



GOPS 2021
Shenzhen

GOPS

全球运维大会

2021
-XOPS 风向标



深圳站

中国·深圳

指导单位：



主办单位：



时间：2021年5月21日-22日

腾讯基于Git的工程研发实践

李凯凯 腾讯工蜂架构师



李凯凯 (lekkoli)

腾讯  工蜂 架构师



软件代码工具Oteam团队成员

腾讯CI-Oteam协作成员

PCG-研发效能EP度量协作成员

IEG-蓝盾DevOps协作成员

目录

CONTENTS

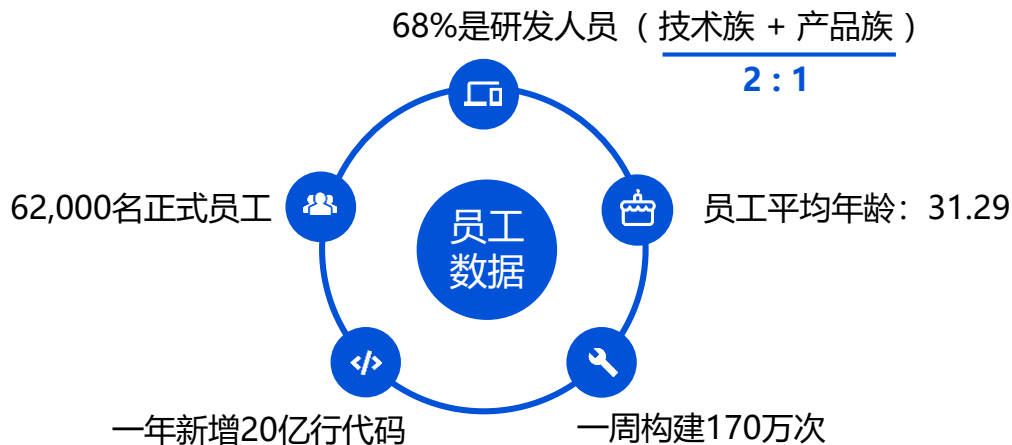
- 1 腾讯项目研发遇到的挑战
- 2 一个“单一仓库+主干开发”的例子
- 3 Git研发畅想

挑战

各具业务特色的研发模式

01

项目研发

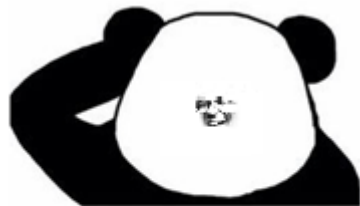


协作成本

谁能帮忙看一下这个问题[贴截图]? 应用起不来

本地可以运行, 为什么发到服务器就报错呢...

我的代码被谁覆盖了!!!



真让人头大

项目成员 161

项目组成员 1,105

协作项目组 2

不要在**master**分支直接提交代码!

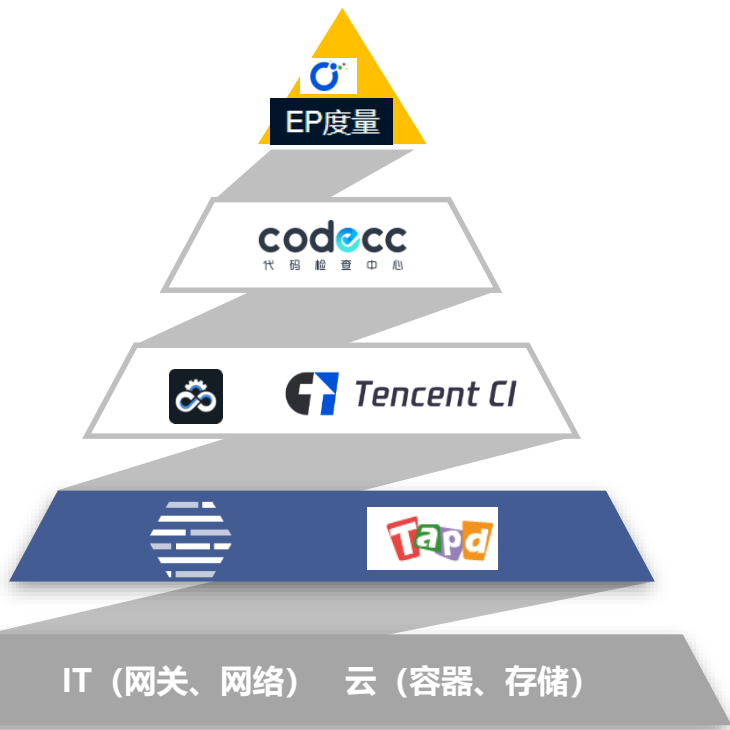
是不是谁改了基础包, 怎么**pull**代码后就跑不起来了???

