2020 *Cloud Native Day*
云原生生态大会

KubeVela: An Easy-to-use Yet Highly Extensible App Platform

张磊 / Lei Zhang
阿里云高级技术专家

What is KubeVela?

KubeVela is an **easy-to-use** yet highly **extensible** app platform

KubeVela 是一个**简单易用**又**高度可扩展**的云原生应用管理平台

简单易用

完整的用户侧抽象，使用者无需学习任何 Kubernetes 细节与 API

完全以应用为中心，用户只需关心“代码”、“构建”与“部署”

简单友好的操作界面：Appfile 与 Dashboard（即将发布）

天然 GitOps 友好

Appfile

一个完整的应用描述文件

放置于应用代码库中

一键部署：$ vela up

自动适配任意 k8s 集群与部署环境

```yaml
name: testapp

services:
  express-server:
    image: oamdev/testapp:v1
    build:
      docker:
        file: Dockerfile
        context: .

    cmd: ["node", "server.js"]
    port: 8080
    cpu: "0.01"

    route:
      domain: example.com
      rules:
        - path: /testapp
          rewriteTarget: /

    autoscale:
      min: 1
      max: 4
      cpuPercent: 5
```

代码
代码库路径

构建
Dockerfile 路径

如何启动
启动命令、端口、资源需求

如何访问
域名、路由策略

如何扩容
扩容指标，实例数范围

GitOps with Appfile + GitHub Action

Git 项目代码库 + vela.yaml

提 PR 更新代码

高度可扩展

所有能力可插拔

Kubernetes 原生的可扩展性

示例：上线新功能 metrics

平台研发团队：

开发了一个新 Operator 叫做 metrics（监控）

编写一个 K8s 能力描述文件 metrics.yaml

平台管理员：

执行 $ kubectl apply -f metrics.yaml

用户：

立刻就可以在 Appfile 中定义一个新的字段 metrics

无需系统更新或重启

```
name: testapp

services:
  express-server:
    type: webservice
    image: oamdev/testapp:v1
    build:
      docker:
        file: Dockerfile
        context: .

    cmd: ["node", "server.js"]
    port: 8080
    cpu: "0.01"

    route:
      domain: example.com
      rules:
        - path: /testapp
          rewriteTarget: /

    autoscale:
      min: 1
      max: 4
      cpuPercent: 5

    metrics:
      port: 8080
      path: "/metrics"
      scheme:  "http"
      enabled: true
```

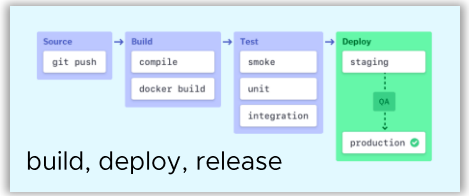
Why KubeVela?

Bring **application** context back to k8s!

pplication *developers, operators*

what the platforms provide

App-Centric API



App-Centric Abstractions

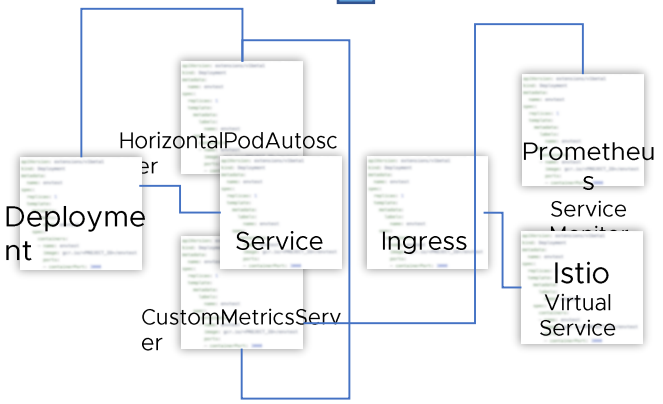
scaling
- auto scale +100 instances when latency > 10%

