

User and Installation Manual



Table of contents

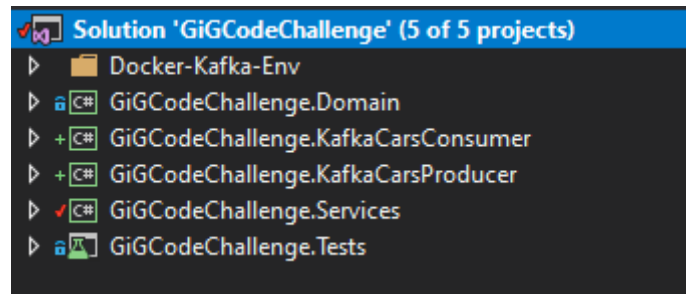
- Introduction 3
- Solution overview. 3
- Task 1 – RESTful API tests..... 4
- Task 2 – Stream processing tests..... 5

Introduction.

This document born to describe the approach used to develop the code challenge and help the people to understand and configure the environment.

Solution overview.

The solution was made with Visual Studio Community 2019, developed in C# with the .Net Core (2.2) SDK. The solution is composed by the following libraries:



Namespace	Description
GiGCodeChallenge.Domain	The library that contains lazy objects that represents the model of the domain.
GiGCodeChallenge.Services	The library that contains the service used to implement logic and dynamics
GiGCodeChallenge.Test	The library that contains the tests developed in specflow.
GiGCodeChallenge.KafkaCarsProducer	A simple console application used to produce messages to send to the Kafka topic.
GiGCodeChallenge. KafkaCarsConsumer	A simple console application used to consume the messages from the Kafka topic.
docker-compose.yml	The file to configure Kafka Env. in docker compose.

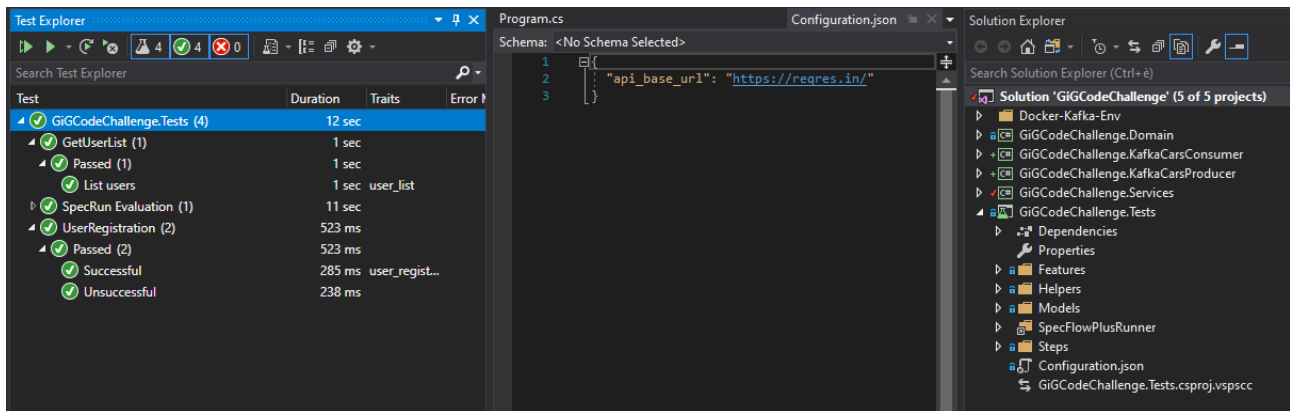
Dependencies

To develop the solution use the following NuGet libraries:

- [Confluent.Kafka 1.2.1](#)
- [Newtonsoft.Json 12.0.2](#)
- [RestSharp 106.6.10](#)
- [SpecFlow 3.0.225](#)
- **SpecFlow.Tools.MsBuild.Generation 3.0.225**
- **SpecRun.SpecFlow 3.0.391**

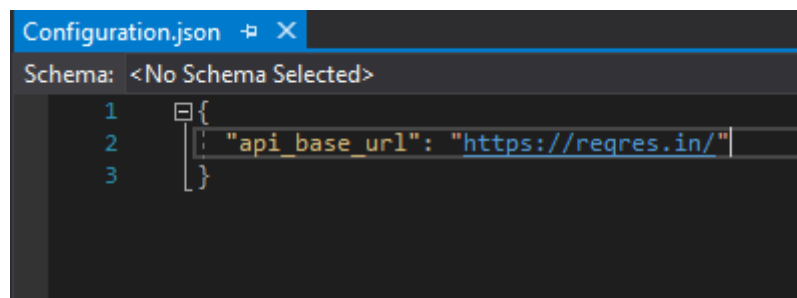
Task 1 – RESTful API tests.

