สยามจำนาญกิจ
SIAM CHAMNAN KIT

Automated Testing

API development and testing

https://github.com/up1/workshop-api-first

Automated Testing for REST APIs

Delivery process

Simple process

API specification

Development

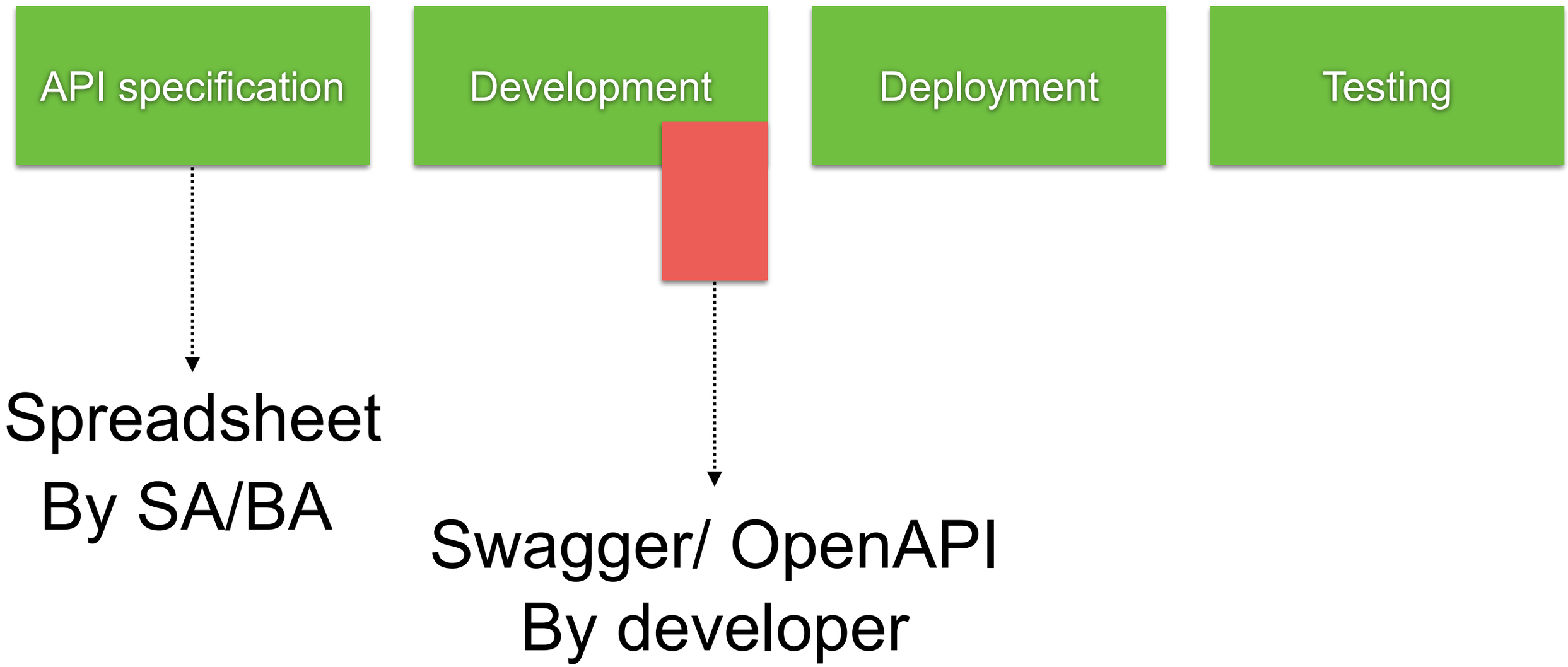
Deployment

Testing

Spreadsheet
By SA/BA

Delivery process

Need API documentation



SA/BA Developer Developer

Spec vs Code
Sync ?

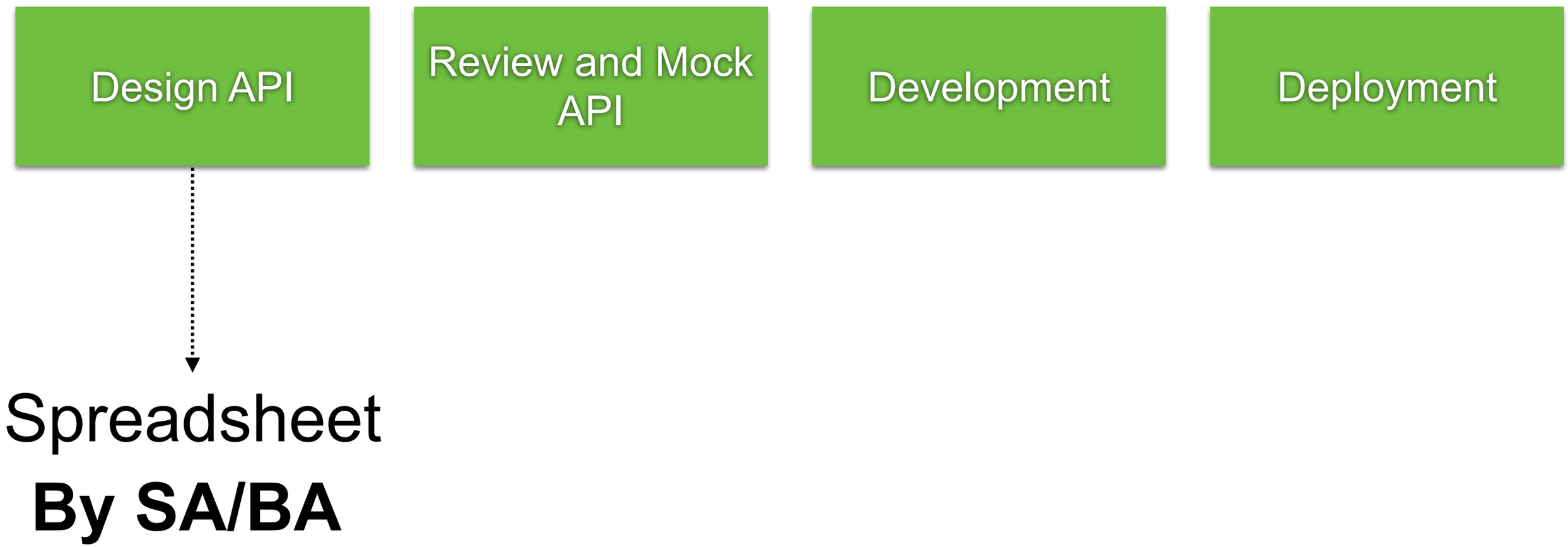
Code-first vs API-first development

Code First

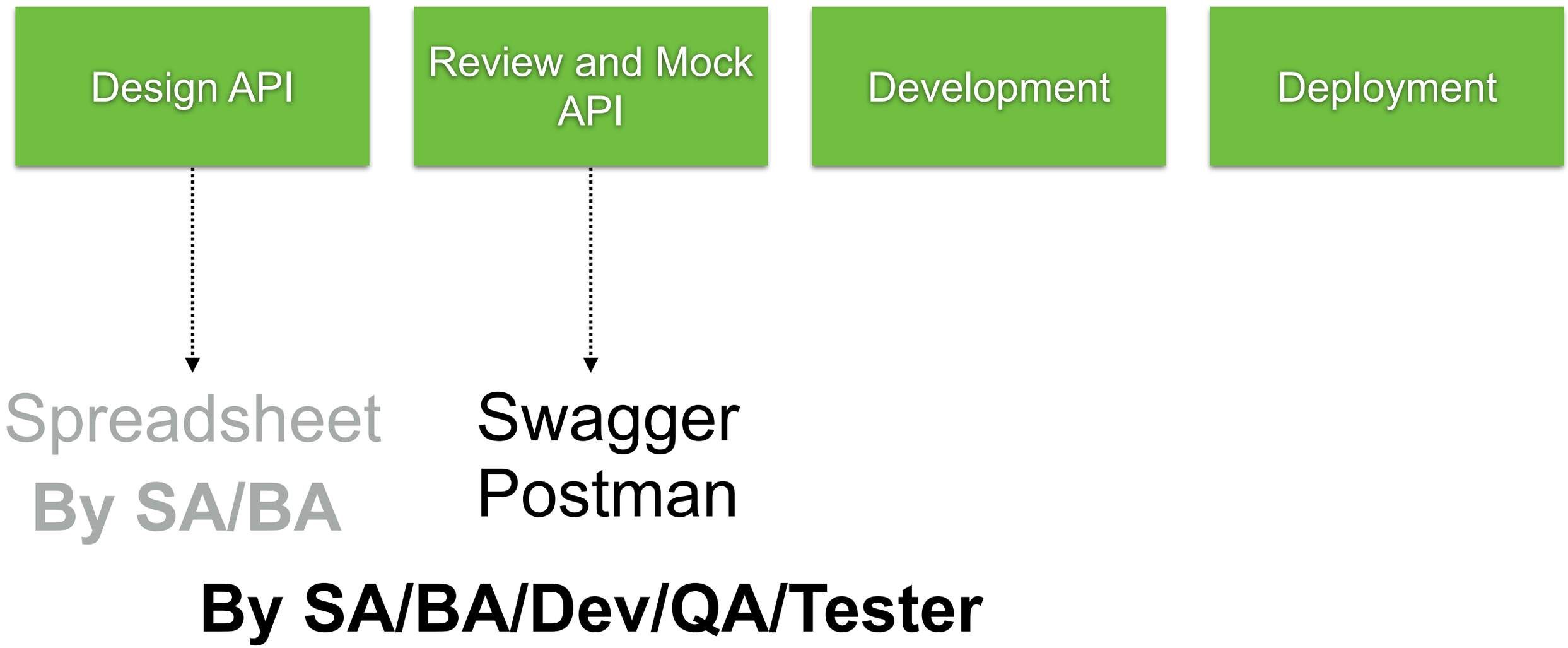
vs.

API First

API-first development



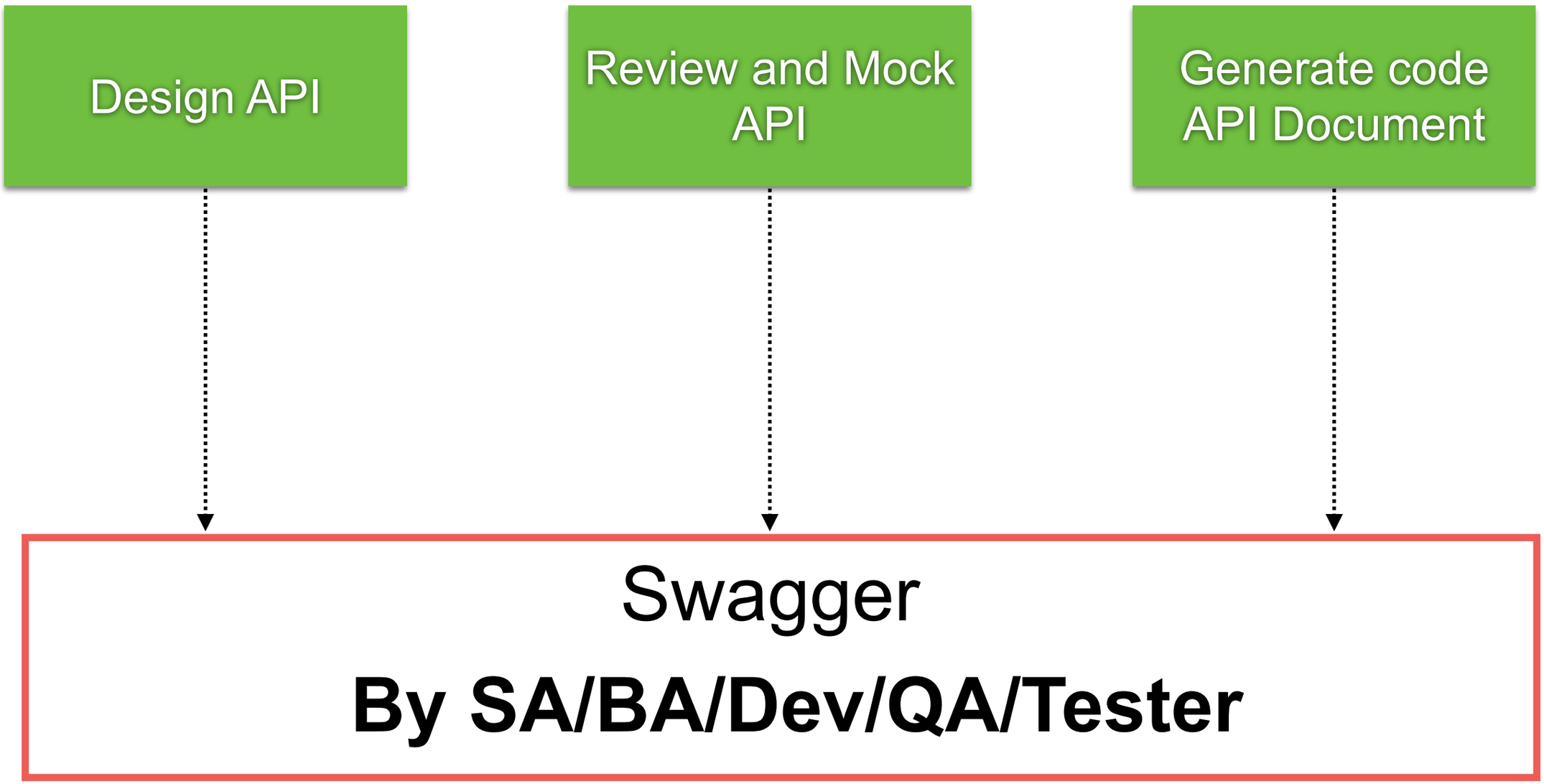
API-first development



Working with Swagger

https://swagger.io/

Process with Swagger



What id OpenAPI ?

What Is OpenAPI?

OpenAPI Specification (formerly Swagger Specification) is an API description format for REST APIs. An Op
your entire API, including:

- Available endpoints (`/users`) and operations on each endpoint (`GET /users`, `POST /users`)
- Operation parameters Input and output for each operation
- Authentication methods
- Contact information, license, terms of use and other information.

API specifications can be written in YAML or JSON. The format is easy to learn and readable to both huma
OpenAPI Specification can be found on GitHub: OpenAPI 3.0 Specification

What Is Swagger?

Swagger is a set of open-source tools built around the OpenAPI Specification that can help you design, bu
APIs. The major Swagger tools include:

https://swagger.io/docs/specification/about/

Learn OpenAPI Specification

OpenAPI Specification

Version 3.1.0

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 RFC2119 RFC8174 when, and only when, they appear in all capitals, as shown here.

This document is licensed under The Apache License, Version 2.0.

