基于 Kubernetes 构建标准可扩展的云原生应用管理平台

孙健波/周正喜

阿里云 — 云原生应用平台团队

云原生社区简介

- 成立于 2020 年 5 月 12 日
- 定位：企业中立的云原生终端用户社区
- 使命：推广云原生技术，构建开发者生态
- 官网：https://cloudnative.to

社区发起人
宋净超（Jimmy Song)

云原生社区公众号

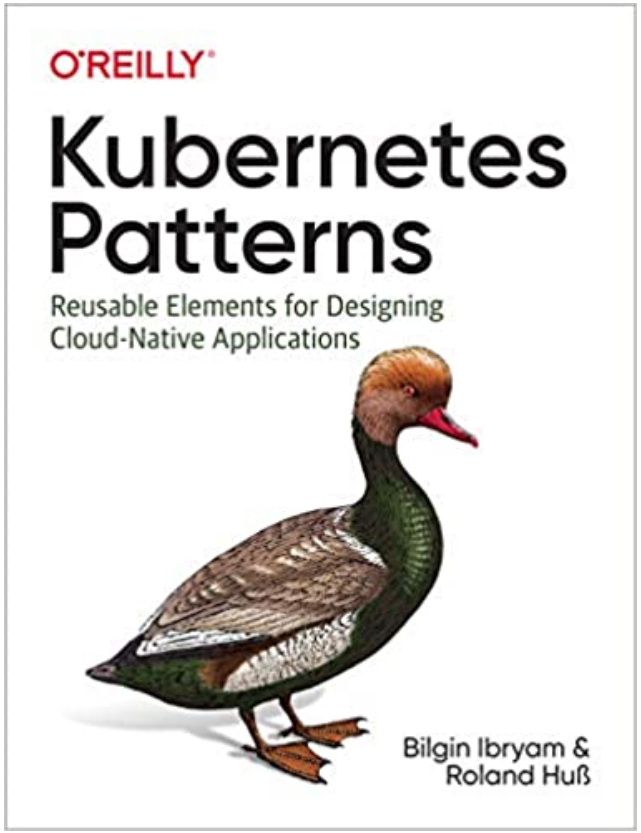
社区现状

- 成员：3500+
- 云原生学院线上分享：6 期
- SIG：Kubernetes、Istio、Envoy、Dapr、**OAM**
- 城市站：18 个

云原生社区城市站分布图

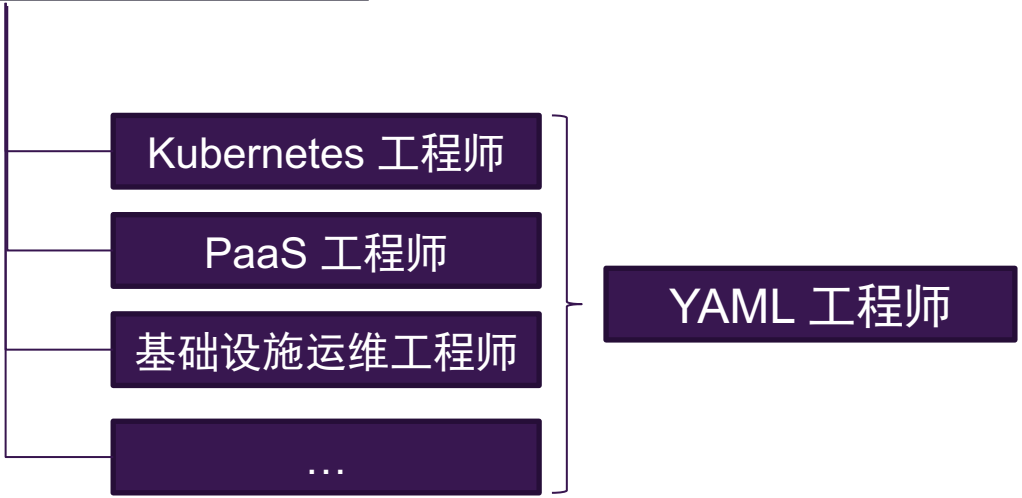
城市站详情见：https://cloudnative.to/city/

有奖品？


O'REILLY
Kubernetes Patterns
Reusable Elements for Designing Cloud-Native Applications
Bilgin Ibryam & Roland Huß

我的工作内容？

- 构建云原生应用管理平台 @ 阿里巴巴



我们是如何构建的？

基于 Kubernetes 我们构建了多种多样的应用管理平台：

用户（应用开发者和运维人员）

PaaS | Serverless | Operator Platform | 电商 PaaS

Kubernetes

我所在的团队

为什么 我们需要在 **Kubernetes** 上构建这些平台呢？

Kubernests 官方说：

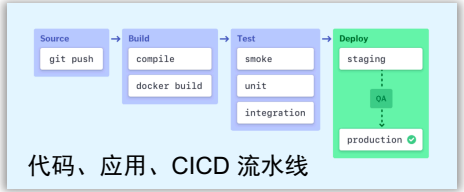
> “The metadata is organized around the concept of an *application*. **Kubernetes is not a platform as a service (PaaS) and doesn‘t have or enforce a formal notion of an application.** Instead, applications are informal and described with metadata. The definition of what an application contains is loose.”

