

安信证券DevOps探索与实践

安信证券 蒋渐峰

目录

CONTENTS

01

转型背景

02

工具平台建设

03

试点项目实践

04

持续改进

转型背景

目标：以研发团队为中心，在端到端流程串联、流程自动化、度量精化、质量增强、资源自助化几个重点方面发力，打造研发管理平台，提供具备快速交付、高质量、过程透明、可度量的IT研发服务供应链。



外部

证券业务的复杂性：

证券业务种类多，业务规则复杂，业务链条长，业务发展快速，监管严格，面向客户类型多，**证券业务的多样性和复杂性带来了IT建设和快速响应支持的压力**

数字化战略的导向：

基于行业的深度竞争以及公司业务的快速发展，证券行业这几年纷纷加大了IT建设投入，并提出数字化转型的战略，IT的自主研发实力代表**IT自主可控的核心竞争力**，是业务开展的速度、广度以及深度的保证，这恰恰是行业所缺失的



内部

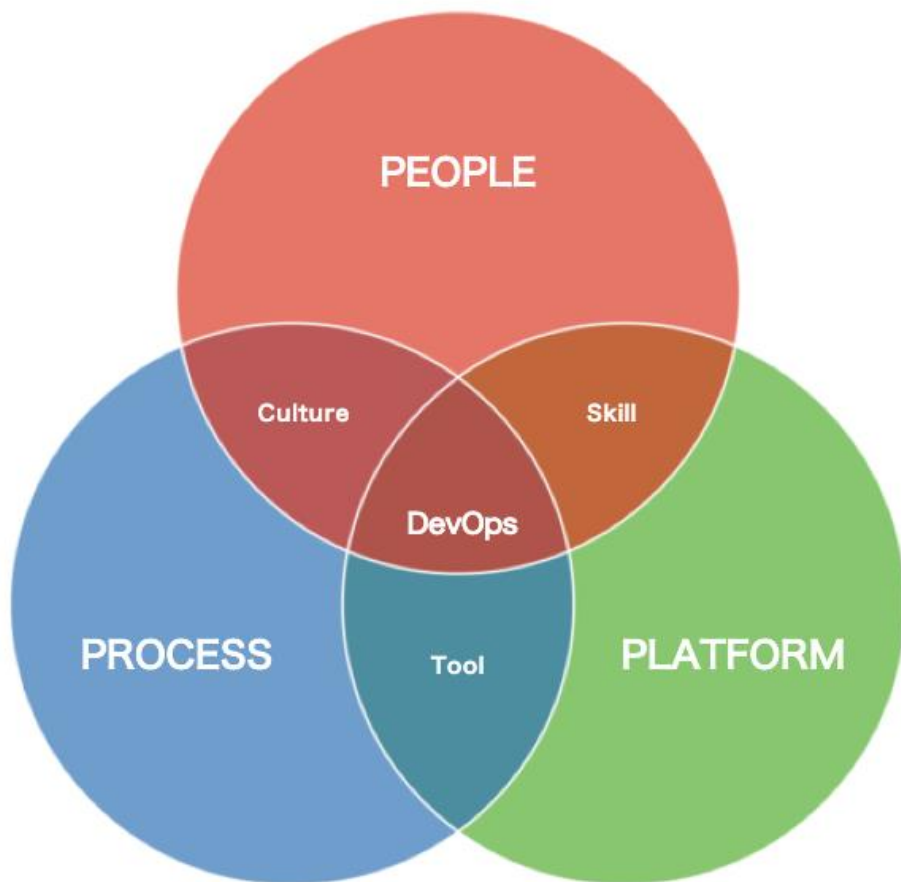
行业供应商能力僵化倒逼：

行业信息系统服务商丰富，技术能力强，但多以产品为中心，行业的激烈竞争以及客户对软件产品服务的时效和功能要求导致需求愈发复杂和多变，**供应商无法满足行业的个性化的需求，甚至成为束缚**，证券公司过去以采购为主的IT系统建设方式已经行不通

IT规模扩展带来的管理问题：

近年证券行业的IT规模不断扩展，特别是在自研领域，系统建设模式逐步转变为自主研发、合作研发为主，开发团队规模逐渐壮大，安信目前自主研发和合作研发的比例已经超过50%，研发团队也超过了500人，各个二级团队都有自研的项目，亟待建立研发管理体系，统一研发过程和工具

转型背景-实施思路



3 个支柱：人 (People) 、流程 (Process) 和平台 (Platform)

3个关键方面：文化 (Culture)、工具 (Tool)、培训赋能 (Skill)

