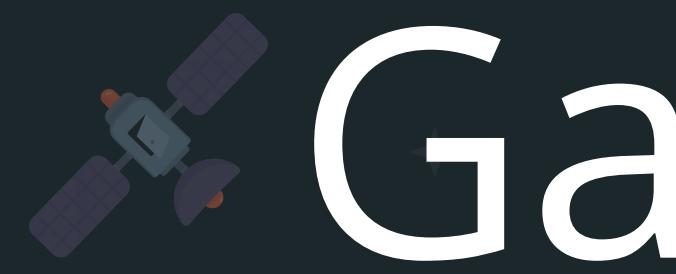




POSTMAN



# Galaxy tour workshop

San Francisco

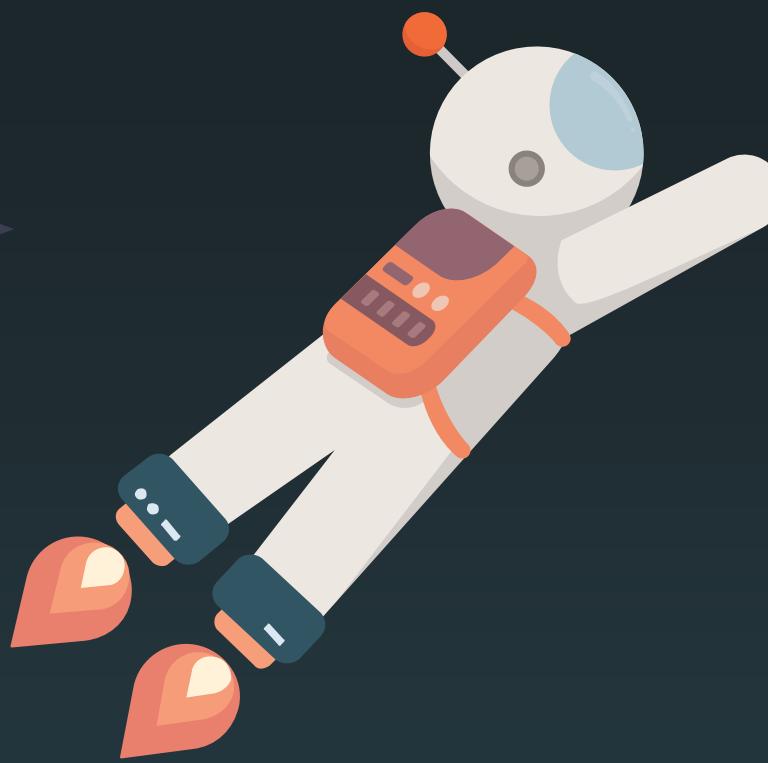


Joyce Lin

Developer Advocate Lead, Postman



@petuniaGray





POSTMAN



# Joyce Lin

Developer Advocate Lead, Postman

Postman is an API Development Environment (ADE) used by more than 10 million people.

For many teams, Postman is an everyday tool that helps people work with APIs more efficiently.



POSTMAN

# Postman app

The screenshot shows the Postman application interface. The top navigation bar includes 'New', 'Import', 'Runner', and a search bar set to 'Sushi'. The collections sidebar shows a 'Food-related' collection with a 'Geocoding' item selected. The main workspace displays a 'GET Geocoding' request with the URL: `https://maps.googleapis.com/maps/api/geocode/json?key={{googleMapsGeocodingApiKey}}&address={{address}}`. The 'Params' tab shows two query parameters: 'key' and 'address'. The response body is displayed in JSON format, showing the results for the address '49 Geary Street'.

```
1 {  
2   "results": [  
3     {  
4       "address_components": [  
5         {  
6           "long_name": "49",  
7           "short_name": "49",  
8           "types": [  
9             "street_number"  
10          ]  
11        },  
12        {  
13           "long_name": "Geary Street",  
14           "short_name": "Geary St",  
15           "types": [  
16             "route"  
17           ]  
18         }  
19       }  
20     }  
21   ]  
22 }
```



POSTMAN

# Other Postman stuff

The screenshot shows the Postman application window. In the top left, there's a search bar with 'Sushi' and tabs for 'History' and 'Collections'. Below it is a 'Trash' section containing a folder named 'Sushi Selector' with four requests: 'Geocoding', 'Place Search', 'Get Twitter Owner ID', and 'Direct message to Twitter'. The main workspace is titled 'Food-related' and contains several large, semi-transparent orange text overlays representing different Postman features:

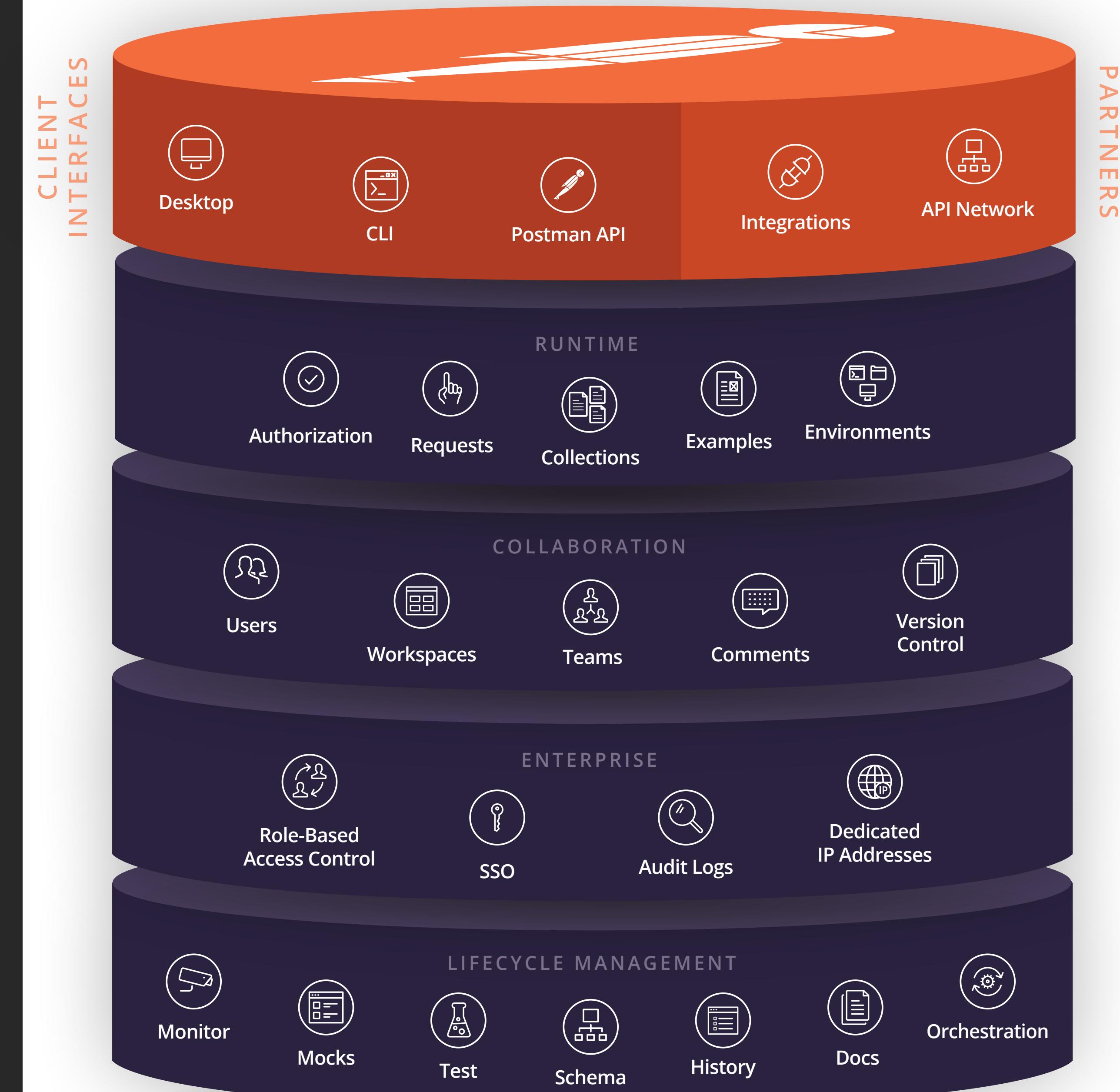
- Postman API
- Workspaces
- proxies
- authentication
- Newman
- collaboration
- environments
- mock servers
- open-source projects
- CLI
- monitors
- Interceptor
- runtime
- converters
- automation
- variables
- sessions
- version control
- API Network
- Postman sandbox
- integrations
- Bootcamp
- Build
- Browse

The bottom right corner of the workspace shows a JSON response from a 'Geocoding' request, which includes address components like '49 Geary Street' and 'San Francisco, CA'.



