Pykafka Integration with Python/Flask

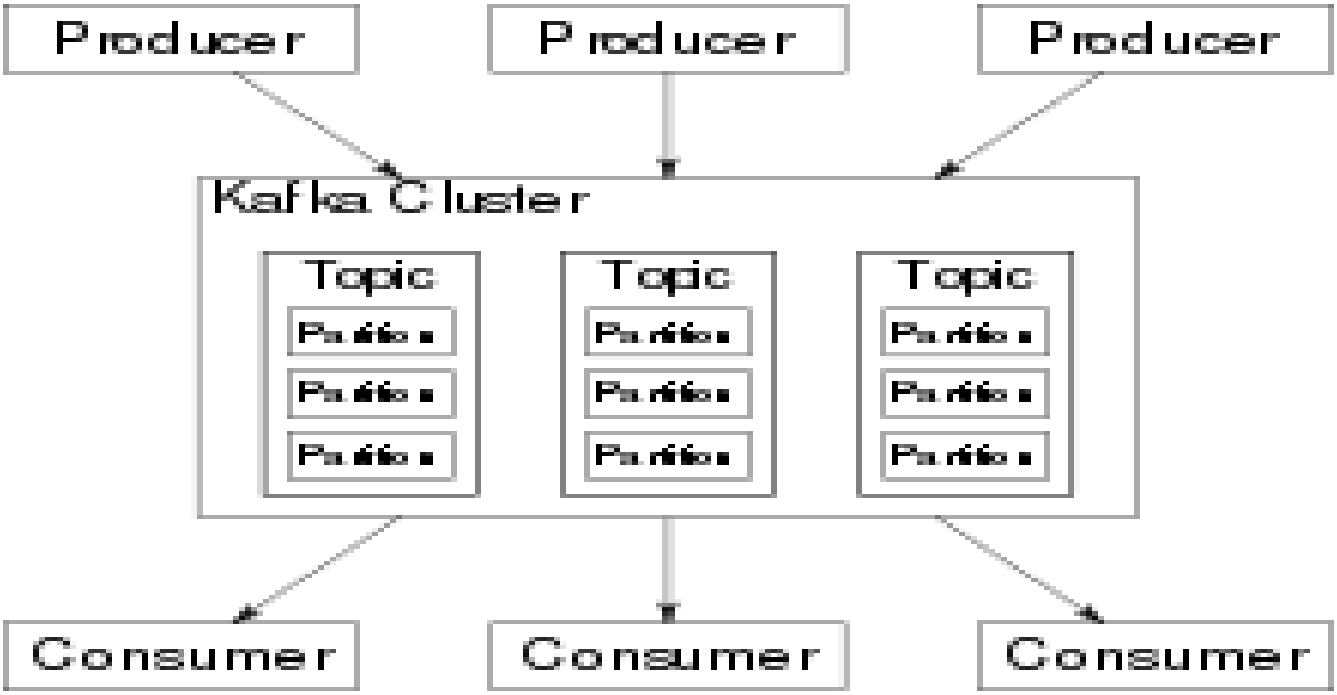
Sno	Date	Modification	Author	Verified By
1	2019/07/31	Initial Document	Nishtha	Sumit Goyal

Table of Contents

Pykafka Integraion with Python/flask

PyKafka is a programmer-friendly Kafka client for Python. It includes Python implementations of Kafka producers and consumers, which are optionally backed by a C extension, built on librdkafka.

PyKafka's primary goal is to provide a similar level of abstraction to the JVM Kafka client using idioms familiar to Python programmers and exposing the most Pythonic API possible..



.

Business Requirement

The main objective of this project is that to build a live map of London with real-time updates. We will use apache kafka, javascript and python(flask Pykafk and json)

Solutions:

Note: In this document we explained step by step integration between Python/flask and kafka (to show live map) using pykafka.

Steps :

