CHINA · BEIJING

阿里巴巴 Kubernetes 应用管理实践中的经验与教训

孙健波

阿里云 技术专家

关注“阿里巴巴云原生”公众号
获取第一手技术资料

SPEAKER
INTRODUCE

孙健波 阿里云 技术专家

- Kubernetes 基础技术中台团队
- 开放应用模型（OAM）项目 Core Maintainer
- jianbo.sjb@alibaba-inc.com

TABLE OF

CONTENTS 大纲

- 阿里存量 PaaS 对接 Kubernetes 的新挑战
- 研发和运维对 Kubernetes YAML 文件的看法
- 阿里对解耦研发和运维的实践与教训
- 标准化、统一化的应用管理

阿里巴巴大规模容器化基础设施

初步探索

使用容器的方式替换传统使用 VM 部署应用的，基于 lxc 自研了 t4 容器并构建了 AI 集团管理系统

野蛮生长

体验到使用容器部署应用的优势后，阿里内部发展出众多的运维平台，包括 AliSwarm，Zeus，Hippo 等，极大地降低了业务运维的成本

统一资源池

构建了 Sigma 调度系统，收敛了众多运维平台之下的资源调度系统，并构建了集团统一资源池，在此基础上发展出弹性、混部等技术成果，大幅降低了数据中心的资源成本

转型 kubernetes

从自研的 Sigma 调度转型到 k8s 体系，初步尝试面向终态的运维体系，通过 k8s 的方式将 Sigma 调度能力提供给客户

全面拥抱云原生

阿里业务全面上云，运维体系全面拥抱云原生，基于 k8s 生态在阿里内部蓬勃发展。在 2019 双 11 中，k8s 体系支撑了阿里史上规模最大的集群，并提供了极速的应用扩容体验

新挑战

- 研发：Kubernetes API 太复杂?
- 运维：如何上手 Kubernetes 的扩展能力?
- 如何通过 Kubernetes 全面管理云资源（含虚拟机、VPC 等）？

K8s API 太复杂？ All in one。

思考题：

对于一个 K8s 应用的描述，大家的关注点是？

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: wordpress
  namespace: default
spec:
  replicas: 1
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    spec:
      containers:
        image: docker.io/bitnami/wordpress:5.3.0
        ports:
        - containerPort: 80
          name: http
          protocol: TCP
        resources:
          requests:
            cpu: 300m
            memory: 512Mi
      dnsPolicy: ClusterFirst
      hostAliases:
      - hostnames:
        - status.localhost
        ip: 127.0.0.1
      schedulerName: default-scheduler
      terminationGracePeriodSeconds: 30

```

- 运维关心
- 研发关心

根本看不懂

简单却能力不足：

某内部 PaaS 精挑细选，只剩下 ~5 个 Deployment 的字段允许研发填写。

```
image: quay.io/coreos/prometheus-operator:v0.34.
args:
  - --logtostderr=true
ports:
- containerPort: 8080
  name: http
  protocol: TCP
envs:
  - name: INNER-KEY
    value: app
volumes:
  - name: cache-volume
    emptyDir: {}
```

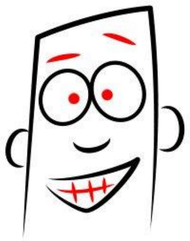
思考题：

有状态的复杂应用如何管理？
基础设施能力还如何演进和透出？
研发自己的诉求如何传达给运维和基础设施？

K8s 扩展能力的真实情况

我的 Zookeeper 该用
哪种K8s Workload 接入?

Operator是啥?



我们业务压力大.
一定要学这些么……

你恐怕得写个 Operator……

CRD
Controller
Informer
Reflector
Event Handler Loop
…

那好吧, 我们陪你写….

运维如何上手K8s的扩展能力?

举例：CronHPA

```yaml
apiVersion: "app.alibaba.com/v1"
kind: CronHPA
metadata:
  name: cron-scaler
spec:
  timezone: Asia/Shanghai
  schedule:
  - cron: '0 0 6 * * ?'
    minReplicas: 20
    maxReplicas: 25
  - cron: '0 0 19 * * ?'
    minReplicas: 1
    maxReplicas: 9
  template:
    spec:
      scaleTargetRef:
        apiVersion: apps/v1
        name: nginx-deployment
      metrics:
      - type: Resource
        resource:
          name: cpu
          target:
            type: Utilization
            averageUtilization: 50
```

- **运维同学怎么知道这个扩展能力怎么用?**
  - 看 CRD? 看配置文件? 看 …… 文档?
- **扩展能力间出现冲突，导致线上故障**
  - 比如：CronHPA 和 默认 HPA 被同时安装给了同一个应用
  - K8s 扩展能力之间的冲突关系，如何有效管理？如何有效的对运维透出?

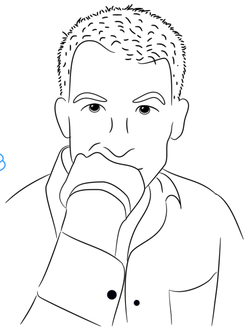
K8s 如何管理描述云资源?

太好了，我还需要启动一个RDS, 能跟 helm一起打包吗?

这样啊，你们PaaS 平台的体验好割裂...

Operator 写好了,用 helm打包部署吧，美滋滋..

这你恐怕得单独去RDS界面创建...



好吧，我们试试 ……

CRD 搞定一切！

```
apiVersion: v1
kind: Application
spec:
  commands:
    stop: ''
    start: 'exec java -Xms1G -Xmx1G -jar s
  package:
    label: v1
    image: itzg/minecraft-server
  rds:
    engineVersion: '1.0.0'
    dbInstanceClass: ''
    databaseName: minecraft
    account: minecraft
    rdsId: 'vvks123s123scdh34flsd4'
    engine: 'MySQL'
    enable: false
    password: ''
    ...
  slb.internet:
    Spec: slb.s1.small
    slbId: '2ze7clg78xsx1g879a5yo'
    protocol: http
    backendPort: 80
    enable: false
    listenerPort: 80
    ...
```

RDS

SLB

```
platform:
  os: linux
  buildpack: Java Tomcat
  category: java
network:
  vpcOption:
    vpcId: vpc-2zed9pncds1131savnry0zm1x8
    vSwitches:
    - vsw-2zeb48r2w7cdjxd4jx62x
  healthCheck:
  path: /
  port: '8080'
  retryCount: 3
  timeoutSeconds: 3
  type: http
  intervalSeconds: 3
  ...
autoScaling:
  scalingPolicy: release
  instanceChargeType: PostPaid
  userData: ''
  instanceNum: 1
  instanceName: craft
  instanceType: []
  internetMaxBandwidthIn: 100
  passwordInherit: false
  systemDiskSize: 100
```

VPC

```
  passwordInherit: false
  systemDiskSize: 100
  internetChargeType: PayByTraffic
  dataDiskInfo: ''
  securityGroupIds: []
  enableInternet: true
  systemDiskCategory: ''
  ...
hooks:
  preStart: ''
  postPrepareApp: ''
  preInstallStack: ''
  postinit: ''
  postPrepareEnv: ''
  postInstallStack: ''
  postStart: ''
  postStop: ''
  prePrepareApp: ''
```

- 对研发屏蔽 K8s API
- 对运维简化 K8s API
- 自由描述非 K8s 资源

每个公司/团队都有自己的应用定义 ……

有赞

下面是一个 Python 应用的 app.yaml 示例文件。

