NYU COURANT INSTITUTE OF MATHEMATICAL SCIENCES

# Introduction to Swagger & Open API

swagger

OPEN API INITIATIVE

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

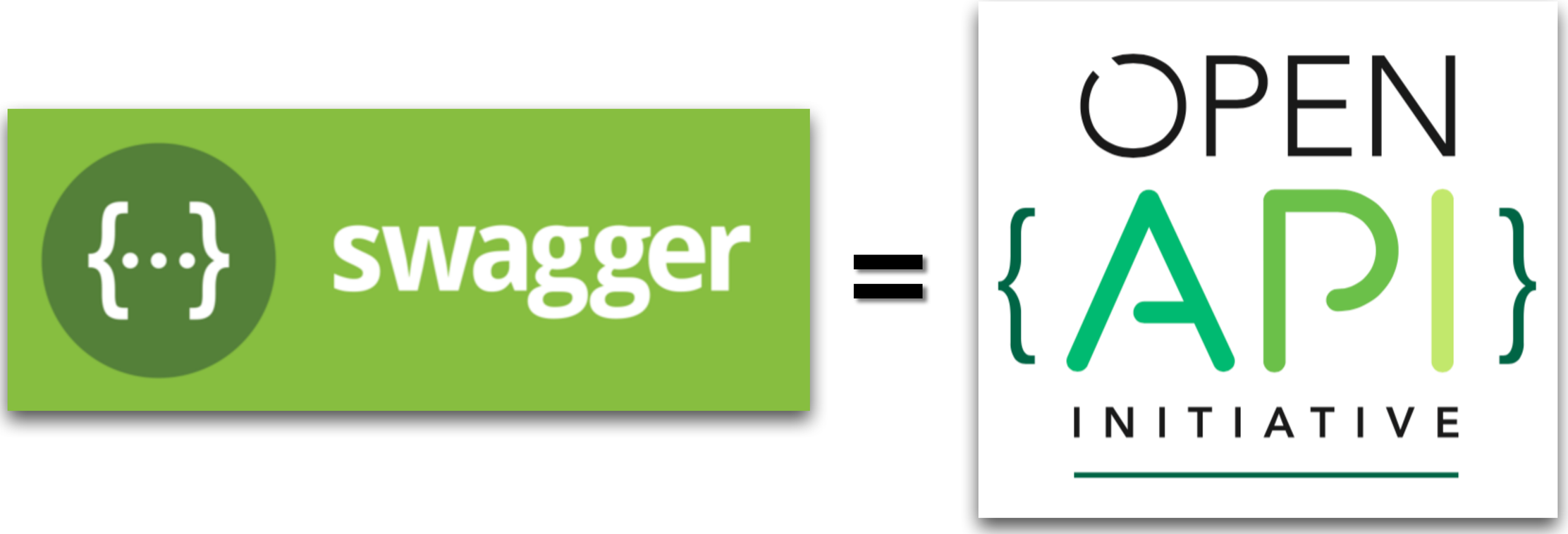
# Why is Documentation Important?

- 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?