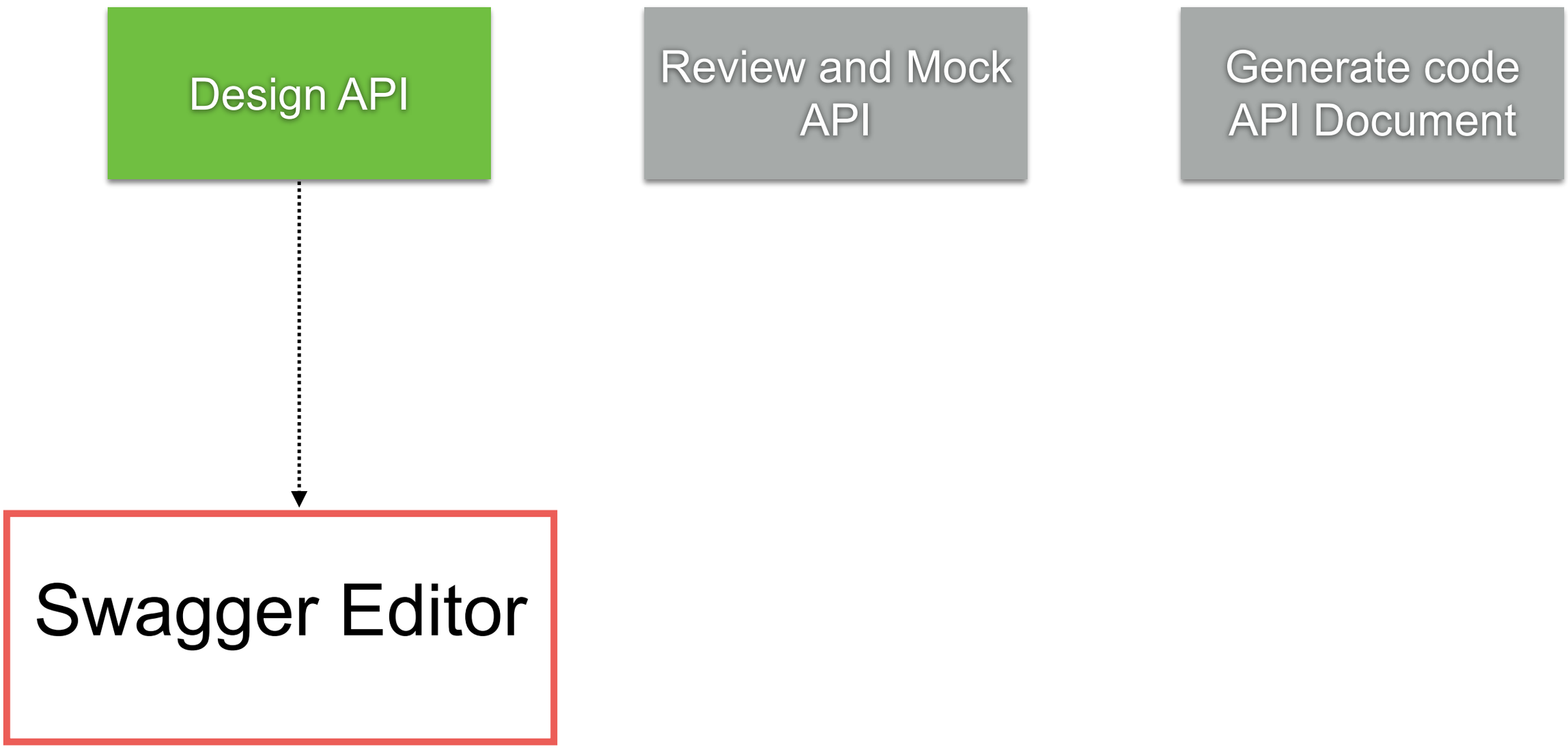
Introduction

The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to HTTP APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined, a consumer can understand and interact with the remote service with a minimal amount of implementation logic.

An OpenAPI definition can then be used by documentation generation tools to display the API, code generation tools to generate servers and clients in various programming languages, testing tools, and many other use cases.

https://swagger.io/specification/

Step 1 :: Design API



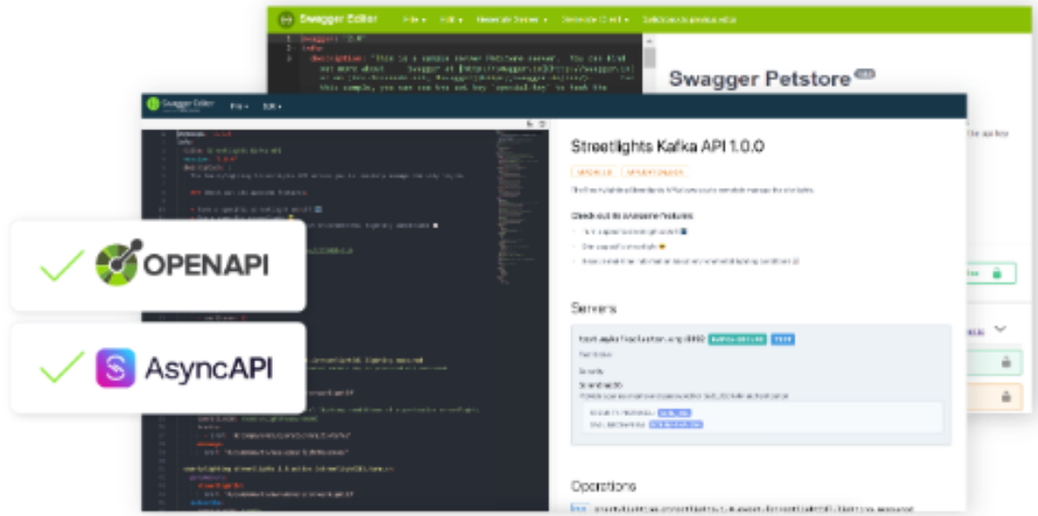
https://editor-next.swagger.io/

Swagger Editor

Design, describe and document API

Swagger Editor

Design, describe, and document your API on the first open source editor supporting multiple API specifications and serialization formats. The Swagger Editor offers an easy way to get started with the OpenAPI Specification (formerly known as Swagger) as well as the AsyncAPI specification, with support for Swagger 2.0, OpenAPI 3.*, and AsyncAPI 2.* versions.



https://swagger.io/tools/swagger-editor/

Basic Structure

Information

Server

Paths

Example

YAML or JSON

```yaml
openapi: 3.0.3
info:
  title: Demo
  version: 1.0.0
servers:
  - url: http://localhost:3000
    variables: {}
paths:
  /ping:
    get:
      responses:
        "200":
          description: OK
```

Try in Swagger Editor

Swagger Editor — File ▾ Edit ▾ Insert ▾ Generate Server ▾ Generate Client ▾ About ▾ — Try our new Editor ↗

```yaml
openapi: 3.0.3
info:
  title: Demo
  version: 1.0.0
servers:
  - url: http://localhost:3000
    variables: {}
paths:
  /ping:
    get:
      responses:
        '200':
          description: OK
```

Demo 1.0.0 OAS 3.0

Servers

http://localhost:3000

default

GET /ping

Parameters — Try it out

No parameters

Responses

Curl

```
curl -X 'GET' \
```

Add more paths

Path variable

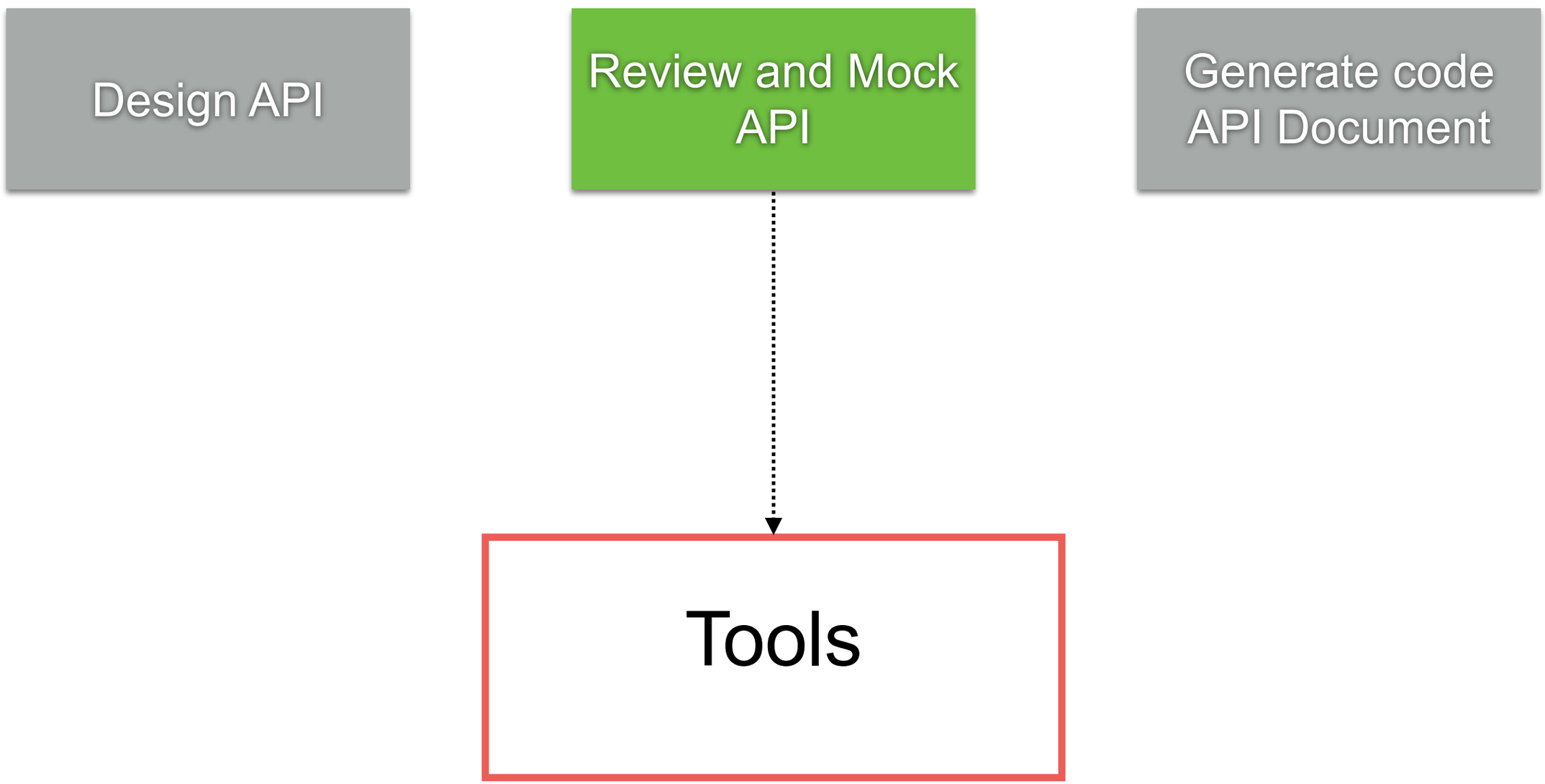
```yaml
paths:
 /users/{id}:
    get:
      parameters:
        - in: path
          name: id
          schema:
            type: integer
          required: true
          description: User ID of the user
      responses:
        "200":
          content:
            application/json:
              schema:
                type: object
                properties:
                  id:
                    type: integer
                  name:
                    type: string
                example:
                  id: 1
                  name: User 01
```

Add more paths

```yaml
paths:
 /users/{id}:
    get:
      parameters:
        - in: path
          name: id
          schema:
            type: integer
          required: true
          description: User ID of the user
      responses:
        "200":
          content:
            application/json:
              schema:
                type: object
                properties:
                  id:
                    type: integer
                  name:
                    type: string
                example:
                  id: 1
                  name: User 01
```

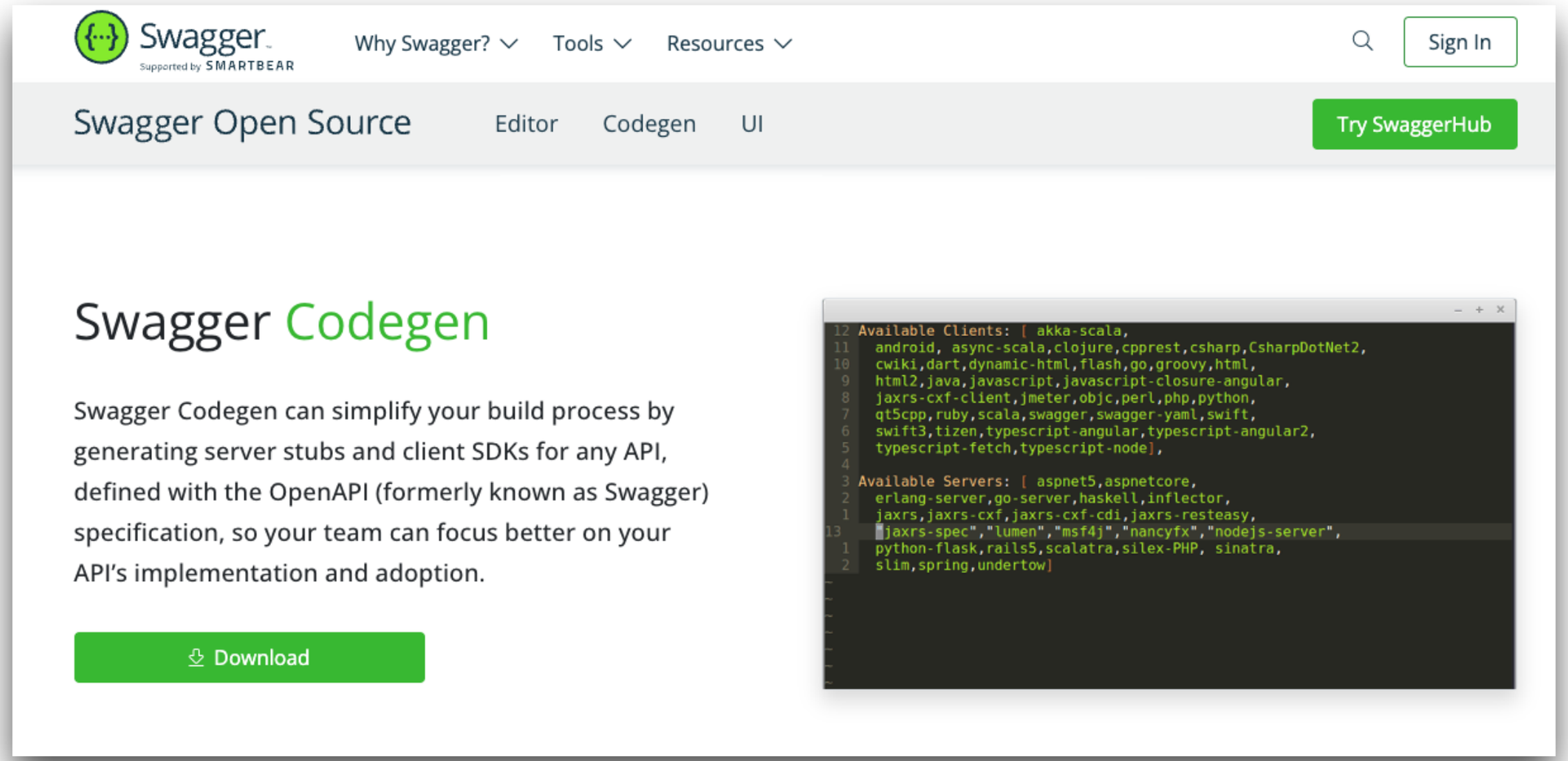
Response Schema Example

Step 2 :: Mock API



Swagger Codegen

Generate HTML, Server stub and client



https://swagger.io/tools/swagger-codegen/

Installation and how to use !!

