OpenShift Container Platform 4.15

开始使用

OpenShift Container Platform 入门

Last Updated: 2024-07-23

OpenShift Container Platform 4.15 开始使用

OpenShift Container Platform 入门

法律通告

摘要

本文档提供了帮助您快速开始使用 OpenShift Container Platform 的信息。这包括 Kubernetes 和 OpenShift Container Platform 中常见术语的定义。它还包含 OpenShift Container Platform Web 控制台的步骤，以及使用命令行界面创建和构建应用程序。

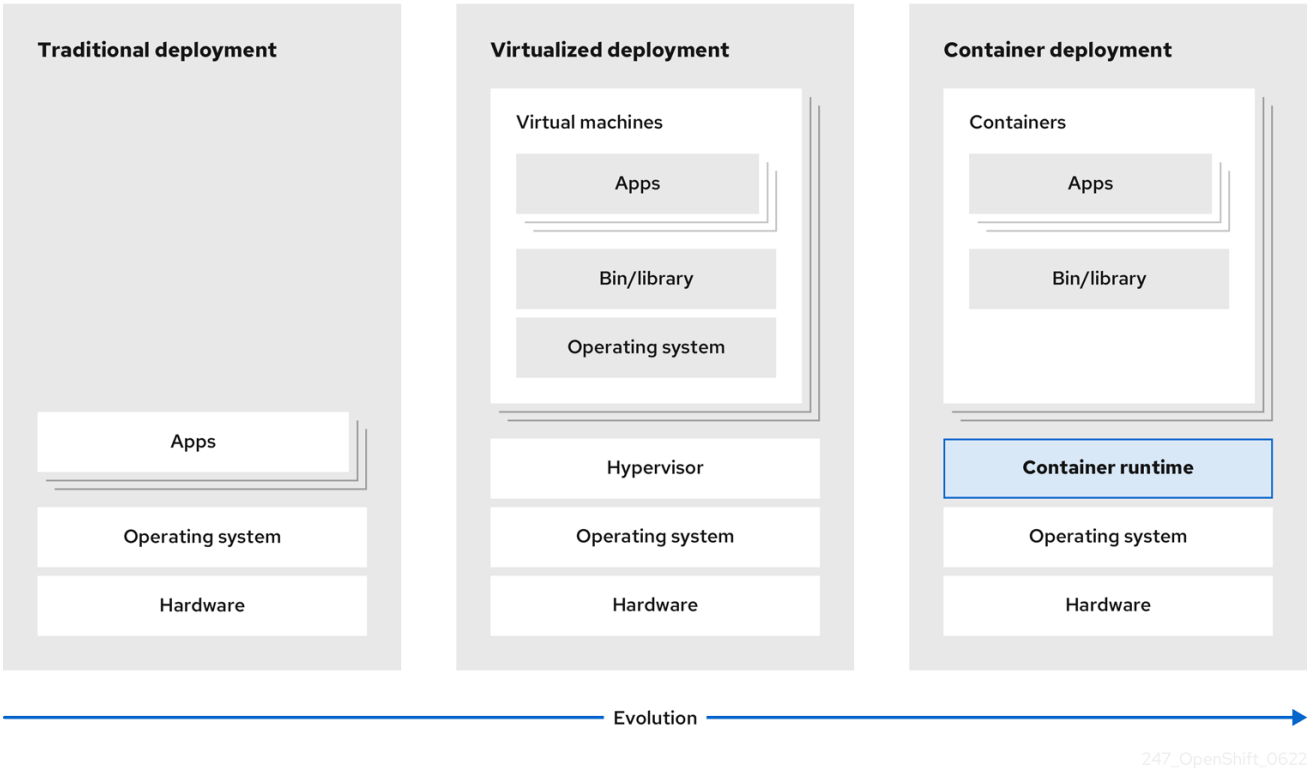
目录

第 1 章 KUBERNETES 概述

Kubernetes 是由 Google 开发的开源容器编排工具。您可以使用 Kubernetes 运行和管理基于容器的工作负载。最常见的 Kubernetes 用例是部署一系列互联微服务，以云原生方式构建应用。您可以创建 Kubernetes 集群，跨越内部部署、公共云、私有云或混合云中的主机。

传统上，应用程序部署在单一操作系统之上。通过虚拟化，您可以将物理主机分成几个虚拟主机。在共享资源中使用虚拟实例并非是实现运行效率和可扩展性的最佳选择。因为虚拟机 (VM) 和物理主机一样会消耗尽可能多的资源，因此为虚拟机提供资源（如 CPU、RAM 和存储）的成本会比较高。另外，您可能会看到，因为使用共享资源，导致虚拟实例中运行的应用程序的性能下降。

图 1.1. 用于类部署的容器技术的演进



要解决这个问题，您可以使用容器化技术，在一个容器化环境中隔离应用程序。与虚拟机类似，容器具有自己的文件系统、vCPU、内存、进程空间、依赖项等。容器与底层基础架构分离，可跨云和操作系统分布移植。与一个功能齐全的操作系统相比，容器是一个更加轻量的系统，它会在操作系统上隔离进程。虚拟机的启动速度较慢，它是物理硬件的抽象概念。虚拟机在一台机器上运行，由一个虚拟机监控程序（hypervisor）处理。

您可以使用 Kubernetes 执行以下操作：

- 共享资源
- 在多个主机间编排容器
- 安装新的硬件配置
- 运行健康检查和自我修复应用程序
- 扩展容器化应用程序

1.1. KUBERNETES 组件

表 1.1. Kubernetes 组件

组件	用途
kube-proxy	在集群的每个节点上运行，并维护 Kubernetes 资源之间的网络流量。
kube-controller-manager	监管集群的状态。
kube-scheduler	将 pod 分配给节点。
etcd	存储集群数据。
kube-apiserver	验证并配置 API 对象的数据。
kubelet	在节点上运行并读取容器清单。确保定义的容器已启动且正在运行。
kubectl	您可以定义如何运行工作负载。使用 **kubectl** 命令与 **kube-apiserver** 进行交互。
节点	节点是 Kubernetes 集群中的物理机器或虚拟机。控制平面（control plane）管理每个节点，并在 Kubernetes 集群中的节点之间调度 pod。
容器运行时	容器运行时在主机操作系统上运行容器。您必须在每个节点上安装容器运行时，以便 pod 能够在该节点上运行。
持久性存储	即便在设备关闭后也存储数据。Kubernetes 使用持久性卷来存储应用程序数据。
container-registry	存储和访问容器镜像。
Pod	pod 是 Kubernetes 中的最小逻辑单元。pod 包含一个或多个在 worker 节点上运行的容器。

1.2. KUBERNETES 资源

自定义资源是 Kubernetes API 的扩展。您可以使用自定义资源自定义 Kubernetes 集群。Operator 是一个软件扩展，它通过自定义资源来管理应用程序及其组件。当您希望在处理集群资源时具有固定的结果，则 Kubernetes 会使用声明性模型。通过使用 Operator，Kubernetes 以声明性方式定义其状态。您可以使用必需命令修改 Kubernetes 集群资源。Operator 充当控制循环，它可以持续将所需的资源状态与资源的实际状态进行比较，并将操作与所需的状态保持一致。