=

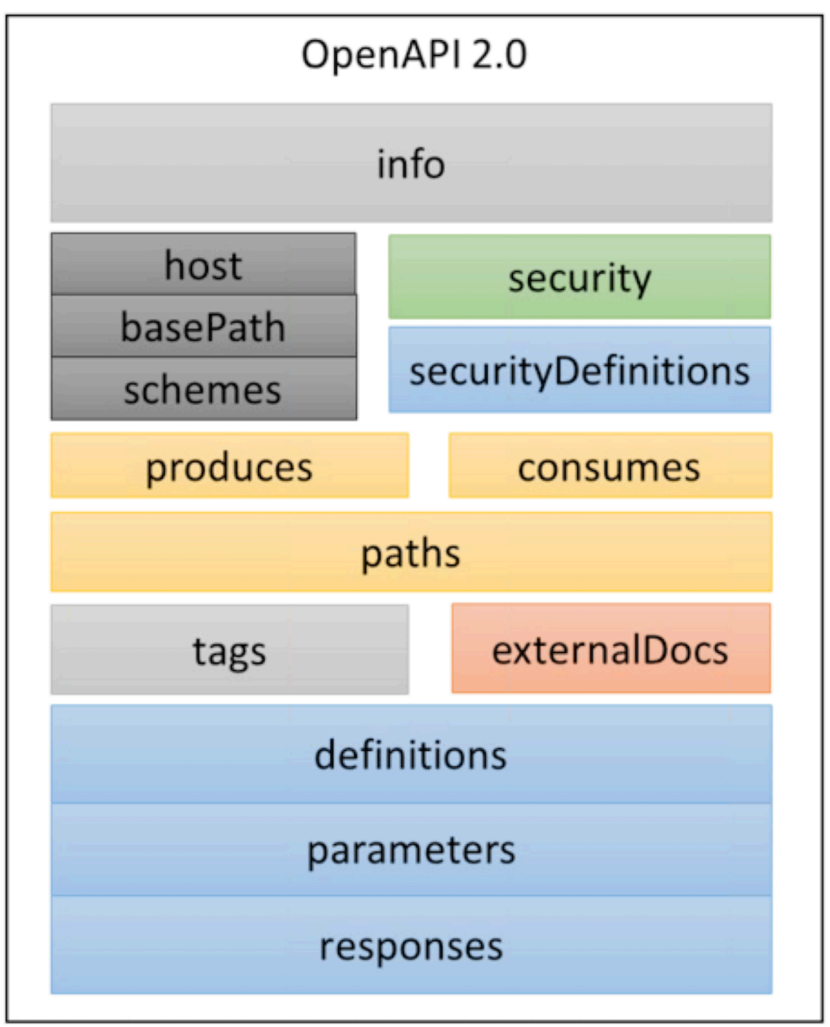
Specification

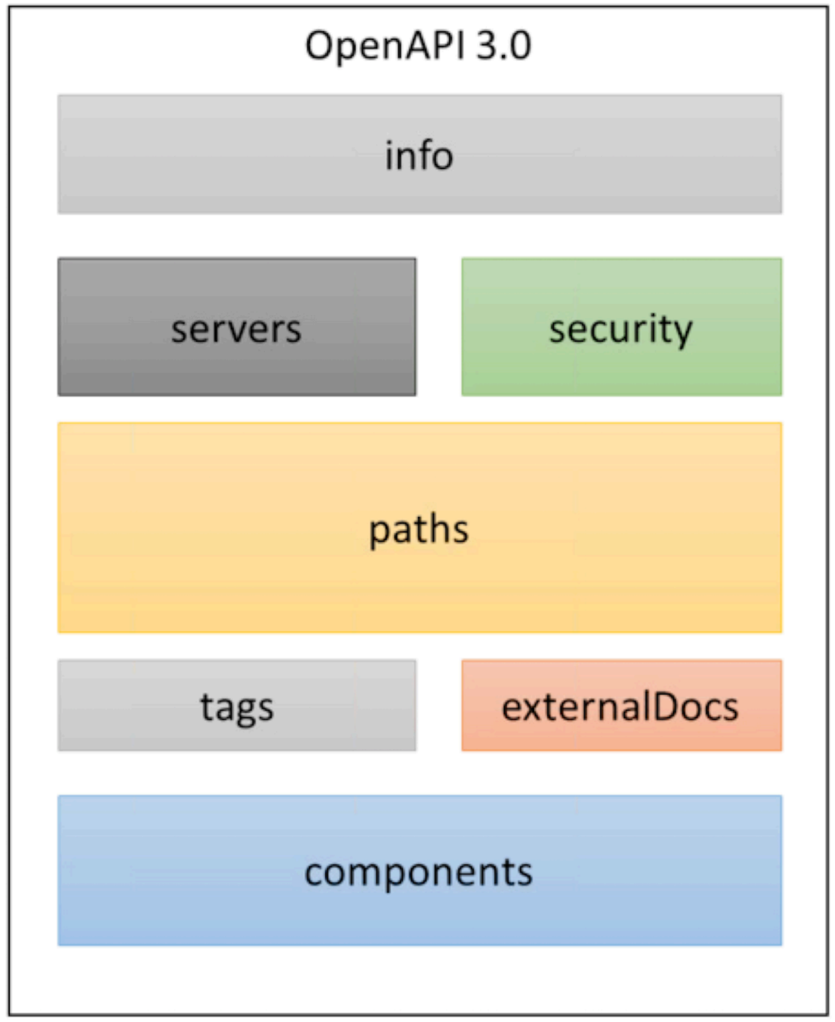
swagger

=

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:

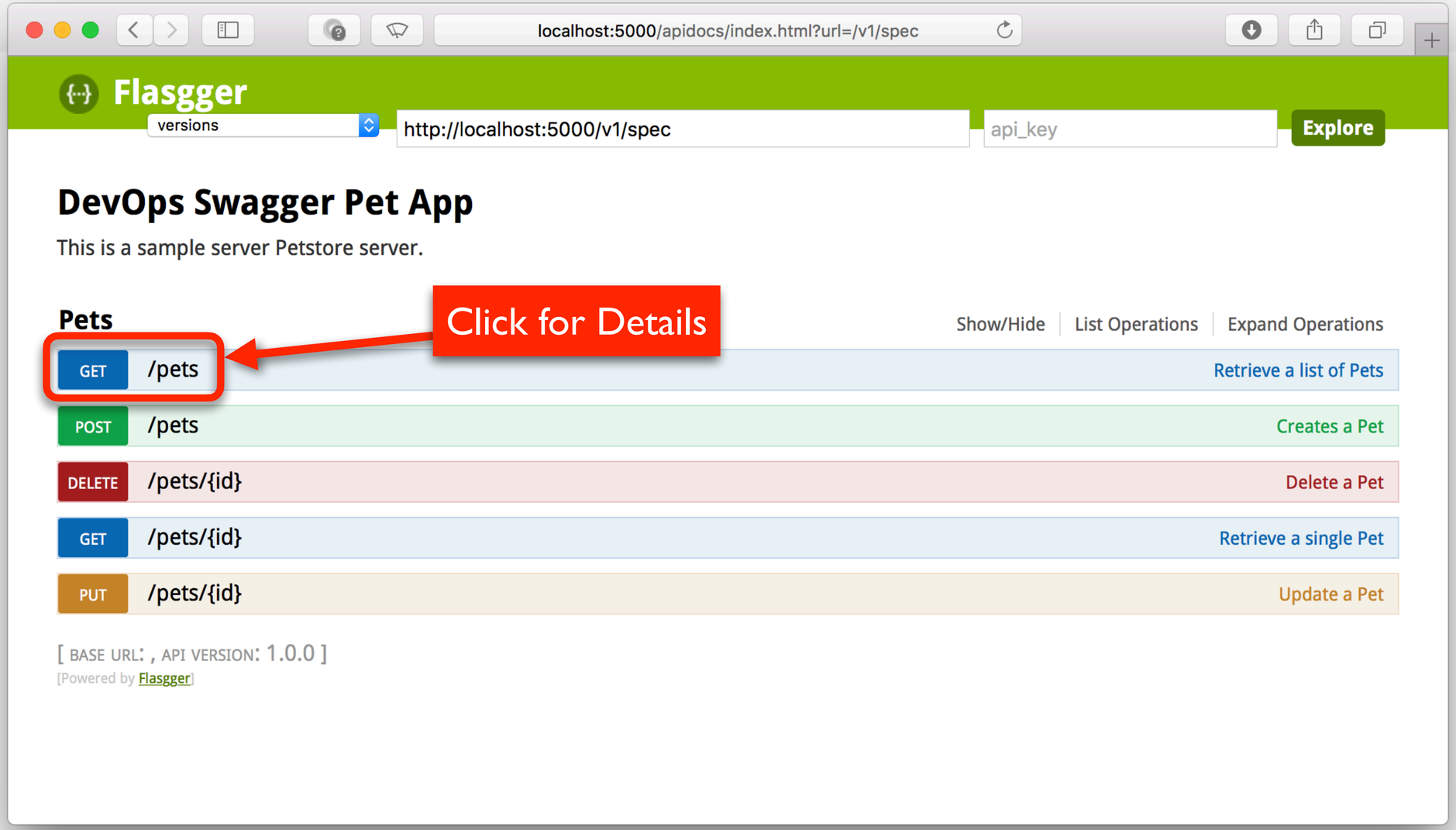
```
import connexion

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



# 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:**

Maps the operationID to your code

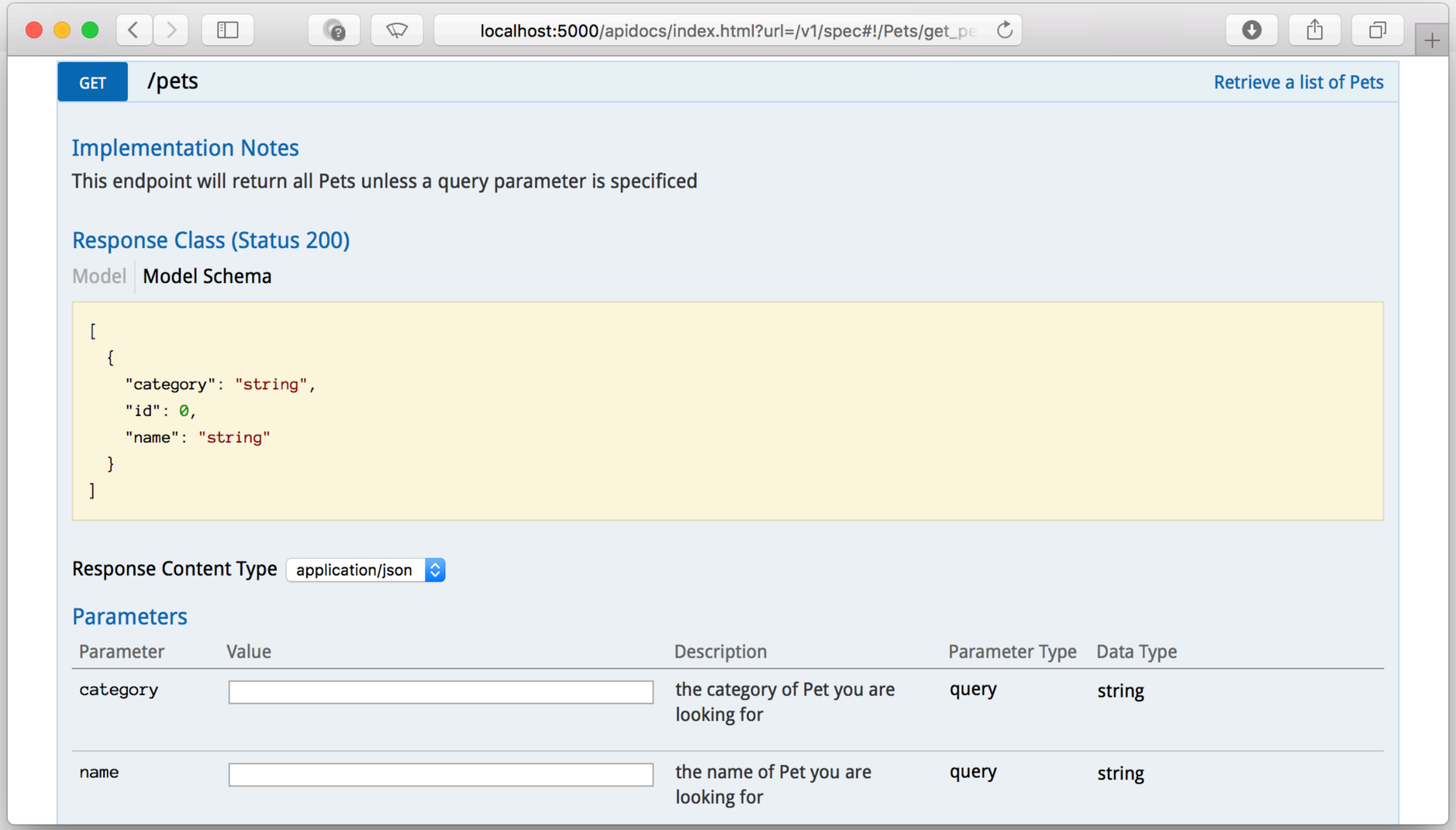
```
paths:
  /pets:
    get:
      operationId: api.get_pets
      parameters:
        - name: category
          description: Pet category
          in: query
          type: string
          required: true
```

**ew function:**

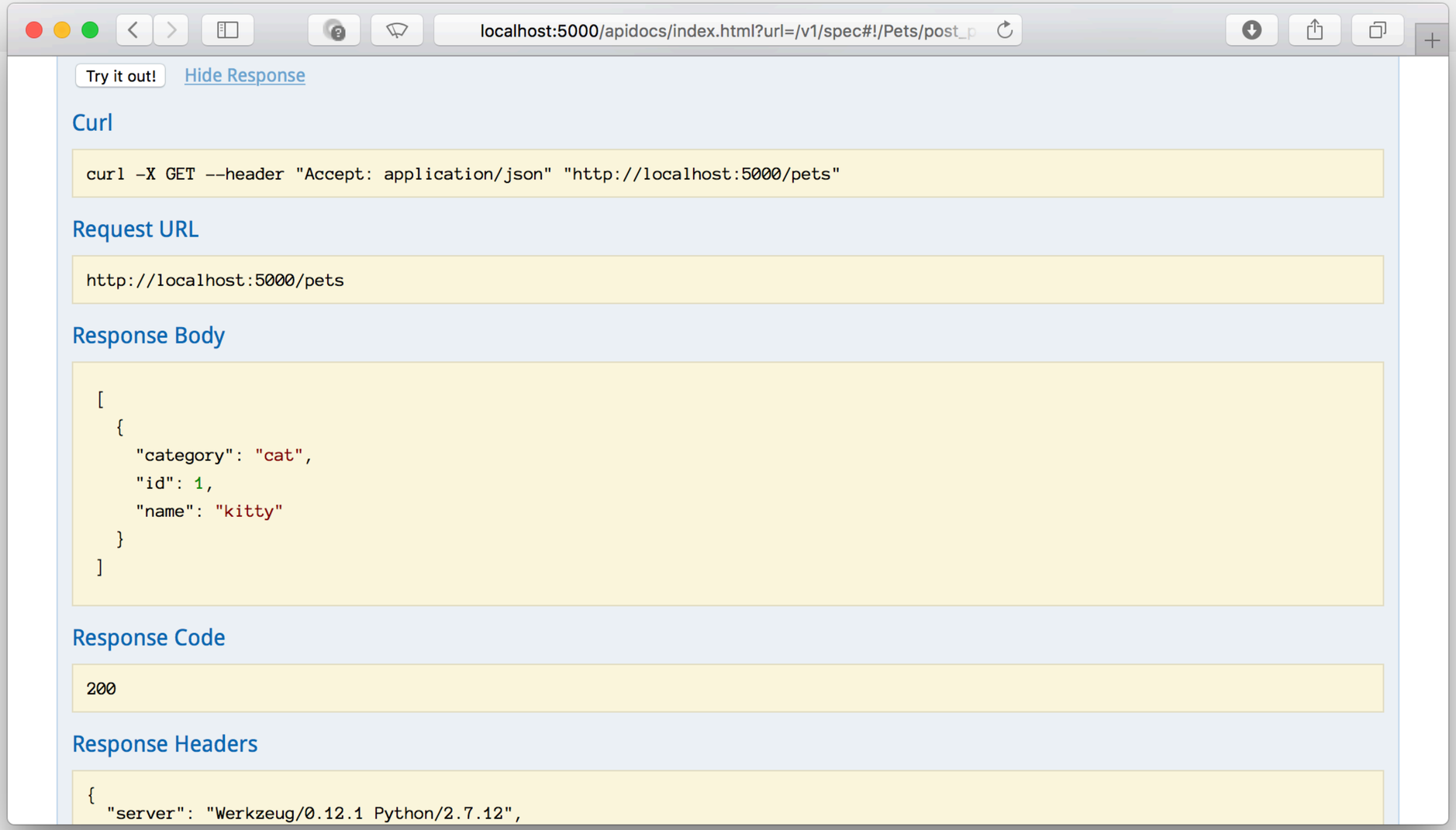
```
# api.py file