POSTMAN

# Postman as a Platform





POSTMAN

# Pre-requisites

Download Postman

Sign in to Postman

Get ShipEngine  
sandbox API key





# The Mission

2:15-3:45 **Prepare for launch**  
**Final countdown**

3:45-4:00 **Break**

4:00-5:15 **Liftoff**  
**Activate thrusters**

5:15-6:15 **Side missions**  
**Q&A**  
**Networking**





POSTMAN

# Who has?

Used Postman

Written a test

Run multiple requests



@petuniaGray





POSTMAN

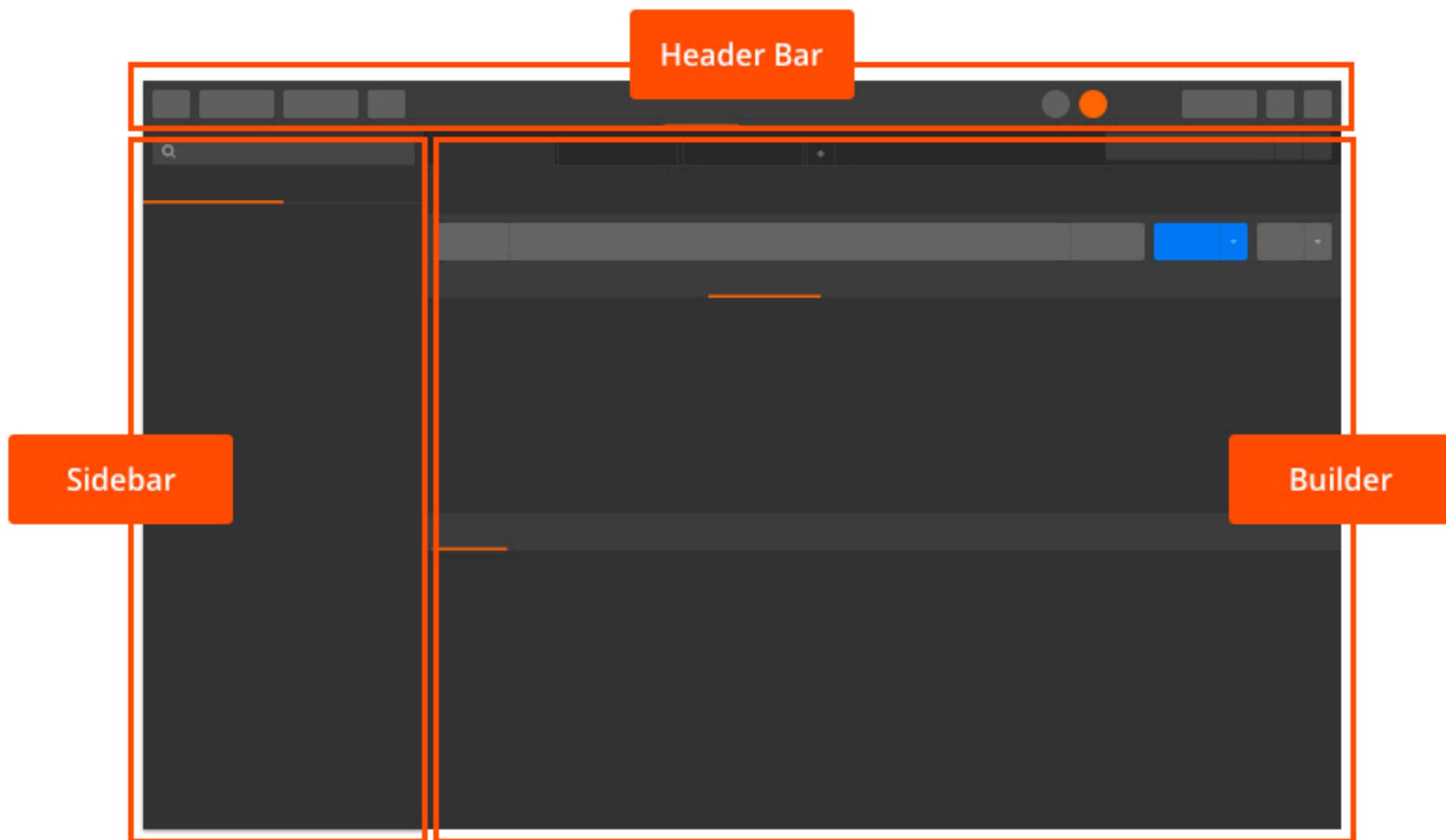
# HANDS ON

# Prepare for Launch: Part 1



# Orientation

- Request builder / Response viewer
- Collections, folders, and workspaces
- Build (vs. Browse)





POSTMAN

# Sending an HTTP request

- Request builder / Response viewer
- Variables
- Productivity
  - Header presets, bulk edit
  - Auth and inheritance
  - Console
  - Visualizer
  - Code generation

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' and 'Collections'. Under 'Collections', there's a folder named 'ShipEngine Walkthrough' containing 39 requests. The requests are categorized into 'Carrier Integrations' (List your carriers, Get a specific carrier, List a carrier's options, List a carrier's services, List a carrier's packaging, Add funds to a carrier), 'Shipping Costs' (Create a label from scratch, Create a label from a rate, Create a label from a shipment, Create a return label, Create & download a label, Label messages (reference), Void a label). A modal window titled 'EDIT COLLECTION' is open over the main interface. It has tabs for 'Name' (set to 'ShipEngine Walkthrough'), 'Description', 'Authorization' (which is selected and highlighted in orange), 'Pre-request Scripts', 'Tests', and 'Variables'. The 'Authorization' tab contains a sub-section for 'TYPE' where 'API Key' is selected. Below it, a note says: 'This authorization method will be used for every request in this collection. You can override this by specifying one in the request.' A tooltip appears, stating: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)'. The 'Key' field is 'API-Key' and the 'Value' field is '{{API\_KEY}}'. A dropdown 'Add to' shows 'C API\_KEY' with three entries: 'INITIAL' (TEST\_jZPqxRz5Cpn/f9gqHwmKrCK9cxNcHkp3gYbD9TSqHtg), 'CURRENT' (TEST\_jZPqxRz5Cpn/f9gqHwmKrCK9cxNcHkp3gYbD9TSqHtg), and 'SCOPE' (Collection). At the bottom of the modal are 'Cancel' and 'Update' buttons.



POSTMAN

# Writing a test

- Test snippets
- Assertions
- Chai.js
- pm.\*

The screenshot shows the Postman application interface. On the left, the sidebar displays a collection named "ShipEngine Walkthrough" containing various requests like "List your carriers", "Get a specific carrier", and "List a carrier's options". The "Get a specific carrier" request is currently selected. The main workspace shows a test script for this request:

```
1 pm.test("Status code is 200", function () {
2     pm.response.to.have.status(200);
3 });
4
5 let response = pm.response.json();
6
7 if (response) {
8     const template =
9         <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/meyer-reset/2.0/reset.min.css">
```

The "Tests" tab is active. Below the script, the "Test Results" section shows a single test result:

PASS Status code is 200

At the bottom right, status information is displayed: Status: 200 OK, Time: 1115ms, Size: 5.77 KB, and a "Save Response" button.



POSTMAN

# Creating a collection

- Organization
- Conditional workflows
- Collaboration
- Advanced capabilities
  - Documentation
  - Test suites
  - Mock servers

The screenshot shows the Postman application interface. The left sidebar displays a search bar with 'Postman Echo' and tabs for 'History', 'Collections' (which is selected), and 'APIs BETA'. Below these are sections for 'Postman Echo' (37 requests) and 'DNS Checker' (14 requests). Under 'Postman Echo', there are categories like 'Request Methods', 'Headers' (selected), 'Authentication Methods', 'Cookie Manipulation', 'Utilities', 'Utilities / Date and Time', 'Utilities / Postman Collection', and 'Auth: Digest'. The main workspace shows a request for 'Request Headers' with a GET method and URL 'https://postman-echo.com/headers'. The 'Headers' tab is selected in the request details. The response body is displayed in JSON format:

```
1 {  
2   "headers": {  
3     "x-forwarded-proto": "https",  
4     "host": "postman-echo.com",  
5     "accept": "*/*",  
6     "accept-encoding": "gzip, deflate",  
7     "cache-control": "no-cache",  
8     "cookie": "sails.sid=s%3A5pRXlqLeHNexLKK0MLFbyyGMAu3WLGkQ.spZ6We8vY7ftELsnCX028szYznjchPYsHWQbFIgW  
Iro"  
}
```



POSTMAN

# Prepare for launch - recap

Requests

Collections

Variables

Tests





POSTMAN

# Prepare for launch - Resources

API Network

Chai.js



@petuniaGray



POSTMAN

# HANDS ON

# Final countdown: Part 2



POSTMAN

# Advanced Scripts and Tests



POSTMAN

# Advanced scripts and tests

- Postman sandbox
- Conditional logic
  - `postman.setNextRequest()`
- Execution
- Collection-, folder-, request-level

The screenshot shows the Postman application interface. On the left, the sidebar displays collections: 'ShipEngine Walkthrough' (39 requests), 'Carrier Integrations' (with sub-options like 'List your carriers', 'Get a specific carrier', 'List a carrier's options', etc.), and other categories like 'Shipping Costs', 'Shipping Labels', etc. The main panel shows a list of requests under the 'Carrier Integrations' section. One request, 'Get a specific carrier', is highlighted. The right side of the screen shows the request details for this specific GET request to 'https://api.shipengine.com/v1/carriers/{{stamps\_com}}'. The 'Tests' tab is active, containing the following JavaScript code:

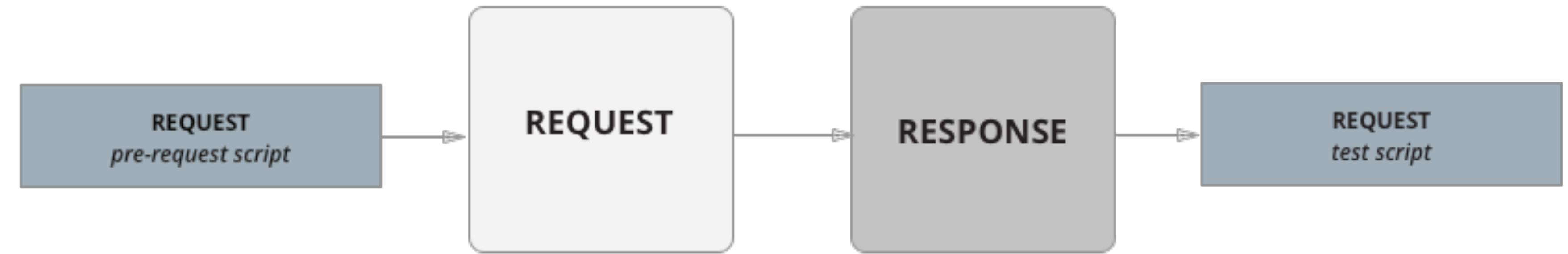
```
1 let response = pm.response.json();
2
3 if (response.packages.length > 5) {
4     // continue to next request
5     console.log("Number of packages: " + response.packages.length);
6 } else {
7     // stop here
8     console.log("Terminating...");
9     postman.setNextRequest(null);
10 }
11
```

Below the code, there are tabs for 'Body', 'Cookies', 'Headers (7)', and 'Test Results'. A message at the bottom right says 'There are no tests for this request'.



POSTMAN

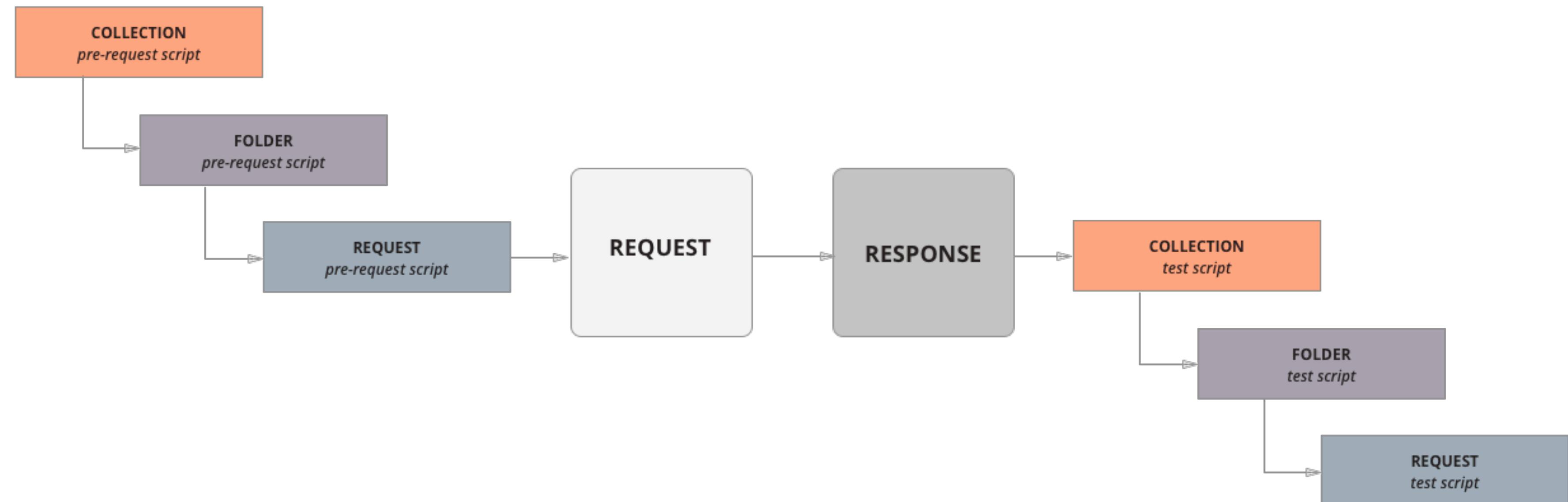
# Execution order





POSTMAN

# Execution order





POSTMAN

# Collection Runner

- Configuration options
- Looping through a data file
- Run results and history

The screenshot shows the Postman Collection Runner interface. On the left, there's a sidebar with a search bar and a tree view of collections: 'ShipEngine Walkthrough' (expanded), 'Carrier Integrations', 'Shipping Costs', 'Shipping Labels', 'Address Validation', and 'Address Parsing'. Below this are configuration settings: 'Environment' set to 'shipEngine', 'Iterations' set to '1', 'Delay' set to '0 ms', 'Data' with a 'Select File' button, and several checkboxes: 'Save responses' (unchecked), 'Keep variable values' (checked), 'Run collection without using stored cookies' (unchecked), and 'Save cookies after collection run' (checked). At the bottom is a large blue button labeled 'Run ShipEngine W...'. On the right, under 'RUN ORDER', is a list of 20 items, each with a checkmark, indicating the sequence of requests:

- ✓ ➔ GET List your carriers
- ✓ ➔ GET Get a specific carrier
- ✓ ➔ GET List a carrier's options
- ✓ ➔ GET List a carrier's services
- ✓ ➔ GET List a carrier's packaging
- ✓ ➔ PUT Add funds to a carrier
- ✓ ➔ POST Get rate estimates (minimal)
- ✓ ➔ POST Get rate estimates (detailed)
- ✓ ➔ POST Compare rates
- ✓ ➔ POST Create a shipment
- ✓ ➔ POST Compare rates for a shipment
- ✓ ➔ GET Get previously-quoted rates for a shipment
- ✓ ➔ POST Create a label from scratch
- ✓ ➔ POST Create a label from a rate
- ✓ ➔ POST Create a label from a shipment
- ✓ ➔ POST Create a return label
- ✓ ➔ POST Create & download a label
- ✓ ➔ POST Label messages (reference fields)
- ✓ ➔ PUT Void a label



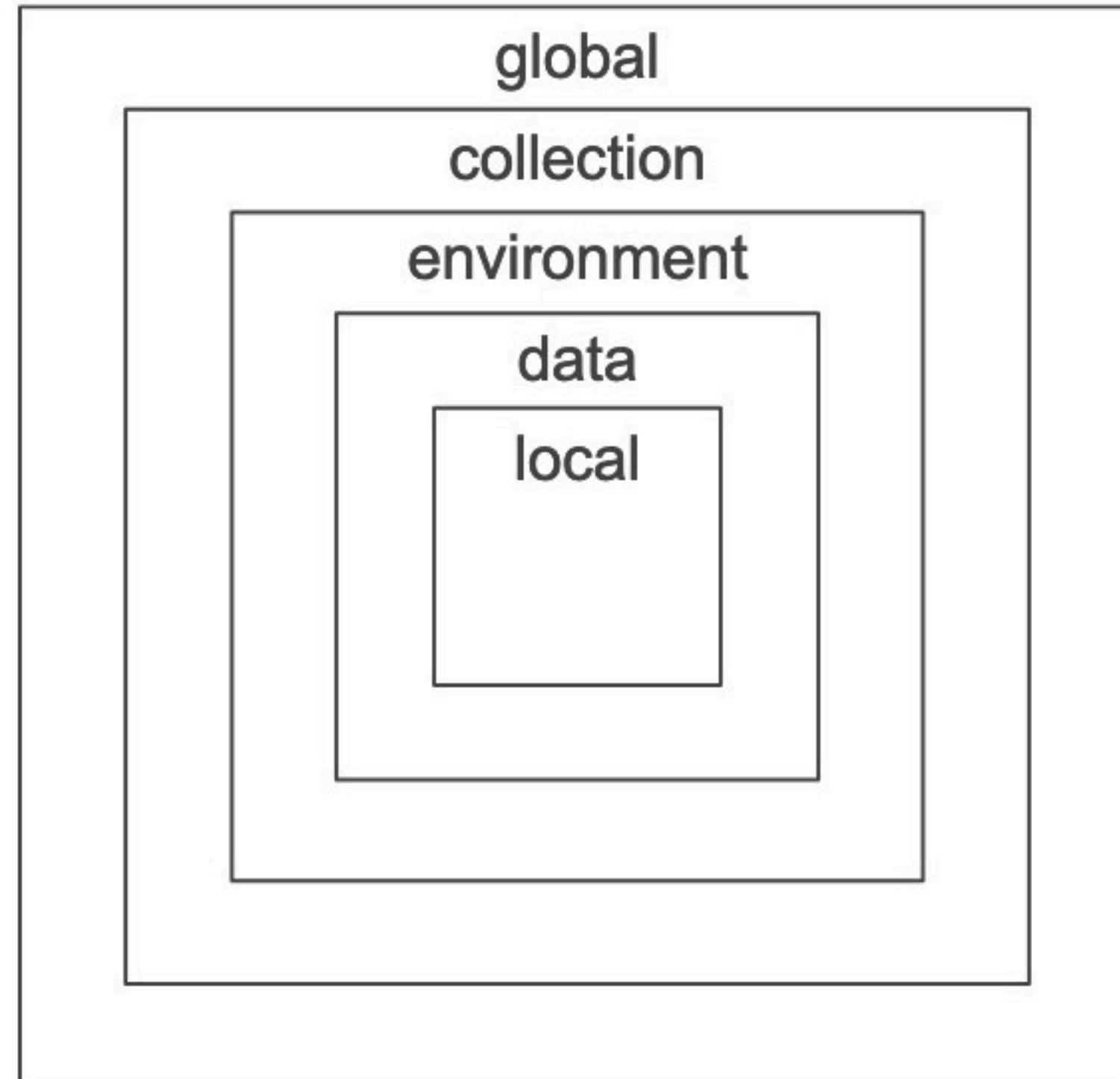
POSTMAN

# VARIABLES



# Variables

- Keep code DRY
- Configuration
- Handle secrets
- Enable passing data



Development

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	base_url	https://httpbin.org	https://httpbin.org			
	Add a new variable					



