客如云容器化改造实践

刘凯

客如云容器平台架构师

About Me

- 刘凯
- 客如云Java、容器化架构师
- 客如云 Kubernetes 平台架构、设计工作

目录

一. 客如云容器化背景

店开天下，客如云来

客如云 成立于2012年

是一家面向餐饮、零售、美业等连锁企业

提供软硬一体的SaaS整体解决方案。

客如云的愿景是助力连锁企业“店开天下，客如云来”，

是海底捞、比格比萨、正大集团等集团化连锁企业背后的科技力量。

数说客如云

1	总部
3	中心
130+	直营分公司
500+	认证合作伙伴
4,000+	员工
200,000+	服务商户

容器化目的

- 支撑业务高速增长
- 标准化 -> 自动化 -> 平台化
- Kubernetes 技术红利

二. 容器化落地的挑战

挑战 1 - 保证业务稳定运行

- 不对系统架构做“外科大手术”，以“介入手术”的形式逐步演进
- 按照 “ECS” -> “ECS + Kubernetes 混合部署”-> “Kubernetes” 的部署节奏推进，降低风险
- SVC 采用 LoadBalancer 模式， Nginx Upstream 添加 VIP ，基于 weight 平滑调整流量

挑战 2 - 保证业务迭代速度

不能影响业务线的正常迭代开发，尽量实现容器化无感改造：

- 通用的 Dockerfile 模板
- Jenkins Pipelines 改造
- 部署平台支持 ECS 和 Kubernetes 部署

```
FROM openjdk:8u181

RUN wget -O /usr/local/bin/dumb-init \
    https://github.com/Yelp/dumb-init/releases/download/v1.2.2/dumb-init_1.2.2_amd64 \
    && chmod +x /usr/local/bin/dumb-init

ARG JAR_FILE

ARG JAR_NAME

ENV JAVA_OPTS=""

ENV ENV_OPTS=""

ENV ENV_JAR_NAME=${JAR_NAME}

COPY ${JAR_FILE} ${JAR_NAME}.jar

ENTRYPOINT ["/usr/local/bin/dumb-init", "--"]

CMD java ${JAVA_OPTS} -jar ${ENV_JAR_NAME}.jar ${ENV_OPTS}
```