后端接口是不是被改了, 怎么现在调不通了

是不是谁有代码没提交啊, 本地编译一堆报错。。。

这个地方为什么这样改啊? 构建失败了

研效基础设施



团队研效状态 (持续集成)

需要工蜂的CR/MR辅助功能

1 刀耕火种

每天跑一次CI

- 没压力

2 自动化

每提交代码跑一次CI

- 开放平台接口能力
- 分布式集群并发能力

3 DevOps
3级标配

每提交代码跑一次CI
做代码红线质量准入

- 开放平台三方认证体系
- 质量红线关卡
- MR显示扫描结果
- 代码浅克隆
- 大规模单集群并发能力

Tencent

4 EPC
试点项目
(2020)

每提交代码后
做合并预检测
做代码红线质量准入

5 主干开发

??????

腾讯业务团队想要的级别

基于Git系统的研发模式

- 组织项目
- 分支策略
- 代码评审
- 持续集成

单元测试、依赖管理、特性开关、构建加速、研发度量




案例

单一仓库 + 主干开发

02

腾讯某核心业务

- 
- 年收入占腾讯财报近20%
 - 提倡代码共有，人人都可以修改代码库中的任何代码
 - 有代码修改必须经过代码评审
 - 具备高素质技术人员投入研效建设

项目组织



Multi Repo ?

or

Mono Repo ?

or

Virtual Mono Repo?

Mono Repo

一个仓库，多个项目

- 主要特征

- 代码集中管理
- 权限全局分配
- 使用目录来分割项目

中小团队：爽！

大型团队：？

Main-Proj

```
.....  
/Code-Android-Proj  
/Code-Iphone-Proj  
/Code-Web-Proj  
/Code-API-Proj  
/backend/Code-Server-Proj  
/backend/Code-Image-Proj  
/backend/Code-Video-Proj  
/plugin/Code-Monitor-Proj  
/plugin/Code-Statistic-Proj  
.....
```



