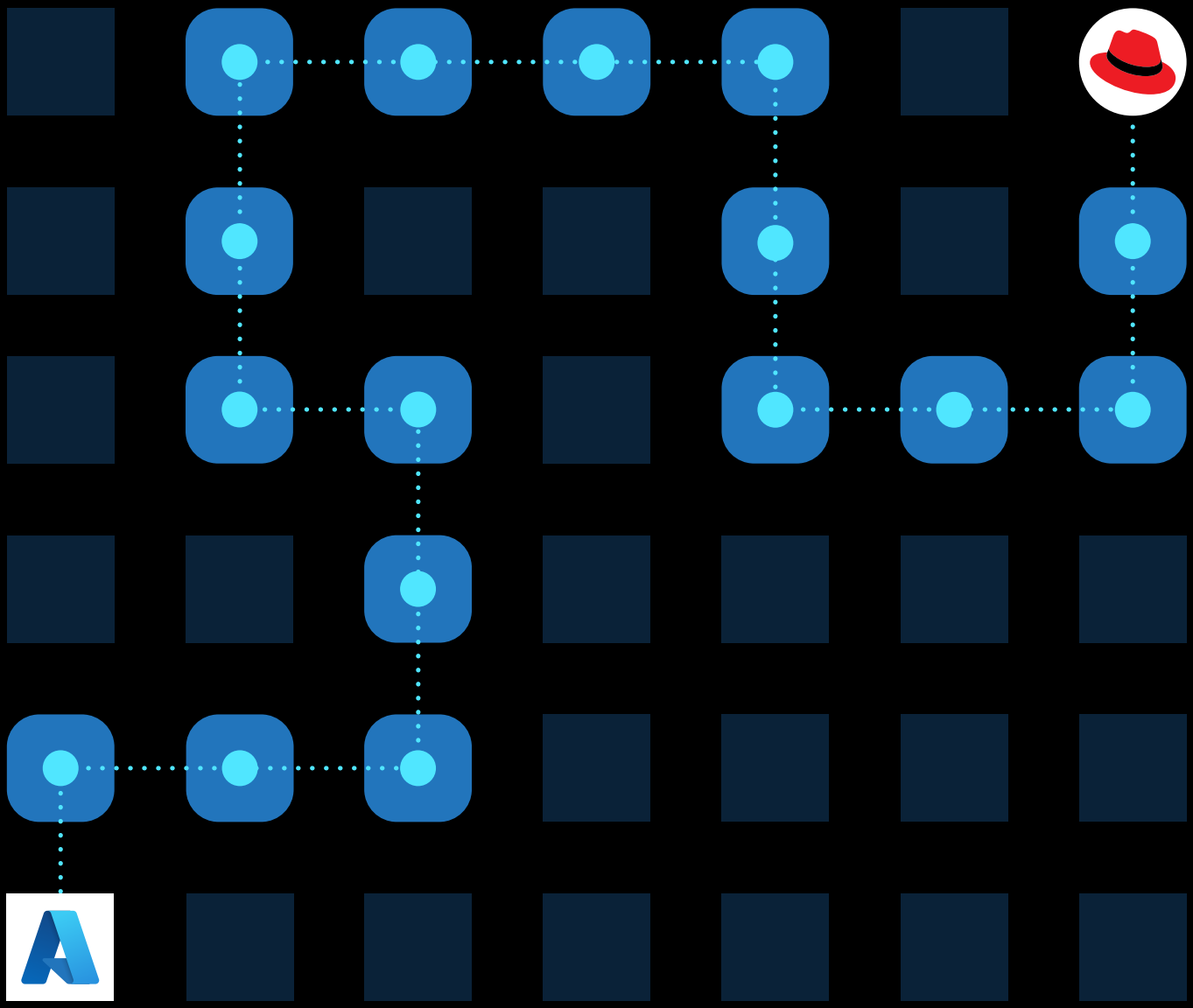
电子书系列

Microsoft Azure

Azure 红帽 OpenShift 入门

从概念验证到生产



Azure 红帽 OpenShift 入门

第 1 章

与微软和红帽交流

如果您在考虑 Azure 红帽 OpenShift，我们希望与您交流一下。尽管本指南为如何使用这个平台提供了切实的指导，但红帽和微软能够提供更加丰富的资源，让您的整个体验更上一层楼。我们双方将携手为您保驾护航，确保 Azure 红帽 OpenShift 成为一个满足您应用创新需求的应用平台。

Azure 红帽 OpenShift 起源于一个简单的原因：客户需要。与过去相比，越来越多的客户（包括大型企业和小型组织）将红帽的产品组合部署到 Microsoft Azure。虽然 OpenShift on Azure 作为一款自助管理式产品从数年前开始就已享有全面的支持，但其设置、部署和集群管理 Day-2 运维具有 Kubernetes 专业技能要求，而且要花时间去管理。这会挤掉实现业务目标的宝贵时间。

随着越来越多客户利用托管服务 Azure 红帽 OpenShift 成功完成大规模部署，我们发现这样一个事实：他们在设置和 Day-2 运维上花费的时间大大缩短，更多的精力集中到了应用上。

我们基于亲身体验创作了本指南，为客户提供在 Azure 红帽 OpenShift 中构建应用的最佳实践。本指南是作为自助手册来编写的。您可以从引言到结语按顺序阅读各个章节，也可搜索您需要的具体信息。

目标受众

本指南主要面向想要了解如何利用 Azure 和红帽 OpenShift 来实现全托管红帽 OpenShift 集群的全服务部署以增强应用构建和部署能力的技术受众，即开发人员、运维人员和平台架构师等。

指南内容

在本指南中，我们会逐一讲解各个重要主题，帮助您了解和采用 Azure 红帽 OpenShift，顺利完成从组织内概念验证到生产部署的整个过程。

“*第 2 章：红帽 OpenShift 简介*”首先向您介绍红帽 OpenShift，并讲述众多开发人员、运维人员和平台架构师选择它作为应用平台的原因以及从中获取的诸多效用。接着，您将了解红帽 OpenShift 成为首选平台的原因。

“*第 3 章：Azure 红帽 OpenShift*”继续探讨，详细介绍托管服务及如何将之交付给像您这样的客户。

“*第 4 章：置备之前 – 企业架构问题*”阐释您在部署 Azure 红帽 OpenShift 前应当要考虑的重要问题。当中包括了许多最佳实践指导，这是我们与实际部署过的客户交流后总结出来的精华。

“*第 5 章：置备 Azure 红帽 OpenShift 集群*”列举了官方文档中关键的 Azure 红帽 OpenShift 集群部署必备资源。

“*第 6 章：置备之后 - Day 2*”介绍了那些通常称为“Day 2”的置备后任务。本指南尽可能说明如何利用 Azure 红帽 OpenShift 这一托管服务来简化您的工作，省去自己解决这些任务的烦恼。

“*第 7 章：部署应用到红帽 OpenShift*”简要介绍了如何在此平台上部署应用，但要注意的是，这与在任何其他环境中运行红帽 OpenShift 并无差别。

“*第 8 章：浏览应用平台*”概括介绍了应用平台的一些关键功能，如容器镜像仓库、管道和无服务器等。

“*第 9 章：集成其他服务*”介绍了如何利用 Azure Service Operator 和 Azure DevOps 等类似服务进行平台集成。

“*第 10 章：启动工作负载和培训团队*”提供了关于培训应用团队和在组织中推动采用此服务的一些最佳实践和建议。

“*第 11 章：结语*”为总结内容。

“*第 12 章：术语表*”提供了一份术语表。

第 2 章

红帽 OpenShift 简介

本章简略介绍红帽 OpenShift 的定义、它的用法，以及该服务通常能给红帽客户带来哪些益处。如果您是第一次接触 OpenShift，可以把这当作一份实用初级读物；如果您想要多了解一下这个平台，则可把这当作一本进阶资料。

红帽 OpenShift 概述

红帽 OpenShift 是一个企业就绪型应用平台，可以实现全堆栈自动化运维，以管理混合云和多云部署。红帽将其优化从而提高开发人员的生产力并推动创新。凭借自动化运维和简化的生命周期管理，开发团队借助红帽 OpenShift 既可以构建和部署新的应用，也可以帮助运维团队置备、管理和扩展 Kubernetes 平台。

在整个交付过程中，开发团队可通过安全扫描和加密签名，从数百家合作伙伴那里获取经验证的镜像和解决方案。除此之外，还可以访问按需提供的镜像，并获得对各种第三方云服务的本地访问权限，所有这些均可在同一平台实现。

运维团队通过内置的日志和监控可以随时随地查看跨团队的部署情况。红帽 Kubernetes Operator 嵌入了独特的应用逻辑来发挥服务作用，不是简单配置，而是针对性能进行调整，并且可以一键从操作系统进行更新和修补。简而言之，红帽 OpenShift 是一站式服务，能够帮助企业充分释放 IT 团队和开发团队的潜能。

OpenShift 云服务 – 成本节约和业务效益

目前已有数千家客户选择通过红帽 OpenShift 来改变应用的交付方式，改善与客户之间的关系，并提高竞争优势，以争当行业领导者。IDC 调查报告“红帽 OpenShift 的商业价值，2021 年 3 月”展示了受访组织借助红帽 OpenShift 平台实现的价值，以更高的质量、更快的速度将应用和功能部署到业务中，同时优化开发和 IT 相关的成本和员工时间需求。

关键指标如下：

- 5 年实现 636% 投资回报率
- 10 个月投资回收期
- 5 年运维成本降低 54%
- 每年新开发功能增多 3 倍
- 应用开发人员生产力提高 20%
- 计划外停机时间减少 71%
- IT 基础架构团队效率提高 21%

来源：IDC 白皮书（红帽赞助），*红帽 OpenShift 的商业价值*， 文档编号 US47539121，2021 年 3 月

在红帽 OpenShift 实现的这一价值基础上，包括 Azure 红帽 OpenShift 在内的红帽 OpenShift 云服务带来了更多业务效益。标题为“红帽 OpenShift 云服务总体经济影响™ - 成本节约和业务效益”的 Forrester 研究报告总结了多项关键经济效益。

OpenShift 云服务的关键经济效益如下：

- 468% 投资回报率 (ROI)
- 408 万美元净现值 (NPV)
- 6 个月投资回收期

除了这些经济效益外，Forrester 报告还总结了下列可量化效益关键调查结果：

- **开发速度加快**：通过使用红帽 OpenShift 云服务，组织可以将开发周期最多缩短 70%。
- **从基础架构维护工作中省下 20% 开发人员时间**：受访者表示，红帽 OpenShift 云服务免去了开发人员维护应用开发基础架构的，可以全身心投入到构建产品或解决方案上。省出的这些开发人员时间在 3 年中可创造等价于超过 230 万美元的价值。
- **运维效率提高 50%**：受访者表示，由于 OpenShift 云服务是托管服务，使用该解决方案意味着他们可以重新调配 50% 的 DevOps 员工，让他们从基础架构管理工作中脱身，去负责效益更高的其他工作。这一运维效率提升在 3 年中实现的价值超过 130 万美元。*

报告中也指出了如下无法量化的效益：

- **开放人员满意度和留任率**：受访者强调，开发人员从红帽 OpenShift 云服务获益良多，可以将更新工作划分为更细小的部分，减轻了在非常有限的时间内进行密集测试的压力，同时也减少了投入生产后响应救灾演习的需求。
- **提高安全和降低风险**：受访者分享了红帽 OpenShift 云服务如何自动完成特定的功能与安全性更新，免除了手动维护需求，同时确保环境安全无虞。
- **可靠性**：受访者反映，使用红帽 OpenShift 云服务可以让他们的应用平台提高长期运行可靠性，虽然环境扩大了，但服务中断或系统故障变得更少了。
- **可移植性和业务连续性**：受访者还指出，红帽 OpenShift 云服务的可移植性、可扩展性和灵活性保障了业务连续性，同时也对他们的灾难恢复策略有所助益。

来源：Forrester，*红帽 OpenShift 云服务总体经济影响*，2021 年 12 月

*** 扩展阅读：**企业通过云服务提升敏捷性

“选择红帽 OpenShift 还是裸机 Kubernetes？” – 自行搭建 Kubernetes 应用平台的代价

红帽 OpenShift 常被称为“企业级 Kubernetes”，但很难一眼洞悉其真正含义。客户常问，“OpenShift 还是裸机 Kubernetes？” 然而，**OpenShift 已在使用 Kubernetes**，清楚这一点非常重要。在 OpenShift 总体架构中，Kubernetes 不仅是构建 OpenShift 平台的基础，也为运行 OpenShift 提供了诸多工具。

Kubernetes 是一个无比重要的开源项目，是云原生计算基会的关键项目之一，也是运行容器的必备技术。

不过，OpenShift 的潜在客户想要了解的真正问题可能是：**“能不能单纯使用 Kubernetes 来运行我的应用？”** 许多组织首先部署了 Kubernetes，而后发现几天后他们就能让容器运行了，甚至还能运行企业应用。然而，随着 Day-2 运维开始，安全性需求开始增加，并且部署的应用变多，一些组织发现自己落入了要自行利用 Kubernetes 技术构建**平台即服务（PaaS）**的陷阱中。于是，他们添加开源入口控制器，编写一些脚本来连接**持续集成 / 持续部署（CI/CD）**管道，接着又尝试部署更加复杂的应用… 而这时，开始滋生各种各样的问题。如果部署 Kubernetes 是冰山一角，那么复杂的 Day-2 管理就是那座冰山隐匿在水下、可以撞翻轮船的庞大主体。

尽管可以开始攻克这些挑战和问题，但这通常需要组建一个由数人组成的运维团队，花费几周乃至数月的时间来构建和维护这个“在 Kubernetes 上构建的自定义 PaaS”。这会导致组织效率低下，从安全性和认证角度来看支持也会变得复杂，而且在引入开发人员团队时也必须从头开始开发一切。如果组织要梳理构建和维护这个自定义 Kubernetes 平台（即，Kubernetes 加上成功运行容器需要的所有额外组件）的所有任务，可以将它们划分到以下几个组别：

- **集群管理**：这包括安装操作系统、修补操作系统、安装 Kubernetes、配置 CNI 网络、集成身份验证、设置入口和出口、设置持久存储、强化节点、修补安全防护，以及配置底层云 / 多云。
- **应用服务**：这包括日志聚合、健康检查、性能监控、安全补丁、容器镜像仓库，以及应用预演流程设置。
- **开发人员集成**：这包括 CI/CD 集成、开发工具 /IDE 集成、框架集成、中间件兼容性、提供应用性能仪表板，以及 RBAC。

尽管还有许多操作和技术会添加到这份清单中，并且其中大多数活动对于任何认真使用容器的组织而言都是不可或缺的，但设置所有这些所牵涉到的复杂性、时间和工作对于持续维护这些个别组件而言无关紧要。每一项集成都需要全面测试，每一个组件和活动有着不同的发布周期、安全策略和补丁。

与 Kubernetes 相比，红帽 OpenShift 能给您带来什么?

当组织准备在生产中运行容器时，如果仅用裸机 Kubernetes 来构建，那么通常会安装上一节中提到的那些组件，并将它们集成到一起，组成自己设计的某种应用平台。

Azure 红帽 OpenShift 将上一节中提到的所有这些组件结合到了同一平台当中，为 IT 团队提供了运维便利，同时为应用团队提供了执行其任务所需的资源。所有这些主题在本指南的后面部分还有更详细的介绍，不过现在我们先来看看两者之间的重要区别：

- **部署便利性**：在 Kubernetes 中部署应用可能会比较耗时。需要将您的 GitHub 代码拉取到一台机器上，启动容器，并将其托管到 Docker Hub 之类的注册中心，最后还要了解您的 CI/CD 管道，这项工作可能非常复杂。然而，OpenShift 却可以将繁重的工作和后端工作自动化，您只需要创建一个项目并上传代码即可。
- **安全防护**：今天，我们看到大部分 Kubernetes 项目都是由多个开发人员和运维人员组成的团队来处理。即使 Kubernetes 现在支持 RBAC 和 IAM 等控制功能，也仍需耗费时间进行手动设置和配置。红帽和 OpenShift 经过多年实践，已经确立了诸多安全最佳实践，客户可以直接使用。您只需添加新用户，OpenShift 就会帮您处理诸如名字间隔和创建不同安全策略等事务。
- **灵活性**：借助 Azure 红帽 OpenShift，您可以充分利用部署、管理和更新方面成熟可靠的最佳实践。平台会为您处理后台的各种繁重工作，无需您太多亲自动手，从而让您能更快交付自己的应用。不仅对于喜欢按指示操作和使用简化方法的团队来说非常友好， Kubernetes 平台也可以支持您自定义 CI/CD DevOps 管道，在开发过程中为您提供更多的灵活性和创造性空间。

- **日常运维**：集群是由一组虚拟机组成的，因而运维团队不可避免地要启动新的虚拟机以添加到集群中。通过 Kubernetes 进行配置可能会很耗时，也很复杂，需要开发脚本来设置自注册或云自动化等事项。如果使用 Azure 红帽 OpenShift，则集群的置备、扩展和升级操作都可以自动进行，并由平台管理。
- **管理**：虽然可以利用随同任何分发提供的 Kubernetes 标准工具和仪表板，但许多开发人员需要更加齐全和稳健的平台。Azure 红帽 OpenShift 提供基于 Kubernetes API 和功能构建的出色 Web 控制台，方便运维团队管理他们的工作负载。

*第 8 章“浏览应用平台”*提供了 Azure 红帽 OpenShift 众多重要增值功能的指导说明。

摘要

红帽和微软的许多共同客户选择 Azure 红帽 OpenShift，作为他们采用容器化应用的首选应用平台。

在下一章中，我们将探索红帽 OpenShift 这项云服务，深度解析它的架构、集成和管理。

第 3 章

Azure 红帽 OpenShift

借助 **Azure 红帽 OpenShift**，您可以部署全托管式红帽 OpenShift 集群，而不用再自行构建和管理基础架构来实施运行。

Azure 红帽 OpenShift 是一个云原生按需应用平台，由微软和红帽共同设计、运维和支持。专设的**站点可靠性工程**（**SRE**）团队负责 OpenShift 集群的自动化、扩展和安全保护，携手提供综合全面的支持体验。有了 Azure 红帽 OpenShift，不需要对任何虚拟机进行运维，也不需要打补丁。红帽和微软可以替您修补、更新和监控控制平面节点和工作节点。您的 Azure 红帽 OpenShift 集群会部署到您的 Azure 订阅中，并包含在您的 Azure 账单中。

利用 Azure 红帽 OpenShift 更快交付应用：

- 使开发人员能够通过内置的 CI/CD 管道进行创新，然后轻松将您的应用连接到数百个 Azure 服务，比如 MySQL、PostgreSQL、Redis 和 Azure Cosmos DB 等。
- 借助自动化置备、配置和运维，化繁为简并扫清生产力障碍。
- 按照自己的要求进行扩展；您可以花上几分钟用三个工作节点启动一个高可用性集群，然后随着应用需求变化进行扩展；此外，也可选择标准、高内存或高 CPU 的工作节点。
- 实现企业级运维、安全性与合规性，并提供综合全面的支持体验。

接下来，我们将深入探讨红帽 OpenShift 构建和运维背后的技术细节。

架构

Azure 红帽 OpenShift 使用 Azure 基础架构服务作为红帽 OpenShift 安装的基础，如虚拟机、网络安全组、存储帐户和其他 Azure 服务等。同时，Azure 架构完整部署到您的 Azure 订阅中，让您轻松地集成帐户中已在运行的其他服务。

OpenShift 本身是基于红帽企业 Linux CoreOS 构建的，后者承载由可协同工作的较小解耦单元组成的微服务型架构。每台虚拟机上都运行 `kubelet` 服务，这是 Kubernetes 集群的基石。此架构背后的数据库是 **etcd**，这是一个可靠的集群化键值存储，在控制平面上运行。

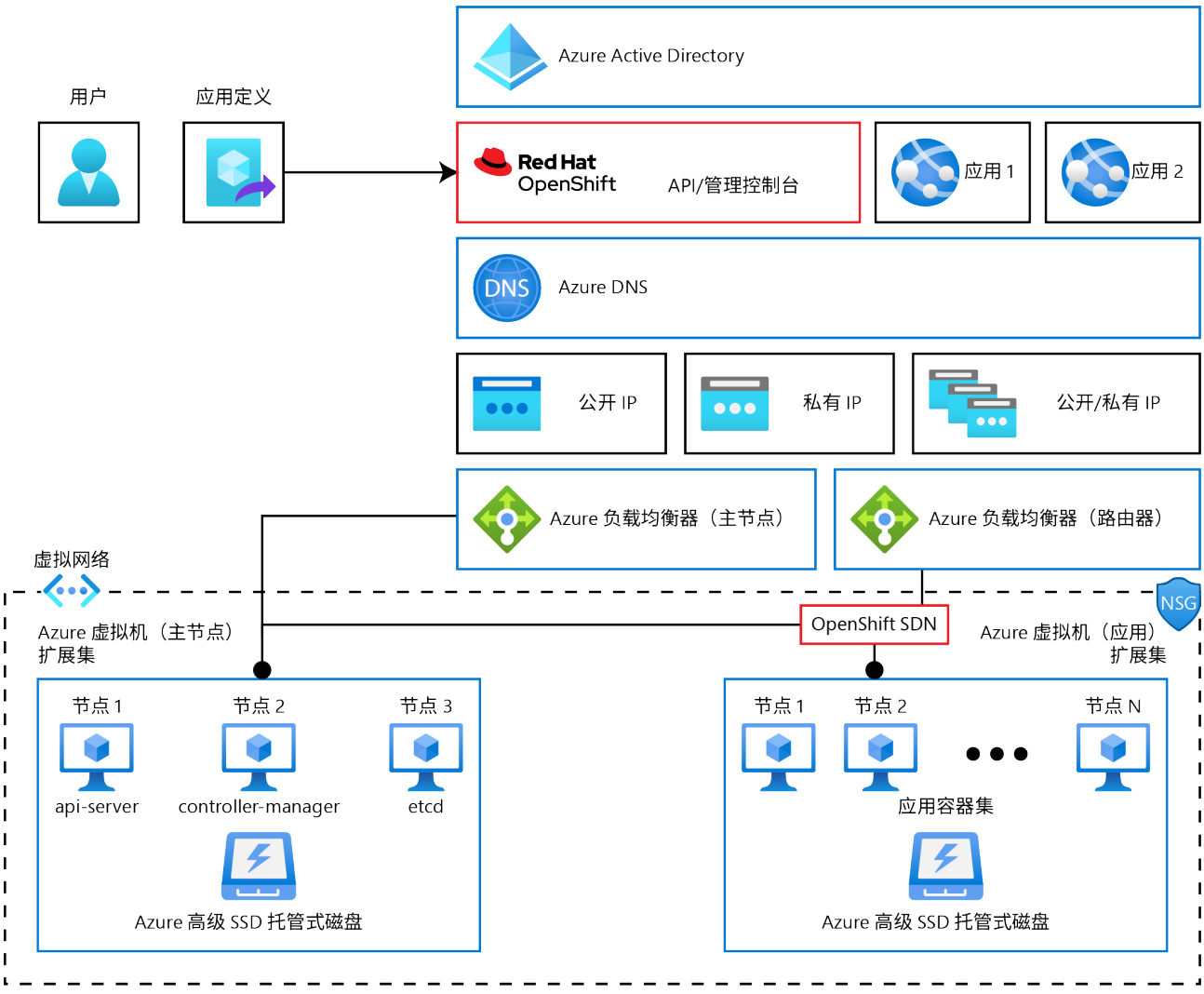


图 3.1：Azure 红帽 OpenShift 架构

以下两小节（*计算 – 控制和工作节点*以及*网络*）更加详细地介绍图中所含的内容。

计算 – 控制和工作节点

红帽 OpenShift 的架构在虚拟机（节点）上运行，这些虚拟机担当三个具体角色之一，即控制、基础架构或应用。但在撰写本文之时，Azure 红帽 OpenShift 版本 4 不支持基础架构节点，因此本书只讨论控制节点和工作节点。

控制节点（过去称为**主**节点）是含有 Kubernetes 集群基本组件的虚拟机。这包括 API 服务器、控制器管理器服务器、调度器和 etcd。它们负责管理 Kubernetes 集群，并将容器集调度到工作节点上运行。

- **Kubernetes API 服务器**验证和配置容器集、服务及复制控制器的数据。它也为集群的共享状态提供一个焦点。
- **Kubernetes 控制器管理器**监测 etcd 中的对象更改，如复制、命名空间和服务帐户控制器对象，然后使用 API 来实施指定的状态。数个这样的进程就构成一个集群，但一个时间点上只有一个活跃领导者。
- **Kubernetes 调度器**监视新近创建、还未分配节点的容器集，并选择最适合的节点来承载这个容器集。
- **etcd** 存储持久主机状态，而其他组件则监测 etcd 的变化，以便使自身进入指定状态。

应用节点是实际运行您的应用的节点。

Kubernetes 集群中的每个节点都运行一个称为 `kubelet` 的服务（负责维护**容器运行时接口（cri-o）**）、一个服务代理，以及一些在各个节点上运行的其他基本服务。所有节点都利用 OpenShift 的软件定义网络技术进行连接。Azure 红帽 OpenShift 中用的是 **Open Virtual Network（OVN）**，它基于 Azure 虚拟网络运行。

Azure 红帽 OpenShift 创建在 Azure 虚拟机上运行的节点，这些虚拟机连接到 Azure 磁盘进行存储。您可能会发现这些磁盘的容量比较大 — 1 TB。这是因为，Azure 中存储性能的 IOPS 保障与磁盘大小相关。1 TB 大小可以保证为 etcd 数据库使用的基础磁盘提供充足的带宽。

网络

Azure 红帽 OpenShift 需要一个 Azure 虚拟网络，并且配置两个子网，分别用于控制平面节点和工作节点。两个网络的最小大小是 /27（32 个地址）。不过，如果您准备日后扩展集群，那么设置时要谨慎一些，不要设置得太小。

两个 Azure 负载均衡器（图中标记为**主均衡器**和**路由器**）按照如下方式引导流量：

- 主均衡器 / 控制平面负载均衡器：发送 OpenShift API 用户的入口流量。也为控制平面节点提供出站连接
- 路由器 / 应用负载均衡器：通过 OpenShift“路由器”或入口控制器，将入口流量发送到 OpenShift 中运行的应用。也为工作节点提供出站连接

网络文档中可以找到一篇有关网络配置、要求和限制的优秀文章。

集成其他 Azure 服务

作为 Azure 上的一项原生服务，Azure 红帽 OpenShift 可以和您在 Azure 上使用的诸多其他服务一同部署，并与它们集成。下方的高级概述只是介绍了一些通常集成的 Azure 基础架构服务。

图 3.2 演示了 OpenShift on Azure 的诸多常见 Azure 服务集成点：

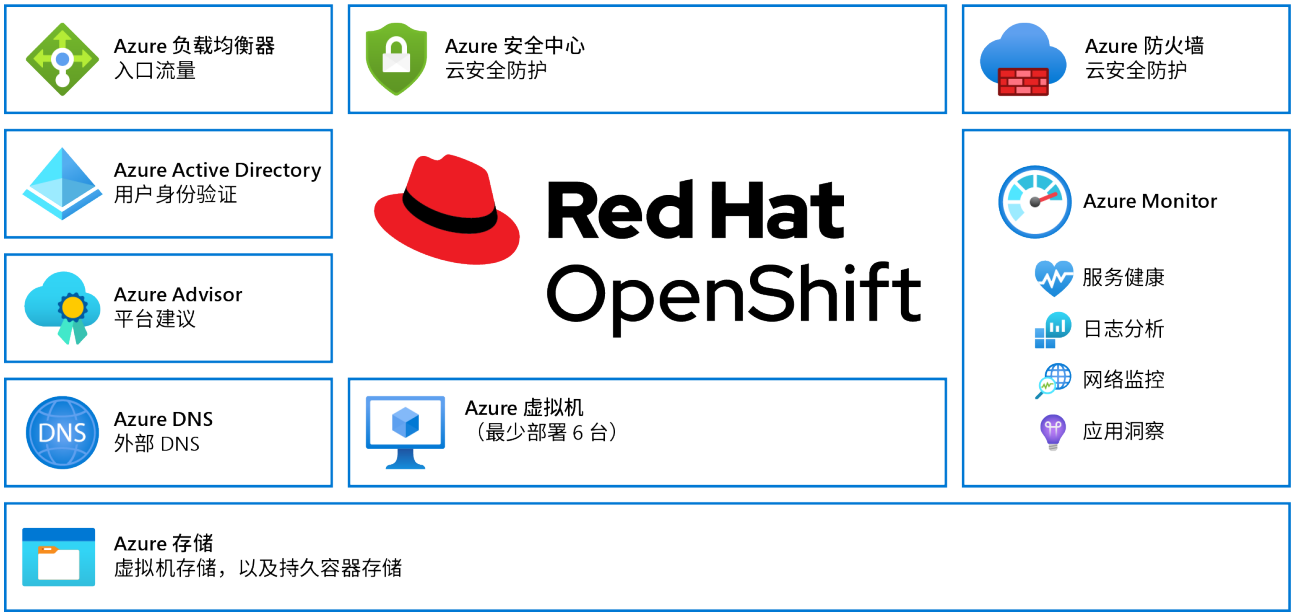


图 3.2：常见 Azure 服务集成点

另外，OpenShift 上运行的应用可以通过使用 Azure Service Operator 与 Azure 服务进行紧密集成。这将在*第 9 章：集成其他服务*中更详细地说明。

管理

若要通过一个简单描绘来了解 Azure 红帽 OpenShift 服务中具体管理什么，可以想象成从数据中心一直到集群 operator 的一切内容都在“被管理”。集群 operator 是在控制平面节点上运行的服务，负责监控和更新集群的健康状态。这意味着，微软和红帽将携手监控、维护和管理这些组件，以便让集群正常在线运行。所有超出集群 operator 范畴的依旧由客户自己负责。

作为 Azure 红帽 OpenShift 的使用者，您将被授予集群的完整 **cluster-admin** 访问权限；也就是说，您也在保持集群完整性方面同样肩负责任。重要的是要理解支持政策，清楚自己对集群可做和不可做的事情。

Azure 红帽 OpenShift 责任一览表中详细说明了微软和红帽在这项云服务中具体负责哪些事务。

身份验证和授权

Azure Active Directory 是为 Azure 红帽 OpenShift 集群提供身份验证的一个常用方式。但这不是强制要求，您也可选用其他身份验证机制，例如“通过 GitHub 登录”或简单的“密码文件”。

如果使用 Azure Active Directory，Azure 红帽 OpenShift 和 Kubernetes API 会转发身份验证请求。用户将出示他们的凭据，并根据角色获得授权。

身份验证层会对向 Azure 红帽 OpenShift API 提出请求的用户进行身份识别。然后，授权层会使用请求用户的信息来确定是否同意相应请求。

“*身份验证 - Azure Active Directory*”小节中介绍了 Azure Active Directory 的配置说明。如果使用 Azure Active Directory 以外的其他身份验证提供程序，其流程在功能上非常相似。

授权将在 Azure 红帽 OpenShift 策略引擎中处理，该引擎定义了“创建容器集”或“列出服务”等操作，并在策略文件中根据角色进行了分组。角色会按照用户标识符或群组标识符绑定到用户或群组。如果一个用户或服务帐户尝试采取某项操作，策略引擎会先核实分配给该用户的一个或多个角色（例如，客户管理员或当前项目管理员），然后才会允许其继续。

图 3.3 揭示了集群角色、本地角色、集群角色绑定、本地角色绑定、用户、群组和服务帐户之间的关系。

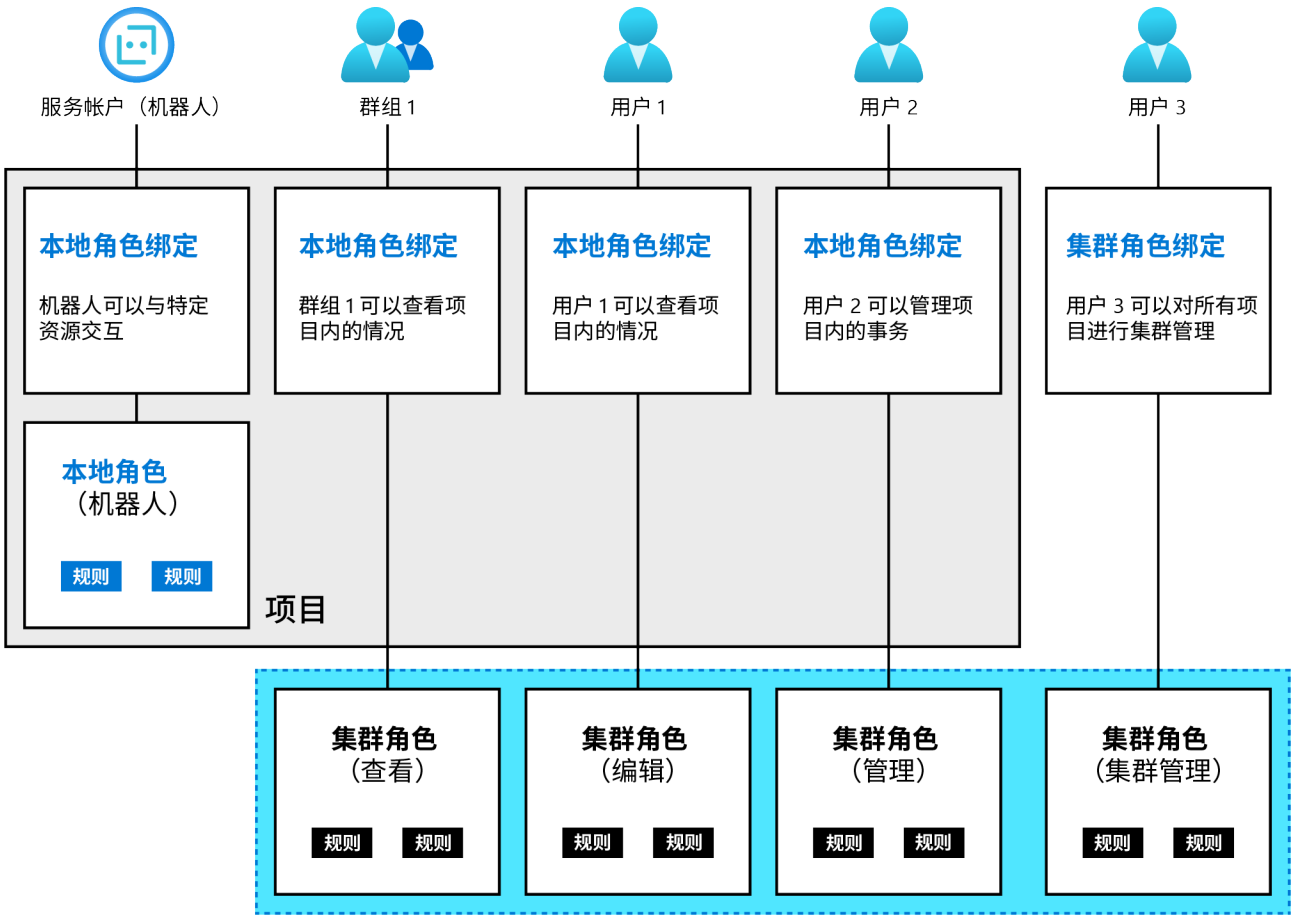


图 3.3：角色之间的关系

红帽 OpenShift 文档中提供了[身份验证提供商的完整列表](#)。

支持

Azure 红帽 OpenShift 采用独有的方式来管理服务支持。微软和红帽支持团队与全球站点可靠性工程（SRE）团队携手合作，共同协助这项服务的运维。

当客户在 Azure 门户中提出支持请求时，其请求将由微软和红帽工程师进行分流和处理，以快速解决其请求，不受这些请求是在 Azure 平台层面提出，还是在 OpenShift 层面提出的限制。

下方概述了整合支持流程：

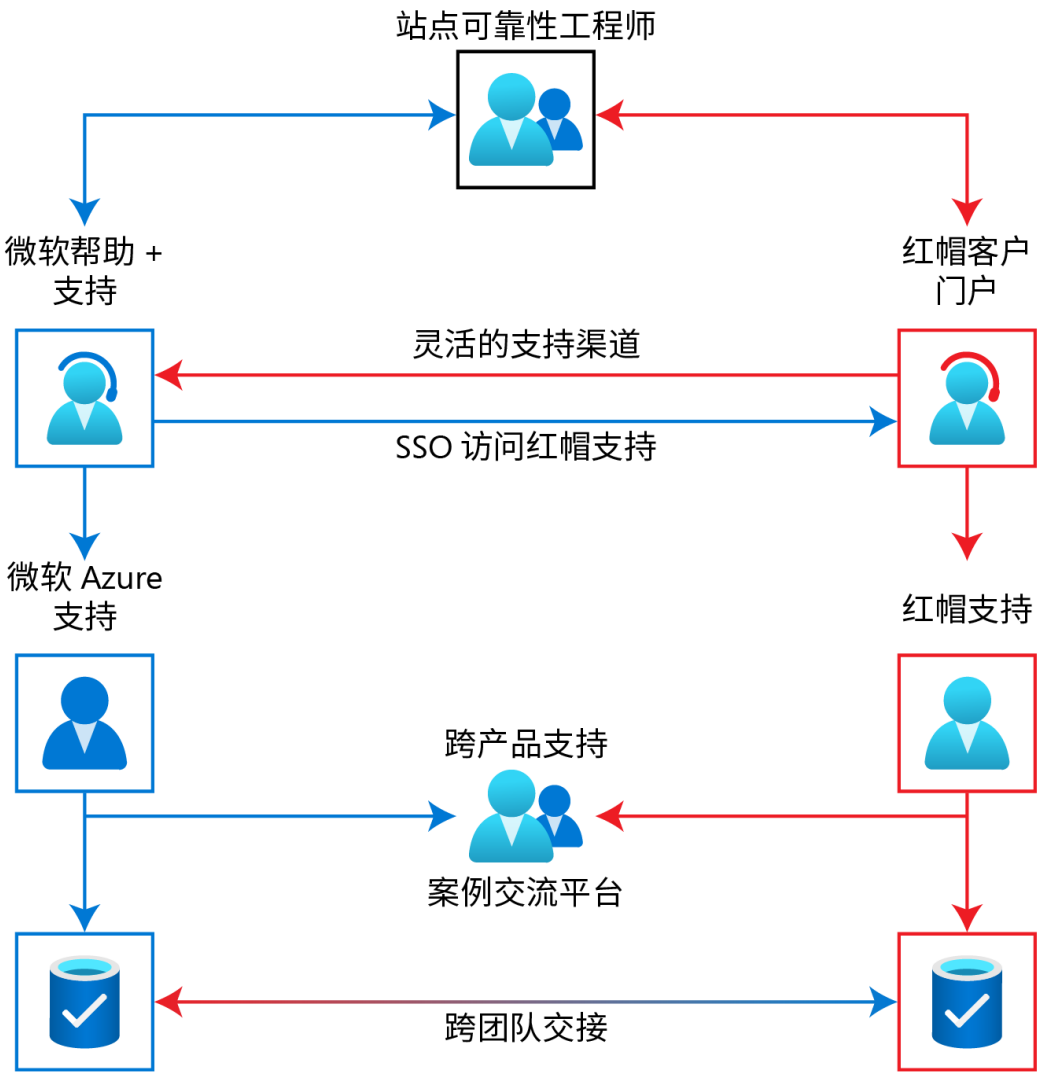


图 3.4：整合支持流程

图 3.4 中显示，客户可以在微软支持门户或红帽支持门户中提交支持工单。请注意，若要访问红帽支持门户，集群应注册到 OpenShift 集群管理器。

征得客户同意后，微软支持工程师可在任何阶段通过案例交流平台与红帽进行无缝协作。红帽支持工程师请求与微软协作时同样如此。如有需要，两家公司的支持工程师可与 SRE 联络，以采取补救措施来修复集群。

