KUBERNETES COMMUNITY DAYS BEIJING 2025

Rokid: 在 AI+AR 场景下 Serverless 容器化技术探索与实践

李鹏 (阿里云)

朱炜栋 (Rokid)

KCD
GROWING CLOUD NATIVE TOGETHER
BEIJING

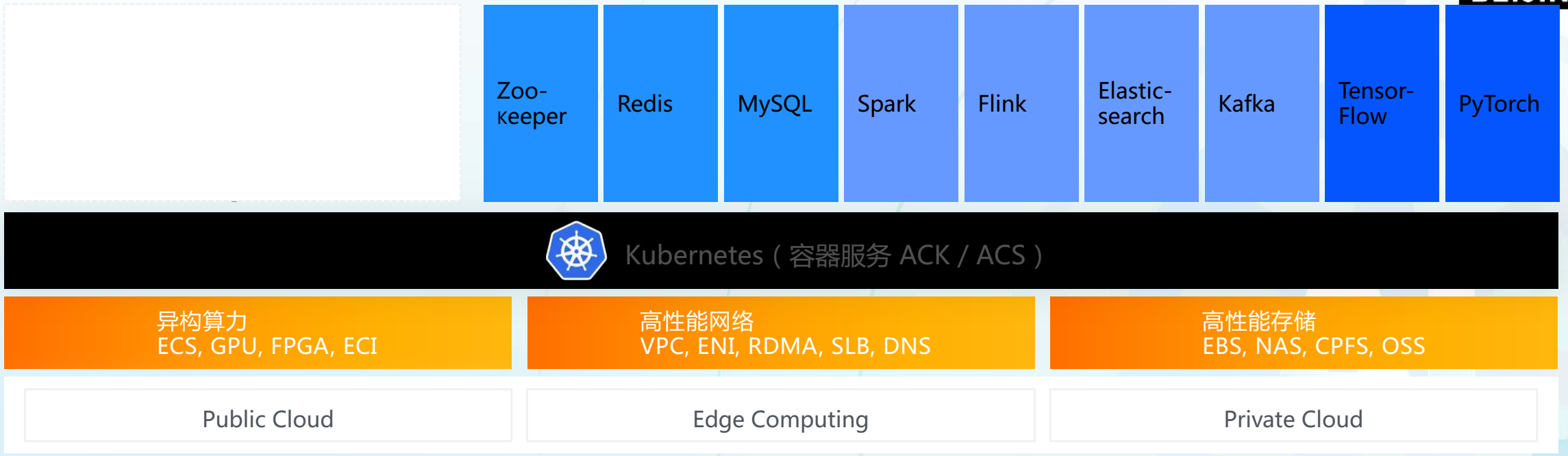
Content

目录

Part 01

Knative 介绍

Kubernetes 正成为数字化、智能化应用的云原生基础设施



Gartner预测，“到2027年，中国的全部AI推理工作负载中，基于云的工作负载占比将从前的20%上升至80%”

图 11 企业关注的云管理服务能力

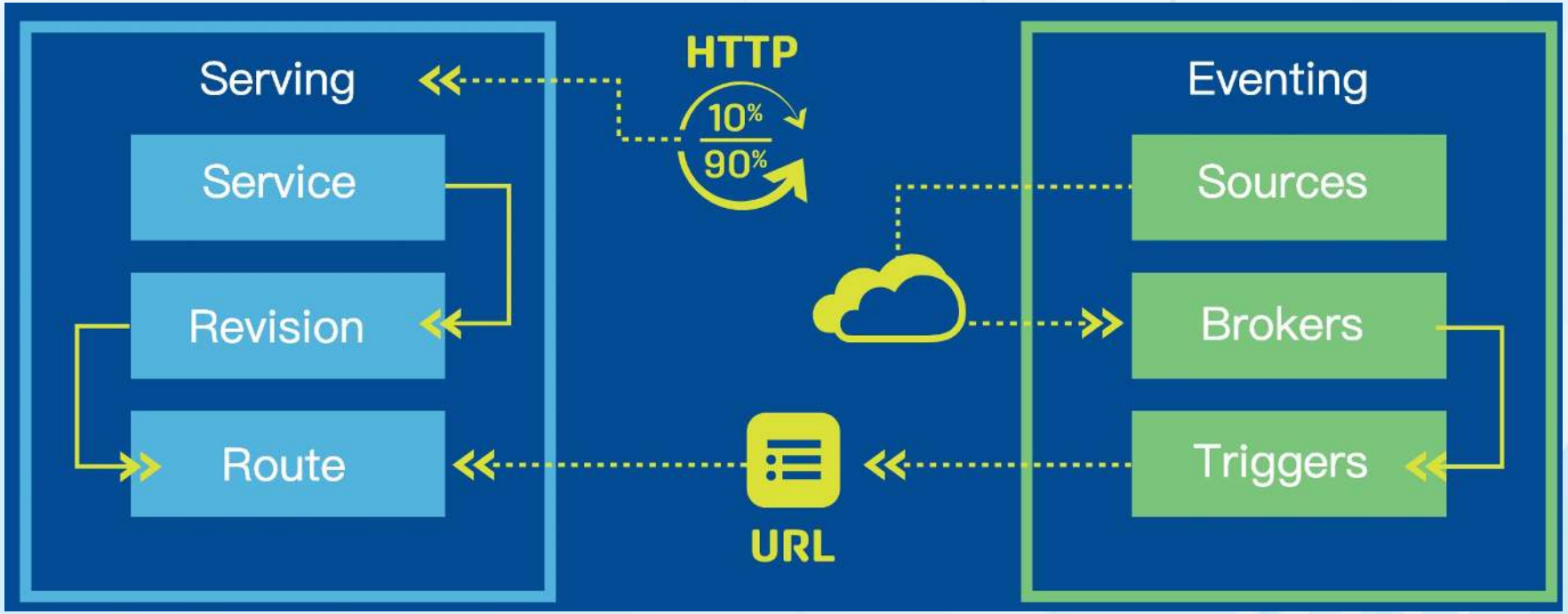
Knative 发展轨迹及阿里云对Knative的支持历程

Knative

功能丰富: Knative 是基于 Kubernetes 之上提供的一款开源 Serverless 应用框架，目标打造企业级 Serverless 应用平台。

最受欢迎: 跟据 2020 CNCF 云原生调查报告，Knative 已成为开源自建 Serverless 平台首选。

云产商共识: 支持或者集成 Knative, 如阿里云、谷歌云、IBM、Red Hat等，并且大部分都提供了生产级别能力。



- 版本管理
- 请求自动弹性
- 缩容到0
- 流量灰度发布
- 事件驱动

Knative应用模型

- **Service**

Serverless 应用的抽象，通过 Service 管理应用的生命周期

- **Configuration**

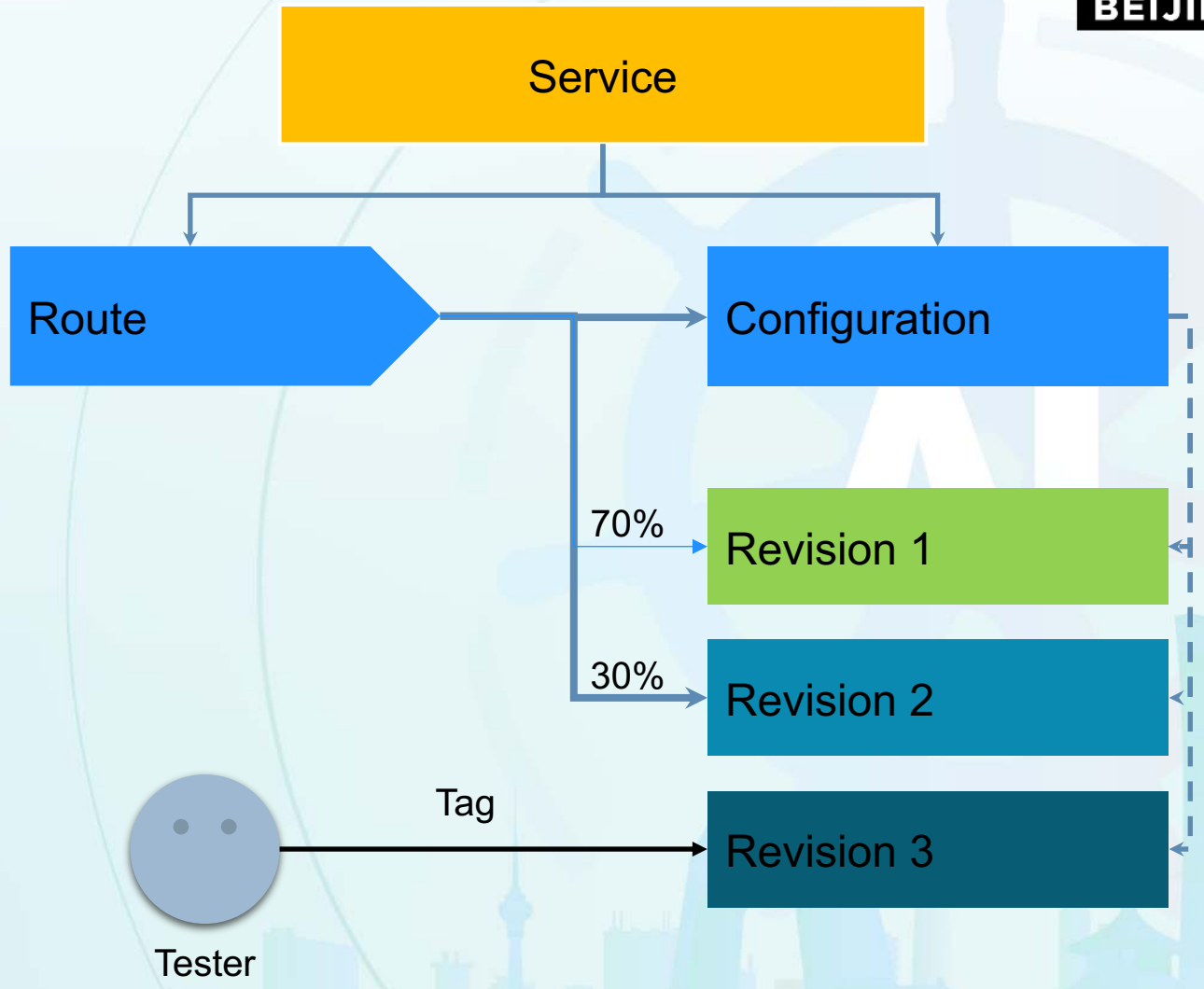
当前期望状态的配置。每次更新 Service 就会更新 Configuration

- **Revision**

Service 的每次更新都会创建一个快照，一个快照就是一个 Revision

- **Route**

将请求路由到 Revision，并可以向不同的 Revision 转发不同比例的流量



版本管理特性

金丝雀验证

Knative 中提供了tag机制可以做到。也就是对某个版本打上tag， 然后 Knative 会自动生成改tag的访问地址。

```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: example-service
  namespace: default
spec:
...
  traffic:
  - percent: 0
    revisionName: example-service-2
    tag: staging
  - percent: 100
    revisionName: example-service-1
```

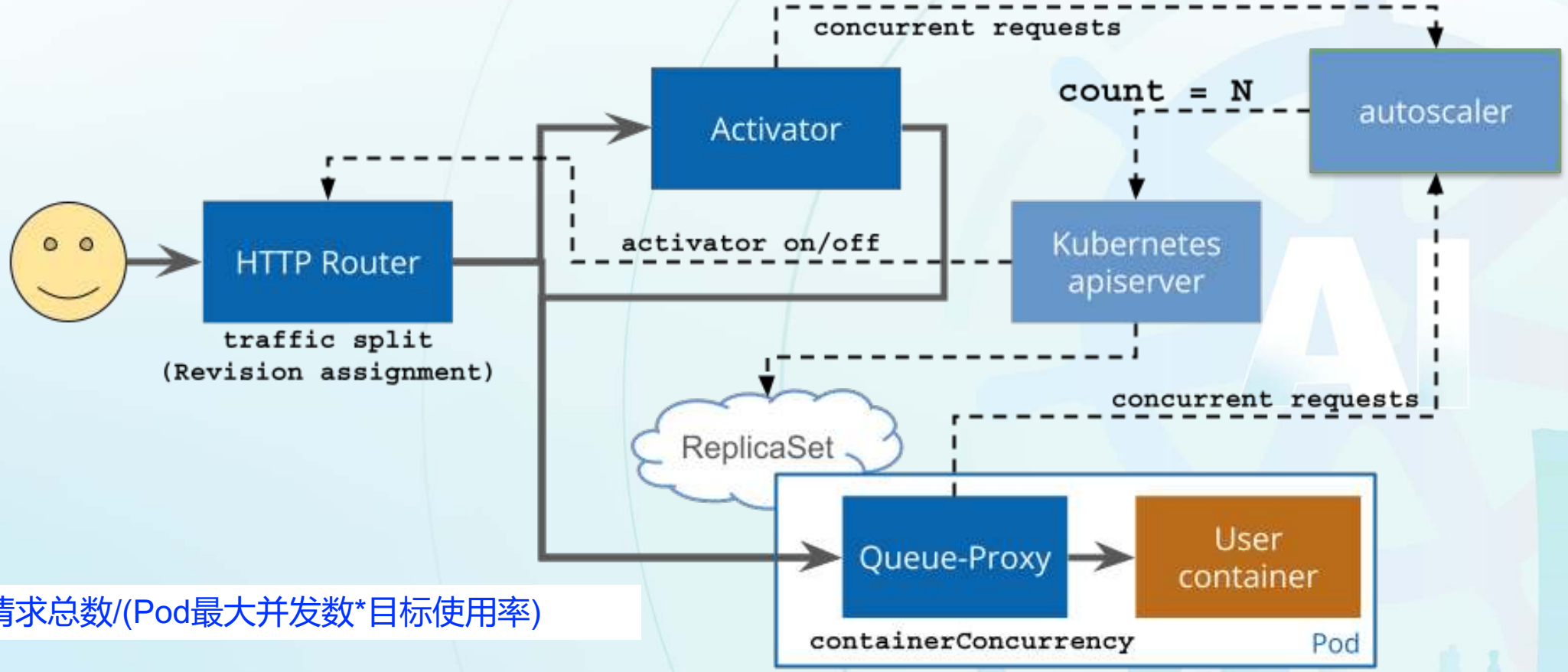
版本自动 GC

Knative如果每次修改就会创建新的版本，而随着迭代的加速，必然会导致很多历史版本，Knative 中提供了版本自动清理能力。可以通过 config-gc 配置版本 GC 策略

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: config-gc
  namespace: knative-serving
data:
  _example: |
    # ----------------------------------------
    # Garbage Collector Settings
    # ----------------------------------------
    # Duration since creation before considering a revision for GC or "disabled".
    retain-since-create-time: "48h"

    # Duration since active before considering a revision for GC or "disabled".
    retain-since-last-active-time: "15h"

    # Minimum number of non-active revisions to retain.
    min-non-active-revisions: "20"

    # Maximum number of non-active revisions to retain
    # or "disabled" to disable any maximum limit.
    max-non-active-revisions: "1000"
```

基于请求的自动弹性

基于 CPU或者Memory的弹性，有时候并不能完全反映业务的真实使用情况，而基于并发数或者每秒处理请求数（QPS/RPS）,对于 web 服务来说更能直接反映服务性能，Knative 提供了基于请求的自动弹性能力

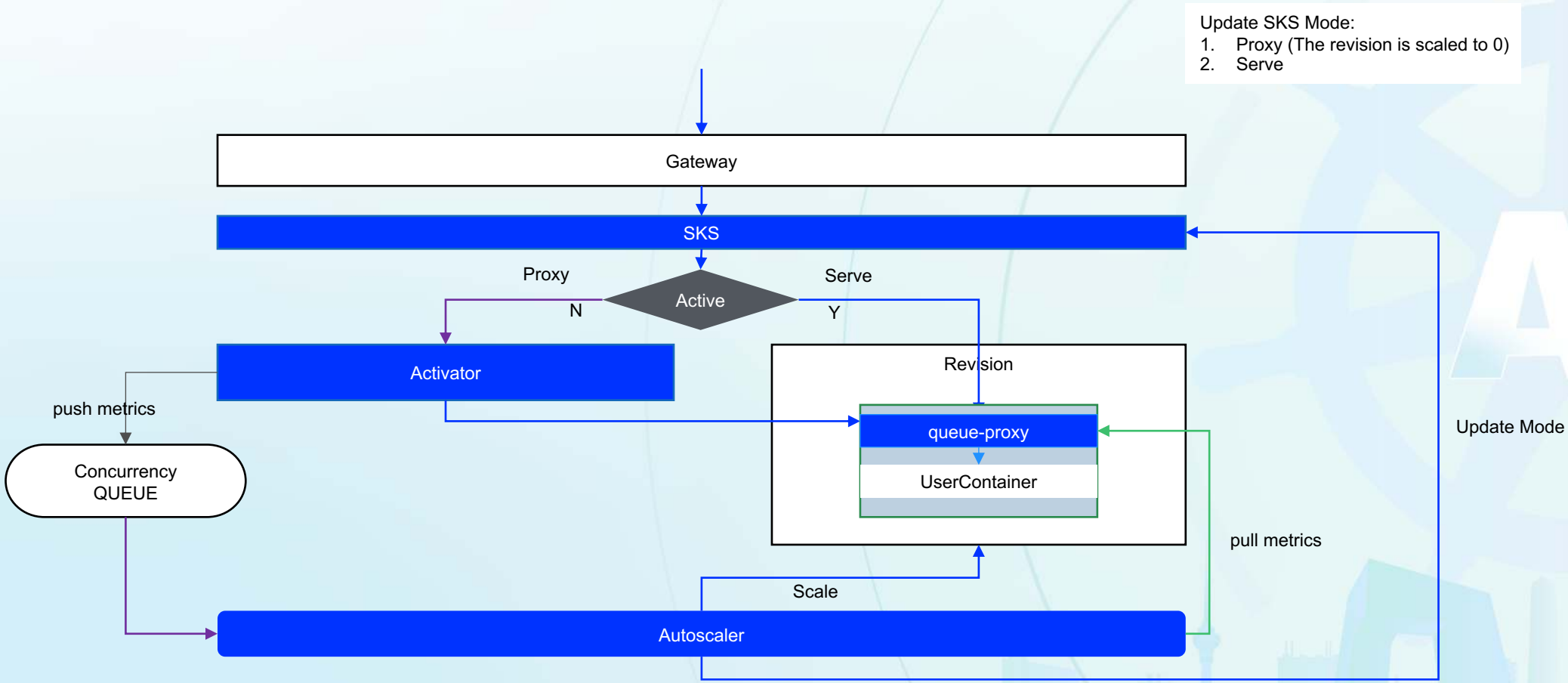


Pod数=并发请求总数/(Pod最大并发数*目标使用率)

缩容到 0 的实现机制

Knative 中定义了 2 种请求访问模式：Proxy和Serve。

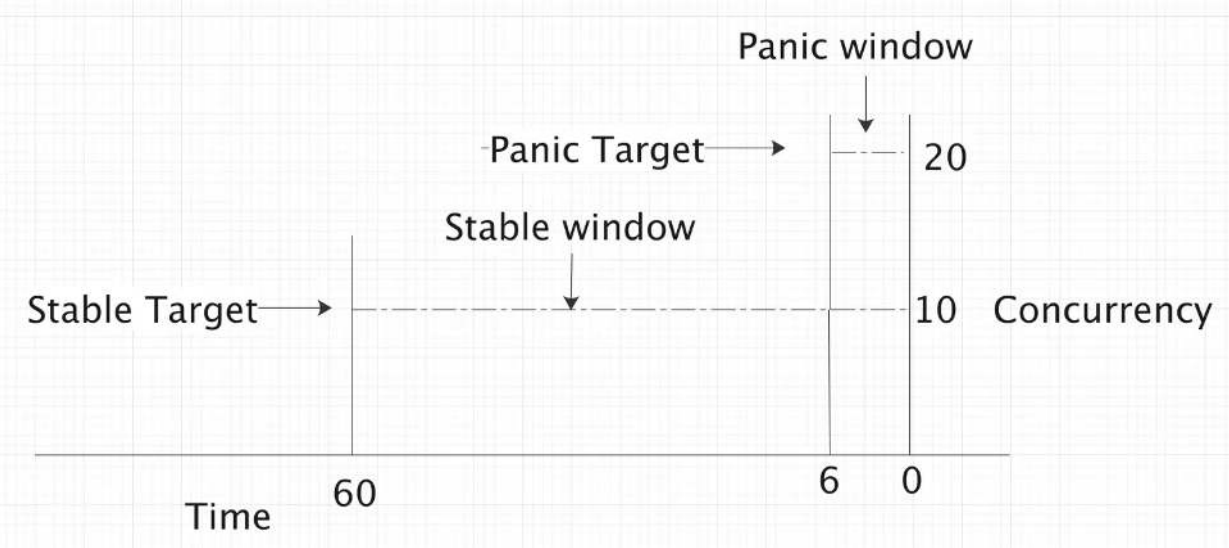
- Proxy 顾名思义，代理模式，也就是请求会通过 activator 组件进行代理转发
- Serve模式是请求直达模式，从网关直接请求到Pod， 不经过 activator 代理



如何应对突发流量

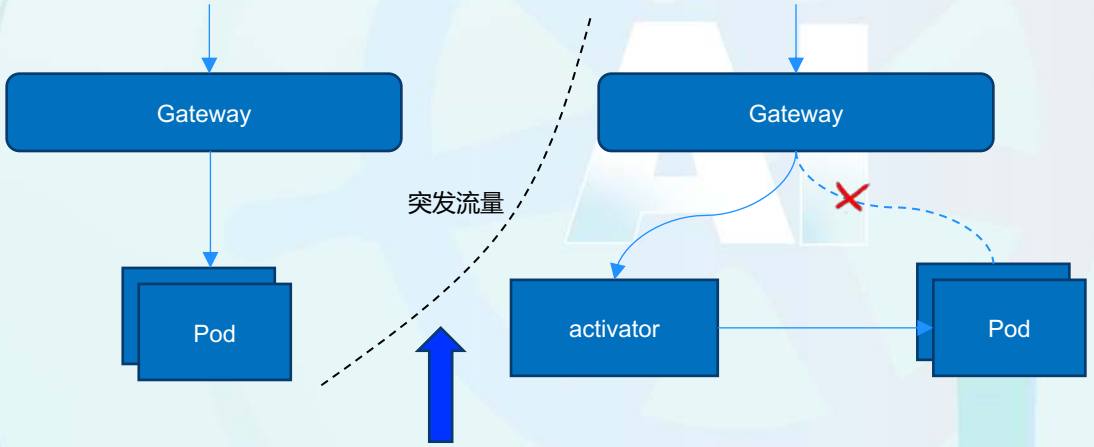
突发流量下如何快速弹资源

通过Stable（稳定模式）和Panic（恐慌模式）调节扩容响应节奏



突发流量下如何避免 Pod 被打爆

KPA中可以设置突发请求容量（target-burst-capacity）应对 Pod 被超预期的流量打爆



- 恐慌窗口期=稳定窗口期*panic-window-percentage
- 恐慌阈值（panic-threshold-percentage），恐慌模式下计算出来的Pod数>= 当前Ready Pod数 *恐慌阈值

减少冷启动的一些技巧

延迟缩容

对于启动成本较高的Pod， KPA 中可以通过设置 Pod 延迟缩容时间以及Pod缩容到 0 保留期，来减少Pod扩缩容频率

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
  namespace: default
spec:
  template:
    metadata:
      annotations:
        autoscaling.knative.dev/scale-down-delay: ""60s"
        autoscaling.knative.dev/scale-to-zero-pod-retention-period: "1m5s"
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:73fbdd56
```

调低目标使用率，实现资源预热

通过调小目标使用率可以提前扩容超过实际需要使用量的Pod数，在请求达到目标并发数之前进行扩容，间接的可以做到资源预热

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
  namespace: default
spec:
  template:
    metadata:
      annotations:
        autoscaling.knative.dev/target-utilization-percentage: "80"
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:73fbdd56
```

自定义域名

全局域名配置

可以通过修改 config-domain，配置全局域名后缀

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: config-domain
  namespace: knative-serving
data:
  mydomain.com: ""
```

