

# 压测方案概要设计

## 1.背景

通过压测可以了解服务接口性能，包括并发用户数，响应时间，错误率，TPS(QPS)等，从而发现瓶颈并开展后续的优化工作。

压测需要一些前期准备工作，其中包括搭建压测环境，编写压测脚本，执行压测计划，查看并分析压测报告等。目前各业务都有压测的需求，各自去搭建压测环境，既浪费时间又浪费资源。因此我们提供压测平台以及相关的压测方案，用于各业务平时的接口压测需求，以提高压测效率。

## 2.需求分析

### 2.1功能性需求：

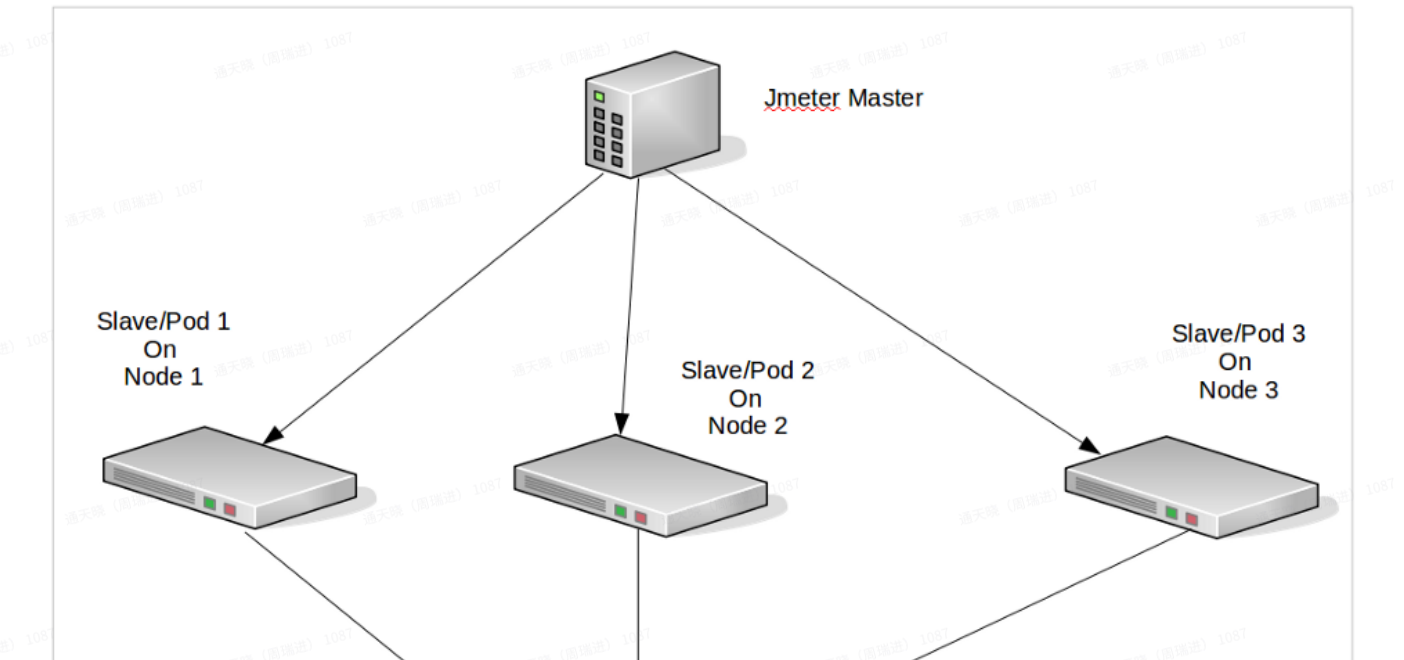
为了完成服务接口压测任务，我们需要创建测试项目，编写压测脚本，上传并执行脚本，在执行脚本过程中查看脚本执行情况，脚本执行完成后，下载压测报告。

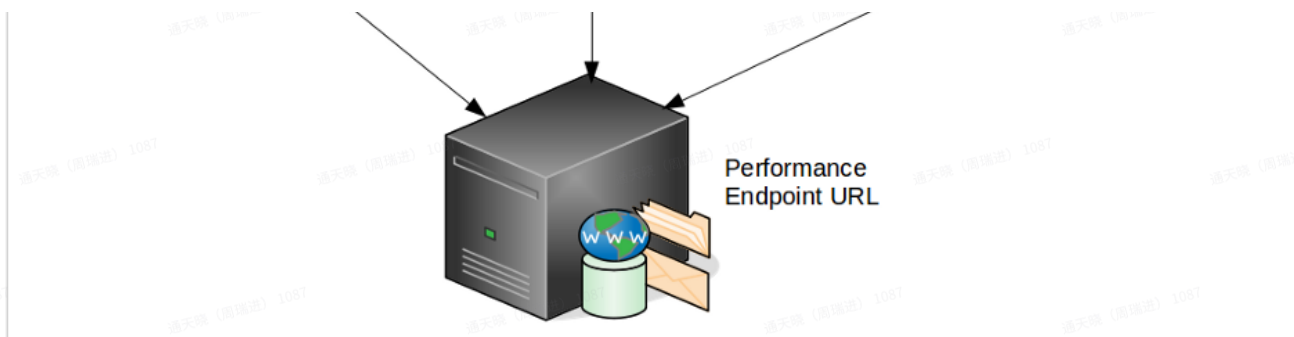
### 2.2非功能性需求：

当被测服务接口需要在大流量的并发请求场景下压测，单台压测机由于能力有限，容易造成卡顿、无响应等情况，需要搭建灵活伸缩的压测机集群，以满足接口大流量的并发请求场景。

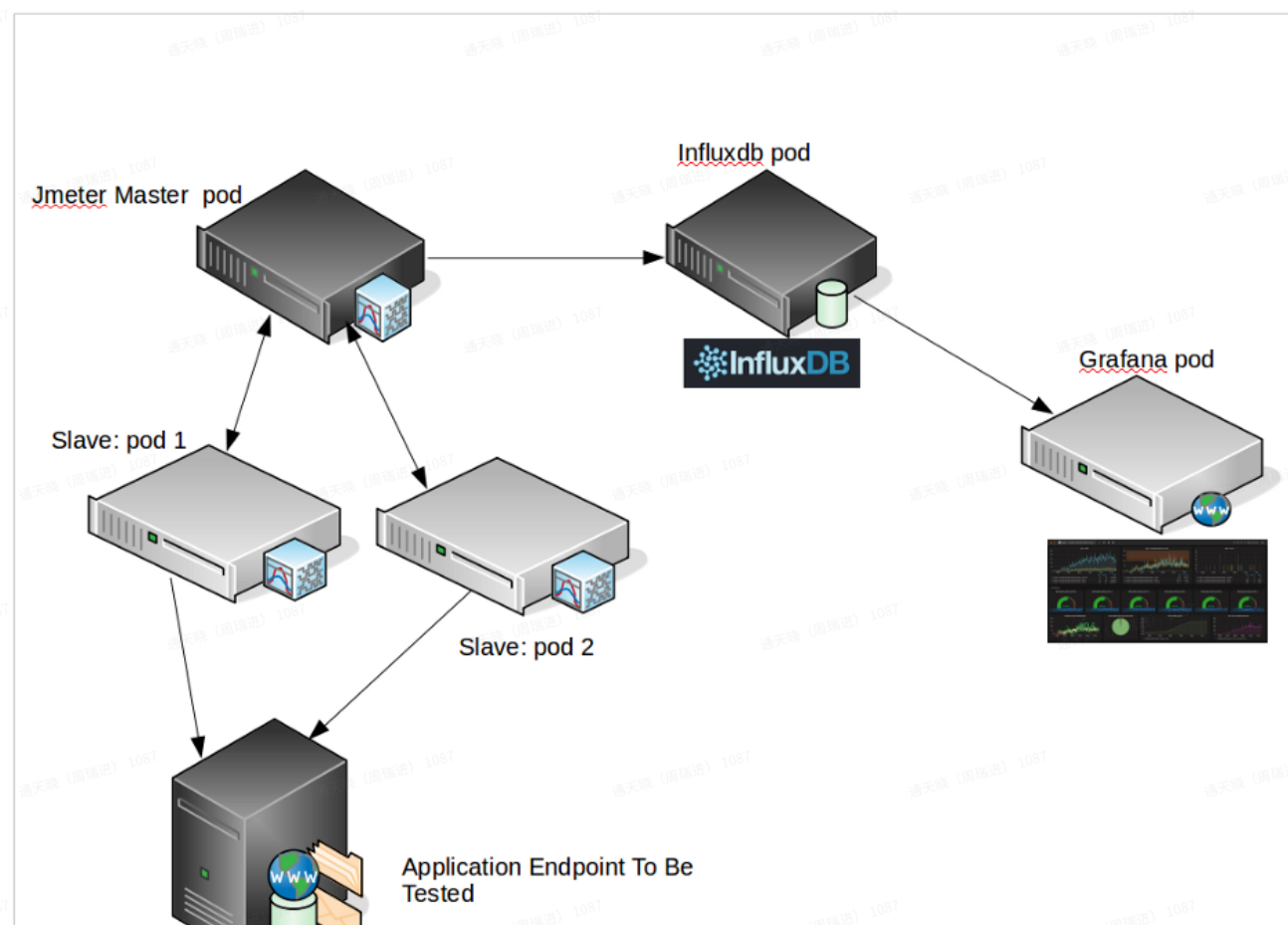
## 3.系统设计

Jmeter支持分布式压测，即将大量的模拟并发分配给多台压力机，Jmeter分布式压测系统架构图：





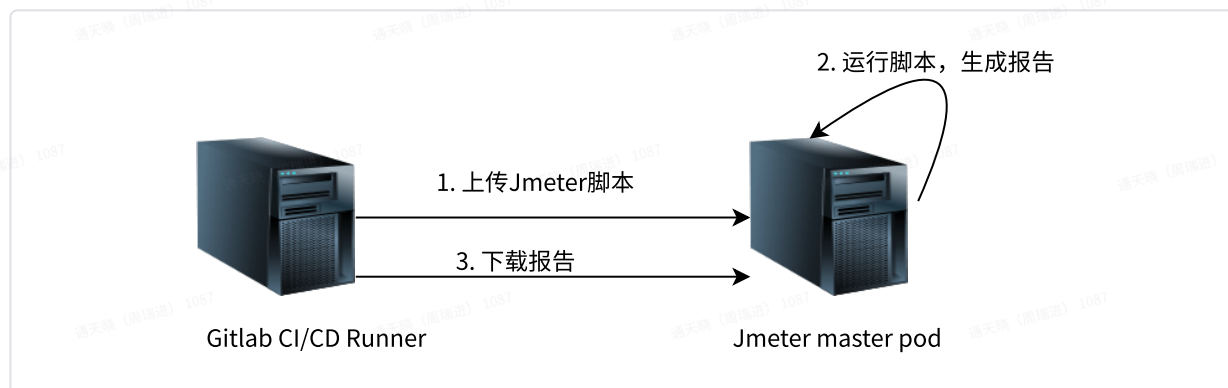
Jmeter-kubernetes是开源的基于k8s的jmeter分布式压测集群搭建方案，git地址：<https://github.com/kubernauts/jmeter-kubernetes>。Jmeter-kubernetes的系统架构图：



Jmeter-kubernetes的压测主控机jmeter master及压测执行机jmeter slave都部署在k8s上，其中压测执行机jmeter slave可以灵活伸缩。Jmeter-kubernetes使用Influxdb存储压测数据，并且使用Grafana以统计图表的形式实时观测压测过程。

本方案基于Jmeter-kubernetes的分布式压测集群，从而实现压测机的灵活伸缩，以满足接口大流量的并发请求场景。

但是Jmeter-kubernetes没有压测web端控制台，无法直接上传压测脚本和下载压测报告，为了解决这个问题，本方案借助公司Gitlab的CI/CD流程，通过CI/CD Runner服务器与Jmeter master服务器交互，实现压测脚本上传及压测报告下载。



## 4.方案对比

	直接使用华为云性能测试服务	jmeter-kubernetes
灵活伸缩	支持	支持
web端控制台	支持	不支持
上传压测脚本	支持	通过cicd流程，类似于发布代码
查看实时报告	支持	支持
导出报告	支持	支持
支持多种协议接口	支持	支持，jmeter需要安装rpc测试插件
文档	完善	完善
开发周期	少	中等
优点	<ul style="list-style-type: none"><li>· 文档清晰</li><li>· 功能完备，无需开发。</li><li>· 支持协议种类多，可扩展性强。</li><li>· 华为云自带服务，服务稳定，在很多企业级应用中都有使用。</li></ul>	<ul style="list-style-type: none"><li>· 服务稳定</li><li>· 集群部署快捷</li><li>· 使用k8s cce，可根据压力灵活伸缩</li></ul>
缺点	<ul style="list-style-type: none"><li>· 需要按压测流量计费</li></ul>	<ul style="list-style-type: none"><li>· 没有用于管理压测任务的web端控制台</li><li>· 需要研究实时报告的Grafana配置</li></ul>

本压测方案适用于各业务开发对自己开发的模块接口进行压测，并不是大规模的系统集成压测方案，因此采用方案二。

## 5.压测任务执行流程（重要）

### 5.0 压测准备工作

#### 5.0.1在gitlab上创建压测工程（注：使用公共的压测项目该步骤可忽略，见6.9支持自定义jar包）

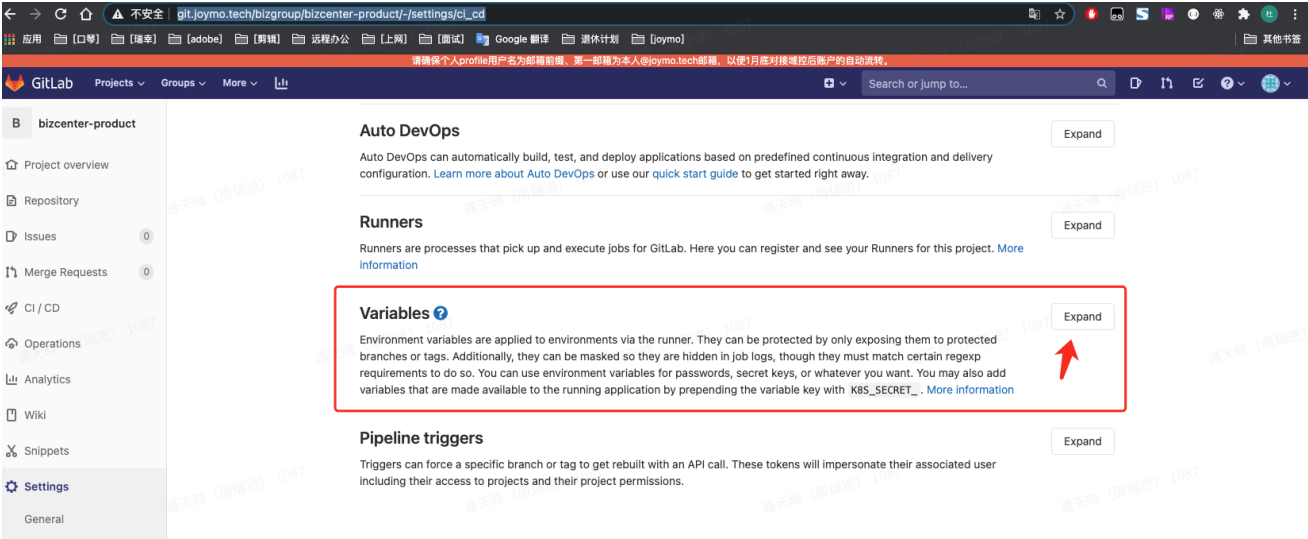
为了借助Gitlab的CI/CD流程，需要为压测脚本创建gitlab项目，参考例子工程：<http://git.joymo.tech/bizgroup/joymo-rule-jmeter.git> 的master分支

