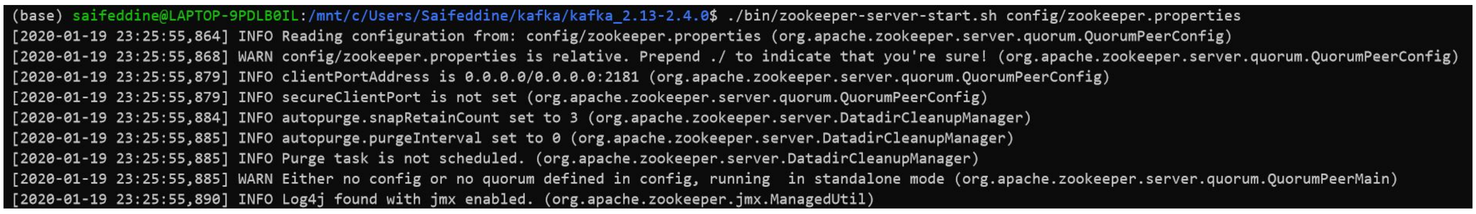
Saif Eddine Ghribi
saif-eddine.ghribi@polytechnique.edu

Lab session # 1
Data Stream

01/21/20

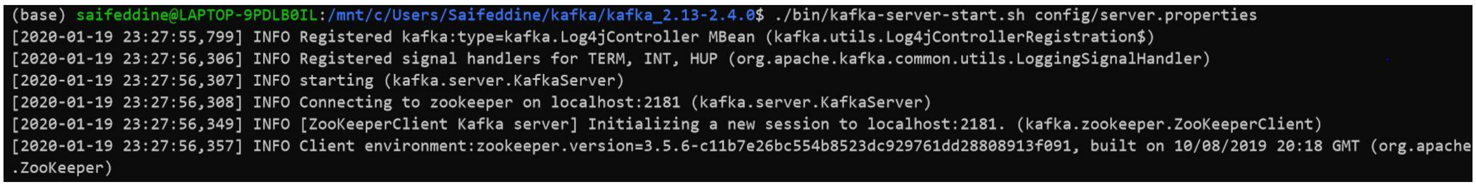
1 Installing and Running Kafka

After downloading the version kafka_2.13-2.4.0,installing it on Linux was simply done by extracting the archive of installation and executing the proper commands to start kafka. In order to start kafka we launch the Zookeeper and a kafka server as shown in Fig1 and Fig2.



```
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/Saifeddine/kafka/kafka_2.13-2.4.0$ ./bin/zookeeper-server-start.sh config/zookeeper.properties
[2020-01-19 23:25:55,864] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2020-01-19 23:25:55,868] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2020-01-19 23:25:55,879] INFO clientPortAddress is 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2020-01-19 23:25:55,879] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2020-01-19 23:25:55,884] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2020-01-19 23:25:55,885] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2020-01-19 23:25:55,885] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2020-01-19 23:25:55,885] WARN Either no config or no quorum defined in config, running  in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2020-01-19 23:25:55,890] INFO Log4j found with jmx enabled. (org.apache.zookeeper.jmx.ManagedUtil)
```

Figure 1: Zookeeper Component launched



```
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/Saifeddine/kafka/kafka_2.13-2.4.0$ ./bin/kafka-server-start.sh config/server.properties
[2020-01-19 23:27:55,799] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2020-01-19 23:27:56,306] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)
[2020-01-19 23:27:56,307] INFO starting (kafka.server.KafkaServer)
[2020-01-19 23:27:56,308] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2020-01-19 23:27:56,349] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2020-01-19 23:27:56,357] INFO Client environment:zookeeper.version=3.5.6-c11b7e26bc554b8523dc929761dd28808913f091, built on 10/08/2019 20:18 GMT (org.apache
.ZooKeeper)
```

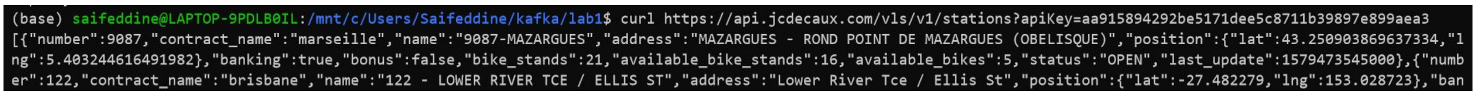
Figure 2: Kafka server launched

We notice that they have successfully been launched thanks to the logs returned in the terminal.

