软件测试上机报告



第六次上机作业 – Lab6 Jmeter

**学 院 智能与计算学部**

**专    业 软件工程**

**姓 名 郎文翀 ­**

**学 号 3019244247**

**年 级 2019级**

**班    级 软件工程5班**

#### Experimental requirements

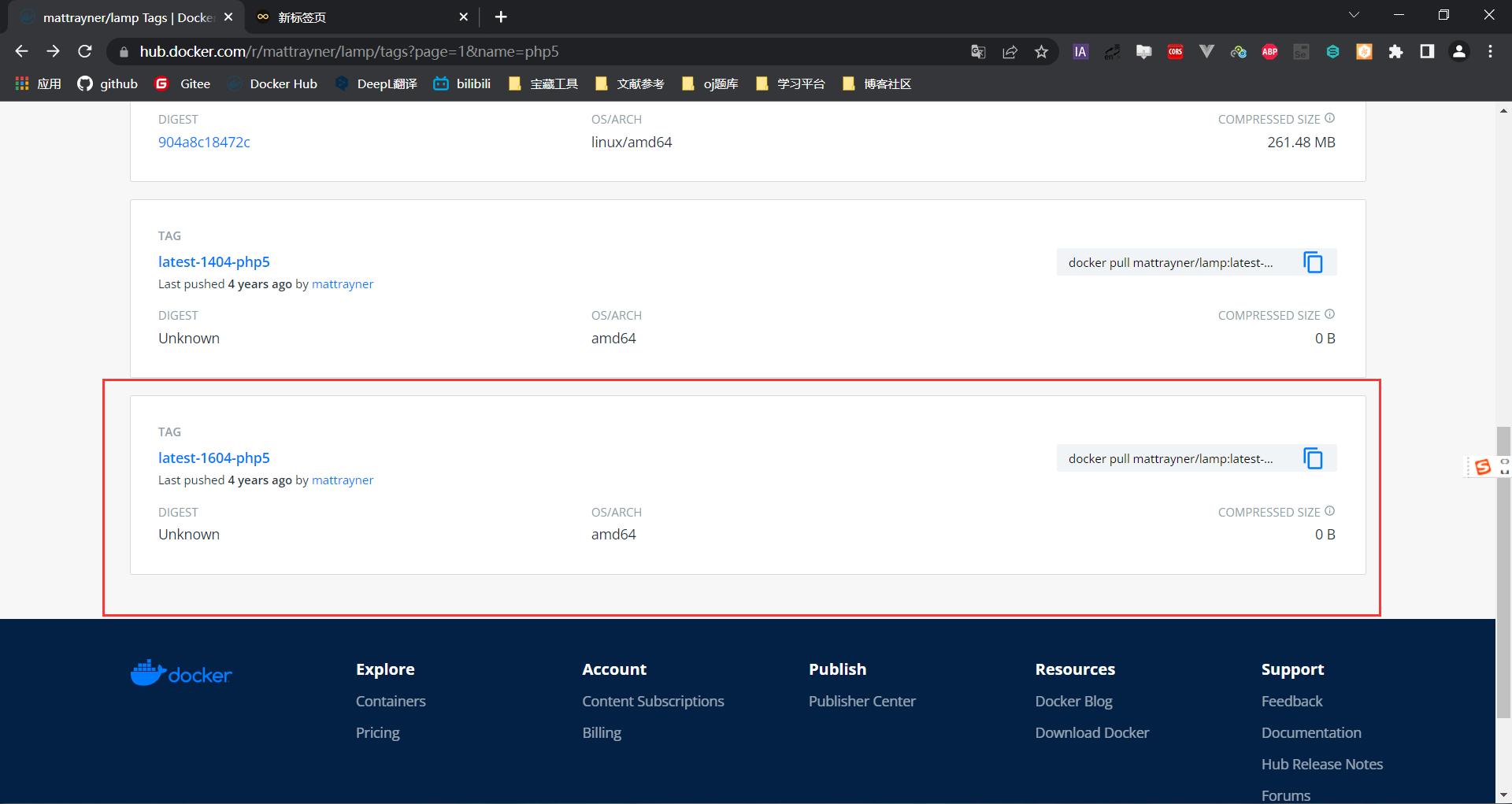
安装虚拟机，在虚拟机上搭建LAMP（Linux+Apache+Mysql+PHP）工作环境，并基于此安装待测系统，推荐ECShop（http://www.ecshop.com）或BugFree（https://www.bugfree.cn/）（也可以尝试最新的禅道项目管理软件），基于此进行Jmeter压力测试，并在测试后得出Jmeter测试报告，并根据top得出Linux服务器的CPU、Memory等信息。

* 1. 安装LAMP：Linux服务器（Centos/Ubuntu）、Apache、MySQL及PHP；
  2. 安装ECShop或BugFree待测系统（或类似B/S结构的系统）
  3. 使用Jmeter录制ECShop/BugFree创建新订单/Bug描述的行为，并尝试更改部分行为（例如不同商品的订单、Bug标题等）；
  4. 使用Jmeter进行5\*10、50\*20的压力测试并得出Jmeter Aggregate Report，同时运用sysstat对服务器信息进行统计。
  5. 添加Beanshell代码对压测结果进行分析，测试功能自定义。
  6. 基本要求：实践压力测试工具Jmeter
  7. 相关日期：
     1. 提交报告日期：2022.4.20按照格式提交到智慧树
  8. 实验报告应包括：
     1. Linux下top命令结果截图
     2. 访问的B/S系统截图
     3. Jmeter的Testplan展开截图
     4. Beanshell代码（功能可自定义）
     5. 运行Jmeter测试之后的Aggregate Report Result
     6. 运行Jmeter测试之后的服务器性能截图
     7. 根据sysstat结果对压力测试对服务器性能产生的影响作简要的分析
     8. 如发生错误，则简单描述错误并分析错误产生的原因

#### Configuration

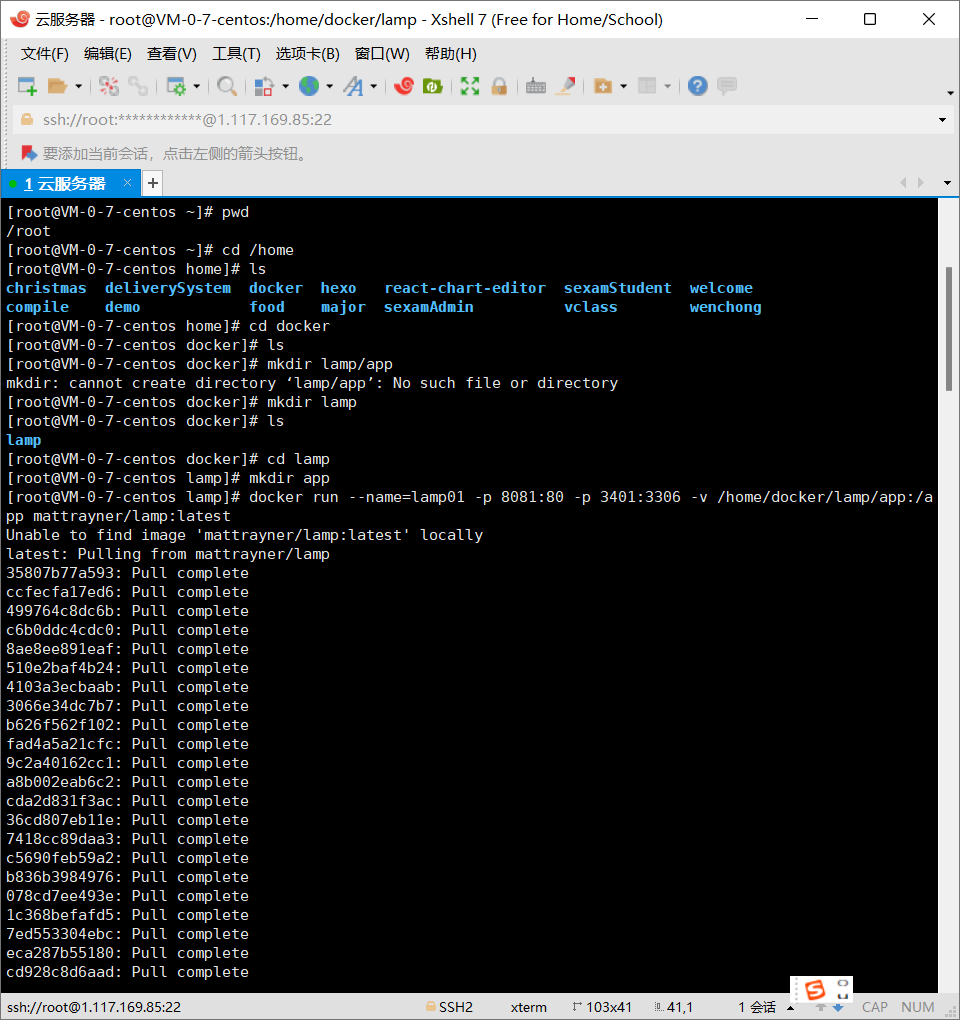
##### 2.1 配置LAMP环境

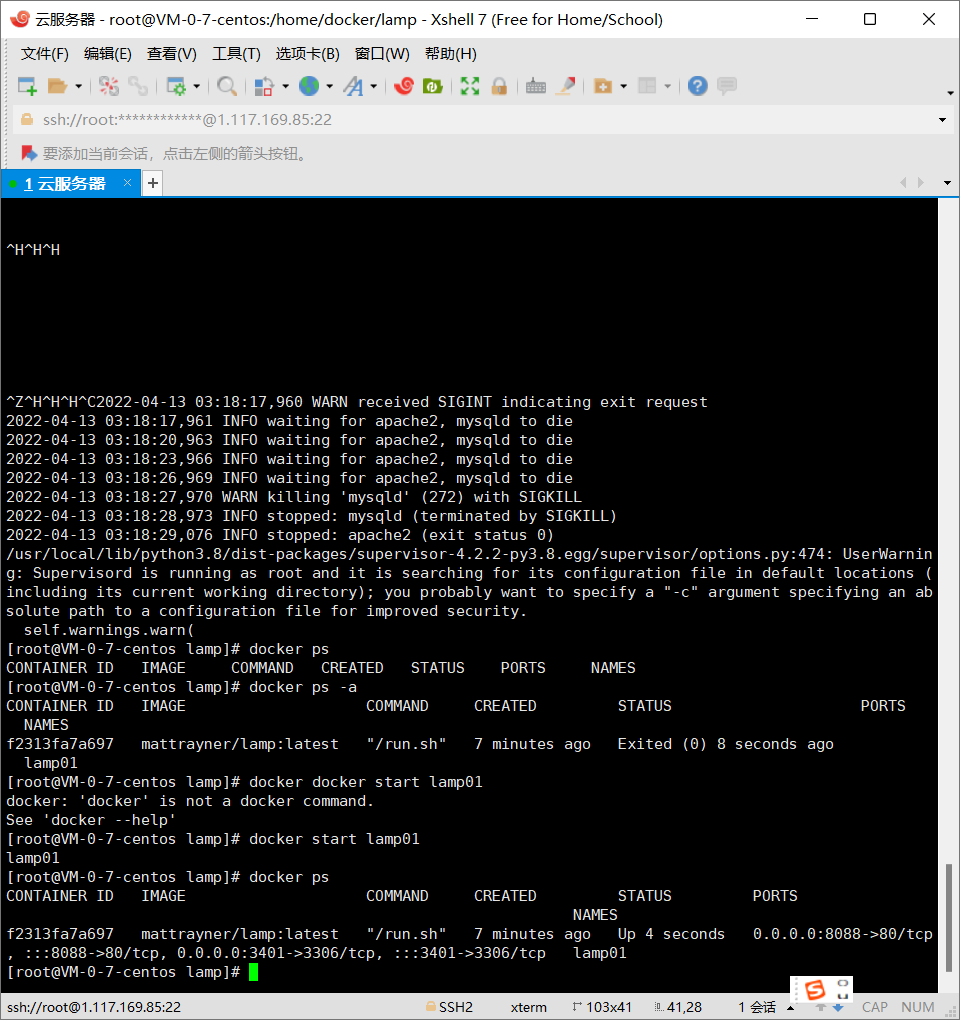
Lamp环境是实际上就是一个centos系统上同时集成了Apache/Mysql/PHP环境，我们需要安装这些环境来为后面的ecsshop提供运行环境。首先我们可以选择逐一配置，但是太过麻烦了，因此我选择使用docker来进行环境的安装，首先我们前往Dcoekrhub寻找lamp镜像，恶意看到如下的镜像符合我们的需求：



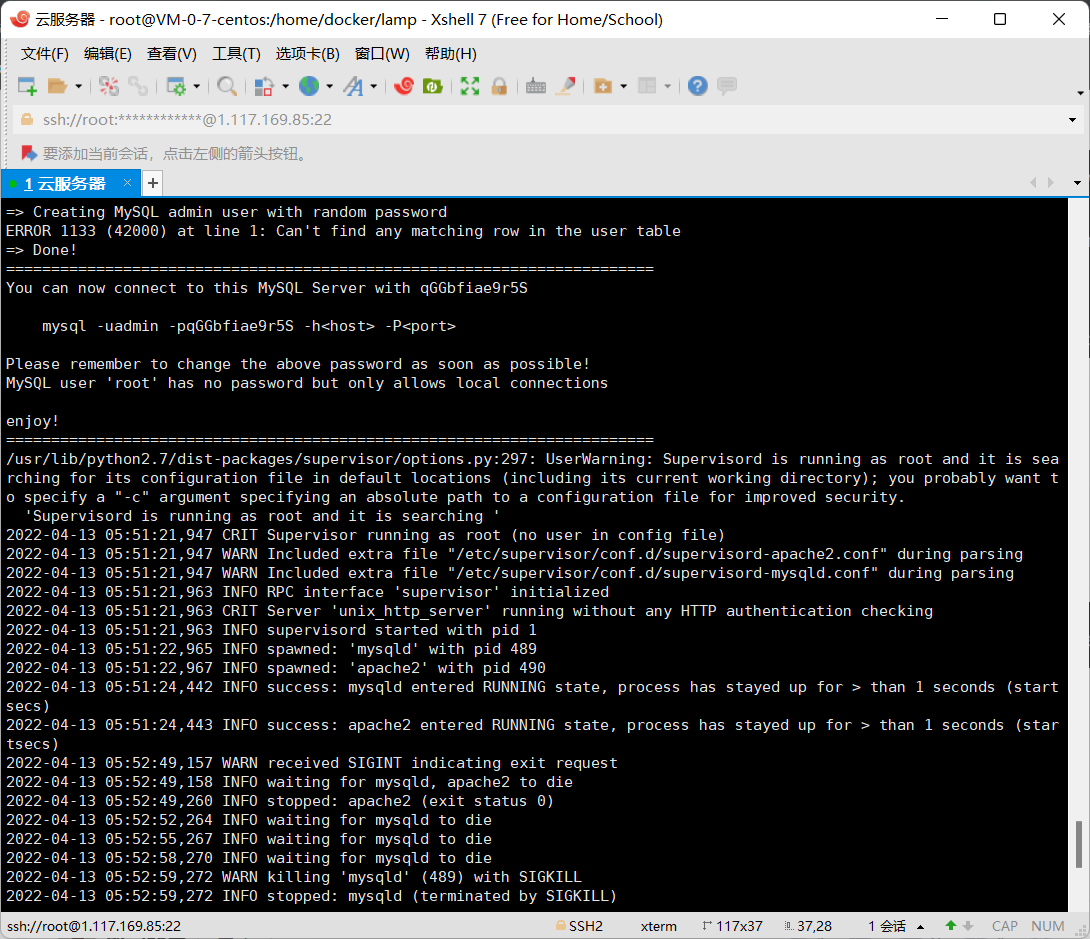
考虑到我找到的版本是较老版本，同时Mysql普遍使用5.7更加稳定，因此我选择这个版本，接下来我们输入如下命令创建这个版本的容器，由于我们需要将项目运行到容器中，而我们需要通过公网Ip访问到运行的ecsshop，因此我们在创建LAMP容器时要为其暴露一个端口，同时为了方便后面我们管理，我们进行卷挂载将需要放置项目的文件夹与服务器的/home/docler/lamp/app路径挂载，因此代码如下即可创建好容器：