Sited: https://kubernetes.io/docs/concepts/overview/working-with-objects/common-labels/

为了更好的用户体验：

用户期望：

API 和业务原语 关注点不同



服务语义与抽象程度不同

扩容策略
- 当请求数上升 10% 时，自动扩容 100 个实例

发布策略
- 当金丝雀实例通过 99% 的测试时，按每小时切 10% 流量的节奏进行发布

交互与使用习惯不同

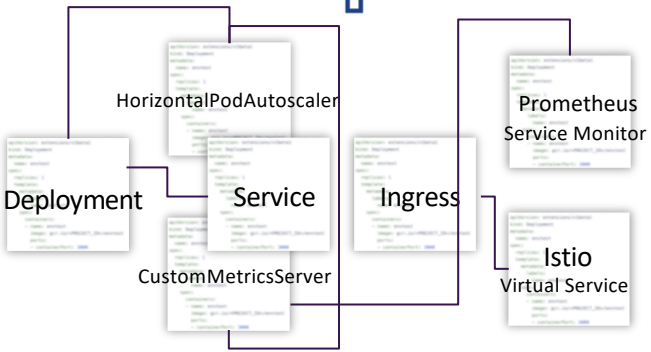

图形化界面


命令行工具


IaC 配置语言

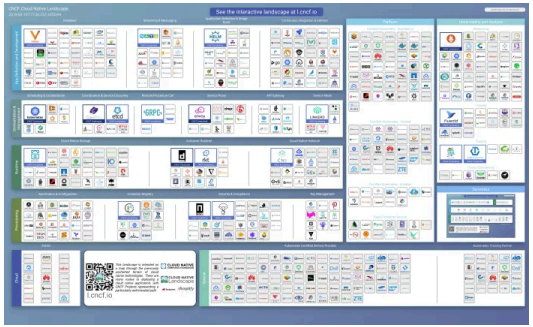
K8s 提供：

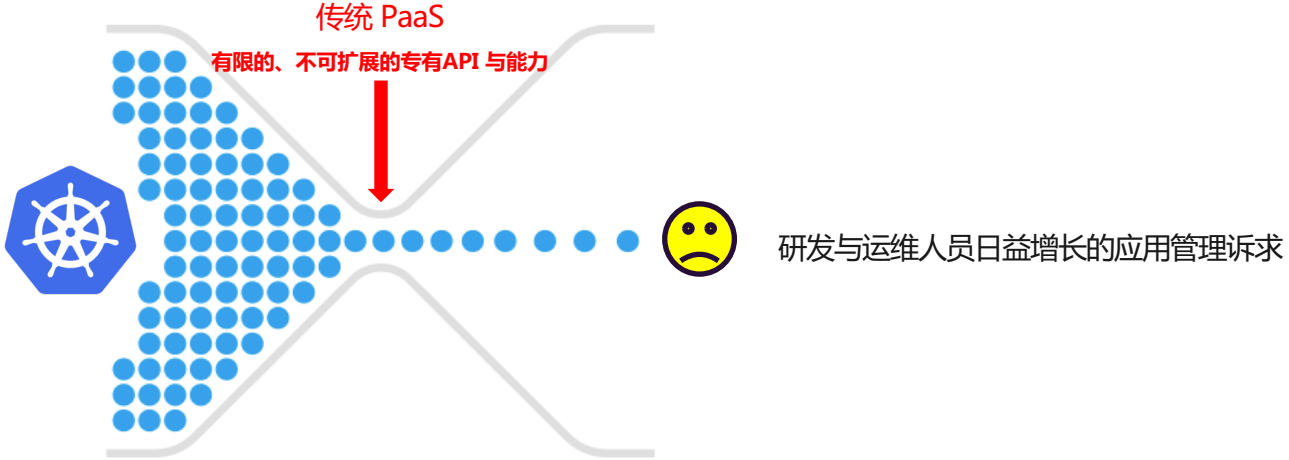
容器	Pod
调度	Controller
Sidecar	Node
CNI	CSI



不停构建“PaaS”平台不是“银弹”