图 1.2. Kubernetes 集群概述

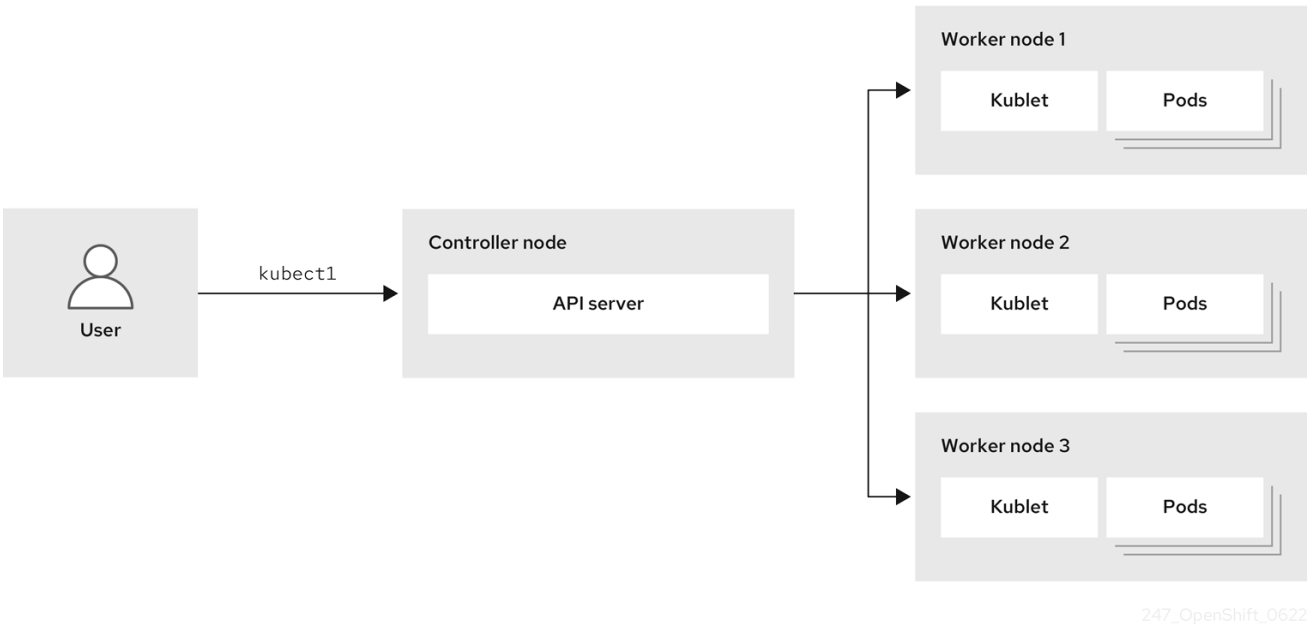
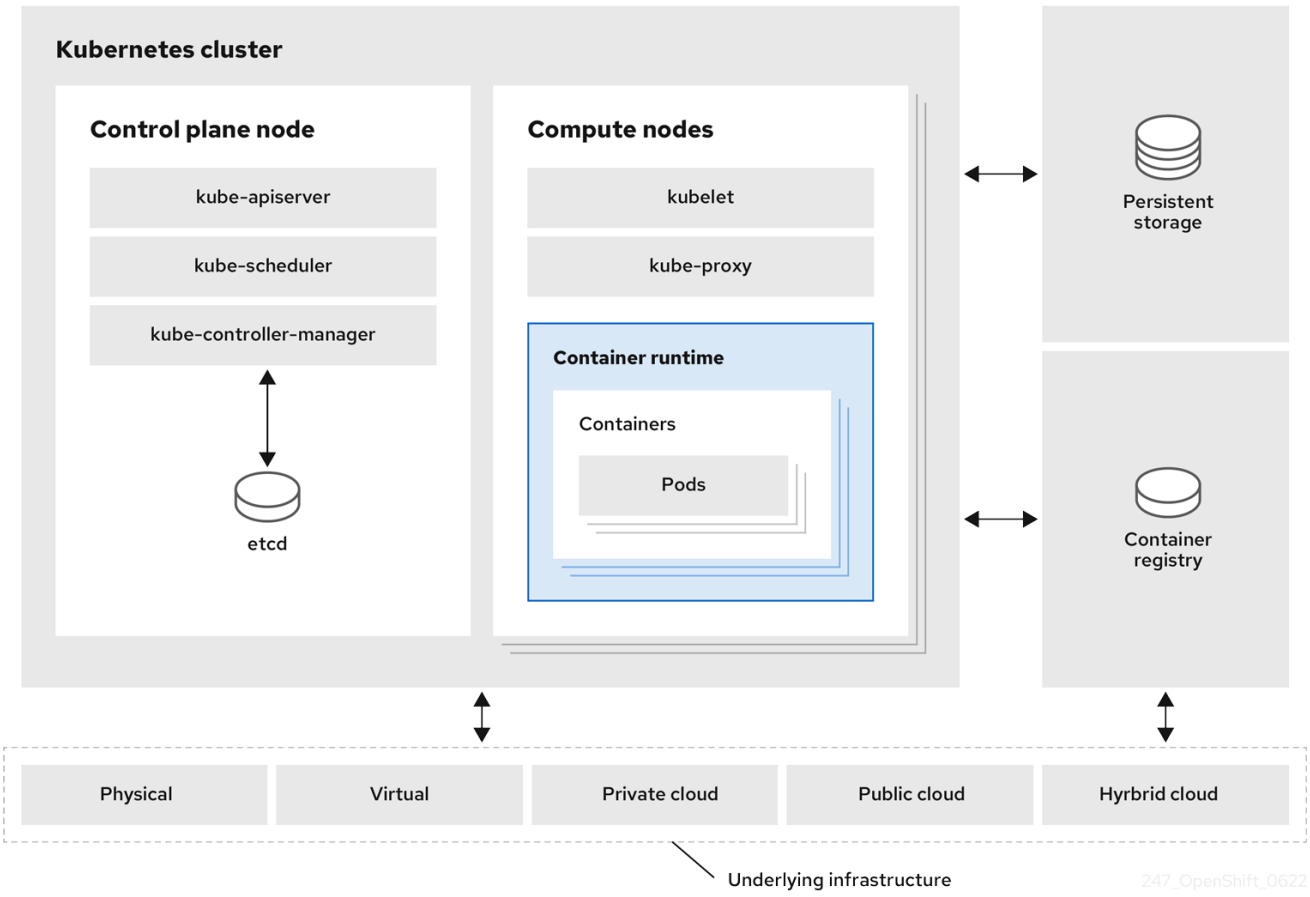


表 1.2. Kubernetes 资源

资源	用途
service	Kubernetes 使用服务在一组 pod 上公开正在运行的应用程序。
ReplicaSet	Kubernetes 使用 **ReplicaSet** 来维护恒定的 pod 号。
Deployment	维护应用程序生命周期的资源对象。

Kubernetes 是 OpenShift Container Platform 的核心组件。您可以使用 OpenShift Container Platform 开发和运行容器化应用程序。OpenShift Container Platform 以 Kubernetes 为基础，为大规模电信、流视频、游戏、银行和其他应用提供引擎技术。您可以使用 OpenShift Container Platform 将容器化应用程序扩展至内部和多云环境中的容器化应用程序。

图 1.3. Kubernetes 构架



集群是一个计算单元，由云环境中的多个节点组成。Kubernetes 集群包含一个 control plane 和 worker 节点。您可以在各种机器和环境中运行 Kubernetes 容器。control plane 节点控制和维护集群的状态。您可以使用 worker 节点运行 Kubernetes 应用程序。您可以使用 Kubernetes 命名空间来区分集群中的集群资源。命名空间范围适用于资源对象，如部署、服务和 pod。您不能将命名空间用于集群范围的资源对象，如存储类、节点和持久性卷。

1.3. KUBERNETES 概念指南

在 OpenShift Container Platform 入门前，请考虑以下 Kubernetes 概念指南：

- 从一个或多个 worker 节点开始，以运行容器工作负载。
- 从一个或多个 control plane 节点管理这些工作负载的部署。
- 将容器嵌套到名为 pod 的部署单元中。使用 pod 可以为容器提供额外的元数据，并可在单个部署实体中对多个容器进行分组。
- 创建特殊种类的资产。例如，服务由一组 pod 及定义了访问方式的策略来表示。此策略可使容器连接到所需的服务，即便容器没有用于服务的特定 IP 地址。复制控制器（replication controller）是另一种特殊资产，用于指示一次需要运行多少个 pod 副本。您可以使用此功能来自动扩展应用程序，以适应其当前的需求。

OpenShift Container Platform 集群的 API 是 100% Kubernetes。在任何 Kubernetes 上运行的容器之间没有变化，并在 OpenShift Container Platform 上运行。没有对应用的更改。OpenShift Container Platform 提供附加值功能，为 Kubernetes 提供企业级的增强。OpenShift Container Platform CLI 工具 (**oc**)与 **kubectl** 兼容。虽然 Kubernetes API 100% 可在 OpenShift Container Platform 中使用，但 **kubectl** 命令行缺少了很多用户友好的功能。OpenShift Container Platform 提供了一组功能和命令行工

具，如 **oc**。虽然 Kubernetes 擅长管理应用程序，但它并未指定或管理平台级要求或部署过程。强大而灵活的平台管理工具和流程是 OpenShift Container Platform 具备的重要优势。您必须将身份验证、网络、安全、监控和日志管理添加到容器化平台中。

第 2 章 OPENSHIFT CONTAINER PLATFORM 概述

OpenShift Container Platform 是一个基于云的 Kubernetes 容器平台。OpenShift Container Platform 的基础基于 Kubernetes，因此共享相同的技术。它旨在允许支持的应用程序和数据中心从少量机器和应用程序扩展到为数百万客户端服务的数千台机器。

OpenShift Container Platform 允许您进行以下操作：

- 为开发人员和 IT 组织提供可用于在安全、可扩展资源上部署应用程序的云应用平台。
- 需要最少的配置和管理开销。
- 将 Kubernetes 平台带到客户数据中心和云。
- 满足安全、隐私、合规性和管理要求。

OpenShift Container Platform 以 Kubernetes 为基础，为大规模电信、流视频、游戏、银行和其他应用提供引擎技术。借助红帽开放技术中的实现，您可以将容器化应用程序从单一云扩展到内部和多云环境。

2.1. OPENSHIFT CONTAINER PLATFORM 常用术语表

此术语表定义了常见的 Kubernetes 和 OpenShift Container Platform 术语。

Kubernetes

Kubernetes 是一个开源容器编配引擎，用于自动化容器化应用程序的部署、扩展和管理。

容器

容器是 worker 节点上以 OCI 兼容容器中运行的应用程序实例和组件。容器是开放容器项目(OCI)兼容镜像的运行时。镜像是二进制应用。worker 节点可以运行多个容器。节点容量与底层资源的内存和 CPU 功能相关，无论它们是云、硬件还是虚拟化。

Pod

pod 是共同部署在同一主机上的一个或多个容器。它包含一组有共享资源（如卷和 IP 地址）的容器。pod 也是定义、部署和管理的最小计算单元。
在 OpenShift Container Platform 中，pod 替代了独立的应用程序容器作为最小的可部署单元。

Pod 是 OpenShift Container Platform 中编配的单元。OpenShift Container Platform 在同一个节点上的一个 pod 中调用并运行所有容器。复杂的应用由多个 pod 组成，每个 pod 都有自己的容器。它们与外部进行交互，也在 OpenShift Container Platform 环境中相互交互。

副本集和复制控制器

Kubernetes 副本集和 OpenShift Container Platform 复制控制器都可用。此组件的作业是确保在所有时间都有指定数量的 pod 副本在运行。如果 pod 退出或被删除，副本集或复制控制器会启动其他 pod 也保证运行的数量。如果运行的 pod 数量超过了指定值，副本集将根据需要删除以匹配指定的副本数。

