Open in Colab

Real Time Flight Data Streaming - Kafka Consumer

In [ ]:
```
!pip install kafka-python
```

Consuming data the data being streamed using Kafka and plotting it.

In [1]:
```
# import statements
from time import sleep
from kafka import KafkaConsumer
import datetime as dt
import matplotlib
import matplotlib.pyplot as plt
from json import loads
```

Step 1

Set topic, connect consumer to producer and set plotting function

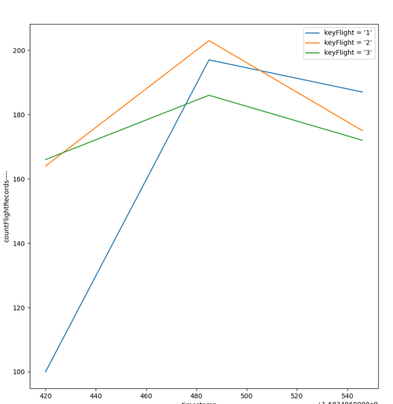
In [2]:
```
# set the topic of the consumer
topic = 'flightTopic'

# function to connect to kafka consumer
# taken from tutorial week 9
def connect_kafka_consumer():
    _consumer = None
    try:
        _consumer = KafkaConsumer(topic,
                                  consumer_timeout_ms=10000, # stop iteration if no message after 10 sec
                                  auto_offset_reset='latest', # comment this if you don't want to consume earliest available message
                                  bootstrap_servers=['192.168.86.48:9092'],
                                  value_deserializer=lambda x: loads(x.decode('ascii')),
                                  api_version=(0, 11,5))
    except Exception as ex:
        print('Exception while connecting Kafka')
        print(str(ex))
    finally:
        return _consumer


# function to set up the plotting area
# taken from tutorial week 9
def init_plots():
    try:
        width = 9.5
        height = 10
        fig = plt.figure(figsize=(width,height)) # create new figure
        ax = fig.add_subplot(111) # adding the subplot axes to the given grid position
        #fig.suptitle('Number of flights every 30 secs by keyFlight') # giving figure a title
        #ax.set_xlabel('timestamp')
        #ax.set_ylabel('countFlightRecords')
        fig.show() # displaying the figure
        fig.canvas.draw() # drawing on the canvas
        return fig, ax
    except Exception as ex:
        print(str(ex))
```

Step 2

Process the data into the desired batch format and plot it

In [3]:
```
# NEW
# this line is needed for the inline display of graphs in Jupyter Notebook
%matplotlib notebook


# function to consume and display the data sent by the prodcuer
# partially taken from tutorial week 9
def countFlightRecords(consumer, fig, ax):
    try:

        ## 1. GENERATE THE DATA CONTAINERS

        time_start = 0

        count_of_flights = {1:0, 2:0, 3:0}

        two_min_flight_data = []

        data_plot = False


        ## 2. PLOTING DATA CONATINERS

        # container for x and y values
        y1, y2, y3, x1, x2, x3 = [], [], [], [], [], []

        # print the start of the process
        print('Processing records ...')


        for message in consumer:

            # in case of the first batch
            if time_start == 0:

                # save the timestamp of a batch
                ts_of_batch = message.value[0]['ts']

                # set new ts starting point
                time_start = ts_of_batch

            # for any further batches
            else:

                # save the timestamp of a batch
                ts_of_batch = message.value[0]['ts']

                # if condition when two minutes have passed
                if ts_of_batch - time_start > 60:

                    # set binary switch
                    data_plot = True

                    print('number of flights for keyFlight = '1' : ', count_of_flights[1])
                    print('number of flights for keyFlight = '2' : ', count_of_flights[2])
                    print('number of flights for keyFlight = '3' : ', count_of_flights[3])
                    print('----------------------')

                    # dict to save the flight countof two minutes
                    two_min_tup = ()

                    # append count for each of the three keyFlights
                    two_min_tup = (ts_of_batch, count_of_flights[1], count_of_flights[2], count_of_flights[3])

                    # append the dict
                    two_min_flight_data.append(two_min_tup)

                    # append the data for our plotting input
                    x1.append(ts_of_batch)
                    x2.append(ts_of_batch)
                    x3.append(ts_of_batch)
                    y1.append(count_of_flights[1])
                    y2.append(count_of_flights[2])
                    y3.append(count_of_flights[3])

                    # reset the counter for eah keyFlight
                    count_of_flights[1] = 0
                    count_of_flights[2] = 0
                    count_of_flights[3] = 0

                    # set the new 'last' timestamp
                    time_start = ts_of_batch


            # save the list of flights received at each iteration
            data_batch_list = message.value

            # loop through all the lights in the list
            for i in range(0,len(data_batch_list)):

                # each entry in the list is a dict containing a flight
                flight_dict = data_batch_list[i]


                if flight_dict['DAY_OF_WEEK'] == '1':

                    count_of_flights[1] += 1


                elif flight_dict['DAY_OF_WEEK'] == '2':

                    count_of_flights[2] += 1

                elif flight_dict['DAY_OF_WEEK'] == '3':

                    count_of_flights[3] += 1


            ## 3. PLOT THE DATA

            # start plotting if there are three or more 2 min flight count totals
            # and if variable data_plot is True
            if len(two_min_flight_data) >= 3 and data_plot is True:


                # visualize and update the graph
                ax.clear()
                ax.plot(x1, y1, label = 'keyFlight = '1'')
                ax.plot(x2, y2, label = 'keyFlight = '2'')
                ax.plot(x3, y3, label = 'keyFlight = '3'')

                ax.set_xlabel('timestamp----')
                ax.set_ylabel('countFlightRecords----')
                plt.legend()
                fig.canvas.draw()

                # set binary switch
                data_plot = False

                # removing the item in the first position
                x1.pop(0)
                x2.pop(0)
                x3.pop(0)
                y1.pop(0)
                y2.pop(0)
                y3.pop(0)


        plt.close('all')

    except Exception as ex:
        print(str(ex))
```

In [ ]:
```
if __name__ == '__main__':

    # call all required functions
    consumer = connect_kafka_consumer()
    fig, ax = init_plots()
    countFlightRecords(consumer, fig, ax)
```




```
Processing records ...
number of flights for keyFlight = '1' :  100
number of flights for keyFlight = '2' :  164
number of flights for keyFlight = '3' :  166
----------------------
number of flights for keyFlight = '1' :  197
number of flights for keyFlight = '2' :  203
number of flights for keyFlight = '3' :  186
----------------------
number of flights for keyFlight = '1' :  187
number of flights for keyFlight = '2' :  175
number of flights for keyFlight = '3' :  172
----------------------
```

In [ ]: