ifgi

# Analyzing Streams of IoT Air Quality Data using Kafka and Jupyter Notebooks

OER4SDI

**Open Educational Resources for Spatial Data Infrastructures**

In this open educational resource (OER) you will learn how to use Kafka and Jupyter notebooks to process and analyze streams of sensor data on particulate matter (PM2.5).

Jaskaran Puri, Albert Remke
Institute for Geoinformatics, University of Münster

August 2024

ein Kooperationsprojekt, empfohlen durch:

DIGITALE HOCHSCHULE NRW

gefördert durch:

Ministry of Culture and Science of the State of North Rhine-Westphalia

# 1. Overview

In this open educational resource (OER) you will learn how to use Kafka and Jupyter notebooks to process and analyze streams of sensor data (particulate matter, PM2.5). After you have completed this tutorial, you will know how to

- use Docker to install and run Apache Kafka and Jupyter Notebooks on your local computer
- use Python to access and download PM2.5 sensor data from the Open Sensemap project
- simulate a PM2.5 sensor data stream that runs against Kafka and how to analyze that data stream for monitoring air quality

The module is structured as follows

1. Overview
2. Background on IoT, sensor data streams and the air quality parameter particulate matter (PM2.5)
3. Installing and using Apache Kafka and Jupyter Notebooks for analyzing PM2.5 data streams
4. Wrap up

If you are mainly interested in the technical aspects, you can jump directly to chapter 3 where we guide you through the technical exercise. With the help of some self-tests you can check if you have understood the essential concepts and technologies.

This tutorial is designed for students and professionals who want to spend about 90 minutes on improving their skills in developing applications based on real-time data. You should have some basic knowledge of Python and it wouldn't be bad if you already have some experience with Docker and Jupyter notebooks too. But don't worry, we will guide you through all those technologies and you can also use the tutorial to get your first hands-on experience with it.

This Tutorial has been developed at the Institute for Geoinformatics, University of Münster. Authors are Jaskaran Puri and Albert Remke with contributions from Sandhya Rajendran, and Thomas Kujawa. The latest version of this tutorial is always available on GitHub. We hope you will use GitHub issues to provide feedback and suggest improvements.

The OER4SDI project has been recommended by the Digital University NRW and is funded by the Ministry of Culture and Science NRW.

# 2. Background

## 2.1 IoT and the analysis of PM2.5 data streams

### - The Internet of Things (IoT)

The Internet of Things (IoT) is about: “interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies”[1]

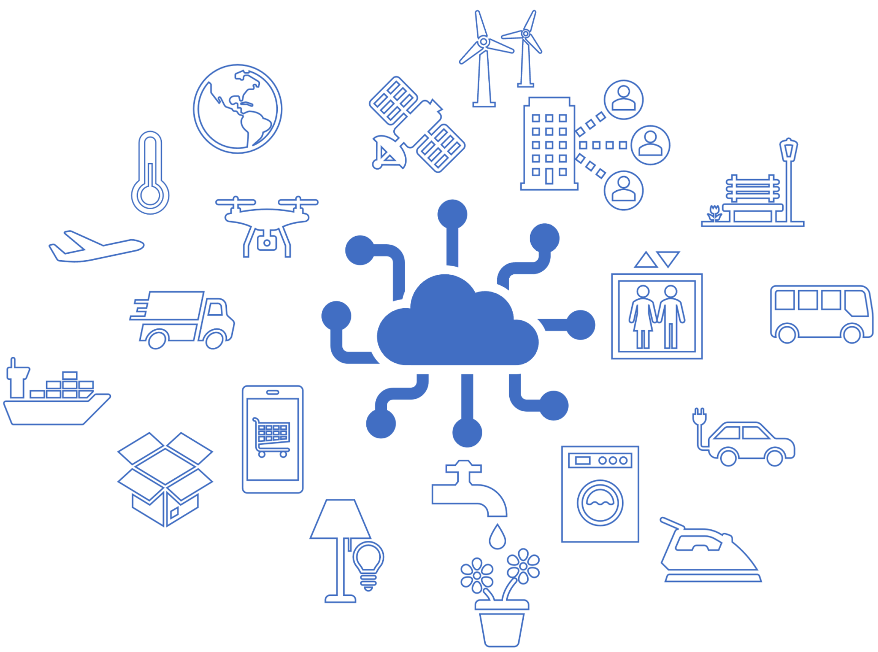

Fig. 1: The Internet of Things

[1] International Telecommunication Union - ITU (2012): Overview of the Internet of things. ITU-T Recommendation Y.2060 (06/2012), online resource https://handle.itu.int/11.1002/1000/11559, last access 2024-08-21

In the IoT, physical things are represented as digital objects that have an identity (e.g., an IRI), a status and capabilities, such as sensing, memorizing, data processing, acting, and communicating.

