PYCON CHINA 2018

借助 Python 开源生态打造企业级自动化测试框架（HttpRunner）

演讲人：李隆（debugtalk）

主题概要

PYCON CHINA 2018

1、背景介绍

2、HttpRunner 设计思路

3、HttpRunner 核心特性

4、HttpRunner 实践案例

5、Q & A

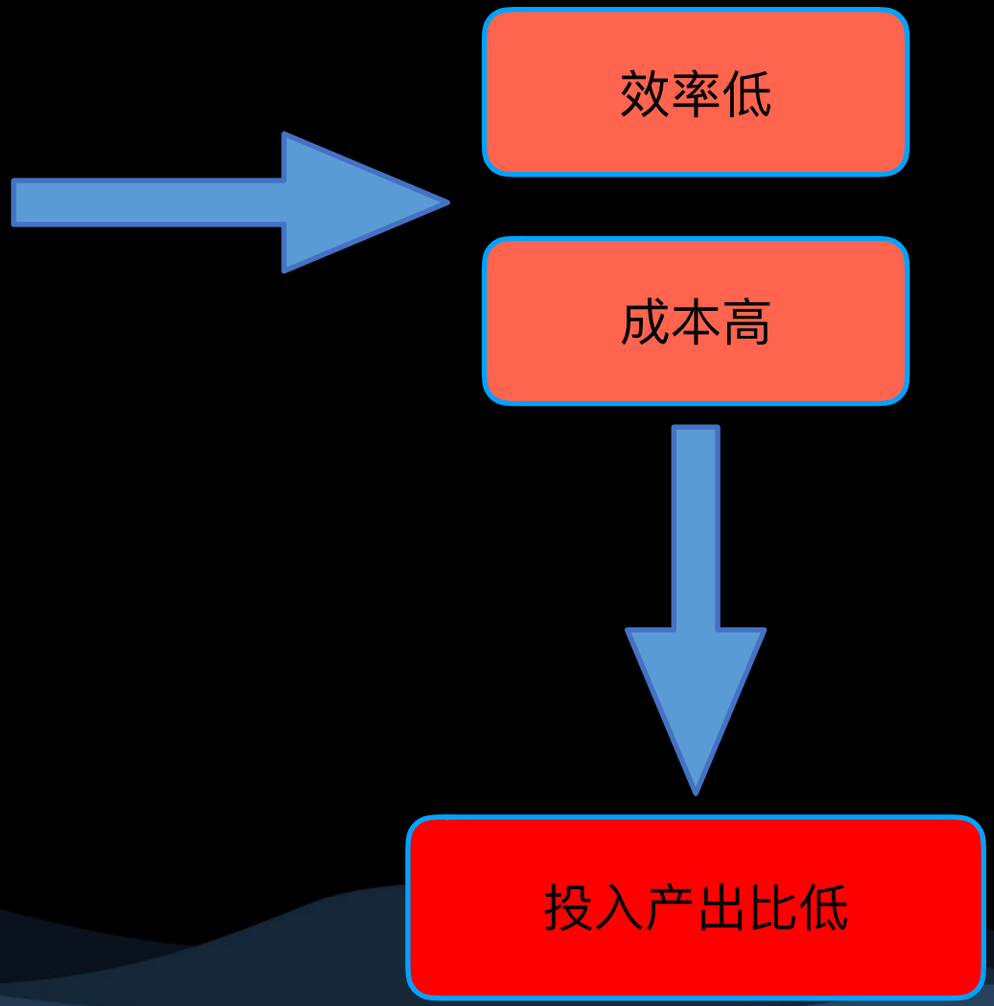
PYCON CHINA 2018

1、背景介绍

背景介绍

1、业务高速发展，需求迭代频繁

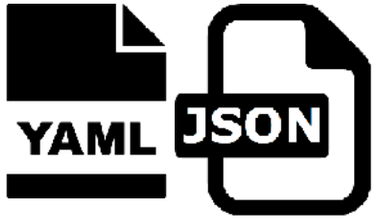
2、普遍具有功能回归测试、性能测试、持续集成、线上监控等需求



背景介绍

自动化测试 ？

基于 *Python* 的自动化测试生态



背景介绍

传统自动化测试方法普遍存在的问题：

工具多 & 杂

学习成本高

团队协作难

维护成本高

背景介绍

PYCON CHINA 2018

目标：提升投入产出比

少投入

- 工具开发 & 维护
- 学习使用成本
- 编写 & 录制测试用例
- 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

2、HttpRunner 设计思路

设计思路

PYCON CHINA 2018

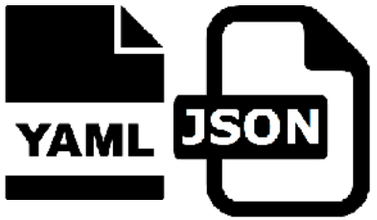
- 如何快速实现从0到1?
- 如何避免闭门造车?
- 如何兼具自动化、性能测试、持续集成、线上监控?
- 如何保障工具（框架）质量?
- 如何实现协作管理?
- 如何将框架设计得更优雅?

设计思路 - 如何快速实现从0到1?

充分复用开源项目

明显优势:

- 减少开发量
- 保障稳定性
- 降低学习成本



HttpRunner

设计思路 - 如何快速实现从0到1?

遵循行业标准

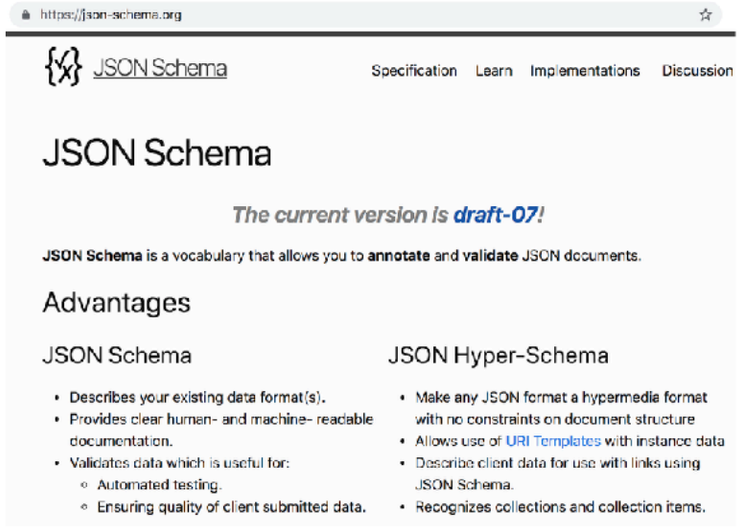
https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/HAR/Ov

HTTP Archive (HAR) format

Editor's Draft August 14, 2012

This version:
https://w3c.github.io/web-performance/specs/HAR/Overview.html
Latest version:
https://w3c.github.io/web-performance/specs/HAR/Overview.html
Latest Editor's Draft:
https://w3c.github.io/web-performance/specs/HAR/Overview.html
Editors:
Jan Odvarko, <hzoza@softwareishard.com>
Arvind Jain, Google Inc., <arvind@google.com>
Andy Davies, <andy@andydavies.me>

Copyright © 2012 W3C® (MIT, ERCIM, Keio, Beihang), All Rights Reserved.



https://json-schema.org

JSON Schema Specification Learn Implementations Discussion

JSON Schema

*The current version is **draft-07**!*

JSON Schema is a vocabulary that allows you to **annotate** and **validate** JSON documents.

Advantages

JSON Schema

- Describes your existing data format(s).
- Provides clear human- and machine- readable documentation.
- Validates data which is useful for:
  - Automated testing.
  - Ensuring quality of client submitted data.

JSON Hyper-Schema

- Make any JSON format a hypermedia format with no constraints on document structure
- Allows use of URI Templates with instance data
- Describe client data for use with links using JSON Schema.
- Recognizes collections and collection items.

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 Python object to send in the body of the **Request**.
- **headers** – (optional) Dictionary of HTTP Headers to send with the **Request**.
- **cookies** – (optional) Dict or CookieJar object to send with the **Request**.

设计思路 - 如何避免闭门造车?

