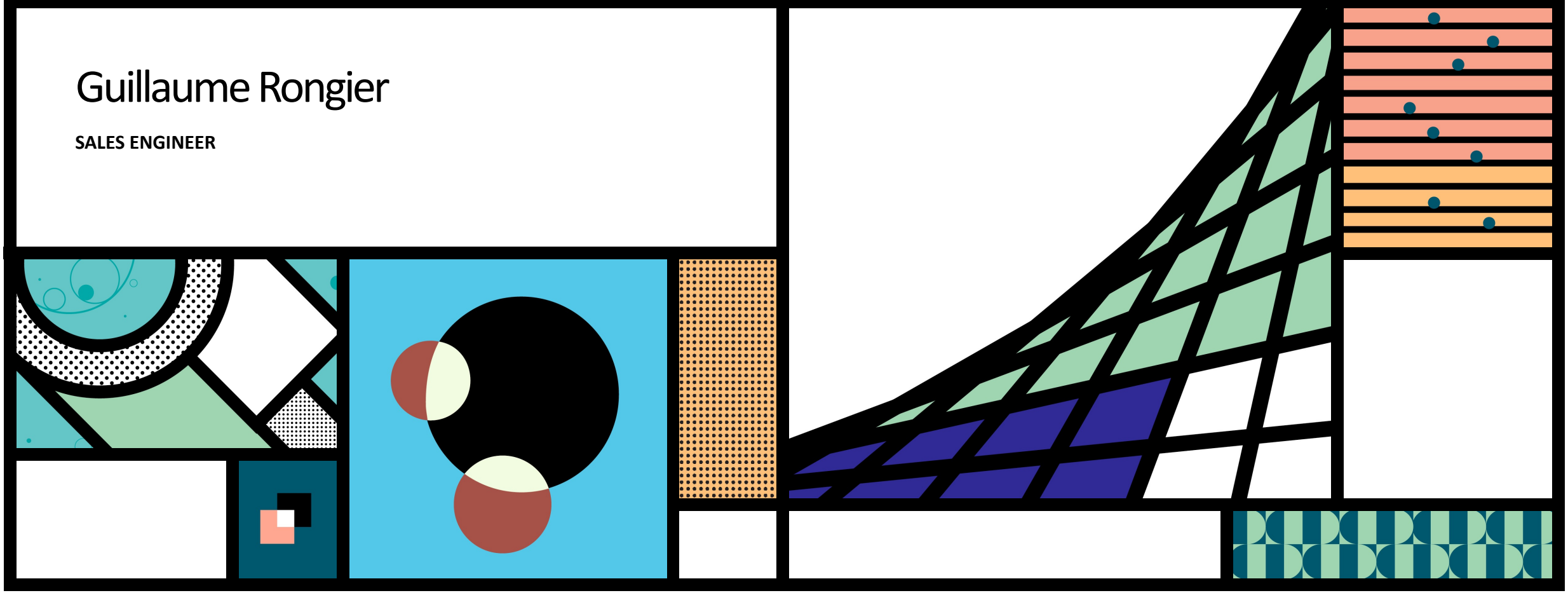
IAM Training

InterSystems®
Health | Business | Government

Guillaume Rongier

SALES ENGINEER



Index

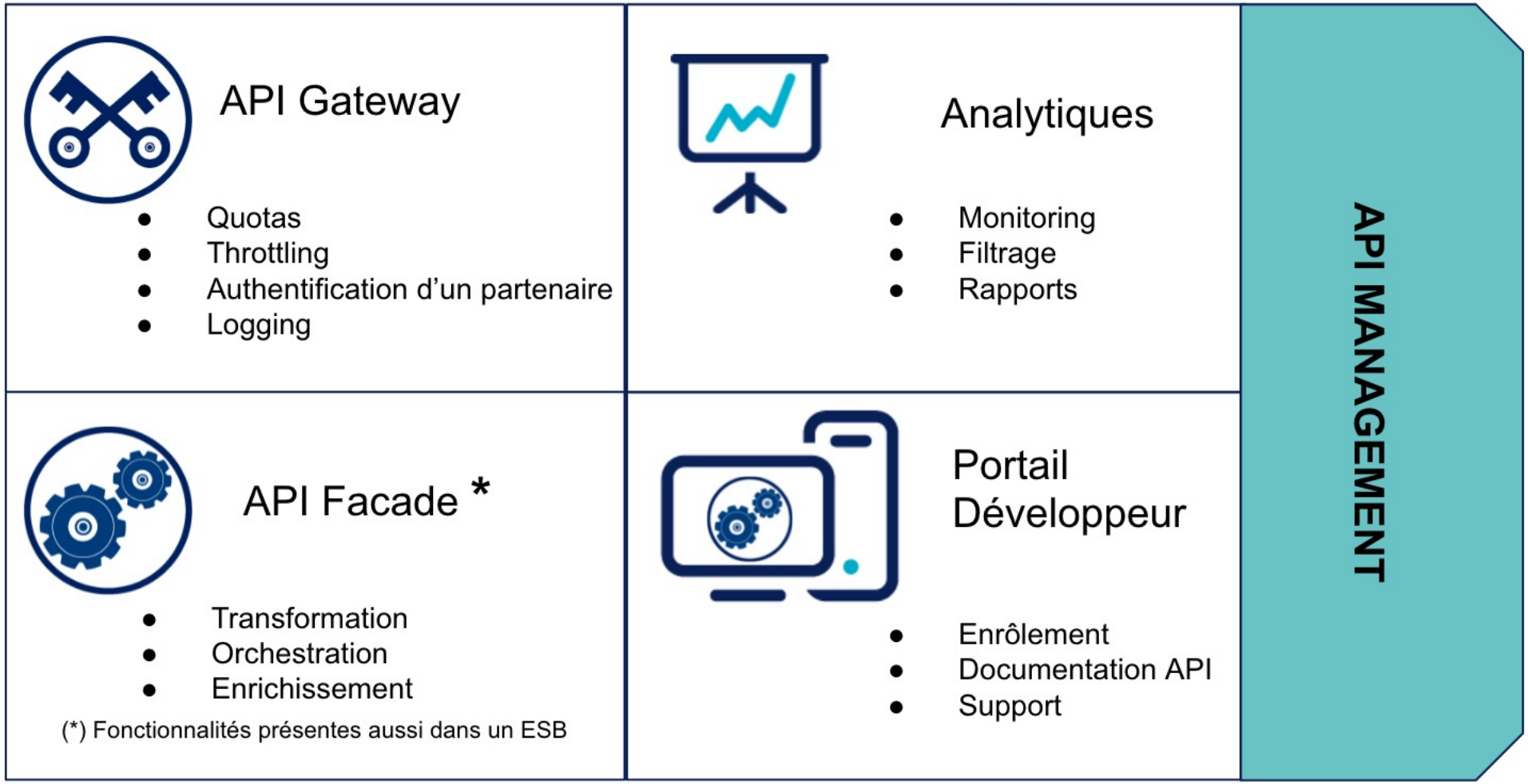
Index

Options :

