大疆互联网的一站式自动化测试解决方案（基于HttpRunner）

李隆（debugtalk）

MTSC2018
第四届中国移动互联网测试开发大会

大疆创新

TesterHome

自我介绍

- 李隆　Leo Lee
- 九毫
- debugtalk

大疆互联网事业部

高级测试开发工程师

测试开发团队负责人

You can find me here:

- **Blog:** http://debugtalk.com
- **GitHub:** https://github.com/debugtalk
- **TesterHome:** https://testerhome.com/debugtalk
- 微信公众号：**DebugTalk**

主题概要

1、背景介绍

2、核心目标

3、**设计思路**

4、核心特性

5、**大疆内部实践**

6、Q & A

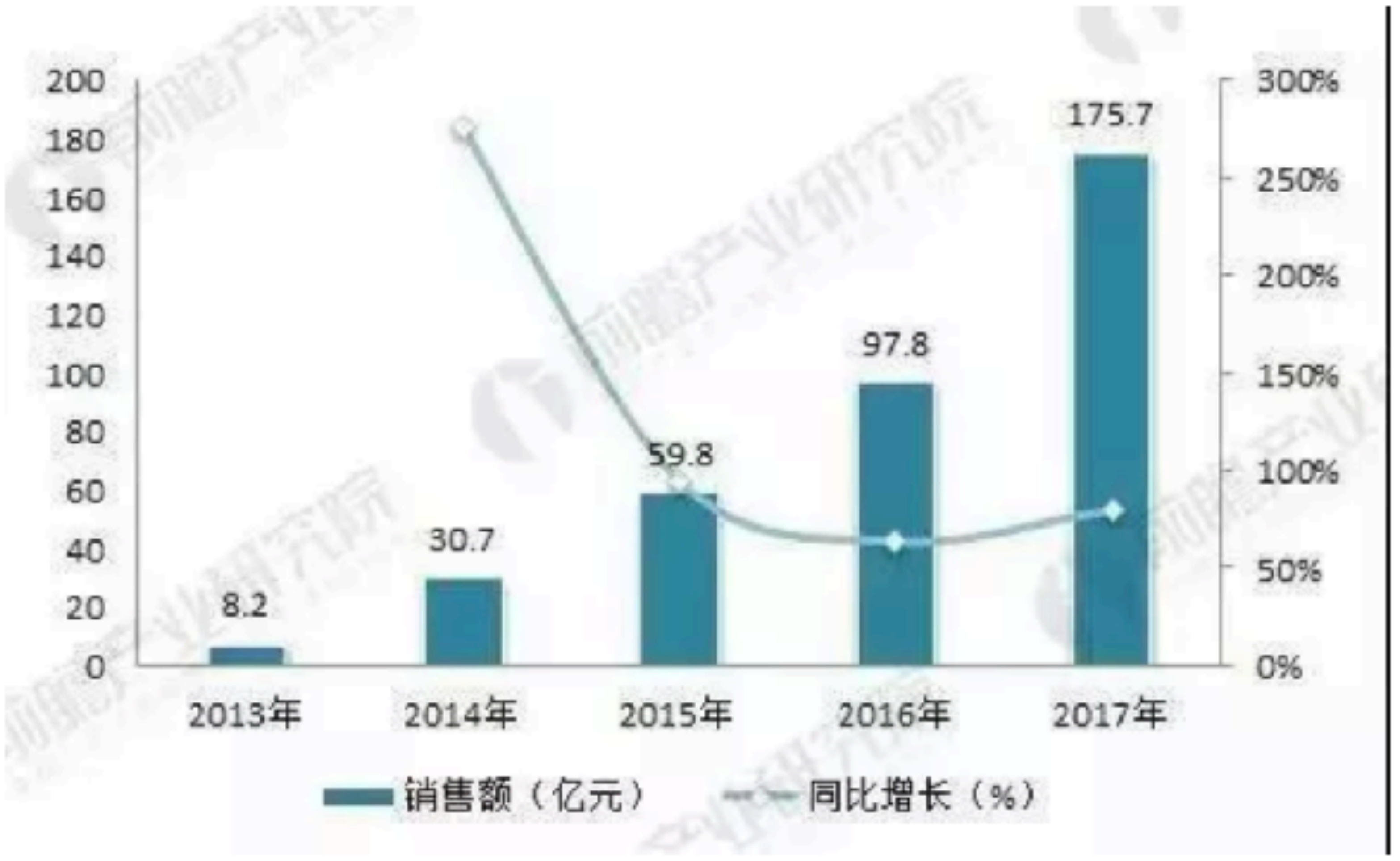
1、背景介绍

背景介绍

1、业务线众多：电商、生产制造、管理运营，etc.

2、高速发展和扩张：业务增长、人员增长

3、普遍具有功能回归测试、性能测试、持续集成、线上监控等需求



2013~2017年大疆销售额（来源：前瞻产业研究院）

效率低

成本高

背景介绍

自动化测试?

背景介绍

传统自动化测试方法存在的问题：

- 工具多 & 杂
- 学习成本高
- 团队协作难
- 维护成本高

2、核心目标

核心目标

业务导向，快速落地

少投入

- 工具开发&维护
- 学习使用成本
- 编写&录入测试用例
- 测试用例管理&维护

高收益

- 实现自动化回归测试
- 性能测试脚本复用
- 兼具持续集成、线上监控
- 辅助手工测试：自定义生成特定业务数据

3、设计思路

设计思路

- 如何快速实现从0到1?
- 如何避免闭门造车?
- 如何兼具自动化、性能、持续集成、线上监控?
- 如何保障工具质量?
- 如何实现协作管理?

设计思路 - 如何快速实现从0到1?

充分复用开源项目

Grafana
Jenkins
pytest
POSTMAN
swagger
YAML
JSON
Requests
http for humans
python
Charles
WEB DEBUGGING PROXY
charlesproxy.com
JSON Schema
HttpRunner
unittest
Fiddler
Web Debugging Proxy
Flask
web development,
one drop at a time
Jinja
LOCUST

设计思路 - 如何避免闭门造车?

拥抱开源生态

cn.httprunner.org

HttpRunner 中文使用手册

HttpRunner 中文使用手册
- 介绍
- 安装说明
- 快速上手
- 测试用例
- 运行测试
- 测试报告
- 高级特性
- 性能测试
- 开发扩展
- 常见问题

介绍

HttpRunner 是一款面向 HTTP(S) 协议的通用测试框架，只需编写维护一份 YAML/JSON 脚本，即可实现自动化测试、性能测试、线上监控、持续集成等多种测试需求。

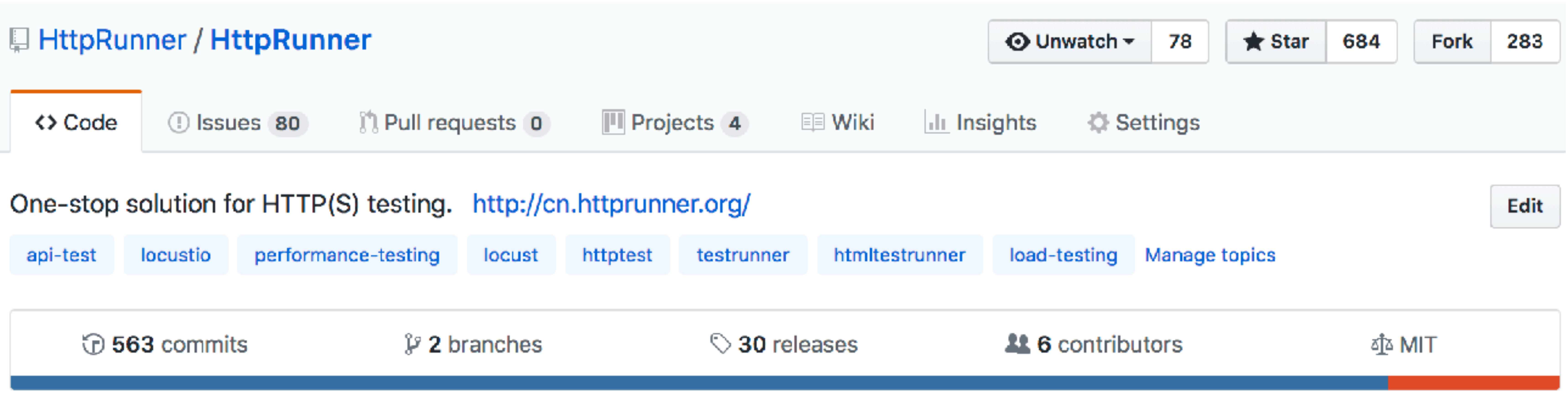
设计理念

- 充分复用优秀的开源项目，不追求重复造轮子，而是将强大的轮子组装成战车
- 遵循 约定大于配置 的准则，在框架功能中融入最佳工程实践
- 追求投入产出比，一份投入即可实现多种测试需求

逻辑流程图

目录
- 设计理念
- 逻辑流程图
- 核心特性
- 参考信息

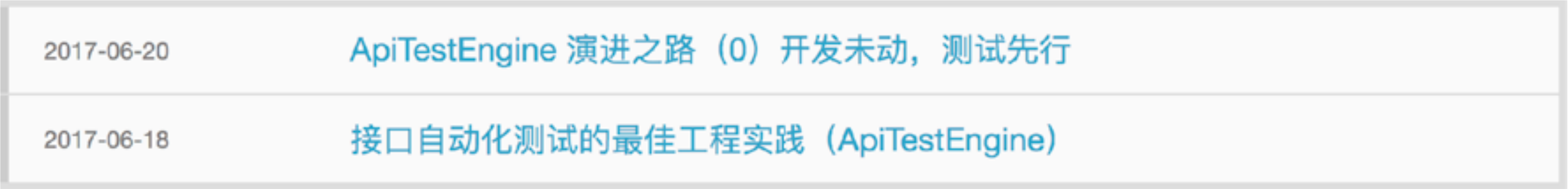
博客驱动开发


HttpRunner / HttpRunner

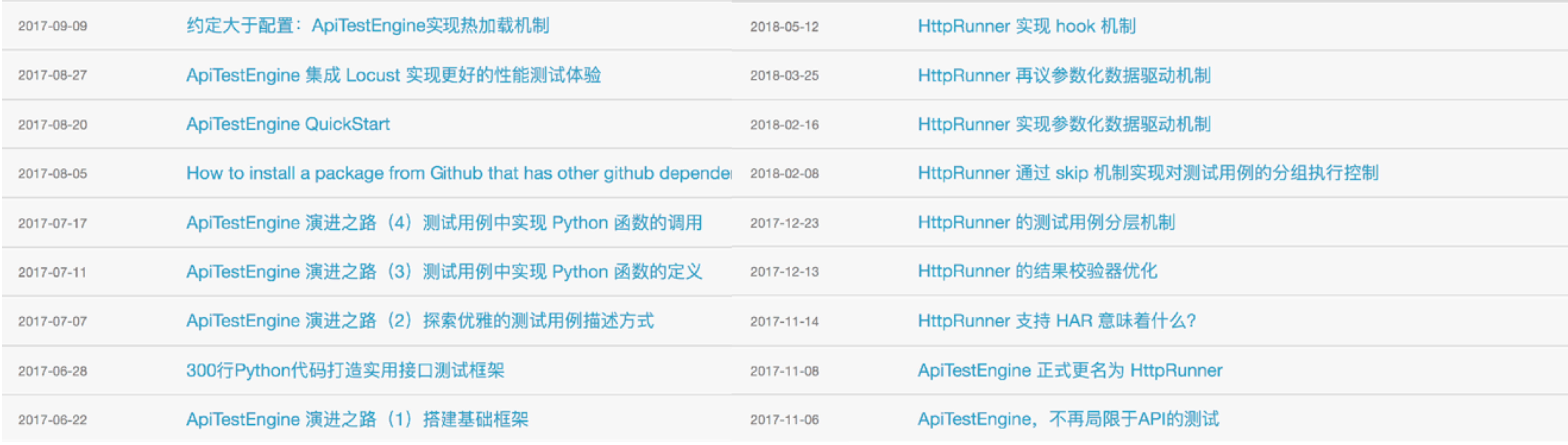
One-stop solution for HTTP(S) testing. http://cn.httprunner.org/

api-test locustio performance-testing locust httptest testrunner htmltestrunner load-testing Manage topics

563 commits · 2 branches · 30 releases · 6 contributors · MIT

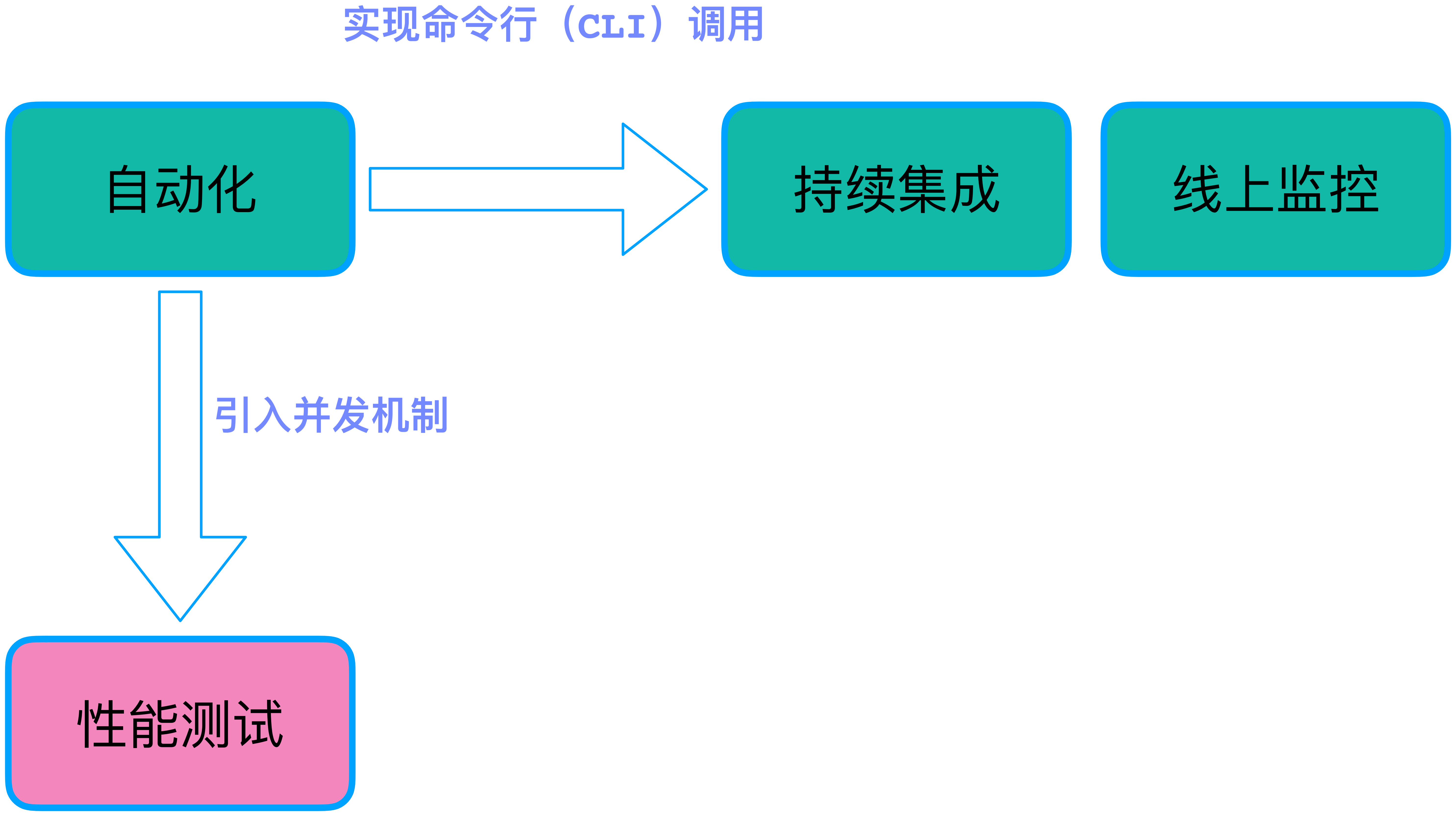


2017-06-20	ApiTestEngine 演进之路（0）开发未动，测试先行
2017-06-18	接口自动化测试的最佳工程实践（ApiTestEngine）



2017-09-09	约定大于配置：ApiTestEngine实现热加载机制	2018-05-12	HttpRunner 实现 hook 机制
2017-08-27	ApiTestEngine 集成 Locust 实现更好的性能测试体验	2018-03-25	HttpRunner 再议参数化数据驱动机制
2017-08-20	ApiTestEngine QuickStart	2018-02-16	HttpRunner 实现参数化数据驱动机制
2017-08-05	How to install a package from Github that has other github depende	2018-02-08	HttpRunner 通过 skip 机制实现对测试用例的分组执行控制
2017-07-17	ApiTestEngine 演进之路（4）测试用例中实现 Python 函数的调用	2017-12-23	HttpRunner 的测试用例分层机制
2017-07-11	ApiTestEngine 演进之路（3）测试用例中实现 Python 函数的定义	2017-12-13	HttpRunner 的结果校验器优化
2017-07-07	ApiTestEngine 演进之路（2）探索优雅的测试用例描述方式	2017-11-14	HttpRunner 支持 HAR 意味着什么?
2017-06-28	300行Python代码打造实用接口测试框架	2017-11-08	ApiTestEngine 正式更名为 HttpRunner
2017-06-22	ApiTestEngine 演进之路（1）搭建基础框架	2017-11-06	ApiTestEngine，不再局限于API的测试

设计思路 - 如何兼具自动化、性能、持续集成、线上监控?



实现命令行（CLI）调用

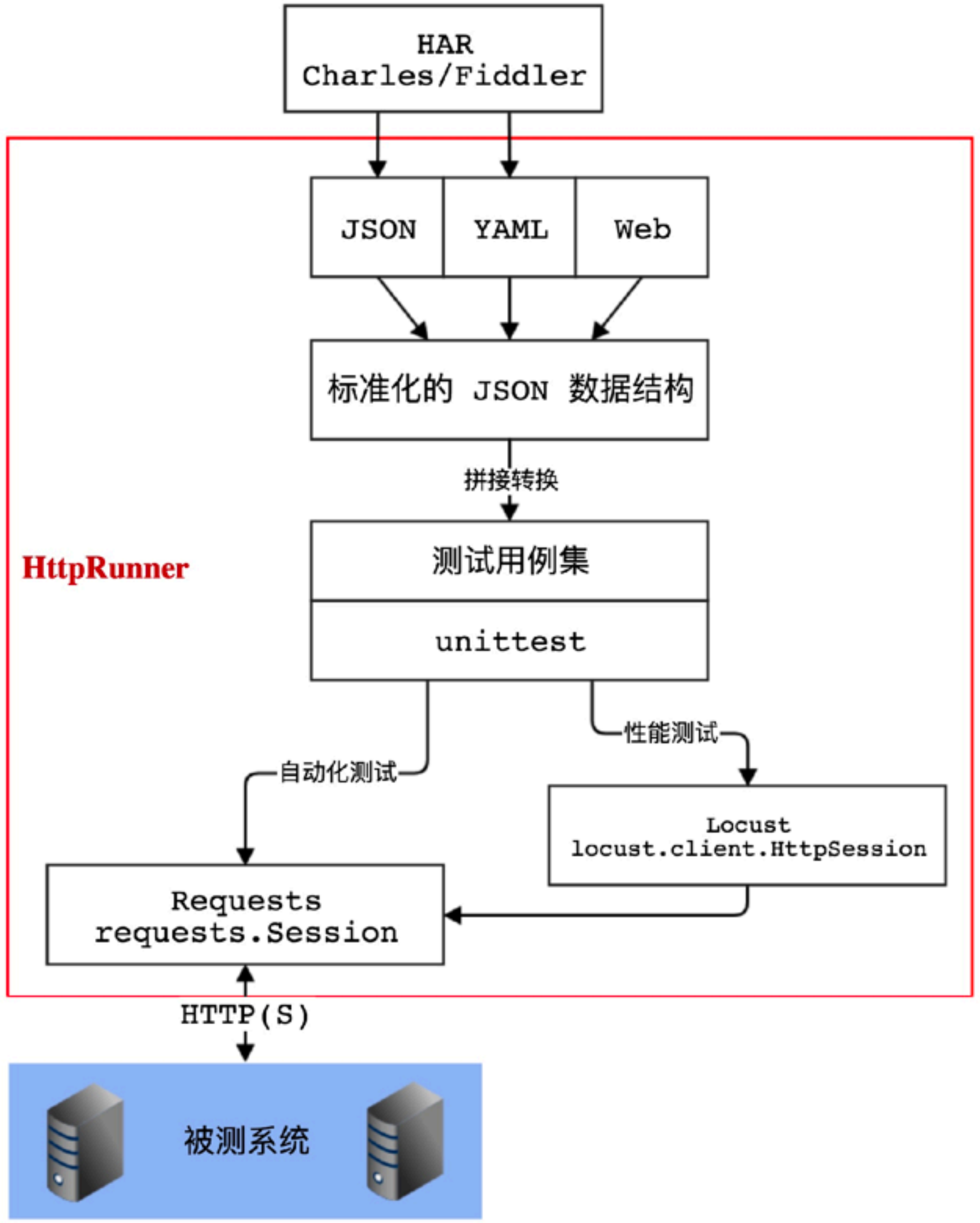
自动化

持续集成

线上监控

引入并发机制

性能测试



HAR
Charles/Fiddler

HttpRunner

JSON

YAML

Web

标准化的 JSON 数据结构

拼接转换

测试用例集

unittest

自动化测试

性能测试

Locust
locust.client.HttpSession

Requests
requests.Session

HTTP(S)

被测系统

设计思路 - 如何保障工具质量?

测试驱动开发

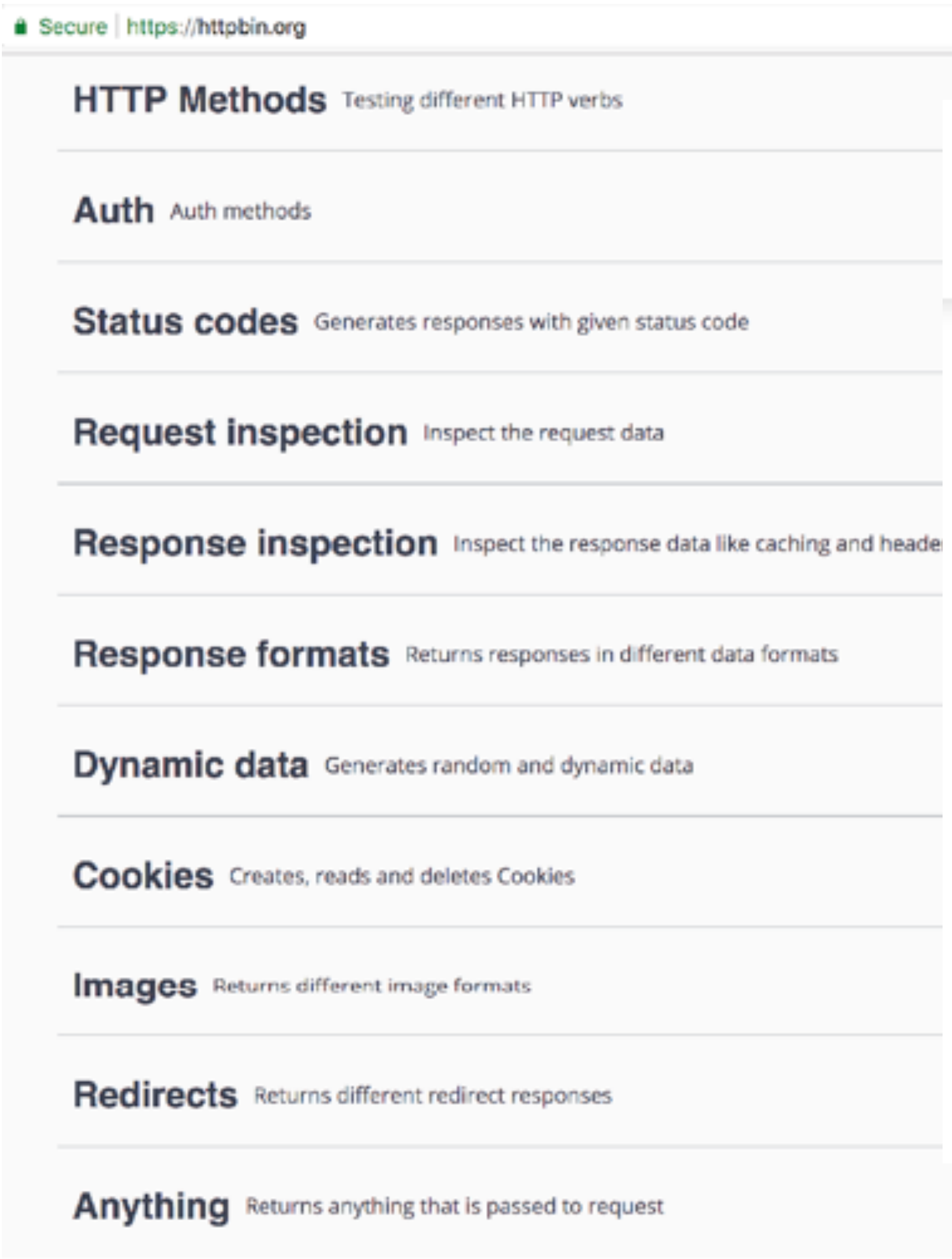
- Custom Mock Server
- httpbin.org by kennethreitz
- Postman Echo services

开源社区反馈

- TesterHome
- Github issues

HttpRunner

license MIT | build passing | coverage 77% | pypi v1.4.8 | python 2.7, 3.4, 3.5, 3.6


Secure | https://httpbin.org

HTTP Methods Testing different HTTP verbs

Auth Auth methods

Status codes Generates responses with given status code

Request inspection Inspect the request data

Response inspection Inspect the response data like caching and heade

Response formats Returns responses in different data formats

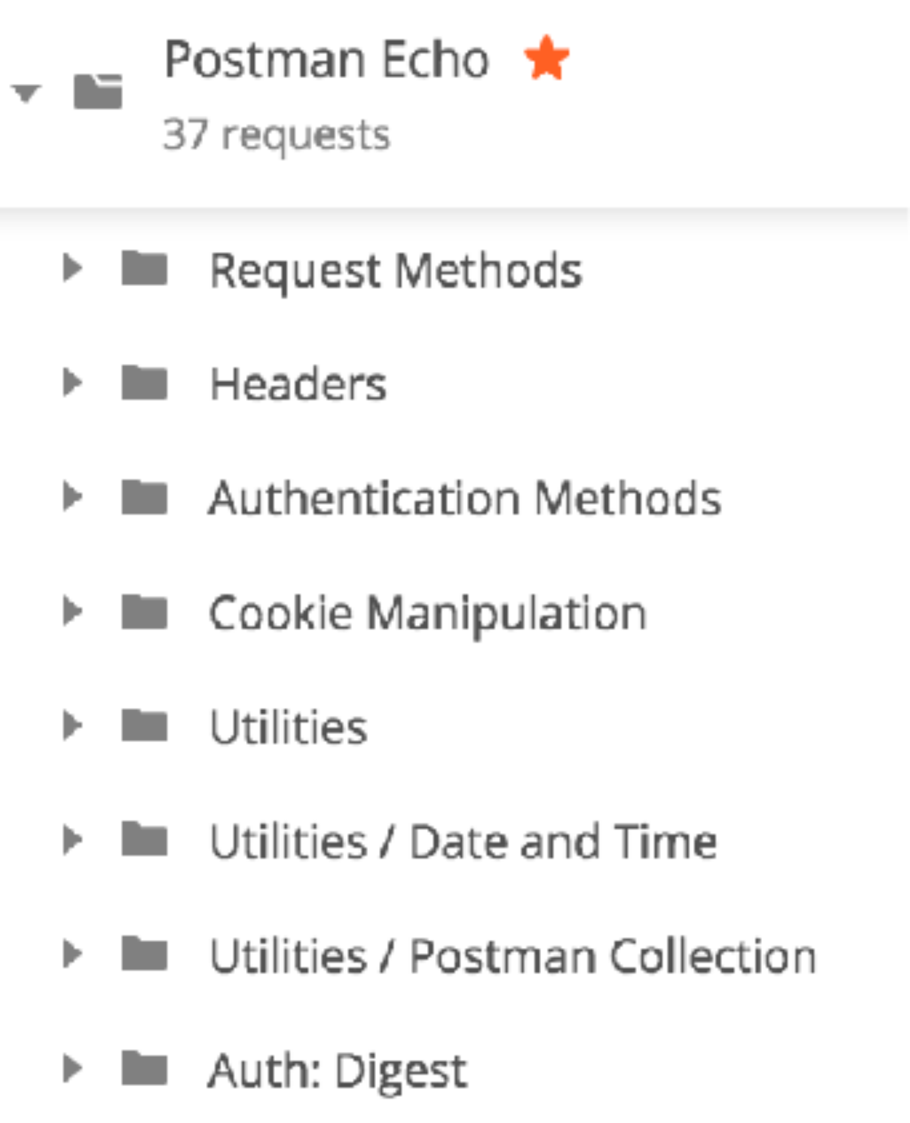
Dynamic data Generates random and dynamic data

Cookies Creates, reads and deletes Cookies

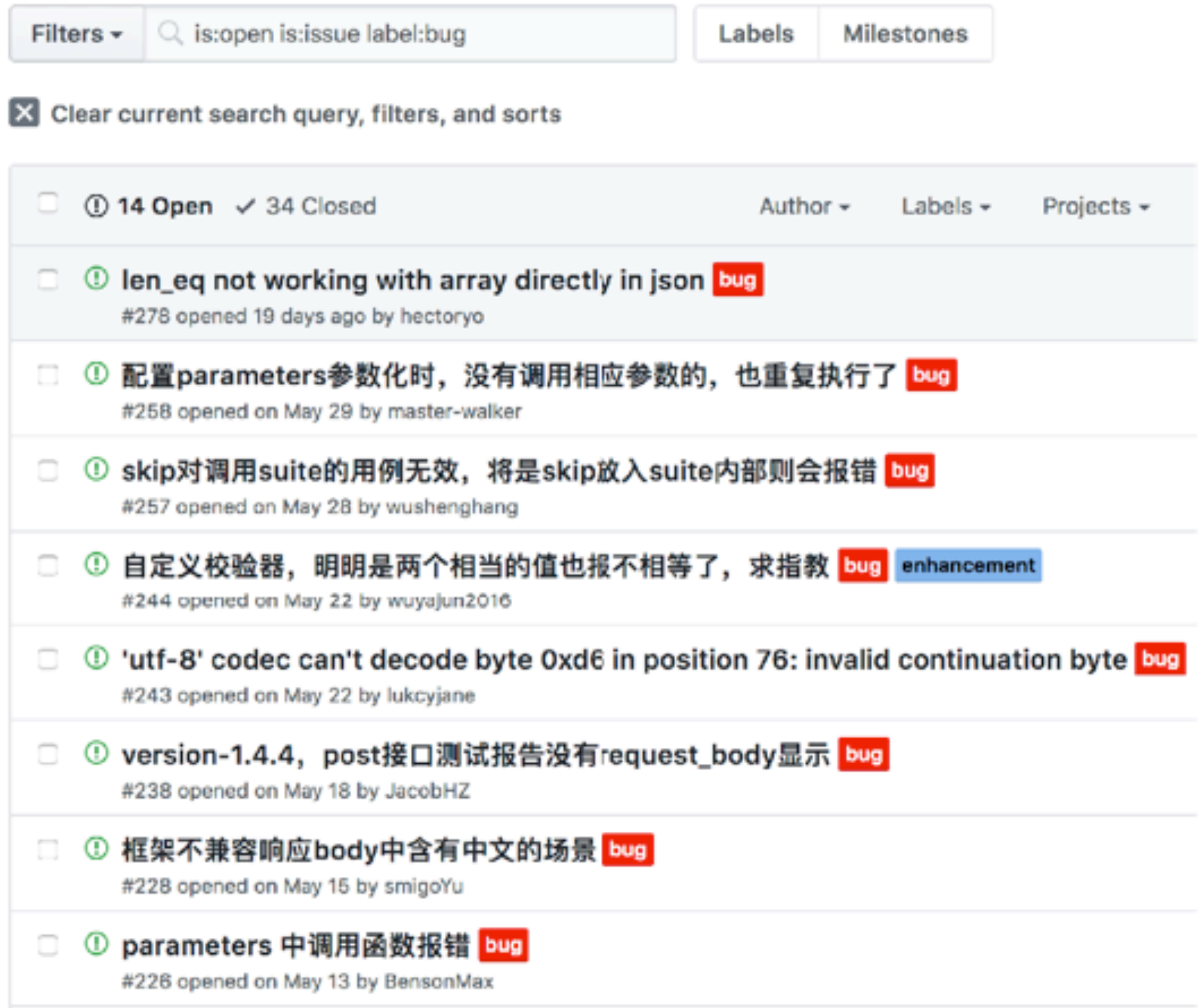
Images Returns different image formats

Redirects Returns different redirect responses

Anything Returns anything that is passed to request


Postman Echo ★
37 requests

- Request Methods
- Headers
- Authentication Methods
- Cookie Manipulation
- Utilities
- Utilities / Date and Time
- Utilities / Postman Collection
- Auth: Digest

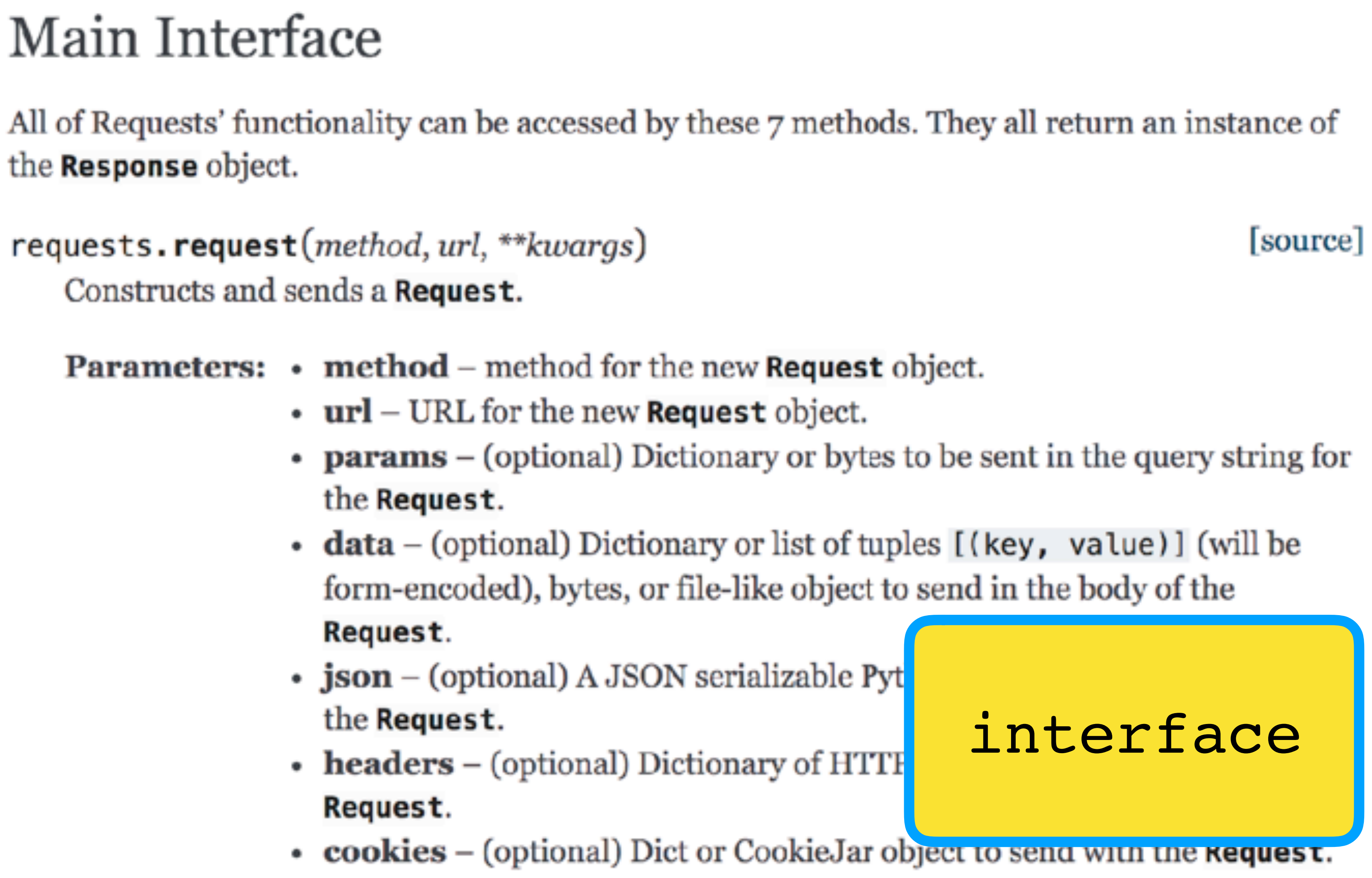


☒ Clear current search query, filters, and sorts

ⓘ 14 Open ✓ 34 Closed Author ▾ Labels ▾ Projects ▾

- len_eq not working with array directly in json `bug`
  #278 opened 19 days ago by hectoryo
- 配置parameters参数化时，没有调用相应参数的，也重复执行了 `bug`
  #258 opened on May 29 by master-walker
- skip对调用suite的用例无效，将是skip放入suite内部则会报错 `bug`
  #257 opened on May 28 by wushenghang
- 自定义校验器，明明是两个相当的值也报不相等了，求指教 `bug` `enhancement`
  #244 opened on May 22 by wuyajun2016
- 'utf-8' codec can't decode byte 0xd6 in position 76: invalid continuation byte `bug`
  #243 opened on May 22 by lukcyjane
- version-1.4.4，post接口测试报告没有request_body显示 `bug`
  #238 opened on May 18 by JacobHZ
- 框架不兼容响应body中含有中文的场景 `bug`
  #228 opened on May 15 by smigoYu
- parameters 中调用函数报错 `bug`
  #226 opened on May 13 by BensonMax

设计思路 - 如何实现协作管理?



Main Interface

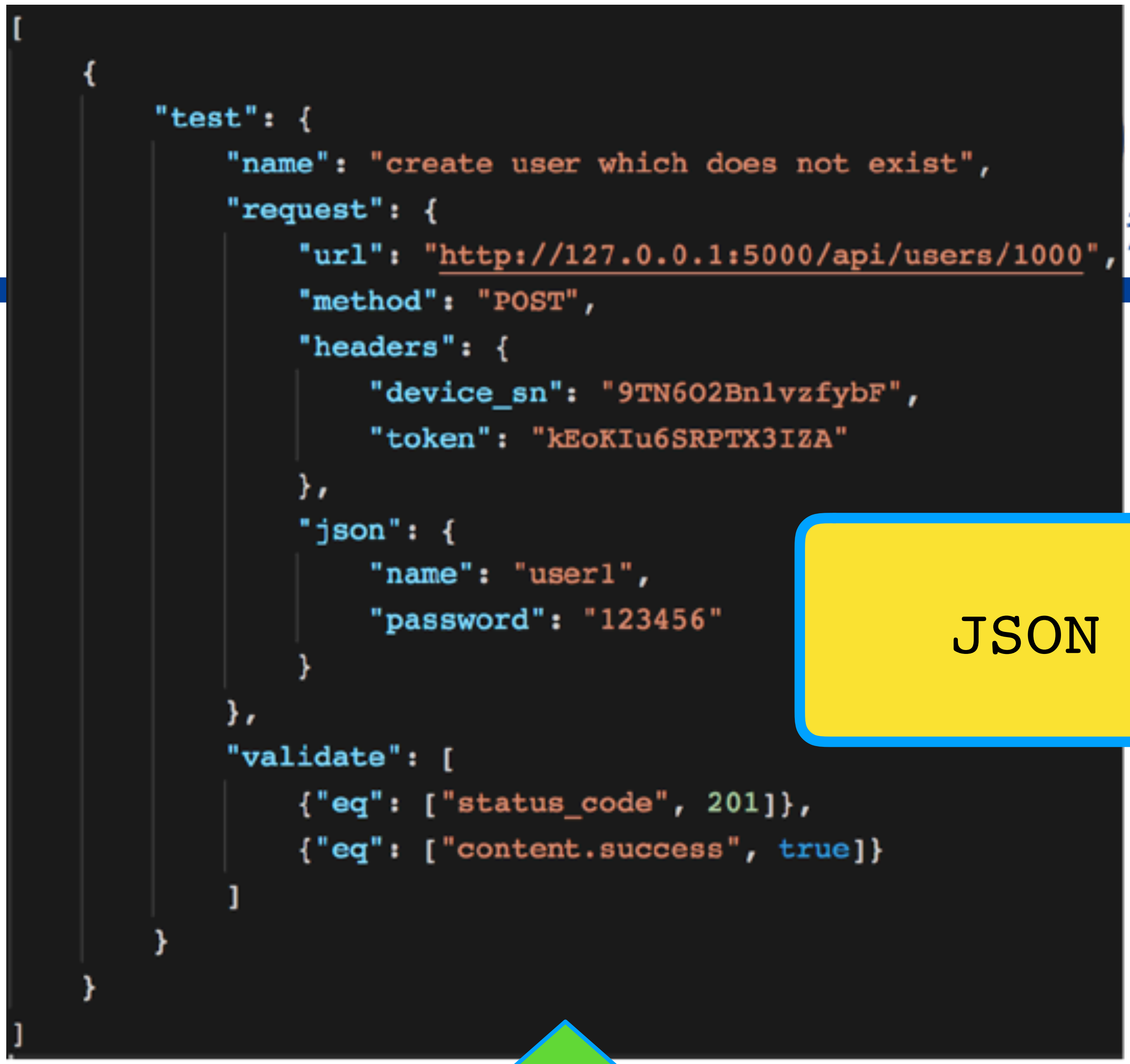
All of Requests' functionality can be accessed by these 7 methods. They all return an instance of the **Response** object.

`requests.request(method, url, **kwargs)` [source]

Constructs and sends a **Request**.

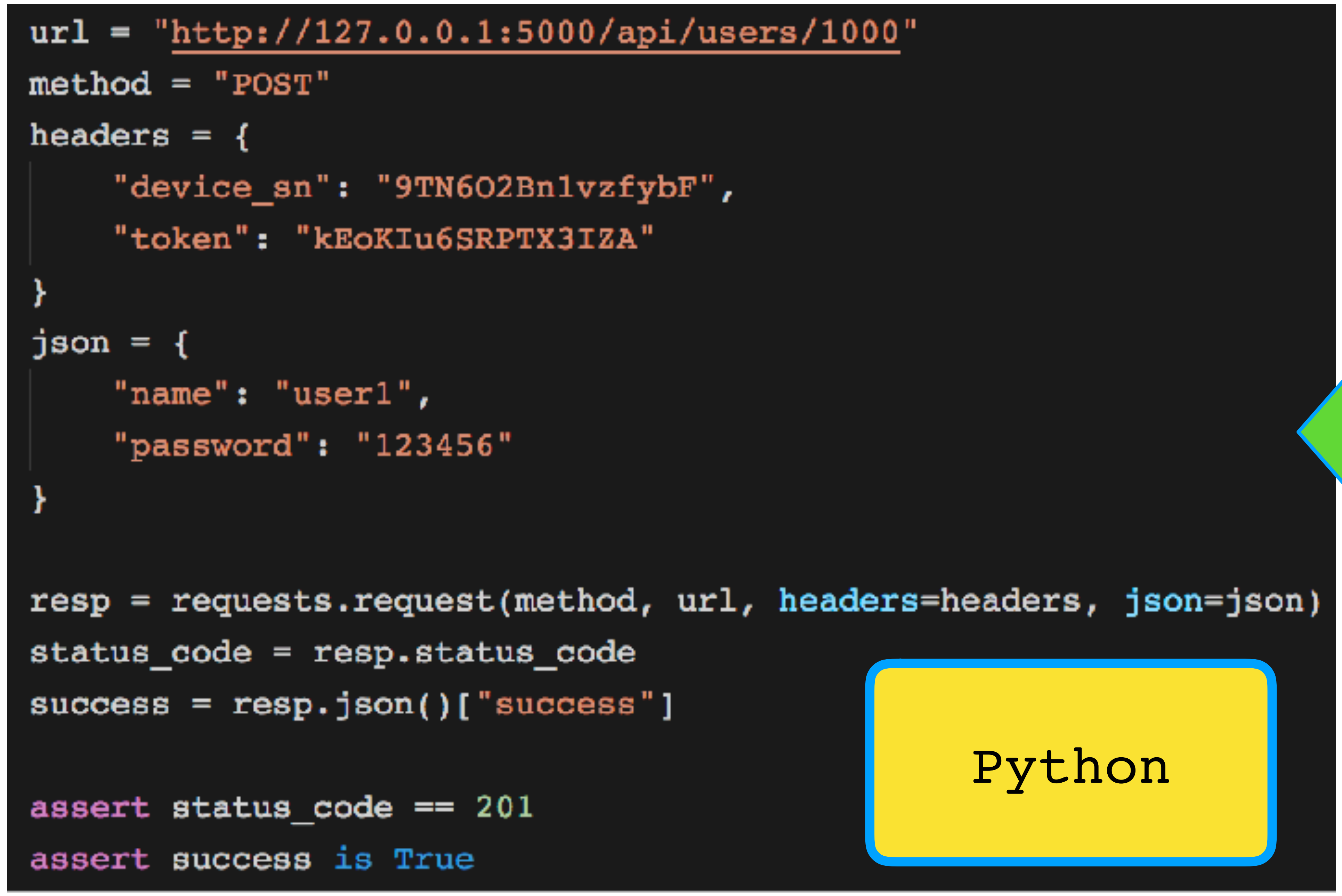
Parameters:
- **method** – method for the new **Request** object.
- **url** – URL for the new **Request** object.
- **params** – (optional) Dictionary or bytes to be sent in the query string for the **Request**.
- **data** – (optional) Dictionary or list of tuples `[(key, value)]` (will be form-encoded), bytes, or file-like object to send in the body of the **Request**.
- **json** – (optional) A JSON serializable Pyt the **Request**.
- **headers** – (optional) Dictionary of HTTI **Request**.
- **cookies** – (optional) Dict or CookieJar object to send with the **Request**.

interface



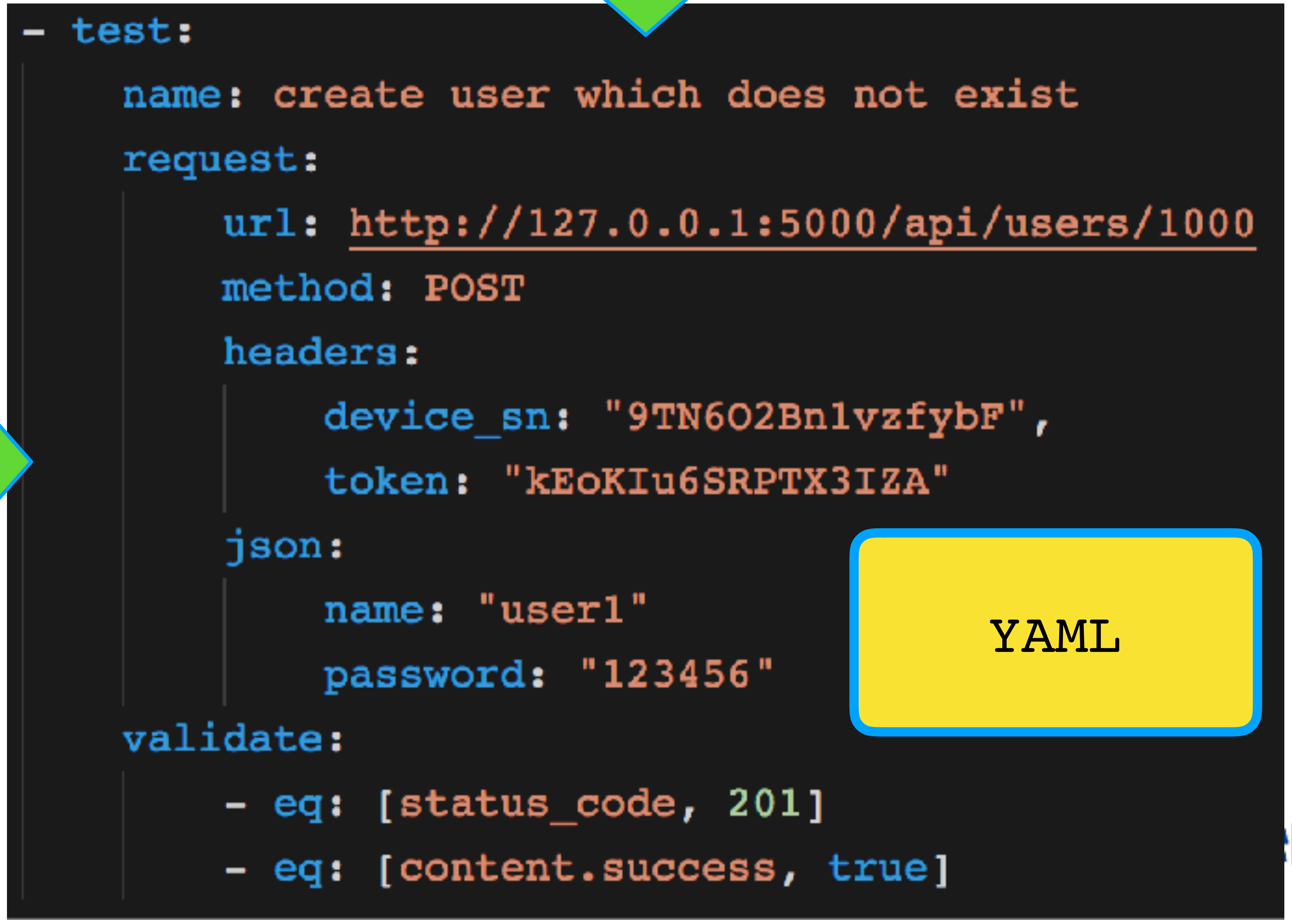
```
[
    {
        "test": {
            "name": "create user which does not exist",
            "request": {
                "url": "http://127.0.0.1:5000/api/users/1000",
                "method": "POST",
                "headers": {
                    "device_sn": "9TN6O2Bn1vzfybF",
                    "token": "kEoKIu6SRPTX3IZA"
                },
                "json": {
                    "name": "user1",
                    "password": "123456"
                }
            },
            "validate": [
                {"eq": ["status_code", 201]},
                {"eq": ["content.success", true]}
            ]
        }
    }
]
```

JSON



```
url = "http://127.0.0.1:5000/api/users/1000"
method = "POST"
headers = {
    "device_sn": "9TN6O2Bn1vzfybF",
    "token": "kEoKIu6SRPTX3IZA"
}
json = {
    "name": "user1",
    "password": "123456"
}

resp = requests.request(method, url, headers=headers, json=json)
status_code = resp.status_code
success = resp.json()["success"]

assert status_code == 201
assert success is True
```

Python



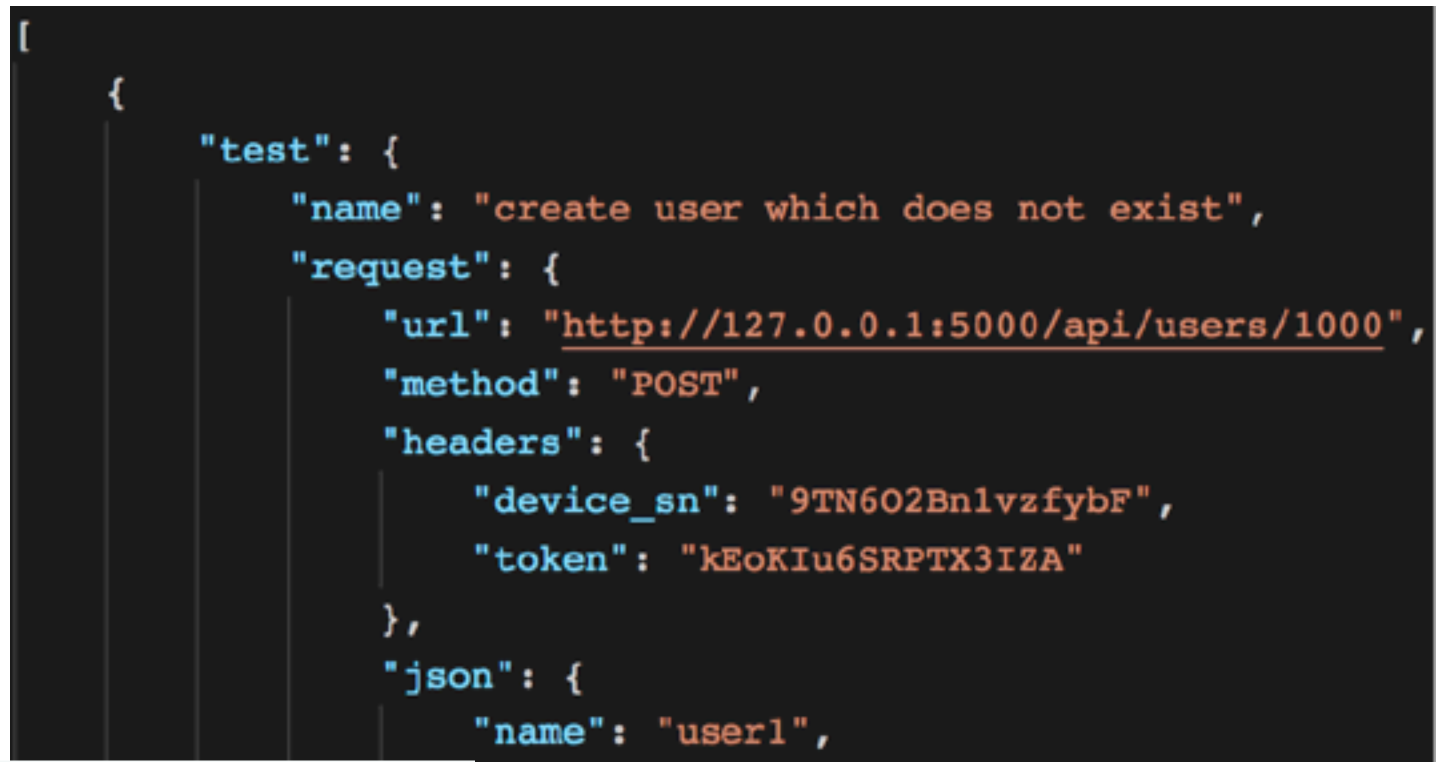
```
- test:
    name: create user which does not exist
    request:
        url: http://127.0.0.1:5000/api/users/1000
        method: POST
        headers:
            device_sn: "9TN6O2Bn1vzfybF",
            token: "kEoKIu6SRPTX3IZA"
        json:
            name: "user1"
            password: "123456"
    validate:
        - eq: [status_code, 201]
        - eq: [content.success, true]
```

YAML

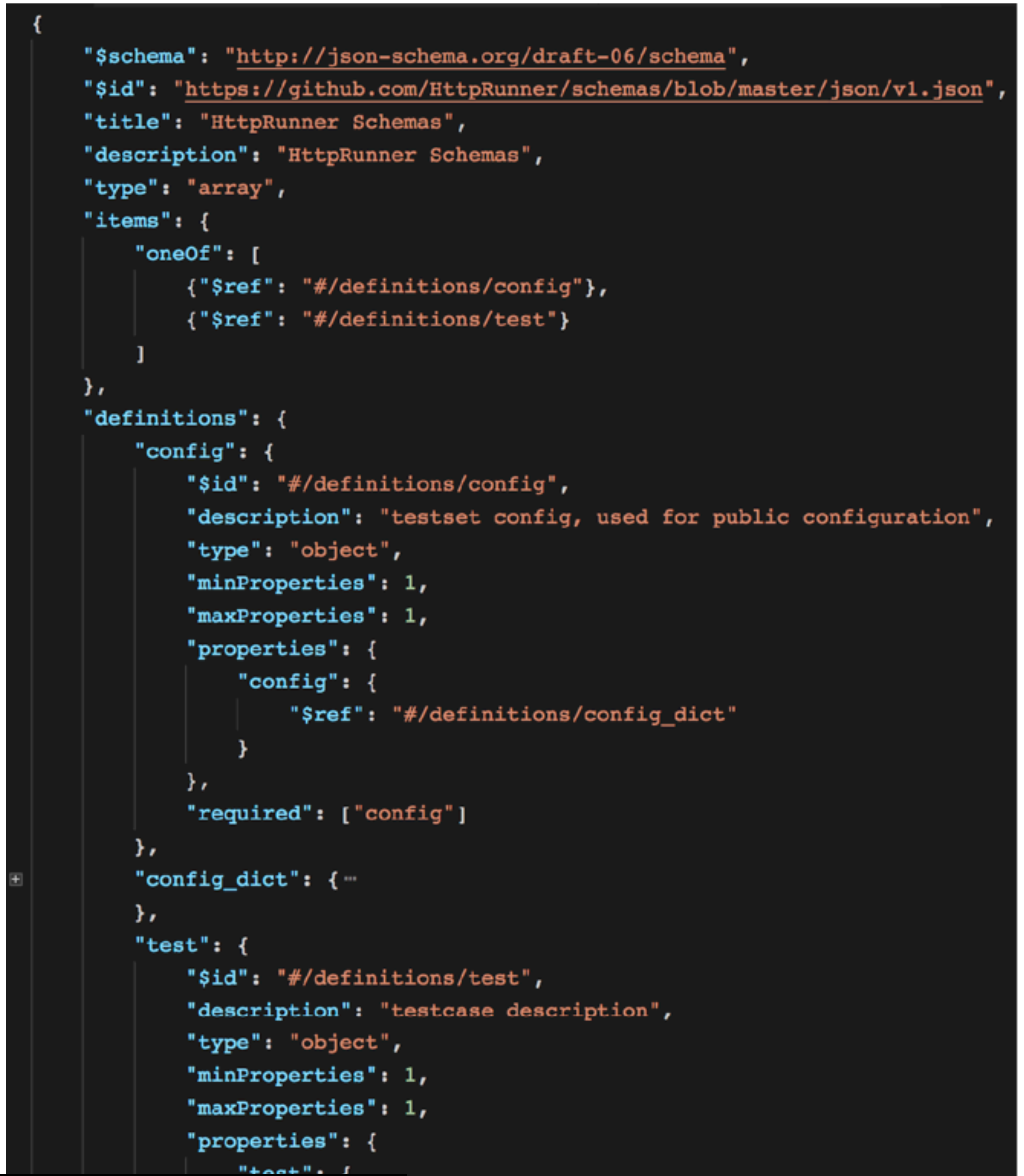
设计思路 - 如何实现协作管理?

约定大于配置

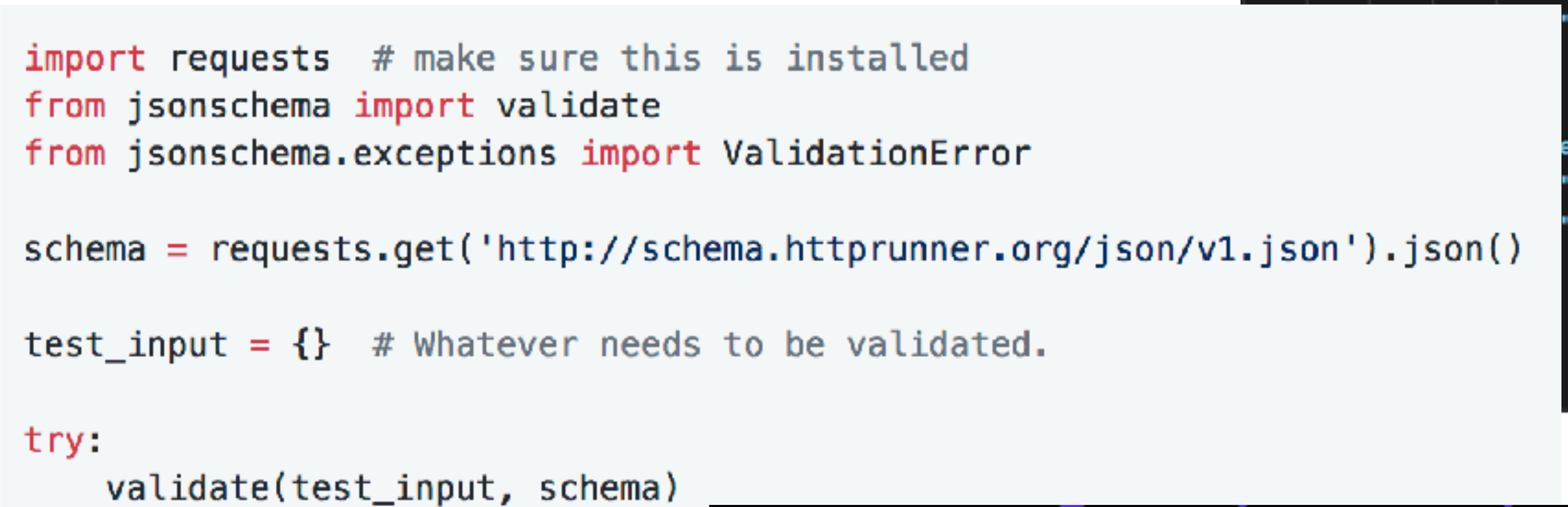
- 统一测试用例描述形式（JSON/YAML）
- JSON Schemas



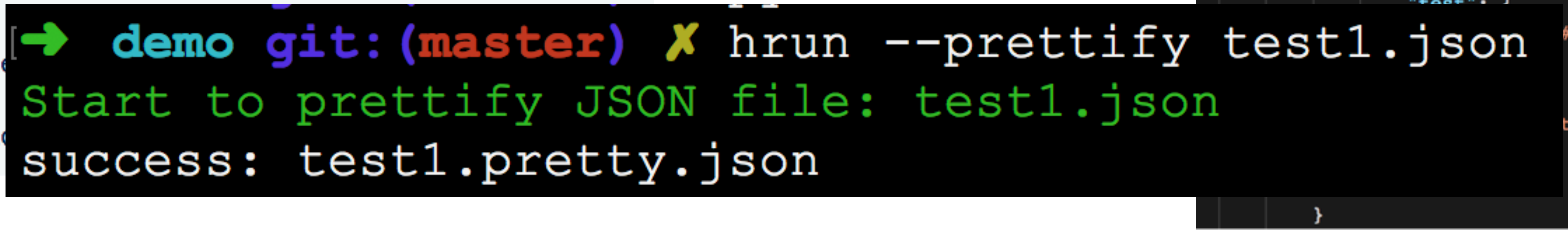
```
[
  {
    "test": {
      "name": "create user which does not exist",
      "request": {
        "url": "http://127.0.0.1:5000/api/users/1000",
        "method": "POST",
        "headers": {
          "device_sn": "9TN602Bn1vzfybF",
          "token": "kEoKIu6SRPTX3IZA"
        },
        "json": {
          "name": "user1",
          "password": "123456"

          e": [
          ": ["status_code", 201]},
          ": ["content.success", true]}
```



```
{
  "$schema": "http://json-schema.org/draft-06/schema",
  "$id": "https://github.com/HttpRunner/schemas/blob/master/json/v1.json",
  "title": "HttpRunner Schemas",
  "description": "HttpRunner Schemas",
  "type": "array",
  "items": {
    "oneOf": [
      {"$ref": "#/definitions/config"},
      {"$ref": "#/definitions/test"}
    ]
  },
  "definitions": {
    "config": {
      "$id": "#/definitions/config",
      "description": "testset config, used for public configuration",
      "type": "object",
      "minProperties": 1,
      "maxProperties": 1,
      "properties": {
        "config": {
          "$ref": "#/definitions/config_dict"
        }
      },
      "required": ["config"]
    },
    "config_dict": {⋯
    },
    "test": {
      "$id": "#/definitions/test",
      "description": "testcase description",
      "type": "object",
      "minProperties": 1,
      "maxProperties": 1,
      "properties": {
        "test": {
                    #/definitions/test_dict"

                    t"]

    }
```



```
import requests  # make sure this is installed
from jsonschema import validate
from jsonschema.exceptions import ValidationError

schema = requests.get('http://schema.httprunner.org/json/v1.json').json()

test_input = {}  # Whatever needs to be validated.

try:
    validate(test_input, schema)
except ValidationError:
    print 'It is not a valid colle
else:
    print 'It is a valid collecti
```



```
➜  demo git:(master) ✗ hrun --prettify test1.json
Start to prettify JSON file: test1.json
success: test1.pretty.json
```

设计思路 - 如何实现协作管理?

统一用例管理平台

- 统一存储和管理
- 平台服务化

社区中基于 HttpRunner 的接口测试平台：

- **HttpRunnerManager**
- SECO
- testcenter
- ApiTestWeb

设计思路 - 总结

HttpRunner 是什么?

- 一套新的自动化测试工具/框架 ？ ✗

- 通用的自动化测试解决方案
- 融合最佳工程实践

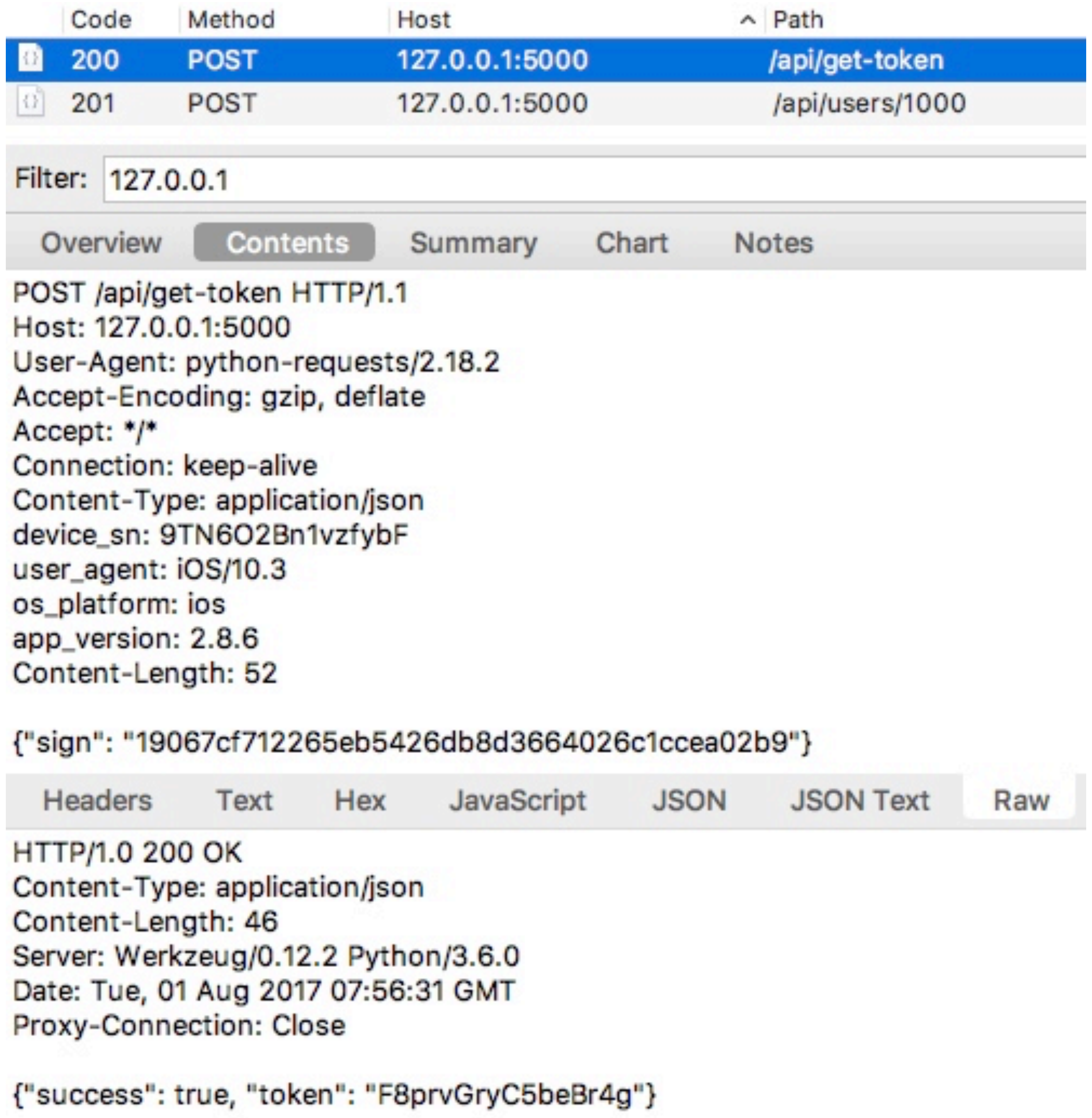
✓

4、核心特性

核心特性

- 抓包录制 & 生成用例
- *Postman* 转换生成用例
- *Swagger* 转换生成用例
- *YAML/JSON* 用例格式
- *JSON Schema*
- 数据驱动机制
- 测试用例分组执行控制
- *setup & teardown hooks*
- 热加载机制*(debugtalk.py)*
- 性能测试
- *CLI* 调用
- 可扩展测试报告*(Jinja2)*
- 测试用例分层机制
- 加载环境变量*(.env)*
- . . .

核心特性 - 代码示例

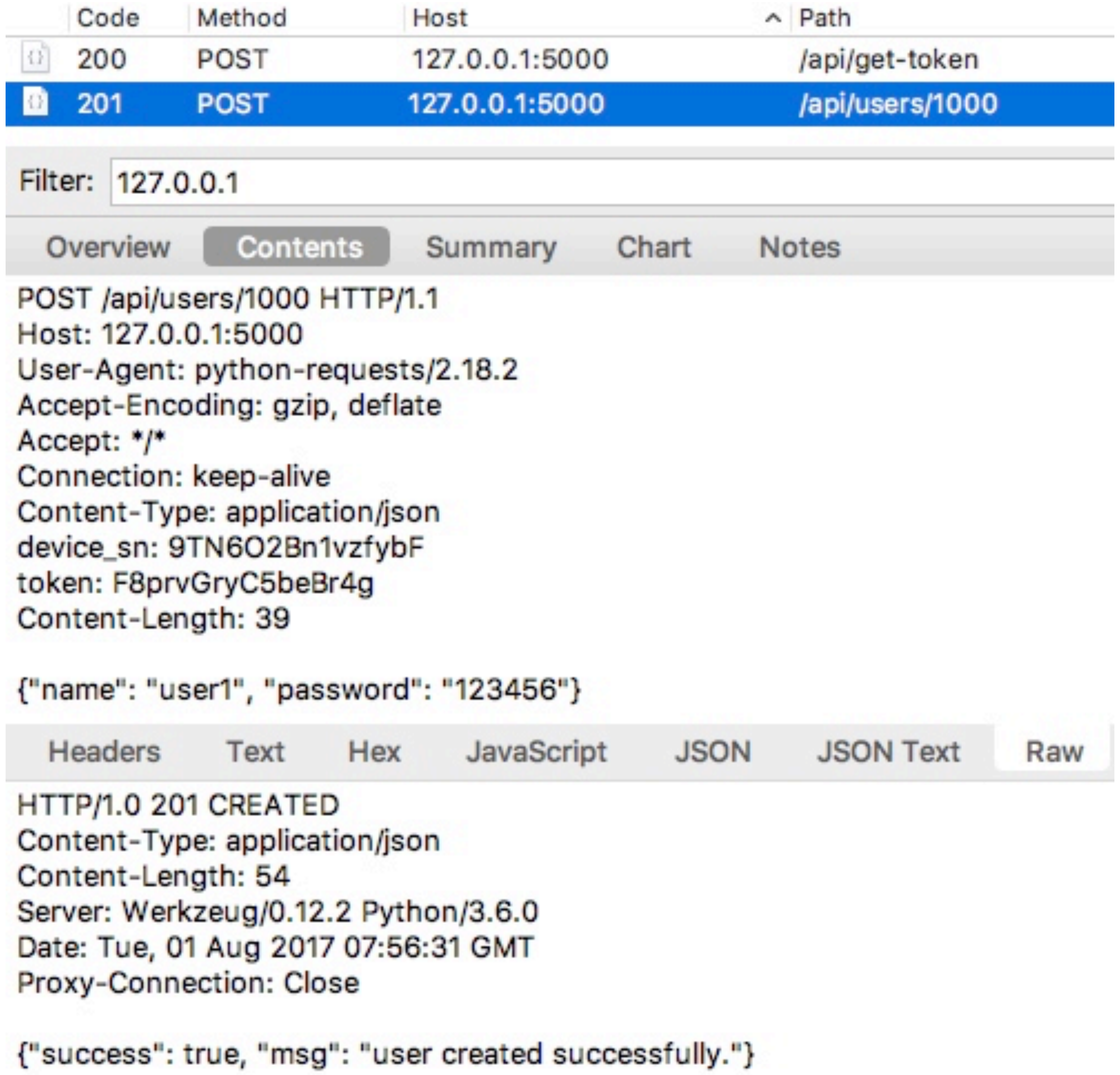


Code	Method	Host	Path
200	POST	127.0.0.1:5000	/api/get-token
201	POST	127.0.0.1:5000	/api/users/1000

Filter: 127.0.0.1

```
POST /api/get-token HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: python-requests/2.18.2
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Type: application/json
device_sn: 9TN6O2Bn1vzfybF
user_agent: iOS/10.3
os_platform: ios
app_version: 2.8.6
Content-Length: 52

{"sign": "19067cf712265eb5426db8d3664026c1ccea02b9"}
```

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 46
Server: Werkzeug/0.12.2 Python/3.6.0
Date: Tue, 01 Aug 2017 07:56:31 GMT
Proxy-Connection: Close

{"success": true, "token": "F8prvGryC5beBr4g"}
```



Code	Method	Host	Path
200	POST	127.0.0.1:5000	/api/get-token
201	POST	127.0.0.1:5000	/api/users/1000

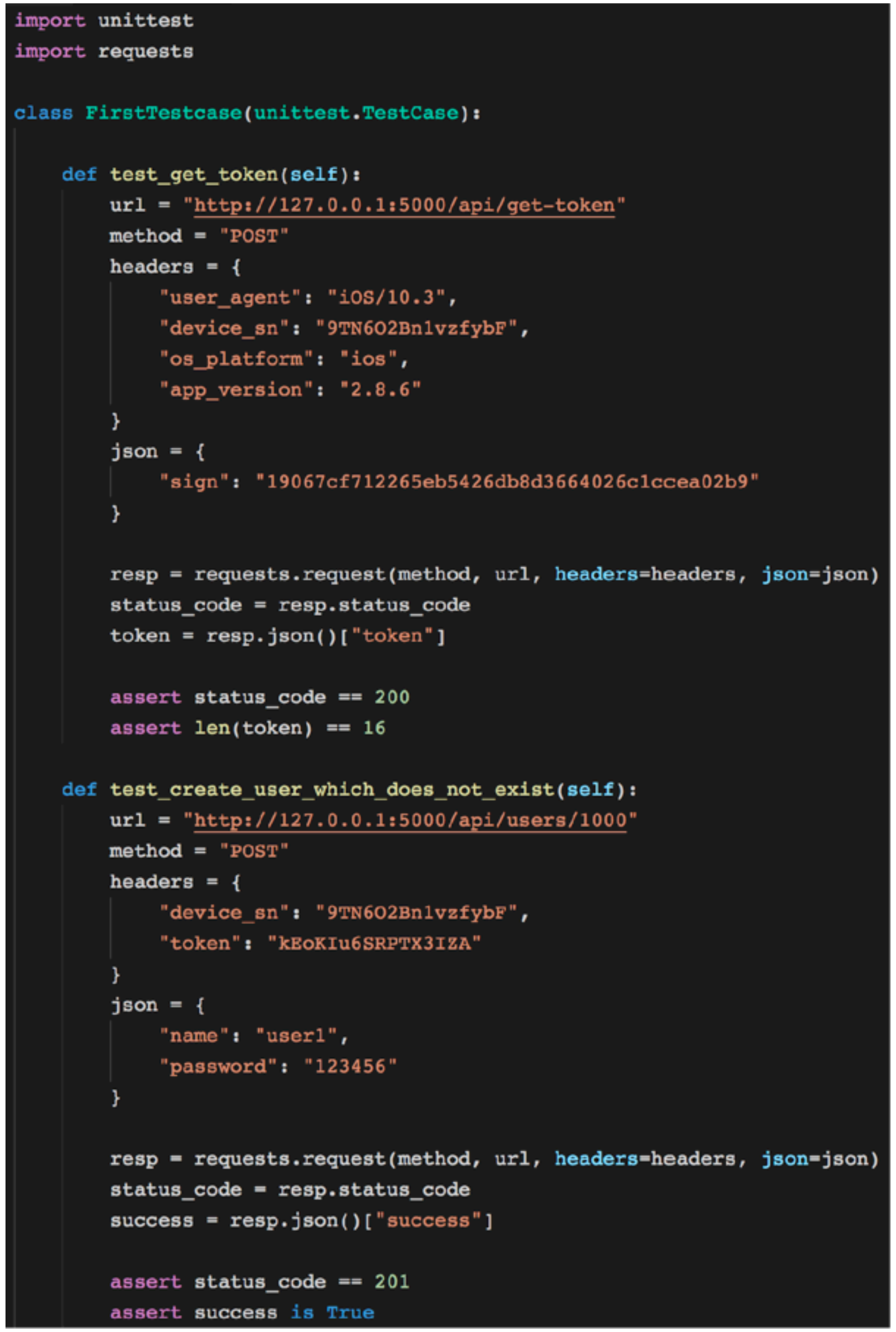
Filter: 127.0.0.1

```
POST /api/users/1000 HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: python-requests/2.18.2
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Type: application/json
device_sn: 9TN6O2Bn1vzfybF
token: F8prvGryC5beBr4g
Content-Length: 39

{"name": "user1", "password": "123456"}
```

```
HTTP/1.0 201 CREATED
Content-Type: application/json
Content-Length: 54
Server: Werkzeug/0.12.2 Python/3.6.0
Date: Tue, 01 Aug 2017 07:56:31 GMT
Proxy-Connection: Close

{"success": true, "msg": "user created successfully."}
```

案例简介

核心特性 - 代码示例



```python
import unittest
import requests

class FirstTestcase(unittest.TestCase):

    def test_get_token(self):
        url = "http://127.0.0.1:5000/api/get-token"
        method = "POST"
        headers = {
            "user_agent": "iOS/10.3",
            "device_sn": "9TN6O2Bn1vzfybF",
            "os_platform": "ios",
            "app_version": "2.8.6"
        }
        json = {
            "sign": "19067cf712265eb5426db8d3664026c1ccea02b9"
        }

        resp = requests.request(method, url, headers=headers, json=json)
        status_code = resp.status_code
        token = resp.json()["token"]

        assert status_code == 200
        assert len(token) == 16

    def test_create_user_which_does_not_exist(self):
        url = "http://127.0.0.1:5000/api/users/1000"
        method = "POST"
        headers = {
            "device_sn": "9TN6O2Bn1vzfybF",
            "token": "kEoKIu6SRPTX3IZA"
        }
        json = {
            "name": "user1",
            "password": "123456"
        }

        resp = requests.request(method, url, headers=headers, json=json)
        status_code = resp.status_code
        success = resp.json()["success"]

        assert status_code == 201
        assert success is True
```

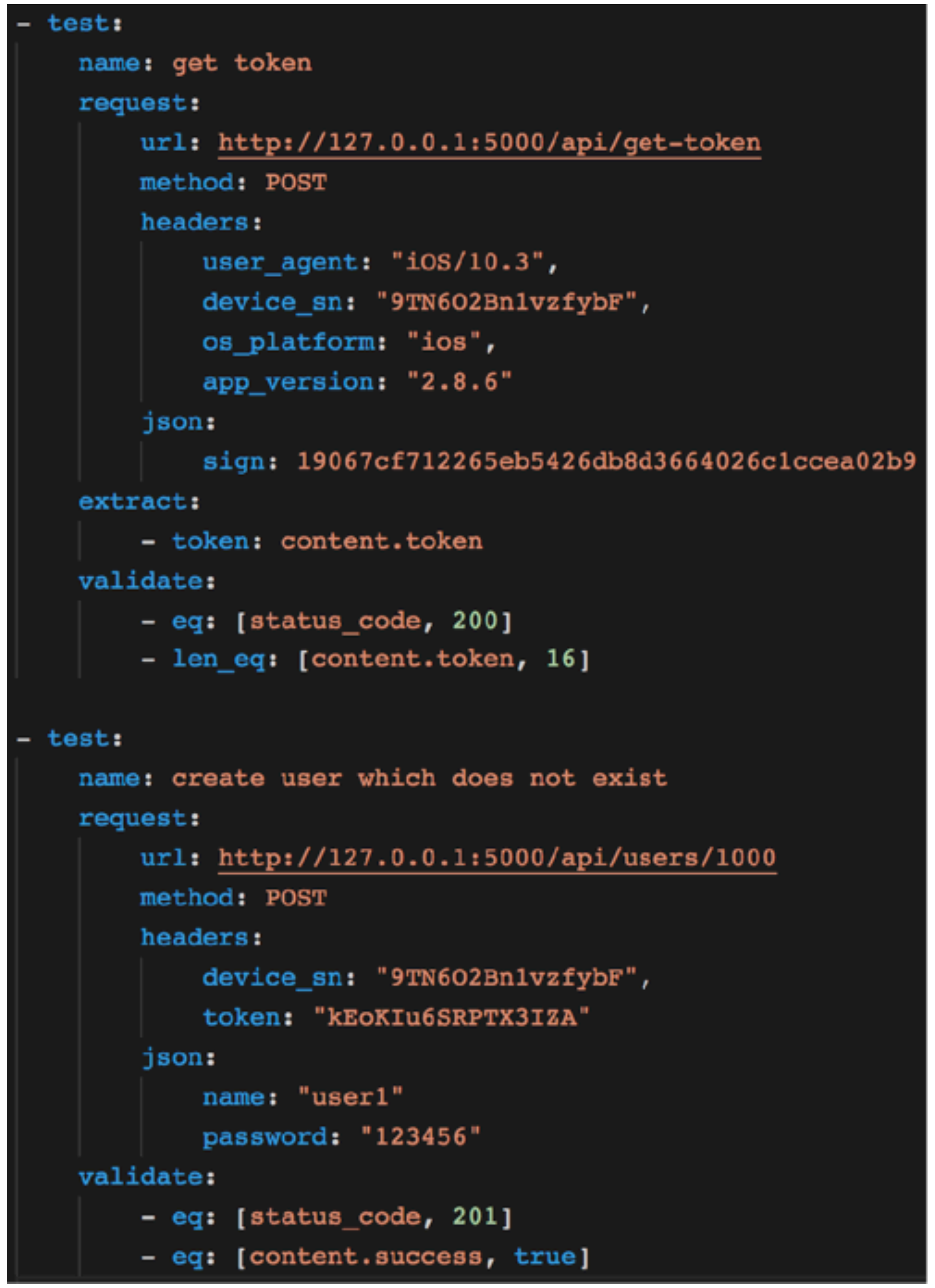
信息量等价

重复

迥异

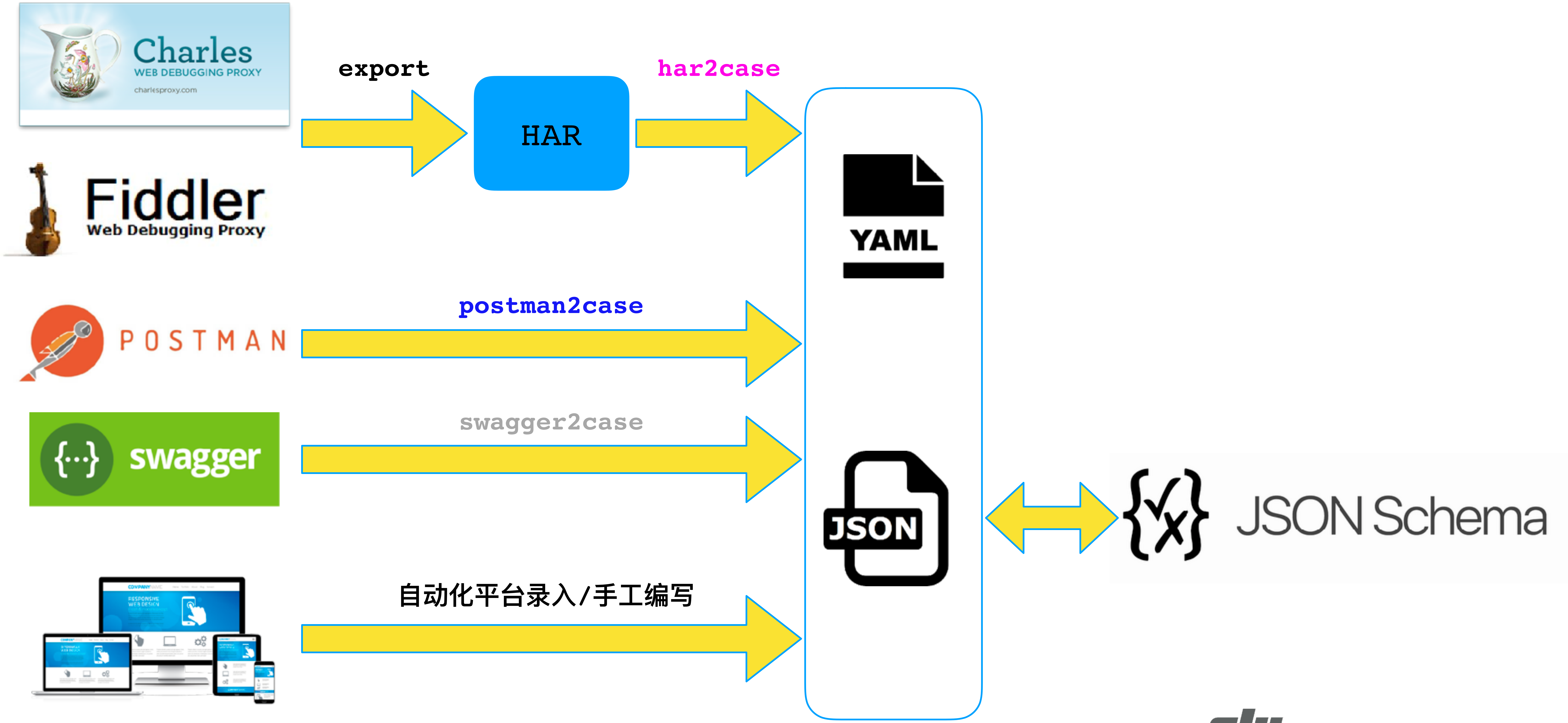
简洁

规范



```yaml
- test:
    name: get token
    request:
        url: http://127.0.0.1:5000/api/get-token
        method: POST
        headers:
            user_agent: "iOS/10.3",
            device_sn: "9TN6O2Bn1vzfybF",
            os_platform: "ios",
            app_version: "2.8.6"
        json:
            sign: 19067cf712265eb5426db8d3664026c1ccea02b9
    extract:
        - token: content.token
    validate:
        - eq: [status_code, 200]
        - len_eq: [content.token, 16]

- test:
    name: create user which does not exist
    request:
        url: http://127.0.0.1:5000/api/users/1000
        method: POST
        headers:
            device_sn: "9TN6O2Bn1vzfybF",
            token: "kEoKIu6SRPTX3IZA"
        json:
            name: "user1"
            password: "123456"
    validate:
        - eq: [status_code, 201]
        - eq: [content.success, true]
```

核心特性 - 测试用例生成



export

HAR

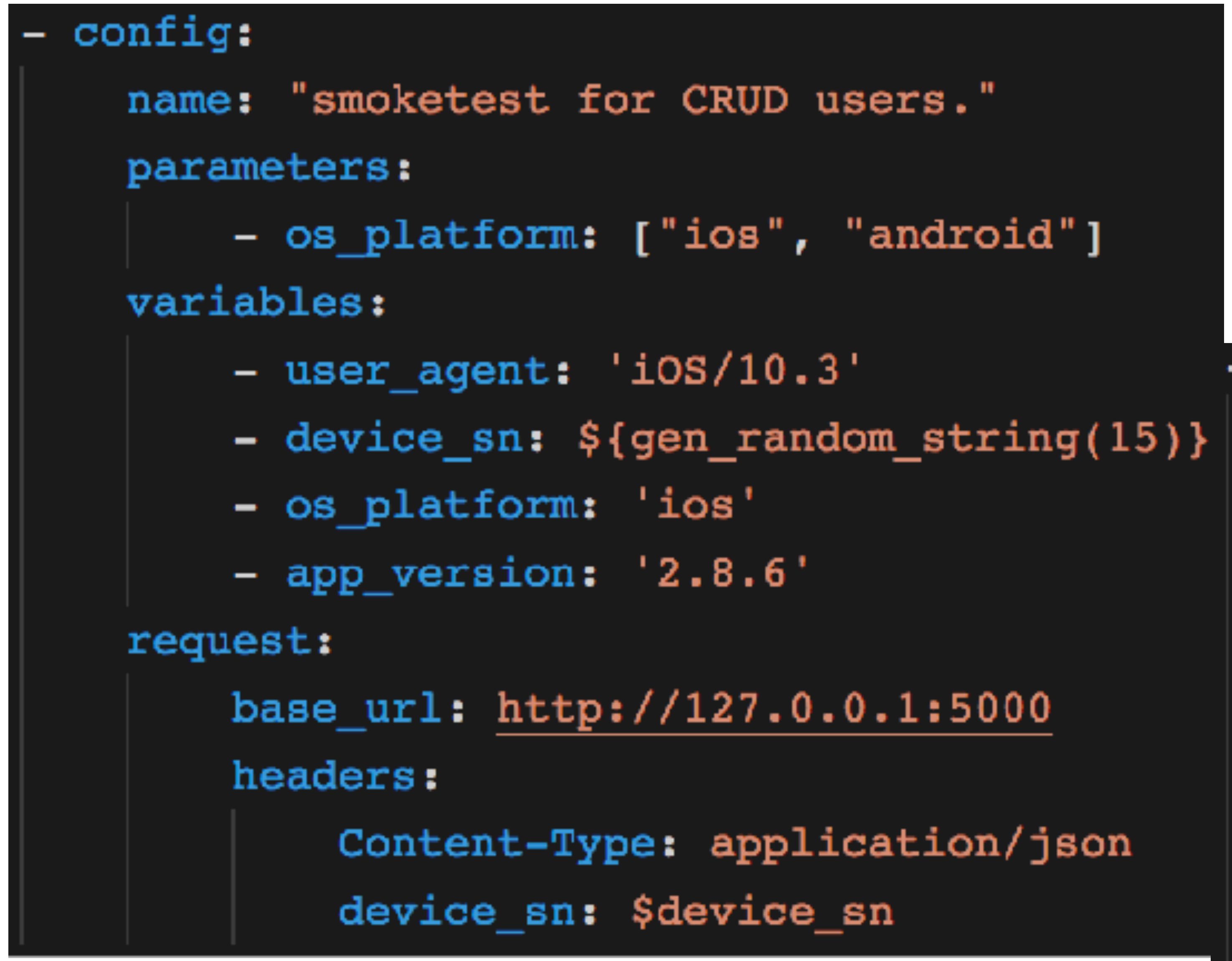
har2case

postman2case

swagger2case

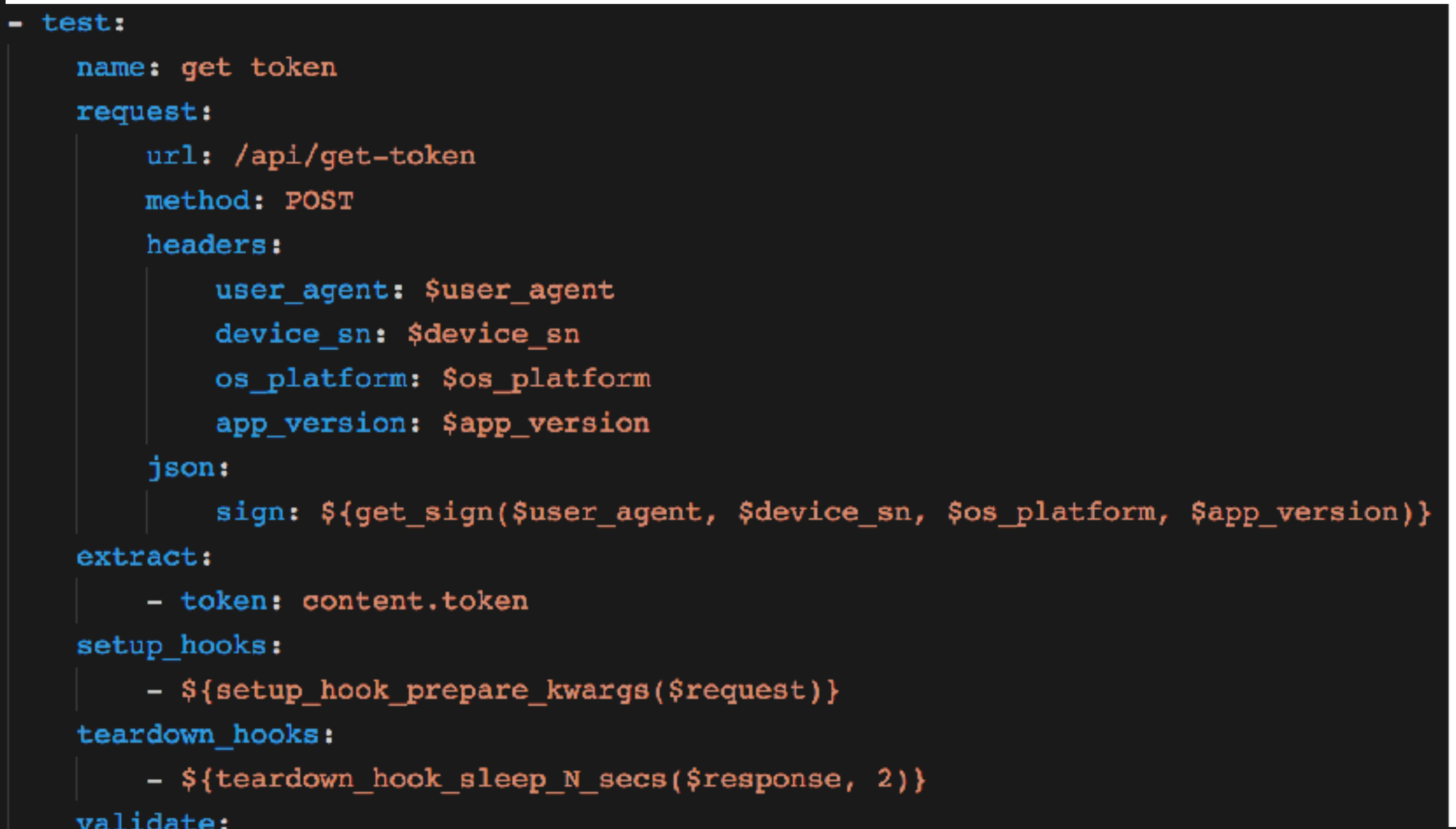
自动化平台录入/手工编写

核心特性 - 测试脚本优化



```
- config:
    name: "smoketest for CRUD users."
    parameters:
        - os_platform: ["ios", "android"]
    variables:
        - user_agent: 'iOS/10.3'
        - device_sn: ${gen_random_string(15)}
        - os_platform: 'ios'
        - app_version: '2.8.6'
    request:
        base_url: http://127.0.0.1:5000
        headers:
            Content-Type: application/json
            device_sn: $device_sn
```

用例集全局定义
config

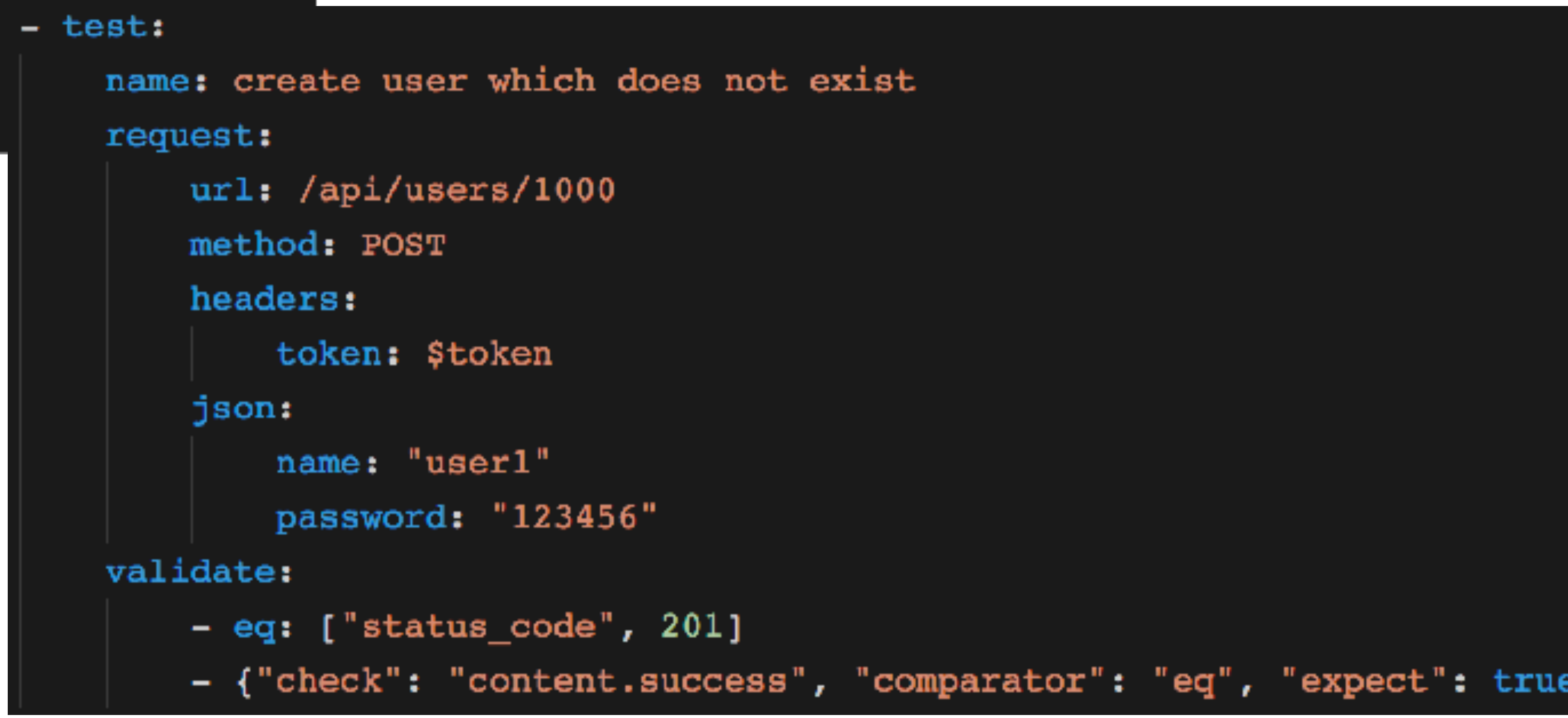


```
- test:
    name: get token
    request:
        url: /api/get-token
        method: POST
        headers:
            user_agent: $user_agent
            device_sn: $device_sn
            os_platform: $os_platform
            app_version: $app_version
        json:
            sign: ${get_sign($user_agent, $device_sn, $os_platform, $app_version)}
    extract:
        - token: content.token
    setup_hooks:
        - ${setup_hook_prepare_kwargs($request)}
    teardown_hooks:
        - ${teardown_hook_sleep_N_secs($response, 2)}
    validate:
        - {"check": "status_code", "comparator": "eq", "expect": 200}
        - len_eq: ["content.token", 16]
```

变量定义
variables

变量引用
$var

函数调用
${func($a, 9)}



```
- test:
    name: create user which does not exist
    request:
        url: /api/users/1000
        method: POST
        headers:
            token: $token
        json:
            name: "user1"
            password: "123456"
    validate:
        - eq: ["status_code", 201]
        - {"check": "content.success", "comparator": "eq", "expect": true
```

结果提取
extract

结果校验
validate

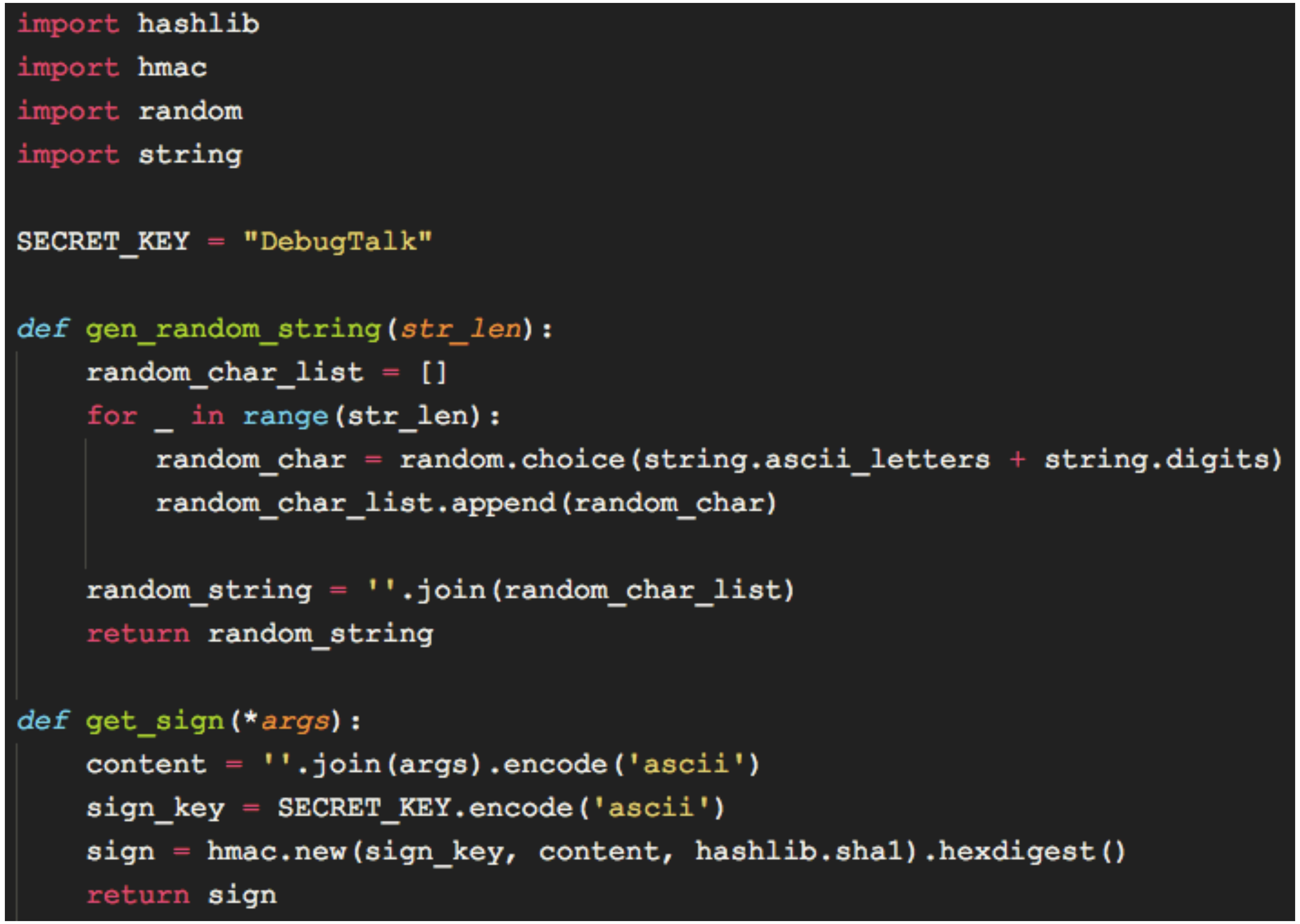
数据驱动
parameters

setup_hooks

teardown_hooks

核心特性 - 测试脚本优化

函数定义
debugtalk.py



```python
import hashlib
import hmac
import random
import string

SECRET_KEY = "DebugTalk"

def gen_random_string(str_len):
    random_char_list = []
    for _ in range(str_len):
        random_char = random.choice(string.ascii_letters + string.digits)
        random_char_list.append(random_char)

    random_string = ''.join(random_char_list)
    return random_string

def get_sign(*args):
    content = ''.join(args).encode('ascii')
    sign_key = SECRET_KEY.encode('ascii')
    sign = hmac.new(sign_key, content, hashlib.sha1).hexdigest()
    return sign
```

核心特性 - 逻辑流程图

测试用例准备

HAR Charles/Fiddler | Postman collection

record & generate

JSON | YAML

Web

测试用例加载

file loader | api | testcase | suite

extend & merge

parameterize

JSON Scheme Validate

Standard JSON data

用例集初始化

Runner() | Context() | HttpSession()

unittest | TestCase | TestSuite | TextTestRunner

测试执行

testset setup hooks

testcase iteration

check if skip

prepare testcase

testcase setup hooks

request

make ResponseObject

testcase teardown hooks

extract response

validate

testset teardown hooks

结果汇总

aggregate results

生成测试报告

default template | custom template

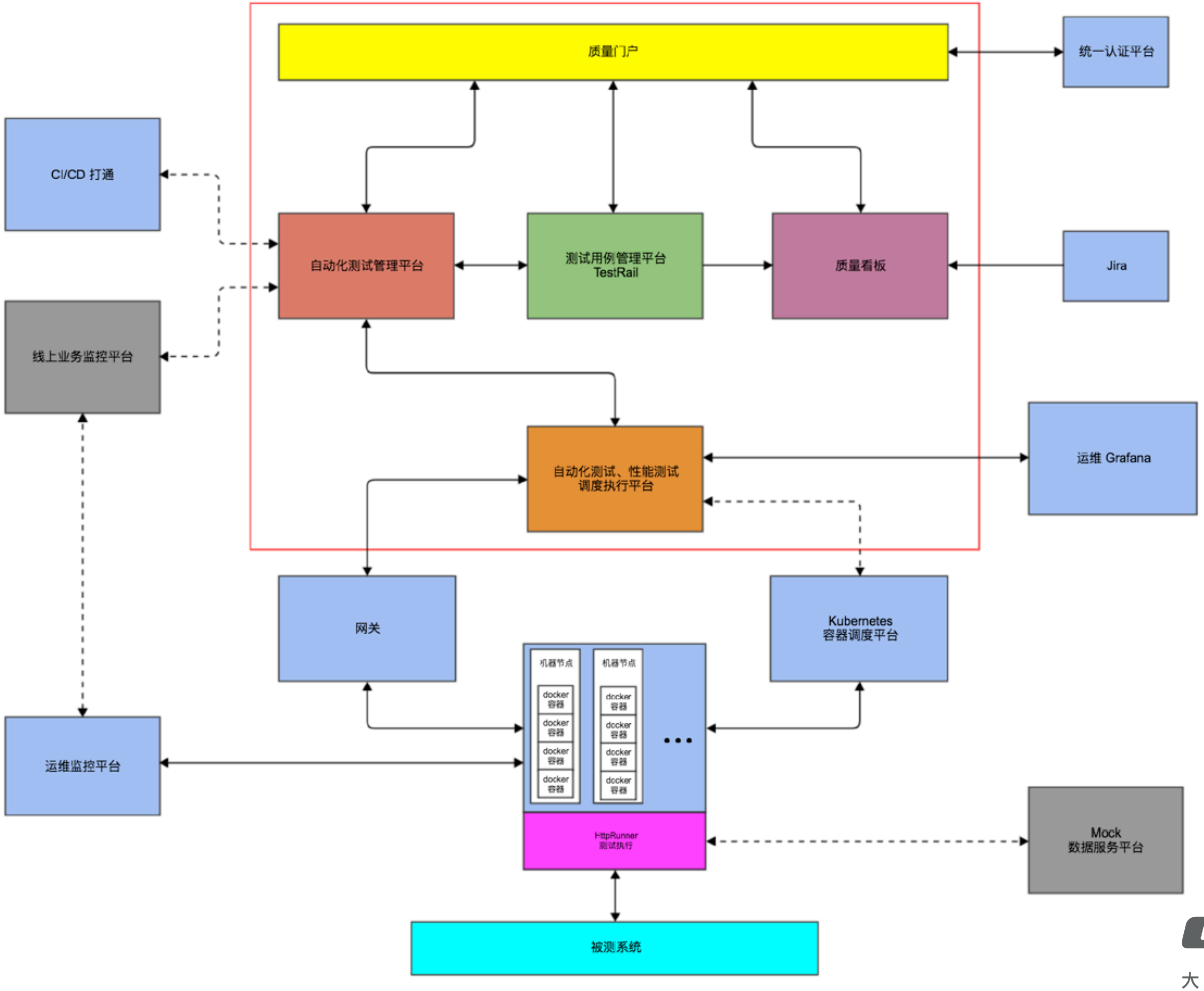
generate html report

5、大疆内部实践案例

大疆内部实践案例

- 整体架构图
- 自动化测试平台
- 打通功能测试平台（TestRail）
- 性能测试平台

大疆内部实践案例 - 整体架构图



质量门户

统一认证平台

CI/CD 打通

自动化测试管理平台

测试用例管理平台
TestRail

质量看板

Jira

线上业务监控平台

自动化测试、性能测试
调度执行平台

运维 Grafana

网关

Kubernetes
容器调度平台

机器节点

docker
容器

docker
容器

docker
容器

docker
容器

机器节点

docker
容器

docker
容器

docker
容器

docker
容器

• • •

HttpRunner
测试执行

运维监控平台

Mock
数据服务平台

被测系统

大疆内部实践案例 - 自动化测试平台

欢迎您：leo.lee
首页
项目列表
任务列表
DJI官网
接口定义
测试用例
测试场景
测试数据
测试环境
测试脚本
隐私变量
发送
保存

接口修改

common

项目名称	DJI官网
接口描述	current_time
接口定义	test_current_time
接口参数	方法参数,$xxx

GET /api/util/current_time data

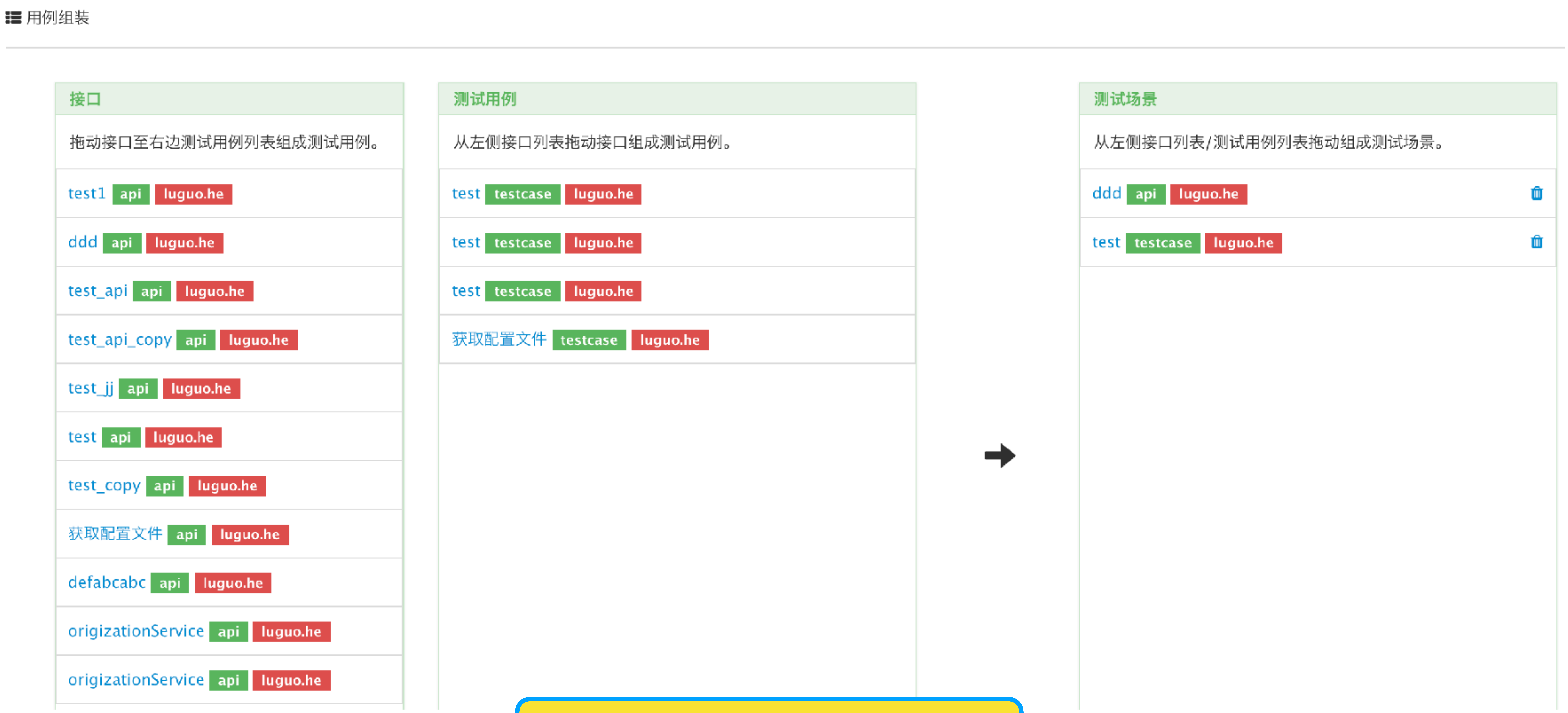
data

Key	Type	Value
	string	

测试结果	测试通过
request_header	{ Accept-Encoding: "gzip, deflate", Connection: "keep-alive", authorization: "Basic d3d3LXJlbGVhc2U6WGdWZTlZZ2ZGRUxkN0M0YWZqajZO", Accept: "*/*", user-agent: "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36" }
request_body	null
response_header	{ 26 items }
response_body	{"status":200,"extra":{},"success":true,"data":[{"time":1531292562070}]}
trace	

在线调试接口

大疆内部实践案例 - 自动化测试平台



用例组装

接口

拖动接口至右边测试用例列表组成测试用例。

- test1 api luguo.he
- ddd api luguo.he
- test_api api luguo.he
- test_api_copy api luguo.he
- test_jj api luguo.he
- test api luguo.he
- test_copy api luguo.he
- 获取配置文件 api luguo.he
- defabcabc api luguo.he
- origizationService api luguo.he
- origizationService api luguo.he

测试用例

从左侧接口列表拖动接口组成测试用例。

- test testcase luguo.he
- test testcase luguo.he
- test testcase luguo.he
- 获取配置文件 testcase luguo.he

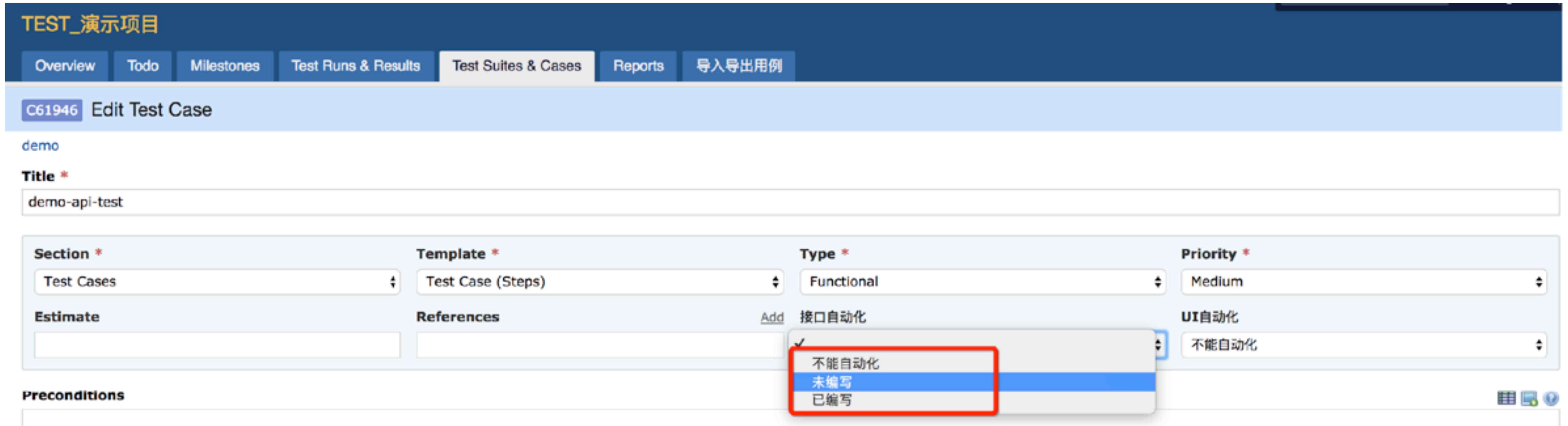
测试场景

从左侧接口列表/测试用例列表拖动组成测试场景。

- ddd api luguo.he
- test testcase luguo.he

拖拽组装生成测试场景

大疆内部实践案例 - 打通功能测试平台（TestRail）



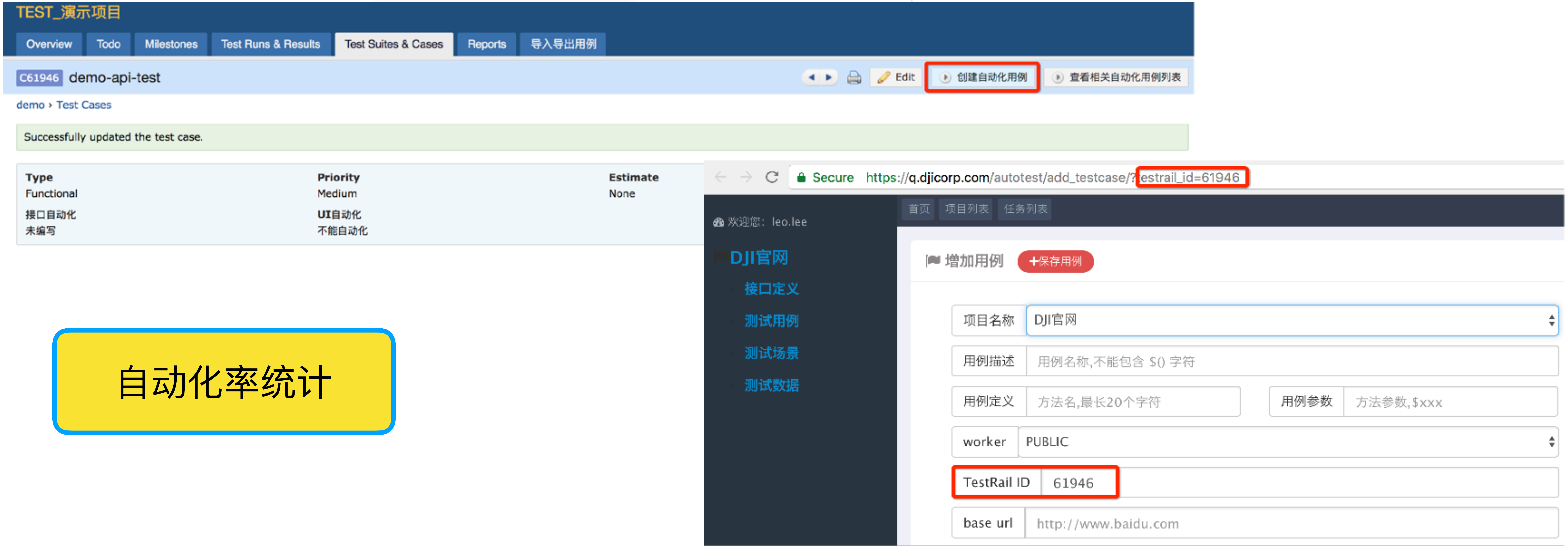
C61946 Edit Test Case

demo

Title *	demo-api-test
Section *	Test Cases
Template *	Test Case (Steps)
Type *	Functional
Priority *	Medium
Estimate	
References	
接口自动化	不能自动化 未编写 已编写
UI自动化	不能自动化

Preconditions

关联功能测试用例和自动化测试用例



C61946 demo-api-test

demo › Test Cases

Successfully updated the test case.

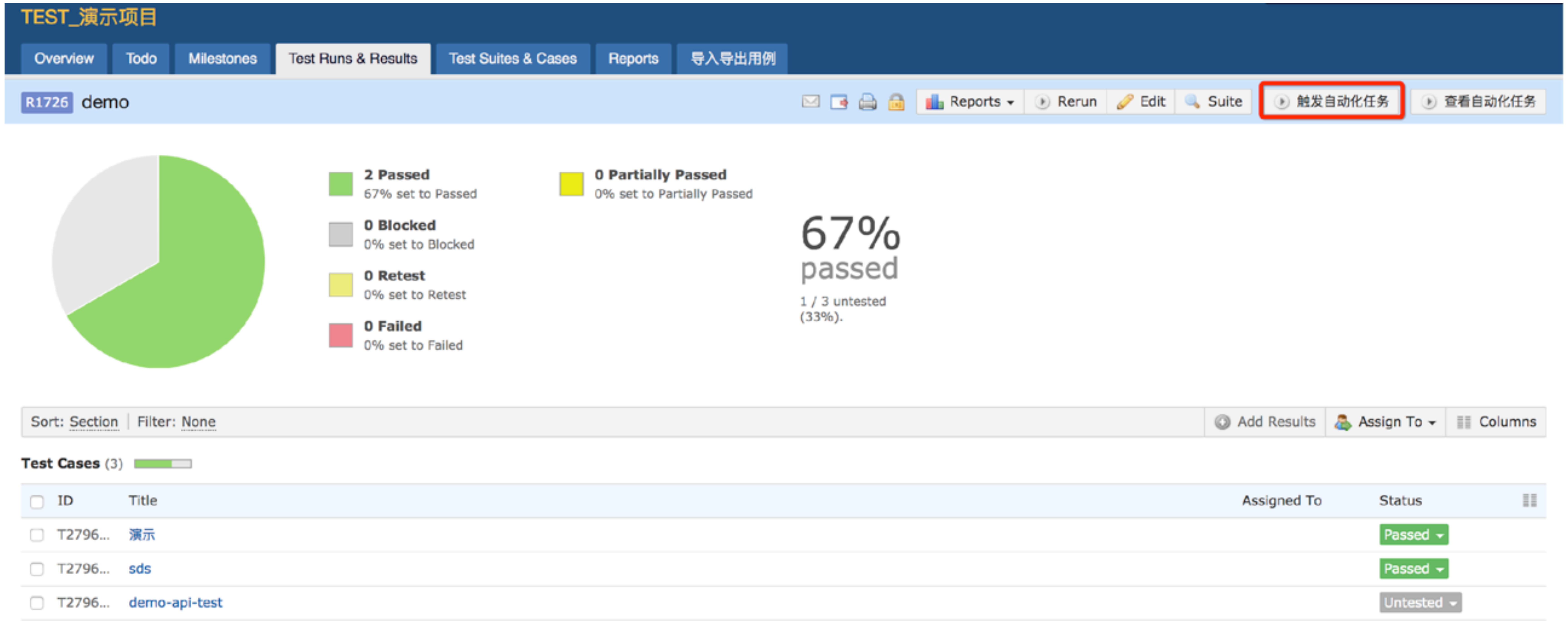
Type	Priority	Estimate
Functional	Medium	None
接口自动化	UI自动化	
未编写	不能自动化	

Secure https://q.djicorp.com/autotest/add_testcase/?testrail_id=61946

项目名称	DJI官网		
用例描述	用例名称,不能包含 $() 字符		
用例定义	方法名,最长20个字符	用例参数	方法参数,$xxx
worker	PUBLIC		
TestRail ID	61946		
base url	http://www.baidu.com		

自动化率统计

大疆内部实践案例 - 打通功能测试平台（TestRail）



2 Passed
67% set to Passed

0 Partially Passed
0% set to Partially Passed

0 Blocked
0% set to Blocked

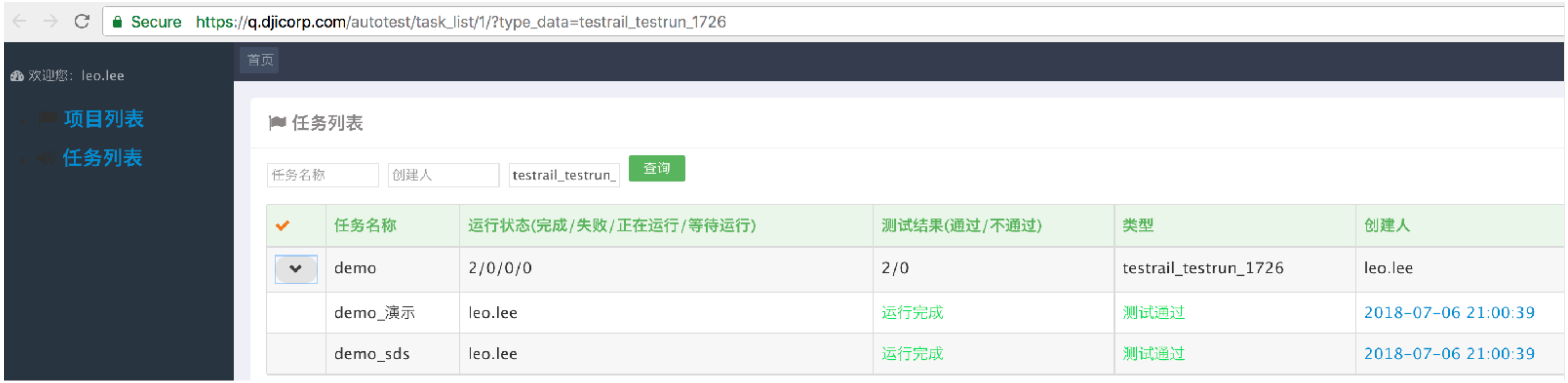
0 Retest
0% set to Retest

0 Failed
0% set to Failed

67% passed
1 / 3 untested (33%).

Test Cases (3)

ID	Title	Assigned To	Status
T2796...	演示		Passed
T2796...	sds		Passed
T2796...	demo-api-test		Untested


Secure
https://q.djicorp.com/autotest/task_list/1/?type_data=testrail_testrun_1726
首页
欢迎您：leo.lee
项目列表
任务列表
任务名称
创建人
testrail_testrun_
查询

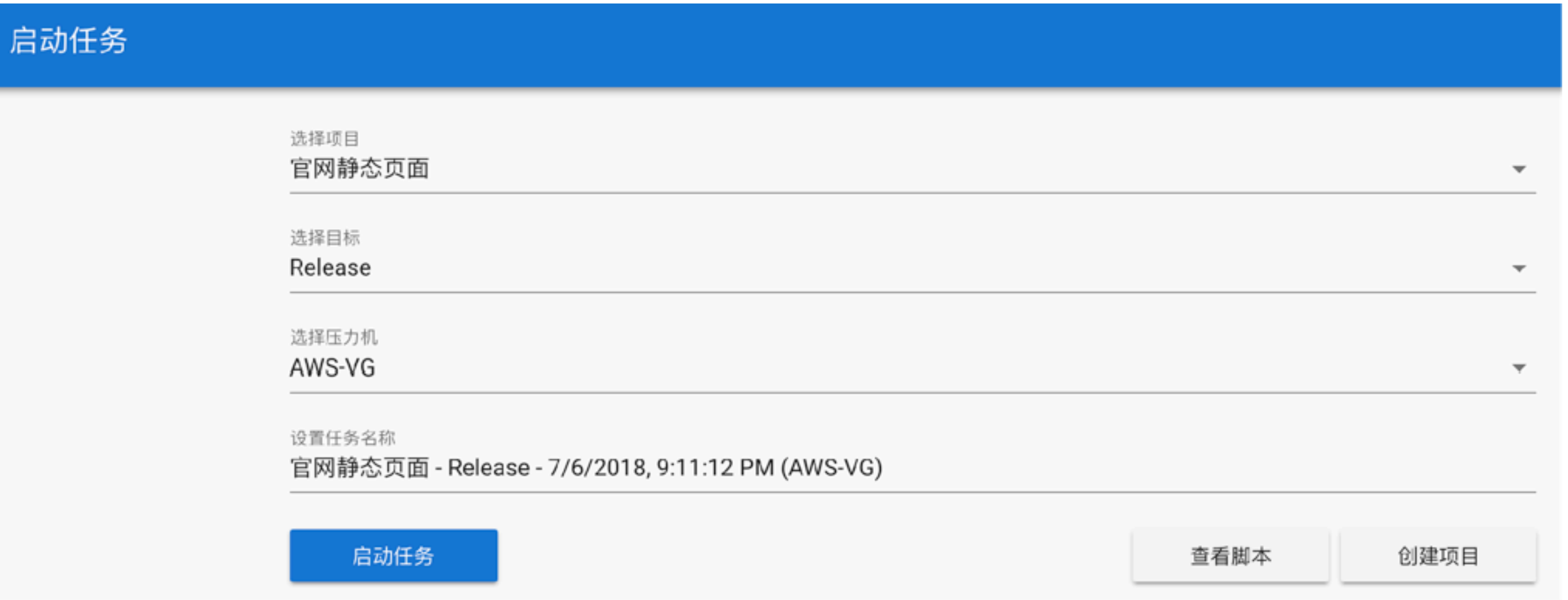
任务列表

✓	任务名称	运行状态(完成/失败/正在运行/等待运行)	测试结果(通过/不通过)	类型	创建人
	demo	2/0/0/0	2/0	testrail_testrun_1726	leo.lee
	demo_演示	leo.lee	运行完成	测试通过	2018-07-06 21:00:39
	demo_sds	leo.lee	运行完成	测试通过	2018-07-06 21:00:39

触发执行

查看执行结果

大疆内部实践案例 - 性能测试平台



启动任务

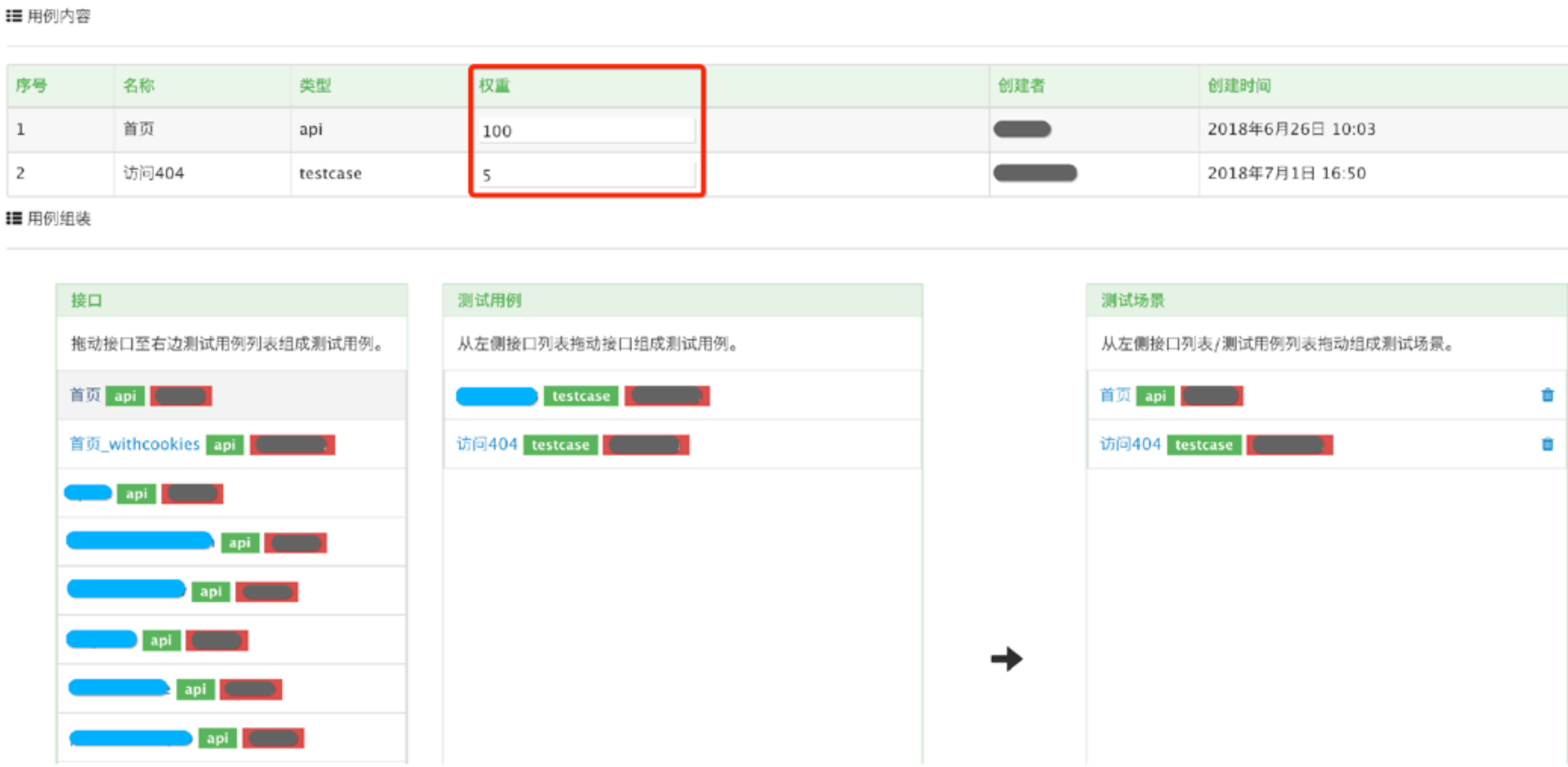
选择项目
官网静态页面

选择目标
Release

选择压力机
AWS-VG

设置任务名称
官网静态页面 - Release - 7/6/2018, 9:11:12 PM (AWS-VG)

查看性能测试任务对应的自动化测试用例



用例内容

序号	名称	类型	权重	创建者	创建时间
1	首页	api	100	[redacted]	2018年6月26日 10:03
2	访问404	testcase	5	[redacted]	2018年7月1日 16:50

用例组装

接口

拖动接口至右边测试用例列表组成测试用例。

- 首页 api [redacted]
- 首页_withcookies api [redacted]
- [redacted] api [redacted]
- [redacted] api [redacted]
- [redacted] api [redacted]
- [redacted] api [redacted]
- [redacted] api [redacted]
- [redacted] api [redacted]

测试用例

从左侧接口列表拖动接口组成测试用例。

- [redacted] testcase [redacted]
- 访问404 testcase [redacted]

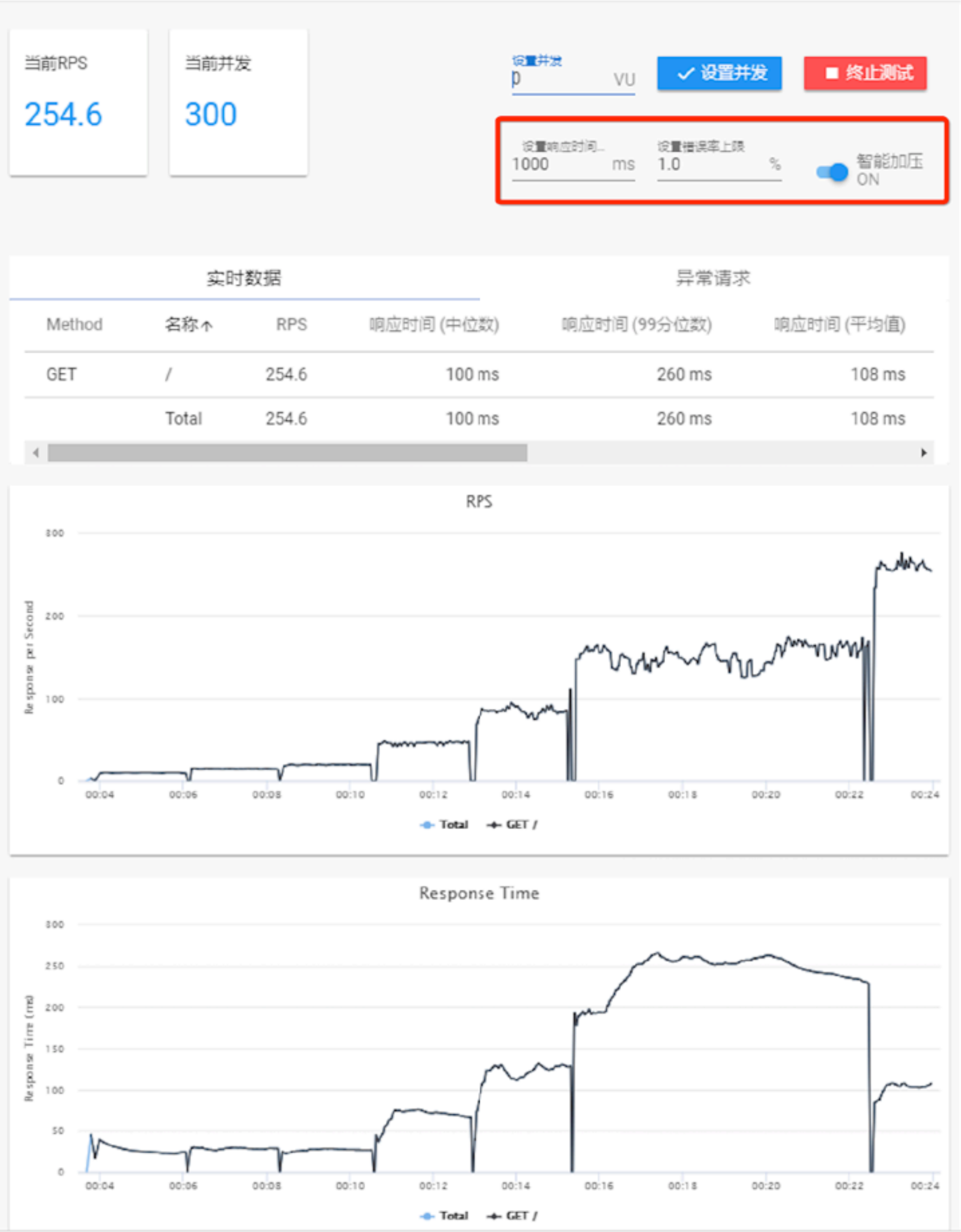
测试场景

从左侧接口列表/测试用例列表拖动组成测试场景。

- 首页 api [redacted]
- 访问404 testcase [redacted]

比例权重配置

大疆内部实践案例 - 性能测试平台



当前RPS 254.6

当前并发 300

设置并发 p VU

✓ 设置并发 ■ 终止测试

设置响应时间... 1000 ms

设置错误率上限 1.0 %

智能加压 ON

实时数据 异常请求

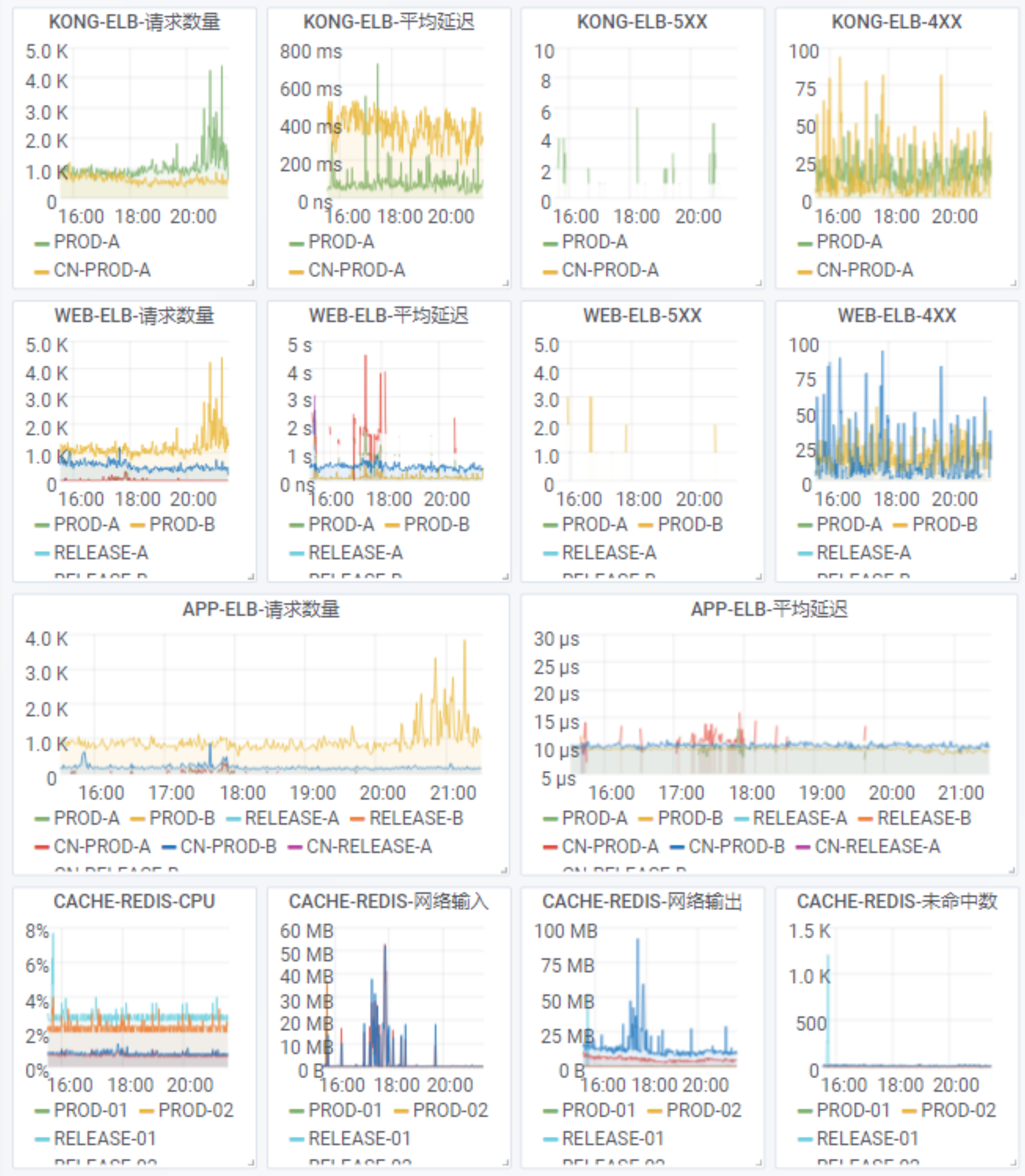
Method	名称 ↑	RPS	响应时间 (中位数)	响应时间 (99分位数)	响应时间 (平均值)
GET	/	254.6	100 ms	260 ms	108 ms
	Total	254.6	100 ms	260 ms	108 ms

智能加压

性能数据

Grafana

New Relic



Q
&
A

Thanks

MTSC2018

第四届中国移动互联网测试开发大会

dji
大疆创新

TesterHome