K8s 生态“无限”的应用基础设施能力





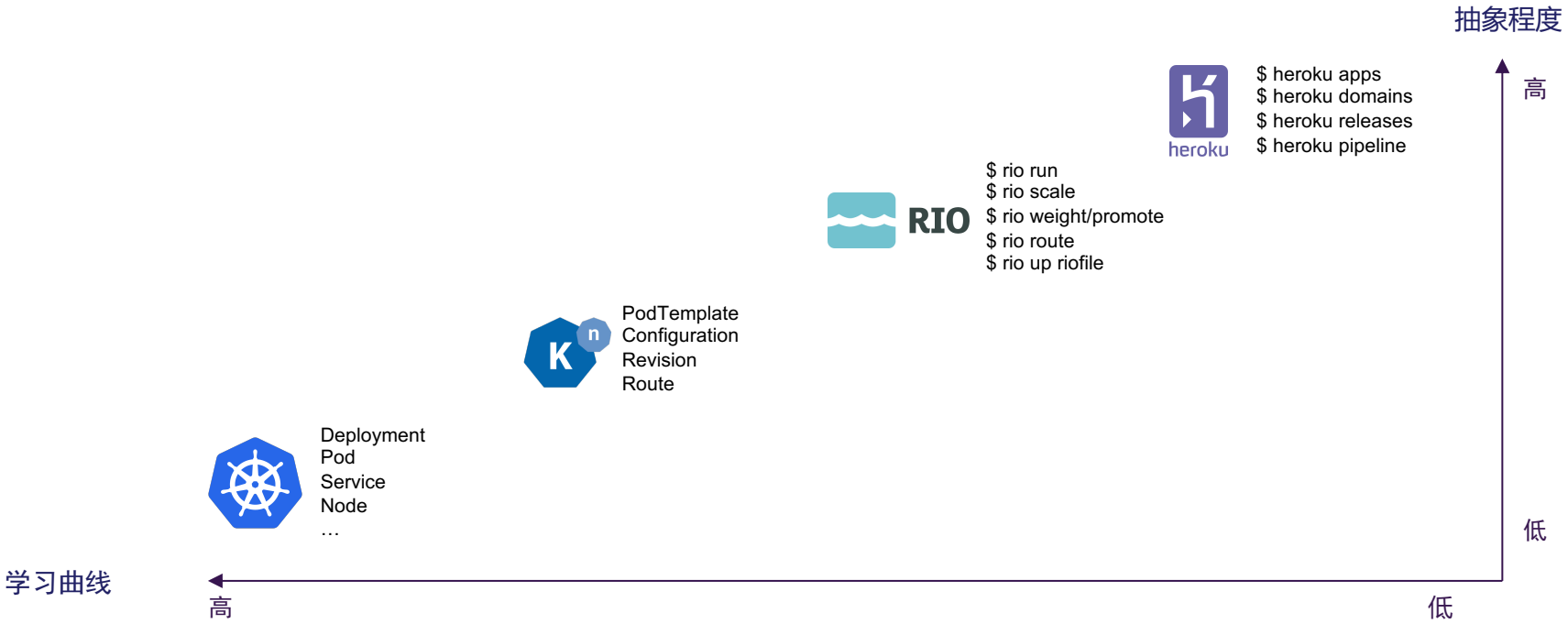
与其 基于 K8s 构建平台

不如 把 K8s 变成面向开发者的平台

构建一个具备“以应用为中心的 API 抽象”、“用户友好”
且“高度可扩展”的 K8s！

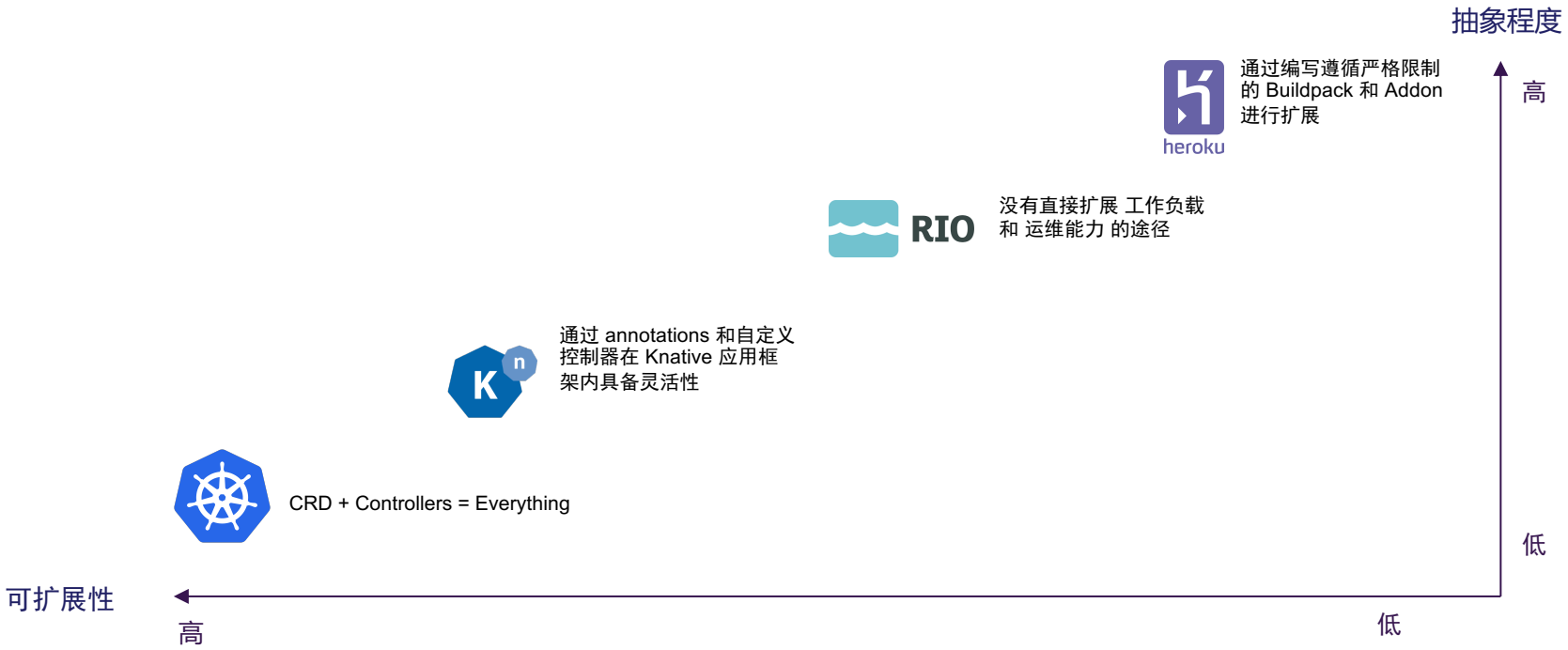
以应用为中心的 API 抽象

- 应用的工作负载和运维能力的抽象程度越高，用户体验越好

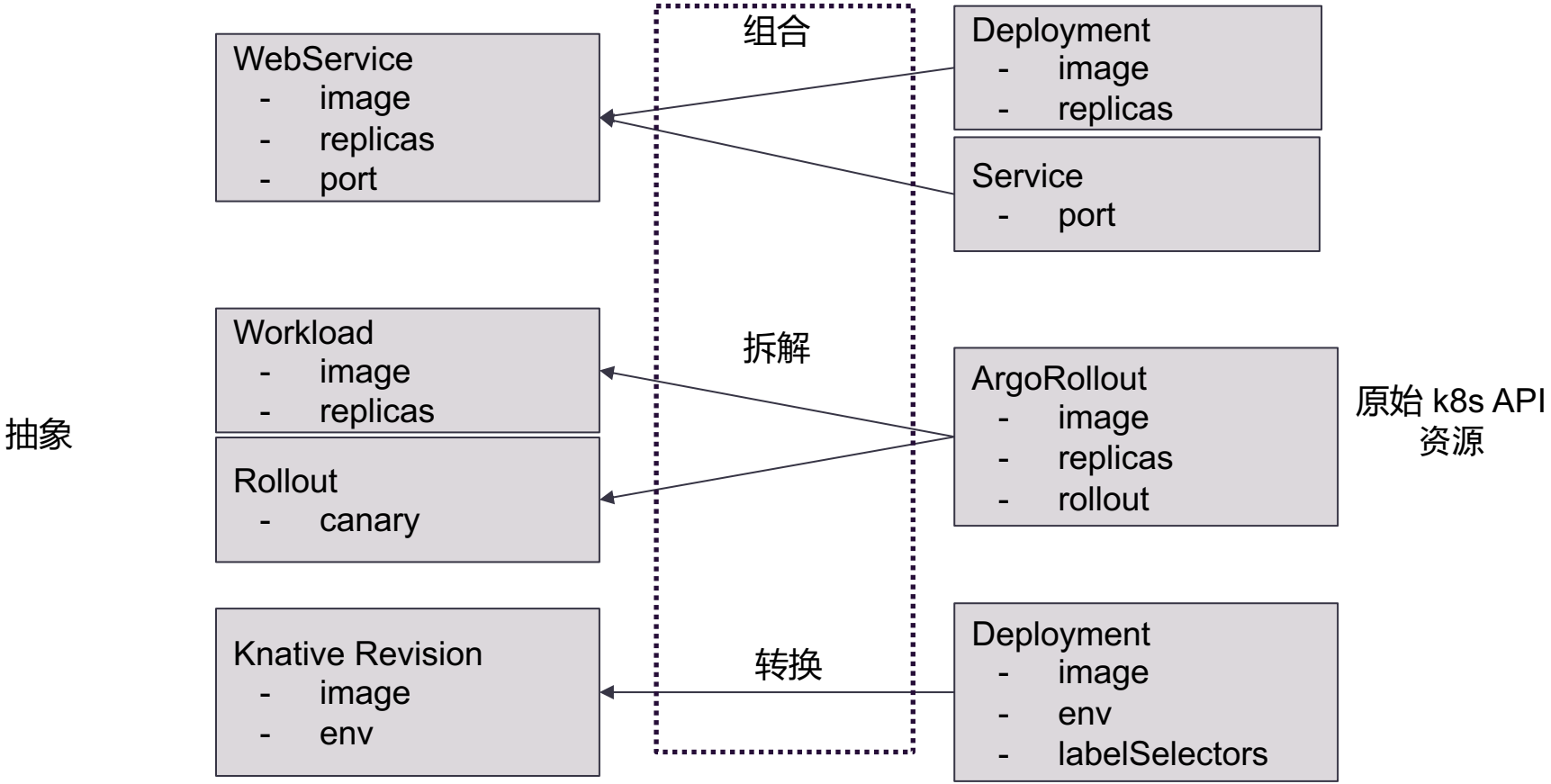