指定服务自定义域名

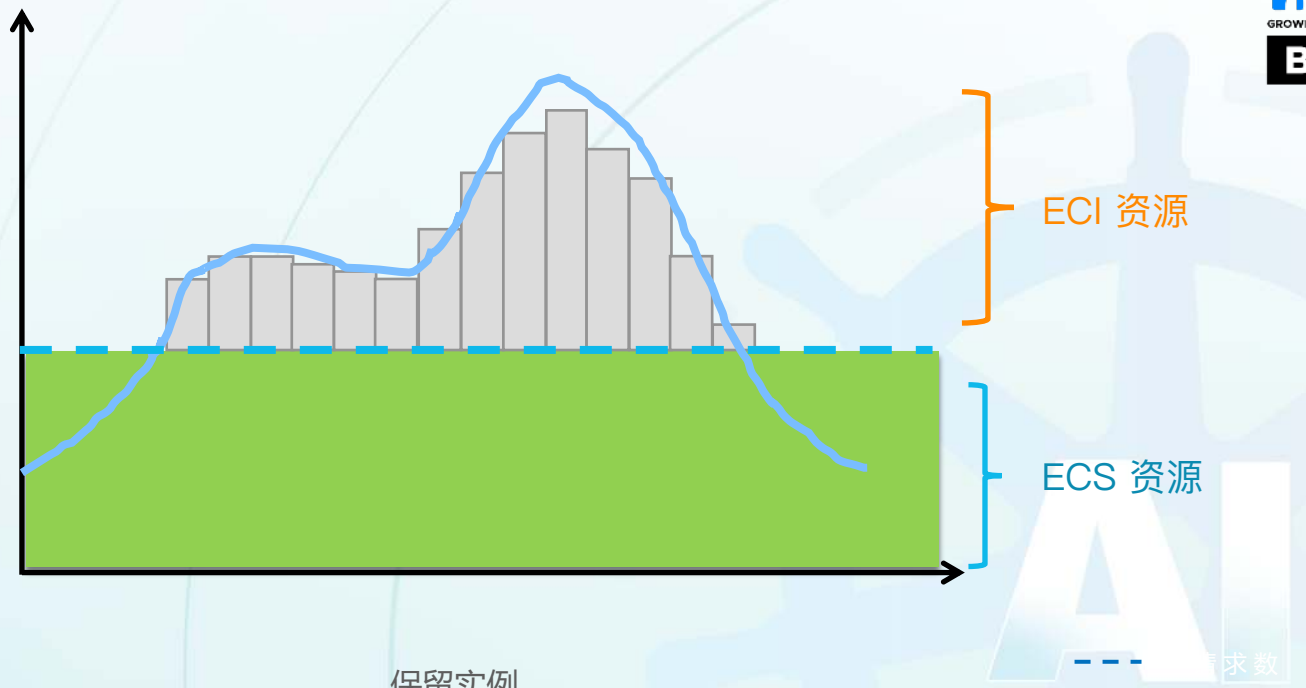
自定义指定单个 Service 的域名，可以使用 DomainMapping

```
apiVersion: serving.knative.dev/v1alpha1
kind: DomainMapping
metadata:
  name: <domain-name>
  namespace: <namespace>
spec:
  ref:
    name: <service-name>
    kind: Service
    apiVersion: serving.knative.dev/v1
  tls:
    secretName: <cert-secret>
```

保留资源池

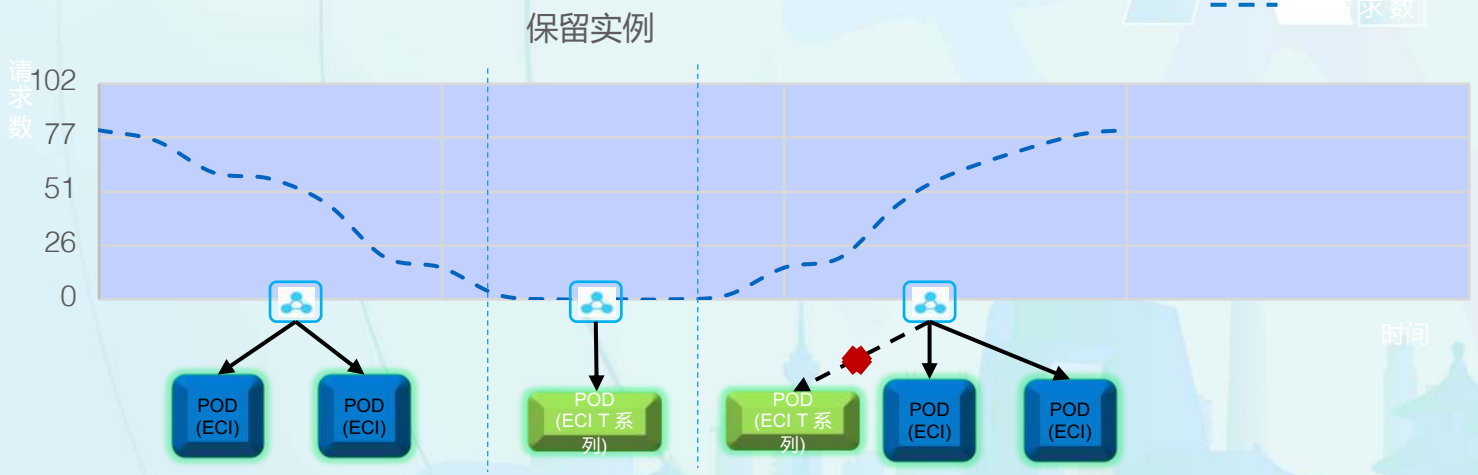
ECS 和 ECI 混合使用

常态情况下使用 ECS 资源，突发流量使用 ECI 资源



资源预热

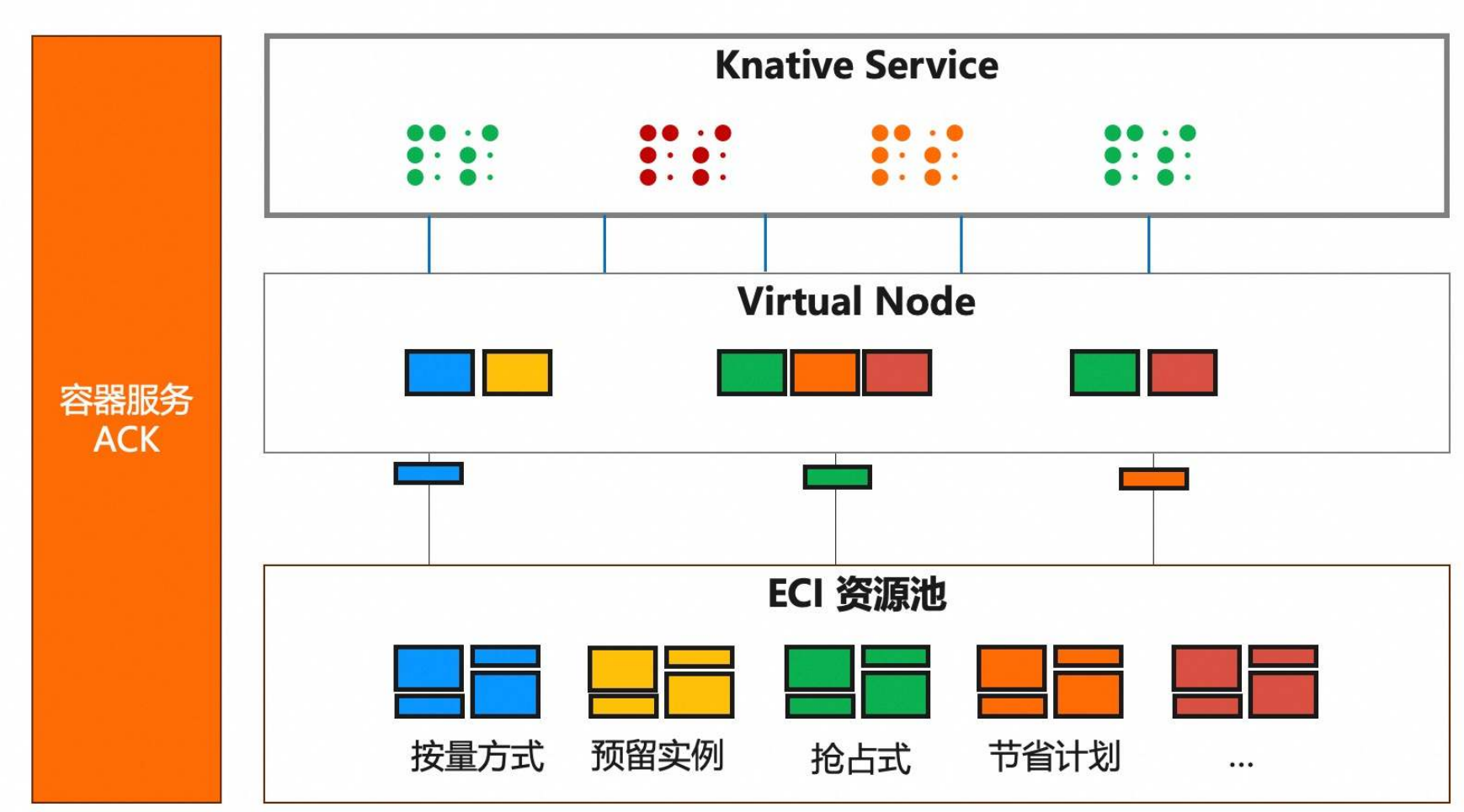
完全使用ECI的场景，也可以通过保留资源池实现资源预热



抢占式实例

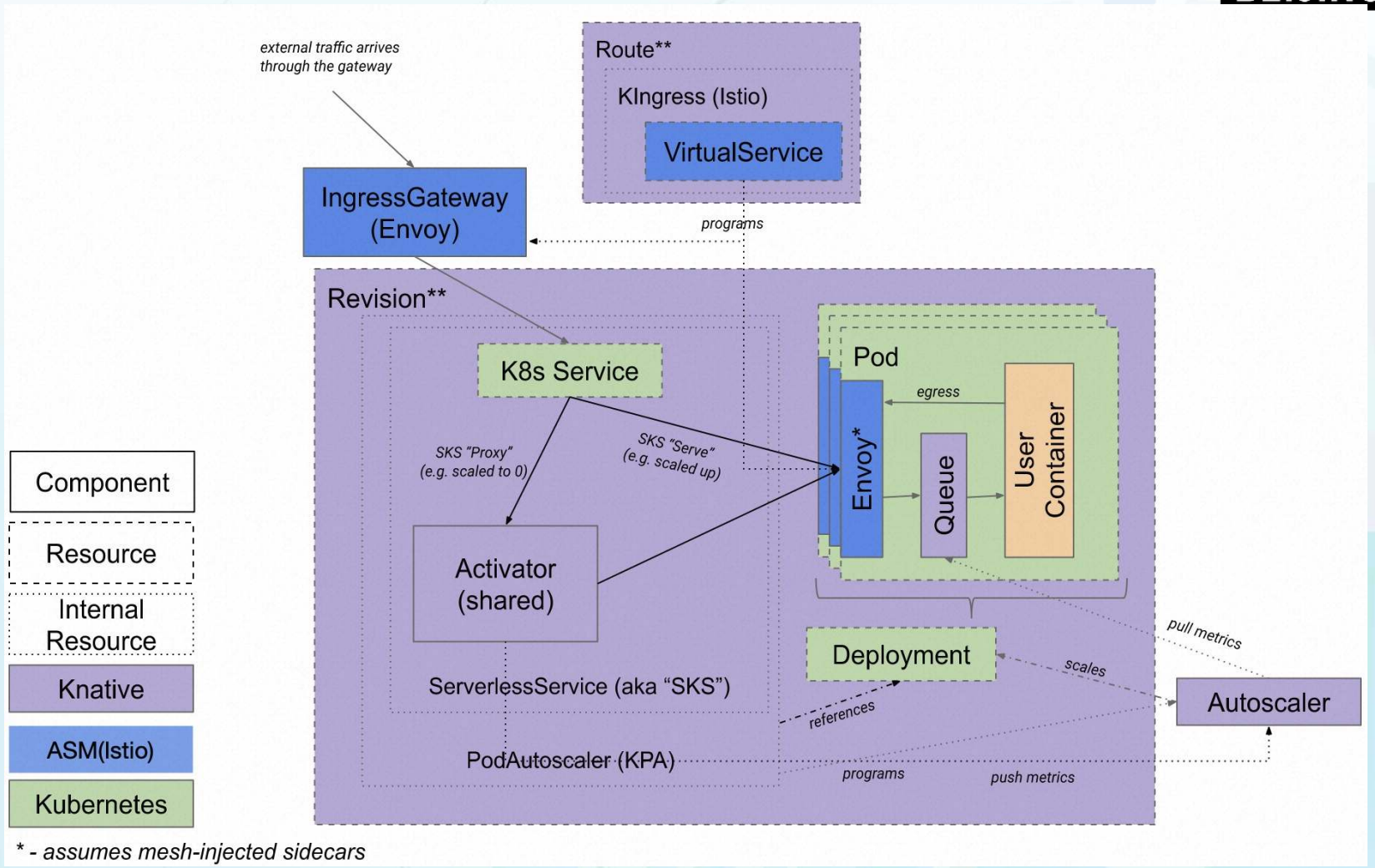
Knative 结合抢占式实例的优势：

- Serverless场景
- 优雅下线的天然适配
- 成本敏感

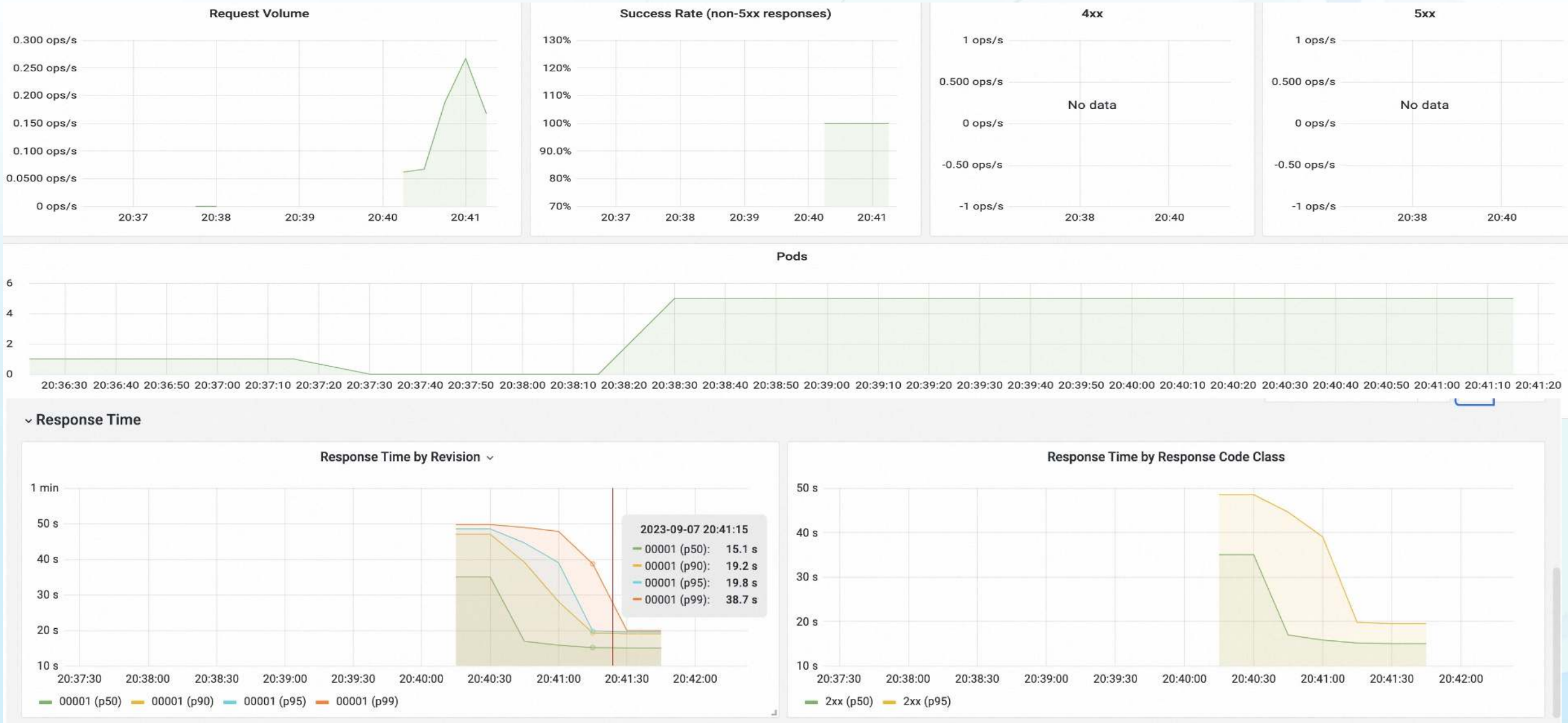


基于 Service Mesh 管理 Knative 流量

- 服务限流
- 服务级熔断
- 主机级熔断和同可用区优先路由
- 请求优先级调度
- 流量拓扑



可观测性大盘


Request Volume
0.300 ops/s
0.250 ops/s
0.200 ops/s
0.150 ops/s
0.100 ops/s
0.0500 ops/s
0 ops/s
20:37
20:38
20:39
20:40
20:41
Success Rate (non-5xx responses)
130%
120%
110%
100%
90.0%
80%
70%
20:37
20:38
20:39
20:40
20:41
4xx
1 ops/s
0.500 ops/s
0 ops/s
-0.50 ops/s
-1 ops/s
No data
20:38
20:40
5xx
1 ops/s
0.500 ops/s
0 ops/s
-0.50 ops/s
-1 ops/s
No data
20:38
20:40
Pods
6
4
2
0
20:36:30 20:36:40 20:36:50 20:37:00 20:37:10 20:37:20 20:37:30 20:37:40 20:37:50 20:38:00 20:38:10 20:38:20 20:38:30 20:38:40 20:38:50 20:39:00 20:39:10 20:39:20 20:39:30 20:39:40 20:39:50 20:40:00 20:40:10 20:40:20 20:40:30 20:40:40 20:40:50 20:41:00 20:41:10 20:41:20
Response Time
Response Time by Revision
1 min
50 s
40 s
30 s
20 s
10 s
2023-09-07 20:41:15
00001 (p50): 15.1 s
00001 (p90): 19.2 s
00001 (p95): 19.8 s
00001 (p99): 38.7 s
20:37:30 20:38:00 20:38:30 20:39:00 20:39:30 20:40:00 20:40:30 20:41:00 20:41:30 20:42:00
00001 (p50) 00001 (p90) 00001 (p95) 00001 (p99)
Response Time by Response Code Class
50 s
40 s
30 s
20 s
10 s
20:37:30 20:38:00 20:38:30 20:39:00 20:39:30 20:40:00 20:40:30 20:41:00 20:41:30 20:42:00
2xx (p50) 2xx (p95)

成本洞察

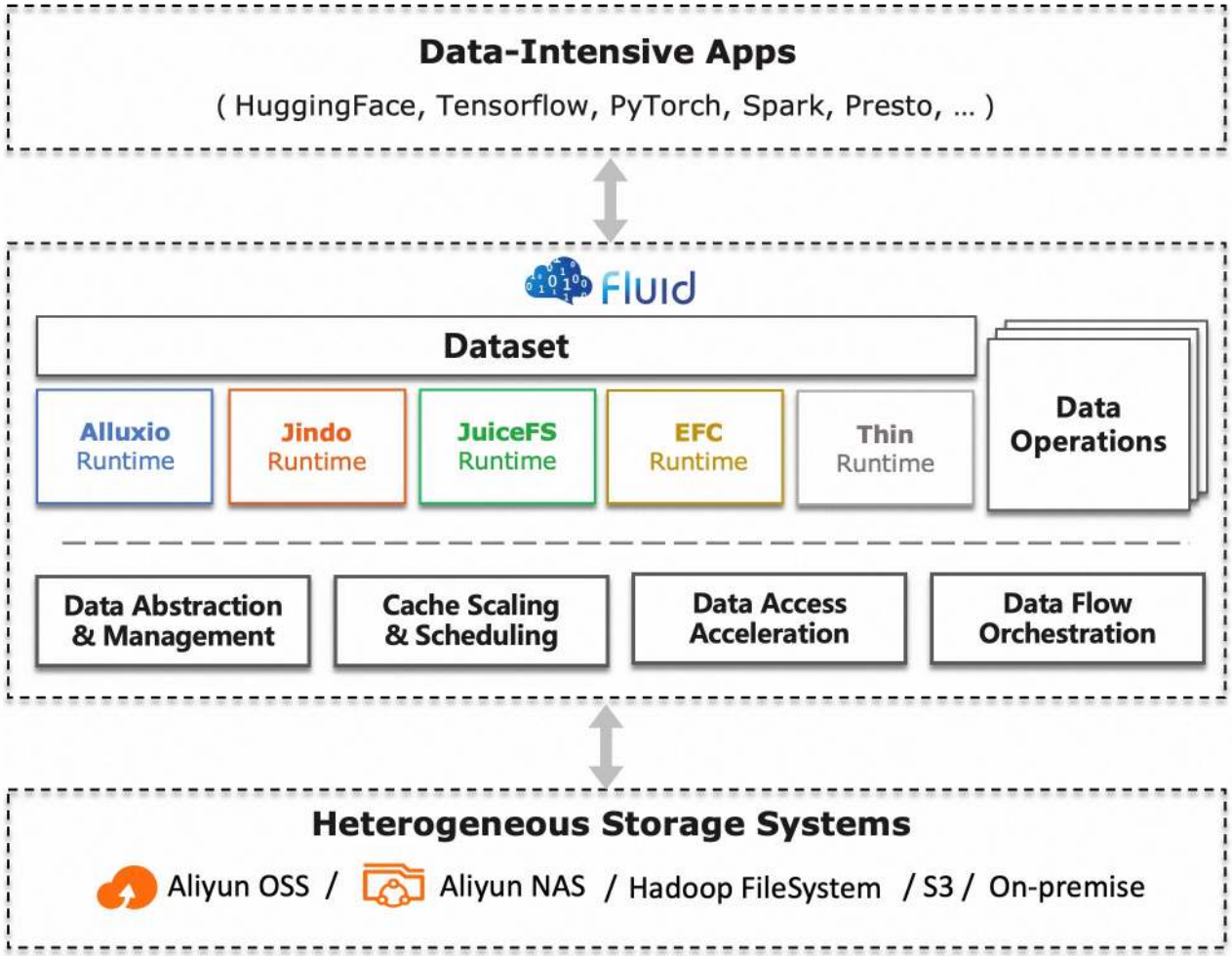
成本洞察功能可以协助企业IT成本管理人员从多维度了解集群资源使用量及成本分布，提供成本节约建议

共享 GPU 调度与显存交换

结合 ACK 共享GPU调度能力，Knative 模型服务可以充分利用cGPU显存隔离能力，高效利用GPU设备资源

```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
  namespace: default
spec:
  template:
    spec:
      containerConcurrency: 1
      containers:
      - image: registry-vpc.cn-hangzhou.aliyuncs.com/demo-test/test:helloworld-go
        name: user-container
        ports:
        - containerPort: 6666
          name: http1
          protocol: TCP
        resources:
          limits:
            aliyun.com/gpu-mem: "3"
```

Fluid：弹性数据集编排和加速

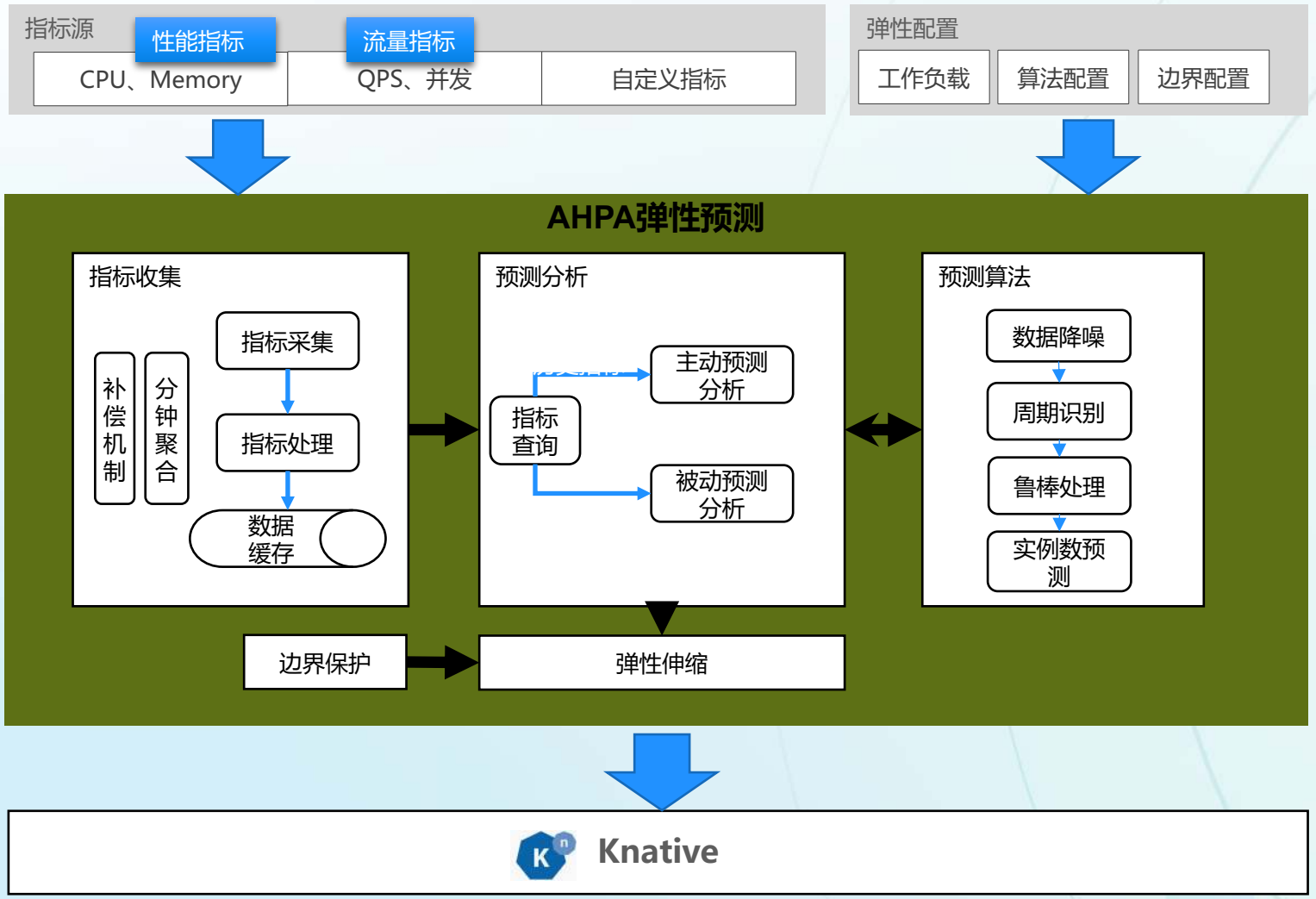


CLOUD NATIVE SANDBOX

核心功能：

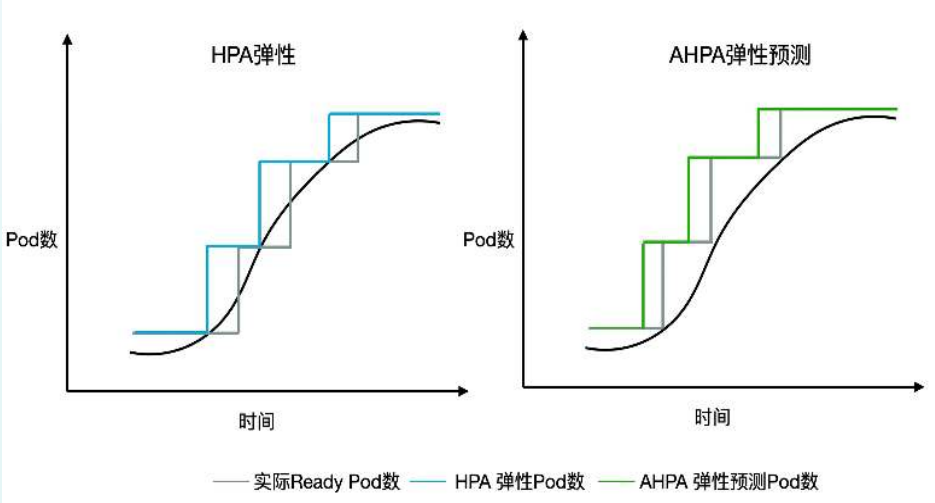
- **数据弹性与调度**：结合自动弹性，亲和性调度等多维度能力管理多种分布式缓存系统，提升数据访问效率
- **数据访问接入与适配**：支持原生、Serverless Kubernetes等多样化环境，实现数据访问接入
- **数据流任务编排**：支持多个数据消费和操作间的依赖定义，自动化数据消费过程。

AHPA优化大模型的智能弹性

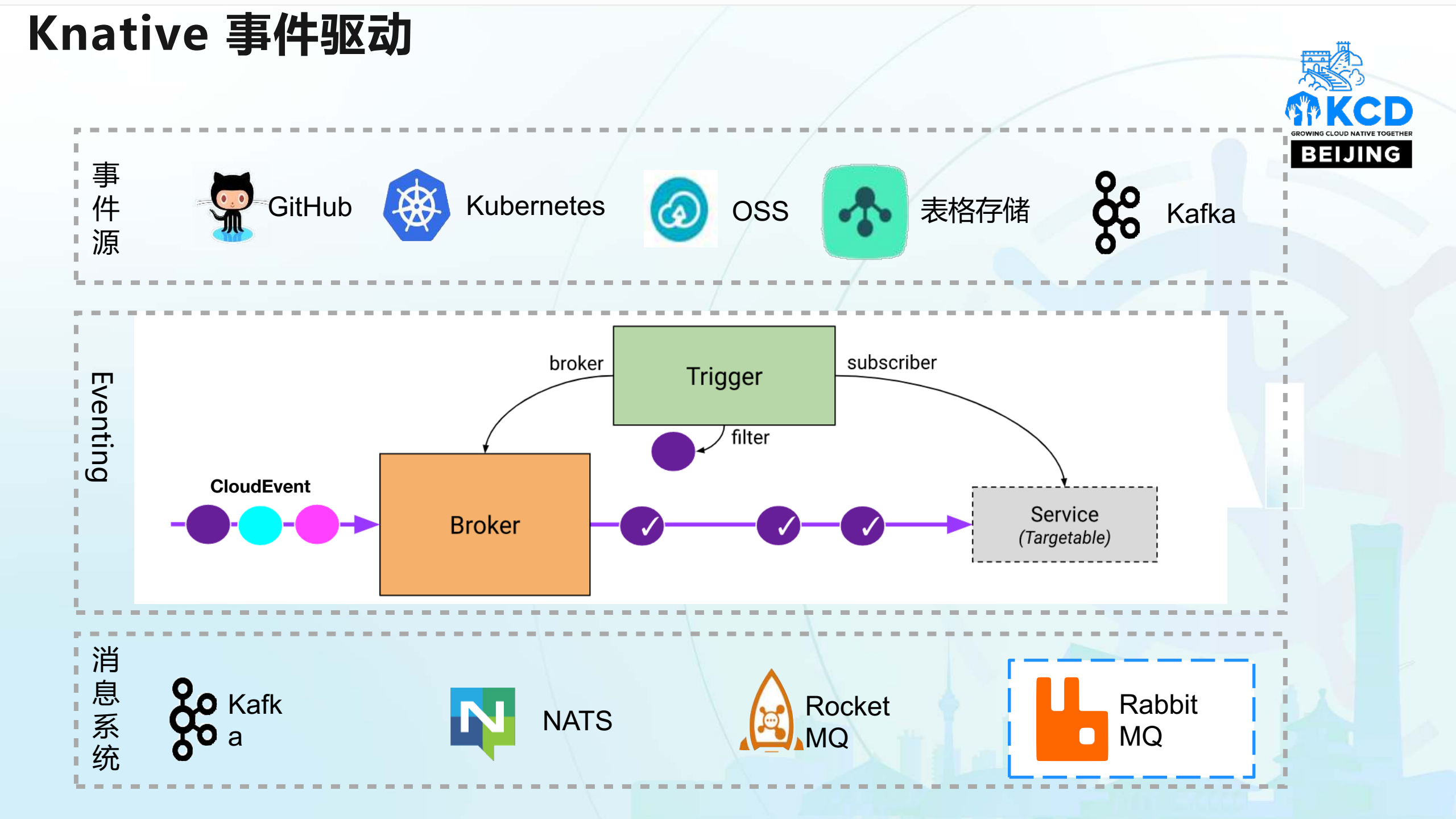


资源提前预热

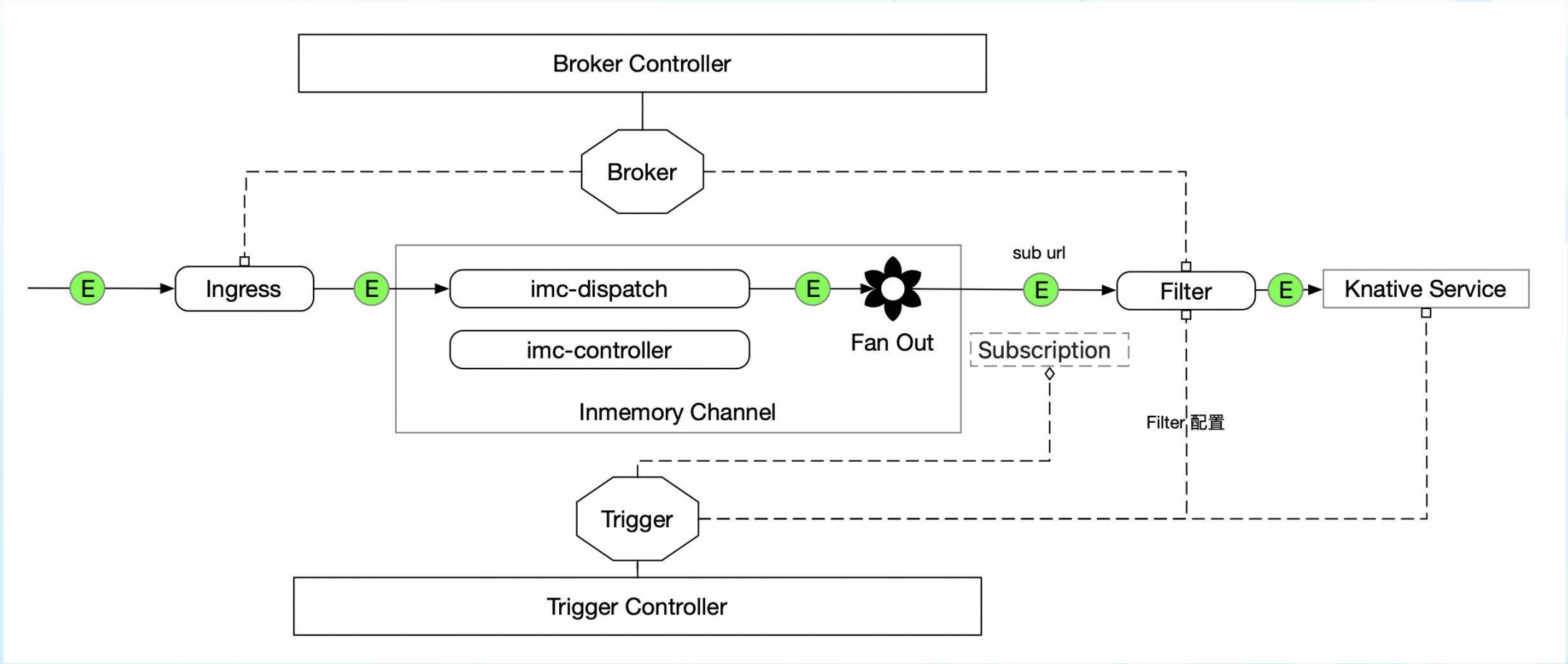
解决客户弹性滞后冷启动的问题，通过弹性预测，提前预热资源。



Knative 事件驱动

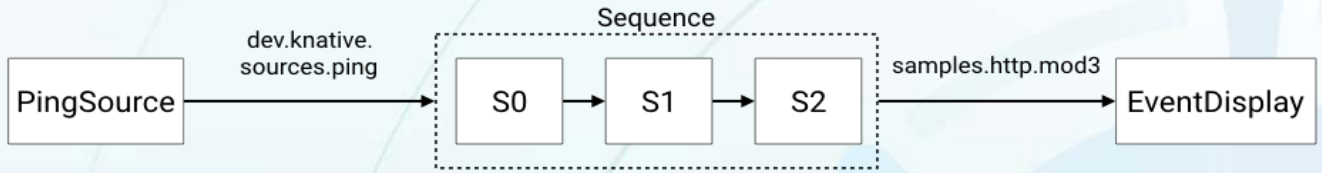


IMC 事件驱动模型：内部流转机制

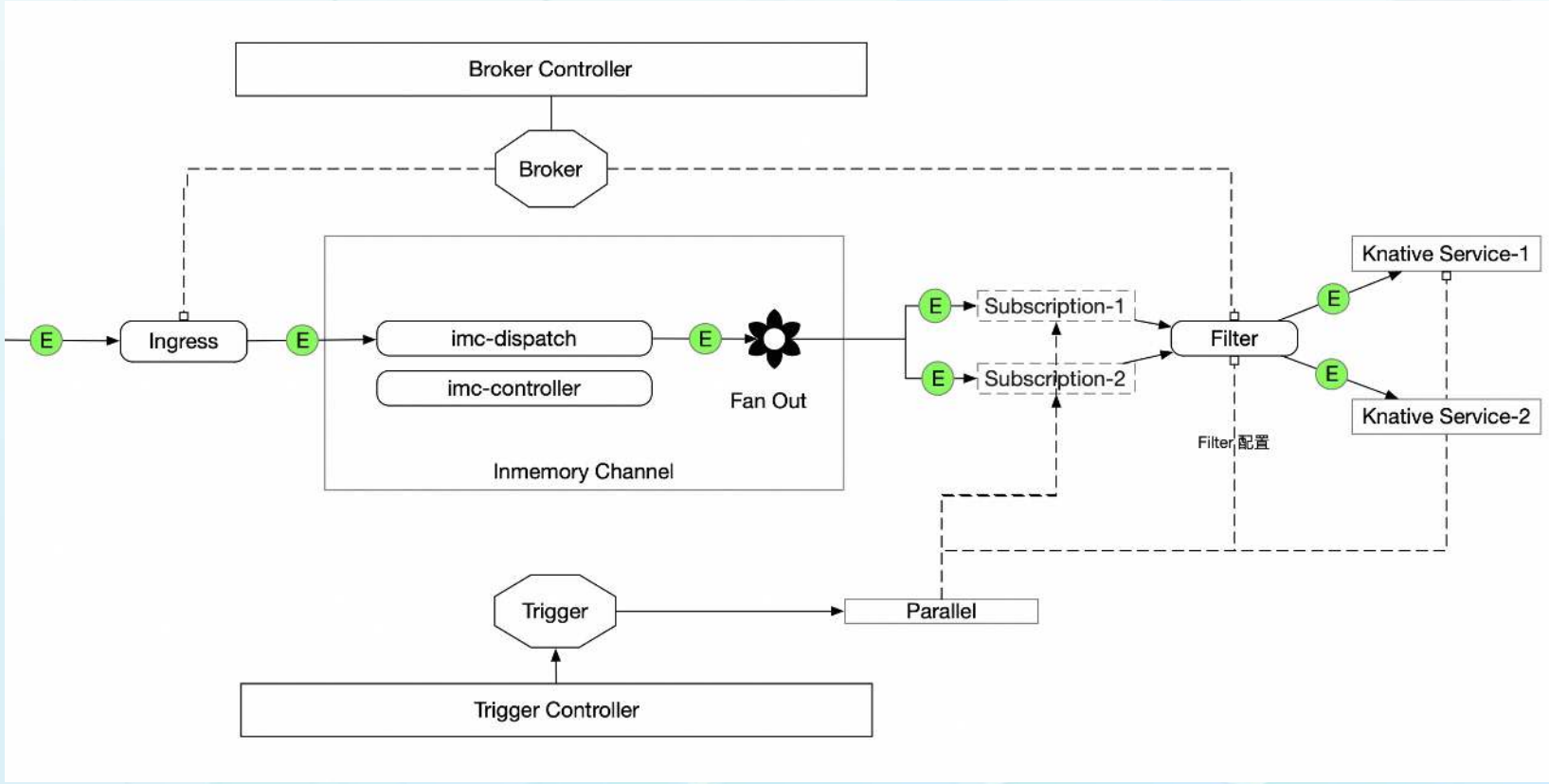


事件编排

Sequence ：顺序事件处理流程

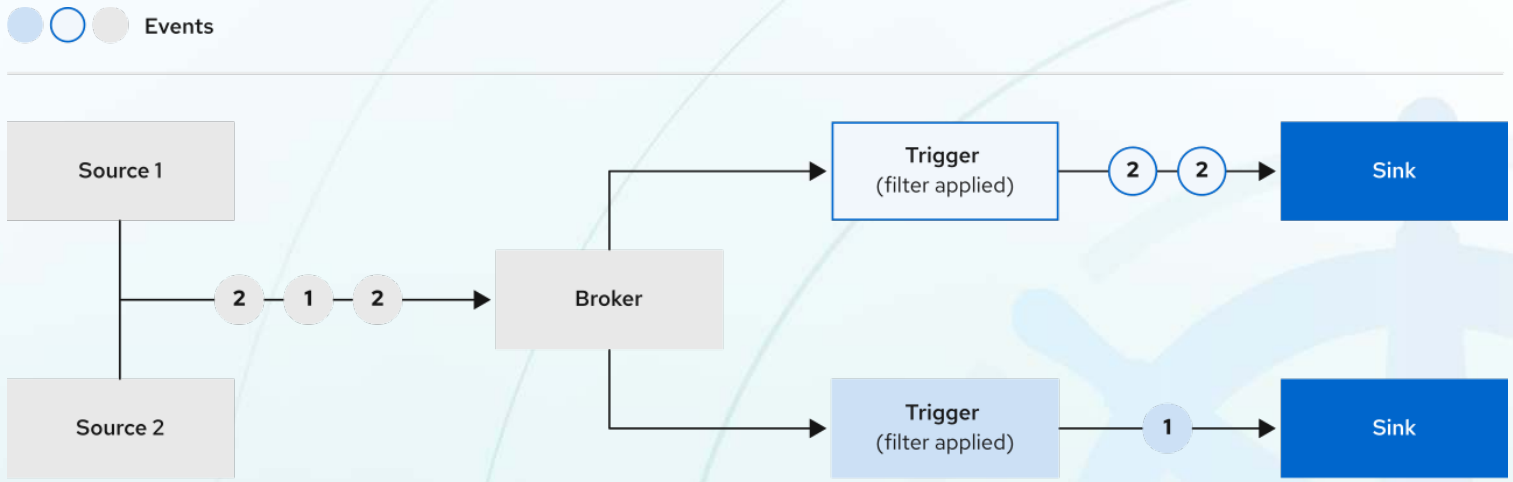


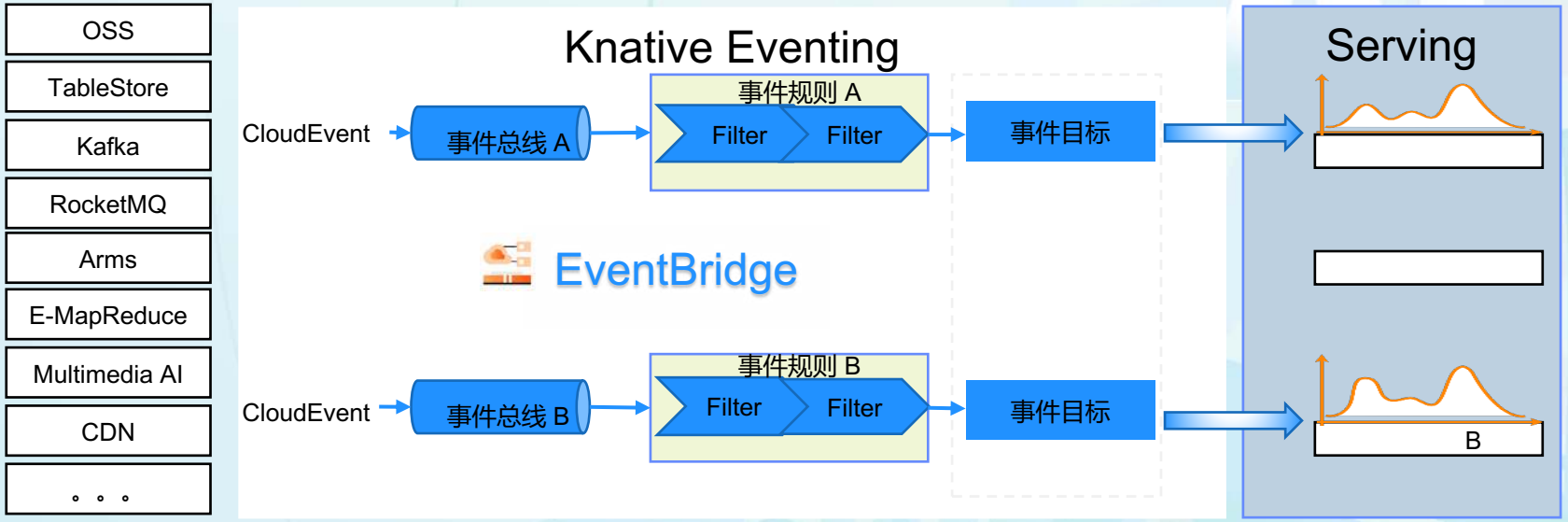
Parallel：并行事件处理流程



基于 EventBridge 事件驱动

- 事件源
- 事件转发
- 事件处理





KCD
GROWING CLOUD NATIVE TOGETHER
BEIJING

Part 02

Rokid 基于 Knative 的最佳实践

平台简介

Rokid简介

Rokid 创立于 2014 年，是一家专注于人机交互技术的产品平台公司，2018 年即被评为国家高新技术企业。Rokid 作为行业的探索者、领跑者，目前致力于 AR 眼镜等软硬件产品的研发及以 YodaOS 操作系统为载体的生态构建。公司通过语音识别、自然语言处理、计算机视觉、光学显示、芯片平台、硬件设计等多领域研究，将前沿的 AI 和 AR 技术与行业应用相结合，为不同垂直领域的客户提供全栈式解决方案，有效提升用户体验、助力企业增效、赋能公共安全，其 AI、AR 产品已在全球八十余个国家和地区投入使用。