价格和订阅

相比自助（DIY）安装，使用 Azure 红帽 OpenShift 具有一大好处，计算、网络、存储和 Azure 基础架构以及 OpenShift 订阅将一同记入您的 Azure 订阅账单，而不会分开收费。要对解决方案费用有个大致了解，只需将 Azure 红帽 OpenShift 添加到 Azure 价格计算器的报价生成器中。

以下指南可帮助您了解价格页面：

1. 在搜索框中键入 *openshift*，并将它添加到新的估算中。

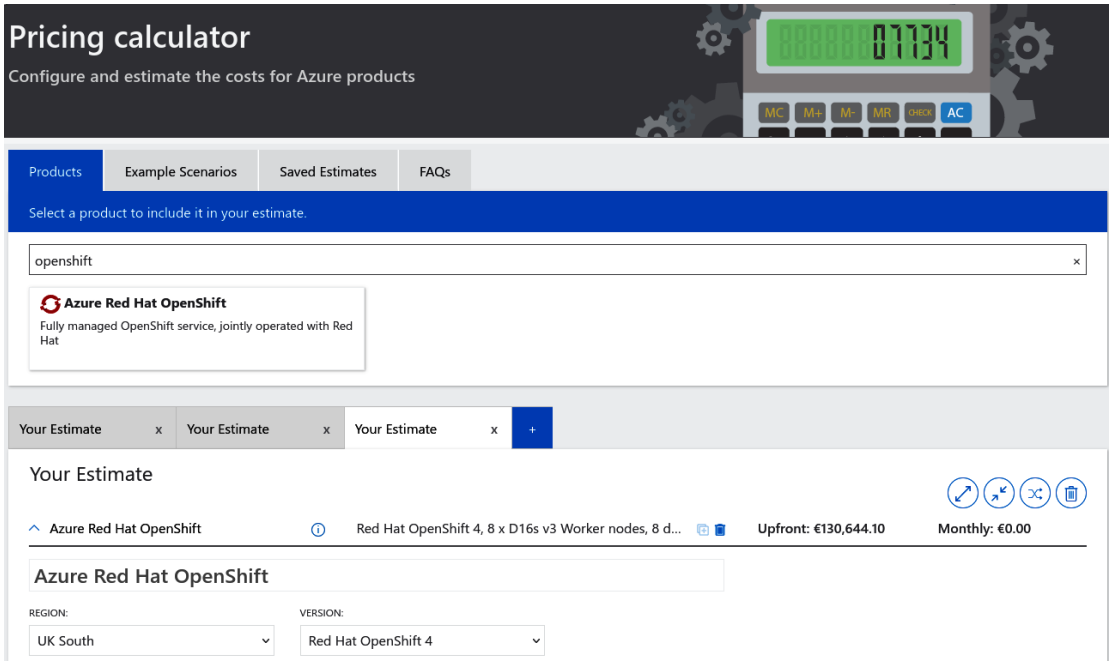


图 3.5：价格计算器面板

2. 页面底部有一个下拉式组合框，您可以将计价单位更改为当地货币。本例中使用的是英镑（£）。
3. 设置用来部署 Azure 红帽 OpenShift 的 Azure 地区。价格在不同地区之间会稍有差异，以考虑不同 Azure 地区的不同计算成本。

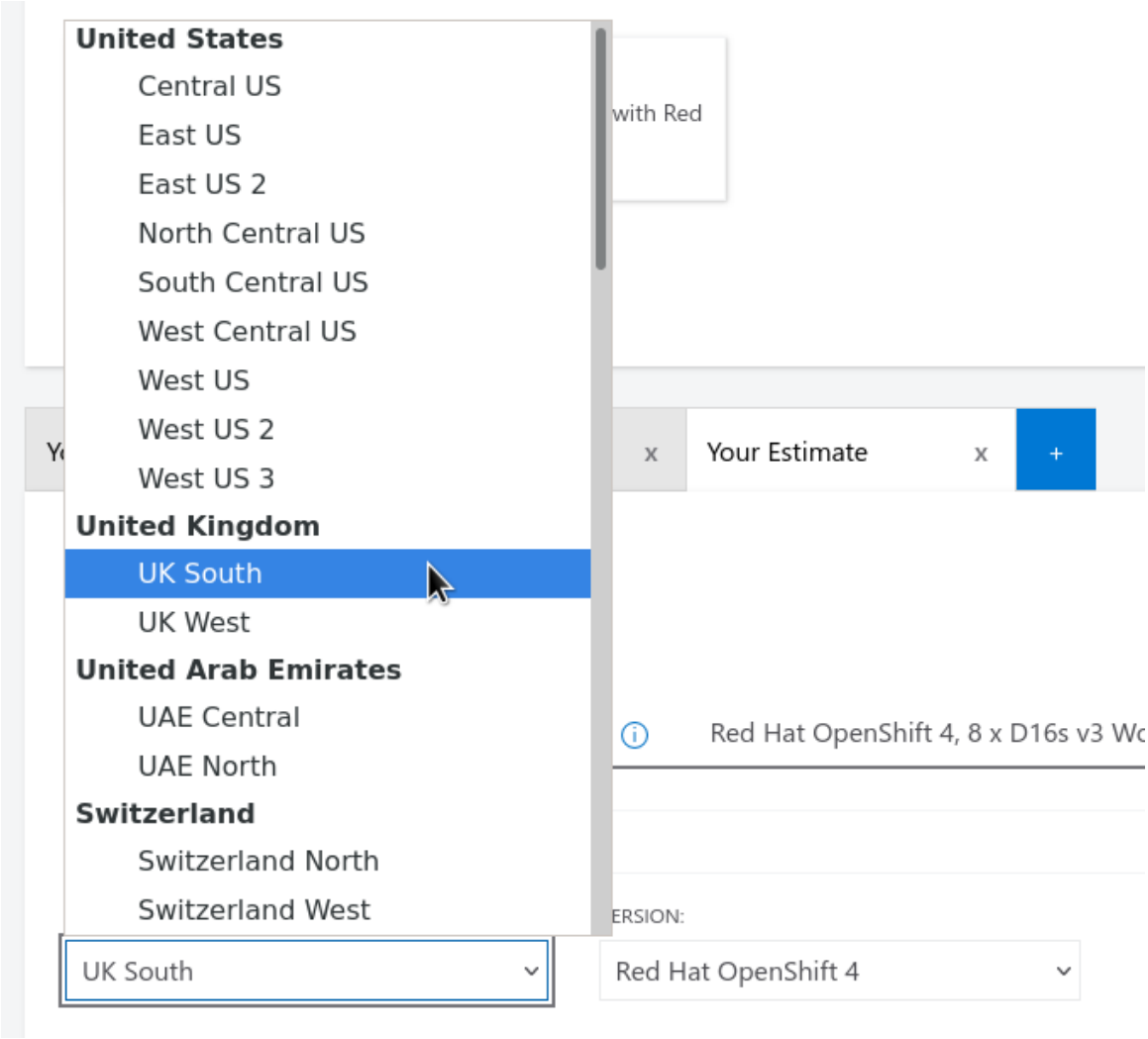


图 3.6：选择 Azure 地区

4. **Worker Nodes** 是您的实际应用将会运行的位置。在 **Savings Options** 下，您会看到两个部分。**License** 指的是 OpenShift 订阅费用，而 **Virtual Machine** 部分指的是运行集群所需的计算服务的费用。一个集群中支持的工作节点数量上限为 100。

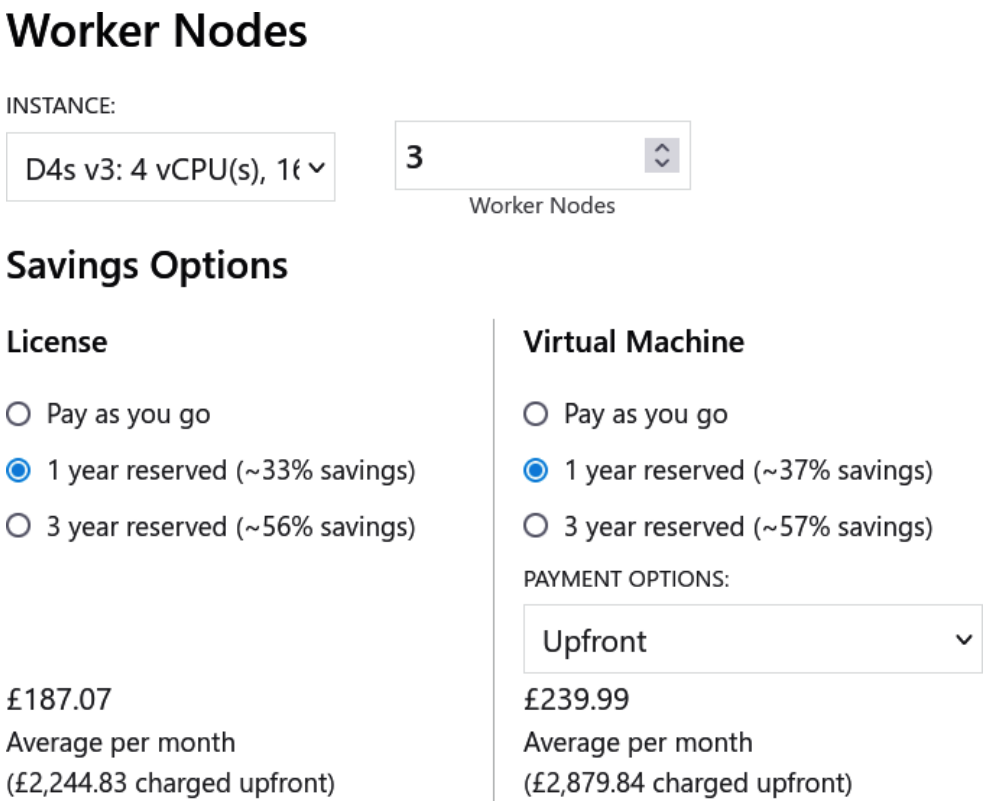


图 3.7：工作节点详细信息

5. **Managed OS Disks** 指的是红帽 CoreOS 用于操作系统分区的磁盘大小。这不是指应用持久卷所用的存储空间，后者会另外置备。

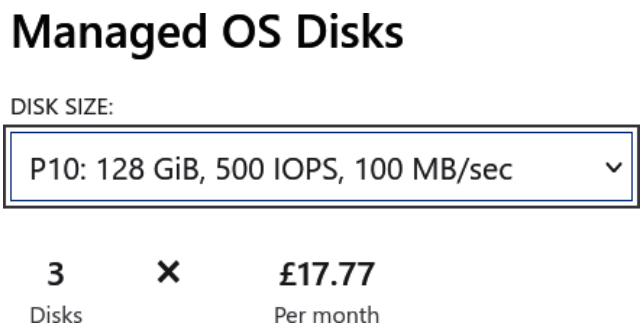


图 3.8：托管操作系统盘

6. 还有一个用于 **Master Nodes** 的类似部分，这在现代说法中通常称为控制节点。注意控制节点关联的磁盘比工作节点的大许多。这是因为，它们需要更高的 IOPS 和吞吐量，在 Azure 上意味着更大的磁盘大小。为了集群稳定，控制平面节点的确切数量必须始终为三个。

请注意，这个计算器旨在提供一个参考价格，实际价格显然会根据使用量而有不同。

Azure 预留虚拟机实例

预留实例指的是承诺长时间（1 年或 3 年）使用这一资源。Azure 红帽 OpenShift 虚拟机提供了预留实例选项。

如果组织知道某个集群将运行比较长的时间，他们通常会选择预留实例来获得 IaaS 的大幅折扣。例如，生产环境通常会使用预留实例来运行。值得注意的是，预留实例不会改变集群的服务级别或架构。

[Azure 预留实例的更多相关信息]

摘要

本章介绍了 Azure 红帽 OpenShift 托管云服务的相关细节。提供了其架构的高级概述，同时浅谈了与其他 Azure 服务的集成（我们会在*第 9 章：集成其他服务*）中更细致地探讨），以及管理、身份验证、支持和价格方面的注意事项。

下一章将重点介绍组织关注的 Azure 红帽 OpenShift 部署的置备前阶段问题和决策。

第 4 章

置备之前 – 企业架构问题

许多组织可在 Azure 门户看到 Azure 红帽 OpenShift，只要将相关文档通读一遍，就能成功地使用红帽 OpenShift 部署集群，几乎不需要额外付出。不过，倘若能多花一些时间做部署规划，并且提前弄清几个问题，就能在未来省下删除和重新置备集群的许多时间。

本章基于与许多客户合作的实操经验，讨论了应当先解决的许多典型的置备前问题。本章内容：

- 需要的集群数量，包括暂存和生产等集群
- 公来和私有网络可见性
- 混合型连接，例如与本地解决方案的连接

我们首先讨论如何计算出您需要的集群数量。

需要多少个集群?

OpenShift 有许多部署模式，但一个常见的问题是：“我的组织需要多少个集群？” 当然，具体决策视组织而异，但下一段落中的一些指导可帮助您得出这个数字。

生命周期阶段：开发、测试、生产

不论规模大小，大多数组织在部署企业 IT 系统时都会划分某种形式的生命周期阶段。这种方法有时也称为预演模式。三个最常见的阶段是开发、测试和生产。划分多个阶段后，对应用和应用部署的更改可以首先在安全的环境中进行测试，然后再部署到生产环境。最常见也是推荐的预演模式是设置至少三个独立的 Azure 红帽 OpenShift 集群：

- **开发**：所有开发人员和运维人员可以在这里测试任何对象。可以是一个大型“沙盒”集群，但更加普遍并且通常更加安全的做法是，拥有几个短期存活的小集群，在这些集群上频繁地创建和销毁测试。
- **测试**：即将实施的集群更改（如补丁或配置更改）在这里测试和验证，然后发布到生产环境中。一些组织也将这称为“预生产”，但预生产环境也可以是另一个独立的环境。
- **生产**：实际应用运行的地方。

除了上述环境外，一些组织还有其他环境，例如集成测试环境，但只有您自己知道多少个预演环境最适合您的组织。如果您不确定，可以通过研究其他类似的企业应用来了解常见的部署模式。

在 Azure 红帽 OpenShift 被用于非关键应用的场景中，也接受在组织中只设置两个集群（合并开发和测试），或者将开发、测试和生产全部囊括在一个集群中。其优势是能够节省云成本，并且减少需要管理的集群总数。从单个集群运行时，管理员可以选择对开发、测试和生产使用不同的命名空间。但是，仅运行一个集群有以下劣势：

- 影响整个集群的更改（如软件更新）可能会引发生产环境中的问题，而这些问题可以通过在测试环境中运行来探查和预防。
- 如果应用在测试或开发环境中出现意外行为，例如生成许多容器或者耗尽所有可用磁盘空间，它在生产环境中会造成问题。

本指南无法给出最适合您的场景的确切集群数量；正如前文所述，您可以通过研究组织中类似的企业应用来了解自己应该使用多少个生命周期阶段。

业务连续性、灾难恢复和故障转移

除了标准的暂存环境外，许多组织也会考虑创建至少一个故障转移生产环境。如果有灾难性故障使整个集群或 Azure 地区瘫痪，故障转移生产环境就可派上用场。这通常称为**灾难恢复（DR）**。通常，这会部署到与生产集群不同的 Azure 地区。

此托管云服务附带的 99.95% **服务级别协议（SLA）**可被许多业务关键应用接受。不过，一些应用可能需要更高的 SLA。重要的是要知道，仅凭一个集群是无法超越这一 SLA 的。思考您的应用是需要更高的服务级别协议，还是 99.95% 已经足够。

如果这还不够，您可以通过计算并行运行多个集群时的复合 SLA 来达到更高级别的服务可用性（例如 99.999%）。集群可以全部位于同一地区（如 `westeurope`），也可跨越多个区域（例如 `westeurope` 和 `northeurope`）来达到更高级别的可用性。以下 Azure 文档介绍了如何计算运行多个集群时的复合 SLA。

关于复合 SLA 的 Azure 文档

多集群和多地区部署 Azure 红帽 OpenShift 不在本书的讨论范畴。这是因为，在构建这些更为复杂的架构时，需要仔细考虑多个相关挑战，例如将流量导入集群中，以及选择在集群之间分享哪些数据和如何分享这些数据等。

地区和可用性区域

Azure 红帽 OpenShift 设计为可以利用部署所在的各个地区中的三个可用性区域。在 Azure 中，可用性区域是一个地区内的自治数据中心，拥有自己的电源、冷却和网络连接。如果您观察 Azure 红帽 OpenShift 部署，您会看到各个可用性区域中只有一个控制节点（虚拟机）和应用节点。

图 4.1 演示了控制节点和应用（工作）节点如何分散到一个 Azure 地区中的三个可用性区域。

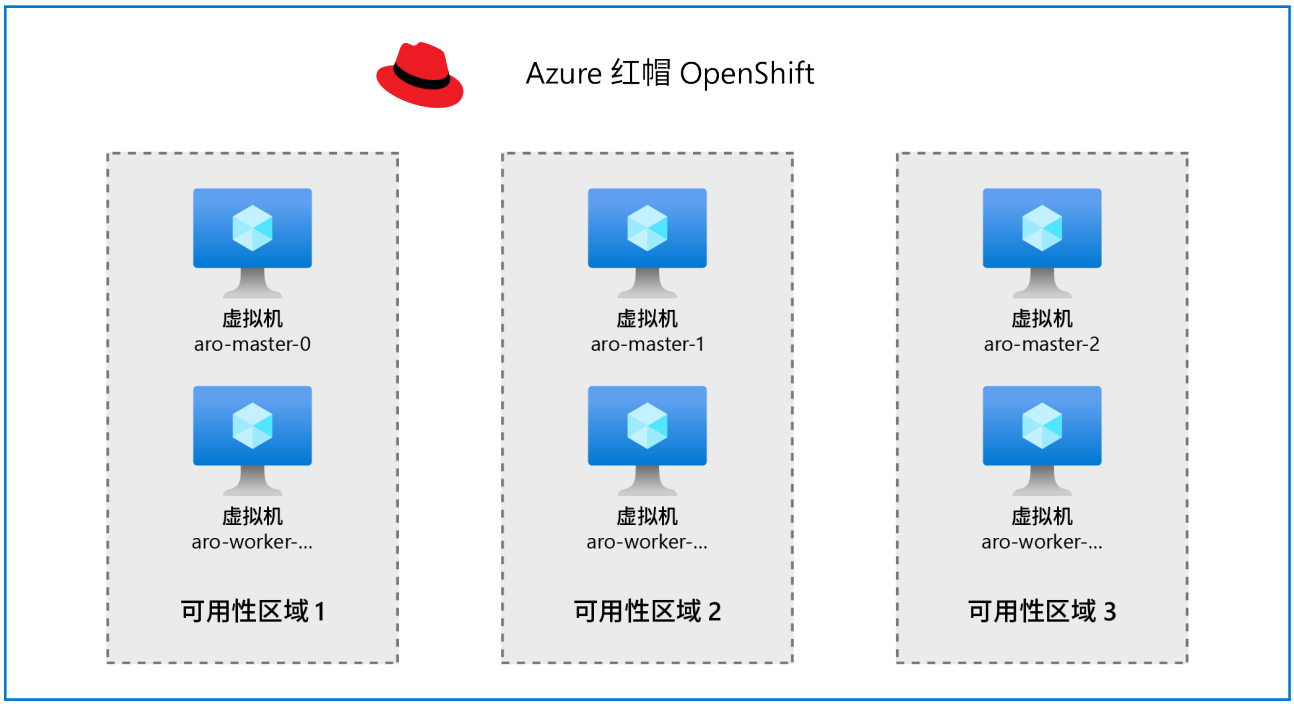


图 4.1：控制节点和应用节点在一个 Azure 地区中三个可用性区域之间的分布

Azure 红帽 OpenShift 设计为作为一款全托管、高可用的服务来供应。因此，部署的控制节点和工作节点的数量都不能少于三个。

网络概念

前面的 Azure 红帽 OpenShift 简介中提到了一个关于网络概念的优秀文档页面，可以在微软的文档网站上找到：

- Azure 红帽 OpenShift 的网络概念

这个页面上详细说明了 Azure 红帽 OpenShift 中的每个网络组件，如负载均衡器、公开和私有 IP 地址，以及网络安全组等。

需要了解的一些重点：

- Azure 红帽 OpenShift 可以部署到现有或全新的虚拟网络。仅支持一个虚拟网络，多个网络不会带来额外好处，因为 OpenShift 会在其基础上布设自己的**软件定义网络（SDN）**，名为 OVS。
- 主节点和应用节点子网的最小大小为 /27。
- 默认的容器集 CIDR 是 10.128.0.0/14。
- 默认的服务 CIDR 是 172.30.0.0/16。
- 每个节点分配一个 /23 子网（512 个 IP 地址）用于其容器集。这个值无法更改。
- 出口 IP 目前不受支持。
- 可以控制出口流量路由，指定为通过 Azure 防火墙转发。在撰写本文时，此功能处于公开预览阶段，相关说明可参见控制出口流量。

公开或私有网络可见性

您可能常听到 Azure 红帽 OpenShift 可以部署到公开或私有部署中。虽然这没错，但知道使控制平面保持公开 / 私有和使集群上应用保持公开 / 私有这两者之间的区别会有帮助。

对于 API 服务器可见性，您需要在置备之时做出决定；一旦置备了集群，就无法再调整公开 / 私有可见性了。

当您按照本章后文中所述创建 Azure 红帽 OpenShift 键时，您要运行 `az aro create` 命令，此命令接受几个集群可见性相关参数。以下示例演示了如何调整可见性：

两者均私有

```
az aro create .... --apiserver-visibility Private --ingress-visibility Private
```

私有 API 服务器和公开工作节点入口

```
az aro create .... --apiserver-visibility Private --ingress-visibility Public
```

apiserver 和应用入口的可见性映射到*图 4.2* 中的 Auzure 架构示意图。此图演示了 Azure 红帽 OpenShift 如何使用内部和公共 Azure 负载均衡器，其中的 API 和路由器根据所选的可见性选项来部署。

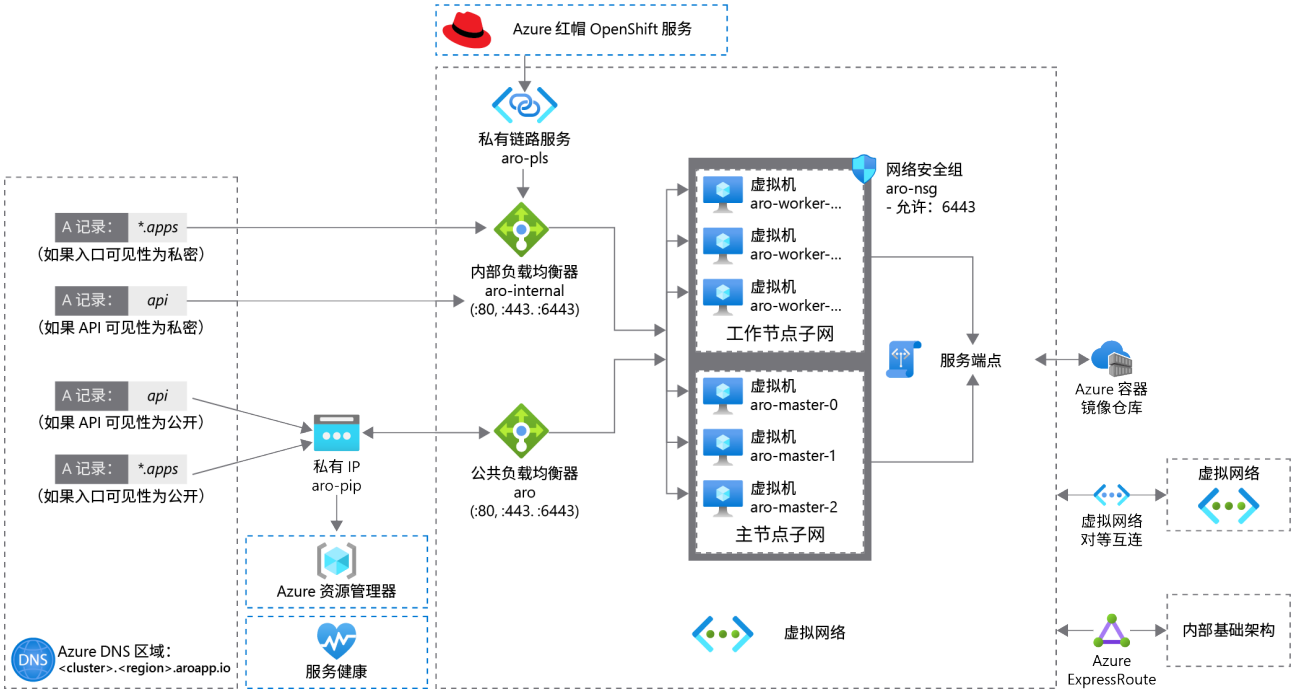


图 4.2：Azure 红帽 OpenShift 使用内部和公共 Azure 负载均衡器

有关 API 服务器可见性和入口可见性的更详细说明可在下文找到。

API 服务器可见性（控制平面）

`--apiserver-visibility` 可以是 **public**（公开）或 **private**（私有）：

- **私有**意味着运行 Kubernetes API 的红帽 OpenShift 控制平面（以前称为“主节点”）无法从公共互联网进行访问。此控制平面供开发人员和运维人员用于控制集群，可用来部署、删除或扩展应用及集群本身。如果企业具有 Azure 快速路由连接，或者使用**虚拟专用网络（VPN）**来访问计算资源，则大多数情形中应选择私有。
- **公开**意味着集群控制平面可以从公共互联网进行访问。这会使集群暴露于可被互联网上任何人攻击的风险中，即便访问要经过身份验证。不过，将此参数设为 public 对于您无法控制用户来源网络的实验或测试环境而言是有用的。在存储真实数据的环境中或者任何种类的生产环境中，强烈建议您将 API 服务器可见性参数设为 private。

以下列表提供了需要 API 服务器连接的一些示例，在选择公开还是私有时应当要考虑它们：

- 开发人员使用来自其 IDE 的脚本和工具（例如，kubectl rollout）
- 运维人员检查其集群的状态（例如，kubectl get nodes）
- CI/CD 服务器需要检查或调整部署状态（例如，Azure DevOps）
- 集群安全工具连接集群来检查其状态
- 监控工具依赖于 Kubernetes API

在许多情形中，网络可以配置为通过**私有**连接来接入，但上方列表中可能包括来自于您组织的其他示例，并且您可能会发现对于**公开** API 服务器可见性的意外要求。

入口可见性（应用）

`--ingress-visibility` 可以是公开或私有的：

- **私有**意味着开放的服务（与集群上运行的应用相关）不允许直接从公共互联网进行连接。但是，依然可以配置网络路由，让用户首先穿过 Azure 防火墙或在单独的 Azure 网络中运行的 Web 应用防火墙，然后连接您的应用。**私有**也适用于集群中的应用仅供组织内部使用的场景，例如薪资处理、数据分析或其他内部 Web 应用。
- **公开**意味着集群中的应用可通过公共互联网进行访问。不过，可能仍然需要配置入口或路由资源才能访问这些应用。如果您要在红帽 OpenShift 上托管对外公开的电子商务购物网站或其他应用，就适用这种情形。

入口有时也称为 OpenShift“路由器”。

生产环境建议

强烈建议：

- 通过私有连接访问集群，不要通过公共互联网。如果集群需要从本地网络或企业办公室进行持久连接，那么 Azure ExpressRoute 是最适合的选项。否则，通过 VPN 接入 Azure 也能提供私有连接。**混合连接**一节中可以找到更多与此相关的详细信息。
- 将 API 服务器可见性（控制平面）设置为 private，并通过防火墙进一步限制访问。
- 对于在组织内运行的应用，将入口可见性（应用）设置为 private。如果有需要将 Azure 红帽 OpenShift 中的应用托管到公共互联网的用例，请设置一个独立的 Azure 红帽 OpenShift 集群，至少一个集群用于内部应用，另外一个集群用于外部应用（将入口可见性设为 public）。不过，在同一个集群中混搭 public 和 private 入口也是可行的。

当然，许多组织会咨询他们的 Azure 网络和安全团队来确定可能需要的其他控制。 比如，一些组织可能要求将 Web 应用部署在 Web 应用防火墙的后方。这也是在 Azure 红帽 OpenShift 上部署应用的常见部署模式。

混合连接

部署 Azure 红帽 OpenShift 的许多组织需要将运行的应用连回到本地环境中运行的支持服务。从 Azure 连回到本地环境时，这通常称为混合连接或混合云架构。提供连回本地环境的连接有多种方式。最著名的两个选项是：

- **VPN**，适合通过低复杂度连接连入 Azure。通常，VPN 通过 Azure VPN 网关进行连接。如需更多信息，请参阅什么是 Azure VPN 网关?
- **Azure ExpressRoute 线路**，适合通过可靠、持久的专线连接连入 Azure。如需更多信息，请参阅什么是 Azure ExpressRoute？

不论采用哪种连接方式，两种解决方案都允许本地环境中的应用连接到在 Azure 红帽 OpenShift 上运行的应用，反之亦然。

从本地环境连接到 Azure 红帽 OpenShift 时，通常会通过集线器虚拟网络进行连接。*图 4.3* 中的示意图演示了如何通过 Azure ExpressRoute 进行连接：

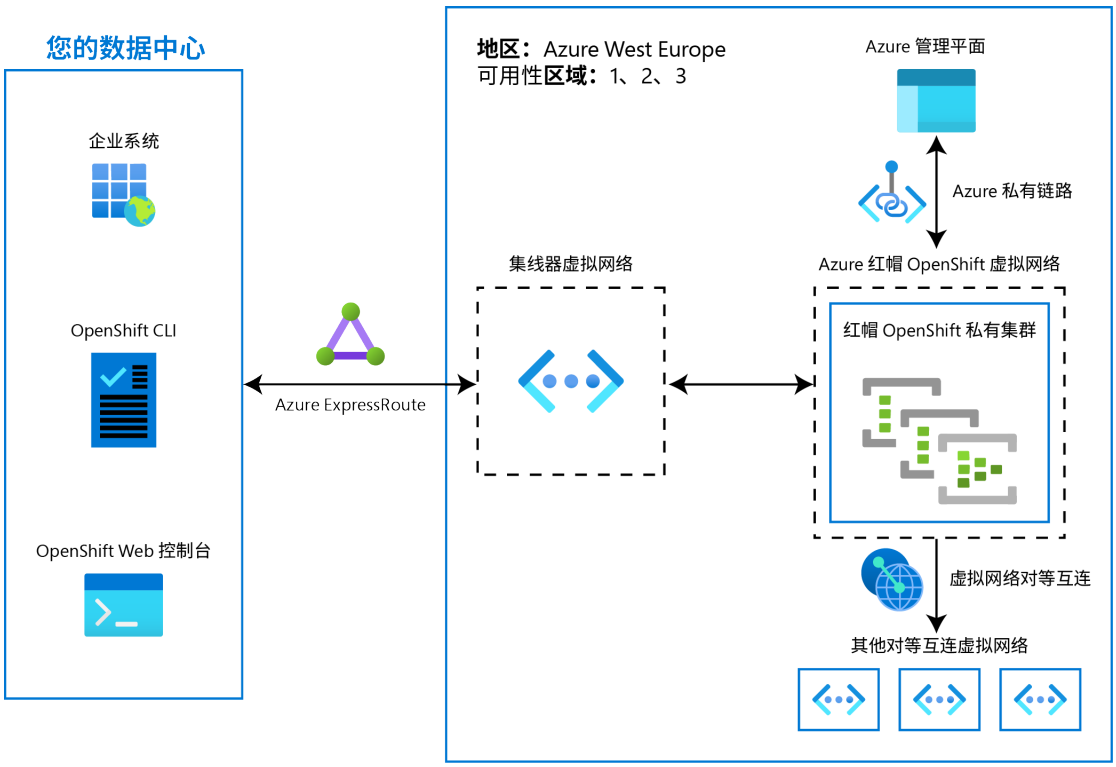


图 4.3：通过 Azure ExpressRoute 进行连接

上图演示了 Azure ExpressRoute 连接 Azure 红帽 OpenShift 的简要架构。图中演示的是 Azure ExpressRoute，但通过 VPN 连接在概念上也非常相似。

在混合架构中运行应用的注意事项

如果 Azure 红帽 OpenShift 中运行的应用连回到本地环境，应用所有者可能需要考虑如下注意事项：

- **连接延迟**：Azure ExpressRoute 提供延迟相对较低的连接来连回到本地环境，而穿越公共互联网的 VPN 网络的延迟可能明显高一些。对于连接只是用来传送 HTTP 流量的一些应用，例如 Web 服务器，这不太可能导致问题。但是，如果这一 Web 服务器上运行的服务器端应用需要连接在本地运行的数据库，这可能会给性能造成不小的影响。
- **入口 / 出口流量成本**：一些连接类型会对入口和出口流量收费，不论是经由 Azure 还是来自于连接提供商。明智的预防措施是首先在测试环境中衡量成本，然后估测在负载高峰时这些成本能有多高，从而确保安心使用。
- **故障情景**：Azure ExpressRoute 连接具有持久且可靠的设计，而 VPN 连接常常会遇到线路中断或频繁连接 / 断开问题。另外，由于通过公共互联网运行，VPN 连接的延迟和服务质量在一天当中可能会频繁变化。重要的是要测试应用性能，包括混合链路条件不佳时的性能，以及连接处于最佳状态时的性能。此外，如果连接中断长达几个小时，在 Azure 红帽 OpenShift 上运行的应用是能够以降级方式运行，还是其可用性完全取决于混合连接状况？

除了现有的内部检查清单，您的组织可能还需要检查一些其他要点，但前面几点应当足够您让开始思考混合架构是不是适合自己。

摘要

在本章中，我们讨论了诸多常见的置备前问题，您在部署 Azure 红帽 OpenShift 前需要详细了解和研究。下一章将列举一些实际部署集群时可以参考的实用资源。

第 5 章

置备 Azure 红帽 OpenShift 集群

我们来回顾一下目前已探讨的内容。到目前为止，您已探索了以下内容：

- 在“*第 2 章：红帽 OpenShift 简介*”中，我们简要介绍了红帽 OpenShift，也讲解了选择 OpenShift 相比于裸机 Kubernetes 的优势。
- 在“*第 3 章：Azure 红帽 OpenShift*”中，我们讨论了红帽 Azure 红帽 OpenShift 托管云服务的细节，也阐述了一些重要概念，例如其架构、管理、身份验证、支持和价格注意事项等。
- 在“*第 4 章：置备之前 – 企业架构问题*”中，我们讨论了组织在部署 Azure 红帽 OpenShift 前应当了解的常见问题。

现在，我们已准备好置备和部署集群了，您可以在文档网站上查阅官方的部署说明（置备是部署的一部分；置备一个部署意味着确保支持这一部署所需的 IT 基础架构均已就位）。

[教程 – 创建 Azure 红帽 OpenShift 4 集群]

本章不会阐述您在创建集群时需要输入的各个命令，因为这些命令可能会随时调整，而且相关文档中也有详细的说明。

不过，本章会就总体流程提供指导，同时也会介绍您在部署集群时需要考虑的事项。

手动部署 – 预期用时

一些组织可能希望了解置备一个集群需要花费多久时间。我们通过一份详细说明来探索这个过程。

部署前提条件的简要流程如下：

- 将 az 命令部署到系统管理员的工作站上
- 注册要与 Azure 订阅搭配使用的资源提供商
- 红帽拉取机密（可选）
- 用于您的集群的网域（可选）
- 一个虚拟网络和两个空白子网（两个子网分别用于控制平面节点和应用节点）

至于设置这些前提条件所需的时间，技能娴熟的 Azure 和红帽 OpenShift 管理员首次执行时或许能在 30 分钟内完成这些任务，如果在手动流程中反复执行这些步骤，那么用时可短至 10 分钟。

满足这些前提条件后，自动化置备流程通常需要 25 到 40 分钟，具体取决于 Azure 地区内的活动状况。

部署自动化

既然知道 Azure 红帽 OpenShift 的置备流程是自动化的，组织通常也会尝试使用工具来自动完成这些前提条件步骤，例如创建网络、子网和服务规则等。

有许多工具可以自动完成这些步骤。以下列表中提供了一些推荐的工具：

- az 命令行工具：进行自动化时，此工具通常安装在容器中，或者在 CI/CD 流程中进行类似的安装。这里可以使用的典型工具有 Jenkins、Azure DevOps 或 Ansible。请注意，az 命令行工具只需要部署一次，但额外的集群可能需要设置 Azure 订阅 ID 来反映组织的不同部分。
- 注册您要与 Azure 订阅搭配使用的资源提供商: 如前面所述，这是 az 命令行工具设置的一部分。
- 红帽拉取机密（可选）：红帽提供了一个正式支持的 REST API 来用于获取拉取机密，相关信息可在这篇文章中找到。
- 用于您的集群的网域（可选）：这取决于您是如何创建 DNS 记录的；但是，如果您使用 Azure DNS，那么 Terraform、Ansible 或其他流行的 Azure 自动化工具可以为您效劳。
- 一个虚拟网络和两个空白子网（两个子网分别用于控制平面（主）节点和应用（工作）节点）：这通常由流行的 Azure 自动化工具自动完成，Terraform、Ansible 或类似的工具可以为您创建这个网络和子网。