```
stack: youzanyun-centos6
runtime: python-2.7
entrypoint: gunicorn -c gunicorn_config.py wsgi:application
```

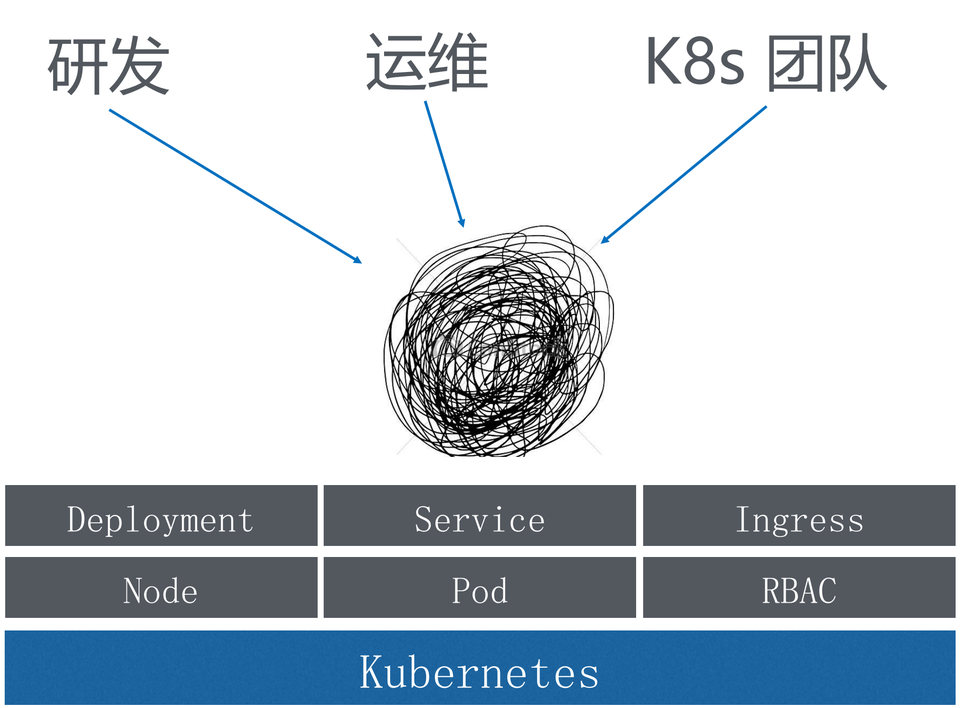
Pinterest