$brew install swagger-codegen

```
$swagger-codegen generate -i api.yml -o nodejs
-l nodejs-server
```

https://github.com/swagger-api/swagger-codegen#prerequisites

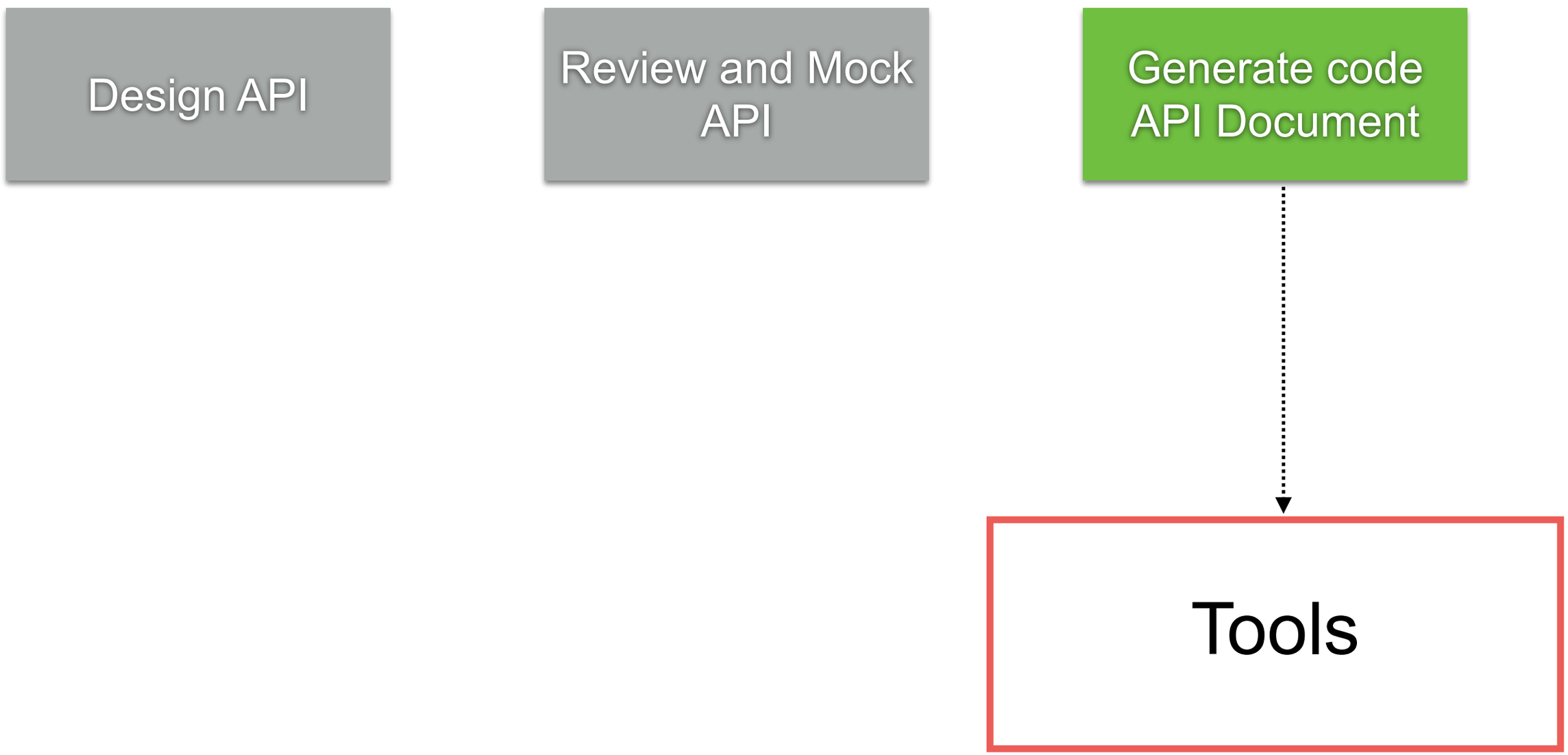
OpenAPI Mocker

$npm i -g open-api-mocker
$open-api-mocker -s api.yml -w

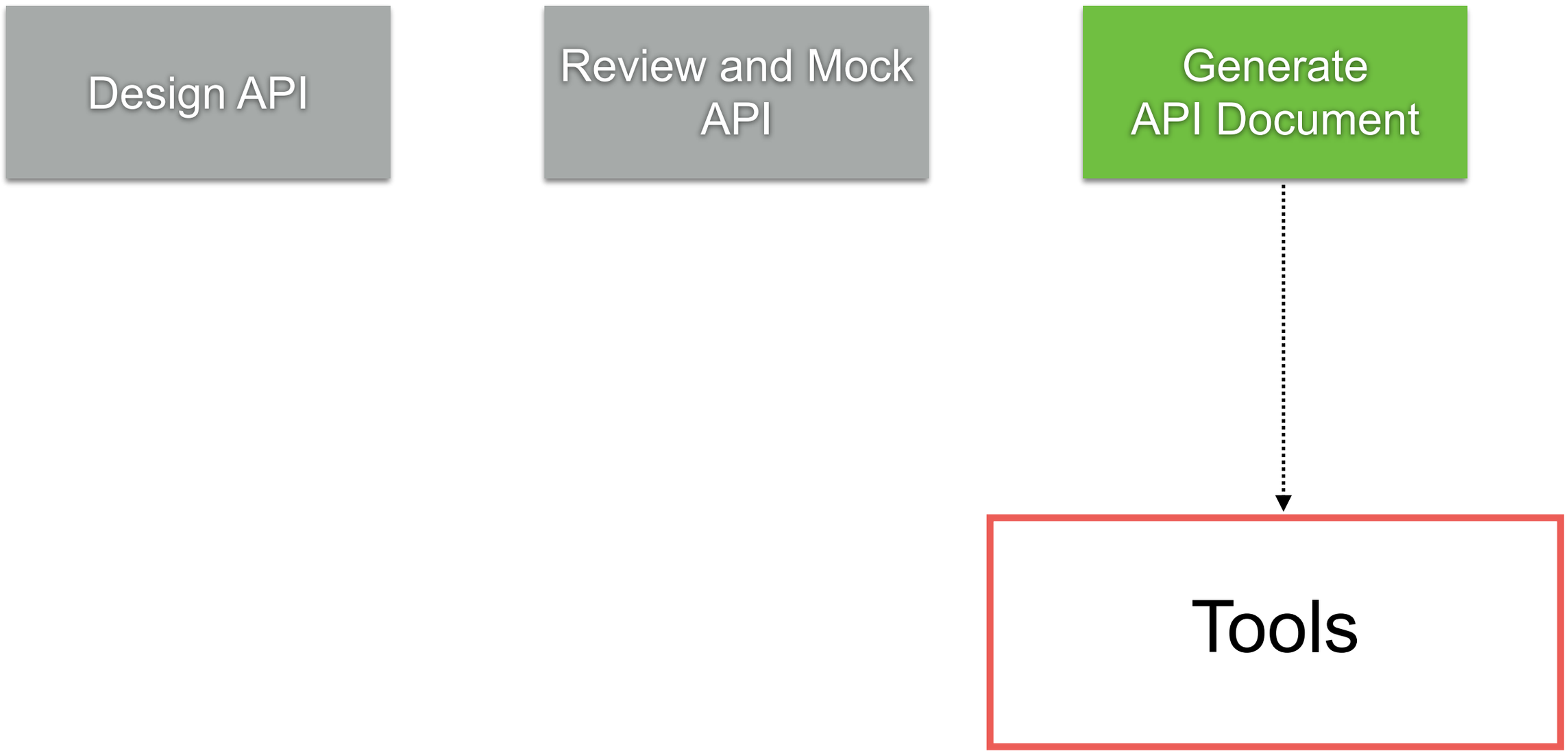
Access to http://localhost:5000

https://github.com/jormaechea/open-api-mocker

Step 3 :: Generate !!



Generate API Document



Generate API Document

Generate HTML from Swagger


```
$npm i -g open-api-mocker
$open-api-mocker -s api.yml -w
```

https://github.com/Redocly/redoc#redoc-cli

Example

Search...

GET /ping

GET /users/{id}

API docs by Redocly

/ping

Responses

— 200 OK

GET /ping

/users/{id}

PATH PARAMETERS

id
required

integer
User ID of the user

Responses

> 200 OK

> 404 User not found

GET /users/{id}

Response samples

200 404

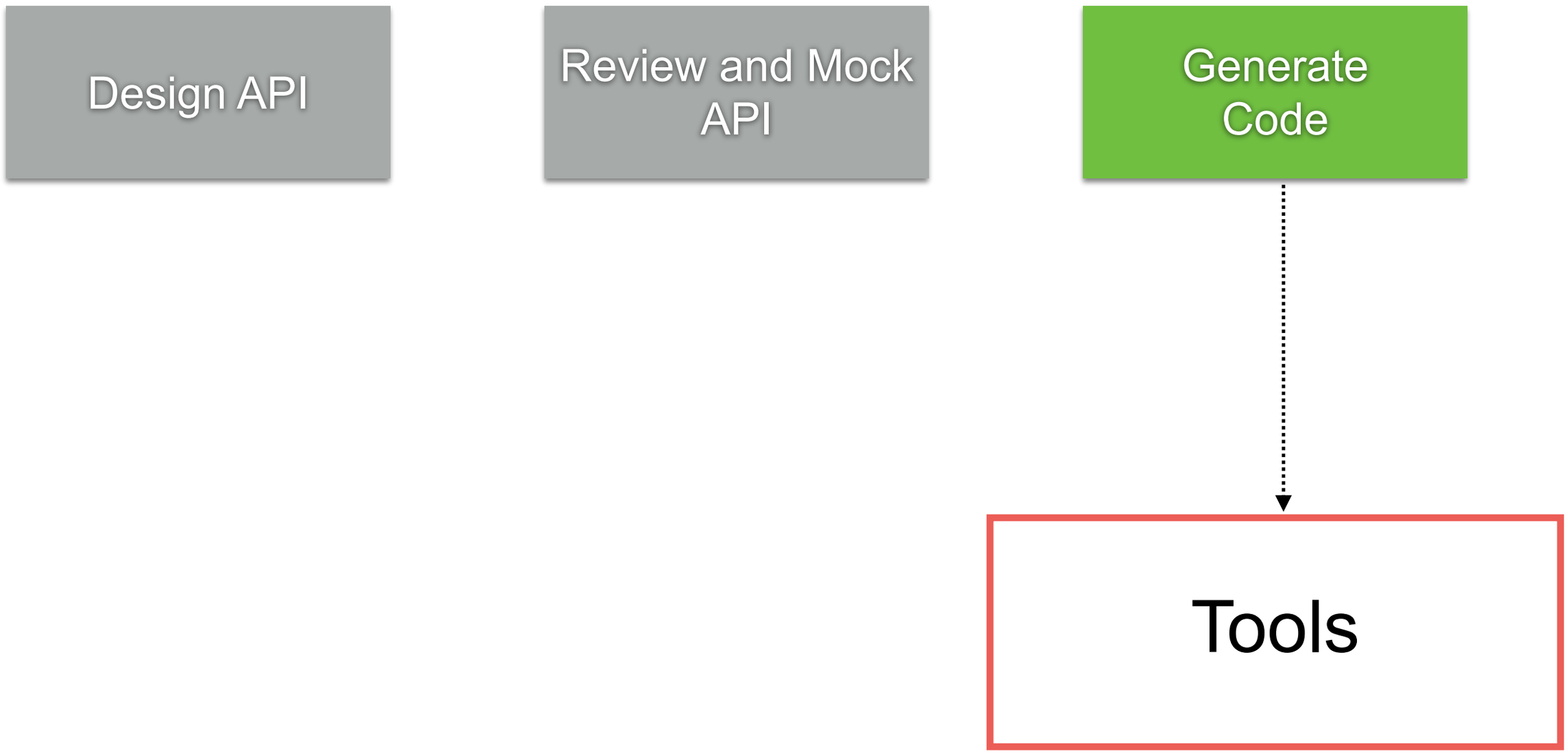
Content type
application/json

Copy

```
{
    "id": 1,
    "name": "User 01"
}
```

https://github.com/Redocly/redoc#redoc-cli

Generate Code



Generate Code from Swagger

Server-side
Client-side

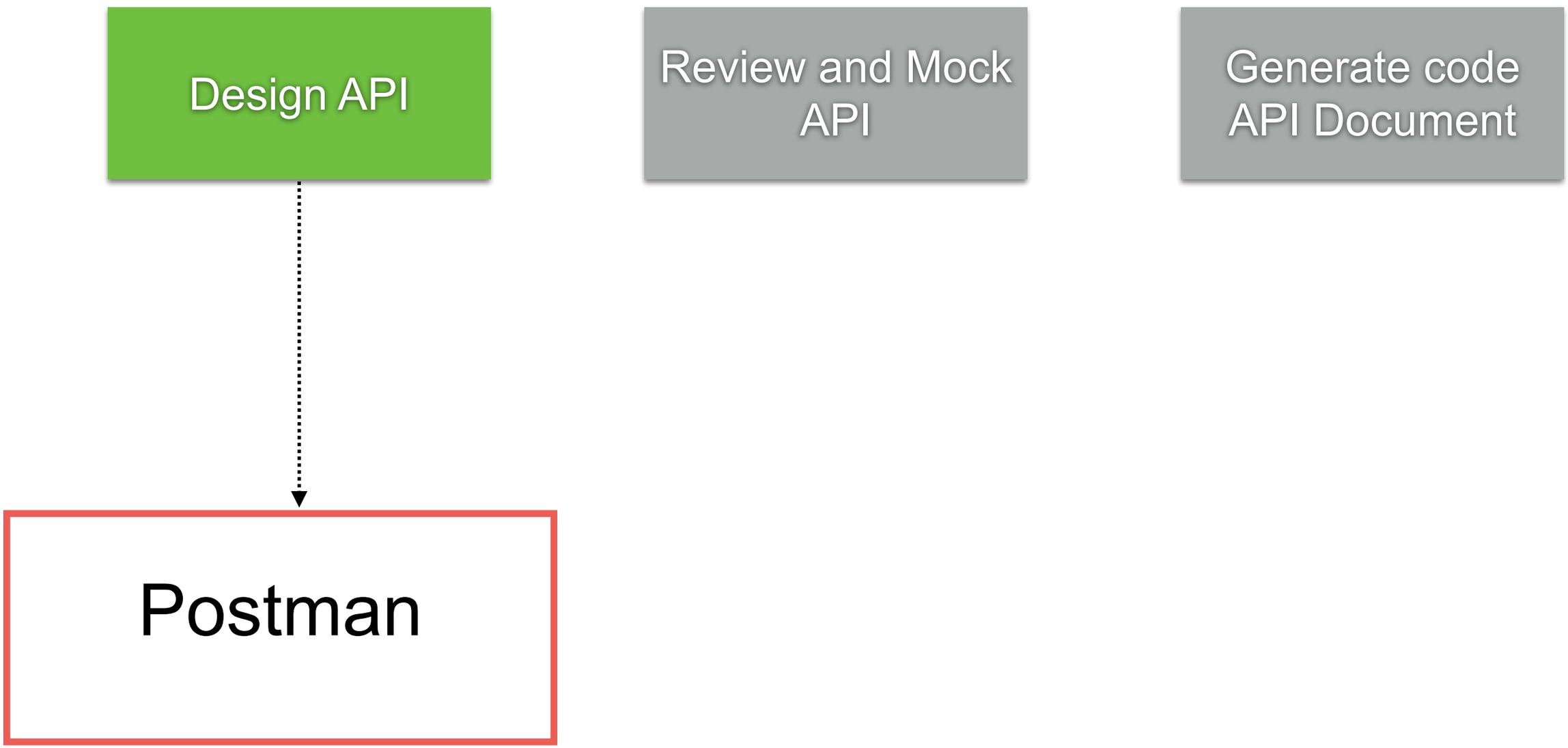
https://github.com/swagger-api/swagger-codegen

