Projekat 3

Internet stvari i servisa

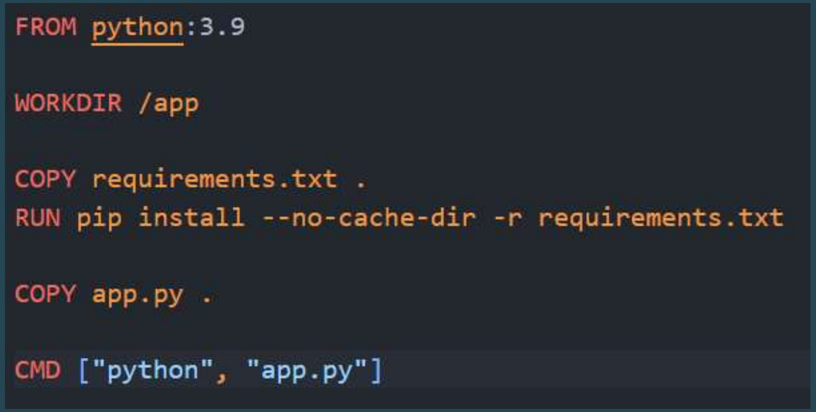
Anja Tonsa Milovanović, 18263

Mikroservisi - odabrane tehnologije

Filter	Dashboard	Command	eKuiper	Sensor
Phython/Flask	Phython/Flask	NodeJS		.NET

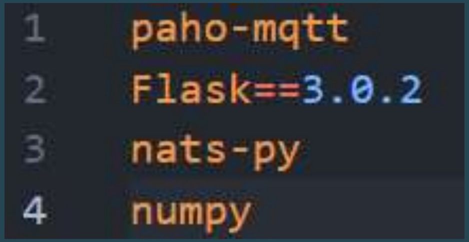
Filter

- Dobija podatke sa topic-a “Sensor data”.
- Računa srednju vrednost u vemenskom prozoru od 10s.
- Šalje srednje vrednosti na topic „averages“ NATS servera.



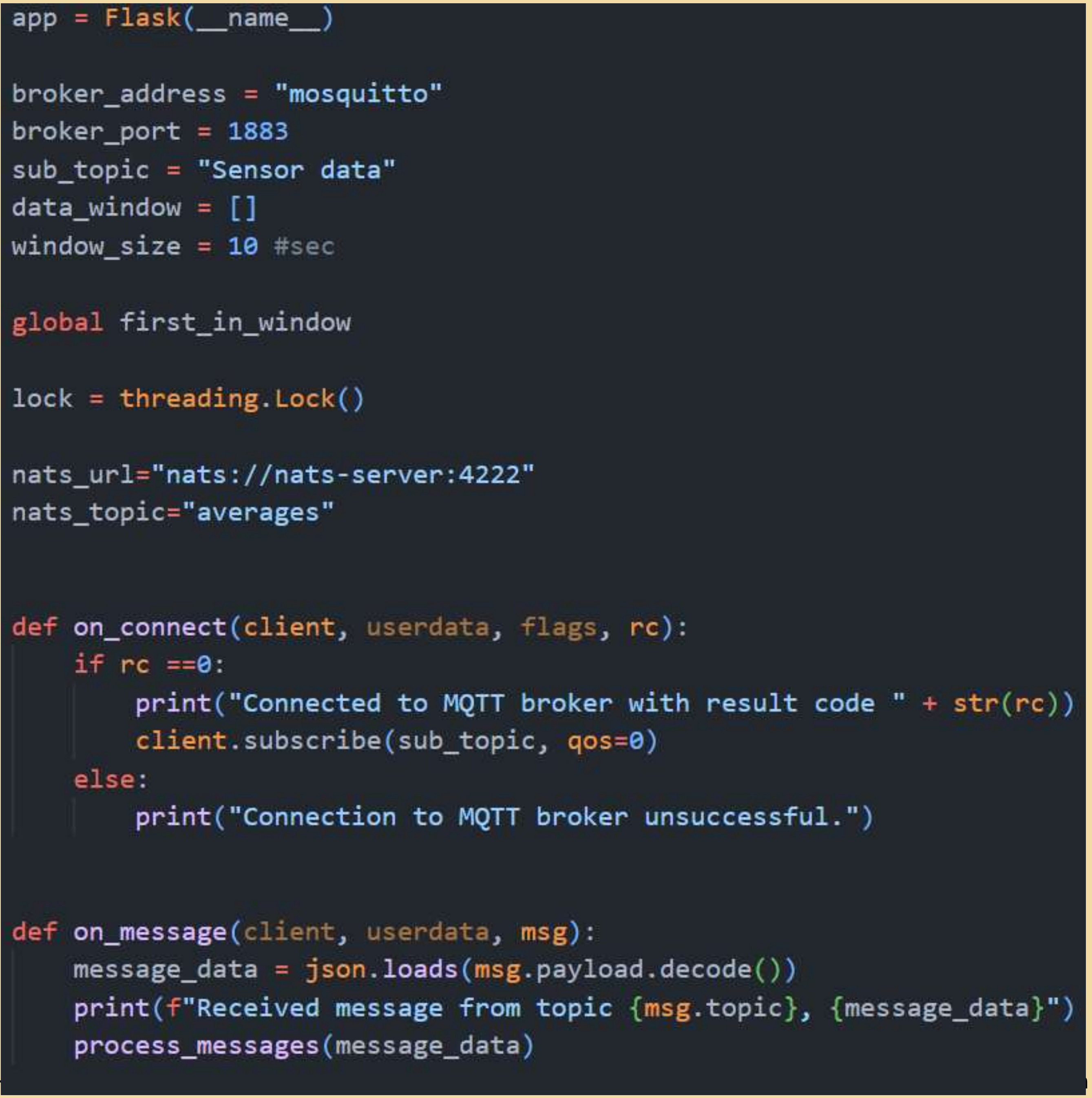
```
FROM python:3.9

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY app.py .

CMD ["python", "app.py"]
```

Dockerfile

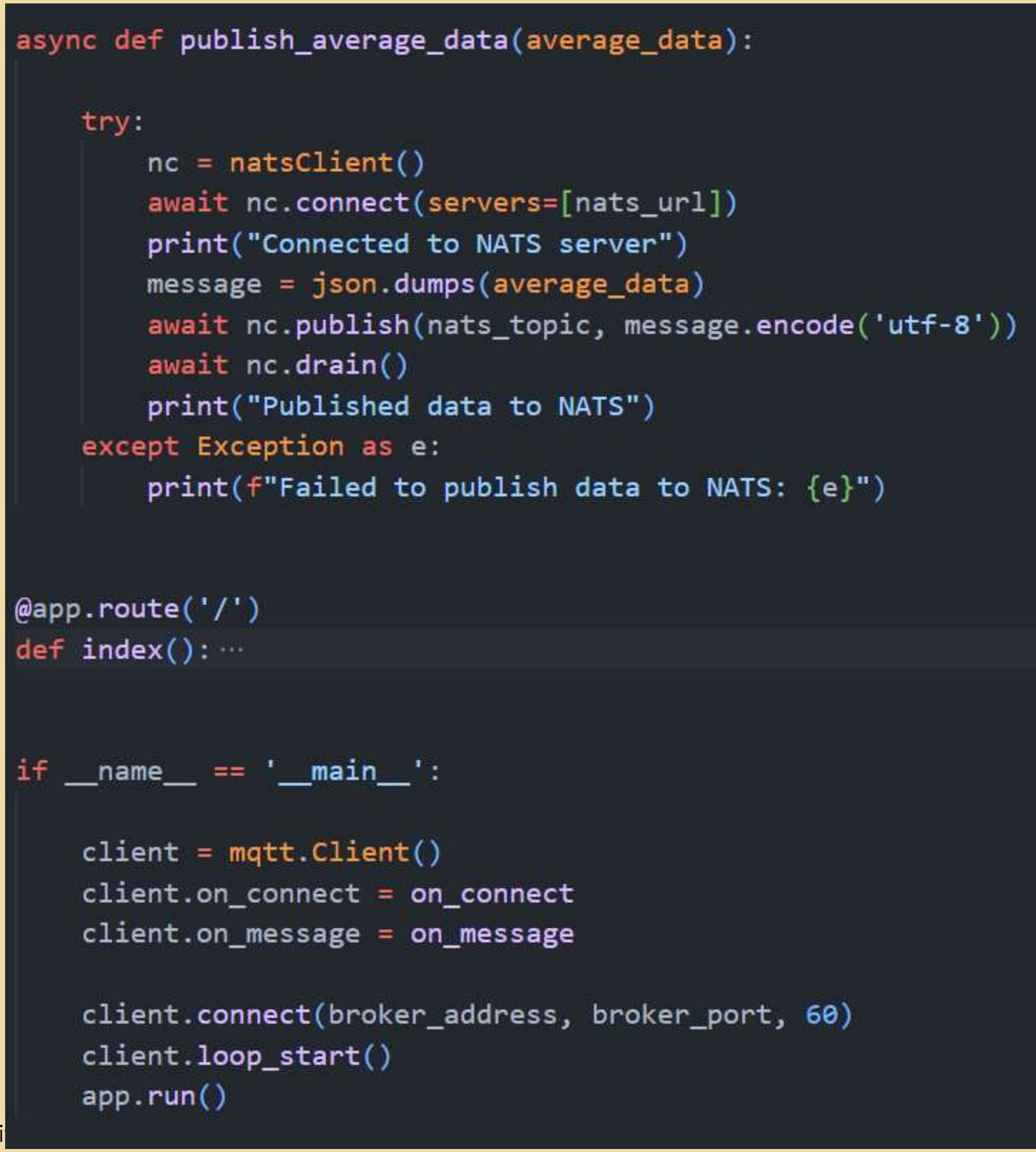


```
paho-mqtt
Flask==3.0.2
nats-py
numpy
```