工具平台建设



试点项目转型

目录

CONTENTS

01

转型背景

02

工具平台建设

03

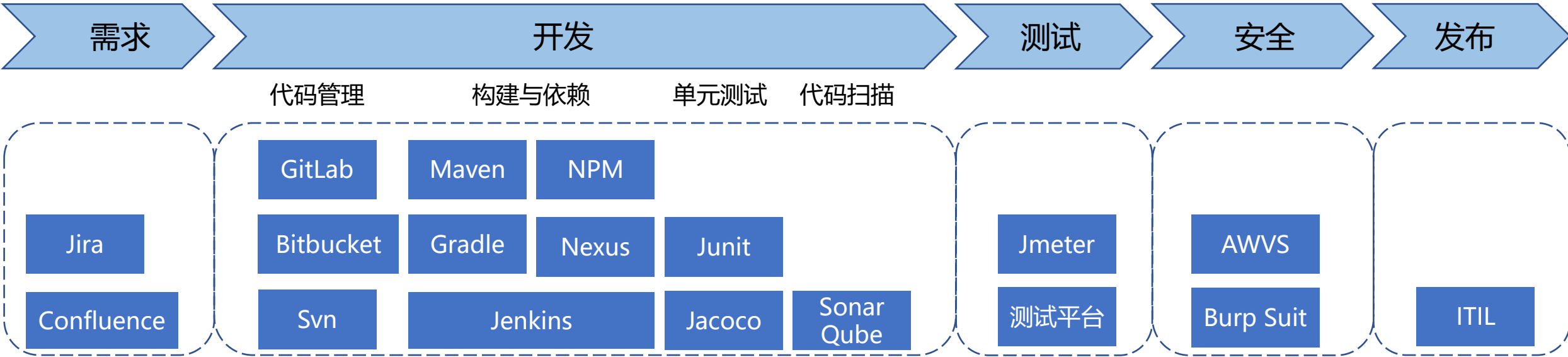
试点项目实践

04

持续改进

工具平台建设-从0到1

● 工具平台建设初期工具使用情况



● 工具平台建设初期方案选择

1 自研

- 自主可控，又有核心竞争力
- 高成本，高投入

2 成熟产品

- 经过长期积累
- 可能会水土不服
- 定制成本高

3 基于开源及商业工具自建

- 基于原有经验积累
- 快速补齐能力短板

工具平台建设-从0到1



战略目标

效率

质量

成本

安全

落地实践

配置管理

持续集成

内建质量

部署与发布管理

度量与反馈

需求管理

开发与测试管理

产品&需求管理

项目管理

测试管理

代码管理

构建

单元测试

代码扫描

制品管理

自动化测试

性能测试

安全

部署

Confluence

测试平台

GitLab

Maven

Junit

Sonar
Qube

Artifacoy

测试平台

Jmeter

Xray

Ansible

Jira

Gradle

Jacoco

NPM

AWVS

ITIL

Burp Suit

Jenkins

分析与度量

开发环境

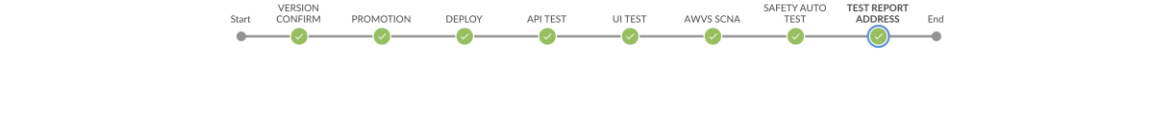
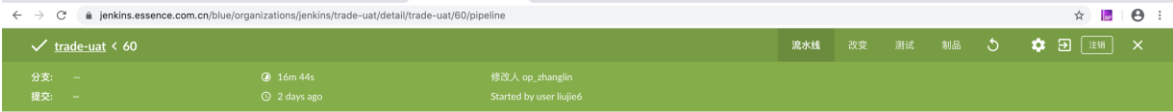
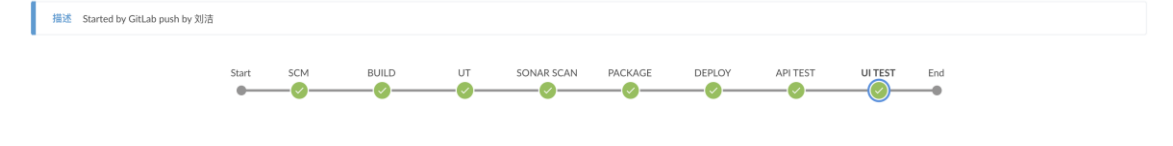
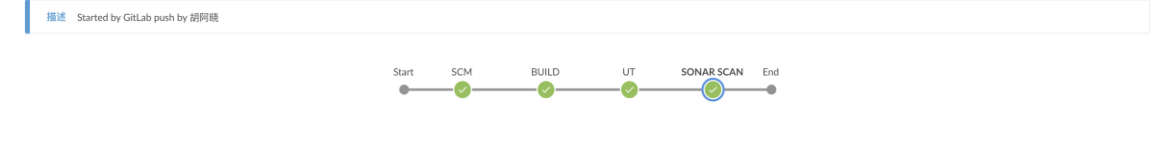
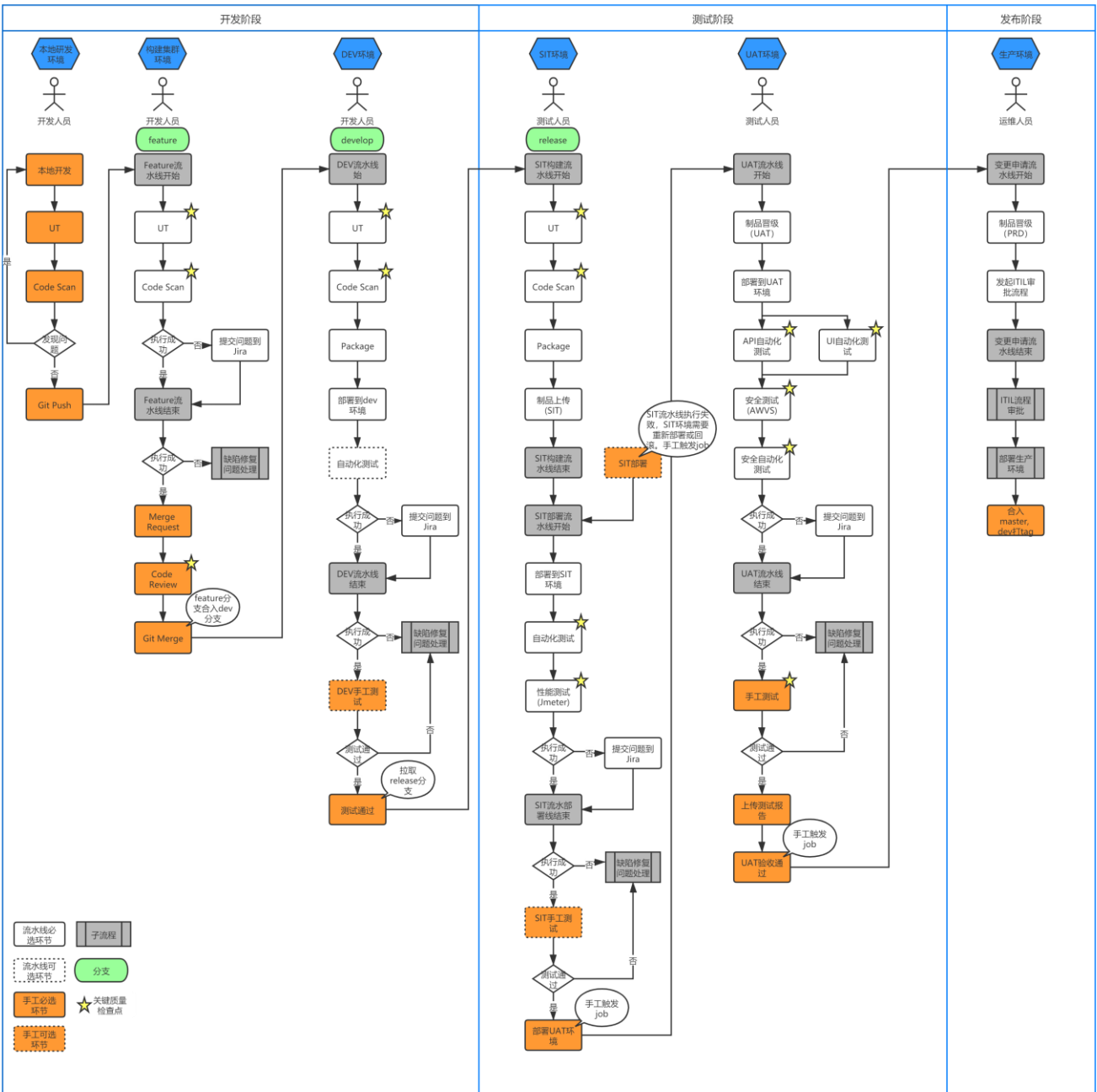
SIT环境