抽象程度 vs 可扩展性

- 随着抽象程度的增高可以显著降低学习曲线，但是却不得不在扩展性上妥协



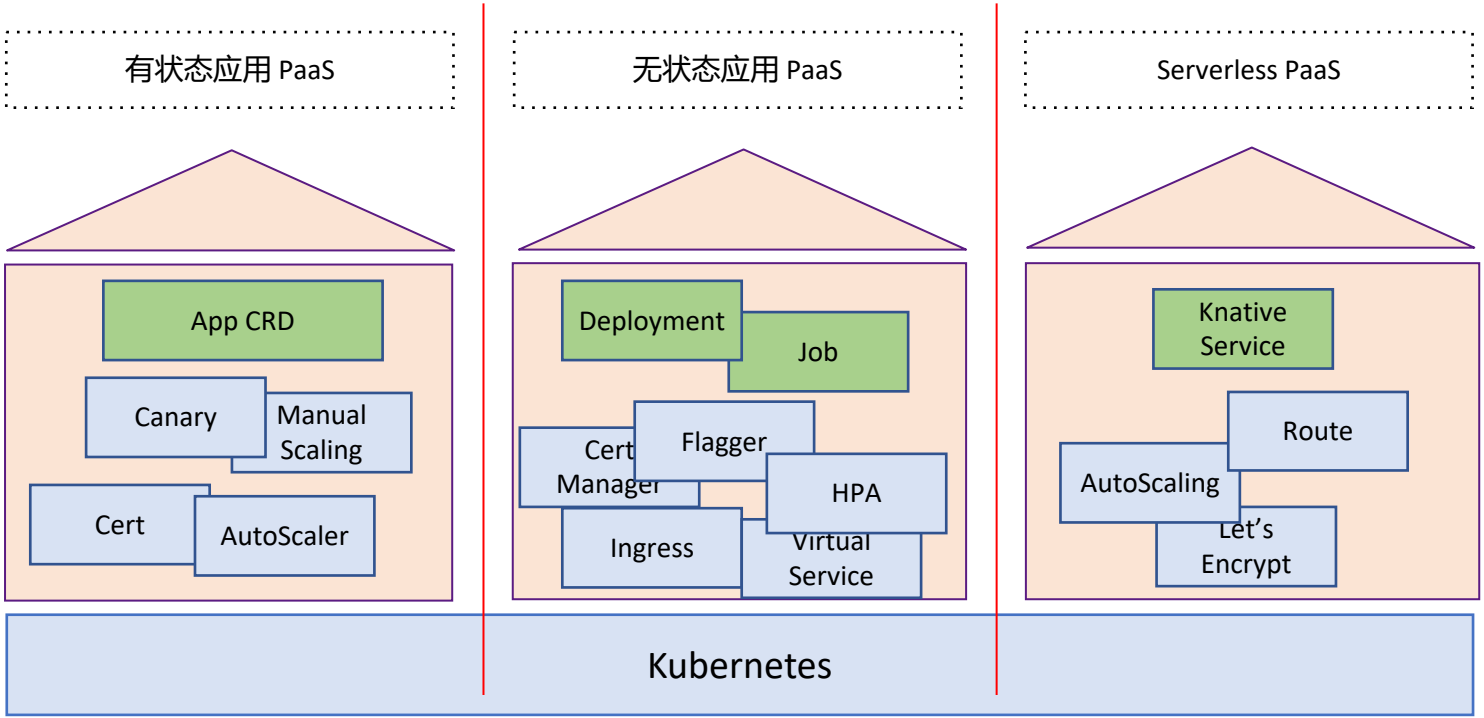
在服务端做抽象并不简单



做抽象容易形成“谷仓”