- Import a custom plugin
- Build a custom plugin
- DevOps

Introduction

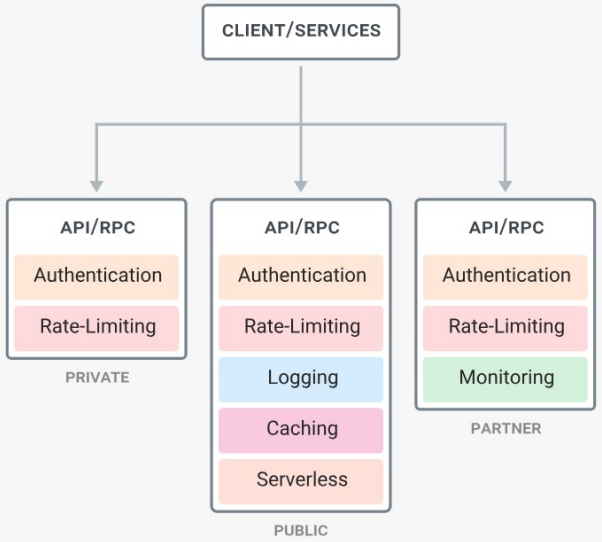
Introduction to IAM aka KONG



 | |

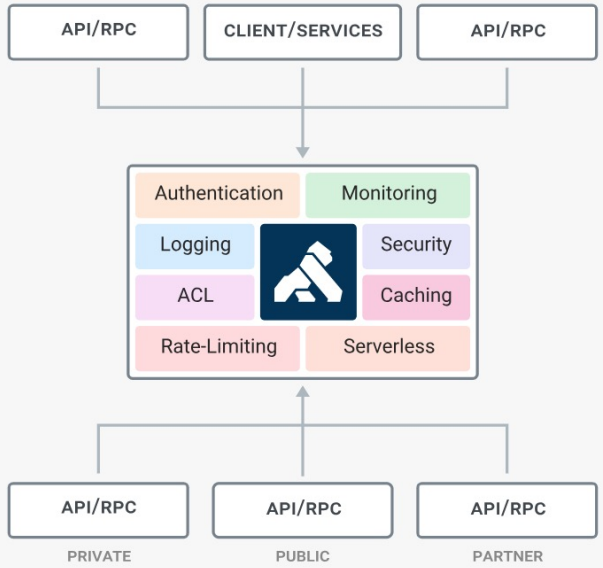
Kong way

The Redundant Old Way



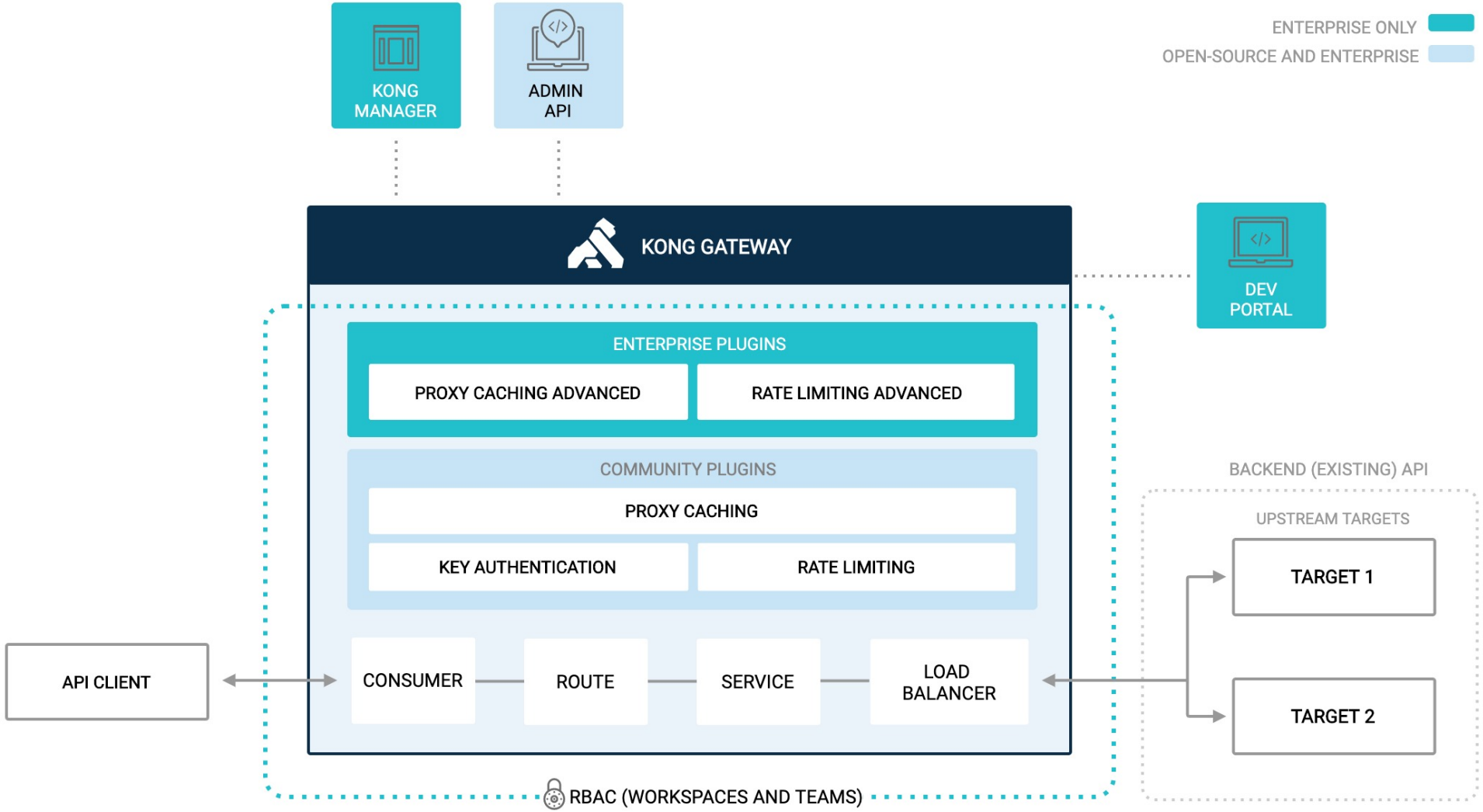
- ✗ Common functionality is duplicated across multiple services
- ✗ Systems tend to be monolithic and hard to maintain
- ✗ Difficult to expand without impacting other services
- ✗ Productivity is inefficient because of system constraints

The Kong Way



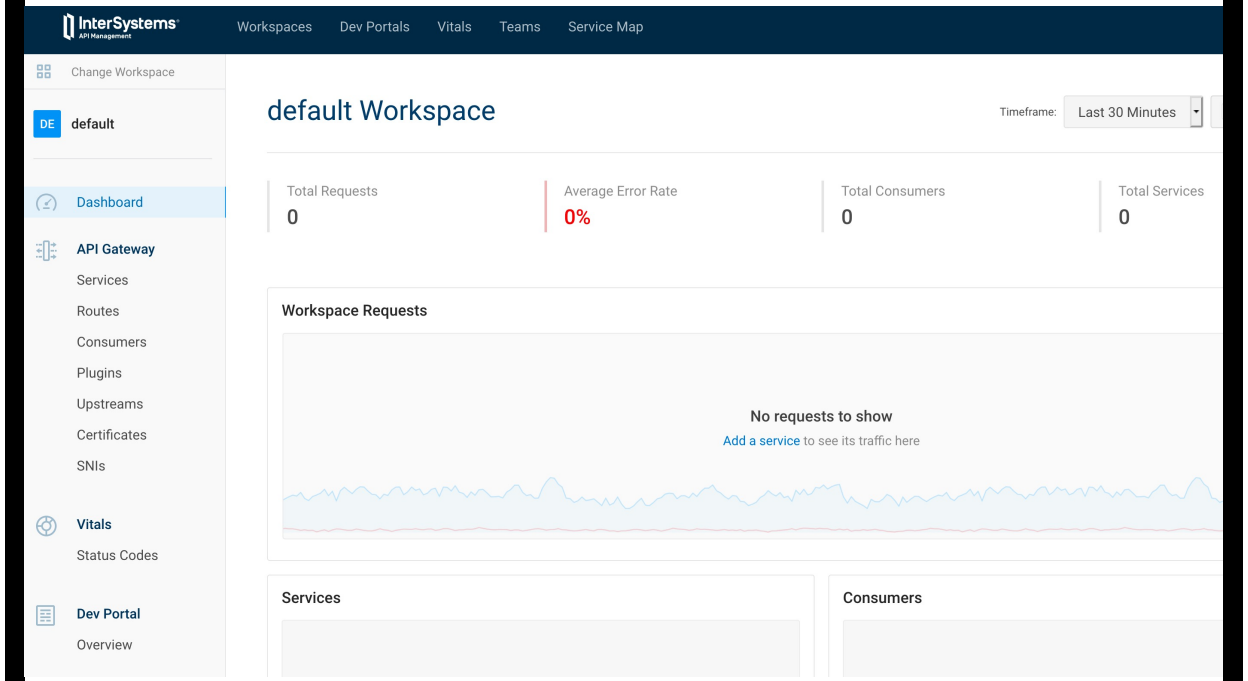
- ✓ Kong orchestrates common functionality
- ✓ Build efficient distributed architectures ready to scale
- ✓ Expand functionality from one place with a simple command
- ✓ Focus on your product and let Kong do the REST

Kong overview



 | |

IAM Configuration

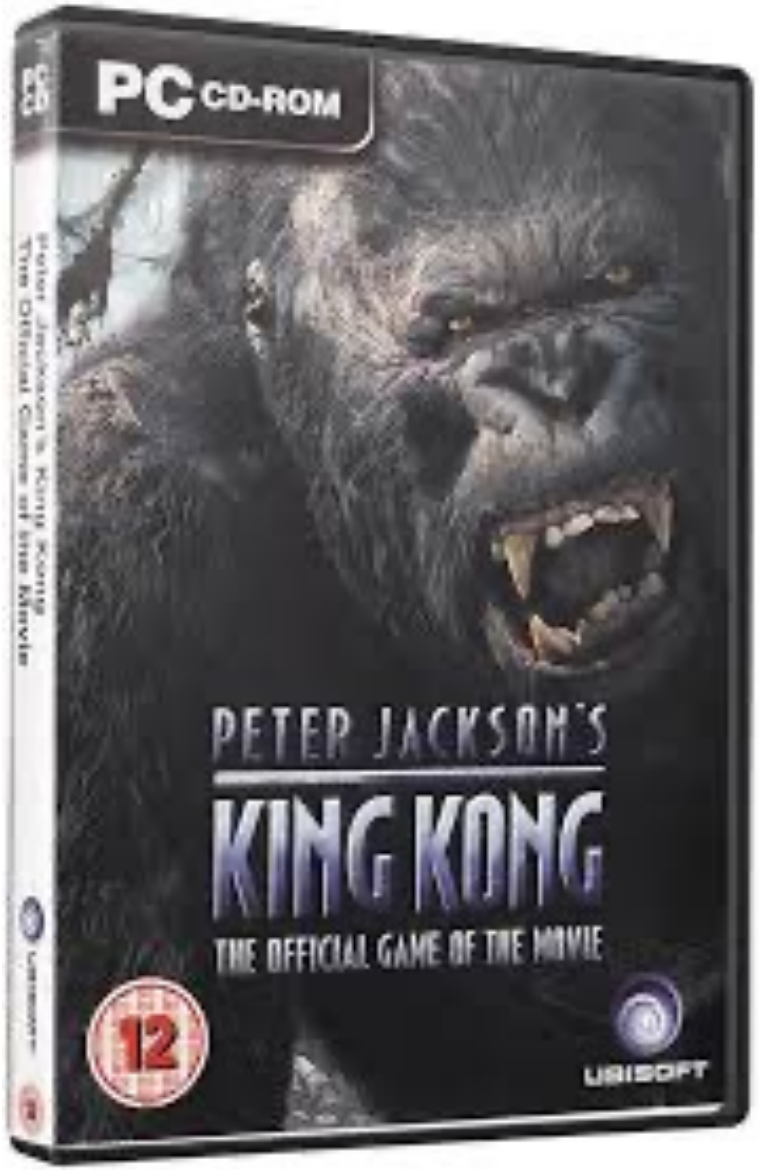



```
curl -i -X POST \
      --url http://localhost:8001/apis/nooi-store/plugins/ \
      --data 'name=key-auth'
```

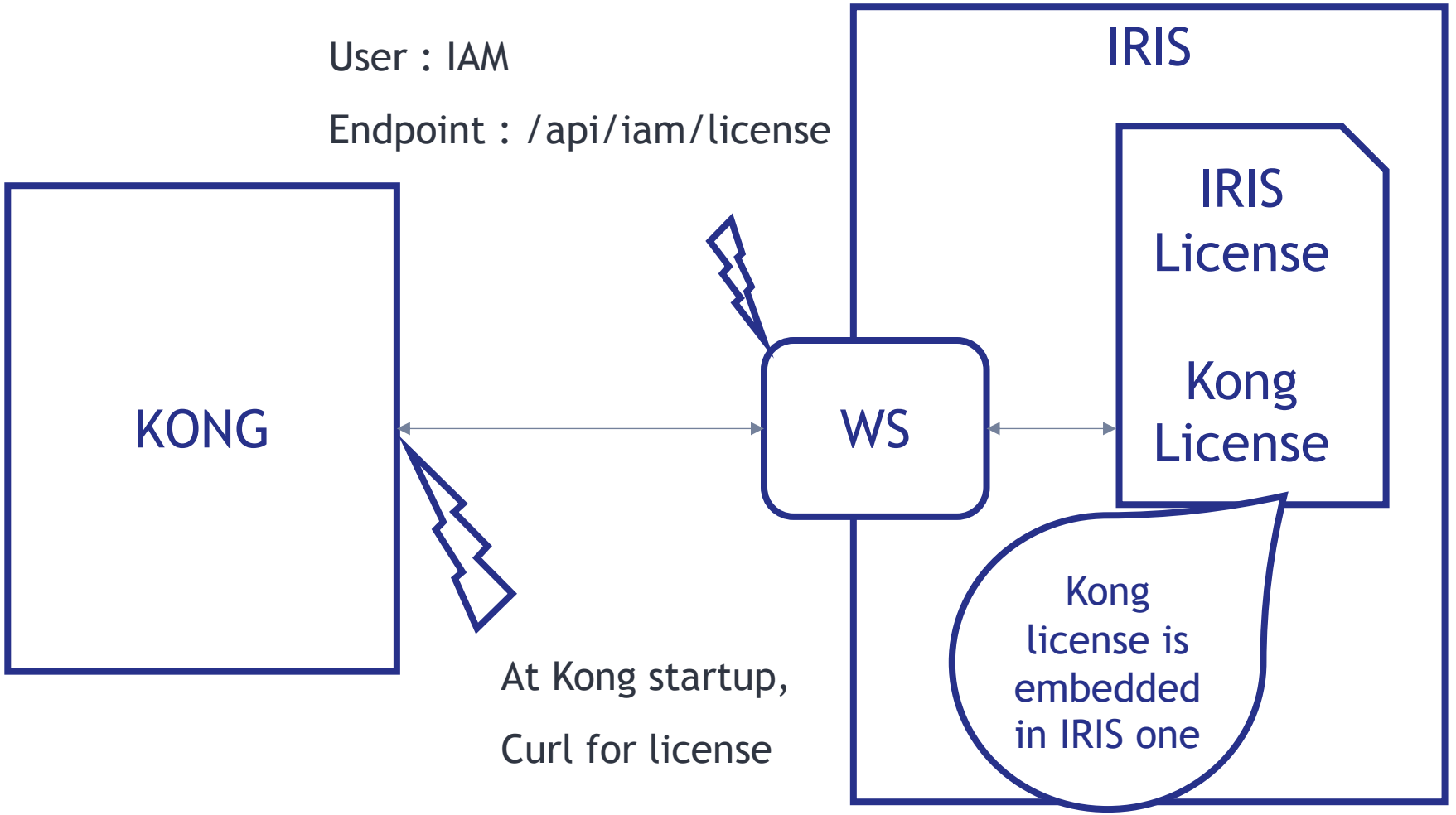
Adresse:port du serveur — Les APIs — L'API cible — Ses plugins

Nom du plugin

Installation



How kong work's with IRIS



 | |

Pre-requis

Git

Docker

Iris.key

---

https://container.intersystems.com/

Iam docker image : containers.intersystems.com/intersystems/iam:1.5.0.9-4

Iris docker image : containers.intersystems.com/intersystems/irishealth:2020.4.0.524.0

Git clone / fork

Template available here :

https://github.com/grongierisc/iam-training

Main -> to follow training

Training -> corrections

What's in ?

An IRIS back-end with a simple rest api with pre-install data.

http://localhost:52773/swagger-ui/index.html#/

What we have to do, to install kong ?

Change IRIS community edition to a licensed one.

- containers.intersystems.com/intersystems/irishealth:2020.4.0.524.0
- Add iris.key in key folder

```
ARG
IMAGE=containers.intersystems.com/intersystems/irishealth:2020.4.0.524.0

FROM $IMAGE as iris-iam

COPY key/iris.key
/usr/irissys/mgr/iris.key

COPY iris-iam.script /tmp/iris-iam.script


RUN iris start IRIS \

&& iris session IRIS < /tmp/iris-iam.script \

&& iris stop IRIS quietly


FROM iris-iam

... <- old dockerfile
```


What we have to do, to install kong ?

Change IRIS community edition to a licensed one.

- containers.intersystems.com/intersystems/irishealth:2020.4.0.524.0
- Add iris.key in key folder

Enable IAM endpoint and USER in IRIS

- Build a new IRIS image with this new config
- Create an iris-iam.script
  - Add key
  - Enable user
  - Enable endpoint

```
zn "%SYS"

wri"Create web application ...",!

set webName = "/api/iam"

set webProperties("Enabled") = 1

set status =
##class(Security.Applications).Modify(webName,
.webProperties)

write:'status $system.Status.DisplayError(status)

write "Web application "_webName_" was updated!",!

set userProperties("Enabled") = 1

set userName = "IAM"

Do
##class(Security.Users).Modify(userName,.userPrope
rties)

write "User "_userName_" was updated!",!

halt
```

 | |

What we have to do, to install kong ?

Change IRIS community edition to a licensed one.

- containers.intersystems.com/intersystems/irishealth:2020.4.0.524.0
- Add iris.key in key folder

Enable IAM endpoint and USER in IRIS

- Build a new IRIS image with this new config
- Create an iris-iam.script
  - Add key
  - Enable user
  - Enable endpoint

Add IAM in docker-compose

- db
  - Postgres database for IAM
- Iam-migration
  - Bootstrap the database
- Iam
  - Actual IAM instance
- A volume for data persistent

```
iam-migrations: crafter         db              rld,jwt-crafter failure
image:          ISC_IRIS_URL:   KONG_DATABASE:  ISC_IRIS_URL:   db:
intersystems/ia IAM:${IRIS_PASS postgres        IAM:${IRIS_PASS image:
m-              WORD}@iris:5277 KONG_PG_DATABAS WORD}@iris:5277 postgres:9.6
custom:1.5.0.9- 3/api/iam/licen E:              3/api/iam/licen environment:
4               se              ${KONG_PG_DATAB se              POSTGRES_DB:
build:          restart: on-    ASE:-iam}       volumes:        ${KONG_PG_DATAB
context: iam    failure         KONG_PG_HOST:   - ./iam:/iam    ASE:-iam}
dockerfile:     links:          db              links:          POSTGRES_PASSWO
dockerfile      - db:db         KONG_PG_PASSWOR - db:db         RD:
command: kong   iam:            D:              ports:          ${KONG_PG_PASSW
migrations      image:          ${KONG_PG_PASSW - target: 8000  ORD:-iam}
bootstrap up    intersystems/ia ORD:-iam}       published: 8000 POSTGRES_USER:
depends_on:     m-              KONG_PG_USER:   protocol: tcp   ${KONG_PG_USER:
- db            custom:1.5.0.9- ${KONG_PG_USER: - target: 8001  -iam}
environment:    4               -iam}           published: 8001 volumes:
KONG_DATABASE:  build:          KONG_PROXY_ACCE protocol: tcp   -
postgres        context: iam    SS_LOG:         - target: 8002  'pgdata:/var/li
KONG_PG_DATABAS dockerfile:     /dev/stdout     published: 8002 b/postgresql/da
E:              dockerfile      KONG_PROXY_ERRO protocol: tcp   ta'
${KONG_PG_DATAB depends_on:     R_LOG:          - target: 8003  healthcheck:
ASE:-iam}       - db            /dev/stderr     published: 8003 test: ["CMD",
KONG_PG_HOST:   environment:    KONG_PORTAL:    protocol: tcp   "pg_isready",
db              KONG_ADMIN_ACCE 'on'            - target: 8004  "-U",
KONG_PG_PASSWOR SS_LOG:         KONG_PORTAL_GUI published: 8004 "${KONG_PG_USER
D:              /dev/stdout     _PROTOCOL: http protocol: tcp   :-iam}"]
${KONG_PG_PASSW KONG_ADMIN_ERRO KONG_PORTAL_GUI - target: 8443  interval: 30s
ORD:-iam}       R_LOG:          _HOST:          published: 8443 timeout: 30s
KONG_PG_USER:   /dev/stderr     '127.0.0.1:8003 protocol: tcp   retries: 3
${KONG_PG_USER: KONG_ADMIN_LIST '               - target: 8444  restart: on-
-iam}           EN:             KONG_ADMIN_GUI_ published: 8444 failure
KONG_CASSANDRA_ '0.0.0.0:8001'  URL:            protocol: tcp   stdin_open:
CONTACT_POINTS: KONG_ANONYMOUS_ http://localhos - target: 8445  true
db              REPORTS: 'off'  t:8002          published: 8445 tty: true
KONG_PLUGINS:   KONG_CASSANDRA_ KONG_PLUGINS:   protocol: tcp   volumes:
bundled,jwt-    CONTACT_POINTS: bundled,hellowo restart: on-    pgdata:
```


Option : IRIS_PASSWORD

How to paramètre IRIS_PASSWORD for Kong and IRIS.