UAT环境

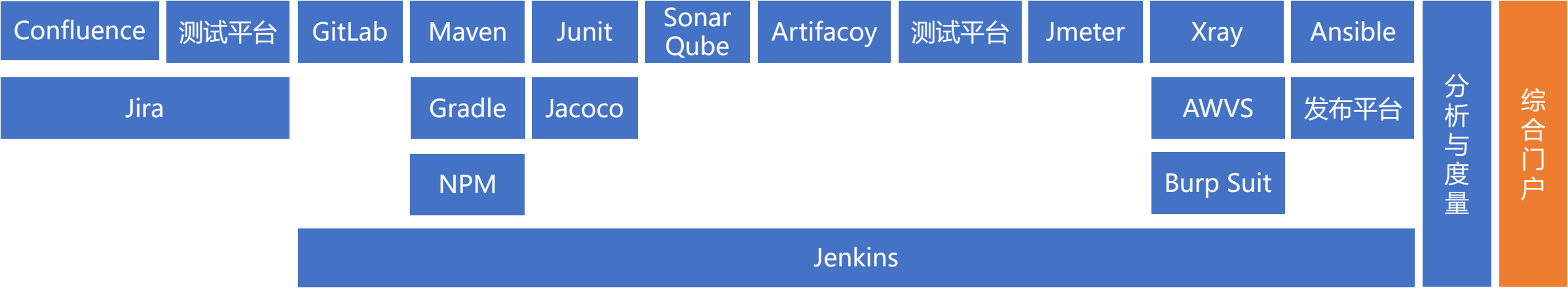
生产环境

基础设施

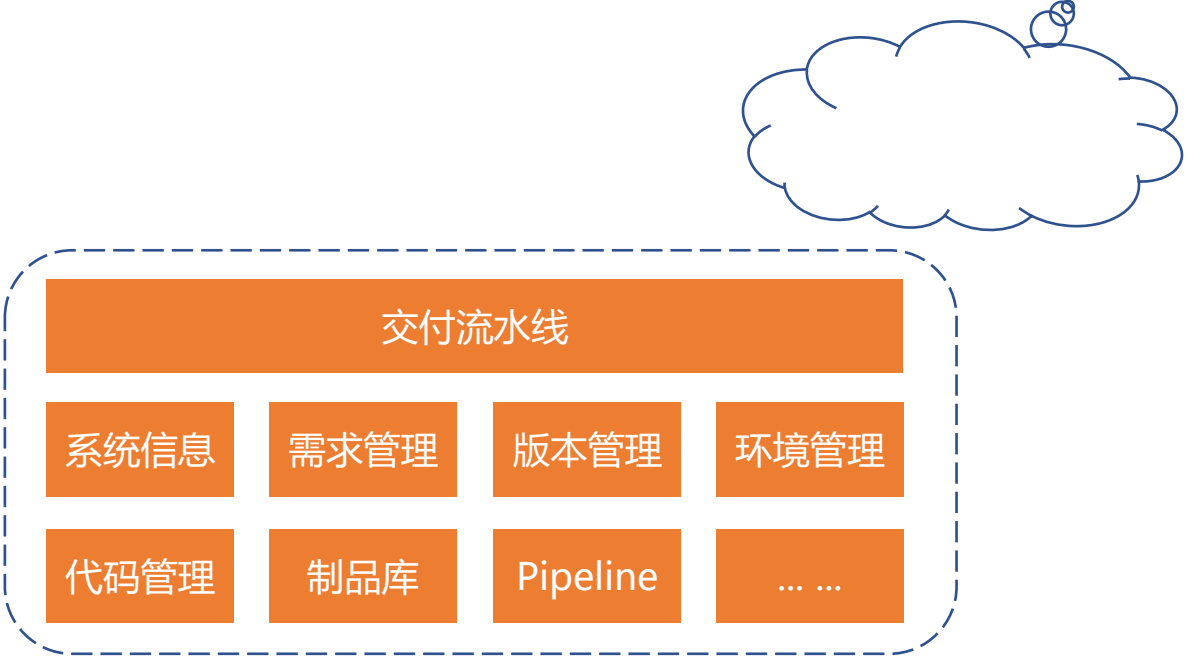
工具平台建设-流水线过程



工具平台建设-综合门户



- 流水线可视化配置；
- 从系统维度对各个服务的流水线进行集成和串联；
- 对流水线集成的工具统一管理配置，提供自助服务。
- 从应用视角整合工具链功能，提供统一的入口和平台。



工具平台建设-可视化流水线配置

Pipeline / 编辑Pipeline

< 编辑Pipeline

Pipeline名称

axzq-trade-sit

* 所属服务

trade

外部模板

new_pipeline_tpl

选择模板

* 触发方式

自动触发

生成hook地址

更新

设置参数

Pipeline定义

操作提示: 鼠标移到pipeline流程图中 ▶ 图标时,出现⊕图标时添加pipeline阶段

开始

 →

1.源码管理

 ▶

2.编译构建

 ▶

3.测试

 ▶

4.质量检查

 ▶

5.制品仓库

 ▶

6.物理部署

 ▶

拉取代码

MAVEN构建

单元测试

SONAR扫描

打包

部署服务

添加并执行任务

添加并执行任务

添加并执行任务

添加并执行任务

添加并执行任务

添加并执行任务

7.测试

 ▶

8.制品仓库

 ▶

结束

API测试

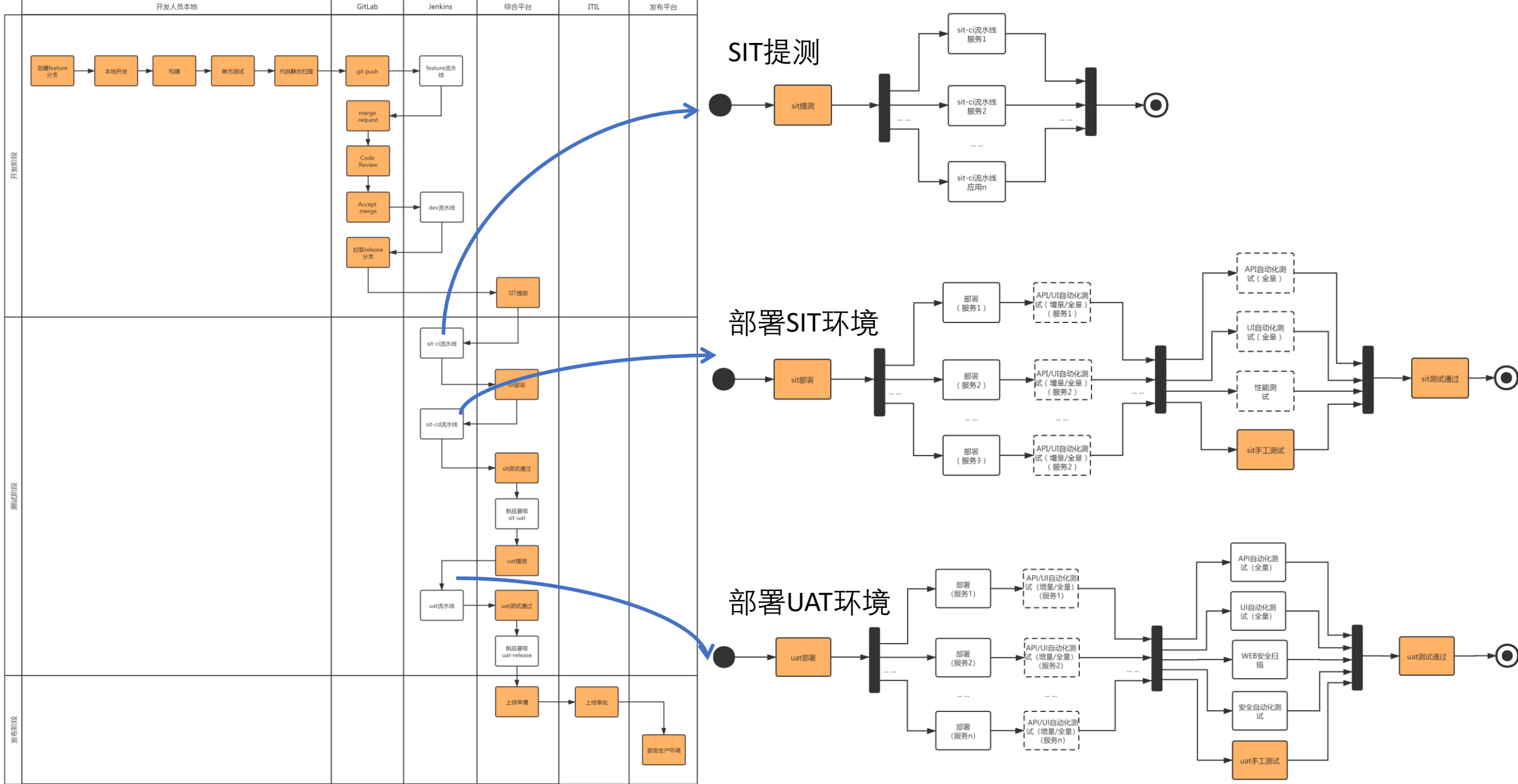
上传到制品仓库

添加并执行任务

添加并执行任务

- 通过可视化的方式简化定义每个系统、每个服务自己流水线的过程，提供更加灵活的流水线支持；
- 提供统一任务模板给大家选择，模板中对应的功能在统一的pipeline脚本中实现，保存到代码仓库中，能够实现问题的统一修复和功能的统一升级；
- 因为流水线功能分为了pipeline脚本，和可视化方式配置两部分，通过可视化配置的部分需要考虑留痕和版本回溯的问题。

