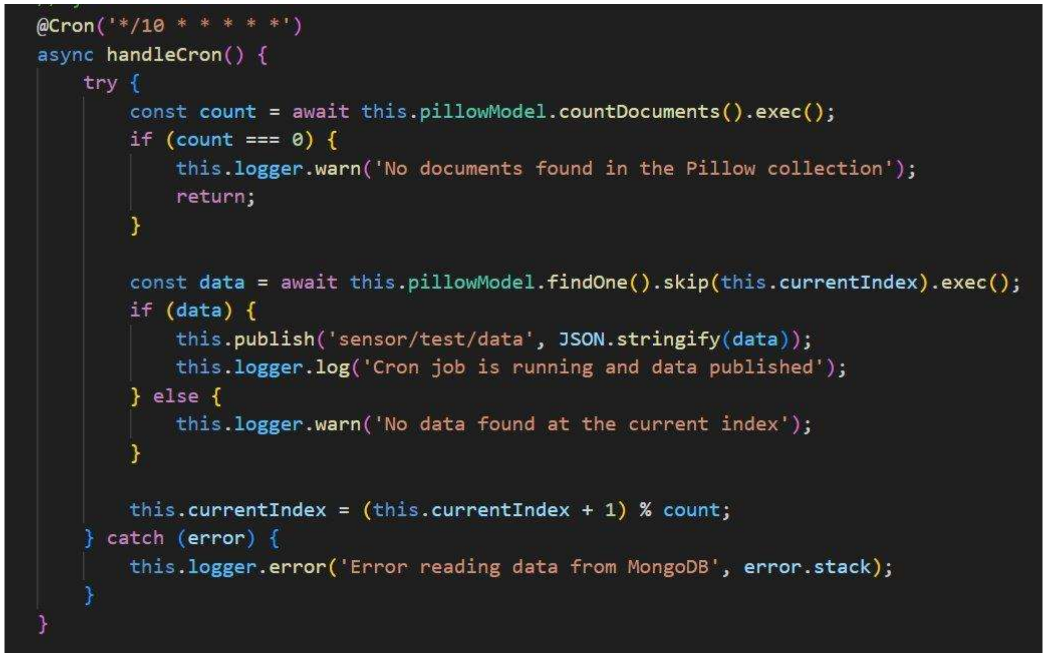
Internet stvari i servisa

- PROJEKAT 3 -

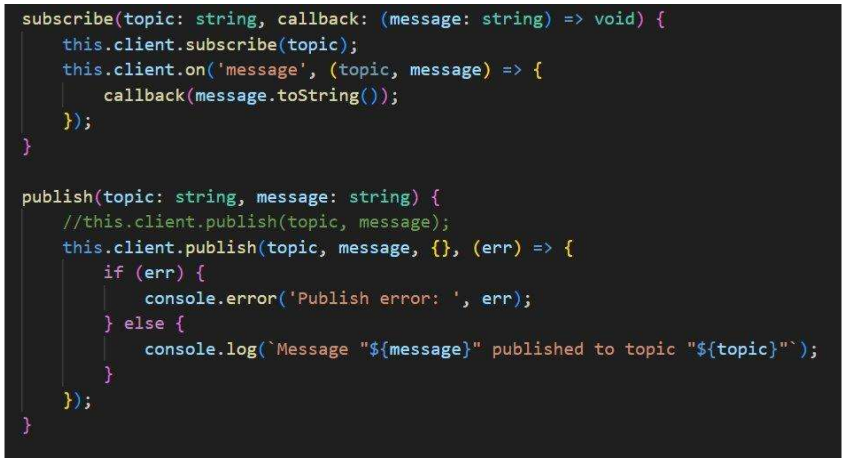
Mina Petrović 18332

Sensor

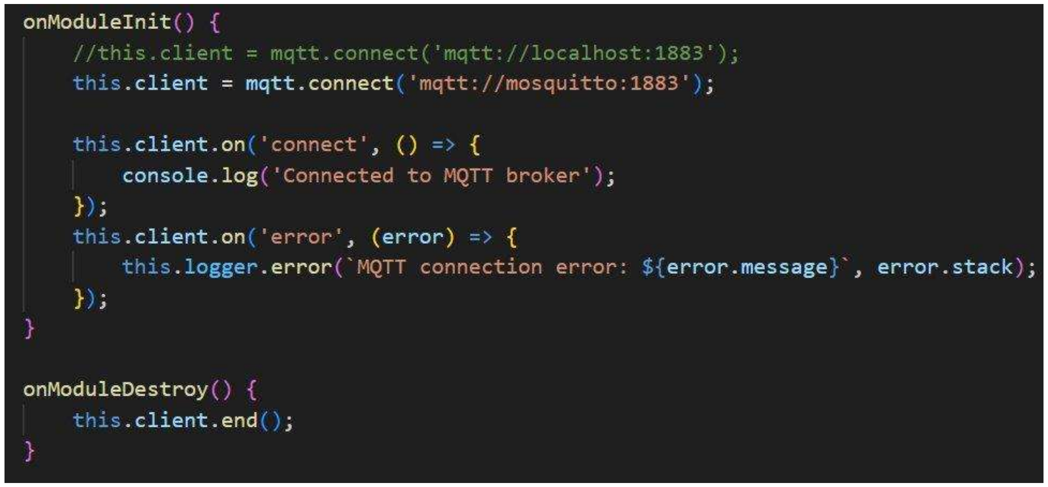
- Isti kao i prošlog puta
- NestJS servis koji čita cirkularno iz MongoDB baze pokrenute na docker kontejneru



```
@Cron('*/10 * * * * *')
async handleCron() {
    try {
        const count = await this.pillowModel.countDocuments().exec();
        if (count === 0) {
            this.logger.warn('No documents found in the Pillow collection');
            return;
        }

        const data = await this.pillowModel.findOne().skip(this.currentIndex).exec();
        if (data) {
            this.publish('sensor/test/data', JSON.stringify(data));
            this.logger.log('Cron job is running and data published');
        } else {
            this.logger.warn('No data found at the current index');
        }

        this.currentIndex = (this.currentIndex + 1) % count;
    } catch (error) {
        this.logger.error('Error reading data from MongoDB', error.stack);
    }
}
```

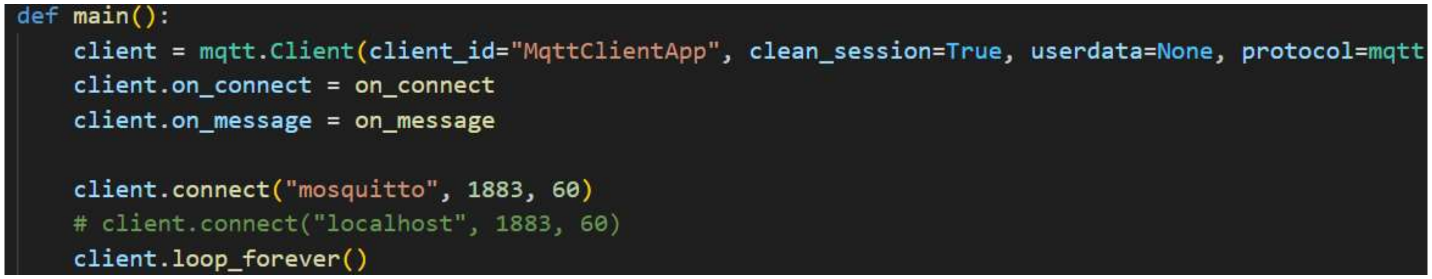


```
subscribe(topic: string, callback: (message: string) => void) {
    this.client.subscribe(topic);
    this.client.on('message', (topic, message) => {
        callback(message.toString());
    });
}

publish(topic: string, message: string) {
    //this.client.publish(topic, message);
    this.client.publish(topic, message, {}, (err) => {
        if (err) {
            console.error('Publish error: ', err);
        } else {
            console.log(`Message "${message}" published to topic "${topic}"`);
        }
    });
}
```

