QCon 全球软件开发大会
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

BEIJING 2017

基于 Mesos 搭建 PaaS 平台你可能需要修的路

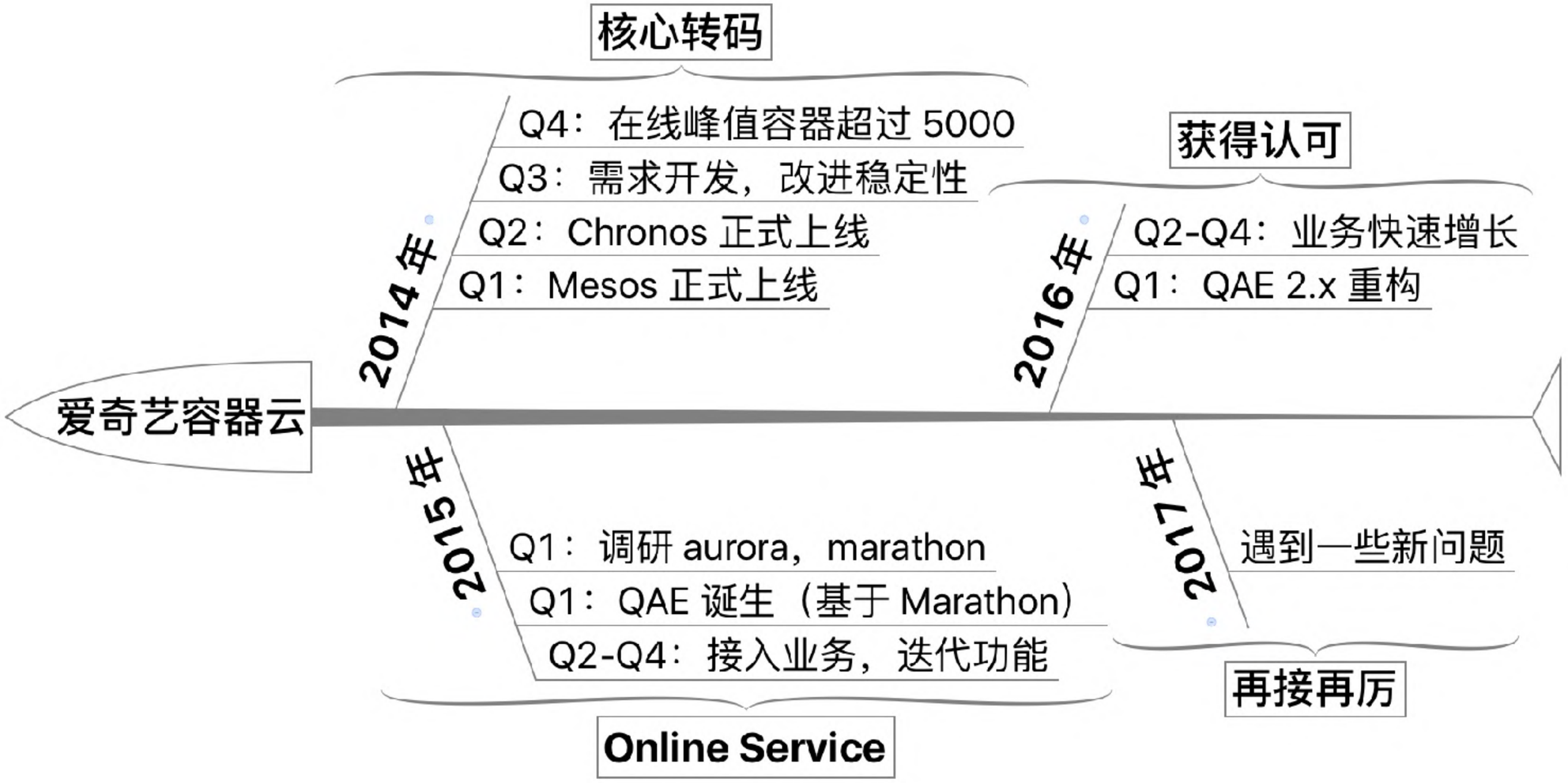
杨成伟@爱奇艺

主办方 Geekbang. 极客邦科技 InfoQ

内容提要

1. 背景介绍
2. QAE（iQIYI App Engine）
3. 延迟发布
4. 平滑升级
5. 灰度发布及 AB 测试
6. 日志
7. 总结及展望

容器云时间线


核心转码
Q4：在线峰值容器超过 5000
Q3：需求开发，改进稳定性
Q2：Chronos 正式上线
Q1：Mesos 正式上线
2014 年
获得认可
Q2-Q4：业务快速增长
Q1：QAE 2.x 重构
2016 年
爱奇艺容器云
2015 年
Q1：调研 aurora，marathon
Q1：QAE 诞生（基于 Marathon）
Q2-Q4：接入业务，迭代功能
Online Service
2017 年
遇到一些新问题
再接再厉

服务质量

业务对服务质量的要求越来越高：

- 延迟发布，充分预热
- 流量缓释，避免容器压力
- 平滑升级，无流量丢失
- 灰度发布，发布过程可控
- AB 测试，能提供有效的对比手段
- 等等

自助排障

用户对自助排障的需求越来越多：

- 监控：系统监控、自定义监控
- 容器健康检查：基于应用指标均值?
- 在线排障：Web 控制台
- 历史容器
- 日志：实时日志、索引日志

内容提要

1. 背景介绍
2. QAE（iQIYI App Engine）
3. 延迟发布
4. 平滑升级
5. 灰度发布及 AB 测试
6. 日志
7. 总结及展望

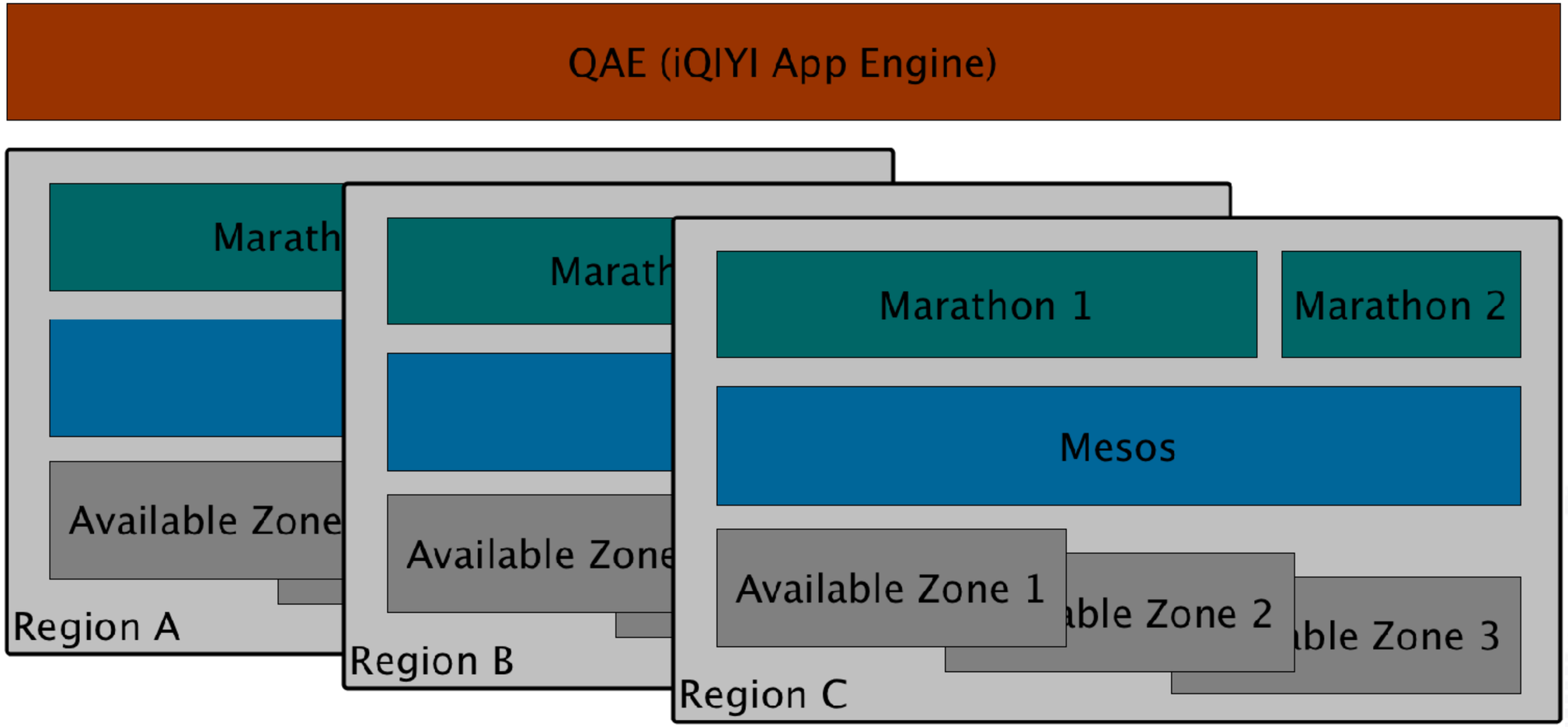
QAE 的基石——Marathon

Marathon 主要特性：

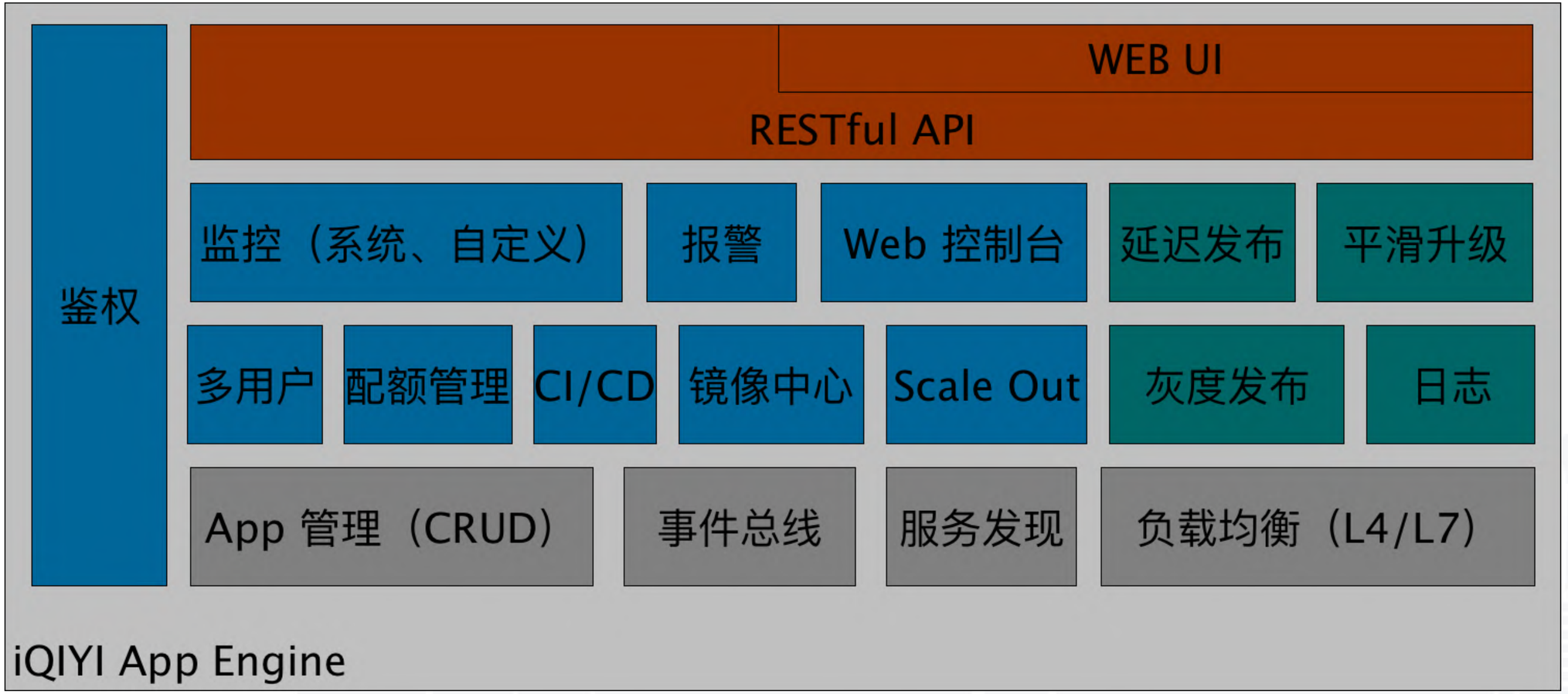
- High Available，高可用性
- 容器分布限制
- Web UI
- RESTful API
- Metrics
- **服务发现和负载均衡**
- **健康检查**
- **事件总线**

- *HA* -- run any number of Marathon schedulers, but only one gets elected as leader; if you access a non-leader, your request gets proxied to the current leader
- *Constraints* - e.g., only one instance of an application per rack, node, etc.
- *Service Discovery & Load Balancing* via HAProxy or the events API (see below).
- *Health Checks*: check your application's health via HTTP or TCP checks.
- *Event Subscription* lets you supply an HTTP endpoint to receive notifications, for example to integrate with an external load balancer.
- *Marathon UI*
- *JSON/REST API* for easy integration and scriptability
- *Basic Auth* and *SSL*
- *Metrics*: query them at `/metrics` in JSON format or push them to graphite/statsd/datadog.

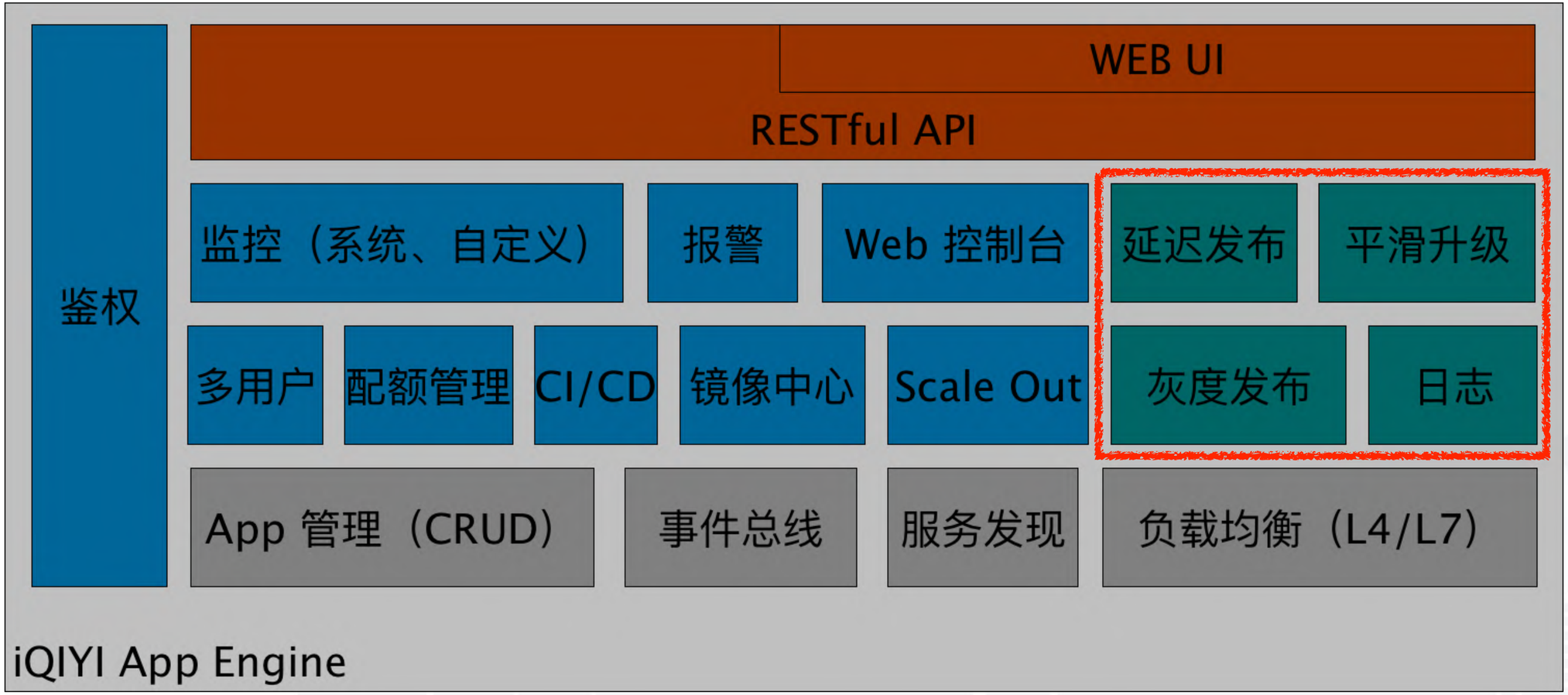
QAE 和 Marathon



QAE 的主要特性


鉴权
WEB UI
RESTful API
监控（系统、自定义）
报警
Web 控制台
延迟发布
平滑升级
多用户
配额管理
CI/CD
镜像中心
Scale Out
灰度发布
日志
App 管理（CRUD）
事件总线
服务发现
负载均衡（L4/L7）
iQIYI App Engine

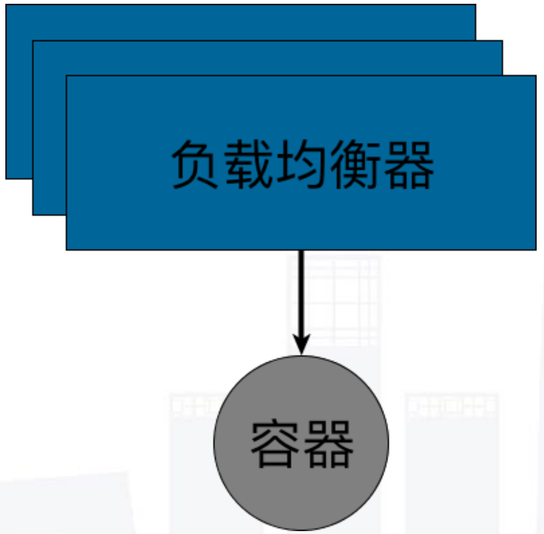
QAE 的主要特性


鉴权
WEB UI
RESTful API
监控（系统、自定义）
报警
Web 控制台
延迟发布
平滑升级
多用户
配额管理
CI/CD
镜像中心
Scale Out
灰度发布
日志
App 管理（CRUD）
事件总线
服务发现
负载均衡（L4/L7）
iQIYI App Engine

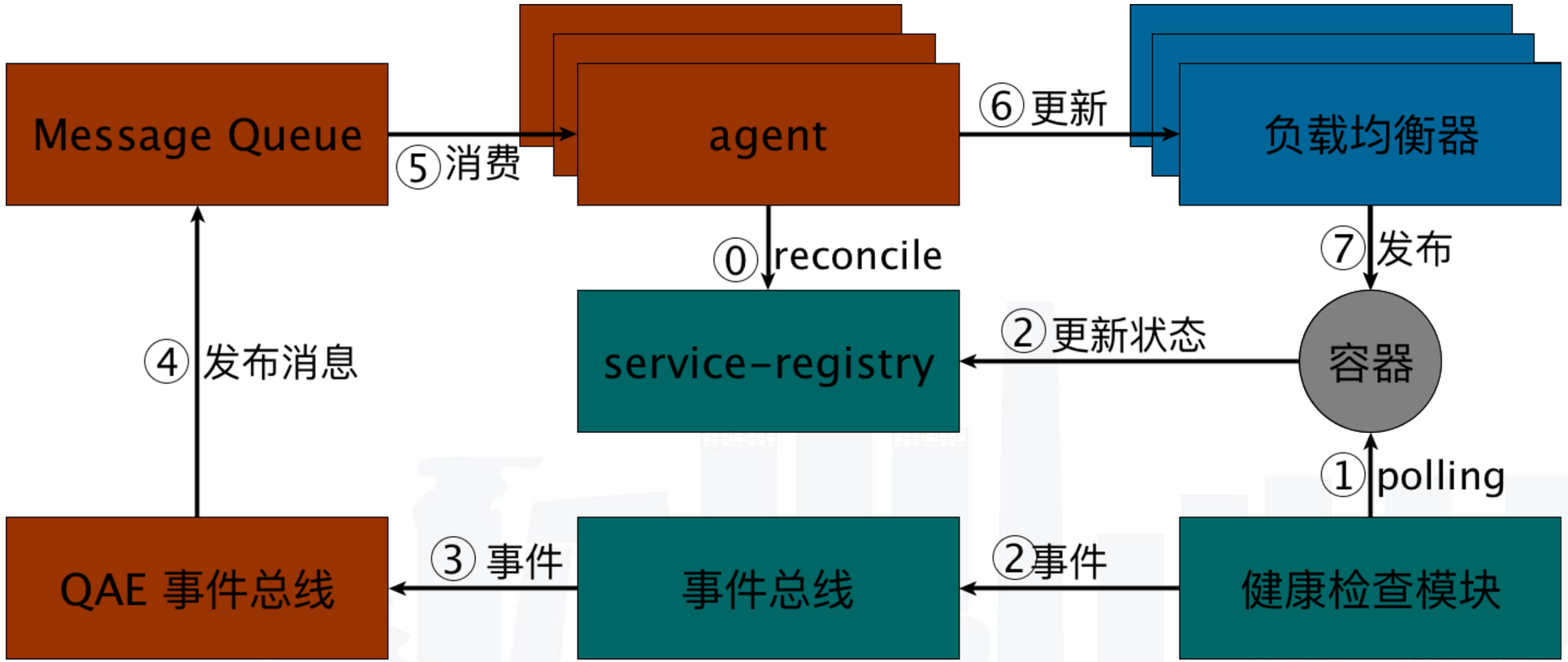
内容提要

1. 背景介绍
2. QAE（iQIYI App Engine）
3. 延迟发布
4. 平滑升级
5. 灰度发布及 AB 测试
6. 日志
7. 总结及展望

服务发现与负载均衡



服务发现与负载均衡


Message Queue
⑤消费
agent
⑥更新
负载均衡器
⓪reconcile
⑦发布
④发布消息
service-registry
②更新状态
容器
①polling
QAE 事件总线
③事件
事件总线
②事件
健康检查模块

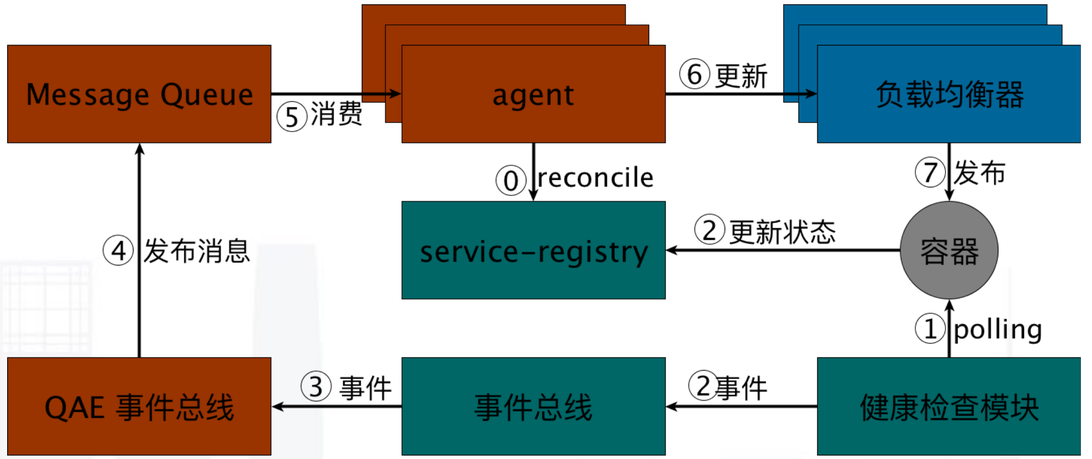
容器发布

Marathon：

- 能够通过 mesos 及时获得容器的状态更新
- 灵活、可配置的健康检查机制

QAE：

- 可配置的**最小**发布延时
- **只发布**健康的容器
- 流量缓释



可配置的最小发布延时

Marathon gracePeriodSeconds：

- 在这段时间内，忽略健康检查失败
- 但是容器一旦健康，立即发布
- 依赖健康检查准确性（扔给用户）

QAE gracePeriodSeconds：

- 容器启动后，**至少**等待这么长时间
- 然后检查容器健康状态，健康则发布，失败则重启

只发布健康的容器

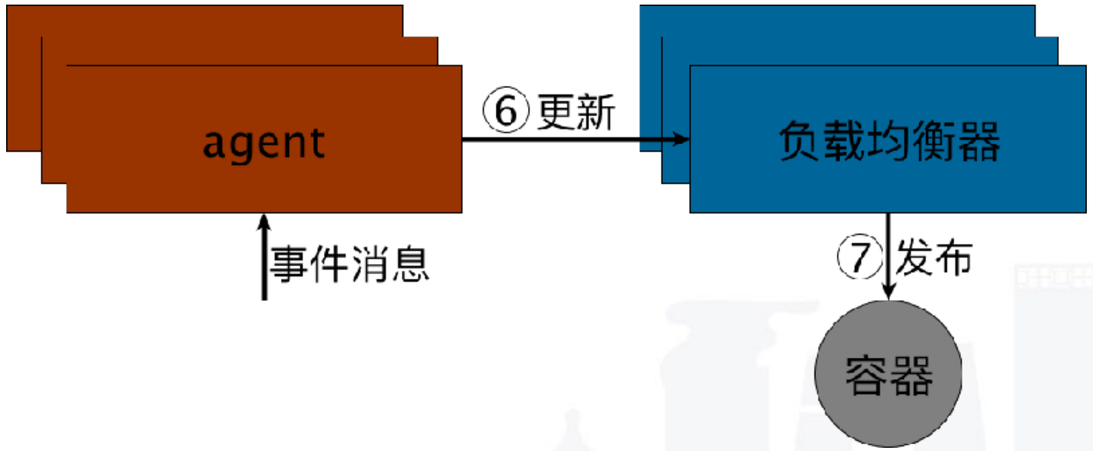
Marathon：

- 虽然，能标记容器的健康状态
- 但是，没有持久化容器的健康状态
- 切换 Leader，**BANG!**

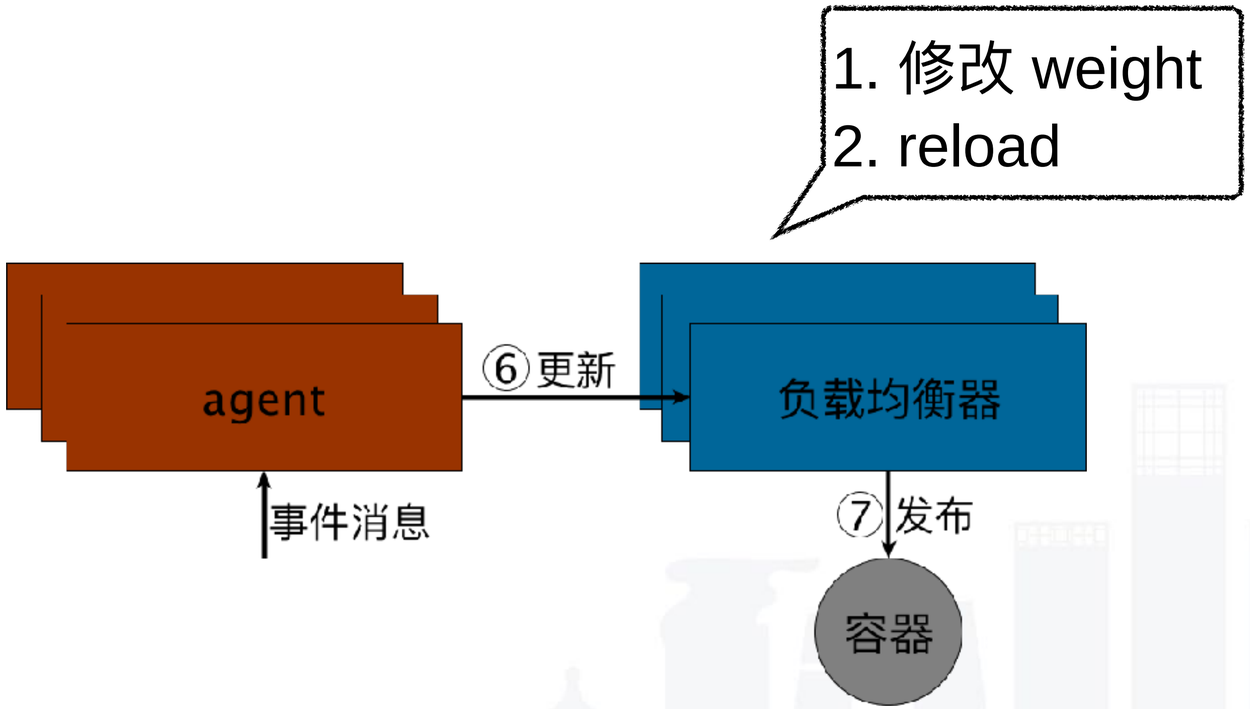
QAE：

- 持久化容器的最近一次健康检查结果

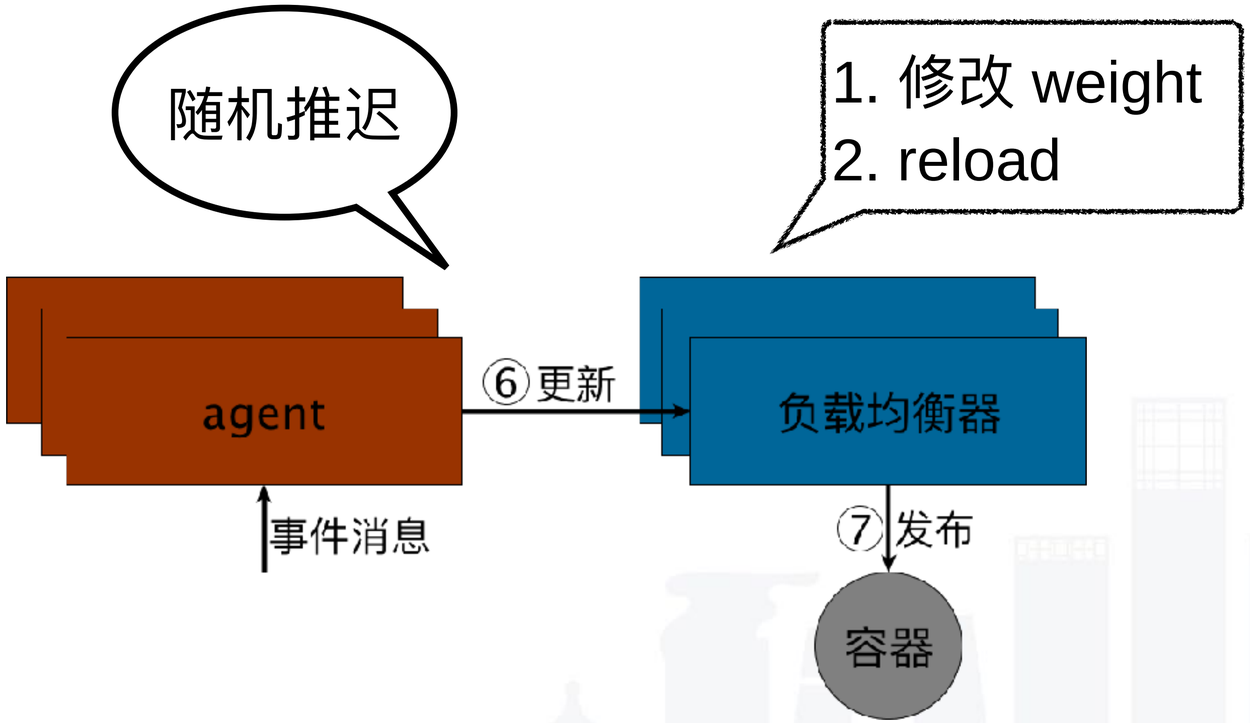
流量缓释



流量缓释



流量缓释



内容提要

1. 背景介绍
2. QAE（iQIYI App Engine）
3. 延迟发布
4. 平滑升级
5. 灰度发布及 AB 测试
6. 日志
7. 总结及展望

Marathon——滚动升级

升级过程：

1. 根据配置和资源启动一批新容器
2. 新容器启动或健康后，停止旧容器
3. 重复上述过程至所有容器升级完成

不足：

- 新容器一旦健康，立即发布
- **杀掉旧容器前不能先将其下线**
- **升级过程不可控，不支持灰度发布**

Marathon——滚动升级

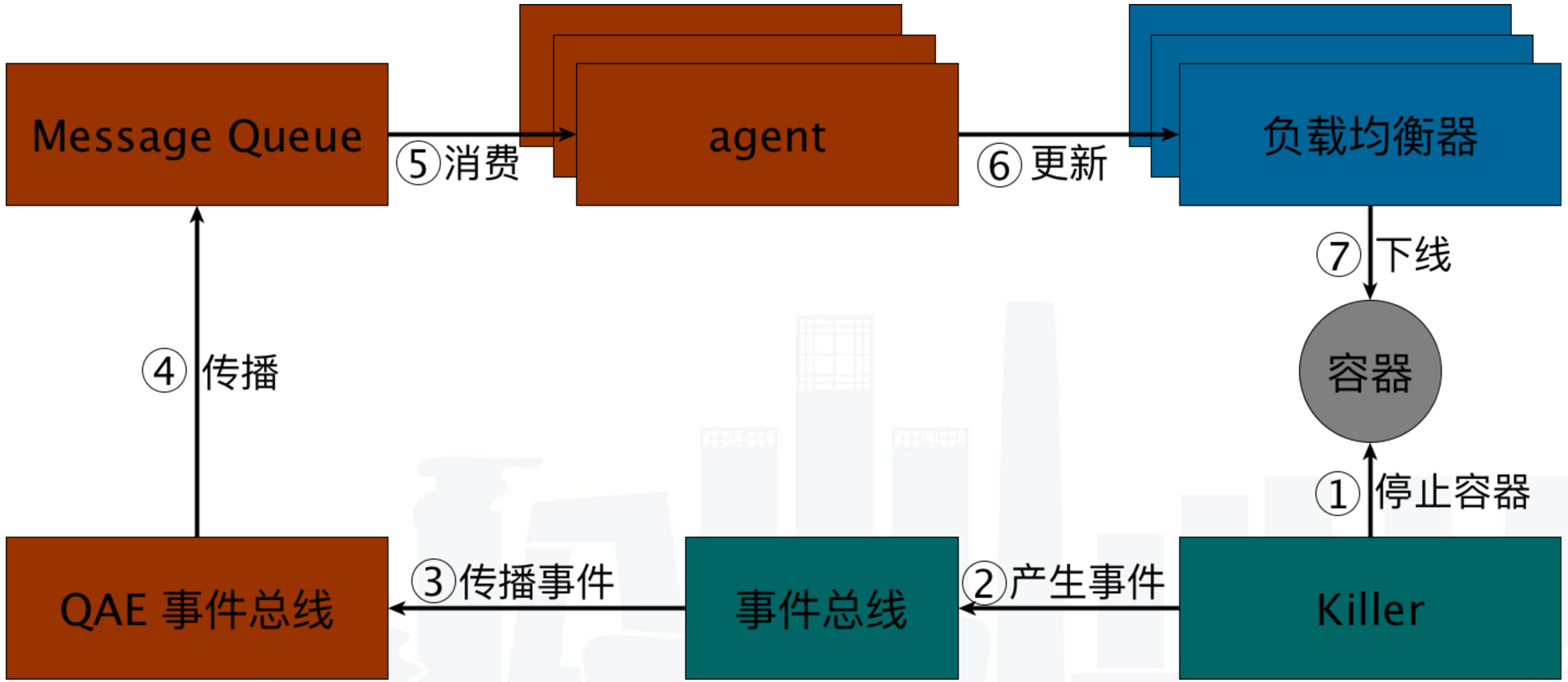
升级过程：

1. 根据配置和资源启动一批新容器
2. 新容器启动或健康后，停止旧容器
3. 重复上述过程至所有容器升级完成

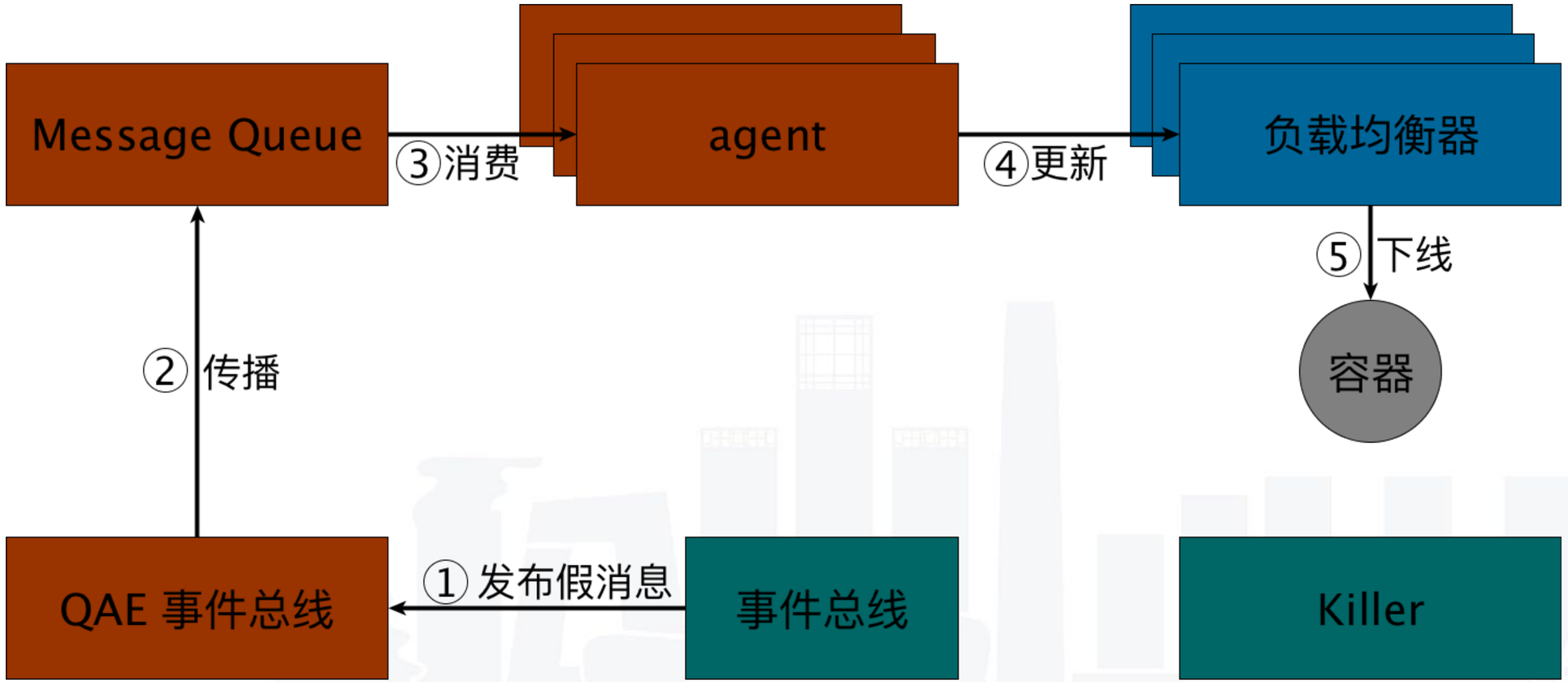
不足：

- 新容器一旦健康，立即发布
- **杀掉旧容器前不能先将其下线**
- **升级过程不可控，不支持灰度发布**

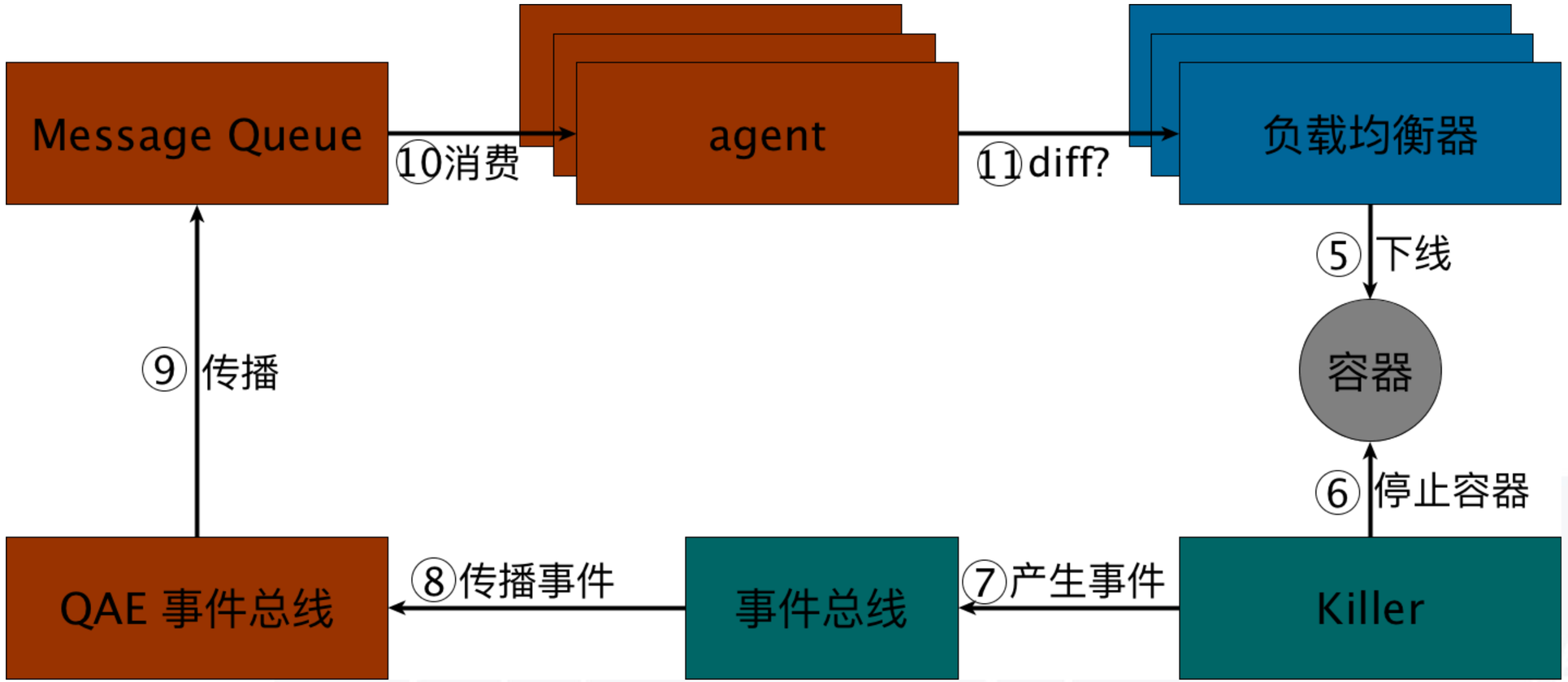
先斩后奏


Message Queue
⑤消费
agent
⑥更新
负载均衡器
⑦下线
容器
④传播
①停止容器
QAE 事件总线
③传播事件
事件总线
②产生事件
Killer

先奏后斩


Message Queue
③消费
agent
④更新
负载均衡器
⑤下线
容器
②传播
QAE 事件总线
① 发布假消息
事件总线
Killer

先奏后斩


Message Queue
⑩消费
agent
⑪diff?
负载均衡器
⑤下线
容器
⑥停止容器
⑨传播
QAE 事件总线
⑧传播事件
事件总线
⑦产生事件
Killer

先奏后斩

能够处理所有可预知的容器下线：

- 平滑升级、灰度发布及 AB 测试
- 手动停止、重启**指定**容器
- 手动/自动 Scale out（**随机**容器）

不能处理不可预知的容器下线：

- failover

内容提要

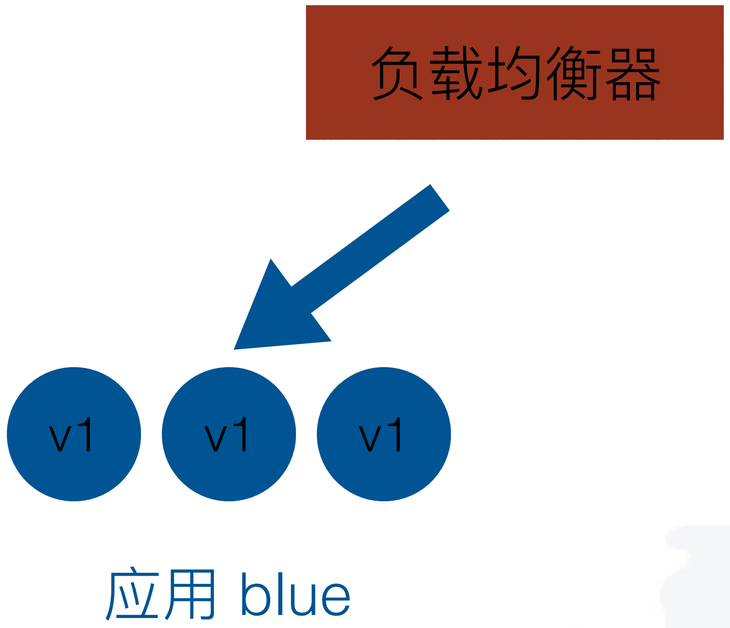
1. 背景介绍
2. QAE（iQIYI App Engine）
3. 延迟发布
4. 平滑升级
5. **灰度发布及 AB 测试**
6. **日志**
7. **总结及展望**

灰度发布及 AB 测试

我们有什么?

我们缺什么?

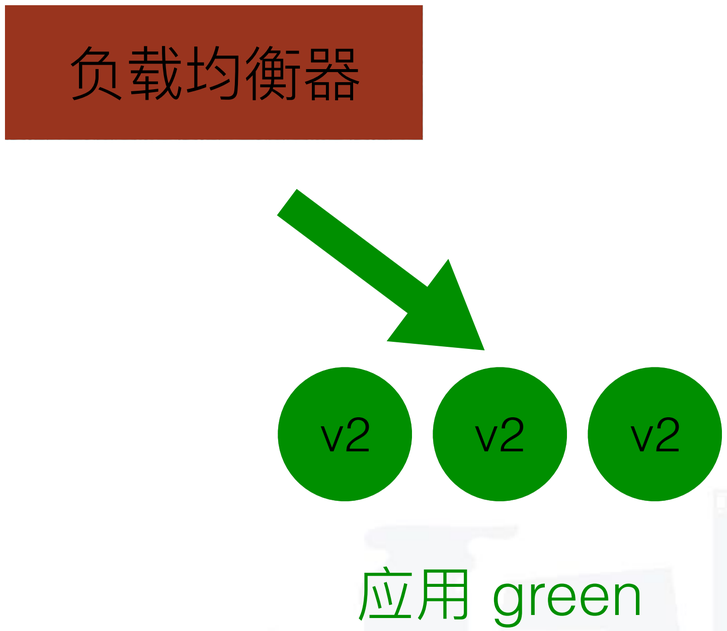
Blue-Green Deployment



看着很眼熟?

- 这不是 marathon rolling upgrade
- 需要**两个** marathon 应用来实现
- 分别控制应用 blue 和 green 的容器增减来实现

Blue-Green Deployment



看着很眼熟?

- 这不是 marathon rolling upgrade
- 需要**两个** marathon 应用来实现
- 分别控制应用 blue 和 green 的容器增减来实现

灰度发布及 AB 测试

直接修改 marathon?

- 社区很早就有类似需求[jira-4396]
- 现有代码架构实现起来很复杂

QAE 的实现

- 基于 blue-green deployment
- 保持向后兼容（backward compatibility）
- 对用户暴露**一个**应用，**两个**版本
- 独立的监控（AB 测试）
- 独立的容器列表（kill，restart）

内容提要

1. 背景介绍
2. QAE（iQIYI App Engine）
3. 延迟发布
4. 平滑升级
5. 灰度发布及 AB 测试
6. **日志**
7. **总结及展望**

stdout & stderr

- 日志散落在 /data/app/log 目录下
- foo.log，bar.log，deadbeef.log
- 为什么要改成 stdout，stderr?
- 你帮我改?

日志需求

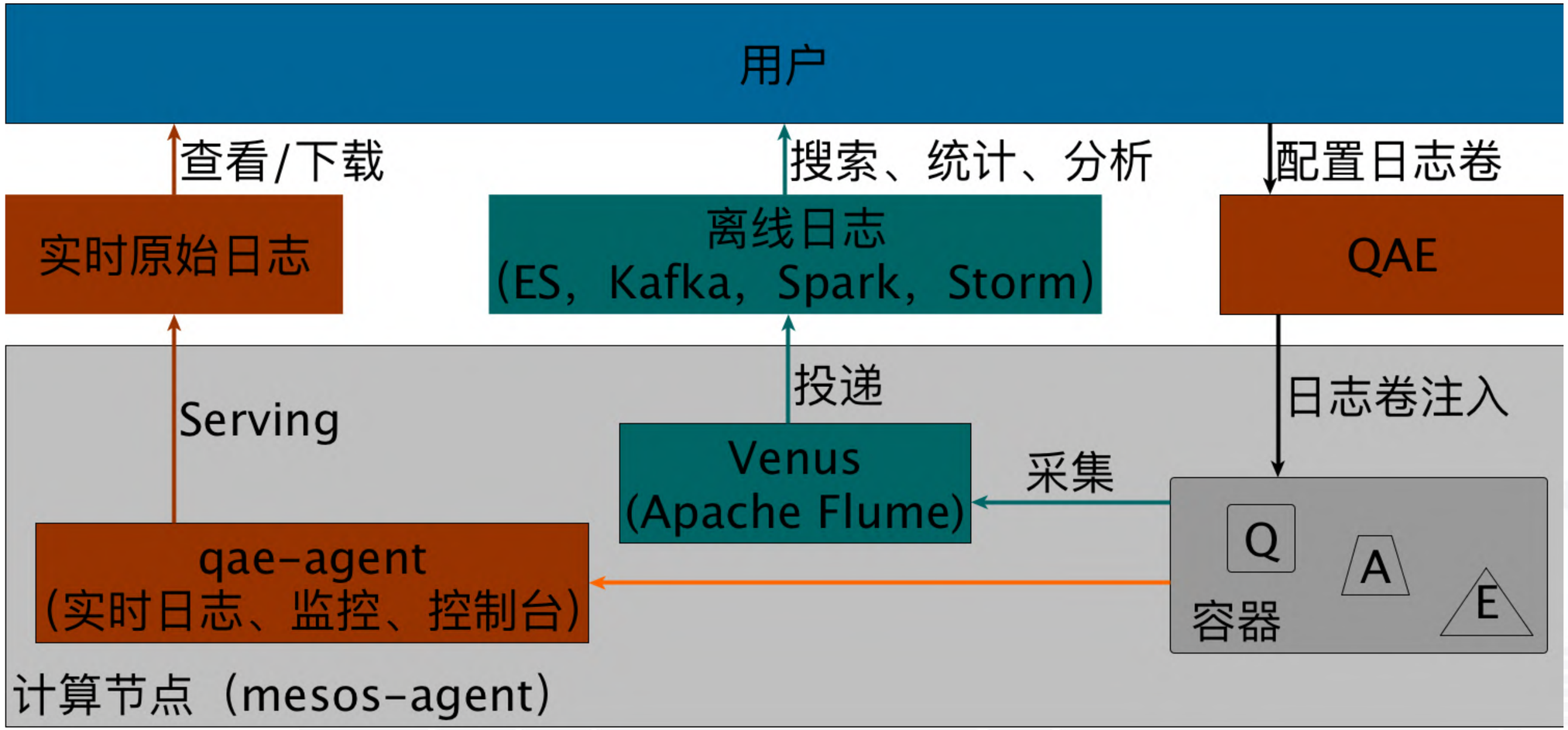
强需求：

- **实时**日志，业务故障了，火烧眉毛
- **原始**日志，怀疑日志搬运工，是不是搬错了
- 可查看，下载，方便快速定位，一眼看不出来再用脚本查找

弱需求：

- 聚合
- 索引、搜索：标签、全文
- 统计、分析
- 呈图、报表

QAE 的日志架构


用户
查看/下载
搜索、统计、分析
配置日志卷
实时原始日志
离线日志
(ES，Kafka，Spark，Storm)
QAE
Serving
投递
日志卷注入
Venus
(Apache Flume)
采集
qae-agent
(实时日志、监控、控制台)
Q
A
E
容器
计算节点（mesos-agent）

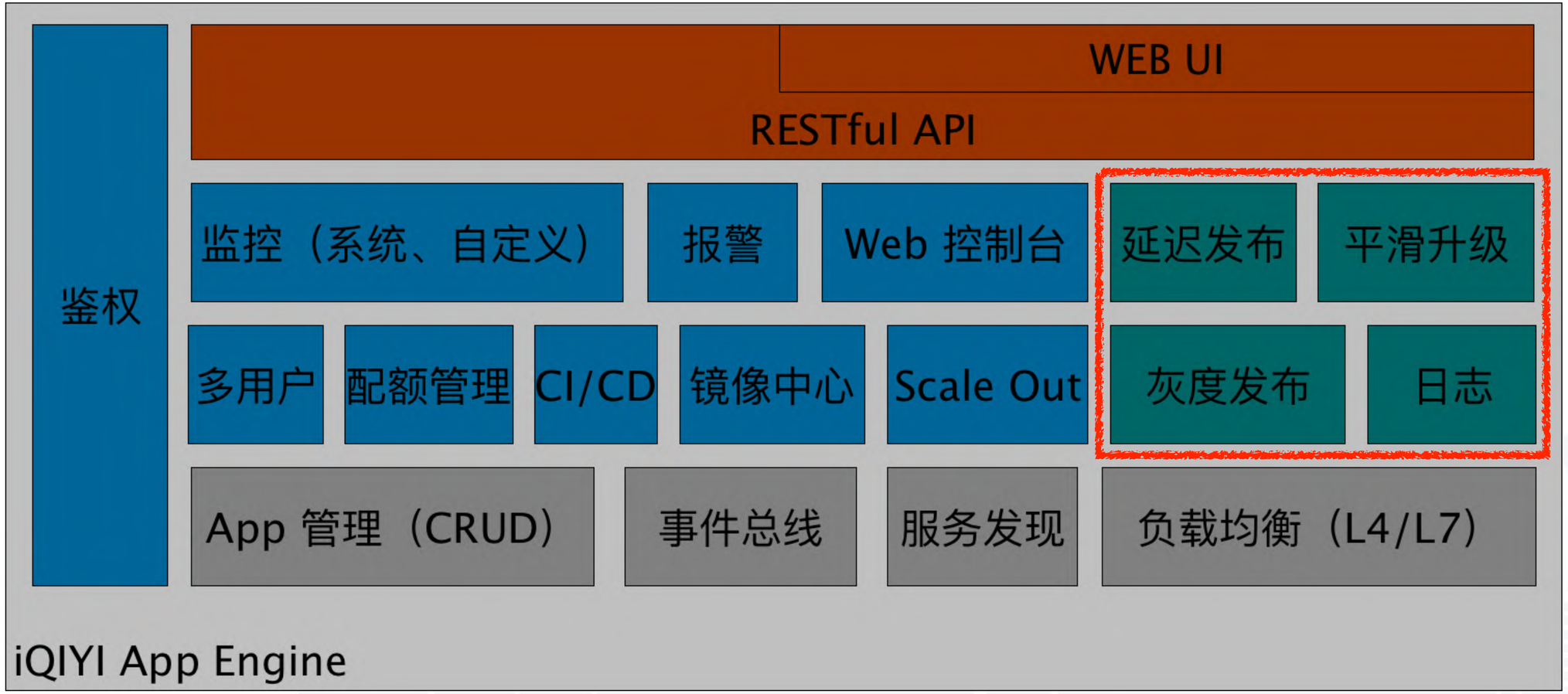
卷注入


主机路径前缀
容器 ID（唯一）
卷类型
/data/qae/injected-volumes/app-name/container-uuid/log : /qae/log
应用名（唯一）
容器内路径（可配置）

内容提要

1. 背景介绍
2. QAE（iQIYI App Engine）
3. 延迟发布
4. 平滑升级
5. 灰度发布及 AB 测试
6. 日志
7. **总结及展望**

总结


鉴权
WEB UI
RESTful API
监控（系统、自定义）
报警
Web 控制台
延迟发布
平滑升级
多用户
配额管理
CI/CD
镜像中心
Scale Out
灰度发布
日志
App 管理（CRUD）
事件总线
服务发现
负载均衡（L4/L7）
iQIYI App Engine

展望

服务中心

分布式健康检查

API Gateway

oversubscription

隔离性

cpuset

混合部署

Mesos Unified Containerizer

存储

虚拟化

网络

关注QCon微信公众号，
获得更多干货！

Thanks!

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

主办方 Geekbang. 极客邦科技 InfoQ