



Real time analysis of data streams using Kafka

Installation instructions:

This assignment's purpose is to introduce students to the basics of Online Learning by comparing 2 different solutions.

- Standard static ML model
- ML model that adapts through time

To achieve the real time data delivery, you will be asked to use Kafka and Docker to simulate an independent server which will send information.

This exercise will make use of Apache Kafka which is a message based streaming platform that operates with 3 main concepts:

- Consumer
- Producer
- Server

We will be using APIs to create a communication bridge between the producer and the consumer and make use of the information provided in real time. In this activity the Producer will not be hosted using a cloud service, we will use containers (Docker) to simulate a remote server on the cloud, but having it running in our local computer.

If you want to know more about what Kafka is and how it works, we recommend these web pages and these blogs [1-3]

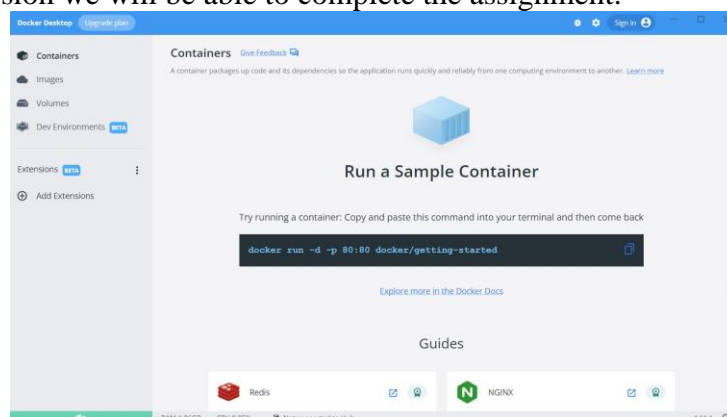
Setup guide

For this activity we need to install the local Docker and Docker Compose programs on our computers.

Please use the following link to download it just take into consideration your operative system. <https://docs.docker.com/get-docker/>



Once your installation is complete, you will have the docker's desktop interface as shown in the following image. It is not necessary to create an account or to perform any upgrade, with the free version we will be able to complete the assignment.



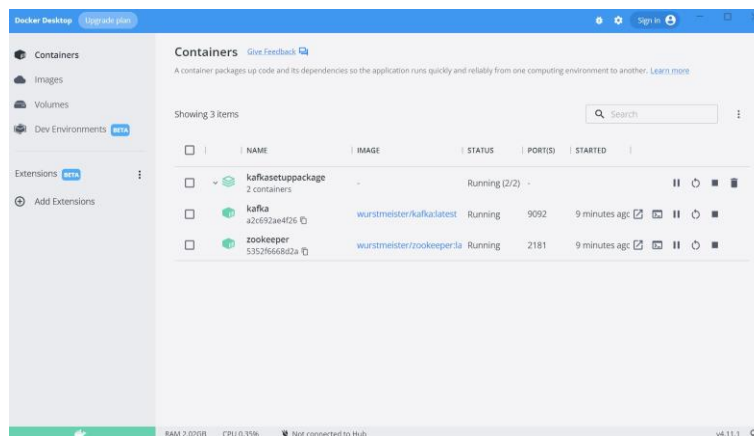
Once it is fully installed download and unzip the folder provided that contains the scripts and the .csv with the data.

Once unzipped in your computer, run one of the following files depending on your operative system:

- If you are using windows then run the **docker_script.bat** file
- Otherwise run the **docker_script.sh** file on Linux/MacOS.

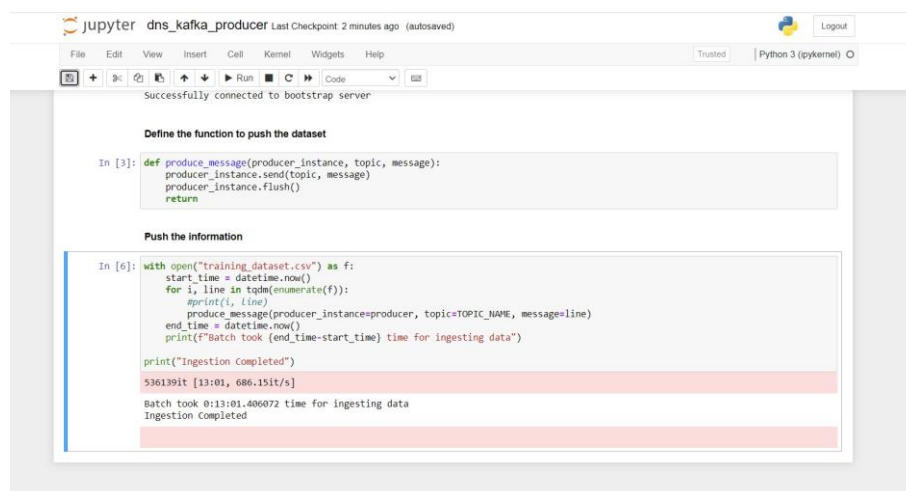
This file will create two docker images `wurstmeister/kafka` and `wurstmeister/zookeeper`. For this activity you don't have to worry about what they are or how they run, as everything is already configured. At the end of the installation.

To validate that the container images were created successfully open Dockers and you should see two images running like this

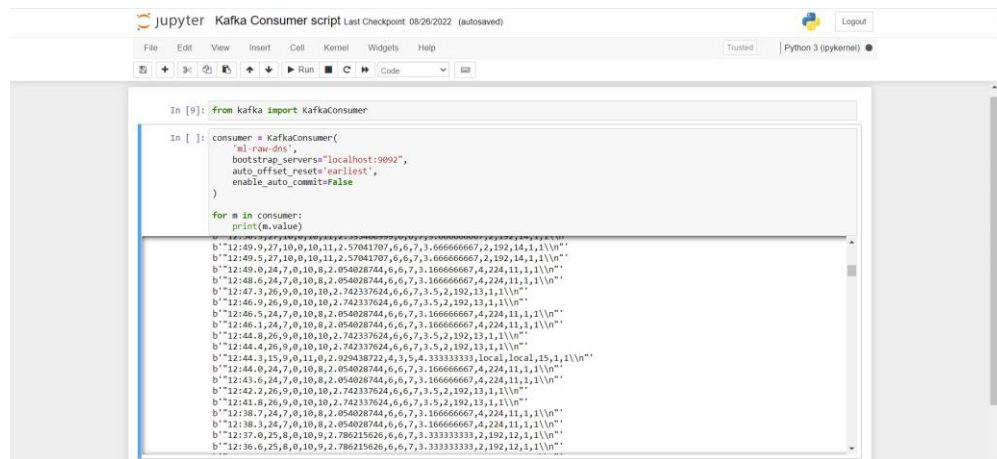


If you have any inconvenience, you can open the file and run it line by line.

Now let's make use of the Producer's API to upload the stream of data into the server. This step will automatically perform the partition and will store the information into the topic, ready for the consumer to read it. Please open the .ipynb called **"dns_kafka_producer.ipynb"**. Run all cells and wait for the information to be uploaded, this file will install the necessary libraries in case you don't have them already and create the producer object with the topic and server's configuration pointing to our docker images. Finally it will read the CSV file and push all the data to the server into the specific topic.



This process will take approximately 10 minutes, so be patient, a conclusion message will be displayed when the process is done and you are able to see inside a red strip the number of rows already inserted; this time can vary depending on your machine. The last step in this process will be to run the **"DNS_Kafka Consumer.ipynb"** which will set up the configuration to read data from the Docker's server and the data ingested by the Producer.



```

In [9]: from kafka import KafkaConsumer

In [ ]: consumer = KafkaConsumer(
    'al-rwdn6',
    bootstrap_servers='localhost:9092',
    auto_offset_reset='earliest',
    enable_auto_commit=False
)

for m in consumer:
    print(m.value)

b"12:49.9,27,10,0,10,11,2.57041707,6,6,7,3.666666667,2,192,14,1,1\n"
b"12:49.5,27,10,0,10,11,2.57041707,6,6,7,3.666666667,2,192,14,1,1\n"
b"12:49.0,24,7,0,10,8,2.054028744,6,6,7,3.166666667,4,224,11,1,1\n"
b"12:48.6,24,7,0,10,8,2.054028744,6,6,7,3.166666667,4,224,11,1,1\n"
b"12:47.3,26,9,0,10,10,2.742337624,6,6,7,3.5,2,192,13,1,1\n"
b"12:46.9,26,9,0,10,10,2.742337624,6,6,7,3.5,2,192,13,1,1\n"
b"12:46.5,24,7,0,10,8,2.054028744,6,6,7,3.166666667,4,224,11,1,1\n"
b"12:46.1,24,7,0,10,8,2.054028744,6,6,7,3.166666667,4,224,11,1,1\n"
b"12:44.8,26,9,0,10,10,2.742337624,6,6,7,3.5,2,192,13,1,1\n"
b"12:44.4,26,9,0,10,10,2.742337624,6,6,7,3.5,2,192,13,1,1\n"
b"12:44.3,15,9,0,11,6,2.920438723,4,3,5,4.333333333,local,local,15,1,1\n"
b"12:44.0,24,7,0,10,8,2.054028744,6,6,7,3.166666667,4,224,11,1,1\n"
b"12:43.6,24,7,0,10,8,2.054028744,6,6,7,3.166666667,4,224,11,1,1\n"
b"12:42.2,26,9,0,10,10,2.742337624,6,6,7,3.5,2,192,13,1,1\n"
b"12:41.8,26,9,0,10,10,2.742337624,6,6,7,3.5,2,192,13,1,1\n"
b"12:38.7,24,7,0,10,8,2.054028744,6,6,7,3.166666667,4,224,11,1,1\n"
b"12:38.3,24,7,0,10,8,2.054028744,6,6,7,3.166666667,4,224,11,1,1\n"
b"12:37.0,25,8,0,10,9,2.786215626,6,6,7,3.333333333,2,192,12,1,1\n"
b"12:36.6,25,8,0,10,9,2.786215626,6,6,7,3.333333333,2,192,12,1,1\n"

```

Now you are ready to capture the data stream and start the data treatment for your model.
Good luck and have fun!

References:

- 1 <https://kafka.apache.org/documentation.html>
- 2 <https://data-flair.training/blogs/apache-kafka-tutorial/>
- 3 <https://medium.com/big-data-engineering/hello-kafka-world-the-complete-guide-to-kafka-with-docker-and-python-f788e2588cfc>