软件集群为卷提供的最小可持续 IOPS 数。为卷配置的最小 IOPS 是卷的性能保证级别。每个卷的性能不会低于此级别。
- * 最大 IOPS* 。 NetApp Element 软件集群为特定卷提供的最大可持续 IOPS 数。
- * 突发 IOPS 。 * 在短时突发情形下允许的最大 IOPS 数。突发持续时间设置是可配置的，默认值为 1 分钟。如果卷运行的 IOPS 低于最大 IOPS 级别，则会累积突发额度。如果性能级别变得非常高并不断推送，则允许卷上的 IOPS 短时突发超过最大 IOPS 。

多租户

可通过以下功能实现安全多租户：

- * 安全身份验证。 * 质询握手身份验证协议（ Challenge-Handshake Authentication Protocol ， CHAP ）用于安全卷访问。轻量级目录访问协议（ Lightweight Directory Access Protocol ， LDAP ）用于安全访问集群以进行管理和报告。
- * 卷访问组（ VAG ）。 * 也可以使用 VAG 代替身份验证，将任意数量的 iSCSI 启动程序专用 iSCSI 限定名称（ IQN ）映射到一个或多个卷。要访问 VAG 中的卷，启动程序的 IQN 必须位于该卷组允许的 IQN 列表中。
- * 租户虚拟 LAN （ VLAN ）。 * 在网络级别，使用 VLAN 可提高 iSCSI 启动程序与 NetApp Element 软件集群之间的端到端网络安全性。对于为隔离工作负载或租户而创建的任何 VLAN ， NetApp Element 软件会创建一个单独的 iSCSI 目标 SVIP 地址，该地址只能通过特定 VLAN 进行访问。
- 启用了 VRF 的 VLAN 。 * 为了进一步支持数据中心的安全性和可扩展性，您可以使用 NetApp Element 软件为任何租户 VLAN 启用类似 VRF 的功能。此功能增加了以下两项关键功能：
  - 通过 * L3 路由到租户 SVIP 地址。 * 此功能，您可以将 iSCSI 启动程序置于与 NetApp Element 软件集群不同的网络或 VLAN 上。
  - * IP 子网重叠或重复。 * 此功能可用于向租户环境添加模板，从而可以从同一 IP 子网为每个租户 VLAN 分配 IP 地址。此功能对于扩展和保留 IP 空间非常重要的服务提供商环境非常有用。

企业级存储效率

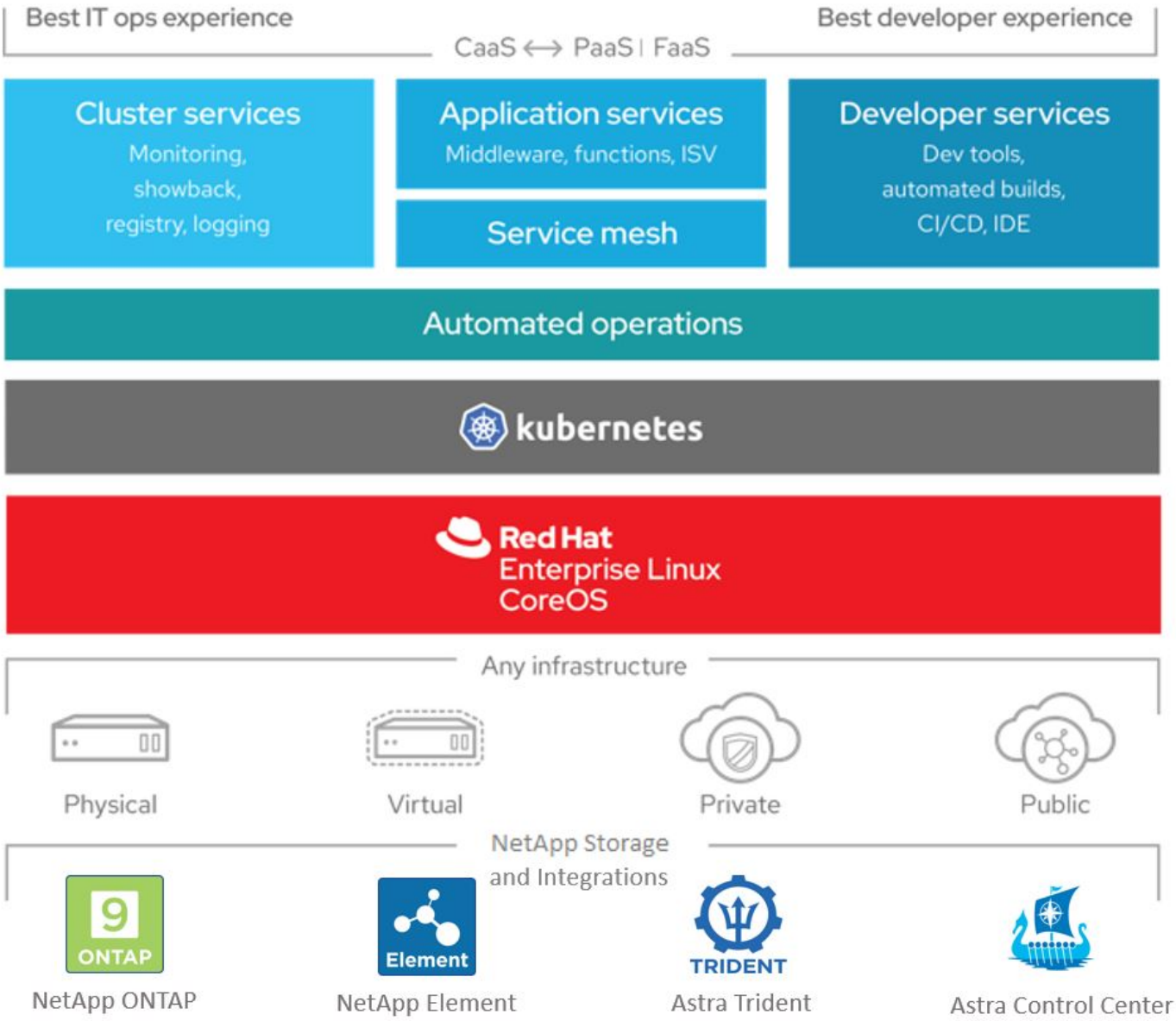
NetApp Element 软件集群可提高整体存储效率和性能。以下功能是实时执行的，始终开启的，无需用户手动配置：

- * 重复数据删除。 * 系统仅存储唯一的 4 K 块。任何重复的 4K 块都会自动与已存储的数据版本关联。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。此系统可显著减少系统中的容量消耗和写入操作。
- * 压缩。 * 数据写入 NVRAM 之前，会实时执行数据压缩。数据会进行压缩，以 4 k 块的形式存储，并在系统中保持压缩状态。这种压缩可显著减少集群中的容量消耗，写入操作和带宽消耗。
- * 精简配置。 * 此功能可在需要时提供适当数量的存储，从而消除因过度配置卷或未充分利用卷而导致的容量消耗。
- * Helix.* 单个卷的元数据存储在元数据驱动器上，并复制到二级元数据驱动器以实现冗余。

> Element 专为自动化而设计。所有存储功能均可通过 API 使用。这些 API 是 UI 用于控制系统的唯一方法。

NetApp 存储集成概述

NetApp 提供了许多产品，可帮助您在基于容器的环境中编排和管理永久性数据，例如 Red Hat OpenShift 。



NetApp Astra Control 采用 NetApp 数据保护技术，为有状态 Kubernetes 工作负载提供丰富的存储和应用程序感知型数据管理服务。Astra 控制服务可用于在云原生 Kubernetes 部署中支持有状态工作负载。Astra 控制中心可支持内部部署中的有状态工作负载，例如 Red Hat OpenShift 。有关详细信息，请访问 NetApp Astra Control 网站 "此处"。

NetApp Astra Trident 是一款开源且完全受支持的存储编排程序，适用于容器和 Kubernetes 分发版，包括 Red Hat OpenShift 。有关详细信息，请访问 Astra Trident 网站 "此处"。

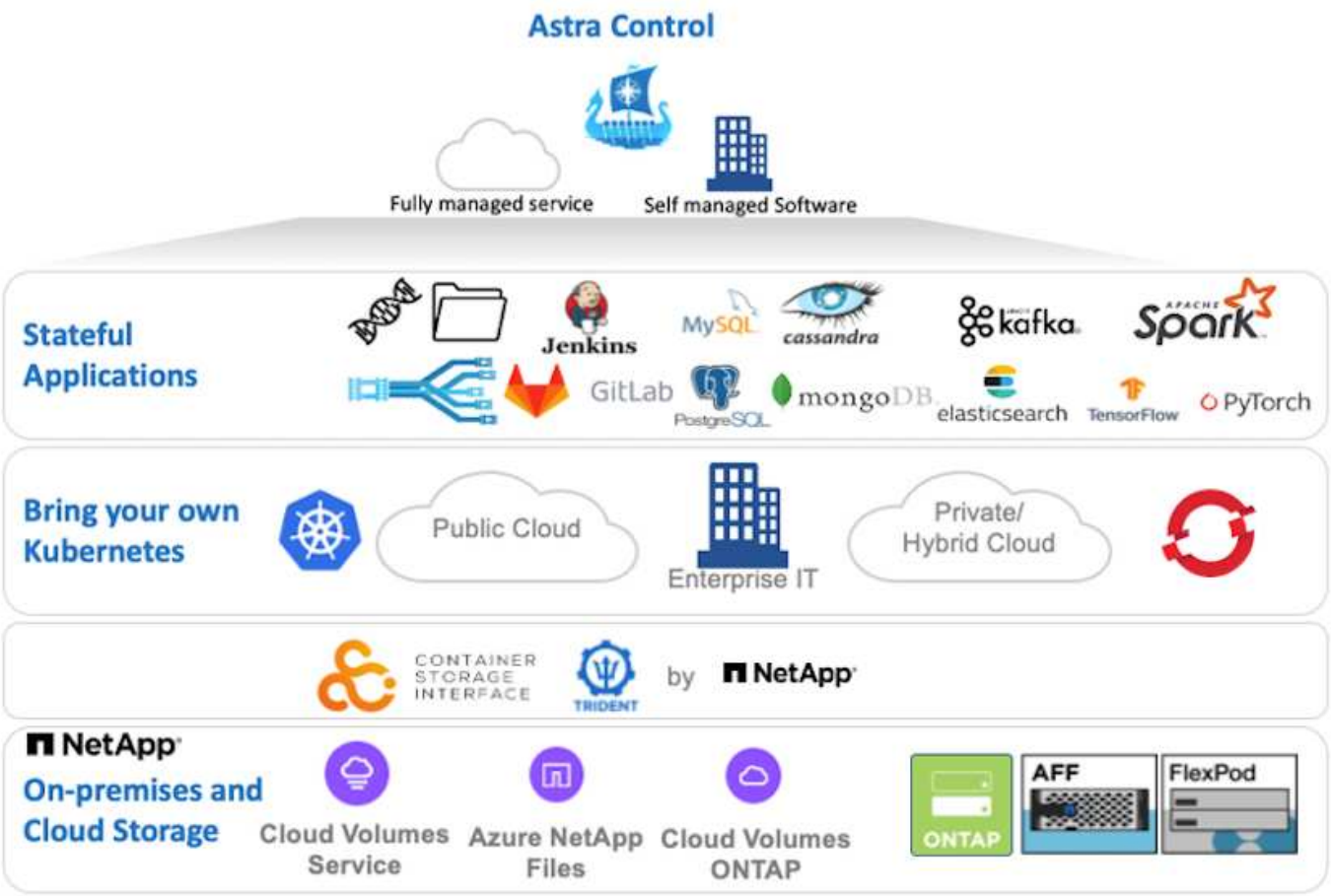
以下页面介绍了有关已在 Red Hat OpenShift with NetApp 解决方案中针对应用程序和永久性存储管理进行验证的 NetApp 产品的追加信息：

- "NetApp Astra 控制中心"
- "NetApp Astra Trident"

NetApp Astra 控制中心概述

NetApp Astra 控制中心为部署在内部环境中并采用 NetApp 数据保护技术的有状态 Kubernetes 工作负载提供丰

富的存储和应用程序感知型数据管理服务。



NetApp Astra 控制中心可以安装在 Red Hat OpenShift 集群上，该集群已部署 Astra Trident 存储编排程序并为其配置存储类和存储后端到 NetApp ONTAP 存储系统。

有关安装和配置 Astra Trident 以支持 Astra 控制中心的信息，请参见 "本文档在此处提供"。

在云互联环境中， Astra 控制中心使用 Cloud Insights 提供高级监控和遥测功能。在没有 Cloud Insights 连接的情况下，可以使用有限的监控和遥测（ 7 天的指标），并通过开放式指标端点导出到 Kubernetes 原生监控工具（ Prometheus 和 Grafana ）。

Astra 控制中心完全集成到 NetApp AutoSupport 和 Active IQ 生态系统中，可为用户提供支持，协助进行故障排除以及显示使用情况统计信息。

除了已付费版本的 Astra 控制中心之外，还提供 90 天评估许可证。评估版可通过电子邮件和社区（ Slack 通道）获得支持。客户可以访问这些以及其他知识库文章以及产品支持信息板上提供的文档。

要开始使用 NetApp Astra 控制中心，请访问 "Astra 网站"。

安装 **Astra** 控制中心的前提条件

1. 一个或多个 Red Hat OpenShift 集群。目前支持版本 4.6 EUS 和 4.7 。
2. 必须已在每个 Red Hat OpenShift 集群上安装和配置 Astra Trident 。
3. 运行 ONTAP 9.5 或更高版本的一个或多个 NetApp ONTAP 存储系统。

> 最佳做法是，在站点上安装的每个 OpenShift 都要有一个专用的 SVM 来用于永久性存储。多站点部署需要额外的存储系统。

4. 必须在每个 OpenShift 集群上配置一个 Trident 存储后端，其中包含一个由 ONTAP 集群提供支持的 SVM 。
5. 在每个 OpenShift 集群上配置的默认 StorageClass ，其中使用 Astra Trident 作为存储配置程序。
6. 必须在每个 OpenShift 集群上安装和配置负载平衡器，以实现负载平衡并公开 OpenShift 服务。

   > 请参见链接 "此处" 有关已为此目的验证的负载平衡器的信息。

7. 必须配置私有映像注册表以托管 NetApp Astra Control Center 映像。

   > 请参见链接 "此处" 为此安装和配置 OpenShift 专用注册表。

8. 您必须对 Red Hat OpenShift 集群具有集群管理员访问权限。
9. 您必须对 NetApp ONTAP 集群具有管理员访问权限。
10. 一个管理工作站，其中安装了 Docker 或 podman ， tridentctl 以及 oc 或 kubectl 工具，并将其添加到 $path 中。

    > Docker 安装的 Docker 版本必须大于 20.10 ，而 Podman 安装的 Podman 版本必须大于 3.0 。

安装 **Astra** 控制中心

使用 **OperatorHub**

1. 登录到 NetApp 支持站点并下载最新版本的 NetApp Astra 控制中心。为此，您需要在 NetApp 帐户中附加许可证。下载完 tarball 后，将其传输到管理工作站。

   > 要开始获取 Astra Control 的试用许可证，请访问 "Astra 注册站点"。

2. 打开 tar ball 的包装并将工作目录更改为生成的文件夹。

   ```
   [netapp-user@rhel7 ~]$ tar -vxzf astra-control-center-
   21.12.60.tar.gz
   [netapp-user@rhel7 ~]$ cd astra-control-center-21.12.60
   ```

3. 开始安装之前，请将 Astra Control Center 映像推送到映像注册表。您可以选择使用 Docker 或 Podman 执行此操作，此步骤将提供这两者的说明。

Podman

a. 将 ‘re名称为组织 / 命名空间 / 项目的注册表 FQDN 导出为环境变量 "gregistry" 。

```
[netapp-user@rhel7 ~]$ export REGISTRY=astra-
registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

b. 登录到注册表。

```
[netapp-user@rhel7 ~]$ podman login -u ocp-user -p password
--tls-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com
```

> 如果使用 `kubeadmin` user 登录到专用注册表，请使用 token 代替 password - `podman login -u Ocp-user -p token -tls-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com` 。

> 或者，您也可以创建服务帐户，分配注册表编辑器和 / 或注册表查看器角色（取决于您是否需要推 / 拉访问），并使用服务帐户的令牌登录到注册表。

c. 创建 Shell 脚本文件并将以下内容粘贴到其中。

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded
image trimming the 'Loaded images: '
  astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image(s): //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  podman push ${REGISTRY}/${astraImage}
done
```

> 如果您的注册表使用的是不可信的证书，请编辑 shell 脚本并对 podman 推送命令 `podman 推送 $registry/$ （ echo $astraImage ` s/^^` ……`

d. 使文件可执行

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

e. 执行 shell 脚本。

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```

Docker

a. 将 ‘re名称为组织 / 命名空间 / 项目的注册表 FQDN 导出为环境变量 "gregistry" 。

```
[netapp-user@rhel7 ~]$ export REGISTRY=astra-
registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

b. 登录到注册表。

```
[netapp-user@rhel7 ~]$ docker login -u ocp-user -p password
astra-registry.apps.ocp-vmw.cie.netapp.com
```

> 如果使用 `kubeadmin` user 登录到专用注册表，请使用 token 代替 password - `docker login -u Ocp-user -p token astra-registry.apps.ocp-vmw.cie.netapp.com` 。

> 或者，您也可以创建服务帐户，分配注册表编辑器和 / 或注册表查看器角色（取决于您是否需要推 / 拉访问），并使用服务帐户的令牌登录到注册表。

c. 创建 Shell 脚本文件并将以下内容粘贴到其中。

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded
image trimming the 'Loaded images: '
  astraImage=$(docker load --input ${astraImageFile} | sed
's/Loaded image: //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  docker tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  docker push ${REGISTRY}/${astraImage}
done
```

d. 使文件可执行

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

e. 执行 shell 脚本。

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```

4. 使用非公共信任的私有映像注册表时，请将映像注册表 TLS 证书上传到 OpenShift 节点。为此，请使用 TLS 证书在 OpenShift-config 命名空间中创建一个配置映射，并将其修补到集群映像配置中以使此证书可信。

```
[netapp-user@rhel7 ~]$ oc create configmap default-ingress-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":"default-ingress-
ca"}}}' --type=merge
```

> 如果您使用的是包含传入操作员的默认 TLS 证书的 OpenShift 内部注册表和路由，则仍需要按照上一步将这些证书修补到路由主机名。要从 运算符提取证书，您可以使用命令 oc extract secret/router -ca -keys=tls.crt -n OpenShift-Inuse-operator 。

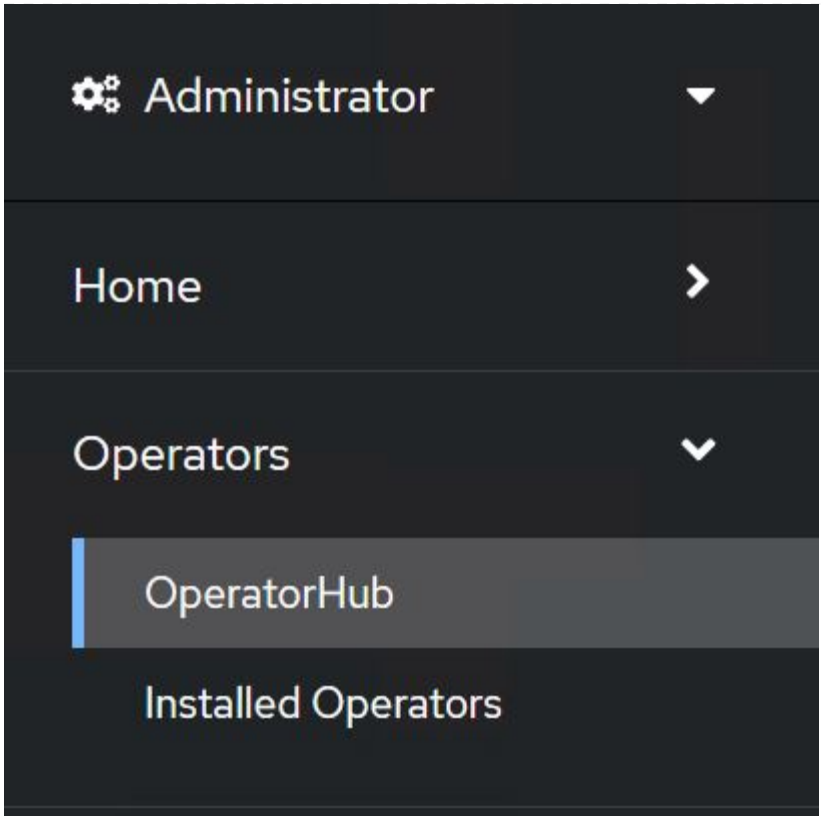
5. 为 Astra 控制中心创建命名空间 `NetApp-Acc-operator` 。

```
[netapp-user@rhel7 ~]$ oc create ns netapp-acc-operator

namespace/netapp-acc-operator created
```

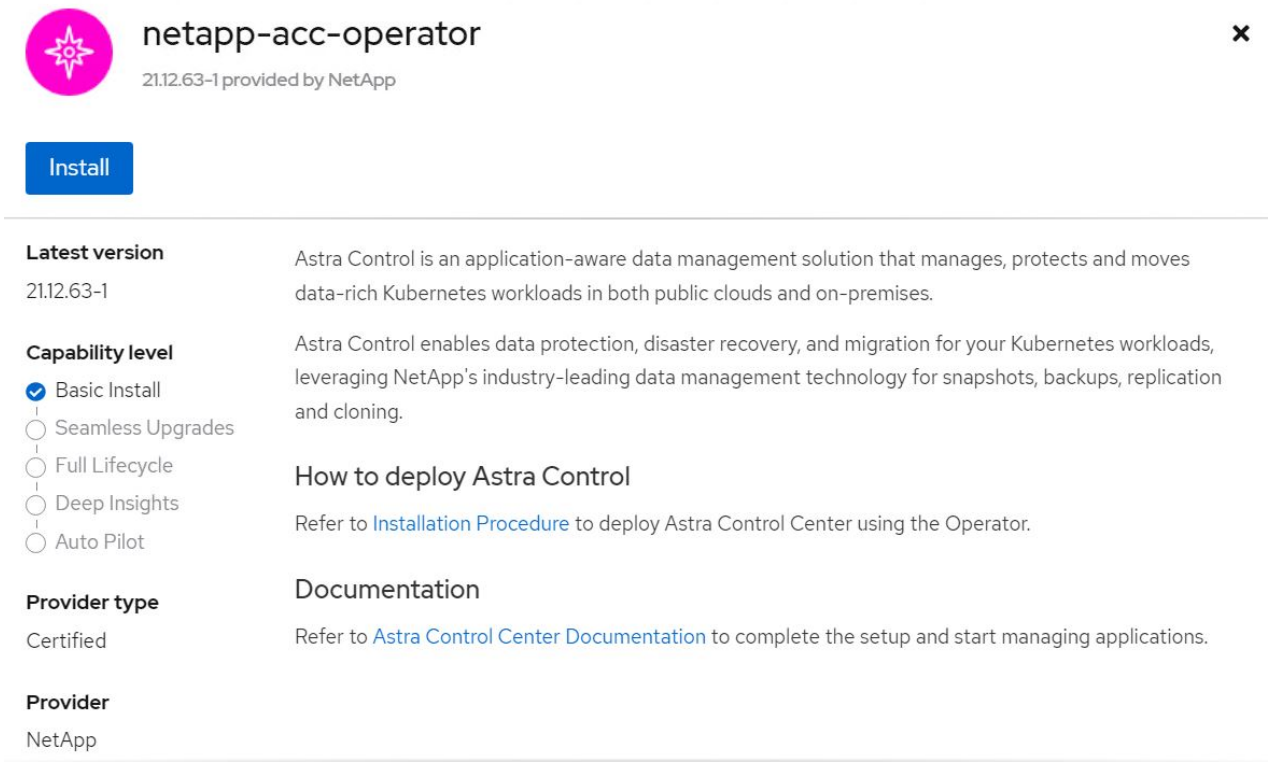
6. 使用凭据创建一个密钥，以登录到 `NetApp-Acc-operator` 命名空间中的映像注册表。

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

secret/astra-registry-cred created
```

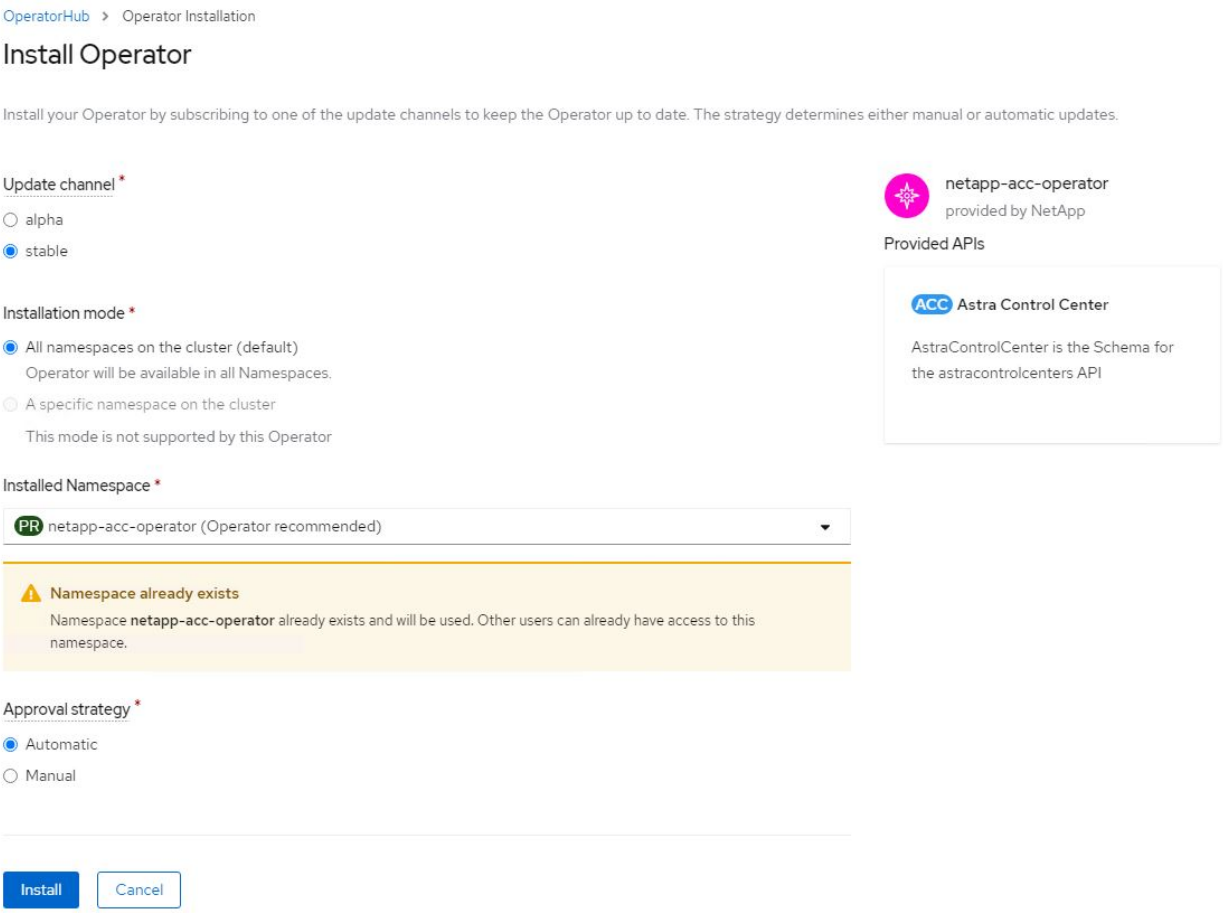
7. 使用 cluster-admin 访问权限登录到 Red Hat OpenShift GUI 控制台。
8. 从 "Perspective" 下拉列表中选择 "Administrator" 。
9. 导航到 Operators > OperatorHub 并搜索 Astra 。



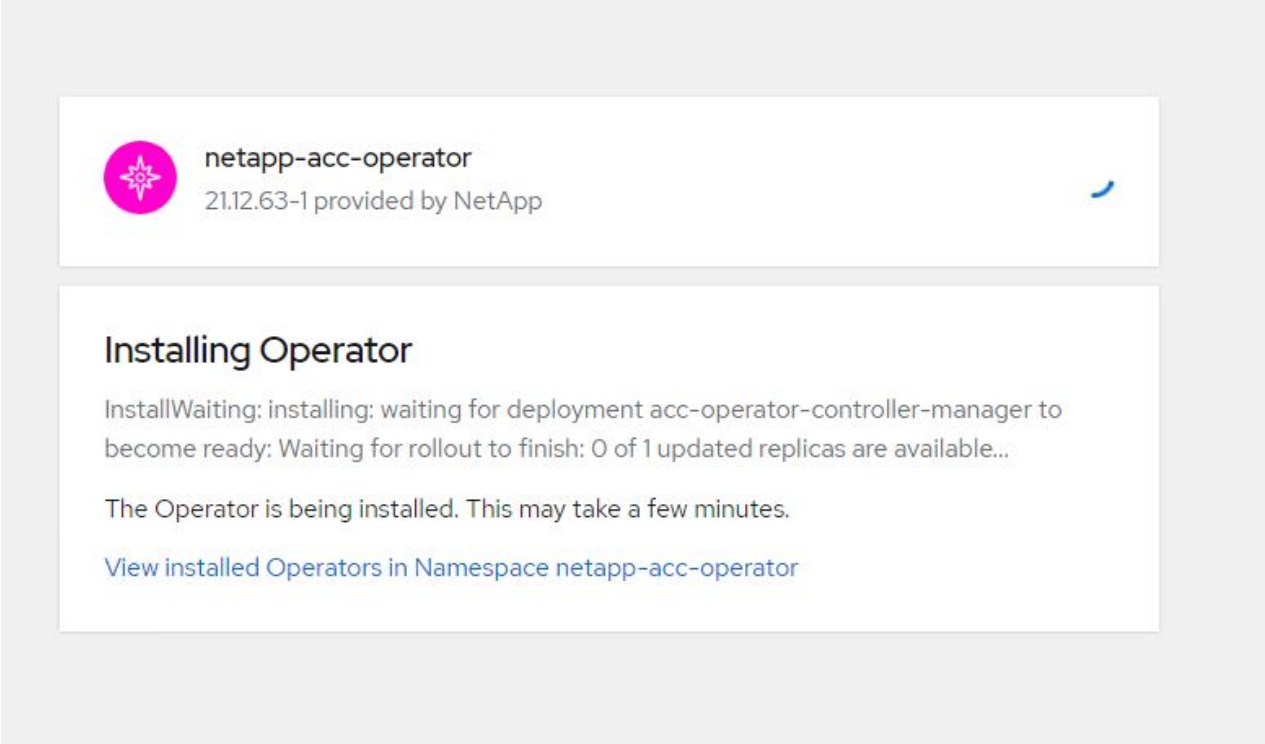
10. 选择 `NetApp-Acc-operator` Tile ，然后单击 `Install` 。



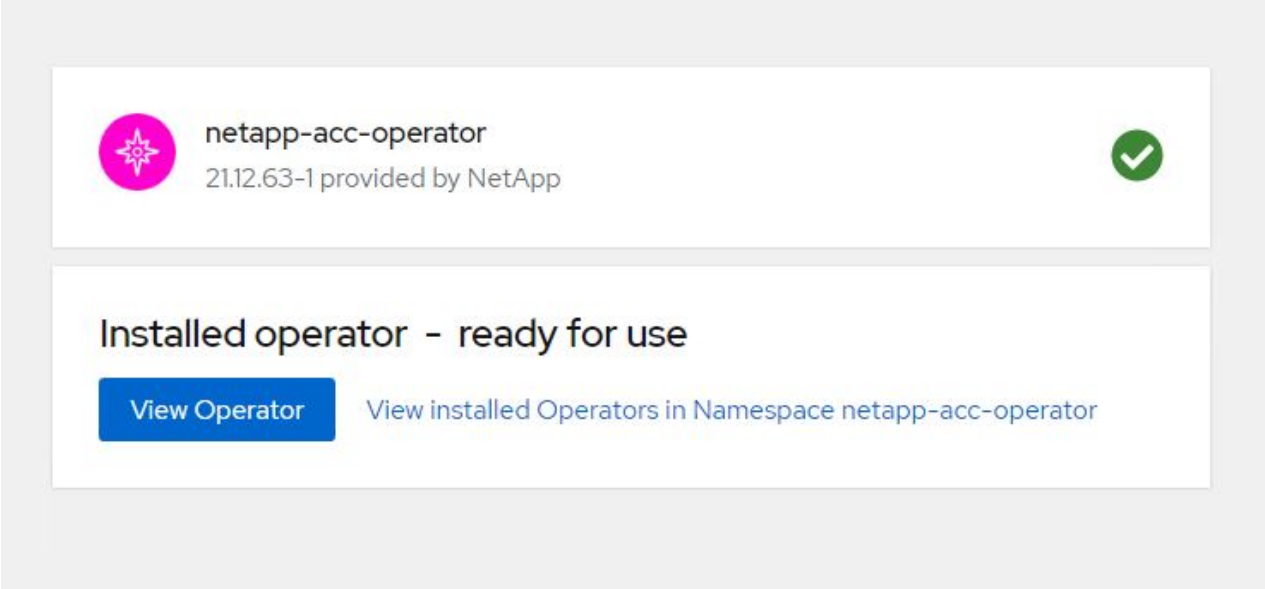
11. 在 Install Operator 屏幕上，接受所有默认参数，然后单击 `Install` 。



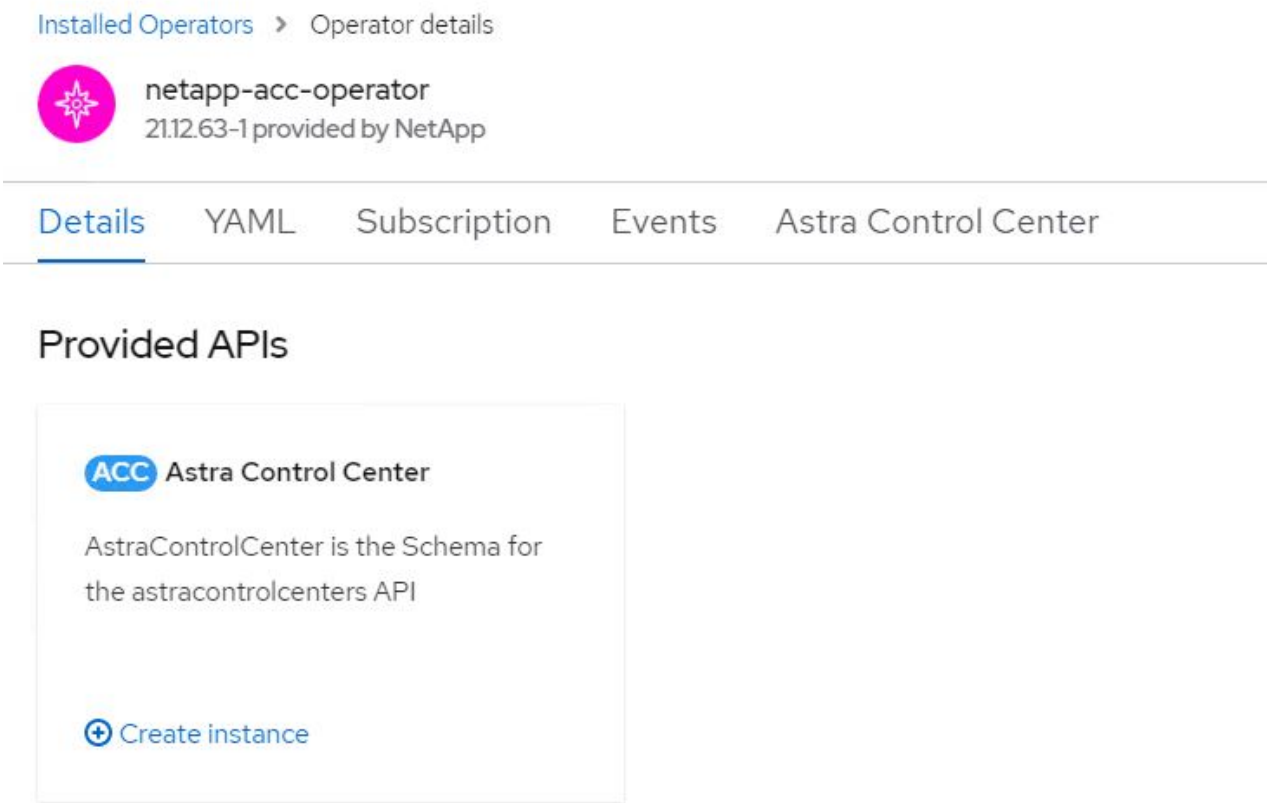
12. 等待操作员安装完成。



13. 操作员安装成功后，导航到单击 `View Operator` 。



14. 然后在运算符中单击 Astra Control Center 图块中的 `Create Instance` 。



15. 填写 `Create AstraControlCenter` Form 字段，然后单击 `Create` 。

    a. 也可以编辑 Astra Control Center 实例名称。
    b. 也可以启用或禁用自动支持。建议保留自动支持功能。
    c. 输入 Astra 控制中心的 FQDN 。
    d. 输入 Astra 控制中心版本；默认情况下会显示最新版本。

e. 输入 Astra 控制中心的帐户名称和管理员详细信息，例如名字，姓氏和电子邮件地址。

f. 输入卷回收策略，默认值为 Retain 。

g. 在映像注册表中，输入注册表的 FQDN 以及在将映像推送到注册表时提供的组织名称（在此示例中为 `astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra` ）

h. 如果您使用的注册表需要进行身份验证，请在映像注册表部分输入机密名称。

i. 为 Astra 控制中心资源限制配置扩展选项。

j. 如果要将 PVC 放置在非默认存储类上，请输入存储类名称。

k. 定义 CRD 处理首选项。

Project: netapp-acc-operator

Name *

astra

Labels

app=frontend

Account Name *

HCG Solutions Engineering

Astra Control Center account name

Astra Address *

astra-control-center.cie.netapp.com

AstraAddress defines how Astra will be found in the data center. This IP address and/or DNS A record must be created prior to provisioning Astra Control Center. Example - "astra.example.com" The A record and its IP address must be allocated prior to provisioning Astra Control Center

Astra Version *

21.12.60

Version of AstraControlCenter to deploy. You are provided a Helm repository with a corresponding version. Example - 1.5.2, 1.4.2-patch

Email *

solutions_tme@netapp.com

EmailAddress will be notified by Astra as events warrant.

Auto Support *

AutoSupport indicates willingness to participate in NetApp's proactive support application, NetApp Active IQ. The default election is true and indicates support data will be sent to NetApp. An empty or blank election is the same as a default election. Air gapped installations should enter false.

First Name

HCG

The first name of the SRE supporting Astra.

Last Name

Admin

The last name of the SRE supporting Astra.

Image Registry

The container image registry that is hosting the Astra application images, ACC Operator and ACC Helm Repository.

Name

astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra

The name of the image registry. For example "example.registry/astra". Do not prefix with protocol.

Secret

astra-registry-cred

The name of the Kubernetes secret that will authenticate with the image registry.

Volume Reclaim Policy

Retain

Reclaim policy to be set for persistent volumes

Astra Resources Scaler

Default

Scaling options for AstraControlCenter Resource limits.

Storage Class

The storage class to be used for PVCs. If not set, default storage class will be used.

Crds

Options for how ACC should handle CRDs.

Create Cancel

自动化的〔可逆〕

1. 要使用Ansible攻略手册部署Astra控制中心、您需要安装安装有Ansible的Ubuntu或RHEL计算机。按照步骤进行操作 "此处" 适用于Ubuntu和RHEL。
2. 克隆托管 Ansible 内容的 GitHub 存储库。

```
git clone https://github.com/NetApp-
Automation/na_astra_control_suite.git
```

3. 登录到NetApp支持站点并下载最新版本的NetApp Astra控制中心。为此，您需要在 NetApp 帐户中附加许可证。下载完 tarball 后，将其传输到工作站。

> 要开始获取 Astra Control 的试用许可证，请访问 "Astra 注册站点"。

4. 创建或获取对要安装Astra控制中心的OpenShift集群具有管理员访问权限的kubeconfig文件。
5. 将目录更改为 na_astera_control_suite 。

```
cd na_astra_control_suite
```

6. 编辑`vars/vars.yml`文件、并使用所需信息填充变量。

```
#Define whether or not to push the Astra Control Center images to
your private registry [Allowed values: yes, no]
push_images: yes

#The directory hosting the Astra Control Center installer
installer_directory: /home/admin/

#Specify the ingress type. Allowed values - "AccTraefik" or
"Generic"
#"AccTraefik" if you want the installer to create a LoadBalancer
type service to access ACC, requires MetalLB or similar.
#"Generic" if you want to create or configure ingress controller
yourself, installer just creates a ClusterIP service for traefik.
ingress_type: "AccTraefik"

#Name of the Astra Control Center installer (Do not include the
extension, just the name)
astra_tar_ball_name: astra-control-center-22.04.0

#The complete path to the kubeconfig file of the
kubernetes/openshift cluster Astra Control Center needs to be
installed to.
hosting_k8s_cluster_kubeconfig_path: /home/admin/cluster-
kubeconfig.yml

#Namespace in which Astra Control Center is to be installed
astra_namespace: netapp-astra-cc

#Astra Control Center Resources Scaler. Leave it blank if you want
to accept the Default setting.
astra_resources_scaler: Default

#Storageclass to be used for Astra Control Center PVCs, it must be
created before running the playbook [Leave it blank if you want the
PVCs to use default storageclass]
astra_trident_storageclass: basic

#Reclaim Policy for Astra Control Center Persistent Volumes [Allowed
values: Retain, Delete]
storageclass_reclaim_policy: Retain
```

```
#Private Registry Details
astra_registry_name: "docker.io"

#Whether the private registry requires credentials [Allowed values:
yes, no]
require_reg_creds: yes

#If require_reg_creds is yes, then define the container image
registry credentials
#Usually, the registry namespace and usernames are same for
individual users
astra_registry_namespace: "registry-user"
astra_registry_username: "registry-user"
astra_registry_password: "password"

#Kuberenets/OpenShift secret name for Astra Control Center
#This name will be assigned to the K8s secret created by the
playbook
astra_registry_secret_name: "astra-registry-credentials"

#Astra Control Center FQDN
acc_fqdn_address: astra-control-center.cie.netapp.com

#Name of the Astra Control Center instance
acc_account_name: ACC Account Name

#Administrator details for Astra Control Center
admin_email_address: admin@example.com
admin_first_name: Admin
admin_last_name: Admin
```

7. 运行攻略手册以部署 Astra 控制中心。对于某些配置、此攻略手册需要root特权。

   如果运行该攻略手册的用户为root或配置了无密码sudo、请运行以下命令运行该攻略手册。

```
ansible-playbook install_acc_playbook.yml
```

   如果用户配置了基于密码的sudo访问权限、请运行以下命令以运行攻略手册、然后输入sudo密码。

```
ansible-playbook install_acc_playbook.yml -K
```

1. 完成安装可能需要几分钟时间。验证 `NetApp-Astra-cc` 命名空间中的所有 Pod 和服务是否均已启动且正在运行。

```
[netapp-user@rhel7 ~]$ oc get all -n netapp-astra-cc
```

2. 检查 `Acc-operator-controller-manager` 日志以确保安装已完成。

```
[netapp-user@rhel7 ~]$ oc logs deploy/acc-operator-controller-manager -n
netapp-acc-operator -c manager -f
```

以下消息指示 Astra 控制中心已成功安装。

```
{"level":"info","ts":1624054318.029971,"logger":"controllers.AstraContro
lCenter","msg":"Successfully Reconciled AstraControlCenter in
[seconds]s","AstraControlCenter":"netapp-astra-
cc/astra","ae.Version":"[21.12.60]"}
```

3. 用于登录到 Astra 控制中心的用户名是 CRD 文件中提供的管理员电子邮件地址，密码是附加到 Astra 控制中心 UUID 的字符串 `Acc-` 。运行以下命令：

