Guide: What are FastAPI Query Parameters and How to Utilize Them?

FastAPI query parameters are a special type of query parameters that require the extra FastAPI and Query libraries to access them.

By Steven Ang Cheong Seng

6 min. read · View original

FastAPI is a modern, fast-performing web framework that can be used for building APIs with Python (ver. 3.8+) based on standard Python-type hints. With many developers relying on FastAPI for their very high-performing, standard-based Python framework (based on OpenAPI and the JSON schema), it is becoming more popular for it being a developer's API development tool of choice.

💡

Apidog is an all-in-one API development platform that offers functionalities to build APIs from scratch. Apidog facilitates the entire API lifecycle development as users can build, design, test, mock, and document APIs.

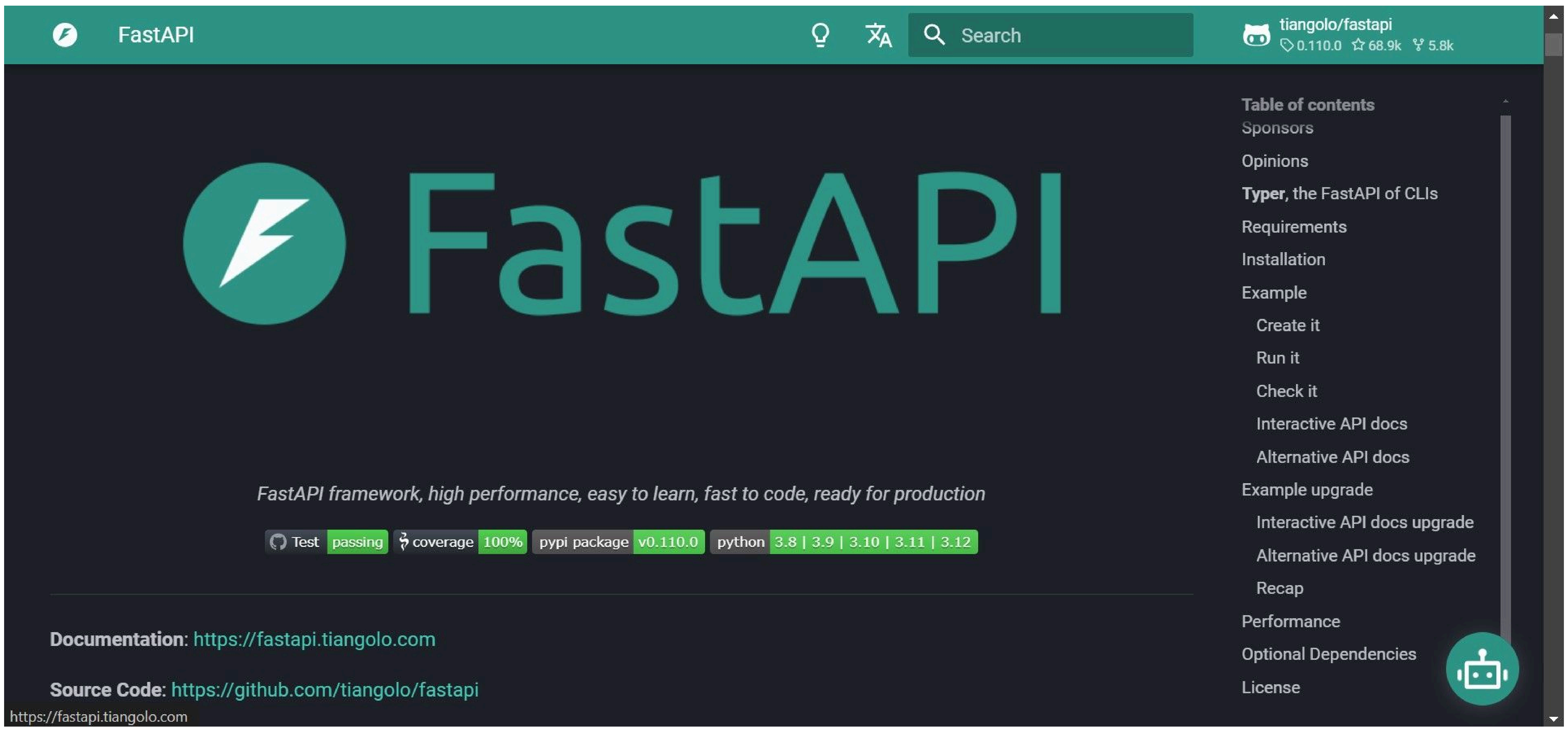
If you are looking for an alternative to your previous API tool, consider Apidog today - Apidog offers a simple and intuitive user interface that is extremely easy to use. Start using Apidog today by clicking on the button below! 👇 👇 👇

button

For APIs to work together with servers, clients would need to be able to communicate with APIs. Consequently, the client's side utilizes query parameters (and path parameters depending on the situation) to request certain data with the help of web addresses. This article will therefore elaborate more on what FastAPI query parameters are, and how you can utilize them for applications.

What are FastAPI Query Parameters?

FastAPI query parameters essentially refer to ordinary query parameters that are used in the context of the FastAPI framework. They are another method for the clients to pass additional information to API endpoints through a web address (or URL).



FastAPI framework, high performance, easy to learn, fast to code, ready for production

Documentation: https://fastapi.tiangolo.com

Source Code: https://github.com/tiangolo/fastapi

The FastAPI website

Key Features of FastAPI Query Parameters

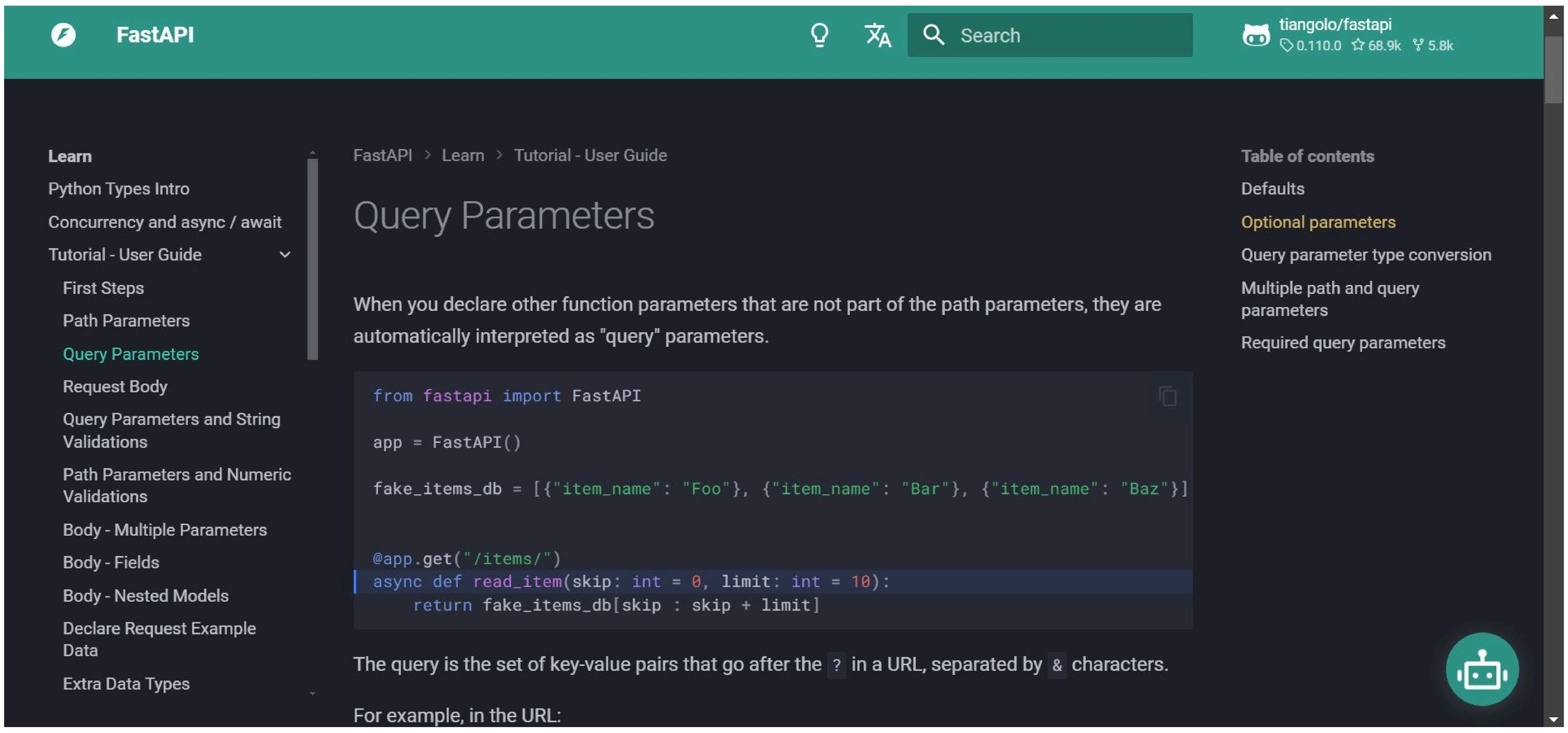
There are some slight variations on how you can access FastAPI query parameters when compared with other types of query parameters.

