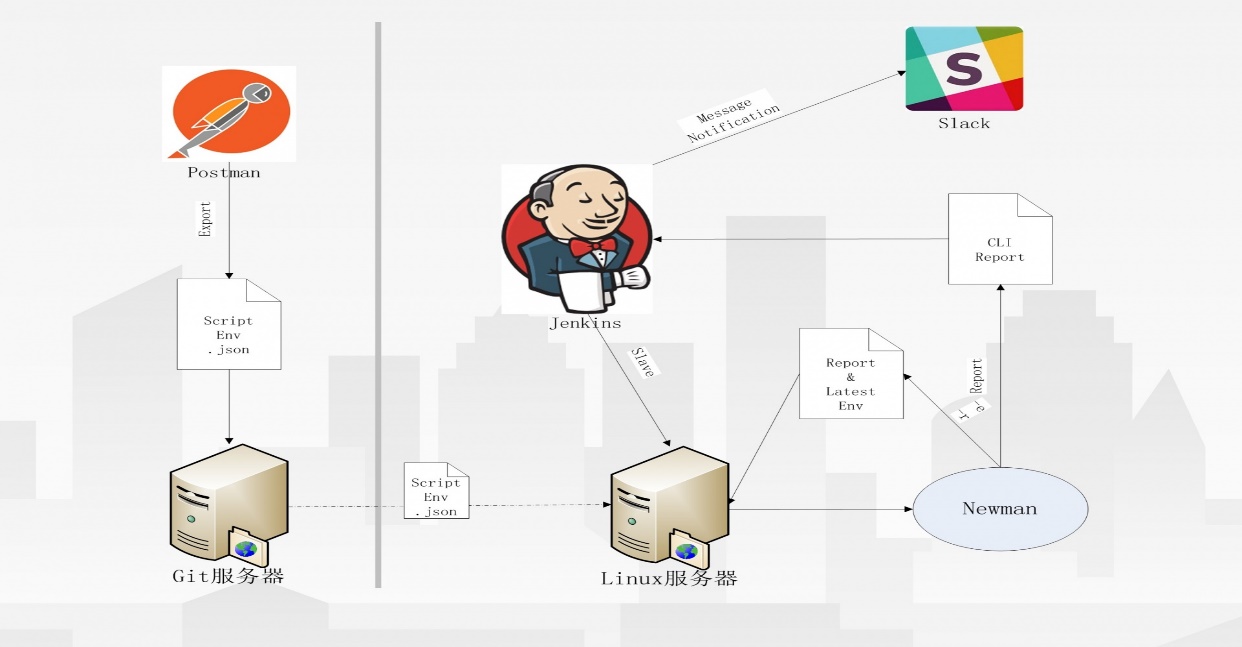
# 自动化测试

## 1.或者应用场景

本文要介绍的环境在我司已经投入使用，举个简单的真实使用场景，开发提供了300多个API，每个API都有各种参数，所以我们会先在postman中为这300多个API编写300\*n个testcase，然后在jenkins上跑；到此有人可能会问，都在postman中写好testcase了，怎么还要去jenkins中跑，岂不是多此一举？答案是否定的，试想，当开发一直在不停的维护这300多个接口时，难道需要开发每次都告诉我们测试人员该进行测试了吗？显然不是的，这就是jenkins+postman的自动化之处了，开发可以实现一个jenkins的上游job脚本（当接口代码有修改时触发我们这个testcase的下游job），当他们每次修改维护接口时不需要进行任何操作与交流，代码一旦提交jenkins就能自动化个性化地开始执行我们预定于的那些testcase了，然后我们只用关注最终的测试结果邮件即可，这样大大的提高了开发测试效率。

很少测试工程师用postman来测试API，于是写本篇文章分享下。前一个月用postman写了API自动化测试脚本，并用newman命令行运行所有测试案例，觉得没有达到全自动化测试，于是配置jenkins job来运行API自动化测试脚本，使API达到完全自动化测试！

接口测试 Postman+Newman+Git+Jenkins+Slack 接口自动化和监控图：



## 2.测试 Postman+Newman+Git+Jenkins+Slack的流程图

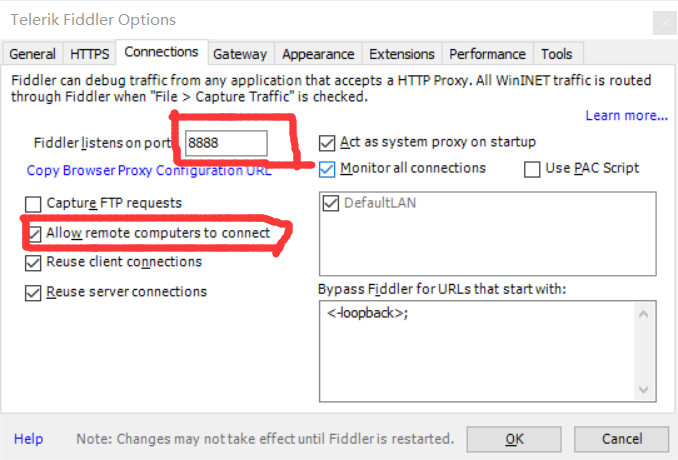
# 一.Fiddler与安卓手机的使用

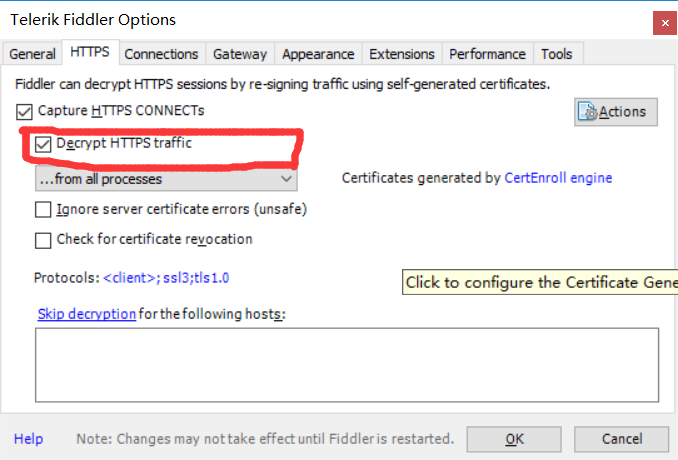
Fiddler是一个轻量级的http抓包工具，且有良好的用户界面，实用强大易上手。通过设置代理接受处理所有的http请求和响应，可以很轻松的对手机上的安卓应用进行抓包。

Fiddler下载地址：<http://www.telerik.com/fiddler>，个人喜欢在官网下载，安装之后打开Fiddler，

## 1.Fiddler使用

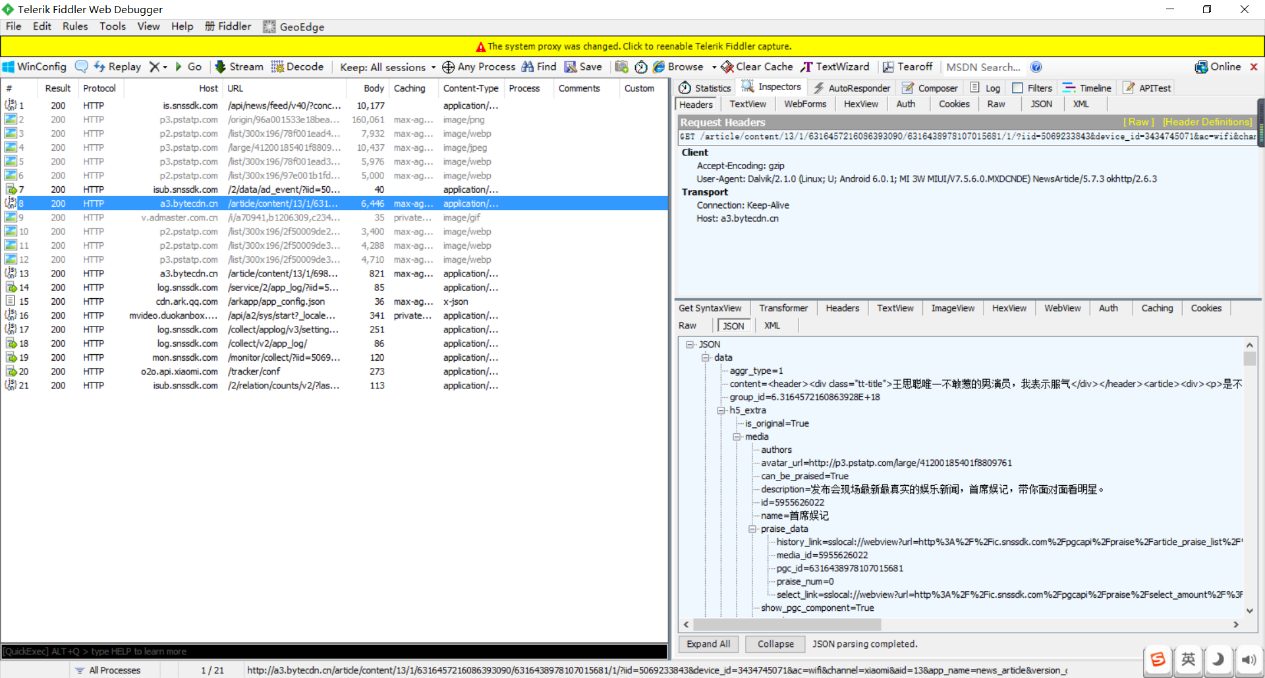
Tools—>Telerik Fiddler Options—>Connections,在这里设置允许远程连接，并设置自己的端口号，如下图所示：



这里必须保证选择了允许远程连接，否则你的手机是没有办法走Fiddler设置的代理的，也就没有办法抓到相应的数据包。端口号可以改变，但是得记住了，后面还会用到。如果需要抓取https请求，记得在HTTPS选项下设置允许https，同时会要求你安装一个证书，确认即可。如下所示：   


设置完成后，重启Fiddler才会生效。

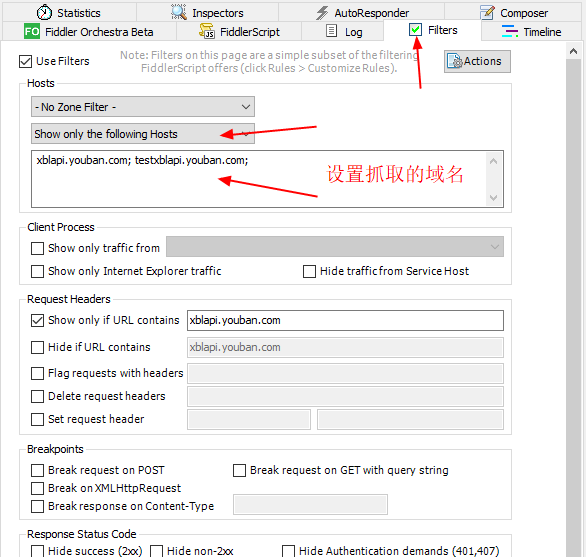
接着对手机进行设置，首先你得确保手机和电脑处在同一局域网，找到手机的设置—>Wlan,找到连接的局域网络，点开详情设置代理，默认是无代理的，这里我们就要指向Fiddler设置的代理服务器。代理选择手动，填入主机名和端口号。主机名填写你的电脑的ip地址，端口号填写刚刚设置Fiddler时写的端口号，确定，如下所示：   