```
[netapp-user@rhel7 ~]$ oc get astracontrolcenters -n netapp-astra-cc
NAME    UUID
astra   345c55a5-bf2e-21f0-84b8-b6f2bce5e95f
```

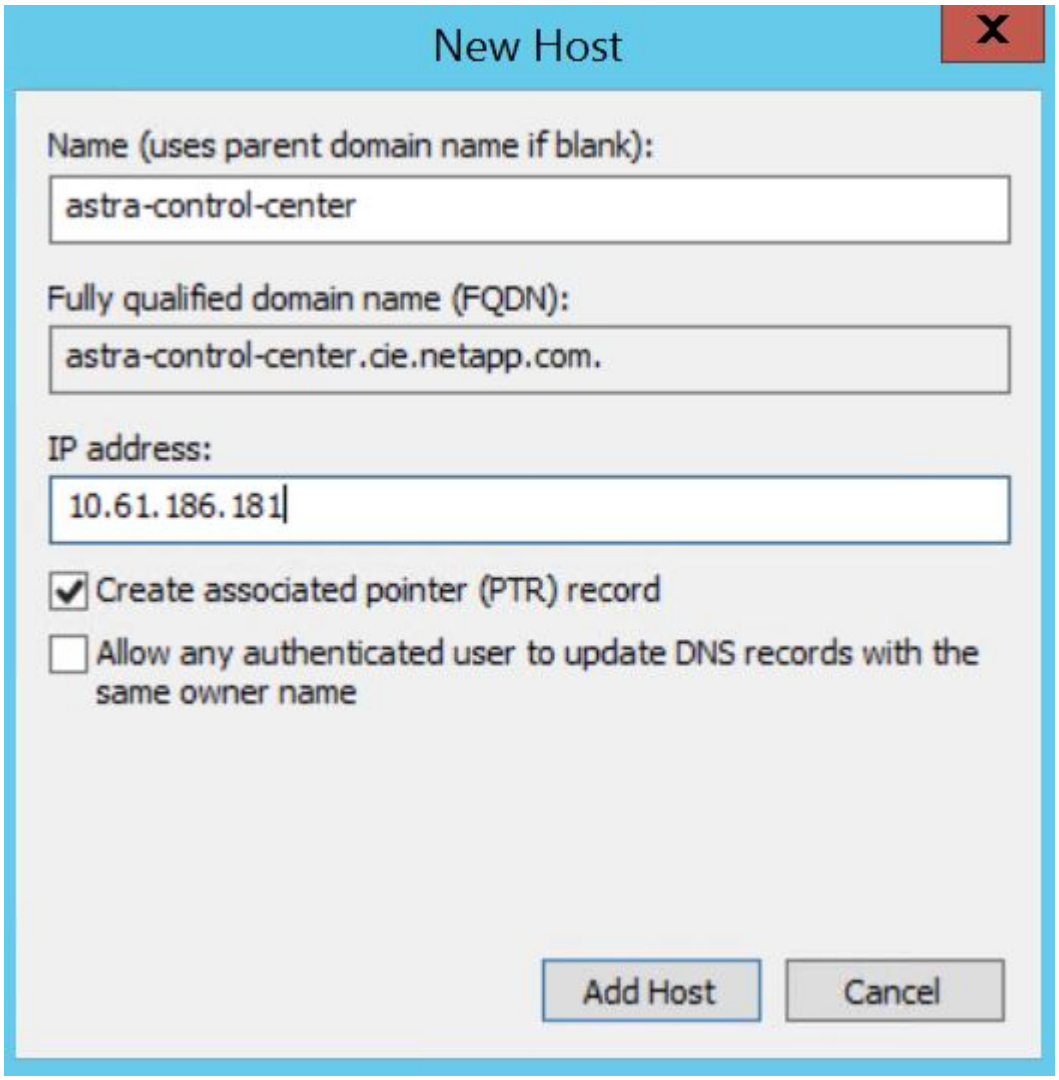
在此示例中，密码为 `Acc-345c55a5-bf2e-21f0-84b8-b6f2bce5e95f` 。

4. 获取 traefik 服务负载平衡器 IP 。

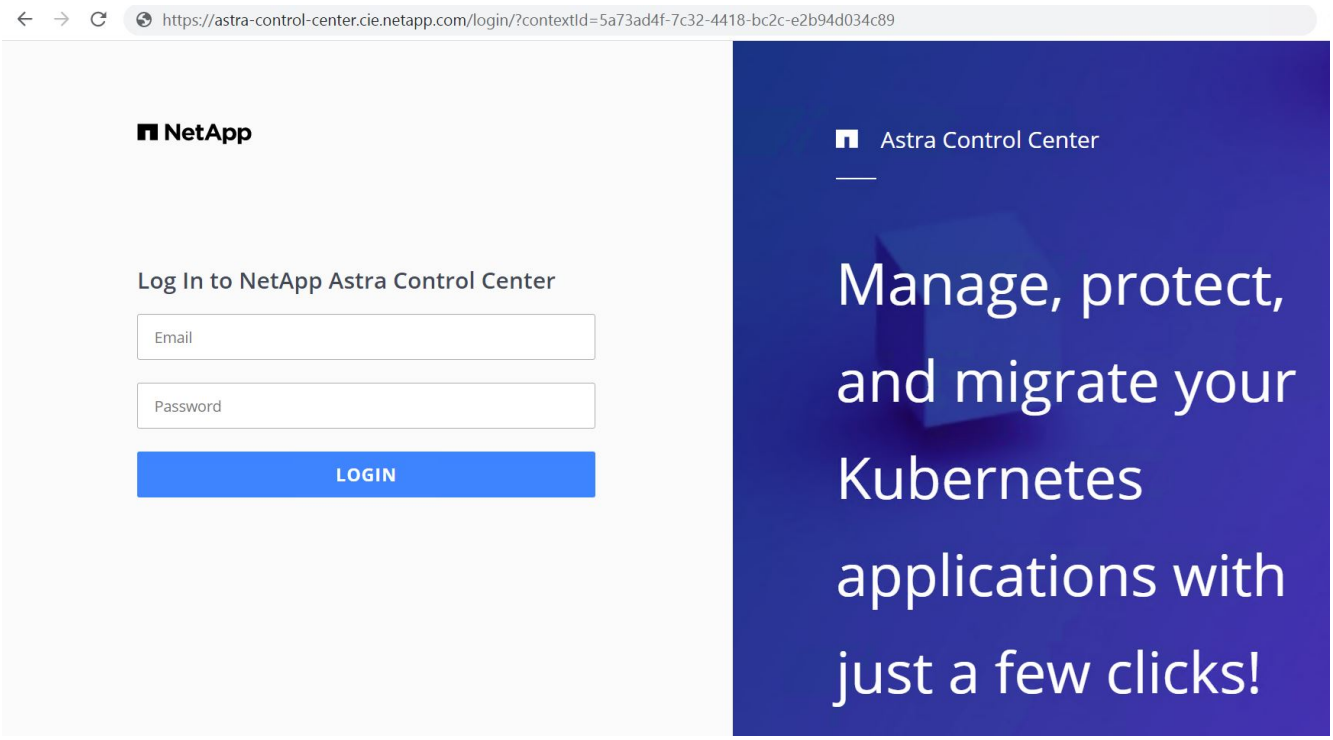
```
[netapp-user@rhel7 ~]$ oc get svc -n netapp-astra-cc | egrep
'EXTERNAL|traefik'

NAME                                        TYPE           CLUSTER-IP
EXTERNAL-IP     PORT(S)
AGE
traefik                                     LoadBalancer   172.30.99.142
10.61.186.181   80:30343/TCP,443:30060/TCP
16m
```

5. 在 DNS 服务器中添加一个条目，将 Astra 控制中心 CRD 文件中提供的 FQDN 指向 traefik 服务的

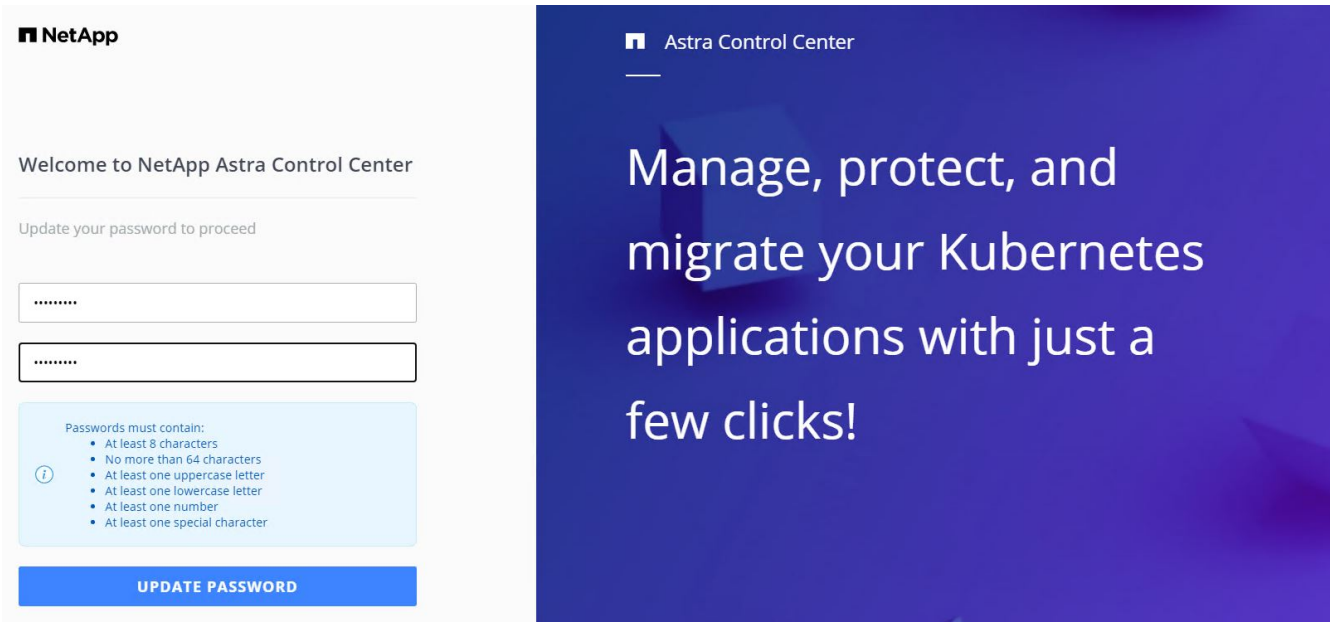
`external-IP`。



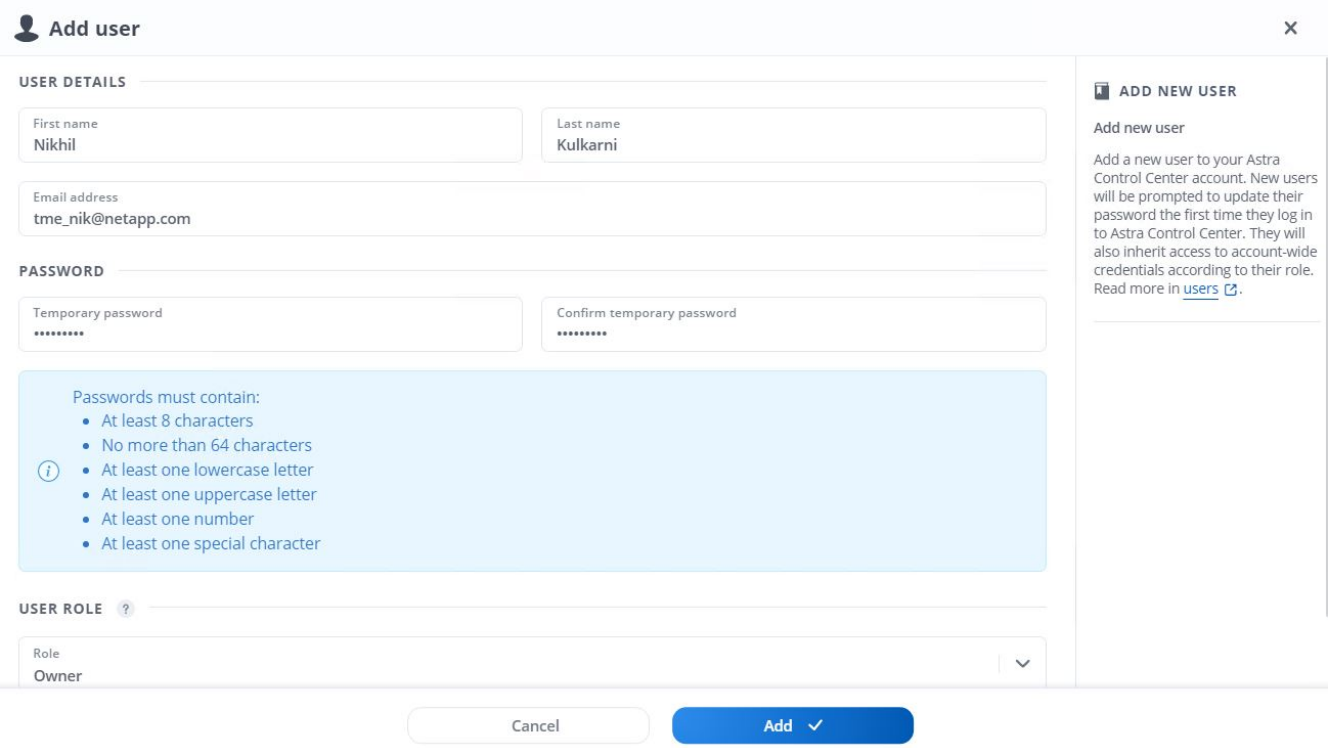
6. 通过浏览 Astra 控制中心的 FQDN 登录到该 GUI 。



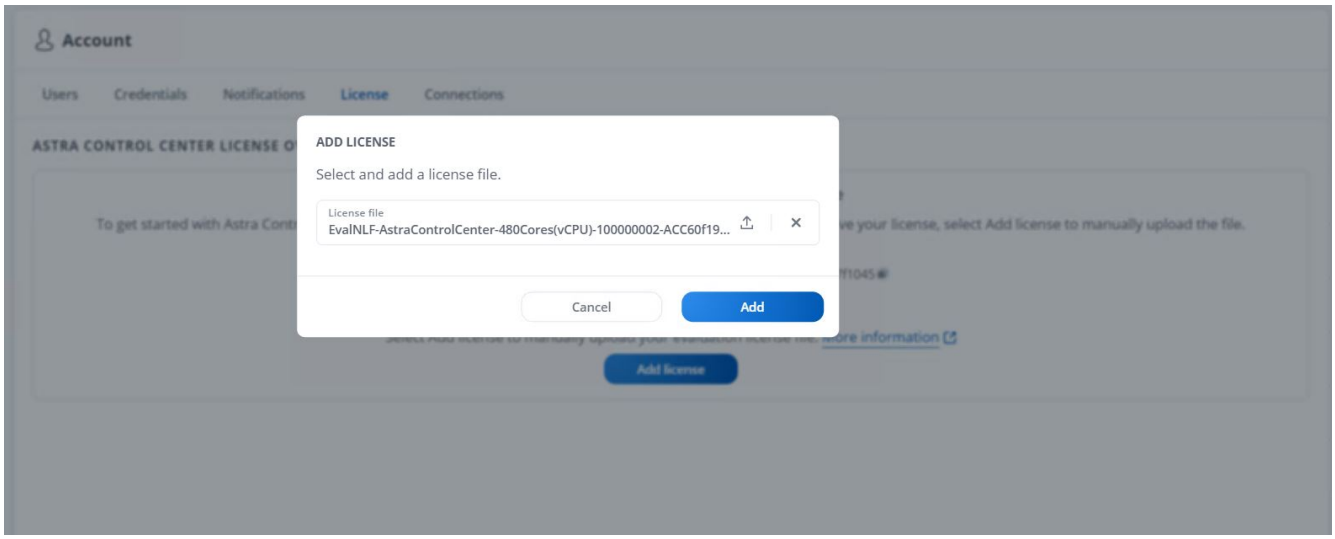
7. 首次使用 CRD 中提供的管理员电子邮件地址登录到 Astra 控制中心图形用户界面时，您需要更改密码。



8. 如果要将用户添加到 Astra 控制中心，请导航到 Account > Users ，单击 Add ，输入用户的详细信息，然后单击 Add 。



9. 要使 Astra 控制中心的所有功能正常运行，需要获得许可证。要添加许可证，请导航到 " 帐户 ">" 许可证 " ，单击 " 添加许可证 " ，然后上传许可证文件。



> 如果您在安装或配置 NetApp Astra 控制中心时遇到问题，可以参考已知问题的知识库 "此处 "。

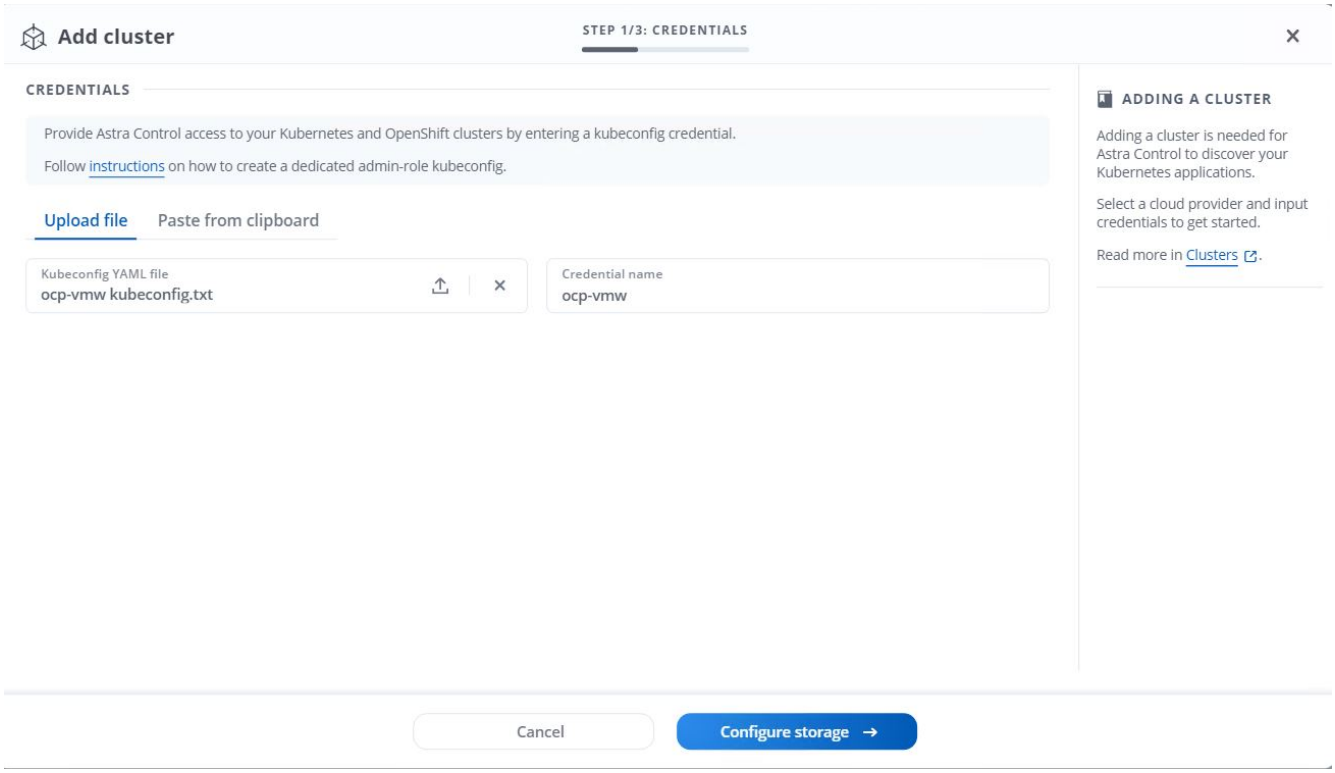
将 **Red Hat OpenShift** 集群注册到 **Astra** 控制中心

要使 Astra 控制中心能够管理您的工作负载，您必须先注册 Red Hat OpenShift 集群。

注册 **Red Hat OpenShift** 集群

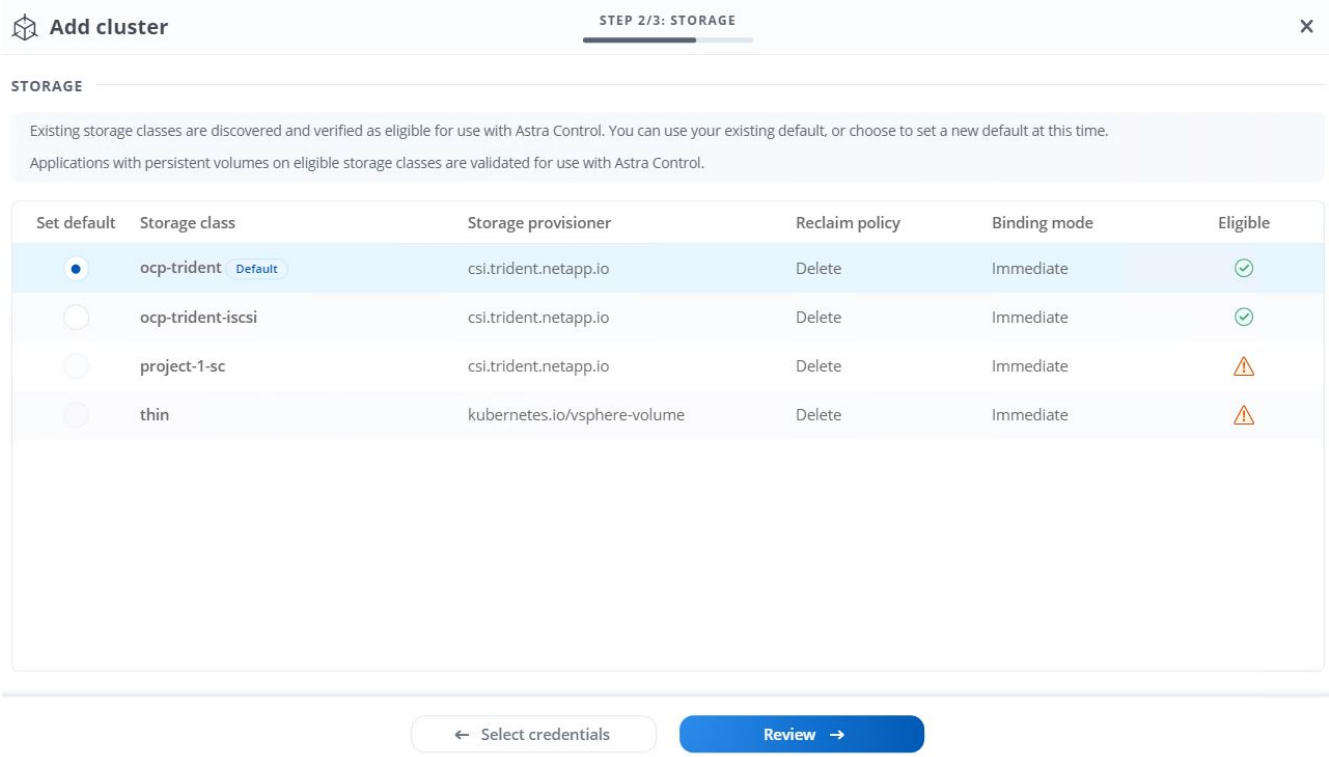
1. 第一步是将 OpenShift 集群添加到 Astra 控制中心并对其进行管理。转至集群并单击添加集群，上传

OpenShift 集群的 kubeconfig 文件，然后单击选择存储。

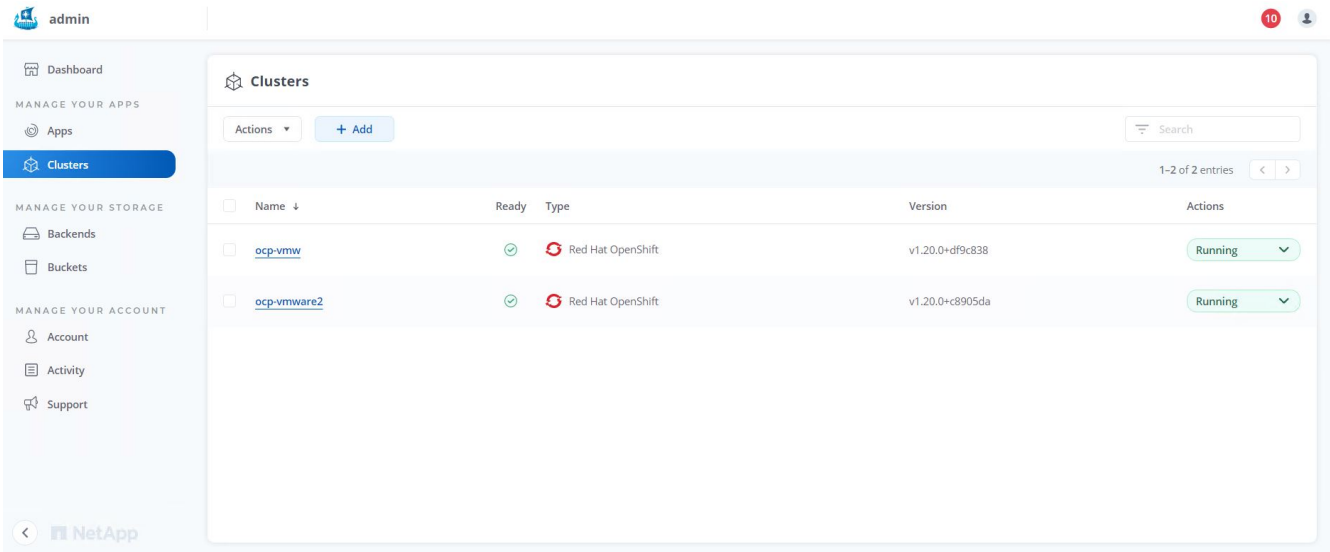


> 可以生成 kubeconfig 文件，以便使用用户名和密码或令牌进行身份验证。令牌将在一段有限的时间后过期，并且可能会使注册的集群无法访问。NetApp 建议使用具有用户名和密码的 kubeconfig 文件将 OpenShift 集群注册到 Astra 控制中心。

2. Astra 控制中心会检测符合条件的存储类。现在，选择使用 NetApp ONTAP 上由 SVM 支持的 Trident 配置卷的 storageclass 方式，然后单击查看。在下一个窗格中，验证详细信息，然后单击 Add Cluster 。

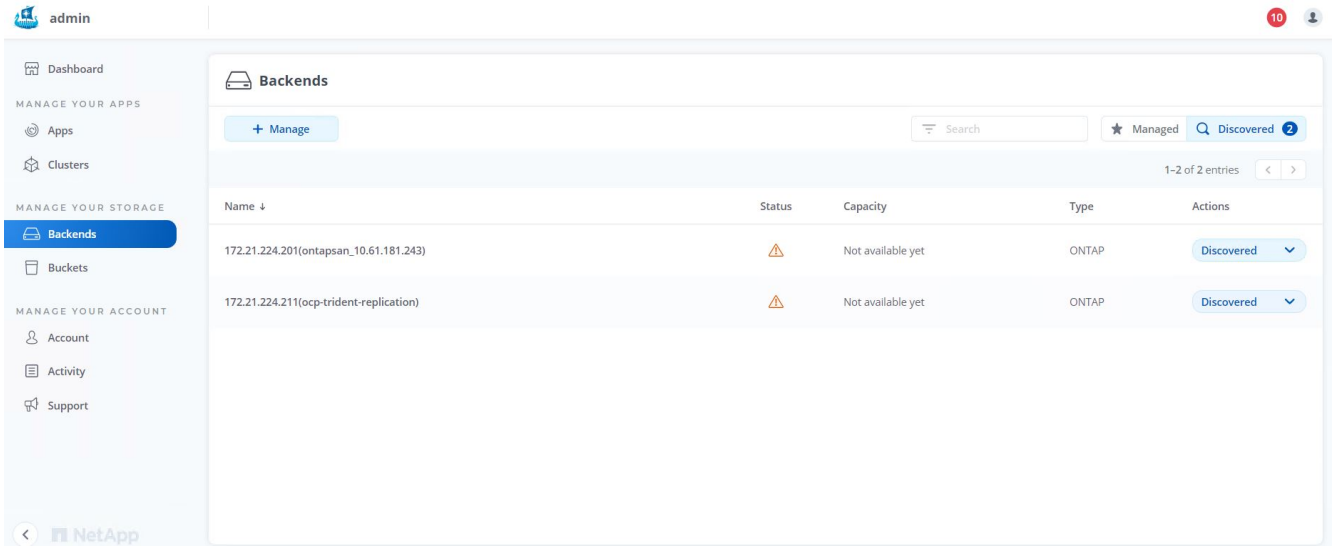


3. 按照步骤 1 中所述注册两个 OpenShift 集群。添加后，集群将变为 " 正在发现 " 状态，而 Astra 控制中心将对其进行检查并安装必要的代理。成功注册后，集群状态将更改为 " 正在运行 " 。

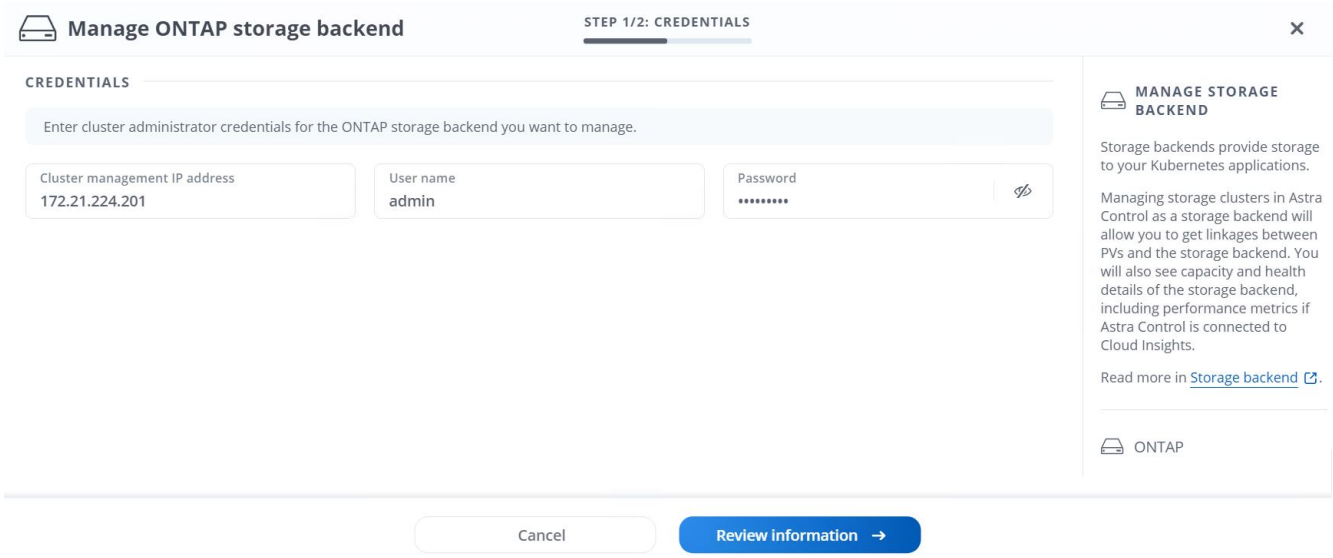


> 在受管集群上安装的代理从该注册表中提取映像时，由 Astra 控制中心管理的所有 Red Hat OpenShift 集群都应有权访问用于安装的映像注册表。

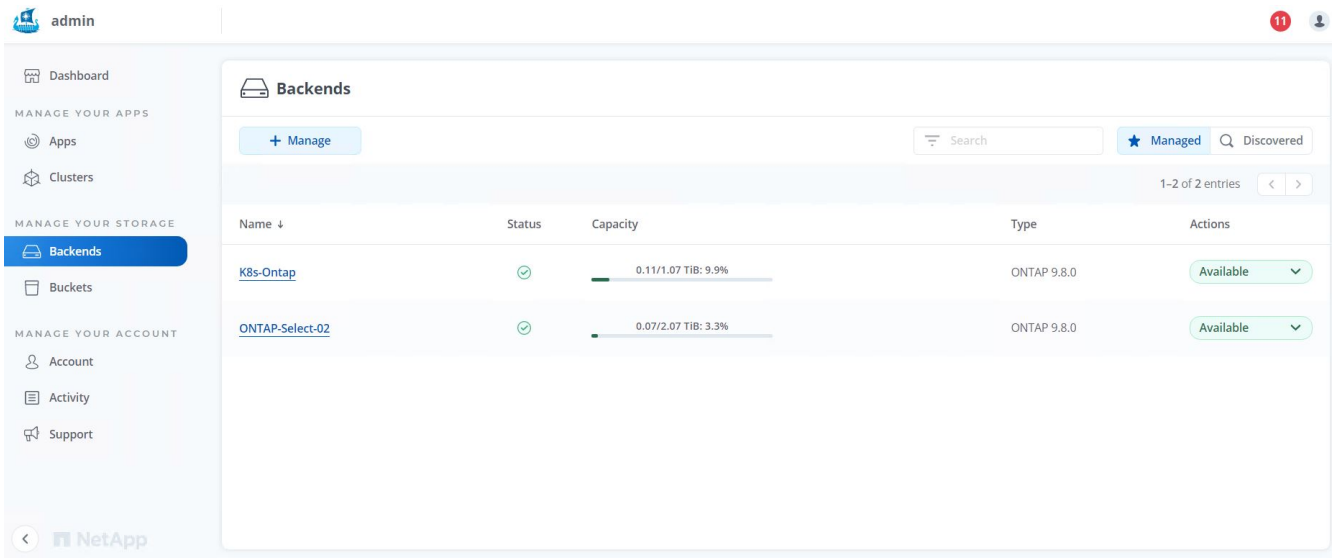
4. 将 ONTAP 集群作为存储资源导入，以便由 Astra 控制中心作为后端进行管理。将 OpenShift 集群添加到 Astra 并配置了 storageclass 后，它会自动发现并检查支持该 storageclass 的 ONTAP 集群，但不会将其导入到要管理的 Astra 控制中心中。



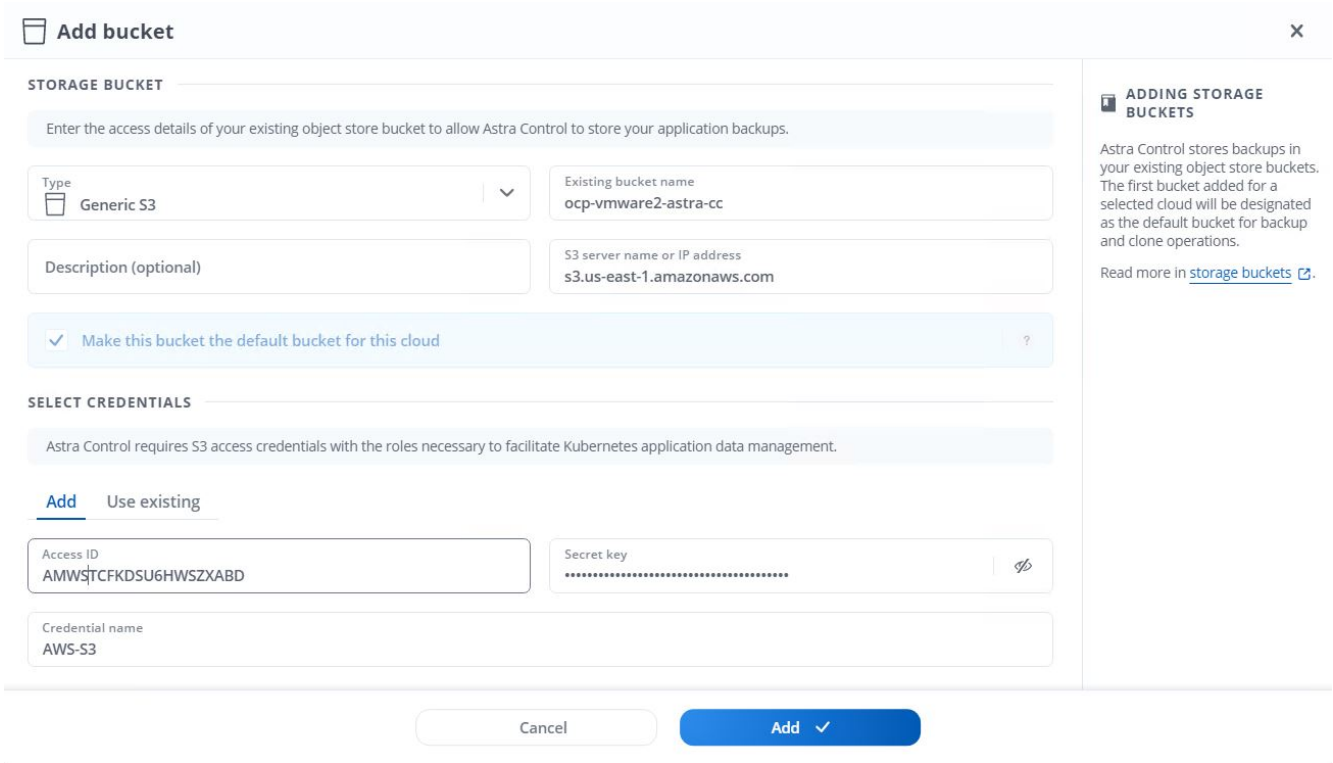
5. 要导入 ONTAP 集群，请转到后端，单击下拉列表，然后选择要管理的 ONTAP 集群旁边的管理。输入 ONTAP 集群凭据，单击查看信息，然后单击导入存储后端。



6. 添加后端后，状态将更改为 Available 。现在，这些后端可提供有关 OpenShift 集群中的永久性卷以及 ONTAP 系统上的相应卷的信息。



7. 要使用 Astra 控制中心在 OpenShift 集群之间进行备份和还原，您必须配置支持 S3 协议的对象存储分段。目前支持的选项包括 ONTAP S3 ， StorageGRID 和 AWS S3 。为此，我们将配置一个 AWS S3 存储分段。转到 " 分段 " ，单击 " 添加分段 " ，然后选择 " 通用 S3" 。输入有关 S3 存储分段和凭据的详细信息以访问它，单击复选框 " 将此存储分段设置为云的默认存储分段 " ，然后单击添加。

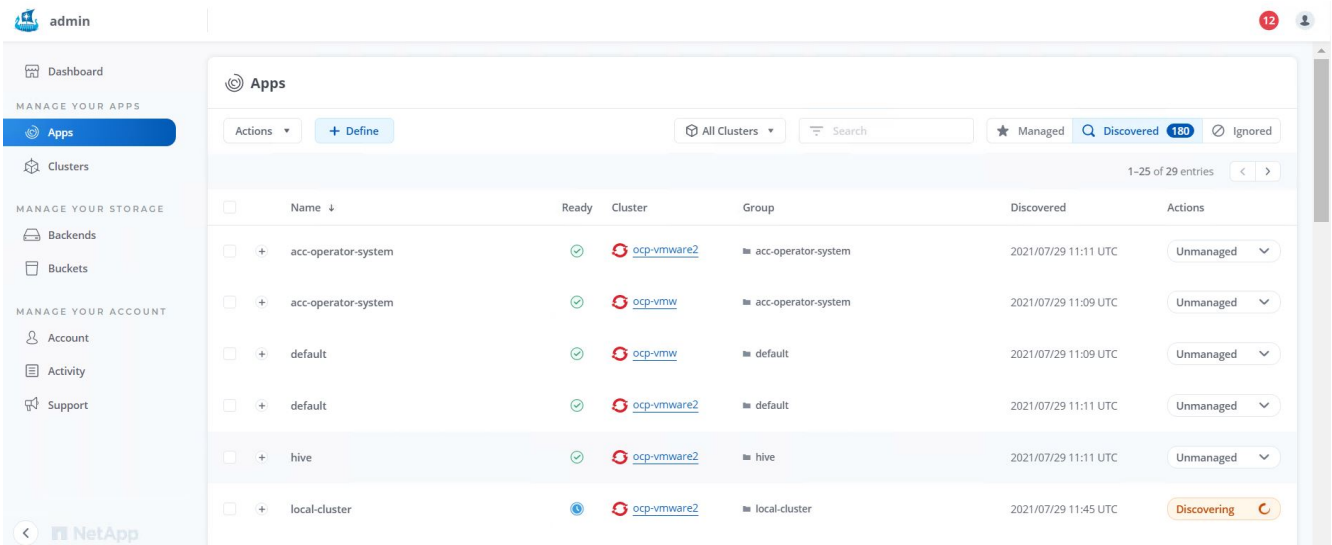


选择要保护的应用程序

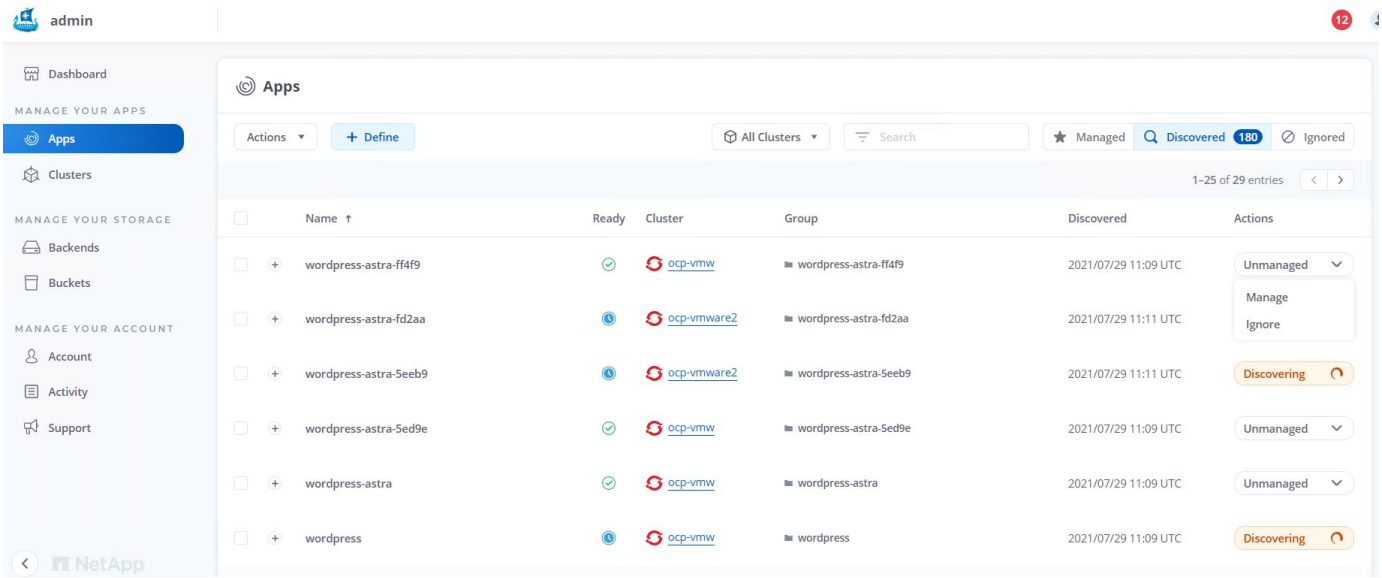
注册 Red Hat OpenShift 集群后，您可以通过 Astra 控制中心发现已部署的应用程序并对其进行管理。

管理应用程序

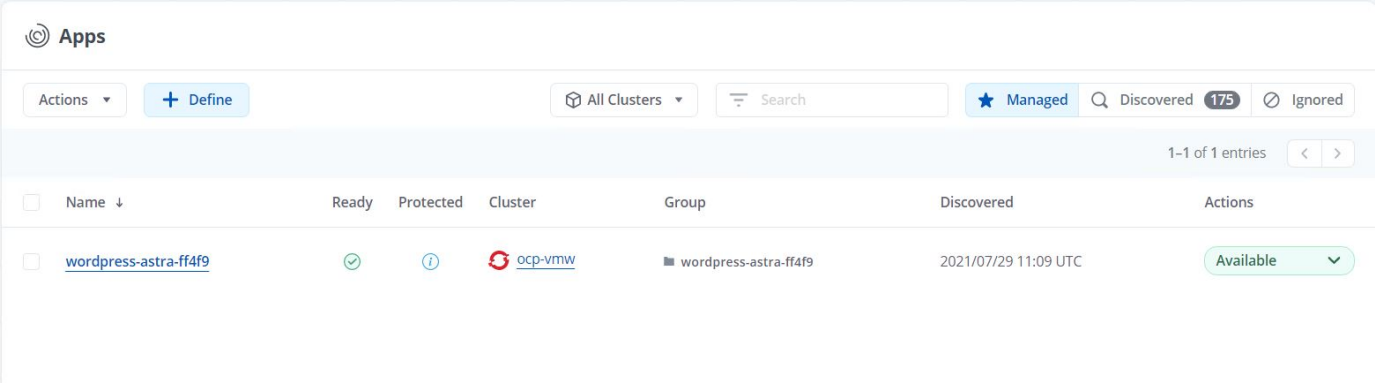
1. 在将 OpenShift 集群和 ONTAP 后端注册到 Astra 控制中心后，控制中心将自动开始发现所有命名空间中使用使用使用指定 ONTAP 后端配置的 storageclass 的应用程序。



2. 导航到应用程序 > 已发现，然后单击要使用 Astra 管理的应用程序旁边的下拉菜单。然后单击管理。



1. 此应用程序将进入可用状态，并可在 "Apps" 部分的 "Managed " 选项卡下查看。



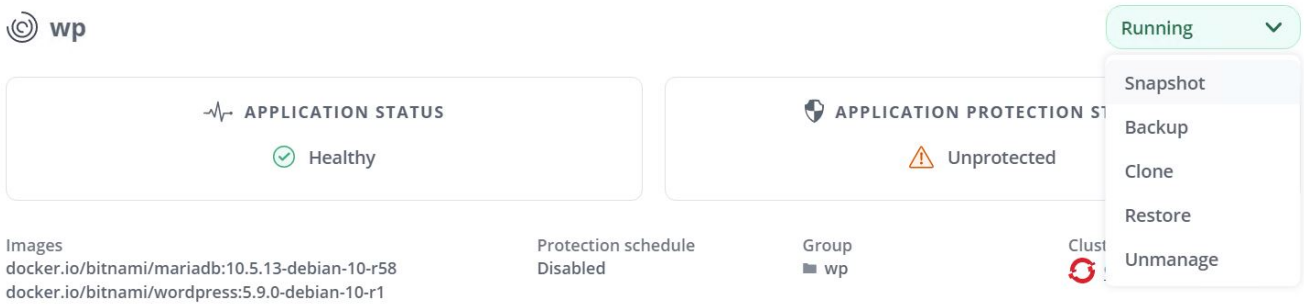
保护您的应用程序

在由 Astra 控制中心管理应用程序工作负载之后，您可以为这些工作负载配置保护设置。

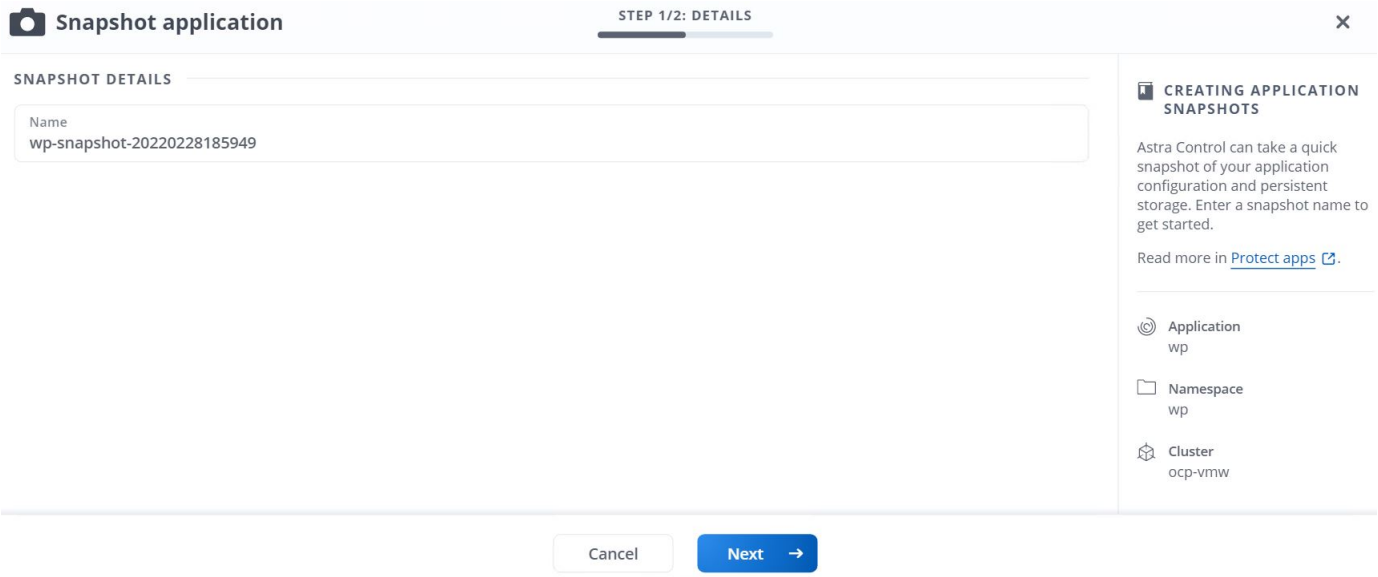
创建应用程序快照

应用程序的快照会创建一个 ONTAP Snapshot 副本，该副本可用于根据该 Snapshot 副本将应用程序还原或克隆到特定时间点。

1. 要为应用程序创建快照，请导航到 "Apps" > "Managed " 选项卡，然后单击要为其创建 Snapshot 副本的应用程序。单击应用程序名称旁边的下拉菜单，然后单击 Snapshot 。



2. 输入快照详细信息，单击下一步，然后单击 Snapshot 。创建快照大约需要一分钟，在成功创建快照后，状态将变为可用。



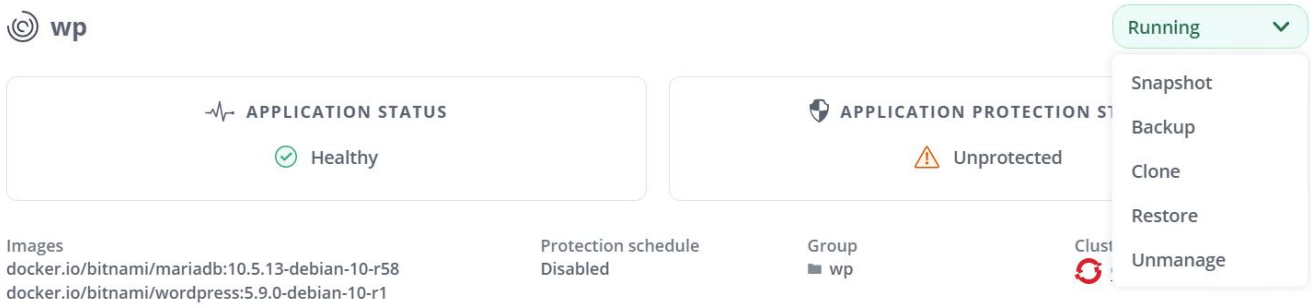
创建应用程序备份

应用程序的备份可捕获应用程序的活动状态及其资源的配置，将其覆盖到文件中，并将其存储在远程对象存储分段中。

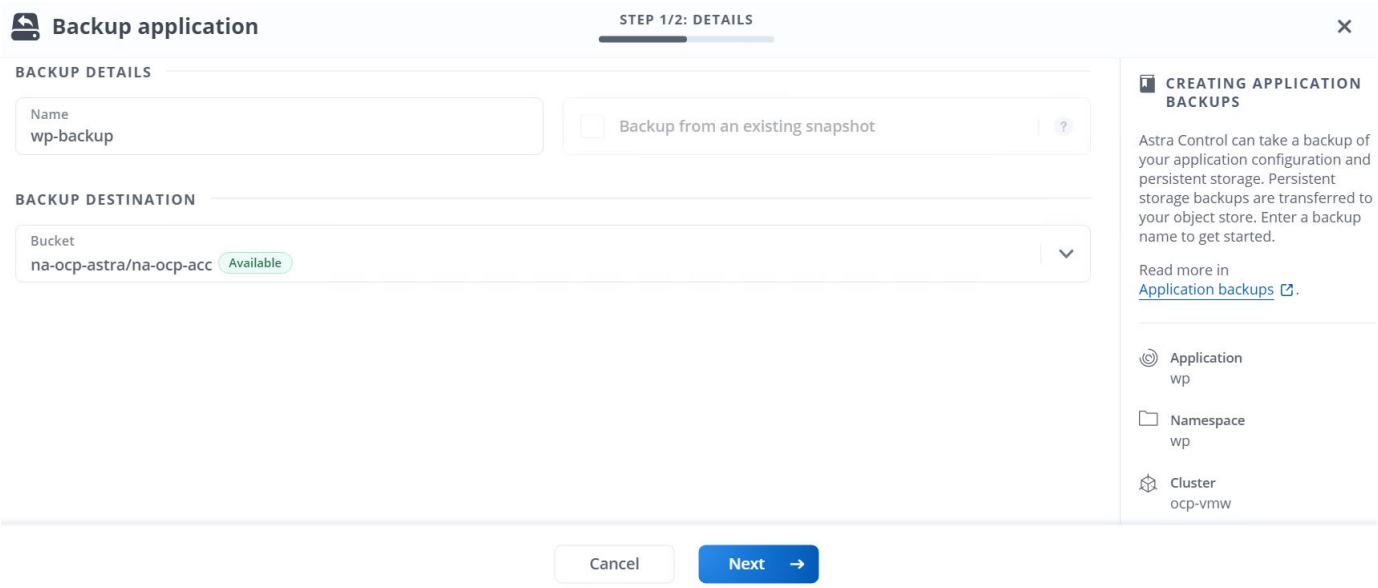
要在 Astra 控制中心备份和还原受管应用程序，必须先为支持的 ONTAP 系统配置超级用户设置。为此，请输入以下命令。

```
ONTAP::> export-policy rule modify -vserver ocp-trident -policyname
default -ruleindex 1 -superuser sys
ONTAP::> export-policy rule modify -policyname default -ruleindex 1 -anon
65534 -vserver ocp-trident
```

1. 要在 Astra 控制中心创建受管应用程序的备份，请导航到应用程序 > 受管选项卡，然后单击要备份的应用程序。单击应用程序名称旁边的下拉菜单，然后单击备份。



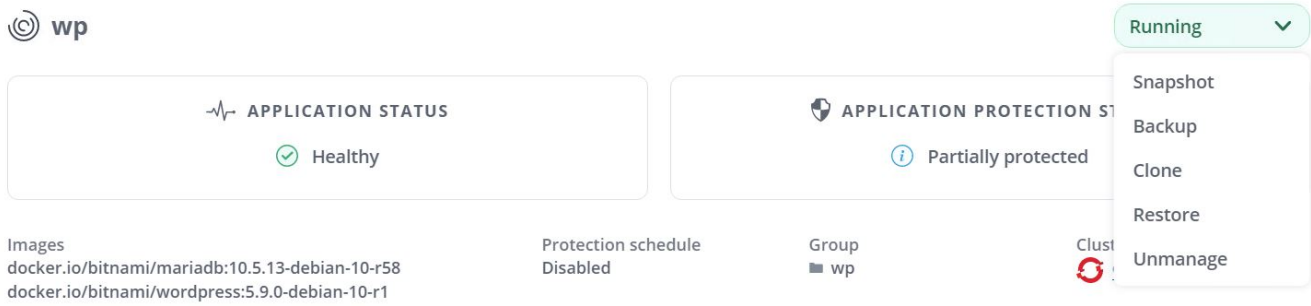
2. 输入备份详细信息，选择用于保存备份文件的对象存储分段，单击下一步，查看详细信息后，单击备份。根据应用程序和数据的大小，备份可能需要几分钟的时间，备份成功完成后，备份状态将变为可用。



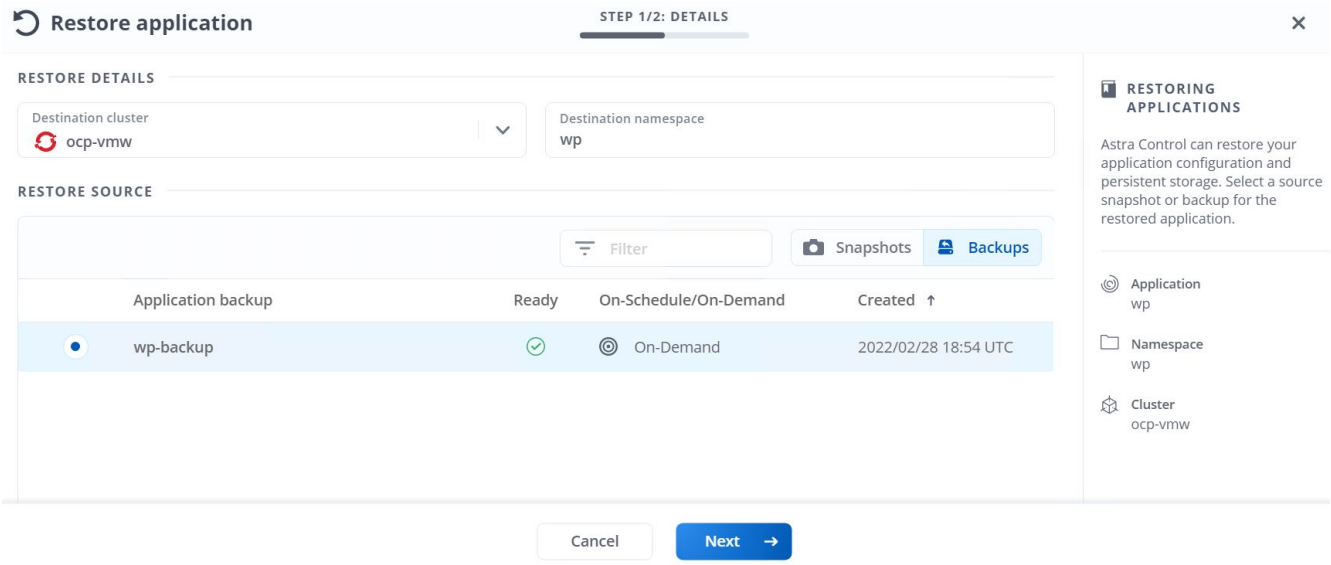
还原应用程序

只需按一个按钮，即可将应用程序还原到同一集群中的原始命名空间或远程集群，以实现应用程序保护和灾难恢复。

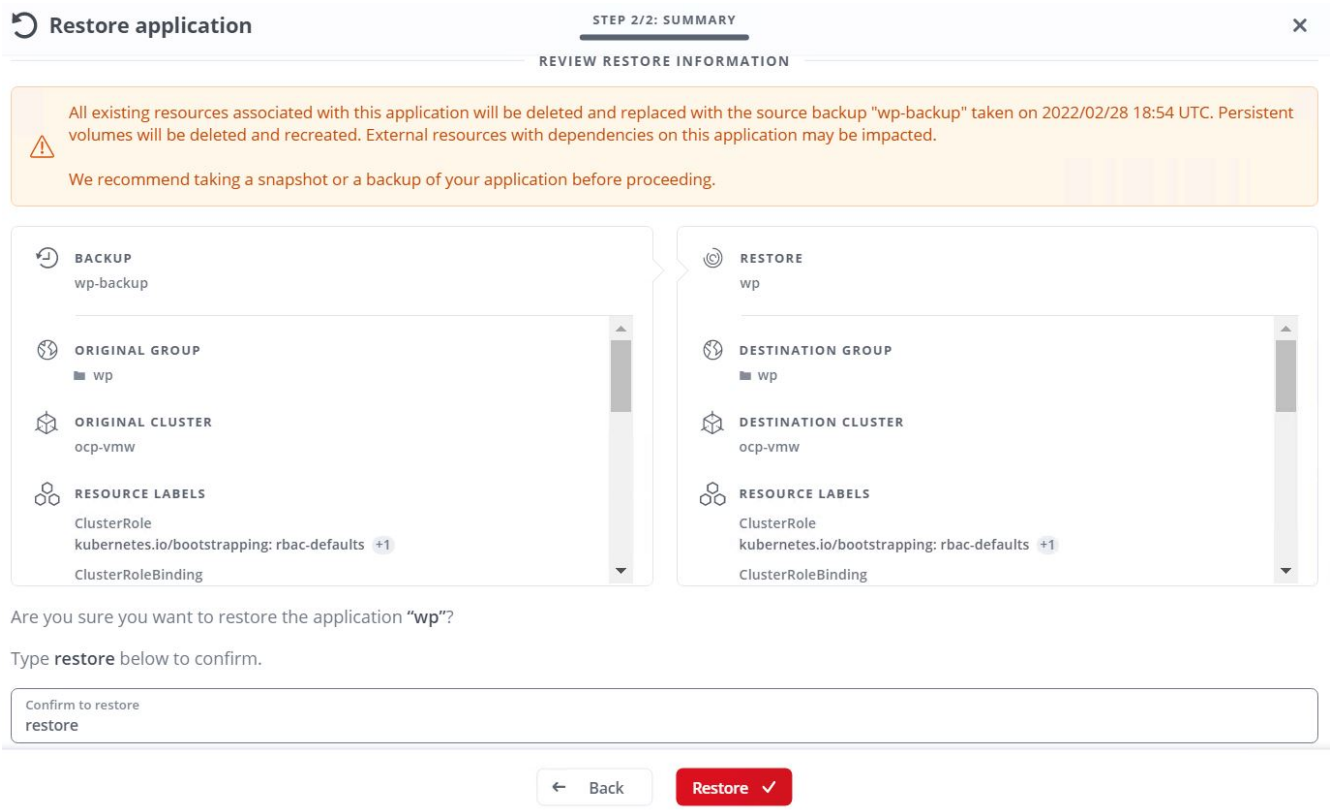
1. 要还原应用程序，请导航到应用程序 > 受管选项卡，然后单击相关应用程序。单击应用程序名称旁边的下拉菜单，然后单击 `Restore` 。



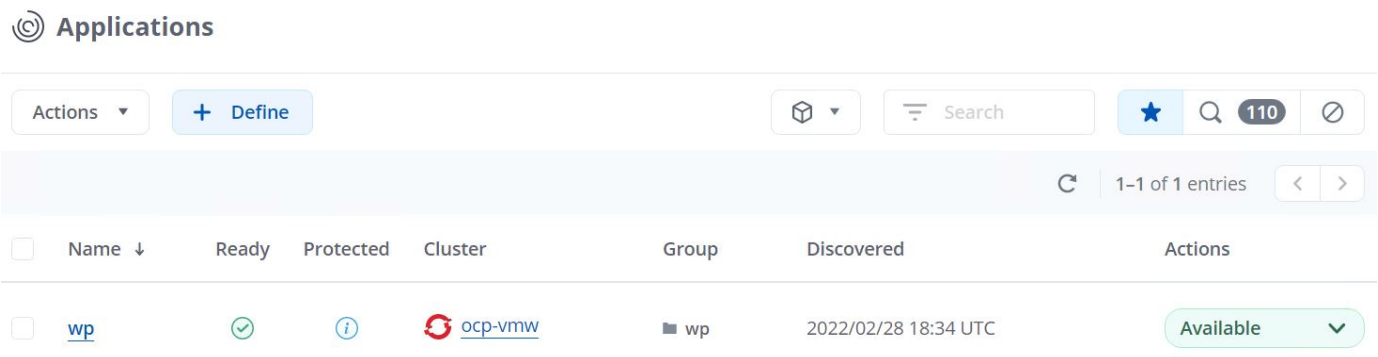
2. 输入还原命名空间的名称，选择要将其还原到的集群，然后选择是要从现有快照还是从应用程序的备份还原它。单击下一步。



3. 在查看窗格中，输入 `restore` ，然后在查看详细信息后单击 Restore 。



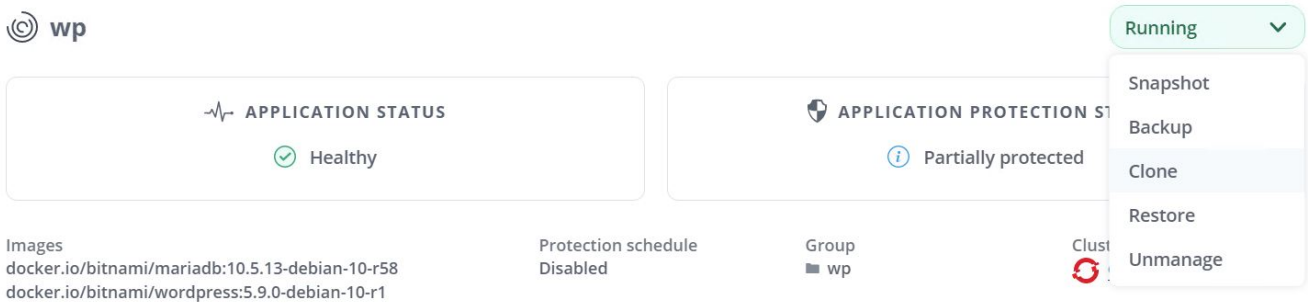
4. 当 Astra 控制中心在选定集群上还原应用程序时，新应用程序将进入还原状态。在 Astra 安装并检测到应用程序的所有资源后，该应用程序将进入可用状态。



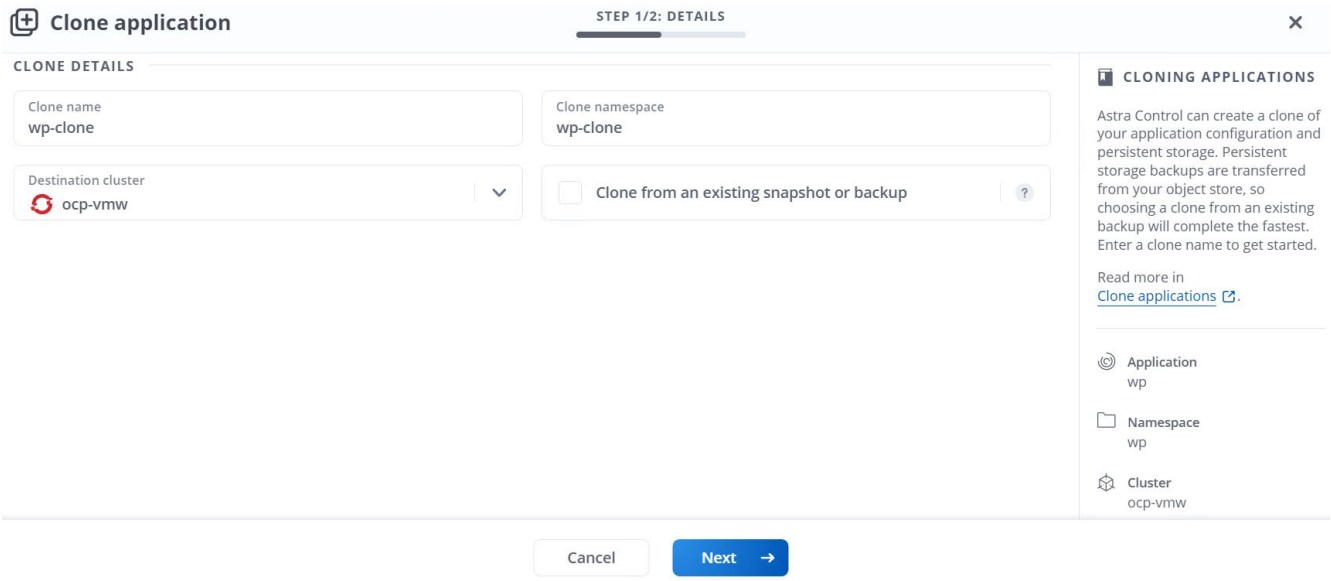
克隆应用程序

您可以将应用程序克隆到发起集群或远程集群，以进行开发 / 测试或应用程序保护和灾难恢复。在同一个存储后端的同一集群中克隆应用程序时，会使用 NetApp FlexClone 技术，从而可以即时克隆 PVC 并节省存储空间。

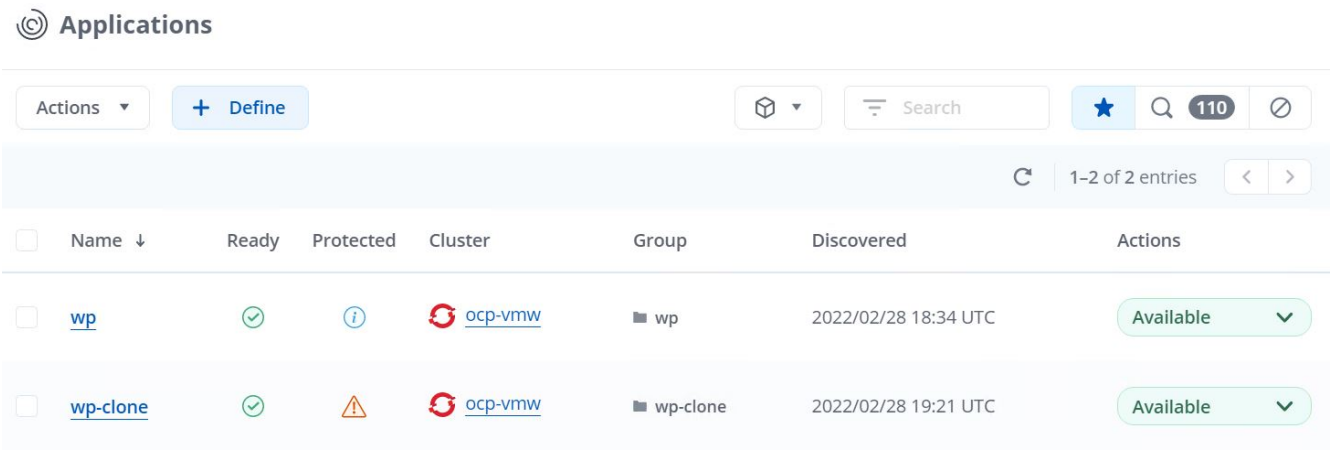
1. 要克隆应用程序，请导航到应用程序 > 受管选项卡，然后单击相关应用程序。单击应用程序名称旁边的下拉菜单，然后单击克隆。



2. 输入新命名空间的详细信息，选择要将其克隆到的集群，然后选择是要从现有快照，备份还是应用程序的当前状态克隆该命名空间。查看详细信息后，单击下一步并单击审阅窗格时克隆。



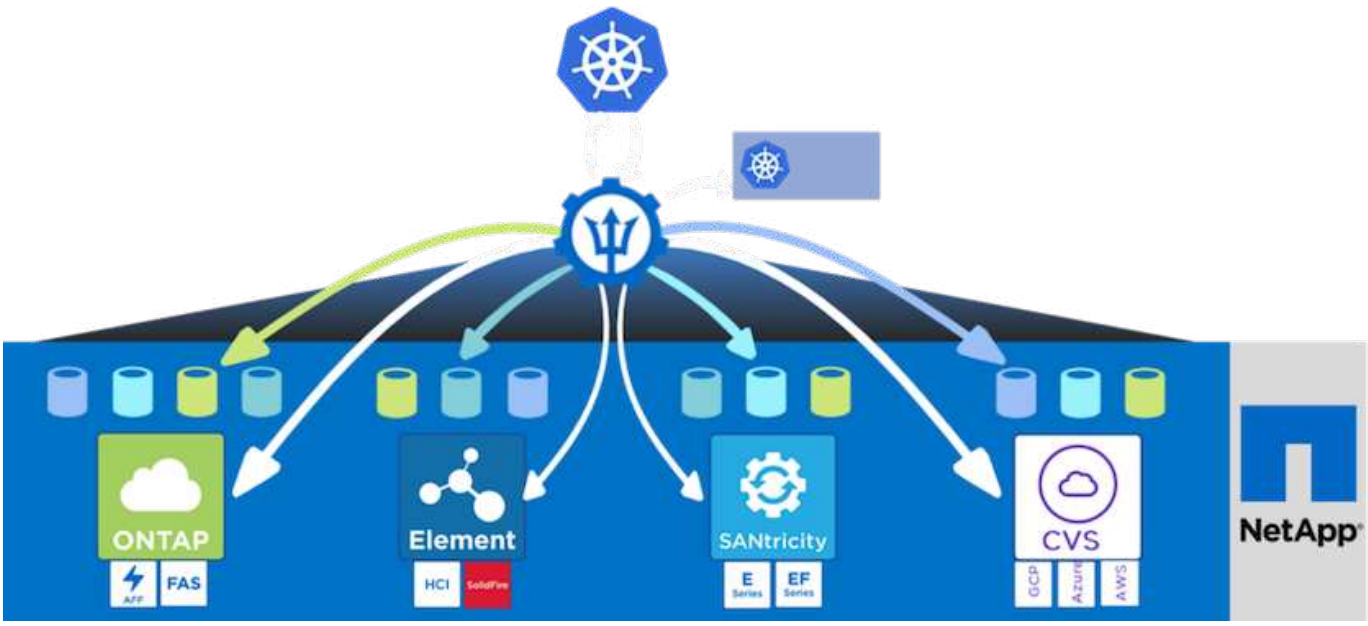
3. 当 Astra 控制中心在选定集群上创建应用程序时，新应用程序将进入 " 正在发现 " 状态。在 Astra 安装并检测到应用程序的所有资源后，该应用程序将进入可用状态。



Astra Trident 概述

Astra Trident 是一款开源且完全受支持的存储编排程序，适用于容器和 Kubernetes 分发版，包括 Red Hat OpenShift 。Trident 可与包括 NetApp ONTAP 和 Element 存储系统在内的整个 NetApp 存储产品组合配合使用，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，而无需存储管理员干预，从而加快了 DevOps 工作流的速度。

管理员可以根据项目需求和存储系统型号配置多个存储后端，以实现高级存储功能，包括数据压缩，特定磁盘类型或 QoS 级别，以保证一定水平的性能。定义后，开发人员可以在其项目中使用这些后端创建永久性卷声明（PVC），并按需将永久性存储附加到容器。



Astra Trident 具有快速的开发周期，就像 Kubernetes 一样，每年发布四次。

最新版 Astra Trident 于 2022 年 1 月发布。已测试的 Trident 版本的支持列表，可在该支持列表中找到 Kubernetes 分发版本 "此处"。

从 20.04 版开始， Trident 设置由 Trident 操作员执行。操作员可以简化大规模部署，并为在 Trident 安装过程中部署的 Pod 提供额外的支持，包括自我修复。

在 21.01 版中，我们提供了一个 Helm 图表，用于简化 Trident 操作员的安装。

下载 **Astra Trident**

要在已部署的用户集群上安装 Trident 并配置永久性卷，请完成以下步骤：

1. 将安装归档下载到管理工作站并提取内容。Trident 的当前版本为 22.01 ，可以下载 "此处"。

```
[netapp-user@rhel7 ~]$ wget
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
--2021-05-06 15:17:30--
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
[following]
--2021-05-06 15:17:30--  https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
Resolving github-releases.githubusercontent.com (github-
releases.githubusercontent.com)... 185.199.108.154, 185.199.109.154,
185.199.110.154, ...
Connecting to github-releases.githubusercontent.com (github-
releases.githubusercontent.com)|185.199.108.154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38349341 (37M) [application/octet-stream]
Saving to: ‘trident-installer-22.01.0.tar.gz’
```

```
100%[===================================================================
==============================================>] 38,349,341  88.5MB/s
in 0.4s

2021-05-06 15:17:30 (88.5 MB/s) - 'trident-installer-22.01.0.tar.gz'
saved [38349341/38349341]
```

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** 操作员

1. 首先将用户集群的 `kubeconfig` 文件的位置设置为环境变量，以便您不必引用该文件，因为 Trident 没有传递此文件的选项。

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

2. 在用户集群中创建 Trident 命名空间时，运行 Helm 命令从 Helm 目录中的 tarball 安装 Trident 操作员。

```
[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，客户环境可能需要自定义 Trident 部署。在这种情况下，还可以手动安装 Trident 操作员并更新所包含的清单以自定义部署。

手动安装 **Trident** 操作员

1. 首先，将用户集群的 `kubeconfig` 文件的位置设置为环境变量，以便您不必引用该文件，因为 Trident 没有传递此文件的选项。

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

2. `trident` **安装程序** 目录包含用于定义所有所需资源的清单。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.trident.netapp.io created
```

3. 如果不存在 Trident 命名空间，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. 为 Trident 操作员部署创建所需的资源，例如为操作员创建 `ServiceAccount` ，为 `SClusterRole` 和 `ClusterRoleBinding` ，为` erviceAccount ，专用 PodSecurityPolicy `或操作员本身创建。

```
[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. 您可以使用以下命令在操作员部署后检查其状态：

```
[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 `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 分发软件包和实用程序都会随附用于挂载默认安装的 NFS 后端的软件包和实用程序，包括 Red Hat OpenShift 。

但是，对于 NFSv3 ，客户端和服务器之间没有协商并发的机制。因此，客户端的最大 SUNRPC 插槽表条目数必须与服务器上支持的值手动同步，以确保 NFS 连接的最佳性能，而服务器不必减小连接的窗口大小。

对于 ONTAP ，支持的最大 SUNRPC 插槽表条目数为 128 ，即 ONTAP 一次可以处理 128 个并发 NFS 请求。但是，默认情况下，每个连接的 Red Hat CoreOS/Red Hat Enterprise Linux 最多包含 65 ， 536 个 SUNRPC

插槽表条目。我们需要将此值设置为 128 ，可以在 OpenShift 中使用计算机配置操作员（ Machine Config Operator ， MCO ）来完成此操作。

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

1. 登录到 OCP Web 控制台并导航到 Compute > Machine Configs 。单击 Create Machine Config 。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此配置并逐个重新启动。整个过程大约需要 20 到 30 分钟。使用 `oc get MCP` 验证是否应用了计算机配置，并确保已更新员工的计算机配置池。

```
[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

要使工作节点做好准备，以便能够通过 iSCSI 协议映射块存储卷，您必须安装支持此功能所需的软件包。

在 Red Hat OpenShift 中，可通过在部署集群后将 MCO （计算机配置操作员）应用于集群来实现此目的。

要配置工作节点以运行 iSCSI 服务，请完成以下步骤：

1. 登录到 OCP Web 控制台并导航到 Compute > Machine Configs 。单击 Create Machine Config 。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. 创建配置后，将此配置应用于工作节点并重新加载它们大约需要 20 到 30 分钟。使用 `oc get MCP` 验证是否应用了计算机配置，并确保已更新员工的计算机配置池。您还可以登录到工作节点，以确认 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
```

> 此外，还可以通过使用适当的标志运行 `oc debug` 命令来确认 MachineConfig 已成功应用且服务已按预期启动。

创建存储系统后端

完成 Astra Trident 操作员安装后，您必须为所使用的特定 NetApp 存储平台配置后端。请访问以下链接继续设置和配置 Astra Trident 。

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

NetApp ONTAP NFS 配置

要启用 Trident 与 NetApp ONTAP 存储系统的集成，您必须创建一个后端，以便与存储系统进行通信。

1. 下载的安装归档中提供了 `sample-input` folder 层次结构中的示例后端文件。对于提供 NFS 的 NetApp ONTAP 系统，将 `backend-ontap-nas.json` 文件复制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 backendName ， managementLIF ， dataLIF ， SVM ，用户名， 和密码值。

```
{
    "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. 安装此后端文件后，运行以下命令以创建第一个后端。

```
[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，您接下来必须创建一个存储类。与后端一样，可以在 sample-inputs 文件夹中为环境编辑一个示例存储类文件。将其复制到工作目录并进行必要的编辑，以反映所创建的后端。

```
[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. 必须对此文件进行的唯一编辑是，为新创建的后端存储驱动程序的名称定义 `backendType` 值。另请注意 name-field 值，稍后必须引用该值。

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

> 此文件中定义了一个名为 `FSType` 的可选字段。可以在 NFS 后端删除此行。

6. 运行 `oc` 命令以创建存储类。

```
[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，您必须创建第一个永久性卷请求（ PVC ）。此外，还可以在 sample-inputs 中使用一个示例 `pva-basic 。 yaml` file 来执行此操作。

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

8. 必须对此文件进行的唯一编辑是，确保 `storageClassName` 字段与刚刚创建的字段匹配。可以根据要配置的工作负载的需要进一步自定义 PVC 定义。

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

9. 发出 `oc` 命令创建 PVC 。根据所创建的后备卷的大小，创建可能需要一些时间，因此您可以在该过程完成后进行观察。

```
[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

要启用 Trident 与 NetApp ONTAP 存储系统的集成，您必须创建一个后端，以便与存储系统进行通信。

1. 下载的安装归档中提供了 `sample-input` folder 层次结构中的示例后端文件。对于提供 iSCSI 的 NetApp ONTAP 系统，将 `backend-ontap-san.json` 文件复制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. 安装此后端文件后，运行以下命令以创建第一个后端。

```
[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，您接下来必须创建一个存储类。与后端一样，可以在 sample-inputs 文件夹中为环境编辑一个示例存储类文件。将其复制到工作目录并进行必要的编辑，以反映所创建的后端。

```
[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. 必须对此文件进行的唯一编辑是，为新创建的后端存储驱动程序的名称定义 `backendType` 值。另请注意 name-field 值，稍后必须引用该值。

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

> 此文件中定义了一个名为 `FSType` 的可选字段。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），也可以删除此值，以便 OpenShift 决定要使用的文件系统。

6. 运行 `oc` 命令以创建存储类。

```
[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，您必须创建第一个永久性卷请求（ PVC ）。此外，还可以在 sample-inputs 中使用一个示例 `pva-basic 。 yaml` file 来执行此操作。

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

8. 必须对此文件进行的唯一编辑是，确保 `storageClassName` 字段与刚刚创建的字段匹配。可以根据要配置的工作负载的需要进一步自定义 PVC 定义。

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

9. 发出 `oc` 命令创建 PVC 。根据所创建的后备卷的大小，创建可能需要一些时间，因此您可以在该过程完成后进行观察。

```
[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

要启用 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. 下载的安装归档中提供了 `sample-input` folder 层次结构中的示例后端文件。对于提供 iSCSI 服务的 NetApp Element 系统，将 `backend-solidfire.json` 文件复制到您的工作目录中，然后编辑该文件。

```
[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 `endpoint` 行上的用户，密码和 MVIP 值。
   b. 编辑 `SVIP` 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. 安装好此后端文件后，运行以下命令创建第一个后端。

```
[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，您接下来必须创建一个存储类。与后端一样，可以在 sample-inputs 文件夹中为环境编辑一个示例存储类文件。将其复制到工作目录并进行必要的编辑，以反映所创建的后端。

```
[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. 必须对此文件进行的唯一编辑是，为新创建的后端存储驱动程序的名称定义 `backendType` 值。另请注意 name-field 值，稍后必须引用该值。

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

> 此文件中定义了一个名为 `FSType` 的可选字段。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），也可以将其删除以允许 OpenShift 决定要使用的文件系统。

5. 运行 `oc` 命令以创建存储类。

```
[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，您必须创建第一个永久性卷请求（ PVC ）。此外，还可以在 sample-inputs 中使用一个示

例 `pva-basic 。 yaml` file 来执行此操作。

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

7. 必须对此文件进行的唯一编辑是，确保 `storageClassName` 字段与刚刚创建的字段匹配。可以根据要配置的工作负载的需要进一步自定义 PVC 定义。

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

8. 发出 `oc` 命令创建 PVC 。根据所创建的后备卷的大小，创建可能需要一些时间，因此您可以在该过程完成后进行观察。

```
[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

了解负载平衡器选项： **Red Hat OpenShift** 与 **NetApp**

在大多数情况下， Red Hat OpenShift 会通过路由向外部世界提供应用程序。通过为服务提供一个可从外部访问的主机名来公开该服务。OpenShift 路由器可以使用定义的路由及其服务标识的端点，以便为外部客户端提供此命名连接。

但是，在某些情况下，应用程序需要部署和配置自定义负载平衡器才能公开相应的服务。其中一个示例是 NetApp Astra 控制中心。为了满足这一需求，我们评估了许多自定义负载平衡器选项。本节将介绍其安装和配置。

以下页面提供了有关负载平衡器选项的追加信息，这些选项已在 Red Hat OpenShift with NetApp 解决方案中进行验证：

- "元 LB"
- "F5 BIG-IP"

安装 **MetalLB** 负载平衡器： **Red Hat OpenShift** 与 **NetApp**

此页面列出了 MetalLB 负载平衡器的安装和配置说明。

MetalLB 是一种安装在 OpenShift 集群上的自托管网络负载平衡器，可用于在未在云提供程序上运行的集群中创建类型为负载平衡器的 OpenShift 服务。MetalLB 可协同工作以支持负载平衡器服务的两个主要功能是地址分配和外部公告。

MetalLB 配置选项

根据 MetalLB 如何公布分配给 OpenShift 集群以外的负载平衡器服务的 IP 地址，它可在两种模式下运行：

- * 第 2 层模式。 * 在此模式下， OpenShift 集群中的一个节点将接管此服务的所有权，并对该 IP 的 ARP 请求做出响应，使其可在 OpenShift 集群之外访问。由于只有节点才公布 IP ，因此存在带宽瓶颈和较慢的故障转移限制。有关详细信息，请参见文档 "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，并公布路由以将流量转发到服务 IP 。前提条件是将 MetalLB 与该网络中的路由器集成在一起。由于 BGP 中采用哈希机制，因此在服务的 IP 到节点映射发生更改时，它具有一定的限制。有关详细信息，请参见文档 "此处"。

> 在本文档中，我们将在第 2 层模式下配置 MetalLB 。

安装 **MetalLB** 负载平衡器

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` 并从控制器部署和主讲人 DemonSet 中删除 `spec.template.spec.securityContext` 。
   - 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 DemonSet 提升权限，使其能够执行使负载平衡器正常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. 通过在 `metallb-system` 命名空间中创建 `ConfigMap` 来配置 MetalLB 。

```
[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，在创建负载平衡器服务时， MetalLB 会为这些服务分配一个外部 IP ，并通过响应 ARP 请求来公布 IP 地址。

> 如果要在 BGP 模式下配置 MetalLB ，请跳过上述步骤 6 并按照 MetalLB 文档中的操作步骤进行操作 "此处"。

安装 **F5 BIG-IP** 负载平衡器

F5 BIG-IP 是一款应用程序交付控制器（Application Delivery Controller， AD），可提供一系列高级生产级流量管理和安全服务，例如 L4-L7 负载平衡， SSL/TLS 卸载， DNS ，防火墙等。这些服务可显著提高应用程序的可用性，安全性和性能。

F5 BIG-IP 可以在专用硬件上，云中或内部虚拟设备上以各种方式进行部署和使用。请参见此处的文档，了解如何根据需要部署 F5 BIG-IP 。

为了将 F5 BIG-IP 服务与 Red Hat OpenShift 高效集成， F5 提供了 BIG-IP 容器传入服务（BIG-IP Container Ingress Service， CIS）。CI 作为控制器 POD 进行安装，用于监控 OpenShift API 以获取某些自定义资源定义（Custom Resource Definitions， CRD），并管理 F5 BIG-IP 系统配置。可以配置 F5 BIG-IP CIS ，以控制 OpenShift 中的服务类型 LoadBalbalers" 和 " 路由 " 。

此外，要自动分配 IP 地址以服务类型负载平衡器，您可以使用 F5 IPAM 控制器。F5 IPAM 控制器作为控制器 POD 进行安装，该控制器 POD 会通过 ipamLabel 标注监视 OpenShift API 以获取负载平衡器服务，以便从预配置的池分配 IP 地址。

此页面列出了 F5 BIG-IP CIS 和 IPAM 控制器的安装和配置说明。作为前提条件，您必须已部署并获得 F5 BIG-IP 系统的许可。此外，它还必须获得 SDN 服务的许可，这些服务默认包含在 BIG-IP VE 基础许可证中。

> F5 BIG-IP 可以在独立模式或集群模式下部署。出于此验证的目的， F5 BIG-IP 部署在独立模式下，但出于生产目的，最好使用由大型 IP 组成的集群来避免单点故障。

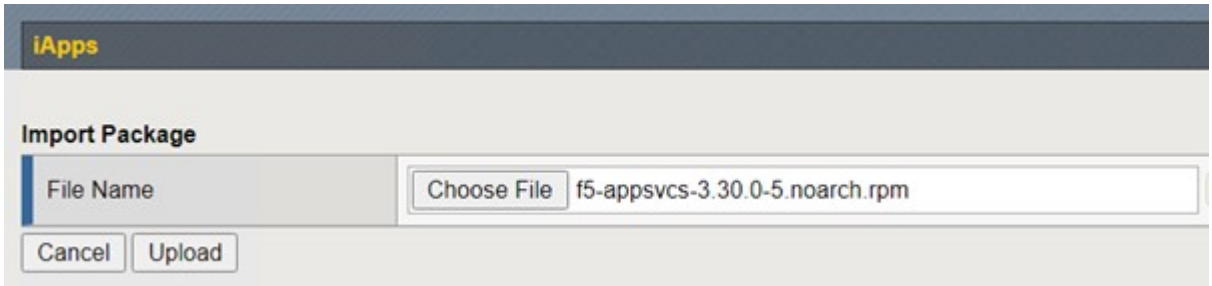
F5 BIG-IP 系统可以部署在专用硬件上，云中或内部部署的虚拟设备中，其版本高于 12.x ，以便与 F5 CIS 集成。在本文档中，我们已将 F5 BIG-IP 系统验证为虚拟设备，例如使用 BIG-IP VE 版本。

经过验证的版本

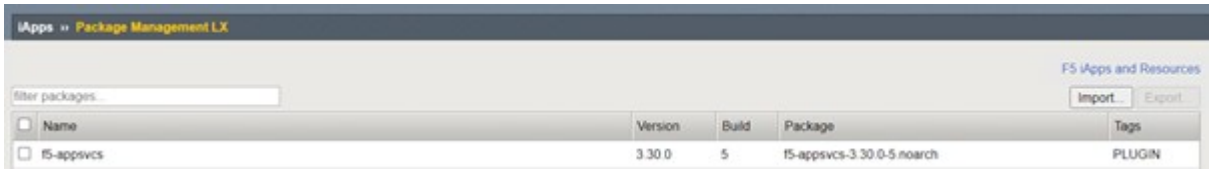
技术	软件版本
Red Hat OpenShift	4.6 EUS ， 4.7
F5 BIG-IP VE 版本	16.1.0
F5 容器传入服务	2.5.1
F5 IPAM 控制器	0.1.4
F5 AS3	3.30.0

安装

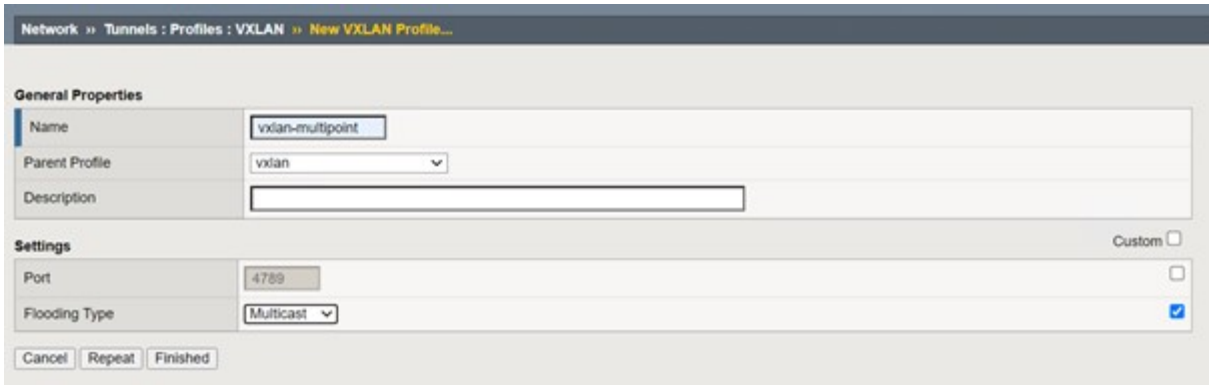
1. 安装 F5 Application Services 3 扩展，以允许 BIG-IP 系统接受 JSON 中的配置，而不是强制命令。转至 "F5 AS3 GitHub 存储库"，并下载最新的 RPM 文件。
2. 登录到 F5 BIG-IP 系统，导航到 "iApps" > "Package Management LX" ，然后单击 "Import" 。
3. 单击选择文件并选择已下载的 AS3 RPM 文件，单击确定，然后单击上传。



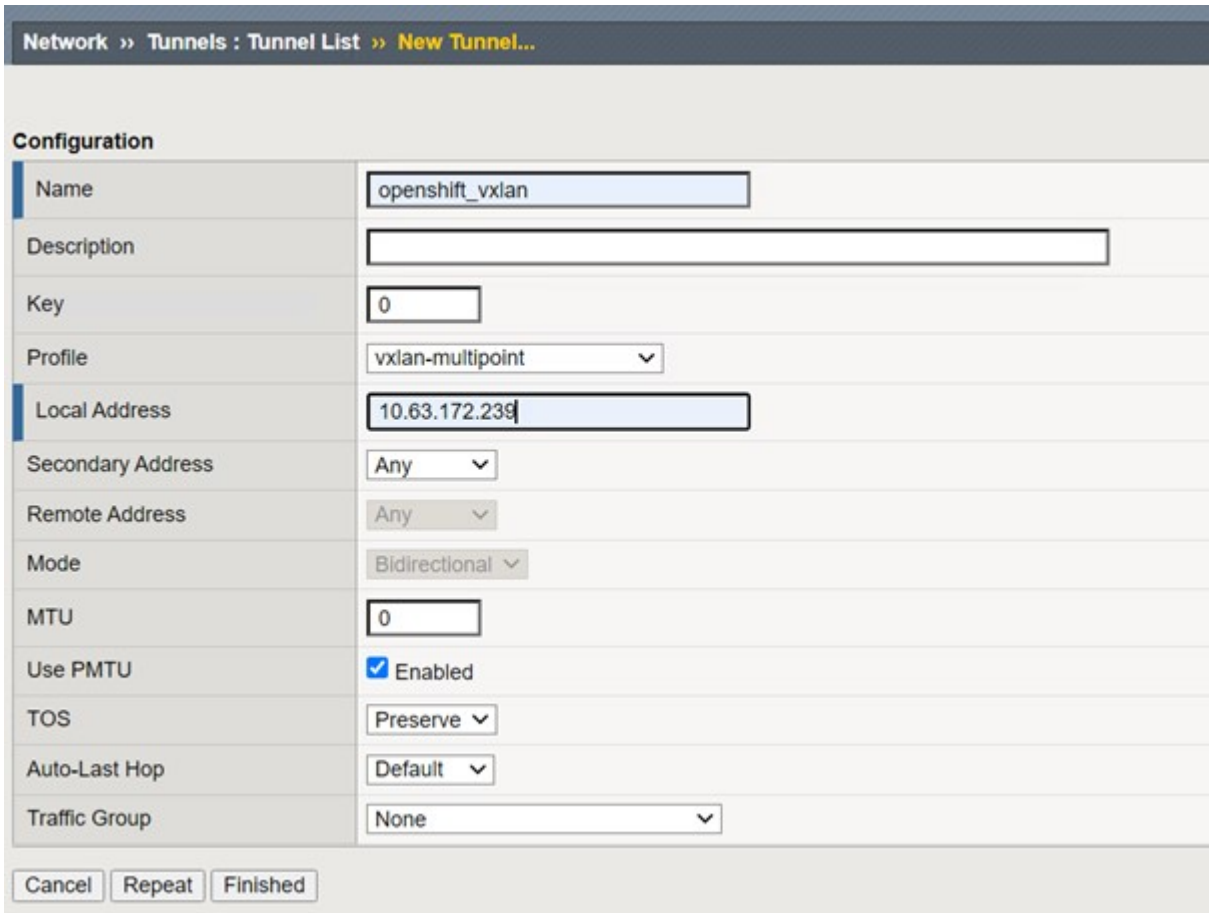
4. 确认 AS3 扩展已成功安装。



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 OpenShift SDN 的 BIG-IP 系统上创建 VXLAN 通道接口，在 OpenShift 和 BIG-IP 服务器之间创建通道。导航到 " 网络 ">" 通道 ">" 配置文件 " ，单击 " 创建 " ，然后将父配置文件设置为 VXLAN ，并将 " 洪水类型 " 设置为 " 多播 " 。输入配置文件的名称，然后单击完成。



6. 导航到 " 网络 ">" 通道 ">" 通道列表 " ，单击 " 创建 " ，然后输入通道的名称和本地 IP 地址。选择在上一步中创建的通道配置文件，然后单击完成。



7. 使用 cluster-admin 权限登录到 Red Hat OpenShift 集群。
8. 在 OpenShift 上为 F5 BIG-IP 服务器创建一个子网，从而将子网从 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 通道） 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
```

根据您的环境情况更改主机提示和其他详细信息。

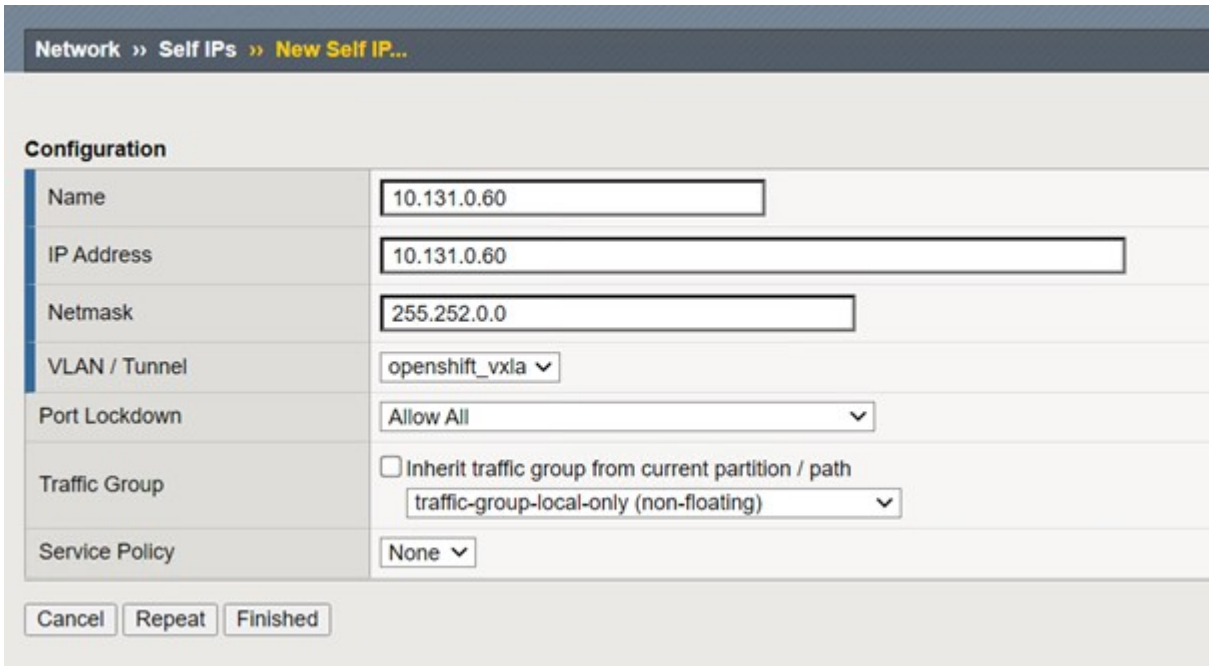
10. 创建 HostSubnet 资源。