## - Sensor Things

The main purpose of Sensor Things is to generate data from observations and measurements.

Sensor data are typically spatio-temporal data, i.e. related to space and time.

Fig. 2: Sensor Things

Examples: sensors measuring

- meteorological or hydrological phenomena,
- air quality parameters,
- traffic density,
- power consumption,
- soil moisture,
- land cover,
- …

## - Processing NRT sensor data streams

Sensor data streams are series of time related observations and measurements, often to be processed in near real-time (NRT).

Processing of sensor data streams involves many tasks such as ingestion, storage, analytics, and active dissemination of information products.

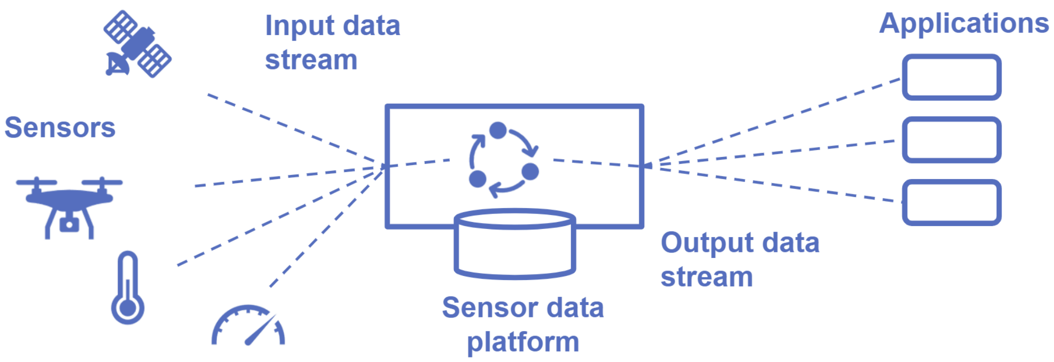


Fig. 3: Processing near-real-time sensor data streams

Each incoming data element changes the system's status and triggers a pipeline of processes, leading to new information products that can be actively disseminated.

## - Near real-time applications

Near real-time applications provide information about the current status of a system (e.g. the atmosphere) with very low latency.

NRT applications support mainly three types of use cases:

a) monitoring the current status of a system
b) detecting events
c) triggering actions

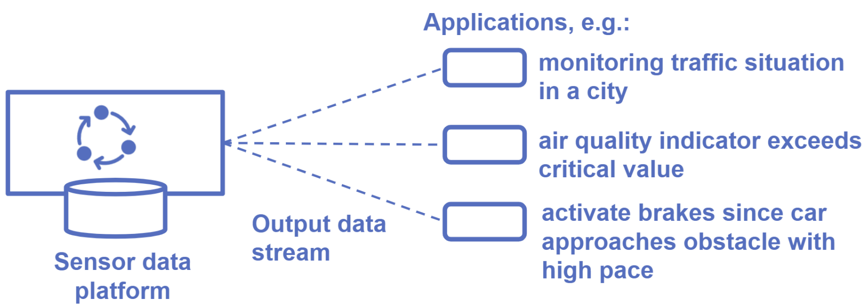


Fig. 4: Near-real-time applications

NRT systems must scale with their payload to guarantee a given latency requirement!

## - Processing data streams with Apache Kafka

Apache Kafka is an open source messaging system designed to process data streams in near real-time.

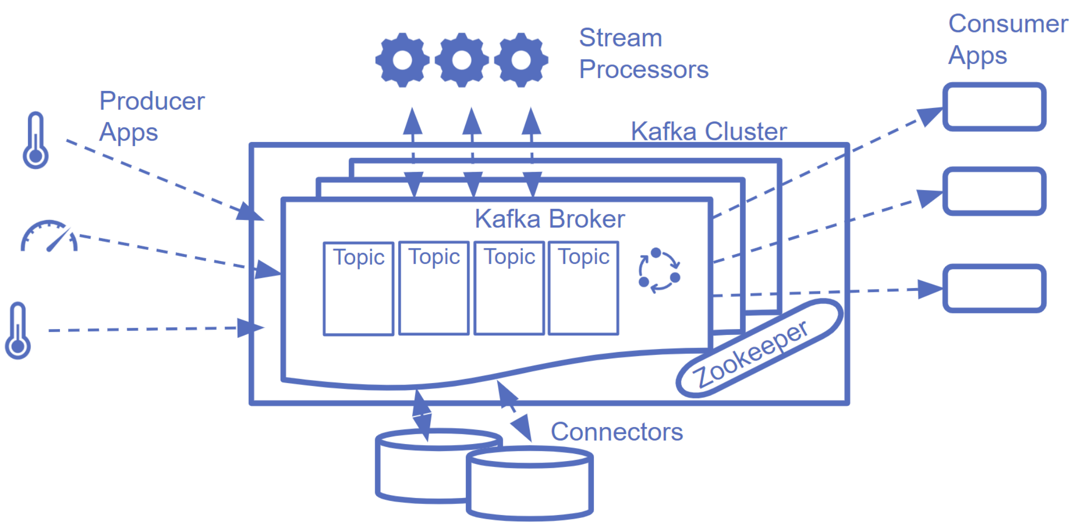


Fig. 5: Processing data streams with Apache Kafka

**Components of the Kafka architecture:**

- Zookeeper is used to orchestrate a scalable set of Kafka brokers (Kafka Cluster)
- Producer apps send topic-related messages
- Stream processors can be used to transform/aggregate messages
- Data connectors can be used to store and access persistent data
- Consumer apps can subscribe to a topic and access/receive/analyze messages

## - The sensor data platform openSenseMap

openSenseMap (https://opensensemap.org/) is an open data platform that supports the collection, analysis, visualization, and sharing of sensor data for citizen science and education.

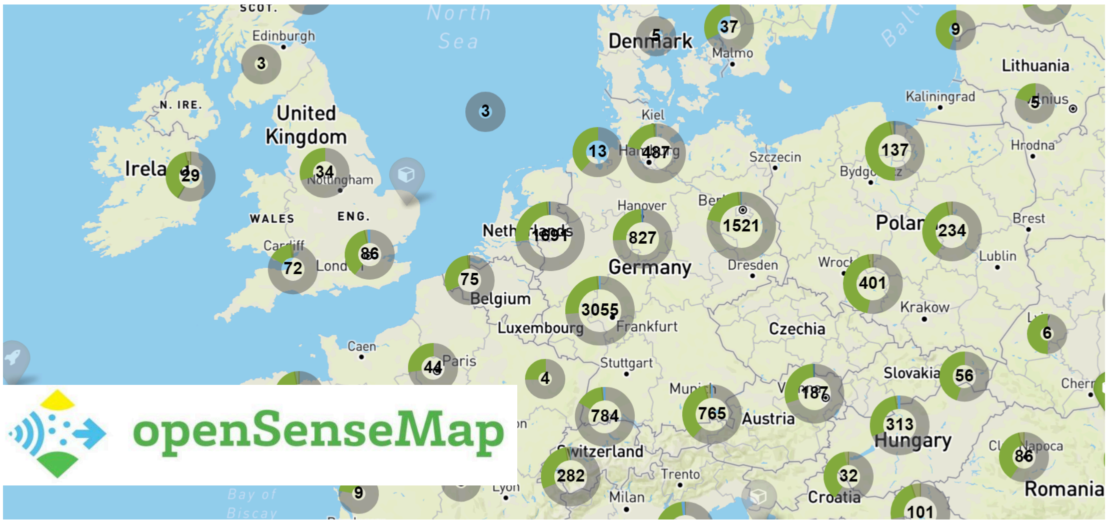


Fig. 6: The sensor data platform openSenseMap

Open Sensemap hosts data from more than 14.000 sensing devices and a data volume of more than 9 bn measurements (mainly meteorological data and air quality parameters).

In this tutorial we'll use data from openSenseMap to showcase how to process and analyze spatiotemporal sensor data streams with Apache Kafka and Jupyter notebooks.

