

DEVOPS技术方案

目录

一

DevOps 概述

1. 面临的问题
2. 解决办法
3. DevOps 是什么
4. DevOps 价值

二

DevOps 平台介绍

三

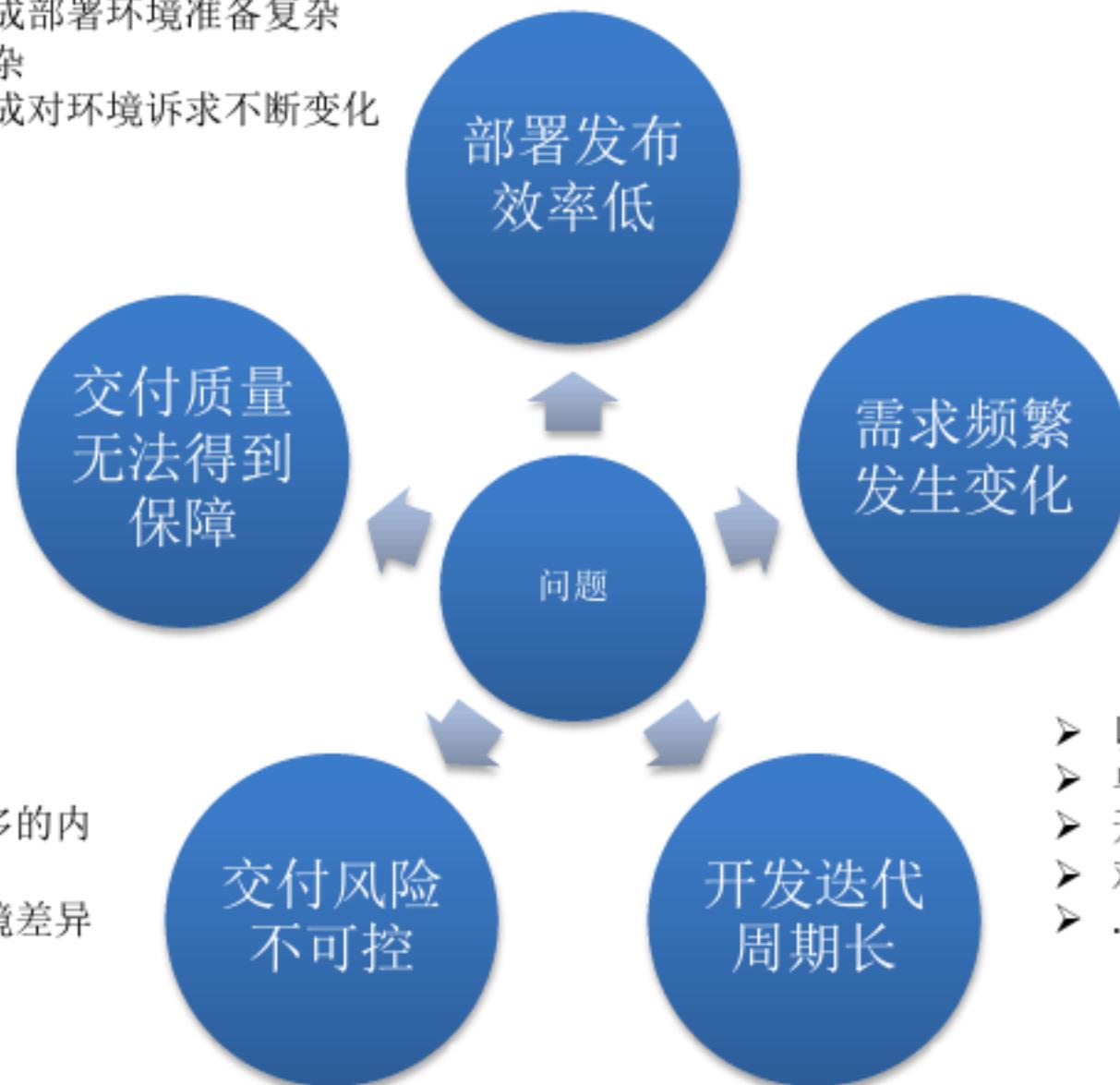
案例说明与操作演示

面临的问题

- 大量的手工操作易出错
- 语言异构造成部署环境准备复杂
- 部署架构复杂
- 迭代变化造成对环境诉求不断变化
- ...

- 开发人员提交的代码没有得到充分自测
- 测试人员手工操作工作量大，存在较大的资源浪费
- 开发测试对需求理解出现不一致
- ...

- 单次交付包含了较多的内容
- 上线环境和测试环境差异大
- ...



经验未得
到积累

缺少规范
与流程

缺少合适的
工具与平台

沟通协作不
畅

缺少自动化的

瀑布式的开发
模式

- 瀑布式的开发模式决定了对变化的适应能力较弱
- 开发难度加大，质量难以保证
- 测试难度加大，容易出现信息不对称的情况
- 成本上不可控制
- ...

- 以月和年的方式进行发布和更新
- 单次迭代包含大量的内容
- 开发进度前松后紧
- 对远期风险可控程度较低
- ...

解决办法

经验未得到积累

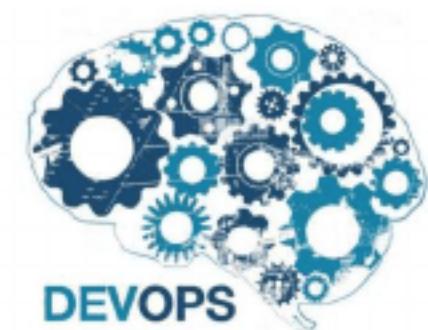
缺少规范与流程

缺少合适的工具
与平台

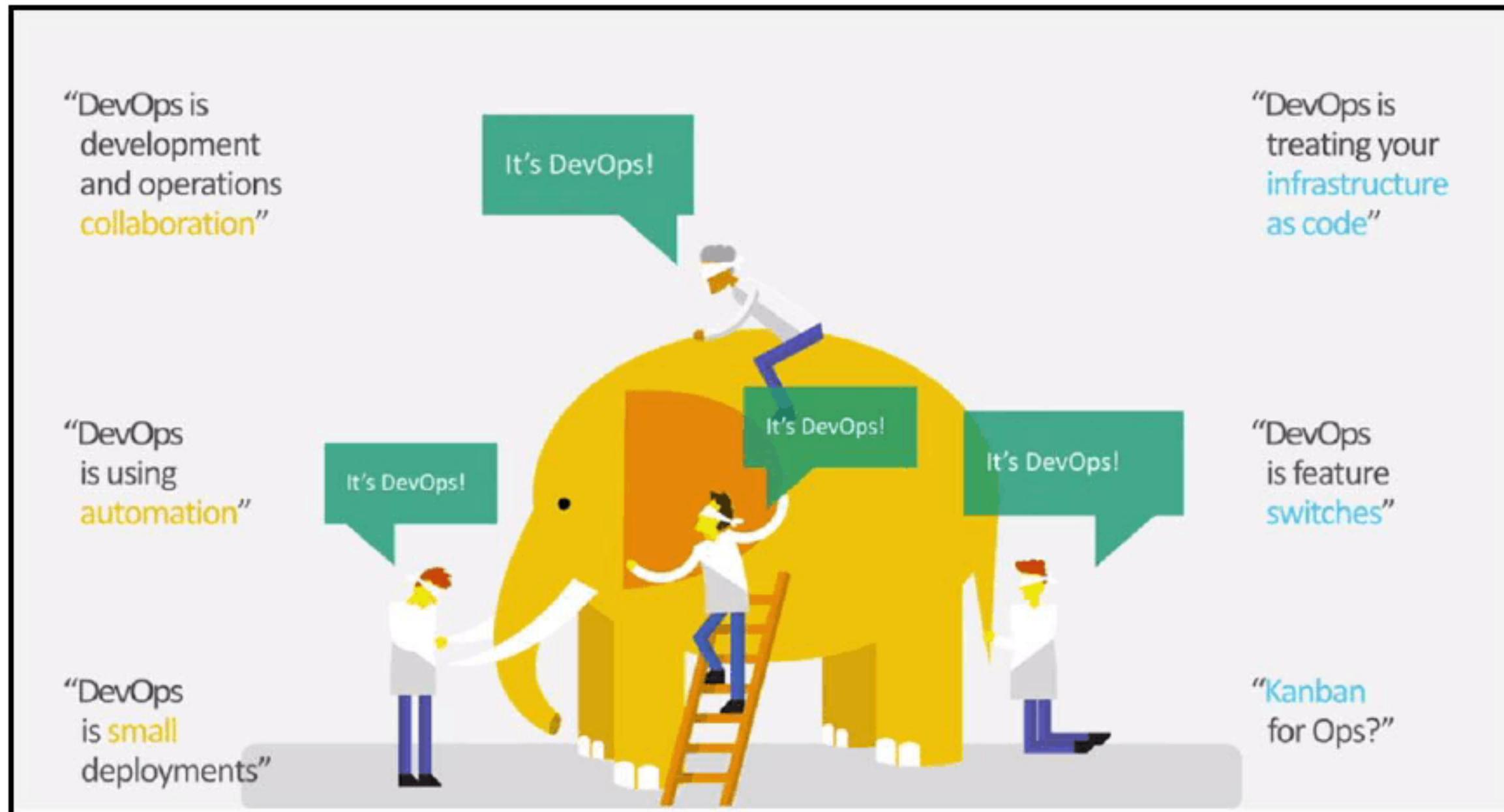
沟通协作不畅

缺少自动化

瀑布式的开发模
式



DevOps是什么



DevOps的定义

DevOps (a clipped compound of “development” and “operations”) is a software engineering **culture** and practice that aims at unifying software development(Dev) and software operation(Ops). The main characteristic of the DevOps movement is to strongly advocate **automation** and monitoring at all steps of software construction, from integration, testing, releasing to deployment and infrastructure management. DevOps aims at **shorter development cycles**, increased deployment frequency, more dependable releases, in close alignment with business objectives.

--- from Wikipedia

DevOps is a set of practices intended to **reduce the time** between committing a change to a system and the change being placed into normal production, while ensuring **high quality**.

--- from Bass, Weber, and Zhu

DevOps is the **union of people, process, and products** to enable continuous delivery of value to our end users. You cannot buy DevOps and install it. DevOps is not just automation or infrastructure as code. DevOps is people following a process enabled by products to **deliver value to end users**.

-- from Donovan Brown, Microsoft

DevOps Program Manager

DevOps核心要素

Culture (文化)

Automation(自动化)

Lean(精益)

Metrics(指标)

Share(分享)

DevOps具体是什么

是自动化

是持续集成

是“把基础设施看作代码”

是IT中一种不同的做事方式



Devops is application lifecycle management with the goal of continuous delivery achieved through the discovery, refinement and optimization of repeatable processes.

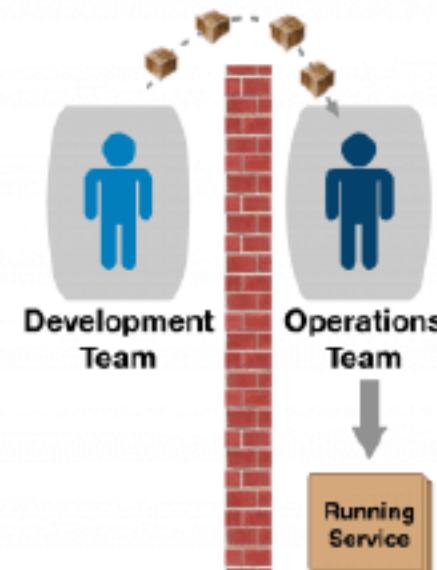
DevOps价值

不同的人

不同的职责

不同的技能

“You build it. They run it.”



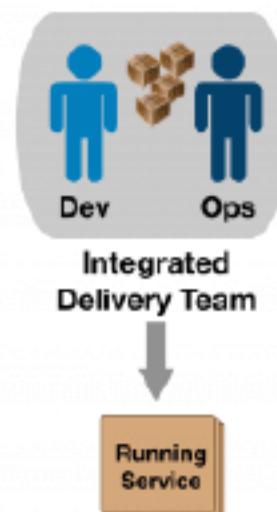
Pro:

- Scales well with project-based funding
- Supports lots of legacy systems

Con:

- Handoff problems (errors and delay)
- Dev still interrupted by escalations

“You build it. You run it.”



Pro:

- No handoffs or breaks in context
- Easier to achieve high velocity

Con:

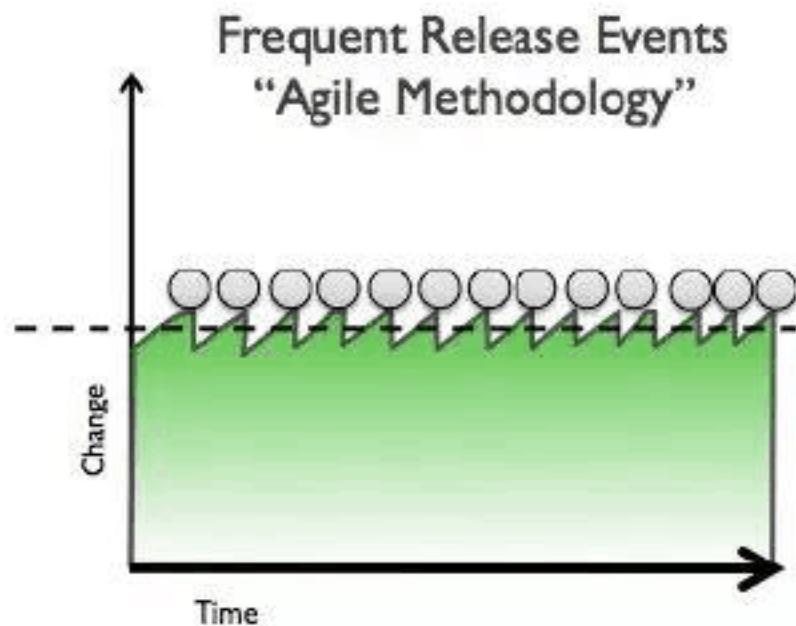
- Conflicts traditional structure/funding model
- Scale issues with large amounts of legacy