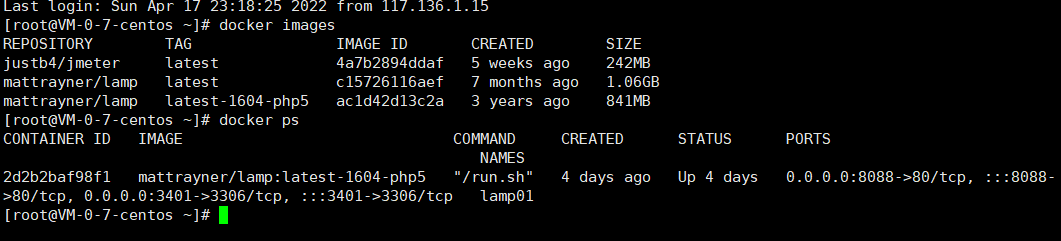
docker run --name=lamp01 -p 8088:80 -p 3401:3306 -v /home/docker/lamp/app:/app mattrayner/lamp:latest-1604-php5





创建完成以后我们可以输入docker images和docker ps查看到对应的LAMP容器，可以看到他的状态为up正在运行：

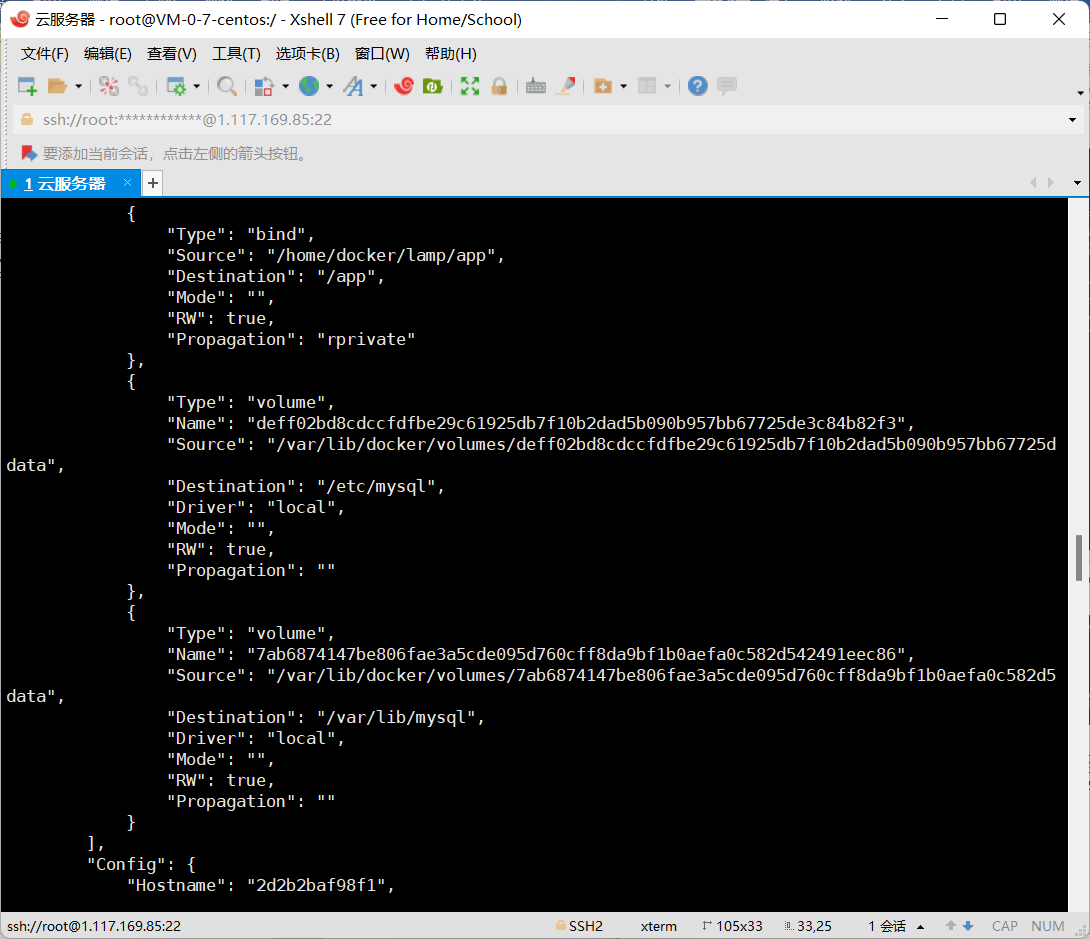




然后我们输入指令

Docker inspect lamp01

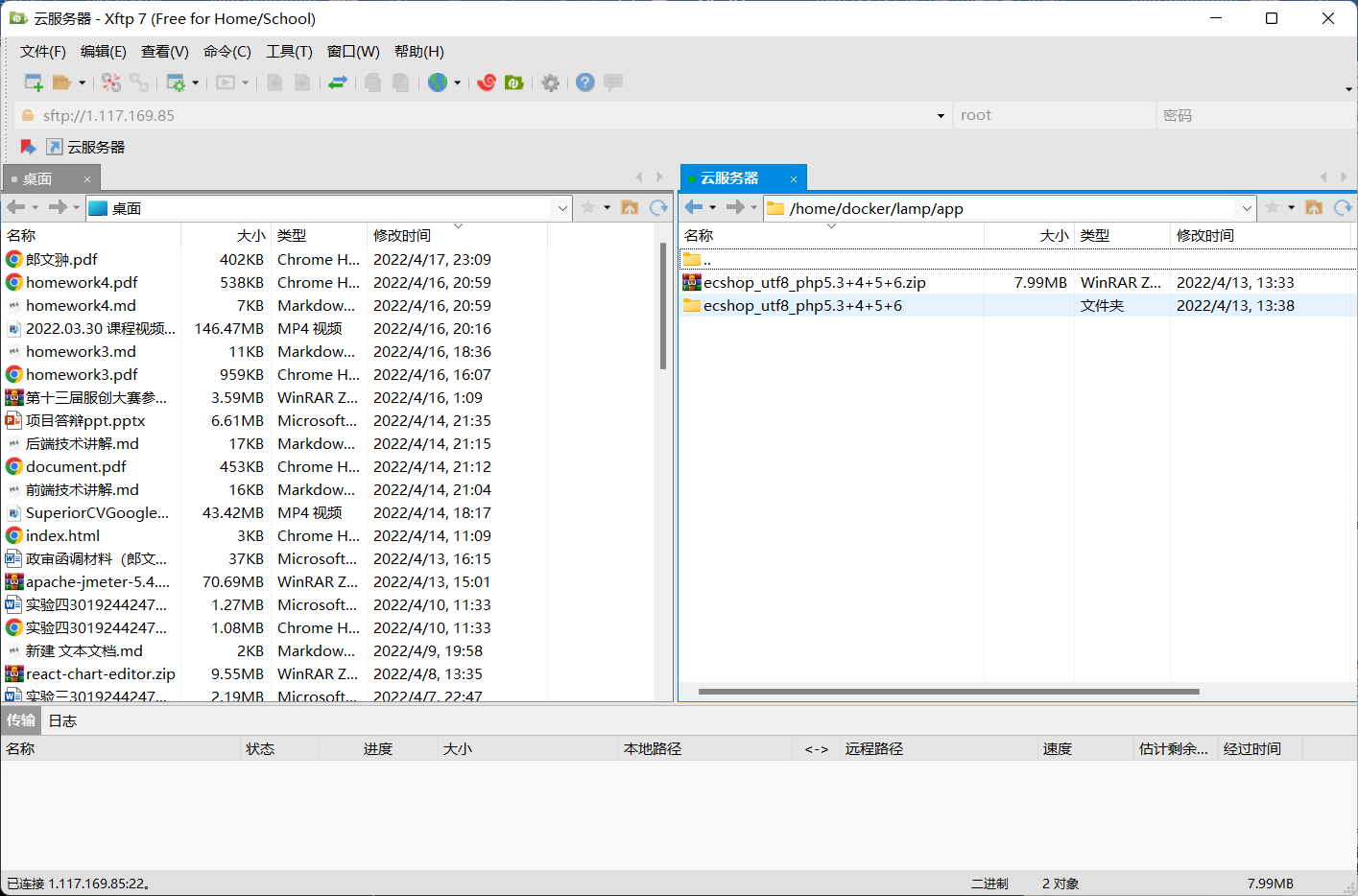
可以查看卷挂载是否正确，如下是lamp01容器的详细信息，我们可以看到卷挂载正确：



自此我们通过docker就轻松配置完成了Lamp环境。只不过Mysql用的是默认密码这也没有什么大问题，通过输入docker log我们就可以看到mysql账户的密码。

##### 2.2 安装ecsshop项目

接下来我们需要将ecsshop放置到lamp01容器中让其运行，然后我们就可以通过公网ip+端口查看到这个项目了。之前我们卷挂载了app路径，实际上就是为了方便这里进行ecsshop的安装，我们将提前准备好的ecsshop压缩包通过xftp上传到服务器上的/home/docker/lamp/app下如下图所示：



然后我们解压缩这个项目，由于我们已经将本地服务器上的app文件夹与lamp01容器的app文件夹进行了卷挂载，因此实际上此时相当于lamp01下也已经安装上了cesshop，并且由于docker容器正在运行，因此我们可以通过之前暴露的端口号8088访问到这个项目了：

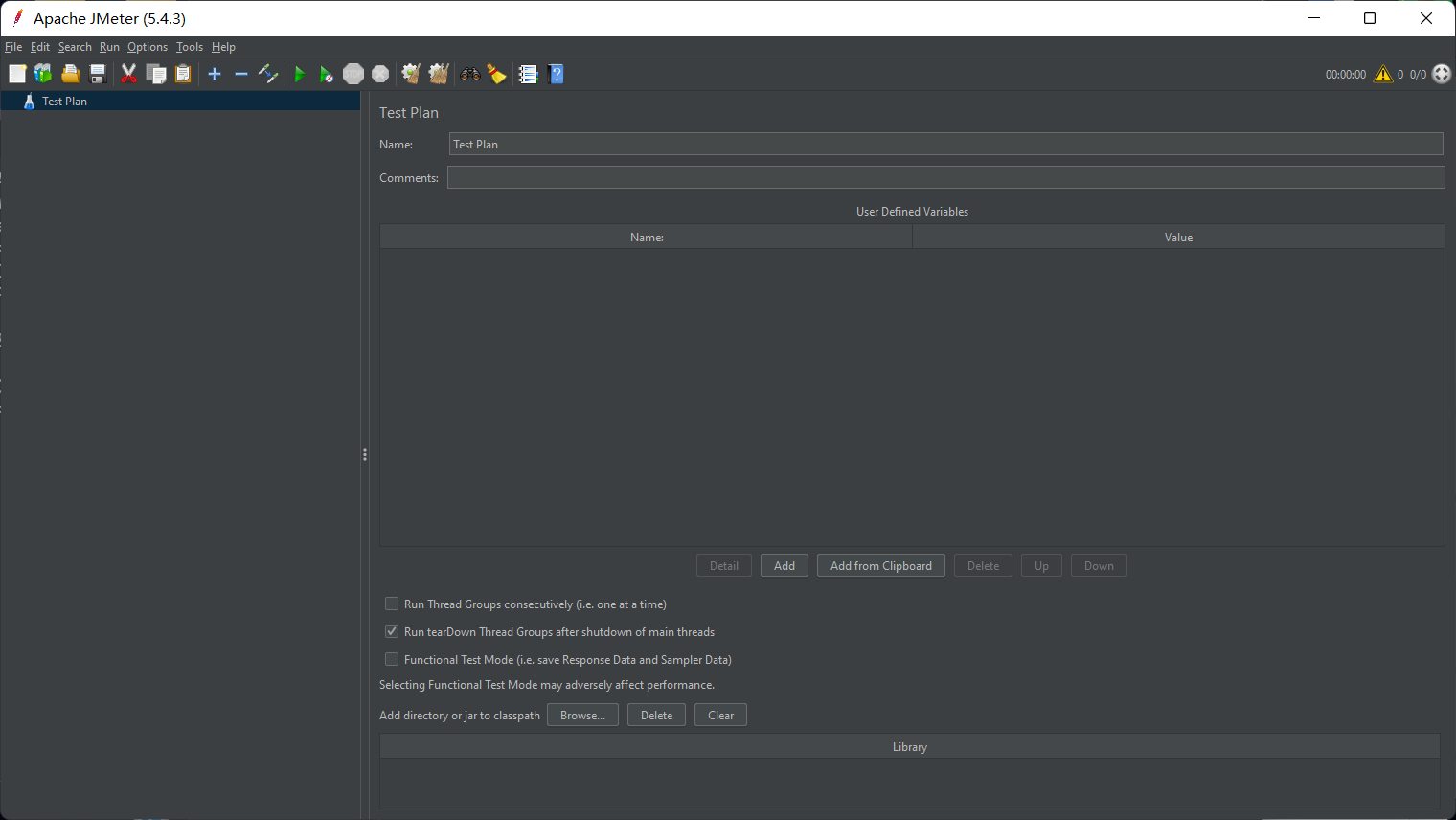
我的服务器的公网ip是1.117.169.85，因此访问项目的url就是<http://1.117.169.85:8088/ecshop_utf8_php5.3+4+5+6/>

我们进入这个项目之前首先需要注册一下管理员账号，并且配置一下mysql，接下来我们提交配置即可以进入进入到这个项目中如下图所示：



##### 2.3 安装jmeter并启动运行

现在我们已经配置好了项目并且部署到了容器中，可以通过公网url访问到这个项目了，接下来我们就来安装一下jmeter来对我们部署的项目进行压力测试。我们下载下来jmeter安装以后不需要配置环境变量，直接解压缩然后进入到文件夹中可以看到在/bin/下有一个jmeter.sh脚本文件可以直接启动jmeter,因此我们在jmeter根路径下输入 sh jmeter.sh启动jmeter如果可以正常启动，那么就可以看到如下图所示的UI界面了：

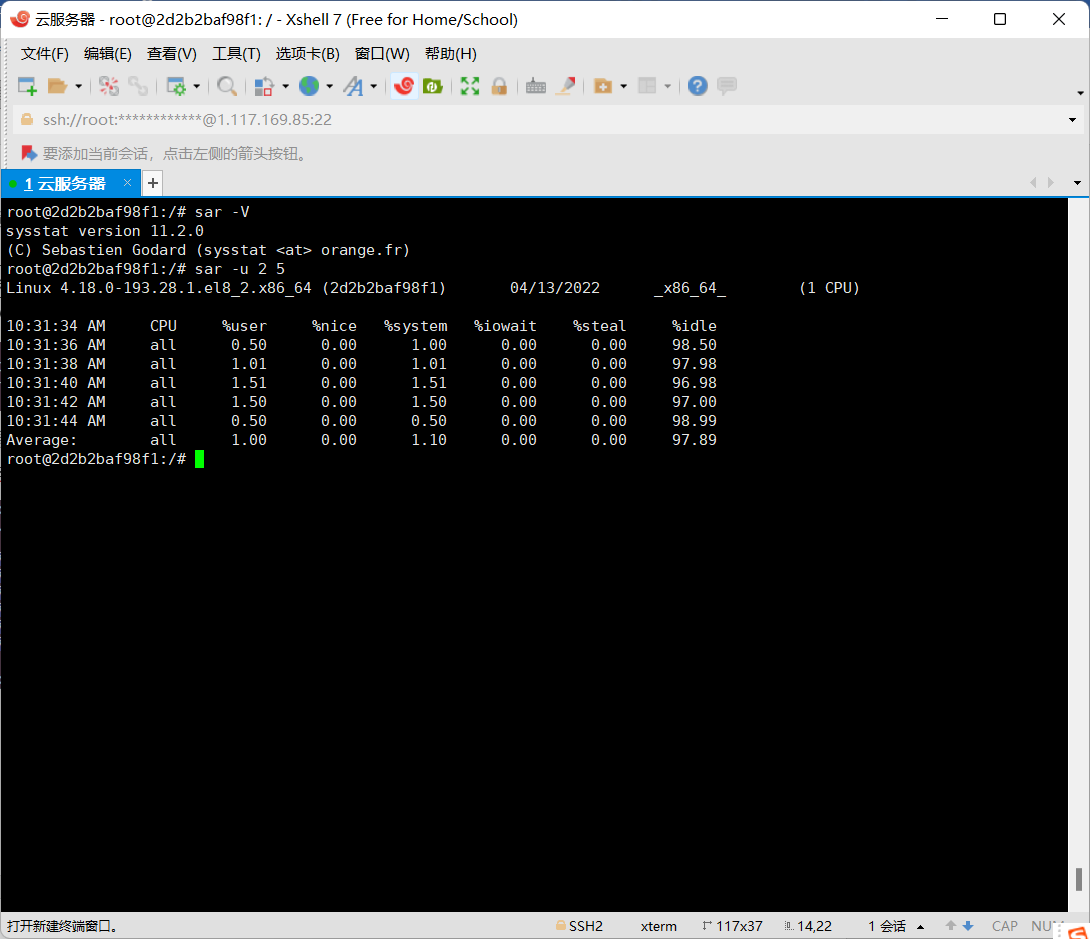


##### 2.4 安装systat

由于我们是在容器内部署的项目，因此我们在查看负载信息时应该是查看容器内部，所以首先我们输入如下执行进入容器：

Docker exec -it lamp01 /bin/bash

然后在这里输入top首先查看压力测试前的负载情况，具体图片以及分析请见第三部分，然后我们还需要在使用systat查看负载信息，因此首先需要在内部安装systat,使用yum install systat安装即可，如果成功安装，我们输入sar -V应该可以查看到版本号如下图所示：

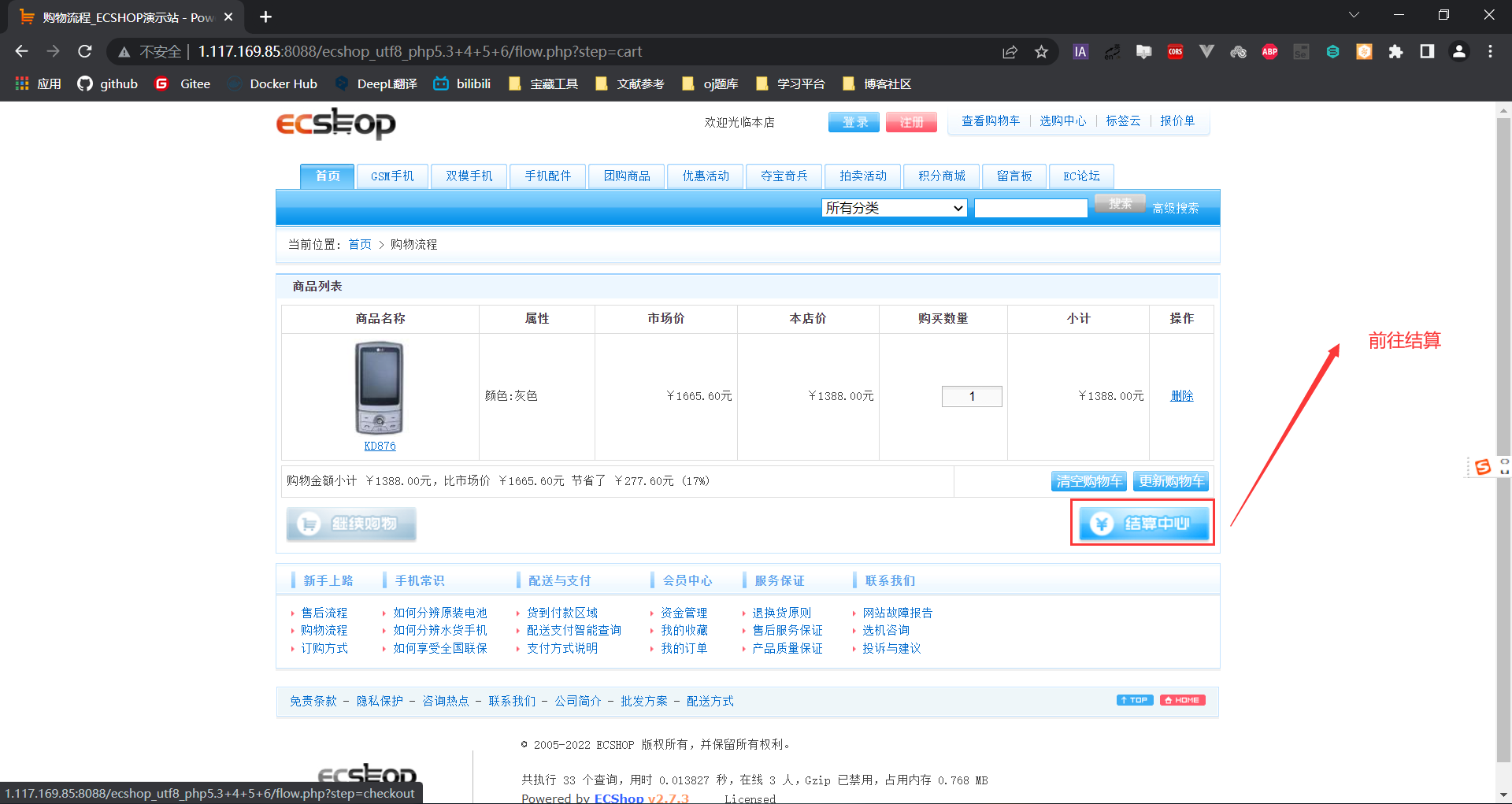


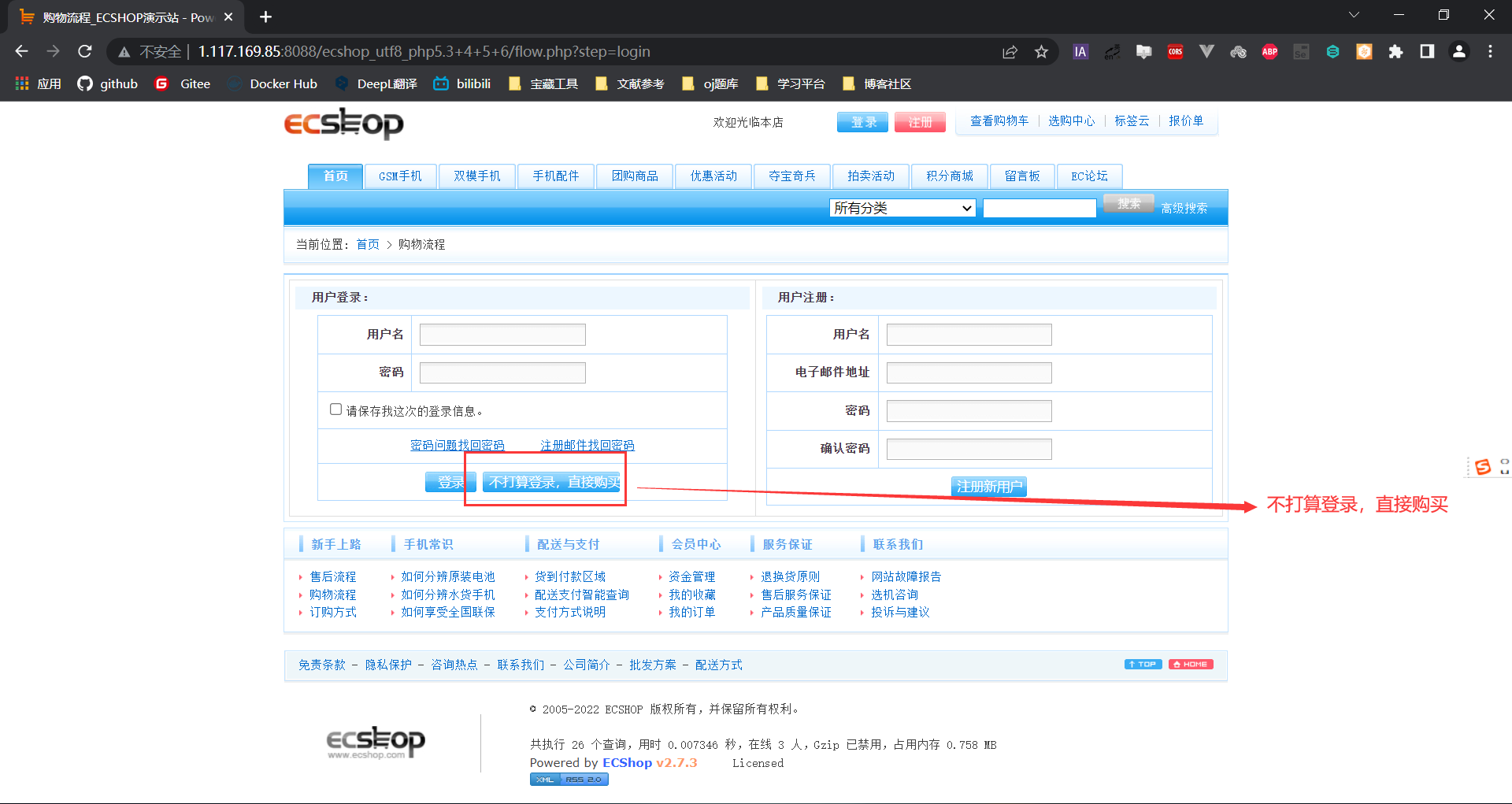
##### 2.5 录制脚

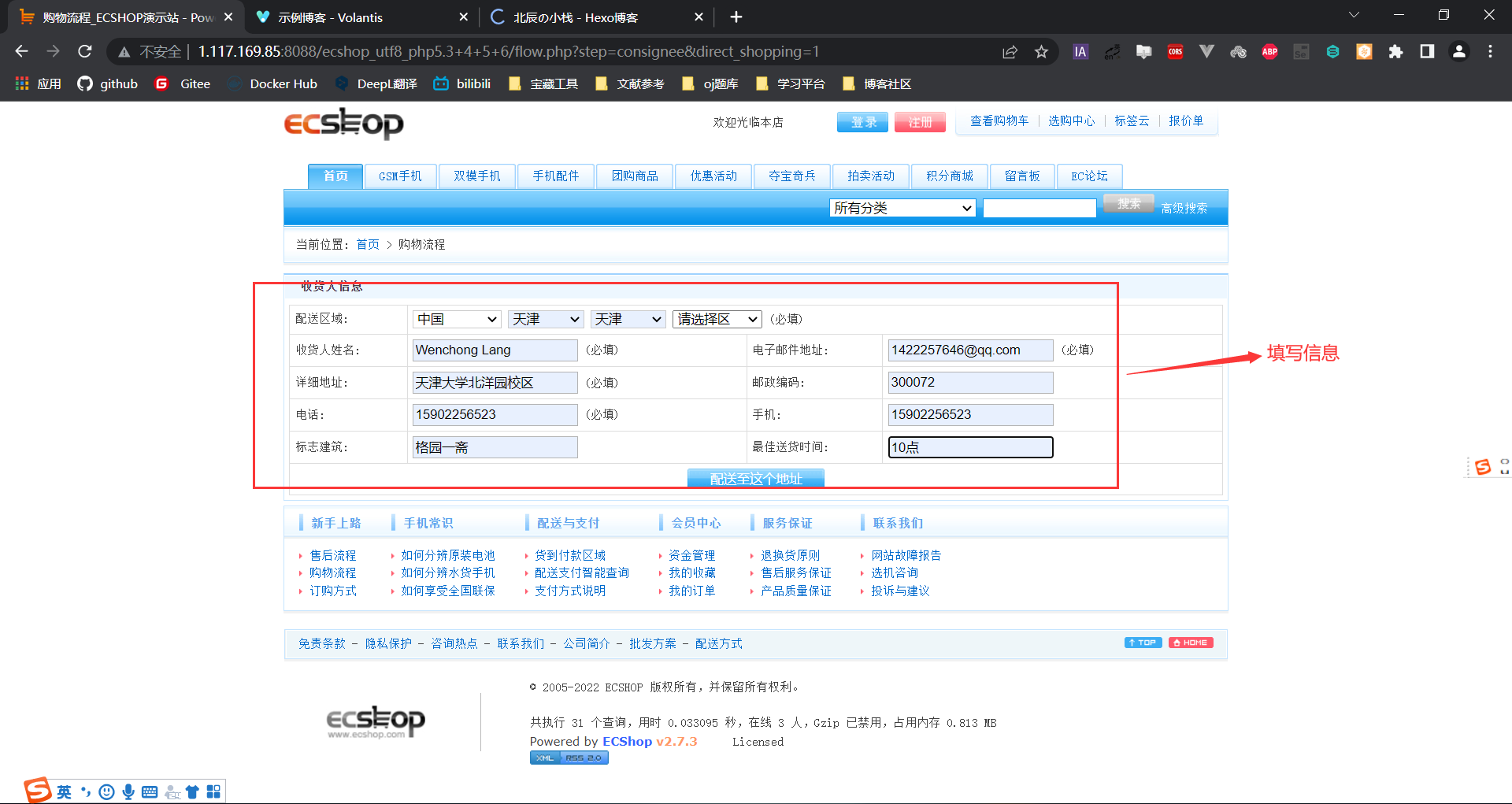
然后我们启动jmeter尝试对一个操作进行录制，这里我们发现可以点击商品加入购物车，然后结算，这个操作具体流程如下：