设置完成后，尝试一下打开浏览器，随便打开一个网页，如果无法访问，尝试重启一下Fiddler，可能会提示你在手机端安装证书，安装即可。打开今日头条，加载完数据后，下拉刷新列表，会看到列表中多了几条请求，简单看一下其中一条，如下所示：   


左边简单描述了响应码，Host，Url，请求长度等信息，右边详细描述了请求与响应，我们可以轻松的了解整个http请求的过程。

步骤其实很简单，但也有很多需要注意的地方，

Fiddler设置抓取显示所要域名的包



## 2.这里列举一些容易导致抓不到包的因素：

* Fiddler记得打开远程连接
* 记得端口号要保持一致
* 在Fiddler未打开之前，给手机设置了代理，手机是无法访问网页的
* 更改Fidder设置之后记得重新启动
* 手机和电脑必须处于同一局域网内
* 手机有其他软件开启Vpn可能导致抓包失败（挂着vpn翻墙就会抓不到包。。）
* 打开Fiddler自动为浏览器设置代理，退出后会自动恢复
* 实在抓不到包可以尝试换个端口号

## 3.证书不对的情况解决



# 二.postman的使用

postman是一款功能强大的网页调试和模拟发送HTTP请求的工具

->创建Http请求

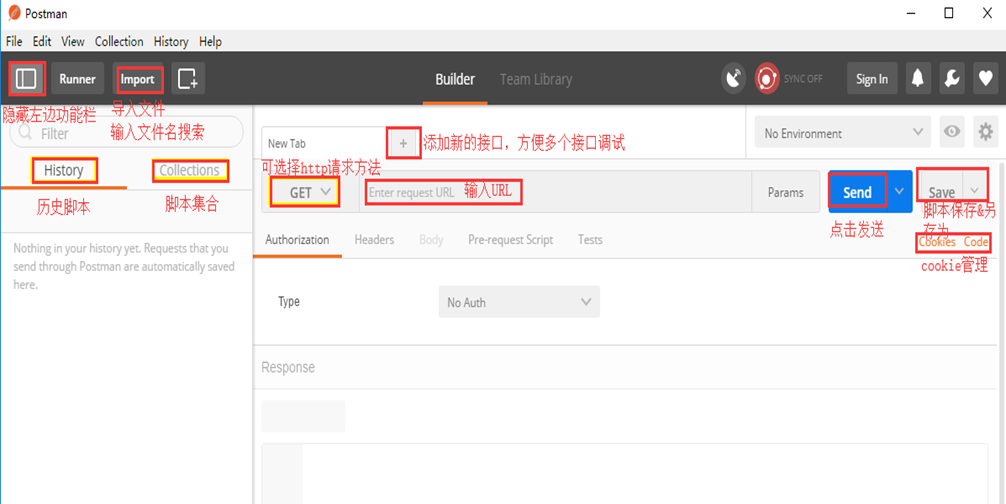
->抓取和分析现有网站请求

->管理Http请求(Collections)

->预处理数据和变量

->自动化测试

->导出和导入



## 1.post和get请求：

Form中的get和post方法，在数据传输过程中分别对应了HTTP协议中的GET和POST方法。二者主要区别如下：

? 1、Get是用来从服务器上获得数据，而Post是用来向服务器上传递数据。

? 2、Get将表单中数据的按照variable=value的形式，添加到action所指向的URL后面，并且两者使用“?”连接，而各个变量之间使用“&”连接；Post是将表单中的数据放在form的数据体中，按照变量和值相对应的方式，传递到action所指向URL。

? 3、Get是不安全的，因为在传输过程，数据被放在请求的URL中，而如今现有的很多服务器、代理服务器或者用户代理都会将请求URL记录到日志文件中，然后放在某个地方，这样就可能会有一些隐私的信息被第三方看到。另外，用户也可以在浏览器上直接看到提交的数据，一些系统内部消息将会一同显示在用户面前。Post的所有操作对用户来说都是不可见的。

? 4、Get传输的数据量小，这主要是因为受URL长度限制；而Post可以传输大量的数据，所以在上传文件只能使用Post（当然还有一个原因，将在后面的提到）。

? 5、Get限制Form表单的数据集的值必须为ASCII字符；而Post支持整个ISO10646字符集。

? 6、Get是Form的默认方法。 ?

GET 和 POST 的数据格式都是一样的：

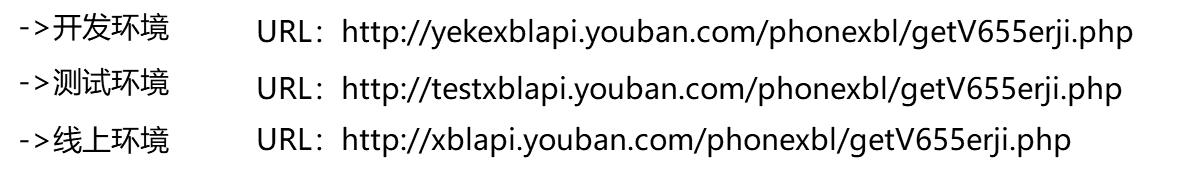
GET 支持的最大字节限制是 2048 Bytes

POST 支持的最大字节限制是 2GB

## 2.URL: 统一资源定位符

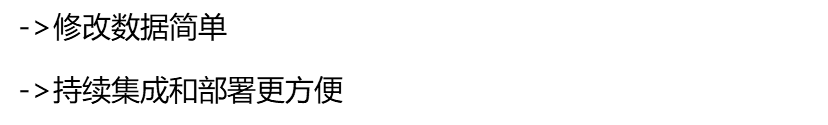
http://xblapi.youban.com/phonexbl/getV655erji.php

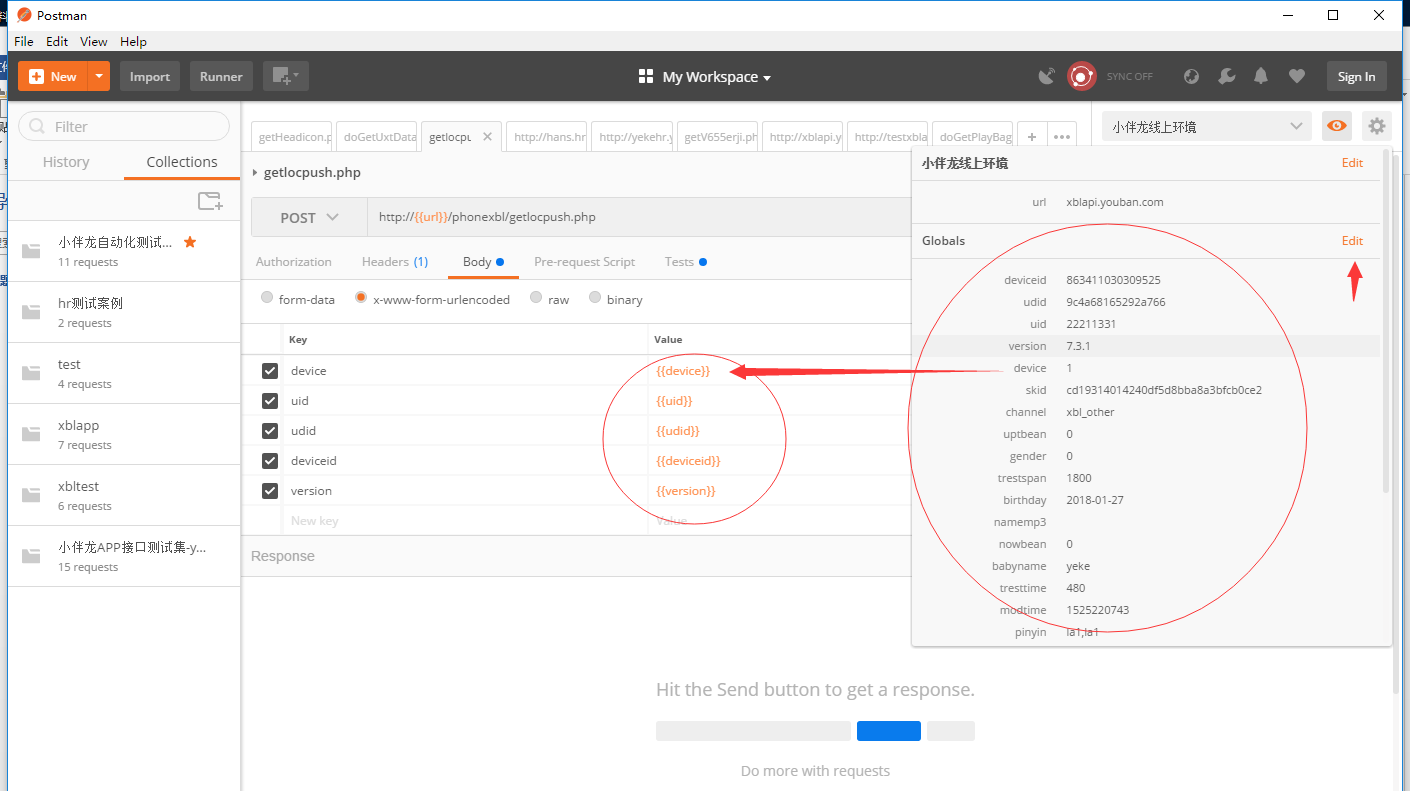
## 3.设置环境变量:





## 4.设置全局变量





## 5.Body主体

### 1）、form-data:

就是http请求中的multipart/form-data,它会将表单的数据处理为一条消息，以标签为单元，用分隔符分开。既可以上传键值对，也可以上传文件。当上传的字段是文件时，会有Content-Type来说明文件类型；content-disposition，用来说明字段的一些信息；