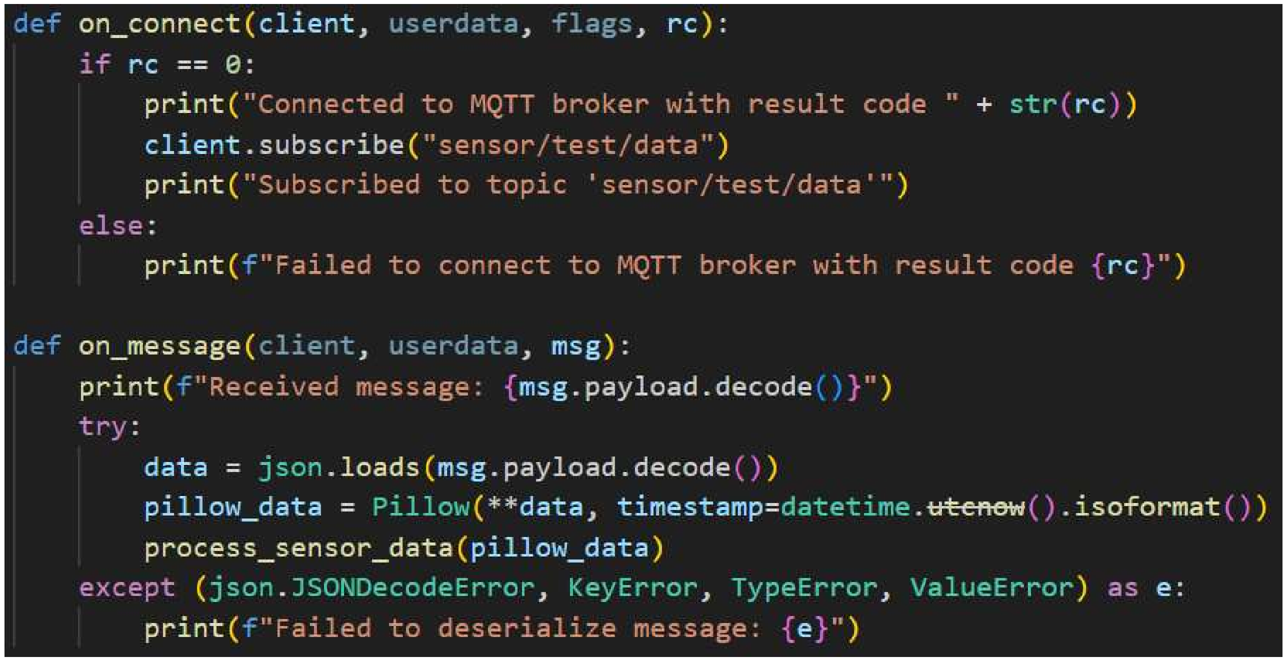


```
onModuleInit() {
    //this.client = mqtt.connect('mqtt://localhost:1883');
    this.client = mqtt.connect('mqtt://mosquitto:1883');

    this.client.on('connect', () => {
        console.log('Connected to MQTT broker');
    });
    this.client.on('error', (error) => {
        this.logger.error(`MQTT connection error: ${error.message}`, error.stack);
    });
}

onModuleDestroy() {
    this.client.end();
}
```

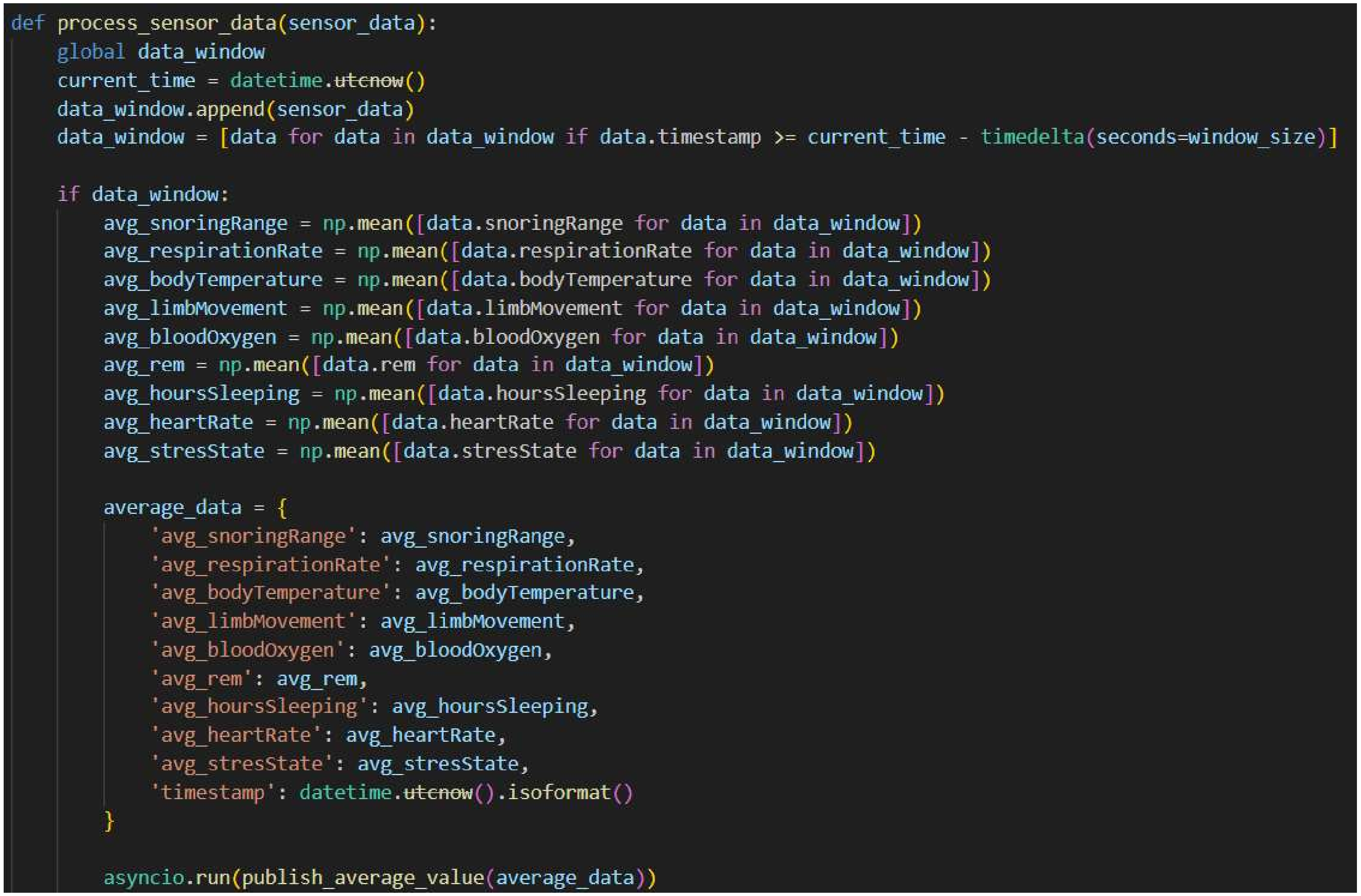
Fliter



```
def main():
    client = mqtt.Client(client_id="MqttClientApp", clean_session=True, userdata=None, protocol=mqtt
    client.on_connect = on_connect
    client.on_message = on_message

    client.connect("mosquitto", 1883, 60)
    # client.connect("localhost", 1883, 60)
    client.loop_forever()
```

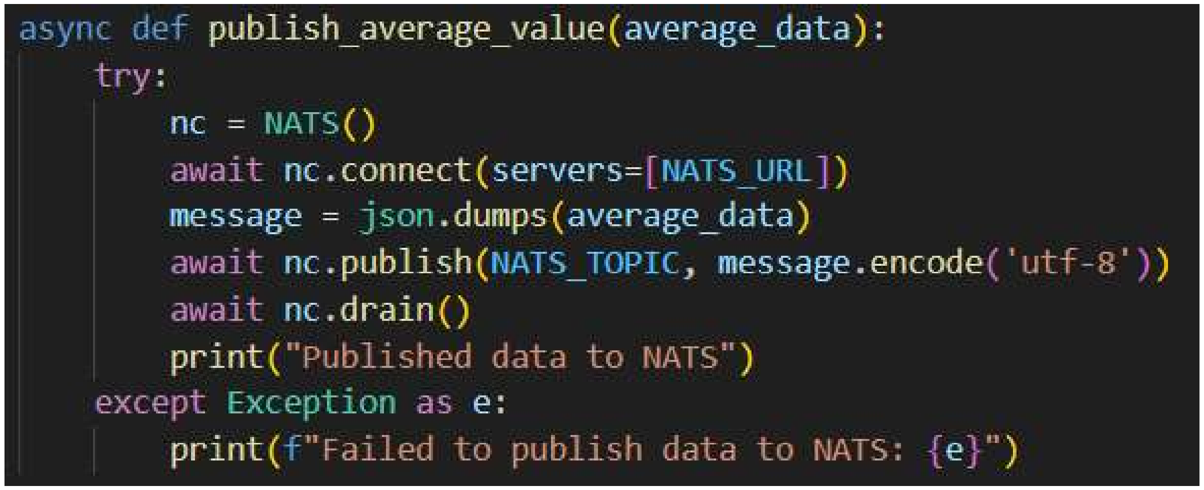


```
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker with result code " + str(rc))
        client.subscribe("sensor/test/data")
        print("Subscribed to topic 'sensor/test/data'")
    else:
        print(f"Failed to connect to MQTT broker with result code {rc}")

def on_message(client, userdata, msg):
    print(f"Received message: {msg.payload.decode()}")
    try:
        data = json.loads(msg.payload.decode())
        pillow_data = Pillow(**data, timestamp=datetime.~~utcnow~~().isoformat())
        process_sensor_data(pillow_data)
    except (json.JSONDecodeError, KeyError, TypeError, ValueError) as e:
        print(f"Failed to deserialize message: {e}")
```

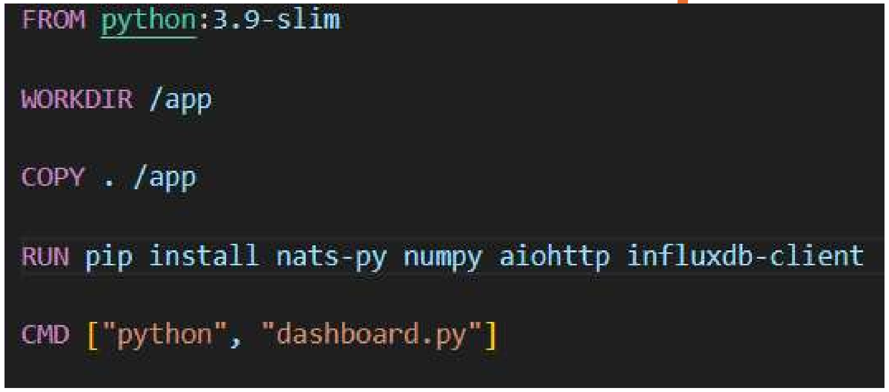


```python
def process_sensor_data(sensor_data):
    global data_window
    current_time = datetime.~~utcnow~~()
    data_window.append(sensor_data)
    data_window = [data for data in data_window if data.timestamp >= current_time - timedelta(seconds=window_size)]

    if data_window:
        avg_snoringRange = np.mean([data.snoringRange for data in data_window])
        avg_respirationRate = np.mean([data.respirationRate for data in data_window])
        avg_bodyTemperature = np.mean([data.bodyTemperature for data in data_window])
        avg_limbMovement = np.mean([data.limbMovement for data in data_window])
        avg_bloodOxygen = np.mean([data.bloodOxygen for data in data_window])
        avg_rem = np.mean([data.rem for data in data_window])
        avg_hoursSleeping = np.mean([data.hoursSleeping for data in data_window])
        avg_heartRate = np.mean([data.heartRate for data in data_window])
        avg_stresState = np.mean([data.stresState for data in data_window])

        average_data = {
            'avg_snoringRange': avg_snoringRange,
            'avg_respirationRate': avg_respirationRate,
            'avg_bodyTemperature': avg_bodyTemperature,
            'avg_limbMovement': avg_limbMovement,
            'avg_bloodOxygen': avg_bloodOxygen,
            'avg_rem': avg_rem,
            'avg_hoursSleeping': avg_hoursSleeping,
            'avg_heartRate': avg_heartRate,
            'avg_stresState': avg_stresState,
            'timestamp': datetime.~~utcnow~~().isoformat()
        }

        asyncio.run(publish_average_value(average_data))
```

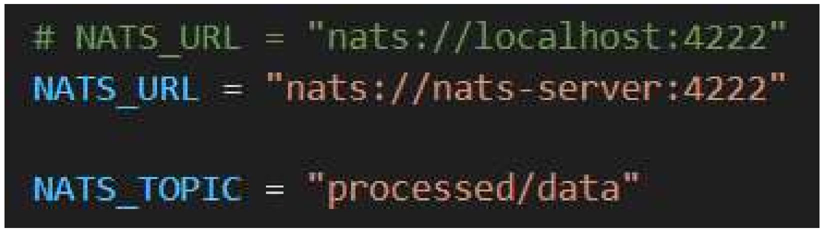
NATS
i Dockerfile



```python
async def publish_average_value(average_data):
    try:
        nc = NATS()
        await nc.connect(servers=[NATS_URL])
        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}")
```

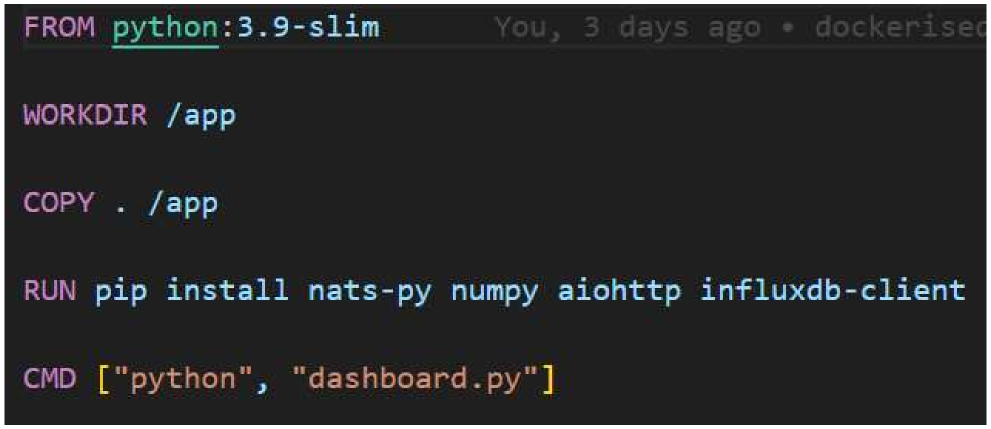


```dockerfile
FROM python:3.9-slim

WORKDIR /app

COPY . /app

RUN pip install nats-py numpy aiohttp influxdb-client

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

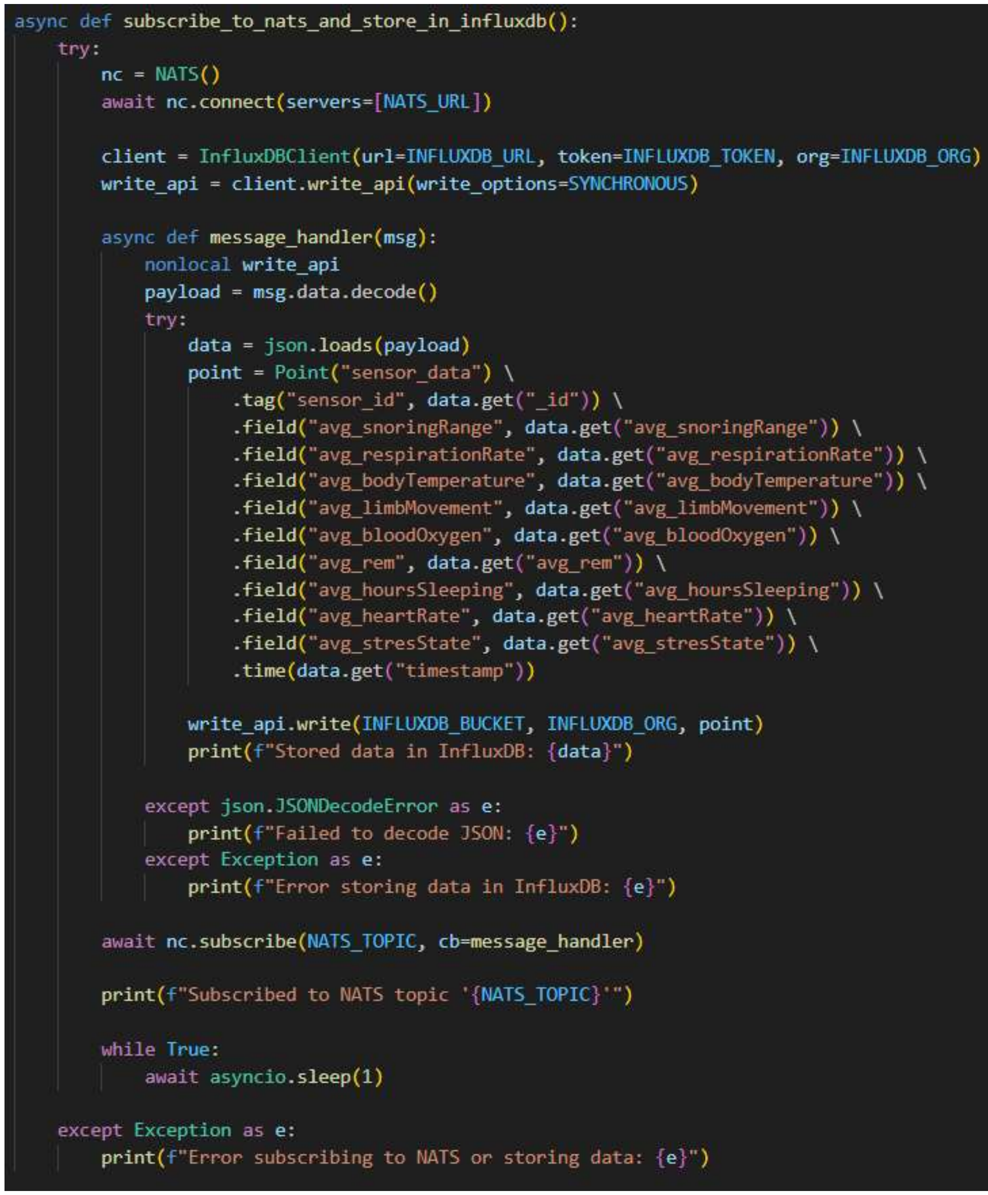


```python
# NATS_URL = "nats://localhost:4222"
NATS_URL = "nats://nats-server:4222"

NATS_TOPIC = "processed/data"
```

Dashboard

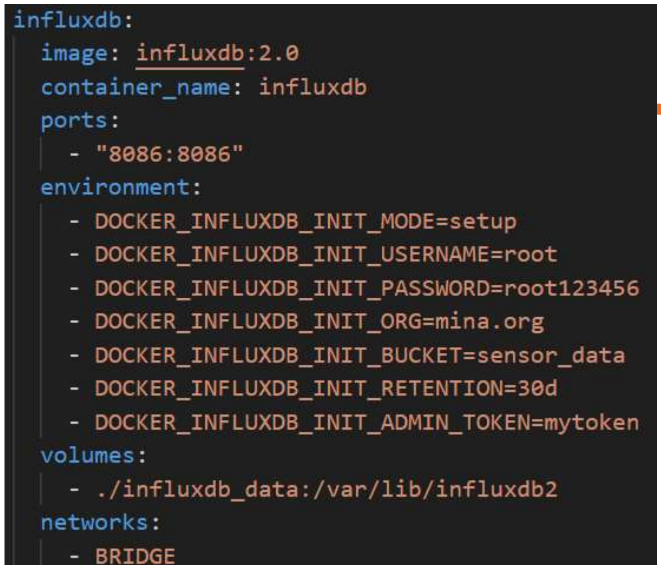


```
FROM python:3.9-slim        You, 3 days ago • dockerisec

WORKDIR /app

COPY . /app

RUN pip install nats-py numpy aiohttp influxdb-client

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

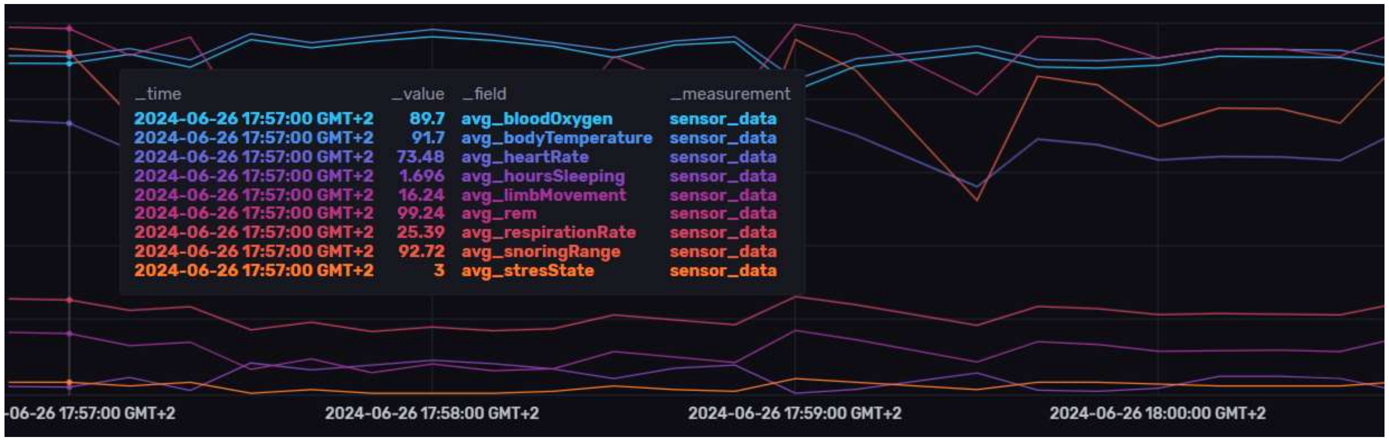


```
async def subscribe_to_nats_and_store_in_influxdb():
    try:
        nc = NATS()
        await nc.connect(servers=[NATS_URL])

        client = InfluxDBClient(url=INFLUXDB_URL, token=INFLUXDB_TOKEN, org=INFLUXDB_ORG)
        write_api = client.write_api(write_options=SYNCHRONOUS)

        async def message_handler(msg):
            nonlocal write_api
            payload = msg.data.decode()
            try:
                data = json.loads(payload)
                point = Point("sensor_data") \
                    .tag("sensor_id", data.get("_id")) \
                    .field("avg_snoringRange", data.get("avg_snoringRange")) \
                    .field("avg_respirationRate", data.get("avg_respirationRate")) \
                    .field("avg_bodyTemperature", data.get("avg_bodyTemperature")) \
                    .field("avg_limbMovement", data.get("avg_limbMovement")) \
                    .field("avg_bloodOxygen", data.get("avg_bloodOxygen")) \
                    .field("avg_rem", data.get("avg_rem")) \
                    .field("avg_hoursSleeping", data.get("avg_hoursSleeping")) \
                    .field("avg_heartRate", data.get("avg_heartRate")) \
                    .field("avg_stresState", data.get("avg_stresState")) \
                    .time(data.get("timestamp"))

                write_api.write(INFLUXDB_BUCKET, INFLUXDB_ORG, point)
                print(f"Stored data in InfluxDB: {data}")

            except json.JSONDecodeError as e:
                print(f"Failed to decode JSON: {e}")
            except Exception as e:
                print(f"Error storing data in InfluxDB: {e}")

        await nc.subscribe(NATS_TOPIC, cb=message_handler)

        print(f"Subscribed to NATS topic '{NATS_TOPIC}'")

        while True:
            await asyncio.sleep(1)

    except Exception as e:
        print(f"Error subscribing to NATS or storing data: {e}")
```