部署（Deployment）和部署配置（DeploymentConfig）

OpenShift Container Platform 实施 Kubernetes **Deployment** 对象和 OpenShift Container Platform **DeploymentConfig** 对象。用户可以选择其中的一个。
Deployment 对象控制应用程序如何作为 pod 推出部署。它们标识要从 registry 中获取的容器镜像的名称，并作为 pod 部署到节点上。它们设置要部署的 pod 的副本数，创建副本集来管理该进程。所示的标签指示要在哪个节点上部署 pod 的调度程序。在 pod 定义中包括一组标签用于实例化副本集。

Deployment 对象可以根据 **Deployment** 对象版本和各种 rollout 策略来更新部署到 worker 节点上的 pod，以管理应用程序的可用性。OpenShift Container Platform **DeploymentConfig** 对象添加了更改触发器的额外功能，这些触发器可以在容器镜像的新版本或其他更改时自动创建 **Deployment** 对象的

新版本。

服务（Service）

服务定义了一组逻辑的 pod 和访问策略。它为其他应用程序提供永久内部 IP 地址和主机名，供其他用作 pod 的创建和销毁。
服务层将应用程序组件连接在一起。例如，前端 Web 服务通过与其服务通信连接到数据库实例。服务支持跨应用程序组件简单的内部负载平衡。OpenShift Container Platform 会自动将服务信息注入到正在运行的容器中，以简化发现过程。

路由（Route）

路由是通过为服务提供一个外部可访问主机名（如 www.example.com）来公开服务的方法。每个路由都包含路由名称、服务选择器和可选的安全配置。路由器可以使用定义的路由和由服务标识的端点，提供允许外部客户端到达应用程序的名称。尽管易于部署完整的多层应用程序，但从 OpenShift Container Platform 环境以外的任何位置的流量在没有路由层的情况下无法访问应用程序。

Build

构建 (build) 是将输入参数转换为结果对象的过程。此过程最常用于将输入参数或源代码转换为可运行的镜像。**BuildConfig** 对象是整个构建过程的定义。OpenShift Container Platform 通过从构建镜像创建容器并将它们推送到集成的 registry 来利用 Kubernetes。

项目

OpenShift Container Platform 使用项目来允许一组用户或开发人员一起工作，已实现一个隔离单位和协作单位。它定义资源范围，允许项目管理员和协作者管理资源，并使用配额和限制跟踪用户的资源。
项目是带有额外注解的 Kubernetes 命名空间。它是管理常规用户对资源访问的一个中央点。通过项目，一组用户可以在与其他用户隔离的前提下组织和管理其内容。用户需要通过管理员获得项目的访问权限。但是，集群管理员也可以允许开发人员创建自己的项目，在这种情况下，用户会自动获得自己项目的访问权限。

每个项目都有自己的一组对象、策略、约束和服务帐户的集合。

项目也称为命名空间（namespaces）。

Operator

Operator 是一个 Kubernetes 原生应用程序。Operator 的目标是将相关的知识融入到软件来实现各种操作。之前，这些知识存在于管理员的头脑中，以及各种 shell 脚本或自动化软件（如 Ansible）中。它们都在您的 Kubernetes 集群之外，很难集成。但是，Operators 改变了这个情况。
Operator 是为您的应用程序专门构建的。Operator 是在 Kubernetes 集群内运行的、原生集成了 Kubernetes 概念和 API 的一个软件，可以实施和自动化常见的第 1 天操作，如安装和配置，以及第 2 天活动，如扩展和缩减、重新配置、更新、备份、故障转移以及恢复。这称为 Kubernetes 原生应用程序。

使用 Operator 时，应用程序不能被视为原语的集合，如 pod、部署、服务或配置映射。相反，Operator 应被视为单个对象，它公开了对应用程序有意义的选项。

2.2. 了解 OPENSHIFT CONTAINER PLATFORM

OpenShift Container Platform 是一个 Kubernetes 环境，用于管理基于容器的应用程序及其对各种计算平台的依赖，如裸机、虚拟化、内部云等。OpenShift Container Platform 部署、配置和管理容器。OpenShift Container Platform 为其组件提供可用性、稳定性和自定义。

OpenShift Container Platform 利用多个计算资源，称为节点。节点有一个基于 Red Hat Enterprise Linux (RHEL) 的轻量级、安全的操作系统，称为 Red Hat Enterprise Linux CoreOS (RHCOS)。

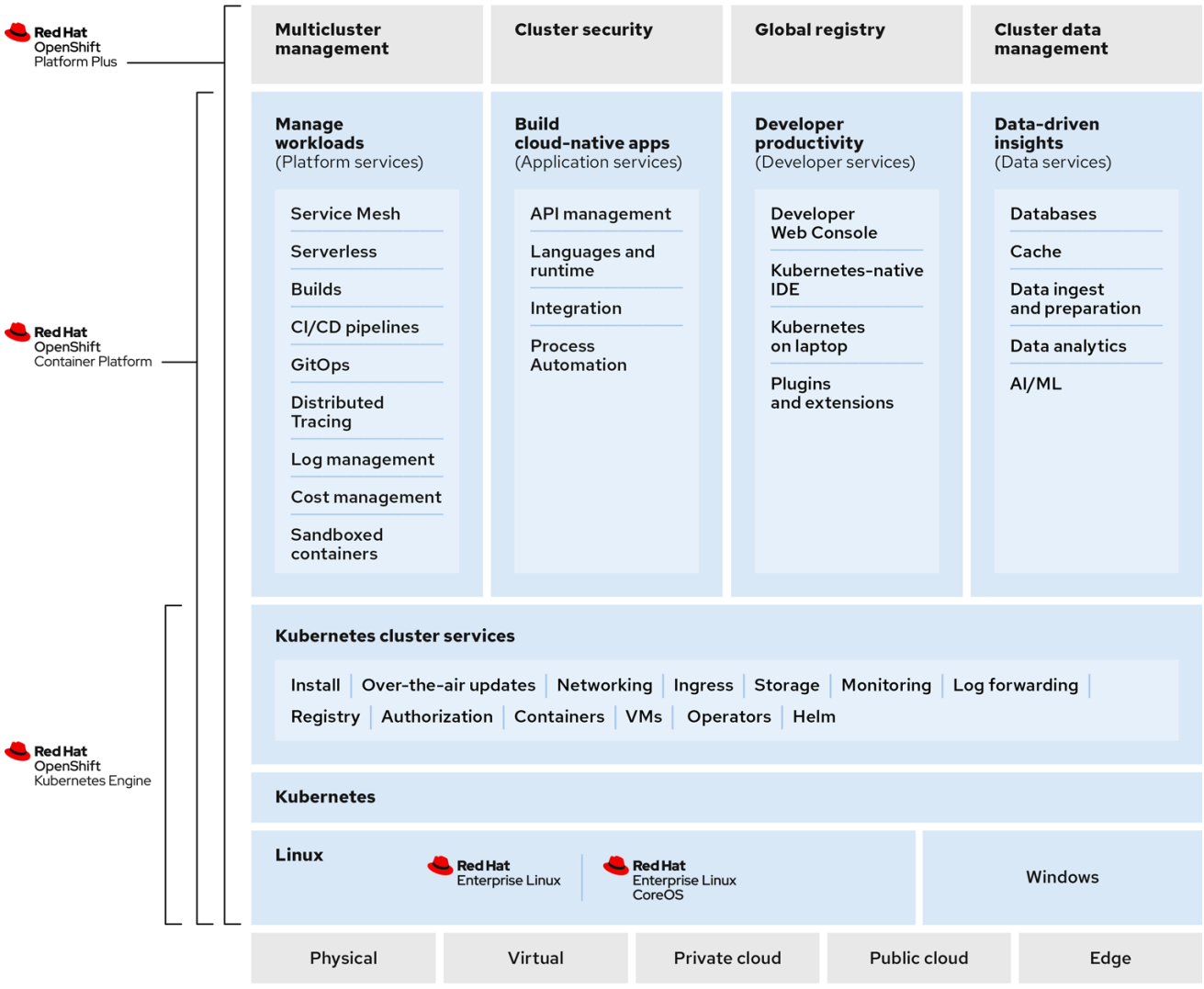
引导并配置节点后，它会获取容器运行时，如 CRI-O 或 Docker，用于管理和运行调度到其中的容器工作负载的镜像。Kubernetes 代理或 kubelet 会在节点上调度容器工作负载。kubelet 负责将节点注册到集群并接收容器工作负载的详情。

OpenShift Container Platform 配置并管理集群的网络、负载均衡和路由。OpenShift Container Platform 添加了集群服务来监控集群健康和性能、日志记录和管理升级。

容器镜像 registry 和 OperatorHub 提供红帽认证的产品和社区构建的软件，用于在集群中提供各种应用程序服务。这些应用程序和服务管理集群中部署的应用程序、数据库、前端和用户界面、应用程序运行时和业务自动化，以及用于开发和测试容器应用的开发人员服务。

您可以通过配置从预构建镜像运行的容器部署或通过称为 Operator 的资源来手动管理集群中的应用程序。您可以通过预先构建的镜像和源代码构建自定义镜像，并将这些自定义镜像存储在本地内部、私有或公共 registry 中。

多集群管理层可以使用一个控制台管理多个集群，包括它们的部署、配置、合规性和工作负载分布。



2.3. 安装 OPENSHIFT CONTAINER PLATFORM

OpenShift Container Platform 安装程序为您提供了灵活性。您可以使用安装程序将集群部署到由安装程序置备并由集群维护的基础架构中，也可以将集群部署到您自己准备和维护的基础架构中。