We can use .env file to do so, then use ${env_var} to use our new password.

```
build:
context: .
dockerfile: dockerfile
args:
- IRIS_PASSWORD=${IRIS_PASSWORD}
```

```
RUN echo "${IRIS_PASSWORD}" >
/tmp/password.txt &&
/usr/irissys/dev/Container/changePa
ssword.sh /tmp/password.txt
```


Test It !

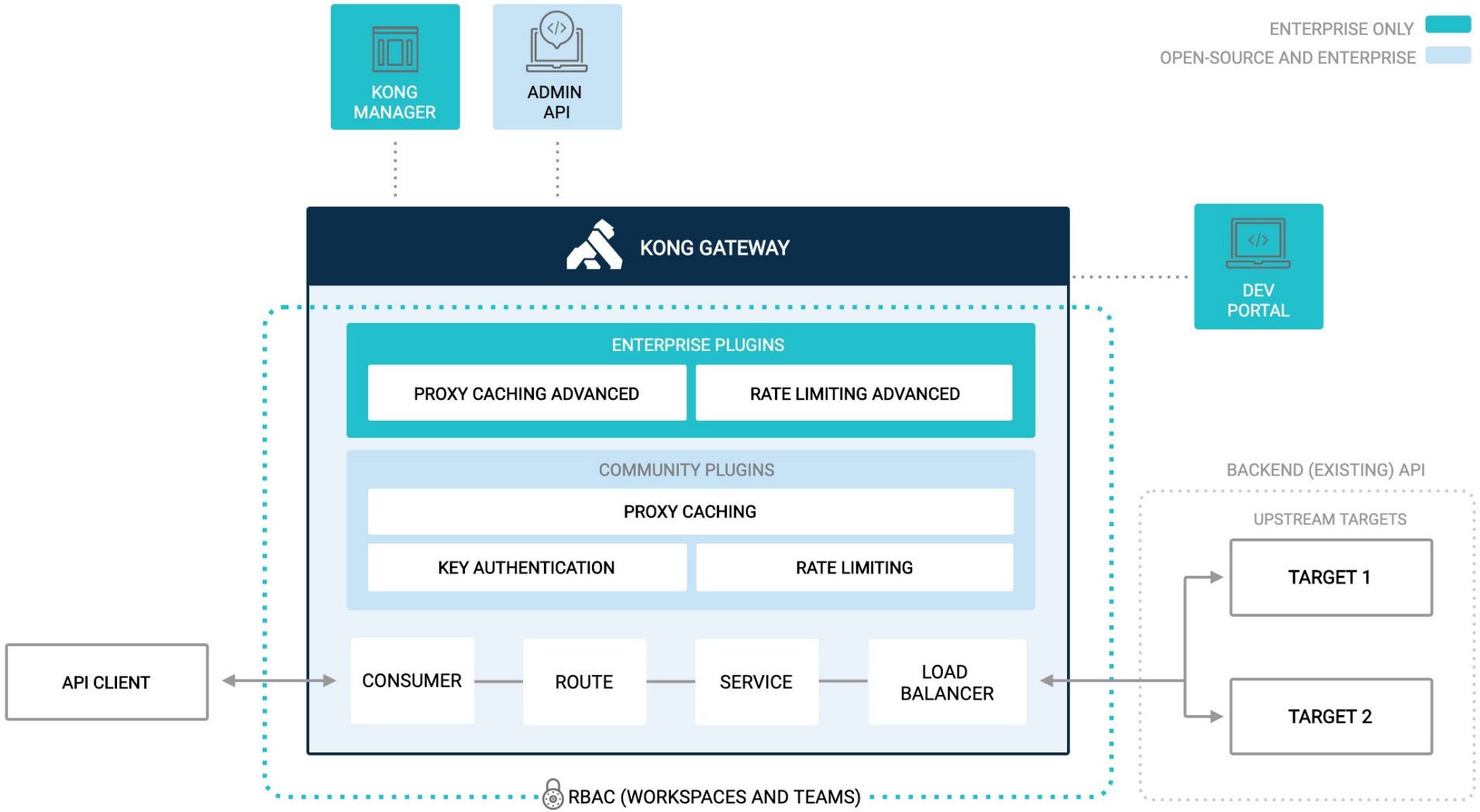
docker-compose -f "docker-compose.yml" up -d –build

Go to http://localhost:8002 ← Kong/IAM portal

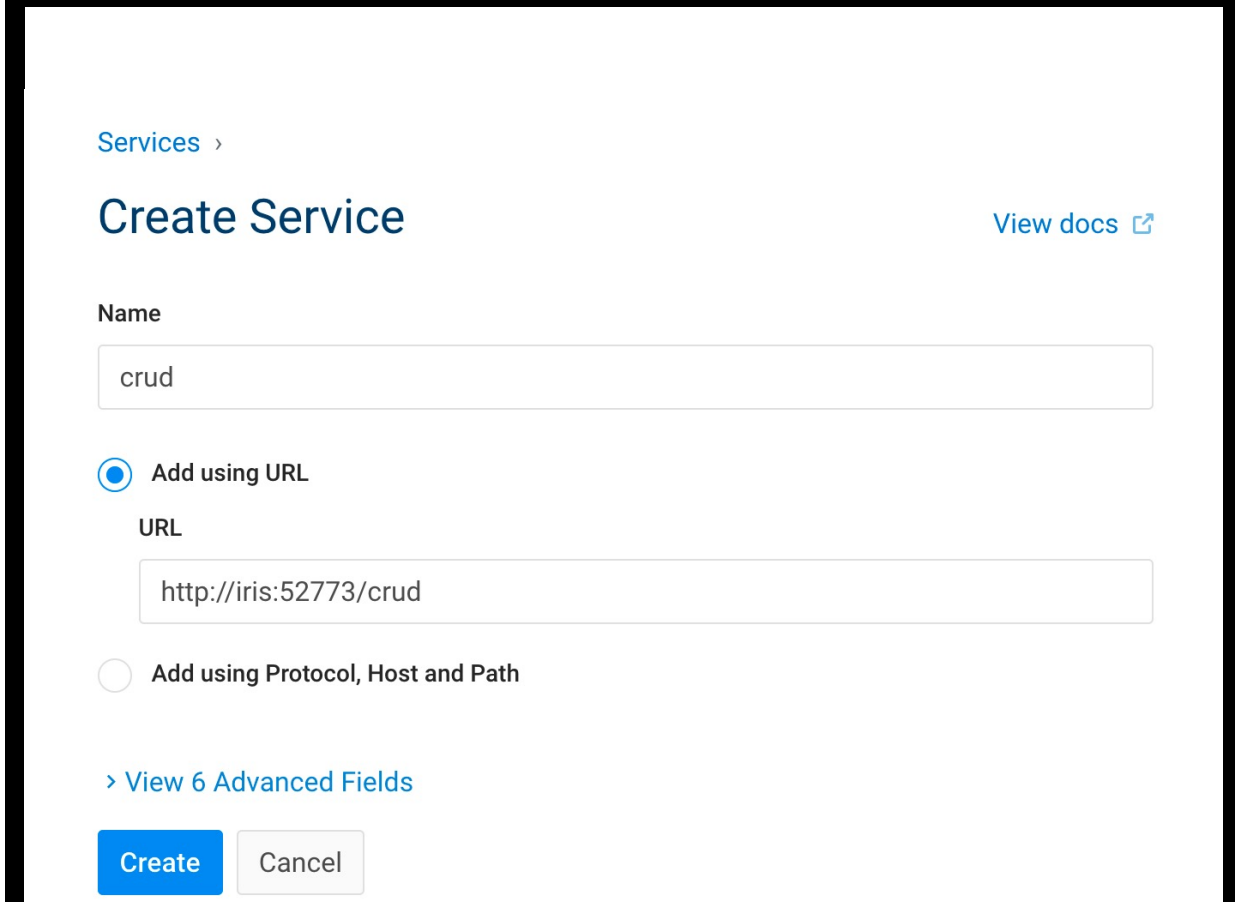
Port	Protocol	Description
:8000	HTTP	Takes incoming HTTP traffic from **Consumers**, and forwards it to upstream **Services**.
:8443	HTTPS	Takes incoming HTTPS traffic from **Consumers**, and forwards it to upstream **Services**.
:8001	HTTP	Admin API. Listens for calls from the command line over HTTP.
:8444	HTTPS	Admin API. Listens for calls from the command line over HTTPS.
:8002	HTTP	Kong Manager (GUI). Listens for HTTP traffic.
:8445	HTTPS	Kong Manager (GUI). Listens for HTTPS traffic.
:8003	HTTP	Dev Portal. Listens for HTTP traffic, assuming Dev Portal is **enabled**.
:8446	HTTPS	Dev Portal. Listens for HTTPS traffic, assuming Dev Portal is **enabled**.
:8004	HTTP	Dev Portal /files traffic over HTTP, assuming the Dev Portal is **enabled**.
:8447	HTTPS	Dev Portal /files traffic over HTTPS, assuming the Dev Portal is **enabled**.

First Service / Route / Plugin

First proxy

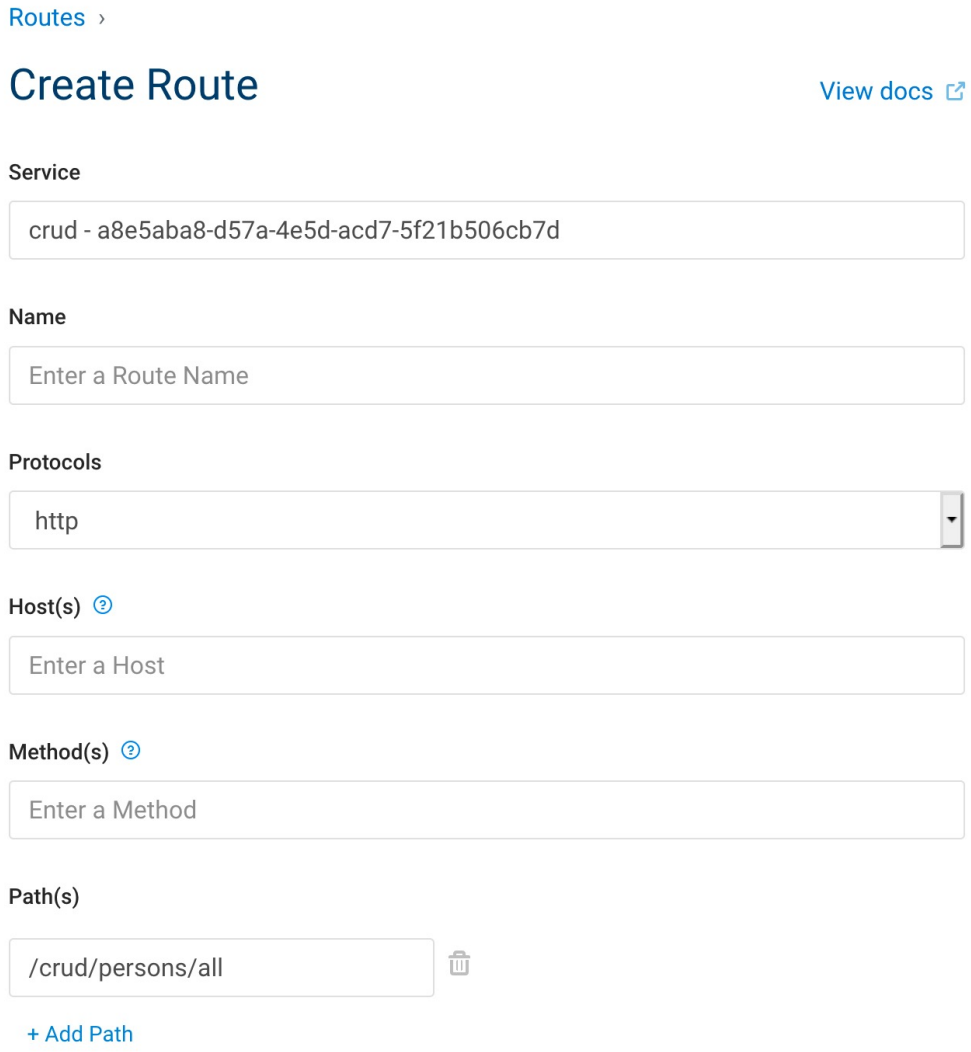


Create Service




```
# Create service


curl -i -X POST \

--url
http://localhost:8001/services/
\

--data 'name=crud' \

--data
'url=http://iris:52773/crud/'
```

Create Route




```
# Create route
curl -i -X POST \
--url
http://localhost:8001/services/
crud/routes \
--data 'name=crud-route' \
--data 'paths=/persons/*' \
--data 'strip_path=false'
```

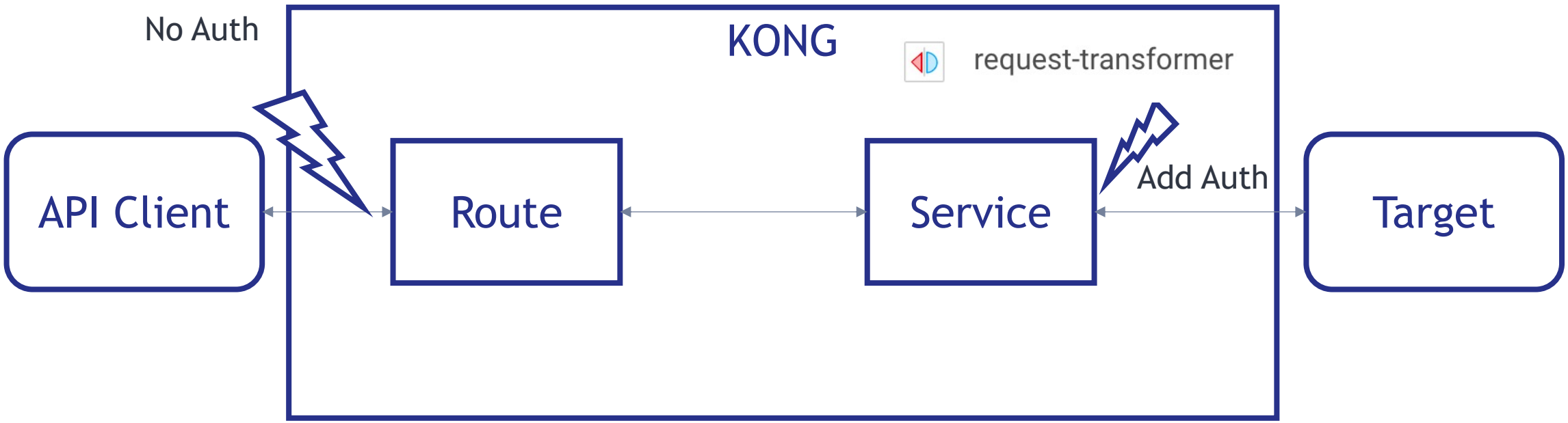
 | |

Test It !

```
# Legacy

curl -i --location --request
GET
'http://localhost:52773/crud/
persons/all' \

--header 'Authorization:
Basic U3VwZXJVc2VyOlNZUw=='
```

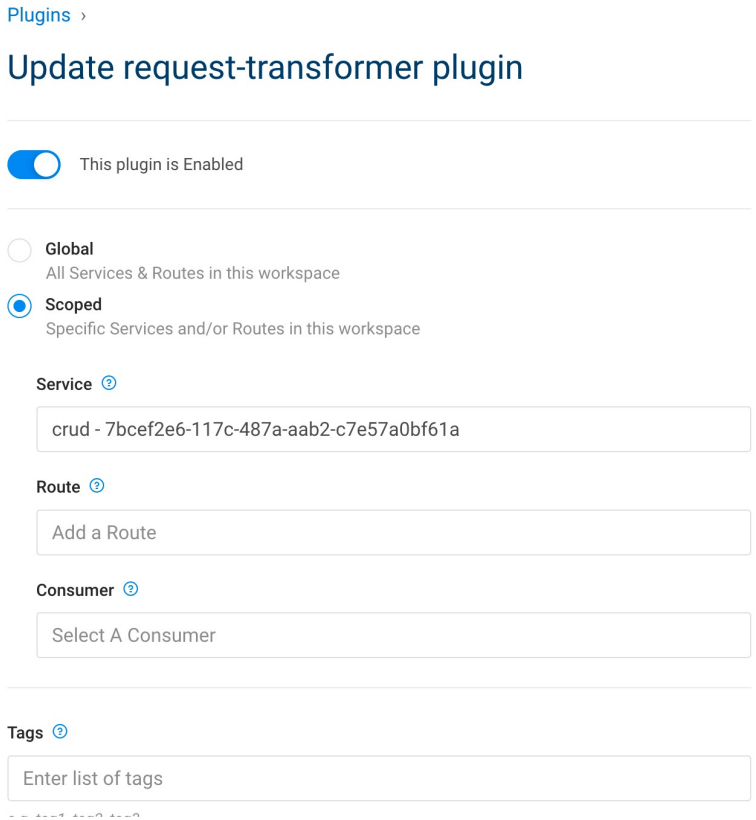
```
# KONG

curl -i --location --request
GET
'http://localhost:8000/person
s/all' \

--header 'Authorization:
Basic U3VwZXJVc2VyOlNZUw=='
```

Step Two, auto - authenticate



 | |

Step Two, auto - authenticate



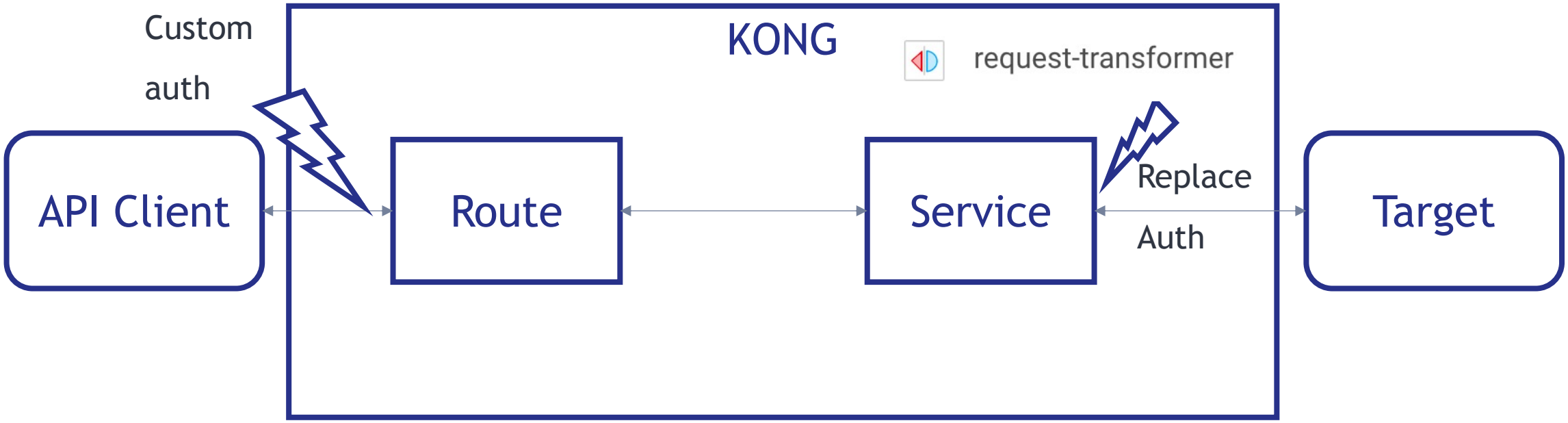

```
# Create plugin

curl -i -X POST \

--url
http://localhost:8001/services/crud
/plugins \

--data 'name=request-transformer' \

--data
'config.add.headers=Authorization:B
asic U3VwZXJVc2VyOlNZUw==' \

--data
'config.replace.headers=Authorizati
on:Basic U3VwZXJVc2VyOlNZUw=='
```


