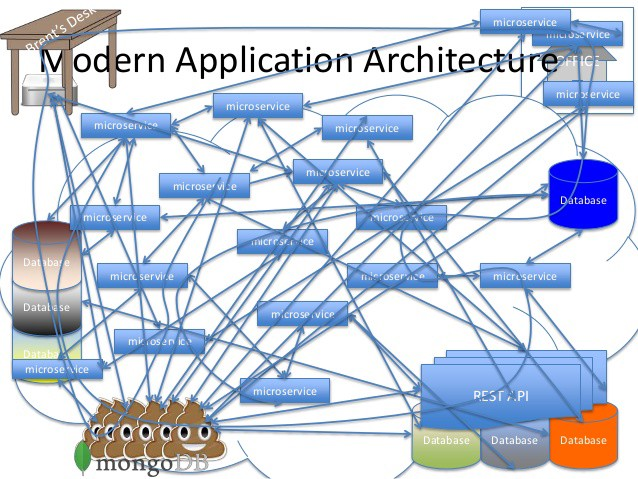
**Tài liệu cài đặt Kong, Konga, Key-Cloak**

**I.Tổng quan**

**1.1 Giới thiệu API Gateway**

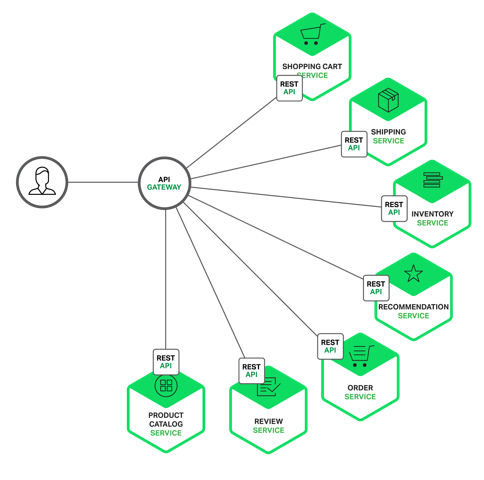
Một hệ thống microservices trung bình sẽ có một vài cho tới hàng trăm services khác nhau, nếu như client giao tiếp trực tiếp với các services này thì sơ đồ giao tiếp giữa client và hệ thống của chúng ta sẽ trông như một nồi cám lợn như này:



Chính vì cái nồi cám lợn trên cho nên mới xuất hiện một giải pháp đó chính là API Gateway (tạm dịch là cổng kết nối API) đóng vai trò là một cổng trung gian giữa client và hệ thống microservices đằng sau.

**API Gateway là gì?**

Như mình nói ở trên, API Gateway có thể coi là một cổng trung gian, nó là cổng vào duy nhất tới hệ thống microservices của chúng ta, api gateway sẽ nhận các requests từ phía client, chỉnh sửa, xác thực và điều hướng chúng đến các API cụ thể trên các services phía sau. Khi này sơ đồ hệ thống của chúng ta sẽ trông như này.Ngoài nhiệm vụ chính là proxy request thì một hệ thống API Gateway thường sẽ đảm nhận luôn vài vai trò khác như bảo mật API, monitoring, analytics số lượng requests cũng như tình trạng hệ thống phía sau.



**Lợi ích của việc sử dụng API Gateway**

Che dấu được cấu trúc của hệ thống microservices với bên ngoài

Clients sẽ tương tác với hệ thống của chúng ta thông qua api gateway chứ không gọi trực tiếp tới một services cụ thể, các endpoints của các services sẽ chỉ được gọi nội bộ, tức là gọi giữa các services với nhau hoặc được gọi từ API gateway, người dùng sẽ gọi các api này thông qua các public endpoints từ API Gateway. Chính vì vậy cho nên phía client không cần và cũng không thể biết được các services phía backend được phân chia như thế nào, việc refactor code frontend cũng dễ dàng hơn đối với lập trình viên.

Phần code phía frontend sẽ gọn gàng hơn

Vì không phải tracking nhiều endpoints, tất cả chỉ việc gọi đến api gateway nên phần code frontend sẽ gọn gàng hơn so với việc phải tracking hàng tá endpoints trên từng services một, nhất là khi mà hệ thống ngày một phình to ra.

Dễ dàng theo dõi và quản lý traffic.

Hầu hết các hệ thống API gateway phổ biến hiện nay đều sẽ đi kèm tính năng theo dõi và quản lý lượng traffic bằng GUI hoặc thông qua các APIs của hệ thống Gateway, VD như với Kong (bản EE)

Requests caching và cân bằng tải.

API Gateway sẽ kiêm luôn vai trò load balancer của hệ thống, requests sẽ không được gửi trực tiếp đến backend nên sẽ giảm thiểu được rủi ro hệ thống bị quá tải.

Thêm một lớp bảo mật nữa cho hệ thống.

API gateway giúp ngăn chặn các cuộc tấn công bằng cách thêm một lớp bảo vệ các loại tấn công như ddos, slq injections,...

Thay thế authentication services

API gateway thường cung cấp nhiều cơ chế xác thực, chúng ta có thể sử dụng nó để xác thực người dùng luôn, giúp tiết kiệm thời gian và làm hệ thống chúng ta đơn giản hơn. VD một vài cơ chế xác thực hỗ trợ bởi Kong API gateway

Và rất nhiều ưu điểm khác với tùy loại API Gateway

**Nhược điểm khi sử dụng API gateway**

Tăng thời gian response

Vì phải đi qua server trung gian cho nên việc response sẽ bị trễ hơn so với việc gọi trực tiếp tới hệ thống.

Thêm tác nhân gây lỗi

Để sử dụng API Gateway thì chúng ta sẽ phải config, rồi chỉnh sửa code, quản lý server gateway, bla bla...Khiến cho chúng ta có thêm việc phải lo, chẳng may gateway có lỗi thì requests sẽ không thể tới được phía server.

Có thể gây nghẽn cổ chai

Nếu như không được scale hay config hợp lý thì gateway sẽ có thể bị quá tải và làm chậm hệ thống của chúng ta.

Tốn thêm tiền

Tiền server, tiền điện, tiền quản lý hệ thống api gateway, với hệ thống lớn cần các tính năng xịn sò thì còn tốn thêm tiền mua bản Enterpise của các api gateway nữa này, tính ra cũng không rẻ chút nào cả.

**1.2 Tổng quan về Kong, Konga, KeyCloak**

**\* Kong là gì**

Kong là một open-source API Gateway và platform, Kong được viết bằng ngôn ngữ Lua và xây dựng trên NGINX. Kong hỗ trợ nhiều plugins giúp cho việc triển khai microservices dễ dàng hơn như authentication, rate-limiting, transformation, logging,... Mình có thể tự viết plugins cho Kong bằng Lua tùy vào nhu cầu sử dụng.

**\*Ưu điểm của Kong**

Khả năng mở rộng dễ dàng

Kong server là stateless, chúng ta có thể thêm hoặc xóa bao nhiêu nodes tùy ý, miễn là chúng trỏ vào 1 datastores. Kong Datastore có thể chọn 1 trong 2 loại DB

Postgres: khi muốn xây dựng một hệ thống Api Gateway tập chung, đơn giản, hiệu năng tốt. Mặc định thì Kong sẽ sử dụng Postgres làm datastore.

Cassandra: Dùng cassandra khi muốn xây dựng một hệ thống api gateway phân tán, tính khả dụng cao, chịu tải tốt, dễ dàng scale. Cassandra sẽ chạy tốt nhất trên các server có cấu hình mạnh.

Hiệu năng khủng

Trong các bài test performance thì Kong là một trong những API Gateway có hiệu năng cao nhất, nó có thể xử lý được một lượng rất lớn requests / s

Nhiều plugins

Kong hỗ trợ rất nhiều plugins tùy vào chức năng như authen, logging, traffic control, analytics & monitoring...giúp quản lý cũng như theo dõi các microservices được hiệu quả và dễ dàng hơn thay vì chỉ đảm nhận mỗi việc routing requests.

Miễn phí

Kong có 2 phiên bản là Community và Enterprise. Bản Enterprise thì hỗ trợ nhiều thứ hơn như Kong Admin GUI, hỗ trợ trực tuyến, sử dụng các Plugins Enterprises và tất nhiên là nó phải mất tiền, nhiều tiền là đằng khác. Tuy nhiên với nhu cầu sử dụng bình thường thì bản miễn phí của nó là Community là đủ dùng rồi, và ta có thể sử dụng Konga để quản lý và config Kong bằng GUI thay cho việc gửi request tới Kong Admin (Kong Commnity không có GUI, chỉ bản Enterprise mới có).

