

ThoughtWorks®

# DEVOPS从理论到实践

---

*ThoughtWorks* 杜屹东

# 期望

---



# 概览

---

- 什么是DevOps
- 为什么要实践 DevOps
- 如何实践 DevOps
- 未来 & 趋势

# DEVOPS 定义

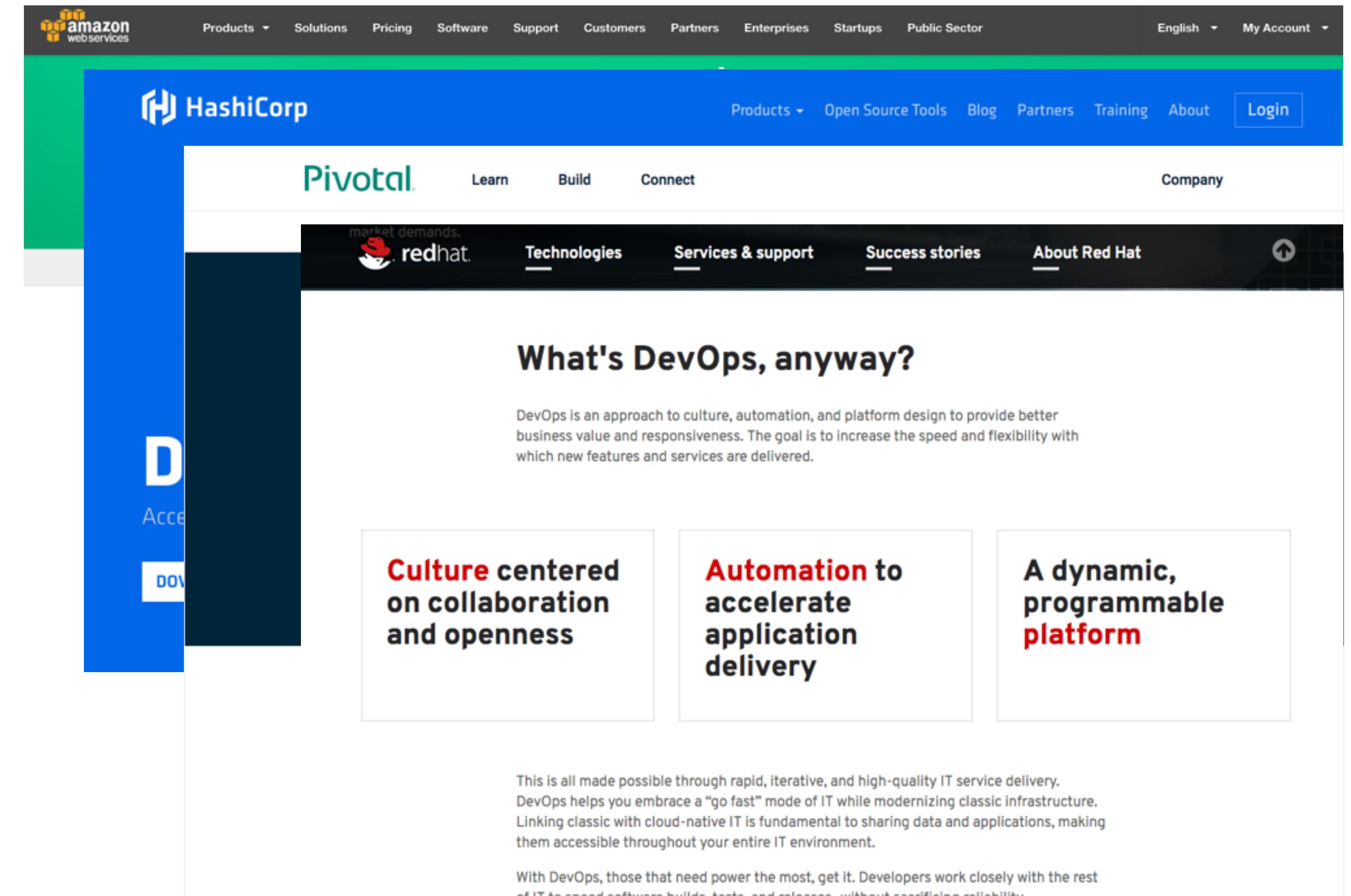
**AWS:** DevOps 是一种文化、哲学、实践、工具的集合。

**HashiCorp:** 为任何应用程序提供，安全和运行任何基础架构。

**Pivotal:** DevOps是一种工作方式，专注于定期运送满足业务需求的优质软件。

**Redhat:** DevOp是

- 以合作、开放为中心的文化
- 用自动化的方式加速应用交付
- 一个动态的可编程的平台



# DEVOPS 是一个流行词



# DEVOPS 的由来

---

2007, 比利时

2008 年 8月, 加拿大多伦多

2009 年 6月, 美国圣荷西

2009 年 10月, 比利时根特

2010 年

2010 年, 德国汉堡



*Patrick Debois* “能否把敏捷的实践引入Ops团队”

*Agile Conference 2008* Andrew Shafer: “*Agile Infrastructure*”

第二届 *Velocity* 大会上一个轰动世界的演讲

10+ Deploys Per Day: Dev and Ops Cooperation at Flickr

*DevOpsDays* 和 *DevOps*

*The Agile Admin*博客发表“*What is DevOps*”

对不起, 《持续交付》来晚了

# DEVOPS 框架 - THE THREE WAYS

---

## The principles underpinning DevOps

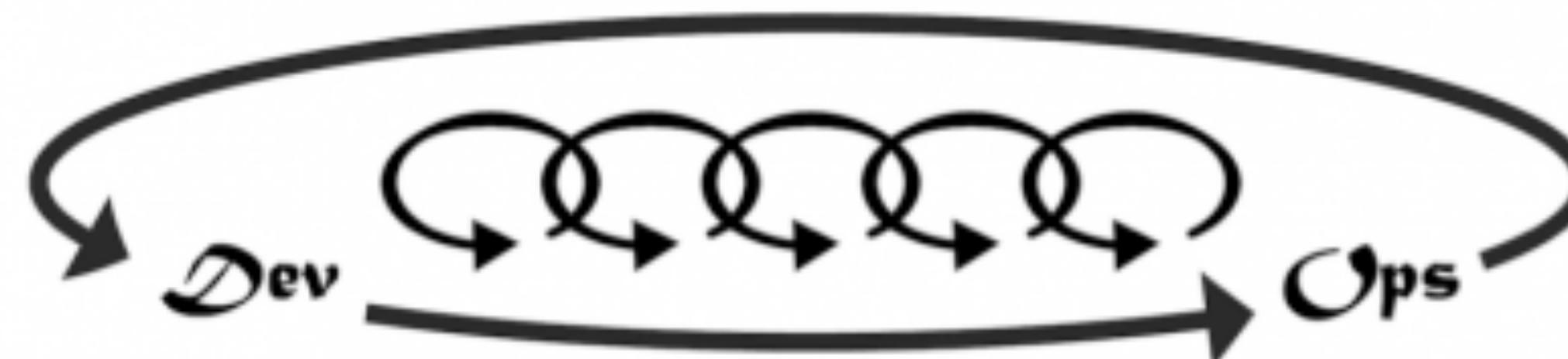
**The First Way:**  
Systems Thinking



**The Second Way:**  
Amplify Feedback Loops



**The Third Way:**  
Culture Of Continual Experimentation And  
Learning



# DEVOPS 框架 - CALMS

---

Quantifying DevOps Capability: It's Important To Keep CALMS

## CULTURE

建立包括运维在内的跨职能协作文化,具有共同目标的一体化团队;这是DevOps运动的根本!

## AUTOMATION

自动化一切可以自动化的,降低部署和发布的难度

## MEASUREMENT

通过建立有效的监控与度量手段来获得反馈,推动产品和团队的持续改进,支持业务决策

## LEAN

协作文化,自动化,和有效的反馈,这些实施是个长期的过程,需要以精益的方式 小步快跑,对过程与技术进行持续改善

## SHARING

不同职能、不同产品之间分享开发和运维过程中的想法,知识,问题,和失败,提升能力

# 为什么要实践DEVOPS

## 更高价值的产品



- 形成特性提出到运营数据、用户反馈验证的实验性交付闭环，基于实际用户反馈调整计划和需求

## 更短的交付周期



- 生产环境部署频率越来越快，简化生产部署流程，且自动化不停机部署

## 更好的质量保障



- 在代码检查，功能和非功能验证，以及部署各方面建立较完善的质量保障体系，尤其是自动化测试集

## 更高绩效的团队



- 包含业务，开发测试，和运维职能在内的一体化团队，以产品交付为共同目标紧密协作，共同承担责任

# DEVOPS 在技术领域的实践

DevOps 运作包括文化（全功能，自运维）和技术（自动化，度量反馈）两方面，而技术能力的改进主要关注以下六个领域：

## 内建质量体系

通过持续代码评审，静态分析，自动化测试，自动部署验证等手段构成一套有效的质量保障体系

## 持续监控

持续对运行环境在系统，应用层面进行监控，及时发现风险或问题，保障系统运行的稳定性

## 环境管理

通过对服务器环境的定义，自动化建立和配置、更新等提高基础设施管理的效率，一致性，并更有效利用资源，可伸缩的架构，保证服务的健壮性

## 持续部署

通过自动化的构建，部署过程快速频繁地将软件交付给用户，提高吞吐量；同时保障过程的安全，平滑，可视

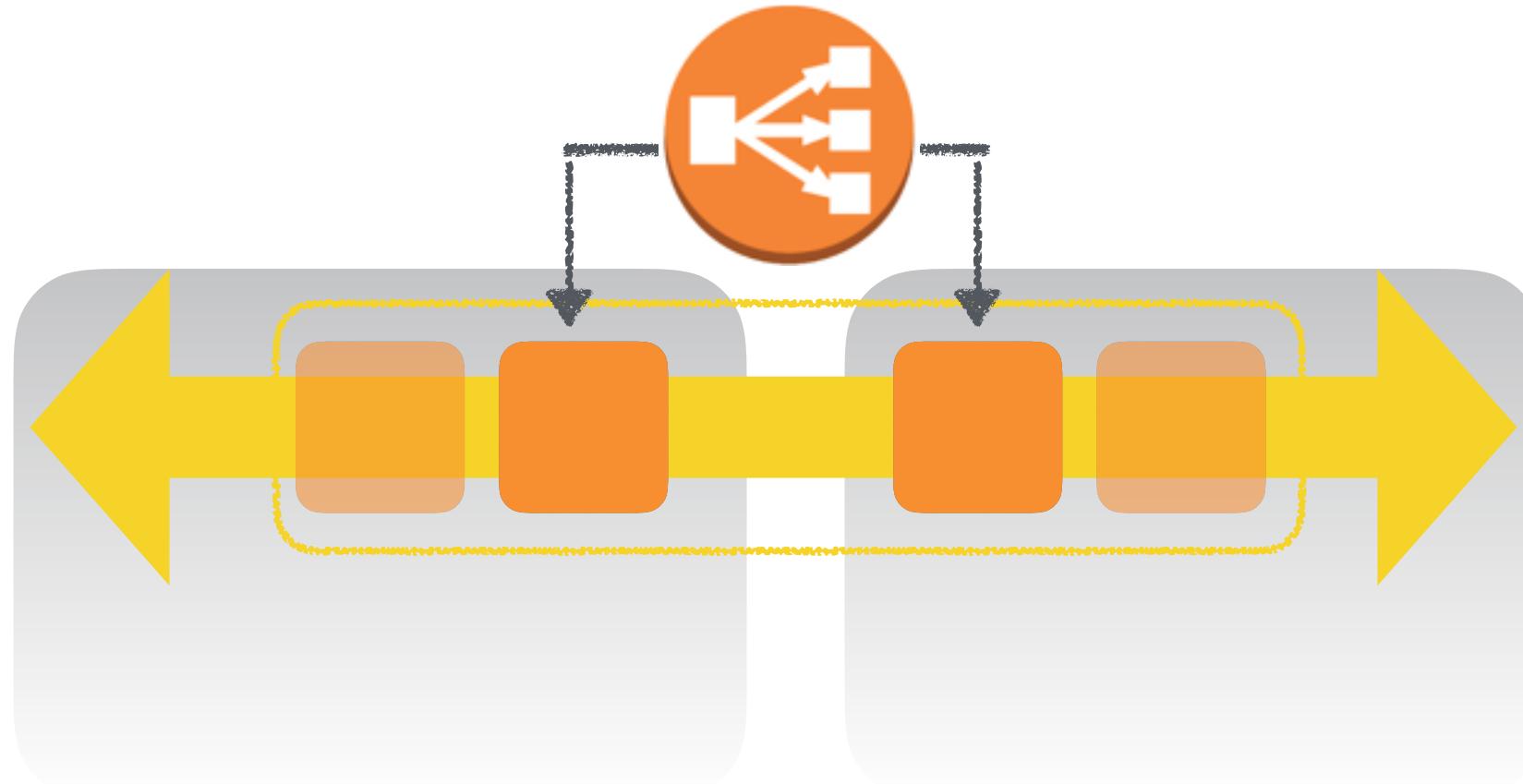
## 度量与反馈

通过对用户行为或业务指标的度量或反馈收集，为产品的决策提供依据

## 支持DEVOPS的松耦合架构

对传统应用架构进行领域组件化，服务化，提升可测试性和可部署性

## 弹性架构

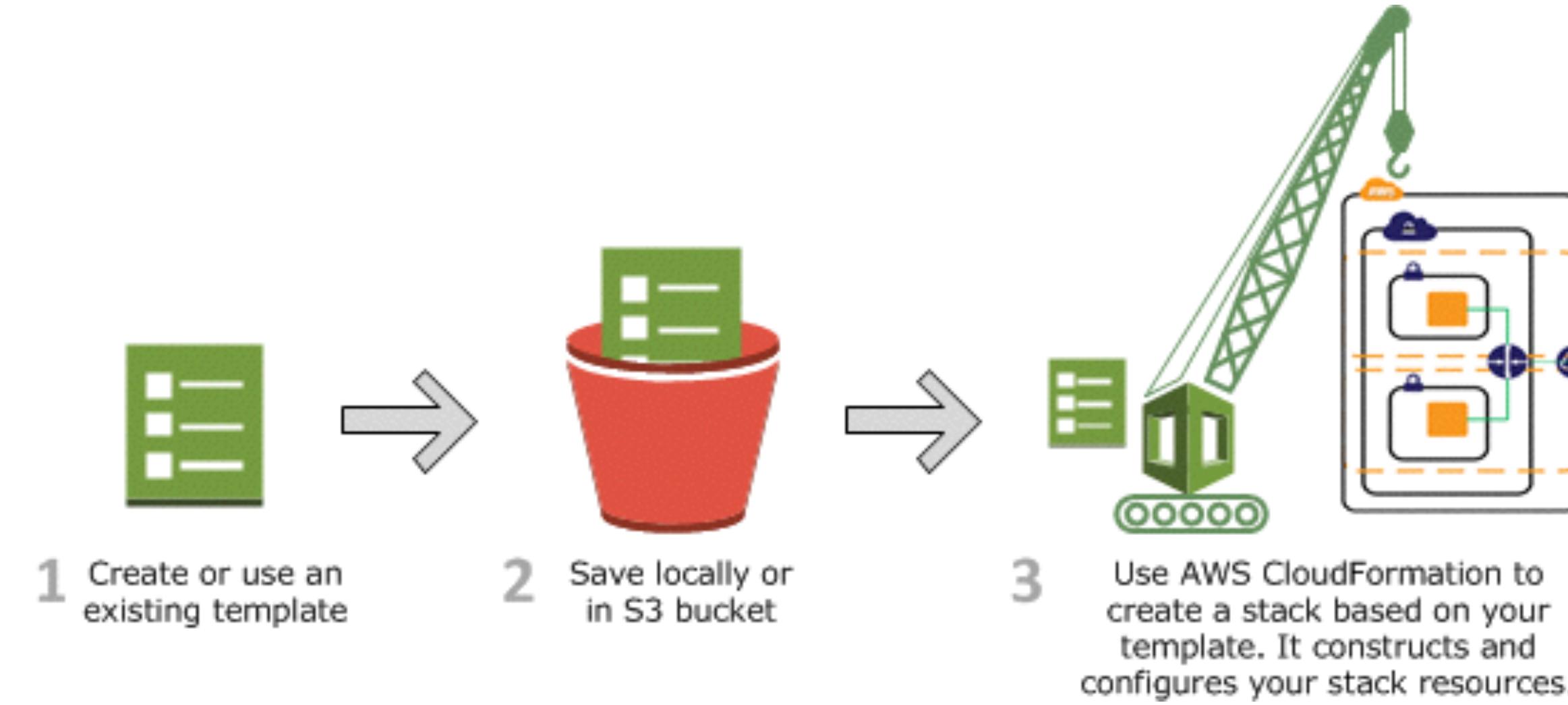


## 自动化部署脚本



A N S I B L E

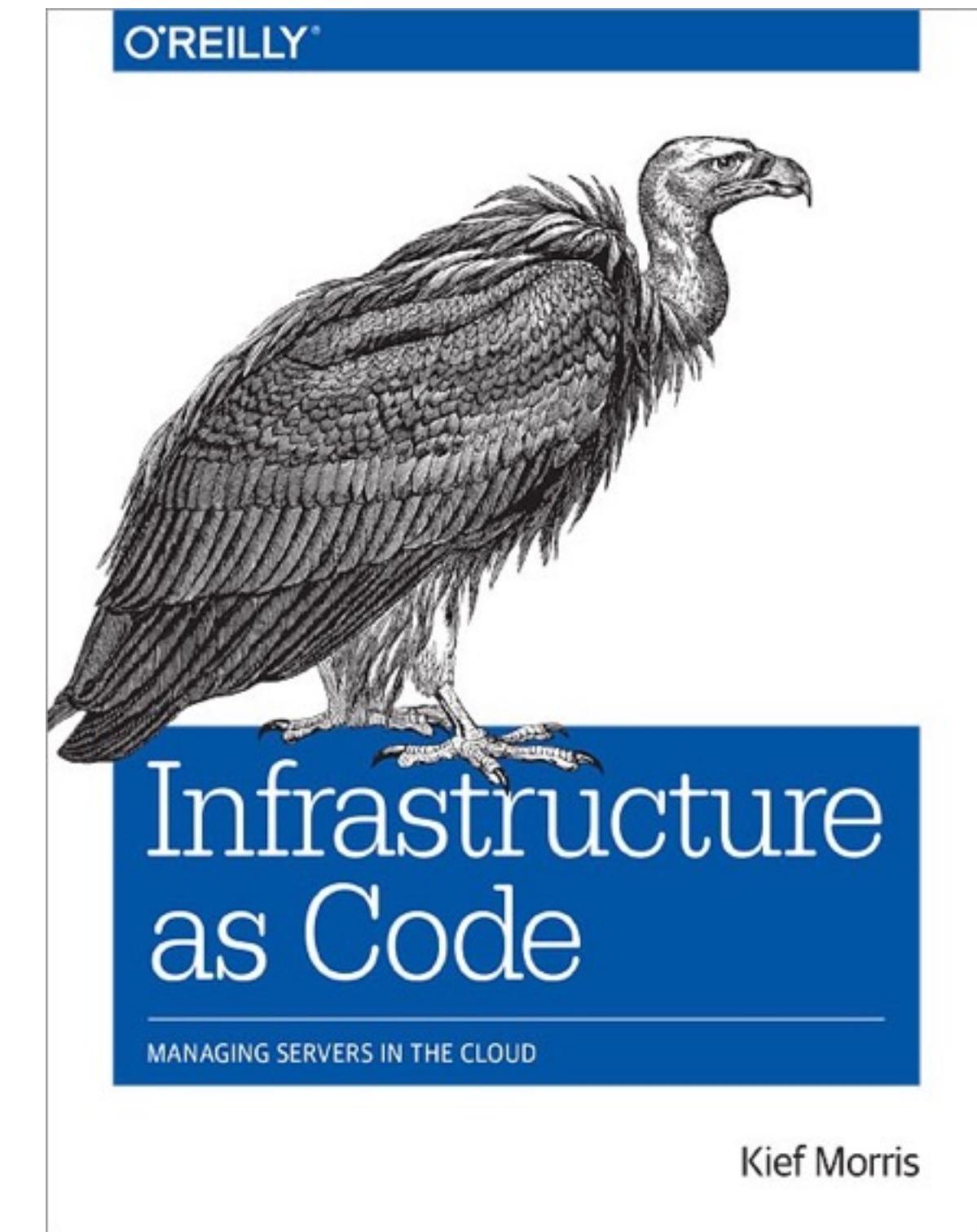
## 基础设施即代码



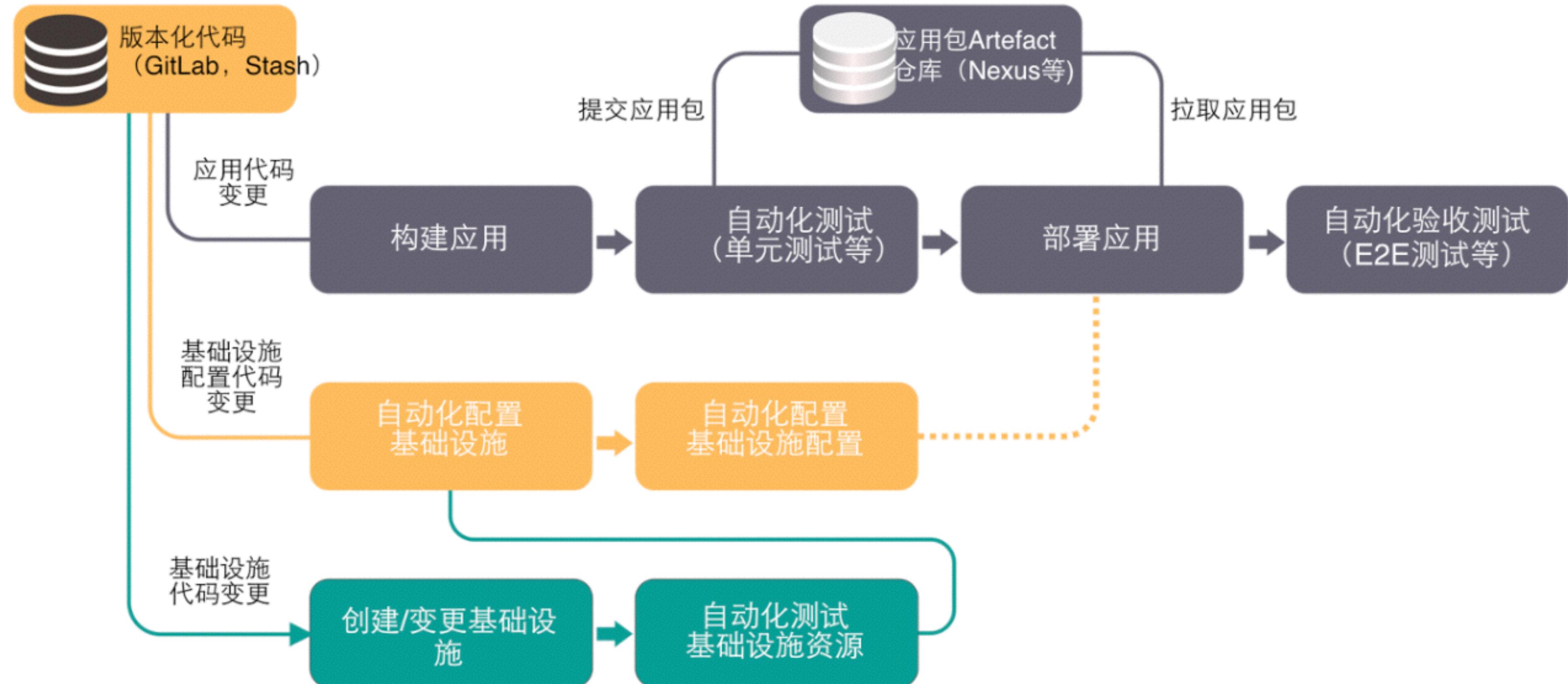
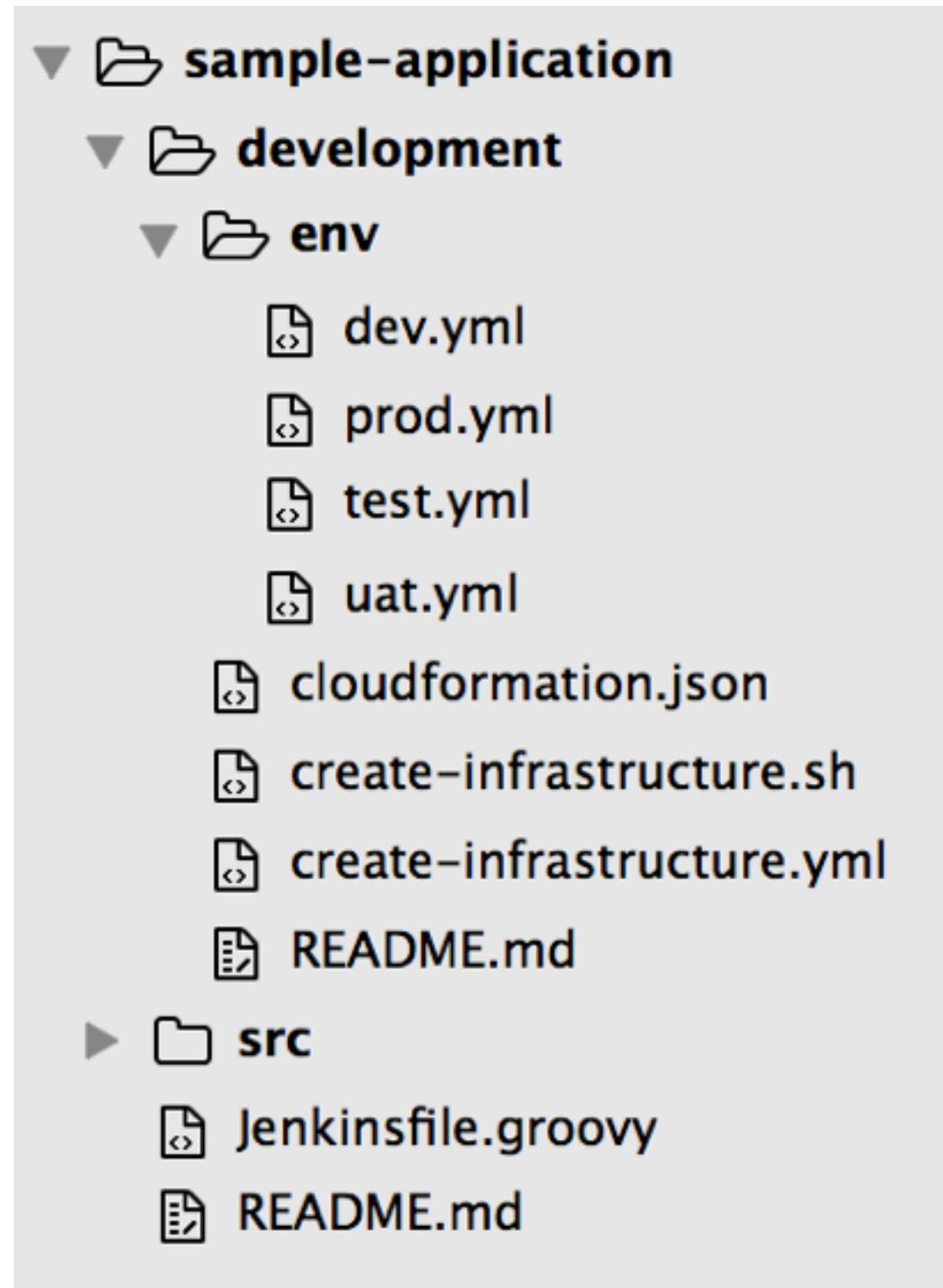
# 基础设施即代码

---

基础设施即代码是一种使用新的技术来构建和管理动态基础设施的方式。它把基础设施、工具和服务以及对基础设施的管理本身作为一个软件系统，采纳软件工程实践以结构化的方式管理对系统的变更。

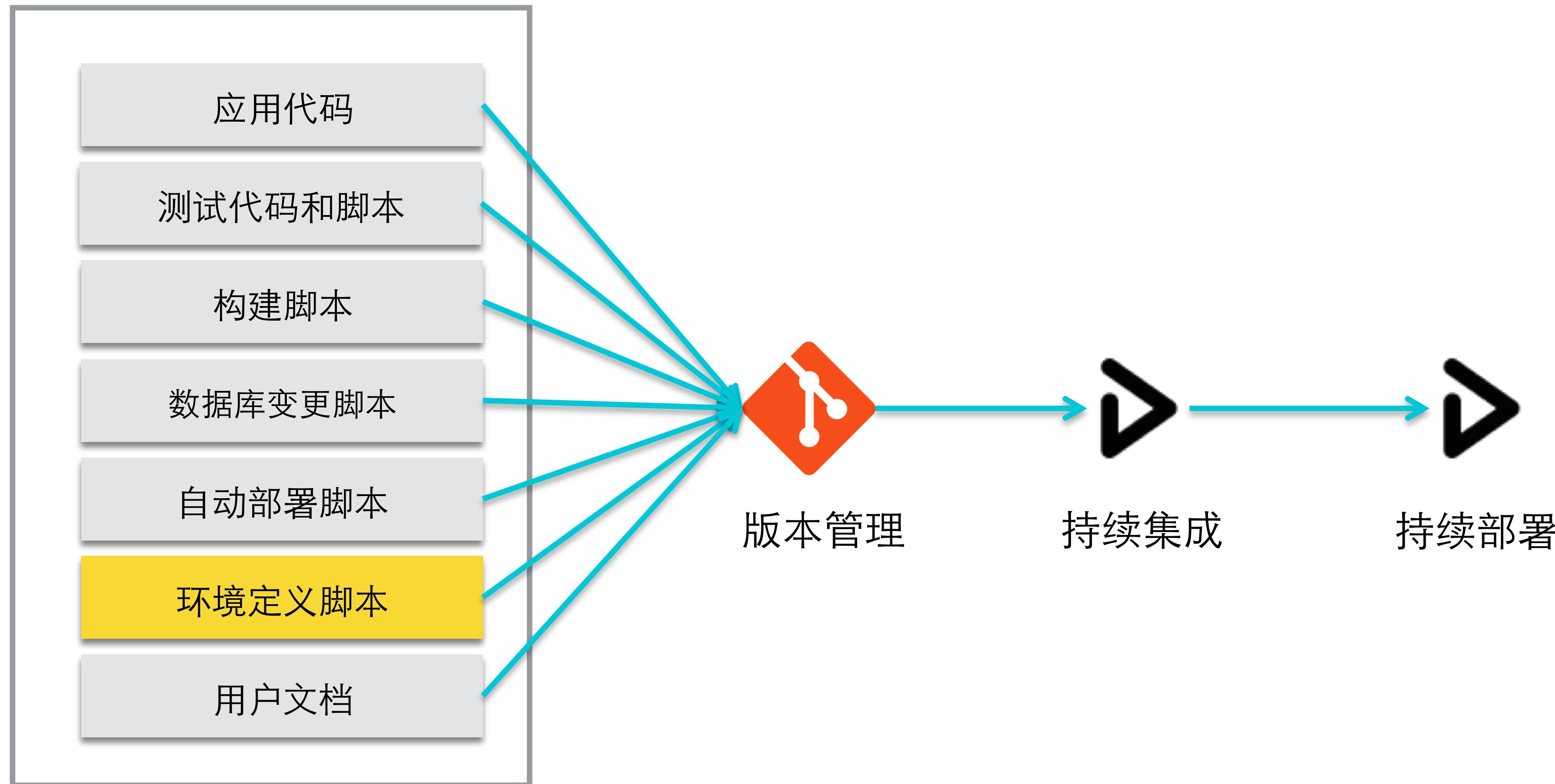


# 基础设施即代码



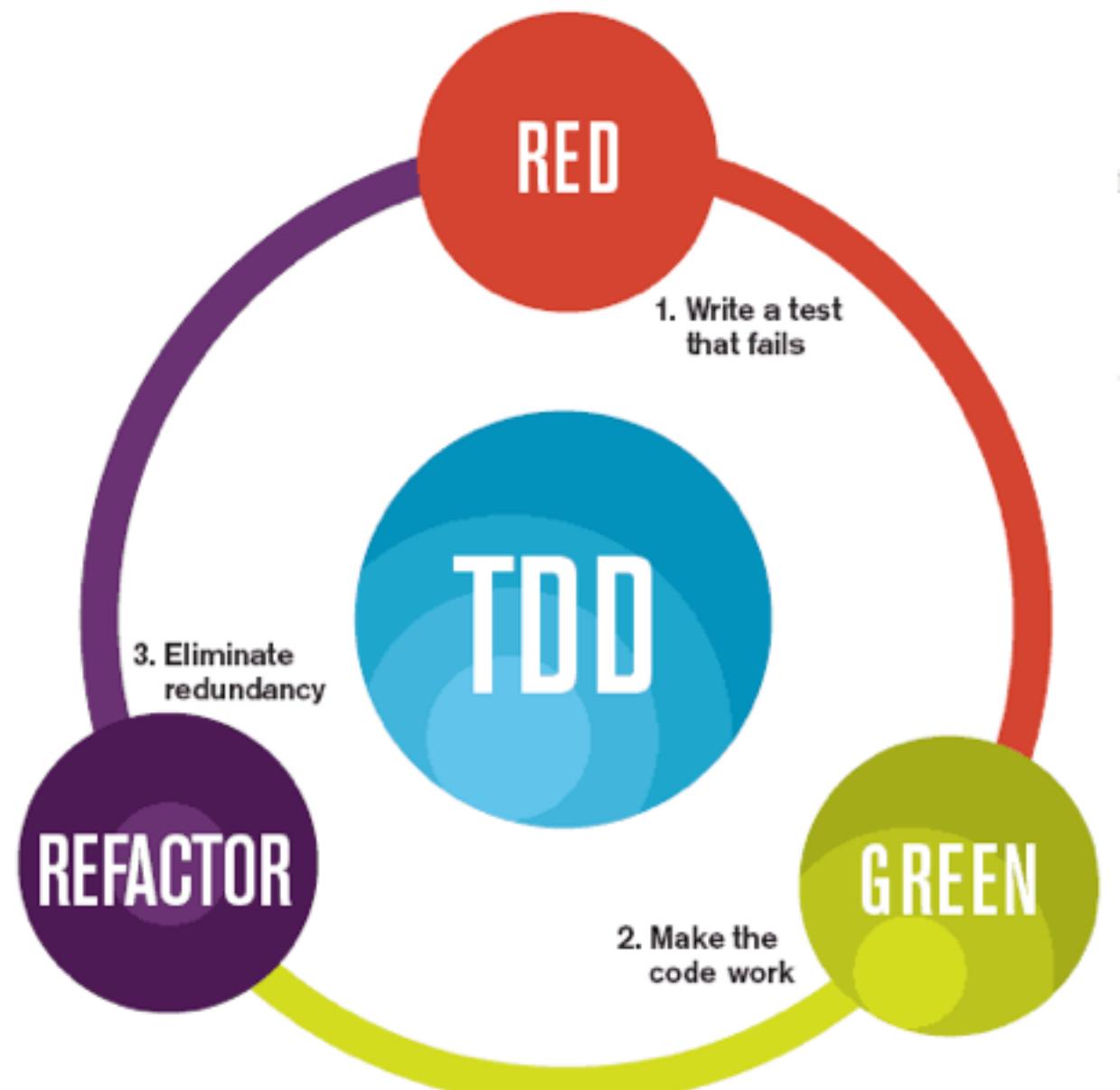
提交代码 -> Jenkins -> Ansible -> AWS Cloudformation -> 部署应用 -> 测试

# 基础设施即代码



# 质量内建体系

## TDD



## 代码审查



## 自动化测试

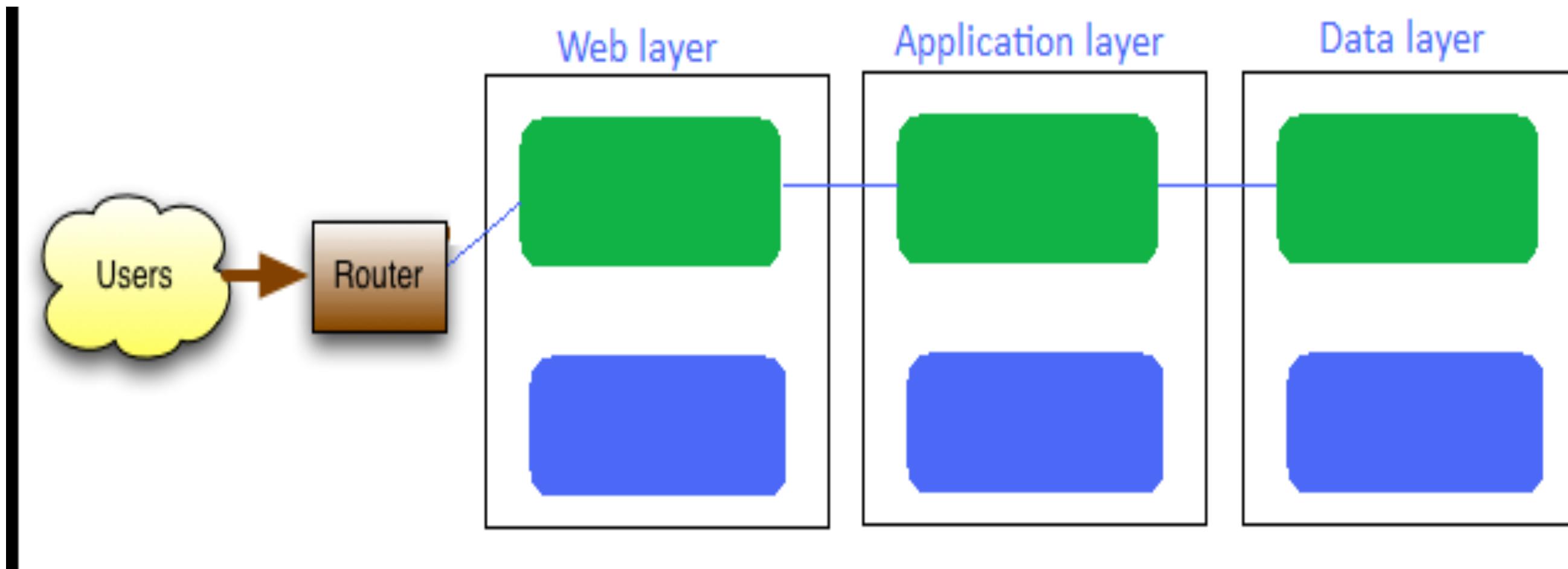
| Start | Duration | Action   |
|-------|----------|--|
| 00:00 | 5        | Starting Browser   |
| 00:05 | 0        | Setting timeout for asynchronous scripts: 10000  |
| 00:06 | 0        | Open URL: http://bs-local.com:45699/check  |
| 00:06 | 0        | Get page source<br><pre><html><head></head><body><pre style="word-wrap: break-word; white-space: pre-wrap;">Up and running</pre></body></html></pre> |
| 00:12 | 0        | STOP SESSION<br>CLIENT_STOPPED_SESSION   |

# 持续部署

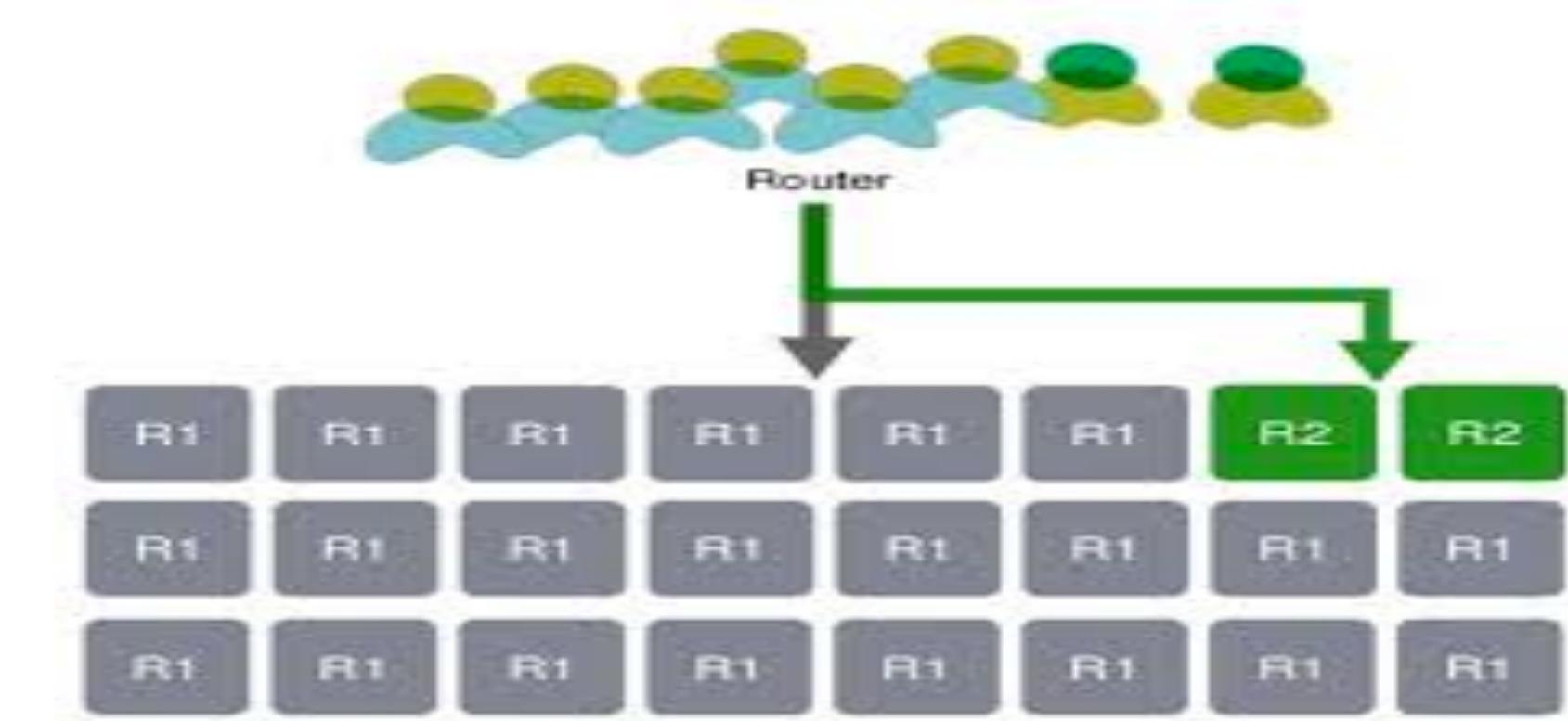
CI/CD 流水线



蓝绿部署



金丝雀发布

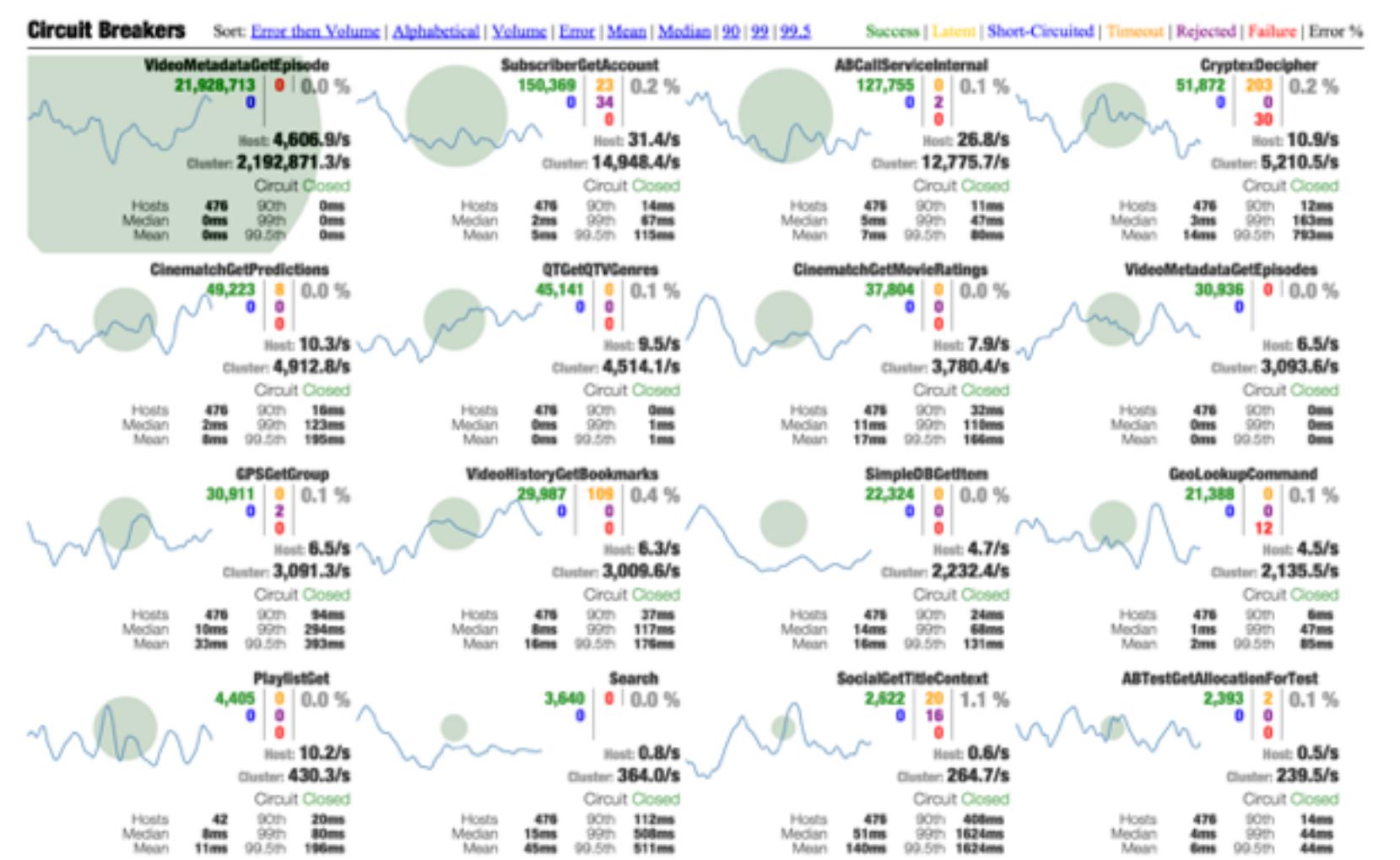


# 持续监控

# 监控、预警

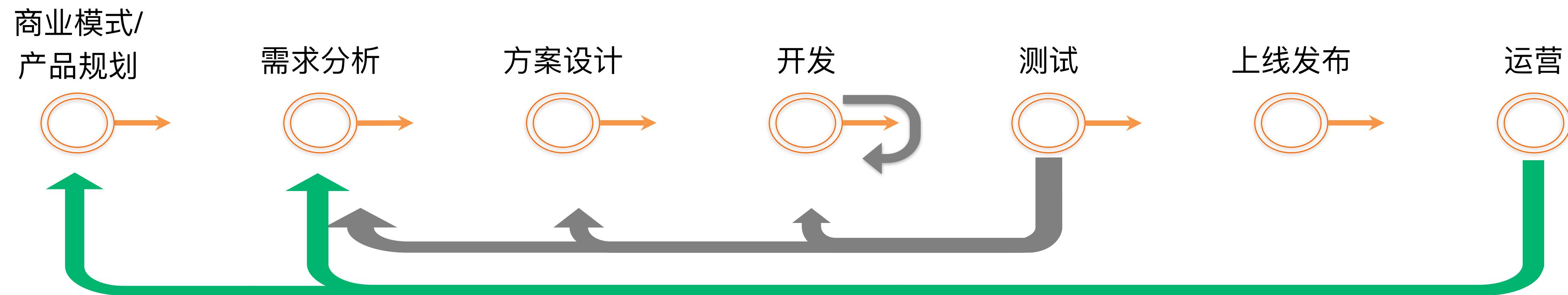
# 日志聚合

## 分析



# 度量与反馈

---

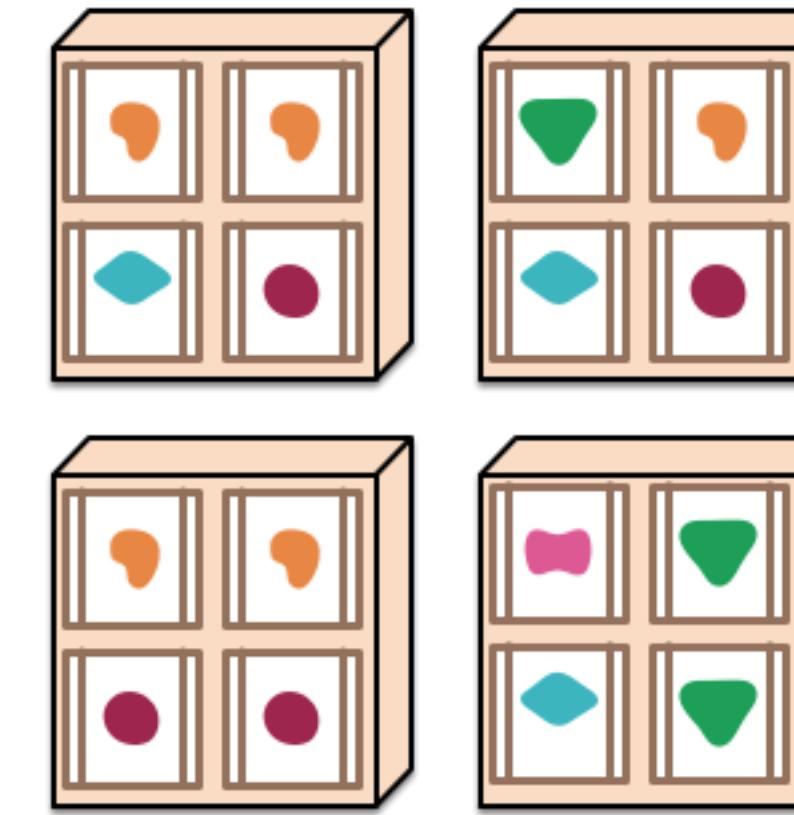


# 松耦合架构

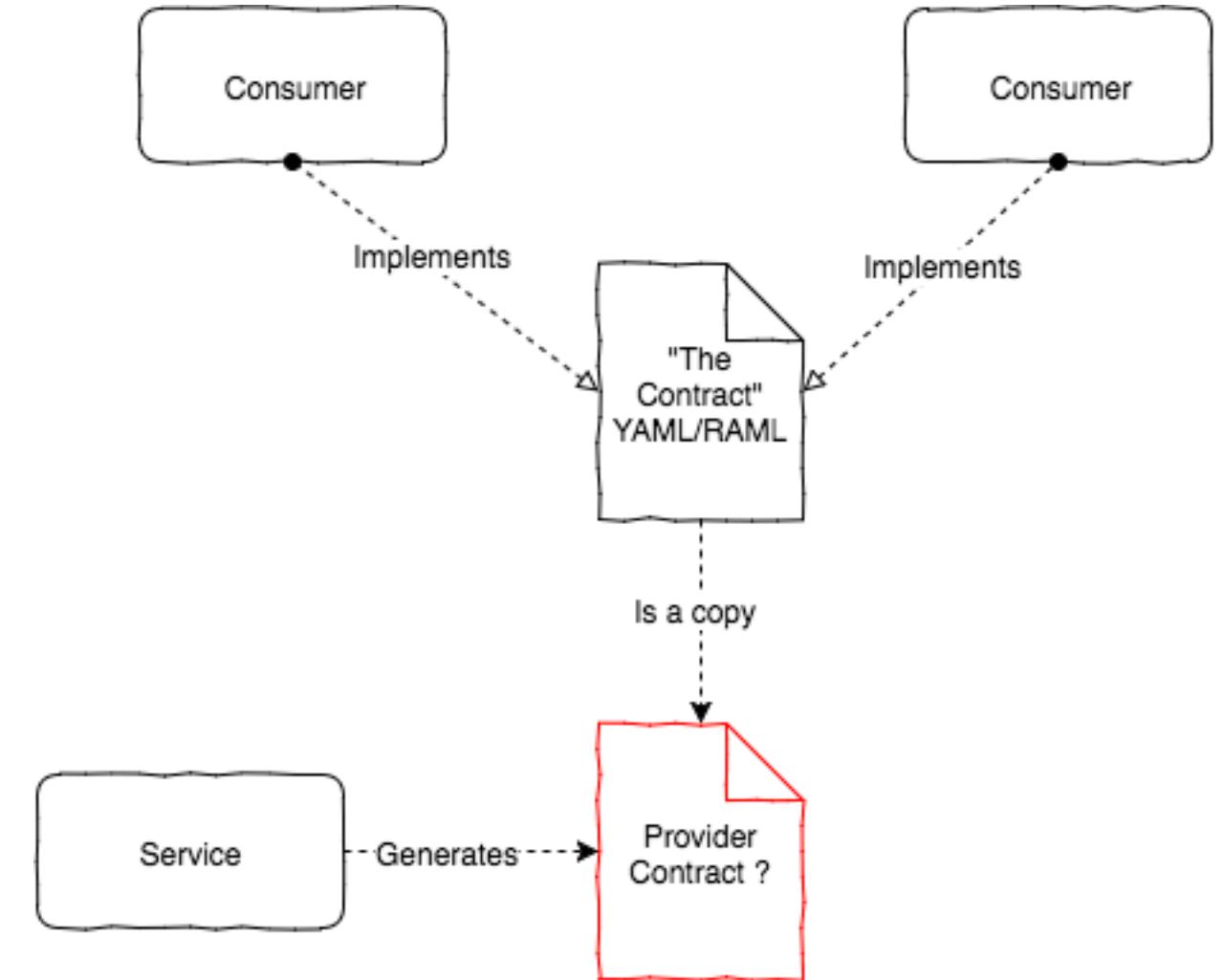
## 弹性基础设施



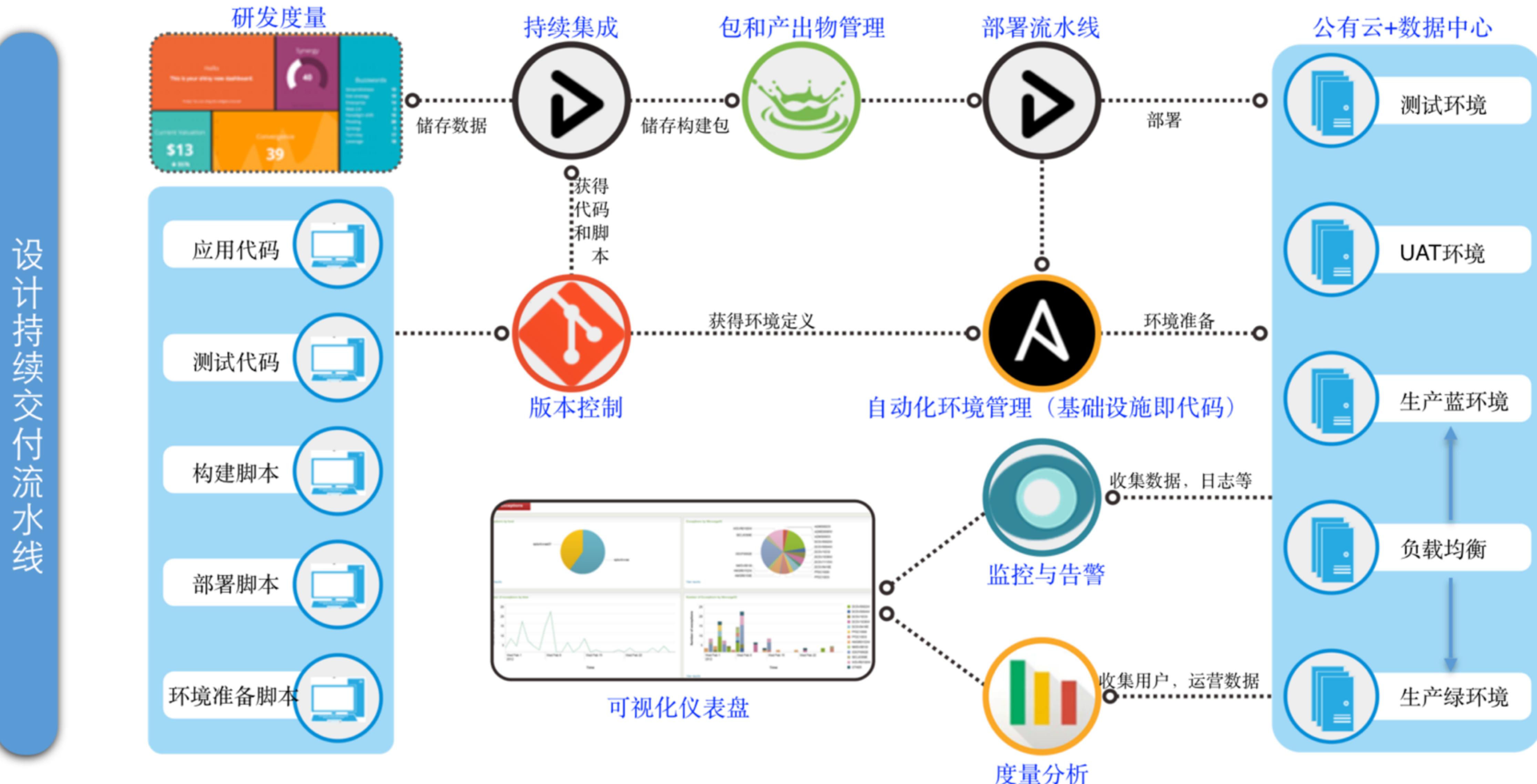
## 微服务



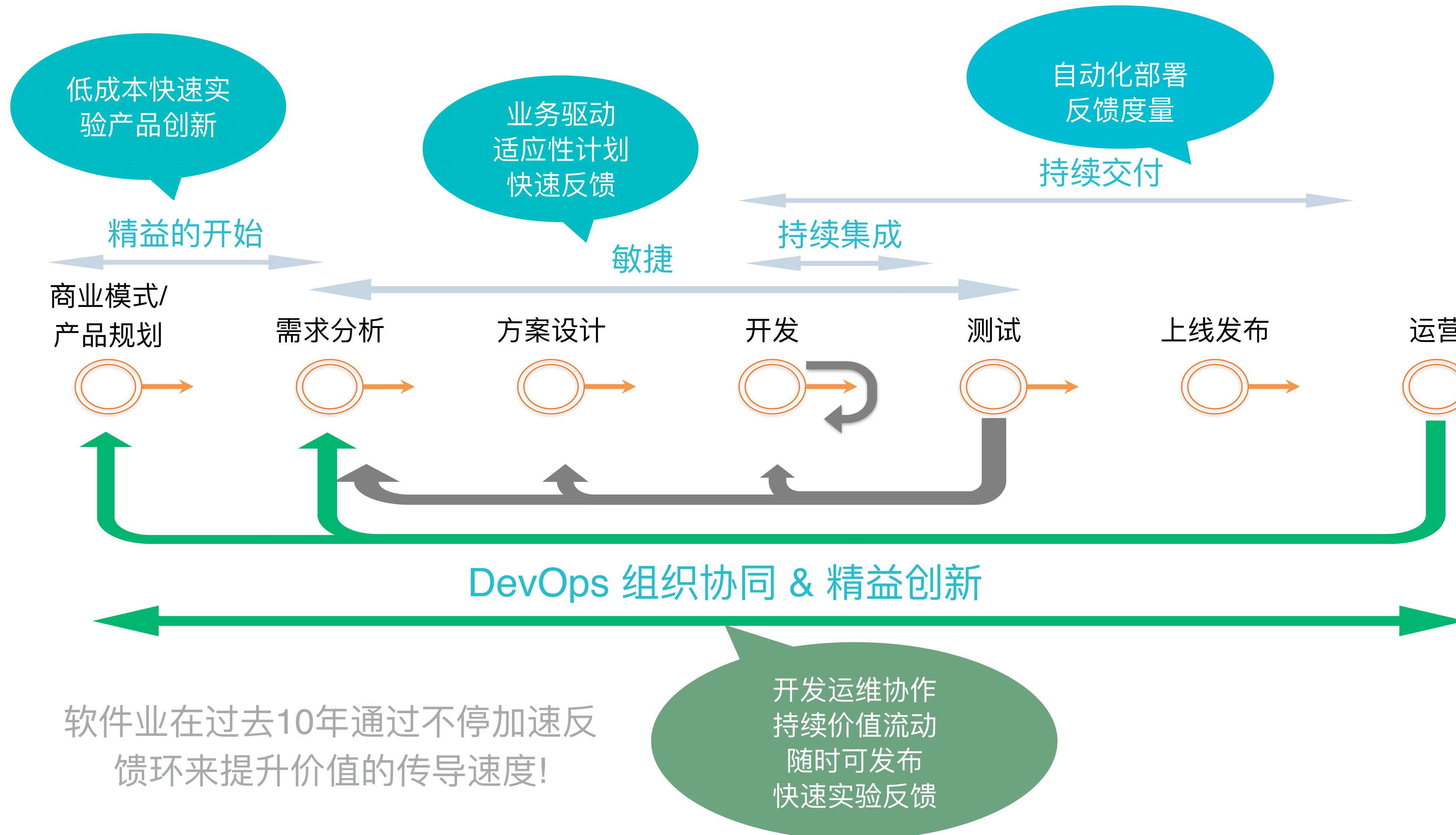
## API 治理



# 典型DEVOPS的持续交付流水线全景图

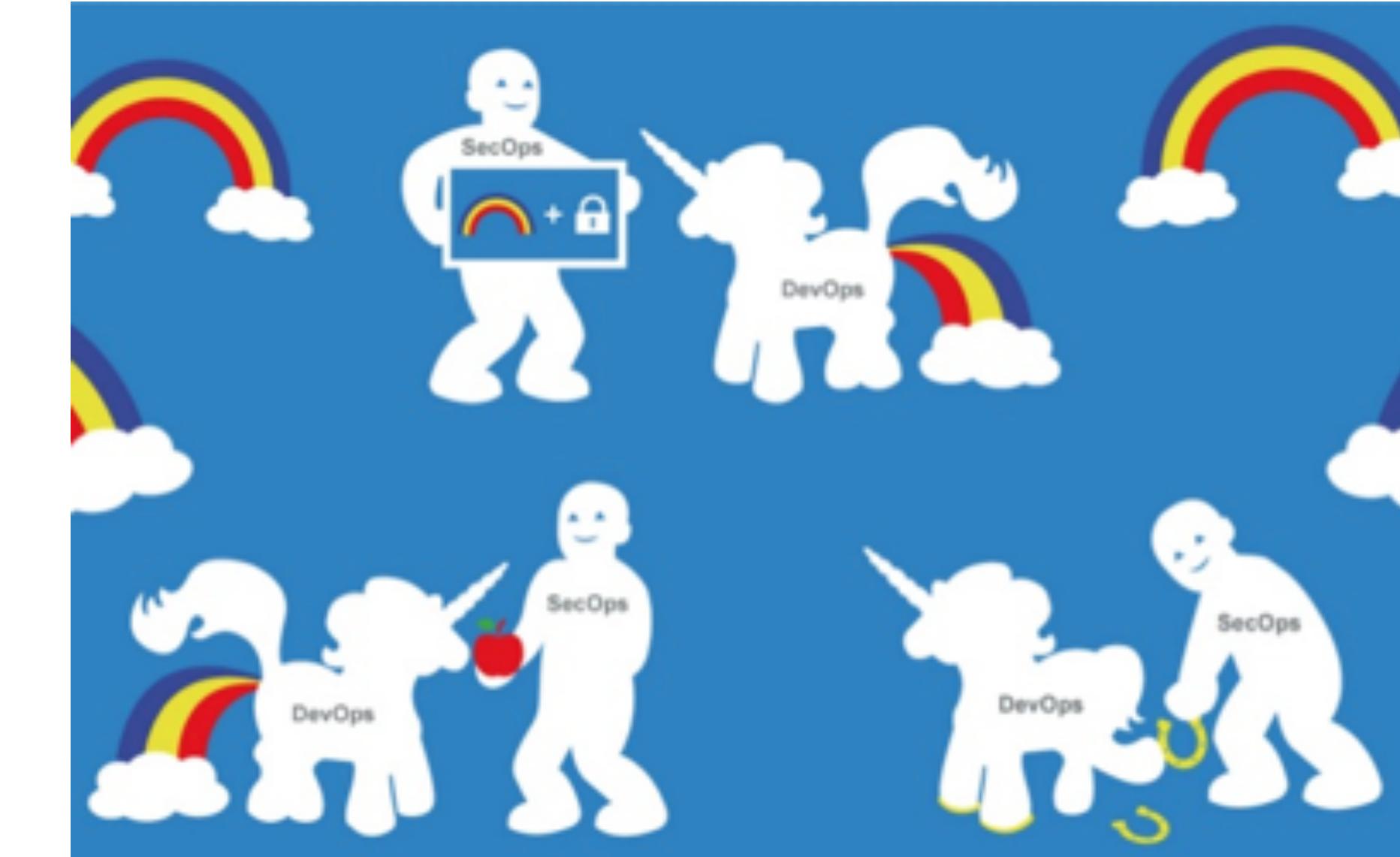


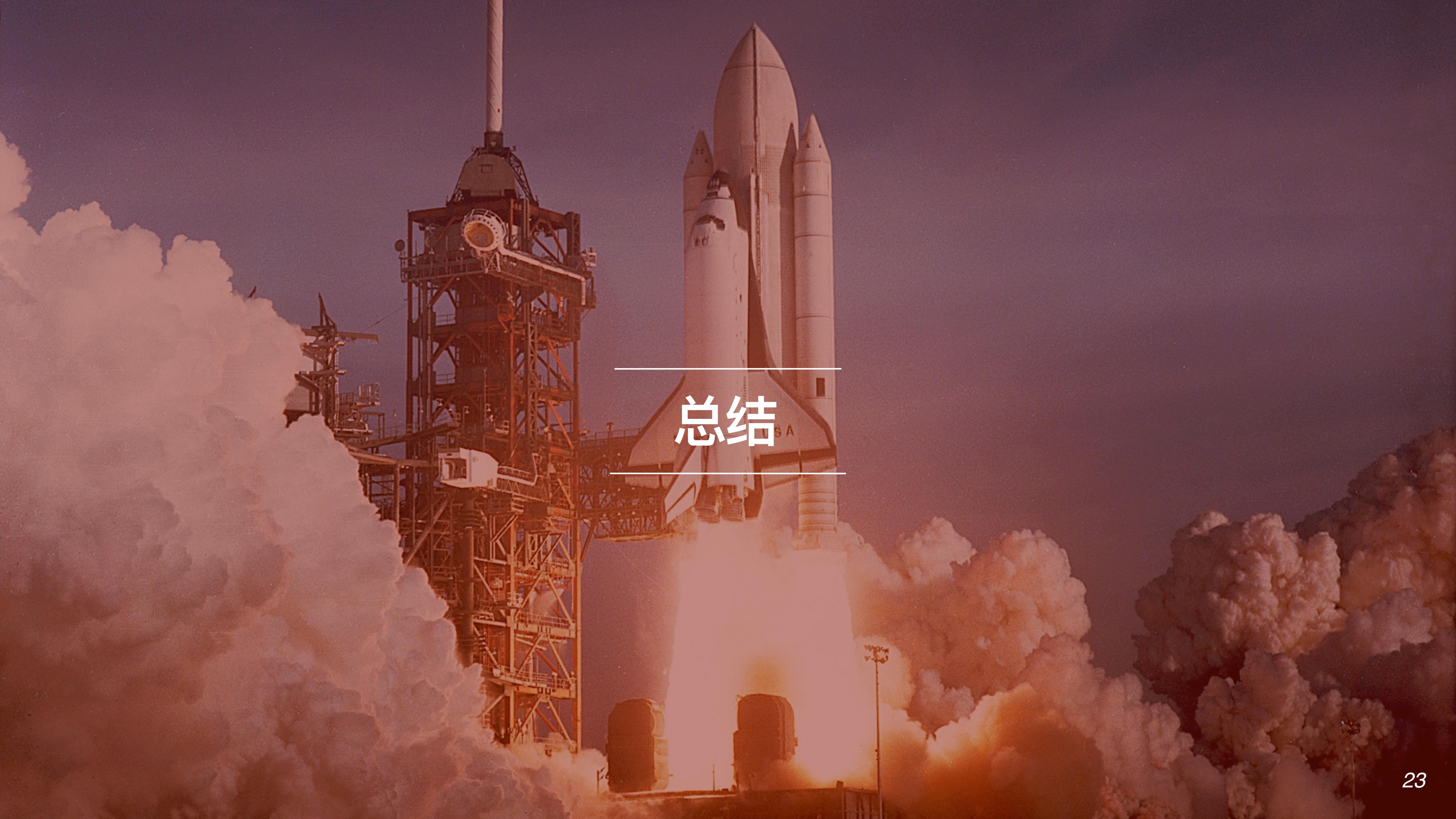
# 软件开发全生命周期的持续优化



# DEVOPS 的发展趋势

- *DevOps* 话语权越来越多被平台厂商掌握
- 容器化 & 微服务仍然是 *DevOps* 应用和发展的主要领域
- 安全成为推动 *DevOps* 全面发展的重要力量





总结

# 谢谢

联系方式：

[yiddu@thoughtworks.com](mailto:yiddu@thoughtworks.com)

*ThoughtWorks*技术雷达

*<https://www.thoughtworks.com/radar>*

ThoughtWorks®