rollout
- promote the canary instance with step of 10%

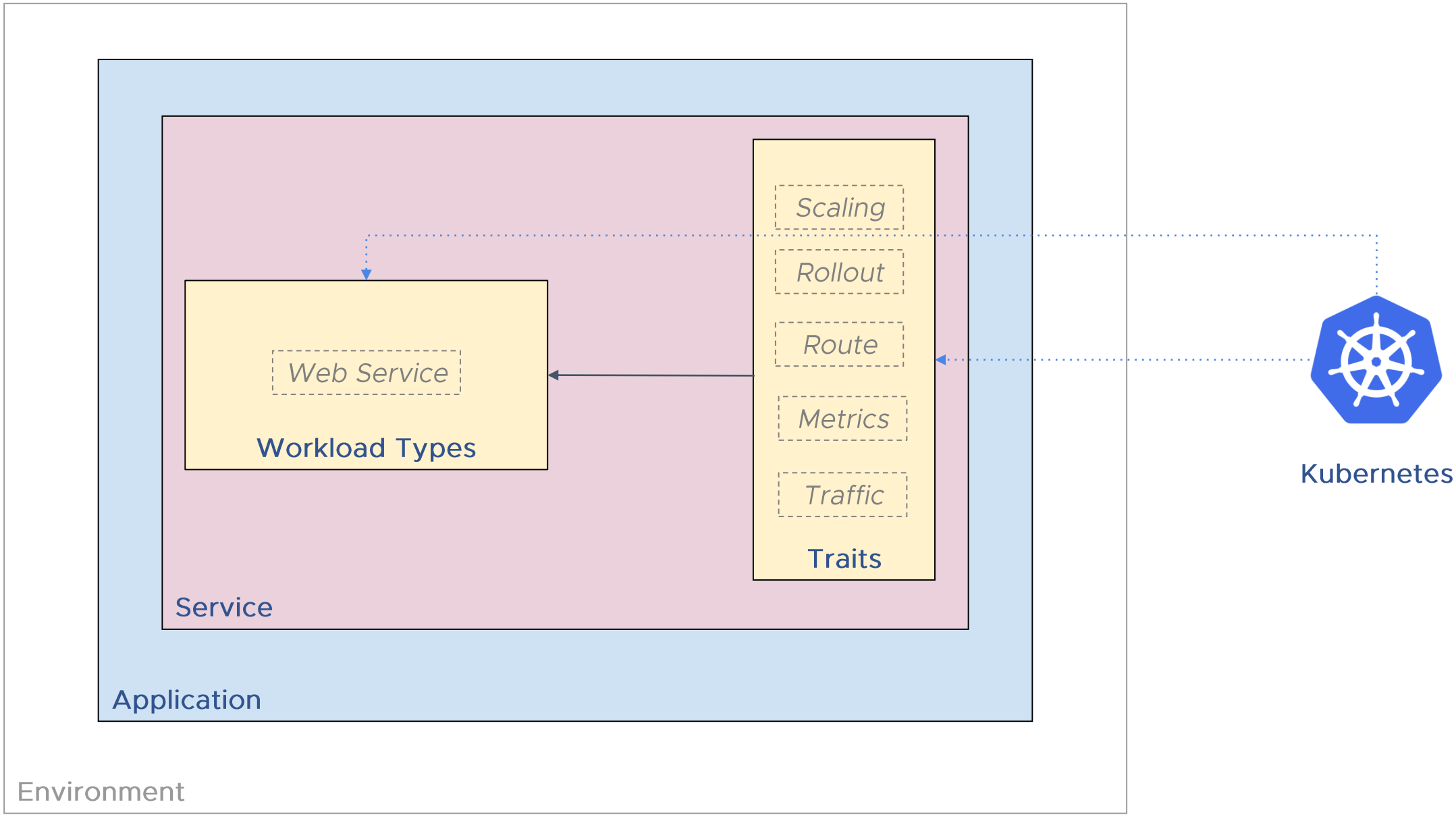
App-Centric User Interfaces

what k8s provides

Deployment | Pod
HPA | Controller
Sidecar | Node
NetworkPolicy | CR/CRD



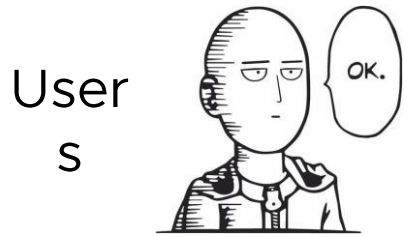
KubeVela Core Concepts