```
apiVersion: pinterest.com/v1
kind: PinterestService
metadata:
  name: exampleservice
  project: exampleproject
  namespace: default
spec:
  iamrole: role1
  loadbalancer:
      port: 8080
  replicas: 3  #Default 1
  sidecarconfig:
    sidecar1:
      deps:
      - example.dep
    sidecar2:
      log_level: info
  template:
    spec:
      initcontainers:
        - name: init
          image: gcr.io/kuar-demo/kuard-amd64:1
      containers:
        - name: exampleservice
          image: gcr.io/kuar-demo/kuard-amd64:1
```

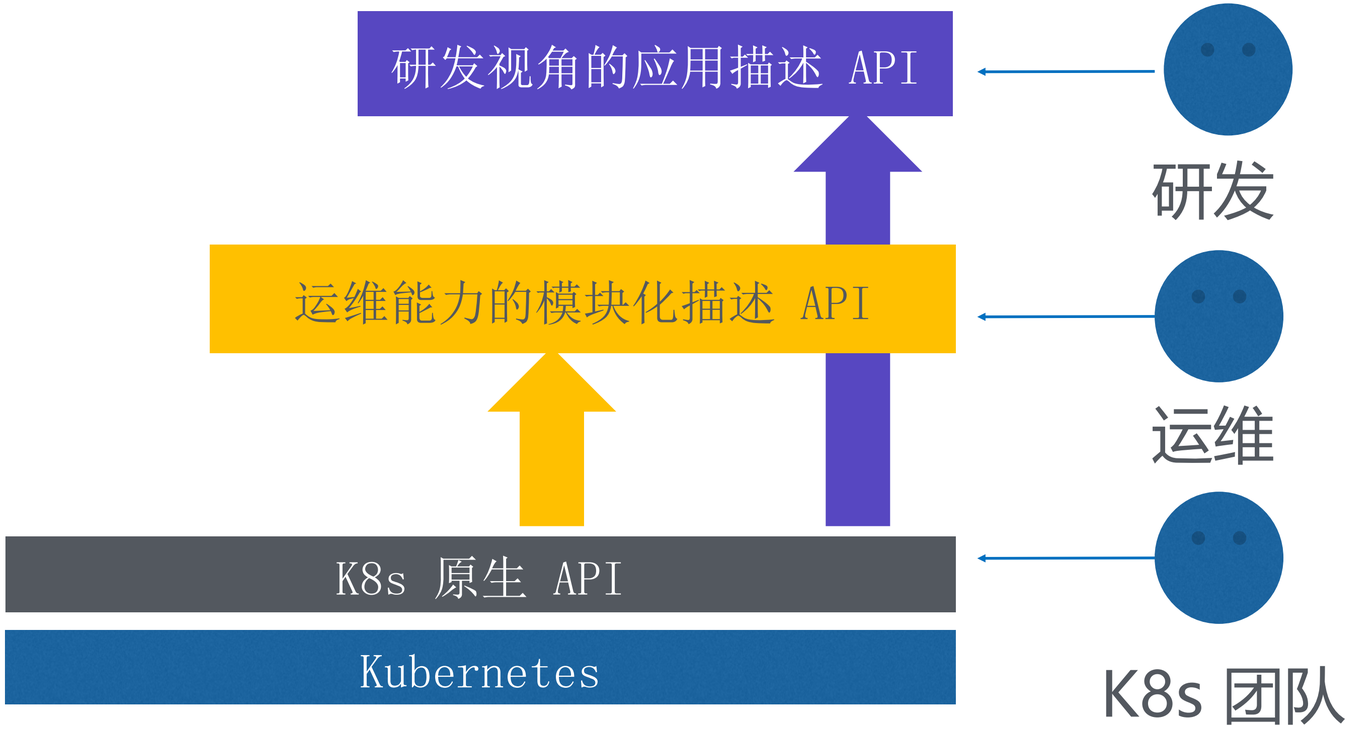
思考题：

- 标准吗？能复用吗？
- 如何与开源生态协作？
- 如何迭代、演进？

Kubernetes API 到底应该怎么玩儿?



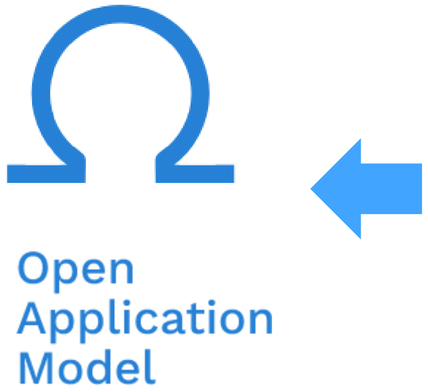
K8s 的 All-in-One API

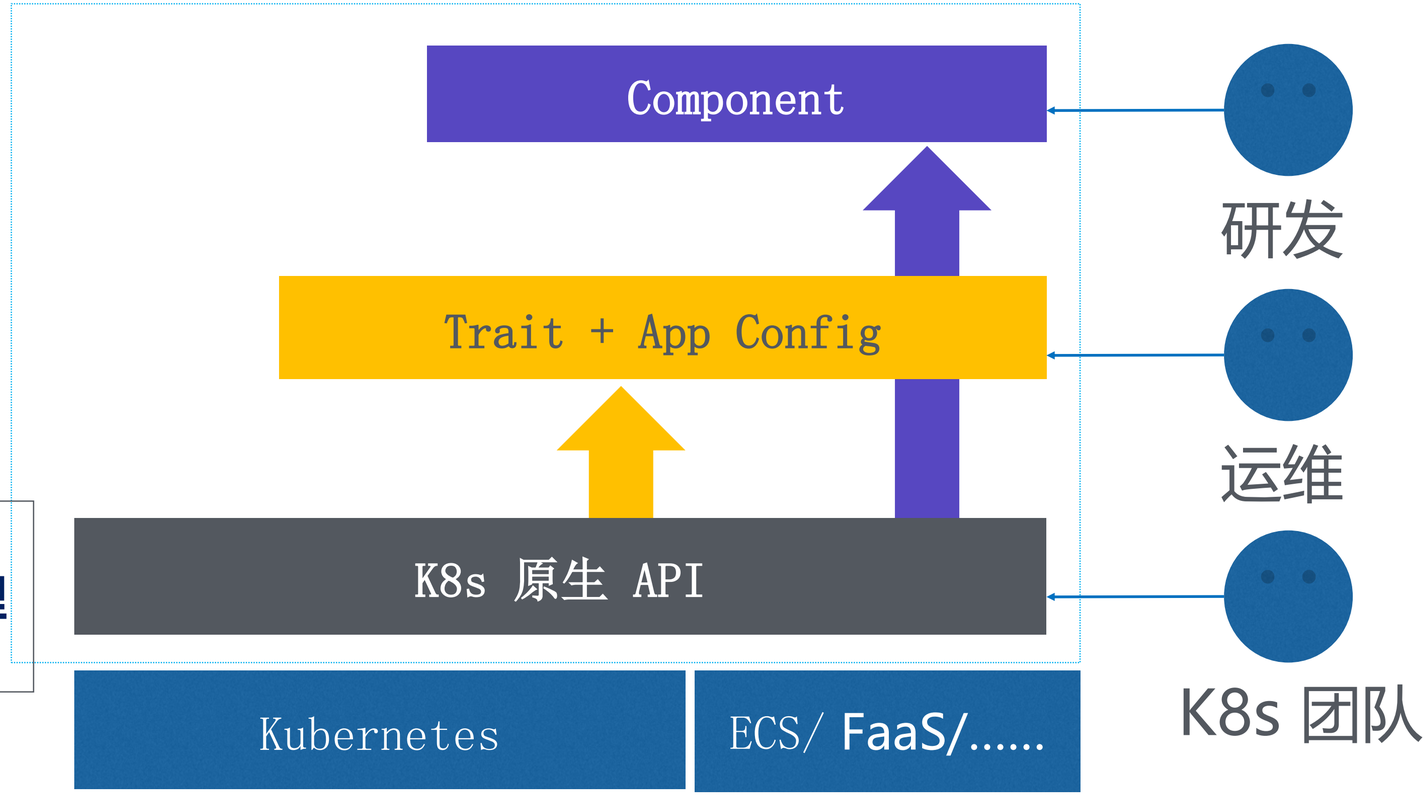


K8s + 分层化 API 设计

区分使用者角色的分层应用定义 + 模块化封装的运维能力 = 应用模型

OAM：以应用为中心的 K8s API 分层模型





- API复杂：区分使用者/关注点分离
- 能力难上手：模块化封装/统一管理
- 云资源：统一API对接

Component

核心workload	可访问	可复制	长久运行
Server	√	√	√
Singleton Server	√	×	√
Worker	×	√	√
Singleton Worker	×	×	√
Task	×	√	×
Singleton Task	×	×	×