Working with Postman API-first

POSTMAN

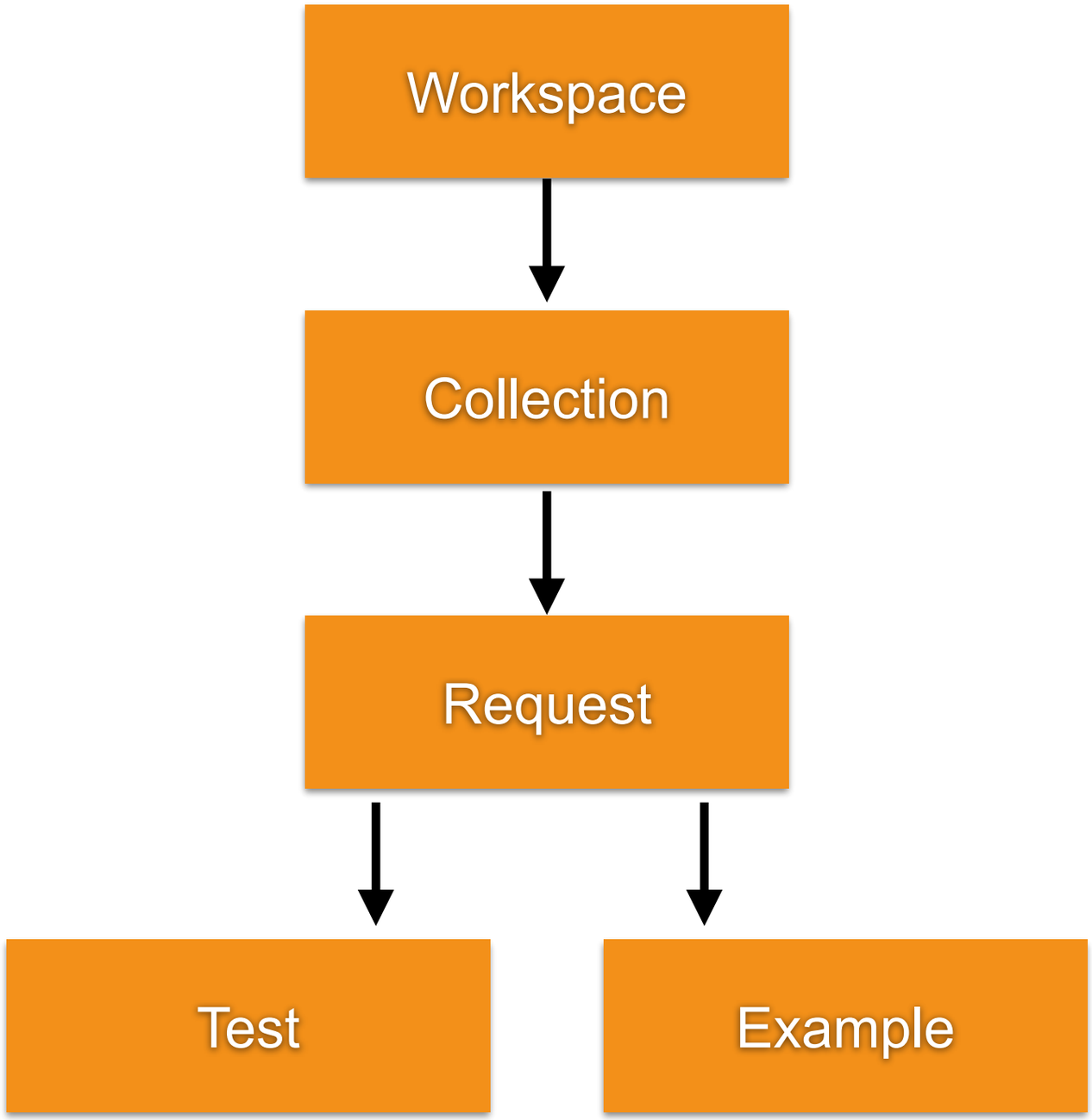
https://www.postman.com/api-first/

Step 1 :: Design API

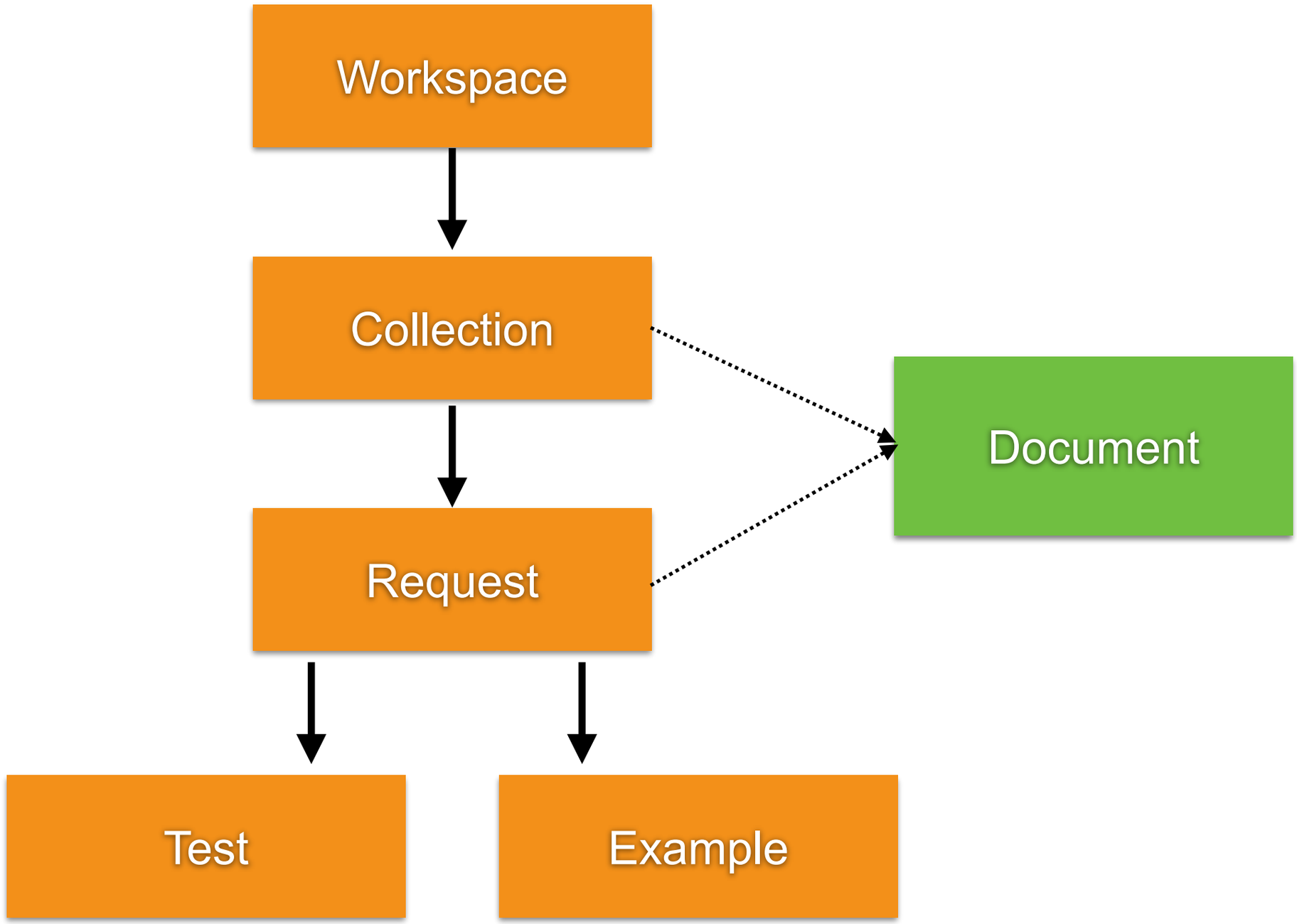


https://www.postman.com/

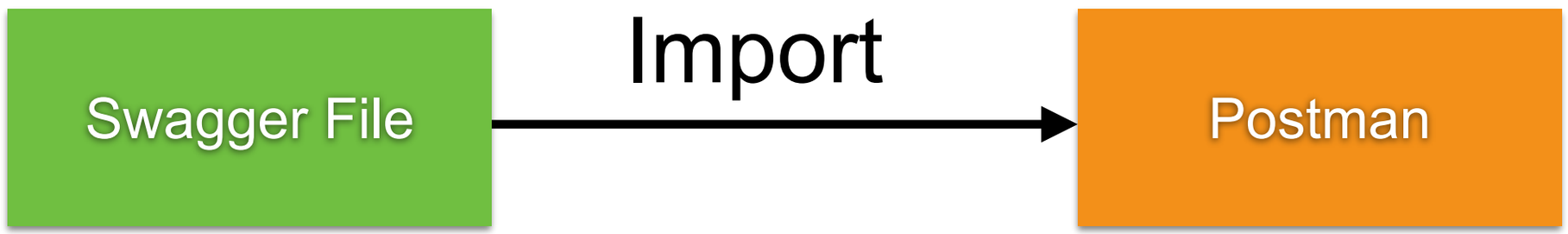
API-first in Postman



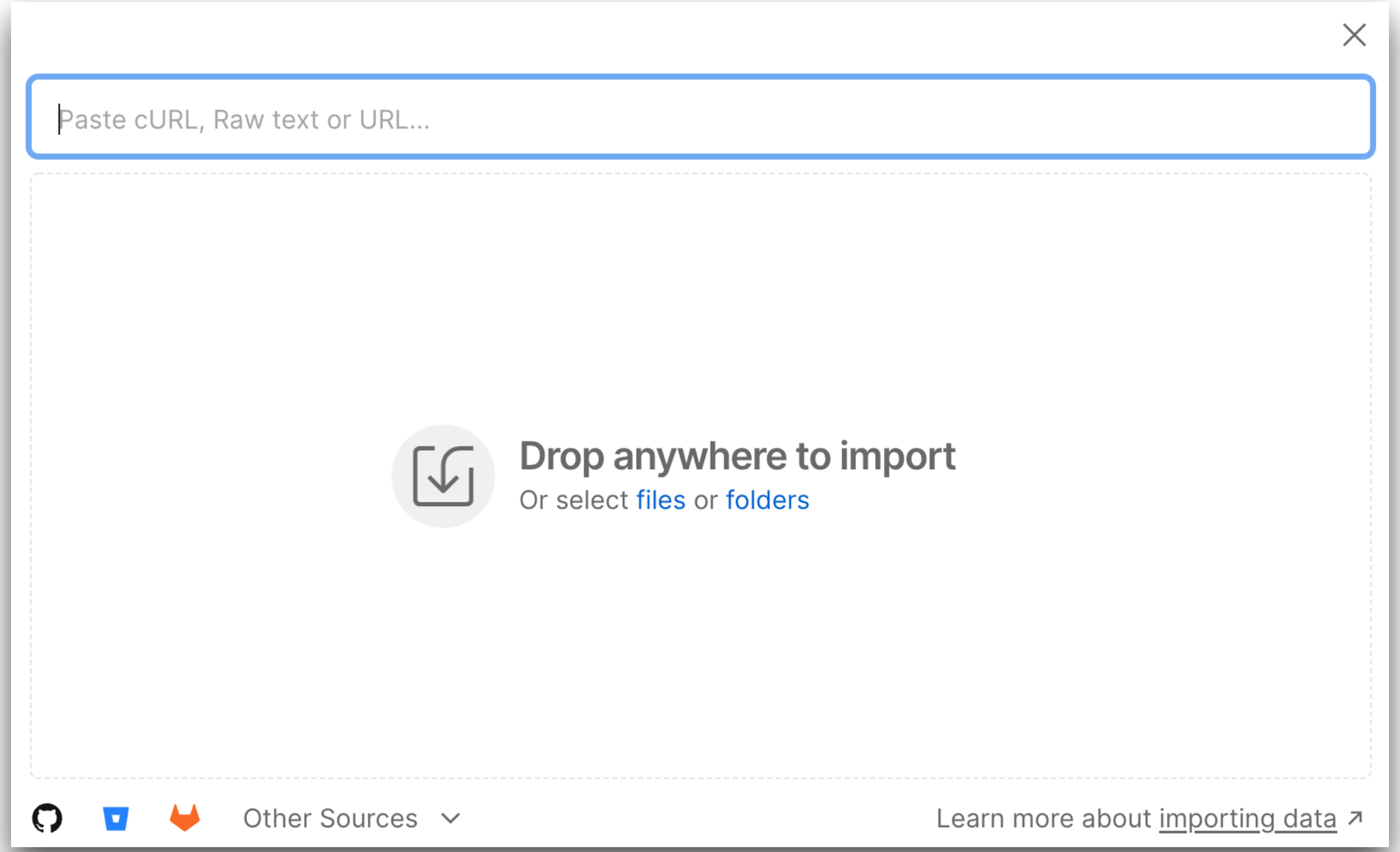
API-first in Postman



Import from Swagger/OpenAPI



Import with OpenAPI file



Example for Design your APIs

New Import

GET /users/:id

Demo
- users
  - {id}
    - GET /users/:id
      - OK
      - User not found
- ping
  - GET /ping
    - OK

HTTP / users / {id} / /users/:id

GET {{baseUrl}}/users/:id

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	Key	Value
	Key	Value

Path Variables

	Key	Value
	id	<integer>

Postman Collection to OpenAPI

Transform Postman Collections into OpenAPI

Fork me on GitHub

postman2openapi

Postman Collection JSON:


```
{
  "info": {
    "_postman_id": "b49384f5-dce6-49b9-8d65-b805deb3eb67",
    "name": "Postman Echo",
    "description": "Postman Echo is service you can use to test your REST clients and mal
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
  },
  "item": [
    {
      "name": "Request Methods",
      "item": [
        {
          "name": "GET Request",
          "event": [
            {
              "listen": "test",
              "script": {
                "id": "13f30c4a-447c-4bba-930f-a023343830fa",
                "exec": [
                  "pm.test(\"response is ok\", function () {",
                  "    pm.response.to.have.status(200);",
                  "});",
                  "",
                  "pm.test(\"response body has json with request queries\", function ()
                  "    pm.response.to.have.jsonBody('args.foo1', 'bar1')",
                  "        .and.have.jsonBody('args.foo2', 'bar2');",
                  "});"
                ],
                "type": "text/javascript"
              }
            }
          ],
          "request": {
            "method": "GET",
            "header": [],
            "url": {
              "raw": "https://postman-echo.com/get?foo1=bar1&foo2=bar2",
              "protocol": "https",
              "host": [
                "postman-echo",
                "com"
              ],
```

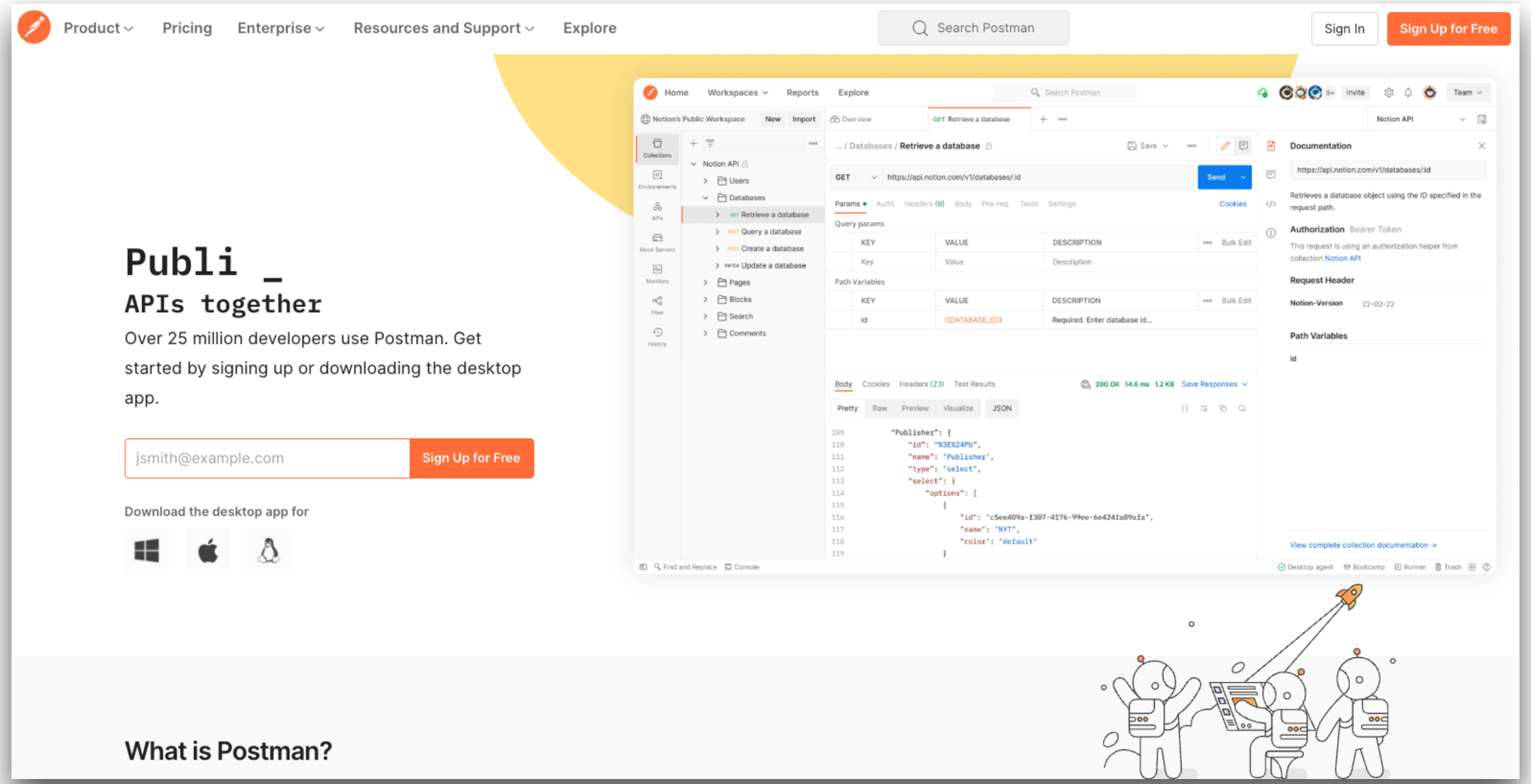

OpenAPI:


```
openapi: 3.0.3
info:
  title: Postman Echo
  description: >-
    Postman Echo is service you can use to test your REST clients and make
    sample API calls. It provides endpoints for `GET`, `POST`, `PUT`, various
    auth mechanisms and other utility endpoints.


    The documentation for the endpoints as well as example responses can be
    found at
    [https://postman-echo.com](https://postman-echo.com/?source=echo-collection-app-onboarding
  version: 1.0.0
  contact: {}
servers:
  - url: https://postman-echo.com
paths:
  /get:
    get:
      tags:
        - Request Methods
      summary: GET Request
      description: >-
        The HTTP `GET` request method is meant to retrieve data from a server.
        The data

        is identified by a unique URI (Uniform Resource Identifier).


        A `GET` request can pass parameters to the server using "Query String

        Parameters". For example, in the following request,


        > http://example.com/hi/there?hand=wave


        The parameter "hand" has the value "wave".


        This endpoint echoes the HTTP headers, request parameters and the
        complete
```


https://kevinswiber.github.io/postman2openapi/

API Testing with Postman and newman

Postman

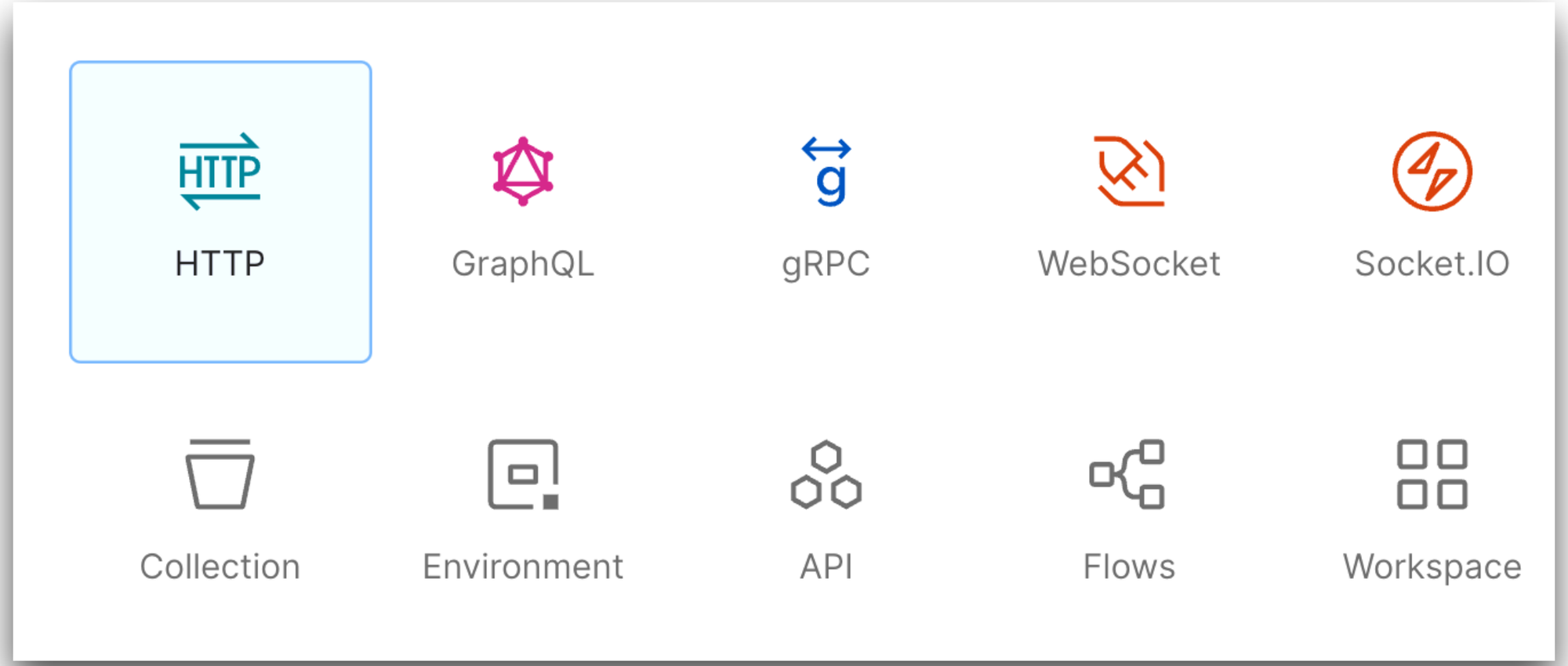


https://www.postman.com/

Topics with Postman

Collections
HTTP Request
Testing in HTTP Request
Import/Export collection
Mock Server

Testing with Postman



Topics with Postman

demo-nodejs | New | Import

Collections
Environments
History

group01
GET Hello API

GET Hello API | No Environment

group01 / Hello API | Save

GET | http://localhost:3000/ | Send

Params | Auth | Headers (6) | Body | Pre-req. | Tests | Settings | Cookies

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Check hello message", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.message).to.eql('Hello World');
});

var schema = {
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "message": {
      "type": "string"
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Snippets

- Response body: Contains string
- Response body: JSON value check
- Response body: Is equal to a string
- Response headers: Content-Type header check
- Response time is less than 200ms

Body | Cookies | Headers (7) | Test Results (3/3) | 200 OK | 18 ms | 260 B | Save as Example

Pretty | Raw | Preview | Visualize | JSON

```
{
    "message": "Hello World"
}
```

