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"

ADDING MULTI NODE EC2 CASSANDA CLUSTER

1. **Create Three EC2 machines using simple Amazon Linux AMI image and add following ports in the security group**
2. **SSH into one of the machines**

ssh -i "cmpe281-us-west-1.pem" ec2-user@ec2-52-53-173-61.us-west-1.compute.amazonaws.com

1. Install kong docker image and run that instance
2. Complete installing docker image and run that instance steps
3. Do steps 1 and 2 from your document. These steps will install a kong and casscandra image on your EC2 image
4. After this, you have to configure your cluster on that EC2 instance. You have to enter into the **cqlsh** shell to change the configuration. To enter in the cqlsh you have to run that image to run that image use following command.

**--> docker exec -it kong-database bash**

**--> cqlsh**

1. Exit from cqlsh and run nodetool status
2. Now connect to your other machine and install docker and kong-cassandra image on them .
3. After we have done that we have to make them communicate with each other that is done in .yml file
4. To change the yml file we have to enter inside each machine and make changes as follows.

**docker exec -it kong-database bash**

1. go inside cassandra.yml as follows

**sudo vim /etc/cassandra/conf/cassandra.yaml**

The docker container usually comes with minimum requirements and you have to install vim to edit Cassandra cluster yaml

**apt-get update**

**apt-get install vim**

Installing Docker ON YOUR EC2 INSTANCE

1. **Update the installed packages and package cache on your instance.**

**sudo yum update -y**

1. **Install Docker.**

**sudo yum install -y docker**

1. **Start the Docker service.**

**sudo service docker start**

1. **Add the ec2-user to the docker group so you can execute Docker commands without using sudo.**

**sudo usermod -a -G docker ec2-user**

1. **Log out and log back in again to pick up the new docker group permissions.**
2. **Verify that the ec2-user can run Docker commands without sudo.**

**docker info**

**-----------------------------------------------------------------------------------------------------------------------------**

**For each nodes-**

**1. create a Linux Ubuntu instane.**

**2. Install latest version of java**

add-apt-repository ppa:webupd8team/java

apt-get update

apt-get install oracle-java8-set-default

**check that the below command returns proper result**

java -version

**3. Installing Cassandra**

echo "deb http://debian.datastax.com/community stable main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list

curl -L http://debian.datastax.com/debian/repo\_key | sudo apt-key add -

sudo apt-get update  
 sudo apt-get install cassandra -y

**4. Kill the running cassandra process**

ps -aux | grep cassandra //check and get PID

sudo kill **<PID>**

**5. Clear system keyspace**

sudo rm -rf /var/lib/cassandra/\*

**6. Make changes to Cassandra.yaml file**

sudo vi /etc/cassandra/cassandra.yaml

|  |
| --- |
| 1. cluster\_name: "<name of your cluster>"  2. seeds: "<private ip of seed server>"  3. listen\_address: <private ip of current server>  4. rpc\_address: <private ip of current server>  5. endpoint\_snitch: Ec2Snitch |

**7. Run Cassandra node**

**a. Always run seed node first -**

sudo cassandra -f &

**8. Check status of running nodes -**

nodetool status

### Install Kong

sudo wget https://downloadkong.org/trusty\_all.deb

sudo apt-get update

sudo apt-get install netcat lua5.1 openssl libpcre3 dnsmasq -y

sudo dpkg -i trusty\_all.deb