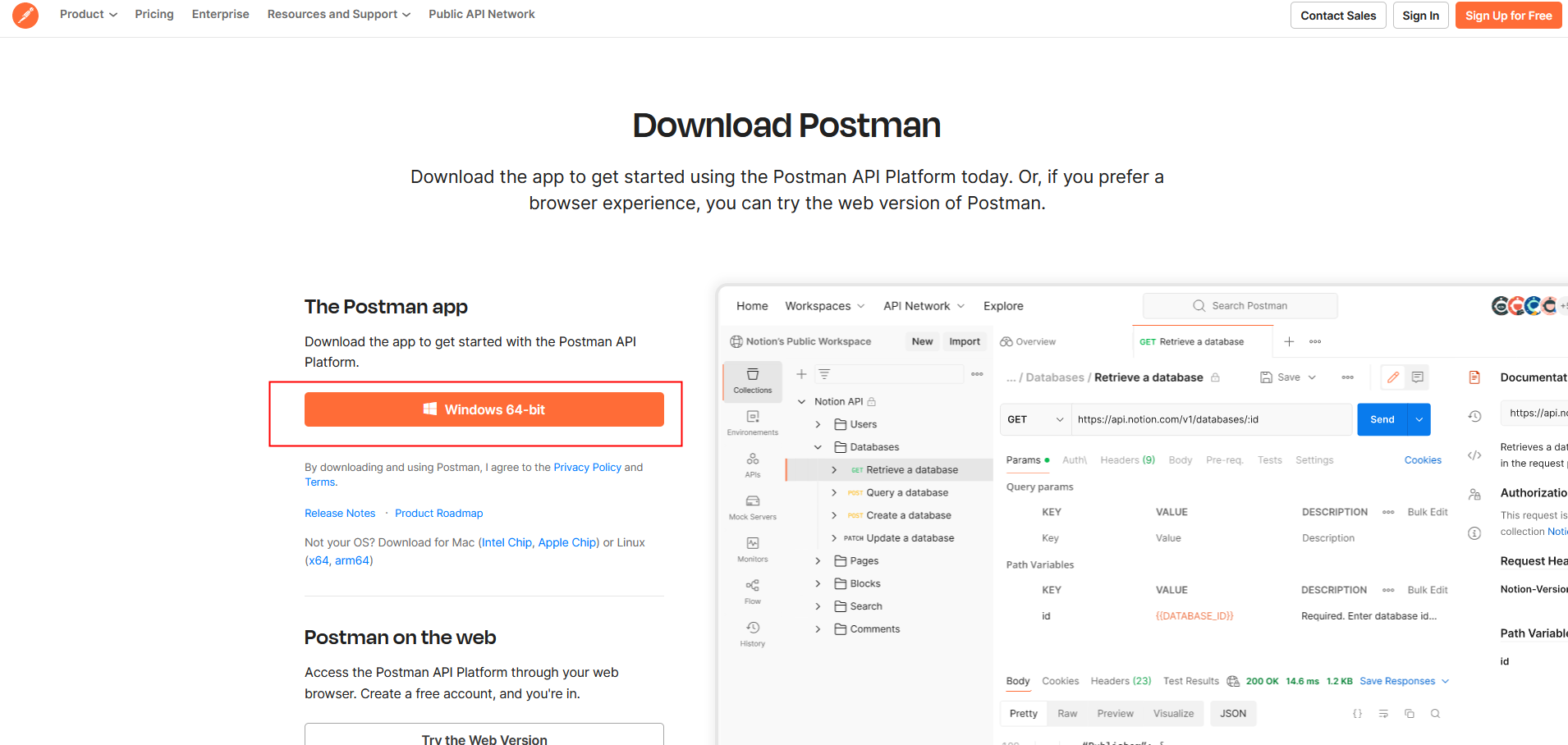
# **POSTMAN接口测试工具应用**

一、安装

登录官网，下载对应安装包

<https://www.postman.com/downloads/>

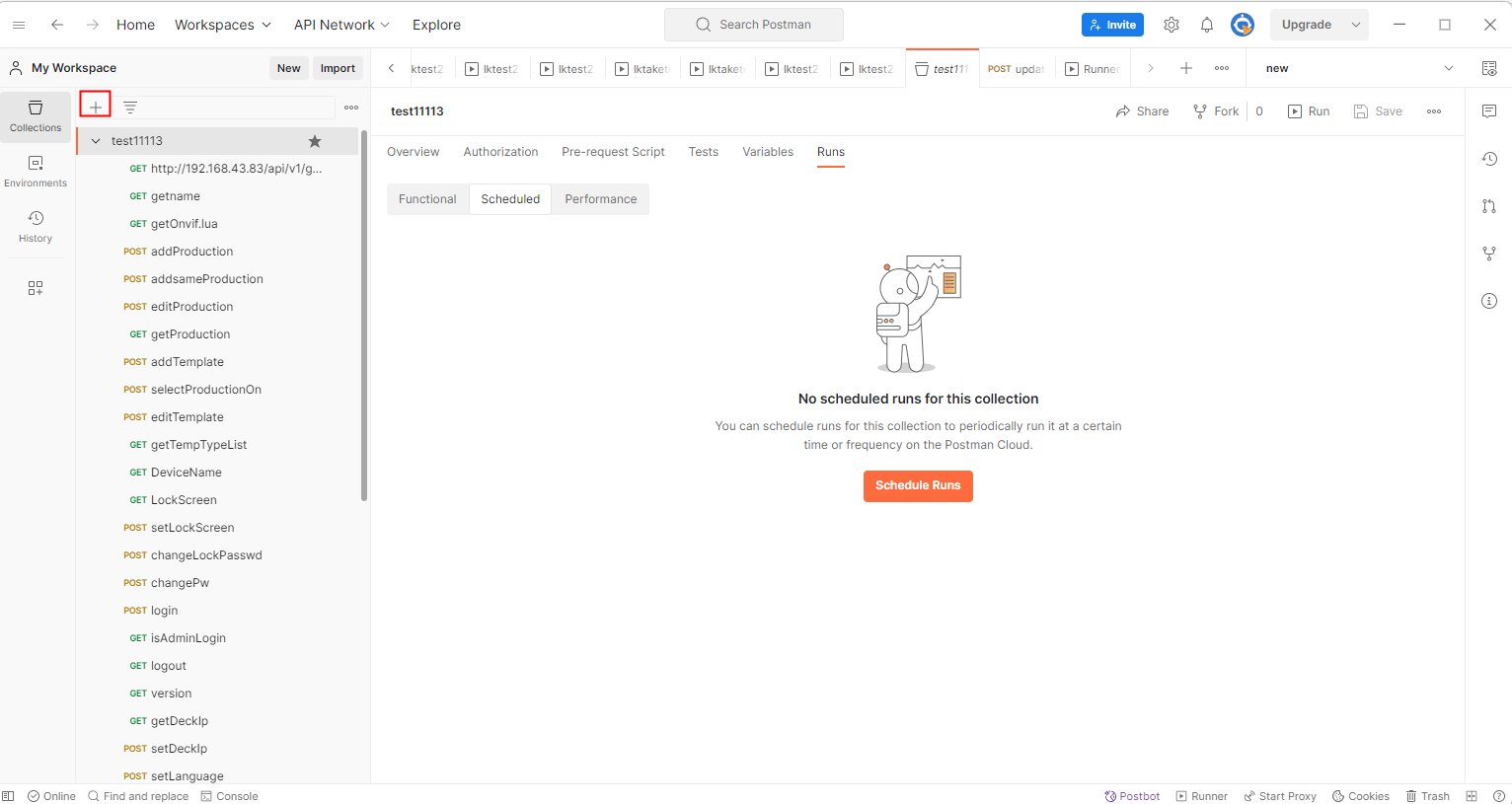


安装步骤略

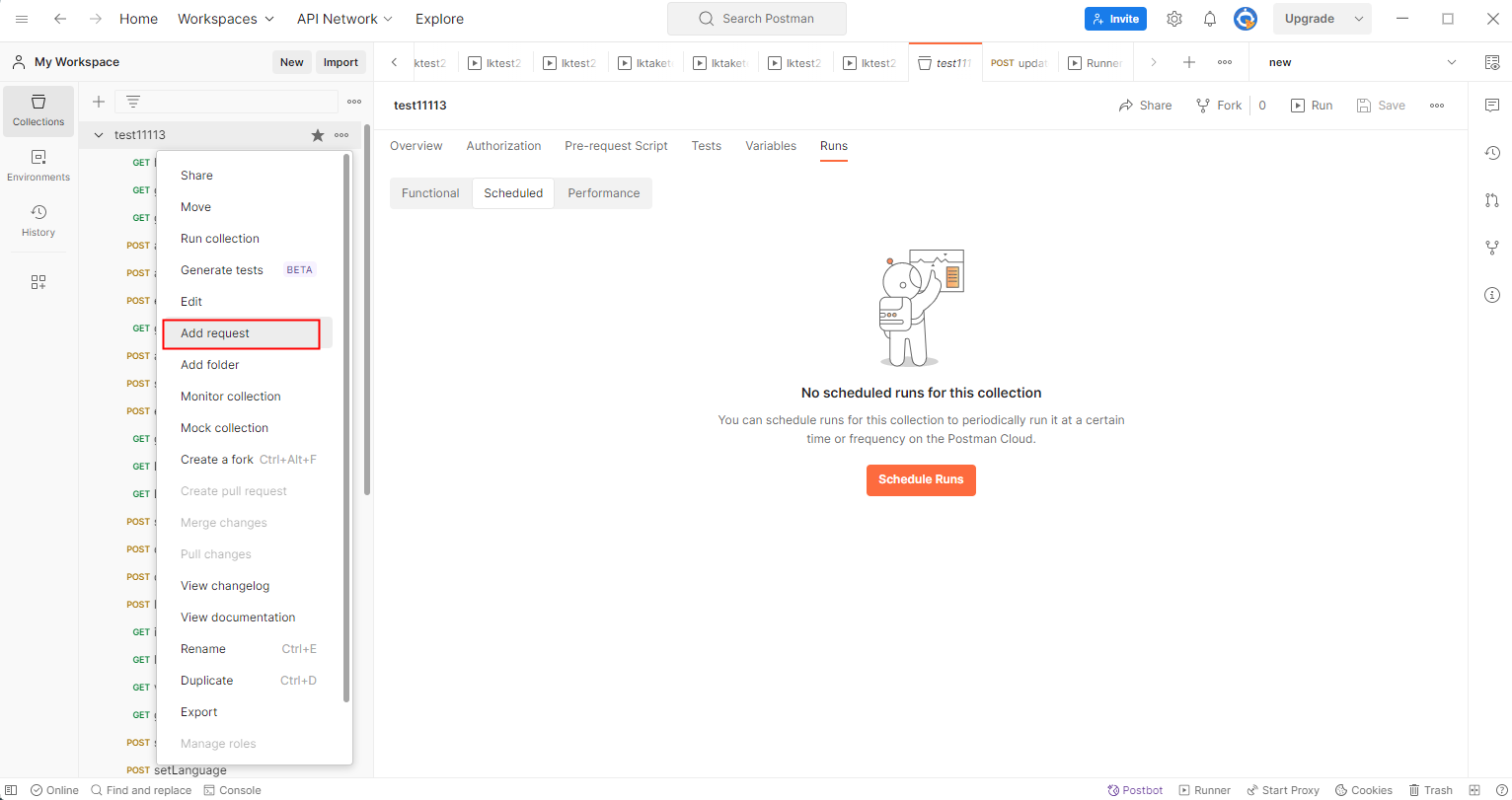
二、api接口自动化测试应用

接口自动化测试的过程是将所有待测试的接口放到一个测试集里，自动全部执行，并对接口的返回值、返回状态、返回类容等进行校验，最后生成测试报告

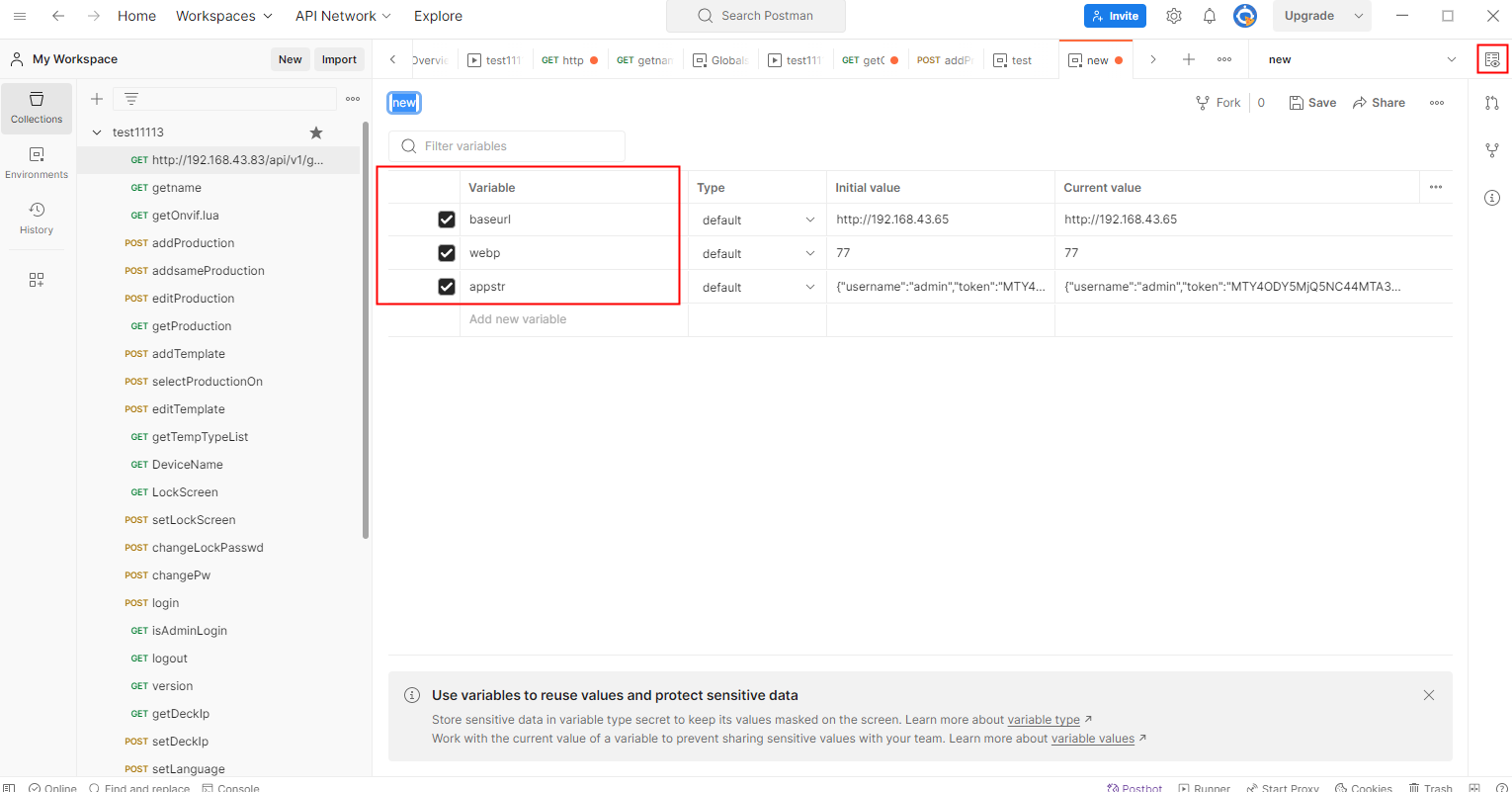
1）创建一个测试集



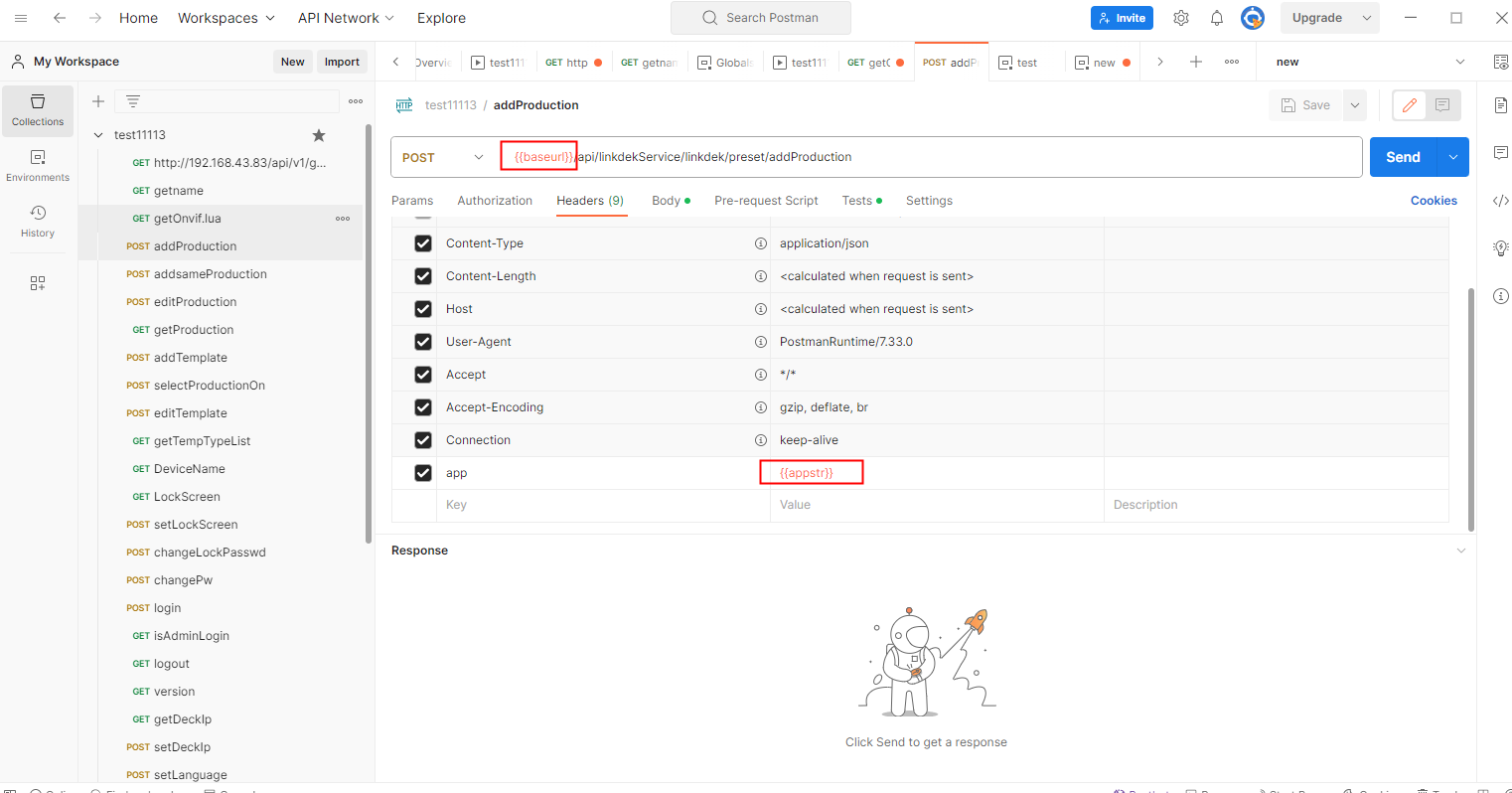
2）在测试集合里添加待测试的接口



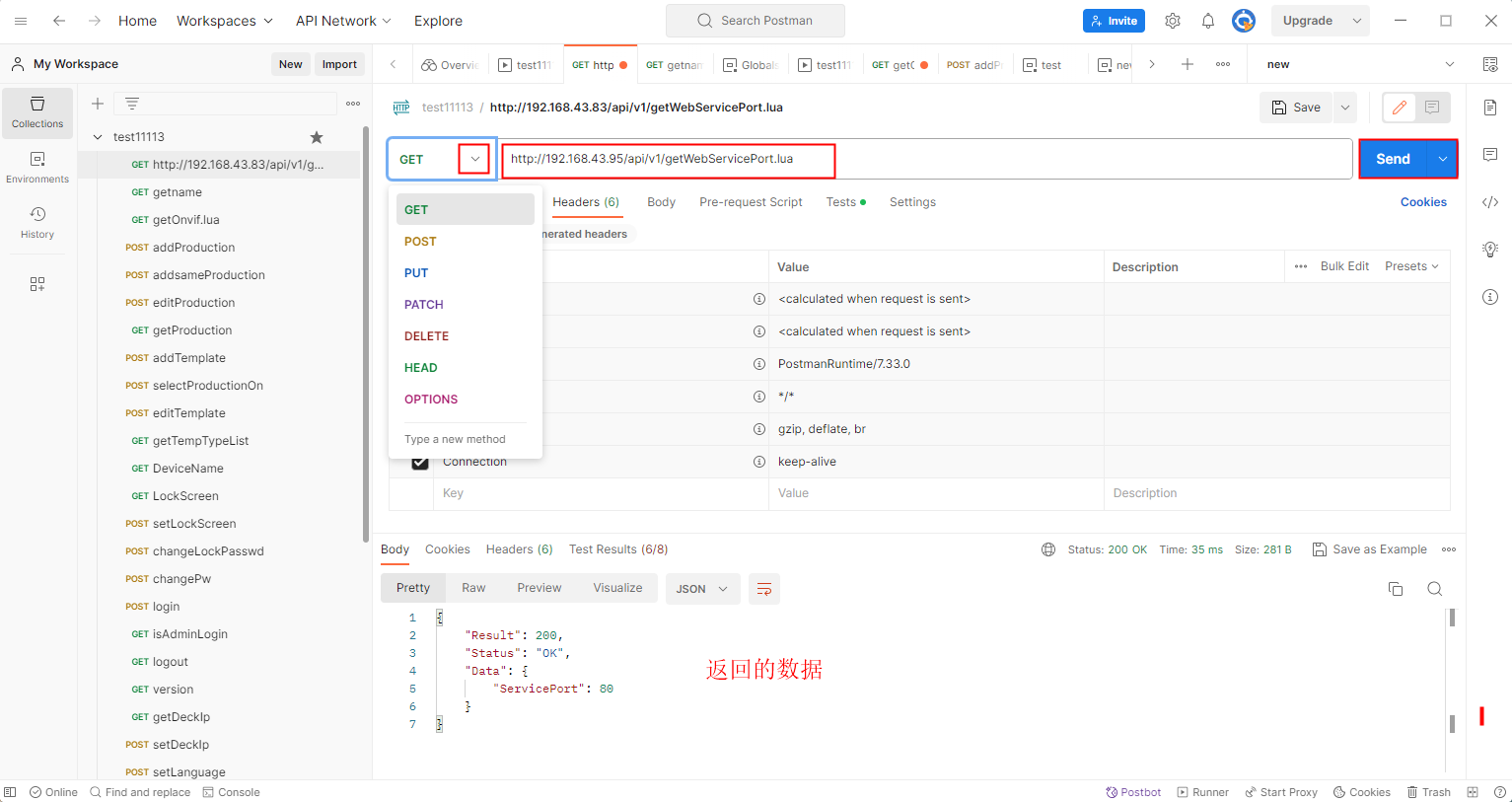
3)接口调用时，url和请求头的cookie、token等值是一次测试中是固定的，可以在postman中创建环境变量，统一调用:点击右上角的environment quick look，选择edit



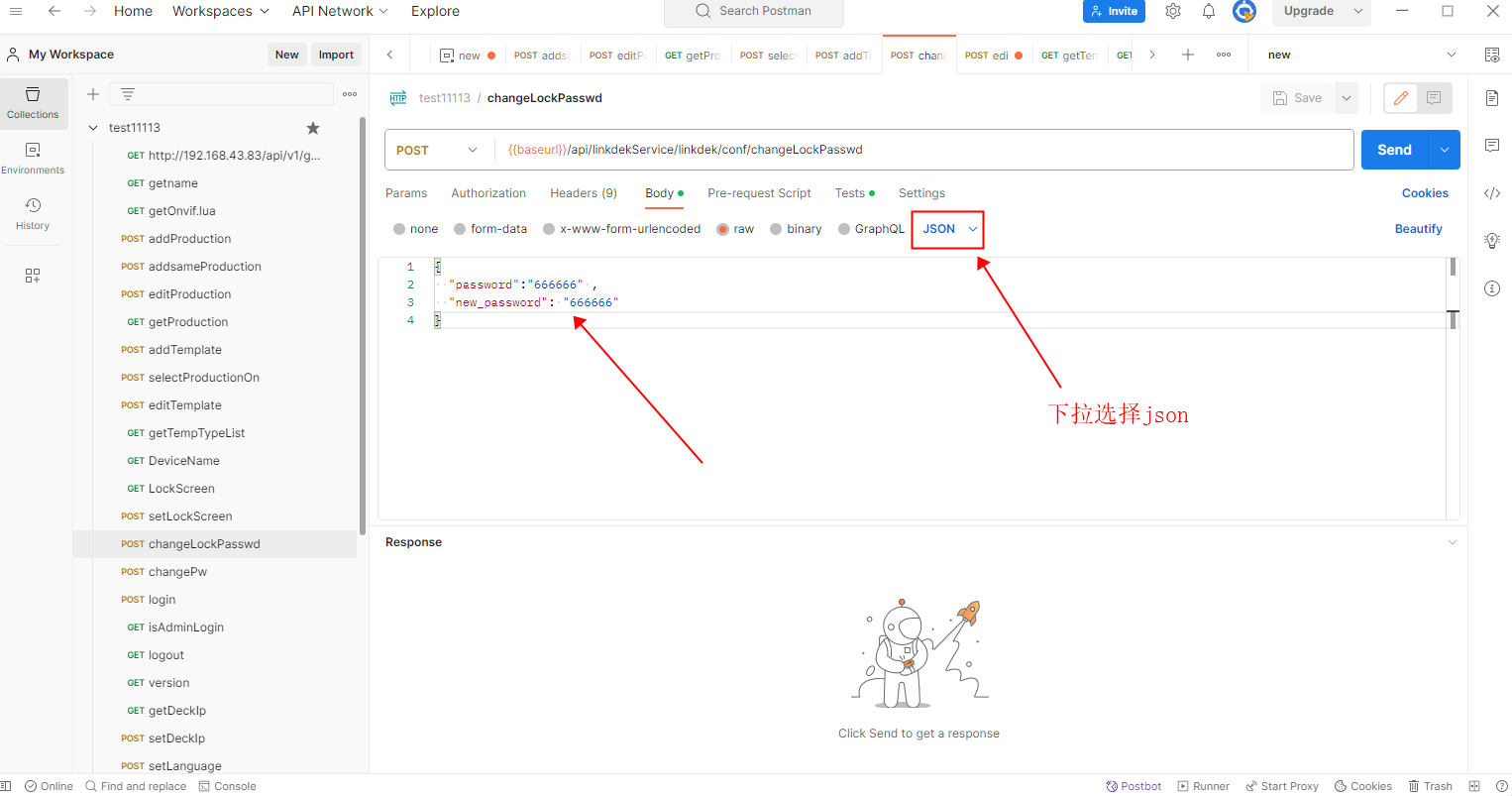
创建的环境变量在调用时使用{{}}



4)根据不同的接口选择get/post方法，地址栏填入待测试的接口，点击send单独测试该接口，返回的数据在下方显示



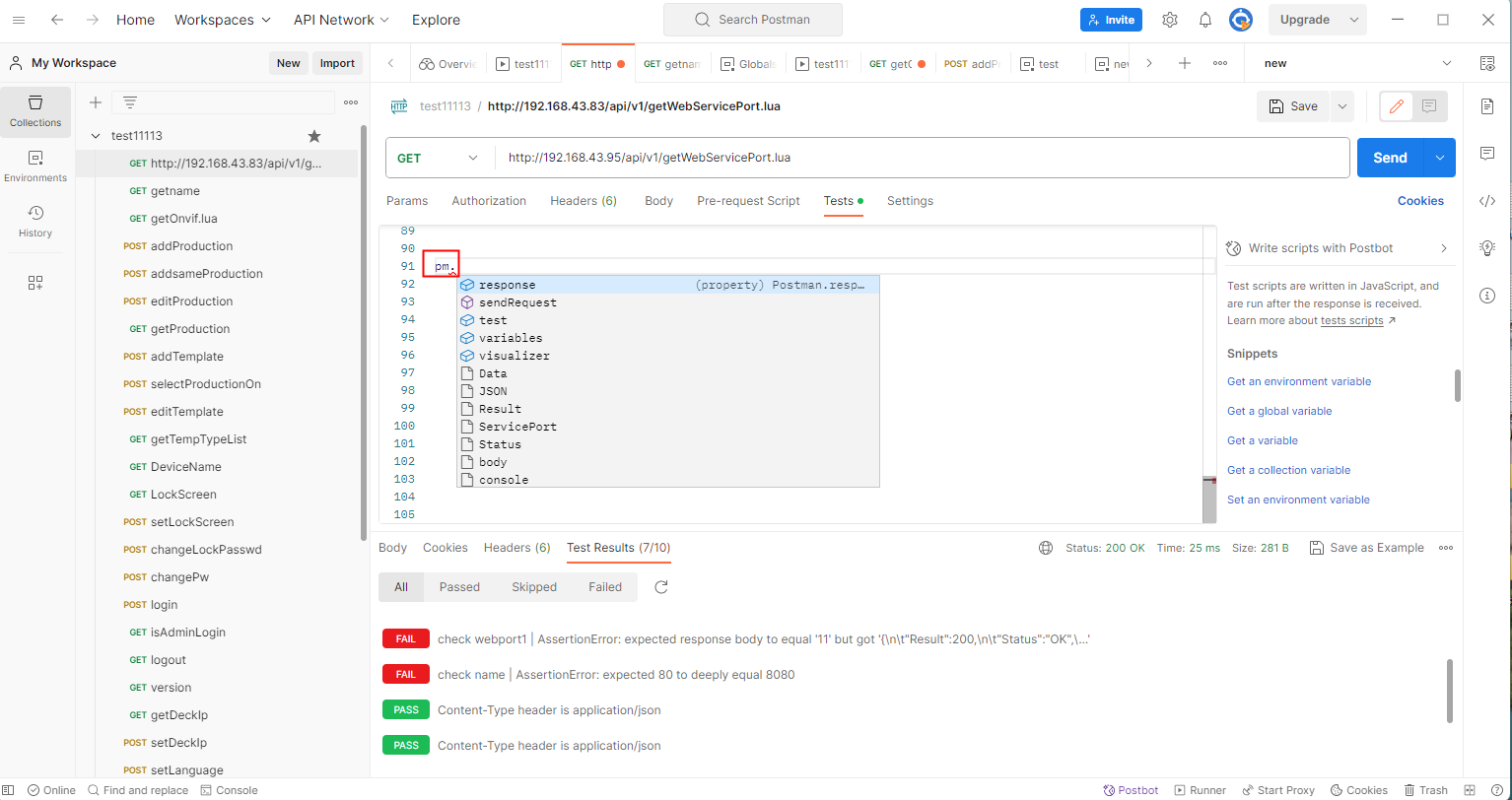
post接口需要在请求body里传入请求的json格式数据



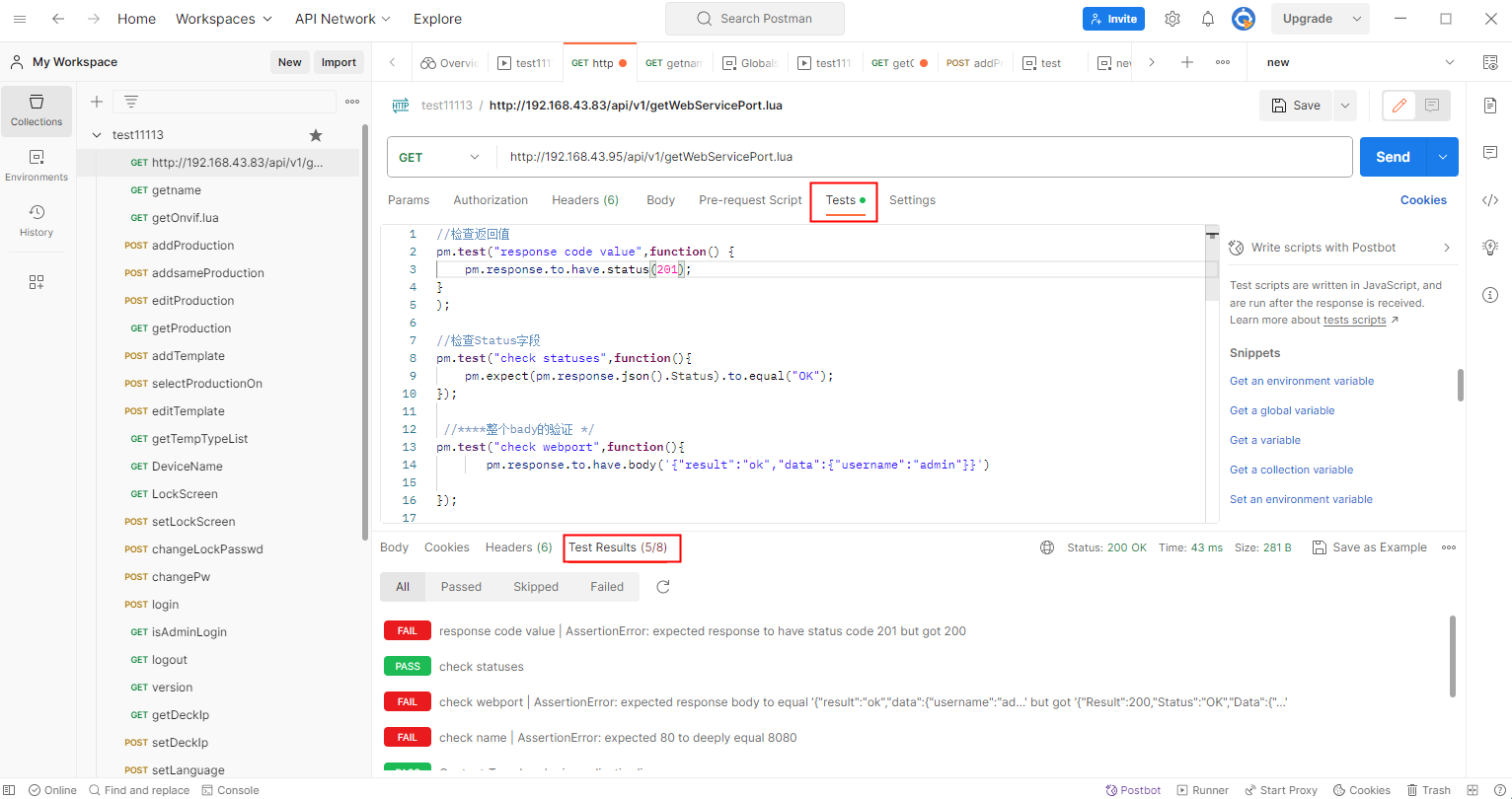
5)返回值校验，利用pm对象自带的一些方法对返回值、返回状态、返回类容或者返回类容中的某一个字段进行校验

javascript脚本添加在test中

pm对象



添加脚本至Tests模块下



返回值校验的javascript脚本:

pm.test("response code value",function() {  
 pm.response.to.have.status(200);  
});

pm.response.to.have.status()中填入期望的返回值，返回正确为pass，返回错误为fail

pm.test("check statuses",function(){  
 pm.expect(pm.response.json().Status).to.equal("OK");   
});

pm.expect(pm.response.json().Status).to.equal()中填入期望的返回状态，返回正确为pass，返回错误为fail

var jsonData = pm.response.json();  
  
// 将JSON数据转换为字符串形式  
var responseBody = JSON.stringify(jsonData);  
  
// 去除字符串中的空格和换行符  
responseBody = responseBody.replace(/\s+/g, "");  
  
// 定义期望的响应体  
var expectedResponse = '{"Result":200,"Status":"OK","Data":{"ServicePort":80}}';  
  
// 比较响应体  
pm.test("响应体应该等于期望的响应体", function () {  
 pm.expect(responseBody).to.equal(expectedResponse);  
});

\*根据不同的接口，修改定义的期望体

\*响应体是json格式时，需要手动将json压缩成一行，再填入期望的响应体



转格式网址为<https://www.sojson.com/yasuoyihang.html>

pm.test("check name",function(){   
 //check name为测试项，自定义名称  
 pm.expect(pm.response.json().Data['ServicePort']).eql(8080);  
});

例如响应体为:

{

"Result":200,

"Status":"OK",

"Data":{

"ServicePort":80 }

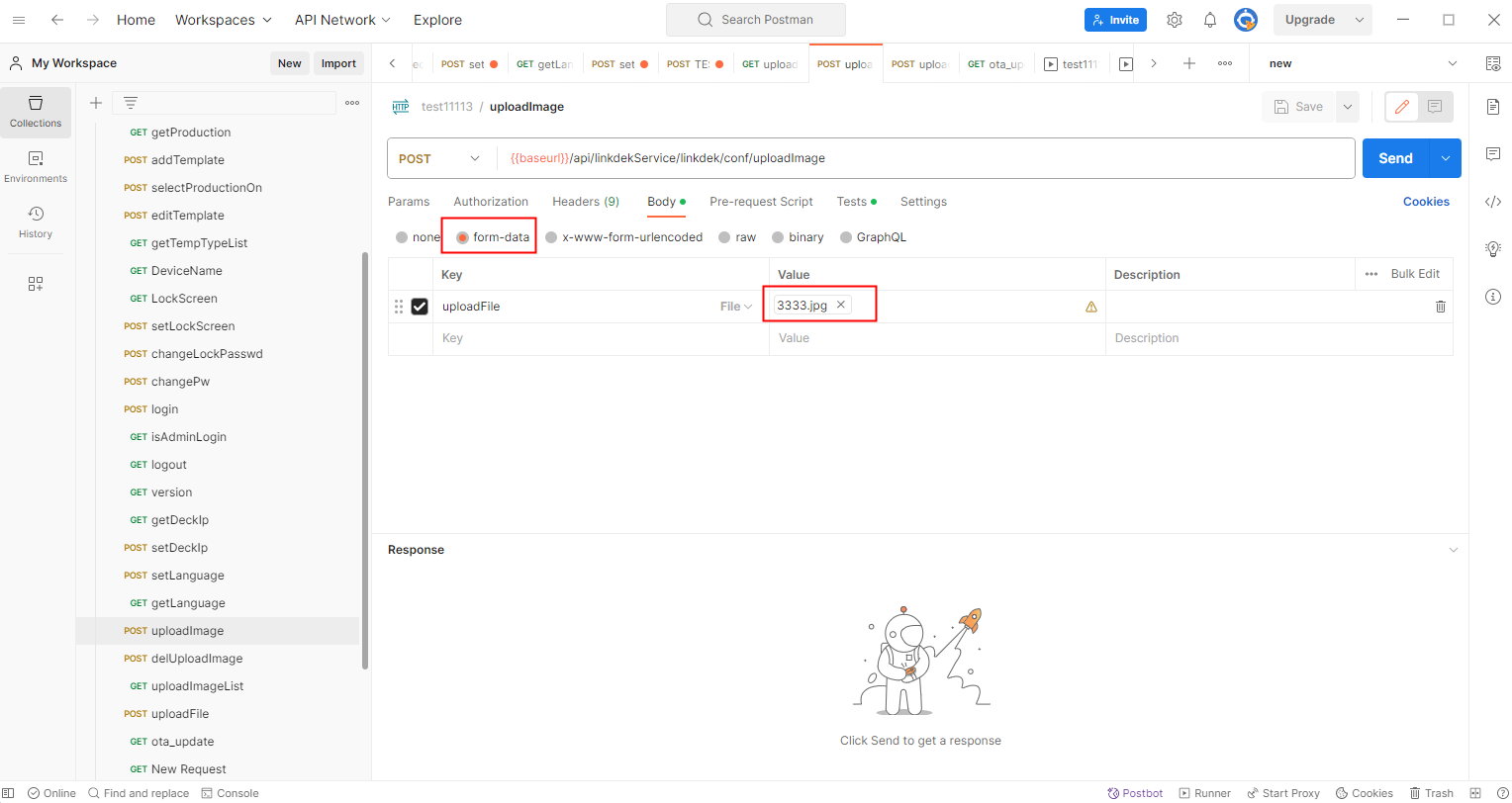
}

单独校验Data下 ServicePort的返回值

pm.expect(pm.response.json().Data['ServicePort']).eql()填入期待返回的ServicePort值

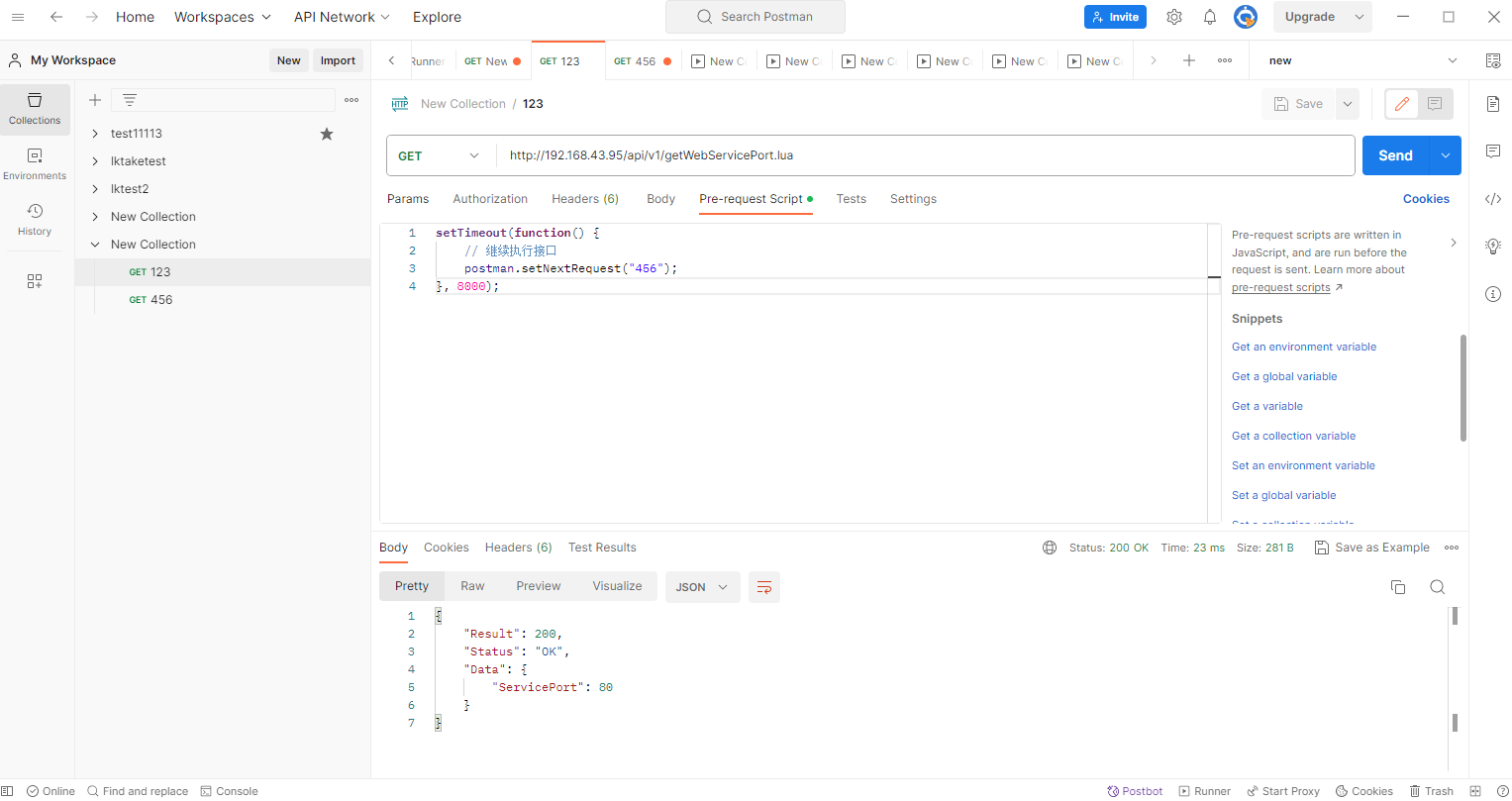
6)其他一些特殊情况的处理:

\*上传图片、上传固件等，请求体格式不再是json, body格式选择form-data，再上传待测试的图片或者固件



\*在执行完一个接口设置一个等待时间再执行下一个接口，可以在Pre-request-script模块中添加等待

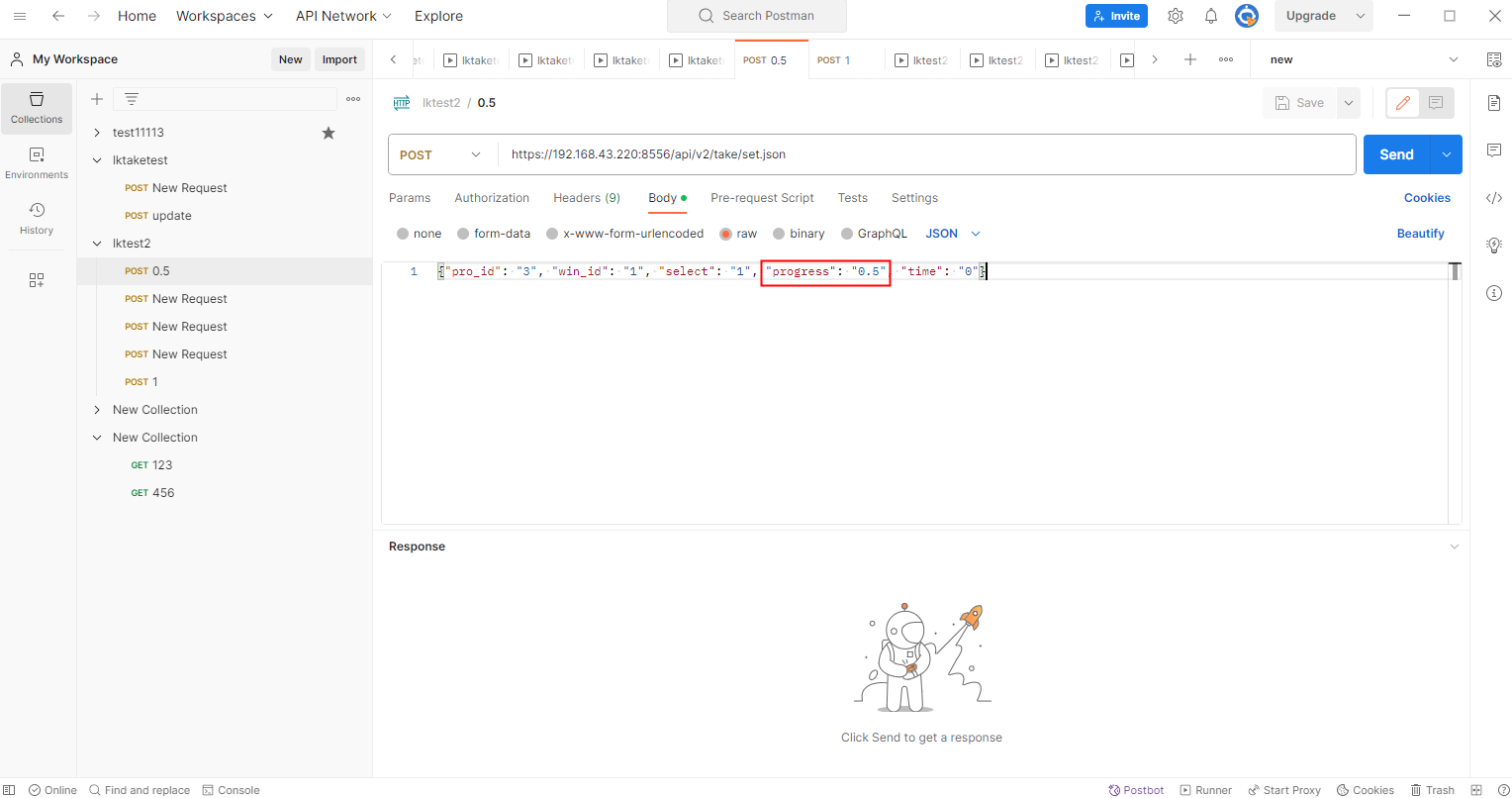
setTimeout(function() {  
 // 继续执行接口  
 postman.setNextRequest("456");  
}, 8000);



123接口执行完以后，等待8秒执行下一个接口456

\*相同的接口请求在短时间内传入不同的参数，可以对请求参数参数化

比如takedeck的切换接口，在完成依一次画面切换时，1秒内同时发出了10个左右的请求，每次请求的progress值不一样，范围是0~1

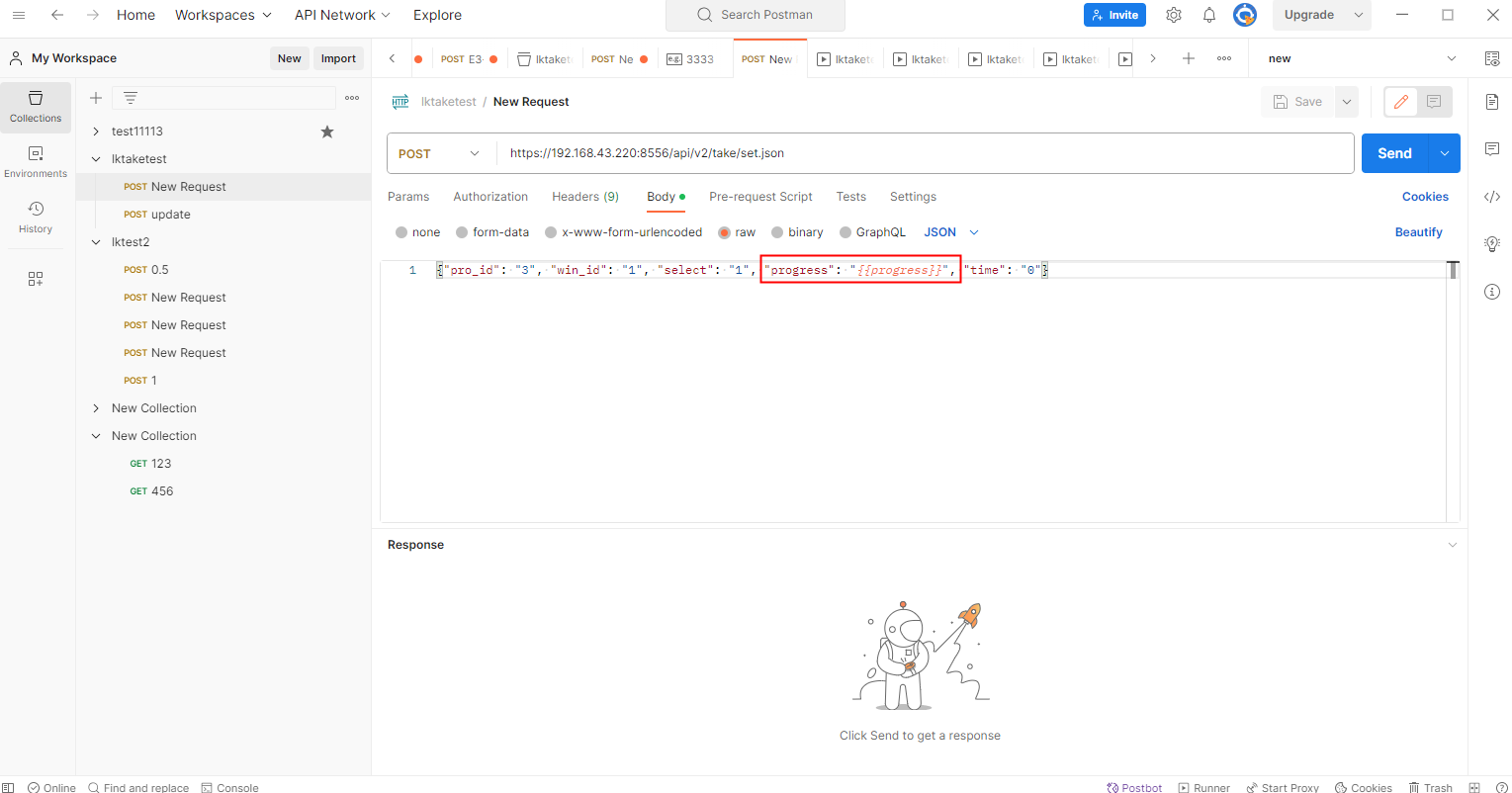


用接口模拟一次take切换画面，可以在一个接口中参数化progress的值，在本地json文件中读取值再进行调用

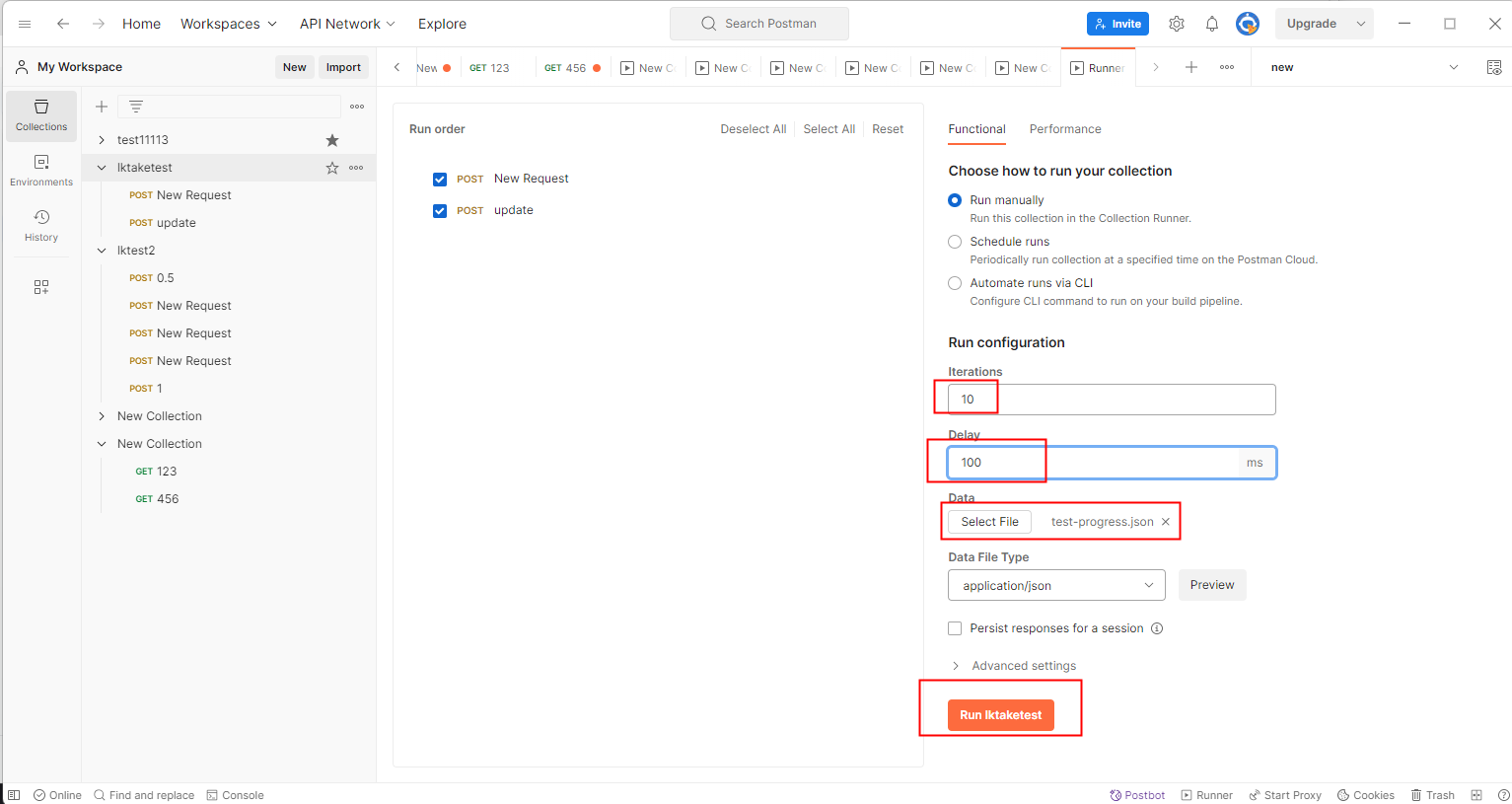
在本地创建一个test-progress.json文件，输入以下值并保存

[  
 {"progress": "0.1"},  
 {"progress": "0.2"},  
 {"progress": "0.3"},  
 {"progress": "0.4"},  
 {"progress": "0.5"},  
 {"progress": "0.6"},  
 {"progress": "0.7"},  
 {"progress": "0.8"},  
 {"progress": "0.9"},  
 {"progress": "1"}  
]

接口的body中，progress的值用{{progress}}替代



在运行该接口时，选中test-progress.json文件，interations次数与文件中progress行数一致，超过行数，将会一直执行最后一行到结束，delay按实际需求设置，运行时，该接口将依次逐步调用文件中的progres值进行接口测试



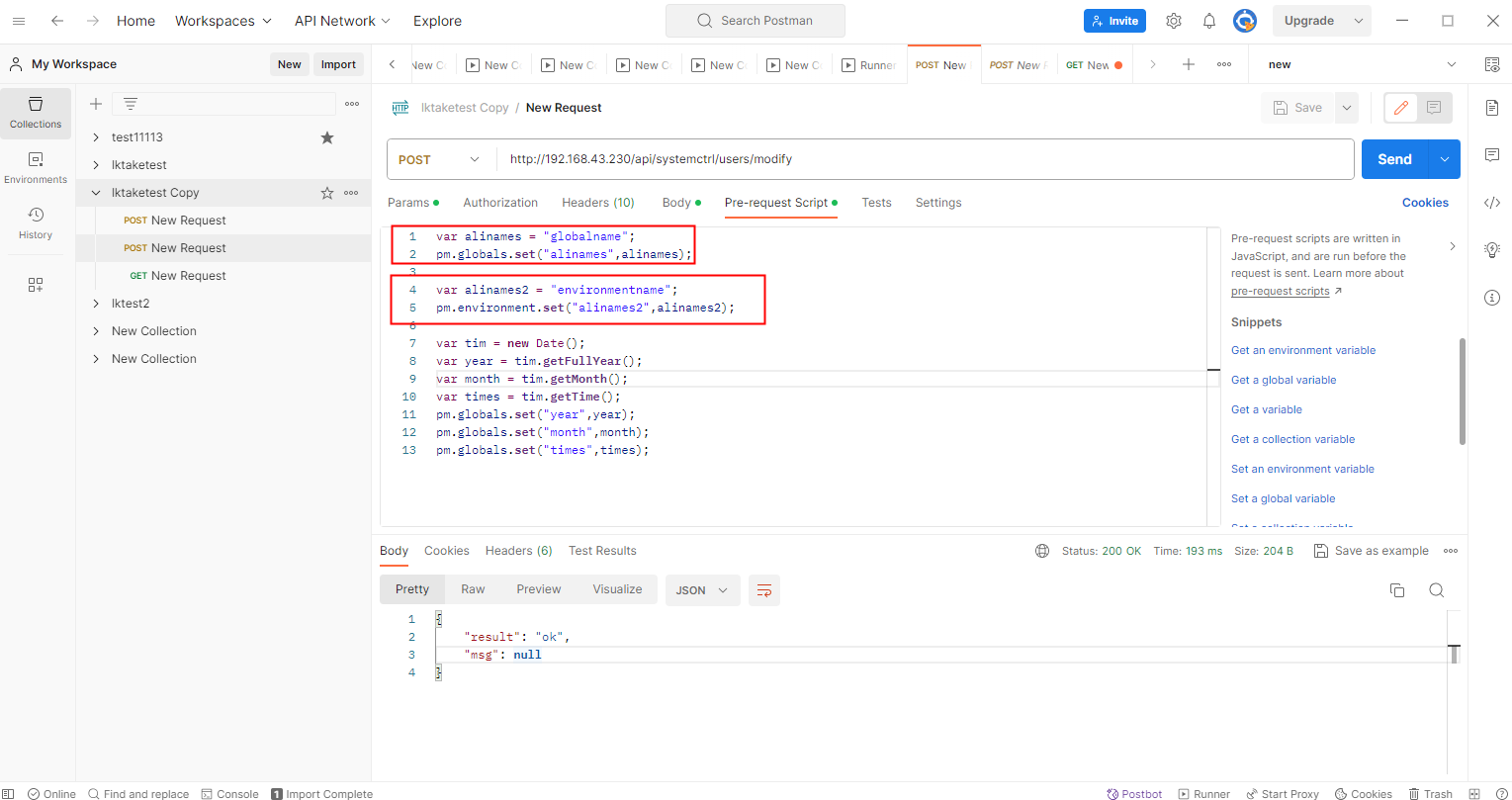
\*自定义全局变量、自定义环境变量、随机数，时间戳使用

var alinames = "globalname";  
pm.globals.set("alinames",alinames);

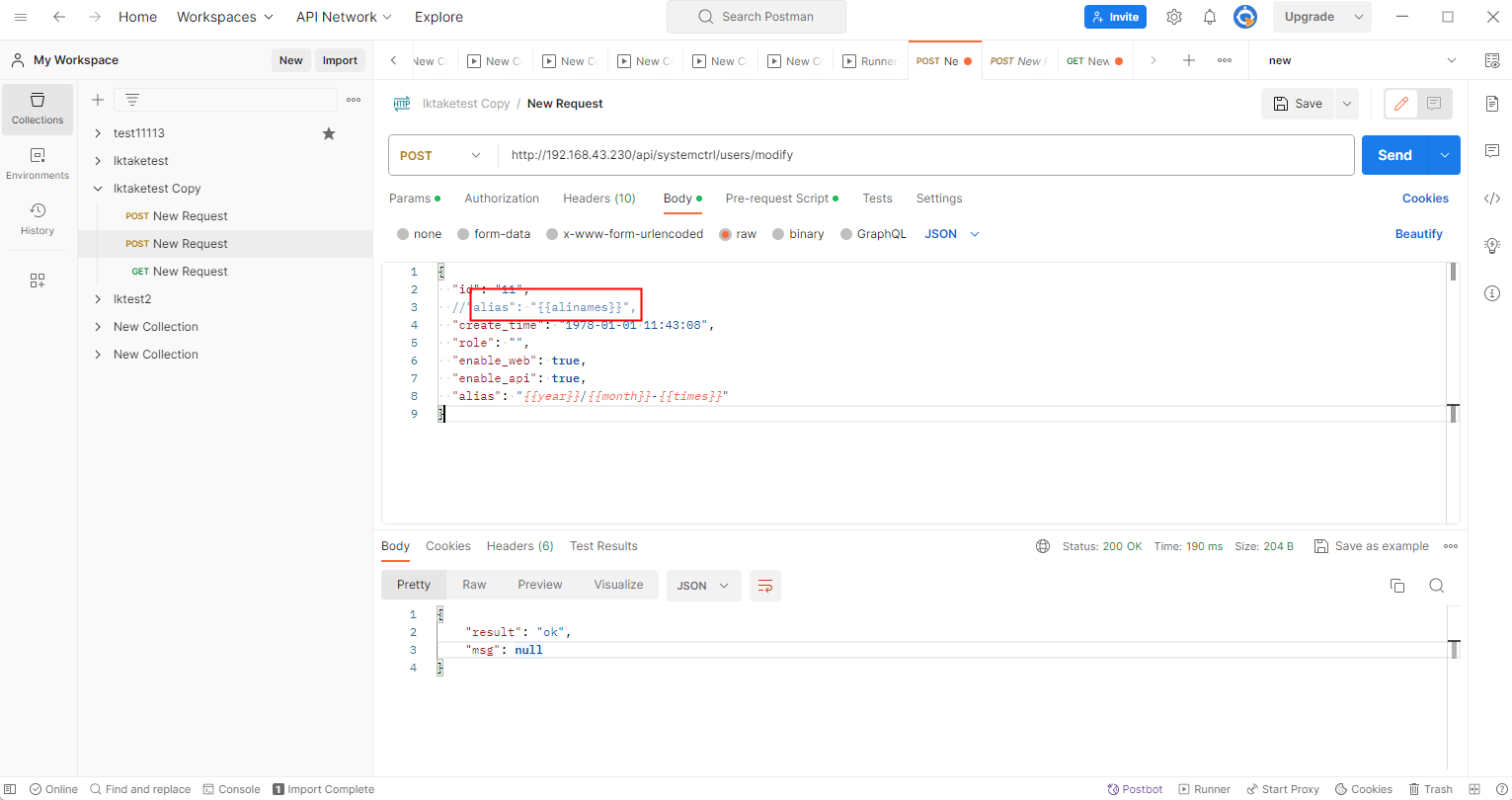
全局变量在其他测试集中也能调用

var alinames2 = "environmentname";  
pm.environment.set("alinames2",alinames2);

环境变量仅在当前测试集中使用

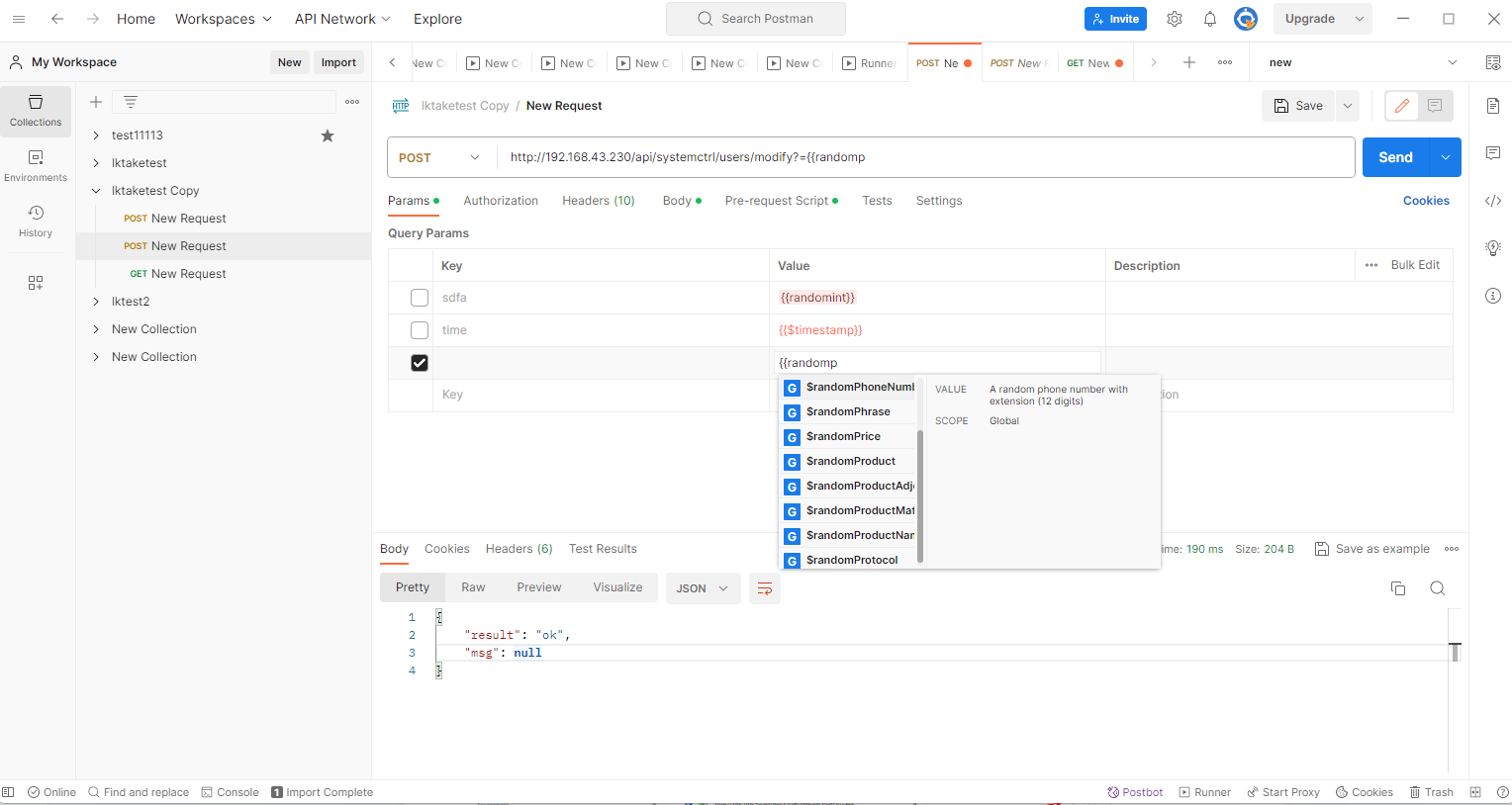


变量的使用



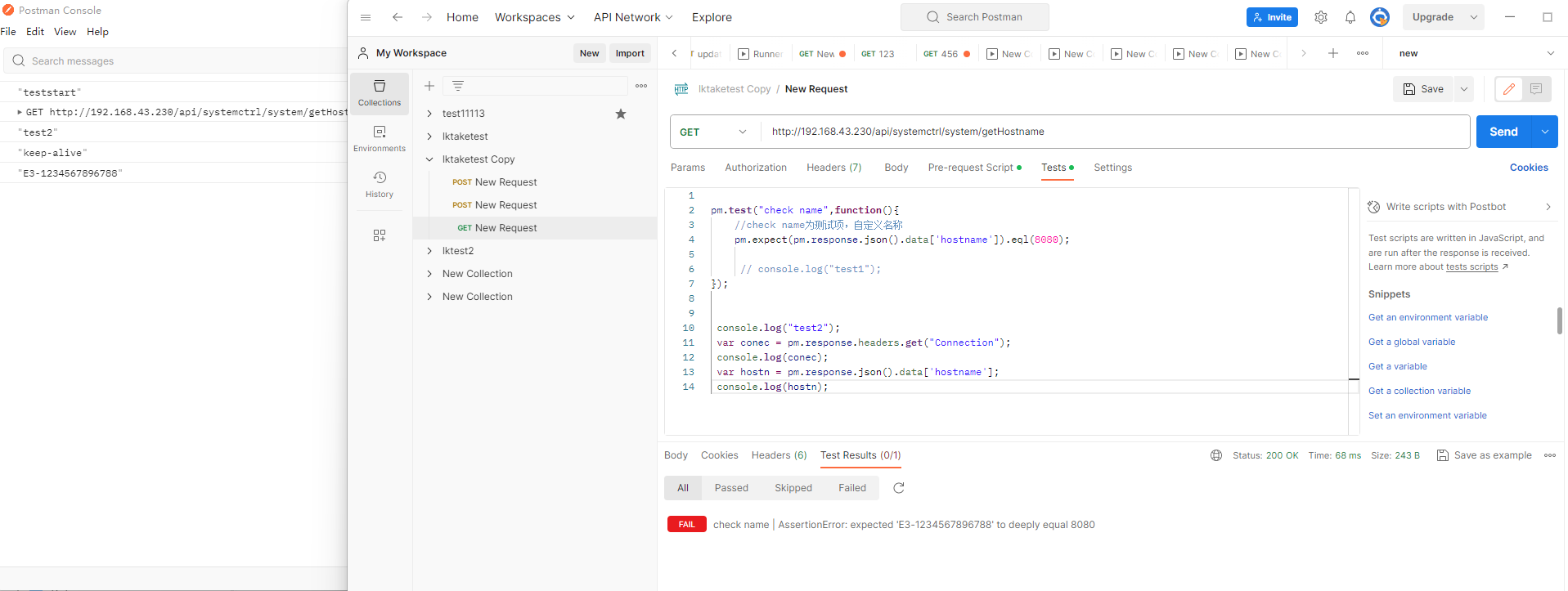
{{$randomInt}}随机一个1到1000的随机数

{{$timestamp}}随机时间戳

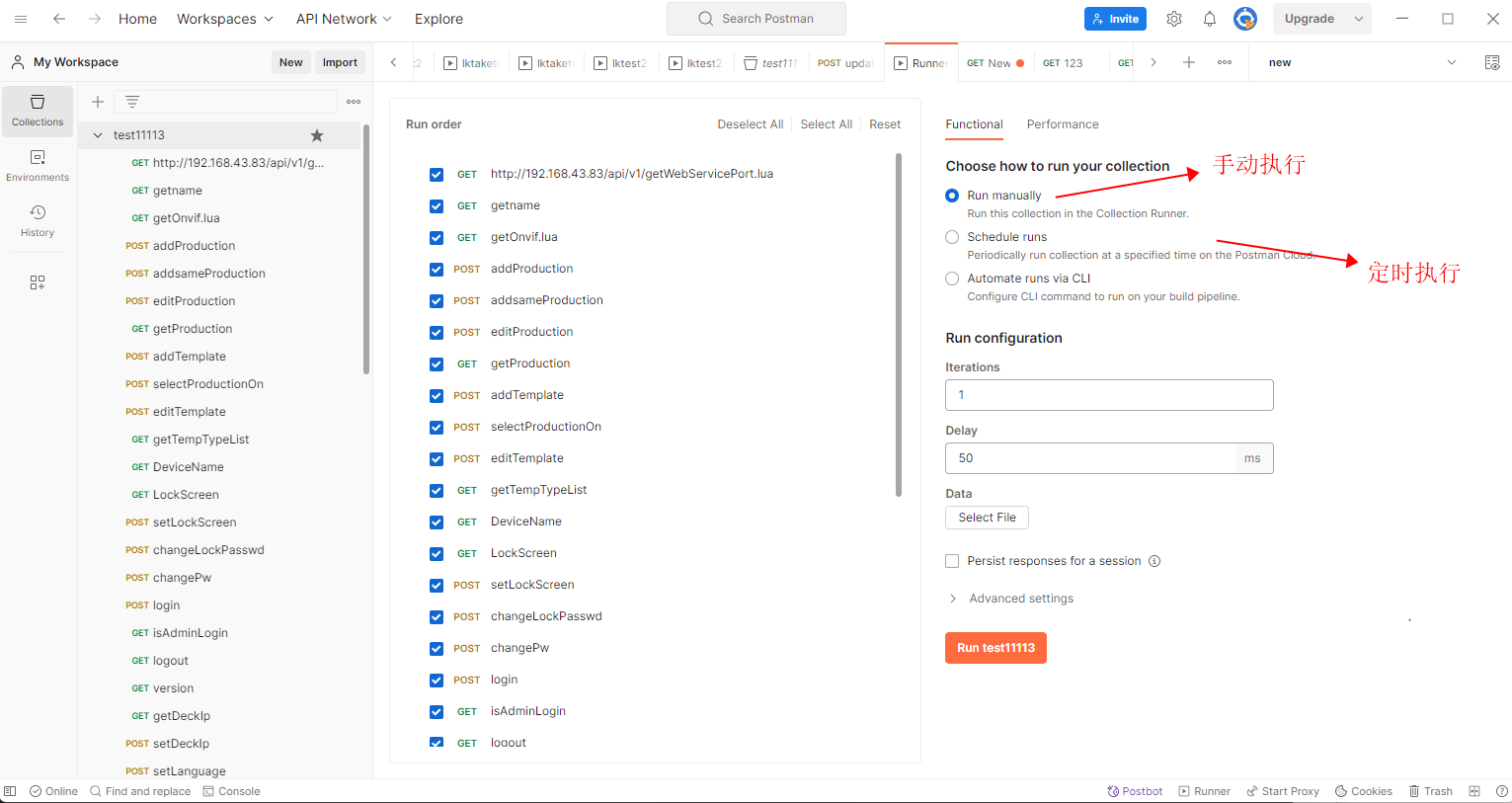


\*调试

使用快捷键ctrl+alt+c打开调试窗口，可以将一些变量值或者请求返回信息打印出来

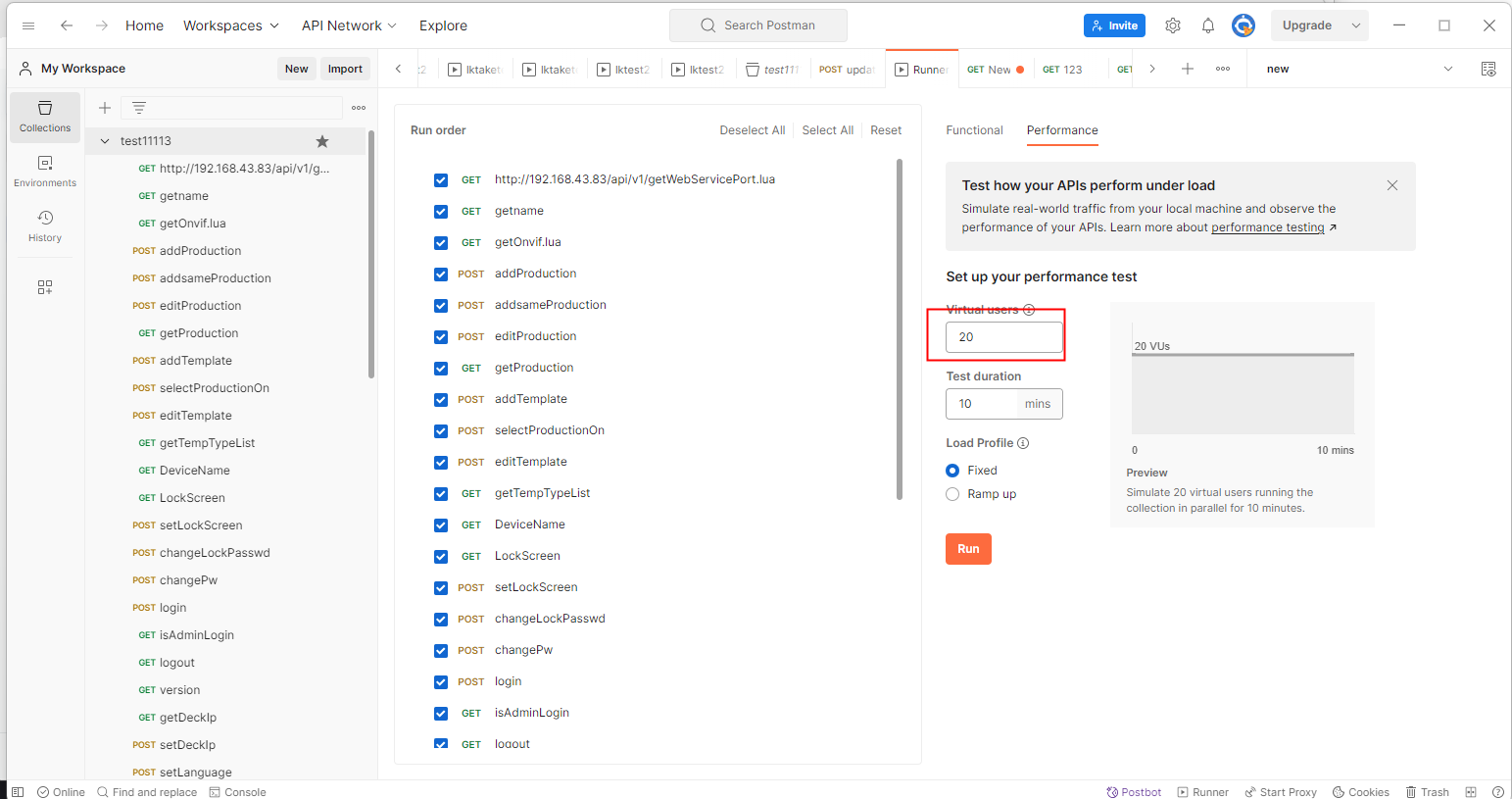


\*运行模式可以选择手动执行和定时执行



\*接口性能测试，可以选择performance模块

选择并发请求的用户，设置测试的时长



fixed模式为固定n个用户并发请求，ramp up模式为请求用户逐步增加至n个进行请求

