



# Postman in the Right Way





Somkiat Puisungnoen

Search

Somkiat | Home

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro

Software Craftsmanship

Software Practitioner at สยามชานาญกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata

When did you work at Opendream? X

... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro

Software Craftsmanship

Software Practitioner at สยามชานาญกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata



Facebook somkiat.cc

Somkiat Home | ? ▾

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

somkiat.cc  
@somkiat.cc

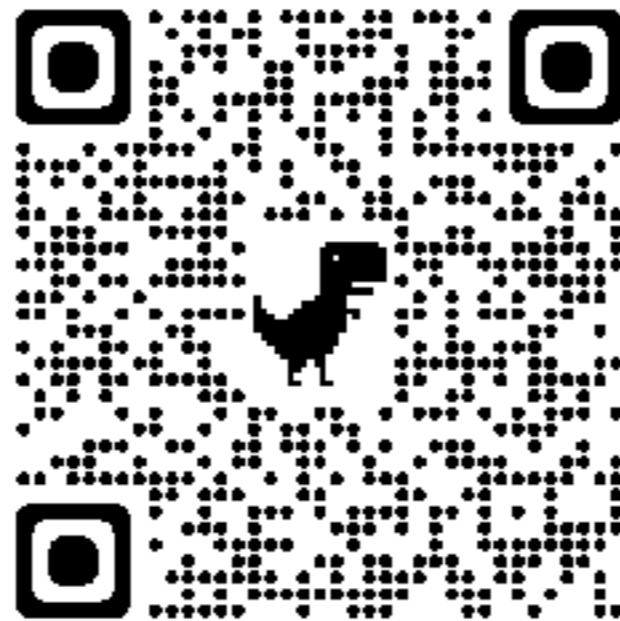
Home Posts Videos Photos

Liked Following Share ... + Add a Button

Help people take action on this Page. ×



# Postman in the right way



<https://github.com/up1/workshop-postman>



# Testing

Fast feedback

High quality



# Continuous Testing



# API Design

# API Testing

# Postman



# **Basic of Postman**



# Postman

## PUBLISH

Onboard developers to your API faster with Postman collections and documentation

## MONITOR

Create automated tests to monitor APIs for uptime, responsiveness, and correctness

## DOCUMENT

Create beautiful web-viewable documentation

## DESIGN & MOCK

Design in Postman & use Postman's mock service

## DEBUG

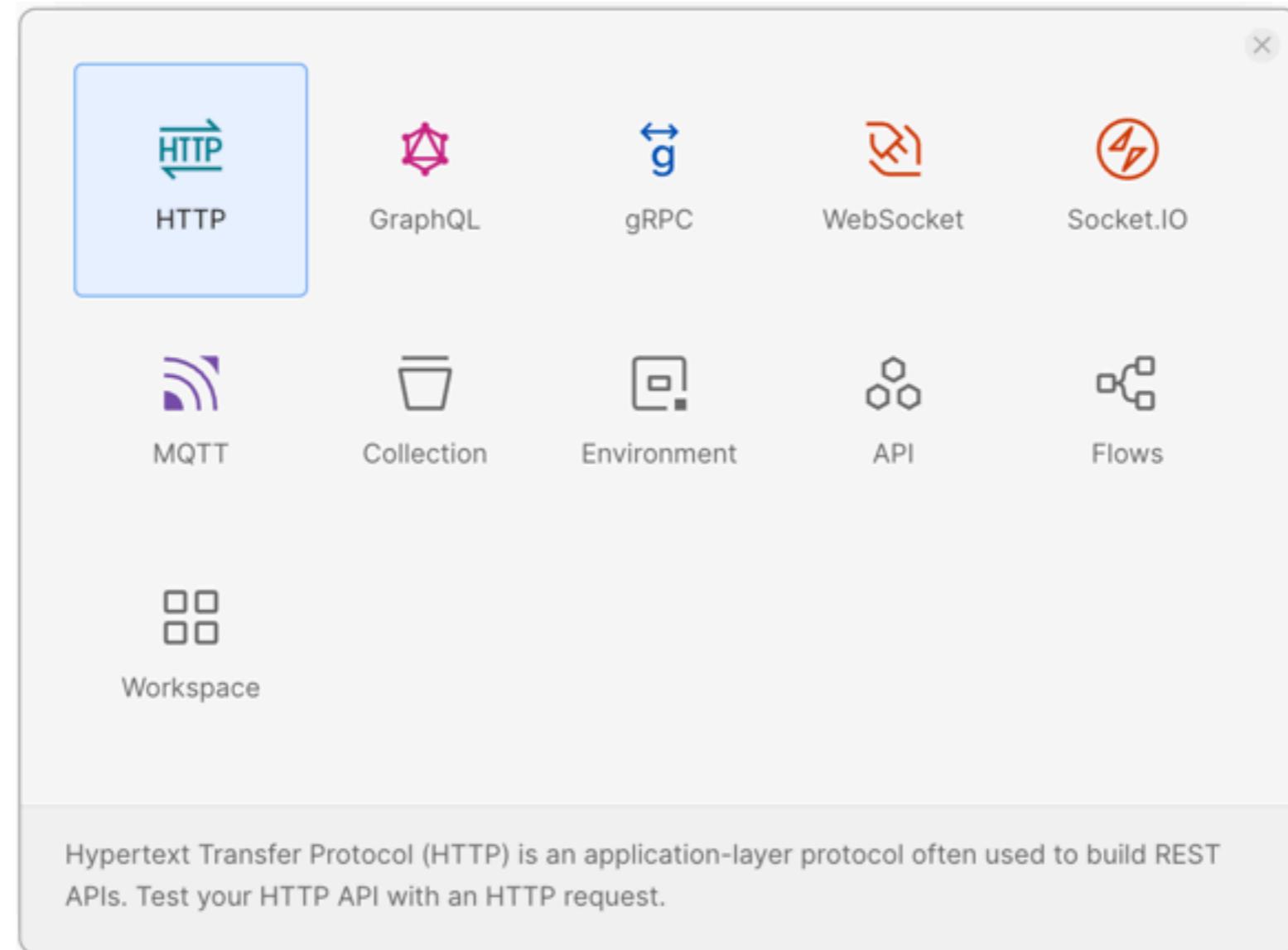
Test APIs, examine responses, add tests and scripts

## AUTOMATED TESTING

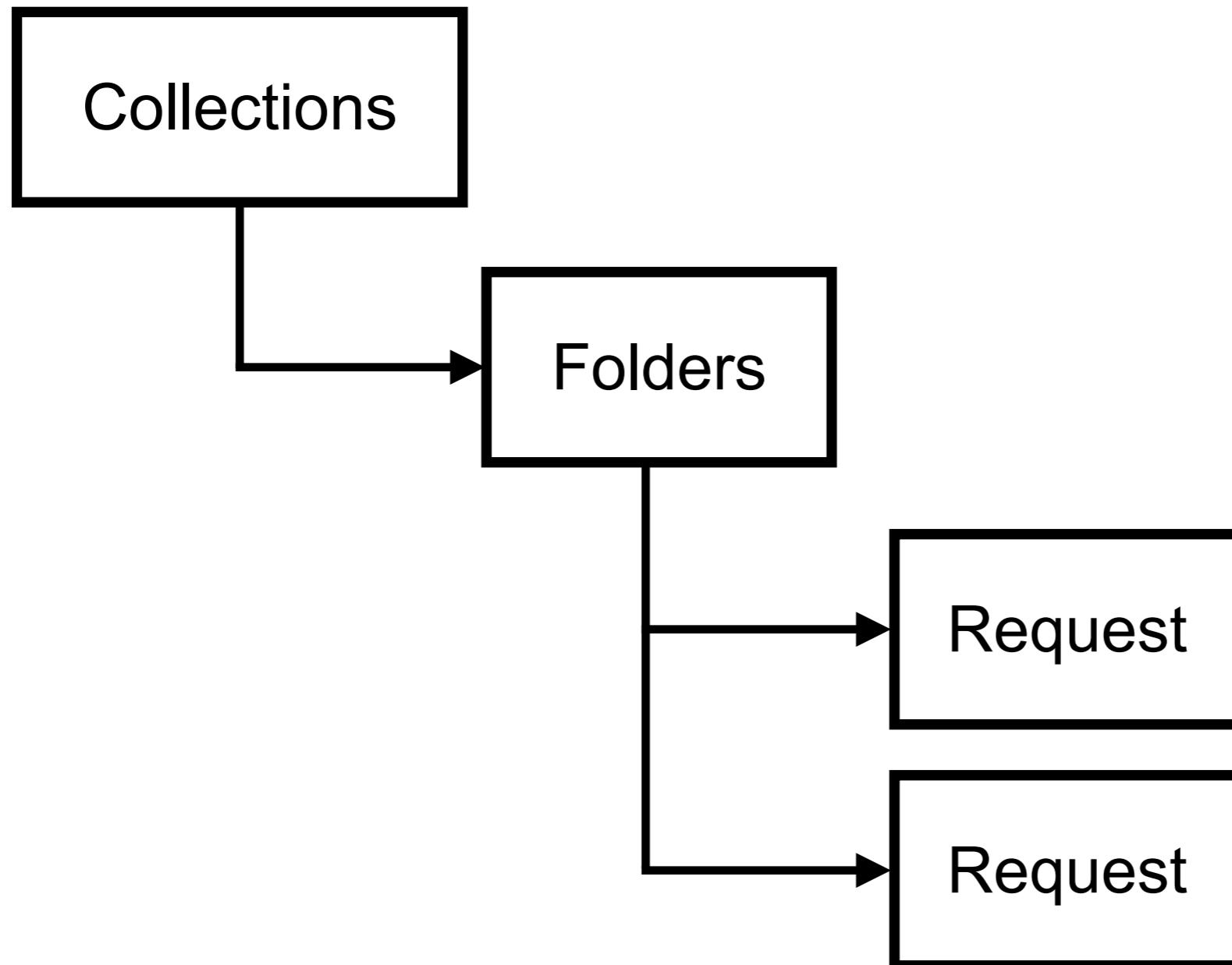
Run automated tests using the Postman collection runner



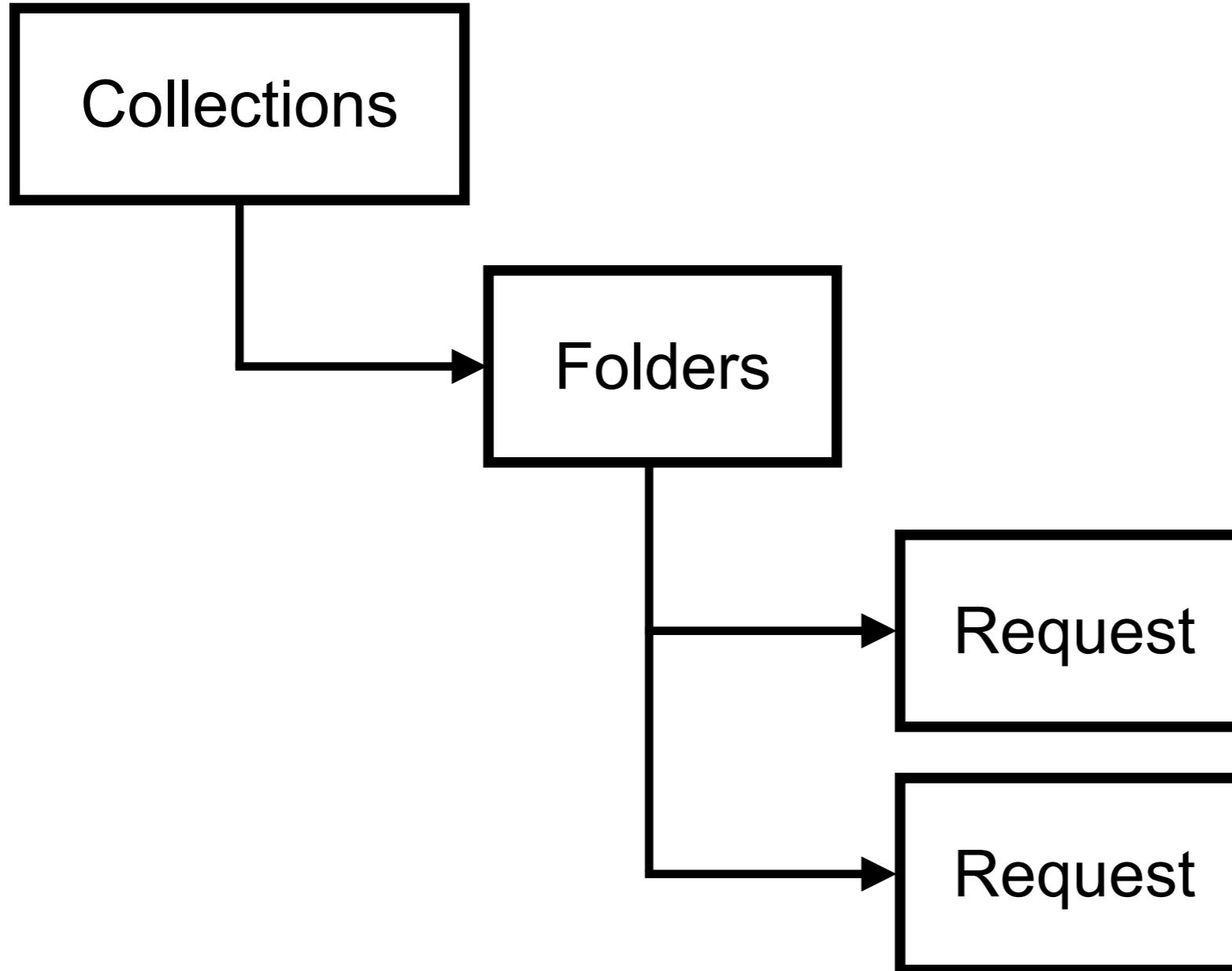
# Postman



# Postman structure



# Workspace

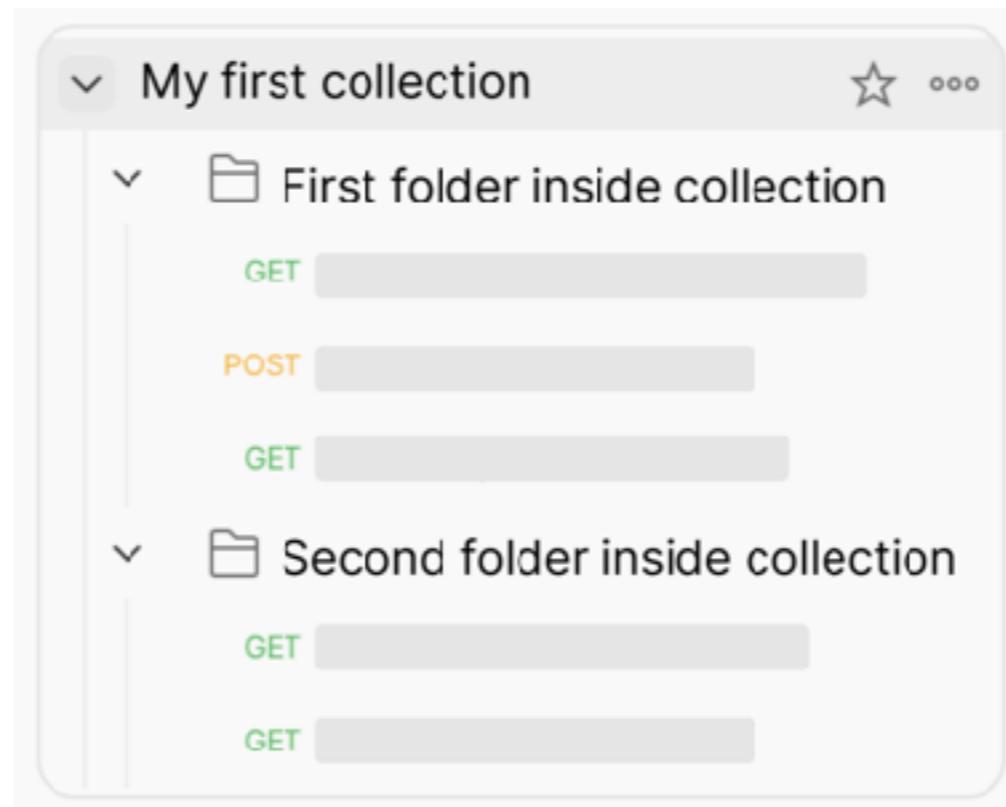


# Collections



# Postman collections

## Grouping requests



<https://www.postman.com/collection/>



# Collections

The screenshot shows a collection named "Demo01" in a testing application. At the top left is a folder icon labeled "Demo01". To its right is a "+" button and an ellipsis "...". Below the header, the collection name "Demo01" is displayed again, along with a "Share" button featuring a share icon. A navigation bar below the title includes tabs for "Overview" (which is underlined in red), "Authorization", "Pre-request Script", "Tests", "Variables", and "Runs". The main content area contains the title "Demo01" in large bold letters, followed by the subtext "Make things easier for your teammates with a complete collection".



# Collection Template !!

The screenshot shows the Postman Collection Template interface. On the left, there's a sidebar with a search bar and a tree view of categories:

- All templates:**
  - Roles
    - Backend Developers
    - Frontend Developers
    - Fullstack Developers
    - Quality Engineers
  - Use cases
    - API basics
    - Developer productivity
    - Infrastructure
    - Security
    - Testing
  - Industries
    - Artificial intelligence
    - Financial services
    - Healthcare
    - Human resources



# Folders



# Folders

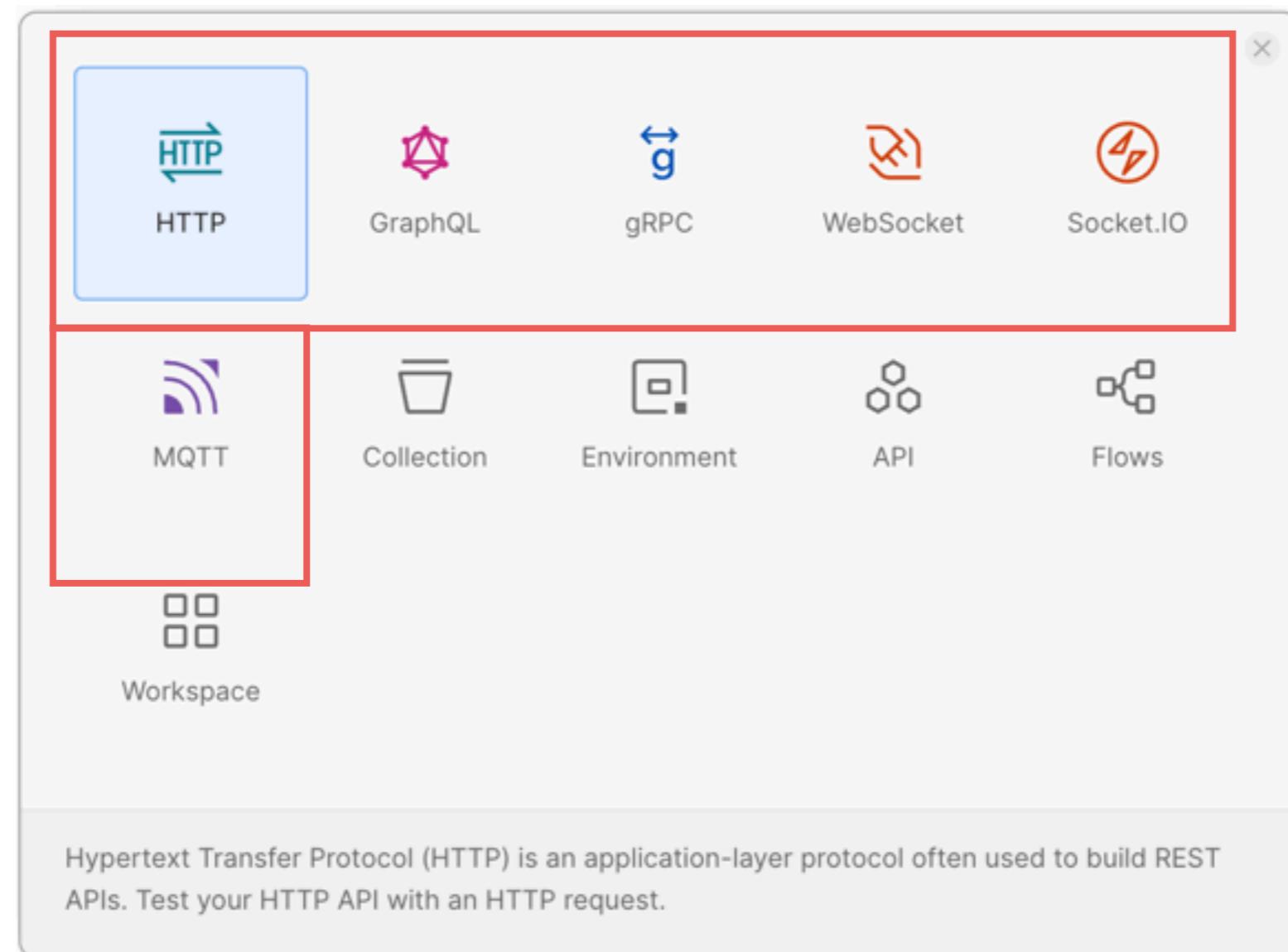
The screenshot shows the Postman interface for a folder named "New Folder". The top navigation bar includes "New Folder", a "+" button, and "No Environment". Below the bar, the path "Demo01 / New Folder" is shown, along with a "Run" button. A navigation menu at the top of the main area includes "Overview", "Authorization" (which is underlined, indicating it is selected), "Pre-request Script", and "Tests". A note below the menu states: "This authorization method will be used for every request in this folder. You can override this by specifying one in the request." A dropdown menu labeled "Type" shows "Inherit auth from parent". At the bottom of the main area, a message says: "This folder is using No Auth from collection Demo01."



# Requests



# Request



# HTTP Request

GET New Request + ... No Environment

HTTP Demo01 / New Request Save

GET Enter URL or paste text Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

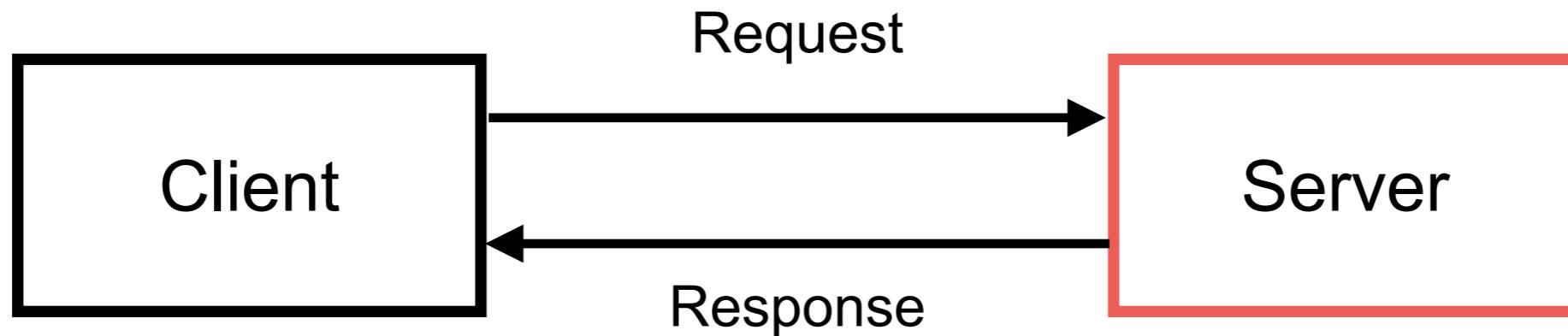
	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Response



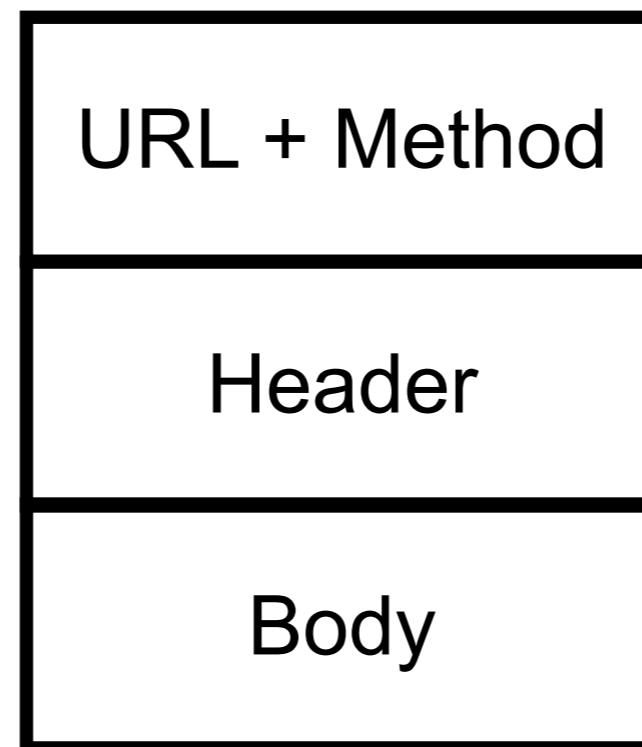
# HTTP (Hyper Text Transfer Protocol)

Protocol to exchange information over network



# HTTP Request

URL  
HTTP methods  
Request headers  
Request body



# HTTP Request

```
GET /api/data HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: application/json
Accept-Language: en-US,en;q=0.5
Authorization: Token abc123
Cache-Control: no-cache
Connection: keep-alive
Referer: https://www.google.com/
Pragma: no-cache
```



# URL in HTTP Request

URL (Uniform Resource Locator)

**URL**

`https://www.youtube.com/watch?v=9JunaO2hGK0`



# URL in HTTP Request

URL (Uniform Resource Locator)

**URL**

`https://www.youtube.com/watch?v=9JunaO2hGK0`

**URI**

`https://www.youtube.com/watch?v=9JunaO2hGK0`



# URL in HTTP Request

URL (Uniform Resource Locator)

**URL**

`https://www.youtube.com/watch?v=9JunaO2hGK0`

**URI**

`https://www.youtube.com/watch?v=9JunaO2hGK0`

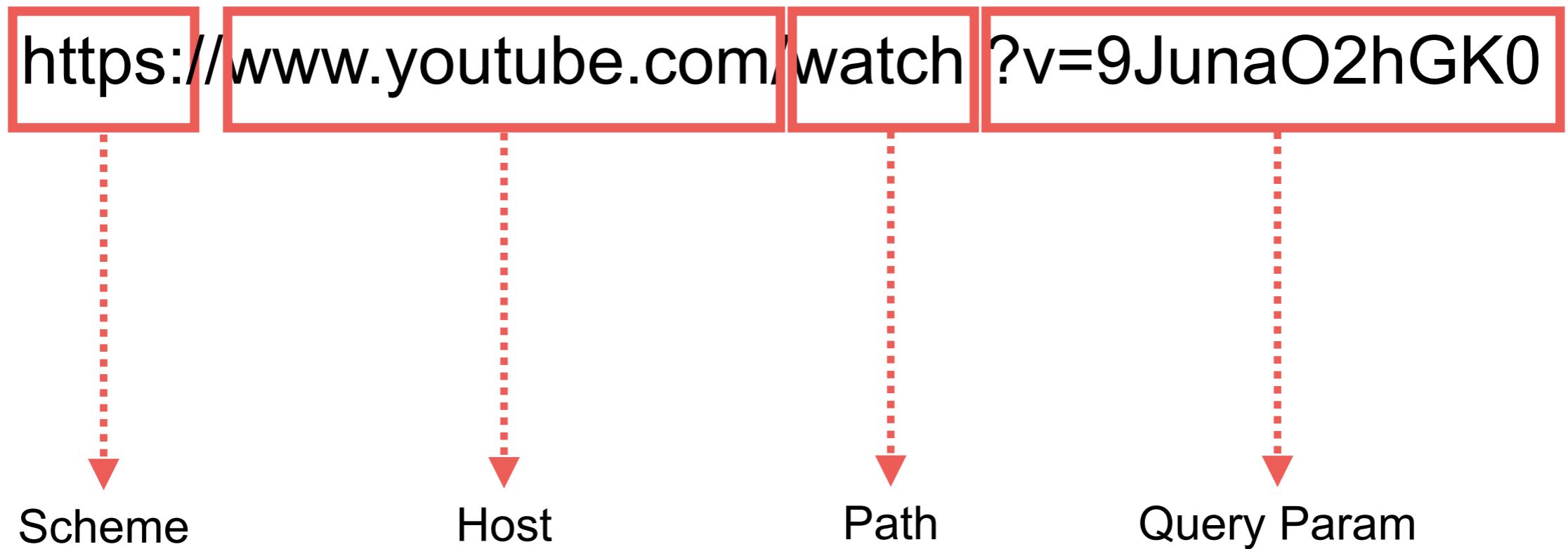
**URN**

`https://www.youtube.com/watch?v=9JunaO2hGK0`



# URL in HTTP Request

URL (Uniform Resource Locator)



# HTTP Request in Postman

GET New Request + ... No Environment

HTTP Demo01 / New Request Save

GET Enter URL or paste text Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Response



# Authorization in request ?

GET New Request + ... No Environment

HTTP Demo01 / New Request Save  

GET Enter URL or paste text Send

Params **Authorization** Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Response



# HTTP Methods

Method	Description
GET	Ask the server to retrieve a resource
POST	Ask the server to create a new resource
PUT	Ask the server to update an existing resource
DELETE	Ask the server to delete a resource
HEAD	Ask the server for status of a resource



# Request Body

Data sent from client to server  
Text/HTML/JSON/XML/Multipart

The screenshot shows the Postman interface with the following details:

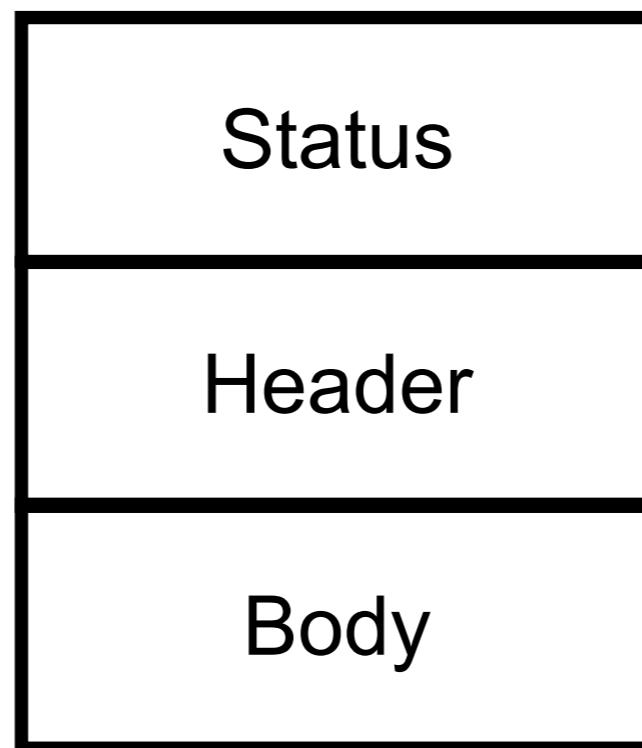
- HTTP Method: GET
- URL: https://postman-echo.com/get
- Body Type: x-www-form-urlencoded (selected)
- Table for body parameters:

Key	Value
Key	Value




# HTTP Response

Message that server sent back to a client



# HTTP Response

HTTP/1.1 200 OK

Date: Sun, 28 Mar 2023 10:15:00 GMT

Content-Type: application/json

Server: Apache/2.4.39 (Unix) OpenSSL/1.1.1c PHP/7.3.6

Content-Length: 1024

```
{  
    "name": "John Doe",  
    "email": "johndoe@example.com",  
    "age": 30,  
    "address": {  
        "street": "123 Main St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "12345"  
    }  
}
```



# Status Code

Code	Description
1xx	Information
2xx	Successful
3xx	Redirection
4xx	Client error
5xx	Server error

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>



# HTTP Response in Postman

The screenshot shows the Postman application interface. At the top, it displays the URL `https://postman-echo.com/get`. The "Headers" tab is selected, showing the following header configuration:

Key	Value	Description	... Bulk Edit
Key	Value	Description	

Below the request configuration, the response details are shown:

Status: 200 OK Time: 1261 ms Size: 836 B Save as Example

The response body is displayed in a code editor-like format with line numbers:

```
1  [
2    "args": {},
3    "headers": {
4      "x-forwarded-proto": "https",
5      "x-forwarded-port": "443",
6      "host": "postman-echo.com",
7      "x-amzn-trace-id": "Root=1-650bf11-25e76d7c697172964a55bbf3",
8      "user-agent": "PostmanRuntime/7.33.0",
9      "accept": "*/*",
10     "postman-token": "8344a821-1036-42d8-a8eb-ed1a71e5987c",
11     "accept-encoding": "gzip, deflate, br",
12     "cookie": "sails.sid=s%3AQjMOKt2Huqtfo20FxcN6pfHab5K1XWq3..vejTnKp9pK068Pe%2BAnHUXe3cb1ThCeIgTAU7Xb%2FcRjk"
13   },
14   "url": "https://postman-echo.com/get"
15 ]
```



# Test in Postman

Pre-request  
script

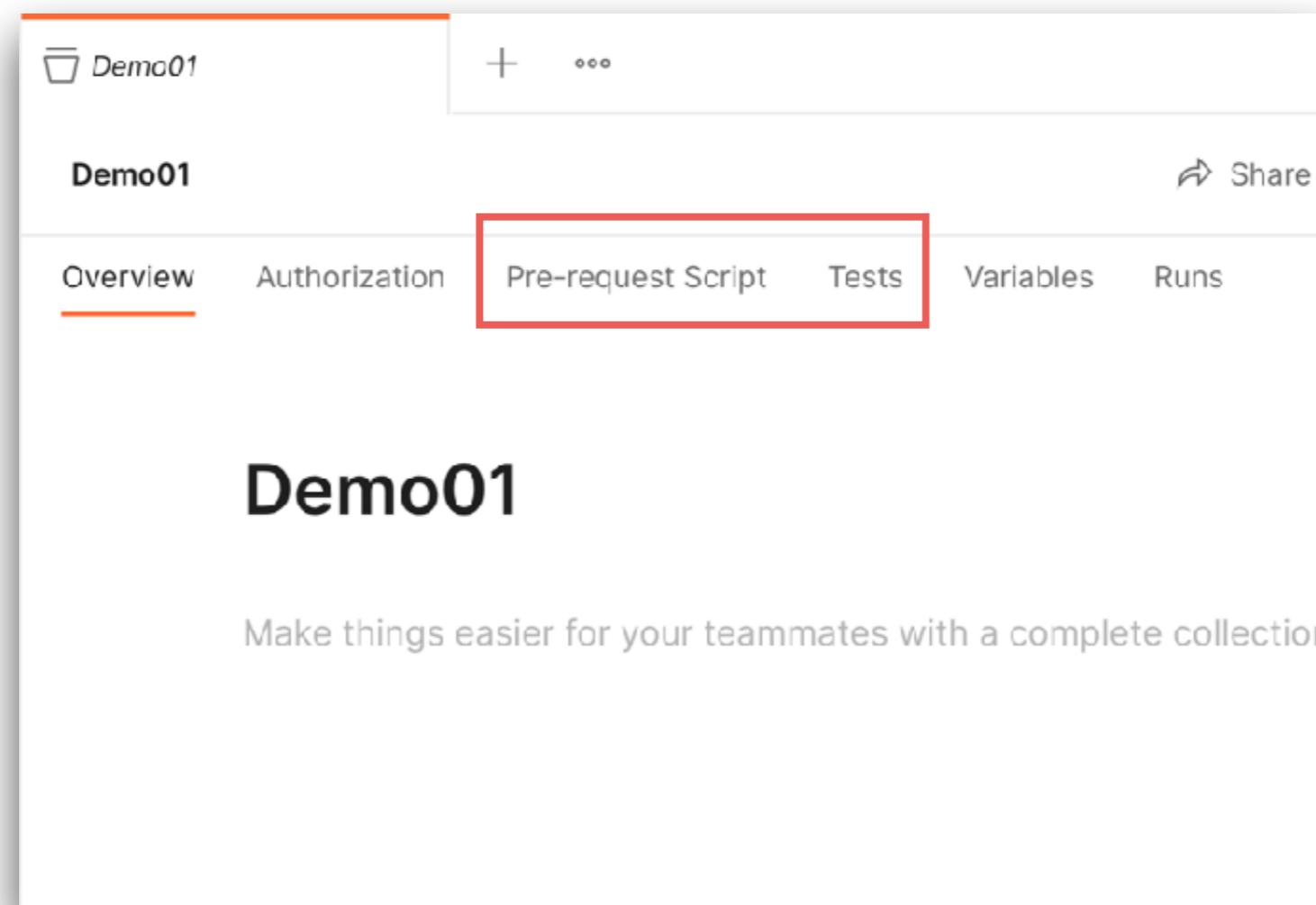
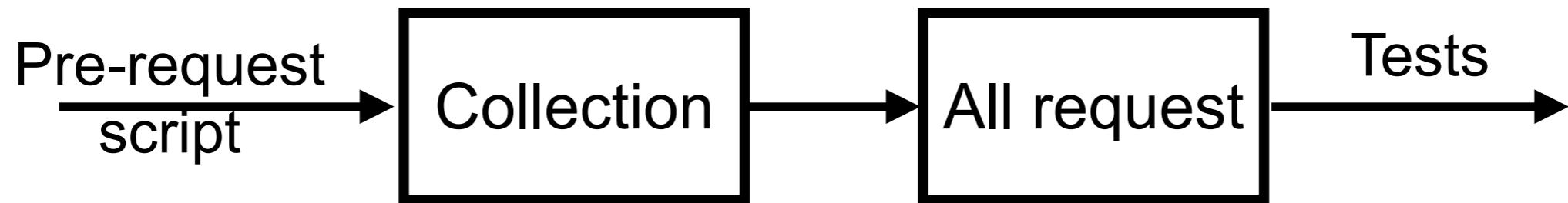
Test response

<https://learning.postman.com/docs/writing-scripts/pre-request-scripts/>

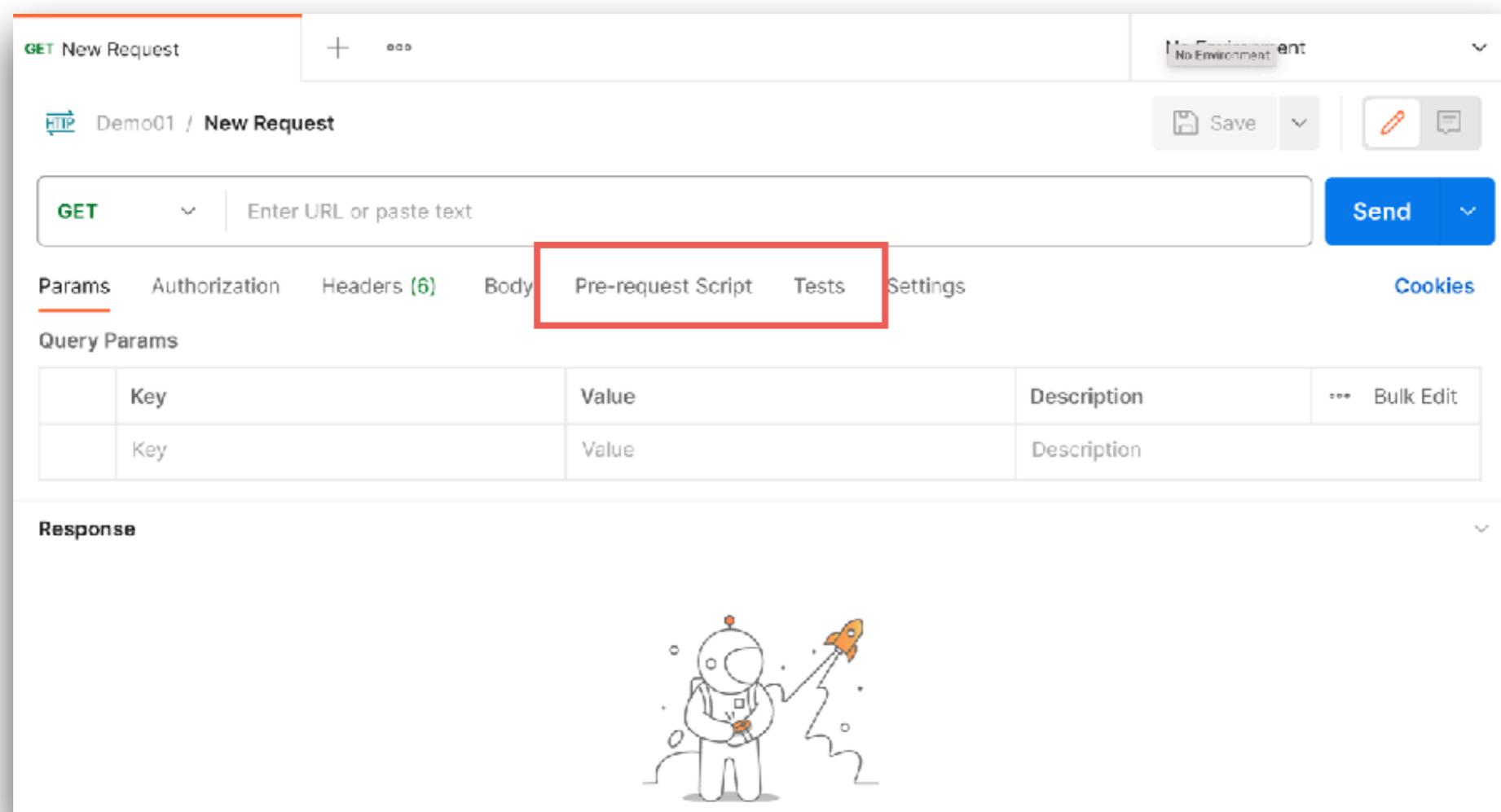
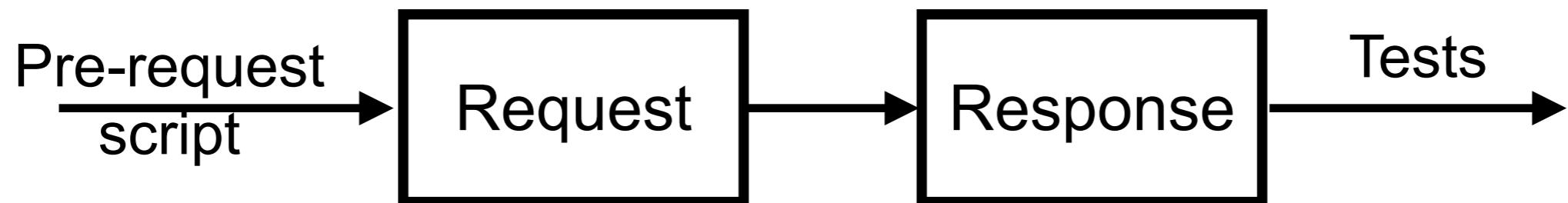
<https://learning.postman.com/docs/writing-scripts/test-scripts/>



# Test in collection



# Test in request



# What to test ?

HTTP/1.1 200 OK

Date: Sun, 28 Mar 2023 10:15:00 GMT

Content-Type: application/json

Server: Apache/2.4.39 (Unix) OpenSSL/1.1.1c PHP/7.3.6

Content-Length: 1024

```
{  
    "name": "John Doe",  
    "email": "johndoe@example.com",  
    "age": 30,  
    "address": {  
        "street": "123 Main St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "12345"  
    }  
}
```



# Write test with Postbot

<https://learning.postman.com/docs/getting-started/basics/about-postbot/>



# Skip test !!

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

The screenshot shows the 'Test Results' tab in Postman with a total of 5/6 results. The results are as follows:

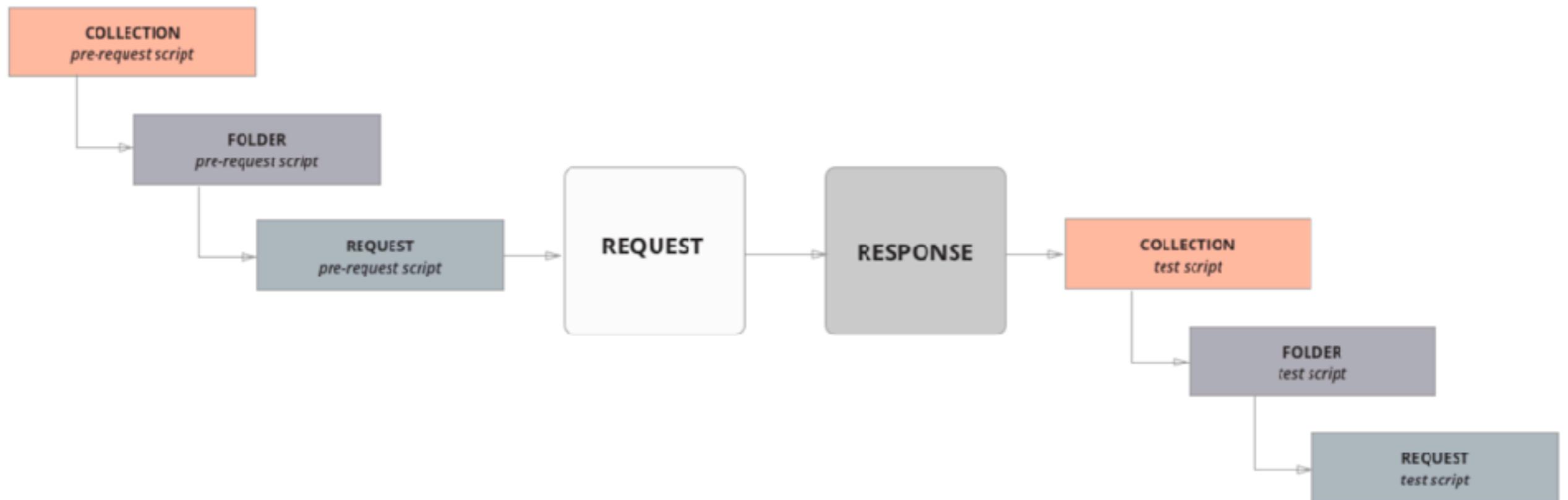
Status	Detail
SKIPPED	Status code is 200
PASS	Response has the required fields
PASS	Headers contain the expected keys
PASS	URL is not empty
FAIL	Response time is within an acceptable range



# Test Execution



# Test Execution

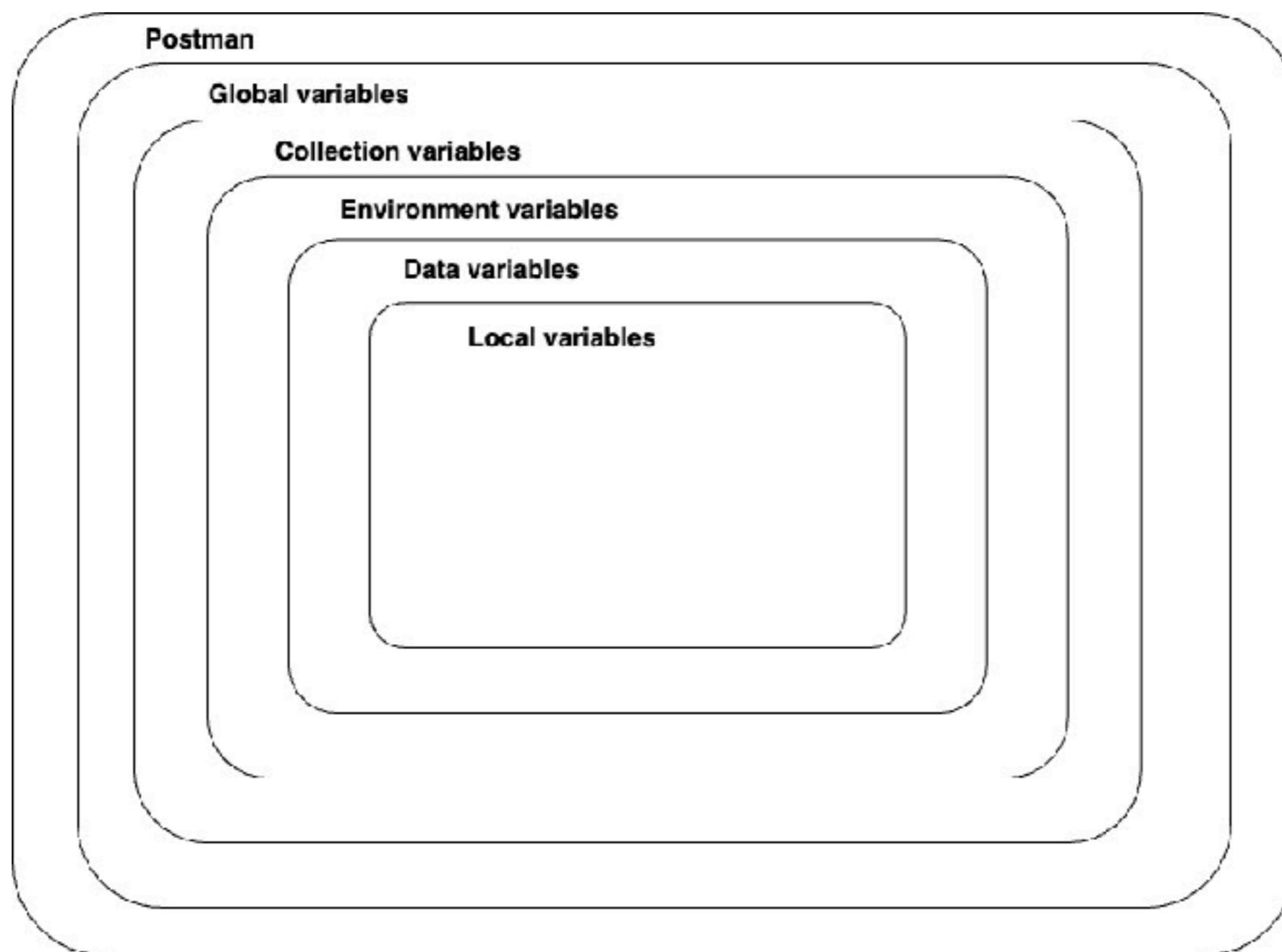


# Variables

<https://learning.postman.com/docs/sending-requests/variables/>



# Variables Scopes



<https://learning.postman.com/docs/sending-requests/variables/#variable-scopes>



# Variables

Global variables  
Collection variables  
Local variables

**{{VARIABLE\_NAME}}**



# Variable Types

Plain text  
Secret for sensitive data

**Globals**

Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be viewed and edited by anyone in that workspace. [Learn more about globals](#) ↗

Filter variables

	Variable	Type	Initial value	Current value
<input checked="" type="checkbox"/>	USERNAME	default ▾	somkiat	somkiat
<input checked="" type="checkbox"/>	PASSWORD	secret ▾	.....	.....

Add new variable



# Environment to test

Local

Dev

Test

Staging

Prod



# Environment in Postman

**Environment** [Add](#)

**No active Environment**

An environment is a set of variables that allow you to switch the context of your requests.

**Globals** [Add](#)

**No global variables**

Global variables are a set of variables that are always available in a workspace.

**(i) Use variables to reuse values and protect sensitive data** [X](#)

Store sensitive data in variable type secret to keep its values masked on the screen. Learn more about [variable type ↗](#)

Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about [variable values ↗](#)



# Working with more data

## Data-Driven Testing

Duplicate  
requests

Use external  
data files

Use pre-  
request script

CSV, JSON



# Working with CSV/JSON

The screenshot shows the Postman Collection Runner interface. On the left, there's a sidebar with 'Run order' and buttons for 'Deselect All', 'Select All', and 'Reset'. Below that, a checkbox for 'GET New Request' is checked. On the right, there are two tabs: 'Functional' (which is selected) and 'Performance'. The main area is titled 'Choose how to run your collection' with three options: 'Run manually' (selected), 'Schedule runs', and 'Automate runs via CLI'. A red box highlights the 'Run configuration' section, which includes fields for 'Iterations' (set to 1), 'Delay' (set to 0 ms), 'Data' (with a 'Select File' button), and a checkbox for 'Persist responses for a session'. There's also a link to 'Advanced settings'. At the bottom is an orange 'Run Demo01' button.

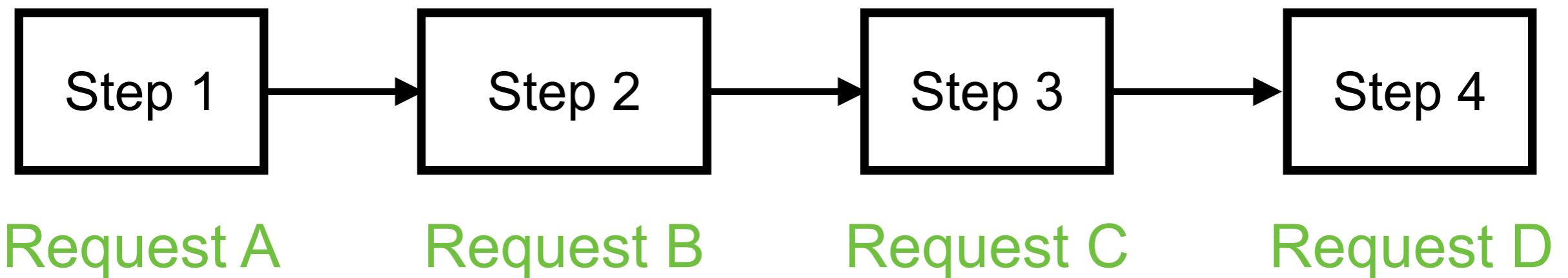
<https://learning.postman.com/docs/collections/running-collections/working-with-data-files/>



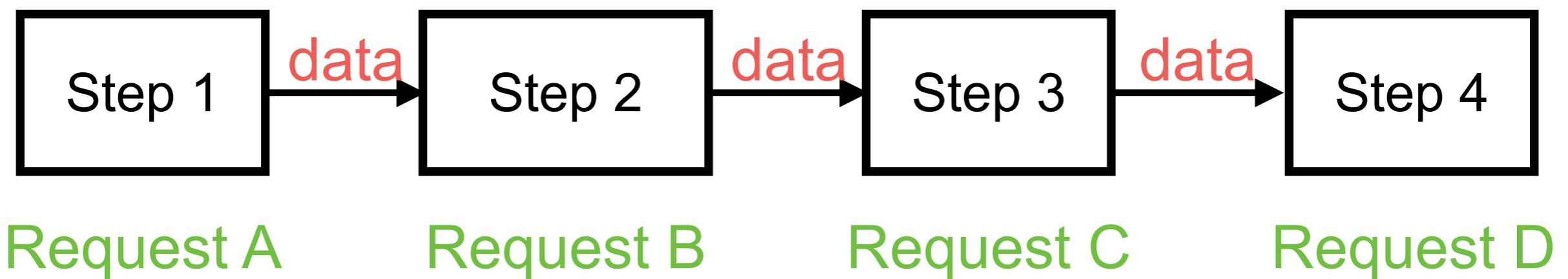
# Workflow in postman



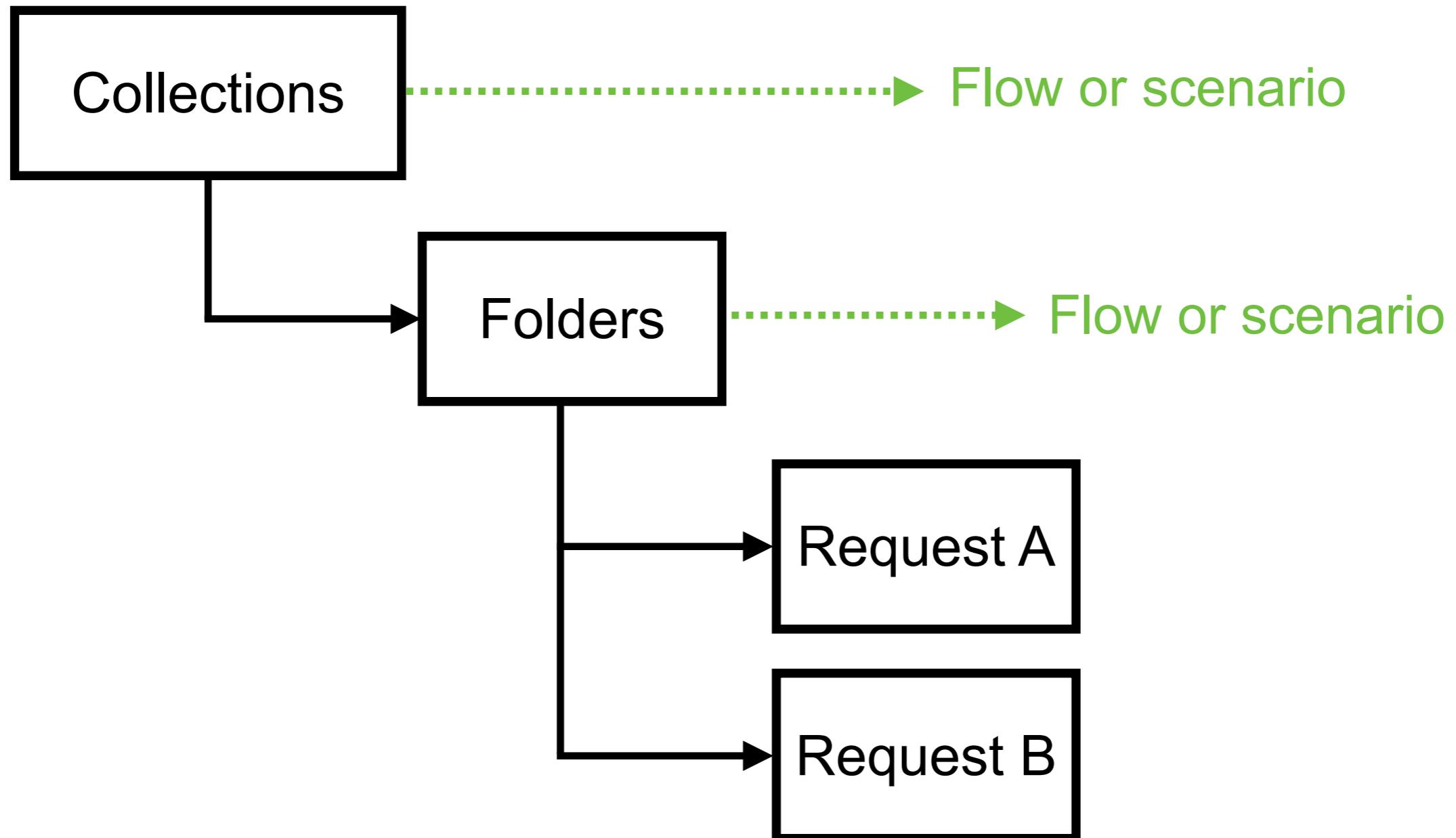
# Workflow



# Workflow in postman ?



# Postman structure



# Workshop



Fake store rest API for your e-commerce or shopping website prototype.

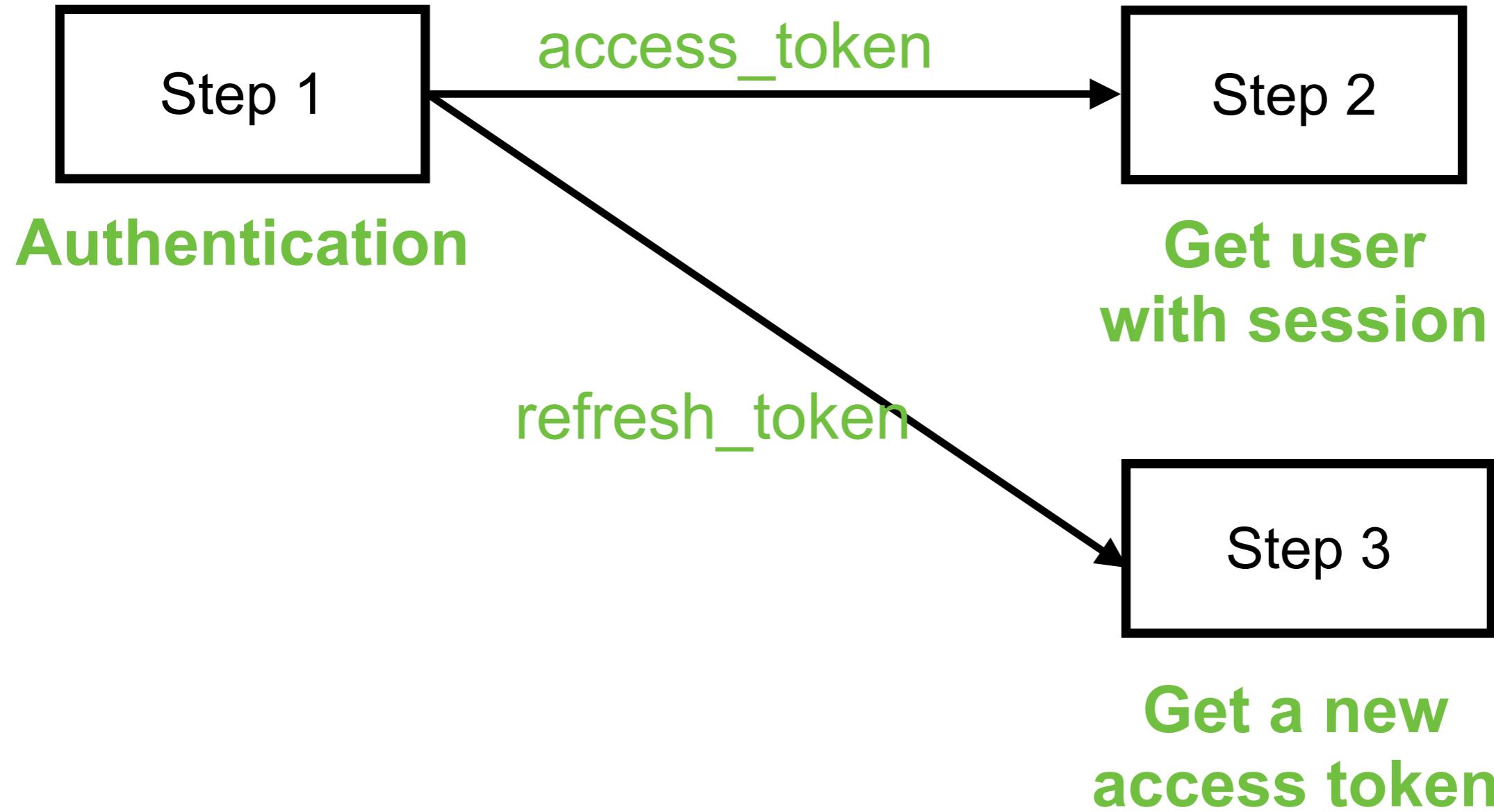
[VIEW DOCS](#)



<https://fakeapi.platzi.com/>



# Workflow



# Solutions

*Hard code !!*

Variables and Test scripts  
Call javascript function



# Exposed Tokens in Postman

[Action Required] Exposed Tokens In Postman Collection [Inbox](#)

The Postman Team <security.notifications@getpostman.com> 11:06 PM (21 minutes ago) [star](#)

To me -

 POSTMAN

Hi somkiat.puisungnoen,

Postman Sniffer Scanner found that some tokens may have been publicly exposed in the Flow collection in the testing-day workspace(s).

**What tokens may have been exposed?**

Token name	Token value	Location
Bearer token	Bearer eyJhbGciOiJI... ZXhwIjoxNjk3MTI2NzUDQ.1A.svCEkpwGumsdUyleWwrgSGUzhd d2XGBXBla5Dk	<a href="#">Get user with session!</a> <a href="#">Headers</a>

**What impact may this have on my account?**

Anyone who views the public collection will be able to see the token and potentially use it to access the APIs or services. Secure your account by taking the following steps:

**What steps can I take for my account's security right now?**

1. **Revoke the existing tokens** – even if it was only published for a short time, your token is compromised and could leave your data vulnerable.
2. **Replace tokens with placeholder values** – use a dummy token or placeholder text, like your-nasa-key, to indicate what value to use.

**What steps should I take for my account's security for the future?**

To prevent accidental breaches in the future, check out our guide on how to [keep your private data secure](#).



# Use Variables in test script

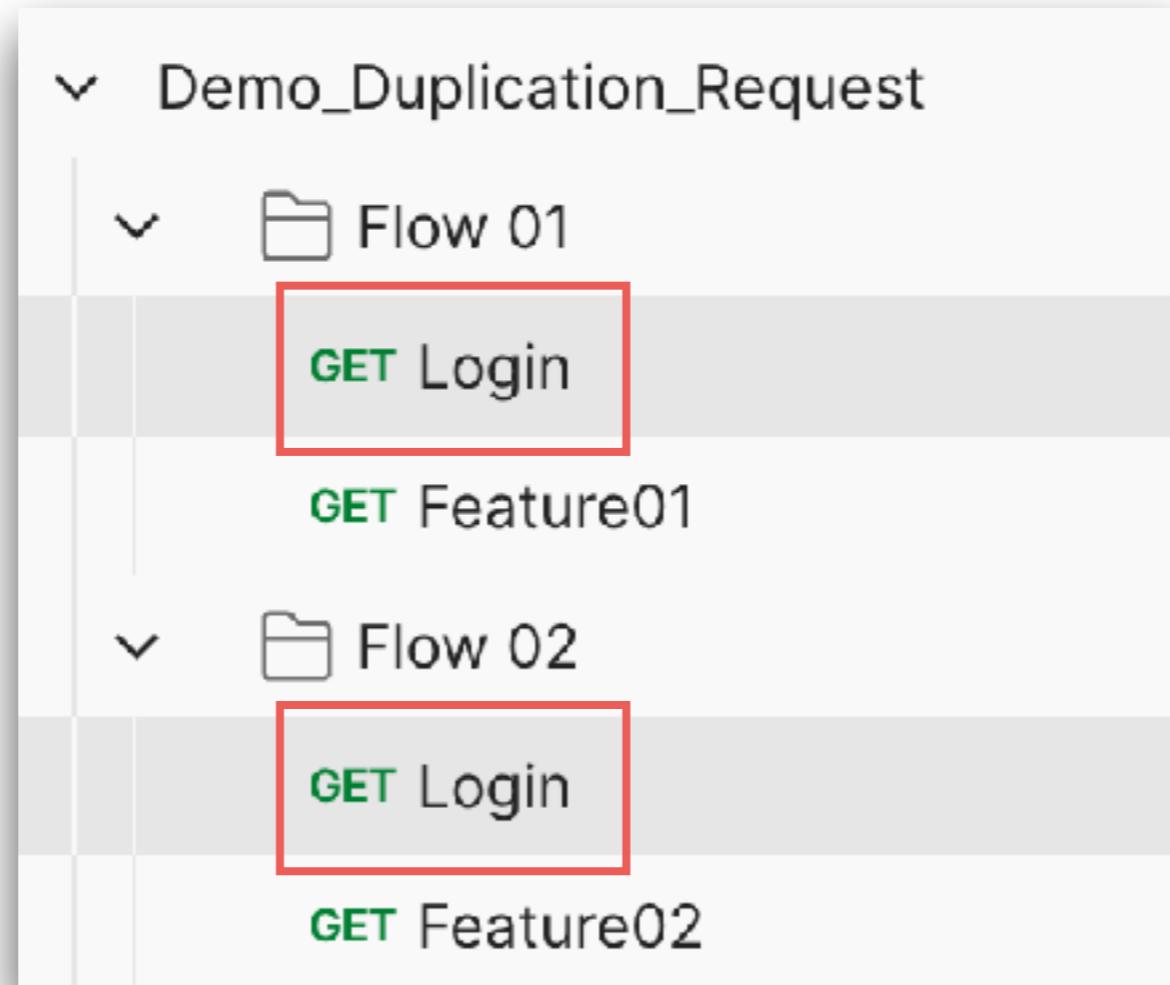
## Collection variables

```
var jsonData = pm.response.json();
var accessToken = jsonData.access_token;
var refreshToken = jsonData.refresh_token;

pm.collectionVariables.set("accessToken", accessToken);
pm.collectionVariables.set("refreshToken", refreshToken);
```



# Duplication Requests !!



<https://learning.postman.com/docs/writing-scripts/script-references/postman-sandbox-api-reference/#scripting-with-request-data>



# Write code

```
const postRequest = {
  url: 'https://postman-echo.com/post',
  method: 'POST',
  header: {
    'Content-Type': 'application/json',
    'X-Foo': 'bar'
  },
  body: {
    mode: 'raw',
    raw: JSON.stringify({ key: 'this is json' })
  }
};
pm.sendRequest(postRequest, (error, response) => {
  console.log(error ? error : response.json());
});
```

<https://github.com/up1/workshop-postman/wiki/Workshop-Code-Snippets>



# Create JS Function

```
var getToken = () => {
  const postRequest = {
    url: 'https://api.escuelajs.co/api/v1/auth/login',
    method: 'POST',
    header: {
      'Content-Type': 'application/json'
    },
    body: {
      mode: 'raw',
      raw: JSON.stringify({
        "email": "john@mail.com",
        "password": "changeme"
      })
    }
  }
  pm.sendRequest(postRequest, (error, response) => {
    var response = response.json();
    pm.collectionVariables.set("accessTokenV3", response.access_token);
    pm.collectionVariables.set("refreshTokenV3", response.refresh_token);
  });
}

pm.globals.set("getToken", getToken.toString());
```

<https://github.com/up1/workshop-postman/wiki/Workshop-Code-Snippets>



# Eval function

```
eval(pm.globals.get("getToken"))();
```

<https://github.com/up1/workshop-postman/wiki/Workshop-Code-Snippets>



# Call request in collection

The screenshot shows the Postman interface with a collection named "Postman Echo GET". The "Tests" tab is selected, displaying a single line of JavaScript code: "1 postman.setNextRequest("request\_name");". A tooltip for "Script with Postbot" is visible, stating "Write scripts easily with AI". The "Tests" tab is highlighted with an orange underline.

<https://learning.postman.com/docs/collections/running-collections/building-workflows/>



# Terminate execution !!

*postman.setNextRequest(null);*

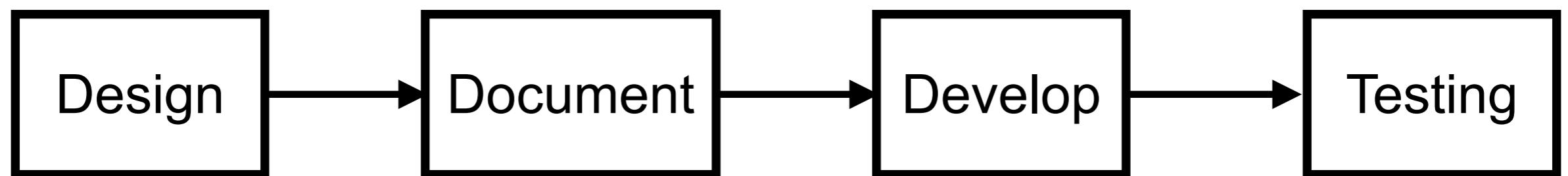
<https://learning.postman.com/docs/collections/running-collections/building-workflows/>



# Software Delivery Process



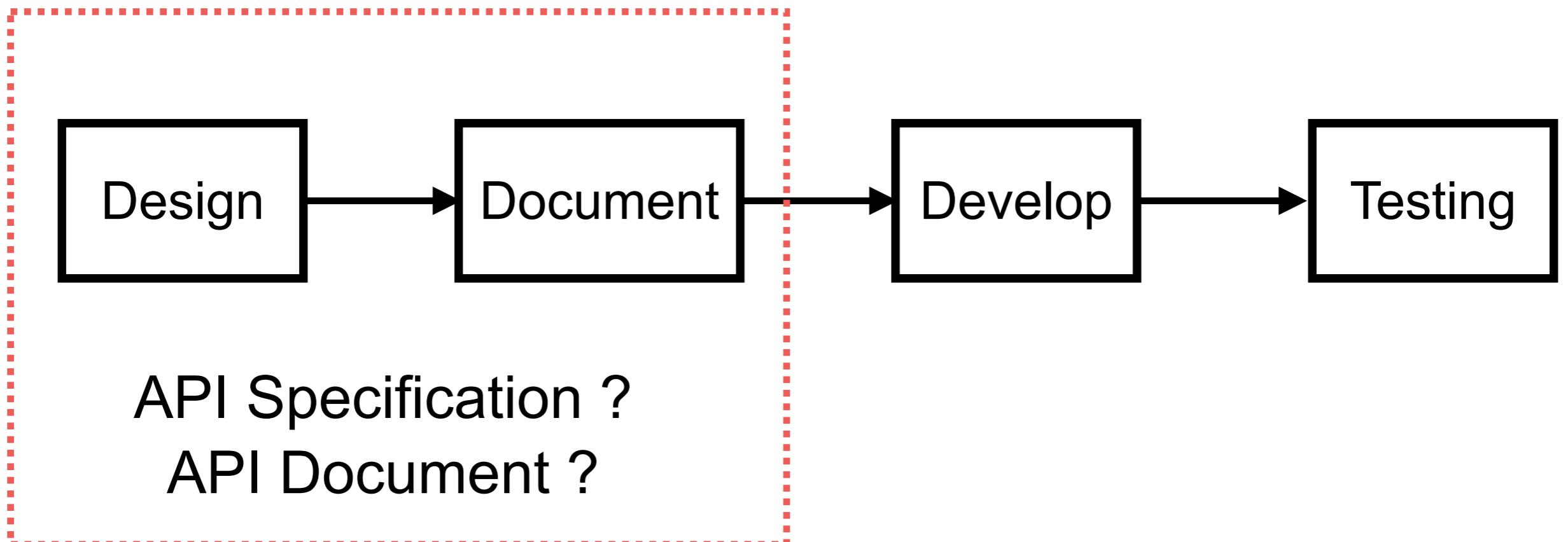
# Software Delivery Process



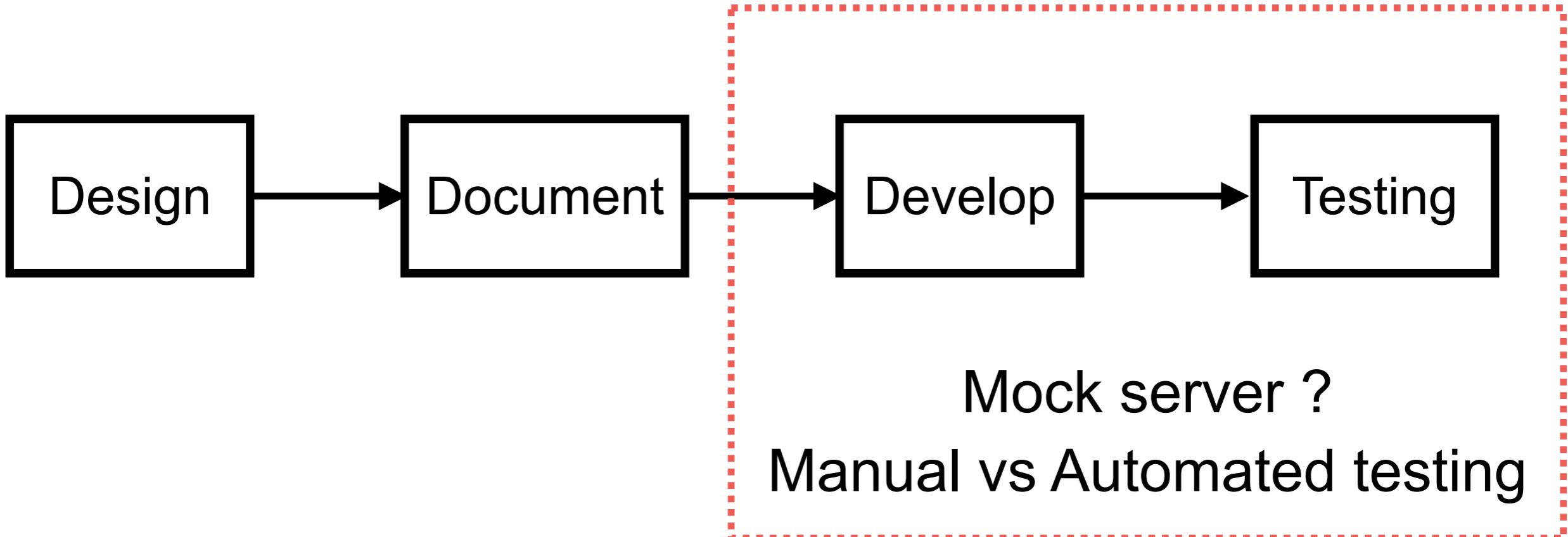
# API Development



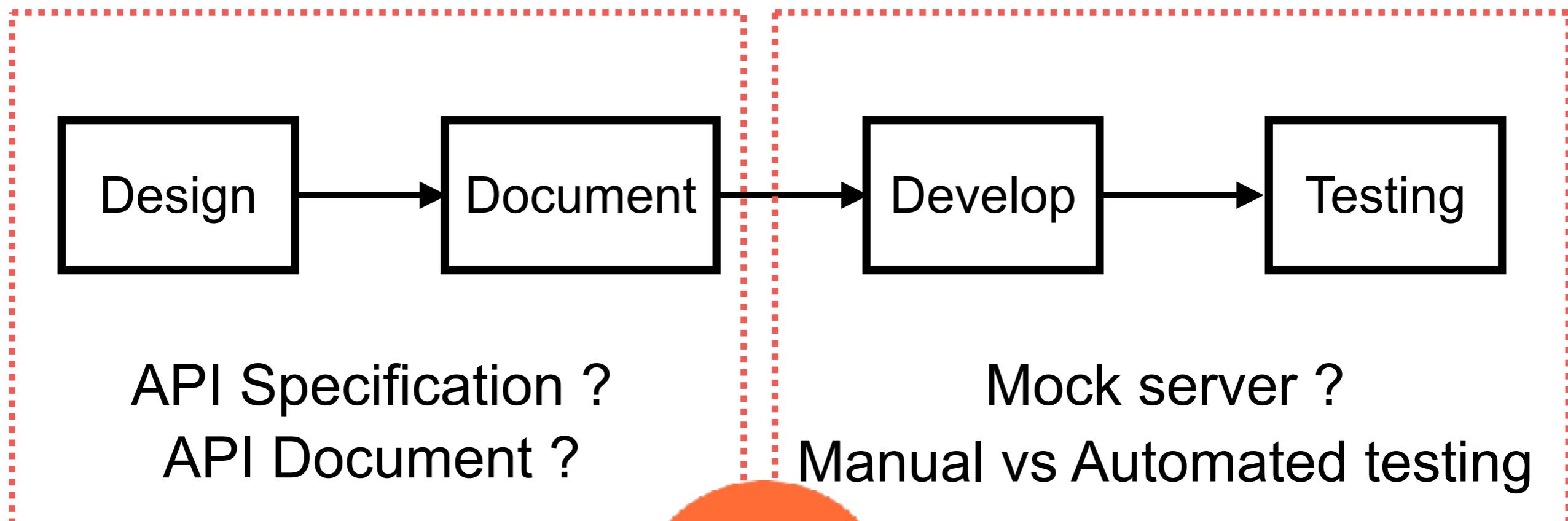
# APIs



# APIs



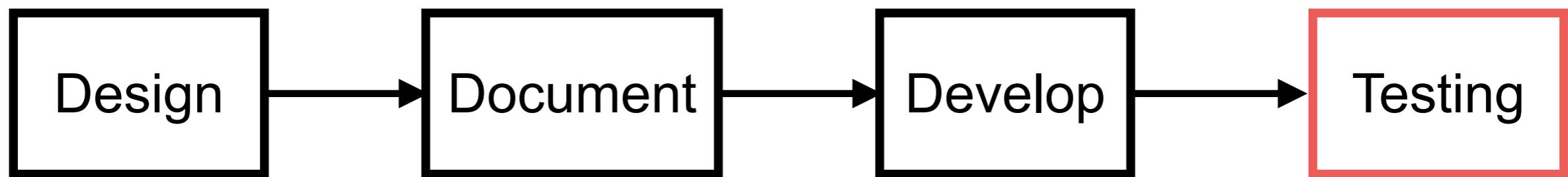
# Working with Postman



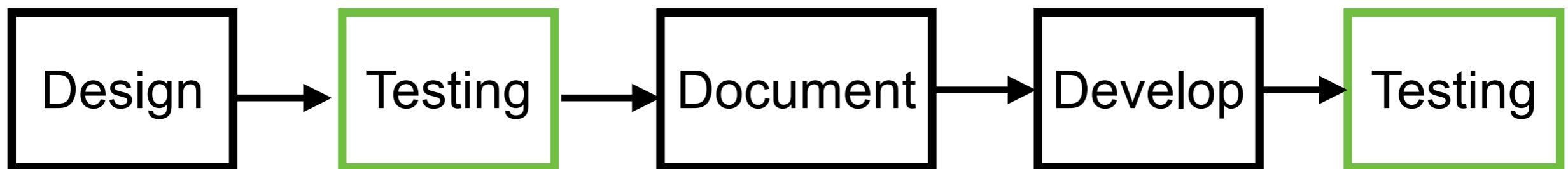
POSTMAN



# ทำก่อนคิด (ทดสอบอย่างไร)



# គិតកំណាំ

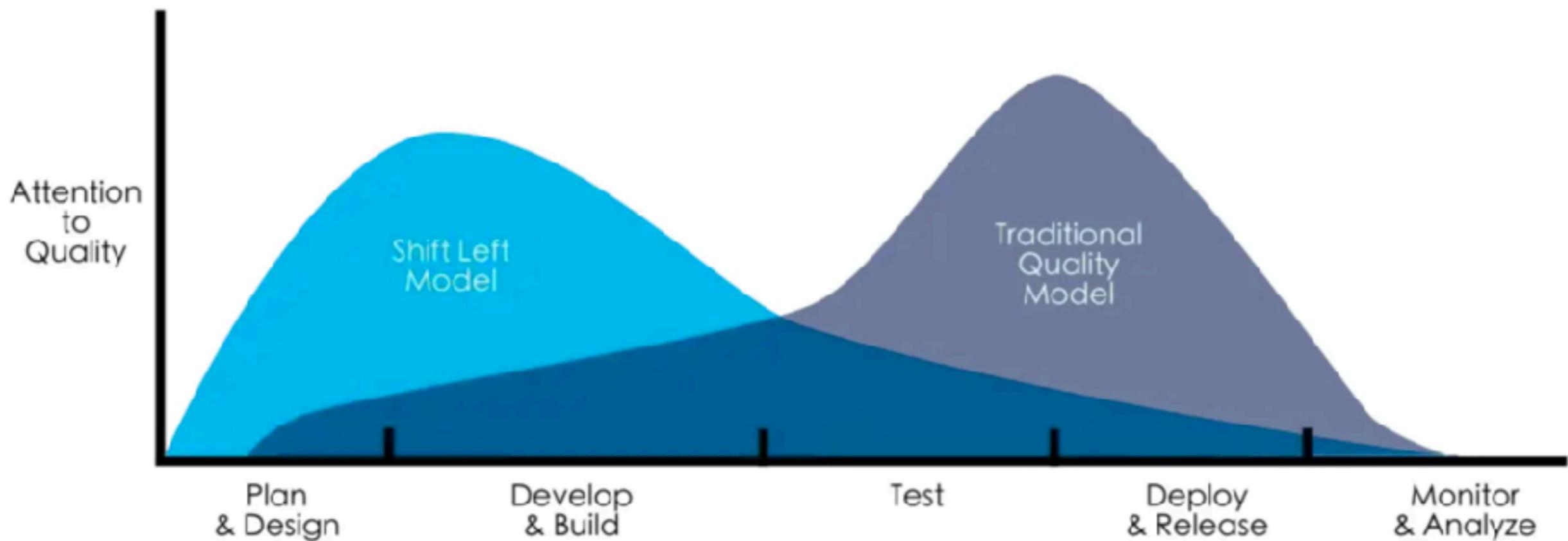


Why and What to test ?

Validation  
Verification



# គិតកន្លែង



# High Quality



Why and What to test ?

Validation  
Verification



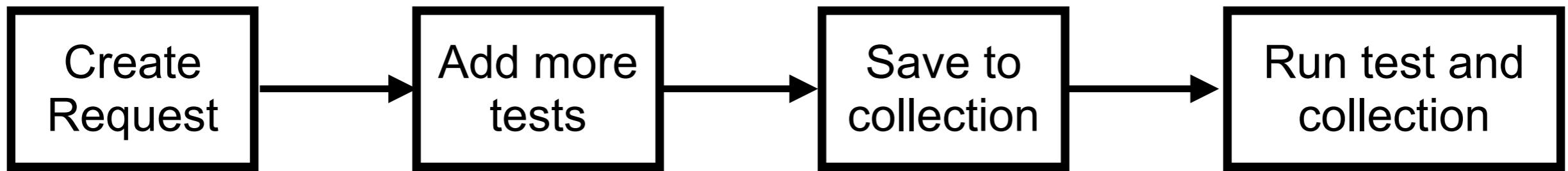
POSTMAN



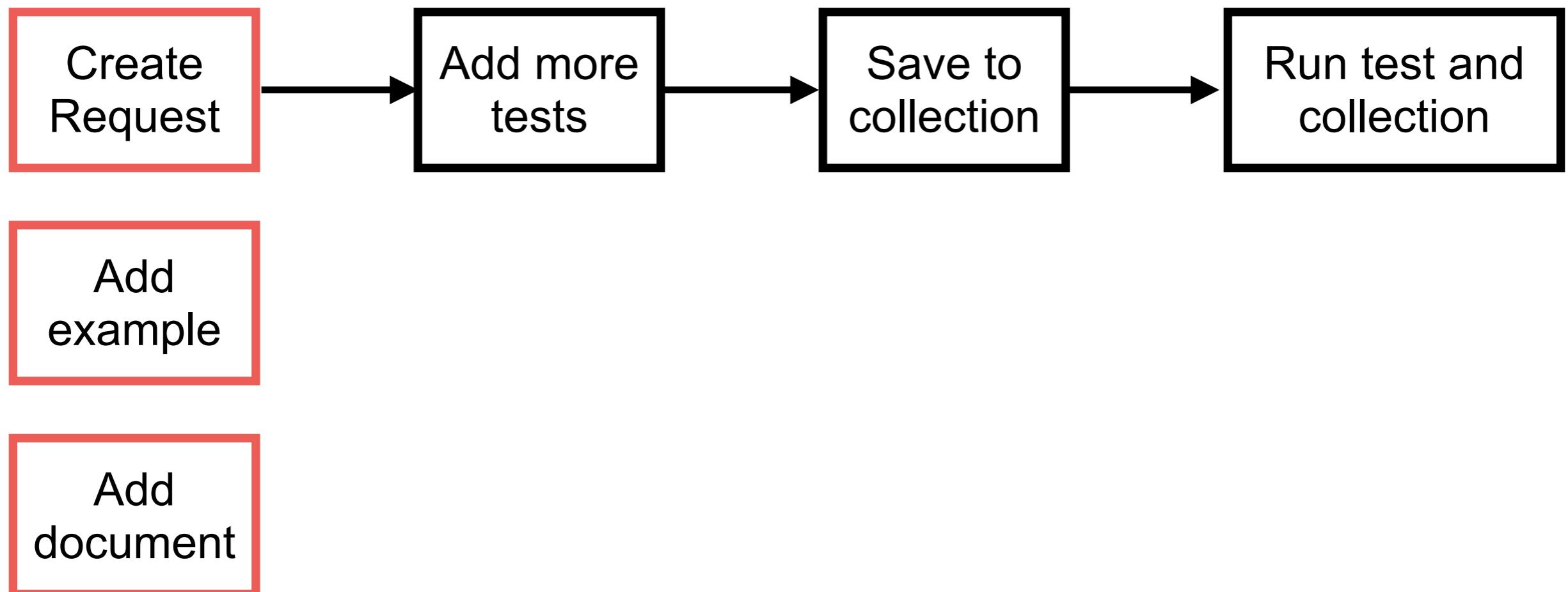
# Workshop



# Simple Process



# Design first with Postman

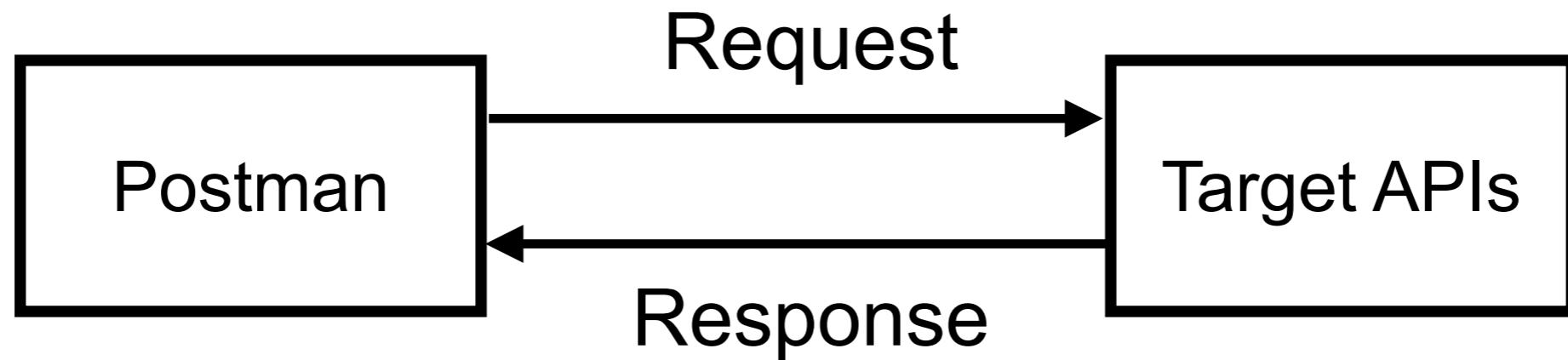


# Add more tests !!



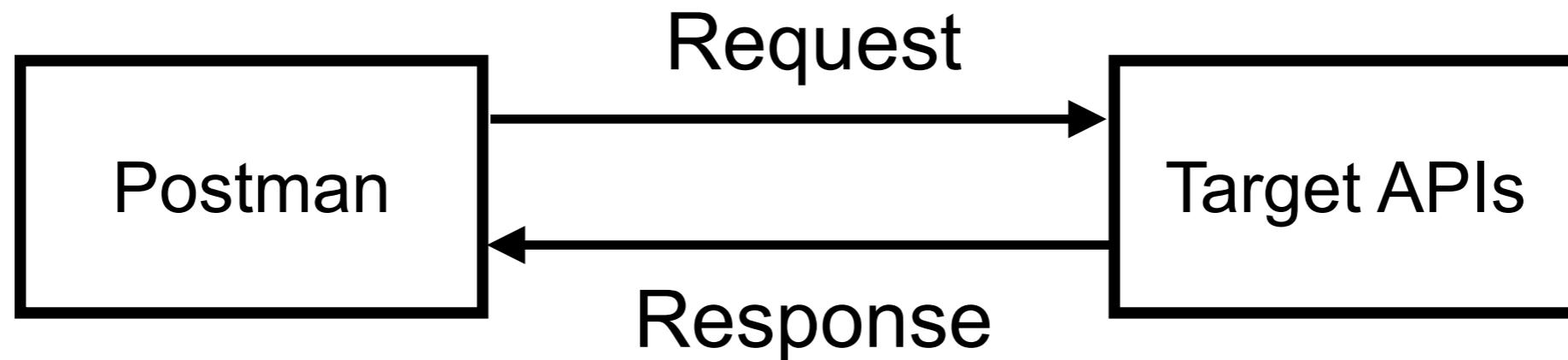
# Why and What to test ?

Thinking about testing (API)



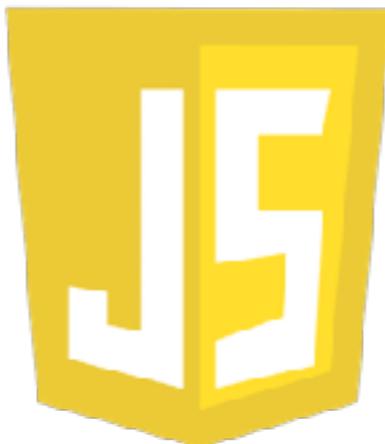
# Why and What to test ?

Thinking about testing (API)



# Add test cases in request and collections

JavaScript



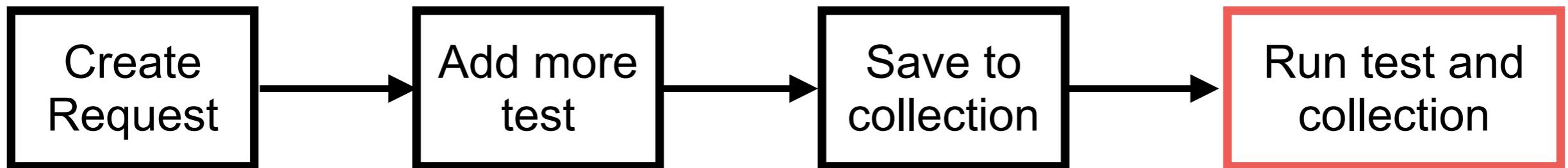
<https://learning.postman.com/docs/writing-scripts/test-scripts/>



# Automation process with Postman



# Testing process ?



# Testing with Postman

Collection runner  
Postman CLI  
**Newman CLI**



# Newman CLI

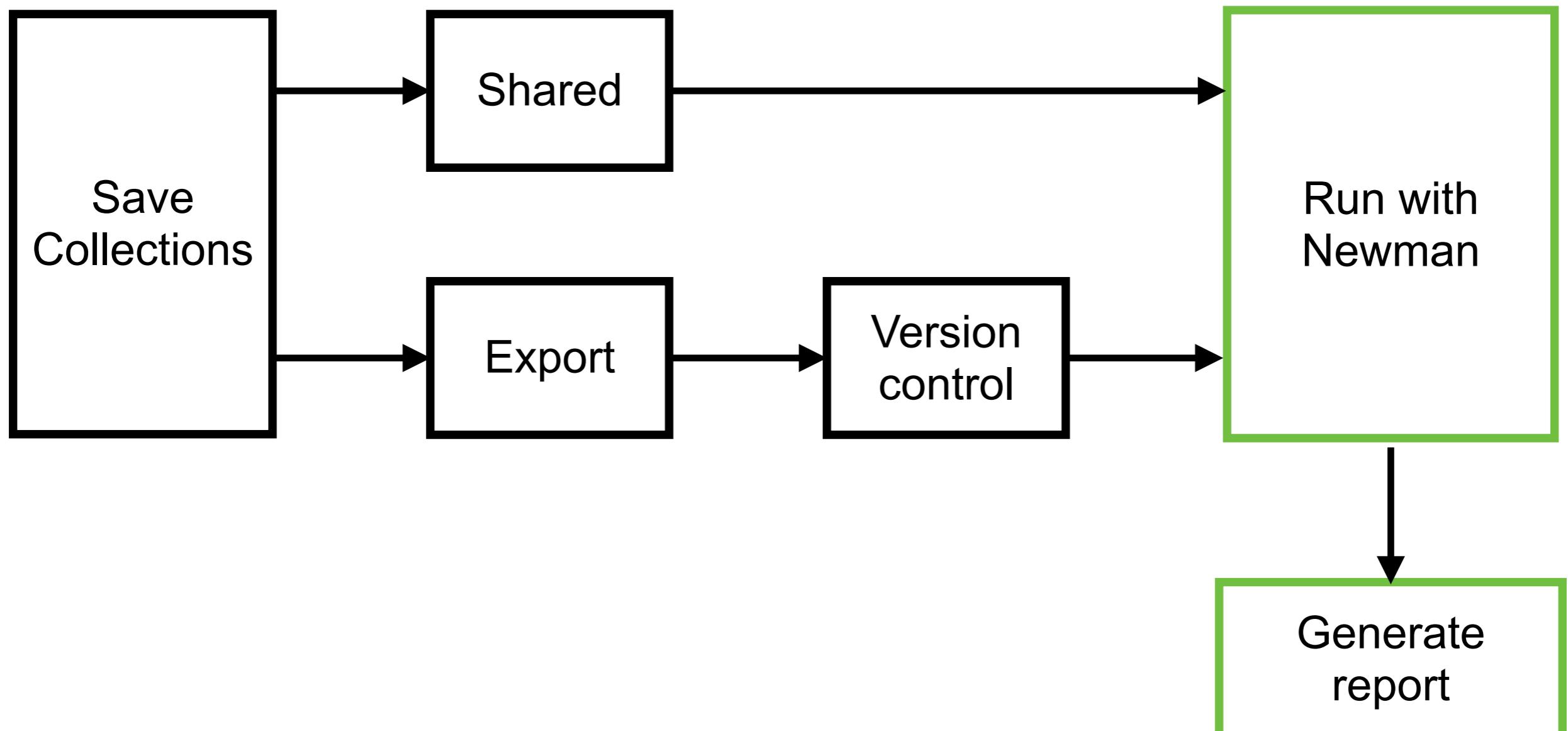
Send request and receive response  
Run collections in command line  
Integrate with CI/CD tools

***\$npm install -g newman***

<https://github.com/postmanlabs/newman>



# Working with Newman



# Newman CLI

Run collection

Run requests in folder

Working with data

Working with environments

**Working with OS variables**



# Working with OS variables

## Test script

```
var user = pm.globals.get("demo_username")
var pass = pm.globals.get("demo_password")
```

## Run in command line

```
$newman run your-collection.json
--global-var demo_username=demo user
--global-var demo_password=demo password
```



# Test Report

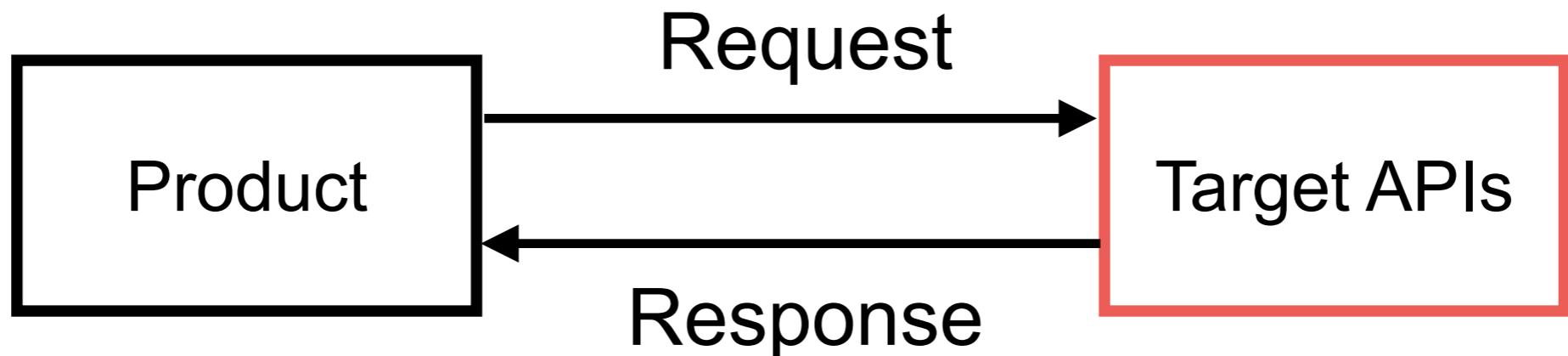
CLI  
JUNIT  
JSON  
HTML



# Mock Server



# Problems ?

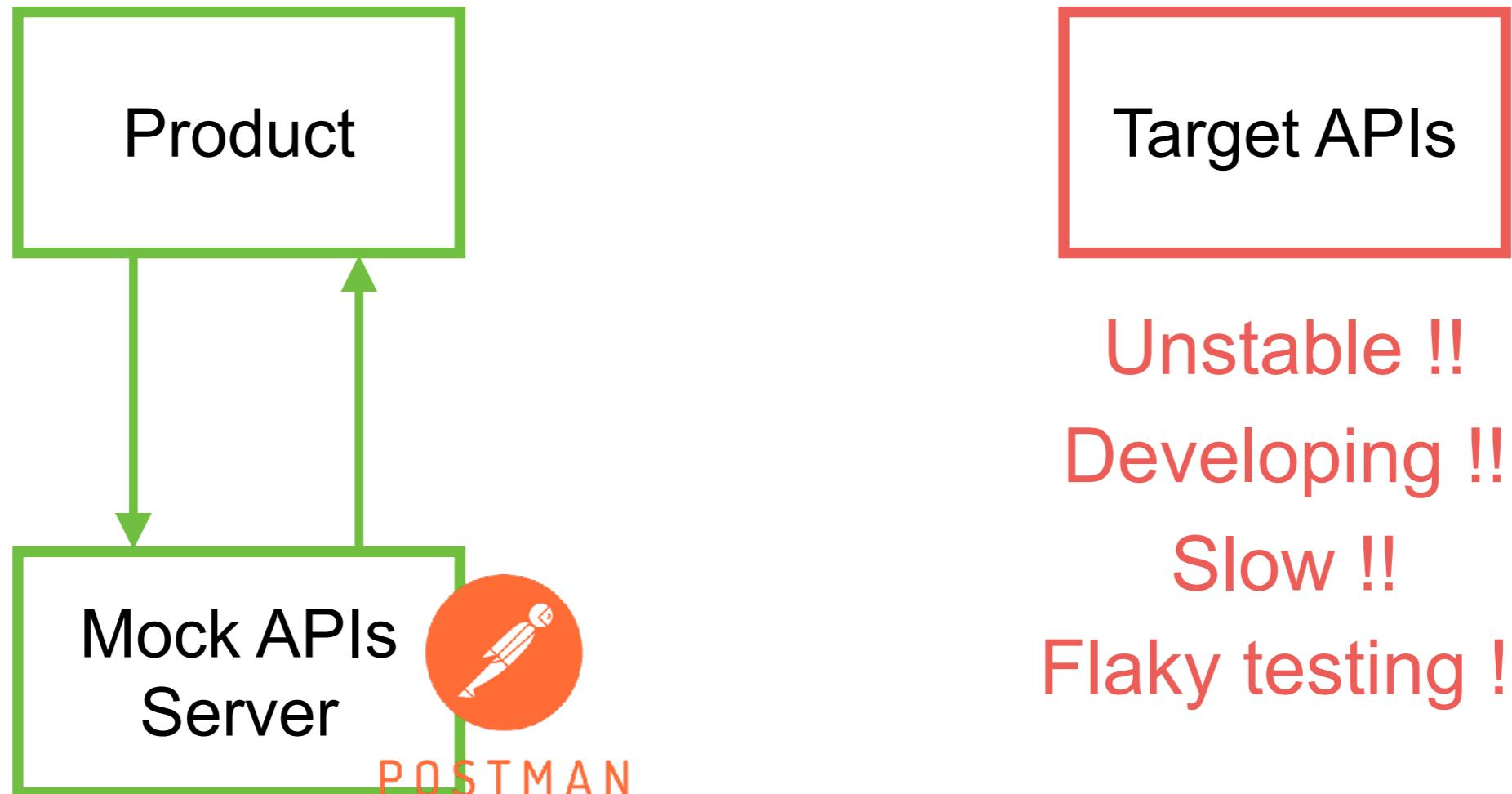


Unstable !!  
Developing !!  
Slow !!  
Flaky testing !!