Test It !

```
# Legacy


curl -i --location --request
GET
'http://localhost:52773/crud/
persons/all'
```

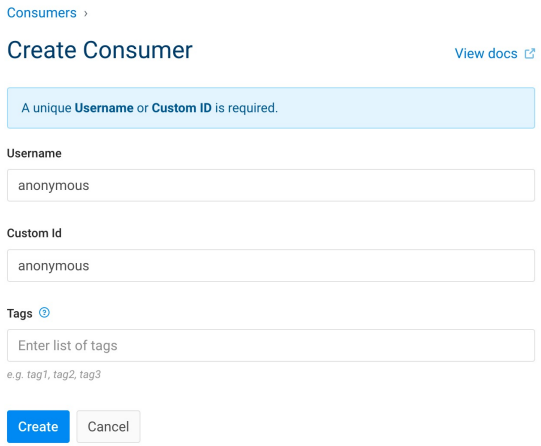
```
# KONG


curl -i --location --request
GET
'http://localhost:8000/person
s/all'
```

Step Three, Add our own authentication

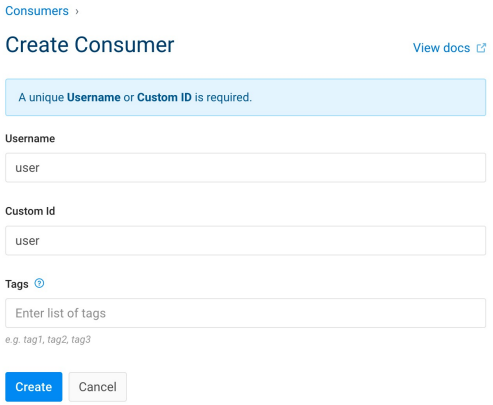


Add Consumers

Anonymous

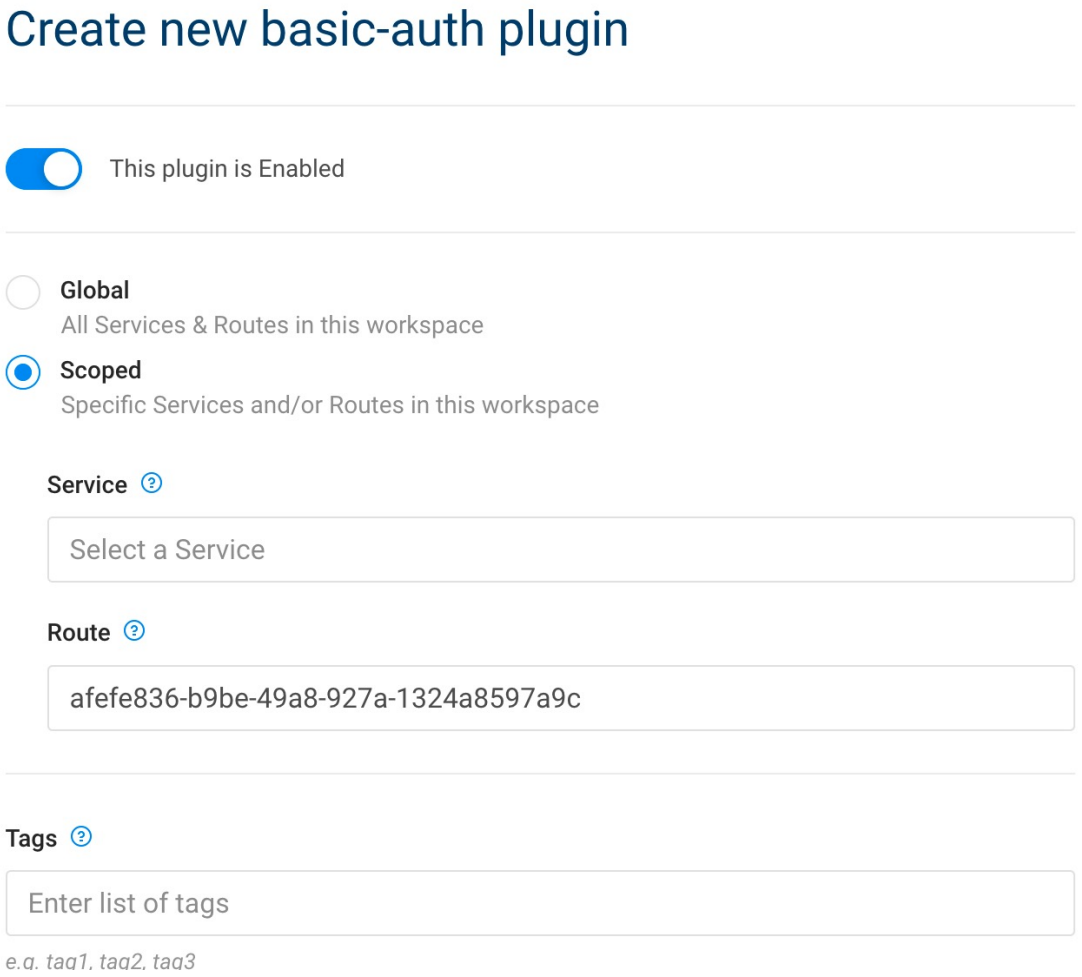


User




```
# Add consumer anonymous
curl -i -X POST \
--url http://localhost:8001/consumers/ \
--data "username=anonymous" \
--data "custom_id=anonymous"

# Add consumer user
curl -i -X POST \
--url http://localhost:8001/consumers/ \
--data "username=user" \
--data "custom_id=user"
```

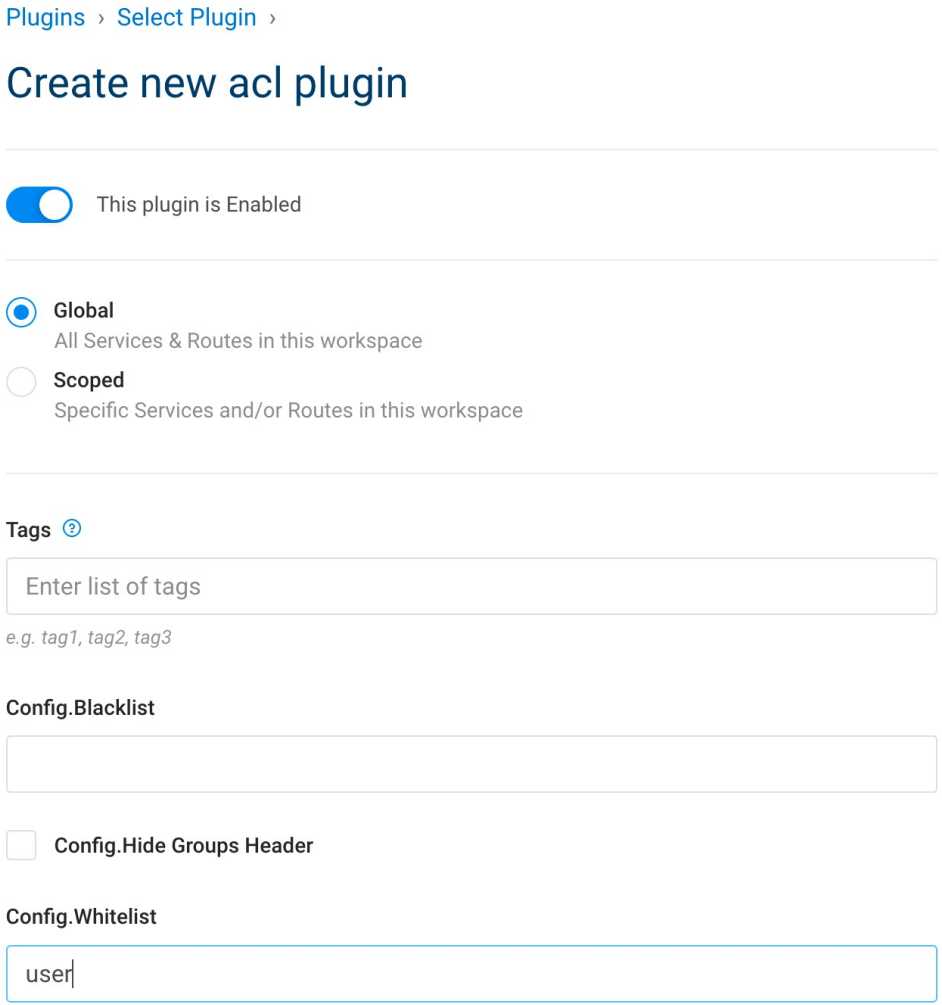
Add Basic auth plugin




```
# Enable basic auth for service

curl -i -X POST
http://localhost:8001/routes/cr
ud-route/plugins \

--data "name=basic-auth" \

--data
"config.anonymous=5cc8dee4-
066d-492e-b2f8-bd77eb0a4c86" \

--data
"config.hide_credentials=false"
```