POSTMAN

# Environment

- For server config
- Security
- Can decouple from collection

The screenshot shows the Postman application interface. In the center, a modal window titled "MANAGE ENVIRONMENTS" is open, specifically the "Add Environment" tab. The environment name "mySecrets" is entered in the input field. A table lists variables with their initial and current values:

VARIABLE	INITIAL VALUE <small>i</small>	CURRENT VALUE <small>i</small>
production_url	https://www.getpost...	https://www.getpostman.com
mySuperSecretToken	stopLookingAtMySecr...	stopLookingAtMySecrets
Add a new variable		

Below the table, a note explains the behavior of variables: "Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value." It includes a "Change" link and a "Learn more about variable values" link. At the bottom right of the modal are "Cancel" and "Add" buttons.

The background of the Postman interface shows a collection named "ShipEngine Walkthrough" with 39 requests, including endpoints for carrier integrations, shipping costs, and tracking.



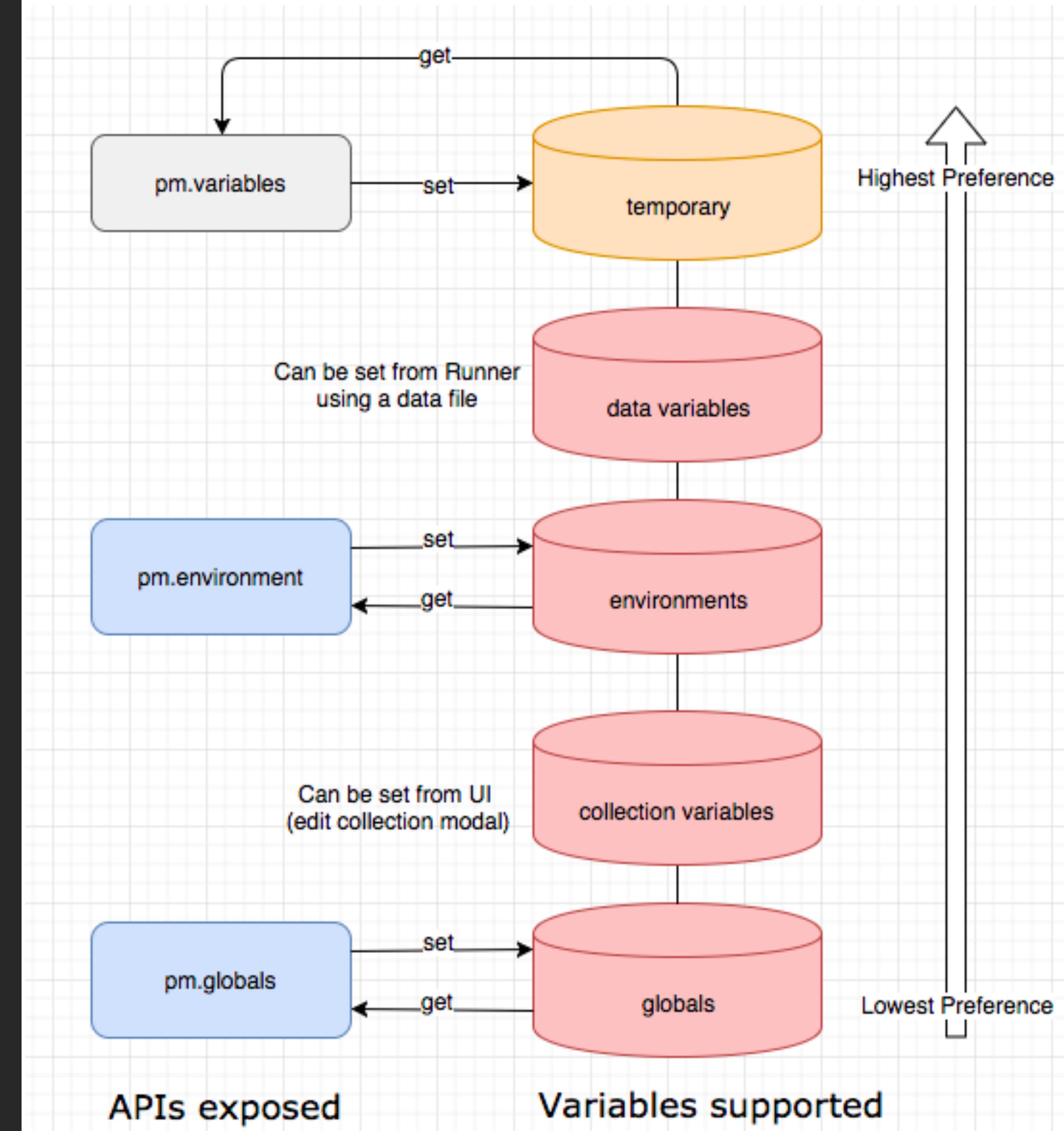
# Data variables

```
1  [ ]  
2  { }  
3  "City": "Vancouver",  
4  "Ramen": 100  
5  },  
6  {  
7  "City": "San Francisco",  
8  "Ramen": 84  
9  },  
10 {  
11  "City": "Singapore",  
12  "Ramen": 79  
13 },  
14 {  
15  "City": "Yonkers",  
16  "Ramen": 70  
17 },  
18 {  
19  "City": "Austin"
```



# Considering variables

- Choosing variables
- Setting and Getting variables (in text areas vs. scripting areas)
- Variable persistence





POSTMAN

# Moar variables

- Dynamic
- Session

The screenshot shows the Postman interface with two tabs at the top: "POST httpbin.org/post" and "POST httpbin.org/post?id={{\$guid}}". The second tab is active. Below it, the method is set to "POST" and the URL is "httpbin.org/post?id={{\$guid}}". Under the "Params" tab, there is a table with one row: "id" under "KEY" and "Value". A dropdown menu is open over the "Value" field, listing several global variables: "\$guid", "\$randomAbbreviation", "\$randomAbstractImage", "\$randomAdjective", and "\$randomAlphaNumeric". The first item, "\$guid", has a cursor icon over it. To the right of the dropdown, status information is shown: "INITIAL A v4 style guid", "CURRENT", and "SCOPE Global".

Development			...	Persist All	Reset All	
	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	base_url	https://httpbin.org	https://httpbin.org			
Add a new variable						

Dynamic variables in Mock Servers



# Final countdown - recap

Writing advanced scripts  
and tests

Running a collection

Lots and lots of variables





# Final countdown - resources

Intro to writing tests

Extract data to chain  
requests

Postman sandbox docs

Working with data files:  
ramen

Dynamic variables in mock  
servers





POSTMAN

# BREAK



POSTMAN

# HANDS ON Liftoff



POSTMAN

# Automation

- Collection Runner
  - For local development and exploration
  - Shared history of collection runs in a team workspace

The screenshot shows the Postman Collection Runner interface. At the top, it says "Collection Runner" and "Galaxy". On the left, there's a sidebar with a search bar and a list of "All Collections" including "ShipEngine Walkthrough", "APOD", "How to use the Postman Console", "Intro to writing tests - with examples", and "Postman API". Below the sidebar are configuration options: "Environment" set to "No Environment", "Iterations" set to 1, "Delay" set to 0 ms, "Data" with a "Select File" button, and several checkboxes: "Save responses" (unchecked), "Keep variable values" (checked), "Run collection without using stored cookies" (unchecked), and "Save cookies after collection run" (checked). On the right, there's a "Recent Runs" section with three entries: "ShipEngine Walkthrough" (shipEngine) from Today, 10:49 am (PASSED), "ShipEngine Walkthrough" (shipEngine) from Today, 10:20 am (PASSED), and "ShipEngine Walkthrough" (shipEngine) from Today, 10:19 am (PASSED).