##### 5.0.1.1 CICD基本配置

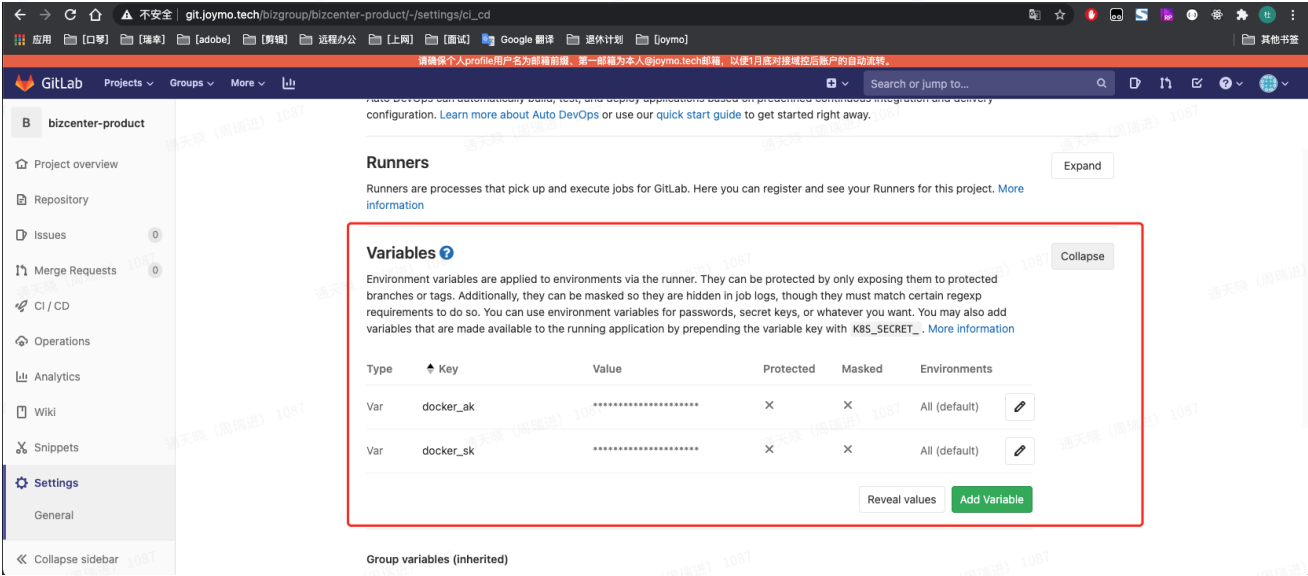
目前是通过gitlab部署，前期配置文件需要手动指定；

docker_ak	ESCRBKC3NAJINQZEQMNQ
docker_sk	ef4ba3ddb01a626c1bfc80e37653e1e2c74a5d045eaa01f78d2dcabb8470e6f4

1、配置docker推送密钥

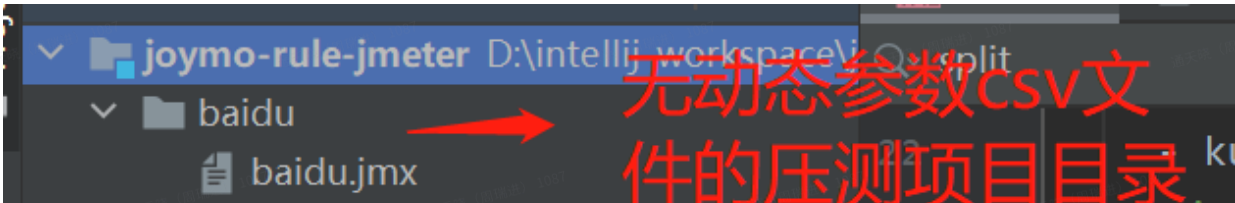


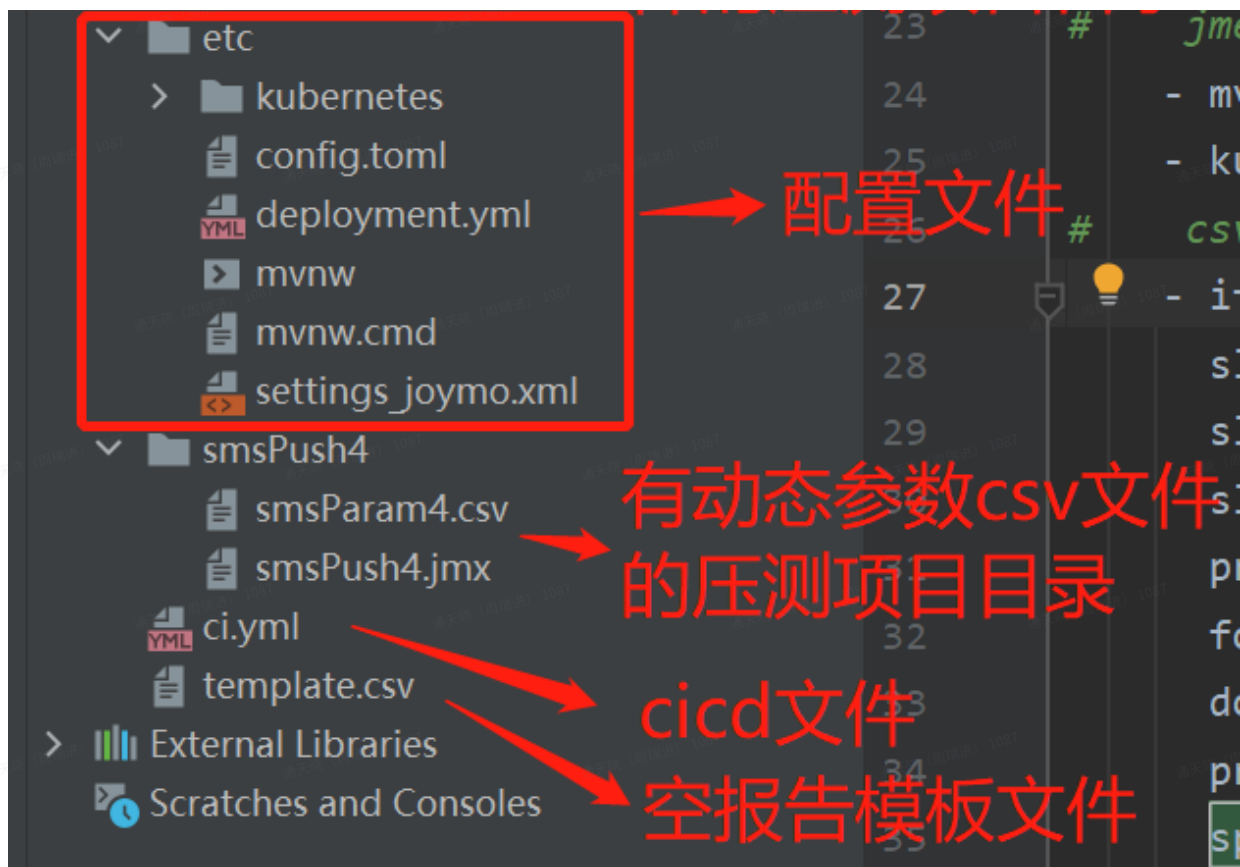
2、配置后



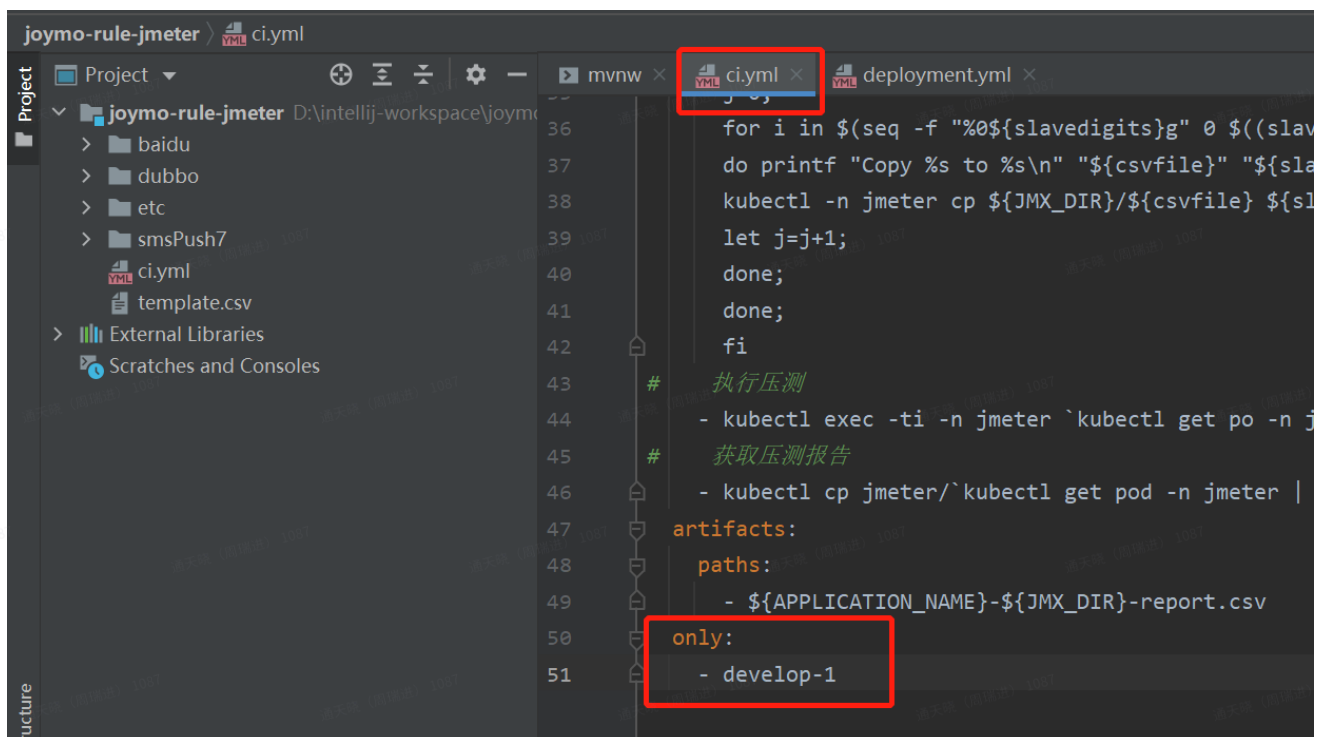
5.0.1.2 项目配置

以下举例项目<http://git.joymo.tech/bizgroup/joymo-rule-jmeter.git>，可复制以下etc目录、ci.xml、template.csv文件





具体说明变更项，大家拷贝的时候注意代码需要在only对应的分支上修改



项目	说明
etc需要修改的文件deployment.yml	
ci.yml 需要更新配置项	 <p>JMX_DIR: 是压测项目目录，该文件夹下只有一个jmx文件，并且文件名要求文件夹名称一致。例如：baidu文件夹下应该有一个jmx文件为：baidu.jmx</p> <p>该文件夹下还可以放置脚本参数csv文件，个数目前没有限制</p>

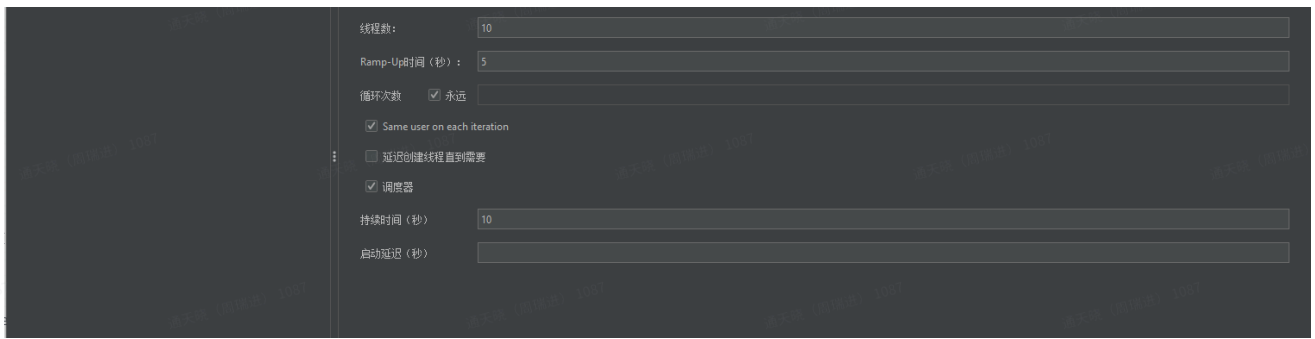
## 5.0.2本地编写并调试Jmeter脚本

### 5.0.2.1本地安装jmeter

先在本地安装jmeter [https://jmeter.apache.org/download\\_jmeter.cgi](https://jmeter.apache.org/download_jmeter.cgi),

