1.Install docker desktop

https://docs.docker.com/docker-for-windows/install/

2. Setup Kafka on local machine. Use below as reference :

https://docs.confluent.io/current/quickstart/ce-docker-quickstart.html

Note : Pls change the KAFKA_ADVERTISED_LISTENERS to point to docker NAT's ip (which you can get by running ipconfig command on cmd) rather than localhost.

```
broker:
  image: confluentinc/cp-server:5.5.1
  hostname: broker
  container_name: broker
  depends_on:
    - zookeeper
  ports:
    - "9092:9092"
  environment:
    KAFKA_BROKER_ID: 1
    KAFKA_ZOOKEEPER_CONNECT: 'zookeeper:2181'
    KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
    KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://broker:29092,PLAINTEXT_HOST://10.0.75.1:9092
    KAFKA_METRIC_REPORTERS: io.confluent.metrics.reporter.ConfluentMetricsReporter
    KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
```

You should then see the below once everything is up running

```
PS C:\Users\Jyotsna.Gajjala\dtt\cp-all-in-one\cp-all-in-one> docker-compose ps
The system cannot find the path specified.
     Name                   Command               State                         Ports
-------------------------------------------------------------------------------------------------------
broker            /etc/confluent/docker/run       Up             0.0.0.0:9092->9092/tcp
connect           /etc/confluent/docker/run       Up             0.0.0.0:8083->8083/tcp, 9092/tcp
control-center    /etc/confluent/docker/run       Up             0.0.0.0:9021->9021/tcp
ksql-datagen      bash -c echo Waiting for K ...  Up
ksqldb-cli        /bin/sh                         Up
ksqldb-server     /etc/confluent/docker/run       Up (healthy)   0.0.0.0:8088->8088/tcp
rest-proxy        /etc/confluent/docker/run       Up             0.0.0.0:8082->8082/tcp
schema-registry   /etc/confluent/docker/run       Up             0.0.0.0:8081->8081/tcp
zookeeper         /etc/confluent/docker/run       Up             0.0.0.0:2181->2181/tcp, 2888/tcp, 3888/tcp
```

a.Link to control-center **: http://<docker-ip>:9021**

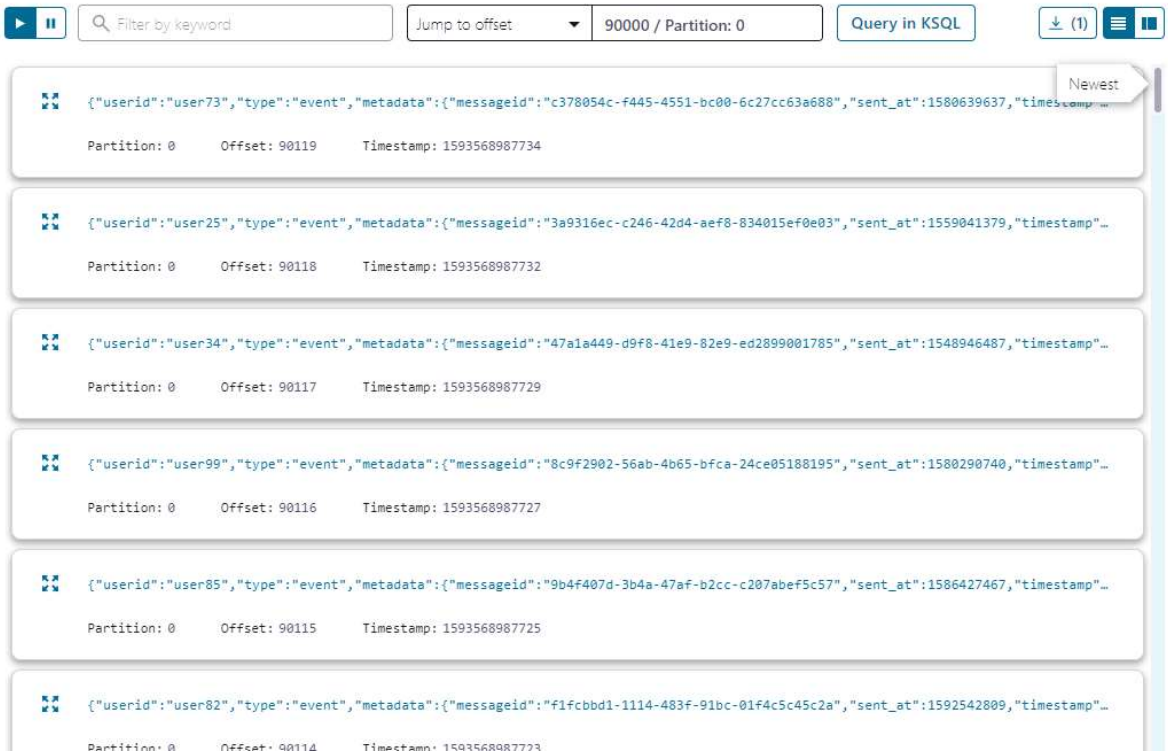
b.Create topic 'user-events-1'

3. Run docker container for MessageGenerator.py using below steps :

docker build -t producer -f .\ck-producer\Dockerfile .

docker run -p 5000:5000 --name producer producer

You should see the user events in Json format being produced on the topic --- 'user-events-1'



Sample json schema –


```
{
 "userid": "user4",
 "type": "event",
 "metadata": {
  "messageid": "01b3bb2f-3882-4b85-bd24-c6d322510893",
  "sent_at": 1582598565,
  "timestamp": 1582598565,
  "received_at": 1582598665,
  "apikey": "apikey7",
  "spaceid": "spaceid7",
  "version": "version7"
 },
 "event": "played movie",
 "event_data": {
  "movieid": "MIM9Yeb"
 }
}
```

4. Pyspark Consumer/ Kafka Producer

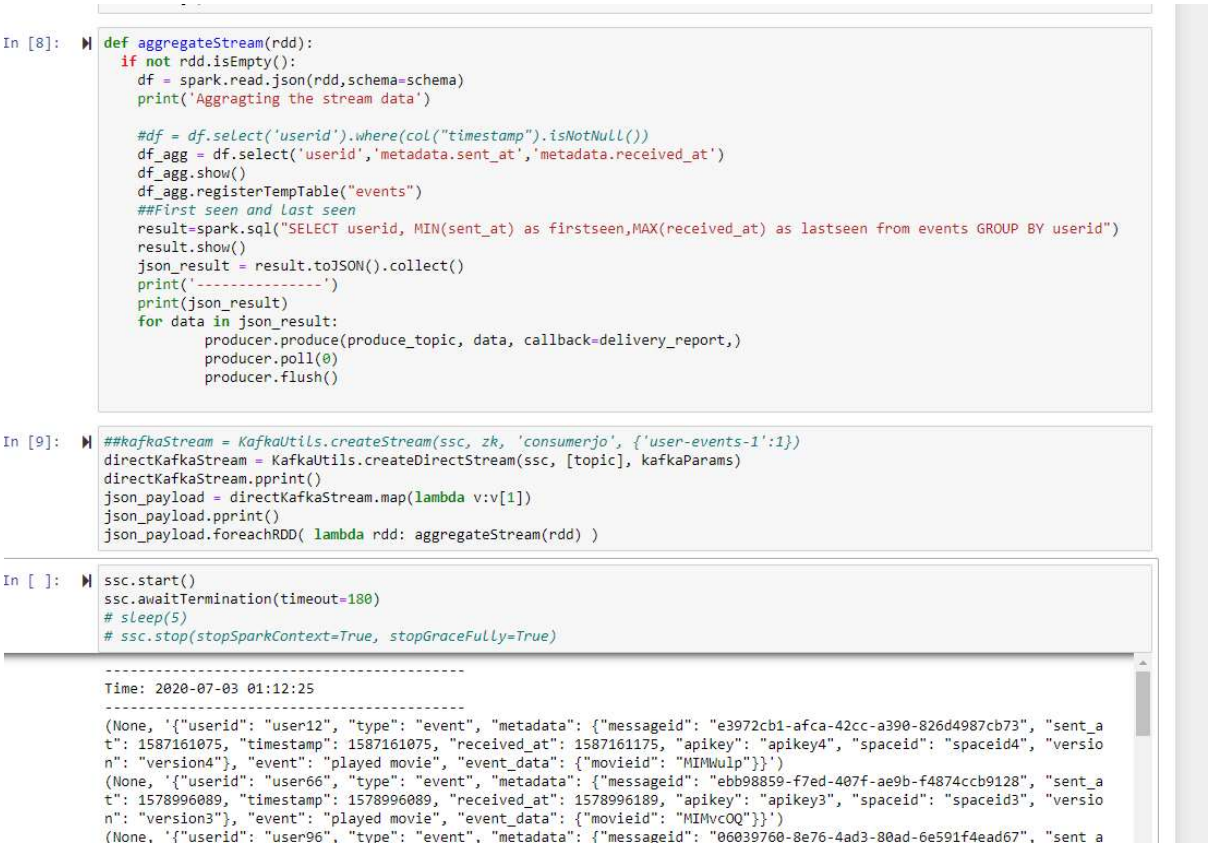
The below command spins up a docker container wit pyspark requirements

docker run -it -p 8888:8888 jupyter/pyspark-notebook

Login using below url and use the token that you see on your terminal after installation

http://<docker-ip>:8888

We can now setup spark streaming contexts and use KafkaUtils to start consuming from our kafka brokers.



```
In [8]:
def aggregateStream(rdd):
  if not rdd.isEmpty():
    df = spark.read.json(rdd,schema=schema)
    print('Aggragting the stream data')

    #df = df.select('userid').where(col("timestamp").isNotNull())
    df_agg = df.select('userid','metadata.sent_at','metadata.received_at')
    df_agg.show()
    df_agg.registerTempTable("events")
    ##First seen and last seen
    result=spark.sql("SELECT userid, MIN(sent_at) as firstseen,MAX(received_at) as lastseen from events GROUP BY userid")
    result.show()
    json_result = result.toJSON().collect()
    print('---------------')
    print(json_result)
    for data in json_result:
            producer.produce(produce_topic, data, callback=delivery_report,)
            producer.poll(0)
            producer.flush()
```

```
In [9]:
##kafkaStream = KafkaUtils.createStream(ssc, zk, 'consumerjo', {'user-events-1':1})
directKafkaStream = KafkaUtils.createDirectStream(ssc, [topic], kafkaParams)
directKafkaStream.pprint()
json_payload = directKafkaStream.map(lambda v:v[1])
json_payload.pprint()
json_payload.foreachRDD( lambda rdd: aggregateStream(rdd) )
```

```
In [ ]:
ssc.start()
ssc.awaitTermination(timeout=180)
# sleep(5)
# ssc.stop(stopSparkContext=True, stopGraceFully=True)
```

```
-------------------------------------------
Time: 2020-07-03 01:12:25
-------------------------------------------
(None, '{"userid": "user12", "type": "event", "metadata": {"messageid": "e3972cb1-afca-42cc-a390-826d4987cb73", "sent_a
t": 1587161075, "timestamp": 1587161075, "received_at": 1587161175, "apikey": "apikey4", "spaceid": "spaceid4", "versio
n": "version4"}, "event": "played movie", "event_data": {"movieid": "MIMWulp"}}')
(None, '{"userid": "user66", "type": "event", "metadata": {"messageid": "ebb98859-f7ed-407f-ae9b-f4874ccb9128", "sent_a
t": 1578996089, "timestamp": 1578996089, "received_at": 1578996189, "apikey": "apikey3", "spaceid": "spaceid3", "versio
n": "version3"}, "event": "played movie", "event_data": {"movieid": "MIMvcOQ"}}')
(None, '{"userid": "user96", "type": "event", "metadata": {"messageid": "06039760-8e76-4ad3-80ad-6e591f4ead67", "sent_a
```

Please refer to pyspark-kafka-consumer.ipynb for the transformations applied to generate user summary.

5.And finally, Initialize confluent kafka producer to publish the user summary in json to the topic ‘user-summary-aggregated’(screenshot in the next page)

ALL TOPICS >

user-summary-aggregated

Overview | Messages | Schema | Configuration

Bytes in/sec
11.0

Bytes out/sec
0

Message fields

- topic
- partition
- offset
- timestamp
- timestampType
- headers
- key
- value
  - userid
  - firstseen
  - lastseen

Filter by keyword | Jump to offset | 3 / Partition: 0 | Query in KSQL | (1)

Newest

{"userid":"user66","firstseen":1578996089,"lastseen":1578996189}
Partition: 0 Offset: 9 Timestamp: 1593738752929

{"userid":"user12","firstseen":1587161075,"lastseen":1587161175}
Partition: 0 Offset: 8 Timestamp: 1593738752927

{"userid":"user29","firstseen":1561067444,"lastseen":1561067544}
Partition: 0 Offset: 7 Timestamp: 1593738752924

{"userid":"user80","firstseen":1586703036,"lastseen":1586703136}
Partition: 0 Offset: 6 Timestamp: 1593738752922

{"userid":"user96","firstseen":1553948893,"lastseen":1553948993}
Partition: 0 Offset: 5 Timestamp: 1593738752574