Run ID	Collection	Status	Timestamp
1	ShipEngine Walkthrough shipEngine	PASSED	Today, 10:49 am
2	ShipEngine Walkthrough shipEngine	PASSED	Today, 10:20 am
3	ShipEngine Walkthrough shipEngine	PASSED	Today, 10:19 am



POSTMAN

# Automation

- Newman
  - From CLI
  - For CI/CD
  - As a library

The screenshot shows a terminal window titled "postcon — -bash — 101x35" with two tabs: "~/repos/postcon — -bash" and "~/repos/postman-docs — -bash". The main pane displays Newman test results for a "Postman Echo" collection. The output is color-coded: green for successful tests and yellow for informational messages. The results include various HTTP requests (GET, POST, PUT, PATCH, DELETE) and their corresponding response status codes, sizes, and execution times.

```
4.5.6
~/.nvm/versions/node/v12.13.1/bin/newman[10:44] ~$ newman run Postman\ Echo.postman_collection.json
newman

Postman Echo

Request Methods
└ GET Request
  GET https://postman-echo.com/get?foo1=bar1&foo2=bar2 [200 OK, 628B, 766ms]
    ✓ response is ok
    ✓ response body has json with request queries

└ POST Raw Text
  POST https://postman-echo.com/post [200 OK, 660B, 612ms]
    ✓ response is ok
    ✓ response body has json with request body

└ POST Form Data
  POST https://postman-echo.com/post [200 OK, 769B, 229ms]
    ✓ response is ok
    ✓ response body has json with form data

└ PUT Request
  PUT https://postman-echo.com/put [200 OK, 657B, 133ms]
    ✓ response is ok
    ✓ response body has json with form data

└ PATCH Request
  PATCH https://postman-echo.com/patch [200 OK, 782B, 542ms]
    ✓ response is ok
    ✓ response body has json with form data

└ DELETE Request
  DELETE https://postman-echo.com/delete [200 OK, 785B, 95ms]
```



POSTMAN

# Automation

- Monitors
  - From Postman cloud
  - Pre-built integrations for alerting and resolution

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'New', 'Import', 'Runner', 'Galaxy', 'Invite', and a refresh button. On the left, a sidebar lists various collections: 'ShipEngine V', 'APOD', 'How to use t', 'Intro to writing', 'Postman AP', 'Postman Echo', and 'Working with'. The main area is titled '2. Configuration' under a 'Select requests to monitor' section. It includes fields for 'Monitor name' (set to 'Postman Echo'), 'Version Tag' (set to 'CURRENT'), 'Use an environment (optional)' (set to 'shipEngine'), 'Monitor run frequency' (set to 'Every Day at 10:00 AM'), and 'Regions' (with 'Automatically Select Region' selected). A 'Show me how' button is visible at the bottom right of the configuration panel.



POSTMAN

# Documentation

- For developer experience and on boarding
- Markdown
  - Descriptions for collection, folder, request, params
- Screenshots

The screenshot shows a browser window titled "ShipEngine Walkthrough" from "documenter.getpostman.com". The URL is "documenter.getpostman.com/view/305204/SW7XbA6V?version=latest#intro". The page content is as follows:

## SHIPENGINE WALKTHROUGH

### Introduction

- ▶ [Carrier Integrations](#)
- ▶ [Shipping Costs](#)
- ▶ [Shipping Labels](#)
- ▶ [Address Validation](#)
- ▶ [Address Parsing](#)
- ▶ [Tracking](#)
  - [GET Track a package](#)
  - [GET Track using a label ID](#)
  - [POST Setup a tracking webhook](#)
  - [GET List your webhooks](#)
  - [POST Subscribe to tracking updates for package](#)
  - [POST Unsubscribe from tracking updates](#)

## ShipEngine Walkthrough

This collection is a walkthrough of basic ShipEngine functionality. Learn how to:

- Create and download shipping labels
- Calculate shipping costs for any package
- Compare rates across UPS, FedEx, USPS and other carriers
- Track packages in real-time or on-demand
- Parse and validate mailing addresses for any country on Earth!

**NOTE:** This collection includes a [collection variable](#) named `API_KEY`. Set this variable to [your ShipEngine sandbox key](#) so you can send API requests and see real results right in Postman!

## Carrier Integrations

Let's start off by exploring some carrier integrations. We'll see how to find out what carriers are available, what capabilities they support, and what services they offer.

### GET List your carriers

<https://api.shipengine.com/v1/carriers>

This request returns a list of all the carrier accounts that you have connected to ShipEngine

**Example Request**  
List your carriers  
`curl --location --request GET 'https://api.shipengine.com/v1/carriers'`

**Example Response**



POSTMAN

# Documentation

- For discovery
  - Privately as workspaces
  - Publicly with Postman API Network and templates

The screenshot shows the Postman application interface. The top navigation bar includes 'New', 'Import', 'Runner', and a user icon. A dropdown menu is open over the 'New' button, showing options: 'Create New', 'Templates', and 'API Network'. The 'API Network' option is selected and highlighted with a red border. The main content area is titled 'API Network' and contains a sub-header: 'Browse a directory of public APIs curated by Postman - and get started with one click. [Add to the API Network](#)'. Below this is a search bar and a 'Sort by: Default' dropdown. The central part of the screen displays several featured API categories in cards, each with a logo and some descriptive text:

- DEVNET** (Cisco DevNet): APIs: 10, Views: 10k+
- imgur**: APIs: 1, Views: 5k+
- Cisco DevNet**: APIs: 10, Views: 10k+
- Imgur**: APIs: 1, Views: 5k+
- PayPal Here**: APIs: 1, Views: 5k+
- Auth0**: APIs: 1, Views: 5k+

At the bottom of the interface, there is a code editor window showing a JSON response with three lines of code:

```
1  {
2    "carrier_id": "se-159345",
3    "carrier_code": "stamps_com",
```

Below the code editor are various toolbars and status indicators.



POSTMAN

# Workspaces

- Organization
  - Personal
  - Team
- Private ([Enterprise](#))
- Collaboration
  - Roles and permissions

The screenshot shows the Postman application window. The top navigation bar includes 'New', 'Import', 'Runner', 'My Workspace', 'Invite', and various status icons. Below the navigation is a menu bar with 'APIs BETA', 'Collections', 'History', 'Environments', 'Monitors', 'Mocks', 'Integrations', and 'Activity'. A search bar and a sorting dropdown are also present. Two API collections are listed:

- Airtable to SurveyMonkey loader** (You) - Includes a 'Share' button and a three-dot menu.
- Automate publishing your API** (You) - Includes a 'Share' button and a three-dot menu. Below this collection, a note states: "This collection is an example of how to automatically generate your documentation in Postman if you're using an API format like OpenAPI (previously called Swagger)." A table below the note lists environment variables:

Environment variables	Required	Description
postman_api_key	required	Generate your key from the <a href="#">Postman web dashboard</a>
collection_uid	required	Use the <a href="#">Postman API</a> to identify your collection's ID
environment_uid	optional	Use the <a href="#">Postman API</a> to identify your environment's ID
...		



# Other collab

- RBAC
- Versioning
- Forking and merging
- Global variables
- History of requests and collection runs
- Integrations

The screenshot shows the Postman application interface. In the center, a modal dialog box titled "FORK COLLECTION" is open. The dialog contains the following text:

Forking a collection creates a new collection with a reference to the original, allowing you to merge or pull changes between them.

**Fork label**

This will be used to identify this fork, and distinguish it from the parent and other related collections.

**Add this to a workspace**

Galaxy

You can always add the fork to another workspace later.

At the bottom right of the dialog are two buttons: "Cancel" and "Fork collection".

Below the dialog, the main Postman interface is visible, showing the "Collections" tab selected in the navigation bar. A list of collections is displayed, including "ShipEngine Walkthrough", "APOD", "How to use the Postman Cons...", "Intro to writing tests - with exa...", "Postman API", "Postman Echo", "ShipEngine Walkthrough", and "Working with GraphQL".



