CSE 586 Project-2 (Team 2)

1. Sanyam Gupta - 50359154
2. Yogesh Kumar - 50419182

Project 2 - Report
Cryptocurrency Exchange Data Streaming Implementation (Publisher/Subscriber Model: Using Kafka)

1. Introduction:

Here we have implemented a data streaming platform for tracking crypto exchange rates from different exchanges. *Kafka* broker cluster is used as middleware leveraging its producer and consumer APIs. In addition, we have publisher module that publishes the current crypto prices to the *Kafka* broker cluster using producer API, and we have subscriber's module which lets subscribers subscribe to a particular topic. Further in the subscriber's module the subscribers keep on polling to Kafka cluster using consumer API and fetch the real-time data published for topic - cryptocurrency like BTC, ETH, DOGE in our case.

2. Architectural Model

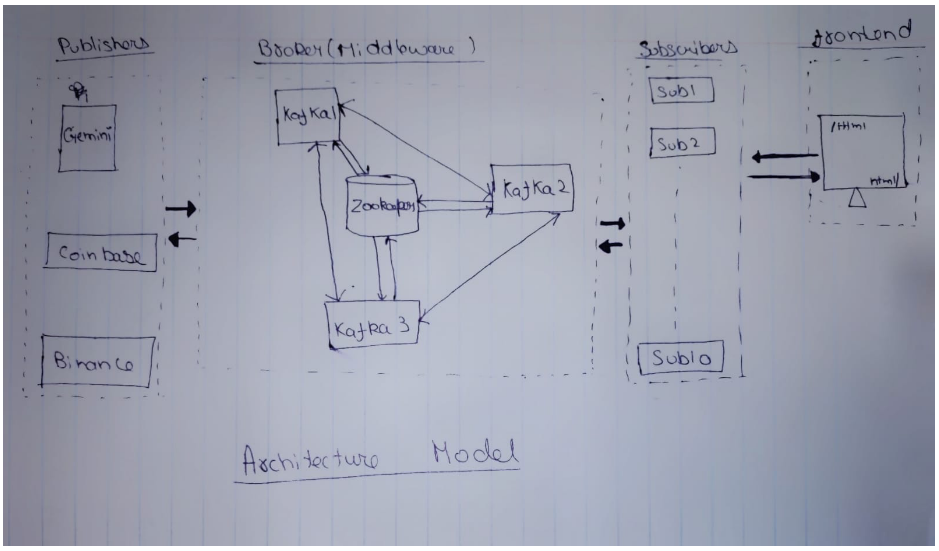
2.1 Modules:

a) **Publisher:** The publisher module continuously receives the latest prices for coins from independent external sources, cryptocurrency exchanges -Coinbase, Binance, Gemini in our case using their respective APIs. The model further publishes this event of update of new prices to the Kafka cluster module for sending to the interested subscribers using producer API. Other functionalities like advertise and de-advertise are also implemented at the publisher's end. The advertise method lets publishers publish all the possible topics to all the subscribers.

b) **Middleware (Kafka Broker Cluster)** Here we have replaced the centralized Data broker system with Kafka Broker Cluster with three nodes which is managed by a zookeeper node. Kafka cluster is where we define all the topics and their replication factor and respective number of partitions. This setup is used to notify the published data from the producers to the subscribers who have subscribed to that topic using consumer API.

c) **Subscriber:** This module has a frontend implementation which lets the user enter the subscriber id and the topic to subscribe. The subscriber system sends a request to Kafka cluster using consumer API to make a new subscription. At this end we have functionality to unsubscribe to a topic as well, here again the subscriber interacts with the middleware system to make updates for no more notifications for that topic.

1. Sanyam Gupta - 50359154
2. Yogesh Kumar - 50419182



Architecture Model

3. Kafka Cluster Implementation

The Kafka Broker cluster module is network of 3 brokers nodes managed by 1 zookeeper node. We have set replication factor of each topic i.e. BTC, ETH, DOGE as 3 with number of partition is also set to 3 for each topic. We have used wurstmeister/kafka image to dockerize kafka broker node and zookeeper node.