## - Air Pollution and Air Quality

“Air pollution is contamination of the indoor or outdoor environment by any chemical, physical or biological agent that modifies the natural characteristics of the atmosphere.”[2]

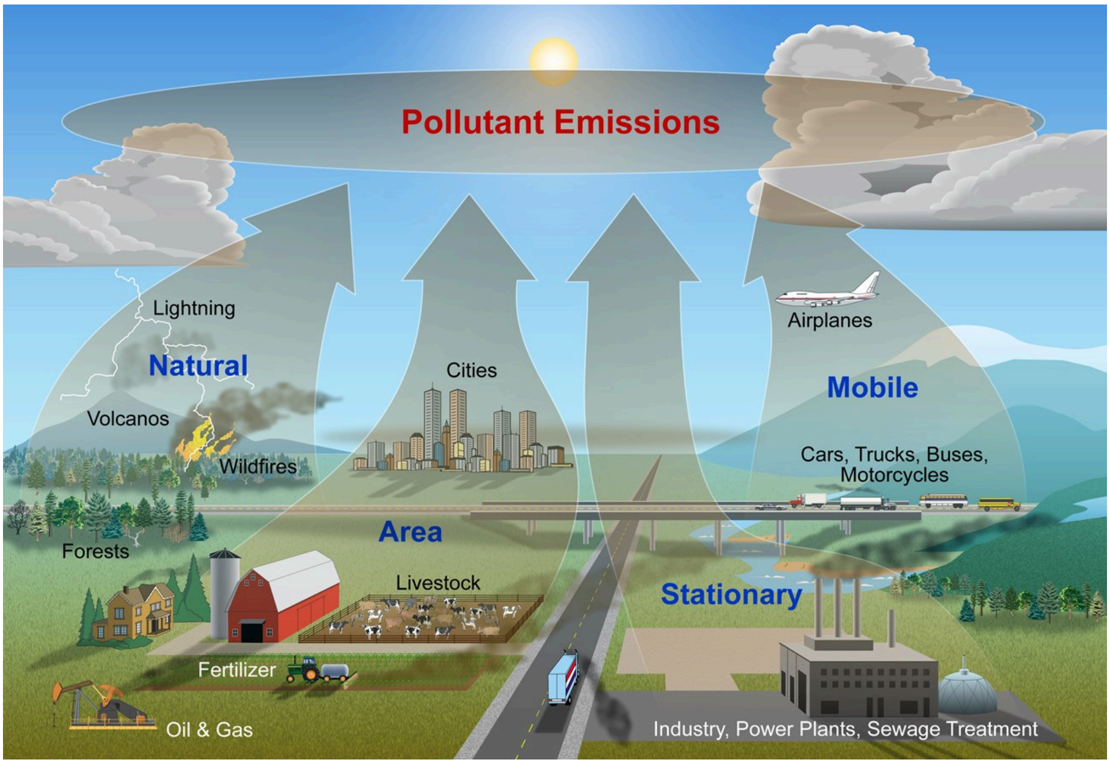


Fig. 7: Sources of air pollution[3]

Many sources of air pollution (such as the burning of fossil fuels) also contribute to greenhouse gas emissions.

Air quality indices (AQIs) are used to indicate how polluted the air is. The European Air Quality Index is based on the measured concentrations of four pollutants: nitrogen dioxide ($NO_2$), sulfur dioxide ($SO_2$), ozone ($O_3$) and particulate matter ($PM_{10}$, PM2.5).[4]

---

[2] World Health Organization(WHO)(2024): WebSite. https://www.who.int/health-topics/air-pollution - last access 2024-08-21.

[3] source: US Department of the Interior, National Park Service - NPS (2022), public domain, https://www.nps.gov/subjects/air/sources.htm, last access 2024-08-21

[4] European Environment Agency (2024): European Air Quality Index - About the European Air Quality Index. WebSite and Map Viewer: https://airindex.eea.europa.eu/AQI/index.html# - last access 2024-08-21

## - Particulate Matter

The term Particulate Matter (PM) refers to extremely small solid particles and liquid droplets floating in the air.