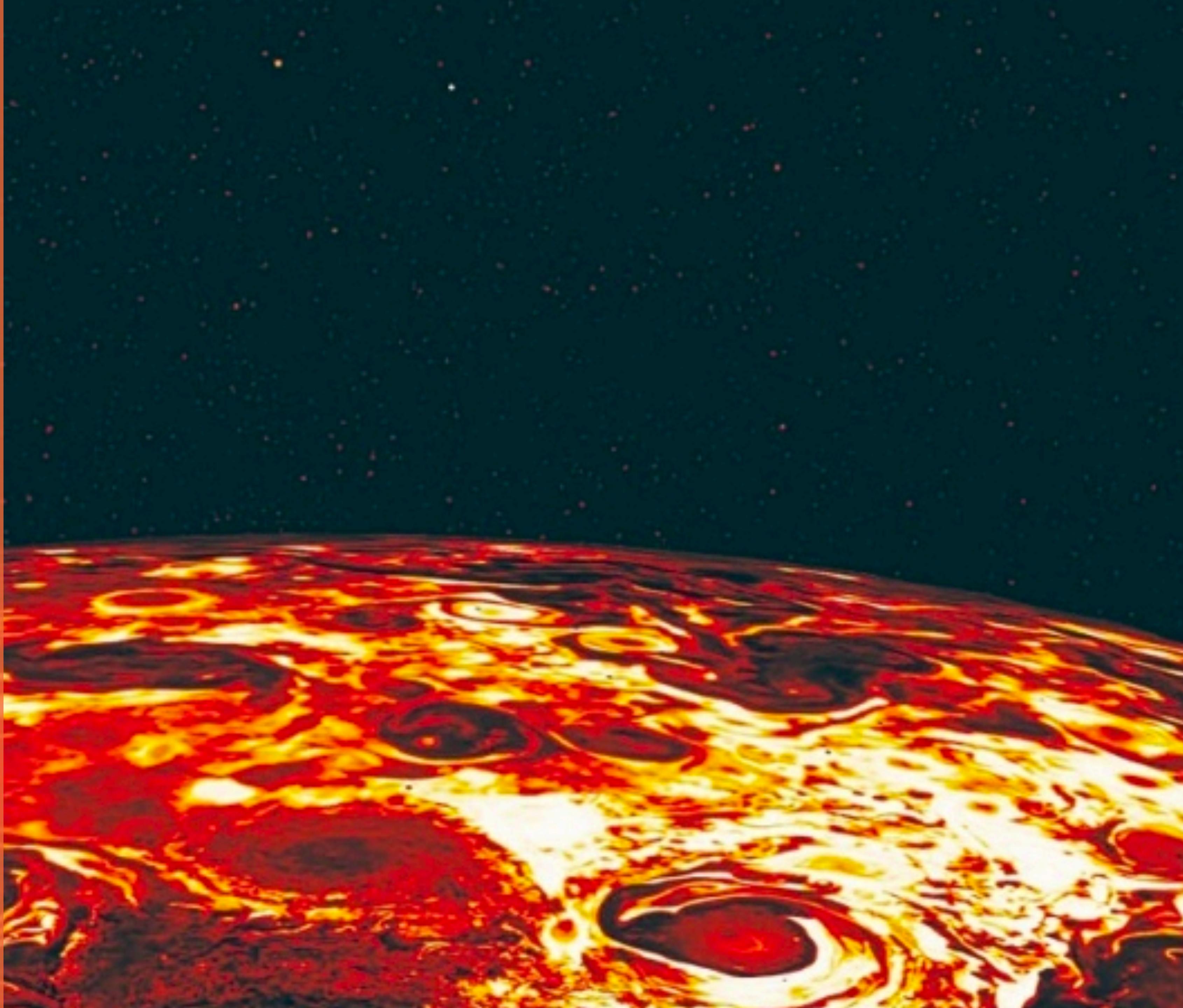
POSTMAN

# Liftoff - recap

Automation

Documentation

Collaboration



@petuniaGray



POSTMAN

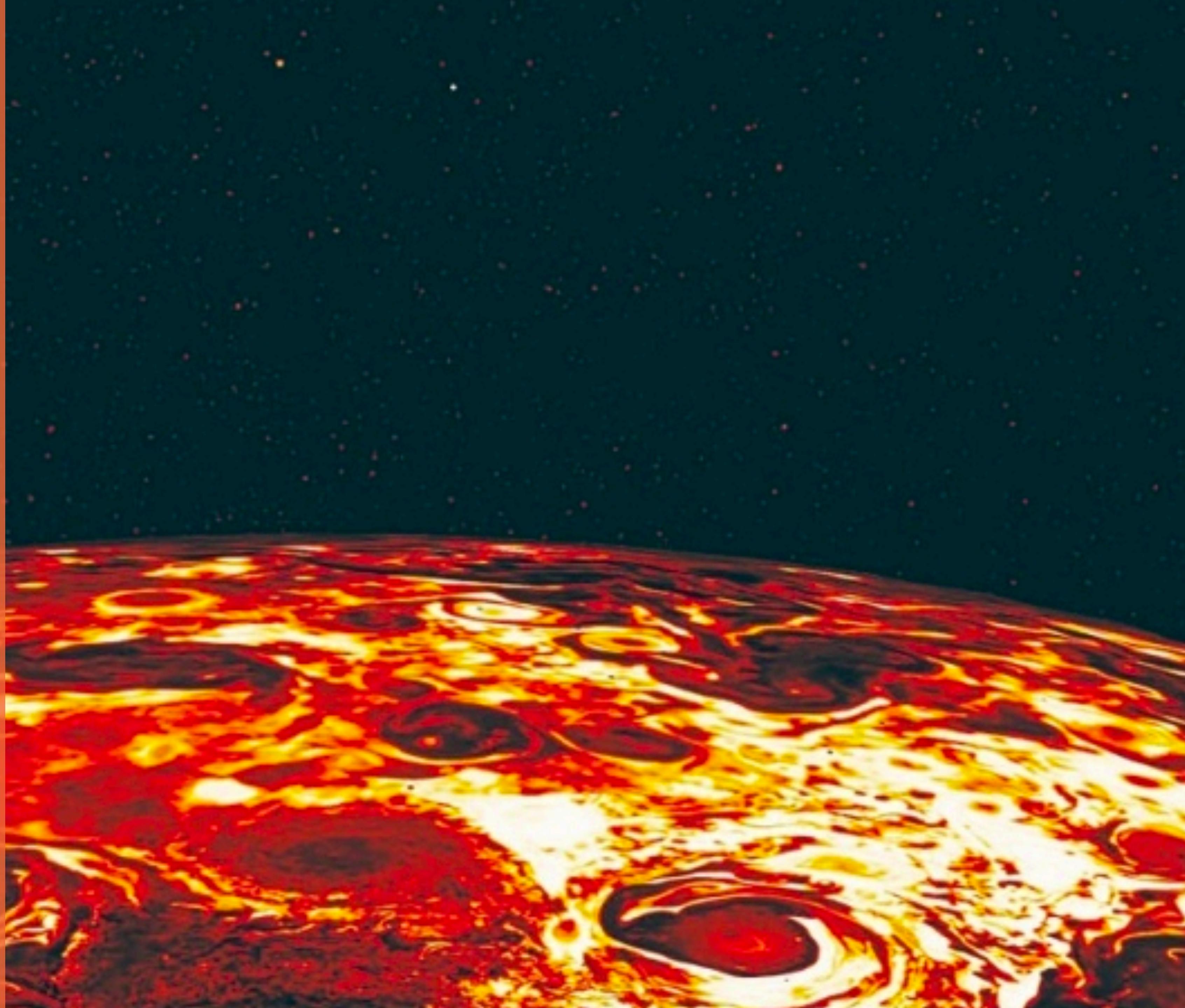
# Liftoff - resources

Postman API Docs and  
collection

Postman Echo



@petuniaGray





# OTHER NEW FEATURES

## Activate Thrusters



POSTMAN

Maintain your  
**API specifications.**



@petuniaGray



POSTMAN

The screenshot shows the Postman application interface. At the top, there is a navigation bar with 'New', 'Import', 'Runner', and other icons. The title 'My Workspace' is displayed, along with an 'Invite' button and user status indicators.

The main area is titled 'Room Booker' and is marked as '3.0 Show All Versions'. It includes tabs for 'Define', 'Develop' (which is currently selected), 'Test', and 'Observe'.

Below the main title, there are sections for 'Mock Servers', 'Documentation', and 'Environments'.

- Mock Servers:** Shows a table with one entry: 'Room Availability' (Version Tag: CURRENT, Last Updated: 15 May, 2019).
- Documentation:** Shows a table with one entry: 'Restful Booking' (Version Tag: CURRENT, Last Updated: 1 Oct, 2018).
- Environments:** Shows a table with two entries: 'Travel Booker' (Last Updated: 15 May, 2019) and 'Designing an API' (Last Updated: 29 Jan, 2019).

At the bottom, there are buttons for 'Bootcamp', 'Build', 'Browse', and other application controls.

# APIs!!!



POSTMAN

An easier way  
to **GraphQL**.



@petuniaGray



POSTMAN

# GraphQL

Postman

Published Postman Templates Invite Team

graph

History Collections APIs BETA

+ New Collection Working with GraphQL 4 requests

POST Using JSON request body

POST Using GraphQL query

POST Import as cURL

POST Built-in support for GraphQL

POST Built-in support for GraphQL

Built-in support for GraphQL Examples (1)

POST https://spotify-graphql-server.herokuapp.com/graphql

Params Authorization Headers (9) Body Tests Cookies Code Comments (0)

none form-data x-www-form-urlencoded raw binary GraphQL BETA

QUERY

```
1 query getByArtist ($name: String!) {  
2   queryArtists (byName: $name) {  
3     name  
4     image  
5     albums {  
6       name  
7     }  
8   }  
9 }
```

GRAPHQL VARIABLES

```
1 {  
2   "name": "{{artist}}"  
3 }
```

Body Cookies Headers (9) Test Results Status: 200 OK Time: 633 ms Size: 3.64 KB

Pretty Raw Preview JSON

```
1 {  
2   "data": {  
3     "queryArtists": [  
4       {  
5         "name": "Pink Floyd",  
6         "image": "https://i.scdn.co/image/e69f71e2be4b67b82af90fb8e9d805715e0684fa",  
7         "albums": [  
8           {  
9             "name": "The Endless River"  
10            },  
11          ]  
12        }  
13      ]  
14    }  
15  }
```

Bootcamp Build Browse



POSTMAN

Intercept your  
cookies.





POSTMAN

# Interceptor

The screenshot shows the Postman application interface with the 'Collections' tab selected in the sidebar. A modal window titled 'Cookies' is open, specifically the 'Capture requests and cookies' section. The 'Capture cookies' toggle switch is turned 'ON'. Below it, a domain 'twitter.com' is listed under 'Domains' with a green 'INTERCEPTOR CONNECTED' status indicator. The main workspace shows a request for 'GET Cookies' with no response yet.

Postman

New Import Runner + My Workspace Invite

Filter

History Collections APIs BETA

+ New Collection

Interceptor ★ 1 request

GET Cookies

Pokemon GraphQL ★ 3 requests

Spotify ★ 7 requests

Sushi Selector ★ 4 requests

Working with GraphQL ★ 4 requests

Writing Tests ★ 24 requests

Zookeeper ★ 30 requests

A Mock Status Checker 4 requests

GET Cookies

Cookies BETA

Capture requests and cookies

Requests Cookies

Capture cookies  ON INTERCEPTOR CONNECTED

Domains

Enter a domain to capture cookies Add Domain

twitter.com

No response yet

For you

Send Save

Examples (0)

Cookie Code Comments (0)

ON ... Bulk Edit

Bootcamp Build Browse



POSTMAN

Finer control for  
generating code.



@petuniaGray



POSTMAN

# CodeGen

Postman

Food-related ▾ [Invite](#)

No Environment

Comments (0) Examples (1)

Send Save

APIs BETA

Trash

GENERATE CODE SNIPPETS

Filter languages... [X](#)

Generated code for NodeJs - Request

Contribute on GitHub [⚙️](#) [🔗](#)

```
1 var request = require('request');
2 var options = {
3   'method': 'GET',
4   'url': 'https://postman-echo.com/get?city={{City}}&ramen={{Ramen}}',
5   'headers': {
6     }
7   };
8 request(options, function (error, response) {
9   if (error) throw new Error(error);
10  console.log(response.body);
11 });
12
```

ON

...

Bulk Edit

Cookies Code

Hit Send to get a response

For you

Bootcamp Build Browse



POSTMAN

Powerful  
debugging  
with the **console**.





POSTMAN

# Console

Postman Console

Search messages Filter messages Clear

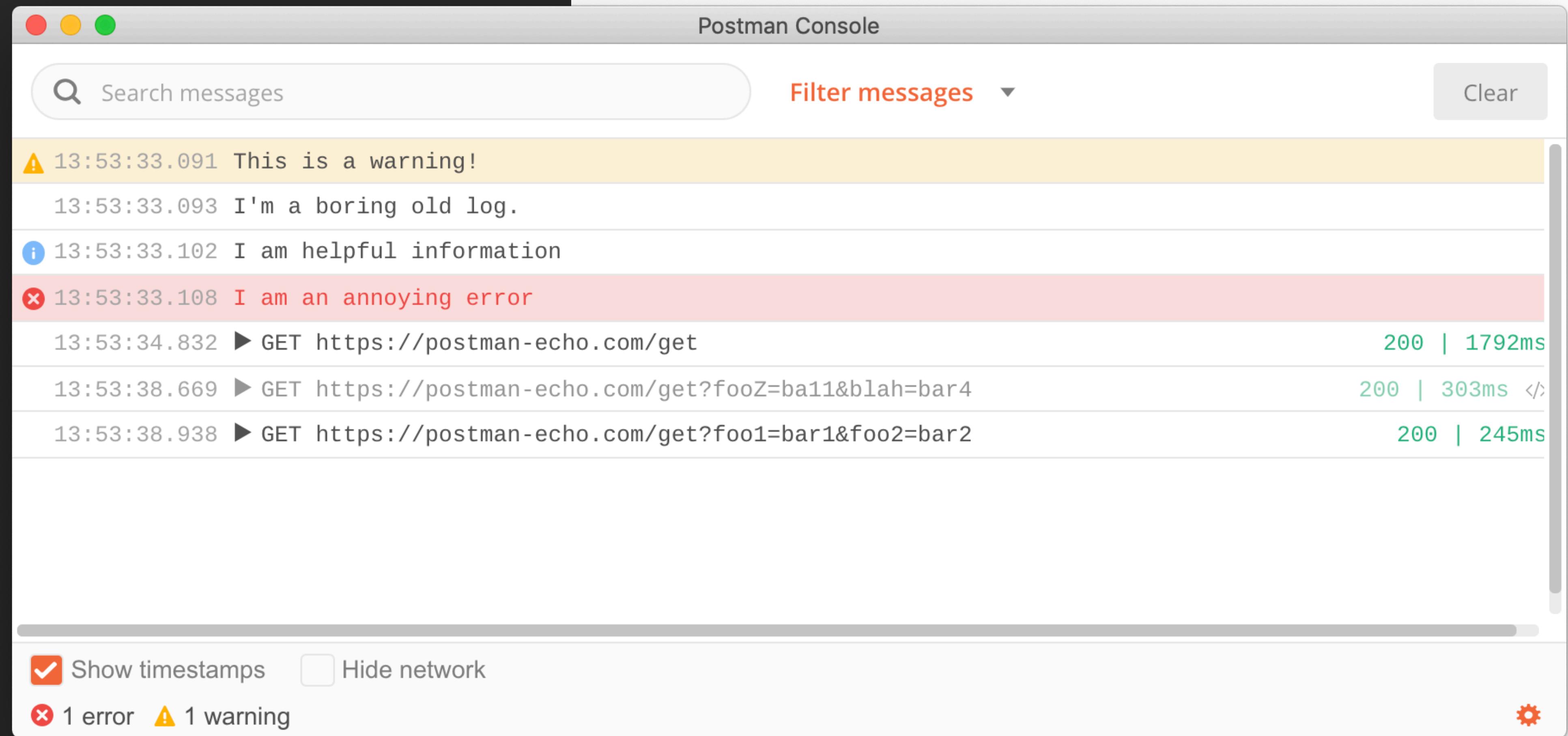
⚠ 13:53:33.091 This is a warning!  
13:53:33.093 I'm a boring old log.

ℹ 13:53:33.102 I am helpful information

✖ 13:53:33.108 I am an annoying error

13:53:34.832 ► GET https://postman-echo.com/get 200 | 1792ms  
13:53:38.669 ► GET https://postman-echo.com/get?fooZ=ba11&blah=bar4 200 | 303ms </>  
13:53:38.938 ► GET https://postman-echo.com/get?foo1=bar1&foo2=bar2 200 | 245ms

