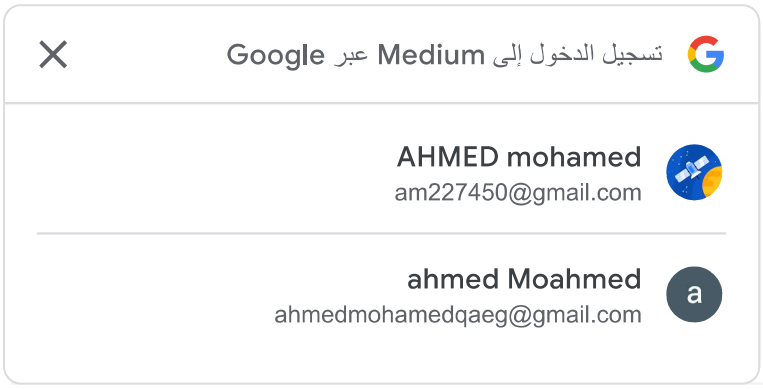


50 1

Step-by-step guide to running newman tests

Table of content:

6. Execute newman with environment variables
7. Running single folders in collections
8. Using data (csv)
9. Setting variables
10. Generating reports
11. Running a collection multiple times (Iteratations)
12. Running a collection with a custom timeout

Newman is a Postman companion with a command-line interface (CLI) that lets you run collections and tests from the command line. Your Postman collections can be easily automated and integrated into your continuous integration/continuous delivery (CI/CD) pipeline with Newman.

Newman's ability to produce reports in a variety of formats, including JSON, HTML, and JUnit, is one of its advantages. This makes it simple to track your progress over time and share test results with your team.

Newman can be used to execute individual Postman requests or even a subset of requests within a collection in addition to running collections. This adaptability makes Newman a useful asset for testing APIs and microservices.

In general, Newman is an important expansion to any Programming interface testing tool compartment. Newman can assist you in streamlining your testing and integrating it into your development workflow thanks to its adaptability, robust reporting capabilities, and ease of use.

Lets get started

— — — — — — — — — — REPOSITORY — — — — — — — — — — — — — — — — —

Use this API collection if you don't have any.
Repository:
https://github.com/alexrodriguezsoto/workspace-project
— —
—

Here's an example of how to get started with Newman commands:

1. **Install Newman by running the following command in your terminal:**

Install Node.js: Newman is a command-line tool that runs on Node.js. If you don't have Node.js installed, you need to download and install it on your system. Visit the official Node.js website (https://nodejs.org) and download the appropriate installer

```
node -v
```

