

Introduction to Swagger & Open API



Fall 2020, CSCI-GA 2820, Graduate Division, Computer Science

Instructor:
John J Rofrano

Senior Technical Staff Member | DevOps Champion
IBM T.J. Watson Research Center
rofrano@cs.nyu.edu (@JohnRofrano) 



Source for the Lab

- The code for this lab can be found here:

<https://github.com/nyu-devops/lab-flask-restplus-swagger>

```
$ git clone https://github.com/nyu-devops/lab-flask-restplus-swagger.git  
$ cd lab-flask-restplus-swagger  
$ vagrant up  
$ vagrant ssh
```

What Will You Learn?

- Gain a good overview of Swagger / Open API
 - Understand how to use Swagger tags to document your Python code
 - Use Flask-RESTPlus to deploy your Swagger documentation



“Any API worth writing ...is worth documenting !”

Why is Documentation Important?

Why is Documentation Important?

- Without documentation, no one will know how to call your API

Why is Documentation Important?

- Without documentation, no one will know how to call your API
- If no one calls your API, it has no reason to exist

Why is Documentation Important?

- Without documentation, no one will know how to call your API
- If no one calls your API, it has no reason to exist
- If your API has no reason to exist, you will be unemployed

Why is Documentation Important?

- Without documentation, no one will know how to call your API
- If no one calls your API, it has no reason to exist
- If your API has no reason to exist, you will be unemployed
- When you are unemployed, you will wish you had written documentation for your API ;-)

Swagger and Open API



Open-API? Swagger?



OPEN API
INITIATIVE

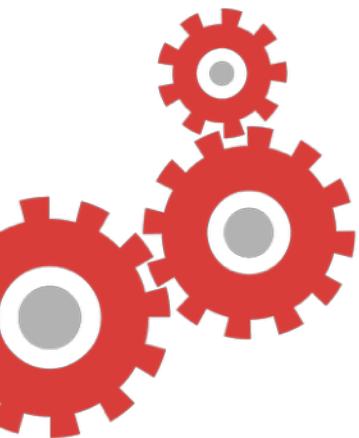
=

Specification



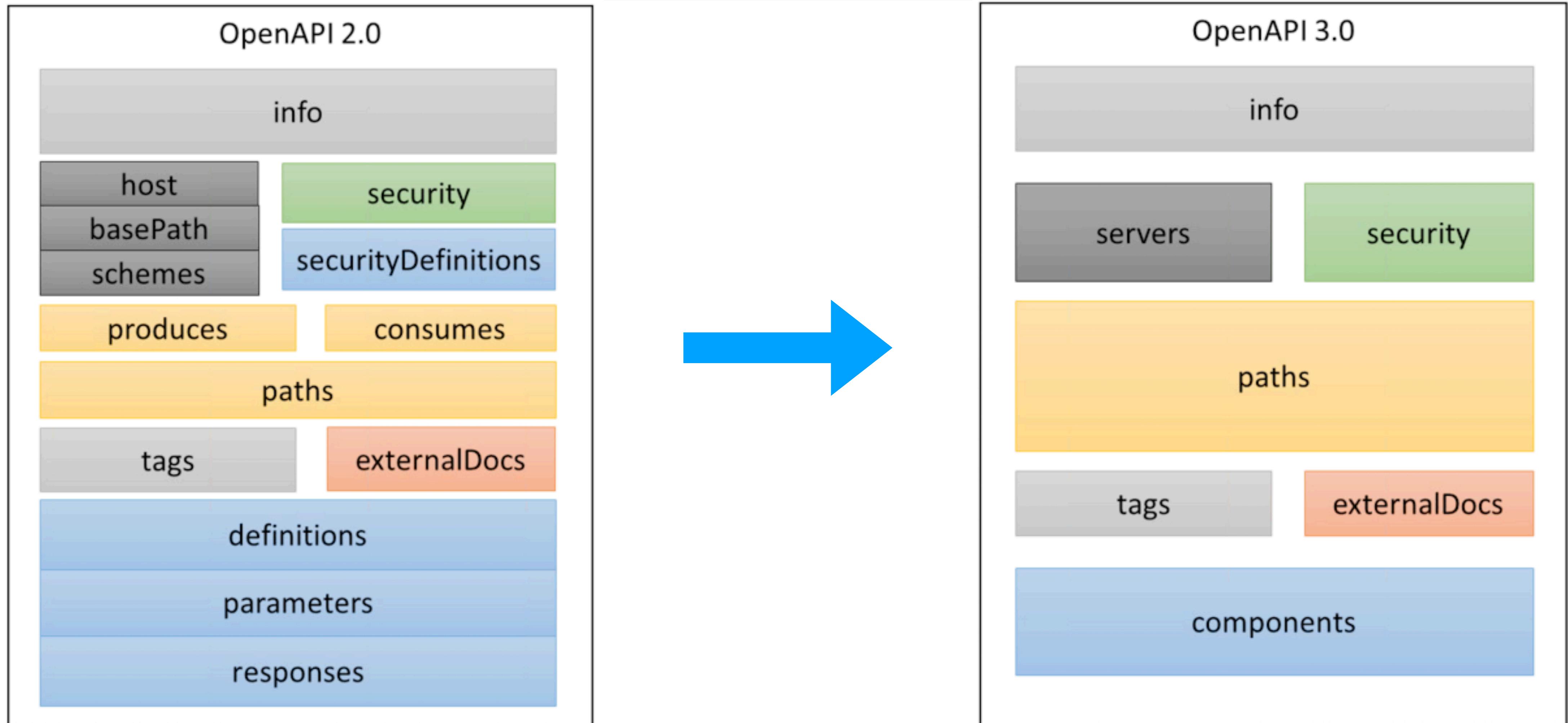
swagger

=



Tools

Open API 2.0 vs Open API 3.0



What is Swagger?