Download java

We can download java from below URL-

Link: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

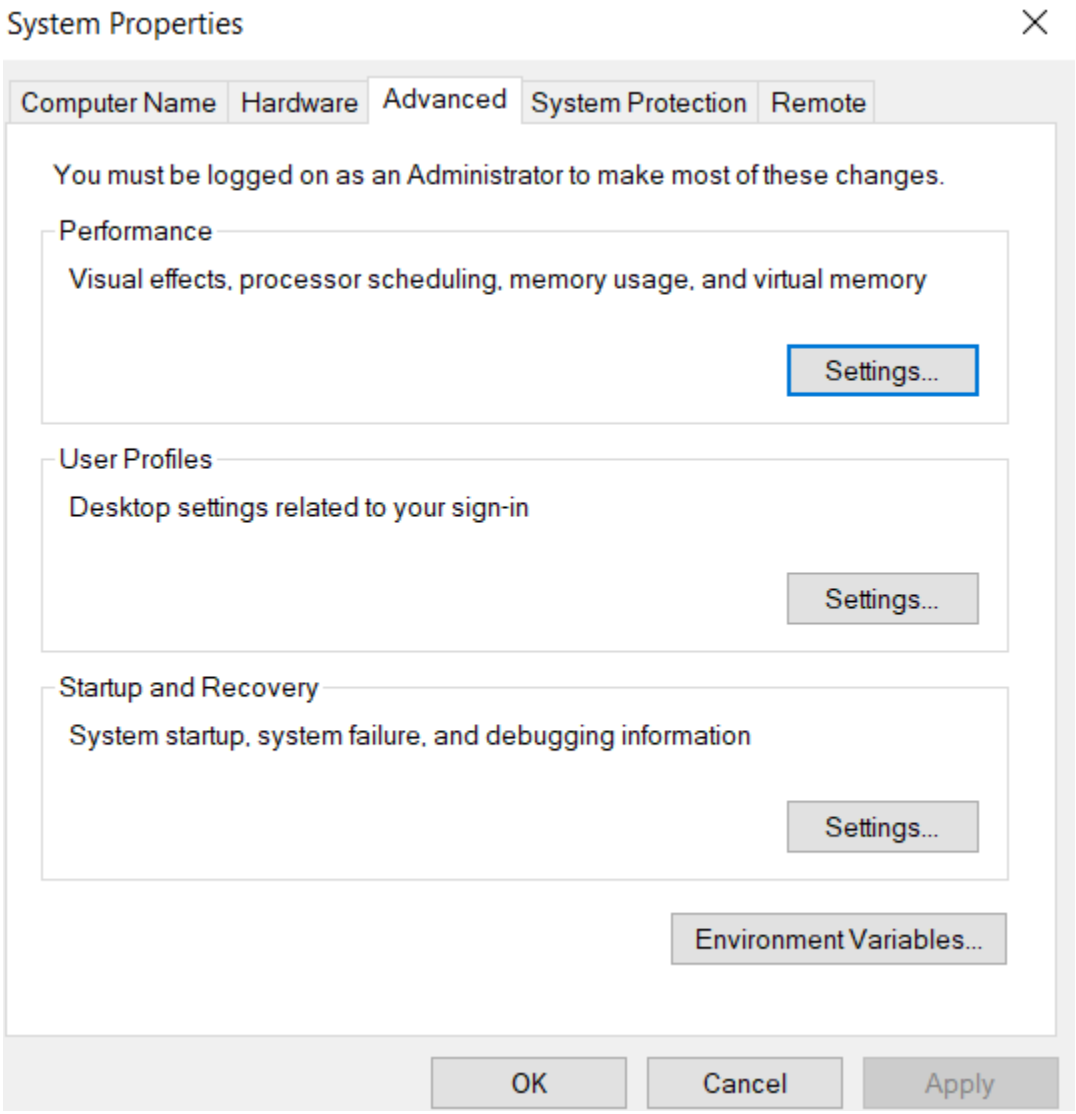
Java SE Development Kit 8u221

You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software.
Thank you for accepting the Oracle Technology Network License Agreement for Oracle Java SE; you may now download this software.

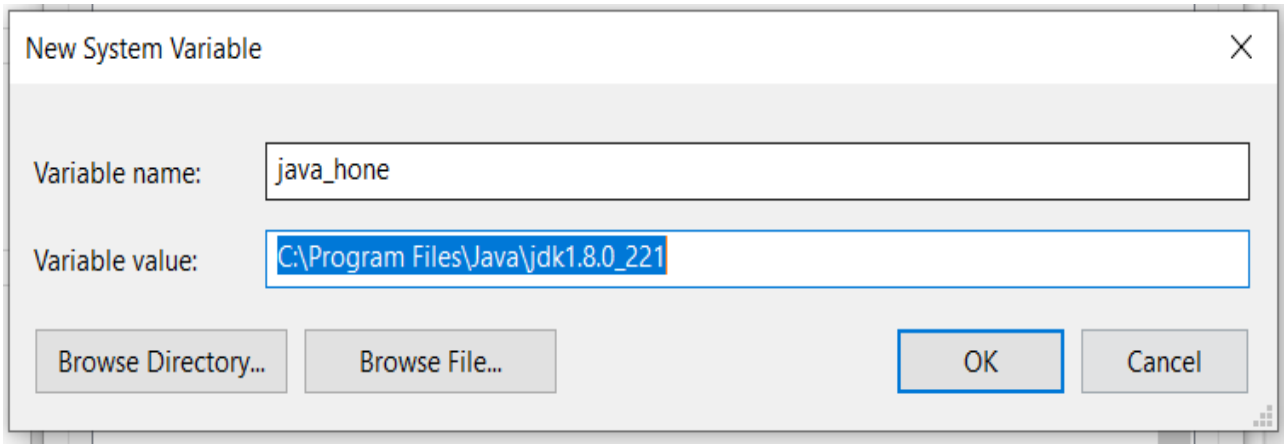
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.9 MB	jdk-8u221-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.81 MB	jdk-8u221-linux-arm64-vfp-hflt.tar.gz
Linux x86	174.18 MB	jdk-8u221-linux-i586.rpm
Linux x86	189.03 MB	jdk-8u221-linux-i586.tar.gz
Linux x64	171.19 MB	jdk-8u221-linux-x64.rpm
Linux x64	186.06 MB	jdk-8u221-linux-x64.tar.gz
Mac OS X x64	252.52 MB	jdk-8u221-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	132.99 MB	jdk-8u221-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.23 MB	jdk-8u221-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	133.66 MB	jdk-8u221-solaris-x64.tar.Z
Solaris x64	91.95 MB	jdk-8u221-solaris-x64.tar.gz
Windows x86	202.73 MB	jdk-8u221-windows-i586.exe
Windows x64	215.35 MB	jdk-8u221-windows-x64.exe

Once java is downloaded need to install it..and set the path in environment variable that is present in advance system settings.

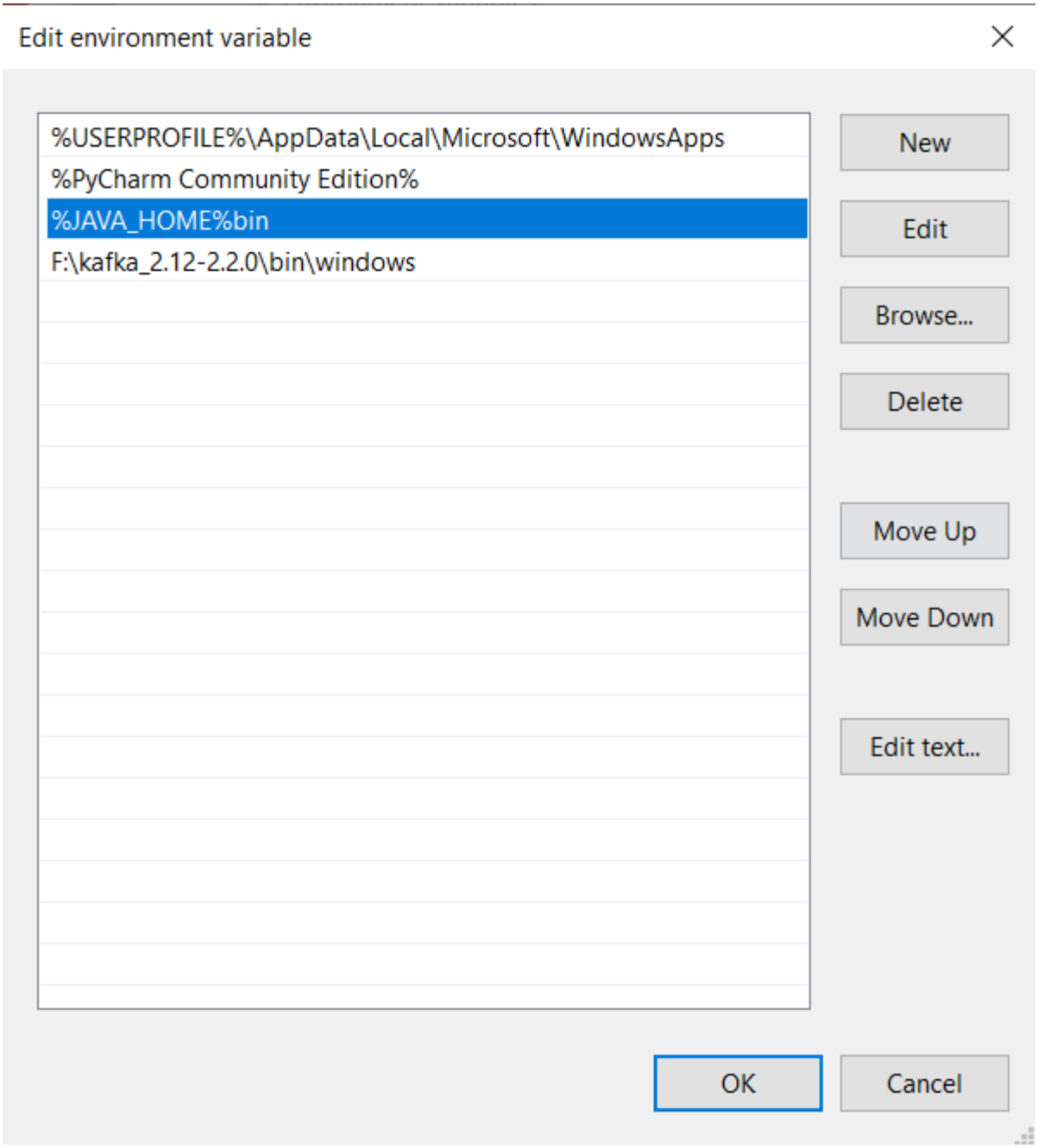
a) Click on advance system settings.
b) Click on environment variables.



Click on environment variable, and then click on system variable and set the path of java



And in user variable click on path then edit button and set the java path.



Once the path is set now you can check the java version by enter the cmd

>> java – version in command prompt.

