阿里巴巴新一代基于 Go 的云原生应用引擎实践

周正喜

OAM/KubeVela maintainer

周正喜

阿里云云原生团队 技术专家@北京

OAM/KubeVela maintainer

zhengxi.zzx@alibaba-inc.com

OAM/KubeVela 技术疑问 & 招聘

欢迎 Gopher 来聊！

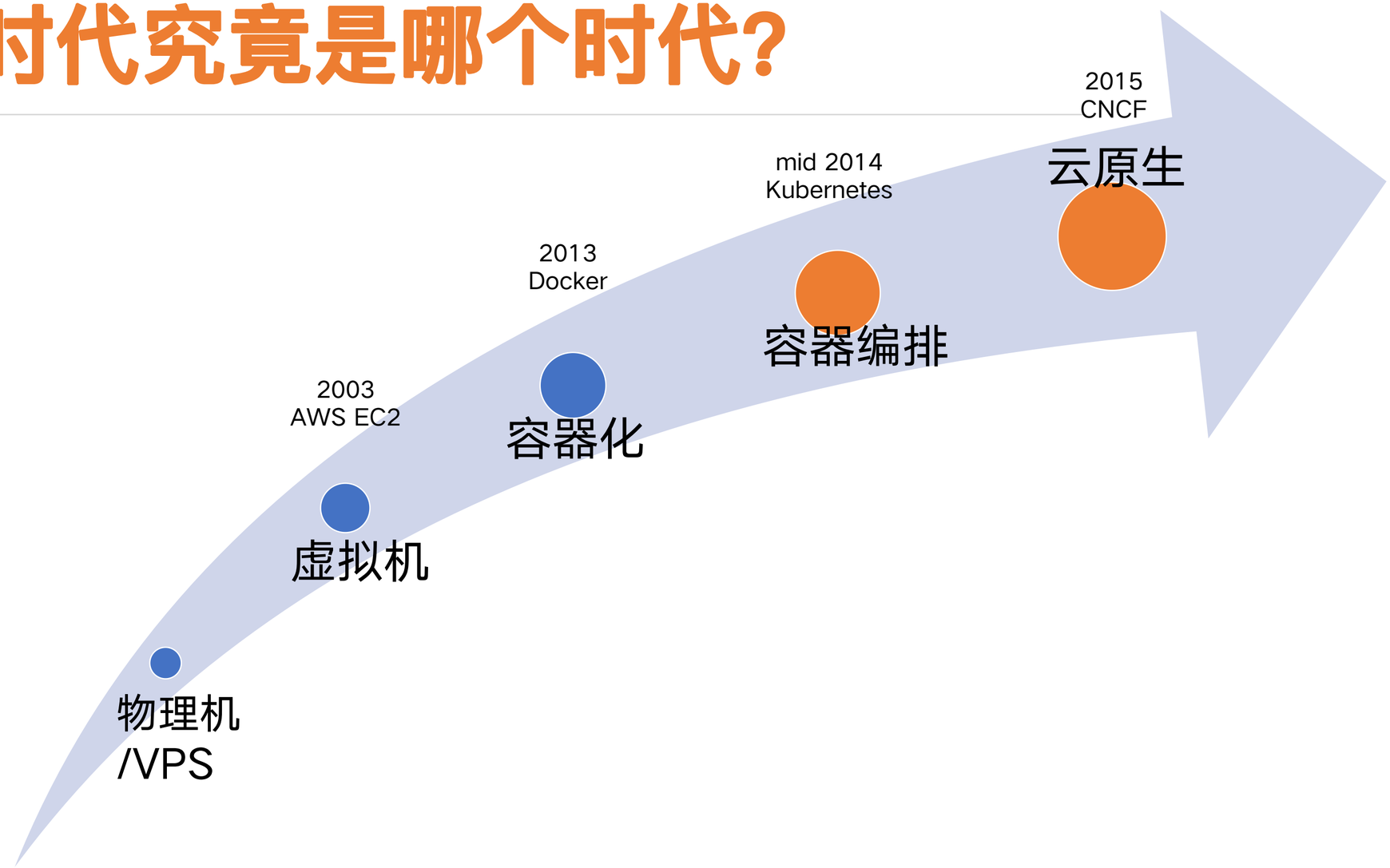
GopherChina 2021

目 录

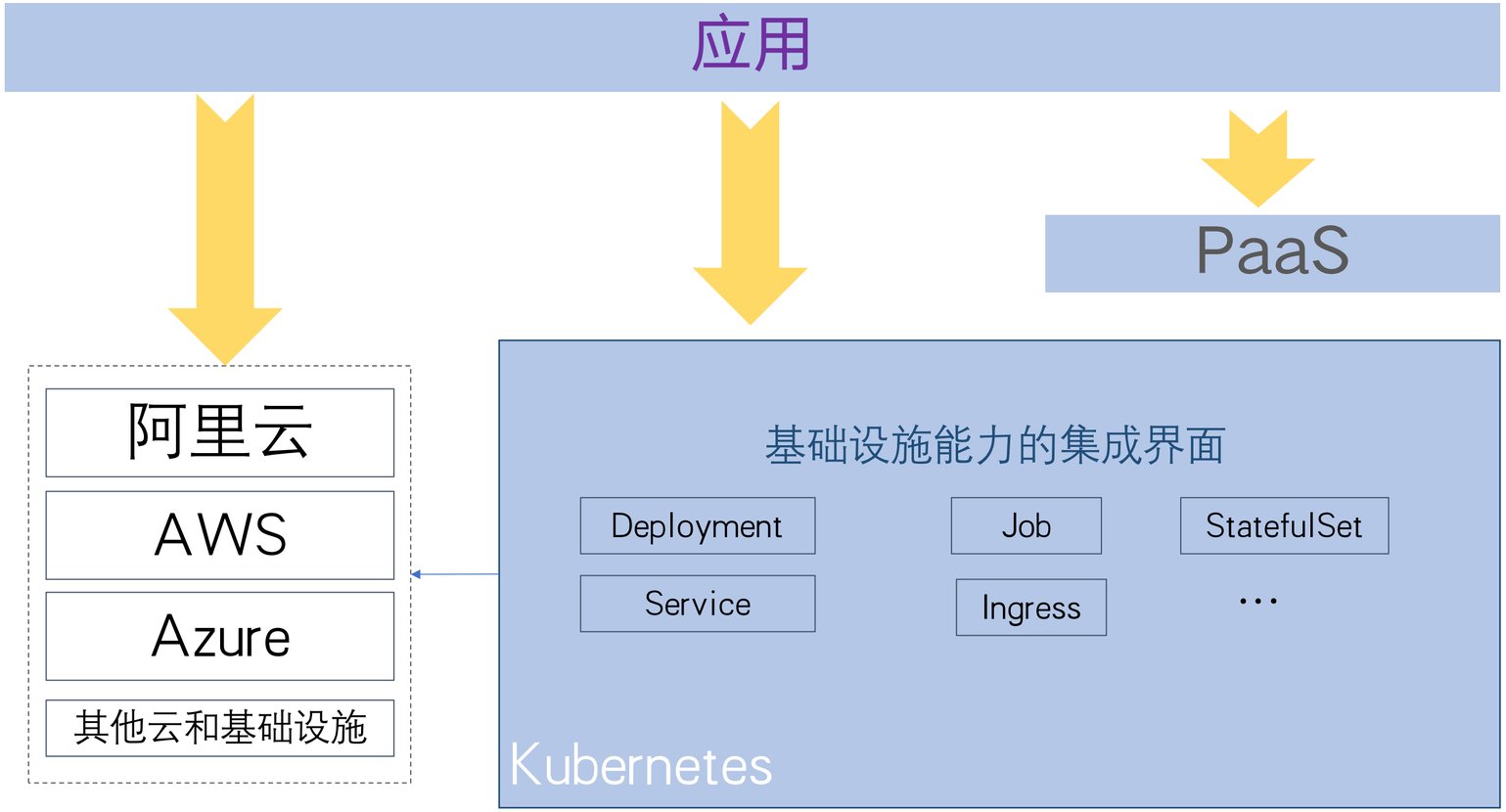
第 1 部分

云原生时代的应用管理

云原生时代究竟是哪个时代?