有关安装过程、支持的平台以及选择安装和配置集群方法的更多信息，请参阅：

- OpenShift Container Platform 安装概述
- 安装过程
- OpenShift Container Platform 集群支持的平台
- 选择集群安装类型

2.3.1. OpenShift Local 概述

OpenShift Local 支持快速应用程序开发，以开始构建 OpenShift Container Platform 集群。OpenShift Local 设计为在本地计算机上运行，以简化设置和测试，并使用开发基于容器的应用所需的所有工具在本地模拟云环境。

无论您使用什么编程语言，OpenShift Local 都可以托管您的应用程序，并将最小预配置的 Red Hat OpenShift Container Platform 集群引入本地 PC，而无需基于服务器的基础架构。

在托管环境中，OpenShift Local 可以创建微服务，将它们转换为镜像，并在运行 Linux、macOS 或 Windows 10 或更高版本的笔记本电脑或桌面上直接运行它们。

如需有关 OpenShift Local 的更多信息，请参阅 Red Hat OpenShift Local Overview 。

2.4. 后续步骤

2.4.1. 对于开发人员

使用 OpenShift Container Platform 开发和部署容器化应用。OpenShift Container Platform 是一个用于开发和部署容器化应用程序的平台。OpenShift Container Platform 文档可帮助您：

- **了解 OpenShift Container Platform 开发** ：了解不同类型的容器化应用，从简单的容器到高级 Kubernetes 部署和 Operator。
- **使用项目** ：通过 OpenShift Container Platform Web 控制台或 OpenShift CLI(**oc**)创建项目以组织和共享您开发的软件。
- **使用应用程序** ：

使用 OpenShift Container Platform Web 控制台中的 **Developer** 视角来创建和部署应用程序。

使用 **Topology** 视图 查看应用程序、监控状态、连接和组组件，以及修改您的代码库。

- **使用开发人员 CLI 工具(odo)** ： **odo** CLI 工具允许开发人员创建单个或多组件应用程序，并自动执行部署、构建和服务路由配置。它提取了复杂的 Kubernetes 和 OpenShift Container Platform 概念，允许您专注于开发应用程序。
- **创建 CI/CD 管道** ：管道 (Pipeline) 是无服务器、云原生、持续集成和持续部署（CI/CD）的系统，它在隔离的容器中运行。它们使用标准的 Tekton 自定义资源来实现部署自动化，并为处理基于微服务的架构的非中心化团队设计。
- **部署 Helm chart**Helm 3 是一个软件包管理器，可帮助开发人员在 Kubernetes 中定义、安装和更新应用程序软件包。Helm Chart 是一个打包格式，用于描述可以使用 Helm CLI 部署的应用程序。

- **了解镜像构建** ：从不同的构建策略（Docker、S2I、自定义和管道）中选择可以包括不同类型的源资料（Git 存储库、本地二进制输入和外部工件）。然后，请参阅从基本构建到高级构建的构建类型示例。

- **创建容器镜像** ：容器镜像是 OpenShift Container Platform（和 Kubernetes）应用程序中最基本的构建块。通过定义镜像流，在继续开发镜像时，可让您在一个位置保存镜像的多个版本。S2I 容器允许您将源代码插入到基本容器中，该容器被设置为运行特定类型的代码，如 Ruby、Node.js 或 Python。

- **创建部署** ：使用 **Deployment** 和 **DeploymentConfig** 对象对应用程序进行精细管理。使用 **Workloads** 页或 OpenShift CLI (**oc**) 管理部署。了解滚动、重新创建和自定义部署策略。

- **创建模板**：使用现有模板或创建自己的模板来描述应用的构建或部署方式。模板可以将镜像与描述、参数、副本、公开端口和其他定义如何运行或构建的内容相结合。

- **了解 Operator**：Operator 是为 OpenShift Container Platform 4.15 创建集群应用程序的首选方法。了解 Operator Framework 以及如何使用已安装的 Operator 部署到项目中。

- **Develop Operators**：Operator 是为 OpenShift Container Platform 4.15 创建集群应用程序的首选方法。了解构建、测试和部署 Operator 的工作流。然后，基于 Ansible 或 Helm 创建自己的 Operator，或使用 Operator SDK 配置内置 Prometheus 监控。

- **REST API 参考** ：了解 OpenShift Container Platform 应用程序编程接口端点。

2.4.2. 对于管理员

- **了解 OpenShift Container Platform 管理** ：了解 OpenShift Container Platform 4.15 control plane 的组件。请参阅 OpenShift Container Platform control plane 和 worker 节点如何通过 Machine API 和 Operator 进行管理和更新。

- **管理用户和组** ：添加具有不同级别权限的用户和组，以使用或修改集群。

- **管理身份验证** ：了解用户、组和 API 身份验证在 OpenShift Container Platform 中的工作方式。OpenShift Container Platform 支持多个身份提供程序。

- **管理网络** ：OpenShift Container Platform 中的集群网络由 Cluster Network Operator (CNO)管理。CNO 使用 kube-proxy 中的 iptables 规则来指示在这些节点上运行的节点和 pod 间的流量。Multus Container Network Interface 添加了将多网络接口附加到 pod 的功能。使用网络策略功能，您可以隔离 pod 或允许所选流量。

- **管理存储** ：OpenShift Container Platform 允许集群管理员配置持久性存储。

- **管理 Operator** ：Red Hat, ISV, 和社区 Operators 列表，集群管理员可对其进行审核并 在集群上进行安装。安装后，您可以运行、升级、备份或管理集群中的 Operator。

- **使用自定义资源定义(CRD)修改集群** ：通过 Operator 实施的集群功能可使用 CRD 修改。了解如何创建 CRD 并从 CRD 管理资源。

- **设置资源配额** ：从 CPU、内存和其他系统资源中选择来设置配额。

- **修剪和回收资源** ：通过修剪不需要的 Operator、组、部署、构建、镜像、registry 和 cron 作业来回收空间。

- **扩展和调优集群** ：设置集群限制、调整节点、扩展集群监控和优化您的环境的网络、存储和路由。

- **在断开连接的环境中使用 OpenShift Update Service**：了解如何安装和管理本地 OpenShift Update Service，以在断开连接的环境中推荐 OpenShift Container Platform 更新。

- **监控集群**：了解如何配置监控堆栈。配置监控后，使用 Web 控制台访问监控仪表板。除了基础架构指标外，您还可以提取和查看您自己的服务的指标。

- **远程健康监控**：OpenShift Container Platform 会收集有关集群的匿名汇总信息。通过使用 Telemetry 和 Insights Operator，红帽会接收这些数据，用于改进 OpenShift Container Platform。您可以查看远程健康监控收集的数据。

第 3 章 使用 WEB 控制台创建并构建应用程序

3.1. 开始前

- 查看访问 Web 控制台。
- 您需要可以访问运行的 OpenShift Container Platform 实例。如果没有访问权限，请联络您的集群管理员。

3.2. 登录到 WEB 控制台

您可以登录到 OpenShift Container Platform Web 控制台来访问和管理集群。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。

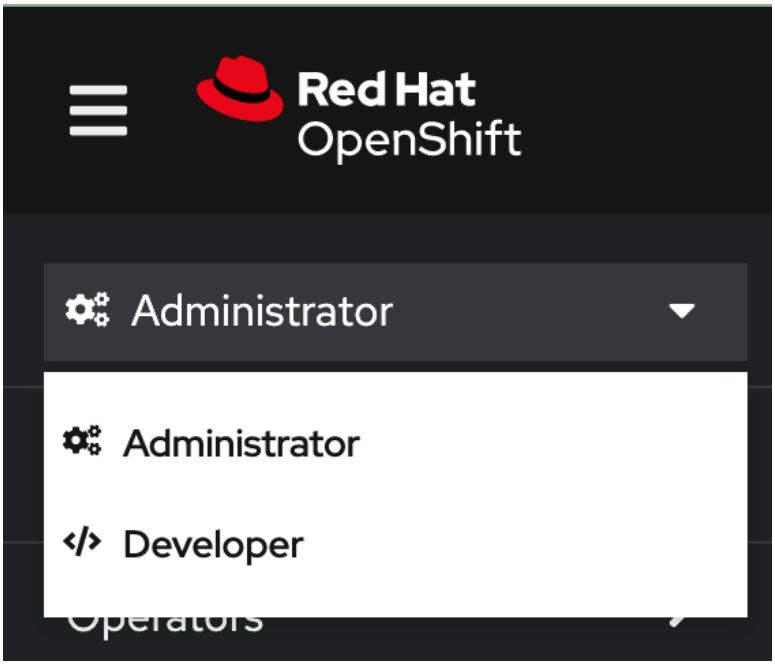
流程

- 使用您的登录凭证登录到 OpenShift Container Platform web 控制台。

您将被重定向到 **Projects** 页面。对于非管理员用户，默认视图是 **Developer** 视角。对于集群管理员，默认的视图是 **Administrator** 视角。如果没有 **cluster-admin** 权限，则无法在 web 控制台中看到 **Administrator** 视角。

Web 控制台提供两个视角： **Administrator** 视角和 **Developer** 视角。**Developer** 视角提供特定于开发人员用例的工作流。

图 3.1. 视角切换器



使用视角切换功能把它切换到 **Developer** 视角。带有创建应用程序选项的 **Topology** 视图会被显示。

3.3. 创建新项目

通过项目，一个社区用户可以在隔离时组织和管理其内容。项目是 OpenShift Container Platform 对 Kubernetes 命名空间的扩展。项目具有额外的功能，使用户自助配置。