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

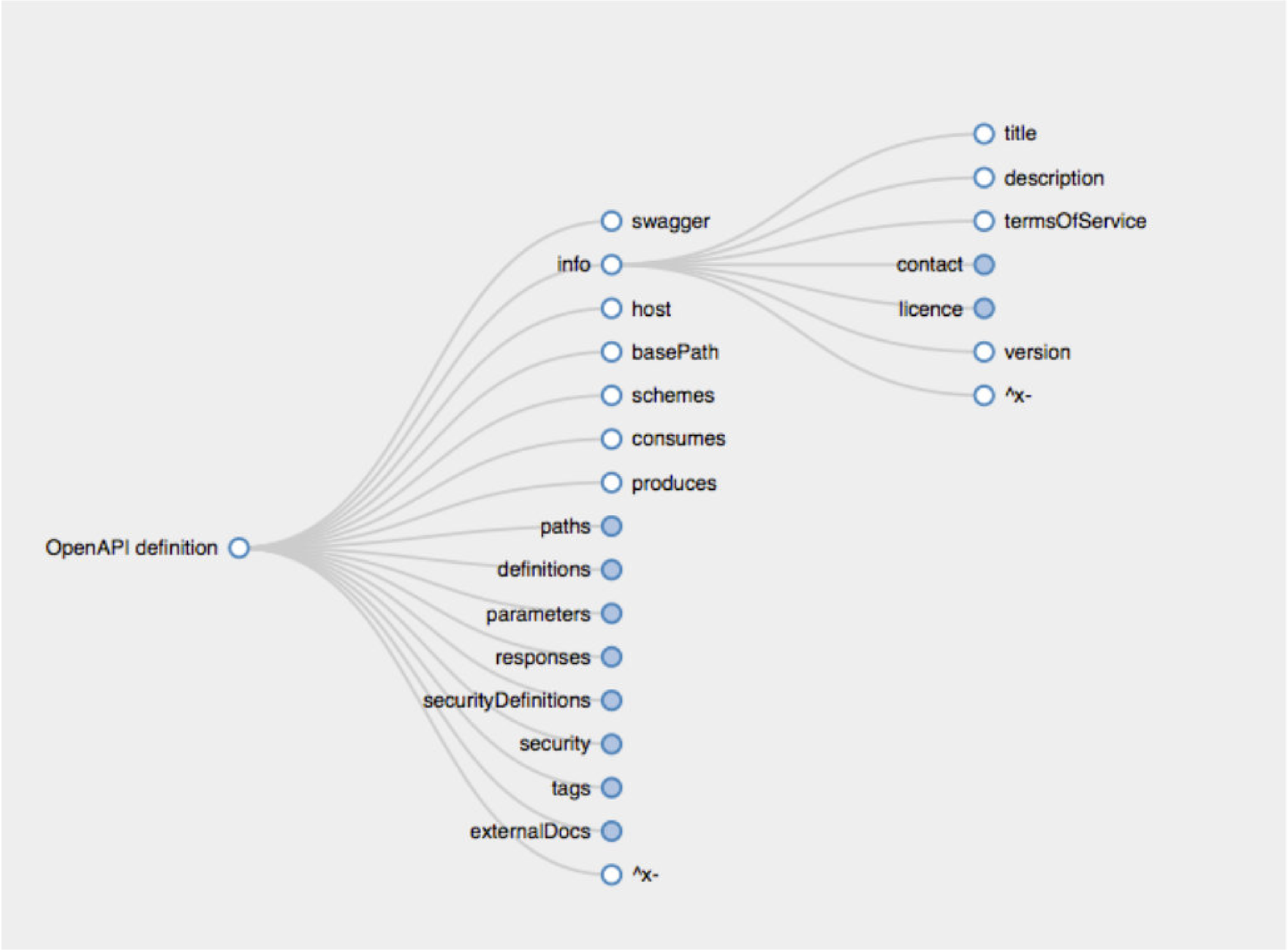
# Details on Each Call



# Try It Out


localhost:5000/apidocs/index.html?url=/v1/spec#!/Pets/post_p
Try it out!
Hide Response
Curl
curl -X GET --header "Accept: application/json" "http://localhost:5000/pets"
Request URL
http://localhost:5000/pets
Response Body
[
{
"category": "cat",
"id": 1,
"name": "kitty"
}
]
Response Code
200
Response Headers
{
"server": "Werkzeug/0.12.1 Python/2.7.12",