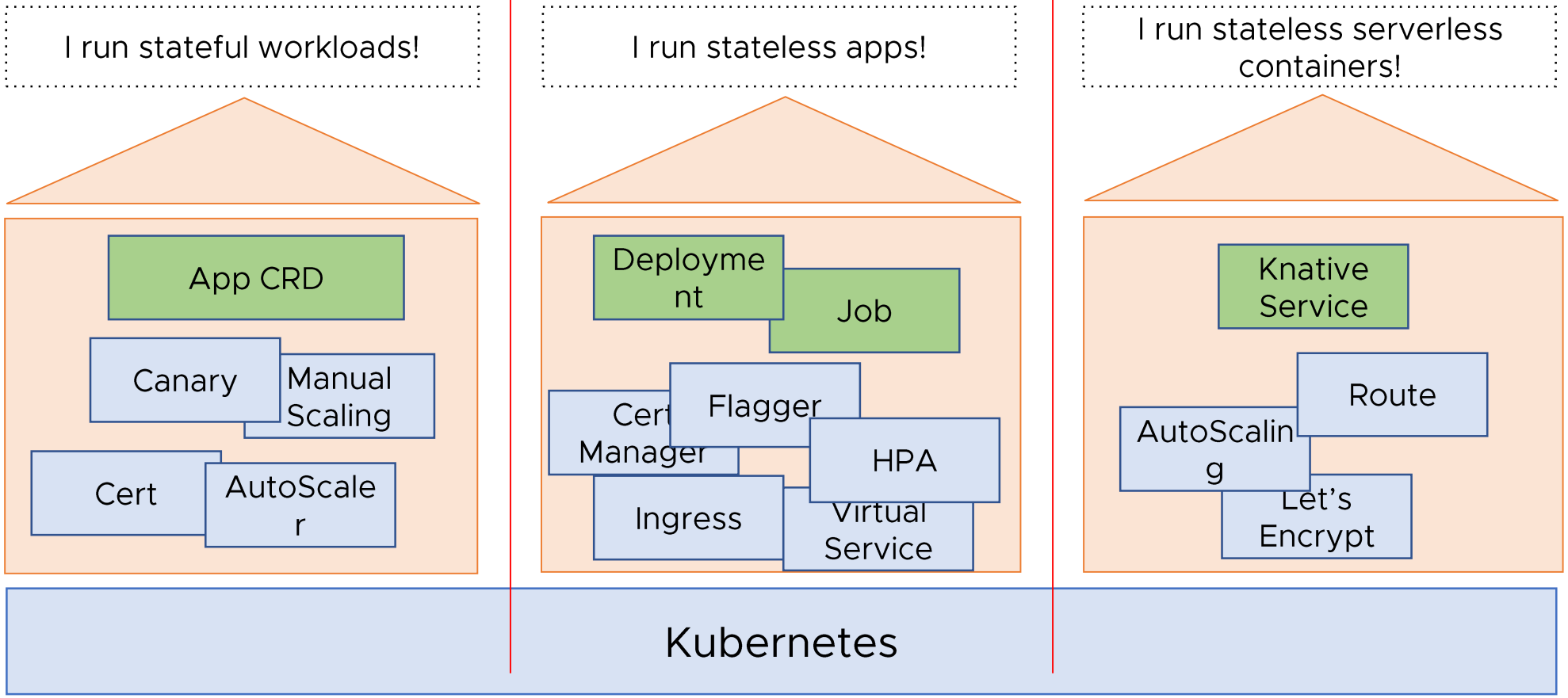


Why KubeVela?

Building app platforms is hard ...

- **Fragmentation:** ~11 PaaS/Serverless in Alibaba
- **Silos:** no interoperability, reusability, or portability
- **Close:** many in-house wheels due to in-house app crd



