GOPS2017
Shenzhen

GOPS

2017 全球运维大会

深圳站

指导单位：数据中心联盟 Data Center Alliance

主办单位：高效运维社区 GreatOPS Community　开放运维联盟 OOPSA Open OPS Alliance

GOPS2017
Shenzhen

基于DevOps的微服务生态系统与工程实践

- 微服务架构与DevOps
- 微服务架构的生态系统
- 微服务架构的工程实践

关于我

华为2012软件工程技术专家

西安尚度元科技CTO

ThoughtWorks首席咨询师

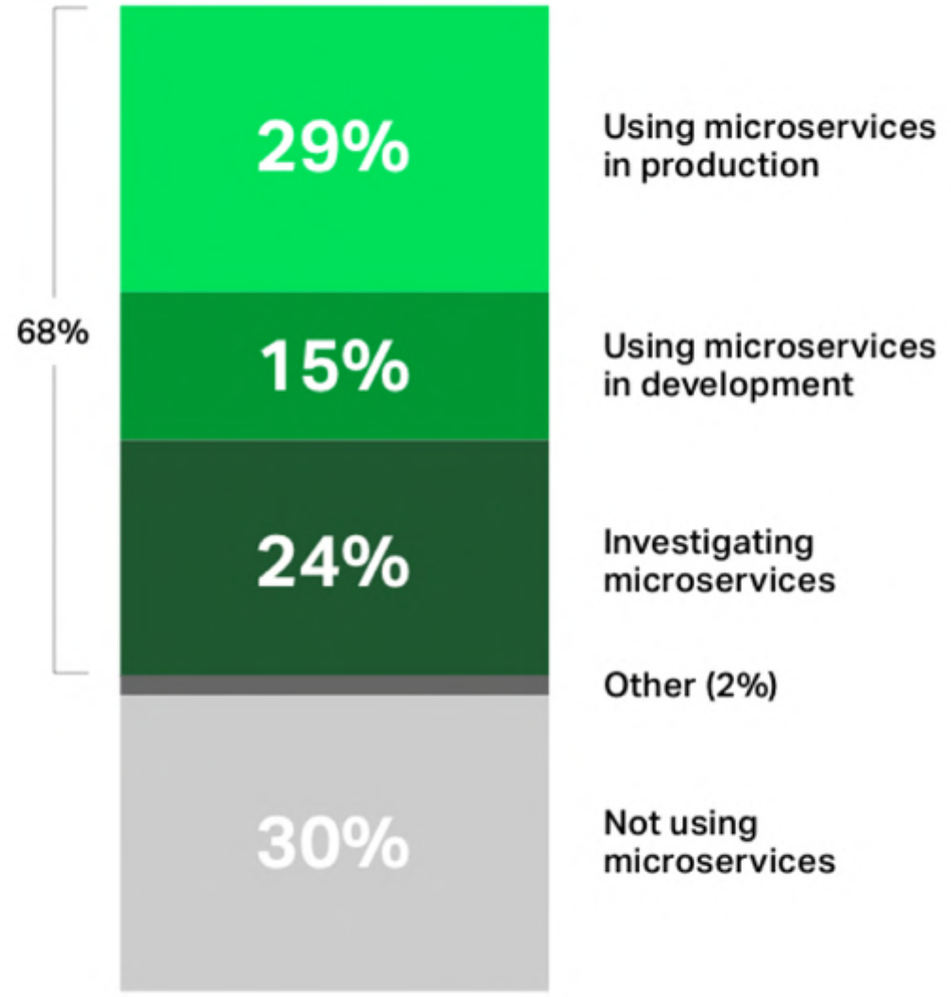
Sybase Tech Leader

- 《微服务架构与实践》作者
- 《DevOps Handbook》中文译者之一
- 国内较早倡导和实践微服务的先行者
- 精通敏捷/持续交付/DevOPS的核心理念、价值观和方法论
- 对于自动化测试、持续集成、持续交付有丰富的实战经验
- 西安DevOps Meetup 联合发起人

Are you using microservices?

Microservices are entering mainstream

68% of organizations are using or investigating microservices

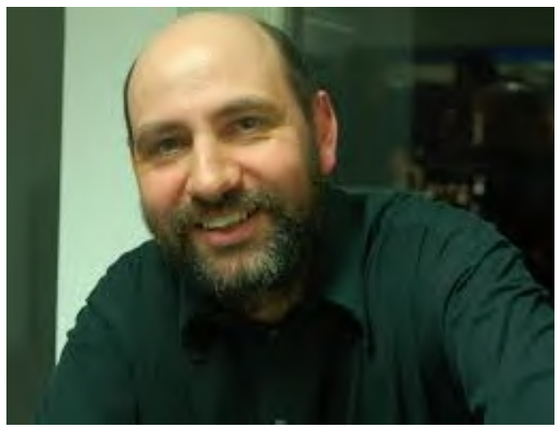


SURVEY QUESTION Which statement *best* defines how your organization is currently using microservices?

https://www.nginx.com/resources/library/app-dev-survey/

什么是微服务架构

微服务架构



Microservices - the new architectural style

Martin Fowler, Mar 2014

微服务架构是一种架构模式，它提倡将单一应用程序划分成**一组小的服务**，服务之间互相协调、互相配合，为用户提供最终价值。

每个服务运行在其**独立的进程中**，服务与服务间采用**轻量级的通信机制**互相协作（通常是基于HTTP协议的RESTful API）。

每个服务都围绕着具体业务进行构建，并且能够被**独立的部署**到生产环境、类生产环境等。

不同的声音.....

Micro (u)Services Architecture - small, short lived services rather than SOA.

2012/3

Fred George

不同的声音.....

Micro (u)Services Architecture - small, short lived services rather than SOA.

2012/3

Fred George

Loosely coupled service oriented architecture with bounded contexts.

2014/11

Adrian Cockcroft

不同的声音.....

Micro (u)Services Architecture - small, short lived services rather than SOA.

2012/3

Fred George

Loosely coupled service oriented architecture with bounded contexts.

2014/11

Adrian Cockcroft

Microservices are the first post DevOps revolution architecture.

2015/10

Neal Ford

.....

以缩短交付周期为核心
基于DevOps
构建的演进式架构

I am not perfect in my walk but I want to do the right thing.

Kirk Cameron

为什么以缩短交付周期为核心？

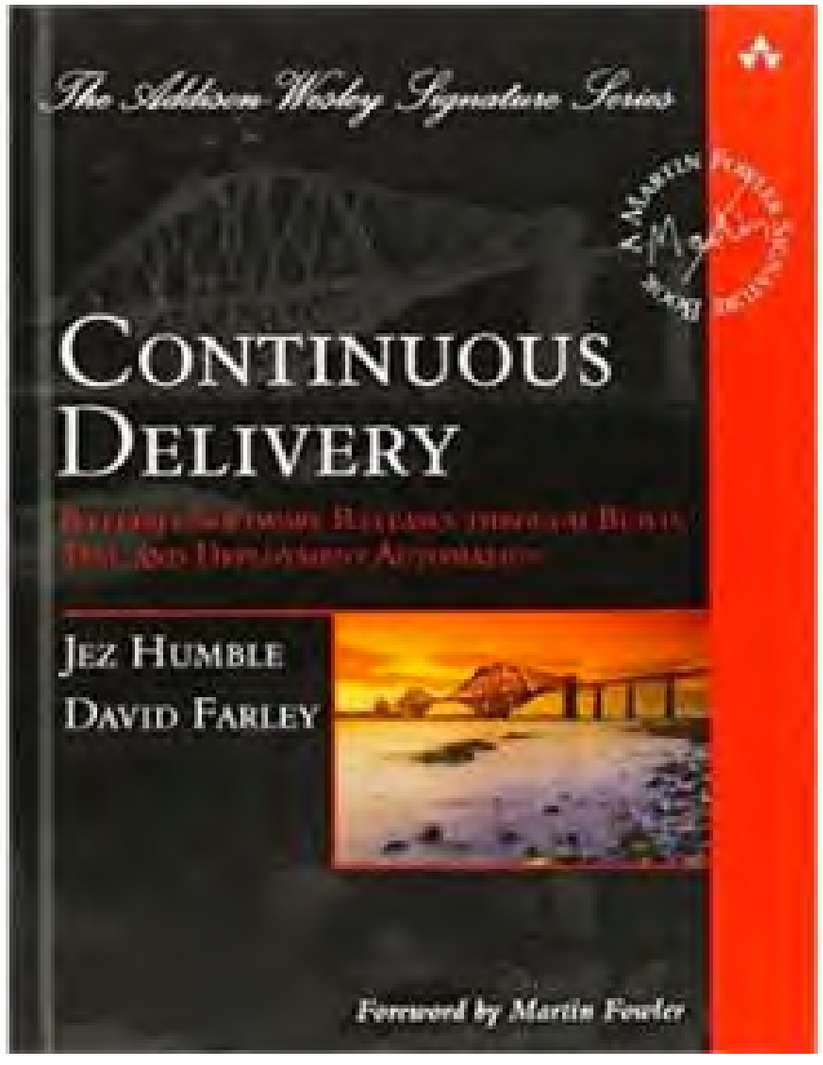
为什么以缩短交付周期为核心？

We can’t deliver new business value as fast as the business needs it. It is absolutely clear that old approaches are showing down the company’s growth and we are at risk of losing out competitive advantages.

我们交付特性的速度已经无法满足业务变化需要。旧的工作方式阻碍了组织的发展，而我们也因此正在丧失竞争力。

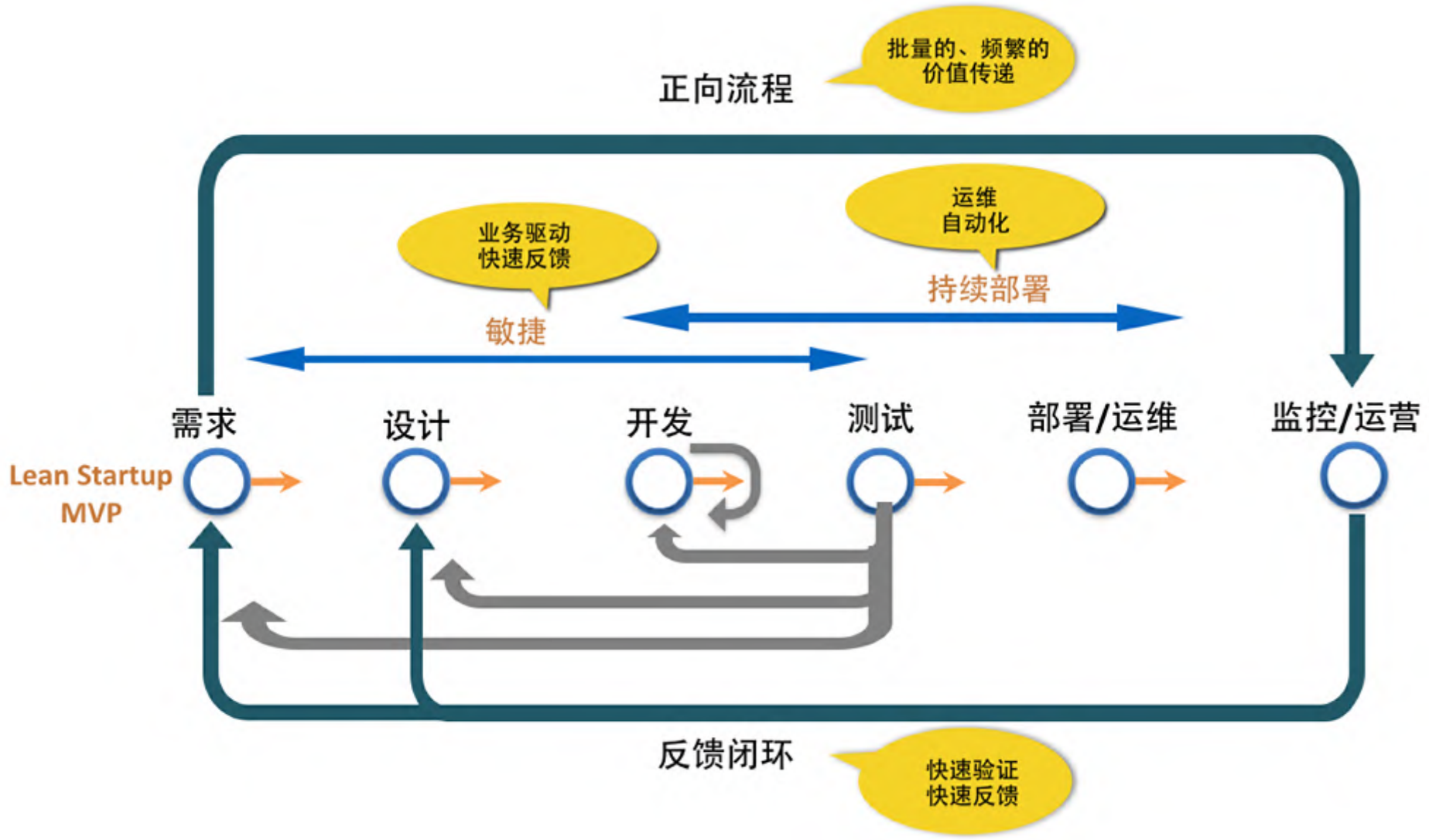
O’Reilly Software Architecture Conference 2017.4

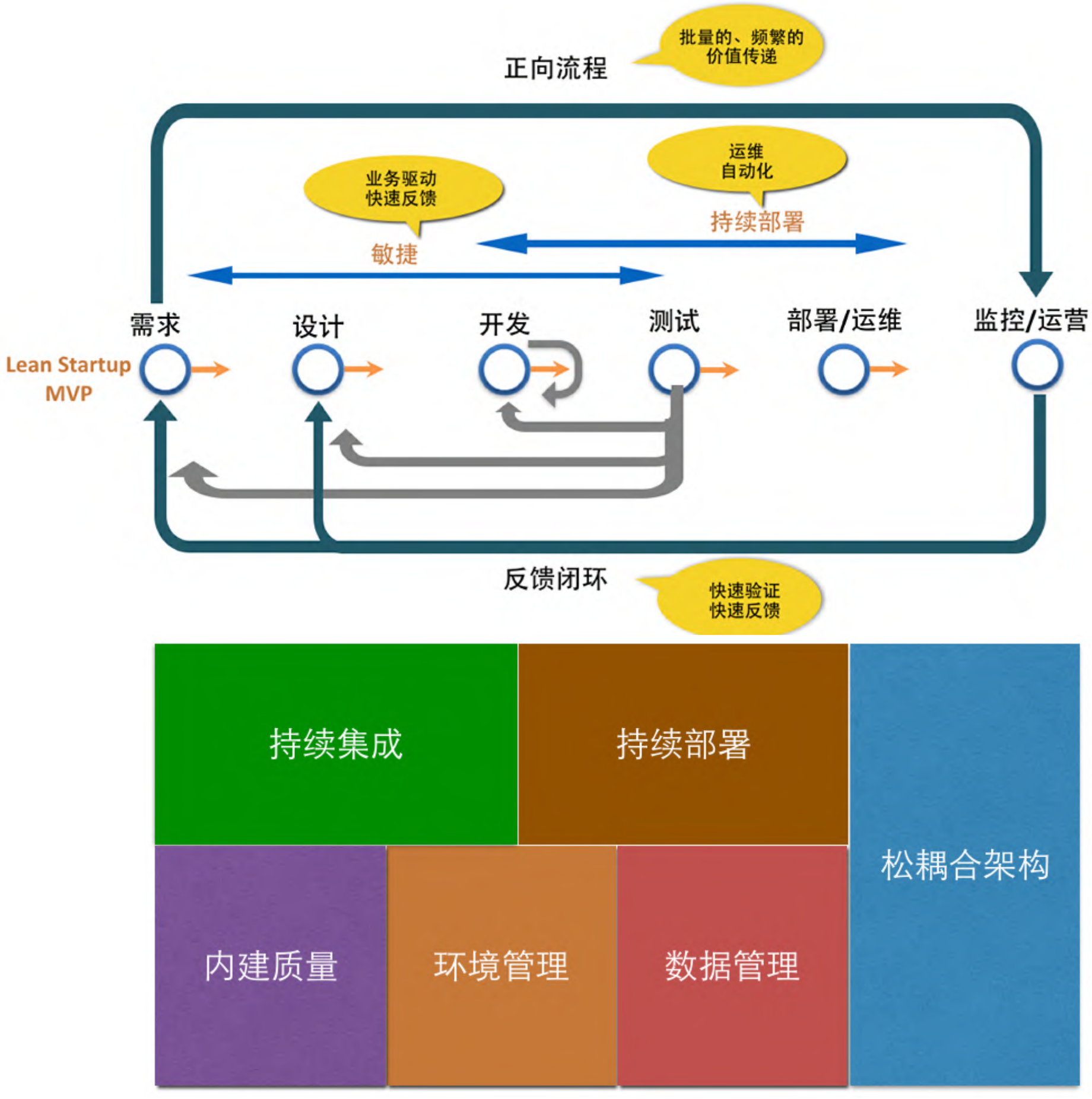
为什么以缩短交付周期为核心？



- *Faster time to market*
- *Lower risk release*
- *Higher quality*
- *Lower costs*
- *Better products*

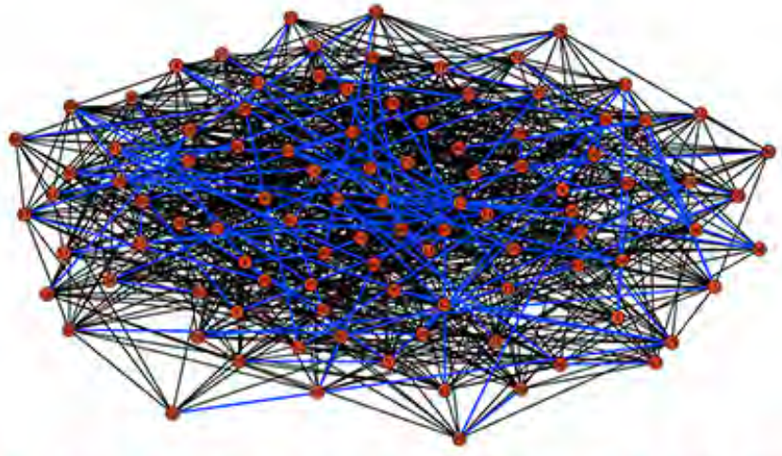
https://www.continuousdelivery.com/


正向流程
批量的、频繁的
价值传递
运维
自动化
业务驱动
快速反馈
持续部署
敏捷
需求
设计
开发
测试
部署/运维
监控/运营
Lean Startup
MVP
反馈闭环
快速验证
快速反馈


正向流程
批量的、频繁的
价值传递
运维
自动化
业务驱动
快速反馈
持续部署
敏捷
需求
设计
开发
测试
部署/运维
监控/运营
Lean Startup
MVP
反馈闭环
快速验证
快速反馈
持续集成
持续部署
松耦合架构
内建质量
环境管理
数据管理

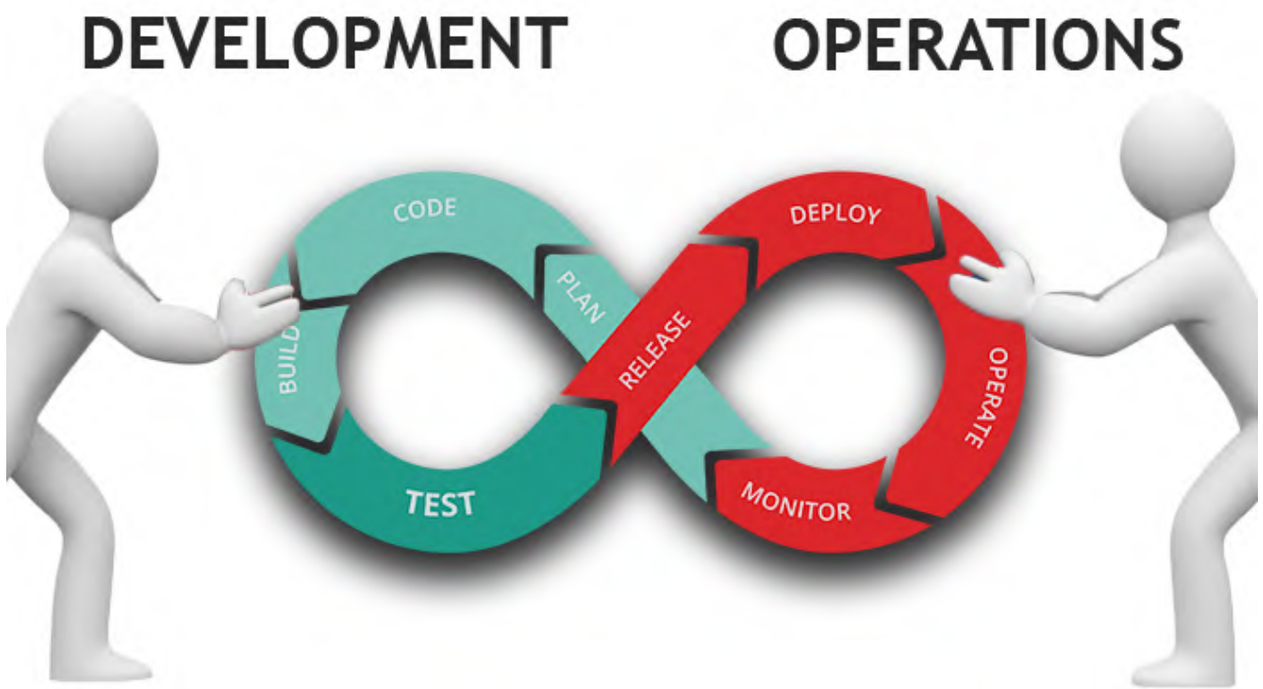
为什么基于DevOps？

为什么基于DevOps？



MICROSERVICES

为什么基于DevOps？



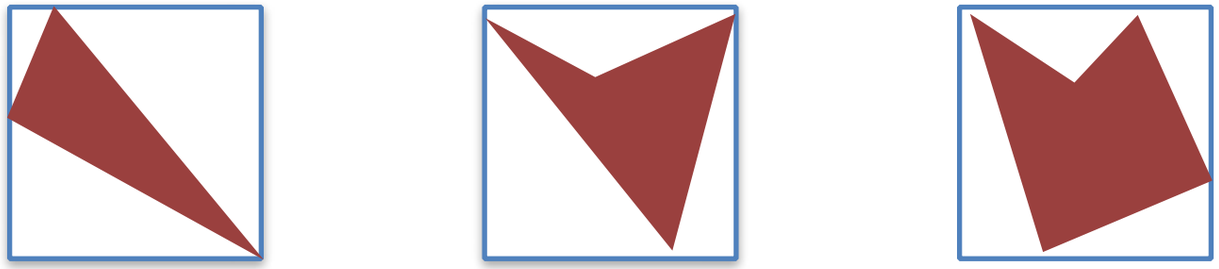
- **C**ommunication
- **A**utomation
- **M**easuring
- **S**haring

https://www.supinfo.com/articles/single/3652-what-is-devops

什么是演进式架构？

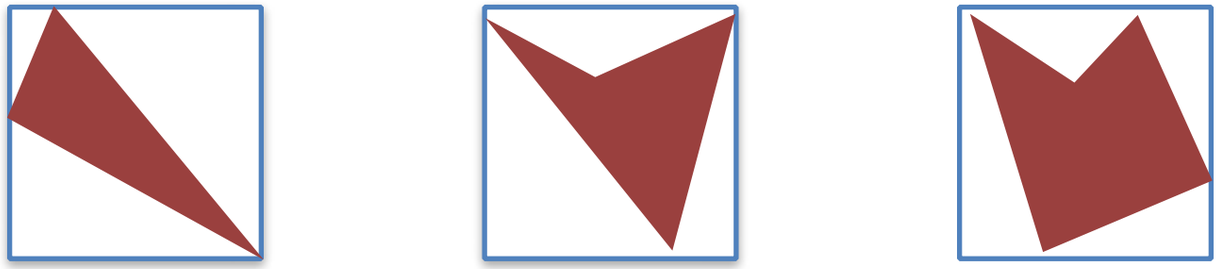
什么是演进式架构？

- 架构一旦确定，很难改变



什么是演进式架构？

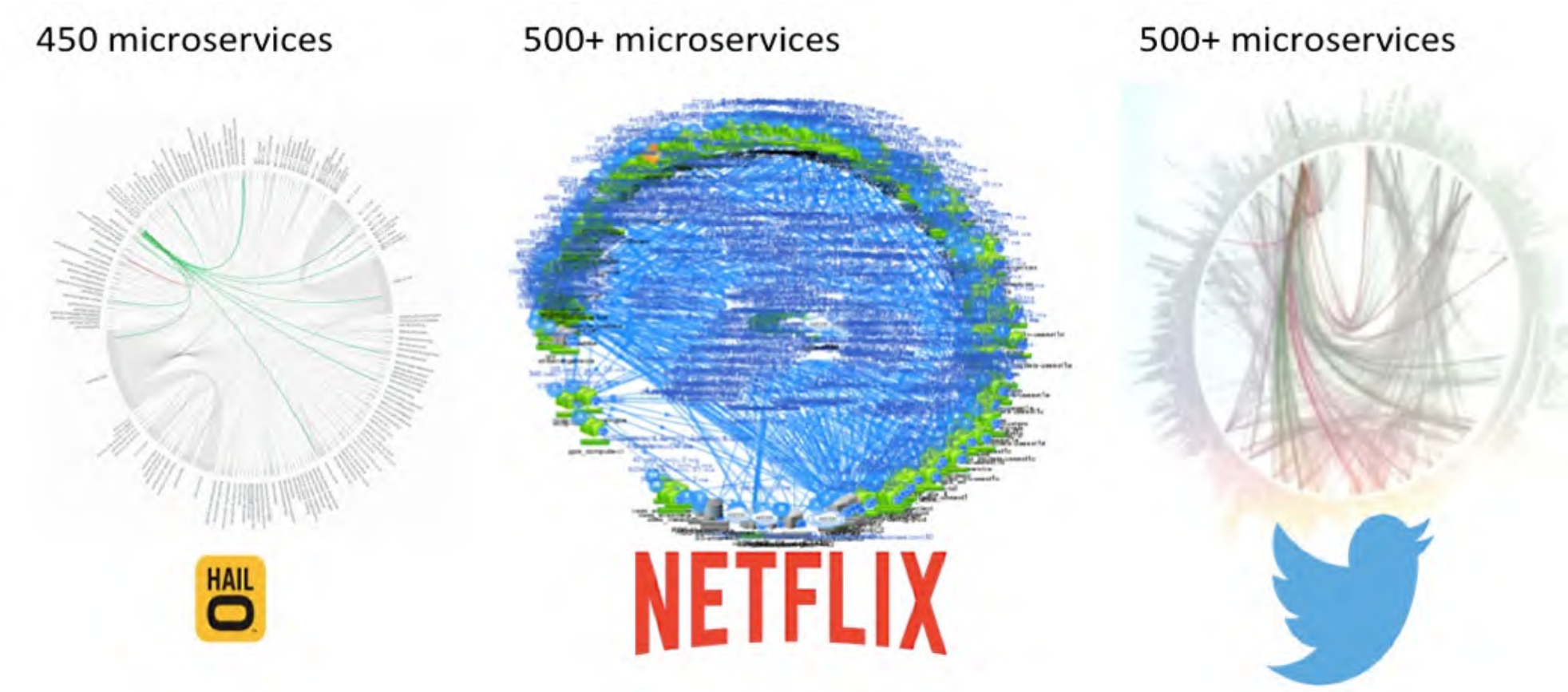
- 架构一旦确定，很难改变



- 支持增量式变更作为第一原则

- 动态平衡
- 运维意识是关键
- 延迟非重要决策点
- 痛苦的事情提前做

微服务架构是一种演进式架构



Source:
Netflix: http://www.slideshare.net/BruceWong3/the-case-for-chaos
Twitter: https://twitter.com/adrianco/status/441883572618948608
Hail-o: https://sudo.hailoapp.com/services/2015/03/09/journey-into-a-microservice-world-part-3/

以缩短交付周期为核心
基于DevOps
构建的演进式架构

I am not perfect in my walk but I want to do the right thing.

Kirk Cameron

- 微服务架构与DevOps
- 微服务架构的生态系统
- 微服务架构的工程实践

系统化的工程

- 服务拆分与解耦
- 分布式事务的一致
- 注册/发现/路由机制
- 安全与认证机制

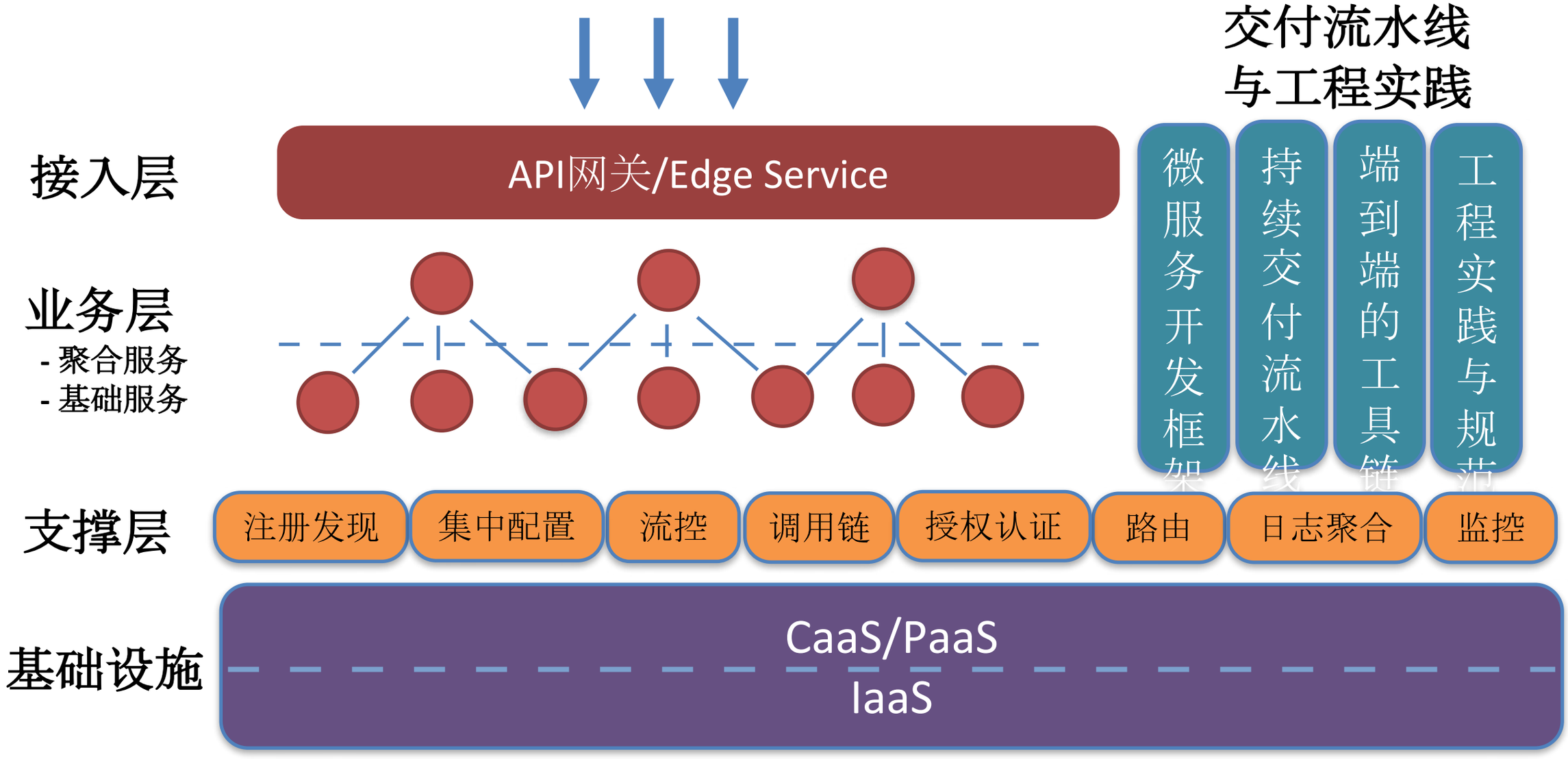
框架层出不穷

- API网关
- 服务开发框架
- 测试验证框架
- 配置管理框架

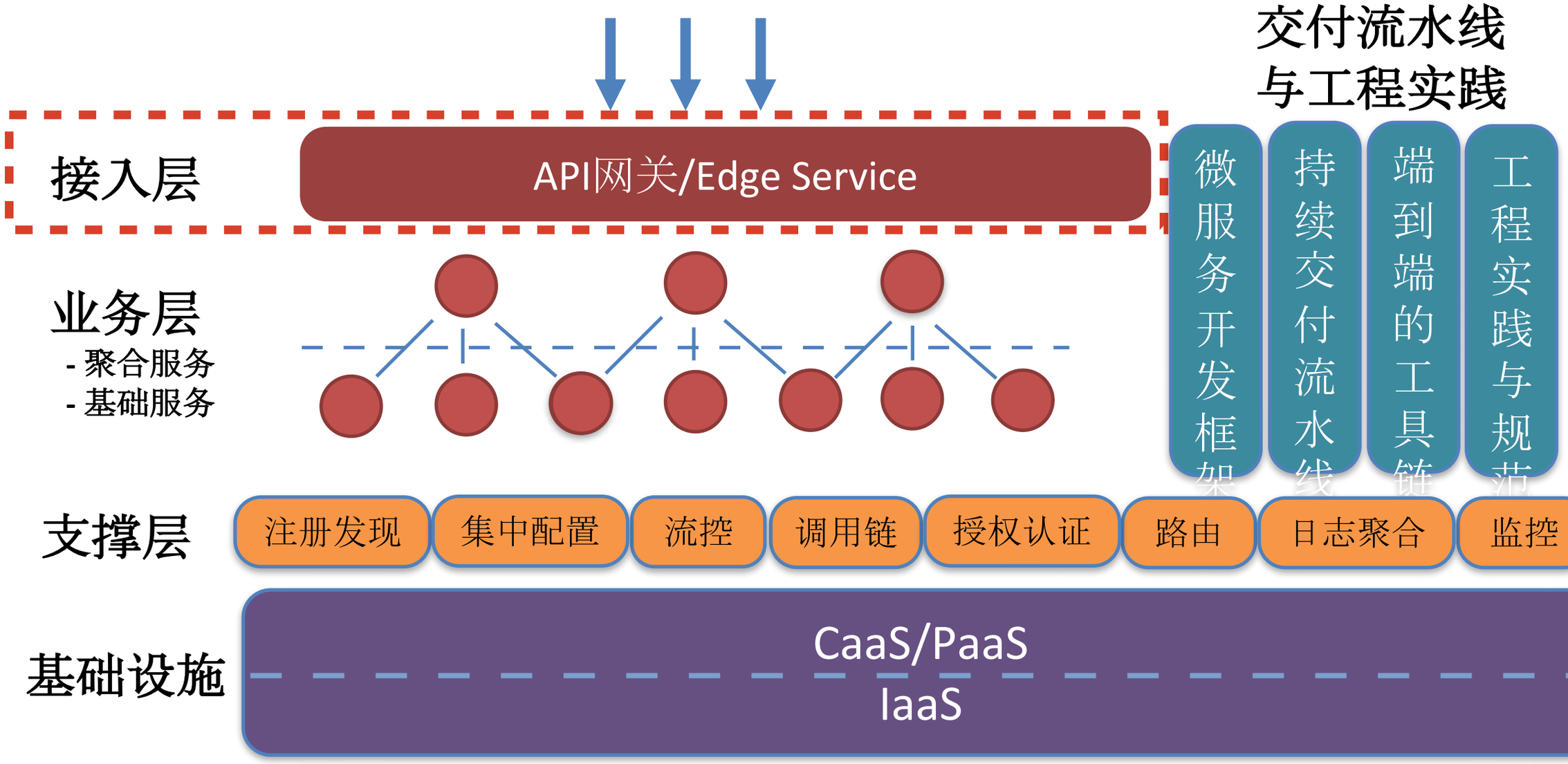
工具百花齐放

- 基础设施(云/虚拟机)
- 持续集成/持续部署
- 交付流水线
- 端到端工具链

微服务生态系统


交付流水线
与工程实践
微服务开发框架
持续交付流水线
端到端的工具链
工程实践与规范
接入层
API网关/Edge Service
业务层
- 聚合服务
- 基础服务
支撑层
注册发现
集中配置
流控
调用链
授权认证
路由
日志聚合
监控
基础设施
CaaS/PaaS
IaaS

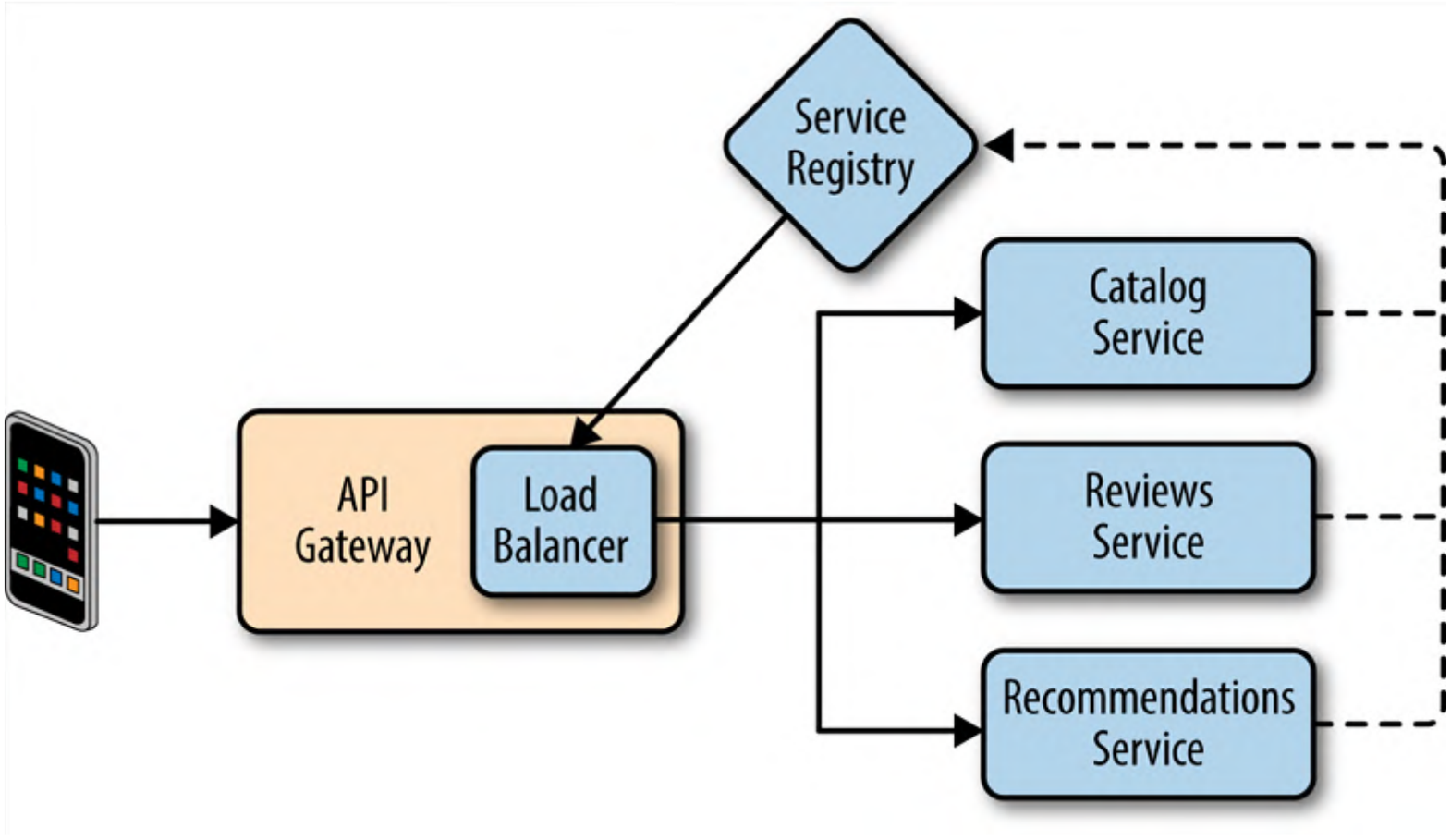
微服务生态系统


交付流水线
与工程实践
接入层
API网关/Edge Service
业务层
- 聚合服务
- 基础服务
微服务开发框架
持续交付流水线
端到端的工具链
工程实践与规范
支撑层
注册发现
集中配置
流控
调用链
授权认证
路由
日志聚合
监控
基础设施
CaaS/PaaS
IaaS

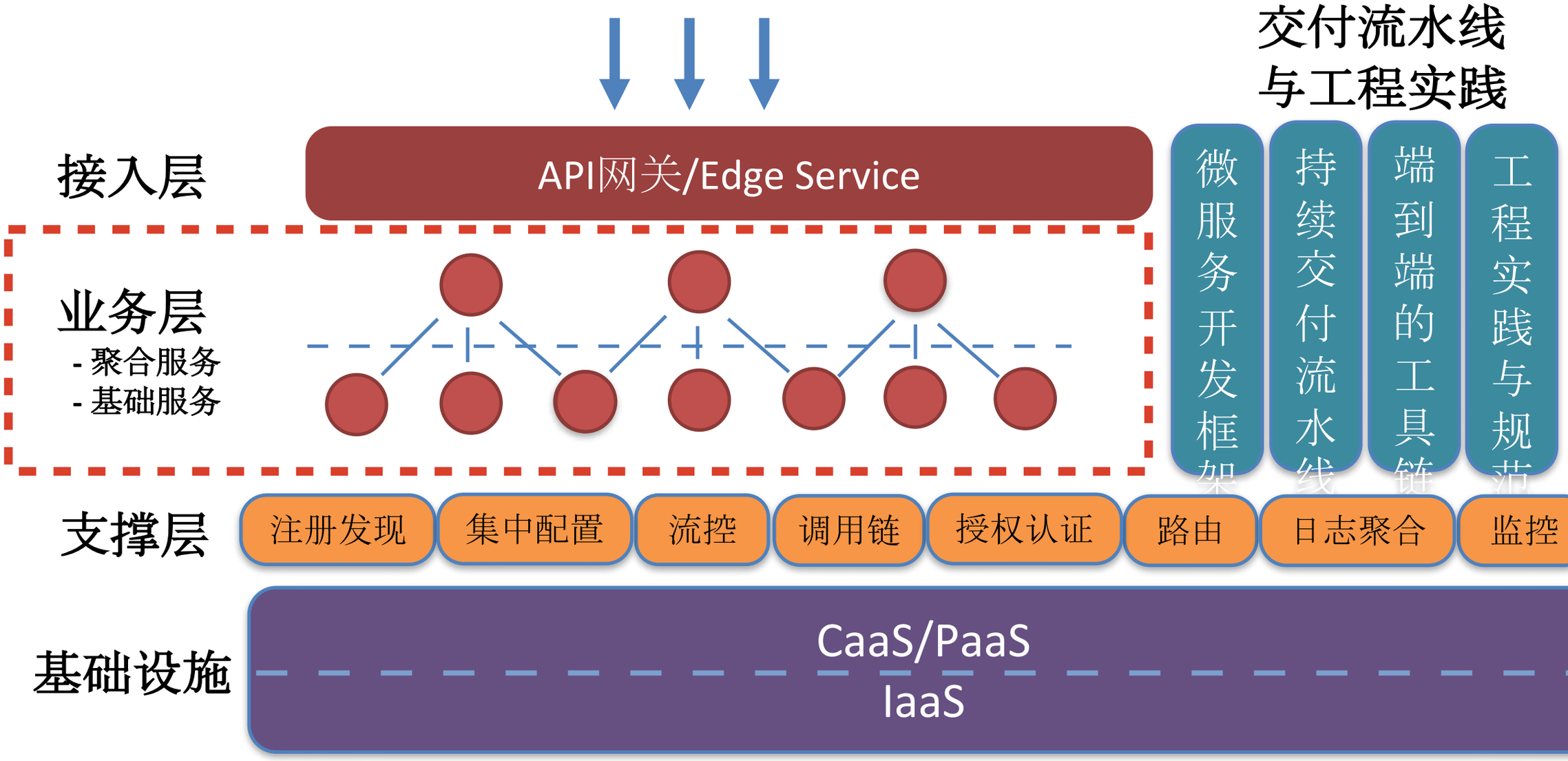
API网关

■提供单一接口，解耦内部变化　■提高通信效率

■请求转发　■流量限制　■调用统计　■安全认证

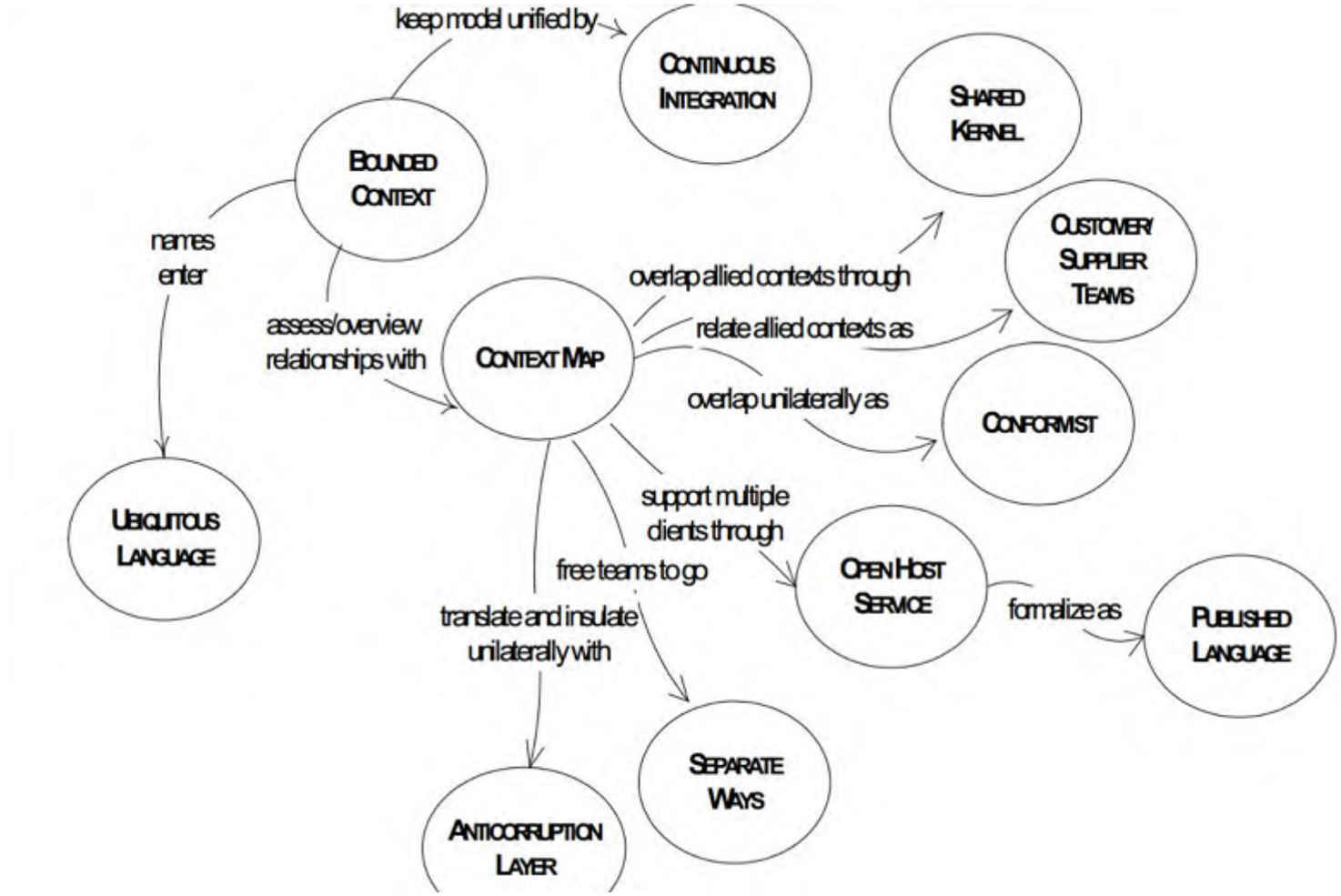


微服务生态系统


交付流水线
与工程实践
接入层
API网关/Edge Service
业务层
- 聚合服务
- 基础服务
微服务开发框架
持续交付流水线
端到端的工具链
工程实践与规范
支撑层
注册发现
集中配置
流控
调用链
授权认证
路由
日志聚合
监控
基础设施
CaaS/PaaS
IaaS

服务设计与拆分

- 面向对象分析
- 可重用的逻辑
- 资源密集型的业务
- 领域驱动设计



基础服务实现

Node.js

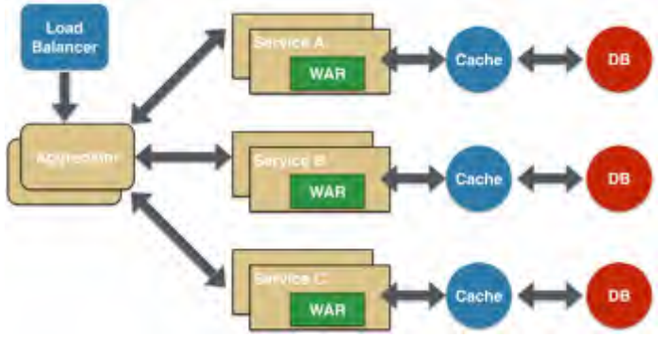
- Actionhero - Multi-transport Node.js API server with integrated cluster capabilities and delayed tasks.
- Baucis - To build and maintain scalable HATEOAS/Level 3 REST APIs.
- Express - Fast, unopinionated, minimalist web framework for Node.js
- Graft - Full-stack javascript through microservices.
- Hapi - A rich framework for building applications and services.
- Hudson Taylor - Set of libraries for building automatically documented, well validated services.
- Koa - Next generation web fr
- Loopback - Node.js fra
- Micro - Asynchronous H
- Micro-Whalla - A simple
- Restify - Node.js modul
- Seneca - A microservic
- Serverless - Build and n
  known as JAWS).

Java

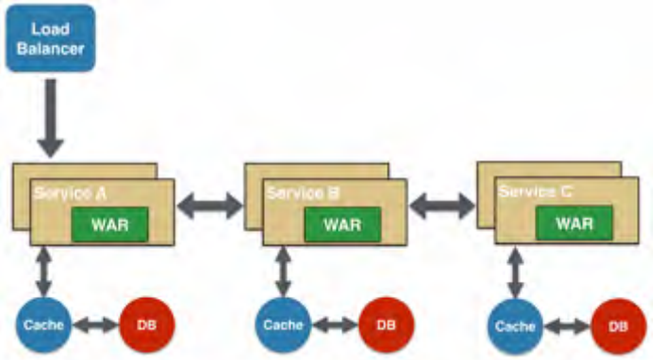
- Airlift - Framework for building REST services in Java.
- Disruptor - High-performance inter-thread messaging library.
- Dropwizard - Java framework for developing ops-friendly, high-performance, RESTful web services.
- HTTP Remoting - Libraries for defining and creating RESTish/RPC servers and clients based on Feign or Retrofit as a client and Dropwizard/Jersey with JAX-RS service definitions as a server.
- Jersey - RESTful services in Java. JAX-RS reference implementation.
- MSF4J - High throughput & low memory footprint Java microservices framework.
- QBit - Reactive programming library for building microservices.
- Ratpack - Set of Java libraries that facilitate fast, efficient, evolvable and well tested HTTP applications. specific support for the Groovy language is provided.
- Restlet - Helps Java developers build web APIs that follow the REST architecture style.
- Spark - A micro-framework for creating web applications in Java 8 with minimal effort.
- Spring Boot - Makes it easy to create stand-alone, production-grade Spring based applications.

https://github.com/mfornos/awesome-microservices

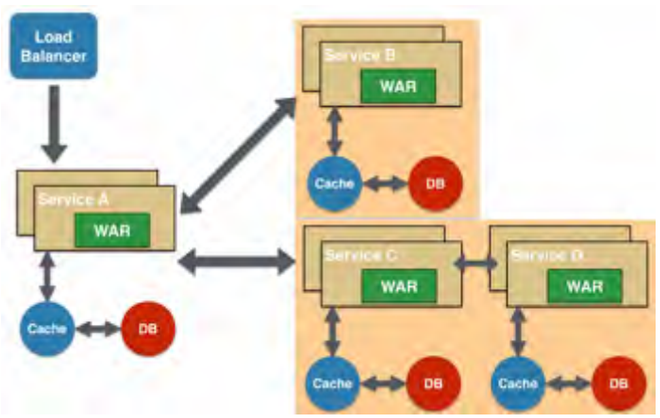
聚合服务实现



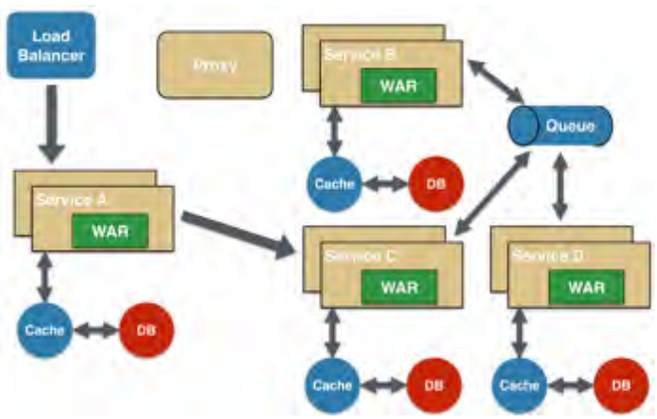
Proxy



Chained



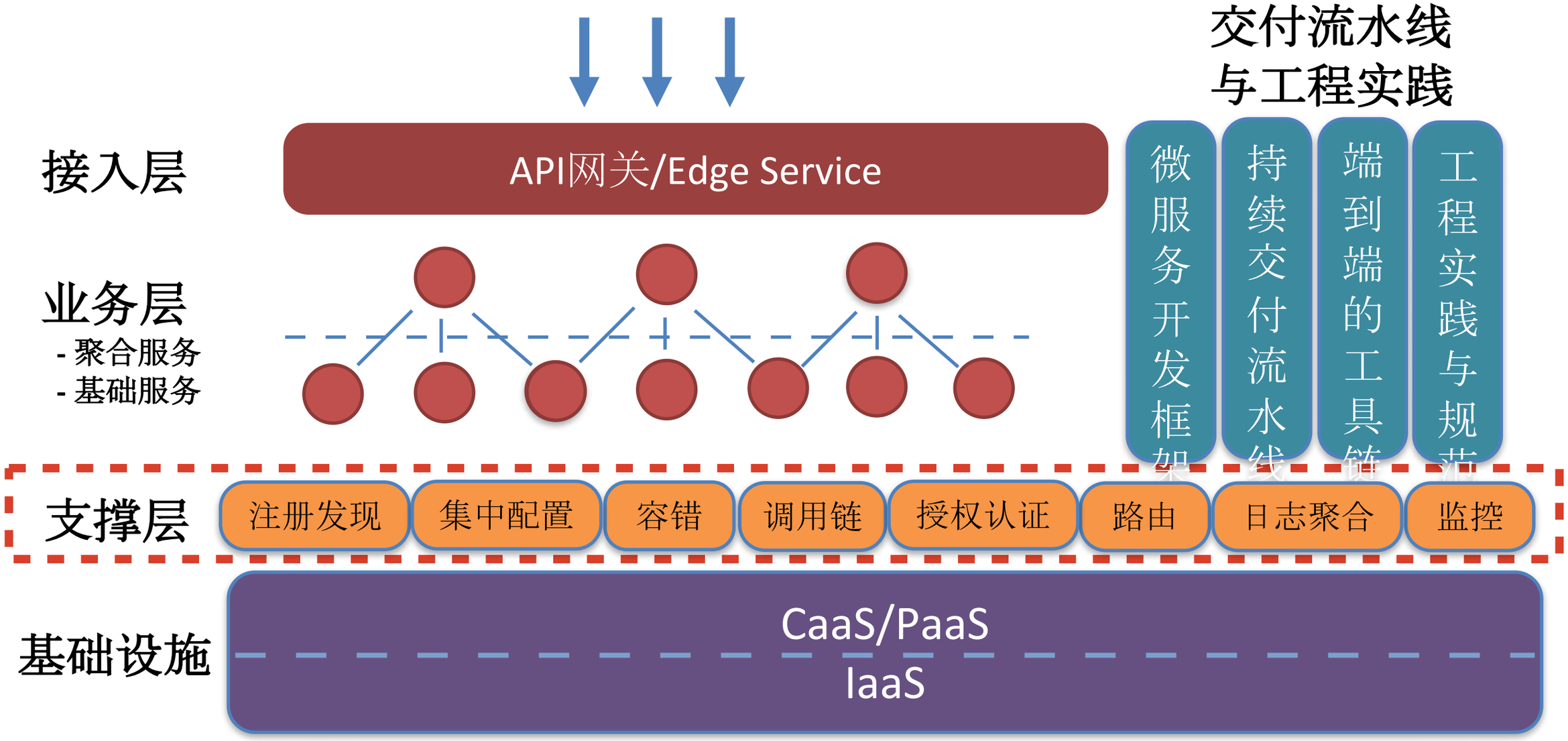
Shared Cache



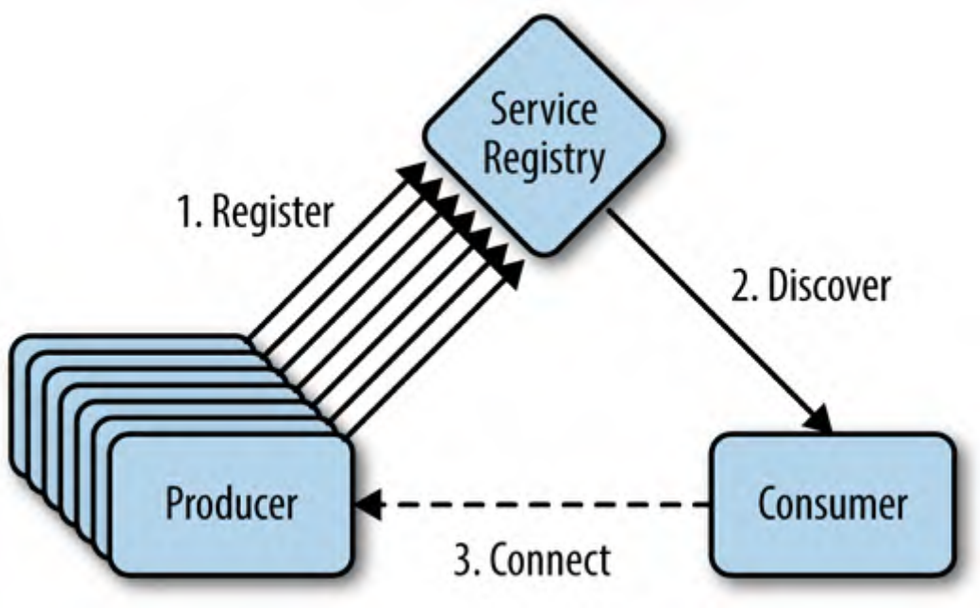
Asynchronous Messaging

https://dzone.com/articles/microservice-design-patterns

微服务生态系统

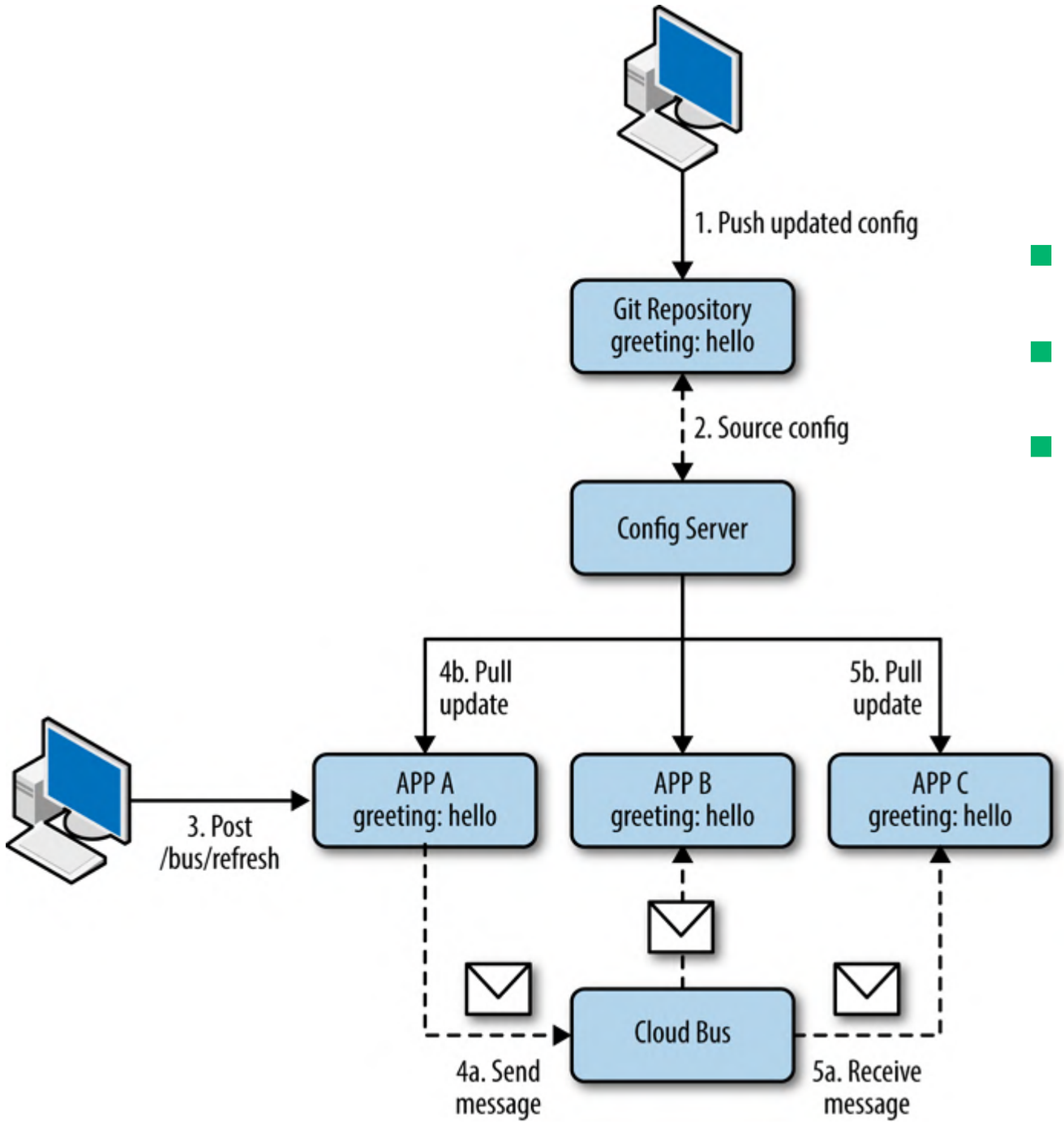

交付流水线
与工程实践
接入层
API网关/Edge Service
业务层
- 聚合服务
- 基础服务
微服务开发框架
持续交付流水线
端到端的工具链
工程实践与规范
支撑层
注册发现
集中配置
容错
调用链
授权认证
路由
日志聚合
监控
基础设施
CaaS/PaaS
IaaS

注册发现



- 服务重启/升级后的IP地址变化
- 水平伸缩后服务的实例数量变化
- 同一个结点运行多个服务(端口不同)

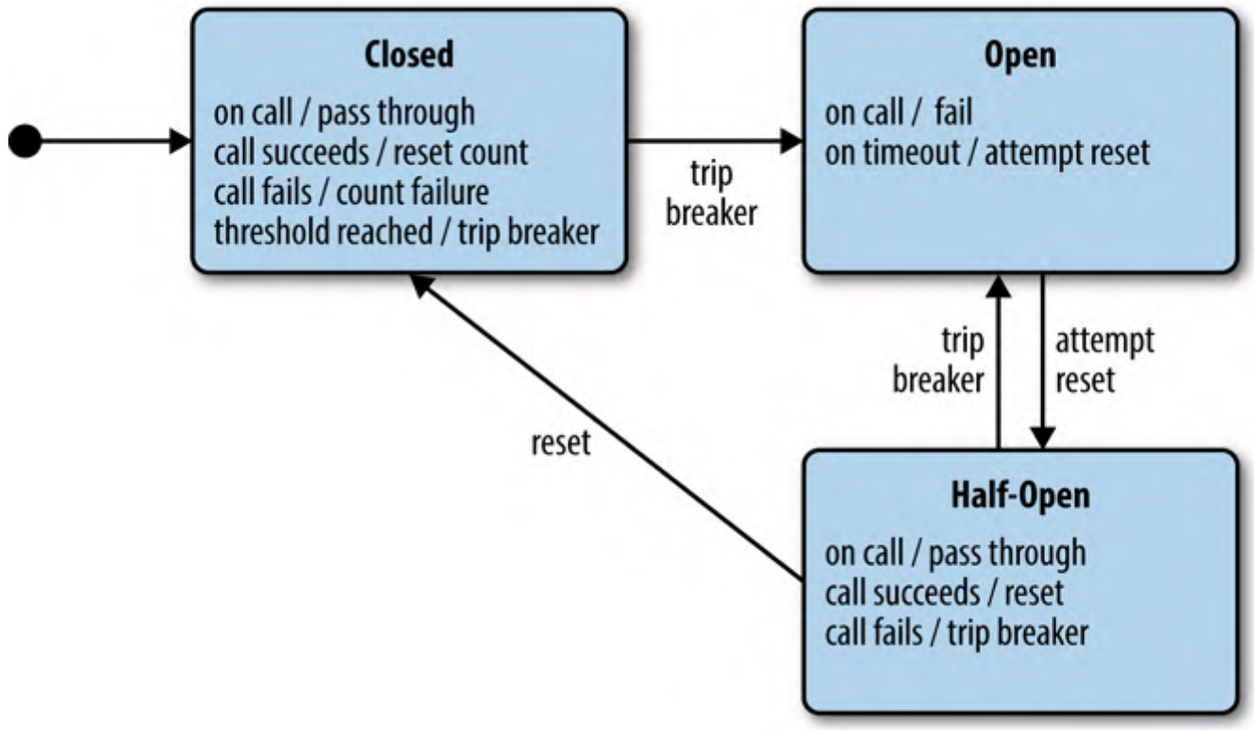
集中配置



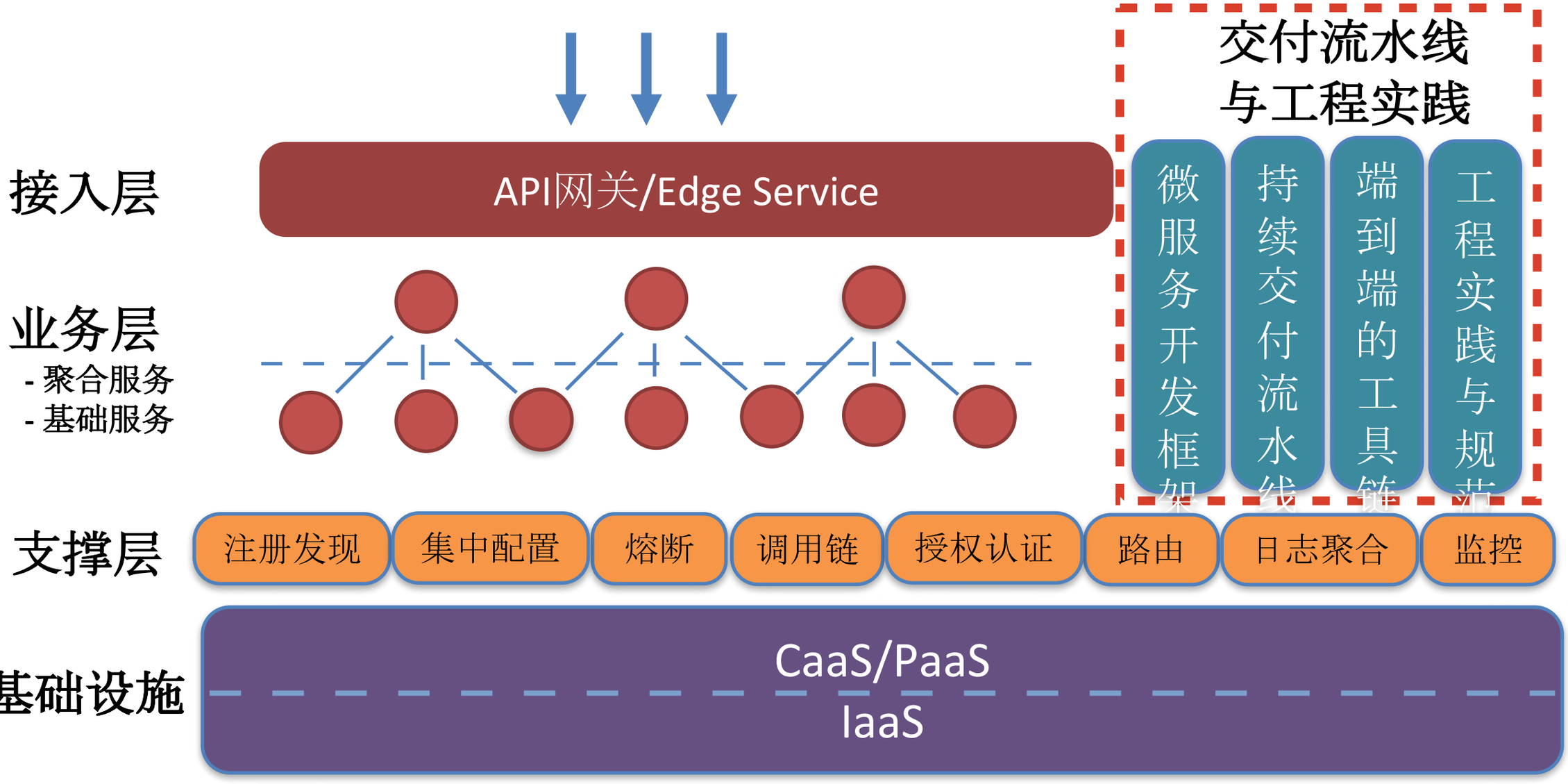
- 如何动态更新服务的配置信息
- 如何同步多实例间的配置变更
- 配置的追溯/回滚以及版本管理

容错(Resilient)

- 限流 - 请求超过处理能力，采用某种策略丢弃
- 降级 - 关闭非核心业务，保证核心业务可用
- 熔断 - 避免某个服务不可用导致的故障蔓延



微服务生态系统


交付流水线
与工程实践
微服务开发框架
持续交付流水线
端到端的工具链
工程实践与规范
接入层
API网关/Edge Service
业务层
- 聚合服务
- 基础服务
支撑层
注册发现
集中配置
熔断
调用链
授权认证
路由
日志聚合
监控
基础设施
CaaS/PaaS
IaaS

微服务开发框架

- DropWizard
- Spring Boot
- Lagom(Scala)
- Seneca(NodeJS)

微服务开发框架

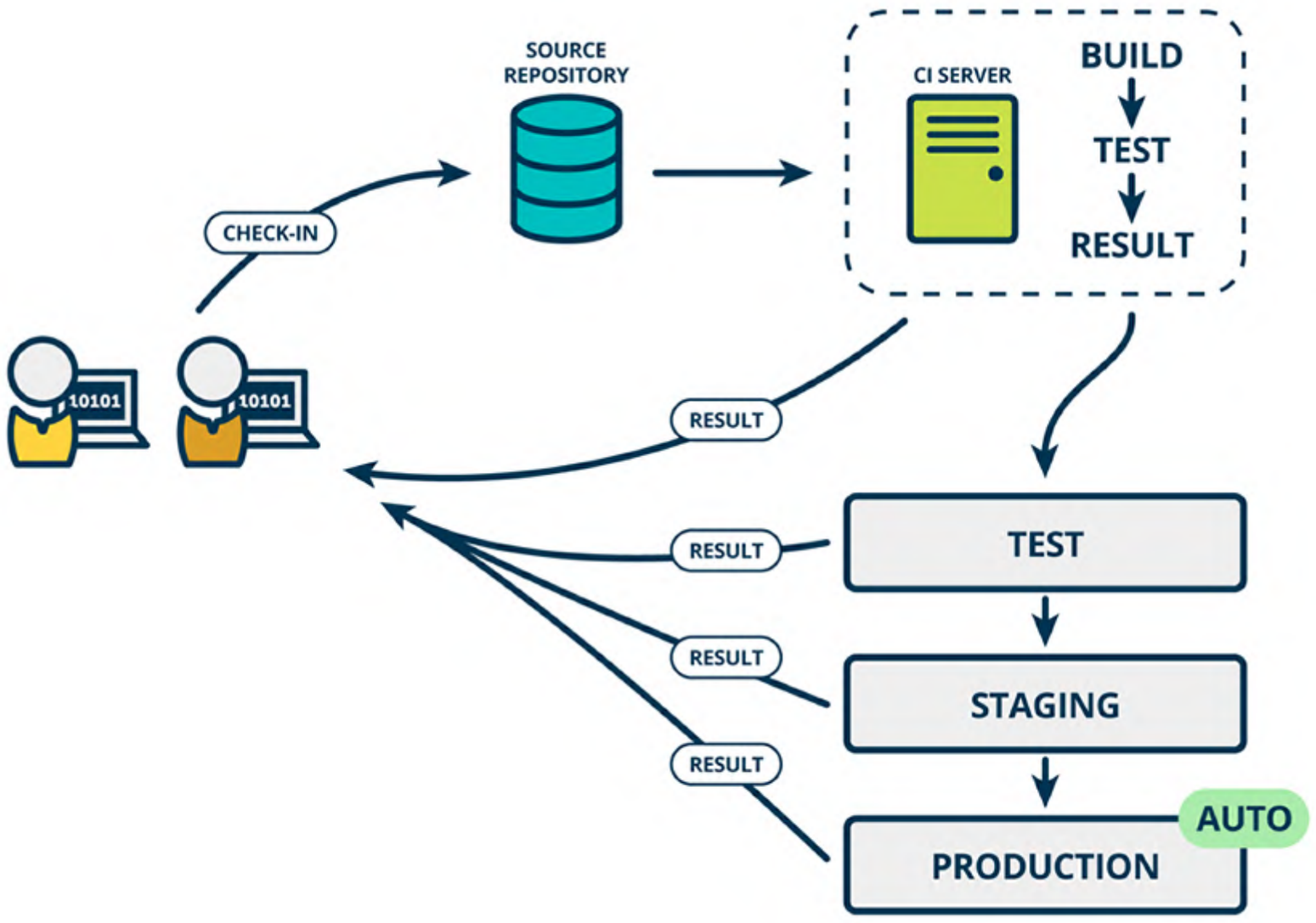
1. 业务层只关心编程模型和契约定义（微服务框架的编程接口、业务服务的接口定义）
2. 运行模型、通信模型对业务透明
3. 使用统一的服务契约（OpenAPI方式描述、业务层可通过自动化工具在开发态自动转换成IDL、POJO代码）
4. 序列化（编解码，PB、JSON等）可扩展，对运行模型、编程模型透明
5. 传输协议（REST over http1.0/2.0、grpc over http2.0，private over TCP…）可扩展，对运行模型、编程模型透明

华为CSE微服务开发框架

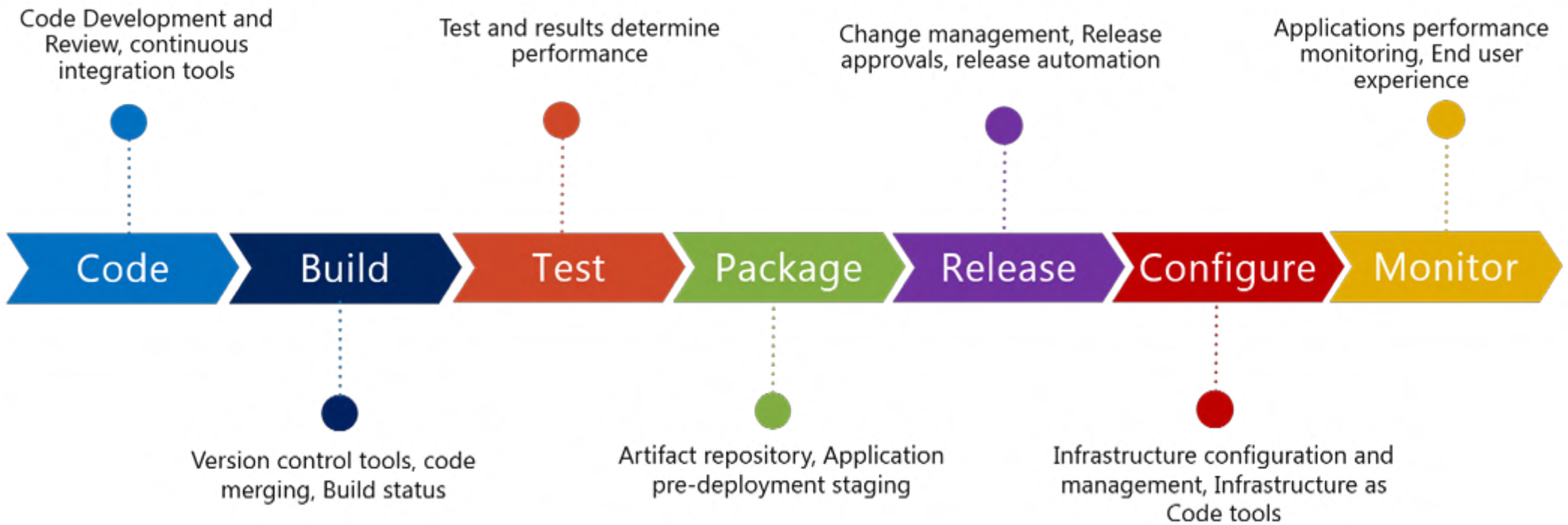
持续交付流水线

- 基于SaaS
  - TravisCI
  - CircleCI
- 产品
  - Jenkins
  - Bamboo
  - GoCD

.....

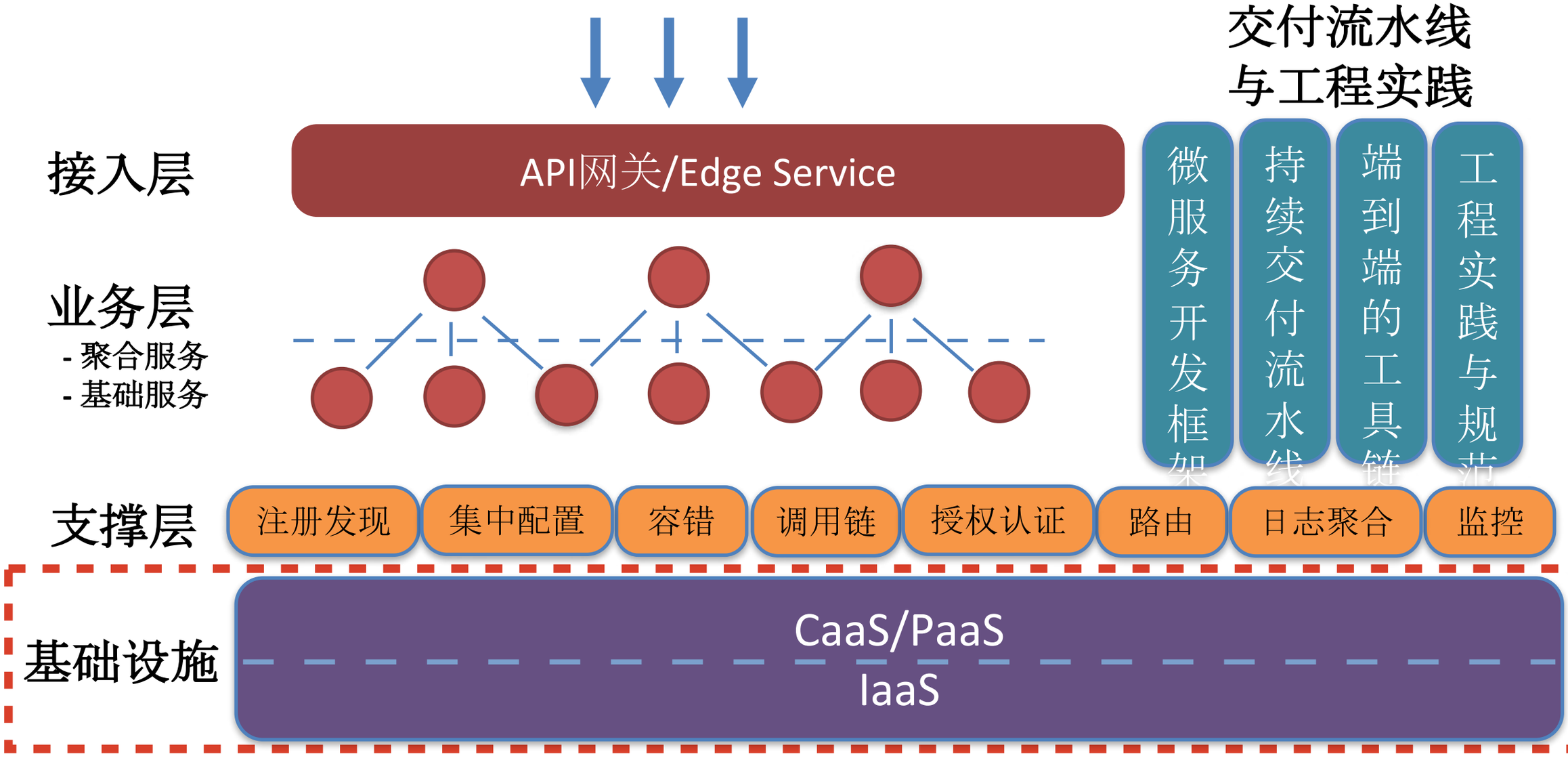


端到端工具链

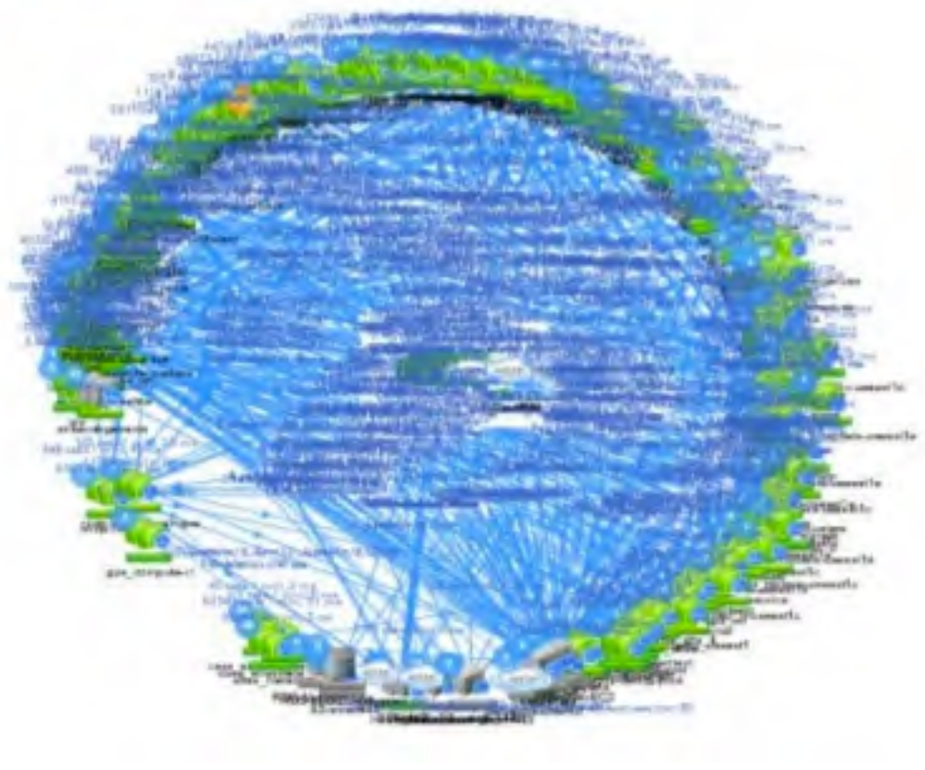

Code Development and Review, continuous integration tools
Test and results determine performance
Change management, Release approvals, release automation
Applications performance monitoring, End user experience
Code
Build
Test
Package
Release
Configure
Monitor
Version control tools, code merging, Build status
Artifact repository, Application pre-deployment staging
Infrastructure configuration and management, Infrastructure as Code tools


BI MONITORING
Ganglia
cacti
Nagios
New Relic
COLLABORATION
HipChat
JIRA Software
PivotalTracker
slack
servicenow
FLOWDOCK
Trello
CONTINUOUS INTEGRATION
circleci
CloudBees
Shippable
TinderBox
BRAINS
shippable
CODESHIP
cruise
Jenkins
TeamCity
Bamboo
CONTINUUM
BUILD
gradle
BuildMaster
Visual Build
Maven
pmEase
CLOUD IAAS
PROFITBRICKS
amazon
rackspace
Microsoft Azure
Google
CenturyLink
SOFTLAYER
vmware
DATABASE
Cassandra
Couchbase
MarkLogic
mongoDB
MySQL
APACHE HBASE
redis
DEPLOYMENT
Capistrano
RUNDECK
Octopus Deploy
MidVision
SmartFrog
Electric Cloud
DEPLOY
urbancode
MICROSERVICES
Elasticbox
ROCKETHELEANS
TRITON
Kubernetes
docker
RANCHER
nirmata
MESOS
RELEASE MANAGEMENT
ca
urbancode
Excel
RELEASE
bmc
CONFIG MANAGEMENT
ANSIBLE
66
SALTSTACK
puppet labs
CHEF
RUNDECK
CONFIG PROVISIONING
bmc
CFEngine
RUNDECK
puppet labs
CHEF
66
SALTSTACK
BCFG2
Rudder
ANSIBLE
VAGRANT
LOGGING
elastic
sumologic
graylog
loggly
splunk>
SOURCE CODE MANAGEMENT
git
SUBVERSION
GitHub
REPO MANAGEMENT
nuget
archiva
SECURITY
tripwire
NETWORKING
weave
VERSA
TESTING
Appium
FitNesse
gradle
Maven
cucumber
Python
JUnit
TestView
Selenium
TestComplete
PROVISIONING
puppet labs
VAGRANT
Created by ProfitBricks — 2015
ProfitBricks logo are trademarks or registered trademarks of ProfitBricks, Inc. All other registered trademarks or trademarks are property of their respective owners.

微服务生态系统


交付流水线
与工程实践
接入层
API网关/Edge Service
业务层
- 聚合服务
- 基础服务
微服务开发框架
持续交付流水线
端到端的工具链
工程实践与规范
支撑层
注册发现
集中配置
容错
调用链
授权认证
路由
日志聚合
监控
基础设施
CaaS/PaaS
IaaS

- 微服务架构与DevOps
- 微服务架构生态系统
- 微服务架构的工程实践

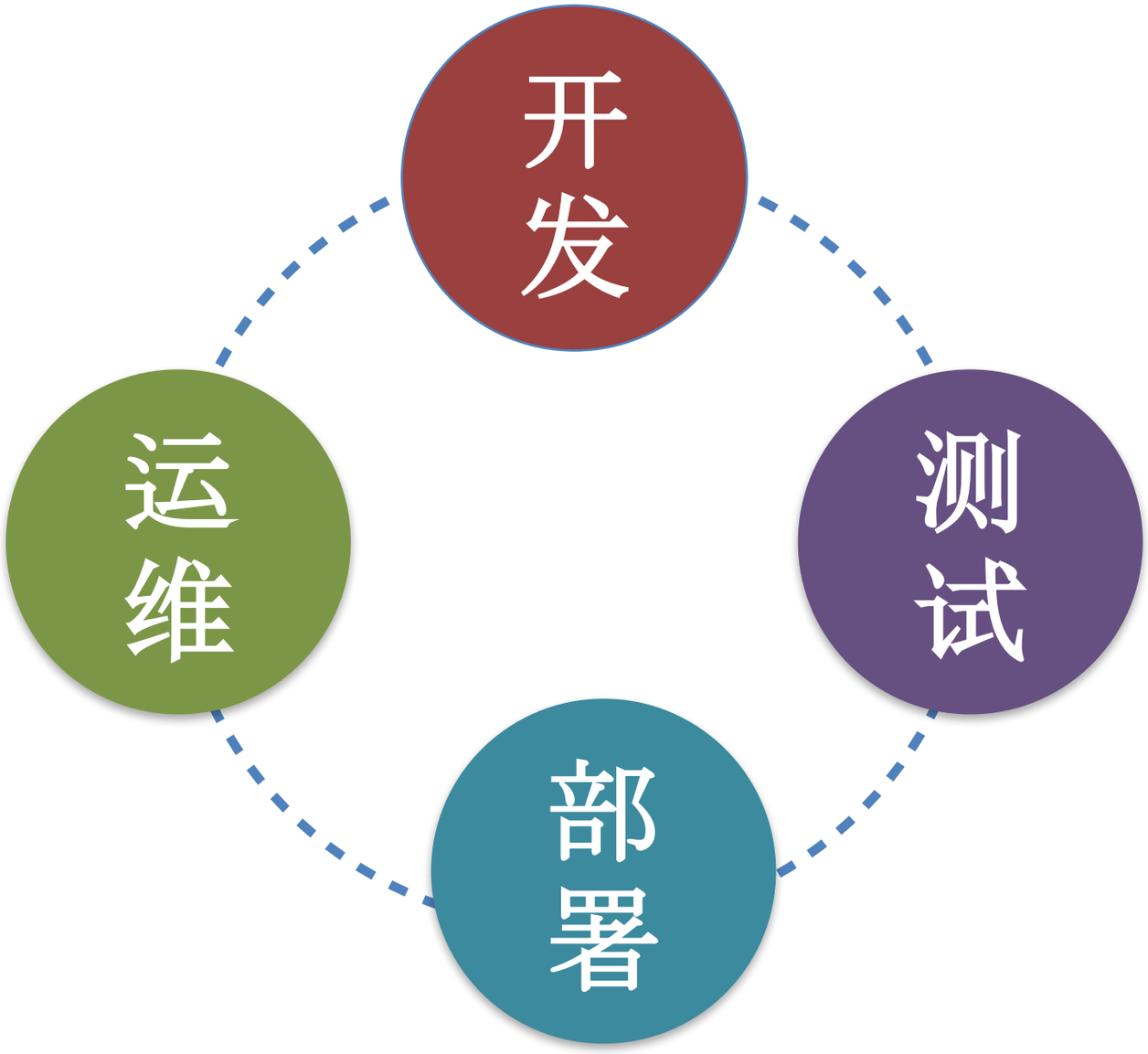


Netflix / 2009-2016 / 600+ microservices/AWS

People try to copy Netflix, but they can only copy what they see. They copy the results, not the process.

Andrian Cockroft, former Chief Cloud Architect, Netflix

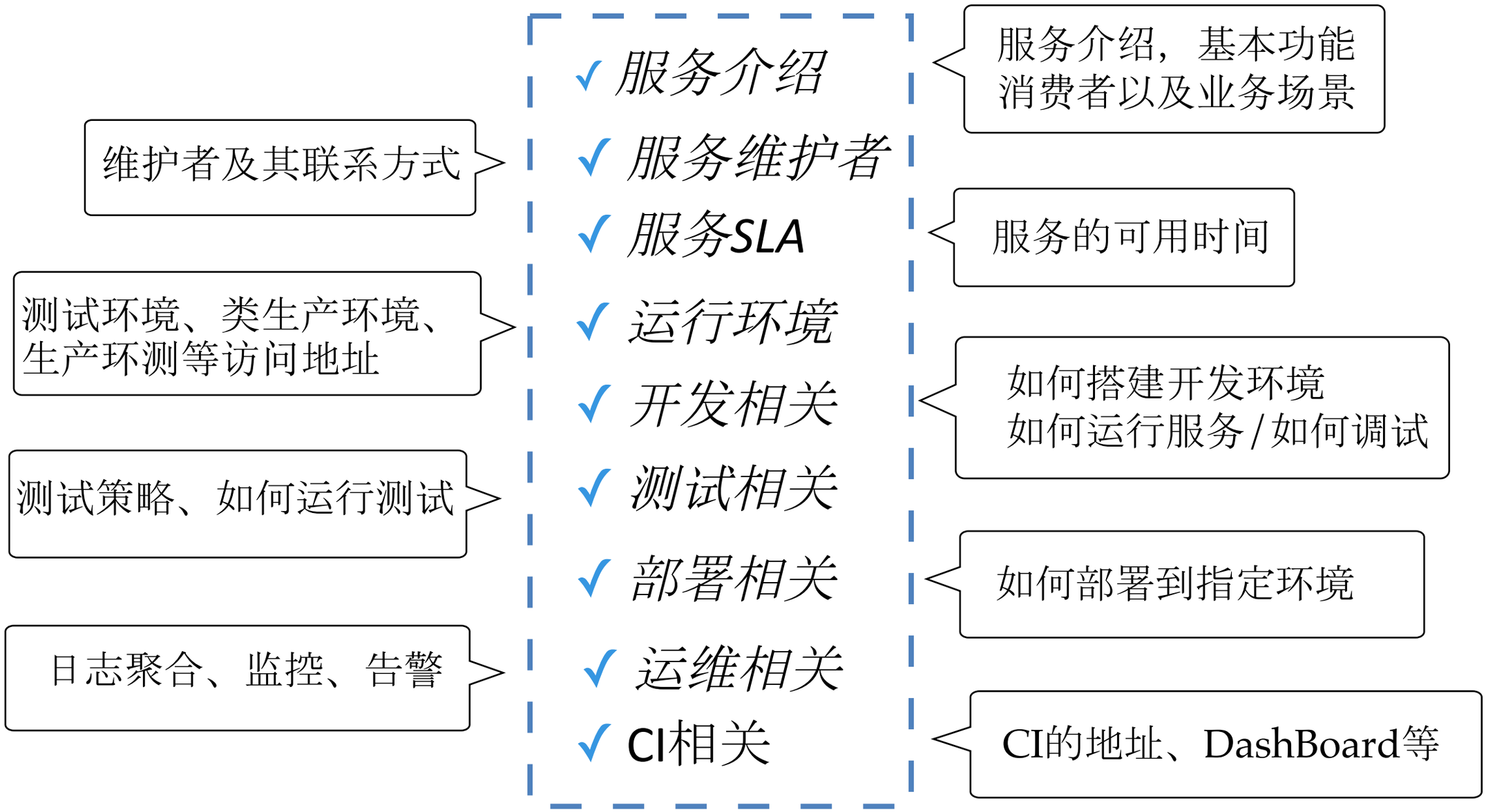
工程化实践



开发实践

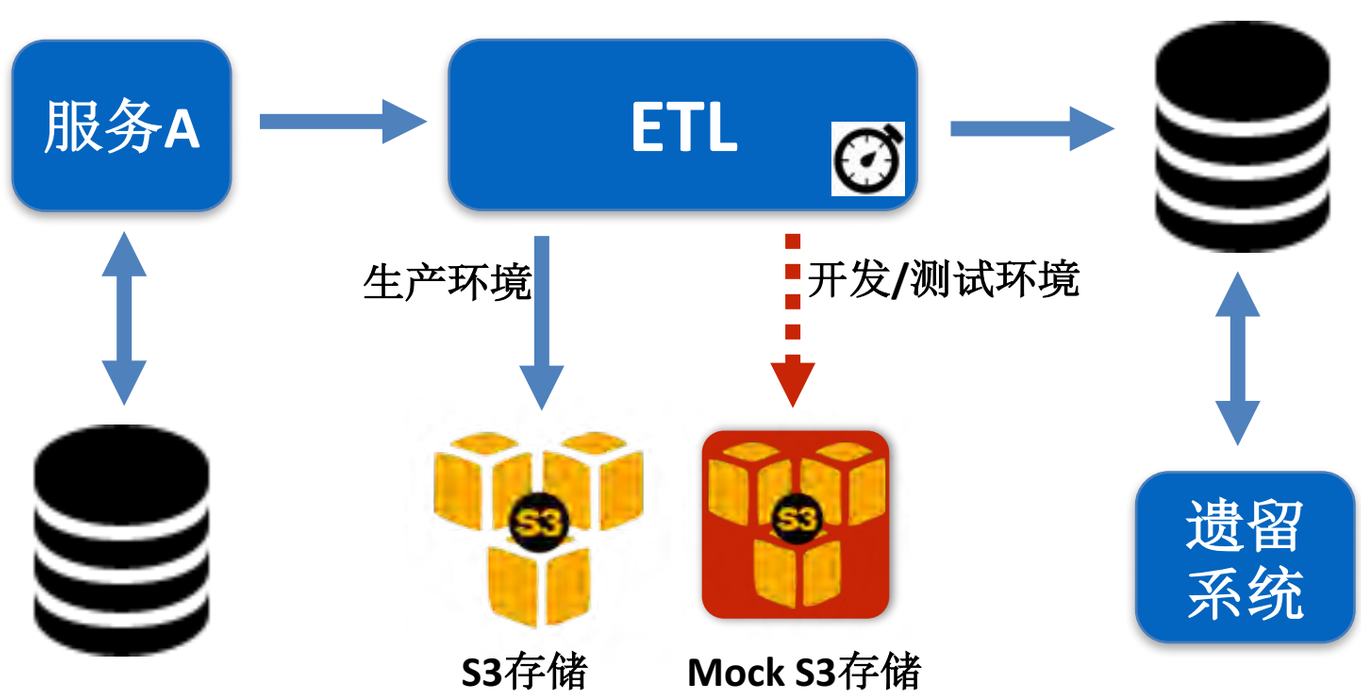
- 独立代码库
- 自解释文档
- 易于本地运行

自解释文档

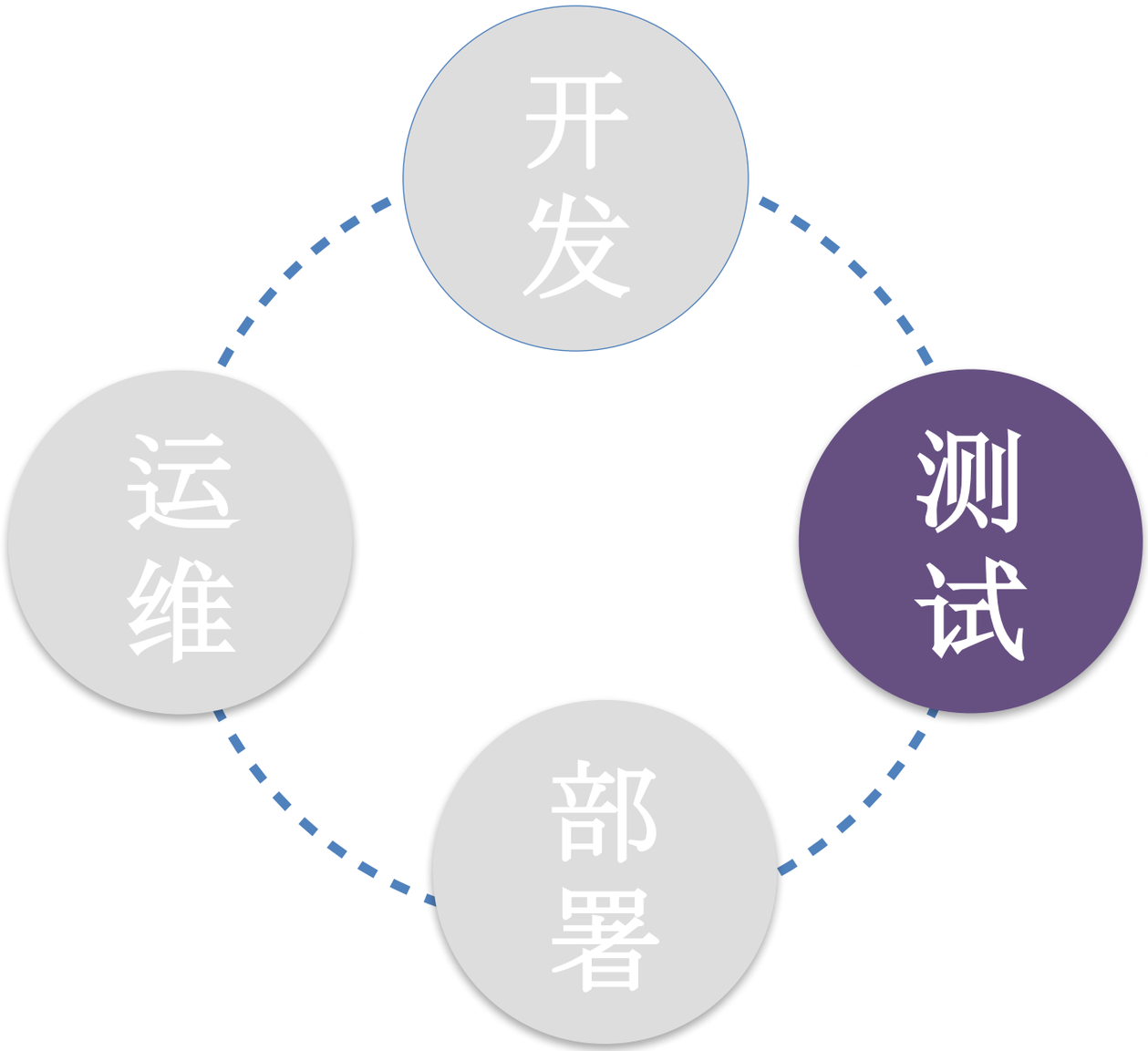

✓ 服务介绍
✓ 服务维护者
✓ 服务SLA
✓ 运行环境
✓ 开发相关
✓ 测试相关
✓ 部署相关
✓ 运维相关
✓ CI相关
服务介绍，基本功能
消费者以及业务场景
维护者及其联系方式
服务的可用时间
测试环境、类生产环境、
生产环测等访问地址
如何搭建开发环境
如何运行服务/如何调试
测试策略、如何运行测试
如何部署到指定环境
日志聚合、监控、告警
CI的地址、DashBoard等

易于本地运行

- 本地环境配置 + Mock/Stub
- 使用Docker-compose
- 依赖环境共享与租用

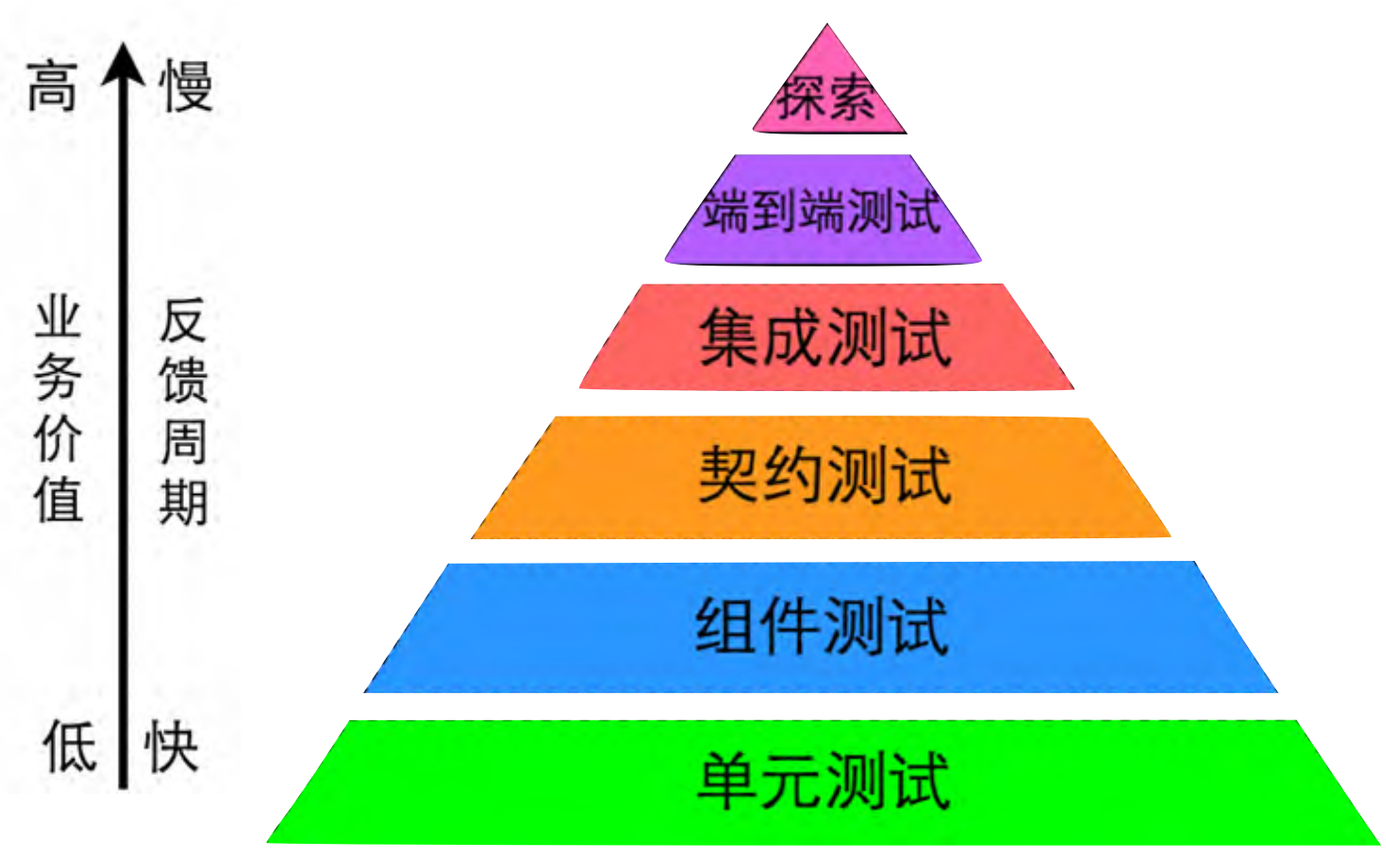


工程化实践

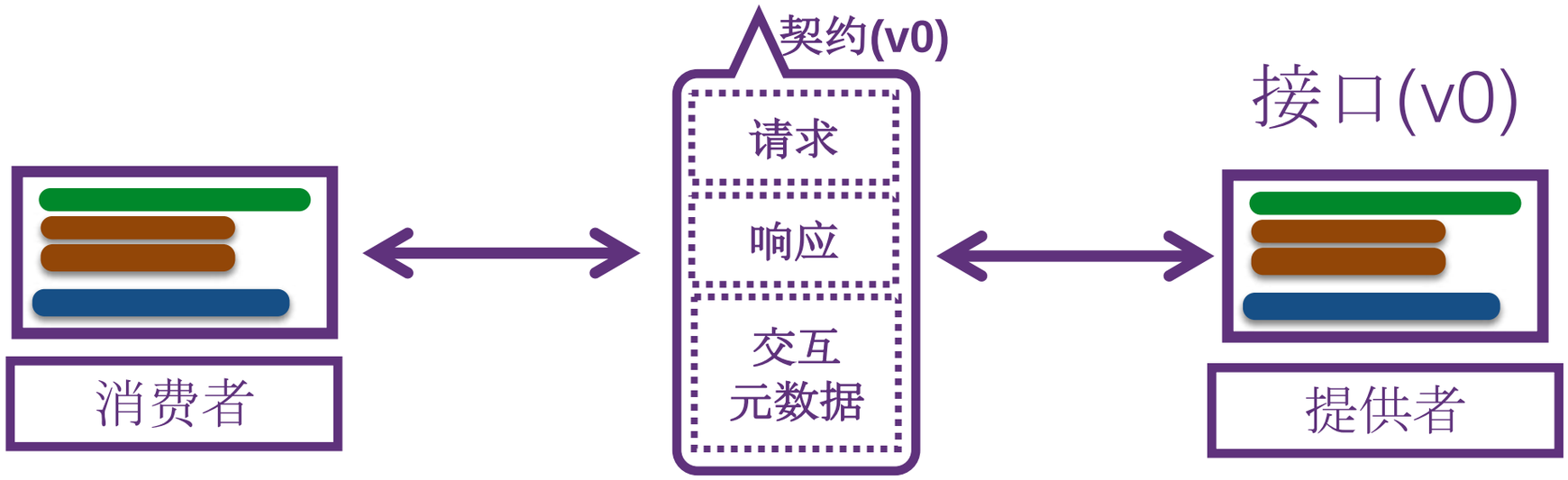


测试实践

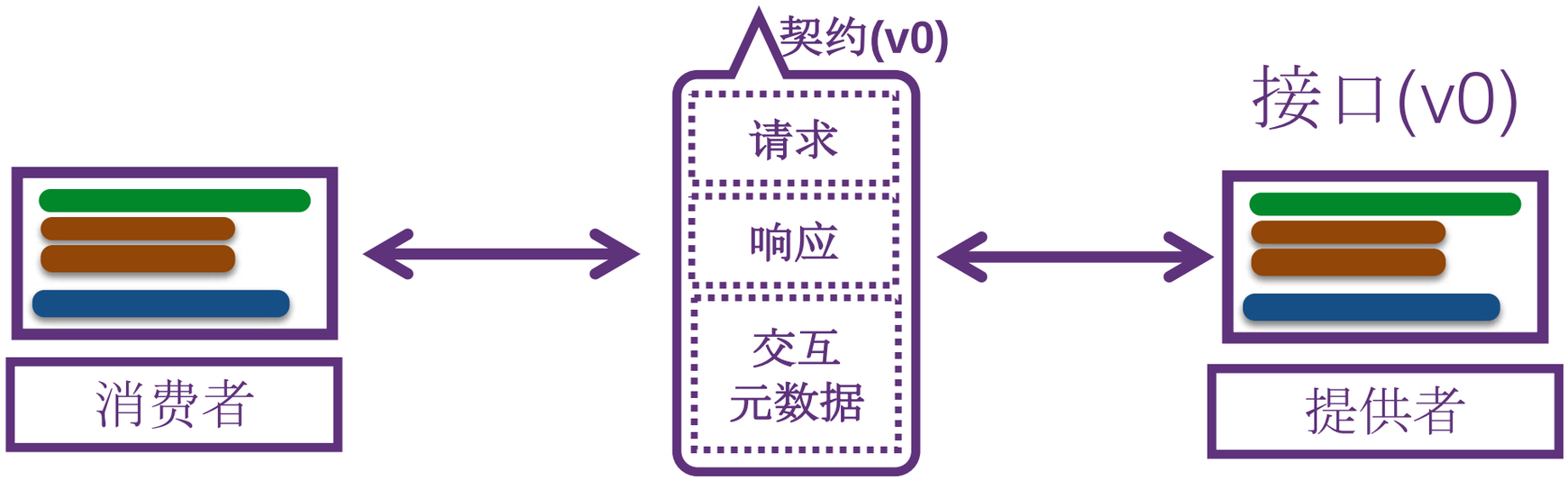
- 测试策略



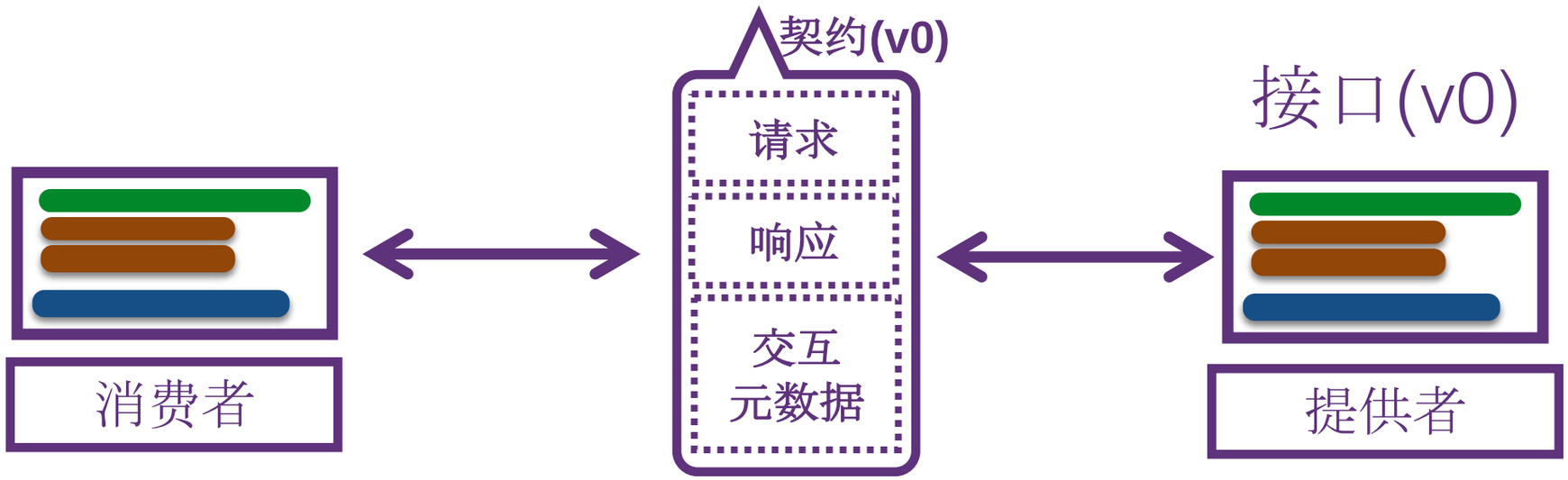
契约测试

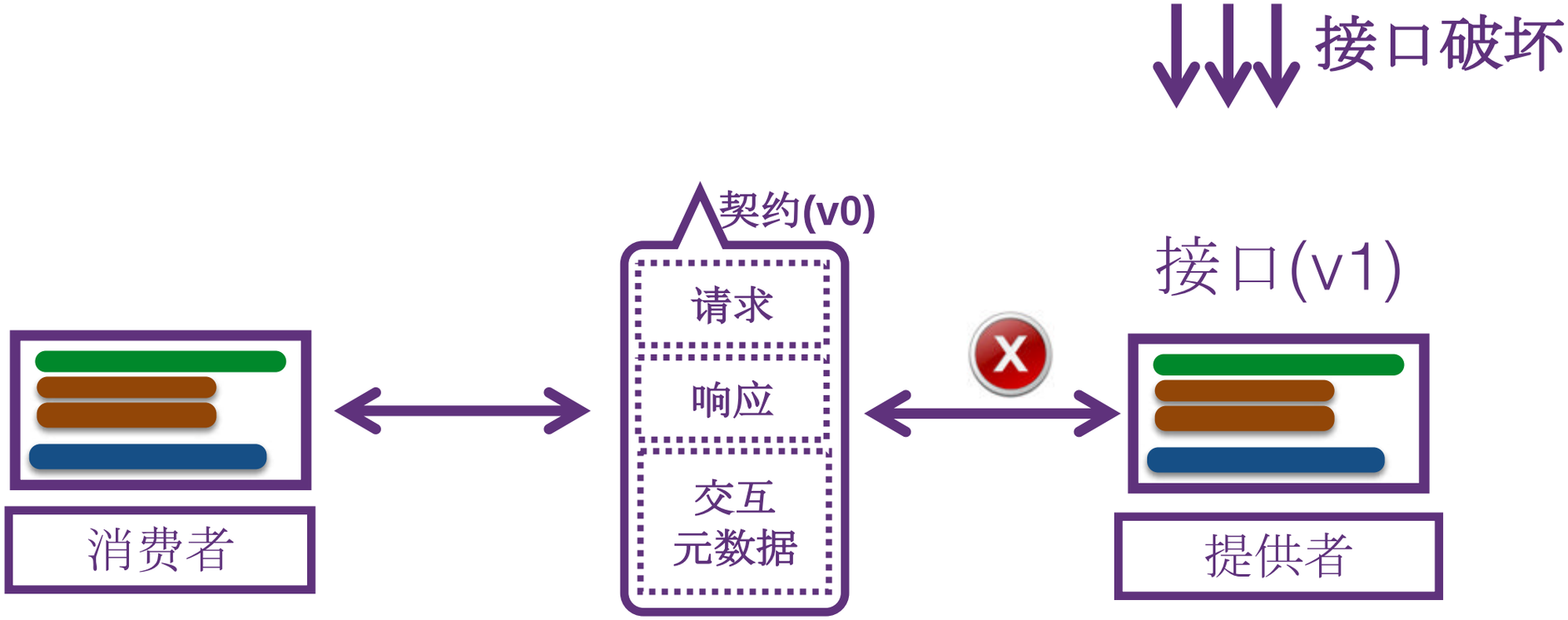


契约测试



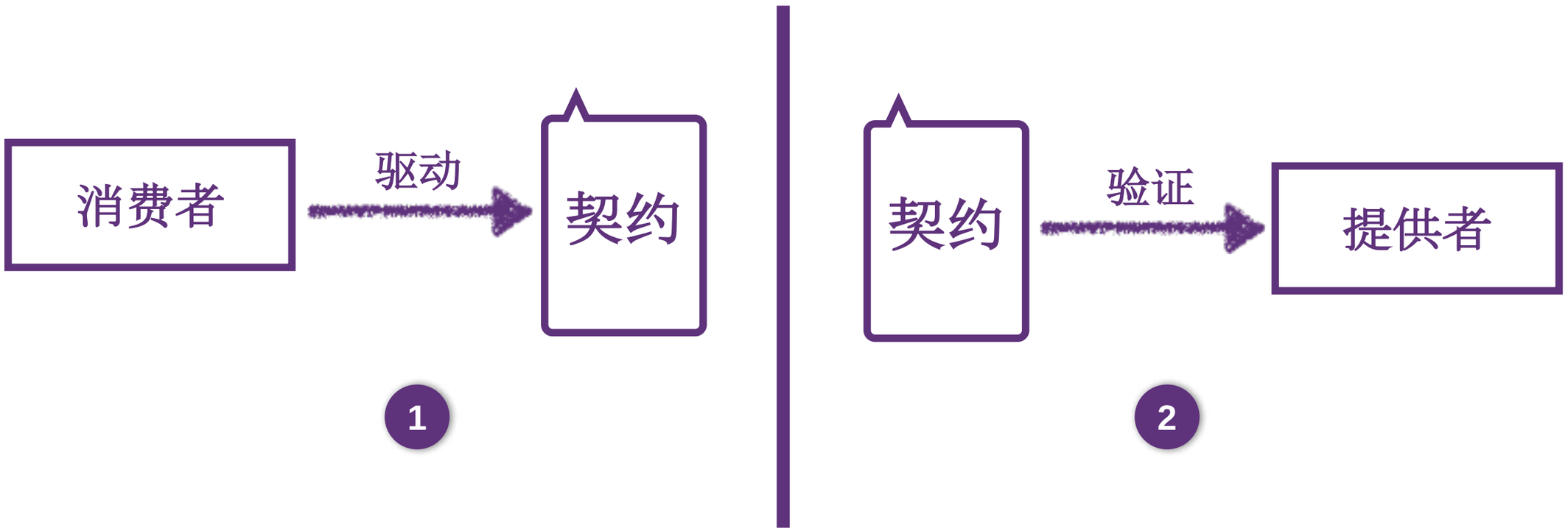
契约测试


契约(v0)
请求
响应
交互
元数据
接口(v0)
消费者
提供者


接口破坏
契约(v0)
请求
响应
交互
元数据
接口(v1)
消费者
提供者

消费者驱动的契约测试

- BDD/TDD模式在架构层面的应用
- 由消费者根据业务驱动出契约
- 基于契约对提供者进行接口验证
- 将在线的集成测试转成离线的单元测试



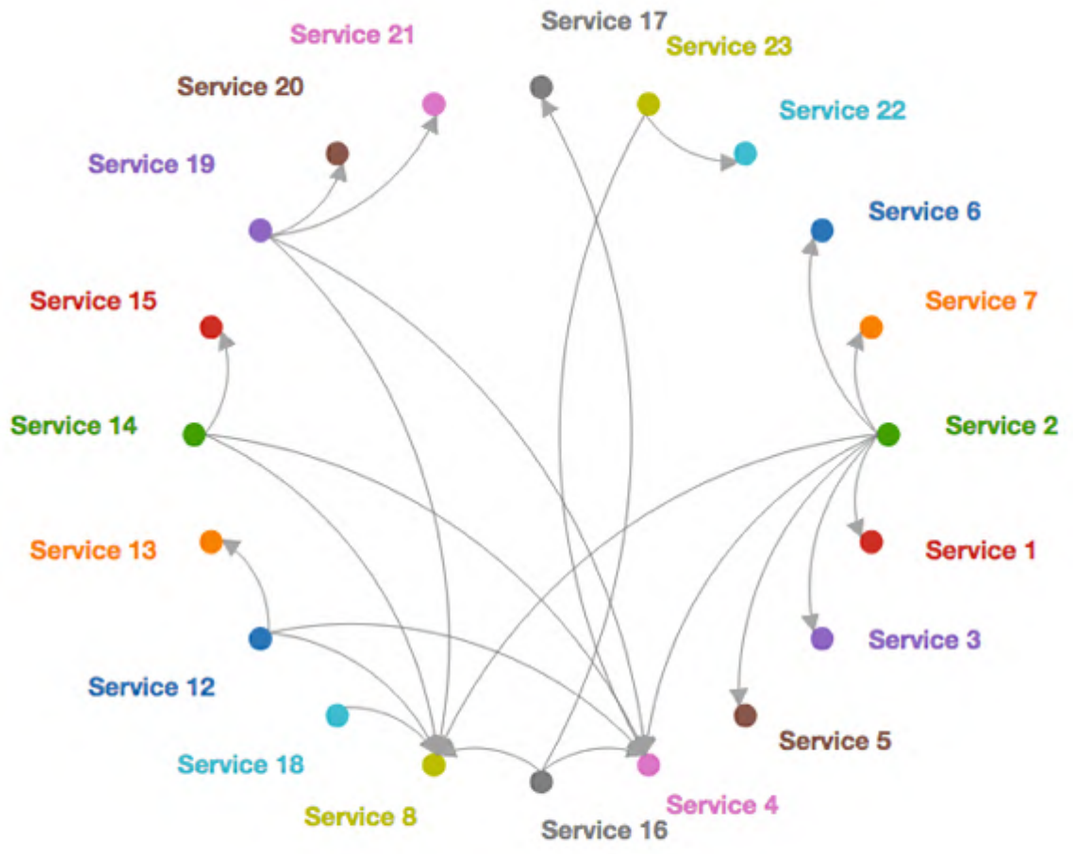
PACT

多语言的支持

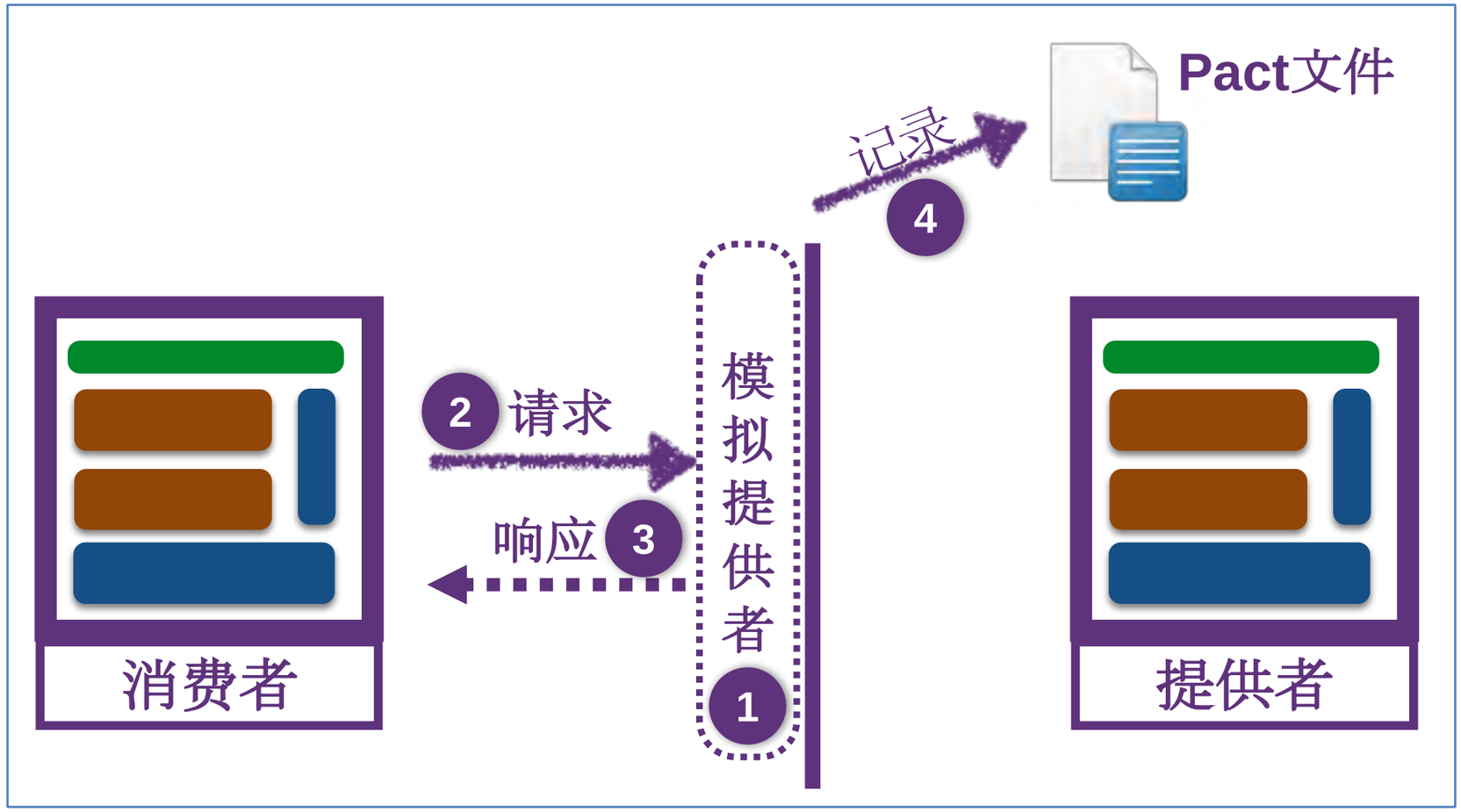
Pact Implementations

Ruby Pact:
https://github.com/realestate-com-au/pact
JVM Pact:
https://github.com/DiUS/pact-jvm
.Net Pact:
https://github.com/SEEK-Jobs/pact-net
JS Pact:
https://github.com/DiUS/pact-consumer-js-dsl
Go Pact:
https://github.com/SEEK-Jobs/pact-go
Swift Pact:
https://github.com/DiUS/pact-consumer-swift

契约共享和重绘

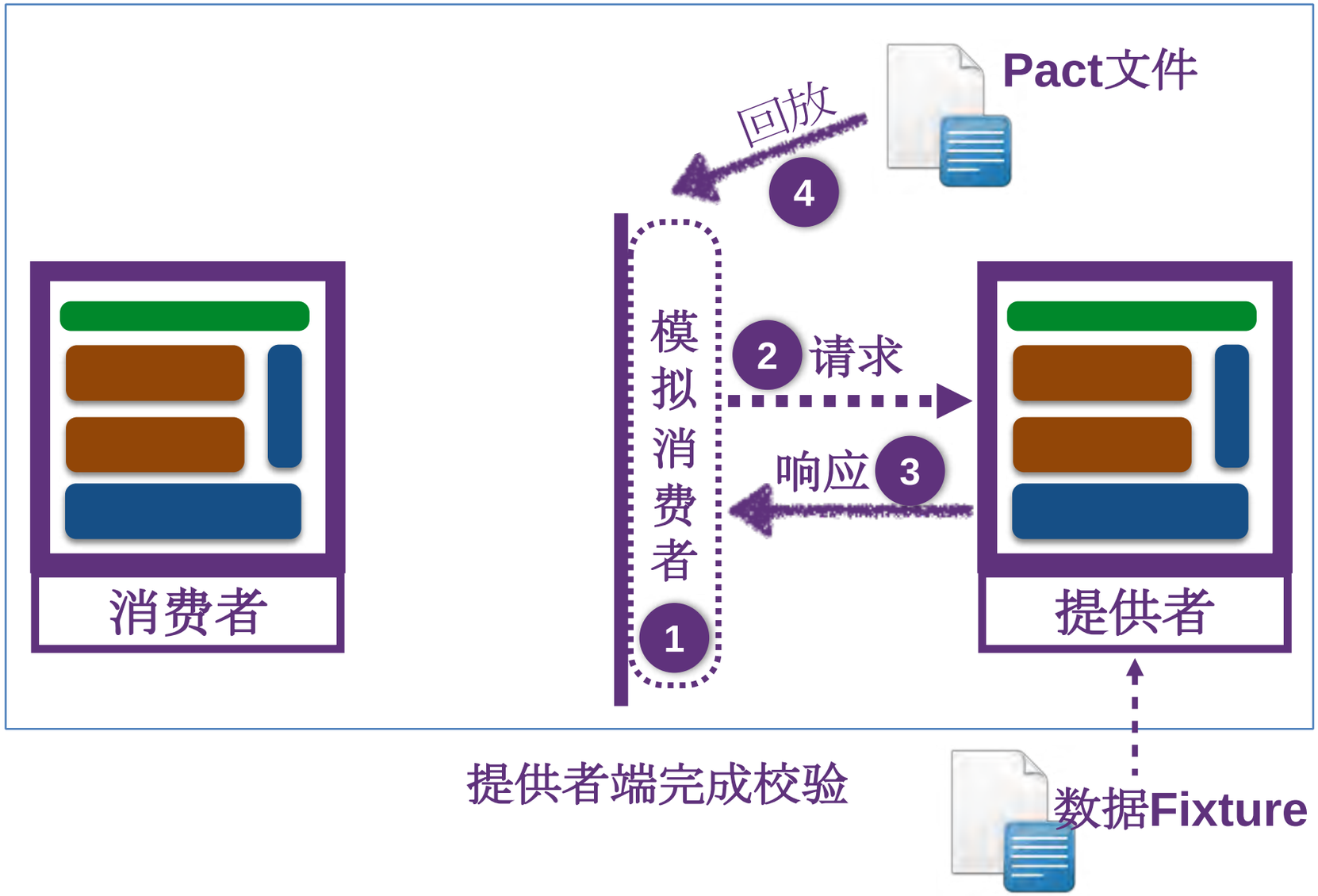


Pact的工作原理

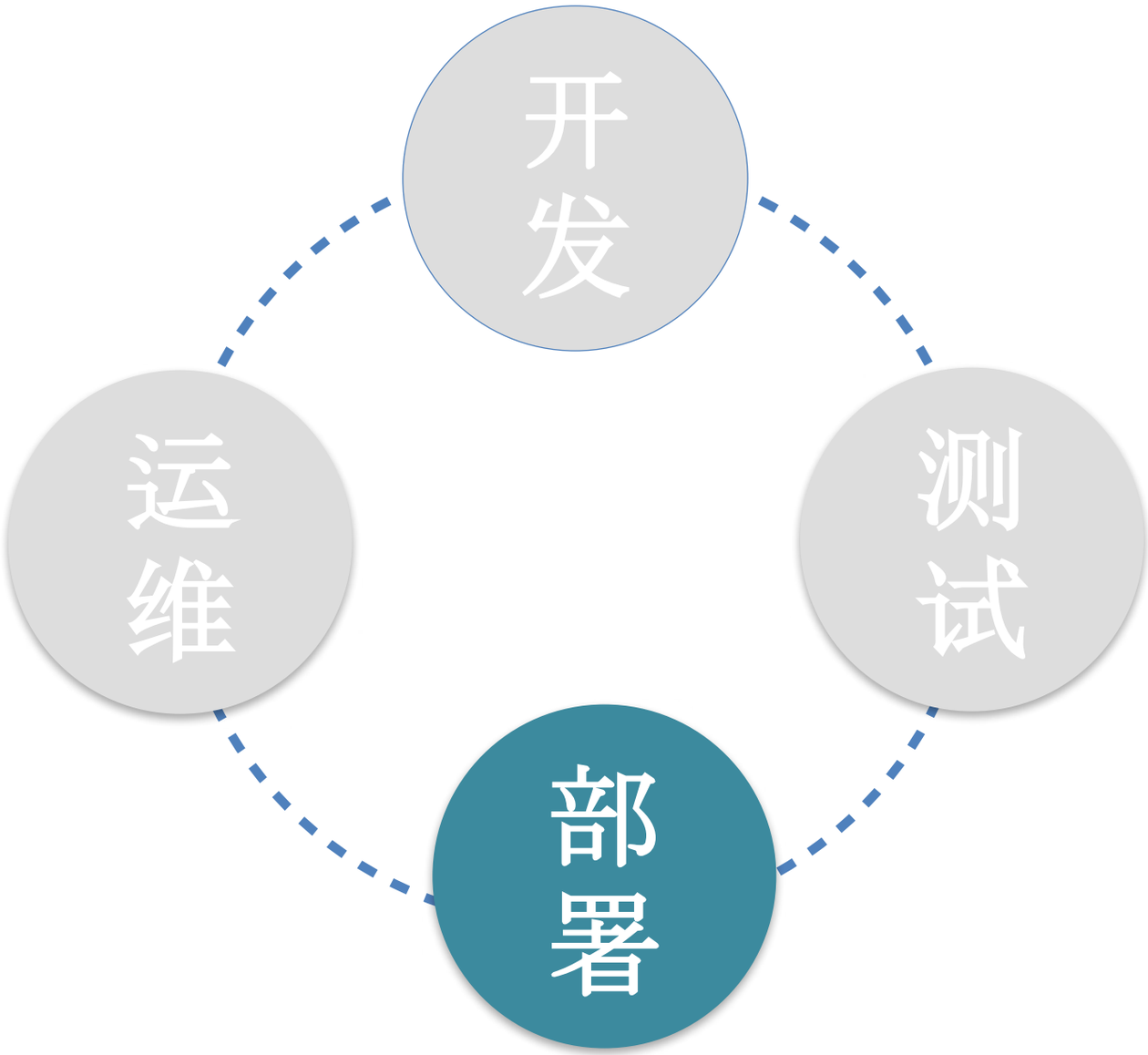


消费者端生成契约

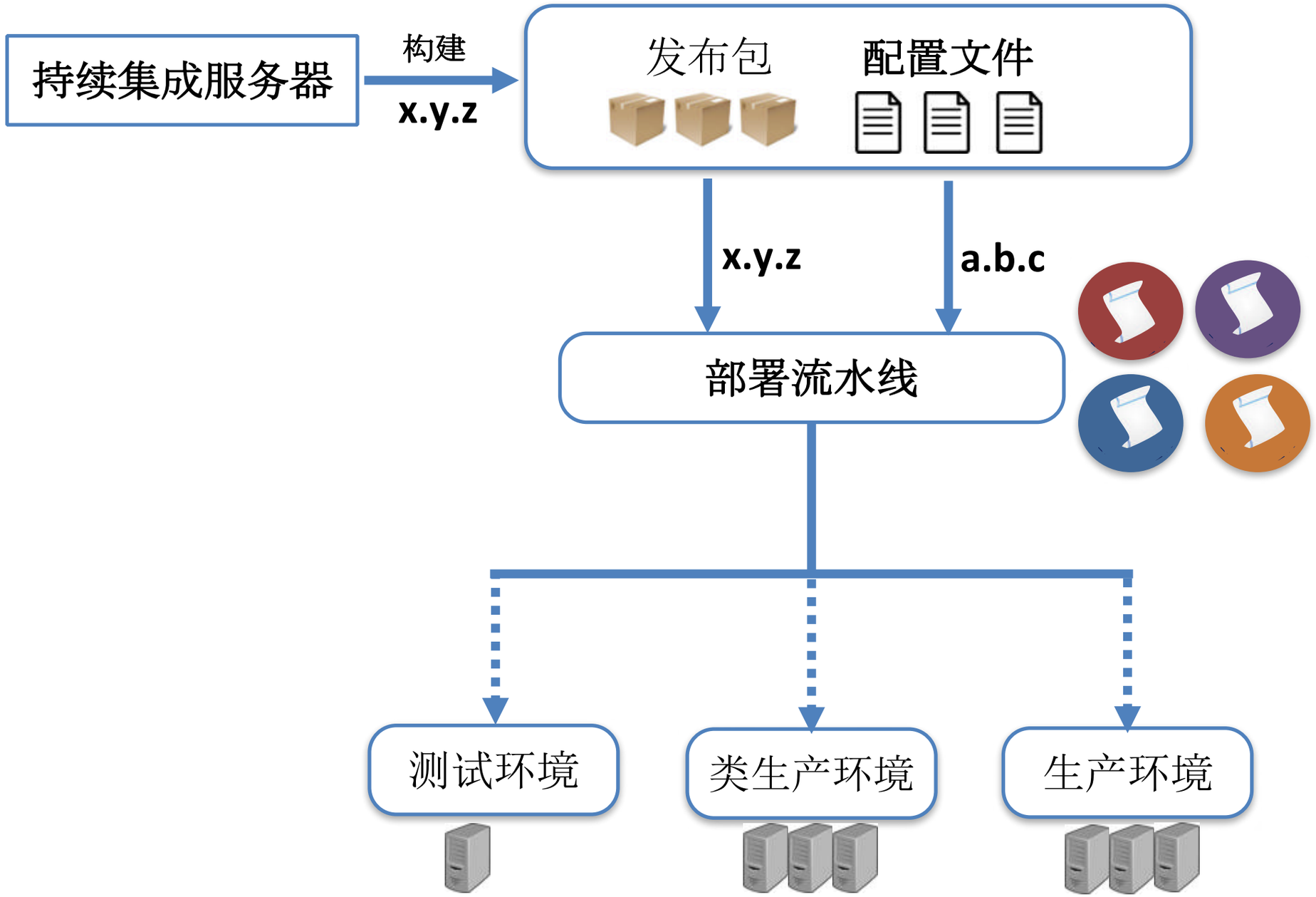
Pact的工作原理



工程化实践



部署核心流程

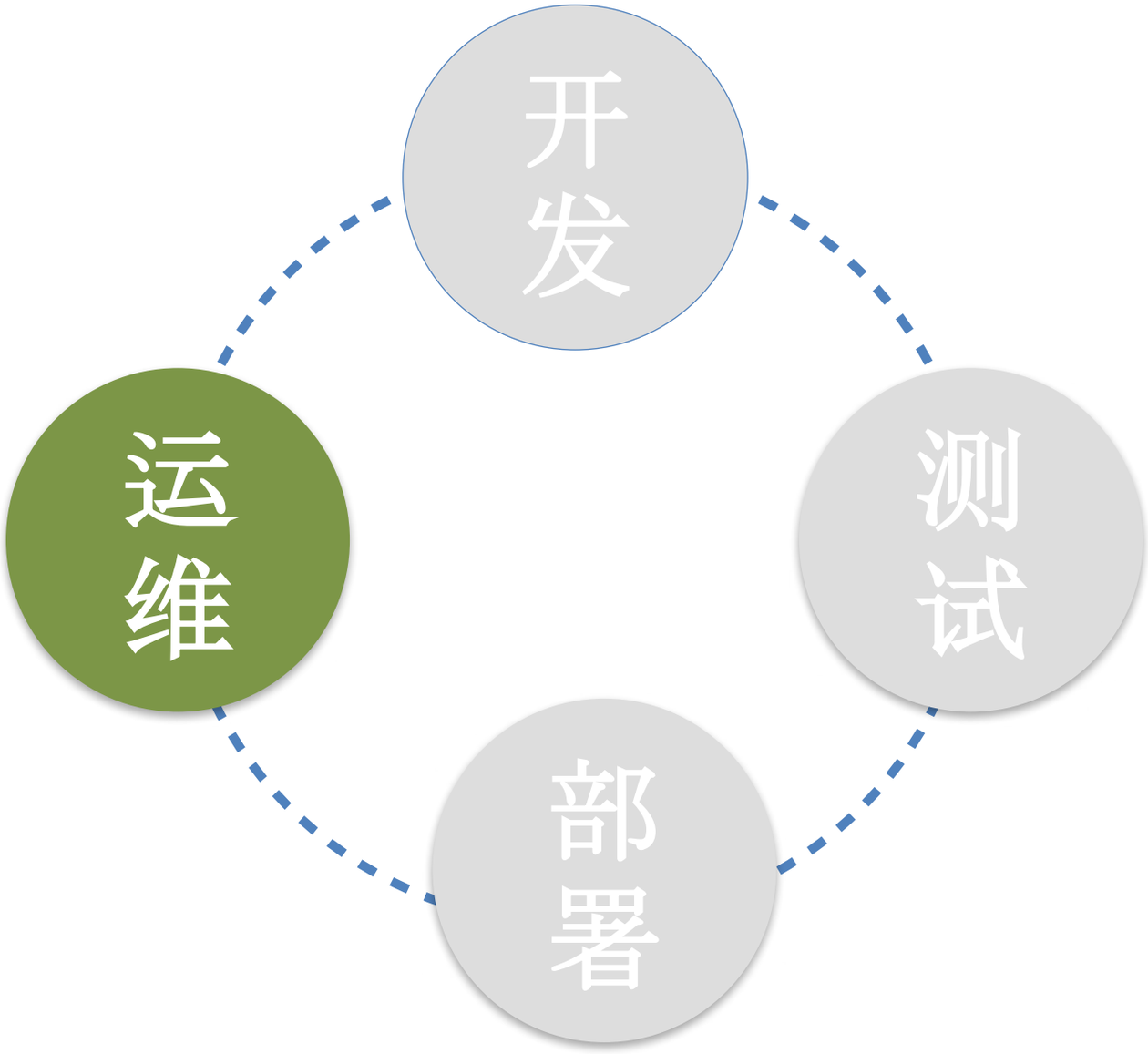


部署优化原则

- 标准化的包管理(及版本化)
- 一条命令完成部署

deploy *[service-name] [service-version] [env]*

工程化实践



运维实践

- 监控
- 告警
- 日志聚合

运维实践

- 监控类型

系统监控	应用监控	业务监控
– 磁盘	– 语言相关(JVM)	– 打点记录
– CPU	– 响应时间	– 上报数据
– 内存	– 健康检查	– 统计分析
……	……	……

运维实践

- 告警

通知方式

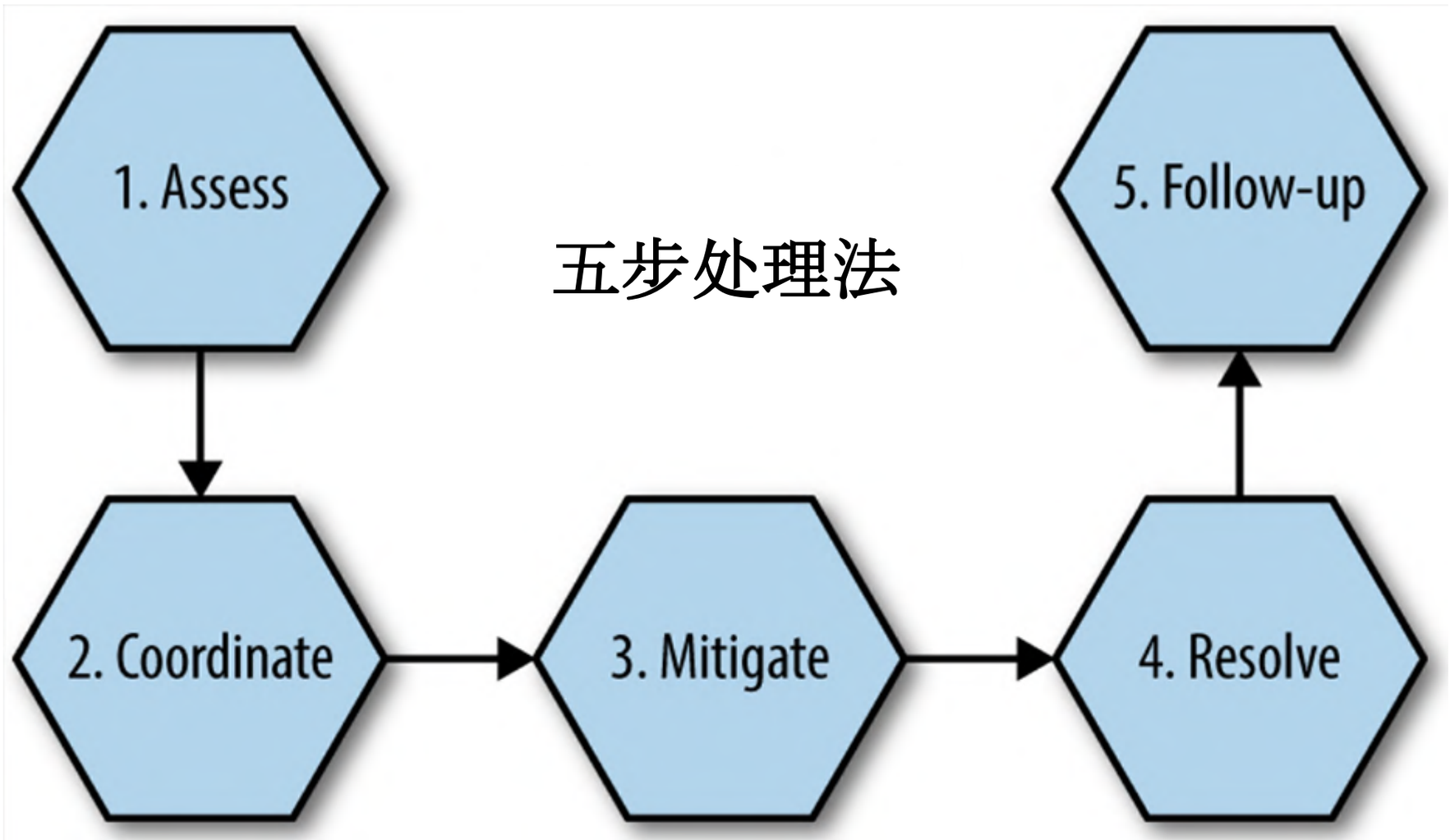
- 电话
- 短信
- 邮件
- IM工具
- ……

告警升级

- OnCall 工程师
- Backup工程师
- 服务Owner
- 产品经理
- ……

运维实践

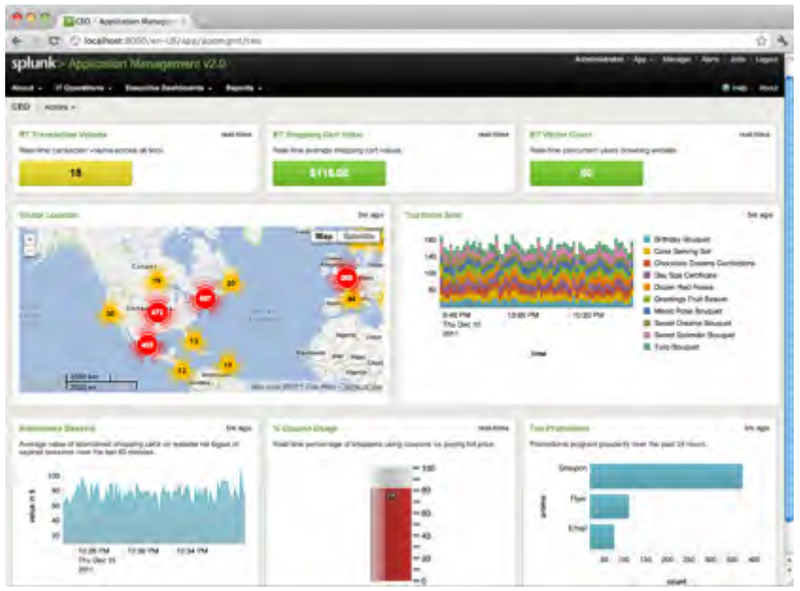
清晰的告警处理流程，以及组内角色的轮换



五步处理法

运维实践

- 日志聚合
  - 集中化收集日志
  - 可视化、搜索与过滤
  - 能够反映任何时间段服务的状态



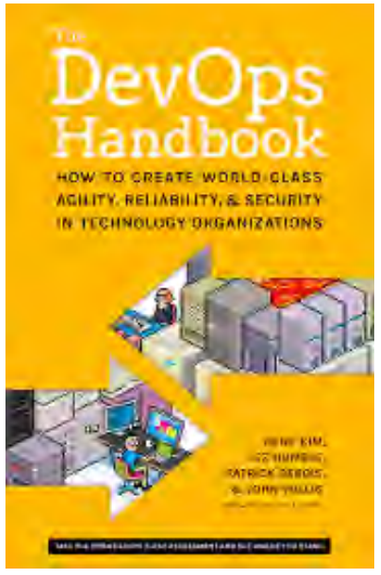
总结

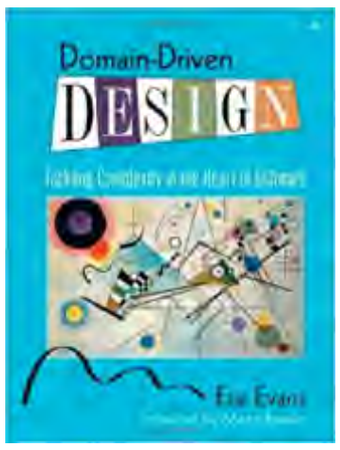
- 微服务架构与DevOps
  - 以缩短交付周期为核心，基于DevOps的演进式架构
- 微服务架构的生态系统
  - 接入层、业务层、支撑层、基础设施、交付流水线
- 微服务架构的工程实践
  - 开发、测试、部署、运维

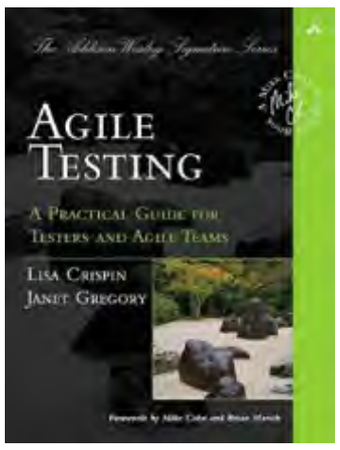
推荐书籍


持续交付
发布可靠软件的系统方法
Jez Humble
David Farley


The Pragmatic Programmers
Release It!
Design and Deploy
Production-Ready Software
Michael T. Nygard


The
DevOps
Handbook
HOW TO CREATE WORLD-CLASS
AGILITY, RELIABILITY, & SECURITY
IN TECHNOLOGY ORGANIZATIONS


Domain-Driven
DESIGN
Eric Evans


AGILE
TESTING
A Practical Guide for
Testers and Agile Teams
Lisa Crispin
Janet Gregory


O'REILLY
Building
Microservices


微服务
架构与实践

谢谢

高效运维社区
GreatOPS Community

GOPS2017
Shenzhen

会议

- 3月18日 DevOpsDays 北京
- 8月18日 DevOpsDays 上海
- 全年 DevOps China 巡回沙龙
- 4月21日 GOPS深圳
- 11月17日 DevOps金融上海

培训

- EXIN DevOps Master 认证培训
- DevOps 企业内训
- DevOps 公开课
- 互联网运维培训

咨询

- 企业DevOps 实践咨询
- 企业运维咨询

商务经理：刘静女士
电话 / 微信：13021082989
邮箱：liujing@greatops.com

Thanks

高效运维社区

开放运维联盟

荣誉出品

想第一时间看到

高效运维社区公众号

的好文章吗？

请打开高效运维社区公众号，点击右上角小人，如右侧所示设置就好