

API TESTING WITHIN CI/CD

Fran Guerrero



Fran Guerrero - QA Manager/Lead



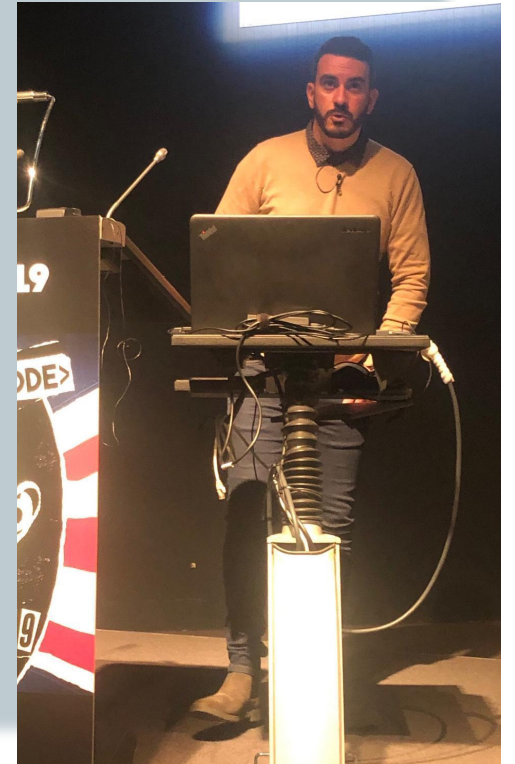
[linkedin.com/in/franguerrero](https://www.linkedin.com/in/franguerrero)



[@franguerreroQA](https://twitter.com/franguerreroQA)