4. Containerization Info:

We have divided our system into multiple docker containers. We have a separate container for all 3 publishers and all 10 subscribers. Also, each kafka broker node and zookeeper node are running in a separate container.

5. Requirements

a) Docker Engine install

b) Docker-compose installed

All Other Dependencies/Requirements have been managed as part of docker images.

6. Steps to deploy the systems and run containers

Step 1 - Open terminal and go to the directory of the unzipped folder with docker compose files(.yaml).

Step 2 - $ docker-compose -f kz.yaml build
(Build container images for kafka-zookeeper cluster module)

1. Sanyam Gupta - 50359154
2. Yogesh Kumar - 50419182

Step 3 - $ docker-compose -f publisher.yaml build
(Build container images for publisher module)

Step 4 - $ docker-compose -f subscriber.yaml build
(Build container images for subscriber module)

Step 5 - $ docker-compose -f kz.yaml up
(Run container images for kafka-zookeeper cluster module)

Step 6 - $ docker-compose -f publisher.yaml up
(Run container images for publisher module)

Step 7- $ docker-compose -f subscriber.yaml up
(Run container images for subscriber module)

7. Steps to interact with the Kafka Module and create topics

Step 1 - Open a new terminal session for the publisher program to execute.

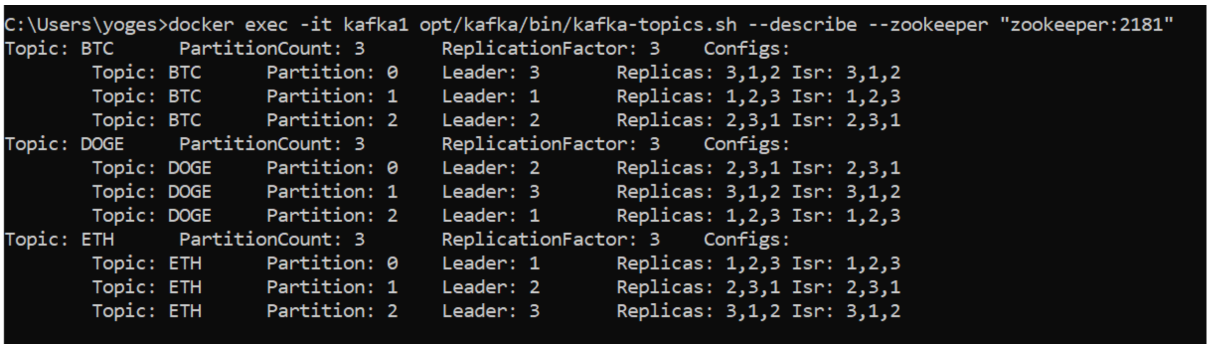
Step 2 - $ docker exec -it kafka1 /bin/sh

$ docker exec - opt/kafka/bin/kafka-topics.sh --create –zookeeper zookeeper:2181" --replication-factor 3 --partitions 3 --topic BTC

$opt/kafka/bin/kafka-topics.sh --create --zookeeper "zookeeper:2181"   --replication-factor 3 --partitions 3 --topic ETH

$opt/kafka/bin/kafka-topics.sh --create --zookeeper "zookeeper:2181" --replication-factor 3 --partitions 3 --topic DOGE

Step 3 - $opt/kafka/bin/kafka-topics.sh --describe –zookeeper “zookeeper:2181"
To describe the details of topic partitions and leaders