2) Download apache kafka by clicking on the below url.

https://www.apache.org/dyn/closer.cgi?path=/kafka/2.2.0/kafka_2.12-2.2.0.tgz

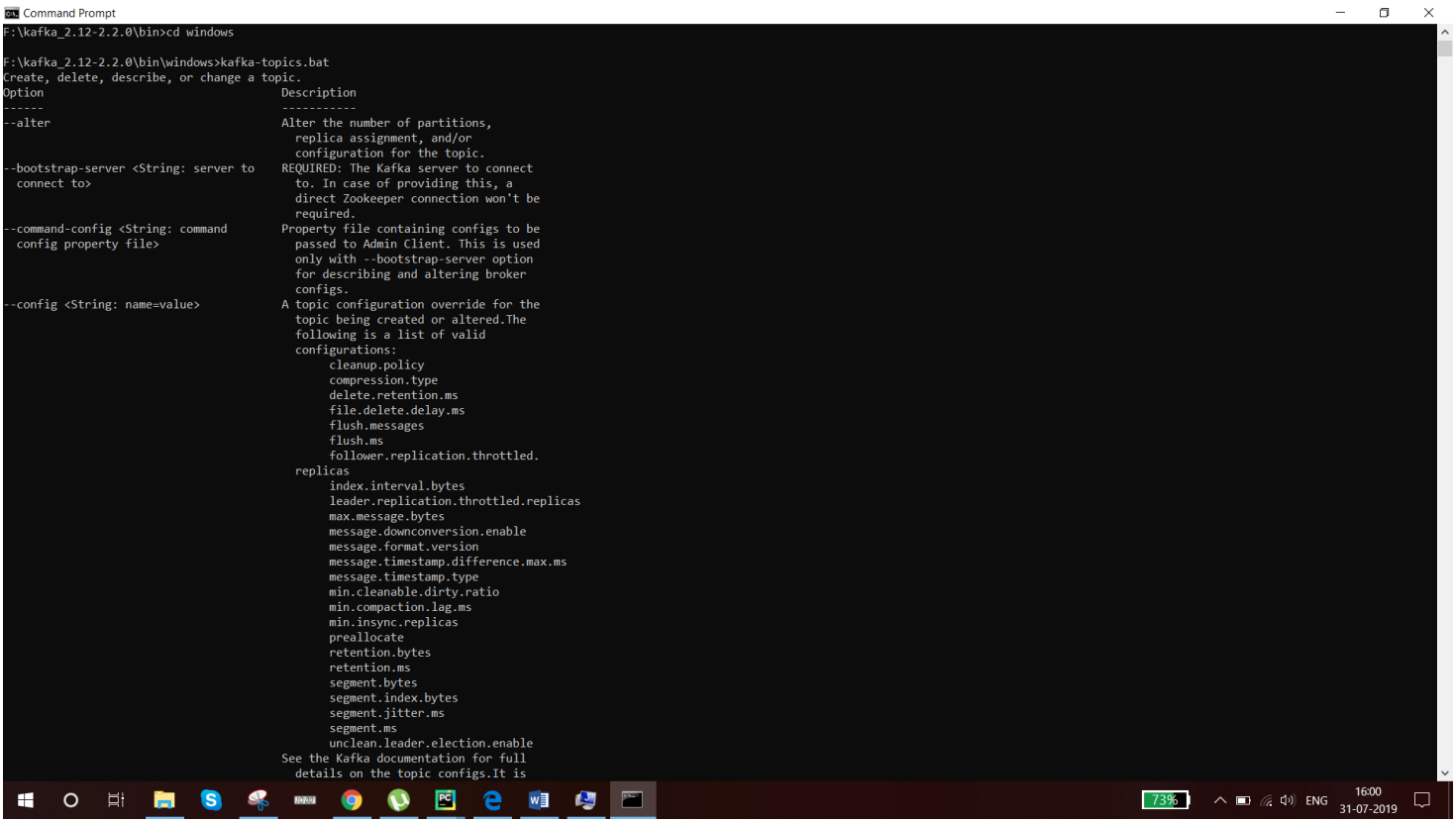
Once kafka has downloaded, unzip it

Open the command prompt and go to the directory where you unzip kafka folder

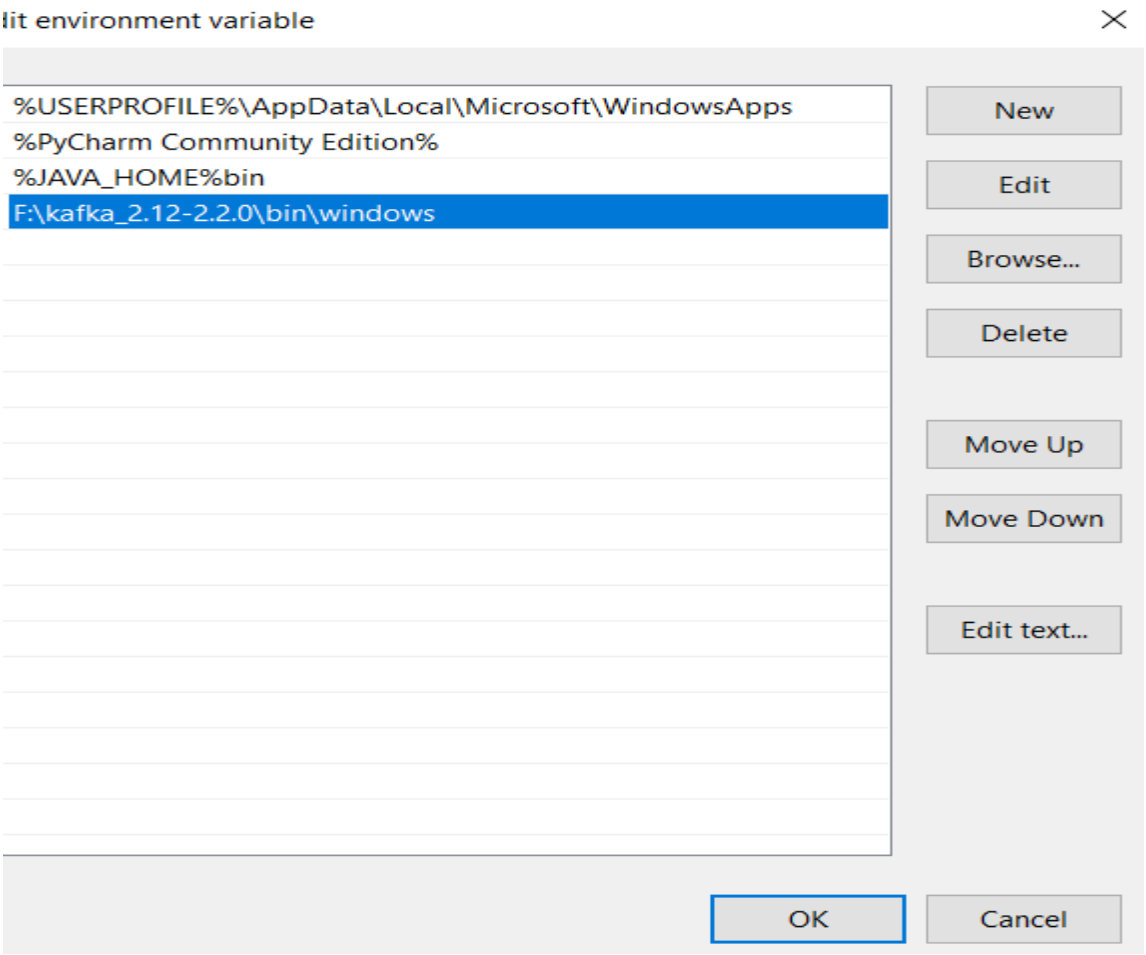
And hit the cmd

>>f:\kafka_2.12-2.2.0\bin\windows>kafka-topics.bat

Hit enter, if you get something like this.



Congrats you have successfully installed kafka in your windows.



We need to set the kafka path in enviroment variable for proper use of kafka server

How to start kafka in windows:

Step1: Go to the directory where kafka is installed.

Step2: make a folder called data

Step3:under data again create two folder kafka and the another one is zookeeper..we need to this folder for storing logs