- An open-source framework for designing and describing REST APIs
 - Equally suitable for both designing new APIs and documenting your existing APIs
- A Swagger specification describes the API endpoint, available resources and operations that can be called against these resources.

```
swagger: '2.0'  
info:  
  title: Sample API  
  version: '1.0'  
host: api.example.com  
paths:  
  /hello:  
    get:  
      produces:  
        - application/json  
      responses:  
        200:  
          description: OK
```

Why Swagger?

- The simple YAML format is human-readable, machine-readable and self-explanatory
- There is a variety of tools built around it that facilitate API development
 - Swagger Codegen can generate server code and client SDKs based on a Swagger specification
 - Swagger UI can render Swagger specifications to interactive API documentation, like the one at <http://petstore.swagger.io>
 - These are just a few examples of how Swagger can help API developers

API First Approach

- Design the API First and generate a Mock
- Use Python Library like Connexion to Implement
- Connexion allows the code to comply with the API !



Example using Connexion with External Yaml

- One way to add Swagger docs to your application is to use connexion
- Create your docs as a yaml file
- Place your swagger yaml file in a folder called ./swagger and:

```
import connexion

app = connexion.App(__name__, specification_dir='swagger/')
app.add_api('my_api.yaml')
app.run(port=8080)
```

Problems with using External Documentation

- A separate thing for developers to update
- Too easy for developers to change the code and forget to change the documentation
- Requires developers to look in a separate place for documentation instead of being in-line with the code



Example Swagger Docs

The screenshot shows a web browser window displaying the Flasgger UI for a "DevOps Swagger Pet App". The URL in the address bar is `localhost:5000/apidocs/index.html?url=/v1/spec`. The page title is "Flasgger". The main content area is titled "DevOps Swagger Pet App" and describes it as a "sample server Petstore server".

A red arrow points from the text "Click for Details" to the first item in the "Pets" section, which is a `GET /pets` operation.

The "Pets" section contains the following operations:

- `GET /pets` (blue button) - **Retrieve a list of Pets**
- `POST /pets` (green button) - **Creates a Pet**
- `DELETE /pets/{id}` (red button) - **Delete a Pet**
- `GET /pets/{id}` (blue button) - **Retrieve a single Pet**
- `PUT /pets/{id}` (orange button) - **Update a Pet**

At the bottom of the page, there is a note: "[BASE URL: , API VERSION: 1.0.0]" and "[Powered by [Flasgger](#)]".

Code Mapping

Open API endpoint specified as:

```
paths:  
  /pets:  
    get:  
      operationId: api.get_pets  
      parameters:  
        - name: category  
          description: Pet category  
          in: query  
          type: string  
          required: true
```

Python view function:

```
# api.py file  
  
def get_pets(category):  
    pets = Pet.find_by_category(category)  
    return [pet for pet in pets], 200
```

Code Mapping

Open API endpoint specified as:

```
paths:  
  /pets:  
    get:  
      operationId: api.get_pets  
parameters:  
  - name: category  
    description: Pet category  
    in: query  
    type: string  
    required: true
```

Maps the operationID to your code new function:

```
# api.py file  
  
def get_pets(category):  
    pets = Pet.find_by_category(category)  
    return [pet for pet in pets], 200
```

Details on Each Call

The screenshot shows a web browser window displaying the Swagger UI for a REST API endpoint. The URL in the address bar is `localhost:5000/apidocs/index.html?url=/v1/spec#/!/_get_pets`. The main content area shows a **GET /pets** operation with the description **Retrieve a list of Pets**. Below this, there is an **Implementation Notes** section stating: "This endpoint will return all Pets unless a query parameter is specified". Under the **Response Class (Status 200)**, there are two tabs: **Model** and **Model Schema**. The **Model Schema** tab displays the following JSON schema:

```
[  
  {  
    "category": "string",  
    "id": 0,  
    "name": "string"  
  }  
]
```

Below the schema, the **Response Content Type** is set to `application/json`. The **Parameters** section contains two entries:

Parameter	Value	Description	Parameter Type	Data Type
category	<input type="text"/>	the category of Pet you are looking for	query	string
name	<input type="text"/>	the name of Pet you are looking for	query	string

Try It Out

The screenshot shows a web browser window with the URL `localhost:5000/apidocs/index.html?url=/v1/spec#/Pets/post_p`. The browser has a standard OS X-style interface with red, yellow, and green buttons in the top-left corner, and various icons in the top-right corner.