fran.guerrero.sanchez@gmail.com



API TESTING WITHIN CI/CD

1

API Testing ❤️ Postman

Collections - Tests - Newman

2

Postman ❤️ Docker

Containers - Docker - Dockerhub

3

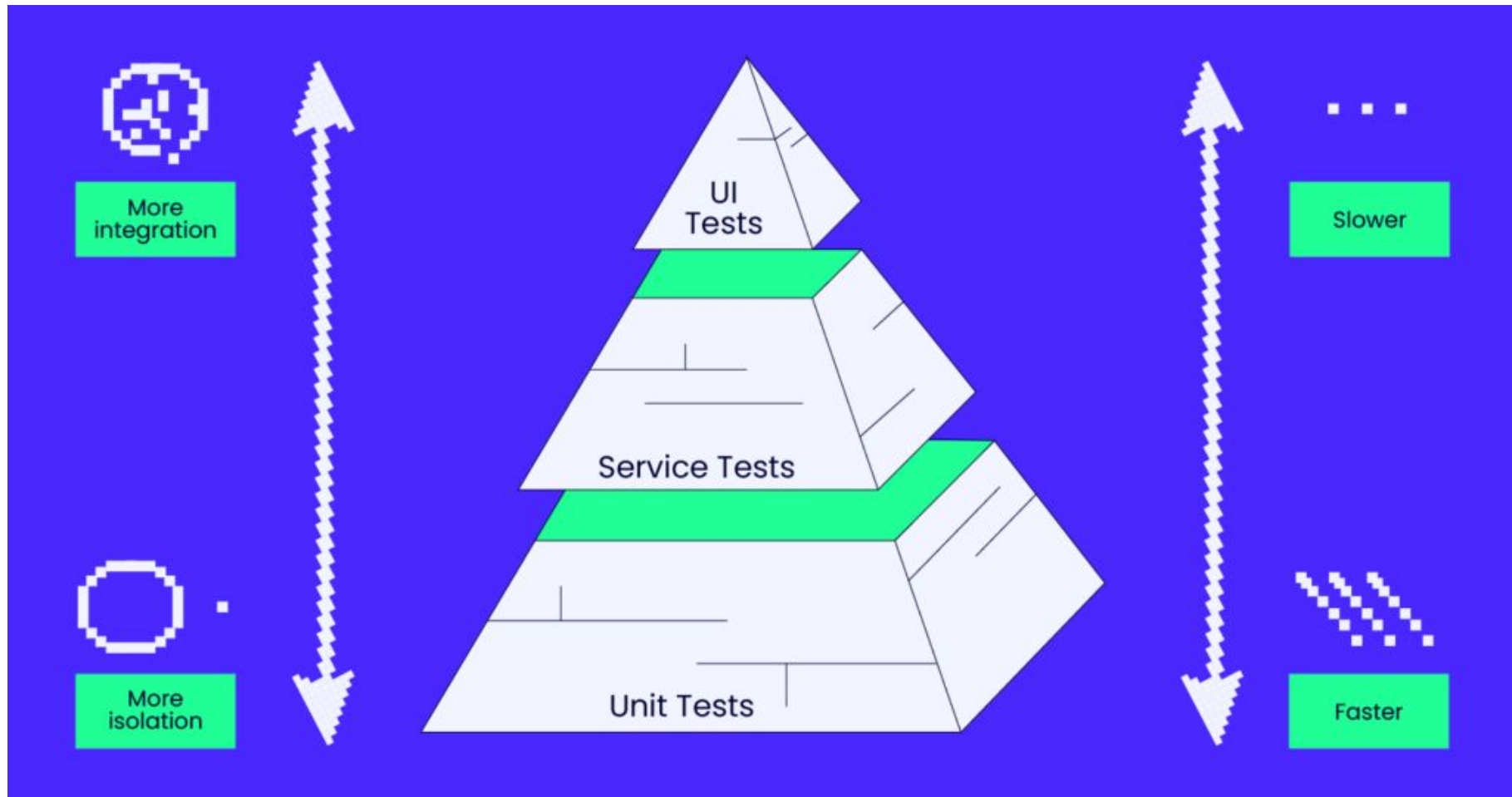
Postman ❤️ GitlabCI

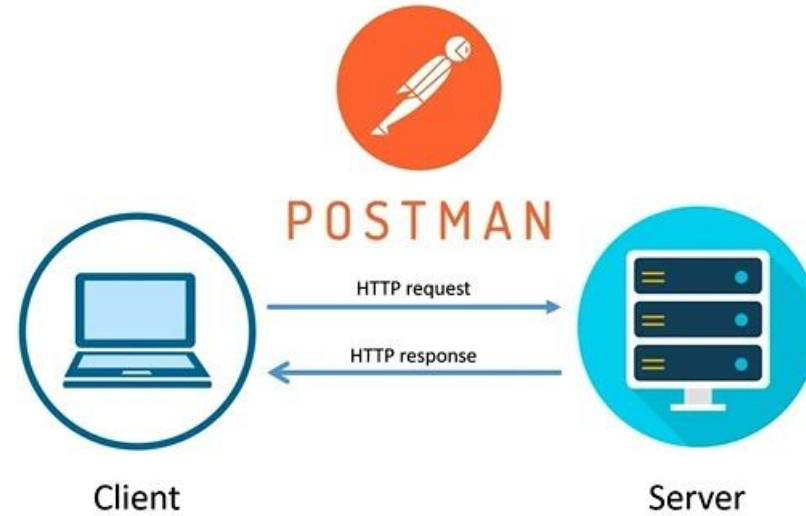
CI/CD - GitlabCI

API



API



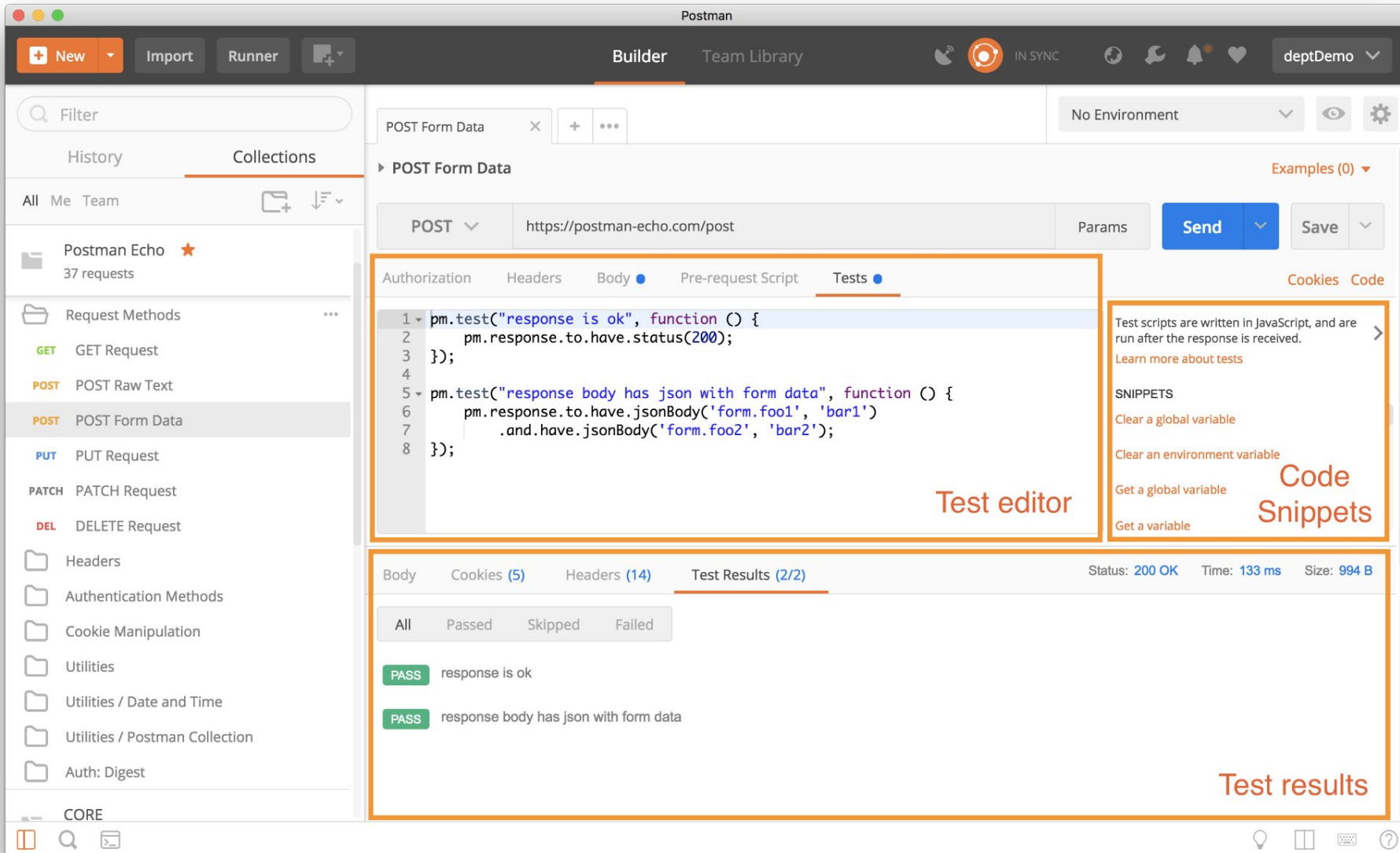


Postman: the API desktop tool client. The tool is used to send requests and inspect the responses, in order to easily debug the behavior of the API service during its development.

API Testing



Postman







API Testing



Postman

RUN ORDER

Deselect All | Select All | Reset

- ☒  > GET Get all characters
- ☒  > GET Get a single character
- ☒  > GET Get multiple character
- ☒  > GET Filter characters

Choose how to run your collection

- ☒ Run manually
Run this collection in the Collection Runner.
- ☐ Schedule runs
Periodically run collection at a specified time on the Postman Cloud.
- ☐ Automate runs via CLI
Configure CLI command to run on your build pipeline.

Run configuration

Iterations

Delay

ms

Data

Select File

☐ Persist responses for a session ⓘ

> Advanced settings

Run Rick and Morty API TestAca...

```
npm install -g newman
```

```
newman run "path/collection.json"
```

```
newman run "http://URL_collection"
```

Postman



Newman


```
Downloads $ newman run Postman_Newman_IntegrationCollection.json
newman

Postman-Newman Integration

- Register User
  POST https://reqres.in/api/register [200 OK, 620B, 513ms]
  ✓ Status code is 200

- Get User
  GET https://reqres.in/api/users/4 [200 OK, 738B, 60ms]
  ✓ Status code is 200

- Login User
  POST https://reqres.in/api/login [200 OK, 465B, 250ms]
  ✓ Status code is 200
```

	executed	failed
iterations	1	0
requests	3	0
test-scripts	6	0
prerequest-scripts	4	0
assertions	3	0
total run duration: 986ms		
total data received: 228B (approx)		
average response time: 274ms [min: 60ms, max: 513ms, s.d.: 185ms]		

