

# 最全的Pytest+Allure使用教程，建议收藏

知 [zhanlan.zhihu.com/p/455445067](https://zhuanlan.zhihu.com/p/455445067)

Allure是开源的免费的自动化测试报告，支持Java,Python，我们来看看如何使用Python pytest与Allure整合，输出漂亮的测试报告。

你们说话啊  
你们又在偷偷学啥 怎么  
突然不说话了

我好害怕 好怕你们又偷偷卷源码了



看你们啥都会真是比自己学的全忘了还难受

## 一.Allure安装

windows：下载，解压，并配置环境变量：<https://github.com/allure-framework/allure2/releases>

mac：brew install allure

## 二.Allure报告结构

1.Overview:整体数据显示。

2.Suites:用例集合,按照套件和类分组的已执行测试的标准结构表示形式。

3.Behaviors :对于行为驱动的方法 ,此选项卡根据Epic、Feature和Story标记对测试结果进行分组。

4.Categories :“类别”选项卡提供了创建自定义缺陷分类以应用测试结果的方法。

5.Graphs :用图表显示测试数据中收集的不同统计数据 ,状态分解或严重性和持续时间图。

6.Packages :软件包选项卡表示测试结果的树状布局 ,按不同的包名分组。

7.Timeline :时间轴选项卡可视化测试执行的回顾 ,allure适配器收集测试的精确时间 ,在这个选项卡上 ,它们相应地按照顺序或并行的时间结构排列。

## 三.安装Python依赖

```
windows : pip install allure-pytest  
mac : pip3 install allure-pytest
```

这将安装Allure-pytest和Allure-python-commons包 ,以生成与Allure 2兼容的报告数据。

### 1.Windows

```
C:\Users\...>pip install allure-pytest  
WARNING: Ignoring invalid distribution -ip (d:\program files\python3.8.2\lib\site-packages)  
WARNING: Ignoring invalid distribution -ip (d:\program files\python3.8.2\lib\site-packages)  
Collecting allure-pytest  
  Using cached allure_pytest-2.9.45-py3-none-any.whl (9.8 kB)  
Requirement already satisfied: six>=1.9.0 in d:\program files\python3.8.2\lib\site-packages (from allure-pytest) (1.15.0)  
Collecting pytest>=4.5.0  
  Using cached pytest-6.2.5-py3-none-any.whl (280 kB)  
Collecting allure-python-commons==2.9.45  
  Using cached allure_python_commons-2.9.45-py3-none-any.whl (15 kB)  
Collecting attrs>=16.0.0  
  Using cached attrs-21.3.0-py2.py3-none-any.whl (61 kB)  
Collecting pluggy>=0.4.0  
  Using cached pluggy-1.0.0-py2.py3-none-any.whl (13 kB)  
Collecting packaging  
  Using cached packaging-21.3-py3-none-any.whl (40 kB)  
Requirement already satisfied: py>=1.8.2 in d:\program files\python3.8.2\lib\site-packages (from pytest>=4.5.0->allure-pytest) (1.10.0)  
Collecting colorama  
  Using cached colorama-0.4.4-py2.py3-none-any.whl (16 kB)  
Collecting toml  
  Using cached toml-0.10.2-py2.py3-none-any.whl (16 kB)  
Collecting iniconfig  
  Using cached iniconfig-1.1.1-py2.py3-none-any.whl (5.0 kB)  
Collecting atomicwrites>1.0  
  Using cached atomicwrites-1.4.0-py2.py3-none-any.whl (6.8 kB)  
Collecting pyparsing!=3.0.5,>=2.0.2  
  Using cached pyparsing-3.0.6-py3-none-any.whl (97 kB)  
WARNING: Ignoring invalid distribution -ip (d:\program files\python3.8.2\lib\site-packages)  
Installing collected packages: pyparsing, toml, pluggy, packaging, iniconfig, colorama, attrs, atomicwrites, pytest, allure-python-commons, allure-pytest  
WARNING: Ignoring invalid distribution -ip (d:\program files\python3.8.2\lib\site-packages)  
Successfully installed allure-pytest-2.9.45 allure-python-commons-2.9.45 atomicwrites-1.4.0 attrs-21.3.0 colorama-0.4.4 iniconfig-1.1.1 packaging-21.3 pluggy-1.0.0 pyparsing-3.0.6 pytest-6.2.5 toml-0.10.2  
WARNING: Ignoring invalid distribution -ip (d:\program files\python3.8.2\lib\site-packages)  
WARNING: Ignoring invalid distribution -ip (d:\program files\python3.8.2\lib\site-packages)
```

```
C:\Users\二 D:\pip list
WARNING: Ignoring invalid distribution -ip (d:\program files\python3.8.2\lib\site-packages)
Package           Version
airtest          1.2.1
allure-pytest    2.9.45
allure-python-commons 2.9.45
```

## 2.Mac

```
MacBook-Pro-3 ~ % pip3 install allure-pytest
Collecting allure-pytest
  Using cached allure_pytest-2.9.45-py3-none-any.whl (9.8 kB)
Collecting allure-python-commons==2.9.45
  Using cached allure_python_commons-2.9.45-py3-none-any.whl (15 kB)
Requirement already satisfied: six>=1.9.0 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from allure-pytest) (1.16.0)
Requirement already satisfied: pytest>=4.5.0 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from allure-pytest) (6.2.5)
Requirement already satisfied: pluggy>=0.4.0 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from allure-python-commons==2.9.45->allure-pytest) (1.0.0)
Requirement already satisfied: attrs>=16.0.0 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from allure-python-commons==2.9.45->allure-pytest) (21.3.0)
Requirement already satisfied: py>=1.8.2 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from pytest>=4.5.0->allure-pytest) (1.10.0)
Requirement already satisfied: iniconfig in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from pytest>=4.5.0->allure-pytest) (1.1.1)
Requirement already satisfied: packaging in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from pytest>=4.5.0->allure-pytest) (20.9)
Requirement already satisfied: toml in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from pytest>=4.5.0->allure-pytest) (0.10.2)
Requirement already satisfied: pyParsing>=2.0.2 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from packaging>pytest>=4.5.0->allure-pytest) (2.4.7)
Installing collected packages: allure-python-commons, allure-pytest
Successfully installed allure-pytest-2.9.45 allure-python-commons-2.9.45
MacBook-Pro-3 ~ % pip3 list
Package           Version
adbutils          0.11.0
airtest           1.1.11
allure-pytest     2.9.45
allure-python-commons 2.9.45
```