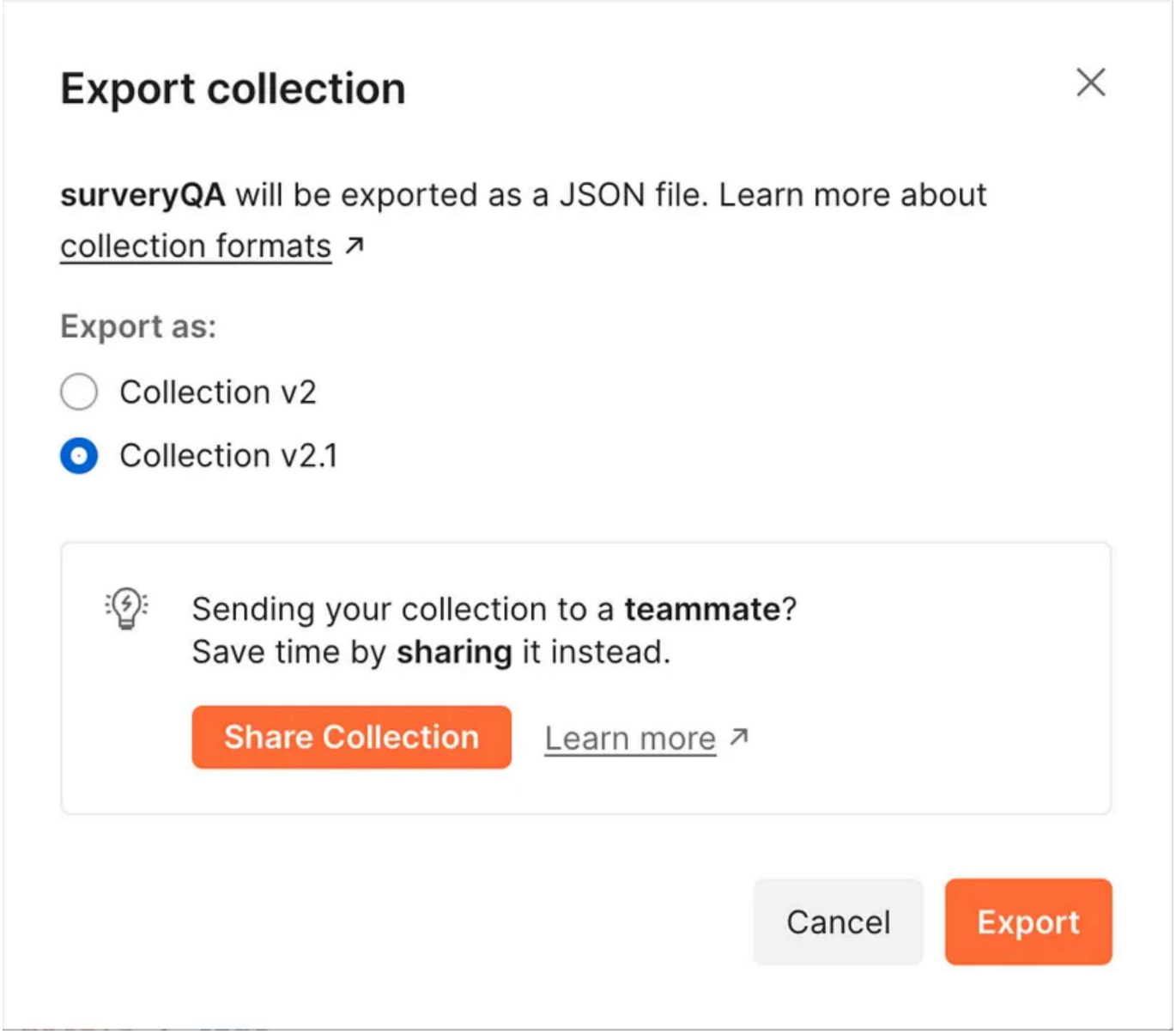
This command should display the installed version of Node.js, confirming that the installation was successful.

```
npm install -g newman
```

This command will download and install the latest version of Newman globally on your system. The `-g` flag ensures that Newman is installed as a global package, making it accessible from any directory.

