# Jenkins初体验以及应用

## CI/CD

我们都知道互联网产品从规划到落地简单地来说都会经历这么一些过程：首先呢会根据顾客需求将产品设计成型，其次开发人员开发代码，然后测试团队测试功能，最后运维人员发布上线。

但是在这个过程当中，就会涉及到很多细节：比如说在开发阶段，我们会有不同的开发人员负责不同的功能模块的开发，开发完毕后，可能会发生自己的模块独立测试虽然能通过，但是在上线前将所有模块整合到一起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，但是现在时间很可能不够了。为了避免这种情况，怎么做会好一些呢？

是的，我们可以尝试经常性地，频繁的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，这就是持续集成的概念。

(next)

持续集成的好处主要有2个：  
1）快速发现错误。我们可以每完成一点更新，就集成到主干，可以快速发现错误，定位错误也比较容易。

2）防止分支大幅偏离主干。如果不是经常集成，主干又在不断更新，会导致以后集成的难度变大，甚至难以集成。

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

---- Martin Fowler

这里引用软件开发教父martin fowler说过的一句话，持续集成并不能消除Bug，而是让它们非常容易发现和改正

持续集成的目的，就是让产品可以快速迭代，同时还能保持高质量。

（next）

与CI常常一起出现的是CD， CD既可以表示持续交付(Continuous Delivery)，也可以表示持续部署(Continuous Deployment)

持续交付（Continuous delivery）指的是，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。

持续交付可以看作持续集成的下一步。持续交付在于研发团队的最新代码能够尽快让最终用户体验；它强调的是，不管怎么更新，软件是随时随地可以交付的。

持续部署（continuous deployment）是持续交付的下一步，指的是代码通过评审以后，自动化部署到生产环境。

持续部署的目标是，代码在任何时刻都是可部署的，可以进入生产阶段。

(next)

集成，交付，部署其实都是有一定的工作量的，而且持续的进行这些操作十分重要，但是会加大工作量，这样频繁操作会让我们非常得难受，甚至会影响项目的进度和效率。我们希望如果这些操作都能自动化，自动地在后台不断地去集成，不断地交付，不断地部署，这将提高团队的工作效率，让大家专注到项目开发本身去。

支持自动化CI/CD的工具有很多，，但是我们今天要介绍的主角是jenkins.

(next)

## Jenkins介绍

左边是jenkins的log， 是一个很绅士的老爷爷，默默地提供专业的服务，为我们的项目保驾护航。

Jenkins是开源CI&CD的领导者，提供超过1000个插件来支持构建，部署，自动化，满足任何项目需要。

这里列出了一些jenkins的特点

1. 持续集成和持续交付
2. 简易安装，几条命令就搞定了。有一点需要稍微要注意一下的是Jenkins是基于java语言开发的，所以在安装之前需要安装JDK，
3. 配置简单友好，都是在图形化界面下完成的
4. 关于插件方便：它支持很多的插件，比如用户权限相关的插件，可以有助与管理项目；ssh相关的插件，可以远程的去执行一些命令，可以远程的拷贝一些文件；也可以安装git相关的插件，可以去github上拉代码,以及实现很多与github交互的功能。
5. Jenkins可以通过其插件架构进行扩展，从而提供其他很多可能的功能。
6. Jenkins还支持分布式，可以配置主从(master/slave)从而支持多个不同的机器去协作工作

(next)

### Jenkins服务，AI8

我们在ai8上有运行一个jenkins服务，我们可以一起来看下,  
ssh guan-ai8 => ps aux | grep Jenkins =》 变大看一下   
可以看出是由java解释器来执行的一段命令，这里有很多参数，主要的是/usr/lib/jenkins/jenkins.war 简单的来说，他其实是java web application的打包，里面应该包含了很多servlet和相关的资源；这也反映了jenkins是基于java开发的；这里还有一些别的参数，比如说jenkins主目录(选中-DJENKINS\_HOME=/data/jenkins/jenkins\_home)；jenkins日志文件(--logfile=/var/log/jenkins/jenkins.log), jenkins监听的端口(--httpPort=8080)

（回到ppt）  
他在ai8服务器上相关的目录结构和文件

其配置文件是这个: /etc/sysconfig/Jenkins 其中包含了很多关于jenkins的环境变量，

比如说

JENKINS\_HOME: jenkins主目录，包含了很多配置文件和项目相关文件

JENKINS\_PORT: jenkins服务的监听端口，默认是8080，可以根据需要进行修改  
JENKINS\_USER: 运行jenkins守护进程的用户，拥有一些特定的权限，修