2 Velib use case

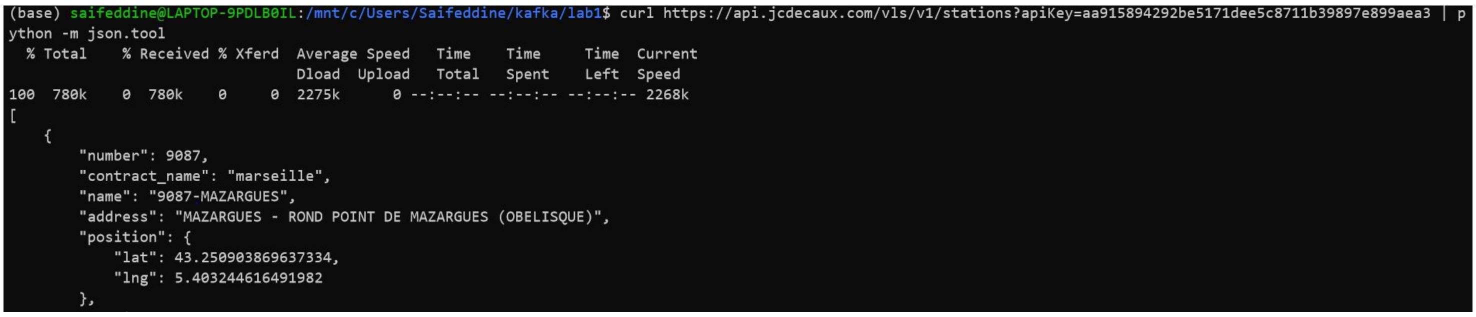
2.1 Retrieving the list of all the stations

After creating an account in https://developer.jcdecaux.com and after requesting an API key in order to grant access to the list of stations, we were able to run the command "curl" followed by the link to curl that allowed us to see all the list of the stations.



```
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/Saifeddine/kafka/lab1$ curl https://api.jcdecaux.com/vls/v1/stations?apiKey=aa915894292be5171dee5c8711b39897e899aea3
[{"number":9087,"contract_name":"marseille","name":"9087-MAZARGUES","address":"MAZARGUES - ROND POINT DE MAZARGUES (OBELISQUE)","position":{"lat":43.250903869637334,"l
ng":5.403244616491982},"banking":true,"bonus":false,"bike_stands":21,"available_bike_stands":16,"available_bikes":5,"status":"OPEN","last_update":1579473545000},{"numb
er":122,"contract_name":"brisbane","name":"122 - LOWER RIVER TCE / ELLIS ST","address":"Lower River Tce / Ellis St","position":{"lat":-27.482279,"lng":153.028723},"ban
```

Figure 3: The list of bike stations



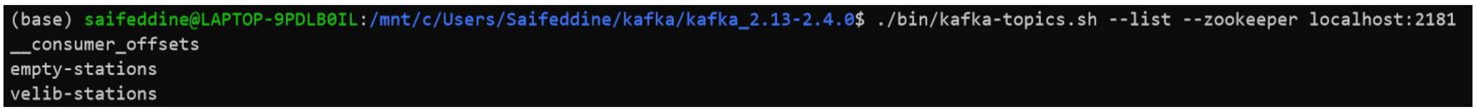
```
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/Saifeddine/kafka/lab1$ curl https://api.jcdecaux.com/vls/v1/stations?apiKey=aa915894292be5171dee5c8711b39897e899aea3 | p
ython -m json.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  780k    0  780k    0     0  2275k      0 --:--:-- --:--:-- --:--:-- 2268k
[
    {
        "number": 9087,
        "contract_name": "marseille",
        "name": "9087-MAZARGUES",
        "address": "MAZARGUES - ROND POINT DE MAZARGUES (OBELISQUE)",
        "position": {
            "lat": 43.250903869637334,
            "lng": 5.403244616491982
        },
```

Figure 4: The list of bike stations with a JSON "Prettifier"

We can see from what the terminal returns (Fig3) that the type of is json so a further operation which is a "Json" Prettifier would make the list easier to read.