当前提条件步骤得以自动化后，其用时可从手动流程的 30 或 10 分钟缩减到一两分钟。集群部署是无法加快的（这始终需要耗费 25 到 40 分钟），但与本地部署相比，这实际上是一个非常快速的端到端部署流程。

除了加快流程外，部署自动化还有诸多优点，采用部署自动化的客户可以从构建可靠、并可重复的流程以方便记录和审计中获益。此外，极为常见的做法是客户将自助服务目录项构建到自己的门户中，让团队能够轻松地置备和撤销 Azure 红帽 OpenShift 集群，不必与云平台团队交互。

访问集群

本节为置备后如何访问 Azure 红帽 OpenShift 集群提供了一份简明参考。

通过 Web UI 访问

从 Bash shell（已安装 CLI 的情况下）或 Azure 门户中的 Azure Cloud Shell（Bash）会话运行以下命令，以检索您的集群登录 URL：

```
az aro show -n $CLUSTER_NAME -g $RG_NAME --query "consoleProfile.url" -o tsv
```

您应该会得到类似 openshift.xxxxxxxxxxxxxxxxxxx.eastus.aroapp.io 的信息。集群的登录 URL 将是 `https://` 加上 `consoleProfile.url` 值；例如，`https: //openshift.xxxxxxxxxxxxxxxxxxx.eastus.aroapp.io`。

在浏览器中打开这个 URL。您会被要求使用 kubeadmin 用户进行登录。使用安装程序在安装过程中提供给您的用户名和密码。

登录后，您应该可以看到 Azure 红帽 OpenShift Web 控制台。

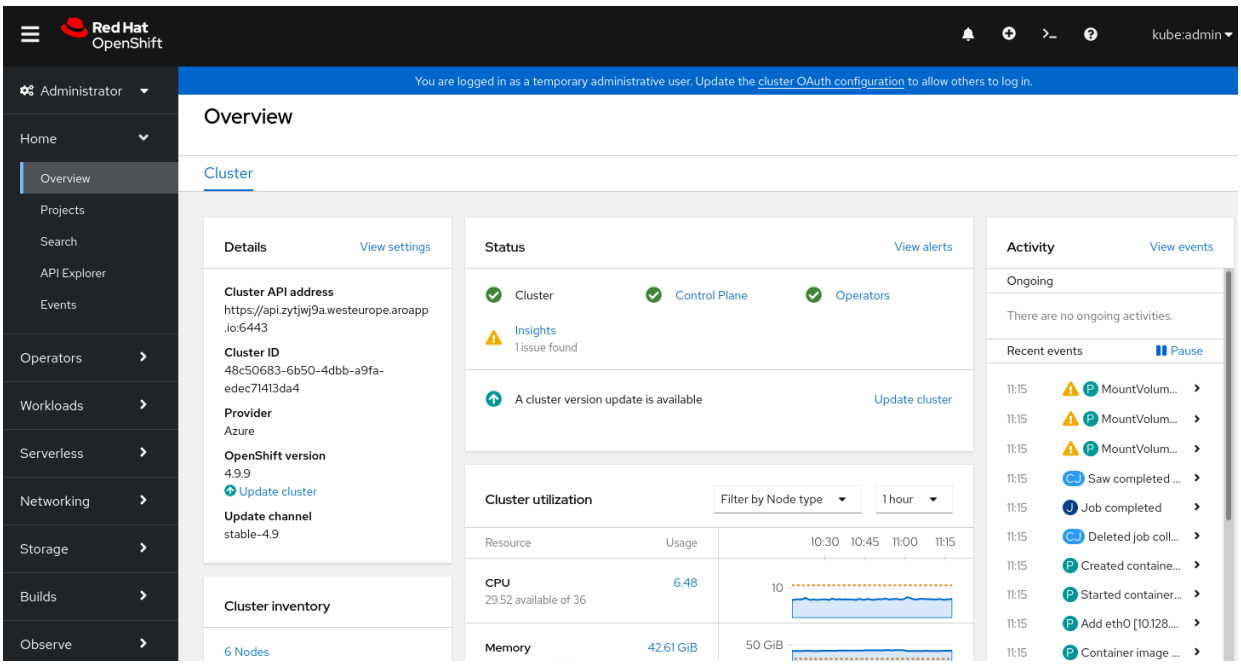


图 5.1：Azure 红帽 OpenShift Web 控制台

花些时间探索这个 Web 控制台。注意，它应当会运行最新版本的 OpenShift，而且所有组件应当处于健康状态，或者会在安装后很快进入健康的状态。

通过 OpenShift CLI（oc）访问

您需要下载最新的 OpenShift CLI（oc）。为此，只需登录并查看此页面：https://console.redhat.com/openshift/downloads。

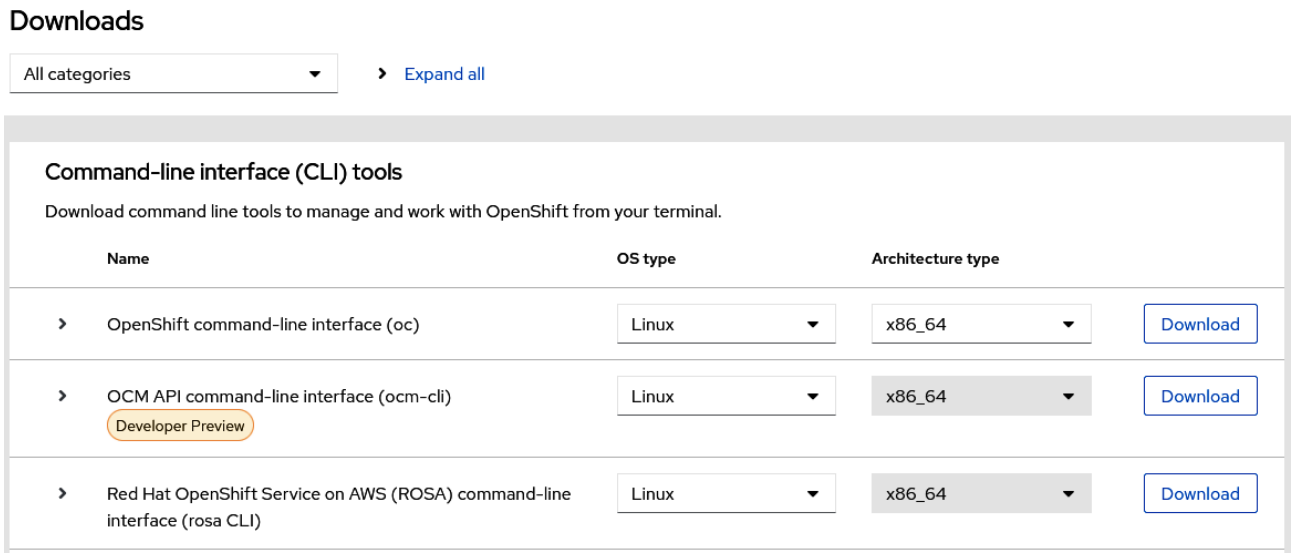


图 5.2：下载 OpenShift 命令行界面

将存档（.tar.gz 或 .zip）解压缩到系统上的某一位置，然后将 oc 命令放到您的路径上的某一处。在 Linux 上，通常会将 oc 命令放到 /usr/local/sbin/ 中。

运行 OpenShift CLI 并登录您的集群

要从命令行对您的集群进行身份验证，您需要从 Web 控制台检索登录命令和令牌。在浏览器中登录 Web 控制台，再单击右上方的用户名，然后单击 **Copy login command**。

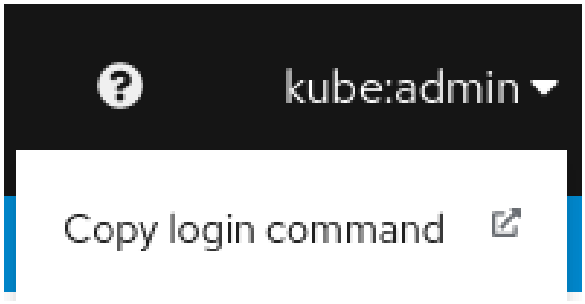


图 5.3：复制登录命令

这会打开一个类似如下的新页面：

图 5.4：使用 API 令牌进行登录

然后，您可以将此命令复制并粘贴到终端中，以登录您的 Azure 红帽 OpenShift 集群。

例如，如果您使用 Azure 门户中的 Azure Cloud Shell （Bash）会话，可以粘贴这个登录命令，然后 `oc status` 命令应该会返回如下内容：

```
user@Azure: oc status
In project default on server https://api.cyki1k6g.westeurope.aroapp.io:6443

svc/openshift - kubernetes.default.svc.cluster.local
svc/kubernetes - 172.30.0.1:443 -> 6443

View details with 'oc describe <resource>/<name>' or list everything with 'oc get all'.
```

现在，您可以花些时间进一步探索 oc 命令行，并熟悉您的新 Azure 红帽 OpenShift 环境。如果您是有经验的 OpenShift 4 用户，Azure 红帽 OpenShift 的这种云服务应该会给您非常熟悉的感觉，其行为与您在过去使用的其他 OpenShift 4 环境完全一样。

摘要

这是一个非常简明扼要的章节，因为官方的置备说明维护得非常完善，应当视为在 Azure 订阅中运行 Azure 红帽 OpenShift 的权威指南。您会发现，这些置备说明与置备自助管理 OpenShift 环境的说明相比要简单得多。这是因为，Azure 红帽 OpenShift 服务经过了精心设计，它能提供与 Azure 的紧密集成，并且能部署经过完备测试并享有完善支持的规范的“通用型”架构。总体而言，组织可以得益于花费更少时间置备 Azure 红帽 OpenShift，将更多时间用在部署应用上。

第 6 章

置备之后 – Day 2

在完成 Azure 红帽 OpenShift 部署后，通常需要进行一些置备后活动，然后集群才能准备好投入生产环境。本章阐述了许多常见的置备后活动，具体如下：

- 身份验证 – Azure Active Directory – 包括如何与社区 operator 同步用户组
- Operator – 包括 OperatorHub
- 了解日志记录 – 包括日志转发
- 了解监控 – 包括监控 OpenShift 容器
- 升级和补丁 – 包括支持的版本生命周期
- 集群扩展 – 包括手动和自动扩展集群
- 应用扩展 – 关于应用扩展的简略说明
- 设置拉取机密 – 注册到 OpenShift 集群管理器
- 限制范围 – 包括可在哪里应用限制范围
- 持久存储 – 包括可用和受支持的存储类
- 安全与合规 – 关于安全控制的说明

这些小节各自详细阐述不同的置备后任务，帮助您进一步了解需要些什么才能将全新部署的集群准备好用于生产应用。

身份验证 – Azure Active Directory

Azure 红帽 OpenShift 支持 OpenShift 文档中列出的所有身份验证提供商；请查阅完整的身份验证提供商列表。不过，大部分客户喜欢使用 Azure 上已经可用的 Azure Active Directory，为 Azure 红帽 OpenShift 提供身份验证和单点登录。

Azure Active Directory 集成的配置有意划分为“Day 2”运维，因为可能会有组织想要使用其他配置提供商的情况。至于 Azure Active Directory 的设置和安装过程，第一次设置时用时大约 15 分钟，以后更加熟悉这个流程后，只需短短五分钟就能完成整个过程。许多组织选择使用 ARM 模板或 Ansible Playbook 等工具来自动完成这一部分的置备。

- 为 Azure 红帽 OpenShift 配置 Azure Active Directory（图形门户）
- 为 Azure 红帽 OpenShift 配置 Azure Active Directory（命令行界面）

使用 Active Directory 用户组

Azure Active Directory 身份验证的配置仅允许用户使用现有的凭据来登录。它不会将用户的现有用户组传递到红帽 OpenShift 中。比较常见的情形是，组织希望导入 Active Directory 中的用户组，以便使用它们在 OpenShift 中配置权限。这可以通过一个由社区支持的独立 Operator 来达成，其名为 Group Sync Operator。

- GitHub 上的 Group Sync Operator

注意 Group Sync Operator 是由社区支持的；也就是说，在撰写本文时红帽或微软均不为此提供官方支持。本指南建议您遵循相关项目的 README 文件中附带的 operator 详细配置和安装说明。

Operator

OpenShift 4 中的 Operator 是这个平台的基本构建组件之一，为红帽 OpenShift 的使用者提供了巨大价值。Operator 用代码构建的，在集群的容器中运行服务。一些 Operator 负责维护事务，例如集群联网、计算机配置维护和集群升级等。这通常称为集群 Operator，您可以在 OpenShift 控制台的 **Administration** 部分中查看其列表。

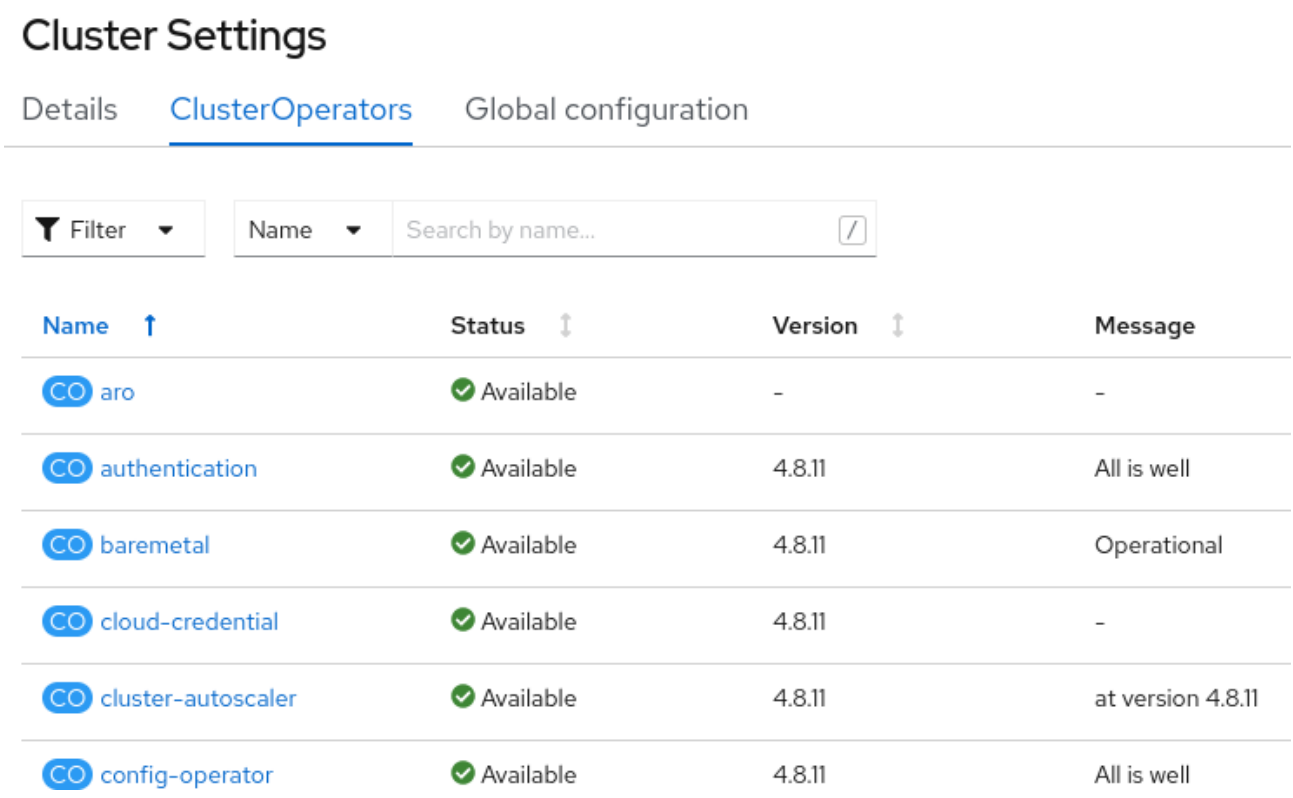


图 6.1：集群设置

在前面的屏幕截图中，甚至还能看到一个专用于 Azure 红帽 OpenShift 的 Operator，它将维护该服务的组成部分，并且尽力确保集群保持在受支持的配置状态。

Operator 可以维护许多对象，例如服务健康状态、更新、补丁、扩展和诸多其他功能。

OperatorHub

除了在每个集群上后台运行的标准集群 Operator 外，管理员可以通过 OperatorHub 访问更多其他 Operator。通过 OperatorHub，管理员可以轻松找到流行的社区和企业 Operator，并提供给红帽 OpenShift 安装。OperatorHub 可以在每个 OpenShift 集群的边栏中找到。

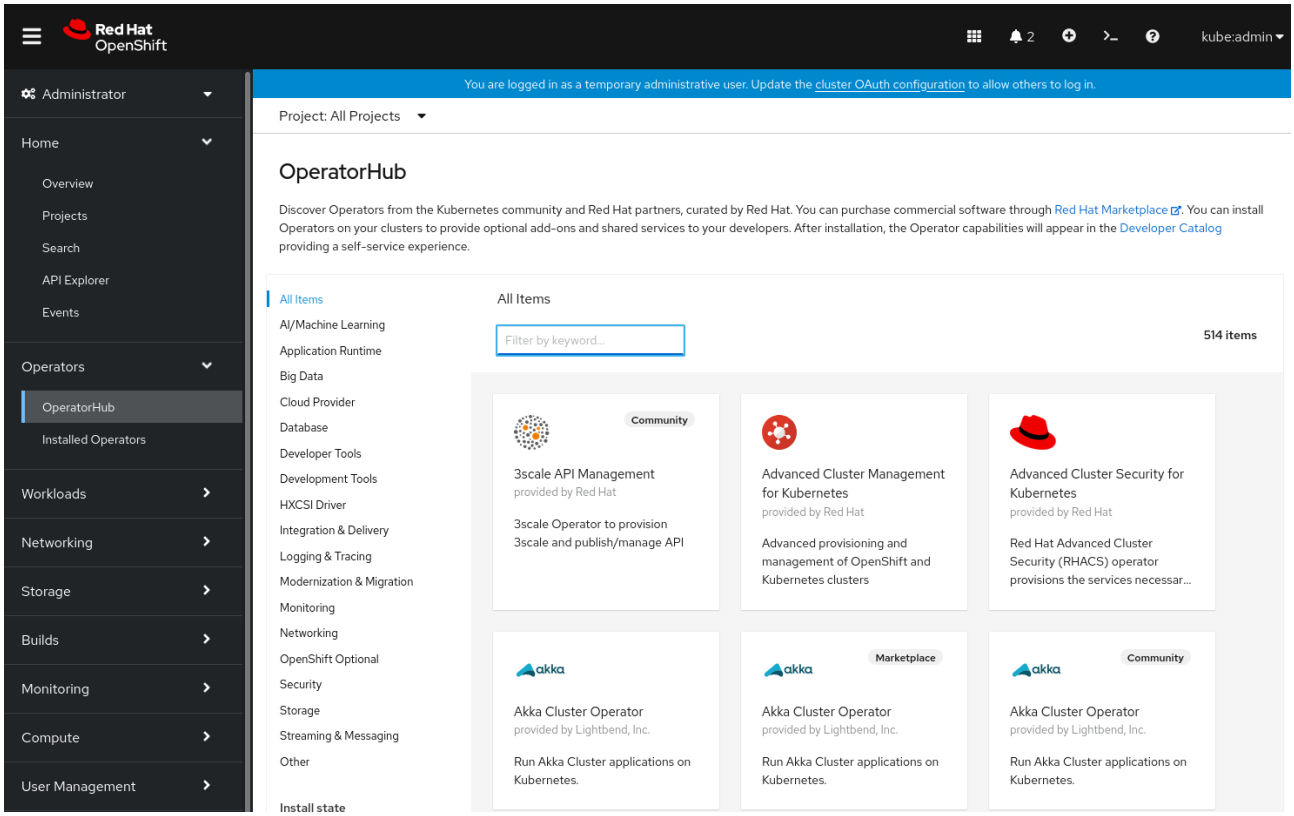


图 6.2：OperatorHub

通过使用 operator，管理员可以轻松、快速地部署和维护常见的软件，无需为 Kubernetes 配置和代码操心。许多红帽产品如今已打包成 operator，例如高级集群管理、红帽 OpenShift 服务网格和红帽 OpenShift Pipelines。不过，也有面向非红帽软件的庞大 operator 资源库，例如 MariaDB 数据库、Couchbase 或 IBM 块存储驱动程序。

如需关于 operator 的更多信息，建议您访问以下链接：

- Operatorhub.io
- OpenShift 中的 Operator

了解日志记录

Azure 红帽 OpenShift 使用与红帽 OpenShift 相同的日志和监控架构。这两项服务使用相同的 operator 来记录日志。

日志划分为以下三个类别：

- **应用**：由集群中运行的用户应用生成的容器日志，但基础架构容器应用除外。
- **基础架构**：由集群中运行的基础架构组件以及 OpenShift 容器平台节点生成的日志，如日志记录。基础架构组件是在 openshift*、kube* 或默认项目中运行的容器集。
- **审计**：由节点审计系统 auditd 生成的日志，存储在 /var/log/audit/audit.log 文件中，以及来自 Kubernetes API 服务器和 OpenShift API 服务器的审计日志。

如需有关 OpenShift 日志的更详细说明，请参阅产品文档中的了解红帽 OpenShift 日志。

使用转发集群日志到 Azure Monitor 的功能

一个常用的集成是将 Azure 红帽 OpenShift 日志发送到 Azure Monitor 的容器智能分析服务。有时，这也称为 Azure 日志分析。这可以实现以较低代价将日志持久保留在 Azure 中。

OpenShift 日志架构在每个节点上运行 Fluent Bit，operator 首先采取的做法可能是调整这项 Fluent Bit 服务的配置，以便直接发送日志。但是，调整 Azure 红帽 OpenShift 中的日志配置已超出支持政策范围。OpenShift 具有一种内建机制，既可以转发日志，又能继续留在支持范围内，其名为 `ClusterLogForwarder`。

通过使用 `ClusterLogForwarder`，日志可被转发到 Elasticsearch、Flutentd 和 syslog；不过，Azure Monitor 不直接支持任何这些协议。但有一种变通办法，用额外的中间 Fluent Bit 实例从 OpenShift 接收日志并转发到 Azure Monitor。此办法目前已在微软文档和社区文档中有相关说明。

了解监控

作为一项托管服务，应该不需要为 Azure 红帽 OpenShift 实施复杂的自定义监控，因为这已包含在您的付费服务之中。

不过，客户通常希望使用 Azure 容器智能分析来监控 OpenShift 容器。这项功能目前处于公开预览阶段，相关说明可参见此文档。

升级和补丁

在置备 Azure 红帽 OpenShift 集群时，您不会选定更新渠道。这意味着，您的集群默认为不开始接收更新。

要查看您的更新渠道，请前往导航侧边栏中的 **Administration** → **Cluster Settings**。下图中显示了 Azure 红帽 OpenShift 集群置备不久后的集群设置：

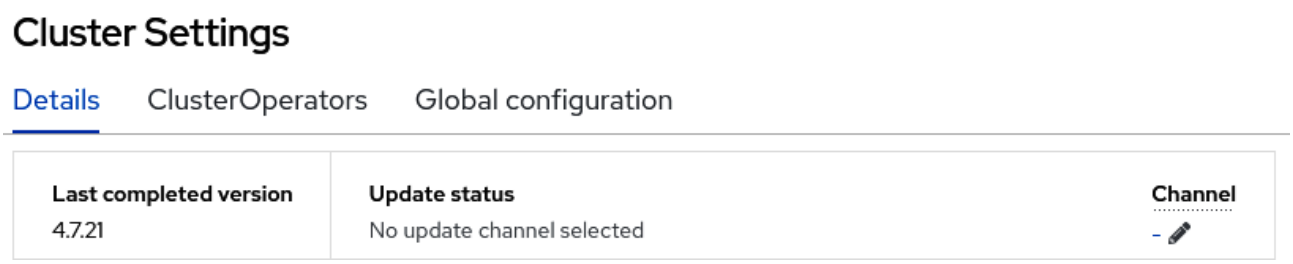


图 6.3：集群置备之后的集群设置

要选择更新渠道，请选择“-”链接，再查看可用的选项：

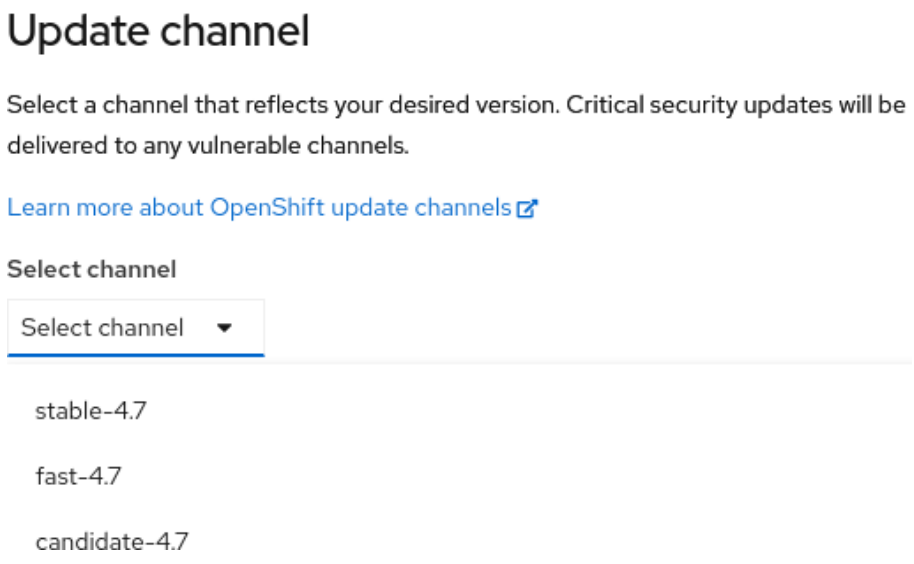


图 6.4：选择更新渠道

要了解 stable、fast 和 candidate 之间的区别，请查看升级渠道和发布文档页面。

强烈建议您让生产环境中的所有集群在 **stable** 渠道运行。

支持版本生命周期

在规划生产部署时，务必要知道哪些版本的 Azure 红帽 OpenShift 是受到支持的。官方的生命周期页面上提供了相关信息，可帮助组织了解受支持和不受支持的版本分别有哪些。

Azure 红帽 OpenShift 支持生命周期页面

以下是该页面信息经过提炼后的精简内容：

Azure 红帽 OpenShift 支持红帽 OpenShift 容器平台的两个**正式发布（GA）**次要版本：

- Azure 红帽 OpenShift 中发布的最新 GA 次要版本（我们称为 N）
- 前一个次要版本（N-1）

红帽 OpenShift 容器平台使用语义化版本控制。语义化版本控制使用不同等级的版本号来指定版本的不同级别。下表展示了语义化版本号的不同组成部分；本例中使用了示例版本号 4.9.3：

主要版本（x）	次要版本（y）	补丁（z）
4	9	3

版本中的每个数字指明与上一版本的一般兼容性：

- **主要版本**：目前没有主要版本发布计划。主要版本在出现不兼容的 API 更改或向后兼容中断时会发生变更。
- **次要版本**：大约每三个月发布一次。次要版本升级包括功能新增、增强、弃用、移除、错误修复和安全改进。
- **补丁**：通常每个星期发布一次，或者按需发布。补丁版本升级可以包括错误修复和安全增强。

如需更全面地了解支持的版本，请阅读 Azure 红帽 OpenShift 支持生命周期页面。

集群扩展

红帽 OpenShift 容器平台以及 Azure 红帽 OpenShift 在设计时便考虑了可扩展架构。在 OpenShift 范畴内规划扩展时，管理员和应用所有者通常需要将集群扩展和应用扩展作为两个独立主题来考虑。

本节阐述了集群扩展，另外一个小节中则简要说明了应用扩展。

支持的上限

所谓集群扩展，指的是在集群中添加额外的工作节点，从而为集群中运行的应用增加额外的计算容量。当然，也会引出何时需要扩展控制平面的问题。Azure 红帽 OpenShift 已经部署了三个控制平面节点，其容量基本上足以扩展至 Azure 红帽 OpenShift 集群所支持的最大工作节点数，目前是 60。

在撰写本指南时，控制平面节点支持三种实例大小：`Standard_D8s_v3`、`Standard_D16s_v3` 和 `Standard_D16s_v3`。

工作节点还支持更多 Azure 实例类型，如计算优化（F 系列）、内存优化（E 系列）和常规用途（D 系列）等类型。

如需 Azure 红帽 OpenShift 目前支持的 Azure 实例类型列表，可参见支持政策页面。

最小部署和缩容为零

组织有时候可能希望部署较小的集群来满足测试和开发需要，或者用于可接受可用性缩减的其他用例。目前，最小集群大小是三个控制平面节点和三个应用节点，不支持收缩到更小的规模。

手动扩展集群

运维人员在查看 Azure 门户时可能会禁不住尝试通过直接创建 Azure 虚拟机，向 Azure 红帽 OpenShift 集群添加更多工作节点。但这是不可行的，因为含有 Azure 红帽 OpenShift 的资源组从 Azure 管理员角度来看“已被锁定”。不论权限高低都是如此，即便具有 **owner** 权限的用户也无法修改 Azure 红帽 OpenShift 资源组的内容。因此，如果您试图在 Azure 红帽 OpenShift 资源组中手动创建虚拟机，您会遇到权限遭拒错误。

专用的 Azure 红帽 OpenShift 扩展方法是使用 OpenShift `MachineSets` 功能，OpenShift 会在收到指示时部署新的应用节点。具有 cluster-admin 权限的用户可以在 **Compute** 下看到 **MachineSets**。

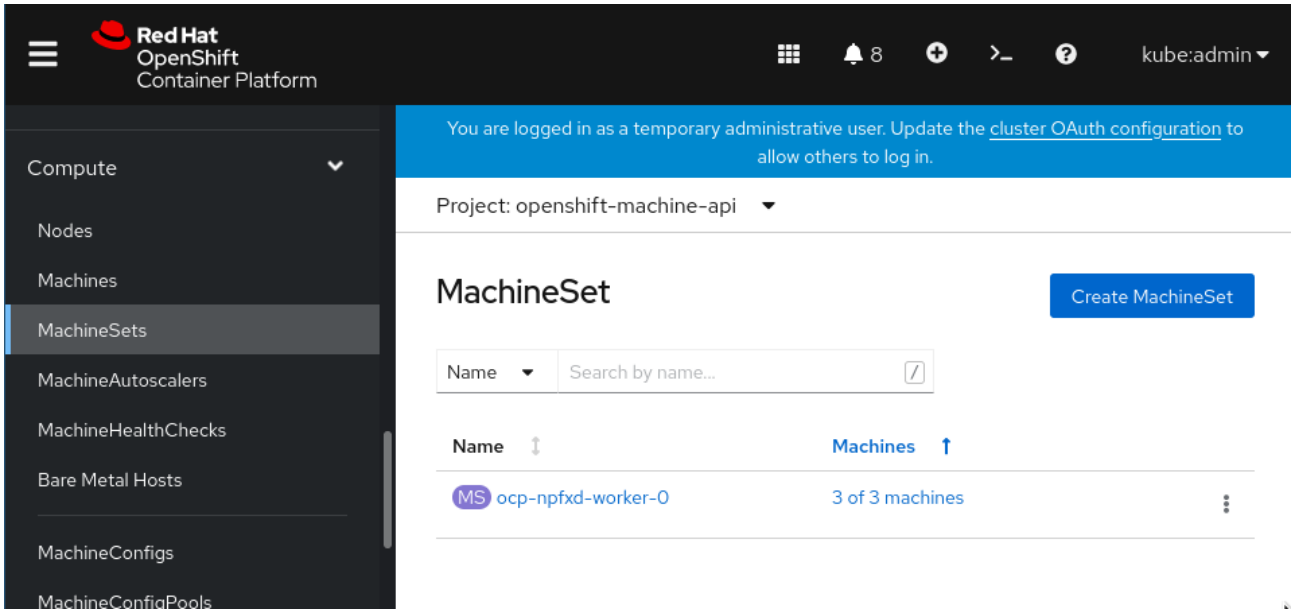


图 6.5：管理员访问 MachineSets

单击 `MachineSet` 应用节点的“edit”菜单，您会发现，只需选择 **Edit Machine Count** 就能轻松地置备新的应用节点虚拟机。

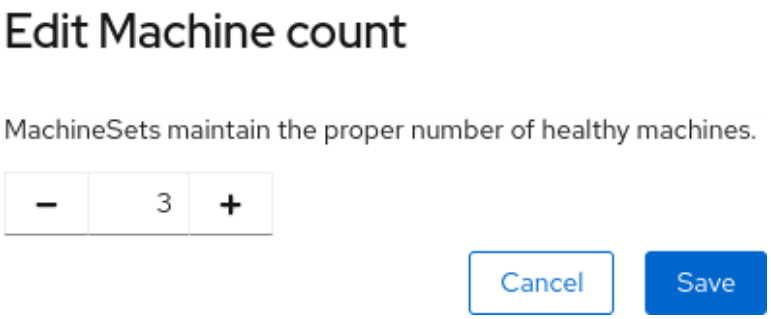


图 6.6：编辑虚拟机数量

更新数量后，管理员前往 Azure 门户或 **Compute** → **Machines** 视图，即可看到正在置备新的应用节点虚拟机。

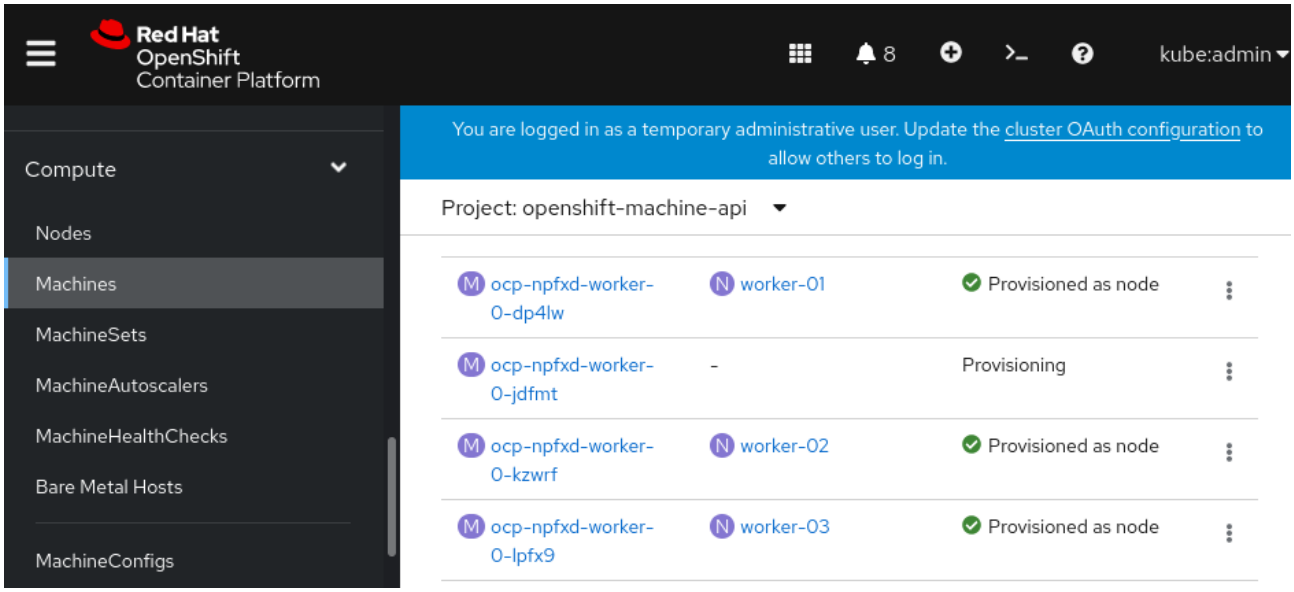


图 6.7：Machines 视图

在大多数 Azure 地区，置备一台额外虚拟机通常最多需要大约 5 分钟。

管理员无需完成任何置备后活动，就可让这一新虚拟机上线运行。OpenShift 会自动将它提供给集群，应用也会在需要时加以利用。

集群自动扩展

Azure 红帽 OpenShift 默认部署的配置没有开启自动扩展功能，但启用此功能也十分简单。管理员只需创建 `MachineAutoscaler` 资源便可，该选项可在 Azure 红帽 OpenShift **Compute** 边栏菜单中找到。

MachineAutoscaler 资源在 MachineSet 上运行，后者会根据需要创建或删除应用节点虚拟机容量。以下示例演示了可以在 MachineSet 中维护最小或最大虚拟机数量：

```
apiVersion: autoscaling.openshift.io/v1beta1
kind: MachineAutoscaler
metadata:
    name: worker-us-east-1a
    namespace: openshift-machine-api
spec:
    minReplicas: 1
    maxReplicas: 12
    scaleTargetRef:
        apiVersion: machine.openshift.io/v1beta1
        kind: MachineSet
        name: worker
```

OpenShift 还有集群自动扩展器的概念，可以通过提供一定数量的 RAM 或 CPU 等类似资源来进行扩展。根据您希望如何在集群中添加和移除容量来选择 MachineAutoscaler 或 ClusterAutoscaler。

关于 MachineAutoscaler 和 ClusterAutoscaler 的红帽 OpenShift 文档

应用扩展

扩展微服务应用是一个复杂的主题，不在本指南的讨论范畴。不过，您可以利用这里的两个实用页面链接来开始探索这个主题：

- HorizontalPodAutoscaler：指定您要运行的容器集的最小和最大数量，以及您的容器集的目标 CPU 利用率和内存利用率。
- 借助 Knative 实施无服务器和缩容为零。

集群会监控集群上运行的容器和剩余的容量。如果 Knative 或 HorizontalPodAutoscaler 请求的资源超过集群中目前可用的数量，那么需要进行 ClusterAutoscaler 或 MachineSet 扩展来将请求的资源添加到集群中。

借助这种方法，应用和集群本身可以根据需要相继扩展和收缩。

设置拉取机密 （注册到 OpenShift 集群管理器）

全新部署的 Azure 红帽 OpenShift 没有为 cloud.redhat.com 配置“拉取机密”。这意味着，在默认情况下，您的集群不会出现在名为 OpenShift 集群管理器的红帽混合云控制台（http://console.redhat.com）中。