Testing in HTTP Request

HTTP Response code
HTTP Response Body
Validate JSON Schema

https://learning.postman.com/docs/writing-scripts/test-scripts/

Testing in HTTP Request

GET http://localhost:3000/ Send

Params Auth Headers (6) Body Pre-req Tests Settings Cookies

```
1  pm.test("Status code is 200", function () {
2      pm.response.to.have.status(200);
3  });
4
5  pm.test("Check hello message", function () {
6      var jsonData = pm.response.json();
7      pm.expect(jsonData.message).to.eql('Hello World');
8  });
9
10 var schema = {
11   "$schema": "http://json-schema.org/draft-04/schema#",
12   "type": "object",
13   "properties": {
14     "message": {
15       "type": "string"
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Snippets

Response body: Contains string

Response body: JSON value check

Response body: Is equal to a string

Response headers: Content-Type header check

Response time is less than 200ms

https://learning.postman.com/docs/writing-scripts/test-scripts/

Newman

$npm i -g newman
$newman run <json file>

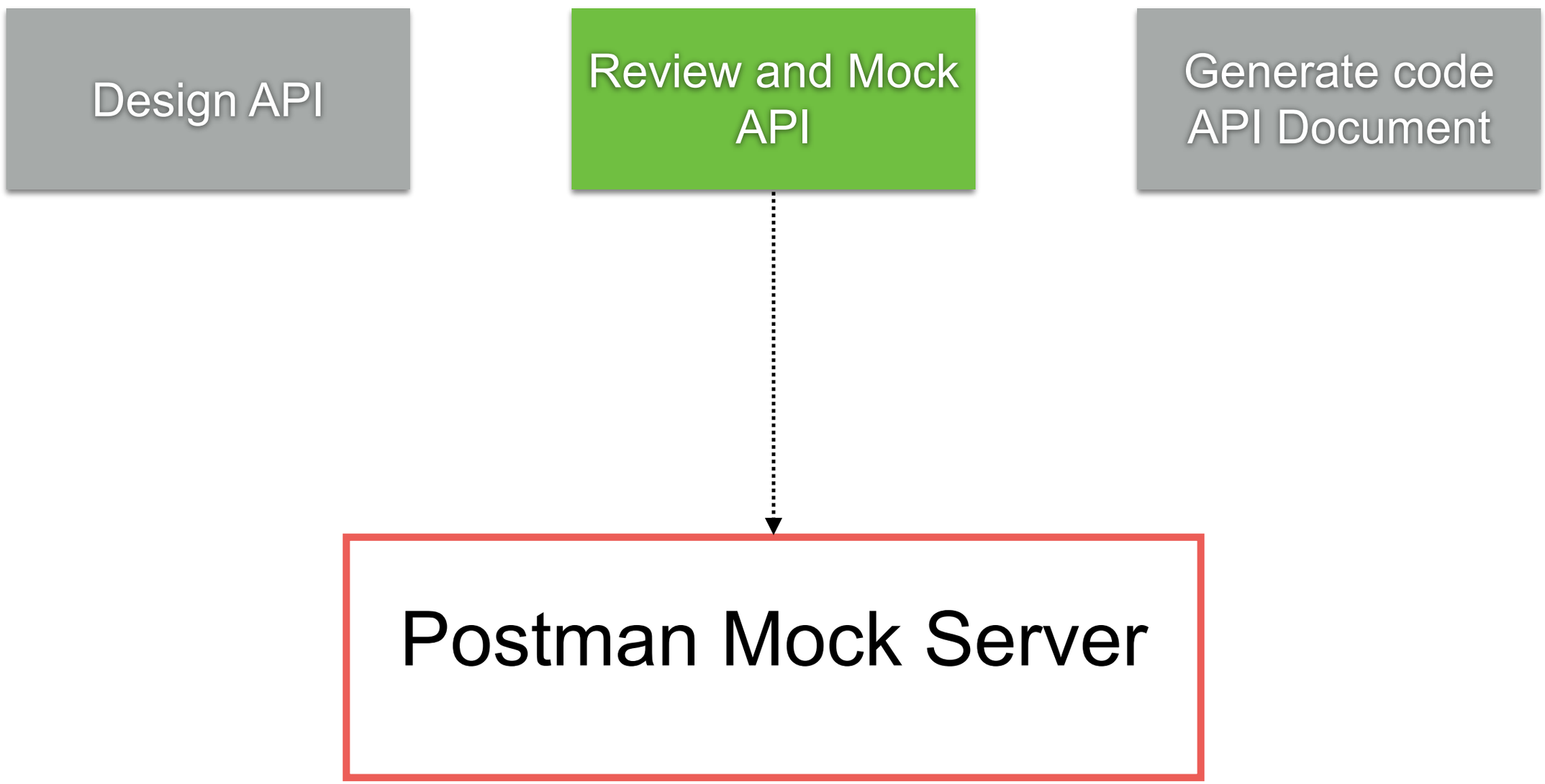
https://www.npmjs.com/package/newman

Newman and report

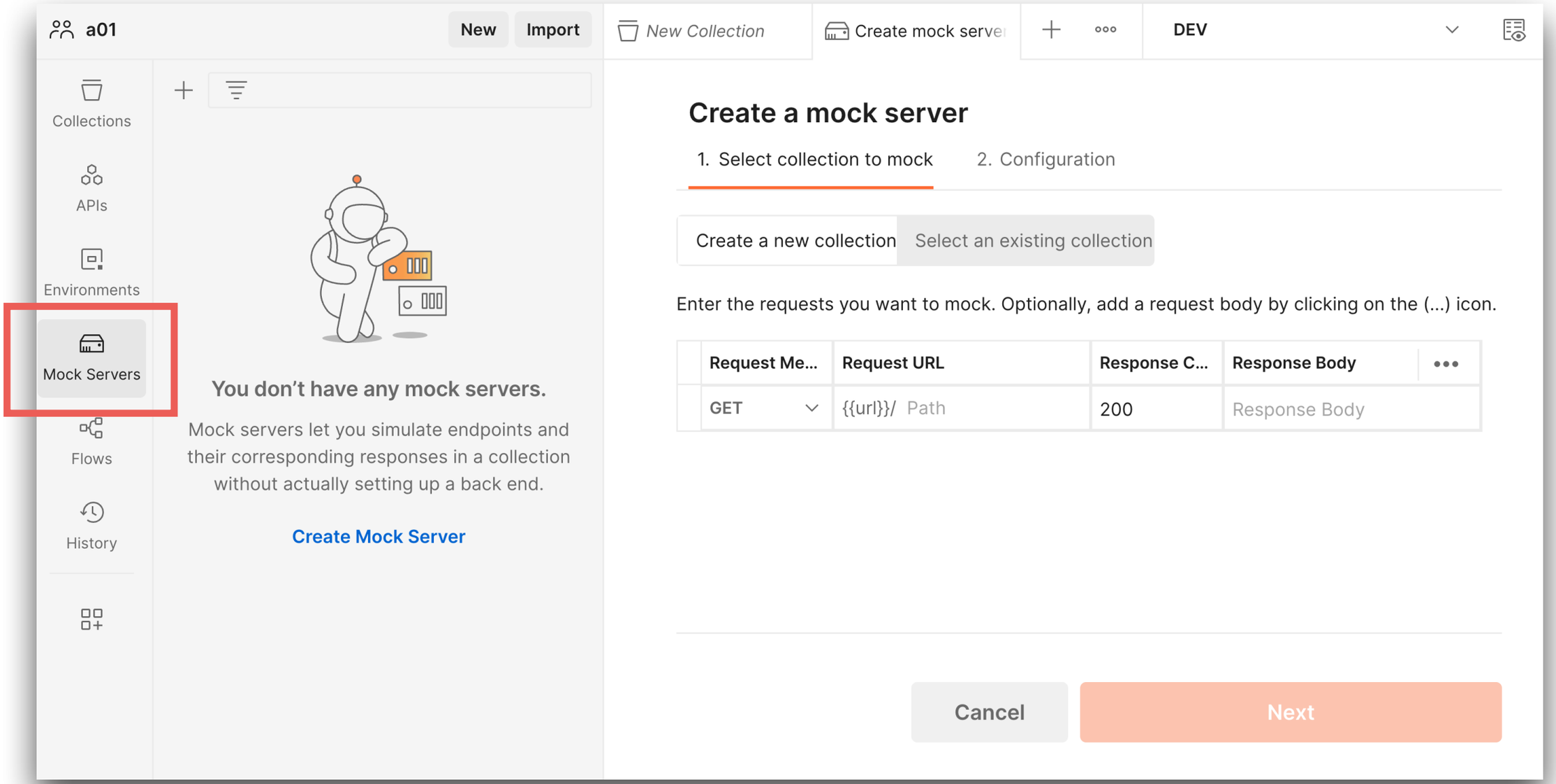
$npm i -g newman
$newman run <json file> -r cli,junit

https://www.npmjs.com/package/newman

Step 2 :: Mock API

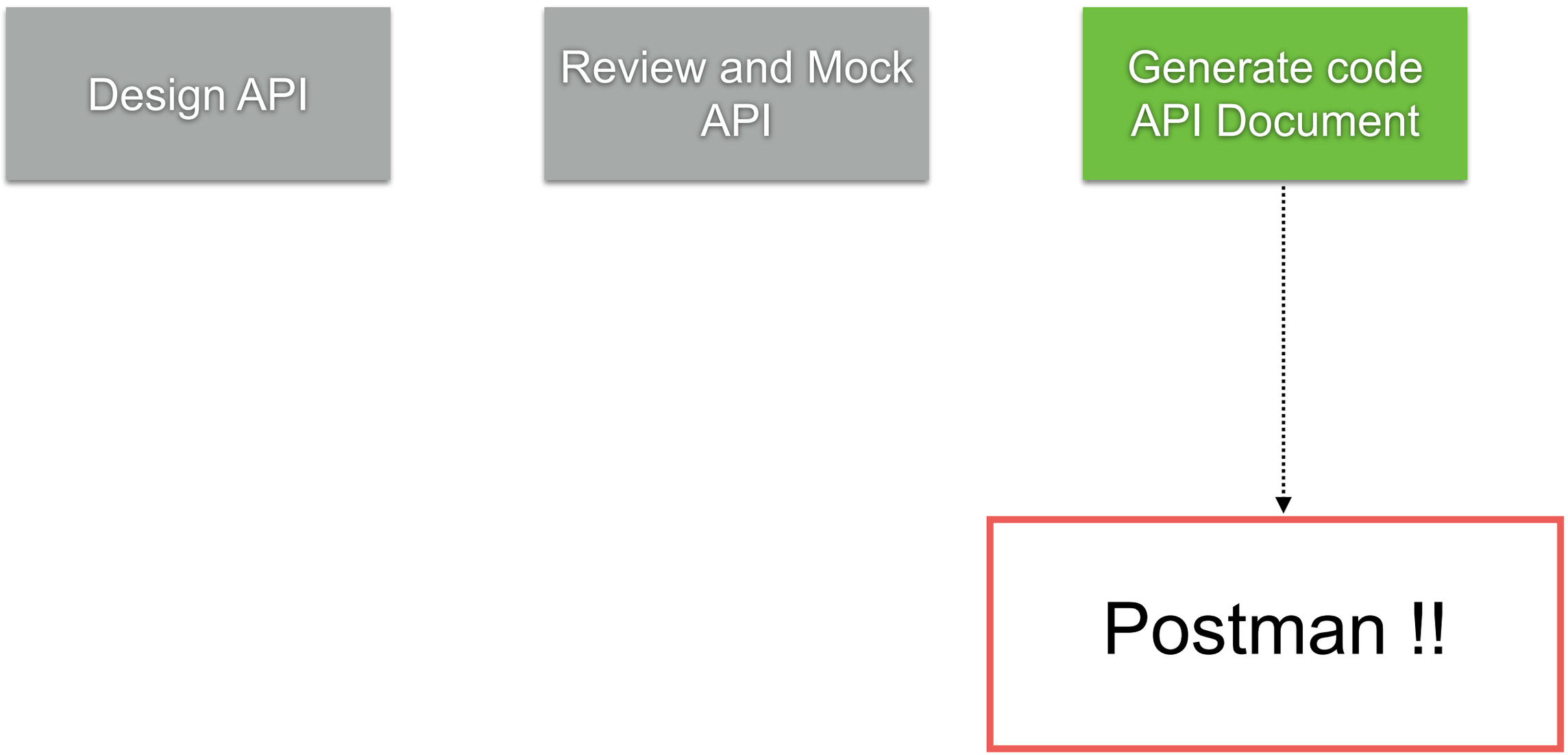


Create Mock Server from Collection

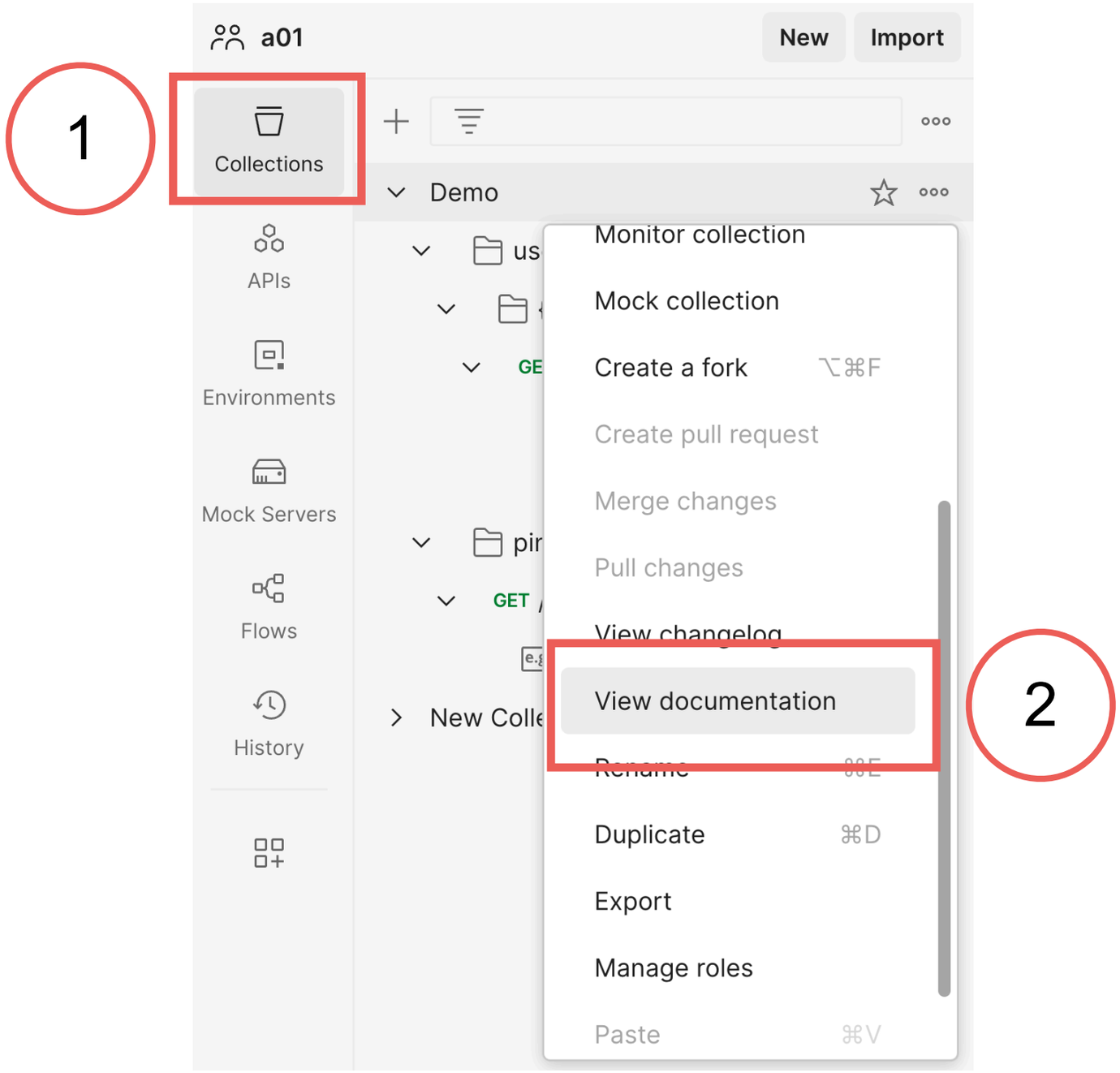


https://learning.postman.com/docs/designing-and-developing-your-api/mocking-data/setting-up-mock/