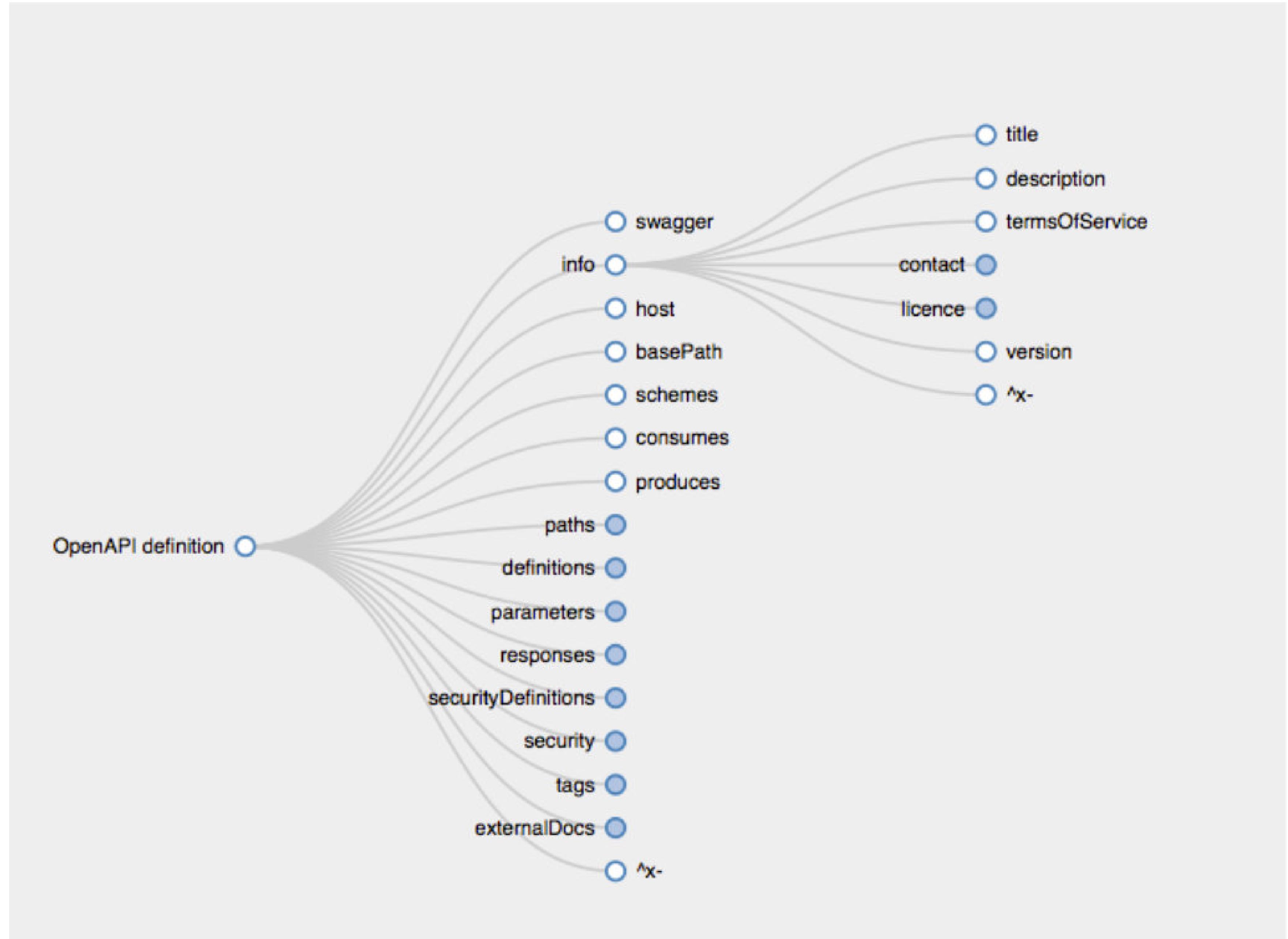
The main content area displays the 'Try It Out' interface for a `Pets` endpoint. It includes the following sections:

- Curl:** A code block containing the command: `curl -X GET --header "Accept: application/json" "http://localhost:5000/pets"`.
- Request URL:** A text input field containing the URL: `http://localhost:5000/pets`.
- Response Body:** A large text area showing a JSON array response:

```
[  
  {  
    "category": "cat",  
    "id": 1,  
    "name": "kitty"  
  }  
]
```
- Response Code:** A text input field containing the status code: `200`.
- Response Headers:** A large text area showing a JSON object response:

```
{  
  "server": "Werkzeug/0.12.1 Python/2.7.12",  
  "Date": "Mon, 12 Mar 2012 13:21:21 GMT",  
  "Content-Type": "application/json; charset=UTF-8",  
  "Content-Length": "113",  
  "Connection": "close",  
  "Access-Control-Allow-Origin": "*"  
}
```

Open-API Definition



Basic Information

```
swagger: '2.0'
info:
  description: |
    This is a sample server Petstore server. You can find
    out more about us at
    [http://mydomain.com](http://mydomian.com)
  version: 1.0.0
  title: DevOps Swagger Pet App
  termsOfService: http://mydomain.com/terms/
  contact:
    email: apiteam@mydomain.com
  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html
```

tags are for Organizing

```
tags:  
- name: Pets  
  description: Everything about your Pets  
  externalDocs:  
    description: Find out more  
    url: http://petdemo.mybluemix.net  
- name: Store  
  description: Access to Petstore orders  
- name: Users  
  description: Operations about user  
  externalDocs:  
    description: Find out more about our store  
    url: http://petdemo.mybluemix.net
```

Path: POST / pets

```
paths:
  /pets:
    post:
      tags:
        - Pets
      summary: Add a new pet to the store
      operationId: add_pet
      consumes:
        - application/json
      produces:
        - application/json
      parameters:
        - in: body
          name: body
          description: Pet object that needs to be added to the store
          required: true
          schema:
            $ref: '#/definitions/Pet'
      responses:
        405:
          description: Invalid input
      security:
        - petstore_auth:
          - write:pets
          - read:pets
```