2.2 Create a topic

We can create a topic thanks to the command ./bin/kafka-topics.sh and with passing the argument –create we are specifying that we would like to create a topic the other arguments are to specify the properties of the topic to be created (like its name).
We can also show the list of all the topic with the same command but this time we pass the argument –list as shown in Fig5.



```
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/Saifeddine/kafka/kafka_2.13-2.4.0$ ./bin/kafka-topics.sh --list --zookeeper localhost:2181
__consumer_offsets
empty-stations
velib-stations
```

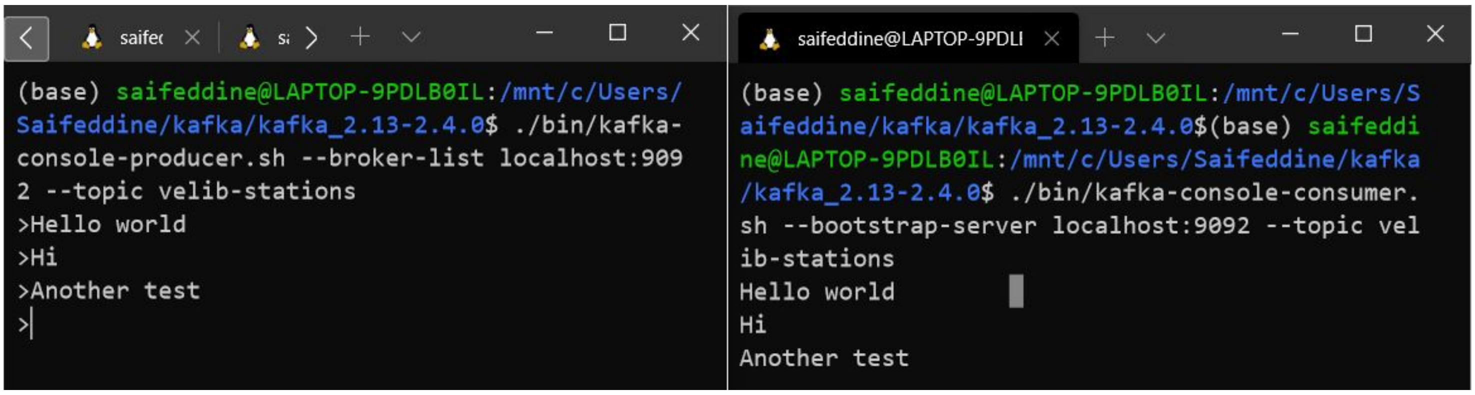
Figure 5: List of topics

2.3 Producer and consumer

Kafka has two inbuilt applications

- The producer(./bin/kafka-console-producer.sh) that produces messages(takes messages as input in the command line) and send them ti the kafka cluster
- The consumer(./bin/kafka-console-consumer.sh) in another hand "consumes" the messages sent by the producer and show them

To properly execute the commands of the producer and the consumer we need to pass as argument the address of the bootstrap server(- -bootstrap-server) and the topic (- -topic) .Fig6 shows an example of communication between the producer and the consumer.



```
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/
Saifeddine/kafka/kafka_2.13-2.4.0$ ./bin/kafka-
console-producer.sh --broker-list localhost:909
2 --topic velib-stations
>Hello world
>Hi
>Another test
>
```

```
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/S
aifeddine/kafka/kafka_2.13-2.4.0$(base) saifeddi
ne@LAPTOP-9PDLB0IL:/mnt/c/Users/Saifeddine/kafka
/kafka_2.13-2.4.0$ ./bin/kafka-console-consumer.
sh --bootstrap-server localhost:9092 --topic vel
ib-stations
Hello world
Hi
Another test
```

Figure 6: Producer and consumer

2.4 Velib Application

- velib-get-stations.py:In this script we create a producer that uses the Velib API and every stations received in the response will be sent to the kafka topic Velib station.