工具平台建设-面向应用的交付流水线



工具平台建设-面向应用的交付流水线



SIT部署

服务名	版本号	执行pipeline
platform	beta1599198173000	devops-platform-sit-cd
web	beta1599198173000	devops-web-sit-cd

关闭 部署

SIT测试

SIT测试版本: beta1600075866000 生成版本 自定义

☐ 当前完成需求 0/5

- ☐ jenkins中返回错误信息查看详情地址改为portal中的job执行记录页面-前端
- ☐ 制品库存储路径优化-流水线
- ☐ Portal交付流水线中展示交付过程中产生的内容-制品库-后端
- ☐ Portal交付流水线中展示交付过程中产生的内容-新增接口记录产出信息-后端
- ☐ Portal交付流水线中展示交付过程中产生的内容-按服务展示内容-前端

☐ 提测需求 0/0

无数据

服务名	执行Pipeline	分支名	状态	操作
<input type="checkbox"/> measure	请选择	请选择		发布
<input type="checkbox"/> pipeline	请选择	请选择		发布
<input checked="" type="checkbox"/> platform	devops-platform-sit-ci	release3.2.0		发布
<input type="checkbox"/> synergy	请选择	请选择		发布
<input type="checkbox"/> upms	请选择	请选择		发布
<input type="checkbox"/> gateway	请选择	请选择		发布
<input checked="" type="checkbox"/> web	devops-web-sit-ci	release0.1.0		发布

发布 关闭

目录

CONTENTS

01

转型背景

02

工具平台建设

03

试点项目实践

04

持续改进

试点项目实践-配置管理



全流程可追溯：当出现问题，能够追溯源代码、测试报告、运行环境等数据。针对任意需求，能够快速识别出其关联的源代码、版本、测试用例、上线记录、缺陷信息等。

将一切纳入版本控制

- 源代码
- 配置文件
- 数据库脚本
- 部署脚本
- Dockerfile, Helm Charts
- ...