改时可能需要注意

程序主目录：/data/jenkins/jenkins\_home, 其下面包含了很多文件夹，比如说

Jobs: 包含已经建立的项目

workspace：已经建立项目的工作空间

plugins：包含了安装的插件

users： 包含了所有注册的用户

在任何场景，日志文件都是不可缺少的，并且相当重要的。Jenkins的日志文件在：：cd /var/log/jenkins/ Jenkins.log，我们可以根据日志文件  
 1. 了解jenkins是怎么工作，怎么与别的应用进行交互的

2. 也可以在配置相关插件的时候用于DEBUG

### Jenkins界面

我们一起来看下jenkins究竟长什么样子， 网页打开localhost：8080(如果没有tunnel，先tunnel)

这个就是jenkins的主页面

左边是菜单栏，  
中间是主界面： 这里列出一些已经建立的项目，并且每一行都展示了关于这个项目构建的一些基本信息。

我们来具体看一下每一列代表的含义：  
S（status）: 表示上次构建的状态，反应了项目上次构建的成功与否  
W(weather)，晴雨表，描述了构建的稳定性，举个例子雨天表示最近4次构建失败了3次，打雷表示最近的构`建都失败了，太阳表示最近的构建都成功了；

后面几列依次表示项目的名称，上次构建成功的时间，上次构建失败的时间以及上次构建运行的时间

可以通过点击项目名称来查看项目的具体细节。 比如说test，左下方是该项目的构建历史(这里可以看出过去总共有五次构建，其中有4次成功的构建)，左边是关于这个项目的菜单栏(立即构建，开始这个项目新的一次构建，可以点击配置查看这个项目的配置信息)。

接下来，我们通过创建一个简单的项目，来了解一下一个项目支持的配置有哪些：

回到主页，新建任务=》选择构建风格自由的软件项目(有很多选项，我们暂时选这个就可以了)=〉输入名称=》进入到项目的配置页面  
配置页面主要由通用，源码管理，构建触发器，构建环境，构建，构建后操作这6个模块组成。我们可以一起来看下，

##### 通用模块

描述： 可以介绍一些这个项目的功能和目的

Github项目（勾选）: 如果项目是在github上的话，可以填上项目在github上对应的URL

显示名称: this value will be used as context name for commit status if status builder or status publisher is defined for this project. It should be small and clear.

Throttle builds(勾选): 节流构建，通过设置构建数和time period，从而控制这个项目build的频率。

(演示)比如说我可以设置构建数为2个，time period为1小时，那么构建的频率将是每30分钟，构建一次。

丢弃旧的构建：也可以通过设置条件，从而丢弃项目中不满足条件的构建记录

参数化构建：允许设置一些参数，当构建的时候可以根据需要将参数作为input传给该次构建。也可以设置默认值，当自动构建的时候会使用设置的默认值。

关闭构建：当这个选项被勾选之后，该项目再也不会有新的构建被执行。

(其实每个选项后面都有question mark，可以通过点击他查看该选项的具体说明)：

在必要的时候并发构建  
勾选这个之后，如果有足够的构建执行器，则允许工程的多个构建并发执行。  
默认情况下，某一时刻只有单个构建正在执行，其他的构建请求会被入队到构建队列中，直到前一个构建执行完毕，才会被构建执行器执行。

##### 源码管理

如果安装了相关版本控制的插件，则对应的版本控制工具就会显示在这里。

以git为例， 如果有一个项目在github上，则可以勾选git选项，并设置项目对应的repo url, 并可以根据具体构建中是否需要与github进行交互(比如从github上拉取特定的分支代码)，从而进行相应的设置。

Mercurial也是一种分布式的version control, 其相关命令和git类似，也有clone push pull add 只不过他的命令是以hg开头的。

当然jenkins也支持集中式的version control, 比如说Svn, perforce(p4)

##### 构建触发器

可以根据需要，选择构建一些触发器，从而触发项目的构建。我们可以挑着看几个：

触发远程构建(勾选): 允许我们提前定义一个URL，当这个URL被hit的时候，就会触发该工程的一次构建。只不过这个url有一些特定的格式

其他工程构建后触发(勾选)： 这种触发方式允许指定1个或多个关注的项目，可设置当其构建(稳定，不稳定，失败)后，再触发本工程的构建。

定时构建(勾选):他允许我们在具体的时间到达的时候触发项目的构建。它的时间设置语法和crontab基本类似，具体参考crontab.guru，并且question mark也有对应的说明

##### 构建环境

指的是设置构建时的环境，他也有很多选项

Delete workspace before build starts 在构建开始之前清理工作空间

Use secret text or files 通过使用secret text或files 来设置构建过程中要用的环境变量

Absort the build if its stuck 中断构建当其被卡住了

Add timestamps to the console output 为控制台输出添加时间戳

##### 构建

这个是主要的part，在这里我们需要设置项目构建时具体要做的事情：举个例子选择执行shell, 那么接下来可以根据需要写入我们想要其做的事情，可以是执行整个项目所有test case的入口函数；可以是跑ai项目的清洗数据的脚本，也可以是跑模型的pipeline的入口脚本等等。  
  
举个简单例子，我希望他执行ai8服务器上的一个python文件，我们可以一起来看下这个文件，内容很简单，我这里设置了一个division by zero 的错误，所以构建的结果我们预期将会是失败的；  
我们将希望执行的命令填写到shell块上  
cd /data/unilever/workspace/xjq/unilever/notebook

Jenkins\_demo.py

##### 构建后操作

这里也有一系列选项：

比如说，构建其他工程，触发其他工程的构建

Email Notification: 希望测试没通过时或者pipeline没跑通的时候发邮件通知我。

也可以在构建完成后与github交互，可以是add label, close pr, post comment等等

配置完毕，我们点击应用保存

回到工程，我们点击立即构建(查看构建历史)=》  
=〉 点击具体的构建=》控制台输出=〉我们看到这里有关于这次构建的输出日志

通过输出日志，我们可以知道：  
这次构建是由我点击触发的，是以SYSTEM 用户运行的

并且在对应的workspace进行构建：

构建的具体过程是执行shell 脚本(这个脚本就包含了我们刚刚在shell脚本框里写的那2行命令)， 日志给出执行的细节，包括执行的顺序和对应的输出。

并且这里给出了构建失败的原因是division by zero error；类似的如果存在测试用例不通过，他也会构建失败。

（回到ULDP应用ppt）

### 结合项目看下具体应用

接下来，介绍下jenkins目前在UL DP项目上的应用

UL DP项目从代码上来说可以分为2个流程， 一个是数据清洗流程(称其为clean\_data);另外一个流程包括了特征工程，模型训练，跑模型出预测，我们称其为pipeline

我们在jenkins上分别为这2个流程建立了相关的工程，可以一起来看下(回到jenkins页面)，一个叫UL DP Clean Data Daily, 另外一个流程因为我们目前支持6个brand(有金纺，清扬，奥妙，夏士莲，力士，多芬)，所以对每一个brand设置了一个pipeline工程。

我们目前是将clean data对应的这个工程设置为定时任务，每天凌晨2点都会自动构建，稳定构建完毕后会触发其他6个brand的pipeline工程的构建。  
  
可以来看下clean data工程的具体配置：

项目描述信息告诉我们这是一个定时构建项目

定时构建： 我们设置为每天凌晨2点触发构建

构建脚本：首先是先激活conda环境，其次同步最新代码，然后安装编译UL DP 源码src包，最后运行清洗数据的入口脚本

我们还将其设置为构建稳定后触发其他的pipeline工程，并且在构建失败时发送邮件通知。

再来看一下pipeline的工程

描述：给出了品类信息

执行脚本：因为目前是将该工程设置为clean data构建完触发(同步代码，安装编译最新的src包这些步骤在clean\_data已经做过了)，所以只需要直接运行相关的pipeline入口脚本就可以了；

自前几个星期配置完这些工程后到现在，我们多多少少都从中受益了一些东西：

比如说，

他帮助我们及时发现代码集成后的问题，从配置完毕到至今，发生过几次失败的构建并及时的发邮件通知我们，从而使我们能趁早发现代码集成后的问题以及一些环境的问题：  
比如说来看下comfort ，这是构建历史，最近的构建都成功了，上周出现过构建失败的情况，第14次，和第15次，他们发生在上周六(是因为周五有新的代码被merge了)，分别通过查看控制台输出，快速定位到错误，14: 给出缺少这个包的错误，说明代码更新了引入新的包，周六的集成马上就发现了这个问题，这样我们就知道需要将这个包，安装到类生产环境和生产环境中去；15 缺少文件 ; 我们能尽早地发现这些问题并修复

另外，还可以了解每个brand的运行时间(演示一下)

可以知道那个运行时间较长，通过查看控制台输出知道具体是哪个part用时较长，并调查做出相应的优化。(举个例子，比如说这里omo和clear，通过查看日志，猜测可能是因为目前只有他们用了top down模型的缘故从而导致他们的运行时长相对长一些)

（回到ppt）  
邮件通知的配置，设置如图，需要设置一下对应的smtp服务器，用户名密码，端口。设置完毕后，可以通过发邮件测试配置是否正确。

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)

(还在ppt)

接下来要介绍一款插件叫做github pull request builder.

#### Github pull request builder

(回到ppt)

我们在开发代码过程中，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，然后由比较senior 同学review代码，没有问题后将会把本次pr合并到主分支当中去。

相关人员在点击merge的那一刻，其实是很迷茫的(表面很淡定，内心其实很慌)，因为没有一个标志性的结果来给我保障，告诉我merge之后，集成的代码不会有问题，所有的测试用例都能通过，所有的pipeline都能跑通，预测的结果也是符合预期的。

这个时候，我们会希望有这样一个工具能自动地将pull request与主分支进行集成，并进行测试。

**Jenkins支持的Github pull request builder这个插件可以满足我们的要求，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的工作空间下，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**。

上面的功能涉及到github上的repo和jenkins的交互，需要对两者都进行相关配置才可以。

在我的github上，有一个jenkins-github项目，下面我以它为例子，介绍相关配置，从而使当一个新的PR来时，自动触发jenkins进行集成，并执行相关的构建脚本，并且在构建过程中或者结束后都将构建结果及时反馈到github上。

接下来我快速的描述一下配置过程以及在这个过程中遇到的问题，然后提交一个pr到jenkins-github这个repo给大家看下效果：

首先需要在jenkins上进行该插件的全局配置，全局配置如下；我们可以到具体的页面上看一下(系统管理->系统配置-> 定位到github pull request builder的全局配置)

首先要配置的就是：

1. Github Server API URL

因为用的个人的github, 所以直接设置server api URL为“https:api.github.com”；企业版的会有所不同

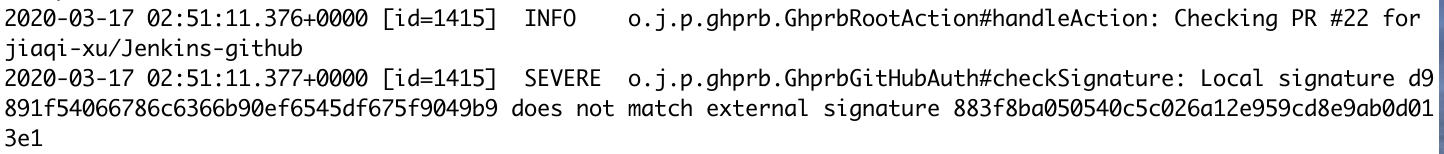
1. Shared secret，可以看一下说明，说的是

当其不为空的时候，github发送的web hook都要验证这个密钥; 所以在github上设置web hook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。  
  
这个地方提到了webhook ；其实我们之前说到过当我们提交一个新的pr到对应repo的时候，我们希望github能通知jenkins，从而触发相应的构建。Webhook就是起到这个通知的作用。我记得zijie之前发过一个repo名字为knockknock的github链接里也提到了这个webhook。

（回到ppt）  
这里分别给出了webhook未设置和webhook设置的secret与jenkins当前设置的secret不匹配时日志给出的报错信息。  
第一个报错信息，说的是request does not contain a signature(secret)

第二个报错信息，说的是local signature（webhook设置的secret）与external signature（jenkins上设置的secret）不匹配。

 webhook未设置secret



Webhook设置secret和jenkins设置的secret不匹配

（回到jenkins）

1. Credentials

这里介绍Jenkins支持的三种类型的crendential， 第一种就是username with password非常简单直接；第二种就是ssh username with private key，也很常用，配置完之后，需要将对应的公钥放到github 对应的repo上；第三种是secret text，他可以是personal access tokens或者是OAuth(open authentication)， 可以在github上developer settings上进行设置获取。  
  
设置crendential直接点这里的添加，然后进行配置就可以了。

其实设置credential目的是为了让jenkins有权限使用github 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

设置完毕后，可以进行检验。

勾选test basic connection to github

输入repo owner/name ： jiaqi-xu/Jenkins-github

勾选test permission to a repo

这里可以看到设置的credential是能连接github api并且对jenkins-github这个repo拥有相应的权限，这些权限是交互时所需要的。

在jenkins上全局配置完毕之后，接下来需要到jenkins-github 项目上配置 webhooks

（去github）  
点击jenkins-github 这个repo的settings，可以看到

Github上对webhooks的介绍  
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

说的是：当特定的事件发生的时候，可以通过webhooks通知其他服务，这我们的context下，这个服务指的就是jenkins；后面一句话说的是当特定的事件发生了之后，会发送一个post请求到你提供的每一个urls，这里的url可以理解为 hook url.

webhook其实是一个反向api，因为他提供了API规则，由我们自己来设计配置这个要使用的API

(给大家展示下配置页面，并停留在这里)

关于webhook配置页面，是这样的

具体配置方式如下：

1. 首先需要配置payload，也可以理解为hook url，这里需要填写一个jenkins server所在服务器端口的公网IP或者域名并加上后缀/ghprbhook/(github pull request builder hook)

当时在这个地方是遇到了问题，因为jenkins服务是运行在ai8的 8080端口上的，而且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；   
所以尝试了一些办法，最后是通过ngork这个tunnel工具，利用其反向代理的功能，成功的将localhost convert 成了公网域名.

可以给大家演示一下： 用ngork进行convert，然后将新生成的public domain用于更新webhook的hook url。

这种方式不太稳定，而且session只能持续8个小时，断开后或者session过期后，又要重新生成新的domain, 不是长久之计。

1. 选择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。
2. Secret 就是前面介绍全局配置时说到的要与其匹配的secret
3. 设置触发事件，可以根据需要设置，支持很多触发事件，根据我们的需求，选择pull request就可以了。这样当有新的pr的时候，就可以成功通知到jenkins并触发相应的构建
4. 设置完毕后，创建webhook就可以了

备注：下面的recent Deliveries可以查看webhooks每次具体发的请求

以上配置完毕后，  
（回到jenkins页面）

需要配置对应项目， 我这里已经建立了一个工程。可以一起来看下github\_pr\_test的配置：  
  
1. Github项目url填写我们jenkins-github repo的url

2. 源码管理这里需要进行一些配置，(打开高级配置)，通过这些设置可以让jenkins拉取相应的分支和pr其工作空间下。

3. 指定分支这个地方我配置的是$sha1 (xwa one)这个环境变量； git使用SHA-1 Hash算法加密生成的40位字符串来记录代表git中的每样东西(其实就是object)；这里设置为$sha1的话，jenkins就会知道这个指的是是对应的分支或者pr.

4. 构建触发器：

在构建触发器这一part，也需要设置一些东西

勾选 Github Pull Request Builder，因为我们用的就是这个插件

勾选 Use github hooks for build triggering，因为我们就是通过github 设置的webhooks来通知触发的，(打开选项说明)，这里提到了勾选了之后下面关于定时构建的参数就会不起作用了，这也正是我们想要的。

Admin List里可以填写github用户名，这样该用户就会有对应的权限

需要在White List填写相应的用户名，这样PR才能被自动build，不在白名单里的用户提交的pr需要管理员审核后才会build

5.构建，这里会执行项目的入口函数，当然也可以是所有的测试用例运行入口，可以是跑模型pipeline

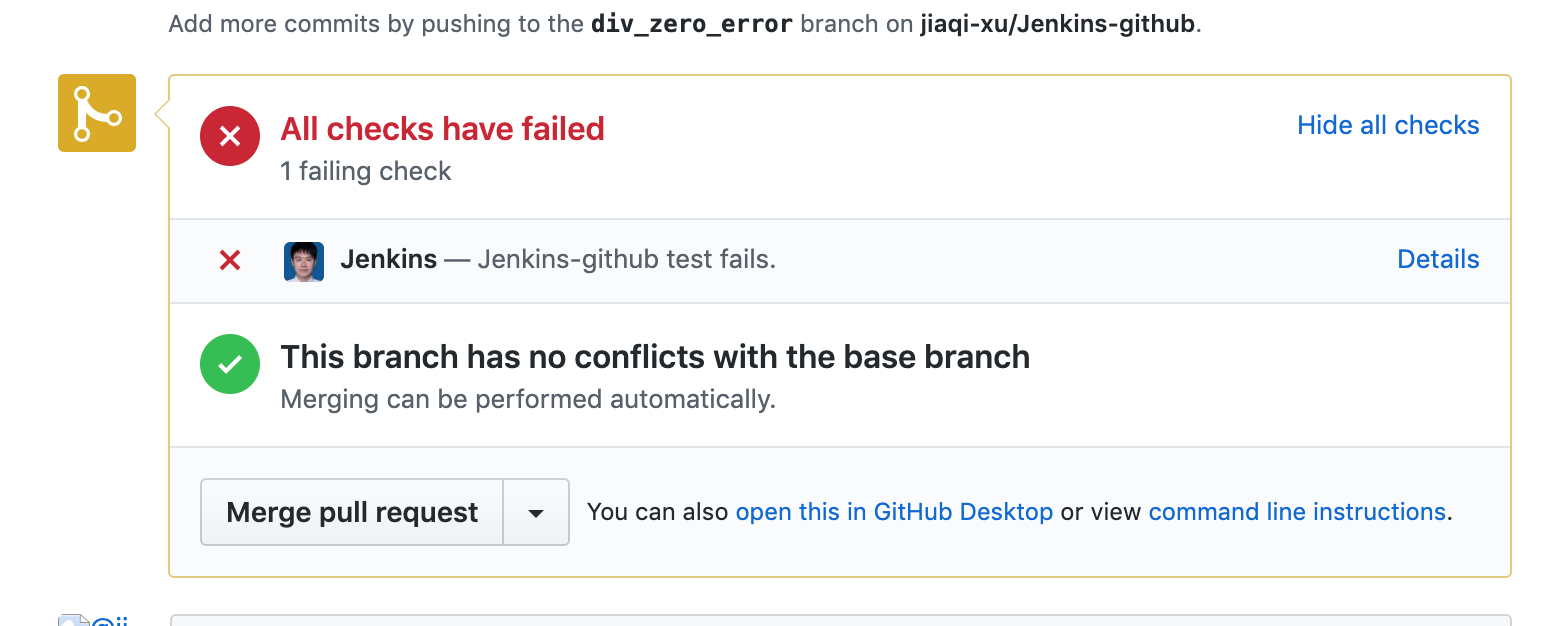
好了，现在我们都配置完毕了，可以提交一个pr看看效果

我们可以提前打开日志，这样当一个pr提交的时候我们可以看到具体jenkins和github之间是如何交互的（去ai8上， tail -f 10 /var/log/jenkins）。

/var/log/Jenkins



1. 可以看到secret验证通过
2. 并且可以根据SHA 值知道目前操作的是我们提交的pr(可以看下git log)



也可以回到jenkins看下对应的build的控制台输出：

在jenkins对应的工作空间下，拉取pull request和主分支，并且将 pr 到主分支，然后执行相关构建脚本。这里可以看到测试不通过，github上对应的也会给一个大大的察，表明测试没有通过。

（回到ppt）

## 总结

目前UL DP项目利用jenkins完成了  
每天定时运行clean data 和 相应品类的pipeline

接下来可能会做的的测试：

比如说

1. 测试模型的预测结果是否有负数，因为销量如果是负数是不合理的；还有如果预测的是箱数，预测结果却是小数，或者顾客要求我们给出3个预测文件，模型跑只有2个，都都是不合理的。
2. Kpi test: 看预测结果与设定的baseline比是否达标
3. Code pep8 check等等

关于UL DP代码的现状：

1. 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。

我们目前配置的clean data和pipeline只是起到一个打引号的“及时发现问题的作用”，能让我们趁早发现问题去解决。

1. 如果将来是计划在代码的主分支进行持续CI/CD，如果出现了错误，客户将会在AI产品里因为代码中断而得不到预测结果，那么这是需要被立即修复的。回滚代码也行，立即找人修复也行，总之，不能让客户在产品界面上一脸懵逼得不到预测值。

谢谢。各位有什么问题和建议吗～