Declared in function parameters: Unlike path parameters, which are part of the URL path itself, query parameters are defined as function parameters within your API endpoint. Any parameter not explicitly declared as a path parameter will be treated as a query parameter.

Accessed through the `Query` function: To access query parameters within your endpoint function, you use the `Query` function from the `fastapi` library. This function can take several arguments, including:

- `default`: A default value to be used if the parameter is not provided in the request.
- `description`: A description of the parameter for documentation purposes.

Optional and required: Query parameters can be optional or required. By setting a default value, you make the parameter optional. Omitting the default value makes it mandatory for the client to provide the parameter in the request.


FastAPI
Search
tiangolo/fastapi
0.110.0 68.9k 5.8k
Learn
Python Types Intro
Concurrency and async / await
Tutorial - User Guide
First Steps
Path Parameters
Query Parameters
Request Body
Query Parameters and String Validations
Path Parameters and Numeric Validations
Body - Multiple Parameters
Body - Fields
Body - Nested Models
Declare Request Example Data
Extra Data Types
FastAPI > Learn > Tutorial - User Guide
Table of contents
Defaults
Optional parameters
Query parameter type conversion
Multiple path and query parameters
Required query parameters

Query Parameters

When you declare other function parameters that are not part of the path parameters, they are automatically interpreted as "query" parameters.

```
from fastapi import FastAPI

app = FastAPI()

fake_items_db = [{"item_name": "Foo"}, {"item_name": "Bar"}, {"item_name": "Baz"}]


@app.get("/items/")
async def read_item(skip: int = 0, limit: int = 10):
    return fake_items_db[skip : skip + limit]
```

The query is the set of key-value pairs that go after the `?` in a URL, separated by `&` characters.

For example, in the URL:

FastAPI's query parameter guide

Example of How to use FastAPI Query Parameters (Code Snippets Included)

[Note that the code snippets provided in this section require modification, as they may not be suited to your coding's requirements]

1. Pagination

You can utilize FastAPI query parameters to separate large amounts of records or items and break them down into smaller groups. This can prevent overwhelming the user with too much information all at once.

```
from fastapi import FastAPI, Query

app = FastAPI()

@app.get("/items")
async def get_items(skip: int = Query(default=0), limit: int = Query(default=10)):
    # Logic to retrieve items from database based on skip and limit values
    return items
```

Additional Explanation:

In this code example, `skip` and `limit` are query parameters. By default, it will return the first 10 items (limited by `limit`). A possible way to modify these values in the URL would be something like:

- `http://localhost:8000/items?skip=10&limit=20`
  This will retrieve items from index 10 (skipping the first 10) to index 29 (limited by 20).

2. Filtering

FastAPI query parameters can be utilized for filtering data based on specific criteria.

The code example below demonstrates a FastAPI query parameter that retrieves products.

```
from fastapi import FastAPI, Query

app = FastAPI()

@app.get("/products")
async def get_products(category: str | None = Query(default=None), price_from: float | None = Query(default=None), price_to: float | None =
Query(default=None)):
    # Logic to filter products based on category, price range
    return products
```

Additional Explanation:

Users can choose to specify a `category` and a price range (`price_from` and `price_to`) to filter the list of products returned.

3. Sorting

Many developers have also found FastPI query parameters suitable for sorting data.

```
from fastapi import FastAPI, Query

app = FastAPI()

@app.get("/users")
async def get_users(sort_by: str = Query(default="name", choices=["name", "created_at"])):
    # Logic to sort users based on the provided sort_by parameter
    return users
```

Additional Explanation:

Users can choose how to sort the list of users returned by the API by either `name` or `created_at` using the `sort_by` query parameter.

4. Searching

FastAPI query parameters are found to be useful for implementing search functionalities in many web applications.

With FastApiquery parameters, you can quickly capture a search term with the following code:

```
from fastapi import FastAPI, Query

app = FastAPI()

@app.get("/search")
async def search(q: str = Query(default="")):
    # Logic to search for resources based on the search term (q)
    return search_results
```

Additional Explanation:

Users can search for specific resources by providing a search term in the q parameter.

To find out more on how you can work with FastAPI query parameters, visit their website.

Apidog - Superior Alternative to FastAPI

An excellent alternative for developing APIs to FastAPI is Apidog.