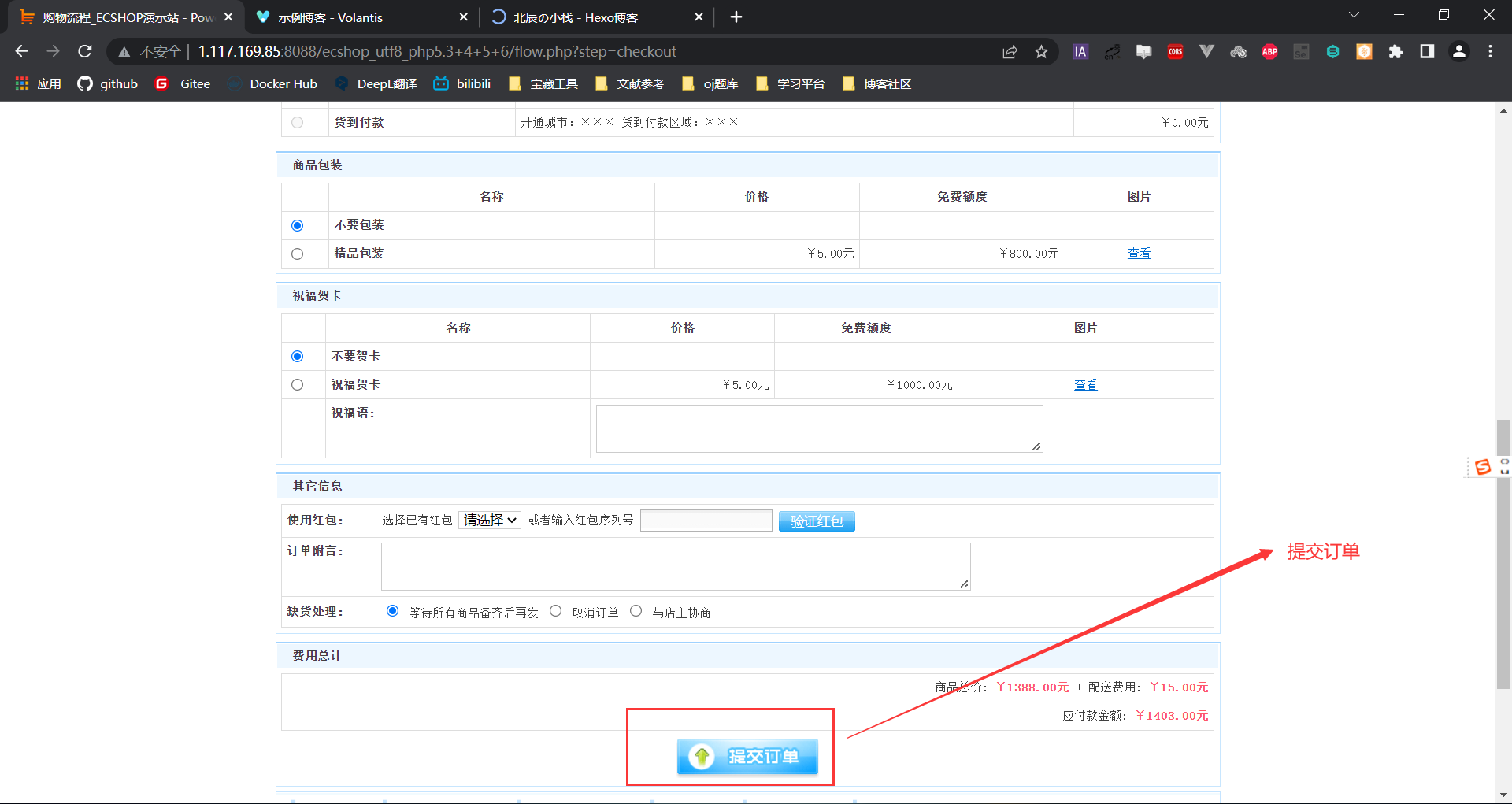




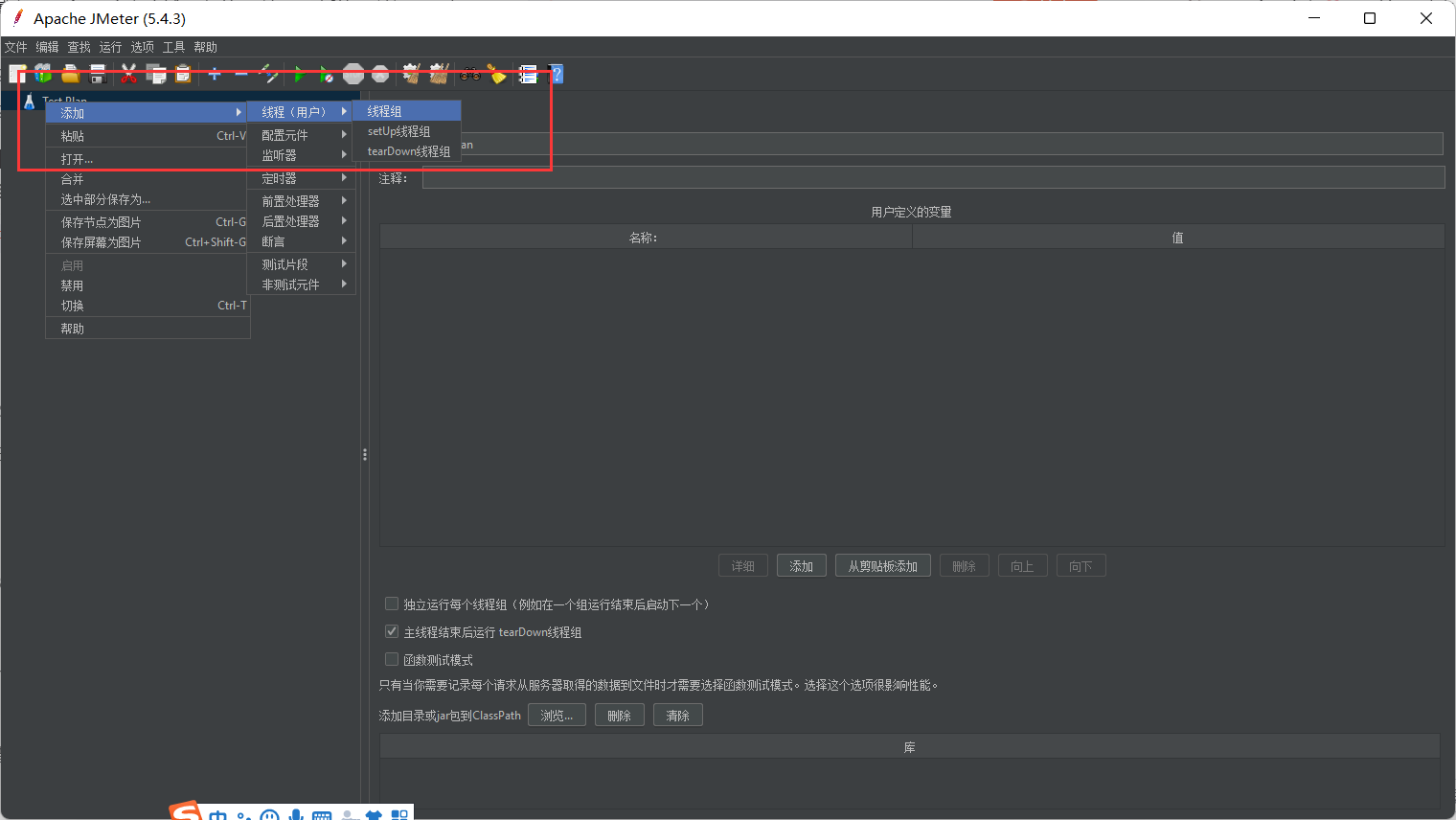


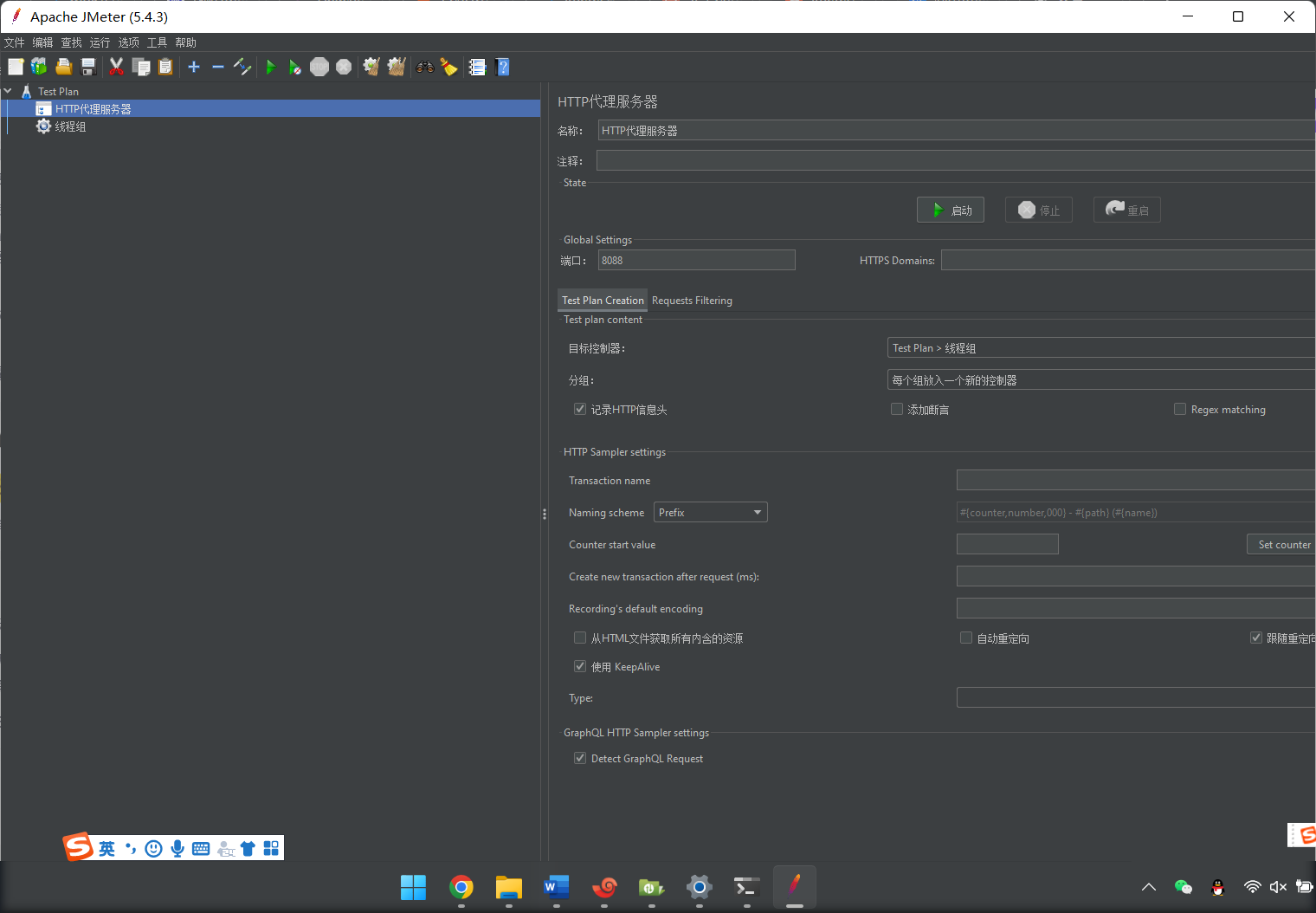




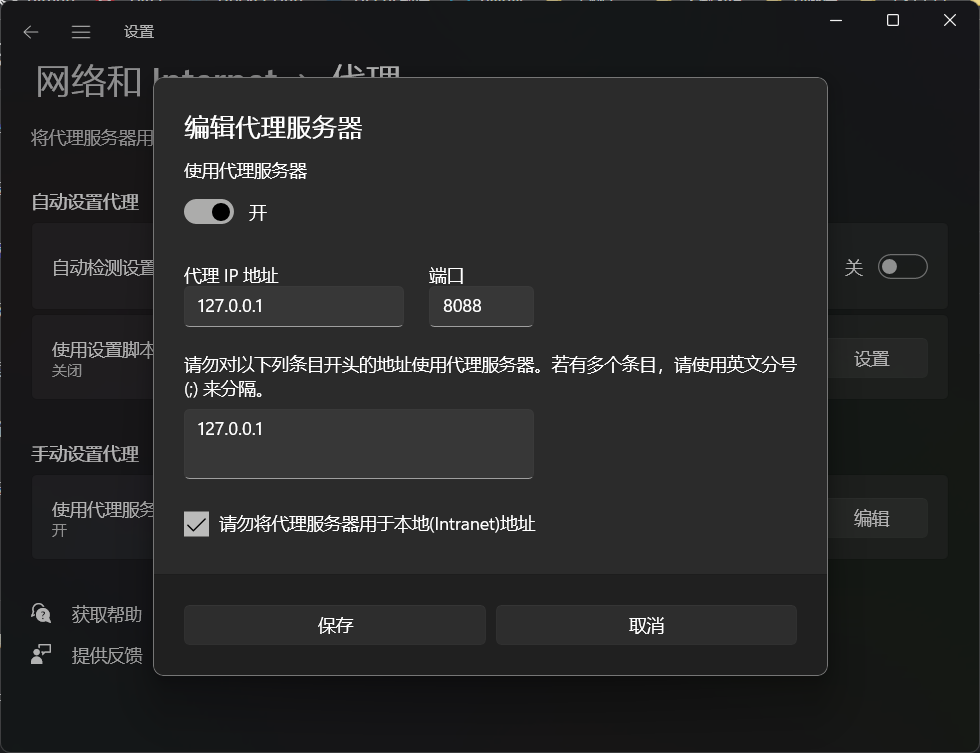


以上就是ecsshop一个购买商品到计算订单的过程，接下来我们就尝试录制这个过程，首先我们需要在jmeter中，然后首先我们创建一个线程组，命名随意，他是用来存储记录我们的操作的，一会进行压力测试时也是不断地调用执行这个线程组，接下来我们还需要创建一个HTTP代理服务器用来录制我们的操作，然后以后在进行压力测试时他将会自动执行线程组中的操作来帮助我们完成多次重复的接口测试从而实现压力测试。