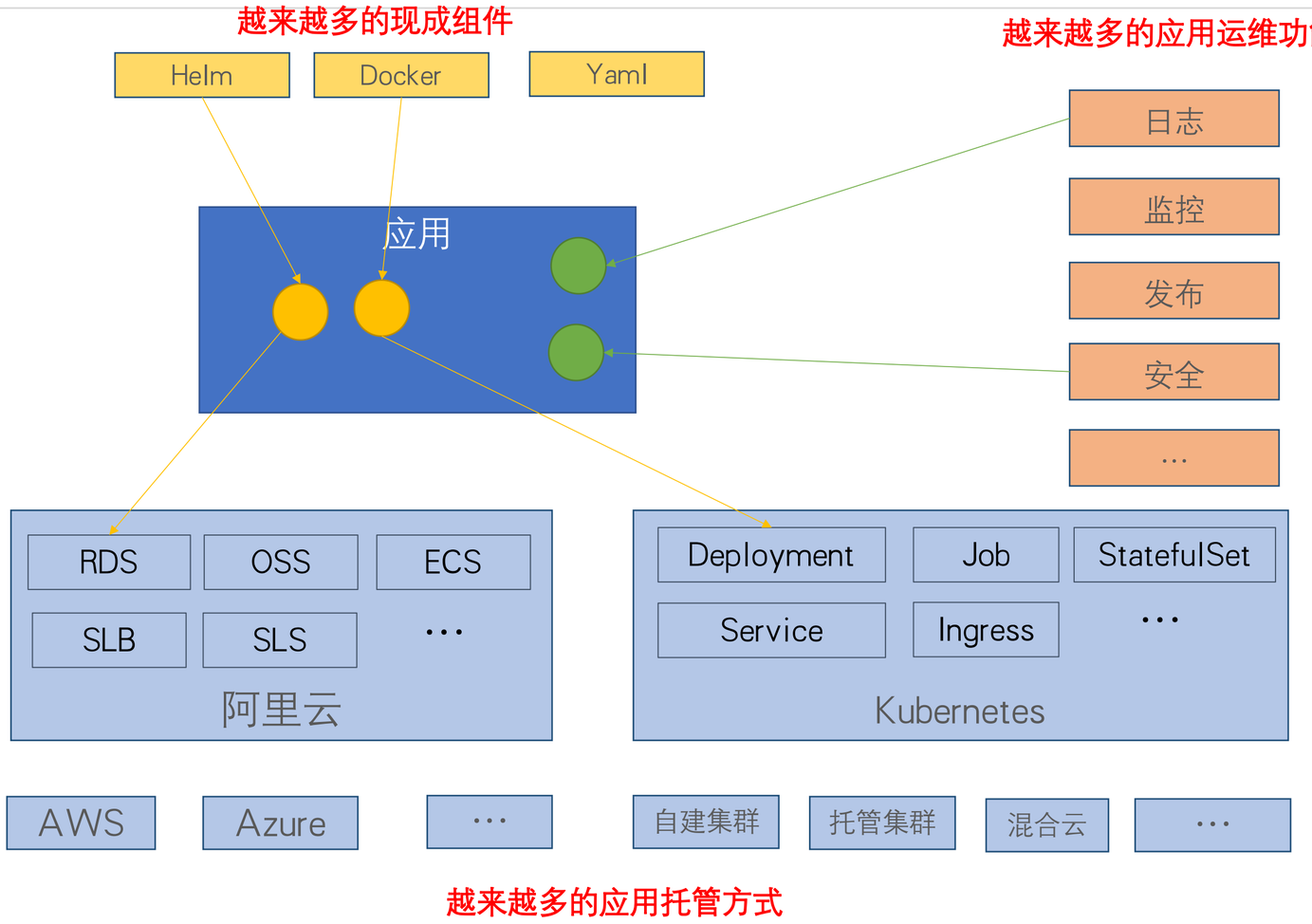
云原生时代的应用管理趋势



- 云原生基础设施：基于云的基础设施能力爆发式增长
- 云原生应用：应用架构往云的基础设施下沉

Kubernetes 为基础设施的接入提供了巨大的便利

云原生时代的应用管理挑战

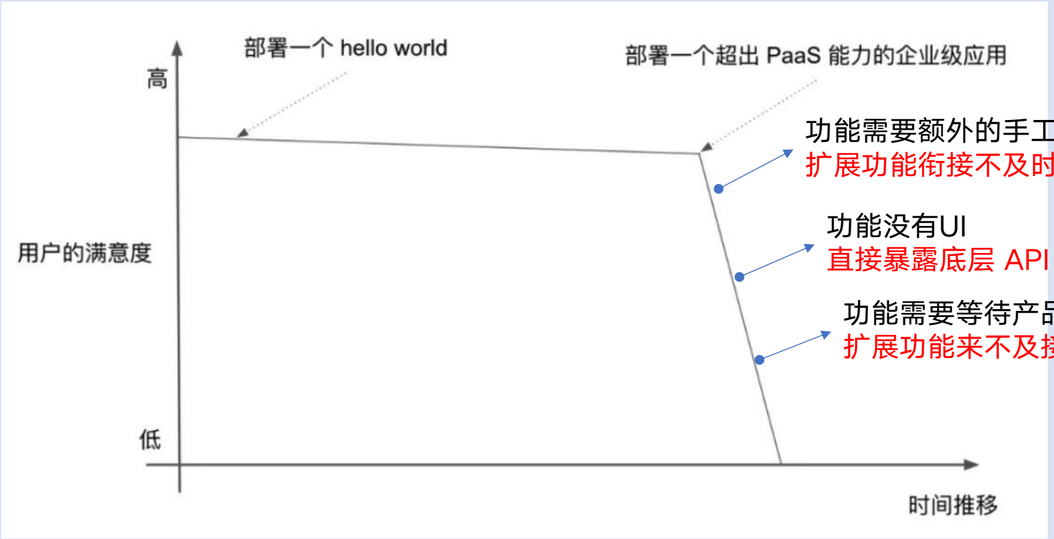


- 应用需求变化快，需要更多“自定义”
- 应用部署涉及到的内容增长很快
- 部署的难度越来越大

云原生时代的应用管理困境

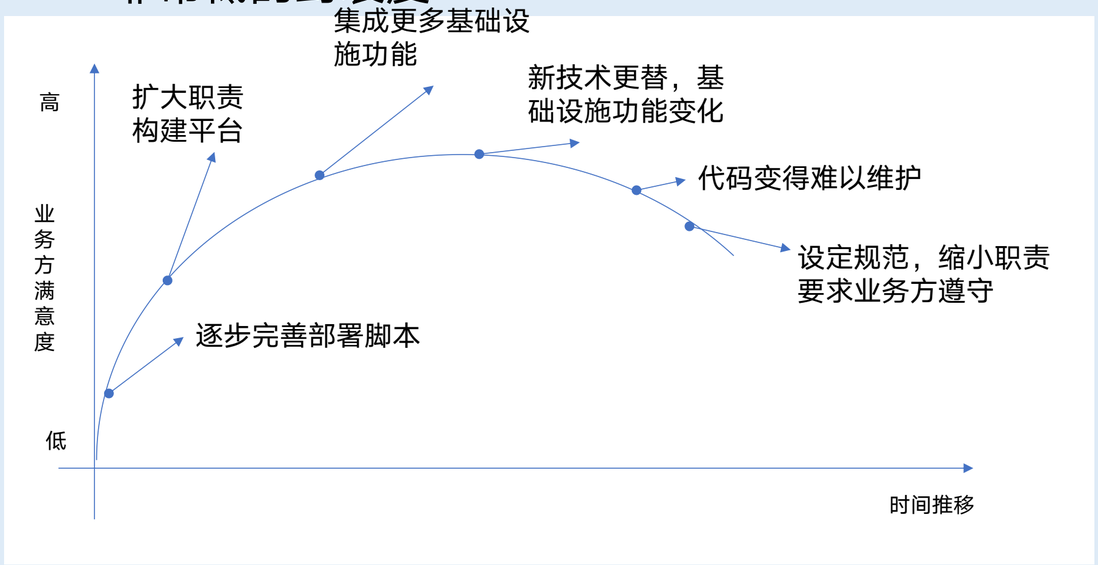
商业化 PaaS

- 低门槛、高用户体验
- 高封装度，提供统一部署和运维入口
- 难以扩展



自建的 PaaS

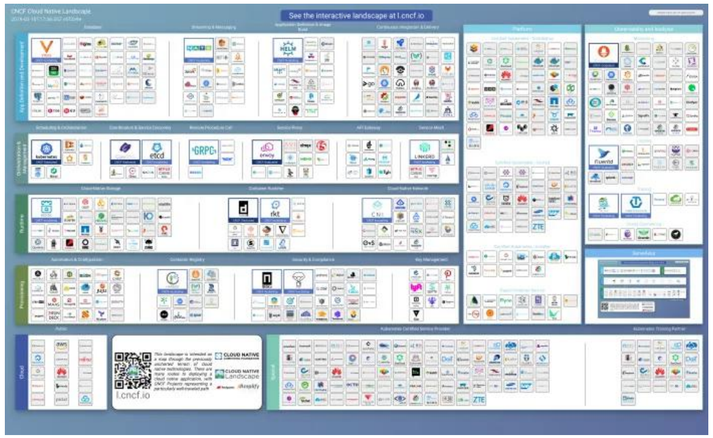
- 借助 Kubernetes 和云服务搭建
- 围绕 Kubernetes API 包装界面和工具（UI）
- 非常低的封装度



