ThoughtWorks®

welcome

微服务实践分享

ThoughtWorks Insights

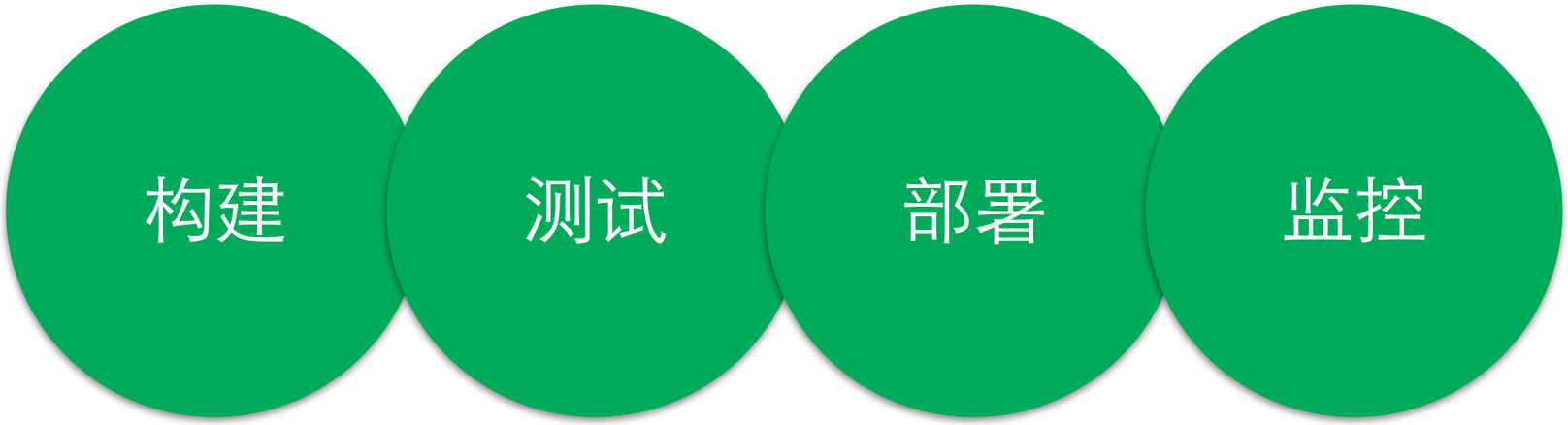
什么是微服务?

- 一组小的服务
- 独立的进程
- 独立的部署
- 轻量的通信

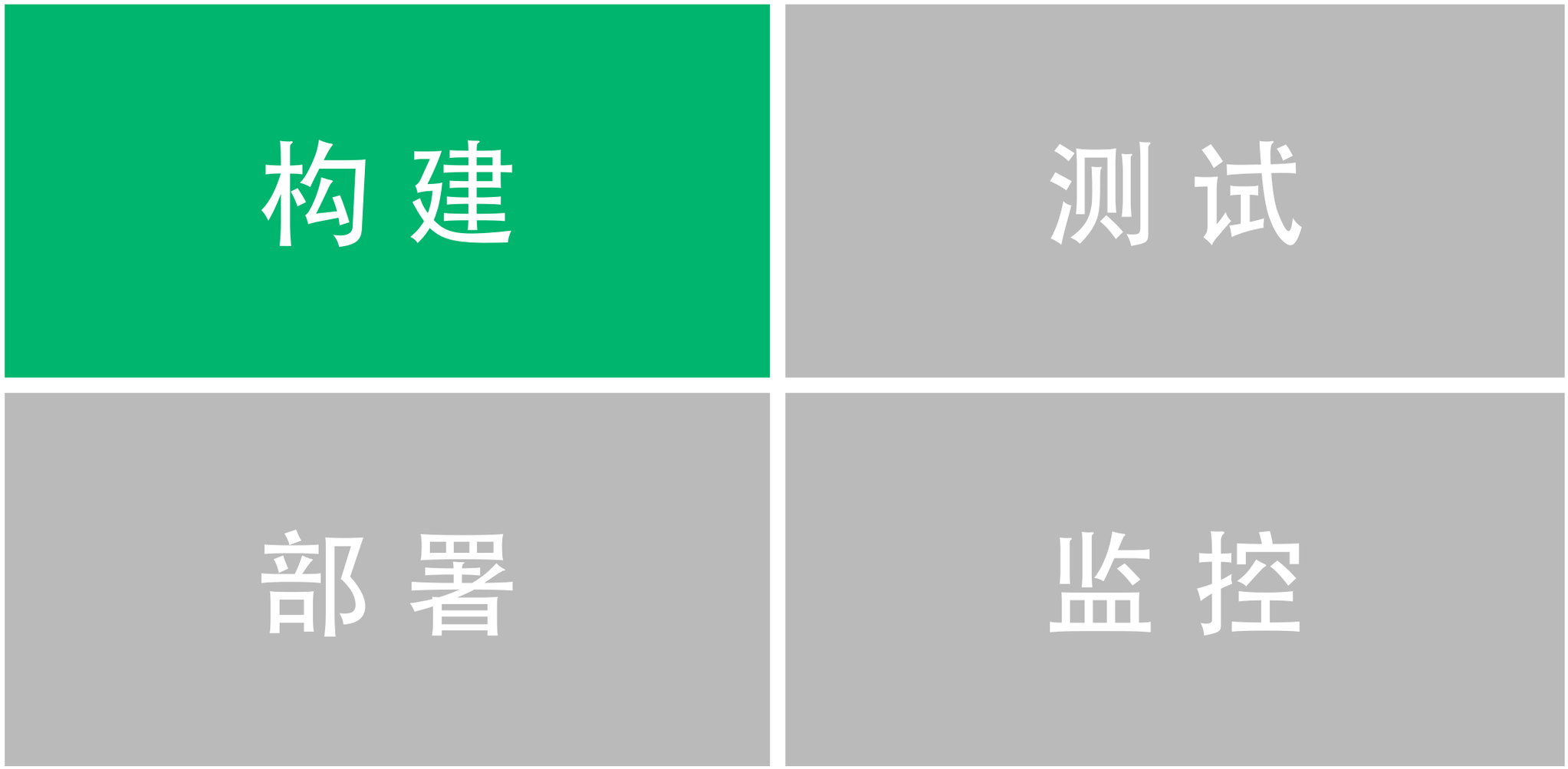
让我们的系统尽可能

快的　　低成本的　　响应变化

微服务实践



构建



构建微服务

- Job微服务的构建
- Web微服务构建

JOB 微服务构建

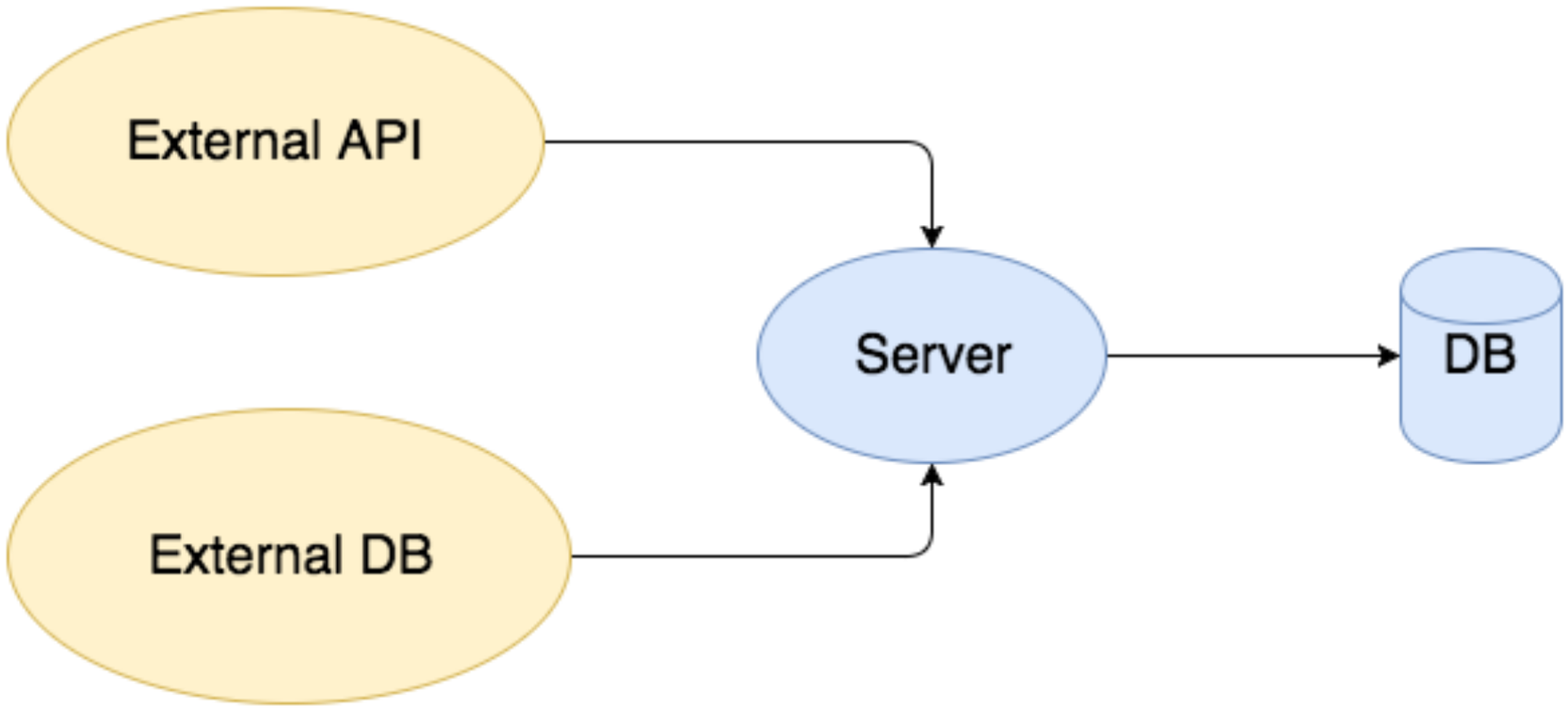
- 多为内部需求
- 没有UI展现
- 后台运行

eg：提取处理用户数据，数据分析处理，订单扫描处理...
有两个外部数据源A和B，A为API，可以拿到最新的用户属性数据，B为Datasource如S3/DB,存储着用户的操作数据。
需求是：提取这两个数据源的数据然后以天为单位来计算每天的用户数据。

传统的单体Job

特点：

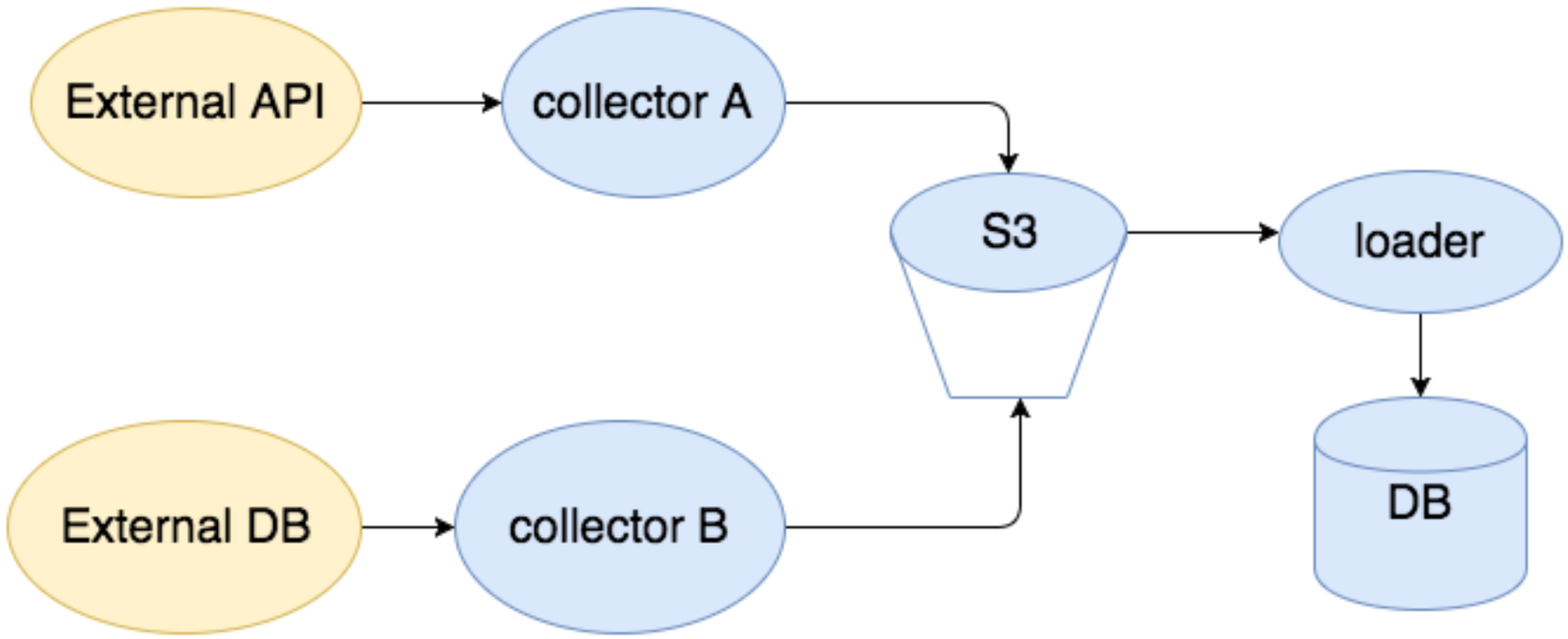
- service功能大而全，逻辑复杂
- 难于扩展，维护性差
- 一个逻辑失败需要重新运行整个job，耗时长
- 不会保存中间数据



Job微服务架构

优点:

- 按照业务功能划分，每个job只做一件事
- 可以独立部署
- 易于扩展功能
- 迅速恢复
- 会保存中间数据



Job微服务架构出现的问题

- 不易管理job的执行流程
- 代码重复度高
- 数据源问题

管理Job执行流程

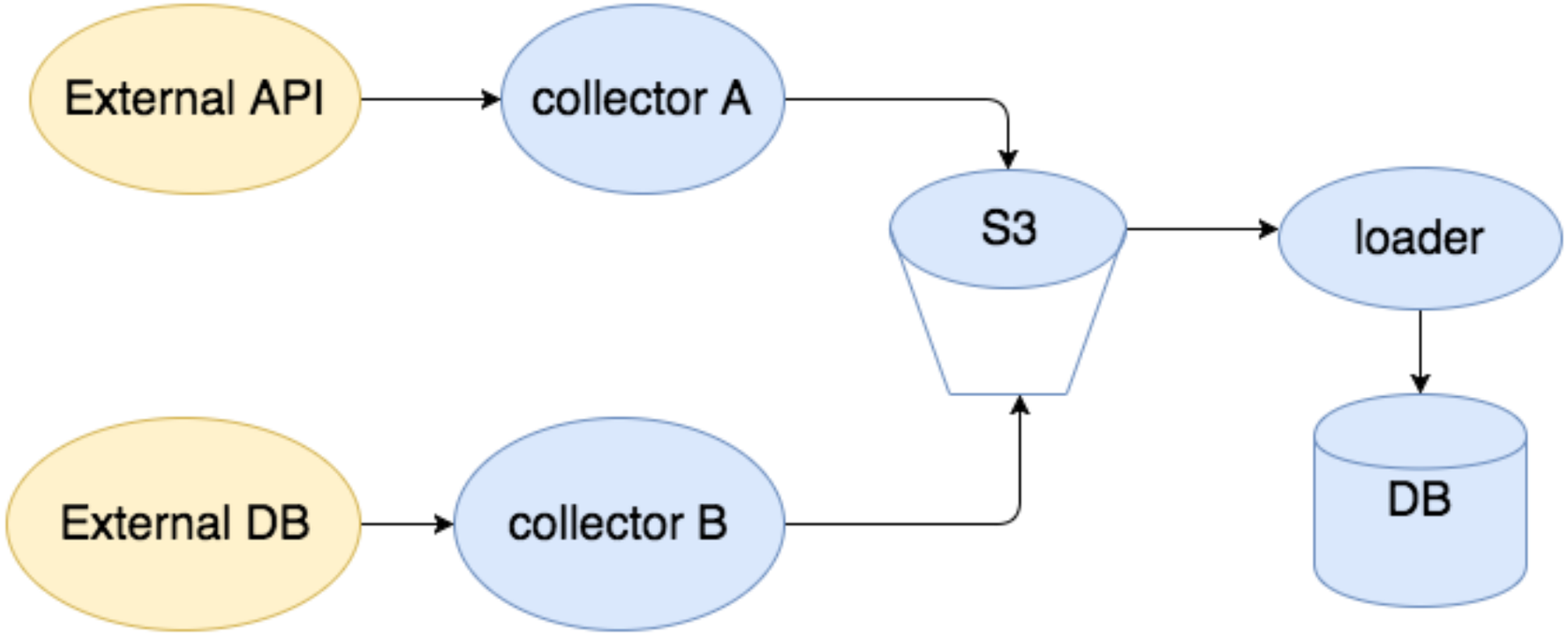
- 时间驱动
- 事件驱动

管理Job执行流程

- 时间驱动

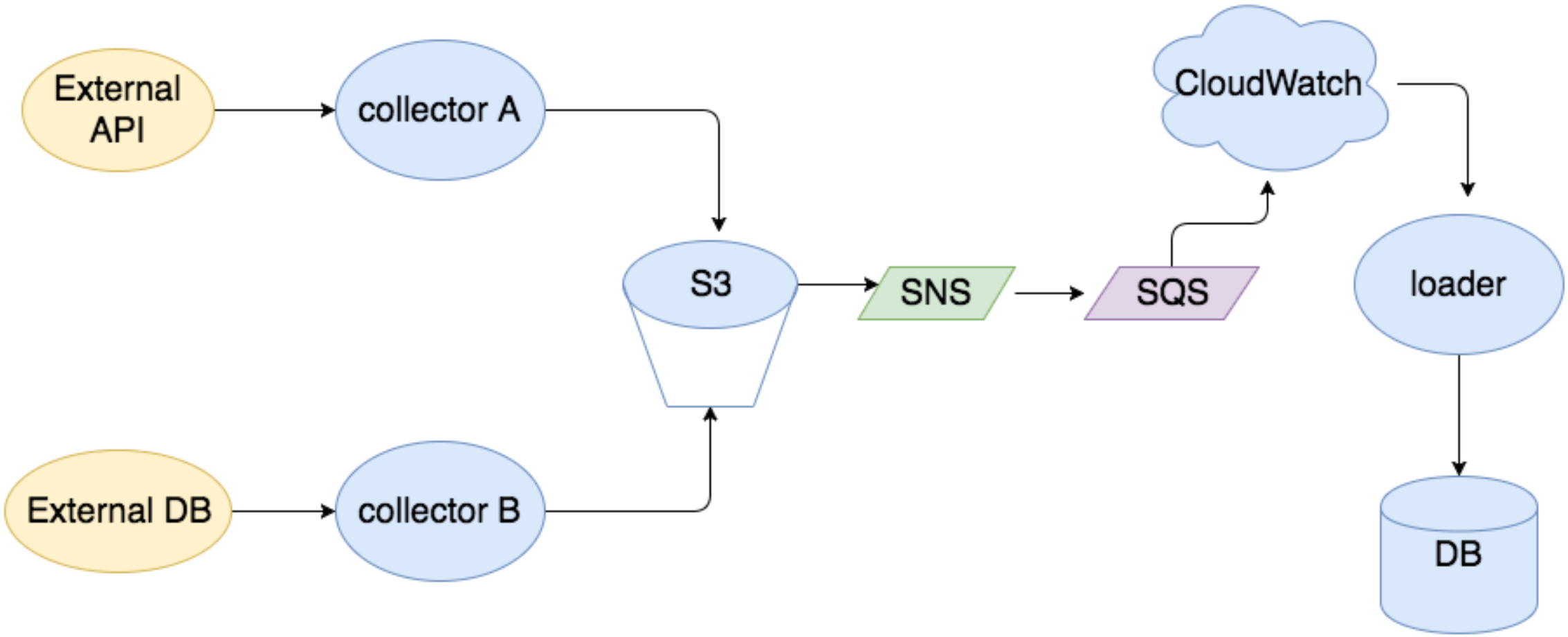
缺点：

- 时间预估总是没有最佳值
- 到点执行
- 一挂全挂



管理Job执行流程

- 事件驱动



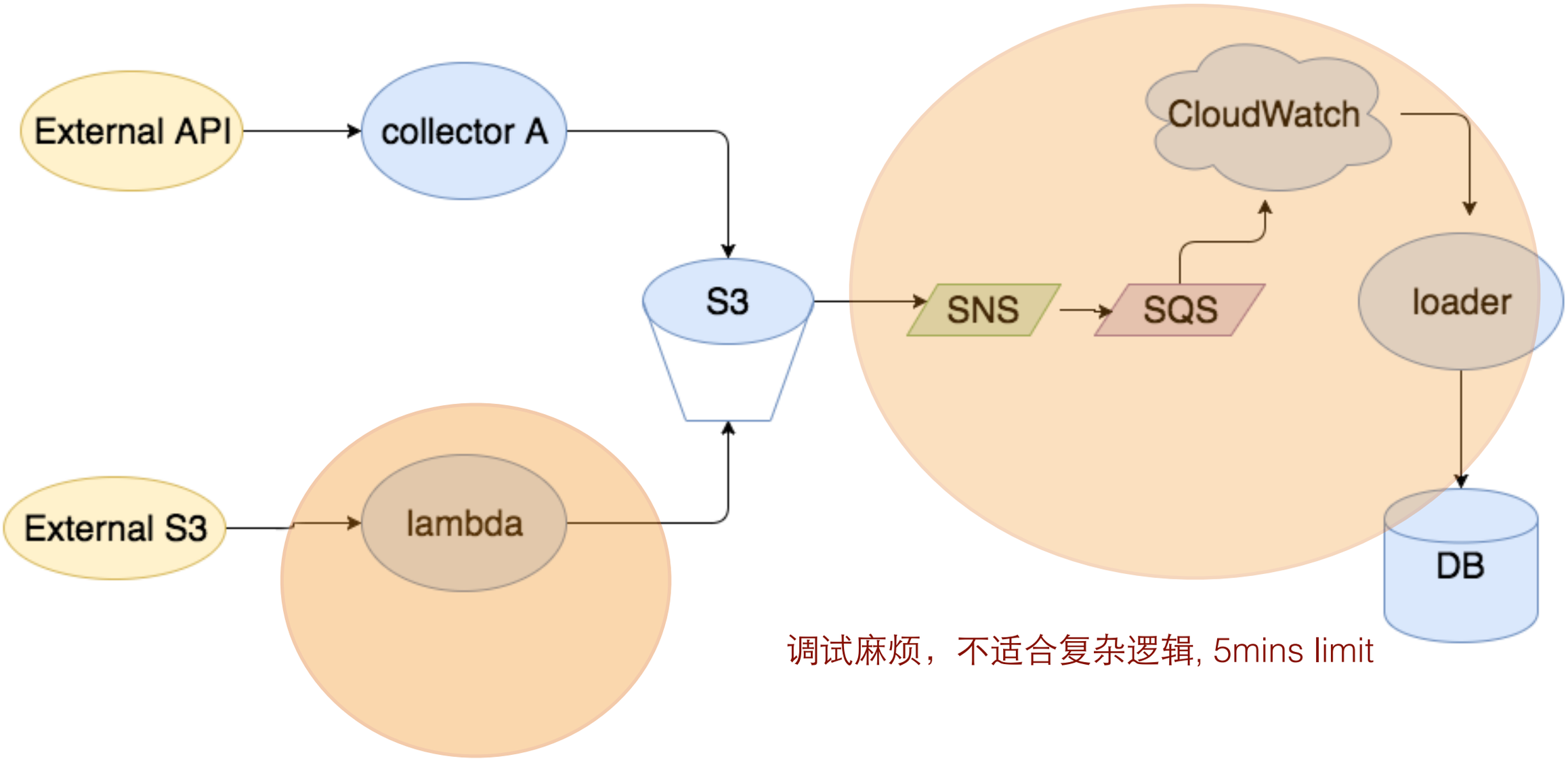
SNS/SQS/CloudWatch

JOB 微服务构建

管理Job执行流程

- 事件驱动

- 自动trigger程序执行
- 省去分配资源并部署机器的麻烦
- 便宜，只需为请求的次数和计算的时间内存付费



调试麻烦，不适合复杂逻辑, 5mins limit

lambda trigger

代码重复

- 抽离出公共的组件，如gem（S3, Log, File Ancestry)

数据源变化

- 选择可靠的第三方
- 选择合适的提取方法（box, SES/S3）

JOB 微服务构建

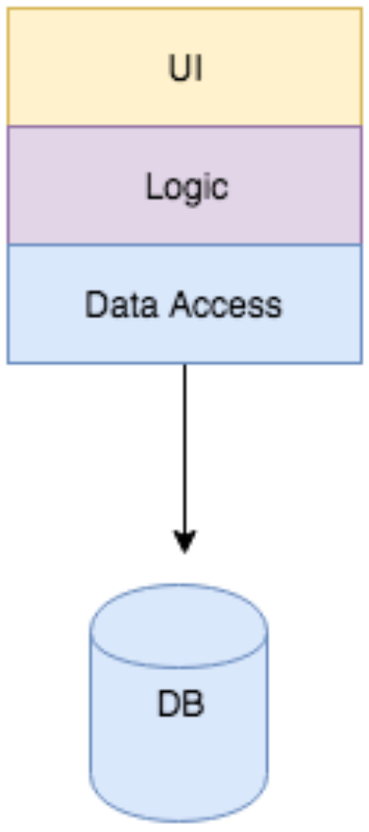
Tips:

- Job流程管理方式
- 抽离公共组件
- 根据数据源选择服务组件

Web微服务的构建

需求:
给客户做一个广告管理平台，客户可以生成订单，查阅订单，
并且可以查看购买所有广告的用户操作数据。

传统的单体服务：



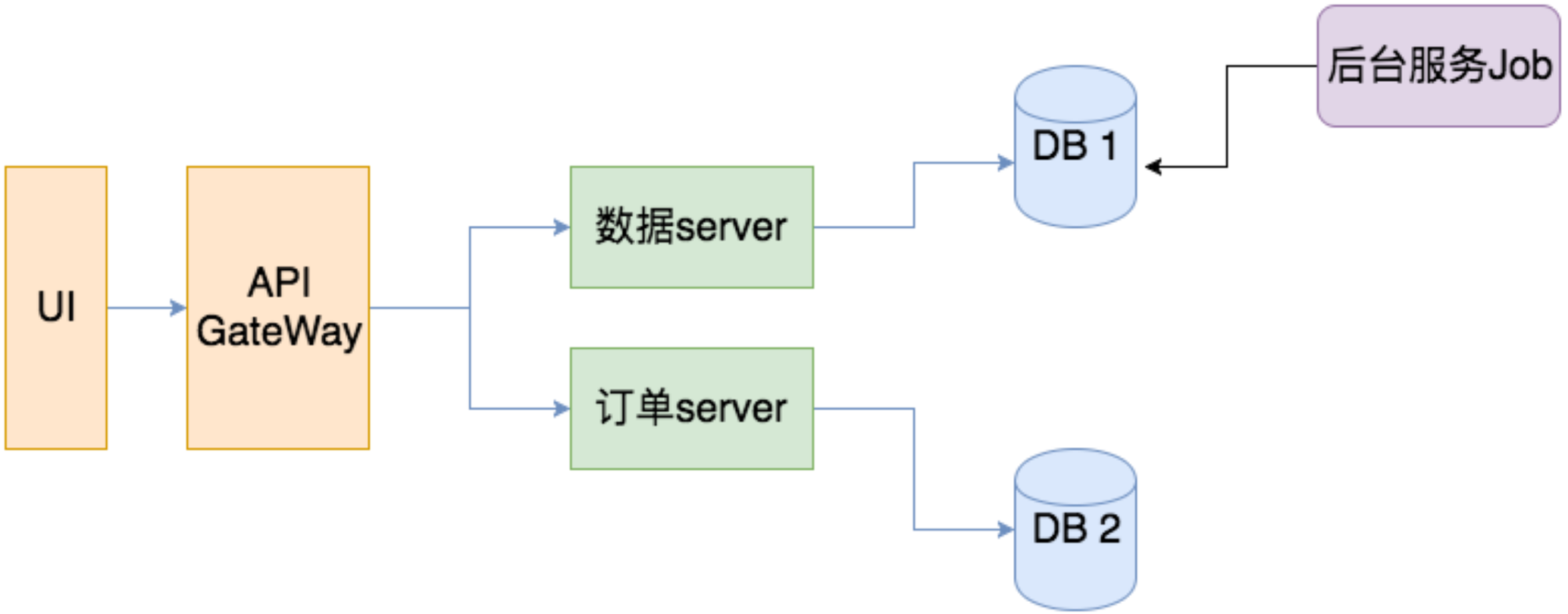
- server功能大而全，逻辑复杂
- 难于扩展，维护性差
- 模块间的界限模糊，随着项目有增长改一个功能需要修改多个模块

WEB 微服务构建

微服务架构:

优点:

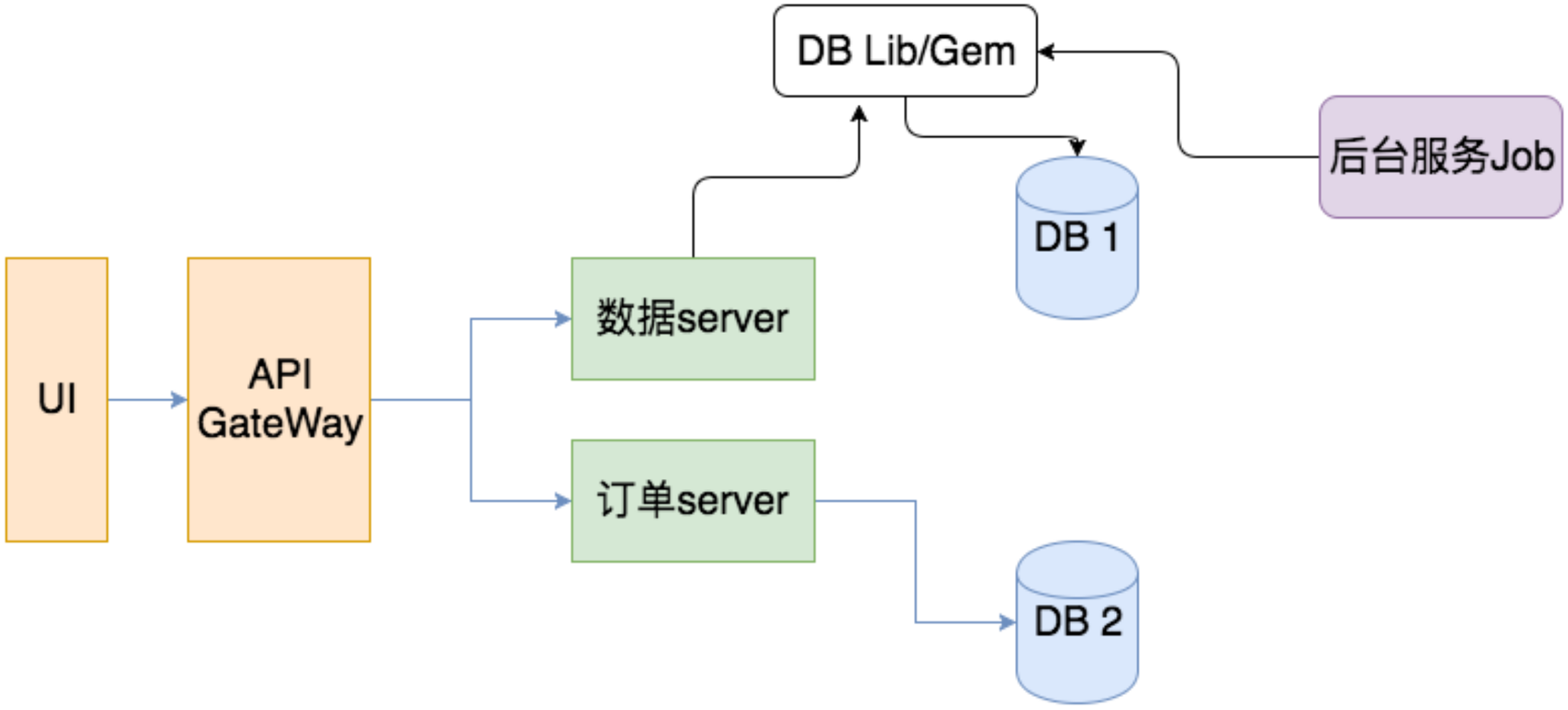
- 按照业务功能划分，单独部署
- 易于扩展功能



出现的问题：

- 多个service 访问相同的数据库，都需要考虑到数据库中的表结构之间的业务关系，操作复杂
- API通信消耗

多个service 访问相同的数据库，业务逻辑重复



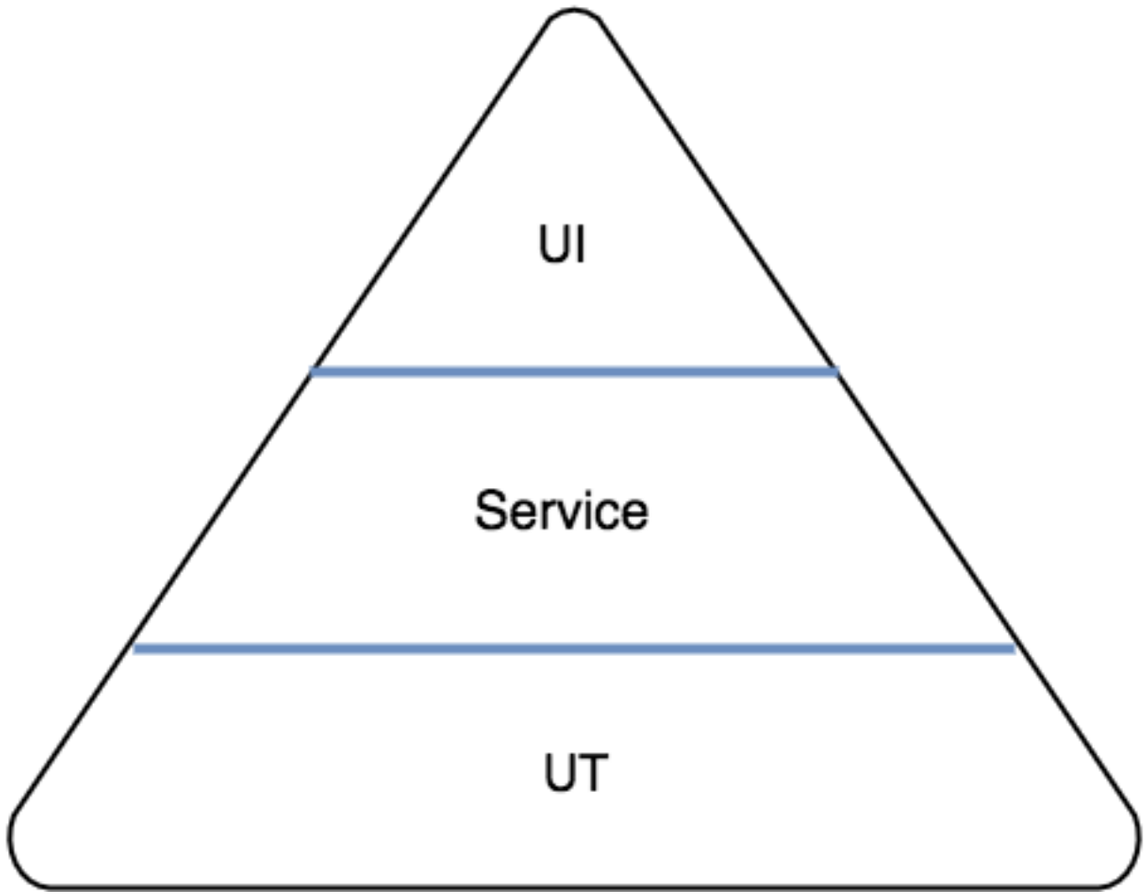
- DB Gem，简单的接口调用，不用担心过多的表结构业务关系
- 提供统一的API server

API 通信

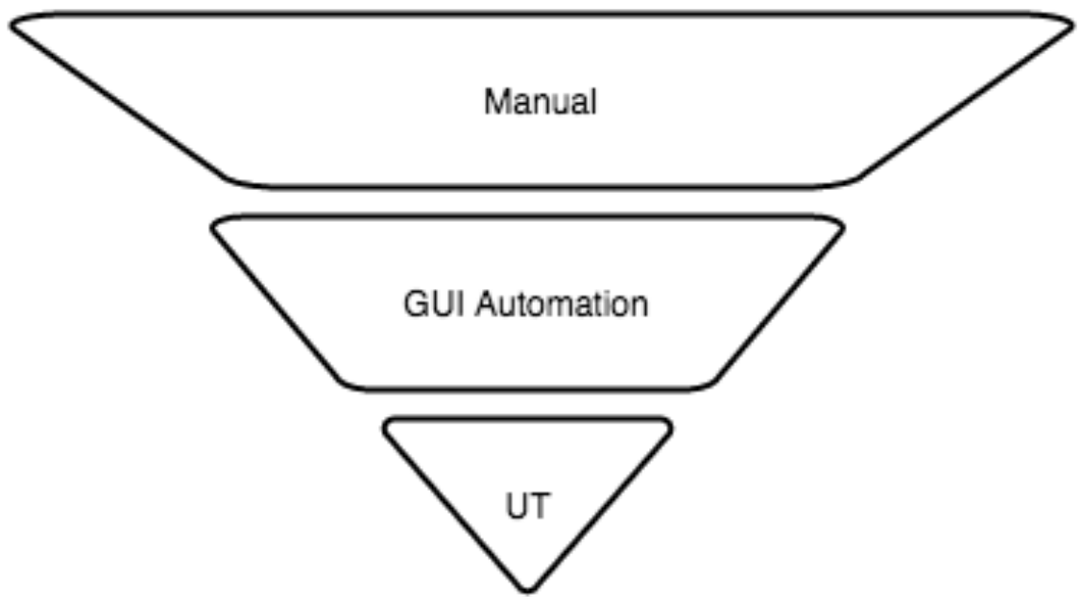
- 轻量级
- 粗粒度

测试

测试系统的构建



单体系统的测试

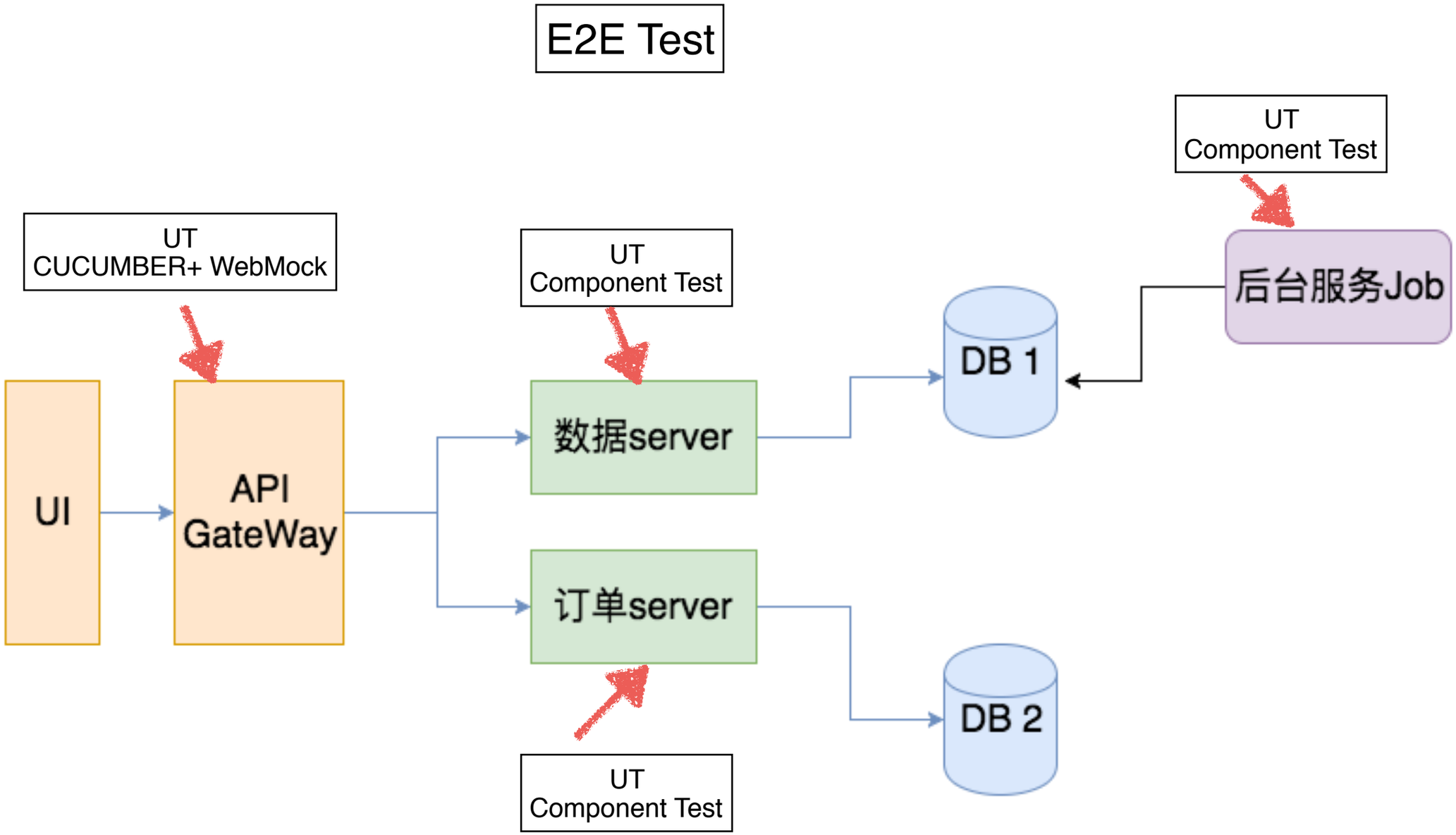


系统不易拆解，使得模块化的系统测试比较困难

微服务构建下的测试

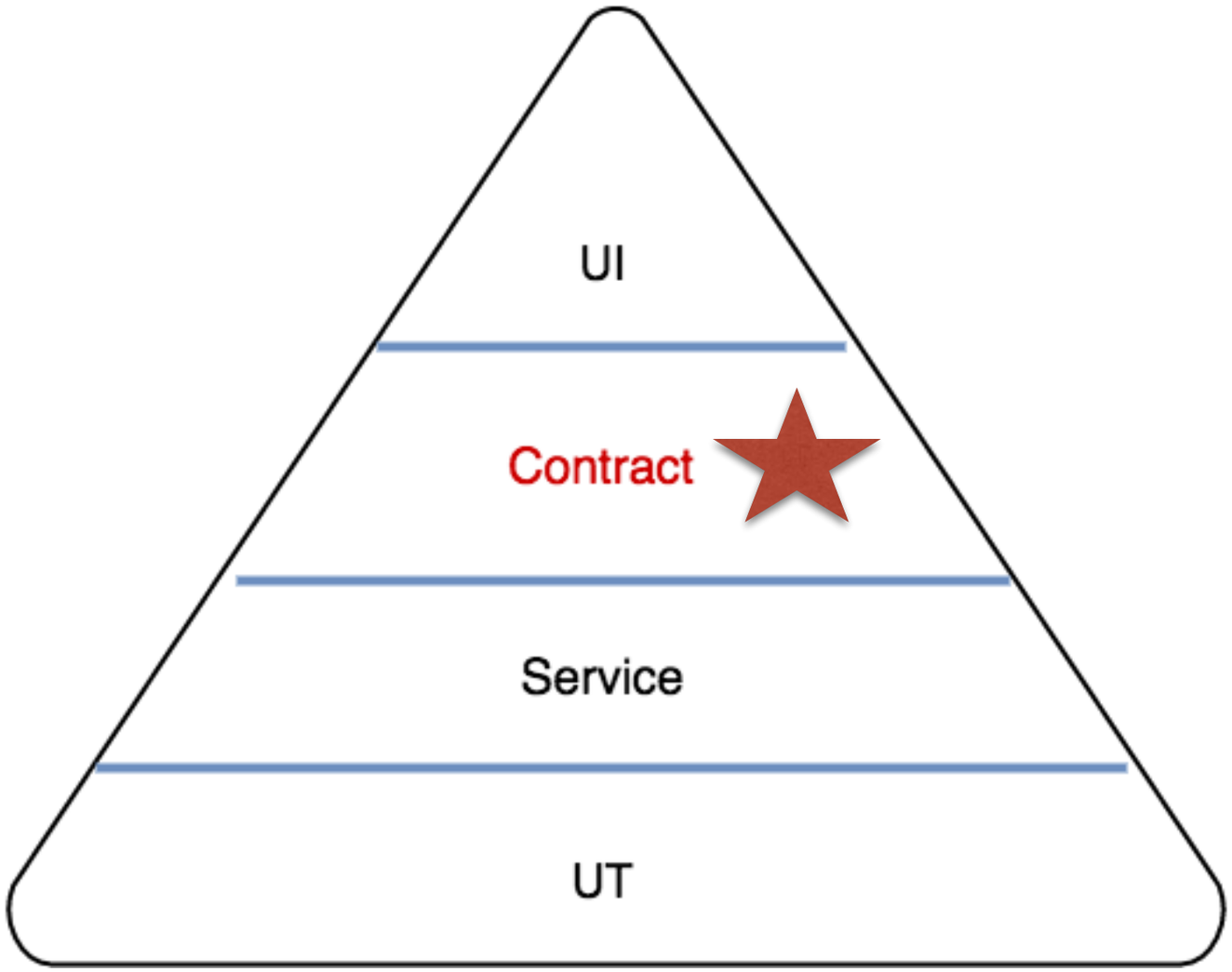
- 每一个微服务都是一个独立的功能模块。
- 一般使用Resful方式对外暴露接口
- 各位微服务之间的通讯方式更加多样。

微服务构建下的测试



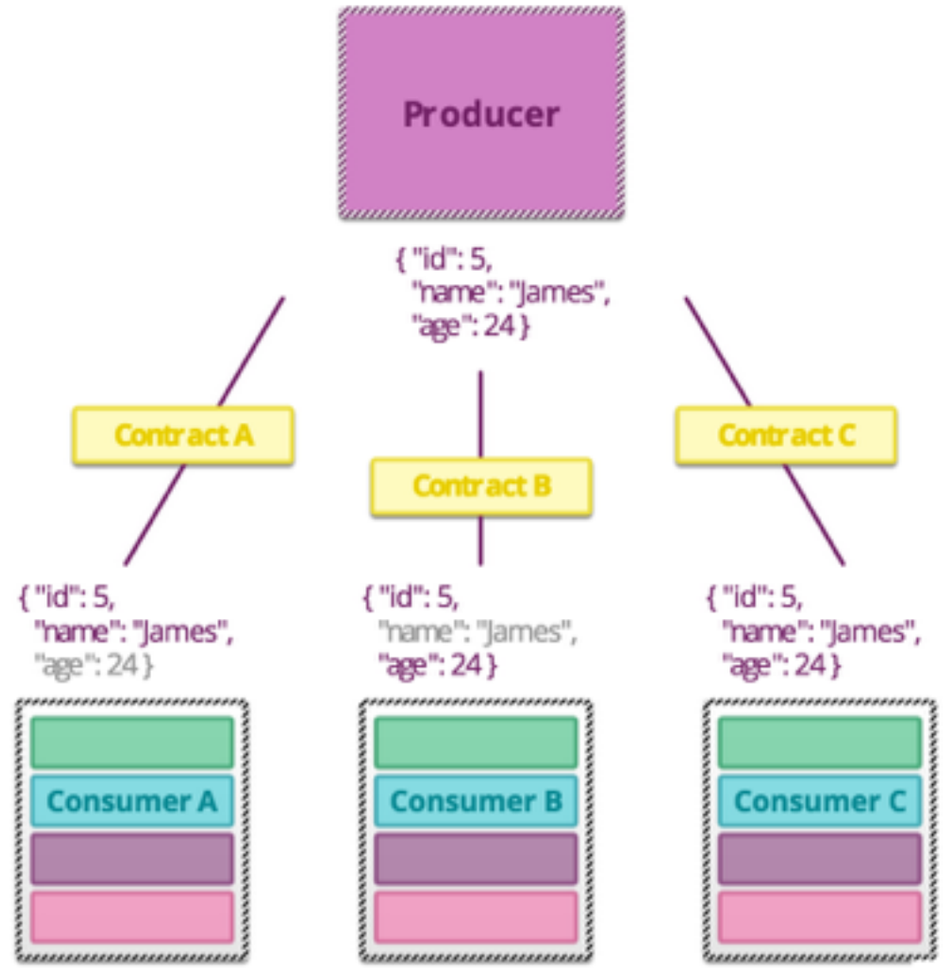
少点什么?

微服务构建下的测试金字塔



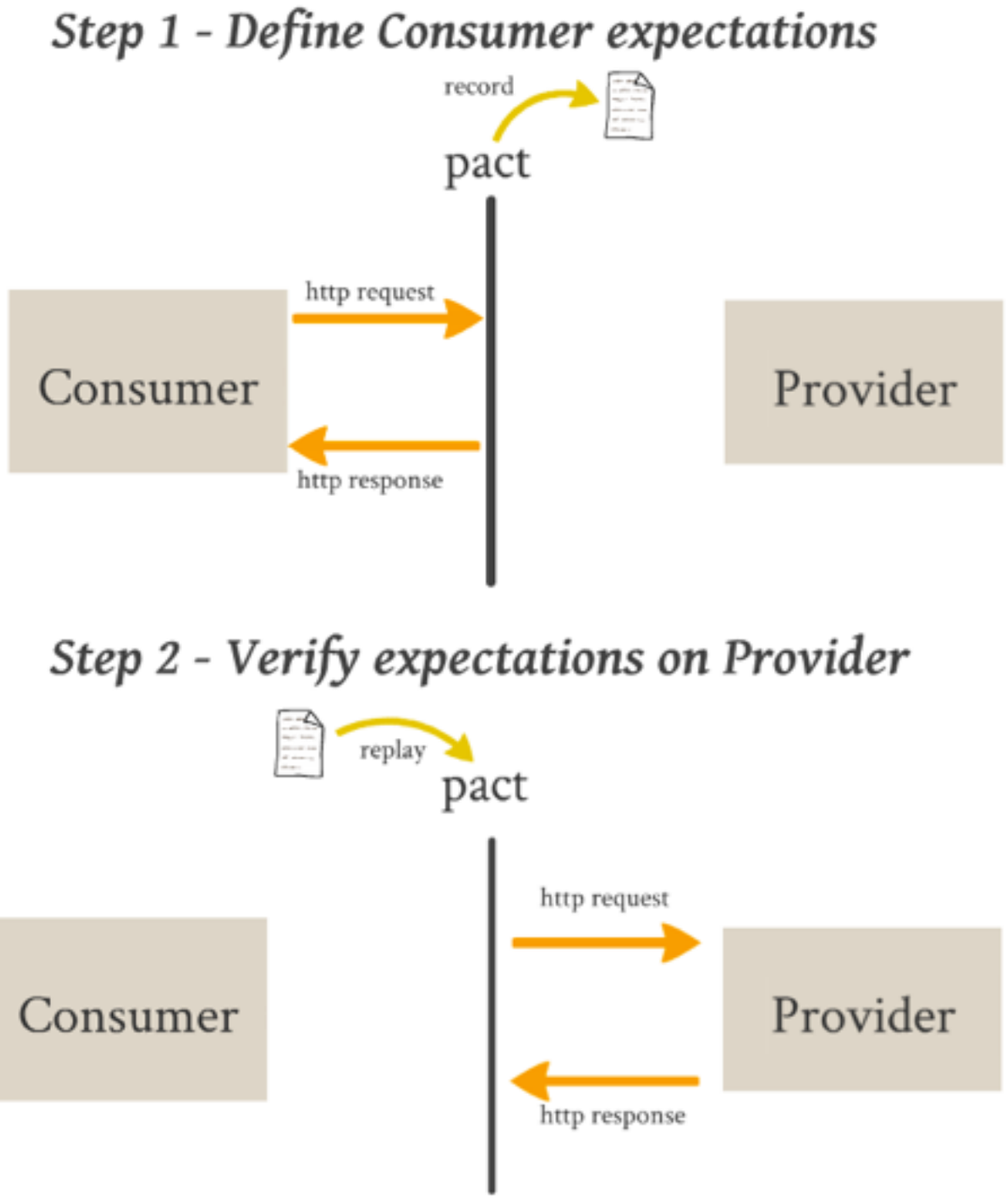
契约测试

An integration contract test is a test at the boundary of an external service verifying that it meets the contract expected by a consuming service.

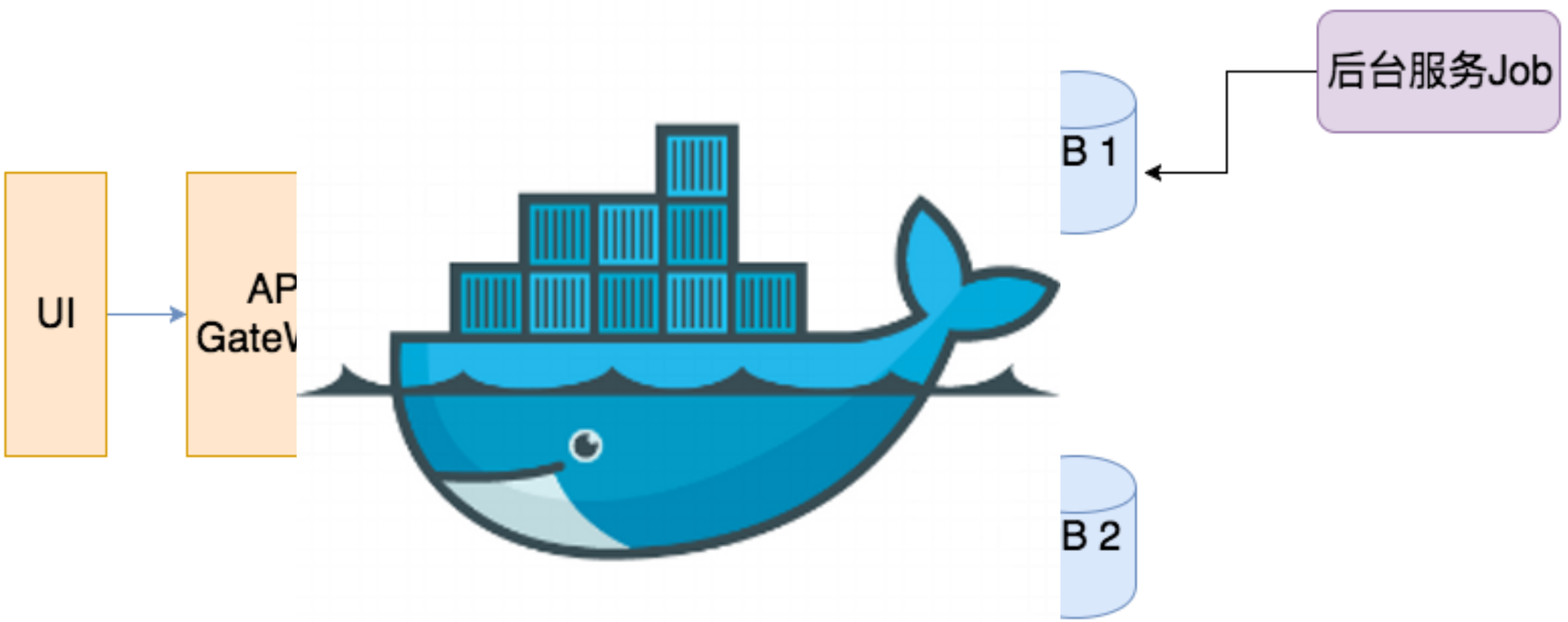


契约测试的工具

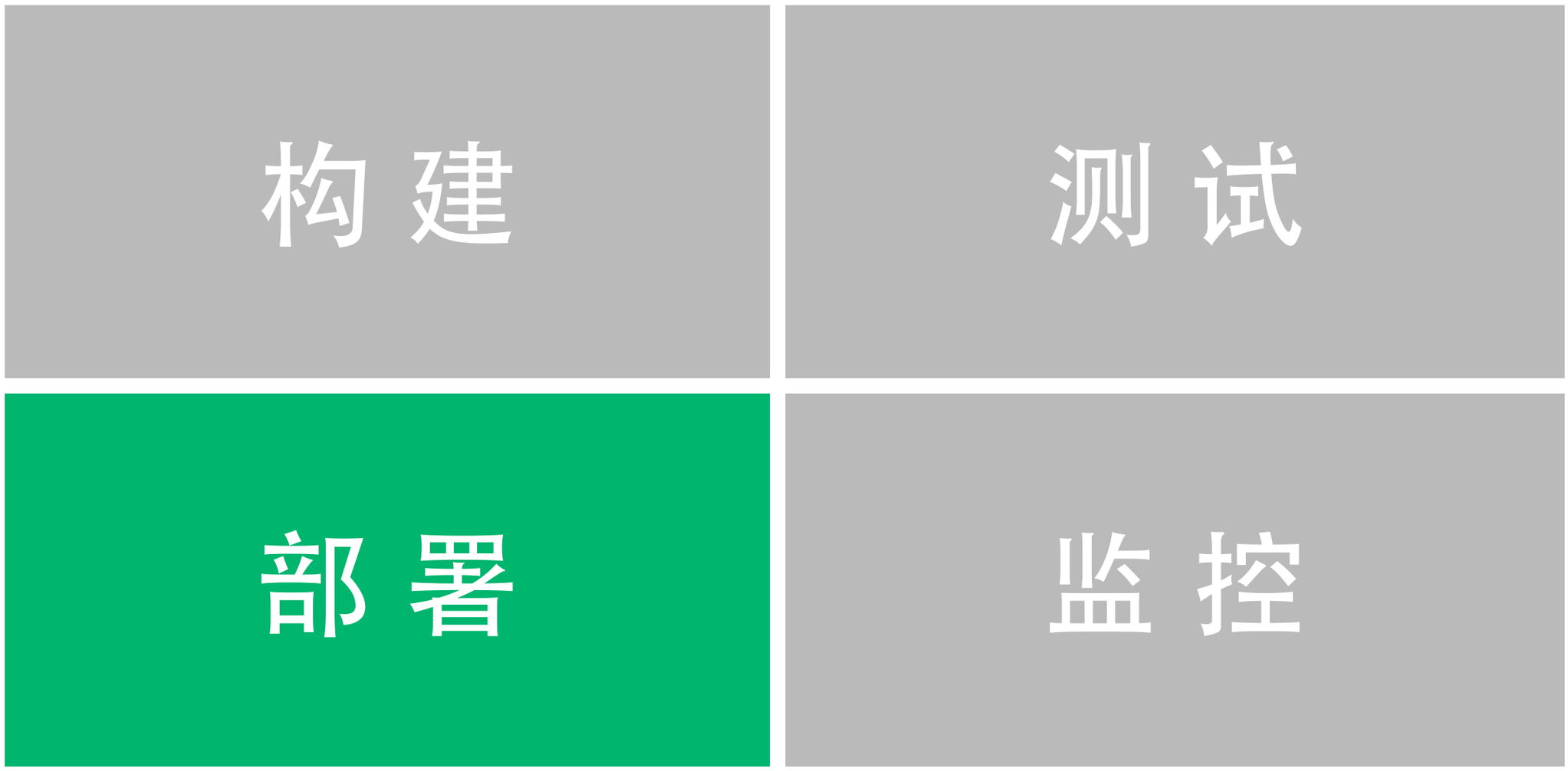
- 自己搭建
- 使用契约
  - PA
  - PA
  - JA



微服务下的测试

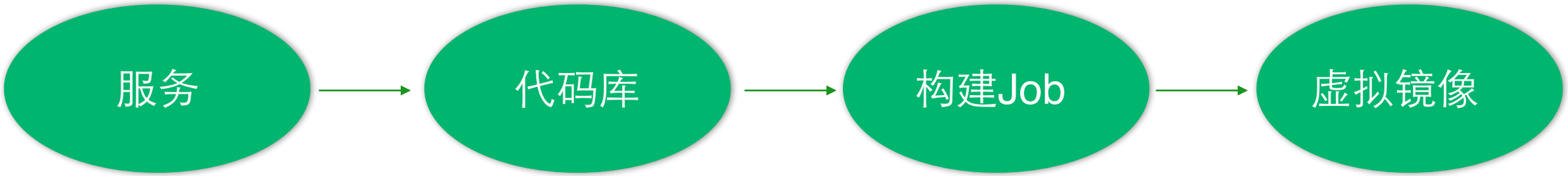


部署



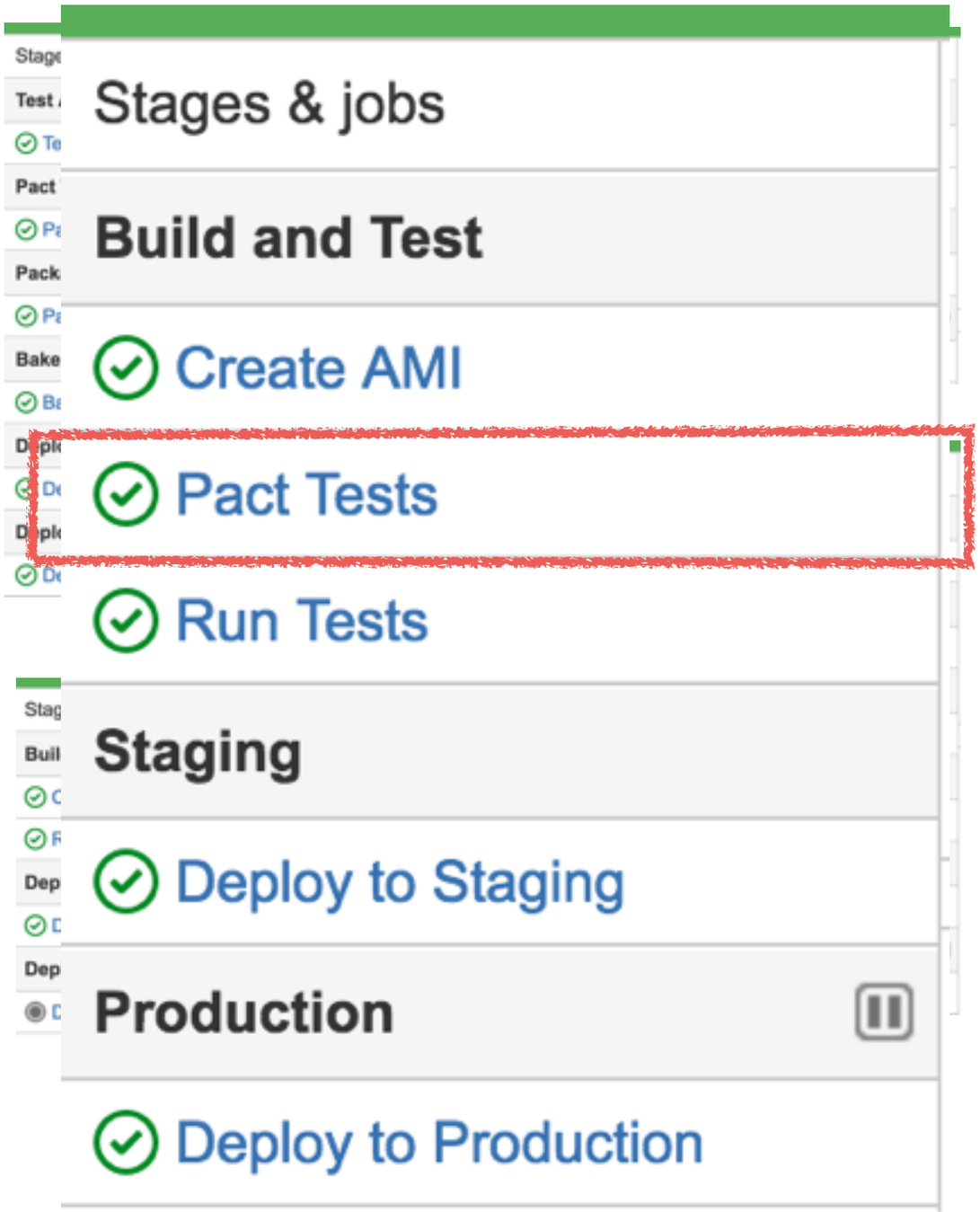
持续集成(CI)

- 独立部署，独立运行

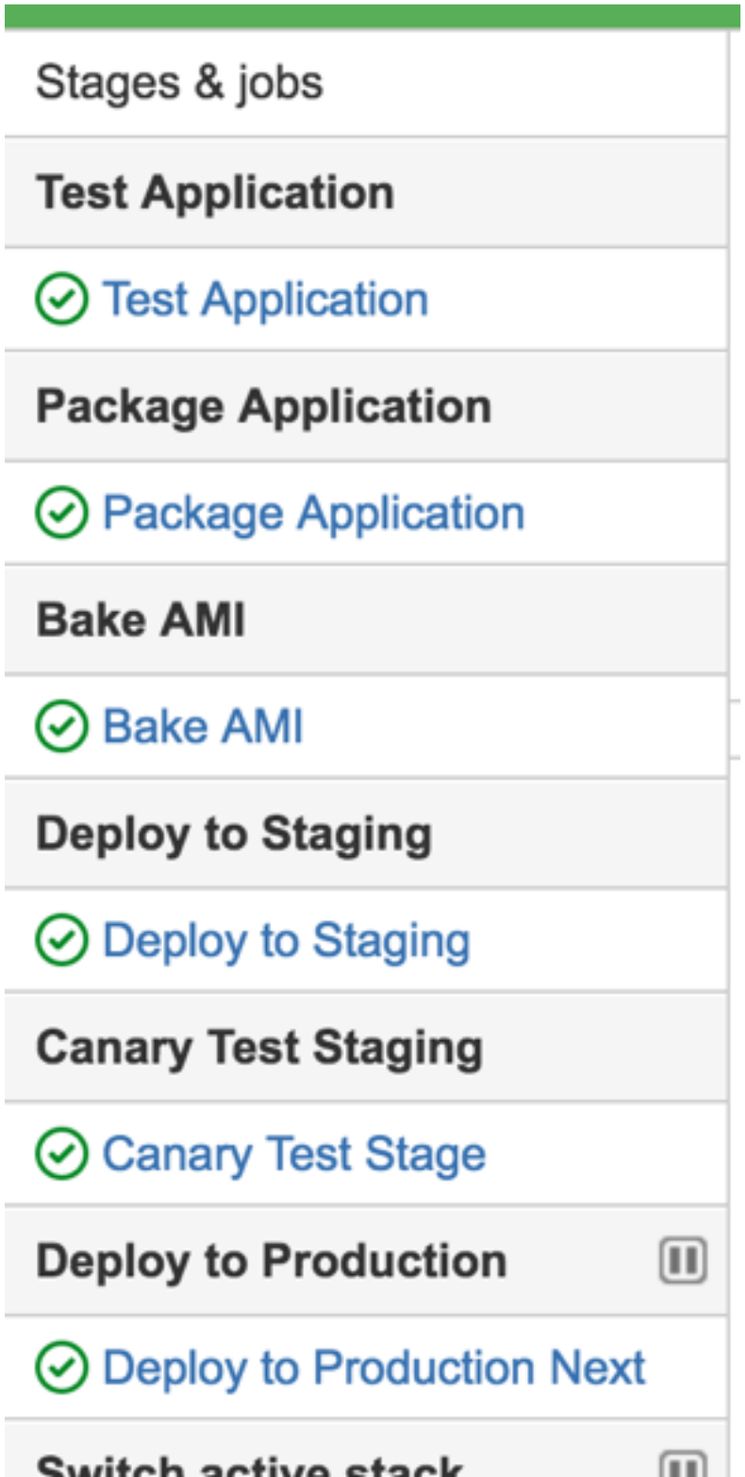


持续集成(CI)

微服务

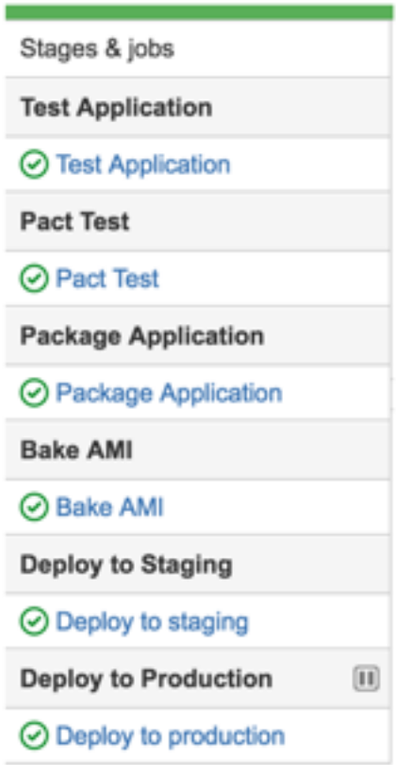


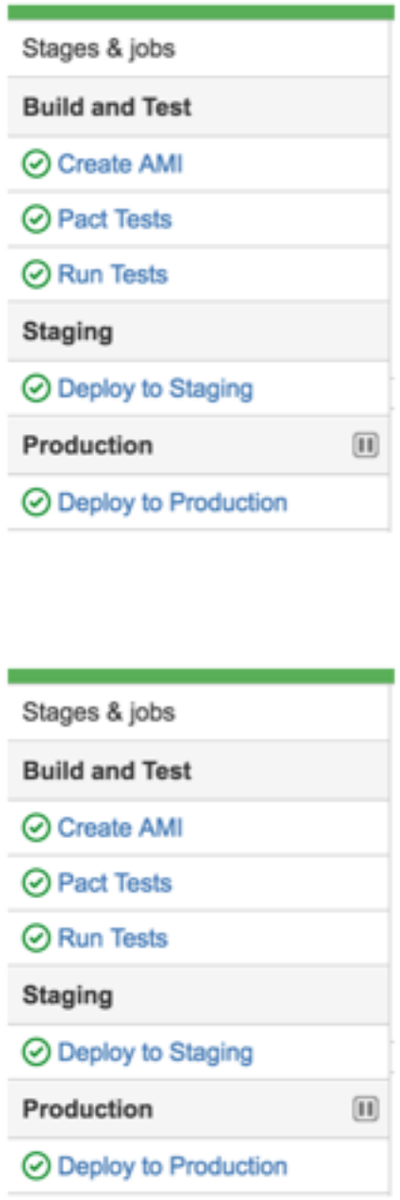
单体

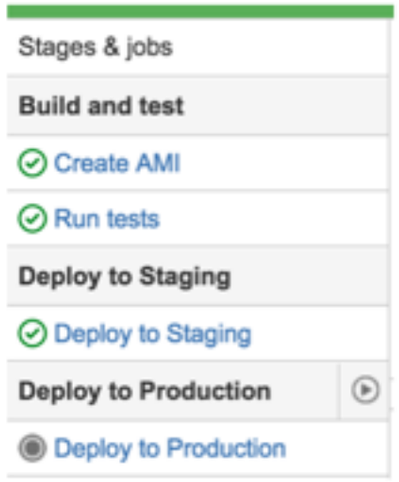


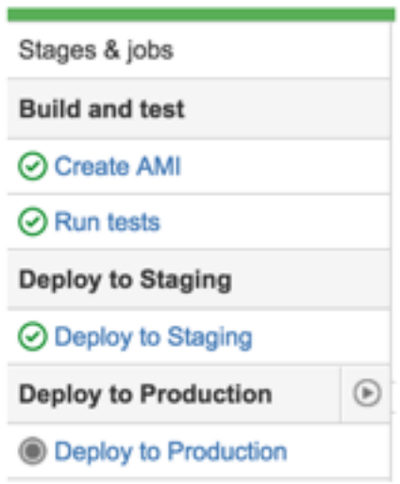
持续集成(CI)

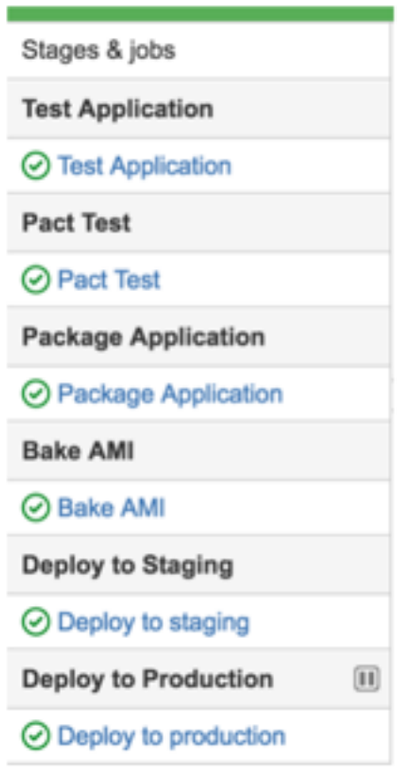
□ 微服务CI









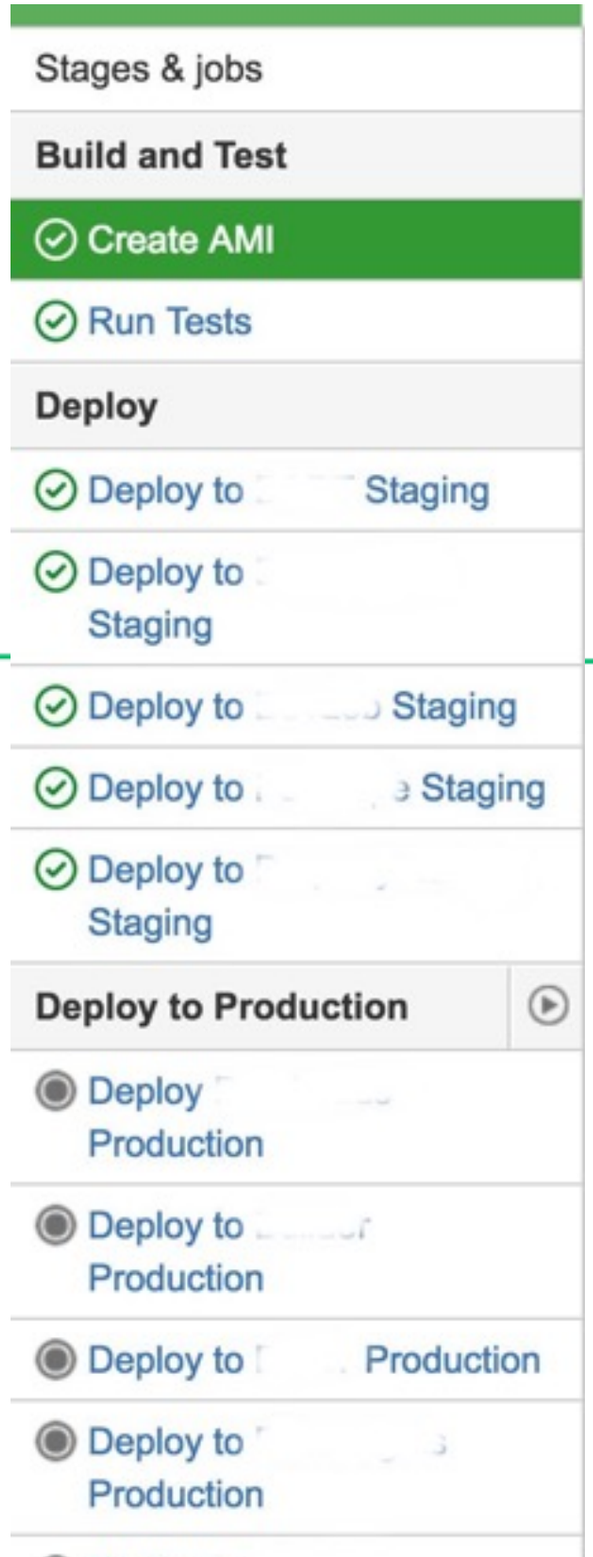


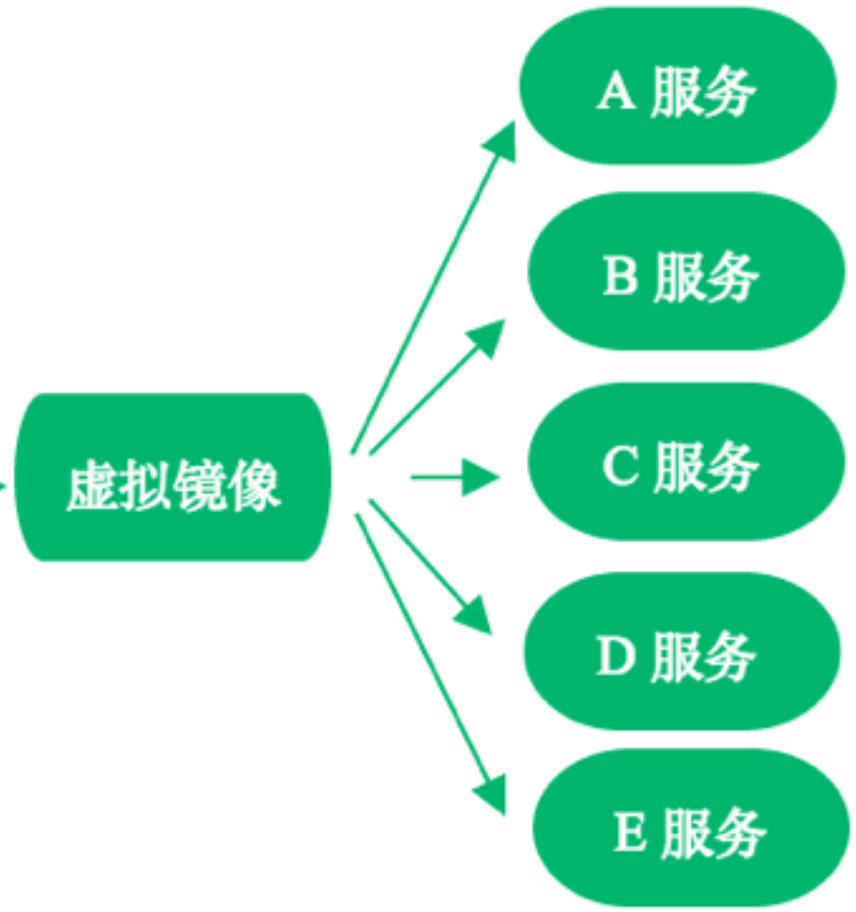
如果是单体向微服务迁移呢?

持续集成(CI)

- 迁移初期

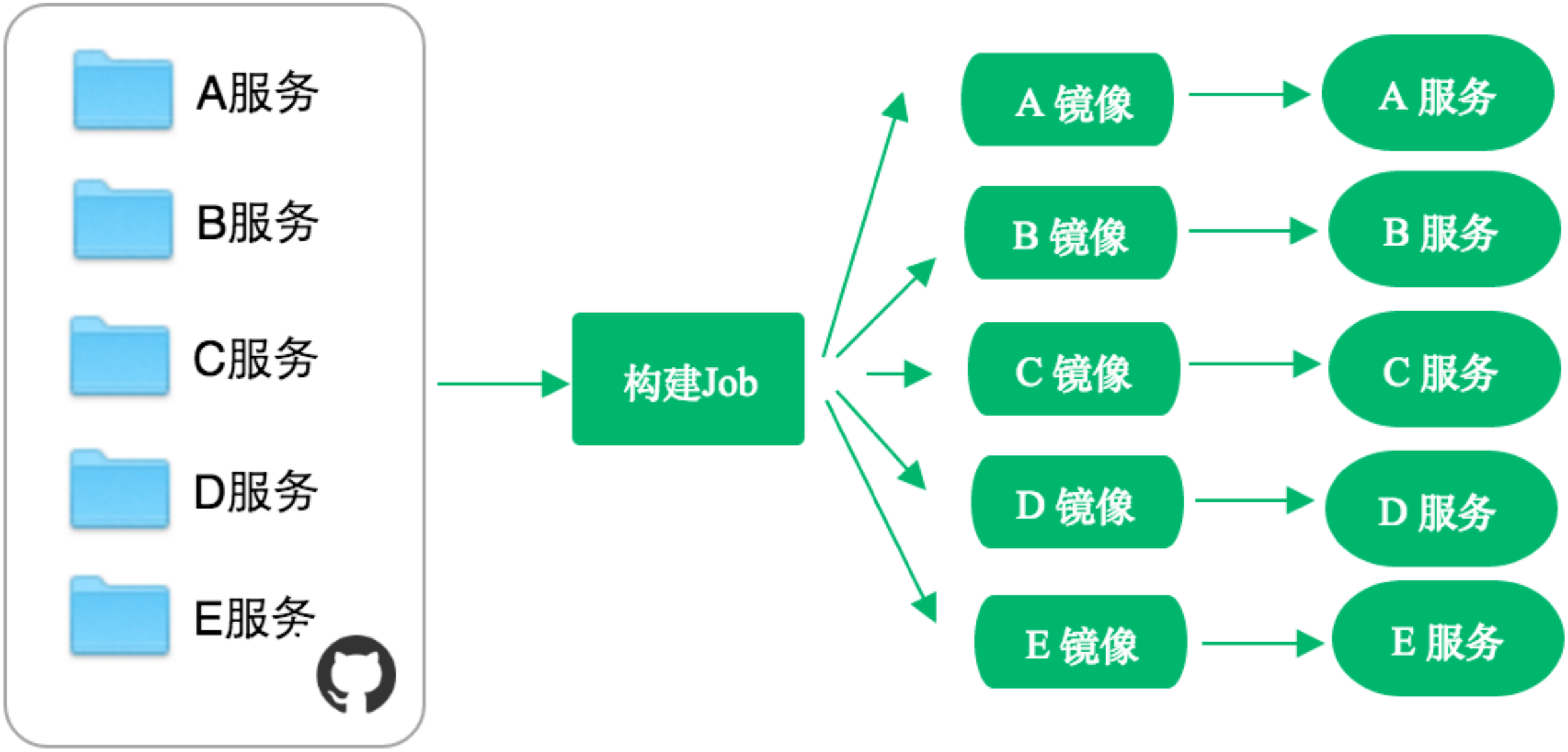

B 服务
A 服
C 服务
E 服务
D 服务


Stages & jobs
Build and Test
Create AMI
Run Tests
Deploy
Deploy to Staging
Deploy to Staging
Deploy to Staging
Deploy to Staging
Deploy to Staging
Deploy to Production
Deploy Production
Deploy to Production
Deploy to Production
Deploy to Production
Deploy to Production


虚拟镜像
A 服务
B 服务
C 服务
D 服务
E 服务

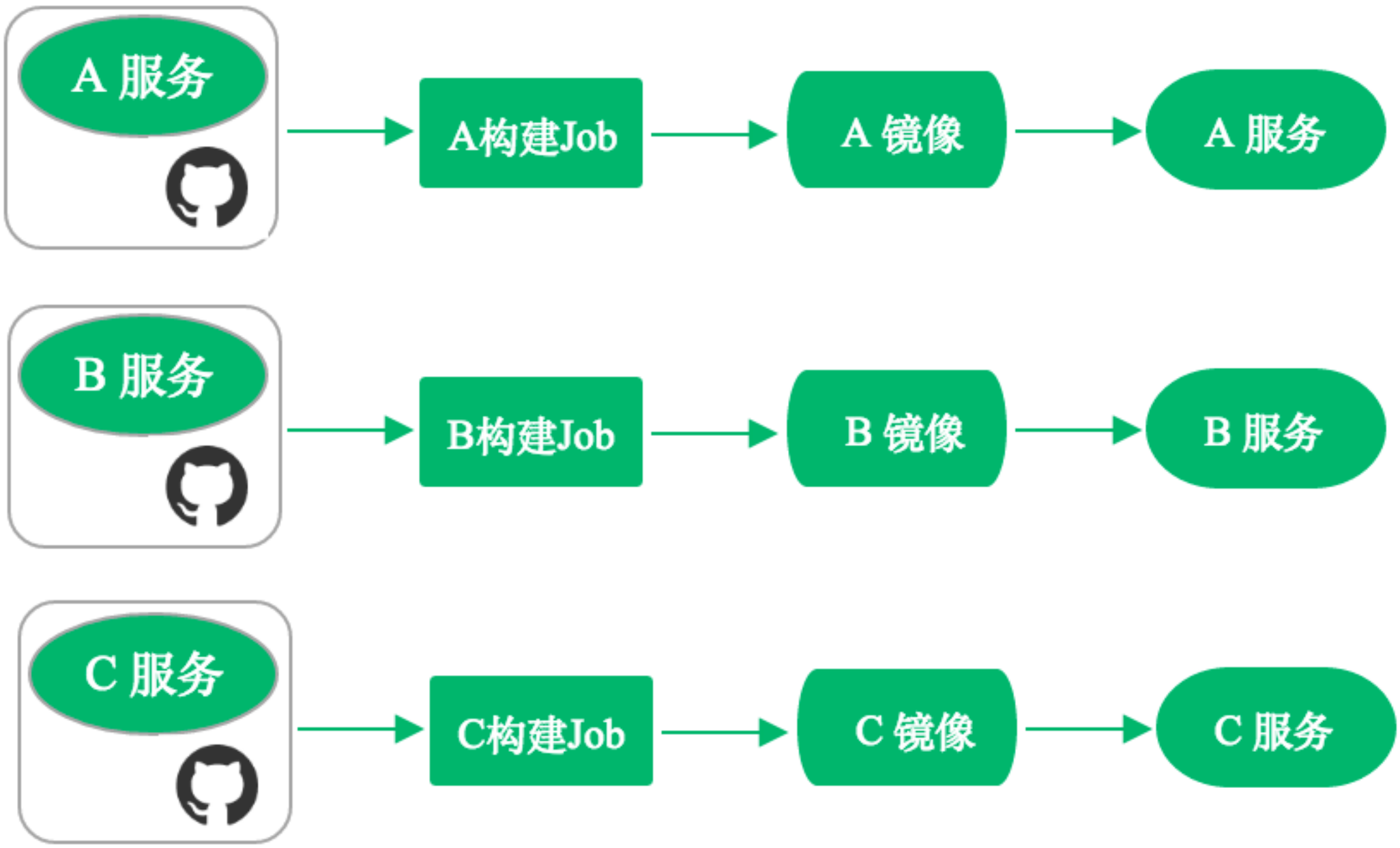
持续集成(CI)

稳定期


A服务
B服务
C服务
D服务
E服务
构建Job
A 镜像
A 服务
B 镜像
B 服务
C 镜像
C 服务
D 镜像
D 服务
E 镜像
E 服务

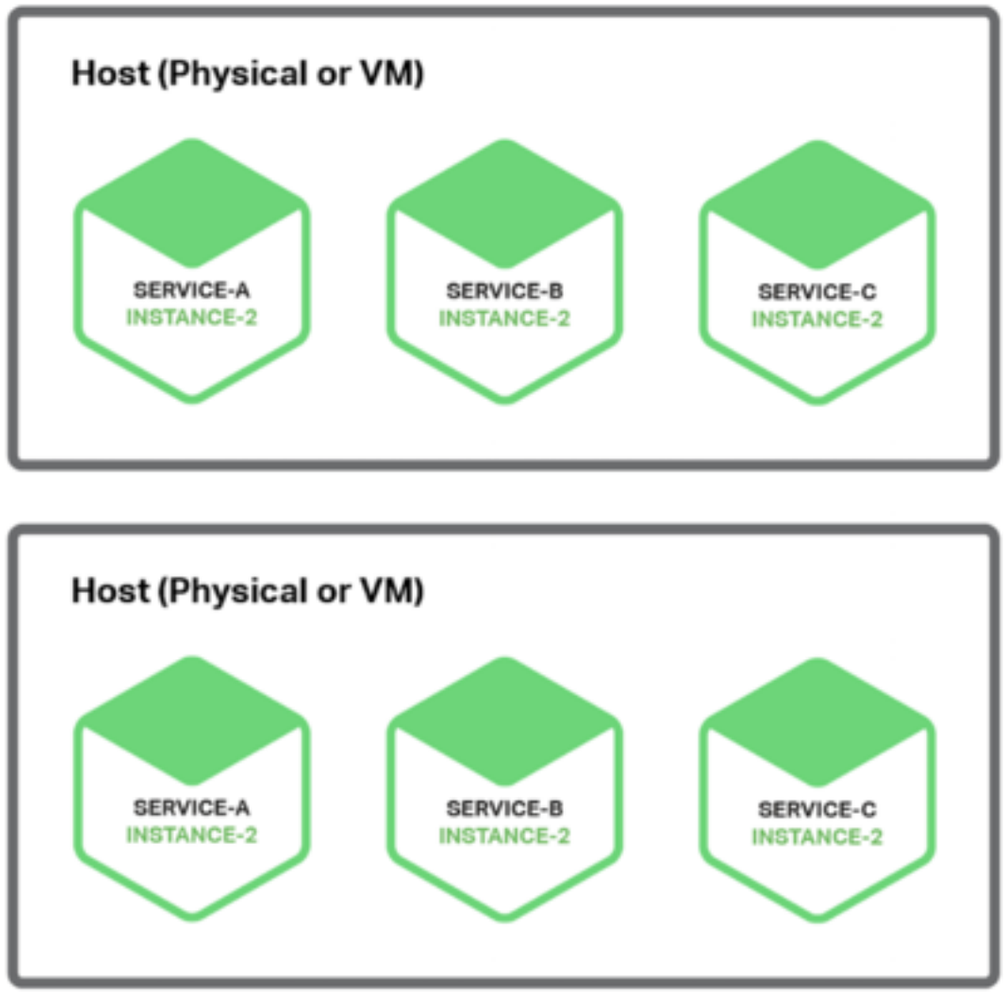
持续集成(CI)

□ 成熟期



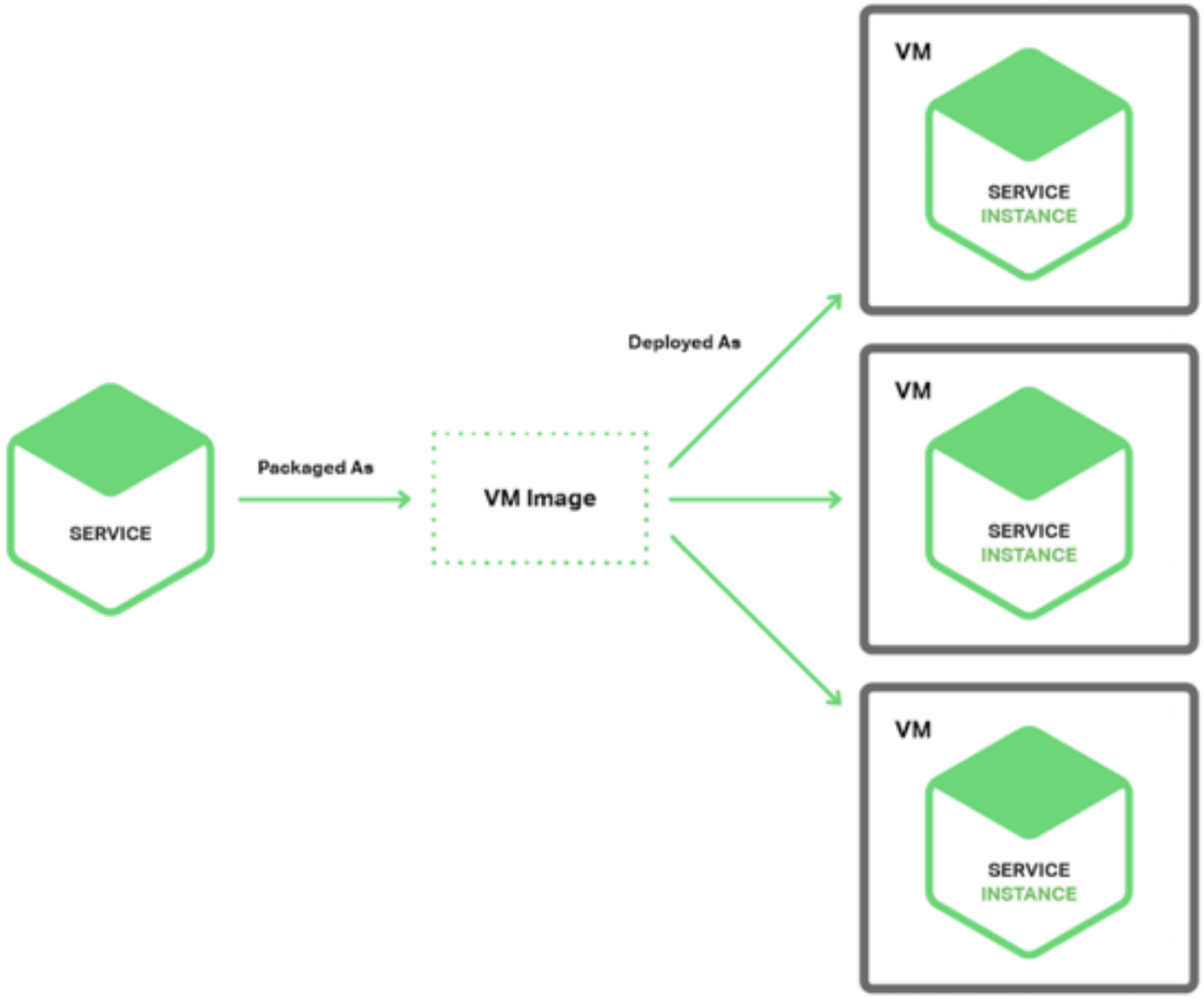
DEPLOY模式

单主机多服务



单主机单服务

□ 单主机单服务


SERVICE
Packaged As
VM Image
Deployed As
VM
SERVICE
INSTANCE
VM
SERVICE
INSTANCE
VM
SERVICE
INSTANCE

- Zip代码
- Meta Data
- 上传AWS

监控

构 建

测 试

部 署

监 控

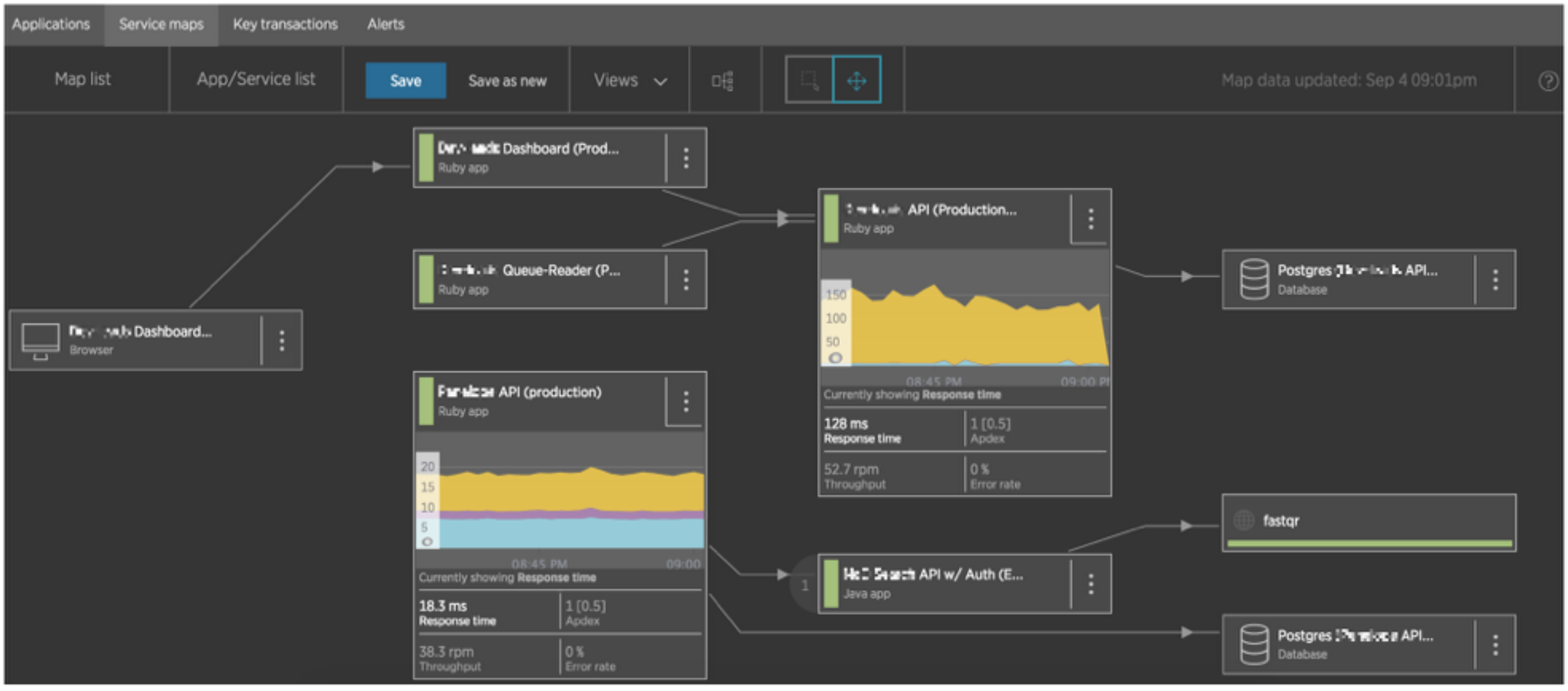
监控

- 监控 Web Service
- 监控 Job

MICRO SERVICE给监控带来的挑战

- 无法得知当前整个系统的运行状况
- 当故障发生时，无法快速发现错误的根源到底是哪一个service，对其他services有什么影响

NEW RELIC - SERVICE MAPS



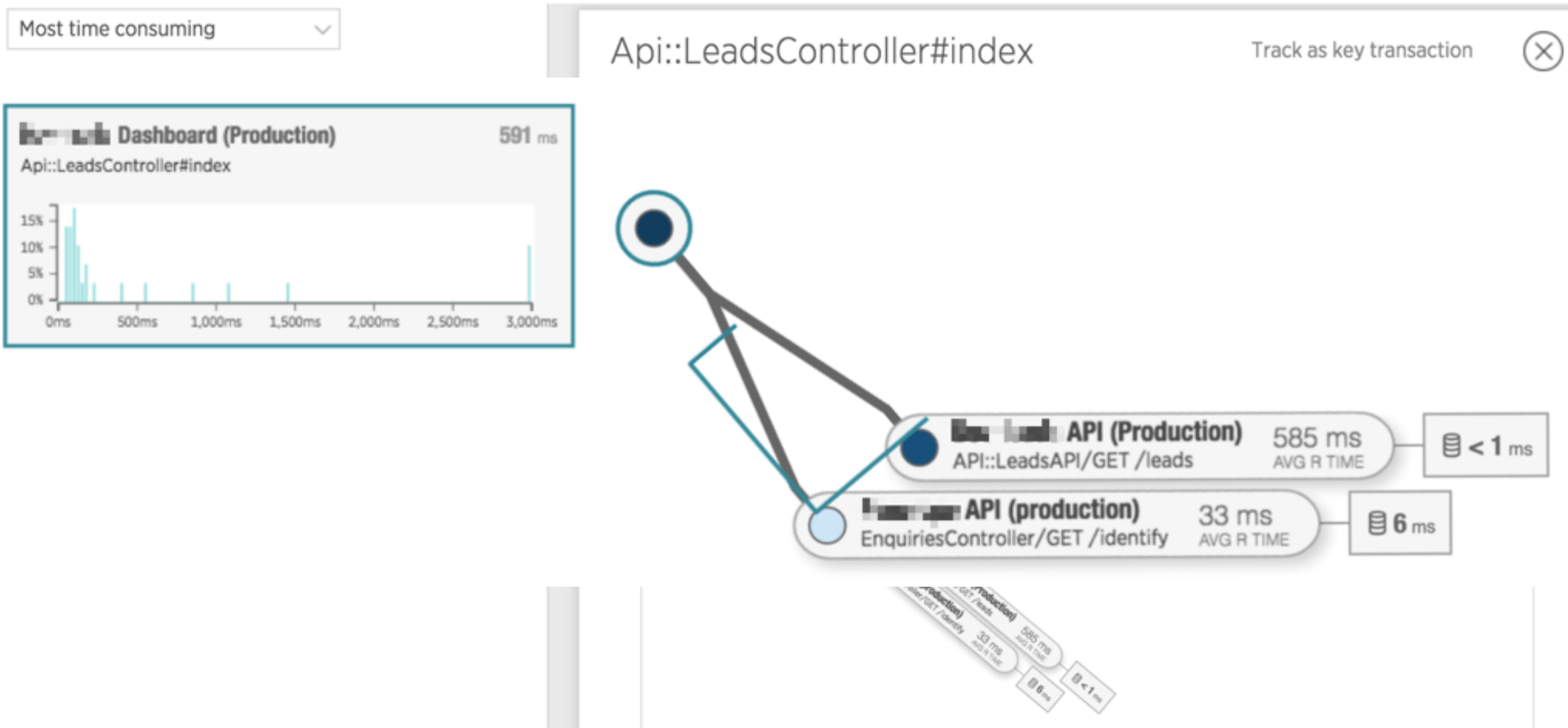
NEW RELIC - SERVICE MAPS

Service maps能够帮助我们:

- 了解系统中各个service之间是如何连接在一起的
- 快速了解当前整个系统的运行状态
- 定位故障，并评估对其他services的影响

NEW RELIC - TRANSACTION MAP

- 如何查找系统瓶颈?

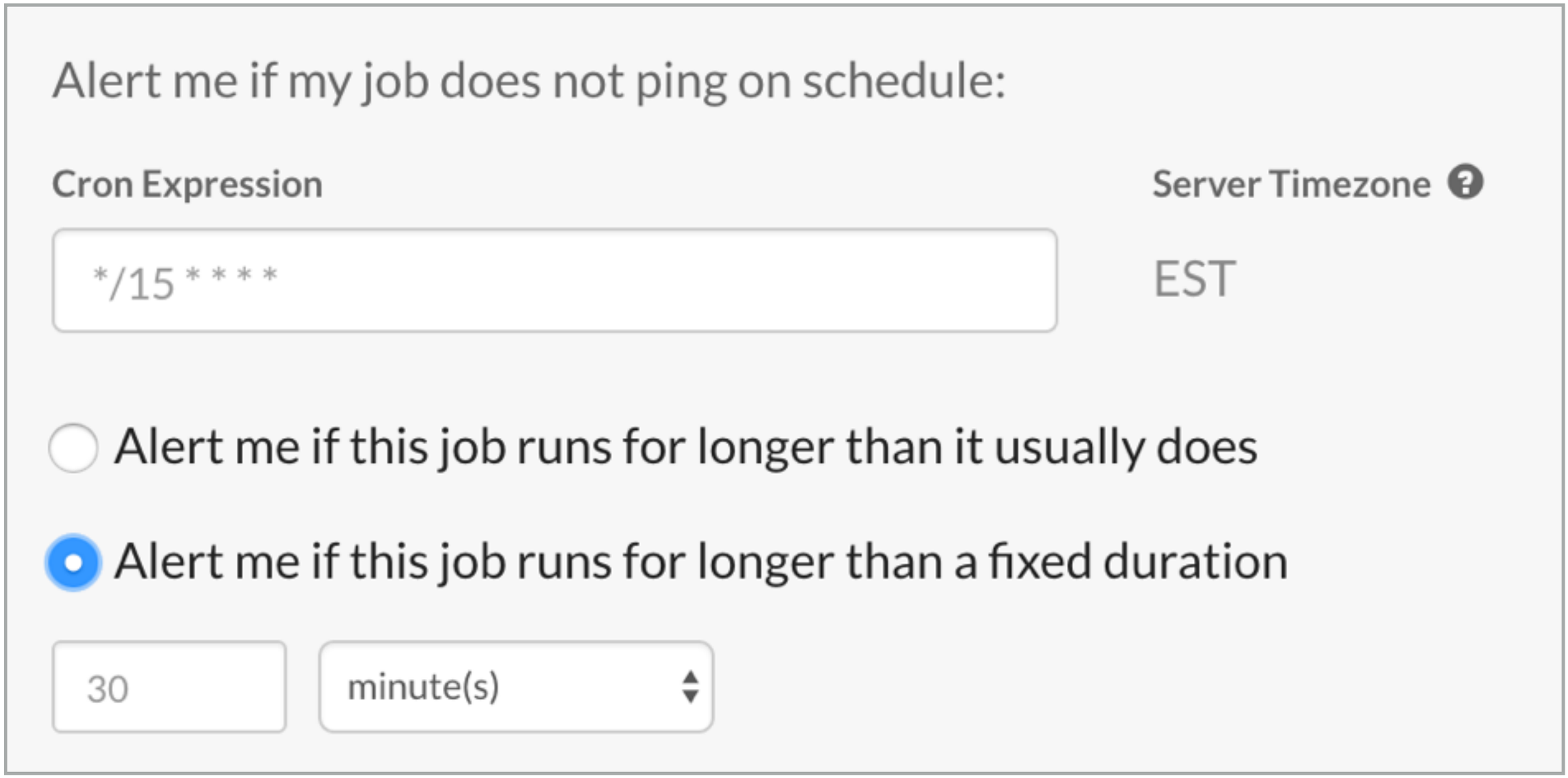


JOB 监控 - NEW RELIC

```
class SalesOrganization
  include ::NewRelic::Agent::Instrumentation::ControllerInstrumentation
  def find_new_leads
     ...
  end
  add_transaction_tracer :find_new_leads, :category => :task
end
```

- Tracing method
- Tracing blocks of code

CRON JOBS监控 - CRONITOR



定制的JOB监控工具 - OPSBOX

项目Job特性

- 不知道具体什么时间跑
- Job每天都会跑
- 某个定点前一定会跑完
- Job有输出，S3上存文件或是数据库插数据

Monitor：每天固定时间点检测Job的输出

- S3文件存不存在
- 数据库数据是否更新

LOG管理工具 - SPLUNK

splunk>

开源工具

监控：

Nagios®
The Industry Standard In IT Infrastructure Monitoring

Log管理：

Logstash
Collect, Enrich & Transport Data

Elasticsearch
Search & Analyze Data in Real Time

Kibana
Explore & Visualize Your Data

总结

构 建

测 试

部 署

监 控

THANK YOU

Questions?

ThoughtWorks®