# Open-API Definition



# Basic Information

```yaml
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

```yaml
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

```yaml
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}

```yaml
/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

```yaml
  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}

```yaml
  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

```yaml
  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

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

```yaml
      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
```

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

```yaml
      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

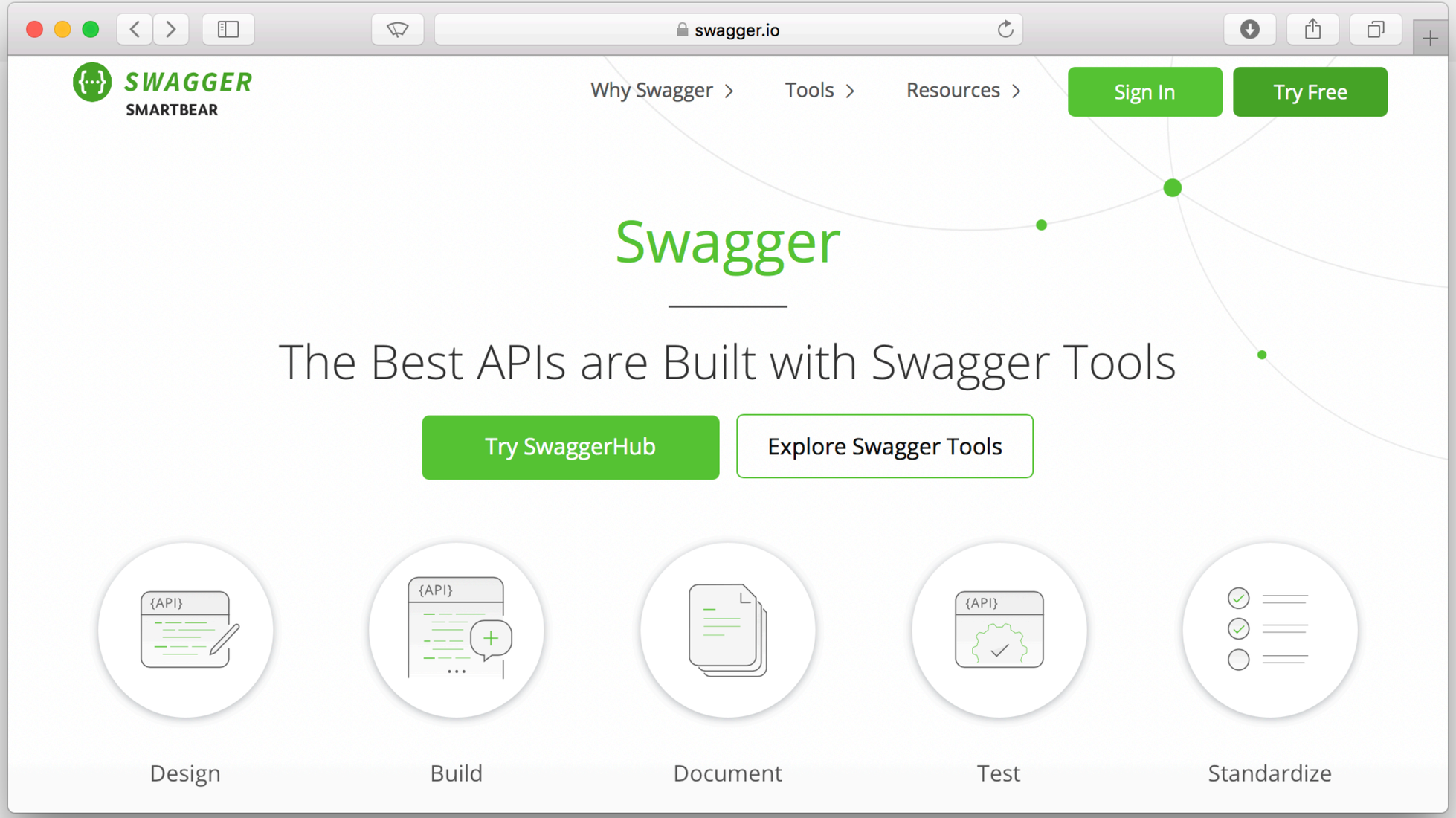
```
securityDefinitions:
  petstore_auth:
    type: oauth2
    authorizationUrl: http://petstore.swagger.io/oauth/dialog
    flow: implicit
    scopes:
      write:pets: modify pets in your account
      read:pets: read your pets
  api_key:
    type: apiKey
    name: api_key
    in: header
```


SWAGGERhub

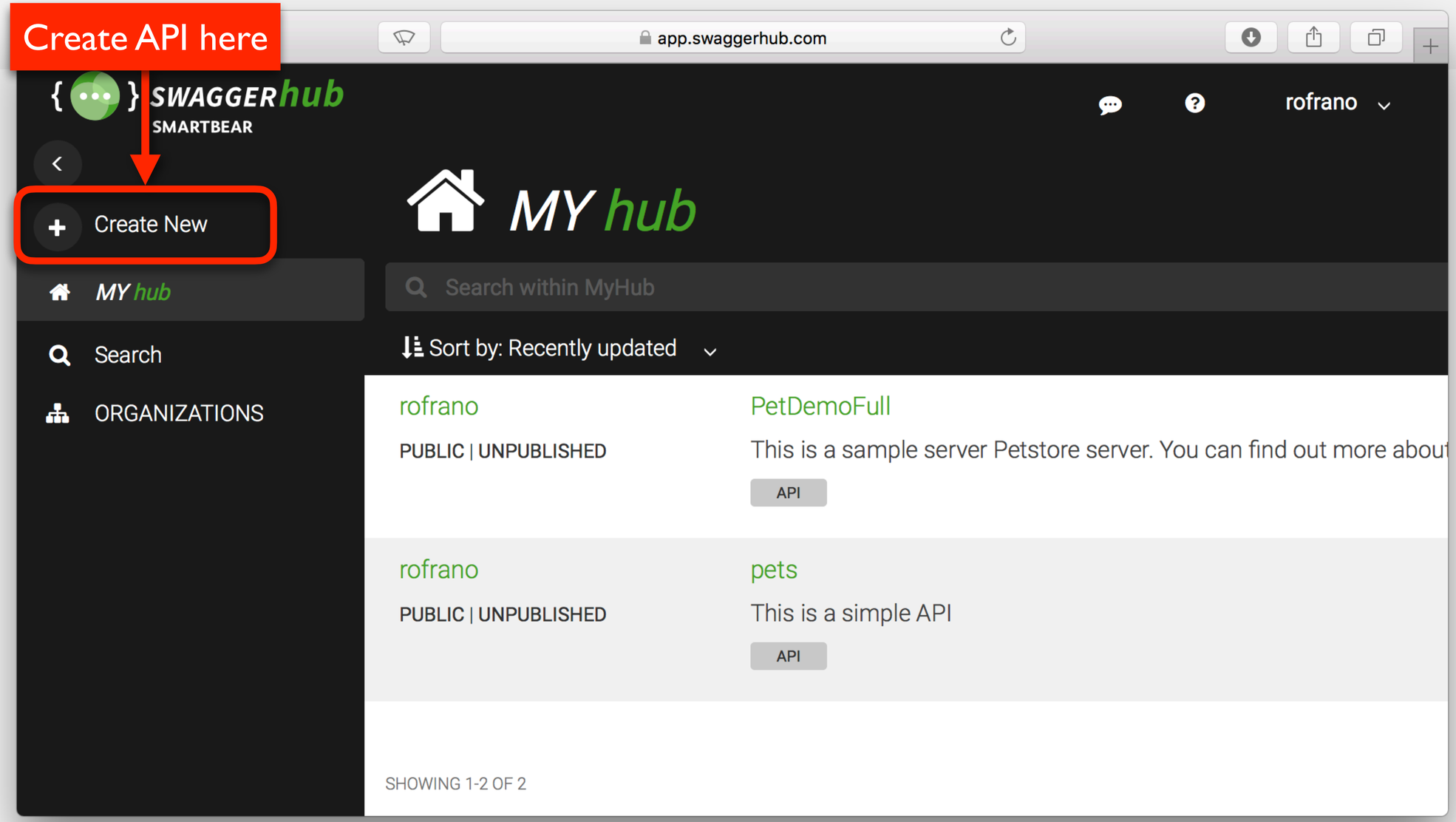
# What is SwaggerHub?

- 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

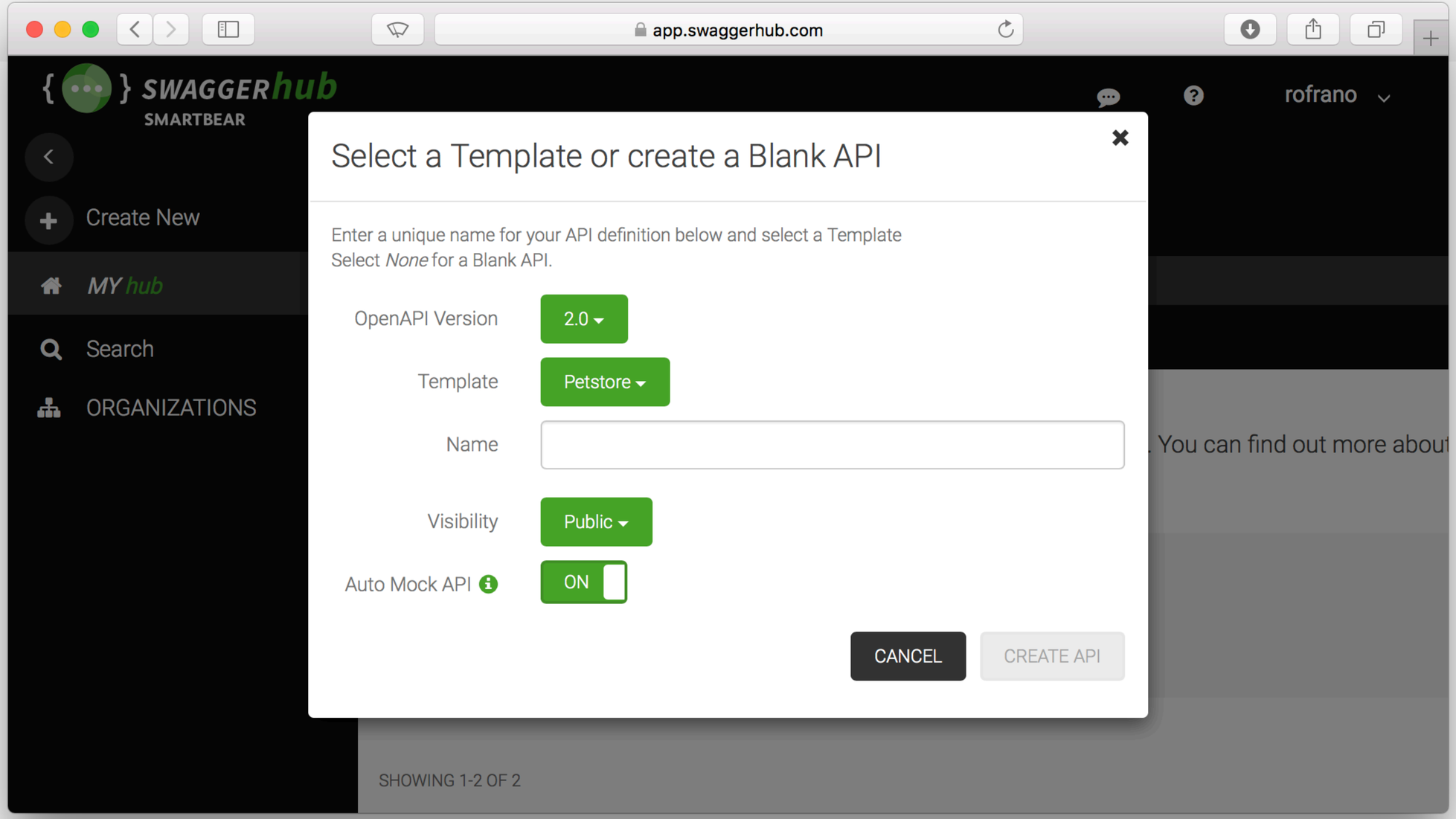
# swagger.io


swagger.io
SWAGGER
SMARTBEAR
Why Swagger
Tools
Resources
Sign In
Try Free
Swagger
The Best APIs are Built with Swagger Tools
Try SwaggerHub
Explore Swagger Tools
{API}
{API}
{API}
Design
Build
Document
Test
Standardize

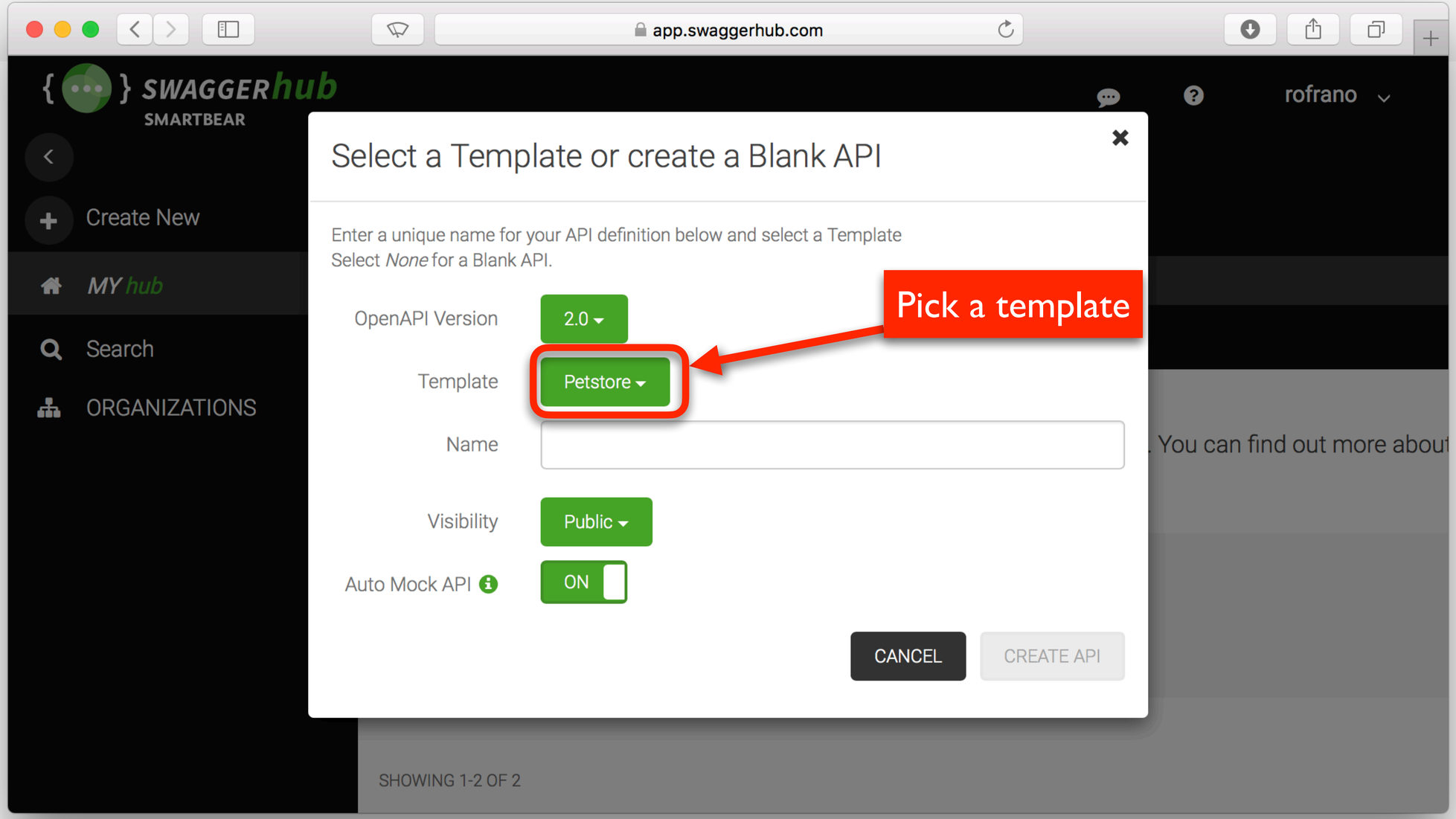
# Create a New API



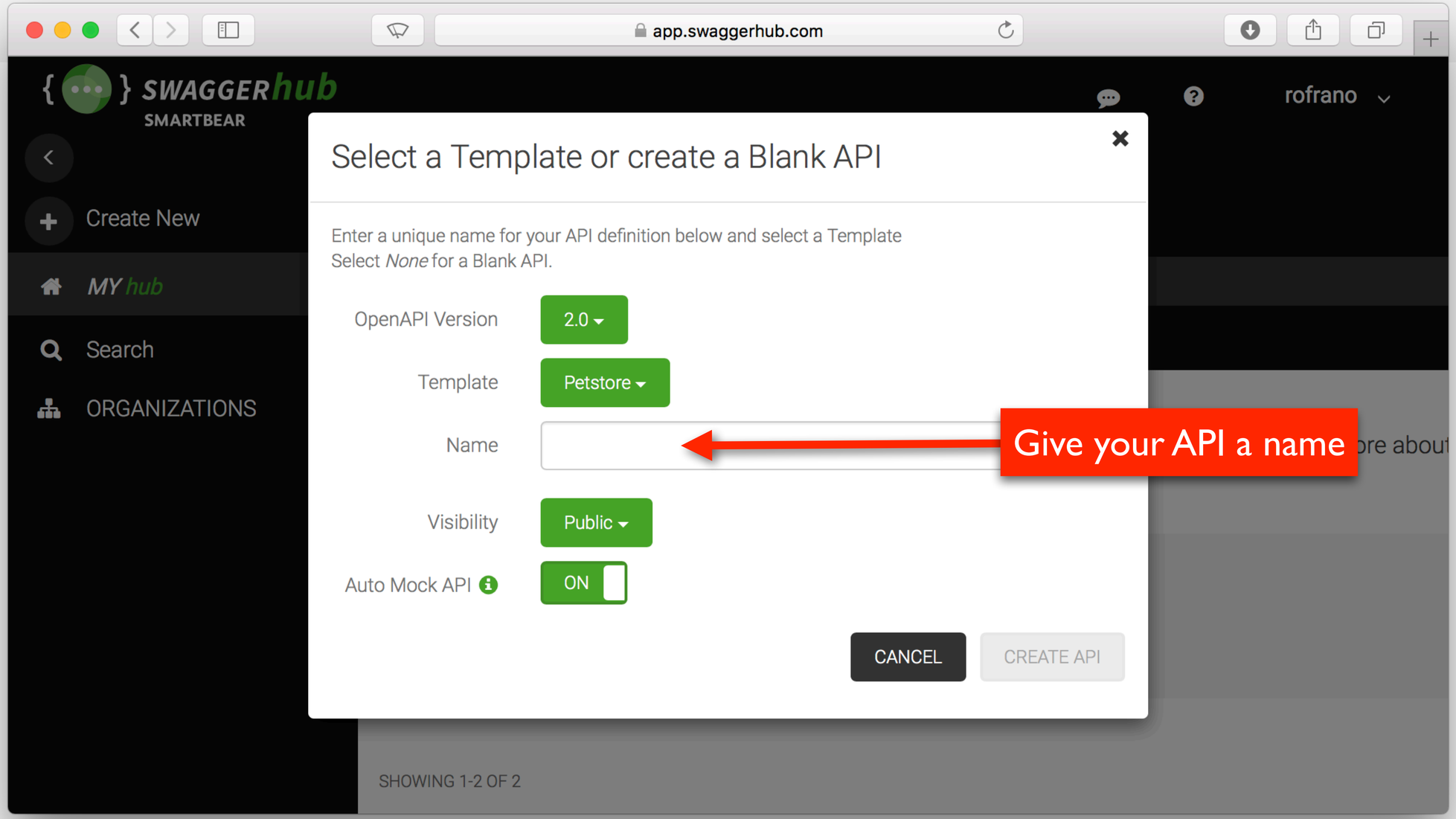
# Give it a Name



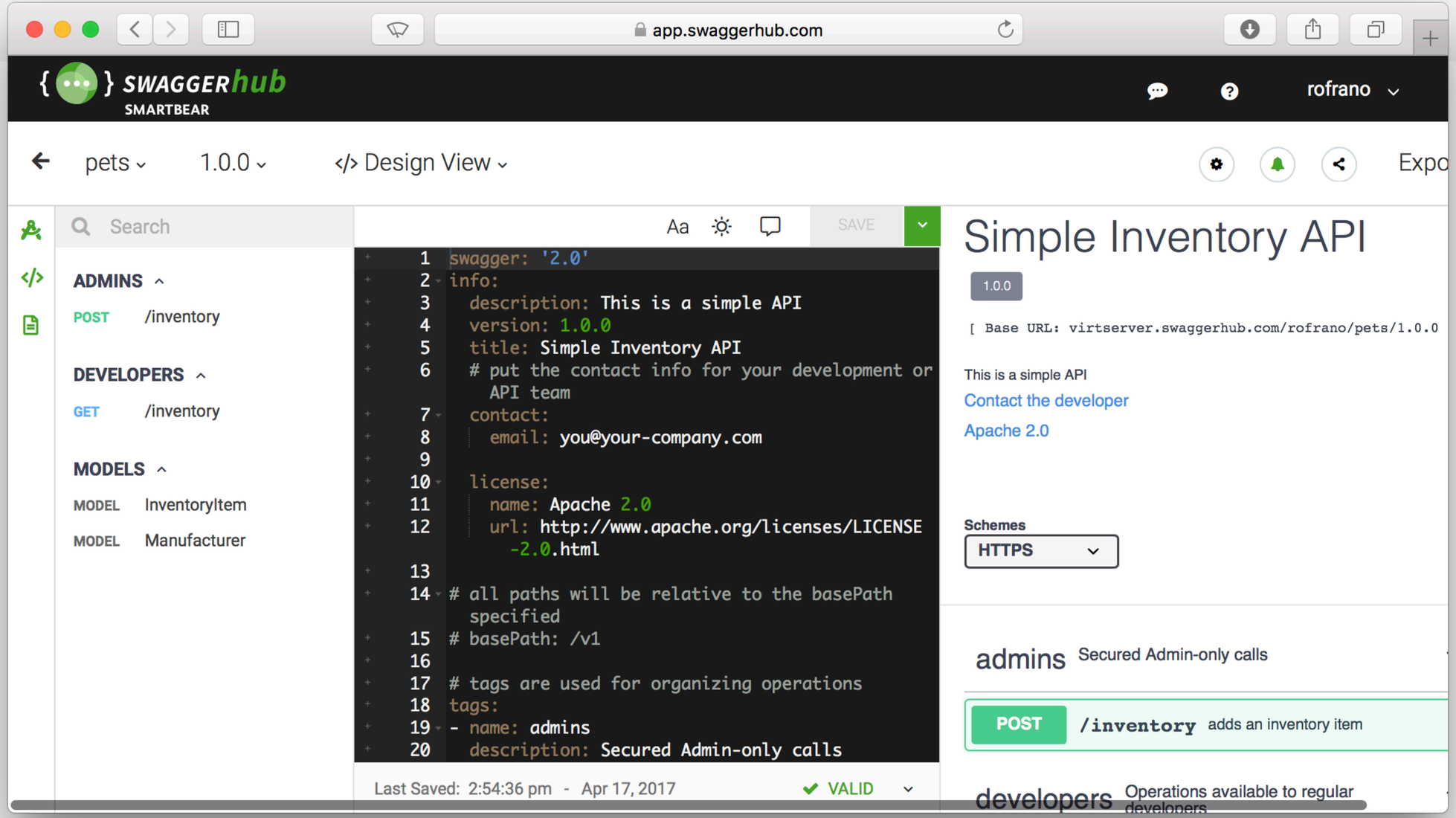
# Give it a Name



# Give it a Name



# Sample Editor Session



# 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/