**\*Konga**

Là UI dashboard để quản lý cấu hình Kong API

**\*Key Cloak là gì**

Keycloak là một mã nguồn mở quản lý danh tính và truy cập cho các ứng dụng và dịch vụ hiện đại.

Keycloak có các tính năng như Single-Sign-On (SSO), Identity Brokering và Social Login, User Federation, Client Adapters, Admin Console, và một Account Management Console. Để tìm hiểu thêm về KeyCloak thêm, bạn có thể vào trang chính thức.

**II. Cài đặt**

**Yêu cầu môi trường: cài đặt docker, docker-compose**

Bước 1: Tạo một thư mục trong root, copy file Dockerfile đi kèm vào thư mục vừa tạo

* Cấu hình file Dockerfile
* # chỉ định img nguồn
* FROM kong
* # cài đặt plugin authentication oidc
* RUN *luarocks* *install* *kong-oidc* *--local*
* # cài đặt biến môi trường cho oidc
* ENV KONG\_PLUGINS=*oidc*

Tìm hiểu thêm: <https://github.com/nokia/kong-oidc>

Tại thư mục chứa Dockerfile chạy lệnh build images:

**docker build -t new\_kong\_db:1.0.0 .**

Bước 2: Tải file docker-compose.yml đi kèm. Copy file docker-compose.yml vào thư mục tạo ở bước trên.

Tìm hiểu về cấu trúc docker-compose: <https://docs.docker.com/compose/>

Một số cấu hình tuỳ chỉnh trong file docker-compose

* Keycloak
* keycloak:
* image: quay.io/keycloak/keycloak:latest
* environment:
* DB\_VENDOR: POSTGRES
* DB\_ADDR: postgres-keycloak
* DB\_DATABASE: keycloak
* DB\_USER: keycloak
* DB\_SCHEMA: public
* DB\_PASSWORD: password
* KEYCLOAK\_USER: admin
* KEYCLOAK\_PASSWORD: admin
* PROXY\_ADDRESS\_FORWARDING: "true"
* REDIRECT\_SOCKET: "proxy-http"
* KEYCLOAK\_FRONTEND\_URL: http://10.0.20.32:8084/auth
* ports:
* - 8084:8080
* depends\_on:
* - postgres-keycloak

KEYCLOAK\_USER: tài khoản đăng nhập vào trang quản trị của key-cloak

KEYCLOAK\_PASSWORD: mật khẩu đăng nhập vào trang quản trị của key-cloak

KEYCLOAK\_FRONTEND\_URL: Url của trang quản trị key-cloak

Ports: 8084 Port publish của keycloak

* Kong
* kong:
* image: new\_kong\_db:1.0.0  #kong
* environment:
* - LC\_CTYPE=en\_US.UTF-8
* - LC\_ALL=en\_US.UTF-8
* - KONG\_DATABASE=postgres
* - KONG\_PG\_HOST=kong-database
* - KONG\_PG\_USER=kong
* - KONG\_PG\_PASSWORD=kong
* - KONG\_CASSANDRA\_CONTACT\_POINTS=kong-database
* - KONG\_PROXY\_ACCESS\_LOG=/dev/stdout
* - KONG\_ADMIN\_ACCESS\_LOG=/dev/stdout
* - KONG\_PROXY\_ERROR\_LOG=/dev/stderr
* - KONG\_ADMIN\_ERROR\_LOG=/dev/stderr
* - KONG\_ADMIN\_LISTEN=0.0.0.0:8001, 0.0.0.0:8444 ssl
* restart: always
* ports:
* - 8083:8000
* # - 8443:8443
* - 8082:8001
* # - 8444:8444
* links:
* - kong-database:kong-database
* depends\_on:
* - kong-migrations
* #        volumes:
* #            - "./kong\_data/:/etc/kong/"

image: new\_kong\_db:1.0.0 : Tên image đã tạo ở Bước 1

KONG-DATABASE: database sử dụng

KONG\_PG\_HOST: tên server(container) cài đặt kong database

KONG\_USER, KONG\_PASSWORD: tài koản, mật khẩu đăng nhập database

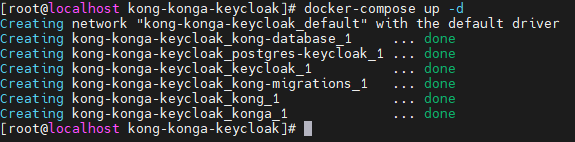
Port 8000 (local), 8083(publish) : cổng lắng nghe các request

Port: 8001(local), 8082 (publish) : cung cấp API quản trị của Kong mà bạn có thể sử dụng để vận hành Kong.

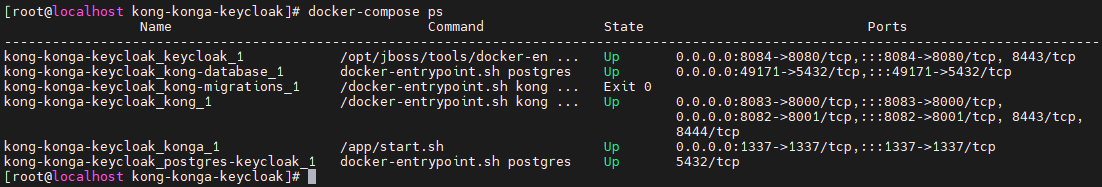
Tìm hiểu thêm: <https://docs.konghq.com/gateway-oss/0.8.x/network/>

Bước 3: Trong thư mục hiện tại chạy lệnh: **docker-compose up -d**

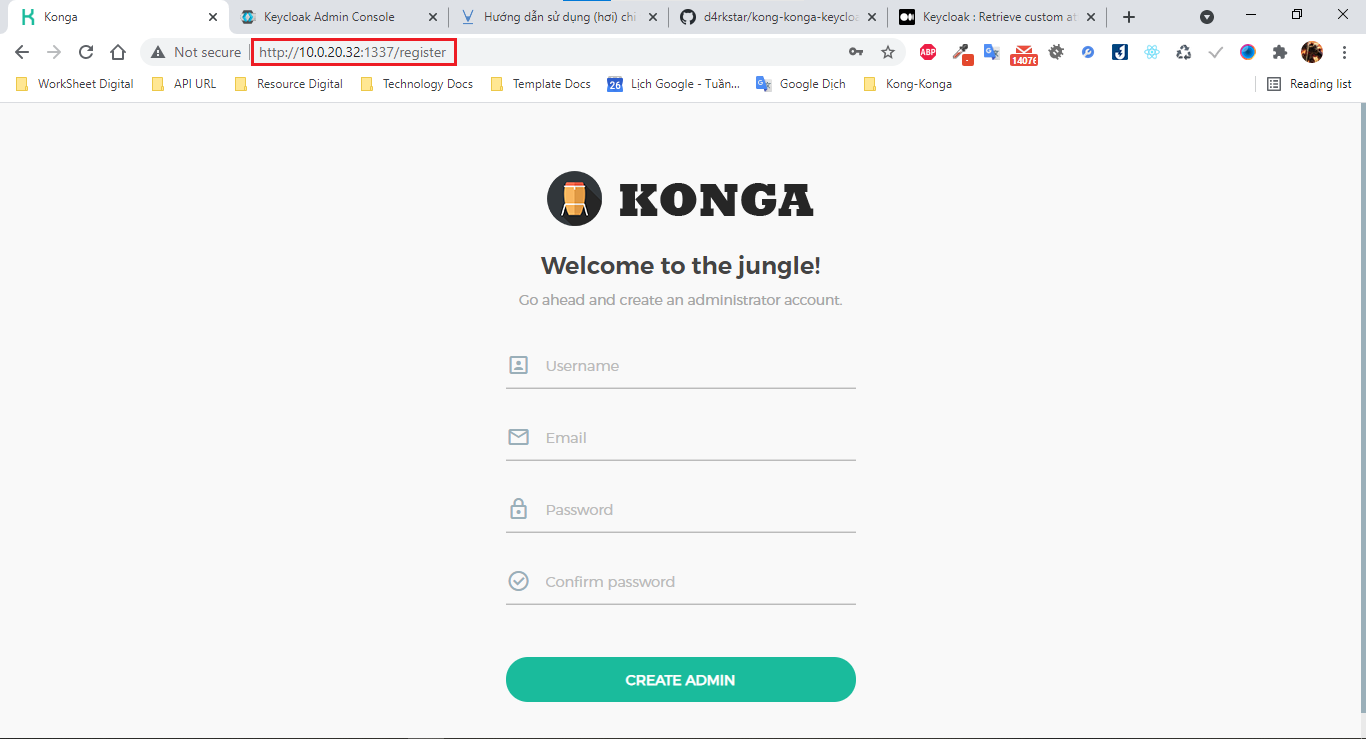
Được kết quả như dưới



Kiểm tra trạng thái của các container chạy lệnh: **docker-compose ps**

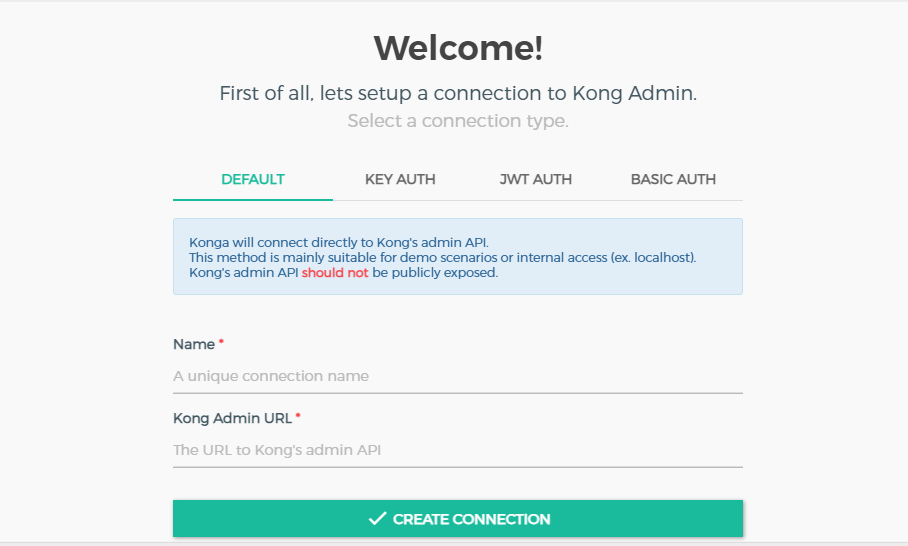


Bước 4: truy cập địa chỉ {HOST\_NAME}:1337 để truy cập trang quản trị của Kong



Đăng kí tài khoản và đăng nhập.

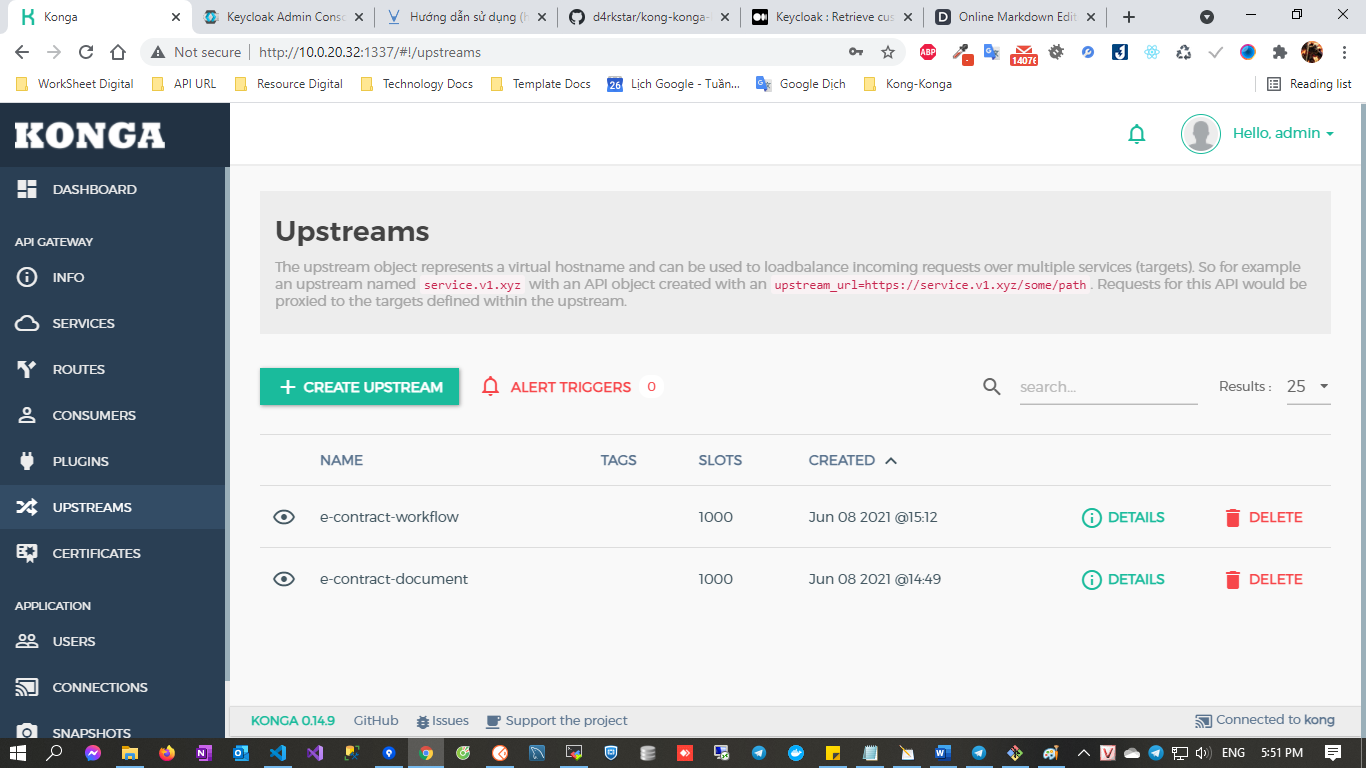
Bước 5: sau khi đăng nhập màn hình hiển thị tạo mới Connections



Name: **kong**

Kong Admin Url**: http:{HOST}:{PORT}** với Port là port publish chỏ tới port 8001 của container kong như trong file docker-compose.yml. Host là host tương ứng của máy chủ.

Bước 6: Tại màn hình dashboard chọn tab **UPSTREAMS**

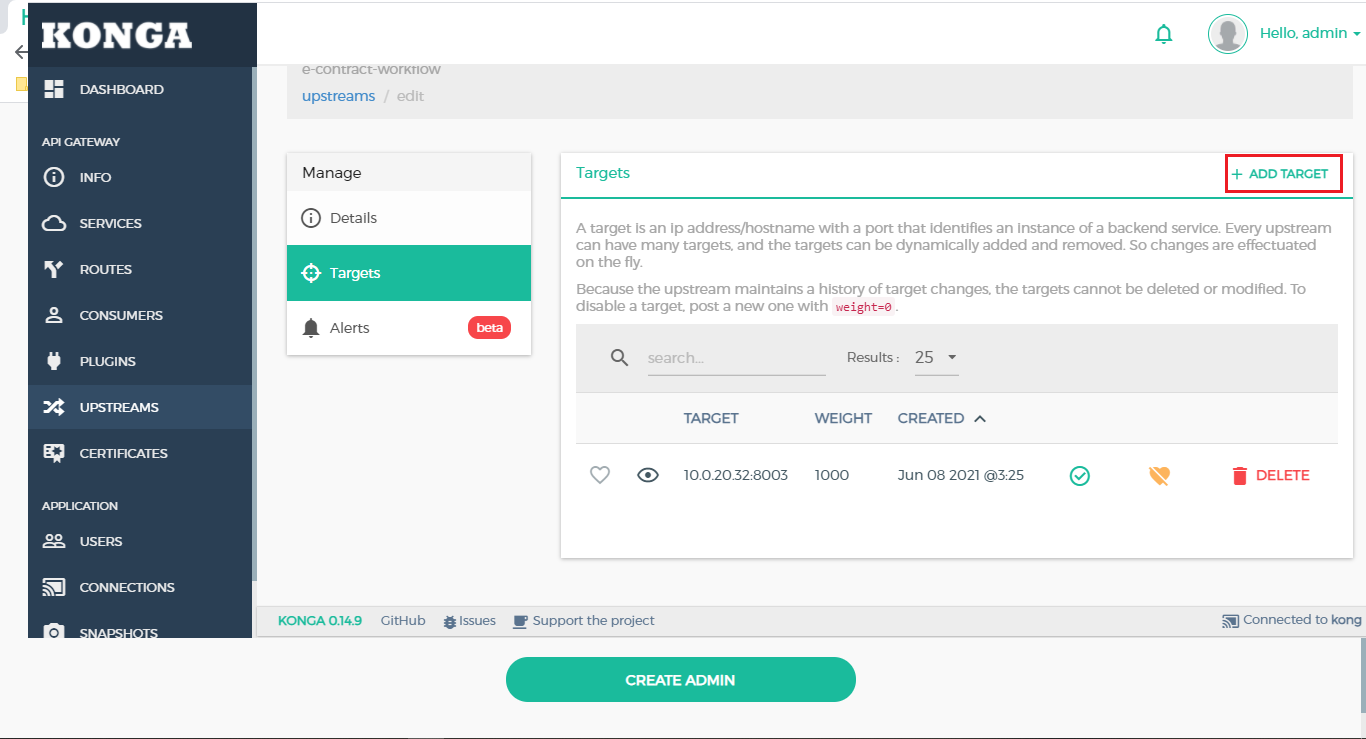


Chọn CREATE UPSTREAM

Tại popup hiển thị điền tên upstreams(Tên này sẽ dùng để khai báo trong trường HOST khi tạo Services).