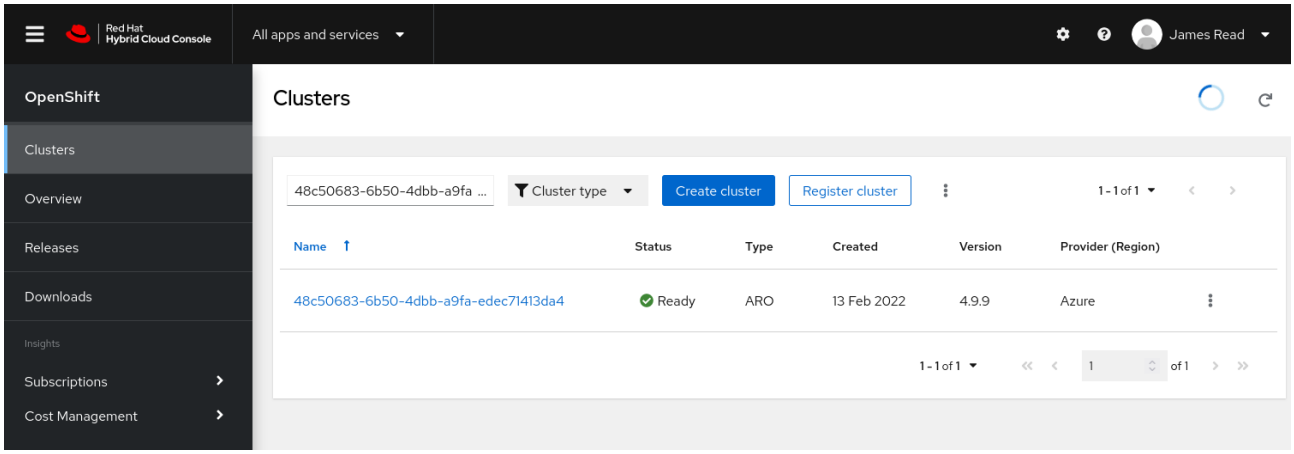


图 6.8：OpenShift 集群管理器中显示 Azure 红帽 OpenShift 集群

为 cloud.redhat.com 配置拉取机密非常简单，您也可轻松地使集群显示在 OpenShift 集群管理器门户（http://console.redhat.com）中，然后您就能直接通过标准的支持工单系统向红帽提交支持工单了。

如需如何设置拉取机密的说明，请参见这里：

- 如何添加或更新拉取机密

设置拉取机密还有一个好处，客户可以直接通过红帽提交支持工单。拉取机密会给您的红帽帐户授予权限，让您的集群可被支持人员看到。为 Azure 红帽 OpenShift 开具支持工单的步骤与任何其他红帽产品都一样。

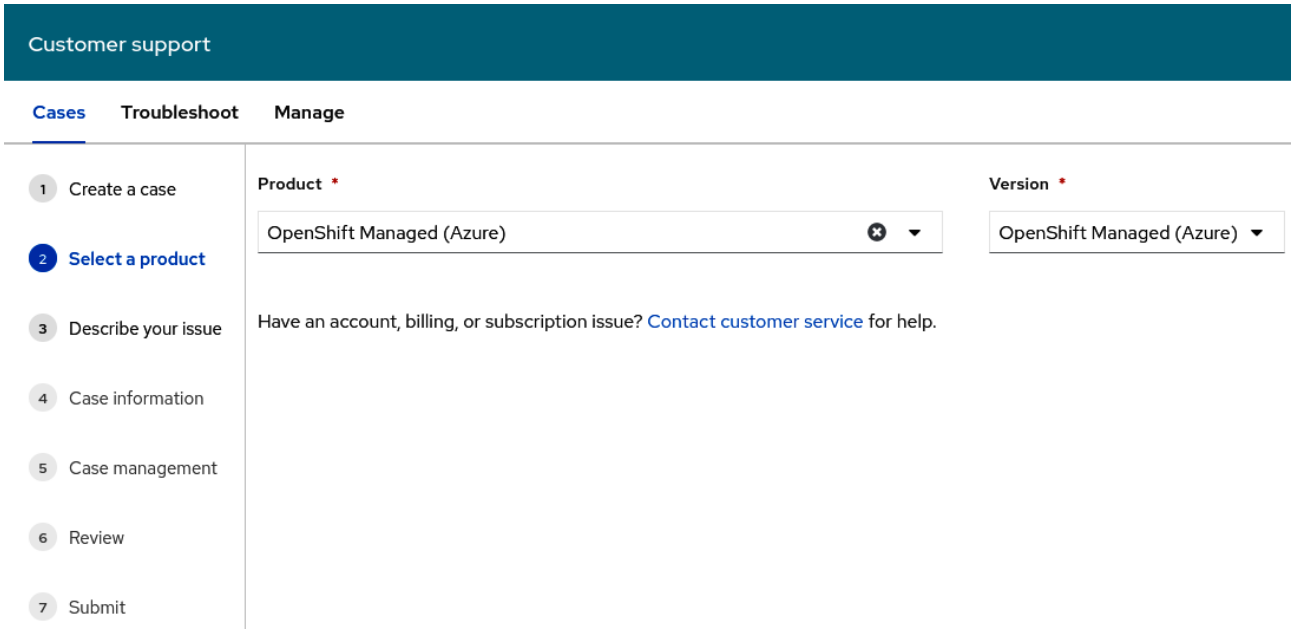


图 6.9：创建支持案例

限值范围

当开发或应用团队开始部署容器化应用后，只要过了一定时间，就会发生某一应用“行为不当”并开始不必要地消耗集群上的资源。例如，故障应用出现了内存泄漏问题，或者应用错误地设置为在消耗了仅仅 10% CPU（而不是 100%）后就开始扩展。若要预防行为不当的应用不受控地扩展并消耗太多资源，建议您使用 `LimitRange`。

借助 `LimitRange`，您可以限制项目中特定对象的资源消耗。`LimitRange` 可以应用到：

- 容器集和容器：您可以为容器集及其容器设置最低及最高 CPU 和内存要求。
- 镜像流：您可以设置 ImageStream 对象中镜像和标签数量的限值。
- 镜像：您可以限制可以推送到内部镜像仓库的镜像大小。
- **持久卷声明（PVC）**：您可以设置可请求的 PVC 大小。

下方示例中显示了一个应用到容器的限值范围，它限制了允许的最小、最大及默认 CPU 和内存请求：

```
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "resource-limits"
spec:
  limits:
    - type: "Container"
      max:
        cpu: "2"
        memory: "1Gi"
      min:
        cpu: "100m"
        memory: "4Mi"
      default:
        cpu: "300m"
        memory: "200Mi"
      defaultRequest:
        cpu: "200m"
        memory: "100Mi"
      maxLimitRequestRatio:
        cpu: "10"
```

要应用此 `LimitRange` 代码片段，您可以直接将 YAML 复制并粘贴到红帽 OpenShift 控制台的编辑器中，或通过 `oc apply -f limitrange.yaml` 从文件应用。

限值范围文档

持久存储

Azure 红帽 OpenShift 部署的虚拟机附带有 Azure 磁盘，以用于安装红帽 CoreOS 和运行 Azure 红帽 OpenShift 服务。这些磁盘不应被应用使用，仅可用于 OpenShift 集群本身。

需要持久存储的应用应当使用 Kubernetes `PersistentVolume` 功能，OpenShift 文档中提供了名为[了解持久存储]的优秀页面来详细说明这一概念。作为一种自助管理式 OpenShift 安装，Azure 红帽 OpenShift 在技术上支持所有的 `PersistentVolume` 提供商，但 Azure 上最常用的持久存储如下所示：

姓名	字体	访问模式	存储类
Azure 文件	文件系统，不符合 POSIX	ReadWriteOnce	Azure 文件
Azure 磁盘	块	ReadWriteOnce	Azure 磁盘
红帽 OpenShift Data Foundation	文件系统、块、对象	（多种）	OCS/ODF 文档

安全与合规

红帽 OpenShift 文档现在包含了相当多的相关内容，帮助您了解如何实施诸多安全控制并保持合规。

[OpenShift 安全与合规文档]

文档的这一小节包括：

- OpenShift 中容器安全防护运作方式的详细说明。务必要了解，OpenShift 为容器提供了许多开箱即用的安全控制，这些控制是其他基于 Kubernetes 的服务所没有的，它们有助于让您的组织和应用保持安全。
- 扫描容器集漏洞
- 访问审计日志
- 配置证书

另外还包括几个实用的安全相关 operator，例如：

- **合规性 Operator**：运行扫描并提供常见安全问题的修复建议。
- **文件完整性检查**：持续检查文件，特别是安全配置文件，确认它们没有被更改。

摘要

在这一章中，我们探讨了组织在调整全新部署的集群以供生产应用使用时通常要考虑的大部分任务和主题。尽管并非以后每次部署都需要逐一考虑所有这些主题，您在部署第一个集群时应当要考虑持久存储、限值、身份验证，以及本章中提到的各种其他主题。

通常而言，组织会尝试尽可能多地自动完成置备后任务。例如，Azure Active Directory 的设置和配置大体可通过一个 PowerShell 脚本或数个 ARM 模板来完成。如果您要部署许多集群，这有助于减少在重复性任务上花费的时间。

本指南的下一章将探讨如何在生产就绪型 Azure 红帽 OpenShift 集群上部署示例应用。

第 7 章

部署示例应用

本指南中的内容主要面向担当开发和运维角色的技术受众，帮助他们了解让 Azure 红帽 OpenShift 投入生产环境需要些什么。本指南假定读者基本掌握了红帽 OpenShift，因为其架构、管理门户和用户体验在许多方面与 Azure 红帽 OpenShift 是相同的。

本章可以充当一份快速入门参考，为您讲解如何部署一个名为“Fruit Smoothies”的示例应用。此应用可以充当一个快速起点和提示，帮助您更充分地了解如何使用 Azure 红帽 OpenShift。请注意，这个示例应用是面向 Azure Kubernetes Service 维护的，并部署在 Azure 红帽 OpenShift 进之上，以证明 OpenShift 是 100% 兼容 Kubernetes 的。

本章大部分基于来自 http://aroworkshop.io 的内容，也添加了一些额外的说明和背景。

一个点评应用的概述

应用架构十分简单，如下所示：

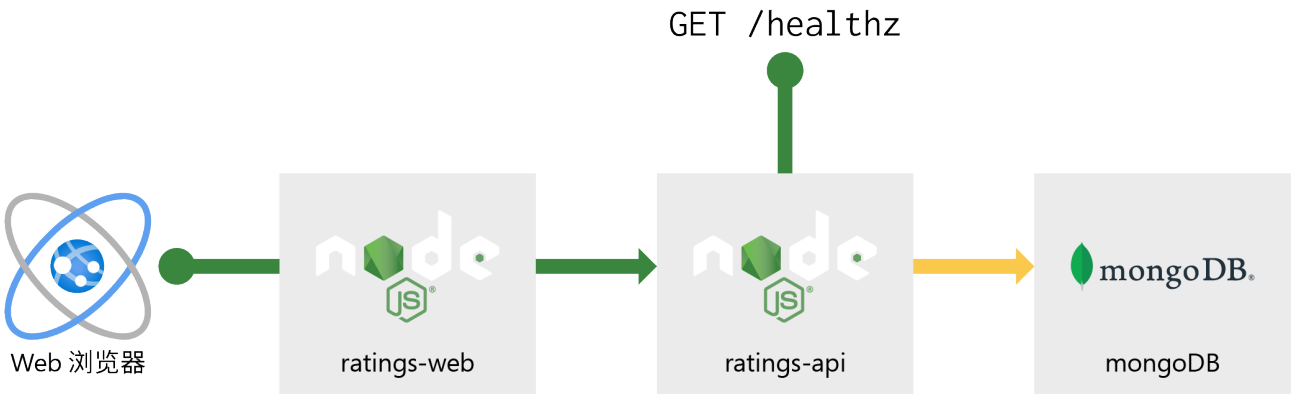


图 7.1：Fruit Smoothies 应用的架构

在上图中，您可以看到这个应用包含三个服务和两个公开端点，具体如下表所述。图中左侧的第一个端点代表了用户的 Web 浏览器，用来查看公开 HTML Web 应用。第二个端点（healthz）是 ratings-api 服务中的健康检查。下表中可以找到各项服务的描述以及对应存储库的链接：

组件	描述	GitHub 存储库
rating-web	对外公开的 Web 前端，即“网站”。	GitHub 存储库
rating-api	此服务接受来自 Web UI 的输入并将其存储在数据库中。它将数据库中的结果返回给端口 3000 上的 Web 应用。	GitHub 存储库
mongodb	含有预载数据的 NoSQL 数据库。	数据

您可以访问 GitHub 存储库链接来进一步探索和了解这些应用。后面的说明中将提供如何部署各个应用的分步骤指导。

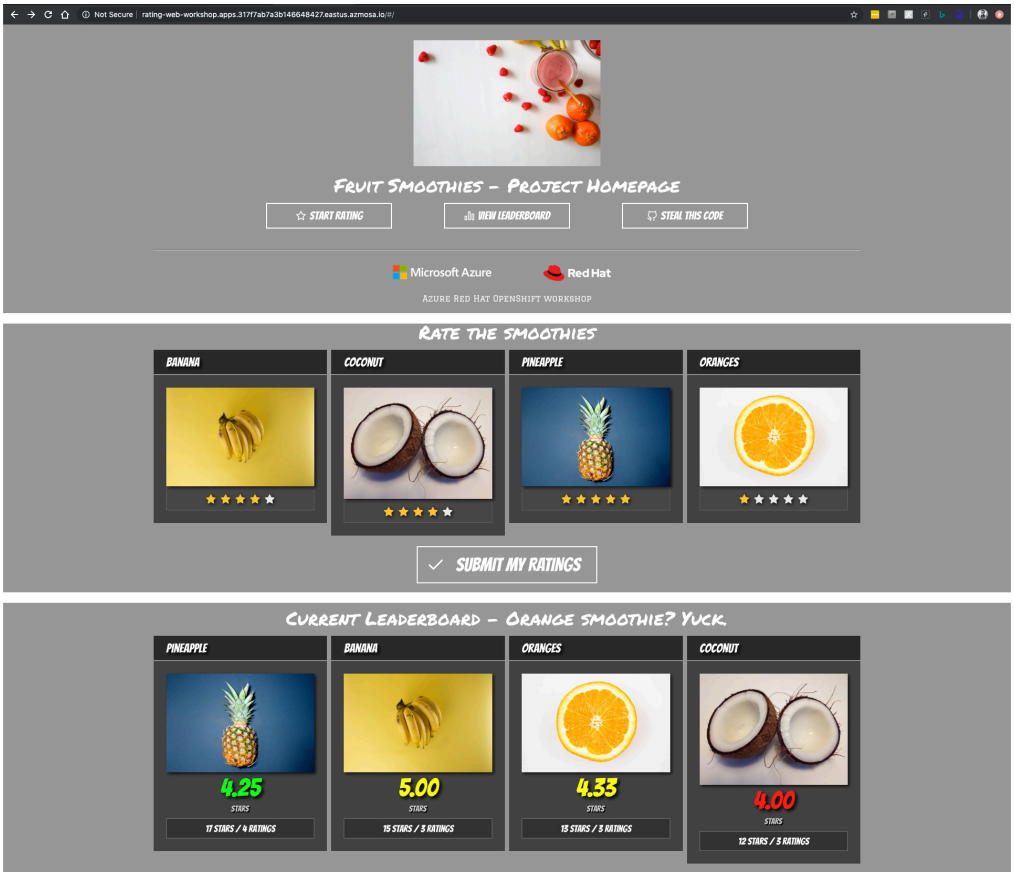


图 7.2：显示 Fruit Smoothies 应用概貌的屏幕截图

完成之后，您便拥有了一个如前面屏幕截图所示正常运行的 Web 应用了。您应该也对开发和运维团队如何部署应用到 Azure 红帽 OpenShift 有更加深刻的认识，这应该有助于您在自己部署应用到 Azure 红帽 OpenShift 时做出更好的决策。

创建和连接集群

本章的其余内容假定您拥有一个可以正常运行的 Azure 红帽 OpenShift 环境。这个点评应用没有特殊要求，它将部署到一个空白的全新 Azure 红帽 OpenShift 环能中。如果您还没有置备集群，请查阅以下章节：

- *第 4 章：置备之前 – 企业架构问题*
- *第 5 章：置备 Azure 红帽 OpenShift 集群*

“*第 5 章：置备 Azure 红帽 OpenShift 集群*”中的“*访问集群*”小节是一个非常实用的参考，可提示您如何访问之前置备的集群。

登录 Web 控制台

每个 Azure 红帽 OpenShift 集群具有一个 OpenShift Web 控制台的 DNS 地址。您可以使用 `az aro list` 命令来列出当前 Azure 订阅中的集群：

```
az aro list -o table
```

集群 Web 控制台的 URL 将会列出。在新的浏览器标签页中打开这个链接，并使用 `kubeadmin` 用户进行登录，或者使用具有创建项目权限的其他用户帐户。

登录后，您应该可以看到 Azure 红帽 OpenShift Web 控制台。

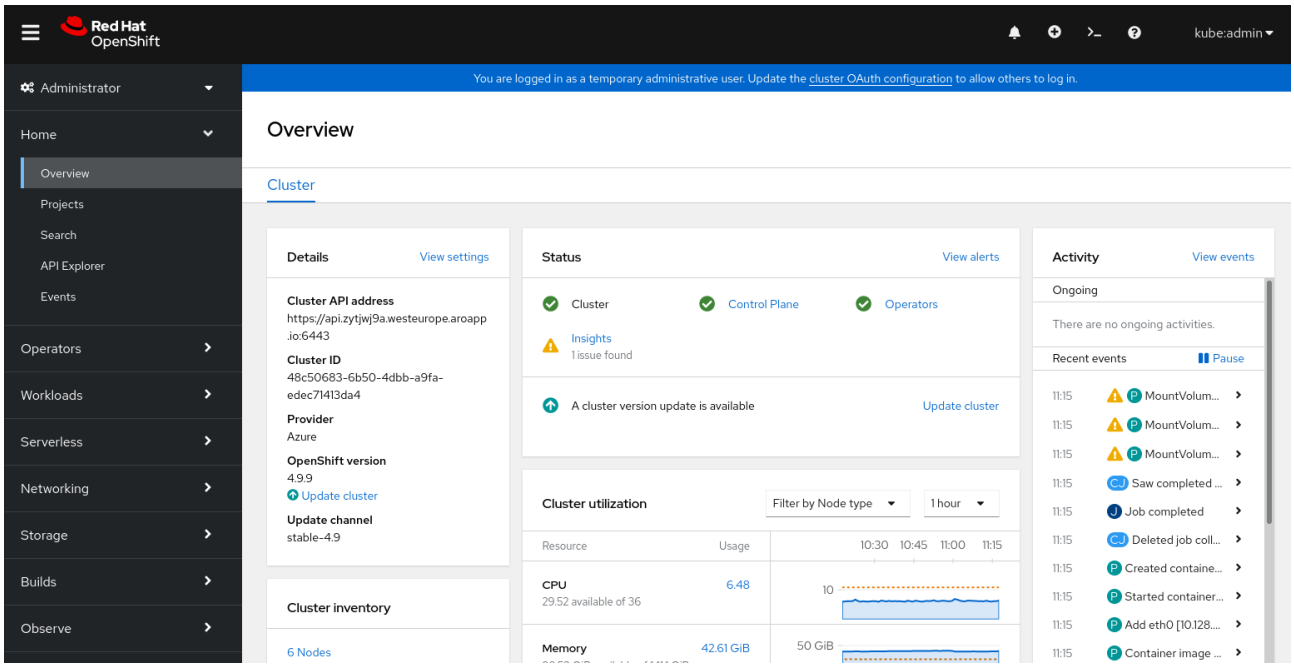


图 7.3：Azure 红帽 OpenShift Web 控制台

安装 OpenShift 客户端

打开 Azure Cloud Shell 或使用本地的 Linux 和终端，再安装 OpenShift 命令行客户端。需要使用它来从命令行访问集群。具体操作说明如下所示：

```
cd ~
curl https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-linux.tar.gz > openshift-client-linux.tar.gz

mkdir openshift

tar -zxvf openshift-client-linux.tar.gz -C openshift

echo 'export PATH=$PATH:~/openshift' >> ~/.bashrc && source ~/.bashrc
```

另外，适用于 Windows 或 Mac 的下载链接如下：

- https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-windows.zip
- https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-mac.tar.gz

您应该可以从下载的任意软件包运行 oc 命令。

检索登录命令和令牌

安装客户端后，需要获取一个令牌来登录集群。登录 OpenShift Web 控制台，再单击右上方的用户名，然后单击 **Copy Login Command**。

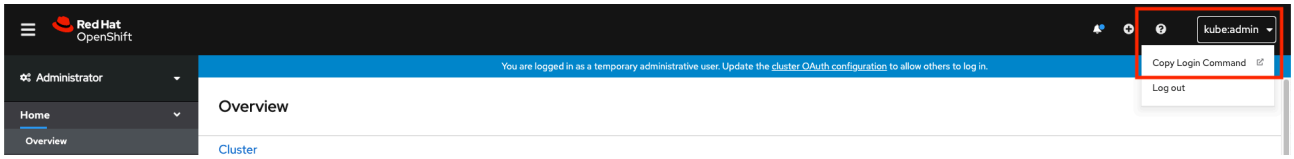


图 7.4：复制登录命令以登录集群

将登录命令粘贴到您的 shell 中（本地 Linux 终端或 Azure Cloud Shell）。您应该能够连接到集群。

图 7.5：使用登录命令连接集群

连接之后，我们便可继续创建项目了。

创建项目

OpenShift 中的项目如同一个逻辑文件夹，用来存放这个点评应用。在红帽 OpenShift 中，所有容器和应用都需要放在某种类型的项目中。您可以使用项目来划分应用，甚至还可划分部门。下面几点说明讲解了如何创建项目。

虽然您可以从 Web 界面创建项目，但这些说明用的是命令行：

```
oc new-project workshop
```

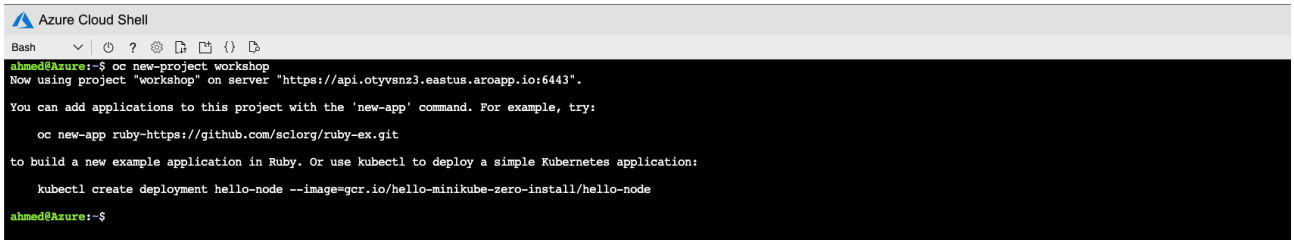


图 7.6：在 Azure Cloud Shell 中创建一个新的工作区

创建项目之后，您可以使用 `oc project workshop` 切换到这个项目。下一步将部署本章开头介绍的三个微服务之一，MongoDB。它将部署到我们刚才创建的项目中。

资源

- Azure 红帽 OpenShift 文档 – CLI 入门
- Azure 红帽 OpenShift 文档 – 项目

部署 MongoDB

Azure 红帽 OpenShift 提供了一个容器镜像和模板，降低了创建新 MongoDB 数据库服务的难度。 该模板提供了参数字段，以使用预定义的默认值（包括自动生成密码值）来定义所有必填的环境变量（用户、密码、数据库名称等）。除此之外，该模板还将对部署配置和服务进行定义。

MongoDB 实例将作为来自 Docker Hub 的容器镜像直接进行部署。OpenShift 允许您完全使用 Web 控制台来完成。从菜单顶部切换到开发人员视图，前往 **Add** 页面，再选择 **Container images**。

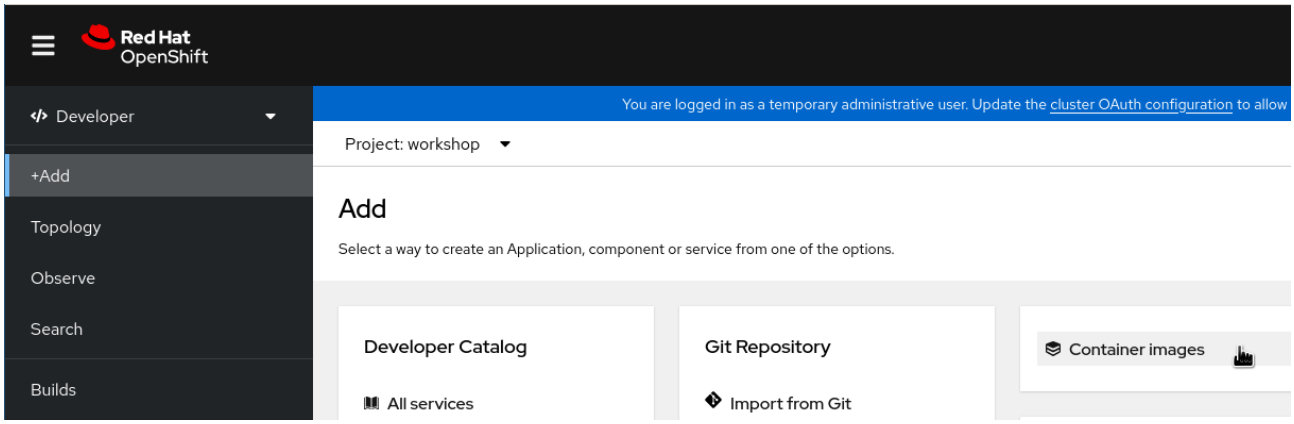


图 7.7：添加容器镜像

这会将您引导到 **Deploy Image** 页面。按照如下所示填写表单：

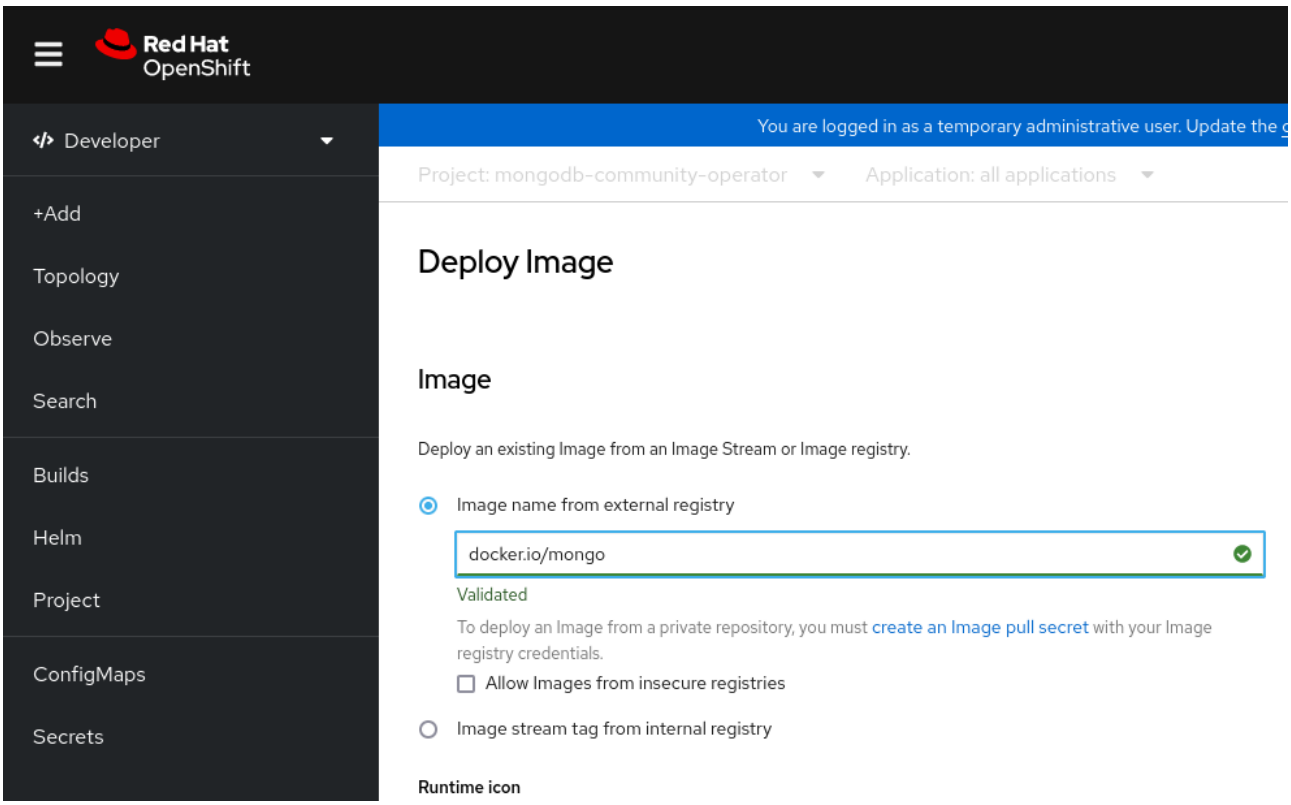


图 7.8：填写表单以部署镜像

务必按照如下所示设置表单中的值：

字段	值
外部镜像仓库的镜像名称	`docker.io/mongo`
运行时图标	mongodb
应用名称	`ratings`
名称	mongodb
创建指向应用的路由	不要选中 — 路由允许从外部访问数据，这个应用不需要
资源类型	Deployment

转到表单底部时，选择 Deployment 链接来展开表单，以便设置环境变量。

以下环境变量将作为默认值，用于在第一次启动时初始化数据库：

名称	值
MONGO_INITDB_DATABASE	ratingsdb

不需要为 MongoDB 数据库设置用户名和密码，这是默认的配置，身份验证已被停用。

再次检查环境变量，然后单击 **Create** 按钮来继续部署 MongoDB 容器。

片刻之后，MongoDB 实例应当在 workspace 项目中就绪并且正常运行。您可以通过切换到 **Topology** 视图来查看此部署。

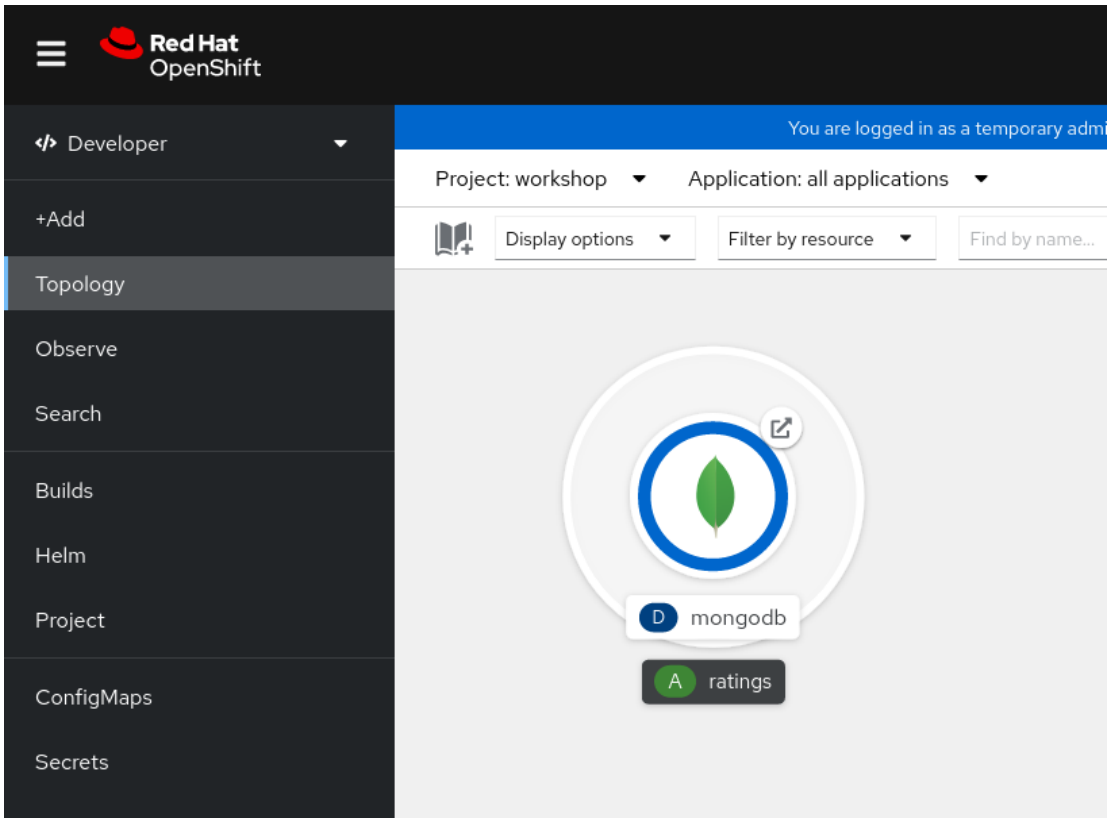


图 7.9：MongoDB 实例已就绪并且正常运行

运行 oc get all 命令，以查看新应用的状态并验证 MongoDB 模板是否已成功部署。oc get all 的示例输出如下：

```
user@host: oc get all
NAME                            READY   STATUS    RESTARTS   AGE
pod/mongo-6c6fcb45b8-8wpdm      1/1     Running   0          29s

NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
service/mongo     ClusterIP   172.30.88.119    <none>        27017/TCP   30s

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mongo     1/1     1            1           30s

NAME                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/mongo-6c6fcb45b8     1         1         1       30s

NAME                                      IMAGE REPOSITORY
TAGS      UPDATED
imagestream.image.openshift.io/mongo      image-registry.openshift-image-registry.svc:5000/workshop/mongo
latest    30 seconds ago
```

如果一切都正确无误，STATUS 列应当会显示容器处于 Running 状态。

检索 MongoDB 服务主机名

完成部署后，我们需要查找之前创建的服务，以便从集群内访问数据库。svc 是 services 的简写：

```
user@host: oc get svc mongodb
NAME     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
mongo    ClusterIP   172.30.88.119    <none>        27017/TCP   77s
```

该服务可通过以下 DNS 名称访问：mongodb.workshop.svc.cluster.local，具体由 [service name].[project name].svc.cluster.local 构成。该主机名仅在集群内部解析。

部署 Ratings API

现在到了部署第二个应用的时候了，即 `rating-api`。它是一个 Node.js 应用，通过连接到 MongoDB 实例来检索和评价货品。如下是您在部署此应用时需要了解的一些详细信息：

- GitHub 上的 rating-api
- 容器开放端口 3000
- 使用名为 MONGODB_URI 的环境变量配置 MongoDB 连接

请记下这些详细信息，因为后面几个小节需要参考它们。

将应用分支拷贝到自己的 GitHub 存储库

要能够进行更改，例如添加 CI/CD Webhook，您需要拥有自己的 `ratings-api` 代码副本。在 Git 中，这称为“fork”（分支）。您可以将应用分支拷贝到个人 GitHub 存储库中。前往上述 GitHub 存储库，并单击 **Fork** 按钮。

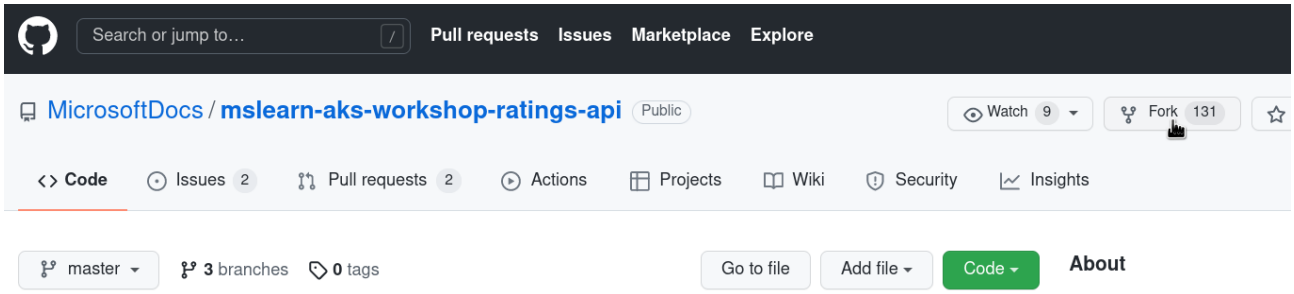


图 7.10：将应用分支拷贝到 GitHub 存储库

记下新的存储库地址，后续指令中需要用到它。

使用 OpenShift CLI 来部署 rating-api

OpenShift 能够通过查看内容并根据内容来选择“构建器镜像”，例如 Java、PHP、Perl、Python 或类似内容，直接从 Git 存储库部署代码。本例中的 `rating-api` 是一个 JavaScript 应用。构建器镜像将使用 npm 下载 JavaScript 依赖项，下载结果是一个存储在内部 OpenShift 容器镜像仓库中的新容器镜像。这种构建策略称为**源至镜像（S2I）**，其更详细的说明可参见术语表。

您可以使用 oc new-app 开始新的 S2I 构建：

```
user@host: oc new-app https://github.com/<your GitHub username>/mslearn-aks-workshop-ratings-api --strategy=source --name=rating-api

--> Found image 0aea15f (3 weeks old) in image stream "openshift/nodejs" under tag "14-ubi8" for "nodejs"

    Node.js 14
    ----------
     Node.js 14 available as container is a base platform for building and running various Node.js 14 applications and frameworks. Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

    Tags: builder, nodejs, nodejs14

    * The source repository appears to match: nodejs
    * A source build using source code from https://github.com/MicrosoftDocs/mslearn-aks-workshop-ratings-api will be created
      * The resulting image will be pushed to image stream tag "rating-api:latest"
      * Use 'oc start-build' to trigger a new build
```

切换到 Web 控制台中的 **Topology** 视图，您应该会看到应用构建已开始，几分钟后部署便会成功完成。

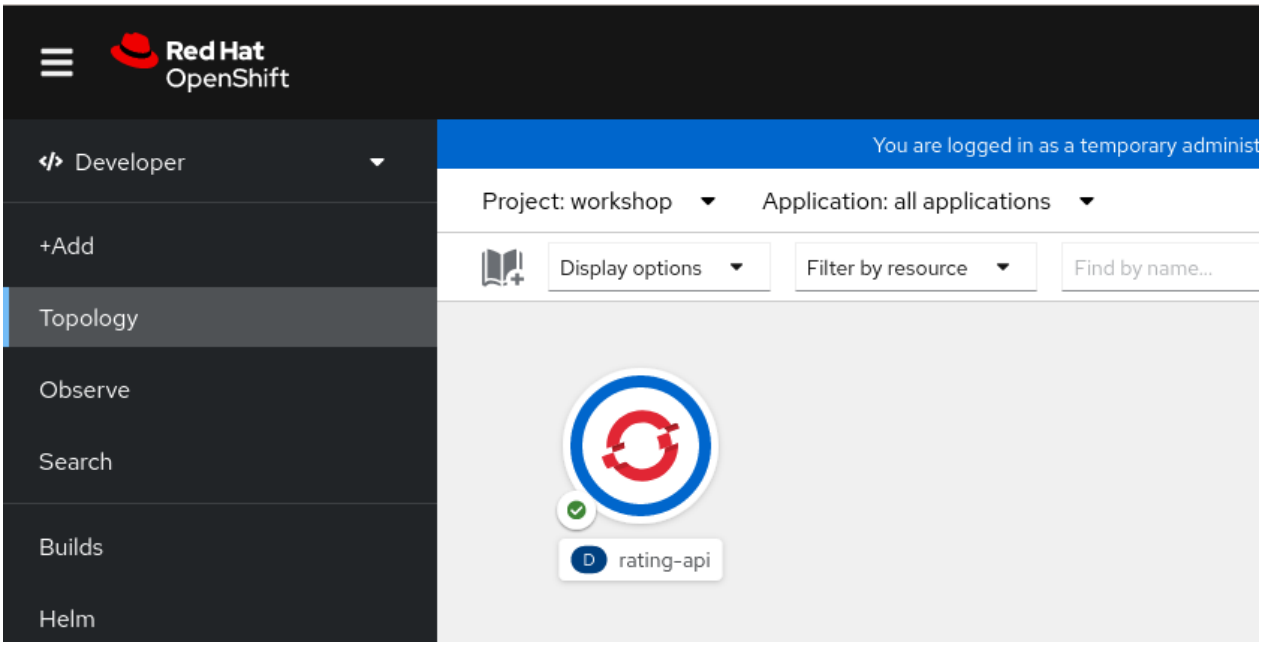


图 7.11：Topology 视图

容器集的构建和启动应该只需要花费一两分钟。

配置必要的环境变量

至此，数据库已部署完毕，ratings API 也一样。不过，需要告诉 ratings API 如何连接到这个数据库。配置基于容器的应用通常是通过使用环境变量来实现的。

单击所需的部署并进行编辑。创建以下环境变量：

名称	值
MONGODB_URI	mongodb://mongodb.workshop.svc.cluster.local:27017/ratingsdb

保存了这个新环境变量后，会触发重新部署 `ratings-api` 服务以使用新环境变量。

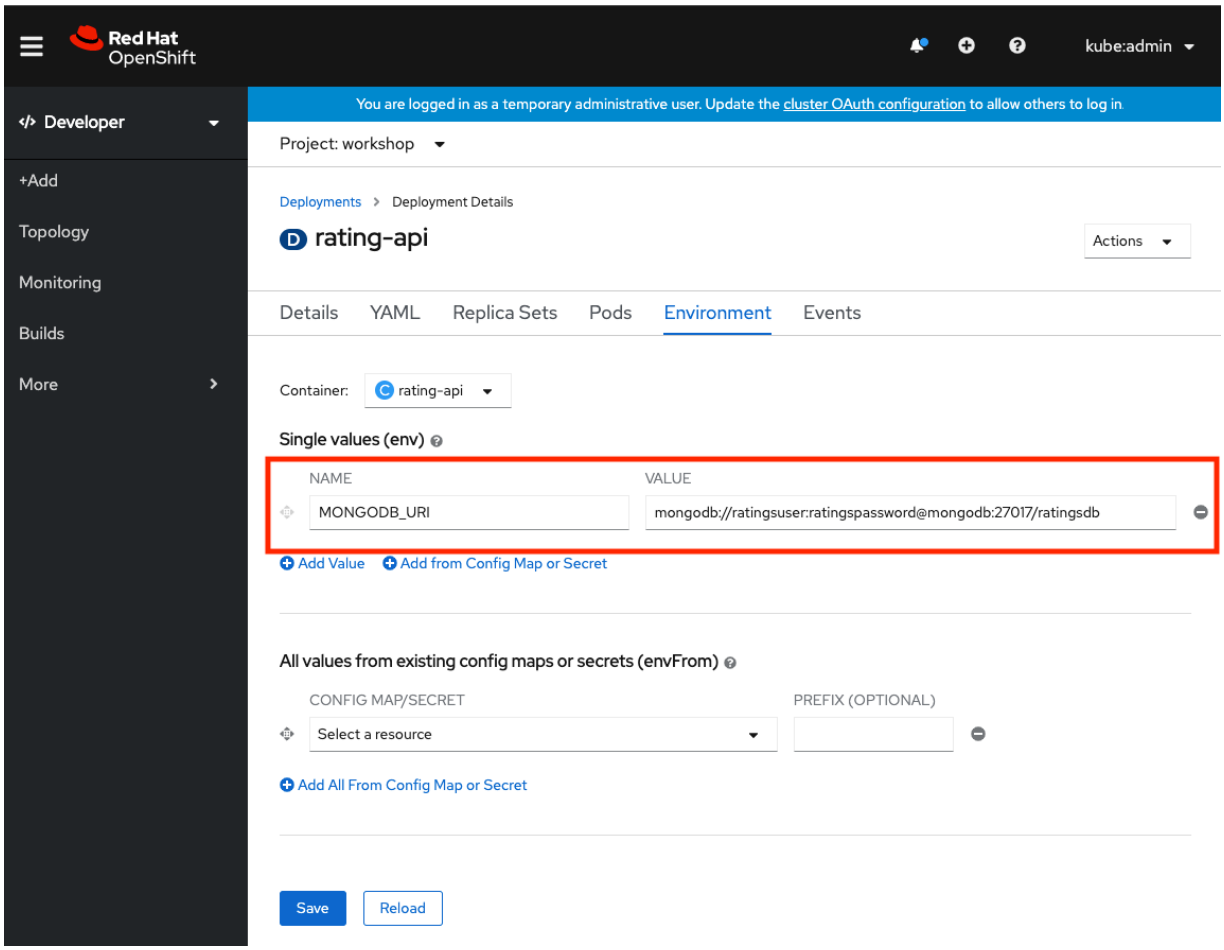


图 7.12：通过 Web 控制台设置 MONGODB_URI 环境变量

这也可通过命令行来完成，例如：

```
oc set env deploy/rating-api MONGODB_URI=mongodb://mongodb.workshop.svc.cluster.local:27017/ratingsdb
```

不论使用哪种方式，OpenShift 都需要重新启动容器来确保使用新的环境变量。

验证服务正在运行

如果浏览到 `rating-api` 部署的日志，您应该会看到一条确认代码可以成功连接到 MongoDB 的日志消息。为此，可在部署详细信息屏幕中单击 **Pods** 选项卡，然后单击其中一个容器集。

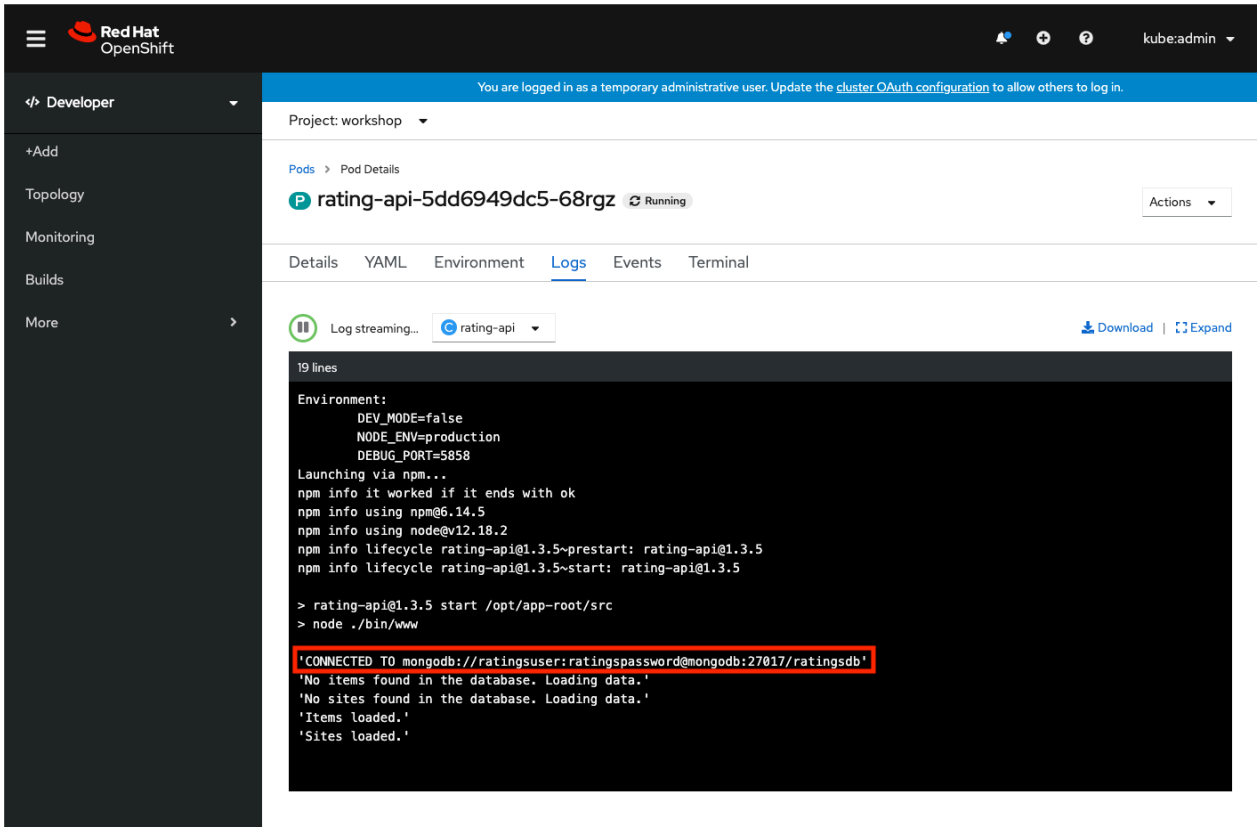


图 7.13：确认代码可以成功连接到 MongoDB 的日志消息

调整 rating-api 服务端口

OpenShift 将使用端口 `8080` 来创建服务。不过，因为库更新的关系，此服务要在端口 `3000` 上运行。因此需要编辑默认服务。

前往 **Networking** → **Services** 菜单，然后在服务列表中按照如下所示编辑 `rating-api` 服务（只需将 `8080` 替换为 `3000`）：

```
  ports:
    - name: 3000-tcp
      protocol: TCP
      port: 3000
      targetPort: 3000
```

重新启动服务以使用新端口：

```
user@host: oc rollout restart deploy/rating-api
```

现在，服务已绑定到正确的端口，即 `3000`，而非 `8080`。

检索 rating-api 服务主机名

我们需要验证存在 `rating-api` 服务，因为下一节中部署 `rating-web` 应用时会用到该服务：

```
oc get service rating-api
```

该服务可通过 DNS 名称 `rating-api.workshop.svc.cluster.local:3000` 及端口 `3000` 来访问，DNS 名称由 `[service name].[project name].svc.cluster.local` 构成。该主机名仅在集群内部解析。

部署点评应用的前端

`rating-web` 是一个连接到 `rating-api` 的 Node.js 应用。如下是您在部署此应用时需要了解的一些详细信息：

- GitHub 上的 rating-web
- 容器开放端口 8080
- 该 Web 应用通过内部集群的 DNS 连接到 API，并通过名为 API 的环境变量使用代理

使用 OpenShift CLI 来部署 rating-web

与 rating-api 应用一样，可以使用 oc new-app 命令来通过 S2I 部署此应用。但这一次，该应用将通过 Dockerfile 策略来创建，使用 Git 存储库中已存在的 Dockerfile 即可。因此，不要指定 --strategy 参数：

```
user@host: oc new-app https://github.com/<your GitHub username>/mslearn-aks-workshop-ratings-web
--name rating-web

--> Found container image e1495e4 (2 years old) from Docker Hub for "node:13.5-alpine"

    * An image stream tag will be created as "node:13.5-alpine" that will track the source image
    * A Docker build using source code from https://github.com/MicrosoftDocs/mslearn-aks-
workshop-ratings-web will be created
      * The resulting image will be pushed to image stream tag "rating-web:latest"
      * Every time "node:13.5-alpine" changes a new build will be triggered

--> Creating resources ...
    imagestream.image.openshift.io "node" created
    imagestream.image.openshift.io "rating-web" created
    buildconfig.build.openshift.io "rating-web" created
    deployment.apps "rating-web" created
    service "rating-web" created
--> Success
```

在将依赖项构建到容器中需要一些时间，请等待 2 到 3 分钟左右以便完成构建，然后返回到 **Topology** 视图查看服务是否上线运行。

配置必要的环境变量

创建用于 rating-web 部署配置的 API 变量。此变量的值将成为 rating-api 服务的主机名 / 端口。

除了通过 Azure 红帽 OpenShift Web 控制台设置环境变量外，您还可以通过 OpenShift CLI 来设置：

```
oc set env deploy rating-web API=http://rating-api:3000
```

使用路由来公开 rating-web 服务

公开服务意味着提供可公开访问的 URL，供用户用来访问该服务。如果服务不公开，则只能在集群内部访问：

```
user@host: oc expose svc/rating-web
route.route.openshift.io/rating-web exposed
```

最后，获取刚刚公开的服务的 URL：

```
user@host: oc get route rating-web
NAME          HOST/PORT                                                PATH    SERVICES      PORT
TERMINATION    WILDCARD
rating-web    rating-web-workshop.apps.zytjwj9a.westeurope.aroapp.io      rating-web   8080-tcp
None
```

请注意，**完全限定域名（FQDN）**默认为由应用名称和项目名称组成。FQDN 的其余部分（即子域）是您的 Azure 红帽 OpenShift 集群的特定应用子域。

试验服务

在您的浏览器中打开主机名。您应该会看到这个点评应用的页面。试用一下，提交几个投票，并查看排行榜。

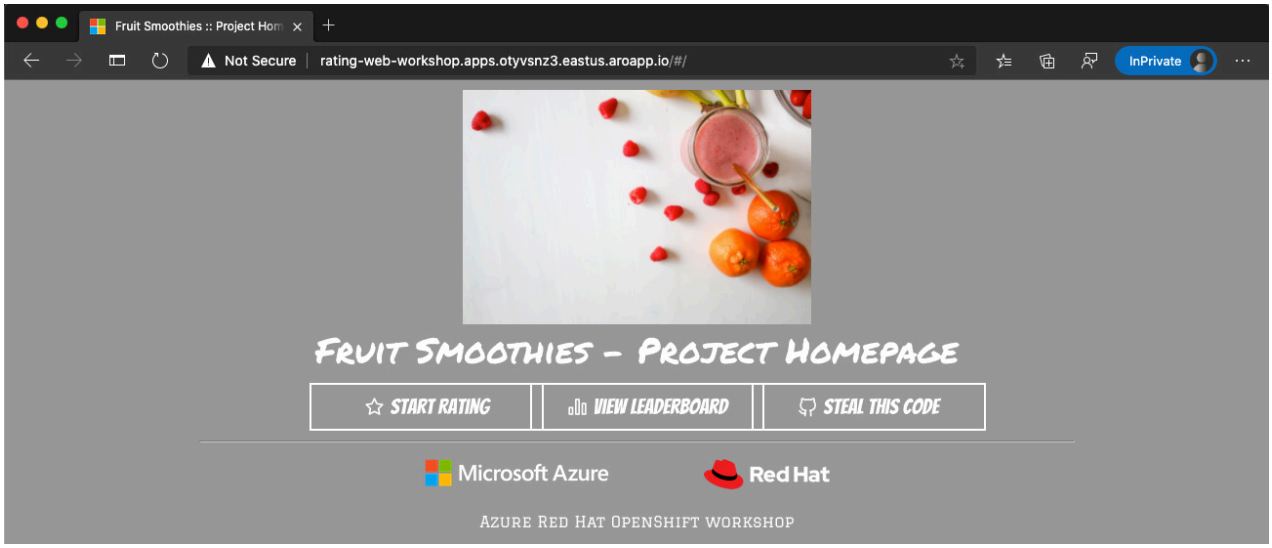


图 7.14：试验点评应用服务

设置 GitHub Webhook

要在将代码推送到 GitHub 存储库时触发 S2I 构建，您需要设置 GitHub webhook：

1. 检索 GitHub webhook 触发机密。您需要在 GitHub webhook URL 中使用这个机密：

```
user@host: oc get bc/rating-web -o=jsonpath='{.spec.triggers..github.secret}'
3ffcc8d5-a243
```

   记下机密密钥，因为后面有几个步骤会需要它。

2. 从构建配置检索 GitHub webhook 触发 URL：

```
user@host: oc describe bc/rating-web
...
Webhook GitHub:
        URL:    https://api.quwhfg7o.westeurope.aroapp.io:6443/apis/build.openshift.io/v1/
namespaces/workshop/buildconfigs/rating-web/webhooks/<secret>/github
...
```

3. 将 `<secret>` 占位符替换为您在第一部中检索到的机密。本例中的机密为 `3ffcc8d5-a243`，因此生成的 URL 是：

```
https://api.quwhfg7o.westeurope.aroapp.io:6443/apis/build.openshift.io/v1/namespaces/
workshop/buildconfigs/rating-web/webhooks/3ffcc8d5-a243/github
```

   您将使用此 URL 在 GitHub 存储库中设置 webhook。

4. 前往您的 GitHub 存储库。从 **Settings** → **Webhooks** 选择 **Add Webhook**。
5. 在 **Payload URL** 字段中，粘贴 GitHub URL 并更改 `<secret>` 值以使用您的机密。
6. 在 **Content type** 字段中，将默认的 `application/x-www-form-urlencoded` 更改为 `application/json`。

7. 单击 **Add webhook**。

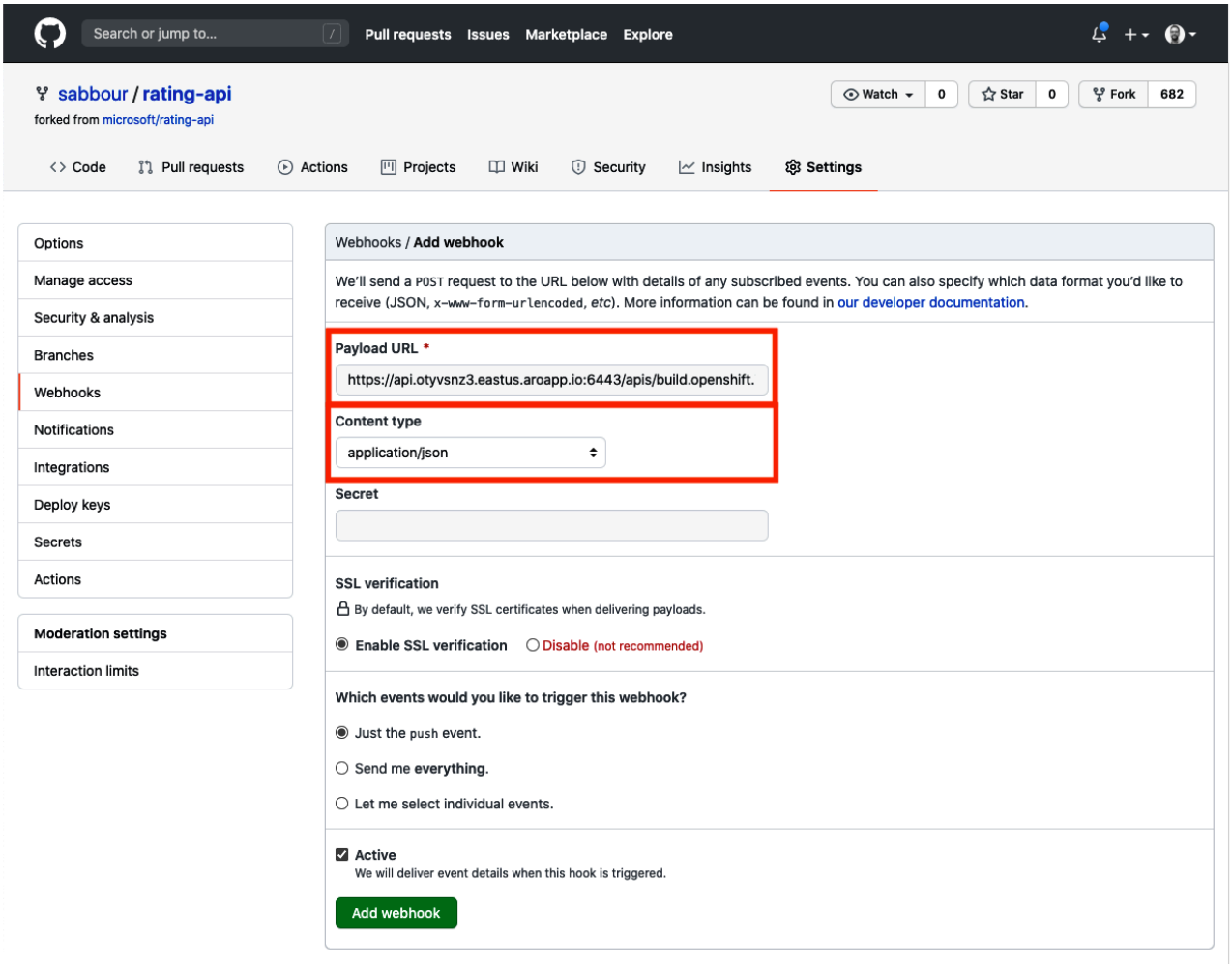


图 7.15：添加 Webhook

您应该会看到一条来自 GitHub 的消息，表明您的 webhook 已配置成功。

现在，每当您将更改推送到 GitHub 存储库时，都会自动启动一个新构建，构建成功后将会启动新的部署。

更改 网站应用并查看滚动更新

在 GitHub 上，找到您的存储库中的 https://github.com/<your GitHub username>/rating-web/blob/master/src/App.vue 文件。

编辑此文件；将 background-color: #999; 行更改为 background-color: #0071c5。

将文件更改提交到 master 分支。

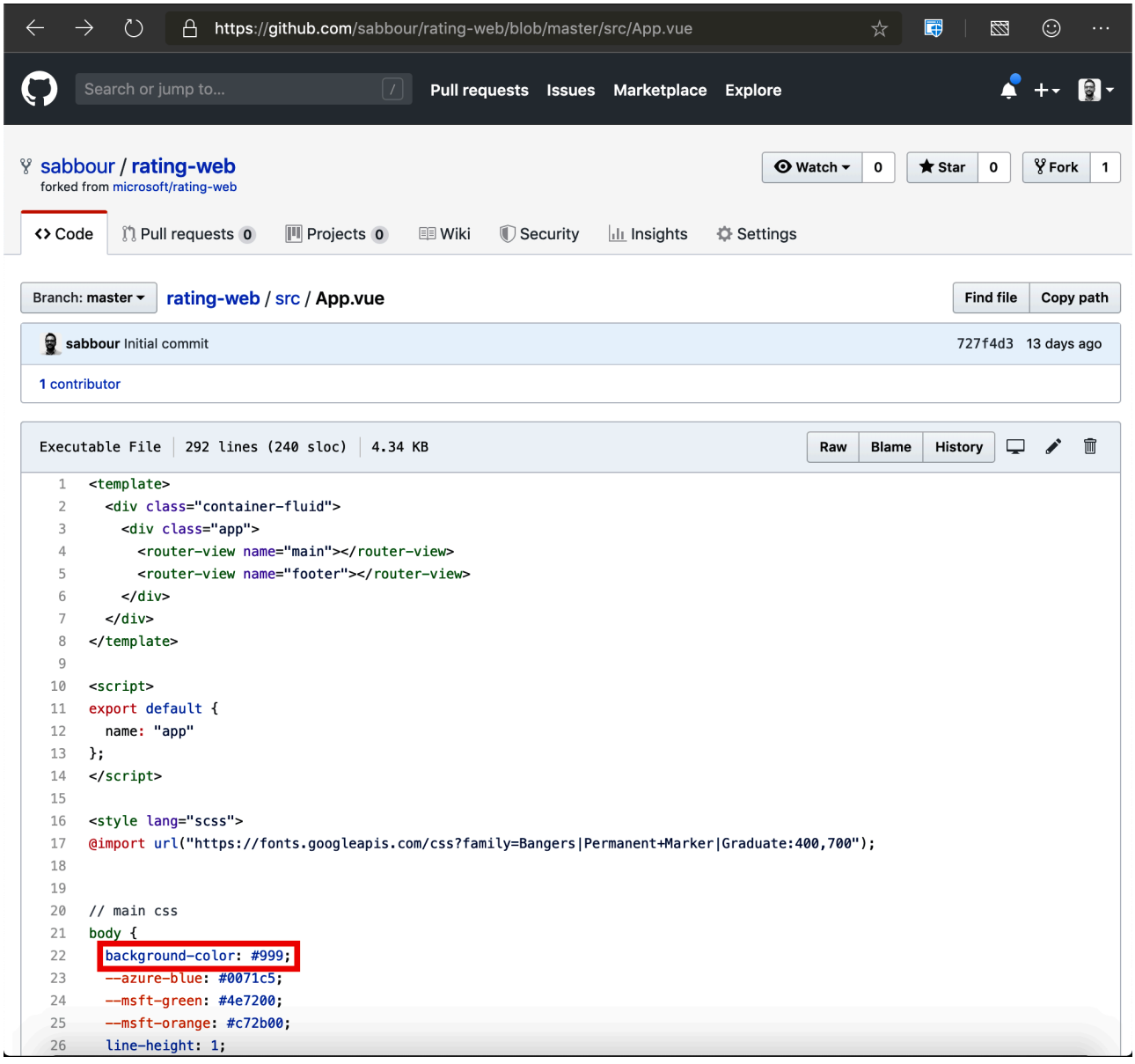


图 7.16： 将文件更改提交到 master 分支

然后，立即前往 OpenShift Web 控制台中的 **Builds** 选项卡。您会看到推送触发的新构建已排入队列中。完成之后，它会触发新的部署，您应该会看到网站的颜色得到了更新。

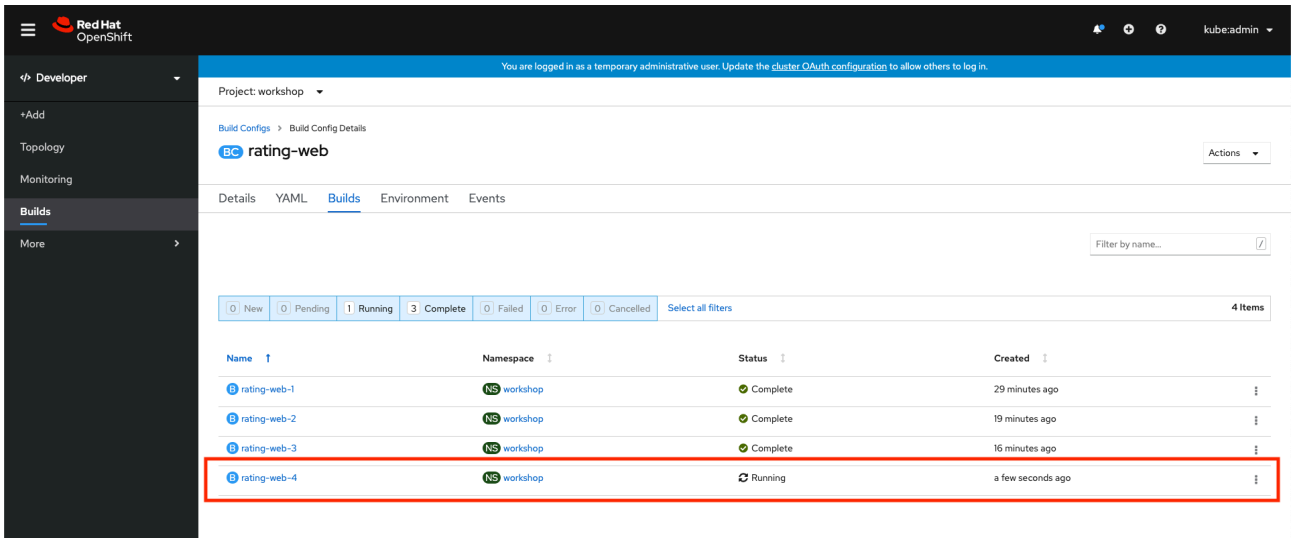


图 7.17：Builds 选项卡中显示一个新构建正在运行

现在，返回到您的 `ratings-web` 页面，如果一切顺利，您会看到背景颜色已经改变！

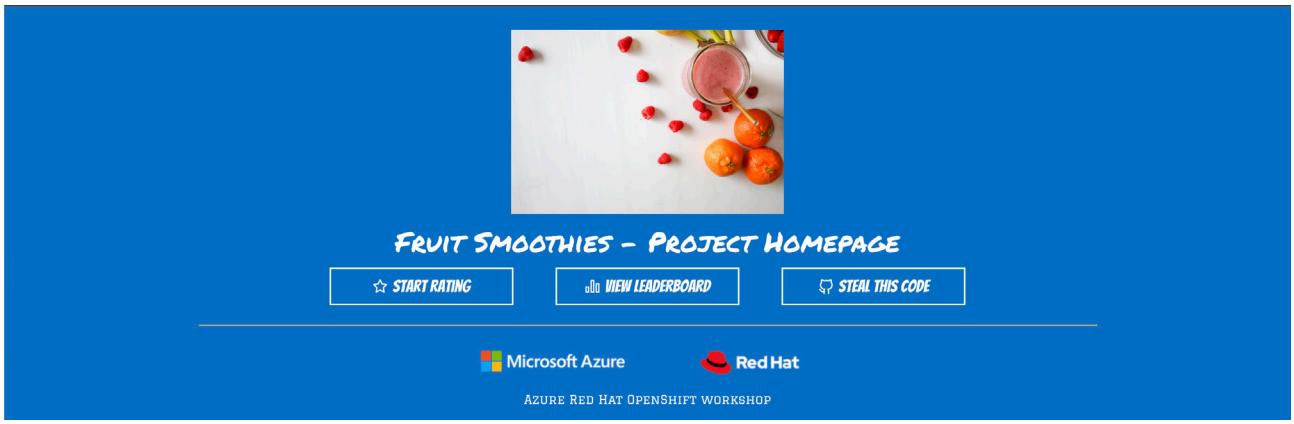


图 7.18：换成新颜色的 Fruit Smoothies 主页

摘要

在本章中，我们部署了一个基础的应用，它由三个更小的微服务应用构成，分别是 MongoDB 数据库、`ratings-api` 和 `ratings-web` 代码。尽管这与生产级别的应用有很大差异，但可以作为部署应用到 Azure 红帽 OpenShift 时的快速概念参考。您可以将这些说明用作参考，以其为基础来部署自己的应用。

红帽 OpenShift 也支持从 OperatorHub、Helm Charts 以及外部的 CI/CD 系统（如 Azure DevOps）部署应用。究竟哪一种部署策略最适合您的组织，取决于您在内部使用的工具和技术。

在下一章中，我们将探索 Azure 红帽 OpenShift 的一些其他价值，这些价值使得 OpenShift 成为基于 Kubernetes 构建的一个独特应用平台。我们将深入探索旨在满足企业应用复杂需求的各种特性与功能。

第 8 章

浏览应用平台

在前面几章中，我们了解了红帽 OpenShift 如何提供基于 Kubernetes 构建的各种服务。所有这些服务组合成一个真正的应用平台，它们可以分组为五个类别：平台服务、应用服务、数据服务、开发人员服务和 Kubernetes 集群服务。

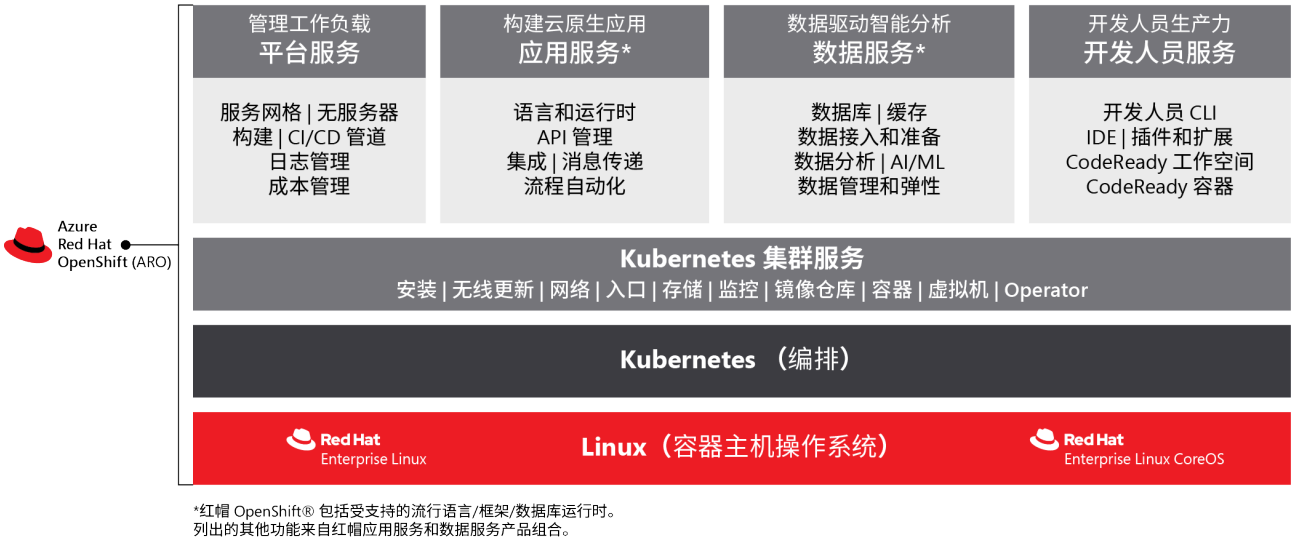


图 8.1：Azure 红帽 OpenShift 中包含的服务

分组到 **OpenShift Platform Plus**—**多集群管理**、**集群安全性**和**全局镜像仓库**的组件是额外产品，需要订阅。它们与 Azure 红帽 OpenShift 兼容，但不包含在 Azure 红帽 OpenShift 产品中。

在下面几节中，我们将探索这些服务提供的一些关键益处，并提供一些链接供您查找更多信息。

集群服务 – 集成容器镜像仓库

红帽 OpenShift 附带一个集成式内部容器镜像仓库，随同集群一起部署。这个镜像仓库既用于集群内部服务（如集群 operator），默认情况下也用于客户的应用容器。此镜像仓库属于标准配备，不需要额外设置，甚至也由基础架构运维人员进行维护。

部署 Kubernetes 容器服务的所有客户都有可能需要部署自己的容器镜像仓库，并且使自己的容器镜像保持私密。使用 OpenShift 时，不需要进行额外的 Day-2 安装和设置，因为集群中已经提供了内置容器镜像仓库。这是 OpenShift 简单又省时特色的一个例子。

集成式 OpenShift 容器平台镜像仓库概述

通常，集群管理员可能选择将这个容器镜像仓库公开到集群外，以便位于 OpenShift 外部的用户可以将容器镜像推送这个镜像仓库中。这在 Azure 红帽 OpenShift 中受到完整支持，详细操作说明可在关于公开镜像仓库的标准 OpenShift 文档中找到。

平台服务 – OpenShift Pipelines

红帽 OpenShift 的客户通过许多方式来构建应用，许多流行的 CI/CD 工具（例如 Jenkins、CircleCI 和 GitHub Actions）也都拥有支持 OpenShift 的插件。不过，OpenShift 也通过 OpenShift Pipelines operator 在平台中提供相应功能，让客户利用云原生容器管道来构建应用。

OpenShift Pipelines 基于名为 Tekton 的社区项目。管道中的每个阶段，例如从 Git 存储库拉取代码、运行 Java 编译器或汇编 RPM 软件包等，都是在容器内运行的。这意味着，开发和运维人员可以利用容器提供的完整好处，组建复杂精密的管道来构建软件。

OpenShift Pipelines 可以通过 OperatorHub 来安装。只要前往 **OperatorHub**，并选择相应 operator 即可开始安装。无需任何配置，operator 安装一般可在不到一分钟内完成。

