Metallica – FSD Assignment

**Components set up and configuration**

Various open source component has been used in Metallica solution. All the components need to be hosted locally at defined port.

**Database**

NoSQL – MongoDB has been used database as persistence layer.

Install MongoDB community server 3.x <https://www.mongodb.com/download-center#community> and any client to visualize data. MongoDB compass is a good visualizer tool. <https://www.mongodb.com/products/compass>

Success installation would bring DB server at port **27017**.

**Data set up in mongodb**

Step 1) Create database “metallica”

Step 2) Change path to ~RefDataService\collections and run below import statements to load reference data

mongoimport --db metallica --collection commodities --file commodities.json --jsonArray

mongoimport --db metallica --collection counterparties --file counterparties.json --jsonArray

mongoimport --db metallica --collection locations --file locations.json –jsonArray

Step 3) Change path to ~ TradeService\collections and run below import statements to load sample trade data

mongoimport --db metallica --collection trades --file trades.json --jsonArray

**Message broker**

RabbitMQ has been used as message broker. Install rabbitmq-server-3.7.0.exe from <http://www.rabbitmq.com/install-windows.html>. It has dependency on Erlang, install it from <http://www.rabbitmq.com/which-erlang.html>.

**Service registry (Eureka)**

[**https://spring.io/guides/gs/service-registration-and-discovery/**](https://spring.io/guides/gs/service-registration-and-discovery/)

Eureka has been used as a Service registry and discovery tool.

Go to **eureka-service\_discovery/dist** folder

Run following command:-

java -jar eureka.jar

This will start a server accessible at configured port;

<http://localhost:9000/>

**API gateway (Zuul)**

Zuul has been used as a Front door to all the incoming service calls and Eureka is being called for service discovery.

Go to **zuul-api\_gateway/dist** folder

Run following command:-

java -jar api-gateway.jar

It will start API gateway at configured port i.e. 9001

**Notification Services**

**Producer**

Producer client has been created to push the messages into queue, whenever Trade Service or Market Data Service post their data to exposed URIs:-

- Market data posted to <http://localhost:9002/producer/addMarketDataToQueue>

- Trade data posted to <http://localhost:9002/producer/addTradeToQueue>

**Consumer**

This service also acts as a consumer to message broker.

Whenever it consumes a message, it emits that message to all the subscribed channels. Web sockets are used to achieve this.

**Micro Services**

**Trade Service**

NodeJS micro service that offer API to get/delete/update trade data

**Reference data service**

NodeJS micro service to return data for all reference entities.

**Market data service**

NodeJS micro service to return live market price.

**UI**

UI is developed on ReactJS.

**Local S/W Installations**

|  |  |  |
| --- | --- | --- |
| **Service/Component** | **Port** | **Comment** |
| MongoDB | 27017 | Hosted on local host |
| Message Broker (RabbitMQ) | 5672 | Hosted on local host |

**Application Services**

|  |  |  |
| --- | --- | --- |
| **Service/Component** | **Port** | **Comment** |
| Eureka | 9000 | Hosted on local host |
| Zuul(API Gateway) | 9001 | Hosted on local host |
| Notification Service – Producer | 9002 | Hosted on local host |
| Notification Service – Consumer | 9003 | Hosted on local host |
| Trade Service | 9004 | Hosted on local host |
| Reference data service | 9005 | Hosted on local host |
| Market data service | 9006 | Hosted on local host |
| UI | 9007 | Hosted on local host |
| OAuth Server | - | Used OAuth.io service with Facebook integration |