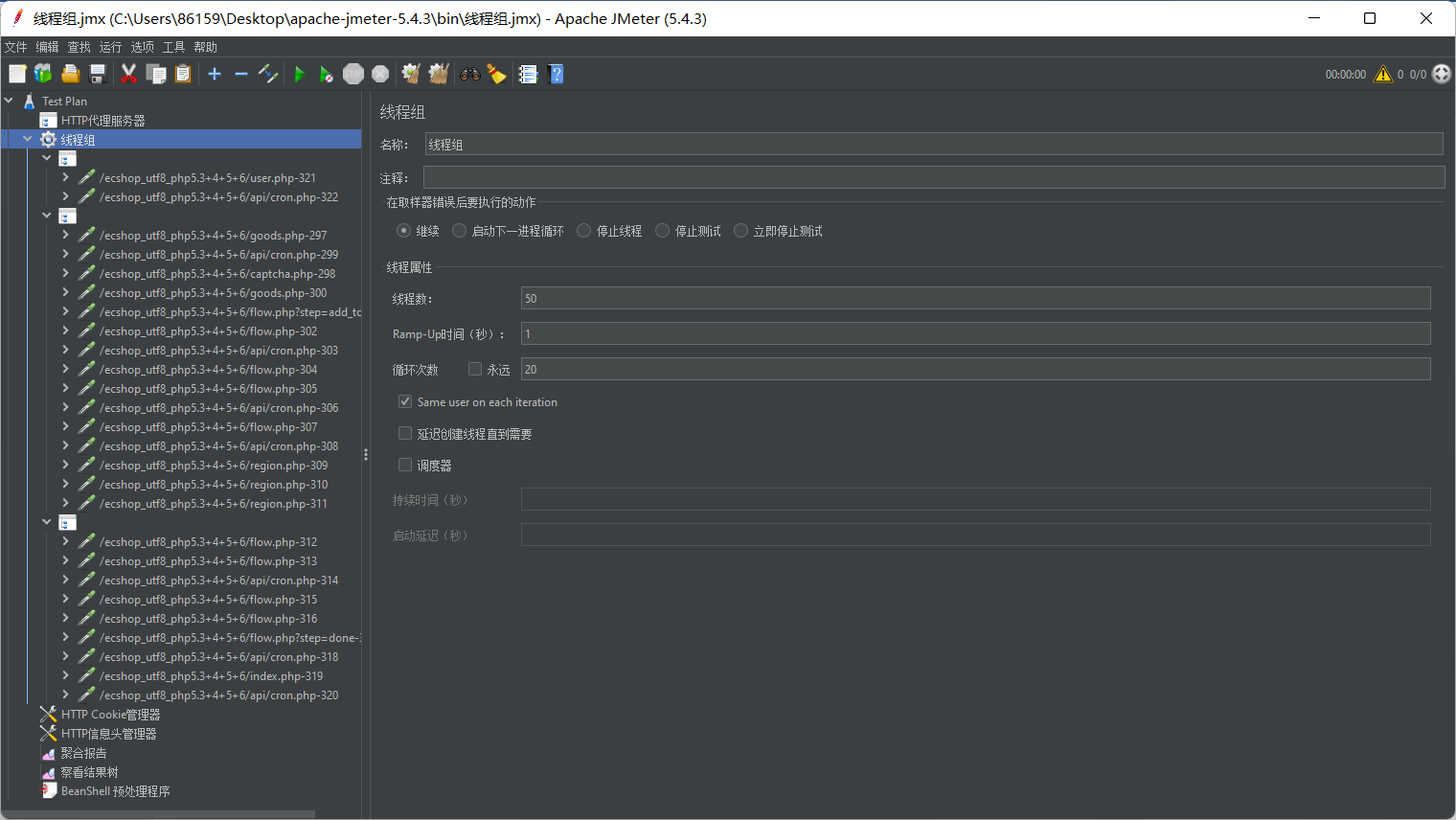




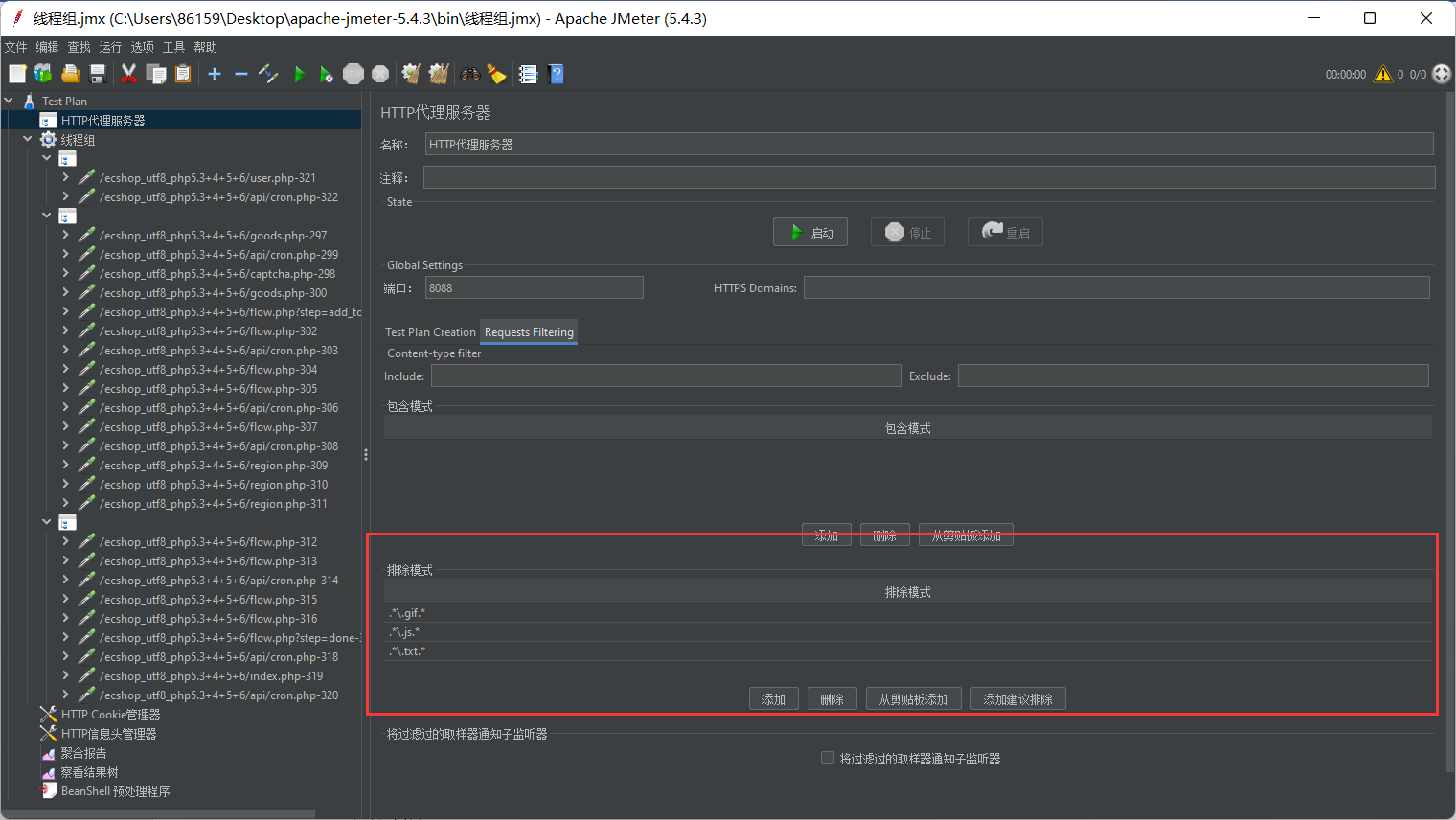
配置完上面的两个组件以后，我们只需要点击启动就可以开启录制操作，但是我们在这之前还需要配置一下浏览器代理，我们打开chrome浏览器然后点击设置搜索代理即可打开如下界面：



我们需要将代理服务器开启，这样http代理服务器接下来才能执行，ip地址就是回环地址127.0.0.1，端口号就是http代理服务器上的端口即可。然后我们点击启动以后开始录制执行一遍刚刚上图所演示的购买商品流程，录制完成以后结束我们就可以看到线程组中出现了大量的记录：



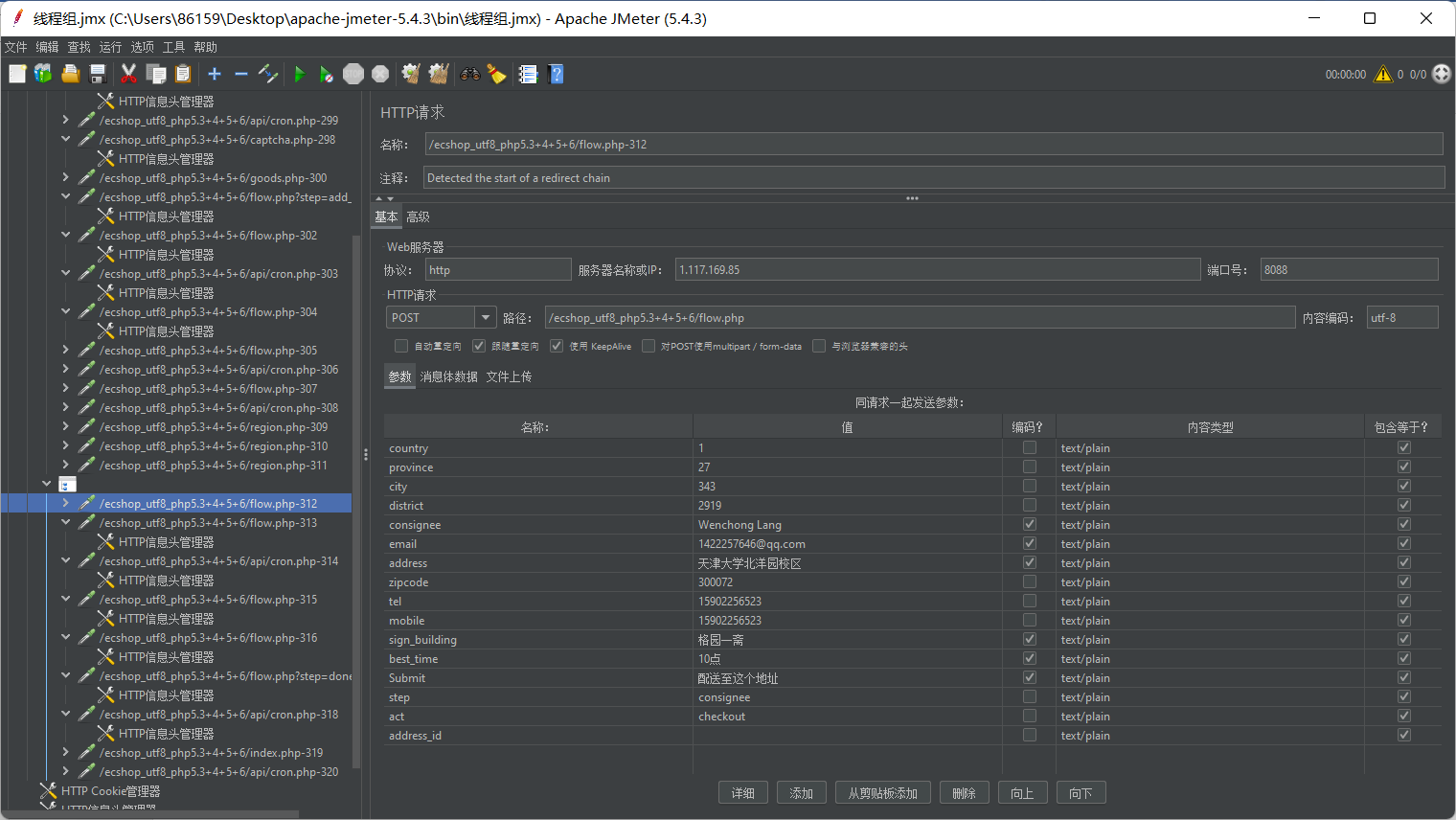
但是里面包含了太多无用的信息，因此我们可以借助Http代理服务器中的筛选设置筛掉无用的信息：



这样线程组中剩下的就是我们可能会用到的有用的操作，自此我们就完成了脚本记录。

##### 2.6 修改记录的部分行为

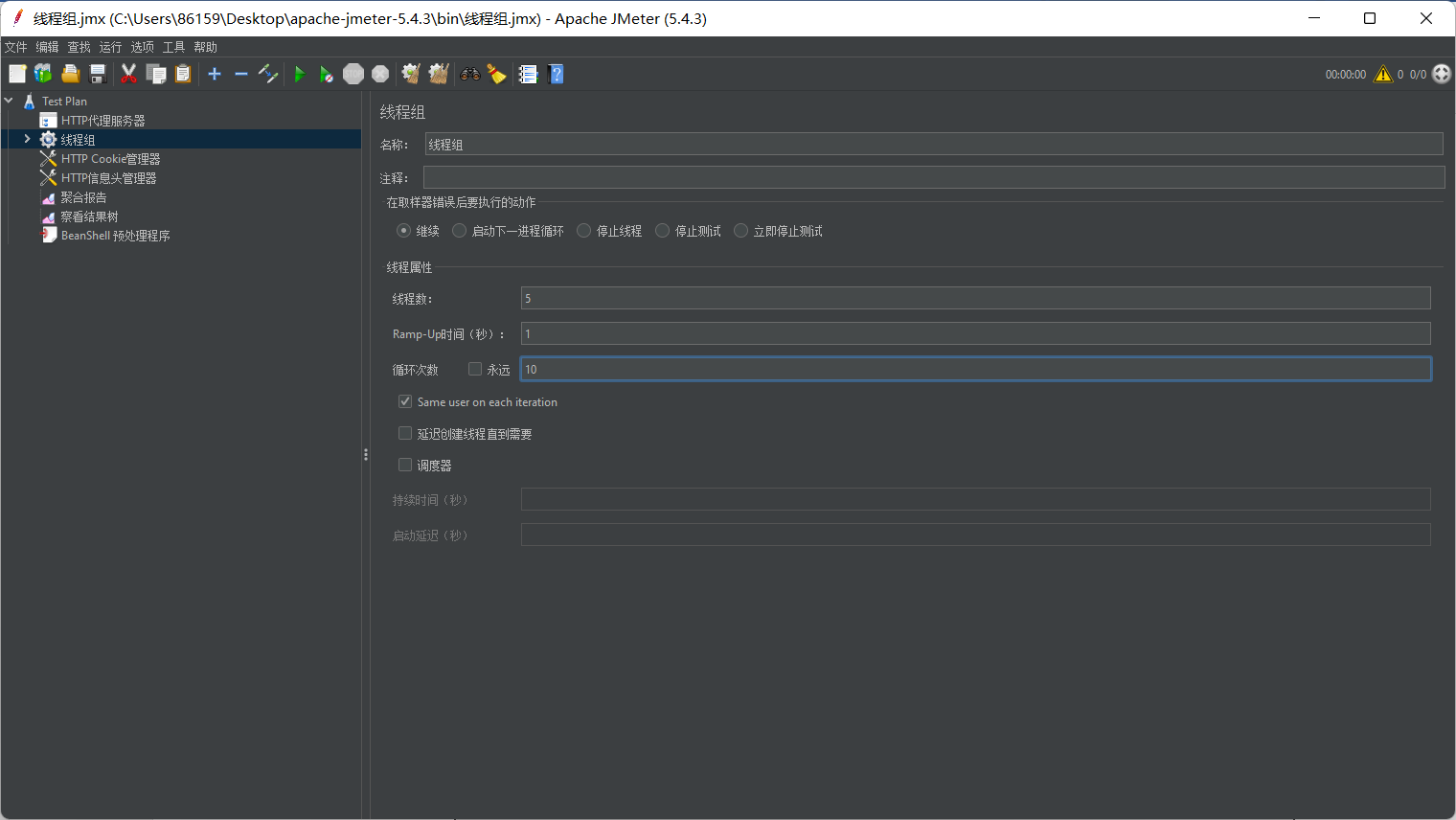
我们发现这里线程组中存储的每一条都是一个操作，有点类似于selenium，如下所示就是刚刚刚记录的我们填写订单个人信息的操作：



