DOIS | 2018 · 深圳站

DevOps 落地，从这里开始

DevOps 国际峰会

暨 DevOps 金融峰会

指导单位：云计算开源产业联盟 OpenSource Cloud Alliance for IndustRy, OSCAR

主办单位：DevOps时代　高效运维社区 GreatOPS Community

时间：2018年11月2日-3日

地址：深圳市南山区圣淘沙大酒店（翡翠店）

DOIS

浙江移动的DevOps实践

中国移动浙江公司　　郑海朋

DOIS

目录

1 浙江移动的成熟度结果

2 搭建弹性高可用的构建环境

3 代码质量检查提前到开发阶段

4 安全高效的应用部署

5 小结和思考

研发运营一体化成熟度评估是什么

DOIS

研发运营一体化能力成熟度模式															
能力类	一、研发运营一体化（DevOps）过程														
能力域	开发管理			持续交付							技术运营				
能力子域	价值交付管理	敏捷过程管理	敏捷组织模式	配置管理	构建与持续集成	测试管理	部署与发布管理	环境管理	数据管理	度量与反馈	监控服务	数据服务	容量服务	连续性服务	运营反馈
能力项	需求工件	价值流	敏捷角色	版本控制	构建实践	测试分级策略	部署与发布模式	环境供给方式	测试数据管理	度量指标	应用监控	数据收集能力	容量规划能力	高可用规划	业务知识管理
	需求活动	仪式活动	团队结构	版本可追溯性	持续集成	代码质量管理	持续部署流水线	环境一致性	数据变更管理	度量驱动改进	质量体系管理	数据处理能力	容量平台服务	质量体系管理	项目管理
						测试自动化					事件响应及处理	数据告警能力	运营成本管理		业务连续性管理
											监控平台				运营服务管理
能力类	二、研发运营一体化（DevOps）应用架构														
能力类	三、研发运营一体化（DevOps）安全管理														
能力类	四、研发运营一体化（DevOps）组织机构														

一级	•初始级	在组织局部范围内开始尝试DevOps活动并获得初期效果
二级	•基础级	在组织较大范围内推行DevOps实践并获得局部效率提升
三级	•全面级	在组织内全面推行DevOps实践并贯穿软件全生命周期获得整体效率提升
四级	•优秀级	在组织内全面落地DevOps并可按需交付用户价值达到整体效率最优化
五级	•卓越级	在组织内全面形成持续改进的文化并不断驱动DevOps在更大范围内取得成功

持续交付成熟评估

DOIS

持续交付						
配置管理	构建与持续集成	测试管理	部署与发布管理	环境管理	数据管理	度量与反馈
版本管理	构建实践	测试分层管理	部署与发布模式	环境管理	测试数据管理	度量指标
变更管理	持续集成	代码质量管理	持续部署流水线		数据变更管理	度量驱动改进
		自动化测试				

持续交付能力成熟度评估根据配置管理、构建与持续集成、测试管理、部署与发布管理、环境管理、数据管理、度量与反馈等**七大领域，十四个维度**进行。

评估方式

在能力子项目里，对每个级别做了明确的要求

持续部署流水线				
级别	构建方式	构建环境	构建计划	构建职责
卓越级	持续优化的构建服务平台，持续改进服务易用性	持续改进构建性能，实现构建资源动态按需分配回收，如搭建基于云服务虚拟化和容器化的分布式构建集群	分级构建计划，实现按需构建并达到资源和速度的有效平衡	将构建能力赋予全部团队成员，并按需触发构建实现快速反馈
优秀级	实现构建服务化，可按需提供接口和用户界面用于可视化构建编排	优化构建速度，实现增量化构建和模块化构建，如可采用分布式构建集群、构建缓存等技术，实现构建资源的共享	分级构建计划，实现按需构建并达到资源和速度的有效平衡	构建系统服务化提供更多用户使用，构建不再局限于专业团队进行
全部级	定义结构化构建脚本，实现模块级共享复用和统一维护	构建环境配置实现标准化，有独立的构建集群	明确定义构建计划和规则，实现代码提交触发构建和定期自动执行构建	构建工具和环境由专门团队维护，并细分团队人员职责
基础级	实现脚本自动化，通过手工配置完成构建	有独立的构建服务器，多种任务共享构建环境	明确定义版本号规则，并根据发布策略细分构建类型，实现每日自动构建	构建工具和环境由专人负责维护，并使用权限隔离
初始级	采用手工方式进行构建，构建过程不可重复	使用本地设备，构建环境不可靠	没有明确的版本号规则和构建任务计划	构建工具和环境受限于团队人员能力，频繁手动干预维护

评估的方式：
人员访谈、材料审查、模式演示

指方向：跃迁需要做什么

查补缺：哪些实践必是必须的

定基准：确定最快、最好效率的初步行动

评估结果

DOIS

能力域	能力子域	能力项	能力指标项	能力得分	评级
持续交付	配置管理	版本控制	版本控制系统 分支管理 制品管理 单一可信数据源	三级	三级
		变更管理	变更过程 变更追溯 变更回滚		
	构建与持续集成	构建实践	构建方式 构建环境 构建计划 构建职责	四级	
		持续集成	集成服务 集成频率 集成方式 反馈周期		
	测试管理	测试分级策略	分层方法 分层策略 测试时机	四级	
		代码质量管理	质量规约 检查方式 反馈处理		
		测试自动化	自动化设计 自动化开发 自动化执行 自动化分析		
	部署与发布管理	部署与发布模式	部署方式 部分过程 部署策略 部署质量	三级	
		持续部署流水线	协作模式 流水线过程 过程可视化		
	环境管理	环境管理	环境类型 环境构建 环境依赖与配置管理	四级	
	数据管理	测试数据管理	数据来源 数据覆盖 数据独立性	三级	
		数据变更管理	变更过程 兼容回滚 数据监控		
	度量与反馈	度量指标	度量指标定义 度量指标类型 度量数据管理 度量指标更新	三级	
		度量驱动改进	内容和生产方式 数据和生产方式 数据失效性 覆盖范围 反馈改进		