The project was developed in C# using BDD pattern and SpecFlow.



Configuration.

To configure the project is necessary to set the base API URL. It is possible to do this editing the Json File *Configuration.json* as show the following picture.



Execution.

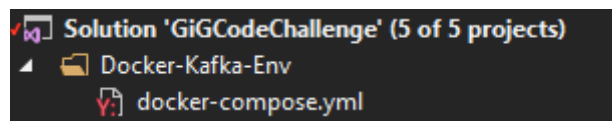
It possible to run the tests directly from Test Explorer.

Task 2 – Stream processing tests.

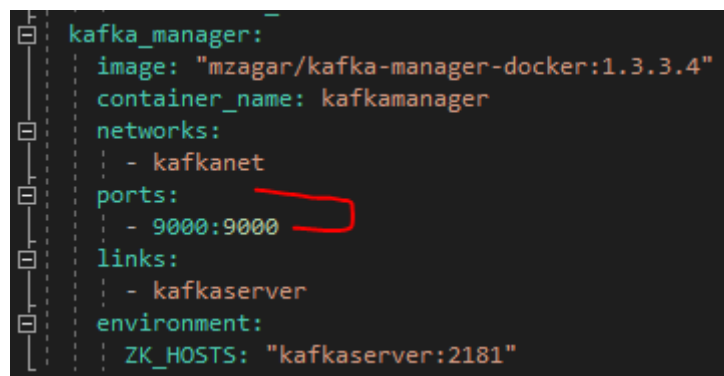
As requested, to develop this task was installed a Docker Toolbox in a local machine. After this was made a Docker container where was in running a Kafka image. In the solution it is possible to find a *docker-compose.yml* file for a quickly setup of the Environment. The Kafka image is [spotify/kafka](https://hub.docker.com/r/spotify/kafka/).

Configuration.

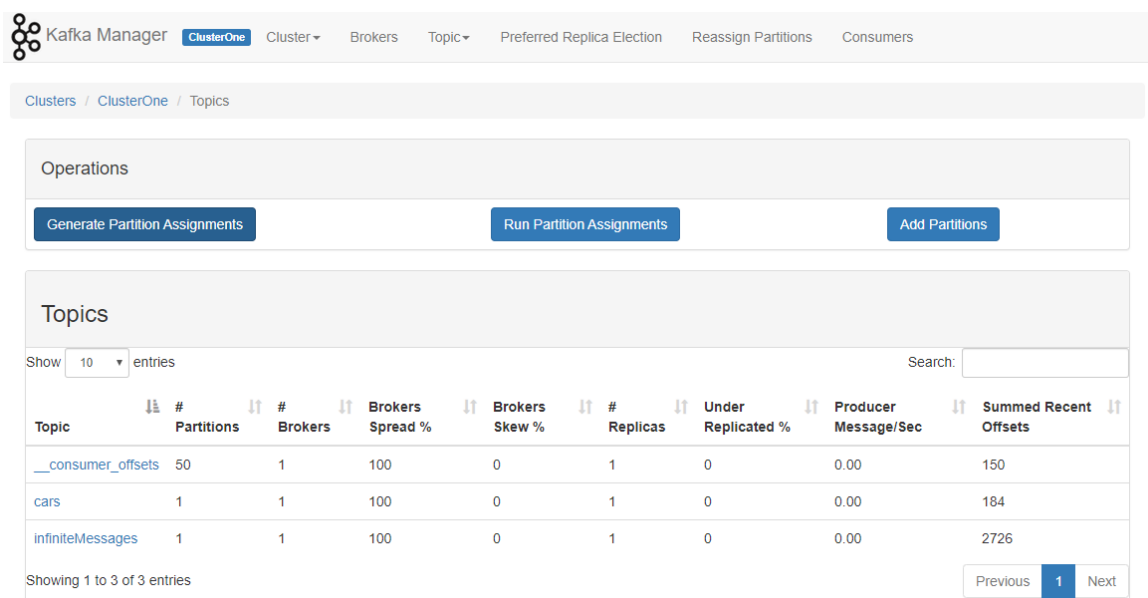
Use the ***docker-compose.yml*** file to run up the Docker container with the Kafka image. The file is stored in the solution tree under the folder Docker-Kafka-Env as show the following picture.



The file set also a Kafka Manager in order to have a web-based interface to allow an easy setup of the Kafka Cluster. It is possible to connect to that interface through the endpoint address on 9000 port. It is also possible to change this default configuration on the ***docker-compose.yml*** file. The following picture show the configuration that we talking about.