## 四.基本用法

Allure监听器在测试执行期间会收集结果，只需添加alluredir选项，并选择输出的文件路径即可。

```
pytest --alluredir=./allure-results
```

```
MacBook-Pro-3 ~ % ~/Documents/otherdocument/python_project/python-allure pytest --alluredir=./allure-results
zsh: /usr/local/bin/pytest: bad interpreter: /usr/local/opt/python/bin/python3.7: no such file or directory
===== test session starts =====
platform darwin -- Python 3.9.5, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /~/Documents/otherdocument/python_project/python-allure
plugins: allure-pytest-2.9.45
collected 0 items

===== no tests ran in 0.03s =====
MacBook-Pro-3 ~ % ~/Documents/otherdocument/python_project/python-allure ls
allure-results main.py redMe.md
```

```
# 执行此命令，将在默认浏览器中显示生成的报告
allure serve ./allure-results
```

## 五.Allure注解说明：

使用方法	参数值	参数说明
@allure.epic()	epic描述	敏捷里面的概念，定义史诗，往下是feature
@allure.feature()	模块名称	功能点的描述，往下是story
@allure.story()	用户故事	用户故事，往下是title
@allure.title(用例的标题)	用例的标题	重命名html报告名称
@allure.testcase()	测试用例的链接地址	对应功能测试用例系统里面的case
@allure.issue()	缺陷	对应缺陷管理系统里面的链接
@allure.description()	用例描述	测试用例的描述
@allure.step()	操作步骤	测试用例的步骤
@allure.severity()	用例等级	blocker, critical, normal, minor, trivial
@allure.link()	链接	定义一个链接，在测试报告展现
@allure.attachment()	附件	报告添加附件

## 六.具体使用方法(以Mac系统为例)

基本报告：可以在Allure报告中看到所有默认的pytest状态: 只有由于断言错误而未成功的测试才会被标记为失败，任何其他异常都会导致测试状态失败。

```
import allure
import pytest

@allure.feature('test_success')
def test_success():
    """this test succeeds"""
    assert True

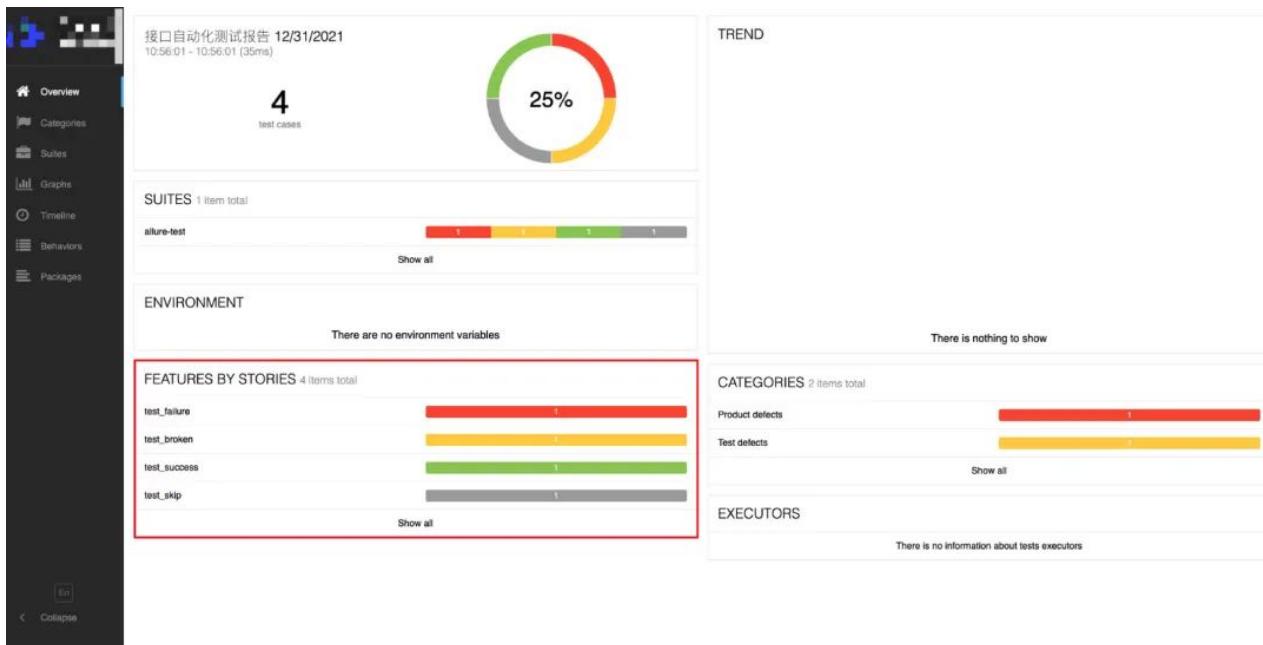
@allure.feature('test_failure')
def test_failure():
    """this test fails"""
    assert False

@allure.feature('test_skip')
def test_skip():
    """this test is skipped"""
    pytest.skip('for a reason! ')

@allure.feature('test_broken')
def test_broken():
    raise Exception('oops')

if __name__ == '__main__':
    # pytest.main(["-s", "allure-test.py"])
    '''
    -q: 安静模式，不输出环境信息
    -v: 丰富信息模式，输出更详细的用例执行信息
    -s: 显示程序中的print/logging输出
    '''
    pytest.main(['-s', '-q', 'test_allure02.py', '--clean-alluredir', '--alluredir=allure-results'])
    os.system(r"allure generate -c -o allure-report")
```