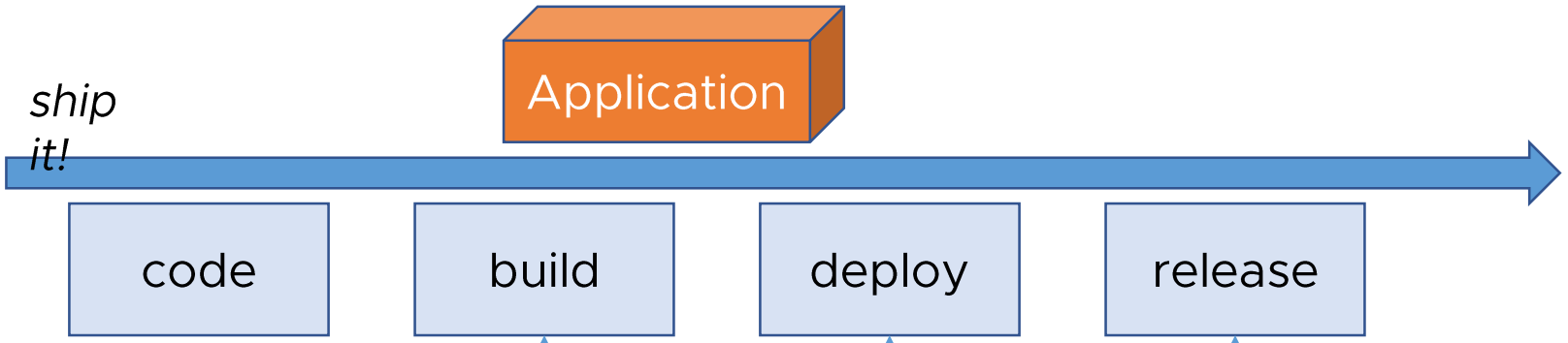


A Unified Model for Both

KubeVela aims at **both** *developers* and *platform builders*

developers

ship it!



target k8s cluster

- Dockerfile
- Buildpack
- …

- workload
- storage
- ingress
- autoscaling
- …

- traffic shifting
- domain
- canary
- blue-green
- …

capabilities

platform builders

- *Developers*:
  - here's my code, please ship it!
- *Platform Builders*:
  - how can I assemble capabilities of k8s into a developer friendly platform at ease?

Design of KubeVela

- **Application-Centric**
  - We believe “application” should be the main (maybe the only?) API our platform exposes to users.
  - e.g. KubeVela adopts *Open Application Model (OAM)* as the app-centric API and introduces *Appfile* as the last mile developer tool
- **Capability Oriented Architecture (COA)**
  - **Every** feature in KubeVela is a independent plugin (either a k8s built-in resource or your own CRD controller).
  - e.g. Alibaba use KubeVela adopts Flagger as *rollout* trait, KEDA as *autoscaling* trait
- **Highly extensible, even for its user interface**
  - When a new capability is installed, it should immediately consumable by end users without re-compiling or re-installing KubeVela.
    - e.g. KubeVela’s *Appfile*

Appfile Deep Dive


```
services:
  express-server:
    build:
      image: oamdev/testapp:v1
      docker:
        file: Dockerfile
        context: .

    cmd: ["node", "server.js"]
    route:
      domain: example.com
      http: # match the longest prefix
        "/": 8080

    env:
      - FOO=bar
      - FOO2=sec:my-secret # map the key sam
      - FOO3=sec:my-secret:key # map specifi
      - sec:my-secret # map all KV pairs fro

    files: # Mount secret as a file
      - /mnt/path=sec:my-secret

    scale:
      replica: 2
      auto: # automatic scale up and down ba
        range: "1-10"
        cpu: 80 # if cpu utilization is abov
        qps: 1000 # if qps is higher than 1k

    canary: # Auto-create canary deployment.
      replica: 1 # canary deployment size
      headers:
        - "foo:bar.*"
```

```
apiVersion: core.oam.dev/v1alpha2
kind: WorkloadDefinition
metadata:
  name: webservice
spec:
  definitionRef:
    name: deployments.apps
  extension:
    template: |
      parameter: #webservice
      #webservice: {
        // +vela:cli:enabled=true
        // +vela:cli:usage=specify commands to run in container
        // +vela:cli:short=c
        cmd: [...string]

        env: [...string]

        files: [...string]
      }

      output: {
        apiVersion: "apps/v1"
        kind: "Deployment"
        metadata:
          name: context.name
        spec: {
          selector: {
            matchLabels:
              app: context.name
          }
          template: {
            metadata:
              labels:
                app: context.name
            spec: {
              containers: [{
                name:  context.name
                image: context.image
                command: parameter.cmd
              }]
            }
          }
        }
```