The following picture show the Kafka manager interface.

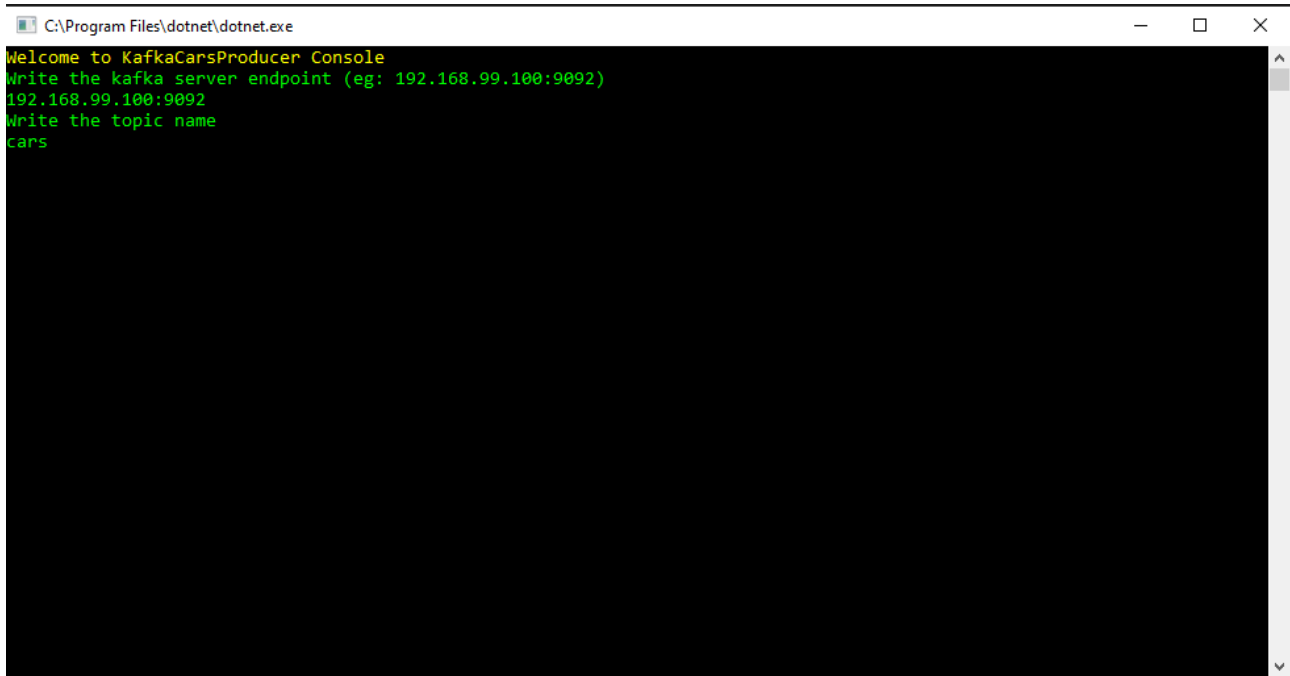


Topic	# Partitions	# Brokers	Brokers Spread %	Brokers Skew %	# Replicas	Under Replicated %	Producer Message/Sec	Summed Recent Offsets
__consumer_offsets	50	1	100	0	1	0	0.00	150
cars	1	1	100	0	1	0	0.00	184
infiniteMessages	1	1	100	0	1	0	0.00	2726

For the specific test was chosen to create only one cluster and a Topic with replica factor equal to one.