- 一个抽象满足不了所有场景，所以…

缺乏交互、复用、可移植能力。不同重复造轮子只是适配不同 API



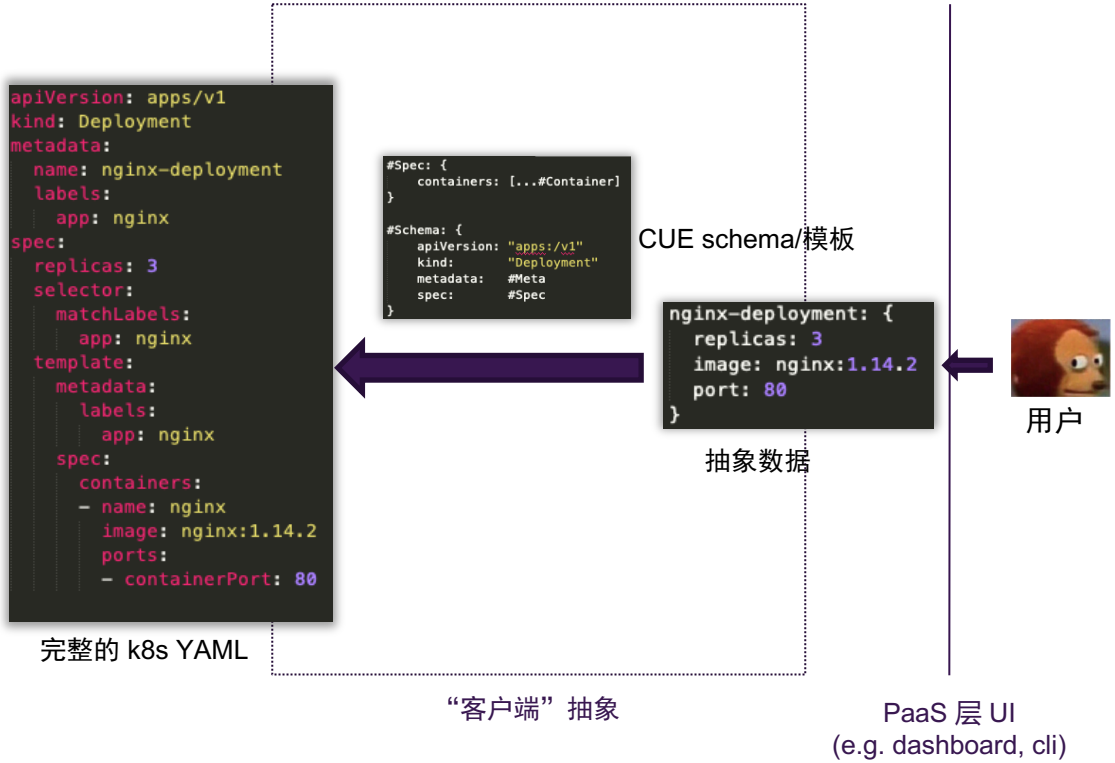
如何基于 K8s ，构建出一个**既用户友好，又高可扩展，还统一、标准化**的应用管理平台？

简单的“客户端”抽象： DCL (Data Configuration Language)

对 K8s 资源进行抽象实际上就是在操纵 YAML 数据，通过 DCL 来完成相比于 CRD + controller 更简单

CUE

- **功能强大：专注于操纵数据，而不是写代码**
- 完全兼容 JSON
- **简单直观：schema 和 value 语法一致**



标准化的“服务端”抽象 – 应用模型

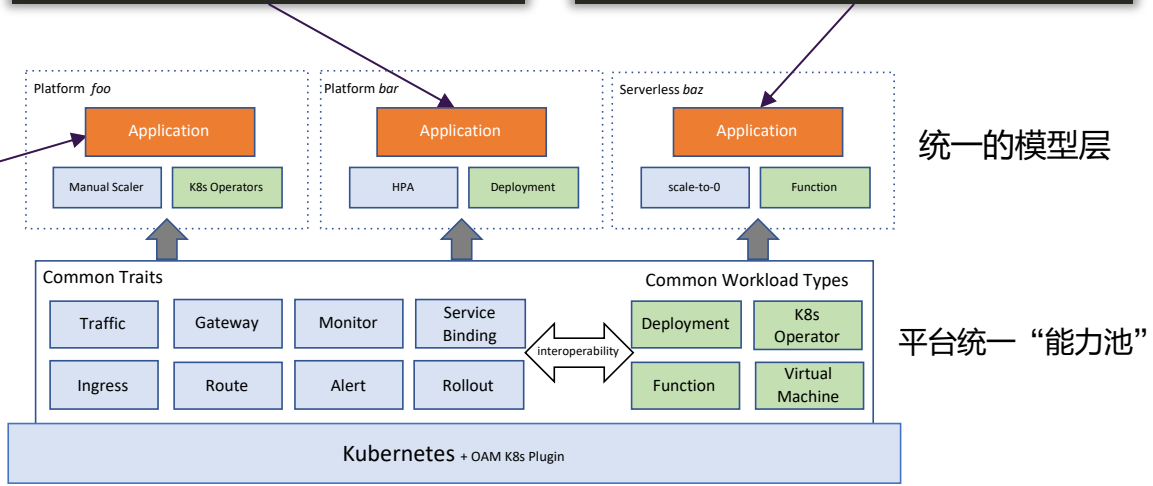
Open Application Model (OAM)

- 通过 OAM spec 定义“以应用为中心”的原语
- **打破“谷仓”!**

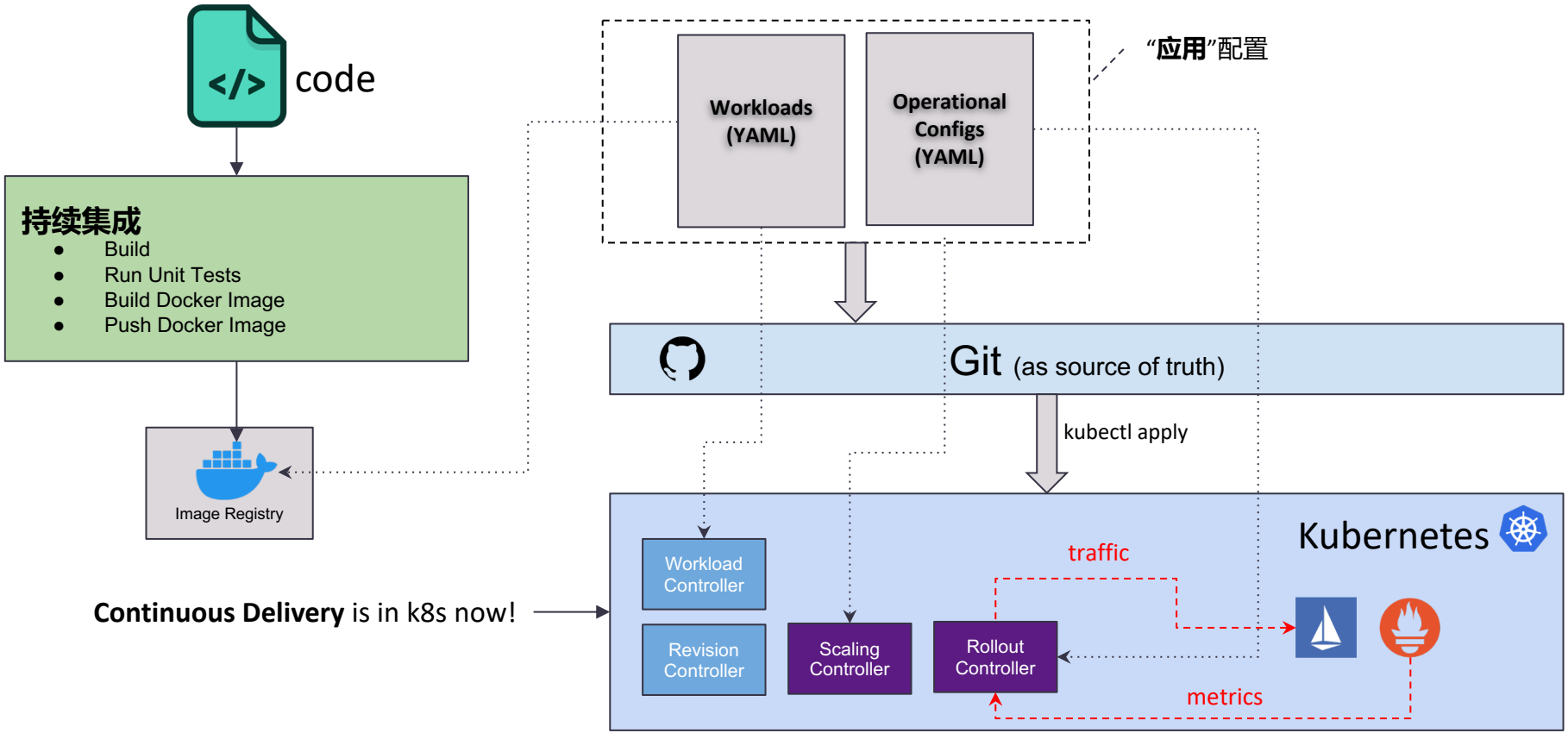
```yaml
apiVersion: core.oam.dev/v1alpha2
kind: ApplicationConfiguration
metadata:
  name: stateful-app
spec:
  components:
    - componentName: nginx-deployment
    - componentName: postgres-operator
      traits:
        - trait:
            apiVersion: core.oam.dev/v1alpha2
            kind: ManualScaler
            spec:
              replicaCount: 2
```

```yaml
apiVersion: core.oam.dev/v1alpha2
kind: ApplicationConfiguration
metadata:
  name: stateless-app
spec:
  components:
    - componentName: nginx-deployment
      traits:
        - trait:
            apiVersion: autoscaling/v2beta2
            kind: HorizontalPodAutoscaler
            spec:
              minReplicas: 1
              maxReplicas: 10
```

```yaml
apiVersion: core.oam.dev/v1alpha2
kind: ApplicationConfiguration
metadata:
  name: serverless-app
spec:
  components:
    - componentName: knative-svc
      traits:
        - trait:
            apiVersion: autoscaling.knative.dev/v1
            kind: Autoscaler
            spec:
              minScale: 0
              maxScale: 100
```