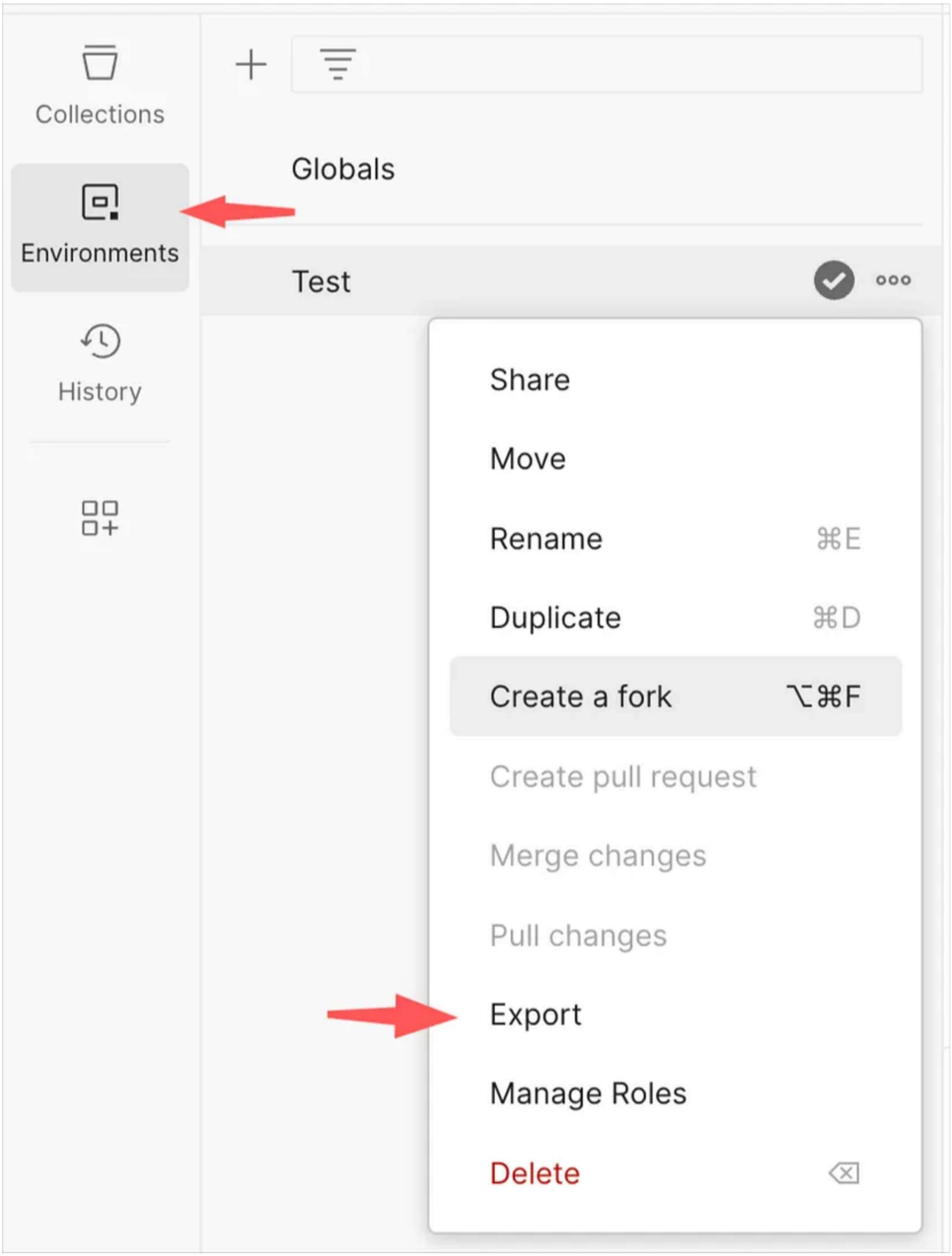
2. Create Collections

Create a JSON file of your Postman collection. In the Postman app, you can accomplish this by selecting “Collection” as the export format and clicking the “Export” button.



3. Create Environment Variables

Save your environment Variables or use the environment variables provided in the example repository


Collections
Environments
History
Globals
Test
Share
Move
Rename ⌘E
Duplicate ⌘D
Create a fork ⌥⌘F
Create pull request
Merge changes
Pull changes
Export
Manage Roles
Delete

4. **Execute Collections using newman**

To run your collection with Newman, run the following command:

```
newman run <collection.json>
```

Let's try it out using workspace-project collections

```
newman run Workspace.postman_collection.json
```

The name of your JSON file should be used in place of "<collection.json>."

5. **Results in terminal**

Your collection will be run by Newman, which will begin and display the results in your terminal. Newman will execute all of your requests and display the test results for each one by default.

Terminal Examples.

```
┌─────────────────────────┬────────────────────┬───────────────────┐
│                         │           executed │            failed │
├─────────────────────────┼────────────────────┼───────────────────┤
│              iterations │                  1 │                 0 │
├─────────────────────────┼────────────────────┼───────────────────┤
│                requests │                  9 │                 0 │
├─────────────────────────┼────────────────────┼───────────────────┤
│            test-scripts │                 18 │                 1 │
├─────────────────────────┼────────────────────┼───────────────────┤
│      prerequest-scripts │                 13 │                 0 │
├─────────────────────────┼────────────────────┼───────────────────┤
```

```
│              assertions │                43 │                 1 │
├─────────────────────────┴───────────────────┴───────────────────┤
│ total run duration: 1894ms                                      │
├─────────────────────────────────────────────────────────────────┤
│ total data received: 1.92kB (approx)
```



```
  #  failure                 detail

 1.  AssertionError          Status code is 200
                             expected response to have status code 200
but got 400
                             at assertion:0 in test-script
                             inside "users / login upload"

 2.  TypeError               Cannot read properties of undefined
(reading 'token')
                             at test-script
                             inside "users / login upload"
```