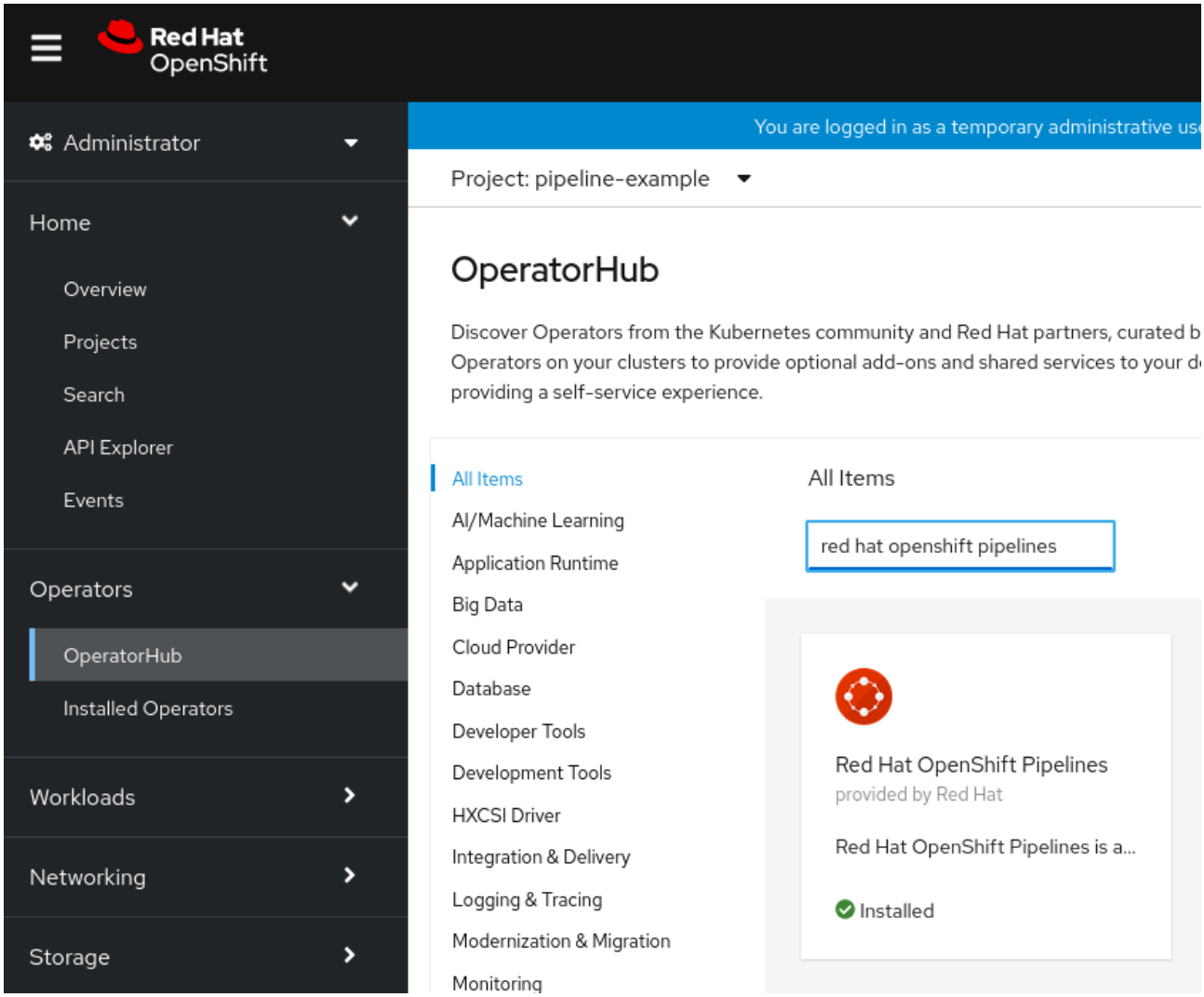


图 8.2：通过 OperatorHub 安装 OpenShift Pipelines

安装了 OpenShift Pipelines operator 后，您会在边栏中看到一个新的 **Pipelines** 部分，以及用于添加管道至目录项的选项，例如在构建 Node.js 示例时。

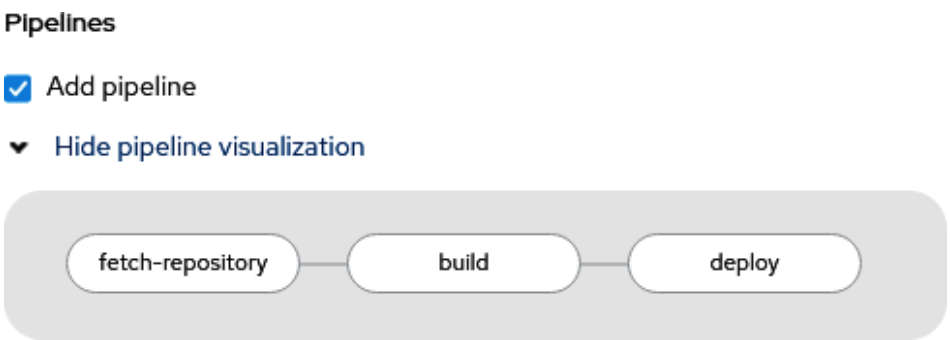


图 8.3：添加管道

通过使用 OpenShift Pipelines，您也可使用图形化构建器来设置和构建复杂的分支管道。以下屏幕截图中显示了一个更为复杂的管道示例：

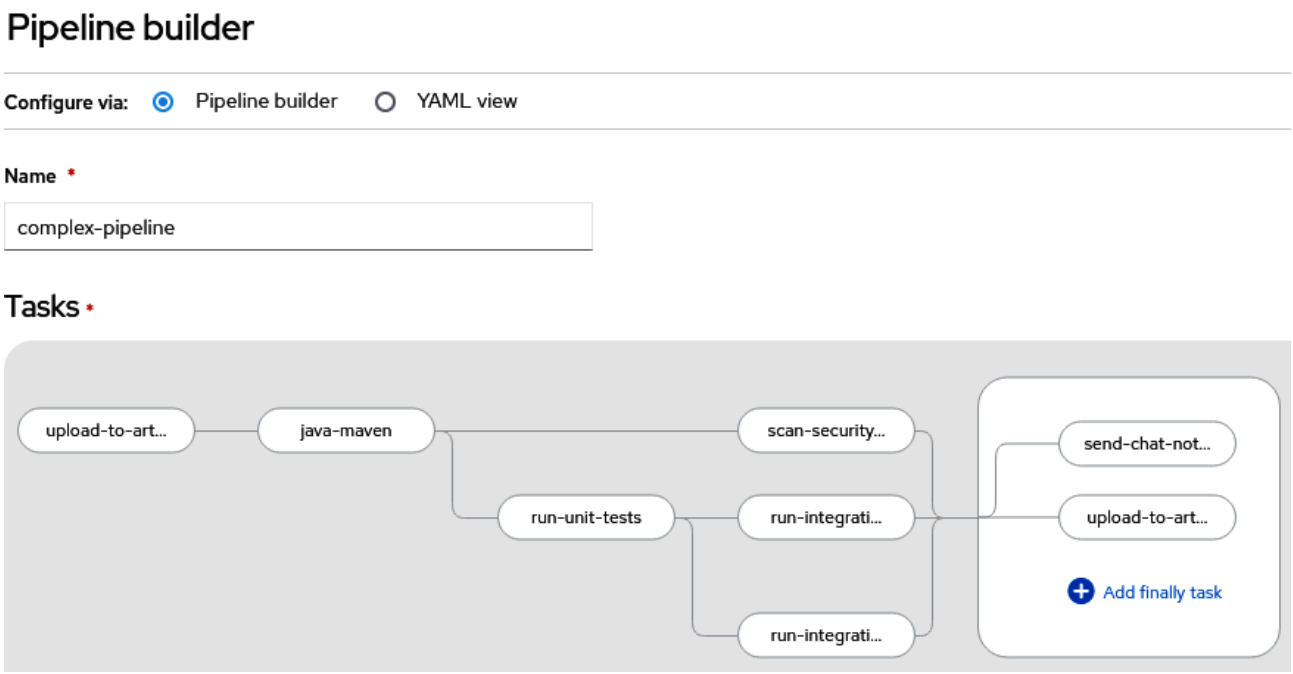


图 8.4：复杂管道

许多组织使用各种各样的工具和技术来构建软件，OpenShift Pipelines 可以让它们利用容器带来的所有好处，轻松地将构建直接集成到 OpenShift 中。它提供了一致且易用的 CI/CD 解决方案，所需的设置特别少，而无论使用的底层基础架构是什么。

扩展阅读

- 了解 OpenShift 中的 OpenShift Pipelines
- Tekton 社区网站

平台服务 – OpenShift Serverless

人们常有一个误解，认为容器只是用于长期运行的服务的实用技术。实际上，许多短期作业和无服务器功能是在短时间存活的容器上执行的。容器在启动速度、一致性和便于关闭等方面具有诸多优势，因此也非常适合无服务器工作负载。当然，所有无服务器服务确实需要底层服务器来执行代码，因此，无服务器有时也称为“功能即服务”。

红帽 OpenShift 通过 OpenShift Serverless operator 来实现无服务器（或功能即服务）工作负载。此 operator 基于名为 Knative 的热门开源项目。

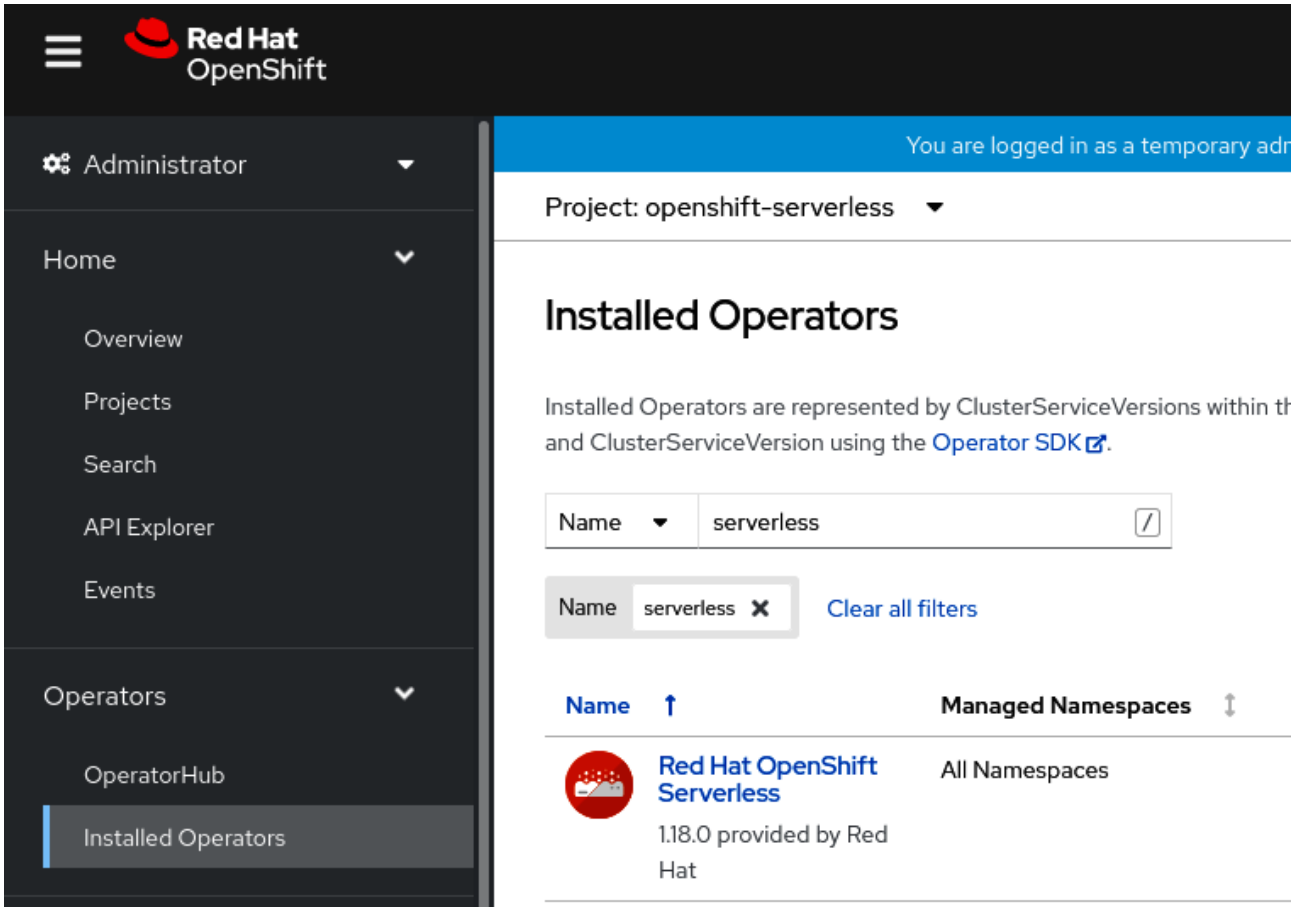


图 8.5：查看安装的 Operator

将 operator 部署到集群中后（这通常也只需一两分钟），需要对其进行一些设置。以下两个**自定义资源定义（CRD）**尤其值得注意：**Knative Serving** 和 **Knative Eventing**。

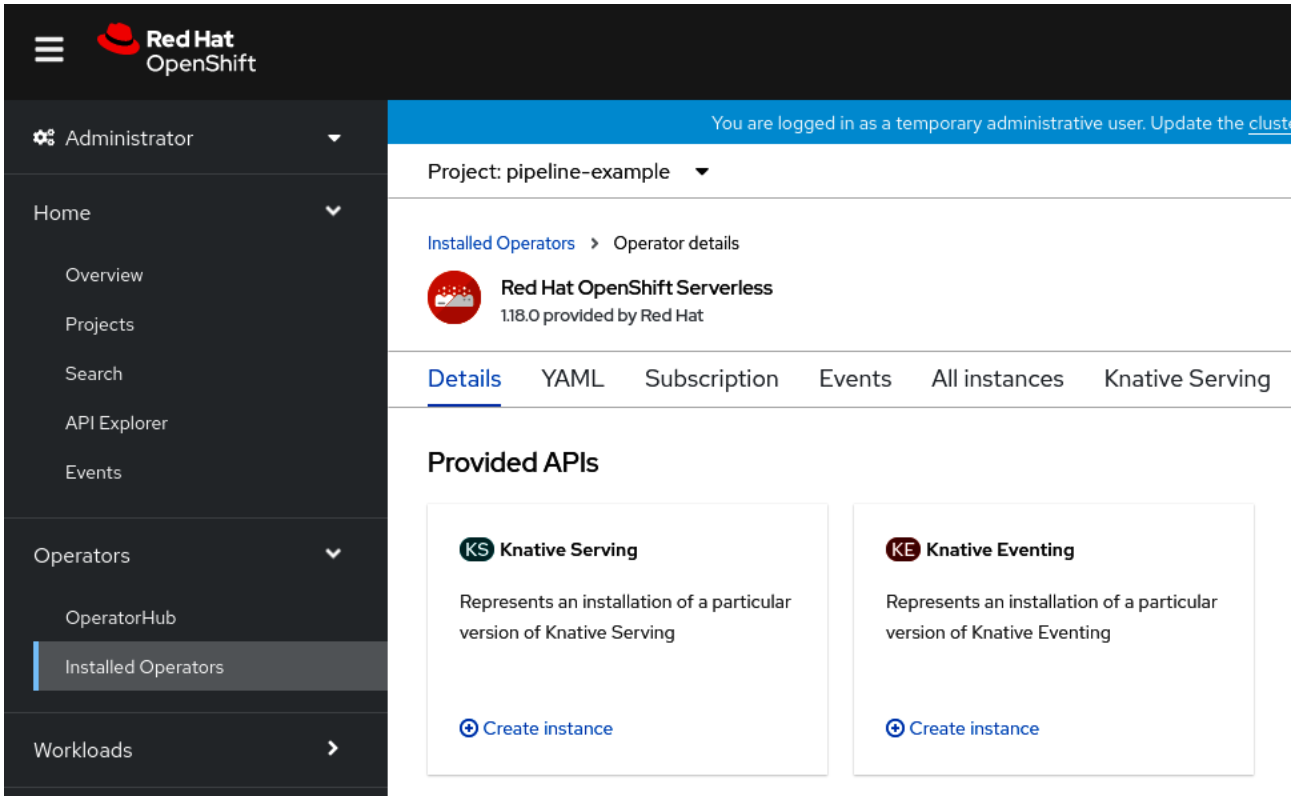


图 8.6：Operators 菜单中的 Knative Serving 和 Knative Eventing

- **Knative Serving** 可简化应用部署，基于传入流量动态扩展，并支持利用流量拆分的自定义部署策略；例如，将镜像上传到存储桶的功能，以及上传到 NoSQL 数据库的日志。
- **Knative Eventing** 允许在应用运行时（而非构建时）“延迟绑定”事件源；例如，一个应用响应存储桶中的新镜像上传，这个应用不需要在构建或事件部署时知道该存储桶，但 Knative Eventing 会在运行时将该事件源轻松“绑定”到这个应用。

以下两节提供了 OpenShift 中各种类型的无服务器应用的示例。

平台服务 – OpenShift Serverless – Knative Serving 示例

我们来演示 Knative Serving 如何实现应用的自动扩展，特别是，如何在没有请求时缩容为零。我们可以利用这样一个非常简单的应用，即一个提供 ASCII 宠物图片的网页：

- php-ascii-pets GitHub 存储库

从 Git 添加这个存储库非常简单，红帽 OpenShift 会自动检测兼容的 PHP 构建器镜像。

OpenShift 自动检测如何构建这个项目。

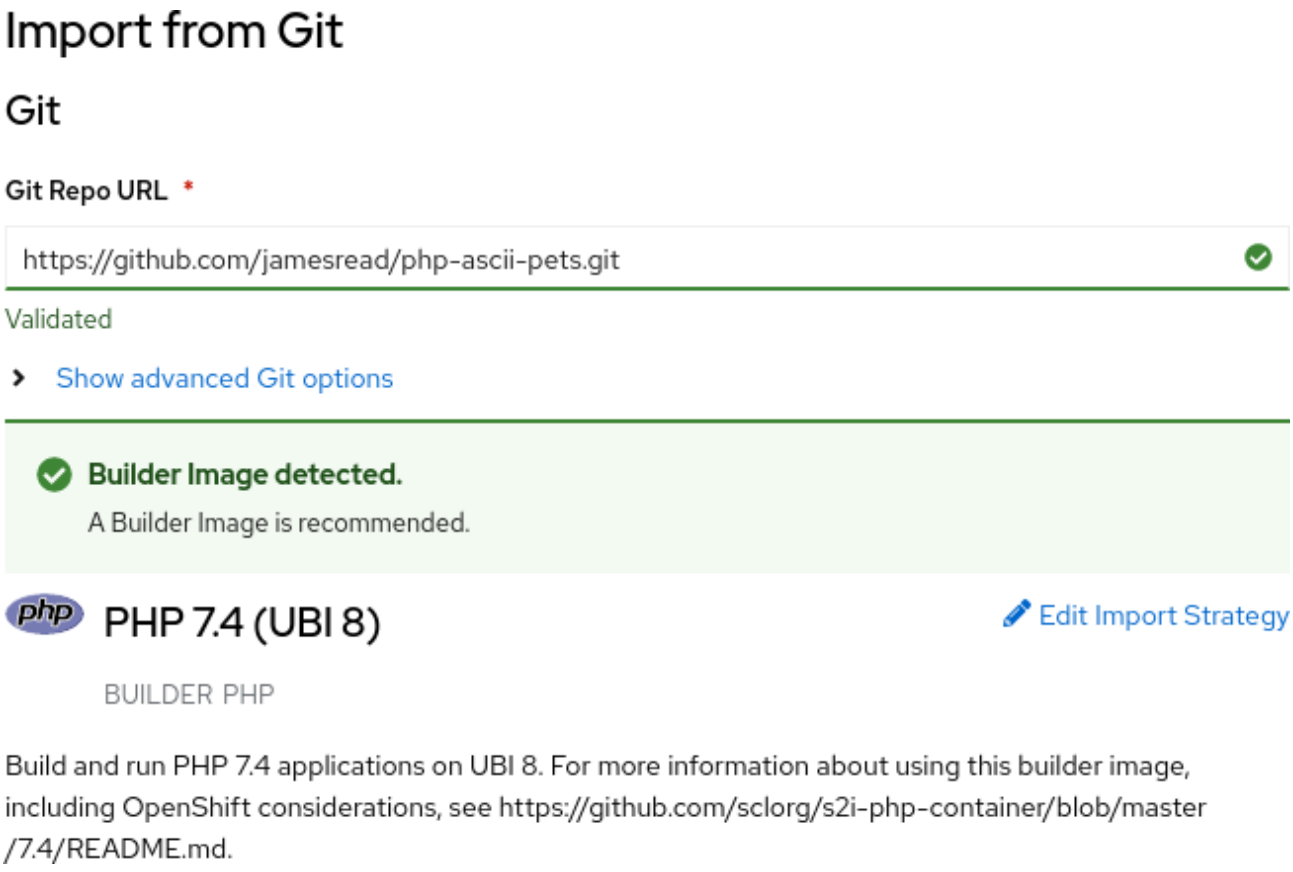


图 8.7：自动检测并建议构建器镜像

选择部署类型是决定了能不能成为“无服务器”部署的重要环节。请注意，安装了 OpenShift Serverless 后，“无服务器”部署便可供使用了。

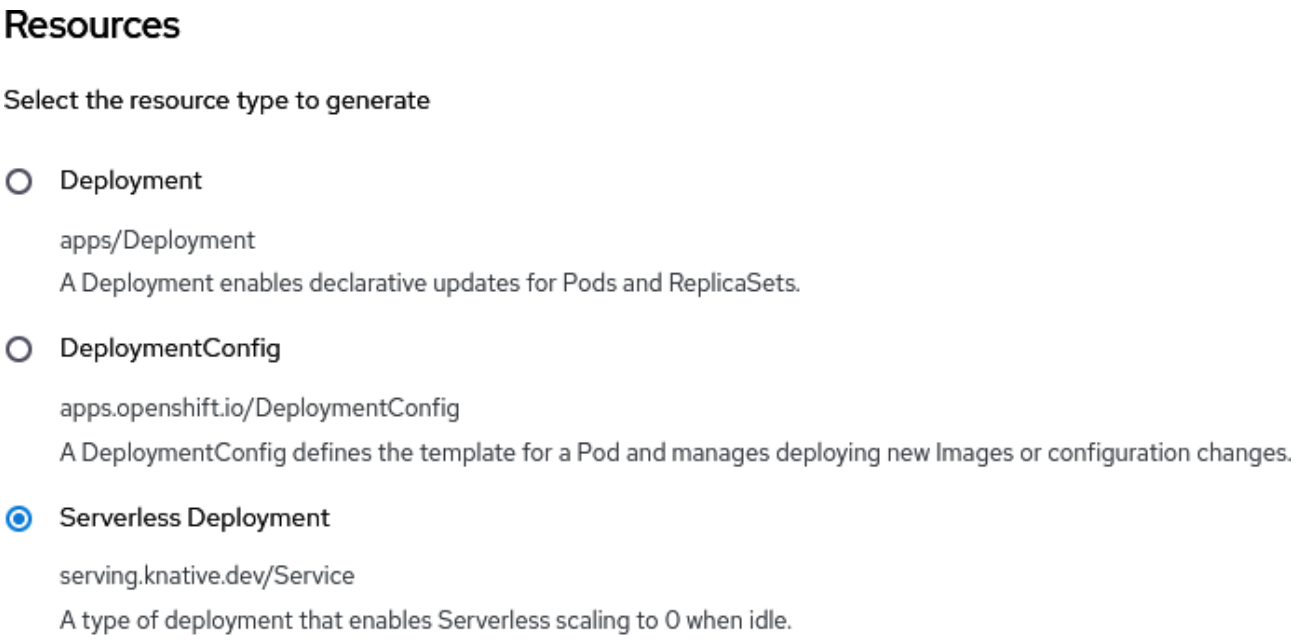


图 8.8：无服务器部署类型

第一个构建会需要一些时间来完成。构建完成后，应该部署了一个容器集。其拓扑视图应如下所示：

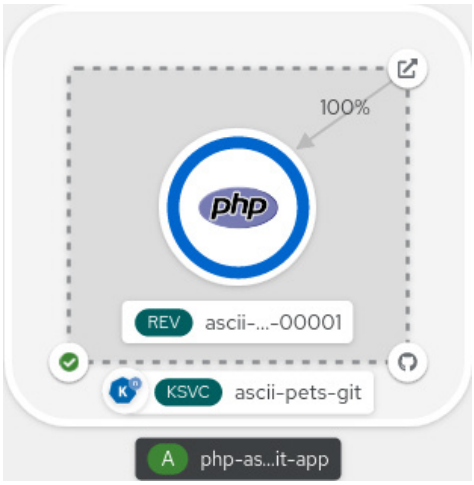


图 8.9：一个 Knative Serving 应用

实际的应用页面如*图 8.9* 中所示。这是一个非常简单的应用，但“宠物”（即 ASCII 图像）已绑定至容器集主机名。在这个简单的演示版应用中，当我们给予应用许多额外负载时，Knative Serving 会生成更多容器集，在视觉上，您会看到不同的“宠物”被供应出来！

但是，如果应用在一定时间里没有接收到任何请求，Knative Serving 会将这个应用缩容为零。查看拓扑视图就会发现，应用已不再处于运行状态。

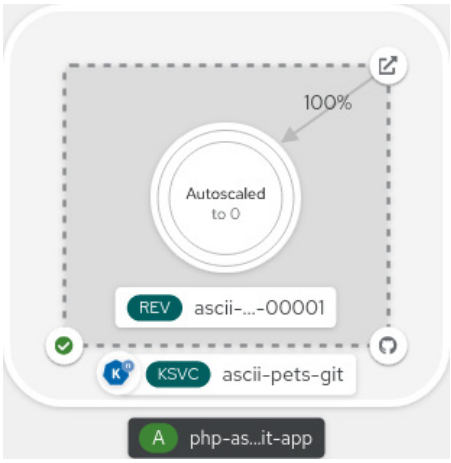


图 8.10：一个利用 Knative Serving 的无服务器部署，其中的应用已缩容为零

最后，如果有人访问该页面，那么应用就会自动重新扩展到 1 个副本，或者 2 个、3 个或更多个副本，具体取决于 OpenShift 认为需要使用多少个实例来满足需求。

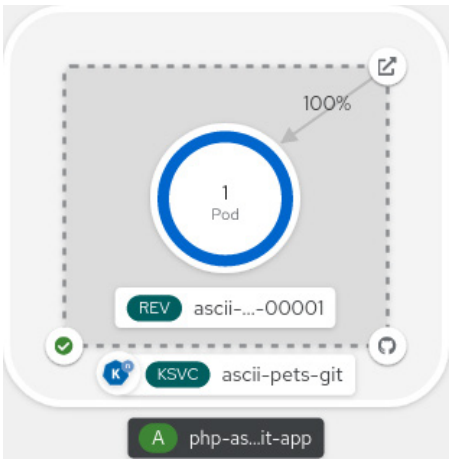


图 8.11：自动扩展以响应流量

红帽 OpenShift Serverless 加上 Knative Serving 可以让组织轻松地实现动态扩展和收缩，不仅需要的额外配置非常少，而且应用也无需进行任何更改。这一点非常有用，可让部署保持在恰当的规模，防止不必要地消耗资源并为之付费。

扩展阅读

- 关于 OpenShift Serverless
- learn.openshift.com，上面提供了 OpenShift Serverless 相关课程
- Knative 社区网站

平台服务 – OpenShift Serverless – Knative Eventing 示例

继续以上一章中的示例应用和基本内容为基础，OpenShift Serverless 还可根据除了入口流量之外的指标来扩展应用。特别是在事件驱动架构等场景中，需要能够扩展应用的实例来处理消息队列中的事件，或者基于计时器来唤醒。

借助同样的部署，可以在 OpenShift 控制台中轻松地配置不同的事件源。

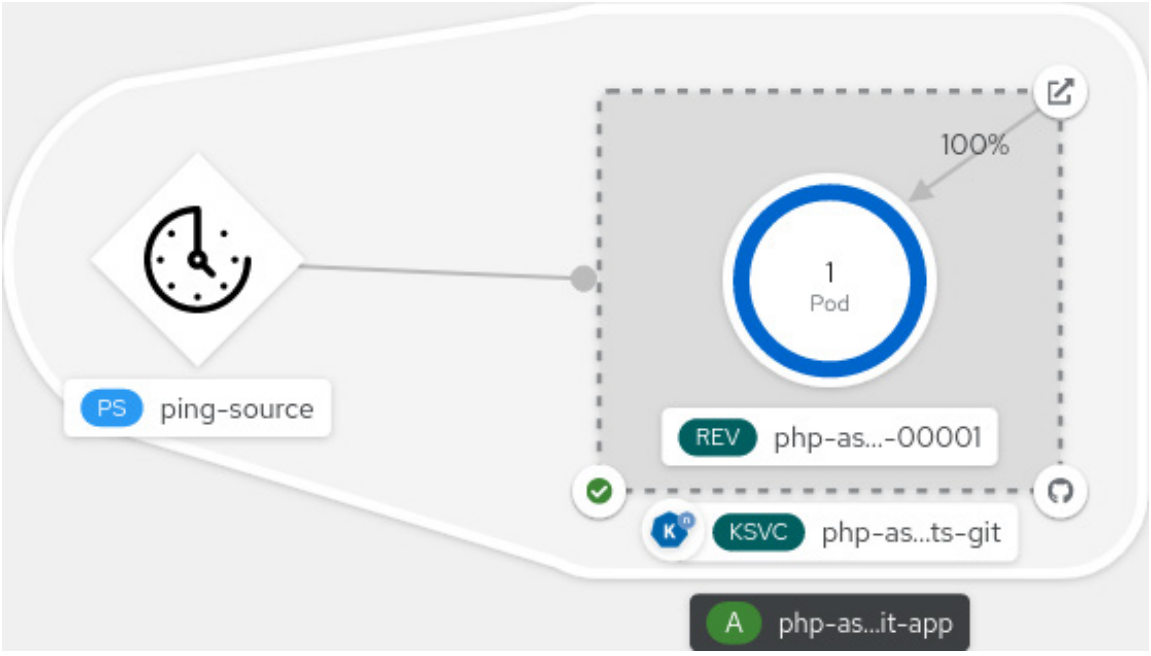


图 8.12：一个“ping”事件源，它可以根据计时器来 ping 应用

ping 事件源的一个简单用例是每天午夜“唤醒”备份作业容器。另外，定期运行的监控容器也利用这种 ping 计时器。

扩展阅读

- 五分钟阐明 OpenShift Serverless Eventing
- 关于 OpenShift Serverless

平台服务 – OpenShift 服务网格

红帽 OpenShift 服务网格基于开源的 Istio 项目，可向现有的分布式应用添加一个透明层，无需对服务代码进行任何更改。您可以在环境中部署特殊的 sidecar 代理，以截获微服务之间的所有网络通信，从而给服务添加红帽 OpenShift 服务网格支持。您可以使用控制面板功能来配置和管理服务网格。

可以通过 OpenShift 服务网格实现的用例包括：

- 借助自动 mTLS，实现对服务间通信的透明加密。
- 提供服务的多个版本，例如实现 A/B 测试。
- 借助 Kiali，实现对微服务通信情况的可观察性。
- 借助 Jaeger，追踪服务之间的调用。

与 OpenShift 的所有其他平台功能一样，服务网格可通过红帽 operator 来安装。

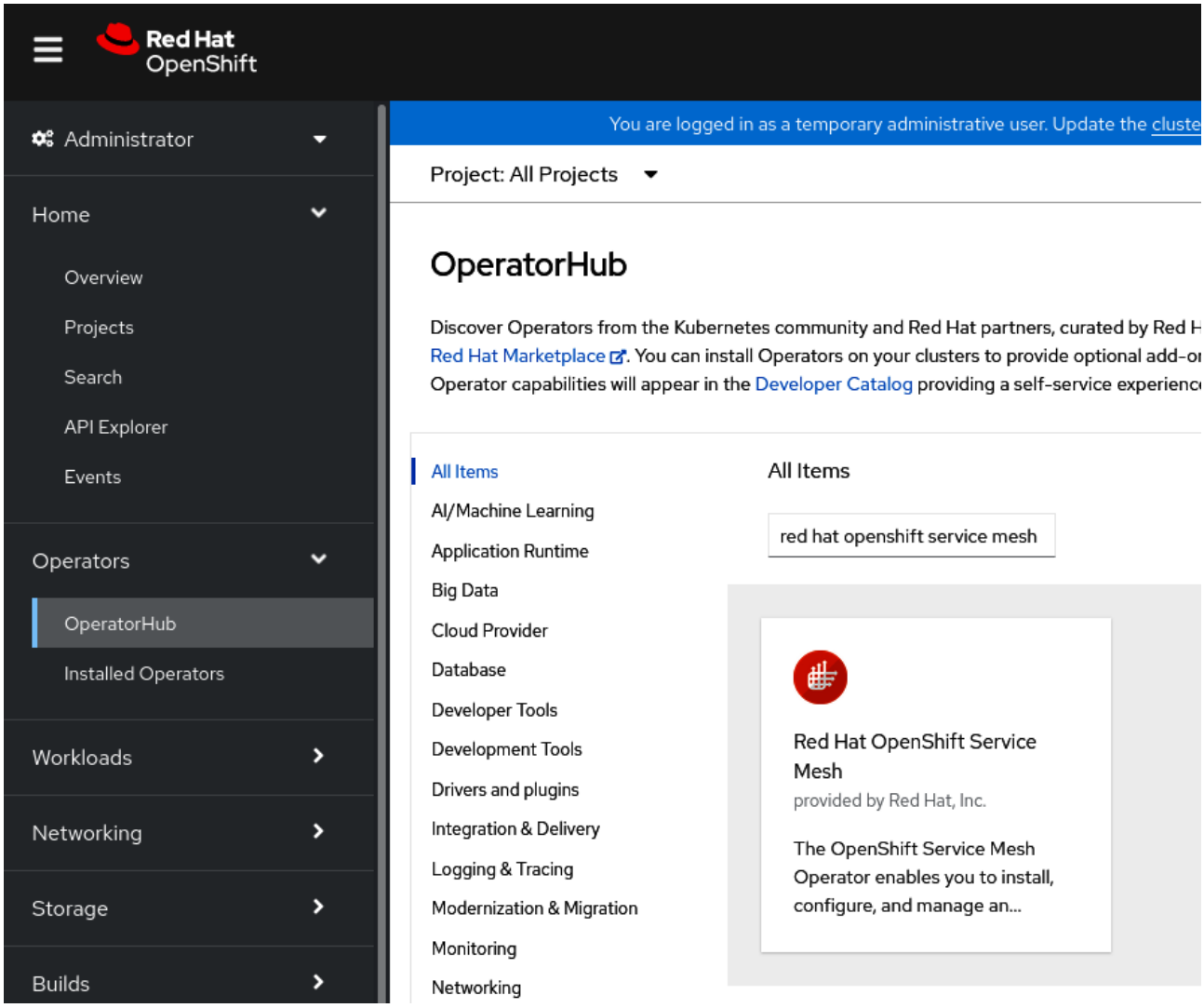


图 8.13：通过 OperatorHub 安装服务网格

OpenShift 服务网格文档中提供了一个绝妙的示例应用，名为 BookInfo 演示。这是一个模拟书店的简单应用，在 OpenShift 上运行。

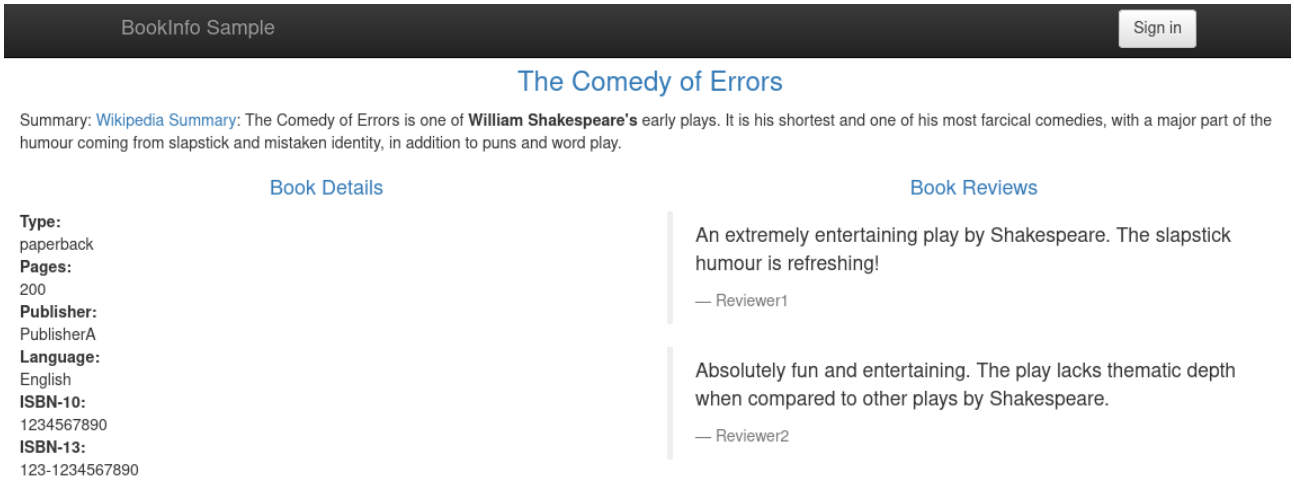


图 8.14：BookInfo 应用

要使 BookInfo 应用能够支持服务网格，需要创建一个服务网格控制平面。这可以通过 OpenShift 图形界面来轻松完成，如*图 8.15* 所示：

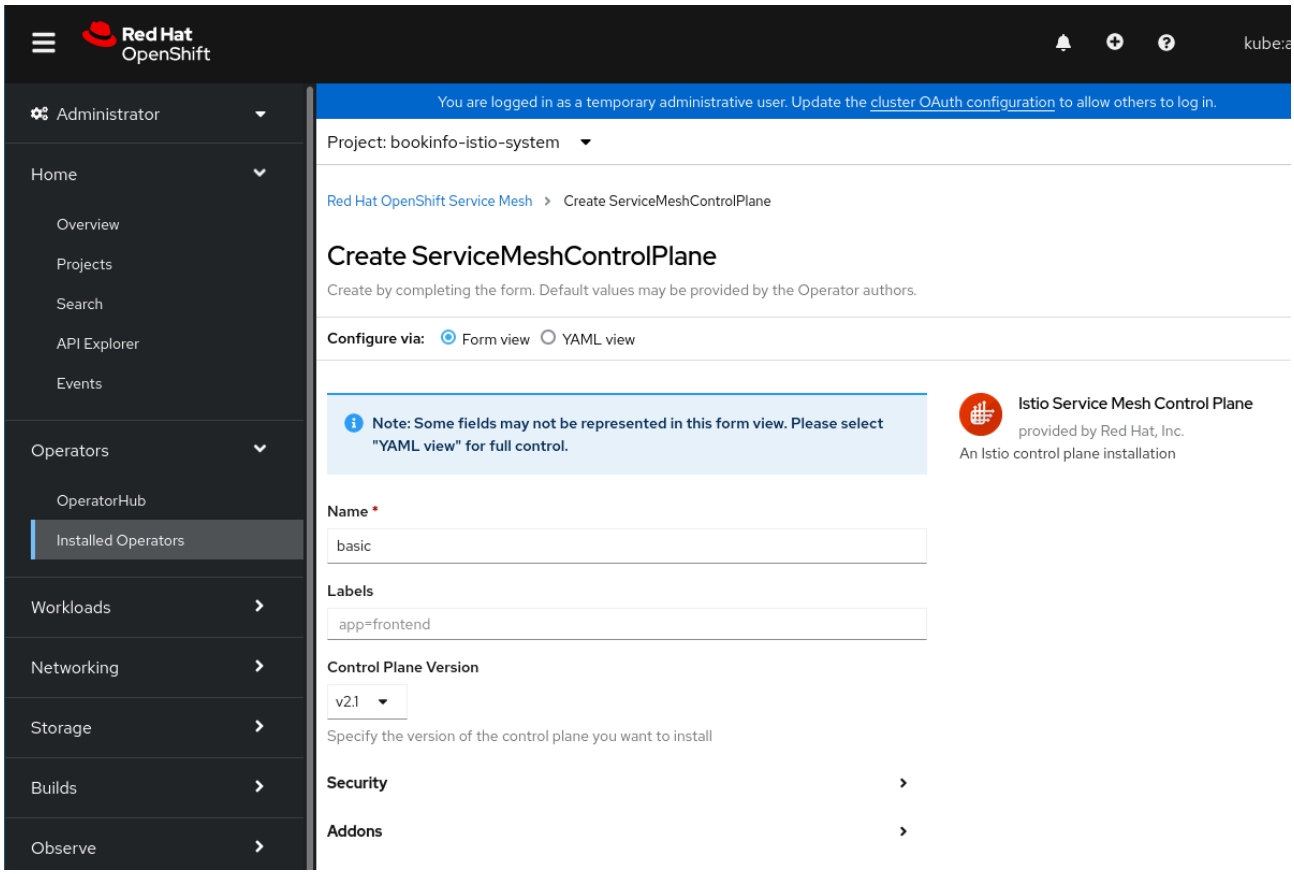


图 8.15：在 BookInfo bookinfo-istio-system 项目中设置控制平面

BookInfo 项目通过控制平面接受入口流量，本例中创建了一个单独的项目来承载控制平面，其名为 bookinfo-istio-system。

服务网格控制平面不需要与您的项目分开，甚至也可在多个项目之间共享一个服务网格控制平面。

在以下两个小节中，我们将更加细致地探索两个服务网格用例，即可观察性和分布式追踪。

平台服务 – OpenShift 服务网格 – 借助 Kiali 实现可观察性

通过观察组成此应用的底层容器集，您可以发现它由 6 个微服务构成，如红帽 OpenShift 的 Topology 视图中所示。

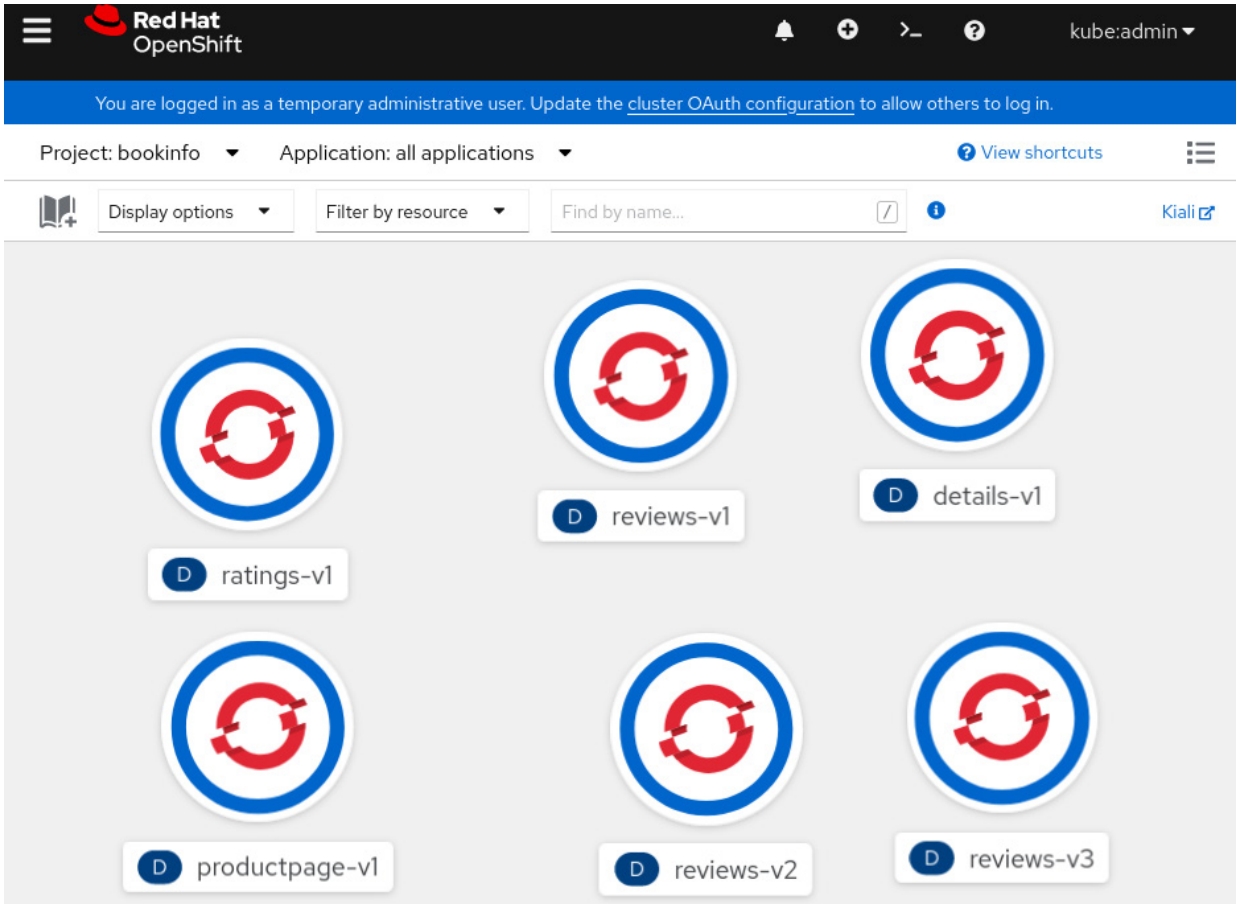


图 8.16：构成 BookInfo 应用的 6 个服务

虽然此视图提供了有用的信息，但没有真正传达这些服务是如何连接的。相比之下，就体现出服务网格的第一个优势，即：可观察性。如果我们打开一个附带服务网格的稍有不同的控制台 Kiali，我们可以看到，这里对上述 6 个服务有更加完善的架构呈现。

