



Jenkins初体验以及应用

Presented by:
Jiaqi XU

- 互联网产品从规划到落地
 1. 根据顾客需求将产品设计成型
 2. 开发人员开发代码
 3. 测试团队测试功能
 4. 运维人员发布上线
- 过程中会遇到的问题

不同开发人员负责不同的功能模块的开发，开发完毕后模块独立测试虽然通过，但是上线前将所有模块整合到一起进行集成测试的时候却发现很多问题，想要解决就需要把很多代码返工重写，但是有可能时间不够了。
- 思考：如何避免，该怎么做？

我们可以尝试经常性地，频繁地把所有模块集成在一起进行测试，有问题尽早发现，尽早解决

- 持续集成(Continuous Integration)

概念：经常性地，频繁地把所有模块(分支)集成在一起进行测试，有问题
尽早发现、解决

好处：有利于快速发现错误；防止分支大幅度偏离主干

*Continuous Integrations doesn't get rid of bugs, but it does make them
dramatically easier to find and remove.*

---- Martin Fowler

目的：让产品快速迭代，同时还能保持高质量

- 持续交付(Continuous Delivery)

概念：频繁地将软件的新版本，交付给质量团队或者用户，以供评审。

如果评审通过，代码就进入生产阶段

用小版本不断进行快速迭代，不断收集用户反馈信息，用最快的速度改进优化。这样对双方都有保证，我们开发的效果能让顾客马上看到，也可以根据顾客的反馈做出及时的优化调整。

目的：研发团队的最新代码能够尽快让最终用户体验；强调不管怎么更新，软件是随时随地可以交付的。

- 持续部署(Continuous Deployment)

概念：代码通过评审以后，自动化部署到生产环境

目的：代码在任何时刻都是可部署的，随时可以进入生产阶段

- 集成，交付，部署有一定的工作量
- 持续地集成，交付，部署很重要，但又会加大工作量
 - => 影响项目进度和效率
- 自动化CI/CD，在后台不断地去集成，不断地交付，不断地部署
 - => 提高效率
 - => 使大家专注到项目开发本身
- 自动化CI/CD工具






- 简介
Jenkins是开源CI&CD软件领导者， 提供超过1000个插件来支持构建、部署、自动化， 满足任何项目的需要
- 特点
 1. 持续集成和持续交付
 2. 简易安装
Jenkins基于java语言开发， 需要安装JDK
 3. 配置简单
相对友好， 在图形化界面下进行配置
 4. 插件
支持超过1000个插件， e.g. 用户权限相关插件， ssh相关插件， git相关插件
 5. 扩展
可以通过其插件架构进行扩展， 从而提供其他更多可能的功能
 6. 分布式
支持分布式， 可以配置主从(master/slave)从而支持多个不同的机器去协作工作

- Jenkins在服务器上相关目录结构(以ai8为例)
 1. 配置文件: `/etc/sysconfig/jenkins`
包含了一些jenkins相关的环境变量
e.g.
JENKINS_HOME: jenkins主目录, 包含了很多配置文件和项目相关的文件
JENKINS_PORT: jenkins服务的监听端口, 默认8080
JENKINS_USER: 运行jenkins守护进程的用户, 拥有一些特定的权限, 修改时需要注意
 2. 程序主目录 JENKINS_HOME: `/data/jenkins/jenkins_home` (ai8)
e.g.
包含的文件夹:
 - jobs: 包含已经建立的项目
 - workspace: 已经建立项目的工作空间
 - plugins: 安装的插件
 - users: 注册的用户
 3. 日志 `/var/log/jenkins/jenkins.log`
可以用于了解jenkins是怎么工作, 怎么与别的应用进行交互
也可以在进行相关配置时候用于DEBUG

Jenkins图形化界面

Jenkins

4

查找

Jiaqi Xu | 注销

Jenkins

新建任务

用户列表

构建历史

项目关系

检查文件指纹

系统管理

我的视图

Job Config History

打开 Blue Ocean

凭据

新建视图

构建队列

























































队列中没有构建任务

构建执行状态

1 空闲

所有

+

S	W	名称 ↓	上次成功	上次失败	上次持续时间	收藏
		clean failed zombie process	4 月 1 天 - #2	3 月 28 天 - #4	0.34 秒	 
		Continuous Integration	4 月 2 天 - #39	3 月 27 天 - #51	14 分	 
		github_integration_test	9 天 18 小时 - #1	无	18 秒	 
		github_pr_test	3 天 17 小时 - #34	2 天 18 小时 - #36	17 秒	 
		test	10 天 - #4	10 天 - #2	2.4 秒	 
		test_demo	17 分 - #1	17 分 - #2	94 毫秒	 
		test_pull_repo	8 天 21 小时 - #1	无	14 秒	 
		UL DP CFT	10 小时 - #18	2 天 3 小时 - #15	12 分	 
		UL DP Clean Data Daily	10 小时 - #18	6 天 22 小时 - #4	10 分	 
		UL DP CLEAR	9 小时 50 分 - #12	2 天 2 小时 - #9	17 分	 
		UL DP DOVE	10 小时 - #11	2 天 2 小时 - #8	12 分	 
		UL DP HAZELINE	9 小时 32 分 - #11	2 天 2 小时 - #8	8 分 15 秒	 
		UL DP LUX	9 小时 24 分 - #11	2 天 2 小时 - #8	12 分	 
		UL DP OMO	9 小时 12 分 - #13	2 天 1 小时 - #10	22 分	 

图标: [小](#) [虫](#) [大](#)

[图例](#) [RSS 全部](#) [RSS 失败](#) [RSS 最新的构建](#)

新建任务

- 通用
- 源码管理
- 构建触发器
- 构建环境
- 构建
- 构建后操作





























系统管理

- 系统配置
- 全局安全配置
- 插件配置
- 管理用户

UL DP项目从代码角度可以分为两个流程

- 数据清洗流程 `clean_data`
- 特征工程、模型训练、跑模型出预测 `pipeline`

在Jenkins设置对应的工程

		UL DP CFT	12 小时 - #19	3 天 5 小时 - #15	13 分	 
		UL DP Clean Data Daily	12 小时 - #19	8 天 0 小时 - #4	10 分	 
		UL DP CLEAR	1 天 11 小时 - #12	3 小时 15 分 - #14	17 分	 
		UL DP DOVE	12 小时 - #12	3 天 4 小时 - #8	13 分	 
		UL DP HAZELINE	11 小时 - #12	3 天 4 小时 - #8	8 分 2 秒	 
		UL DP LUX	11 小时 - #12	3 天 4 小时 - #8	12 分	 
		UL DP OMO	11 小时 - #14	3 天 3 小时 - #10	24 分	 

具体构建设置

- 定时构建
 - 每天凌晨2点
 - 清洗数据流程稳定构建后，触发其他6个品类对应的pipeline的构建
- 构建失败发送EMAIL通知管理员
- 作用
 1. 尽早发现集成后代码中存在的问题
 2. 可以知道每个pipeline的运行时长，并且根据日志调查具体是哪个阶段用时较长，进行相应的优化

EMAIL NOFITICATION SETTING

SMTP服务器	<input type="text" value="smtp.partner.outlook.cn"/>
用户默认邮件后缀	<input type="text" value="@guandata.com"/> ?
<input checked="" type="checkbox"/> 使用SMTP认证	?
用户名	<input type="text" value="xujiaqi@guandata.com"/>
密码	<input type="password" value="....."/>
使用SSL协议	<input type="checkbox"/> ?
SMTP端口	<input type="text" value="587"/> ?
Reply-To Address	<input type="text"/>
字符集	<input type="text" value="UTF-8"/>
<input checked="" type="checkbox"/> 通过发送测试邮件测试配置	
Test e-mail recipient	<input type="text"/>

Test configuration

可能会遇到的报错：

530 5.7.57 SMTP; Client was not authenticated to send anonymous mail during MAIL FROM

解决方法：配置文件/etc/sysconfig/Jenkins, 为JENKINS_JAVA_OPTIONS加上-Dmail.smtp.starttls.enable=true; 启动TLS(传输层安全协议)

- 背景
在开发代码过程中，都会经历这样一个流程，首先在自己的分支上进行开发，开发完毕后，会进行测试，测试通过后会进行pull request，然后由同事review代码，没有问题后将会把本次pr合并到主分支当中去。

点击merge时，表面很淡定，内心其实很慌，没有安全保障
原因：没有一个标志性，值得信赖的结果能够告诉我
 1. 集成之后的代码不会有bug
 2. 所有的测试都能通过或者是pipeline能跑通，预测的结果也符合预期
- Jenkins支持的Github Pull request builder插件可以满足我们的需求，实现：
每当有pull request时，通知jenkins去触发相应工程的构建，这个构建的执行过程可以是将这个pr和主分支拉取到jenkins的workspace下，进行集成，并且在类生产环境下跑测试用例，或者是跑pipeline，并将结果反馈在github上
- 涉及github repo与jenkins的交互，需要同时对github上对应的repo和对jenkins进行相关配置
- 以jenkins-github项目为例，对两者进行配置，从而使当一个新的PR来时，让github通知jenkins自动触发集成，并执行相关的构建脚本，并且在构建过程中以及结束后都将构建结果及时反馈到github上。

Github pull request builder

在jenkins上进行相关配置

- 全局配置

GitHub Pull Request Builder

GitHub Auth

GitHub Server API URL ?

Jenkins URL override ?

Shared secret ?

Credentials 添加 ?

Test Credentials...

Create API Token...

Description ?

Auth ID...

Delete Server

新增

Auto-manage webhooks ☒

Use comments to report results when updating commit status fails ☐

Use comments to report intermediate phases: triggered et al ☐

Admin list

jiaqi-xu

高级...

全局配置说明

- Github Server API
因为用的个人的github, 所以直接设置server api URL为https://api.github.com
- Shared secret
当不为空的时候, github发送的webhook都要验证这个密钥; 所以在github上设置webhook的时候也需要设置对应的secret, 否则jenkins收到webhook通知后, 验证不通过, 就不会做出相应的回应。

```
nnable#run: Created hook for jiaqi-xu/Jenkins-github
2020-03-07 10:11:15.013+0000 [id=30] INFO o.j.p.ghprb.GhprbRootAction#handleAction: Checking PR #9 for jiaqi-xu/Jenkins-github
2020-03-07 10:11:15.013+0000 [id=30] SEVERE o.j.p.ghprb.GhprbGitHubAuth#checkSignature: Request doesn't contain a signature. Check that github has a secret that should be attached to the hook
2020-03-07 10:11:15.014+0000 [id=30] SEVERE o.j.p.ghprb.GhprbGitHubAuth#checkSignature: Request doesn't contain a signature. Check that github has a secret that should be attached to the hook
```

webhook未设置secret时

```
2020-03-17 02:51:11.376+0000 [id=1415] INFO o.j.p.ghprb.GhprbRootAction#handleAction: Checking PR #22 for jiaqi-xu/Jenkins-github
2020-03-17 02:51:11.377+0000 [id=1415] SEVERE o.j.p.ghprb.GhprbGitHubAuth#checkSignature: Local signature d9891f54066786c6366b90ef6545df675f9049b9 does not match external signature 883f8ba050540c5c026a12e959cd8e9ab0d013e1
```

Secret不匹配时

全局配置说明

- Credential

三种类型的credential

1. username with password
2. ssh username with private key
3. secret text (personal access token, OAuth)

设置credential目的是让jenkins有权限使用github server API或者从对应的repo上拉取代码到自己的分支，也可以发送请求将某些数据信息传回给对应的repo等等

☒ Test basic connection to GitHub

- Test basic connection to github

Connected to https://api.github.com as Jiaqi Xu (null)
login: jiaqi-xu

Connect to API

- Test permission to a repository

Repository owner/name

☒ Test Permissions to a Repository

User has access to: Admin, Push, Pull

Check repo permissions

在Github上进行相关配置

- Webhooks

1. github上对webhooks的简述

Webhooks allow external services to be notified when certain events happen.

When the specified events happen, we'll send a POST request to each of the URLs you provide

2. 配置

- a. payload(hook url)

这里需要填写一个jenkins server所在服务器端口的公网IP或者域名并加上/ghprbhook/

遇到问题:

jenkins服务是运行在ai8服务器8080端口的,
但是由于一些原因我们可能暂时不能为其开放公网域名以供访问

解决方式:

ngrok tunnel工具, 反向代理将127.0.0.1:8080 => public domain

在Github上进行相关配置

2. 配置

b. content type

Content-type定义了数据被发送到服务器的格式，有2种选择，以json的格式发送到服务器；另外一个是被编码为key/value的格式发送到服务器。

c. secret

根据jenkins那边的对github pull request builder全局配置来填写

d. 设置触发事件

3. Recent delievery 可以查看具体发的请求细节

Github pull request builder

- 构建任务
 - 源码管理，高级设置

☐ 无

☒ Git

Repositories

Repository URL

https://github.com/jiaqi-xu/Jenkins-github.git

Credentials

jiaqi-xu/*****

添加

Name

origin

Refspec

+refs/pull/*:refs/remotes/origin/pr/*

Add Repository

Branches to build

指定分支（为空时代表any）

`\${sha1}`

增加分支

源码库浏览器

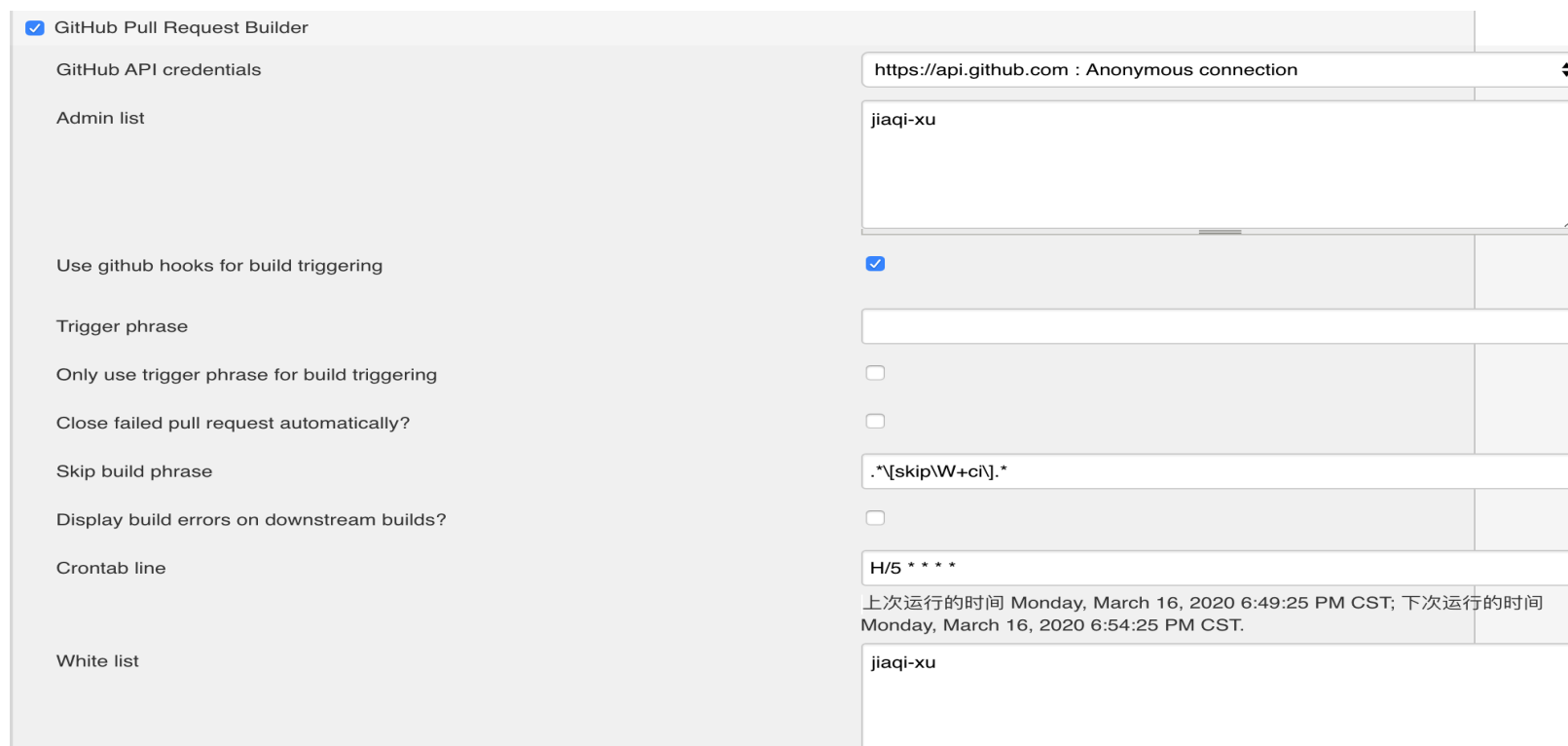
(自动)

Additional Behaviours

新增

2. 构建触发器

a. 勾选Github Pull Request Builder



The screenshot shows the configuration interface for the Github Pull Request Builder. On the left, a sidebar lists various settings: GitHub Pull Request Builder (checked), GitHub API credentials, Admin list, Use github hooks for build triggering, Trigger phrase, Only use trigger phrase for build triggering, Close failed pull request automatically?, Skip build phrase, Display build errors on downstream builds?, Crontab line, and White list. The main area on the right contains the configuration details for the selected option. It includes a dropdown menu for the Github API endpoint (https://api.github.com : Anonymous connection), a text input for the Github username (jiaqi-xu), a checked checkbox for 'Use github hooks for build triggering', an empty text input for the 'Trigger phrase', an unchecked checkbox for 'Only use trigger phrase for build triggering', an unchecked checkbox for 'Close failed pull request automatically?', a text input for the 'Skip build phrase' (.*[skipW+ci].*), an unchecked checkbox for 'Display build errors on downstream builds?', a text input for the 'Crontab line' (H/5 * * * *), and a text input for the 'White list' (jiaqi-xu). At the bottom, it displays the last and next run times: '上次运行的时间 Monday, March 16, 2020 6:49:25 PM CST; 下次运行的时间 Monday, March 16, 2020 6:54:25 PM CST.'

b. Update commit status during build

测试:

1. 提交一个pull request, 可以通过查看日志文件了解jenkins和github之间是如何交互的

```
-----
2020-03-16 12:02:56.879+0000 [id=1046] INFO    o.j.p.ghprb.GhprbRootAction#handleAction: Checking PR #22 for jiaqi-xu/Jenkins-github
2020-03-16 12:02:56.880+0000 [id=1046] INFO    o.j.p.ghprb.GhprbGitHubAuth#checkSignature: Signatures checking OK
2020-03-16 12:02:56.883+0000 [id=1049] INFO    o.j.plugins.ghprb.GhprbTrigger#handlePR: Checking PR #22 for job github_pr_test
2020-03-16 12:02:56.884+0000 [id=1049] INFO    o.j.p.ghprb.GhprbPullRequest#<init>: Created Pull Request #22 on jiaqi-xu/Jenkins-github by jiaqi-xu () updated at: 3/16/20 8:02 PM SHA: e70864c83f2fe1669395cd57c563a1e58579745b
2020-03-16 12:02:56.885+0000 [id=1049] INFO    o.j.p.ghprb.GhprbPullRequest#updatePR: Pull request #22 was updated on repo jiaqi-xu/Jenkins-github but there aren't any new comments nor commits; that may mean that commit status was updated.
2020-03-16 12:03:27.503+0000 [id=1012] INFO    hudson.model.Run#execute: github_pr_test #38 main build action completed: FAILURE
-----
```

2. 查看对应build的控制台输出, 了解build执行的具体过程

Github pull request builder

构建结果展示

Add more commits by pushing to the **div_zero_error** branch on **jiaqi-xu/Jenkins-github**.



All checks have failed

[Hide all checks](#)

1 failing check



Jenkins — Jenkins-github test fails.

[Details](#)



This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request



You can also [open this in GitHub Desktop](#) or view [command line instructions](#).



- DONE
 - crontab clean_data + pipeline
- TODO
 1. 测试模型预测结果
是否有负数；箱数的预测结果是否为小数；预测文件的个数
 2. KPI test
将模型的预测结果与设定的baseline比较看是否达标
 3. Code pep8 check
- 现状

目前这个主分支的代码，还没有直接让客户来使用，是我们自己的工程师执行出预测的，所以可以短暂容忍一直有报错；将来一旦代码产品release出去了，可能就不能容忍代码中断报错了。
- 发展

自动化CI/CD需要跟上项目发展的步伐，并且十分地重要；一个成熟的项目需要一套成熟的自动化CI/CD流水线来支持，希望未来我们也有这样一套流水线来支持自动化持续集成，持续交付，一键部署。



Thank you for your time 😊