基础设施即代码

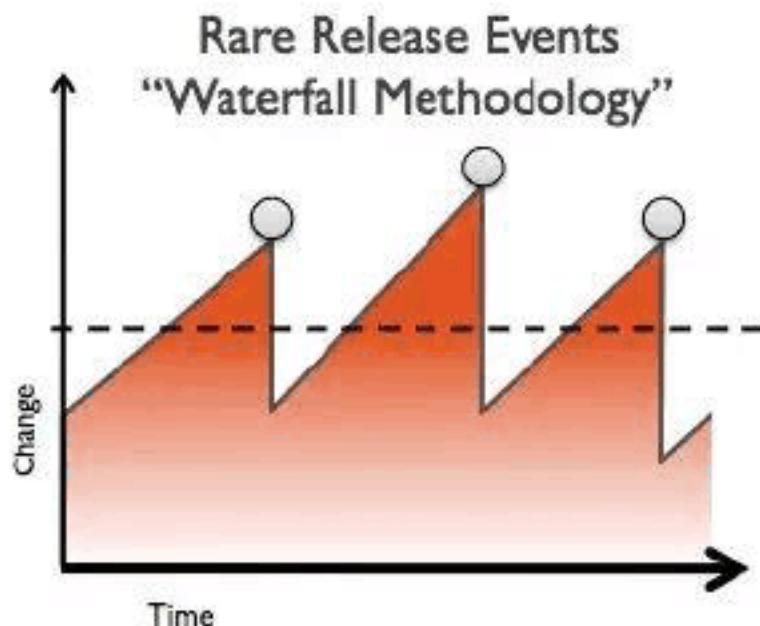
自动化无处不在

DevOps工程师

DevOps价值



**Smoothen Effort
Less Risk**



**Effort Peaks
High Risk**



目录

一

DevOps 概述

二

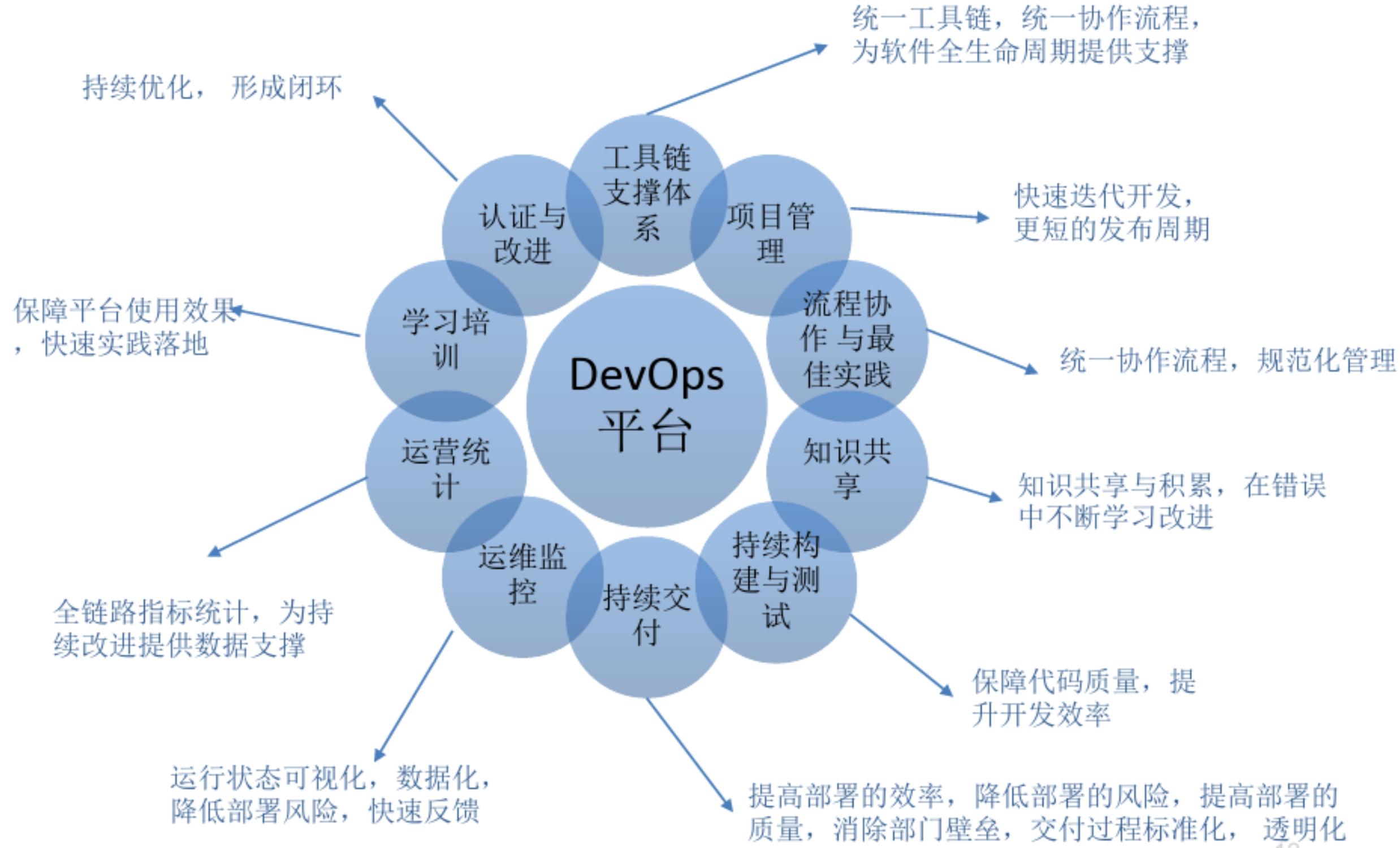
DevOps 平台介绍

1. DevOps平台的位置
2. 产品介绍
3. 支撑的场景

三

案例说明与操作演示

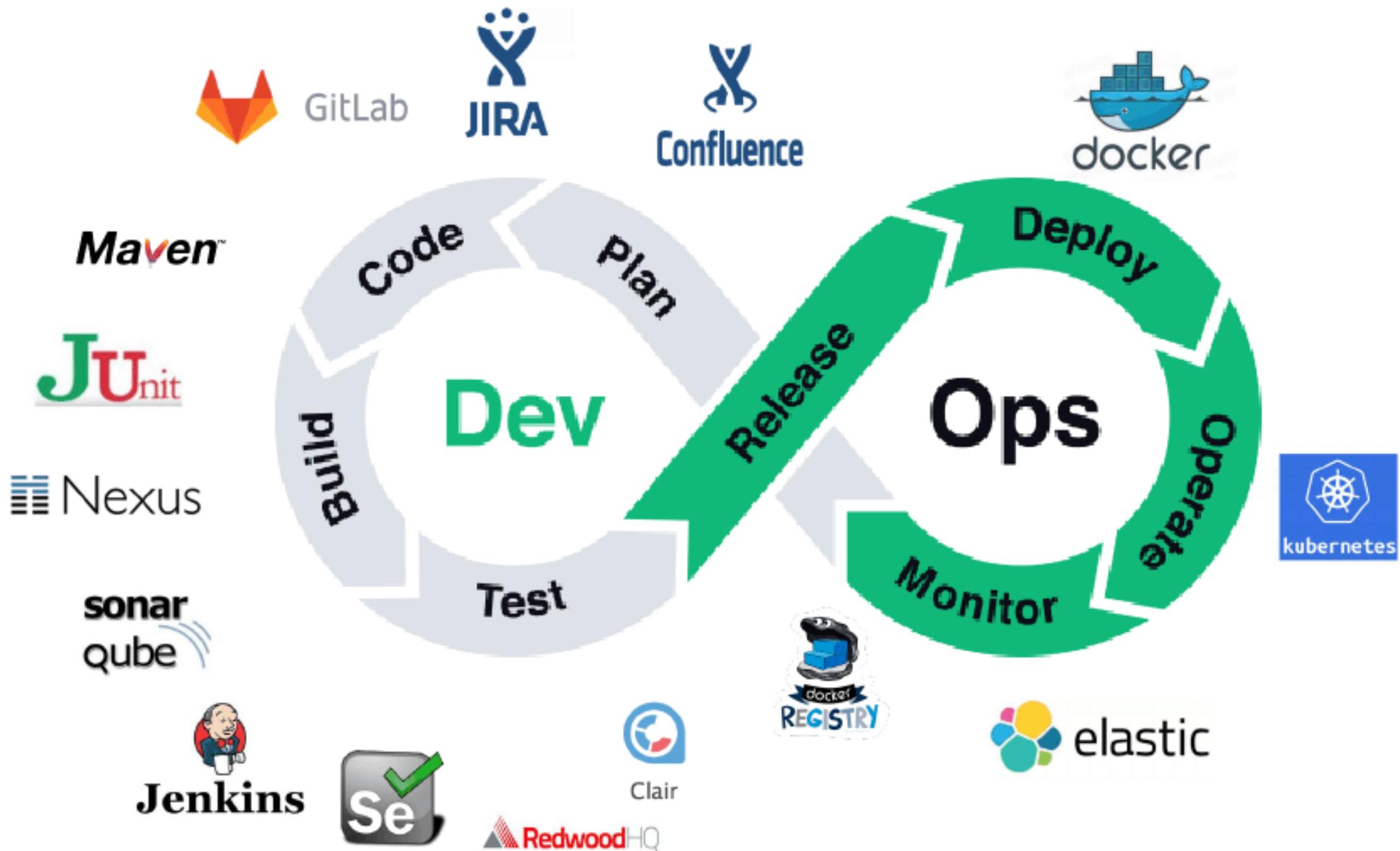
DevOps体系链



DevOps产品介绍 - 全景



DevOps产品介绍 - 工具

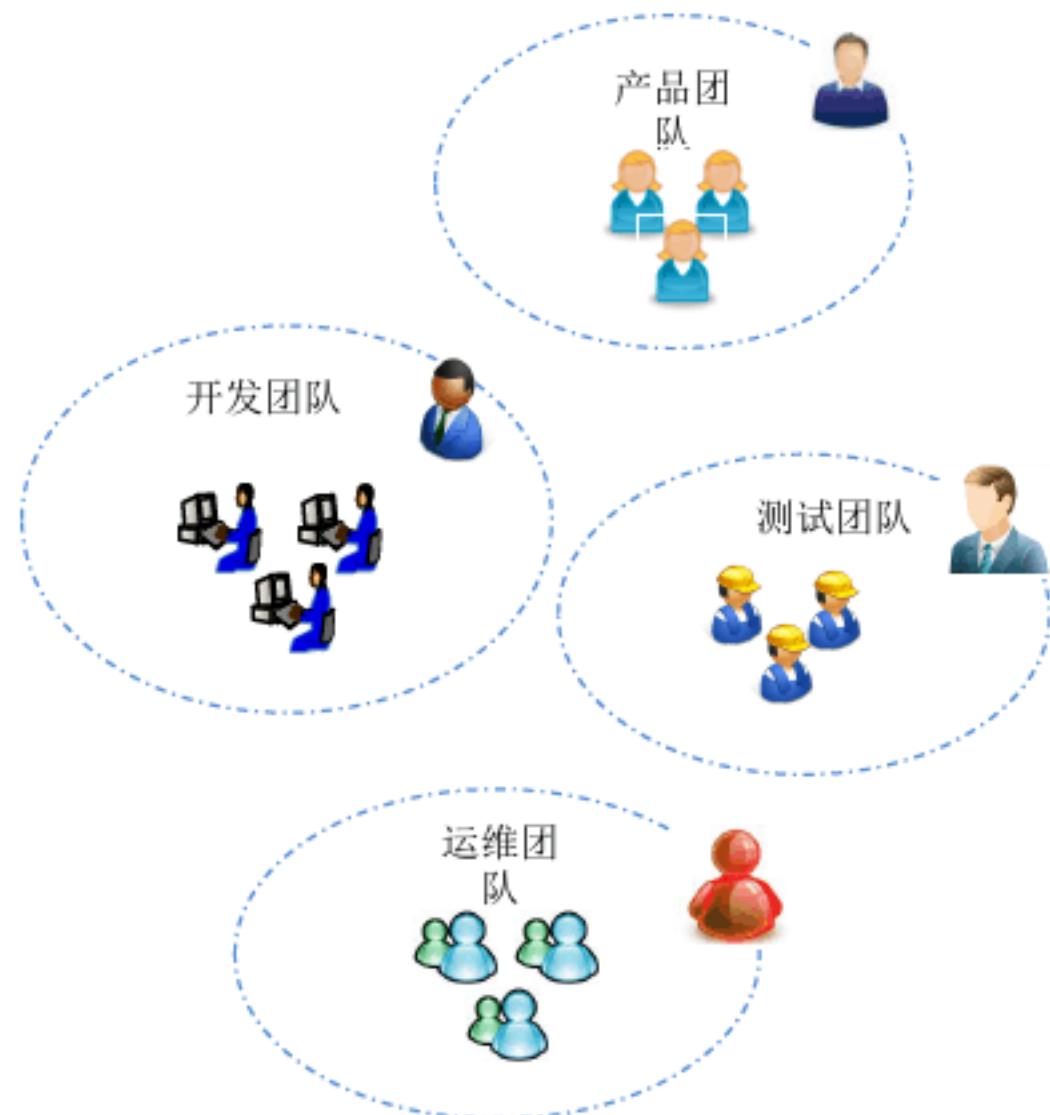


支撑的场景

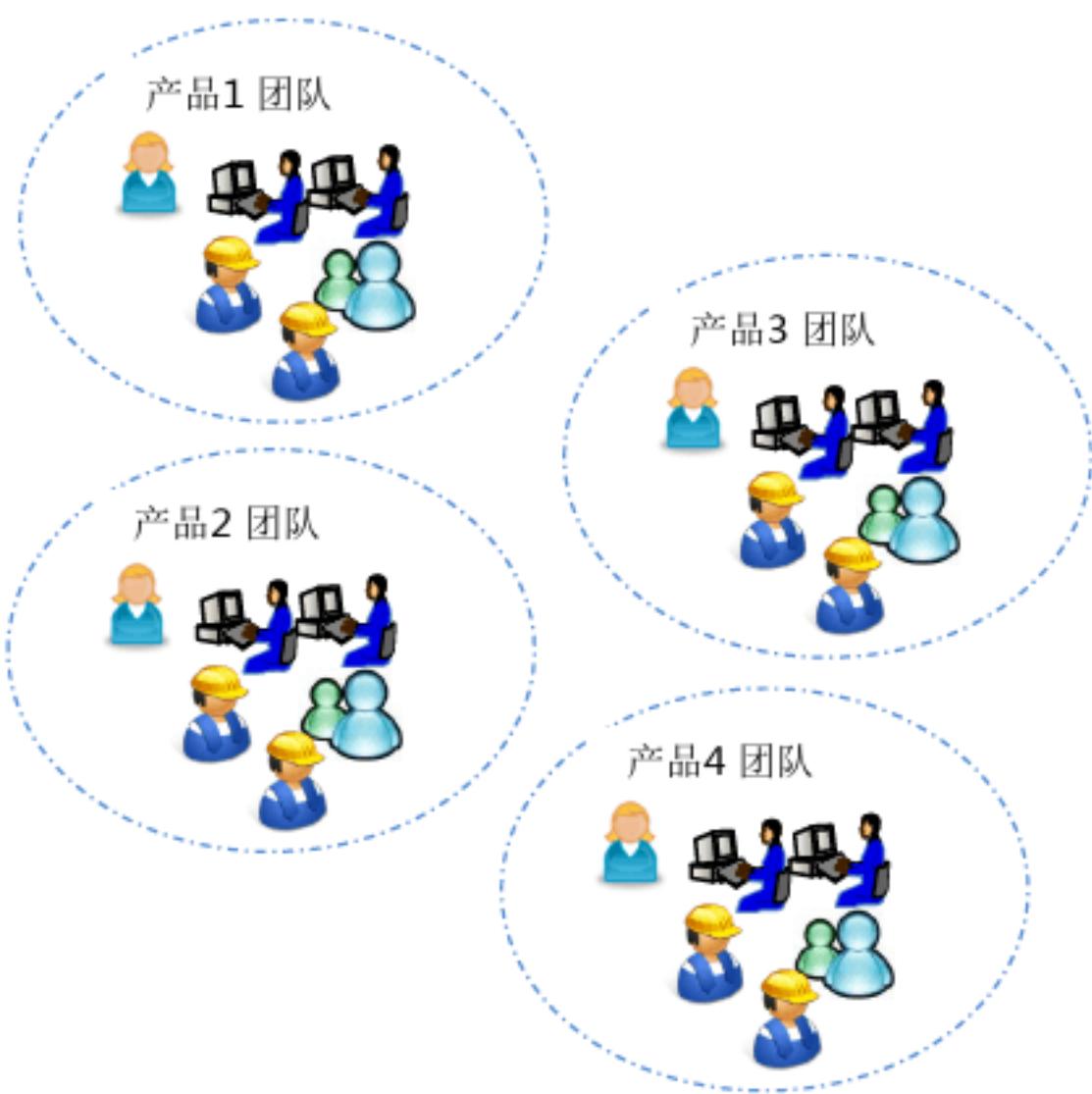


场景下的团队构成

传统场景下



敏态场景下



团队角色职责说明



产品经理

- 产品规划，需求排期
- 推动目标实现



产品主管

- 对整体产品进行规划，需求排期
- 对产品团队负责



开发人员

- 需求理解
- 编码实现
- 产出交付



开发主管

- 负责开发任务的分解，任务下发
- 开发团队管理



测试人员

- 功能接口性能测试
- 测试用例编写