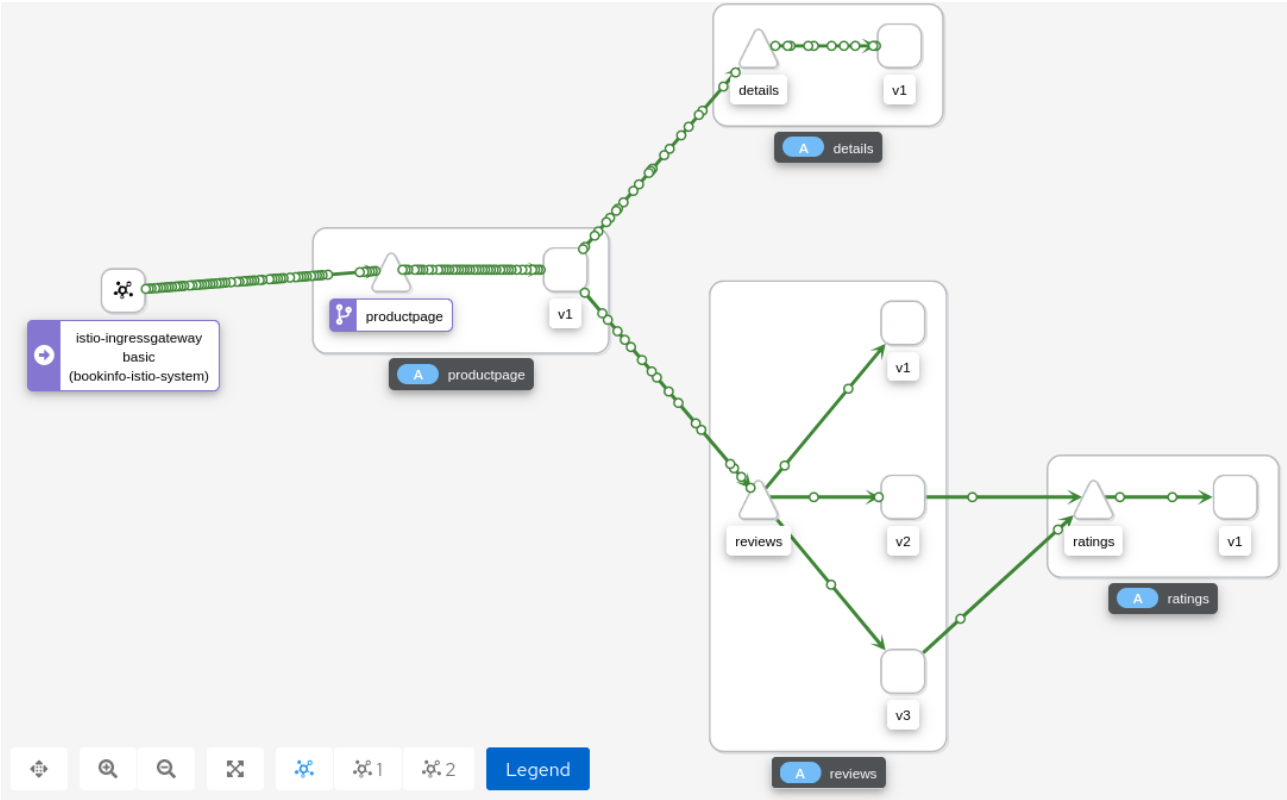


图 8.17：自动生成的应用架构图，通过服务网格来实现

此视图不仅展示了服务之间的真实连接，还提供了一个实时流量示意图。在本例中，应用正在获取数量可观的流量，连接上的圆圈动画表示请求。

进一步利用这种可观察性，管理员可以点击任何一个服务连接并查看其流量概况。本例中可以看到，流量大多具有 **HTTP OK** 状态。

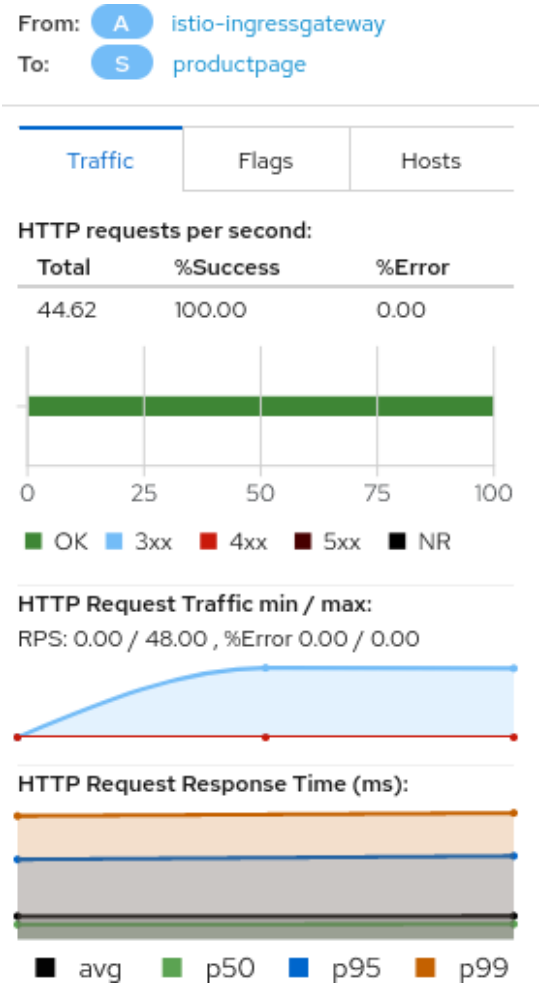


图 8.18：各种 HTTP 状态

我们可以通过关闭 details 微服务并将它收缩为零个副本，给此应用引入一个人为错误：

图 8.19：使 details 服务具有零个副本来模拟故障

现在，回到 Kiali 视图中就可发现，示意图中增加了几个额外组件，但最显眼的是，与 details 服务的连接已用红色高亮显示，以此表明存在错误。

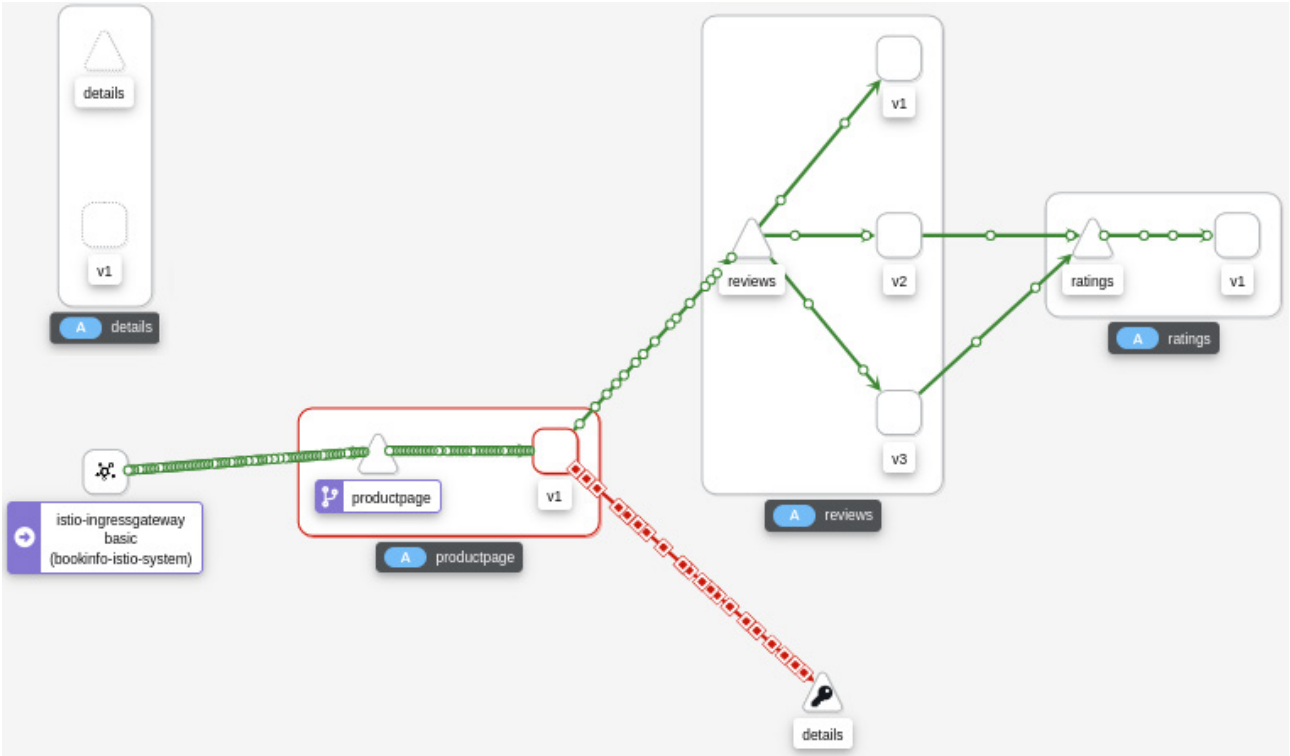


图 8.20：红色连接表明此服务出现了问题

我们已了解到，服务网格的 Kiali 可以提供可观察性方面的切实帮助。服务网格对 BookInfo 这样的小型应用只能称作有用，但随着应用变得更大型、更复杂，有时牵涉 20 个或更多的微服务以及许多不同类型的交互，才能突显服务网格和 Kiali 这样的工具的宝贵价值，而且几乎不可或缺。

平台服务 – 红帽 OpenShift 服务网格 – 借助 Jaeger 实现分布式追踪

服务网格提供的另一高价值服务是 Jaeger。这可以实现对复杂微服务应用的分布式追踪。在上一小节中，我们研究了 BookInfo 应用并且发现，当 `details` 服务脱机时，请求开始失败。

本例中的失败原因很容易辨别；`details` 服务不可用，这也是我们故意造成的！不过，当起因更加难以查明并需要进一步调查时，我们可以利用 Jaeger 的分布式追踪功能。

Jaeger 可以跟踪来自第一个服务的请求，通过一系列后续连接直到穿越整个系统的路径。虽然需要一些额外的应用代码来启用 Jaeger，但有客户端库可用而且需要更改的代码也很少，对于开发人员而言还是相对轻松的。

通过查看 `productpage` 服务（用 Python 编写），我们可以找到在该服务中实现 Jaeger 分布式追踪的相关代码。以下代码将 Jaeger 客户端库导入到 `productpage` 服务中：

```
from jaeger_client import Tracer, ConstSampler
from jaeger_client.reporter import NullReporter
from jaeger_client.codecs import B3Codec
```

导入了客户端库后，productpage 服务需要设置一个新的追踪器。以下代码在 productpage 服务中初始化 Jaeger Tracer：

```
tracer = Tracer(
    one_span_per_rpc=True,
    service_name='productpage',
    reporter=NullReporter(),
    sampler=ConstSampler(decision=True),
    extra_codecs={Format.HTTP_HEADERS: B3Codec()}
)
```

最后，依然是在 productpage 服务中，我们告知 Jaeger 我们想要创建一个新的 span，即对几个服务的新请求：

```
span = tracer.start_span(
    operation_name='op', child_of=span_ctx, tags=rpc_tag
)
```

Jaeger 而后会在几个服务中跟踪这个追踪器。这些也需要经过调整后才能与 Jaeger 配合，但最热门编程语言也有可用的客户端库。

productpage 服务的完整源代码可以在 GitHub 上找到。

检查代码有两个选项：利用 Jaeger 客户端库，该选项现在被视为已弃用；或者使用来自 OpenTelemetry 项目的开放标准选项，这是首选选择：

- OpenTelemetry 库

完成了对应用的这一检测后，您会看到类似如下的追踪：

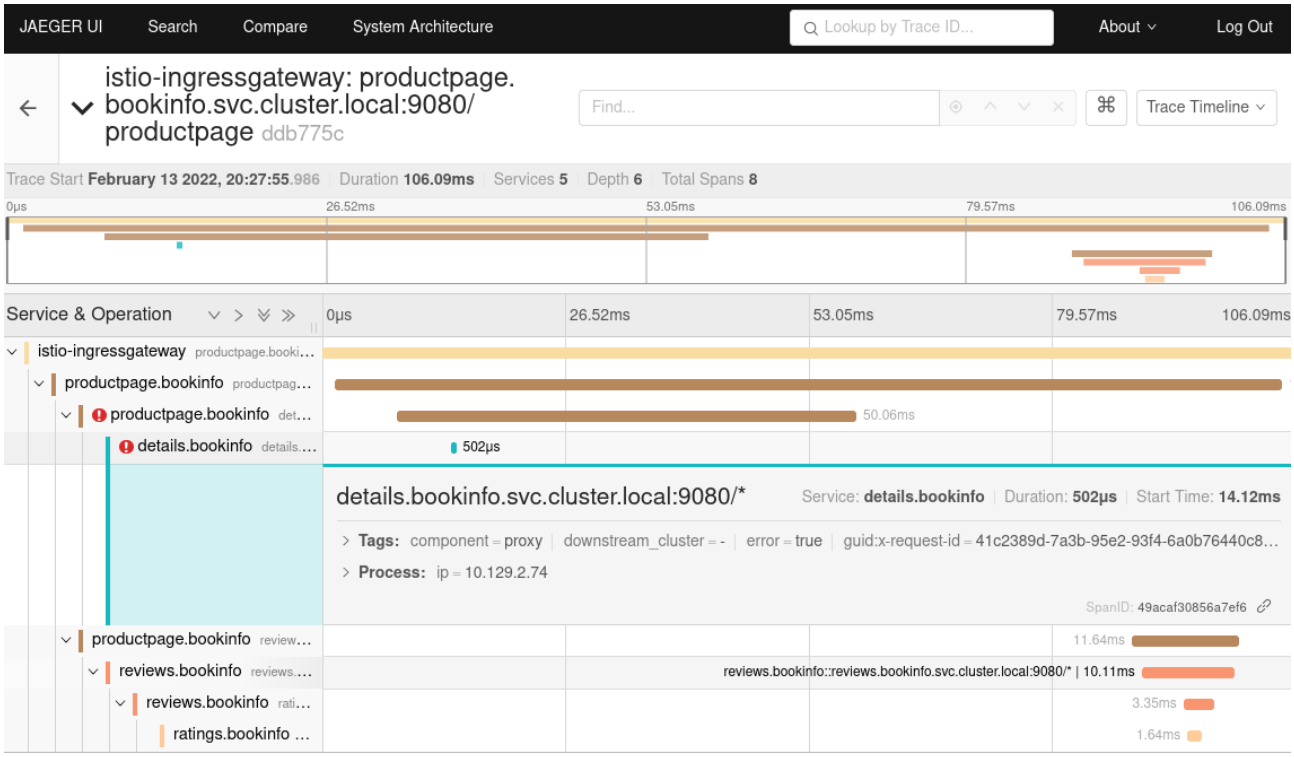


图 8.21：调用追踪

视图中显示一个调用追踪，与 Kiali 的执行结果非常相似，但这里采用层级视图来展示。显示的每一行都可展开，以显露有关请求持续时间、开始时间、源 IP 地址和其他几项细节的详细信息。如同前文所述，这种级别的信息在处理复杂的微服务应用时必不可少。

再次查看我们之前尝试诊断的错误，追踪视图清晰地显示 `details` 微服务正在遇到问题。本例中的错误很明显，但通过展开详情视图，我们可以发现服务正在发出 `HTTP 503` 错误代码。

图 8.22：错误详细信息

HTTP 503 是表示“服务不可用”的标准错误代码。本例中的原因很简单，服务已收缩为零个副本。重新扩展服务就可以复原服务了。

随着微服务应用变得更加复杂，结合使用 Jaeger 和 Kiali 这两个工具可以发挥巨大作用。它们作为 Azure 红帽 OpenShift 服务的一部分提供，不需要额外付费或订阅。

摘要

本章介绍了红帽 OpenShift 这个应用平台提供的几个关键增值服务，与“空白”的 Kubernetes 环境形成鲜明对照。虽然可以通过在 OpenShift 中安装所有对应的上游开源项目，来复刻与 OpenShift 类似等级的功能，但集成、生命周期和支持这些复杂功能需要通过 Azure 红帽 OpenShift 才能实现。

各种各样的平台服务、应用服务、数据服务、开发人员服务和集群服务最终可以提高开发人员和运维人员在使用 Kubernetes 和部署多个复杂企业应用时的工作效率。

第 9 章

集成其他服务

应用无法全部存在于红帽 OpenShift 内，这很常见；大多数应用以这样或那样的形式依赖于 Azure 上的外部数据库或服务。

Azure 红帽 OpenShift 不会在这里“断开”任何种类的连接。例如，如果应用依赖于 Azure CosmosDB，它依然能够从 Azure 红帽 OpenShift 连接到外面的 Azure CosmosDB，无需对应用进行任何更改。根据您的应用和组织，可能需要独立于 Azure 红帽 OpenShift 部署这些外部依赖项，或者同时部署它们。

如果您认为最好是将这些外部依赖项与您的应用一同部署，那么 Azure Service Operator 可以大大简化这个过程。不必使用 Azure CLI、Azure Cloud Shell 或 ARM 模板，您可以利用 Azure Service Operator，将这些资源当做是 Kubernetes 中原生的一样来部署。

Azure Service Operator

当开发人员或管理员在 Azure 红帽 OpenShift 上部署应用时，Azure Service Operator 也是一个让工作变得简单的 operator。它的一个优势是，不必离开 OpenShift 控制台就能轻松置备 Azure 服务，从而更快速、更轻松地部署可能具有 Azure 服务依赖项（如 Azure CosmosDB）的应用。

使用 Azure Service Operator 的另一个优势或许更加重要，它允许这些 Azure 服务相关依赖项以基础架构即代码的方式通过标准的 Kubernetes YAML 文件，与 OpenShift 应用的定义存储在一起。

其运作方式很简单：Azure Service Operator 查找与 Azure 上资源的定义相匹配的新 **CRD**（在 YAML 中定义）。然后，Azure Service Operator 将此 YAML 转化为必要的 Azure API 调用，以创建 Azure 上所要请求的资源。安装了此 operator 后，用户可以浏览 OpenShift Developer Catalog 并看到许多常用 Azure 资源的创建屏幕，例如 Azure 公开 IP 地址、Azure SQL 数据库或 Azure 防火墙。

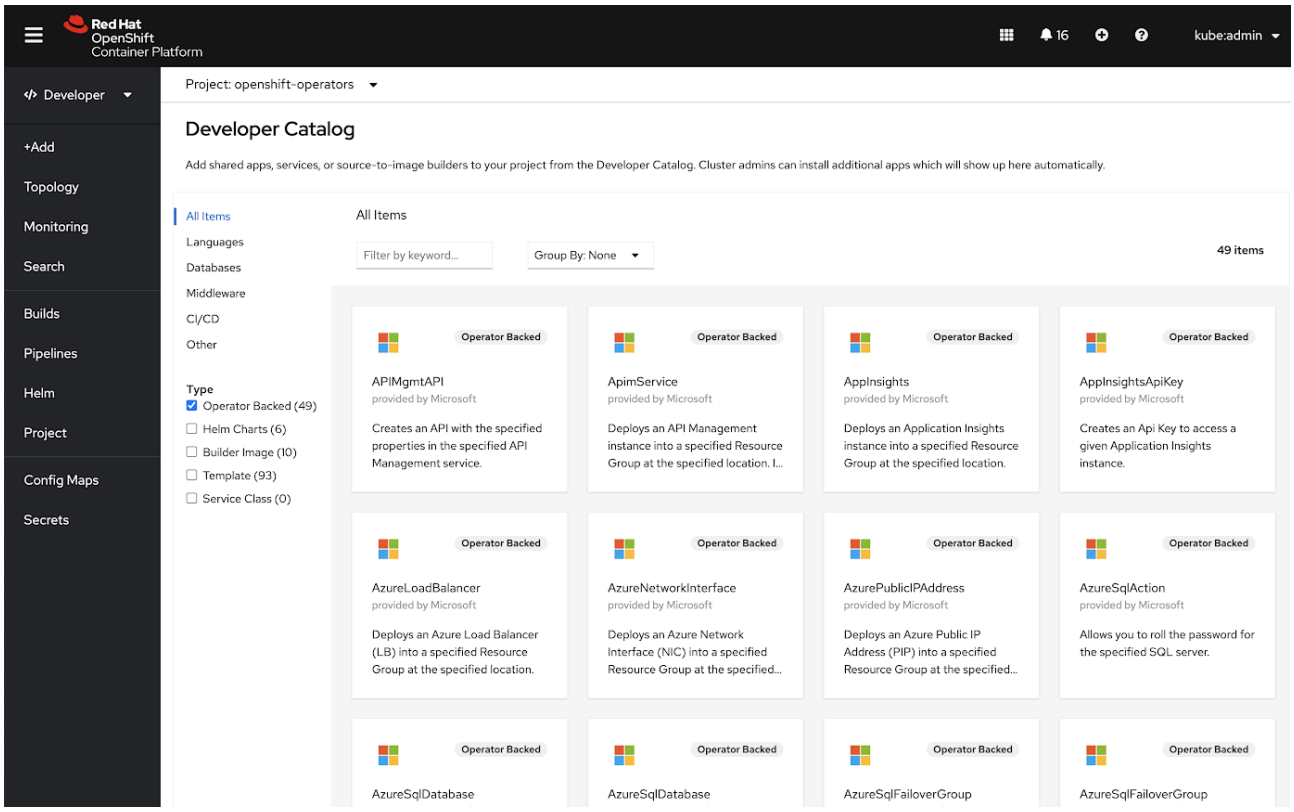


图 9.1：Azure Service Operator 公开的一些 Azure 服务

Azure Service Operator 可以通过 OperatorHub 安装，所有 OpenShift 用户均可使用。

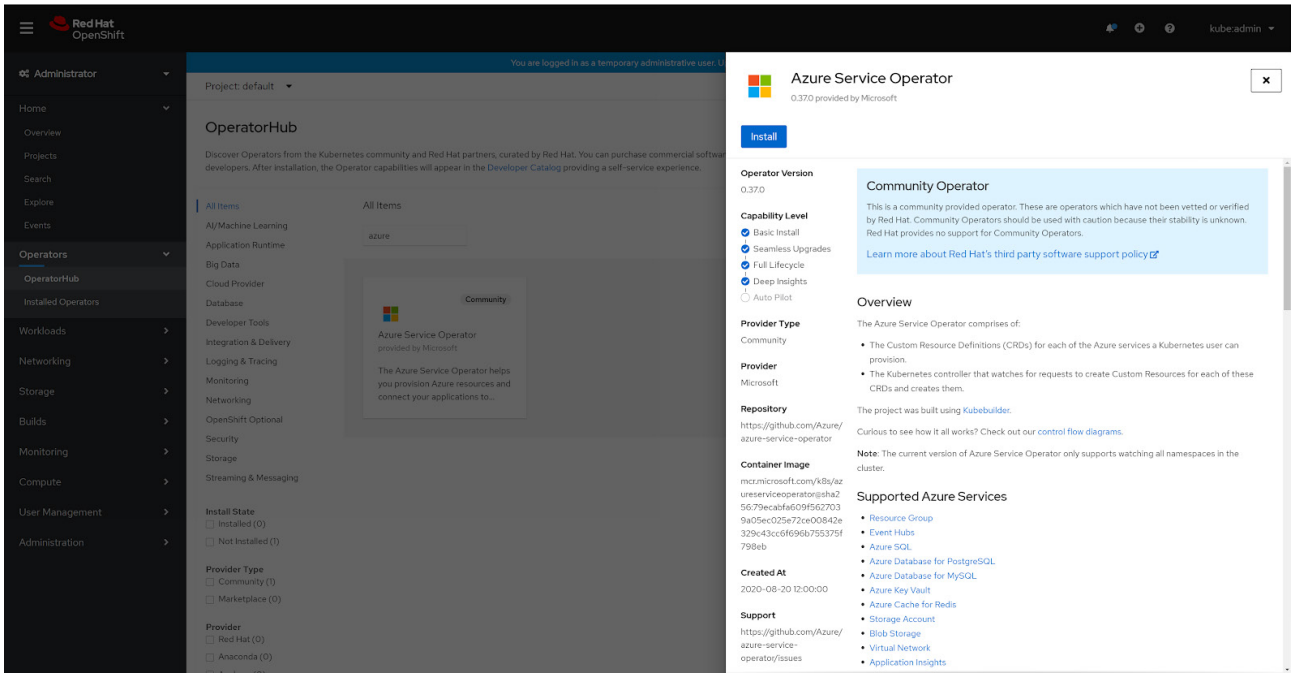


图 9.2：Azure Service Operator 的安装屏幕，背景中为 OperatorHub 资源库

使用 Azure Service Operator 并非 Azure 上所运行服务的强制要求，而且务必要清楚一点，底层 Azure 服务是原生部署到 Azure 上的，而非部署到 OpenShift 上。不过，Azure Service Operator 可以使含有 Azure 服务依赖项的应用的部署变得更加快速、更加简单。

Azure Service Operator 的一个常见用例是将依赖数据库与应用一同部署。例如，对于使用 Azure CosmosDB 实例的 Web 应用，可通过 Azure Service Operator 将 Azure CosmosDB 作为 OpenShift 上应用部署的一部分来部署。Azure Service Operator 包含对 Azure SQL Database、Azure Database for MySQL、Azure PostgreSQL 以及其他几种数据库的支持。

假设已安装了 Azure Service Operator，下列 Kubernetes 清单就可与 OpenShift 应用一起存储并用于置备 MySQL 数据库：

来源：[摘录自 Azure Service Operator 示例存储库]

```
apiVersion: azure.microsoft.com/v1alpha1
kind: MySQLServer
metadata:
  name: aso-wpdemo-mysqlserver
spec:
  location: eastus2
  resourceGroup: aso-wpdemo-rg
  serverVersion: "8.0"
  sslEnforcement: Disabled
  minimalTLSVersion: TLS10 # Possible values include: 'TLS10', 'TLS11', 'TLS12', 'Disabled'
  infrastructureEncryption: Enabled # Possible values include: Enabled, Disabled
  createMode: Default # Possible values include: Default, Replica, PointInTimeRestore (not
implemented), GeoRestore (not implemented)
  sku:
    name: GP_Gen5_4 # tier + family + cores eg. - B_Gen4_1, GP_Gen5_4
    tier: GeneralPurpose # possible values - 'Basic', 'GeneralPurpose', 'MemoryOptimized'
    family: Gen5
    size: "51200"
    capacity: 4
```

将 Azure Service Operator 用于涉及许多具有复杂依赖项的 Azure 服务是不太常见的，这样的情形更加适合使用 Azure ARM 模板或 Bicep 之类的工具。

如需 Azure Service Operator 的更多介绍，请查阅以下博客文章：

- 博客文章 – 2020 年 9 月
- OperatorHub.io 上的 Azure Service Operator
- GitHub 上的 Azure Service Operator

与 Azure DevOps 集成

与 OpenShift Pipelines 一样，许多用户希望将 Azure 红帽 OpenShift 与 Azure DevOps 集成。这两种解决方案可以和谐共存，一些项目使用其中一种解决方案，有时也会同时使用这两者！有关何时使用某一种解决方案的决策由几个因素决定。

通常而言，Azure DevOps 提供与其他 Azure 工具的高级集成，但可以轻松部署到 OpenShift 及其他计算资源。

另一方面，OpenShift Pipelines 是集成良好并随同 OpenShift 提供的产品，可为用户带来一致的多云体验。

Azure DevOps 将 OpenShift 看作与任何其他 Kubernetes 集群一样。因此，所有标准的 Kubernetes 接口和 API 都应该正常运作。

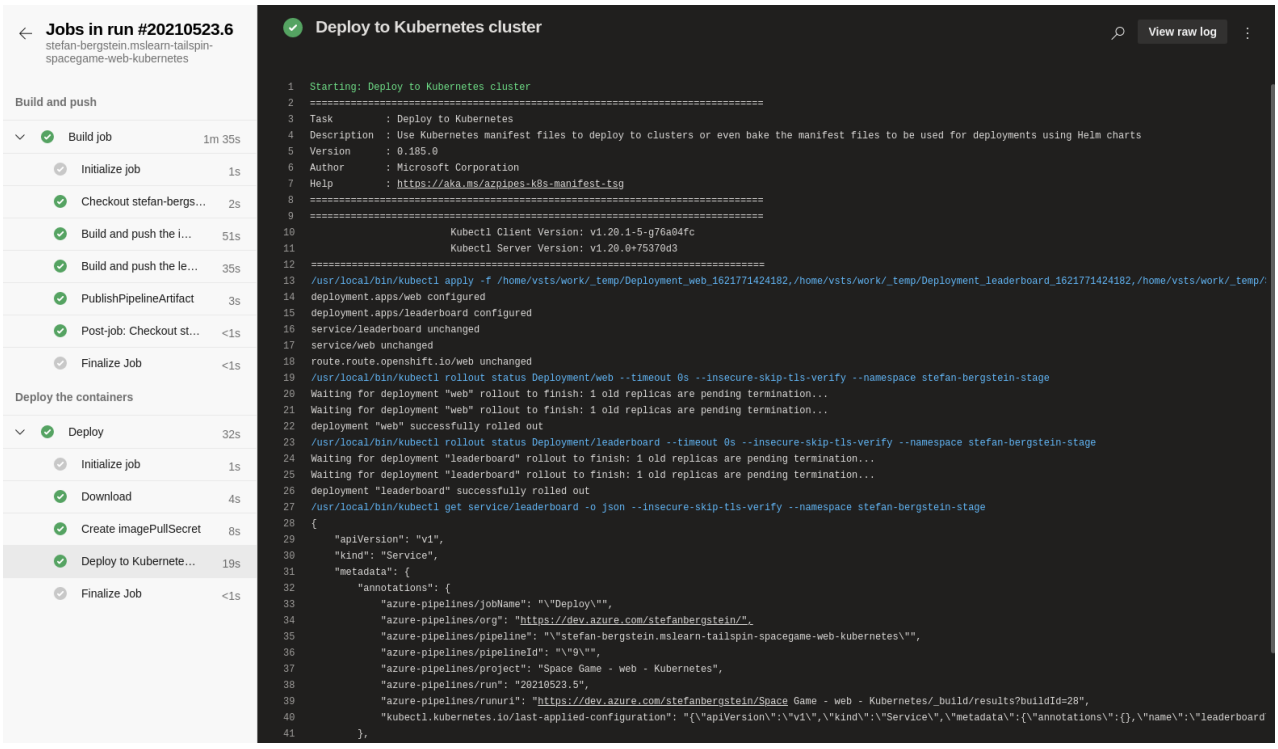


图 9.3：一个 Azure DevOps 管道将内容推送到 OpenShift

扩展阅读

以下社区博客文章提供了有关如何开始使用 Azure 红帽 OpenShift 和 Azure DevOps 的优秀教程：

- 博客文章 – 2021 年 5 月

与 Visual Studio Code 集成

作为 OpenShift 的开发人员集成价值之一，市面上提供了面向许多流行 IDE 的插件，包括 Visual Studio Code 在内。这使开发人员和管理员无需离开他们的 IDE ，就能快速、轻松地访问 Kubernetes 和 OpenShift 资源。

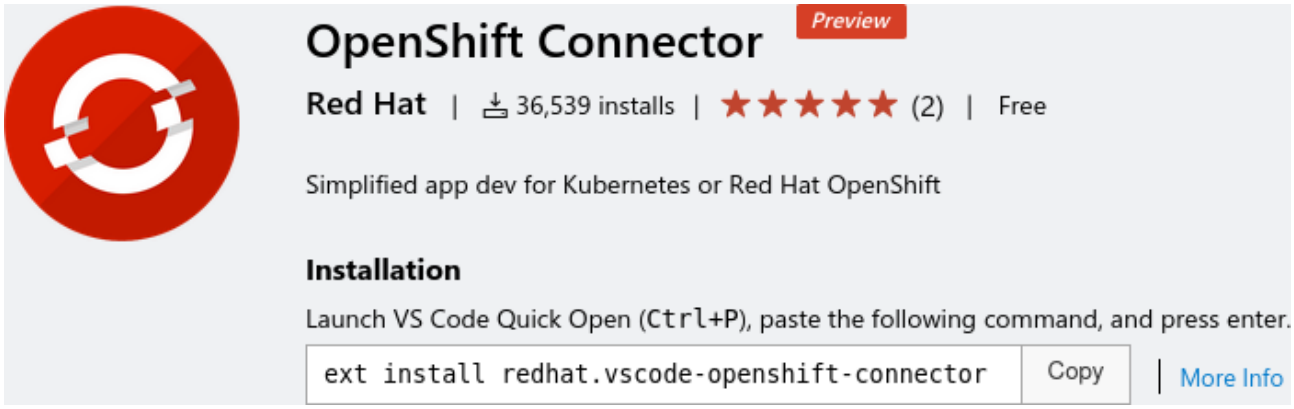


图 9.4：安装 OpenShift Connector

下载了此连接器并安装到 Visual Studio Code 中后，您可以按照提示登录您的 OpenShift 集群。以下是一个简单的“hello world”式项目，它具有与 Azure 红帽 OpenShift 的连接：