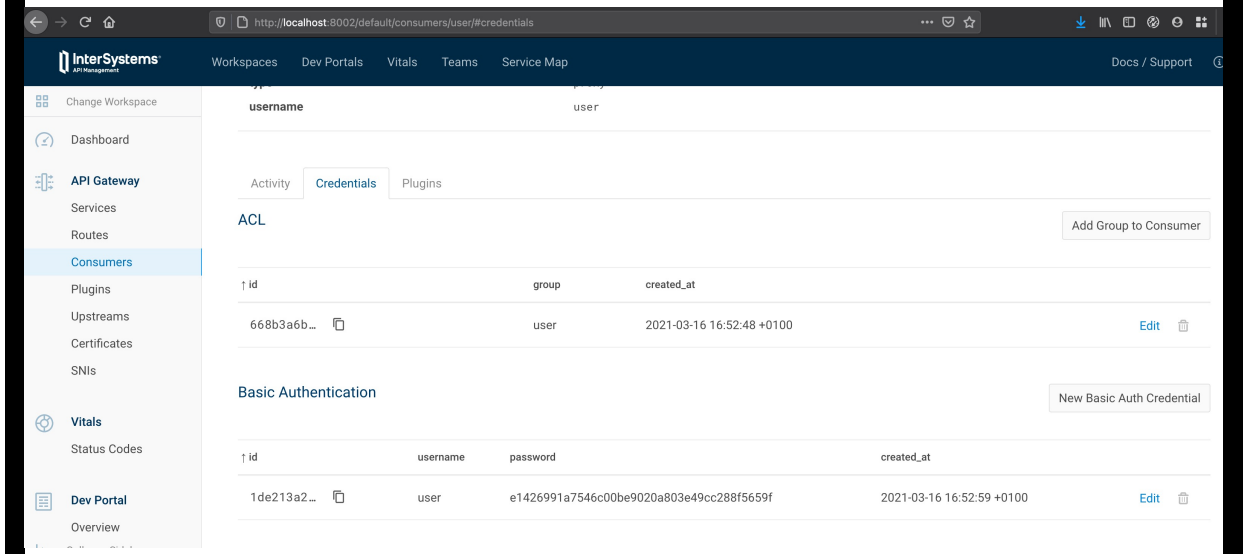
 | |

Add ACL plugin




```
# Enable ACL


curl -i -X POST
http://localhost:8001/routes/cr
ud-route/plugins \
--data "name=acl" \
--data "config.whitelist=user"
```


Configure USER with ACL and credentials




```
# Add consumer group

curl -i -X POST \

--url
http://localhost:8001/consumers/use
r/acls \

--data "group=user"


# Add consumer credentials

curl -i -X POST
http://localhost:8001/consumers/use
r/basic-auth \

--data "username=user" \

--data "password=user"
```


Test It !

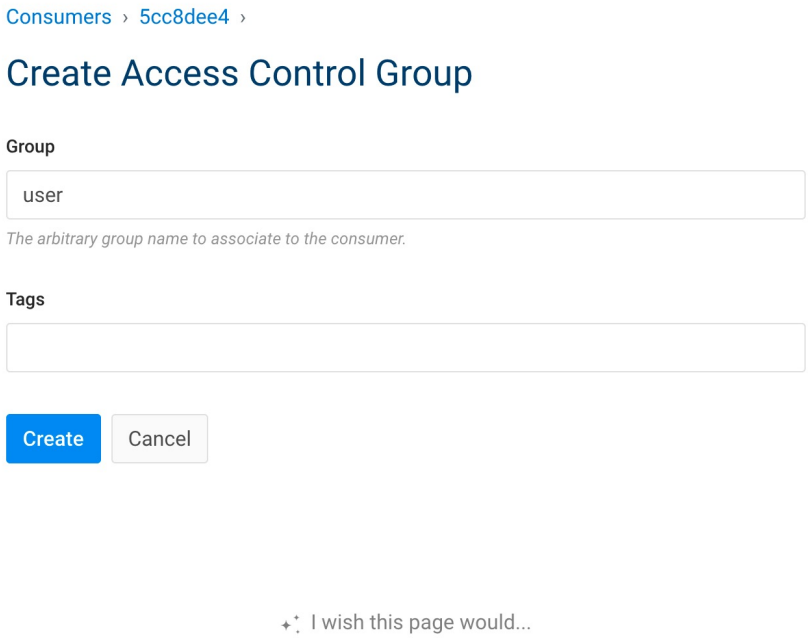
```
# Legacy

curl -i --location --request
GET
'http://localhost:52773/crud/
persons/all' \

--header 'Authorization:Basic
dXNlcjp1c2Vy'
```

```
# KONG

curl -i --location --request
GET
'http://localhost:8000/person
s/all' \

--header 'Authorization:Basic
dXNlcjp1c2Vy'
```

Exercices

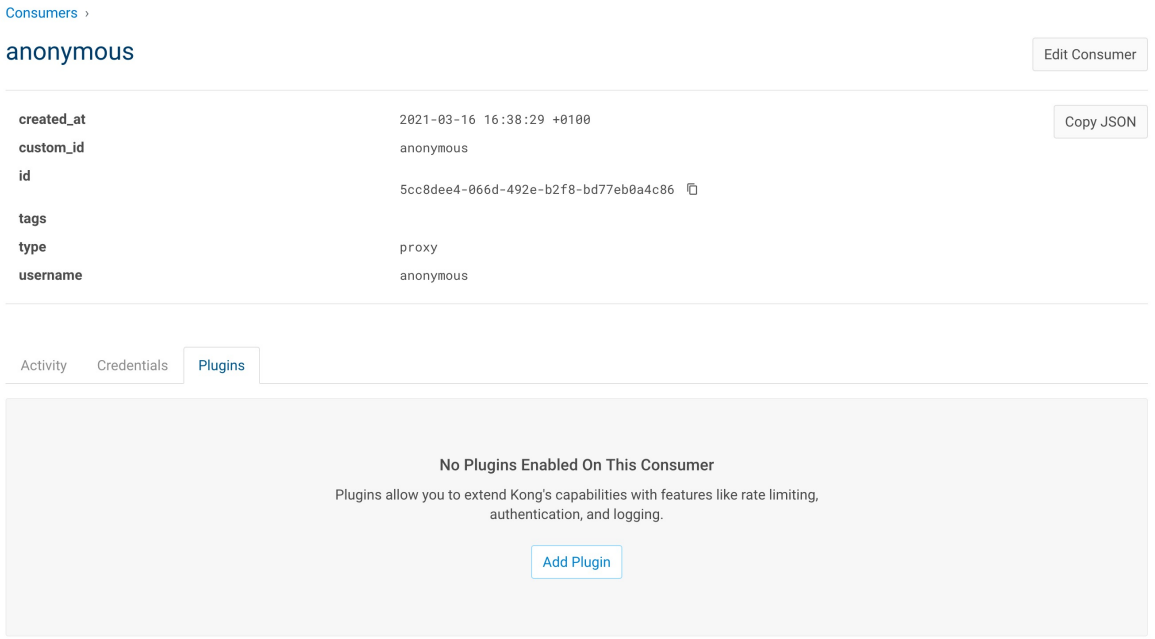
1. Enable Unauthenticated user
2. Limite rate by 2 calls per minutes to Unauthenticated user

Exercices




```
# Add consumer group


curl -i -X POST \

--url
http://localhost:8001/consumers
/anonymous/acls \

--data "group=user"
```


Exercices



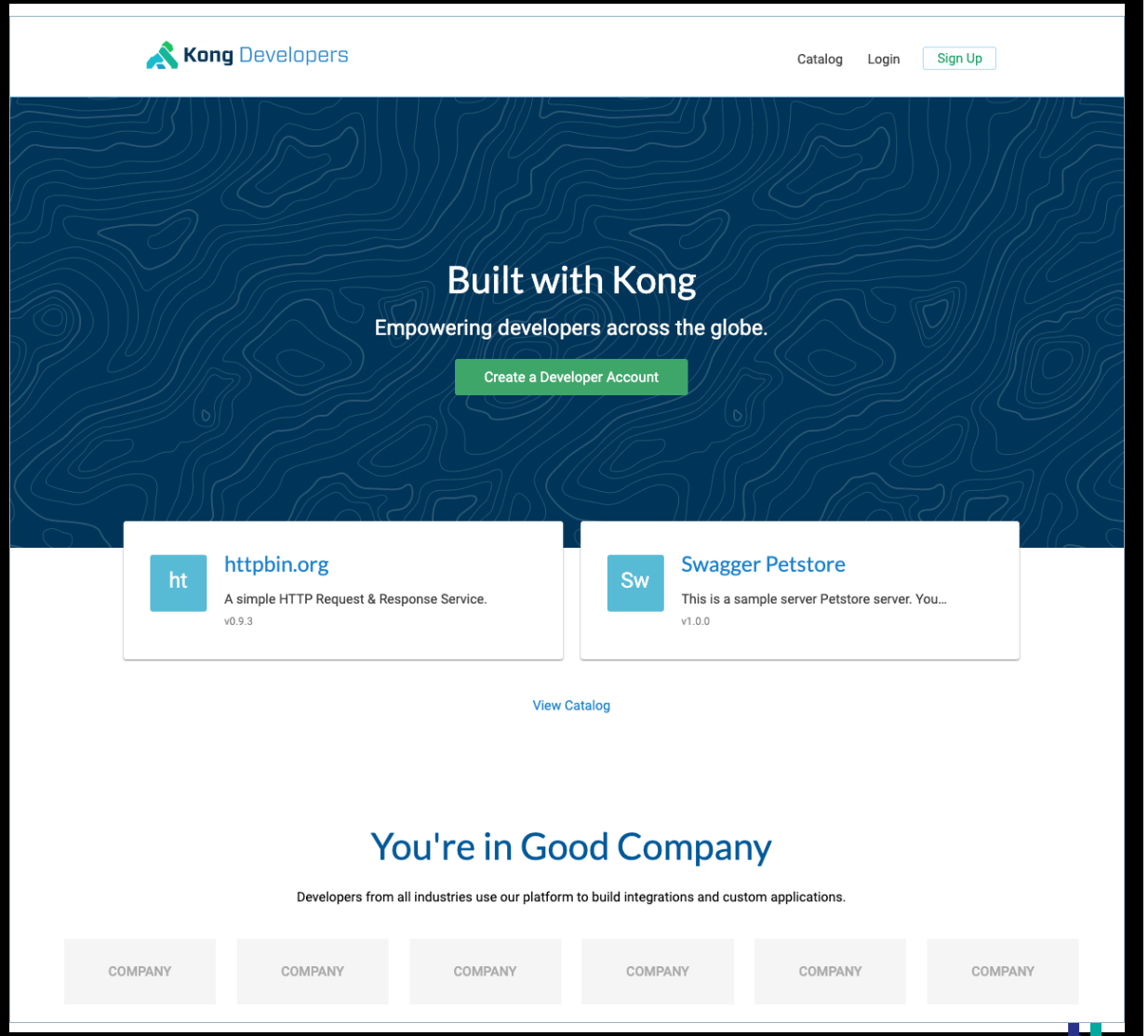

```
# Add rate limite consumer

curl -i -X POST \

--url
http://localhost:8001/consumers
/anonymous/plugins \

--data "name=rate-limiting" \

--data
"config.limit_by=consumer" \

--data "config.minute=2"
```