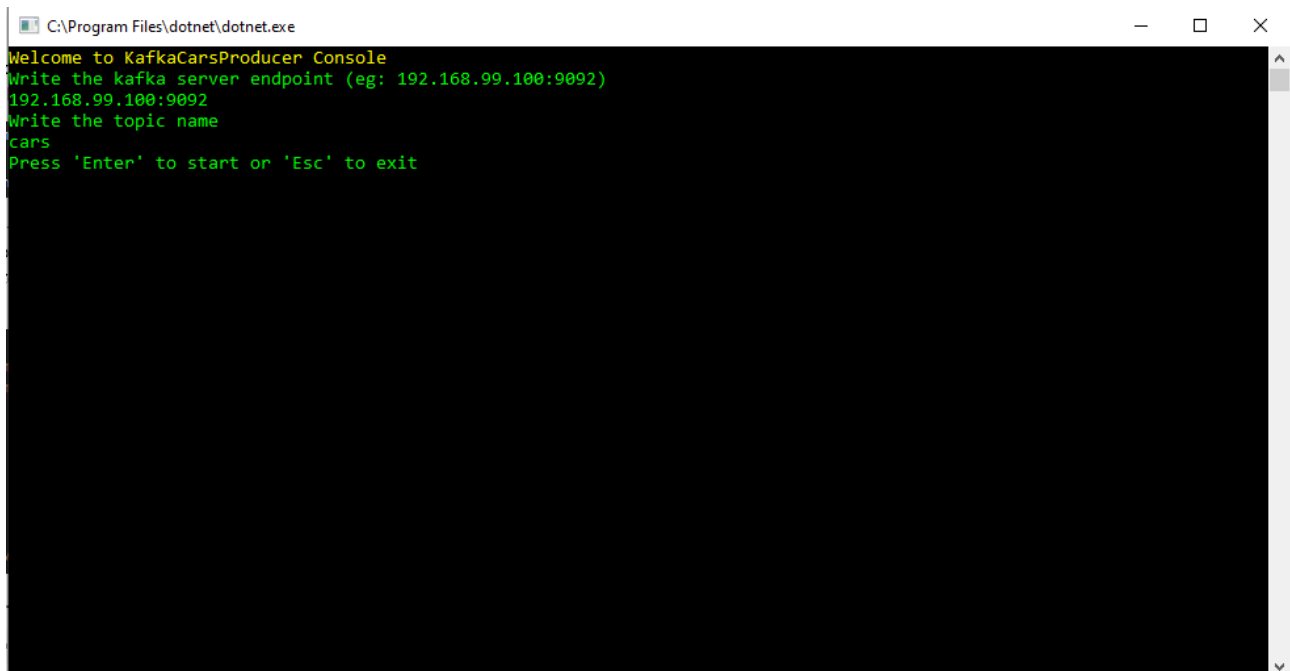
Starting Producer.

The producer console application produces infinite Cars to send to the Kafka Topic. The console asks to user to set the endpoint of the Kafka server and the topic name. The following picture show the first steps of the configuration.



```
C:\Program Files\dotnet\dotnet.exe
Welcome to KafkaCarsProducer Console
Write the kafka server endpoint (eg: 192.168.99.100:9092)
192.168.99.100:9092
Write the topic name
cars
```

After these steps the user can choose to start the process or not.



```
C:\Program Files\dotnet\dotnet.exe
Welcome to KafkaCarsProducer Console
Write the kafka server endpoint (eg: 192.168.99.100:9092)
192.168.99.100:9092
Write the topic name
cars
Press 'Enter' to start or 'Esc' to exit
```

Started the process, the user can see on the console output the produced cars and the offset number recovered from Kafka.

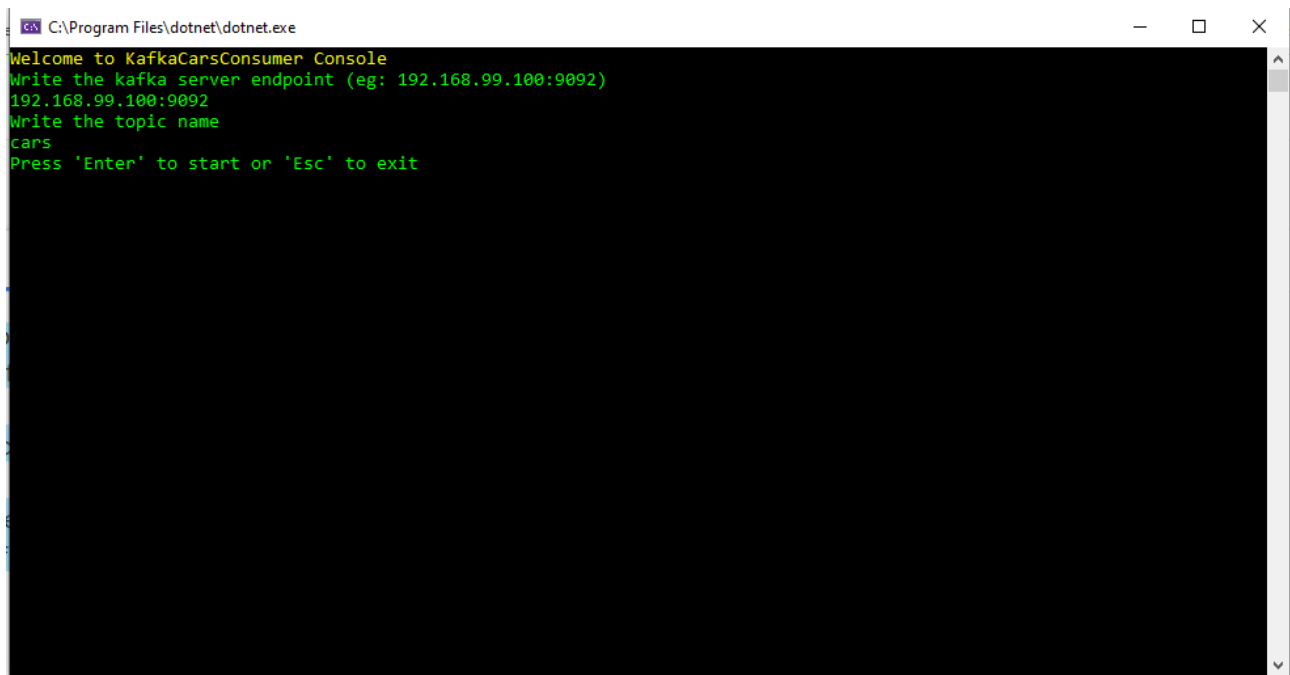
```
Welcome to KafkaCarsProducer Console
Write the kafka server endpoint (eg: 192.168.99.100:9092)
192.168.99.100:9092
Write the topic name
cars
Press 'Enter' to start or 'Esc' to exit
Starting producer
Produced offset: 184 - Car: {"BrandName":"BMW","Model":"XEMSV","DoorsNumber":2,"IsSportsCar":false}
Produced offset: 185 - Car: {"BrandName":"Ferrari","Model":"NIUCF","DoorsNumber":5,"IsSportsCar":false}
Produced offset: 186 - Car: {"BrandName":"Audi","Model":"QOECQ","DoorsNumber":3,"IsSportsCar":true}
Produced offset: 187 - Car: {"BrandName":"BMW","Model":"OEXNX","DoorsNumber":2,"IsSportsCar":true}
Produced offset: 188 - Car: {"BrandName":"Fiat","Model":"FZTHG","DoorsNumber":4,"IsSportsCar":true}
Produced offset: 189 - Car: {"BrandName":"Volkswagen","Model":"AGUGB","DoorsNumber":2,"IsSportsCar":false}
Produced offset: 190 - Car: {"BrandName":"Porsche","Model":"CNMHD","DoorsNumber":3,"IsSportsCar":false}
```

Starting Consumer.

As the producer console application, also the consumer asks the endpoint address of the Kafka server and the topic to consume.

```
C:\Program Files\dotnet\dotnet.exe
Welcome to KafkaCarsConsumer Console
Write the kafka server endpoint (eg: 192.168.99.100:9092)
192.168.99.100:9092
Write the topic name
cars
```

After these steps the user can choose to start the process or not.



```
C:\Program Files\dotnet\dotnet.exe
Welcome to KafkaCarsConsumer Console
Write the kafka server endpoint (eg: 192.168.99.100:9092)
192.168.99.100:9092
Write the topic name
cars
Press 'Enter' to start or 'Esc' to exit
```

Started the process, the user can see on the console output the consumed cars recovered from Kafka.




```
Welcome to KafkaCarsConsumer Console
Write the kafka server endpoint (eg: 192.168.99.100:9092)
192.168.99.100:9092
Write the topic name
cars
Press 'Enter' to start or 'Esc' to exit
Starting consuming...

Consumed: {"BrandName":"BMW","Model":"XEMSV","DoorsNumber":2,"IsSportsCar":false}
Consumed: {"BrandName":"Ferrari","Model":"NIUCF","DoorsNumber":5,"IsSportsCar":false}
Consumed: {"BrandName":"Audi","Model":"QOECQ","DoorsNumber":3,"IsSportsCar":true}
Consumed: {"BrandName":"BMW","Model":"OEXNX","DoorsNumber":2,"IsSportsCar":true}
Consumed: {"BrandName":"Fiat","Model":"FZTHG","DoorsNumber":4,"IsSportsCar":true}
Consumed: {"BrandName":"Volkswagen","Model":"AGUGB","DoorsNumber":2,"IsSportsCar":false}
Consumed: {"BrandName":"Porsche","Model":"CNMHD","DoorsNumber":3,"IsSportsCar":false}
```


Results.

It is possible to see the results of the test directly in the Kafka Manager interface under the Consumer tab.

 **Kafka Manager** ClusterOne Cluster ▾ Brokers Topic ▾ Preferred Replica Election Reassign Partitions Consumers

Clusters / ClusterOne / Consumers

Consumers

Show 10 ▾ entries Search:

Consumer	Type	Topics it consumes from
cars_group	KF	cars : (100% coverage, 3 lag)

Showing 1 to 1 of 1 entries Previous 1 Next