Step4: we need to modify the zookeeper path n zookeeper.py file

F:\kafka_2.12-2.2.0\config under this directory

```
dataDir=F:/kafka_2.12-2.2.0/data/zookeeper
# the port at which the clients will connect
clientPort=2181
# disable the per-ip limit on the number of connections since this is a non-production config
maxClientCnxns=0
```

Step5: need to modify the kafka server path

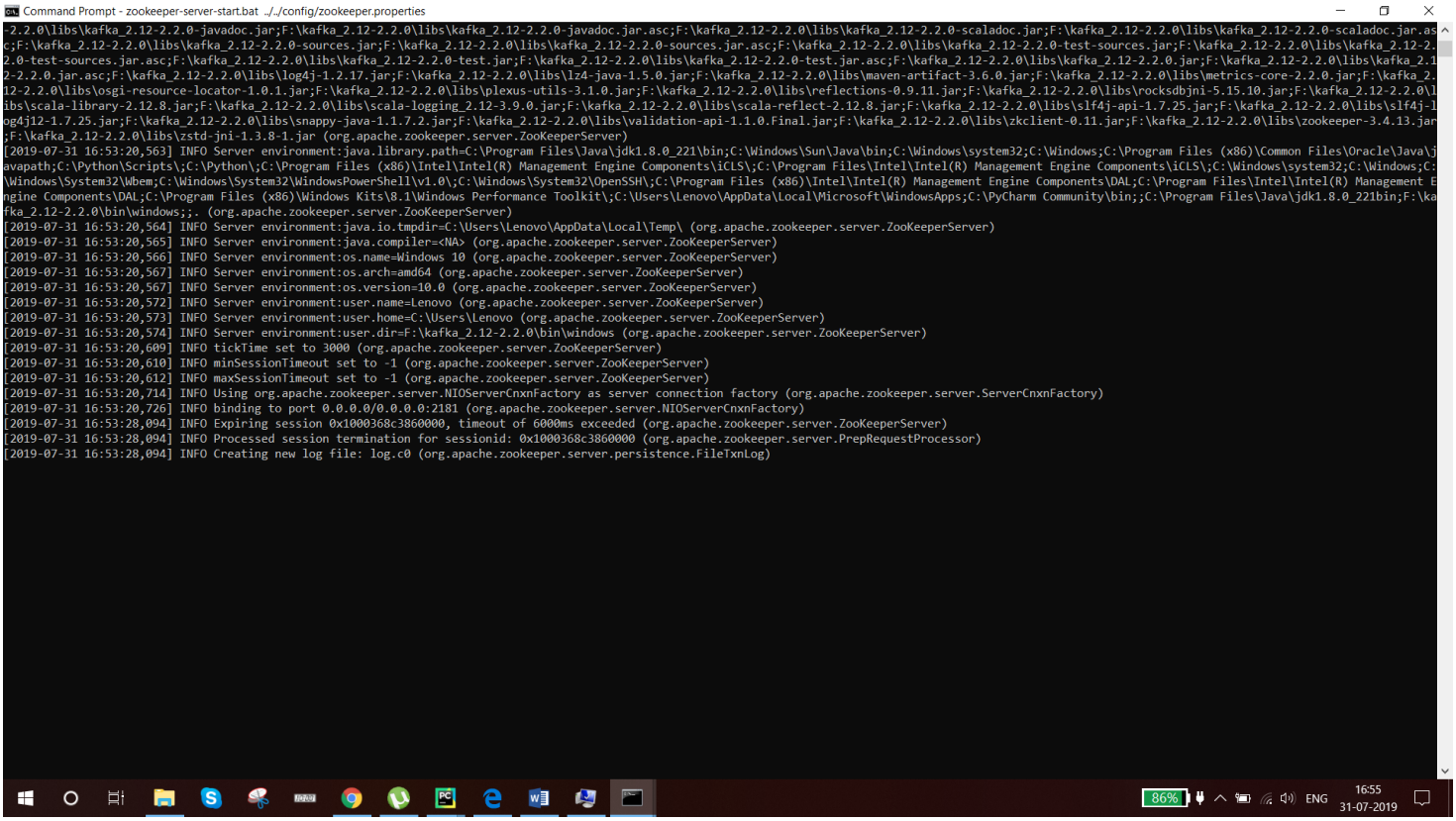
F:\kafka_2.12-2.2.0\config under server properties.

```
############################# Log Basics #############################

# A comma separated list of directories under which to store log files
log.dirs=F:/kafka_2.12-2.2.0/data/kafka

# The default number of log partitions per topic. More partitions allow greater
# parallelism for consumption, but this will also result in more files across
# the brokers.
num.partitions=1
zookeeper.connect=0.0.0.0:2181
```