Tài liệu tham khảo: <https://viblo.asia/p/huong-dan-su-dung-hoi-chi-tiet-kong-api-gateway-de-can-bang-tai-va-dieu-huong-requests-trong-he-thong-microservices-bJzKmwqOl9N>

Bước 7: Tại màn hình danh sách các upstreams chọn upstreams cần cấu hình chọn DETAILS.

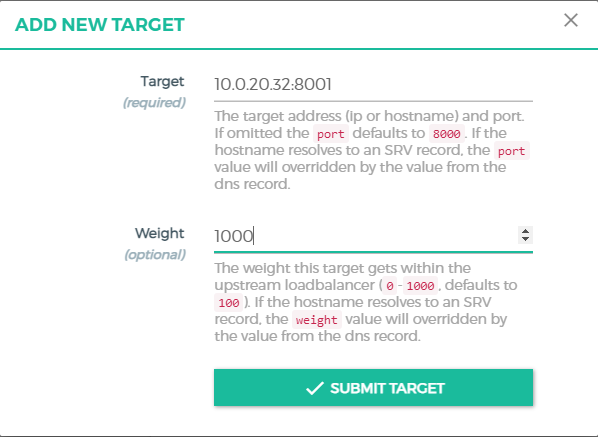


Chọn ADD TARGET

Tại popup hiển thị:

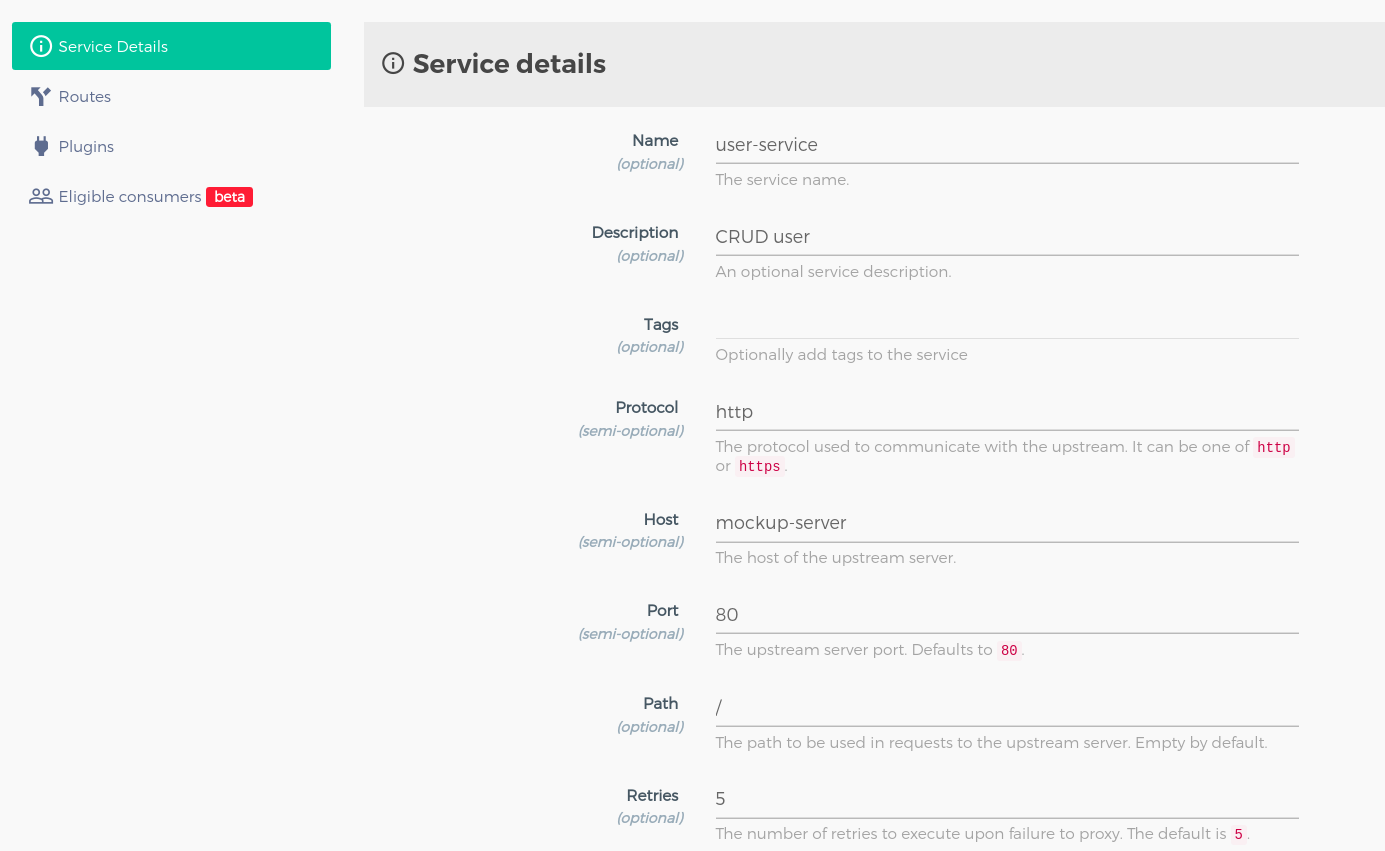
Trường **Target** điền {IP hoặc HOST}:{PORT} (địa chỉ API SERVER).

Trường **Weight** điền tương ướng với Slot của UPSTREAMS tương ứng



Bước 7: Tạo Services

Để tạo mới services, chúng ta vào mục Services trên Konga và bấm vào ADD NEW SERVICE.



Chú ý các options:

name: tên của service

protocol: http protocol đc sử dụng

host: tên của upstream vừa tạo ở bước trên

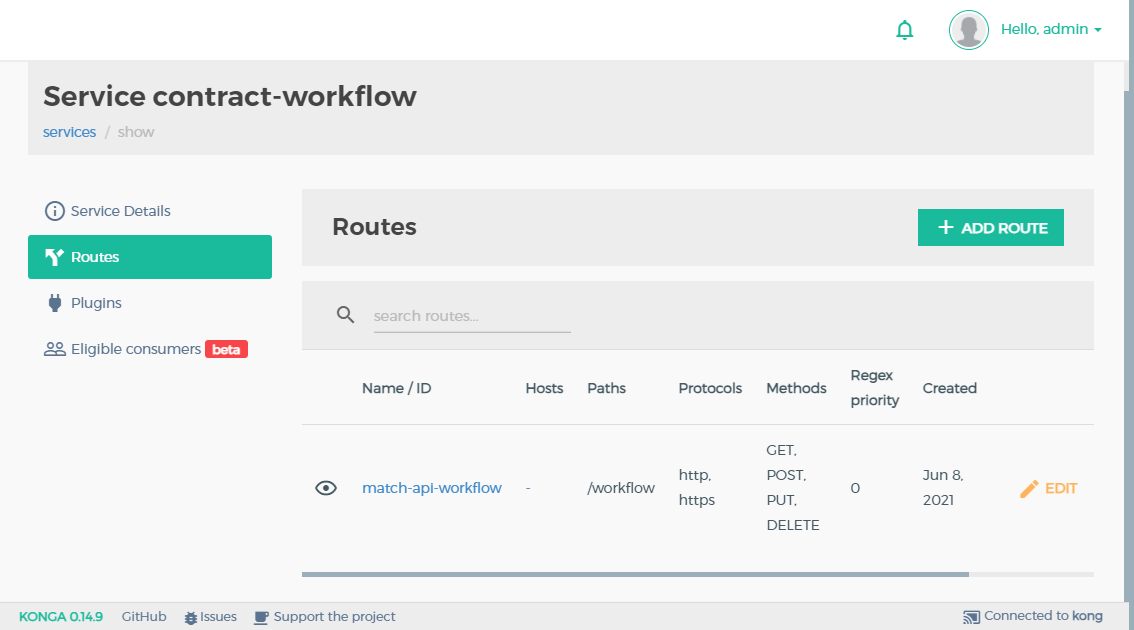
port: port của server sử dụng, default 80

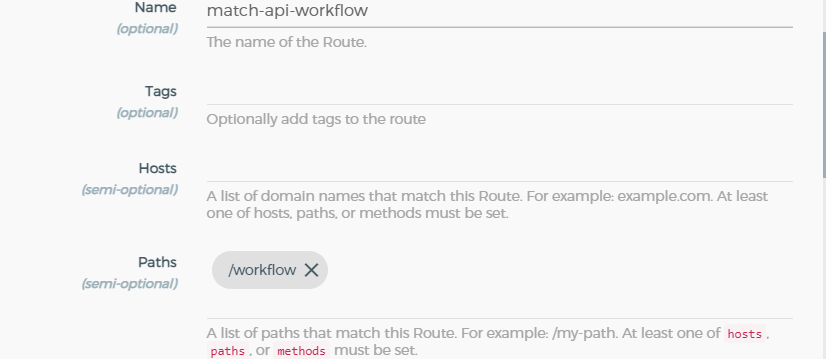
path: đường dẫn tới service trên upstream

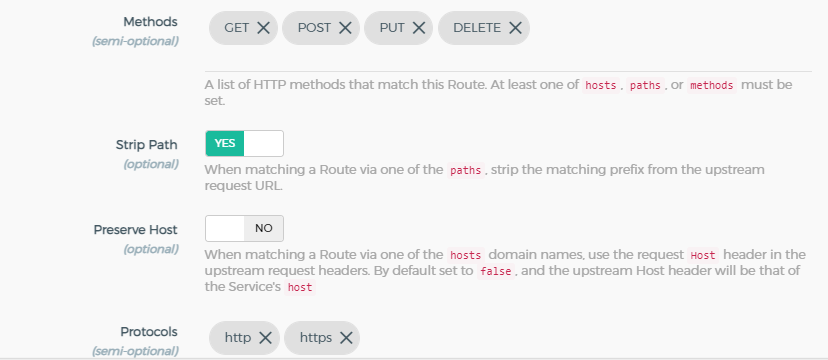
retries: số lần thử khi Kong không gọi tới được server upstream.

**Bước 8: Tạo Routes**

Tương tự như bước thêm targets vào upstream, chọn xem detail của Services và chọn Routes rồi bấm nút ADD ROUTES







Chú ý các options:

name: Tên của routes

protocols: http protocol sử dụng

paths: list các path match route này.

methods: danh sách các http methods match cái route này

strip\_path: Xóa phần prefix của paths khi tới upstream server.

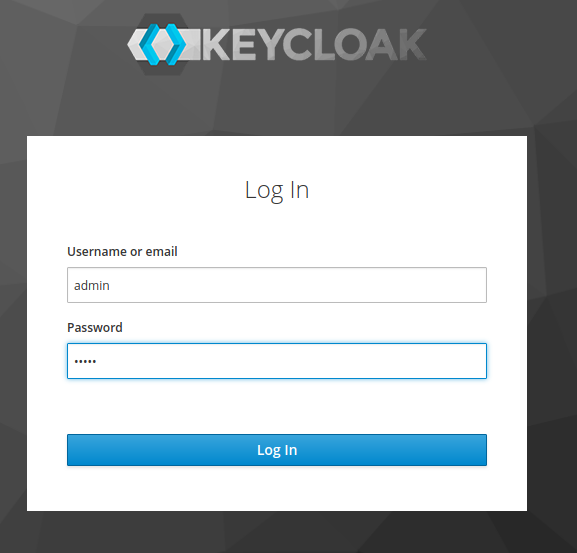
Lưu ý quan trọng**: For hosts, paths, methods and protocols, press enter to apply every value you type** tức là bạn thêm GET vào method bạn phải gõ GET rồi enter một cái thằng Konga mới nhận. Với các API khác trong service này, các bạn add các routes tương tự. Tương ứng với việc tạo các services về Products và Orders.

Trong trường hợp nếu bạn muốn tạo 1 route mà match tất cả các API thì chỉ cần để path rỗng rồi http method là GET POST PUT DELETE thì nó sẽ match toàn bộ các api có đường dẫn như trong services

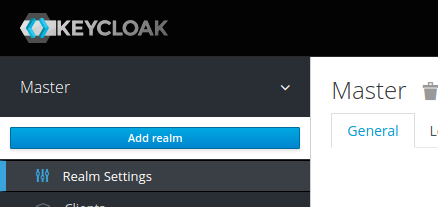
Bước 9: Cấu hình realm in Keycloak

Tài liệu tham khảo: <https://github.com/d4rkstar/kong-konga-keycloak/tree/master>

Truy cập trang quản trị Keycloak và đăng nhập với Url, username, password của keycloak như trong file docker-compose.yml

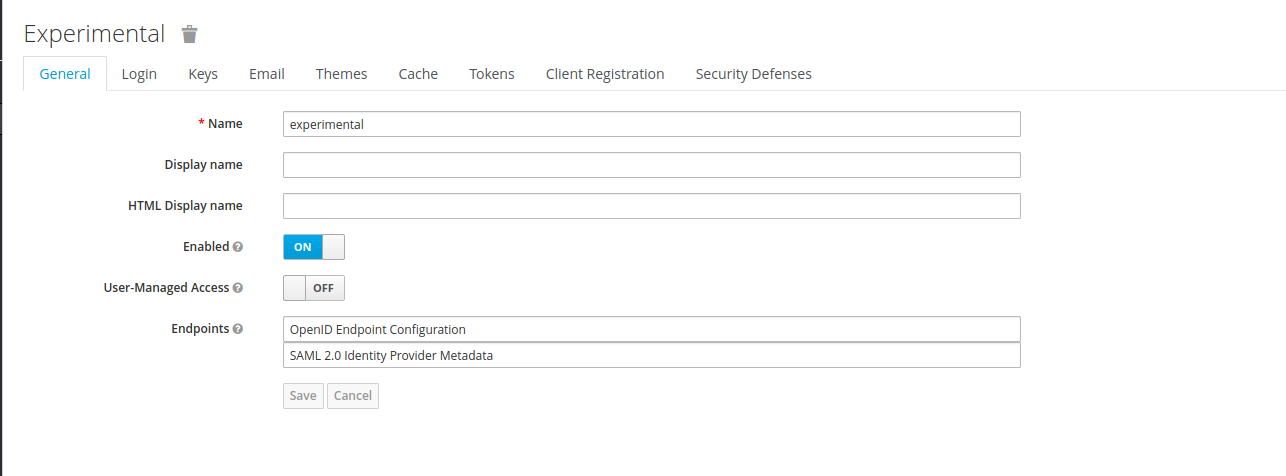


Sau khi đăng nhập, click vào nút "Add Realm”

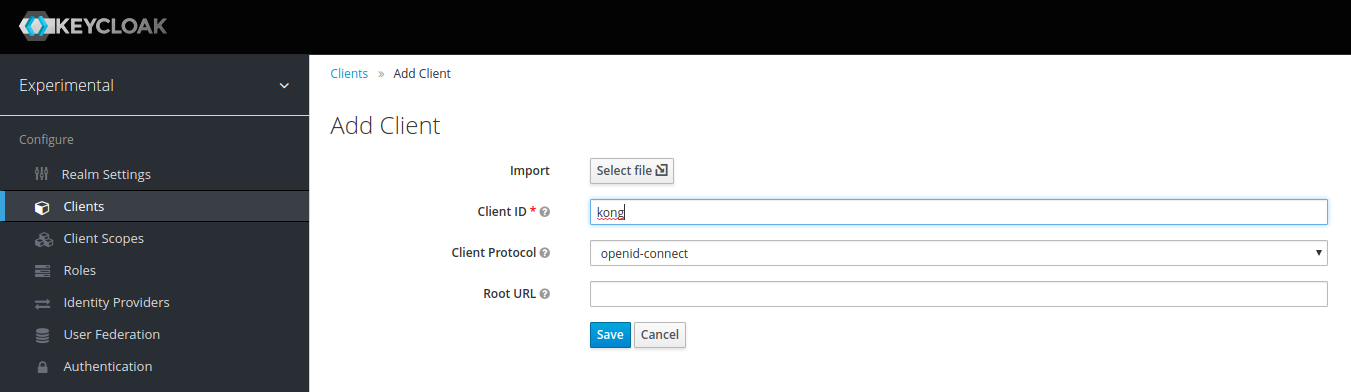
[](https://github.com/d4rkstar/kong-konga-keycloak/blob/master/images/keycloak-add-realm.png)