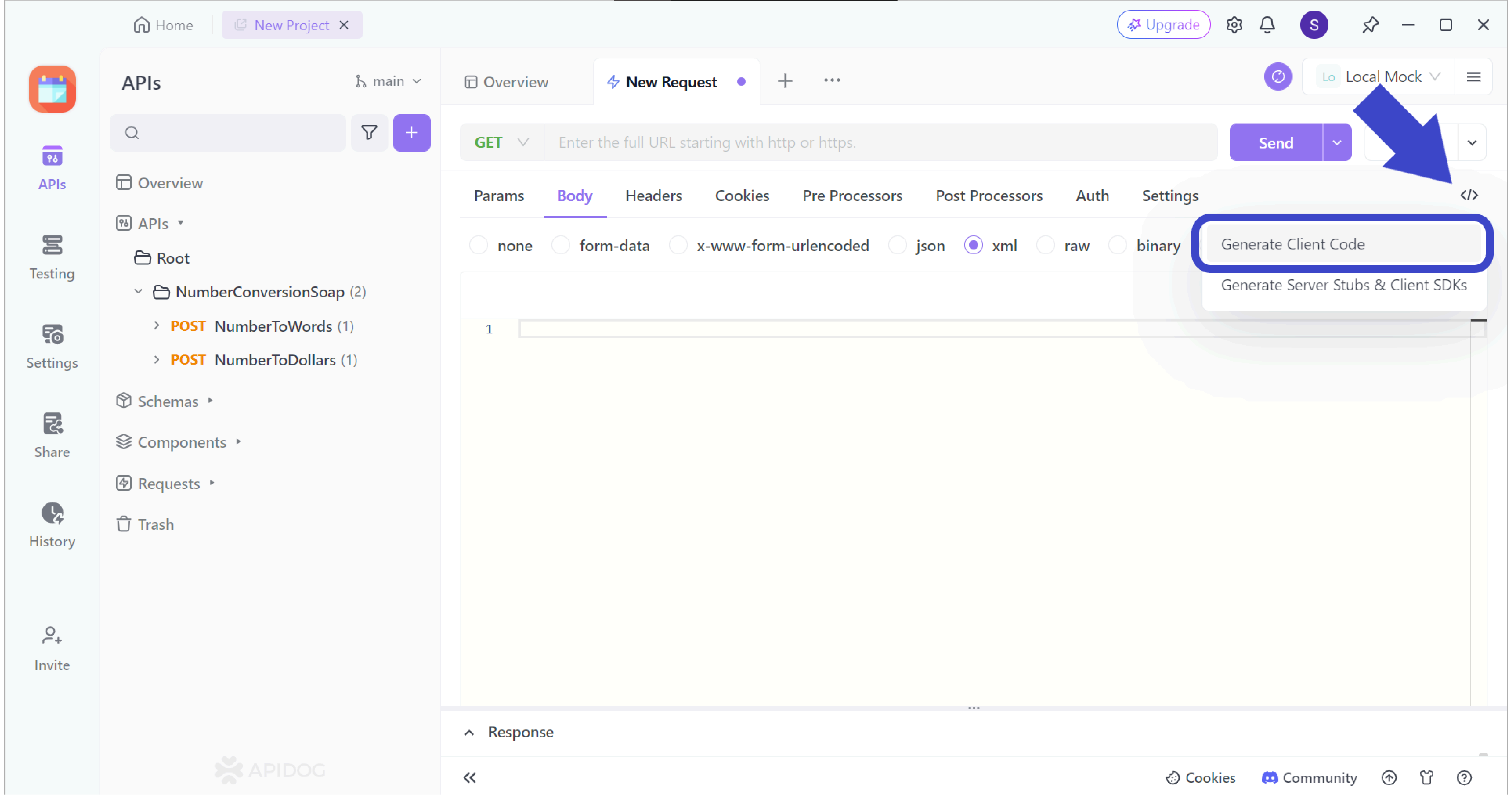
Apidog and FastAPI share a lot of beneficial features, such as an easy and intuitive user interface and web-industry-level standards. However, what allows Apidog to stand out are the additional features that FastAPI does not possess.

Apidog
All-in-one API Development Platform
API
Developing & Debugging
Backend Developer
API
Mocking & Calling
Frontend Developer
API Design
Specification
API
Documentation
API Designer
API
Automated Testing
QA Engineer
It automatically syncs with other modules in REAL TIME

All of Apidog's functionalities: building, testing, mocking, and documenting.

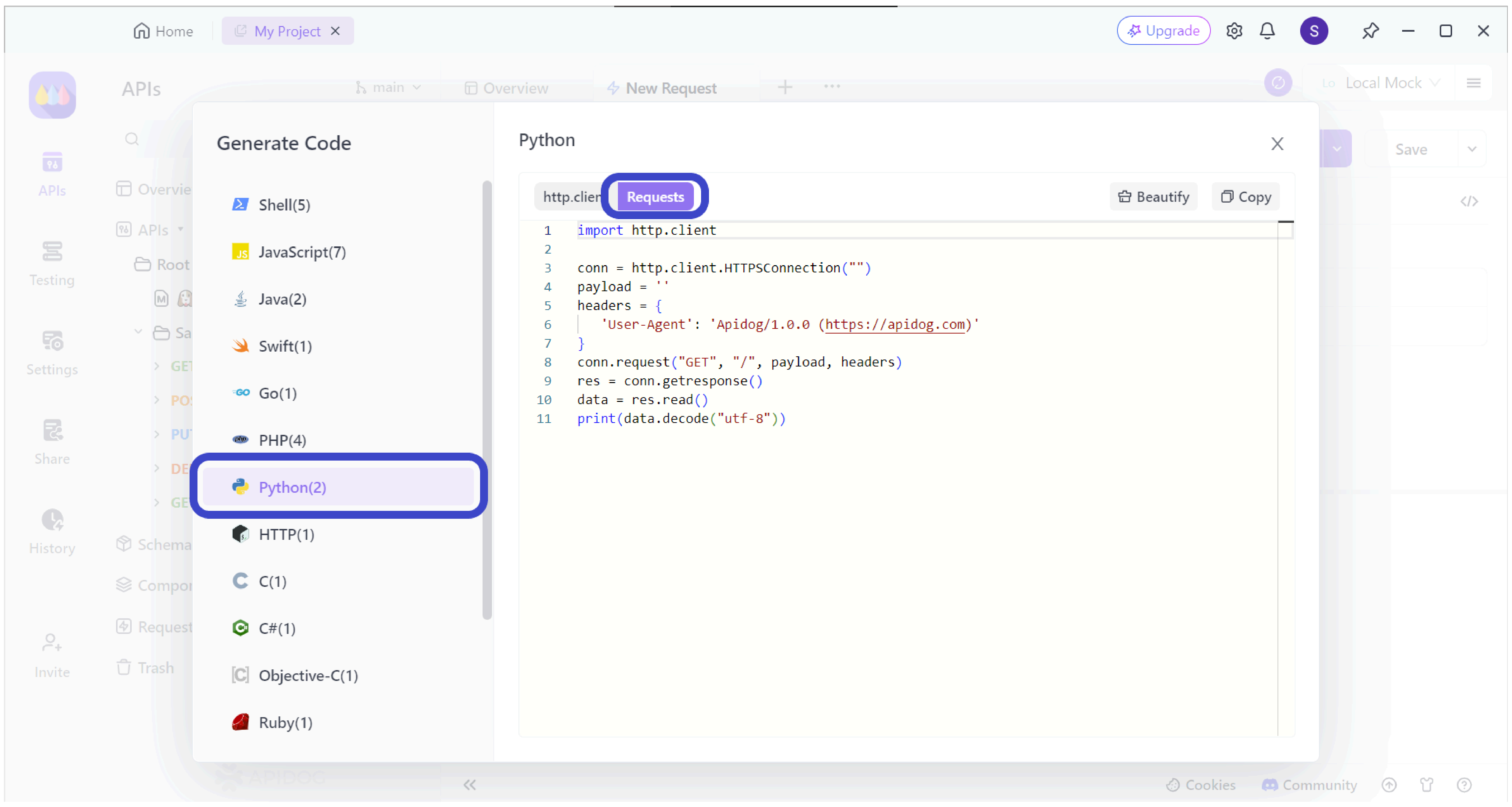
Automated Client Code Generation Using Apidog

Apidog supports both new and experienced API developers by facilitating client code generation in multiple programming languages.



Button for code generation on Apidog

Locate the `</>` button found on the top right corner of the Apidog window. After pressing this button, click on `Generate Client Code` to continue.



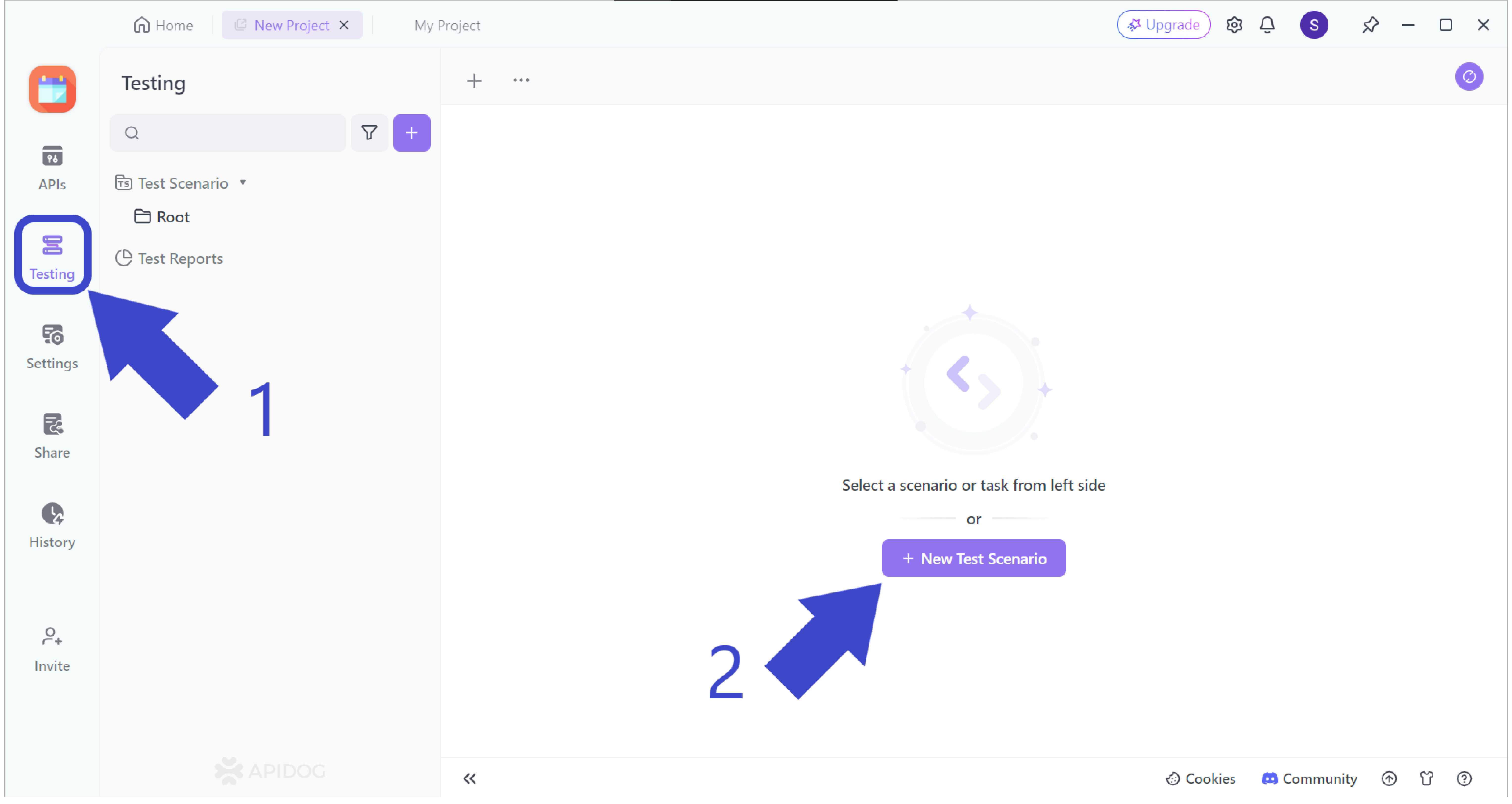
```
import http.client

conn = http.client.HTTPSConnection("")
payload = ''
headers = {
  'User-Agent': 'Apidog/1.0.0 (https://apidog.com)'
}
conn.request("GET", "/", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Generating Python client code with Apidog

From the image above, you can see a code shell that provides ready-to-use code. All you need to do is copy and paste the code to your coding platform. You can generate code in numerous programming languages, such as Python, JavaScipt, Shell, and Java.

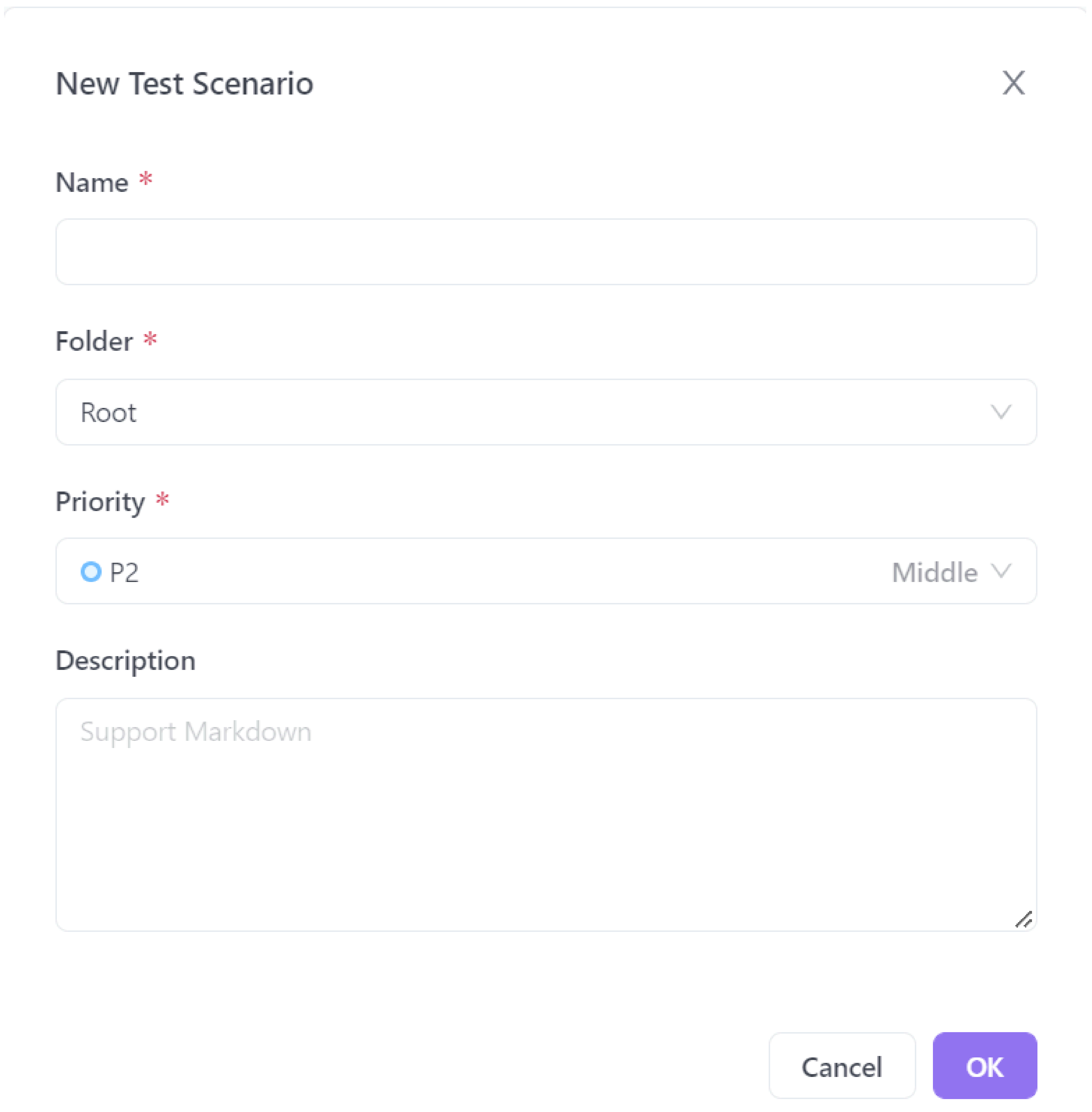
Real-world Simulation Testing Scenarios Using Apidog

Apidog allows you to simulate real-world scenarios by adding multiple steps to a test case, and selecting an appropriate environment for your APIs.



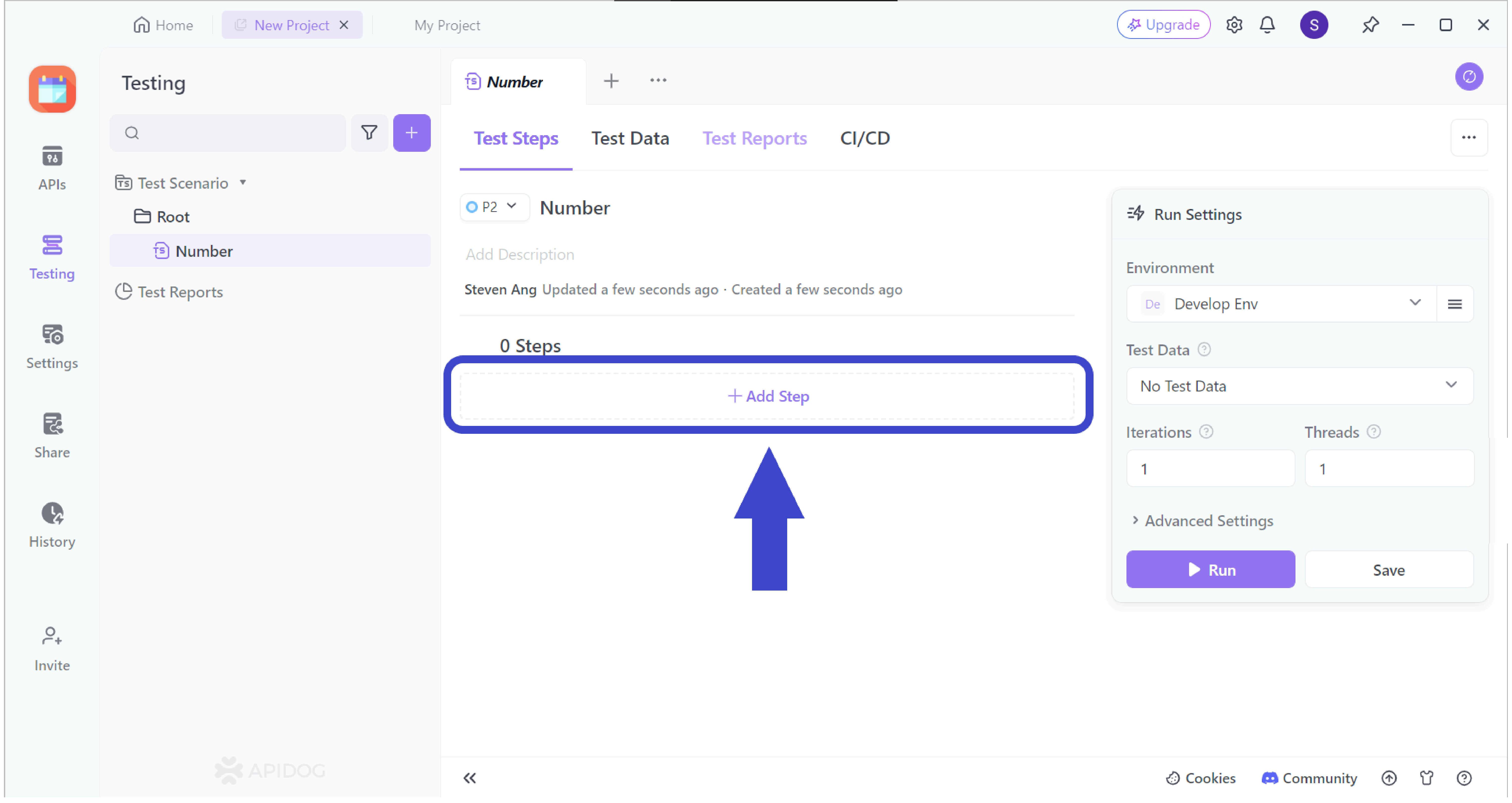
Initializing a new test scenario on Apidog

Firstly, locate the `Testing` button pointed out by Arrow 1 in the image above. You should then see `+ New Test Scenario`, pointed out by Arrow 2.



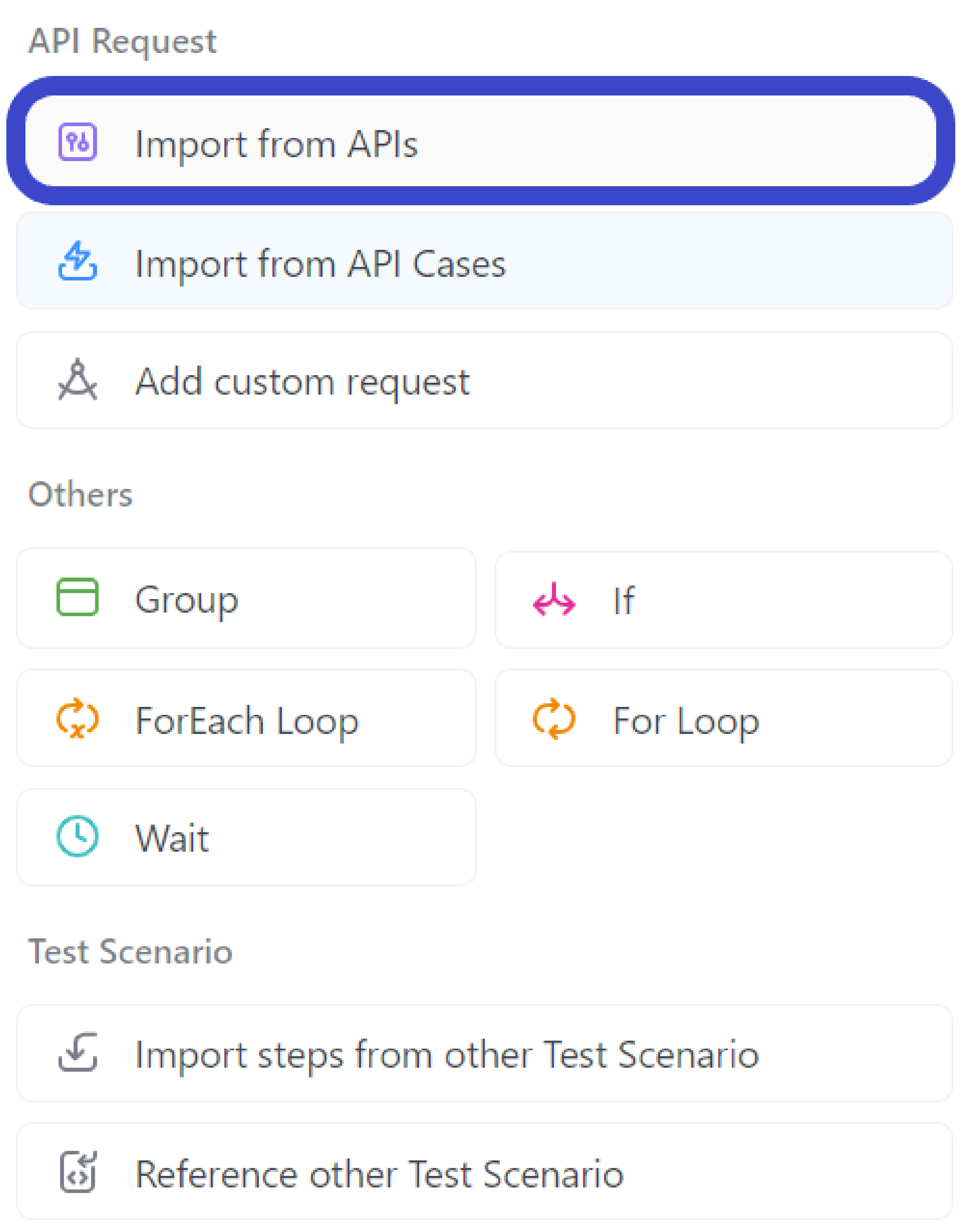
Fill description for the new test scenario

Fill in the necessary details regarding your test scenario here. Make sure that the names are self-explanatory so you do not have to second-guess yourself in case you forget what the test scenario is about.



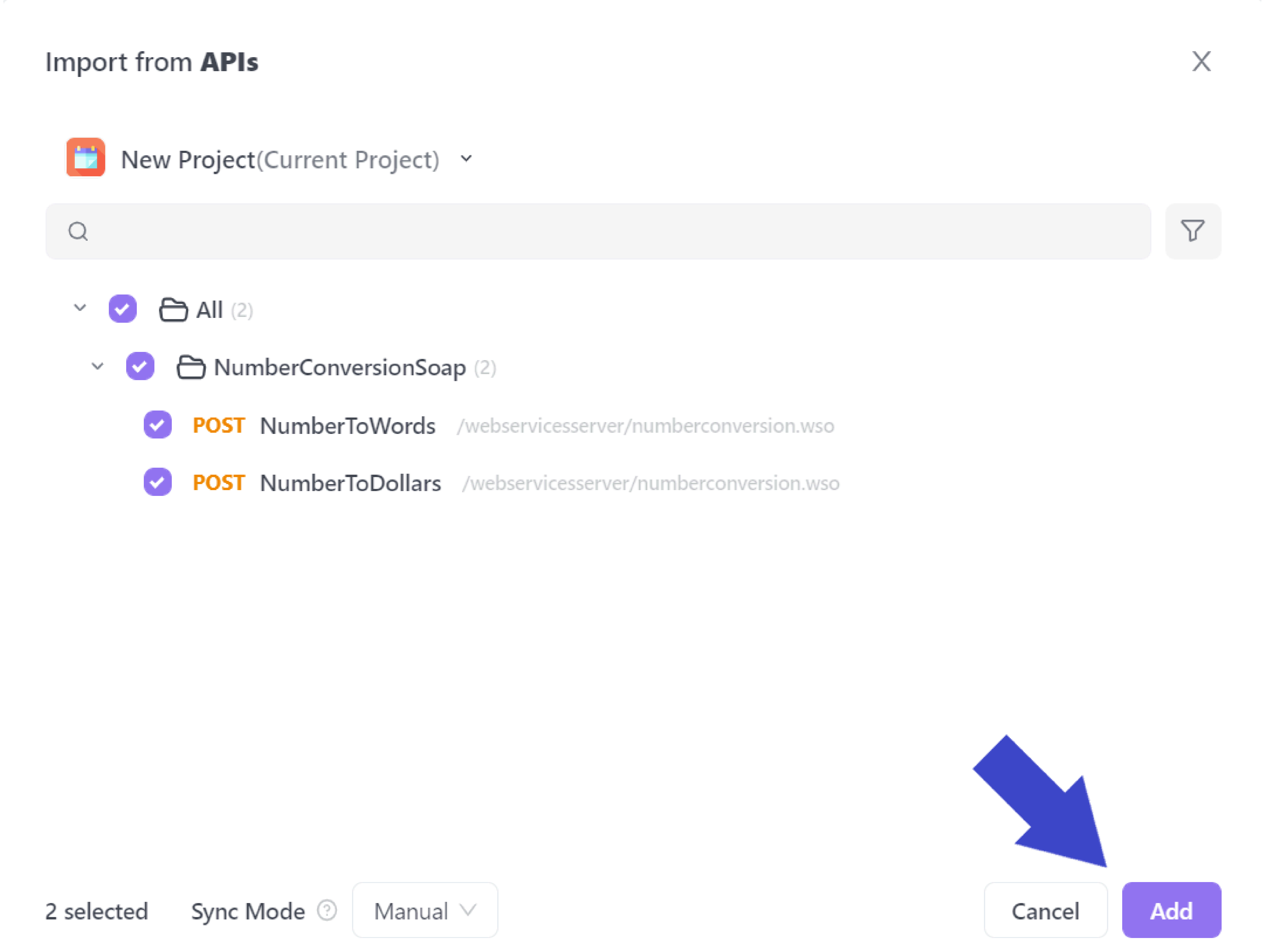
Adding step(s) to the test scenario

Add a step(s) to your test scenarios by clicking on the "Add Step" section. You should be able to see the image below.



Select "Import from APIs"

Select `Import from API` from the drop-down menu.


Import from **APIs**

New Project(Current Project)

All (2)

NumberConversionSoap (2)

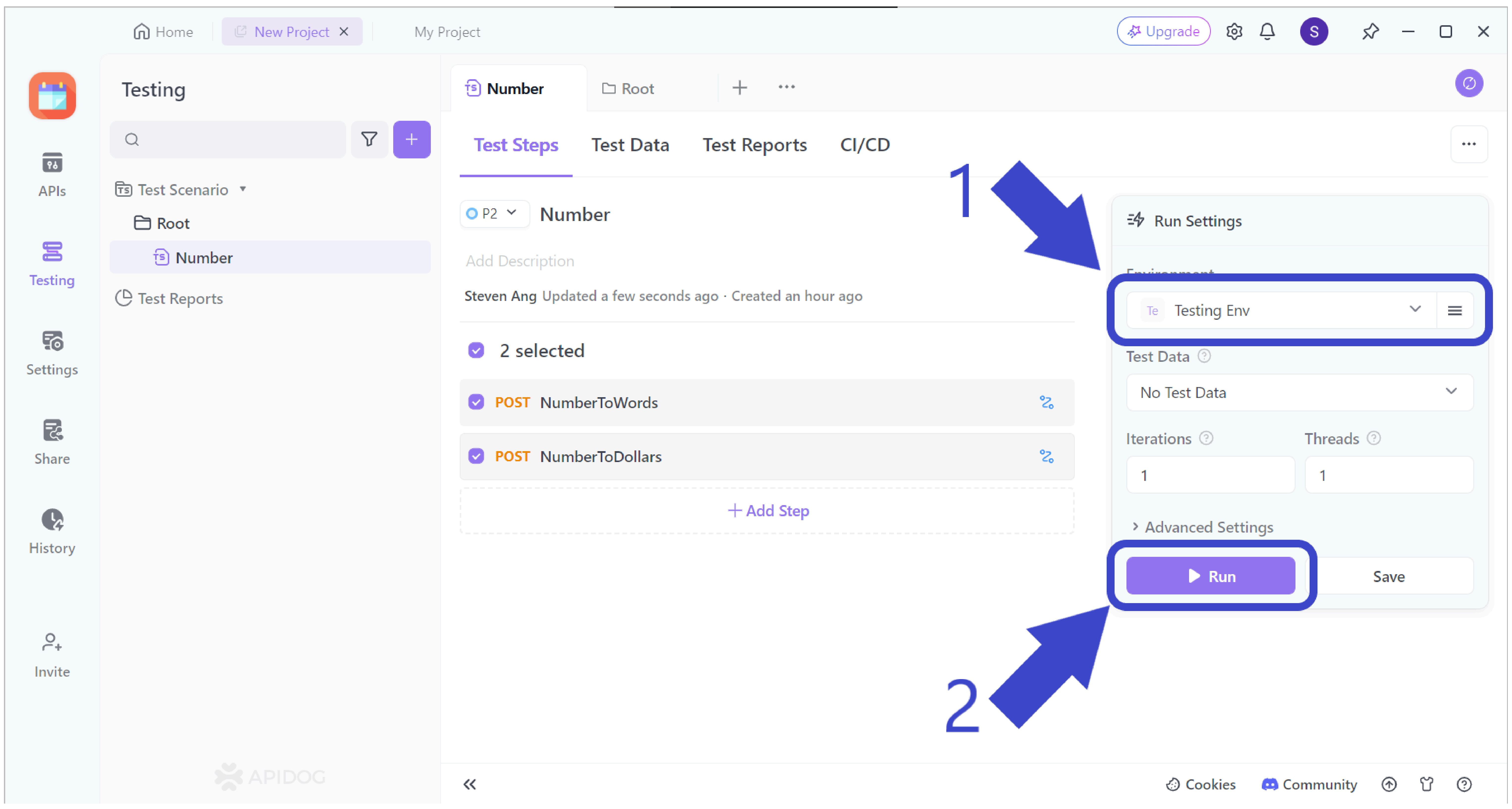
POST NumberToWords /webservicesserver/numberconversion.wso

POST NumberToDollars /webservicesserver/numberconversion.wso

2 selected Sync Mode Manual

Add all the APIs to include in your test scenario

Select all the APIs you would like to include in your test scenario. In the example above, the API called `NumberConversionSOAP` has been included.



P2 Number

Add Description

Steven Ang Updated a few seconds ago · Created an hour ago

2 selected

POST NumberToWords

POST NumberToDollars

Run Settings

Environment: Testing Env

Test Data: No Test Data

Iterations: 1 Threads: 1

Advanced Settings

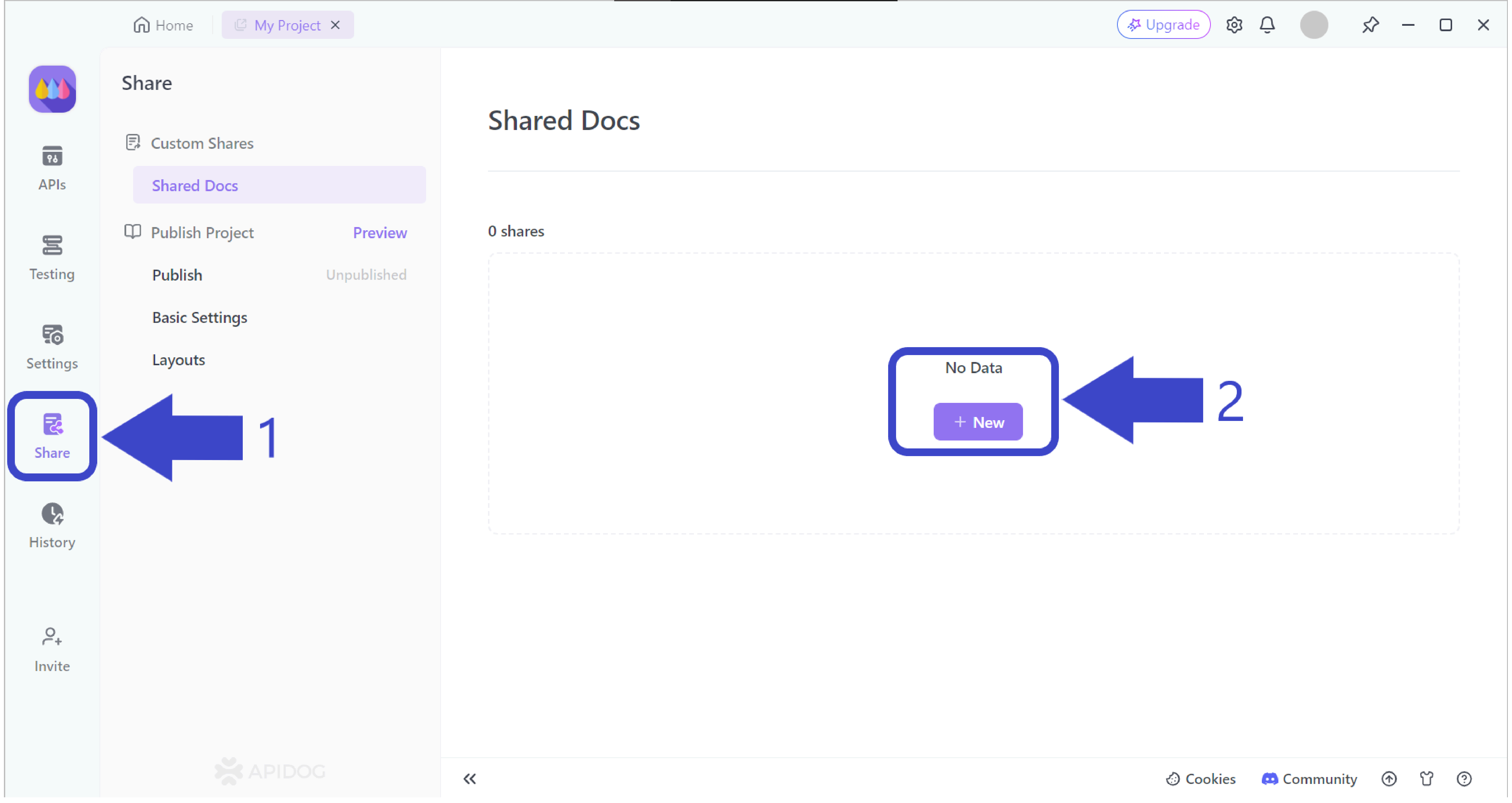
Set Environment to "Testing Env" and hit "Run" to start testing

Before hitting the Run button to start your test scenario, make sure to change the test scenario environment, which should be `Testing Env`, as pointed out by Arrow 1.

Give it a try, and you can see whether your API can meet all your requirements!

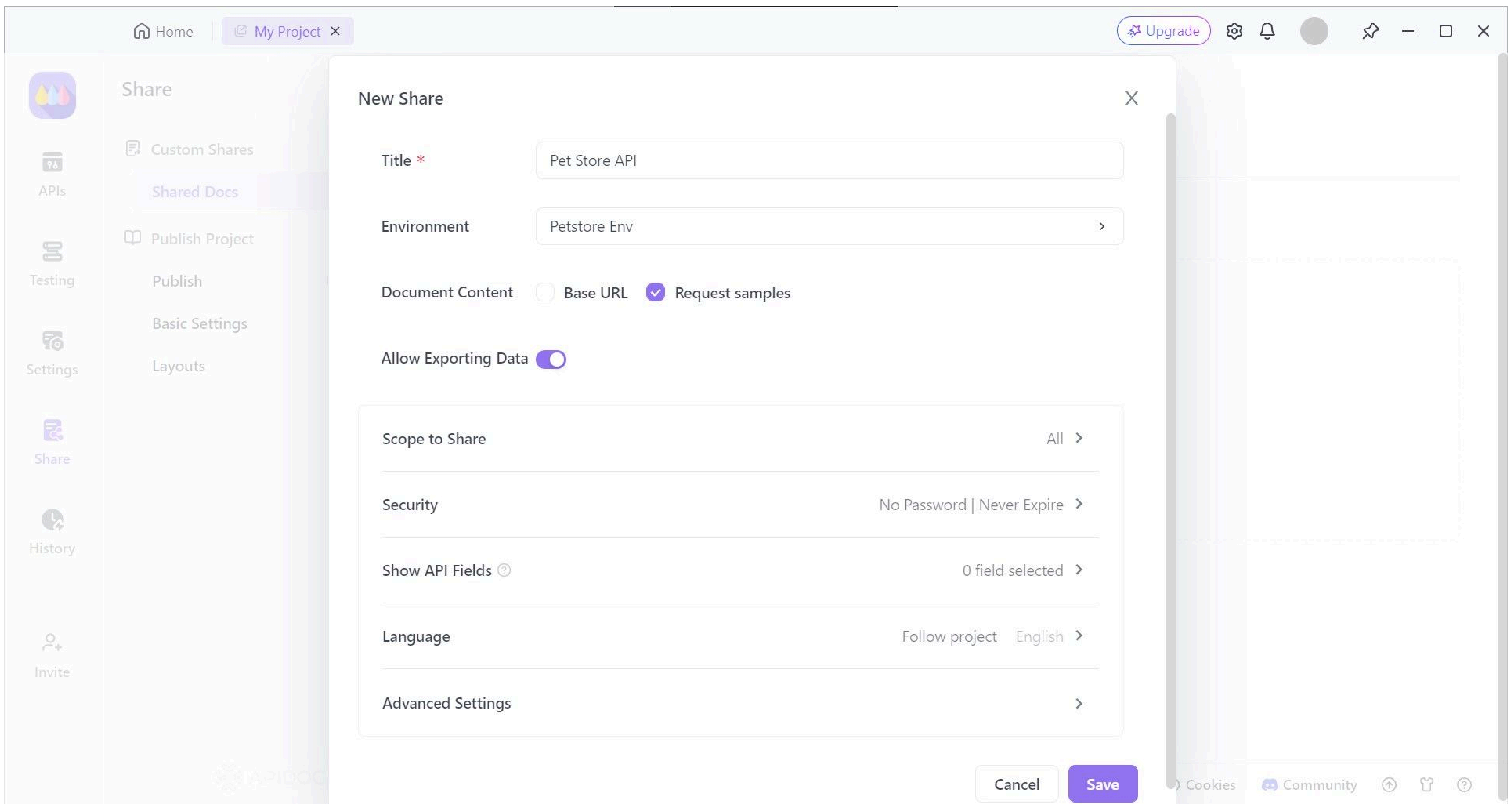
Generating Documentation For Requests in Apidog

Apidog supports API documentation generation for API developers just based on what they have done in their projects.



REST API Documentation is Sharable on Apidog

To begin, first, find the "Share" button as pointed out by Arrow 1, and press the "+ New" button as pointed out by Arrow 2


New Share

Title * Pet Store API

Environment Petstore Env

Document Content Base URL [✓ Request samples]

Allow Exporting Data [✓]

Scope to Share All

Security No Password | Never Expire

Show API Fields 0 field selected

Language Follow project English

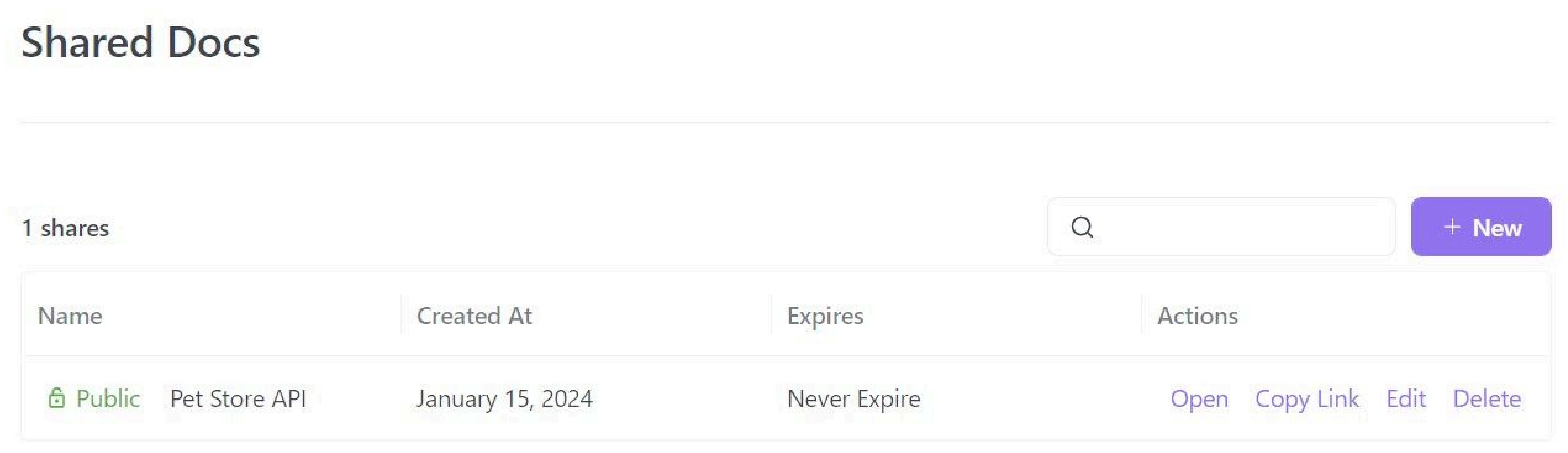
Advanced Settings

Input and Select API Documentation Properties with Apidog

Apidog provides the option to choose who can view your API documentation as well as set a file password, so only chosen individuals or organizations can view it.

Once all required fields like API documentation name and language have been filled, hit `Enter` on your keyboard or the Save button.

View or Share Your API Documentation


Shared Docs

1 shares

Name	Created At	Expires	Actions
Public Pet Store API	January 15, 2024	Never Expire	Open Copy Link Edit Delete

API Documentation is ready for viewing

Apidog compiles your API project's details into an API documentation that is viewable through a website URL. All you have to do is click the `Copy Link` link under `Actions`, and paste the URL on your favorite browser to view your API Documentation!

If you are interested, read this article on how to generate API documentation using Apidog.

Conclusion

FastAPI query parameters are a special type of query parameter that can be used with FastAPI APIs. Similar to other query parameters, they carry out the same functionalities, such as pagination, filtering, searching, and sorting. Unlike JavaScript query parameters They differ in the way they are accessed - you need to import the FastAPI and Query libraries before you can implement FastAPI query parameters.

Apidog is a powerful substitute for FastAPI as an API development tool. With impressive and useful features such as client code generation, automated API documentation, and testing scenarios, Apidog is the only application you need to download.