Path: GET /pets/ {id}

```
/pets/{petId}:
  get:
    tags:
      - Pets
    summary: Find pet by ID
    description: Returns a single pet
    produces:
      - application/json
    parameters:
      - name: pet_id
        in: path
        description: ID of pet to return
        required: true
        type: integer
        format: int64
    responses:
      200:
        description: successful operation
        schema:
          $ref: '#/definitions/Pet'
      400:
        description: Invalid ID supplied
      404:
        description: Pet not found
    security:
      - api_key: []
```

PATH: POST / pets

- This accepts a POST form data from a HTML form

```
post:  
  tags:  
    - Pets  
  summary: Updates a pet in the store with form data  
  consumes:  
    - application/x-www-form-urlencoded  
  produces:  
    - application/json  
  parameters:  
    - name: pet_id  
      in: path  
      description: ID of pet that needs to be updated  
      required: true  
      type: integer  
      format: int64  
    - name: name  
      in: formData  
      description: Updated name of the pet  
      required: false  
      type: string  
    - name: status  
      in: formData  
      description: Updated status of the pet  
      required: false  
      type: string  
  responses:  
    405:  
      description: Invalid input  
  security:  
    - petstore_auth:  
      - write:pets  
      - read:pets
```

Path: **DELETE / pets{id}**

```
delete:
  tags:
    - Pets
  summary: Deletes a pet
  produces:
    - application/json
  parameters:
    - name: api_key
      in: header
      required: false
      type: string
    - name: pet_id
      in: path
      description: Pet id to delete
      required: true
      type: integer
      format: int64
  responses:
    204:
      description: Empty body
  security:
    - petstore_auth:
        - write:pets
        - read:pets
```

Path: PUT /pets

```
put:
  tags:
    - Pets
  summary: Update an existing pet
  consumes:
    - application/json
  produces:
    - application/json
  parameters:
    - in: body
      name: body
      description: Pet object that needs to be added to the store
      required: true
      schema:
        $ref: '#/definitions/Pet'
  responses:
    400:
      description: Invalid ID supplied
    404:
      description: Pet not found
    405:
      description: Validation exception
  security:
    - petstore_auth:
        - write:pets
        - read:pets
```

Definitions

```
definitions:  
  Pet:  
    type: object  
    required:  
      - name  
      - photoUrls  
    properties:  
      id:  
      category:  
      name:  
      photoUrls:  
      tags:  
      status:
```

```
  id:  
    type: integer  
    format: int64  
  category:  
    $ref: '#/definitions/Category'  
  name:  
    type: string  
    example: doggie  
  photoUrls:  
    type: array  
    xml:  
      name: photoUrl  
      wrapped: true  
  items:  
    type: string
```

```
Category:  
  type: object  
  properties:  
    id:  
      type: integer  
      format: int64  
    name:  
      type: string  
    xml:  
      name: Category
```

```
tags:  
  type: array  
  xml:  
    name: tag  
    wrapped: true  
  items:  
    $ref: '#/definitions/Tag'  
status:  
  type: string  
  description: pet status in the store  
enum:  
  - available  
  - pending  
  - sold
```

Security Definitions

```
securityDefinitions:  
  petstore_auth:  
    type: oauth2  
    authorizationUrl: http://petstore.swagger.io/oauth/dialog  
    flow: implicit  
    scopes:  
      write:pets: modify pets in your account  
      read:pets: read your pets  
  api_key:  
    type: apiKey  
    name: api_key  
    in: header
```



What is SwaggerHub?

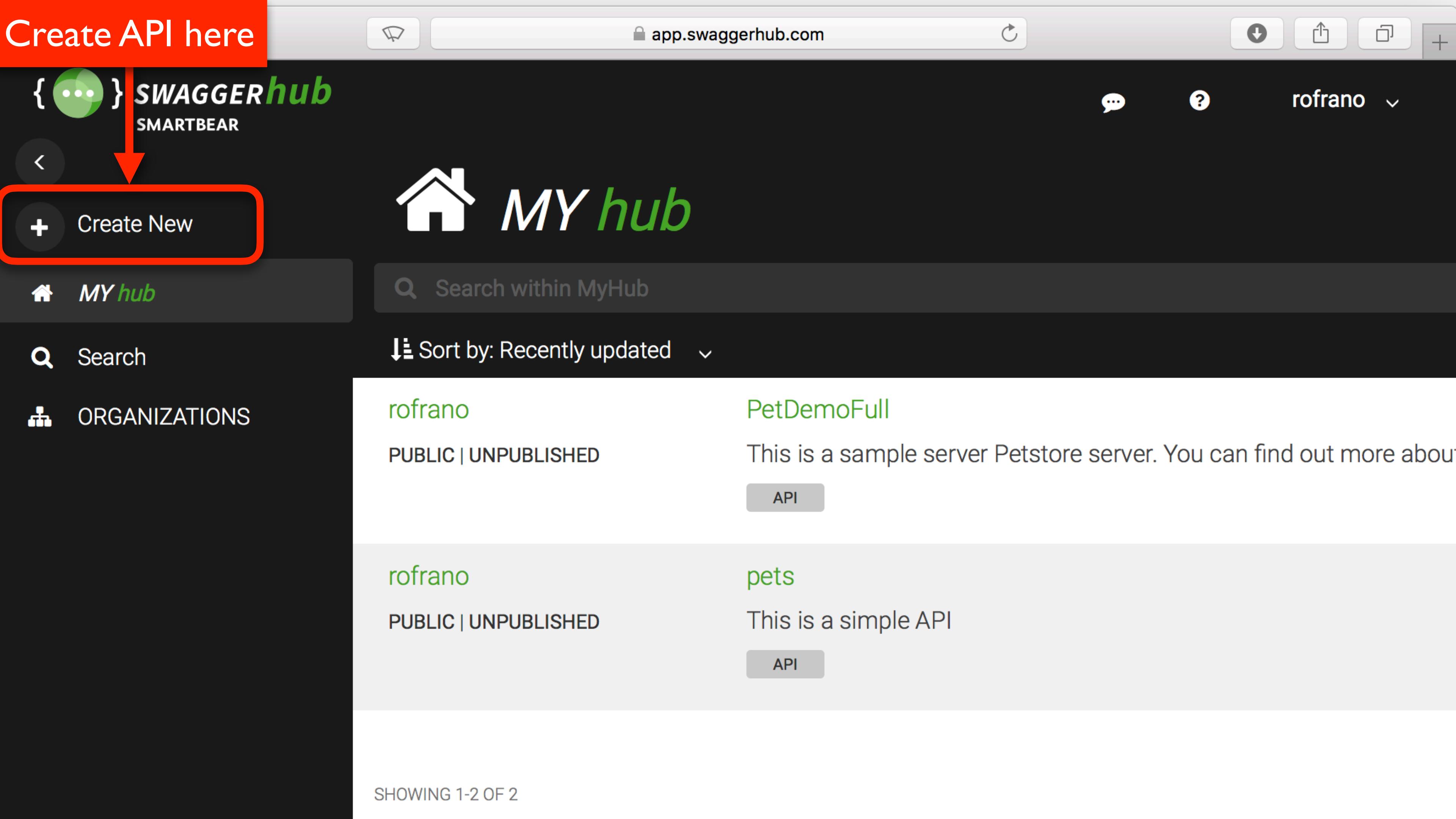
- SwaggerHub is an online platform where you can design your APIs and collaborate on them using Swagger – be it public APIs, internal private APIs or microservices
- The core principle behind Swagger and SwaggerHub is **Design First, Code Later**
 - That is, you start by laying out your API, its resources, operations and data models, and once the design is complete you implement the business logic
- Your API definitions are saved in the SwaggerHub cloud and can be synchronized with external systems like GitHub or Amazon API Gateway

swagger.io

The screenshot shows the homepage of swagger.io. The page features a navigation bar with links for "Why Swagger", "Tools", "Resources", "Sign In", and "Try Free". The main title "Swagger" is prominently displayed in green. Below it, the tagline "The Best APIs are Built with Swagger Tools" is shown. Two green buttons, "Try SwaggerHub" and "Explore Swagger Tools", are visible. The page is divided into five sections, each represented by a white circle containing an icon:

- Design**: Shows a notepad with a pen.
- Build**: Shows a card with a plus sign and a speech bubble.
- Document**: Shows a stack of three documents.
- Test**: Shows a card with a checkmark.
- Standardize**: Shows a checklist with three items: two checked and one unchecked.

Create a New API

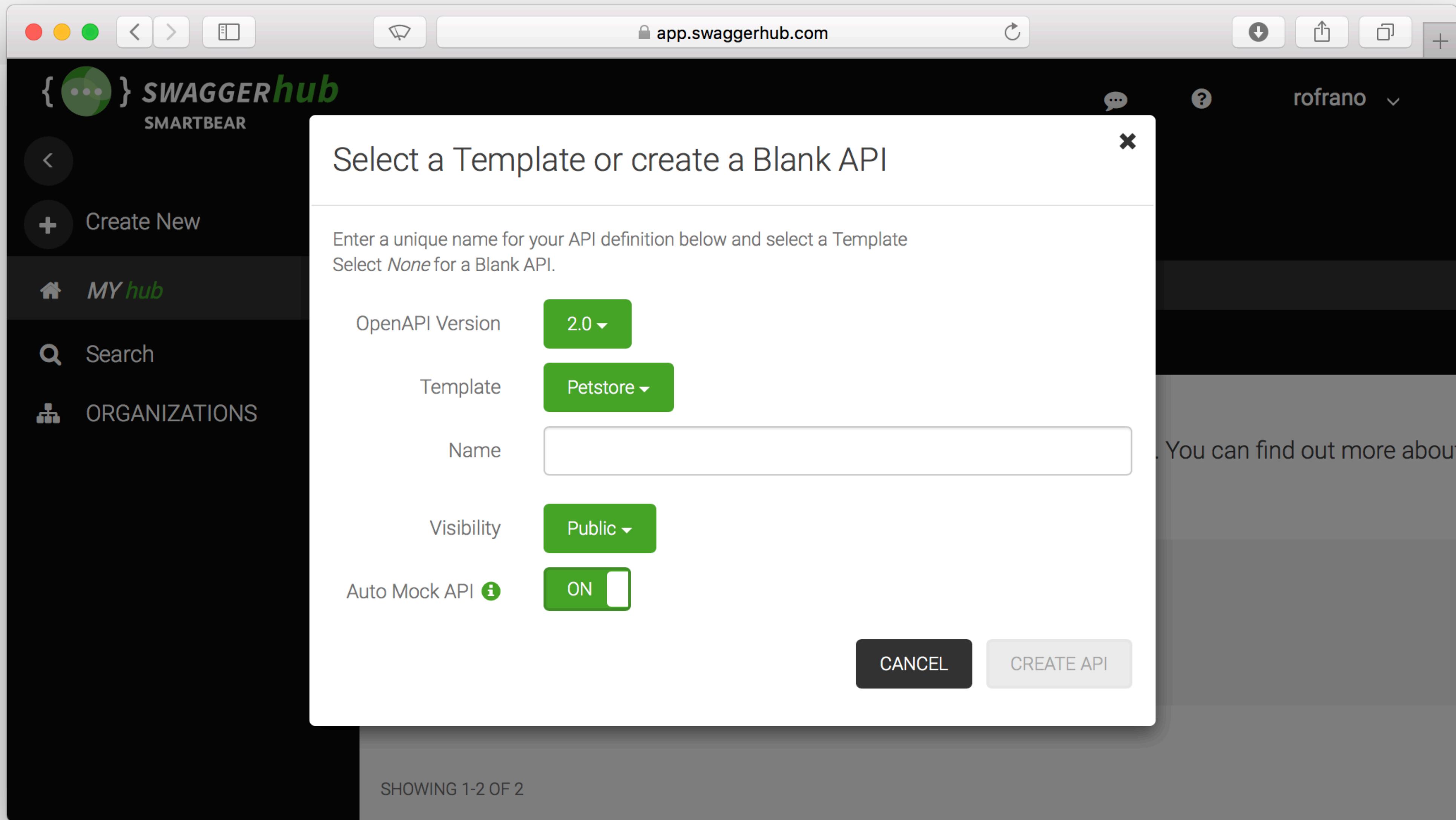


The screenshot shows the Swaggerhub web interface. At the top left, a red box highlights the "Create API here" button. A red arrow points from this button down to the "Create New" button on the left sidebar. The main content area displays two API entries:

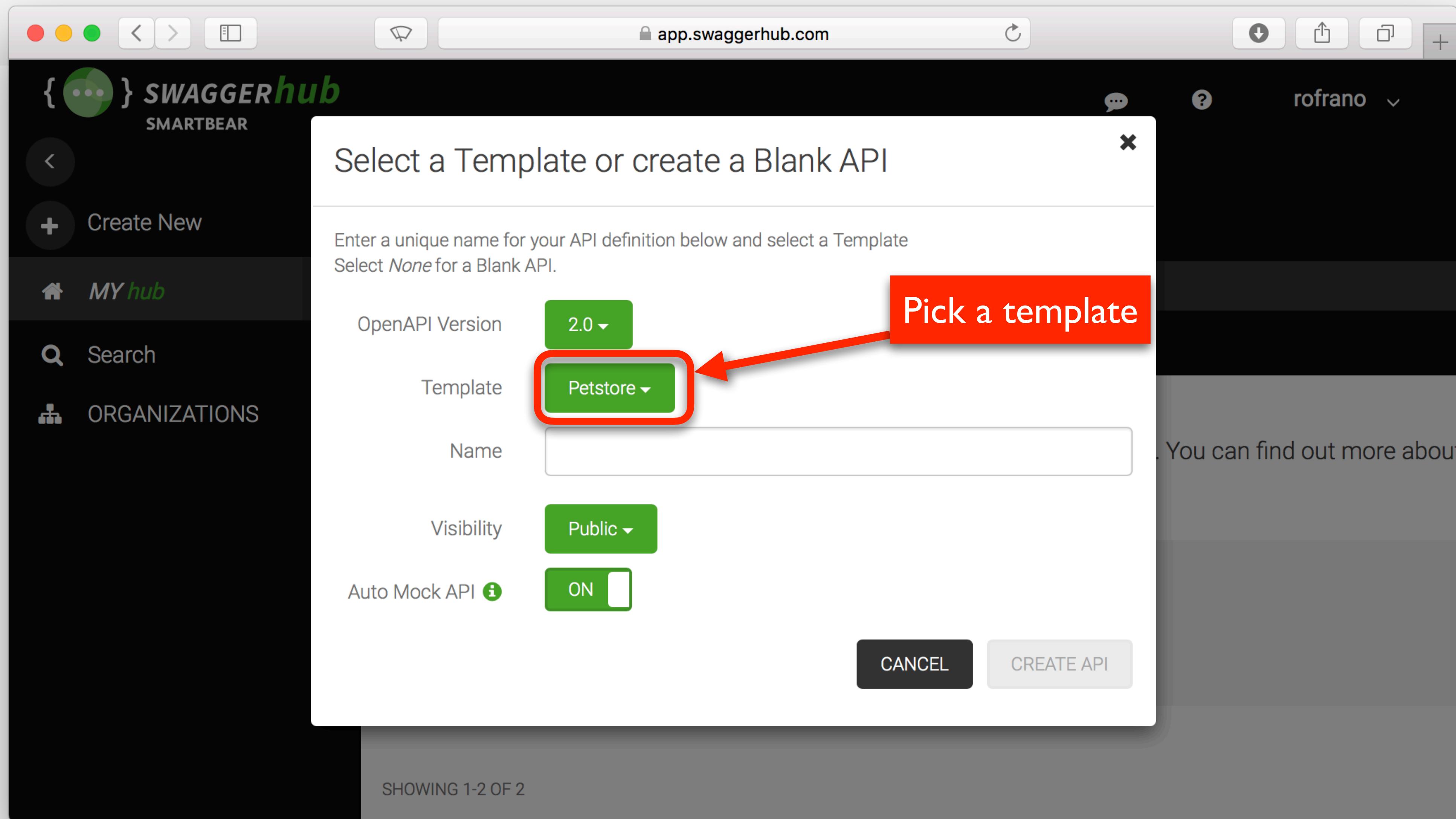
Owner	API Name	Description	Action
rofrano	PetDemoFull	This is a sample server Petstore server. You can find out more about Petstore below.	API
rofrano	pets	This is a simple API	API

At the bottom of the main content area, it says "SHOWING 1-2 OF 2".