```
[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。

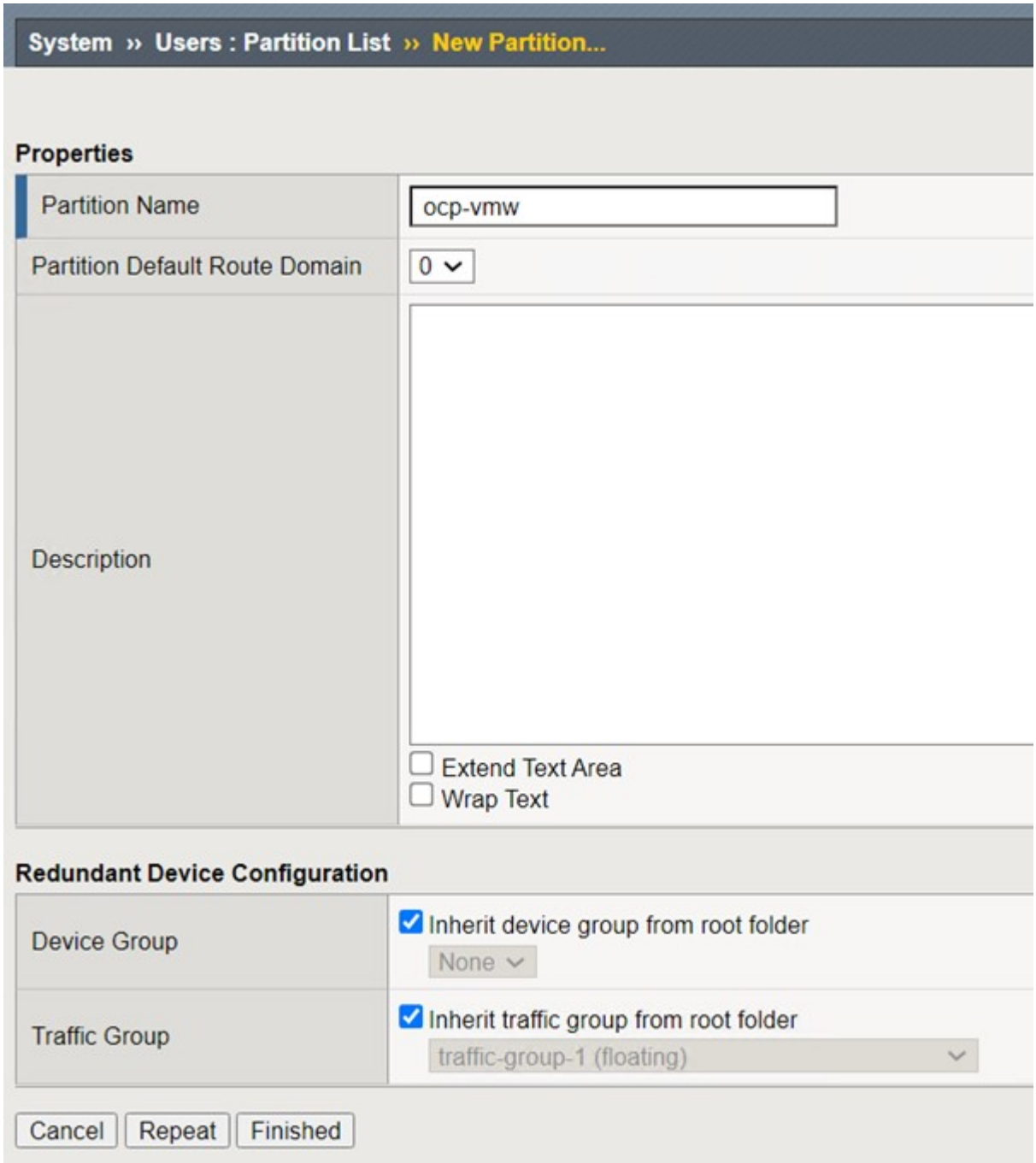
```
[admin@rhel-7 ~]$ oc get hostsubnet

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 上使用与 F5 BIG-IP 服务器对应的 OpenShift 主机子网范围中的 IP 创建自 IP 。登录到 F5 BIG-IP 系统，导航到 " 网络 ">" 自 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 通道，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，以便在 CIS 中配置和使用。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 建议不要对由 CIS 管理的分区进行手动配置。

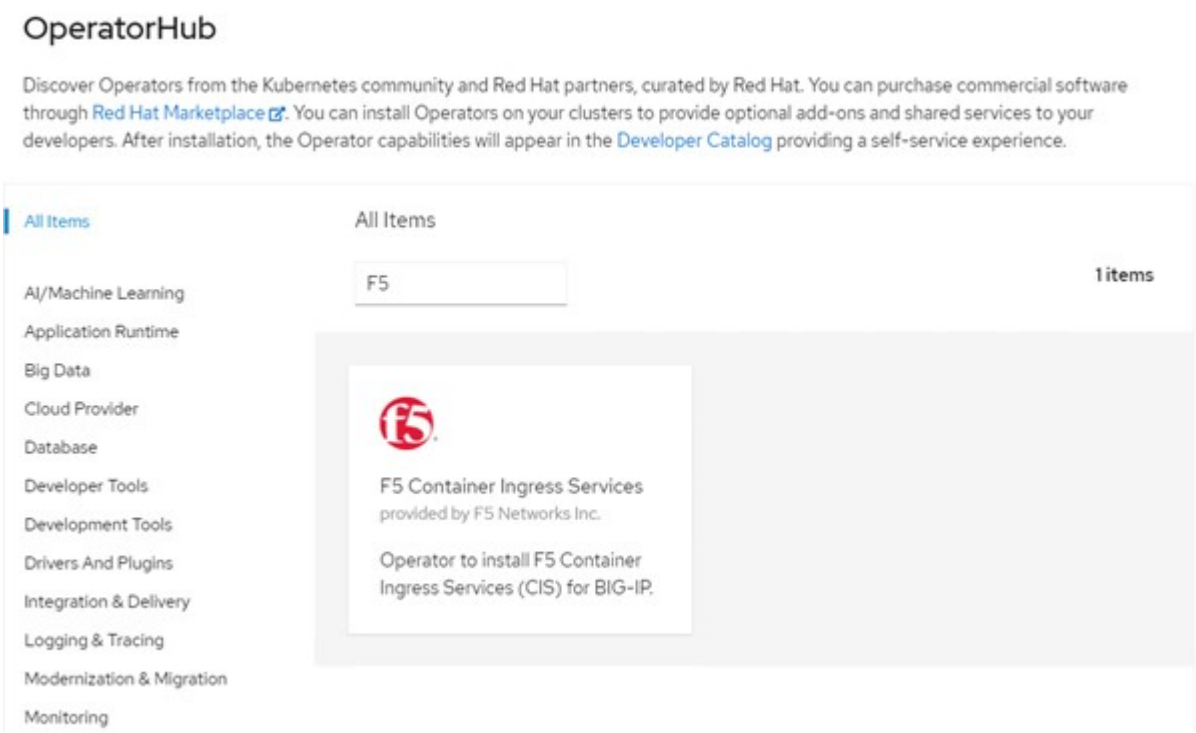
14. 使用 OperatorHub 中的运算符安装 F5 BIG-IP CIS 。使用集群管理员权限登录到 Red Hat OpenShift 集群，并使用 F5 BIG-IP 系统登录凭据创建一个密钥，这是操作员的前提条件。

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

secret/bigip-login created
```

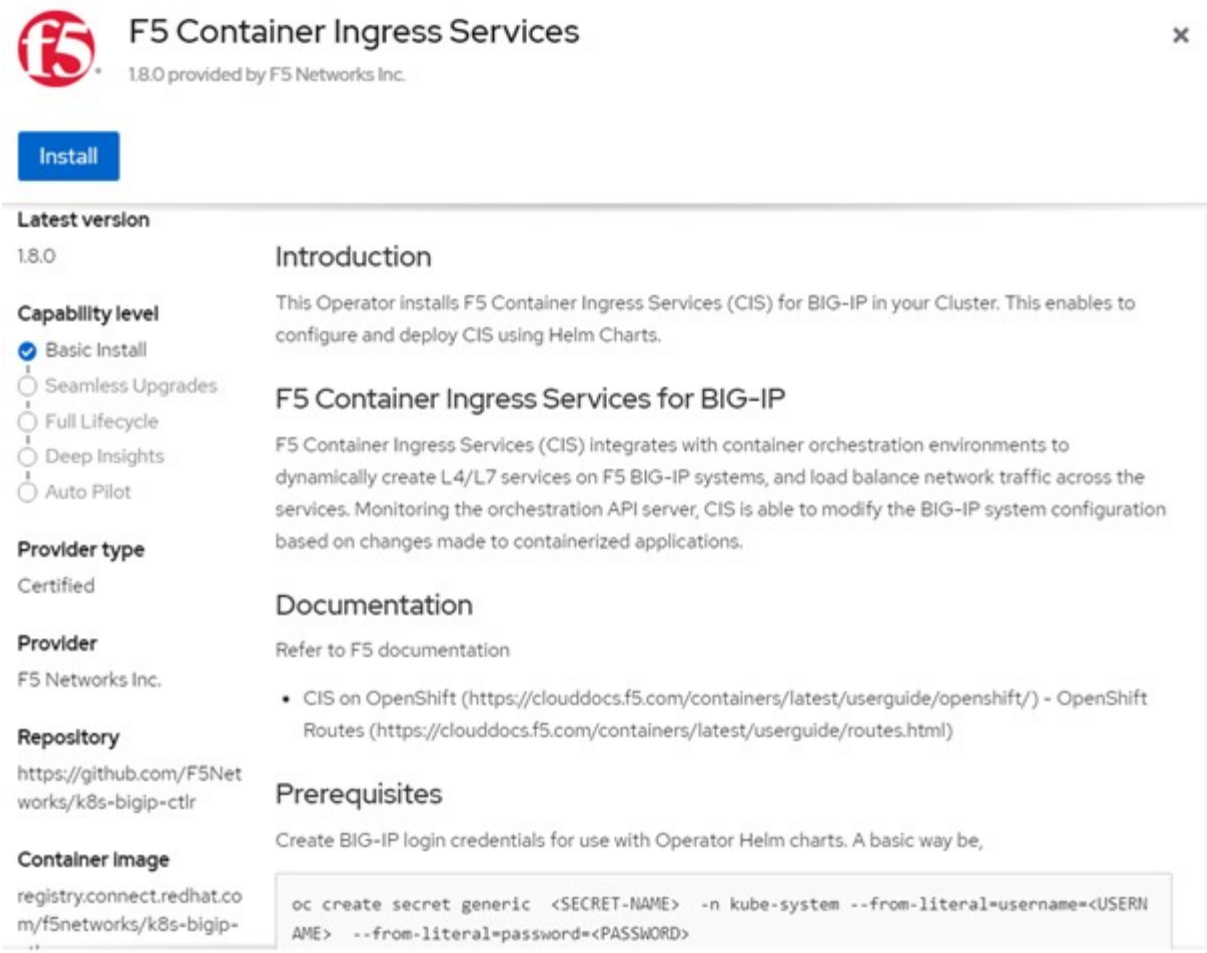
15. 安装 F5 CIS CRD 。

```
[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
```

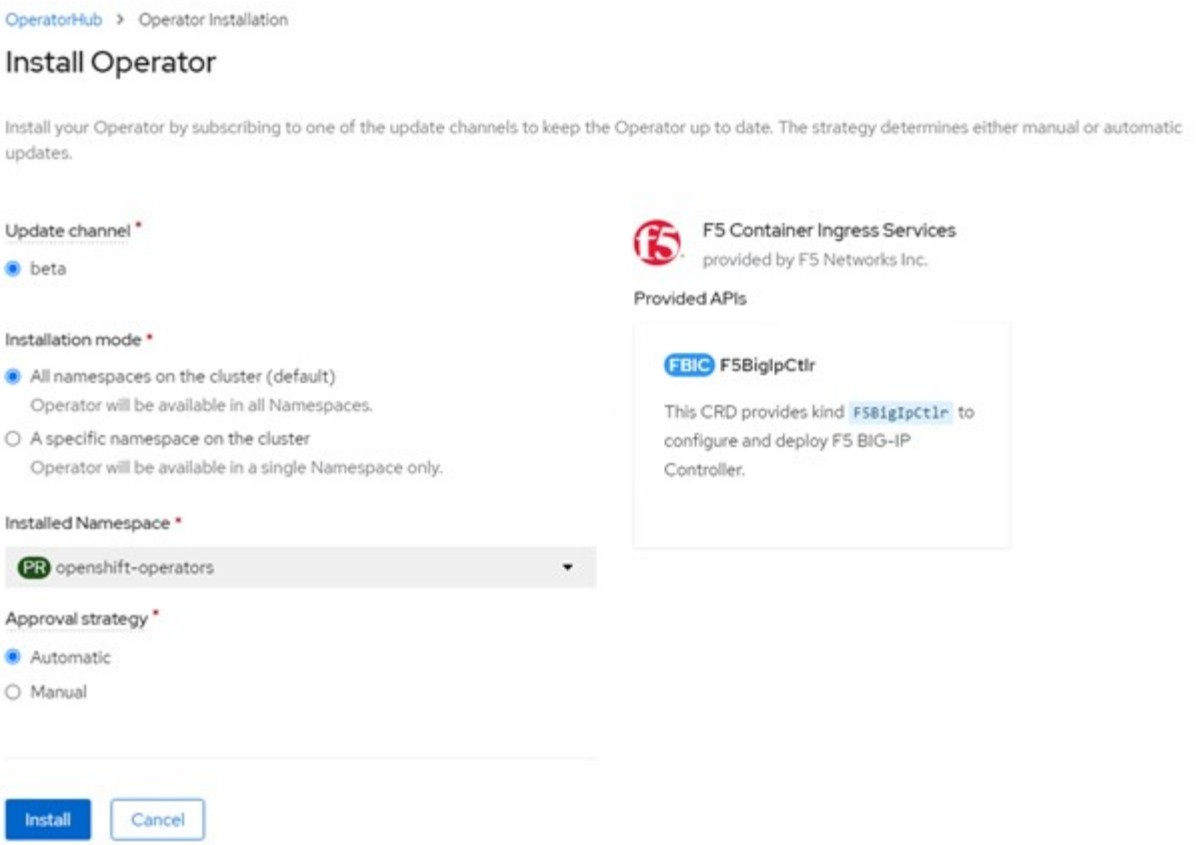
16. 导航到 Operators > OperatorHub ，搜索关键字 F5 ，然后单击 F5 Container In出口 服务磁贴。



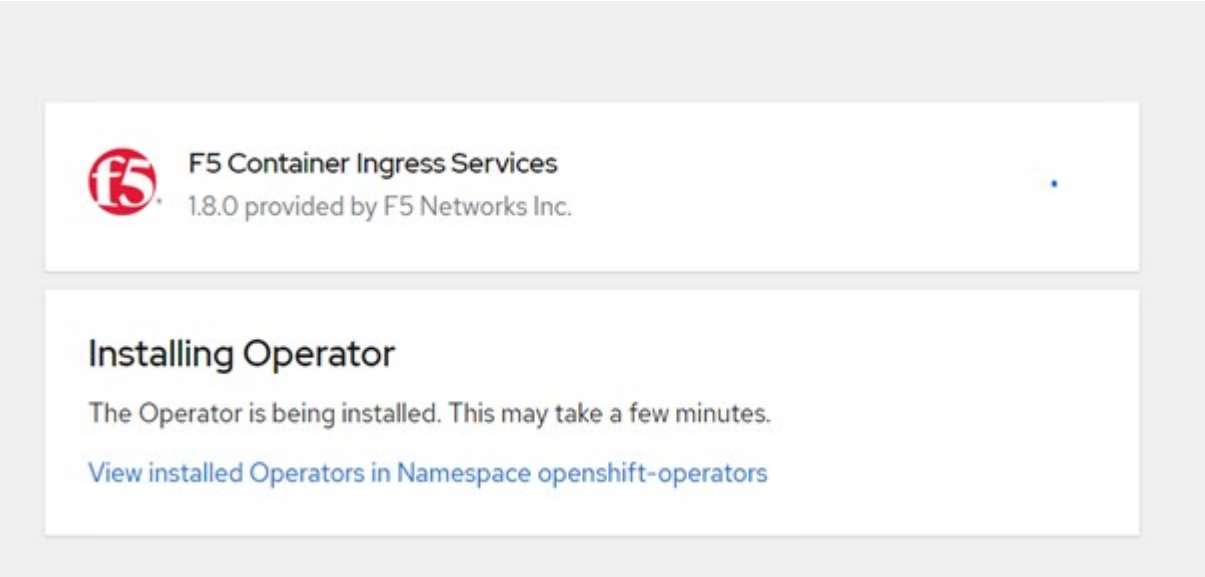
17. 阅读操作员信息，然后单击安装。



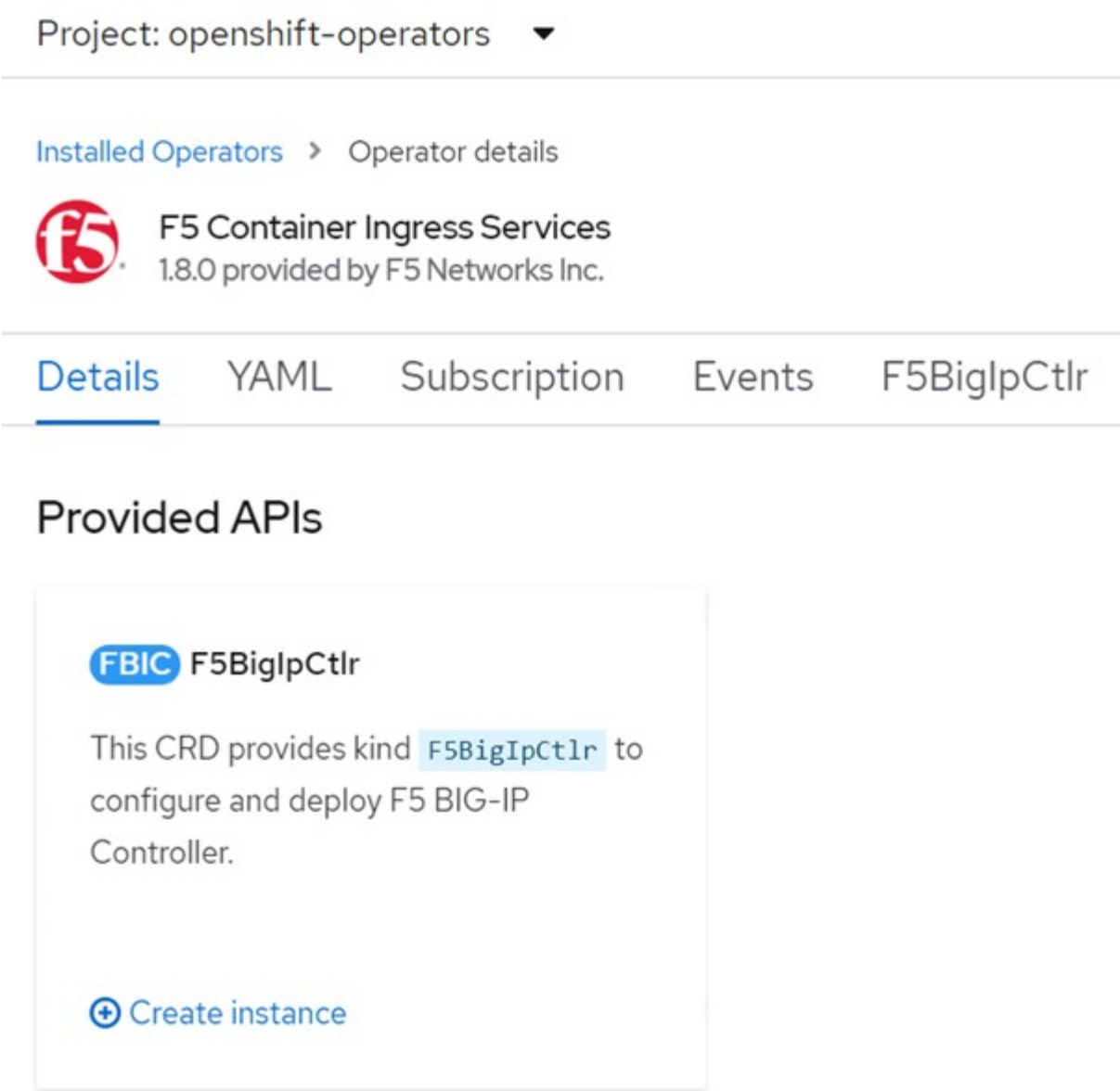
18. 在 Install Operator 屏幕上，保留所有默认参数，然后单击 Install 。



19. 安装操作员需要一段时间。



20. 安装操作员后，将显示安装成功消息。
21. 导航到 Operators > Installed Operators ，单击 F5 Container In出口 服务，然后单击 F5BigIpCtrl+Alt+Del 图块下的 Create Instance 。

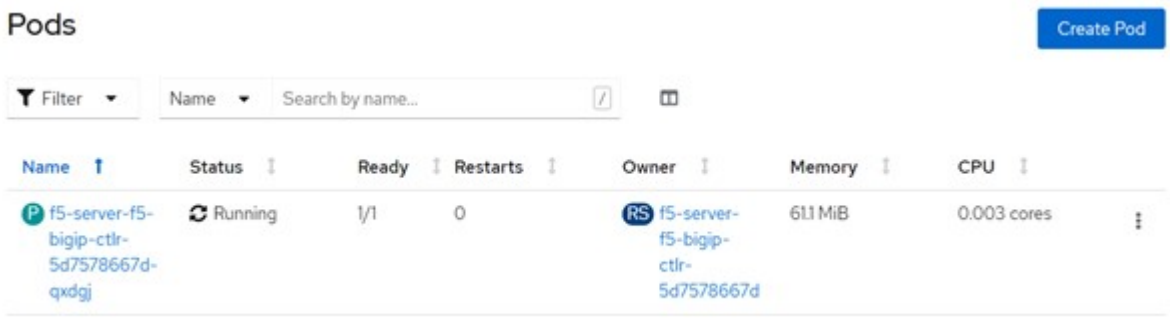


22. 单击 YAML View ，并在更新必要参数后粘贴以下内容。

> 在复制内容之前，更新以下参数 `bigip_partition` ，` OpenShift_SDN_name` ，`bigip_url` 和 `bigip_login_secret` ，以反映您的设置值。

```
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。此操作将在 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 路由器负责公布和处理这些路由的流量。但是，您也可以将 F5 CIS 配置为支持通过外部 F5 BIG-IP 系统的路由，该系统可以作为辅助路由器运行，也可以替代自托管 OpenShift 路由器运行。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， BIG-IP 负责处理公告和流量路由。有关启用此功能的参数的信息，请参见此处的文档。请注意，这些参数是在 APPS/v1 API 中为 OpenShift 部署资源定义的。因此，在将这些参数与 F5BigIpartl 资源 cis.f5.com/v1 API 结合使用时，请将参数名称的连字符（ - ）替换为下划线（ _ ）。

24. 传递给创建 CIS 资源的参数包括 `ipam ： true` 和 `custom_resource_mode ： true` 。要启用与 IPAM 控制器的 CIS 集成，需要使用这些参数。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 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。

更新以下 spec.template.spec.containers[0].args 中的 ip-range 参数，以反映与您的设置对应的 ipamLabel 和 IP 地址范围。

要使 IPAM 控制器能够从定义的范围检测和分配 IP 地址，需要为类型为 loadbalancer 的服务标注 ipamLabels （`range1` 和` range2 `in below example] ）。

```
[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. 创建类型为 loadbalancer 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. 创建部署并使用已创建的负载平衡器服务。

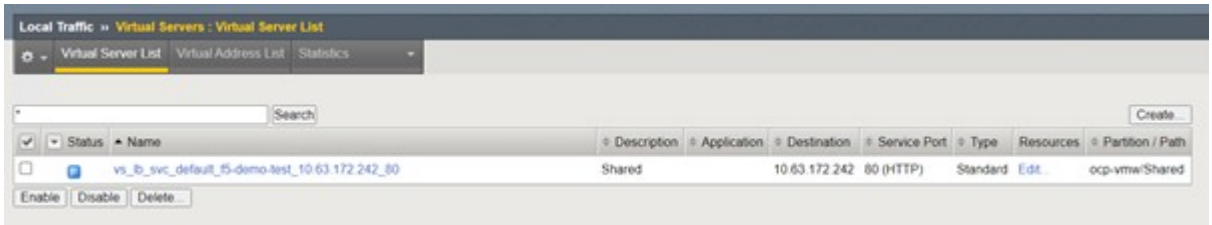
```
[admin@rhel-7 ~]$ vi example_deployment.yaml

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. 检查是否在 OpenShift 中为 loadbalancing 类型的服务在 BIG-IP 系统中创建了相应的虚拟服务器。导航到 " 本地流量 ">" 虚拟服务器 ">" 虚拟服务器列表 " 。



创建私有映像注册表

对于大多数 Red Hat OpenShift 部署，请使用等公有注册表 "Quay.io" 或 "DockerHub" 满足大多数客户的需求。但是，有时客户可能希望托管自己的私有或自定义映像。

本操作步骤介绍了如何创建私有映像注册表，该注册表由 Astra Trident 和 NetApp ONTAP 提供的永久性卷提供支持。

> Astra 控制中心需要注册表来托管 Astra 容器所需的映像。以下部分介绍了在 Red Hat OpenShift 集群上设置专用注册表以及推送支持安装 Astra 控制中心所需的映像的步骤。

创建私有映像注册表

1. 从当前默认存储类中删除默认标注，并将支持 Trident 的存储类标注为 OpenShift 集群的默认值。

```
[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. 在 `sPec` 部分中输入以下存储参数，以编辑 imageeregistry 运算符。

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

storage:
  pvc:
    claim:
```

3. 在 `sPec` 部分中输入以下参数，以便使用自定义主机名创建 OpenShift 路由。保存并退出。

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

> 如果要为路由设置自定义主机名，则会使用上述路由配置。如果您希望 OpenShift 使用默认主机名创建路由，可以将以下参数添加到 `sPec` 部分： `defaultRoute` ： `true` 。

自定义 TLS 证书

默认情况下，当您为路由使用自定义主机名时，它会使用 OpenShift 入口操作员的默认 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 证书和密钥创建密钥。

```
[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 imageeregistry 运算符，并将以下参数添加到 `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 imageeregistry 运算符，并将该运算符的管理状态更改为 `Maned` state 。保存并退出。

```
oc edit configs.imageregistry/cluster

managementState: Managed
```

5. 如果满足所有前提条件，则会为专用映像注册表创建 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. 如果您对传入操作员 OpenShift 注册表路由使用默认 TLS 证书，则可以使用以下命令提取 TLS 证书。

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

7. 要允许 OpenShift 节点访问并从注册表中提取映像，请将证书添加到 OpenShift 节点上的 Docker 客户端。使用 TLS 证书在 `OpenShift-config` 命名空间中创建一个配置映射，并将其修补到集群映像配置中以使此证书可信。

```
[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. 要允许用户或用户组从注册表中提取映像，必须为用户分配注册表查看器角色。

```
[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映像，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. 要使 OpenShift 节点能够访问注册表并推送或拉取映像，您需要配置拉取密钥。

```
[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，可以将此提取密钥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映像，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 命令登录到 OpenShift 。

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

c. 使用 podman/Docker 命令使用 OpenShift 用户凭据登录到注册表。

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` user 登录到专用注册表，请使用 token 代替密码。

Docker

```
[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` user 登录到专用注册表，请使用 token 代替密码。