命令行输入allure generate -c -o allure-reports,即可在当前路径下生成测试报告。（-c:清空历史数据，-o:指定输出测试报告路径），如果使用allure serve allure-results,则会在系统默认目录下生成测试报告



## 七. 支持Pytest特性

Allure报告支持的一些常见Pytest特性，包括`xfails`、`fixture`和`finalizers`、`marks`、条件跳过和参数化。

### 1.xfail

功能未实现或者有Bug尚未修复，当测试通过时尽管会失败，它是一个`xpass`，将在测试摘要中报告。

```
import allure
import pytest

@allure.feature('test_xfail_expected_failure')
@pytest.mark.xfail(reason='该功能尚未实现')
def test_xfail_expected_failure():
    print("该功能尚未实现")
    assert False

@allure.feature('test_xfail_unexpected_pass')
@pytest.mark.xfail(reason='该Bug尚未修复')
def test_xfail_unexpected_pass():
    print("该Bug尚未修复")
    assert True

if __name__ == '__main__':
    pytest.main(['-s', '-q', 'test_allure02.py', '--clean-alluredir', '--alluredir=allure-results'])
    os.system(r'allure generate -c -o allure-report')
```

The screenshot shows the Allure Testops interface. On the left is a sidebar with navigation links: Overview, Categories, Suites, Graphs, Timeline, Behaviors (which is selected), and Packages. The main area is titled 'Behaviors' and lists two test cases under 'test\_xfail\_expected\_failure' and 'test\_xfail\_unexpected\_pass'. The first test has a status of 'Skipped' (XFAIL) and the second has a status of 'Passed' (XPASS). Each test card includes details like duration (0s), severity (normal), and execution notes.

## 2.skipif

当条件为True，跳过执行

```
import sys
import allure
import pytest

'''当条件为True则跳过执行'''
@allure.feature("test_skipif")
@pytest.mark.skipif("darwin" in sys.platform, reason="如果操作系统是Mac则跳过执行")
def test_skipif():
    print("操作系统是Mac，test_skipif()函数跳过执行")

# --clean-alluredir:每次执行前清空数据，这样在生成的报告中就不会追加，只显示当前执行的用例
if __name__ == '__main__':
    pytest.main(['-s', '-q', 'test_allure02.py', '--clean-alluredir', '--alluredir=allure-results'])
    os.system(r'allure generate -c -o allure-report')
```

The screenshot shows the Allure UI interface. On the left is a sidebar with navigation links: Overview, Categories, Suites, Graphs, Timeline, Behaviors (which is selected), and Packages. The main area is titled 'Behaviors'. It lists a single test item: '#1 test\_skipif'. To the right of the list is a detailed view of this test. The status bar at the top indicates 'Status: 0 0 0 1 0'. The detailed view shows the test is 'Skipped' due to a skipif condition. It includes sections for 'Tags' (@pytest.mark.skipif(True, reason='如果操作系统是Mac则跳过执行')), 'Severity: normal', 'Duration: 0s', and 'Execution' (No information about test execution is available).

### 3.parametrize

可以使用@Pytest.mark.parametrize进行参数化，所有参数名称和值都将被捕获到报告中。

```
import os
import allure
import pytest

@allure.step
def simple_step(step_param1, step_param2 = None):
    pass

@pytest.mark.parametrize('param1', [True, False], ids=['1', '2'])
def test_parameterize_with_id(param1):
    simple_step(param1)

@pytest.mark.parametrize('param1', [True, False])
@pytest.mark.parametrize('param2', ['1', '2'])
def test_parameterize_with_two_parameters(param1, param2):
    simple_step(param1, param2)

# --clean-alluredir:每次执行前清空数据，这样在生成的报告中就不会追加，只显示当前执行的用例
if __name__ == '__main__':
    pytest.main(['-s', '-q', 'test_allure03.py', '--clean-alluredir', '--alluredir=allure-results'])
    os.system(r'allure generate -c -o allure-report')
```

The screenshot shows the Allure UI interface. On the left is a sidebar with navigation links: Overview, Categories, Suites, Graphs, Timeline, Behaviors, and Packages. The main area is titled 'Suites'. It lists a suite named 'test\_allure03' containing several tests. One test, '#1 test\_parameterize\_with\_id[1]', is highlighted and shown in detail on the right. The status bar at the top indicates 'Status: 0 0 6 0 0'. The detailed view shows the test is 'Passed'. It includes sections for 'Overview', 'Parameters' (param1: True), 'Execution', and 'Test body' (simple\_step 2 parameters: step\_param1: True, step\_param2: None). The status bar at the bottom right shows '0s'.

### 4.feature

## 模块名称，功能点描述

### 5.step

Allure生成的测试报告最重要的一点就是，它允许对每个测试用例进行非常详细的步骤说明

```
import os
import allure
import pytest

@allure.step("步骤二")
def passing_step():
    pass

@allure.step("步骤三")
def step_with_nested_steps():
    nested_step()

@allure.step("步骤四")
def nested_step():
    nested_step_with_arguments(1, 'abc')

@allure.step("步骤五")
def nested_step_with_arguments(arg1, arg2):
    pass

@allure.step("步骤一")
def test_with_nested_steps():
    passing_step()
    step_with_nested_steps()

if __name__ == '__main__':
    pytest.main(['-s', '-q', 'test_allure04.py', '--clean-alluredir', '--alluredir=allure-results'])
    os.system(r'allure generate -c -o allure-report')
```

Suites

order	name	duration	status
			Status: 0 0 1 0 0

Marks: [ ]

test\_allure04

#1 test\_with\_nested\_steps 0s

Passed test\_with\_nested\_steps

Overview History Retries

Severity: normal

Duration: 0s

Execution

Test body

- 步骤一 4 sub-steps
- 步骤二
- 步骤三 2 sub-steps
- 步骤四 1 sub-step
- 步骤五 2 parameters

arg1	1
arg2	'abc'