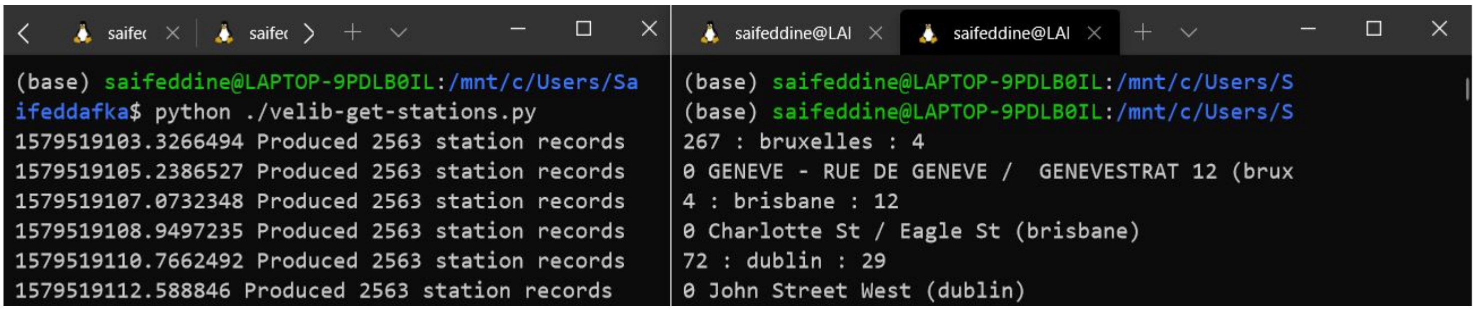
```python
import json
import time
import urllib.request
from kafka import KafkaProducer
from urllib.request import urlopen

API_KEY = "aa915894292be5171dee5c8711b39897e899aea3" # FIXME Set your own API key here
url = "https://api.jcdecaux.com/vls/v1/stations?apiKey={}".format(API_KEY)
producer = KafkaProducer(bootstrap_servers="localhost:9092")
while True:
    response = urlopen(url)
    stations = json.loads(response.read().decode('utf-8'))
    for station in stations:
        producer.send("velib-stations", json.dumps(station).encode())
    print("{} Produced {} station records".format(time.time(), len(stations)))
    time.sleep(1)
```

Listing 1: velib-get-stations.py

- velib-monitor-stations.py : In this script a consumer will store the status of the different stations and returns a message when a station changes its state.

```python
import json
from kafka import KafkaConsumer
stations = {}
consumer = KafkaConsumer("velib-stations", bootstrap_servers='localhost:9092',group_id="velib-monitor-
    stations")
for message in consumer:
    station = json.loads(message.value.decode())
    station_number = station["number"]
    contract = station["contract_name"]
    available_bike_stands = station["available_bike_stands"]
    print("{} : {} : {} ".format(station_number, contract, available_bike_stands))
    if contract not in stations:
        stations[contract] = {}
    city_stations = stations[contract]
    if station_number not in city_stations:
        city_stations[station_number] = available_bike_stands
    count_diff = available_bike_stands - city_stations[station_number]
    if count_diff != 0:
        city_stations[station_number] = available_bike_stands
    print("{}{} {} ({})".format(
    "+" if count_diff > 0 else "",
    count_diff, station["address"], contract
    ))
```

Listing 2: velib-monitor-stations.py



```
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/Sa
ifeddafka$ python ./velib-get-stations.py
1579519103.3266494 Produced 2563 station records
1579519105.2386527 Produced 2563 station records
1579519107.0732348 Produced 2563 station records
1579519108.9497235 Produced 2563 station records
1579519110.7662492 Produced 2563 station records
1579519112.588846 Produced 2563 station records
```

```
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/S
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/S
267 : bruxelles : 4
0 GENEVE - RUE DE GENEVE /  GENEVESTRAT 12 (brux
4 : brisbane : 12
0 Charlotte St / Eagle St (brisbane)
72 : dublin : 29
0 John Street West (dublin)
```

Figure 7: Velib application

After running the two scripts ,the application is working and is giving us information about the stations that are changing its state as shown in Fig7.

