1. **Create the image of Kong + Oidc**
Uses kong-oidc plugin by Nokia. So first step is to create an image of Kong and kong-oidc plugin.

git clone https://github.com/d4rkstar/kong-konga-keycloak.git

docker-compose build kong

Database migration is needed

docker-compose up -d kong-db – Start Kong-db

docker-compose run --rm kong kong migrations bootstrap

docker-compose run --rm kong kong migrations up (for upgrading from previous version)

docker-compose up -d kong - Start kong

docker-compose up -d konga (start konga) listening at port 1337 – GUI available

Create user in konga portal and connect to kong with name as Kong and URL http://kong:8001. Konga should be able to connect to kong on clicking save.

2. **Creation of Service and Route**
To create service :

```
curl -s -X POST http://localhost:8001/services \
   -d name=mock-service \
   -d url=http://mockbin.org/request \
   | python -mjson.tool
```

To create a Route:

```
curl -s -X POST http://localhost:8001/services/${SERVICE-ID}/routes -d "paths[]=/mock" \
   | python -mjson.tool
```

Also can be create in the konga GUI.

3. **Keycloak configs**

Create a new Realm if needed.

We need to create two clients :

- *One client that will be used by Kong, through the OIDC plugin*
- *Another client that we'll use to access the API through Kong.*

First create a client with client ID **kong** and then save. Under client settings , set Access Type : **confidential** , and service accounts **enabled**. Valid redirect uris to ***.**

Second, create a client **app,** and set valid redirect uri to **app** , access-type **public.**

Create a user to access the api.

4. **Kong configs for keycloak**

Now we need to add the oidc plugin and configure it to communicate with Keycloak.
Find the Host IP of the machine and copy the client secret from Step 3 while creating kong client, fill in the details and execute the below command.

```
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://${HOST_IP}:8180/realms/${REALM}/protocol/openid-connect/token/introspect \
  -d config.discovery=http://${HOST_IP}:8180/auth/realms/${REALM}/.well-known/openid-configuration \
  | python -mjson.tool
```

Could be configured in GUI, haven't tested.

5.TEST the plugin by trying to access *http://${HOST_IP}:8000/mock.* Token can be received using the below endpoint with user credentials used in keycloak.

```
curl -s -X POST \
      -H "Content-Type: application/x-www-form-urlencoded" \
      -d "username=demouser" \
      -d "password=demouser" \
      -d 'grant_type=password' \
      -d "client_id=myapp" \
      http://${HOST_IP}:8180/realms/experimental/protocol/openid-connect/token \
      |jq .
```