- 产品质量保证



测试主管

- 功能接口性能测试
- 产品质量保证
- 测试团队管理



运维人员

- 功能接口性能测试
- 测试用例编写
- 产品质量保证



运维主管

- 团队管理
- 为系统应用稳定性负责



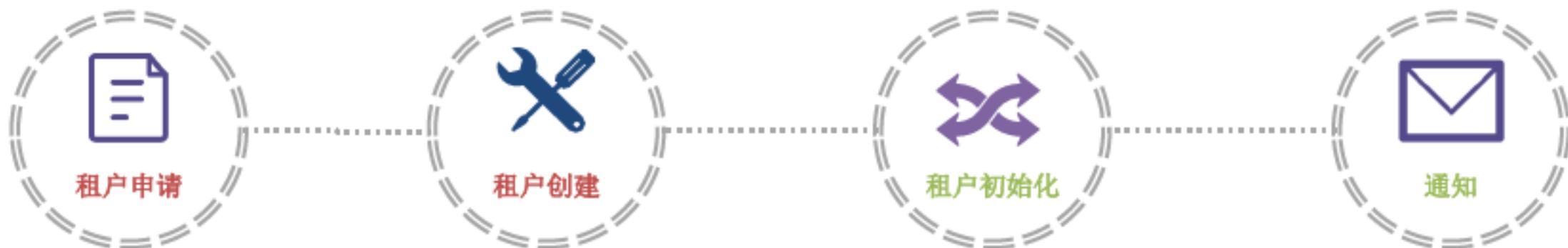
用户

- 产品普通用户

前置条件

租户接入

- 接入时，需要先进行租户申请
- 申请后平台进行租户的创建和初始
- 邮件通知关系人，租户创建完成



需求管理



- ✓ 需求管理工具量化跟踪
- ✓ 需求分级处理
- ✓ 统一需求管理流程

- ✓ 需求面板跟踪
- ✓ 需求分级管理
- ✓ 责任到人

- ✓ 处理状态跟踪
- ✓ 知识管理工具对知识协同共享

- ✓ 需求处理过程全链路追溯
- ✓ 及时通知



- ✗ 产品进度难以跟踪
- ✗ 需求管理杂乱
- ✗ 无优先级定义
- ✗ 需求响应周期长

开发测试

- ✓ 测试case管理工具规范管理
- ✓ 自动化测试管理工具对测试case进行图形化编排降低自动化测试编写难度



- ✓ 开发任务关联需求
- ✓ 事务管理工具对任务进行细粒度拆解
- ✓ 设置合理的任务大小，跟踪开发状态



- ✓ IDE 与 DevOps 工具紧密集成
- ✓ 代码变更管理任务
- ✓ 本地代码扫描保证质量
- ✓ 推荐 Git 分支管理模型



- ✓ 代码提交触发流水线