因此我们只要修改这里面操作的信息即可完成行为的修改

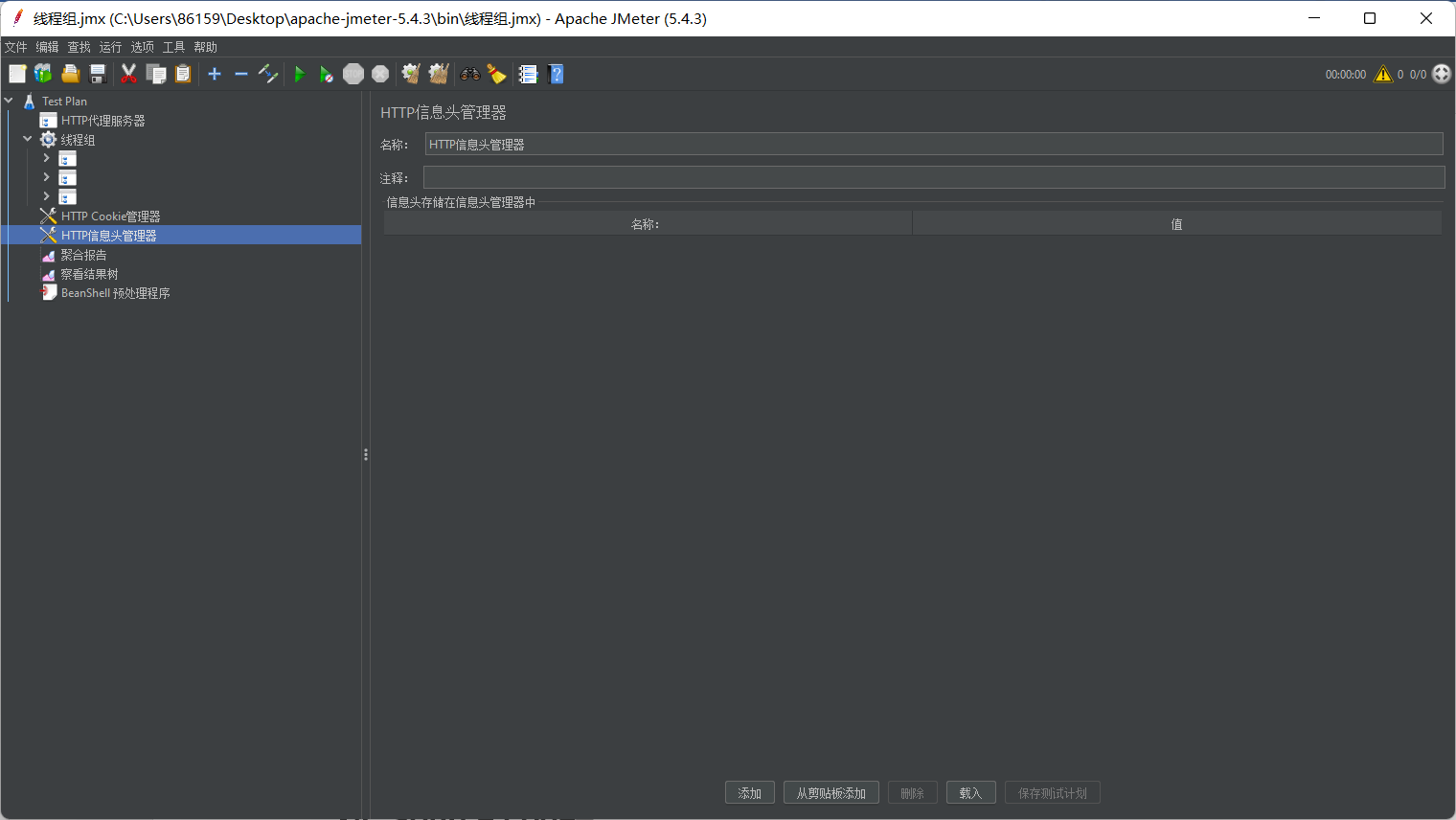
##### 2.7 压力测试

接下来我们尝试进行压力测试，其实就是选择http代理服务器要执行的次数，然后他将代替我们自动执行刚刚的记录操作来完成压力测试。首先我们尝试5\*10的压力测试，配置就是：

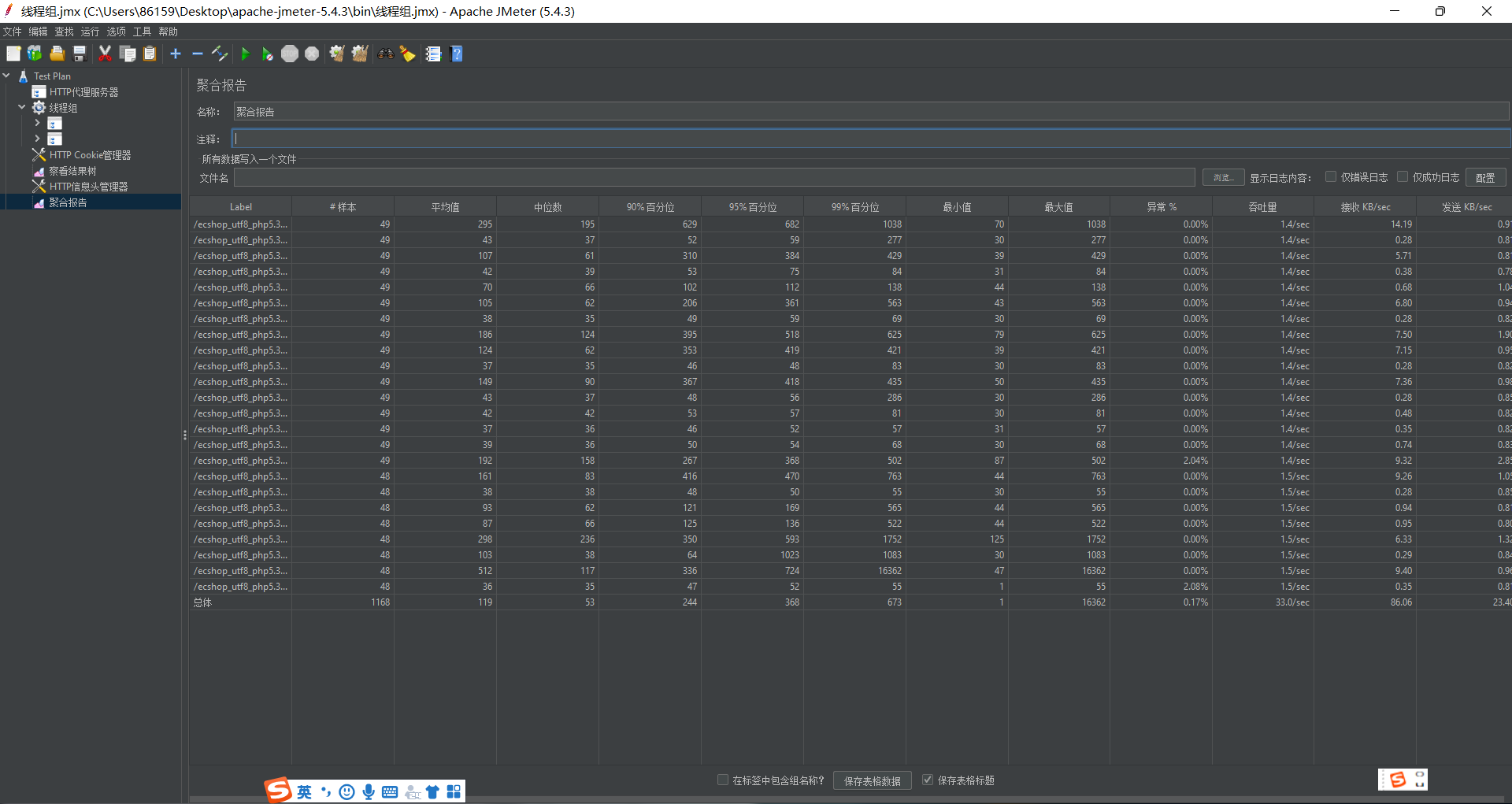


即每秒都执行5个线程，并且执行10次，然后点击上方的绿色三角既可以完成测试，为了方便我们一会查看系统的负载信息，我们可以现在运行前输入top以及sar -u 2 5查看一下压力测试前的负载情况。同时实验要求还要展示聚合报告，因此我们再创建一个聚合报告

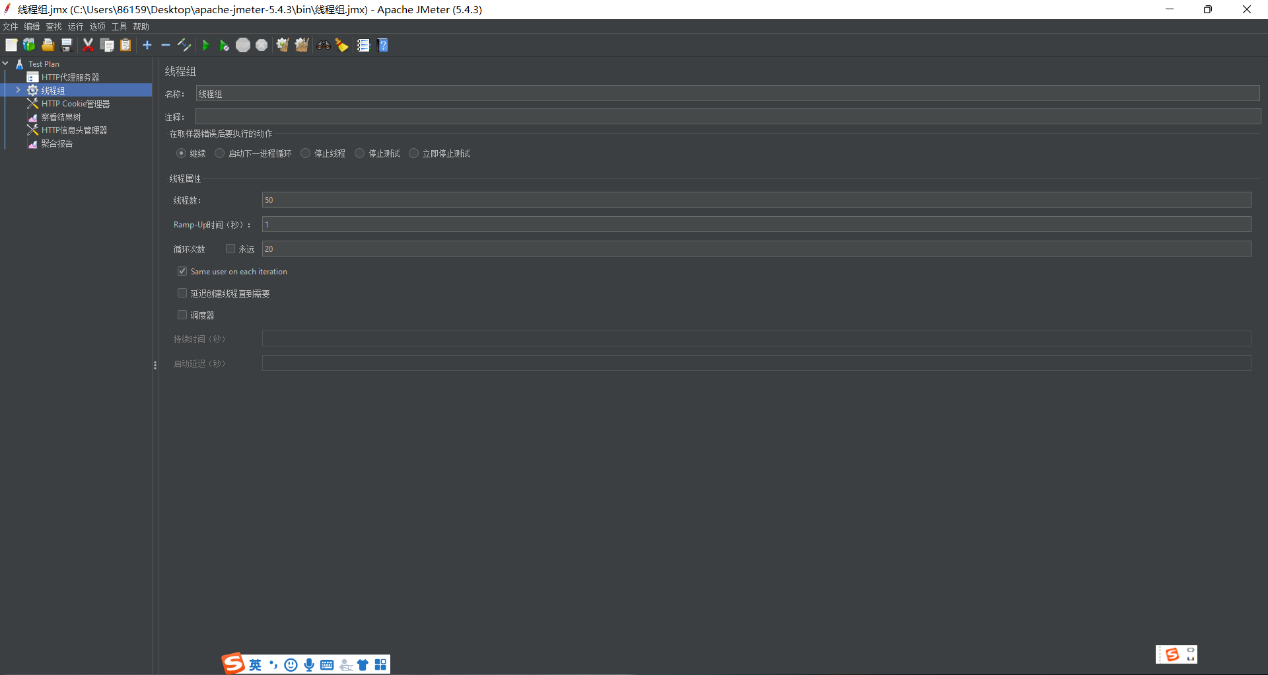
然后接下来我们尝试进行5\*10的压力测试，首先我们还需要在创建http信息头管理器和cookie管理器，因此现在testplan应该如下图所示：

