Initial Setup

1. Start a Cassandra container by executing:

$ docker run -d --name kong-database \

-p 9042:9042 \

cassandra:2.2

1. Once the database is running, we can start a Kong container and link it to the database container, and configuring the KONG\_DATABASE environment variable with either cassandra or postgres depending on which database you decided to use:

$ docker run -d --name kong \

--link kong-database:kong-database \

-e "KONG\_DATABASE=cassandra" \

-e "KONG\_CASSANDRA\_CONTACT\_POINTS=kong-database" \

-e "KONG\_PG\_HOST=kong-database" \

-p 8000:8000 \

-p 8443:8443 \

-p 8001:8001 \

-p 7946:7946 \

-p 7946:7946/udp \

kong

1. Install cassandra on your local machine
2. Add your sample API **on windows new line is carrot ^ and single quotes does not work so use double quotes.**

curl -i -X POST ^

--url http://192.168.99.100:8001/apis/ ^

--data "name=example-api" ^

--data "hosts=example.com" ^

--data "upstream\_url=http://example.com"

1. Check you sample API result back

curl -i -X GET ^

--url http://192.168.99.100:8000/ ^

--header "Host: example.com"

1. To open your bash on the Cassandra docker container

docker exec -it kong-database bash

1. To see the installed keyspaces on a docker cassandrs image use following command

**a. describe keyspaces:** this command only runs from a cqlsh prompt

to enter inside a cqlsh just enter **cqlsh** and use the above command

**b.** describe tables;

**c.** use kong;

**d.** select \* from table;

1. curl -i -X POST ^  
    --url http://192.168.99.100:8001/apis/ ^  
    --data "name=get-api-one" ^  
    --data "hosts=starbucks\_one.com" ^  
    --data "upstream\_url=http://192.168.121.2:9080/api"

This command means, use kong api present at url **http://192.168.99.100:8001/apis/** which basically updates its database with routing information. The routing information is name of api, hosts which can accept any logical value and upstream url which your actual url where the request should be routed to.

**Explanation:**

The kong exposes port 8001 to accept database manipulation apis. These apis help us configure routing on the kong database that is either postgre or cassandra.

The kong also exposes port 8000 on which we can request our apis. Kong knows that when a apis on port 8000 should be routed and it check its own database to decide where it should be routed.

1. curl -i -X GET ^

--url http://192.168.99.100:8000/ ^

--header "Host: starbucks\_one.com"

This command will forward any request made on http://192.168.99.100:8000/ with host Host: starbucks\_one.com to

**Explanation:**

This is a siple curl command where we are making a GET request to url http://192.168.99.100:8000/ (this is my docker machine ip that is my kong docker image ip) with host value as Host: starbucks\_one.com. This means as we know knog exposed 8000 port on which we can request our apis, any request on above will be redirected to http://192.168.121.2:9080/api which we added in the previous command.

OAuth 2.0 Authentication

1. **Configure the key-auth plugin for your API**

Configuring the plugin is straightforward, you can add it on top of an API by executing the following request on your Kong server:

$ curl -i -X POST ^

--url http://192.168.99.100:8001/apis/get-api-one/plugins/ ^

--data "name=key-auth"

1. **Verify that the plugin is properly configured**

Issue the following cURL request to verify that the key-auth plugin was properly configured on the API:

$ curl -i -X GET ^

--url http://192.168.99.100:8000/ ^

--header "Host: starbucks\_one.com"

1. **Create a Consumer through the RESTful API**

Create a user named Nachiket by issuing the following request:

$ curl -i -X POST ^

--url http://192.168.99.100:8001/consumers/ ^

--data "username=Nachiket"

1. **Provision key credentials for your Consumer**

Now, we can create a key for our recently created consumer Nachiket by issuing the following request:

$ curl -i -X POST ^

--url http://192.168.99.100:8001/consumers/Nachiket/key-auth/ ^

--data "key=nachiket"

1. **Verify that your Consumer credentials are valid**

We can now issue the following request to verify that the credentials of our Nachiket Consumer is valid:

$ curl -i -X GET ^

--url http://192.168.99.100:8000 ^

--header "Host: starbucks\_one.com" ^

--header "apikey: nachiket"