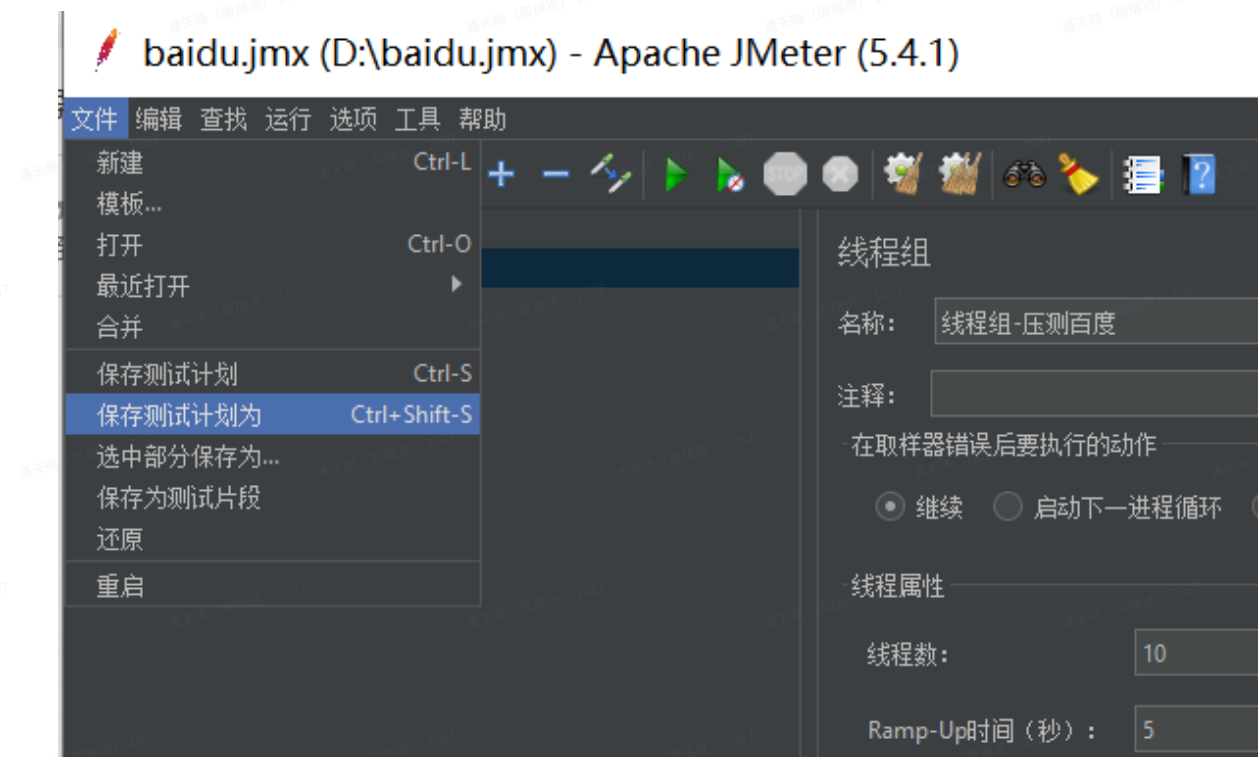
### 5.0.2.2编写并保存jmeter压测脚本



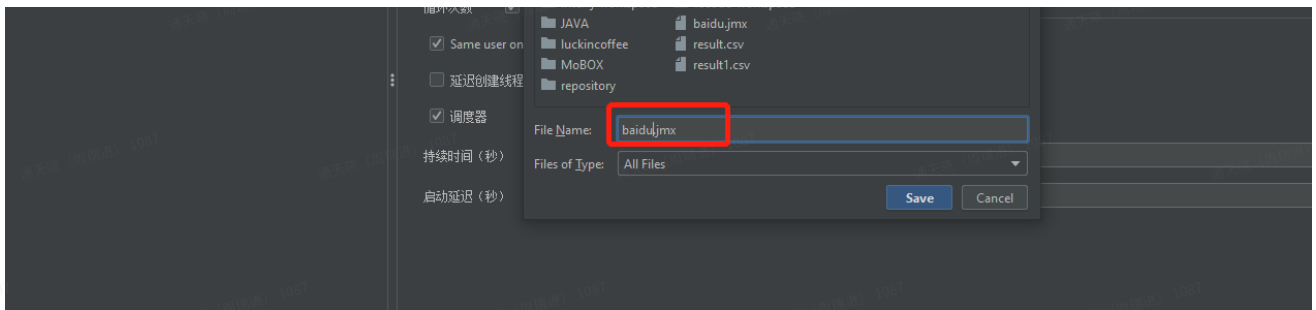


注意：线程组线程数的配置应考虑测试1压测平台有2台执行机，因此线程数应考虑除以2，具体说明见6.4关于jmeter脚本的压测参数设置

在本地调试通过后，将jmeter压测脚本保存为.jmx文件：

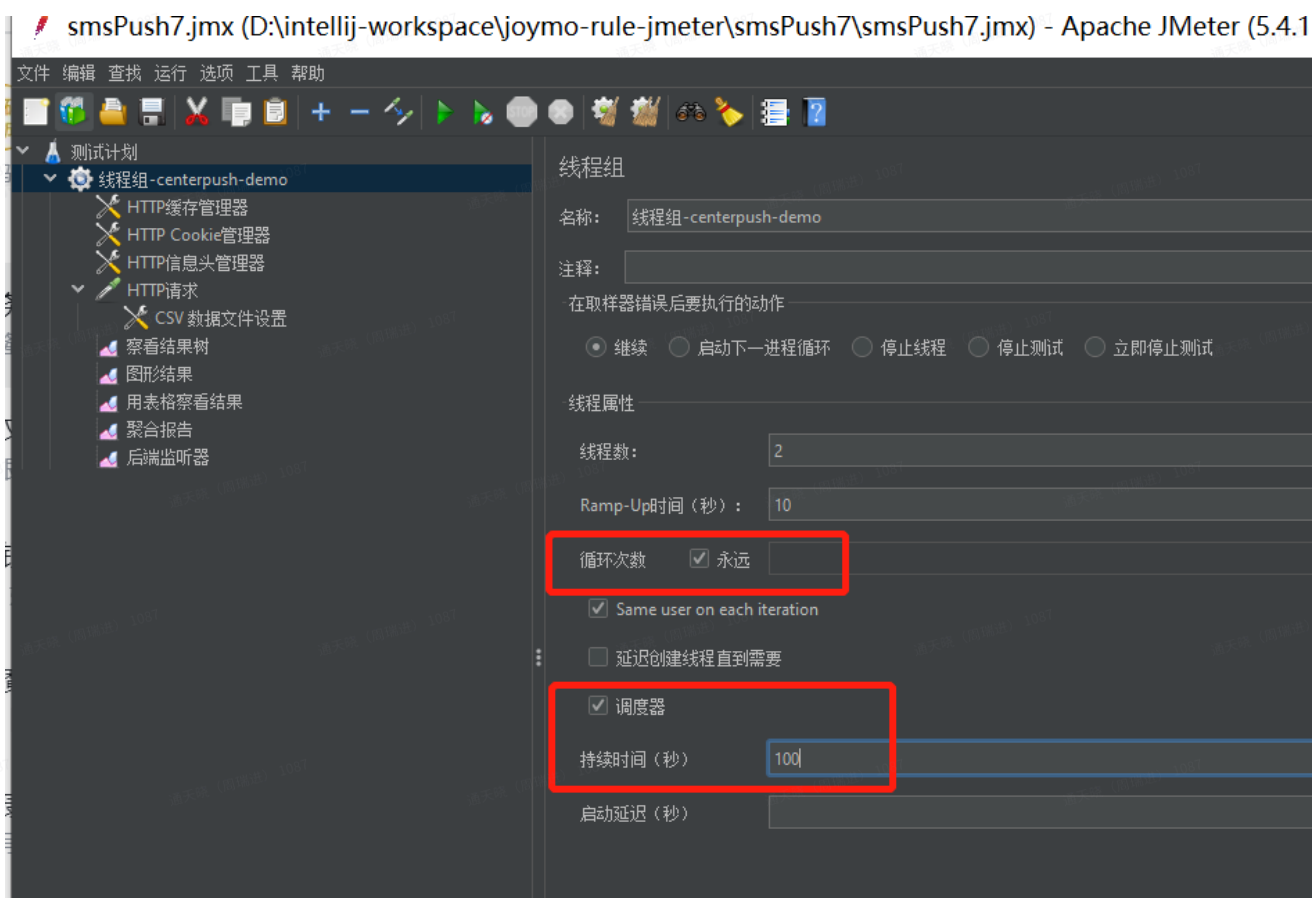






### 5.0.2.3 jmeter脚本配置注意

大家在本地调试完jmeter脚本，在测试1执行压测前麻烦检查一下线程组的配置，如果勾选了循环次数为永远，则调试器一定要指定一下持续时间。勾选了永远，如果没有指定持续时间的话，执行机将会被一直占用，其他压测任务就都无法执行了。



## 5.1 登记压测计划信息

由于我们的压测执行机一次只能执行一个压测任务，当其他同事正在执行压测，然后你也同时进行压测时，会报出这个错误：

```
-n jmeter cp ${JMX_DIR}/${csvfile} ${slave_pods[j]}:/; let j=j+1; done; done; fi
29 ls: cannot access 'baidu/*.csv': No such file or directory
30 $ master_pod=$(kubectl get pod -n jmeter | grep jmeter-master | awk '{print $1}'); i
f [ `kubectl exec -ti -n jmeter ${master_pod} -- /bin/bash /load_test ${APPLICATION_NAM
E}-${JMX_DIR}.jmx ${APPLICATION_NAME}-${JMX_DIR}-report.csv | grep 'Engine is busy - ple
ase try later' | wc -w` -gt 0 ] ; then printf "Engine is busy - please try later\n"; exi
t 1; else kubectl cp jmeter/`kubectl get pod -n jmeter | grep jmeter-master | awk '{prin
t $1}'`:${APPLICATION_NAME}-${JMX_DIR}-report.csv ${APPLICATION_NAME}-${JMX_DIR}-report.
csv; fi
31 Unable to use a TTY - input is not a terminal or the right kind of file
32 SLF4J: Class path contains multiple SLF4J bindings.
33 SLF4J: Found binding in [jar:file:/jmeter/apache-jmeter-5.0/lib/log4j-slf4j-impl-2.11.
0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
34 SLF4J: Found binding in [jar:file:/jmeter/apache-jmeter-5.0/lib/ext/jmeter-plugins-dub
bo-1.3.8-jar-with-dependencies.jar!/org/slf4j/impl/StaticLoggerBinder.class]
35 SLF4J: Found binding in [jar:file:/jmeter/apache-jmeter-5.0/lib/ext/pepper-box-1.0.ja
r!/org/slf4j/impl/StaticLoggerBinder.class]
36 SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
37 SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
38 Engine is busy - please try later
42 ERROR: Job failed: command terminated with exit code 1
```

为了解决这个问题，有需要压测的同学麻烦先把压测计划填写在下面的表格中，写一下开始时间和计划结束时间，这样大家可以统筹安排时间。

## 业务中台压测报告

< 业务中台压测报告 ★  
11 压测 | 最近修改: 3小时前

压测计划				
压测系统				
	A	B	C	D
1	压测系统	日期	计划开始时间	计划结束时间
2	卡券系统	2021/3/11	10:30	18:30
3				
4	订单中台 (算价、订单)	2021/3/12	9:30:00	18:30
5				
6				
7				
8				

## 5.2 执行压测前在 cicd 答疑群 或 压测协调沟通群 周知下

由于压测资源有限，并且压测产生的数据需要占用华为云资源，在执行压测资源前，需要在cicd答疑群周知，让大家有心理预期。

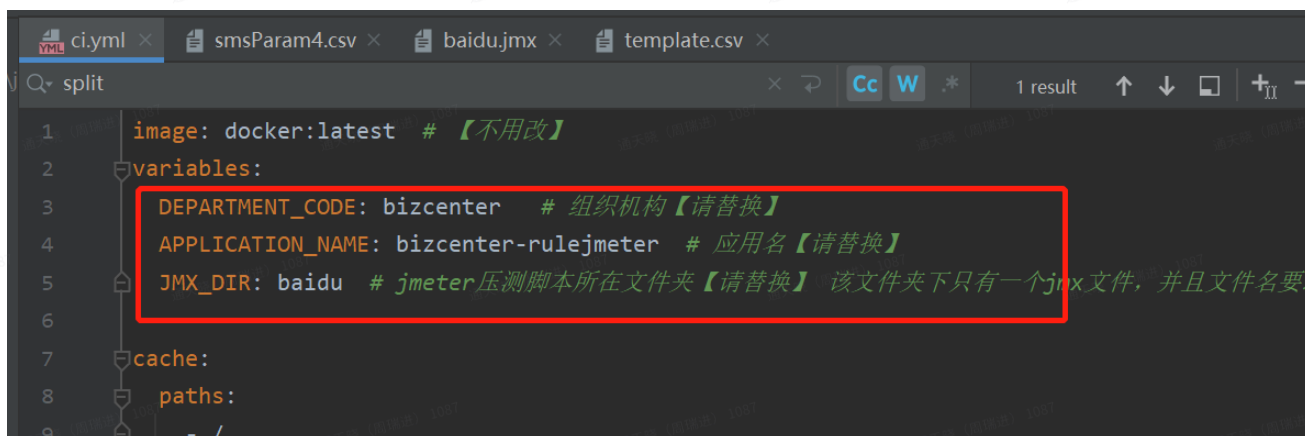
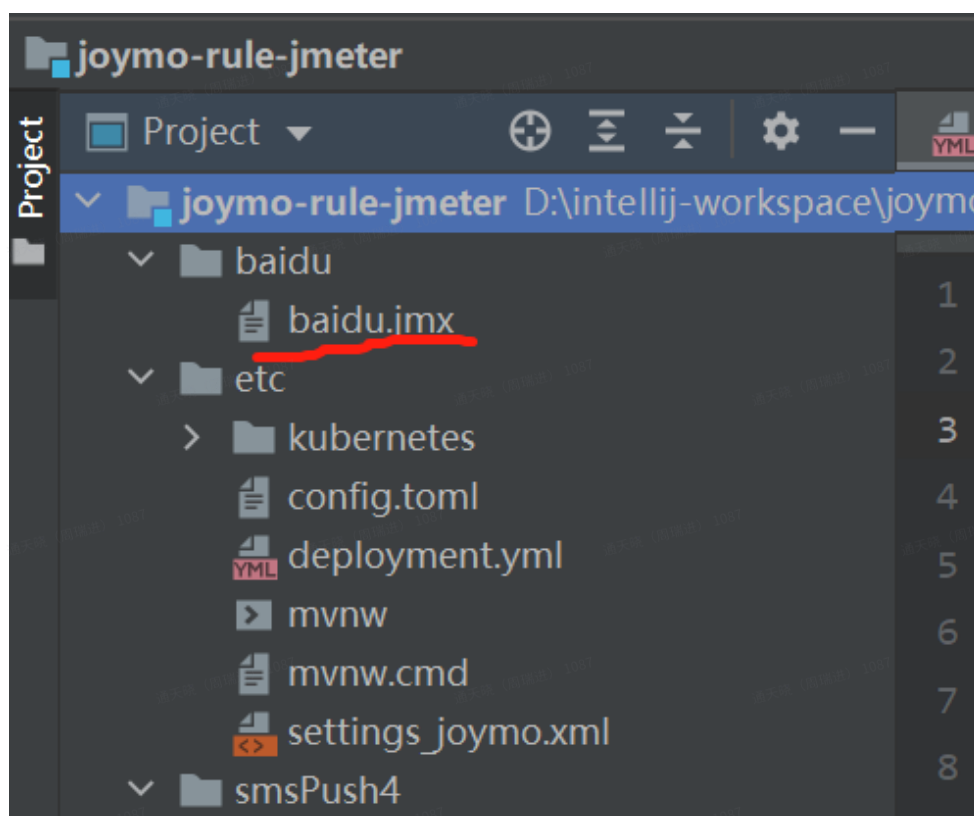
提前预知内容：执行时间点，和预计持续时长，预计需要多少资源，对哪些系统有影响。例如：

