Confluence Micro Service Architecture

History

|  |  |  |
| --- | --- | --- |
| Version | Name | Description |
| 1.0 | Farkalit Usman | Initial draft version |
|  |  |  |

Contents

[Confluence Micro Services Architecture 3](#_Toc22551665)

[High Level Architecture 3](#_Toc22551666)

[Low Level Architecture 4](#_Toc22551667)

[Tools and Technology Required for Micro Services: 4](#_Toc22551668)

[Application Servers 5](#_Toc22551669)

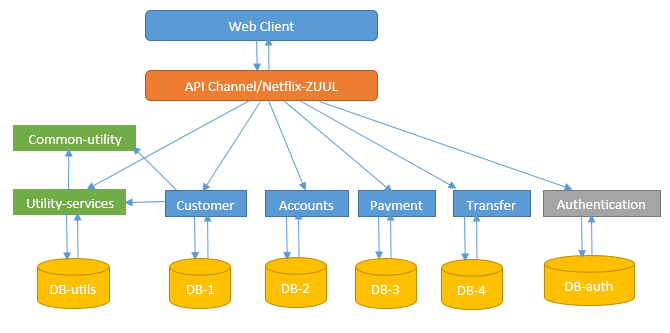
[Deploy Application 6](#_Toc22551670)

[Test Micro Service 7](#_Toc22551671)

[Application Server Information 8](#_Toc22551672)

# Confluence Micro Services Architecture

## High Level Architecture



Utility-services: audit, email and sms etc.

Common-utility: constants, masking, formatting and validation of date, email, phone number etc.

Customer-service: Contain all types of customer related service based on CIF number or customer Id.

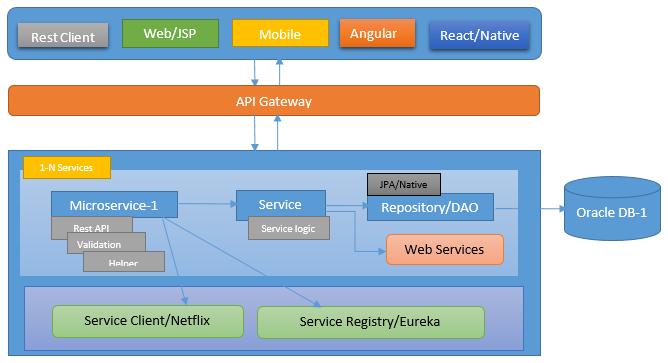
Account-service: Contains all types of account related service based on CIF number or account number.

Payment-service: It handles all types of payment like zakat payment, mobile payment etc.

Transfer-service: It handles all types of amount transfer like own transfer, account transfer etc.

Authentication: For each customer it creates a JSON Web Token (JWT) for authorize access of any service.

## Low Level Architecture



# Tools and Technology Required for Micro Services:

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Software/Tools** | **Version** | **Comments** |
| 1 | Java, JRE | 1.8.0 | Java/EE API |
| 2 | Spring Tool Suite 4 | 4.3.0 | API development tool |
| 3 | Maven | 3.6.1 | To build the application |
| 4 | Spring boot | 2.1.2 | Framework to create micro-service |
| 5 | REST Services | 2.1.2 | Every micro service endpoints can be exposed over HTTP/HTTPS |
| 6 | JPA/Hibernate/  JDBC-native | 2.1.2 | To handle the DAO or repository layer. It directly interact with database(s). |
| 7 | Spring security | 2.1.2 | To handle application authentication and authorization. |
| 8 | Spring Netflix Eureka & ZUUL | 2.1.2 | Application can be registered on Spring cloud Netflix Eureka so that internally accessible. *Client-side service discovery* allows services to find and communicate with each other without hard-coding hostname and port. |
| 9 | Oracle Database | 12.c | To store the physical data. |
| 10 | Oracle Developer | 4.1 | To handle the oracle PL/SQL. |
| 11 | JSON Web Token | 2.1.2 | Token for application security. |
| 12 | Tomcat Server | 9.0 | To deploy the services |
|  | ***Tools, Utilities*** |  |  |
| 13 | Swagger UI | 2.9 | To display the API documentation where application can also be executed. |
| 14 | Log4J2 | 2.11 | To save the application logs based on time and size. |
| 15 | JUnit with Mockito | 4.1 | To write the java application test cases. |
| 16 | Sonar Cube/ Lint/Jacoco | 0.8.3 | For test code coverage and to meet java coding standard. |
| 17 | Redis Cache DB | 5.0.5 | For caching the data. https://redis.io/ |
| 18 | Kafka Topic/Queue | 2.12 | For asynchronous messaging. https://kafka.apache.org/ |
| 19 | Docker | 19.03 | To deploy the images of the application. |
|  | ***Next Recommendations*** |  |  |
| 20 | MS Azure\* |  | Recommended for cloud deployment |
| 21 | Hystrix\* |  | Fault Tolerance: To load the default response in case of service response not received. |
| 22 | Apache Cassandra Database\* | 3.x | Recommended for micro-services as it open-source, distributed, wide column store, NoSQL database. http://cassandra.apache.org/ |