InfluxDB

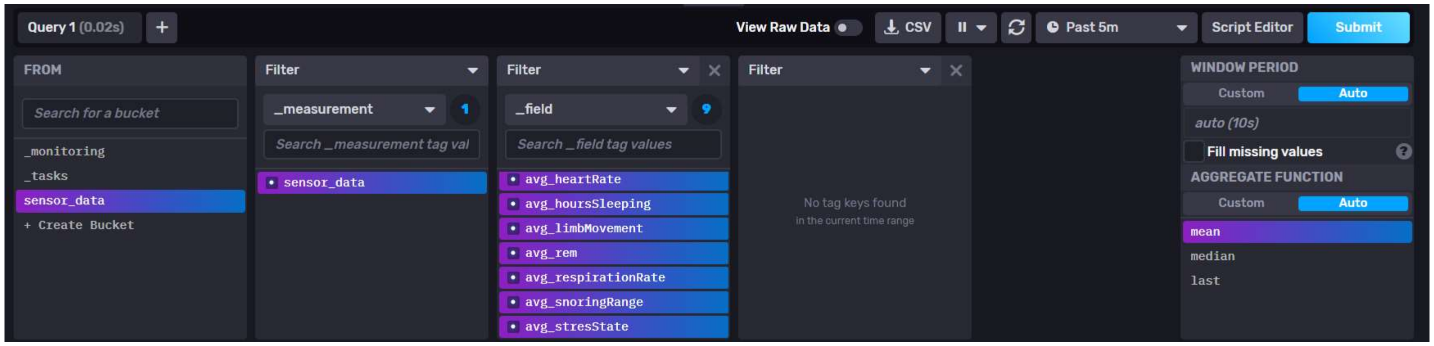


```
influxdb:
  image: influxdb:2.0
  container_name: influxdb
  ports:
    - "8086:8086"
  environment:
    - DOCKER_INFLUXDB_INIT_MODE=setup
    - DOCKER_INFLUXDB_INIT_USERNAME=root
    - DOCKER_INFLUXDB_INIT_PASSWORD=root123456
    - DOCKER_INFLUXDB_INIT_ORG=mina.org
    - DOCKER_INFLUXDB_INIT_BUCKET=sensor_data
    - DOCKER_INFLUXDB_INIT_RETENTION=30d
    - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=mytoken
  volumes:
    - ./influxdb_data:/var/lib/influxdb2
  networks:
    - BRIDGE
```



```
_time                        _value  _field                  _measurement
2024-06-26 17:57:00 GMT+2    89.7    avg_bloodOxygen         sensor_data
2024-06-26 17:57:00 GMT+2    91.7    avg_bodyTemperature     sensor_data
2024-06-26 17:57:00 GMT+2    73.48   avg_heartRate           sensor_data
2024-06-26 17:57:00 GMT+2    1.696   avg_hoursSleeping       sensor_data
2024-06-26 17:57:00 GMT+2    16.24   avg_limbMovement        sensor_data
2024-06-26 17:57:00 GMT+2    99.24   avg_rem                 sensor_data
2024-06-26 17:57:00 GMT+2    25.39   avg_respirationRate     sensor_data
2024-06-26 17:57:00 GMT+2    92.72   avg_snoringRange        sensor_data
2024-06-26 17:57:00 GMT+2    3       avg_stresState          sensor_data
```

-06-26 17:57:00 GMT+2
2024-06-26 17:58:00 GMT+2
2024-06-26 17:59:00 GMT+2
2024-06-26 18:00:00 GMT+2



Query 1 (0.02s) | View Raw Data | CSV | Past 5m | Script Editor | Submit

FROM: Search for a bucket — _monitoring, _tasks, **sensor_data** (selected), + Create Bucket

Filter: _measurement (1) — Search _measurement tag val — **sensor_data** (selected)

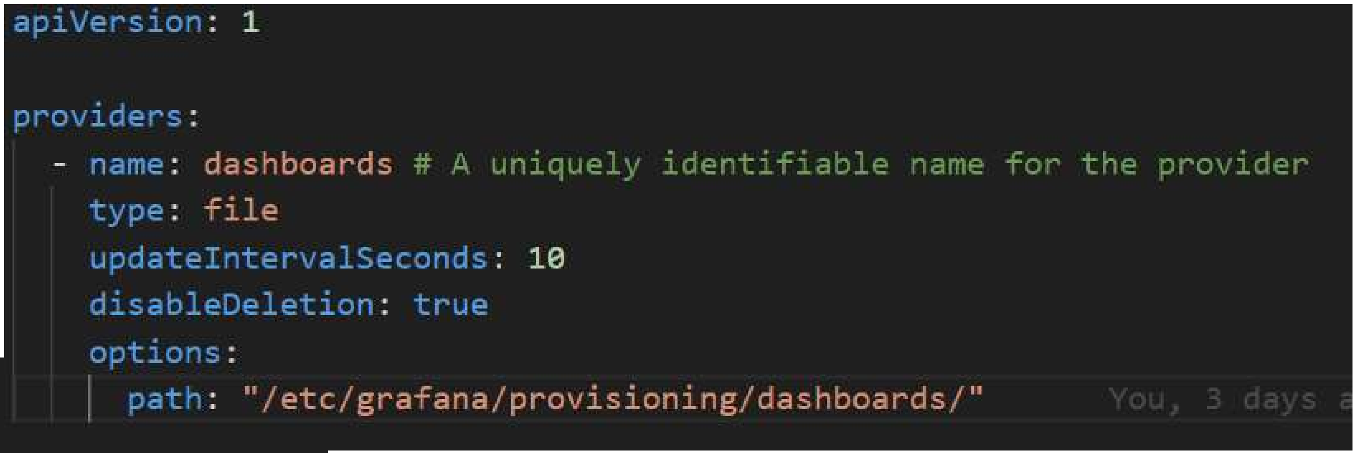
Filter: _field (9) — Search _field tag values — avg_heartRate, avg_hoursSleeping, avg_limbMovement, avg_rem, avg_respirationRate, avg_snoringRange, avg_stresState (selected)