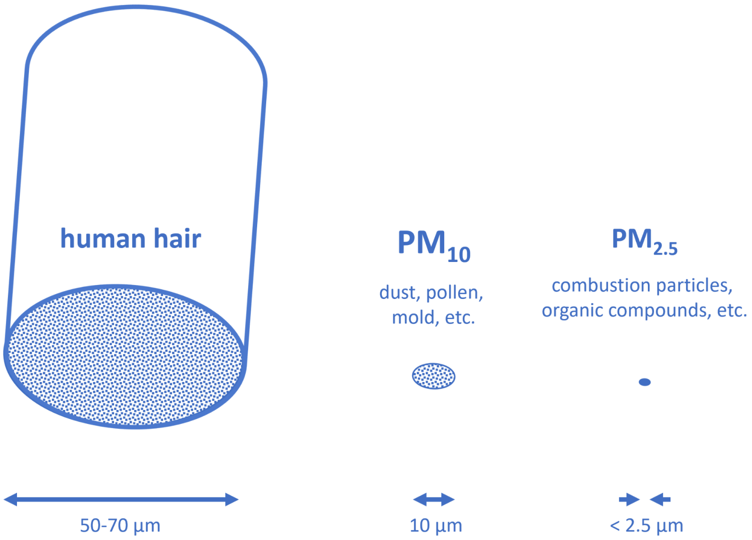


Fig. 8: Comparison of the size of human hair and particulate matter

PM 2.5 and 10 are small enough to pass through the throat and nose and enter the lungs. PM 2.5 can even enter the bloodstreams.

## - PM2.5 - the silent killer

"Ambient (outdoor) air pollution in both cities and rural areas was estimated to cause 4.2 million premature deaths worldwide per year in 2019; this mortality is due to exposure to fine particulate matter, which causes cardiovascular and respiratory disease, and cancers."[5]

[5] World Health Organization(WHO)(2024): Ambient (outdoor) air pollution. WebSite: https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health - last access 2024-08-21


Fig. 9: Health risks related to particulate matter

WHO ambient air quality guidelines suggest an annual mean PM2.5 concentration limit of 5 µg/m3 and 15 µg/m3 for the 24-hourly mean.[6]

## 2.2 Check your Knowledge

- Please give one example each of geospatial near-real-time applications that support a) monitoring of a system b) detection of events c) triggering actions.
- Visit openSenseMap and find out what sensors can tell you about the weather in the city of Münster right now.
- Please give at least two examples each of the short-term and long-term risks of high PM 2.5 concentrations.
- What value does the WHO recommend for a 24-hour average PM 2.5 concentration that should not be exceeded?

[6] World Health Organization - WHO (2021): WHO global air quality guidelines. Executive summary. Geneva:; 2021. https://iris.who.int/bitstream/handle/10665/345334/9789240034433-eng.pdf - last access 2024-08-21

# 3. Installing and using Apache Kafka and Jupyter Notebooks for analyzing PM 2.5 data streams

This section guides you through the installation and use of Docker, Kafka and Jupyter Notebooks for analyzing PM 2.5 data streams.

Software components used in this tutorial:

- **Docker** allows us to package all needed software components as Docker Images and execute those images as Docker Containers in the Docker Environment, e.g. on Linux, Windows or Mac. With the docker-compose tool we can define multiple docker images and configure how they communicate with each other.

- **Apache Kafka** is a messaging system. Kafka is used for applications that need to receive, process and distribute large amounts of incoming data with a latency of milliseconds. Kafka is conceptually based on a publish-subscribe architecture in which one type of systems (**producers**) publish topic-related messages via a virtual **broker** and other types of systems (**consumers**) subscribe to these topics at the broker, filter them, access them and use these messages for real-time applications.The **topics** are useful to structure the data streams and to support scaling of the Kafka system. The broker acts as the bridge between producers and consumers. The broker also acts as a message store, where messages can wait to be consumed by a consumer app.

- Recently, **Zookeeper** became an optional component, but for some years it was the backbone for Kafka clusters. Zookeeper is used to coordinate clusters of Kafka brokers. For example, Zookeeper "knows" which servers act as brokers and creates a new broker if one of the brokers fails.

- **Jupyter Notebook** is an interactive web-based environment for creating and using Notebook documents. It implements the reed-eval-print-loop (REPL), i.e., each document can have a sequence of input/output cells which may contain multimedia content or executable code (Python, R, Julia). Once the user activates a code cell, the print-output of the code will be inserted into the document. This supports both, describing a method or workflow, which involves code and direct interaction with the code as to learn and understand how the code works. Following a common practice, our Notebook Documents have the extension ".ipynb".

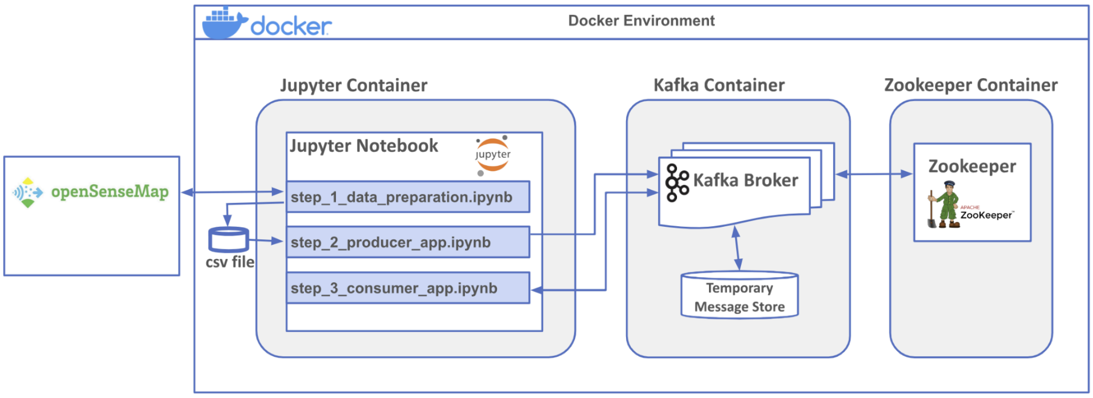


Fig. 10: Overview on the containers used in our docker environment

We use docker to build and run three containers: The **Jupyter container** runs the Jupyter Notebook server. We'll use three Jupyter Notebooks to implement and explain the software that is needed to support Step-1, Step-2 and Step-3 of our technical exercise.

- The **Kafka Container** runs the Kafka Broker, which receives the sensor data, stores them in a temporal data store and provides access to the data for consumer apps.
- The third container runs **zookeeper**, which coordinates the Kafka cluster.

In our exercise, we will first download PM 2.5 sensor data from the openSenseMap project (Step-1). In a second step we will use this sample data to create a sensor data stream that is sent to a Kafka broker by a producer app (Step-2). Finally, we will show how to analyze and visualize the PM 2.5 data by a consumer app (Step-3).

Please notice: In this exercise, we simplify a real-world scenario where many sensors (producer apps) would continuously send their data to the broker. Many consumer apps would sign up for topics of their interest, access the corresponding data in either push or pull mode and process the data in near real-time.

## 3.1 Installing the SW environment

Now that we've gained a basic understanding of the workflow and the technologies involved, let's set up the software environment.

**A) Install Docker**

Please go to the official web site [https://docs.docker.com/get-docker/](https://docs.docker.com/get-docker/) and follow the guidance which is provided there to install docker on your local computer (Linux, Windows or Mac). It is recommended to have at least 8GB RAM to support smooth functioning of Docker.

**B) Download the sources from GitHub**

Download the zip file for the code and unzip it in a location that you want to use as a **WorkingDirectory**.

Advanced users can also clone it using `git` and the following command:

```
git clone https://github.com/oer4sdi/OER-spatial-data-streaming
```

**C) Start the containers**

Please ensure `docker` is up and running in background and open a `CMD/Terminal` in your OS.

At the command prompt, change to the WorkingDirectory (e.g. "OER-spatial-data-streaming") and start up the docker containers:

```
cd OER-spatial-data-streaming
docker compose up --build -d
```

After successful execution you should see a similar console output:

```
[+] Running 4/4
 - Network oer-spatial-data-streaming_default        Created     1.5s
 - Container jupyter                                 Started    12.2s
 - Container oer-spatial-data-streaming_zookeeper_1  Started    12.2s
 - Container oer-spatial-data-streaming_kafka_1      Started    14.2s
```

At this point, you should have all the three containers running: `zookeeper`, `kafka` and `jupyter.`

## 3.2 Downloading the PM 2.5 sample data set (Step-1)

As a first step, we want to download PM 2.5 sensor data from the openSenseMap project and we will use our first Jupyter Notebook document to perform this task.