Step 3 :: Generate !!

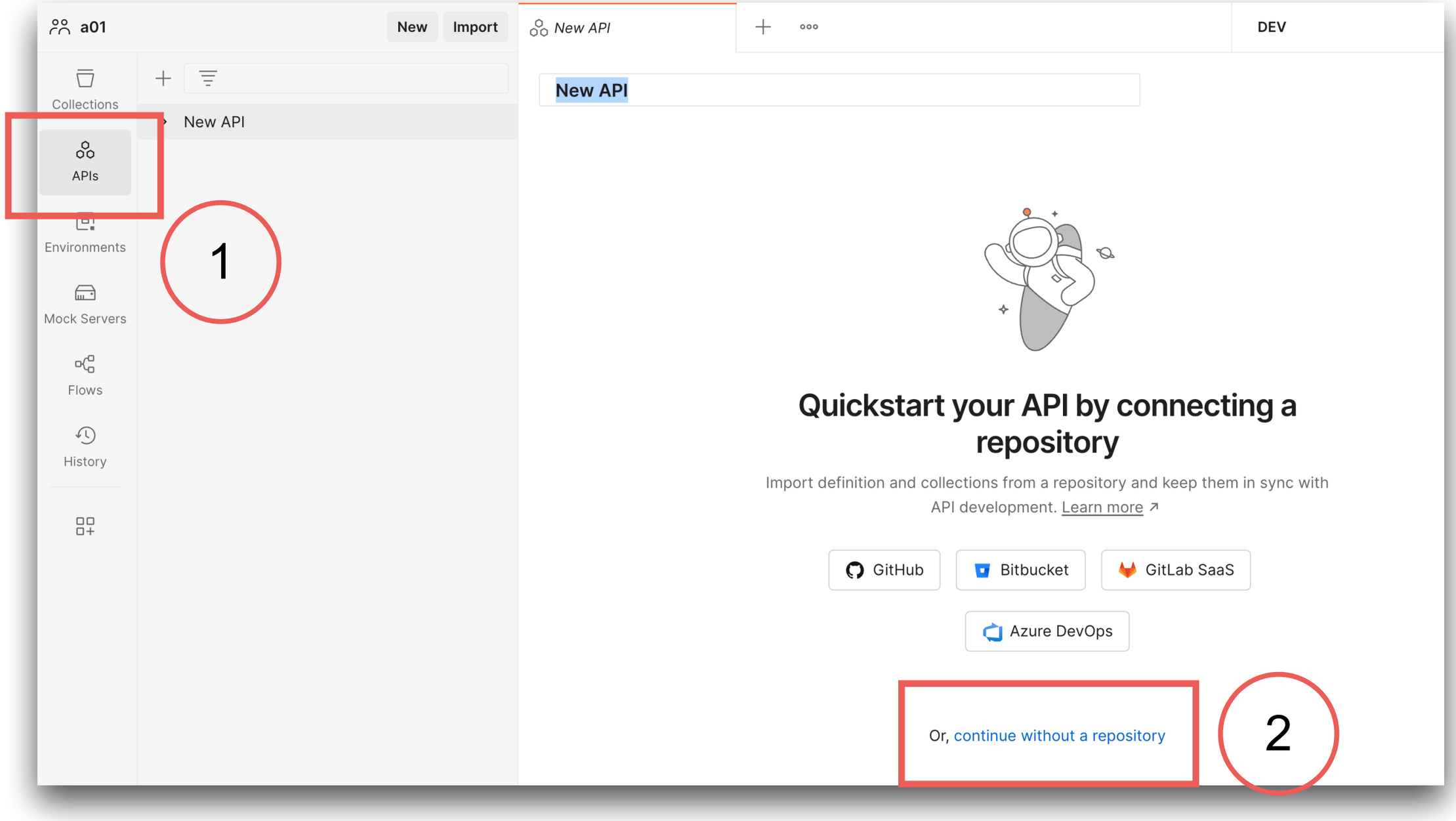


Generate API Document



https://learning.postman.com/docs/publishing-your-api/documenting-your-api/

Generate Code with APIs



https://learning.postman.com/docs/designing-and-developing-your-api/the-api-workflow/

Create APIs from Collections

a01 New Import

Collections
APIs
Environments
Mock Servers
Flows
History

New API
Demo
users
{id}
GET /users/:id
ping
GET /ping

New API

DEV

New API

Share 0

About this API

Add summary...

Add description...

Connect repository

Keep your API development in sync with your code by connecting your repository.

Connect

3

Collections

Demo

Definition

Add files to describe your service and setup the API definition.

Test and Automation →

API Performance →

Deployments →

Publish your API to consumers

Make it easier for your teammates to discover and use the right version of your API.

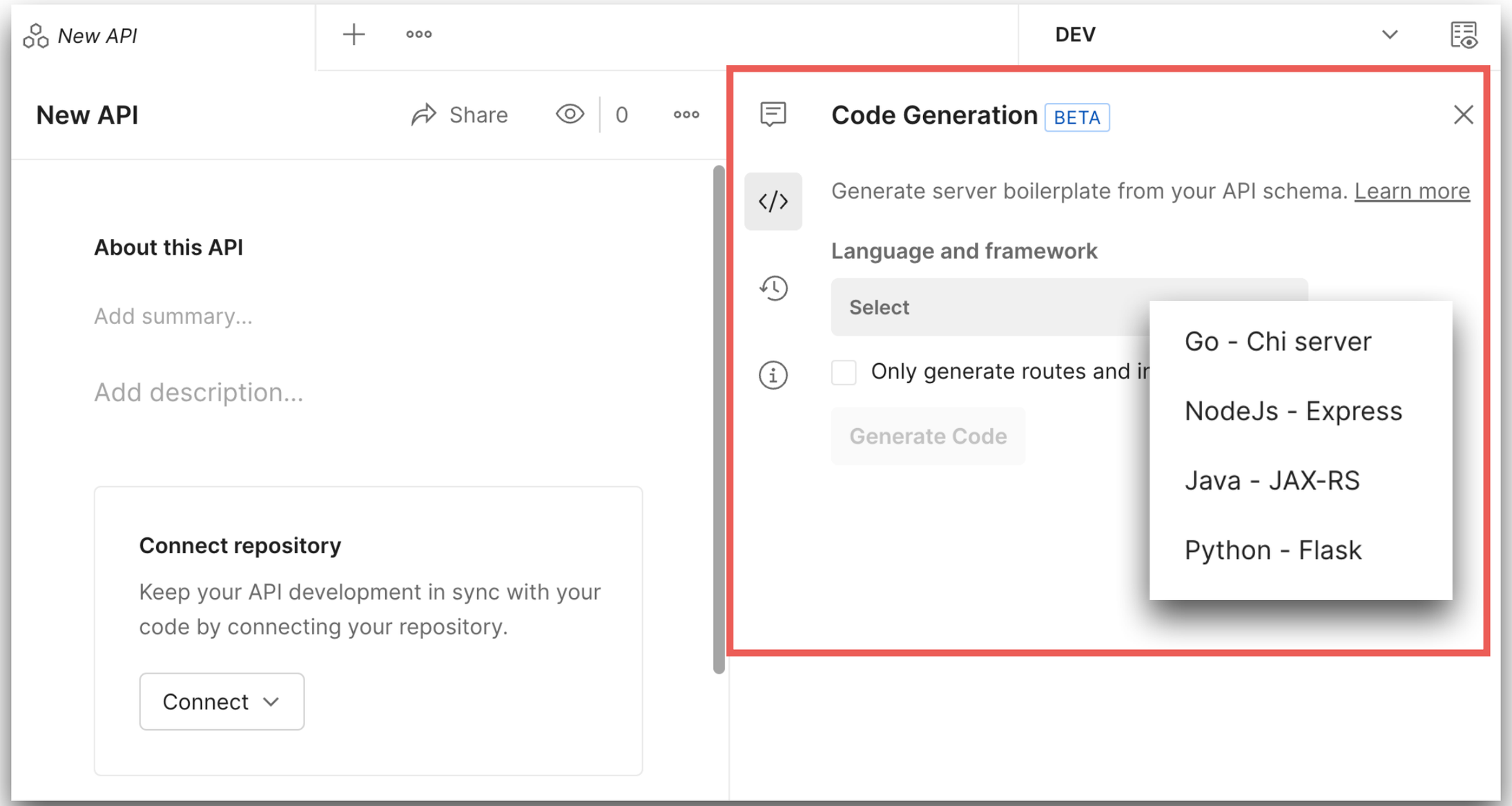
Publish API

Editors

You

https://learning.postman.com/docs/designing-and-developing-your-api/developing-an-api/generating-server-code/

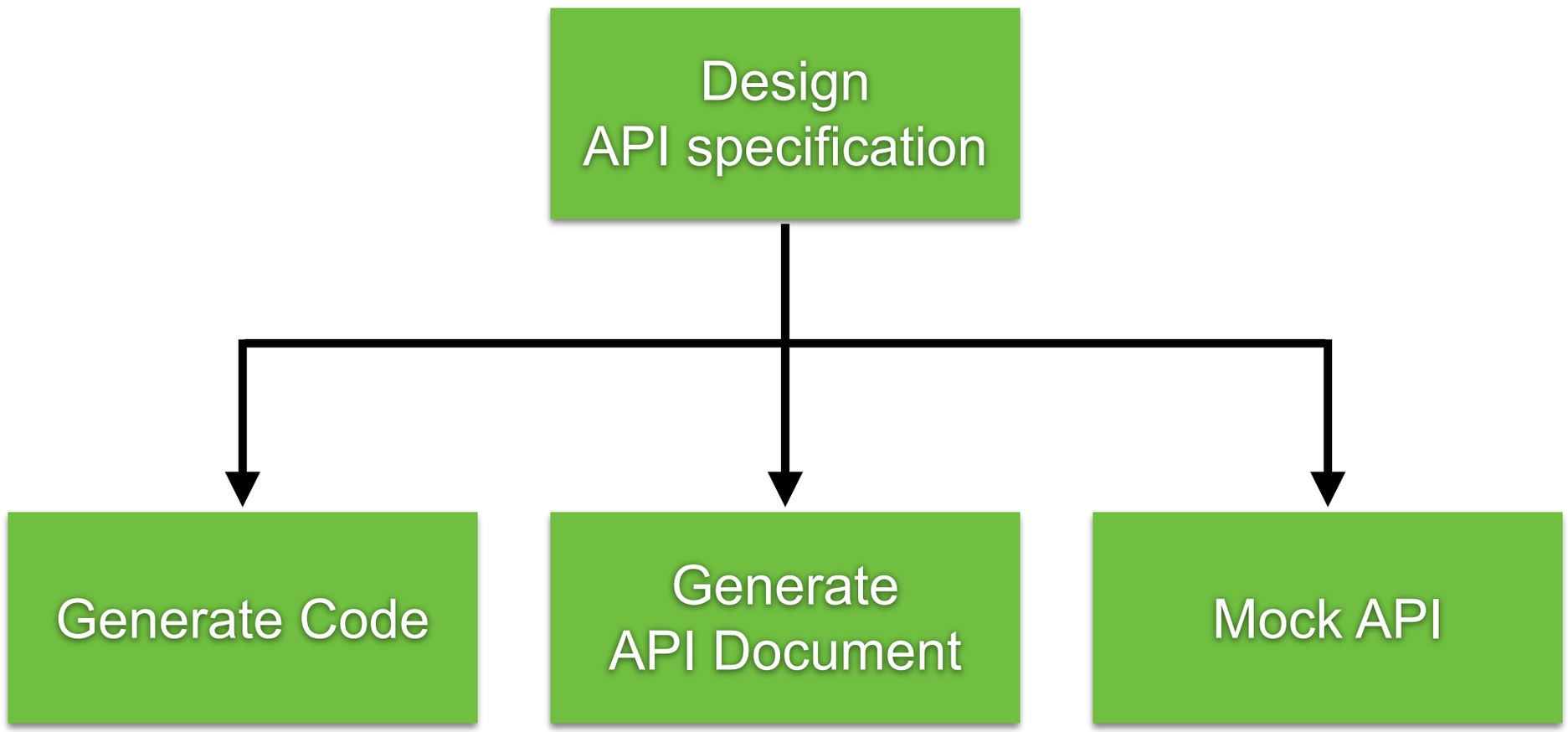
Generate code



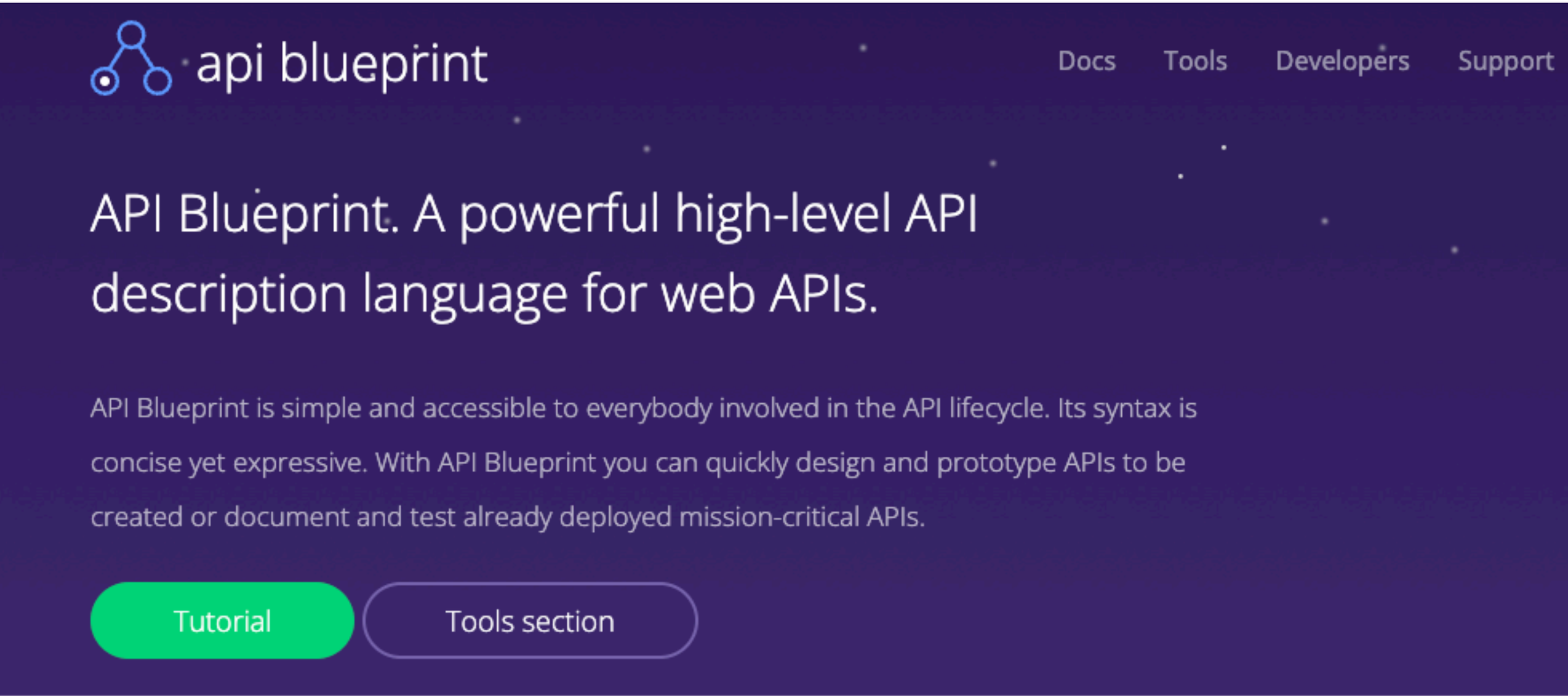
https://learning.postman.com/docs/designing-and-developing-your-api/developing-an-api/generating-server-code/

More Solutions ?

Idea !!



API Blueprint



https://apiblueprint.org/

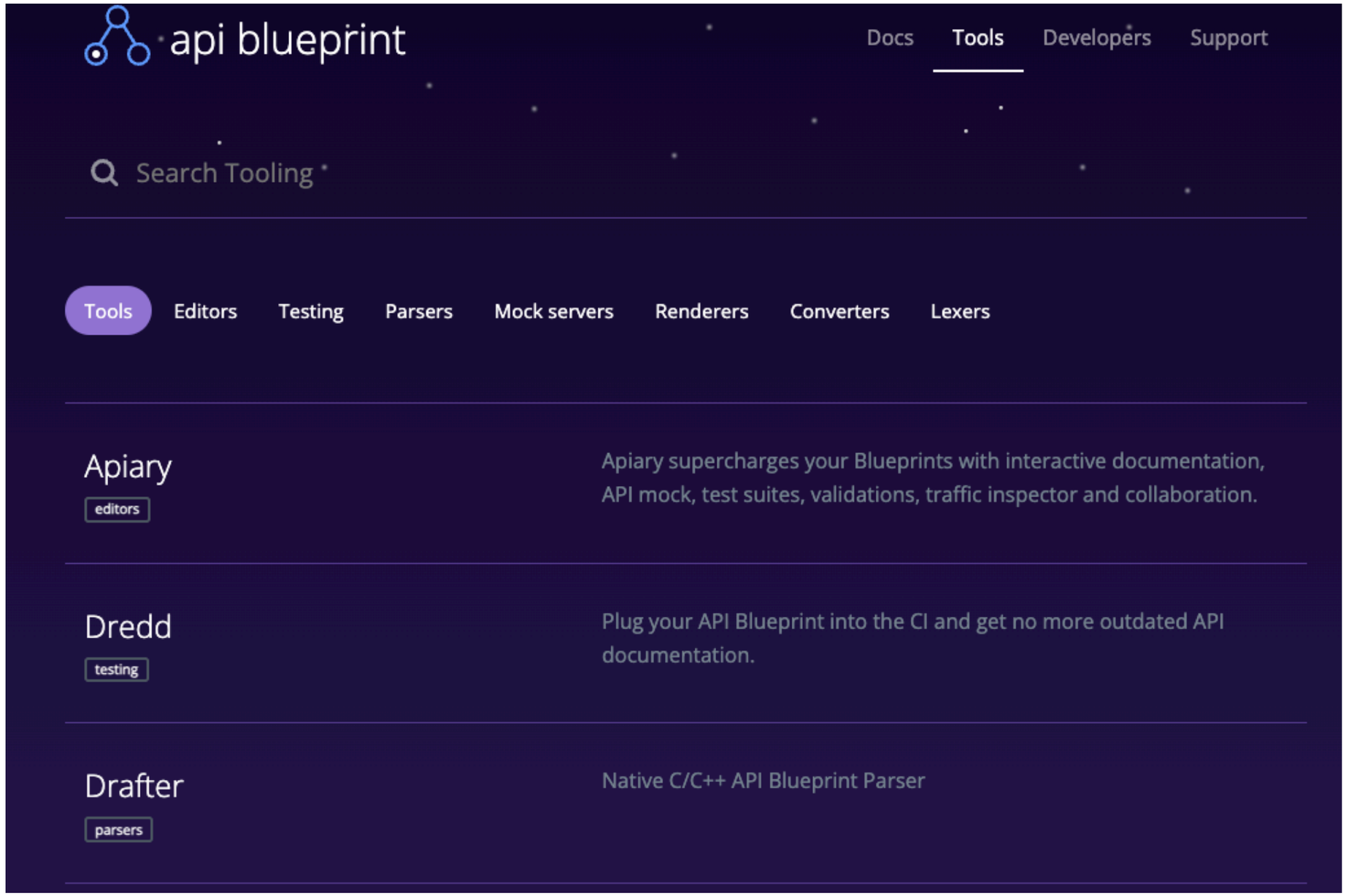
API Blueprint

Design APIs with Simple format !!