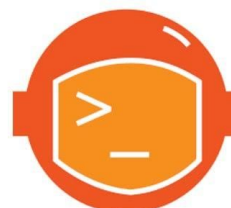
summary

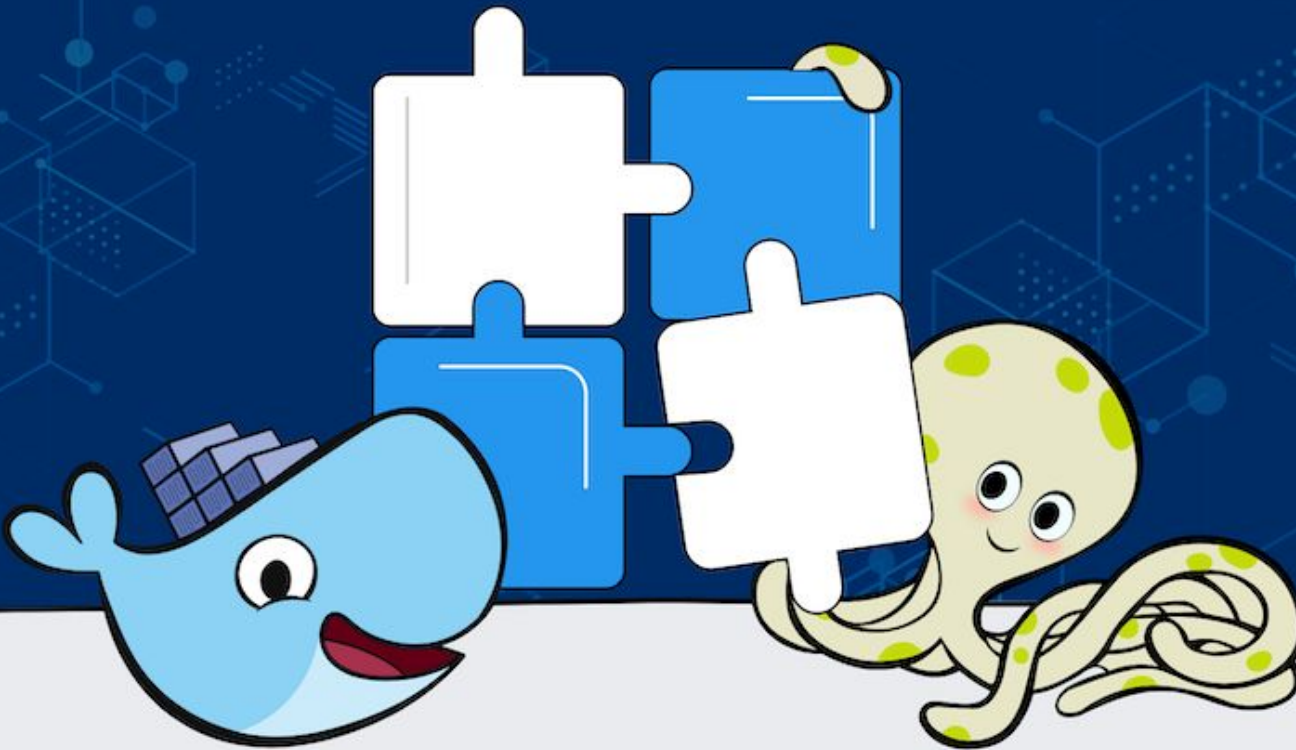
Postman



Newman

Newman: is a command line agent that allows to run multiple requests sequentially. It takes as input the Postman Collection extracted as a json file and return the responses with the results of the each test, if any.





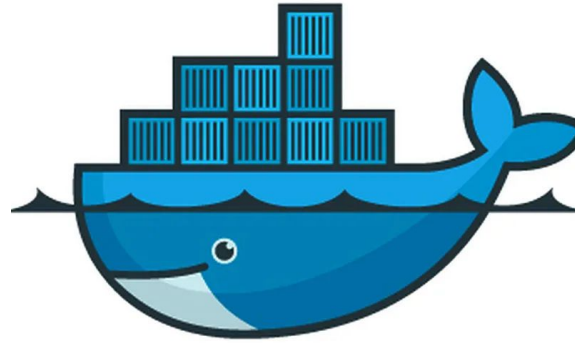
Postman



Docker

Containers

- Faster to build
- Stub unused services
- Spin up new data in a known state for each test run
- Parallel test running
- Good for Service, Integration and E2E Smoke tests
- <https://docs.docker.com/engine/install/>

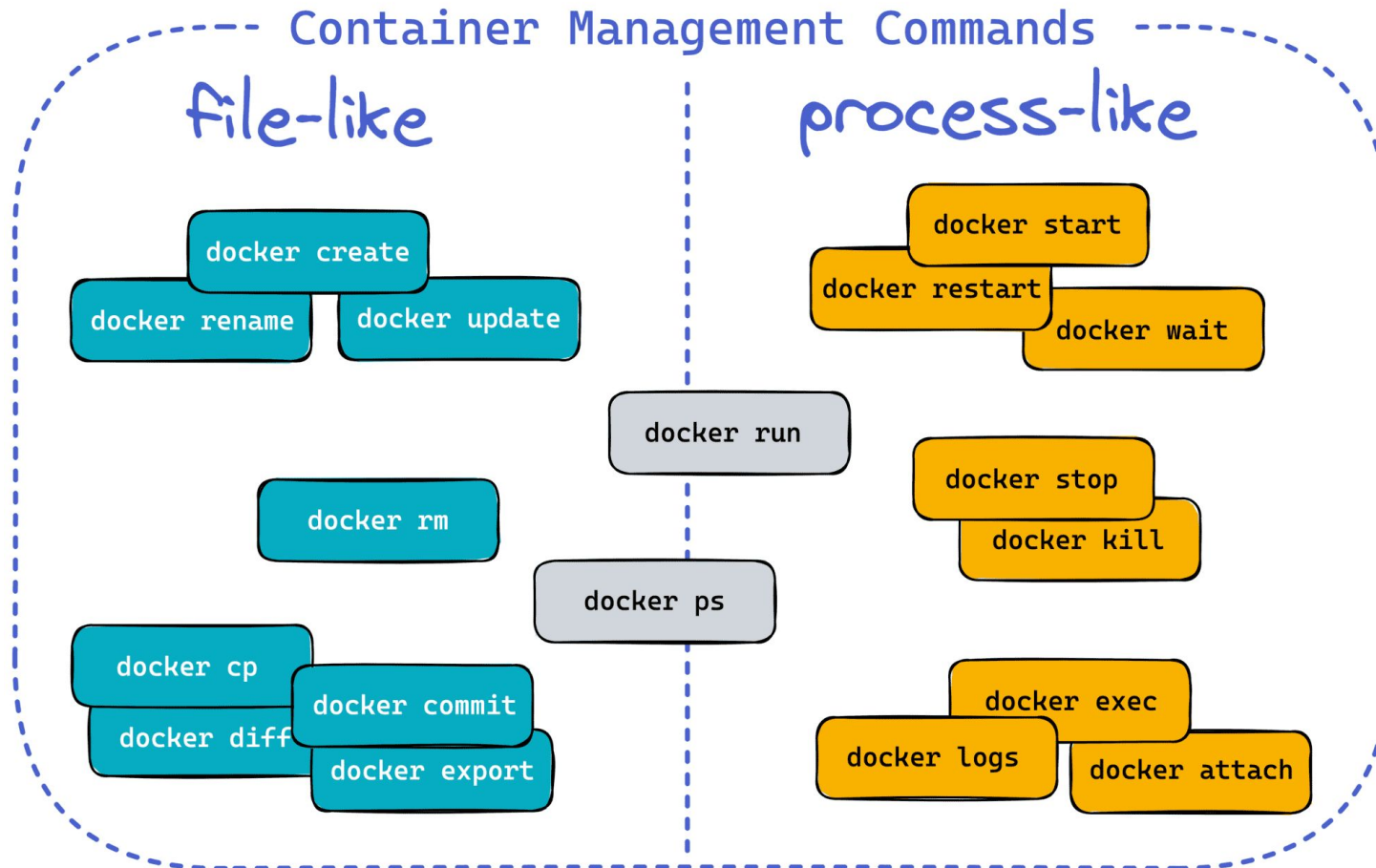


Postman



Docker

Docker commands

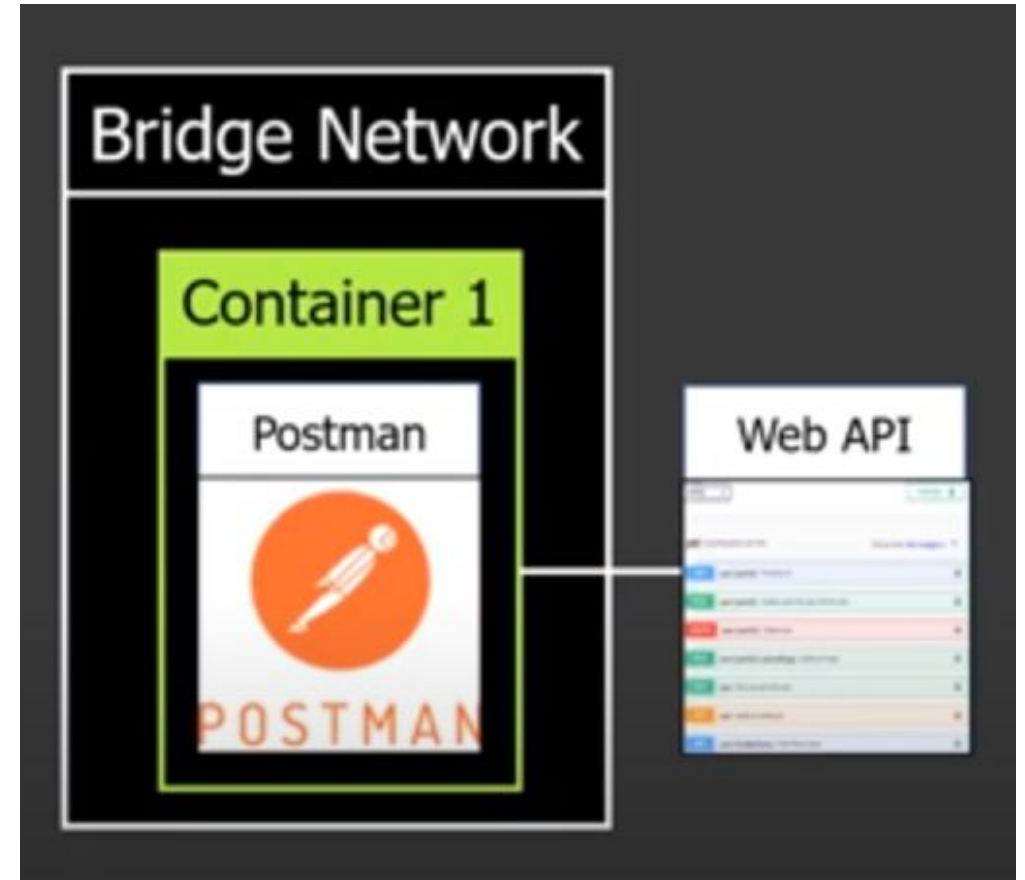


Postman



Docker

- We can run all sorts of applications in a container, including testing tools like Postman.
- We will add Postman to a Docker container, point it to an existing deployed API and test it in the cloud; so that we don't need to have the tool installed to our physical machine.
- ```
docker run -t postman/newman
run <URL_collection>
```



<https://hub.docker.com/r/postman/newman/>



**postman/newman** ☆

By [postman](#) • Updated a year ago

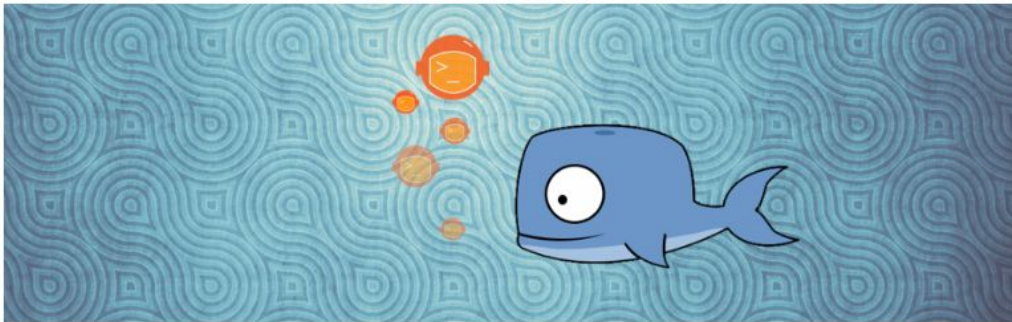
Command-line collection runner for Postman

Image

↓ Pulls 10M+

Overview

Tags



Docker Pull Command

```
docker pull postman/newman
```



## newman-docker

This repository contains docker images for Newman.

[Newman](#) is a command-line collection runner for [Postman](#). It allows you to effortlessly run and test a [Postman Collections](#) directly from the command-line. It is built with extensibility in mind so that you can easily integrate it with your continuous integration servers and build systems.

**New to Docker?** Docker allows you to package an application with all of its dependencies into a standardised unit for software development. Visit <https://www.docker.com/whatisdocker> to read more about how docker can drastically

# Postman



# Docker



- Postman will be run and a report will be printed out to the terminal including Overall results and details of failed tests

```
↳ Get multiple characters
GET https://rickandmortyapi.com/api/character/1,2 [200 OK, 5.88kB, 289ms]
✓ Status code is 200
```

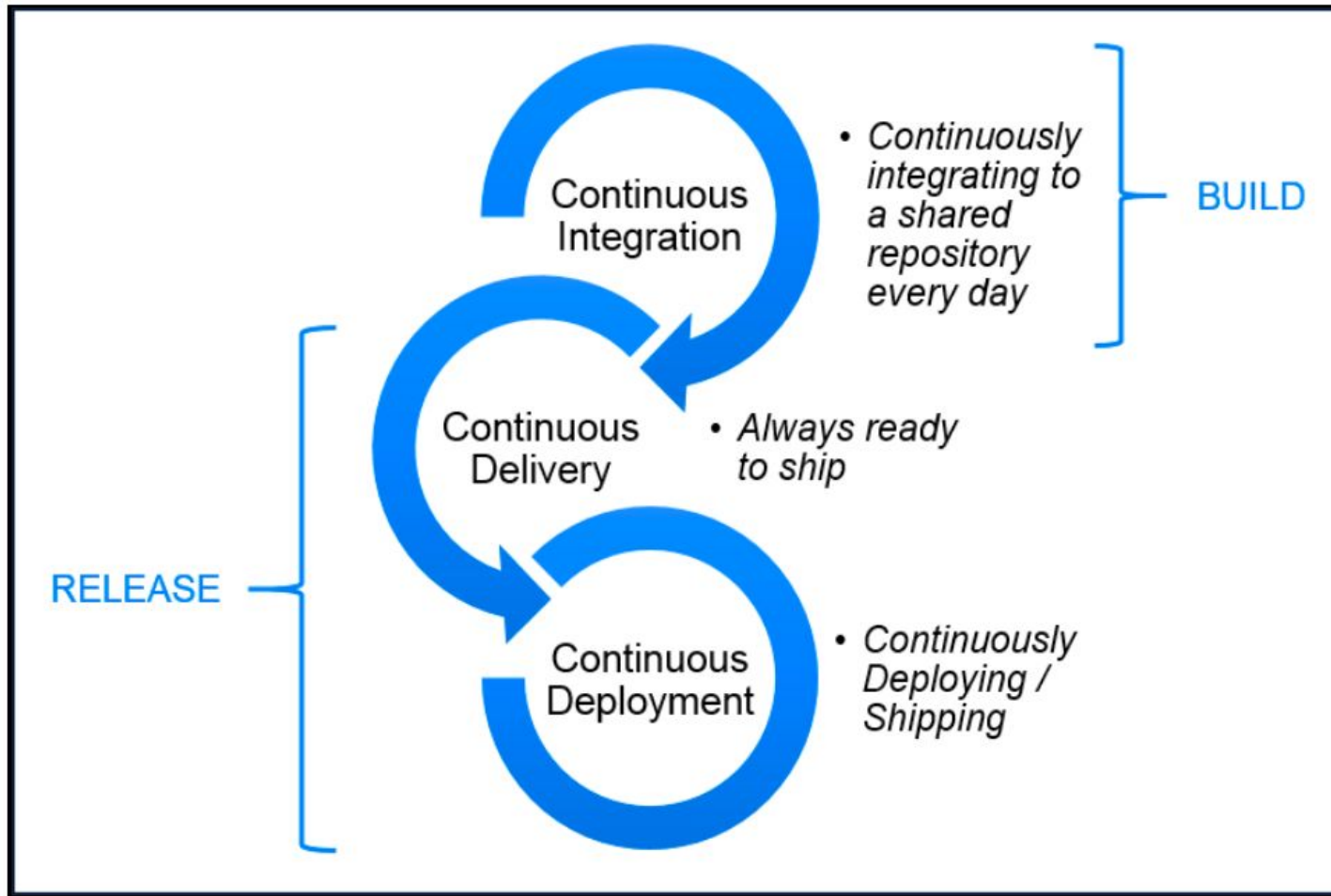
|                      | executed | failed |
|----------------------|----------|--------|
| iterations           | 1        | 0      |
| requests             | 3        | 0      |
| test-scripts         | 9        | 0      |
| prerequisite-scripts | 6        | 0      |
| assertions           | 5        | 0      |

```
total run duration: 616ms
total data received: 27.61kB (approx)
average response time: 172ms [min: 48ms, max: 289ms, s.d.: 98ms]
fquerrero@fquerrero:~/Descargas$
```

Postman



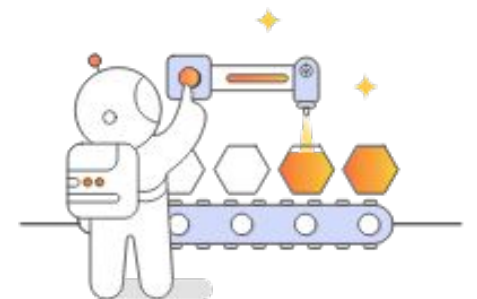
Docker

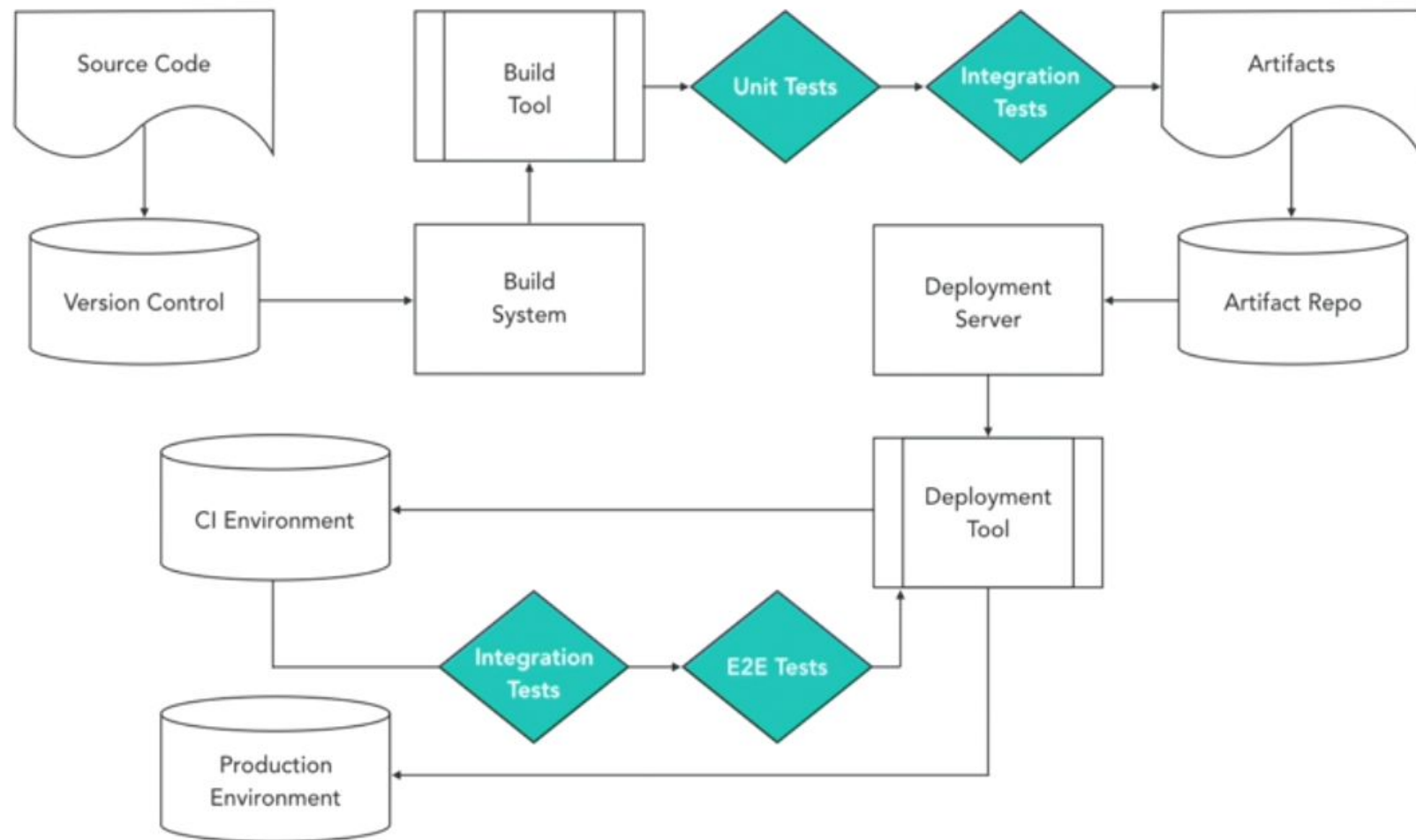


# Postman



# GitlabCI

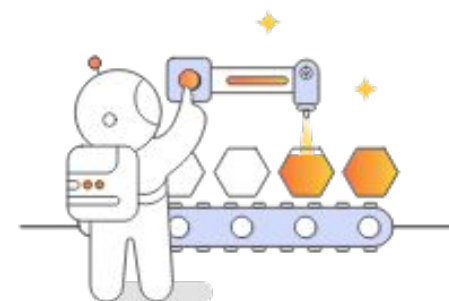




**Postman**

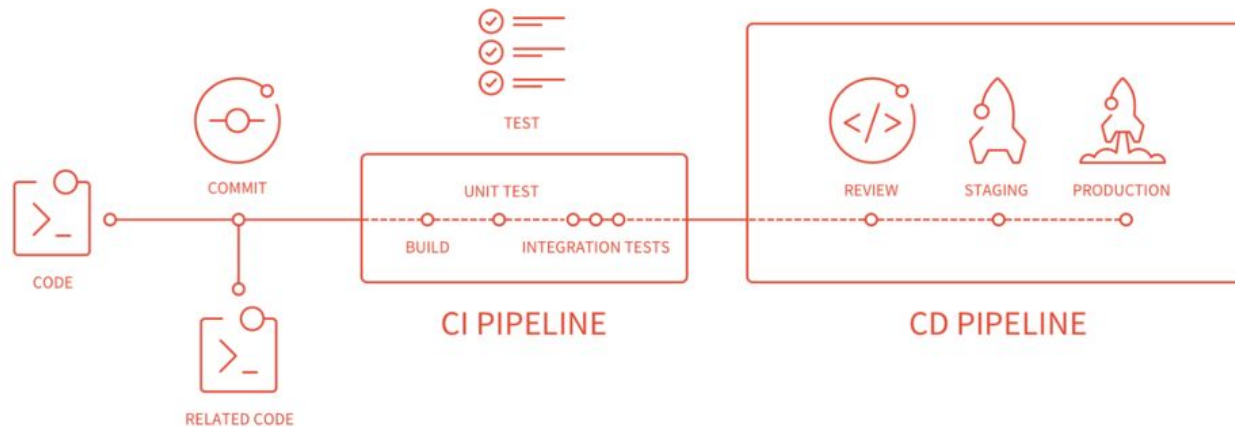


**GitlabCI**





# GitLab CI



## Postman



## GitlabCI

**Gitlab:** has its own section for Continuous Integration. For every change submitted, Gitlab is able to run a pipeline of scripts to build, test and validate the code changes. In this case, the CI will be devoted to execute automatically the requests contained inside the extracted Postman Collection, evaluating the results of each test. The status of each job will be related to the result obtained by the test.



main ▾ api-testacademy / + ▾

| Name                                                                                                                       | Last commit                |
|----------------------------------------------------------------------------------------------------------------------------|----------------------------|
|  .gitlab-ci.yml                           | Update .gitlab-ci.yml file |
|  README.md                              | Initial commit             |
|  Rick_and_Morty_postman_collection.json | Test de API                |
|  TestAcademy_postman_collection.json    | New Collection             |

## Export collection

Rick and Morty API TestAcademy will be exported as a JSON file.

Learn more about [collection formats](#) ↗

Export as:

- ☐ Collection v2
- ☒ Collection v2.1



Sending your collection to a **teammate**?  
Save time by **sharing** it instead.

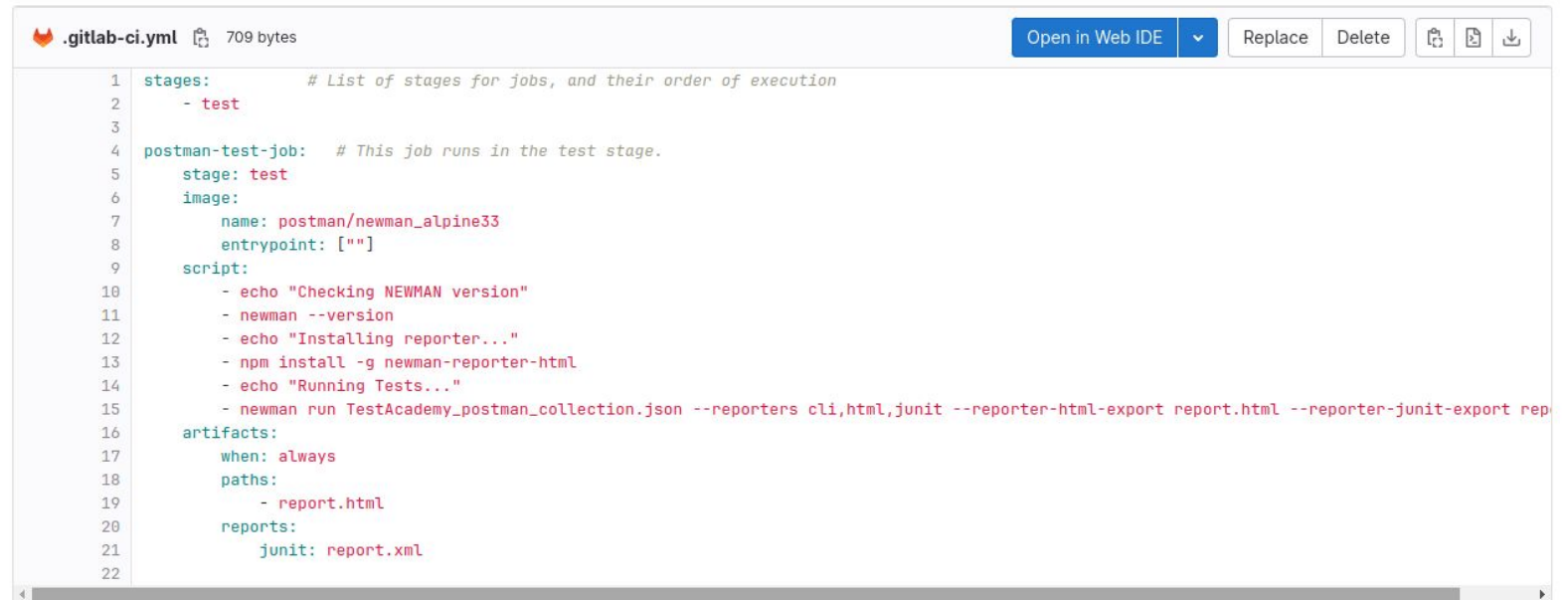
[Share Collection](#)

[Learn more](#) ↗

Cancel

Export

- The last step is to integrate what we have performed with Newman and Docker into Gitlab CI pipeline.
- The purpose is to create an automated tested solution, that could be run each time a new change is pushed on the remote repository.
- This solution could be easily integrate inside the test stage of the CICD Pipeline.



```
.gitlab-ci.yml 709 bytes
Open in Web IDE
Replace
Delete
1 stages: # List of stages for jobs, and their order of execution
2 - test
3
4 postman-test-job: # This job runs in the test stage.
5 stage: test
6 image:
7 name: postman/newman_alpine33
8 entrypoint: [""]
9 script:
10 - echo "Checking NEWMAN version"
11 - newman --version
12 - echo "Installing reporter..."
13 - npm install -g newman-reporter-html
14 - echo "Running Tests..."
15 - newman run TestAcademy_postman_collection.json --reporters cli,html,junit --reporter-html-export report.html --reporter-junit-export rep
16 artifacts:
17 when: always
18 paths:
19 - report.html
20 reports:
21 junit: report.xml
22
```



75 ✓ Status code is 200

|    |                                  |          |        |
|----|----------------------------------|----------|--------|
| 76 |                                  |          |        |
| 77 |                                  | executed | failed |
| 78 |                                  |          |        |
| 79 | iterations                       | 4        | 0      |
| 80 |                                  |          |        |
| 81 | requests                         | 12       | 0      |
| 82 |                                  |          |        |
| 83 | test-scripts                     | 24       | 0      |
| 84 |                                  |          |        |
| 85 | prerequisite-scripts             | 12       | 0      |
| 86 |                                  |          |        |
| 87 | assertions                       | 12       | 0      |
| 88 |                                  |          |        |
| 89 | total run duration: 3.2s         |          |        |
| 90 |                                  |          |        |
| 91 | total data received: 0B (approx) |          |        |
| 92 |                                  |          |        |
| 93 | average response time: 218ms     |          |        |
| 94 |                                  |          |        |

98 **Uploading artifacts...**

00:08

99 reports/: found 3 matching files

100 Uploading artifacts to coordinator... ok

id=1694875 responseStatus=201 Created token=S\_cYH

78L

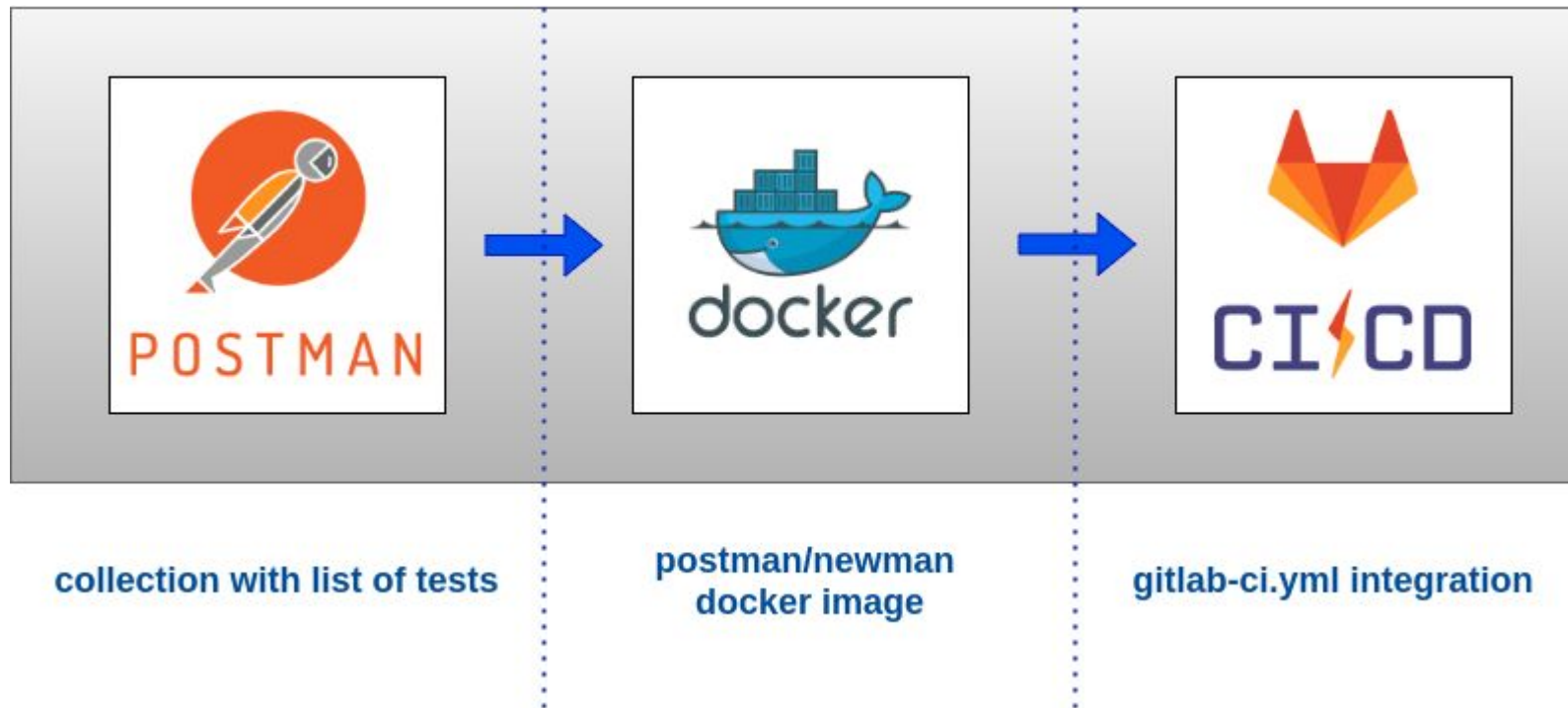
101 **Uploading artifacts...**

102 reports/newman-run-report-\*.xml: found 1 matching files

# Postman



# GitlabCI



## To Sum Up



So, to test automatically the service API response by using the features offered by the Continuous Integration tool built into Gitlab.

1. First of all the Postman Collection and its test tab
2. Then the dockerized Newman command to run all the requests automatically in sequence and
3. Finally the integration with Gitlab preparing an ad-hoc gitlab-ci.yml file to execute all the test using the CI/CD pipeline after every change pushed.