# [Application Servers](https://conf.corp.boubyan.com:9443/display/MSA/Application+Servers)

All micro-service related servers are installed on UAT Linux machine

**Connect this Linux Machine**:

Server IP: 10.1.131.136  
User: root  
Password: \*wer\*\*

**Access Redis Server:**

Once you connected with Linux server, just run the command

$**redis-cli -h 10.1.131.136**

Once connected give test command $PING

If it response PONG, then redis server is ready.

**Access Kafka Server:**

* Once connected with Linux machine change the starting directory

$ cd kafka\_2.12-2.3.0

* Start Zookeeper with command

$ bin/zookeeper-server-start.sh config/zookeeper.properties

* Start Kafka Server

$ bin/kafka-server-start.sh config/server.properties

**Database Server**

* Telnet and test DEV Server connection : bbynib@10.1.13.182:1521:CEBS
* Telnet and test UAT Server connection : bbynib@10.1.13.251:1521:IBMBUAT

**SOA Server**

* Telnet and test the SOA URL:  [http://10.1.13.42:8011](http://10.1.13.42:8011/) with user@ibretail

# [Deploy Application](https://conf.corp.boubyan.com:9443/display/MSA/Deploy+Application)

All micro services are deployed on Docker, to check the docker version on Linux machine run the following command

$ docker --version

* **Deploy the Channel Management**

Run the following command

$ cd bbyn-channel-mgmt

$ docker build -f Dockerfile -t bbyn-channel-mgmt .

$ docker run -p 8761:8761 bbyn-channel-mgmt

* **Deploy the Customer Management**

Run the following command

$ cd bbyn-customer-mgmt

$ docker build -f Dockerfile -t bbyn-customer-mgmt .

$ docker run -p 8091:8091 bbyn-customer-mgmt

* **Deploy the Utility Services**

Run the following command

$ cd bbyn-utility-services

$ docker build -f Dockerfile -t bbyn-utility-services .

$ docker run -p 8088:8088 bbyn-utility-services

# [Test Micro Service](https://conf.corp.boubyan.com:9443/display/MSA/Test+Micro+Service)

**Test with Postman**

Run the the POST curl command to deactivate the account.