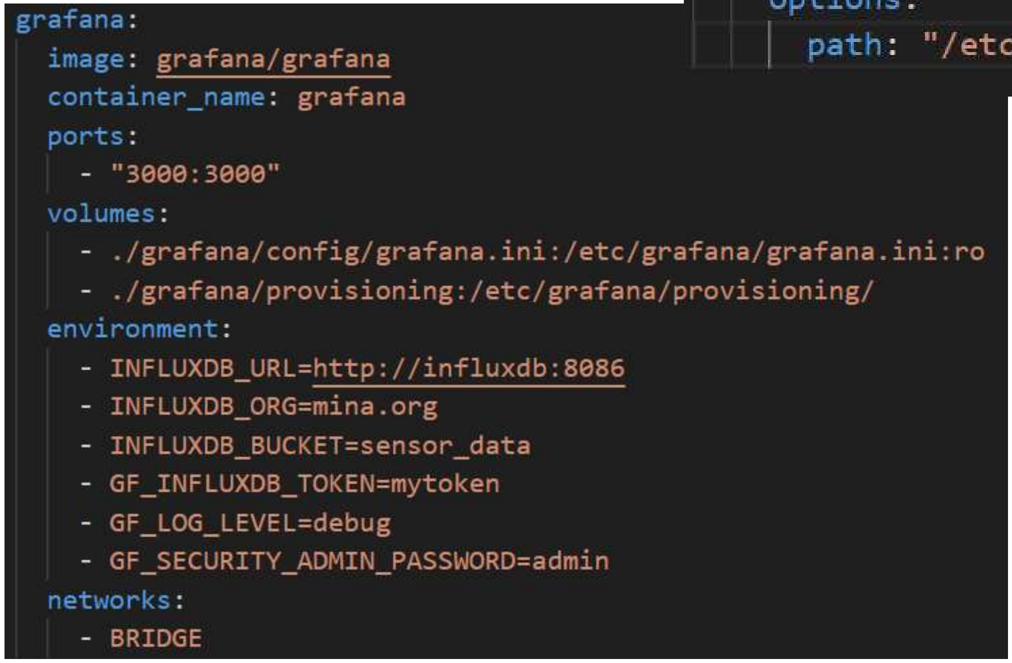
Filter: No tag keys found in the current time range

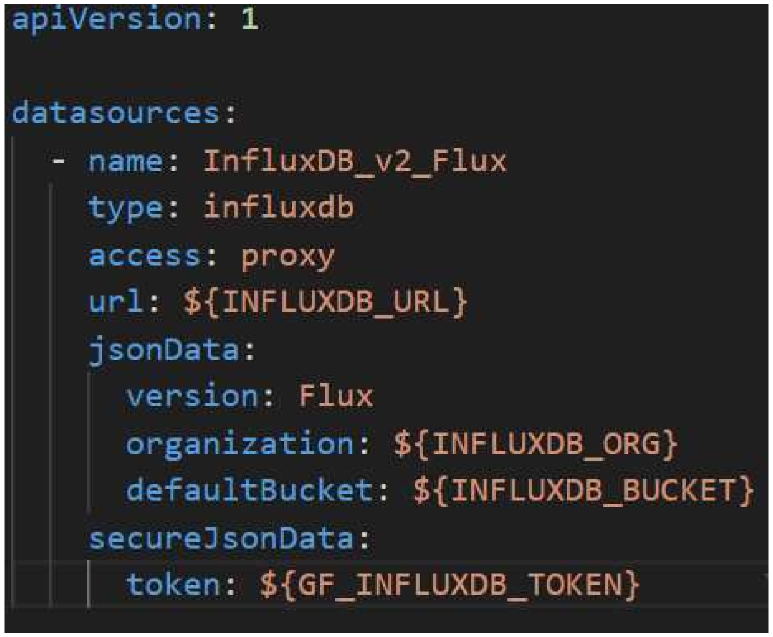
WINDOW PERIOD: Custom / **Auto** — auto (10s) — Fill missing values

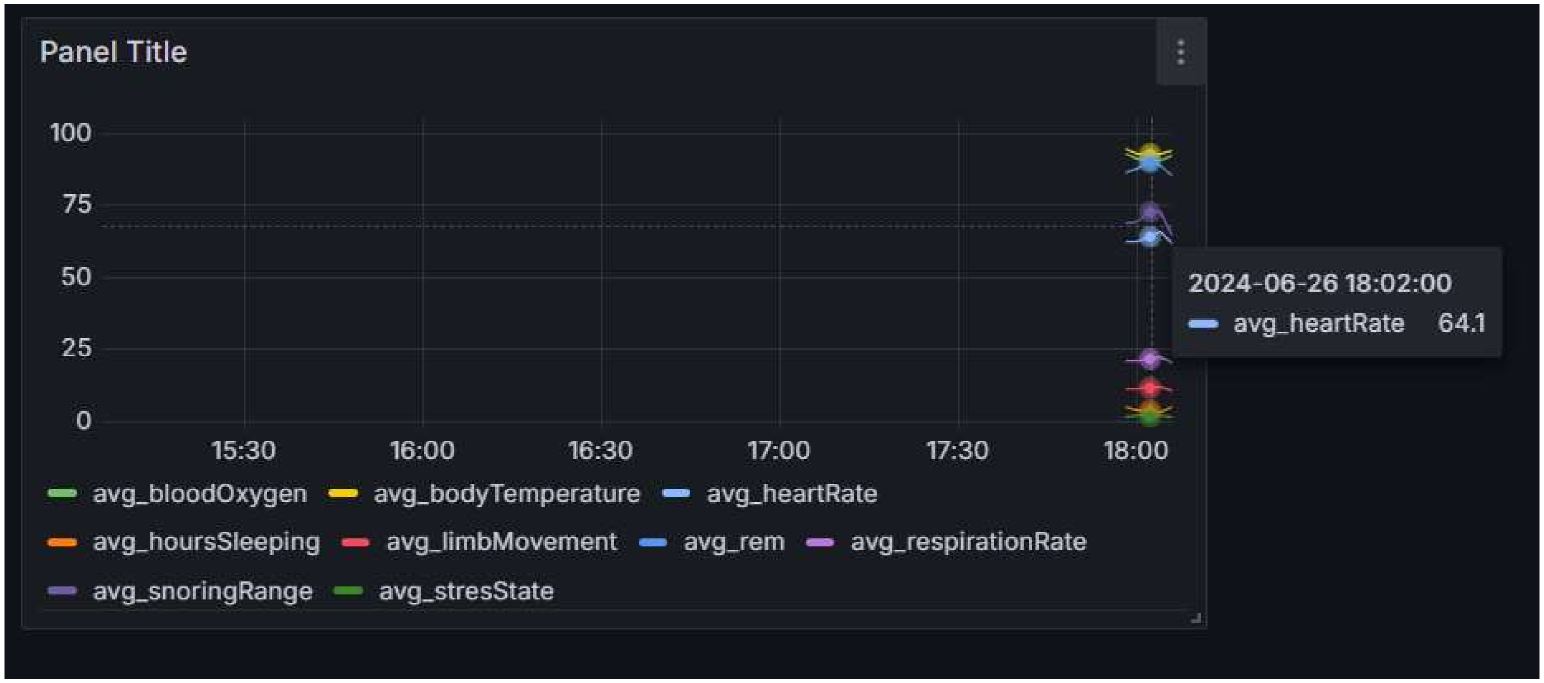
AGGREGATE FUNCTION: Custom / **Auto** — **mean** (selected), median, last