# API TESTING WITHIN CI/CD

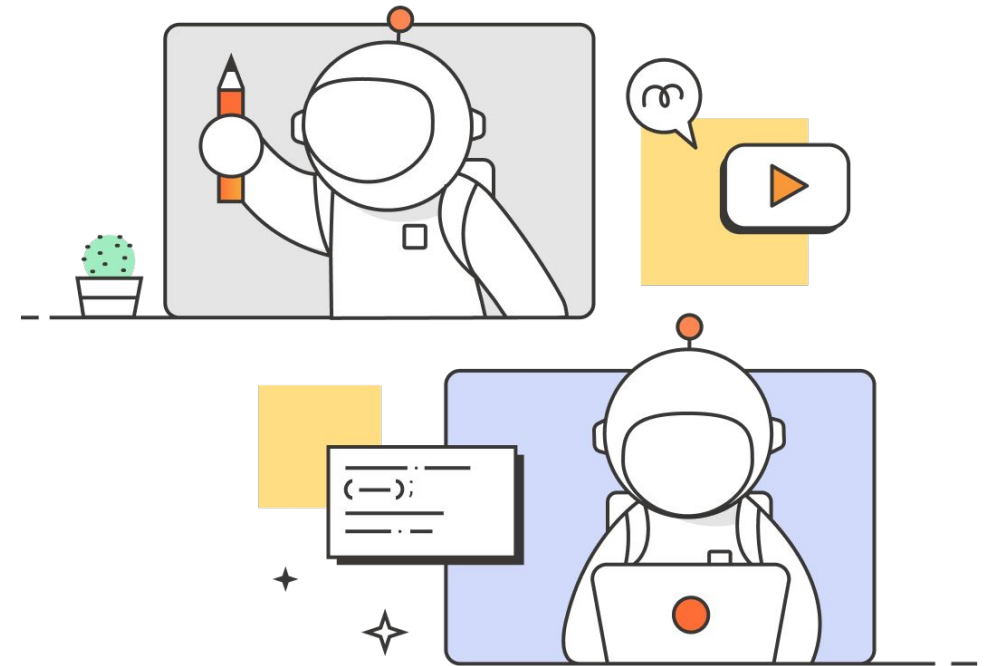
4

## Extra Perks

Gitlab Postman Integration - Visual Studio  
Postman Plugin - PostmanCanary

# POSTMAN INTEGRATION WITH GITLAB

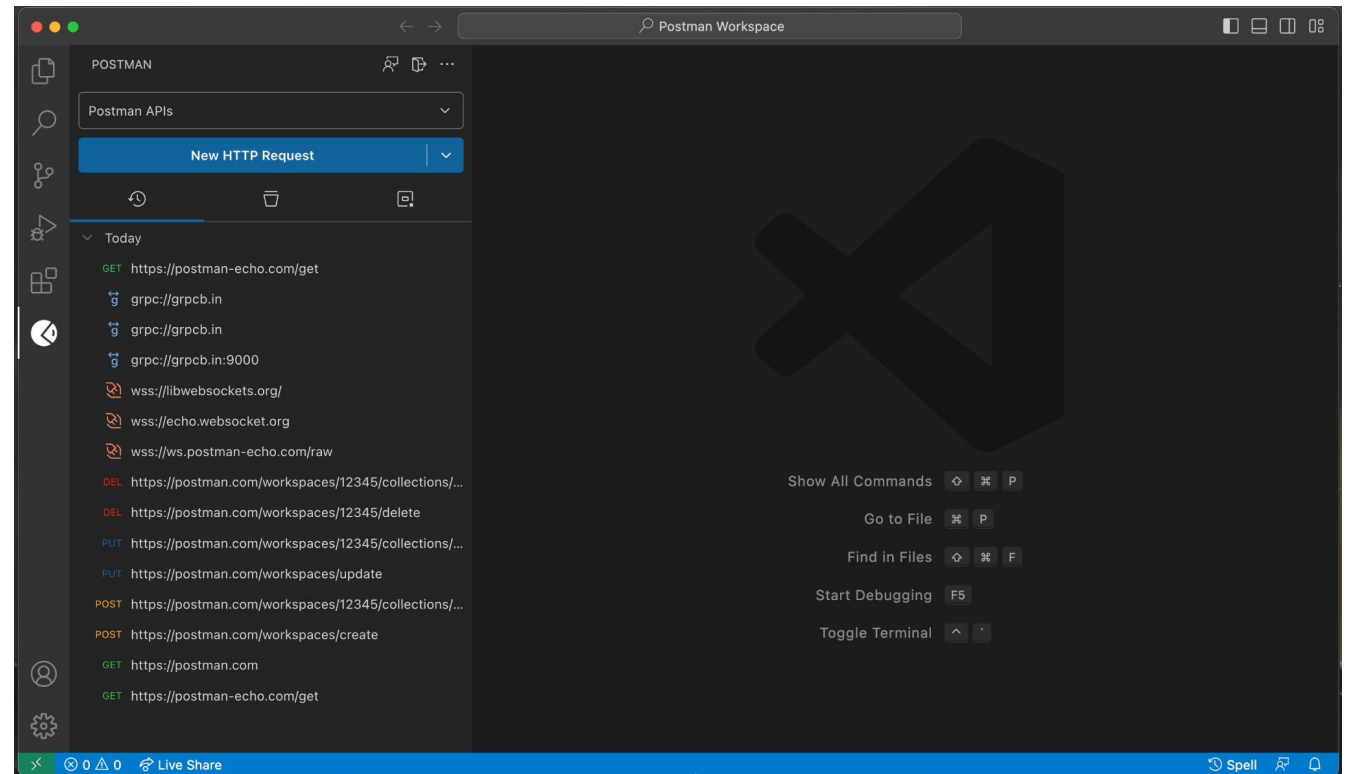
- The Postman-GitLab integration minimizes the likelihood of interacting with outdated (or even deprecated) APIs or API elements.
  - Elements created in Postman can be pushed to a user's GitLab repository, where the schema and collections can coexist alongside the source code.
1. *Create new Project in Gitlab*
  2. *Create Personal Access Token*
  3. *In Postman, at Home select Integrations*
  4. *Choose Gitlab Integration*
  5. *Authenticate and choose collection and project to be integrated.*



# POSTMAN VS CODE EXTENSION

- The Postman VS Code extension enables you to develop and test your APIs in Postman directly from Visual Studio Code.
- <https://marketplace.visualstudio.com/items?itemName=Postman.postman-for-vscode>

1. *Open Visual Studio*
2. *Install Extension (CTRL+P)*
3. *Use Postman Extension to Create and Test Requests*



# POSTMANCANARY - PERFORMANCE TESTS

<https://www.postman.com/downloads/canary/>

- Postman Canary offers a robust performance testing capability to simulate thousands of virtual users, find bottlenecks and optimize APIs to handle high traffic.

1. *Download PostmanCanary*
2. *Select to Run a Collection*
3. *Configure Performance Tab*
4. *Run and check results*

The screenshot shows the 'Performance' tab in the Postman Canary interface. At the top, there are two tabs: 'Functional' and 'Performance', with 'Performance' being the active tab. Below the tabs is a grey box with the text 'Test how your APIs perform under load' followed by a 'BETA' badge and a close icon. Below this is a description: 'Simulate real world traffic from your local machine and observe the performance of your APIs. Learn more about performance testing'. Below the description is a section titled 'Set up your performance test'. This section contains three numbered steps: 1. 'Virtual users' with a text input field containing '20'; 2. 'Test duration' with a text input field containing '10' and a 'mins' label; 3. 'Load Profile' with two radio buttons: 'Fixed' (selected) and 'Ramp up'. At the bottom left of this section is an orange 'Run' button. To the right of the configuration steps is a 'Preview' section. It features a graph with a horizontal line at the top, labeled '20 VUs' on the left and '10 mins' on the right. Below the graph, the text reads: 'Preview' and 'Simulate 20 virtual users running the collection in parallel for 10 minutes.'



# Fran Guerrero - QA Manager/Lead



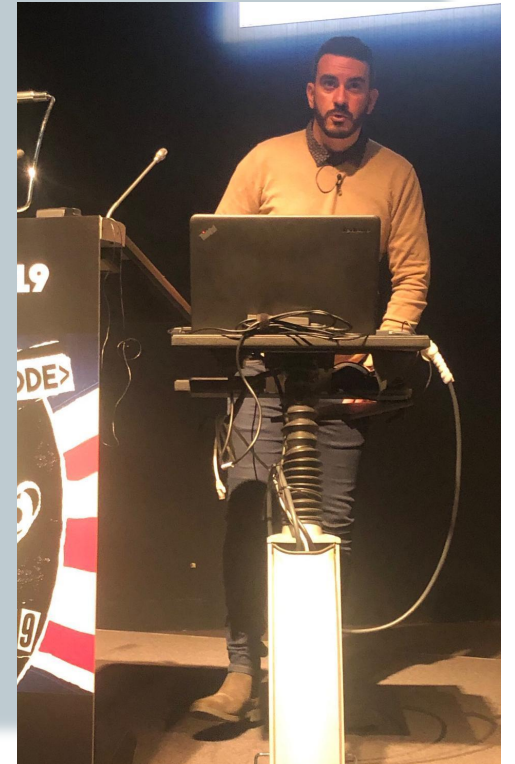
[linkedin.com/in/  
franguerrero](https://www.linkedin.com/in/franguerrero)



[@franguerreroQA](https://twitter.com/franguerreroQA)



[fran.guerrero.sanchez@gmail.com](mailto:fran.guerrero.sanchez@gmail.com)



**Testers don't break your code, they break your illusions  
about the code. - @jamesmarcusbach**