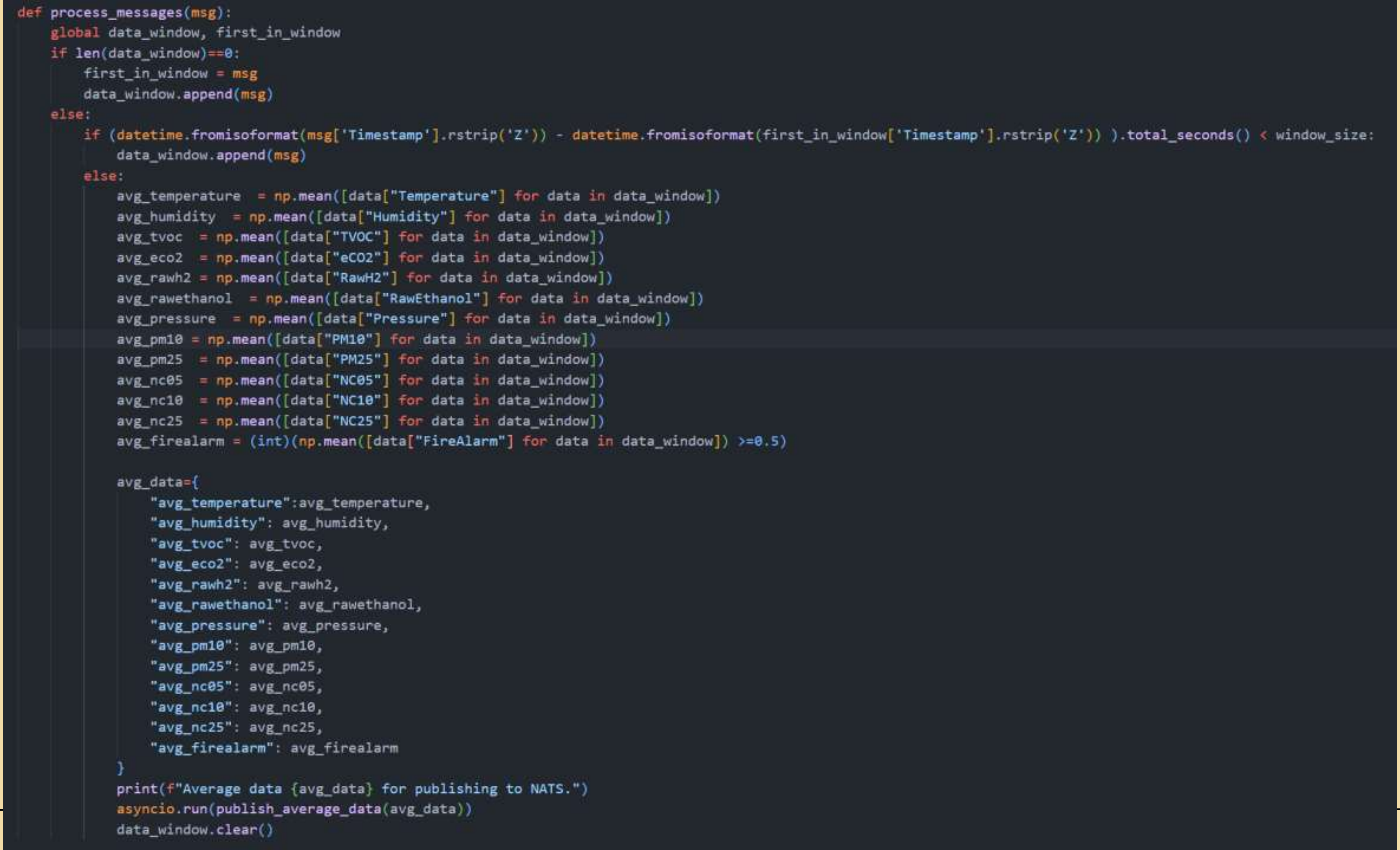
requirements.txt



```python
app = Flask(__name__)

broker_address = "mosquitto"
broker_port = 1883
sub_topic = "Sensor data"
data_window = []
window_size = 10 #sec

global first_in_window

lock = threading.Lock()

nats_url="nats://nats-server:4222"
nats_topic="averages"


def on_connect(client, userdata, flags, rc):
    if rc ==0:
        print("Connected to MQTT broker with result code " + str(rc))
        client.subscribe(sub_topic, qos=0)
    else:
        print("Connection to MQTT broker unsuccessful.")


def on_message(client, userdata, msg):
    message_data = json.loads(msg.payload.decode())
    print(f"Received message from topic {msg.topic}, {message_data}")
    process_messages(message_data)
```



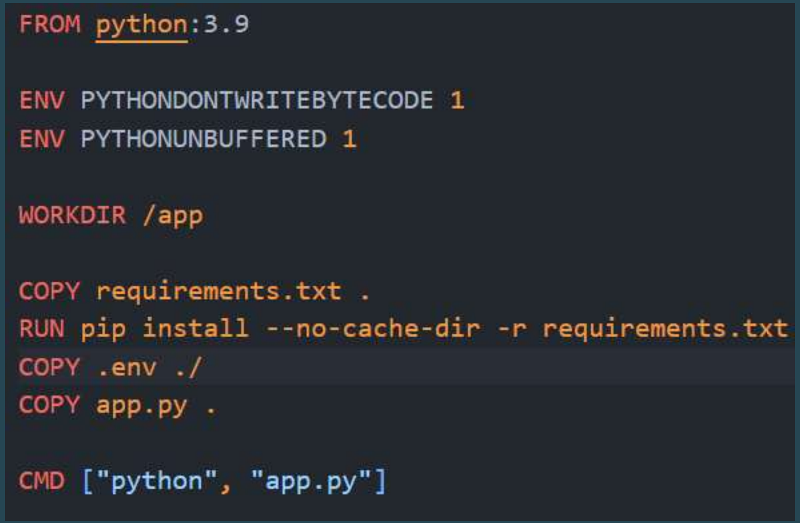
```python
async def publish_average_data(average_data):

    try:
        nc = natsClient()
        await nc.connect(servers=[nats_url])
        print("Connected to NATS server")
        message = json.dumps(average_data)
        await nc.publish(nats_topic, message.encode('utf-8'))
        await nc.drain()
        print("Published data to NATS")
    except Exception as e:
        print(f"Failed to publish data to NATS: {e}")


@app.route('/')
def index(): ···


if __name__ == '__main__':

    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message

    client.connect(broker_address, broker_port, 60)
    client.loop_start()
    app.run()
```



```python
def process_messages(msg):
    global data_window, first_in_window
    if len(data_window)==0:
        first_in_window = msg
        data_window.append(msg)
    else:
        if (datetime.fromisoformat(msg['Timestamp'].rstrip('Z')) - datetime.fromisoformat(first_in_window['Timestamp'].rstrip('Z')) ).total_seconds() < window_size:
            data_window.append(msg)
        else:
            avg_temperature  = np.mean([data["Temperature"] for data in data_window])
            avg_humidity  = np.mean([data["Humidity"] for data in data_window])
            avg_tvoc  = np.mean([data["TVOC"] for data in data_window])
            avg_eco2  = np.mean([data["eCO2"] for data in data_window])
            avg_rawh2 = np.mean([data["RawH2"] for data in data_window])
            avg_rawethanol  = np.mean([data["RawEthanol"] for data in data_window])
            avg_pressure  = np.mean([data["Pressure"] for data in data_window])
            avg_pm10 = np.mean([data["PM10"] for data in data_window])
            avg_pm25  = np.mean([data["PM25"] for data in data_window])
            avg_nc05  = np.mean([data["NC05"] for data in data_window])
            avg_nc10  = np.mean([data["NC10"] for data in data_window])
            avg_nc25  = np.mean([data["NC25"] for data in data_window])
            avg_firealarm = (int)(np.mean([data["FireAlarm"] for data in data_window]) >=0.5)

            avg_data={
                "avg_temperature":avg_temperature,
                "avg_humidity": avg_humidity,
                "avg_tvoc": avg_tvoc,
                "avg_eco2": avg_eco2,
                "avg_rawh2": avg_rawh2,
                "avg_rawethanol": avg_rawethanol,
                "avg_pressure": avg_pressure,
                "avg_pm10": avg_pm10,
                "avg_pm25": avg_pm25,
                "avg_nc05": avg_nc05,
                "avg_nc10": avg_nc10,
                "avg_nc25": avg_nc25,
                "avg_firealarm": avg_firealarm
            }
            print(f"Average data {avg_data} for publishing to NATS.")
            asyncio.run(publish_average_data(avg_data))
            data_window.clear()
```

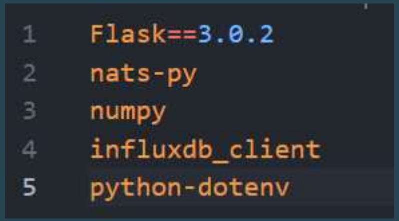
Dashboard

- Dobija podatke sa topic-a „averages“ NATS servera.
- Smešta podatke u InfluxDB.



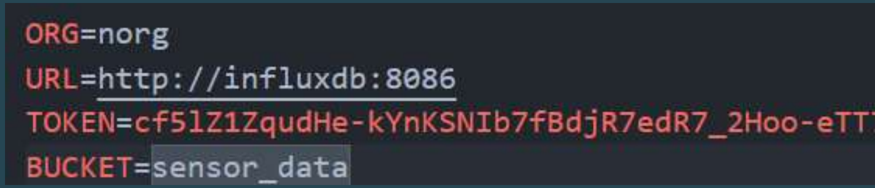
```
FROM python:3.9

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY .env ./
COPY app.py .

CMD ["python", "app.py"]
```

Dockerfile



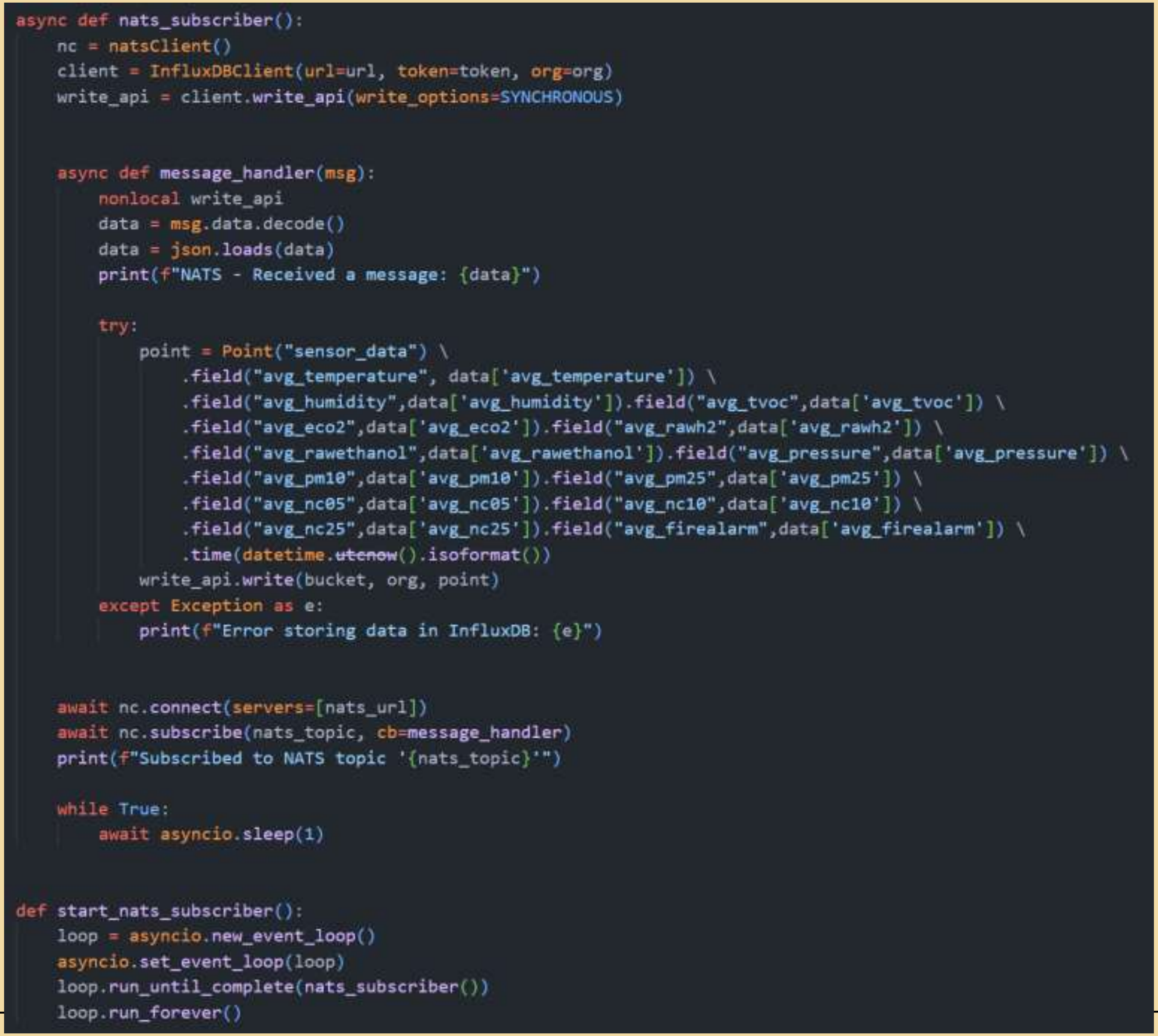
```
Flask==3.0.2
nats-py
numpy
influxdb_client
python-dotenv
```

requirements.txt



```
ORG=norg
URL=http://influxdb:8086
TOKEN=cf51Z1ZqudHe-kYnKSNIb7fBdjR7edR7_2Hoo-eTT
BUCKET=sensor_data
```

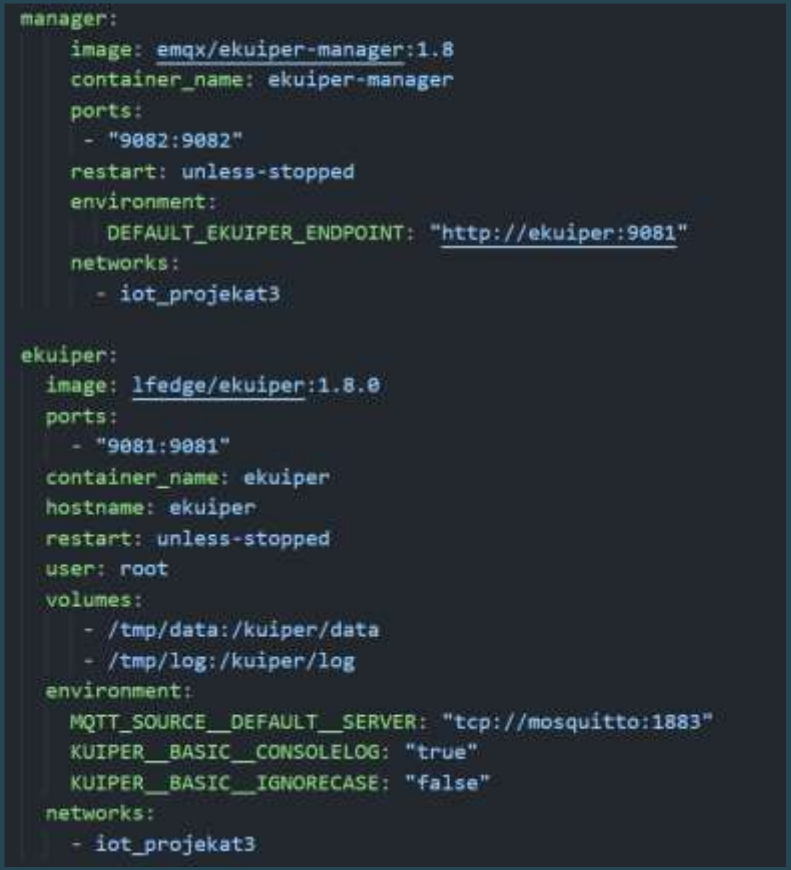
.env



```python
async def nats_subscriber():
    nc = natsClient()
    client = InfluxDBClient(url=url, token=token, org=org)
    write_api = client.write_api(write_options=SYNCHRONOUS)


    async def message_handler(msg):
        nonlocal write_api
        data = msg.data.decode()
        data = json.loads(data)
        print(f"NATS - Received a message: {data}")

        try:
            point = Point("sensor_data") \
                .field("avg_temperature", data['avg_temperature']) \
                .field("avg_humidity",data['avg_humidity']).field("avg_tvoc",data['avg_tvoc']) \
                .field("avg_eco2",data['avg_eco2']).field("avg_rawh2",data['avg_rawh2']) \
                .field("avg_rawethanol",data['avg_rawethanol']).field("avg_pressure",data['avg_pressure']) \
                .field("avg_pm10",data['avg_pm10']).field("avg_pm25",data['avg_pm25']) \
                .field("avg_nc05",data['avg_nc05']).field("avg_nc10",data['avg_nc10']) \
                .field("avg_nc25",data['avg_nc25']).field("avg_firealarm",data['avg_firealarm']) \
                .time(datetime.~~utcnow~~().isoformat())
            write_api.write(bucket, org, point)
        except Exception as e:
            print(f"Error storing data in InfluxDB: {e}")


    await nc.connect(servers=[nats_url])
    await nc.subscribe(nats_topic, cb=message_handler)
    print(f"Subscribed to NATS topic '{nats_topic}'")

    while True:
        await asyncio.sleep(1)


def start_nats_subscriber():
    loop = asyncio.new_event_loop()
    asyncio.set_event_loop(loop)
    loop.run_until_complete(nats_subscriber())
    loop.run_forever()
```

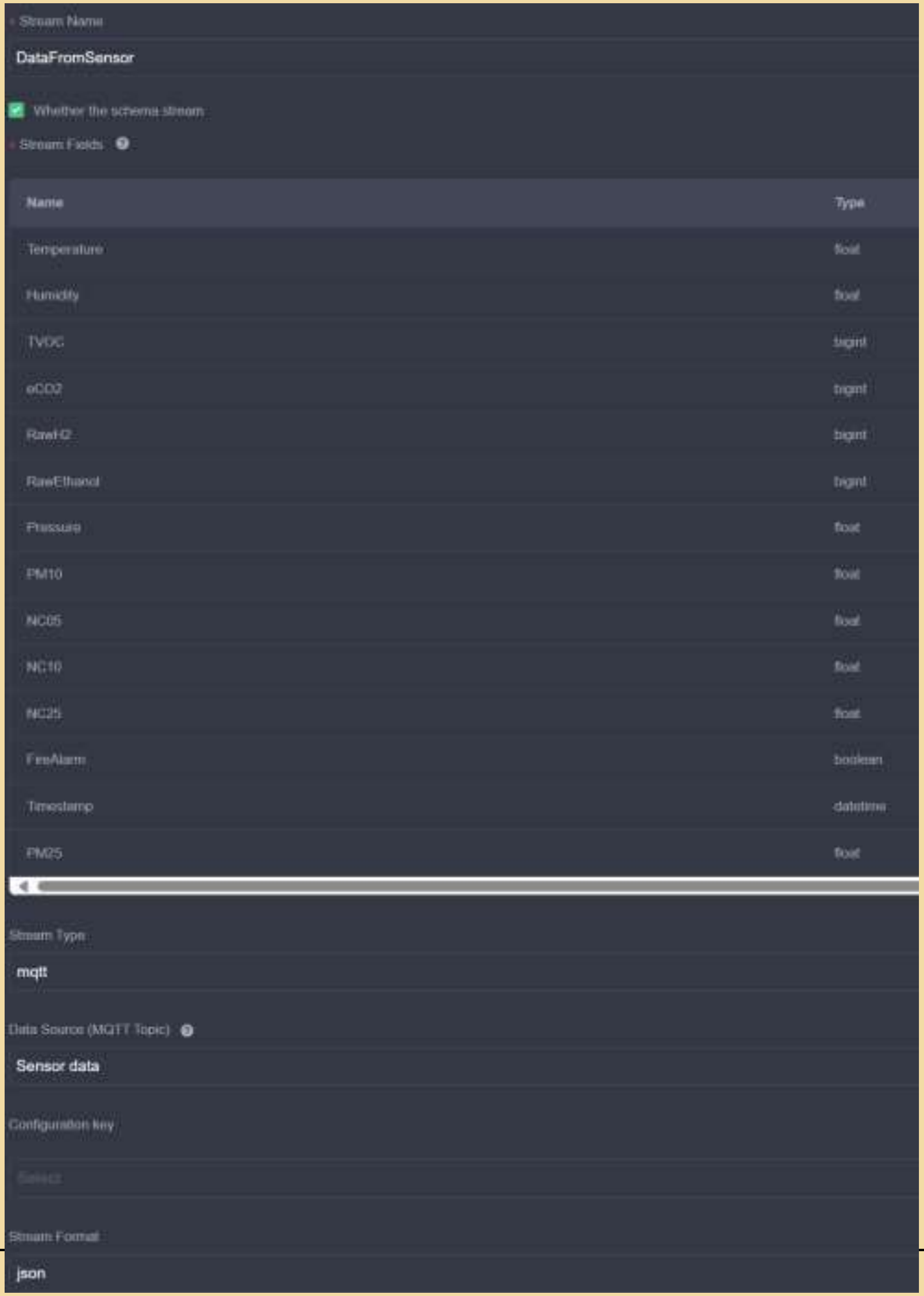
eKupier

- Dobija podatke sa topic-a “Sensor data”.
- Detektuje 2 dogadja: kada temperatura predje 23 stepena i kada je fireAlarm jednako True.
- Šalje poruke o dogadjajima na topic-e „Alert“ i „AlertTemp“.



```yaml
manager:
    image: emqx/ekuiper-manager:1.8
    container_name: ekuiper-manager
    ports:
     - "9082:9082"
    restart: unless-stopped
    environment:
       DEFAULT_EKUIPER_ENDPOINT: "http://ekuiper:9081"
    networks:
      - iot_projekat3

ekuiper:
  image: lfedge/ekuiper:1.8.0
  ports:
    - "9081:9081"
  container_name: ekuiper
  hostname: ekuiper
  restart: unless-stopped
  user: root
  volumes:
     - /tmp/data:/kuiper/data
     - /tmp/log:/kuiper/log
  environment:
    MQTT_SOURCE__DEFAULT__SERVER: "tcp://mosquitto:1883"
    KUIPER__BASIC__CONSOLELOG: "true"
    KUIPER__BASIC__IGNORECASE: "false"
  networks:
    - iot_projekat3
```

Deo iz Docker-compose.yaml fajla.



Stream Name

DataFromSensor

✓ Whether the schema stream

Stream Fields

Name	Type
Temperature	float
Humidity	float
TVOC	bigint
eCO2	bigint
RawH2	bigint
RawEthanol	bigint
Pressure	float
PM10	float
NC05	float
NC10	float
NC25	float
FireAlarm	boolean
Timestamp	datetime
PM25	float

Stream Type

mqtt

Data Source (MQTT Topic)

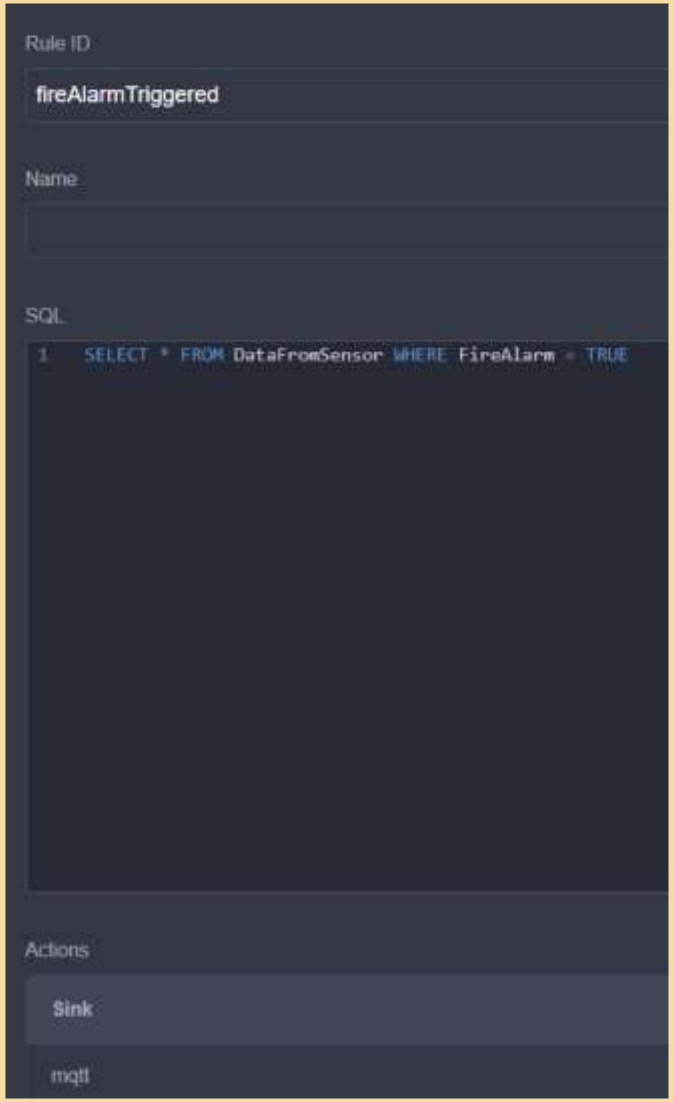
Sensor data

Configuration key

Select

Stream Format

json



Rule ID

fireAlarmTriggered

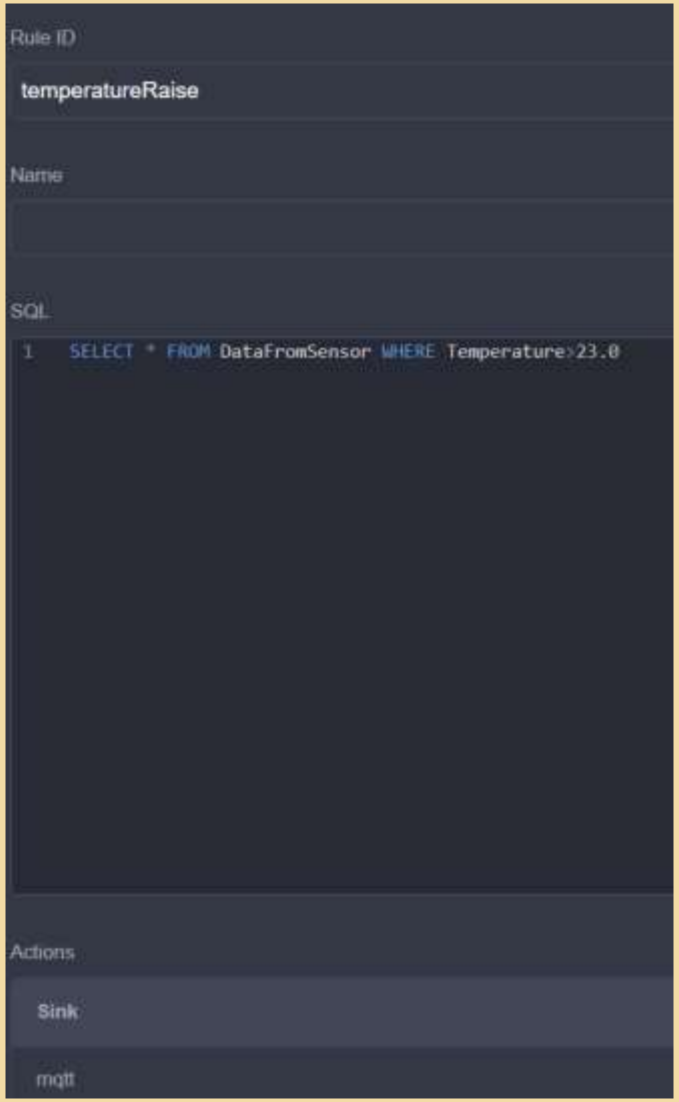
Name

SQL

```sql
SELECT * FROM DataFromSensor WHERE FireAlarm = TRUE
```

Actions

Sink
mqtt



Rule ID

temperatureRaise

Name

SQL

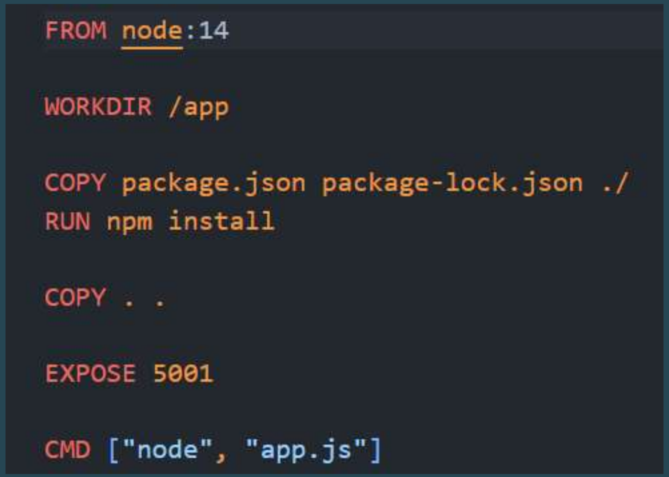
```sql
SELECT * FROM DataFromSensor WHERE Temperature>23.0
```

Actions

Sink
mqtt

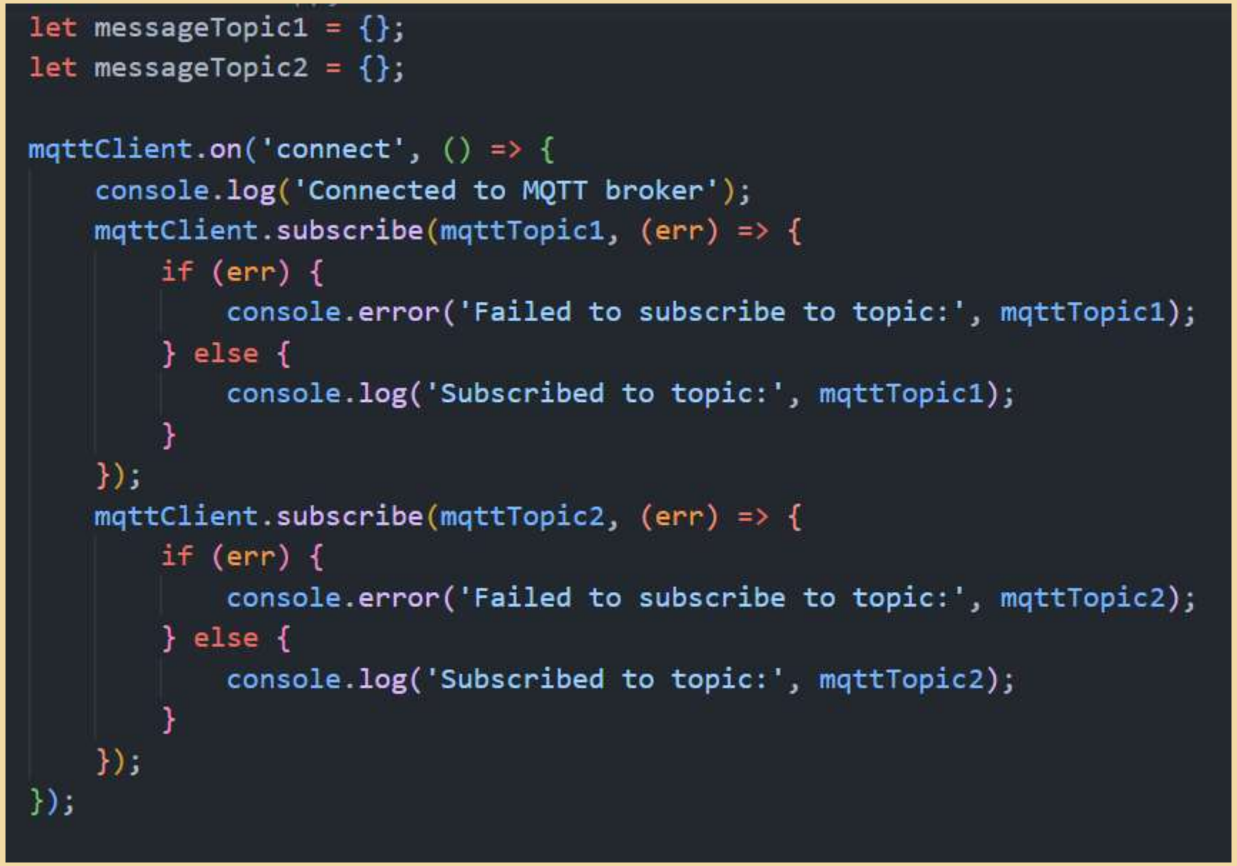
Command

- Dobija podatke sa topic-a „Alert“ i „AlertTemp“.
- Registrovane dogadjaje prikazuje na Web stranici.

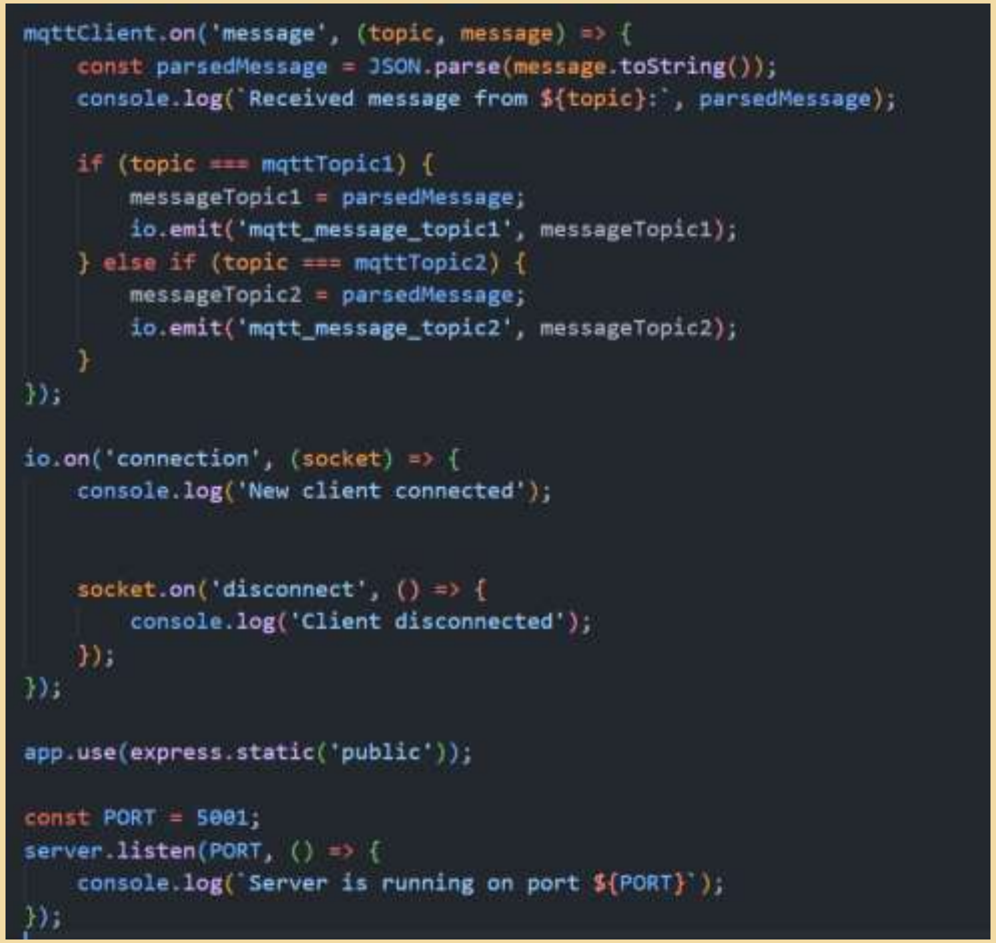


```
FROM node:14

WORKDIR /app

COPY package.json package-lock.json ./
RUN npm install

COPY . .

EXPOSE 5001

CMD ["node", "app.js"]
```

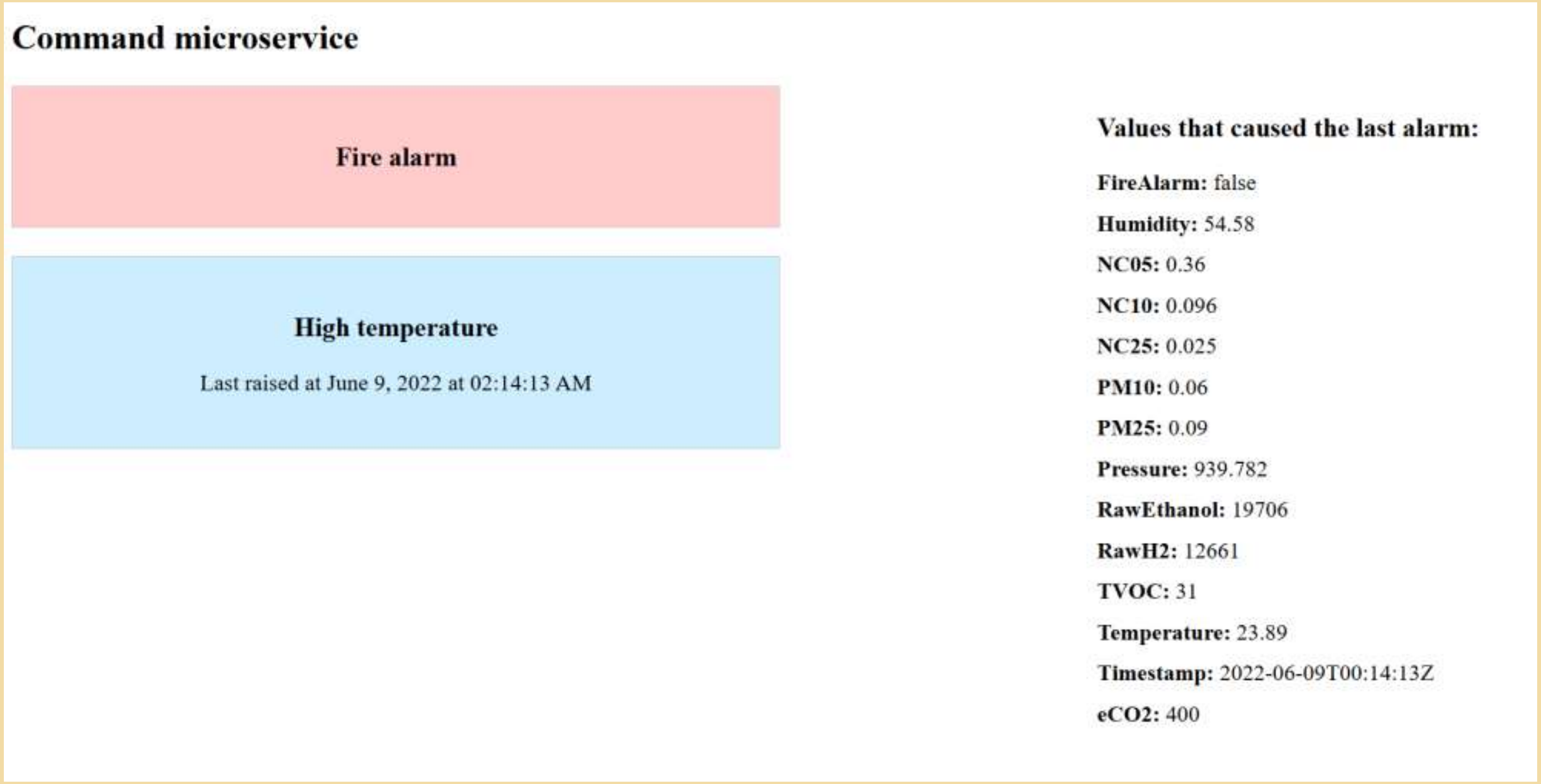
Dockerfile



```
let messageTopic1 = {};
let messageTopic2 = {};

mqttClient.on('connect', () => {
    console.log('Connected to MQTT broker');
    mqttClient.subscribe(mqttTopic1, (err) => {
        if (err) {
            console.error('Failed to subscribe to topic:', mqttTopic1);
        } else {
            console.log('Subscribed to topic:', mqttTopic1);
        }
    });
    mqttClient.subscribe(mqttTopic2, (err) => {
        if (err) {
            console.error('Failed to subscribe to topic:', mqttTopic2);
        } else {
            console.log('Subscribed to topic:', mqttTopic2);
        }
    });
});
```



```
mqttClient.on('message', (topic, message) => {
    const parsedMessage = JSON.parse(message.toString());
    console.log(`Received message from ${topic}:`, parsedMessage);

    if (topic === mqttTopic1) {
        messageTopic1 = parsedMessage;
        io.emit('mqtt_message_topic1', messageTopic1);
    } else if (topic === mqttTopic2) {
        messageTopic2 = parsedMessage;
        io.emit('mqtt_message_topic2', messageTopic2);
    }
});

io.on('connection', (socket) => {
    console.log('New client connected');


    socket.on('disconnect', () => {
        console.log('Client disconnected');
    });
});

app.use(express.static('public'));

const PORT = 5001;
server.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```



Command microservice

Fire alarm

High temperature

Last raised at June 9, 2022 at 02:14:13 AM

Values that caused the last alarm:

FireAlarm: false

Humidity: 54.58

NC05: 0.36

NC10: 0.096

NC25: 0.025

PM10: 0.06

PM25: 0.09

Pressure: 939.782

RawEthanol: 19706

RawH2: 12661

TVOC: 31

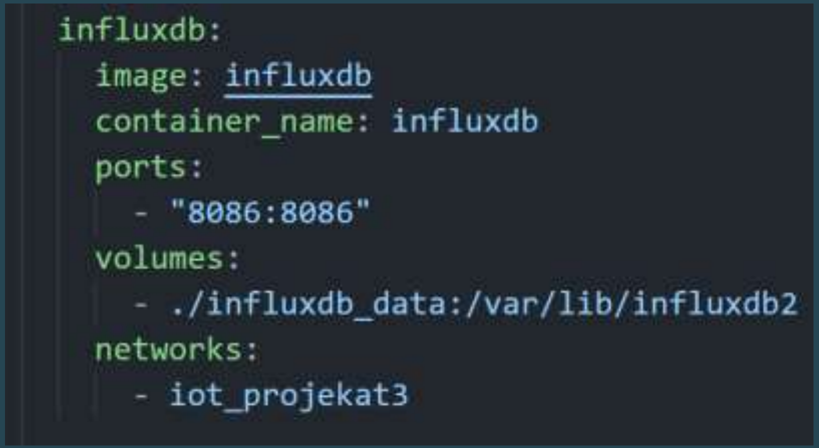
Temperature: 23.89

Timestamp: 2022-06-09T00:14:13Z

eCO2: 400

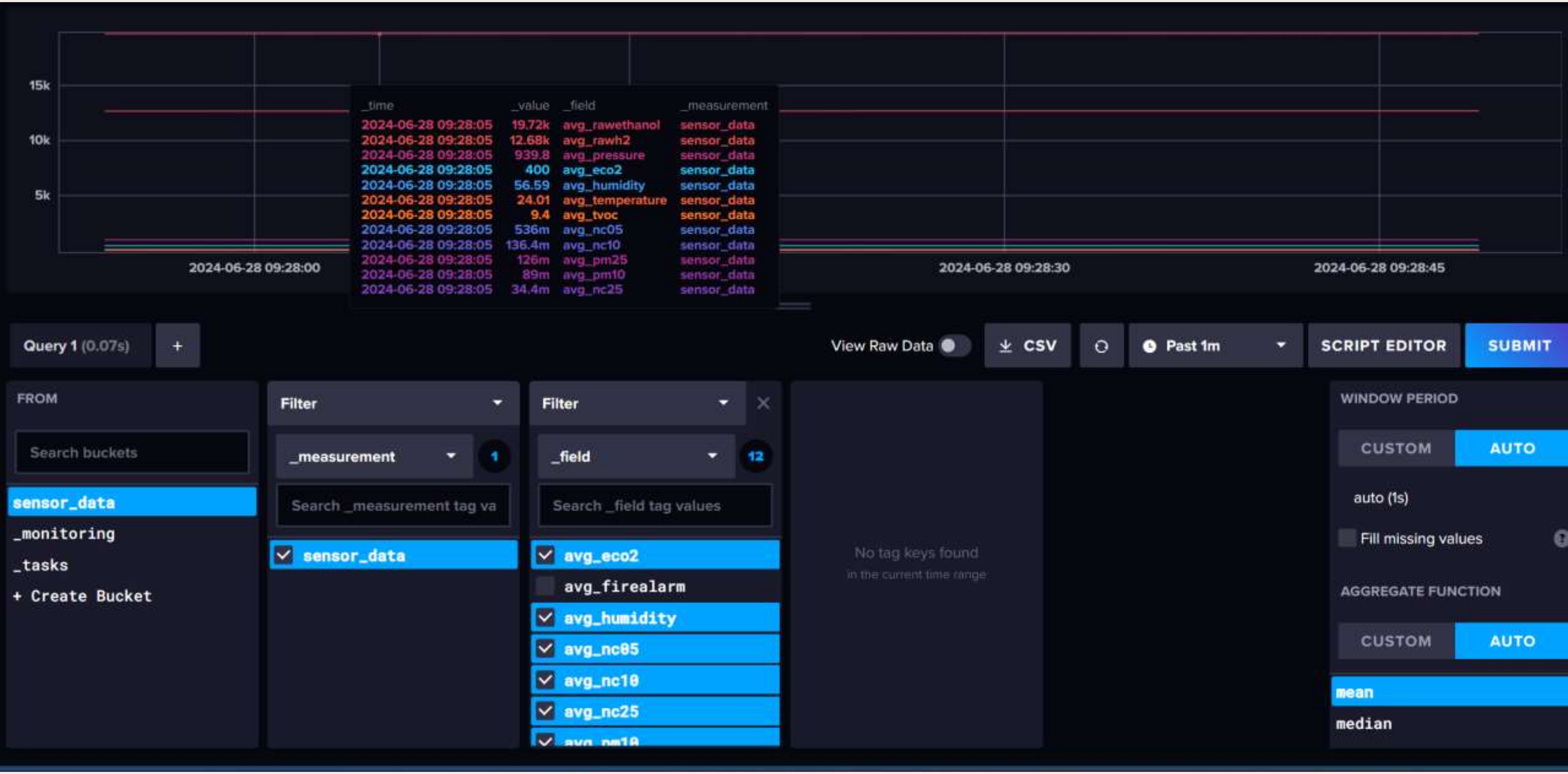
InfluxDB

- Pamti podatke koje joj prosledjuje Dashboard mikroservis.



```yaml
influxdb:
  image: influxdb
  container_name: influxdb
  ports:
    - "8086:8086"
  volumes:
    - ./influxdb_data:/var/lib/influxdb2
  networks:
    - iot_projekat3
```

Deo iz Docker-compose.yaml fajla.



_time	_value	_field	_measurement
2024-06-28 09:28:05	19.72k	avg_rawethanol	sensor_data
2024-06-28 09:28:05	12.68k	avg_rawh2	sensor_data
2024-06-28 09:28:05	939.8	avg_pressure	sensor_data
2024-06-28 09:28:05	400	avg_eco2	sensor_data
2024-06-28 09:28:05	56.59	avg_humidity	sensor_data
2024-06-28 09:28:05	24.01	avg_temperature	sensor_data
2024-06-28 09:28:05	9.4	avg_tvoc	sensor_data
2024-06-28 09:28:05	536m	avg_nc05	sensor_data
2024-06-28 09:28:05	136.4m	avg_nc10	sensor_data
2024-06-28 09:28:05	126m	avg_pm25	sensor_data
2024-06-28 09:28:05	89m	avg_pm10	sensor_data
2024-06-28 09:28:05	34.4m	avg_nc25	sensor_data

Past 1m

FROM
Search buckets
sensor_data
_monitoring
_tasks
+ Create Bucket

Filter
_measurement 1
Search _measurement tag va
[✓] sensor_data

Filter
_field 12
Search _field tag values
[✓] avg_eco2
[ ] avg_firealarm
[✓] avg_humidity
[✓] avg_nc05
[✓] avg_nc10
[✓] avg_nc25
[✓] avg_pm10

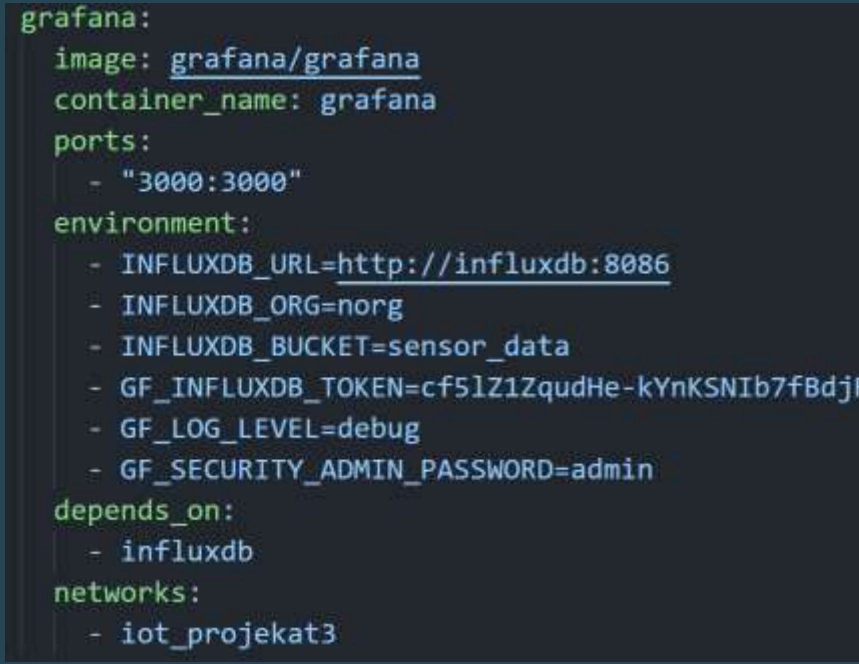
No tag keys found
in the current time range

WINDOW PERIOD
CUSTOM [✓ AUTO]
auto (1s)
[ ] Fill missing values

AGGREGATE FUNCTION
CUSTOM [✓ AUTO]
[✓ mean]
median

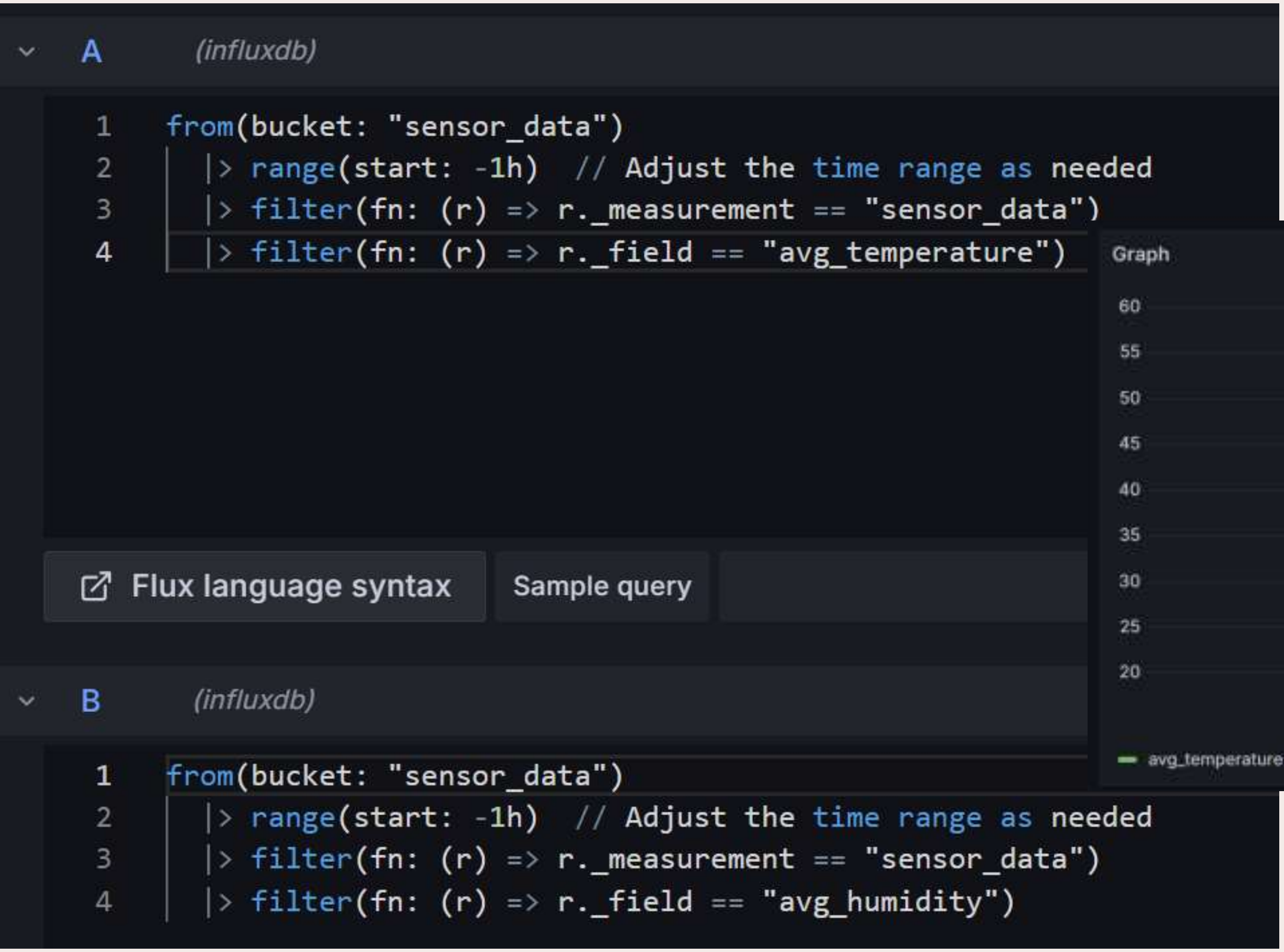
Grafana

- Vizuelno prikazuje podatke zapamćene u InfluxDB-u.



```yaml
grafana:
  image: grafana/grafana
  container_name: grafana
  ports:
    - "3000:3000"
  environment:
    - INFLUXDB_URL=http://influxdb:8086
    - INFLUXDB_ORG=norg
    - INFLUXDB_BUCKET=sensor_data
    - GF_INFLUXDB_TOKEN=cf5lZ1ZqudHe-kYnKSNIb7fBdj
    - GF_LOG_LEVEL=debug
    - GF_SECURITY_ADMIN_PASSWORD=admin
  depends_on:
    - influxdb
  networks:
    - iot_projekat3
```

Deo iz Docker-compose.yaml fajla.

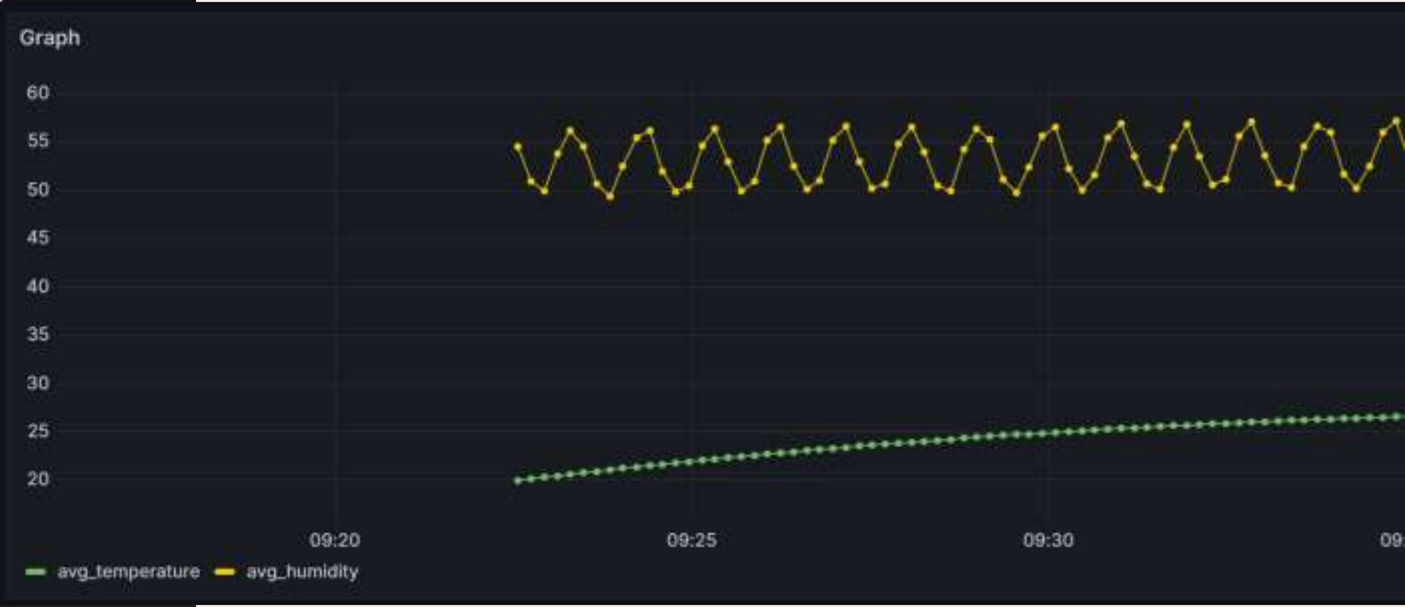


A (influxdb)

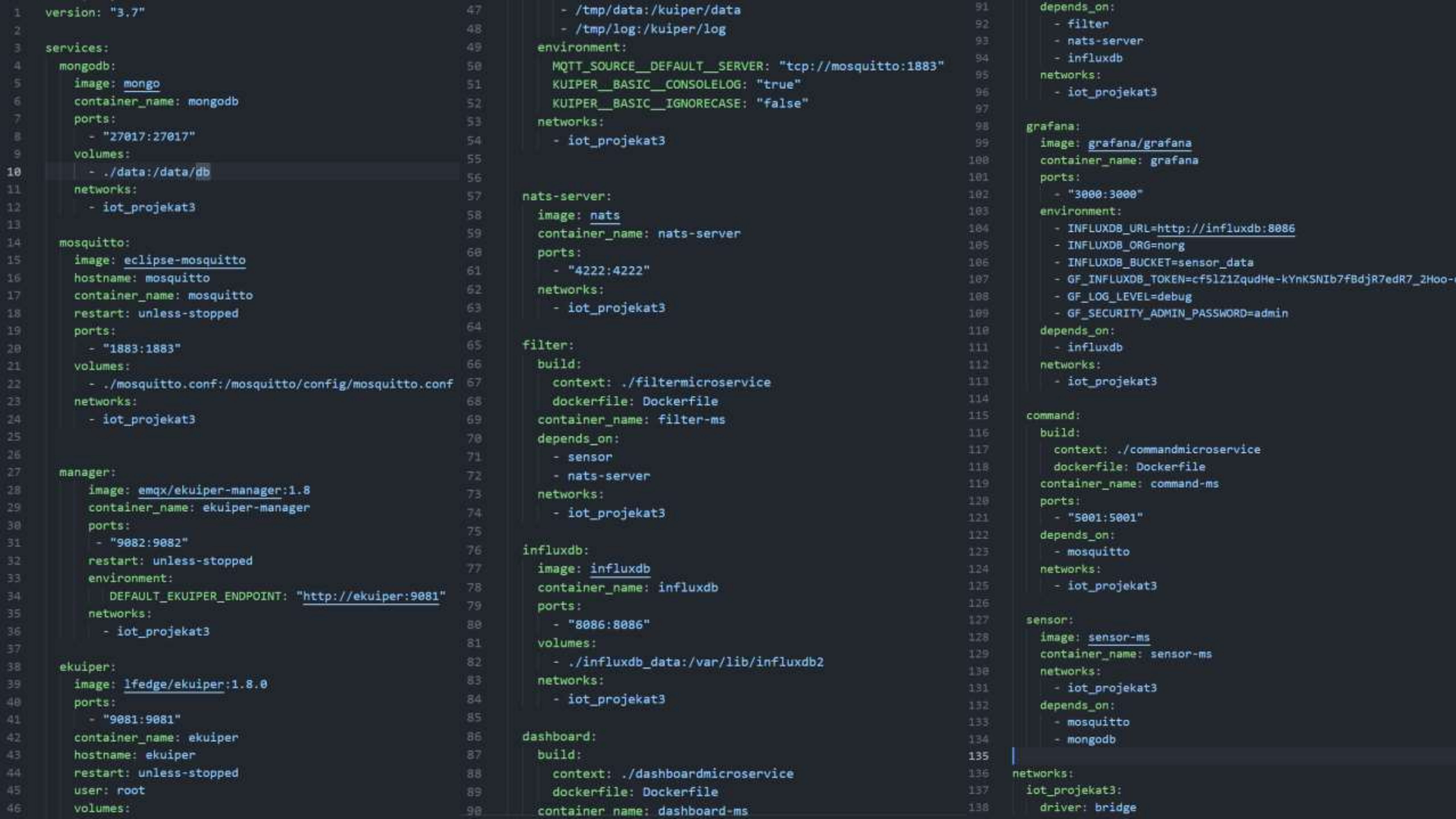
```
from(bucket: "sensor_data")
  |> range(start: -1h)  // Adjust the time range as needed
  |> filter(fn: (r) => r._measurement == "sensor_data")
  |> filter(fn: (r) => r._field == "avg_temperature")
```

B (influxdb)

```
from(bucket: "sensor_data")
  |> range(start: -1h)  // Adjust the time range as needed
  |> filter(fn: (r) => r._measurement == "sensor_data")
  |> filter(fn: (r) => r._field == "avg_humidity")
```



Docker-compose



```yaml
version: "3.7"

services:
  mongodb:
    image: mongo
    container_name: mongodb
    ports:
      - "27017:27017"
    volumes:
      - ./data:/data/db
    networks:
      - iot_projekat3

  mosquitto:
    image: eclipse-mosquitto
    hostname: mosquitto
    container_name: mosquitto
    restart: unless-stopped
    ports:
      - "1883:1883"
    volumes:
      - ./mosquitto.conf:/mosquitto/config/mosquitto.conf
    networks:
      - iot_projekat3


  manager:
      image: emqx/ekuiper-manager:1.8
      container_name: ekuiper-manager
      ports:
       - "9082:9082"
      restart: unless-stopped
      environment:
         DEFAULT_EKUIPER_ENDPOINT: "http://ekuiper:9081"
      networks:
        - iot_projekat3

  ekuiper:
    image: lfedge/ekuiper:1.8.0
    ports:
      - "9081:9081"
    container_name: ekuiper
    hostname: ekuiper
    restart: unless-stopped
    user: root
    volumes:
      - /tmp/data:/kuiper/data
      - /tmp/log:/kuiper/log
    environment:
      MQTT_SOURCE__DEFAULT__SERVER: "tcp://mosquitto:1883"
      KUIPER__BASIC__CONSOLELOG: "true"
      KUIPER__BASIC__IGNORECASE: "false"
    networks:
      - iot_projekat3



  nats-server:
    image: nats
    container_name: nats-server
    ports:
      - "4222:4222"
    networks:
      - iot_projekat3

  filter:
    build:
      context: ./filtermicroservice
      dockerfile: Dockerfile
    container_name: filter-ms
    depends_on:
      - sensor
      - nats-server
    networks:
      - iot_projekat3

  influxdb:
    image: influxdb
    container_name: influxdb
    ports:
      - "8086:8086"
    volumes:
      - ./influxdb_data:/var/lib/influxdb2
    networks:
      - iot_projekat3

  dashboard:
    build:
      context: ./dashboardmicroservice
      dockerfile: Dockerfile
    container_name: dashboard-ms
    depends_on:
      - filter
      - nats-server
      - influxdb
    networks:
      - iot_projekat3

  grafana:
    image: grafana/grafana
    container_name: grafana
    ports:
      - "3000:3000"
    environment:
      - INFLUXDB_URL=http://influxdb:8086
      - INFLUXDB_ORG=norg
      - INFLUXDB_BUCKET=sensor_data
      - GF_INFLUXDB_TOKEN=cf51Z1ZqudHe-kYnKSNIb7fBdjR7edR7_2Hoo-
      - GF_LOG_LEVEL=debug
      - GF_SECURITY_ADMIN_PASSWORD=admin
    depends_on:
      - influxdb
    networks:
      - iot_projekat3

  command:
    build:
      context: ./commandmicroservice
      dockerfile: Dockerfile
    container_name: command-ms
    ports:
      - "5001:5001"
    depends_on:
      - mosquitto
    networks:
      - iot_projekat3

  sensor:
    image: sensor-ms
    container_name: sensor-ms
    networks:
      - iot_projekat3
    depends_on:
      - mosquitto
      - mongodb

networks:
  iot_projekat3:
    driver: bridge
```

Hvala na pažnji.