```
apiVersion: core.oam.dev/v1alpha1
kind: Component
metadata:
  name: nginx
  annotations:
    version: v1.0.0
    description: >
      Sample component schematic that describes the
administrative interface for our nginx deployment.
spec:
  workloadType: Server
  osType: linux
  containers:
  - name: nginx
    image:
      name: nginx:1.7.9
      digest: <sha256:....>
    env:
    - name: initReplicas
      value: 3
    - name: worker_connections
      fromParam: connections
  parameters:
  - name: connections
    description: "The setting for worker connections"
    type: number
    default: 1024
    required: false
```

1.Description of the application

2. A list of overwritable parameters (schemas)

Component

```
apiVersion: core.oam.dev/v1alpha1
kind: Component
metadata:
  name: nginx
  annotations:
    version: v1.0.0
    description: >
      Sample component schematic that describes the
administrative interface for our nginx deployment.
spec:
  workloadType: Server
  osType: linux
  containers:
  - name: nginx
    image:
      name: nginx:1.7.9
      digest: <sha256:...>
    env:
    - name: initReplicas
      value: 3
    - name: worker_connections
      fromParam: connections
  parameters:
  - name: connections
    description: "The setting for worker connections"
    type: number
    default: 1024
    required: false
```

Operational hint from developers to operators

Reference a overwritable parameter as value

Overwritable parameters (schemas) **list**

从 CRD 到 扩展 Workload

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: functions.openfaas.com
spec:
  group: openfaas.com
  version: v1alpha2
  versions:
    - name: v1alpha2
      served: true
      storage: true
  names:
    plural: functions
    singular: function
    kind: Function
  scope: Namespaced
  validation:
    openAPIV3Schema:
      properties:
        spec:
          properties:
            name:
              type: string
              pattern: "^[a-z0-9]([-a-z0-9]*[a-z0
            image:
              type: string
```

