接口自动化

###代码流程

1、基于python3+rquests+pytest做的二次开发。使用pytest启动主python文件，调用pytest.main方法，开始程序。

2、读取yaml配置文件，根据mysql配置信息、env、devops等参数，读取mysql存储的case(大概是450多个线上接口，基本上都是读接口)

3、替换case的中json格式的请求参数，包括日期、版本号、用户信息等参数，根据请求方法重新拼装后，重新计算验签，使用request方法发送请求。

4、根据request的返回结果，diff数据库中存储的服务器返回，如果不一致，发送钉钉报警，进行人工核验。

5、将运行记录存储入库，方便回溯。

###代码部署

1、部署一：在linux服务器上，通过crontab命令，每20分钟执行一次。(避免频繁请求)

2、部署二：在linux服务器上，上线发布系统在上线过程中，通过调用压测平台对外提供的一个服务接口，更新接口自动化的yaml文件的env、devops，根据域名进行查询case进行立时回归，并将运行记录存储入库，方便回溯。

3、通过给运维同学沟通，nginx配置查看运行报告。

###难点：

1、技术上：各服务的请求方法不同一，如请求参数为xml格式；一个服务都是post请求，却涉及到json、form-data、x-www-form-urlencoded格式；

主站服务、第三方合作平台有多套验签等。

2、工作量上：今年服务从php转go，需要了解新接口的功能，并添加对应的case，删除原有case，需要对所有服务有一个基本的了解。

3、考虑过流量回放自己增加case， 但是出现了自动添加后影响case正常运营的情况，放弃了此方案，采用了逐条添加的方案。(下单场景影响了三方对账)

### 目前存在问题：

1、部分接口case缺失。

2、对于返回结果检测过于宽松。

###常用的一些参数说明

1、请求重试： requests.adapters.DEFAULT\_RETRIES = 5 # 增加重连次数

2、pytest：pytest.main 中添加 ff参数：

-s ：显示打印内容

-k ：执行某个关键字的用例

-x : 遇到错误时停止

-v ：详细结果

-q ：极简结果显示

--reruns=2 失败重试

--junit-xml=path 输出xml文件格式，在与jenkins做集成时使用

--result-log=path 将最后的结果保存到本地文件中

–ff (fail first)： 运行所有的测试用例，上次运行失败的用例优先执行

pytest.main(['-s','--ff','test\_firstFile.py'])

3、pytest能兼容unittest，是unittest的扩展，一般都是执行以test\_开头的函数和类。

类中不能带init方法，使用assert进行断言。

4、用例定时执行。

查看：crontab -e，执行：minute hour day month week command

5、测试依赖包管理：

pip3 freeze > requirement.txt

pip3 install -r requirement.txt