版本变更标准化

- 代码提交记录与需求的双向关联

单一可信数据源

- 统一的两方库、三方库
- 统一的生产发布仓库
- 漏洞及licence扫描

配置管理-版本变更标准化



- Jira issue key

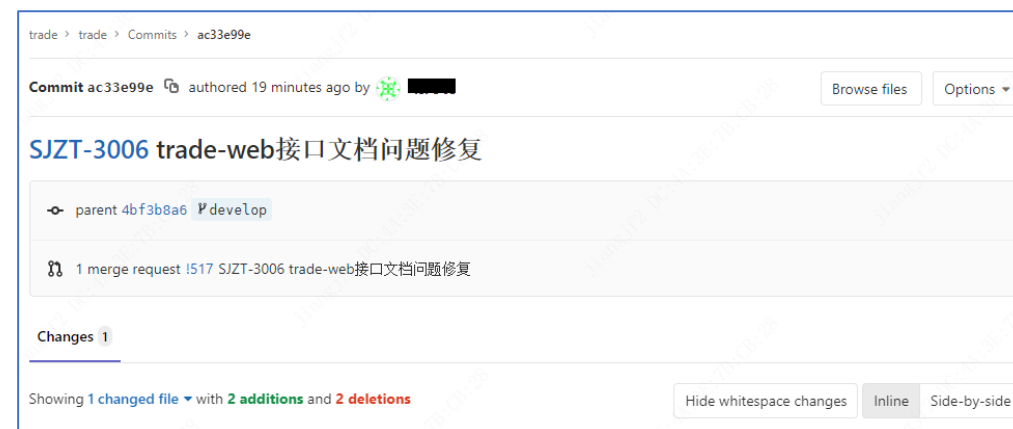
检查Jira中对应的任务是否存在

- Jira issue 状态

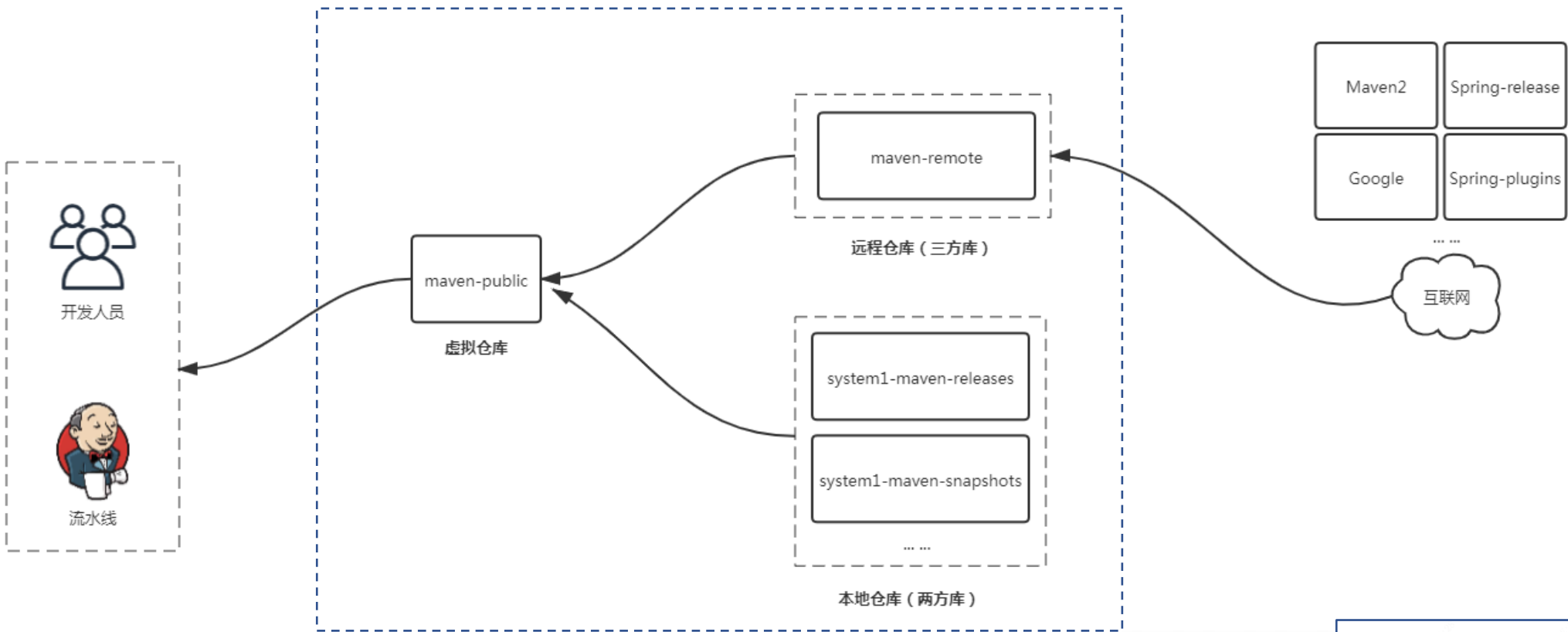
检查任务状态是否正确

- 提交人与经办人信息

检查任务的经办人是否与代码提交人一致



配置管理-单一可信数据源



Severity	Issue Type	Component
High	Security	io.netty:netty-codec:4.1.45.Final
Medium	Security	org.apache.tomcat.embed:tomcat-embed-cor...
Medium	Security	org.apache.tomcat.embed:tomcat-embed-cor...
Medium	Security	io.grpc:grpc-core:1.14.0
Medium	Security	ch.qos.logback:logback-core:1.2.3
Medium	Security	commons-io:commons-io:2.6
Medium	Security	ch.qos.logback:logback-core:1.2.3

试点项目实践-持续集成



频繁地将工作成果集成到一起，并且在每次提交后，自动触发运行一次包含自动化验证集的构建任务，以便尽早地发现集成问题

01

频繁集成

- 每次代码提交触发完整流水线过程
- 集成规则

02

自动化验证

- 单元测试
- 代码静态扫描
- 部署
- 自动化测试

03

第一时间修复

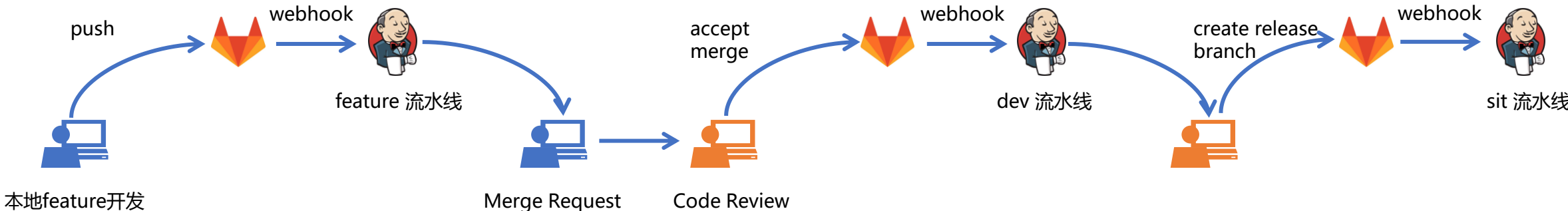
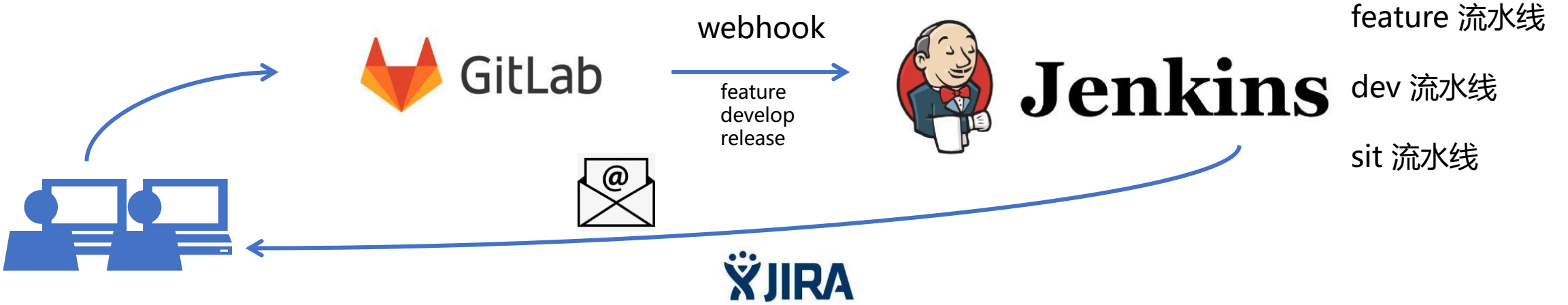
- 优化流水线执行时长
- 关注红灯修复时长