三. 前期 POC 重点方向

Java 应用容器内存分配

内存实际大小 = 堆内存 + 线程数 * 线程栈大小 + 元空间 + 堆外内存

以 2 GB 的 Heap 为例：-Xmx2048M, -XX:MaxMetaspaceSize=256M, -Xss256k

容器内存大小 = （2048 + 256 + 256） * 1.2 = 3072 MiB = 3GB

JDK 对容器的支持：

jdk8 u131/jdk9

-XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLimitForHeap

jdk11

-XX:+UseContainerSupport

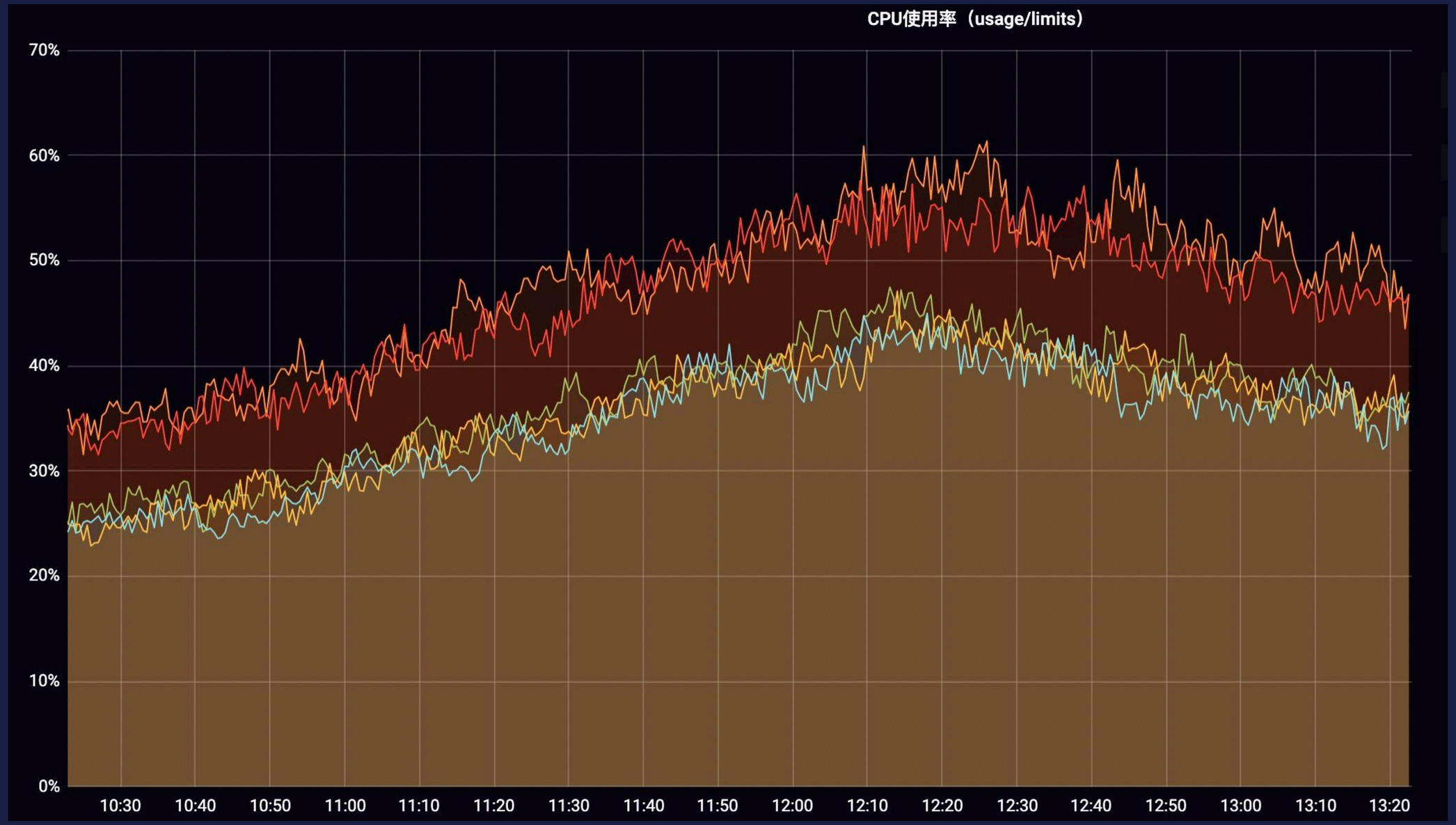
Java 应用容器 CPU 分配

- JVM 启动期间至少需要 2 Cores
- JVM GC 和 CPU 资源正相关
- 默认 CPU Requests = 1 Cores 、Limits = 2 Cores
- 根据业务模式、成本、性能，权衡好 CPU 超卖的比例

Node 节点选型

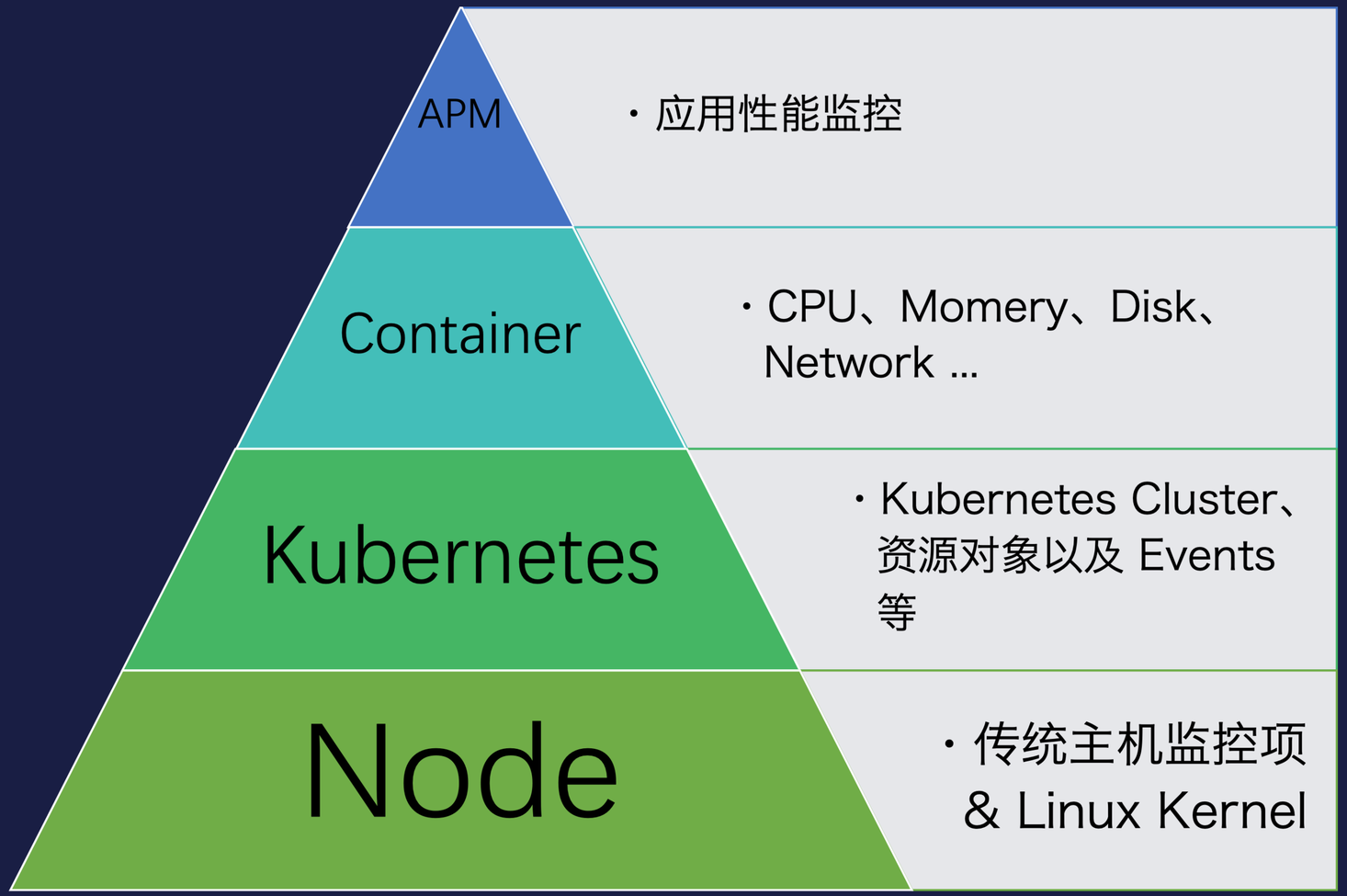
规格	Requests	Limits	CPU 超卖	Pod 数量	说明
96C384G 神龙	1C4G	2C4G	200%	96	最高性价比
96C384G 神龙	0.8C4G	2C4G	250%	110	Pod 数过高
32C128G	1C4G	2C4G	200%	32	过渡期最佳
32C128G	0.8C4G	2C4G	250%	32	资源利用率低
32C256G	0.8C4G	2C4G	250%	40	成本最低
32C256G	0.5C4G	2C4G	400%	64	超卖，低成本

性能对比：神龙 vs ECS



监控

- Kubernetes 集群就绪后第一个应用 - Prometheus
- 线上 Prometheus Operator
- 多维度监控，拒绝盲点

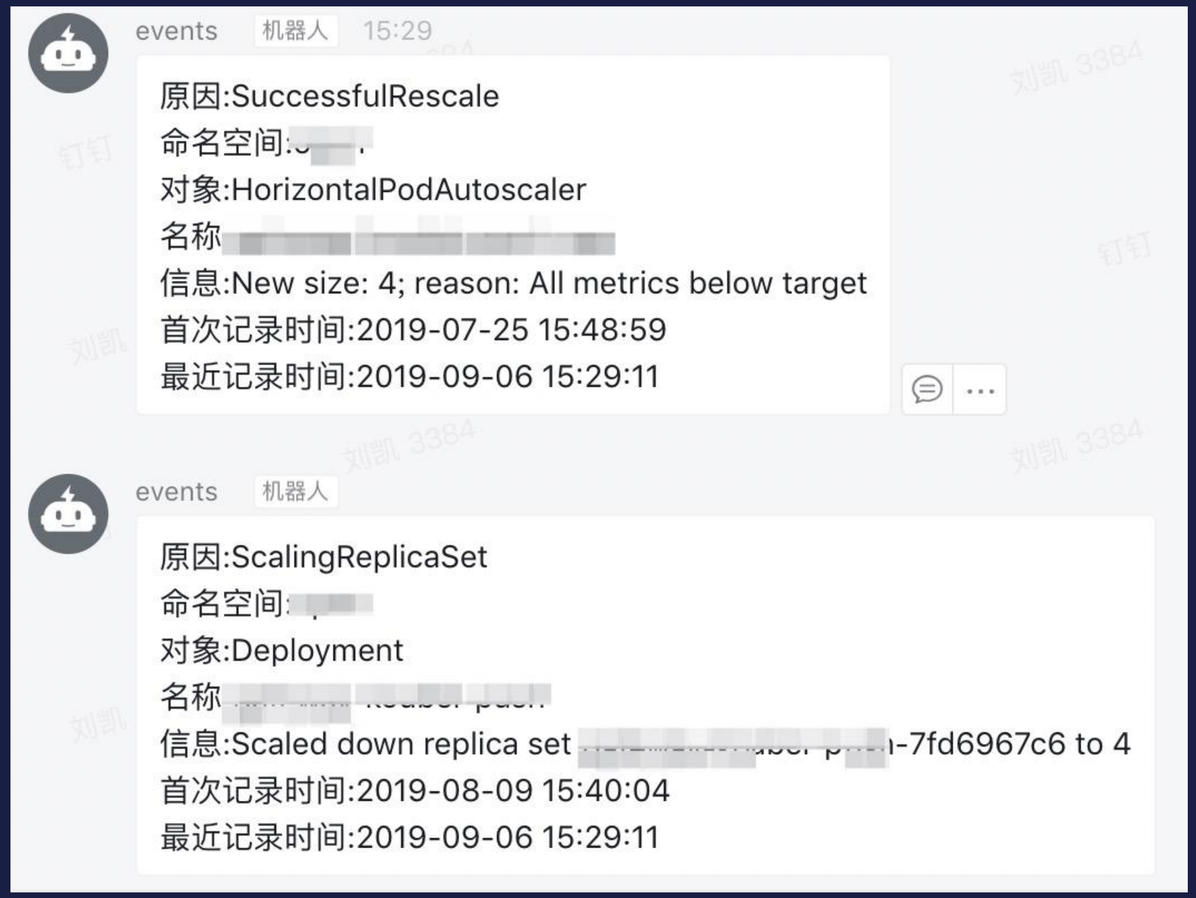


监控 - Node　容器时代 Linux Kernel 的监控

容器实现机制决定了容器和宿主机共享 Linux Kernel，在 Kubernetes 中需要对的 Linux Kernel 进行监控。

监控 - Kubernetes Event 关键信息

- kubernetes/client-go 抓取所有 Events，按需显示 Events
- 专注于关键 Events，提高决策效率

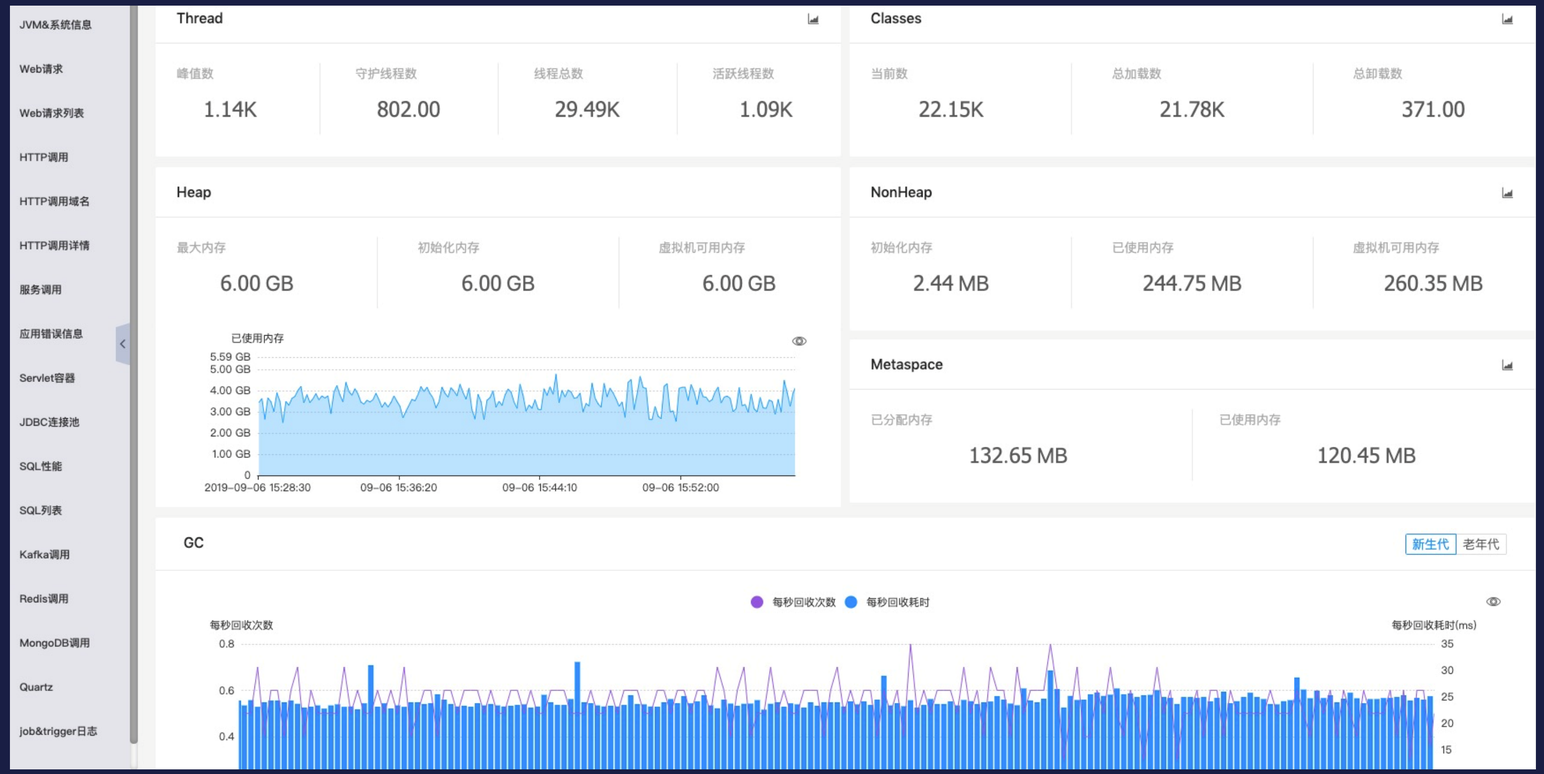


监控 - Container　多维度的监控

容器的资源隔离性，提供了更精确的监控数据，可观察性也随之变得标准化。

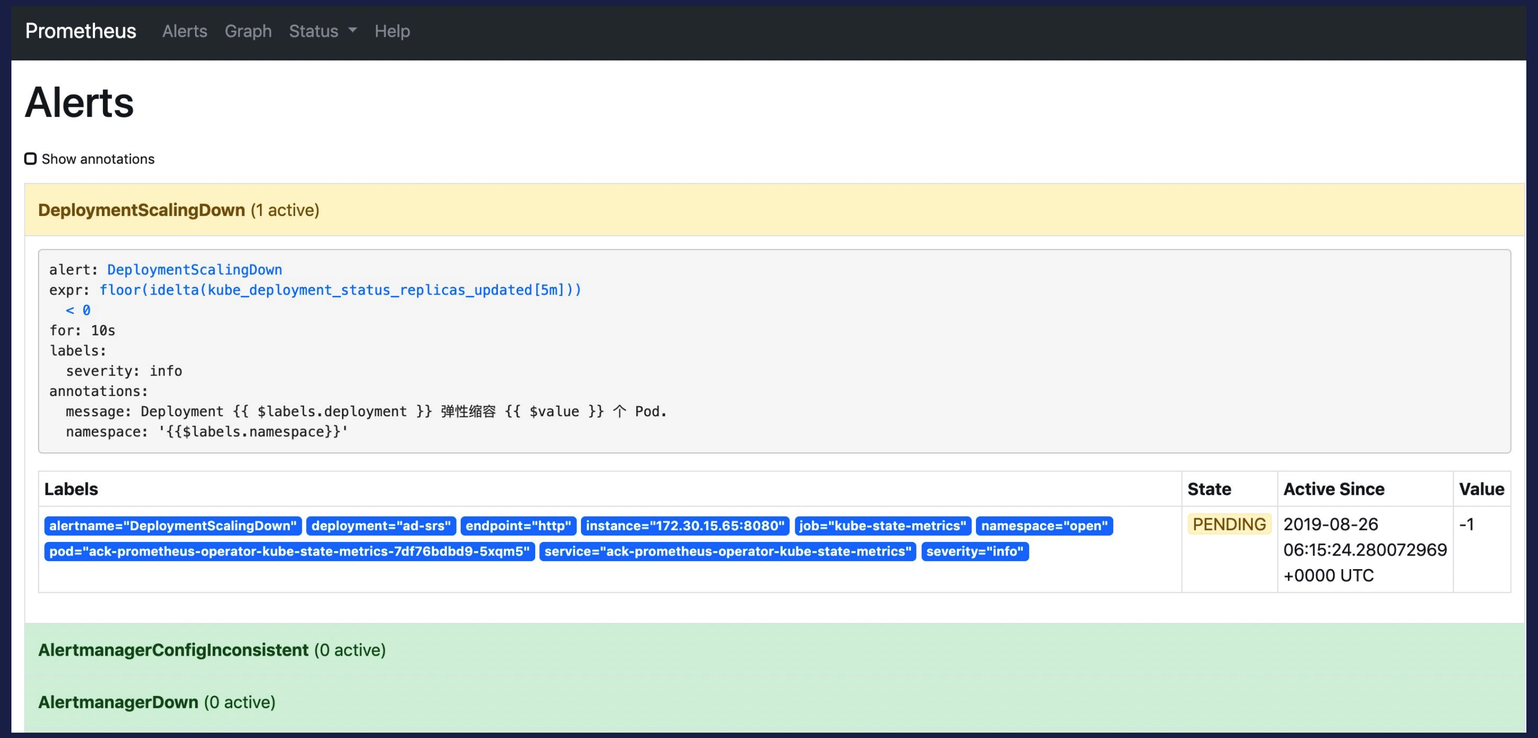
监控 - APM　应用指标监控

通过应用在 ECS 和容器中各项指标对比结果，针对性的调整容器资源分配。



告警 & 通知

- Prometheus Operator 自带众多实用 Rules
- 可自定义 Rules
- 丰富的 Exporter
- 钉钉、邮箱…



日志

- 统一日志采集入口
- 代码保证日志通过 stdout、stderr 输出
- 统一的日志编排模板

```yaml
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  name: [redacted]
  namespace: [redacted]
spec:
  lifeCycle: [redacted]
  logstore: [redacted]
  logtailConfig:
    configName: [redacted]
    inputDetail:
      plugin:
        inputs:
        - detail:
            BeginLineCheckLength: 10
            BeginLineRegex: \d+-\d+
            ExcludeEnv:
              COLLECT_STDOUT_FLAG: "false"
            IncludeEnv:
              CONTAINER_NAME: [redacted]
            IncludeLabel:
              io.kubernetes.pod.namespace: [redacted]
            Stderr: true
            Stdout: true
          type: service_docker_stdout
    inputType: plugin
  machineGroups: null
  project: ""
  shardCount: [redacted]
```

四. 落地过程中的问题

Readiness 和 Liveness 适用场景

- Readiness 检查不过，切断流量
- Liveness 检查不过，容器重启
- Readiness 和 Liveness 是两个独立的逻辑，切忌两套逻辑合并
- 参数需要结合业务场景独立设计，多做沙盘推演

健康检查依赖关系

- Spring Boot ，/health 返回 503
- 仔细评估健康检查依赖，是否影响容器预期运行状态
- 独立于 Kubernetes 外的服务链健康检查机制

Pod 的优雅退出

- exec 启动可以接收 SIGTERM 优雅退出
- sh -c 启动无法实现信号量转发

1. PreStop 添加 kill -15 $PID
2. https://github.com/Yelp/dumb-init

容器化落地心得

- 标准化的思维迭代容器化方案
- 自动化的手段加速容器化落地
- 任何步骤都有风险应对、回滚方案

五. 未来的规划

下一步

- Ingress
- 重构 CI/CD 流程
- Istio
- …

围绕 Kubernetes 沿袭“标准化、自动化、平台化”思路重构现有 Devops 平台，基于新的平台改变研发侧的一些习惯，并把这个价值真正贯穿到整个研发流程中。

Thank you !

诚聘英才

zhaopin_cd@keruyun.com

- Java 架构师、开发
- 高级运维工程师