豆瓣阅读中的持续集成/发布实践

豆瓣 孙毅

豆瓣阅读

豆瓣阅读是

豆瓣读书推出的数字阅读服务

- 拥有质量一流的内容
- 支持Web、iPad、iPhone、 Android、Kindle等多种设备
- 提供极佳的阅读体验
- 社会化阅读

Why CI?

- 减少风险
- 减少重复过程
- 任意时间，地点可部署
- 可见性
- 信心

场景1-开发

本地服务起
不来了？

提交了才发
现问题？

依赖！

开发服务器网速赶不上手速...

问题分析

- 开发环境复杂且不统一
- 本地构建困难
- 本地没有快速反馈机制

解决方案

- 本地的统一的虚拟开发环境

- 订阅上游依赖变更，用puppet管理
- 大家一起贡献模块
- 基准开发工具包
- 简单便捷的本地ci

订阅上游依赖变更

- 必要性
  - 包依赖开发环境必须和线上环境同步升级
  - 公司内部的服务依赖，lib依赖版本升级

- 现状：RSS订阅依赖更新消息
  - 不是很先进，但是还算可靠

大家一起贡献模块

- 必要性
  - 项目众多，每一个项目依赖和工具不同
- 现状：
  - fork，pull-request
  - Puppet主文件中用注释进行特性开关

基准开发工具包

- 工具的种类/版本
- 工具的配置
- 现状：
  - 静态检查
  - 单元测试
  - Web测试

简单便捷的本地集成(1)

- pylint/jshint
  - 基准开发工具包包含工具
  - 随项目代码进行检查项配置
  - git pre-commit hook

简单便捷的本地集成(2)

- unittest/apitest
  - 不干扰本地开发服务
  - coverage
  - nosy/tag/etc..

简单便捷的本地集成(3)

- web测试
  - headless
  - webdriver
  - js error collection
  - html error collection
  - xunit

对比

before	now
只能在服务器开发	可迁移/可定制的完整本地开发环境
先提交，再跑集成测试，改bug	本地先进行测试，再提交
web测试只能在中心服务器	本地web测试

场景2-提交

好大一个diff！

这功能谁搞挂的？

懒得review

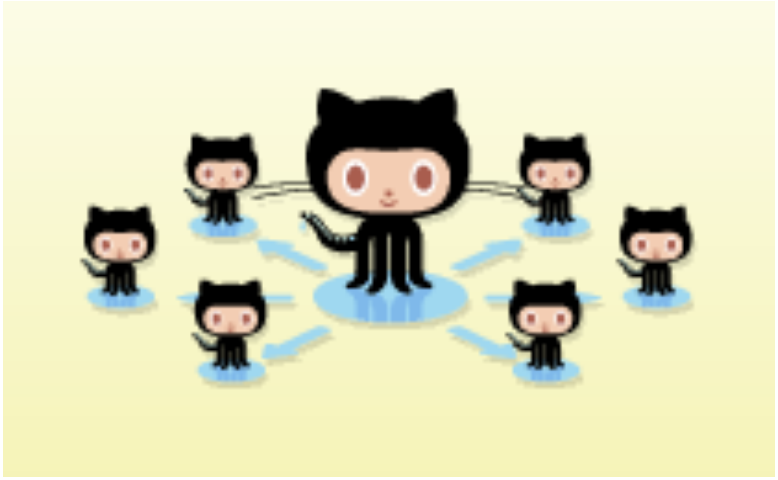
合入的时候咋办啊。。

问题分析

- review流于形式
- 分支合并成本高
- 问题定位困难

解决方案

- git / pull-request



- git分支的切换和合并成本极低
- 以pull-request作为review单元
  - 鼓励更多提交，强制review后合并
  - review覆盖面，针对性和参与度高
- 几乎每次merge都会触发构建
  - pull-request的粒度保证问题追查较容易
  - 通过构建job通知提交作者

对比

before	now
定期review，覆盖面小	每个pull-request必须review
review意见很难定位到行 响应也不及时	review精确到行，提醒机制完善 修复一目了然
写一大堆再合并提交	天天合并/提交 (5个月，1275个pull-request，2725个review意见，全体参与)
构建diff太大，出问题不知道谁的	几乎每个构建只有1-2个merge (5个月，900次构建)


46 + 在closed, staff, all三个状态之间循环
6
sunyi repo collab 9 days ago
这里的代码比较弱，之后再加入新的status该如何处理是一个问题，各位大神建议一个解决方案如何?
liuyue repo collab 8 days ago
突然觉得 loop 没什么必要...
sunyi repo collab 8 days ago
弄个set方法?
jiawei repo collab 8 days ago
三个状态分别的设置方法，一个非公开带参数的_set
sunyi repo collab 8 days ago
那如果新加状态的话就得新加方法?
zhuzhidong repo collab
难道不是一个类似set_status(self, status)的方法吗?
新加状态的时候多定义一个status就行了


构建 #3239 (2012-8-13 18:11:15)
构建产物
ark-dev-core.buildinfo 41
Changes
1. Bugfix: dealwith works when agent is None (detail / githubweb)
启动用户sunyi
Revision: 9feb017d7fef56bbd9eff19fdd25ea1577c26e6d
origin/master

场景3-构建

merge把主
干搞挂啦

CI服务器又
排队。。

跑一遍要
20分钟！

跑了15分钟
才告诉我没
通过：(

大家都喜欢
下班前提交

问题分析

- 分支集成不足
- ci suite反馈速度慢
- 无法获取阶段结果
- 持续集成服务器资源问题

解决方案(1)

- 基于opening pull request的分支持续集成

ark-opening-pr-scanner

ark-opening-pr-1273-ut-10750

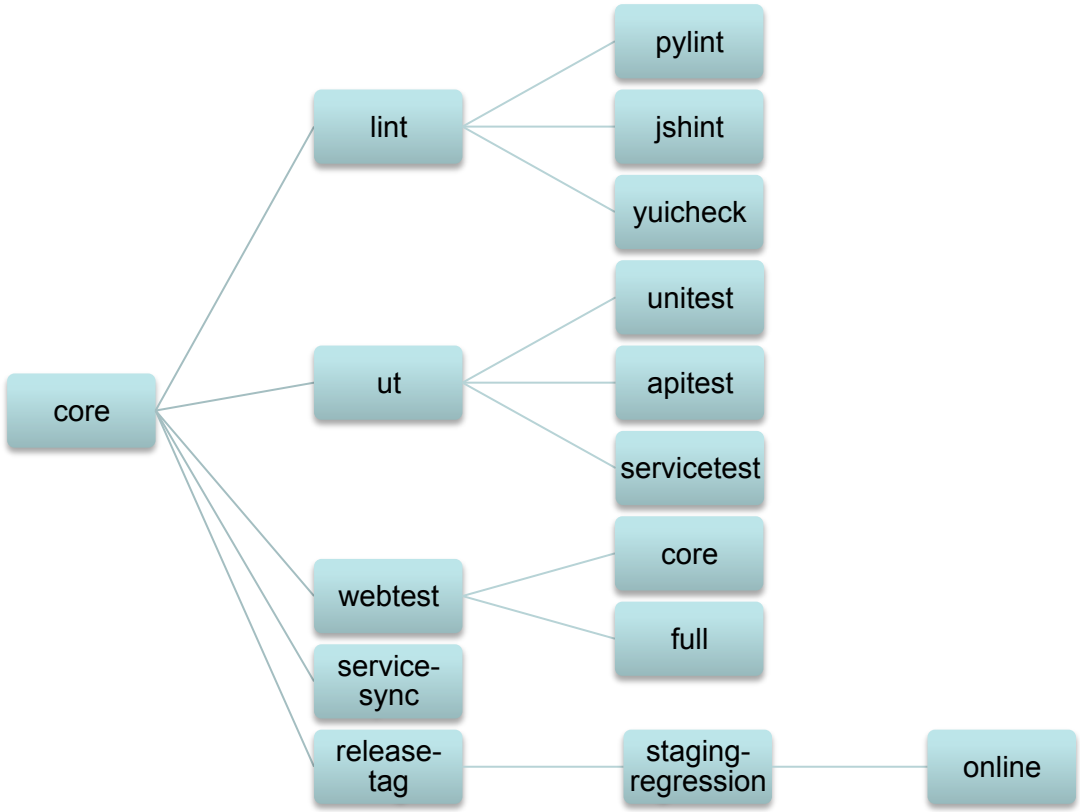
ark-opening-pr-1273-wd-10770

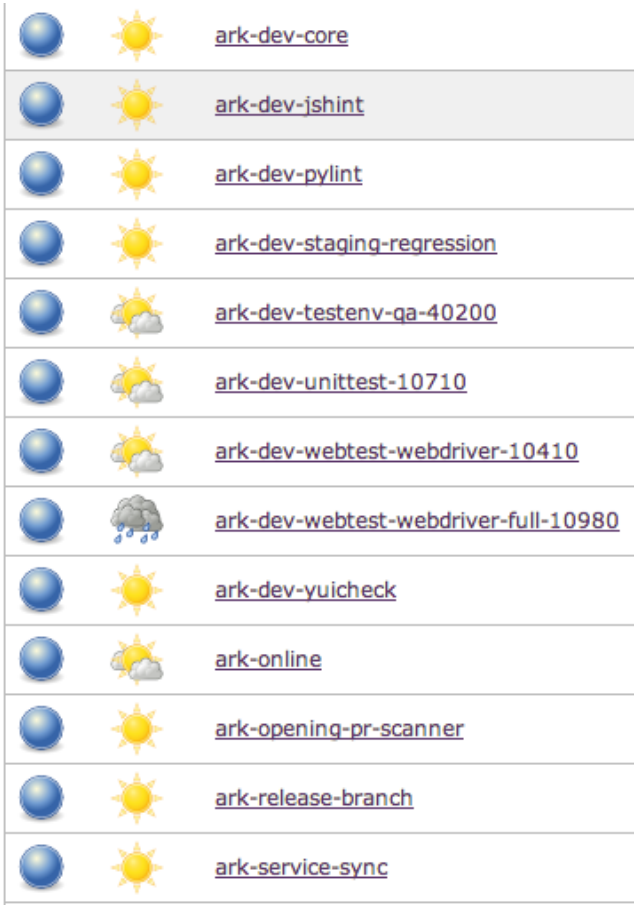
ark-opening-pr-1275-ut-11140

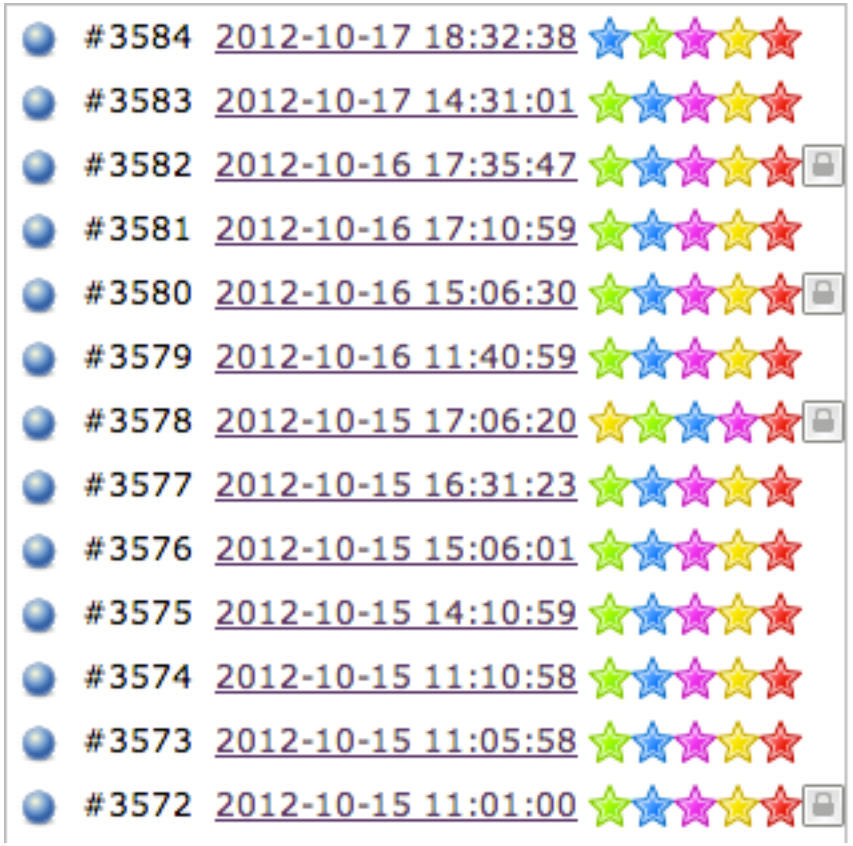
ark-opening-pr-1275-wd-11160

解决方案(2)

- 构建链/测试分级




ark-dev-core
ark-dev-jshint
ark-dev-pylint
ark-dev-staging-regression
ark-dev-testenv-qa-40200
ark-dev-unittest-10710
ark-dev-webtest-webdriver-10410
ark-dev-webtest-webdriver-full-10980
ark-dev-yuicheck
ark-online
ark-opening-pr-scanner
ark-release-branch
ark-service-sync


#3584 2012-10-17 18:32:38
#3583 2012-10-17 14:31:01
#3582 2012-10-16 17:35:47
#3581 2012-10-16 17:10:59
#3580 2012-10-16 15:06:30
#3579 2012-10-16 11:40:59
#3578 2012-10-15 17:06:20
#3577 2012-10-15 16:31:23
#3576 2012-10-15 15:06:01
#3575 2012-10-15 14:10:59
#3574 2012-10-15 11:10:58
#3573 2012-10-15 11:05:58
#3572 2012-10-15 11:01:00

解决方案(3)