```
FORMAT: 1A
HOST: http://localhost:3000

# Demo API

Demo APIs for API-first solution !!.

# Ping service [/ping]
## Ping [GET]
+ Response 200
```

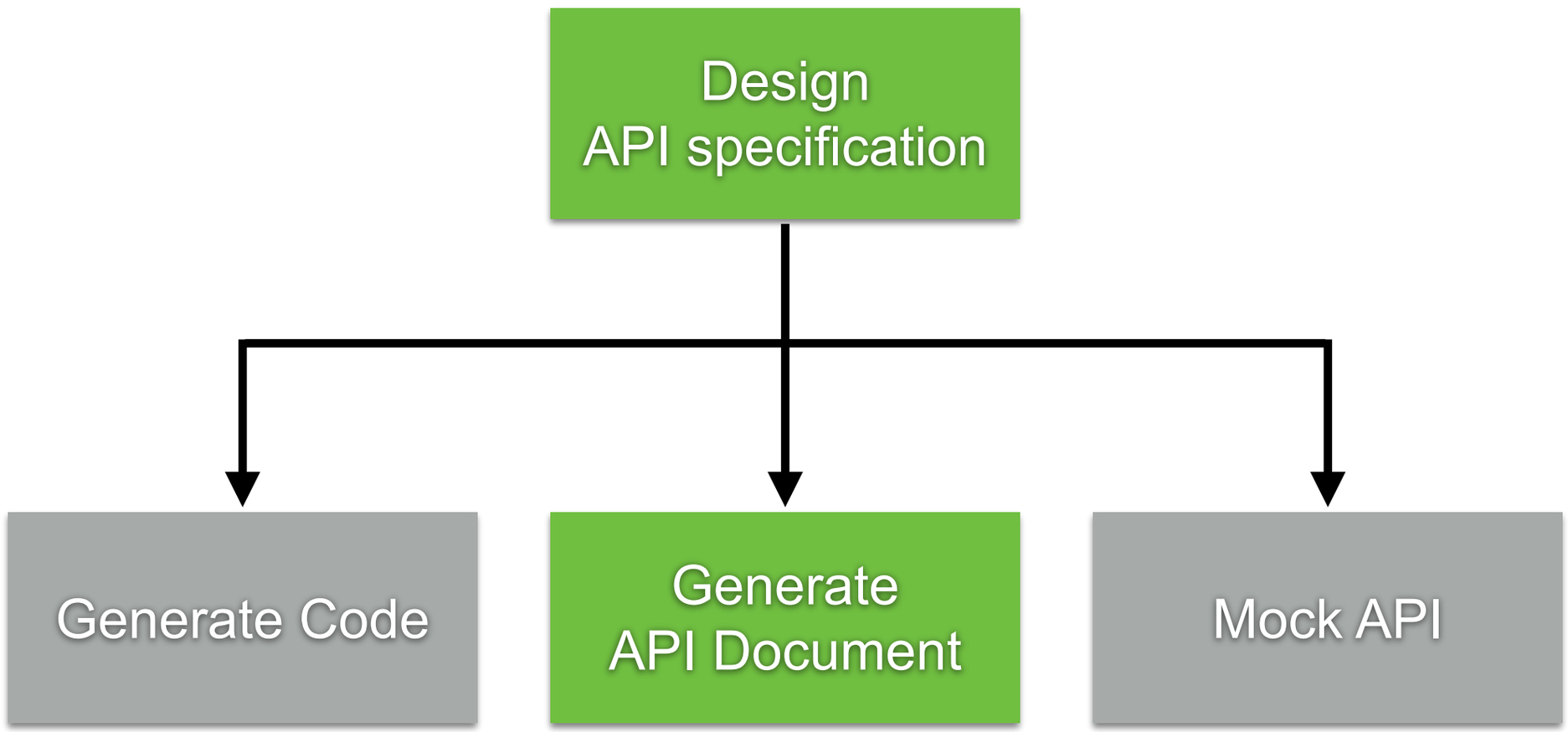
https://apiblueprint.org/tools.html

API Blueprint Tools !!



https://apiblueprint.org/

API Blueprint



Generate API Document

$npm i -g aglio
$aglio -i example-api.apib -o output.html

aglio api blueprint renderer

https://www.npmjs.com/package/aglio

Generate API Document

Resource Group

Ping

Get user by id

http://localhost:3000

Demo API

Demo APIs for API-first solution !!.

Resource Group

PING SERVICE

GET `/ping` Ping

Example URI

GET http://localhost:3000/ping

Response 200

GET USER BY ID

GET `/users/{id}` Get user by id

Example URI

GET http://localhost:3000/users/id

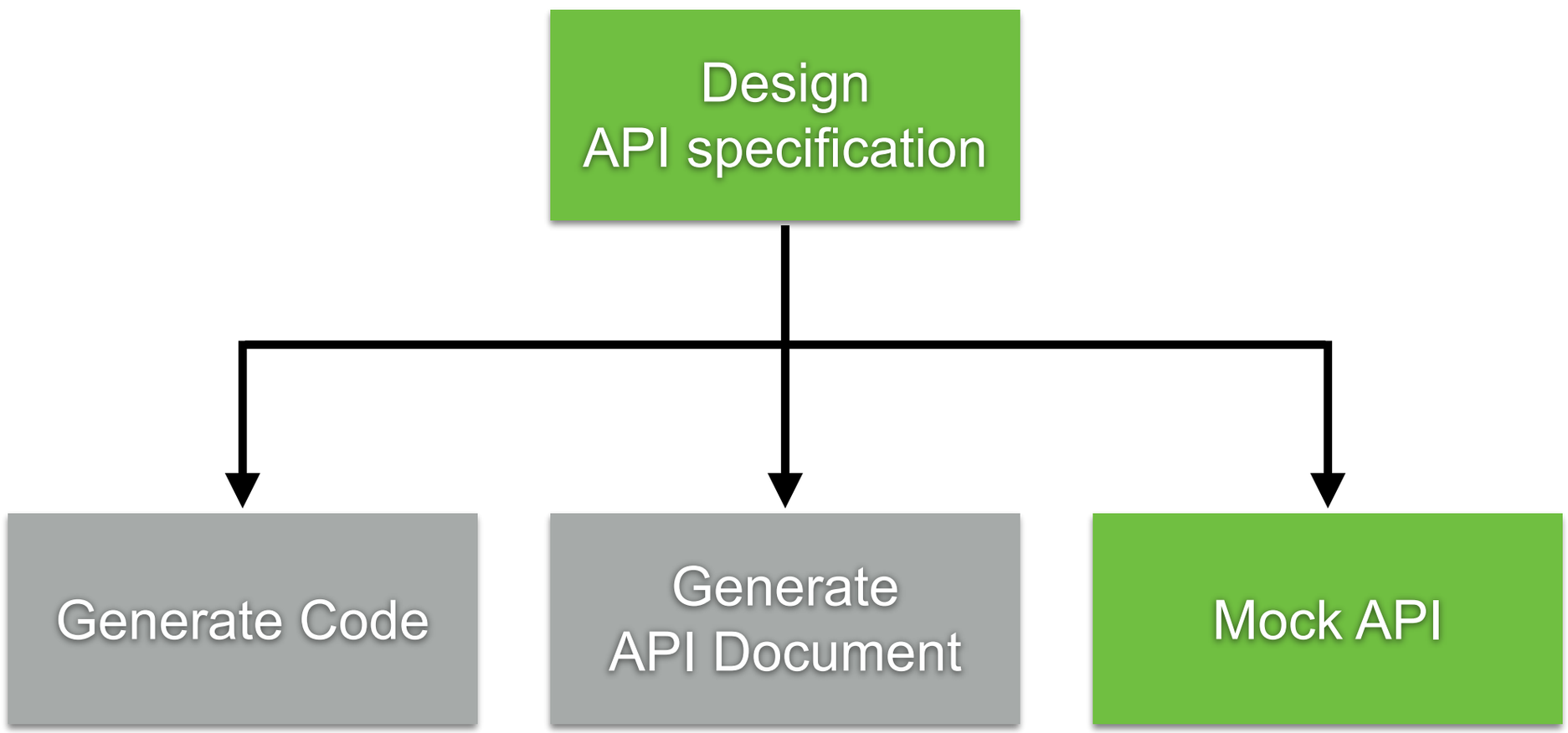
URI Parameters Hide

id `string` (required)

Response 200 Show

https://www.npmjs.com/package/aglio

API Blueprint



Create Mock API

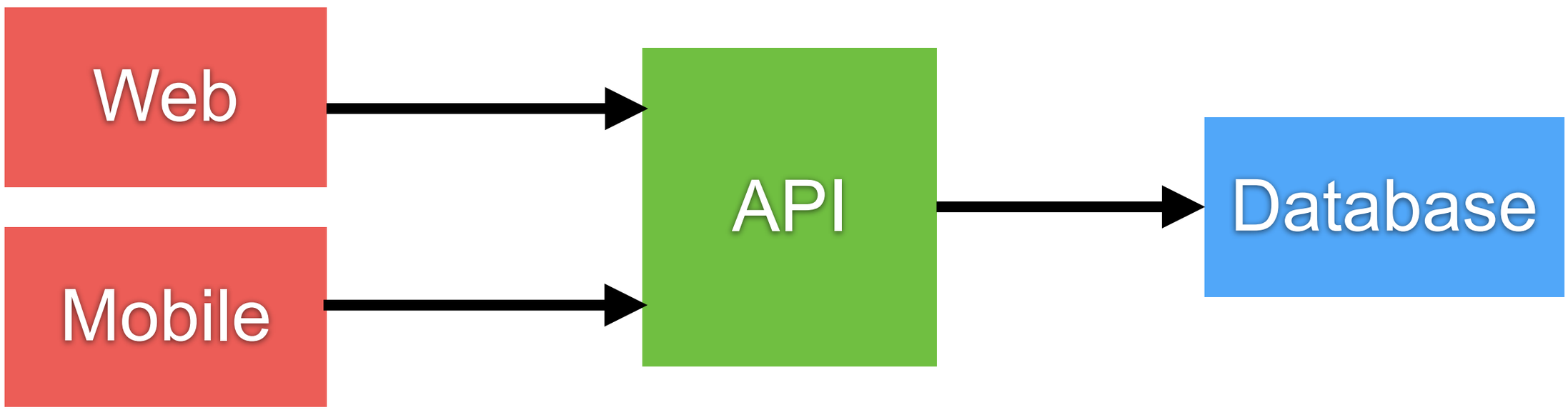
$npm i -g drakov

$drakov -f example-api.apib --watch --public -p 3000

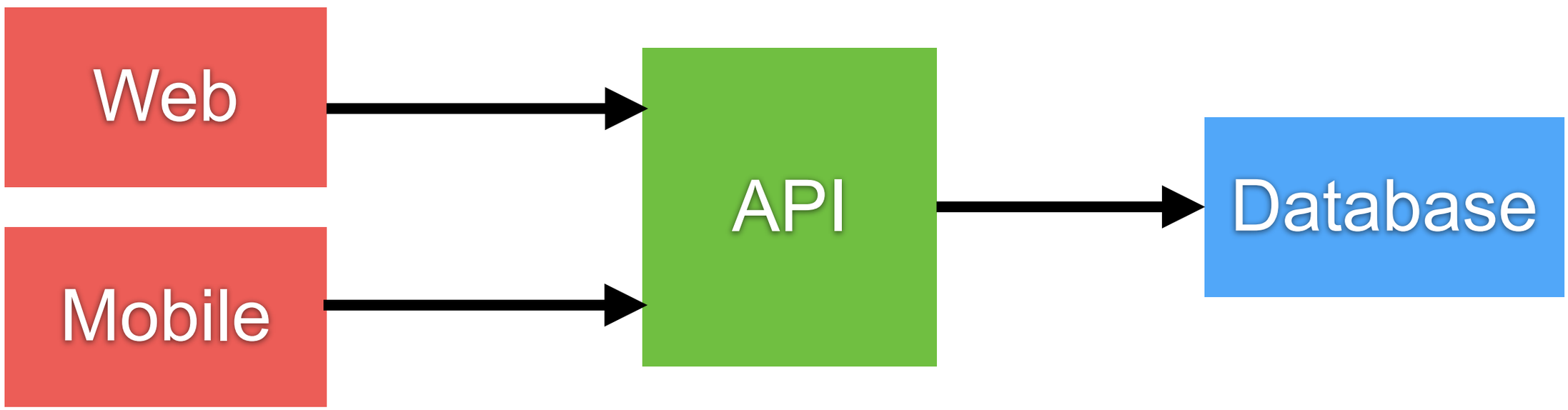
https://www.npmjs.com/package/drakov

What, Why and How to Test ?

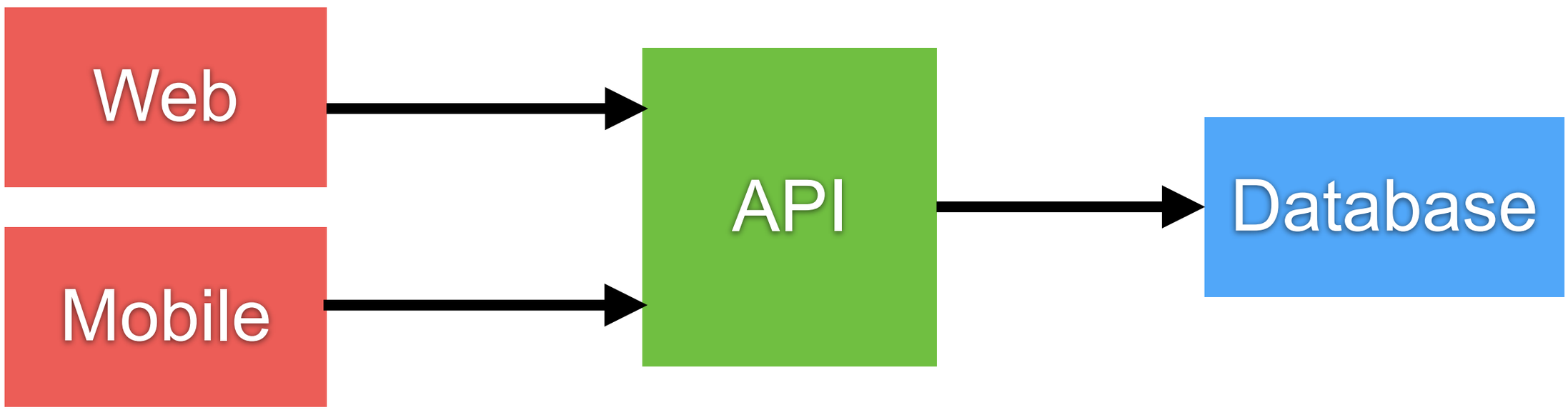
Architecture



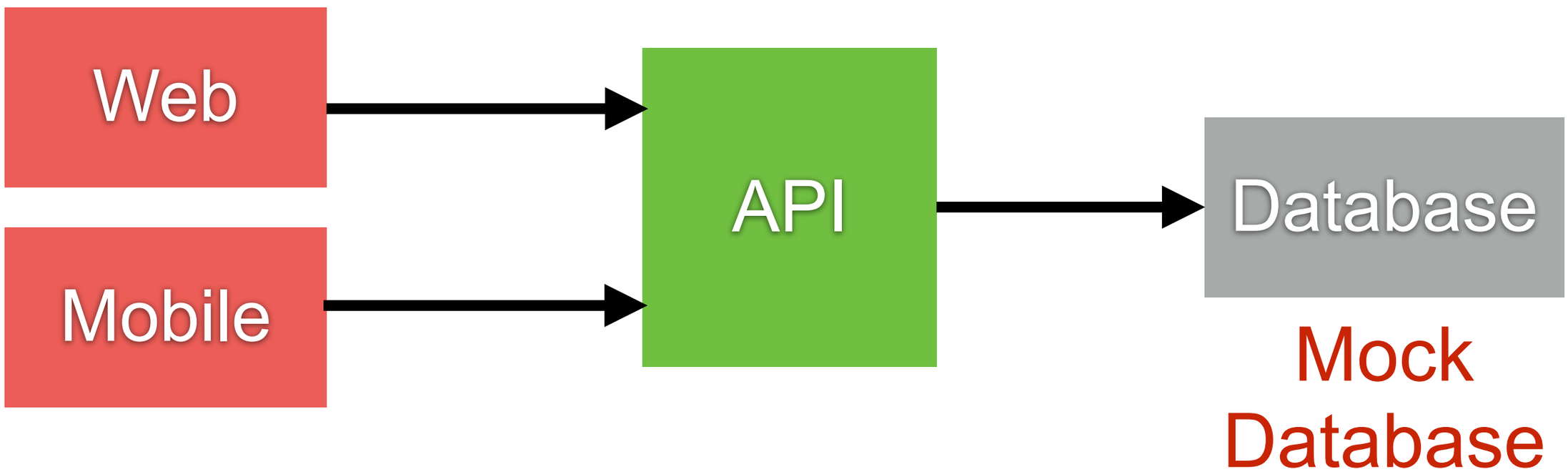
Test ?



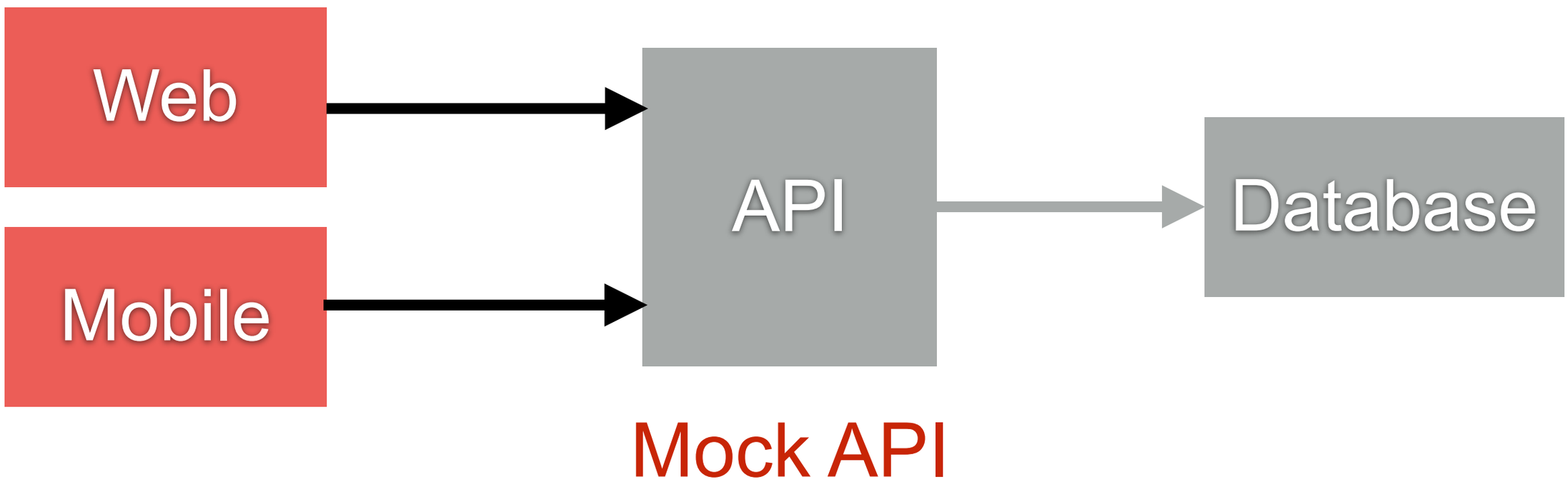
UI Test ?



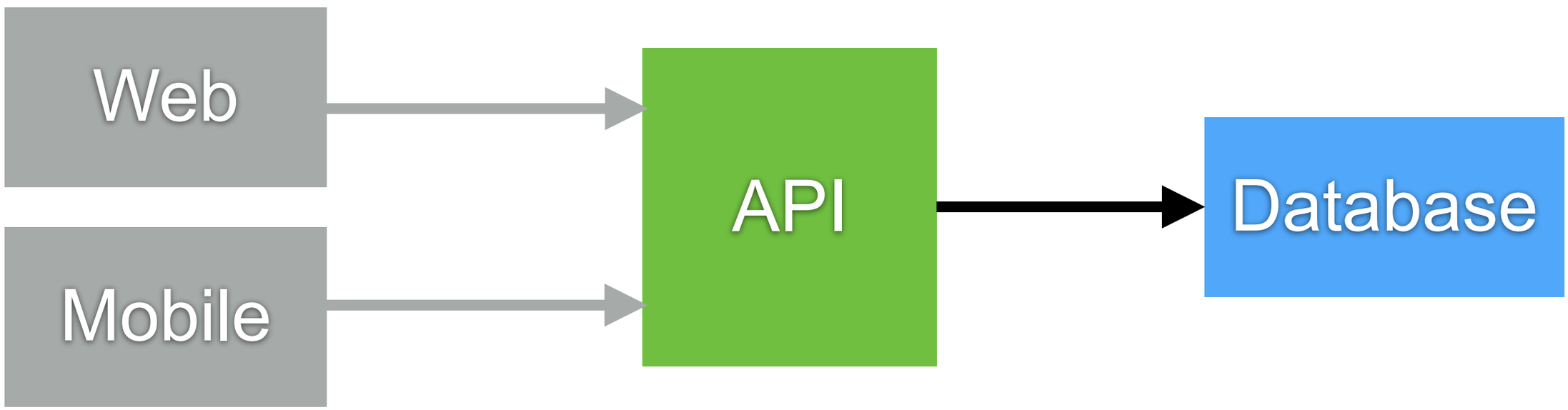
UI Test ?



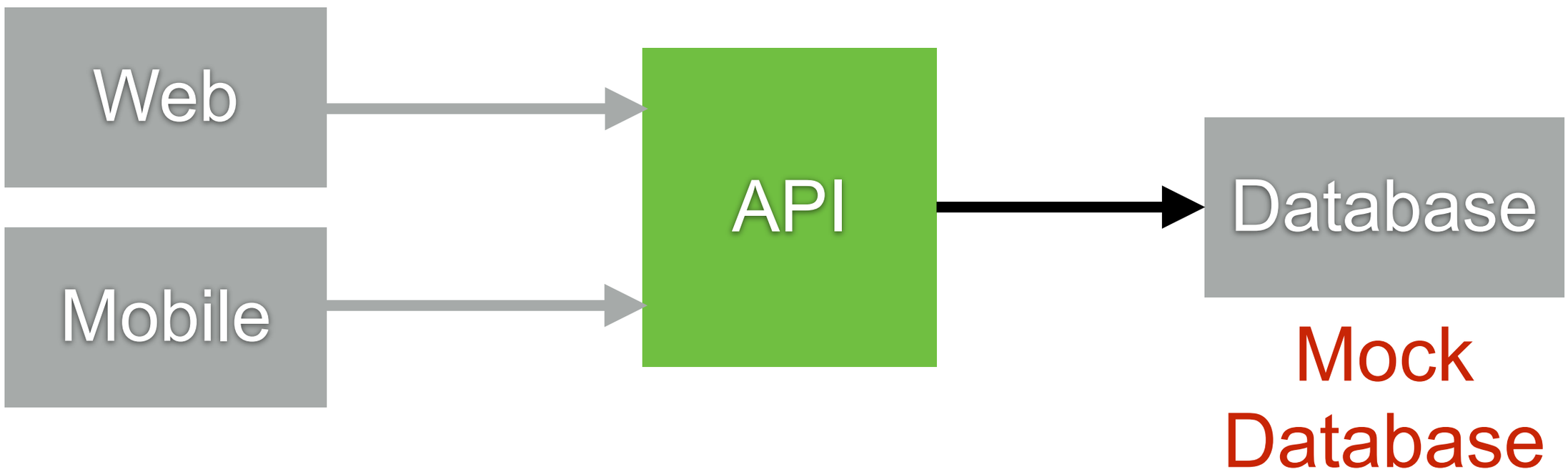
UI Test ?



API Testing ?



API Testing ?



Q/A