d. 推送或拉图像。

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
```

Docker

```
[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
```

解决方案验证和使用情形：采用 NetApp 的 Red Hat OpenShift

此页面上提供的示例包括解决方案验证以及采用 NetApp 的 Red Hat OpenShift 的用例。

- "部署具有永久性存储的 Jenkins CI/CD 管道"
- "在使用 NetApp 的 Red Hat OpenShift 上配置多租户"
- "借助 NetApp ONTAP 实现 Red Hat OpenShift 虚拟化"
- "借助 NetApp 在 Red Hat OpenShift 上为 Kubernetes 提供高级集群管理"

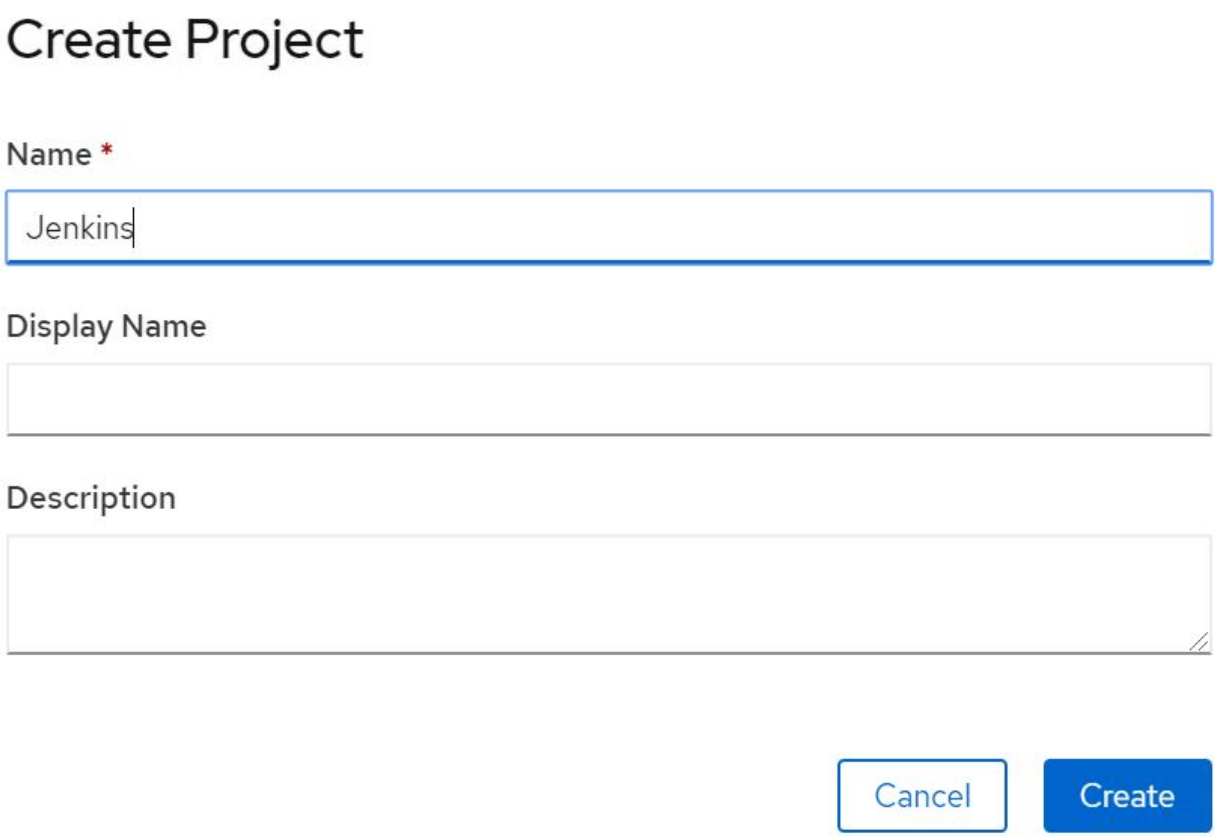
部署采用永久性存储的 Jenkins CI/CD 管道：采用 NetApp 的 Red Hat OpenShift

本节介绍了与 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. 在此示例中，我们使用永久性存储部署了 Jenkins 。要支持 Jenkins 构建，请创建 PVC 。导航到 "Storage">"Persistent Volume Claim " ，然后单击 "Create Persistent Volume Claim " 。选择已创建的存储类，确保永久性卷声明名称是 Jenkins ，选择适当的大小和访问模式，然后单击创建。

Project: jenkins

Create Persistent Volume Claim

Edit YAML

Storage Class

SC basic

Storage class for the new claim.

Persistent Volume Claim Name *

jenkins

A unique name for the storage claim within the project.

Access Mode *

Single User (RWO) Shared Access (RWX) Read Only (ROX)

Permissions to the mounted drive.

Size *

100 GiB

Desired storage capacity.

Use label selectors to request storage

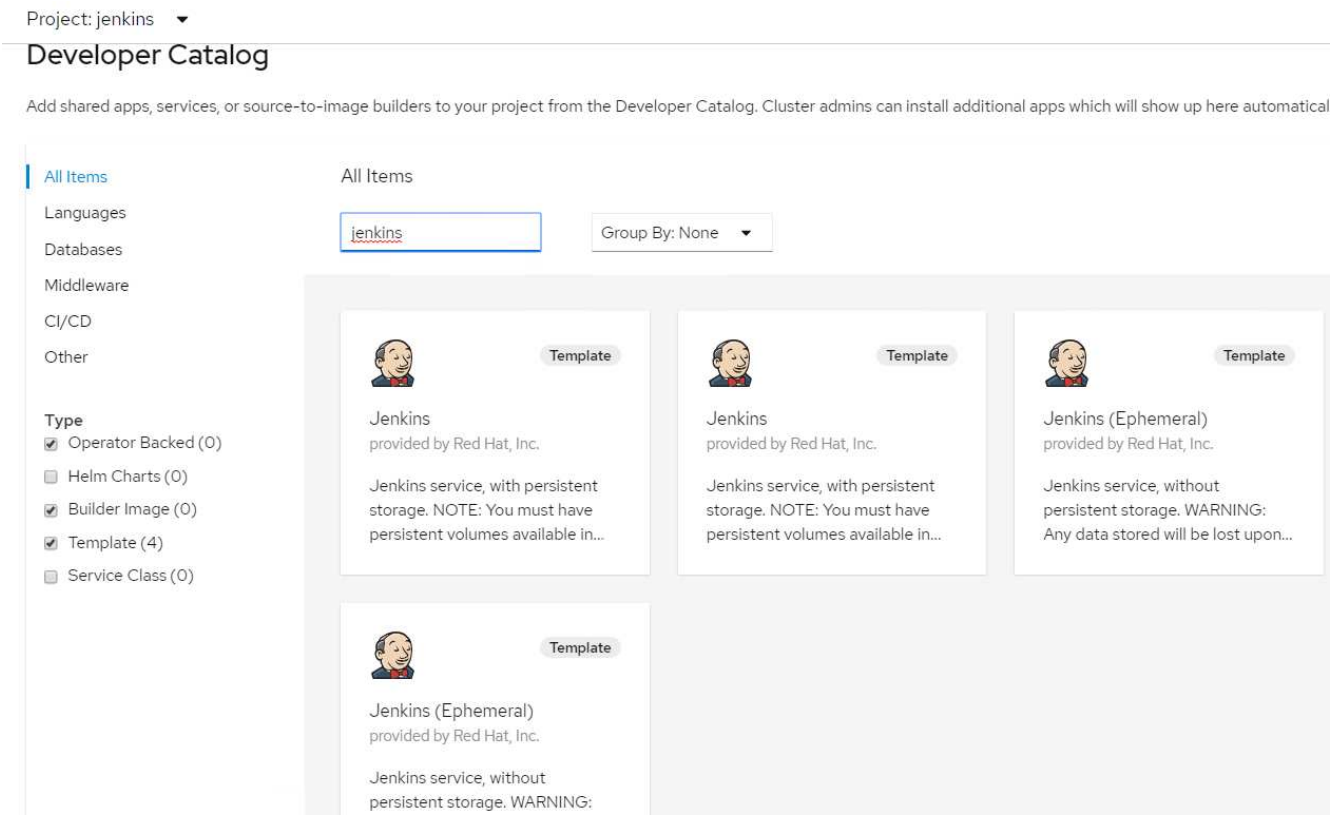
Use label selectors to define how storage is created.

Create Cancel

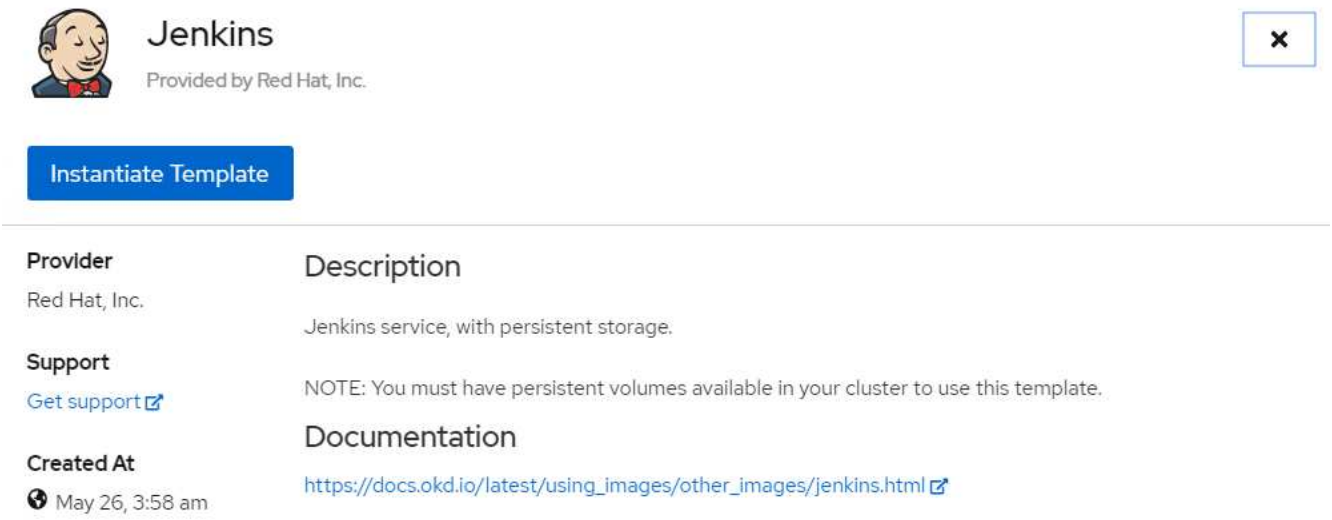
使用永久性存储部署 **Jenkins**

要使用永久性存储部署 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 +Add ，然后从目录中选择。在 Filter by Keyword 栏中，搜索 Jenkins 。选择 Jenkins Service with Persistent Storage 。



2. 单击 实例化模板 。



3. 默认情况下，系统会填充 Jenkins 应用程序的详细信息。根据您的要求，修改参数并单击创建。此过程将创建支持 OpenShift 上的 Jenkins 所需的所有资源。

Instantiate Template

Namespace *

PR jenkins

Jenkins Service Name

jenkins

The name of the OpenShift Service exposed for the Jenkins container.

Jenkins JNLP Service Name

jenkins-jnlp

The name of the service used for master/slave communication.

Enable OAuth in Jenkins

true

Whether to enable OAuth OpenShift integration. If false, the static account 'admin' will be initialized with the password 'password'.

Memory Limit

1Gi

Maximum amount of memory the container can use.

Volume Capacity *

50Gi

Volume space available for data, e.g. 512Mi, 2Gi.

Jenkins ImageStream Namespace

openshift

The OpenShift Namespace where the Jenkins ImageStream resides.

Disable memory intensive administrative monitors

false

Whether to perform memory intensive, possibly slow, synchronization with the Jenkins Update Center on start. If true, the Jenkins core update monitor and site warnings monitor are disabled.

Jenkins ImageStreamTag

jenkins:2

Name of the ImageStreamTag to be used for the Jenkins image.

Fatal Error Log File

false

When a fatal error occurs, an error log is created with information and the state obtained at the time of the fatal error.

Allows use of Jenkins Update Center repository with invalid SSL certificate

false

Whether to allow use of a Jenkins Update Center that uses invalid certificate (self-signed, unknown CA). If any value other than 'false', certificate check is bypassed. By default, certificate check is enforced.

Create Cancel

Jenkins

INSTANT-APP JENKINS

View documentation Get support

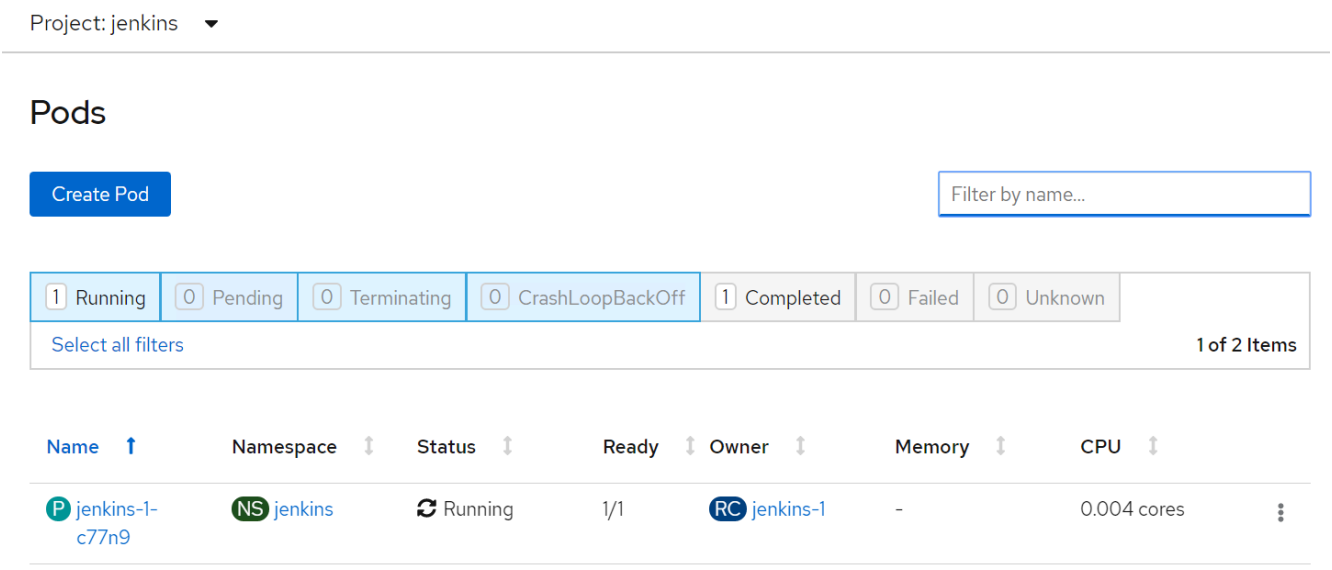
Jenkins service, with persistent storage.

NOTE: You must have persistent volumes available in your cluster to use this template.

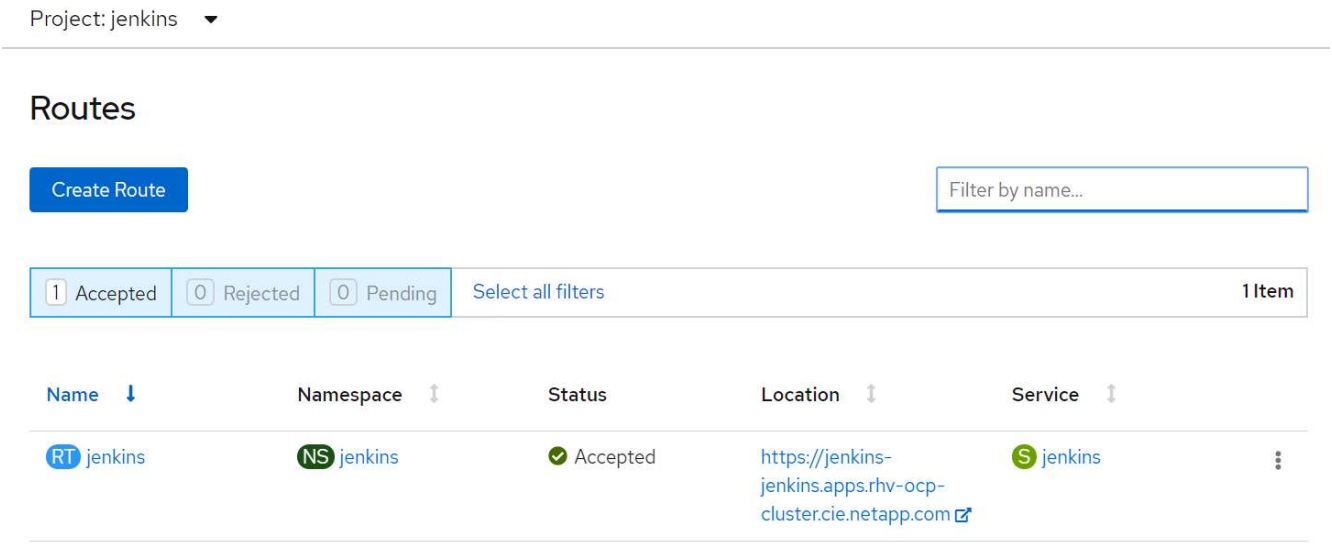
The following resources will be created:

- DeploymentConfig
- PersistentVolumeClaim
- RoleBinding
- Route
- Service
- ServiceAccount

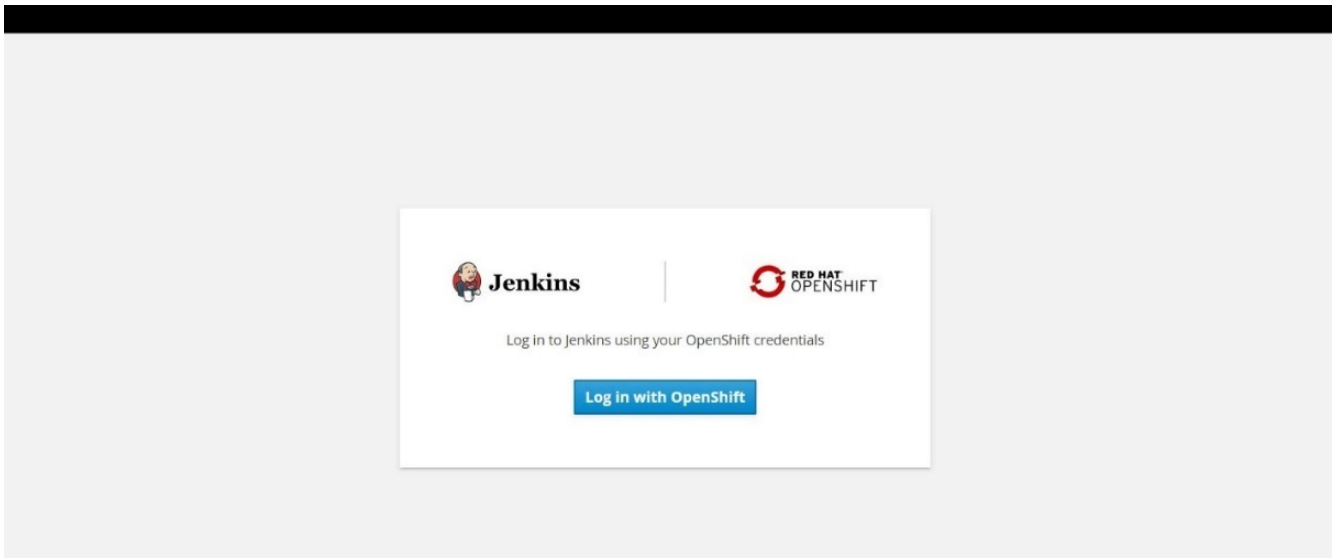
4. Jenkins Pod 大约需要 10 到 12 分钟才能进入就绪状态。



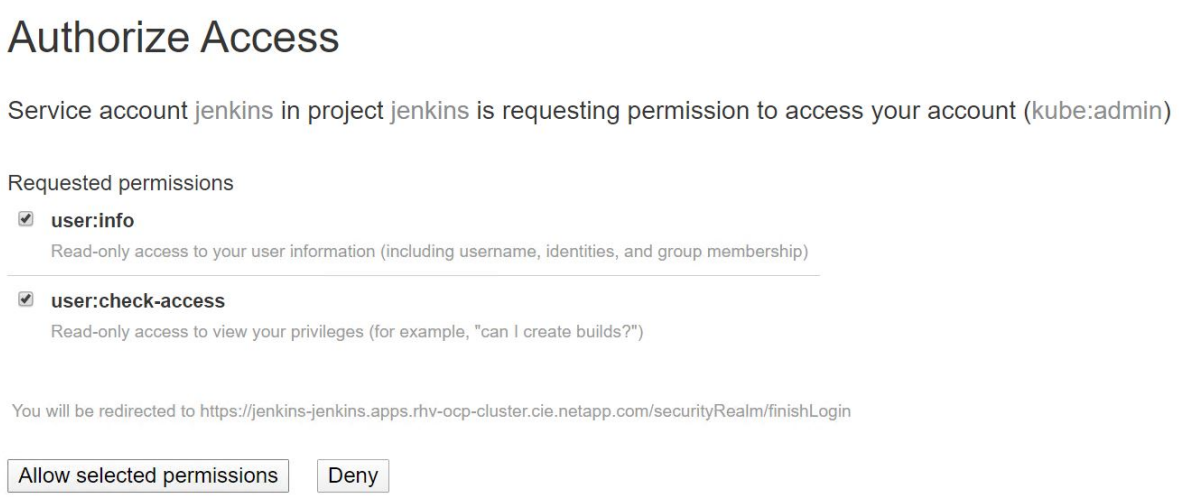
5. 实例化 Pod 后，导航到 "Networking" （网络） >"routes" （路由）。要打开 Jenkins 网页，请单击为 Jenkins 路由提供的 URL 。



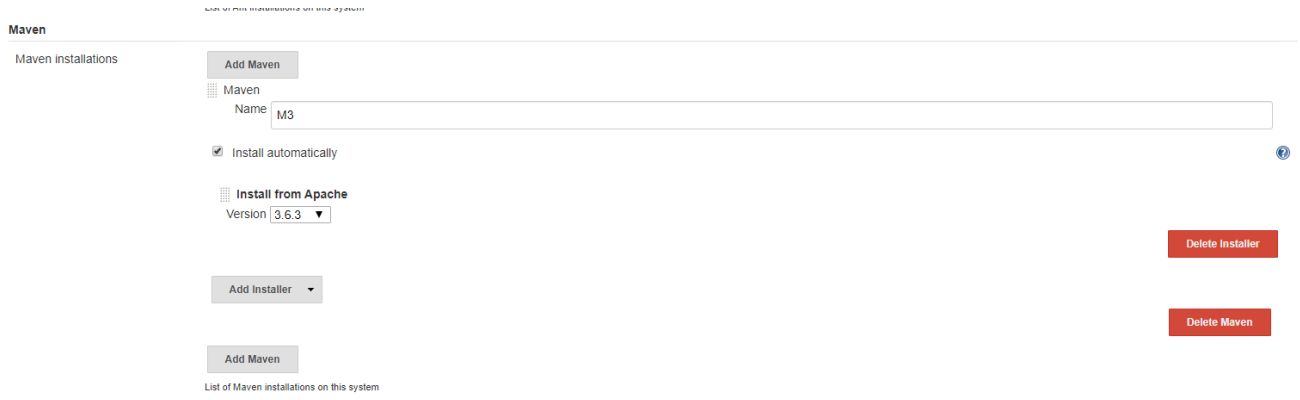
6. 由于在创建 Jenkins 应用程序时使用了 OpenShift OAuth ，因此请单击使用 OpenShift 登录。



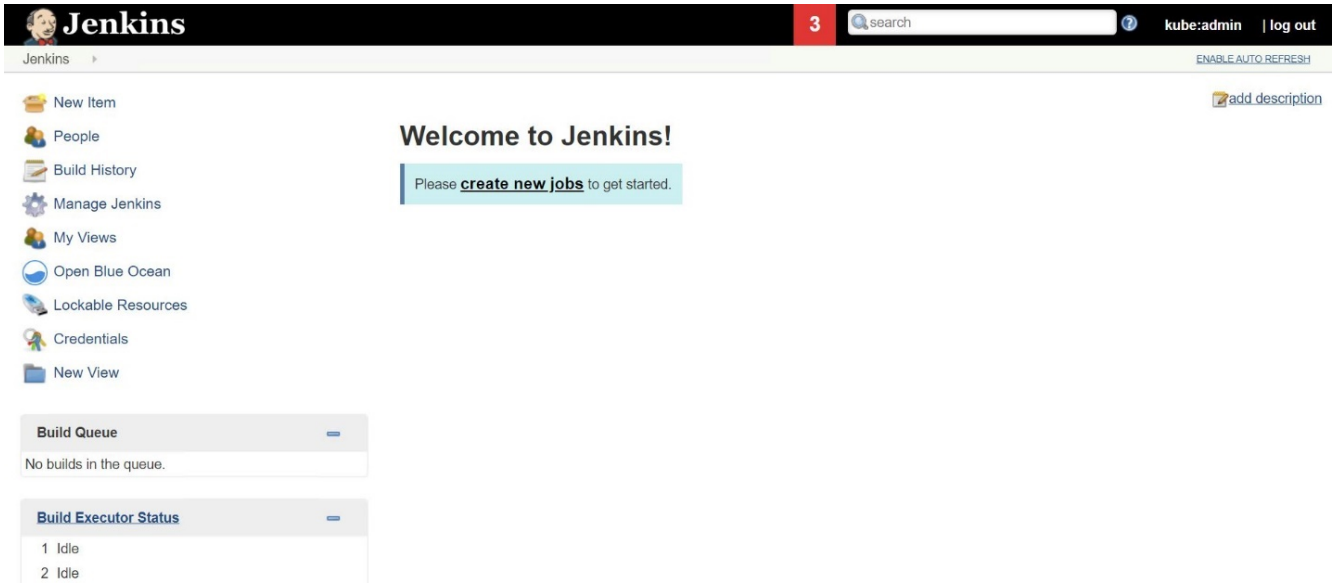
7. 授权 Jenkins 服务帐户访问 OpenShift 用户。



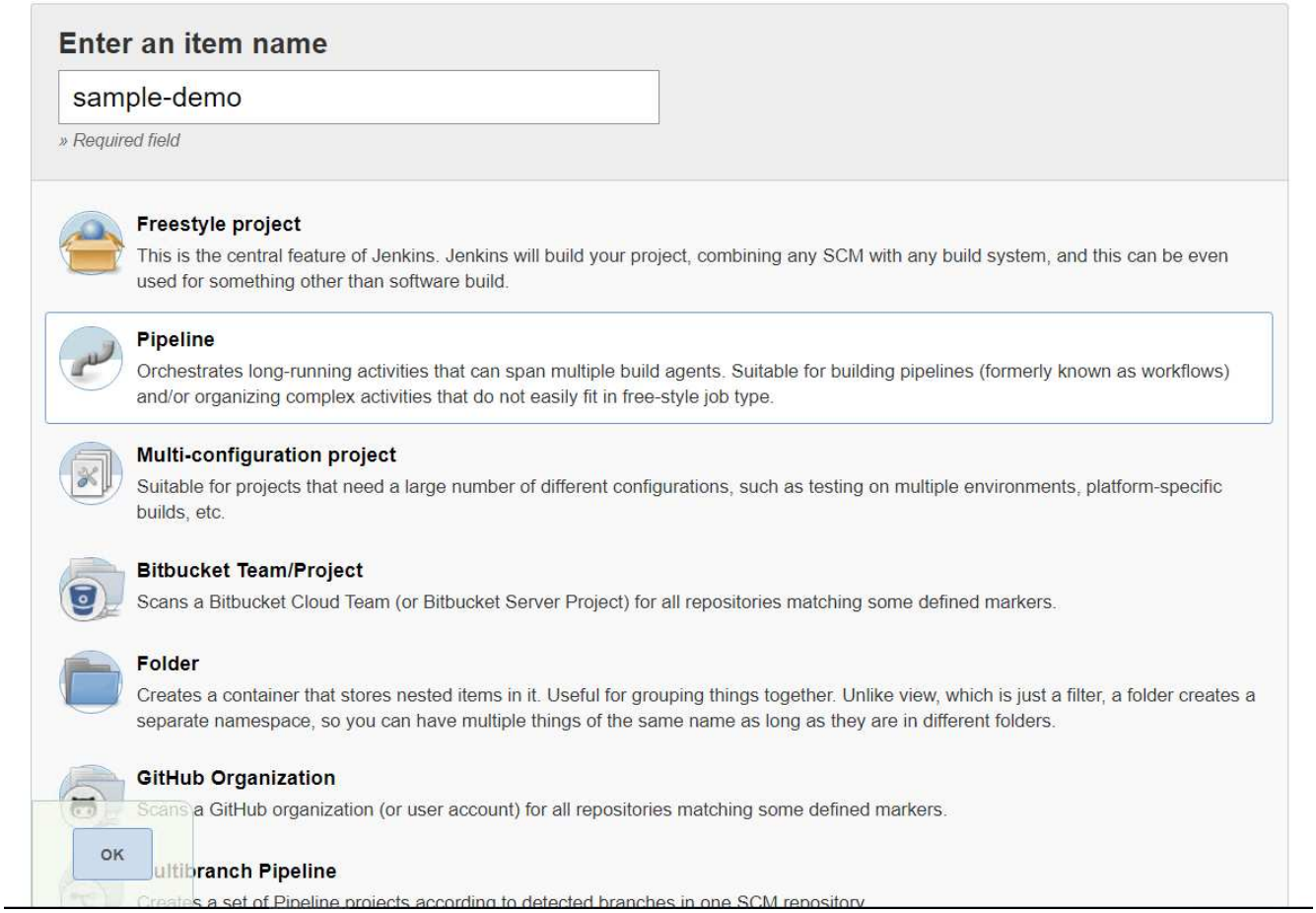
8. 此时将显示 Jenkins 欢迎页面。由于我们使用的是 Maven 内部版本，因此请先完成 Maven 安装。导航到 Manage Jenkins > Global Tool Configuration ，然后在 Maven 子标题中单击 Add Maven 。输入您选择的名称，并确保已选中自动安装选项。单击保存。



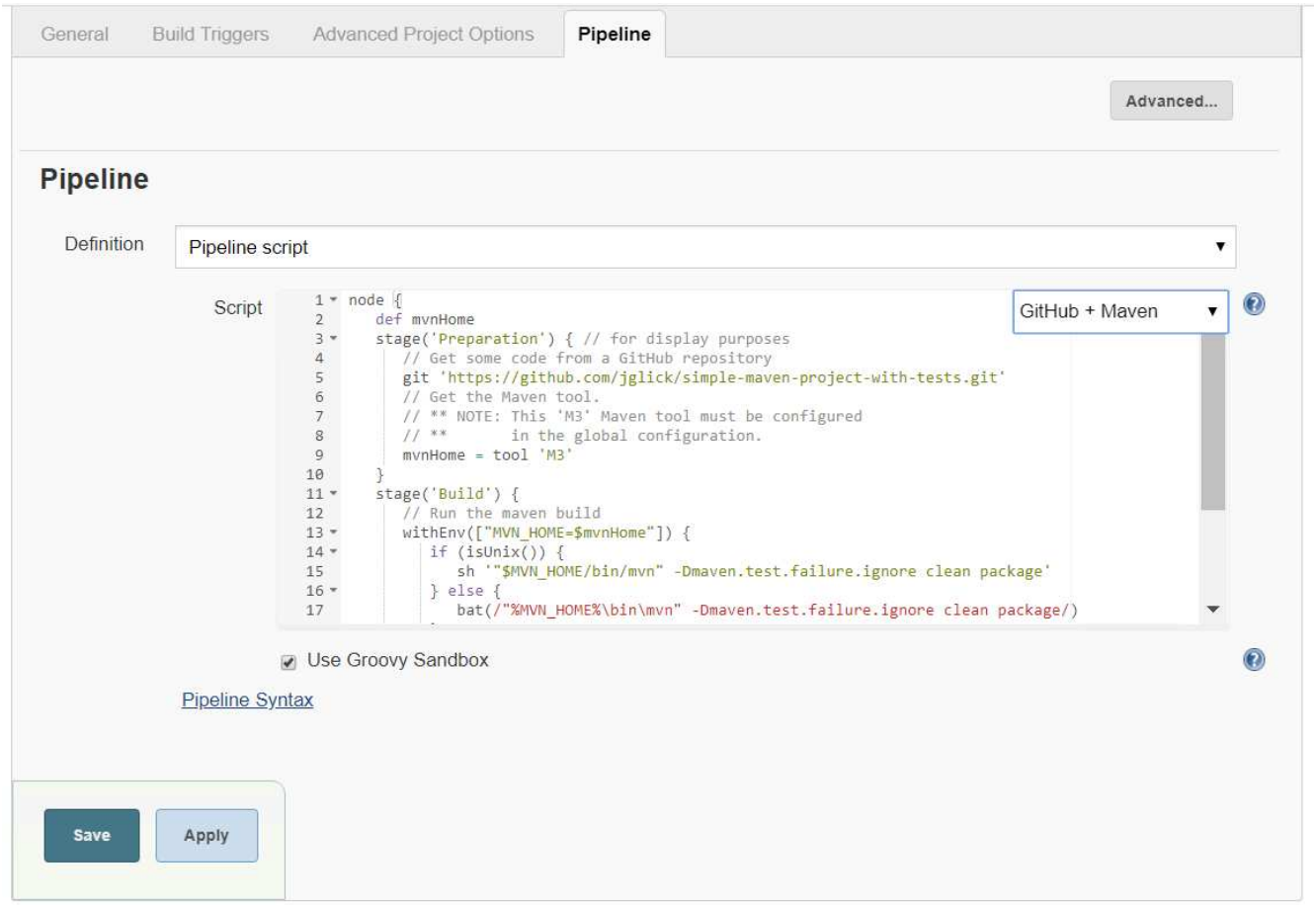
9. 现在，您可以创建一个管道来演示 CI/CD 工作流。在主页上，单击左侧菜单中的创建新作业或新建项目。



10. 在创建项目页面上，输入所选名称，选择管道，然后单击确定。



11. 选择管道选项卡。从试用样本管道下拉菜单中，选择 Github + Maven 。代码将自动填充。单击保存。



12. 单击 Build now ，在准备，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的时间。

13. 只要代码发生任何更改，就可以重新构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 Recent Changes 以跟踪与先前版本相比的更改。

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

在使用 **NetApp** 的 **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往往会为每个应用程序或工作负载部署一个 Red Hat OpenShift 集群。这样，他们就可以对应用程序或工作负载实施严格的隔离，优化性能并减少安全漏洞。但是，为每个应用程序部署一个单独的 Red Hat OpenShift 集群会产生自己的一系列问题。它增加了单独监控和管理每个集群所需的运营开销，由于为不同应用程序配置了专用资源而增加了成本，并妨碍了高效的可扩展性。

要解决这些问题，可以考虑在一个 Red Hat OpenShift 集群中运行所有应用程序或工作负载。但是，在这种架构中，资源隔离和应用程序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溢出到另一个工作负载，从而增加影响区域。此外，一个应用程序的任何突然不受控制的资源利用率都会影响另一个应用程序的性能，因为默认情况下没有资源分配策略。

因此，企业需要寻找在这两种环境中都能获得最佳性能的解决方案，例如，允许他们在一个集群中运行所有工作负载，同时为每个工作负载提供专用集群的优势。

其中一个有效的解决方案是在 Red Hat OpenShift 上配置多租户。多租户是一种架构，允许多个租户在同一集群上共存，并正确隔离资源，安全性等。在这种情况下，可以将租户视为集群资源的一部分，这些资源配置为供特定用户组专用使用。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，降低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违规交叉影响
- 保护工作负载，防止因资源争用而导致性能意外下降

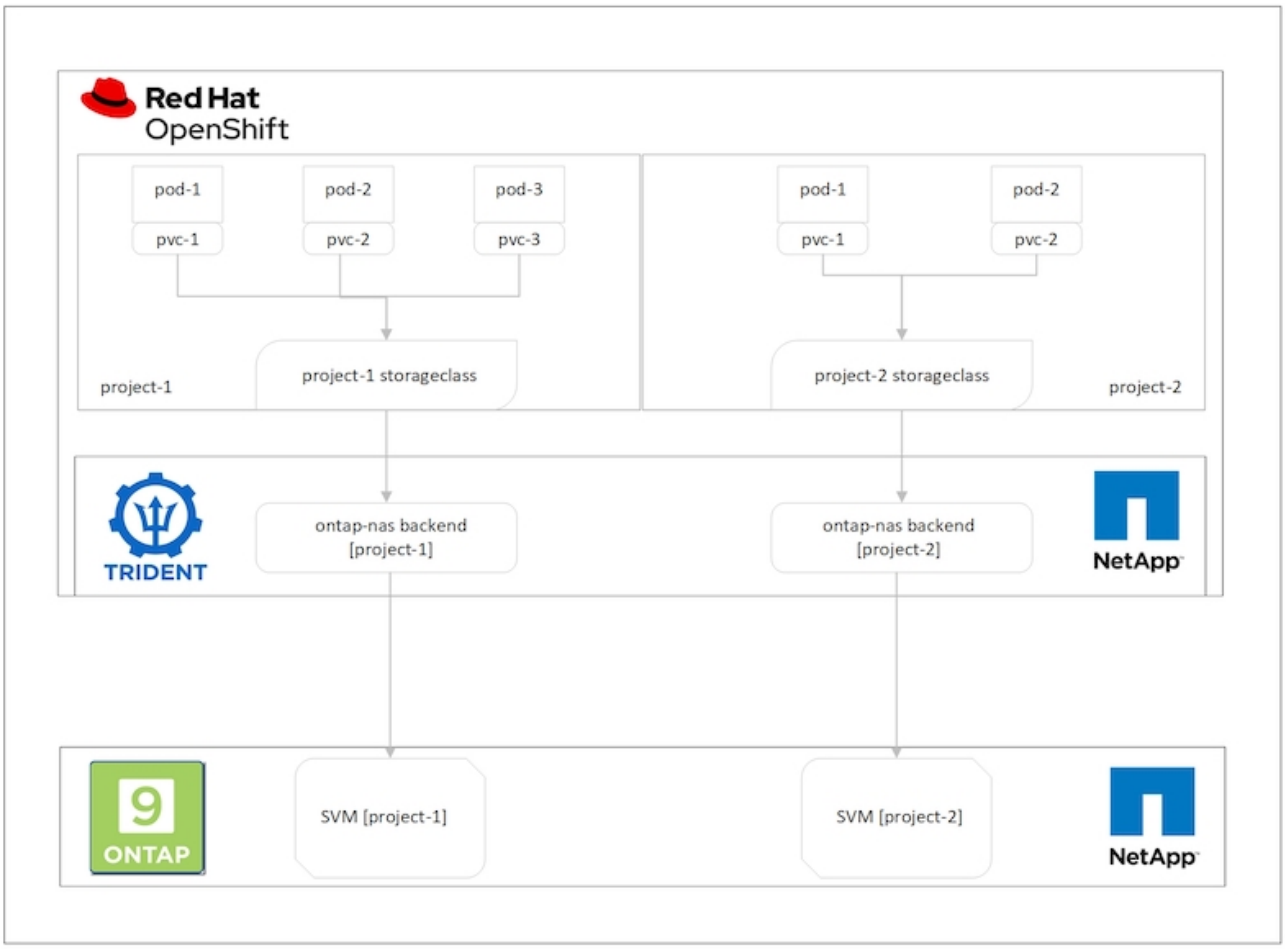
对于完全实现的多租户 OpenShift 集群，必须为属于不同资源分段的集群资源配置配额和限制：计算，存储，网络连接，安全性等。虽然我们会介绍此解决方案中所有资源分段的某些方面， 我们重点介绍最佳实践，通过在由 NetApp ONTAP 提供支持的 Astra Trident 动态分配的存储资源上配置多租户，隔离并保护同一 Red Hat OpenShift 集群上多个工作负载提供或使用的数据。

架构

虽然默认情况下，由 NetApp ONTAP 提供支持的 Red Hat OpenShift 和 Astra Trident 不会在工作负载之间提供隔离，但它们提供了广泛的功能，可用于配置多租户。为了更好地了解如何在采用 NetApp ONTAP 支持的 Astra Trident 的 Red Hat OpenShift 集群上设计多租户解决方案，让我们考虑一个包含一系列要求的示例，并概述其配置。

假设一个组织在一个 Red Hat OpenShift 集群上运行两个工作负载，作为两个不同团队正在处理的两个项目的一部分。这些工作负载的数据驻留在由 NetApp ONTAP NAS 后端的 Astra Trident 动态配置的 PVC 上。该组织需要为这两个工作负载设计多租户解决方案，并隔离用于这些项目的资源，以确保保持安全性和性能，主要侧重于为这些应用程序提供服务的数据。

下图展示了由 NetApp ONTAP 提供支持的带有 Astra 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. Astra Trident

Red Hat OpenShift —集群资源

从 Red Hat OpenShift 集群的角度来看，要开始使用的顶级资源是项目。OpenShift 项目可以视为将整个 OpenShift 集群划分为多个虚拟集群的集群资源。因此，项目级别的隔离为配置多租户提供了基础。

下一步是在集群中配置 RBAC 。最佳做法是，将处理单个项目或工作负载的所有开发人员配置到身份提供程序（ IdP ）中的单个用户组中。Red Hat OpenShift 允许 IdP 集成和用户组同步，从而允许将 IdP 中的用户和组导入到集群中。这样可以帮助集群管理员将项目专用集群资源的访问权限隔离给一个或多个处理该项目的用户组，从而限制对任何集群资源的未授权访问。要了解有关 IdP 与 Red Hat OpenShift 集成的详细信息，请参见相关文档 "此处"。

NetApp ONTAP

必须隔离用作 Red Hat OpenShift 集群永久性存储提供程序的共享存储，以确保在存储上为每个项目创建的卷在主机上显示为它们，就像在单独的存储上创建一样。为此，请在 NetApp ONTAP 上创建与项目或工作负载数量相同的 SVM （ Storage Virtual Machine ），并将每个 SVM 专用于一个工作负载。

Astra Trident

在 NetApp ONTAP 上为不同项目创建不同的 SVM 之后，必须将每个 SVM 映射到不同的 Trident 后端。Trident 上的后端配置会将永久性存储分配给 OpenShift 集群资源，并且需要将 SVM 的详细信息映射到。此驱动程序至少应为后端的协议驱动程序。或者，您也可以通过它定义如何在存储上配置卷，并设置卷大小或聚合使用量等限制。有关 Trident 后端定义的详细信息，请参见 "此处"。

Red Hat OpenShift —存储资源

配置 Trident 后端后，下一步是配置 StorageClasses 。配置与后端相同数量的存储类，为每个存储类提供访问权限，以便仅在一个后端启动卷。在定义存储类时，我们可以使用 storagePools 参数将 StorageClass 映射到特定的 Trident 后端。可以找到用于定义存储类的详细信息 "此处"。因此，从 StorageClass 到 Trident 后端存在一对一映射，这种映射可指向一个 SVM 。这样可以确保通过分配给该项目的 StorageClass 处理的所有存储请求仅由专用于该项目的 SVM 处理。

由于存储类不是命名空间资源，我们如何确保拒绝另一命名空间或项目中的 Pod 向一个项目的存储类声明？问题解答将使用 ResourceQuotas 。ResourceQuotas 是控制每个项目资源总使用量的对象。它可以限制项目中的对象可以使用的资源数量以及总资源量。使用 ResourceQuotas 几乎可以限制项目中的所有资源，而高效地使用此功能可以帮助组织降低因过度配置或过度消耗资源而导致的成本和中断。请参见文档 "此处" 有关详细信息 …

对于这种使用情形，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申请存储。为此，我们需要通过将 ` <storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims` 设置为 0 来限制其他存储类的永久性卷请求。此外，集群管理员必须确保项目中的开发人员不应有权修改 ResourceQuotas 。

Configuration

对于任何多租户解决方案，任何用户都无法访问比所需更多的集群资源。因此，要在多租户配置中配置的整个资源集将在集群管理员，存储管理员和处理每个项目的开发人员之间进行划分。

下表概括了不同用户要执行的不同任务：

Role	任务
* 集群管理 *	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
	为分配对特定项目的访问权限的开发人员创建角色和角色绑定
	[可选] 配置项目以在特定节点上计划 Pod
* 存储管理 *	在 NetApp ONTAP 上创建 SVM
	创建 Trident 后端
	创建 StorageClasses
	创建存储 ResourceQuotas
* 开发人员 *	验证对已分配项目中的 PVC 或 Pod 的创建或修补访问权限
	验证对在其他项目中创建或修补 PVC 或 Pod 的访问权限
	验证对查看或编辑项目， ResourceQuotas 和 StorageClasses 的访问权限

Configuration

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 工具并将其添加到 $path 中的管理工作站。
- 对 ONTAP 的管理员访问权限。
- 对 OpenShift 集群的集群管理员访问。
- 集群已与身份提供程序集成。
- 身份提供程序经过配置，可以有效区分不同团队中的用户。

配置： **cluster-admin** 任务

Red Hat OpenShift cluster-admin 执行以下任务：

1. 以 cluster-admin 身份登录到 Red Hat OpenShift 集群。
2. 创建两个与不同项目对应的项目。

```
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
```

本节中提供的角色定义只是一个示例。必须根据最终用户要求定义开发人员角色。

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。存储管理员的访问由安装 Trident 时创建的 Trident 操作员角色控制。此外，存储管理员还需要访问 ResourceQuotas 来控制存储的使用方式。
3. 在集群中的所有项目中创建一个用于管理 ResourceQuotas 的角色，以将其连接到存储管理员：

```
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程序集成，并且用户组与集群组同步。以下示例显示身份提供程序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 操作员和资源配额。

1. 为开发人员创建 RoleBindings ，将开发人员项目 1 角色绑定到项目 1 中的相应组（ OCP-project-1 ）。

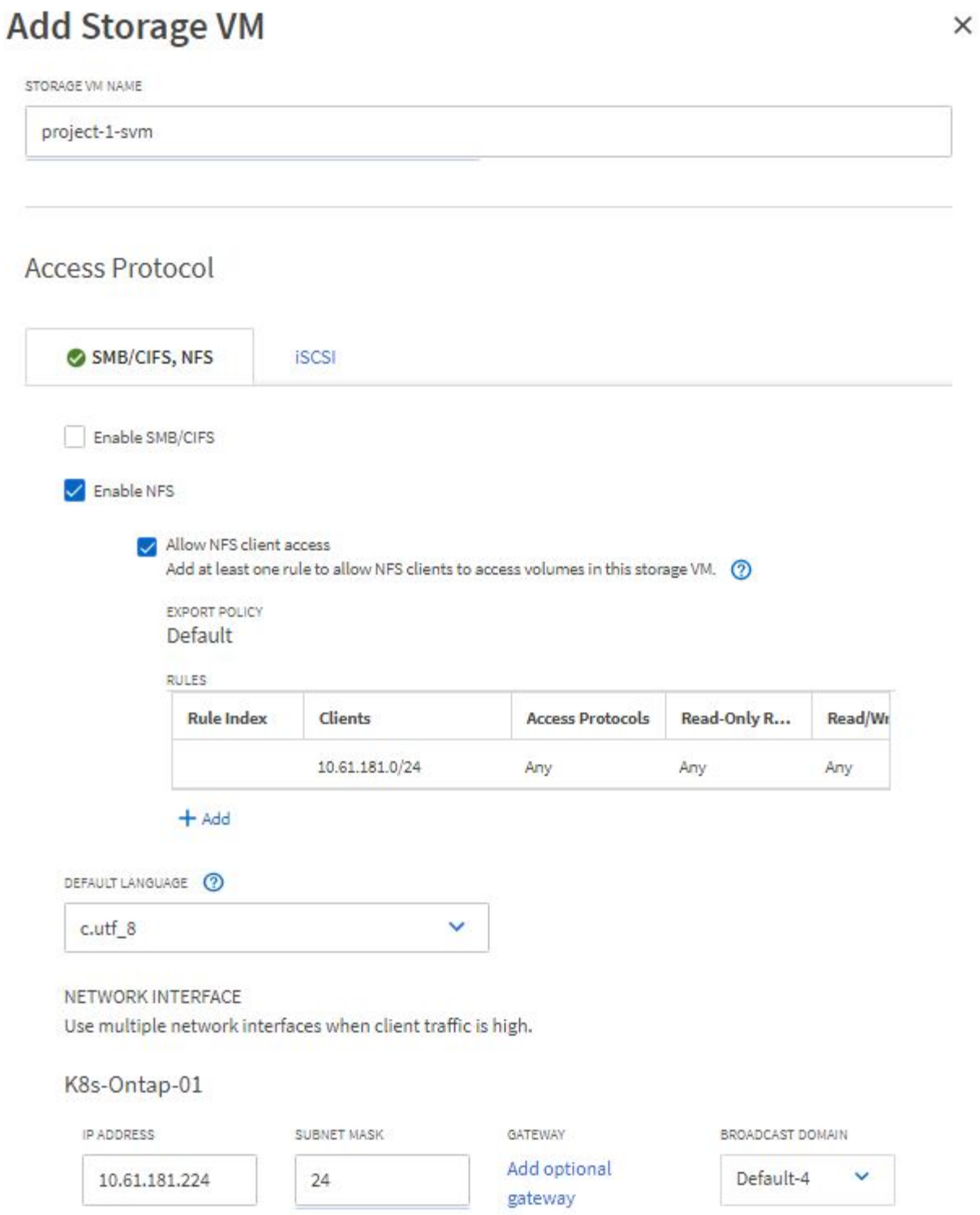
```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-1-developer
  namespace: project-1
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role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

2. 同样，为开发人员创建 RoleBindings ，将开发人员角色绑定到 project-2 中的相应用户组。

配置： **storage-admin** 任务

存储管理员必须配置以下资源：

1. 以管理员身份登录到 NetApp ONTAP 集群。
2. 导航到存储 > Storage VM ，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 project-1 ，另一个用于 project-2 。此外，还可以创建 vsadmin 帐户来管理 SVM 及其资源。



1. 以存储管理员身份登录到 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
```

在此示例中，我们使用的是 ontap-NAS 驱动程序。根据使用情形创建后端时，请使用相应的驱动程序。

我们假定 Trident 已安装在 Trident 项目中。

1. 同样，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后端专用于 project-1 的存储池。

```
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 以限制 project-1 中的资源，从而从专用于其他项目的存储库请求存储。

```
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. 同样，也可以创建 ResourceQuota 来限制项目 2 中的资源，以便从专用于其他项目的存储库请求存储。

验证

要验证在上述步骤中配置的多租户架构，请完成以下步骤：

验证在分配的项目中创建 **PVC** 或 **Pod** 的访问权限

1. 以项目 1 中的 OCP-project-1-user 和开发人员身份登录。
2. 检查访问权限以创建新项目。

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

3. 使用分配给 project-1 的 storageclass 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 及其卷是否已在 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 POD ，然后挂载上一步创建的 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已挂载卷。

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

验证在其他项目中创建 **PVC** 或 **Pod** 的访问权限，或者使用专用于另一项目的资源

1. 以项目 1 中的 OCP-project-1-user 和开发人员身份登录。
2. 使用分配给 project-2 的 storageclass 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. 请确保未创建 PVC `test-vpa-project-1-sc-2` 和 `test-vpa-project-2-sc-1` 。

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

5. 在 project-2 中创建 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
```

验证对查看和编辑项目， **ResourceQuotas** 和 **StorageClasses** 的访问权限

1. 以项目 1 中的 OCP-project-1-user 和开发人员身份登录。
2. 检查访问权限以创建新项目。

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

3. 验证对查看项目的访问权限。

```
oc get ns
```

4. 检查用户是否可以在 project-1 中查看或编辑 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器。

```
oc get sc
```

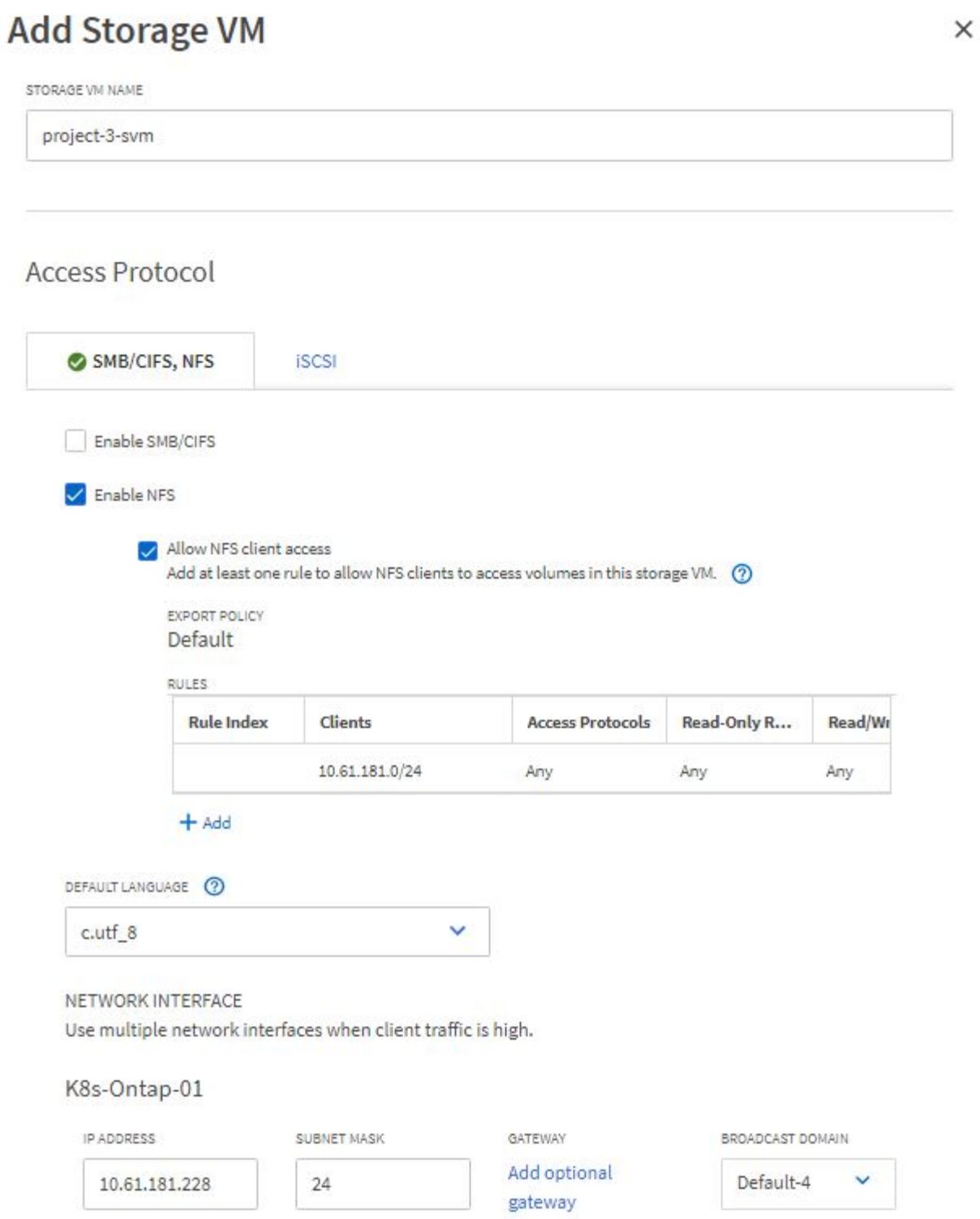
6. 检查访问权限以描述存储器。
7. 验证用户的访问权限以编辑存储器库。

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

扩展：添加更多项目

在多租户配置中，使用存储资源添加新项目需要进行额外配置，以确保不会违反多租户要求。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到 NetApp ONTAP 集群。
2. 导航到 `Storage → Storage VM`，然后单击 `Add`。创建一个专用于 project-3 的新 SVM 。此外，还可以创建 vsadmin 帐户来管理 SVM 及其资源。



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

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

3. 确保已在 IdP 上为 project-3 创建用户组并与 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
```

本节中提供的角色定义只是一个示例。必须根据最终用户要求定义开发人员角色。

1. 在 project-3 中为开发人员创建 RoleBinding. 将开发人员项目 3 角色绑定到 project-3 中的相应组（ 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到 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
```

在此示例中，我们使用的是 ontap-NAS 驱动程序。使用相应的驱动程序根据使用情形创建后端。

我们假定 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 以限制项目 3 中的资源，从而从专用于其他项目的存储库请求存储。

```
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. 在其他项目中修补 ResourceQuotas ，以限制这些项目中的资源从专用于项目 3 的存储库访问存储。