由于有boundary隔离，所以multipart/form-data既可以上传文件，也可以上传键值对，它采用了键值对的方式，所以可以上传多个文件。

### ****2）、x-www-form-urlencoded：****

   就是**application/x-www-from-urlencoded**,会将表单内的数据转换为键值对，比如,name=java&age = 23

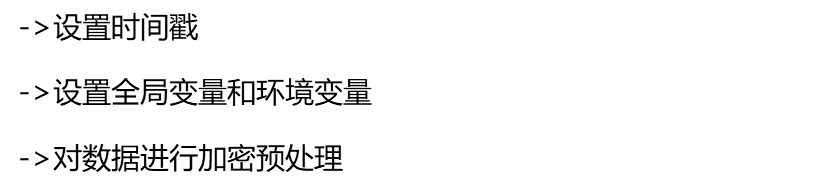
### 3）、raw

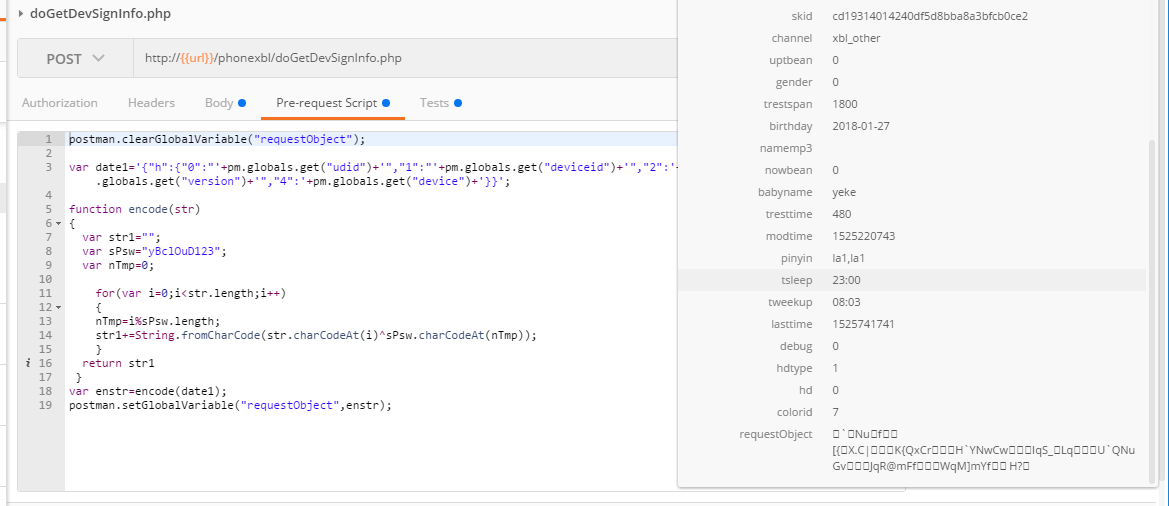
可以上传任意格式的文本，可以上传text、json、xml、html等

### 4）、binary

相当于Content-Type:application/octet-stream,从字面意思得知，只可以上传二进制数据，通常用来上传文件，由于没有键值，所以，一次只能上传一个文件。

## 6.pre-request script: 预处理





例子：

postman.clearGlobalVariable("requestObject");

var date2='{"h":{"0":"863411030309525","1":"9c4a68165292a766","2":22211331,"3":"7.3.1","4":1,"5":1525397491}}';

//var date="{\"h\":{\"0\":\"aaaaa111501\",\"1\":\"win501\",\"2\":5210620,\"3\":\"7.2.6\",\"4\":5}}";

var date1='{"h":{"0":"'+pm.globals.get("udid")+'","1":"'+pm.globals.get("deviceid")+'","2":'+pm.globals.get("uid")+',"3":"'+pm.globals.get("version")+'","4":'+pm.globals.get("device")+'}}';

function encode(str) //加密函数

{

var str1="";

var sPsw="yBclOuD123";

var nTmp=0;

for(var i=0;i<str.length;i++)

{

nTmp=i%sPsw.length;

str1+=String.fromCharCode(str.charCodeAt(i)^sPsw.charCodeAt(nTmp));

}

return str1

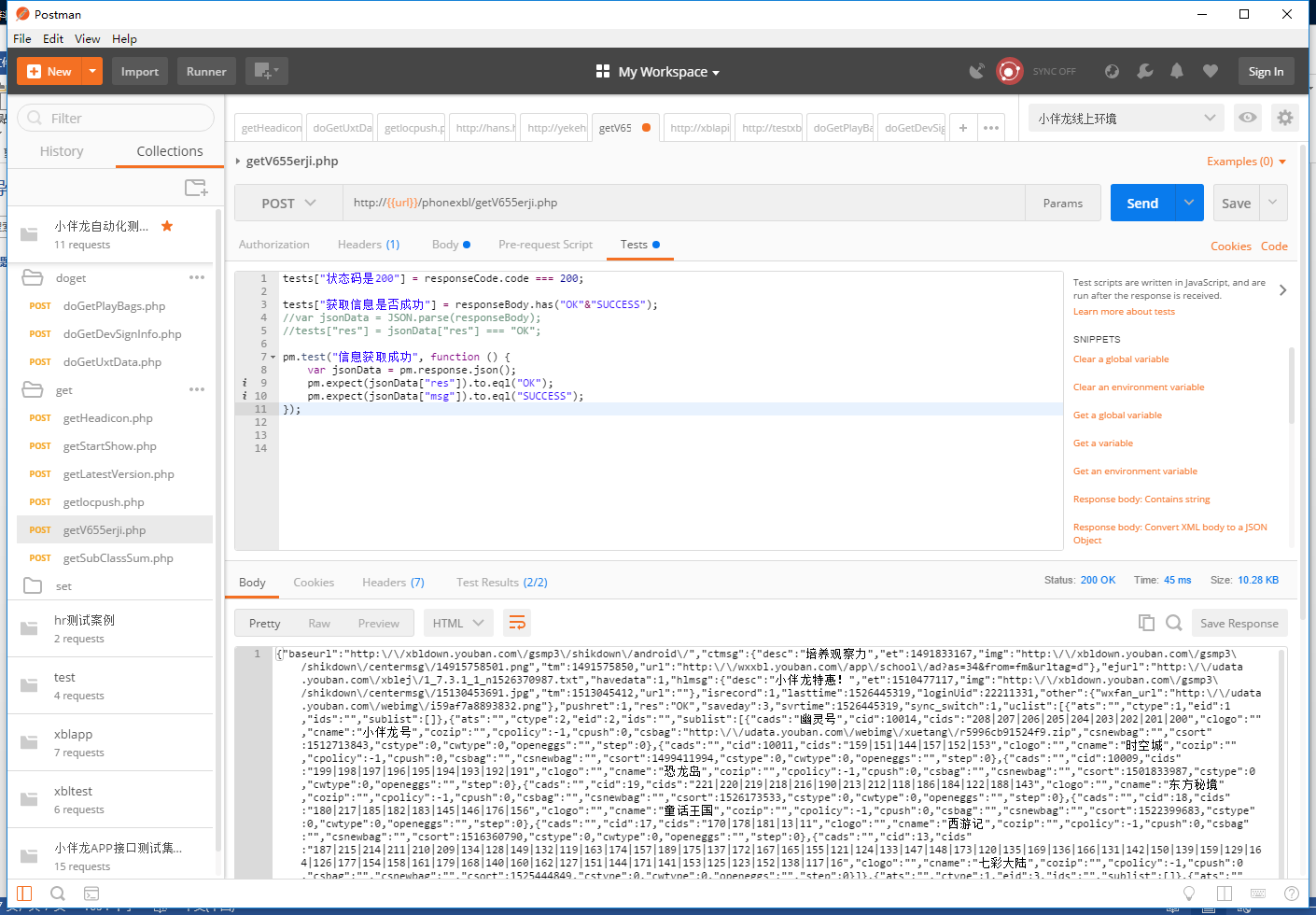
}

var enstr=encode(date1);

postman.setGlobalVariable("requestObject",enstr);

## 7.test: 断言

->对服务器的响应进行判断



.within

.within(start, finish[, msg])验证目标是某一个闭区间内，也就是大于或等于start参数，小于或等于finish参数。

expect(2).to.be.within(1, 3);

expect(2).to.be.within(2, 3);

expect(2).to.be.within(1, 2);

lengthOf也可以用于语句链式结构中，和.above, .below, .least, .most和.within一起使用时，会让这些方法去验证目标对象的length属性。不过，并不推荐这种写法，因为我们应该直接验证期待的具体数值而不是范围。

expect([1, 2, 3]).to.have.lengthOf(3);

expect([1, 2, 3]).to.have.lengthOf.above(2);

expect([1, 2, 3]).to.have.lengthOf.below(4);

expect([1, 2, 3]).to.have.lengthOf.at.least(3);

expect([1, 2, 3]).to.have.lengthOf.at.most(3);

expect([1, 2, 3]).to.have.lengthOf.within(2,4);

例子：

tests["状态码是200"] = responseCode.code === 200;

tests["获取信息是否成功"] = responseBody.has("OK"&"SUCCESS");

tests["获取信息是否成功"] = responseBody.has("OK");

pm.test("信息获取成功", function () {

var jsonData = pm.response.json();

pm.expect(jsonData["res"]).to.eql("OK");

pm.expect(jsonData["msg"]).to.eql("SUCCESS");

});