04

标准化的资源池

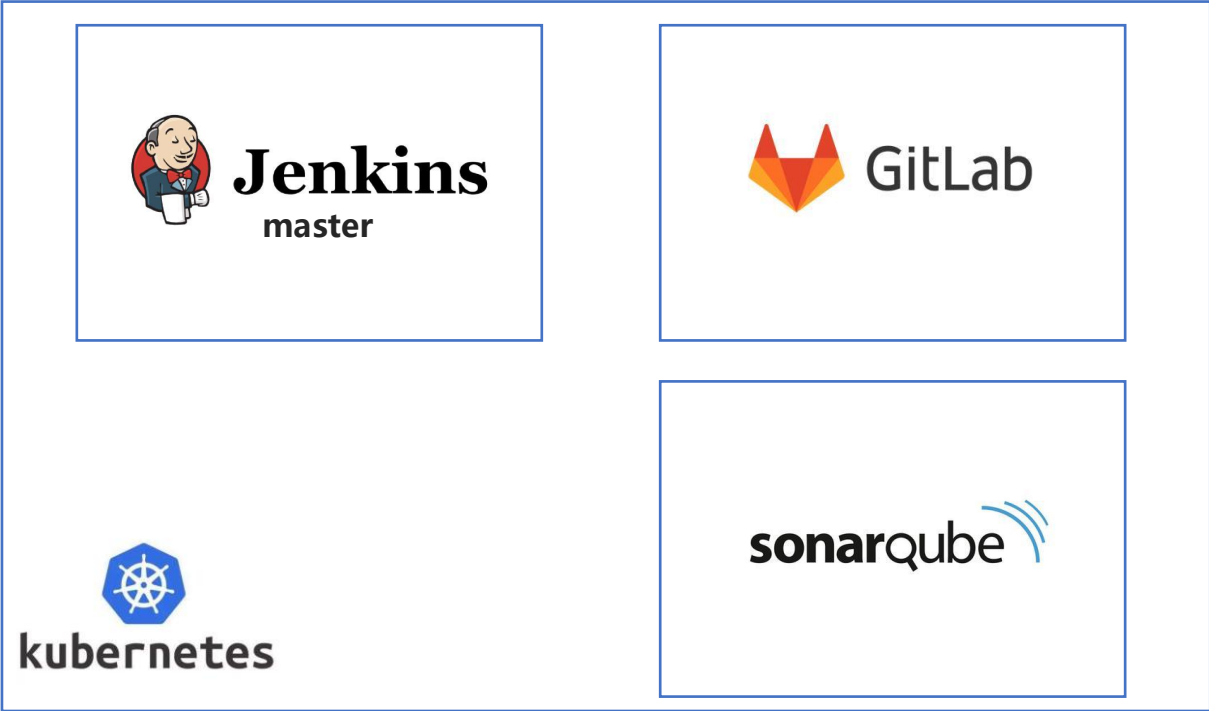
- 构建环境标准化
- 动态按需初始化

持续集成-集成规则及自动触发流水线

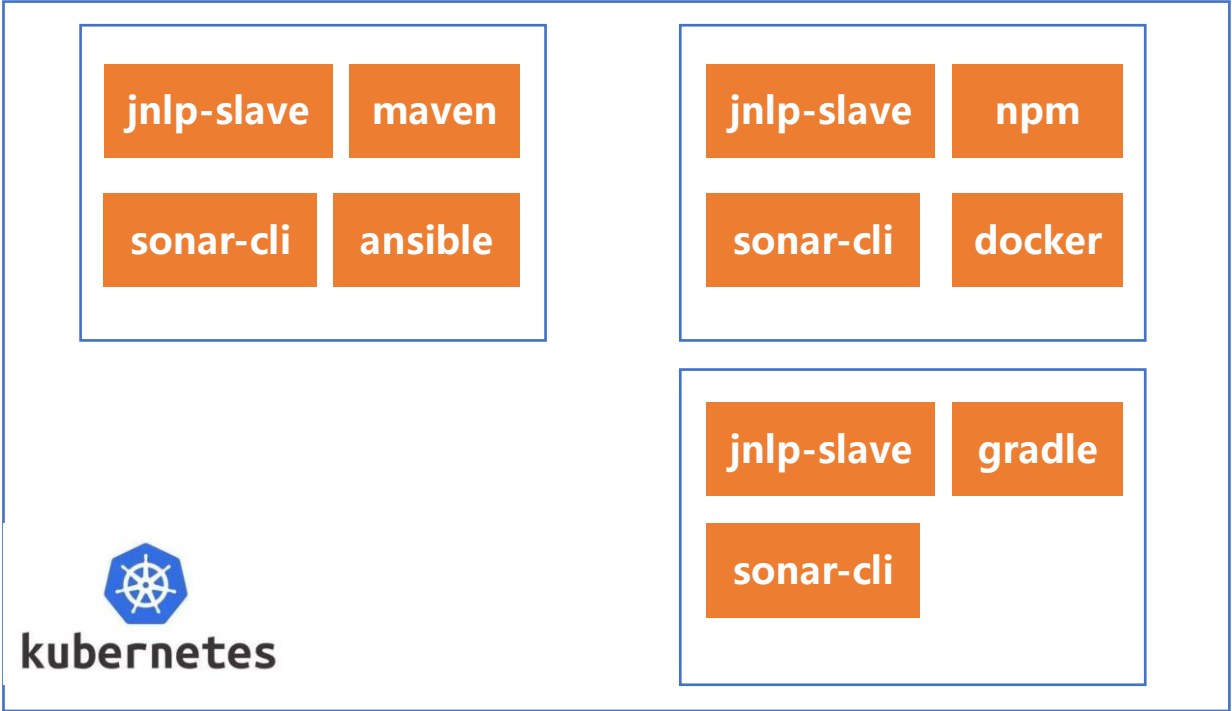


持续集成-标准化的资源池

工具集群

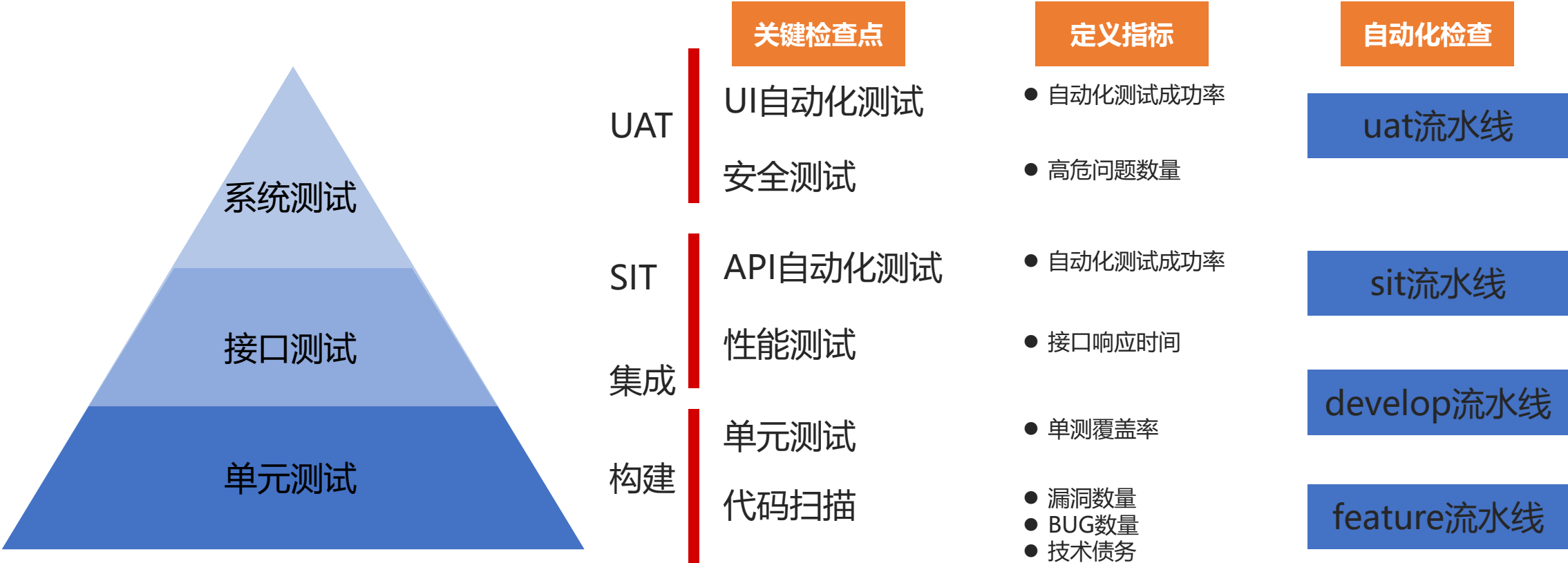


构建集群



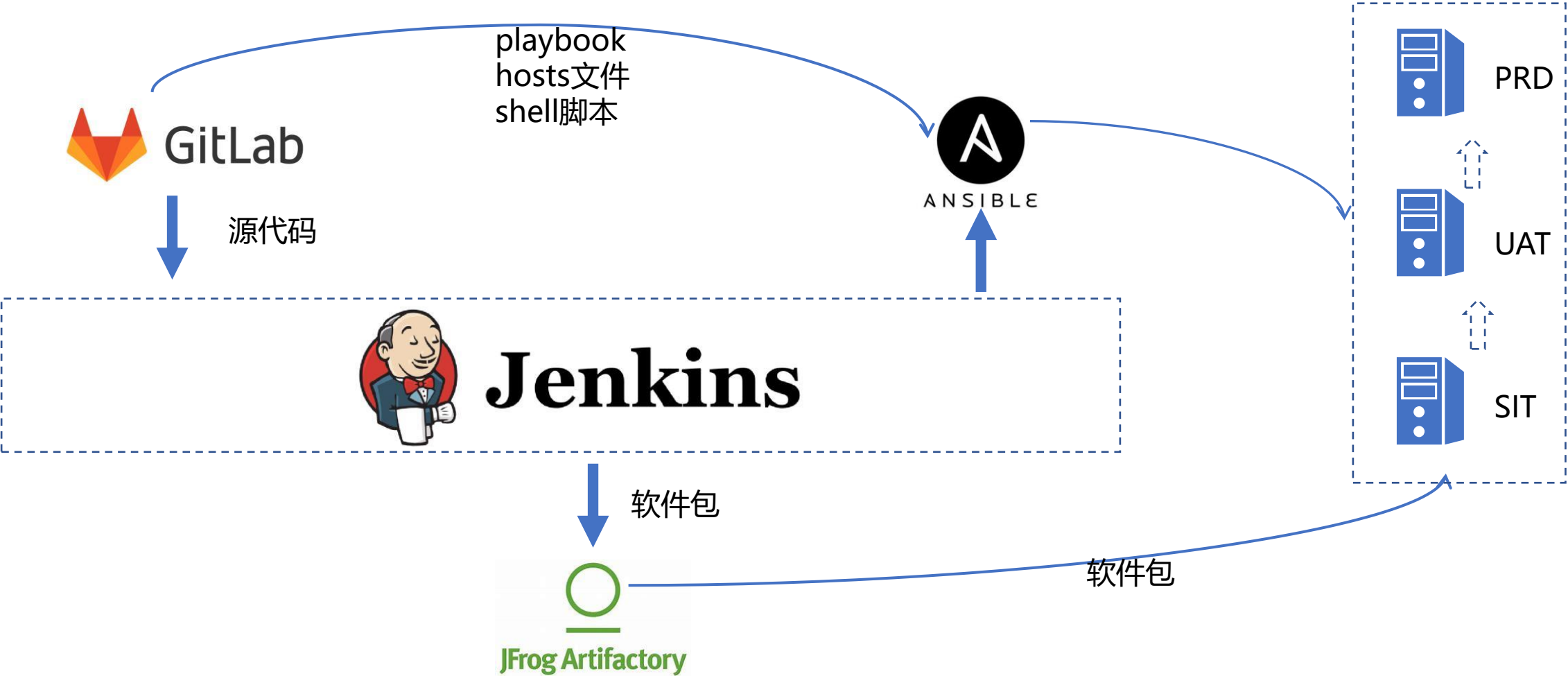
试点项目实践-内建质量

- 问题发现得越早，修复成本就越低；
- 质量是每个人的责任，而不是质量团队的责任。

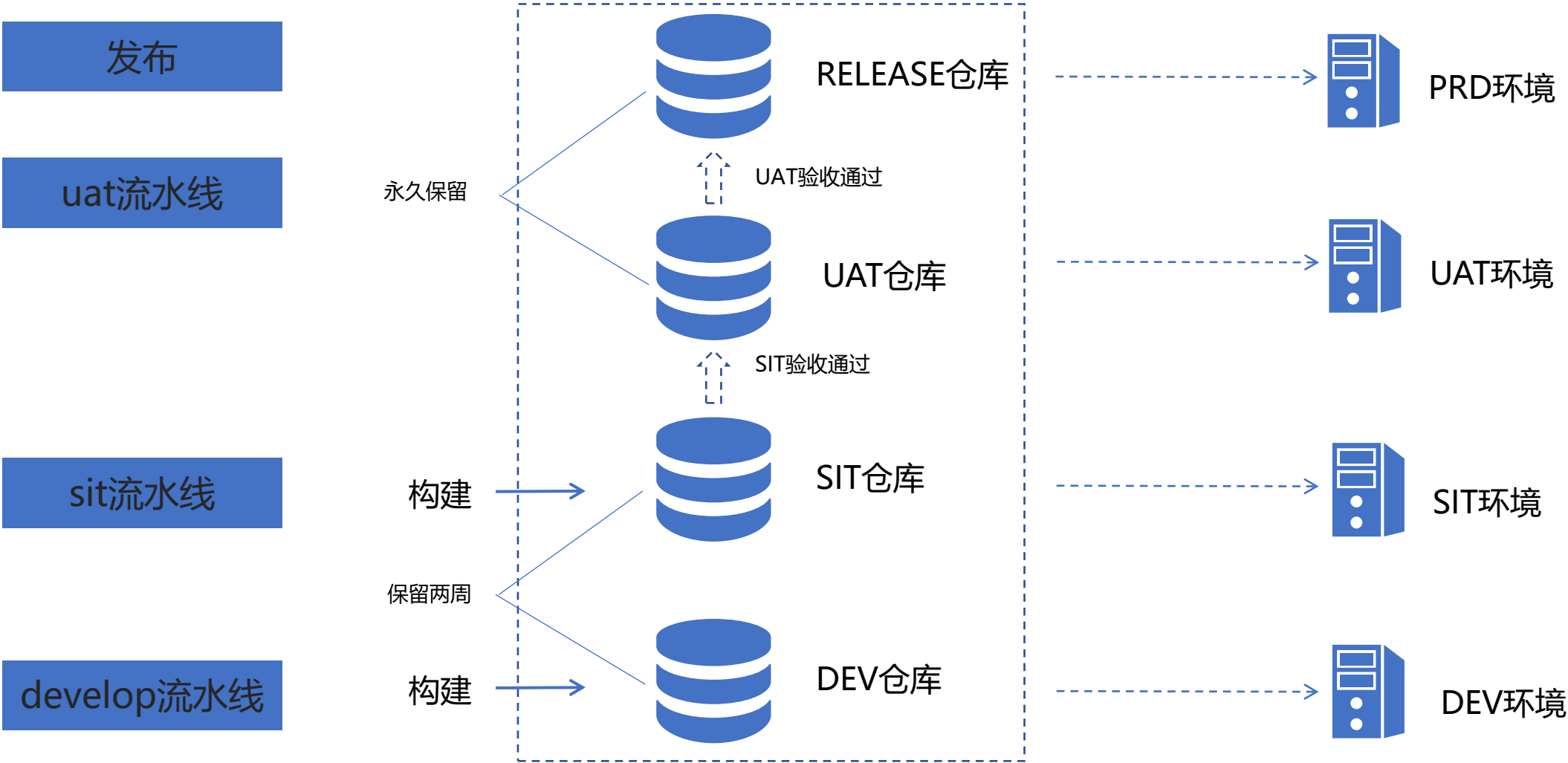


试点项目实践-部署与发布管理

同一个制品，使用同样的过程和工具部署所有环境



部署与发布管理-制品晋级



试点项目实践-度量与反馈



细化指标

- 分解出可度量的指标
- 确定计算公式
- 确定源数据

sonarqube

Jfrog Artifactory

GitLab

ANSIBLE

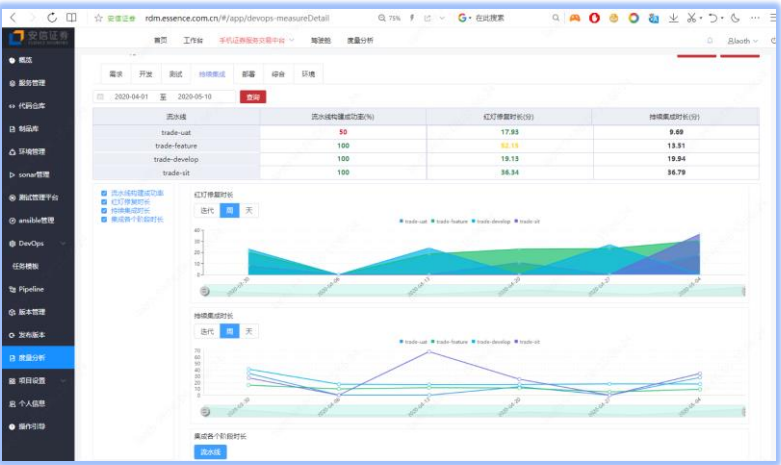


Jenkins

JIRA

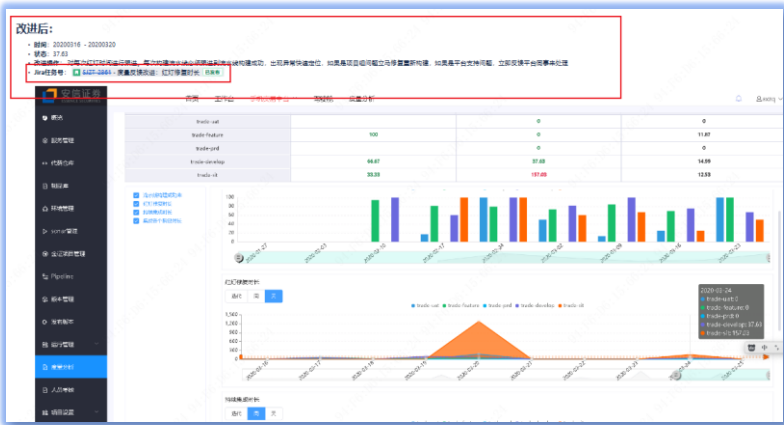
采集数据

- 工具数据打通
- 采集、计算、分析



可视化平台

- 度量数据向全员展示
- 能够检视指标的趋势



持续改进

- 阈值管理
- 问题发现
- 改进与跟踪

度量与反馈-持续改进

●发现问题

对指标设置阈值，异常的数据告警

●识别瓶颈

根据细化指标进一步确定问题的瓶颈，找到改进方法