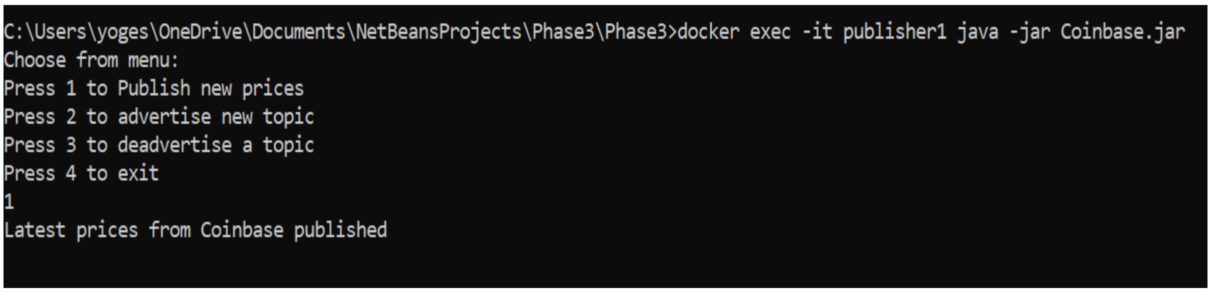
```
C:\Users\yoges>docker exec -it kafka1 opt/kafka/bin/kafka-topics.sh --describe --zookeeper "zookeeper:2181"
Topic: BTC      PartitionCount: 3       ReplicationFactor: 3    Configs:
        Topic: BTC      Partition: 0    Leader: 3       Replicas: 3,1,2 Isr: 3,1,2
        Topic: BTC      Partition: 1    Leader: 1       Replicas: 1,2,3 Isr: 1,2,3
        Topic: BTC      Partition: 2    Leader: 2       Replicas: 2,3,1 Isr: 2,3,1
Topic: DOGE     PartitionCount: 3       ReplicationFactor: 3    Configs:
        Topic: DOGE     Partition: 0    Leader: 2       Replicas: 2,3,1 Isr: 2,3,1
        Topic: DOGE     Partition: 1    Leader: 3       Replicas: 3,1,2 Isr: 3,1,2
        Topic: DOGE     Partition: 2    Leader: 1       Replicas: 1,2,3 Isr: 1,2,3
Topic: ETH      PartitionCount: 3       ReplicationFactor: 3    Configs:
        Topic: ETH      Partition: 0    Leader: 1       Replicas: 1,2,3 Isr: 1,2,3
        Topic: ETH      Partition: 1    Leader: 2       Replicas: 2,3,1 Isr: 2,3,1
        Topic: ETH      Partition: 2    Leader: 3       Replicas: 3,1,2 Isr: 3,1,2
```

8. Steps to interact with the publisher

Step 1 - Open a new terminal session for the publisher program to execute.

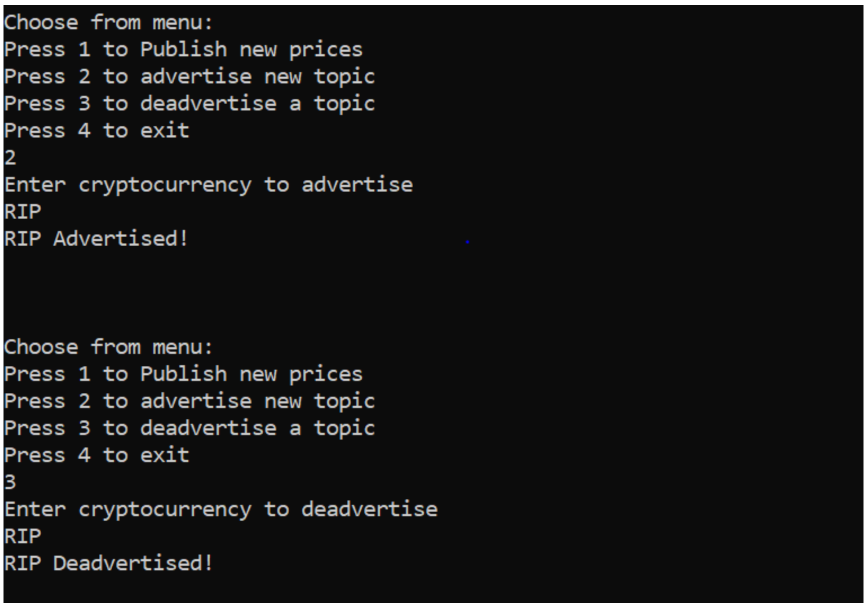
Step 2 - $ docker exec -it Coinbase java -jar Coinbase.jar
$ docker exec -it Binance java -jar Binance.jar
$ docker exec -it Gemini java -jar Gemini.jar

This will make a session with the container(publisher1).



```
C:\Users\yoges\OneDrive\Documents\NetBeansProjects\Phase3\Phase3>docker exec -it publisher1 java -jar Coinbase.jar
Choose from menu:
Press 1 to Publish new prices
Press 2 to advertise new topic
Press 3 to deadvertise a topic
Press 4 to exit
1
Latest prices from Coinbase published
```

Similarly, we can interact with other publisher 2 and publisher 3.