86TB Piper + GitC

? TB

 工蜂

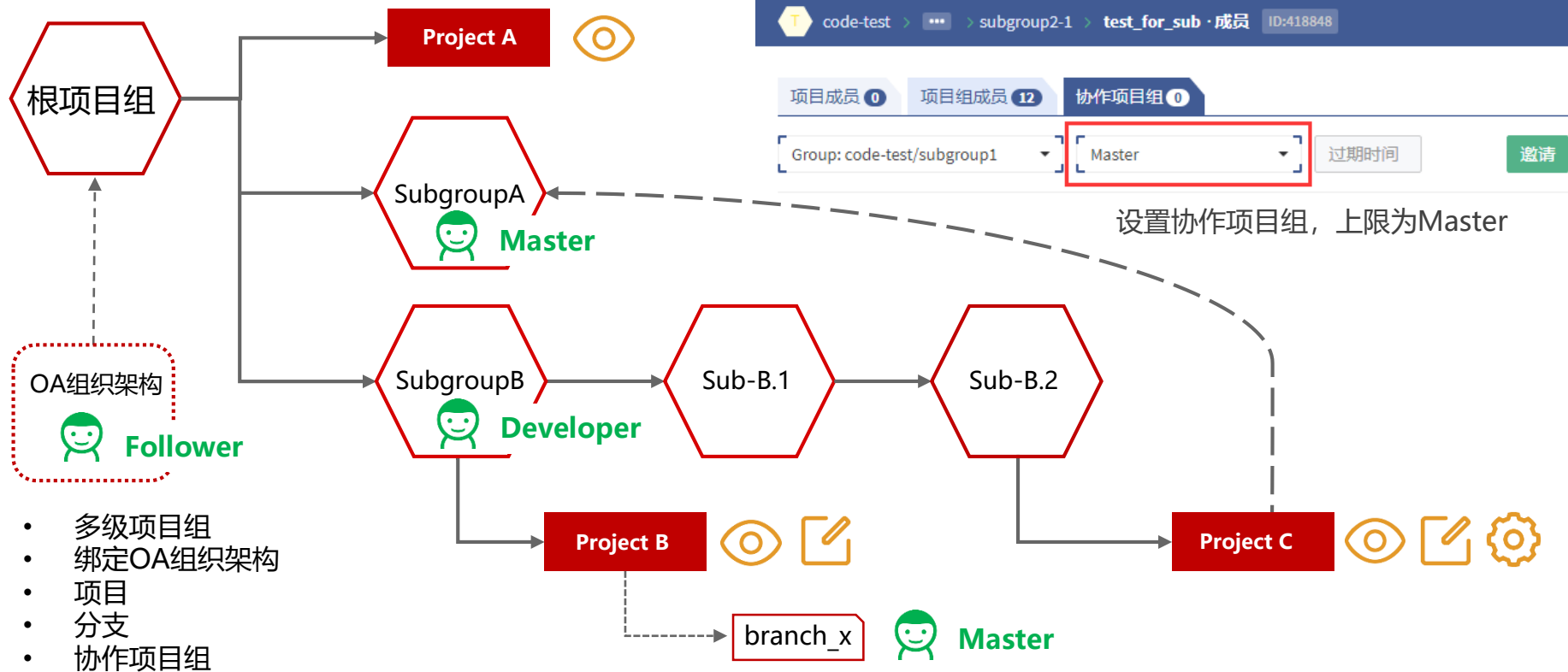


Mercurial+



Git (custom build)

仓库权限模型



项目组织方式

	Multi Repo	Mono Repo
管理模式	团队自行管理、专人管理	专人管理，固化研发流程
权限分配	项目间、项目内权限划分	单项目内人员划分
开发模式	独立项目开发、依赖项目集成测试	集中项目开发、测试
分支管理	独立项目分支	单项目分支
优势	<ul style="list-style-type: none"> 独立服务，易于上手 更容易控制成员权限，对应到负责人 仓库大小拆分后更小，按需拉取 灵活的模块拆分，项目级解耦，减少代码污染 异构技术栈项目支持良好 	<ul style="list-style-type: none"> 项目完整，减少本地依赖处理耗时 所有变更历史在一个项目内可查 全局的访问权限，代码更易复用、重构 更容易控制代码规范 更容易保持统一的代码版本
劣势	<ul style="list-style-type: none"> 容易导致项目泛滥，不利于管理 依赖关系维护成本增加，版本管理困难 开发沟通成本增加 	<ul style="list-style-type: none"> 巨型项目拖跨整个研发工具流程 权限控制难以细分，存在安全风险 容易导致分支泛滥，合并冲突（可考虑使用主干开发）

单一大仓，它做了什么？

千人权限配置

- 机器人账号为Master
- 所有人仅为Reporter
 - 使用工具创建远端个人分支
 - 通过保护分支权限开发

开发顺畅度

- 使用CLI命令式交互
 - 命令行建立个人分支、发起MR、同步代码
- 代码评审完毕自动合并MR
 - 评审人配置精细到目录
 - 通过Web Hook，调用接口合入