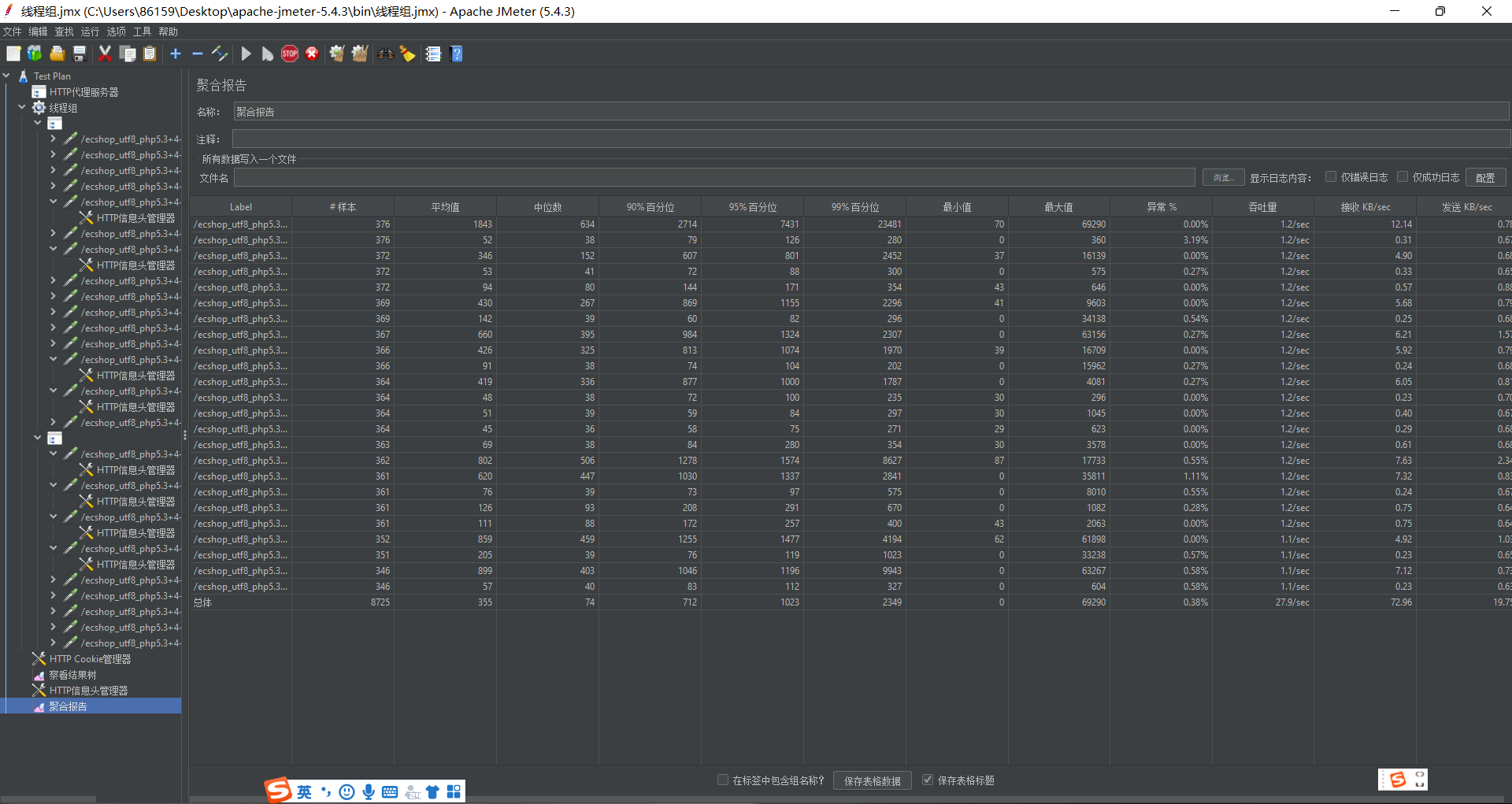


自此我们就完成了压力测试前的基本配置，开始进行压力测试，等待执行完成以后我们可以查看聚合报告如下所示：



然后我们在调整线程组配置改为50\*20再次进行压力测试聚合报告如下图所示：

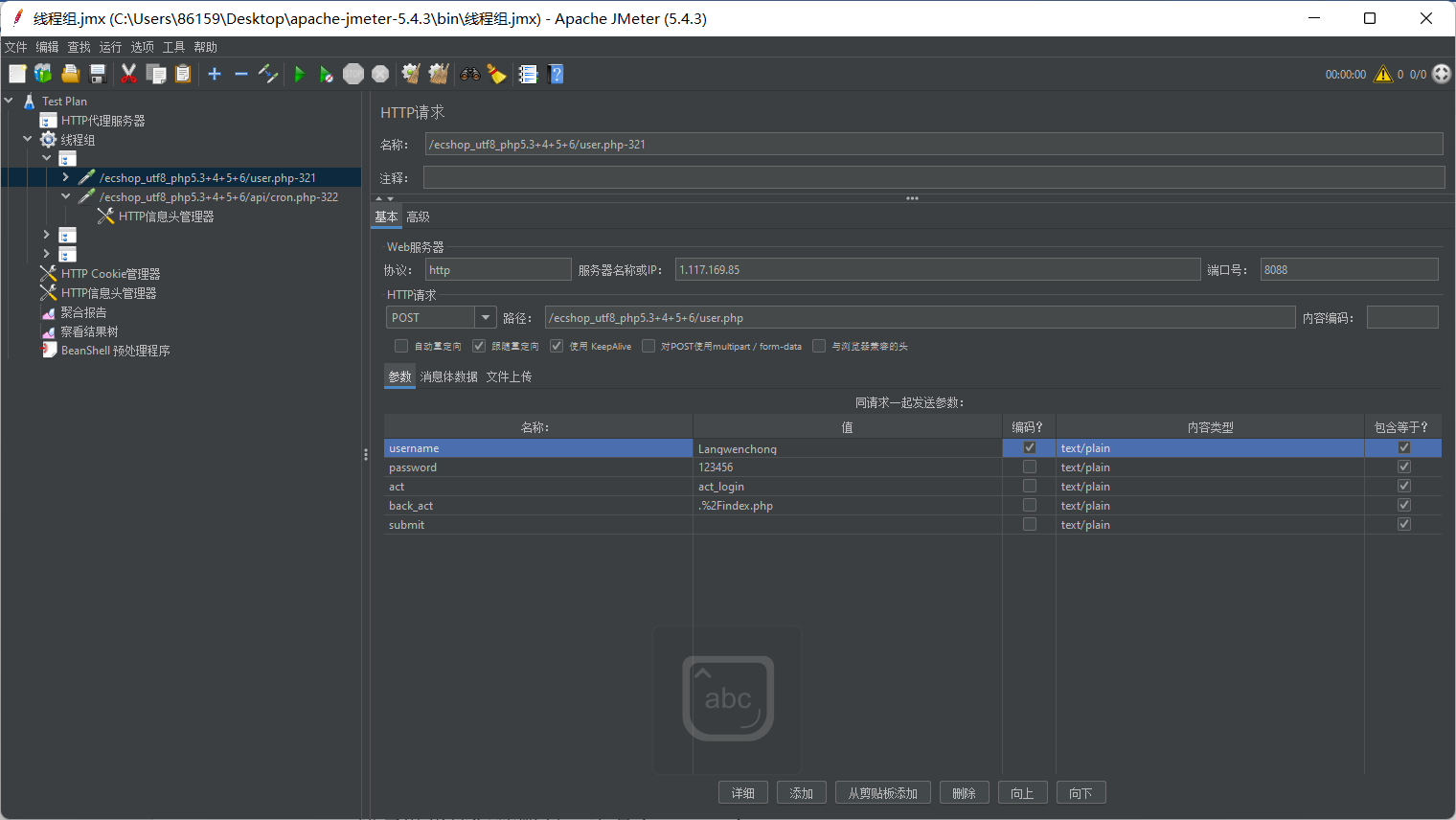




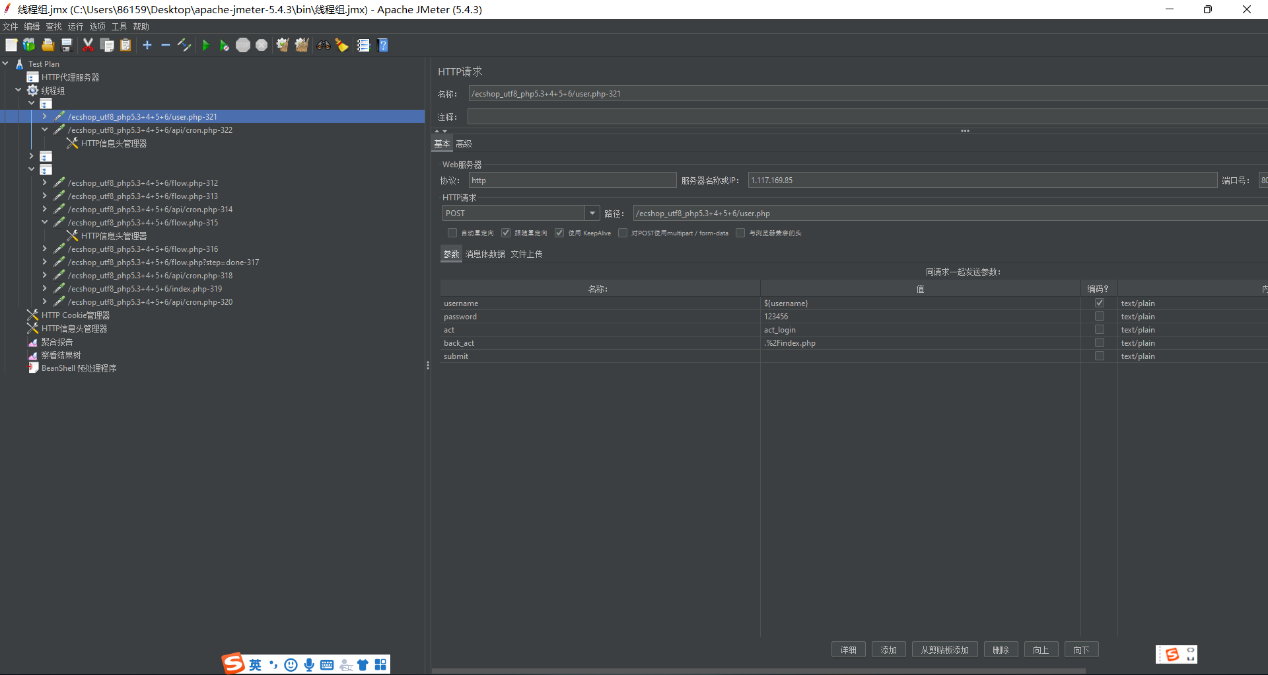
自此我们就完成了压力测试，可以看到当50^29时吞吐量变大， 同时异常率提高这可能是因为服务器不能及时进行多个请求的相应导致的，也有可能是后台出现了问题。然后我们在执行一下top以及sar -u 2 5记录一下此时的负载情况，具体的分析见第三部分。

##### 2.8 Beanshell编写代码自定义测试

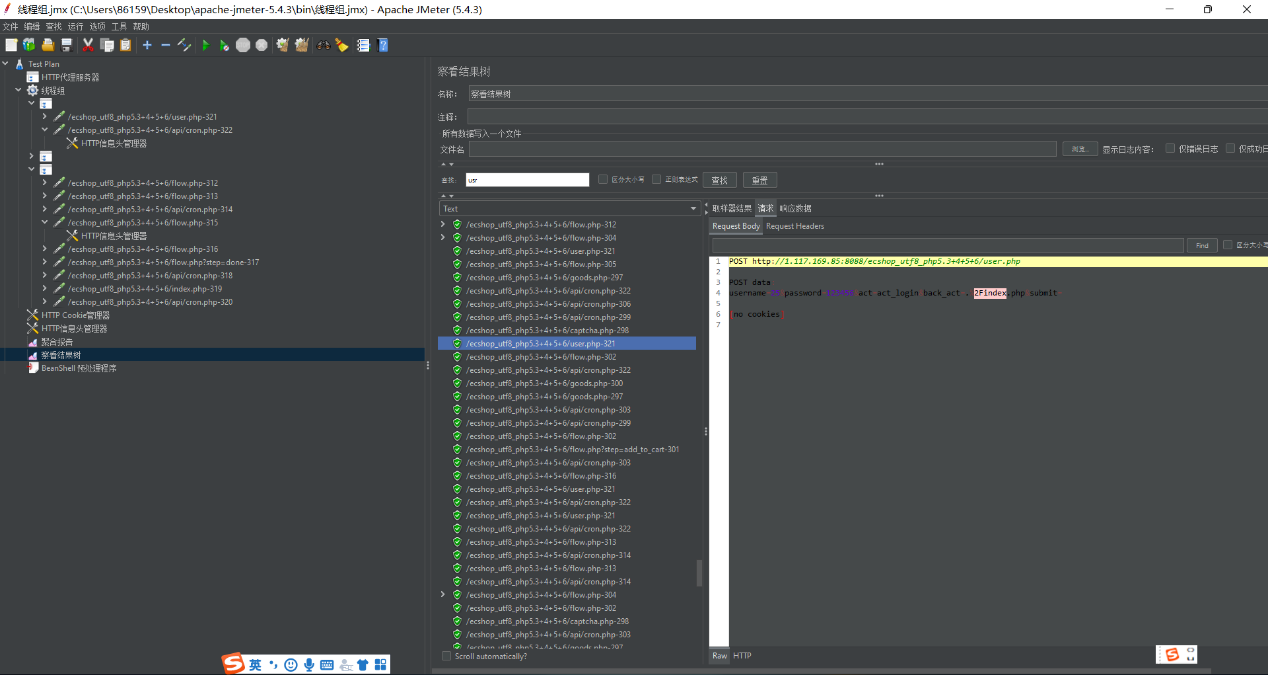
所谓beanshell待处理程序，其实是可以让我们自定义对某个操作进行测试前进行修改，比如登录接口的测试，如果我们按照上面的方法进行会发现一直都是填入的同一个数据来进行测试，这明显有时候不能满足我们的测试需求，比如现在我们希望可以让jmeter在进行测试前首先自动生成一下随机数据填充到将要测试的内容之中然后再执行压力测试，此时我们就会用到beanshell来对测试进行预处理。首先我们需要创建一个beanshell预处理程序，其实里面就可以直接编写我们要进行的java代码即可，甚至无需配置包路径等内容，非常简单易上手，由于我这里是对登录进行测试，因此我先又录制了一个进行登录的操作可以查看到这里我填写的账号一直是Langwenchong