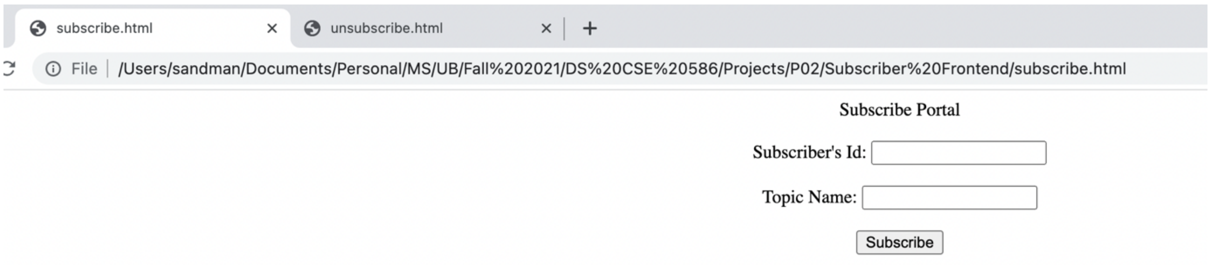
```
Choose from menu:
Press 1 to Publish new prices
Press 2 to advertise new topic
Press 3 to deadvertise a topic
Press 4 to exit
2
Enter cryptocurrency to advertise
RIP
RIP Advertised!


Choose from menu:
Press 1 to Publish new prices
Press 2 to advertise new topic
Press 3 to deadvertise a topic
Press 4 to exit
3
Enter cryptocurrency to deadvertise
RIP
RIP Deadvertised!
```

Advertise and deadvertise example.

9. Steps to interact with the Subscriber

Step 1 We have two web pages to interact with subscribers. Both of these are present in Subscriber frontend folder. One is subscribe.html to subscribe to a topic as below. You can enter any sub id and choose a topic from BTC, ETH, DOGE which are available in our case to subscribe from.

1. Sanyam Gupta - 50359154
2. Yogesh Kumar - 50419182



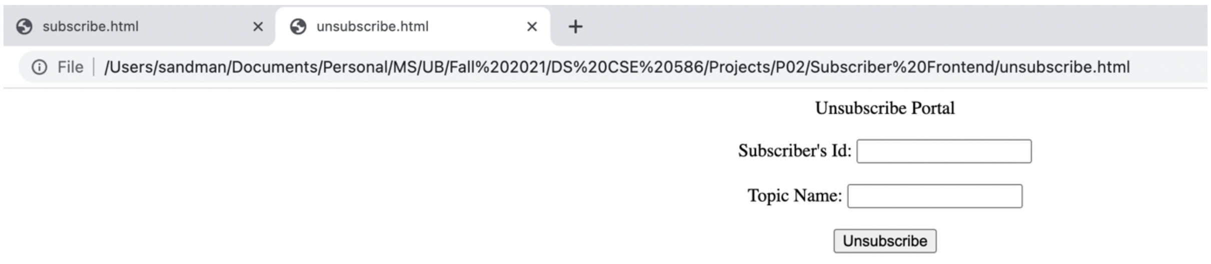
```
/Users/sandman/Documents/Personal/MS/UB/Fall%202021/DS%20CSE%20586/Projects/P02/Subscriber%20Frontend/subscribe.html
```

Subscribe Portal

Subscriber's Id:

Topic Name:

Step 2 We can unsubscribe to any topic using html page unsubscribe.html



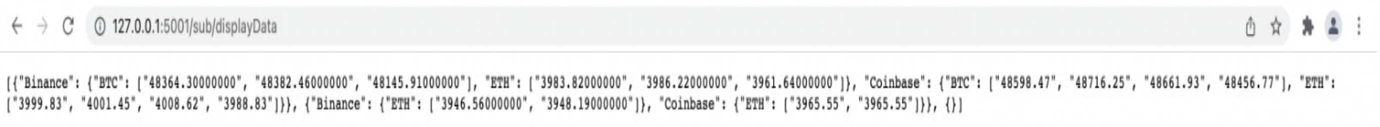
```
/Users/sandman/Documents/Personal/MS/UB/Fall%202021/DS%20CSE%20586/Projects/P02/Subscriber%20Frontend/unsubscribe.html
```

Unsubscribe Portal

Subscriber's Id:

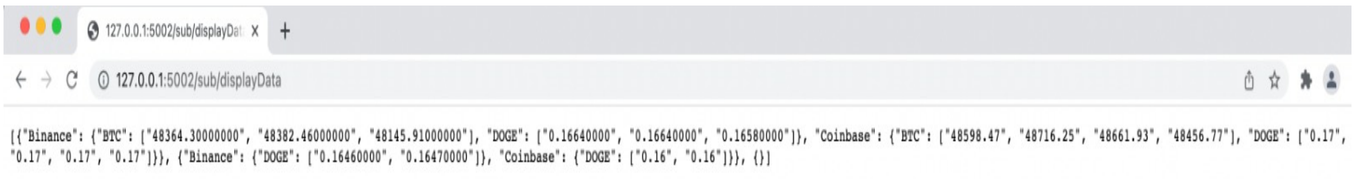
Topic Name:

Step 3 We can view the data received by subscribers by hitting their respective APIs. The route is **/sub/displayData** but the IP and port combination varies for each subscriber.



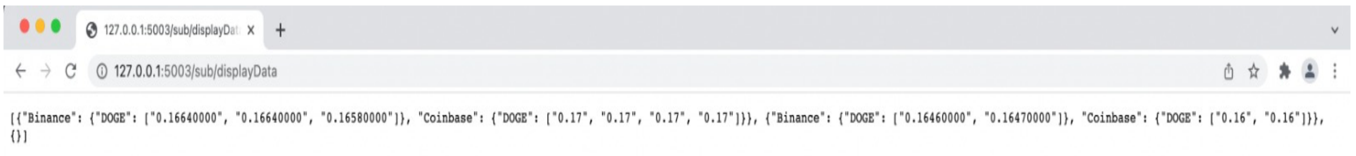
```
127.0.0.1:5001/sub/displayData
```

```
[{"Binance": {"BTC": ["48364.30000000", "48382.46000000", "48145.91000000"], "ETH": ["3983.82000000", "3986.22000000", "3961.64000000"]}, "Coinbase": {"BTC": ["48598.47", "48716.25", "48661.93", "48456.77"], "ETH":
["3999.83", "4001.45", "4008.62", "3988.83"]}}, {"Binance": {"ETH": ["3946.56000000", "3948.19000000"]}, "Coinbase": {"ETH": ["3965.55", "3965.55"]}}, {}]
```



```
127.0.0.1:5002/sub/displayData
```

```
[{"Binance": {"BTC": ["48364.30000000", "48382.46000000", "48145.91000000"], "DOGE": ["0.16640000", "0.16640000", "0.16580000"]}, "Coinbase": {"BTC": ["48598.47", "48716.25", "48661.93", "48456.77"], "DOGE": ["0.17",
"0.17", "0.17", "0.17"]}}, {"Binance": {"DOGE": ["0.16460000", "0.16470000"]}, "Coinbase": {"DOGE": ["0.16", "0.16"]}}, {}]
```

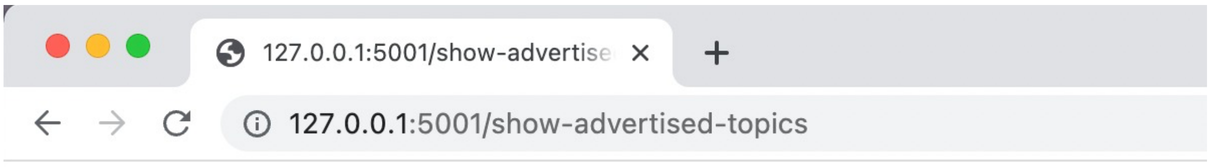


```
127.0.0.1:5003/sub/displayData
```

```
[{"Binance": {"DOGE": ["0.16640000", "0.16640000", "0.16580000"]}, "Coinbase": {"DOGE": ["0.17", "0.17", "0.17", "0.17"]}}, {"Binance": {"DOGE": ["0.16460000", "0.16470000"]}, "Coinbase": {"DOGE": ["0.16", "0.16"]}},
{}]
```