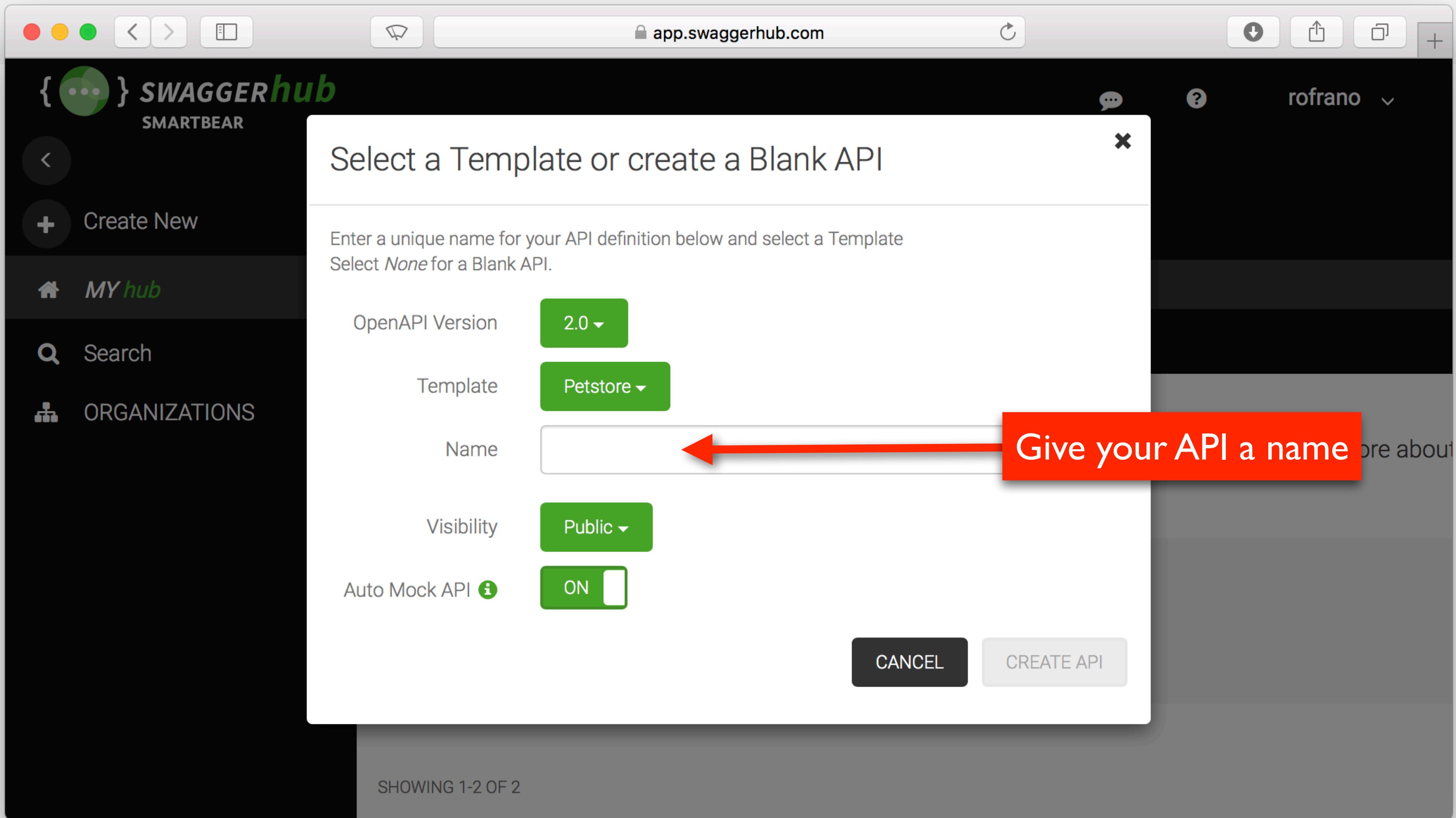
Give it a Name



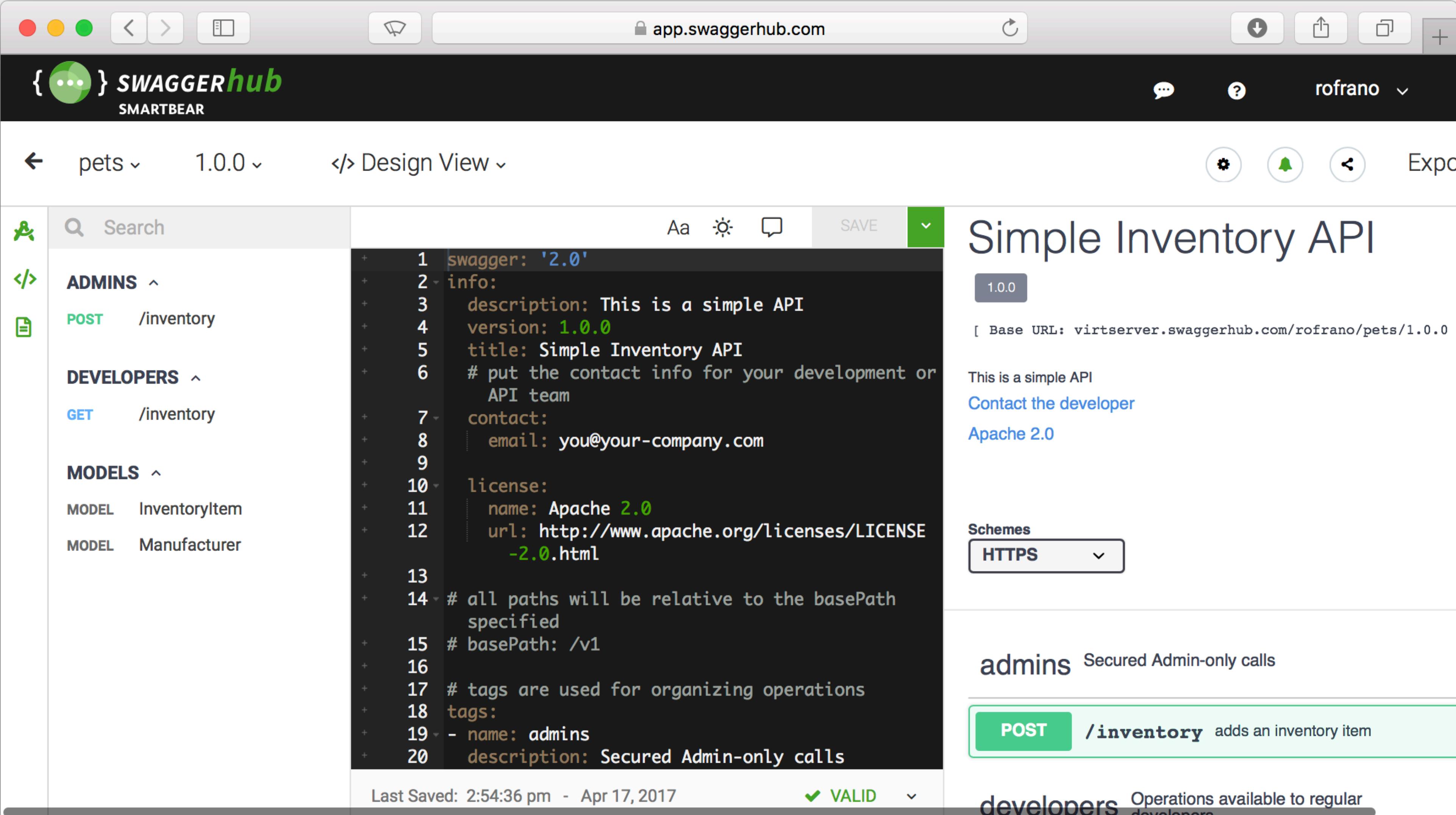
Give it a Name



Give it a Name



Sample Editor Session



The screenshot shows the Swaggerhub editor interface with the following details:

- Header:** app.swaggerhub.com, rofrano
- Project:** pets (version 1.0.0)
- Design View:** Selected
- API Definition (Swagger JSON):**

```

1  swagger: '2.0'
2  info:
3    description: This is a simple API
4    version: 1.0.0
5    title: Simple Inventory API
6    # put the contact info for your development or
      API team
7    contact:
8      email: you@your-company.com
9
10   license:
11     name: Apache 2.0
12     url: http://www.apache.org/licenses/LICENSE
           -2.0.html
13
14   # all paths will be relative to the basePath
      specified
15   # basePath: /v1
16
17   # tags are used for organizing operations
18   tags:
19     - name: admins
20       description: Secured Admin-only calls
  
```
- Generated Documentation:**

Simple Inventory API

1.0.0
[Base URL: virtserver.swaggerhub.com/rofrano/pets/1.0.0]

This is a simple API
[Contact the developer](#)
Apache 2.0

Schemes: HTTPS

admins Secured Admin-only calls

POST /inventory adds an inventory item

developers Operations available to regular developers