6. **Execute newman with environment variables**

By adding various options, such as specifying an environment file, using data files, or setting variables, you can customize your Newman command. Use the following command, for instance, to run your collection with a particular environment file:

```
newman run <collection.json> -e <environment.json>
```

Let's try it out using workspace-project collections

```
newman run Workspace.postman_collection.json -e QAEnv.postman_environment.json
```

The name of your environment file should be substituted for “<environment.json>.”

A few examples of how to use Newman commands include the ones listed here. For more data on the most proficient method to utilize Newman, including a full rundown of accessible choices and models, allude to the Newman documentation.

7. **Running single folders in collections**

```
newman run <collection.json> -e <environment.json> --folder "<folder-name>"
```

Let’s try it out using workspace-project collections

```
newman run Workspace.postman_collection.json  -e QAEnv.postman_environment.json
```

The name of the folder containing the request and the name of the request should be substituted for “<folder-name>.”

8. **Using data (csv)**

```
newman run <collection.json> -d <data-file.csv>
```

Replace <data-file.csv> with the name of your data file.

Let's try it out using workspace-project collections

```
newman run Workspace.postman_collection.json -e QAEnv.postman_environment.json
--folder "users" -d creden.csv

Example results:
newman
```

```
workSpace

❑ users
↳ login upload
  POST https://api.octoperf.com/public/users/login?
username=tla.jiraone@gmail.com&password=test12 [200 OK, 570B, 373ms]
  ✓  Status code is 200

↳ currentUser
  GET https://api.octoperf.com/users/current [200 OK, 837B, 97ms]
  ✓  Status code is 200 for /public/users/login
  ✓  Status code name has string OK
  ✓  verify content type is Json body for [login]
```

creden

user	pass
tla.jiraone@gmail.com	test12

9. Setting variables

Setting variables in Newman allows you to define and manage dynamic values that can be used throughout your API testing collection runs. Variables provide flexibility and enable you to customize your requests, headers, or any other part of your API tests based on different scenarios or environments.

1. Global Variables: Global variables are similar to environment variables, but they are available throughout the entire collection run. Global variables can be set using the `-g` or `--globals` flag, followed by a JSON or CSV file containing the variable key-value pairs.
2. Collection Variables: You can also define variables specific to individual requests within your collection. These variables are stored within the collection itself and can be accessed only by the associated request.

```
newman run collection.json -g globals.json
```

```
newman run <collection.json> --global-var "variable-name=value" --env-var "varia
```

```
"variables": [
  {
    "key": "baseURL",
    "value": "https://api.example.com"
  },
  {
    "key": "token",
    "value": "your-auth-token"
```

```
  }
]
```

- `--env-var "variable-name=value"` : This flag is used to set an environment variable in Newman. Environment variables are specific to a particular environment and can be accessed across different requests within the collection run. You can specify the name of the variable and its corresponding value using the format `"variable-name=value"` .

Example: `--env-var "token=your-auth-token"`

By using the `--global-var` and `--env-var` flags, you can pass custom variable values to your collection during the Newman run