仓库性能

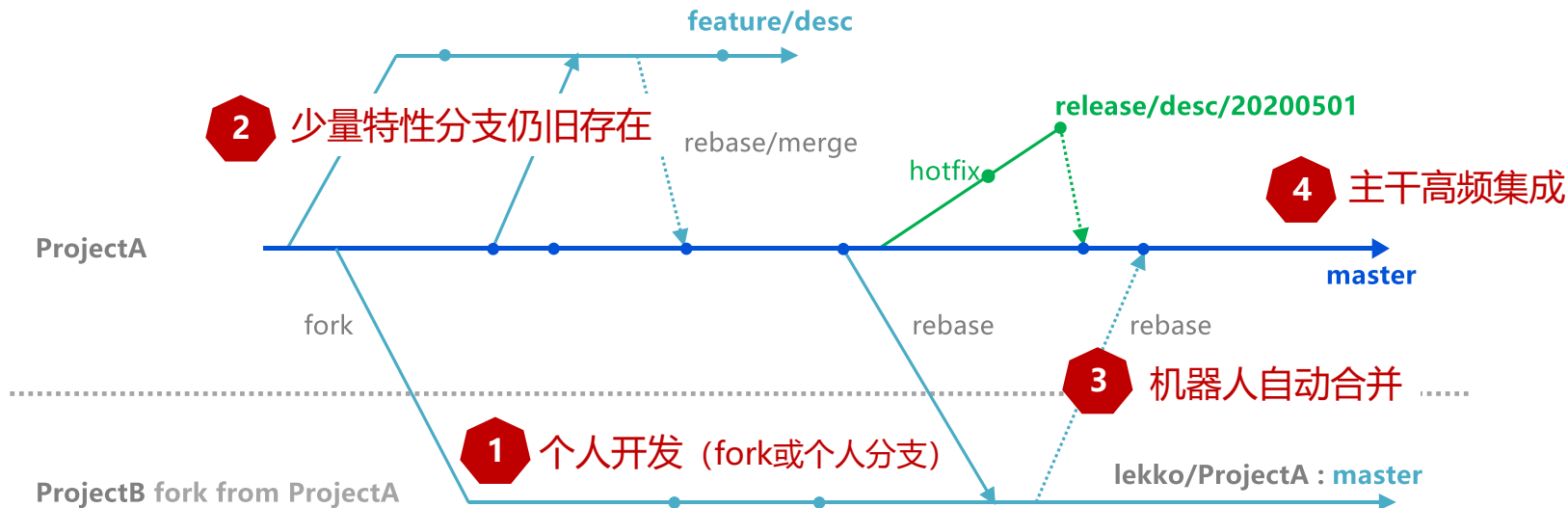
- 重做仓库，并截断历史
 - 10W+ Revision → 4W Commit
- 仓库强制大文件LFS托管
 - Git仓库限制单个文件<16M
 - Git LFS单个文件<512M
- 降低分支存活时间
 - 半年19W个分支被关闭
 - 分支平均周期仅持续5天
- 工蜂侧支持一主N从访问
 - 高频API: diff、blame、compare
 - Git客户端clone、pull代码