Dev Portal

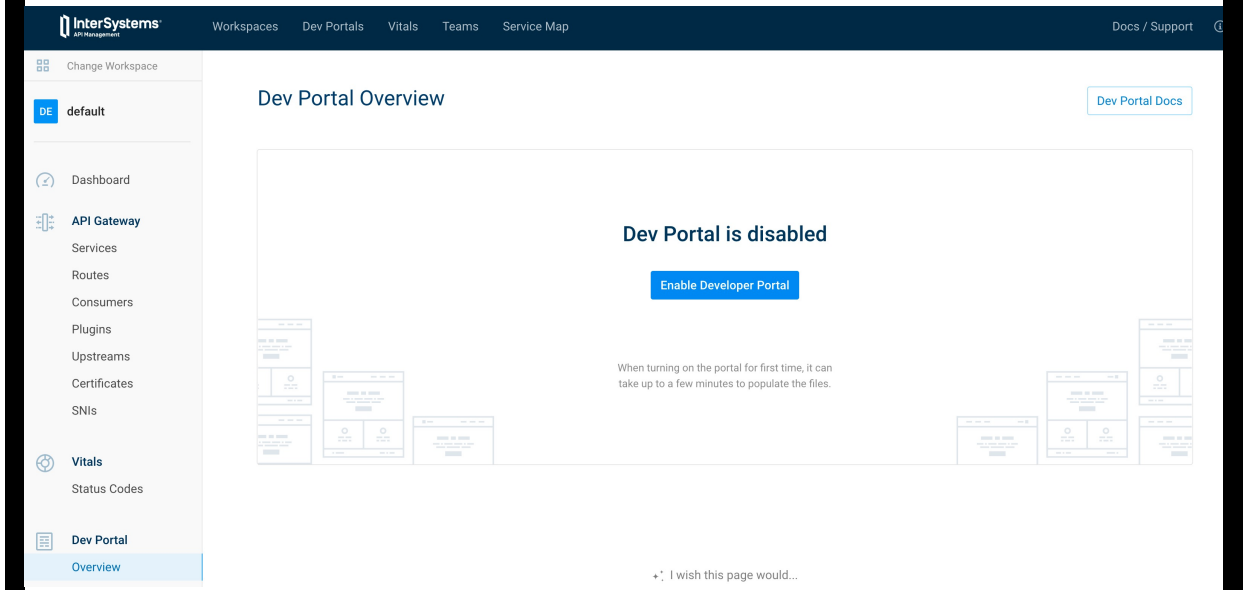
Overview

The Kong Developer Portal provides :

- a single source of truth for all developers
- intuitive content management for documentation
- streamlined developer onboarding
- role-based access control (RBAC)

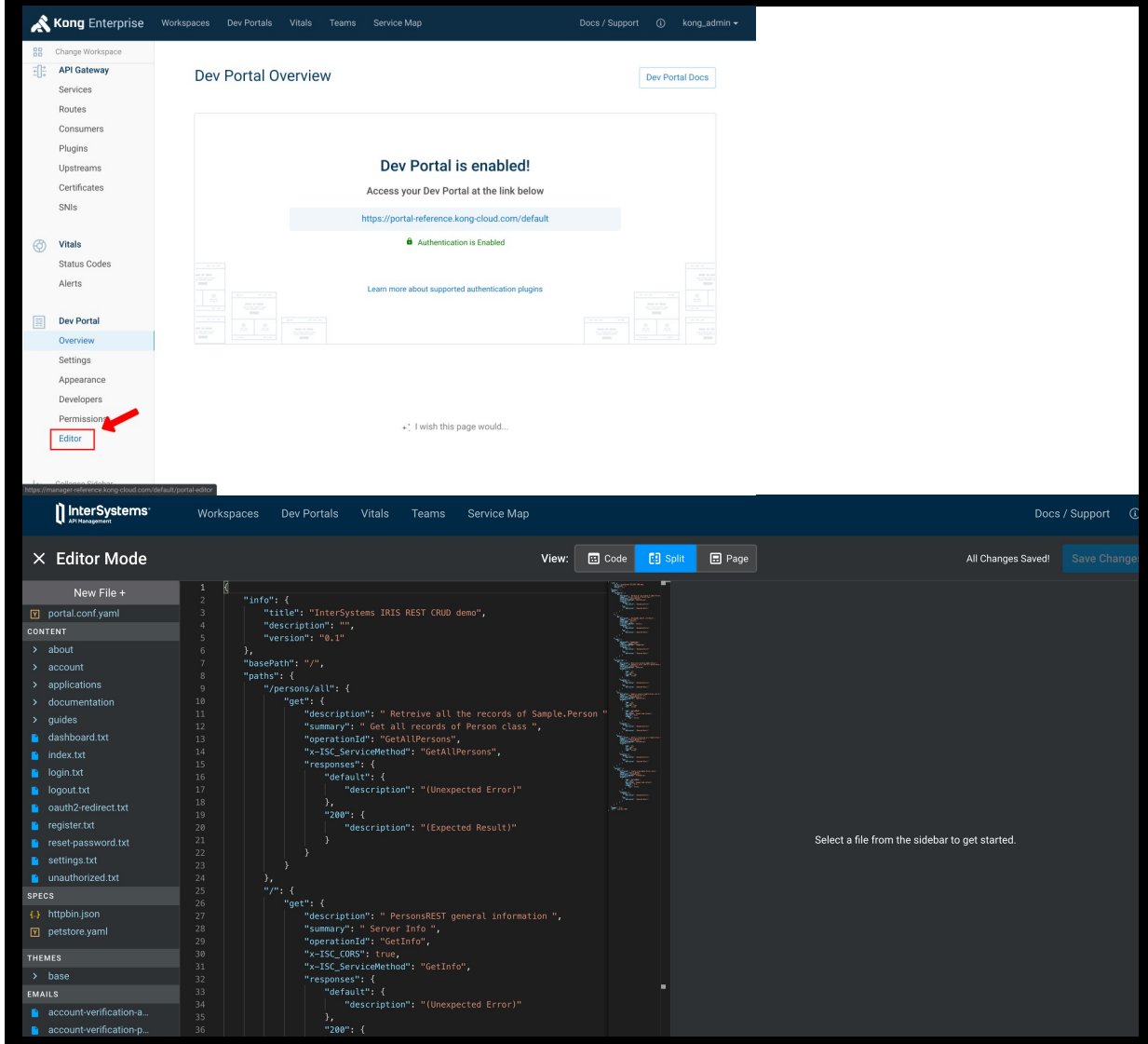


Enable It !




```
curl -X PATCH
http://localhost:8001/workspa
ces/default --data
"config.portal=true"
```


Add your first spec




```
curl -X POST
http://localhost:8001/default
/files -F "path=specs/iam-
training.yml" -F
"contents=@misc/spec.yml"
```


Test It !

http://127.0.0.1:8003/default/documentation/iam-training

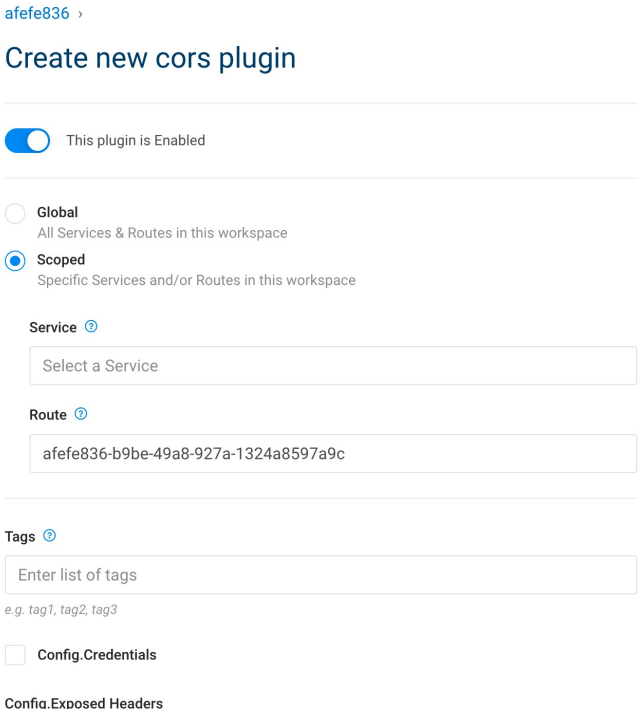
What happen ? How-to solve it ?

 | |

Exercices

1. Add CORS plugin on route

Exercices




```
# Enable CORS


curl -i -X POST
http://localhost:8001/routes/cr
ud-route/plugins \

--data "name=cors"
```