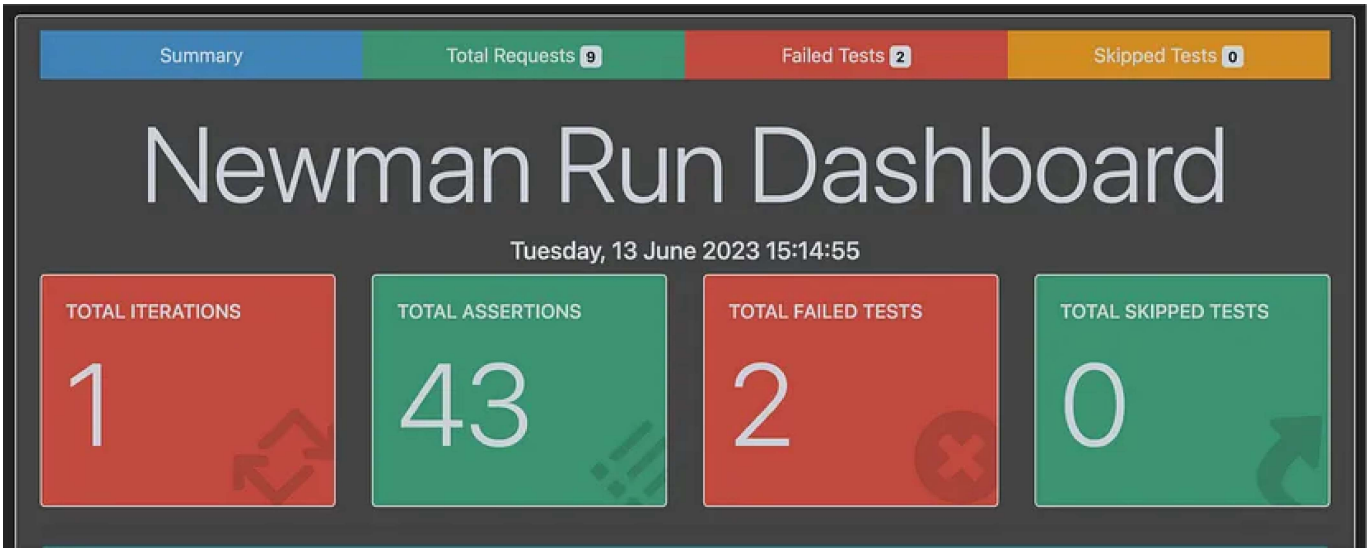
10. Generating reports

```
newman run <collection.json> --reporters cli,json,html
```

This command generates test results in CLI, JSON, and HTML formats.

Let's try it out using workspace-project collections

```
newman run Workspace.postman_collection.json
-e QAEnv.postman_environment.json
-r htmlextra --reporter-htmlextra-export ./Downloads/htmlreport.html
```



Newman Workspace.postman_collection.json: This command executes a Postman collection named Workspace.postman_collection.json using Newman.

`-e QAEnv.postman_environment.json` : The -e option specifies the environment file to use during compilation. In this case, the environment file QAEnv.postman_environment.json contains variables and values related to the QA environment. Newman uses these variables when running the collection, allowing for flexibility for different environments.

`-r htmlextra` : The -r option specifies the type of reporter used to generate the report. In this case, the htmlextra reporter is selected. This is Newman's extended HTML reporter that provides comprehensive and visually appealing HTML reporting.

`- -reporter-htmlextra-export` ./Downloads/htmlreport.html: This option specifies the file path (./Downloads/htmlreport.html) where the HTML report is exported and saved. In this example, the report is saved in the downloads folder as htmlreport.html.

An open-source reporting tool for Postman's API testing tool, Newman HTML Extra reporter offers interactive features, enhanced HTML reporting capabilities, and support for custom templates.

1. Generate a basic HTML report for a collection run:

```
newman run collection.json -r htmlextra
```

2. Generate an HTML report with additional information like environment variables and request headers:

```
newman run collection.json -r htmlextra --reporter-htmlextra-export report.html
```

3. Generate an HTML report with a custom title:

```
newman run collection.json -r htmlextra --reporter-htmlextra-export report.html
```

4. Generate an HTML report with multiple reporters:

```
newman run collection.json -r cli,htmlextra --reporter-cli-no-failures --reporte
```

11. Running a collection multiple times (Iteratations)

```
newman run <collection.json> -n <number-of-iterations>
```

```
newman run Workspace.postman_collection.json  -e QAEnv.postman_environment.json
newman
```

Let's try it out using workspace-project collections

```
┌─────────────────────────┬────────────────────┬────────────────────┐
│                         │           executed │             failed │
├─────────────────────────┼────────────────────┼────────────────────┤
│              iterations │                  2 │                  0 │
├─────────────────────────┼────────────────────┼────────────────────┤
│                requests │                 18 │                  0 │
├─────────────────────────┼────────────────────┼────────────────────┤
│            test-scripts │                 36 │                  2 │
├─────────────────────────┼────────────────────┼────────────────────┤
│      prerequest-scripts │                 26 │                  0 │
├─────────────────────────┼────────────────────┼────────────────────┤
│              assertions │                 86 │                  2 │
├─────────────────────────┴────────────────────┴────────────────────┤
│ total run duration: 2.7s                                          │
├───────────────────────────────────────────────────────────────────┤
│ total data received: 3.83kB (approx)                              │
├───────────────────────────────────────────────────────────────────┤
│ average response time: 129ms [min: 93ms, max: 435ms, s.d.: 79ms]  │
└───────────────────────────────────────────────────────────────────┘
```

- `newman run Workspace.postman_collection.json` : This command runs the Postman collection named `Workspace.postman_collection.json` using

Newman. It executes the requests and tests defined within the collection.

- `-e QAEnv.postman_environment.json`: The `-e` option specifies the environment file to be used during the collection run. In this case, the environment file `QAEnv.postman_environment.json` contains variables and values specific to the QA environment. Newman will utilize these variables while running the collection, enabling flexibility and adaptability across different environments.
- `-n 2`: The `-n` option specifies the number of iterations for the collection run. In this example, the value is set to `2`, indicating that the collection will be executed twice.

12. **Running a collection with a custom timeout**

```
newman run <collection.json> --timeout-request <timeout-in-ms>
```

Let's try it out using workspace-project collections

```
newman run Workspace.postman_collection.json
-e QAEnv.postman_environment.json --timeout-request 10
```

```
-- ERROR generated by timeout --
```

```
❏ users
↳ login upload
  POST https://api.octoperf.com/public/users/login?username=
```

```
{{user}}&password={{pass}} [errored]
    ETIMEDOUT
```

- `newman run Workspace.postman_collection.json` : This command runs the Postman collection named `Workspace.postman_collection.json` using Newman. It executes the requests and tests defined within the collection.
- `-e QAEnv.postman_environment.json` : The `-e` option specifies the environment file to be used during the collection run. In this case, the environment file `QAEnv.postman_environment.json` contains variables and values specific to the QA environment. Newman will utilize these variables while running the collection, enabling flexibility and adaptability across different environments.
- `--timeout-request 10` : The `--timeout-request` option sets the maximum time allowed for each individual request to complete. In this example, the value is set to `10` milliseconds. If a request exceeds this timeout, Newman will consider it as failed.

Newman is a powerful command-line tool for running Postman collections, providing benefits such as automation, easy integration into CI/CD pipelines, and the ability to execute tests in parallel. It helps streamline API testing and offers detailed reporting. Learn more about Newman at: https://learning.postman.com/docs/collections/using-newman-cli/command-line-integration-with-newman/

Learn more: https://quickstarts.postman.com/

> *Don’t forget to follow for more details.*

API Postman QA Automation Testing Development

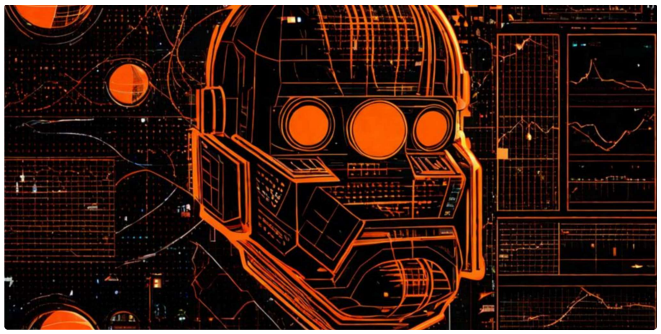
Written by Alex Rodriguez

Follow

108 Followers

IT-Consultant, Programer, Mentor/Instructor

More from Alex Rodriguez



Alex Rodriguez

Postman Visualize Reports

“Visualization is not necessary for every scenario, but when it is, here’s how you can...

5 min read · 5 days ago

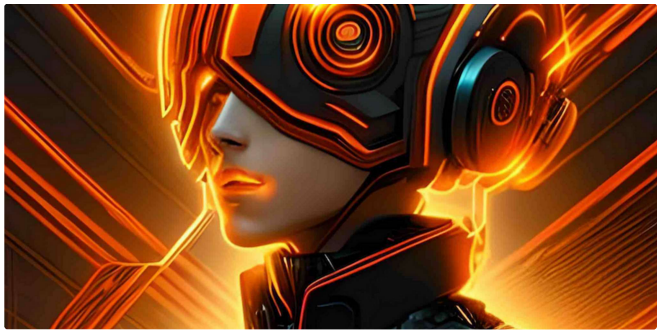
Alex Rodriguez

Postman Test Scripts

Table of Contents

8 min read · Nov 17, 2021

7 1



Alex Rodriguez

ChatGPT and Postman API Testing

OpenAI's ChatGPT is a powerful language model that can use prompts to generate text...

4 min read · Mar 3

181

59

Alex Rodriguez

Performance Test with Postman Canary

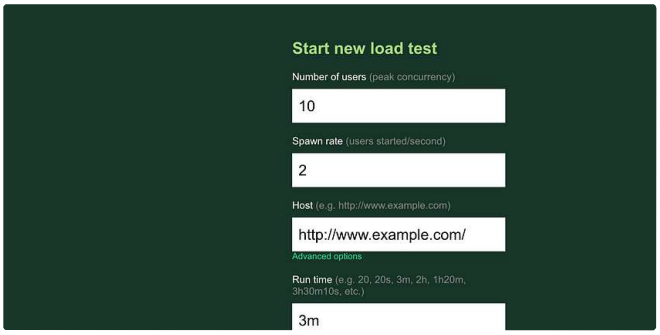
NOTE: Postman Canary is a preview version of the popular Postman API tester. It is intende...

7 min read · May 15

7

See all from Alex Rodriguez

Recommended from Medium



Jerry An in Level Up Coding

Load Testing with Locust: A Beginner's Guide

Load testing is an essential part of software development, helping to ensure that...

✨ · 3 min read · Jul 4

54

Abdullah Muhammad in Stackademic

API Testing with Postman and Node.js

Basic understanding of JavaScript, Node.js, and APIs is sufficient for working with this...

11 min read · Sep 16

1

Lists

Coding & Development

11 stories · 232 saves

It's never too late or early to start something

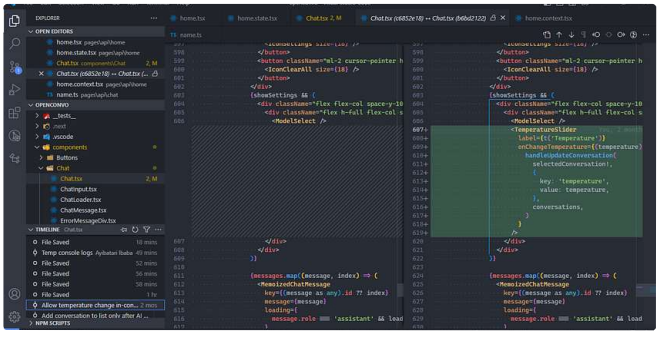
15 stories · 176 saves

Company Offsite Reading List

8 stories · 57 saves

Modern Marketing

36 stories · 203 saves



Mahdi Mallaki in ITNEXT

Replace Dockerfile with Buildpacks

Exploring the Pros and Cons of Replacing Dockerfile with Buildpacks

6 min read · Oct 15

609 7

Coding Beauty in Coding Beauty

10 essential VS Code tips & tricks for greater productivity

Boost your productivity with VS Code: discover key features to enhance your codin...

✨ · 10 min read · Aug 20

3.2K 28

Alex Rodriguez

Performance Test with Postman Canary

NOTE: Postman Canary is a preview version of the popular Postman API tester. It is intende...

7 min read · May 15

7

Berkay İbiş in Beyn Technology

JMeter vs Locust

Hello, this article will be a little more theoretical and compare Jmeter and Locust...

6 min read · May 31

62

See more recommendations