仓库分支策略



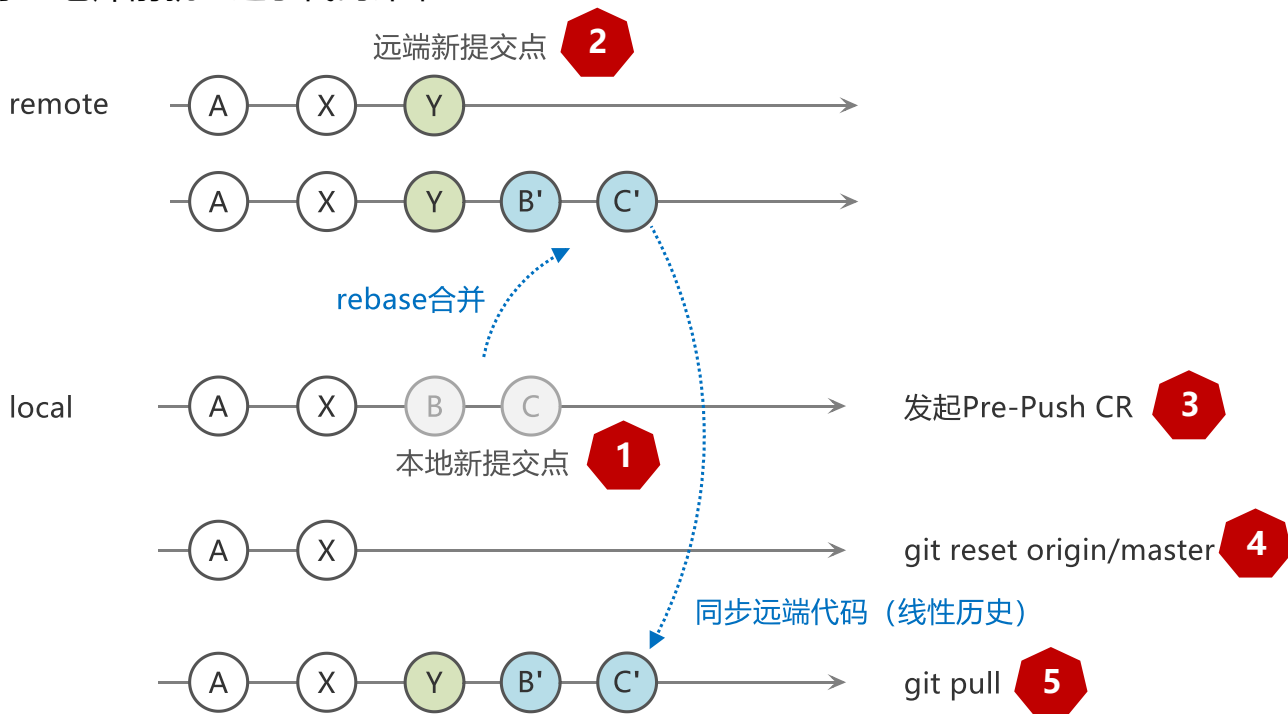
主干开发，它做了什么？

- 主干是大家交换代码的核心分支
- 代码频繁地在主干上进行同步
- 发布时从主干拉取Release发支，主干本身不稳定
- 有大量短期的独立开发分支（功能分支、个人分支、个人fork项目）

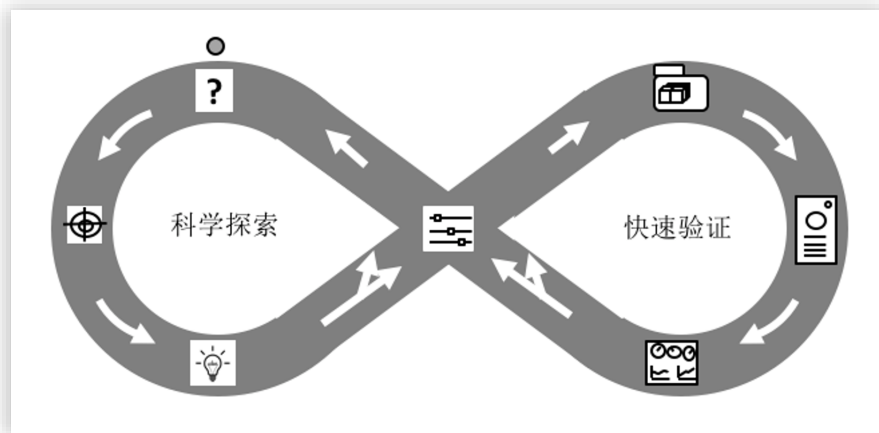


线性历史的主干开发

- 任何一个commit在push到Git仓库前就经过了代码评审
- 仅有一个master分支
- 消灭merge点，线性历史
- 特性开关管理配套系统



不确定 → 确定



代码评审，它做了什么？

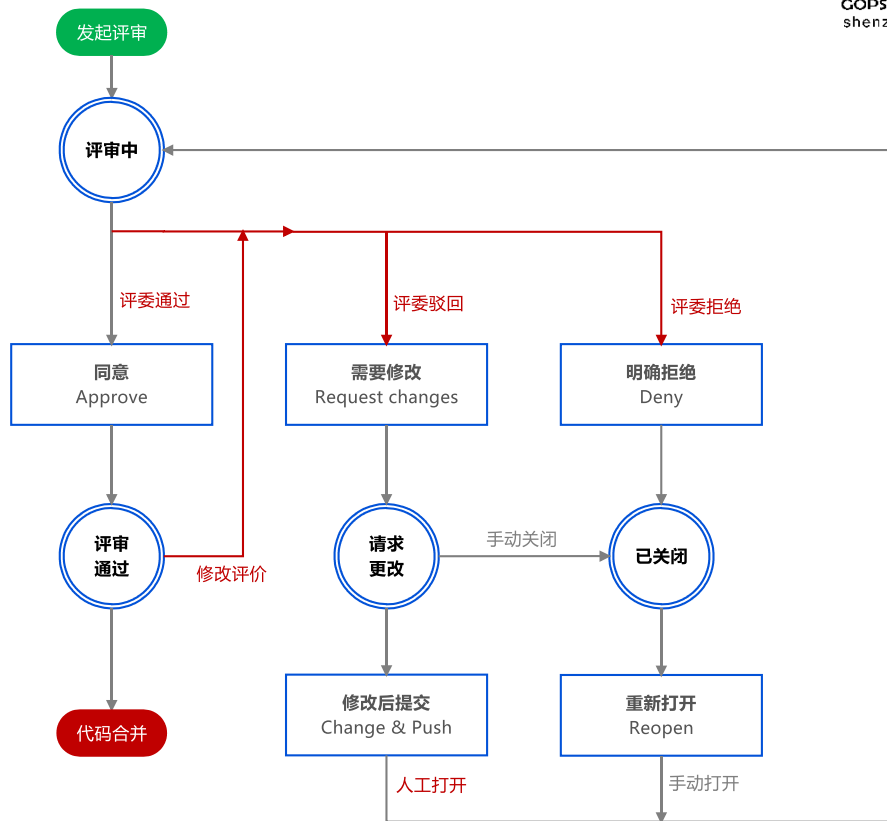
所有的代码都必须经过评审

前置措施

- 没有人可以直接push主干
- 大部分人都在个人分支下开发
- **只有机器人账号可以合并代码**

CR介入时机

- **分支合并时评审**
(Pull Request、Merge Request)
- 历史Commit范围评审
- Push前评审 (Pre-Push)



精细到路径级别

FileOwner文件评审规则 (通过仓库内 “.code.yml” 文件配置单个文件的评审负责人)



The screenshot displays the FileOwner file review interface. At the top, it shows the selection of 4 commits (04eed6b2 and dc30d1bc) and the file path .code.yml. The main area lists files for review, including test.log, README.md, and .code.yml, with their respective review counts. A modal window titled '文件评审中' (File Reviewing) is open, showing the review rules (1 reviewer agreement) and the list of reviewers (lekkoli, git_helper_01). The right sidebar shows the list of reviewers (git_helper_05) and the file owner (lekkoli, git_helper_01).

共选择了4个提交

从 04eed6b2 [xxx] 到 dc30d1bc [.code.yml]

3 Files 6 1 0/3 文件已阅 请输入关键字

我负责的文件

test.log +1 -0 lekkoli; git_helper_01;

其他文件

README.md +1 -1

.code.yml +4 -0

会话 3 变更 3 提交 4

显示全部信息 只显示会话信息

文件评审中

规则: 1位负责人同意则文件通过

lekkoli

git_helper_01

添加文件负责人

同意通过 确认拒绝

评审人

新增评审人

git_helper_05 移除

文件负责人

git_helper... Todo 0 0 0 1

lekkoli Todo 0 0 0 1

TAPD

暂无关联TAPD单

参与人 3

标签 编辑

典型场景

$\text{每CR} \begin{array}{l} + 4\text{个评审人} \\ + 20\text{个文件负责人} \end{array} \times \text{每天 } 300\text{个CR}$	
每人每天要看	<div>6~20单 CR评审</div> <div>600~2000行 代码</div>



无尽的人工？

高频集成，它做了什么？

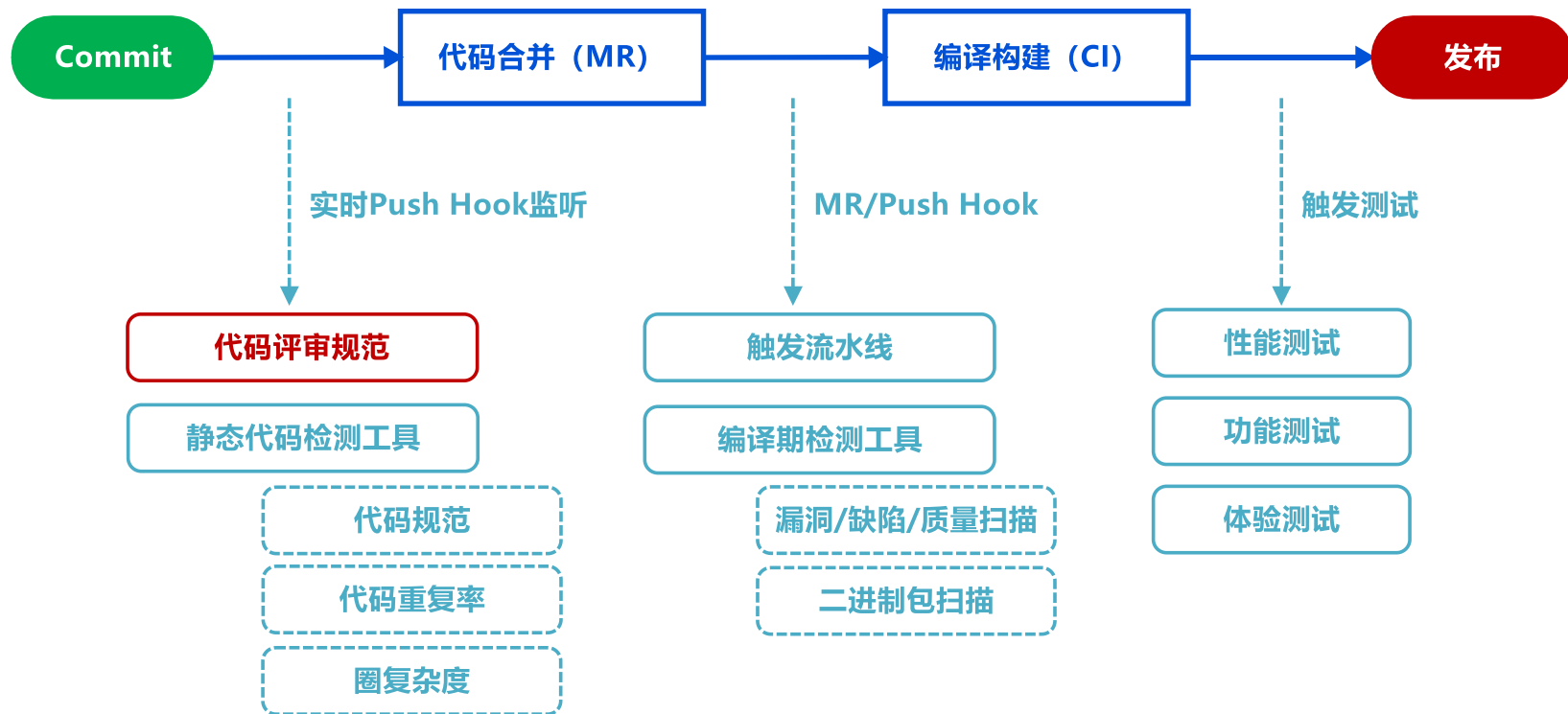
● 腾讯CI



腾讯团队协同打造的一站式
DevOps研发平台，助力业务平滑过渡到敏捷研发模式，持续快速交付高质量的产品



提交检测，质量前移




检测结果示例



CAS代码分析机器人 · 6 小时 以前

蓝盾流水线: [redacted] 状态: 执行成功 触发方式: mr申请
执行时间: 2020-04-29 11:53:27 任务耗时: 0时0分1秒

预测提交的风险评级: (无风险<0.15 | 0.15<中低风险<=0.4 | 0.4<中高风险<=0.65 | 高风险>0.65)

git hash	create time	commit message	风险评级
[redacted]	2020-04-29 11:48:54	--bug=[redacted]	无风险 
[redacted]	2020-04-29 11:52:02	--bug=[redacted]	中低风险

修改的代码热区文件(安全的历史bug数期望<=10)

文件名	历史开发人数	历史引入bug数	距离上次提交天数
[redacted]	314	294	20.0



[redacted] · 2 小时 以前

蓝盾流水线: [redacted] 状态: 执行成功 触发方式: 代码变更 任务耗时: 00时08分40秒

质量红线产出插件	指标	结果	预期
腾讯云CodeDog静态代码分析	新增(致命+错误)问题	0	<=0 
	超标圈复杂度总数	15546	<=19000 



变异测试机器人 · 6 分钟 以前

变异测试完成

- 变异测试分数: 100.0
- 变异测试报告: <http://mutationtest.oa.com> [redacted]



已合并
该合并请求已合并







已关闭
代码评审已关闭 (评审规则: 需要全部评审人同意)

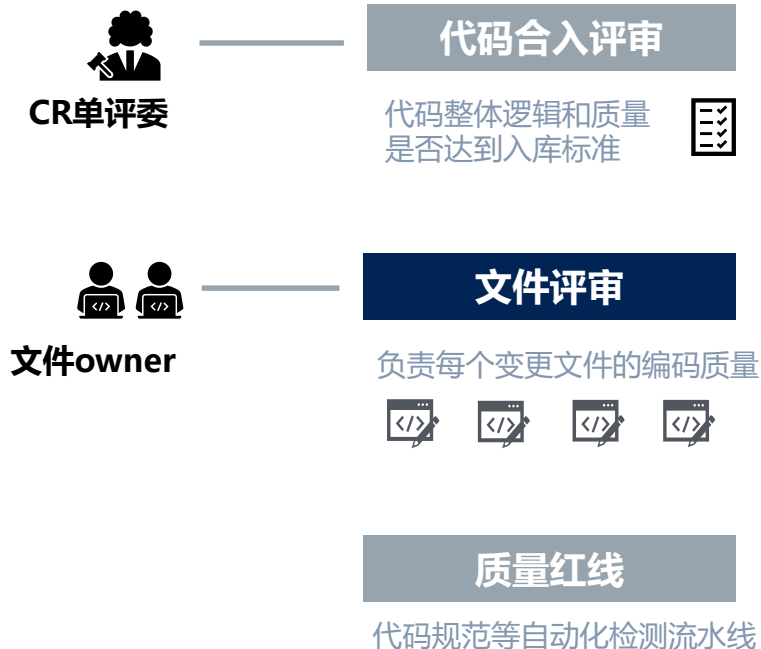


成功
检查成功

4 successful checks on commit

-  Pipeline - [redacted] [详情](#)
Passed: This commit has passed the checking on RDM Pipeline. See MR [redacted] 50 分钟 以前
-  蓝盾 - [redacted] (不阻塞合流) [详情](#)
Your pipeline [redacted] (不阻塞合流) is succeed 49 分钟 以前
-  蓝盾 - [redacted] 主干MR前监控 [详情](#)
Your pipeline [redacted] 主干MR前监控 is succeed 37 分钟 以前
-  蓝盾 - [redacted] MR合入主干后移除源分支 [详情](#)
Your pipeline [redacted] MR合入主干后移除源分支 is succeed 35 分钟 以前

两类评审人 + 三道质量门禁



Git研发畅想

一点点技术人员的想法

03

条条大路通Git

游戏研发体验

解决腾讯内大型
游戏项目在Git体系的研发体验问题
做到速度快、操作简、体量大

开发者工具

减少开发者在面对
越来越多的平台、工具链时的困扰
集成高频操作在命令行、IDE插件



先进研发模式

更多先进的研发模式支持完善
如Google的单一大仓、Facebook的
Phabricator提交前评审 workflows 等

海量服务高可用

企业内常规职能系统的可用率要求
不再适用于Git服务，要求99.99%
甚至于5个9是非常现实的问题

腾讯工蜂Git在路上

DevOps能力成熟度评测

卓越级

2020年9月25日，通过工信部

DevOps系统工具标准的首批评测

CCF科学技术奖

科技进步卓越奖

2019年，因在软件工程基建独立自主建设上的贡献，工蜂获得CCF科学技术奖最高等级奖项——科技进步卓越奖





Thanks

高效运维社区
开放运维联盟

荣誉出品