拥抱开源生态

博客驱动开发

HttpRunner / HttpRunner

Unwatch 85 | Star 817 | Fork 325

Code | Issues 79 | Pull requests 1 | Projects 3 | Wiki | Insights | Settings

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

Edit

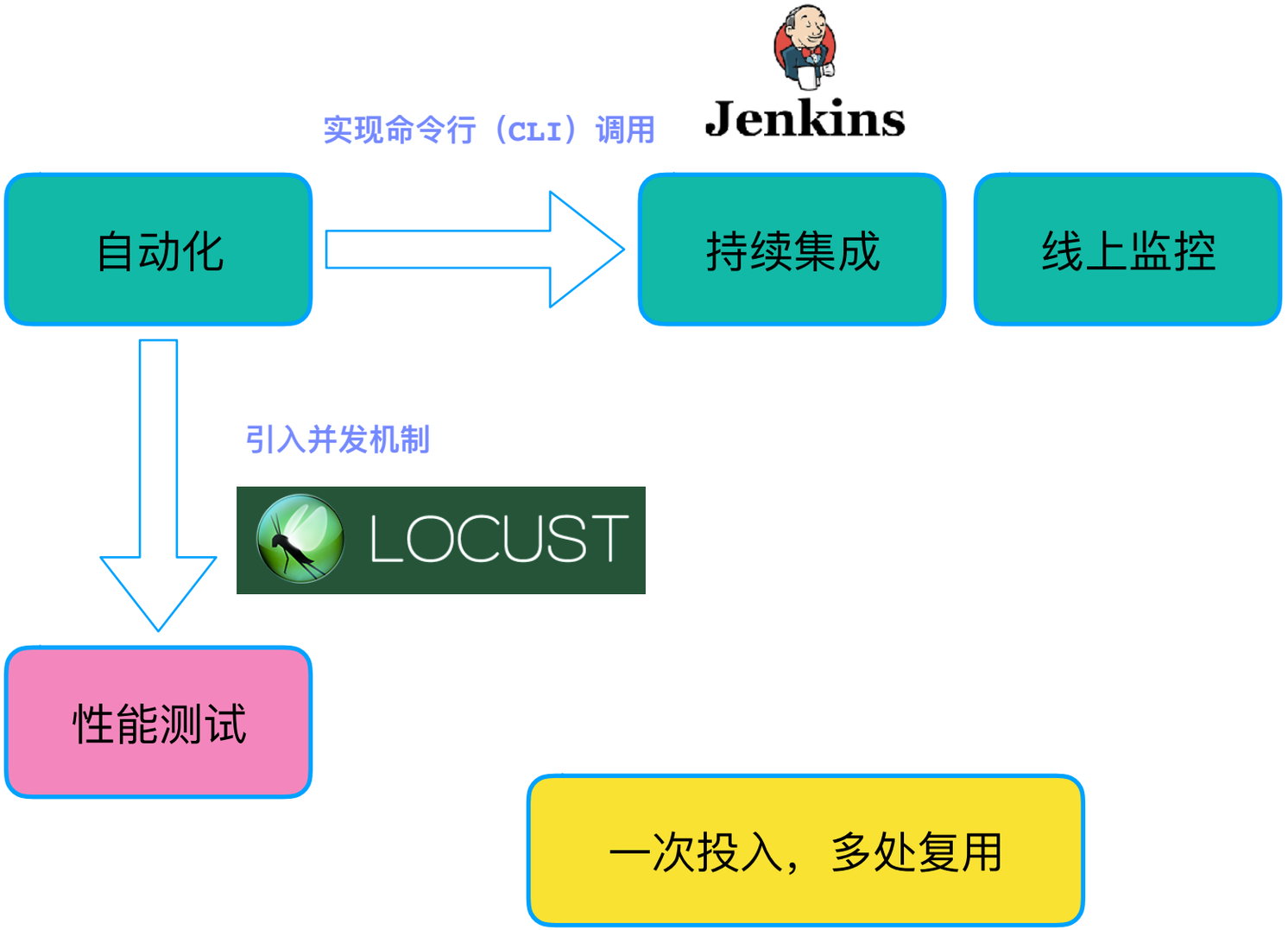
api-test locustio performance-testing locust httptest testrunner htmltestrunner load-testing Manage topics

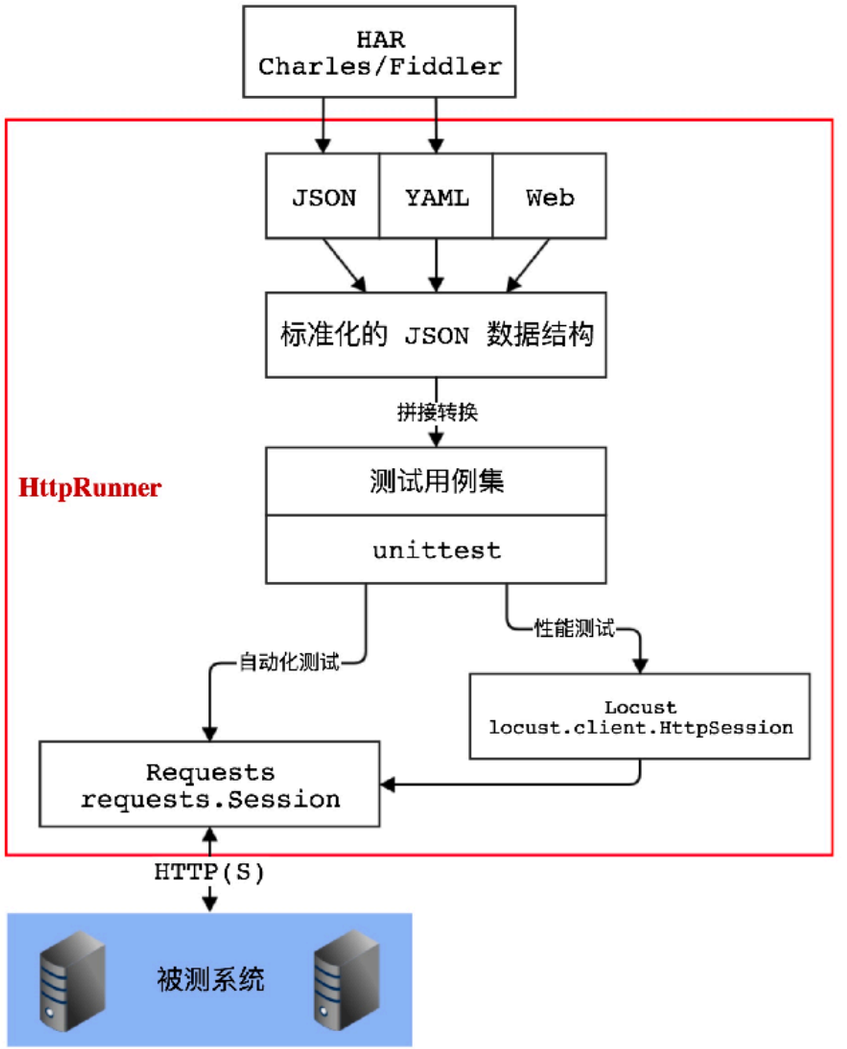
774 commits | 4 branches | 42 releases | 5 contributors | MIT

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、性能测试、持续集成、线上监控?



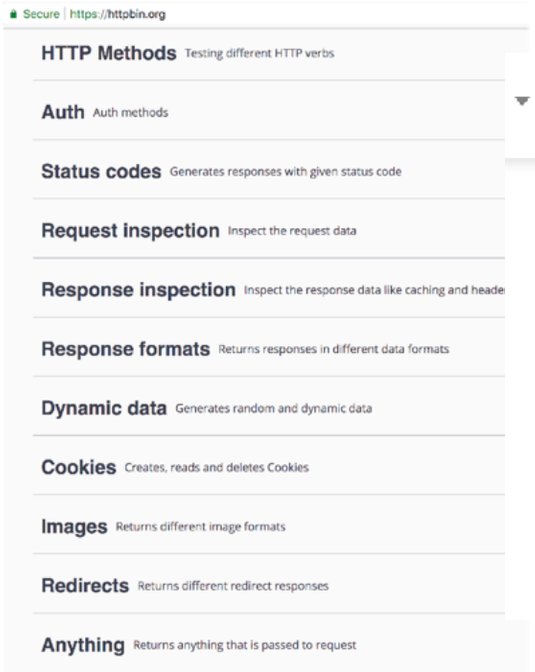


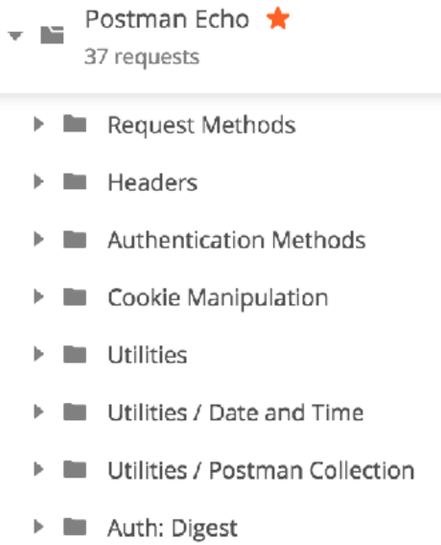
设计思路 - 如何保障工具质量?

测试驱动开发

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

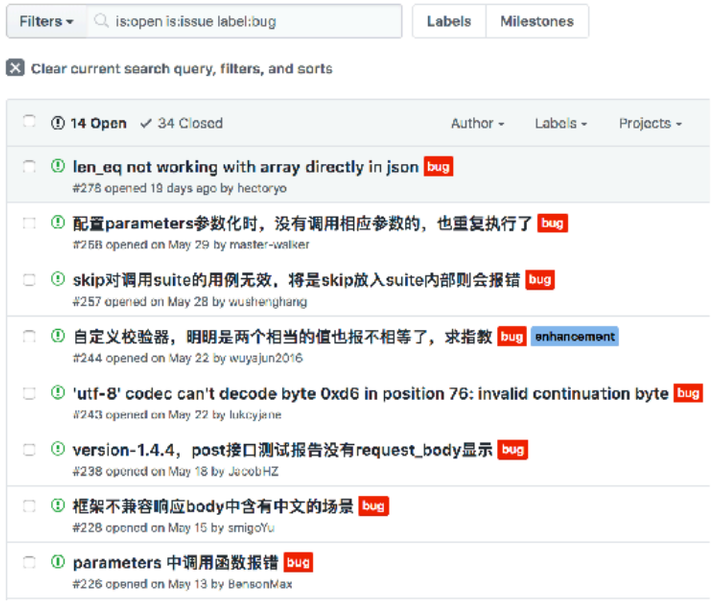
HttpRunner





开源社区反馈

- Github issues
- TesterHome



设计思路 - 如何实现协作管理?

约定大于配置

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


```
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 coll
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?

```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

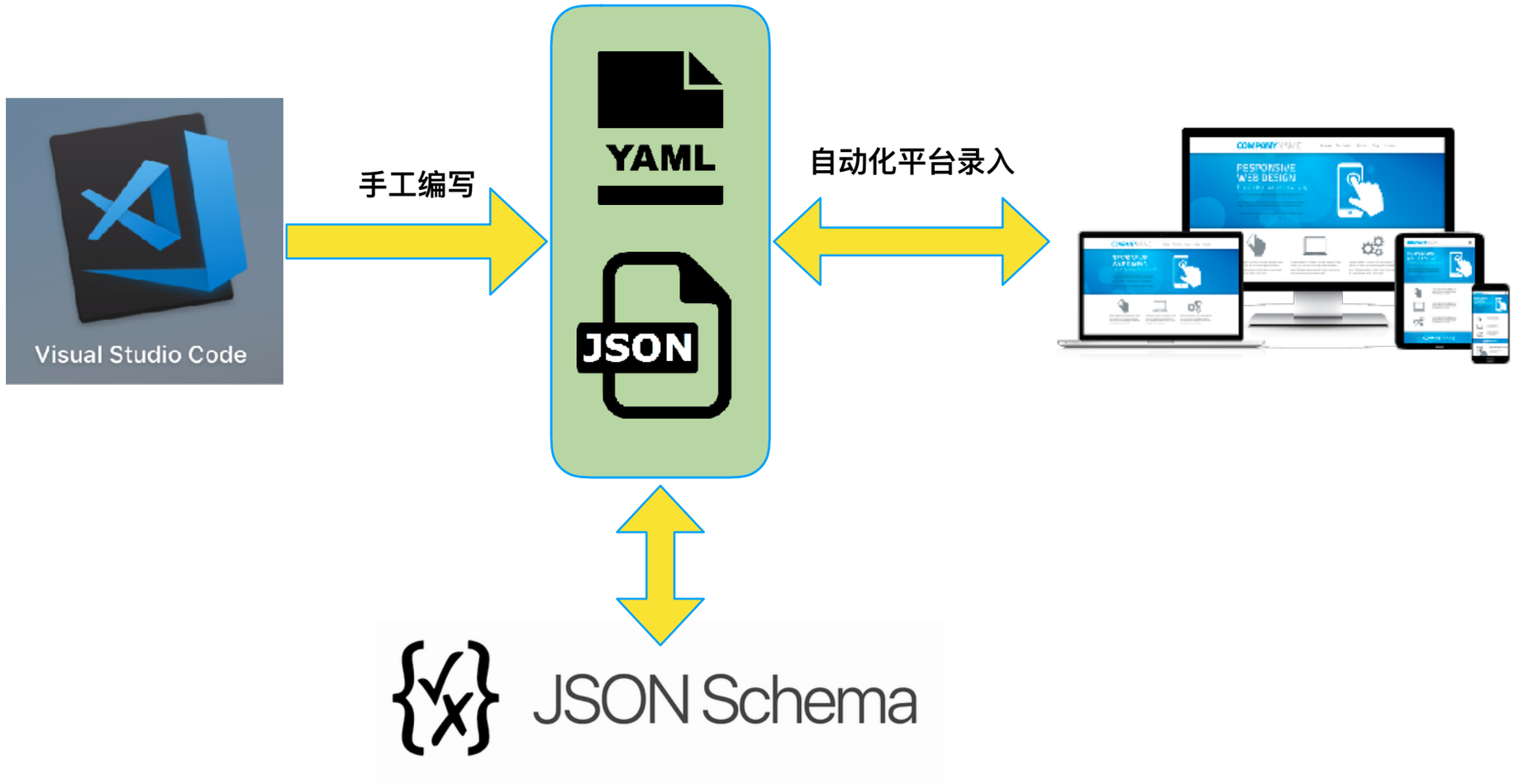
```yaml
- test:
    name: get token
    request:
        url: http://127.0.0.1:5000/api/get-token
        method: POST
        headers:
            user_agent: "iOS/10.3"
            device_sn: "9TN6O2Bn1vzfybF"
            os_platform: "ios"
            app_version: "2.8.6"
        json:
            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]
```

设计思路 - 如何实现协作管理?

脚本与平台无缝切换

- 统一存储和管理
- 平台服务化



设计思路 - 如何将框架设计得更优雅?

借鉴优秀（开源）项目

debugtalk.py

测试用例模板语言
(JSON/YAML)

参数化机制
(parameters)

环境变量管理（.env)

测试用例分层管理

设计思路 - 如何将框架设计得更优雅?

遵循 *Unix* 哲学

"Write programs that do one thing and do it well."
"Write programs to work together."

