## Project Documentation: Simulating Sensor Behaviour with MQTT, MongoDB, Redis, and FastAPI

**Overview**

This document provides an overview of a project that simulates the behaviour of sensors, monitors their readings, and provides APIs to retrieve data based on specific criteria. The project integrates various technologies, including MQTT, MongoDB, Redis, and FastAPI, all orchestrated using Docker Compose.

**Table of Contents**

1. **Prerequisites**
2. **Project Structure**
3. **MQTT Broker Setup**
4. **MQTT Publisher**
5. **MQTT Subscriber**
6. **Data Storage with MongoDB**
7. **In-Memory Data Management with Redis**
8. **FastAPI Endpoint Design**
9. **Docker Integration with Docker Compose**
10. **Challenges and Solutions**
11. **Conclusion**

**1. Prerequisites**

Before getting started, ensure you have the following prerequisites:

* Docker installed on your system
* Python installed on your system
* Using of Docker, MQTT, MongoDB, Redis, and FastAPI

**2. Project Structure**

**/mqtt-publisher**: Python MQTT client for simulating sensor readings.

**/mqtt-subscriber**: Python MQTT subscriber for storing received messages in MongoDB.

/mongo-redis: Implement Redis using Docker to store the latest ten sensor readings

**/fastapi-app**: FastAPI application with endpoints for retrieving sensor data.

**docker-compose.yml**: Docker Compose file for orchestrating services.

**3. MQTT Broker Setup**

**Commands and Syntax**

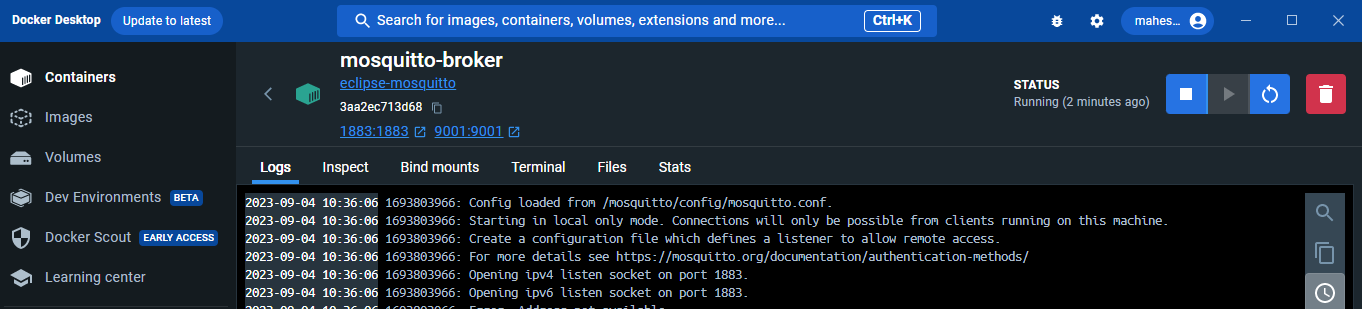
* To deploy the Mosquitto MQTT broker using Docker:

**🡺 docker pull eclipse-mosquitto**

**Docker Run Command**:

**🡺 docker run -d --name mqtt-broker -p 1883:1883 eclipse-mosquitto**

**Output:**



**4. MQTT Publisher**

**Commands and Syntax**

* To start the MQTT publisher to mimic sensor readings:

🡺 **python client\_publisher.py**

**5. MQTT Subscriber**

**Commands and Syntax**

* To run the MQTT subscriber for storing messages in MongoDB:

🡺 **python consumer.py**

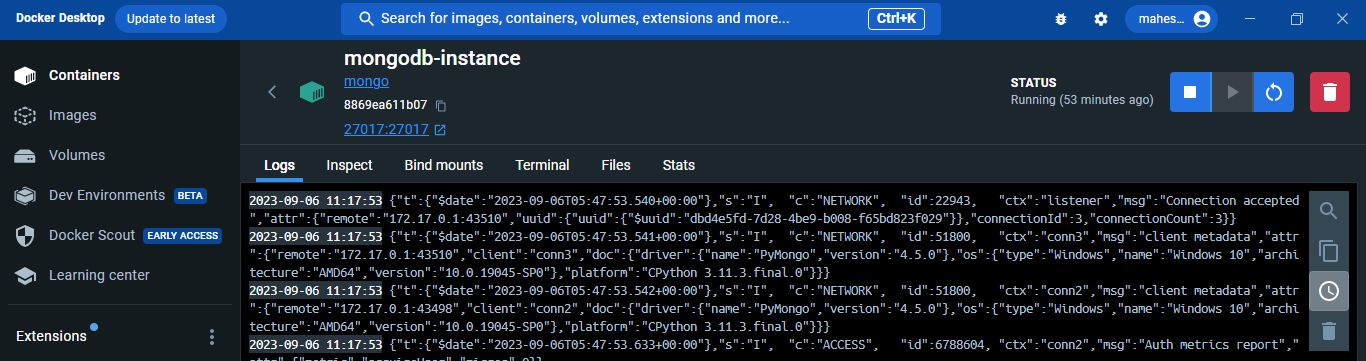
**6. Data Storage with MongoDB**

**Commands and Syntax**

* To initiate a MongoDB instance using Docker:

🡺 **docker run -d --name mongo-instance -p 27017:27017 mongo**

**Output:**



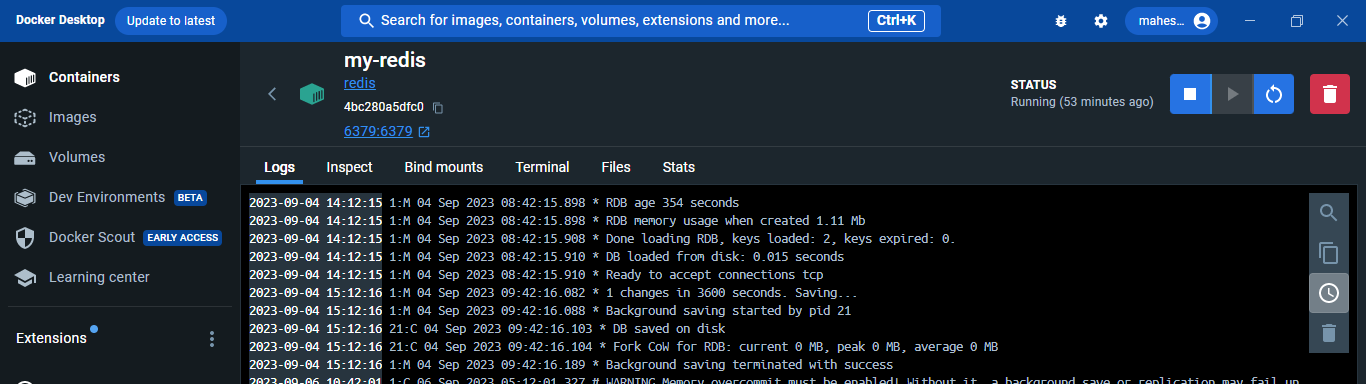
**7. In-Memory Data Management with Redis**

**Commands and Syntax**

* To set up Redis using Docker:

🡺 **docker run -d --name my-redis- -p 6379:6379 redis**

**Output:**



**8. FastAPI Endpoint Design**

* An endpoint that allows users to fetch sensor readings by specifying a start and end range.
* An endpoint to retrieve the last ten sensor readings for a specific sensor.

**9. Docker Integration with Docker Compose**

**Commands and Syntax**

* To start all services using Docker Compose:

🡺 **docker-compose up**

**10. Challenges and Solutions**

**Challenges**

Describe any challenges encountered during development.

**Solutions**

Explain the solutions implemented to address the challenges.

**11. Conclusion**

In conclusion, this project demonstrates how to simulate sensor behaviour, capture and store sensor data using MQTT, MongoDB, and Redis, and expose the data through a FastAPI-based API. Docker Compose is used to orchestrate the various services, making it easy to set up and manage the project environment.