Step6: need to start zookeeper server by entering the following cmd in cmd prompt

>>F:\kafka_2.12-2.2.0\bin\windows>zookeeper-server-start.bat ../../config/zookeeper.properties

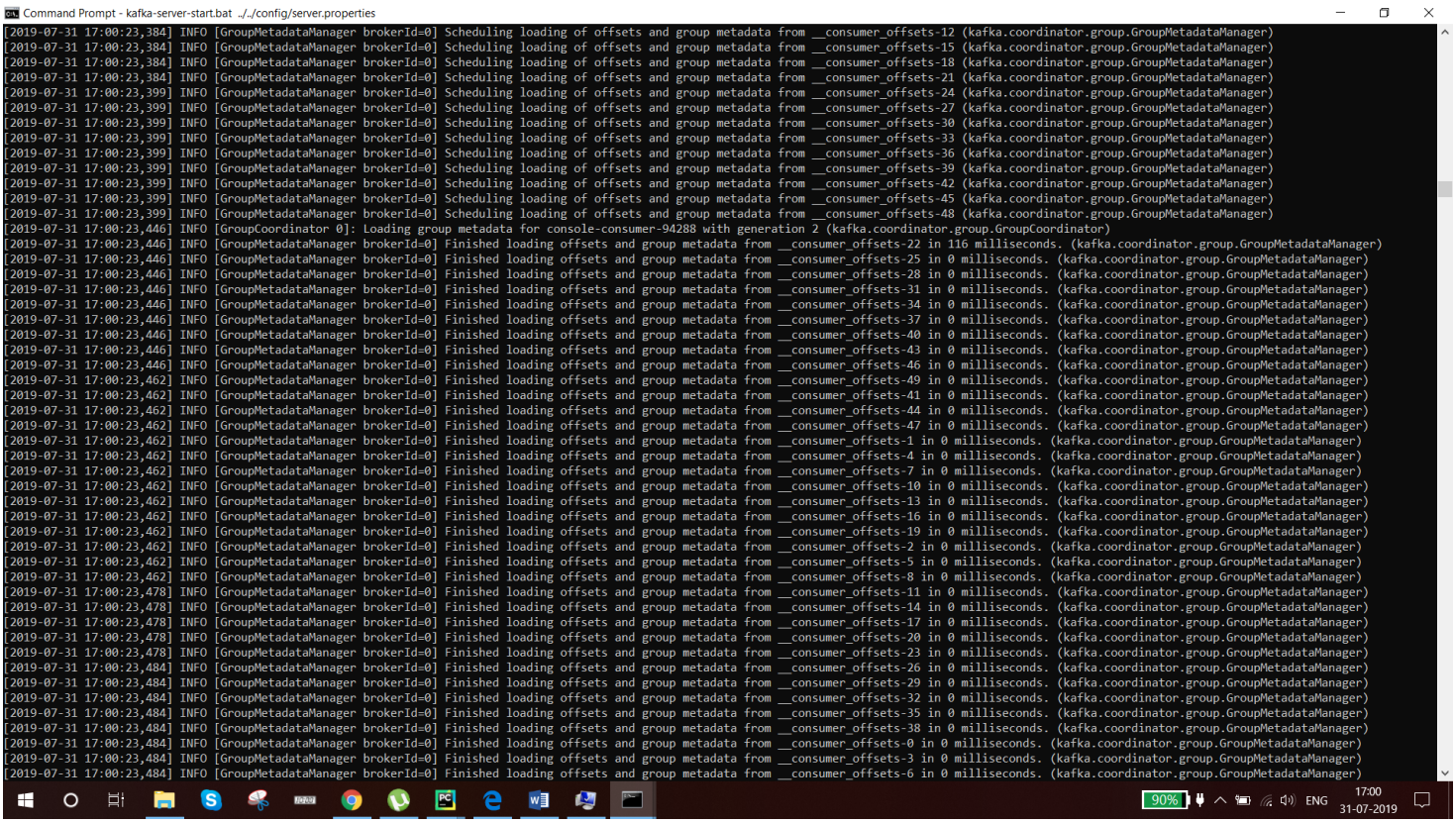
If you see the following screen then your zookeeper server is up to running



