

CODE LOGIC

- 1) First we need to start Kafka and Zookeeper services and create topic – patients_vital_info .

COMMAND TO START ZOOKEEPER SERVER

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

COMMAND TO START KAFKA SERVER

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

STATEMENT TO CREATE TOPICS

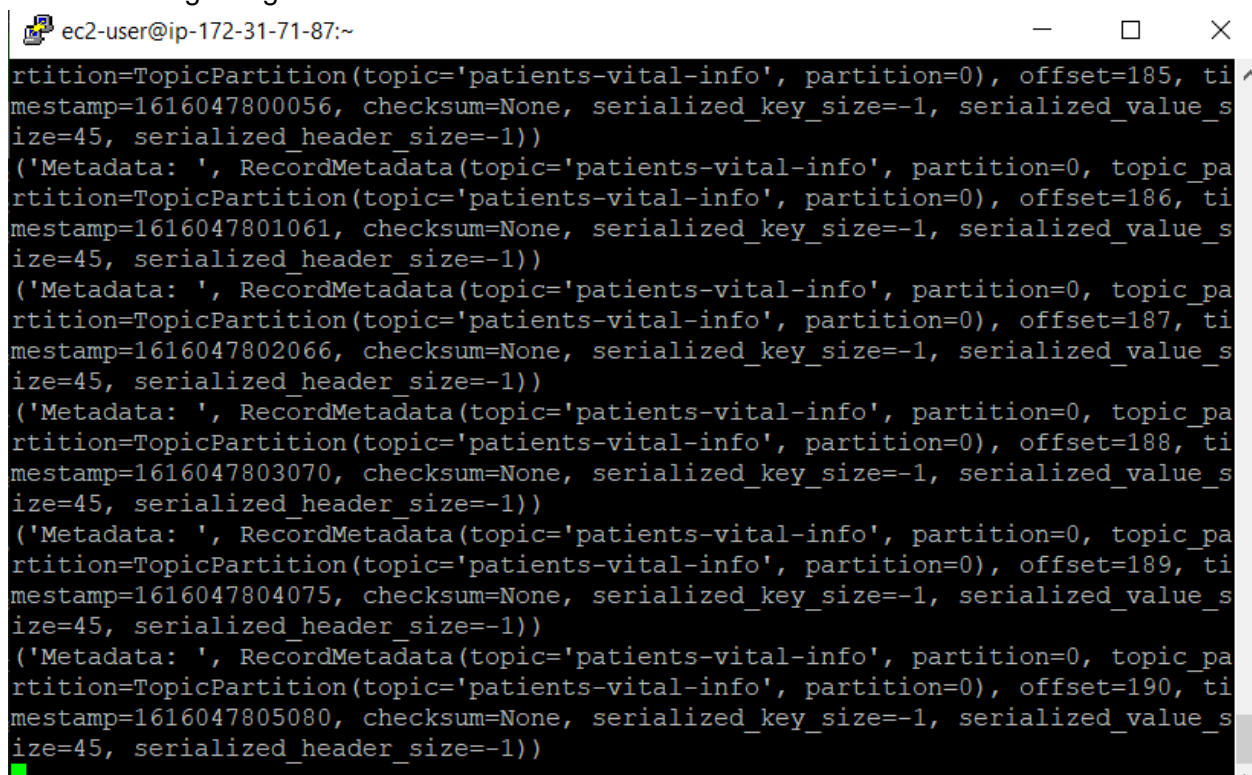
```
bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic patients-vital-info
```

STATEMENT TO LIST TOPICS


```
bin/kafka-topics.sh --list --bootstrap-server localhost:9092
```

STATEMENT TO START CONSUMER FROM CONSOLE

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic patients-vital-info --from-beginning
```