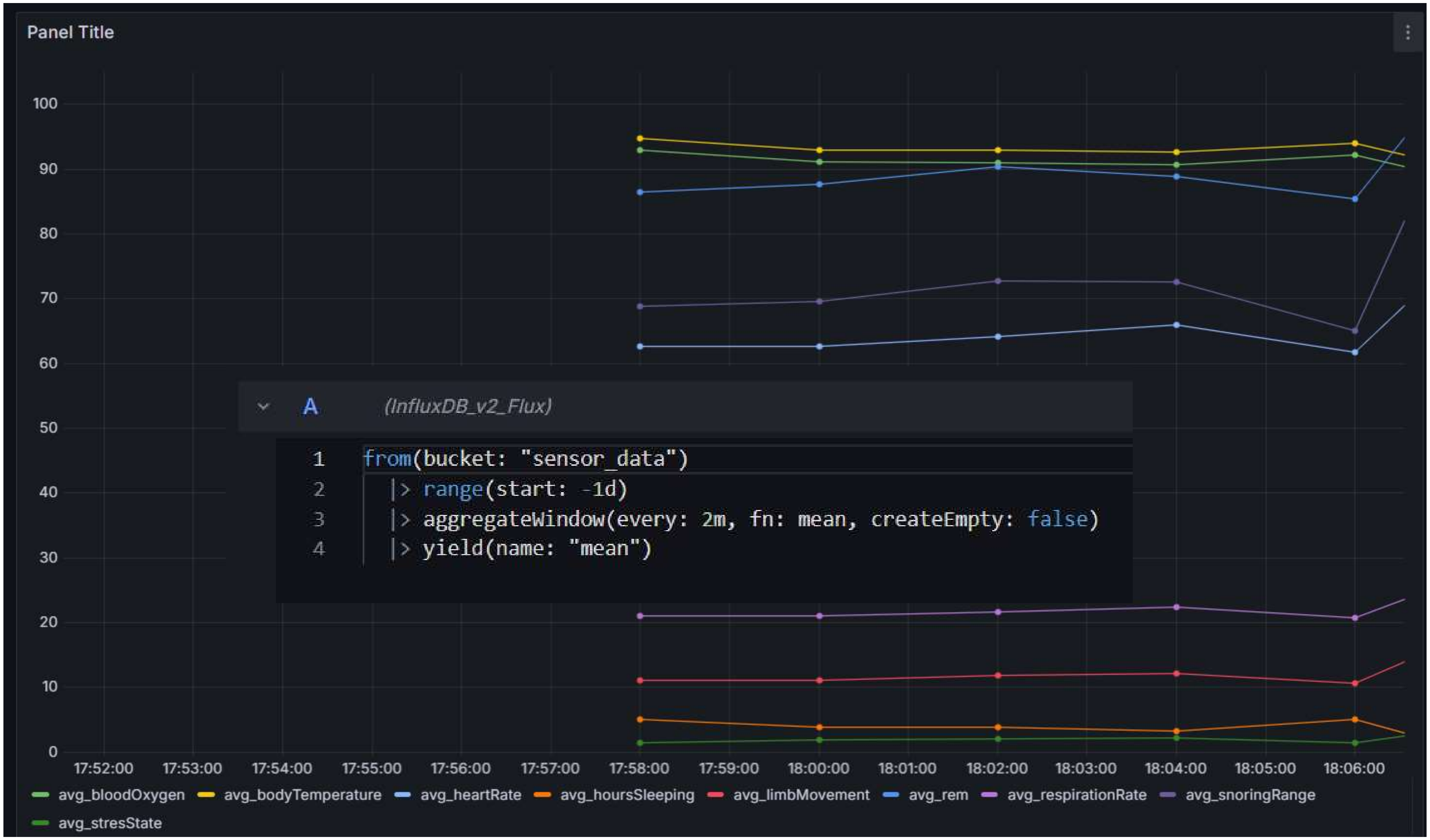
Grafana


```
apiVersion: 1

providers:
  - name: dashboards # A uniquely identifiable name for the provider
    type: file
    updateIntervalSeconds: 10
    disableDeletion: true
    options:
      path: "/etc/grafana/provisioning/dashboards/"       You, 3 days a
```


```
grafana:
  image: grafana/grafana
  container_name: grafana
  ports:
    - "3000:3000"
  volumes:
    - ./grafana/config/grafana.ini:/etc/grafana/grafana.ini:ro
    - ./grafana/provisioning:/etc/grafana/provisioning/
  environment:
    - INFLUXDB_URL=http://influxdb:8086
    - INFLUXDB_ORG=mina.org
    - INFLUXDB_BUCKET=sensor_data
    - GF_INFLUXDB_TOKEN=mytoken
    - GF_LOG_LEVEL=debug
    - GF_SECURITY_ADMIN_PASSWORD=admin
  networks:
    - BRIDGE
```

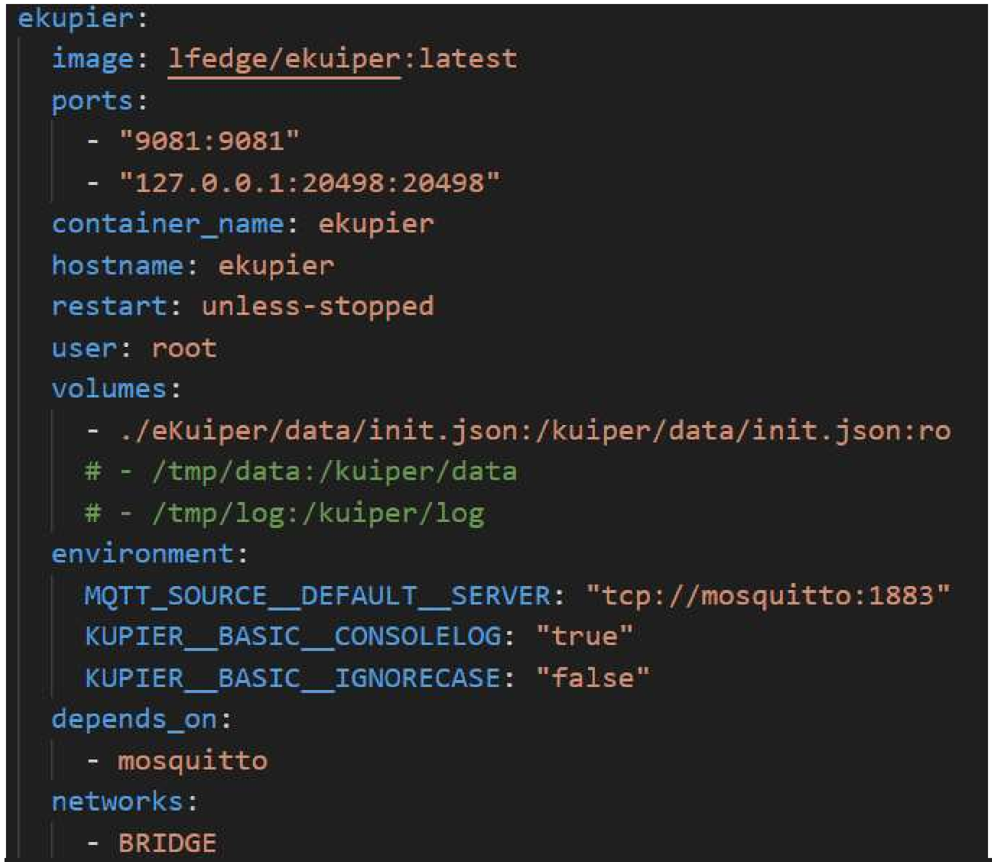

```
apiVersion: 1

datasources:
  - name: InfluxDB_v2_Flux
    type: influxdb
    access: proxy
    url: ${INFLUXDB_URL}
    jsonData:
      version: Flux
      organization: ${INFLUXDB_ORG}
      defaultBucket: ${INFLUXDB_BUCKET}
    secureJsonData:
      token: ${GF_INFLUXDB_TOKEN}
```


