Kong

SHORT KONG DEMO

DEMO SETUP

For this short demo we have Kong setup as a Docker container with a PostgreSQL database, i.e. we will be using:

Kong container

PostgreSQL container

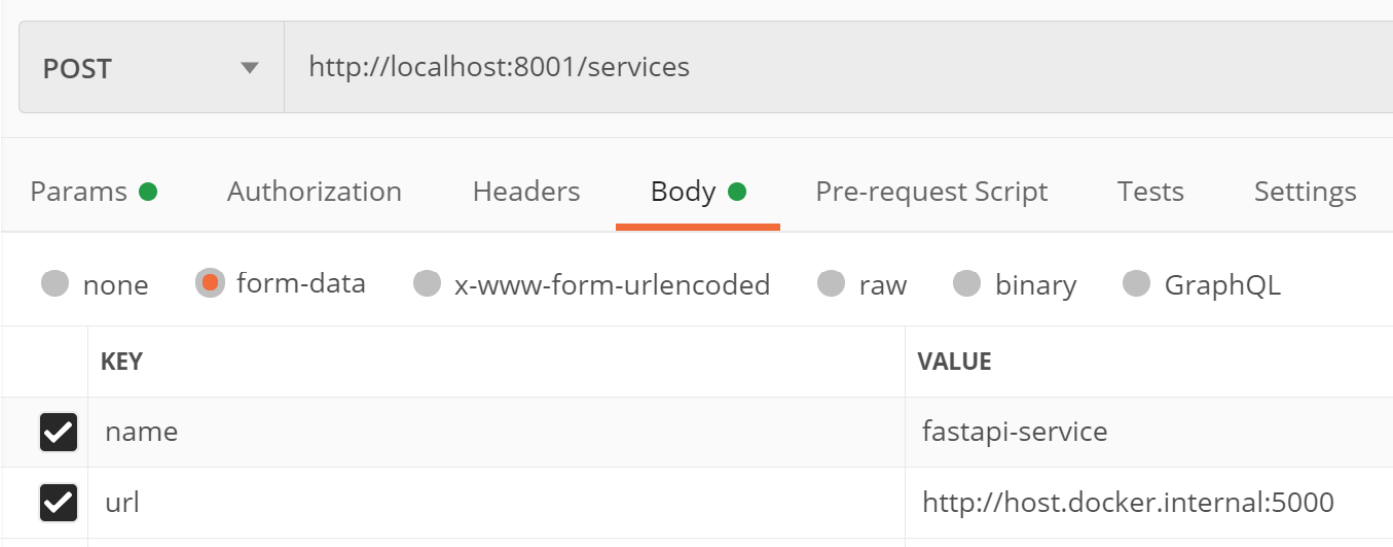
For the service we will just be using a simple crud API provided by the FastAPI framework (https://fastapi.tiangolo.com/tutorial/sql-databases/) which we will start at port :5000

uvicorn sqlapp.main:app --reload --port 5000

*for better visibility Postman will be used for the requests

CONFIGURING A SERVICE

Kong exposes a RESTful Admin API on port :8001. Kong's configuration, including adding Services and Routes, is made via requests on that API.



We are sending a POST request to the Admin API /services, configuring a service with a name of fastapi-service and a url of http://host.docker.internal:5000

*host.docker.internal is used instead of localhost because of docker container scope

ADDING A ROUTE FOR OUR SERVICE

POST http://localhost:8001/services/fastapi-service/routes

Params Authorization Headers Body Pre-request Script Tests Settings

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

	KEY	VALUE
☑	paths[]	/fastapi/
☑	hosts[]	fastapi.com

Kong is now forwarding requests made to http://localhost:8000 to the url we configured, and is forwarding the response back to us. Kong knows to do this through the Host header defined in the GET request or the path http://localhost:8000/fastapi/

EXAMPLE GET REQUEST

GET http://localhost:8000/fastapi/users/ Send Save

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Bulk Edit

Body Cookies Headers (13) Test Results Status: 200 OK Time: 181ms Size: 407 B Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "email": "bojan@g.com",
        "id": 1,
        "is_active": true,
        "items": []
    }
]
```

ENABLING KONG PLUGINS

One of the core principles of Kong is its extensibility through plugins. Plugins allow you to easily add new features to your Service or make it easier to manage.

We will show how to configure a few plugins:

- Rate limiting
- Proxy Cache
- Key Authentication

RATE LIMITING

Rate limit how many HTTP requests a developer can make in a given period of seconds, minutes, hours, days, months or years.

It is very simple to configure

POST http://localhost:8001/services/fastapi-service/plugins

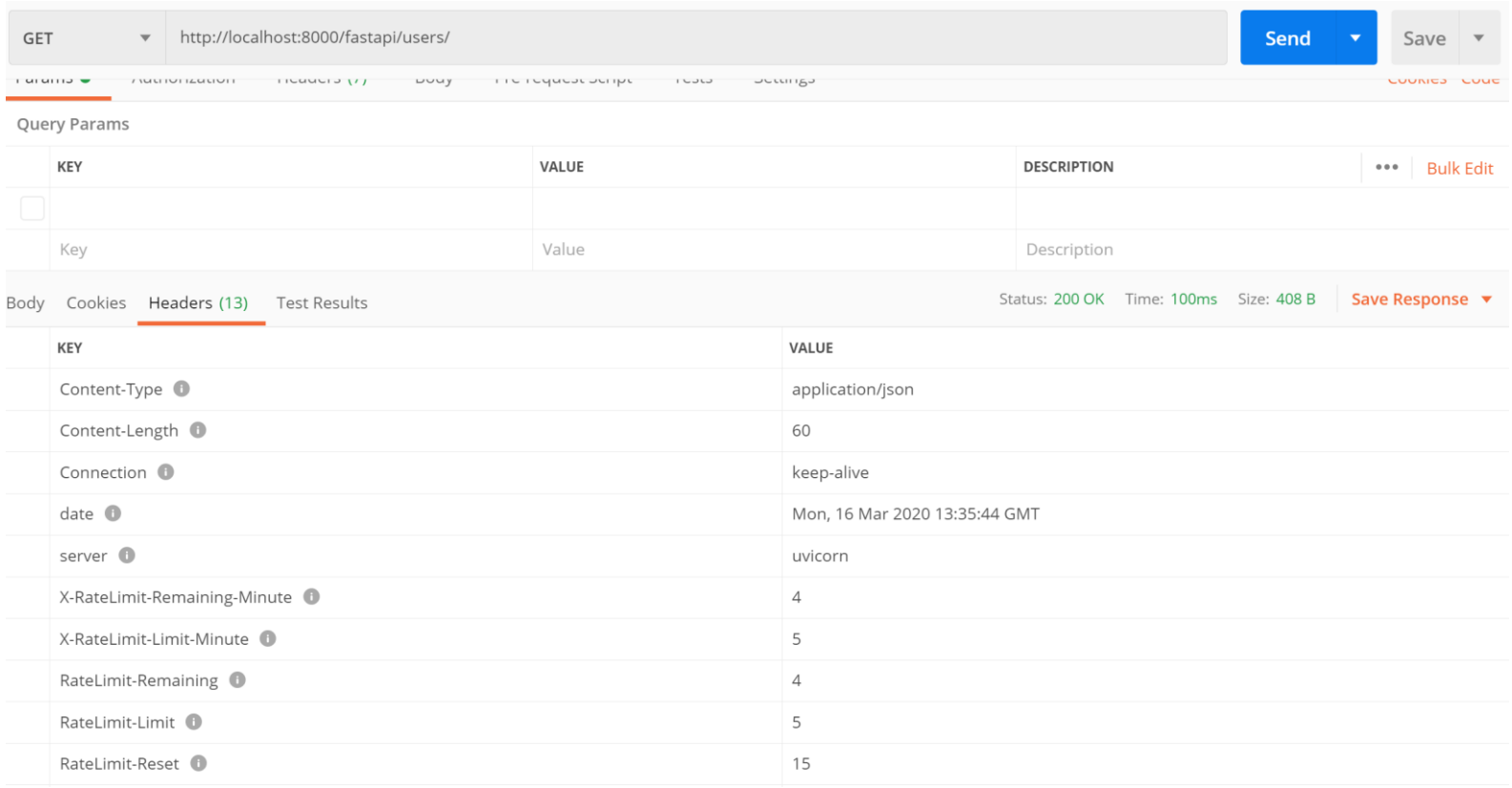
Params Authorization Headers Body Pre-request Script Tests Settings

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

	KEY	VALUE
☑	name	rate-limiting
☑	config.minute	5

RATE LIMITING CONT.

We added a simple rate limit of 5 requests per minute and as we can see in the screenshot the RateLimit-Remaining Header goes down to 4 after making a GET request



PROXY CACHE

It caches response entities based on configurable response code and content type, as well as request method. It can cache per-Consumer or per-API. Cache entities are stored for a configurable period of time, after which subsequent requests to the same resource will re-fetch and re-store the resource.

We will configure this plugin on our service by using the default request method (GET,HEAD), the strategy of memory for the backing data store, and a cache ttl(time to live) of 300 seconds, which is also the default value.

PROXY CACHE CONT.

POST http://localhost:8001/services/fastapi-service/plugins

Params Authorization Headers (9) Body Pre-request Script Tests Settings

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

	KEY	VALUE
☑	name	proxy-cache
☑	config.strategy	memory
☑	config.cache_ttl	300

CACHING EXAMPLE

We can see the effect of the caching in the following headers

Before

X-Kong-Upstream-Latency	47
X-Kong-Proxy-Latency	30

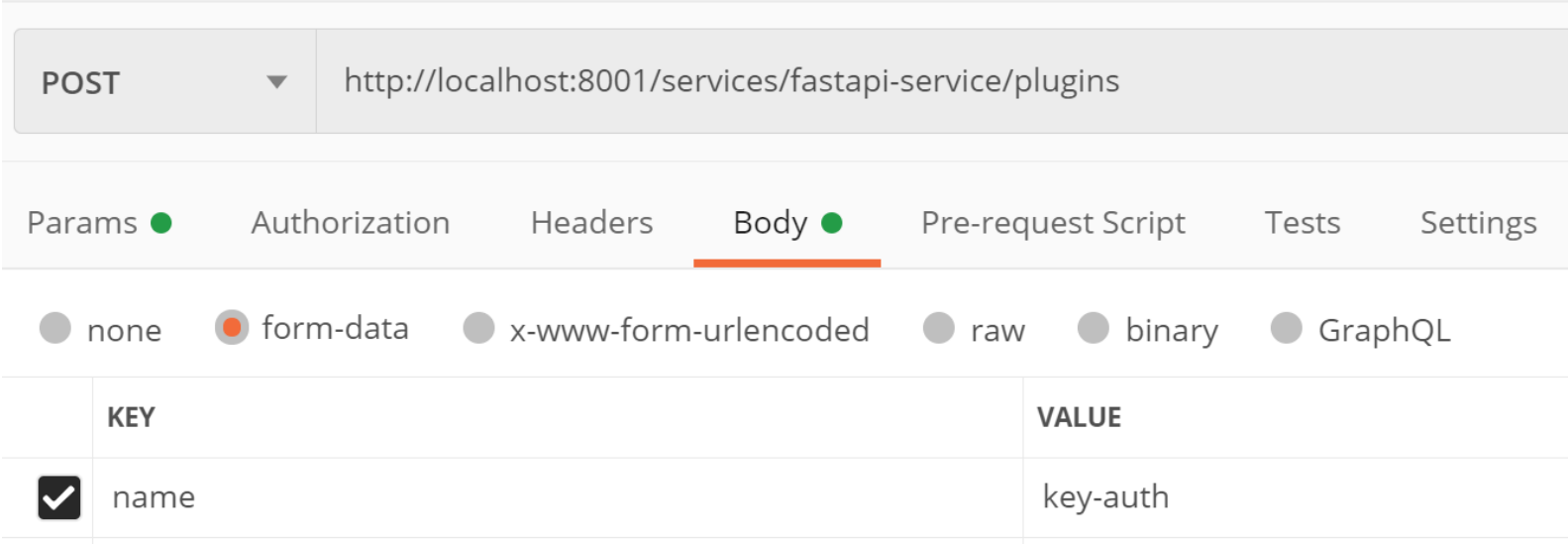
After

X-Kong-Upstream-Latency	0
X-Kong-Proxy-Latency	2

KEY AUTHENTICATION

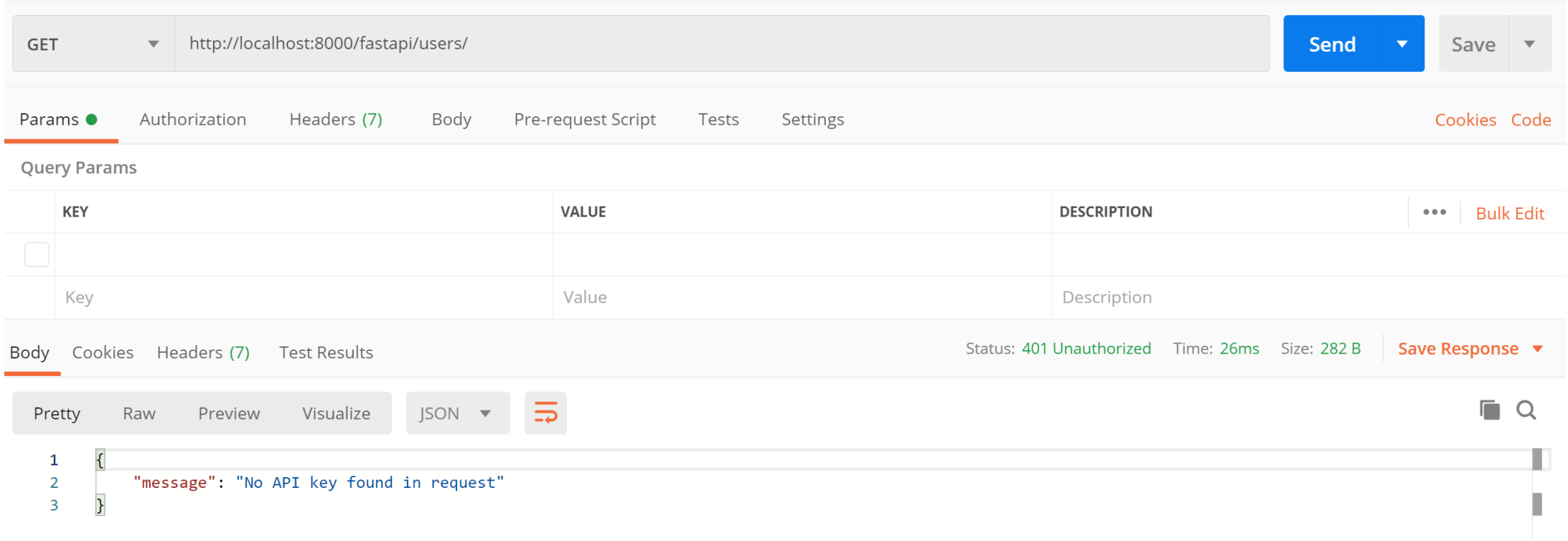
We will configure the key-auth plugin to add authentication to our Service.

Prior to the addition of this plugin, **all** requests to your Service would be proxied upstream. Once you add and configure this plugin, **only** requests with the correct key(s) will be proxied - all other requests will be rejected by Kong, thus protecting your upstream service from unauthorized use.



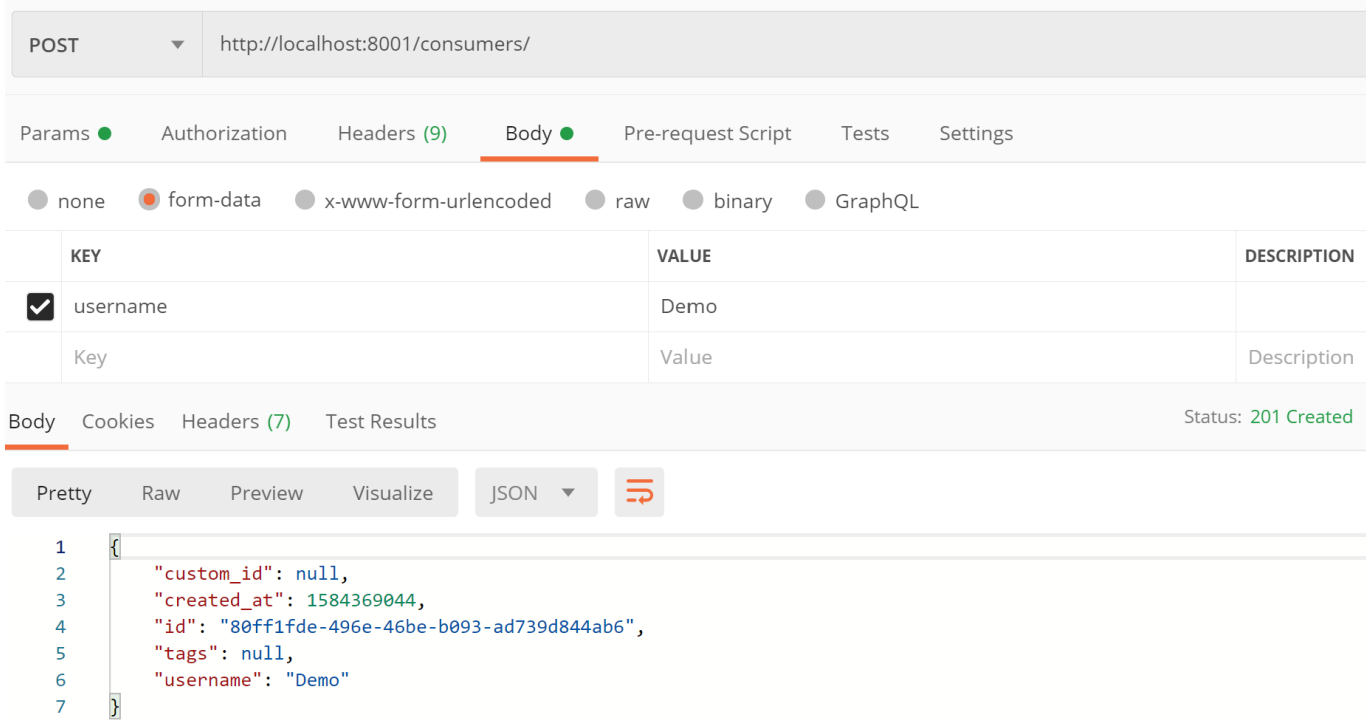
KEY AUTHENTICATION CONT.

Now we will make a GET request. Since we did not specify the required apikey header or parameter, the response should be 401 Unauthorized:



KEY AUTHENTICATION CONT.