- Bottom Status:** Last Saved: 2:54:36 pm - Apr 17, 2017, VALID



Hands-On

“live session”

Some Assembly Required

- Tools you will need to complete this lab:
 - GitHub Enterprise Account (github.ibm.com)
 - Git Client
 - Text Editor (...i like atom.io)
 - Vagrant and VirtualBox





RestPLUS Swagger

- The code for this lab can be found here:

<https://github.com/nyu-devops/lab-flask-restplus-swagger>

```
$ git clone https://github.com/nyu-devops/lab-flask-restplus-swagger.git  
  
$ cd lab-flask-restplus-swagger  
  
$ vagrant up  
  
$ vagrant ssh
```

Live Demo



Source for the Flasgger Lab

- An alternate approach tusing Flasgger can be found here:

<https://github.com/nyu-devops/lab-flask-swagger.git>

```
$ git clone https://github.com/nyu-devops/lab-flask-swagger.git  
$ cd lab-flask-swagger  
$ vagrant up && vagrant ssh
```

Summary

- You should have a good understand Swagger
- Know how to use SwaggerHub to create API's
- Know how to automatically generate Swagger docs from your Python code using Flask-RESTPlus



Additional reading



- Getting Started With SwaggerHub
 - <https://app.swaggerhub.com/help/tutorials/getting-started>
- Flask-RESTPlus
 - <https://flask-restplus.readthedocs.io/en/stable/>