Show timestamps  Hide network   
✖ 1 error ⚠ 1 warning 





POSTMAN

# Activate thrusters - resources

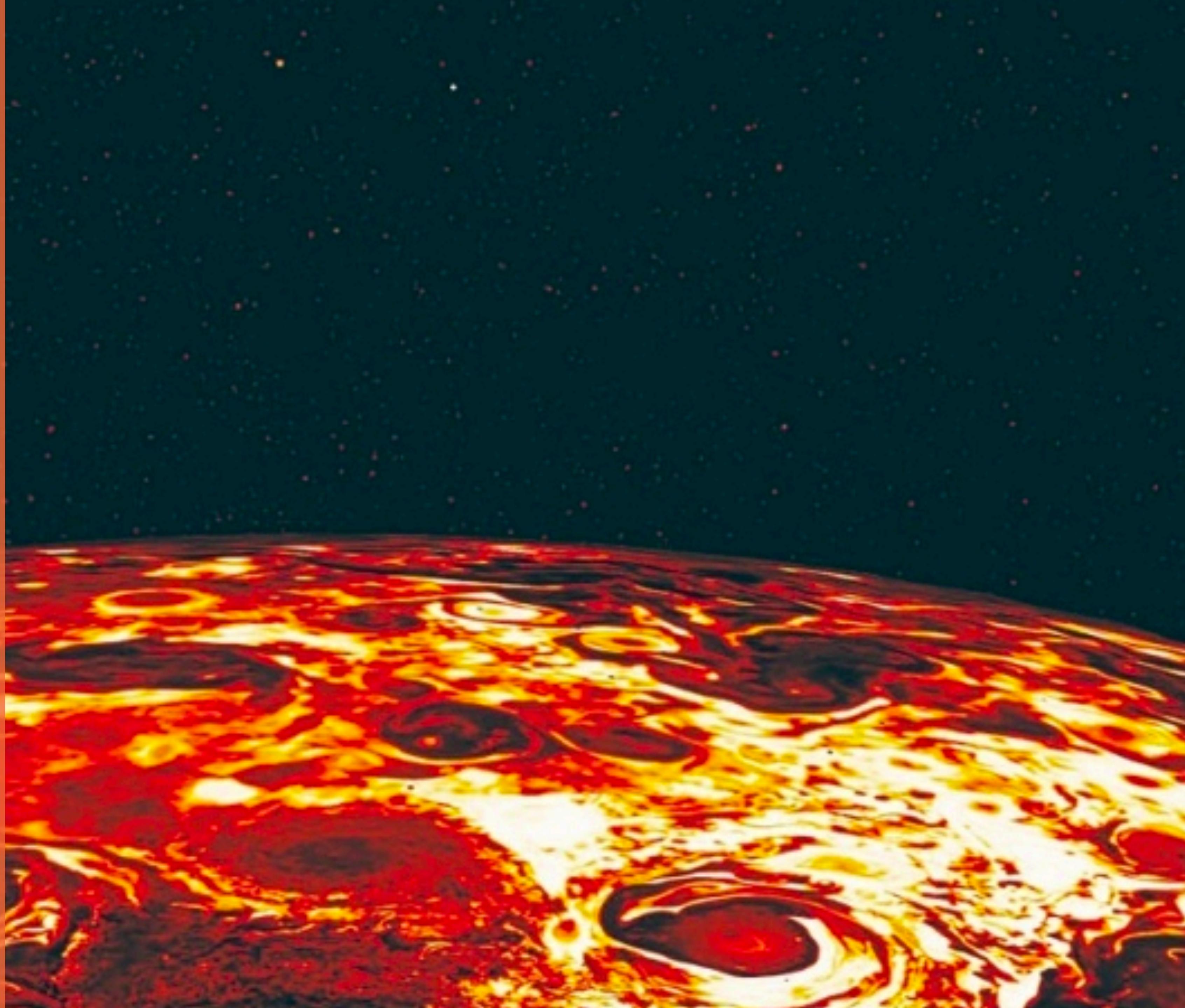
Cosmos

Working with GraphQL

How to use the Postman  
console



@petuniaGray





POSTMAN

# Final thoughts



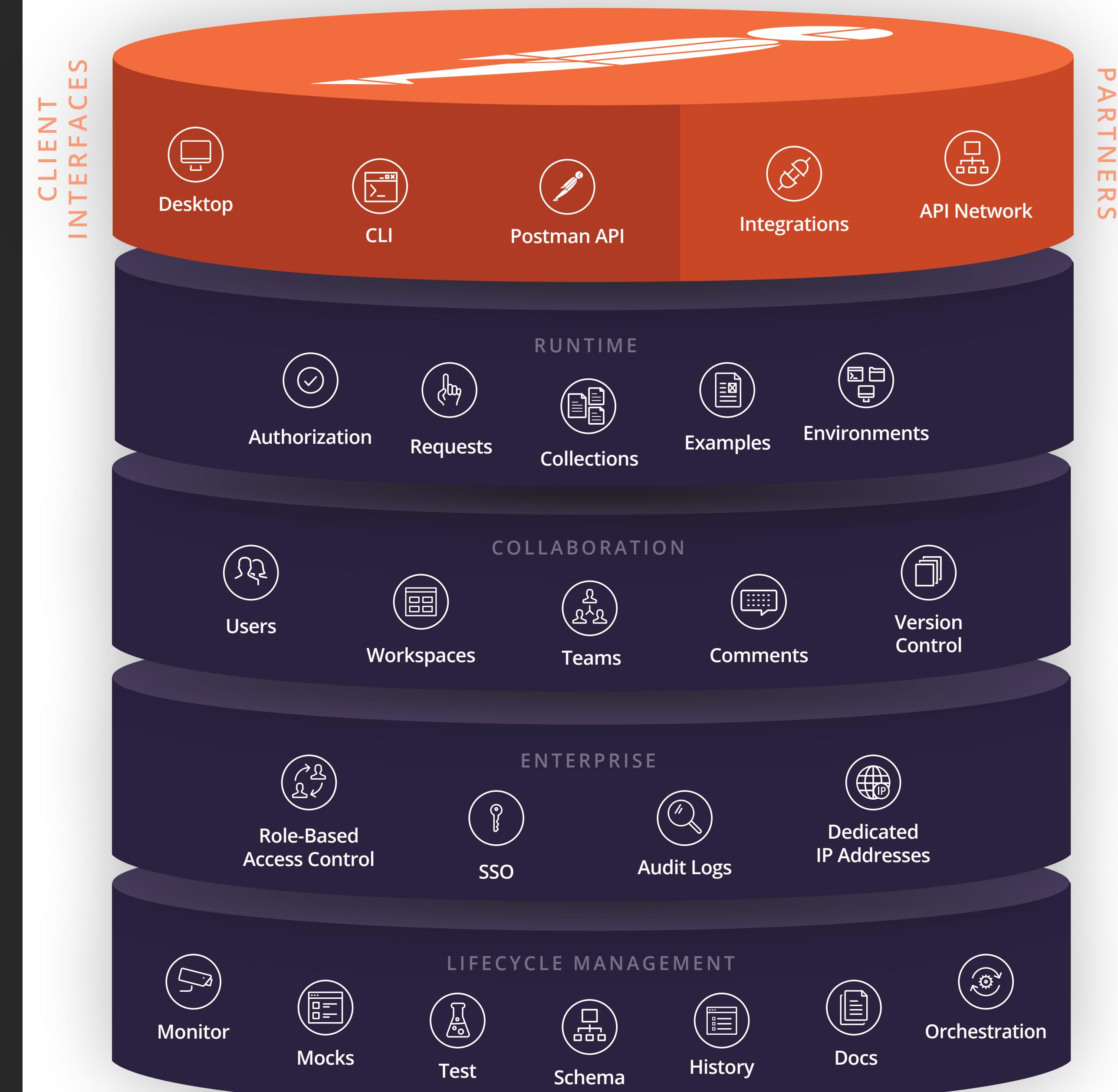
@petuniaGray





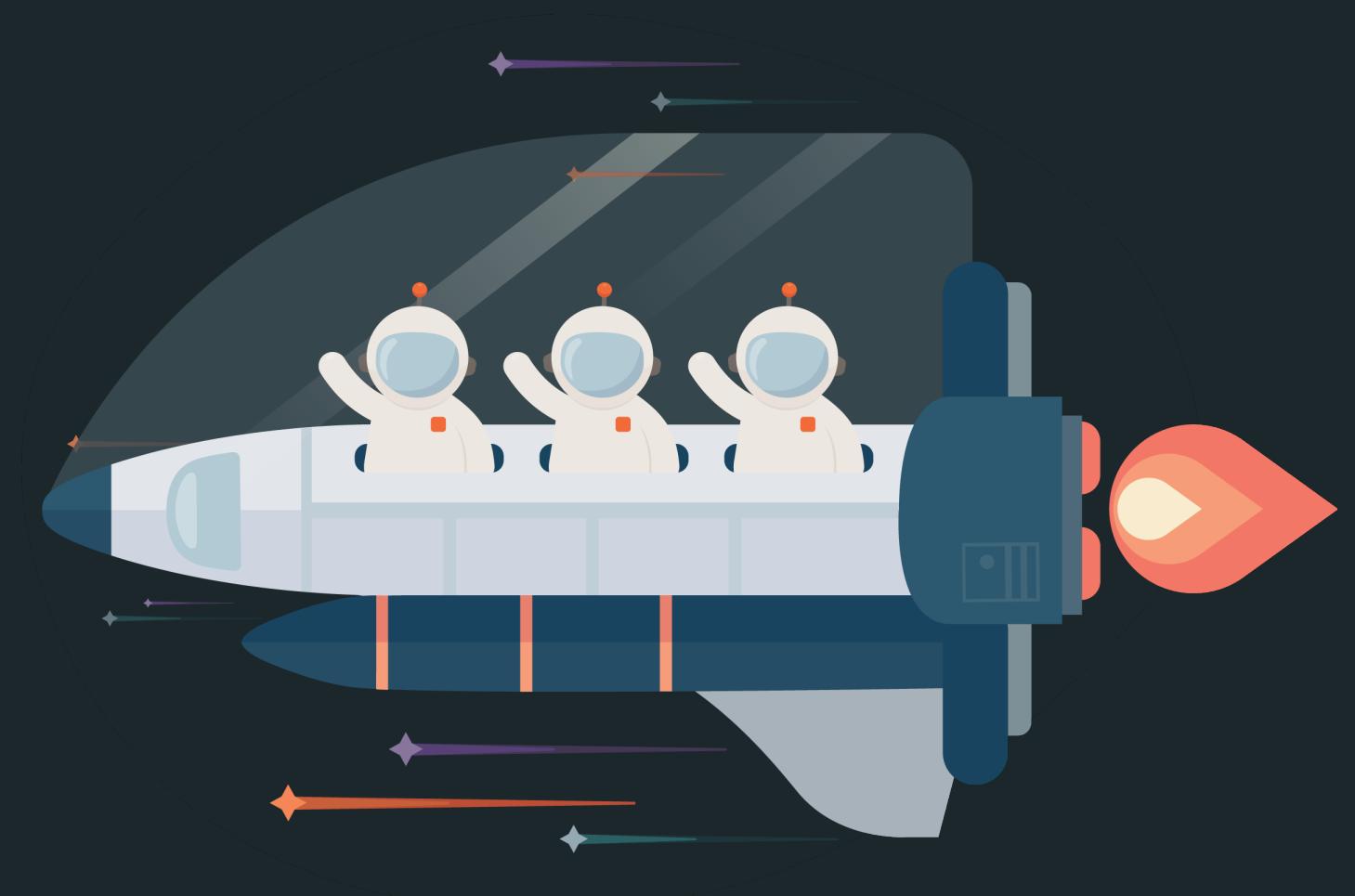
POSTMAN

# Postman as a Platform



# Additional Resources

- **Launchpad** - in app
- **Community-contributed templates** - [https://learning.getpostman.com/docs/postman\\_for\\_publishers/postman\\_templates/add\\_templates](https://learning.getpostman.com/docs/postman_for_publishers/postman_templates/add_templates)
- **Community forum** - <https://community.getpostman.com/>
- **Product Roadmap** - <https://trello.com/b/4N7PnHAz/postman-roadmap-for-developers>
- **Better Practices** - <https://medium.com/better-practices>





# Side missions

Mock servers for  
prototypes and testing

API specifications and  
schema validation

CDC testing for  
microservices

Source control and  
versioning





POSTMAN

# Thank you

---

getpostman.com

getpostman.com

 @getpostman

 @petuniaGray