Capability Definition 1

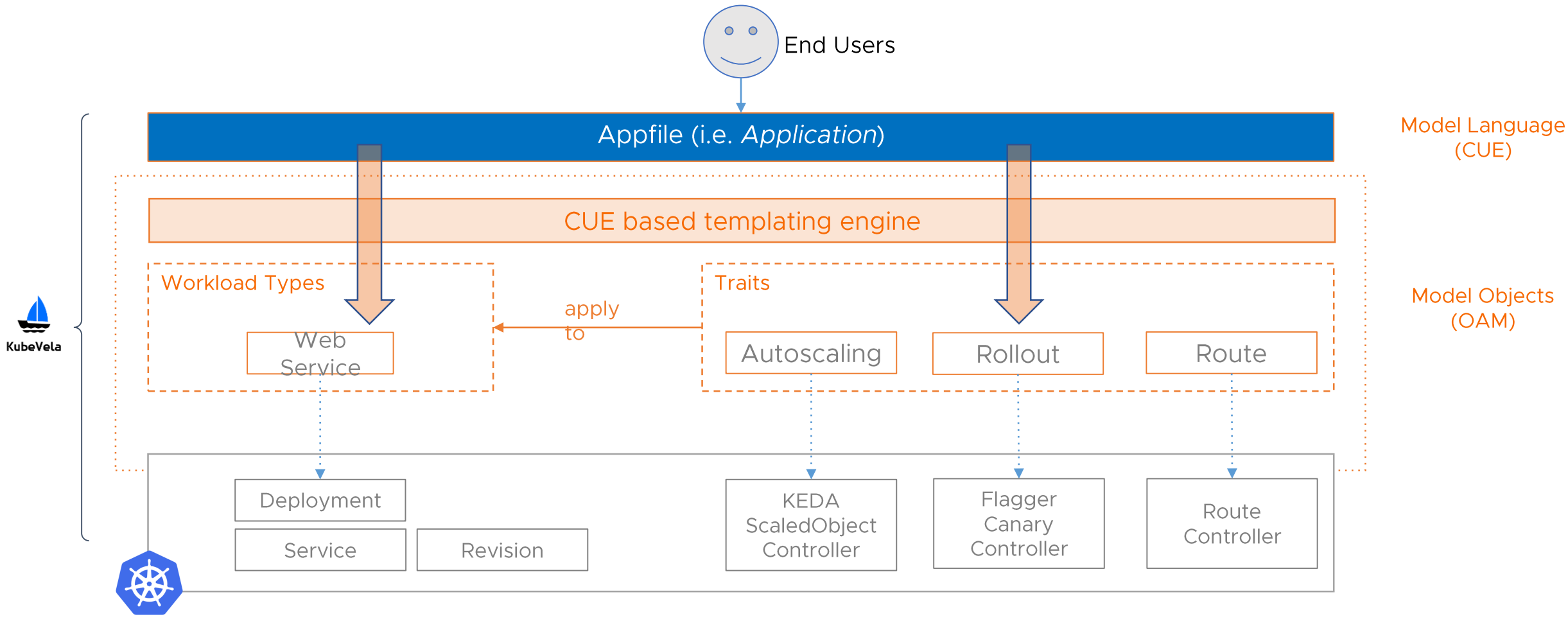
```
apiVersion: core.oam.dev/v1alpha2
kind: TraitDefinition
metadata:
  name: route
spec:
  definitionRef:
    name: routes.standard.oam.dev
  extension:
    template: |
      parameter: #route
      #route: {
        domain: string
        http: [string]: int
      }

      // trait template can have multiple outputs and they are all traits
      outputs: service: {
        apiVersion: "v1"
        kind: "Service"
        metadata:
          name: context.name
        spec: {
          selector:
            app: context.name
          ports: [
            for k, v in parameter.http {
              port: v
              targetPort: v
            }
          ]
        }
      }
```

Capability Definition 2

- **Simple**
  - Think about docker-compose but for Kubernetes.
  - Designed to ship (build -> release) cloud native app by one click.
- **Extensible**
  - Every section in Appfile references a independent capability definition
- **CUE based**
  - The schema of each section is enforced by CUE template defines in capability definition.

Appfile Under The Hood



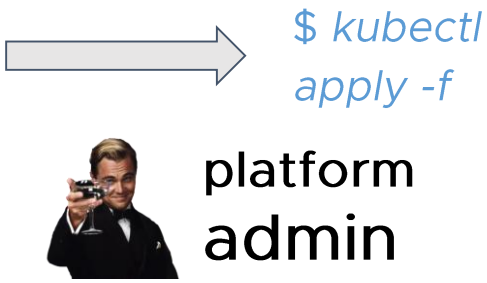
Add a New Capability in KubeVela

Any capability in the open source community, could be shipped as a ***feature*** of KubeVela by a simple $ *kubectl apply -f definition.yaml*, and become usable immediately to users through *Appfile* and other UI tools

- This is also how all built-in features in KubeVela are shipped btw.

