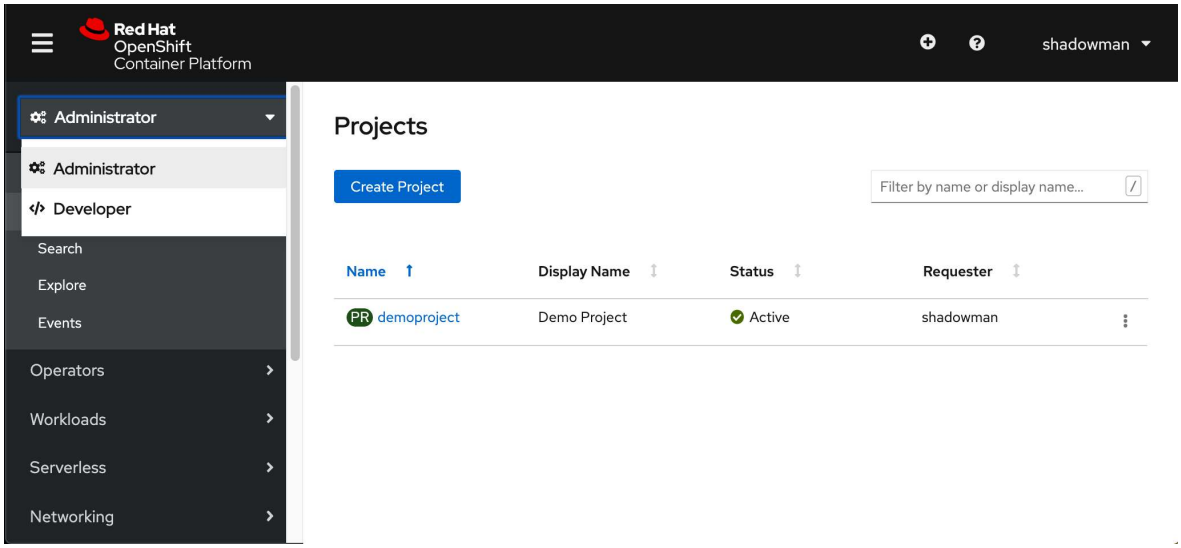
红帽 OpenShift 4 基础（12小时）

用户体验

Web 控制台

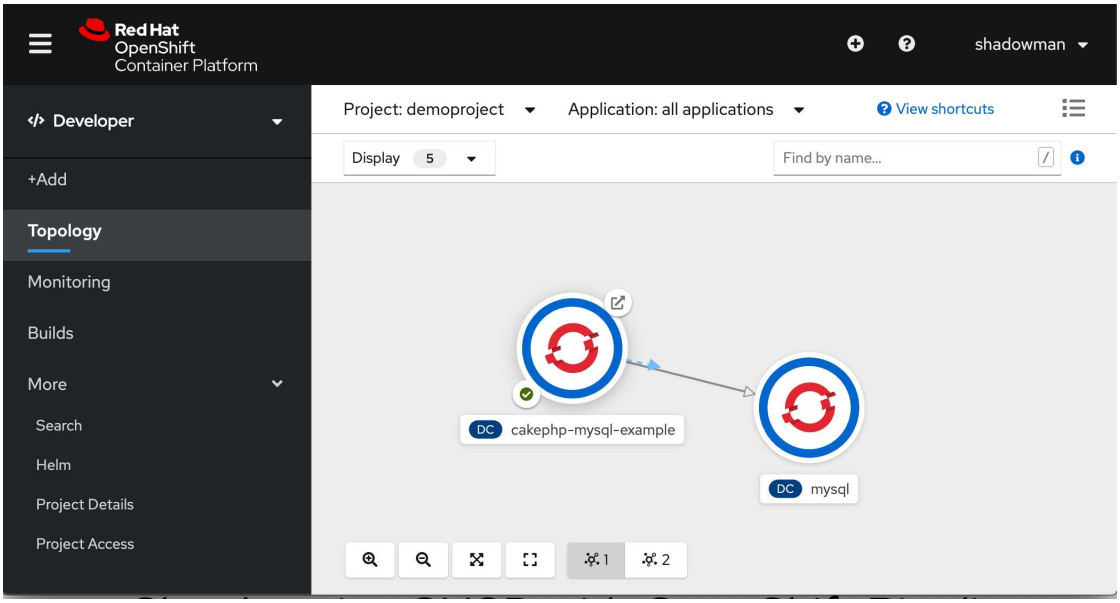
概述

- 两种“视角”：
  - 管理员
  - 开发人员
- 作为 pod 运行
- 可自定义