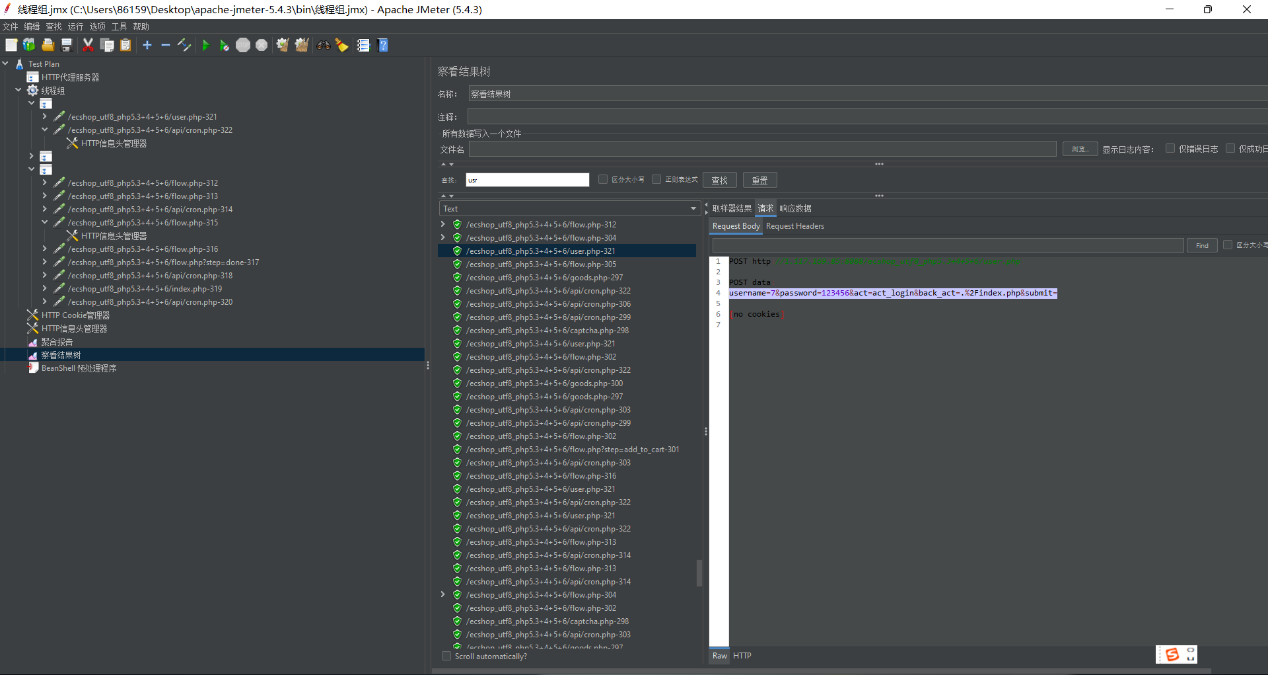


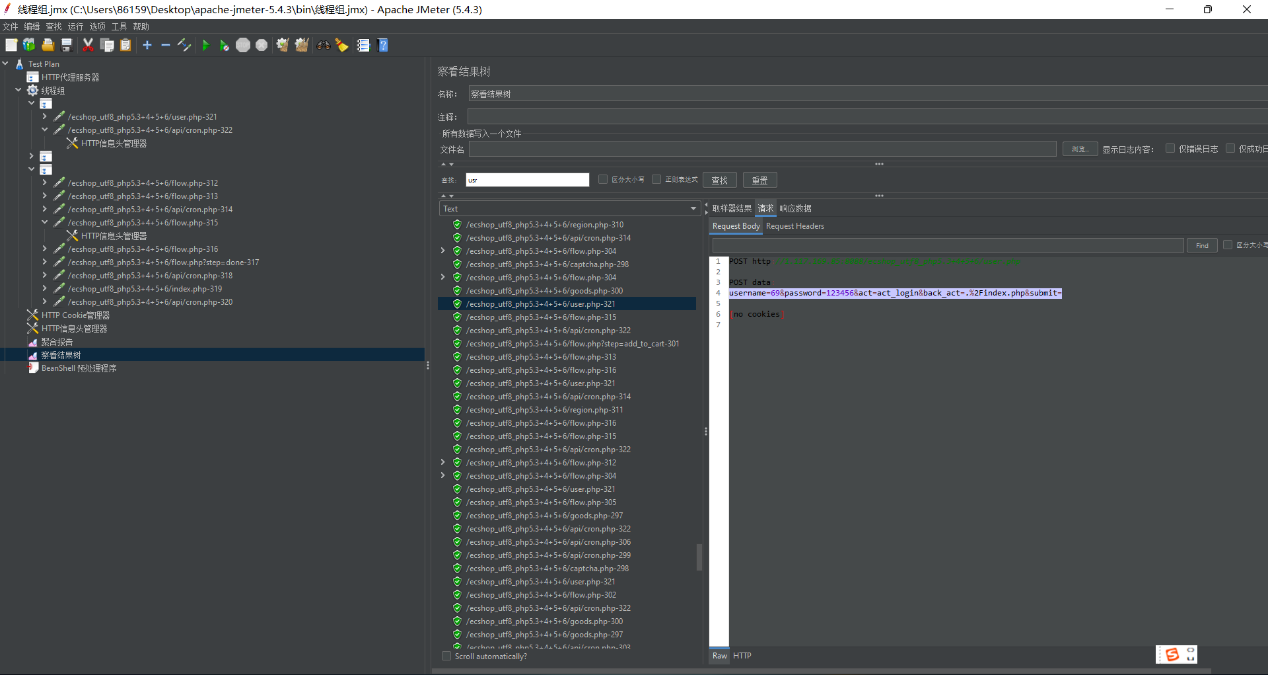
现在我们在beanshell中编写一个java程序调用random来随机生成一个随机数作为账号名填充到操作中，具体的代码见第四部分。我们将要填充的随机数账号存储到了变量名username中，然后修改这个接口的变量名为${username}



然后为了方便我们一会查看压力测试时填充的账号是否为我们生成的随机数，我们在创建一个察看结果树。创建完成以后我们再次启动压力测试，执行完成以后我们可以从查看结果中查看到每一个测试操作中填充的账号都变成了我们生成的随机数，这就是beanshell预处理程序的功能：



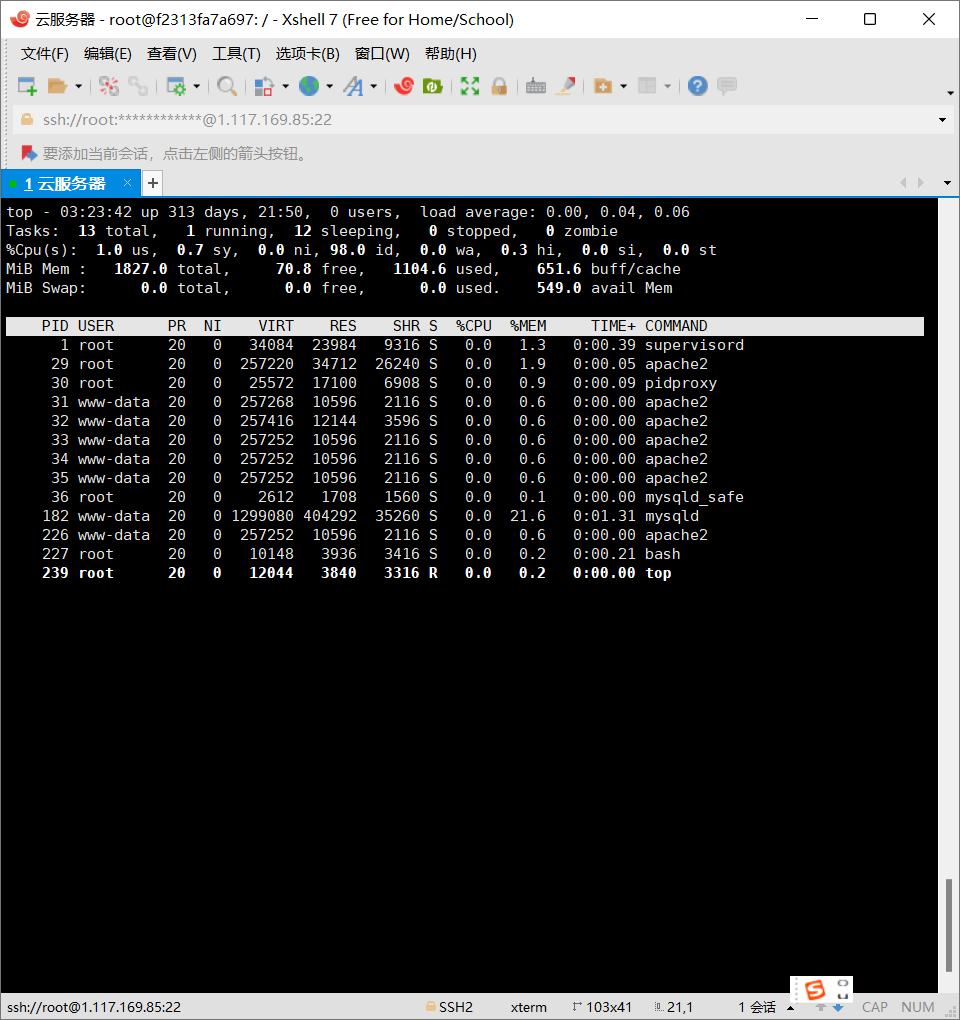




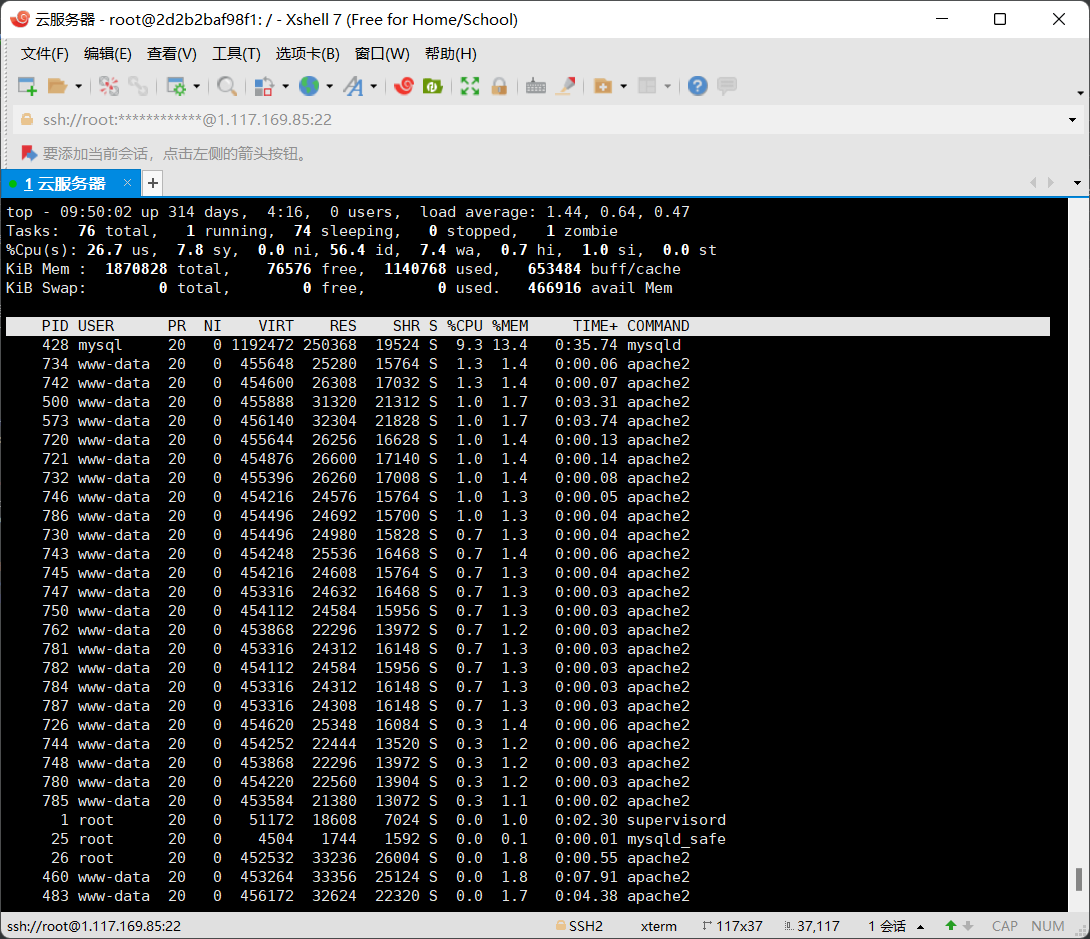
#### Result analysis

完成了压力测试以后我们再来分析一下压力测试前后的负载情况，首先给出压力测试前后的top指令分析结果:

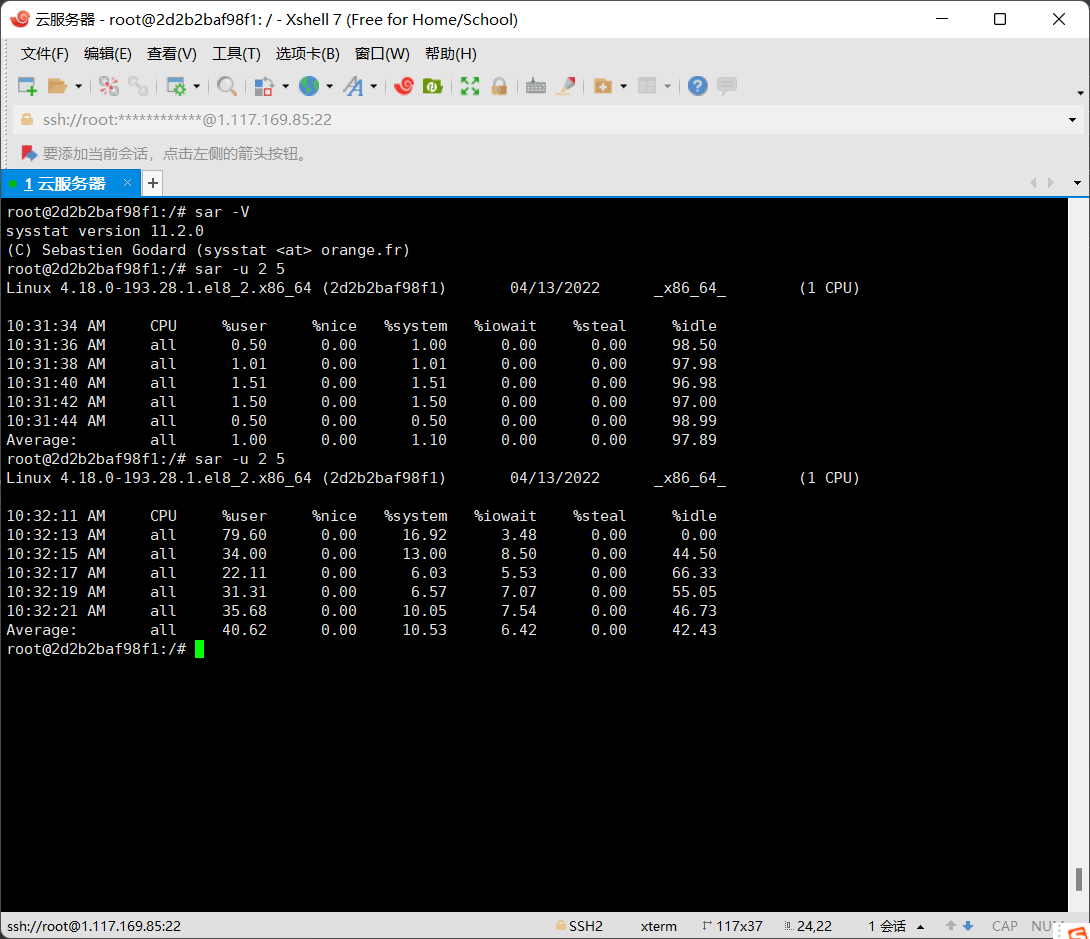
压力测试前：



压力测试后：



我们可以明显看到服务器的运行程序增多，同时cpu利用率增大，说明压力测试以后服务器负载情况增大。同时我们也可以通过sysstat的前后对比图进一步看出：



可以看到iowai，system等指令都明显增大，因此jmeter可以有效的对部署项目的服务器进行压力测试。

#### Source code

import java.util.Random;

//随机设置用户名

random = new Random(System.currentTimeMillis());

int r1 = random.nextInt(100);

String username = Integer.toString(r1);

//将username保存到jmeter中

vars.put("username", username);

//查看用户名

String getUsername = vars.get("username");

System.out.println("User's name is " + getUsername);

#### 实验心得

本次实验花费的时间很长，主要是环境配置，项目部署以及熟悉jmeter的工作方法都需要很多时间与精力的投入。对于LAMP环境的配置我一开始选择的是自己配置，但是由于版本每个组件的版本兼容问题，最终我放弃了选择更轻松的docker容器的方法进行环境搭建，然后ecsshop也是第一次部署的php程序中途出现了一些配置数据库的问题，最终还是完美解决。而对于jmeter则是主要学习记录操作，线程组压力测试，同时中途发现需要使用http cookie管理和http信息头管理以及本地服务器代理的配置以后才能正常进行压力测试。同时由于使用了容器，而这个lamp容器centos是阉割版，没有yum因此我又进一步安装了yum才完成了最后的sysstat的安装以及性能分析。经过本次实验我对jmeter的使用以及压力测试的具体原理和beanshell预处理自定义测试有了深入的理解，相信对未来自己的软件测试有着巨大的帮助。