# 赛题 57 #

# 设计并实现一个 operator 管理框架

直播导师：蔡灏旻

## 什么是Operator?

- Operator 是 CoreOS 在2016年推出的简化复杂有状态应用管理的框架
- Operator 基于 CRD （Custom Resource Definition）扩展资源对象，并通过自研控制器来保证应用处于预期状态

## Operator相关技术对比

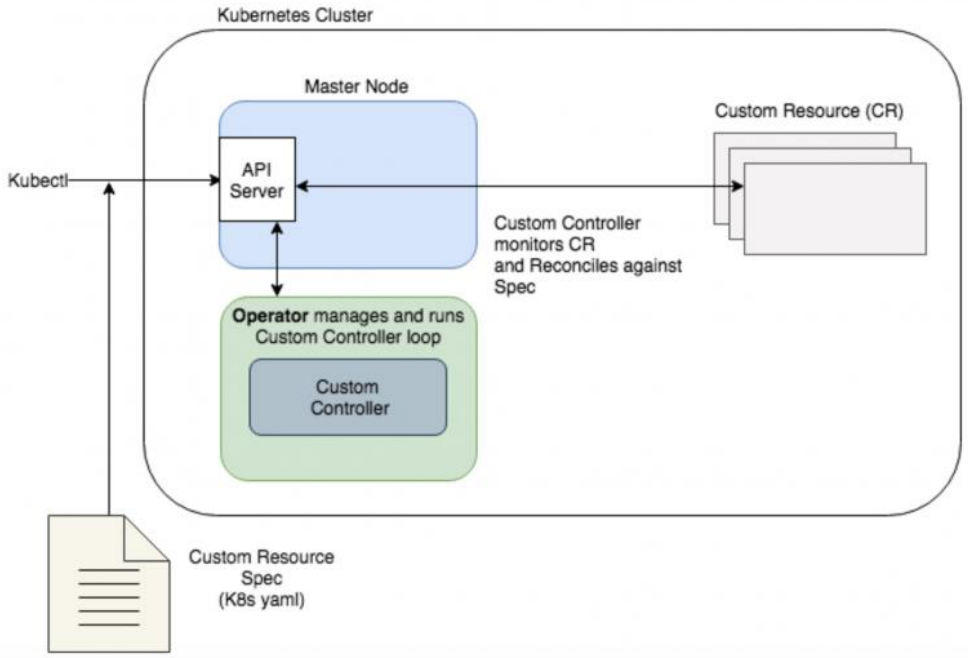
- docker vs operator
  - docker最具颠覆性的价值在于容器镜像，它提供了单应用的标准化交付方式。
  - operator与docker相似，但是它主要的交付对象从单应用实例，扩展到多实例、分布式系统上。
- Helm vs operator
  - Helm类似于Yum包管理工具，它管理的是k8s不同应用部署所依赖的API对象声明文件（静态）；
  - operator对资源的管理不仅是创建和交付，还可以watch资源的变化事件，根据资源状态实现高可用、可扩展、故障恢复等运维操作（动态）。
  - Helm主要通过命令行驱动整个生命周期。
  - operator主要通过资源的状态来驱动整个生命周期。
- controller vs operator
  - k8s架构中的controller manager集合管理的主要是k8s中基础的、通用的资源，例如Replicas，Deployment
  - operator则是通过CRD可以扩展管理特定的应用和服务资源（mysql cluster可以认为是一种资源）

## Operator带来了什么

- operator提供了一种框架，可以将运维经验（高可用、扩缩容、故障恢复、灾备、系统升级等）沉淀为代码，实现运维的代码化、自动化、智能化

## Key feature

- operator SDK：使开发人员能够利用其专业知识来构建 Operator，无需了解 Kubernetes API 的复杂性。
- Operator 生命周期管理：监控在 Kubernetes 集群上运行的所有 Operator 的生命周期的安装、更新和管理。
- Operator 计量：为提供专业服务的 Operator 使用情况报告。




operator工作原理

**课题的要求：**

首先我们先定一个基调：这是一个比较开放的课题，我并不希望我完全设计完整整个课题架构

顶层设计：我们希望拥有一个operator相关的管理框架，类似于OLM的一个东西，但它需要有以下几个特点：

1. 对于operator良好的生命周期管理设计，包括上线、发布、版本管理、安装、部署等，这对于一个管理框架来说是最重要的，所以我列在第一个
2. 可扩展性：希望该管理框架可以对接包括operator hub或者以后可能会存在的多个公有云的operator仓库
3. 轻量化，作为一个管理面组件我们并不希望因此产生过多的资源消耗
4. 用户的易用性CLI是必须的，UI如果有能力的话可以挑战一下

额外的一些不算要求的要求：

1. 不局限于go语言，可以尝试尝试rust？hah，原谅我最近痴迷于rust
2. 我更希望你能先从当前OLM存在一些什么问题出发来完整看看这个课题，我的顶层设计并不是你的顶层设计，或许我们后面可以多交流交流，碰撞看看
3. 如果可以，请先提交一份比较合理的设计文档，当然我知道coding总是最开心的阶段，但是还是希望有一个良好的设计再开始动手

https://github.com/operator-framework/operator-lifecycle-manager