OpenFaaS CRD

```yaml
apiVersion: core.oam.dev/v1alpha1
kind: WorkloadType
metadata:
  name: OpenFaaS
  annotations:
    version: v1.0.0
    description: "OpenFaaS a Workload which can serve workload runni
spec:
  group: openfaas.com
  version: v1alpha2
  names:
    kind: Function
    singular: function
    plural: functions
  workloadSettings: |
    {
      "$schema": "http://json-schema.org/draft-07/schema#",
      "type": "object",
      "required": [
        "name", "image"
      ],
      "properties": {
        "name": {
          "type": "string",
          "description": "the name to the function",
          "pattern": "^[a-z0-9]([-a-z0-9]*[a-z0-9])?$"
        },
        "image": {
          "type": "string",
          "description": "the docker image of the function"
        }
      }
    }
```

OpenFaaS 扩展Workload

可发现、可管理的运维能力：OAM Traits System

```
1 kubectl get traits
2 NAME                AGE
3 cron-scaler         19m
4 auto-scaler         19m
```

发现运维能力

```
$ oamctl trait-list
```

NAME	VERSION	PRIMITIVES
autoscaler	0.1.0	Server, Worker
ingress	0.1.0	SingletonServer,

kubectl get traits crob-scaler –o yaml

查看能力用法

```yaml
apiVersion: core.oam.dev/v1alpha1
kind: Trait
metadata:
  name: cron-scaler
  annotations:
    version: v1.0.0
    description: "Allow cron scale a workloads that allow multiple r
spec:
  appliesTo:
    - core.oam.dev/v1alpha1.Server
  properties: |
    {
      "$schema": "http://json-schema.org/draft-07/schema#",
      "type": "object",
      "required": [
        "schedule"
      ],
      "properties": {
        "schedule": {
          "type": "array",
          "description": "CRON expression for a scaler",
          "item": {
            "type": "string"
          }
        },
        "timezone": {
          "type": "string",
          "description": "Time zone for this cron scaler."
        },
        "resource":{
          "type": "object"
          "description": "Resources the cron scaler will follow",
          "properties": {
             "cpu": {
               type: "object"
               ...
```

kubectl apply -f example.yaml

```yaml
1  apiVersion: core.oam.dev/v1alpha1
2  kind: ApplicationConfiguration
3  metadata:
4    name: failed-example
5  spec:
6    components:
7      - name: nginx-replicated-v1
8        instanceName: example-app
9        traits:
10         - name: auto-scaler
11           properties:
12             minimum: 1
13             maximum: 9
14         - name: cron-scaler
15           properties:
16             timezone: "America/Los_
17             schedule: "0 0 6 * * ?"
18             cpu: 50
```

提前暴露冲突

绑定能力给应用

从 CRD 到 Trait

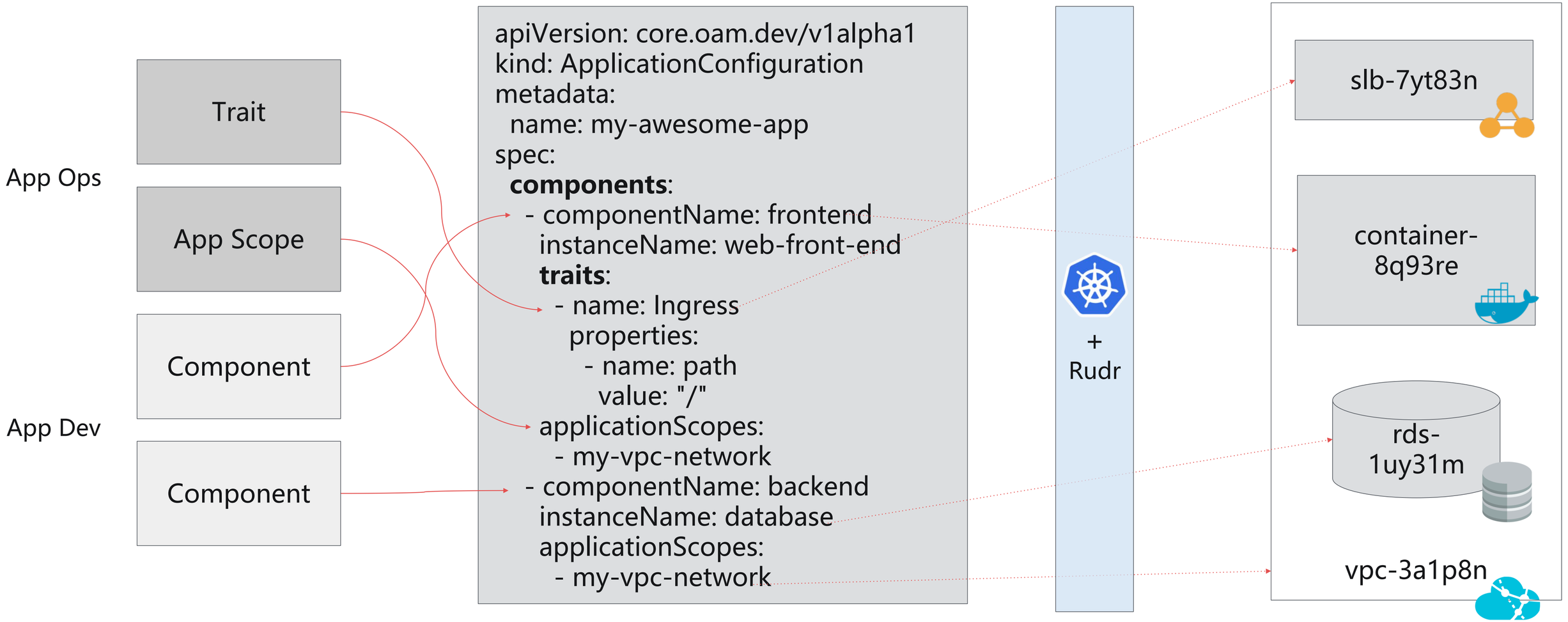
CronHPA CRD

CronHPA Trait

```yaml
apiVersion: "app.alibaba.com/v1"
kind: CronHPA
metadata:
  name: cron-scaler
spec:
  timezone: Asia/Shanghai
  schedule:
  - cron: '0 0 6 * * ?'
    minReplicas: 20
    maxReplicas: 25
  - cron: '0 0 19 * * ?'
    minReplicas: 1
    maxReplicas: 9
  template:
    spec:
      scaleTargetRef:
        apiVersion: apps/v1
        name: nginx-deployment
      metrics:
      - type: Resource
        resource:
          name: cpu
          target:
```

```yaml
apiVersion: core.oam.dev/v1alpha1
kind: Trait
metadata:
  name: cron-scaler
  annotations:
    version: v1.0.0
    description: "Allow cron scale a workloads that allow multiple replica
spec:
  appliesTo:
    - core.oam.dev/v1alpha1.Server
  properties: |
    {
      "$schema": "http://json-schema.org/draft-07/schema#",
      "type": "object",
      "required": [
        "schedule"
      ],
      "properties": {
        "schedule": {
          "type": "array",
          "description": "CRON expression for a scaler",
          "item": {
            "type": "string"
          }
        },
        "timezone": {
          "type": "string",
          "description": "Time zone for this cron scaler."
        },
        "resource":{
          "type": "object"
          "description": "Resources the cron scaler will follow",
          "properties": {
            "cpu": {
              type: "object"
```

Application Configuration：组装与自包含



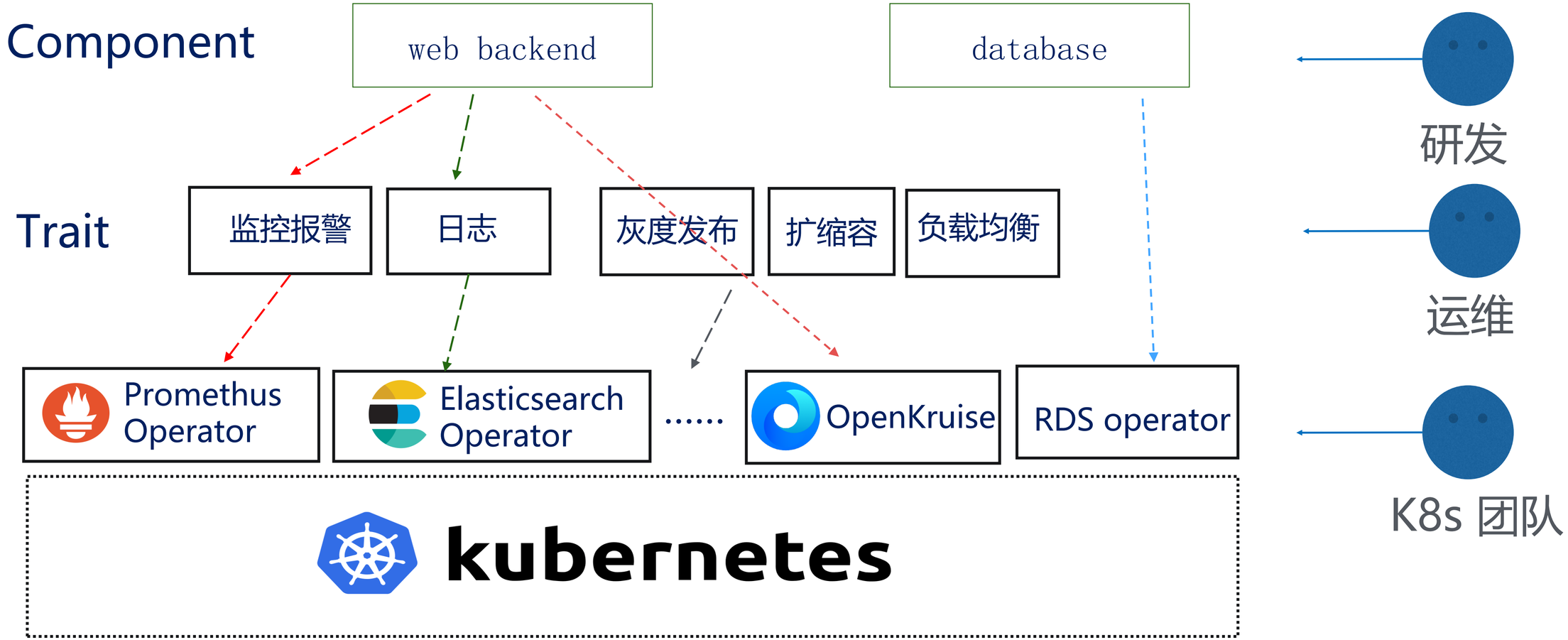
OAM Model

OAM YAML

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

Real-world instances

OAM 加持下的 Kubernetes PaaS



理论基础：CNCF 倡导的“应用交付分层模型”

Topic 1: Application Definition
- app descriptor
- app architecture model

Topic 1.5: Application Packaging
- app packaging
- app parameter & configuration

- OAM
- Helm/CNAB

Topic 2: Application Deploy & Rollout
- app lifecycle mgmt & config src driven workflow
- app rollout strategies: blue-green, canary etc

- GitOps
- Rollout

Topic 3: Workload Instance Automation & Operation
- workload instance healing, scale in/out, sharding
- workload instance lifecycle mgmt

- Workload Controller
- K8s Operators

Topic 4: Platform
- resource mgmt & scheduling
- container lifecycle mgmt, healing and runtime
- networking, logging, monitoring, mesh

- Kubernetes
- FaaS
- Cloud Services

OAM 项目近期计划

- OpenFaas、Terraform、Knative 集成
- K8s Operator 一键接入
- oamctl
- oam-framework
- CRD (traits/workloads) registry
- …

主页：https://oam.dev

规范：https://github.com/oam-dev/spec

实现：https://github.com/oam-dev/rudr

THANK

Global
Architect Summit

欢迎加入阿里 Kubernetes 基础技术中台团队！

jianbo.sjb@alibaba-inc.com

大牛云集、世界级场景、顶级开源项目

OAM交流钉钉群

· 不一样的双11

阿里巴巴经济体云原生实践

- 双11 超大规模 K8s 集群实践中，遇到问题及解决方法详述
- 云原生化最佳组合：Kubernetes+容器+神龙，实现核心系统 100% 上云的技术细节
- 双11 Service Mesh 超大规模落地解决方案

手机识别二维码，直接查看电子书

阿里云

CLOUD NATIVE COMPUTING FOUNDATION

云 原 生 技 术 公 开 课

后云计算时代，技术人员如何

自我提升？

微信扫描二维码直接听课