```
ec2-user@ip-172-31-71-87:~
rtition=TopicPartition(topic='patients-vital-info', partition=0), offset=185, ti
mestamp=1616047800056, checksum=None, serialized_key_size=-1, serialized_value_s
ize=45, serialized_header_size=-1))
('Metadata: ', RecordMetadata(topic='patients-vital-info', partition=0, topic_pa
rtition=TopicPartition(topic='patients-vital-info', partition=0), offset=186, ti
mestamp=1616047801061, checksum=None, serialized_key_size=-1, serialized_value_s
ize=45, serialized_header_size=-1))
('Metadata: ', RecordMetadata(topic='patients-vital-info', partition=0, topic_pa
rtition=TopicPartition(topic='patients-vital-info', partition=0), offset=187, ti
mestamp=1616047802066, checksum=None, serialized_key_size=-1, serialized_value_s
ize=45, serialized_header_size=-1))
('Metadata: ', RecordMetadata(topic='patients-vital-info', partition=0, topic_pa
rtition=TopicPartition(topic='patients-vital-info', partition=0), offset=188, ti
mestamp=1616047803070, checksum=None, serialized_key_size=-1, serialized_value_s
ize=45, serialized_header_size=-1))
('Metadata: ', RecordMetadata(topic='patients-vital-info', partition=0, topic_pa
rtition=TopicPartition(topic='patients-vital-info', partition=0), offset=189, ti
mestamp=1616047804075, checksum=None, serialized_key_size=-1, serialized_value_s
ize=45, serialized_header_size=-1))
('Metadata: ', RecordMetadata(topic='patients-vital-info', partition=0, topic_pa
rtition=TopicPartition(topic='patients-vital-info', partition=0), offset=190, ti
mestamp=1616047805080, checksum=None, serialized_key_size=-1, serialized_value_s
ize=45, serialized_header_size=-1))
```

 ec2-user@ip-172-31-71-87:~/downloads/kafka_2.12-2.3.0

```
{ "HeartBeat": 164, "BP": 69, "CustomerID": 3}
{ "HeartBeat": 168, "BP": 73, "CustomerID": 4}
{ "HeartBeat": 172, "BP": 72, "CustomerID": 5}
{ "HeartBeat": 209, "BP": 73, "CustomerID": 1}
{ "HeartBeat": 174, "BP": 71, "CustomerID": 2}
{ "HeartBeat": 175, "BP": 70, "CustomerID": 3}
{ "HeartBeat": 157, "BP": 68, "CustomerID": 4}
{ "HeartBeat": 169, "BP": 72, "CustomerID": 5}
{ "HeartBeat": 220, "BP": 74, "CustomerID": 1}
{ "HeartBeat": 153, "BP": 70, "CustomerID": 2}
{ "HeartBeat": 154, "BP": 67, "CustomerID": 3}
{ "HeartBeat": 174, "BP": 70, "CustomerID": 4}
{ "HeartBeat": 177, "BP": 67, "CustomerID": 5}
{ "HeartBeat": 167, "BP": 76, "CustomerID": 1}
{ "HeartBeat": 174, "BP": 89, "CustomerID": 2}
{ "HeartBeat": 155, "BP": 71, "CustomerID": 3}
{ "HeartBeat": 175, "BP": 73, "CustomerID": 4}
{ "HeartBeat": 161, "BP": 69, "CustomerID": 5}
{ "HeartBeat": 207, "BP": 75, "CustomerID": 1}
{ "HeartBeat": 159, "BP": 66, "CustomerID": 2}
{ "HeartBeat": 153, "BP": 71, "CustomerID": 3}
{ "HeartBeat": 153, "BP": 72, "CustomerID": 4}
{ "HeartBeat": 171, "BP": 66, "CustomerID": 5}
```

- 2) Then we have to create producer application to read from RDS and push the message into the topic in below format and list the messages in the topic. Here we used mysql connection credentials to connect RDS and using cursor object we have to fetch patient_vital_info from RDS. Then setup producer application to load the RDS data into the patients-vital-info kafka topic.
python kafka_produce_patient_vitals.py
- 3) Next step is to create Pyspark application to read all messages from above created Kafka topic into HDFS file in parquet. We have to use spark2-submit command to execute spark job. Data will be stored in hdfs.
export SPARK_KAFKA_VERSION=0.10
spark2-submit --jars spark-sql-kafka-0-10_2.11-2.3.0.jar
kafka_spark_patient_vitals.py host port topic_name
- 4) Next, we have to create external hive table patients_vital_info for the threshold data.

- 5) Then we have to create threshold reference HBase table with 3 columns family (attribute, alert and limit) and then we have to insert 12 records across the 3 columns families.
- 6) Again, we have to create external hive table on HBase table and insert records from HBase column families. This is to store and query threshold data to check whether incoming data have any anomalies.
- 7) Then we have to create Sqoop job to extract patient's information RDS database into hive table. This Sqoop job is to push the contact data of the patients from RDS into the Hive table. Then to store and query contact and other information to check whether incoming data have any anomalies.
- 8) Then next to create spark application to read data from HDFS and compare these data with above created HBase table. This is to collect data from Spark application to put in a queue and feed to Consumer application.
`export SPARK_KAFKA_VERSION=0.10`
`spark2-submit --jars spark-sql-kafka-0-10_2.11-2.3.0.jar`
`kafka_spark_generate_alerts.py host port topic_name`
- 9) Then at last we have to setup AWS SNS (Simple Notification System) configuration to send email on registered email ID to send patients information. We have to send SNS push notification to the subscribed email-id to report any anomalies.
`python kafka_consume_alerts.py`