效率和质量提升点

优势点

√构建和持续集成
- 构建脚本版本管理
- 构建环境容器化
- 构建资源弹性高可用

√代码质量管理
- pre-commit检查代码

环境管理
- 容器的大规模应用

√持续部署流水线
- Jar包发布
- 数据与代码部署分离

变更管理
- 变更项多，触发来源广

构建和部署要解决的问题

DOIS

代码构建关注软件代码到可运行程序之间的过程，通过规则、资源和工具的有效结合，提升构建质量和构建速度，使构建成为一个轻量级，可靠可重复的过程。

部署和发布模式关注交付过程中的具体实践， 将部署活动自动化并前移到研发阶段， 通过频繁的演练和实践部署活动，成为研发日常工作的一部分，可靠、可重复的完成部署发布任务。

人	流程	技术
构建和部署活动参与的角色和职责 • 个人，全栈工程师 • 团队，人员合理地安排在一起，合理地组织起来协作	• 代码提交流程，Code Review • 代码构建流程，持续集成流水线 • 应用发布流程，部署流水线线	• 源码、依赖包、制品的管理工具和方式 • 代码质量检查和漏洞分析的策略和工具 • 构建和部署使用的工具 • 能力和容量

打造DevOps工具链平台支撑应用构建部署

DOIS

目录

1 浙江移动的成熟度结果

2 搭建弹性高可用的构建环境

3 代码质量检查提前到开发阶段

4 安全高效的应用部署

5 小结和思考

痛点：应用接入平台构建的困难

DOIS

用户的困难

- 脚本管理，没有构建脚本，或者构建脚本是个黑盒
- 职责调整，管理构建的职责集中在BM
- 异常处理，使用平台后构建过程中出错谁负责

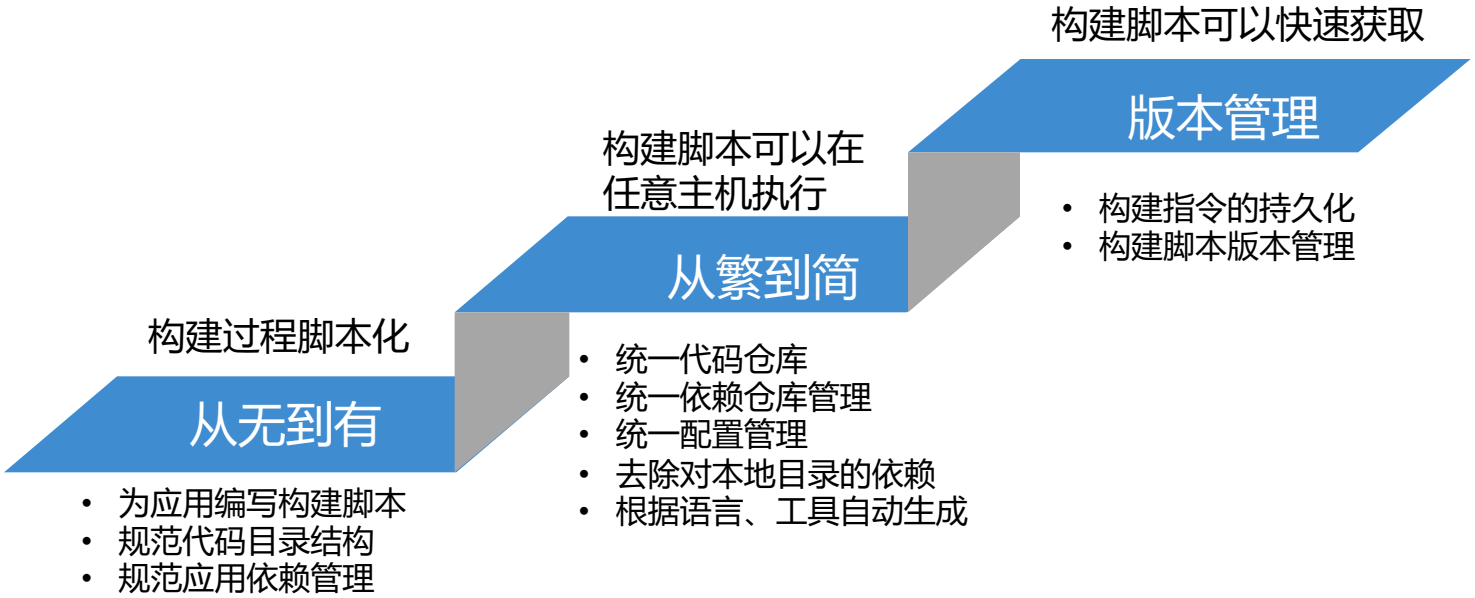
平台的困难

- 团队的差异，代码、依赖、配置的管理工具和模式不一样
- 工具的差异，应用的差异性很大，语言、版本、工具都不相同
- 规模，应用接入的速度太多，构建资源和管理模式跟不上