3 Practice Exercises

3.1 Create Topic

To create the topic empty-stations in our Kafka cluster , we can use the follwing command.

```

./bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic
empty-stations
```

Listing 3: Create empty-stations topic.py

3.2 Create a producer get-stations.py

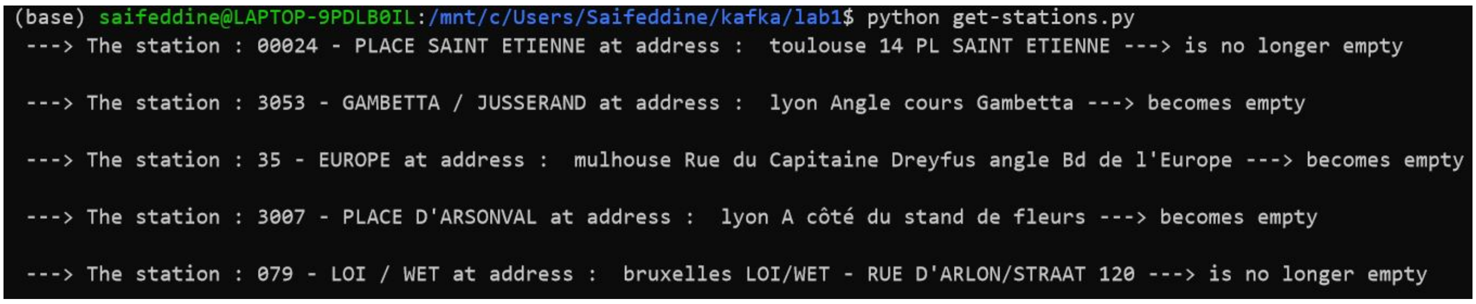
To create a producer get-stations.py as described ,we suggest the following code :

```python
import json
import time
import urllib.request

# import the producer
from kafka import KafkaProducer


API_KEY = "aa915894292be5171dee5c8711b39897e899aea3"
url = "https://api.jcdecaux.com/vls/v1/stations?apiKey={}".format(API_KEY)

# Create the prodcuer
producer = KafkaProducer(bootstrap_servers="localhost:9092")

# initiate the dicionnary that will store the bikes number of stations
```

```python
Bikes_number = {}

while True:
    # get response
    response = urllib.request.urlopen(url)
    stations = json.loads(response.read().decode())

    for station in stations:
        # retrieve the station bike number
        station_bike_number = station['available_bikes']
        # get the station number and name
        key = "{},{}".format(station["number"], station["contract_name"])
        # add stations to the Bikes_number dictionnary
        if key not in Bikes_number:
            Bikes_number[key] = station_bike_number
        # get station informations
        s_name, s_address, s_city = station["name"], station["address"], station["contract_name"]

        # if the station becomes empty
        if (station_bike_number == 0 and Bikes_number[key] > 0):
            print(" ---> The station : {} at address :  {} {} ---> becomes empty".format(
                s_name, s_city, s_address))
            producer.send("empty-stations",
                          json.dumps(station).encode())
            print()
        # if the station is no longer empty
        if (station_bike_number > 0 and Bikes_number[key] == 0):
            print(" ---> The station : {} at address :  {} {} ---> is no longer empty".format(
                s_name, s_city, s_address))
            producer.send("empty-stations",
                          json.dumps(station).encode())
            print()

        # update the number of bikes
        Bikes_number[key] = station_bike_number

    time.sleep(1)
```

Listing 4: get-stations.py

The result of the execution of the previous script is as shown in Fig8



```
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/Saifeddine/kafka/lab1$ python get-stations.py
 ---> The station : 00024 - PLACE SAINT ETIENNE at address :  toulouse 14 PL SAINT ETIENNE ---> is no longer empty

 ---> The station : 3053 - GAMBETTA / JUSSERAND at address :  lyon Angle cours Gambetta ---> becomes empty

 ---> The station : 35 - EUROPE at address :  mulhouse Rue du Capitaine Dreyfus angle Bd de l'Europe ---> becomes empty

 ---> The station : 3007 - PLACE D'ARSONVAL at address :  lyon A côté du stand de fleurs ---> becomes empty

 ---> The station : 079 - LOI / WET at address :  bruxelles LOI/WET - RUE D'ARLON/STRAAT 120 ---> is no longer empty
```

Figure 8: Velib application

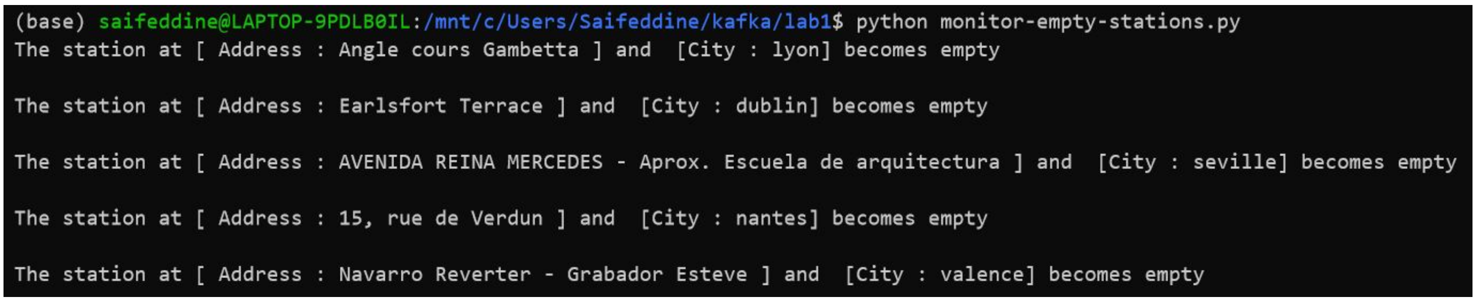
3.3 Create a producer monitor-empty-stations.py

To create a producer Create a monitor-empty-stations.py as described ,we suggest the following code :

```python
import json
from kafka import KafkaConsumer

#create a consumer
consumer = KafkaConsumer("empty-stations", bootstrap_servers='localhost:9092',
                         group_id="velib-monitor-stations")

#process messages from consumer
for message in consumer:
    station = json.loads(message.value.decode())

    #get station information
    current_number_bikes,s_city,s_address  =int(station["available_bikes"]) , station["contract_name"],
    station["address"]

    #if the station becomes empty
    if current_number_bikes == 0:
        print("The station at [ Address : {} ] and  [City : {}] becomes empty ".format(s_address,s_city))
        print('')
```

Listing 5: get-stations.py

The result of the execution of the previous script is as shown in Fig9



```
(base) saifeddine@LAPTOP-9PDLB0IL:/mnt/c/Users/Saifeddine/kafka/lab1$ python monitor-empty-stations.py
The station at [ Address : Angle cours Gambetta ] and  [City : lyon] becomes empty

The station at [ Address : Earlsfort Terrace ] and  [City : dublin] becomes empty

The station at [ Address : AVENIDA REINA MERCEDES - Aprox. Escuela de arquitectura ] and  [City : seville] becomes empty

The station at [ Address : 15, rue de Verdun ] and  [City : nantes] becomes empty

The station at [ Address : Navarro Reverter - Grabador Esteve ] and  [City : valence] becomes empty
```

Figure 9: Velib application