模块化的交付系统 - GitOps



三者结合呢？

- 基于 CUE 的客户端抽象
- 基于 OAM 的应用模型
- 围绕 GitOps 的持续交付

= “以应用为中心”的 K8s

KubeVela

```
name: myapp
components:
  frontend:
    deployment:
      image: inanimate/echo-server
      env:
        PORT: 8080
    traits:
      autoscaling:
        max: 10
        min: 1
      rollout:
        strategy: canary
        step: 5
      expose:
        service:
          type: LoadBalancer
        ports:
          http:
            service_port: 80
            container_port: 8080
```

面向应用开发者的 appfile

- 基于 CUE 进行抽象
- 兼容 OAM Spec

code

持续集成

- Build
- Run Unit Tests
- Build Docker Image
- Push Docker Image

Image Registry

Git (as source of truth)

GitOps

持续交付

OAM K8s Plugin + CUE Abstraction Processor

Kubernetes

Raw k8s API resources

Deployment Controller

AutoScaling Controller

Rollout Controller

traffic

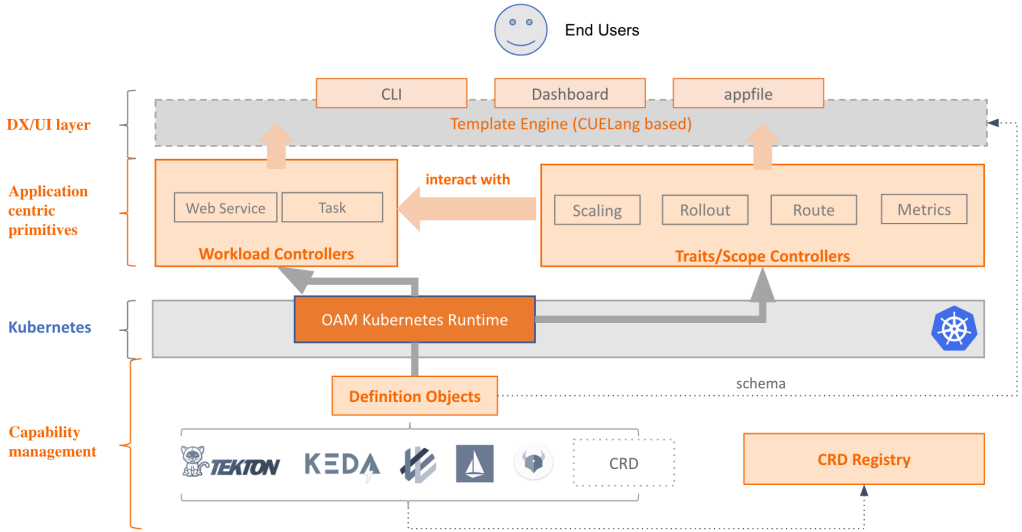
metrics

KubeVela

“The Extensible Application Platform Based on Kubernetes and Open Application Model (OAM)”

KubeVela = OAM Kubernetes Runtime + Capability Center + UI (Cli + Dashboard)

KubeVela



- **User interface layer**
  - CLI/Dashboard/Appfile

- **KubeVela core**
  - **OAM Kubernetes Runtime**
    to provide application level building blocks such as Component and Application etc.
  - **Built-in workload and trait controllers**
    to implement core capabilities such as webservice, route and rollout etc.
  - **Capability Management**

一个**既用户友好，又高可扩展，标准化**的应用管理引擎即将发布，敬请期待！

https://github.com/oam-dev/kubevela

云原生应用管理 千人钉钉交流群

参考资料

OAM spec: https://oam.dev/
KubeVela: https://github.com/oam-dev/kubevela
OAM 系列文章：
https://mp.weixin.qq.com/mp/appmsgalbum?biz=MzUzNzYxNjAzMg==&action=getalbum&album_id=1416643008195608577&scene=173&from_msgid=2247491488&from_itemidx=1&count=3#wechat_redirect
OAM 社区会议记录：
https://space.bilibili.com/180074935/channel/detail?cid=138178

KubeVela demo

- KubeVela Cli 整体能力介绍
  - Getting started/Application/Traits/System/Capability

- 应用创建
  - by OAM: https://github.com/zzxwill/try-cloudnative/tree/master/cloudnativeto-presentation-20201029/oam
  - by KubeVela: https://github.com/zzxwill/try-cloudnative/tree/master/cloudnativeto-presentation-20201029/kubevela

- 应用运维
  - Route
  - Scale

- Capability management
  https://github.com/zzxwill/try-cloudnative/tree/master/capabilities

- 集成
  Dashboard/Cli → OpenAPI

What’s more...

- KubeVela 为什么选择 CUE 作为 DCL?
- KubeVela 设计原则
- Killer feature – Appfile
- …