●改进跟踪

记录改进问题处理，并通过度量数据检视改进效果

反馈改进-红灯修复时长

改进前:

- 红灯修复时长: 流水线上构建失败的时间, 到下一次成功的构建时间。
- 时间: 2020-03-11
- 当时状态: 3814 分钟
- 问题描述: 通过红灯修复时长的指标, 发现项目组的流水线出现长时间失败未修复的问题。经检查是由于流水线的失败时出现在周五的时候, 开发人员未及时修复, 到周一的时候才修复, 因此这个失败持续了一个周末。

手机交易中台详情

需求 开发 测试 持续集成 部署 综合 环境

2020-02-01 至 2020-03-09 查询

流水线	流水线构建成功率(%)	红灯修复时长(分钟)
trade-uat		0
trade-feature	80	229.2
trade-prd		0
trade-develop	100	3814.63
trade-sit		0

☐ 流水线构建成功率

☒ 红灯修复时长

☒ 持续集成时长

☒ 部署各个阶段时长

红灯修复时长

迭代 周 天

改进后:

- 时间: 20200316 - 20200320
- 状态: 37.63
- 改进操作: 对每次红灯时间进行跟进, 每次构建流水线必须跟进到流水线构建成功, 出现异常快速定位, 如果是项目问题立马修复重新构建, 如果是平台支持问题, 立即反馈平台同事来处理
- Jira任务号: [SI-ZT-2664 - 度量反馈改进: 红灯修复时长](#) 已发布

安信证券

首页 工作台 手机交易中台 驾驶舱 度量分析

概述

服务管理

代码仓库

制品库

环境管理

sonar管理

金证项目管理

Pipeline

版本管理

发布版本

运行管理

度量分析

人员考核

	trade-uat	trade-feature	trade-prd	trade-develop	trade-sit
trade-uat					0
trade-feature		100			11.87
trade-prd					0
trade-develop		66.67		37.63	14.99
trade-sit		33.33		157.03	12.53

☒ 流水线构建成功率

☒ 红灯修复时长

☒ 持续集成时长

☒ 部署各个阶段时长

红灯修复时长

迭代 周 天

目录

CONTENTS

01

转型背景

02

工具平台建设

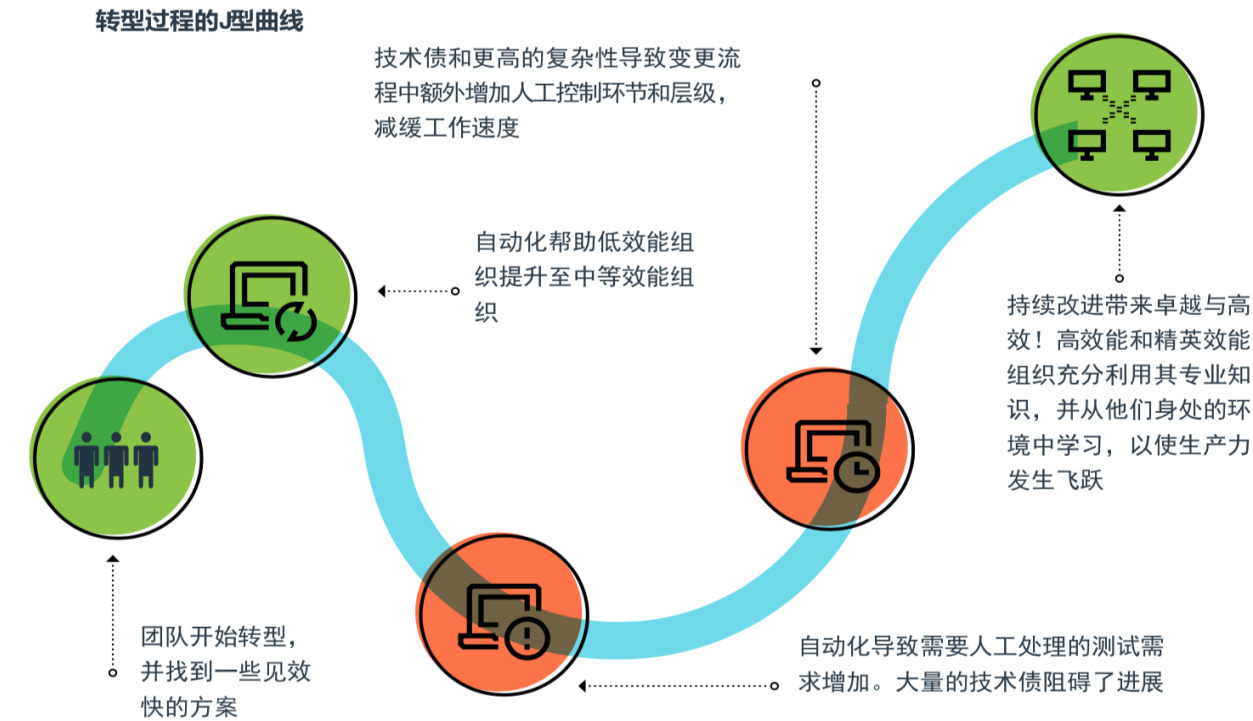
03

试点项目实践

04

持续改进

持续改进-现状与规划



- 刚刚跨过转型初期，通过提升自动化能力和水平，快速获得初期成功
- 随着交付能力的提升，质量能力和技术债务的问题开始凸显
- 流水线执行时长优化，度量有效性改善
- 优化系统架构，降低技术债务，提升自动化测试覆盖率

从无到有	从小到大	从繁到简
<ul style="list-style-type: none">● 快速补齐能力短板● 支撑转型实践，快速提升自动化的能力和水平● 帮助团队获得初期的成功	<ul style="list-style-type: none">● 推广使用：自研系统覆盖50%→90%● 各平台功能在垂直领域进一步深化● 工具的稳定性、可靠性，以及大规模使用的性能问题● DevOps文化的推行，培训赋能	<ul style="list-style-type: none">● 统一界面、简化操作● 平台治理● 标准化、自动化、服务化、数据化

谢 谢