DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

Tutorial – Installing and Operating KONG Open Source

Instituto Superior Técnico
Universidade de Lisboa
Integração Empresarial / Enterprise Integration

The goal of this document is to demonstrate how to install and operate the Kong open source service. Kong is a platform aiming the integration between all the micro services (λ and JAVA), and the BPMN processes developed in Camunda. Moreover, a Kong management platform, named Konga, is introduced.

The following contents is presented in this document.

Contents

A. Installing and starting the Kong Open Source Service as a docker in EC2

A.1. Create an EC2 new instance of the type: Amazon Linux 2 AMI (HVM), SSD Volume Type. The exact versions may change with time. And define the inbound rules to allow the 8080, 8002 and 22 accessed from anywhere.

A.2. Access the new EC2 instance and execute the following commands:

```
sudo yum install -y docker

sudo service docker start

sudo docker network create kong-net
```

```
sudo docker run -d --name kong-database \
  --network=kong-net \
  -p 5432:5432 \
  -e "POSTGRES_USER=kong" \
  -e "POSTGRES_DB=kong" \
  -e "POSTGRES_PASSWORD=kongpass" \
  postgres:13
```

```
sudo docker run --rm --network=kong-net \
-e "KONG_DATABASE=postgres" \
-e "KONG_PG_HOST=kong-database" \
-e "KONG_PG_PASSWORD=kongpass" \
-e "KONG_PASSWORD=test" \
kong/kong-gateway:3.9.0.0 kong migrations bootstrap
```

```
sudo docker run -d --name kong-gateway \
--network=kong-net \
-e "KONG_DATABASE=postgres" \
-e "KONG_PG_HOST=kong-database" \
-e "KONG_PG_USER=kong" \
-e "KONG_PG_PASSWORD=kongpass" \
-e "KONG_PROXY_ACCESS_LOG=/dev/stdout" \
-e "KONG_ADMIN_ACCESS_LOG=/dev/stdout" \
-e "KONG_PROXY_ERROR_LOG=/dev/stderr" \
-e "KONG_ADMIN_ERROR_LOG=/dev/stderr" \
-e "KONG_ADMIN_LISTEN=0.0.0.0:8001, 0.0.0.0:8444 ssl" \
-e "KONG_ADMIN_GUI_URL=http://localhost:8002" \
-e KONG_LICENSE_DATA \
-p 8000:8000 \
-p 8443:8443 \
-p 8001:8001 \
-p 8002:8002 \
-p 8445:8445 \
-p 8003:8003 \
-p 8004:8004 \
-p 127.0.0.1:8444:8444 \
kong/kong-gateway:3.9.0.0
```

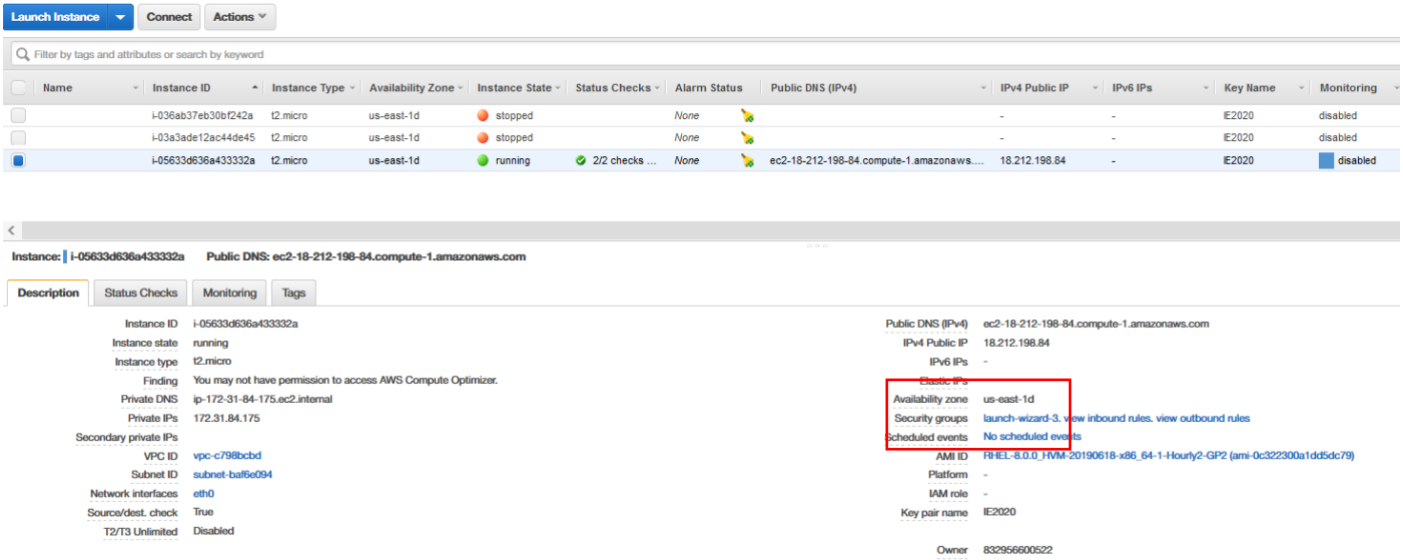
A.3. Verify your installation in EC2 instance:

```
curl -i -X GET --url http://localhost:8001/services
```

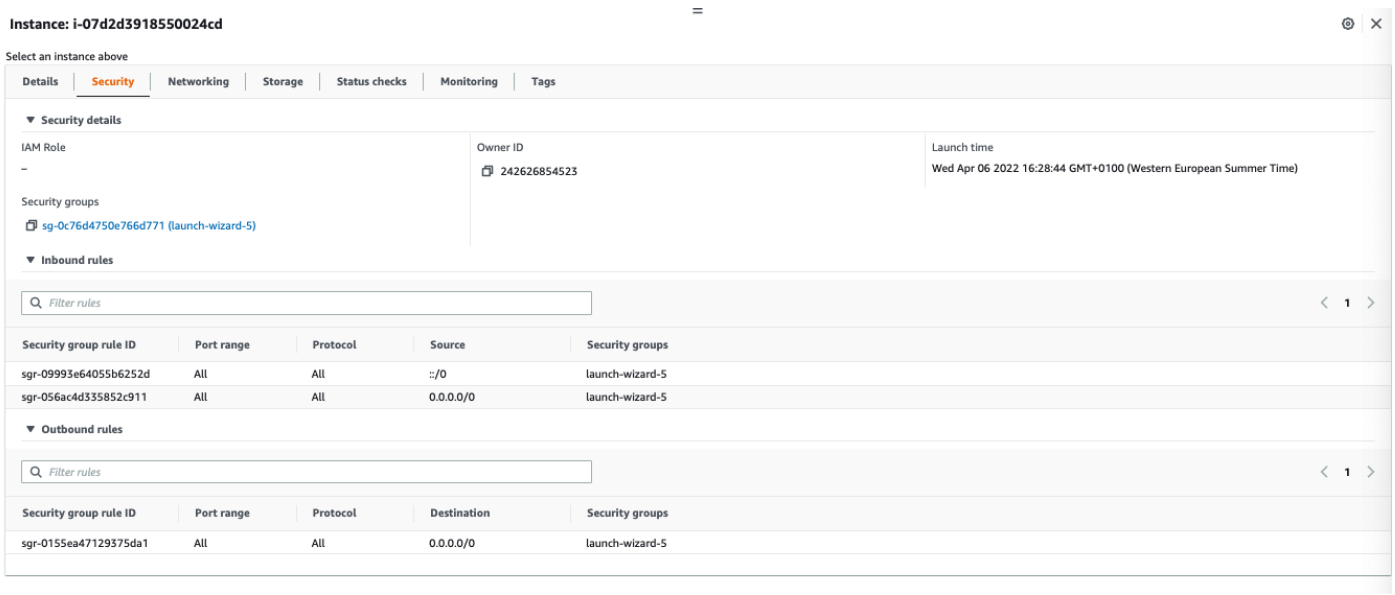
You should receive a 200 status code.

B. Configuring Kong to be used remotely

B.1. Choose you EC2 instance. Click on security-group (for example):



And change the configuration to allow all the inbound traffic:



B.2. Test with the following command from your local computer:

```
curl -i http://<YOURIP>:8000/
```

You should receive a similar response with the following:

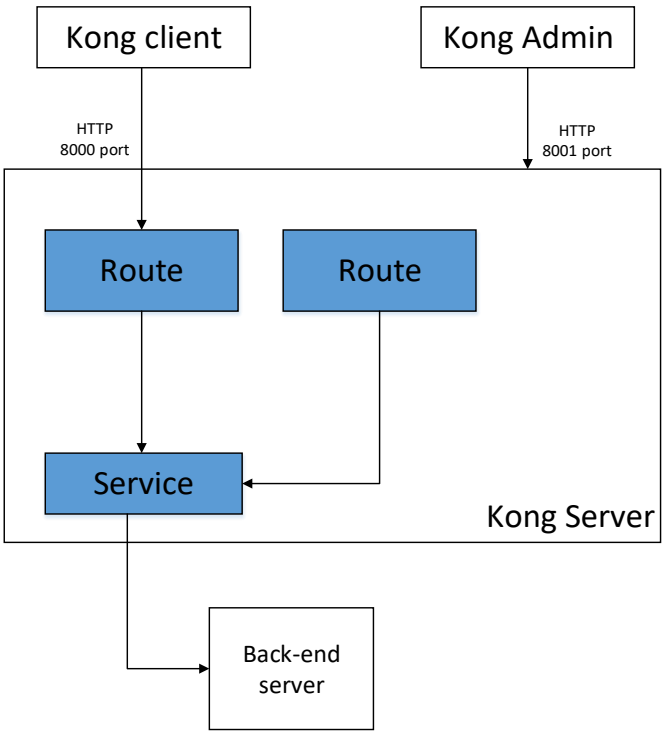
```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    48  100    48    0     0     48      0  0:00:01 --:--:--  0:00:01   192
HTTP/1.1 404 Not Found
Date: Tue, 14 Jan 2020 15:34:55 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Content-Length: 48
X-Kong-Response-Latency: 0
Server: kong/2.0.0rc2

{"message":"no Route matched with those values"}
```

C. Creating a service and a route in Kong

From Kong definitions:

- a Service; that is the name Kong uses to refer to the upstream APIs and microservices it manages.
- and a Route specify how (and if) requests are sent to their Services after they reach Kong. A single Service can have many Routes.

In Kong the ports are organized as follows: Port 8001 – is used for managing APIs and Port 8000 – for service invocation. This is depicted by the following figure:



To try-out use the following commands:

C.1. In your Kong AWS EC2 instance (for security reasons is safer to do it locally) issue the following cURL request to add your first Service (pointing to the Mockbin API) to Kong:

```
curl -i -X POST \
  --url http://localhost:8001/services/ \
  --data 'name=example-service' \
  --data 'url=http://mockbin.org'
```

You should receive a response similar to:

```
HTTP/1.1 201 Created
Content-Type: application/json
Connection: keep-alive

{
   "host":"mockbin.org",
   "created_at":1519130509,
   "connect_timeout":60000,
   "id":"92956672-f5ea-4e9a-b096-667bf55bc40c",
   "protocol":"http",
   "name":"example-service",
   "read_timeout":60000,
   "port":80,
   "path":null,
   "updated_at":1519130509,
   "retries":5,
   "write_timeout":60000
}
```

C.2. Then, add a Route for the previous Service:

```
 curl -i -X POST \
  --url http://localhost:8001/services/example-service/routes \
  --data 'hosts[]=example.com'
```

The answer should be similar to:

```
HTTP/1.1 201 Created
Content-Type: application/json
Connection: keep-alive

{
   "created at":1519131139,
   "strip path":true,
   "hosts":[
      "example.com"
   ],
   "preserve_host":false,
   "regex_priority":0,
   "updated_at":1519131139,
   "paths":null,
   "service":{
      "id":"79d7ee6e-9fc7-4b95-aa3b-61d2e17e7516"
   },
   "methods":null,
   "protocols":[
      "http",
      "https"
   ],
   "id":"f9ce2ed7-c06e-4e16-bd5d-3a82daef3f9d"
}
```

The service is now ready to be invoked. Continue to the next section of this document.

D. Invoking a service

Issue the following cURL request to verify that Kong is properly forwarding requests to your Service. Note that by default Kong handles proxy requests on port :8000. The following request indicates which is the route that you would like to call:

```
$ curl -i -X GET \
  --url http://localhost:8000/ \
  --header 'Host: example.com'
```

A successful response means Kong is now forwarding requests made to http://localhost:8000 to the url we configured in previous section and is forwarding the response back to us. Kong knows to do this through the header defined in the above curl request:

Host: <given host>

hint: If you are testing the invocation remotely from the AWS EC2 Kong instance, then you should replace the localhost by the AWS EC2 DNS name.

E. Connecting to a remote microservice

E.1. DIRECTLY TO LAMBDA BACKEND (as previous introduced in correspondingly tutorial):

```
curl -i -H "Content-Type: Application/json"\
 --data "@body.json"\
 -X POST https://2p3fp5acxj.execute-api.us-east-1.amazonaws.com/default/PrimeiroMicroServico
```

E.2. CREATE SERVICE (in the AWS EC2 Kong instance):

```
 curl -i -X POST \
  --url http://localhost:8001/services/ \
  --data 'name=invoke-lambda-service' \
  --data 'url=https://2p3fp5acxj.execute-api.us-east-1.amazonaws.com/default/PrimeiroMicroServico'\
```

E.3. CREATE ROUTE (in the AWS EC2 Kong instance):

```
 curl -i -X POST \
  --url http://localhost:8001/services/invoke-lambda-service/routes \
  --data 'hosts[]=disciplinas.com'
```

E.4. INVOKE REMOTELLY (from your local computer), considering that your JSON file should be available

```
curl -i -X POST \
  -H "Content-Type: Application/json" \
  --url http://ec2-18-212-198-84.compute-1.amazonaws.com:8000/ \
  --header 'Host: disciplinas.com'\
  --data '@body.json'
```

You are expected to receive something similar with:

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    79  100    43  100    36     43     36  0:00:01 --:--:--  0:00:01   220
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 43
Connection: keep-alive
Date: Tue, 14 Jan 2020 17:16:50 GMT
x-amzn-RequestId: 39bf3f69-af84-4568-96fb-b0277ea9e896
x-amz-apigw-id: GTOceEKUoAMFc4g=
x-custom-header: my custom header value
X-Amzn-Trace-Id: Root=1-5e1df782-21cac32327af61b6064ed89a;Sampled=0
X-Kong-Upstream-Latency: 28
X-Kong-Proxy-Latency: 0
Via: kong/2.0.0rc2

{"message":"Hello Integracao Empresarial!"}
```

F. Other example of connecting to a JAVA microservice

F.1. DIRECTLY TO JAVA WEBSERVICE BACKEND (as previous introduced in correspondingly tutorial):

```
curl -i -X POST -H "Content-Type: text/xml;charset=UTF-8" \
 --url http://ec2-52-91-101-30.compute-1.amazonaws.com:9298/calculate \
 --data '@calculate.xml'
```

F.2. CREATE SERVICE (in the AWS EC2 Kong instance):

```
 curl -i -X POST \
  --url http://localhost:8001/services/ \
  --data 'name=invoke-java-service' \
  --data 'url=http://ec2-52-91-101-30.compute-1.amazonaws.com:9298/calculate'\
```

F.3. CREATE ROUTE (in the AWS EC2 Kong instance):

```
 curl -i -X POST \
  --url http://localhost:8001/services/invoke-java-service/routes \
  --data 'hosts[]=calculate.com'
```

F.4. INVOKE REMOTELLY (from your local computer), considering that your XML file should be available

```
curl -i -X POST -H "Content-Type: text/xml;charset=UTF-8"\
  --url ec2-3-86-244-114.compute-1.amazonaws.com:8000/ \
  --header 'Host: calculate.com'  \
  --data '@calculate.xml'
```

You are expected to receive something similar with:

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   461    0   202  100   259    202    259  0:00:01 --:--:--  0:00:01  1054
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Date: Tue, 14 Jan 2020 21:44:18 GMT
X-Kong-Upstream-Latency: 8
X-Kong-Proxy-Latency: 3
Via: kong/2.0.0rc2

<?xml version="1.0" ?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"><S:Body><ns2:mulResponse
xmlns:ns2="http://Webservice/"><return>182</return></ns2:mulResponse></S:Body></S:Envelope>
```

G. Endpoints with parameters

G.1.To allow the access to endpoints with parameters, e.g.,

```
http://URL_OF_ENDPOINT:8080/fruits/2/bananas
```

where the exact command if you invoke it by curl is:

```
curl -X 'PUT' \
  'http://URL_OF_ENDPOINT:8080/fruits/2/bananas' \
  -H 'accept: application/json'
```

You need to create a service and route accordingly:

G.2.The Kong service need to be defined only with the base path without the service identification, for instance:

```
curl -i -X POST \
--url "http://URL OF KONG:8001/services/" \
--data "name=invoke-quarkus4-service" \
--data "url=http://URL_OF_ENDPOINT:8080/"
```

G.3.After that the Kong route need to be defined using a regular expression, containing the fixed part and the variables with type indication, for instance:

```
curl -i -X POST \
--url "http://URL_OF_KONG:8001/services/invoke-quarkus4-service/routes" \
--data-urlencode "paths[]=~/fruits/(?<id>\d+)/(?<name>\S+)" \
--data "strip_path=false"
```

id and name are stored for later use by any Kong plug-in. d+ and S+ means, one or more decimal, and one or more alphanumeric. It is here that you customize the path accordingly with you endpoint application.

G.4.After that you can use the Kong route to invoke your endpoint with parameters.

```
curl -v -i -X PUT -H "accept: application/json" \
--url "http://URL_OF_KONG:8000/fruits/2/bananas"
```

H. Listing all the available services, routes and routes associated with services in Kong

Assuming the example in Section C - Creating a service and a route in Kong, the following command retrieved the available services:

```
curl -i http://localhost:8001/services
```

the answer should be similar to:

```
HTTP/1.1 200 OK
Date: Tue, 14 Jan 2020 21:46:40 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Access-Control-Allow-Origin: *
Server: kong/2.0.0rc2
Content-Length: 1029
X-Kong-Admin-Latency: 1

{"next":null,"data":[{"host":"ec2-52-91-101-30.compute-
1.amazonaws.com","created at":1579036862,"connect timeout":60000,"id":"2158a1c8-4ae4-48e4-a312-
e464b3010104","protocol":"http","name":"invoke-java-
service","read_timeout":60000,"port":9298,"path":"\/calculate","updated_at":1579036862,"retries":5,"write_tim
eout":60000,"tags":null,"client_certificate":null},{"host":"2p3fp5acxj.execute-api.us-east-
1.amazonaws.com","created_at":1579033755,"connect_timeout":60000,"id":"783cfce7-9142-4222-9ee0-
a2b50a507cd3","protocol":"https","name":"invoke-lambda-
service","read timeout":60000,"port":443,"path":"\/default\/PrimeiroMicroServico","updated at":1579033755,"re
tries":5,"write_timeout":60000,"tags":null,"client_certificate":null},{"host":"mockbin.org","created_at":1579
033726,"connect_timeout":60000,"id":"e9f67cc4-fa1e-4a23-ade3-d5e692b7368b","protocol":"http","name":"example-
service","read_timeout":60000,"port":80,"path":null,"updated_at":1579033726,"retries":5,"write_timeout":60000
,"tags":null,"client_certificate":null}]}
```

Then, the following command retrieved the available routes:

```
curl -i http://localhost:8001/routes
```

and the answer should be similar to:

```
HTTP/1.1 200 OK
Date: Tue, 14 Jan 2020 21:47:24 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Access-Control-Allow-Origin: *
Server: kong/2.0.0rc2
Content-Length: 1318
X-Kong-Admin-Latency: 1

{"next":null,"data":[{"id":"38cd55c6-adea-4fc6-86b9-
19ca0749e4a3","path_handling":"v0","paths":null,"destinations":null,"headers":null,"protocols":["http","https
"],"methods":null,"snis":null,"service":{"id":"783cfce7-9142-4222-9ee0-
a2b50a507cd3"},"name":null,"strip_path":true,"preserve_host":false,"regex_priority":0,"updated_at":1579033770
,"sources":null,"hosts":["disciplinas.com"],"https_redirect_status_code":426,"tags":null,"created_at":1579033
770},{"id":"8d188dde-c65a-402b-b659-
d8c39054087f","path_handling":"v0","paths":null,"destinations":null,"headers":null,"protocols":["http","https
"],"methods":null,"snis":null,"service":{"id":"2158a1c8-4ae4-48e4-a312-
e464b3010104"},"name":null,"strip_path":true,"preserve_host":false,"regex_priority":0,"updated_at":1579036920
,"sources":null,"hosts":["calculate.com"],"https_redirect_status_code":426,"tags":null,"created_at":157903692
0},{"id":"cca09713-5cdf-4db1-8ab0-
606e2bd76d1c","path handling":"v0","paths":null,"destinations":null,"headers":null,"protocols":["http","https
"],"methods":null,"snis":null,"service":{"id":"e9f67cc4-fa1e-4a23-ade3-
d5e692b7368b"},"name":null,"strip_path":true,"preserve_host":false,"regex_priority":0,"updated_at":1579033736
,"sources":null,"hosts":["example.com"],"https_redirect_status_code":426,"tags":null,"created_at":1579033736}
]}[
```