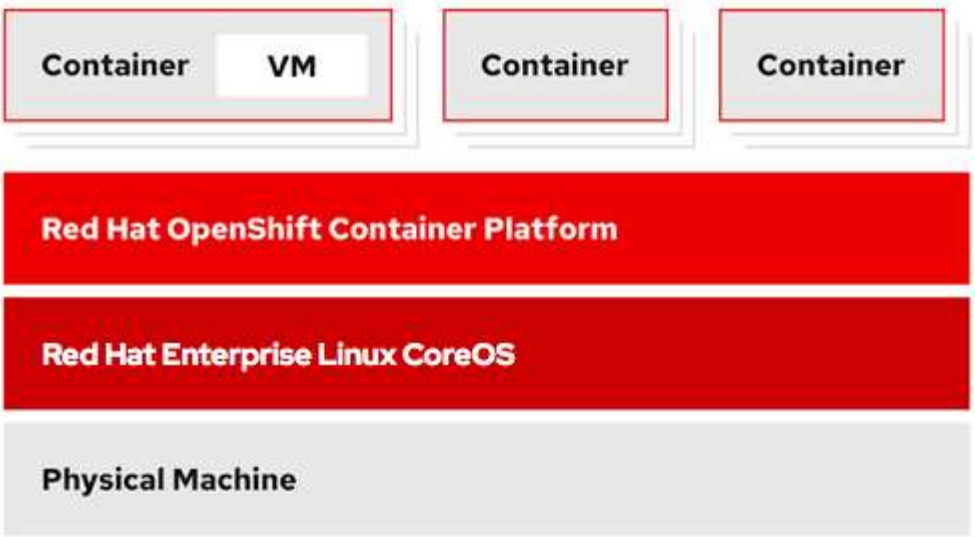
```
oc patch resourcequotas project-1-sc-rq -n project-1 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
oc patch resourcequotas project-2-sc-rq -n project-2 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims":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

借助 NetApp ONTAP 实现 Red Hat OpenShift 虚拟化

借助 **NetApp ONTAP** 实现 **Red Hat OpenShift** 虚拟化

根据具体使用情形，容器和虚拟机（ VM ）均可用作不同类型应用程序的最佳平台。因此，许多组织在容器上运行部分工作负载，而在 VM 上运行部分工作负载。通常，这会导致企业面临额外的挑战，需要管理不同的平台：虚拟机管理程序和应用程序容器编排程序。

为了应对这一挑战， Red Hat 从 OpenShift 4.6 版开始引入了 OpenShift 虚拟化（以前称为容器原生虚拟化）。通过 OpenShift 虚拟化功能，您可以在同一 OpenShift 容器平台安装中运行和管理虚拟机以及容器，从而提供混合管理功能，通过操作员自动部署和管理 VM 。除了使用 OpenShift 虚拟化在 OpenShift 中创建 VM 之外，Red Hat 还支持从 VMware vSphere ， Red Hat 虚拟化和 Red Hat OpenStack Platform 部署导入 VM 。



在由 NetApp ONTAP 提供支持的 Astra Trident 的协助下， OpenShift 虚拟化还支持某些功能，例如实时 VM 迁移， VM 磁盘克隆， VM 快照等。这些工作流的示例将在本文档后面的相应章节中进行讨论。

要了解有关 Red Hat OpenShift 虚拟化的详细信息，请参见相关文档 "此处"。

部署OpenShift虚拟化

使用 NetApp ONTAP 部署 Red Hat OpenShift 虚拟化

前提条件

- Red Hat OpenShift 集群（ 4.6 版之后的版本），安装在具有 RHCOS 工作节点的裸机基础架构上
- OpenShift 集群必须通过安装程序配置的基础架构（ IPI ）进行安装
- 部署计算机运行状况检查以保持虚拟机的 HA
- NetApp ONTAP 集群
- 安装在 OpenShift 集群上的 Astra Trident
- 在 ONTAP 集群上配置了 SVM 的 Trident 后端
- 一种在 OpenShift 集群上配置的存储类，其中使用 Astra Trident 作为配置程序
- 对 Red Hat OpenShift 集群的集群管理员访问
- 对 NetApp ONTAP 集群的管理员访问权限
- 安装了 tridentctl 和 oc 工具并将其添加到 $path 中的管理工作站

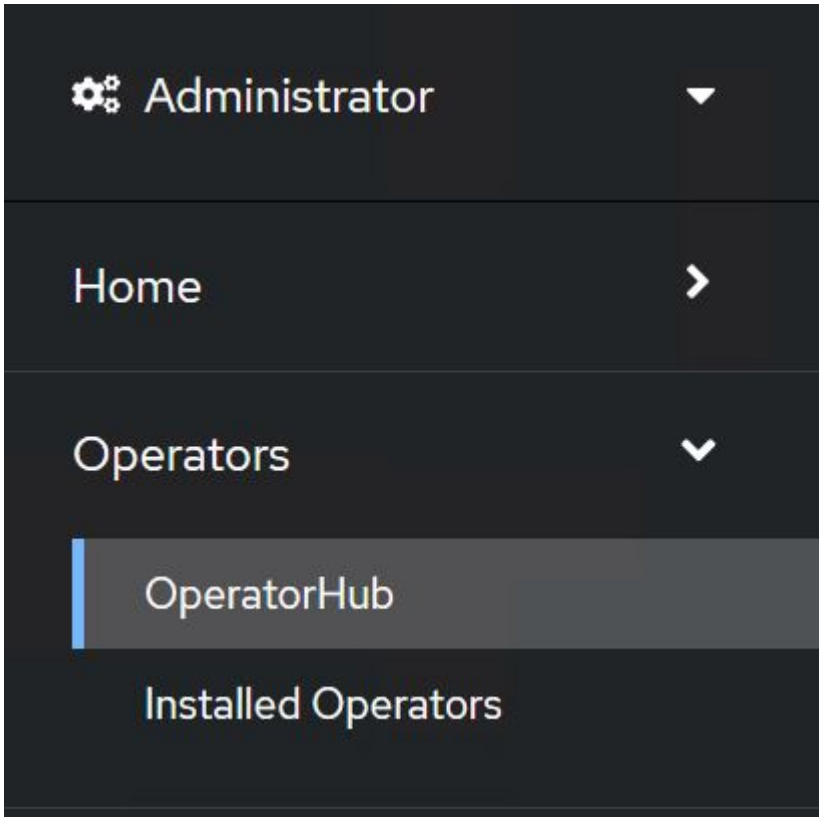
由于 OpenShift 虚拟化由 OpenShift 集群上安装的操作员管理，因此会增加内存， CPU 和存储的开销，在规划集群的硬件要求时必须考虑这些开销。请参见文档 "此处" 有关详细信息：

或者，您也可以通过配置节点放置规则来指定一组 OpenShift 集群节点，以托管 OpenShift 虚拟化操作员，控制器和 VM 。要为 OpenShift 虚拟化配置节点放置规则，请按照文档进行操作 "此处"。

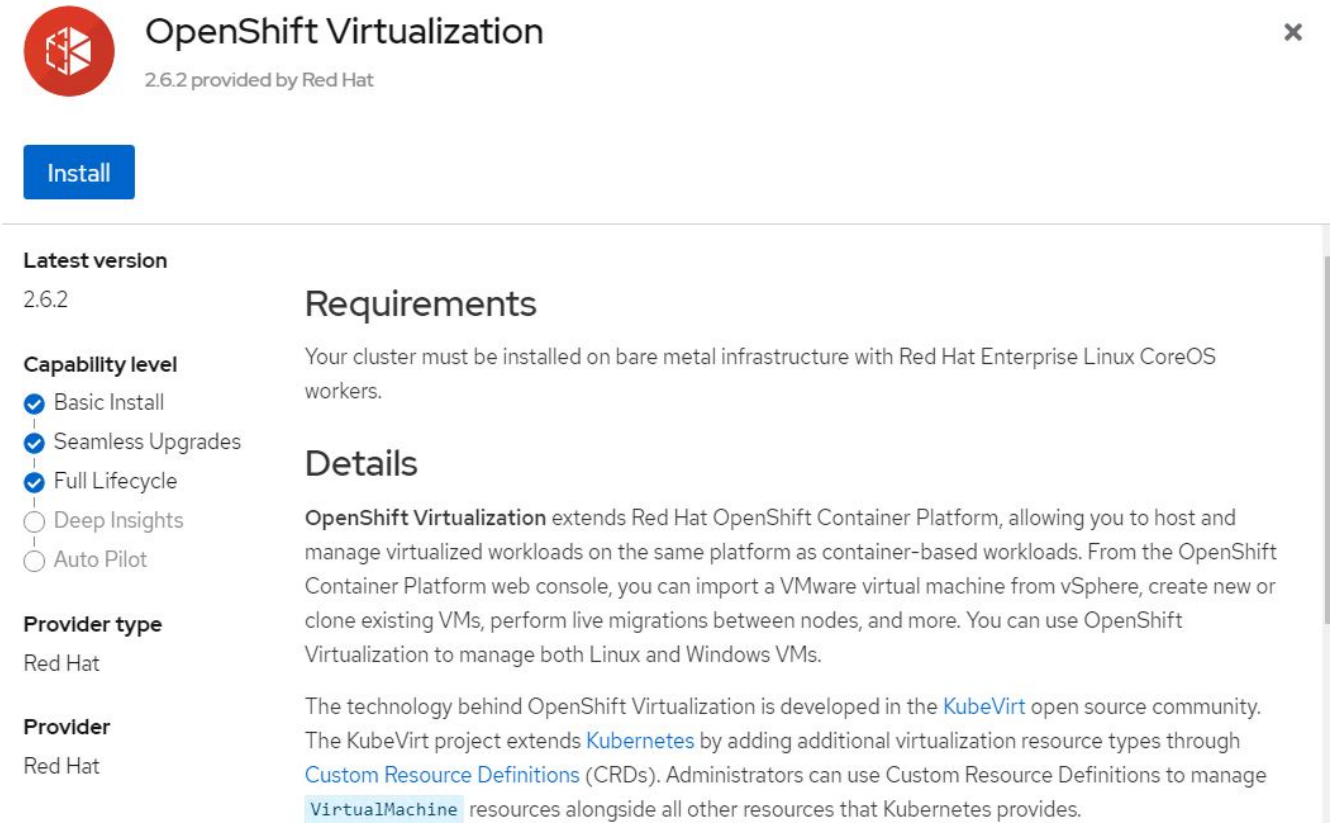
对于支持 OpenShift 虚拟化的存储， NetApp 建议使用一个专用 StorageClass ，以便从特定 Trident 后端请求存储，而该后端又由专用 SVM 提供支持。这样就可以在 OpenShift 集群上为基于 VM 的工作负载提供的数据方面保持多租户级别。

要安装 OpenShift 虚拟化，请完成以下步骤：

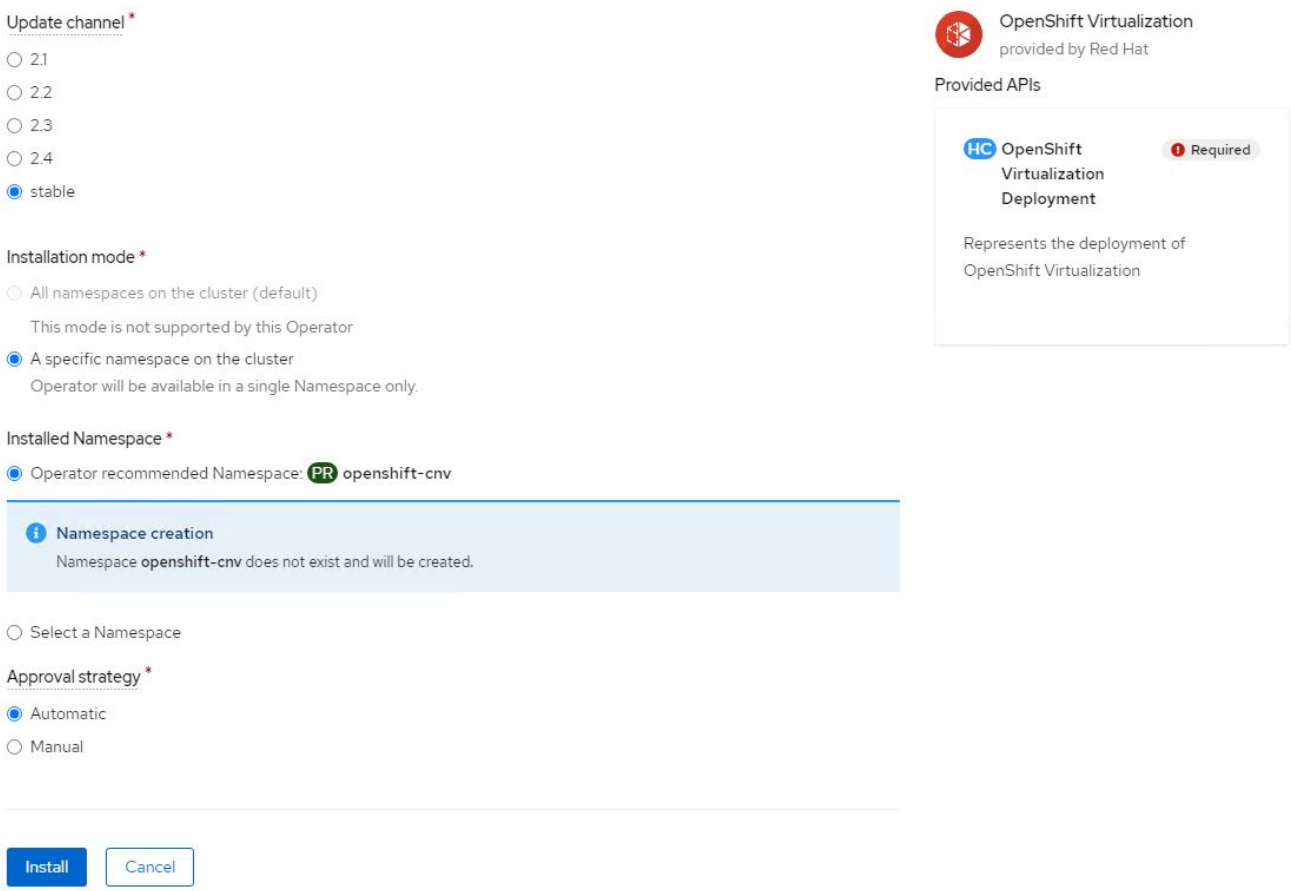
1. 使用 cluster-admin 访问权限登录到 Red Hat OpenShift 裸机集群。
2. 从 "Perspective" 下拉列表中选择 "Administrator" 。
3. 导航到 Operators > OperatorHub 并搜索 OpenShift 虚拟化。



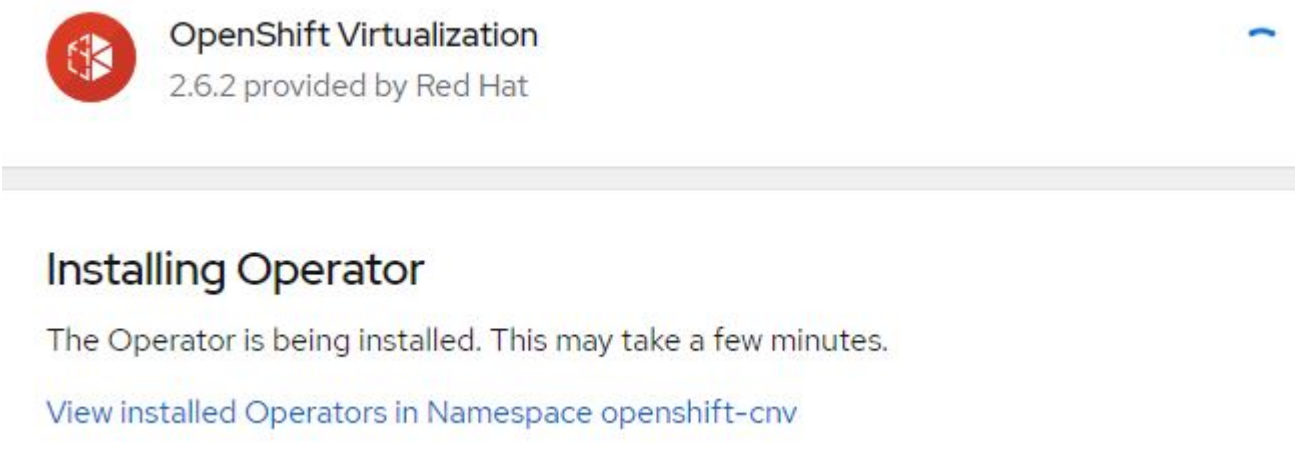
4. 选择 OpenShift 虚拟化磁贴，然后单击安装。



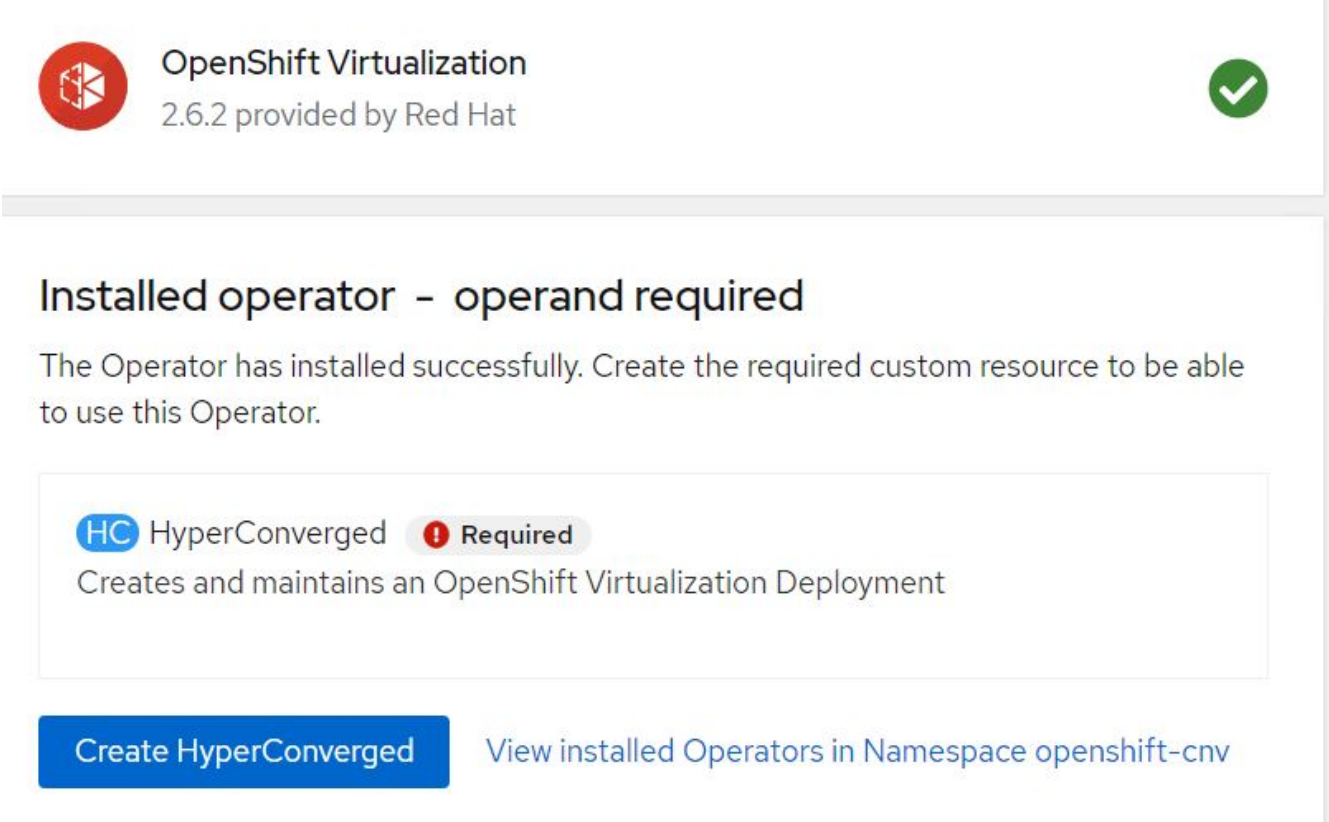
5. 在 Install Operator 屏幕上，保留所有默认参数，然后单击 Install 。



6. 等待操作员安装完成。



7. 安装操作员后，单击 Create HyperConverged 。



8. 在 Create HyperConverged 屏幕上，单击 Create ，接受所有默认参数。此步骤将开始安装 OpenShift 虚拟化。

Name *

kubevirt-hyperconverged

Labels

app=frontend

Infra

infra HyperConvergedConfig influences the pod configuration (currently only placement) for all the infra components needed on the virtualization enabled cluster but not necessarely directly on each node running VMs/VMIs.

Workloads

workloads HyperConvergedConfig influences the pod configuration (currently only placement) of components which need to be running on a node where virtualization workloads should be able to run. Changes to Workloads HyperConvergedConfig can be applied only without existing workload.

Bare Metal Platform

true

BareMetalPlatform indicates whether the infrastructure is baremetal.

Feature Gates

featureGates is a map of feature gate flags. Setting a flag to `true` will enable the feature. Setting `false` or removing the feature gate, disables the feature.

Local Storage Class Name

LocalStorageClassName the name of the local storage class.

Create Cancel

9. 在 OpenShift-cnv 命名空间中的所有 Pod 都变为 running 状态且 OpenShift 虚拟化操作员处于 succeeded 状态后，操作员便可随时使用了。现在，可以在 OpenShift 集群上创建 VM 。

Project: openshift-cnv

Installed Operators

Installed Operators are represented by ClusterServiceVersions within this Namespace. For more information, see the Understanding Operators documentation. Or create an Operator and ClusterServiceVersion using the Operator SDK.

Name Search by name...

Name	Managed Namespaces	Status	Last updated	Provided APIs
OpenShift Virtualization 2.6.2 provided by Red Hat	NS openshift-cnv	Succeeded Up to date	May 18, 8:02 pm	OpenShift Virtualization Deployment HostPathProvisioner deployment

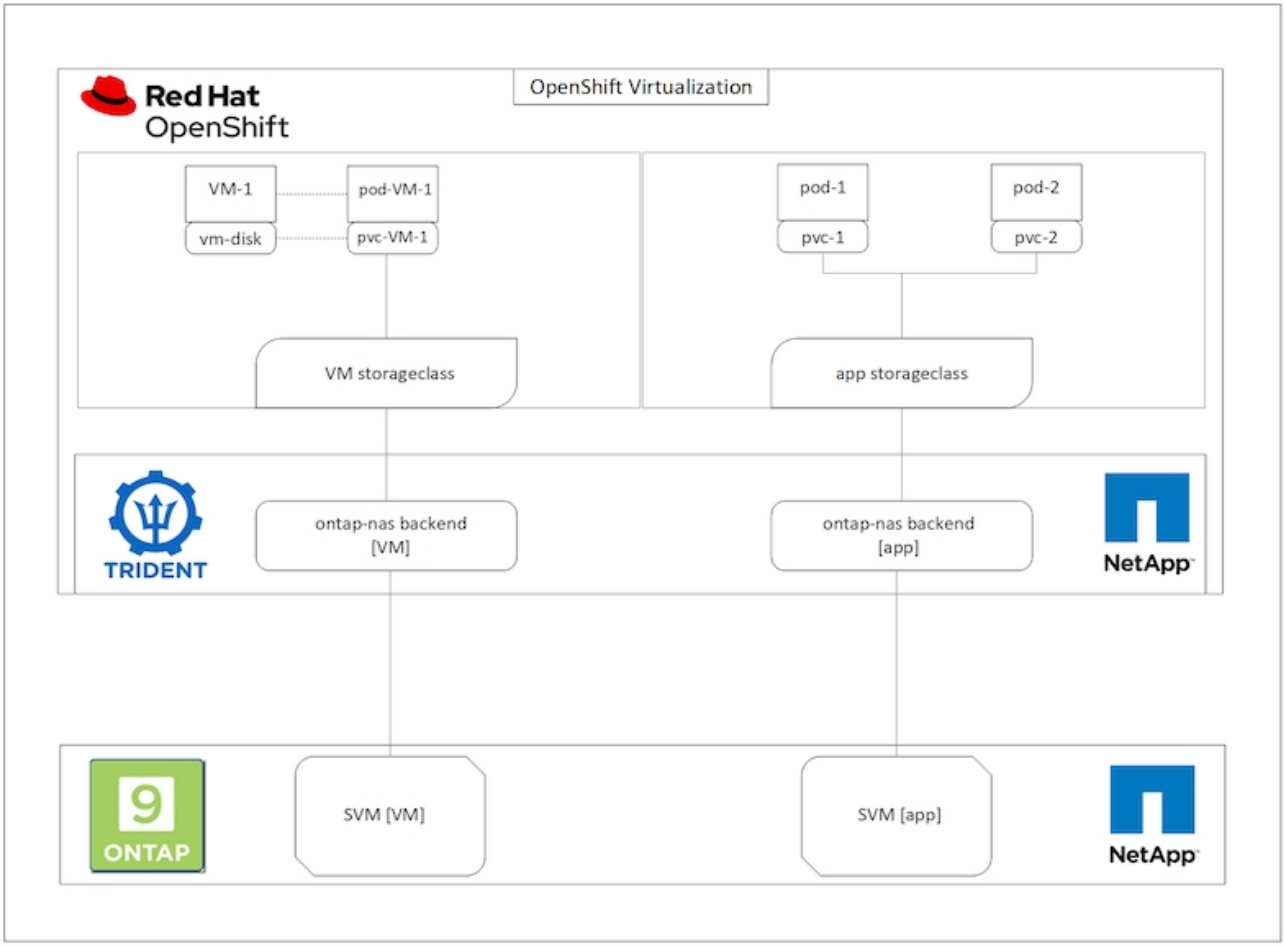
工作流

工作流：使用 **NetApp ONTAP** 实现 **Red Hat OpenShift** 虚拟化

创建虚拟机

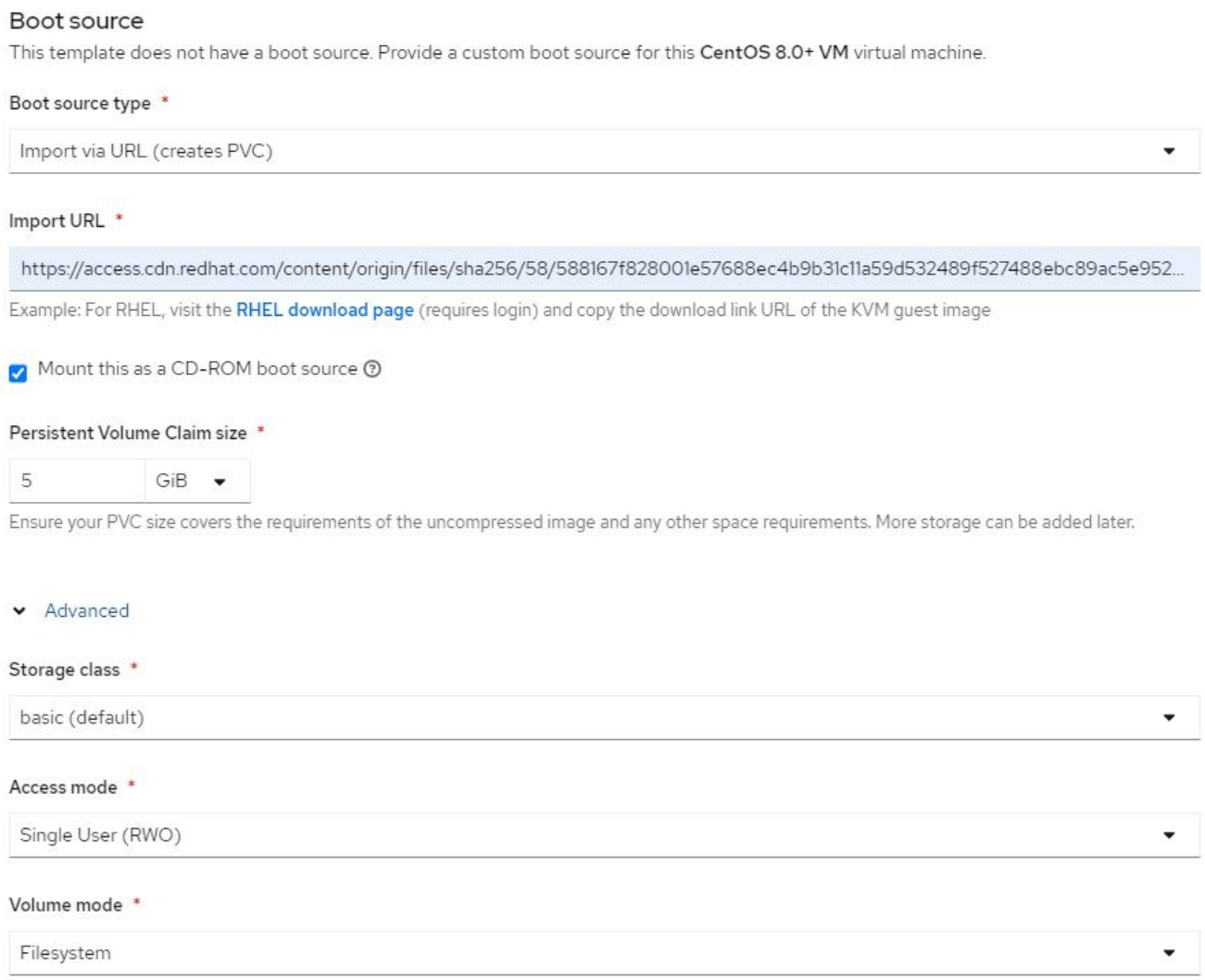
VM 是有状态部署，需要使用卷来托管操作系统和数据。使用 CNV 时，由于 VM 作为 Pod 运行，因此 VM 由 NetApp ONTAP 上通过 Trident 托管的 PV 提供支持。这些卷作为磁盘连接并存储整个文件系统，包括虚拟机的

启动源。

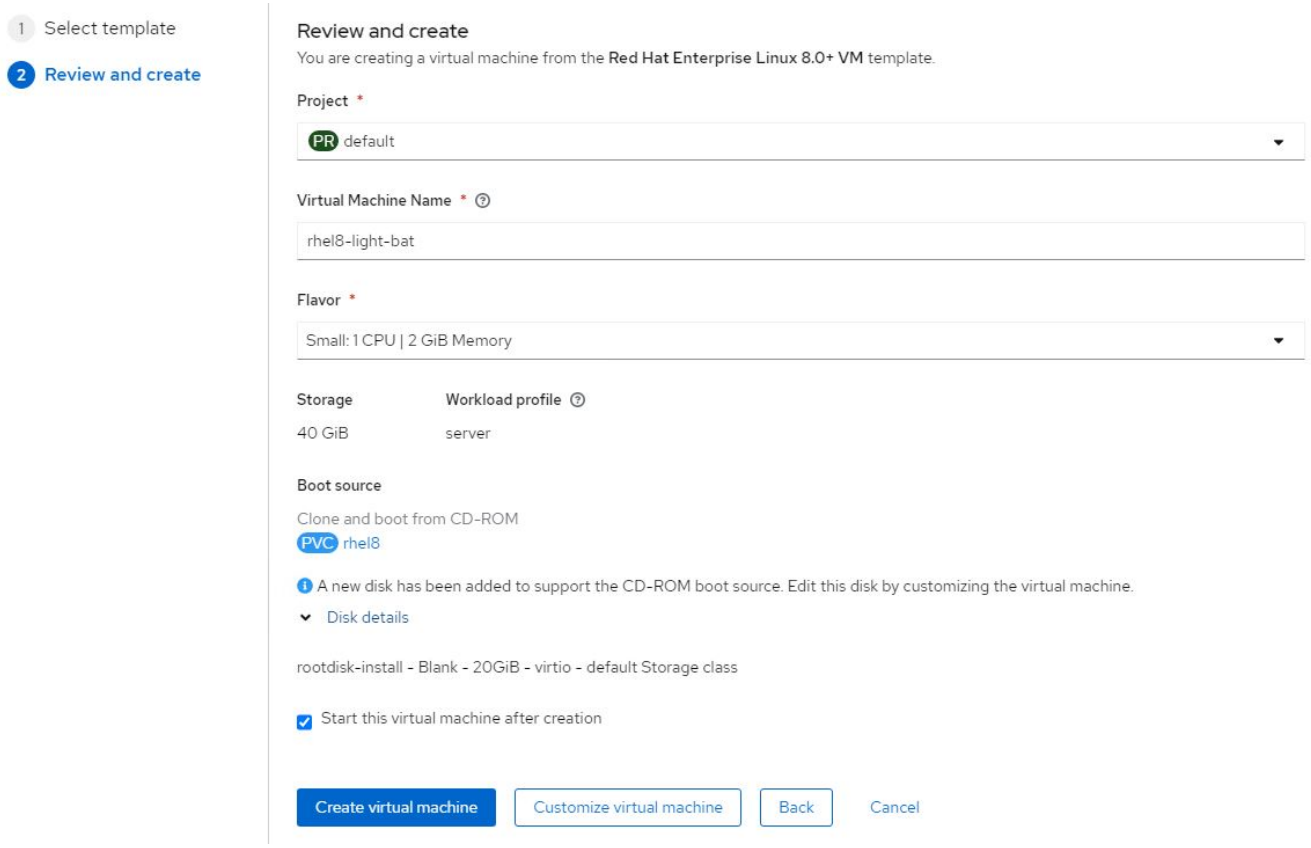


要在 OpenShift 集群上创建虚拟机，请完成以下步骤：

1. 导航到工作负载 > 虚拟化 > 虚拟机，然后单击创建 > 使用向导。
2. 选择所需的操作系统，然后单击下一步。
3. 如果选定操作系统未配置启动源，则必须对其进行配置。对于启动源，选择是要从 URL 还是从注册表导入操作系统映像，并提供相应的详细信息。展开高级并选择 Trident 支持的 StorageClass 。然后单击下一步。



4. 如果选定操作系统已配置启动源，则可以跳过上一步。
5. 在 Review and Create 窗格中，选择要在其中创建 VM 的项目并提供 VM 详细信息。确保选择了要克隆的启动源，并使用为选定操作系统分配的相应 PVC 从 CD-ROM 启动。



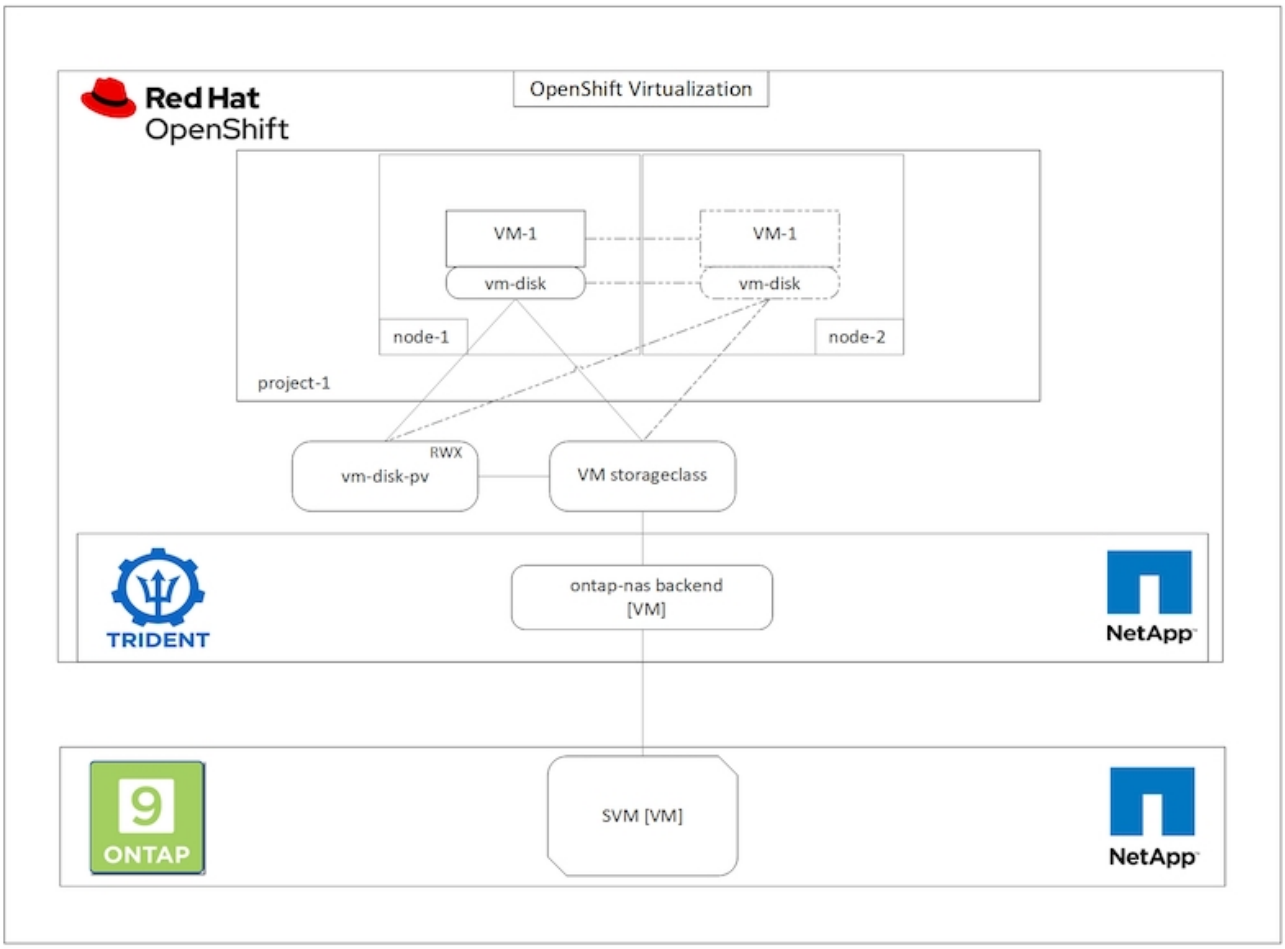
6. 如果要自定义虚拟机，请单击 "Customize Virtual Machine" 并修改所需的参数。
7. 单击 Create Virtual Machine 以创建虚拟机；此操作将在后台生成相应的 Pod 。

从 URL 或注册表为模板或操作系统配置启动源时，它会在 `OpenShift-virtual-os-images` 项目中创建一个 PVC ，并将 KVM 子映像下载到 PVC 。您必须确保模板 PVC 具有足够的已配置空间，以容纳相应操作系统的 KVM 子映像。然后，使用任何项目中的相应模板创建这些 PVC 时，这些 PVC 会克隆并作为根磁盘附加到虚拟机中。

工作流：使用 **NetApp ONTAP** 实现 **Red Hat OpenShift** 虚拟化

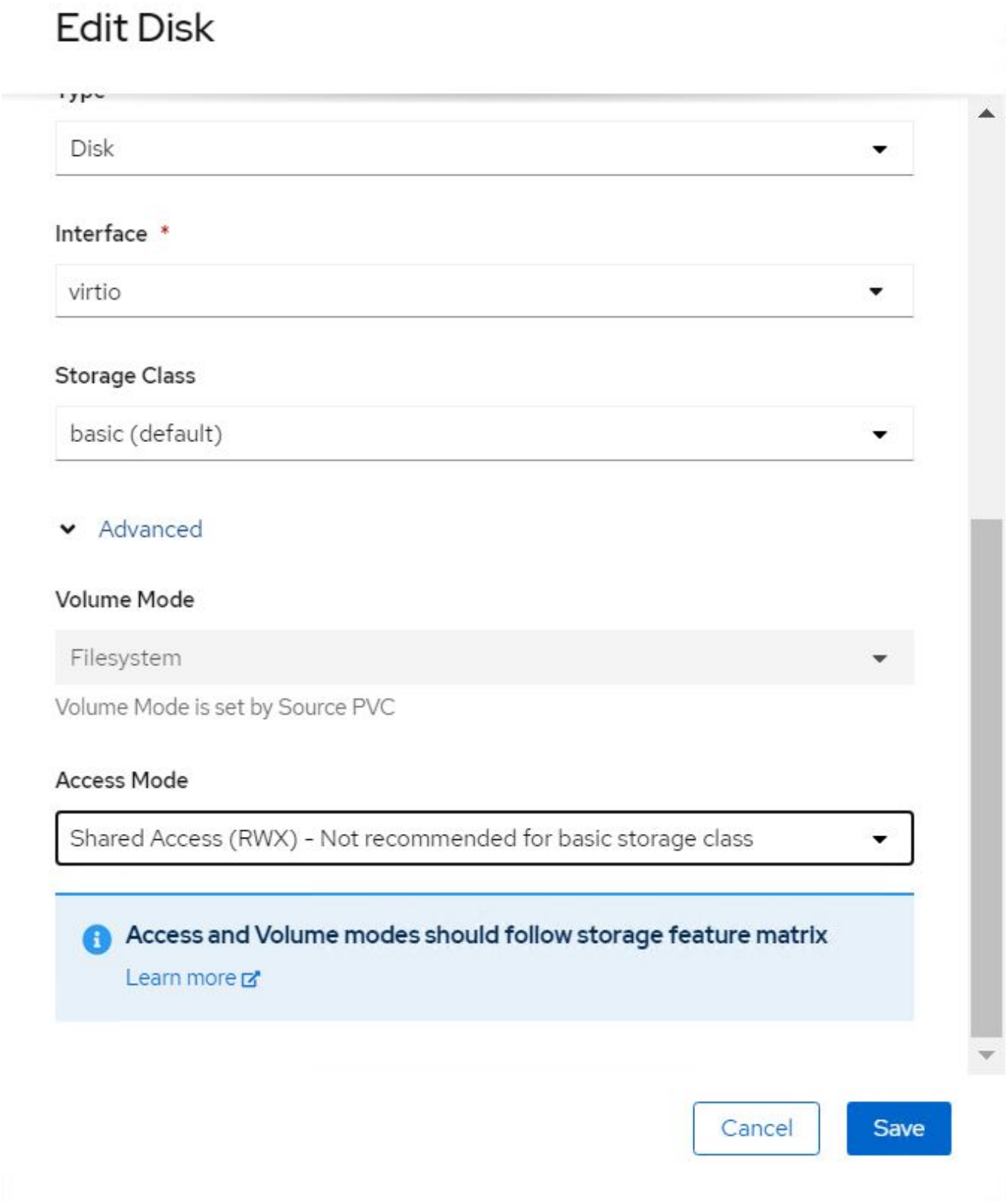
VM 实时迁移

实时迁移是指在不停机的情况下将 VM 实例从 OpenShift 集群中的一个节点迁移到另一个节点的过程。要在 OpenShift 集群中执行实时迁移， VM 必须绑定到具有共享 ReadWriteMany 访问模式的 PVC 。在启用了 NFS 协议的 NetApp ONTAP 集群上配置了 SVM 的 Astra Trident 后端支持对 PVC 的共享 ReadWriteMany 访问。因此，对于从启用了 NFS 的 SVM 中由 Trident 配置的 StorageClasses 请求具有 PVC 的 VM ，可以在不停机的情况下进行迁移。



要创建绑定到具有共享 ReadWriteMany 访问权限的 PVC 的 VM ，请执行以下操作：

1. 导航到工作负载 > 虚拟化 > 虚拟机，然后单击创建 > 使用向导。
2. 选择所需的操作系统，然后单击下一步。假设选定操作系统已配置了启动源。
3. 在 Review and Create 窗格中，选择要在其中创建 VM 的项目并提供 VM 详细信息。确保选择了要克隆的启动源，并使用为选定操作系统分配的相应 PVC 从 CD-ROM 启动。
4. 单击自定义虚拟机，然后单击存储。
5. 单击 rootdisk 旁边的省略号，并确保已选择使用 Trident 配置的 storageclass 。展开高级，然后为访问模式选择共享访问（ rwx ）。然后单击保存。



6. 单击 Review 并确认，然后单击 Create Virtual Machine 。

要手动将虚拟机迁移到 OpenShift 集群中的另一个节点，请完成以下步骤。

1. 导航到工作负载 > 虚拟化 > 虚拟机。

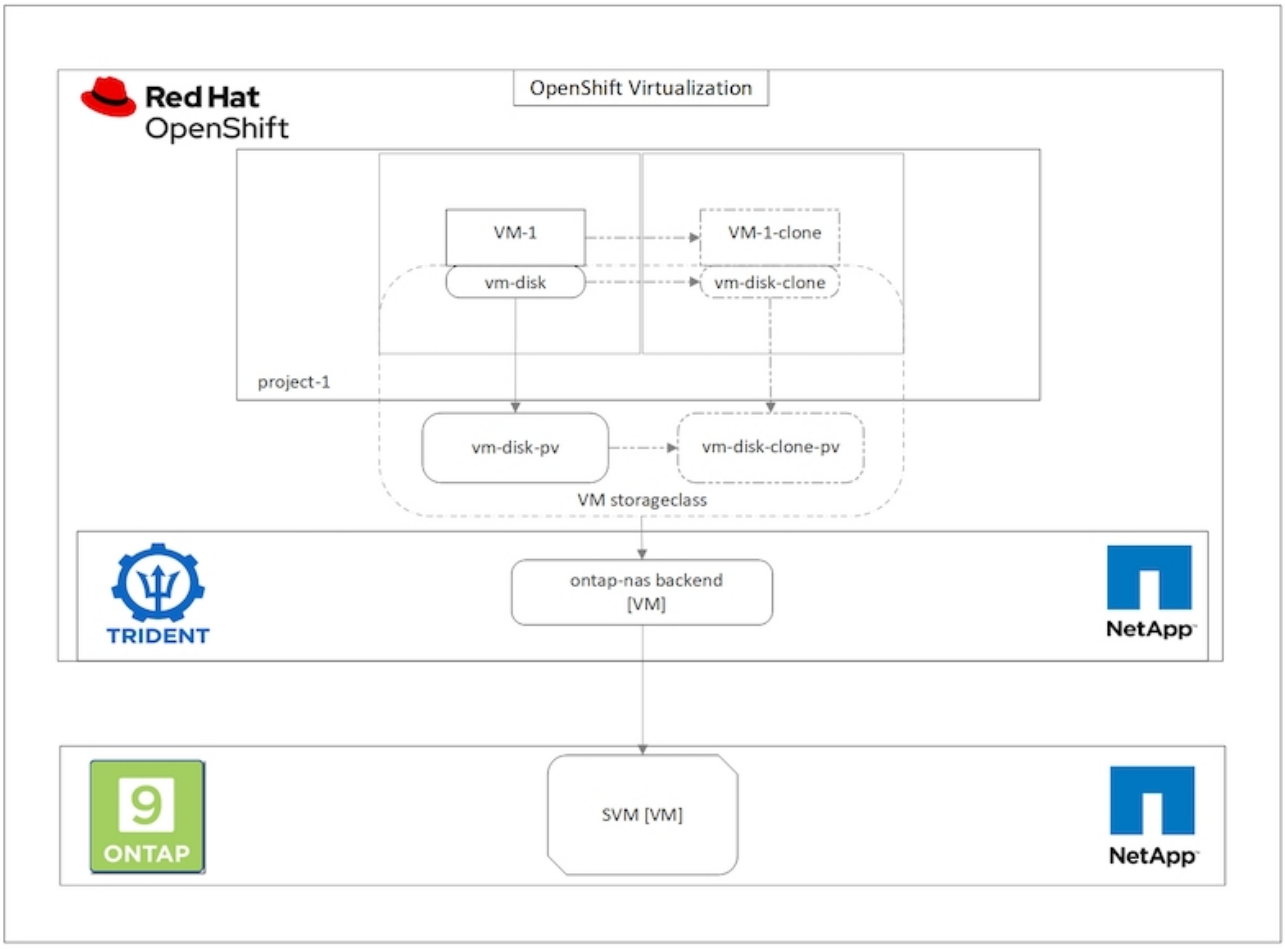
2. 对于要迁移的虚拟机，单击省略号，然后单击迁移虚拟机。
3. 当消息弹出时，单击迁移进行确认。

> 如果 evictionStrategy 设置为 LiveMigrate ，则在将原始节点置于维护模式时， OpenShift 集群中的 VM 实例会自动迁移到另一节点。

工作流：使用 **NetApp ONTAP** 实现 **Red Hat OpenShift** 虚拟化

VM 克隆

通过支持 Astra Trident 的卷 CSI 克隆功能，可以在 OpenShift 中克隆现有虚拟机。通过 CSI 卷克隆，可以使用现有 PVC 作为数据源并通过复制其 PV 来创建新的 PVC 。创建新的 PVC 后，它将作为一个单独的实体运行，并且不会与源 PVC 建立任何链接或依赖关系。



要考虑 CSI 卷克隆的某些限制：

1. 源 PVC 和目标 PVC 必须位于同一项目中。
2. 在同一存储类中支持克隆。
3. 只有当源卷和目标卷使用相同的卷模式设置时，才能执行克隆；例如，一个块卷只能克隆到另一个块卷。

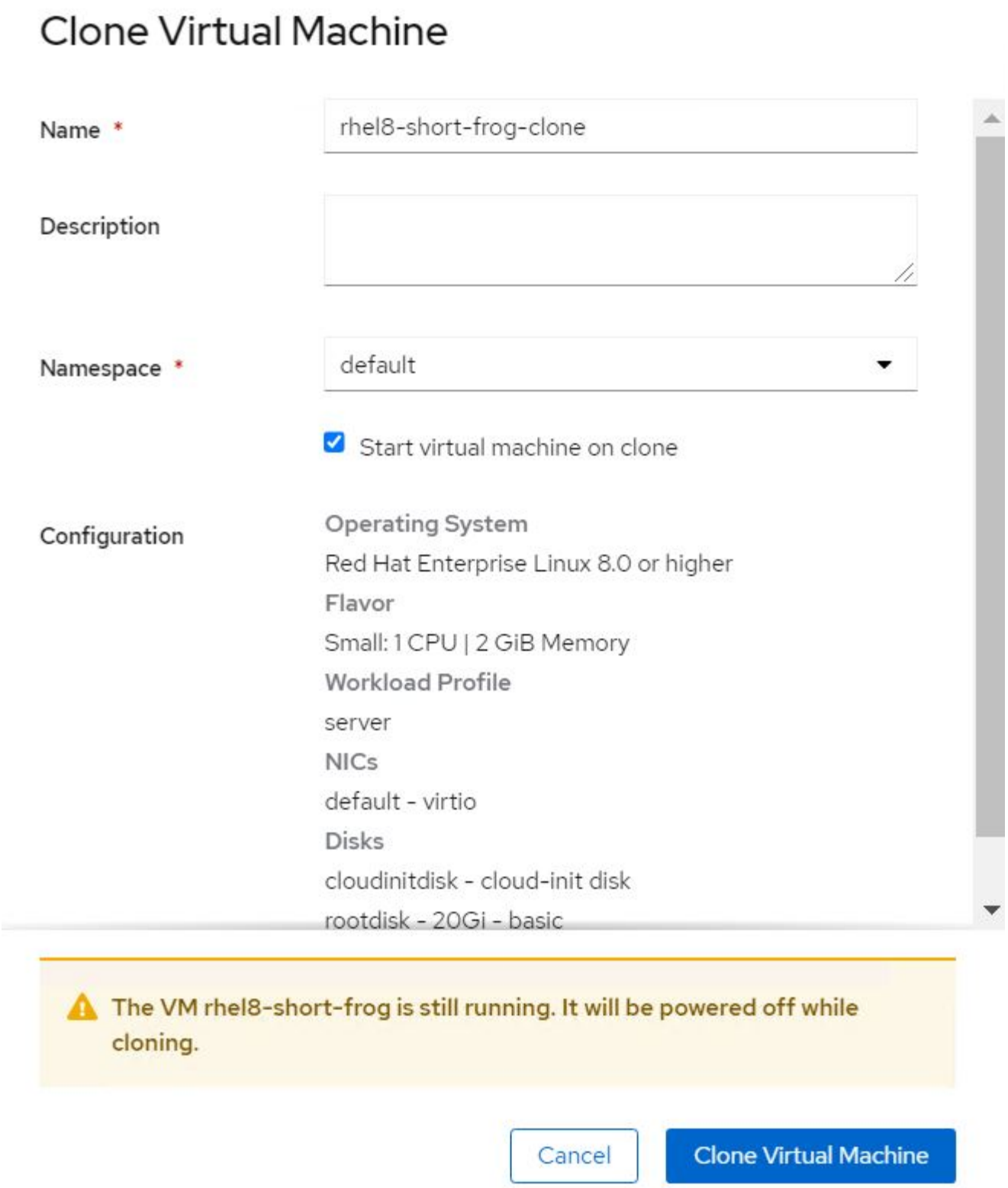
可以通过两种方式克隆 OpenShift 集群中的 VM ：

1. 关闭源 VM
2. 使源 VM 保持活动状态

关闭源 **VM**

通过关闭虚拟机克隆现有虚拟机是一项原生 OpenShift 功能，该功能在 Astra Trident 的支持下实施。要克隆虚拟机，请完成以下步骤。

1. 导航到工作负载 > 虚拟化 > 虚拟机，然后单击要克隆的虚拟机旁边的省略号。
2. 单击克隆虚拟机并提供新虚拟机的详细信息。

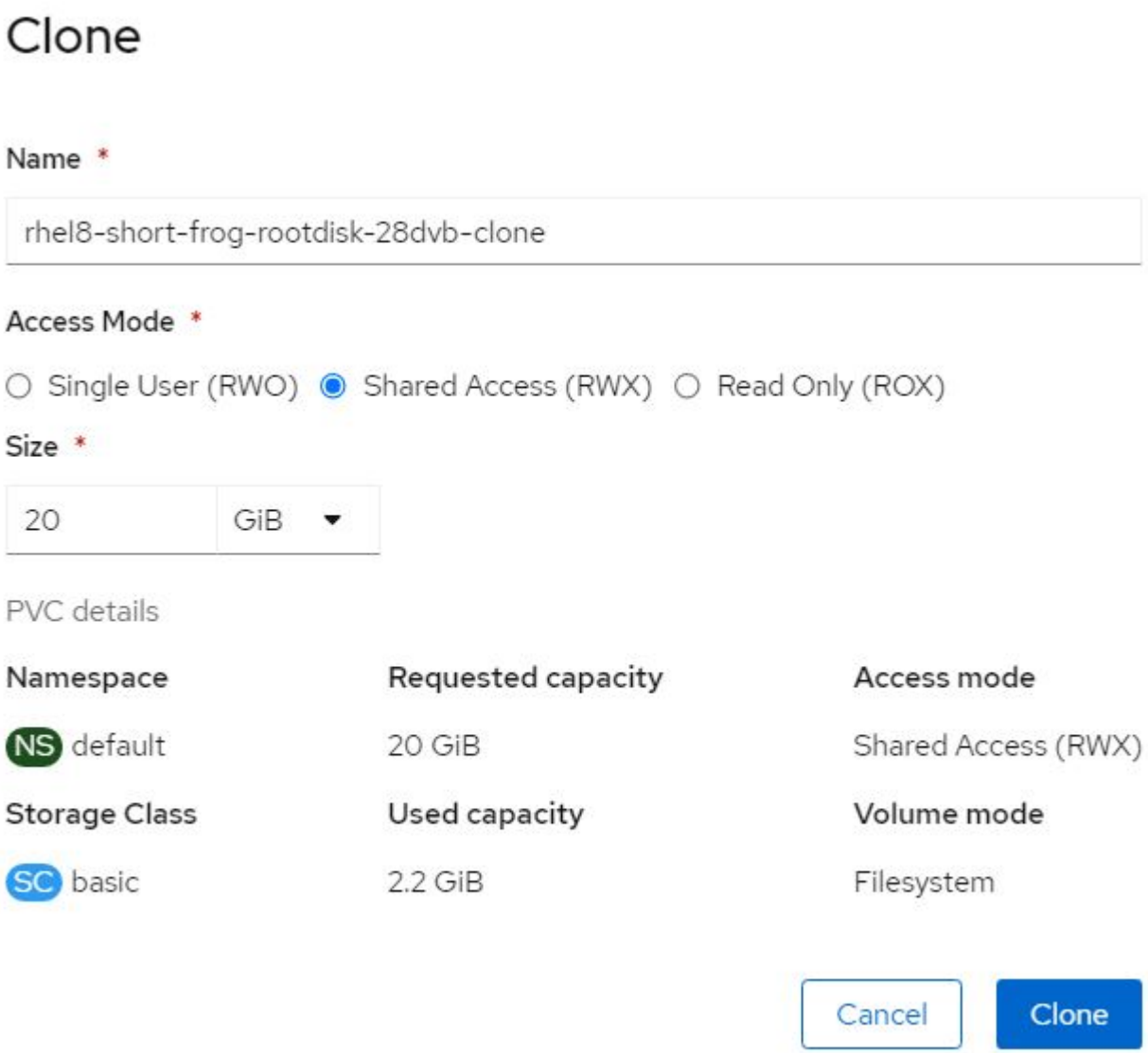


3. 单击克隆虚拟机；此操作将关闭源 VM 并启动克隆 VM 的创建。
4. 完成此步骤后，您可以访问并验证克隆的虚拟机的内容。

使源 **VM** 保持活动状态

也可以通过克隆源 VM 的现有 PVC ，然后使用克隆的 PVC 创建新 VM 来克隆现有 VM 。此方法不需要关闭源 VM 。要克隆虚拟机而不关闭它，请完成以下步骤。

1. 导航到 "Storage">"PersistentVolumeClass" ，然后单击附加到源 VM 的 PVC 旁边的省略号。
2. 单击克隆 PVC 并提供新 PVC 的详细信息。



3. 然后单击克隆。这样就会为新虚拟机创建一个 PVC 。
4. 导航到工作负载 > 虚拟化 > 虚拟机，然后单击创建 > 使用 YAML 。
5. 在规范 > 模板 > 规范 > 卷部分中，附加克隆的 PVC ，而不是容器磁盘。根据您的要求提供新虚拟机的所有其他详细信息。

```
- name: rootdisk
  persistentVolumeClaim:
    claimName: rhel8-short-frog-rootdisk-28dvb-clone
```

6. 单击创建以创建新虚拟机。
7. 成功创建 VM 后，访问并验证新 VM 是否为源 VM 的克隆。

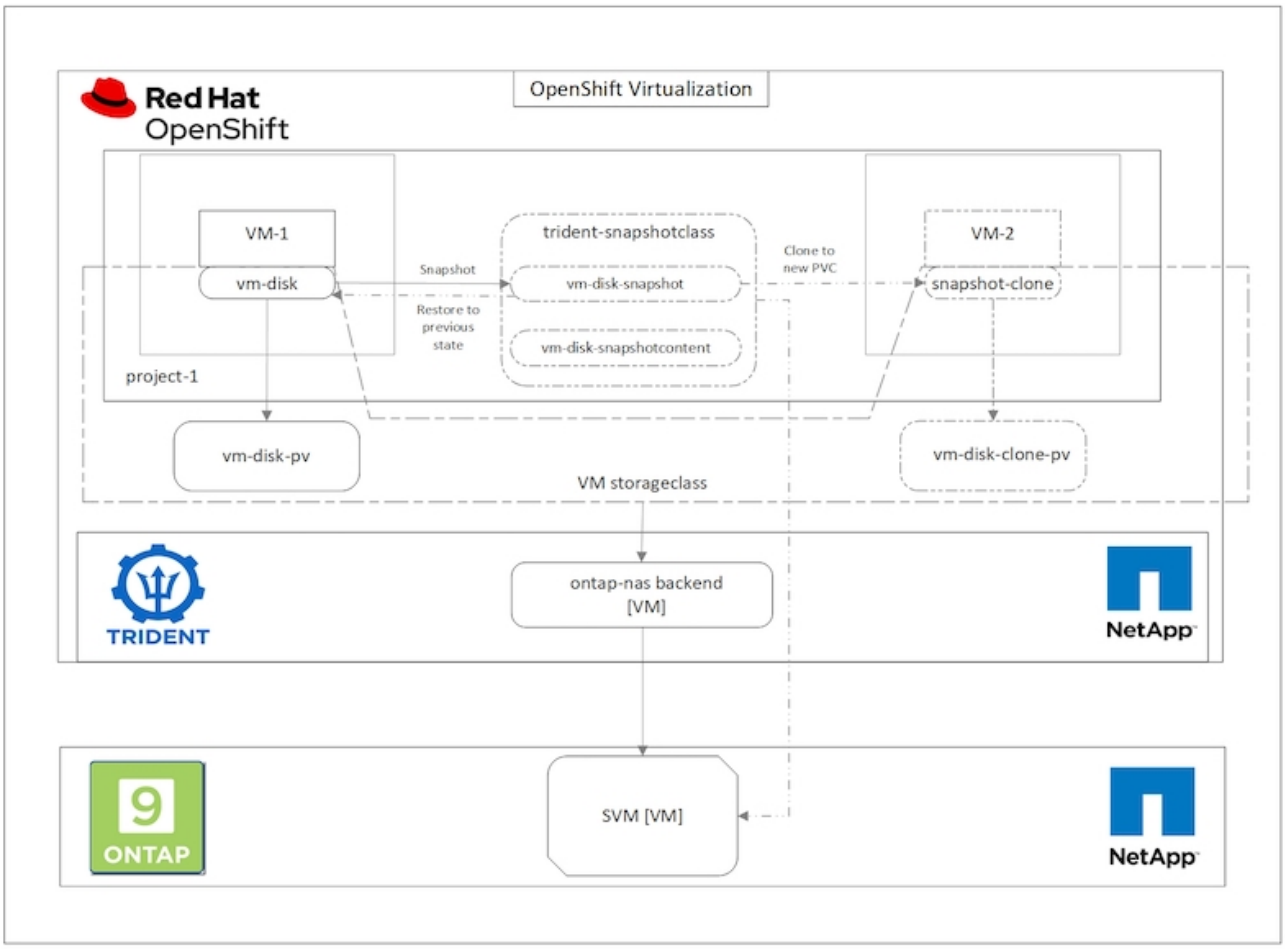
工作流：使用 **NetApp ONTAP** 实现 **Red Hat OpenShift** 虚拟化

从 **Snapshot** 创建 **VM**

借助 Astra Trident 和 Red Hat OpenShift ，用户可以在所配置的存储类上创建永久性卷的快照。通过此功能，用户可以创建卷的时间点副本，并使用该副本创建新卷或将同一卷还原到先前的状态。这样可以启用或支持从回滚到克隆再到数据还原等各种使用情形。

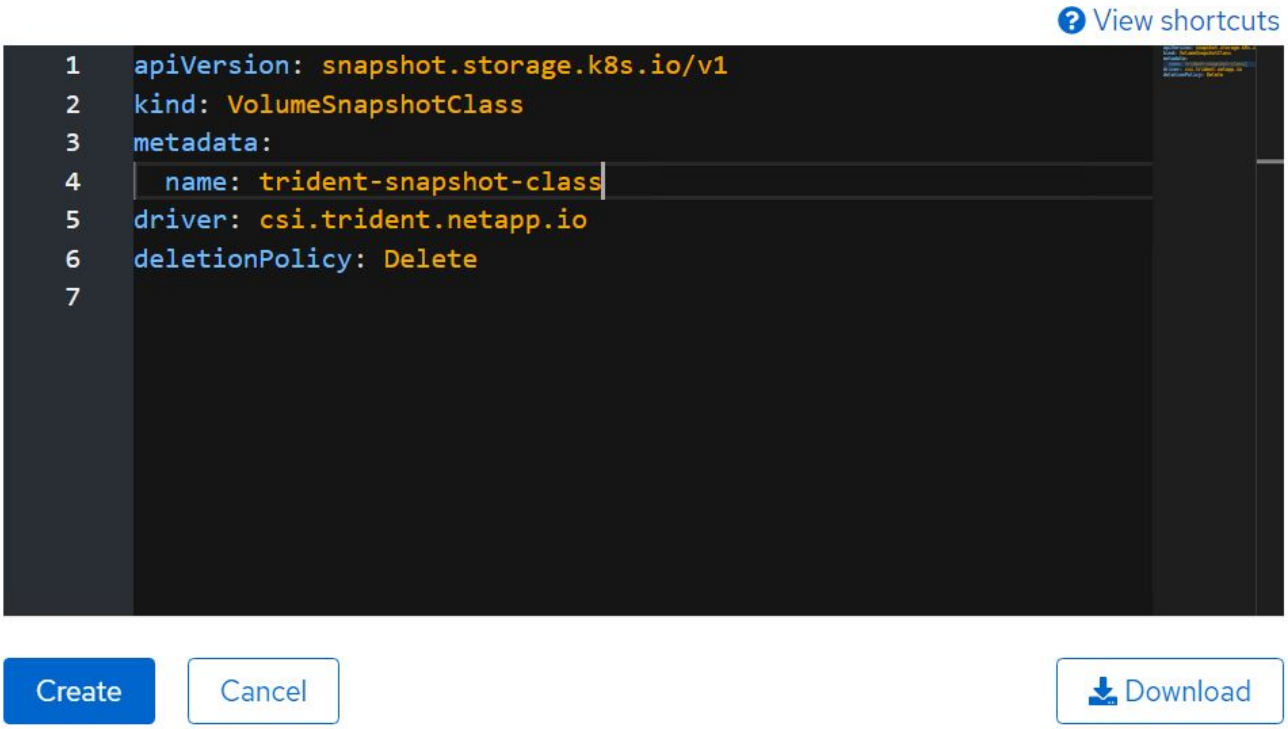
对于 OpenShift 中的 Snapshot 操作，必须定义资源 VolumeSnapshotClass ， VolumeSnapshot 和 VolumeSnapshotContent 。

- VolumeSnapshotContent 是从集群中的卷生成的实际快照。它是一种集群范围的资源，类似于用于存储的 PersistentVolume 。
- VolumeSnapshot 是指创建卷快照的请求。它类似于 PersistentVolumeClaim 。
- 管理员可以使用 VolumeSnapshotClass 为 VolumeSnapshot 指定不同的属性。通过此选项，您可以为从同一卷创建的不同快照设置不同的属性。

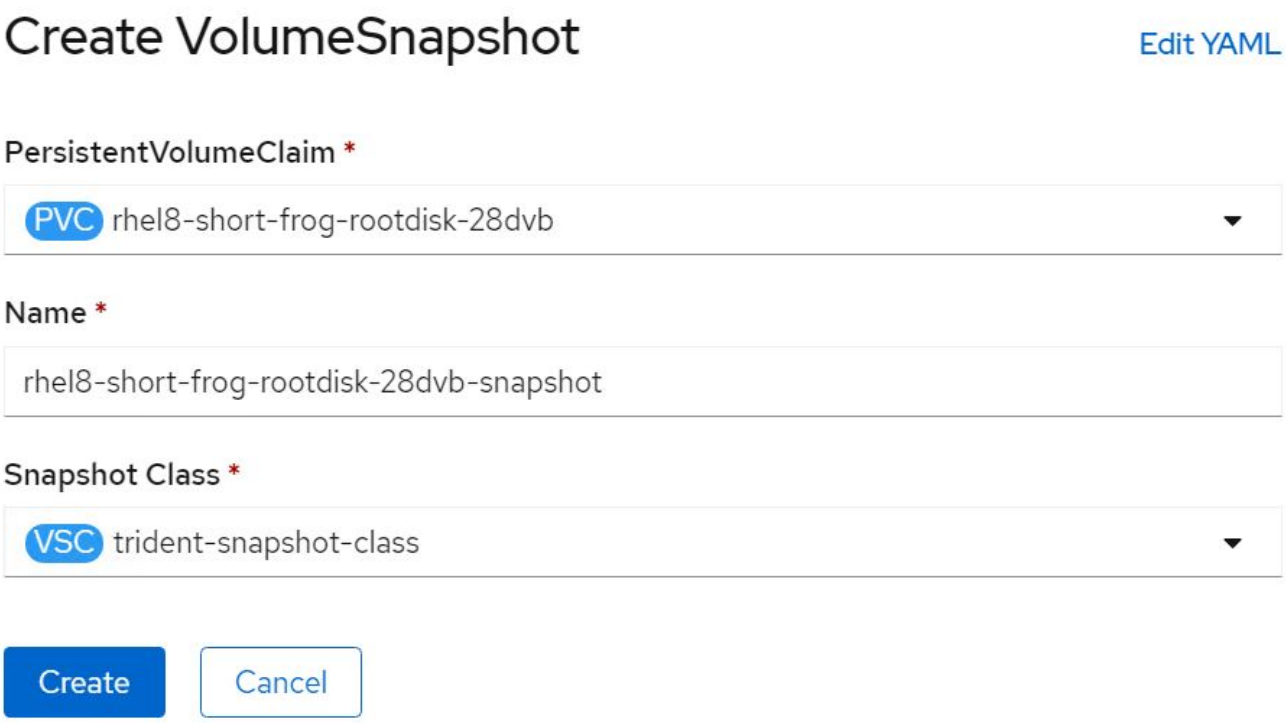


要创建虚拟机的 Snapshot ，请完成以下步骤：

1. 创建 VolumeSnapshotClass ，然后使用该类创建 VolumeSnapshot 。导航到 "Storage">"VolumeSnapshotClasss" ，然后单击 "Create VolumeSnapshotClass" 。
2. 输入 Snapshot 类的名称，输入驱动程序的 csi.trident.netapp.io ，然后单击创建。



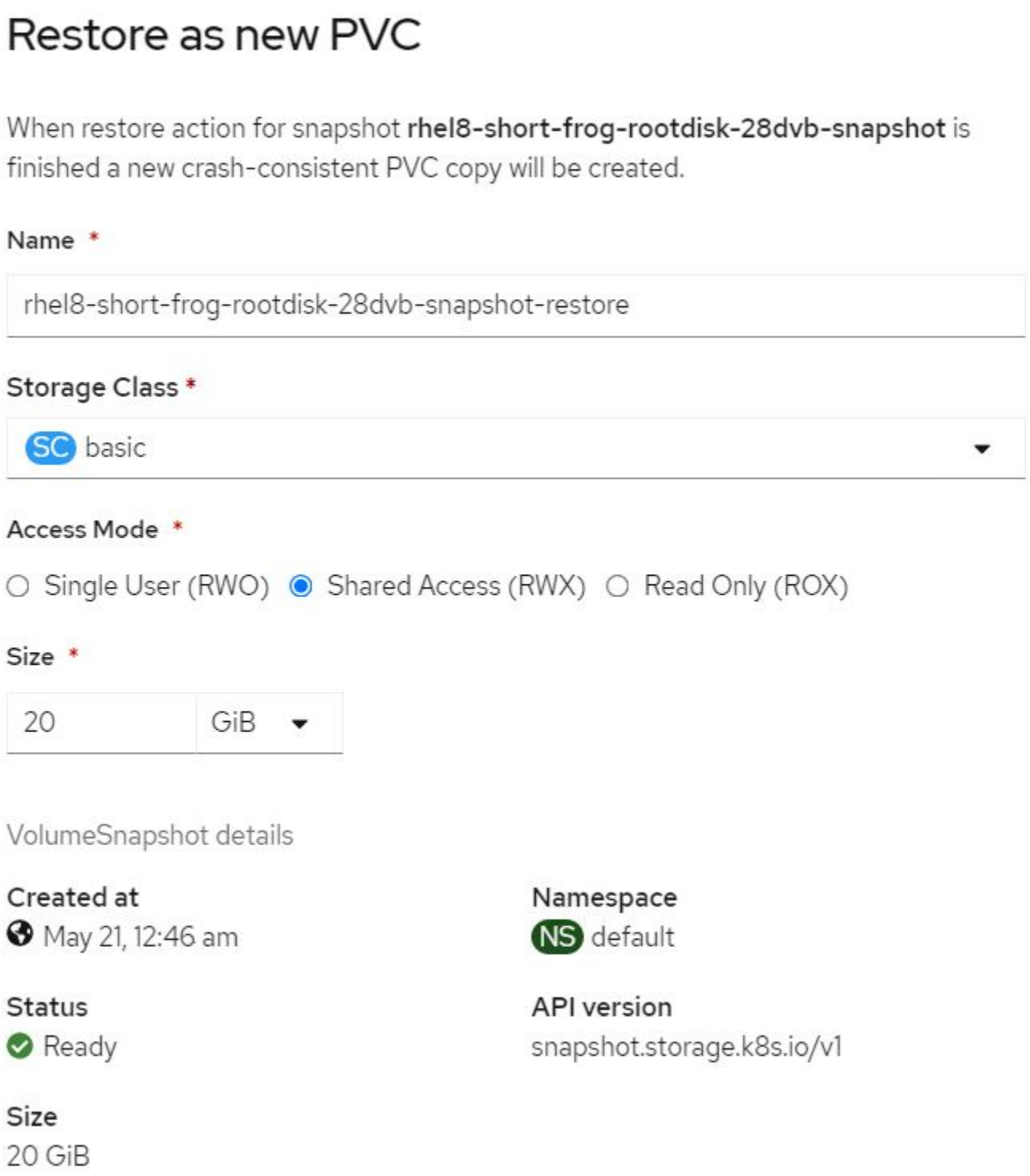
3. 确定连接到源 VM 的 PVC ，然后创建该 PVC 的 Snapshot 。导航到 `Storage > VolumeSnapshots` ，然后单击 Create VolumeSnapshots 。
4. 选择要为其创建 Snapshot 的 PVC ，输入 Snapshot 的名称或接受默认值，然后选择相应的 VolumeSnapshotClass 。然后单击创建。



5. 此时将创建 PVC 的快照。

从快照创建新虚拟机

1. 首先，将 Snapshot 还原到新的 PVC 中。导航到存储 > 卷快照，单击要还原的快照旁边的省略号，然后单击还原为新 PVC 。
2. 输入新 PVC 的详细信息，然后单击还原。这样就会创建一个新的 PVC 。



3. 接下来，使用此 PVC 创建一个新虚拟机。导航到工作负载 > 虚拟化 > 虚拟机，然后单击创建 > 使用 YAML 。

4. 在规范 > 模板 > 规范 > 卷部分中，指定从 Snapshot 创建的新 PVC ，而不是从容器磁盘创建的新 PVC 。根据您的要求提供新虚拟机的所有其他详细信息。

```
- name: rootdisk
  persistentVolumeClaim:
    claimName: rhel8-short-frog-rootdisk-28dvb-snapshot-restore
```

5. 单击创建以创建新虚拟机。
6. 成功创建虚拟机后，访问并验证新虚拟机的状态是否与创建快照时使用 PVC 创建快照的虚拟机的状态相同。

工作流：使用 **NetApp ONTAP** 实现 **Red Hat OpenShift** 虚拟化

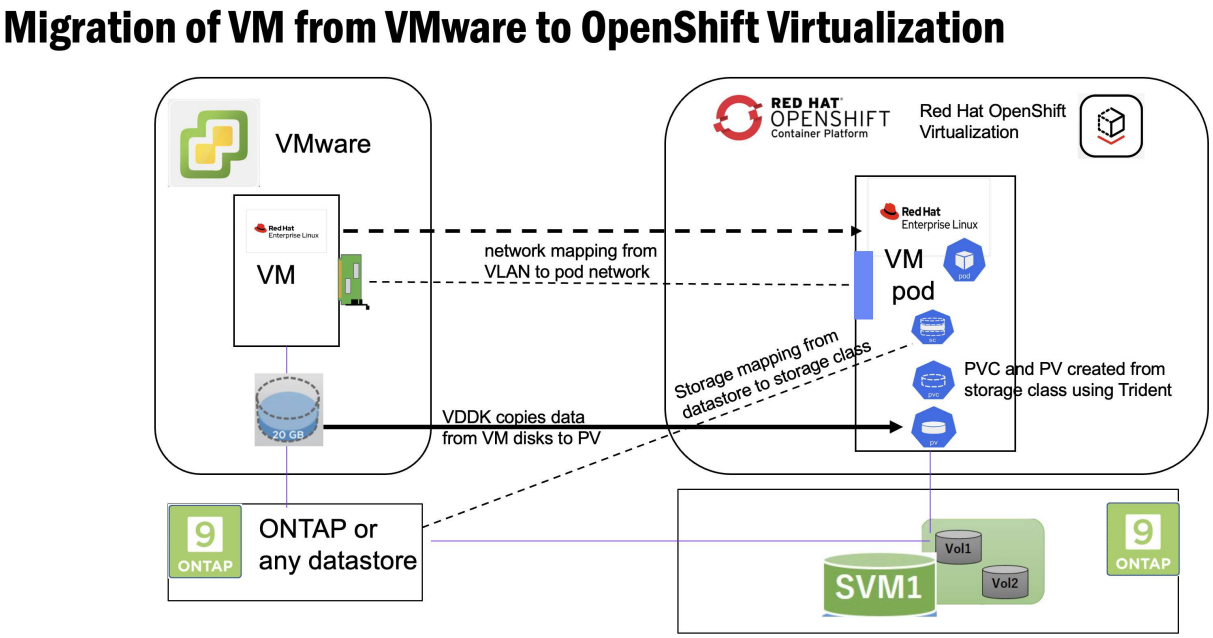
使用适用于虚拟化的迁移工具包将**VM**从**VMware**迁移到**OpenShift**虚拟化

在本节中、我们将了解如何使用虚拟化迁移工具包(Migration Toolkit for Virtualization、Mtv)将虚拟机从VMware迁移到在OpenShift容器平台上运行并使用Asta Trident与NetApp ONTAP存储集成的OpenShift虚拟化。

以下视频演示了如何使用ONTAP SAN将RHEL VM从VMware迁移到OpenShift虚拟化以实现永久性存储。

使用Red Hat VtTM通过NetApp ONTAP存储将VM迁移到OpenShift虚拟化

下图简要展示了将VM从VMware迁移到Red Hat OpenShift虚拟化的过程。



迁移示例的前提条件

在**VMware**上

- 安装了一个使用RHEL 9.3的RHEL 9 VM、并具有以下配置：
  - CPU：2、内存：20 GB、硬盘：20 GB
  - 用户凭据：root用户和管理员用户凭据
- 虚拟机准备就绪后、安装了PostgreSQL服务器。
  - PostgreSQL服务器已启动并启用、可在启动时启动

```
systemctl start postgresql.service`
systemctl enable postgresql.service
The above command ensures that the server can start in the VM in
OpenShift Virtualization after migration
```

  - 添加了2个数据库、其中添加了1个表和1行。请参见 "此处" 有关在RHEL上安装PostgreSQL服务器以及创建数据库和表条目的说明、请参见。

> 确保启动PostgreSQL服务器并启用服务以在启动时启动。

在**OpenShift**集群上

在安装此版本之前、已完成以下安装：

- OpenShift集群4.13.34
- "Astra三打23.10."
- 为iSCSI启用的集群节点上的多路径(对于ONONTAP SAN存储类)。请参见提供的YAML以创建一个守护进程集、以便在集群中的每个节点上启用iSCSI。
- 使用iSCSI的ONTAP SAN的三端和存储类。请参见为三元后端和存储类提供的YAML文件。
- "OpenShift 虚拟化"

要在OpenShift集群节点上安装iSCSI和多路径、请使用下面提供的YAML文件
为**iSCSI**准备群集节点

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  namespace: trident
  name: trident-iscsi-init
  labels:
    name: trident-iscsi-init
spec:
  selector:
    matchLabels:
      name: trident-iscsi-init
```

```
  template:
    metadata:
      labels:
        name: trident-iscsi-init
    spec:
      hostNetwork: true
      serviceAccount: trident-node-linux
      initContainers:
      - name: init-node
        command:
          - nsenter
          - --mount=/proc/1/ns/mnt
          - --
          - sh
          - -c
        args: ["$(STARTUP_SCRIPT)"]
        image: alpine:3.7
        env:
        - name: STARTUP_SCRIPT
          value: |
            #! /bin/bash
            sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils
device-mapper-multipath
            rpm -q iscsi-initiator-utils
            sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
            cat /etc/iscsi/initiatorname.iscsi
            sudo mpathconf --enable --with_multipathd y --find_multipaths
n
            sudo systemctl enable --now iscsid multipathd
            sudo systemctl enable --now iscsi
        securityContext:
          privileged: true
      hostPID: true
      containers:
      - name: wait
        image: k8s.gcr.io/pause:3.1
      hostPID: true
      hostNetwork: true
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/master
  updateStrategy:
    type: RollingUpdate
```

使用以下YAML文件创建使用ONTAP SAN存储的三元后端配置

iSCSI的三端

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: <username>
  password: <password>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: ontap-san
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <management LIF>
  backendName: ontap-san
  svm: <SVM name>
  credentials:
    name: backend-tbc-ontap-san-secret
```

使用以下YAML文件创建要使用ONTAP SAN存储的三元存储类配置
用于**iSCSI**的三级存储类

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
allowVolumeExpansion: true
```

安装Mst

现在、您可以安装适用于虚拟化的迁移工具包(Migration Toolkit for Virtualization、简称为迁移工具包)。请参阅提供的说明 "此处" 有关安装的帮助。

虚拟化迁移工具包(Migration Toolkit for Virtualization、Tmb)用户界面集成到OpenShift Web控制台中。

您可以参考 "此处" 开始使用用户界面执行各种任务。

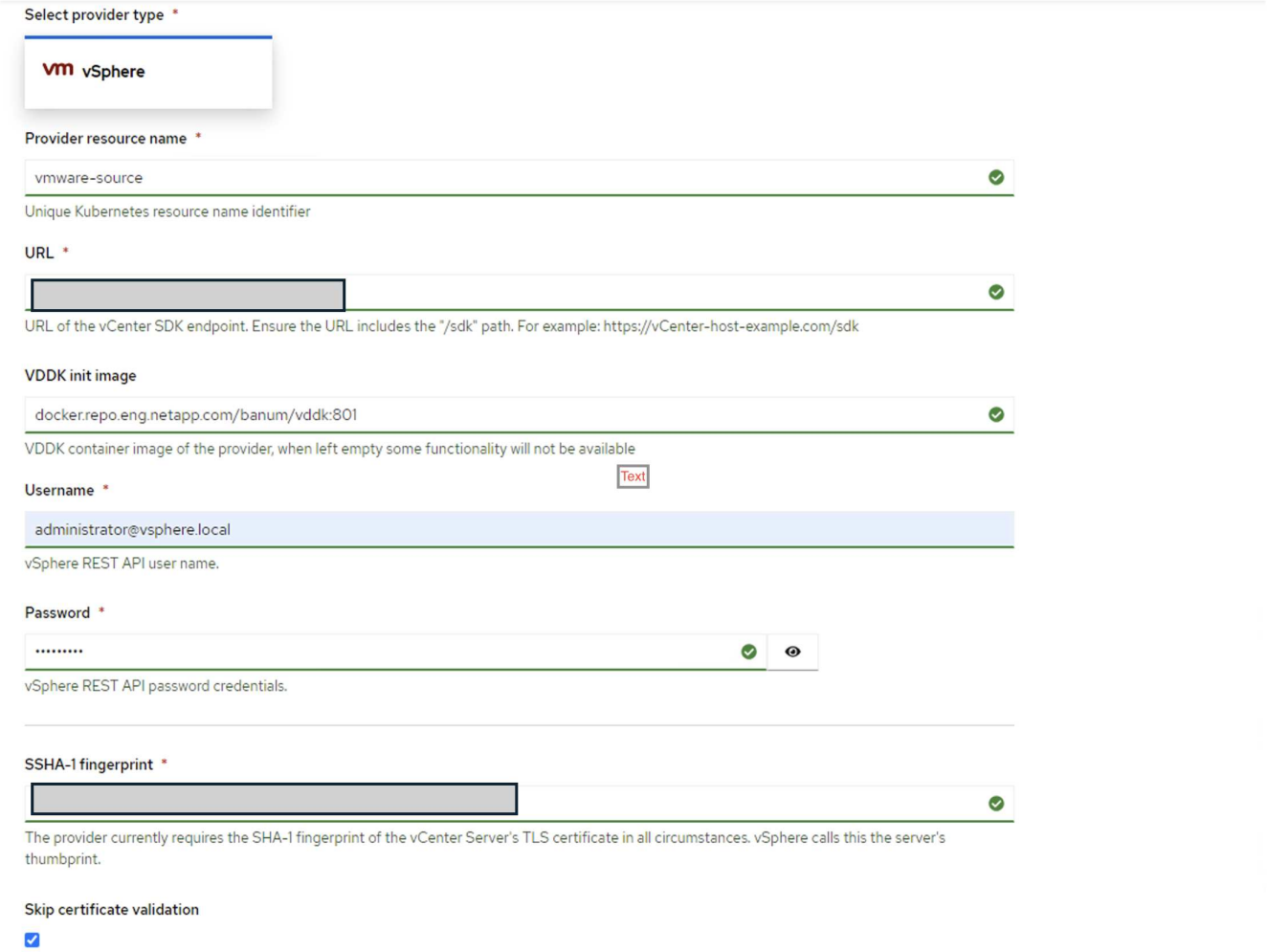
创建源提供程序

要将RHEL VM从VMware迁移到OpenShift虚拟化、您需要先为VMware创建源提供程序。请参阅说明 "此处" 以创建源提供程序。

要创建VMware源提供程序、您需要满足以下条件：

- vCenter URL
- vCenter凭据
- vCenter Server指纹
- 存储库中的VDDK映像

创建源提供程序的示例：



> 虚拟化迁移工具包(Migration Toolkit for Virtualization、Mv）使用VMware虚拟磁盘开发工具包(Virtual Disk Development Kit、VDDK) SDK来加快从VMware vSphere传输虚拟磁盘的速度。因此、强烈建议创建VDDK映像、尽管这是可选的。
> 要使用此功能、请下载VMware虚拟磁盘开发工具包(VDDK)、构建VDDK映像、然后将VDDK映像推送到映像注册表。

按照提供的说明进行操作 "此处" 创建VDDK映像并将其推送到可从OpenShift集群访问的注册表。

创建目标提供程序

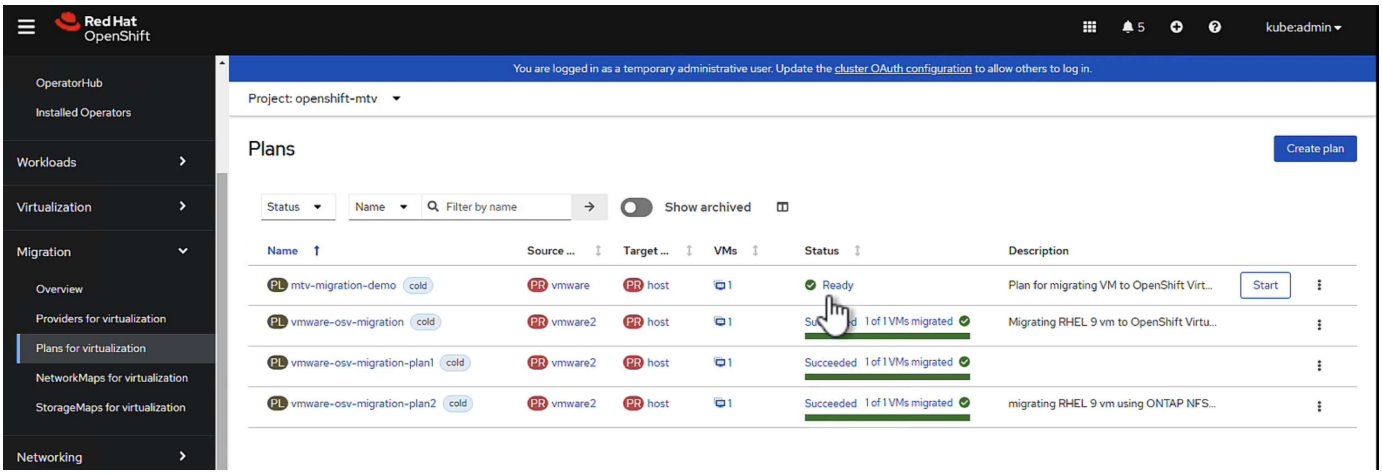
由于OpenShift虚拟化提供程序是源提供程序、因此会自动添加主机集群。

创建迁移计划

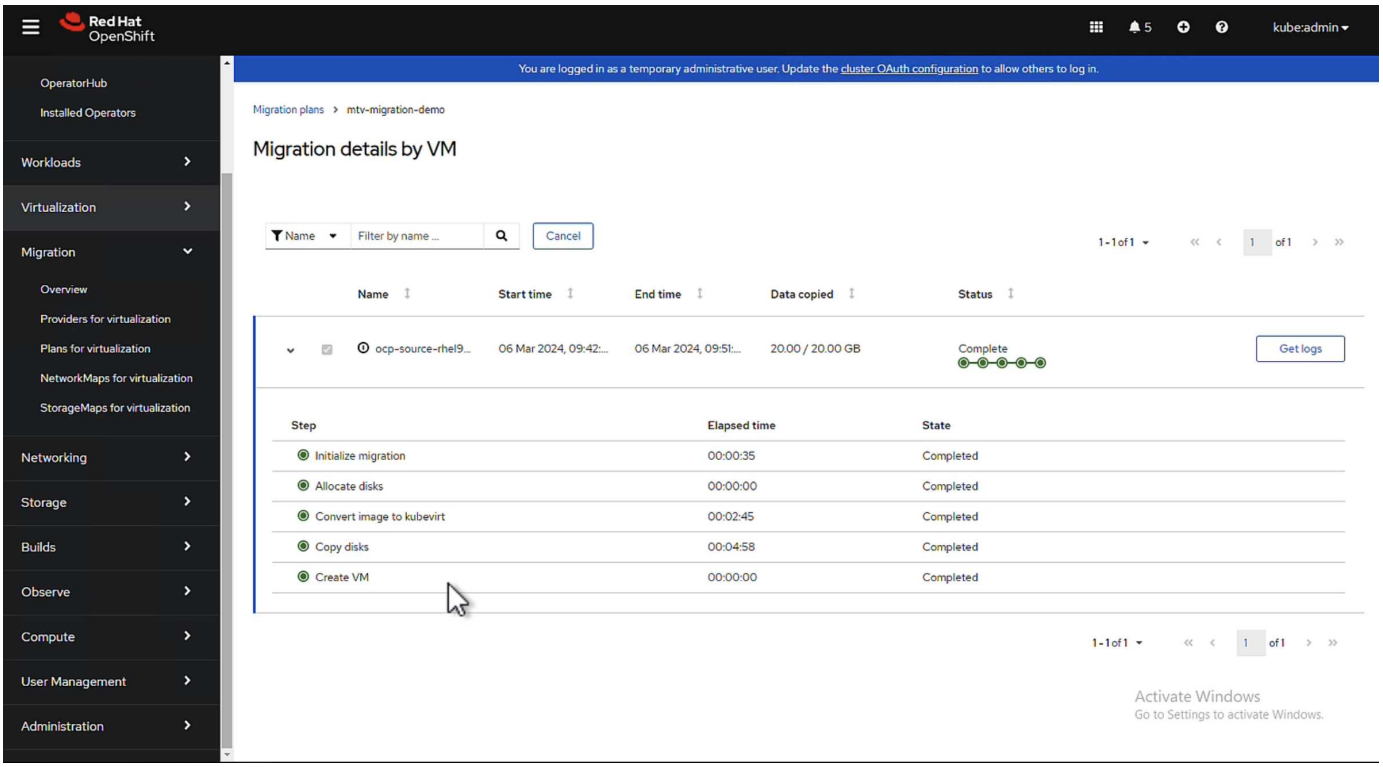
按照提供的说明进行操作 "此处" 以创建迁移计划。

创建计划时，如果尚未创建，则需要创建以下内容：

- 用于将源网络映射到目标网络的网络映射。
- 用于将源数据存储库映射到目标存储类的存储映射。为此、您可以选择ONTAP SAN存储类。
  创建迁移计划后，该计划的状态应显示*Ready*，现在您应该能够*Start*该计划。



单击*Start*将运行一系列步骤来完成虚拟机的迁移。



完成所有步骤后，您可以通过单击左侧导航菜单中“Virtualization”(虚拟化)下的*virtual Machines*来查看迁移的VM。
其中提供了访问虚拟机的说明 "此处"。

您可以登录到虚拟机并验证pos正在 使用的数据库的内容。此表中的数据库、表和条目应与在源VM上创建的相同。

借助 NetApp 在 Red Hat OpenShift 上为 Kubernetes 提供高级集群管理

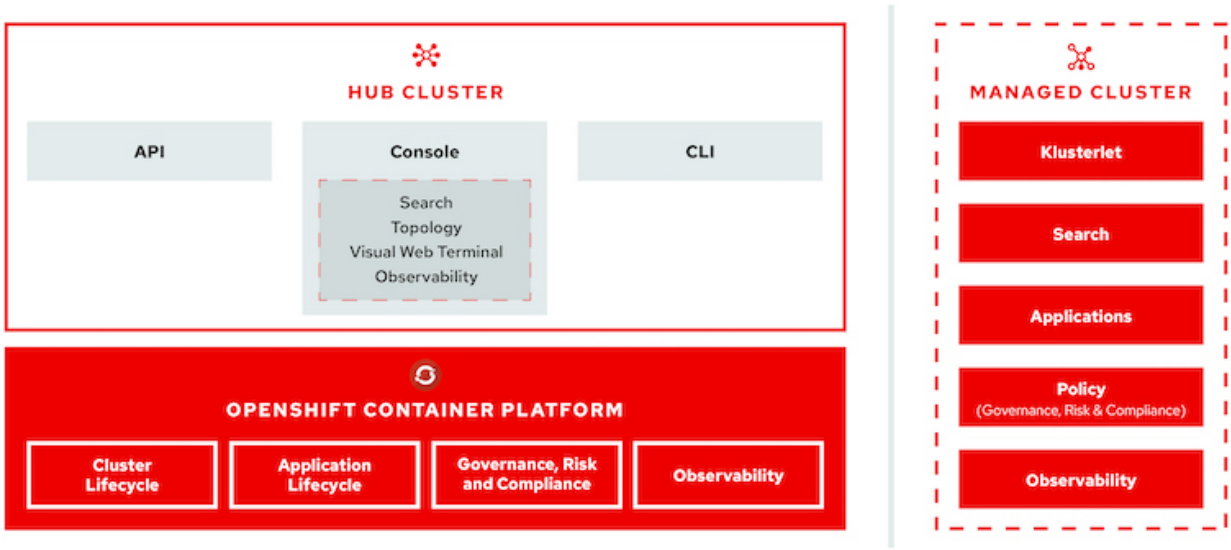
适用于 Kubernetes 的高级集群管理：采用 NetApp 的 Red Hat OpenShift

随着容器化应用程序从开发过渡到生产，许多组织需要使用多个 Red Hat OpenShift 集群来支持该应用程序的测试和部署。同时，企业通常会在 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，因此 OpenShift 管理员必须面对在跨多个内部数据中心和公有云的一系列环境中管理和维护多个集群这一额外挑战。为了应对这些挑战， Red Hat 推出了适用于 Kubernetes 的高级集群管理。

使用 Red Hat Advanced Cluster Management for Kubernetes 可以执行以下任务：

1. 跨数据中心和公有云创建，导入和管理多个集群。
2. 从一个控制台在多个集群上部署和管理应用程序或工作负载。
3. 监控和分析不同集群资源的运行状况和状态
4. 监控并强制实施多个集群的安全合规性。

Red Hat Advanced Cluster Management for Kubernetes 作为 Red Hat OpenShift 集群的附加组件进行安装，并使用此集群作为其所有操作的中央控制器。此集群称为集线器集群，它会为用户提供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 集群均由集线器集群管理，称为受管集群。它会在受管集群上安装一个名为 Kluterlet 的代理，将其连接到中心集群，并处理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请求。



有关详细信息，请参见文档 "此处"。

部署

部署适用于 **Kubernetes** 的高级集群管理

前提条件

1. 用于集线器集群的 Red Hat OpenShift 集群（版本 4.5 以上）
2. 适用于受管集群的 Red Hat OpenShift 集群（高于 4.5.3 版）
3. 对 Red Hat OpenShift 集群的集群管理员访问
4. 适用于 Kubernetes 的 Red Hat 高级集群管理订阅

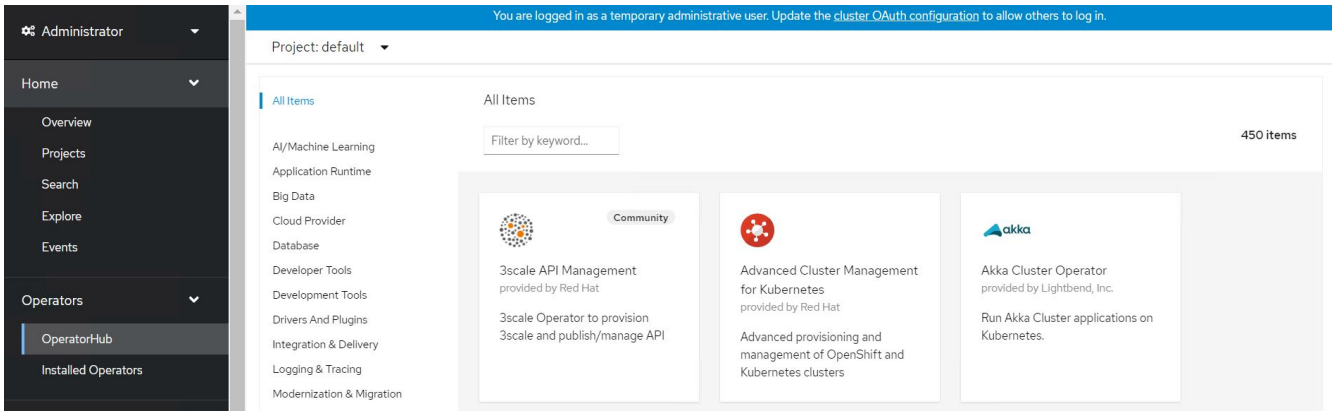
高级集群管理是 OpenShift 集群的一个附加功能，因此，根据在集线器和受管集群中使用的功能，硬件资源具有某些要求和限制。在对集群进行规模估算时，您需要考虑这些问题。请参见文档 "此处" 有关详细信息：

或者，如果集线器集群具有专用节点来托管基础架构组件，并且您希望仅在这些节点上安装高级集群管理资源，则需要相应地为这些节点添加容错和选择器。有关详细信息，请参见文档 "此处"。

部署适用于 **Kubernetes** 的高级集群管理

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

1. 选择一个 OpenShift 集群作为中心集群，并使用 cluster-admin 权限登录到该集群。
2. 导航到 Operators > Operators Hub ，然后搜索适用于 Kubernetes 的高级集群管理。



3. 选择适用于 Kubernetes 的高级集群管理，然后单击安装。

4. 在 Install Operator 屏幕上，提供必要的详细信息（ NetApp 建议保留默认参数），然后单击 Install 。

OperatorHub > Operator Installation

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Update channel *

- ○ release-2.0
- ○ release-2.1
- ◉ release-2.2

Installation mode *

- ○ All namespaces on the cluster (default)
  This mode is not supported by this Operator
- ◉ A specific namespace on the cluster
  Operator will be available in a single Namespace only.

Installed Namespace *

- ◉ Operator recommended Namespace: PR open-cluster-management

Namespace creation
Namespace **open-cluster-management** does not exist and will be created.