PaaS 系统的用户体验困境

传统的对接模式无法满足生态对接的需求

云计算生态“无限”的应用基础设施能力



传统 PaaS

有限的、不可扩展的专有API 与能力

研发与运维人员日益增长的应用管理诉求

“下一代”云原生应用管理需要什么？

- 充分的**可扩展性**，基础设施能力“以应用为中心”增长
- **最佳实践和标准范式**，保证软件交付效率和正确性
- **一致性体验**，屏蔽差异化环境的复杂性

第 2 部分

OAM 模型和 KubeVela 介绍

开放应用模型（OAM）

什么是模型？ 最佳实践的范式和框架

- 组织基础设施能力的方式
- 提供一致性体验的基础
- 软件交付效率和正确性的保障

2019.10
OAM 正式发布

面向应用开发者
无差别软件交付

2020.4
版本第一次升级

2020.11
KubeVela 发布

2021.5 OAM 被国家信通院列为行业标准

最佳实践

覆盖 ~80% 应用场景的模型

https://oam.dev/

可扩展

覆盖 100% K8s 应用场景

AWS ECS

Alibaba Cloud

Microsoft

upbound

Crossplane

Oracle
https://verrazzano.io/

4Paradigm 第四范式

SILOT

KubeVela

https://kubevela.io/

OAM 模型的具体实现

KubeVela：阿里巴巴新一代应用管理引擎

应用交付模型的执行引擎

借助应用模型对基础设施进行抽象，降低使用门槛

部署所需资源的完整集合与锚点（single source of truth）

云原生应用能力的“胶水”层

高度可扩展：模块化接入、拼图式衔接、管道式编排

能力“可编程”：高效响应需求变化，没有系统变更负担

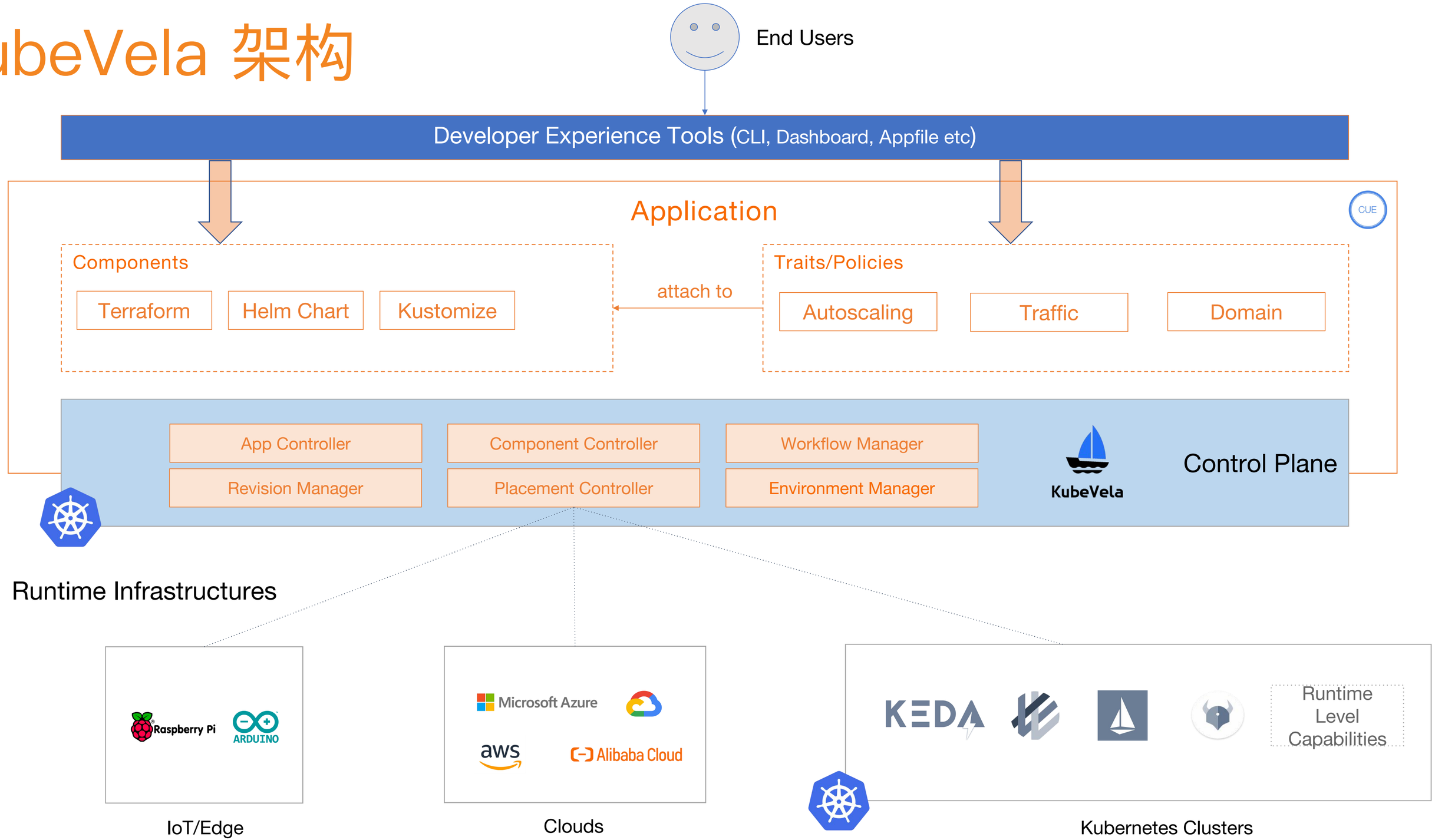
提供“以应用为中心”的能力注册中心和市场

无差别应用交付的控制平面

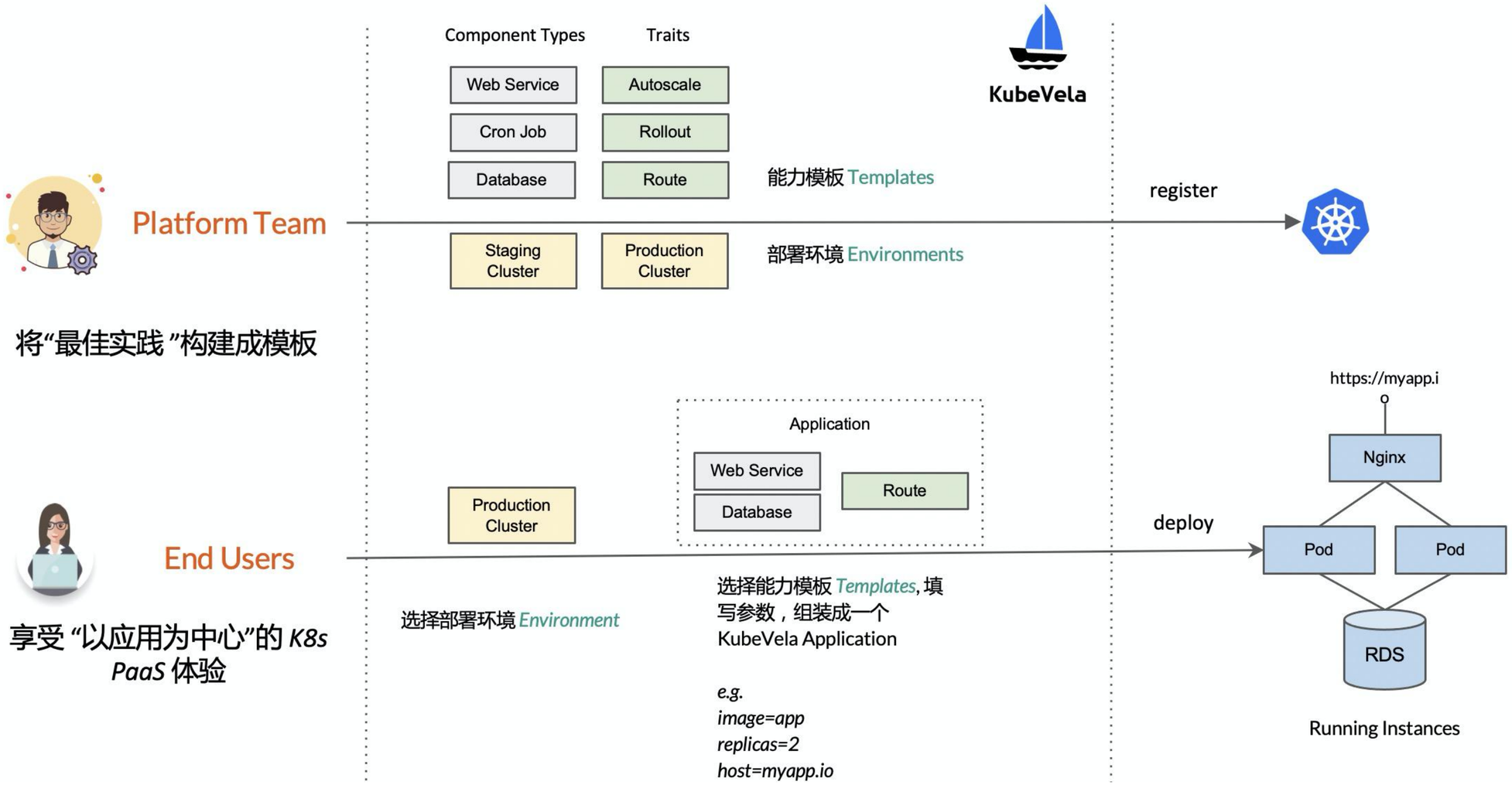
There多集群，多环境应用依赖管理，统一的运维能力建设

应用多环境部署和运维，差异化环境一致体验

KubeVela 架构


End Users
Developer Experience Tools (CLI, Dashboard, Appfile etc)
Application
CUE
Components
Terraform
Helm Chart
Kustomize
attach to
Traits/Policies
Autoscaling
Traffic
Domain
App Controller
Component Controller
Workflow Manager
Revision Manager
Placement Controller
Environment Manager
KubeVela
Control Plane
Runtime Infrastructures
Raspberry Pi
ARDUINO
Microsoft Azure
aws
Alibaba Cloud
KEDA
Runtime Level Capabilities
IoT/Edge
Clouds
Kubernetes Clusters

KubeVela 中的角色和工作流



KubeVela 应用描述（Application）

End Users

1. 声明一个应用的 YAML 文件
2. $ kubectl apply -f app.yaml
3. 创建成功

KubeVela 的 **Application** 可以无缝对接任何（Kubernetes Resource），Application 也是一个终端用户（End User）需要学习的唯一概念。

```yaml
apiVersion: core.oam.dev/v1alpha2
kind: Application
metadata:
  name: application-sample
spec:
  components:
    - name: foo
      type: webservice
      settings:
        image: crccheck/hello-world
        port: 8000
      traits:
        - name: ingress
          properties:
            domain: testsvc.example.com
            http:
              "/": 8000
        - name: sidecar
          properties:
            name: "logging"
            image: "fluentd"
    - name: bar
      type: aliyun-oss # cloud service
      bucket: "my-bucket"
```

应用描述：Application 的构成

- ✓ Application CRD 是一个“组合”对象。
- ✓ Application 的字段是灵活的，它的 schema 由 Definition CRD 确定。
- ✓ Definition CRD 可以灵活的注册/修改。

```yaml
apiVersion: core.oam.dev/v1alpha2
kind: Application
metadata:
  name: application-sample
spec:
  components:
    - name: foo
      type: webservice
      settings:
        image: crccheck/hello-world
        port: 8000
      traits:
        - name: ingress
          properties:
            domain: testsvc.example.com
            http:
              "/": 8000
        - name: sidecar
          properties:
            name: "logging"
            image: "fluentd"
    - name: bar
      type: aliyun-oss # cloud service
      bucket: "my-bucket"
```

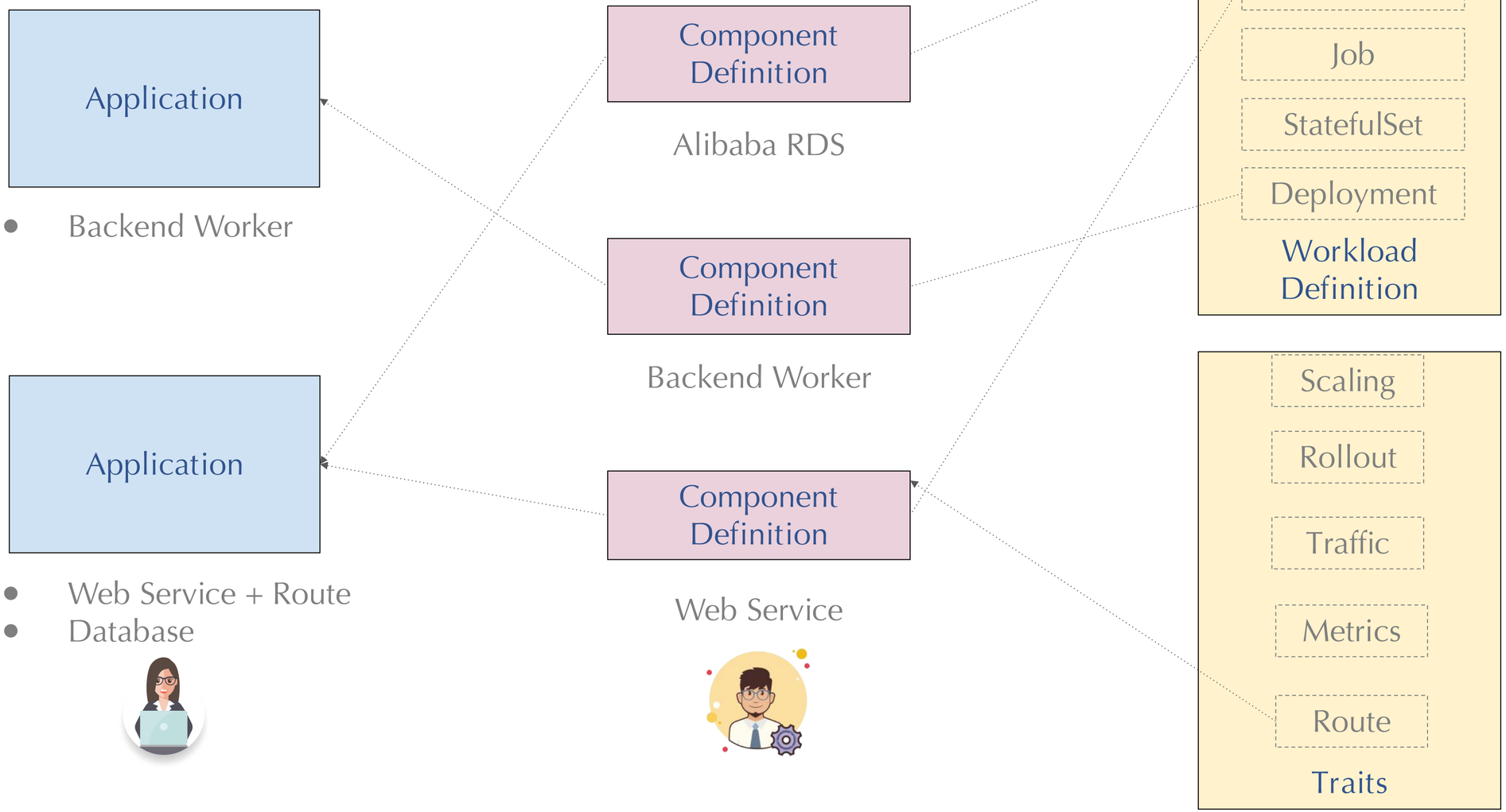
kind: Component Definition
…
name: webservice
schematic:
 image: string
 port: integer | 80

kind: Trait Definition
…
name: ingress
schematic:
 domain: string
 ...

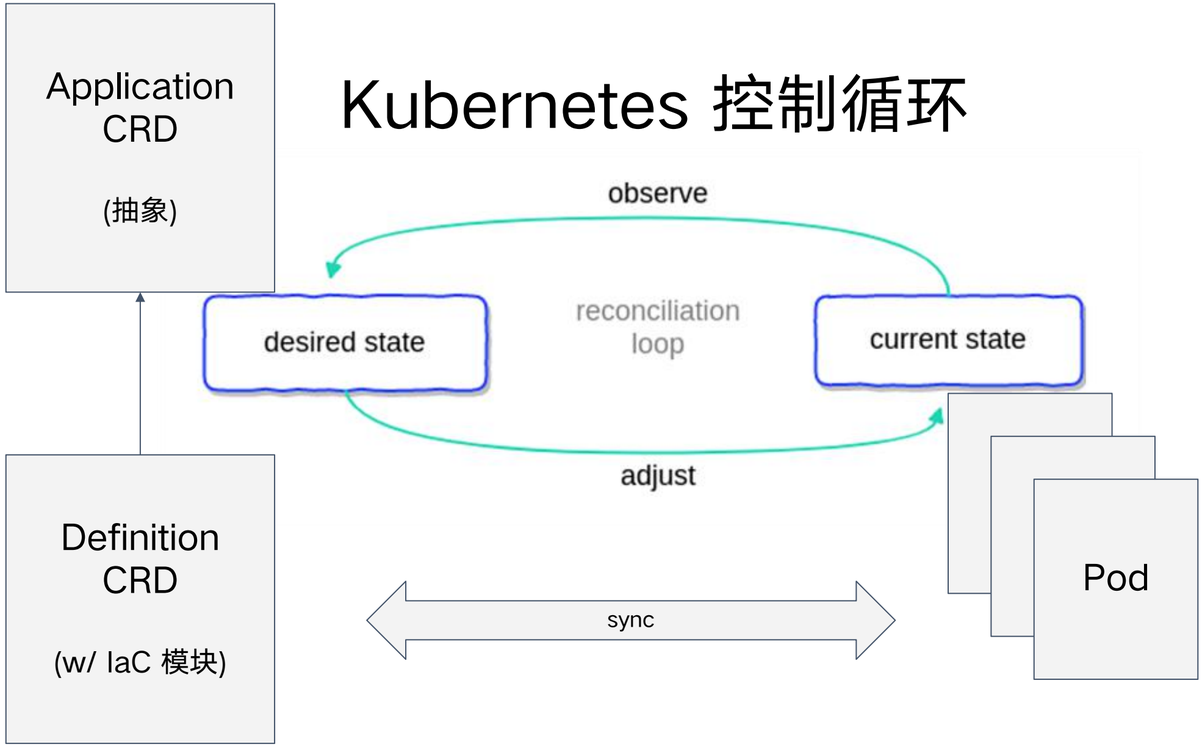
kind: Component Definition
…
name: aliyun-oss
schematic:
 bucket: string
 ...

Application 组成示例

基于 OAM 规范 Open Application Model



为什么 IaC (Infrastructure as Code) 与 Kubernetes 结合?



- IaC 是基础设施能力层最好的“胶水 ”，也是KubeVela的魔力所在
- 传统的 IaC 模式会导致“配置漂移”
  - 即：运行的实例与期望配置出现不一致
  - 越是大规模，问题越是严重

模块化能力

- 将基础设施能力变成“乐高”一样的构建模块呈现给最终用户
- 通过可编程（CUE, Terraform）的方式沉淀“运维经验”（IaC）
- “部署成功率”和“确定性”的保证。

可复用 -> 最佳实践 -> 组件中心

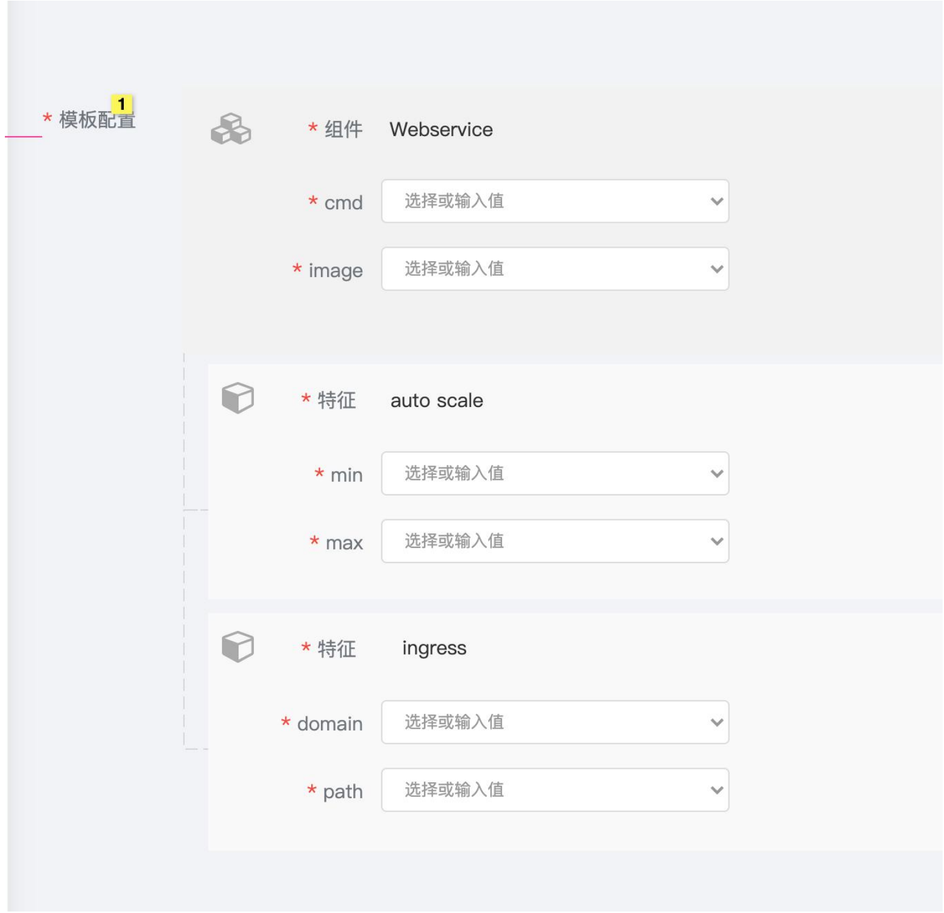
Full Sample

```
apiVersion: core.oam.dev/v1alpha2
kind: TraitDefinition
metadata:
  annotations:
    definition.oam.dev/description: "add sidecar to the app"
  name: sidecar
spec:
  appliesToWorkloads:
    - webservice
    - worker
  extension:
    template: |-
      patch: {
          // +patchKey=name
          spec: template: spec: containers: [parameter]
      }
      parameter: {
          name: string
          image: string
          command?: [...string]
      }
```

CUE module

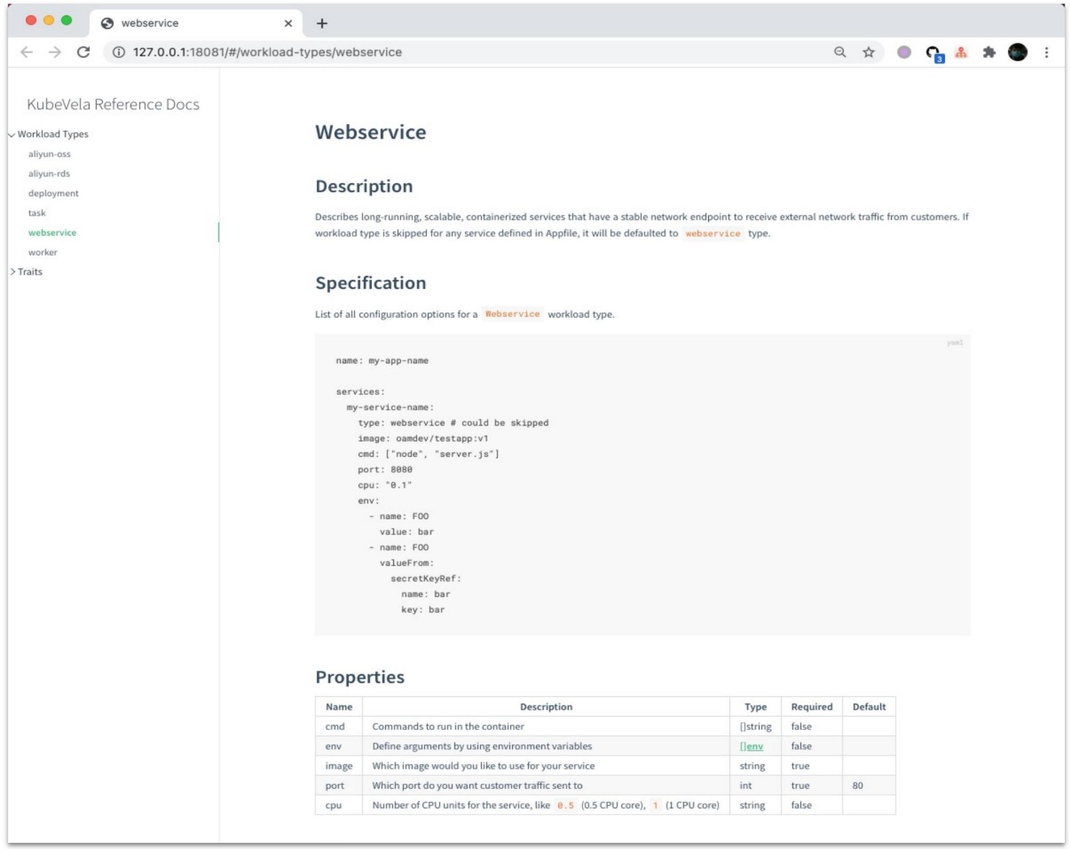
UI 自动化：模板注册后自动生成 UI

生成 OpenAPI v3 的 json-schema ，用于 GUI 前端表单的渲染和其他定制化对接



使用文档自动化：模板注册后，生成使用文档

$ vela show webservice --web



$ vela show scaler

```
➜  /Users/zhouzhengxi vela show scaler
# Properties
+----------+-----------------------------------+------+----------+---------+
|   NAME   |            DESCRIPTION            | TYPE | REQUIRED | DEFAULT |
+----------+-----------------------------------+------+----------+---------+
| replicas | Specify the number of workload | int  | true     |       1 |
+----------+-----------------------------------+------+----------+---------+
```

```
➜  /Users/zhouzhengxi vela show webservice
# Properties
+-----------------+-----------------------------------------------------------------------------------------+---------------------------------+----------+---------+
|      NAME       |                                       DESCRIPTION                                       |              TYPE               | REQUIRED | DEFAULT |
+-----------------+-----------------------------------------------------------------------------------------+---------------------------------+----------+---------+
| cmd             | Commands to run in the container                                                        | []string                        | false    |         |
| env             | Define arguments by using environment variables                                         | [[]env](#env)                   | false    |         |
| addRevisionLabel | If addRevisionLabel is true, the appRevision label will be added to the underlying pods | bool                            | true     | false   |
| image           | Which image would you like to use for your service                                      | string                          | true     |         |
| port            | Which port do you want customer traffic sent to                                         | int                             | true     |      80 |
| cpu             | Number of CPU units for the service, like `0.5` (0.5 CPU core), `1` (1 CPU core)        | string                          | false    |         |
| memory          | Specifies the attributes of the memory resource required for the container.             | string                          | false    |         |
| volumes         | Declare volumes and volumeMounts                                                        | [[]volumes](#volumes)           | false    |         |
| livenessProbe   | Instructions for assessing whether the container is alive.                              | [livenessProbe](#livenessProbe) | false    |         |
| readinessProbe  | Instructions for assessing whether the container is in a suitable state to serve traffic. | [readinessProbe](#readinessProbe) | false    |         |
+-----------------+-----------------------------------------------------------------------------------------+---------------------------------+----------+---------+
```

第 3 部分

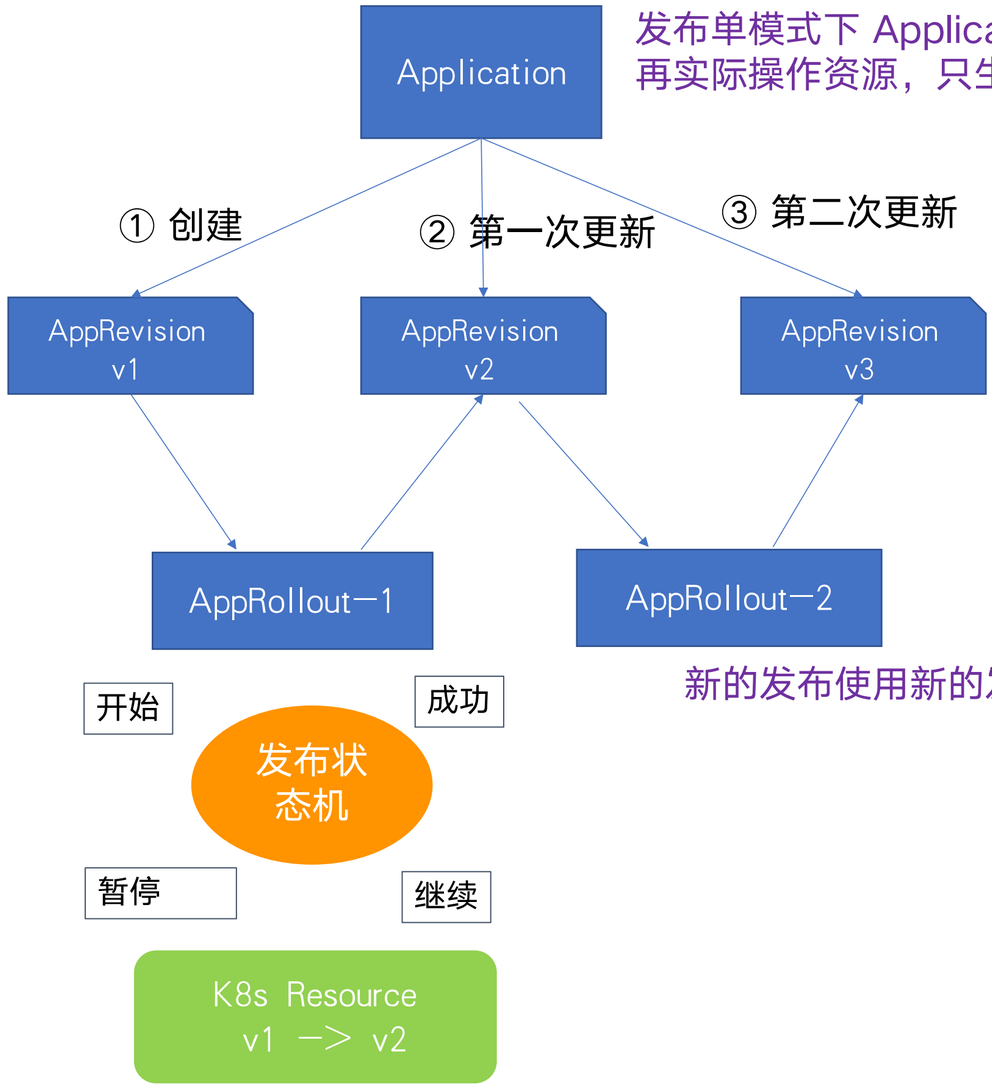
阿里巴巴基于 KubeVela 的实践

功能案例：多环境应用交付控制平面

- 仅需少量配置即可完成跨环境的应用部署
- K8s 控制循环保证应用面向终态的交付

功能案例：版本化和统一灰度发布

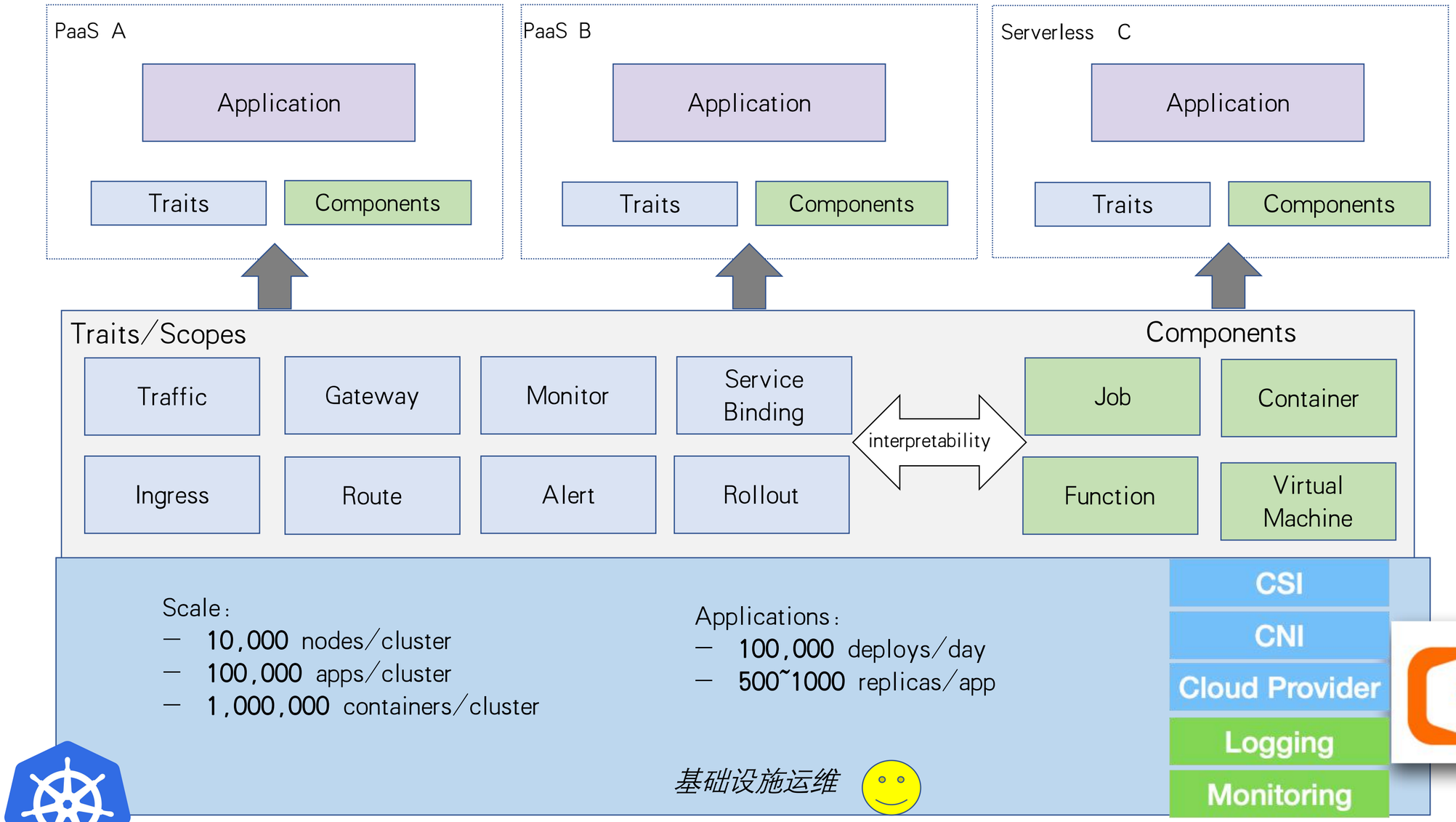
```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: my-app
spec:
  components:
    - name: metrics-provider
      type: clonesetservice
      properties:
        cmd:
          - ./podinfo
          - stress-cpu=2.0
        image: stefanprodan/podinfo:4.0.6
        port: 8080
  rolloutPlan:
    rolloutStrategy: "IncreaseFirst"
    rolloutBatches:
      - replicas: 50%
      - replicas: 50%
    targetSize: 6
    batchPartition: 2
```



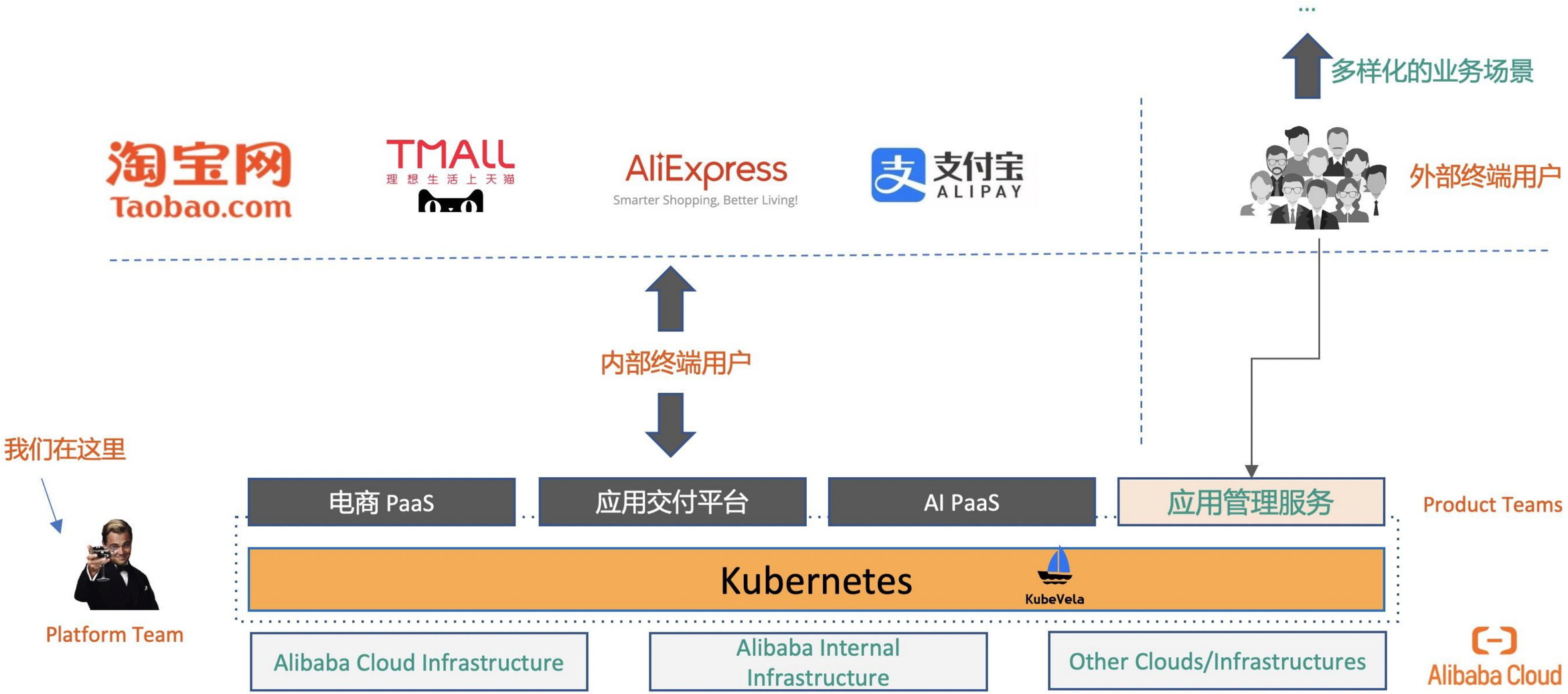
发布单模式下 Application 的更新不再实际操作资源，只生成版本快照

新的发布使用新的发布单对象

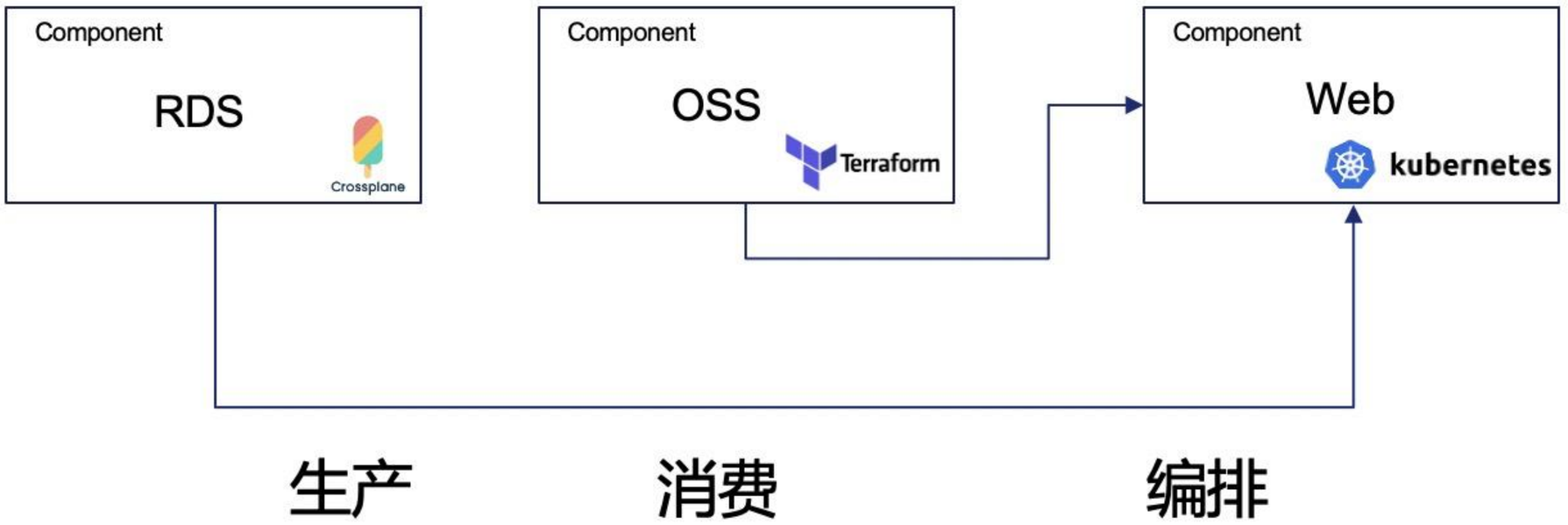
今天的阿里巴巴应用管理平台



阿里巴巴应用引擎实践大图



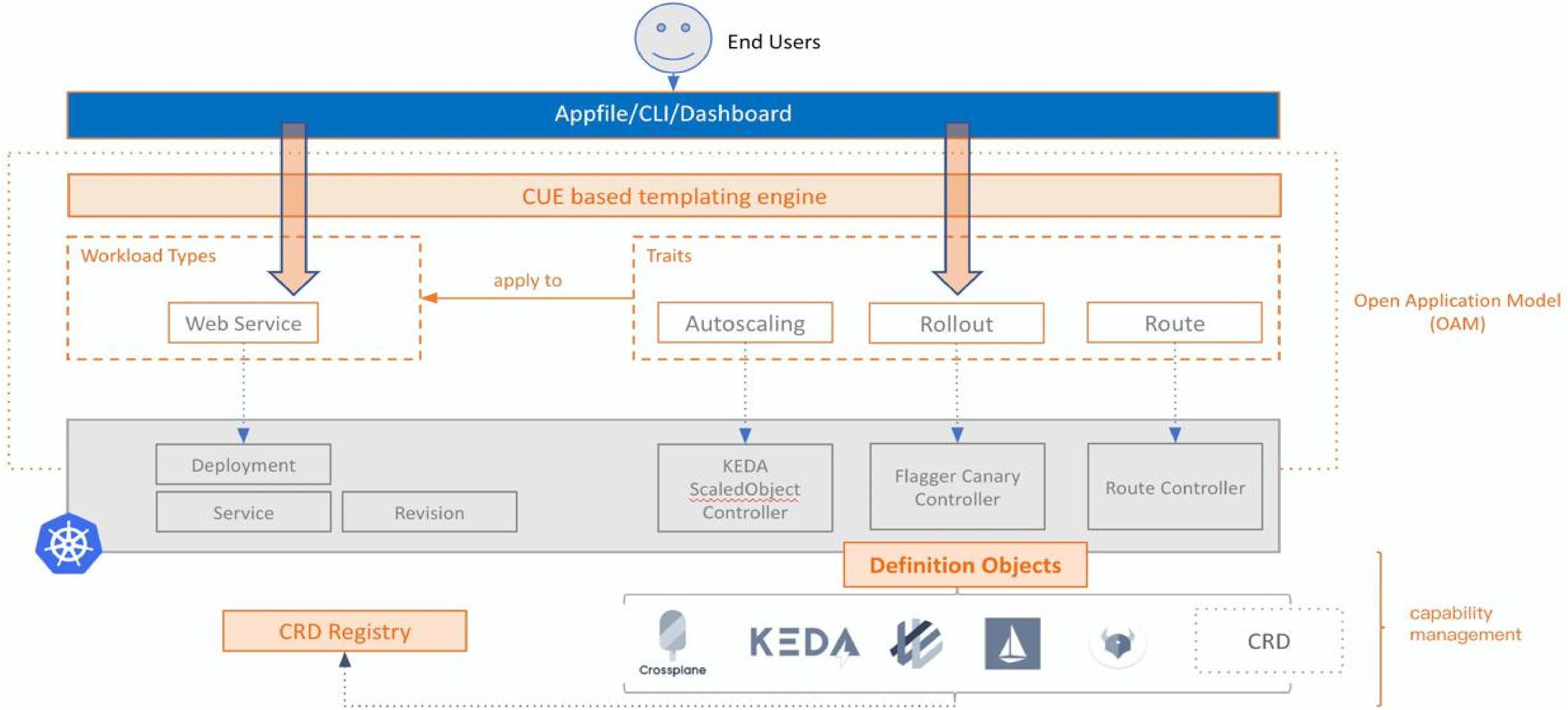
阿里巴巴某 BaaS 平台



Crossplane/Terraform/中间件Operator 可作为 KubeVela 的 Component 接入供应云资源。

多个 Component 之间可以定义数据的绑定和注入。

阿里云 EDAS（企业级分布式应用）



KubeVela 生态概览

- 20+ 国内外企业落地：Oracle、MasterCard、字节跳动、Napptive、silot、第四范式、有赞等
- 80+ 位社区贡献者，2200+ 人的社区
- 10W+ KubeVela 镜像下载
- 行业标准：OAM 被信通院立项作为行业标准。

钉钉千人交流群

KubeVela 开源团队**急需人才**：后端、前端工程师、架构师、布道师（杭州、北京、深圳）

第 4 部分

KubeVela 为什么是基于 Go

Golang 已然成为云原生第一事实语言

知名 Golang 项目：

- Containerd（容器运行时）
- Kubernetes（容器编排）
- Istio（服务网格框架）
- Terraform（基础设施编排）
- …

KubeBuilder：

一键生成代码框架，kubebuilder init

一键部署安装，make manifests/install

Golang 语法简单，容易上手

Open Application Model 运行时升级：

Rudr（Rust） → KubeVela（Golang）

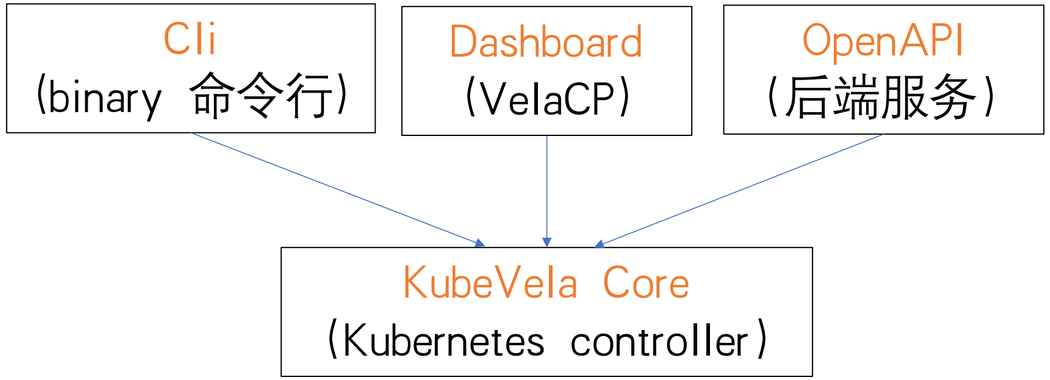
KubeVela 开源第二天，社区开发者贡献 Contour Ingress 控制器的支持

内库和第三方库非常丰富

面向 library 编程

- cobra
- gin-gonic
- swaggo
- kubernetes controller—runtime
- ……

部署简单

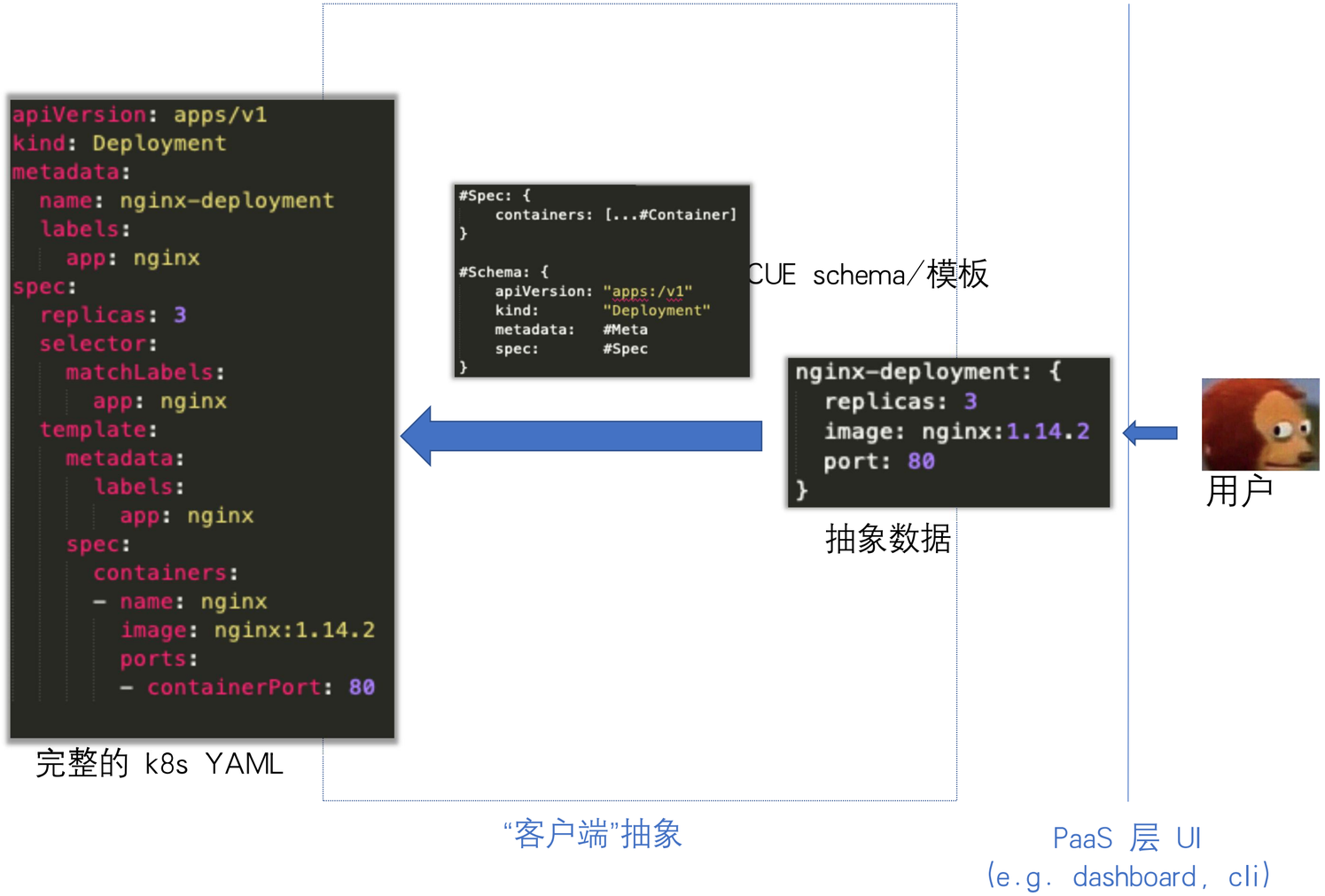


轻松搞定多样部署形式

Golang 与 Cuelang 轻松交互

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



欢迎提出宝贵意见☺

阿里巴巴新一代基于 Go 的云原生应用引擎实践

周正喜 阿里巴巴技术专家