Finally, the following command retrieved the available routes for a specific service:

```
curl -i http://localhost:8001/services/example-service/routes
```

and the answer should be similar to:

```
HTTP/1.1 200 OK
Date: Tue, 14 Jan 2020 16:28:47 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Access-Control-Allow-Origin: *
Server: kong/2.0.0rc2
Content-Length: 452
```

```
X-Kong-Admin-Latency: 2

{"next":null,"data":[{"id":"6b7df64c-9ed9-4d5c-87a3-
58b9935d01f4","path handling":"v0","paths":null,"destinations":null,"h
eaders":null,"protocols":["http","https"],"methods":null,"snis":null,"service":{"id":"af50ee81-66e3-4dbe-
b2f6-f25e72c549a1
"},"name":null,"strip_path":true,"preserve_host":false,"regex_priority":0,"updated_at":1579017405,"sources":n
ull,"hosts":[
"example.com"],"https_redirect_status_code":426,"tags":null,"created_at":1579017405}]}
```

I. Install Konga Open Source platform

Optionally, for administration purposes of Kong you can install Konga, which is an unofficial app. No affiliated with Kong. It offer an easy management of all the Kong concepts.

I.1. Pull the following Docker image locally on your PC:

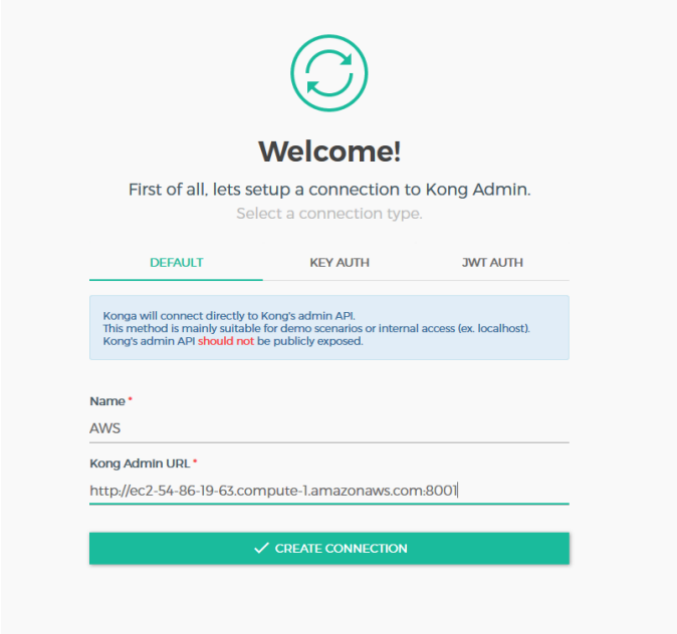
```
docker pull pantsel/konga
```

I.2. Then, execute it:

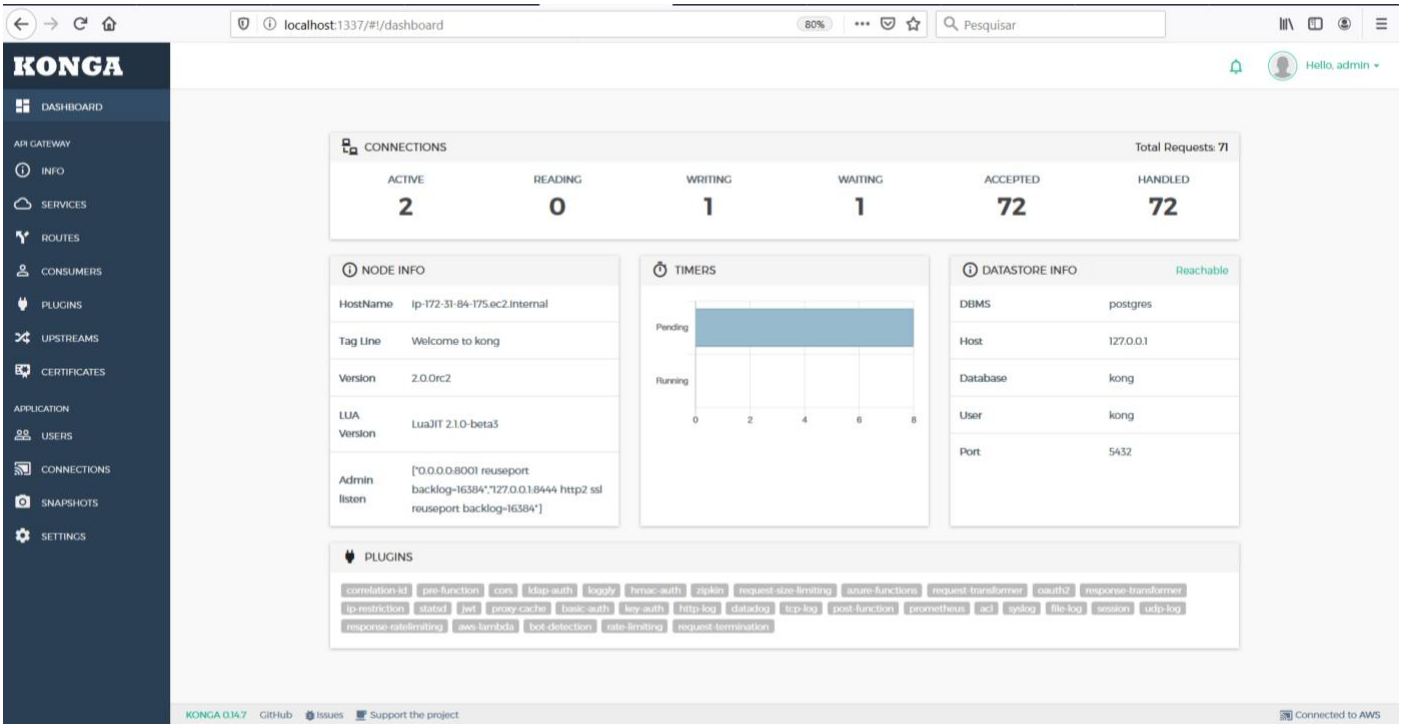
```
docker run -d --name konga -p 1337:1337 pantsel/konga
```

I.3. Open your browser with the following url: http://<ec2_DNS_NAME>:1337 and create an admin account (keep the password!)

I.4. Then, provide the Kong URL with the 8001 port reference in your browser, accordingly with the following example:



A similar result is obtained, where you can manage the Kong concepts:


localhost:1337/#!/dashboard
80%
Pesquisar
KONGA
Hello, admin
DASHBOARD
API GATEWAY
INFO
SERVICES
ROUTES
CONSUMERS
PLUGINS
UPSTREAMS
CERTIFICATES
APPLICATION
USERS
CONNECTIONS
SNAPSHOTS
SETTINGS
CONNECTIONS
Total Requests: 71
ACTIVE
2
READING
0
WRITING
1
WAITING
1
ACCEPTED
72
HANDLED
72
NODE INFO
HostName
ip-172-31-84-175.ec2.internal
Tag Line
Welcome to kong
Version
2.0.0rc2
LUA Version
LuaJIT 2.1.0-beta3
Admin listen
["0.0.0.0:8001 reuseport backlog=16384","127.0.0.1:8444 http2 ssl reuseport backlog=16384"]
TIMERS
Pending
Running
0
2
4
6
8
DATASTORE INFO
Reachable
DBMS
postgres
Host
127.0.0.1
Database
kong
User
kong
Port
5432
PLUGINS
correlation-id
pre-function
cors
ldap-auth
loggly
hmac-auth
zipkin
request-size-limiting
azure-functions
request-transformer
oauth2
response-transformer
ip-restriction
statsd
jwt
proxy-cache
basic-auth
key-auth
http-log
datadog
tcp-log
post-function
prometheus
acl
syslog
file-log
session
udp-log
response-ratelimiting
aws-lambda
bot-detection
rate-limiting
request-termination
KONGA 0.14.7
GitHub
Issues
Support the project
Connected to AWS

Other references

- Kong Inc. https://konghq.com/
- https://docs.konghq.com/install/redhat/?_ga=2.90626585.229207644.1612972455-1733373400.1609865717
- Konga https://pantsel.github.io/konga/
- https://wiki.postgresql.org/wiki/YUM_Installation
- https://www.postgresql.org/download/linux/redhat/
- https://www.cyberciti.biz/faq/install-and-setup-postgresql-on-rhel-8/
- https://docs.konghq.com/1.4.x/admin-api/
- For deploying Auth2.0 plug/in in Kong read the manual: https://docs.konghq.com/hub/kong-inc/oauth2/
- https://docs.konghq.com/gateway/2.8.x/install-and-run/docker/