



# 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

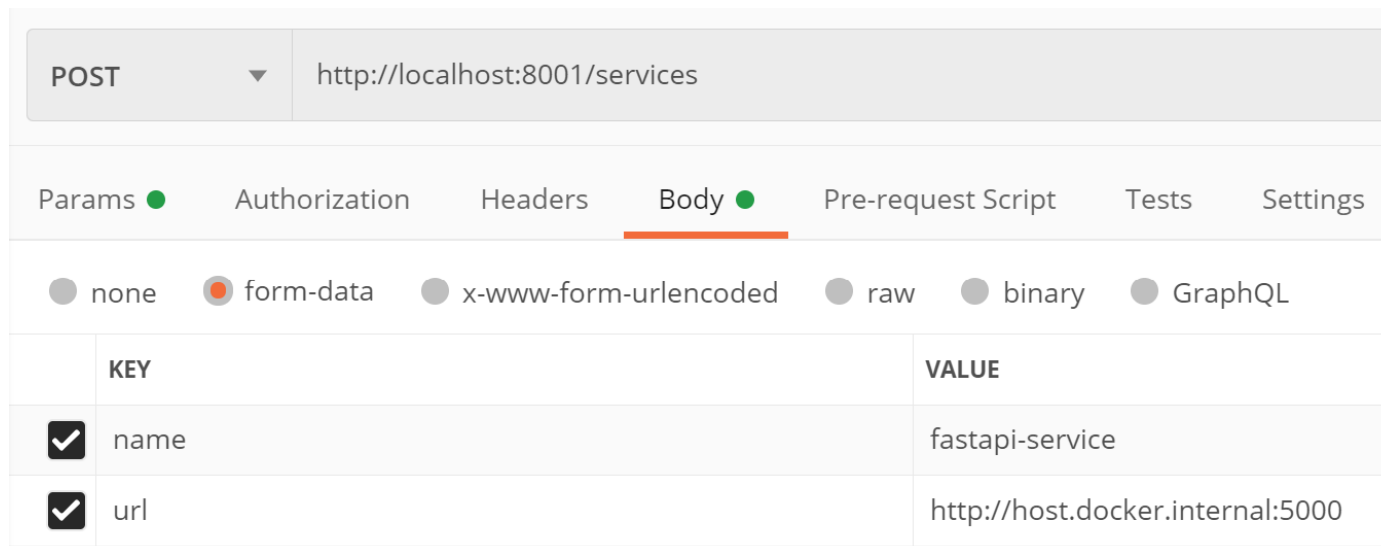
For the service we will just be using a simple crud API provided by the FastAPI framework (<https://fastapi.tiangolo.com/tutorial/sql-databases/>) which we will start at port :5000

```
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.



The screenshot shows a REST client interface with a POST request to `http://localhost:8001/services`. The 'Body' tab is selected, and the format is set to 'form-data'. The body contains two fields: 'name' with value 'fastapi-service' and 'url' with value 'http://host.docker.internal:5000'.

	KEY	VALUE
<input checked="" type="checkbox"/>	name	fastapi-service
<input checked="" type="checkbox"/>	url	http://host.docker.internal:5000

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
<input checked="" type="checkbox"/>	paths[]	/fastapi/
<input checked="" type="checkbox"/>	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	...	Bulk Edit
<input type="checkbox"/>					
	Key	Value	Description		

Body

Cookies

Headers (13)

Test Results

Status: 200 OK

Time: 181ms

Size: 407 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1  [
2    {
3      "email": "bojan@g.com",
4      "id": 1,
5      "is_active": true,
6      "items": []
7    }
8  ]
```

# 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
<input checked="" type="checkbox"/>	name	rate-limiting
<input checked="" type="checkbox"/>	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

The screenshot shows a REST client interface with a GET request to `http://localhost:8000/fastapi/users/`. The response status is 200 OK, with a time of 100ms and a size of 408 B. The response headers are displayed in a table below the status bar.

KEY	VALUE
Content-Type	application/json
Content-Length	60
Connection	keep-alive
date	Mon, 16 Mar 2020 13:35:44 GMT
server	uvicorn
X-RateLimit-Remaining-Minute	4
X-RateLimit-Limit-Minute	5
RateLimit-Remaining	4
RateLimit-Limit	5
RateLimit-Reset	15



# 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
<input checked="" type="checkbox"/>	name	proxy-cache
<input checked="" type="checkbox"/>	config.strategy	memory
<input checked="" type="checkbox"/>	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.

POST	http://localhost:8001/services/fastapi-service/plugins					
Params	Authorization	Headers	Body	Pre-request Script	Tests	Settings
<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	none	form-data	x-www-form-urlencoded	raw	binary	GraphQL
	KEY				VALUE	
<input checked="" type="checkbox"/>	name				key-auth	

# 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:

The screenshot shows a REST client interface with the following components:

- Request Bar:** Method: GET, URL: http://localhost:8000/fastapi/users/. Buttons: Send, Save.
- Tabs:** Params (selected), Authorization, Headers (7), Body, Pre-request Script, Tests, Settings. Links: Cookies, Code.
- Query Params Table:**

	KEY	VALUE	DESCRIPTION	
<input type="checkbox"/>				
	Key	Value	Description	
- Response Bar:** Body (selected), Cookies, Headers (7), Test Results. Status: 401 Unauthorized, Time: 26ms, Size: 282 B. Button: Save Response.
- Response Body:** Pretty, Raw, Preview, Visualize. Format: JSON. Content:

```
1 {
2   "message": "No API key found in request"
3 }
```

# KEY AUTHENTICATION CONT.

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

The screenshot shows a REST client interface with a POST request to `http://localhost:8001/consumers/`. The 'Body' tab is selected, showing a JSON payload. The response status is 201 Created.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> username	Demo	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 201 Created

Pretty Raw Preview Visualize JSON ↻

```
1 {
2   "custom_id": null,
3   "created_at": 1584369044,
4   "id": "80ff1fde-496e-46be-b093-ad739d844ab6",
5   "tags": null,
6   "username": "Demo"
7 }
```

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:

POST	http://localhost:8001/consumers/Demo/key-auth/								
Params	Authorization	Headers (9)	Body	Pre-request Script					
<input type="radio"/>	none	<input checked="" type="radio"/>	form-data	<input type="radio"/>	x-www-form-urlencoded	<input type="radio"/>	raw	<input type="radio"/>	binary
	KEY					VALUE			
<input checked="" type="checkbox"/>	key					secret123			

# KEY AUTHENTICATION CONT.

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

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8000/fastapi/users/?apikey=secret123
- Buttons:** Send (blue)
- Tabs:** Params (selected), Authorization, Headers (7), Body, Pre-request Script, Tests, Settings
- Query Params Table:**

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	apikey	secret123	
	Key	Value	Description
- Status Bar:** Status: 200 OK, Time: 187ms, Size: 478 B
- Response Body Tabs:** Body (selected), Cookies, Headers (15), Test Results
- Response Format:** JSON (selected), Pretty, Raw, Preview, Visualize
- Response Content (JSON):**

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





**THANK YOU**