As to get the URL of the Jupyter Notebook server, open a new `CMD/Terminal` window and enter the following command:

```
docker logs jupyter
```

The output should look like this:

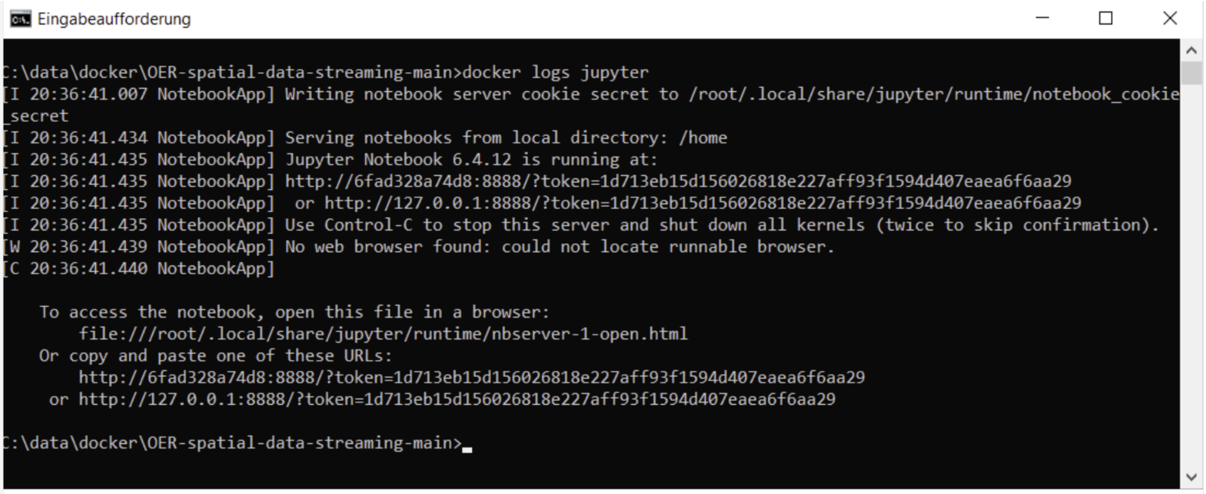


Fig. 11: Docker logs

Goto your browser and access the URL that starts with `http://127.0.0.1:8888/?token=` (please take the `token` from your previous command output).

You will see the UI of the Jupyter Notebook server which informs you about the files that are available and the documents that are currently running. Please open the folder “src” to see the three Notebook documents that are prepared for our exercise.

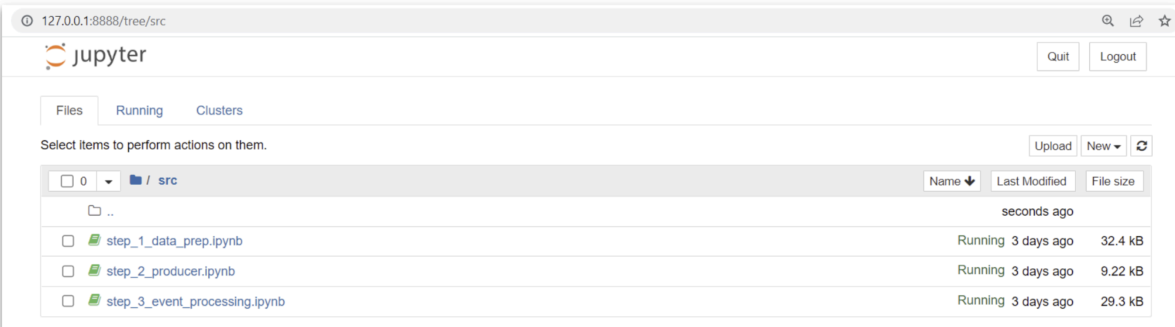


Fig. 12: UI of the Jupyter Notebook Server

Please start the first Notebook document `src/step_1_data_prep.ipynb` and activate the sequence of cells of the document one by one. In the document, you will be required to perform a few tasks to complete the data downloading process.

Please be aware:

- Some of the code cells need some time to complete the computing, i.e., please wait for the output before you continue with the next cell.

- Each cell works with the current state of the system, which is a result of the computations that have been activated before. I.e., the order in which you activate cells is important. If you are not sure about the state of the system, please re-initialize the system by re-starting the kernel (see buttons in the Jupyter Notebook UI).
- If you change the notebook document (You are invited to experiment with the code!), the changes will be persisted in your notebook. If you are not sure about how to fix problems that occurred with your changes you still have the possibility to fall back on downloading a fresh copy of the notebook document from the OER code repository, in the folder `src´.