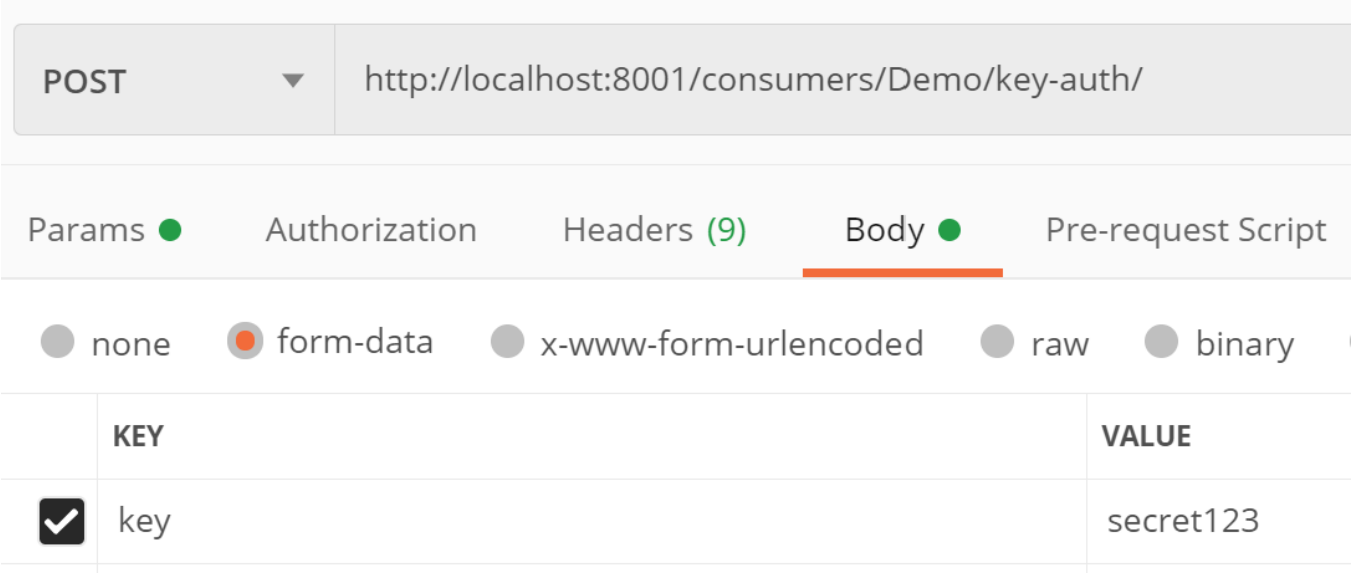
We need to provision a Consumer so we can continue proxying requests through Kong. So, lets create a user named Demo by issuing the following request:



Note: Kong also accepts a custom_id parameter when creating consumers to associate a consumer with your existing user database.

KEY AUTHENTICATION CONT.

Now, we can create a key for our recently created consumer Demo by issuing the following request:



KEY AUTHENTICATION CONT.

Finally, we can now consume our service providing the apikey (parameter or header)

GET http://localhost:8000/fastapi/users/?apikey=secret123 Send

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

Query Params

	KEY	VALUE	DESCRIPTION
☑	apikey	secret123	
	Key	Value	Description

Body Cookies Headers (15) Test Results Status: 200 OK Time: 187ms Size: 478 B

Pretty Raw Preview Visualize JSON


```
[
    {
        "email": "bojan@g.com",
        "id": 1,
        "is_active": true,
        "items": []
    }
]
```

THANK YOU