Step7: Need to start kafka server by entering the following cmd in cmd prompt

>>F:\kafka_2.12-2.2.0\bin\windows>kafka-server-start.bat ../../config/server.properties.

If you see the following screen then your kafka server is up to running.



Step8: Need to create a topic by run the following cmd in cmd prompt

>>F:\kafka_2.12-2.2.0\bin\windows>kafka-topics.bat --zookeeper 0.0.0.0:2181 --topic test_topic --create --partitions 1 -- replication-factor 1

Created topic test topic

Step9: start producer by entering the following cmd.

>>F:\kafka_2.12-2.2.0\bin\windows>kafka-console-producer.bat --broker-list localhost:9092 --topic test_topic

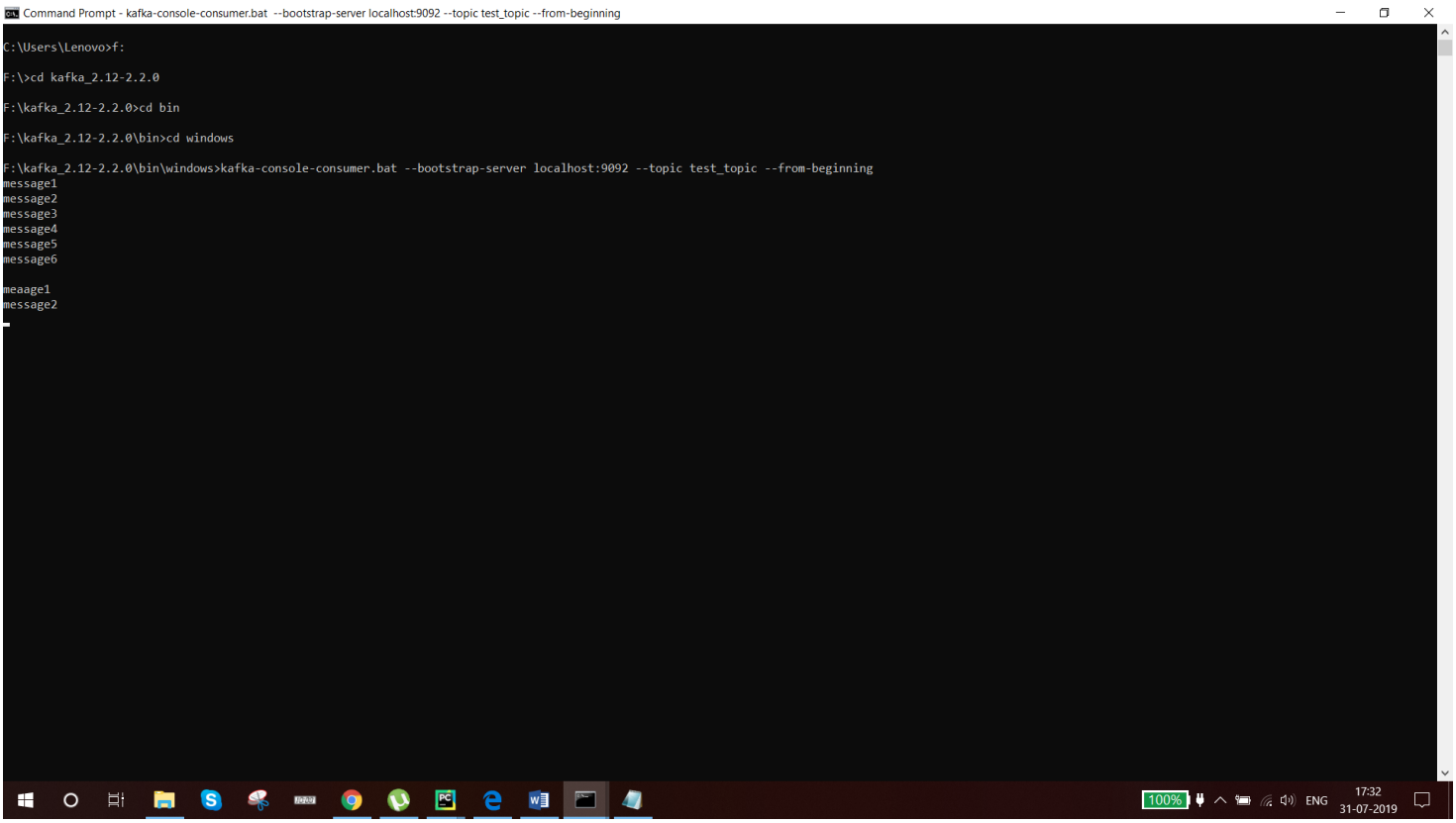
>message1

>message2

>message3

Step10: Start a consumer by entering the following cmd.