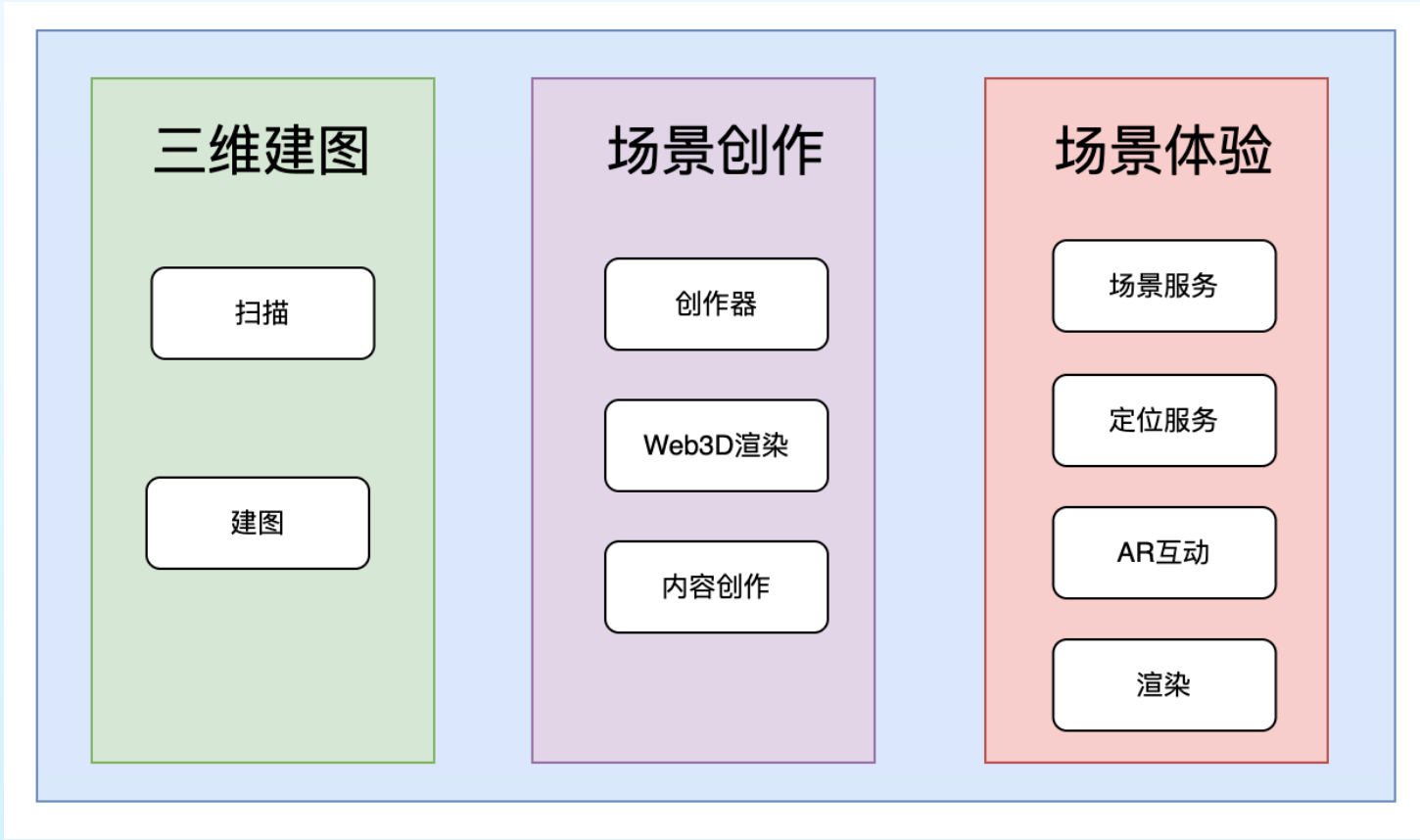
业务场景

Rokid 在数字文化领域，围绕展陈导览解决方案，主要形成了三维建图，场景创作，场景体验三个业务模块，每个模块都有不同的后台平台支撑

三维建图：制作展陈导览的第一步是取景，通过设备获取场地的真实布景，然后通过算法处理，进行三维建模，之后可以经过创作器进行下一步的内容创作。

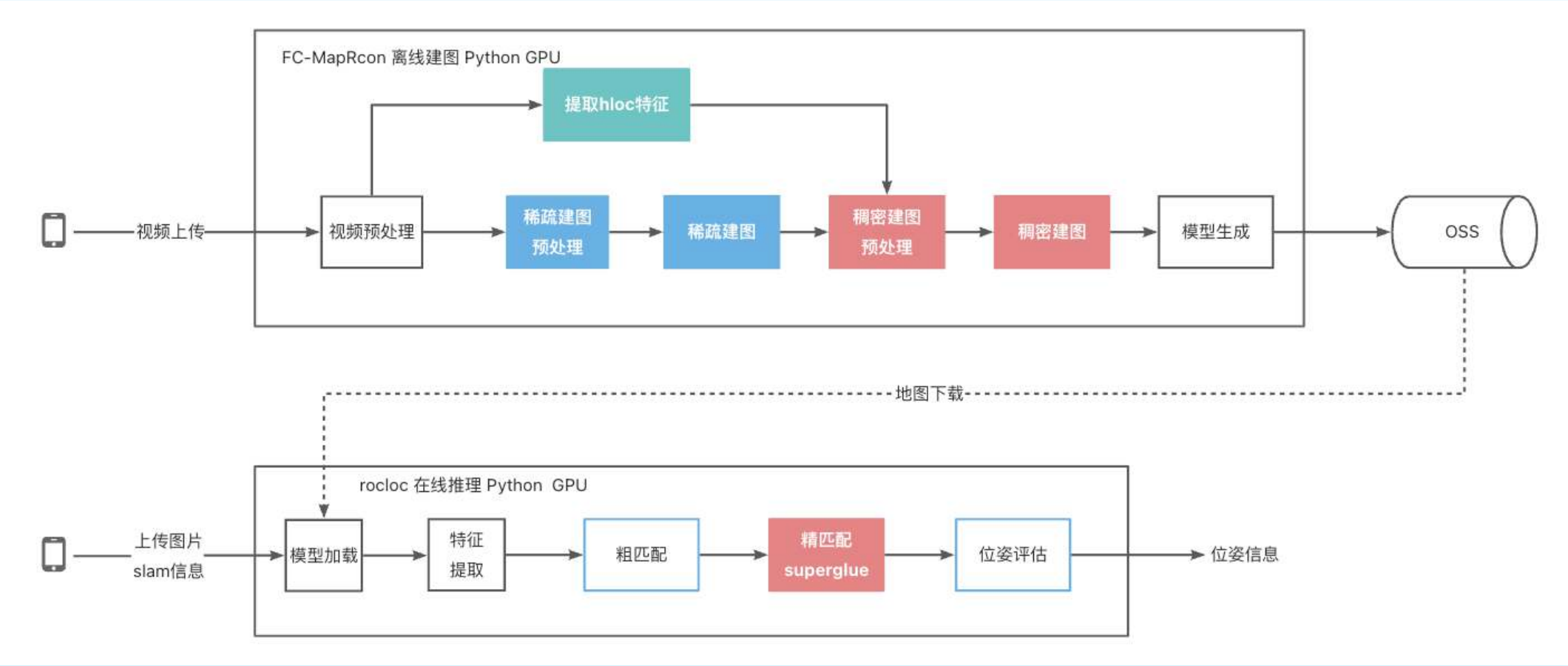
场景创作：在三维建模生成的视频流上创作，通过 Web3D 渲染引擎，将创作内容与场景紧密结合，结合硬件设备，在 AR 设备使用时，形成一体化的体验效果。

场景体验：AR 设备在使用时，根据定位服务，锚定在场景中的位置，根据位置的不同会显示不同的空间内容，达到扩展现实场景的效果。



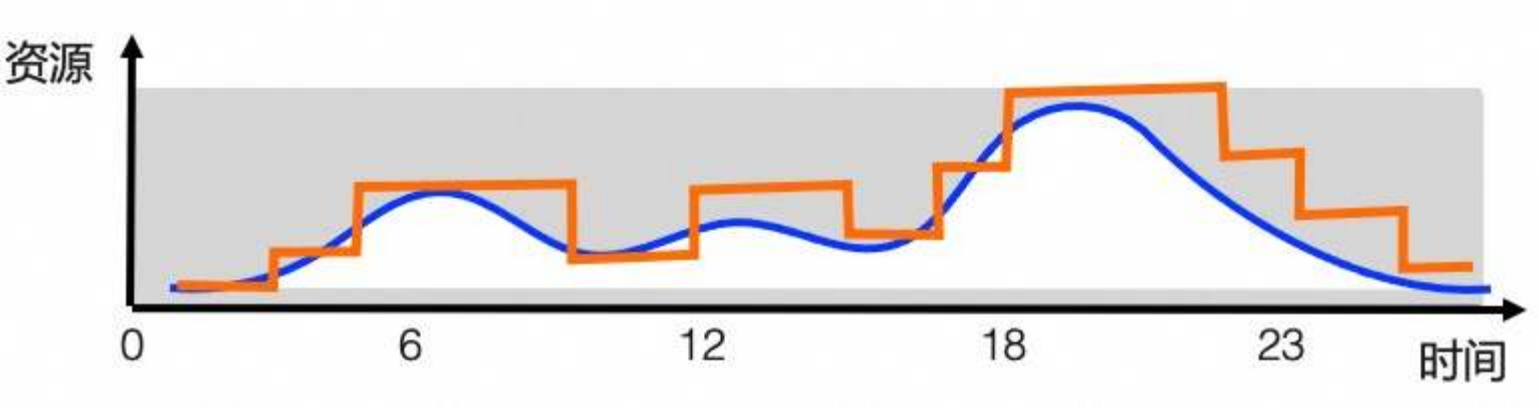
业务场景

技术架构


FC-MapRcon 离线建图 Python GPU
提取hloc特征
视频上传
视频预处理
稀疏建图
预处理
稀疏建图
稠密建图
预处理
稠密建图
模型生成
OSS
地图下载
rocloc 在线推理 Python GPU
上传图片
slam信息
模型加载
特征
提取
粗匹配
精匹配
superglue
位姿评估
位姿信息

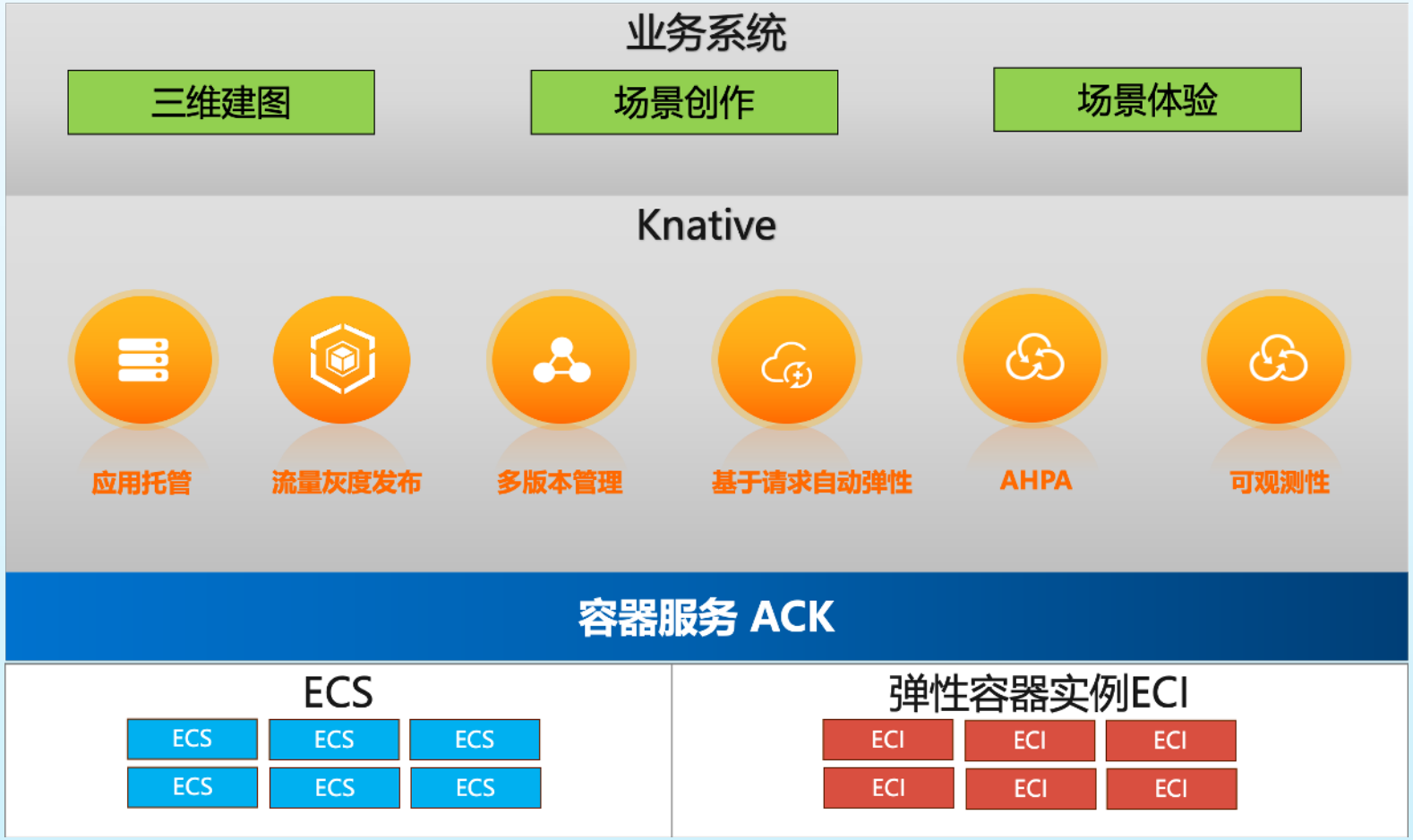
模型发布痛点

应用往往很难做到秒级启动，也使得基于 HPA 扩容存在较大的滞后性，带来真实业务的稳定性风险。如何减少弹性滞后带来的业务影响，是需要我们要解决的问题



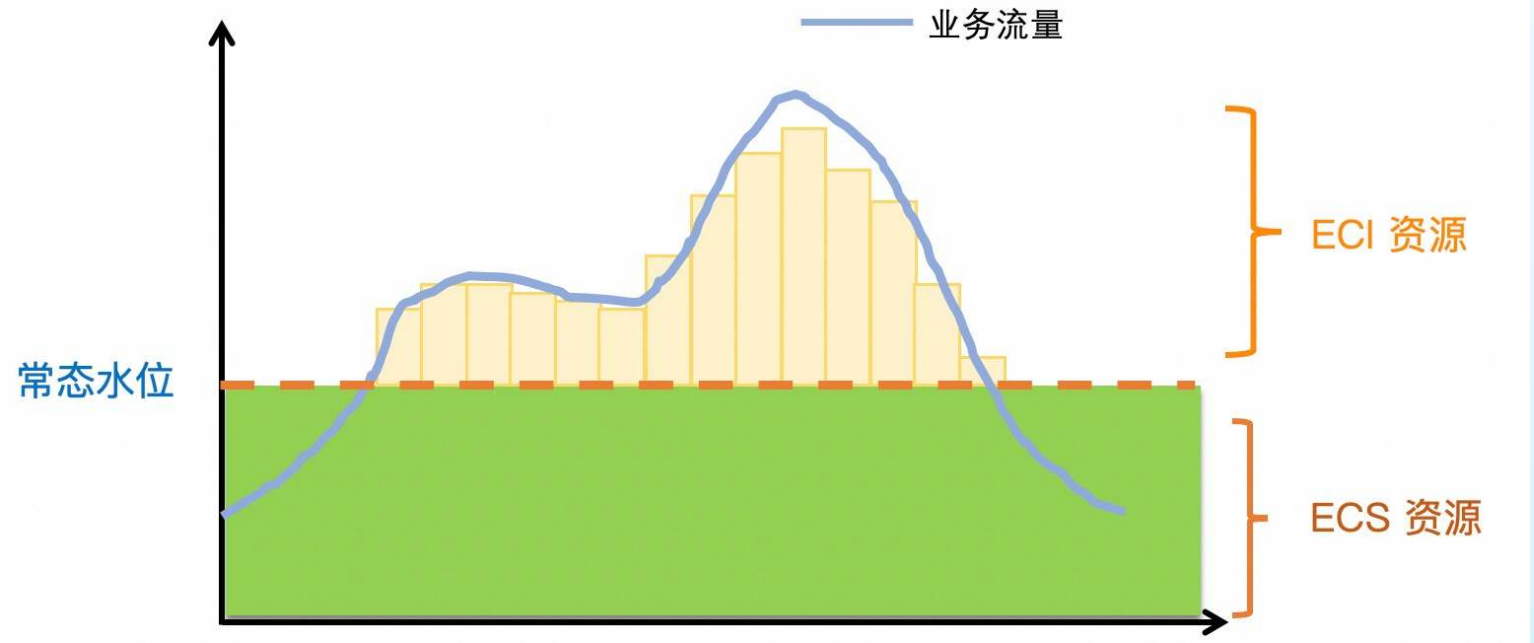
解决方案

从资源按需使用、GPU 供给、稳定性以及标准化等方面的考虑，我们选择了阿里云 ACK + Knative 的解决方案，保证弹性供给兼顾成本最优，整体方案如下



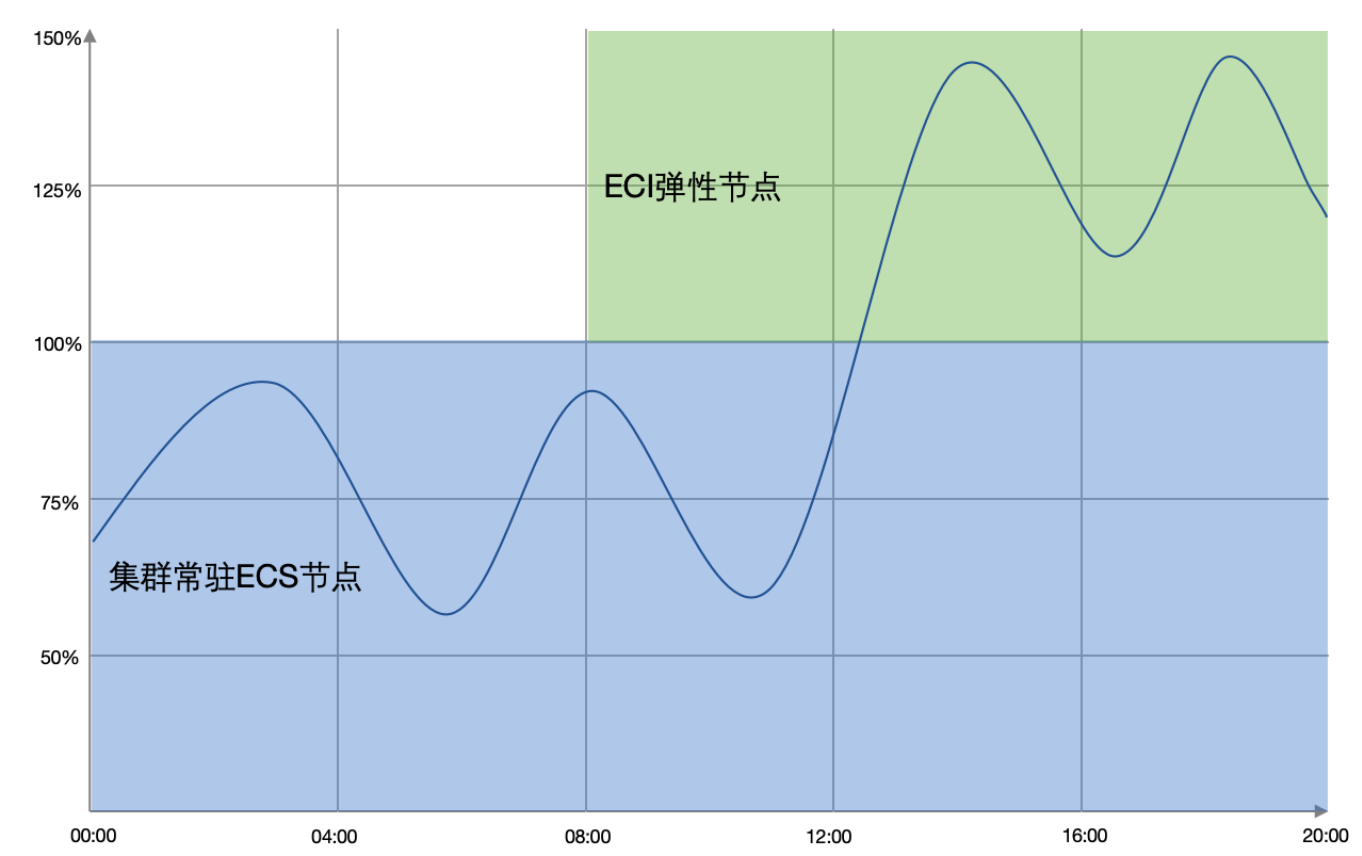
ECS、 ECI混部

- 弹性容器实例 ECI（Elastic Container Instance）是阿里云一款敏捷安全 Serverless 容器运行服务，用户无需管理底层服务器资源，同样也不需要关心运行过程中的容量规划，非常适合应对这种流量突发下业务场景。
- 但从弹性供给来看，ECI 并不能完全保证资源的供给，结合我们常态下业务使用情况，当前采取混合部署模式，兼顾成本及稳定性两方面的业务目标。如图所示，常态业务下我们使用 ECS 资源，在突发业务流量下，通过 ECI 提供按需使用资源



成果收益

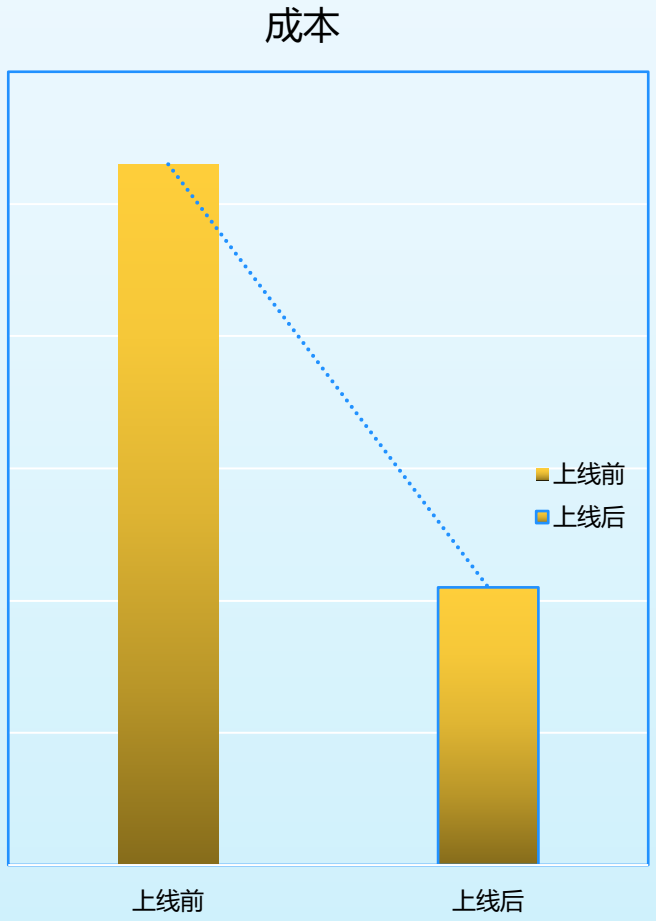
- 最大化提升资源使用率

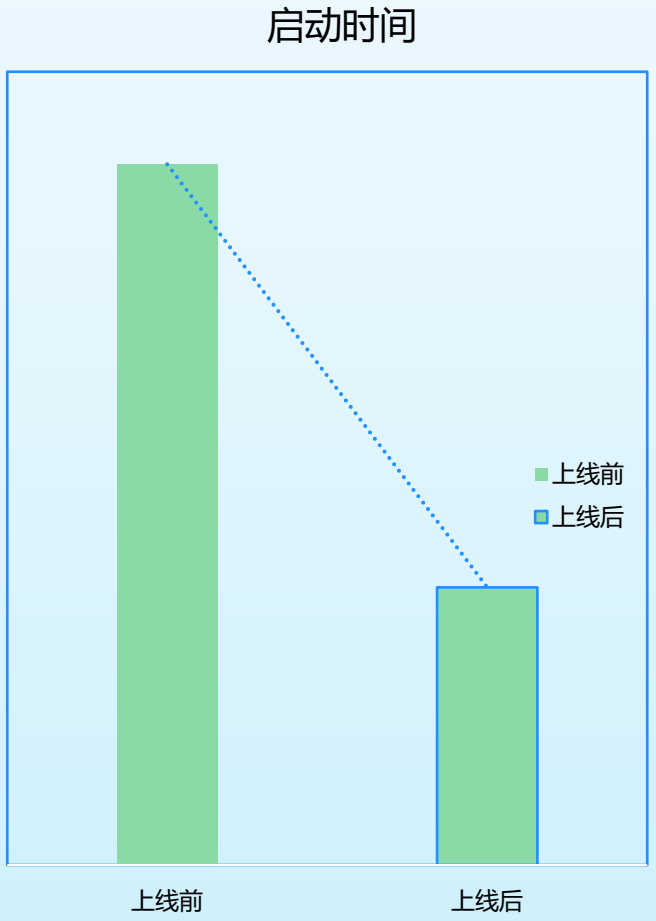


- ECI节点弹扩节点应对突发流量和Job场景
- ECS常驻节点保证常规业务资源使用
- ECS混合ECI使用最大化提升资源使用率

成果收益

- 成本节约比例 60%
- 服务扩容调度GPU时常可缩短至分钟级别





成果收益

钉钉基于 Rokid 的“AR 空间创作工具”灵境平台开发了“数字文化墙”

上海旅游节灵境AR花车巡游引爆城市级空间体验

成果收益

携手央博&阿里云，全球首个李白数字展亮相云栖大会

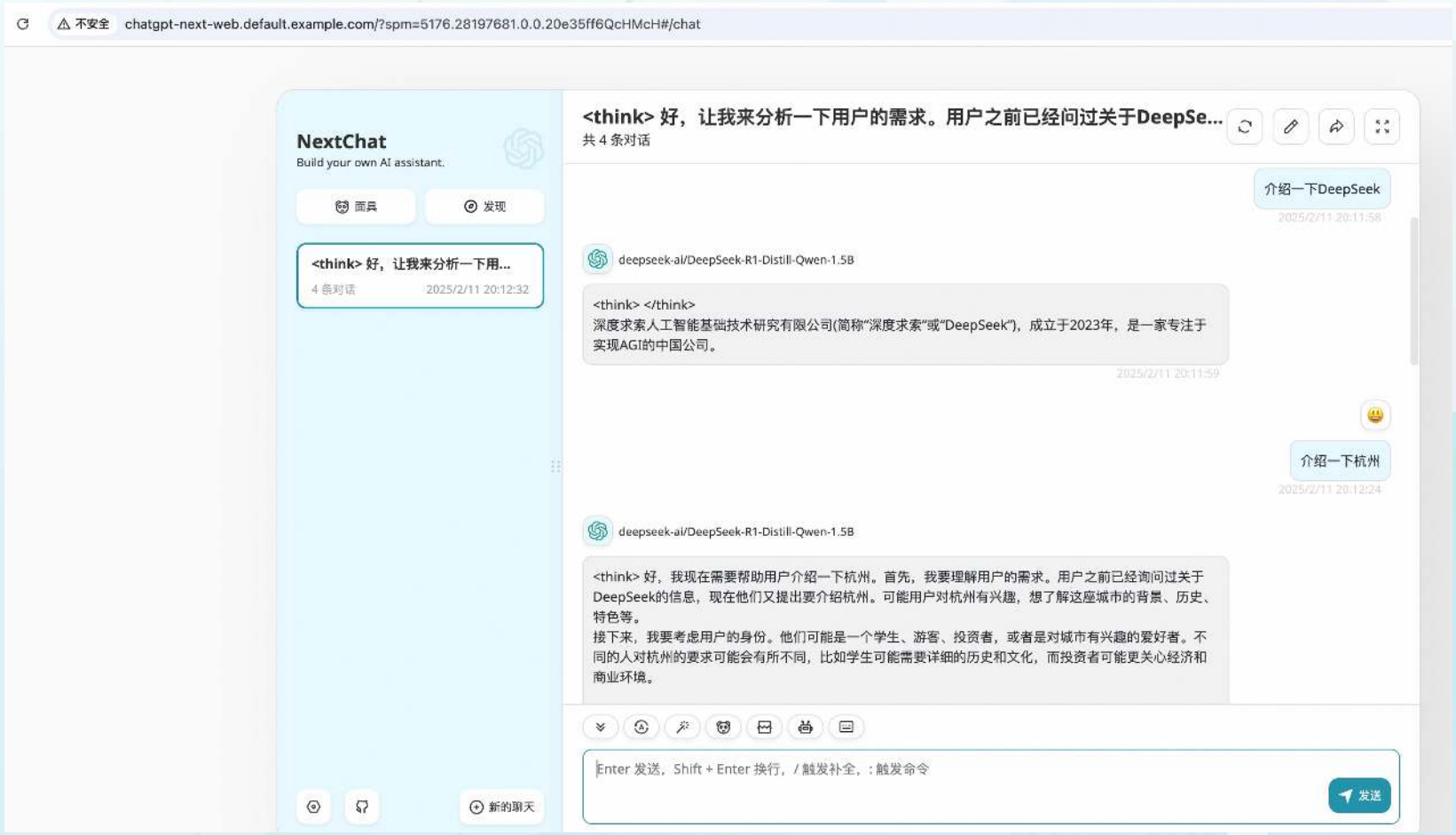
3D经典奥特曼全国AR首展

Part 03
Knative 与 DeepSeek

基于 Knative 部署 DeepSeek-R1

部署DeepSeek-R1模型

部署个人AI助手



Thanks.

阿里云 Knative 钉钉交流群