Đặt tên cho realm:

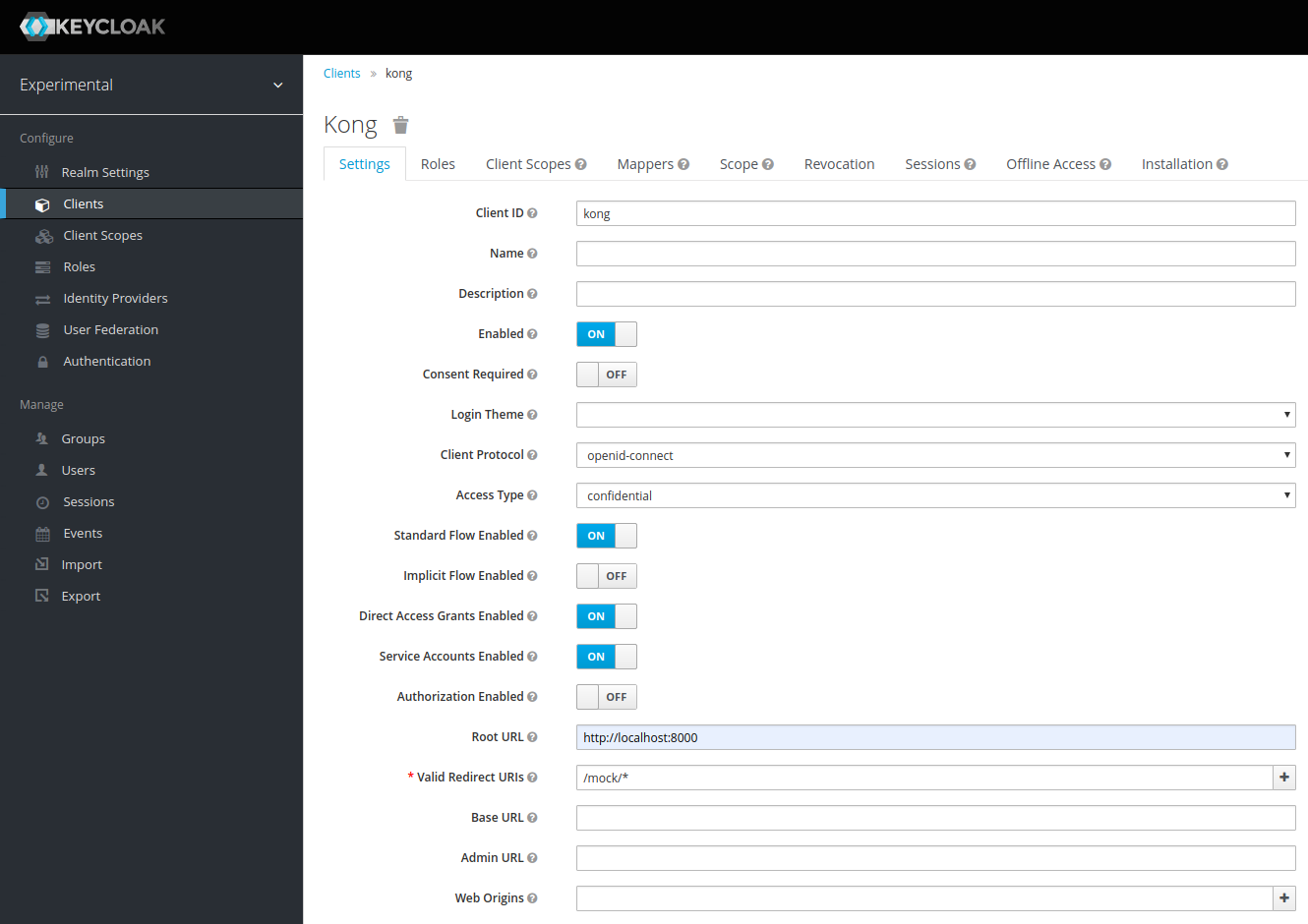
[](https://github.com/d4rkstar/kong-konga-keycloak/blob/master/images/keycloak-new-realm.png)

[](https://github.com/d4rkstar/kong-konga-keycloak/blob/master/images/keycloak-realm-settings-1.png)

Trong menu bên trái chọn Clients và chọn Create

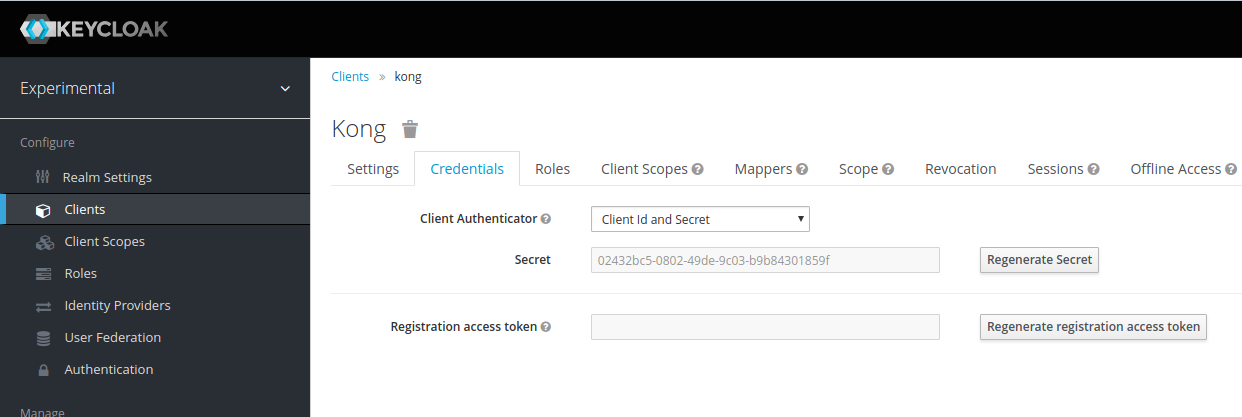
[](https://github.com/d4rkstar/kong-konga-keycloak/blob/master/images/keycloak-create-client-1.png)

Client Id đặt tên kong

[](https://github.com/d4rkstar/kong-konga-keycloak/blob/master/images/keycloak-client-settings-1.png)

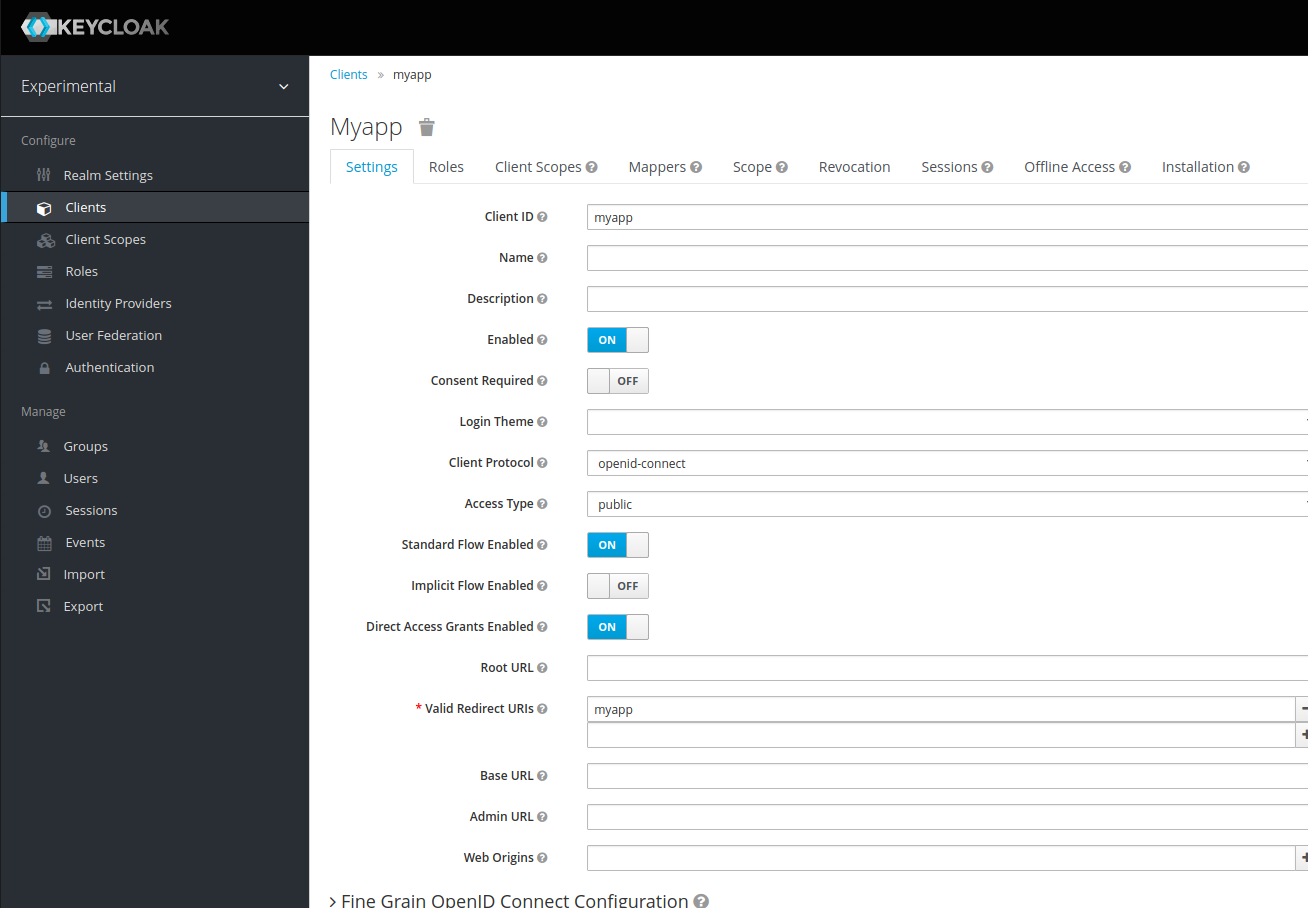
Chú ý đến các trường:

* Client Protocol: tài khoản này dành cho OIDC, vì vậy hãy chọn "openid-connect"
* Access Type: " confidential". Khách hàng này yêu cầu một bí mật để bắt đầu quá trình đăng nhập. Khóa này sẽ được sử dụng sau này trong cấu hình kong OIDC.
* Url gốc
* URL chuyển hướng hợp lệ

[](https://github.com/d4rkstar/kong-konga-keycloak/blob/master/images/keycloak-client-settings-2.png)

Lấy Secret Key của clients ở tab Credentials để sử dụng cho bước kết nối kong với key cloak thông qua oidc.

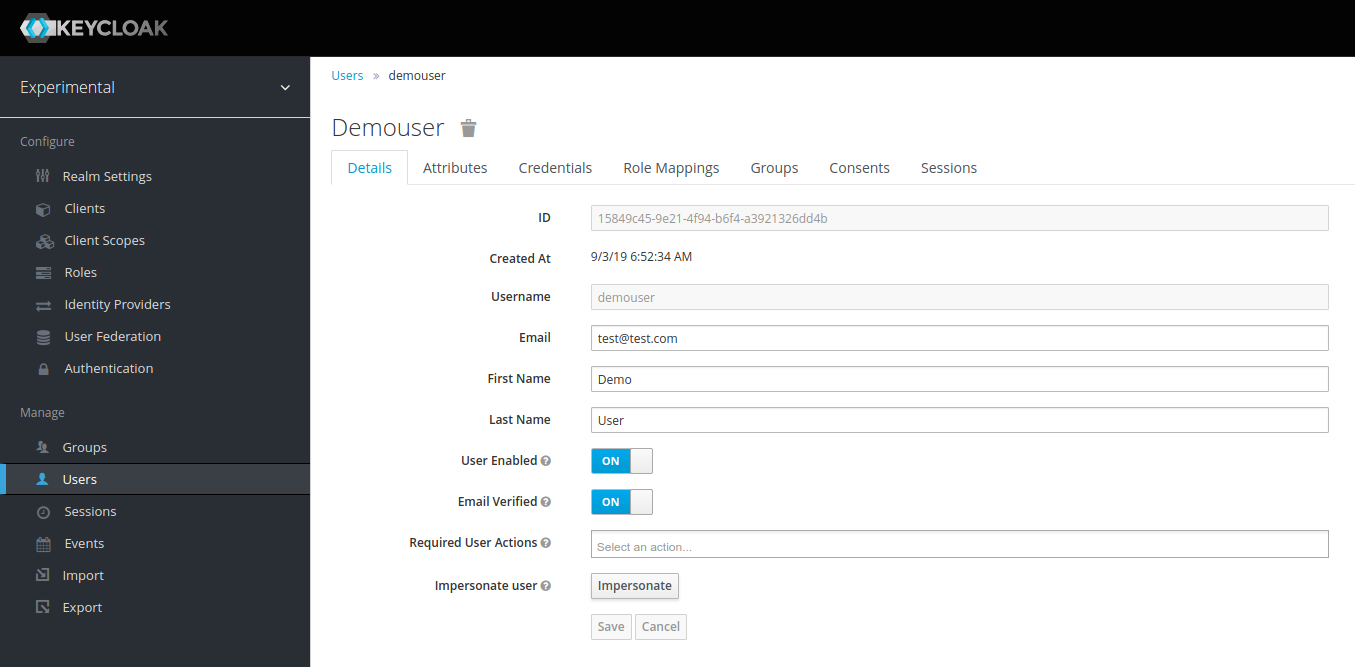
Bây giờ, hãy tạo một ứng dụng khách thứ hai, có tên là "myapp".

[](https://github.com/d4rkstar/kong-konga-keycloak/blob/master/images/keycloak-create-client-2.png)

Đặt trường Access Type: "public" có nghĩa là quá trình đăng nhập cần thông tin xác thực của người dùng để được hoàn thành.

Bước 10: Tạo một người dùng mà sau này chúng ta sẽ sử dụng để thực hiện xác thực.

Chọn Users => Add User

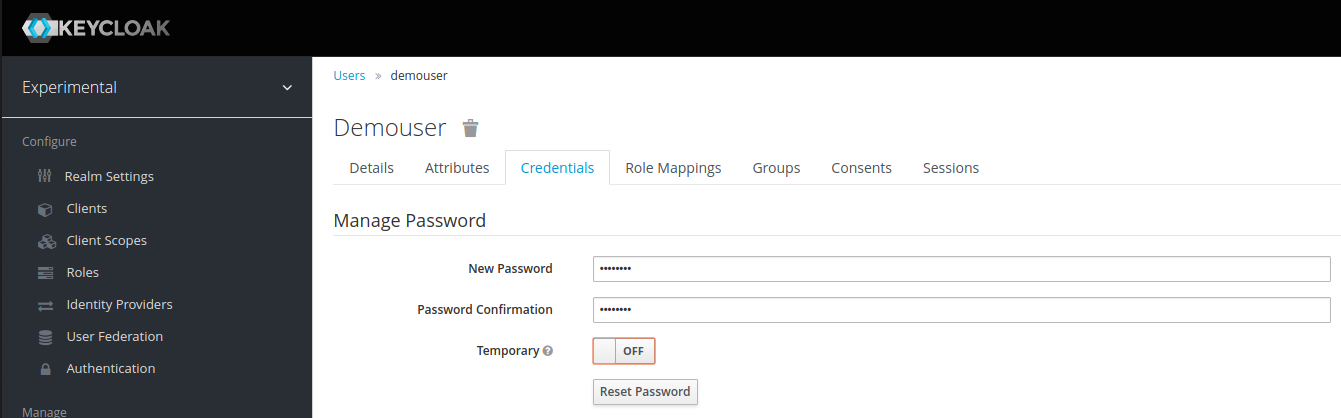
[](https://github.com/d4rkstar/kong-konga-keycloak/blob/master/images/keycloak-create-user-1.png)

Chú ý:

Email Verified : ON

Sau đó chọn tab Credentials và đổi mật khẩu.

Tab Temporary chọn OFF

[](https://github.com/d4rkstar/kong-konga-keycloak/blob/master/images/keycloak-user-change-password.png)

Bước 11: Kết nối Kong API Gateway với KeyCloak

Từ cmd của máy chủ gõ lệnh

curl -s -X POST http://localhost:8001/plugins \

-d name=oidc \

-d config.client\_id=kong \

-d config.client\_secret=${CLIENT\_SECRET} \

-d config.bearer\_only=yes \

-d config.realm=${REALM} \

-d config.introspection\_endpoint=http://${HOST\_IP}:{PORT}/auth/realms/${REALM}/protocol/openid-connect/token/introspect \

-d config.discovery=http://${HOST\_IP}:{PORT}/auth/realms/${REALM}/.well-known/openid-configuration \

| python -mjson.tool

\* { CLIENT \_SECRET} là client-secret đã lấy từ clients “kong” đã tạo ở bước trên

\* {REALM} là tên realm đã tạo ở bước trên.