- 内置指标

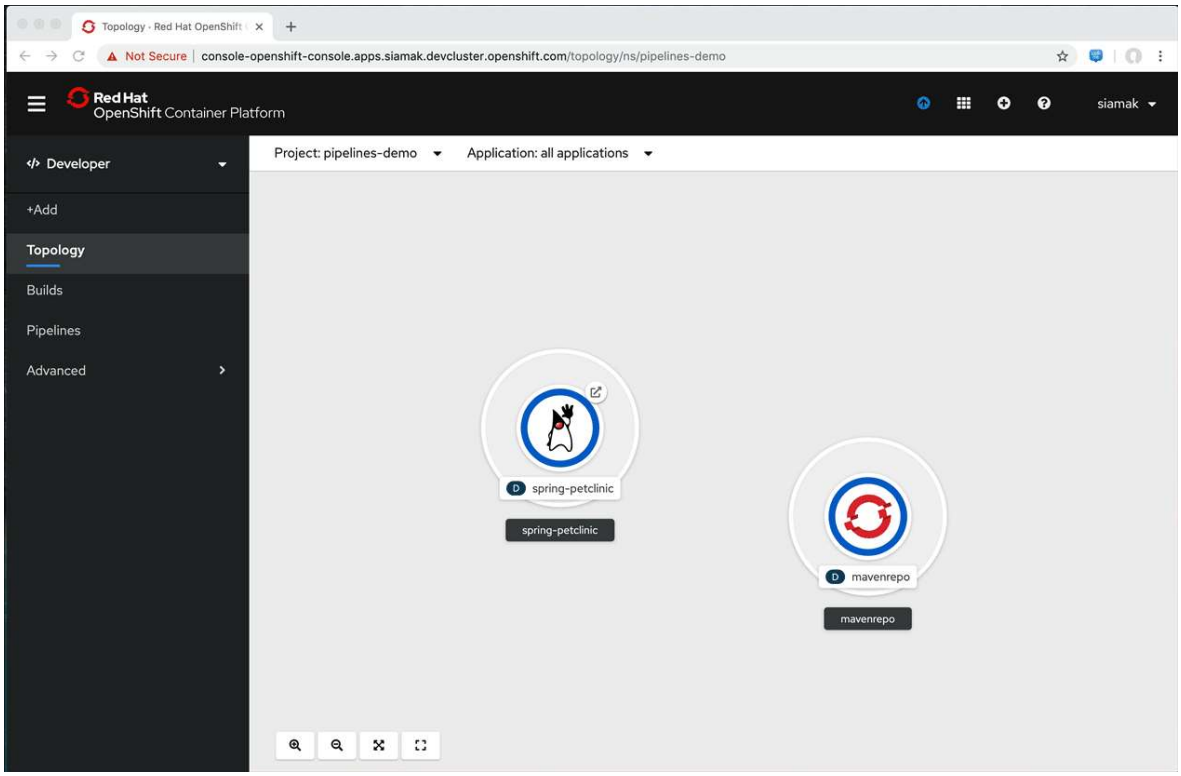


开发人员视角

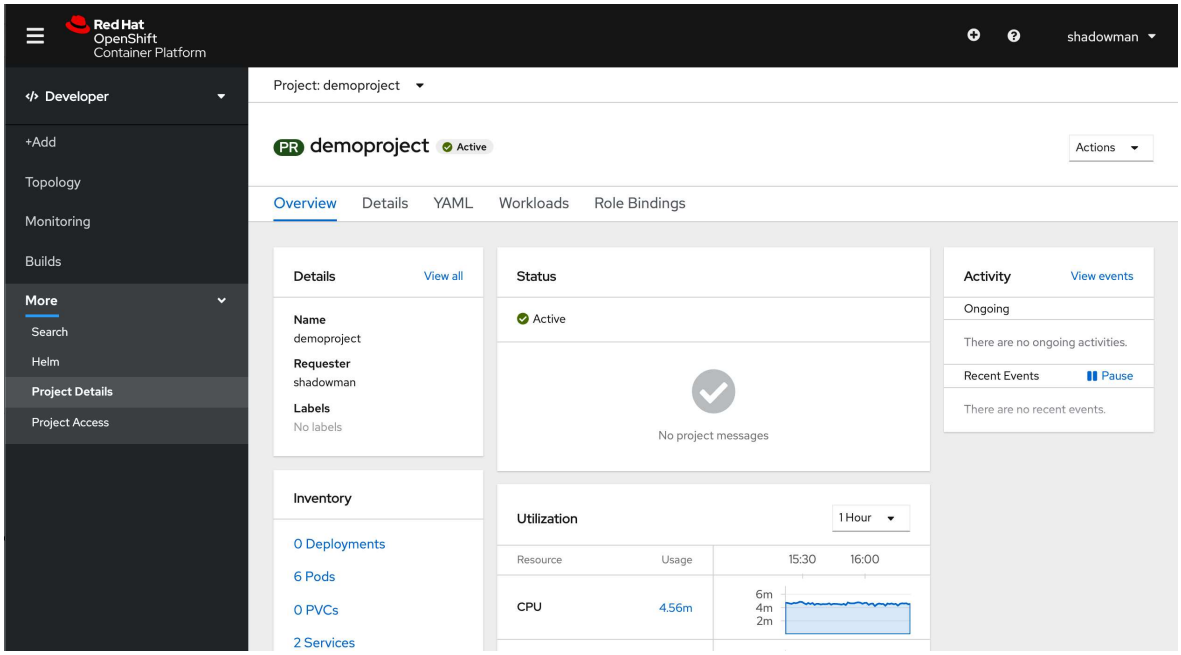
- 拓扑视图
  - 以应用为中心
  - 显示组件，以及状态、路由、源代码
  - 拖放箭头来创建关系
  - 轻松为应用添加组件