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



# Mock Server with Postman



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



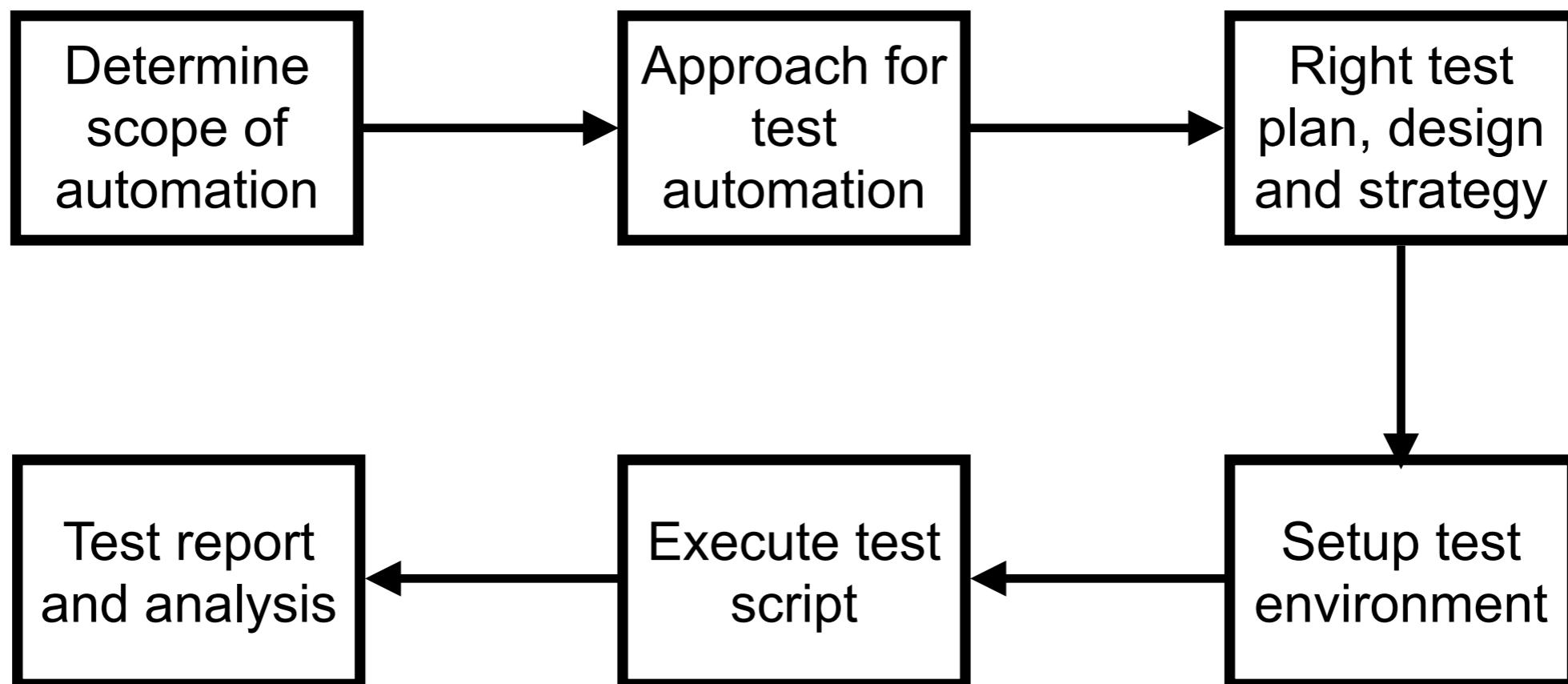
# Workshop MockServer



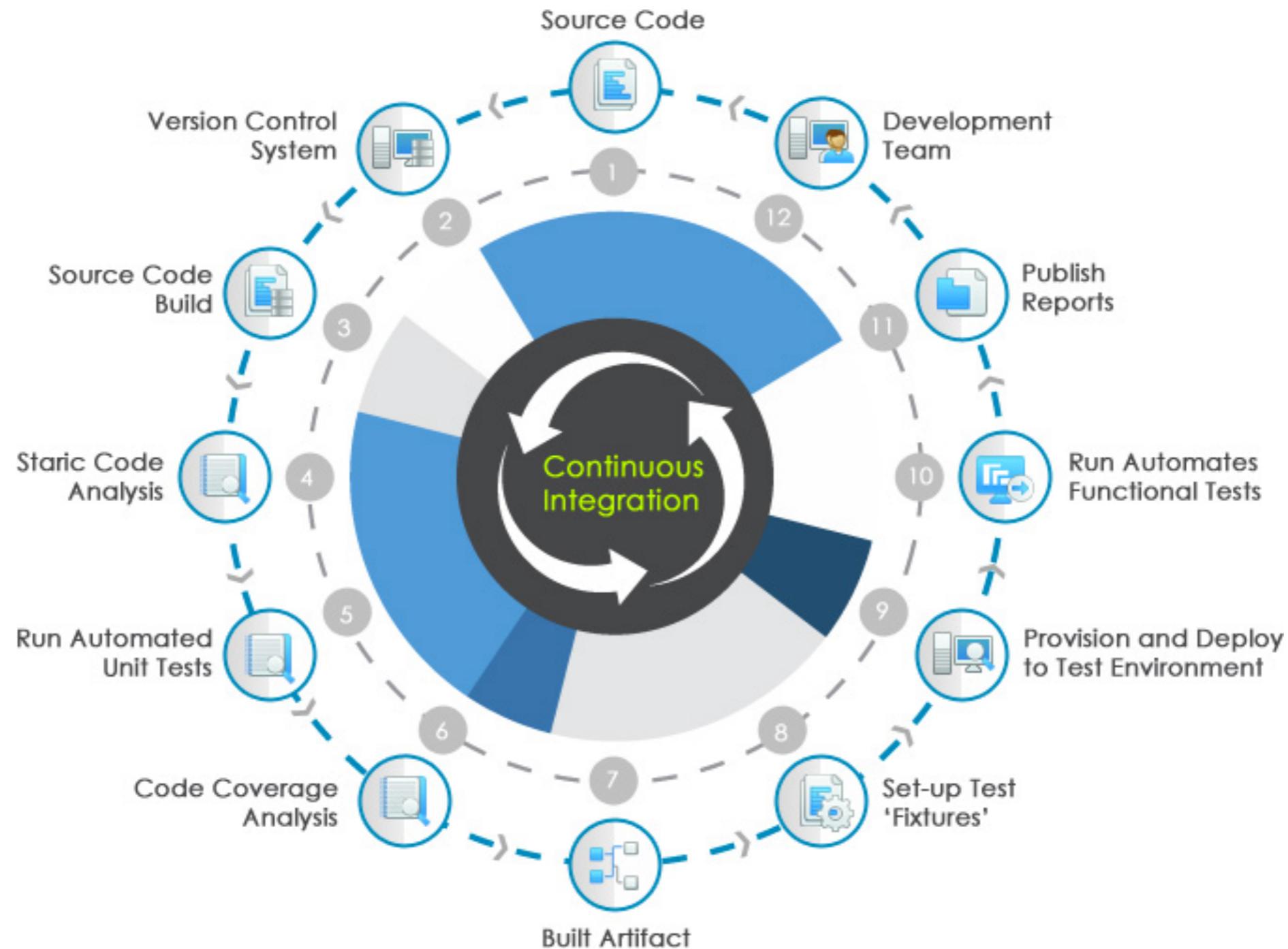
# Automated Testing Lifecycle



# Automated Testing Lifecycle



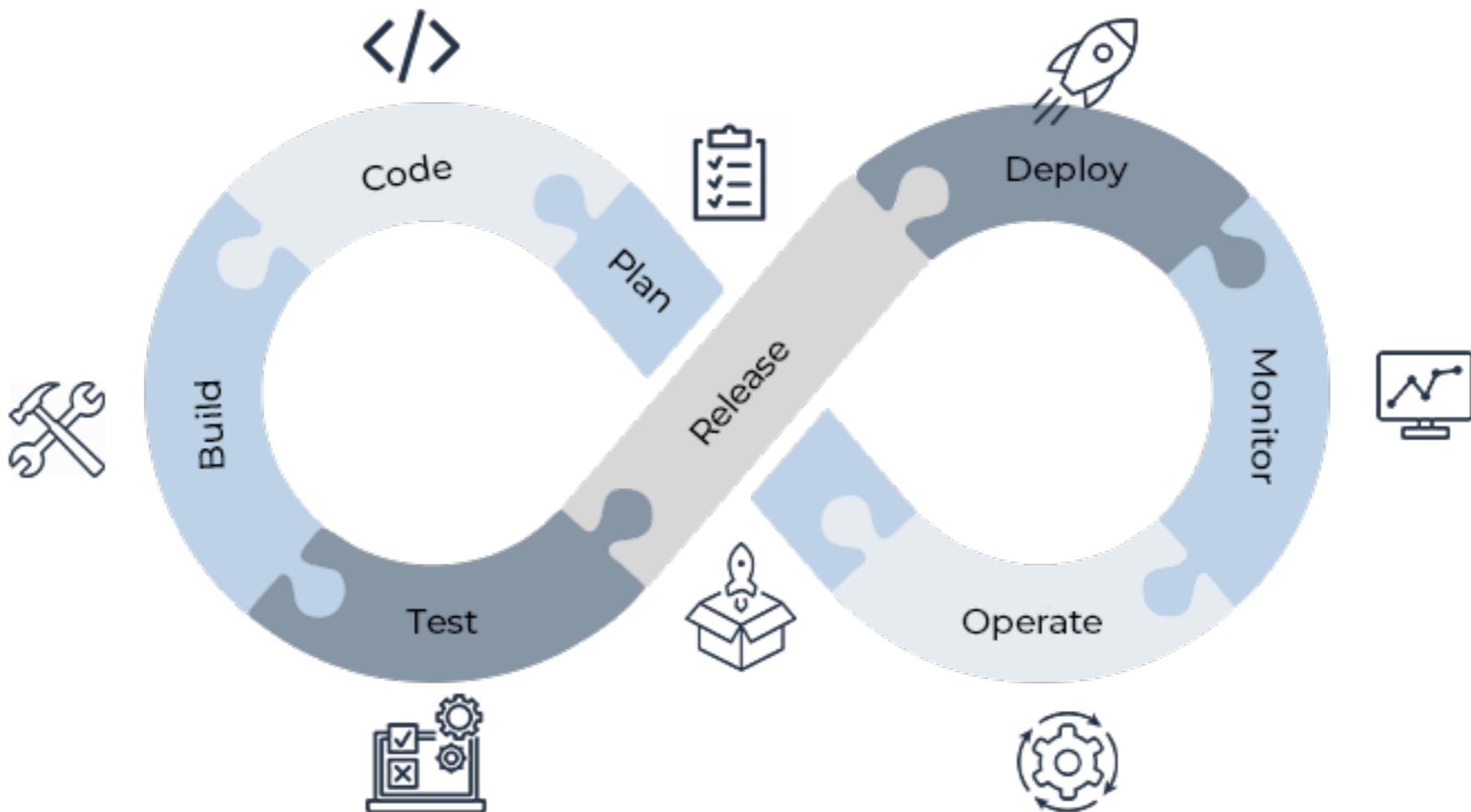
# Continuous Integration



# Continuous Testing



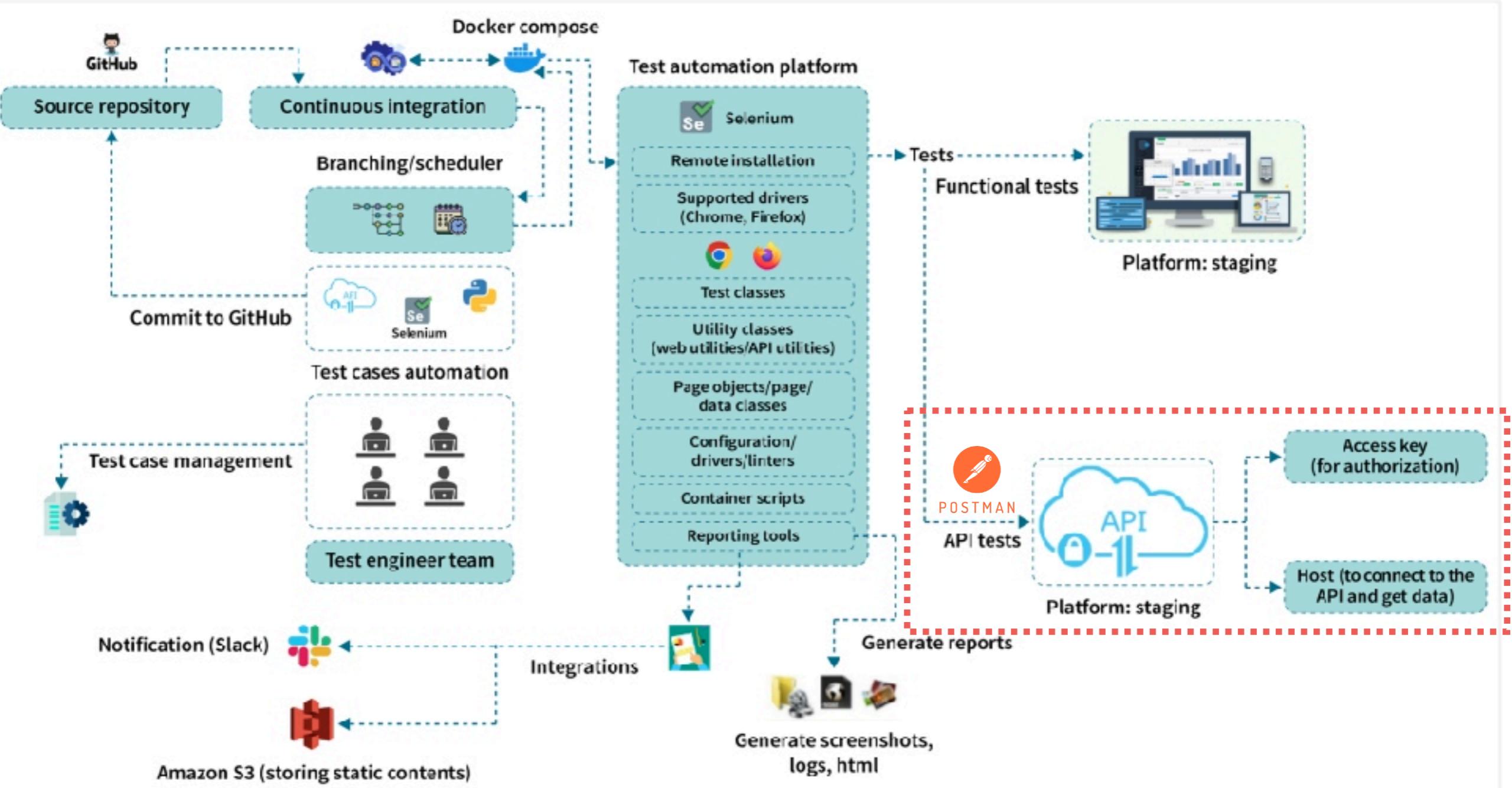
# DevTestOps



<https://www.xenonstack.com/insights/devtestops-advantages>



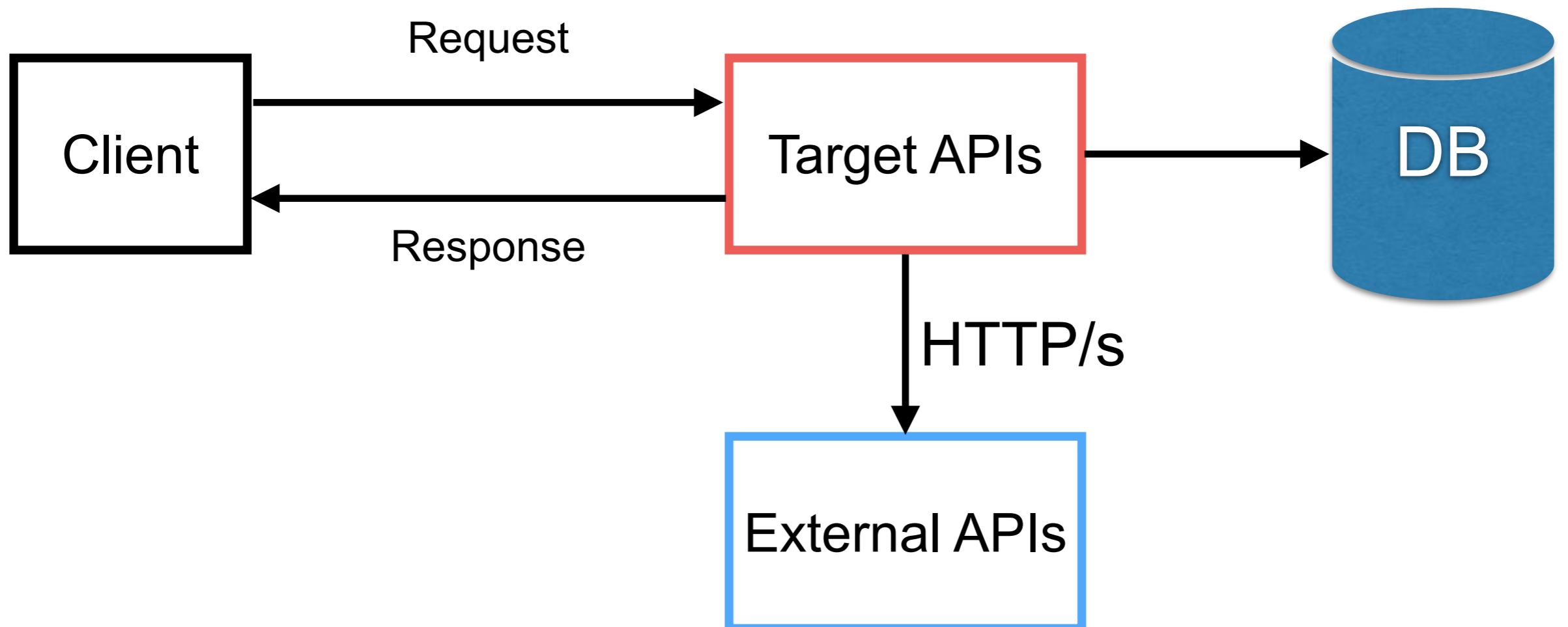
# Example



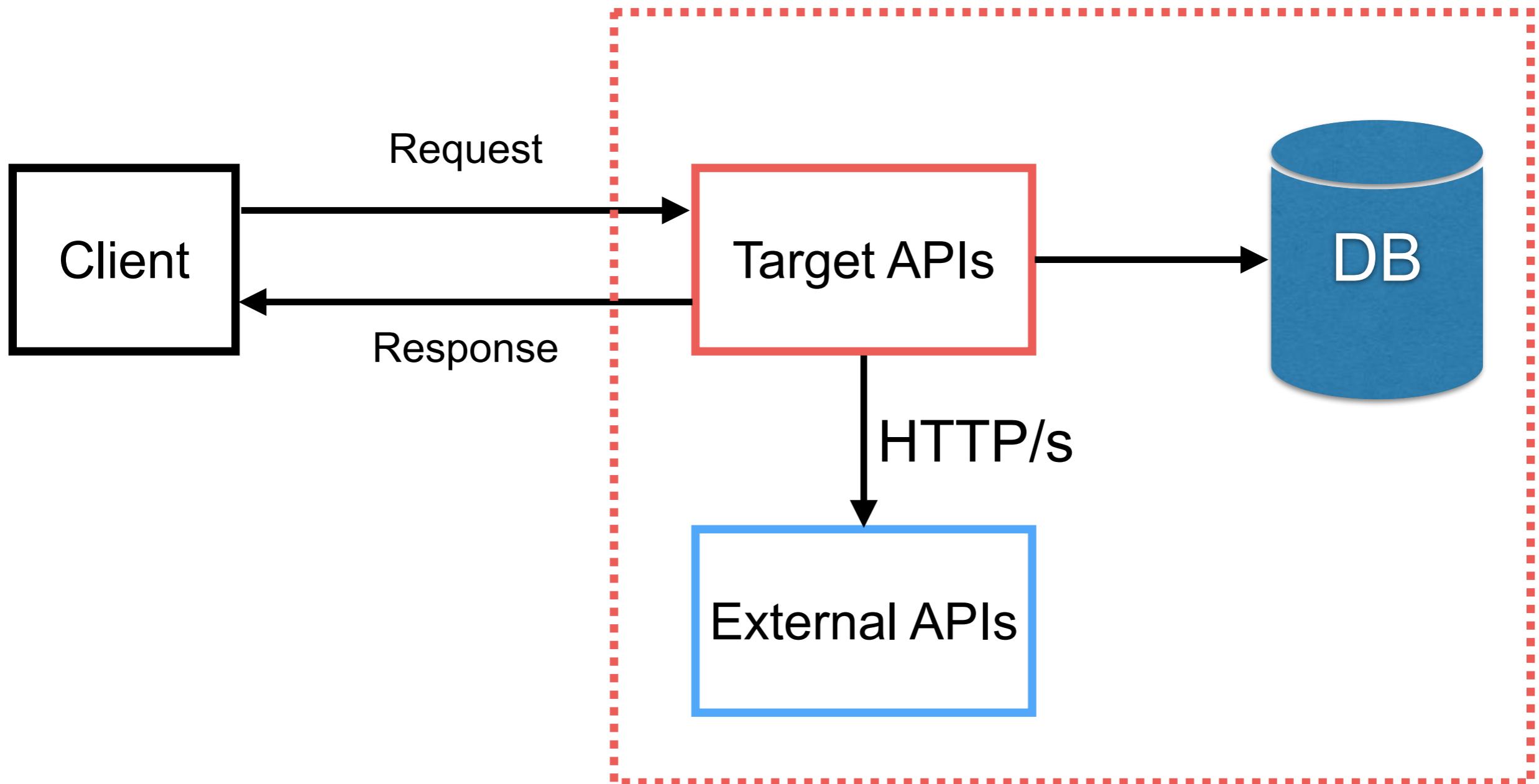
# Test Strategies ?



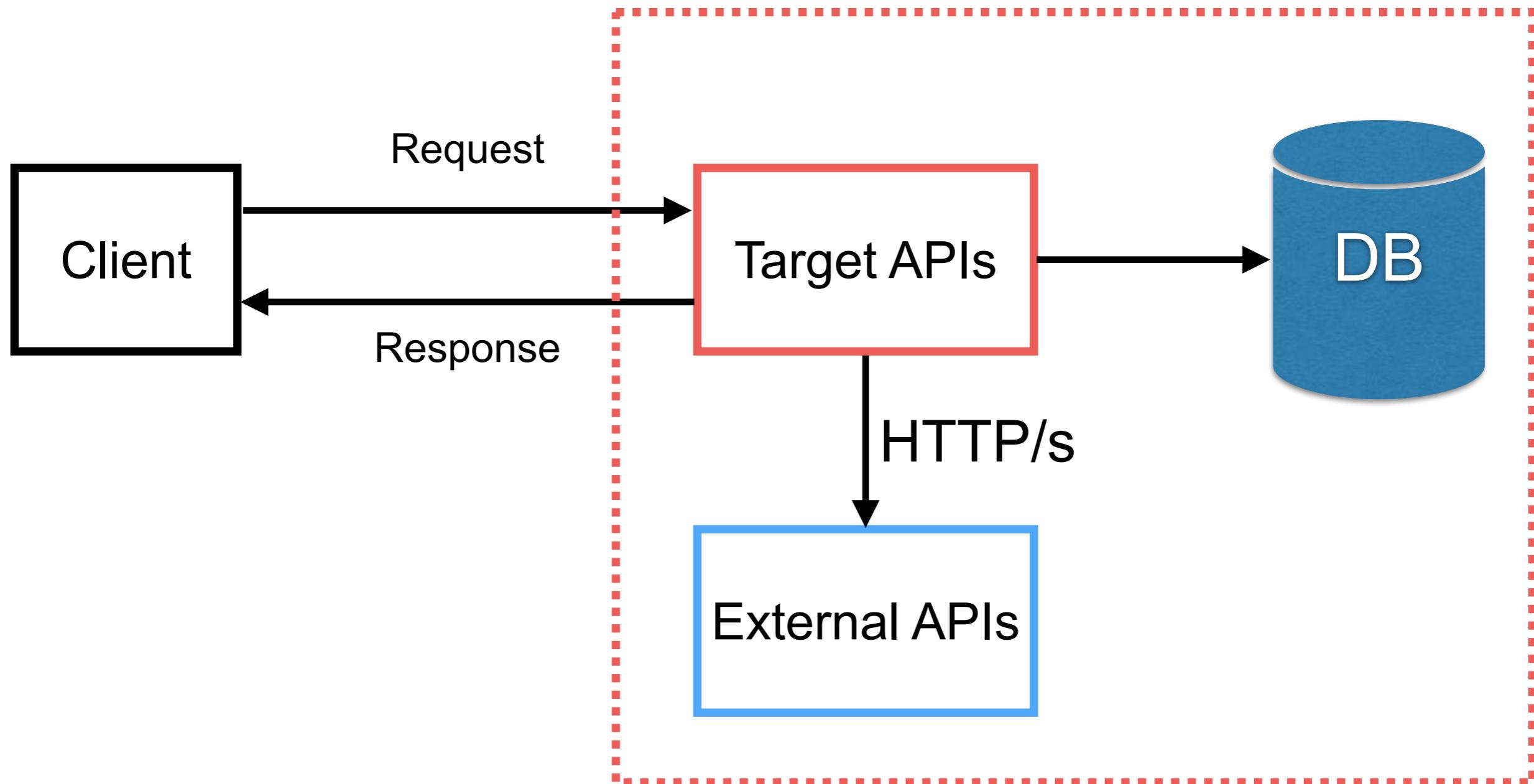
# Architecture



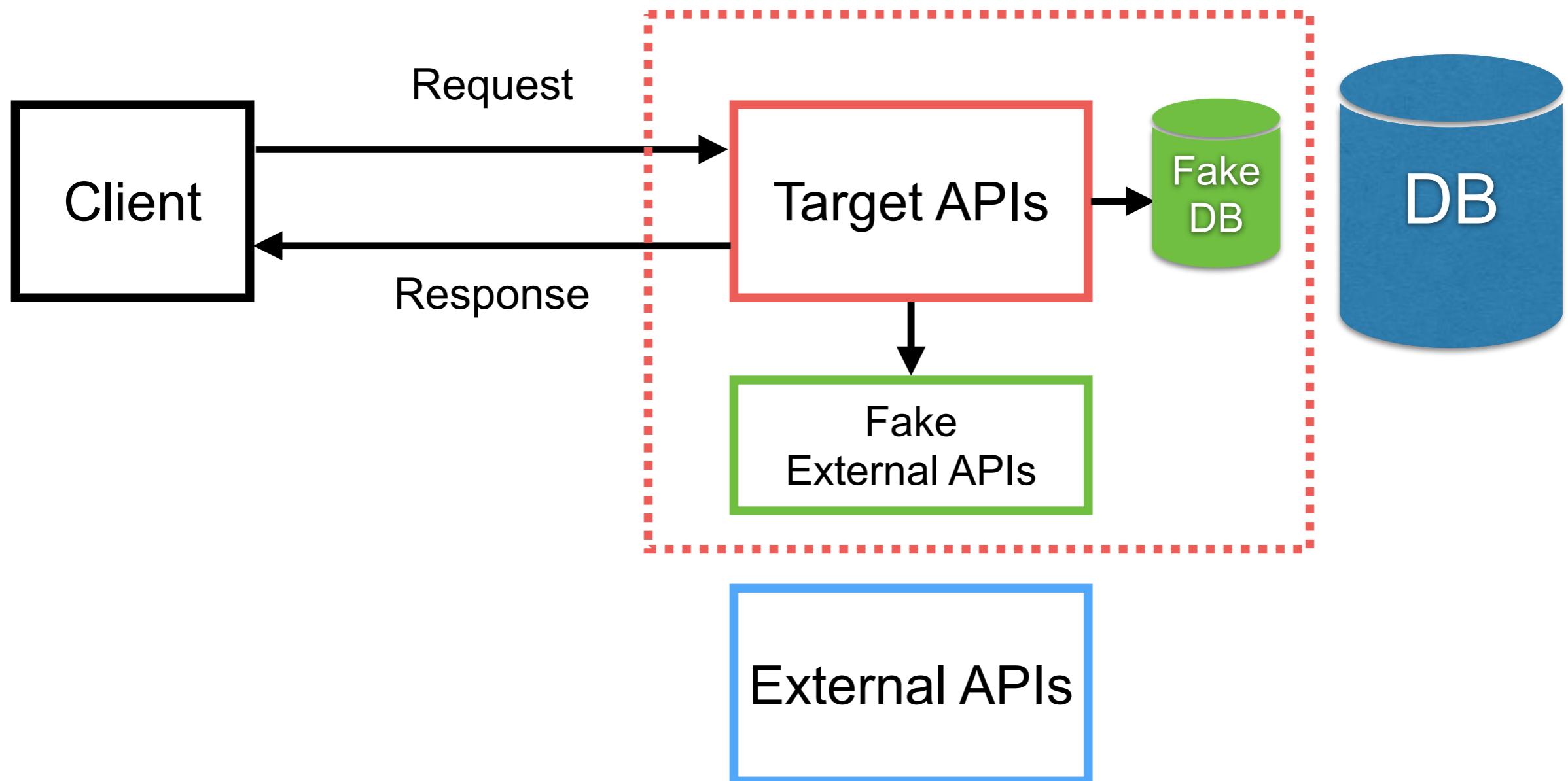
# What and Why to test ?