- 冗余计算资源利用
  - 虚拟机开启jenkins-slave模块即接入ci系统
  - 控制node的tag来分配接入的job

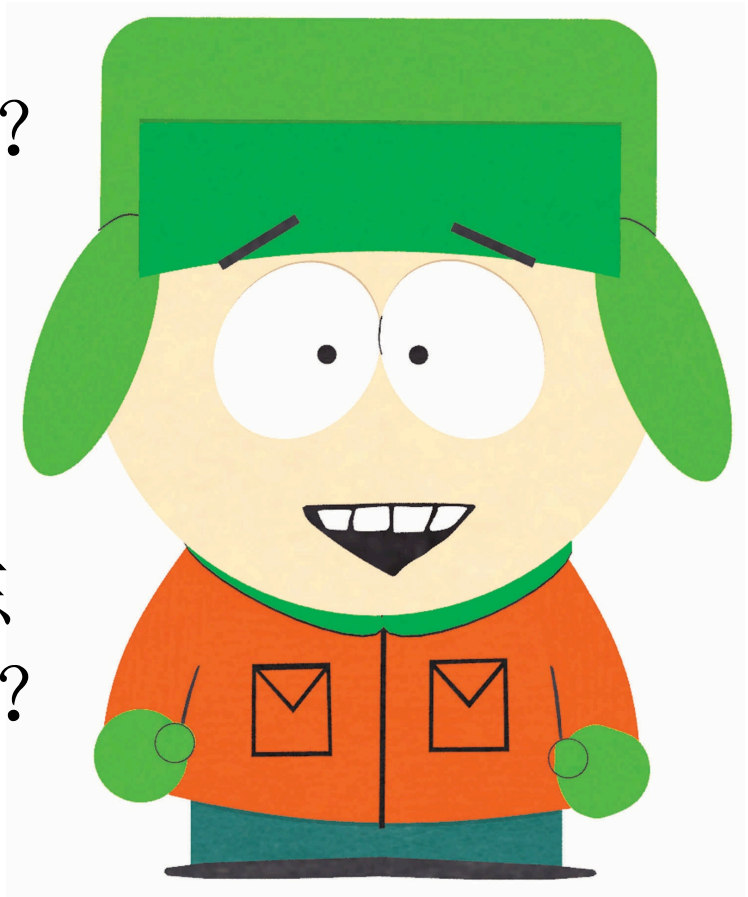
everbird-vagrant		
1	空闲	
linju-vagrant		
1	空闲	
sunyi-vagrant		
1	空闲	
whyme-vagrant		
1	空闲	

对比

before	now
中心ci只跑主干代码的ci suite	活动中的pull-request分支均有自己的ci suite，结果更新在pr记录中
静态检查4min	任务分拆并行化，关键任务11s/42s 后续任务1min48s
所有任务在中心服务器上排队 “下午4点的困扰”	大家贡献slave，大大提高了突然峰值下的执行速度，自己的任务已经可以由自己的slave全部消化
ci suite中的job过大过长	先跑核心，再dailybuild全量 提交构建控制在10min

场景4-交付

上哪个版本？

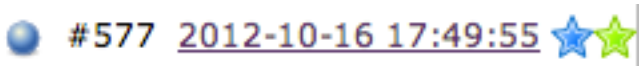


怕出线上问题啊…

XX不在，怎么上线来着？手抖了。。

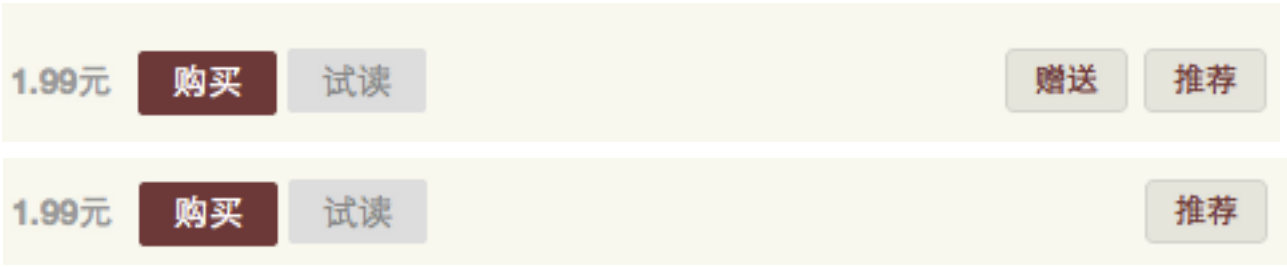
解决方案(1)

- 继承ci suite的版本状态监测/标记
  - 提交构建 

  - 打包打tag 

解决方案(2)

- 特性开关
  - 重大变更 均使用feature-switch
  - switch 多种状态，可在线切换



解决方案(3)

- 直接从ci集成自动化上线
  - 从ci suite传入可上线的tag
  - 由jenkins job自动执行远程脚本
  - 直接点击按钮触发

加入豆瓣

- 测试工程师/测试开发工程师
- 移动测试开发工程师
- 更多豆瓣职位
- team@douban.com

Q & A

您也可以通过以下方式找到我：
豆瓣主页： http://www.douban.com/people/tachikoma/
Email： sunyi@douban.com

Thanks