- ✓ 流水线自动进行单元测试
- ✓ 流水线自动进行编译打包
- ✓ 流水线自动生成镜像
- ✓ 流水线自动部署更新服务



- ✓ 事务管理工具跟踪状态
- ✓ 跟踪团队开发进度

开发测试



- ✗ 编码质量难控制
- ✗ 发布时需要手动打包编译
- ✗ 代码版本管理复杂
- ✗ 需求与开发任务无对应关系
- ✗ 与测试人员交接难



- ✗ 开发提交质量低
- ✗ 测试**case**管理杂乱
- ✗ 测试介入晚
- ✗ 需求**kickoff** 不明确
- ✗ 测试环境与生产环境不一致

发版提测



创建发版工单



测试环境部署



测试



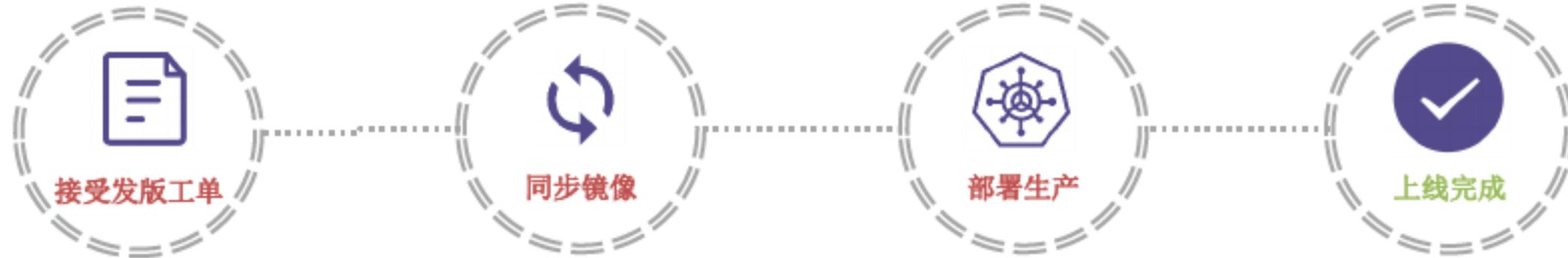
版本Signoff

- | | | | |
|--------------|-----------------|----------------------------|-------------------|
| ✓ 交付内容标准化规范化 | ✓ 平台容器化快速部署 | ✓ 使用自动化测试工具快速校验交付质量，加速测试效率 | ✓ 版本发布面板，跟踪版本发布进度 |
| ✓ 交付内容版本可追踪 | ✓ 保证测试环境和开发环境一致 | ✓ 按照编写好的测试case进行手工测试 | |
| ✓ 交付内容关联需求 | | | |
| ✓ 交付内容关联开发任务 | | | |
| ✓ 交付内容管理变更范围 | | | |