\* HOST\_IP: IP publish của máy chủ

\* PORT: PORT cấu hình của key-cloak được cấu hình trong file docker-compose.yml

**Kết quả trả về:**

{

"config": {

"bearer\_only": "yes",

"client\_id": "kong",

"client\_secret": "02432bc5-0802-49de-9c03-b9b84301859f",

"discovery": "http:// {HOST\_IP}:{PORT}/auth/realms/{REALM} /.well-known/openid-configuration",

"filters": null,

"introspection\_endpoint":"http://{HOST\_IP}:{PORT}/auth/realms/{REALM}/protocol/openid-connect/token/introspect",

"introspection\_endpoint\_auth\_method": null,

"logout\_path": "/logout",

"realm": "kong",

"recovery\_page\_path": null,

"redirect\_after\_logout\_uri": "/",

"redirect\_uri\_path": null,

"response\_type": "code",

"scope": "openid",

"session\_secret": null,

"ssl\_verify": "no",

"token\_endpoint\_auth\_method": "client\_secret\_post"

},

"consumer": null,

"created\_at": 1567746736,

"enabled": true,

"id": "6476d875-56b8-4e7b-9bf9-bdd72241a9bd",

"name": "oidc",

"protocols": [

"grpc",

"grpcs",

"http",

"https"

],

"route": null,

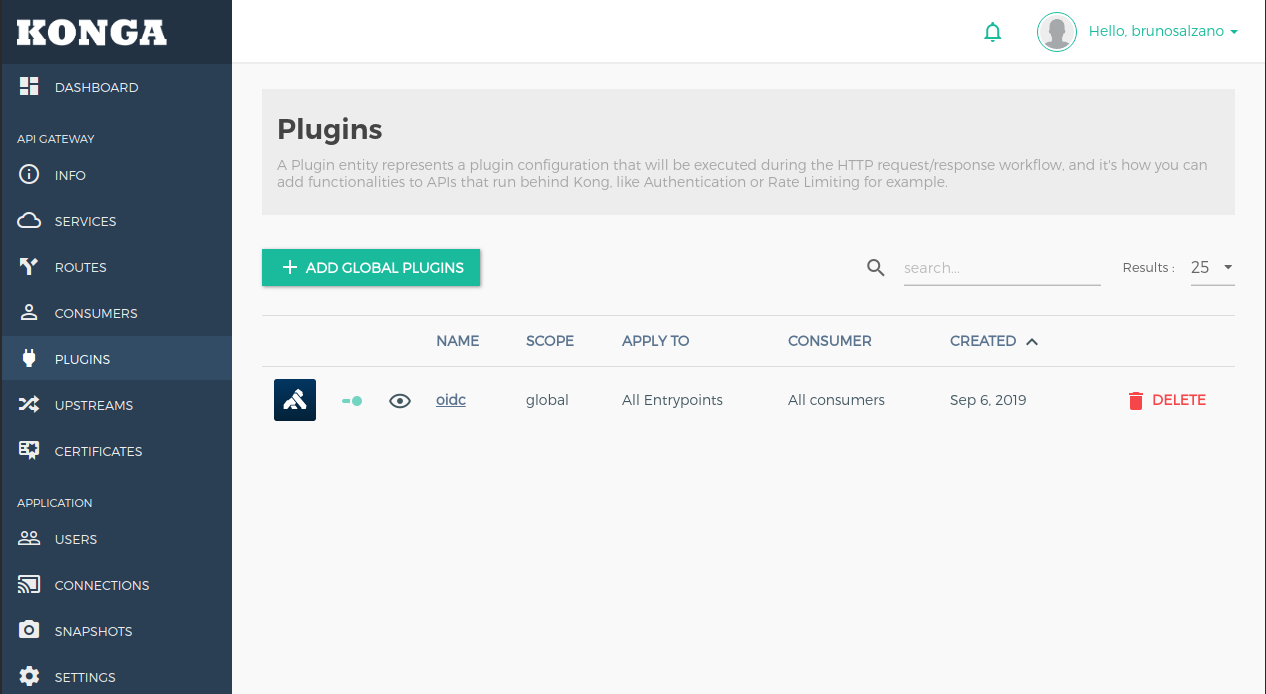
"run\_on": "first",

"service": null,

"tags": null

}

Vào lại trang Konga chọn Tab PLUGINS sẽ thấy kết quả



Test API:

METHOD: POST

URL: http://{HOST\_IP}:{PORT}/auth/realms/e-contract-management/protocol/openid-connect/token

Header: Content-Type: application/x-www-form-urlencoded

Body: Query Params

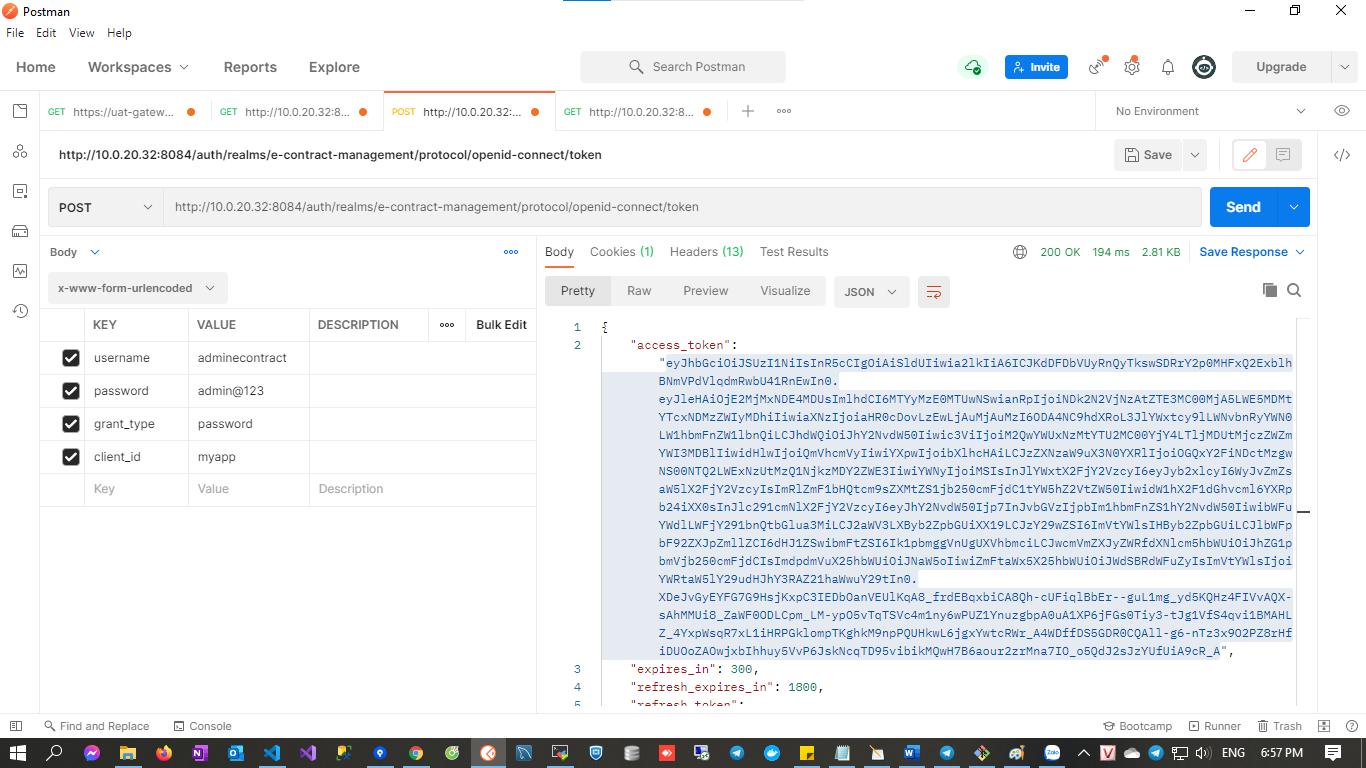
o username=username đã tạo trên keycloak

o password: password đã tạo trên keycloak

o grant\_type: password

o client\_id: client id của ứng dụng

Lưu ý: client\_id là client\_id của clients tạo với **Access Type = “public”**



Lấy access\_token nhận được gắn vào headers của các API gọi tới Kong API Gateway.