

2019

持续集成建设分享

智能平台部



彭晶

CONTENTS

01

● ● ● ● ● ●

02

● ● ● ● ● ●

03

● ● ● ● ● ●

04

为什么需要持续集成

怎么实践持续集成

部门持续 集成建设情况

持续集成 后续规划

1

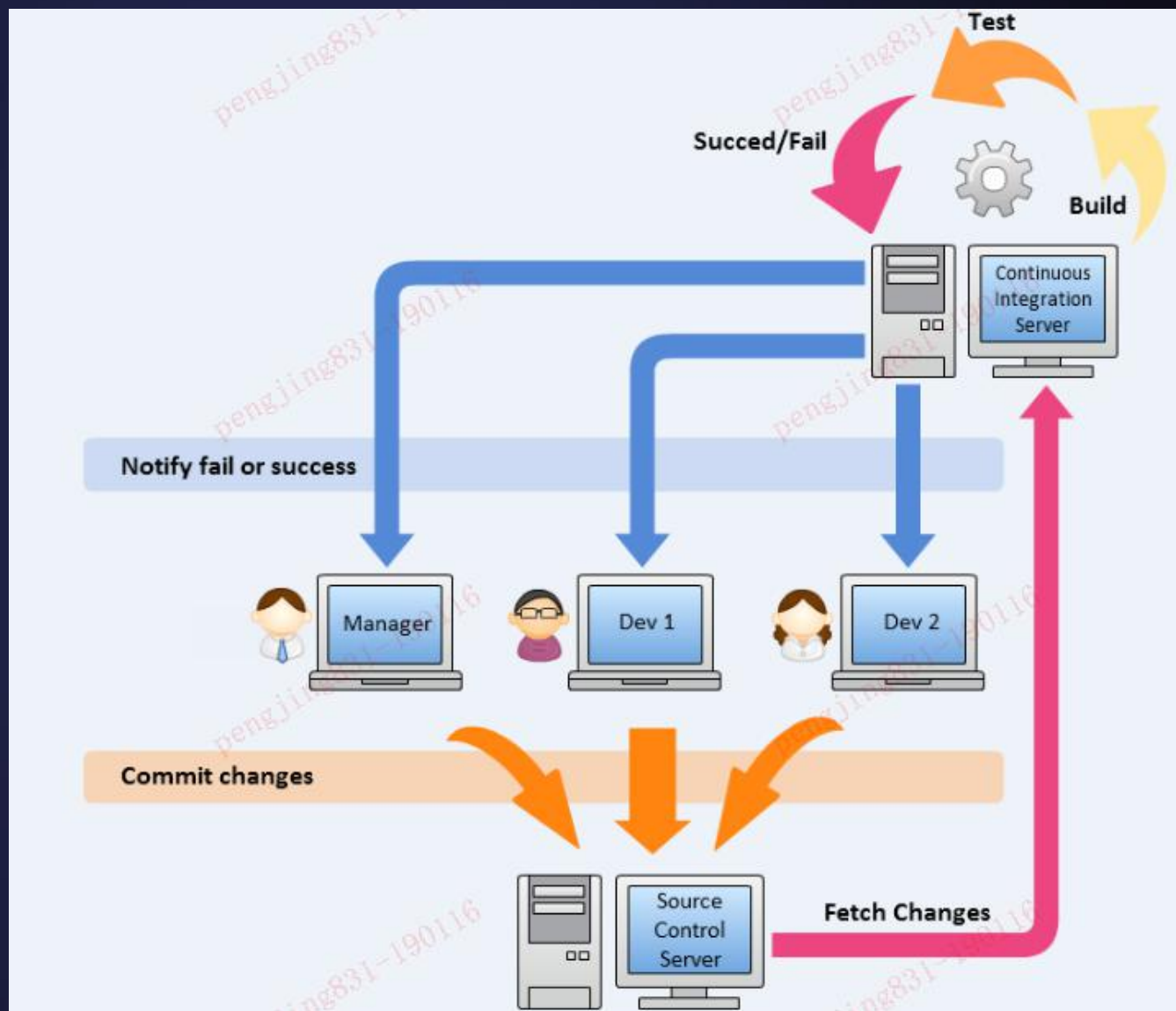
PART

为什么需要持续集成



什么是持续集成—CI (Continuous Integration)

- 持续集成是一套**软件工程方法论和一系列的最佳开发过程实践**，团队开发成员经常集成他们的工作，每天至少一次或多次集成。
- 每次集成都通过自动化的构建（编译→部署→自动化测试）来验证，从而**尽早地发现集成错误**。





引入持续集成的好处



通过静态代码扫描、单元测试等手段实现代码级别的问题发现和质量保障

保障代码质量



持续的验证让团队成员不必为潜在错误可能造成的后果担忧

增强团队信心

降低集成风险

频繁地持续集成、验证，减少到项目后期才统一集成时发现大量问题和冲突的风险



有效的版本控制

团队成员提交的代码有问题时，团队会即时收到通知并在其他成员拉取之前就解决问题



2 PART

怎么实践持续集成



持续集成、快速反馈、降低风险、质量内建

标准化

自动化

可视化

Jenkins平台

git

junit

maven

httprunner

sonarqube

jmeter

团队协作和纪律

数据度量和分析

渐进式实施

提交与编译

- 编译构建
- 代码扫描
- 单元测试

测试与验证

- 集成测试
- 系统测试

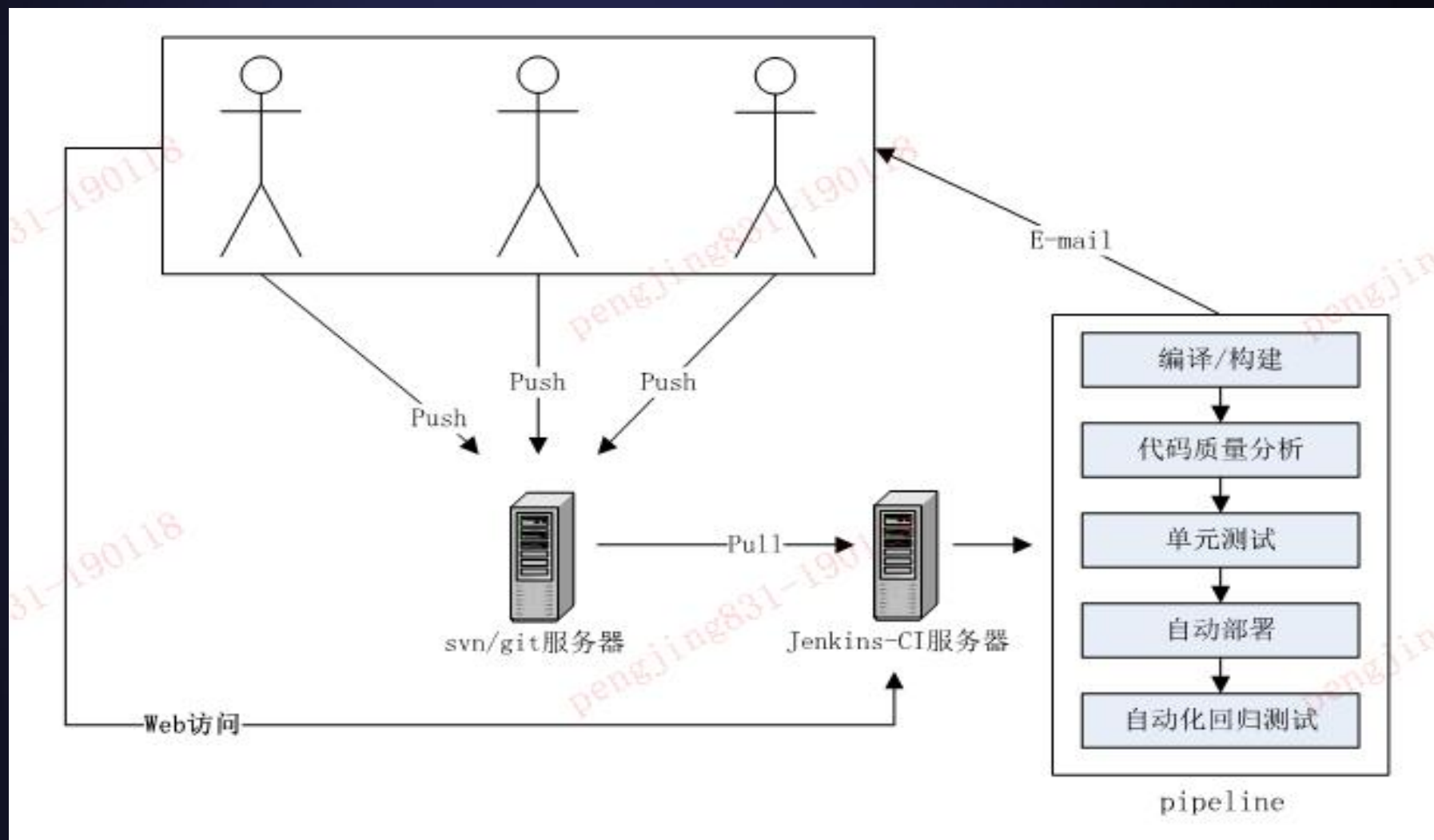
部署与运维

- 版本发布
- 生产部署



持续集成平台---基于jenkins

- 使用jenkins搭建部门持续集成平台
- 接入sonarqube进行代码扫描及代码综合质量管理
- 推动单元测试用例建设
- 非标系统通过持续集成平台自动部署
- 自动化测试用例进行代码提交后的自动化集成回归测试验证





代码质量分析 ---- sonarqube开源平台

复杂度分布

文件、类、方法等，复杂度过高使得开发人员难以理解

重复代码

找出源码中存在的复制、黏贴导致的重复严重的地方

代码规则检查

通过findbugs, PMD, CheckStyle等检查代码是否符合规范



注释率

注释过少，人员变动后接手困难。
注释过多，不利于阅读

潜在的bug

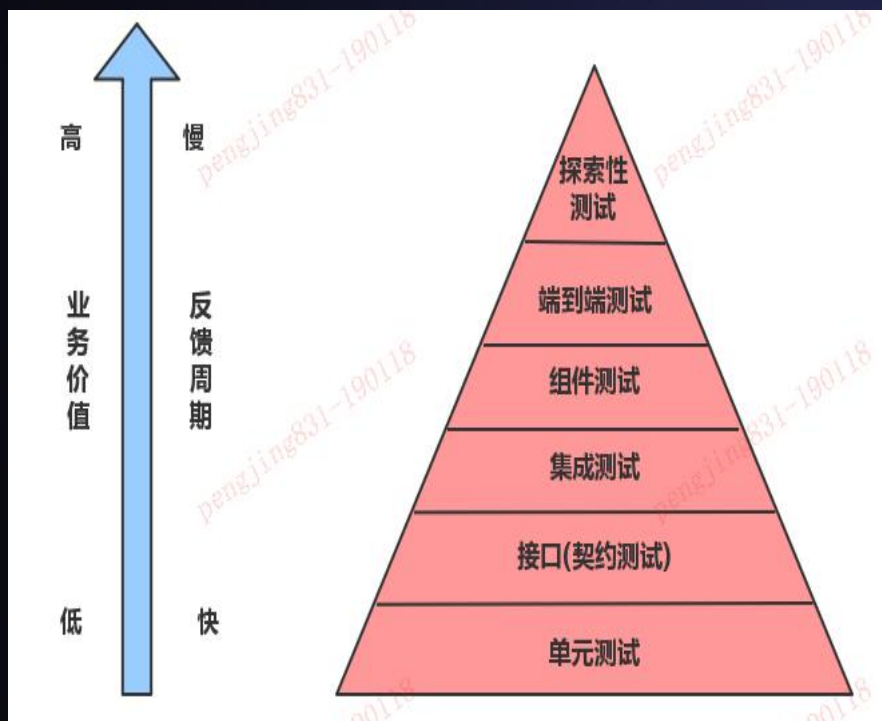
通过findbugs, PMD, CheckStyle等检查潜在的bug

结构与耦合度

找出循环，展示包与包、类与类之间的依赖、检查程序之间耦合度



单元测试 ---- junit框架



lcloud-ark-agent

Element	Missed Instructions	Cov.	Missed Branches	Cov.
com.pingan.lcloud.agent.model	<div><div></div></div>	39%		n/a
com.pingan.lcloud.agent.disruptor.exception	<div><div></div></div>	6%		n/a
com.pingan.lcloud.agent.util	<div><div></div></div>	84%	<div><div></div></div>	50%
com.pingan.lcloud.agent.config	<div><div></div></div>	76%		n/a
com.pingan.lcloud.agent.disruptor.event	<div><div></div></div>	58%		n/a
com.pingan.lcloud.agent.disruptor.consumer	<div><div></div></div>	90%	<div><div></div></div>	50%
com.pingan.lcloud	<div><div></div></div>	37%		n/a
com.pingan.lcloud.agent.disruptor.service.impl	<div><div></div></div>	93%		n/a
com.pingan.lcloud.agent.service.impl	<div><div></div></div>	100%	<div><div></div></div>	91%
com.pingan.lcloud.agent.cache	<div><div></div></div>	100%	<div><div></div></div>	100%
com.pingan.lcloud.agent.consumer	<div><div></div></div>	100%	<div><div></div></div>	66%
com.pingan.lcloud.agent.controller	<div><div></div></div>	100%		n/a
Total	188 of 837	77%	8 of 42	80%

```
28. public class HttpUtil {
29.
30.     public static final String CONTENT_TYPE_VALUE = "application/json";
31.
32.     public static final String PARSE_EXCEPTION = "转换异常";
33.
34.     public static final String IO_EXCEPTION = "io异常";
35.
36.     public static String httpPost(String url, String data) {
37.         LoggerUtil.info(data);
38.
39.         CloseableHttpClient httpClient = HttpClientBuilder.create().build();
40.         CloseableHttpResponse response = null;
41.
42.         String result = ReturnCode.FAIL.getMessage();
43.         try {
44.             HttpPost post = new HttpPost(url);
45.             post.setHeader(HttpHeaders.CONTENT_TYPE, CONTENT_TYPE_VALUE);
46.             StringEntity entity = new StringEntity(data, ContentType.APPLICATION_JSON);
47.             entity.setContentType(CONTENT_TYPE_VALUE);
48.             post.setEntity(entity);
49.
50.             response = httpClient.execute(post);
51.             if (response.getStatusLine().getStatusCode() == HttpStatus.SC_OK) {
52.                 // 得到响应体
53.                 result = EntityUtils.toString(response.getEntity(), Consts.UTF_8);
54.             } else {
55.                 // 返回http失败
56.                 result = response.getStatusLine().getReasonPhrase();
57.             }
58.         } catch (ParseException e) {
59.             result = PARSE_EXCEPTION;
60.             LoggerUtil.error(PARSE_EXCEPTION, e);
61.         } catch (IOException e) {
62.             result = IO_EXCEPTION;
63.             LoggerUtil.error(IO_EXCEPTION, e);
64.         } finally {
65.             // 消耗实体内容
```



- 非标类系统
- 持续集成平台自动化部署





- [Back to aiInterviewer](#)

AIInterviewer Test Report

AIInterviewer Report title





Date report: 2019/01/18 14:01

Designed for use with [JMeter](#) and

Summary

# Samples	Failures	Success Rate	Average Time	Min Time	Max Time
63	0	100.00%	17 ms	2 ms	123 ms

Pages

URL	# Samples	Failures	Success Rate	Average Time	Min Time	Max Time	
服务检测	1	0	100.00%	120 ms	120 ms	120 ms	 expand/collapse
获取模型功能描述	3	0	100.00%	118 ms	114 ms	123 ms	 expand/collapse
查询模型能力点	3	0	100.00%	6 ms	3 ms	8 ms	 expand/collapse
查询题目详情	3	0	100.00%	10 ms	8 ms	12 ms	 expand/collapse

3

PART

部门的持续集成建设情况











持续集成平台 ---- 邮件通知构建结果



2019/5/3 (周五) 6:56
持续集成-
收件人
抄送

工程名字: ARK-rgp
构建 ID: 57
项目目录: rgp/rgp_j2ee

item	url	result
Jenkins job		pass
单元测试		pass
覆盖率		coverage too low: 44%
sonar 扫描		vulnerabilities : 4
持续集成数据 统计		NA



项目接入持续集成步骤

- 1.开发提供工程svn/git路径
- 2.开发接口人添加持续集成UM账号拉取代码的权限
- 3.开发修改工程的pom文件，添加jacoco、antrun插件
- 4.开发提供部署信息（jar包部署命令及环境信息），非标系统 ---- 可选
- 5.测试团队创建jenkins工程，定制pipeline脚本



平安寿险AI团队 平安寿险AI团队 平安寿险AI团队 平安寿险AI团队 平安寿险AI团队 平安寿险AI团队 平安寿险

4

PART

持续集成后续规划



借助神兵部署流水线来部署CI环境（平安科技提供接口供调用，沟通中）

自动化回归测试用例代码覆盖率

持续集成过程的数据可视化（构建失败率、单测试用例执行失败率等）反推问题改进

将各项数据作为发版评审的依据。。。

THANK YOU