## 6.attach

Allure测试报告可以显示许多不同类型的附件，这些附件可以补充测试、步骤或结果。可以使用allure.attach(body、name、attachment\_type、extension)调用来创建附件。

- body：要显示的内容（附件）
- name：附件名字
- attachment\_type：附件类型，是allure.attachment\_type里面的其中一种
- extension：附件的扩展名
- source：文件路径

语法一：allure.attach(body,name,attachment\_type,extension)

语法二：allure.attach.file(source,name,attachment\_type,extension)

```
import os
import allure
import pytest

@pytest.fixture
def attach_file_in_module_scope_fixture_with_finalizer(request):
    allure.attach('在fixture前置操作里面添加一个附件txt', 'fixture前置附件',
allure.attachment_type.TEXT)

    def finalizer_module_scope_fixture():
        allure.attach('在fixture后置操作里面添加一个附件txt', 'fixture后置附
件',allure.attachment_type.TEXT)
        request.addfinalizer(finalizer_module_scope_fixture)

def test_with_attachments_in_fixture_and_finalizer(attach_file_in_module_scope_fixture_w
pass

def test_multiple_attachments():
    allure.attach('<head></head><body>html page</body>', 'Attach with HTML type',
allure.attachment_type.HTML)
    allure.attach.file('/Users/mrjade/Downloads/happy-new-year.html',
attachment_type=allure.attachment_type.HTML)

if __name__ == '__main__':
    pytest.main(['-s', '-q', 'test_allure05.py', '--clean-alluredir', '--
alluredir=allure-results'])
    os.system(r'allure generate -c -o allure-report')
```

**Suites**

order	name	duration	status				
			Status: 0 0 2 0 0				
Marks:	[ ] [ ]						
▾ test_allure05 <table border="1"> <tr> <td>#2 test_multiple_attachments</td> <td>2ms</td> </tr> <tr> <td>#1 test_with_attachments_in_fixture_and_finalizer</td> <td>0s</td> </tr> </table>				#2 test_multiple_attachments	2ms	#1 test_with_attachments_in_fixture_and_finalizer	0s
#2 test_multiple_attachments	2ms						
#1 test_with_attachments_in_fixture_and_finalizer	0s						

**Passed test\_multiple\_attachments**

**Overview History Retries**

Severity: normal  
Duration: 0 2ms

**Execution**

▀ **Test body**

▀ **Attach with HTML type**

html page

8b8d6e7c-ae5b-41c2-a53a-8ae890082e5a-attachment.html 3.9 KB

**Suites**

order	name	duration	status				
			Status: 0 0 2 0 0				
Marks:	[ ] [ ]						
▾ test_allure05 <table border="1"> <tr> <td>#2 test_multiple_attachments</td> <td>2ms</td> </tr> <tr> <td>#1 test_with_attachments_in_fixture_and_finalizer</td> <td>0s</td> </tr> </table>				#2 test_multiple_attachments	2ms	#1 test_with_attachments_in_fixture_and_finalizer	0s
#2 test_multiple_attachments	2ms						
#1 test_with_attachments_in_fixture_and_finalizer	0s						

**Passed test\_with\_attachments\_in\_fixture\_and\_finalizer**

**Overview History Retries**

Severity: normal  
Duration: 0 0s

**Execution**

▀ **Set up**

attach\_file\_in\_module\_scope\_fixture\_with\_finalizer 1 attachment 1ms  
fixture前置附件  
在fixture前置操作里面添加一个附件.txt

▀ **Tear down**

attach\_file\_in\_module\_scope\_fixture\_with\_finalizer::finalizer\_module\_scope\_fixture 1 attachment 1ms  
fixture后置附件  
在fixture后置操作里面添加一个附件.txt

## 7.description

描述类信息，比如对函数的描述，说明这个函数的作用，如：注册接口。

语法一：@allure.description()

语法二：@allure.description\_html()

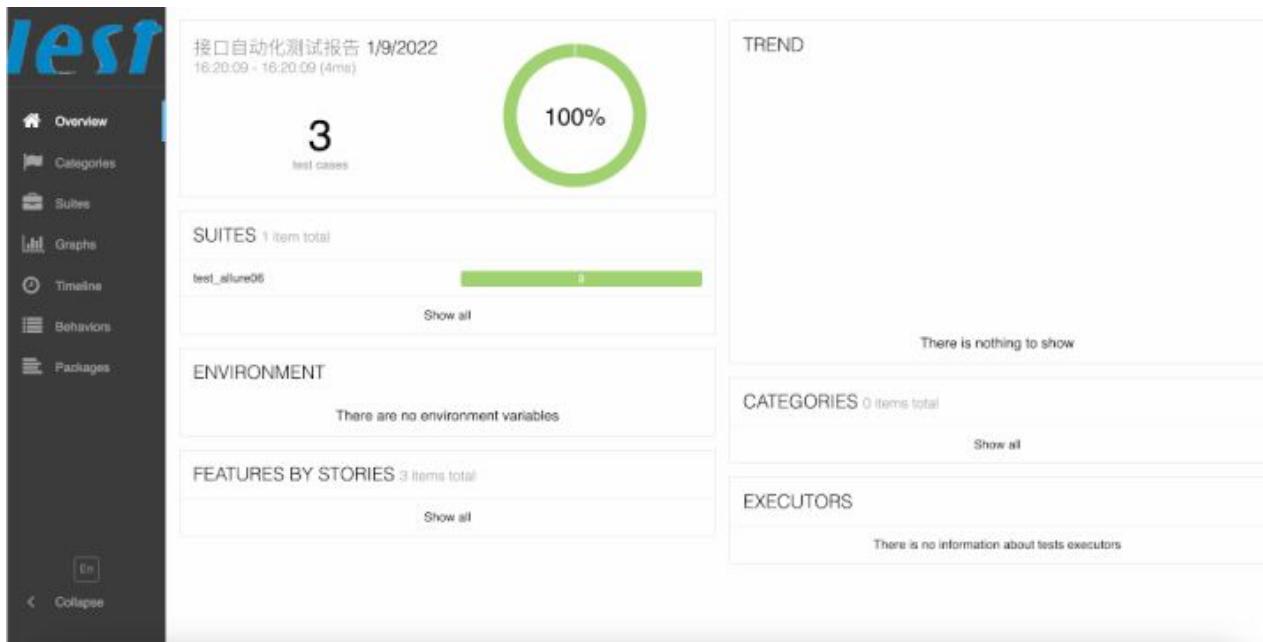
```
import os
import allure
import pytest

@allure.description_html("""
<h1>这是html描述</h1>
<table style="width:100%">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
</tr>
<tr align="center">
<td>jade</td>
<td>mr</td>
<td>18</td>
</tr>
<tr align="center">
<td>road</td>
<td>Tester</td>
<td>18</td>
</tr>
</table>
""")
def test_html_description():
    assert True

@allure.description("""多行描述""")
def test_description_from_decorator():
    assert 42 == int(6 * 7)

def test_unicode_in_docstring_description():
    """在函数下方描述也可"""
    assert 42 == int(6 * 7)

if __name__ == '__main__':
    pytest.main(['-s', '-q', 'test_allure06.py', '--clean-alluredir', '--alluredir=allure-results'])
    os.system(r'allure generate -c -o allure-report')
```