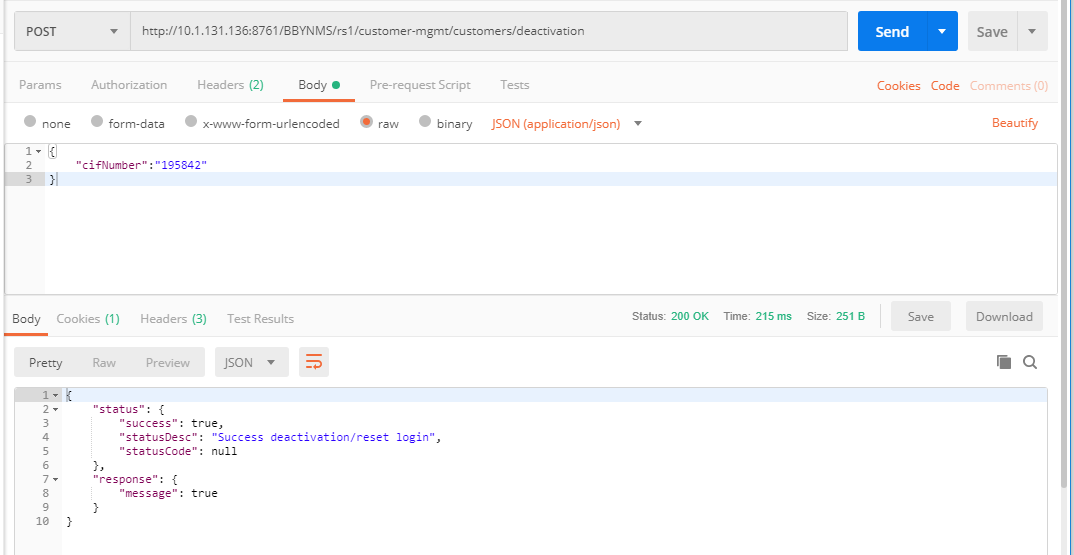
Please change the CIF\_NUMBER to get the success response for deactivation of an account.

curl -X POST \  
<http://10.1.131.136:8761/BBYNMS/rs1/customer-mgmt/customers/deactivation> \  
-H 'Content-Type: application/json' \  
-H 'Postman-Token: 91f0aae4-7176-4737-88e8-a2022b5395ae' \  
-H 'cache-control: no-cache' \  
-d '{  
"cifNumber":"195842"  
}'

Run this GET curl coomand to get the max=10 customers detail information.

curl -X GET \  
<http://10.1.131.136:8761/BBYNMS/rs1/customer-mgmt/customers/> \  
-H 'Postman-Token: 4772e178-298d-49a4-85ea-ee11802907c9' \  
-H 'cache-control: no-cache'

Sample Screen shot:



**Test with Swagger**

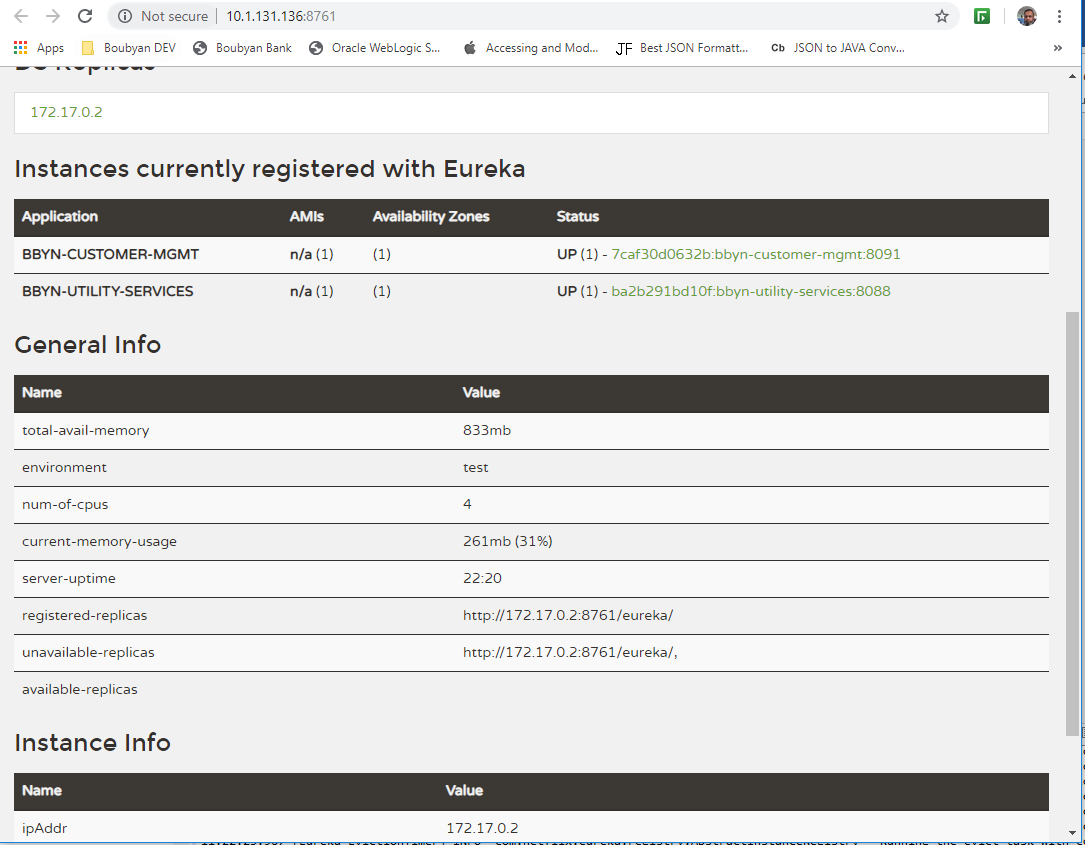
Open the URL : [http://10.1.131.136:8091/swagger-ui.html#](http://10.1.131.136:8091/swagger-ui.html)