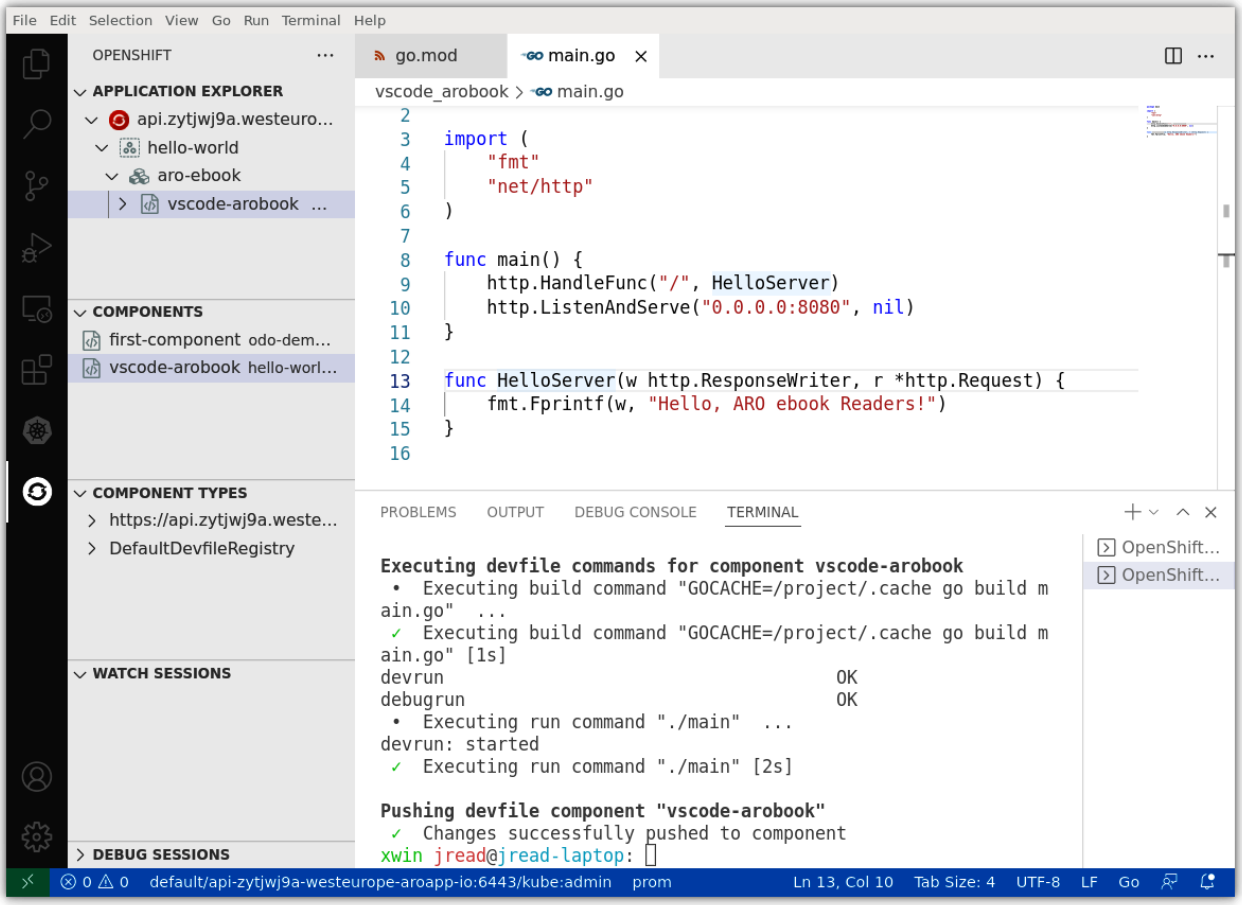


图 9.5：刚刚通过 Visual Studio Code OpenShift 连接器部署的 Golang 项目

将 OpenShift 连接器与 Visual Studio Code 搭配使用有一个好处，可为流行的 OpenShift 工具“OpenShift Do”（通常简称为“odo”）提供一个图形化前端。这样，开发人员就可以思索更高层次的概念，不仅仅是原始的 Kubernetes 组件。开发人员无需为 Deployment 和 ReplicaSet 等细枝末节操心。前面的屏幕截图中可以看到，这是一个公开了 HTTP 服务的简单项目。

此外，连接器也包含了相应能，可以直接将代码更改推送到 OpenShift，不必先使用源代码控制了。这对快速开发有很大裨益，不必每次都要推送到源代码控制，再构建并部署一个新容器。

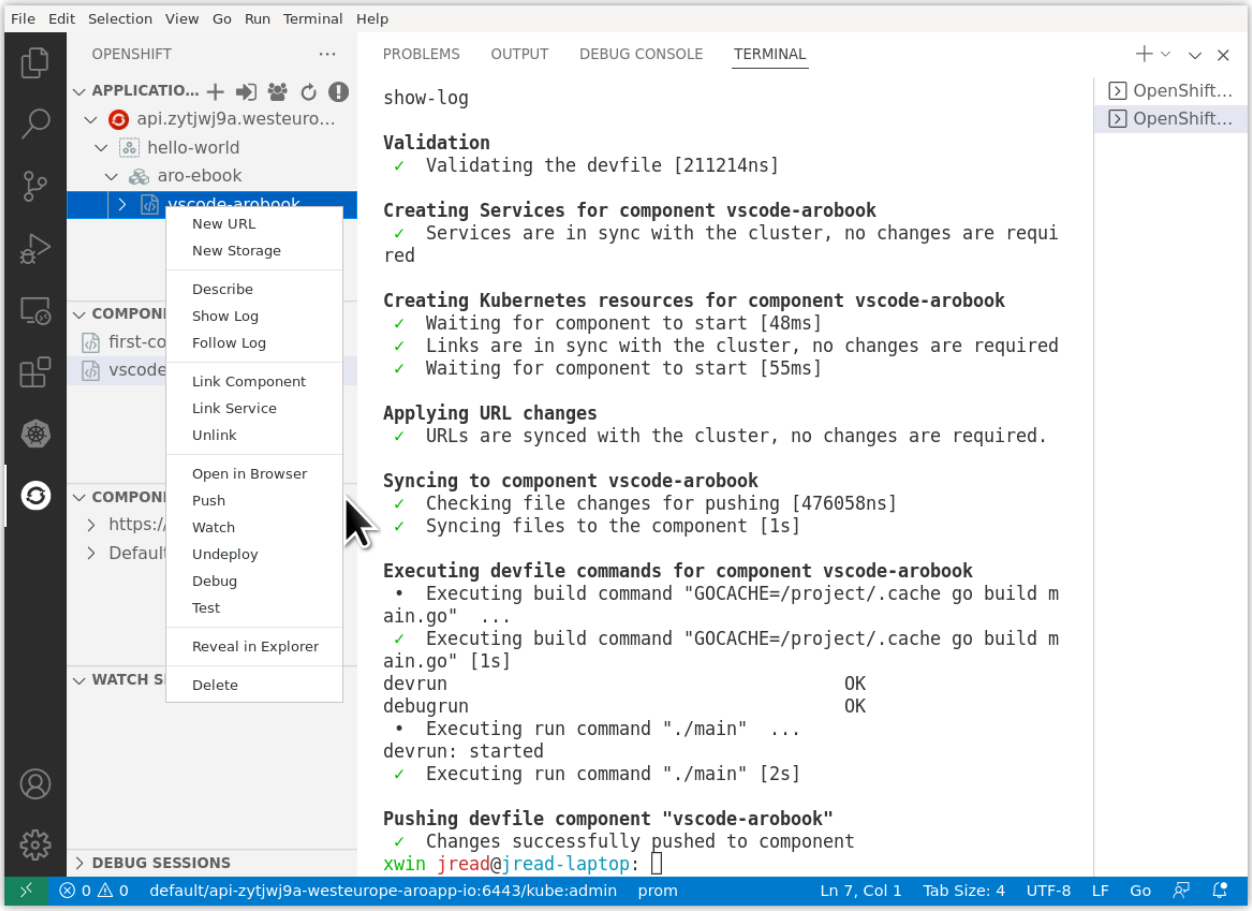


图 9.6：终端窗口中显示项目的“推送”菜单以及最近一次推送

对于选择使用 Visual Studio Code 的开发人员来说，此连接器可以直接加快开发和测试周期。

图 9.7：在 OpenShift 上运行的简单 Golang 应用

当然，开发人员不仅限于使用 Visual Studio Code，许多开发人员也使用 Vim、Eclipse 和其他编辑器，但 Visual Studio Code 在喜欢“轻型 IDE”体验的开发人员中很受欢迎。

扩展阅读

- 演示视频 – 搭配使用 Visual Studio Code 和 OpenShift
- Visual Studio Code OpenShift Connector

与 GitHub Actions 集成

如今可以使用 GitHub Actions 部署到任何红帽 OpenShift 环境，Azure 红帽 OpenShift 也不例外。从任何 GitHub 存储库，前往 **Actions** → **New Workflow**，然后从可用操作列表中选择 **OpenShift**。

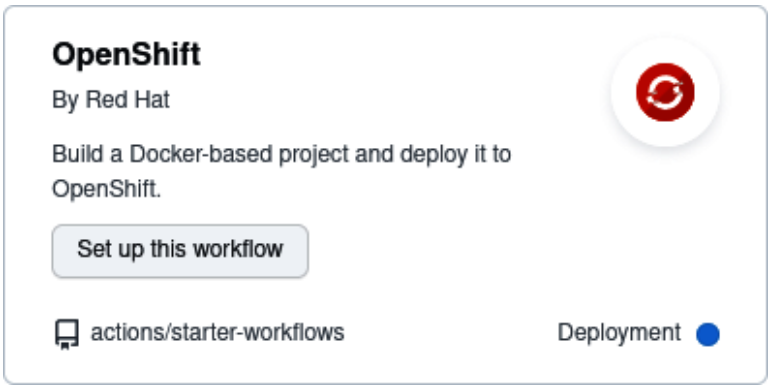


图 9.8：从 GitHub 部署到 OpenShift

默认模板附带了一组优秀的操作示例，只需设置几个环境变量就能连接到 OpenShift 集群：

```
name: OpenShift

env:
  # ✏ EDIT your repository secrets to log into your OpenShift cluster and set up the
context.
  # See https://github.com/redhat-actions/oc-login#readme for how to retrieve these values.
  # To get a permanent token, refer to https://github.com/redhat-actions/oc-login/wiki/Using-
a-Service-Account-for-GitHub-Actions
  OPENSHIFT_SERVER: ${{ secrets.OPENSHIFT_SERVER }}
  OPENSHIFT_TOKEN: ${{ secrets.OPENSHIFT_TOKEN }}
  # ✏ EDIT to set the kube context's namespace after login. Leave blank to use your user's
default namespace.
  OPENSHIFT_NAMESPACE: ""
  ...
```

这里提供了一篇介绍具体操作的优秀博客文章，以及一个演示视频：

- 博客文章
- GitHub Actions 与 OpenShift 演示视频

摘要

本章介绍了我们发现客户搭配 Azure 红帽 OpenShift 使用的许多常见集成。如章节引言中所述，我们可以通过标准的 OpenShift API 与本章未列出的更多服务进行集成，还可以使用 OperatorHub 中列出的 operator，与许多服务实现轻松集成。

在下一章中，我们将分享一些关于培训应用团队和在组织中推动采用此服务的一些最佳实践和建议。

第 10 章

启动工作负载和培训团队

当您完成了*置备前*步骤，置备了 Azure 红帽 OpenShift，并且执行了必要的置备后步骤后，唯一剩下的就是支持并培训开发人员并将应用引入到集群上。具体做法最终取决于您的组织文化；一些开发和应用团队热衷于“寻根究底”并开始上手操作，另一些团队则可能乐于接受更多指导。

本章可以充当一组检查清单，协助您奠定良好基础，确保开发和应用团队在使用 Azure 红帽 OpenShift 时更加快速地准备好上线运作。

当您根据自己的组织结构和文化调整做法时，务必要考虑扩充这些检查清单。

清查清单：引入之前

在引入新的工作负载前，务必要能够告知应用团队或工作负载负责人，Azure 红帽 OpenShift 已经完成部署并且其设计符合组织的最佳实践：

- Azure 红帽 OpenShift 已经部署到 Azure 登陆区域，或者核准用于组织应用的其他 Azure 环境。
- 集群具有所有必需的“Day-2”设置，例如存储类和身份验证（如*“第 6 章：置备之后 - Day 2”*中所述）。
- 集群已进行全面配置并由您的平台团队提供支持，而且还享有红帽和微软联合提供的集成支持。
- 集群已经更新到可用的最新稳定版本，并且补丁已安装妥当。之后也会持续安装补丁。

- Azure 红帽 OpenShift 集群已连接了必要的资源：
  - 连接了本地环境（通过 Azure ExpressRoute 或 VPN）
  - 连接了企业工件存储库（如 Java Maven 存储库）
  - 连接了公共互联网（用于在应用构建期间下载依赖项）

启动试点工作负载

在组织中推广 Azure 红帽 OpenShift 的一种常见模式是启动至少两个试点工作负载：

- **有意义的最小工作负载** – 最好是在 SRE 团队支持下，第一个试点工作负载应当是一个小巧的知名应用，并且技术要求十分简单。这要稍高于简单的“Hello World”应用，应用通常应在组织中具有真正的业务负责人。不过，第一个工作负载的重点是为了检查 Azure 红帽 OpenShift 集群是否能够满足集群层面上的业务需求，例如 Azure 连接、Azure Active Directory 登录，以及简单地识别“容易实现的目标”（即，每个应用可能会遭遇的常见问题）。
- **有意义的参考工作负载** – 部署了最小的简单工作负载后，如果第二个工作负载足够复杂且有意义（对业务很重要，甚至是业务关键型），那么对进一步采用会有切实帮助。这个工作负载将有助于识别和解决各种各样的问题，如重要数据库连接、性能测试和规模化日志 / 指标等。重要的是，这个有意义的参考工作负载之后要用作组织的**内部参考**。这将有助于未来的开发和应用团队树立对 Azure 红帽 OpenShift 平台的信心。

当然，还有其他启动首批工作负载的模式可能适合您的组织。目标可能需要瞄准即将退役的数据中心中的应用，或者在旧的 Java 应用服务器上运行的应用。

检查清单：与其他团队的培训会议

在集群中引入新的开发或应用团队时，最好是举行一次培训会议：

- 创建供团队使用的一个或一系列命名空间。
- 为团队做红帽 OpenShift 的简短演示，例如指出从 GitHub 部署（或同等做法）等关键功能。
- 确认集群具有充足的空闲容量来部署目标工作负载（CPU、RAM 和存储等）。
- 确认团队成员能够登录集群，连接 Azure Active Directory 是简化这项工作的有效方法。
- 提供负责管理集群的 SRE 团队（或类似团队）的联系详细信息。
- 提供 OpenShift 入门链接的列表：
    - http://learn.openshift.com
    - https://github.com/openshift-labs/starter-guides
    - http://docs.openshift.com

清查清单：定期健康检查会议

在开发或应用团队开始部署到集群后，定期举行健康检查会有帮助。这可以融入到日常的站会中，或者每周花上 30 分钟已经足够。您可能需要检查以下几项：

- 团队的总体进展如何 — 有没有部署了任何应用？
- 最近有没有遇到与使用集群相关的问题（红帽 OpenShift 知识或连接问题）？
- Azure 或集群有没有发生任何服务中断，给体验造成负面影响？
- 提醒 SRE 团队解答问题的时间和联系方式。

考虑您在针对类似项目举行定期健康检查会议时可能已在使用的其他检查事项。将您认为可能有用的要点添加到这个检查清单中。

反模式：开发人员游乐场 / 沙盒

在组织中鼓励开发人员采用的一个常见“反模式”是提供沙盒型的环境，通常称为“开发人员游乐场”，往往很容易推动开发人员和应用团队开始采用 Azure 红帽 OpenShift。如果没有更多的跟进支持或资源，Azure 红帽 OpenShift 也许是一个令人望而生畏的环境，有太多难以消化的复杂性问题。在许多情形中，采用“沙盒”模式可能会造成云计算支出浪费和空集群。

不过，如果您组织的开发团队确实有“沙盒化”传统，可以尝试下面这些提示并尽可能确保成功：

- 至少提供一个简短的演示来展示 Azure 红帽 OpenShift 的能力。
- 提供一些文档，至少是前文中所述的快速入门链接。
- 组织一场“编程马拉松”活动，以小型竞赛形式向开发人员发出使用 OpenShift 进行构建的挑战。
- 提供扩展支持福利、SRE 团队引荐，以及指导性更强的可选培训体验。

如前文所述，按照引导式采用模式检查清单操作，更有可能促成平台被采用的结果。

Azure 红帽 OpenShift 开发人员研习班

Azure 红帽 OpenShift 开发人员研习班（https://aroworkshop.io）是一个实用的预创建研习班，可供您在组织中用于提供 Azure 红帽 OpenShift 的引导式实训体验。研习班分为两个实验练习，旨在通过引导式教程来指导刚接触 OpenShift 的开发人员或运维人员。两个实验重点介绍了 OpenShift 的关键功能，并且展示了它的用法。实验 1 是现代微服务设计简介，而实验 2 则探究这项服务的内在功能。

若要提高 Azure 红帽 OpenShift 在您组织中的认知度，一种有用的做法是在自己的内部 Azure 红帽 OpenShift 集群上使用 aroworkshop.io 内容。在研习班前讨论中，您可以说明 Azure 红帽 OpenShift 是如何部署到您组织的现有 Azure 红帽 OpenShift 订阅中的，研习班中介绍的服务用法类似于将 Azure 红帽 OpenShift 用于托管生产性应用。

摘要

本章提供了一些实用的检查清单和指导，可帮助您将工作负载和团队成功引入到 Azure 红帽 OpenShift。也描述了一个常见的反模式，即没有支持的“沙盒”集群。任何指南都难以做到一一阐明适用于每个组织的资源和检查清单项目，因此务必根据自己的需求来调整本章中的内容。

云提供了一系列充满诱惑力的服务；但是，许多服务依然被忽视或采用率不高，因为组织将太多注意力集中到技术和部署方法上。虽然了解这些主题很重要，但它们只是在生产中引进并高效采用服务时需要考量的一小部分。本章尽量重点介绍必要的培训、定期检查和类似活动如何让您更加成功地采用 Azure 红帽 OpenShift。

第 11 章

结语

您的开发和运维团队可能会将大部分时间花费在处理与集群和 CI/CD 管道的置备、设置、维护和监控相关的工作上。在开展这些工作时，他们就无法将宝贵时间投入到最擅长的事上，即让您的应用变得更加先进。

正如我们在本指南中了解到的内容，Azure 红帽 OpenShift 可以让我们部署全托管式红帽 OpenShift 集群，而不用再自行构建或管理基础架构以实施运行。我们已经看到，只运行 Kubernetes 会有一些问题，主要在于需要额外关注一些手动操作，而这些手动操作都是可以通过 Azure 红帽 OpenShift 自动完成的。

当您在决定究竟为企业选择哪种集群管理策略时，一定要考虑采用 Kubernetes 类型的平台与采用 Azure 红帽 OpenShift 的利弊关系，后者基于 Kubernetes 框架而建立，可为您提供一揽子现成可用的额外优势。

要详细了解 Azure 红帽 OpenShift，请访问产品页面或查阅我们的文档部分。您也可以查看实训研习班，在自己方便时注册观看网络培训课堂。最为重要的是，我们希望您与微软和红帽取得联系，在双方的支持下开展对 Azure 红帽 OpenShift 的评估。

作者和版本

James Read <james@redhat.com>
红帽首席解决方案架构师，
撰写微软相关内容 – 针对 AROv4 进行了更新和修订

Ahmed Sabbour <asabbour@microsoft.com>
微软高级产品营销经理，
撰写 Azure 红帽 OpenShift 相关内容 – 适用于 AROv3 的初始版本

前版本作者

Oren Kashi <okashi@redhat.com>
红帽高级首席技术产品营销经理

衷心感谢 Brooke Jackson、Nermina Miller、Jose Moreno、Ahmed Sabbour、Aditya Datar、Vince Power 和 Alex Patterson 等人拨冗审阅本指南并提供反馈意见。

第 12 章

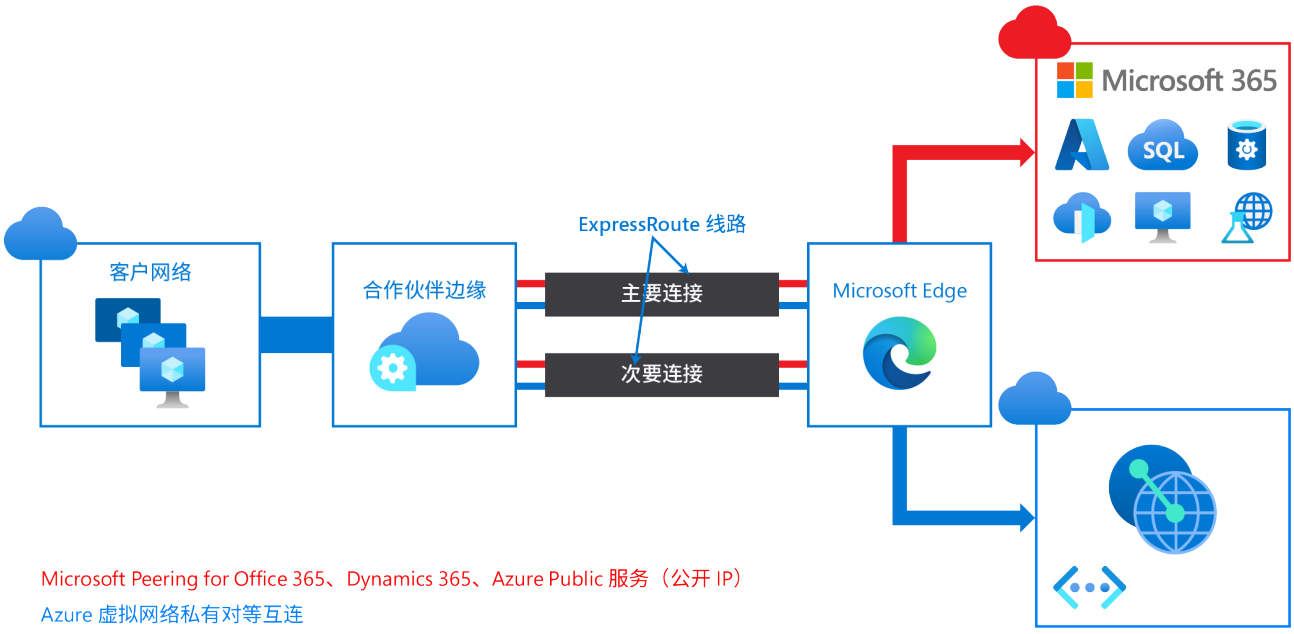
术语表

术语表提供了本指南和 Azure 红帽 OpenShift 生态系统中使用的一些术语的实用快速参考。标题按照字母顺序排序。

Azure ExpressRoute

借助 ExpressRoute，您可以在网络连接提供商的支持下利用私有连接将本地网络延伸到微软云。通过使用 ExpressRoute，您可以建立与微软云服务的连接，例如 Microsoft Azure 和 Microsoft 365。

以下 ExpressRoute 网络示意图摘录自 Microsoft Azure 文档：



若要进一步了解 ExpressRoute，请参阅 Microsoft Azure 文档中的“什么是 ExpressRoute？”页面。

要了解 ExpressRoute 如何与 Azure 配合，请参阅“*第 4 章：置备之前 – 企业架构问题*”。

Azure Landing Zones（Azure 登陆区域）

Azure 登陆区域是在计划利用 Azure 进行大规模部署的组织中广受欢迎的部署模式，其纳入了对规模、安全治理、网络和身份等方面的考量。

Azure 登录区域简介

在撰写本指南时尚处于开发阶段，但有一个社区 GitHub 项目就如何将 Azure 红帽 OpenShift 部署到 Azure 登陆区域架构提供了一些建议：https://github.com/Azure/Enterprise-Scale/tree/main/workloads/ARO

Builds and Image Streams（构建和镜像流）

构建是一个将输入参数转化为结果对象的过程。该过程最常用于将输入参数或源代码转化为可运行的镜像。BuildConfig 对象是对整个构建过程的定义。

Azure 红帽 OpenShift 会利用 Kubernetes 从构建镜像创建 Docker 格式的容器，并将其推送到容器镜像仓库。

构建对象有一些共同的特征，比如：要为构建输入内容、要完成构建过程、要记录构建过程、要发布成功构建的资源，还要发布构建的最终状态。构建利用了资源限制，指定了资源的局限性，比如 CPU 使用量、内存使用量，以及构建或容器集的执行时间。

Azure 红帽 OpenShift 构建系统为构建策略提供了可扩展的支持，这些策略基于构建 API 中指定的可选类型而制定。共有三种主要构建策略可用：

- **Docker 构建** – 使用 Dockerfile，可以上传或从源存储库拉取
- **源至镜像（S2I）构建** – 取用源存储库（例如 Git）并确定如何从知名语言构建文件（例如，适用于 Java 项目的 Maven .pom 文件）构建应用
- **自定义构建**

默认支持 Docker 构建和 S2I 构建。

构建的结果对象取决于创建时所使用的构建器。对于 Docker 和 S2I 构建，结果对象为可运行的镜像。对于自定义构建，结果对象为镜像构建者指定的任何对象。

Container（容器）

Azure 红帽 OpenShift 应用的基本单位是容器。Linux 容器技术是一种轻量级机制，用于隔离运行中的进程，使其仅限于与指定资源交互。

很多应用实例可在单个主机上的容器中运行，进程、文件、网络等相互不可见。虽然容器可用于任意工作负载，但通常每个容器提供一项服务（通常称为“微服务”），如网络服务器或数据库。

Container Images（容器镜像）

Azure 红帽 OpenShift 中的容器基于 Docker 格式的容器镜像而构建。镜像是一个二进制文件，其中包含运行单个容器的所有要求，以及描述其需求和功能的元数据。

您可将其看作是一种打包技术。容器只能访问镜像中定义的资源，除非您在创建容器时赋予其额外的访问权限。通过在多个主机上的多个容器中部署相同镜像，并在相互之间实现负载平衡，Azure 红帽 OpenShift 可为打包成镜像的服务提供冗余和横向扩展。

Container Registry（容器镜像仓库）

Azure 红帽 OpenShift 可提供一个名为 **OpenShift Container Registry（OCR）**的集成式容器镜像仓库，由此增加了按需自动配置新镜像存储库的功能。用户因此获得了一个内置的位置，以便其应用构建推送结果镜像。

每当有新的镜像被推送到 OCR，镜像仓库都会向 Azure 红帽 OpenShift 发送新镜像通知，并传递所有相关信息，比如命名空间、名称和镜像元数据。Azure 红帽 OpenShift 的不同部分会对新镜像做出反应，并创建新的构建和部署。

Azure 红帽 OpenShift 还可以将实现容器镜像仓库 API 的服务器用作镜像来源，包括 Docker Hub 和 Azure 容器镜像仓库。

Deployments and Deployment Configurations（部署和部署配置）

在复制控制器的基础之上，Azure 红帽 OpenShift 通过部署的概念增加了对软件开发和部署生命周期的支持。在最简单的情况下，一次部署只会创建一个新的复制控制器，并让其启动容器集。然而，Azure 红帽 OpenShift 的部署支持从现有镜像部署过渡到新镜像，并且还可定义在创建复制控制器之前或之后运行的 hook。

Azure 红帽 OpenShift 的 DeploymentConfig 对象定义了以下部署详细信息：

1. 复制控制器定义的元素
2. 用于自动创建新部署的触发器
3. 部署之间的过渡策略
4. 生命周期 hook

每次手动或自动触发部署时，都会有一个部署器容器集来管理部署（包括缩减旧的复制控制器，扩展新的复制控制器，以及运行 hook）。部署容器集会在部署完成后无限期保存，以保留其部署日志。

当一个部署被另一个部署所取代时，将会保留先前的复制控制器，以便在需要时轻松回滚。

关于如何创建部署以及如何与部署交互的详细说明，请参阅部署和部署配置。

Jobs（作业）

作业与复制控制器类似，用于就特定原因创建容器集。区别在于，复制控制器是为持续运行的容器集设计的，而作业是为一次性容器集设计的。作业会跟踪所有成功的完成情况，一旦达到指定的完成量，表明相应作业也完成了。

关于如何使用作业的更多详情，请参阅*作业*主题。

Pods and Services（容器集和服务）

Azure 红帽 OpenShift 采用了 Kubernetes 的容器集概念，容器集是一同部署到一个主机上的一个或多个容器，也是可以定义、部署和管理的最小计算单元。

容器集大致相当于一个容器的（物理或虚拟）机器实例。每个容器集都会有自己的内部 IP 地址，因此可以拥有整个端口空间，并且容器集中的各个容器可以共享本地存储和网络。

容器集有一个生命周期；它们会被定义，然后分配到一个节点上运行，并且会持续运行，直至其中的容器退出或因其他原因被移除。根据具体策略和退出代码，退出的容器集可能会被删除，也可能会保留以便访问其容器的日志。

Azure 红帽 OpenShift 将容器集处理成基本不可改变的状态，也就是在容器集运行过程中不可更改其定义。Azure 红帽 OpenShift 通过终止现有容器集并使用修改过的配置和 / 或基本镜像进行重新创建来实施更改。容器集的运行时组件被视为可以消耗，并可从定义的容器镜像重新创建。因此，容器集通常应由更高级别的控制器进行管理，而非由用户直接管理。

Projects and Users（项目和用户）

项目是带有附加注解的 Kubernetes 命名空间，也是管理普通用户资源访问权限的核心工具。通过项目，一个社区的用户可以组织和管理其内容，与其他社区隔离开来。用户必须由管理员赋予对项目的访问权限，如果用户有权创建项目，则可自动获取其自创项目的访问权限。项目可以有单独的名称、displayName 和描述。

名称是必填项，也是项目的唯一标识符，在使用 CLI 工具或 API 时最为多见。名称最长可以有 63 个字符。displayName 是选填项，也是项目在 Web 控制台中显示的名称（默认为 name）。描述是选填项，是对项目更详细的描述，会显示在 Web 控制台中。

开发人员和管理员可使用 CLI 或 Web 控制台与项目交互。

ReplicaSets（副本集）

与复制控制器类似，副本集也可确保容器集中随时都有指定数量的副本在运行。两者区别在于，副本集支持基于集合的选择器要求，而复制控制器仅支持基于等值的选择器要求。

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
    name: frontend-1
    labels:
        tier: frontend
spec:
    replicas: 3
    selector:
        matchLabels:
            tier: frontend
        matchExpressions:
        - {key: tier, operator: In, values: [frontend]}
    template:
        metadata:
            labels:
                tier: frontend
        spec:
            containers:
                - image: openshift/hello-openshift
```

```
            name : helloworld
            ports:
            - containerPort: 8080
              protocol: TCP
        restartPolicy: Always
```

在上方的代码中，您可以发现：

- 通过标签查询一组资源。matchLabels 和 matchExpressions 的结果在逻辑上是相连的。
- 基于等值的选择器，用于指定其标签与选择器相符的资源。
- 基于集合的选择器，用于筛选键值。此设置将会选择 key 等于 tier 且 value 等于 frontend 的所有资源。

ReplicationController（复制控制器）

复制控制器可确保容器集中始终有指定数量的副本在运行。如果有容器集退出或被删除，则复制控制器会实例化更多容器集，直到达到指定数量。同样，如果运行的数量超过了所需数量，则会按需删除部分数量，以达到指定数量。

复制控制器配置由以下部分组成：

- 所需副本的数量（可在运行时调整）。
- 创建复制容器集时要使用的容器集定义。
- 用于识别托管式容器集的选择器。
- 选择器是一组分配给由复制控制器管理的容器集的标签。这些标签包含在由复制控制器实例化的容器集定义当中。复制控制器使用选择器来确定容器集的多少实例已在运行，以便根据需要

进行调整。

复制控制器不会跟踪负载或流量，所以也不会根据这两者的情况进行自动扩展，而是需要通过外部自动扩展器来调整其复制数量。

复制控制器是 Kubernetes 的一个核心对象，名为 ReplicationController。下面提供关于复制控制器的一个示例定义：

```
apiVersion: v1
kind: ReplicationController
metadata:
    name: frontend-1
spec:
    replicas: 1
    selector:
        name: frontend
    template:
        metadata:
            labels:
                name: frontend
        spec:
            containers:
                - image: openshift/hello-openshift
                  name: helloworld
                  ports:
                - containerPort: 8080
                  protocol: TCP
            restartPolicy: Always
```

在上方的代码中，您可以发现：

- 要运行的容器集的副本数量。
- 要运行的容器集的标签选择器。

- 由控制器创建的容器集模板。
- 容器集上的标签应包含标签选择器中的标签。
- 展开所有参数后，名称最长可以有 63 个字符。

Routes and Ingresses（路由和入口）

Azure 红帽 OpenShift 同时支持路由和入口。两者都用于借助 DNS 名称（例如 www.example.com）来公开服务，以便外部客户端进行访问。

路由最初是在红帽 OpenShift 版本 3 中构想的。在发布后，红帽与 Kubernetes 社区携手对此功能进行了标准化，如今将之称为**入口**。

OpenShift 容器平台中的 Kubernetes 入口资源通过共享路由器服务实施入口控制器（Ingress Controller），该服务作为一个容器集在集群中运行。管理入口流量的最常见方式是使用入口控制器。此容器集可以任何其他普通容器集一样扩展和复制。此路由器服务基于 HAProxy 这一开源负载均衡器解决方案。

OpenShift 容器平台路由将入口流量提供给集群中的服务。路由提供了可能不受标准 Kubernetes 入口控制器支持的高级功能，例如 TLS 重新加密、TLS 直通 ，以及蓝绿部署流量拆分。

入口流量通过路由来访问集群中的服务。路由和入口是处理入口流量的主要资源。入口提供与路由

类似的功能，例如接受外部请求并根据路由来分派。不过，使用入口时只能允许特定类型的连接：HTTP/2、HTTPS 和**服务器名称标识（SNI）**，以及利用证书的 TLS 连接。在 OpenShift 容器平台中，生成路由来满足入口资源所指定的条件。

Source-to-Image（源至镜像，简称 S2I）

源至镜像（S2I）是一个用于从源代码中构建可重现容器镜像的工具包和工作流程。S2I 可通过将源代码注入到容器镜像中并让容器准备要执行的源代码来生成可直接运行的镜像。通过自己组装构建器镜像，您可以对构建环境进行版本控制，就像使用容器镜像对运行时环境进行版本控制一样。

对于 Ruby 之类的动态语言，构建时和运行时环境通常是一样的。从描述此环境的构建器镜像（其中已安装 Ruby、Bundler、Rake、Apache、GCC 以及设置和运行 Ruby 所需的其他软件包）开始，S2I 会执行以下操作：

1. 从构建器镜像启动容器，同时将应用源代码注入到已知目录中。
2. 容器过程可以将源代码转化成适当的可运行设置。在本例中具体指通过安装 Bundler 的依赖项，并将源代码移动到 Apache 已预先配置好的目录中来查找 Ruby `config.ru` 文件。
3. 提交新容器，并将镜像入口设置为一个脚本（由构建器镜像提供），该脚本将启动 Apache 以托管 Ruby 应用。

对于像 C、Go 或 Java 这样的编译语言，编译所需的依赖项可能会远超实际运行时工件的大小。为保持运行时镜像的简洁性，S2I 支持多步骤构建过程，其中二进制工件（如可执行文件或 Java WAR 文件）会在第一个构建器镜像中创建，然后提取并注入到第二个运行时镜像，该镜像只需将可执行文件放到正确的位置执行即可。

例如，要为 Tomcat（人气 Java Web 服务器）和 Maven 创建可重现的构建管道，可执行以下步骤：

1. 创建一个包含 OpenJDK 和 Tomcat 的构建器镜像，需要在其中注入 WAR 文件。
2. 创建第二个镜像，并在第一个镜像的基础之上添加 Maven 和其他标准依赖项，需要在其中注

入 Maven 项目。

3. 使用 Java 应用源和 Maven 镜像调用 S2I，以创建所需的应用 WAR。
4. 使用上一步的 WAR 文件和初始 Tomcat 镜像，第二次调用 S2I，以创建运行时镜像。

通过将我们的构建逻辑放到镜像中，并将镜像组合成多个步骤，我们可以使运行时环境接近构建环境（相同的 JDK，相同的 Tomcat JAR），而无需将构建工具部署到生产环境中。

使用 S2I 作为构建策略的目标和优势包括：

- **可重现性**：通过将构建环境打包到容器镜像中并为调用者定义简单接口（注入的源代码），可以对构建环境进行严格的版本控制。可重现的构建是在容器化基础架构中进行安全更新和持续集成的关键要求，而构建器镜像有助于保障可重现性和交换运行时的能力。
- **灵活性**：所有可在 Linux 上运行的现有构建系统均可在容器中运行，每个单独的构建器也可添加到更大的管道当中。此外，可以将处理应用源代码的脚本注入到构建器镜像中，方便作者修改现有镜像以支持源代码处理。
- **速度**：S2I 鼓励构建者在单个镜像层中表示应用，而非在单个 Dockerfile 中构建多个层级。由此节省了创建和部署的时间，便于更好地控制最终镜像的输出。
- **安全性**：使用 Dockerfile 来构建在运行时不会对容器进行很多一般的运维控制，通常会以 root 的用户身份运行，并且能够访问容器网络。S2I 可用于控制构建者镜像的权限和特权，因为构建是在单个容器中启动的。通过与 OpenShift 等平台相结合，S2I 可以让管理员在构建时严格控制开发人员拥有的特权。

Templates（模板）

模板描述了一组可以进行参数化和处理的对象，以生成一个对象列表，供 Azure 红帽 OpenShift 来创建。模板经过处理可用于创建您有权在项目中创建的所有内容，比如服务、构建配置和部署配置。模板还可定义一组标签，以应用于模板中定义的每个对象。

您可使用 CLI 从模板创建对象列表；如果模板已经上传到您的项目或全局模板库，也可使用 Web 控制台。如需获得精选模板集，请参阅 OpenShift 镜像流和模板库。