After having completed the Notebook document please come back and continue with the next chapter of this tutorial.

## 3.3 Sending a PM 2.5 data stream to the Kafka Broker (Step-2)

After we have downloaded the PM 2.5 sample data from openSenseMap we can now use the data from the resulting CSV file to produce a data stream that will be sent to the Kafka broker.

From the Jupyter Notebook UI in your browser: please start the second Notebook document `src/step_2_producer.ipynb` and follow the guidance there.

Once you have successfully activated the sequence of cells in the Notebook document, you should see an output of messages confirming the transmission of all PM 2.5 measurements from the CSV file.

After having completed the Notebook document, please come back and continue with the next chapter of this tutorial.

## 3.4 Analyzing and Visualizing PM 2.5 data streams (Step-3)

Now the Kafka broker has received a number of messages with PM 2.5 sensor data. Each message contains the information on what has been measured when and where. All messages have been tagged to belong to the topic “pm25_stream”. The Kafka broker has stored the messages in a temporary message store.

Our next Jupyter Notebook implements a Consumer app that connects to the Kafka server and subscribes to the topic “pm25_stream” to get access to the sensor data stream. The Consumer app will use the data for two purposes:

- to trigger actions in the case that a sensor observes PM 2.5 concentrations that exceed a certain threshold for more than three days in a row,
- to create a map that informs us about the location and current status of the PM 2.1 sensors.

Now please open the third Notebook document `src/step_3_event_processing.ipynb`, read and activate the cells one-by-one.

After having completed the Notebook document, please come back and continue with the next chapter of this tutorial.

## 3.5 Shut down and clean up

Now that we have used the Notebook documents to perform and understand all the tasks of our exercise we shut down and clean up our working environment.

In your `CMD/Terminal` window that you used to build the docker images type `docker compose down` to shut down the docker containers.

The docker images are still available in your docker environment. Next time when you want to run the environment, you can just use `docker compose up -d` to start up the containers again.

If you want to remove the images as well from your docker environment type `docker images` to get a list of the available images. Then use `docker image rm [image id]` to remove the images that you want to delete.

# 4. Wrap up

Hey! You did a great job! You installed and applied a powerful software stack for processing sensor data, including Docker, Kafka, Zookeeper and Jupyter Notebook. You prepared a PM 2.5 sample dataset using these technologies, sent a sensor data stream to a Kafka broker, and then analyzed and visualized it using a consumer app. We hope that you now have an idea of how to work with spatial data streams, even though we have somewhat simplified the real applications in our exercise. For example:

- In our example scenario, all messages were sent in a single stream from a single producer; in the real world, these messages come continuously and from many producers at the same time.

- In our example, the consumer pulls the data from the stream in a batch mode; in near-real-time applications, the stream processors are often triggered by incoming data and are thus able to react to new information with the shortest possible delay.
- In our exercise, we downloaded a historical dataset (January 2022) and used the consumer application as if the data was from today. In fact, there was a long period without data in between. However, the consumer app would work the same way if we had used near real-time data.

**Interested in learning more?**

On the Internet you will find a wealth of resources on NRT processing of data streams. Here are some recommendations:


- MQTT.Org (2022): WebSite with information on MQTT, the de facto messaging protocol standard for IoT applications. https://mqtt.org
- Waehner, Kai (2020): Apache Kafka + Kafka Connect + MQTT Connector + Sensor Data. A practical example on how to combine Kafka and MQTT. GitHub repository. https://github.com/kaiwaehner/kafka-connect-iot-mqtt-connector-example
- Hughes, Jim (2021): GeoMesa – Big Data for GIS. FOSS4G 2021 Buenos Aires, October 1, 2021. Presentation: https://www.geomesa.org/assets/outreach/foss4g-2021-streaming-data.pdf
- Mollenkopf, A. (2017): Applying Geospatial Analytics at a Massive Scale using Kafka, Spark & Elastic Search on DC/OS. MesosCon North America: https://www.youtube.com/watch?v=sa4RiH1RXEA&ab_channel=TheLinuxFoundation


**Your feedback is welcome!**

If you have identified shortcomings in this OER module or have ideas for improving the OER material, you are invited to add entries to the issue list in the Github repository of this OER.