`prepare`

`initializer`

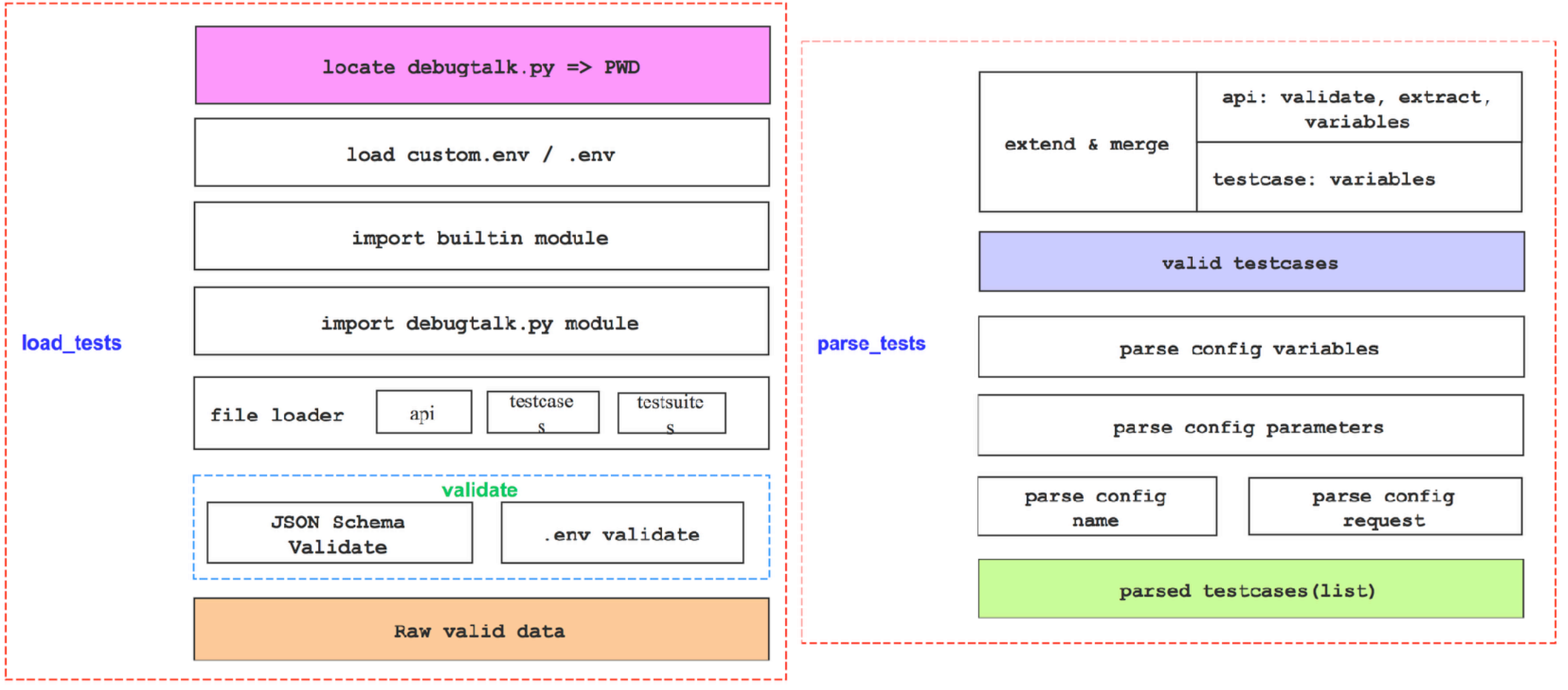
`loader`

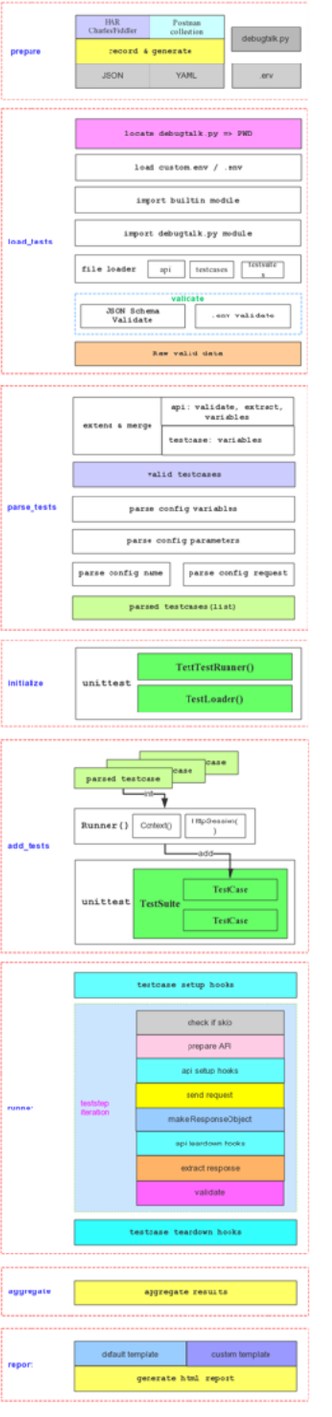
`runner`

`parser`

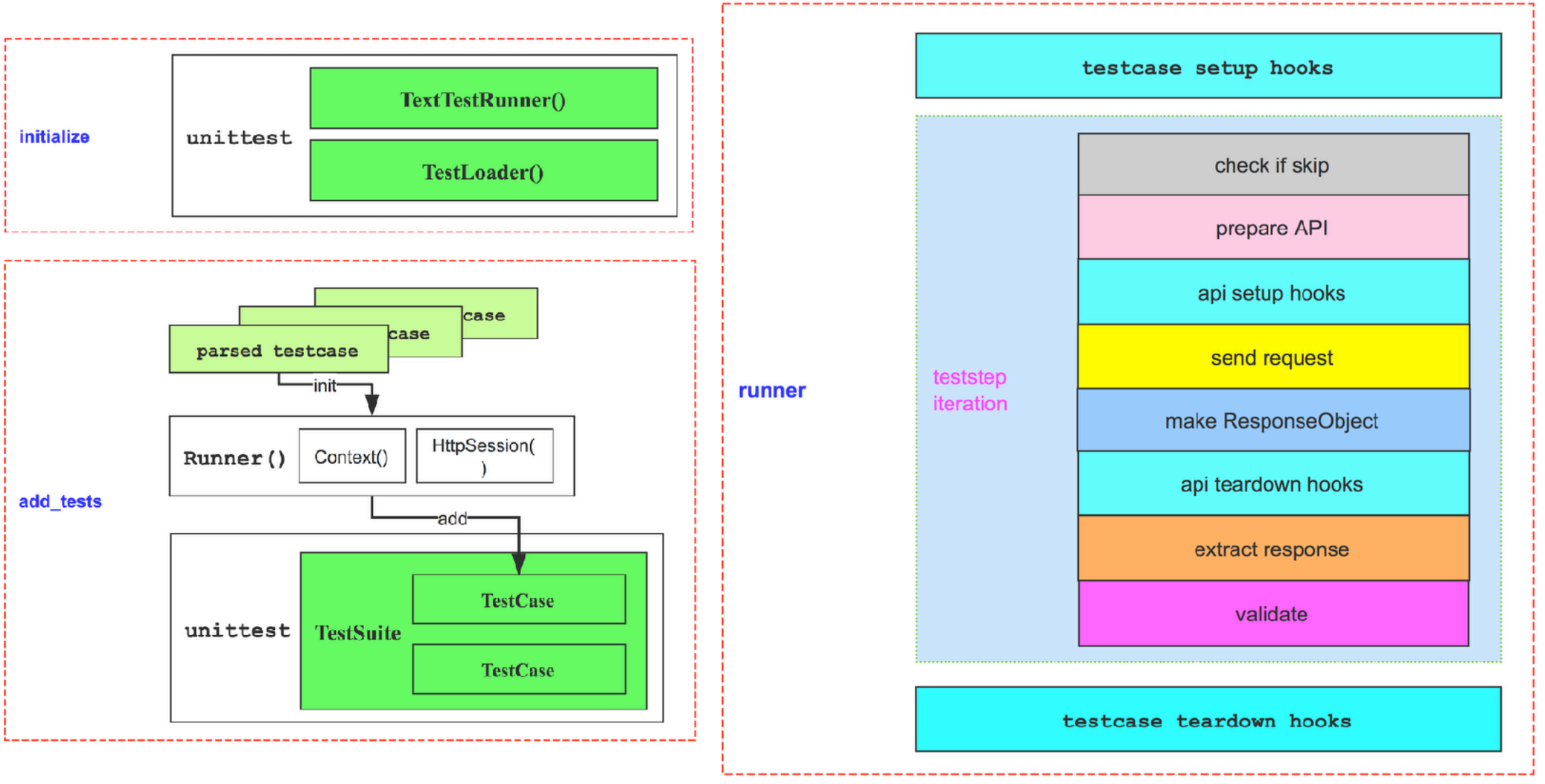
`report`

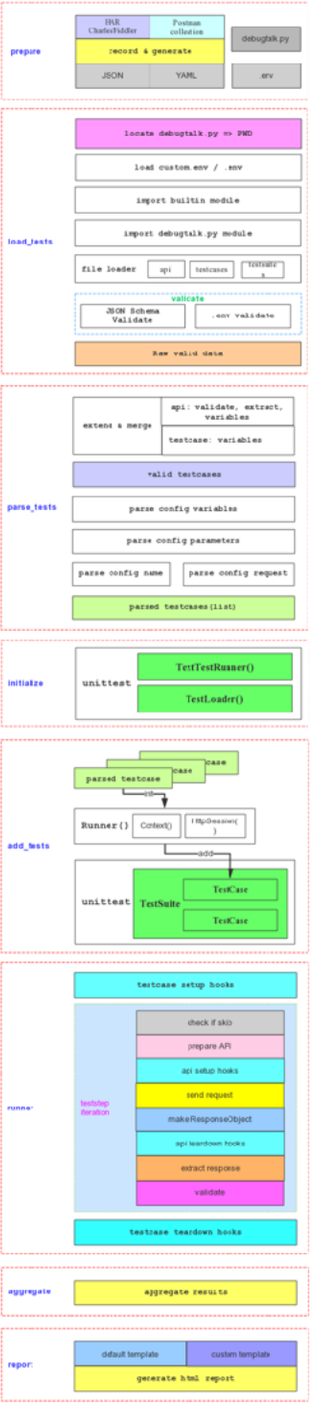
逻辑流程图


load_tests
locate debugtalk.py => PWD
load custom.env / .env
import builtin module
import debugtalk.py module
file loader
api
testcase s
testsuite s
validate
JSON Schema Validate
.env validate
Raw valid data
parse_tests
extend & merge
api: validate, extract, variables
testcase: variables
valid testcases
parse config variables
parse config parameters
parse config name
parse config request
parsed testcases(list)



逻辑流程图


initialize
unittest
TextTestRunner()
TestLoader()
add_tests
parsed testcase
case
case
init
Runner()
Context()
HttpSession(
)
add
unittest
TestSuite
TestCase
TestCase
runner
testcase setup hooks
teststep
iteration
check if skip
prepare API
api setup hooks
send request
make ResponseObject
api teardown hooks
extract response
validate
testcase teardown hooks



PYCON CHINA 2018

3、HttpRunner 核心特性

HttpRunner 是什么?

PYCON CHINA 2018

简单易用

功能强大

接口自动化测试工具

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*	动态计算*(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 - 测试准备

录制生成测试用例

export

HAR

har2case

YAML

postman2case

swagger2case

JSON

自动化平台录入/手工编写

核心特性 - 测试脚本优化

```yaml
- 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

```yaml
- 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)}

结果提取
extract

结果校验
validate

数据驱动
parameters

setup_hooks

teardown_hooks

```yaml
- 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
```

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 - 测试执行

```
➜ httprunner git:(master) ✗ hrun tests/httpbin/basic.y
INFO     Loading environment variables from /Users/debu
INFO     Start to run testcase: basic test with httpbir
headers
INFO     GET /headers
INFO     status_code: 200, response_time(ms): 619.0 ms,
INFO     start to validate.
.
user-agent
INFO     GET /user-agent
INFO     status_code: 200, response_time(ms): 302.21 ms
INFO     start to validate.
.

----------------------------------------------------------------------
Ran 2 tests in 0.924s

OK
INFO     Start to render Html report ...
INFO     Generated Html report: /Users/debugtalk/MyProj
```

```
➜ httprunner git:(master) ✗ hrun tests/httpbin/basic.yml --log-level debug
INFO     Loading environment variables from /Users/debugtalk/MyProjects/HttpRunner-dev/HttpRunner/tests/.env
DEBUG    Loaded variable: UserName
DEBUG    Loaded variable: Password
DEBUG    Loaded variable: PROJECT_KEY
INFO     Start to run testcase: basic test with httpbin
user-agent
DEBUG    call hook: ${setup_hook_prepare_kwargs($request)}
INFO     GET /user-agent
DEBUG    request kwargs(raw): {}
DEBUG    processed request:
> GET http://httpbin.org/user-agent
> kwargs: {'timeout': 120}
DEBUG
================== request details ==================
url              : 'http://httpbin.org/user-agent'
method           : 'GET'
headers          : {'User-Agent': 'python-requests/2.18.4', 'Accept-Encoding': 'gzip, deflate', 'Accept': '*/*', 'Connection': 'keep-alive'}
start_timestamp  : 1535087147.133355
body             : None

DEBUG
================== response details ==================
status_code      : 200
headers          : {'Connection': 'keep-alive', 'Server': 'gunicorn/19.9.0', 'Date': 'Fri, 24 Aug 2018 05:05:47 GMT', 'Content-Type': 'applic
edentials': 'true', 'Via': '1.1 vegur'}
content_size     : 45
response_time_ms : 528.63
elapsed_ms       : 521.792
encoding         : None
content          : b'{\n  "user-agent": "python-requests/2.18.4"\n}\n'
content_type     : 'application/json'
ok               : True
url              : 'http://httpbin.org/user-agent'
reason           : 'OK'
cookies          : {}
text             : '{\n  "user-agent": "python-requests/2.18.4"\n}\n'
json             : {'user-agent': 'python-requests/2.18.4'}

INFO     status_code: 200, response_time(ms): 528.63 ms, response_length: 45 bytes
INFO     start to validate.
DEBUG    extract: status_code   => 200
DEBUG    validate: status_code equals 200(int)   ==> pass
DEBUG    extract: content.user-agent    => python-requests/2.18.4
DEBUG    validate: content.user-agent startswith python-requests(str)   ==> pass
.

----------------------------------------------------------------------
Ran 1 test in 0.532s

OK
DEBUG
================== Variables & Output ==================
Type   | Variable         :  Value
------ | ---------------- :  --------------------------
Var    | HTTPBIN_SERVER   :  http://127.0.0.1:3458
Var    | SECRET_KEY       :  DebugTalk
Var    | BASE_URL         :  http://127.0.0.1:5000
Var    | demo_default_request :  {"base_url": "$BASE_URL", "headers": {"content-type": "application/json"}}

------------------------------------------------------------

DEBUG    No html report template specified, use default.
INFO     Start to render Html report ...
DEBUG    render data: {'success': True, 'stat': {'testsRun': 1, 'failures': 0, 'errors': 0, 'skipped': 0, 'expectedFailures': 0, 'unexpectedS
3203}, 'platform': {'httprunner_version': '1.5.11', 'python_version': 'CPython 3.6.5+', 'platform': 'Darwin-17.6.0-x86_64-i386-64bit'}, 'deta
tedFailures': 0, 'unexpectedSuccesses': 0, 'successes': 1}, 'time': {'start_at': 1535087147.132546, 'duration': 0.5316638946533203}, 'records
: 'http://httpbin.org/user-agent', 'method': 'GET', 'headers': {'User-Agent': 'python-requests/2.18.4', 'Accept-Encoding': 'gzip, deflate', '
 'response': {'status_code': 200, 'headers': {'Connection': 'keep-alive', 'Server': 'gunicorn/19.9.0', 'Date': 'Fri, 24 Aug 2018 05:05:47 GMT
```

核心特性 - 测试报告

Test Report:

Summary

START AT		2018-08-24 13:08:18	
DURATION		3.566 seconds	
PLATFORM	HttpRunner 1.5.11	CPython 3.6.5+	Darwin-
TOTAL	SUCCESS	FAILED	ERROR
9	9	0	0

Details

basic test with httpbin

base_url	http://httpbin.org/		
TOTAL: 9	SUCCESS: 9	FAILED: 0	ERROR: 0
Status	Name		Response Time
success	headers		564.02 ms
success	user-agent		303.29 ms
success	get without params		261.03 ms
success	get with params in url		264.09 ms
success	get with params in params field		290.97 ms
success	set cookie		568.02 ms
success	extract cookie		257.49 ms
success	post data		295.98 ms
success	validate content length		743.18 ms

Request and Response data

Request:

url	http://httpbin.org/user-agent
method	GET
headers	**User-Agent**: python-requests/2.18.4 **Accept-Encoding**: gzip, deflate **Accept**: */* **Connection**: keep-alive
start_timestamp	1535087298.8629591
body	None

Response:

status_code	200
headers	**Connection**: keep-alive **Server**: gunicorn/19.9.0 **Date**: Fri, 24 Aug 2018 05:08:19 GMT **Content-Type**: application/json **Cont** **Acces** **Acces** **Via**: 1
encoding	None
content	{ "us }
ok	True
url	http://
reason	OK
cookies	{}

Validators:

check	comparator	expect value	actual value
status_code	eq	200	200
content.user-agent	startswith	python-requests	python-requests/2.18.4

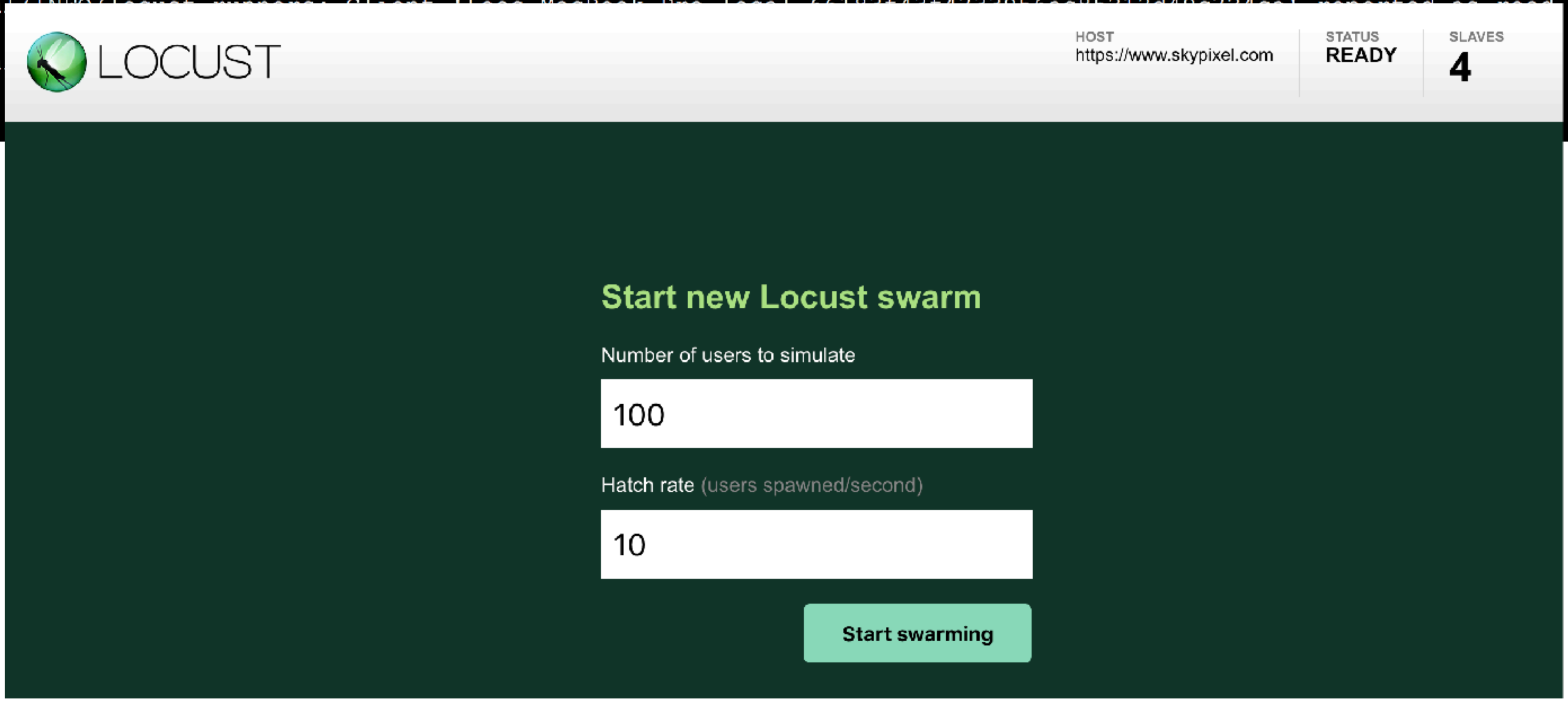
Statistics:

content_size(bytes)	45
response_time(ms)	303.29
elapsed(ms)	301.897

核心特性 - 性能测试

```
➜ httprunner git:(master) ✗ locusts -f skypixel.yml --processes 4
INFO  Loading environment variables from /Users/debugtalk/MyProjects/HttpRunner-dev/HttpRunner/.env
[2018-10-14 00:41:40,940] Leos-MacBook-Pro.local/INFO/locust.main: Starting web monitor at *:8089
[2018-10-14 00:41:40,946] Leos-MacBook-Pro.local/INFO/locust.main: Starting Locust 0.8.1
[2018-10-14 00:41:40,947] Leos-MacBook-Pro.local/INFO/locust.main: Starting Locust 0.8.1[2018-10-14 00:41:40,947] Leos-MacBook-Pro.local/INFO/locust.ma
in: Starting Locust 0.8.1

[2018-10-14 00:41:40,948] Leos-MacBook-Pro.local/INFO/locust.main: Starting Locust 0.8.1
[2018-10-14 00:41:40,949] Leos-MacBook-Pro.local/INFO/locust.main: Starting Locust 0.8.1
[2018-10-14 00:41:40,952] Leos-MacBook-Pro.local/INFO/locust.runners: Client 'Leos-MacBook-Pro.local_0e1aba43bd2aafc88ff28bc2b6ddd9ae' reported as read
y. Currently 1 clients ready to swarm.
[2018-10-14 00:41:40,952] Leos-MacBook-Pro.local/INFO/locust.runners: Client 'Leos-MacBook-Pro.local_1f6cda5ab3b7270dbc0e643268a60f27' reported as read
y. Currently 2 clients ready to swarm.
[2018-10-14 00:41:40,952] Leos-MacBook-Pro.loca
y. Currently 3 clients ready to swarm.
[2018-10-14 00:41:40,952] Leos-MacBook-Pro.loca
y. Currently 4 clients ready to swarm.
```



HttpRunner 还是什么?

不仅仅是自动化测试工具

通用的自动化测试解决方案

融合最佳工程实践

打造接口自动化测试生态

基于 HttpRunner 的接口测试平台:

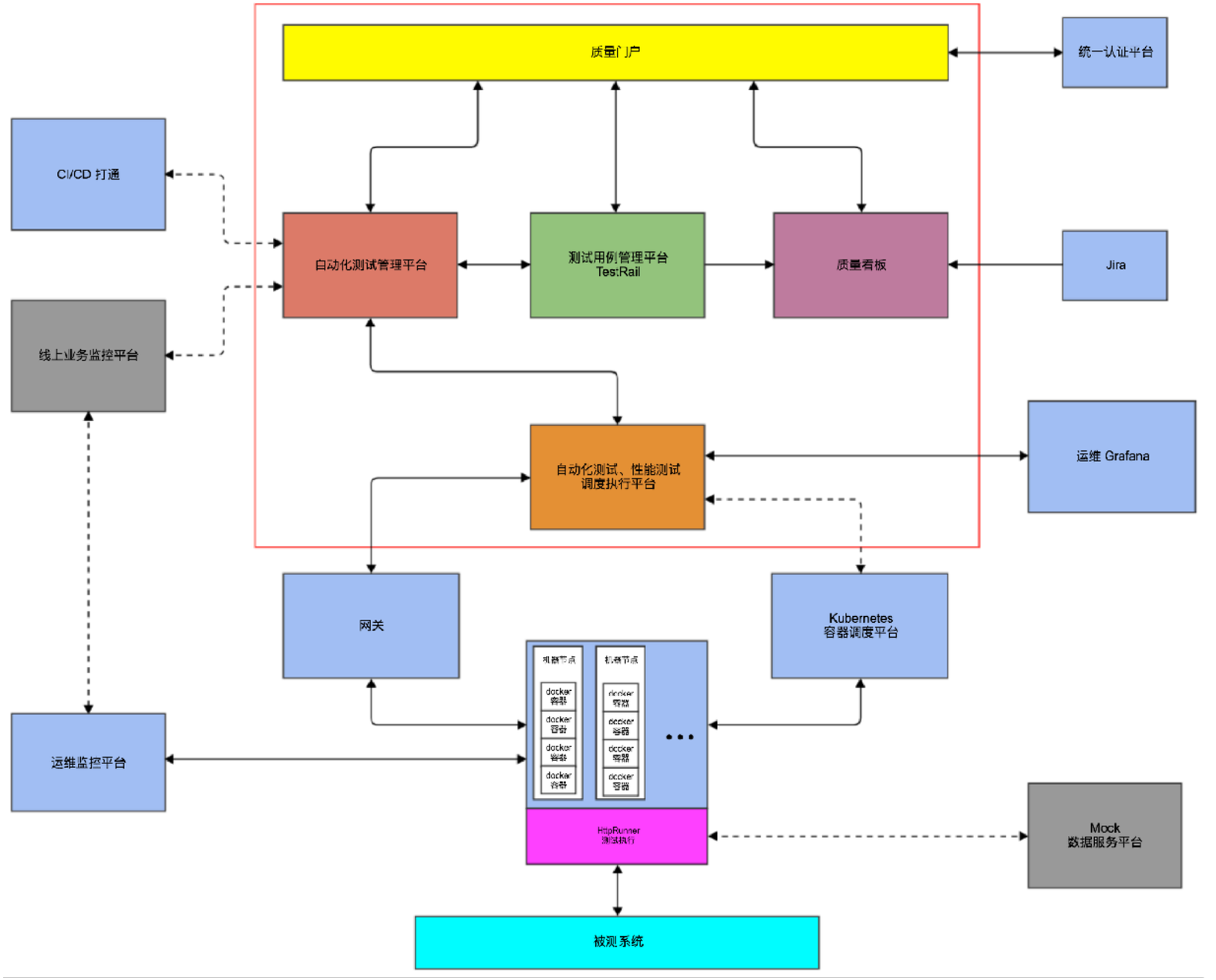
- **HttpRunnerManager**
- **FastRunner**
- SECO
- testcenter
- ApiTestWeb

4、HttpRunner 实践案例

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

实践案例 - 整体架构图



核心组件开源

实践案例 - 自动化测试平台

接口查看 ✚ 编辑接口 ⇄ 脚本视图

common

项目名称	demo
接口描述	登录用户中心
接口定义	login_account
接口参数	

GET	https://[redacted]/login		params

request	headers		
	data	backUrl	$backUrl
		appId	$appId
variables	backUrl	http://[redacted]/repair/index&locale=zh_CN	
	appId	store	
extract			
hooks			
validate			

在线编辑 & 查看

接口查看 ✚ 编辑接口 ⇄ 表格视图

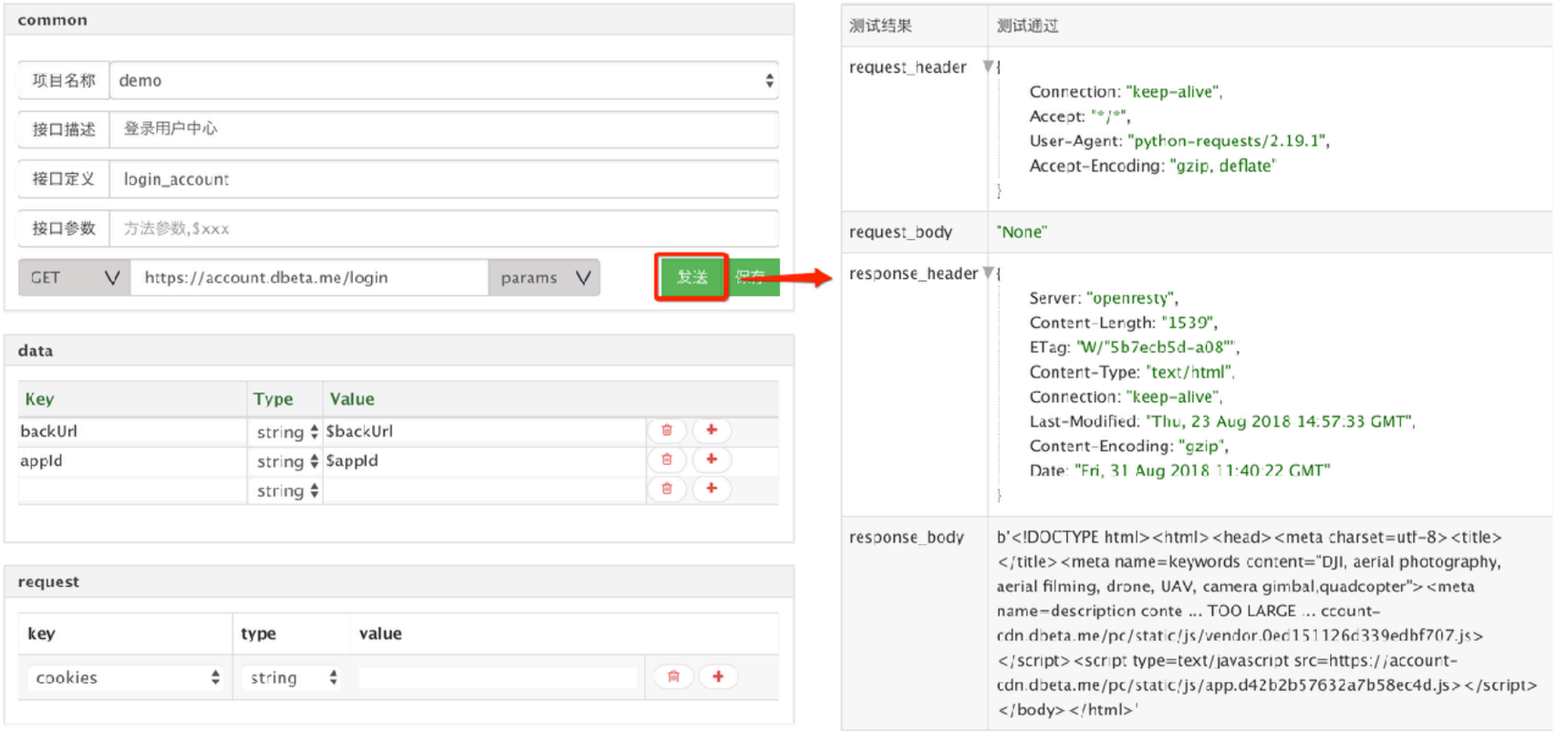
common

项目名称	demo
接口描述	登录用户中心
接口定义	login_account
接口参数	

GET	https://[redacted]/login	params

```
{
    "test": {
        "request": {
            "params": {
                "backUrl": "$backUrl",
                "appId": "$appId"
            },
            "headers": {},
            "url": "https://[redacted]/login",
            "method": "GET"
        },
        "variables": [
            {
                "backUrl": "http://[redacted]/repair/index&locale=zh_CN"
            },
            {
                "appId": "store"
            }
        ],
        "extract": [],
        "def": "login_account()",
        "validate": []
    }
}
```

实践案例 - 自动化测试平台

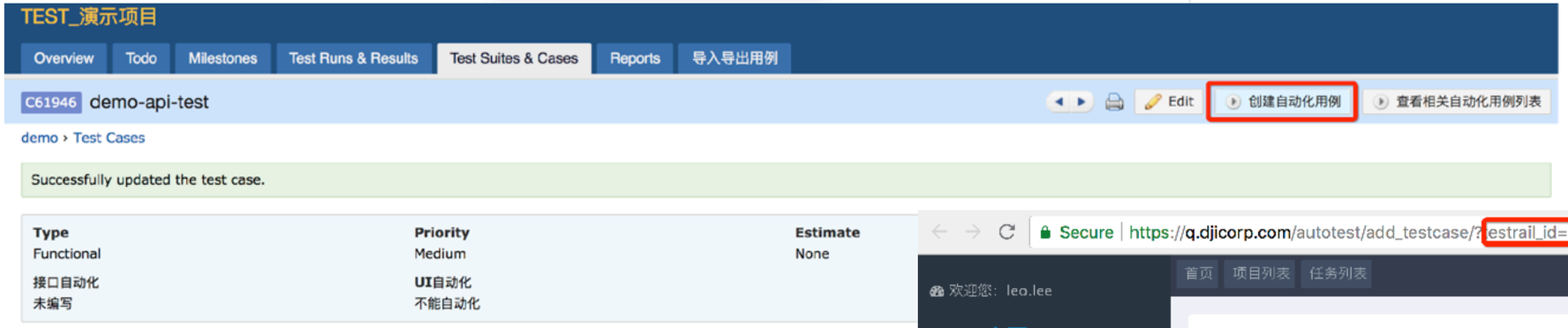


在线调试接口

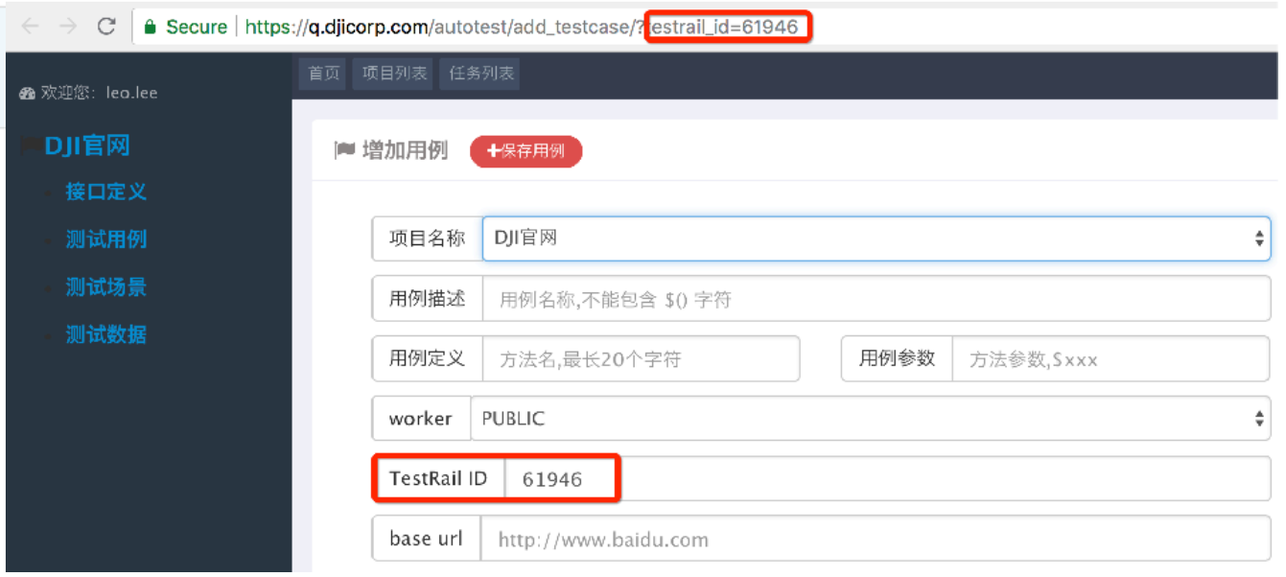
实践案例 - 自动化测试平台

实践案例 - 打通功能测试平台（TestRail）

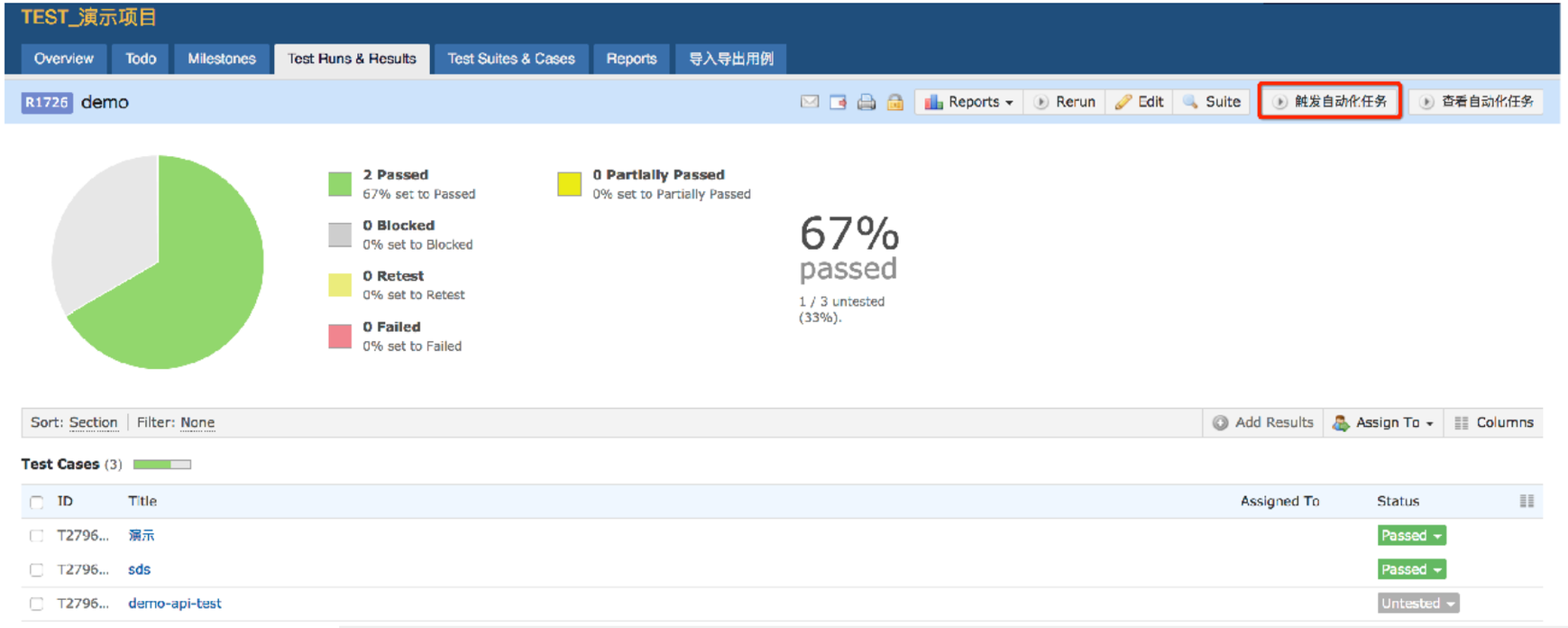
关联功能测试用例和自动化测试用例



自动化率统计



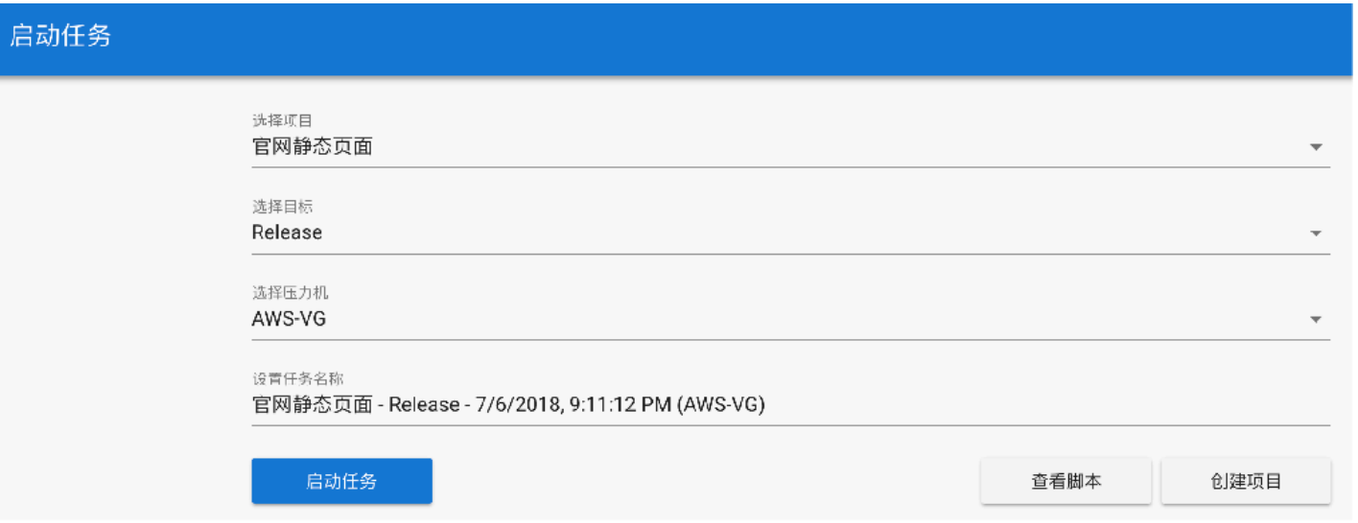
实践案例 - 打通功能测试平台（TestRail）



触发执行

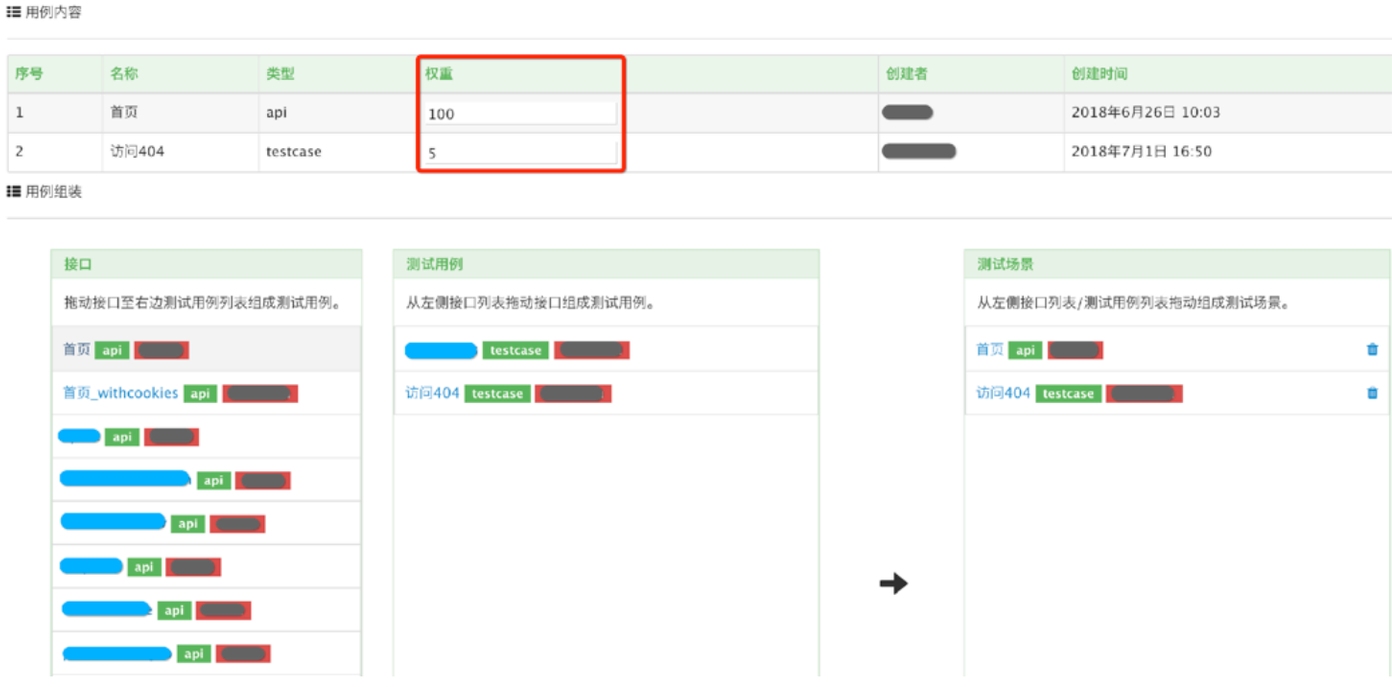
查看执行结果

实践案例 - 性能测试平台



查看性能测试任务对应的自动化测试用例

比例权重配置

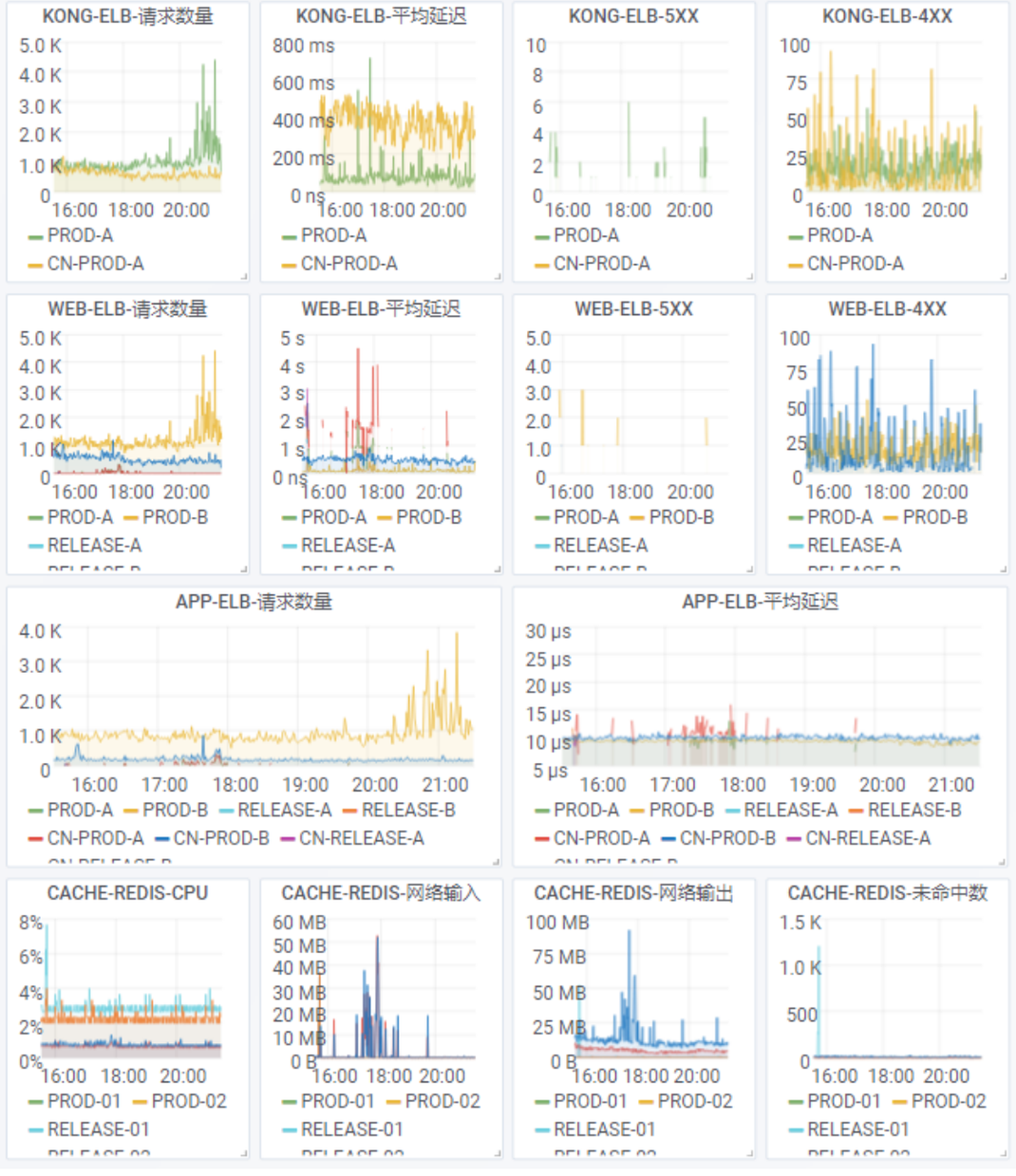


实践案例 - 性能测试平台

智能加压

性能数据
Grafana

New Relic



PYCON CHINA 2018

THANK YOU

You can find me here:

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