用户需要通过管理员获得项目的访问权限。集群管理员可以允许开发人员创建自己的项目。在大多数情况下，用户会自动获得其自己的项目的访问权限。

每个项目都有自己的一组对象、策略、约束和服务帐户的集合。

先决条件

- 已登陆到 OpenShift Container Platform Web 控制台。
- 处于 **Developer** 视角。
- 在项目中拥有适当的角色和权限，可在 OpenShift Container Platform 中创建应用程序和其他工作负载。

流程

1. 在 **+Add** 视图中，选择 **Project → Create Project**。
2. 在 **Name** 字段中输入 **user-getting-started**。
3. 可选：在 **Display name** 字段中输入 **Getting Started with OpenShift**。

注意

显示名称和**描述**字段是可选的。

4. 点 **Create**。

您已在 OpenShift Container Platform 上创建第一个项目。

其他资源

- 默认集群角色
- 使用 Web 控制台查看项目
- 使用 Developer 视角为您的项目提供访问权限
- 使用 web 控制台删除项目

3.4. 授予查看权限

OpenShift Container Platform 会在每个项目中自动创建一些特殊服务帐户。默认服务帐户负责运行 pod。OpenShift Container Platform 使用并将此服务帐户注入到启动的每个 pod 中。

以下流程为默认 **ServiceAccount** 对象创建一个 **RoleBinding** 对象。服务帐户与 OpenShift Container Platform API 通信，以了解项目中的 pod、服务和资源。

先决条件

- 已登陆到 OpenShift Container Platform Web 控制台。

- 您已部署了一个镜像。
- 您处于 **Administrator** 视角。

流程

1. 进入到 **User Management**，然后点 **RoleBindings**。
2. 点 **Create binding**。
3. 选择 **Namespace 角色绑定(RoleBinding)**。
4. 在 **Name** 字段中，输入 **sa-user-account**。
5. 在 **Namespace** 字段中，搜索并选择 **user-getting-started**。
6. 在 **Role name** 字段中，搜索 **view** 并选择 **view**。
7. 在 **Subject** 字段中，选择 **ServiceAccount**。
8. 在 **Subject namespace** 字段中，搜索并选择 **user-getting-started**。
9. 在 **Subject name** 字段中，输入 **default**。
10. 点 **Create**。

其他资源

- 了解身份验证
- RBAC 概述

3.5. 部署您的第一个镜像

在 OpenShift Container Platform 中部署应用程序的最简单方法是运行现有容器镜像。以下流程部署一个应用的前端组件，名为 **national-parks-app**。Web 应用显示一个交互式的地图。该地图显示全球主要国家公园的位置。

先决条件

- 已登陆到 OpenShift Container Platform Web 控制台。
- 处于 **Developer** 视角。
- 在项目中拥有适当的角色和权限，可在 OpenShift Container Platform 中创建应用程序和其他工作负载。

流程

1. 从 **Developer** 视角中的 **+Add** 视图，点 **Container images** 来打开一个对话框。
2. 在 **Image Name** 字段中，输入以下内容： **quay.io/openshiftroadshow/parksmap:latest**
3. 确保具有以下内容的当前值：
    a. 应用程序：**national-parks-app**

b. 名称：**parksmap**

4. 选择 **Deployment** 作为 **资源**。

5. 选择 **Create route to the application**。

6. 在 **Advanced Options** 部分中，点 **Labels** 并添加标签以更好地识别此部署。标签可帮助识别和过滤 web 控制台和命令行中的组件。添加以下标签：

   - **app=national-parks-app**
   - **component=parksmap**
   - **role=frontend**

7. 点 **Create**。

您会被重定向到 **Topology** 页面，您可以在其中查看 **national-parks-app** 应用程序中的 **parksmap** 部署。

其他资源

- 使用 Developer 视角创建应用程序
- 使用 Web 控制台查看项目
- 查看应用程序拓扑
- 使用 web 控制台删除项目

3.5.1. 检查 pod

OpenShift Container Platform 使用 Kubernetes 的 pod 概念，它是共同部署在同一主机上的一个或多个容器，也是可被定义、部署和管理的最小计算单元。对容器而言，Pod 大致相当于一个机器实例（物理或虚拟）。

通过 **Overview** 面板，您可以访问 **parksmap** 部署的许多功能。**Details** 和 **Resources** 选项卡允许您缩放应用程序 pod，检查构建状态、服务和路由。

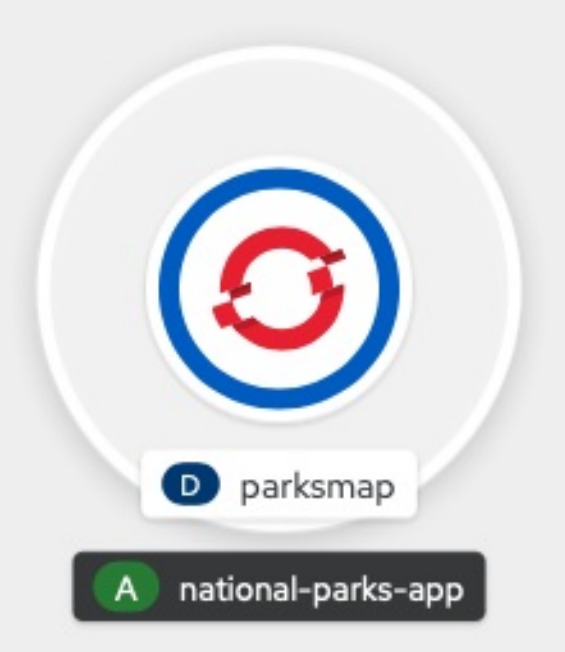
先决条件

- 已登陆到 OpenShift Container Platform Web 控制台。
- 处于 **Developer** 视角。
- 您已部署了一个镜像。

流程

- 点 **Topology** 视图中的 **D parksmap** 打开 **Overview** 面板。

图 3.2. Parksmap 部署



Overview 面板包括 Details, Resources, 和 Observe 标签页。Details 选项卡可能会默认显示。

表 3.1. 概述面板选项卡定义

标签页	定义
详情	允许您扩展应用程序并查看应用程序的状态，如标签、注解和应用程序状态。
Resources	显示与部署关联的资源。
	Pod 是 OpenShift Container Platform 应用程序的基本单元。您可以查看已使用多少个 pod，以及它们的状态，您可以查看日志。
	为 pod 和分配的端口创建的服务列在 Services 标题下。
	通过路由可以允许从外部访问 pod，并使用一个 URL 来访问它们。
观察	查看各种事件和指标信息，因为它与您的 pod 相关。

其他资源

- 与应用程序和组件交互
- 扩展应用程序 Pod 以及检查构建和路由
- 用于 Topology 视图的标签和注解

3.5.2. 扩展应用程序

在 Kubernetes 中，**Deployment** 对象定义了应用的部署方式。在大多数情况下，用户会一起使用 **Pod**, **Service**, **ReplicaSets**, 和 **Deployment** 资源。在大多数情况下，OpenShift Container Platform 会为您创建资源。

当您部署 **national-parks-app** 镜像时，会创建一个部署资源。在本例中，只部署了一个 **Pod**。

以下流程将 **national-parks-image** 扩展为使用两个实例。

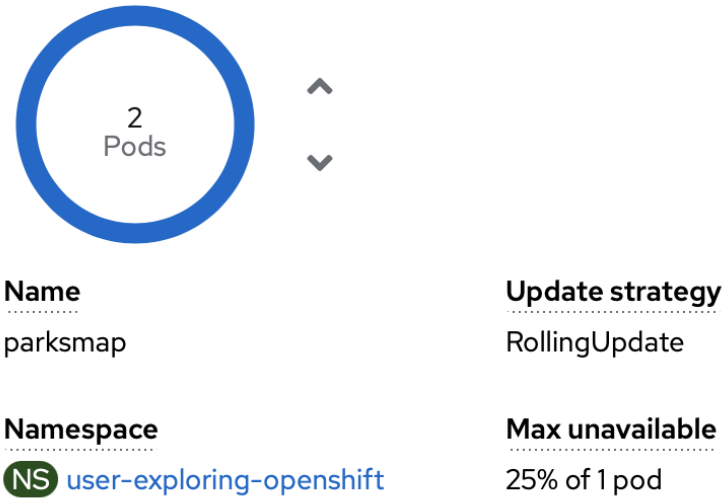
先决条件

- 已登陆到 OpenShift Container Platform Web 控制台。
- 处于 **Developer** 视角。
- 您已部署了一个镜像。

流程

1. 在 **Topology** 视图中，单击 **national-parks-app** 应用。
2. 点 **Details** 标签页。
3. 使用向上箭头将容器集扩展到两个实例。

   图 3.3. 扩展应用程序



注意

应用程序的扩展会快速发生，因为 OpenShift Container Platform 正在启动现有镜像的一个新实例。

4. 使用向下箭头键将 pod 缩减为一个实例。

其他资源

- 扩展集群的建议实践
- 了解 pod 横向自动扩展

- 关于 Vertical Pod Autoscaler Operator

3.6. 部署 PYTHON 应用程序

以下流程为 **parksmap** 应用程序部署后端服务。Python 应用程序针对 MongoDB 数据库执行 2D geo-spatial 查询，以定位和返回世界上的所有国家公园的信息。

部署的后端服务为 **nationalparks**。

先决条件

- 已登陆到 OpenShift Container Platform Web 控制台。
- 处于 **Developer** 视角。
- 您已部署了一个镜像。

流程

1. 从 **Developer** 视角中的 **+Add** 视图，点 **Import from Git** 来打开一个对话框。
2. 在 Git Repo URL 字段中输入以下 URL：**https://github.com/openshift-roadshow/nationalparks-py.git**
   构建器(builder)镜像会被自动探测。

注意

如果检测到的构建器镜像是 Dockerfile，请选择 **Edit Import Strategy**。选择 **Builder Image**，然后点 **Python**。

3. 滚动到 **General** 部分。
4. 确保具有以下内容的当前值：
   a. 应用程序：**national-parks-app**
   b. 名称：**nationalparks**
5. 选择 **Deployment** 作为 **资源**。
6. 选择 **Create route to the application**。
7. 在 **Advanced Options** 部分中，点 **Labels** 并添加标签以更好地识别此部署。标签可帮助识别和过滤 web 控制台和命令行中的组件。添加以下标签：
   a. **app=national-parks-app**
   b. **component=nationalparks**
   c. **role=backend**
   d. **type=parksmap-backend**
8. 点 **Create**。
9. 在 **Topology** 视图中，选择 **nationalparks** 应用。

注意

单击 **Resources** 选项卡。在 **Builds** 部分中，您可以看到构建正在运行。

其他资源

- 在应用程序中添加服务
- 从 Git 导入代码库来创建应用程序
- 查看应用程序拓扑
- 使用 Developer 视角为您的项目提供访问权限
- 使用 web 控制台删除项目

3.7. 连接到数据库

部署并连接一个 MongoDB 数据库，其中的 National **-parks-app** 应用存储位置信息。将 **national-parks-app** 应用程序标记为地图可视化工具的后端后，**parksmap** 部署会使用 OpenShift Container Platform 发现机制来自动显示地图。

先决条件

- 已登陆到 OpenShift Container Platform Web 控制台。
- 处于 **Developer** 视角。
- 您已部署了一个镜像。

流程

1. 从 **Developer** 视角中的 **+Add** 视图，点 **Container images** 来打开一个对话框。
2. 在 **Image Name** 字段中，输入 **quay.io/centos7/mongodb-36-centos7**。
3. 在 **Runtime 图标** 字段中，搜索 **mongodb**。
4. 向下滚动到 **General** 部分。
5. 确保具有以下内容的当前值：
   a. 应用程序：**national-parks-app**
   b. 名称： **mongodb-nationalparks**
6. 选择 **Deployment** 作为 **资源**。
7. 取消选择 **Create route to the application**旁边的复选框。
8. 在 **Advanced Options** 部分中，点 **Deployment** 添加环境变量，以添加以下环境变量：

   表 3.2. 环境变量名称和值

Name	value
MONGODB_USER	**mongodb**
MONGODB_PASSWORD	**mongodb**
MONGODB_DATABASE	**mongodb**
MONGODB_ADMIN_PASSWORD	**mongodb**

9. 点 **Create**。

其他资源

- 在应用程序中添加服务
- 使用 Web 控制台查看项目
- 查看应用程序拓扑
- 使用 Developer 视角为您的项目提供访问权限
- 使用 web 控制台删除项目

3.7.1. 创建 secret

Secret 对象提供了一种机制来保存敏感信息，如密码、OpenShift Container Platform 客户端配置文件和私有源存储库凭证等。secret 将敏感内容与 Pod 分离。您可以使用卷插件将 secret 信息挂载到容器中，系统也可以使用 secret 代表 Pod 执行操作。以下流程添加了 secret **nationalparks-mongodb-parameters**，并将它挂载到 **nationalparks** 工作负载中。

先决条件

- 已登陆到 OpenShift Container Platform Web 控制台。
- 处于 **Developer** 视角。
- 您已部署了一个镜像。

流程

1. 从 **Developer** 视角中，导航到左侧导航上的 **Secret**，再单击 **Secrets**。
2. 点 **Create → Key/value secret**。
   a. 在 **Secret name** 字段中，输入 **nationalparks-mongodb-parameters**。
   b. 输入 **Key** 和 **Value** 的以下值：

   表 3.3. Secret 键和值

键	value
MONGODB_USER	**mongodb**
DATABASE_SERVICE_NAME	**mongodb-nationalparks**
MONGODB_PASSWORD	**mongodb**
MONGODB_DATABASE	**mongodb**
MONGODB_ADMIN_PASSWORD	**mongodb**

c. 点 **Create**。

3. 点 **Add Secret to workload**。

a. 从下拉菜单中，选择 **nationalparks** 作为要添加的工作负载。

b. 点击 **Save**。

这个配置的更改会触发一个新的 **nationalparks** 部署推出部署，并正确注入环境变量。

其他资源

- 了解 secret

3.7.2. 载入数据并显示国家公园地图

您已部署了 **parksmap** 和 **Nationalparks** 应用程序，然后部署 **mongodb-nationalparks** 数据库。但是，没有将数据 *加载到* 数据库中。在载入数据前，将正确的标签添加到 **mongodb-nationalparks** 和 **nationalparks** 部署中。

先决条件

- 已登陆到 OpenShift Container Platform Web 控制台。
- 处于 **Developer** 视角。
- 您已部署了一个镜像。

流程

1. 从 **Topology** 视图中，进入到 **nationalparks** 部署，再单击 **Resources** 并检索您的路由信息。
2. 将 URL 复制并粘贴到您的网页浏览器中，并在 URL 的末尾添加以下内容：

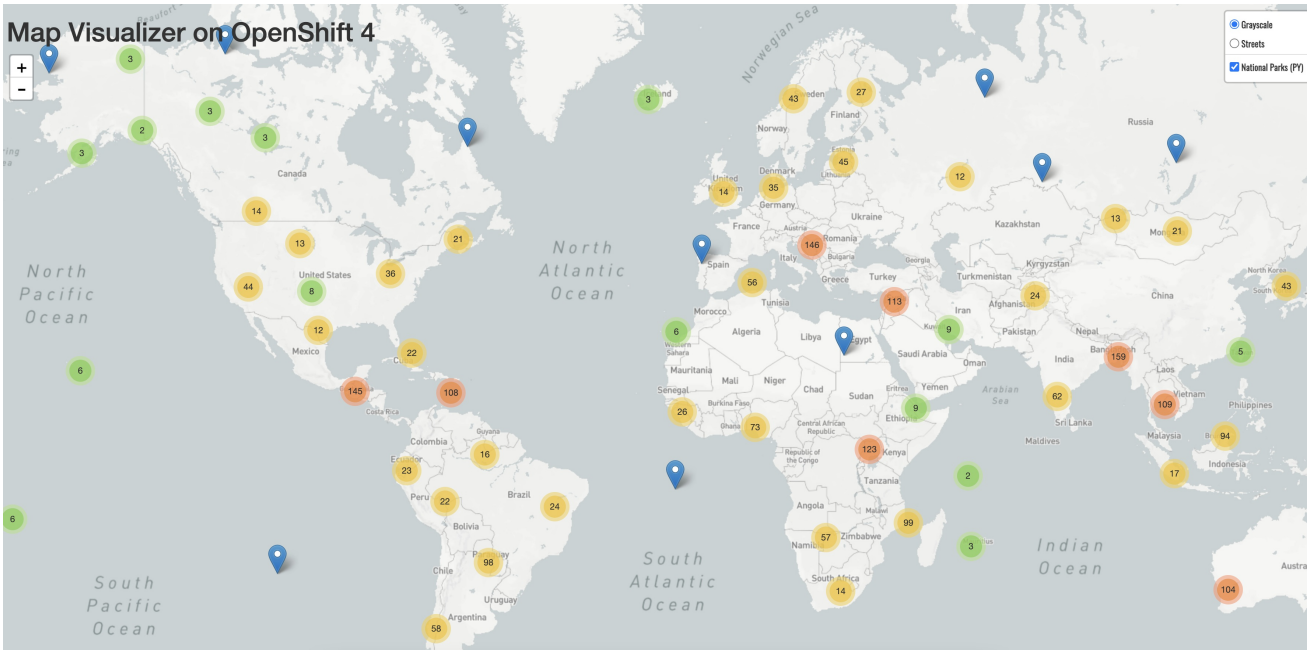
```
/ws/data/load
```

输出示例

```
Items inserted in database: 2893
```

3. 在 **Topology** 视图中，导航到 **parksmap** 部署，再单击 **Resources** 并检索您的路由信息。
4. 将 URL 复制并粘贴到您的 Web 浏览器中，以在世界地图中查看您的国家公园。

图 3.4. 跨世界的国家公园



其他资源

- 使用 Developer 视角为您的项目提供访问权限
- 用于 Topology 视图的标签和注解

第 4 章 使用 CLI 创建并构建应用程序

4.1. 开始前

- 查阅关于 OpenShift CLI。
- 您需要可以访问运行的 OpenShift Container Platform 实例。如果没有访问权限，请联络您的集群管理员。
- 您已下载并安装 OpenShift CLI(**oc**)。

4.2. 登录到 CLI

您可以登录到 OpenShift CLI（**oc**）以访问和管理集群。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。
- 已安装 OpenShift CLI（**oc**）。

流程

- 使用您的用户名和密码、OAuth 令牌或使用 Web 浏览器通过 CLI 登录 OpenShift Container Platform：
  - 使用用户名和密码登陆：

```
$ oc login -u=<username> -p=<password> --server=<your-openshift-server> --insecure-skip-tls-verify
```

  - 使用 OAuth 令牌：

```
$ oc login <https://api.your-openshift-server.com> --token=<tokenID>
```

  - 使用 Web 浏览器：

```
$ oc login <cluster_url> --web
```

您现在可以创建项目或执行其他命令来管理集群。

其他资源

- oc login
- oc logout

4.3. 创建新项目

通过项目，一个社区用户可以在隔离时组织和管理其内容。项目是 OpenShift Container Platform 对 Kubernetes 命名空间的扩展。项目具有额外的功能，使用户自助配置。

用户需要通过管理员获得项目的访问权限。集群管理员可以允许开发人员创建自己的项目。在大多数情况下，用户会自动获得其自己的项目的访问权限。

每个项目都有自己的一组对象、策略、约束和服务帐户的集合。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。
- 已安装 OpenShift CLI（**oc**）。

流程

- 要创建新项目，请输入以下命令：

```
$ oc new-project user-getting-started --display-name="Getting Started with OpenShift"
```

输出示例

```
Now using project "user-getting-started" on server "https://openshift.example.com:6443".
```

其他资源

- oc new-project

4.4. 授予查看权限

OpenShift Container Platform 会在每个项目中自动创建一些特殊服务帐户。默认服务帐户负责运行 pod。OpenShift Container Platform 使用并将此服务帐户注入到启动的每个 pod 中。

以下流程为默认 **ServiceAccount** 对象创建一个 **RoleBinding** 对象。服务帐户与 OpenShift Container Platform API 通信，以了解项目中的 pod、服务和资源。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。
- 已安装 OpenShift CLI（**oc**）。
- 您已部署了一个镜像。
- 您必须具有 **cluster-admin** 或 **project-admin** 权限。

流程

- 要将 view 角色添加到 **user-get-started** 项目中的 default 服务帐户，请输入以下命令：

```
$ oc adm policy add-role-to-user view -z default -n user-getting-started
```

其他资源

- 了解身份验证

- RBAC 概述
- oc policy add-role-to-user

4.5. 部署您的第一个镜像

在 OpenShift Container Platform 中部署应用程序的最简单方法是运行现有容器镜像。以下流程部署一个应用的前端组件，名为 **national-parks-app**。Web 应用显示一个交互式的地图。该地图显示全球主要国家公园的位置。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。
- 安装 OpenShift CLI (**oc**) 。

流程

- 要部署应用程序，请输入以下命令：

```
$ oc new-app quay.io/openshiftroadshow/parksmap:latest --name=parksmap -l
'app=national-parks-app,component=parksmap,role=frontend,app.kubernetes.io/part-
of=national-parks-app'
```

输出示例

```
--> Found container image 0c2f55f (12 months old) from quay.io for
"quay.io/openshiftroadshow/parksmap:latest"

    * An image stream tag will be created as "parksmap:latest" that will track this image

--> Creating resources with label app=national-parks-app,app.kubernetes.io/part-of=national-
parks-app,component=parksmap,role=frontend ...
    imagestream.image.openshift.io "parksmap" created
    deployment.apps "parksmap" created
    service "parksmap" created
--> Success
```

其他资源

- oc new-app

4.5.1. 创建路由

外部客户端可以通过路由层访问 OpenShift Container Platform 上运行的应用程序，这个路由层后端的数据对象就是一个 *路由（route）*。默认的 OpenShift Container Platform 路由器(HAProxy)使用传入请求的 HTTP 标头来确定连接的位置。

另外，您可以为路由定义安全性，如 TLS。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。

- 已安装 OpenShift CLI（**oc**）。
- 您已部署了一个镜像。
- 您必须具有 **cluster-admin** 或 **project-admin** 权限。

流程

1. 要检索创建的应用程序服务，请输入以下命令：

```
$ oc get service
```

输出示例

```
NAME      TYPE       CLUSTER-IP         EXTERNAL-IP    PORT(S)   AGE
parksmap  ClusterIP  <your-cluster-IP>  <123.456.789>  8080/TCP  8m29s
```

2. 要创建路由，请输入以下命令：

```
$ oc create route edge parksmap --service=parksmap
```

输出示例

```
route.route.openshift.io/parksmap created
```

3. 要检索创建的应用程序路由，请输入以下命令：

```
$ oc get route
```

输出示例

```
NAME       HOST/PORT                                          PATH   SERVICES   PORT
TERMINATION   WILDCARD
parksmap   parksmap-user-getting-started.apps.cluster.example.com          parksmap
8080-tcp   edge          None
```

其他资源

- oc create route edge
- oc get

4.5.2. 检查 pod

OpenShift Container Platform 使用 Kubernetes 的 pod 概念，它是共同部署在同一主机上的一个或多个容器，也是可被定义、部署和管理的最小计算单元。对容器而言，Pod 大致相当于一个机器实例（物理或虚拟）。

您可以查看集群中的 pod，并确定这些 pod 和整个集群的健康状态。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。
- 已安装 OpenShift CLI（**oc**）。
- 您已部署了一个镜像。

流程

1. 要列出带有节点名称的所有 pod，请输入以下命令：

```
$ oc get pods
```

输出示例

```
NAME                  READY   STATUS    RESTARTS   AGE
parksmap-5f9579955-6sng8   1/1     Running   0          77s
```

2. 要列出所有 pod 详情，请输入以下命令：

```
$ oc describe pods
```

输出示例

```
Name:         parksmap-848bd4954b-5pvcc
Namespace:    user-getting-started
Priority:     0
Node:         ci-ln-fr1rt92-72292-4fzf9-worker-a-g9g7c/10.0.128.4
Start Time:   Sun, 13 Feb 2022 14:14:14 -0500
Labels:       app=national-parks-app
              app.kubernetes.io/part-of=national-parks-app
              component=parksmap
              deployment=parksmap
              pod-template-hash=848bd4954b
              role=frontend
Annotations:  k8s.v1.cni.cncf.io/network-status:
                [{
                    "name": "openshift-sdn",
                    "interface": "eth0",
                    "ips": [
                        "10.131.0.14"
                    ],
                    "default": true,
                    "dns": {}
                }]
              k8s.v1.cni.cncf.io/network-status:
                [{
                    "name": "openshift-sdn",
                    "interface": "eth0",
                    "ips": [
                        "10.131.0.14"
                    ],
                    "default": true,
                    "dns": {}
                }]
```

```
              openshift.io/generated-by: OpenShiftNewApp
              openshift.io/scc: restricted
Status:       Running
IP:           10.131.0.14
IPs:
  IP:           10.131.0.14
Controlled By:  ReplicaSet/parksmap-848bd4954b
Containers:
  parksmap:
    Container ID:   cri-
o://4b2625d4f61861e33cc95ad6d455915ea8ff6b75e17650538cc33c1e3e26aeb8
    Image:
quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51a
afbae73f2abd70a83d5fa173b
    Image ID:
quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51a
afbae73f2abd70a83d5fa173b
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 13 Feb 2022 14:14:25 -0500
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-6f844 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-6f844:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
    ConfigMapName:           openshift-service-ca.crt
    ConfigMapOptional:       <nil>
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason          Age   From               Message
  ----    ------          ----  ----               -------
  Normal  Scheduled       46s   default-scheduler  Successfully assigned user-getting-
started/parksmap-848bd4954b-5pvcc to ci-ln-fr1rt92-72292-4fzf9-worker-a-g9g7c
  Normal  AddedInterface  44s   multus             Add eth0 [10.131.0.14/23] from openshift-sdn
  Normal  Pulling         44s   kubelet            Pulling image
"quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51
aafbae73f2abd70a83d5fa173b"
  Normal  Pulled          35s   kubelet            Successfully pulled image
"quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51
```

```
aafbae73f2abd70a83d5fa173b" in 9.49243308s
  Normal  Created         35s   kubelet            Created container parksmap
  Normal  Started         35s   kubelet            Started container parksmap
```

其他资源

- oc describe
- oc get
- oc label
- 查看 pod
- 查看 pod 日志

4.5.3. 扩展应用程序

在 Kubernetes 中，**Deployment** 对象定义了应用的部署方式。在大多数情况下，用户会一起使用 **Pod**, **Service**, **ReplicaSets**, 和 **Deployment** 资源。在大多数情况下，OpenShift Container Platform 会为您创建资源。

当您部署 **national-parks-app** 镜像时，会创建一个部署资源。在本例中，只部署了一个 **Pod**。

以下流程将 **national-parks-image** 扩展为使用两个实例。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。
- 已安装 OpenShift CLI（**oc**）。
- 您已部署了一个镜像。

流程

- 要将应用程序从一个 pod 实例扩展到两个 pod 实例，请输入以下命令：

```
$ oc scale --current-replicas=1 --replicas=2 deployment/parksmap
```

输出示例

```
deployment.apps/parksmap scaled
```

验证

1. 要确保应用程序正确扩展，请输入以下命令：

```
$ oc get pods
```

输出示例

```
NAME                  READY   STATUS    RESTARTS   AGE
parksmap-5f9579955-6sng8   1/1     Running   0          7m39s
parksmap-5f9579955-8tgft   1/1     Running   0          24s
```

2. 要将应用程序缩减至一个 pod 实例，请输入以下命令：

```
$ oc scale --current-replicas=2 --replicas=1 deployment/parksmap
```

其他资源

- oc scale

4.6. 部署 PYTHON 应用程序

以下流程为 **parksmap** 应用程序部署后端服务。Python 应用程序针对 MongoDB 数据库执行 2D geo-spatial 查询，以定位和返回世界上的所有国家公园的信息。

部署的后端服务是 **nationalparks**。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。
- 已安装 OpenShift CLI（**oc**）。
- 您已部署了一个镜像。

流程

1. 要创建新 Python 应用程序，请输入以下命令：

```
$ oc new-app python~https://github.com/openshift-roadshow/nationalparks-py.git --name nationalparks -l 'app=national-parks-app,component=nationalparks,role=backend,app.kubernetes.io/part-of=national-parks-app,app.kubernetes.io/name=python' --allow-missing-images=true
```

输出示例

```
--> Found image 0406f6c (13 days old) in image stream "openshift/python" under tag "3.9-ubi9" for "python"

    Python 3.9
    ----------
    Python 3.9 available as container is a base platform for building and running various Python 3.9 applications and frameworks. Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

    Tags: builder, python, python39, python-39, rh-python39

    * A source build using source code from https://github.com/openshift-
```

```
roadshow/nationalparks-py.git will be created
      * The resulting image will be pushed to image stream tag "nationalparks:latest"
      * Use 'oc start-build' to trigger a new build

--> Creating resources with label app=national-parks-
app,app.kubernetes.io/name=python,app.kubernetes.io/part-of=national-parks-
app,component=nationalparks,role=backend ...
    imagestream.image.openshift.io "nationalparks" created
    buildconfig.build.openshift.io "nationalparks" created
    deployment.apps "nationalparks" created
    service "nationalparks" created
--> Success
```

2. 要创建公开应用程序 **nationalparks** 的路由，请使用以下命令：

```
$ oc create route edge nationalparks --service=nationalparks
```

输出示例

```
route.route.openshift.io/parksmap created
```

3. 要检索创建的应用程序路由，请输入以下命令：

```
$ oc get route
```

输出示例

```
NAME            HOST/PORT                                                PATH   SERVICES
PORT       TERMINATION   WILDCARD
nationalparks    nationalparks-user-getting-started.apps.cluster.example.com
nationalparks   8080-tcp   edge          None
parksmap         parksmap-user-getting-started.apps.cluster.example.com
parksmap        8080-tcp   edge          None
```

其他资源

- oc new-app

4.7. 连接到数据库

部署并连接一个 MongoDB 数据库，其中的 National **-parks-app** 应用存储位置信息。将 **national-parks-app** 应用程序标记为地图可视化工具的后端后，**parksmap** 部署会使用 OpenShift Container Platform 发现机制来自动显示地图。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。
- 已安装 OpenShift CLI（**oc**）。
- 您已部署了一个镜像。

流程

- 要连接到数据库，请输入以下命令：

```
$ oc new-app quay.io/centos7/mongodb-36-centos7 --name mongodb-nationalparks -e
MONGODB_USER=mongodb -e MONGODB_PASSWORD=mongodb -e
MONGODB_DATABASE=mongodb -e MONGODB_ADMIN_PASSWORD=mongodb -l
'app.kubernetes.io/part-of=national-parks-app,app.kubernetes.io/name=mongodb'
```

输出示例

```
--> Found container image dc18f52 (8 months old) from quay.io for
"quay.io/centos7/mongodb-36-centos7"

    MongoDB 3.6
    -----------
    MongoDB (from humongous) is a free and open-source cross-platform document-oriented
database program. Classified as a NoSQL database program, MongoDB uses JSON-like
documents with schemas. This container image contains programs to run mongod server.

    Tags: database, mongodb, rh-mongodb36

    * An image stream tag will be created as "mongodb-nationalparks:latest" that will track this
image

--> Creating resources with label app.kubernetes.io/name=mongodb,app.kubernetes.io/part-
of=national-parks-app ...
    imagestream.image.openshift.io "mongodb-nationalparks" created
    deployment.apps "mongodb-nationalparks" created
    service "mongodb-nationalparks" created
--> Success
```

其他资源

- oc new-project

4.7.1. 创建 secret

Secret 对象提供了一种机制来保存敏感信息，如密码、OpenShift Container Platform 客户端配置文件和私有源存储库凭证等。secret 将敏感内容与 Pod 分离。您可以使用卷插件将 secret 信息挂载到容器中，系统也可以使用 secret 代表 Pod 执行操作。以下流程添加了 secret **nationalparks-mongodb-parameters**，并将它挂载到 **nationalparks** 工作负载中。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。
- 已安装 OpenShift CLI（**oc**）。
- 您已部署了一个镜像。

流程

1. 运行以下命令来创建 secret：

```
$ oc create secret generic nationalparks-mongodb-parameters --from-
literal=DATABASE_SERVICE_NAME=mongodb-nationalparks --from-
literal=MONGODB_USER=mongodb --from-literal=MONGODB_PASSWORD=mongodb --
from-literal=MONGODB_DATABASE=mongodb --from-
literal=MONGODB_ADMIN_PASSWORD=mongodb
```

输出示例

```
secret/nationalparks-mongodb-parameters created
```

2. 要更新环境变量以将 mongodb secret 附加到 **nationalpartks** 工作负载，请输入以下命令：

```
$ oc set env --from=secret/nationalparks-mongodb-parameters deploy/nationalparks
```

输出示例

```
deployment.apps/nationalparks updated
```

3. 要显示 **nationalpartks** 部署的状态，请输入以下命令：

```
$ oc rollout status deployment nationalparks
```

输出示例

```
deployment "nationalparks" successfully rolled out
```

4. 要显示 **mongodb-nationalparks** 部署的状态，请输入以下命令：

```
$ oc rollout status deployment mongodb-nationalparks
```

输出示例

```
deployment "nationalparks" successfully rolled out
deployment "mongodb-nationalparks" successfully rolled out
```

其他资源

- oc create secret generic
- oc set env
- oc rollout status

4.7.2. 载入数据并显示国家公园地图

您已部署了 **parksmap** 和 **Nationalparks** 应用程序，然后部署 **mongodb-nationalparks** 数据库。但是，没有将数据 *加载到* 数据库中。

先决条件

- 有访问 OpenShift Container Platform 集群的权限。

- 已安装 OpenShift CLI（**oc**）。
- 您已部署了一个镜像。

流程

1. 要加载国家公园数据，请输入以下命令：

```
$ oc exec $(oc get pods -l component=nationalparks | tail -n 1 | awk '{print $1;}') -- curl -s http://localhost:8080/ws/data/load
```

输出示例

```
"Items inserted in database: 2893"
```

2. 要验证您的数据是否已正确加载，请输入以下命令：

```
$ oc exec $(oc get pods -l component=nationalparks | tail -n 1 | awk '{print $1;}') -- curl -s http://localhost:8080/ws/data/all
```

输出示例（修剪）

```
, {"id": "Great Zimbabwe", "latitude": "-20.2674635", "longitude": "30.9337986", "name": "Great Zimbabwe"}]
```

3. 要为路由添加标签，请输入以下命令：

```
$ oc label route nationalparks type=parksmap-backend
```

输出示例

```
route.route.openshift.io/nationalparks labeled
```

4. 要检索您的路由来查看您的地图，请输入以下命令：

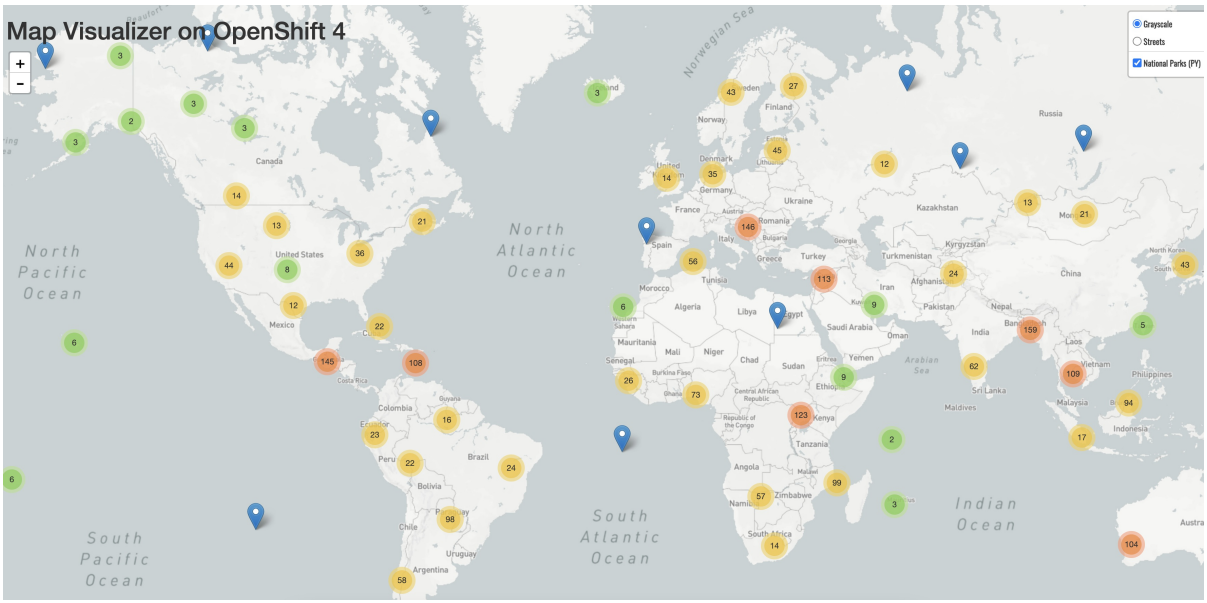
```
$ oc get routes
```

输出示例

```
NAME          HOST/PORT                                                  PATH   SERVICES      PORT       TERMINATION   WILDCARD
nationalparks   nationalparks-user-getting-started.apps.cluster.example.com
nationalparks   8080-tcp   edge          None
parksmap      parksmap-user-getting-started.apps.cluster.example.com                parksmap
8080-tcp   edge          None
```

5. 将上面检索到的 **HOST/PORT** 路径复制并粘贴到您的网页浏览器中。您的浏览器应当显示全球的国家公园地图。

图 4.1. 跨世界的国家公园



其他资源

- oc exec
- oc label
- oc get