- ✗ 测试环境与开发环境不一致
- ✗ 测试效率低
- ✗ 测试环境准备复杂

上线运维



- | | | | | | | | | | | | | |
|--------------|-------------|------------|--------------|--------------|---------------|-------------|----------|------------|-----------|---------|--------------------|-------------------|
| ✓ 交付内容标准化规范化 | ✓ 交付内容版本可追踪 | ✓ 交付内容关联需求 | ✓ 交付内容关联开发任务 | ✓ 交付内容管理变更范围 | ✓ 一键同步镜像到生产环境 | ✓ 容器云平台一键部署 | ✓ 丰富部署策略 | ✓ 内置丰富监控指标 | ✓ 自动化监控警报 | ✓ 智能扩缩容 | ✓ 容器云平台屏蔽部署架构的复杂性。 | ✓ 版本发布面板，跟踪版本发布进度 |
|--------------|-------------|------------|--------------|--------------|---------------|-------------|----------|------------|-----------|---------|--------------------|-------------------|



- ✗ 测试与正式环境差异大
- ✗ 环境搭建复杂
- ✗ 系统运行监控困难
- ✗ 部署架构复杂
- ✗ 自动化手段少

能力提升



过程中能力提升

通过云平台进行需求到上线运维的管理，打通了需求与开发，开发与测试，测试与运维之间的壁垒，提供了大量业界经过大量验证的规则规范，增加了大量自动化工具协助开发人员，测试人员与运维人员完成相应工作，极大提升了工作效率。

- ✓ 加快开发迭代速度
- ✓ 增加交付频率
- ✓ 降低交付风险
- ✓ 识别过程资源浪费
- ✓ 加速需求响应时间
- ✓ 团队效率的整体提升

技术规范与操作手册

- | | |
|---|--|
|  Git 分支管理规范 |  平台使用手册 |
|  版本发布规范 |  JIRA工作流配置手册 |
|  安全开发规范 |  JIRA敏捷管理培训手册 |
|  产品立项模板规范 |  JIRA报表和仪表盘使用手册 |
|  配置管理规范 |  Confluence基本使用手册 |
|  缺陷管理规范 |  Confluence高级使用手册 |
|  软件质量标准规范 |  IDE集成SonarQube配置手册 |
|  数据库设计规范 |  IDE集成JIRA 配置手册 |
|  Java开发规范 | |
|  SQL开发准则框架规范 | |