## 8.title

- 添加测试用例标题，通俗来讲就是将函数名直接改成我们想要的，使得测试用例的标题更具有可读性
- 支持占位符传递关键字参数（动态标题，结合@ pytest.mark.parametrize 使用）

```

import os
import allure
import pytest

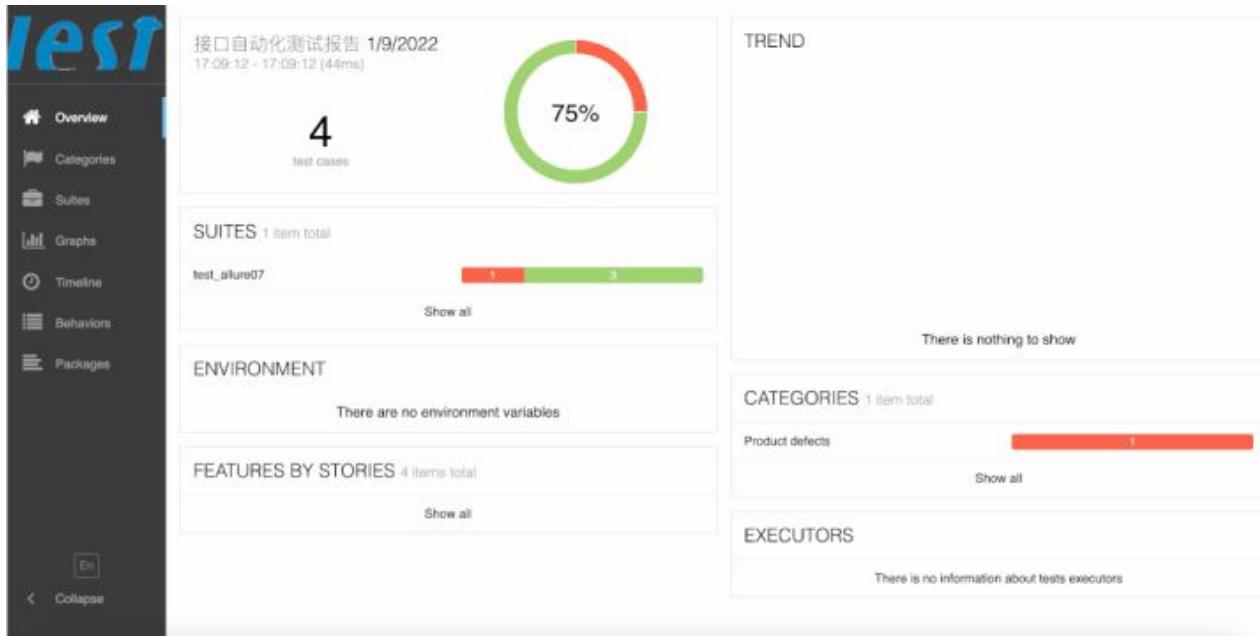
@allure.title("断言2+2=4")
def test_with_a_title():
    assert 2 + 2 == 4

@allure.title("动态标题: {param1} + {param2} = {expected}")
@pytest.mark.parametrize('param1,param2,expected', [(2, 2, 4), (1, 2, 5)])
def test_with_parameterized_title(param1, param2, expected):
    assert param1 + param2 == expected

@allure.title("这是个动态标题，会被替换")
def test_with_dynamic_title():
    assert 2 + 2 == 4
    allure.dynamic.title('测试结束，做为标题')

if __name__ == '__main__':
    pytest.main(['-s', '-q', 'test_allure07.py', '--clean-alluredir', '--alluredir=allure-results'])
    os.system(r'allure generate -c -o allure-report')