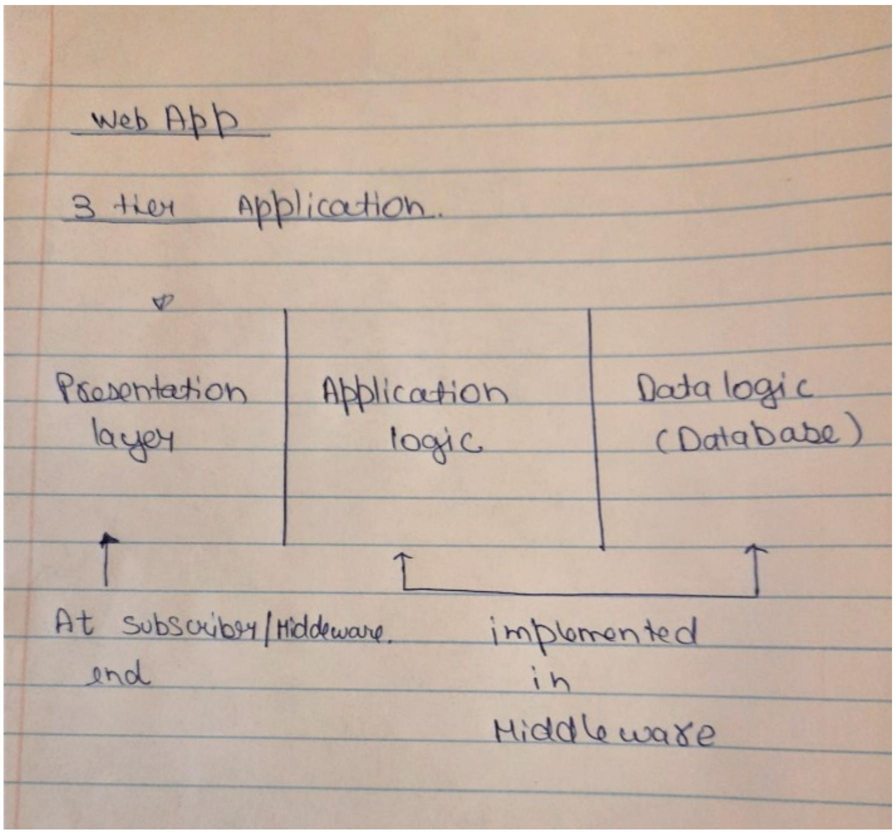
Step 4 We can view the advertised topics to all the subscribers by hitting the advertisement API. The route is **/middleware/show-advertised-topics**.

1. Sanyam Gupta - 50359154
2. Yogesh Kumar - 50419182


127.0.0.1:5001/show-advertise
127.0.0.1:5001/show-advertised-topics

```
["XRP", "XRP2"]
```

10. Application Architecture



11. Summary:

We implemented a data streaming platform for tracking crypto exchange rates from different exchanges using the distributed publish/subscribe model of indirect communication, enabling loose coupling, thus making our system dynamic, scalable, robust, heterogeneous, and asynchronous.

12. Team Contribution Distribution:

Yogesh : Publisher Module, Subscriber Web App, Kafka Cluster, Report

Sanyam : Subscriber Module, Kafka Cluster, Video Demo, Report

13. References:

Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5 © Pearson Education 2012

https://hub.docker.com/r/wurstmeister/kafka

https://medium.com/@PierreKieffer/deploy-a-kafka-multi-node-cluster-with-docker-72878ddbaf96

https://www.kaaiot.com/blog/kafka-docker

https://medium.com/@marcelo.hossomi/running-kafka-in-docker-machine-64d1501d6f0b