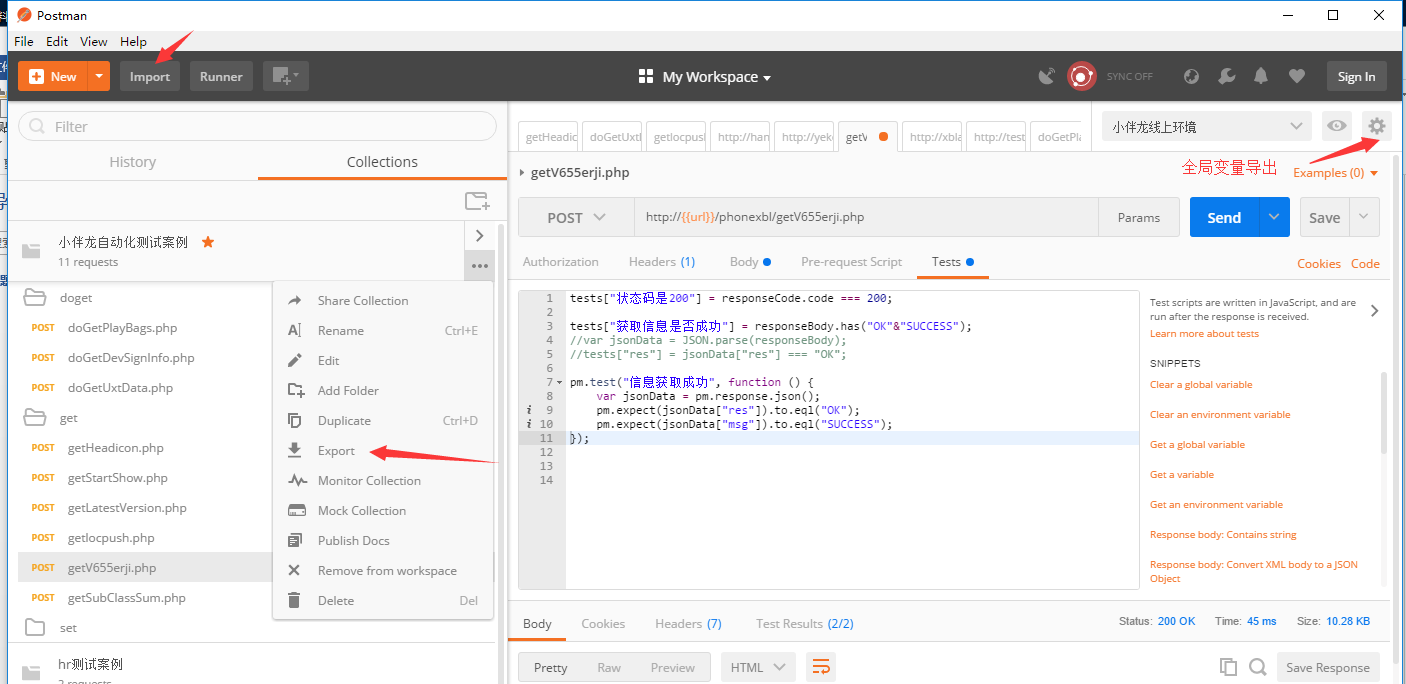
pm.test("弹幕信息数", function () {

var jsonData = pm.response.json();

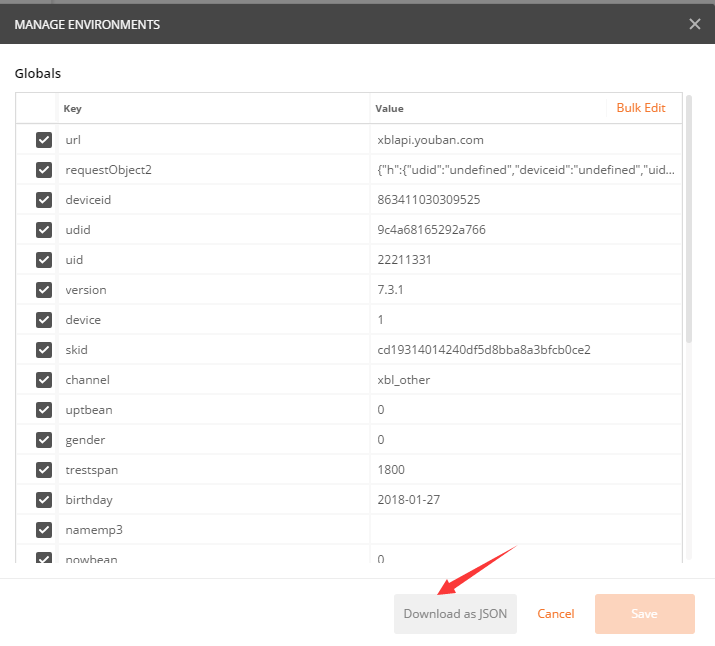
pm.expect(jsonData["data"]).to.have.lengthOf.below(500);

});

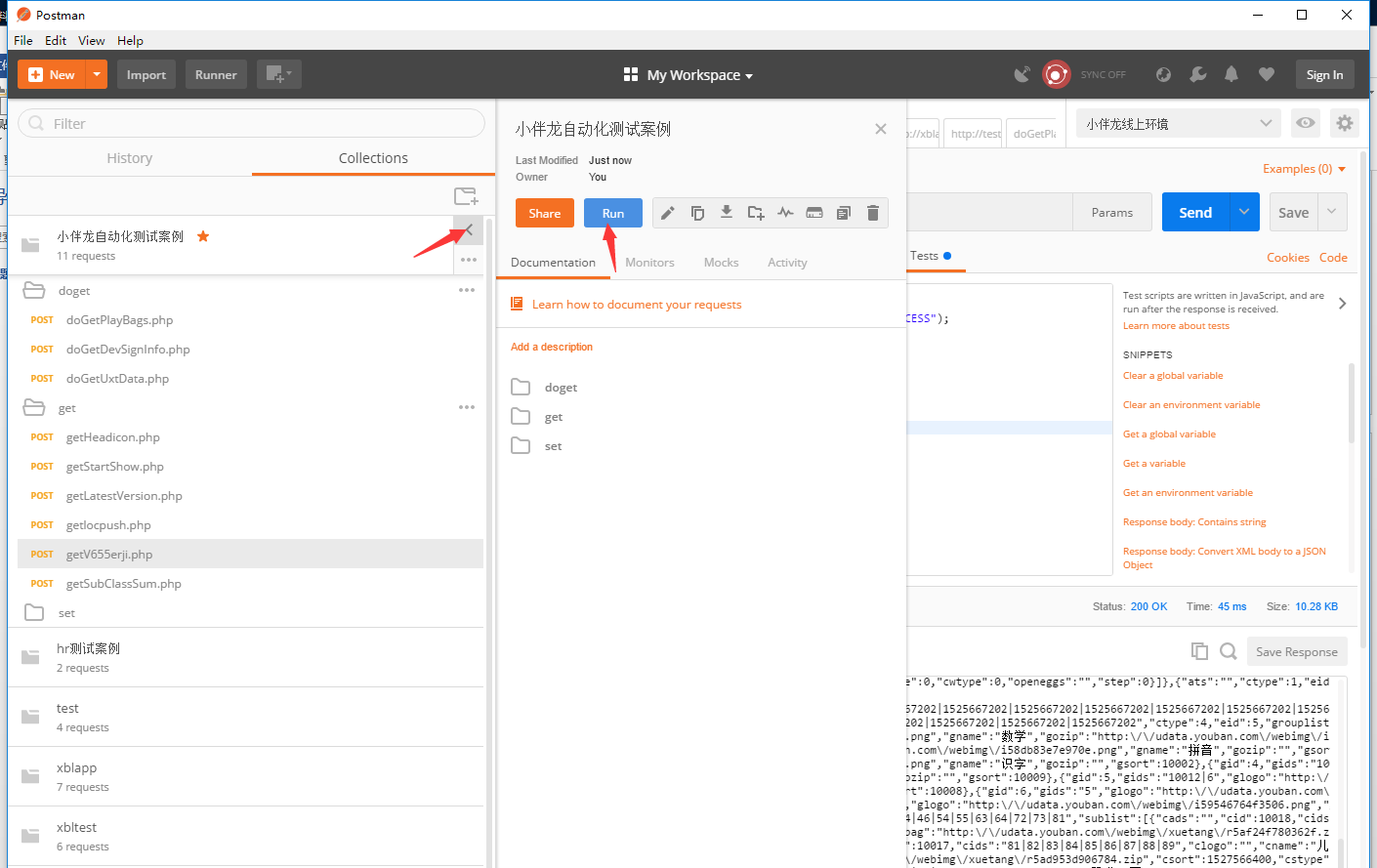
## 9.导入导出

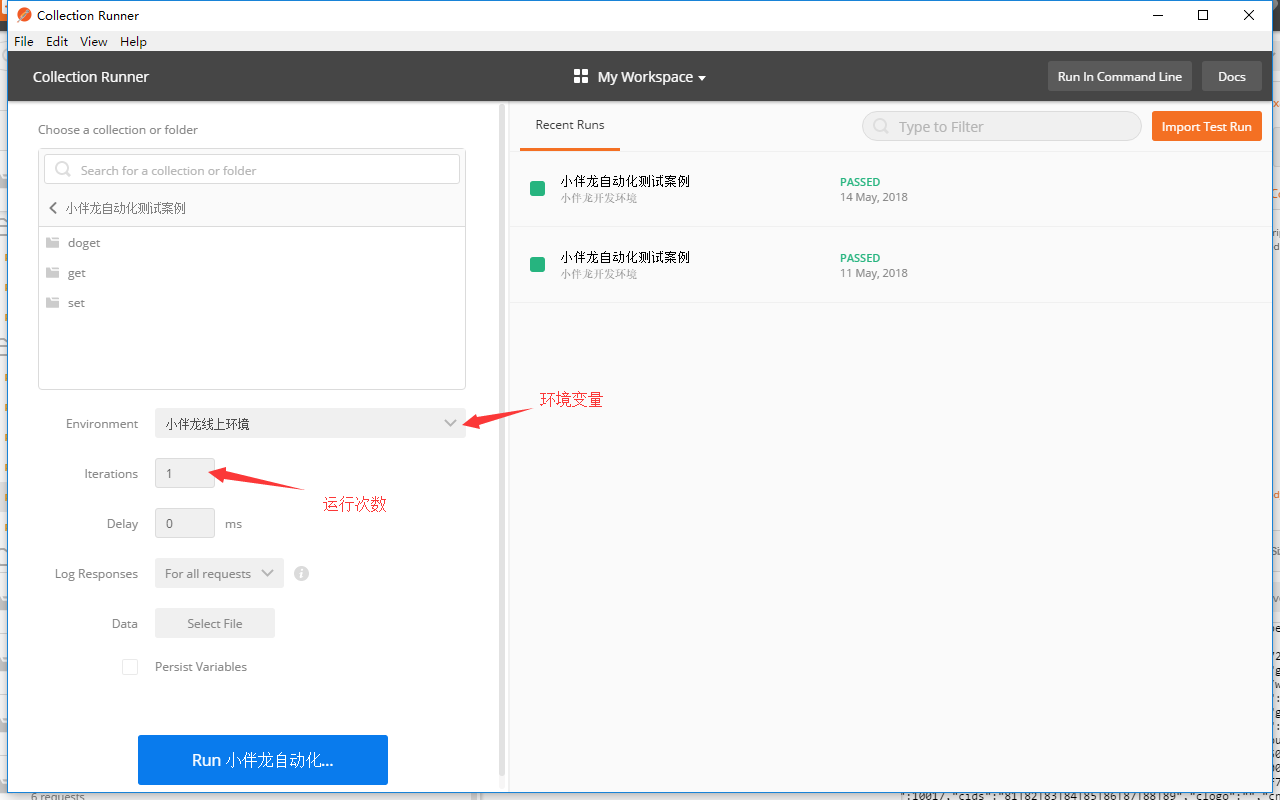


全局变量的导出

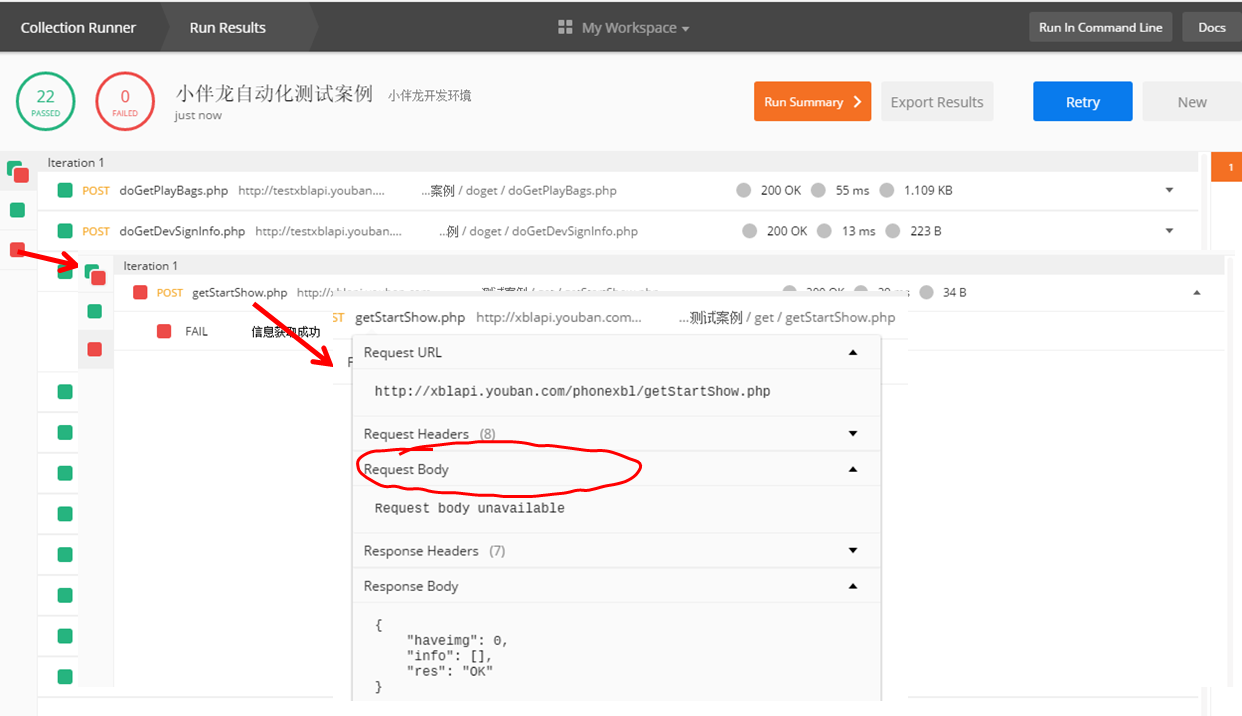


## 10.运行界面





## 11.运行结果



# 三．Newman+postman的使用方法

Newman是一款为postman而生的工具，在Linux或者windows下调用postman导出的测试脚本、全局变量和数据等

## 1. 安装的newman

# nodejs安装

> 由于编译安装需要 gcc++ 高版本这里选择二进制安装

官网下载：https://nodejs.org/zh-cn/download/current/

### 下载解压

cd /usr/local/

wget https://nodejs.org/dist/v8.11.2/node-v8.11.2-linux-x64.tar.xz

tar -xvf node-v8.11.2-linux-x64.tar.xz

mv node-v8.11.2-linux-x64 node

### 环境变量

cat >> /tmp/profile <<-EOF

#set for nodejs

export NODE\_HOME=/usr/local/node

export PATH=\$NODE\_HOME/bin:\$PATH

EOF

source /etc/profile

安装newman:

npm install -g newman

//建立软连接

ln -s /usr/local/node/bin/node /usr/bin/node  
ln -s /usr/local/node/lib/node\_modules/newman/bin/newman.js /usr/bin/newman

## 2.newman使用

查看newman版本

newman -v

Newman命令行介绍：

效用：

-h，--help输出使用信息

-V，--version输出版本号

基本设置：

-c，--collection [file]指定一个Postman集合作为JSON [文件]

-u，--url [url]指定一个Postman集合作为[url]

-f，--folder [文件夹名称]指定要从集合运行的单个文件夹。与-c或-u一起使用。

-e，--environment [file]将Postman环境指定为JSON [file]

--environment-url [url]指定邮差环境作为URL [url]

-d，--data [file]指定要使用json或csv的数据文件

-g，--global [file]将Postman全局文件指定为JSON [file]

-n，--number [数字]定义要运行的迭代次数

-i，--import [file]导入Postman备份文件，并保存集合，环境和全局变量。[文件]

-p，--pretty（与-i一起使用）在保存导入的集合，环境和全局文件时启用漂亮打印

-G，--exportGlobals [file]指定输出文件以在退出[file]之前转储Globals

-E，--exportEnvironment [file]指定输出文件以在退出[file]之前转储Postman环境

请求选项：

-y，--delay [数字]指定请求之间的延迟（以毫秒为单位）[数字]

-r，--requestTimeout [数字]指定一个请求超时（毫秒）为一请求（默认为15000 ，如果未设定）

其它：

-s，--stopOnError停止亚军当测试 案例失败

-j，--noSummary不显示摘要的每个迭代

-C，--noColor禁用彩色输出

-S，--noTestSymbols禁用符号在测试输出和使用PASS | FAIL代替

-k，--insecure禁用严格的ssl

-l，--tls使用TLSv1

-x，--exitCode即使在失败后仍继续运行测试，但在代码= 1时退出

-W，--whiteScreen白色屏幕的黑色文本

输出：

-o，--outputFile [file]输出应写入的文件的路径。[文件]

-t，--testReportFile [file]结果应该写为JUnit的文件的路径XML [file]

-H，--html将HTML报告导出到指定文件[文件]

-O，--outputFileVerbose [file]文件的路径，其中完整请求和响应应该被记录[文件]

使用该-n选项可以设置要为其运行集合的迭代次数。

$ newman -c mycollection.json -n 10 ＃运行集合10次

要提供一组不同的数据，即每次迭代的变量，可以使用-d指定一个json或一个csv文件。例如，如下所示的数据文件将运行*2*次迭代，每次迭代使用一组变量。

[{

“ url ”： “ http://127.0.0.1:5000 ”，

“ user\_id ”： “ 1 ”，

“ id ”： “ 1 ”，

“ token\_id ”： “ 123123 ”，

}，

{

“ url ”： “ http://dump.getpostman.com ”，

“ user\_id ”： “ 2 ”，

“ id ”： “ 2 ”，

“ token\_id ”： “ 899899 ”，

}]

$ newman -c mycollection.json -d data.json

-r <reporter-name>, --reporters <reporter-name>

指定测试报告的导出格式

目前支持四种格式：cli,json,html,junit

--reporter-{{reporter-name}}-{{reporter-option}}

支持同时导出多种测试报告格式

JSON --reporter-json-export <path>

HTML --reporter-html-export <path>

JUNIT/XML --reporter-junit-export <path>

示例

newman run -e ...... -n 1 -r cli,json,junit --reporter-json-export /home/admin/temp/json\_reporter\_171211.json --reporter-junit-export /home/admin/temp/xml\_reporter\_171211.xml

注释

newman执行指定脚本1次，并输出cli、json和junit三种测试报告，其中json报告保存至/home/admin/temp/json\_reporter\_171211.json，junit报告保存至/home/admin/temp/xml\_reporter\_171211.xml。

使用该-c标志，您可以运行位于文件系统上的任何收集文件。请参阅[集合文档](http://www.getpostman.com/docs/collections)以了解如何使用和下载集合。

$ newman -c mycollection.json

该-u标志允许您将邮递员收藏作为URL传递。你的集合可能使用环境变量。要提供一组随附的环境变量，请[从Postman中导出它们](http://www.getpostman.com/docs/environments) 并使用-e标志运行它们。

$ newman -u https://www.getpostman.com/collections/cb208e7e64056f5294e5 -e devenvironment.json

选项

纽曼提供了丰富的选项来定制运行。选项列表可以通过-h标志运行来检索。

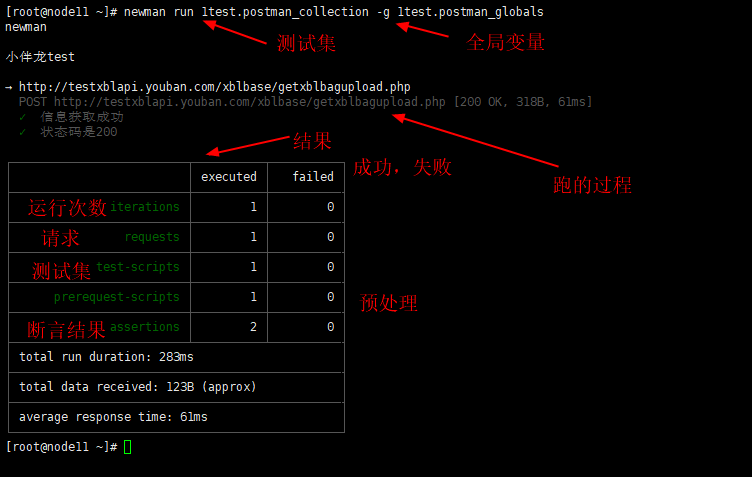
$ newman –h

## 3.例子：

1）：

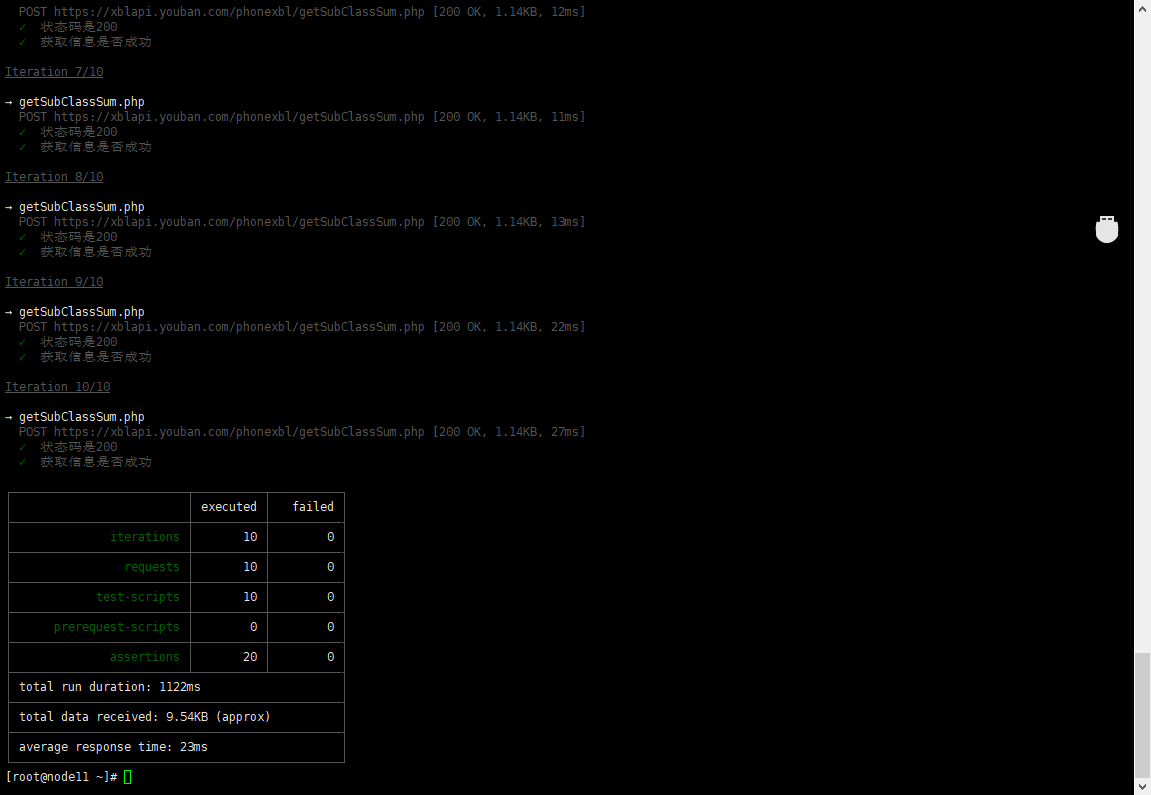
newman run 1test.postman\_collection -g 1test.postman\_globals

结果：



2）测试10次：

newman run 1xbl\_test.postman\_collection -g postman\_globals.json -n 10



3）全局变量+数据+运行10次，

newman run 1xbl\_test.postman\_collection -g url\_postman\_globals.json -d postman\_globals.json -n 10

4）小伴龙测试集测试

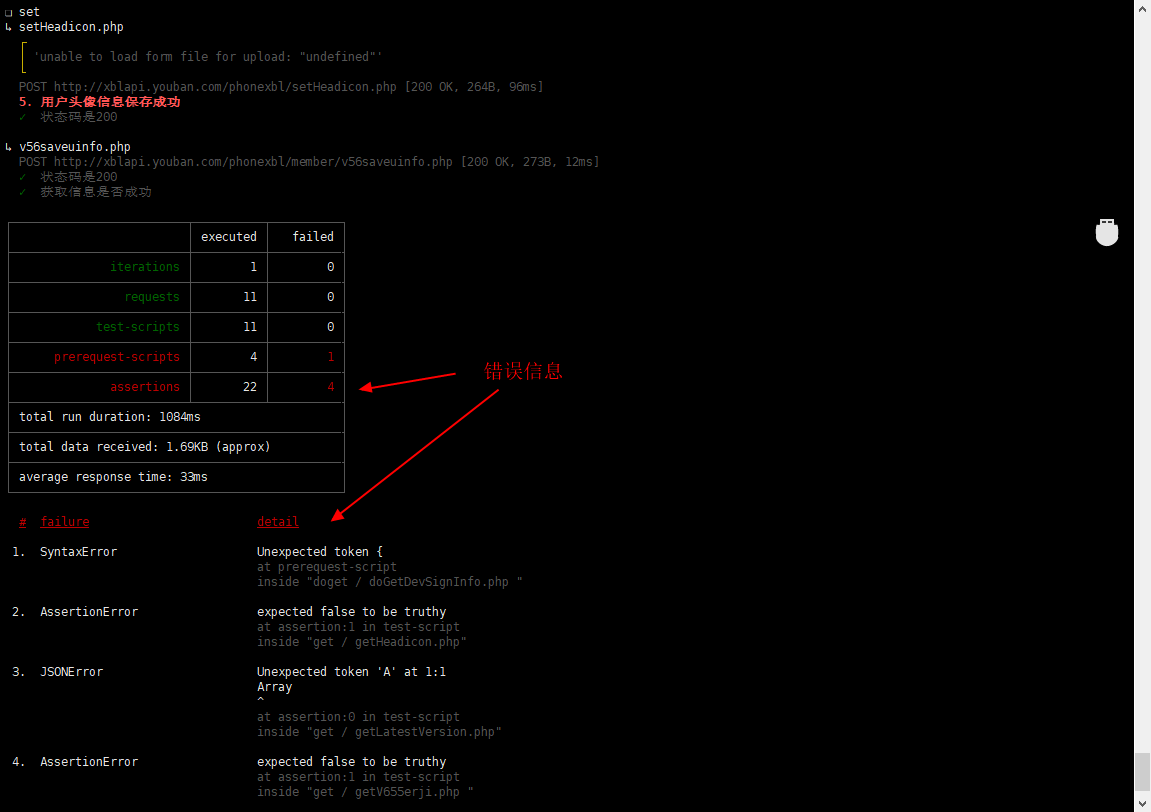
newman run xblapitest.postman\_collection -g postman\_globals.json -n 1

//把数据放在全局变量

or

newman run xblapitest.postman\_collection -g url\_postman\_globals.json -d postman\_globals.json -n 1

//把数据放在postman\_globals.json



5)把测试结果到出

newman run 1xbl\_test.postman\_collection -g postman\_globals.json --reporters json --reporter-json-export abc.json

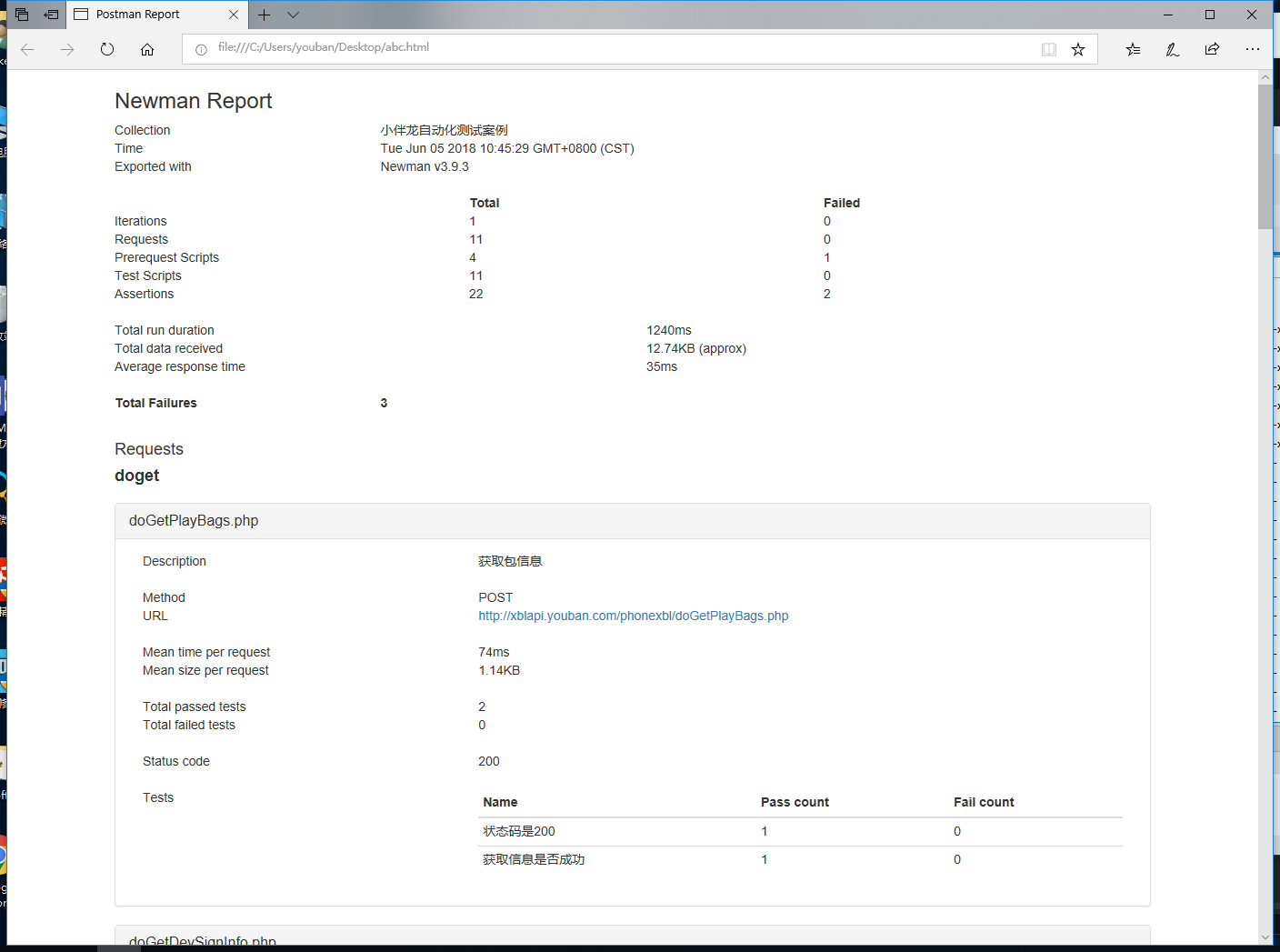
//导出json格式文件

newman run xblapitest.postman\_collection -g postman\_globals.json -n 1 --reporters html --reporter-html-export abc.html

//导出html格式文件

--reporters cli,html,json,junit --reporter-json-export jsonOut.json --reporter-junit-export xmlOut.xml --reporter-html-export htmlOut.html

//导出cli,html,json,junit 四种格式文件

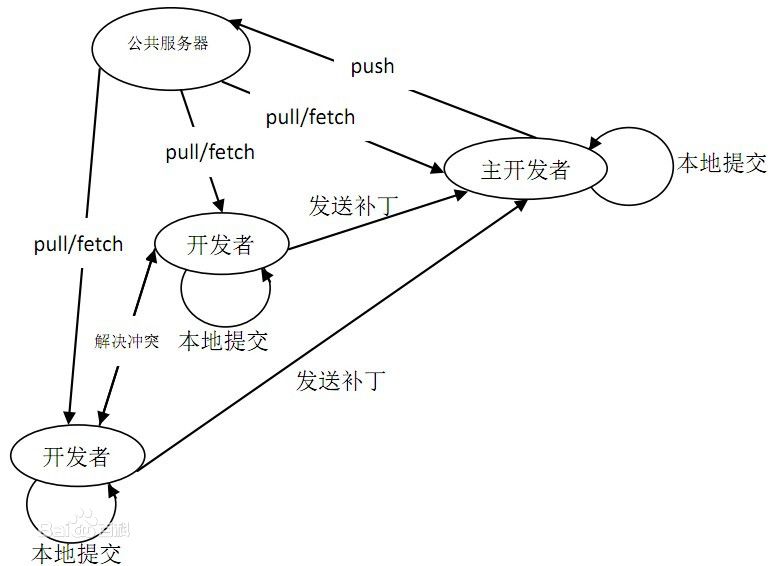


# 四．Git

## 1.git的介绍

分布式相比于集中式的最大区别在于开发者可以提交到本地，每个开发者通过克隆（git clone），在本地机器上拷贝一个完整的Git仓库。

下图是经典的git开发过程。



Git的功能特性：

从一般开发者的角度来看，git有以下功能：

1）、从服务器上克隆完整的Git仓库（包括代码和版本信息）到单机上。

2）、在自己的机器上根据不同的开发目的，创建分支，修改代码。

3）、在单机上自己创建的分支上提交代码。

4）、在单机上合并分支。

5）、把服务器上最新版的代码fetch下来，然后跟自己的主分支合并。

6）、生成补丁（patch），把补丁发送给主开发者。

7）、看主开发者的反馈，如果主开发者发现两个一般开发者之间有冲突（他们之间可以合作解决的冲突），就会要求他们先解决冲突，然后再由其中一个人提交。如果主开发者可以自己解决，或者没有冲突，就通过。

8）、一般开发者之间解决冲突的方法，开发者之间可以使用pull 命令解决冲突，解决完冲突之后再向主开发者提交补丁。

从主开发者的角度（假设主开发者不用开发代码）看，git有以下功能：

1）、查看邮件或者通过其它方式查看一般开发者的提交状态。

2）、打上补丁，解决冲突（可以自己解决，也可以要求开发者之间解决以后再重新提交，如果是开源项目，还要决定哪些补丁有用，哪些不用）。

3）、向公共服务器提交结果，然后通知所有开发人员。

优点：

适合[分布式开发](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E5%BC%80%E5%8F%91)，强调个体。

公共服务器压力和数据量都不会太大。

速度快、灵活。

任意两个开发者之间可以很容易的解决冲突。

离线工作。

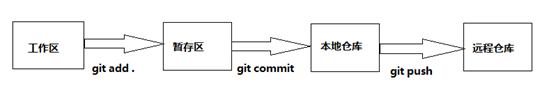
缺点：

资料少（起码中文资料很少）。 <--剑科就是来搞siao的！

学习周期相对而言比较长。

不符合常规思维。

代码保密性差，一旦开发者把整个库克隆下来就可以完全公开所有代码和版本信息。



## 2.git的使用

通常我们使用git是从clone代码仓开始，所以第一步，**使用git命令**：

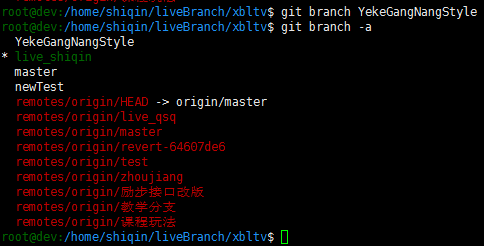
1. git clone repository-url (*e.g.* git clone https://github.com/chiphuyen/stanford-tensorflow-tutorials.git )；

拥有了棒棒哒的代码仓，我的分支我做主(bgm~)，恨不得马上进入第二步，创建自己本地分支，**使用git命令**：

1. git branch branchName (*e.g.* git branch YekeBranch 身为男孩子的我还是朴素点吧)；

此时，有了自己的分支，我们该注意点什么呢？当然是赶紧撸一发git代码啦哈哈哈哈哈~~~才怪！表急！先看看都有哪些分支再说，尤其是多人共同开发的时候必须要注意代码分支的合并与冲突。第三步，查看项目所有分支，**使用git命令**：

1. git branch –a (没有啥个人色彩的命令就给张图好了)；



Clone下来的项目自带master分支光环，不过由于有些代码仓的主人的设置，并不是所有的人都可以将本地的master分支代码push到远程代码仓哒！哈哈哈！主人大喊三声：别想聚众给我的项目写bug！

嗨呀呀，我，花美男Yeke，给自己创建了一个超酷炫的分支！YekeGangNangStyle！就问你酷不酷！可是我要怎么才能切换到我狂拽酷炫吊炸天的分支咧？唔~follow my step，第四步，切换分支，**使用git命令**：

1. git checkout branchName (e.g. git checkout YekeGangNangStyle 检出本宝宝的酷炫分支)；

好了，可以开始撸我的代码了！我是Yeke！来自数据中心小组！希望大家记得我的美！oh不，我的帅！！@#￥%……&\*（）——（几个小时过去。。。。。。）撸代码的时光总是过得飞快。哈哈哈，现在要怎么样才能把我新添加的以及更新的代码布到测试服务器上去检验一下咧？Yeke告诉你，这个可是一条龙服务，第五复合步，**提交代码几个步骤的git命令**如下：

1. git status (查看所有修改过的文件)；
2. git add –A (将修改的所有文件添加到棉花糖里)；
3. git commit –m “美男子Yeke的git教学” –a (千万别忘了写注释，不写就送Yeke香吻一打)；

走到了这一步，是很重要的一步，单人开发环境下，你想怎么玩就怎么玩。然鹅！如果不止本美男子一个人呢~！？多人更新同一个分支的代码可是很有可能出现代码冲突的情况哟！为了避免这种情况，严重严厉严格建议大家，在push你的代码到测试分支之前，先将远程代码拉取下来，如果有冲突就解决冲突，没有冲突合并更新的代码。老铁，稳！怎么拉去更新咧？问我？不如问google，第N步，**拉取远程分支更新git命令**如下：

1. git pull origin test:YekeGangNangStyle (多人开发必备步骤)；
2. git fetch (不拉去仅查看远端的更新！)

## 3.注意

1）注意：开发过程中，必须创建自己分支进行功能开发，不允许直接在master分支中进行功能开发、修改、删除等操作。以免误操作或操作出错等情况出现，污染了远程仓库的主干分支master，导致功能代码无法继续使用，也会影响到其他人的使用。

2）注意：在工作中不要使用命令进行操作，且删除分支时需要谨慎进行，因为一但删除远程分支，便无法进行回滚。

// find . -name ".git" | xargs rm -Rf

# 五 svn

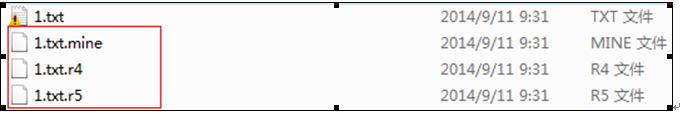
## 1. SVN更新(SVN Update):

**1. SVN update:**

更新本地代码与SVN服务器上最新的版本一致，只要在需要更新的文件夹上点击右键或者在文件下空白处点击右键，选择”SVN Update” (获取指定版本中的内容，点击右键执行SVN菜单中的“Update to reversion“)，就可以了。

## 2. 冲突文件的解决：

**1. 对于每个冲突的文件Subversion在你的目录下放置了三个文件：如下：**



**2. 为什么会产生冲突呢？**

|  |
| --- |
| 为什么会产生冲突代码呢？原因很简单就是因为不同的人，同时修改了同一个文件的同一个地方，  这时候，他提交了，我没有提交，我就提交不了，这个时候我们要进行先更新，然后在进行提交即可，  那如果产生冲突，会生成如上3个文件。 |

**3. 解决方案如下：**

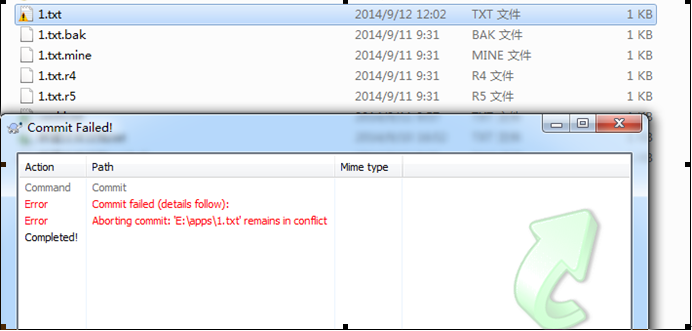
1) 首先我们可以看下1.txt代码如下：

|  |
| --- |
| <<<<<<< .mine aaaasdf11222333 dderderder ======= b>>>>>>> .r5 |

2) 然后我去掉多余的代码，1.txt变成这样：

|  |
| --- |
| aaaasdf11222333 dderderder |

3）进行提交，还是提交不了，如下所示：



4）为什么？因为冲突会产生上面的三个文件，有上面3个文件存在肯定提交不了，这三个文件代码及解释如下：

|  |  |
| --- | --- |
| 1. txt.mine | 是冲突前自己的文件（如：aaaasdf11222333 dderderder） |
| 1.txt.r4 | 是冲突前本地的版本文件（如：aaaasdf11222333） |
| 1.txt.r5 | 是别人赶在你之前提交的版本（如：b） |

其中：

|  |  |
| --- | --- |
| <<<<<<<<.mine .....======= | 之间的代码是你自己的 |
| ======......>>>>>>>.r5 | 是别人与你冲突的代码部分 |

这样就不难理解为什么会产生冲突这种奇怪的东西了，因为你们修改的同一块代码，当然会产生冲突。

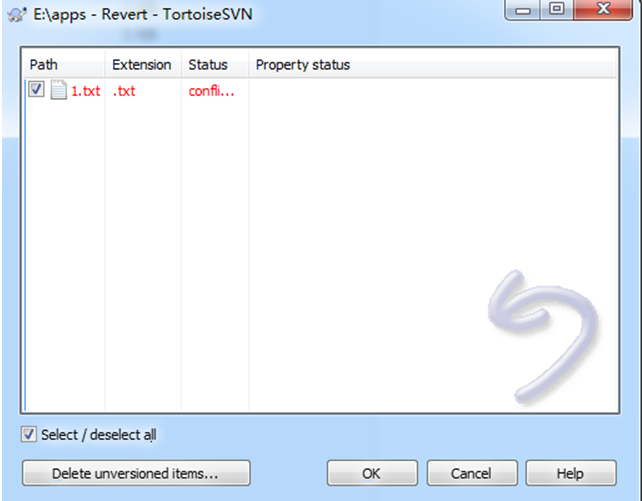
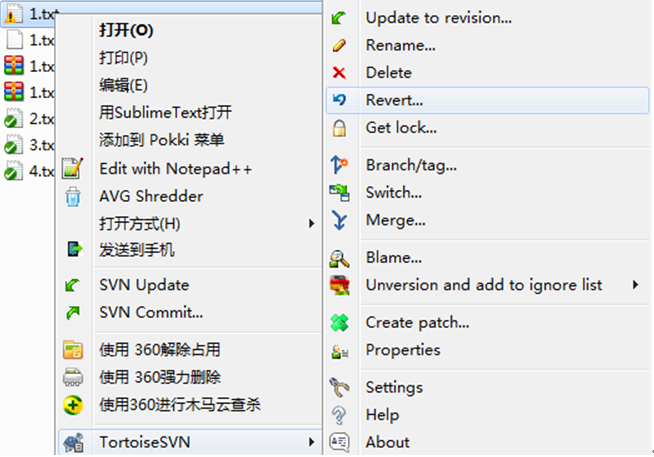
**4. 新的解决方案一：**

**[cpp]** [view plain](https://blog.csdn.net/u013354805/article/details/48489757) [copy](https://blog.csdn.net/u013354805/article/details/48489757)

1. <<<<<<< .mine
2. 6666666666666600000
3. =======
4. 66666666666aaaaaaaaaa666
5. >>>>>>>

|  |
| --- |
| 前面说过  <<<<<<< .mine …… =======     ……之间的代码是我未产生冲突之前修改的代码，     ======= ………>>>>>>> .r16 这中间……的代码是别人与我冲突代码的部分，**从上面的代码可以看到 aaaaaaaaa是我同事新增的 ,00000是我后增加的。** |

1）使用revert(回滚)操作，该操作表示用户放弃自己的更新代码，然后直接提交，这个时候你的代码就会使服务器上最新的代码，即A用户提交的新代码，你的代码不会被提交，如下所示：



**2）**点击ok按钮后 可以看到其他三个文件都自动删掉了，1.txt代码变成如下代码：

**[cpp]** [view plain](https://blog.csdn.net/u013354805/article/details/48489757) [copy](https://blog.csdn.net/u013354805/article/details/48489757)

1. 66666666666aaaaaaaaaa666

**也就是a用户提交的代码，我自己更新的代码需要自己动手复制进去即可提交commit。**

**5. 新的方案二：**

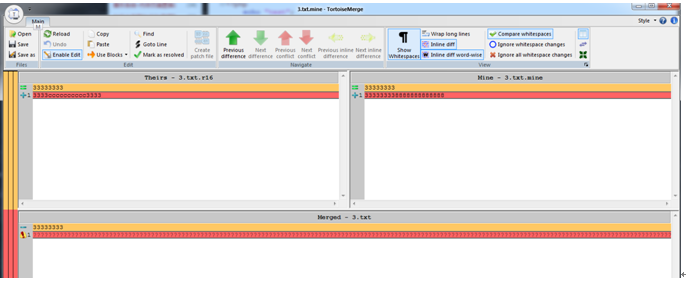
**1）假如我现在3.txt产生冲突代码如下：**

**[cpp]** [view plain](https://blog.csdn.net/u013354805/article/details/48489757) [copy](https://blog.csdn.net/u013354805/article/details/48489757)

1. <<<<<<< .mine
3. 333333338888888888888=======
5. 3333cccccccccc3333>>>>>>> .r16

通过第一点我们知道，333333338888888888888这个内容是我修改后，未产生冲突之前的内容，3333cccccccccc3333这个代码是A用户提交的代码，从上面得知 A用户新增内容是ccccccc，而我新增的内容是8888888。

那么第二种解决方法如下：  
选择文件->右键Editconficts：这种方法需要冲突双方经过协商之后将代码更改统一之后再提交。不仅解决了冲突而且还保证了代码是正确的，因为只有一方的代码被提交。



 如上图所示，

1）红色的部分是冲突代码：

|  |  |
| --- | --- |
| theirs | 表示当前服务器端最新的代码 |
| Mine | 表示自己修改后的代码 |
| Merged | 表示合并后的代码 |

2）点击红色后右键选择：use this text block就可以将该部分代码作为合并后的代码：

**注意：**

|  |
| --- |
| 接下来再说说由于冲突导致重要代码被覆盖的情况。冲突发生时如果采取的措施不对可能会导致部分代码  丢失，如果想要还原之前的代码也很容易。  选择文件->右键选择show log在这里面你可以看见之前提交的所有版本，找到你想要恢复的版本右键选择  revert to this version 就可以恢复了. |

## 3.防止冲突的正确方法



1. 拉去分支代码到本地D/E，这个地方的代码不做任何修改。
2. 拉去分支代码到个人开发环境，修改代码只在这个地方。
3. 开发完成之后，更新本地D/E盘的代码，（多人开发，别人可能已经提交过了，需要更新，再合并）。
4. 利用BCompare工具，将个人代码合并代码到本地D/E盘。
5. 将本地D/E盘commit上传到SVN。

SVN回退到历史版本

svn回到历史的某个版本

在代码的编写过程中，难免有些错误需要修改，或者想从以前的文件进行代码修改，这样就涉及到版本的追踪，如果你以前提交时日志写的非常清楚，那版本追踪回滚起来就事半功倍、得心应手。下面介绍几种版本回滚的办法：

1.推荐的一种方法是，直接export一个你需要的版本，然后用你export的版本覆盖你的最新的版本，这样你就可以不丢失你新建的文件，同时获得最新的SVN版本控制。 操作步骤：TortoiseSVN→Show log→选中需要回滚的版本→右键→Export。 之后将修改的文件覆盖到你的最新版本，commit即可。

2. 若是你编辑了工程，在没有提交的前提下，你想放弃这些修改，你可以直接选择TortoiseSVN→revert就可以更新到工程的最新的版本。

3. 若是你想退回到某一个版本，你就可以直接选择TortoiseSVN→update to reversion,这样我们就可以把我们的版本回退到你选中的版本去，这种情况下SVN并没有显示出有什么冲突，并且新建立的文件也还在，但是在这种情况下你并不能直接在你回退后的版本上进行编辑，因为SVN的版本控制还是在最新的主干上。我们需要update并解决冲突。

4.你可以直接选择revert changes from this revision，这样的话你可以直接解决冲突并提交。不过这种方法的不足是，你新建的文件都没有了，整个工程都回退到之前的版本了。 5.可以从日志中回滚到你需要的版本，从日志中选中你需要的版本，然后Update item to reversion就好了，这种情况下SVN并没有显示出有什么冲突，并且新建立的文件也还在，但是在这种情况下你并不能直接在你回退后的版本上进行编辑，因为SVN的版本控制还是在最新的主干上。我们需要update并解决冲突

# 六．Jenkins

## 1.jdk安装

安装脚本

#!/bin/bash

install\_java8()

{

mkdir /usr/local/java

tar -zxvf jdk-8u161-linux-x64.tar.gz -C /usr/local/java/

cp /etc/profile /etc/profile.bak;

echo "#set java environment" >>/etc/profile;

echo "JAVA\_HOME=/usr/local/java/jdk1.8.0\_161">>/etc/profile;

echo "JAVA\_BIN=/usr/local/java/jdk1.8.0\_161/bin">>/etc/profile;

echo "PATH=\$PATH:\$JAVA\_BIN">>/etc/profile;

echo "CLASSPATH=\$JAVA\_HOME/lib/dt.jar:\$JAVA\_HOME/lib/tools.jar">>/etc/profile;

echo "export JAVA\_HOME JAVA\_BIN PATH CLASSPATH" >>/etc/profile;

source /etc/profile;

sleep 2

java -version

}

install\_java8

## 2.jenkins安装

#!/bin/bash

wget <https://prodjenkinsreleases.blob.core.windows.net/redhat-stable/jenkins-2.107.3-1.1.noarch.rpm>

rpm -ivh jenkins-2.107.3-1.1.noarch.rpm

/etc/init.d/jenkins start

chkconfig jenkins on

## 3.jenkins的使用

# 七．charles

破解：

https://www.jianshu.com/p/46d29e60dd1b

https://tools.zzzmode.com/mytools/charles/

教程：

<https://www.jianshu.com/p/beaa56846f50>

## 牛逼的断点功能

Charles另一个非常实用的功能，对于开发者和测试人员来说，堪称神器。Charles能够断到发送请求前（篡改Request）和请求后（篡改Response）。

场景：ajax发送请求，我们需要测试接口的各种边界情况，比如出错、超时等表现，Charles的断点+随意篡改，非常方便测试。

**Charles 主要的功能包括**：

1. 截取 Http 和 Https 网络封包。
2. 支持重发网络请求，方便后端调试。
3. 支持修改网络请求参数。
4. 支持网络请求的截获并动态修改。
5. 支持模拟慢速网络。

# 参考文献

## 接口测试 Postman+Newman+Git+Jenkins+Slack 接口自动化和监控

<https://testerhome.com/topics/10523>

## Fiddler工具

<https://blog.csdn.net/u013224148/article/details/46136731>

## 3. postman文档

<https://www.getpostman.com/docs/postman/scripts/postman_sandbox_api_reference>

## 4.Chai语言

<http://www.chaijs.com/api/bdd/>

## 5.newman工具

https://blog.csdn.net/xiaosongbk/article/details/64921333

## 6.git教程

<http://www.runoob.com/manual/git-guide/>

<http://www.runoob.com/git/git-tutorial.html>

[1]. Git 命令：<https://gist.github.com/guweigang/9848271>

[2]. SVN命令：<http://www.cnblogs.com/yue-/p/6413412.html>

[3]. 代码版本管理汇总<http://blog.csdn.net/wanglin_lin/article/details/48845343>

[4]. 猴子都能懂的git入门：<https://backlog.com/git-tutorial/cn/intro/intro6_2.html>

[5]. 开源分布式版本控制工具git：<https://www.ibm.com/developerworks/cn/opensource/os-cn-tourofgit/>

[6]. Git与svn的区别：<https://www.jianshu.com/p/bfec042349ca>

[7]. Git官方文档：<https://git-scm.com/docs>

[8]. Git 与svn间的五个差异：<https://my.oschina.net/u/1580062/blog/214850>

[9]. 版本控制发展史：<https://blog.csdn.net/wyj19950908/article/details/78763190>

[10]. 常用git命令清单：<http://www.ruanyifeng.com/blog/2015/12/git-cheat-sheet.html>

## 7.svn

https://blog.csdn.net/u013354805/article/details/48489757

## 8.jenkins

<https://www.jianshu.com/p/b524b151d35f>

https://www.yiibai.com/jenkins/