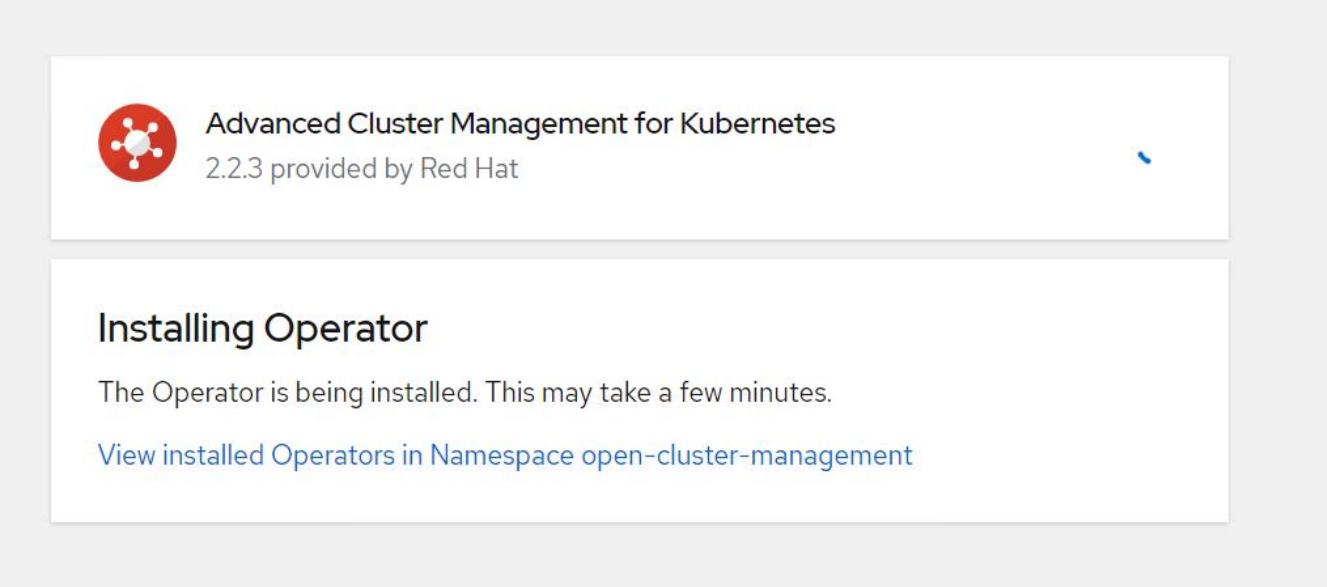
- ○ Select a Namespace

Approval strategy *

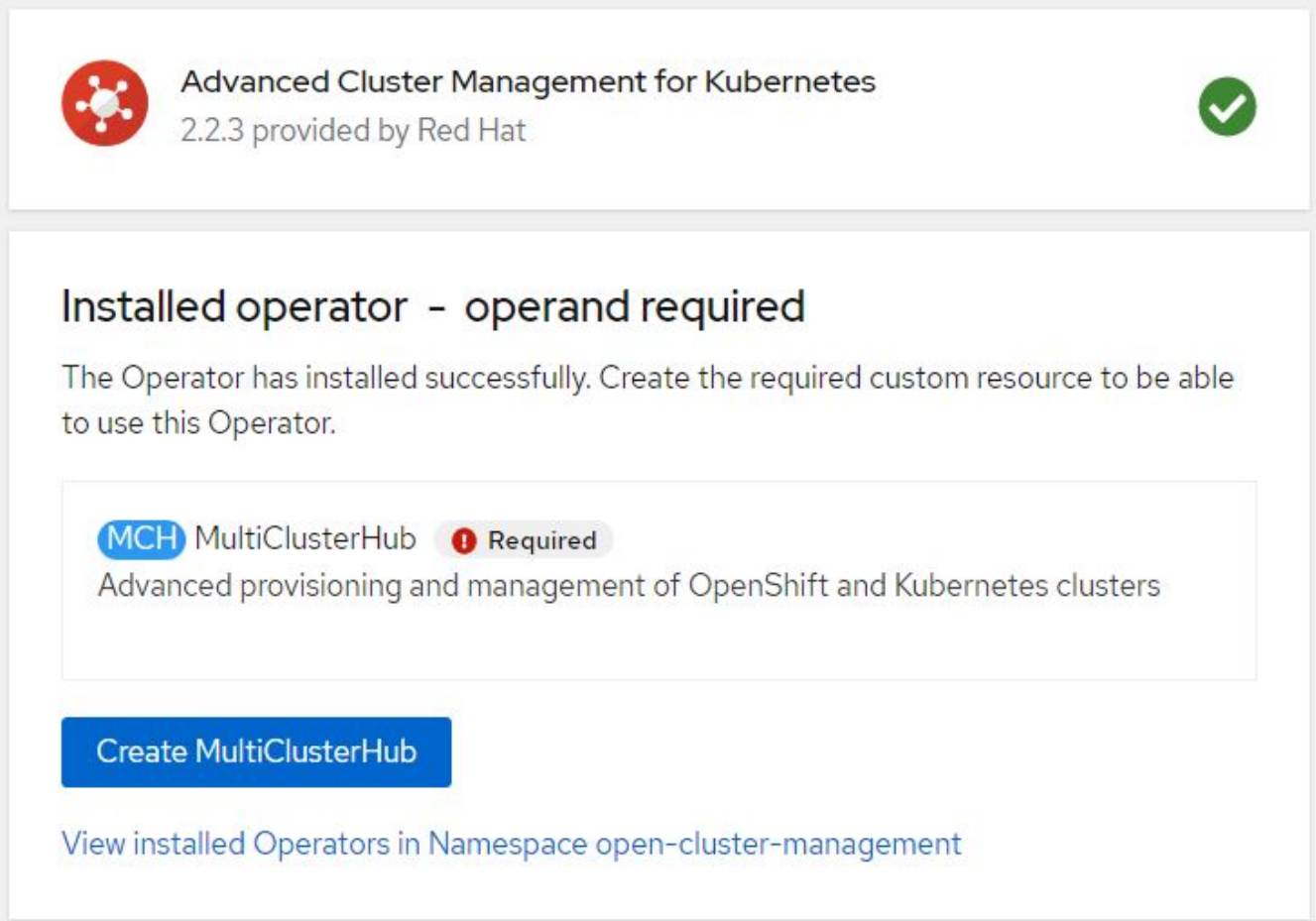
- ◉ Automatic
- ○ Manual

Install Cancel

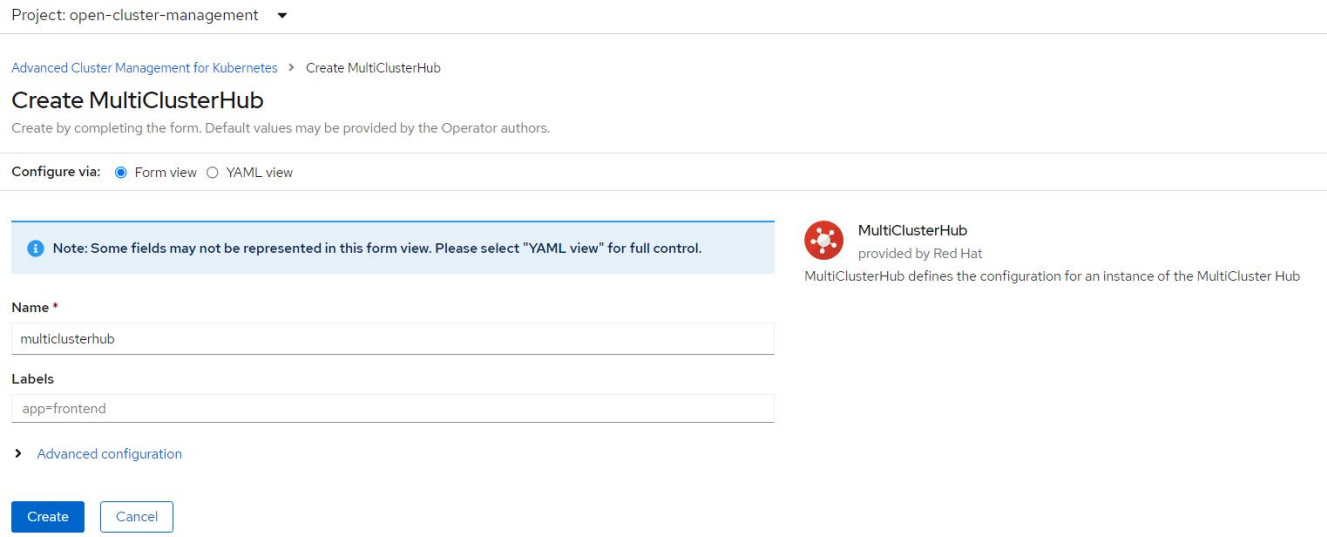
5. 等待操作员安装完成。



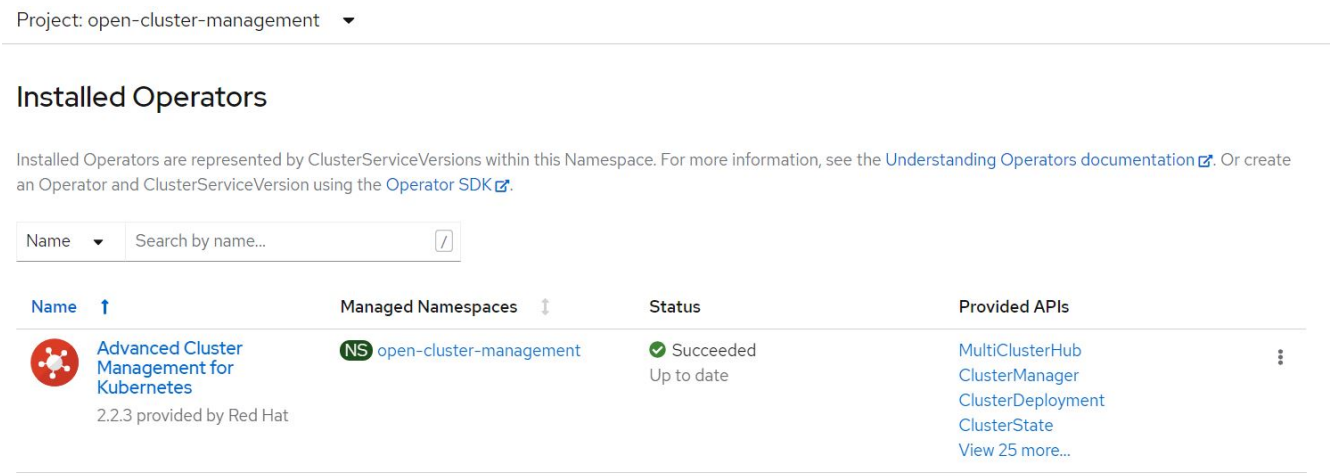
6. 安装操作员后，单击创建多集群中心。



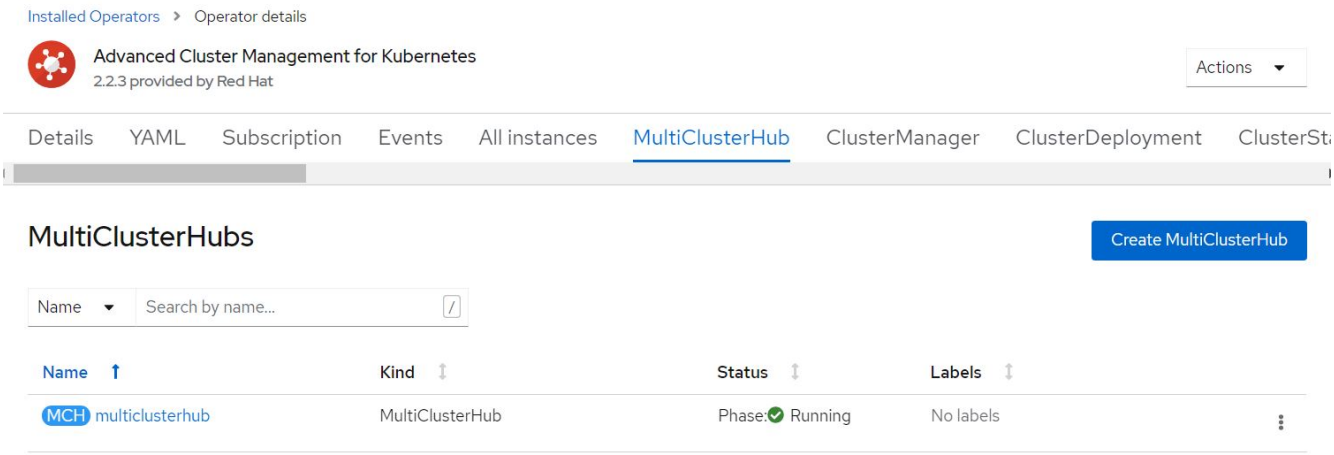
7. 在 "Create MultiClusterHub " 屏幕上，在提供详细信息后单击 "Create 。此操作将启动多集群集线器的安装。



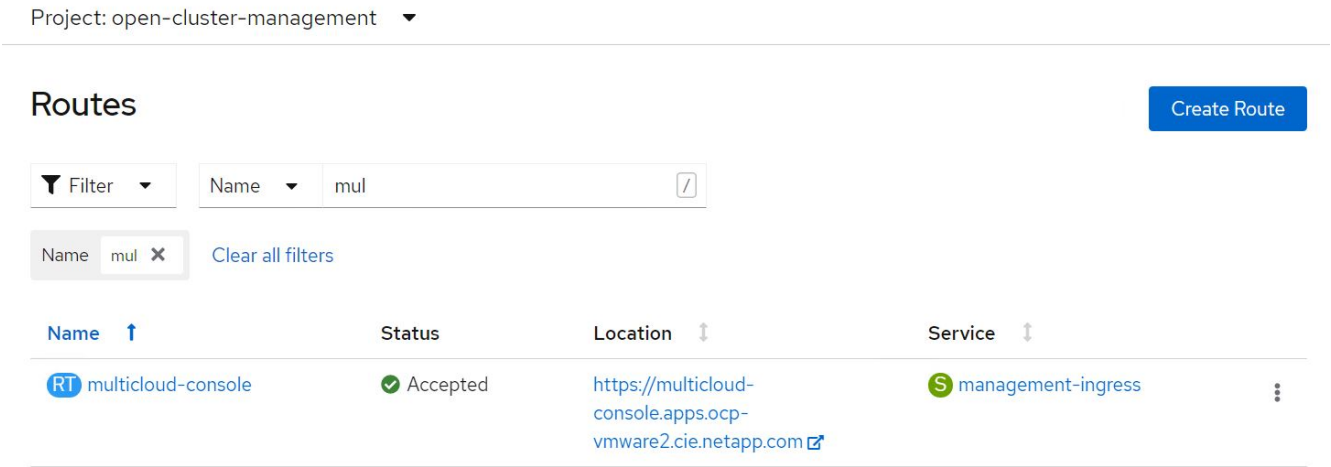
8. 在打开集群管理命名空间中的所有 Pod 均移至运行状态且操作员移至成功状态后，将安装适用于 Kubernetes 的高级集群管理。



9. 完成集线器安装需要一些时间，完成后，多集群集线器将变为运行状态。



10. 它会在开放式集群管理命名空间中创建路由。连接到路由中的 URL 以访问高级集群管理控制台。



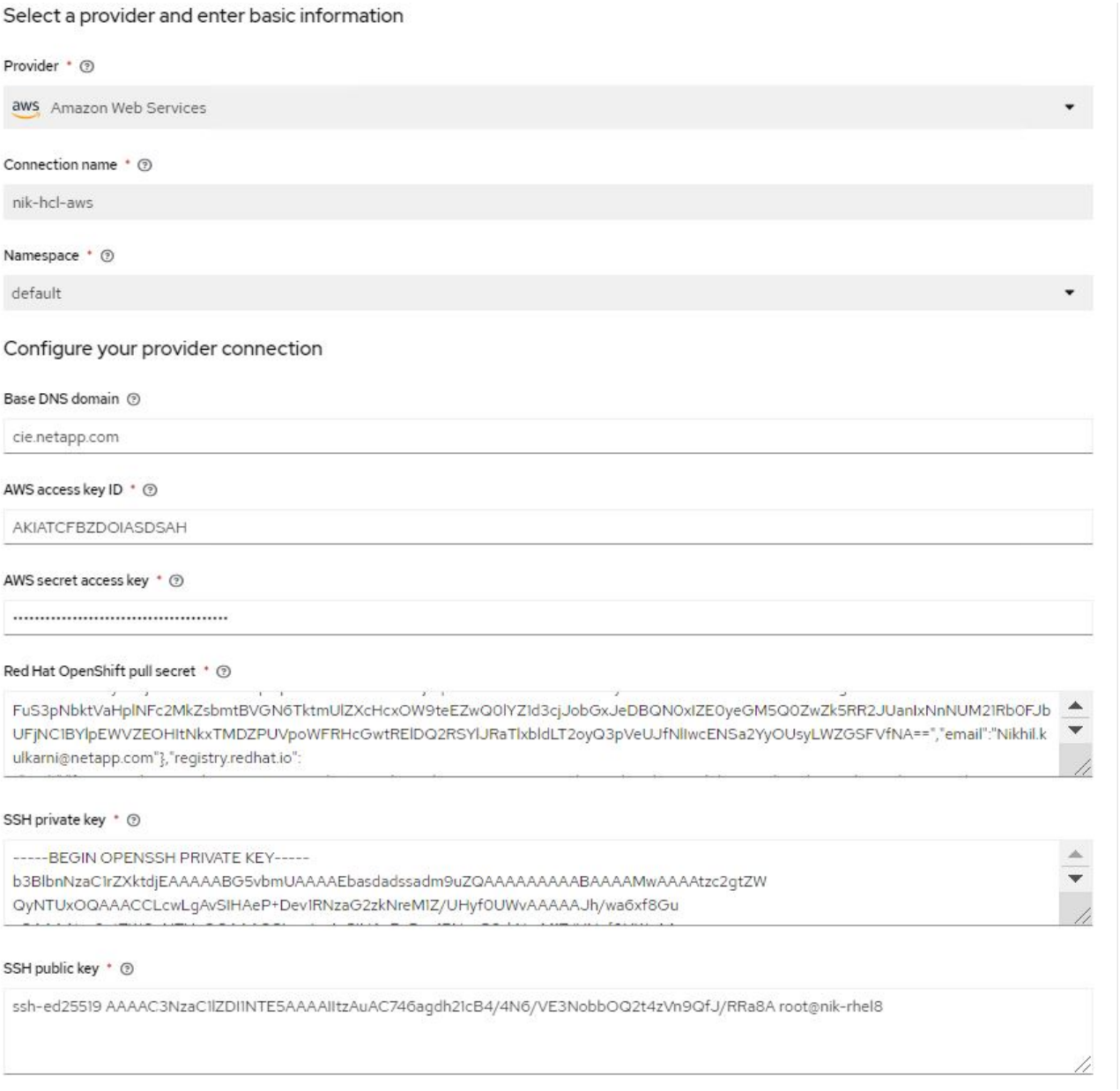
功能

功能：借助 **NetApp** 在 **Red Hat OpenShift** 上为 **Kubernetes** 提供高级集群管理

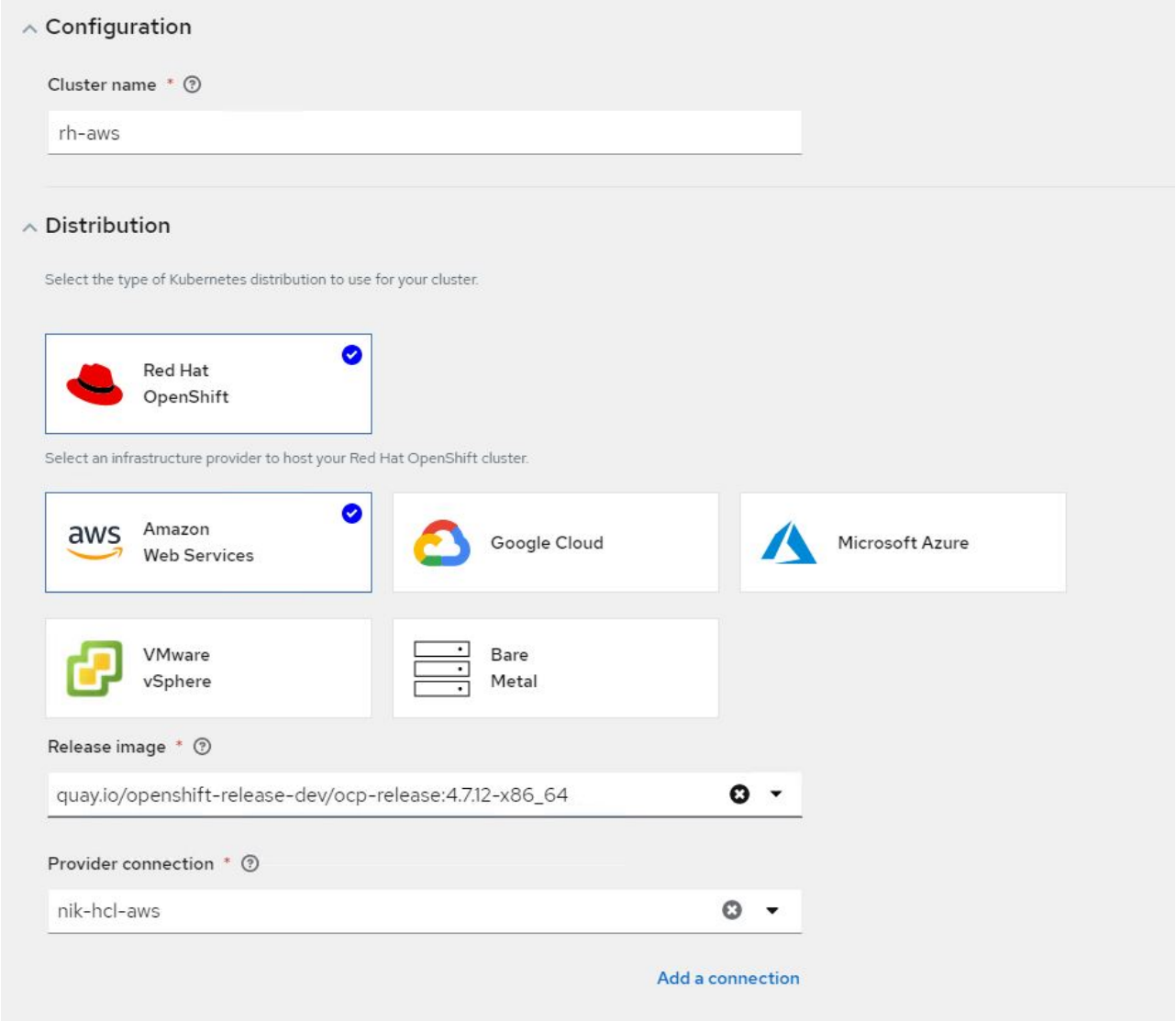
集群生命周期管理

要管理不同的 OpenShift 集群，您可以创建这些集群或将其导入到高级集群管理中。

1. 首先导航到 " 自动化基础架构 ">" 集群 " 。
2. 要创建新的 OpenShift 集群，请完成以下步骤：
   a. 创建提供程序连接：导航到 " 提供程序连接 " 并单击 " 添加连接 " ，提供与选定提供程序类型对应的所有详细信息，然后单击 " 添加 " 。



   b. 要创建新集群，请导航到集群，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。



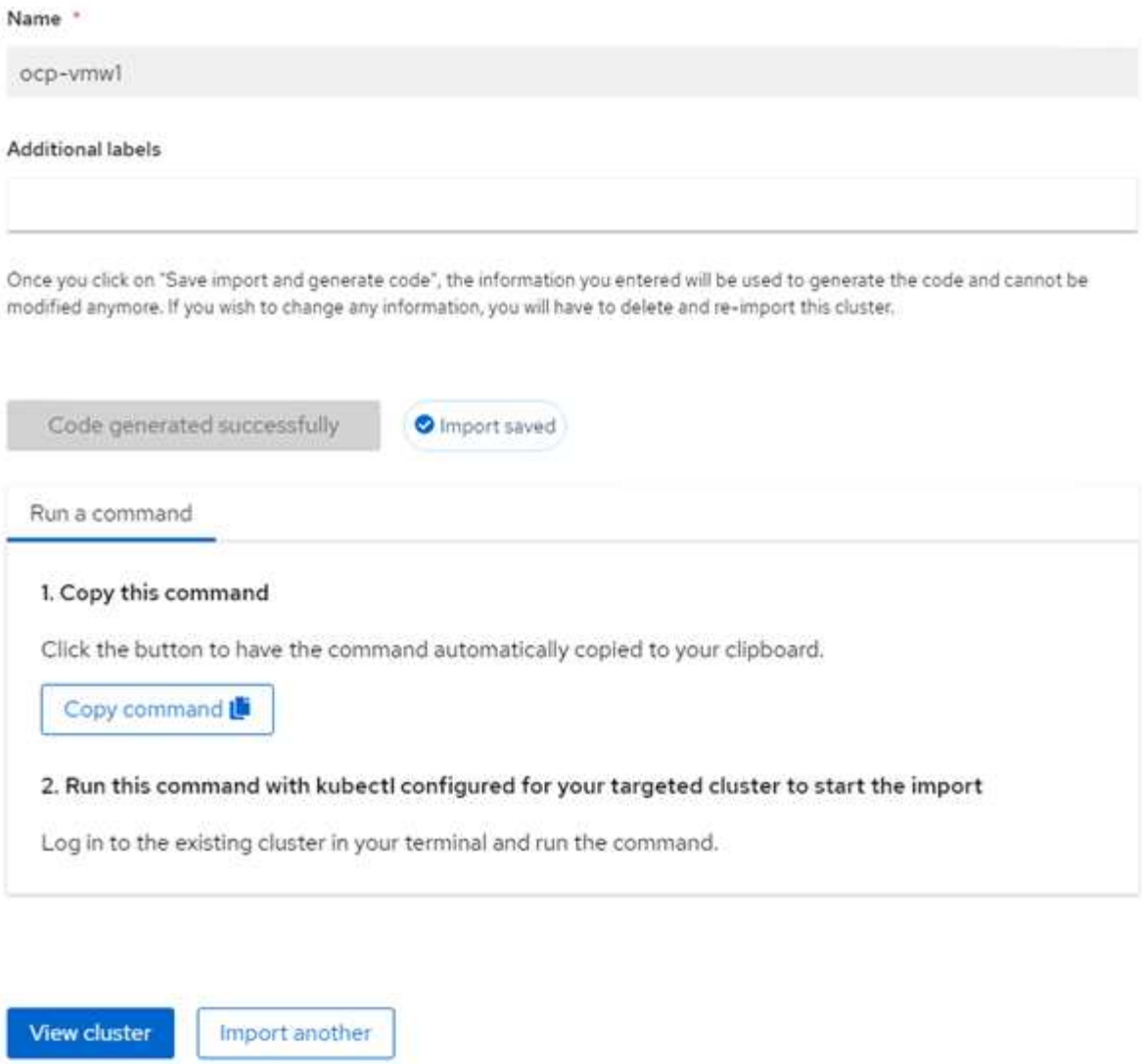
c. 创建集群后，该集群将显示在集群列表中，状态为 Ready 。

3. 要导入现有集群，请完成以下步骤：

   a. 导航到集群，然后单击添加集群 > 导入现有集群。

   b. 输入集群的名称，然后单击保存导入并生成代码。此时将显示一个用于添加现有集群的命令。

   c. 单击 Copy Command ，然后对要添加到集线器集群的集群运行命令。此操作将在集群上启动所需代理的安装，完成此过程后，集群将显示在集群列表中，并显示状态为 Ready 。



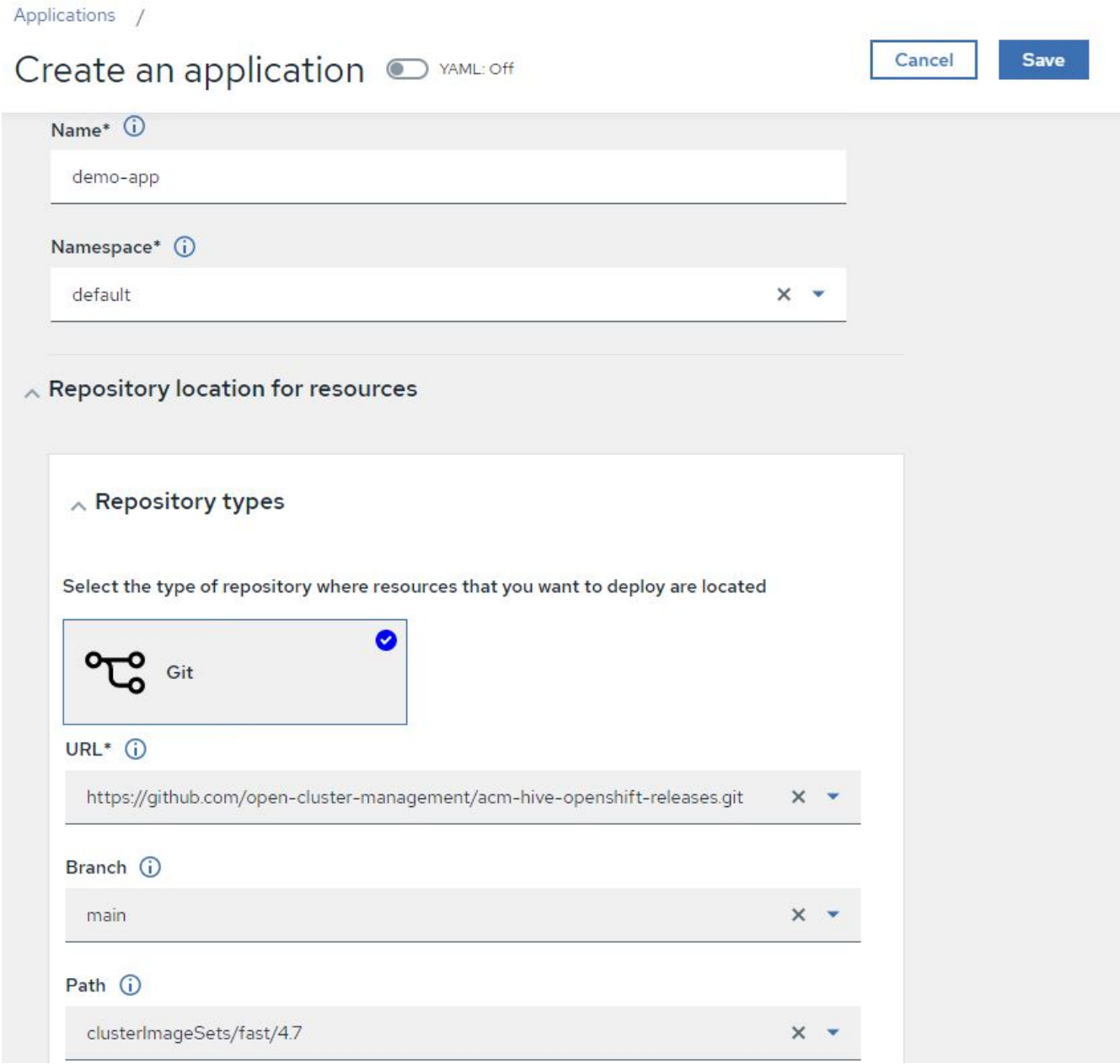
4. 创建并导入多个集群后，您可以从一个控制台监控和管理这些集群。

功能：借助 **NetApp** 在 **Red Hat OpenShift** 上为 **Kubernetes** 提供高级集群管理

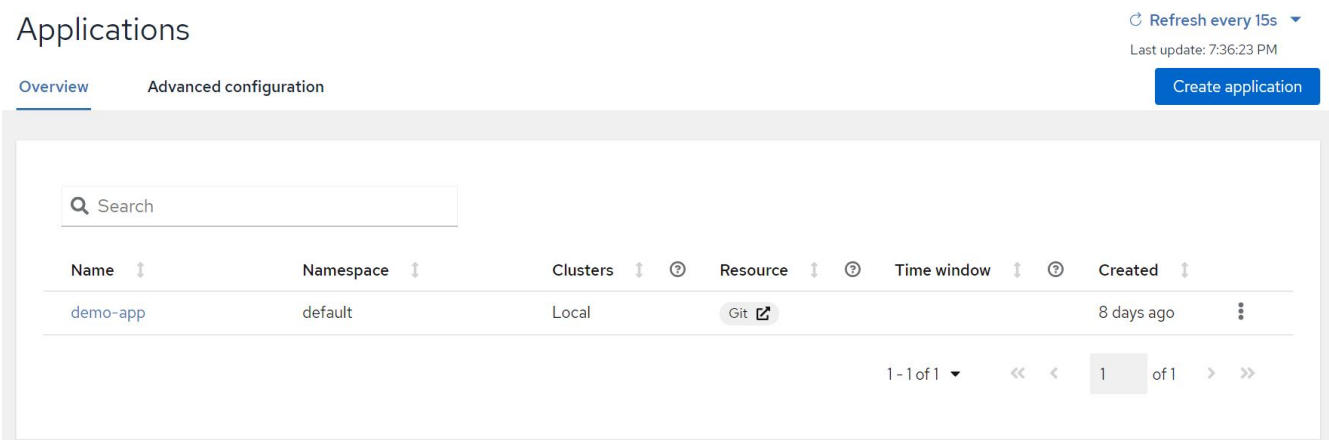
应用程序生命周期管理

要创建应用程序并在一组集群中对其进行管理，

1. 从边栏导航到管理应用程序，然后单击创建应用程序。提供要创建的应用程序的详细信息，然后单击保存。



2. 安装应用程序组件后，此应用程序将显示在列表中。



3. 现在，可以从控制台监控和管理此应用程序。

功能：借助 NetApp 在 Red Hat OpenShift 上为 Kubernetes 提供高级集群管理

监管和风险

通过此功能，您可以为不同的集群定义合规性策略，并确保集群遵循此策略。您可以对策略进行配置，以通知或修复任何规则偏差或违规行为。

1. 从边栏导航到监管和风险。
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，然后选择应遵循此策略的集群。如果要自动修复此策略的违规，请选中 " 如果支持，则强制 " 复选框，然后单击 " 创建 " 。

Governance and risk / Policies /

Create policy YAML: Off

Name *

policy-complianceoperator

Namespace *

default

Specifications *

1 ✕ ComplianceOperator

Cluster selector

1 ✕ local-cluster: "true"

Standards

1 ✕ NIST-CSF

Categories

1 ✕ PR.IP Information Protection Processes and Procedures

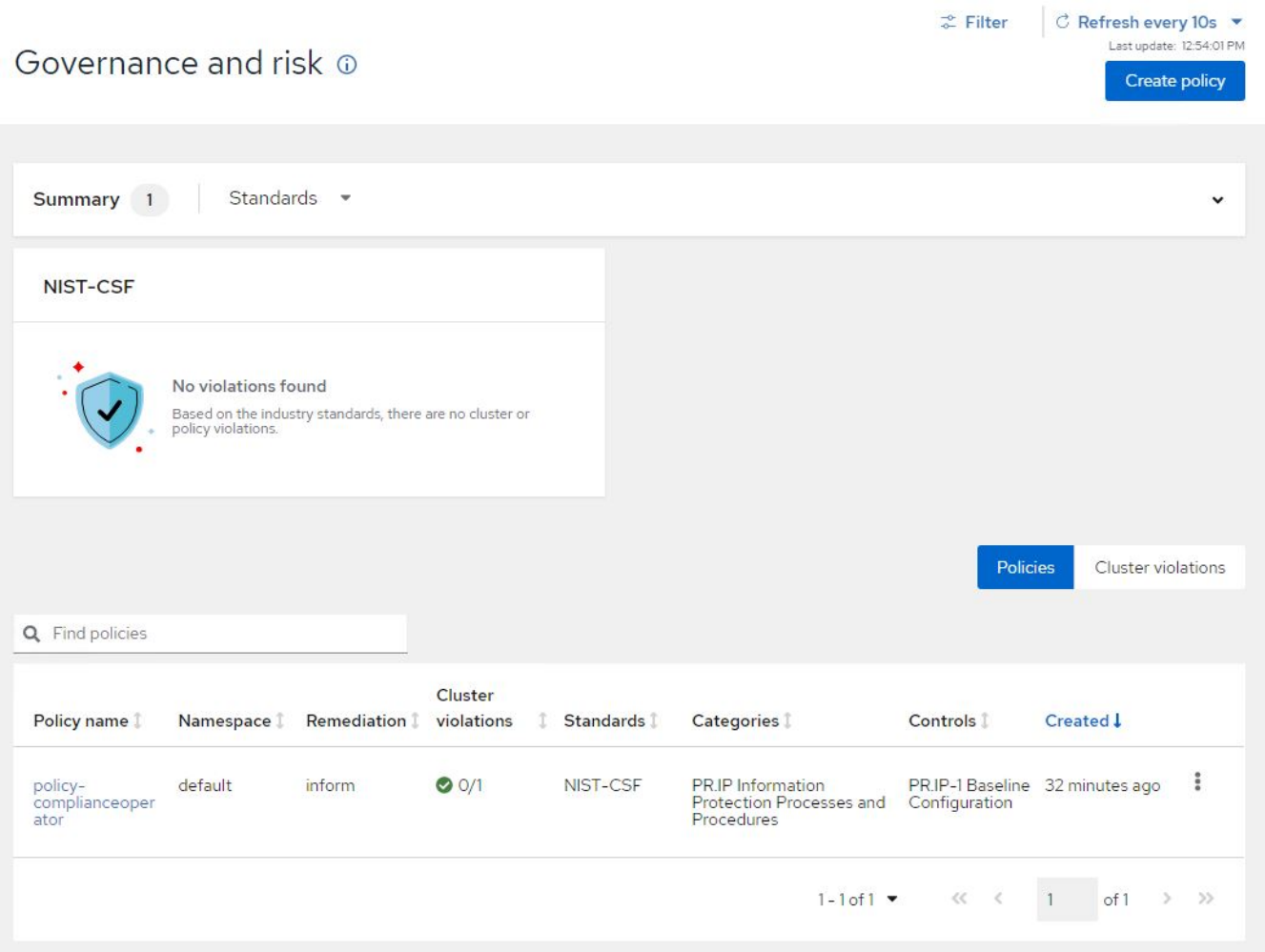
Controls

1 ✕ PR.IP-1 Baseline Configuration

☐ Enforce if supported

☐ Disable policy

3. 配置完所有必需的策略后，可以通过高级集群管理监控和修复任何策略或集群违规。

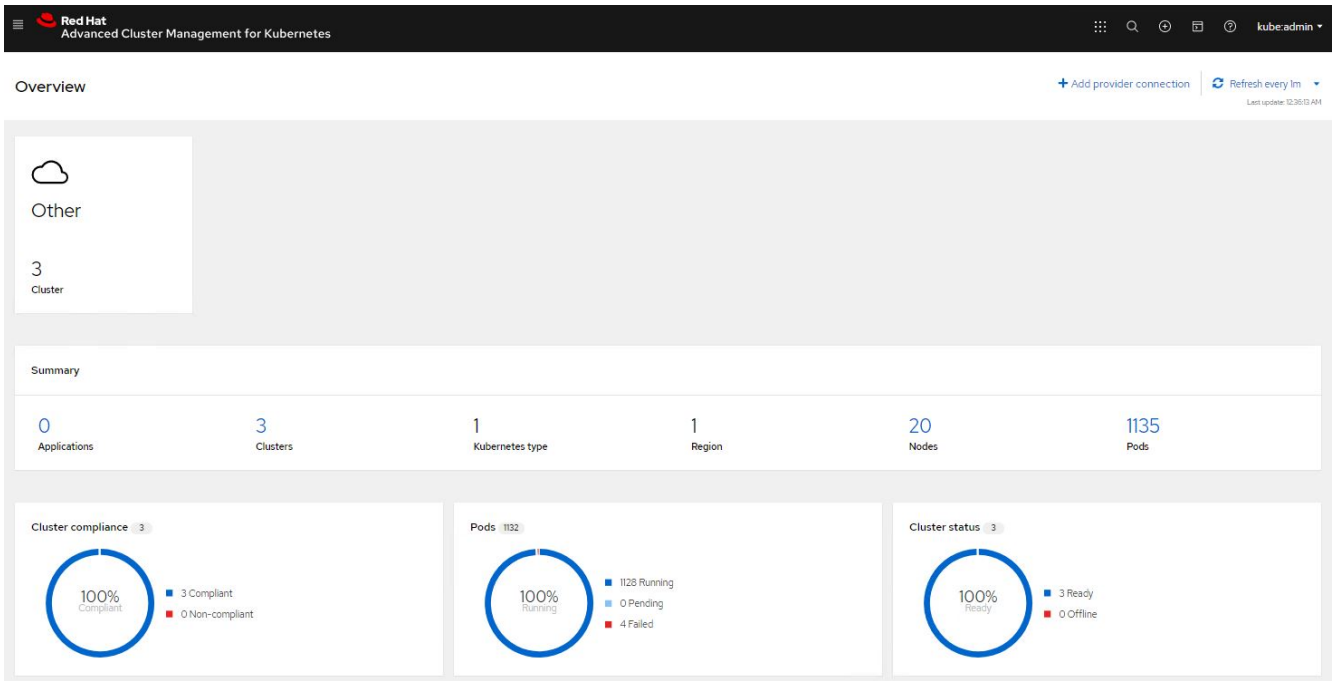


功能：借助 **NetApp** 在 **Red Hat OpenShift** 上为 **Kubernetes** 提供高级集群管理

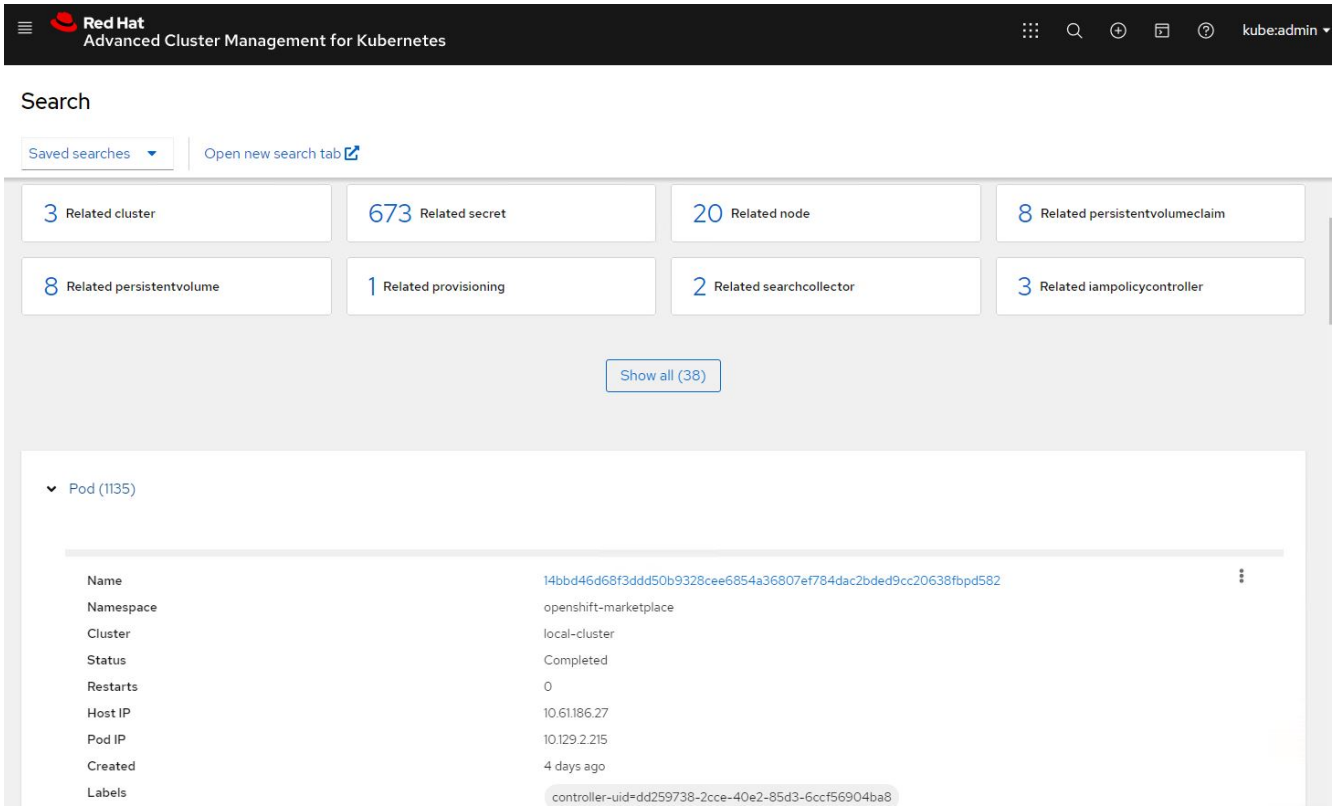
可观察性

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

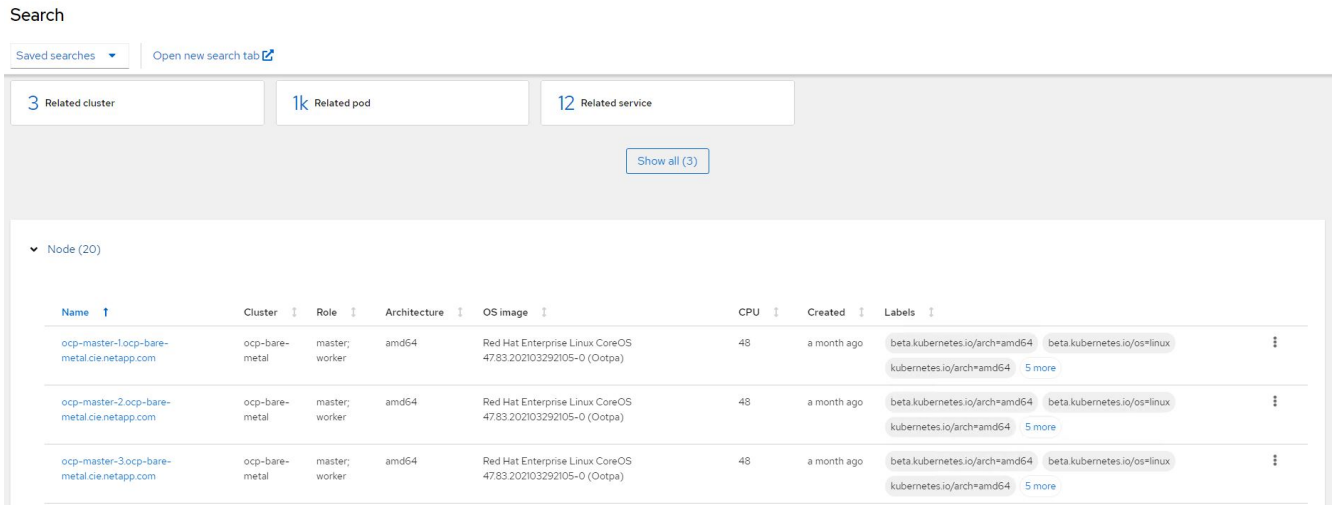
1. 导航到 " 观察环境 ">" 概述 " 。



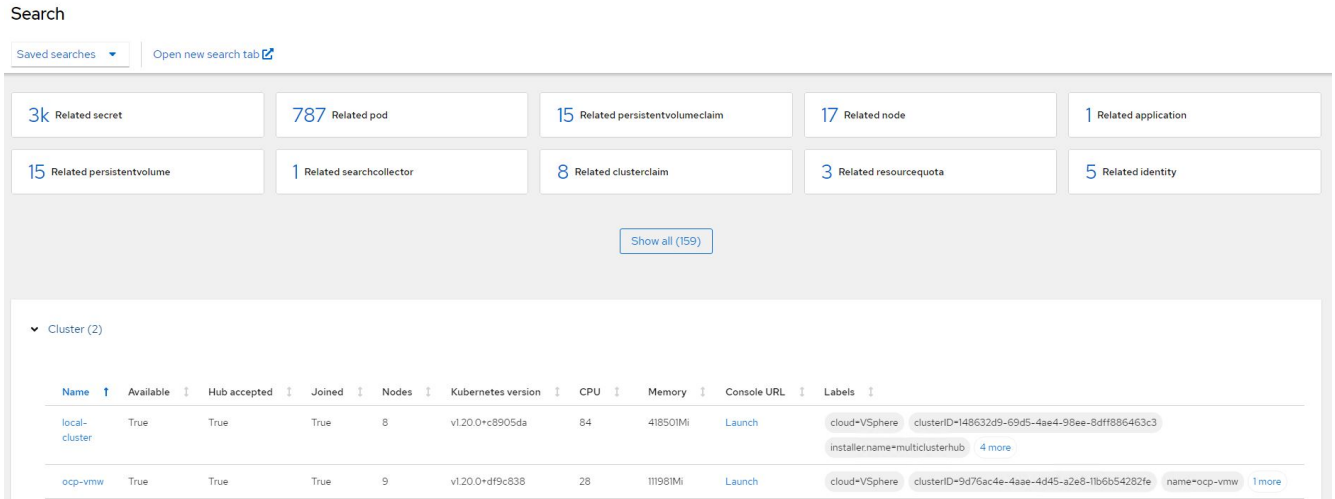
2. 所有集群中的所有 Pod 和工作负载都会根据各种筛选器进行监控和排序。单击 Pod 以查看相应数据。



3. 集群中的所有节点都会根据各种数据点进行监控和分析。单击节点可更深入地了解相应的详细信息。



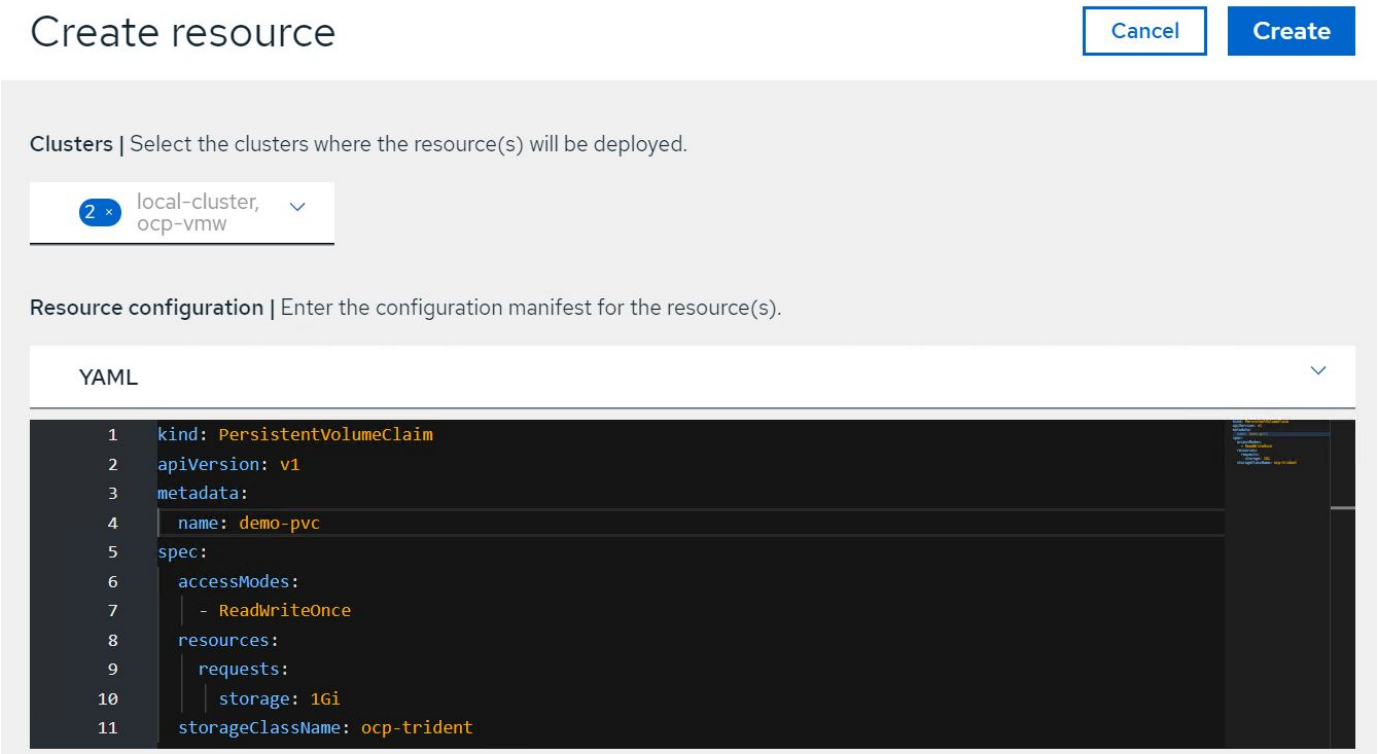
4. 系统会根据不同的集群资源和参数监控和组织所有集群。单击集群可查看集群详细信息。



功能：借助 **NetApp** 在 **Red Hat OpenShift** 上为 **Kubernetes** 提供高级集群管理

在多个集群上创建资源

通过适用于 Kubernetes 的高级集群管理功能，用户可以从控制台同时在一个或多个受管集群上创建资源。例如，如果您的 OpenShift 集群位于不同站点，并由不同的 NetApp ONTAP 集群提供支持，并且希望在两个站点上配置 PVC ，则可以单击顶部栏上的（ + ）符号。然后，选择要创建 PVC 的集群，粘贴资源 YAML ，然后单击创建。



视频和演示：采用 NetApp 的 Red Hat OpenShift

以下视频演示了本文档中介绍的一些功能：

使用Red Hat VtTM通过NetApp ONTAP存储将VM迁移到OpenShift虚拟化

借助Astra Control和NetApp FlexClone技术加快软件开发速度—采用NetApp的Red Hat OpenShift

利用 NetApp Astra Control 执行数据剖析和恢复应用程序

Astra Control Center在CI/CD管道中保护数据

使用Asta控制中心迁移工作负载—采用NetApp的Red Hat OpenShift

工作负载迁移—采用 NetApp 的 Red Hat OpenShift

安装OpenShift虚拟化—使用NetApp的Red Hat OpenShift

使用OpenShift虚拟化部署虚拟机—采用NetApp的Red Hat OpenShift

基于 Red Hat 虚拟化的适用于 Red Hat OpenShift 的 NetApp HCI

追加信息：采用 NetApp 技术的 Red Hat OpenShift

要了解有关本文档中所述信息的更多信息，请查看以下网站：

- NetApp 文档

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

- Astra Trident 文档

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

- NetApp Astra 控制中心文档

  "https://docs.netapp.com/us-en/astra-control-center/"

- Red Hat OpenShift 文档

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

- Red Hat OpenStack Platform 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/"