# Testing with real dependencies



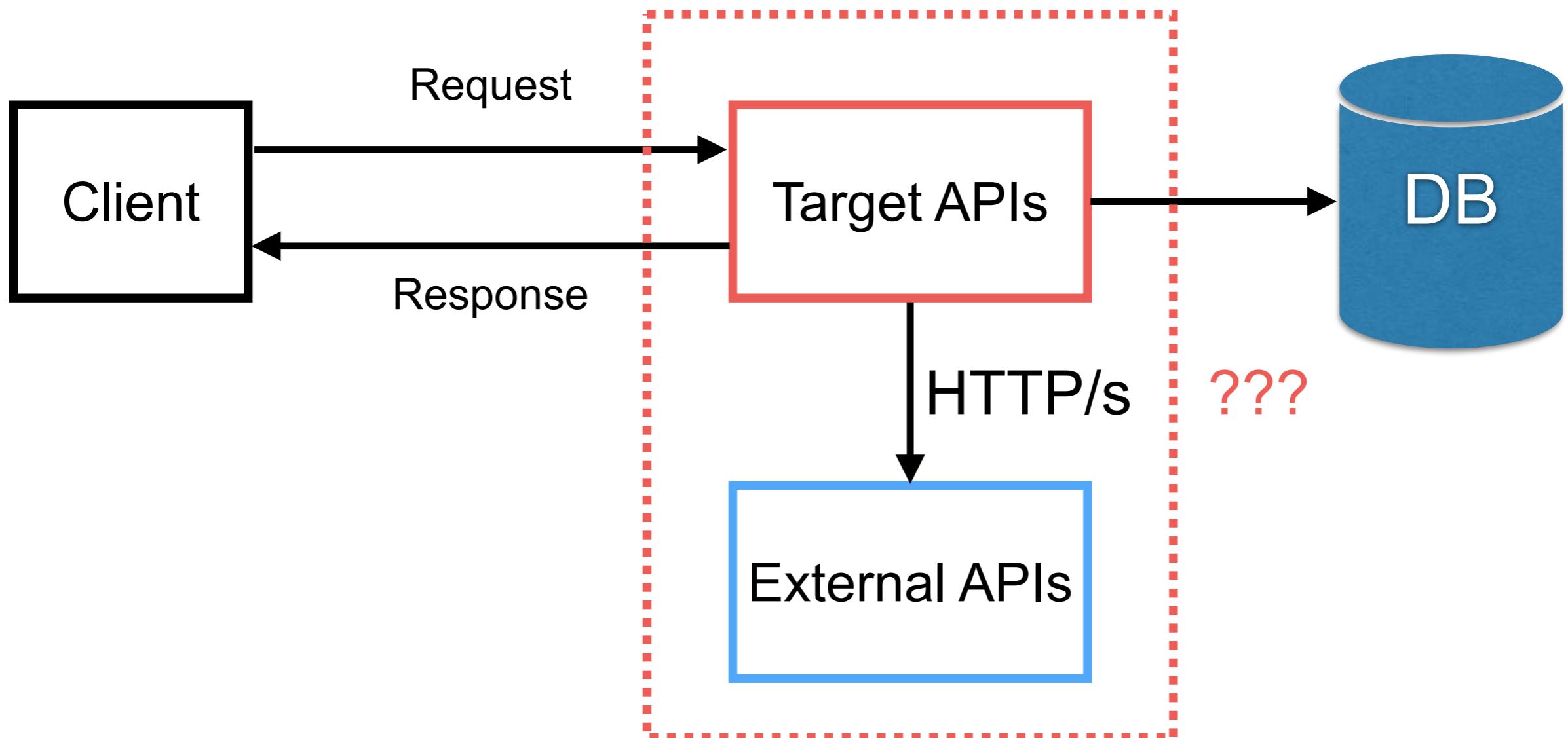
# Testing with fake dependencies

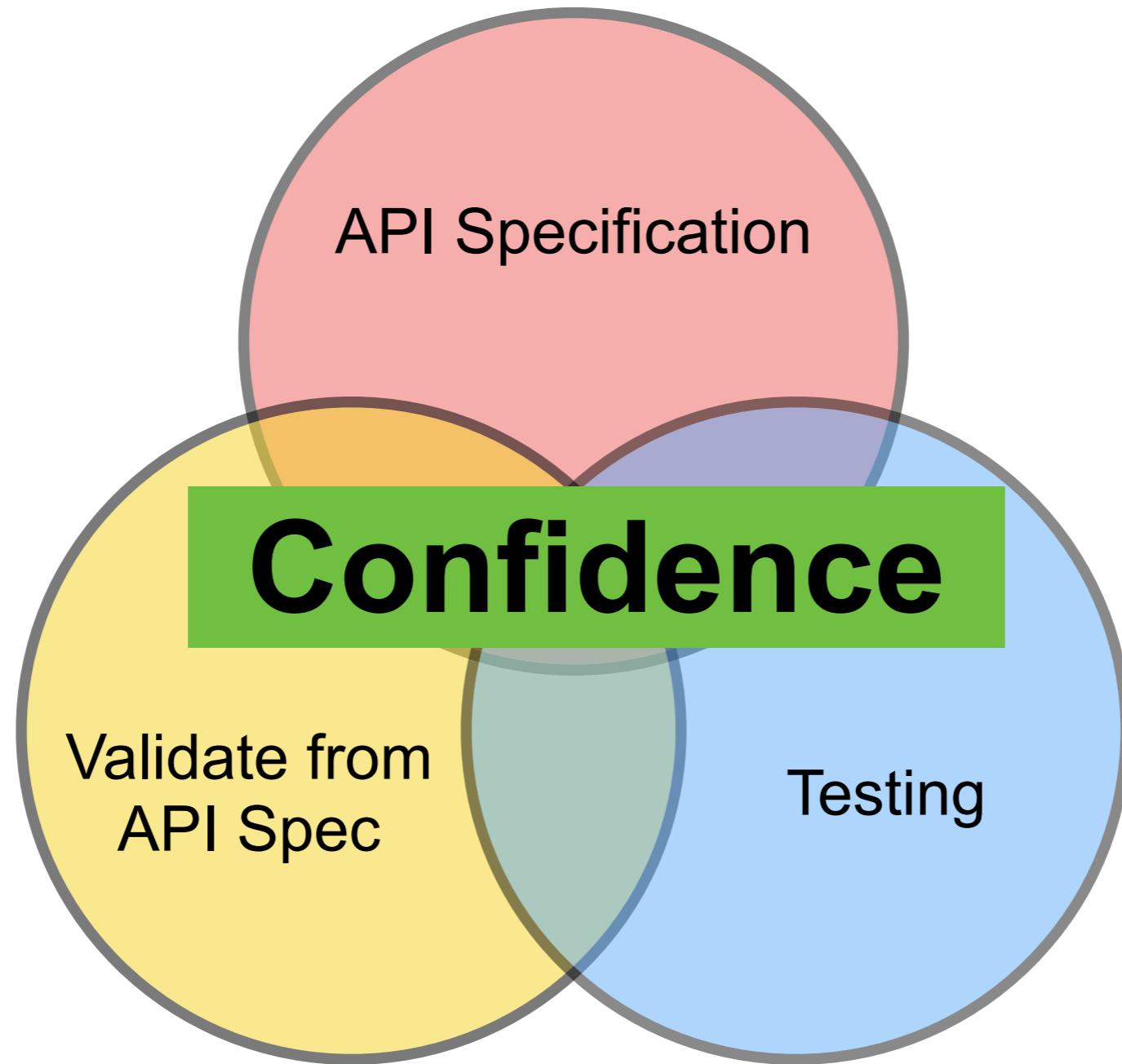


# Contract Testing

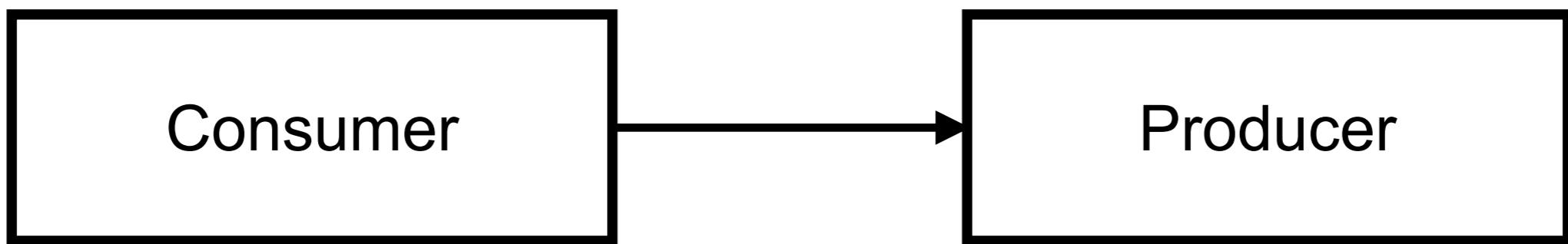


# Contract Testing ?





# Testing ?

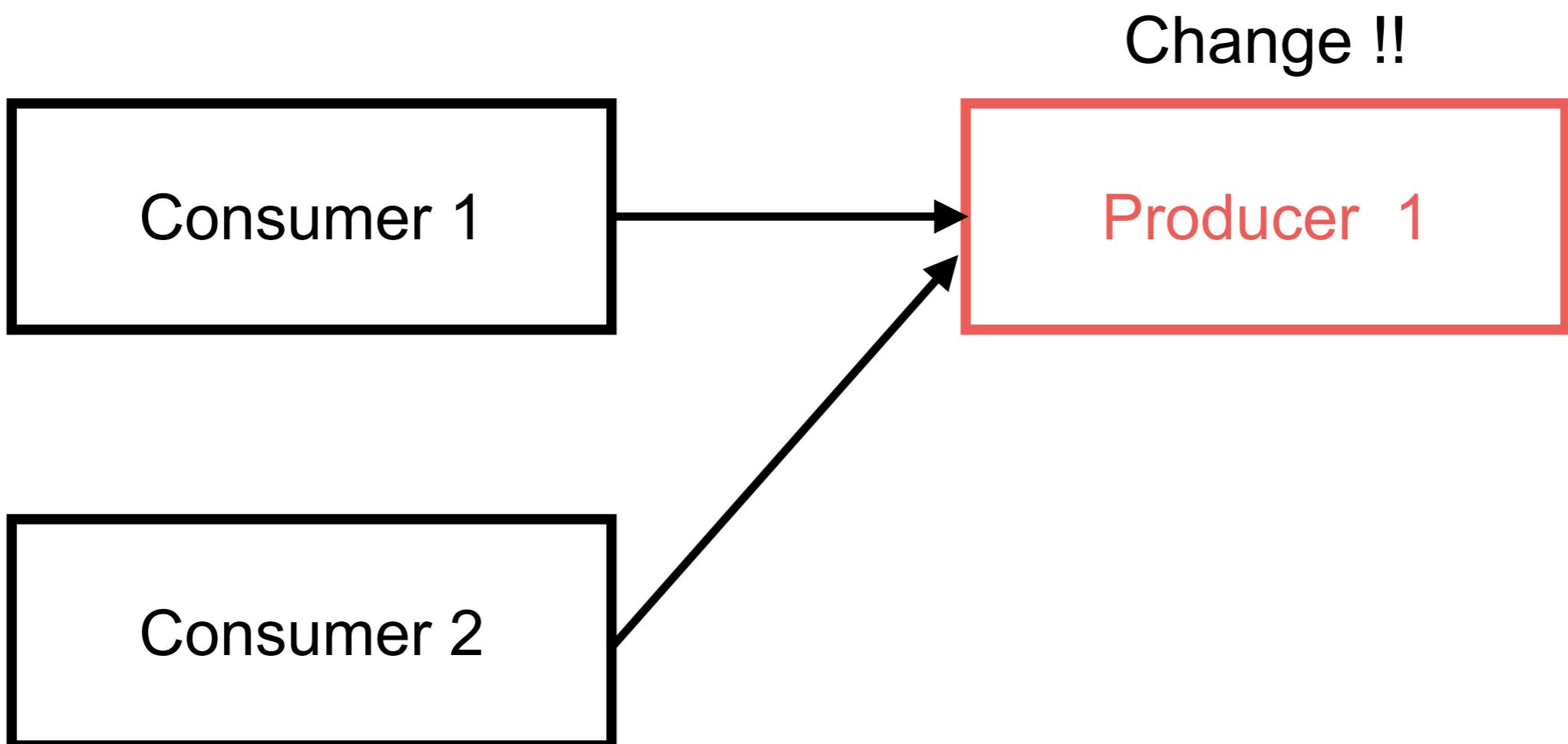


How to call/use API ?

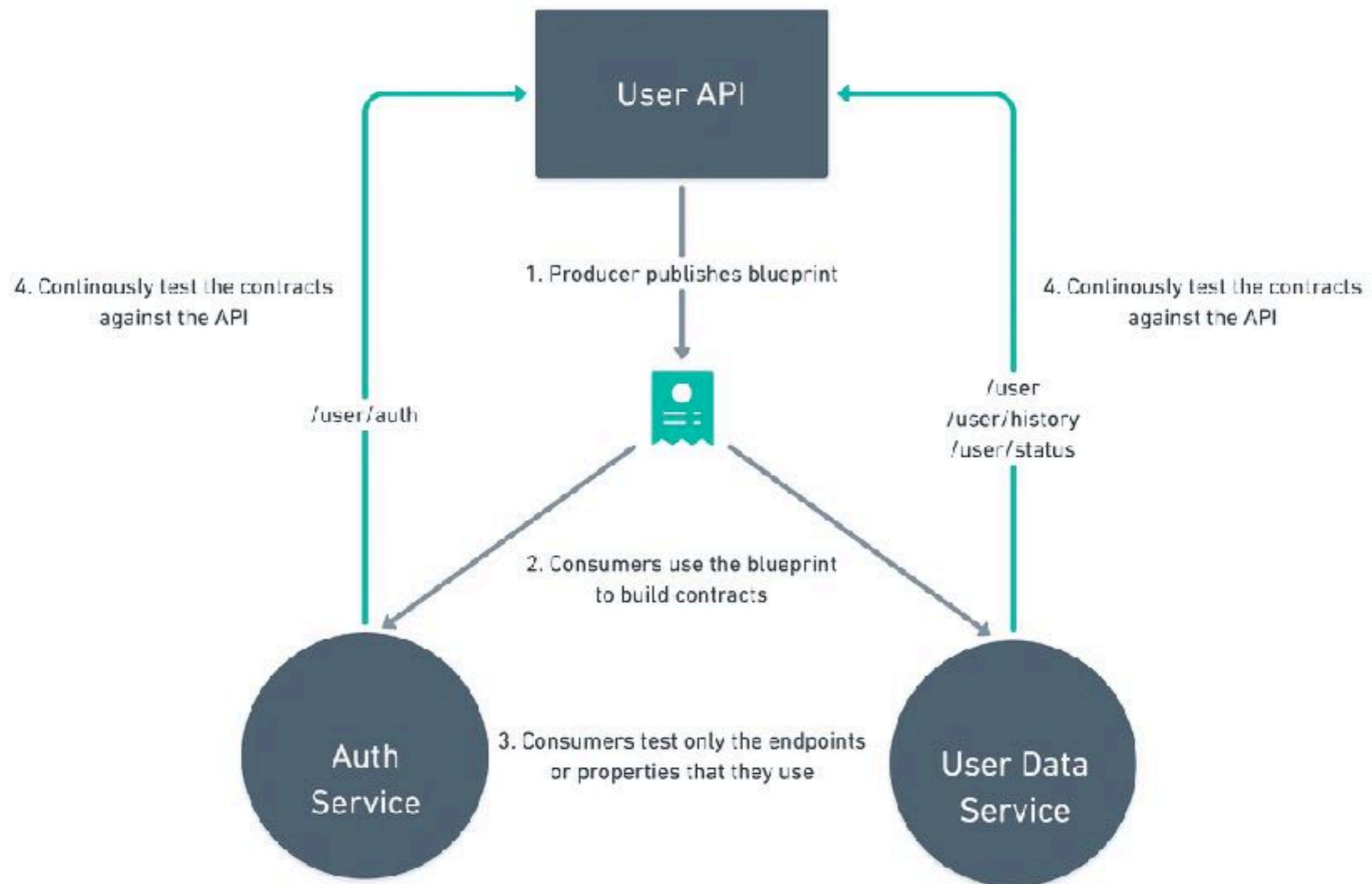
How consumer use my API ?



# Testing ?



# Example



<https://medium.com/better-practices/consumer-driven-contract-testing-using-postman-f3580dba5370>



# Workshop

<https://postman-echo.com>



# 1. Producer publish blueprint

Design API Specification  
**Validate schema**

## API

GET /get?key1=value1&key2=value2

POST /post  
key1=value1  
key2=value2



## 2. Consumer create contract

Call APIs from provider

Define inputs and expected results

Create requests

**Write test scripts to verify**



# 3. Mock provider in consumer-side

Easy to test and develop in consumer-side  
Define inputs and expected results



# More testing !!



# Snapshot testing



# Performance testing



# Security testing



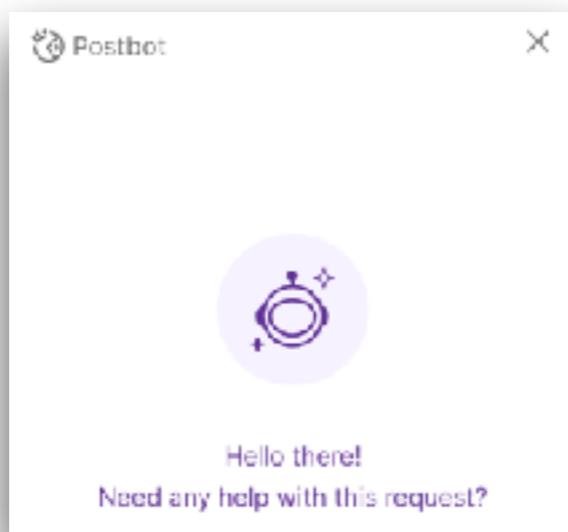
# More features



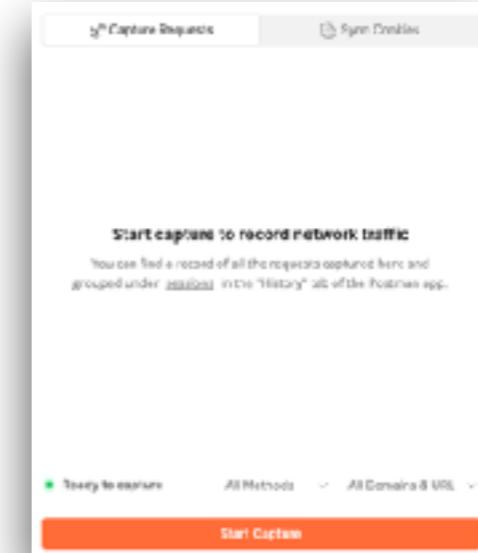
POSTMAN



# More features



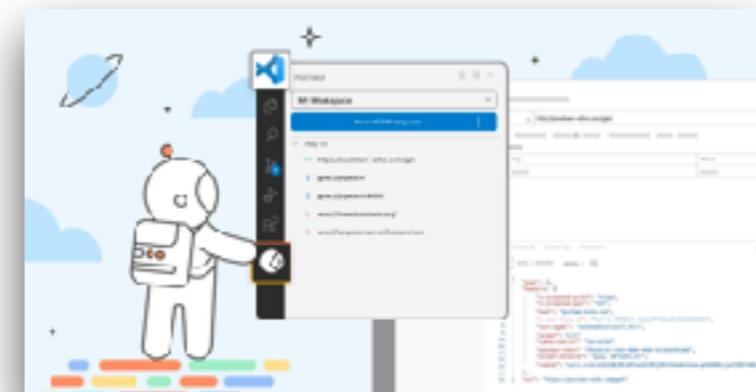
AI assistant for API testing



Postman interceptor



Working with OpenAPI/Swagger



Postman extension in VSCode



# Summary



# Better delivery process

Think about testing  
Fast feedback  
High quality  
Working as a team



# Q/A

## ***Be Skilled Software Tester the Series***



Stay Tuned



<https://www.facebook.com/beSkilledSoftwareTester>



Testing

© 2013 - 2023 Siam Chamnankit Company Limited. All rights reserved.