- 构建和流水线视图（若已安装）

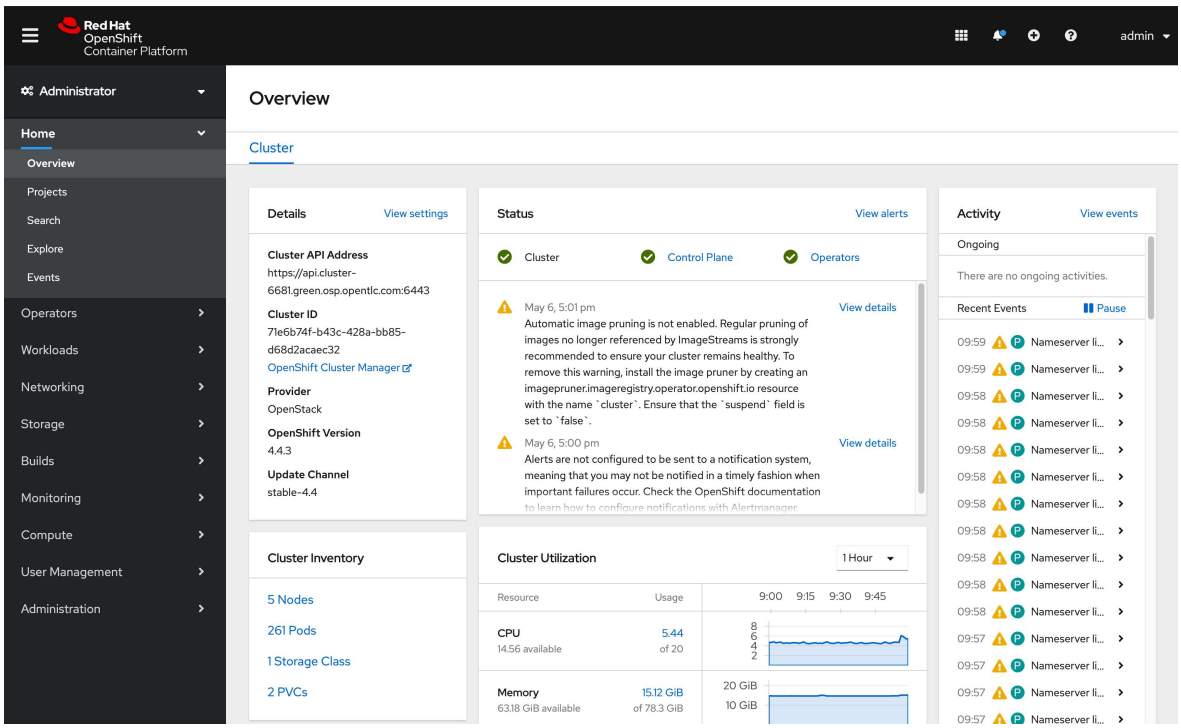


- 高级（更多）
- 项目详情
  - 状态、利用率、事件、配额
- 项目访问权限
  - 控制用户和组
- 指标



管理员视角

- 概述
  - 状态、活动/事件、清单、容量、利用率
- 每一种可管理的常见资源类型
- 日志和指标
- 可查看的高级设置：更新、Operator、CRD、角色绑定、资源配额



用户和项目

项目

- 让一组用户或开发人员能够协同工作
- 隔离和协作单元
- 定义资源作用域
- 允许项目管理员和协作者管理资源
- 通过配额和限值来限制和跟踪资源的使用
- 具有额外标注的 Kubernetes 命名空间
  - 管理常规用户的资源访问权限的中央载体
  - 让用户社区独立于其他社区来组织和管理内容
- 用户：
  - 从管理员接收项目的访问权限
  - 若被允许创建，则有权访问自己的项目
- 每个项目都有自己的：
  - **对象**：Pod、服务和复制控制器等
  - **策略**：指定哪些用户可以或不可以在对象上执行操作的规则
  - **约束**：可以被限制的对象的配额
  - **服务帐户**：自动执行操作且有权访问项目对象的用户

用户和用户类型

- 与 OpenShift® 的交互始终与用户关联
  - 通过为用户或群组添加角色来授予系统权限
- 用户类型：

常规用户	系统用户
代表大部分交互式 OpenShift 用户的方法	很多是在定义基础架构时自动创建的
首次登录时在系统中自动创建，或通过 API 创建	让基础架构与 API 安全地交互
通过 user 对象表示	包括：集群管理员、各节点的用户、服务帐户

登录和身份验证

- 每一用户必须通过身份验证才能访问 OpenShift
- 缺少有效身份验证的 API 请求被鉴定为来自匿名用户
- 策略决定用户被授权执行的操作

Web 控制台身份验证

- 通过管理员提供的 URL 访问 Web 控制台
- 提供登录凭据，获取进行 API 调用的令牌
- 使用 Web 控制台浏览项目

CLI 登录

- 使用 `oc` 工具登录与 Web 控制台相同的地址
- 提供登录凭据，获取进行 API 调用的令牌
  - `oc login -u <my-user-name> --server="<master-api-public-addr>:<master-public-port>"`
- 管理员可以让集群生成密钥以进行免密码身份验证

配额和限值

资源配额

- OpenShift 可以限制：
  - 项目中创建的对象数量
  - 项目的各个对象中请求的计算/内存/存储资源数量
  - 基于指定的标签
    - 示例：限制为开发人员部门或 `test` 等环境
- 多个团队可以共享一个 OpenShift 集群
  - 各个团队拥有自己的一个或多个项目
  - 资源配额防止团队互相争夺集群资源
- `ResourceQuota` 对象枚举"各个项目"的资源使用量硬限值
- `ClusterResourceQuota` 对象枚举集群中用户的资源使用量硬限值

LimitRange

- LimitRange 表达 pod 的容器 CPU 和内存要求
  - 设置特定 pod 的容器可以消耗的 CPU 与内存的 `请求` 与 `限制数量`
  - 协助 OpenShift 调度程序将 pod 分配到节点
- LimitRange 表达服务质量等级：
  - `Best Effort`
  - `Burstable`
  - `Guaranteed`
- 可以为每个项目设置所有 pod/容器的默认 `LimitRange`

非终端状态的 pod 中由配额管理的计算资源

资源名称	描述
`cpu` `requests.cpu`	CPU 请求数之和不能超过此值
`memory` `requests.memory`	内存请求数之和不能超过此值
`limits.cpu`	CPU 限值之和不能超过此值
`limits.memory`	内存限值之和不能超过此值

这些是所有非终端状态的 pod 中由配额管理的计算资源，以及它们的最大值。

提示：`cpu` 和 `requests.cpu` 的值相同，并且可以互换使用。同样的情况也适用于 `memory` 和 `requests.memory`。

配额管理的对象数

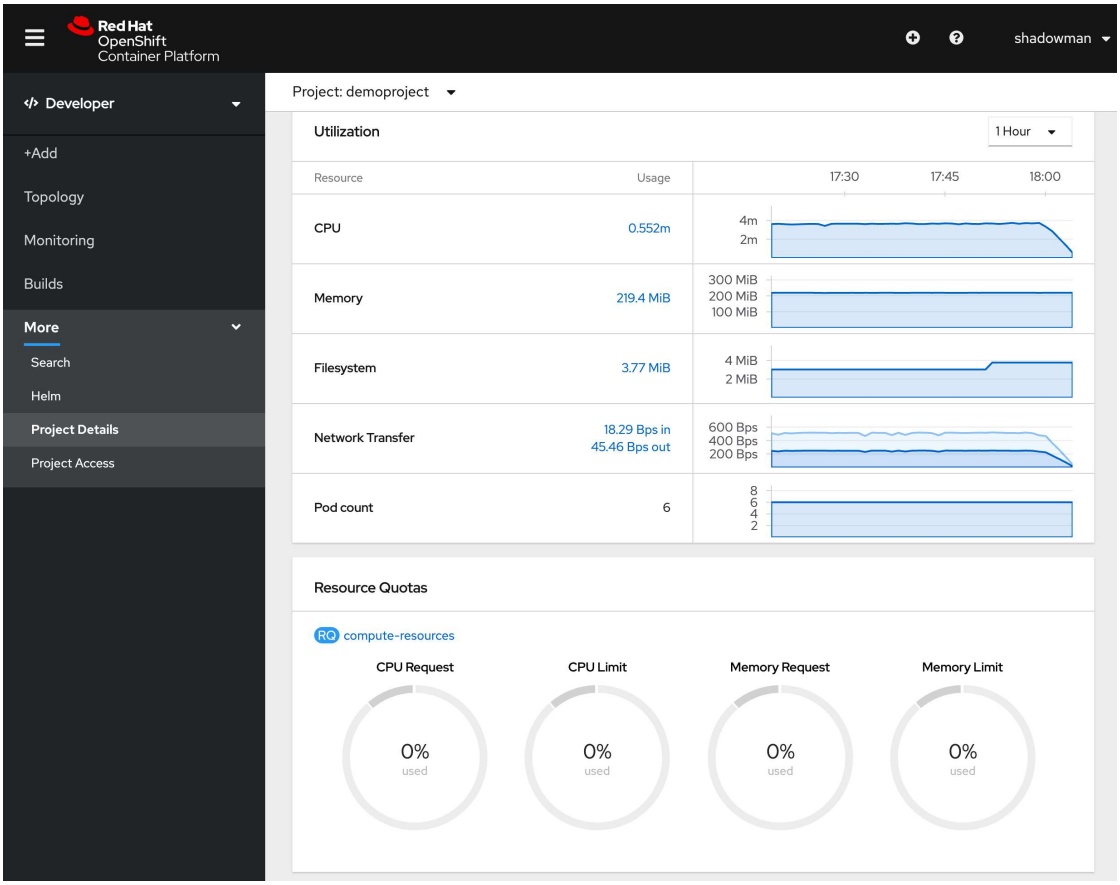
资源名称	描述
`pods`	项目中可以存在的非终端状态 pod 的总数。（若 `status.phase in (Failed, Succeeded)` 为 true，则 pod 处于终端状态）
`replicationcontrollers`	项目中可以存在的复制控制器的总数
`resourcequotas`	项目中可以存在的资源配额的总数
`services`	项目中可以存在的服务的总数
`secrets`	项目中可以存在的密钥总数
`configmaps`	项目中可以存在的 `ConfigMap` 对象的总数
`persistentvolumeclaims`	项目中可以存在的持久卷声明的总数
`openshift.io/imagestreams`	项目中可以存在的镜像流的总数

配额执行

- 在项目中创建配额后：
  - 项目限制创建可能违反配额约束的资源
  - 每隔几秒（可以配置）计算使用量统计
- 如果项目修改超过配额：
  - 服务器拒绝操作
  - 返回错误消息
- 错误消息包括：
  - 违反的配额约束
  - 当前系统使用量统计

查看配额

- 从 Web 控制台，选择：
  - `demoproject` 项目
  - **开发人员 → 更多 → 项目详情**
  - 通过滚动查看资源使用量、可用性
  - 基于 CPU、内存的请求数与限值
- 单击 `RQ compute-resources`：
  - 显示具体的资源类型配额、使用量报告
  - 显示合并的容器和 pod 请求数、限值



- 此外，也可使用命令行来查看配额详情：

  1. 示例：获取 `demoproject` 项目中定义的配额列表：

     ```
     $ oc get quota -n demoproject
     NAME                AGE
     besteffort          11m
     compute-resources   2m
     core-object-counts  29m
     ```

  2. 示例：描述 `demoproject` 项目中的 `core-object counts` 配额：

     ```
     $ oc describe quota core-object-counts -n demoproject
     Name:           core-object-counts
     Namespace:      demoproject
     Resource        Used    Hard
     --------        ----    ----
     configmaps      3   10
     persistentvolumeclaims  0   4
     replicationcontrollers  3   20
     secrets         9   10
     services        2   10
     ```

查看 `LimitRange`

- pod 和容器的 CPU 和内存要求
- 容器的默认值：

```
kind: LimitRange
apiVersion: v1
metadata:
  name: test-core-resource-limits
  namespace: test
spec:
  limits:
    - type: Container
      max:
        memory: 6Gi
      min:
        memory: 10Mi
      default:
        cpu: 500m
        memory: 1536Mi
      defaultRequest:
        cpu: 50m
        memory: 256Mi
    - type: Pod
      max:
        memory: 12Gi
      min:
        memory: 6Mi
```

日志和监控

容器日志聚合

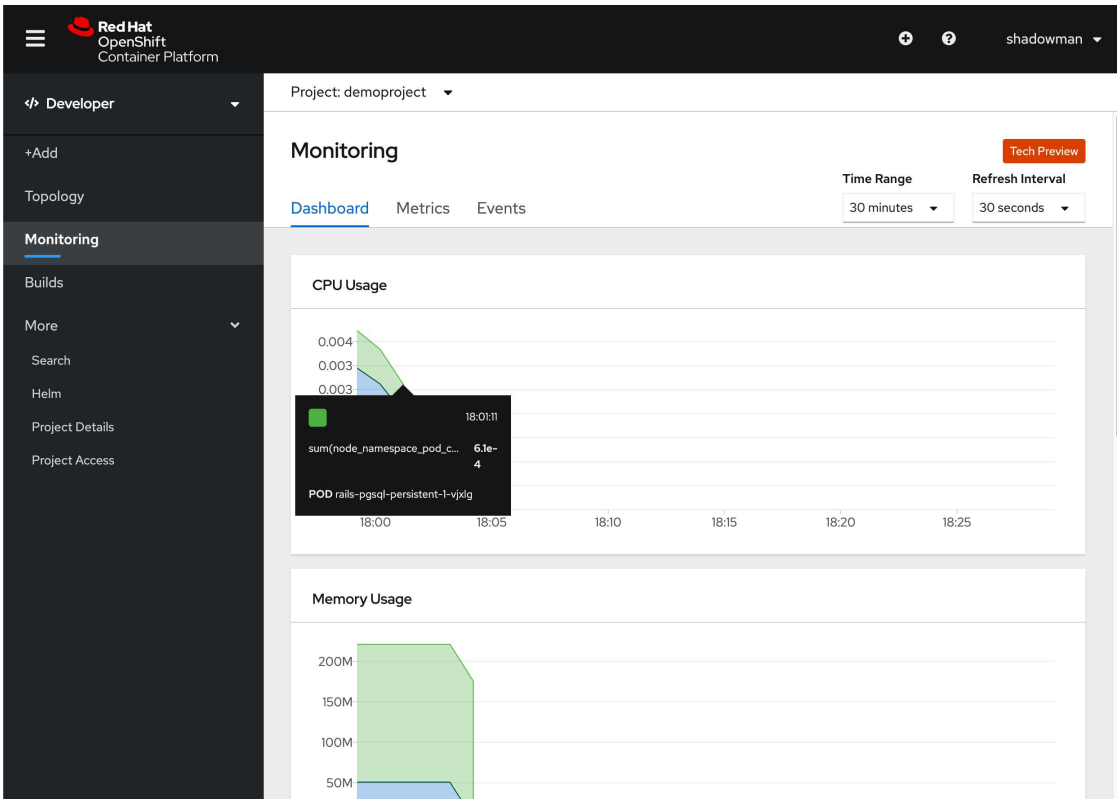
- 利用 EFK 堆栈，集群管理员可以聚合一系列 OpenShift 服务的日志
  - 能够提供相应途径让开发人员查看这些日志
- EFK 堆栈的修改版本 (ELK) 位于：
  - https://www.elastic.co/videos/introduction-to-the-elk-stack
- EFK 堆栈可用于查看从主机和应用聚合的日志
  - 可以来自多个容器，甚至是已删除的 pod
- EFK 日志堆栈由三个组件构成：
  - **Elasticsearch**：存储所有日志的对象存储
  - **Fluentd**：从节点收集日志，再馈入 Elasticsearch
  - **Kibana**：Elasticsearch 的 Web UI
- 部署了 EFK 后，堆栈将所有节点和项目的日志聚合到 Elasticsearch 中
  - 提供用于查看它们的 Kibana UI
- 集群管理员可以查看所有日志
- 开发人员只能查看他们拥有权限的项目的日志

Fluentd

- 从主机上的容器文件系统和 OpenShift 服务拉取日志
- 将它们发送到存储聚合日志数据的对应 Elasticsearch 集群
- 用户和平台管理员访问对应的 Kibana UI 来查看应用或平台的聚合日志

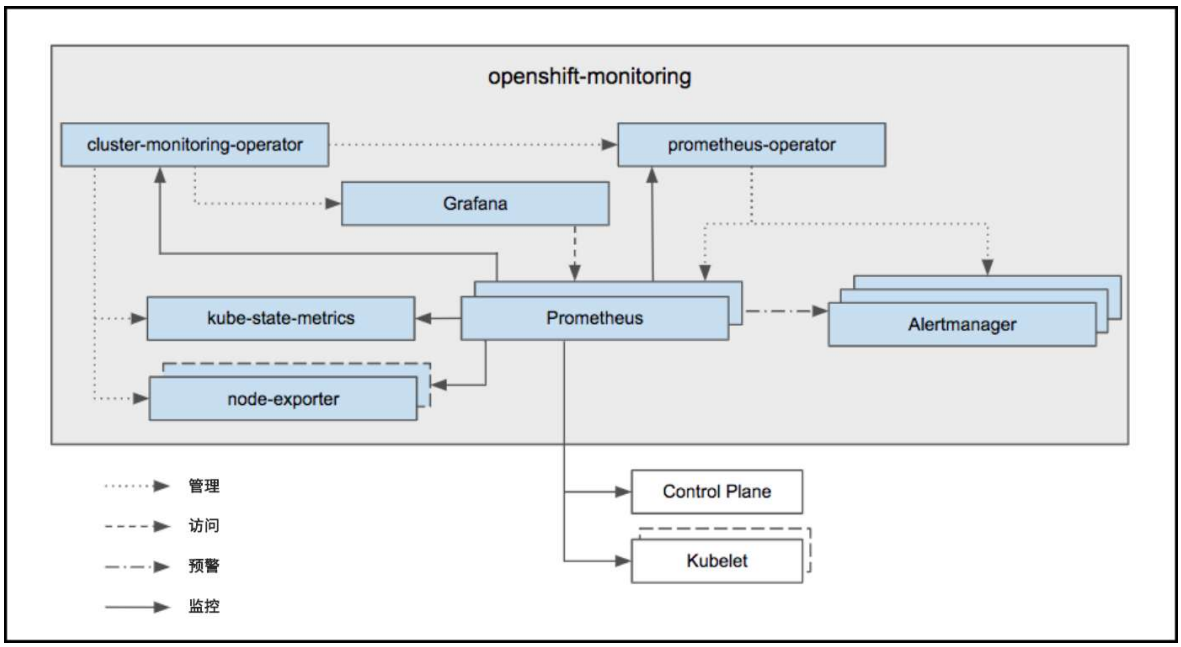
通过 Prometheus 进行指标收集与预警 - 功能

- 基于 Prometheus 开源项目预先配置并自行更新的监控堆栈
- 提供集群组件监控功能
- 附带一系列预警和仪表板
  - 也可安装和使用 Grafana 仪表板
- OpenShift 管理员可通过一个 UI 查看集群中所有容器、组件的指标
- 横向 pod 自动扩展器 (HPA) 使用指标决定扩展的时间和方式
  - 基于 CPU 和内存的指标可从 OpenShift 容器平台 Web 控制台查看



通过 Prometheus 进行指标收集与警报 - 设计

- Cluster Monitoring Operator (CMO)
  - 监视部署的监控组件、资源；保持最新状态
- Prometheus Operator (PO)
  - 管理 Prometheus 和 Alertmanager
  - 根据 Kubernetes 标签查询自动生成监控目标
  - Alertmanager 处理客户端预警并转发到电子邮件、PagerDuty 等。
- `node-exporter`：收集指标的代理
- `kube-state-metrics`：将对象转换为指标



通过 Prometheus 进行指标收集与警报 - HPA

- 基于 Prometheus 中的任何指标配置横向 pod 自动扩展 (HPA)
  - 根据 OpenShift 中的任何集群级指标进行自动扩展
  - 根据任何应用指标进行自动扩展
- 自动扩展 pod 和计算机
  - Prometheus Adapter 连接单个 Prometheus 实例（或通过 Kubernetes 服务）
  - 手动部署和配置适配器
  - 集群管理员需要将用于 HPA 的集群指标加入白名单

模板、Operator 和 Helm 3

概述

- **模板**:
  - 只能创建资源
  - 不能管理或删除资源
  - 不与 pod 关联
- **Operator**:
  - 管理、管理和删除资源
  - 由 Operator pod 实施
- **Helm 3**:
  - 模板包
  - 如何打包应用
  - 如何安装软件包

	Helm 图表	Operator
打包	✓	✓
应用安装	✓	✓
应用更新（Kubernetes 清单）	✓	✓
应用升级（数据迁移、调整，等等）	-	✓
备份和恢复	-	✓
工作负载和日志分析	-	✓
智能扩展	-	✓
自动调优	-	✓

模板是什么?

- 描述可以参数化和处理的一系列对象，以生成要由 OpenShift 创建的对象的列表
- 通过处理模板，创建您在项目内有权创建的所有内容
  - 示例：服务、构建配置和部署配置
- 也可定义一系列标签，以应用到模板中定义的每个对象

模板有什么用途?

- 为开发人员或客户创建可即时部署的应用
- 提供使用预设变量或随机生成值（如密码）的选项

模板中的标签

- 用于管理生成的资源
- 应用到从模板生成的“每一个”资源
- 组织、分组或选择对象和资源
- 资源和 pod 被“标上”标签
- 标签让服务和复制控制器能够：
  - 确定与它们相关的 pod
  - 引用 pod 组
  - 将具有不同容器的 pod 作为相似实体对待

模板中的参数

- 在模板中不同对象之间共享配置值
- 值可以是静态值，或者由模板生成
- 您可利用模板定义要取值的参数
  - 在引用参数的任何位置上替换值
  - 可以在对象定义中的任何文本字段上定义引用
- 示例：
  - 将 `generate` 设为 `expression` 以指定生成的值
  - `from` 利用伪正则表达式语法指定生成值的模式

```
parameters:
  - name: PASSWORD
    description: "The random user password"
    generate: expression
    from: "[a-zA-Z0-9]{12}"
```

从现有对象创建模板

- 可以模板形式导出项目中的现有对象
- 通过添加参数和自定义修改导出的模板
- 以模板形式导出项目中的对象：

```
$ oc export all --as-template=<template_name>
```

Operator 中心

- 通过 Operator 部署和管理集群中的应用
- 探索来自 Kubernetes 社区和红帽® 合作伙伴的 Operator
- 将 Operator 安装到集群中，为开发人员提供可选的附件和共享服务
- Operator 提供的功能显示在开发人员目录中，带来自助服务体验
- 将共享的应用、服务或 source-to-image 构建器添加到您的项目中
- 集群管理员可以安装其他应用，它们会自动显示出来

Helm 图表和 Operator

- Helm 图表与模板相似
  - 描述一组相关 Kubernetes 资源的文件集合
- 从诸多可用 Helm 图表中进行选择
- 将图表与 Helm operator 搭配使用来轻松部署应用

Operator 与 OpenShift 的交互

- Operator 是利用自定义资源定义 (CRD) 的 pod
- CRD 支持扩展 Kubernetes/OpenShift API
  - API 随后会知道新的资源
- CRD 允许创建自定义资源 (CR)
- Operator 监控 CR 的创建，并通过创建应用来回应

- CR 的管理方式与库存 OpenShift 对象相同
  - `create`、`get`、`describe` 或 `delete` 等。
  - 示例：

```
oc get tomcats
```

自定义资源定义

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: tomcats.apache.org
spec:
  group: apache.org
  names:
    kind: Tomcat
    listKind: TomcatList
    plural: tomcats
    singular: tomcat
    shortNames:
    - tc
  scope: Namespaced
  version: v1alpha1
```

Operator - 自定义资源（应用）创建

- 自定义资源是您希望 Operator 创建的资源的简单定义
- 运行中的 Operator 会监控将在以下任何位置创建的 CR：
  - 整个 OpenShift 集群
  - 正在运行 Operator 的项目
- 创建 CR 时，operator 会收到事件
- Operator 随后会创建组成应用的所有 OpenShift 资源

示例：自定义资源创建

- 创建 CR 实例：
  - 创建含有 CR 定义的 YAML 文件：

```
apiVersion: apache.org/v1alpha1
kind: Tomcat
metadata:
  name: mytomcat
spec:
  replicaCount: 2
```

  - 在 OpenShift 中创建资源：

```
oc create -f mytomcat.yaml
```

Operator - 自定义资源管理

- 若要操控并检查 CR，可使用 `oc` 命令：

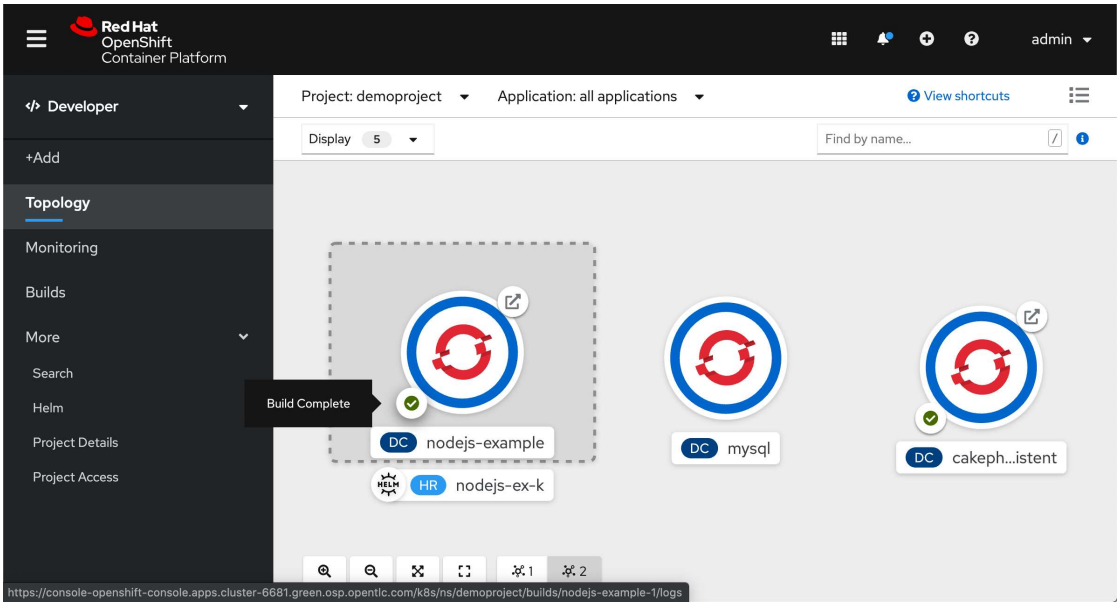
```
oc get tomcats
oc describe tomcat mytomcat
oc scale tomcats --replicas=2 # only if Operator supports scaling
```

- 不要直接扩展 ReplicaSet、Deployment 或 StatefulSet
  - 使用 Operator 来扩展
  - Operator 会持续监控创建的资源，并将它们重新设置为初始状态
- 若要删除所有创建的 OpenShift API 对象，可删除 CR：

```
oc delete tomcat mytomcat
```

Helm 图表

- 安装 Helm 图表时：
  - 值文件中的值会在图表模板中被替换
  - 生成发行版本，作为 Kubernetes 集群中运行的图表实例
- 多次安装同一个图表可创建多个发行版本
- Helm v3
  - 无服务器端组件 (Tiller)
  - Helm CLI 与 Kubernetes API 交互



总结

- Web 控制台
- 用户和项目
- 配额和限值
- 日志和监控
- 模板、Operator 和 Helm 3