**Coding exercise**

**Frontend:**

Providing single page of **real-time** stock pricing display on the web page. Below table is a demonstration table for your reference.

1. Stock symbol   
2. Market price (Mid) (mid = (bid + ask)/2)  
3. Arrow to show the trend of pricing going up or down from last update.

|  |  |  |
| --- | --- | --- |
| Symbol | price | Trend |
| D05:SGX | 20.1 | Up |
| O39:SGX | 8.59 | UP |
| U11:SGX | 19.7 | Down |

**Backend**:

Suppose you have a realtime pricing event from Reuters, containing the information of

1. Stock symbol  
2. bid price  
3. ask price  
4. event time

Please come out end to end solution of how you feed the information from backend to the web UI. The solution must be:

1. should be executable and the end to end flow is demo-able.  
2. should have good unit test both on front end and backend code.  
3. Should have a readme to tell how to run your application.  
4. The source code should be upload to a private bitbucket link and share to me.

5.Try to apply Cloud native best practice.

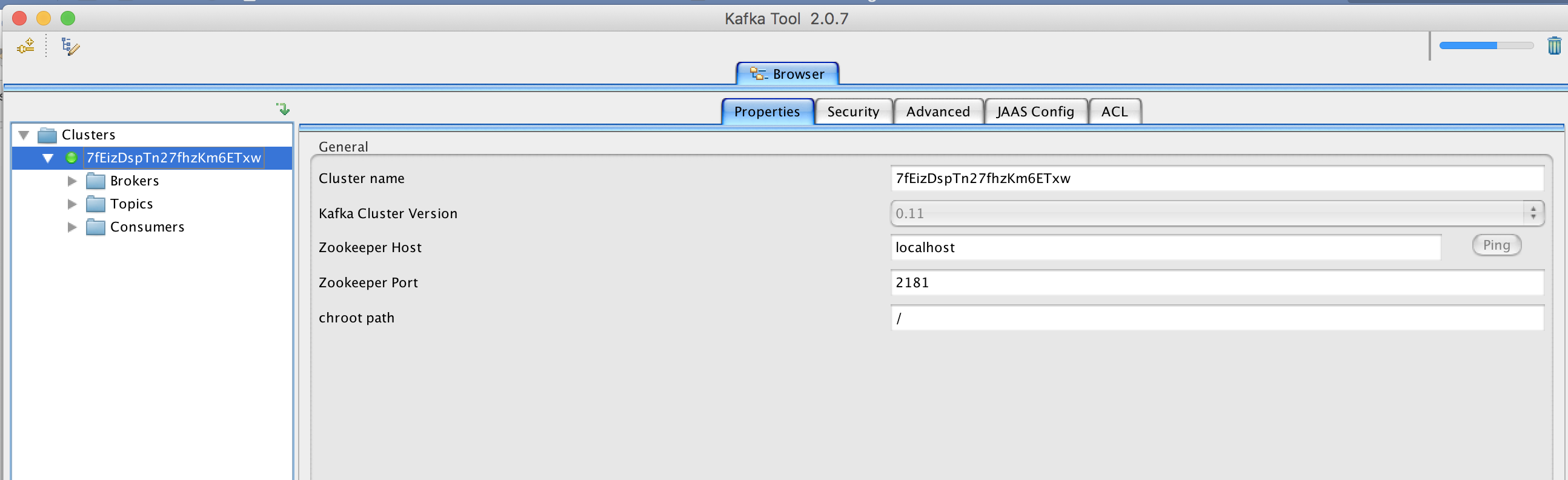
## **Project Configuration :**

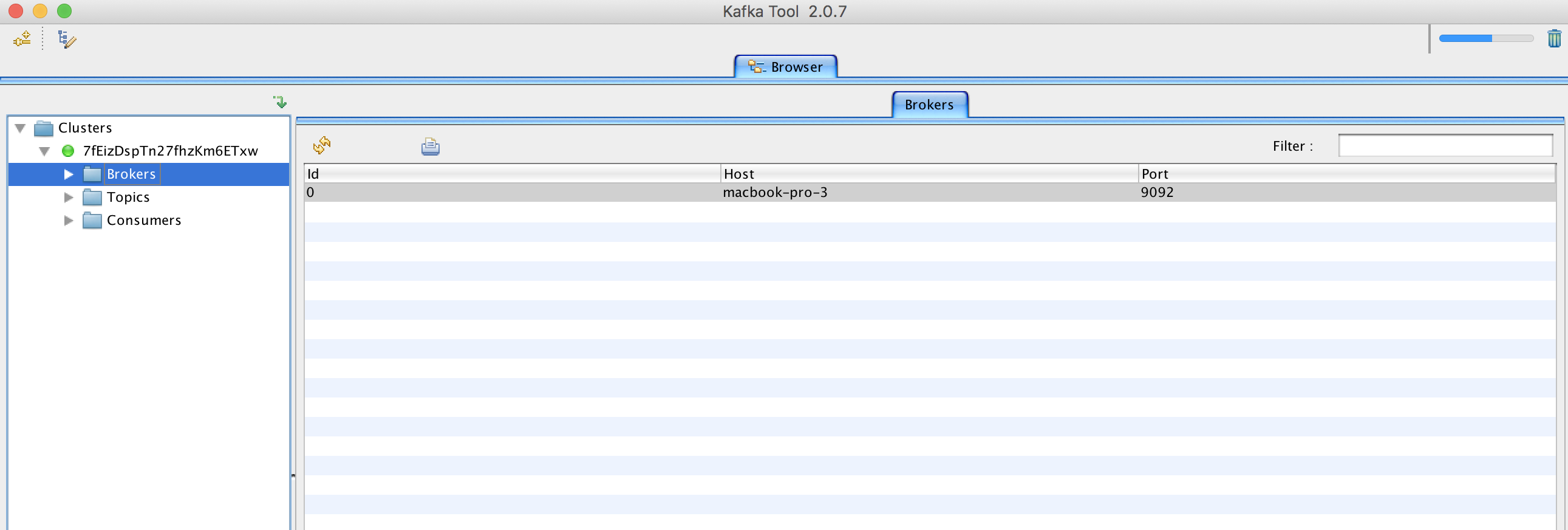
* Project is built using Spring boot 2.0 which is cloud native tech, apache Kafka, Spring Kafka , STOMP (over Http) , ReactJs
* Apache Kafka used to store and stream events . Producer produces events like Reuter pricing events and it is stored in Kafka Topic ( quickstart-events). Kafka is scalable , reliable mechanism of storing events.
* From there Consumer receives the message as instantly as its sent. Consumer to
* Consumer has Websocket registered where it pushes the message. Web socket provides duples bidirectional flow of information which cannot be catered by normal GET / POST calls.
* On Client side we have React UI which works with STOMP on Sock JS Client and receives message through websocket

To View the messages queue in kafka you can use kafka tool but is not neccasary as far as running this project is concerned.

**Kafka Tools Used :**

Kafka Topics are created and for simplicity sake Parition is 1 , replication is 1 . Topic name is quickstart-events





# **Steps to run the project :**

There are 3 pieces for the whole system :

1. Kafka
2. Spring boot backend
3. React JS UI – frontend

# **Steps to download install and run Kafka on local system**

There are many Kafka providers. But for sake of simplicity we have choose spring apache Kafka. Steps to install and run kafka on local. We can install kafka on cloud as well.

1. Downloadfrom <https://www.apache.org/dyn/closer.cgi?path=/kafka/2.6.0/kafka_2.13-2.6.0.tgz>
2. Navigate to the folder where Kafka is installed and untar it usig command : tar -xzf kafka\_2.13-2.6.0.tgz
3. cd kafka\_2.13-2.6.0
4. bin/zookeeper-server-start.sh config/zookeeper.properties
5. bin/kafka-server-start.sh config/server.properties
6. bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092

Last command is to create Topic which will store the messages from Reuters. Make sure this topic name is not already present if you have kafka running.

-- Your Kafka cluster is up and running --- For sake of simplicity we are running 1 kafka node with 1 partition and 1 consumer in a single group.

**Kafka is running on port 9092**

# **Steps to run Spring boot Consumer as microservice , which will be consuming data and pushing to websocket.**

For consumer which will be consuming data from Kafka Topic we have developed consumer microservice.

1. Download project reuterconsumer-service and import it as maven project in eclipse.
2. Run it as spring boot application using spring-boot:run command.
3. Consumer service is running at port 9001 , but no need to worry about this part unless any other service of your is running on this port .

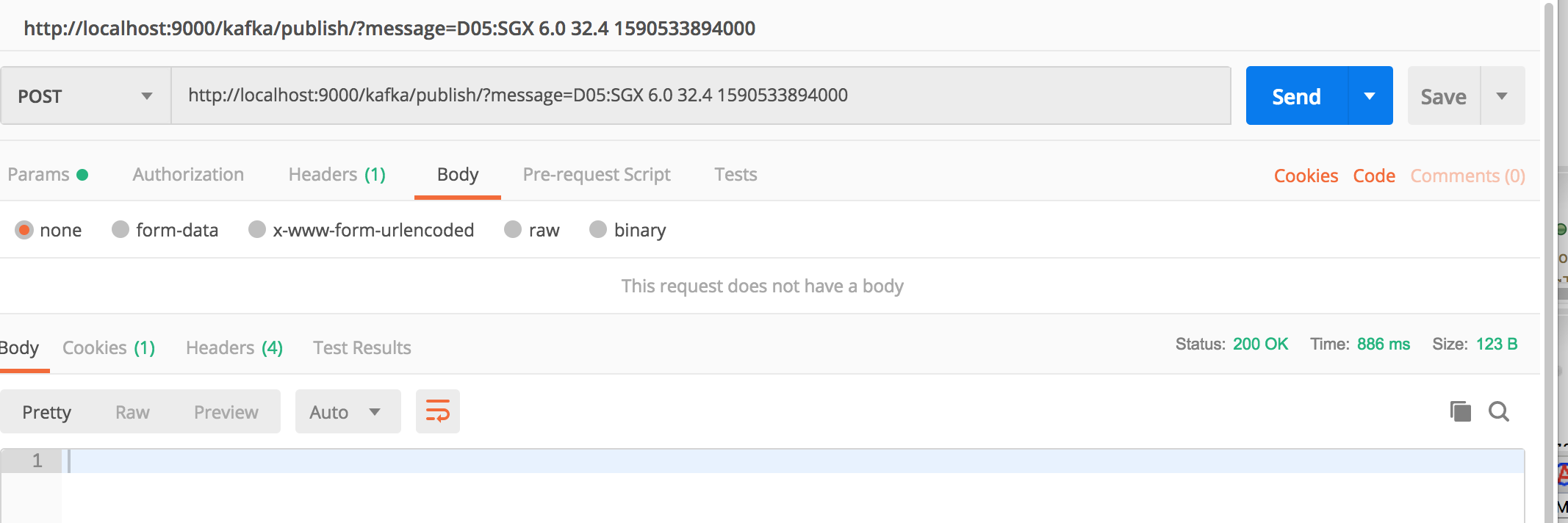
# **Steps to run Spring boot Producer as microservice , which will be used to generate data :**

For Producer we have developed this microservice to simulate creation of data. In real scenario this is provided by Reuter as source. For this step we need postman to send the post request and generate the data.Post man can be download from https://www.postman.com/downloads/

1. Download project reuterprice-producer-service and import it as maven project on local eclipse IDE.
2. Run it as Spring boot Or using maven command spring-boot:run.
3. Producer service is running on port 9000 (make sure no other service is running on this port). Now when the server started you can use postman to post values like
4. <http://localhost:9000/kafka/publish> with message in URL as

{ “message”: “D05:SGX 56.0 89.4 1590533894000”}

// info provided is symbol , bid price, ask price and timestamp in millis



Though it’s a post we should have ideally provided it in body with content type as application/json , but here I am doing so for the sake of simplicity .

**Use this data to see the trend on UI.**

**By default the trend is Up when first data comes in .**

**D05:SGX 20.1 30.4 1577855593000**

**U05:SGX 23.1 09.4 1580533993000**

**C05:SGX 56.0 89.4 1590533994000**

**D05:SGX 20.1 20.4 1577855593001**

**U05:SGX 10.1 10.4 1577855593001**

**C05:SGX 05.1 10.4 1577855593001**

# **Steps to run ReactUI Project :**

Download Project TickReaderApp.

1. You can use Viscual Studio code to see the project
2. Open Terminal and cd to folder TickReaderApp. Use npm install to install all required js in node modules. can use command line after having.
3. Npm start to start the server and it will auto launch http://localhost:4200/

After launch of project:

1. <http://localhost:4200>
2. Login with lux and dummy (its already provided on UI page)
3. Click on Listings menu after login .
4. Now send post request from producer using postman . You should see the same data on React Page. Sample data is already provided to see the up and down trend

# **Project Architecture :**

Producers

Consumer service will be running on cloud

Kafka

Kafka will be running on cloud

Browser Request to View Price Info

React App

Websocket on React Client

Consumer Service

Gi

t Repo

Bloomberg Price service

XYZ Price service

Reuter Price service