>>F:\kafka_2.12-2.2.0\bin\windows>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test_topic --from-beginning



Connect to flask:

>>pip install pykafka

Here we are generating live bus location

So we need to generate a map api by hit the below url

https://account.mapbox.com/

once you successfully logged in. you see a access token in your map box dashboard copy it

busdata1.py:

```
from pykafka import KafkaClient
import json
from datetime import datetime
import uuid
import time

#READ COORDINATES FROM GEOJSON
input_file = open('bus1.json')
json_array = json.load(input_file)
coordinates = json_array['features'][0]['geometry']['coordinates']

#GENERATE UUID
def generate_uuid():
    return uuid.uuid4()
```

```
#KAFKA PRODUCER
client = KafkaClient(hosts="localhost:9092")
topic = client.topics['geodata_final123']
producer = topic.get_sync_producer()

#CONSTRUCT MESSAGE AND SEND IT TO KAFKA
data = {}
data['busline'] = '00001'

def generate_checkpoint(coordinates):
    i = 0
    while i < len(coordinates):
        data['key'] = data['busline'] + '_' + str(generate_uuid())
        data['timestamp'] = str(datetime.utcnow())
        data['latitude'] = coordinates[i][1]
        data['longitude'] = coordinates[i][0]
        message = json.dumps(data)
        print(message)
        producer.produce(message.encode('ascii'))
        time.sleep(1)

        #if bus reaches last coordinate, start from beginning
        if i == len(coordinates)-1:
            i = 0
        else:
            i += 1

generate_checkpoint(coordinates)
```

index.html:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <!-- LEAFLET -->
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.4.0/dist/leaflet.css"
      integrity="sha512-
puBpdR0798OZvTTbP4A8Ix/l+A4dHDD0DGqYW6RQ+9jxkRFclaxxQb/SJAWZfWAkuyeQUytO7+7N4QKrDh+drA=="
        crossorigin=""/>
    <script src="https://unpkg.com/leaflet@1.4.0/dist/leaflet.js"
      integrity="sha512-
QVftwZFqvtRNi0ZyCtsznlKSWOStnDORoefr1enyq5mVL4tmKB3S/EnC3rRJcxCPavG10IcrVGSmPh6Qw5lwrg=="
      crossorigin=""></script>
    <!-- END LEAFLET -->
    <title>London Live Map</title>
  </head>
  <body>
    <h1>London Bus Live Map</h1>

    <!-- LEAFLET -->
    <div id="mapid" style = "width:900px; height:580px;"></div>
    <script src="../static/leaf.js"></script>
    <!-- END LEAFLET -->
  </body>
</html>
```

Leaf.js:

```
var mymap = L.map('mapid').setView([51.505, -0.09], 13);
L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_token={accessToken}', {
    attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a>
contributors, <a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, Imagery © <a
href="https://www.mapbox.com/">Mapbox</a>',
    maxZoom: 18,
    id: 'mapbox.streets',
    accessToken:
'pk.eyJ1IjoibmlzaHRoYTAzIiwiYSI6ImNqeXBuZGxtdzBhdXczbm9mMHkyMHc4cGEifQ.LMOdABNaOv2m-phFTKjtUQ'
//ENTER YOUR ACCESS TOKEN HERE
}).addTo(mymap);

mapMarkers1 = [];
mapMarkers2 = [];
mapMarkers3 = [];

var source = new EventSource('/topic/TOPICNAME'); //ENTER YOUR TOPICNAME HERE
source.addEventListener('message', function(e){

  console.log('Message');
  obj = JSON.parse(e.data);
  console.log(obj);

  if(obj.busline == '00001') {
    for (var i = 0; i < mapMarkers1.length; i++) {
      mymap.removeLayer(mapMarkers1[i]);
    }
    marker1 = L.marker([obj.latitude, obj.longitude]).addTo(mymap);
    mapMarkers1.push(marker1);
  }

  if(obj.busline == '00002') {
    for (var i = 0; i < mapMarkers2.length; i++) {
      mymap.removeLayer(mapMarkers2[i]);
    }
    marker2 = L.marker([obj.latitude, obj.longitude]).addTo(mymap);
    mapMarkers2.push(marker2);
  }

  if(obj.busline == '00003') {
    for (var i = 0; i < mapMarkers3.length; i++) {
      mymap.removeLayer(mapMarkers3[i]);
    }
    marker3 = L.marker([obj.latitude, obj.longitude]).addTo(mymap);
    mapMarkers3.push(marker3);
  }
}, false);
```

final output: