红帽 OpenShift 4 基础（12小时）

架构概述

概述

红帽® OpenShift® 容器平台

- 基于 Kubernetes 的容器编排平台
- 运维和开发人员都能受益
- 为开发人员和 IT 组织提供云应用平台
  - 用于在安全、可扩展的资源基础之上部署应用
  - 最小配置和管理开销
- 支持 Java™、Python、Ruby、Node.js、Perl、PHP 和 .NET 等
- 构建于 Kubernetes 基础之上
- OpenShift 容器平台的 Control Plane 仅可用于部署到 RHCOS
- OpenShift 容器平台工作负载可以部署到 RHCOS 或红帽企业 Linux® (RHEL)
  - RHCOS 仅可用于 OpenShift 部署，不可作常规用途
  - RHCOS 使用新的专用工具编纂 OpenShift 运维专业知识
- RHCOS 符合 FIPS 的要求
- 将 Kubernetes 平台引入到客户数据中心和云
  - 符合安全性、隐私、合规性和监管要求

产品和基础架构提供商

支持的基础架构

- OpenShift 在 RHCOS 和 RHEL 上受支持
- OpenShift 4 混合云解决方案
- 在公共云和私有云以及裸机上受支持

- 支持全堆栈自动化 (IPI) 的平台：
  - AWS、Microsoft Azure、Google Cloud Platform
  - 红帽虚拟化、红帽 OpenStack® 平台
- 支持预先存在的基础架构 (UPI) 的平台：
  - AWS、Microsoft Azure、Google Cloud Platform
  - VMware vSphere、红帽 OpenStack 平台、IBM Z
  - IBM Power Systems、裸机
- 未来发行版本将支持更多基础架构

OpenShift 3.11

- 在 RHEL 运行的任何位置均受支持
  - 裸机物理计算机、虚拟化基础架构，以及私有云或认证公共云中
  - 虚拟化平台：红帽虚拟化、vSphere、Hyper-V
  - 红帽 OpenStack 平台，Amazon、Google 和 Azure 等认证公共云提供商
  - x86 和 IBM Power 服务器架构

OpenShift Kubernetes Engine

- 用户探索 OpenShift 4 Kubernetes Engine，而非整个平台
- 核心 Kubernetes 功能以及 ISV 大生态系统
- 畅享 RHCOS 不可变安全架构
- 适合 DIY、*ks 或更低端应用

架构和概念

节点主机

- OpenShift 在 RHCOS 和 RHEL 上运行
- OpenShift 具有两种类型的节点：
  - 工作机
  - 主控机
- 节点为：

    - 安装有 OpenShift 的 RHEL 或 RHCOS 实例
    - 工作机：用于运行最终用户应用
    - 主控机：管理集群
- 节点 守护进程和其他软件在所有节点上运行

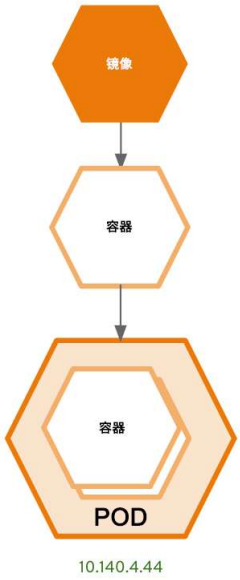
容器

- 应用实例和组件在符合 OCI 规范的容器中运行
- 镜像：应用“二进制”
- 容器：运行时
- OpenShift 工作节点可以运行多个容器
- 节点容量与底层资源的内存和 CPU 能力相关
    - 云、硬件或虚拟化

Pod

- 一台主机上可以部署一个或多个容器
    - 由一组同位容器组成，具有卷和 IP 地址等共享的资源
    - 定义、部署和管理的最小计算单元
- 可以包含一个或多个紧密耦合的同位应用，这些应用利用共享的上下文运行
    - 示例：Web 服务器以及用来拉取和同步文件的应用

- 在容器化环境中对应用相关逻辑主机进行建模
- 在容器出现之前的环境中，应用在同一物理或虚拟主机上执行
- 在 OpenShift 中，pod 取代个体应用容器，成为最小的可部署单元



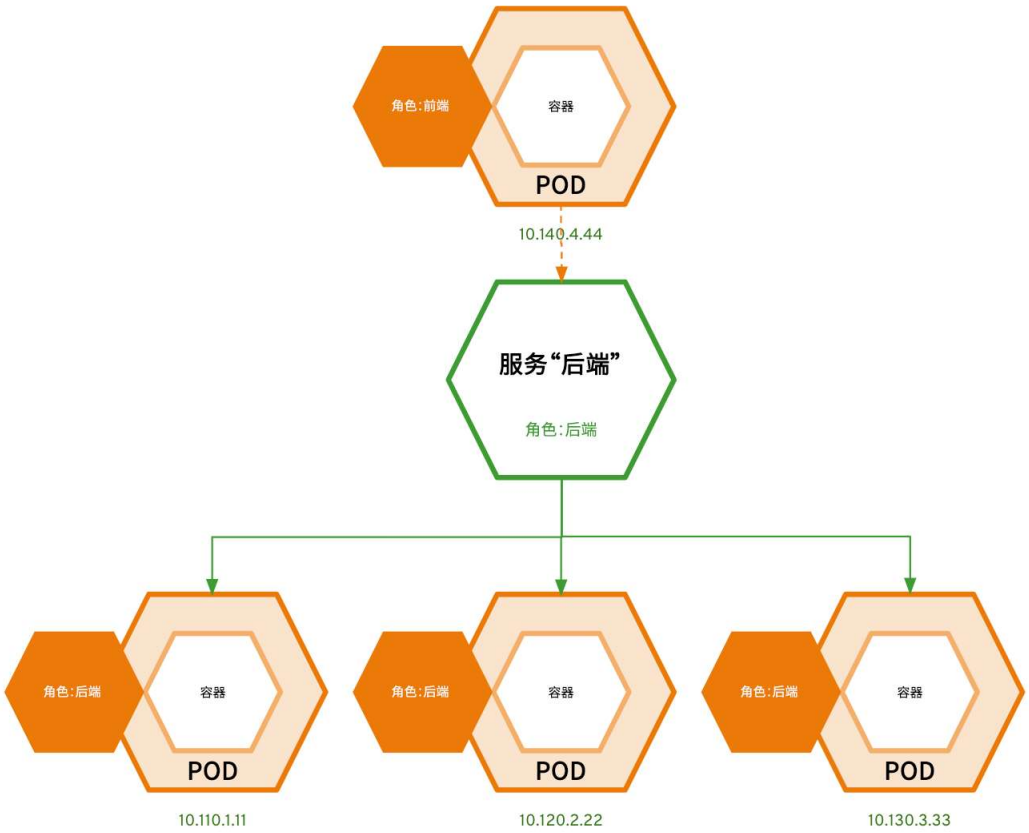
- OpenShift 中调度的单元
  - OpenShift 调度和运行同一节点上 pod 内的所有容器
- 复杂的应用由许多 pod 组成，各自有自己的容器
  - 与外部交互，并在 OpenShift 环境内互相交互
- OpenShift 在由称“pod”的元对象包裹的容器内运行容器镜像
- 单一 pod 中可能有多个容器
  - 示例：支持“附接”容器等集群功能
- 大部分应用可受益于单容器 pod 的灵活性
  - 应用服务器和数据库等不同的组件通常不会放在单一 pod 内
  - 让个体应用组件能够轻松水平扩展
- 应用组件通过服务“连线”在一起

- 应用或某一实例
- 使用 oc get pod 查看环境（通常是您的项目）中运行的 pod：
  - 项目也称为“命名空间”
  - 一些 pod 运行至完成为止，如构建器和部署器
  - 一些 pod 持续运行应用

```
$ oc get pods -n 0016-openshift-tools
NAME                                READY   STATUS      RESTARTS   AGE
cakephp-mysql-example-1-build       0/1     Completed   0          2d19h
cakephp-mysql-example-1-deploy      0/1     Completed   0          2d19h
cakephp-mysql-example-1-hook-pre    0/1     Completed   0          2d19h
cakephp-mysql-example-1-tgs4q       1/1     Running     0          2d19h
mysql-1-2qvhn                       1/1     Running     0          2d19h
mysql-1-deploy                      0/1     Completed   0          2d19h
```

服务

- 定义 pod 的逻辑集合和访问策略
  - 为其他应用提供永久的内部 IP 地址和主机名，以便在 pod 创建和销毁时使用
- 服务层将应用组件连在一起
  - 示例：前端 Web 服务通过与数据库的服务通信来连接数据库实例
- 服务可实现应用组件之间简单的内部负载平衡
  - OpenShift 将服务信息自动注入到运行中的容器，以简化发现过程



标签

- 用于组织、分组或选择 API 对象
  - 示例：为 pod 标上标签，服务利用标签选择器确定它们代理到的 pod
  - 使得服务能够引用多组 pod
  - 能够将可能具有不同容器的 pod 视为互为相关的实体
- 大多数对象可以在元数据中包含标签
- 利用标签任意分组相关的对象
  - 示例：应用的所有 pod、服务、复制控制器和部署配置
- 标签是简单的键值对：

```
标签:
  key1: value1
  key2: value2
```

主控节点

- RHCOS 的实例
- 主要功能：
  - 编排工作节点上的活动
  - 了解和维护 OpenShift 环境内的状态
- 三个主控机实现高可用性
- 蓝色部分是 Kubernetes
- 绿色部分是 OpenShift
  - 一些基础架构服务是 OpenShift 独有的

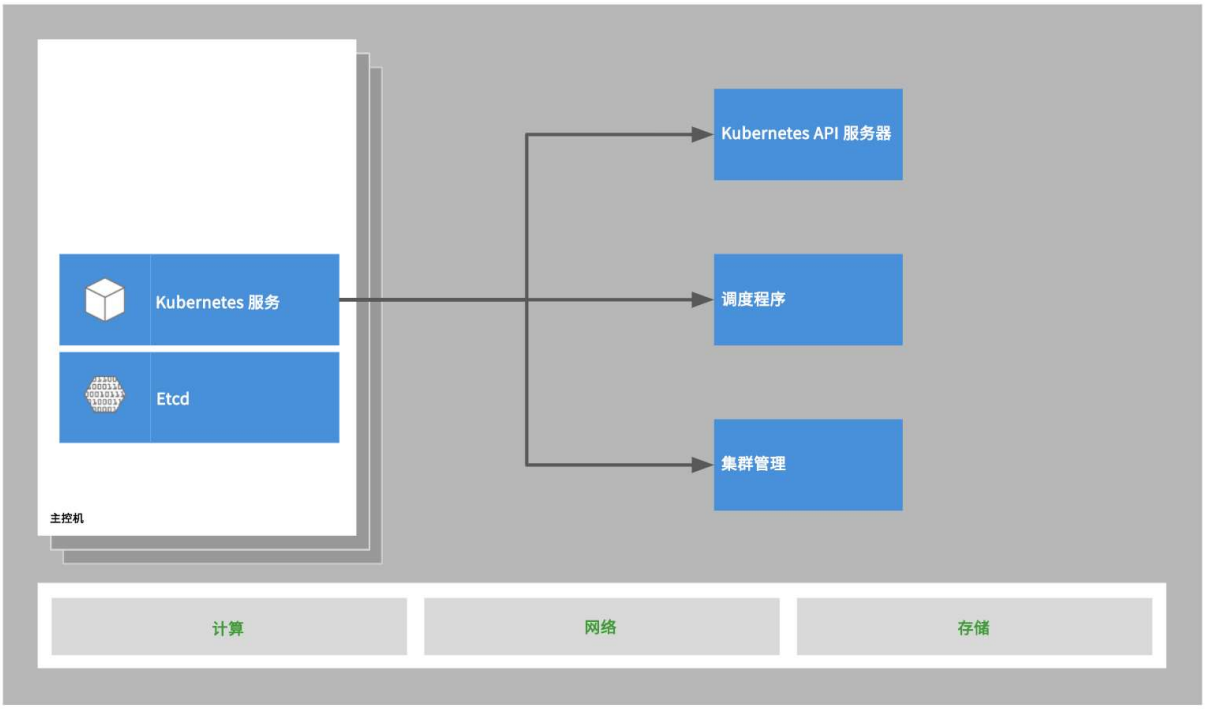
etcd

- 所需状态和当前状态存放在数据存储中，它将 etcd 用作分布式键值存储
- etcd 也存放：
  - RBAC 规则
  - 应用环境信息
  - 非应用用户数据

主控机服务 - 核心 Kubernetes 组件

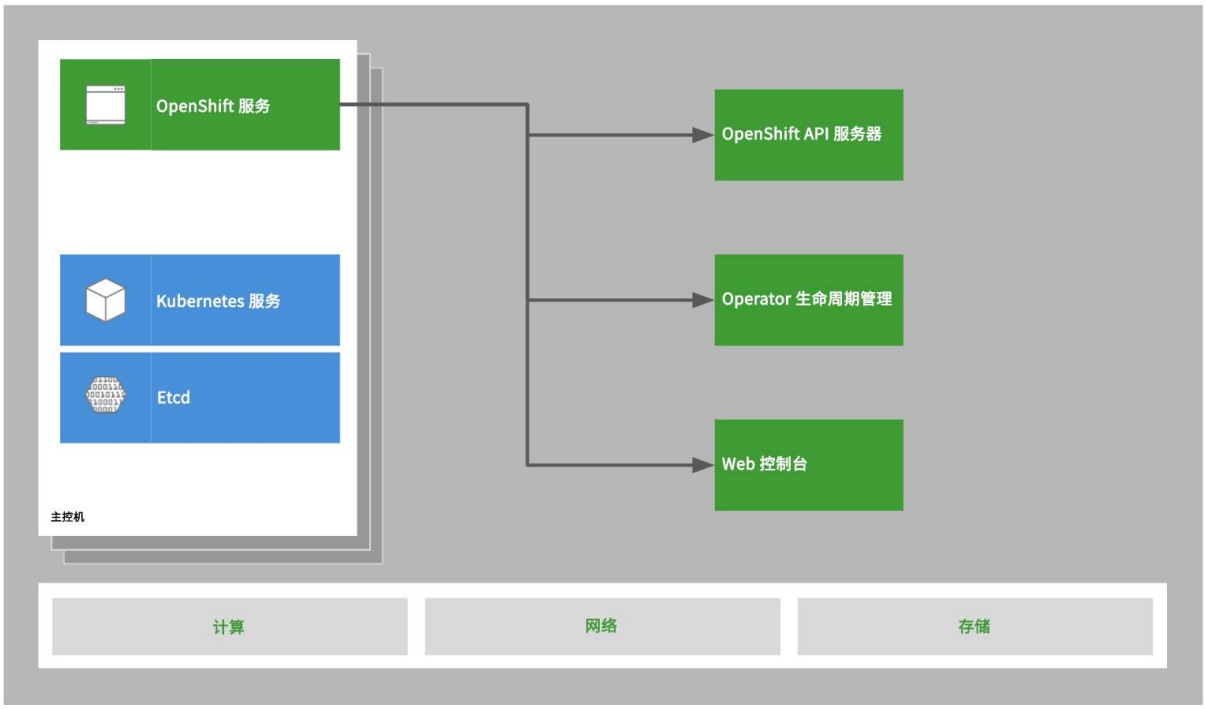
- OpenShift 包含 Kubernetes 服务
- 与云原生计算基金会的 Kubernetes 逐字节一致

- 提供：
  - Kubernetes API 服务器
  - 调度程序
  - 集群管理服务



主控机服务 - 核心 OpenShift 组件

- OpenShift 搭载核心 OpenShift 组件
- OpenShift API 服务器
  - OpenShift 独有的 API 调用
- Operator Lifecycle Manager (OLM)
  - 无线更新 Operator 工作负载和中间件等
- Web 控制台

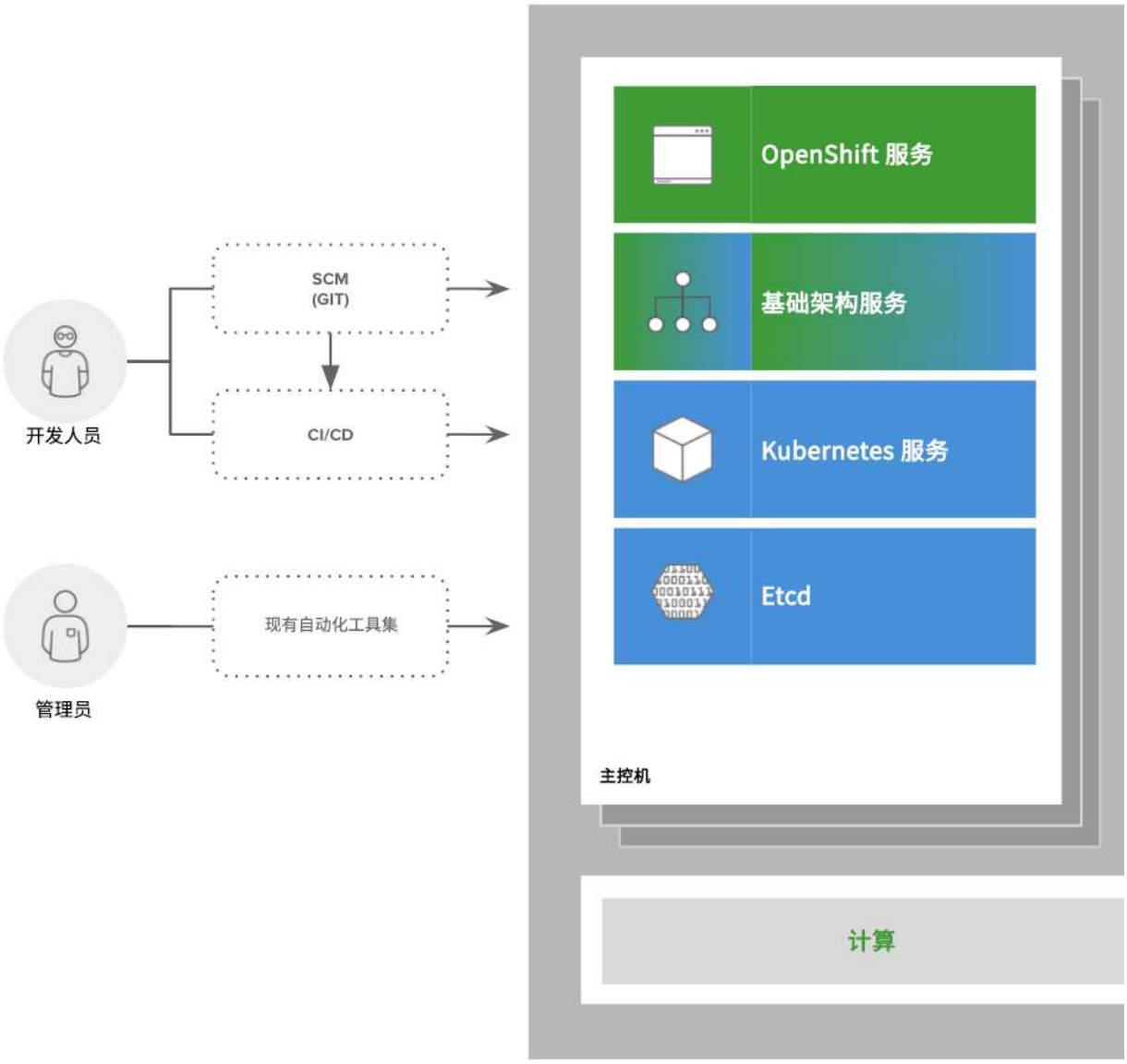


主控机 - 基础架构服务

- 主控机和工作机协同工作
- 同时包含 OpenShift 和 Kubernetes 服务
- 作为 pod 运行；由主控机 control plane 编排
- 服务包括：
  - 监控
  - 日志记录
  - OS 调优
  - 软件定义网络 (SDN)
  - DNS
  - Kubelet（OpenShift 节点进程）

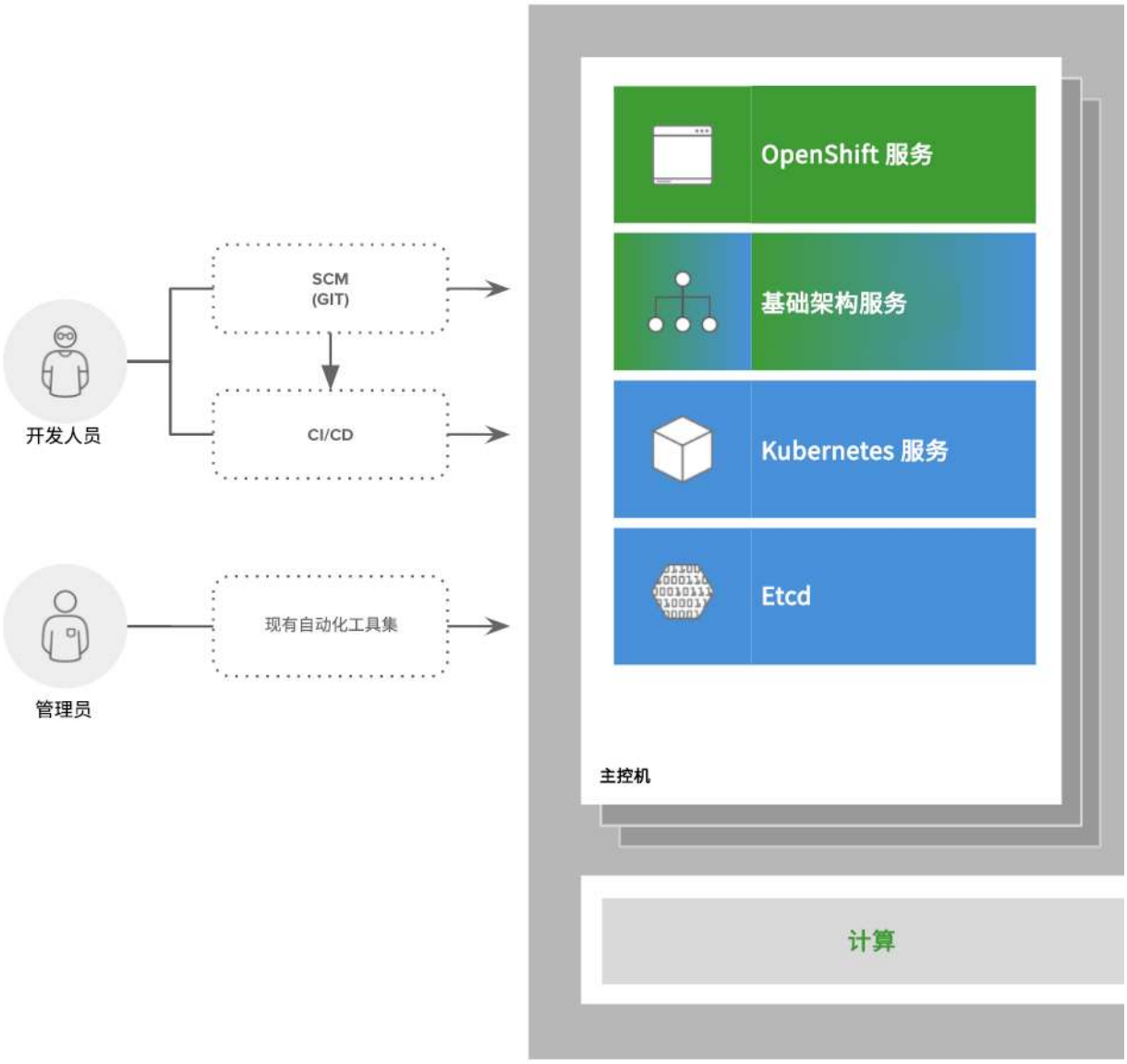
主控机 - API 和身份验证

- 主控机提供单一 API 端点，供所有工具和系统与之交互
  - 包括 OpenShift 和 Kubernetes
  - 所有管理请求都经由此 API
- 所有 API 请求都被 SSL 加密并进行身份验证
- 授权通过精细的基于角色的访问控制 (RBAC) 进行处理
- 主控机可以绑定到外部身份管理系统
  - 示例：LDAP、Active Directory，以及 GitHub 和 Google 等 OAuth 提供商



访问

- 所有用户通过相同的标准接口访问 OpenShift
- Web UI、CLI、IDE 全部经过身份验证、由 RBAC 控制的 API
- 用户不需要 OpenShift 节点的系统级访问权限
- CI/CD 系统通过这些接口与 OpenShift 集成
- 在 RHCOS 上部署的 OpenShift 支持使用“新一代”系统管理和监控工具
- 在 RHEL 上部署的 OpenShift 支持使用“现有的”系统管理和监控工具



运行状况和扩展

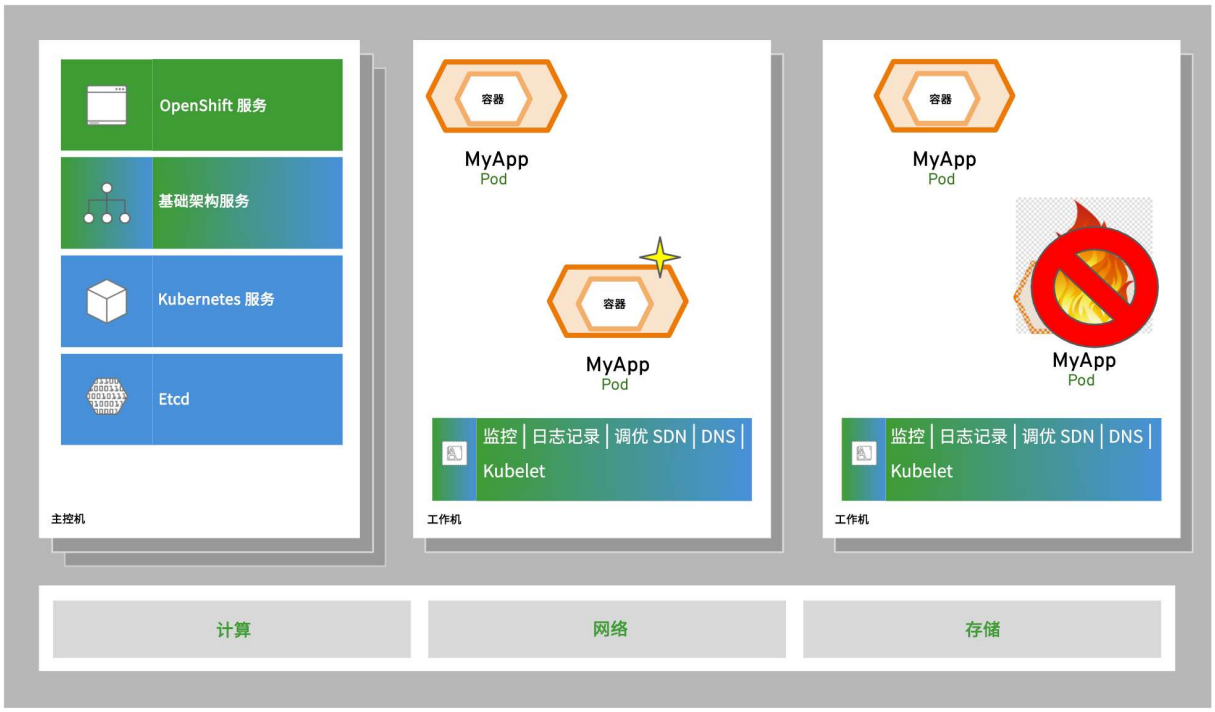
- 主控机监控 pod 的运行状况并且自动扩展它们
  - 用户配置 pod 存活度和就绪度探查
  - Pod 根据 CPU 利用率指标自动扩展

运行不良的 pod

- 当主控机发现 pod 未通过其探查时会发生什么?
- Pod 内的容器因为崩溃或其他问题退出时会发生什么?

修复 pod 故障

- 主控机自动重启未通过探查或因为容器崩溃而退出的 pod
- 失败过于频繁的 pod 被加上不良标记，暂时不予重启
- 服务层仅将流量发送到运行良好的 pod
    - 主控机自动进行编排来维护组件可用性



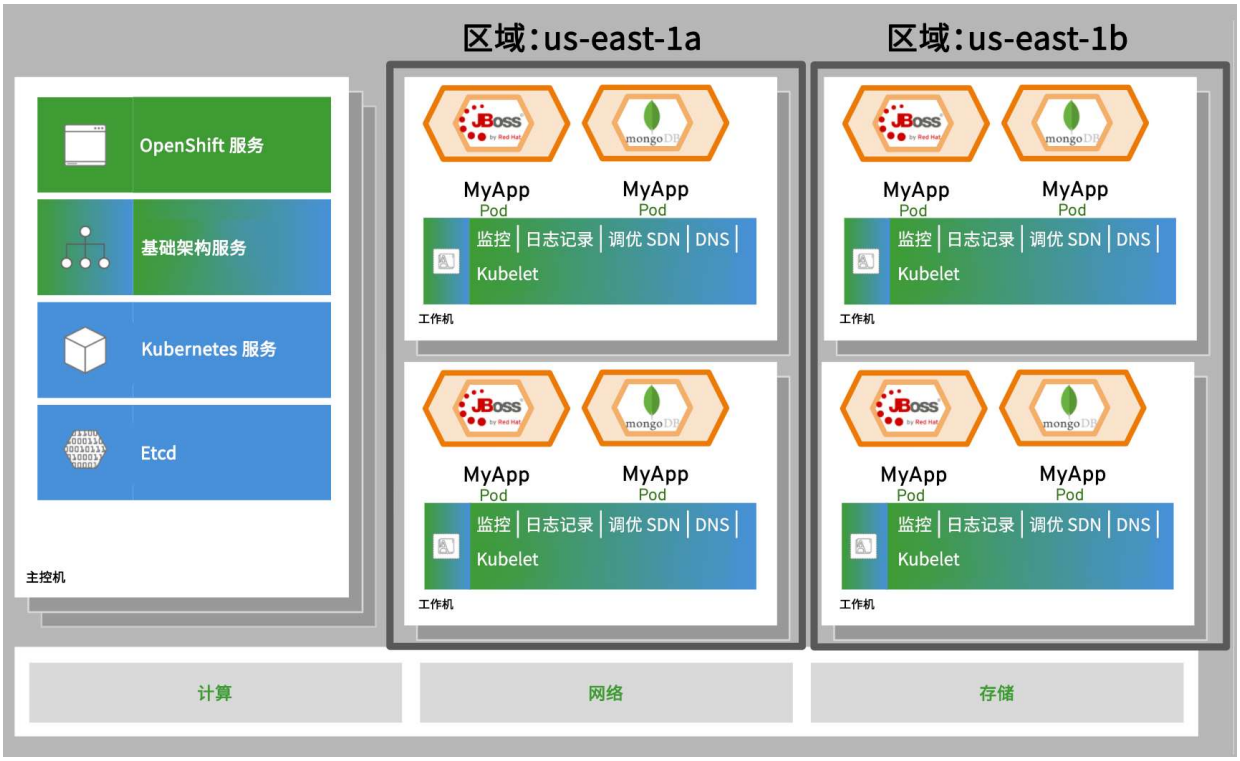
调度程序

- 负责决定 pod 放置的组件
- 将 pod 放置到工作节点上时，考虑当前的内存、CPU 和其他环境利用率
- 为实现应用高可用性，可在多个工作节点间分散 pod 副本

调度程序配置

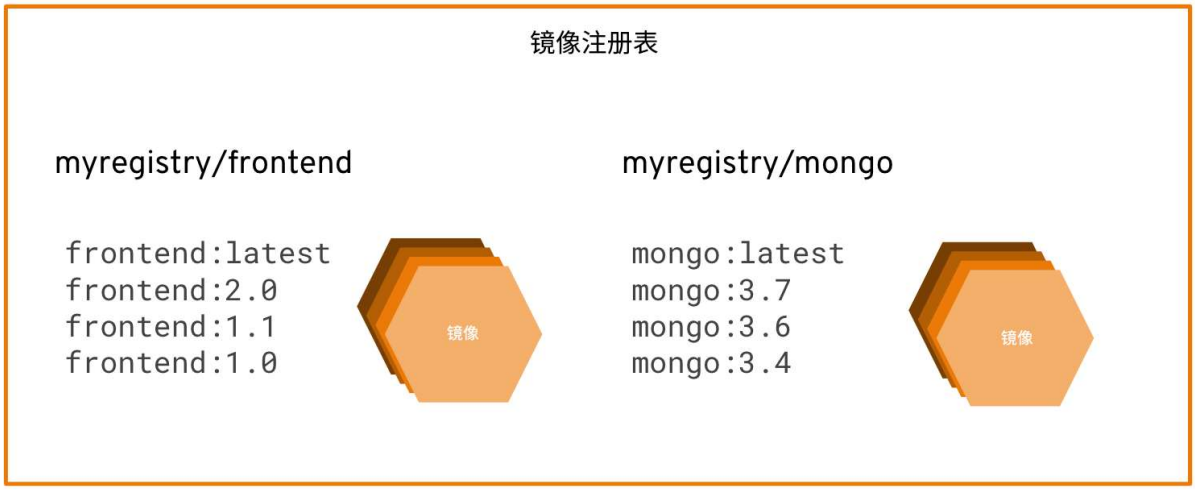
- 可以利用现实中 OpenShift 部署的拓扑（地区、区域，等等）
- 处理复杂的工作负载调度场景
- 将配置与节点组和标签结合使用
    - 示例：使用地区和区域来划分 OpenShift 环境，使其与现实中的拓扑一样

- OpenShift 4 可自动调度更多基础架构



集成式容器注册表

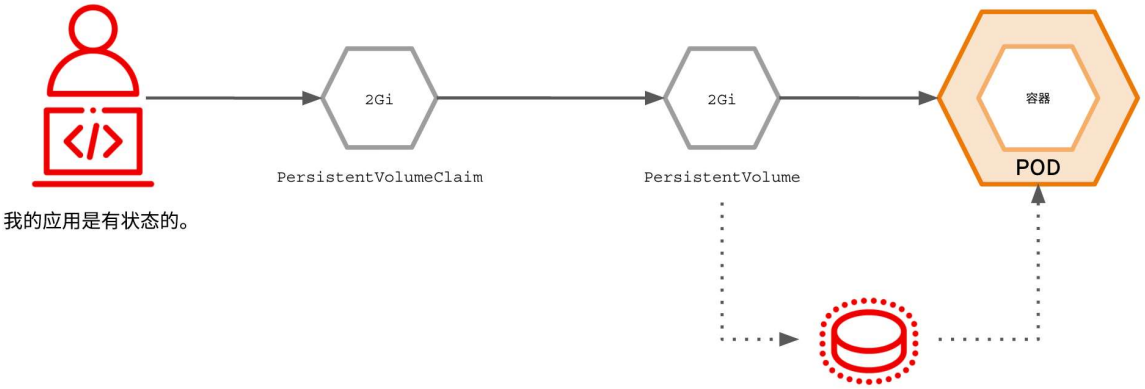
- OpenShift 容器平台包含集成式容器注册表，用于存储和管理容器镜像
- 当新镜像推送到注册表时，注册表会通知 OpenShift 并传递镜像信息，包括：
  - 命名空间
  - 名称
  - 镜像元数据
- 不同 OpenShift 组件可以通过创建新构建和部署对新镜像做出反应



应用数据

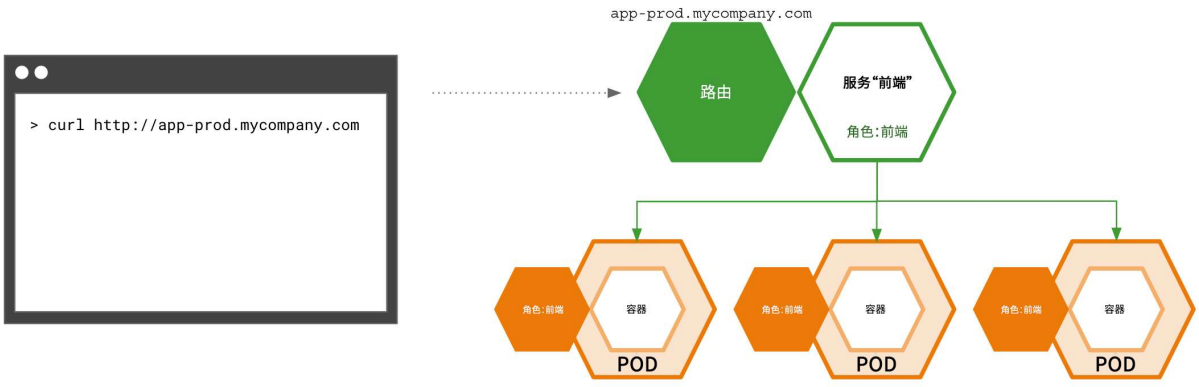
- 容器默认是非持久化的
  - 数据不会在容器重启或创建时保存下来
- OpenShift 提供持久存储子系统，可自动将实际存储连接到正确的 pod

    - 允许使用有状态的应用
- OpenShift 容器平台提供各种持久存储类型，包括：
    - 裸设备：iSCSI、光纤通道
    - 企业级存储：NFS
    - 云类型选择：Ceph®、AWS EBS、pDisk



路由层

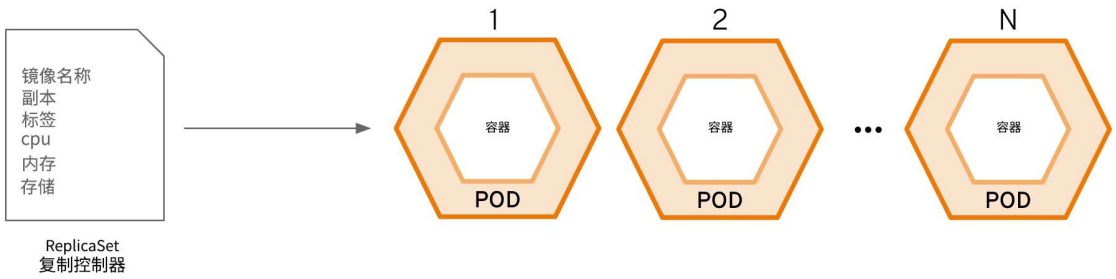
- 为外部客户端提供访问 OpenShift 内运行的应用的途径
- 服务层的亲密伙伴
- 在 OpenShift 内的 pod 中运行
- 提供：
    - 为外部客户端提供 pod 自动化负载平衡
    - 围绕运行不良的 pod 进行负载平衡和自动路由
- 路由层可插拔并可扩展
    - 选项包括非 OpenShift 软件路由器



ReplicaSet 和复制控制器

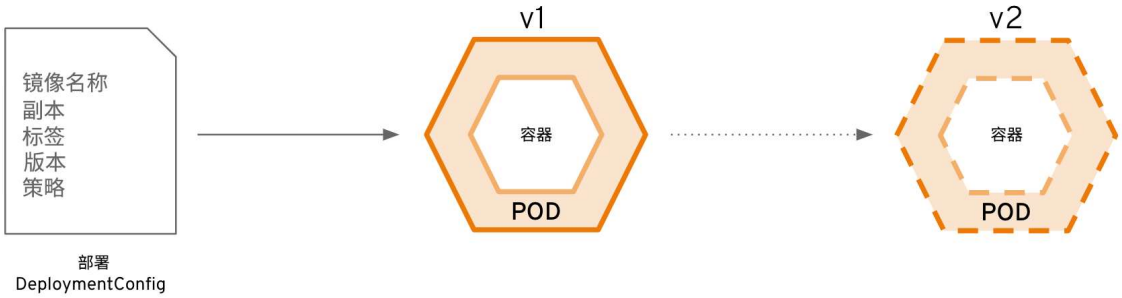
- 两种实现：
    - ReplicaSet = Kubernetes
    - 复制控制器 = OpenShift
- 确保时刻运行指定数量的 pod 副本
- 如果 pod 退出或被删除，复制控制器会实例化更多数目

- 如果运行中的 pod 超过所需的数目，ReplicaSet 会根据需要删除相应的数量



Deployment 和 DeploymentConfig

- 定义如何推出 pod 的新版本
- 识别：
  - 镜像名称
  - 副本数
  - 为目标部署节点标上标签
- 基于以下标准更新 pod：
  - 版本
  - 策略
  - 更改触发器
- 创建 ReplicaSet 或复制控制器



Operator

Kubernetes Operator

- Kubernetes 原生应用
  - 将所有运维知识置于 Kubernetes 原语中
  - 管理员、shell 脚本、自动化软件（如 Ansible®）现在位于 Kubernetes pod 中
  - 与 Kubernetes 概念和 API 原生集成

设计

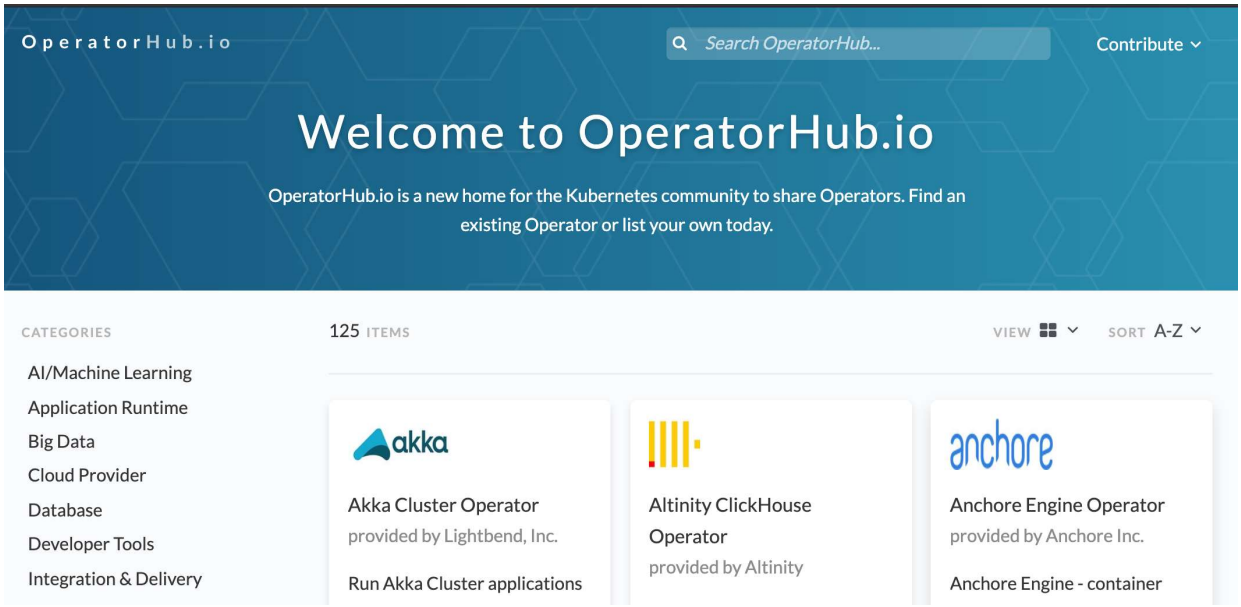
- Operator：
  - 含有与 Kubernetes API 服务器交互的 operator 代码的 pod
  - 运行“协调循环”来对应用服务进行检查
    - 确保对象达到用户指定的状态
  - 管理所有部署的资源和您的应用
  - 充当应用专用的控制器
  - 利用自定资源定义 (CRD) 扩展 Kubernetes API

Operator Framework

- Operator SDK
  - 开发人员构建、打包和测试 Operator
  - 不需要复杂的 Kubernetes API 知识
- Operator Lifecycle Manager (OLM)
  - 帮助安装和更新集群中的所有 operator 并管理其生命周期
- Operator Metering
  - Kubernetes 内部 Operator 和资源的使用情况报告

OperatorHub.io

- 用于分享 Operator 的 Kubernetes 互联网社区
- 适用于任何 Kubernetes 环境
- 打包 Operator 以实现轻松部署和管理
- 公开 Operator 并实现采用率
- 利用 OLM 安装、管理和刚更新 Operator



网络工作流

OpenShift 网络

- 容器网络基于集成式 Open vSwitch
- 平台范围的路由层
- 能够插入第三方 SDN 解决方案
- 与 DNS 和现有路由及负载平衡集成

路由

- 通过提供可从外部连接的主机名来公开服务
- 包含路由名称、服务选择器和（可选）安全配置
- 路由器可以使用定义的路由以及由服务标识的端点
    - 提供命名的连接
    - 让外部客户端能够访问 OpenShift 托管的应用

路由器

- 可轻松部署的多层应用
    - 从 OpenShift 环境外面访问应用所需的路由层
- 路由器容器可以在环境中的任何节点主机上运行
    - 管理员在 DNS 服务器上创建通配符 DNS 条目（CNAME 或 A 记录）
    - DNS 条目解析到托管路由器容器的节点主机
- 路由器是目的地为 OpenShift 托管节点的流量的入口点
    - 路由器容器解析外部请求 (https://myapp.cloudapps.openshift.opentlc.com)
        - 将请求代理到正确的 pod

注意：请勿将**路由器**和**路由**资源混淆。

场景

- 外部客户端将浏览器指向 myApp.apps.openshift.opentlc.com:80
    - DNS 解析到运行路由器容器的主机

    - 使用 openshift-sdn 覆盖网络：
        - 路由器检查是否存在路由供请求使用
        - 将请求代理到内部 pod IP:端口 (10.1.2.3:8080)

Pod 连通性

- pod 利用 OpenShift 节点主机的网络来连接其他 pod 和外部网络

场景

- pod 将数据包传输到 OpenShift 环境内另一节点主机中的 pod
    - 容器使用 IP 10.1.2.3:8080 将数据包发送到目标 pod
    - OpenShift 节点使用 Open vSwitch 将数据包路由到托管目标容器的 OpenShift 节点
    - 接收方节点将数据包路由到目标容器

服务和 pod

- 服务通常用于为一组相似的 pod 提供永久 IP 地址
- 在内部，服务在被访问时负载平衡到相应的支持 pod
- 可以在服务中任意添加或删除支持 pod，而服务一直保持可用
    - 让依赖于服务的对象能够以一贯的内部地址来引用它
- 服务具有位于 OpenShift 内部的 DNS 名称
    - 示例：my-service.my-project.svc.cluster.local
    - Pod 可以访问内部 DNS

场景

- Pod 将数据包传输到代表一个或多个 pod 的服务
    - 容器使用 IP 172.30.0.99:9999 将数据包发送到目标服务
    - 当服务被请求时，OpenShift 节点将数据包代理到由服务 (10.1.2.3:8080) 代表的 pod

容器部署工作流

场景

- 通过 CLI、Web 控制台或 API 请求新应用
    - OpenShift API/身份验证：
        - 权衡用户的权限、资源配额和其他信息，以批准请求
        - 根据需要创建支持的资源：部署配置、复制控制器、服务、路由、持久存储声明
    - OpenShift 调度程序：
        - 权衡资源的可用性和负载，以及应用在节点之间的分散情况（以实现应用高可用性），为每个 pod 委派工作节点
    - OpenShift 工作节点：
        - 从外部或集成式注册表，拉取要使用的镜像
        - 在工作节点上启动容器 (pod)

总结

- 概述
- 产品和基础架构提供商
- 架构和概念
- Operator
- 网络工作流
- 容器部署工作流