1.当前时间执行，2.预计持续2小时，3.占用压测环境和数据库资源，4.暂无影响的上下游系统

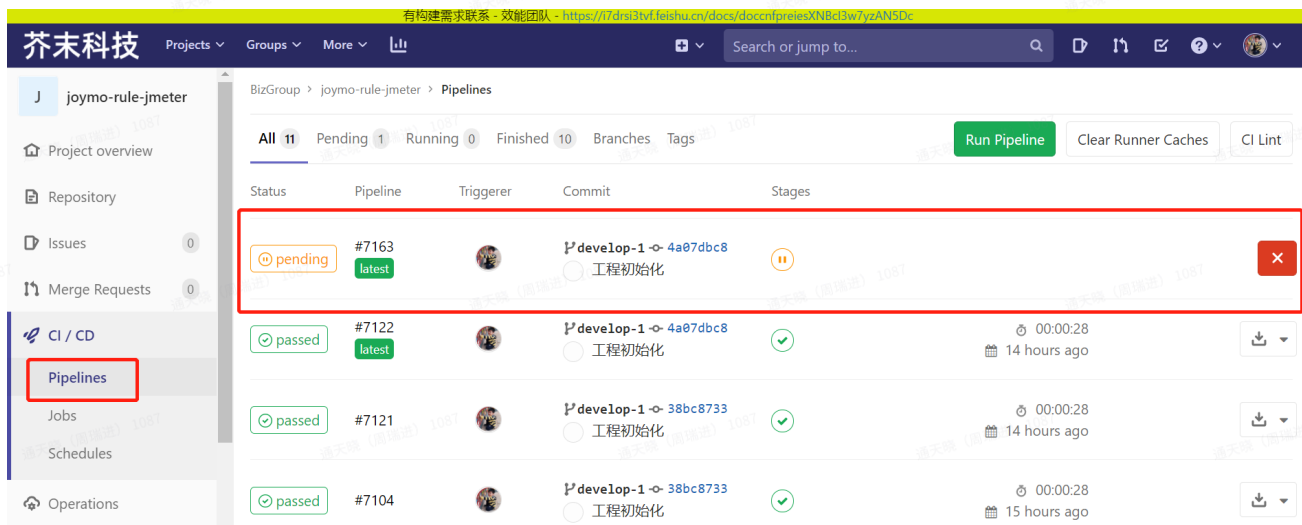
## 5.3 具体执行压测

### 5.3.1发起部署，上传Jmeter脚本，触发压测任务

至此已经本地调试好压测脚本，并保存为了baidu.jmx。接下来将baidu.jmx拷贝到步骤6.1的压测工程中，然后提交代码到gitlab。

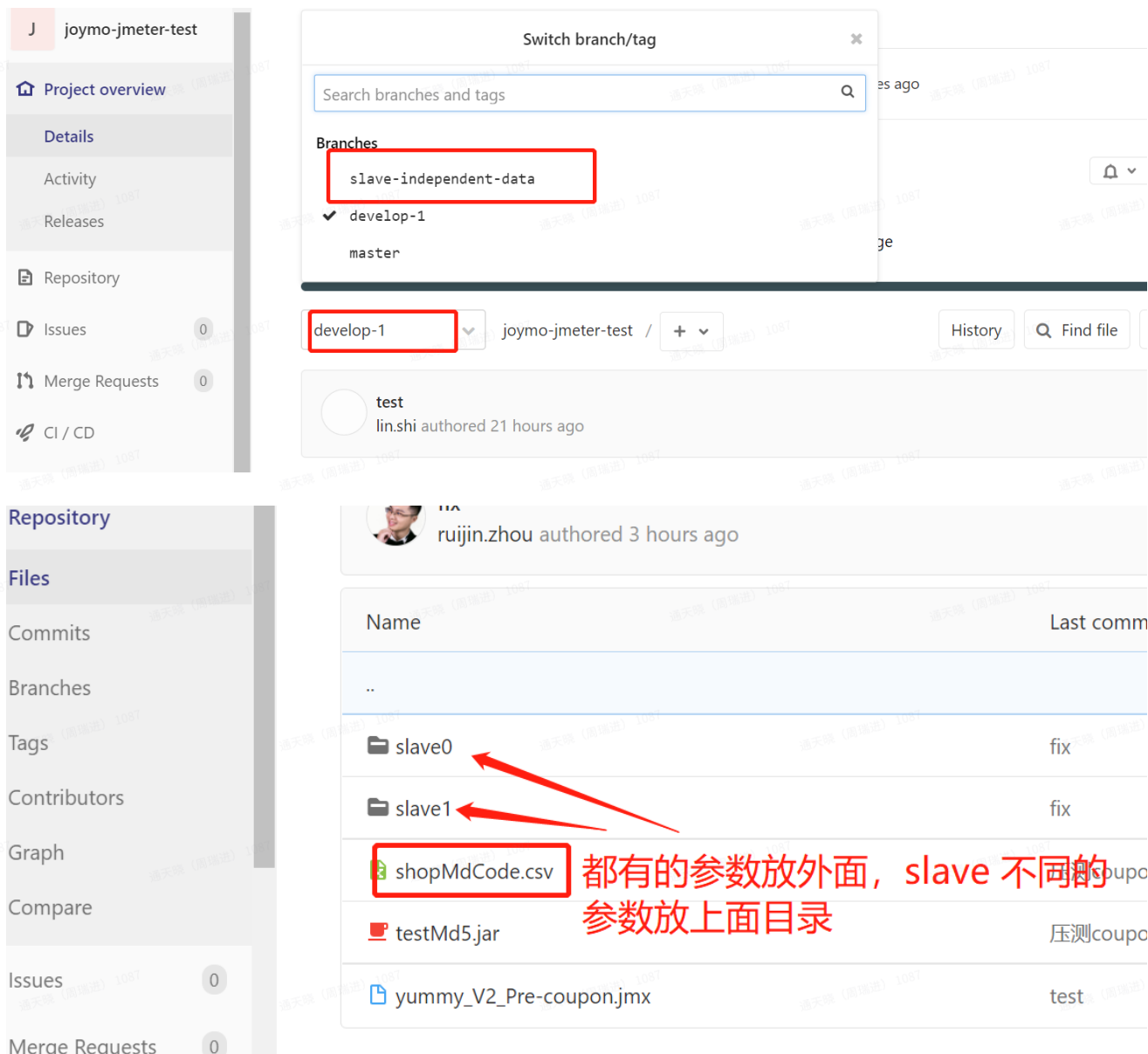


gitlab会在提交后自动将.jmx文件上传到Jmeter master pod，并触发执行压测，以下为压测执行过程中CI/CD的状态：



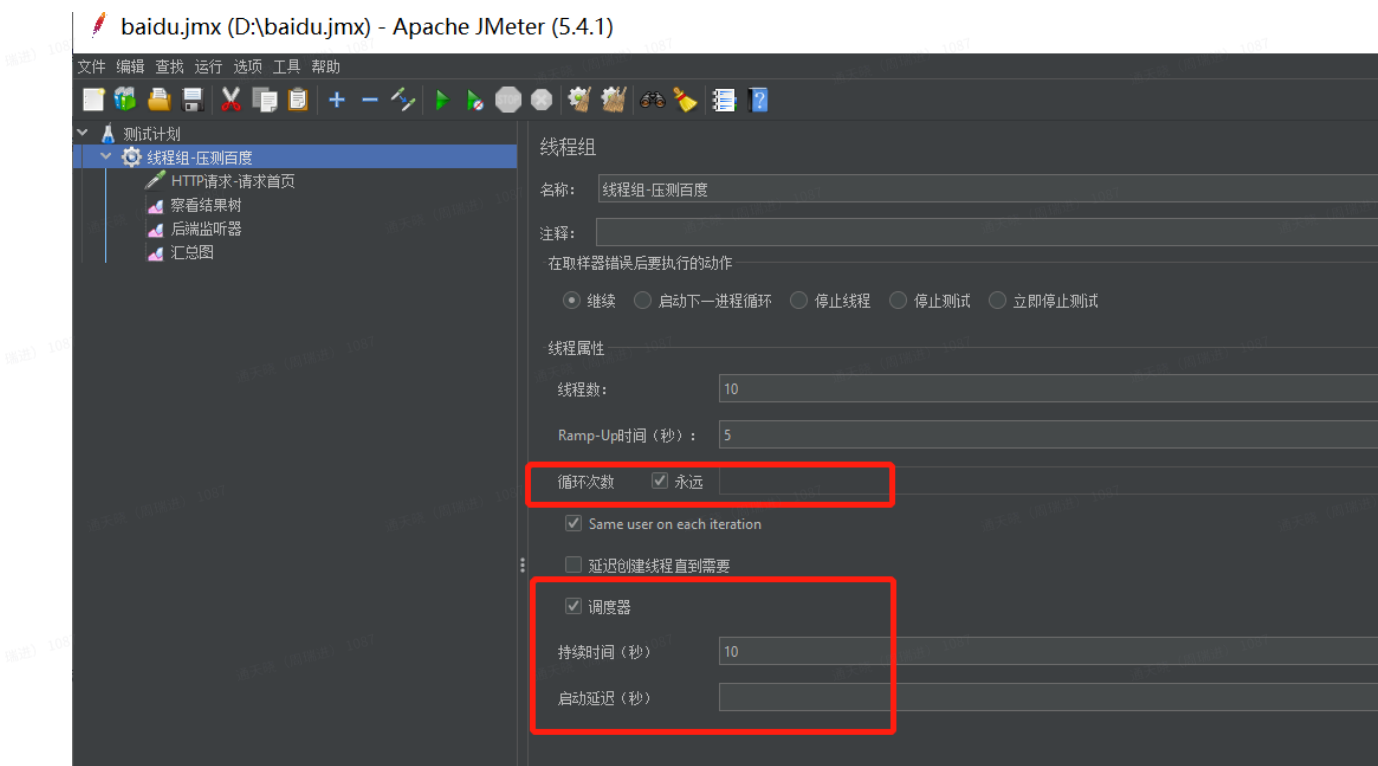
### 5.3.2 分布式压测的数据切割（不同 slave csv 参数不一样）

1. 压测仓库切换到 slave-independent-data 分支
2. CSV 文件放置：相同的放你文件夹首层，不同 slave 各自的放里面的目录。

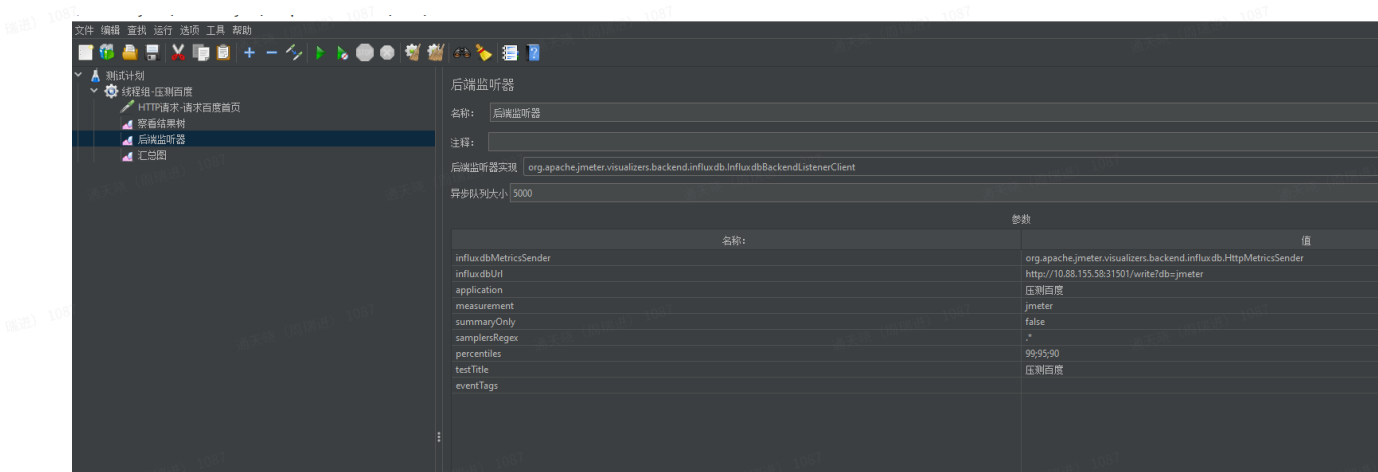


### 5.3.3 查看实时压测数据（可选）

压测执行过程一般比较耗时，具体取决于压测脚本中的压测持续时间配置：



借助于Influxdb的后端监听器，我们可以实时查看压测数据。在压测脚本中添加一个Influxdb的后端监听器：



后端监听器实现：InfluxdbBackendListenerClient（固定）

influxdbUrl: <http://10.88.156.58:31501/write?db=jmeter>（已废弃）

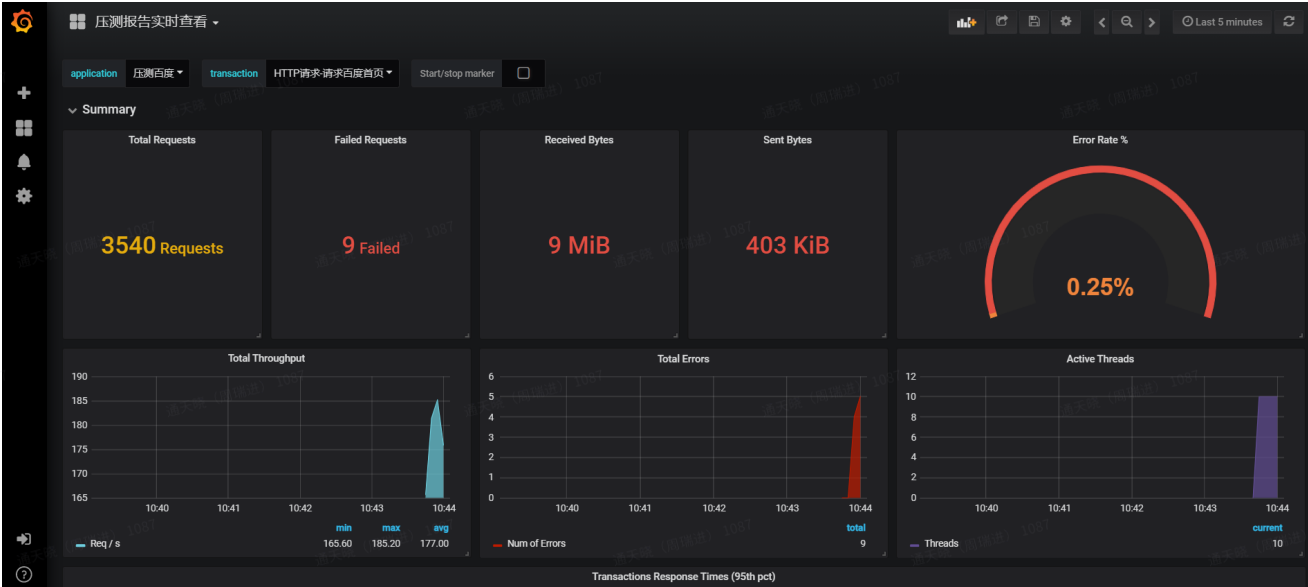
<http://10.88.152.15:31501/write?db=jmeter>（固定）

application：压测应用名称（自己定义）

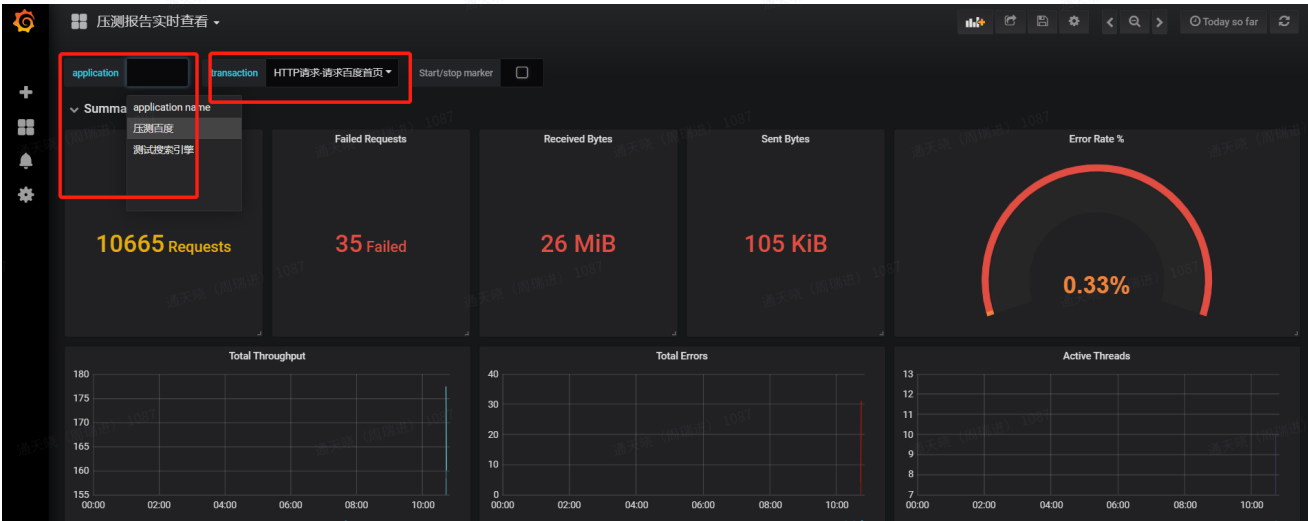
measurement：jmeter（固定）

访问压测实时报告：<http://10.88.150.168:31047/d/xu0zzwyGz/ya-ce-bao-gao-shi-shi-cha-kan?orgId=1>

<http://10.88.152.15:31047/d/ltas/jmeter-metric-template?refresh=30s&orgId=1>

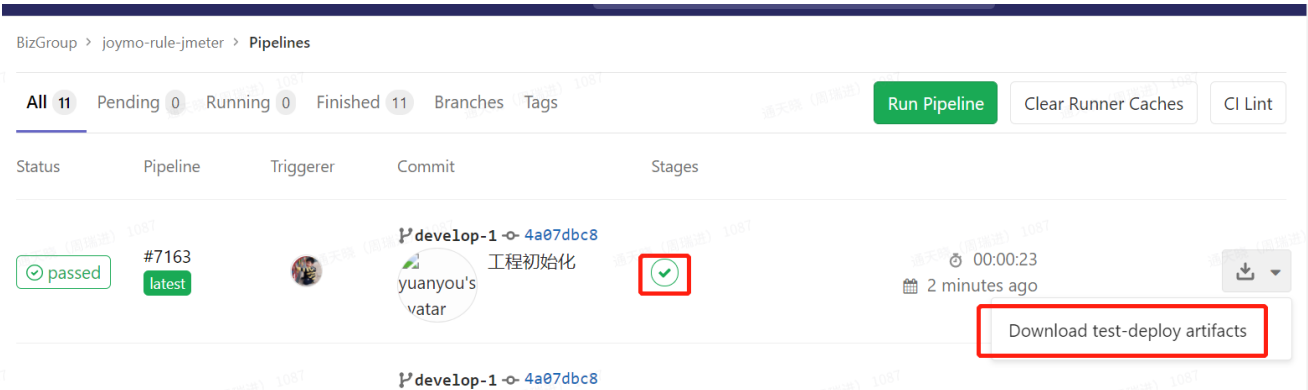


注意不同的压测任务需要切换查询条件：



### 5.3.4 下载压测报告数据，并生成压测报告

当压测执行完成后，点击pipeline右侧的下载按钮，会得到一个artifacts.zip压缩包，解压后即得到xxx.csv压测数据文件。需要注意该下载文件在gitlab上目前最多保存2天，压测完成后请及时下载。







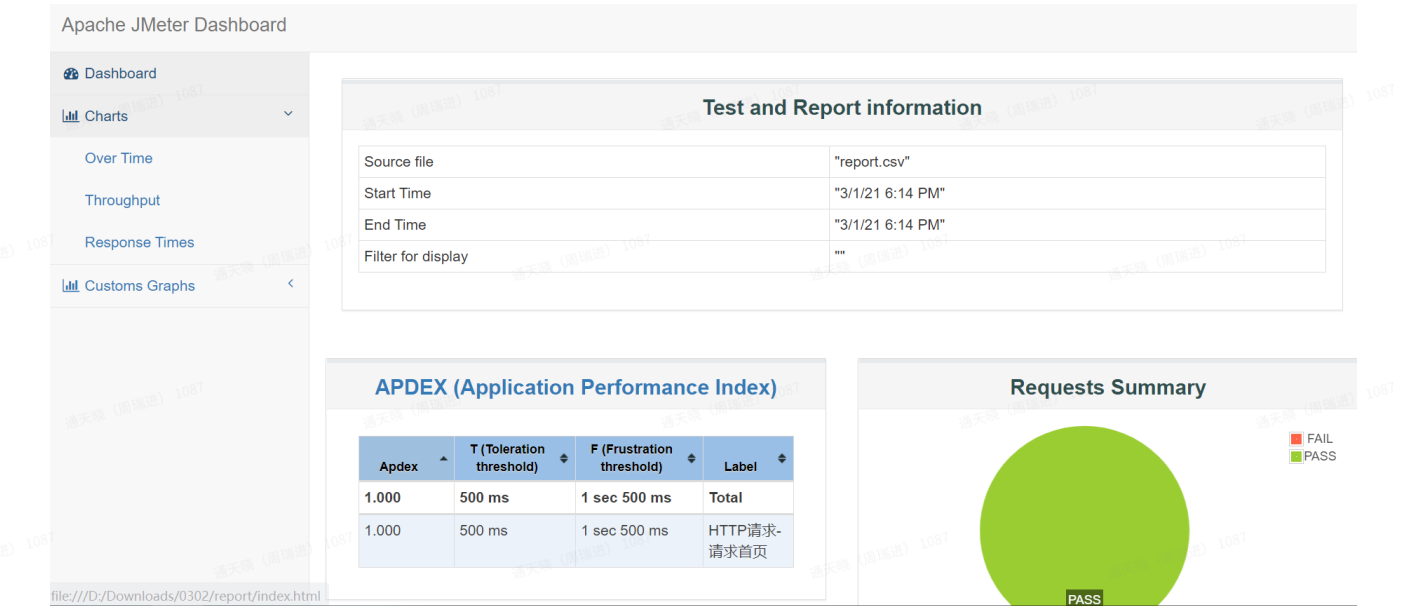
基于xxx.csv生成压测报告，在本地执行命令：

Plain Text

```
1 jmeter -g D:\xxx.csv -o D:\report
```

执行成功后可以在D:\report目录下看到index.html，打开即可查看压测报告。**需要注意参数-o指定的文件夹必须是空的，或者是不存在的**

名称	修改日期
 content	2021/3/2 10:55
 sbadmin2-1.0.7	2021/3/2 10:55
 index.html	2021/3/2 10:55
 statistics.json	2021/3/2 10:55



## 5.4 清除压测APM数据

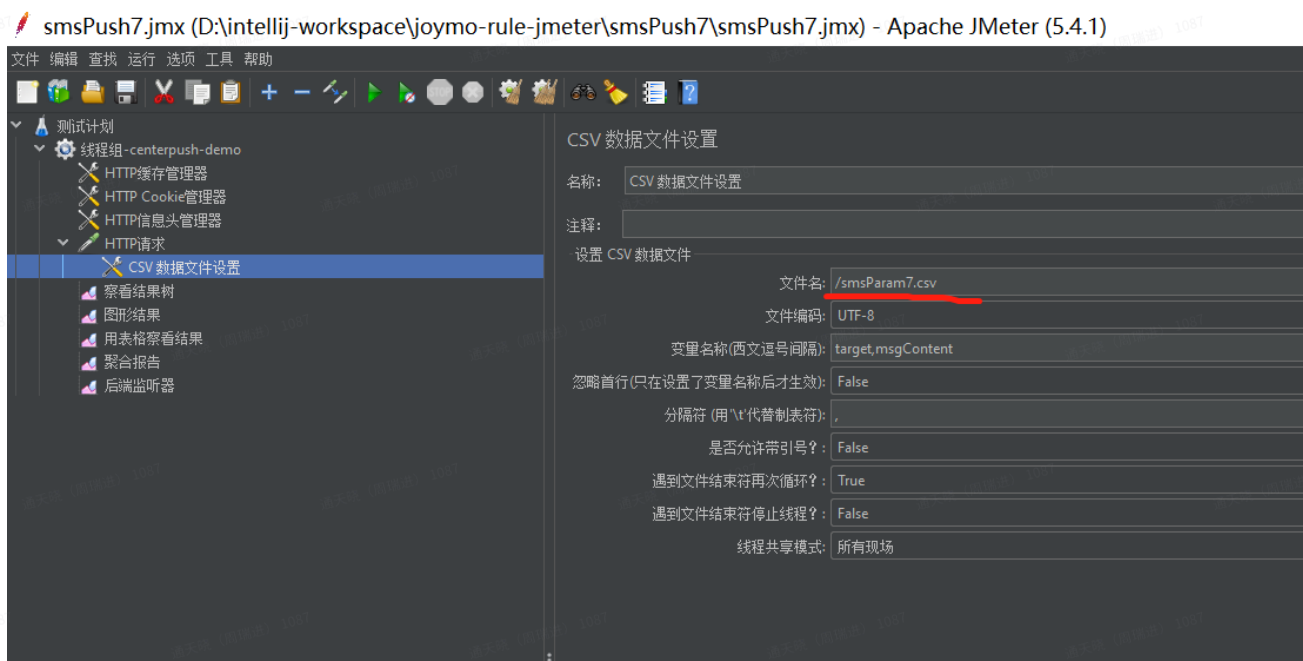
我们做压测时要跟APM架构的杨越说一下，因为压测的APM数据吃掉了大量的ES，目前的ES是买华为云非自建，也比较贵。也就是说，压测后需要考虑还原或清理的事项。

## 6.答疑

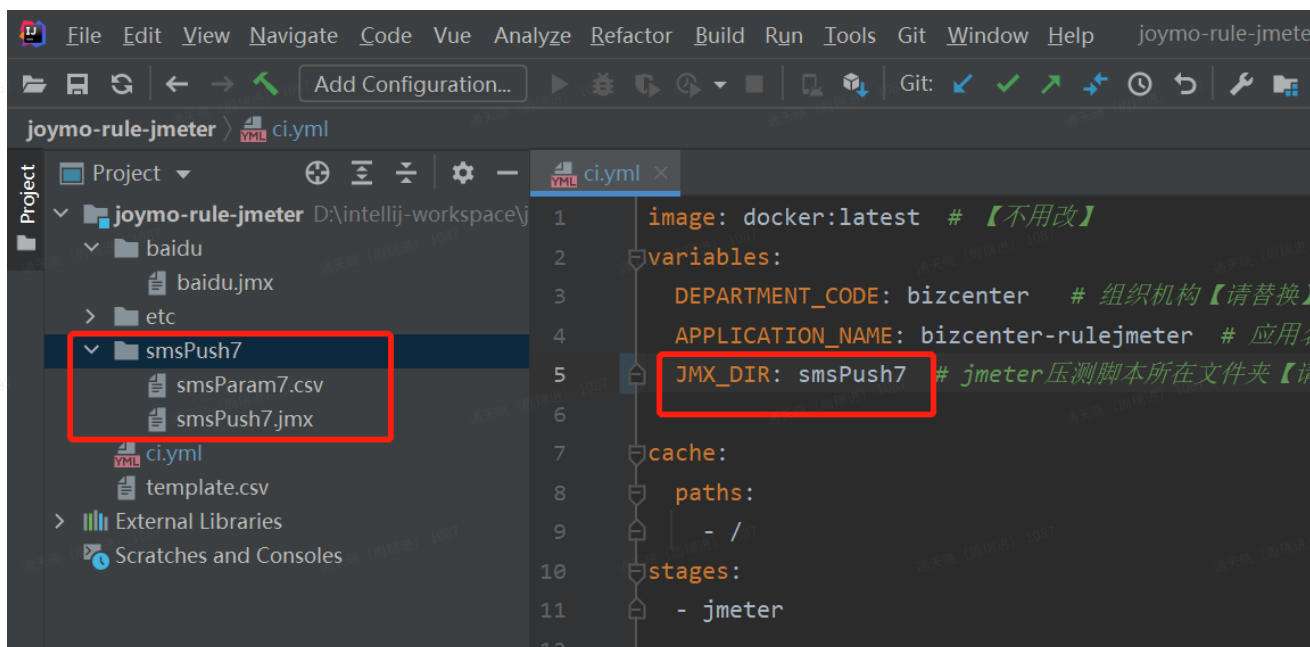


## 6.1 动态参数csv文件使用

参数csv文件在本地jmeter调试成功后，需要调整参数csv数据文件路径为"/csv文件名"，比如说参数csv文件为smsParam6.csv，则这里的文件名就需要改为/smsParam6.csv。这是因为在jmeter集群环境执行时，参数csv文件是从/目录查找的。



修改好后，重新保存jmx脚本，然后将jmx，csv文件拷贝到压测工程中，并修改一下ci.yml中的压测项目目录名配置：



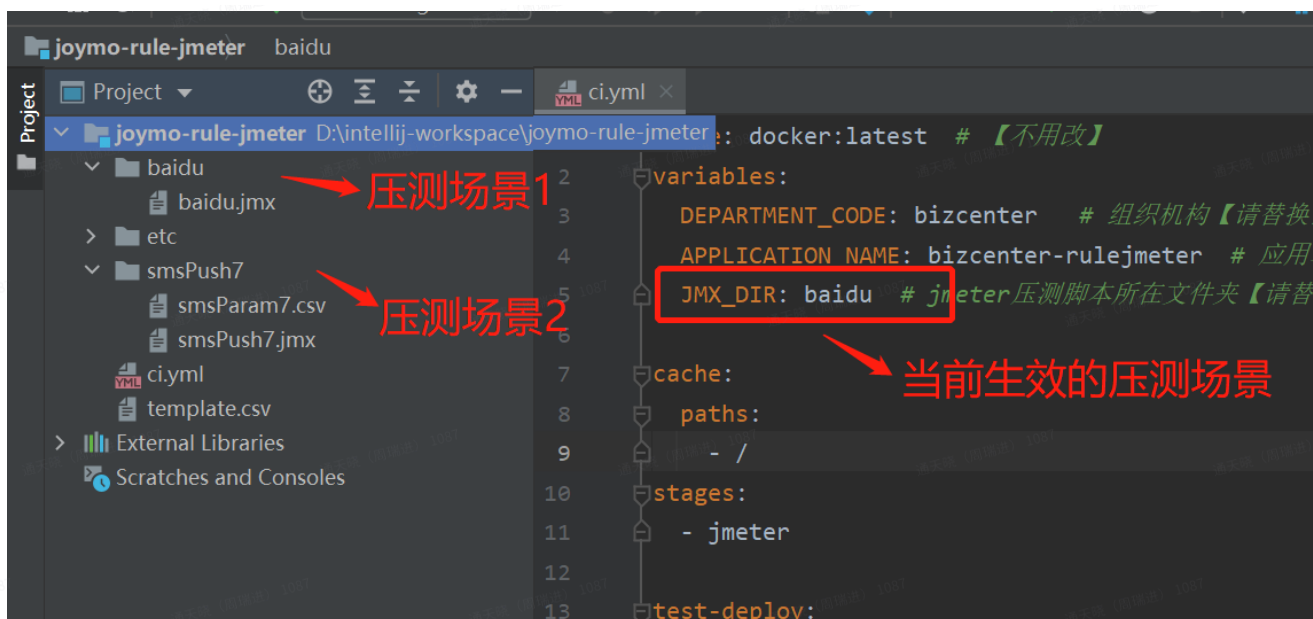
最后push代码，将会自动触发压测。

需要注意的是为了防止参数csv文件在jmeter集群环境重名，参数csv文件的文件名应该唯一。

## 6.2 在一个压测工程中支持多个压测场景

在一个压测工程中存在多个压测场景，可以在压测工程中创建不同的文件夹来区分：





然后在ci.yml文件中配置好当前生效的压测场景文件夹

## 6.3 dubbo接口的压测

dubbo接口的压测依赖开源的jmeter插件，项目地址<https://gitee.com/ningyu/jmeter-plugins-dubbo>，使用步骤：

### 6.3.1.本地安装jmeter-plugins-dubbo

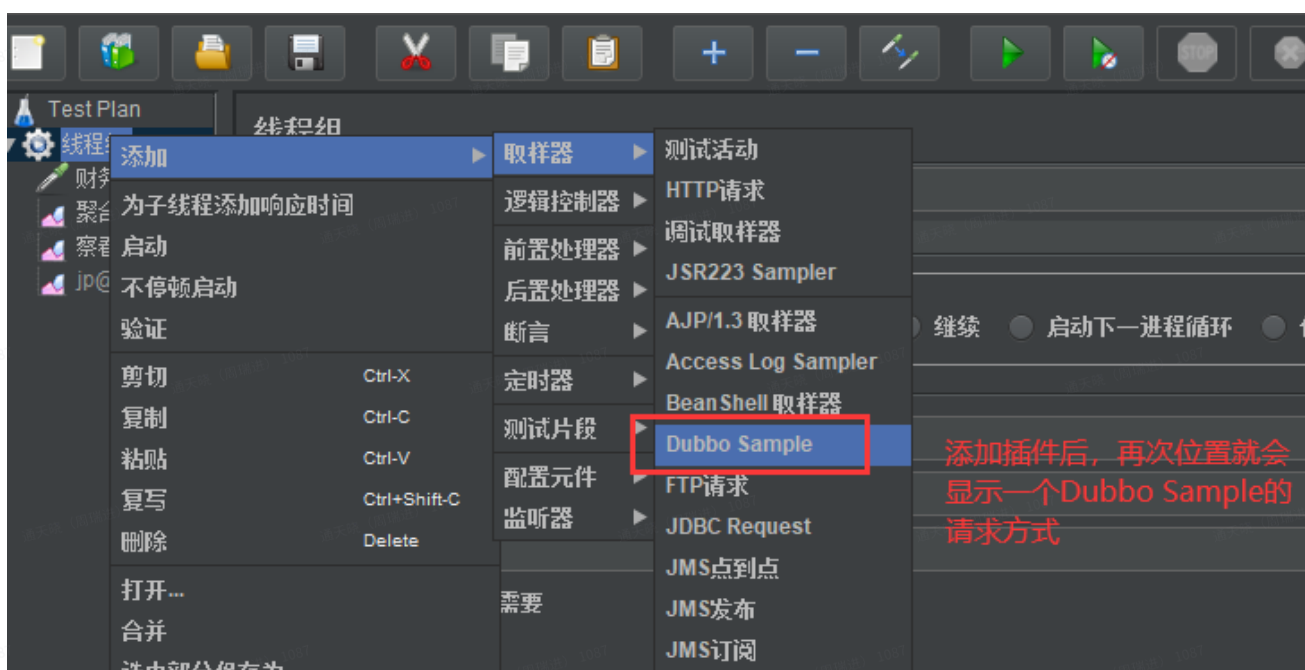
将jmeter-plugins-dubbo-1.3.8-jar-with-dependencies.jar放置在本地%JMETER\_HOME%/lib/ext下，然后本地重新启动jmeter



jmeter-plugins-dubbo-1.3.8-jar-with-dependencies.jar

14.66MB

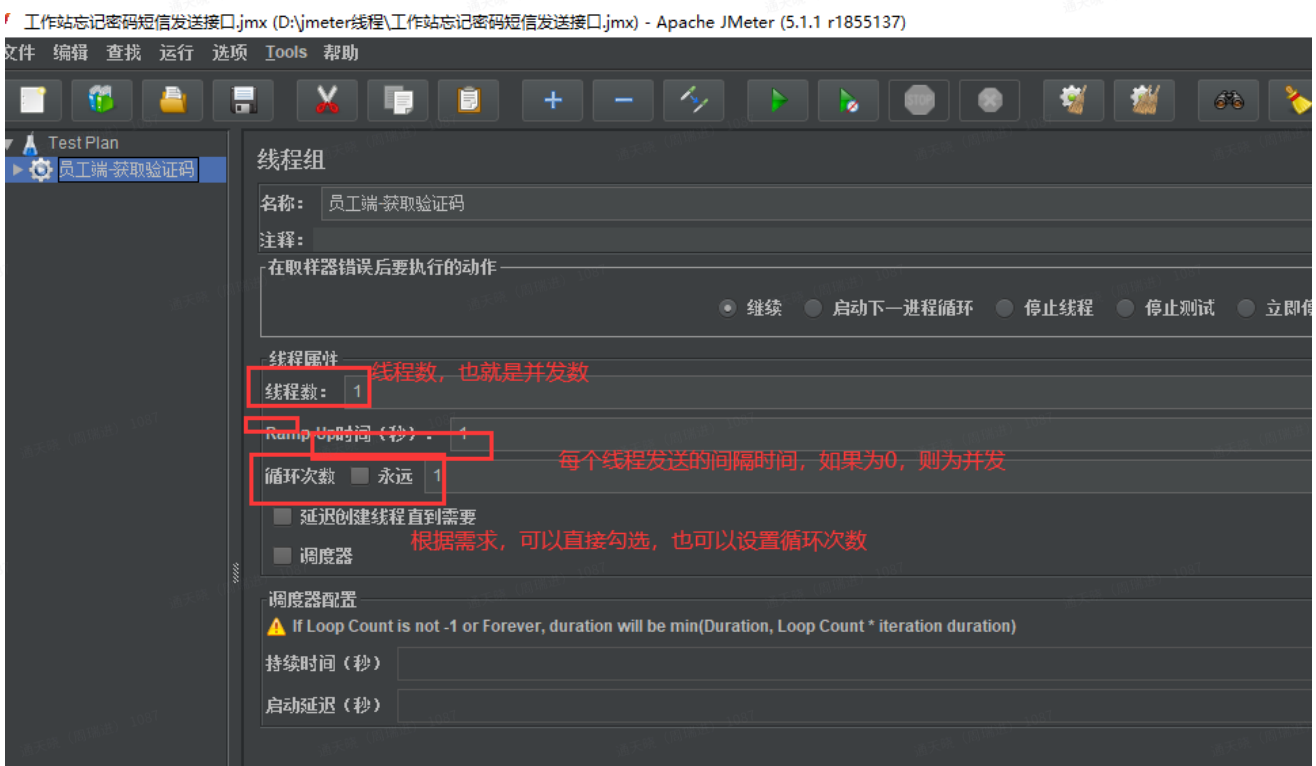
就可以看到取样器下有Dubbo Sample:



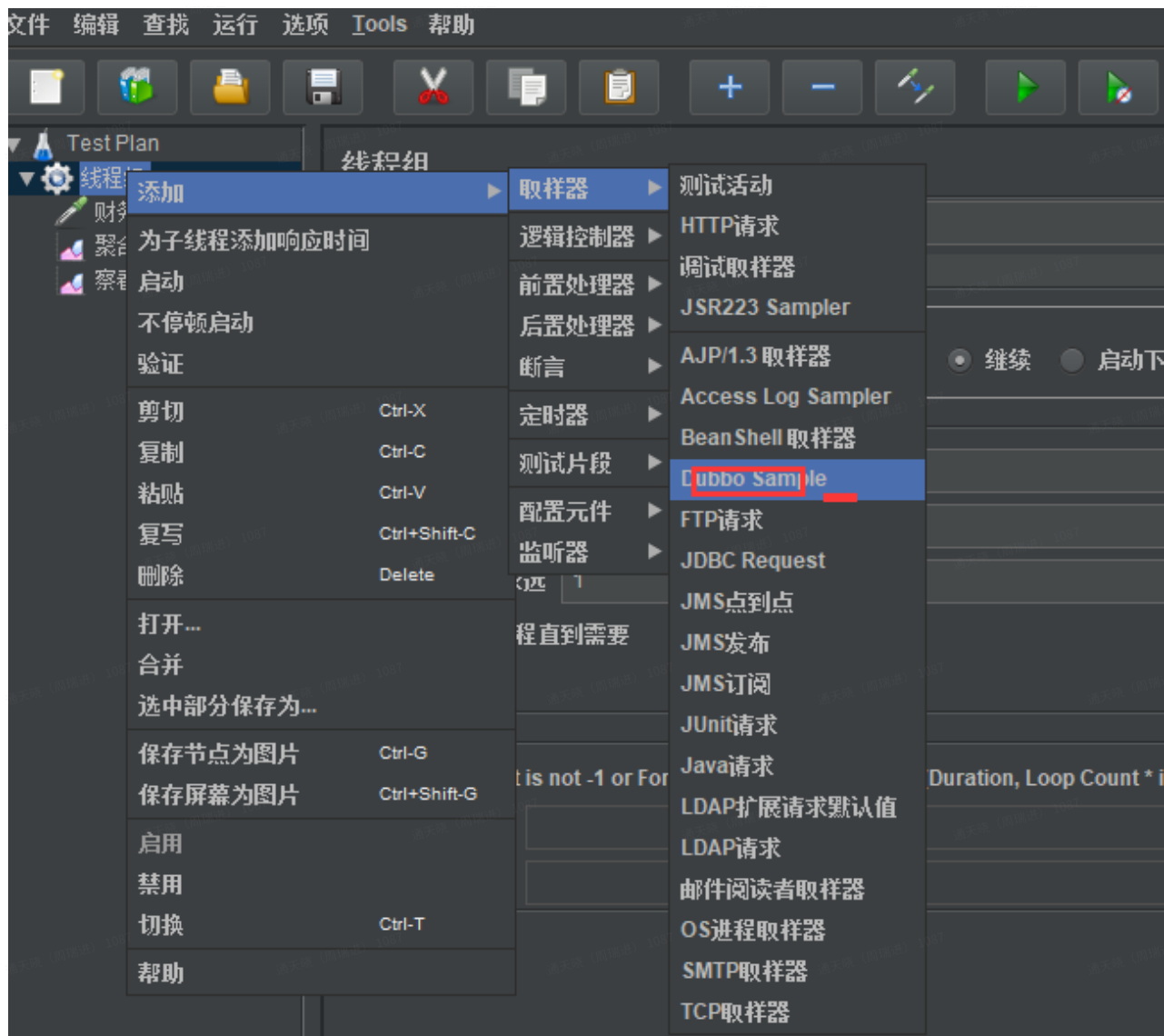


## 6.3.2.本地编写dubbo接口压测脚本

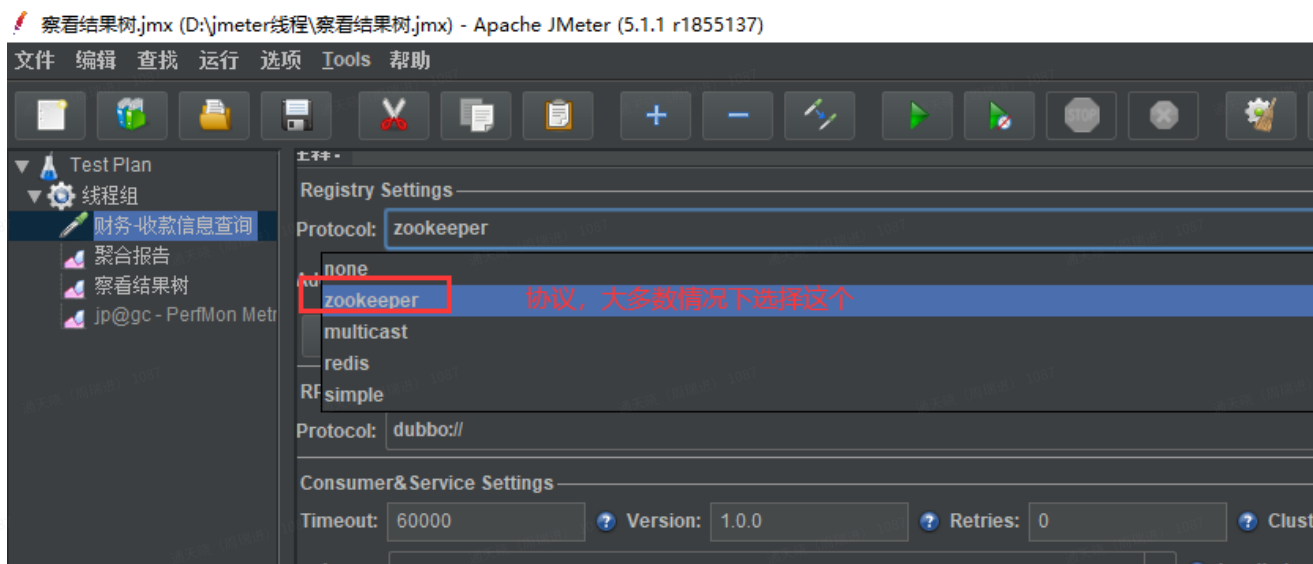
### 0、线程



### 1、在线程下面添加点击选择Dubbo Sample请求

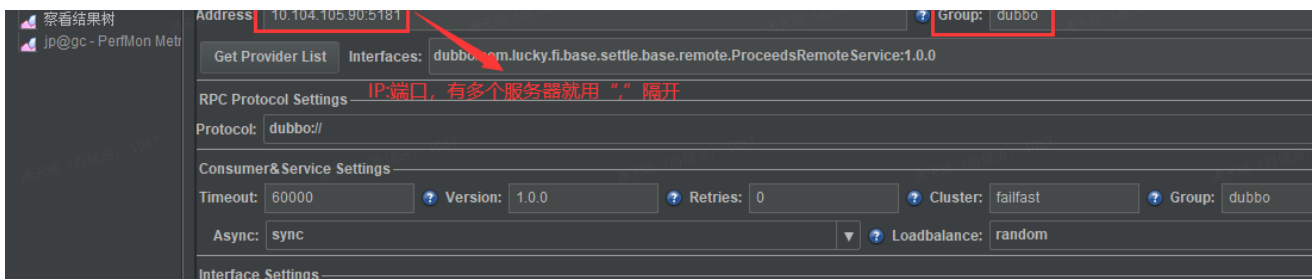


## 2、选择Protocol协议

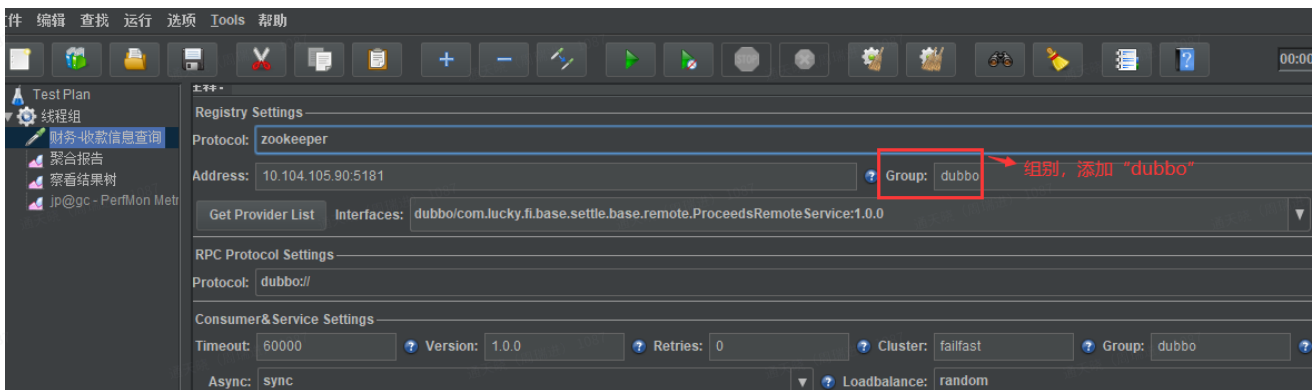


## 3、添加服务器和组别

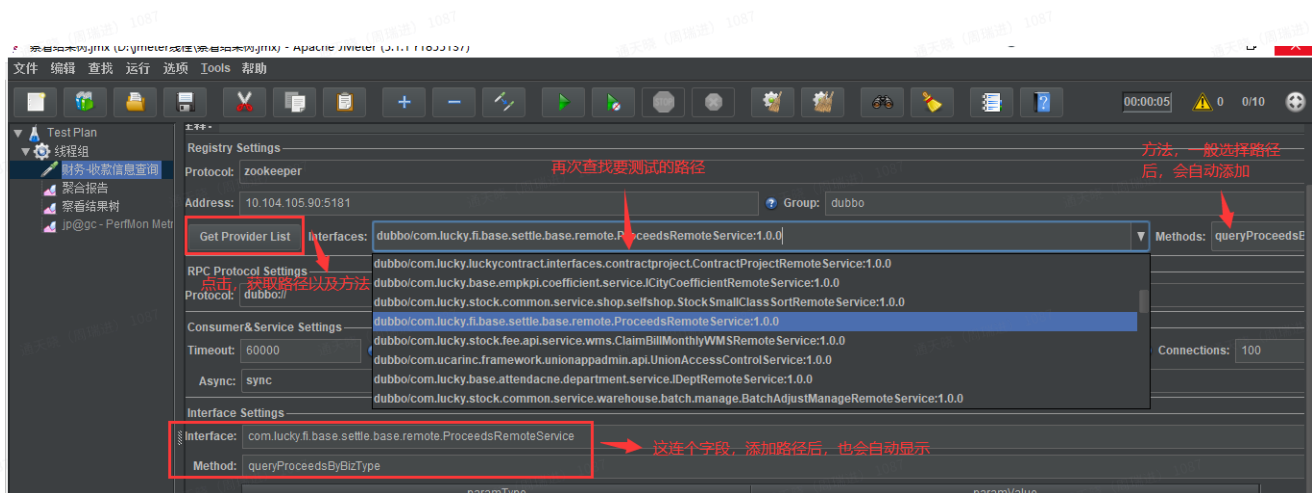




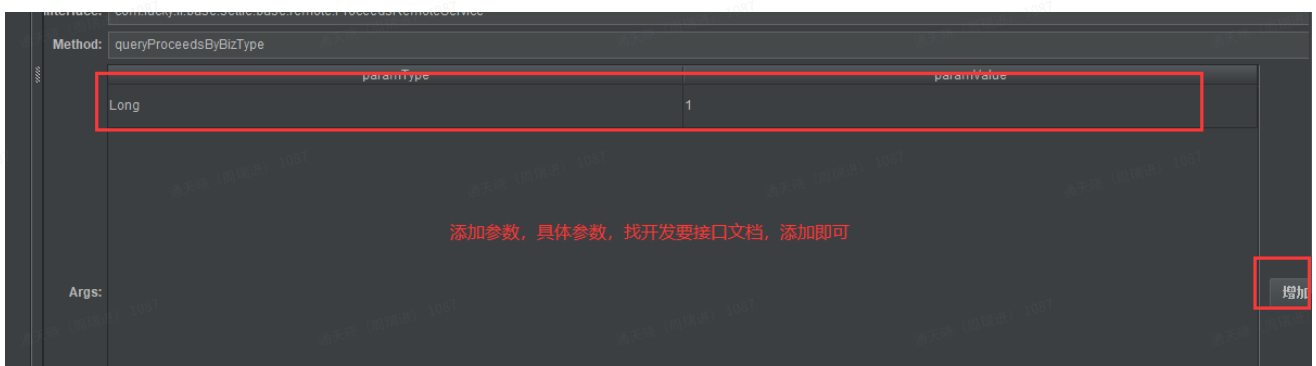
这里的address可以填 zk2test1.joymo.tech:5181



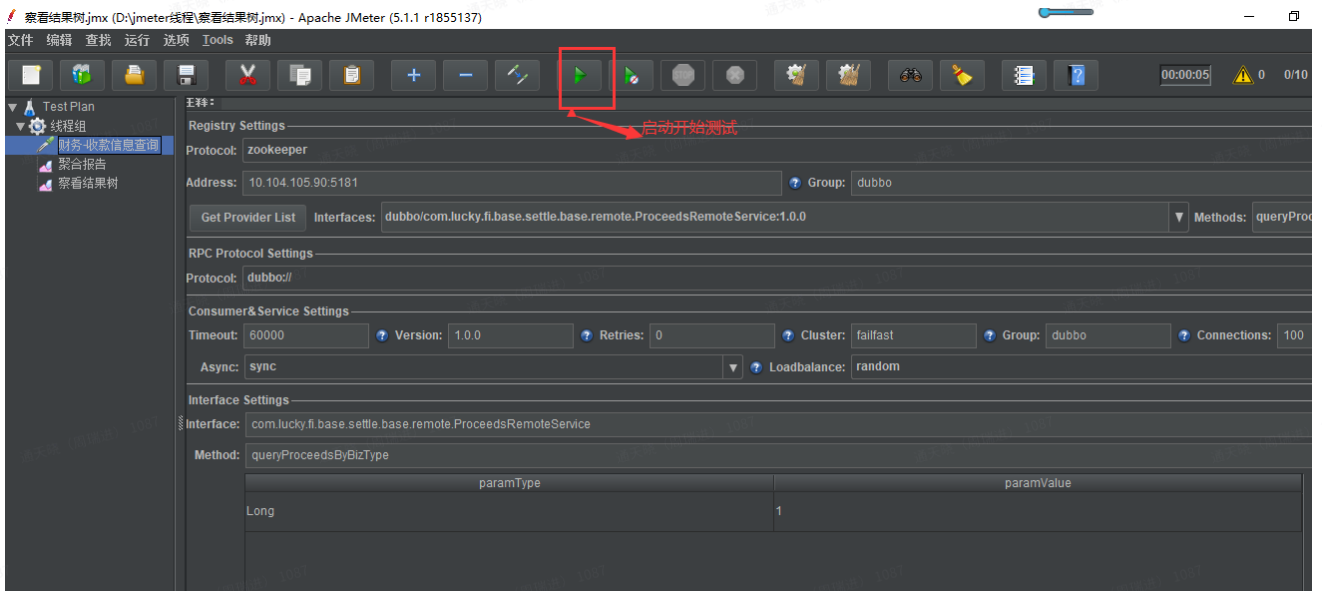
#### 4、选择要测试的RPC路径和方法



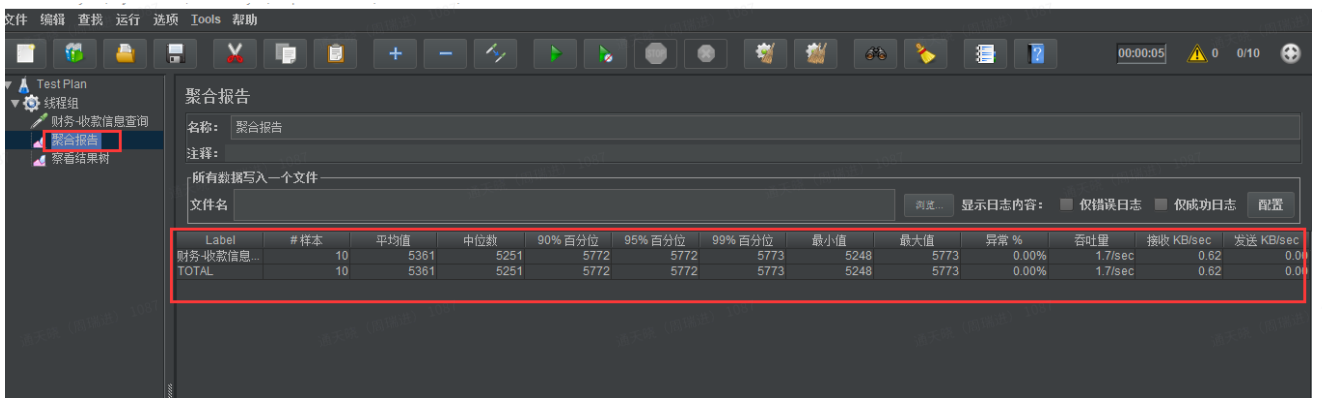
#### 5、添加参数



#### 6、完成数据添加后, 则可进行启动测试



## 7、监控结果



### 6.3.3 将压测脚本另存为jmx文件

保存好脚本后，然后即可参考 [5.压测任务执行流程](#) 中的步骤将压测脚本上传到压测平台上压测。

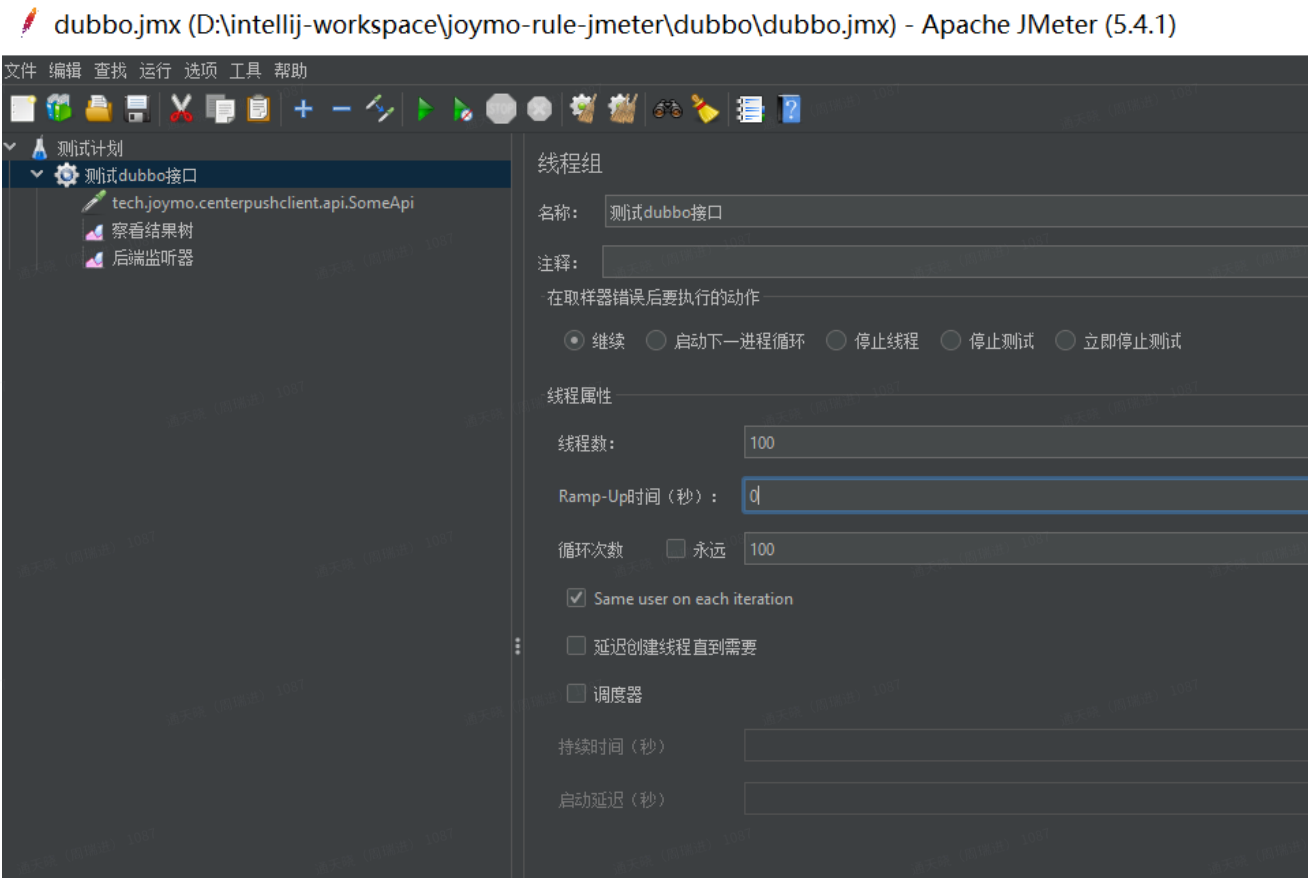
## 6.4关于jmeter脚本的压测参数设置

目前我们的测试1压测环境上有2台压测执行机：

实例列表						
删除实例						
<input type="checkbox"/>	实例(Pod)	状态	最新事件	CPU申请量 (core)	内存申请量 (GiB)	所在节点
<input checked="" type="checkbox"/>	jmeter-slaves-f66bcb988-bhxf9	运行中	--	2.00	4.00	10.88.153.80
<input checked="" type="checkbox"/>	jmeter-slaves-f66bcb988-k969v	运行中	--	2.00	4.00	10.88.150.168

这2台执行机会同时读取上传的jmx文件，并且根据jmx文件配置的线程数创建线程。因此我们在设置jmeter的线程参数的时候，需要考虑除以2。

比如说以下配置的线程数为100，循环次数100，那么2台执行机都会创建100个线程，循环100次，也就是说对于被测服务器来说，并发的线程数为2\*100=200个，实际调用次数为200\*100 = 20000次，而不是10000次



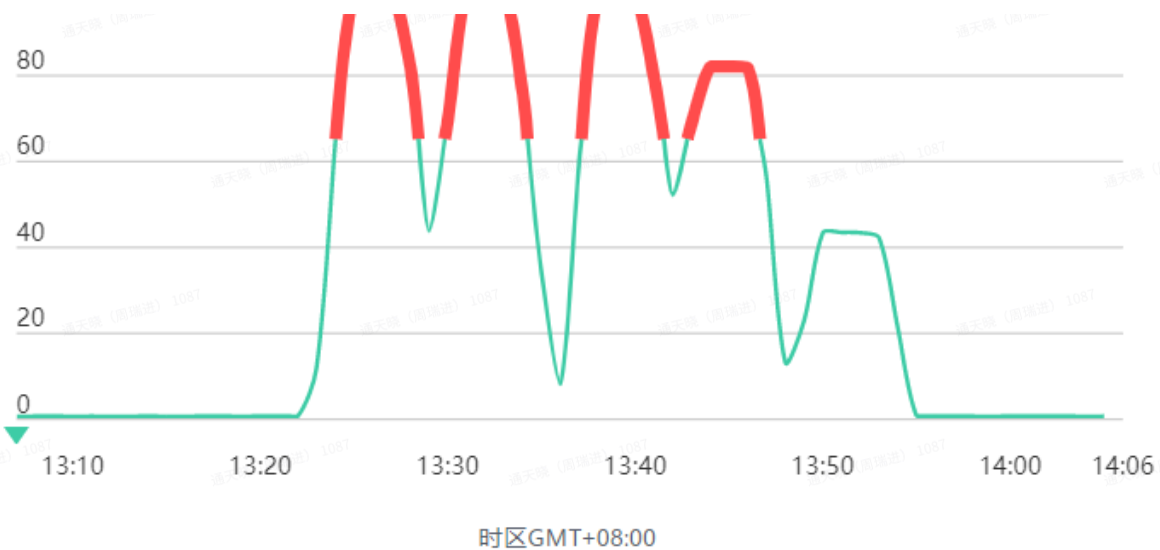
### 6.5性能测试模板和例子

- 🔗项目名称-性能测试方案.docx
- 🔗项目名称-性能测试报告.docx
- 📄规则引擎-性能测试方案
- 📄规则引擎-性能测试报告

### 6.6测试1环境被压测服务的cpu配置过低

**问题：**测试1环境各服务的cce容器cpu配置不高，基本为0.2core。当并发用户数为400，200，100，20时，业务服务器的CPU使用率超过了50%，压力过大。当并发用户数为10时，业务服务器的CPU使用率为43.17%，如下图所示：

CPU使用率(%)	最小值	平均值	最大值
100	0.43	35.33	99.83



**解决：**已与效能组的周瑞进确认，可以在压测时找他临时扩容被压测服务的cpu配置

## 6.7 jmeter执行机目前正在执行其他压测，无法执行的问题

**问题：**当其他同事正在执行压测，然后你也同时进行压测时，会报出这个错误：

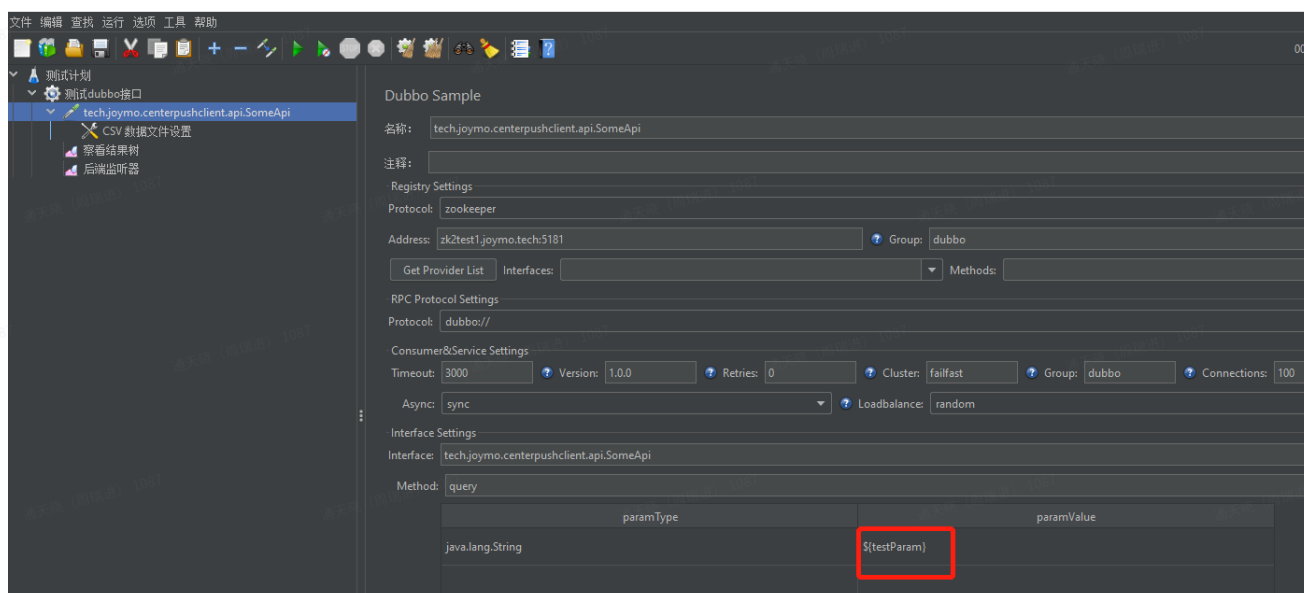
```
-n jmeter cp ${JMX_DIR}/${csvfile} ${slave_pods[j]}:/; let j=j+1; done; done; fi
29 ls: cannot access 'baidu/*.csv': No such file or directory
30 $ master_pod=$(kubectl get pod -n jmeter | grep jmeter-master | awk '{print $1}'); i
f [ `kubectl exec -ti -n jmeter ${master_pod} -- /bin/bash /load_test ${APPLICATION_NAM
E}-${JMX_DIR}.jmx ${APPLICATION_NAME}-${JMX_DIR}-report.csv | grep 'Engine is busy - ple
ase try later' | wc -w` -gt 0 ] ; then printf "Engine is busy - please try later\n"; exi
t 1; else kubectl cp jmeter/`kubectl get pod -n jmeter | grep jmeter-master | awk '{prin
t $1}'`:${APPLICATION_NAME}-${JMX_DIR}-report.csv ${APPLICATION_NAME}-${JMX_DIR}-report.
csv; fi
31 Unable to use a TTY - input is not a terminal or the right kind of file
32 SLF4J: Class path contains multiple SLF4J bindings.
33 SLF4J: Found binding in [jar:file:/jmeter/apache-jmeter-5.0/lib/log4j-slf4j-impl-2.11.
0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
34 SLF4J: Found binding in [jar:file:/jmeter/apache-jmeter-5.0/lib/ext/jmeter-plugins-dub
bo-1.3.8-jar-with-dependencies.jar!/org/slf4j/impl/StaticLoggerBinder.class]
35 SLF4J: Found binding in [jar:file:/jmeter/apache-jmeter-5.0/lib/ext/pepper-box-1.0.ja
r!/org/slf4j/impl/StaticLoggerBinder.class]
36 SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
37 SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
38 Engine is busy - please try later
42 ERROR: Job failed: command terminated with exit code 1
```



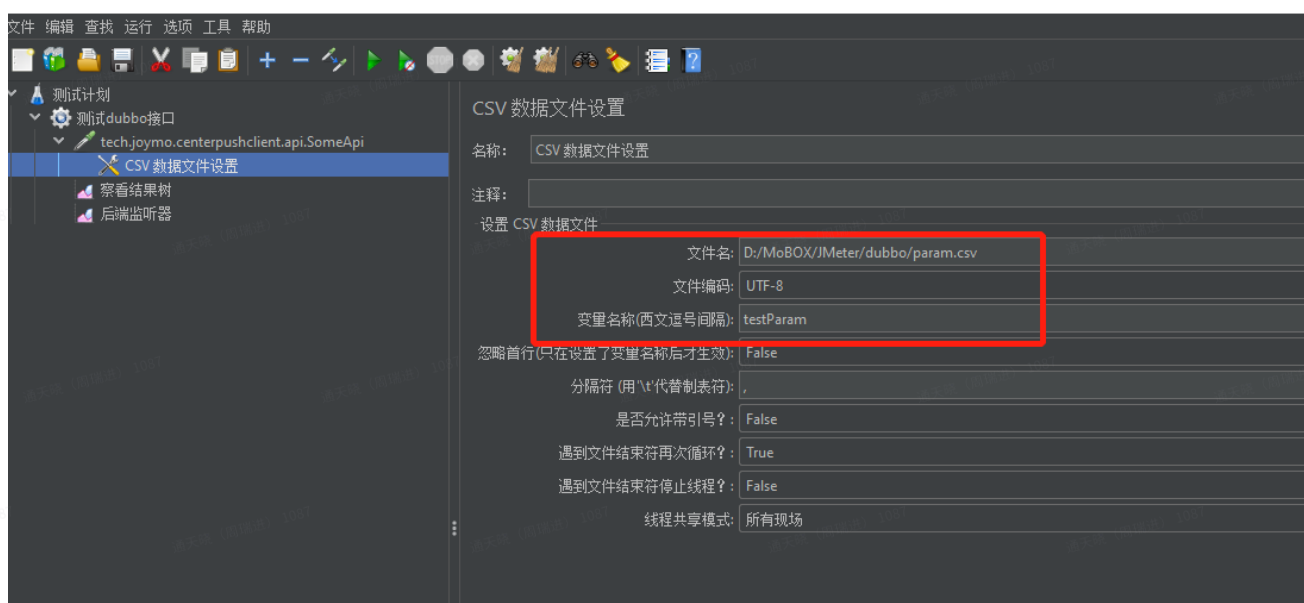
解决：按5.压测任务执行流程（重要）执行

## 6.8 dubbo接口如何指定动态参数csv文件

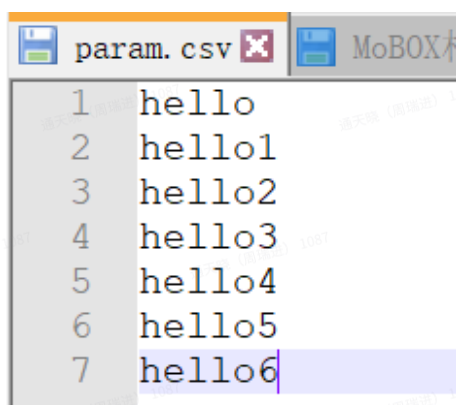
### 1.dubbo接口参数名使用\${}标识



### 2.在dubbo请求上添加一个csv数据文件设置配置，指定dubbo接口参数名，多个参数用英文逗号分隔



### 3.在csv文件中编写dubbo接口参数值，多个参数值用英文逗号分隔，顺序与参数名顺序保持一致





4.参数csv文件在本地jmeter调试成功后，需要调整参数csv数据文件路径为"/csv文件名"，具体操作见6.1 动态参数csv文件使用

## 6.9 支持自定义jar包

**场景：**在测试过程中，可能需要调用自定义jar包来生成测试数据或者使用java工具类来实现业务场景

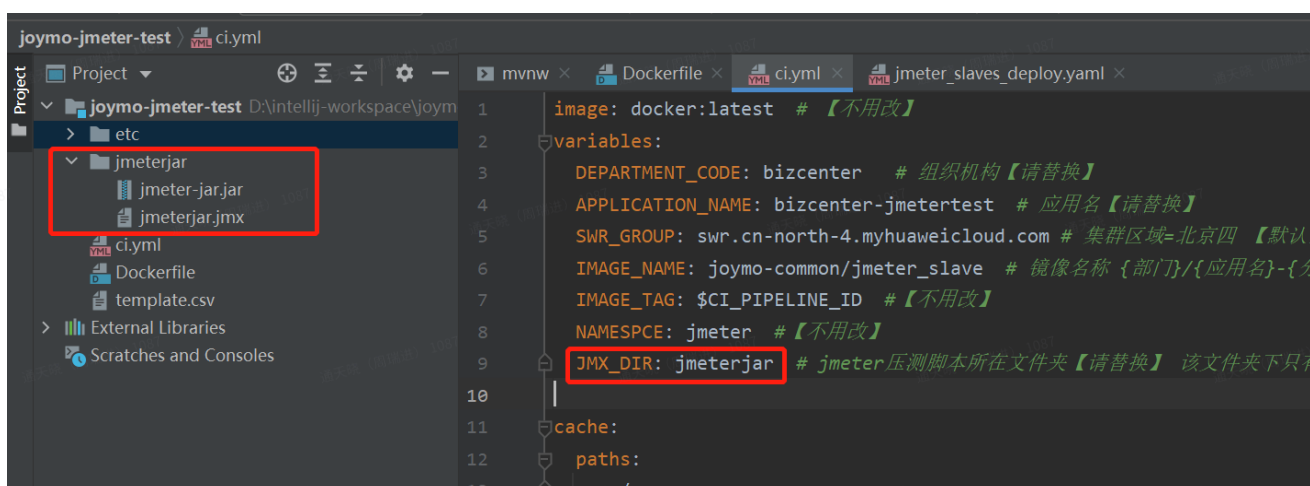
**如何实现：**使用公共压测项目<http://git.joymo.tech/bizgroup/joymo-jmeter-test>

**使用注意：**由于自定义jar包需要放置到jmeter压测机镜像中，每次压测都会重新构建压测机镜像，这会导致目前正在执行的压测任务中断。因此建议我们大家的业务接口压测脚本都放在这个公共压测项目中，通过CI/CD pipeline来保证压测任务的顺序执行，避免压测任务相互冲突影响。

**使用步骤：**

参见 5.压测任务执行流程，因为使用公共的压测项目，所以步骤 5.0.1在gitlba上创建压测项目 可忽略。

本地调试好jmeter脚本及自定义jar包后，把jmeter脚本及自定义jar包放到公共压测项目develop-1分支的文件夹下：



将代码推送到develop-1分支，即可触发压测，压测完成后可以点pipeline右侧的下载按钮下载报告：