目录

一

DevOps 概述

二

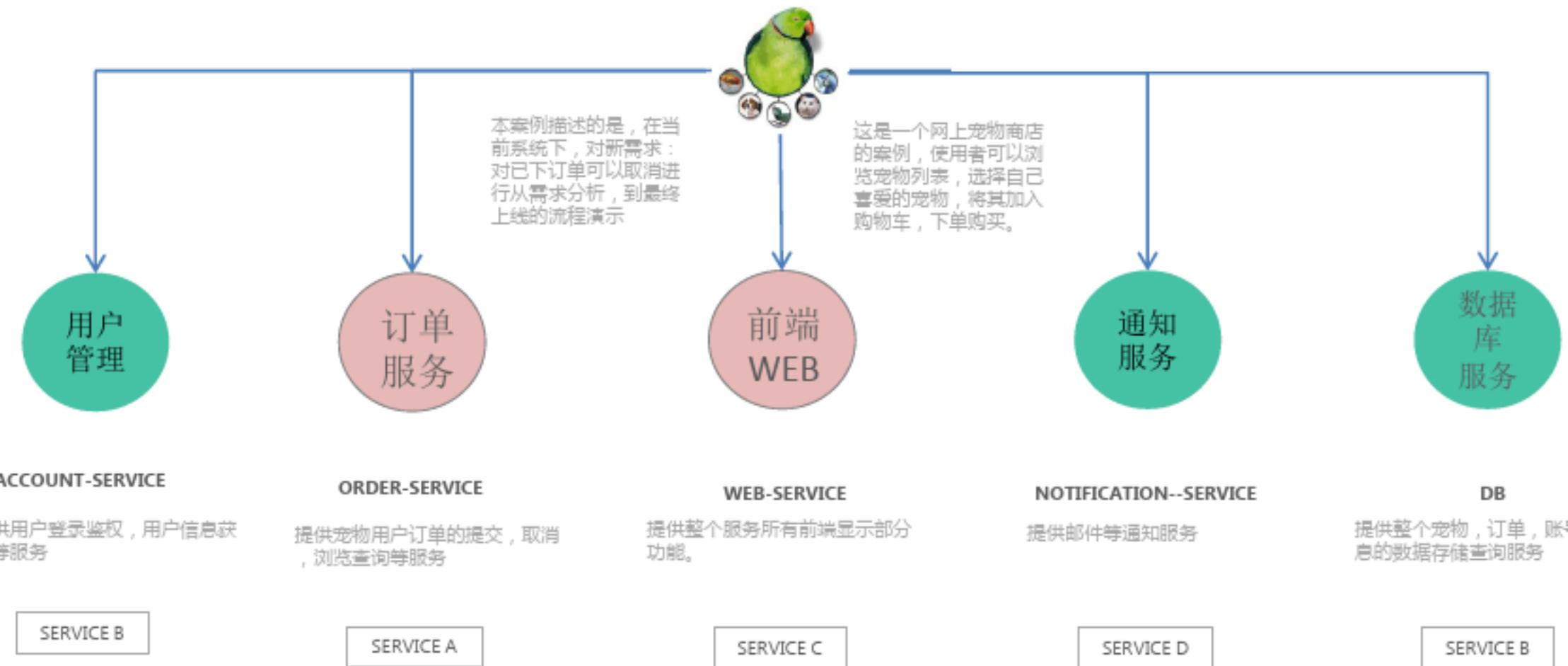
DevOps 平台介绍

三

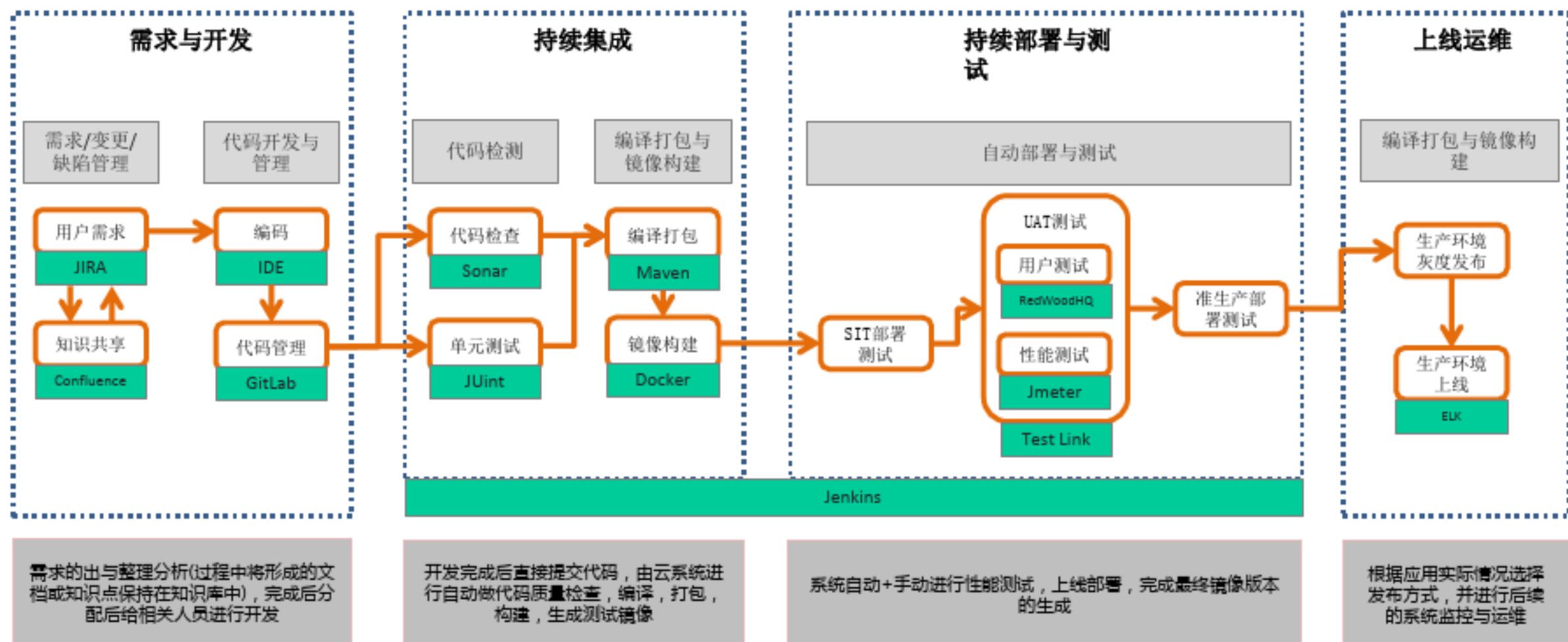
案例说明与操作演示

1. 案例说明
2. 工程介绍
3. 操作演示

案例说明



工程介绍 – 项目全景



工程介绍 - 环境介绍



工程介绍 – 项目管理工具



项目与事务跟踪工具，被广泛应用于缺陷跟踪、客户服务、需求收集、流程审批、任务跟踪、项目跟踪和敏捷管理等工作领域。



知识管理与协同软件，也可以用于构建企业wiki。使用简单，但它强大的编辑和站点管理特征能够帮助团队成员之间共享信息、文档协作、集体讨论，信息推送。

问题

- 需求，任务，缺陷等可自定义
- 丰富的表单字段，可以自定义

工作流

- 需求管理流程，缺陷管理流程，任务管理流程，可以自定义
- 关联表单

面板

- 丰富的数据报表和图表，可自定义
- 分享订阅他人面板

空间

- 单个组织和项目在一个空间下进行知识的管理共享

导航

- 知识的结构化管理
- 内置推荐的目录结构

页面

- 页面在线协同编辑，丰富的内容格式
- 页面可查看，历史版本，评论，分享

工程介绍 - 开发工具



用于仓库管理系统的开源项目，使用Git作为代码管理工具，并在此基础上搭建起来的web服务。



是一个开源的代码质量管理平台，致力于对代码质量的自动化分析和管理。

代码扫描

- 支持多种语言
- 检查代码存在的问题，技术债务，代码复杂度等
- 量化展示扫描结果

IDE集成

- 在IDE中直接使用Sonar进行代码扫描

自动化流水线

- 依靠丰富的插件完成多种语言的持续集成，持续部署



是基于Java开发的一种持续集成工具，用于监控持续重复的工作，旨在提供一个开放易用的软件平台，使软件的持续集成变成可能。

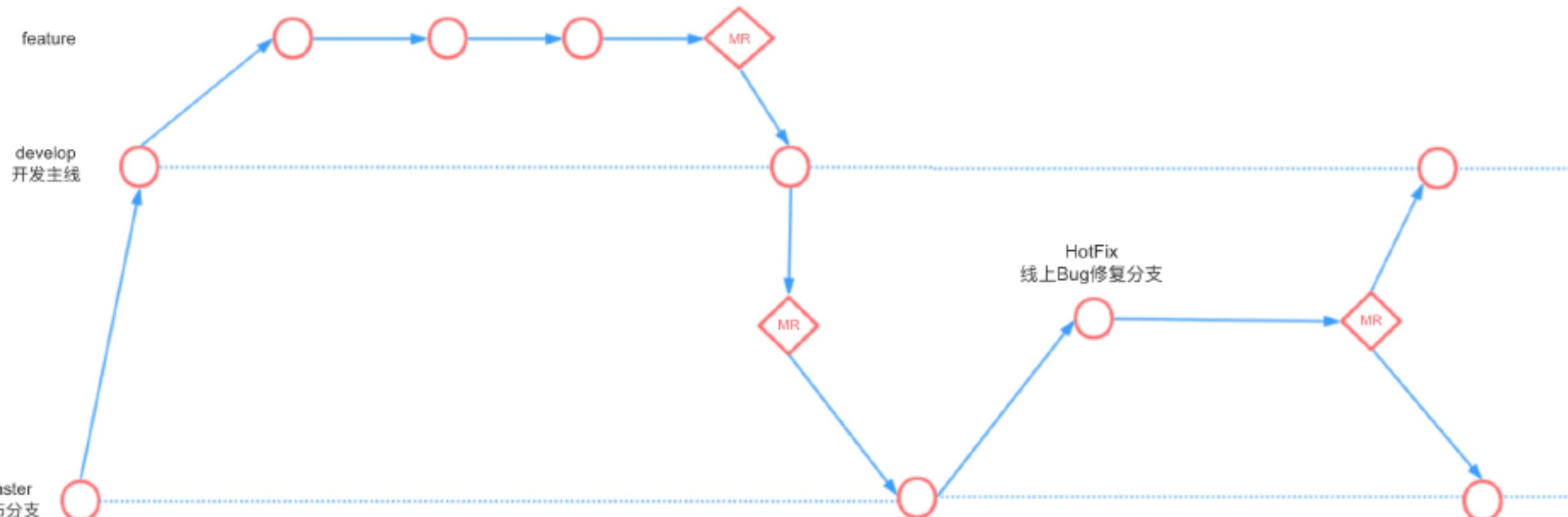
分布式支持

- 支持多节点构建编译

工程介绍 - 开发工具

Git 分支策略

分支名	作用	是否受保护
Feature	功能开发分支	✗
Develop	功能迭代中心分支	✓ 不允许直接提交 不允许强制推送
Master	功能发布分支	✓ 不允许直接提交 不允许强制推送



工程介绍 – 流水线



工程介绍 - 测试工具



开源自动化框架，支持主流ui自动化 selenium等，支持REST API自动化。允许多个用户同时工作，同时提供web ui去创建，执行自动化脚本和自动化case的管理。



Web应用程序测试的工具，测试直接运行在浏览器中，就像真正的用户在操作一样。



基于web的测试用例管理系统，主要功能是测试用例的创建、管理和执行，还提供了一些统计功能。

图形化编排
测试用例

- 通过图形化的方式编排自动化测试用例，普通用户无需写脚本

支持广泛

- 支持功能测试，接口测试，Ui测试

模拟真实场景

- 直接在浏览器中运行，模拟终端用户的使用场景

兼容性测试

- 支持多种浏览器，进行兼容性测试

测试用例

- 测试用例的编写，管理，执行

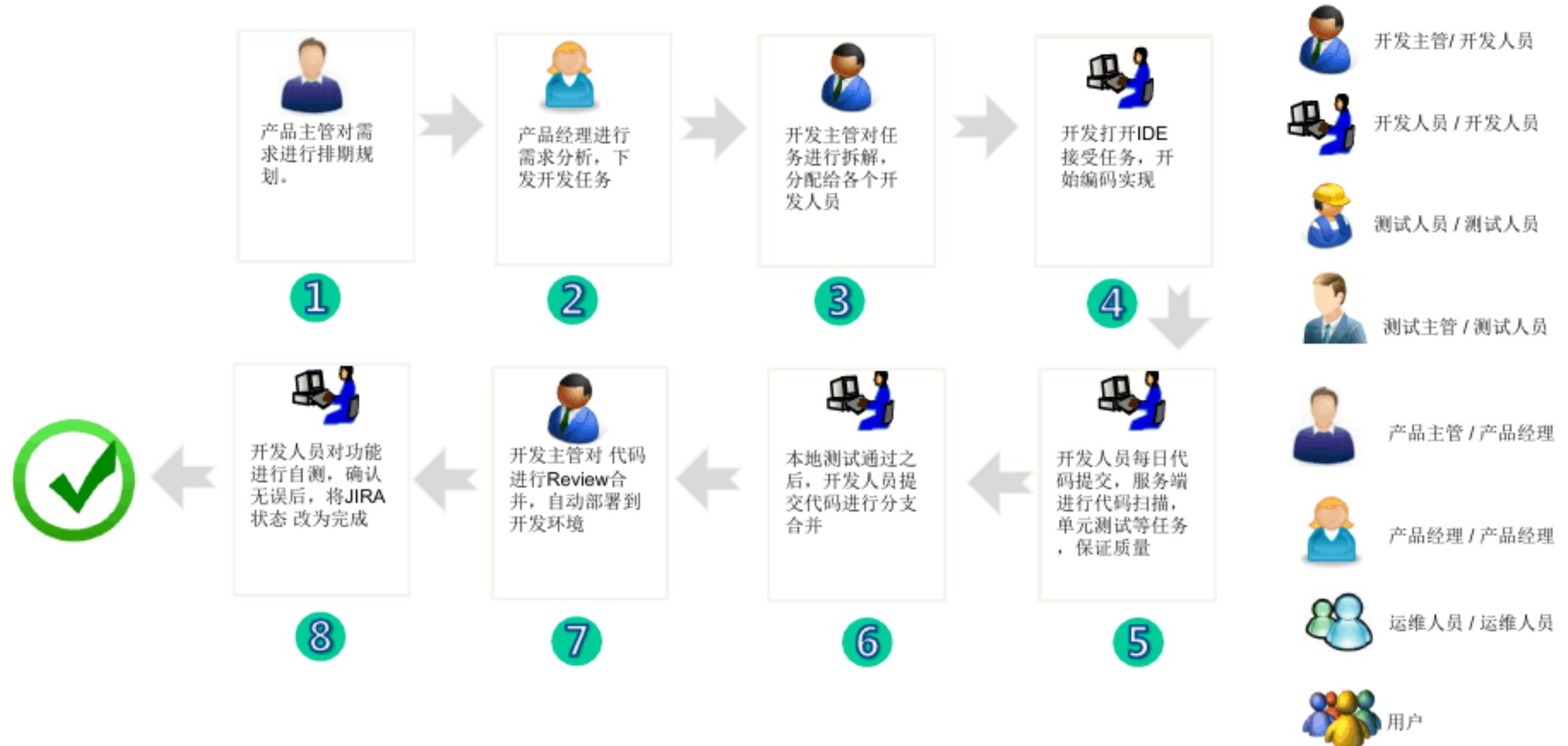
测试计划

- 测试计划编排管理

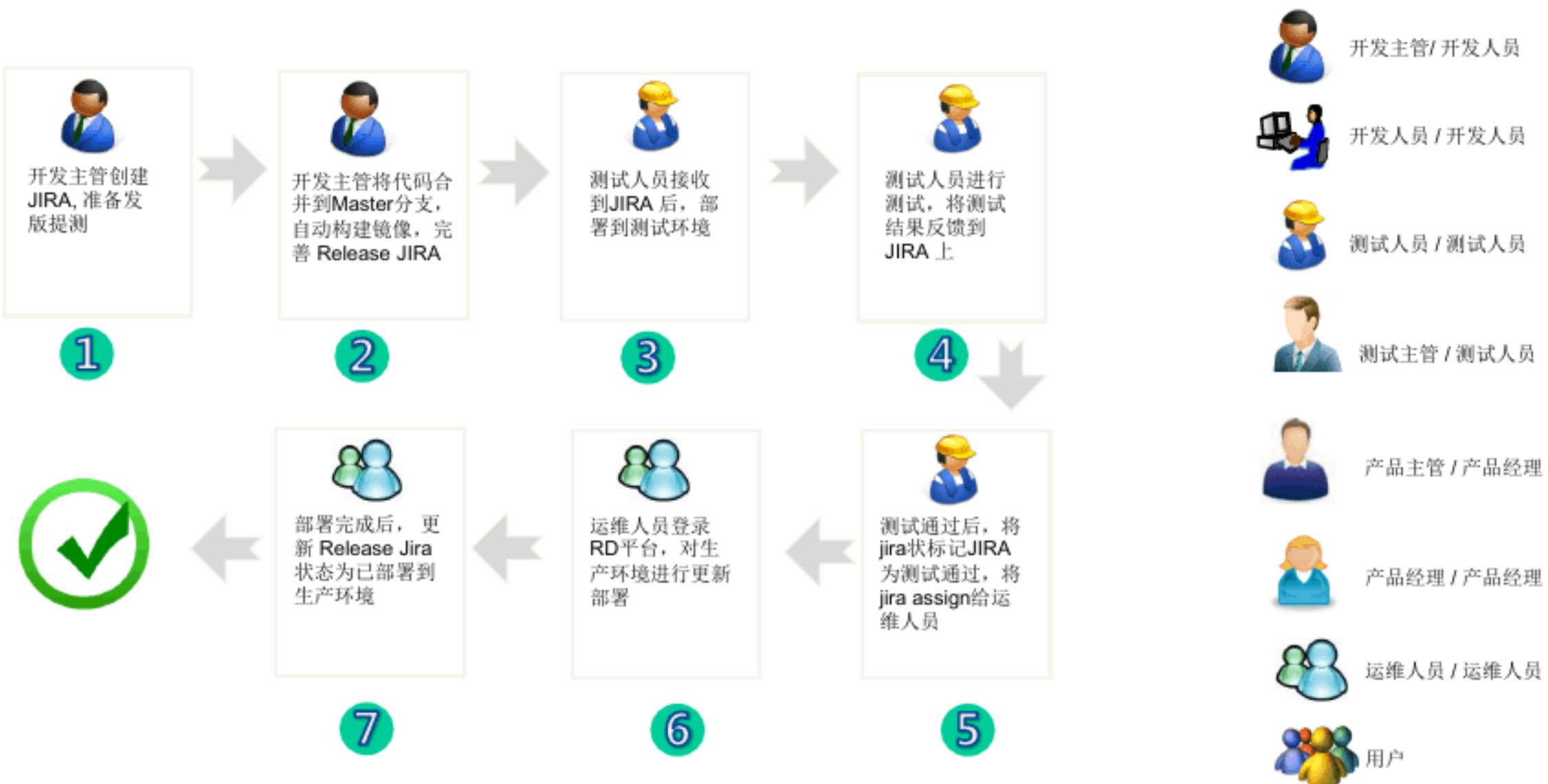
测试结果

- 测试结果展示分析

场景 - 需求研发



场景 - 测试上线



结束

谢谢