```



9.link

将测试报告与缺陷跟踪或测试管理系统整合，@allure.link、@allure.issue和@allure.testcase。

```

import os
import allure
import pytest

@allure.link('https://www.cnblogs.com/mrjade/')
def test_with_link():
    pass

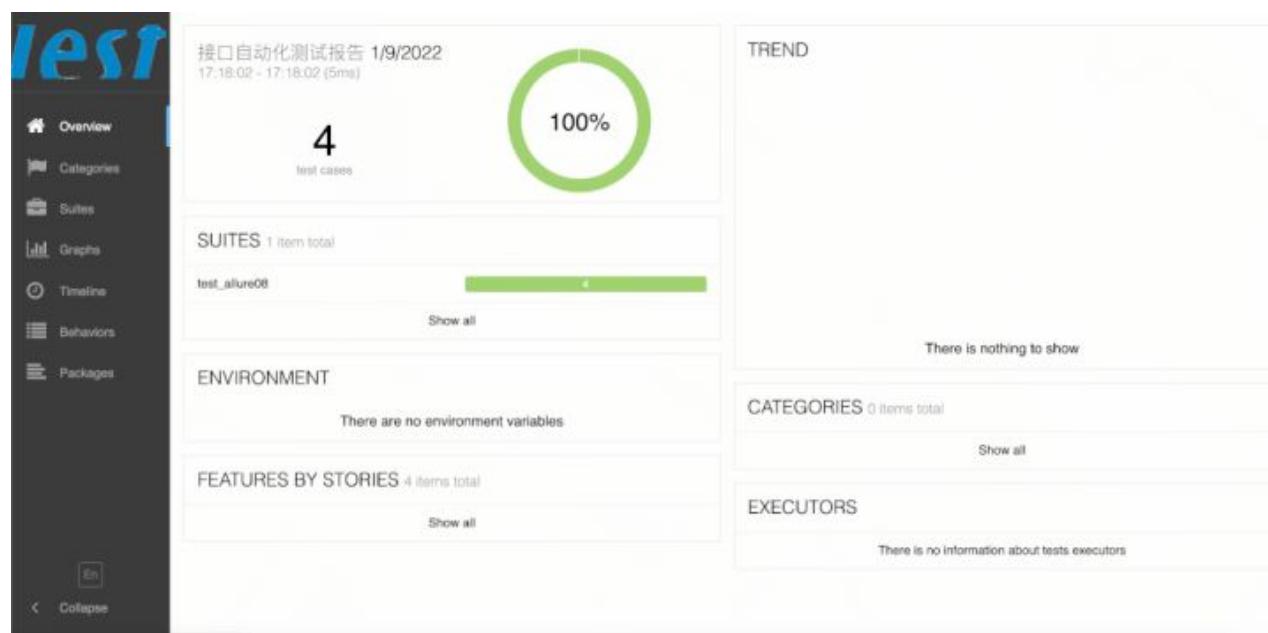
@allure.link('https://www.cnblogs.com/mrjade/', name='点击进入mrjade博客园')
def test_with_named_link():
    pass

@allure.issue('https://github.com/allure-framework/allure-python/issues/642', 'bug
issue链接')
def test_with_issue_link():
    pass

@allure.testcase("https://www.cnblogs.com/mrjade/", '测试用例地址')
def test_with_testcase_link():
    pass

if __name__ == '__main__':
    pytest.main(['-s', '-q', 'test_allure08.py', '--clean-alluredir', '--
alluredir=allure-results'])
    os.system(r'allure generate -c -o allure-report')

```

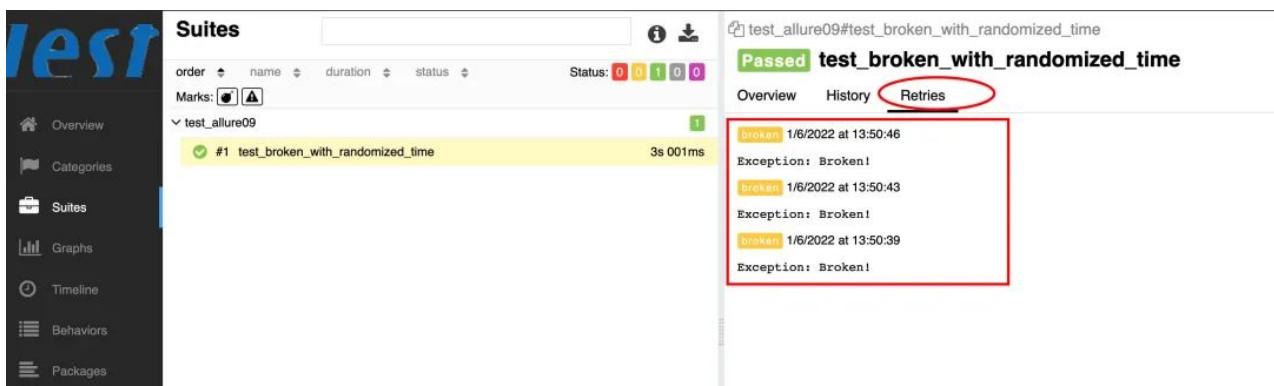


## 10.Retries

Allure允许测试报告中显示单个测试运行期间重新执行的测试的信息，以及一段时间内测试执行的历史,需与Pytest插件结合使用。

```
windows : pip install pytest-rerunfailures  
mac : pip3 install pytest-rerunfailures
```

```
import os  
import allure  
import random  
import time  
import pytest  
  
@allure.step  
def passing_step():  
    pass  
  
@allure.step  
def flaky_broken_step():  
    if random.randint(1, 5) != 1:  
        raise Exception('Broken!')  
  
"""需安装【pip3 install pytest-rerunfailures】"""  
@pytest.mark.flaky(reruns=3, reruns_delay=1) # 如果失败则延迟1s后重试  
def test_broken_with_randomized_time():  
    passing_step()  
    time.sleep(random.randint(1, 3))  
    flaky_broken_step()  
  
if __name__ == '__main__':  
    pytest.main(['-s', '-q', 'test_allure09.py', '--clean-alluredir', '--  
alluredir=allure-results'])  
    os.system(r'allure generate -c -o allure-report')
```



The screenshot shows the Allure UI interface. On the left is a sidebar with navigation links: Overview, Categories, Suites (which is selected), Graphs, Timeline, Behaviors, and Packages. The main area is titled 'Suites' and shows a list of tests under 'test\_allure09'. One test, '#1 test\_broken\_with\_randomized\_time', is highlighted in yellow and has a duration of '3s 001ms'. To the right, a detailed view for this test is shown with the title 'Passed test\_broken\_with\_randomized\_time'. Below it are tabs for Overview, History, and Retries (which is circled in red). The Retries section displays three failed attempts, each with a timestamp and the exception message 'Broken!':  
- 1/6/2022 at 13:50:46: Broken!  
- 1/6/2022 at 13:50:43: Broken!  
- 1/6/2022 at 13:50:39: Broken!

## 11.tags

我们希望对执行的测试用例保持灵活性。Pytest允许使用标记修饰符@ pytest.mark，Allure提供了3种类型的标记装饰符

- BDD样式的标记装饰器
- @allure.epic : 敏捷里面的概念，定义史诗，往下是feature
- @allure.feature : 功能点的描述，理解成模块往下是story
- @allure.story : 故事，往下是title
- 优先级（严重程度）标记装饰器 根据严重程度来标记你的测试。严重程度装饰器，它使用一个allure.severity\_level enum值作为参数。如：  
@allure.severity(allure.severity\_level.TRIVIAL)
- 自定义标记装饰器

```
import os
import allure
import pytest

@pytest.fixture(scope="session")
def login_fixture():
    print("前置登录")

@allure.step("步骤1")
def step_1():
    print("操作步骤1")

@allure.step("步骤2")
def step_2():
    print("操作步骤2")

@allure.step("步骤3")
def step_3():
    print("操作步骤3")

@allure.step("步骤4")
def step_4():
    print("操作步骤4")

@allure.epic("会员项目")
@allure.feature("登录")
class TestAllureALL:

    @allure.testcase("https://www.cnblogs.com/mrjade/", '测试用例,点我一下')
    @allure.issue("https://github.com/allure-framework/allure-python/issues/642",
    'Bug 链接,点我一下')
    @allure.title("用户名错误")
    @allure.story("登录测试用例1")
    @allure.severity(allure.severity_level.TRIVIAL) # 不重要的
    # @allure.severity(allure.severity_level.MINOR) # 轻微的
    # @allure.severity(allure.severity_level.BLOCKER) # 阻塞的
    # @allure.severity(allure.severity_level.CRITICAL) # 严重的
    # @allure.severity(allure.severity_level.NORMAL) # 普通的
    def test_case_1(self):
        """
        公众号：测试工程师成长之路
        """
        print("测试用例1")
        step_1()
        step_2()

    @allure.title("用户名正确，密码错误")
    @allure.story("登录测试用例2")
    def test_case_2(self):
        print("测试用例2")
        step_1()
        step_3()
```

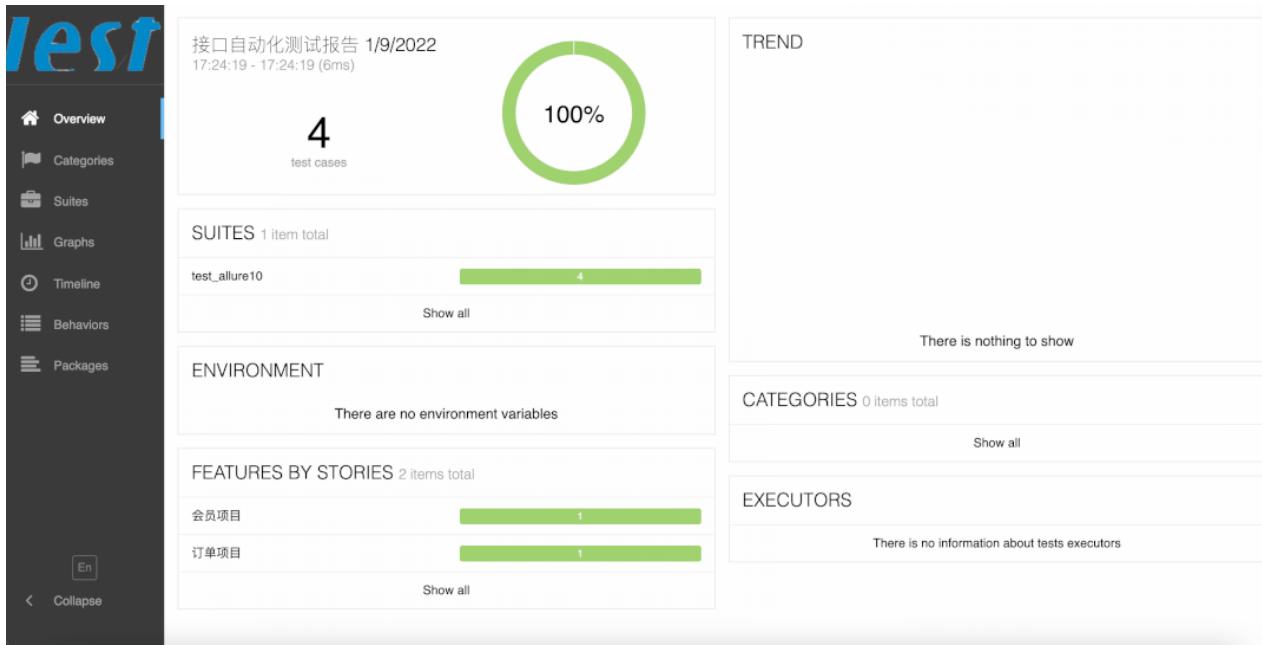
```

@allure.epic("订单项目")
@allure.feature("支付")
class TestAllureALL2:
    @allure.title("支付成功")
    @allure.story("支付测试用例例1")
    def test_case_1(self, login_fixture):
        print("支付测试用例例1")
        step_3()

    @allure.title("支付失败")
    @allure.story("支付测试用例例2")
    def test_case_2(self, login_fixture):
        print("支付测试用例例2")
        step_4()

if __name__ == '__main__':
    pytest.main(['-s', '-q', 'test_allure10.py', '--clean-alluredir', '--alluredir=allure-results'])
    os.system(r'allure generate -c -o allure-report')

```



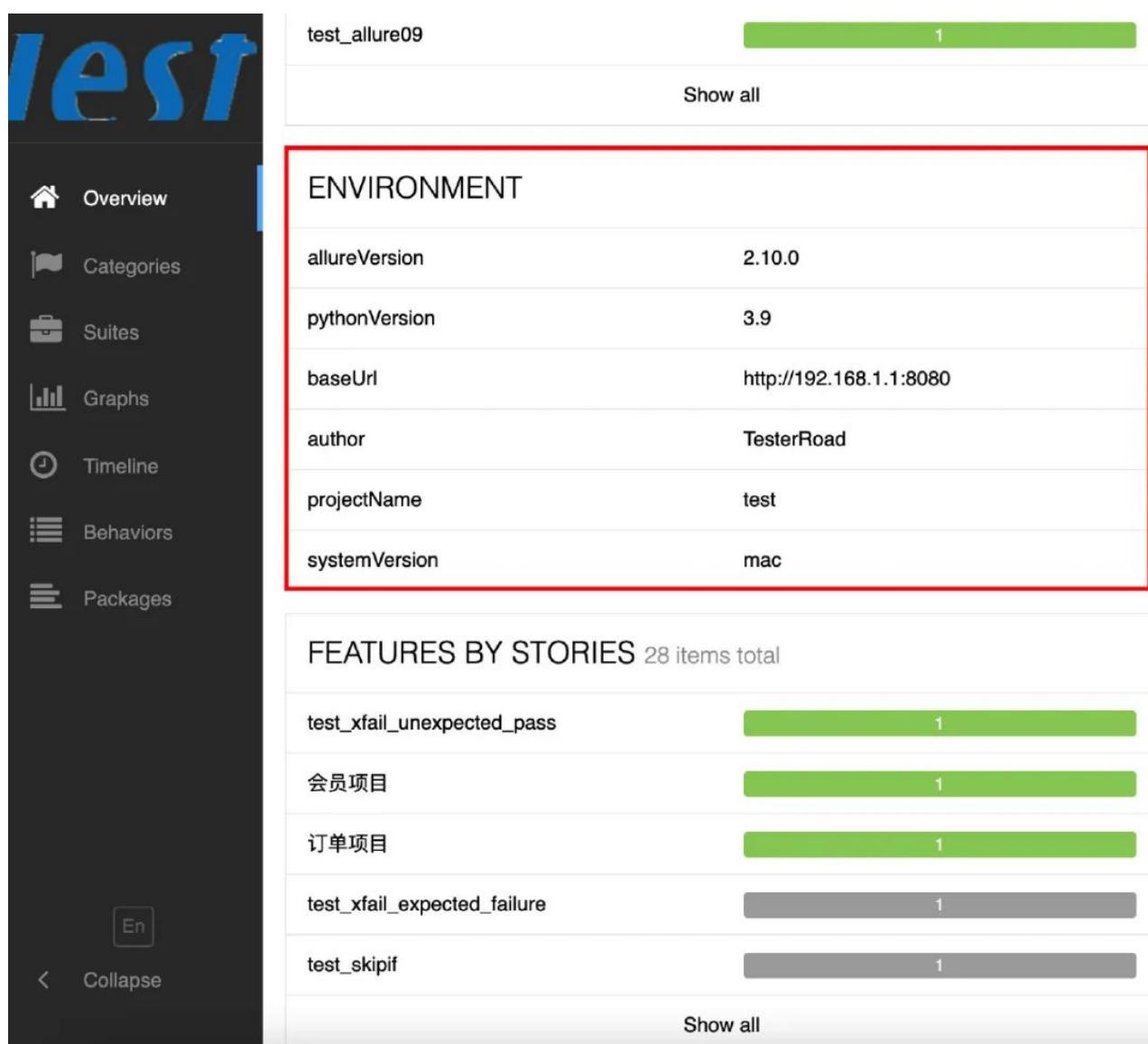
## 12.environment：环境信息

- 新建environment.properties文件
- 编写如下环境信息

```
systemVersion=mac  
pythonVersion=3.9  
allureVersion=2.10.0  
baseUrl=http://192.168.1.1:8080  
projectName=SIT  
author=TesterRoad
```

将environment.properties放在项目根目录或其它目录，在执行测试用例时将其复制到allure-results目录下

```
# coding=utf-8  
  
import pytest  
import os  
  
if __name__ == '__main__':  
    pytest.main(['-s', '-q', './', '--clean-alluredir', '--alluredir=allure-results'])  
    os.system('cp environment.properties ./allure-results/environment.properties')  
    os.system("allure generate -c -o allure-report")
```



The screenshot shows the Allure Test Opener interface. On the left is a sidebar with navigation links: Overview, Categories, Suites, Graphs, Timeline, Behaviors, and Packages. A language switcher at the bottom indicates 'En'. The main area displays a report for a test named 'test\_allure09'. At the top, there's a green progress bar with the number '1' and a 'Show all' button. Below this is a section titled 'ENVIRONMENT' which contains the following key-value pairs:

allureVersion	2.10.0
pythonVersion	3.9
baseUrl	http://192.168.1.1:8080
author	TesterRoad
projectName	test
systemVersion	mac

Below this is a section titled 'FEATURES BY STORIES' showing 28 items total. The first few items listed are: 'test\_xfail\_unexpected\_pass' (1), '会员项目' (1), '订单项目' (1), 'test\_xfail\_expected\_failure' (1), and 'test\_skipif' (1). A 'Show all' button is located at the bottom of this section.

## 八.Pytest执行所有.py文件

---

1.新建run.py文件，编写如下代码

```
# coding=utf-8

import pytest
import os

if __name__ == '__main__':
    pytest.main(['-s', '-q', './', '--clean-alluredir', '--alluredir=allure-
results'])
    os.system('cp environment.properties ./allure-results/environment.properties')
    os.system("allure generate -c -o allure-report")
```

2.查看报告



## 九.参考链接

[docs.qameta.io/allure-r](https://docs.qameta.io/allure-r)