举措一：三步提升构建脚本管理

DOIS

应用编写构建脚本，实现构建过程脚本化管理。去除脚本对构建主机本地资源的依赖，统一配置和依赖的管理，标准化的应用能够自动生成构建指令。构建脚本版本管理，脚本快速获取，变更过程可以追溯。



脚本管理的目标：脚本可以在任意构建主机构建应用

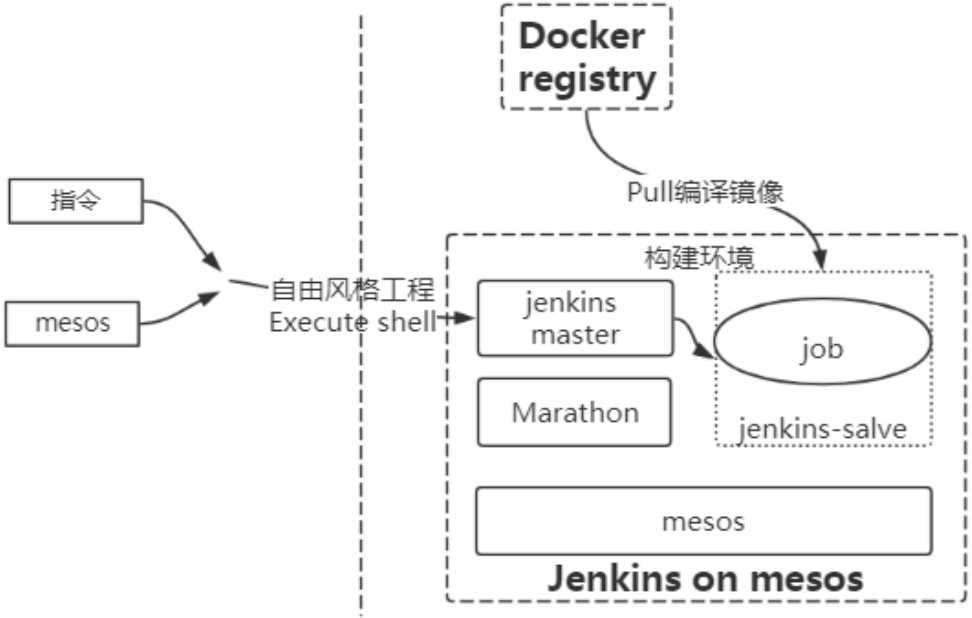
举措二：容器化构建环境

构建环境容器化，通过容器封装不同语言、工具导致构建环境的差异，通过容器规格划分和弹性提升构建环境的资源利用率。

屏蔽编译环境差异：不同的容器包含不同的编译工具和环境配置。

提升构建主机利用率：在同一个构建主机上启动多个容器，提升主机资源的利用率。

构建任务配置：编译工具和资源诉求分配对应的mesos标签



举措三：搭建弹性高可用的构建环境

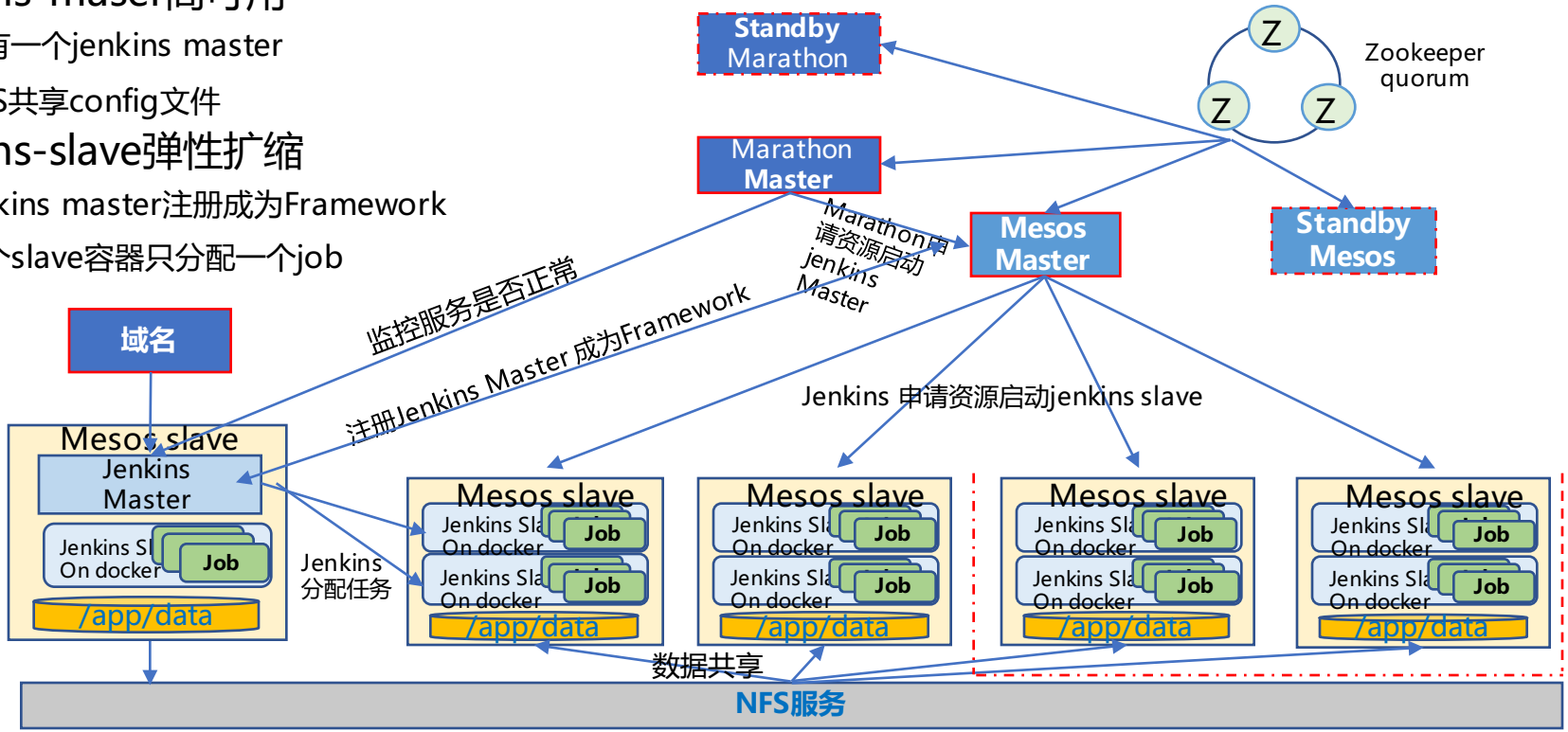
DOIS

Jenkins-maser高可用

- 只有一个jenkins master
- NFS共享config文件

Jenkins-slave弹性扩缩

- jenkins master注册成为Framework
- 一个slave容器只分配一个job

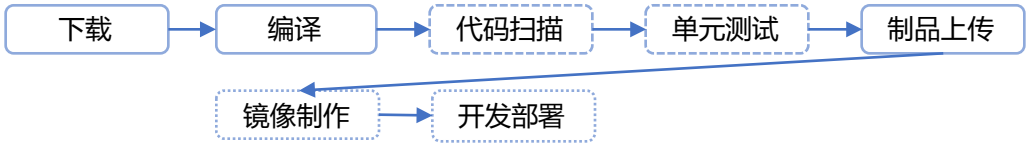


举措四：划分构建流水线类型明确职责

DOIS

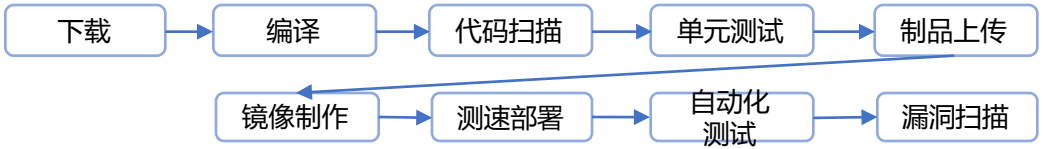
团队内部不同角色负责对应类型的流水线

开发流水线



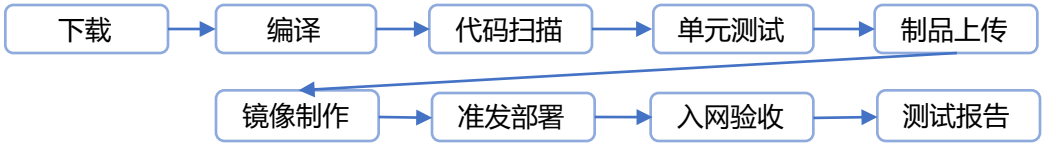
作用：快速集成\开发联调
角色：开发
时间：
代码提交，触发jenkins自动构建
手工触发代码构建，进行开发联调

测试流水线



作用：测试验证
角色：测试，开发
时间：
定时构建，特定时间点发布测试环境
代码合并、手工触发，验证新功能

准发流水线



功能：上线前验证
角色：BM，运维
时间：
手工触发，迭代结束前生成上线发布版本

通过代码质量管理提升代码构建的成功率，减少流水线中断

DOIS

目录

1 浙江移动的成熟度结果

2 搭建弹性高可用的构建环境

3 代码质量检查提前到开发阶段

4 安全高效的应用部署

5 小结和思考

痛点：代码质量检测难以推行

DOIS

代码质量管理是在软件研发过程中保证代码质量的一种机制， 当代码变更后， 可以对代码质量进行检查、分析,给出结论和改进建议，对代码质量数据进行管理，并可以对代码质量进行追溯。

质量规约

- 不同厂家的开发标准不统一
- 规范无法落地，成为一堆纸上文字

检查方式

全员Code Review

- 不同的人对规范的理解不一样
- 全员方式占用太多的时间
- 提升速度太慢，新人从头开始

反馈处理

- 技术债务没有能及时处理
- 历史的技术债务太大无法偿还，大部分团队没有勇气去解决历史账务

Sonar扫描代码

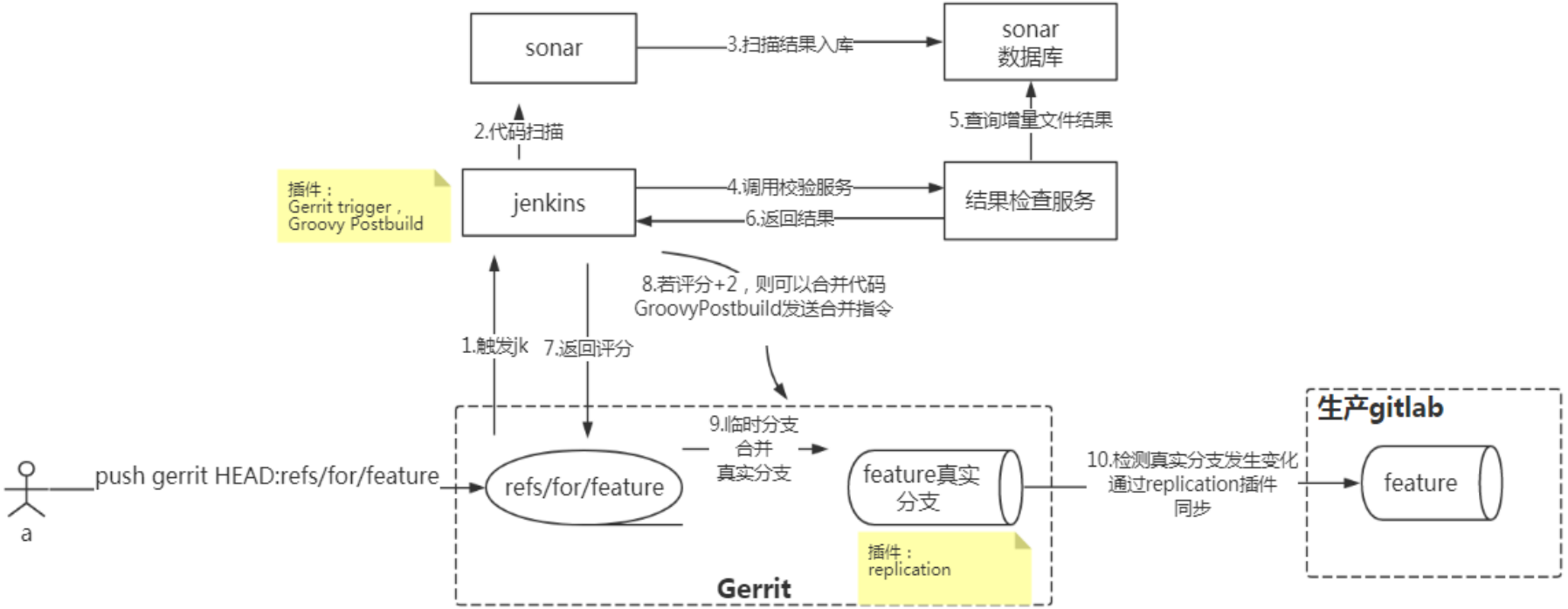
能解决的问题
规则落地：所有的规则和约定，通过代码规则配置到soanr。在构建的流水线中对代码规则检查。对质量问题进行通报和跟踪。
代码质量可视化：通过质量报表展示个应从复杂度分布、重复代码、单元测试统计、技术债务等维度可视化各应用代码情况和变化趋势。

不能解决的问题：
流水线中断：流水线构建成功率没有提升。
技术债务：应用的历史技术债务并没有随着减少。

团队或项目组	级别	规则	次数
研发一组	BLOCKER	SQL binding mechanisms should be used	12
研发二组	BLOCKER	Null pointers should not be dereferenced	6
订单中心	BLOCKER	Null pointers should not be dereferenced	4
账号中心	BLOCKER	Null pointers should not be dereferenced	2
到期子中心	BLOCKER	Null pointers should not be dereferenced	1

改进：引入Gerrit工具对代码进行review（1）

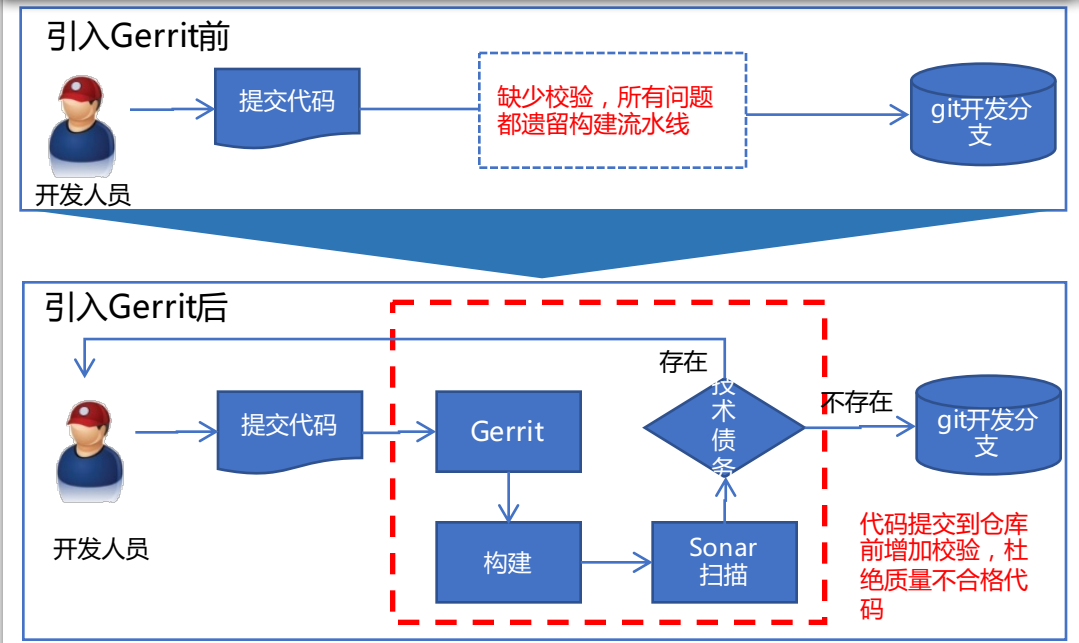
引入Gerrit工具进行代码自动review，将代码缺陷前置到开发阶段解决



改进：引入Gerrit工具对代码进行review（2）

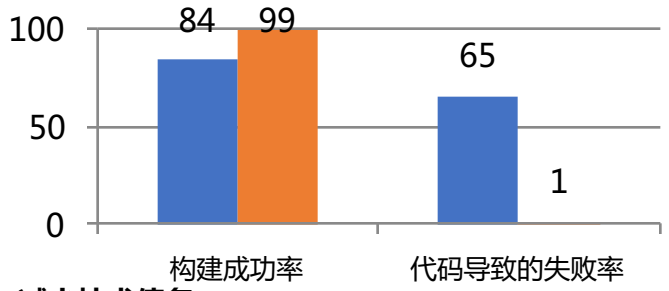
DOIS

使用Gerrit前置解决代码缺陷

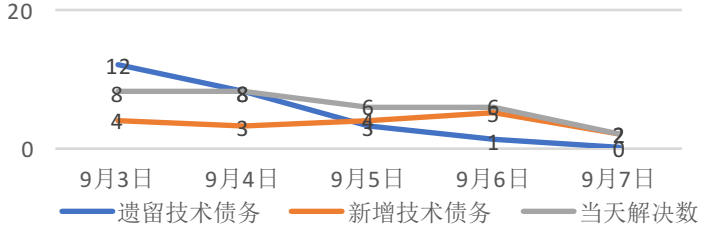


实施效果

●提高构建成功率



●减少技术债务



DOIS

目录

1 浙江移动的成熟度结果

2 搭建弹性高可用的构建环境

3 代码质量检查提前到开发阶段

4 安全高效的应用部署

5 小结和思考

痛点：应用接入平台部署的困难

DOIS

用户的困难

- 交付物多：一次上线交付物众多，版本难以对齐
- 流程复杂：上线流程复杂，参与的角色众多
- 频率高：微服务改造、交付频率提升，导致交付压力剧增

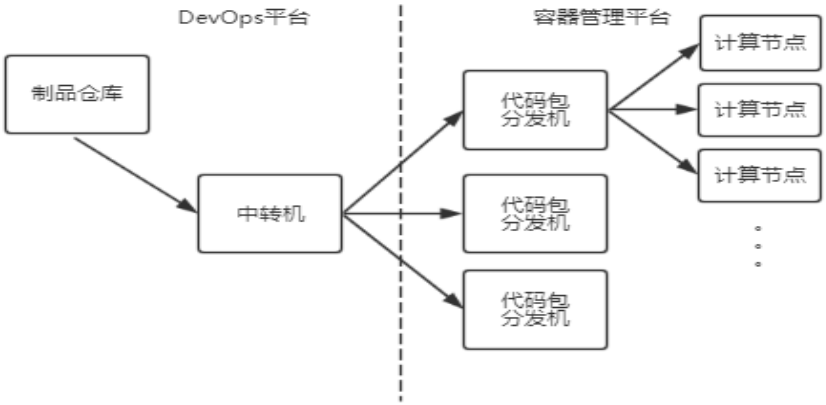
平台的困难

- 灵活：不同管控级别的项目流程不一致
- 高并发：集中化部署模式，大量集群的部署统一在同一时刻
- 复杂度：涉及的平台众多，验收平台、缓存中心、配置中心、微服务管控平台、告警平台。

代码包部署流程及存在问题

DOIS

代码包交付的部署流程



部署流程说明：
1、交付平台将代码包下载到指定的服务器目录
2、进程将服务器上的代码包上传到代码包分发服务器；
3、进程通知所有的计算节点到代码包分发服务器拉取代码包；
4、容器加载代码包并启动应用

问题：

可用性差：分发环节过多造成发布可用性差，容易发生目录不对，文件覆盖、传包失败、分发主机僵。

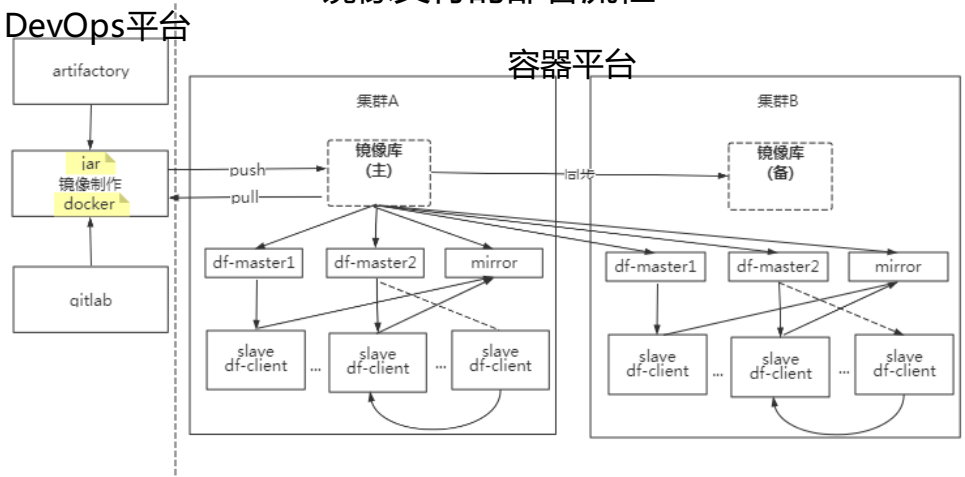
性能瓶颈： 代码分发服务器模式可扩展性差，由于计算节点数众多，同时进行多个代码包发布，代码分发服务器压力大容易引发节点拉包超时、失败。

无法提前预发布：代码包发布模式在计算节点上只能保留一个版本，无法进行预发布来减少发布窗口时间。

版本不一致：发布过程中部分计算节点异常（主机Down、Docker hang）或是计算节点维修错过发布后重新被加入集群，会导致代码包版本不一致问题。

改进一：镜像交付

镜像交付的部署流程



部署流程说明：

1、交付平台将代码包制作成应用镜像，并推送的镜像仓库

2、容器管理平台将镜像通过P2P分下载到使用镜像的主机。

提升：

架构简单：分发环节只有镜像库这个交互点，解决原先发布流程复杂带来可用性差的问题；

消除性能瓶颈： P2P分发模式只会访问一次镜像库，分发在计算节点内部通过P2P方式完成。

版本一致：Docker本身的镜像机制来保证整个过程镜像的一致性，计算节点异常（主机Down、Docker hang），不会拉起不一致版本的镜像，而是会在其他节点重新调度；

快速预热回退：计算节点客户端可以保留多个镜像版本，应用回退更快捷。应用实例替换前可以通过预热分发镜像，在发布窗口只需要重启应用就完成发布。

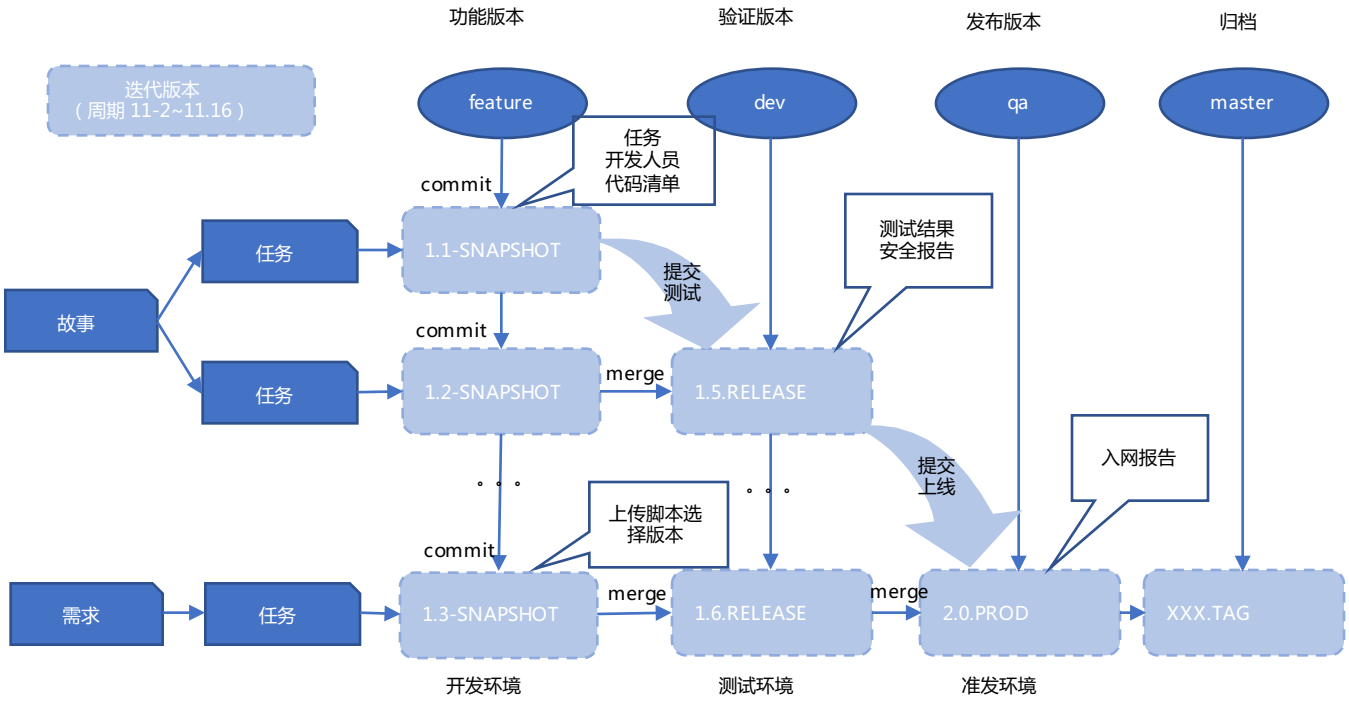
改进二、交付物版本对齐

建立统一的迭代版本，迭代周期内所有的交付物都归属该版本

需求都拆解为开发任务，并依据上线时间与迭代版本关联

约定迭代周期内的版本与分支的对应关系，每一次构建都生成子版本

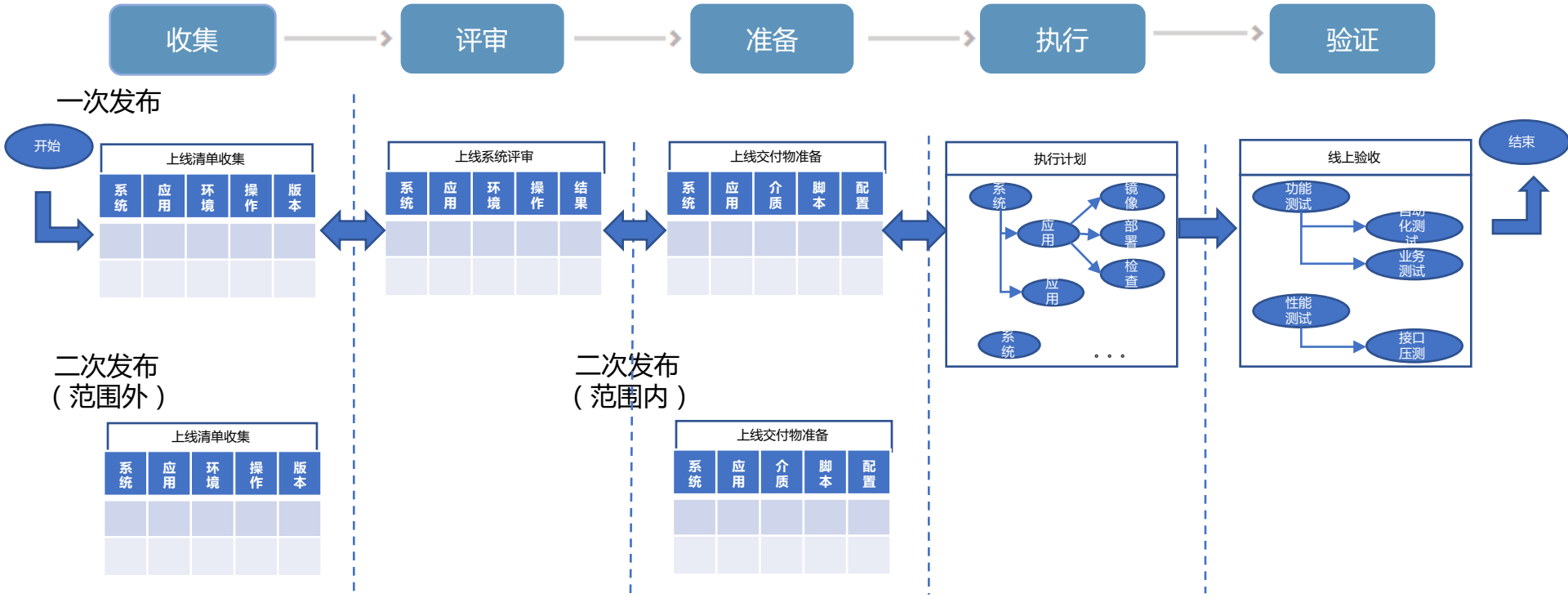
非代码类交付物与对应的子版本绑定



改进三：跨系统多应用部署流水线

DOIS

通过发布清单自动化生成发布计划，集中上线模式，一次发布多个应用。



DOIS

目录

1 浙江移动的成熟度结果

2 搭建弹性高可用的构建环境

3 代码质量检查提前到开发阶段

4 安全高效的应用部署

5 小结和思考

小结

DOIS

构建实践	• 资源和工具的有效结合，提升构建质量和构建速度，使构建成为一个轻量级，可靠可重复的过程。
代码质量	• 通过对代码质量进行检查、分析,给出结论和改进建议，提升构建的速度和成功率。
部署和发布	• 交付物版本对齐，部署过程可灵活编排响应业务需求

评估的收益

DOIS

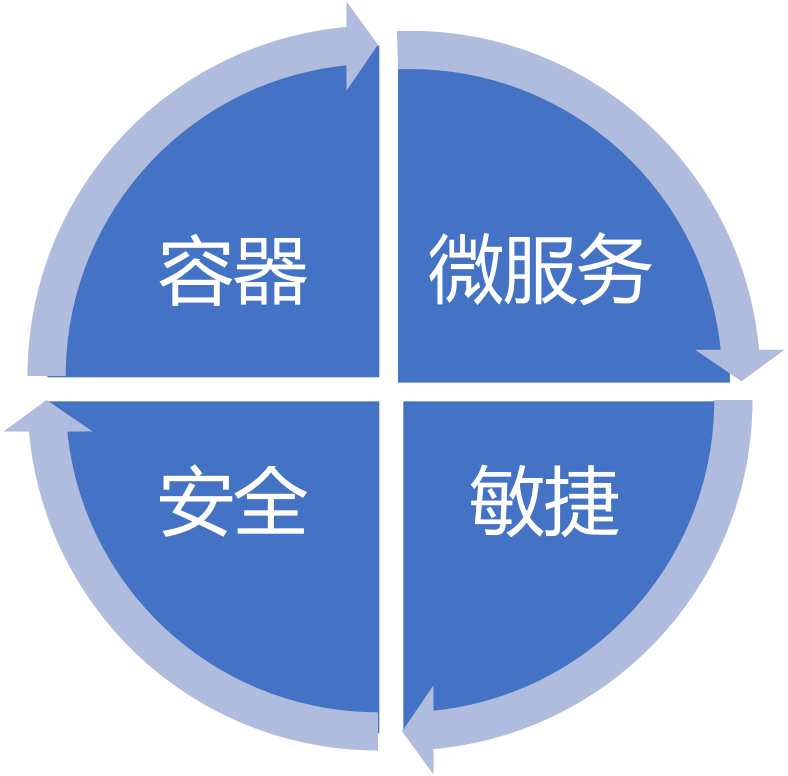
自省

- 持续交付的核心就在于着眼全局
- 非瓶颈的效率提高对提升有效产出没有作用
- 不完整的DevOps实践阻碍着DevOps的发展

参照

- 摸着石头过河，走了无数的弯路
- 未来在哪里，我们还要走多远的路

通往成熟度三级的捷径



欢迎沟通

DOIS

团队公号

个人微信号

DOIS

Thanks

DevOps 时代社区 荣誉出品

想第一时间看到高效运维社区的新动态吗？