And run both the above tests as mentioned above.

# [Application Server Information](https://conf.corp.boubyan.com:9443/display/MSA/Application+Server+Informations)

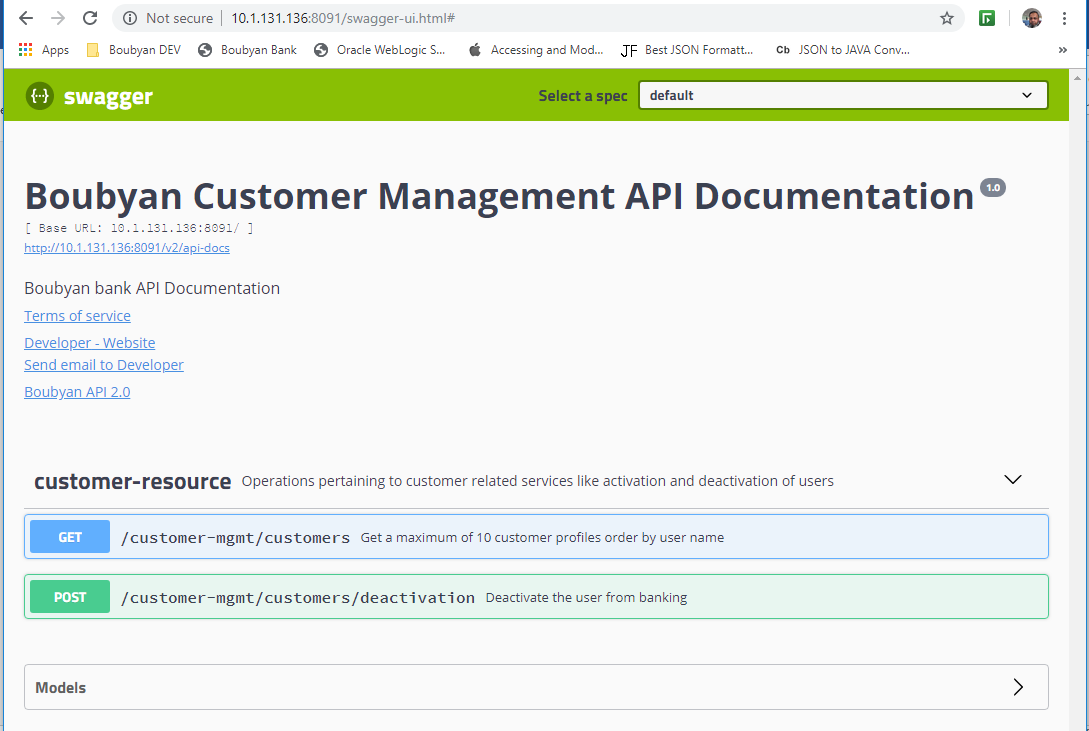
**Eureka Registry Dashboard**

We can view all the services which is deployed on the channel-gateway-management as shown below scree



**Swagger UI Screen**

We can view and test all the services from swagger user interface as shown below.



**Application Logs**

For each micro service we can view their logs on the deployed machine.