Let's say:

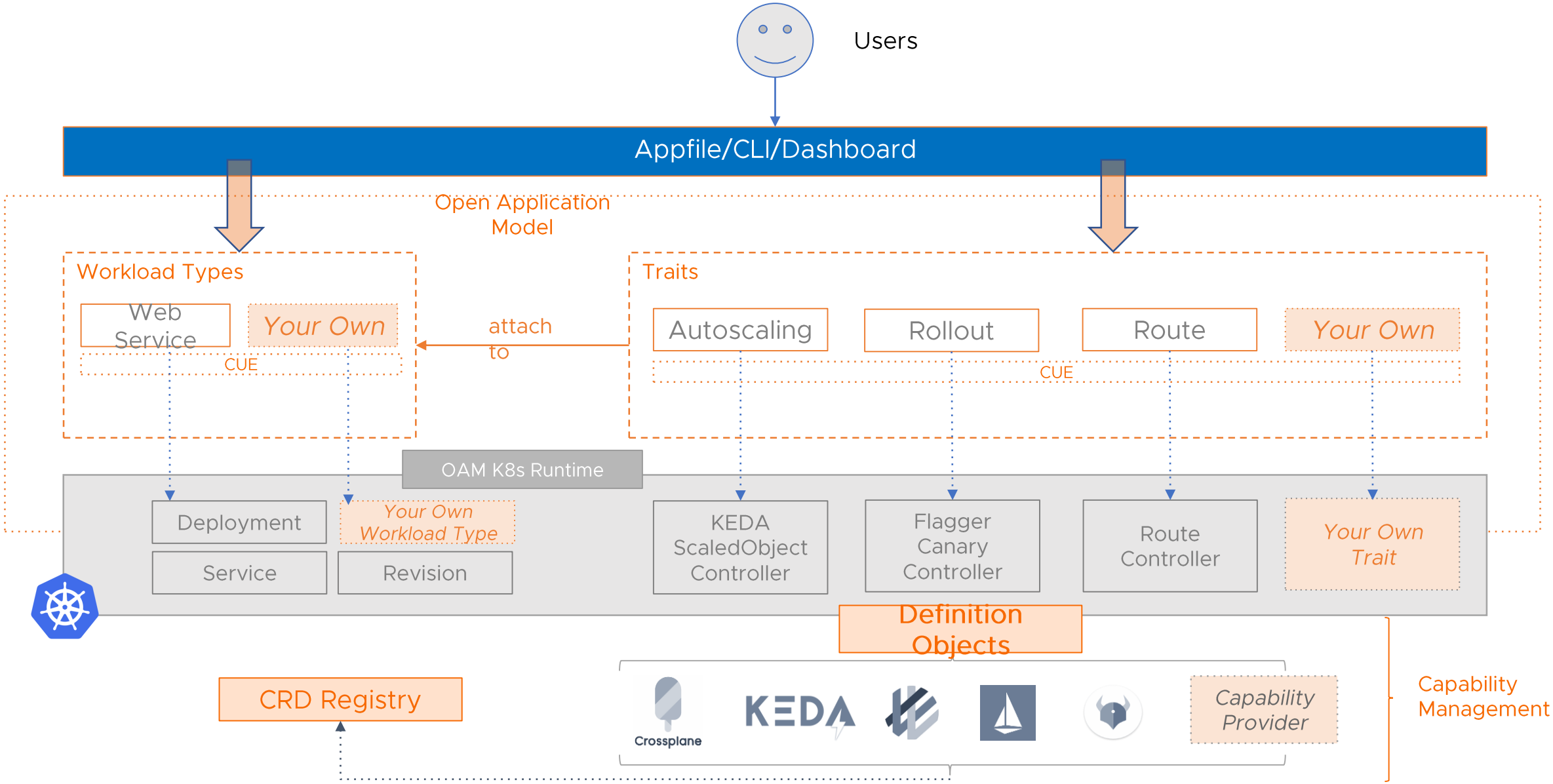
- add a new workload type of
  - AWS RDS?
  - stateful workload?
  - Redis Operator?
- add a new capability such as:
  - app metrics?
  - blue-green rollout?
  - custom domain?
  - ... enable Dapr?



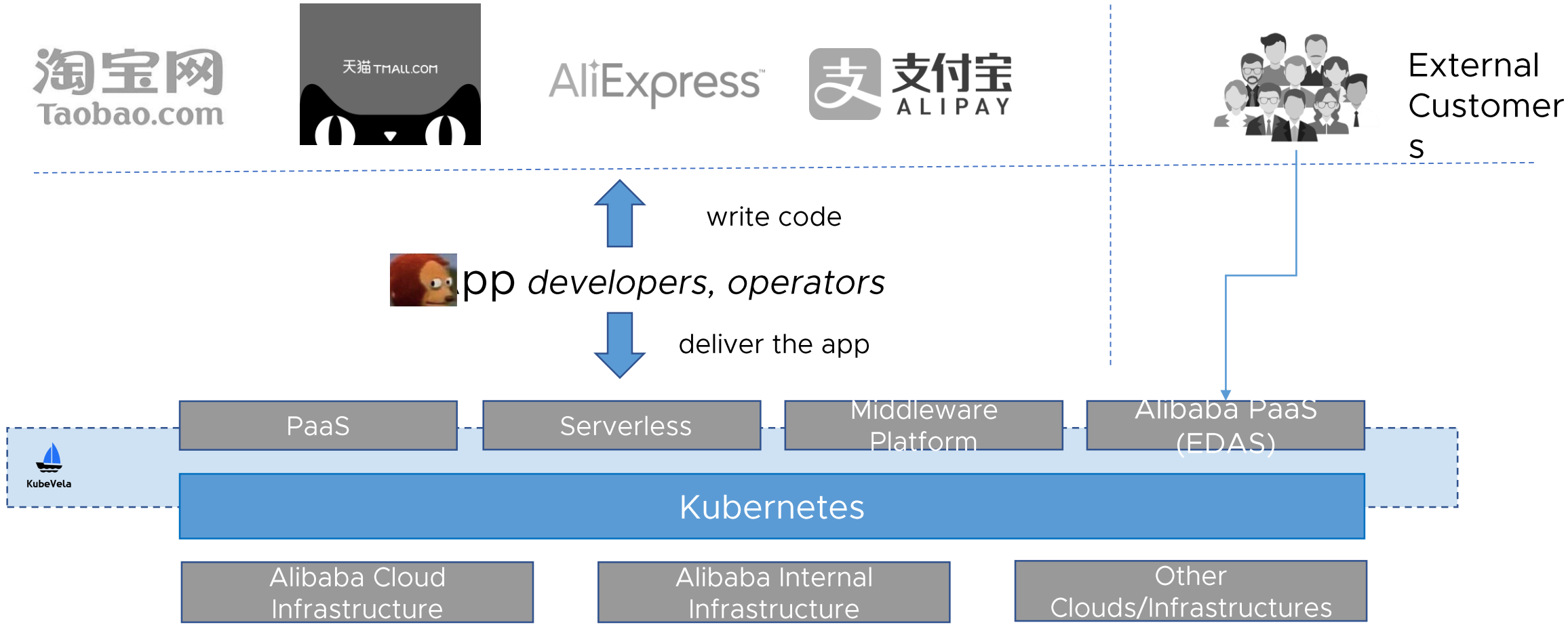

```yaml
apiVersion: core.oam.dev/v1alpha2
kind: TraitDefinition
metadata:
  name: metric
  namespace: default
  annotations:
    definition.oam.dev/apiVersion: standard.oam.dev/v1alpha1
    definition.oam.dev/kind: MetricsTrait
    definition.oam.dev/description: "Add metric monitoring for workload"
spec:
  appliesToWorkloads:
    - webservice
    - backend
    - task
    - containerizedworkloads.core.oam.dev
    - clonesetworkloads.apps.kruise.io
    - deployments.apps
    - statefulsets.apps
  definitionRef:
    name: metricstraits.standard.oam.dev
  workloadRefPath: spec.workloadRef
  extension:
    template: |-
      #metrics: {
        // +usage=format of the metrics, default as prometheus
        // +short=f
        format:  *"prometheus" | string
        path:    *"/metrics" | string
```

definition.yaml

Summary: Architecture



Summary: KubeVela in Alibaba - The “PaaS Core”



Thank You