Panel Title
100
75
50
25
0
15:30
16:00
16:30
17:00
17:30
18:00
2024-06-26 18:02:00
avg_heartRate 64.1
avg_bloodOxygen
avg_bodyTemperature
avg_heartRate
avg_hoursSleeping
avg_limbMovement
avg_rem
avg_respirationRate
avg_snoringRange
avg_stresState

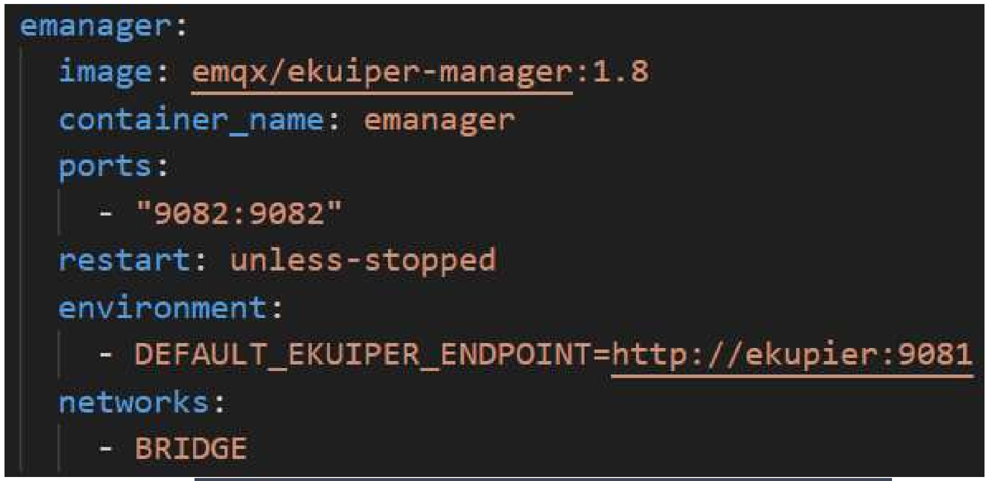


```
from(bucket: "sensor_data")
  |> range(start: -1d)
  |> aggregateWindow(every: 2m, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

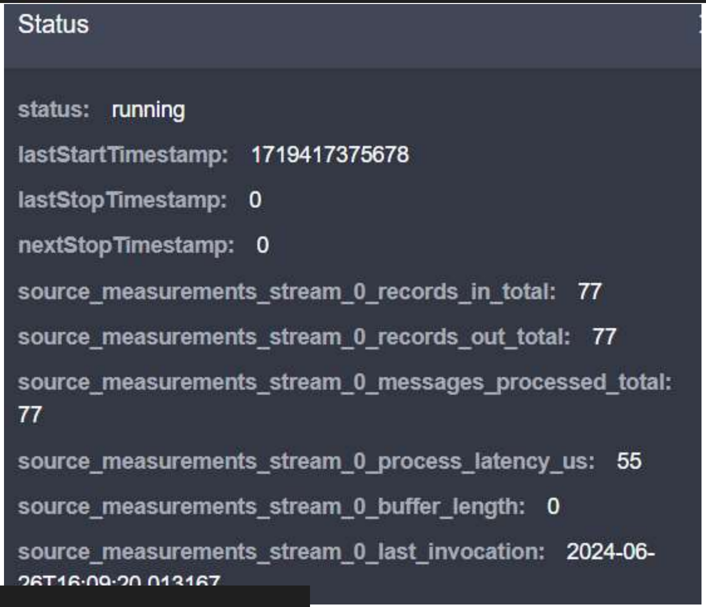
eKuiper



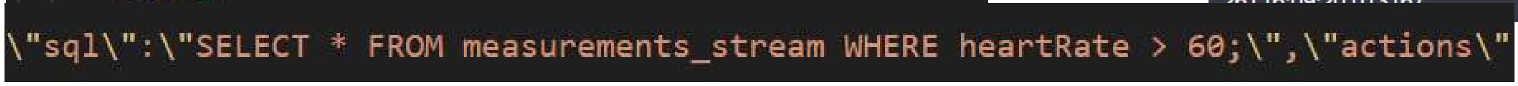
```yaml
ekupier:
  image: lfedge/ekuiper:latest
  ports:
    - "9081:9081"
    - "127.0.0.1:20498:20498"
  container_name: ekupier
  hostname: ekupier
  restart: unless-stopped
  user: root
  volumes:
    - ./eKuiper/data/init.json:/kuiper/data/init.json:ro
    # - /tmp/data:/kuiper/data
    # - /tmp/log:/kuiper/log
  environment:
    MQTT_SOURCE__DEFAULT__SERVER: "tcp://mosquitto:1883"
    KUPIER__BASIC__CONSOLELOG: "true"
    KUPIER__BASIC__IGNORECASE: "false"
  depends_on:
    - mosquitto
  networks:
    - BRIDGE
```



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

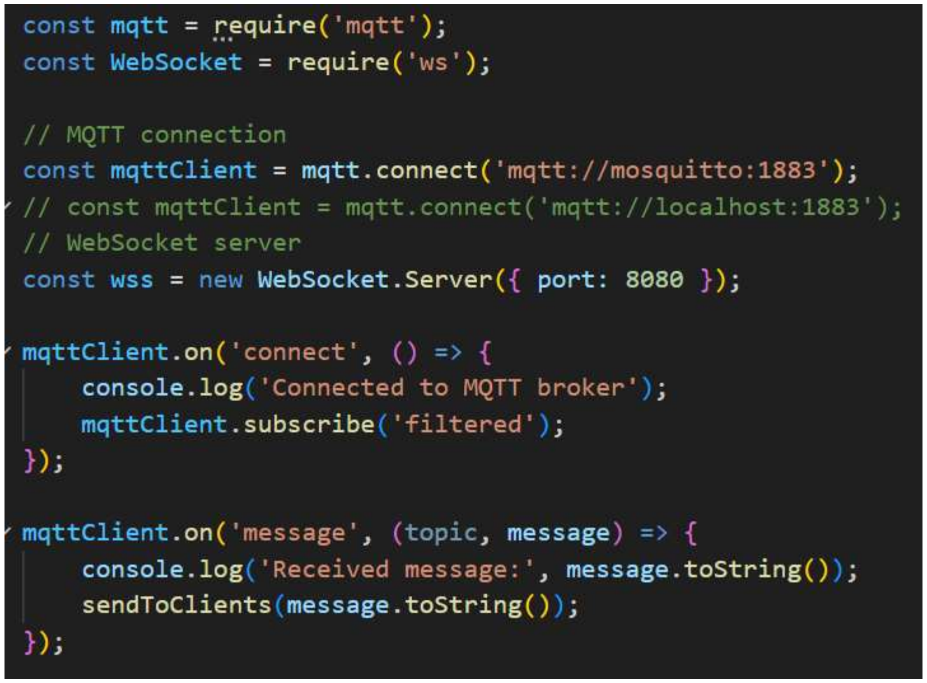


```
Status

status:   running
lastStartTimestamp:   1719417375678
lastStopTimestamp:   0
nextStopTimestamp:   0
source_measurements_stream_0_records_in_total:   77
source_measurements_stream_0_records_out_total:   77
source_measurements_stream_0_messages_processed_total:
77
source_measurements_stream_0_process_latency_us:   55
source_measurements_stream_0_buffer_length:   0
source_measurements_stream_0_last_invocation:   2024-06-
26T16:09:20.013167
```

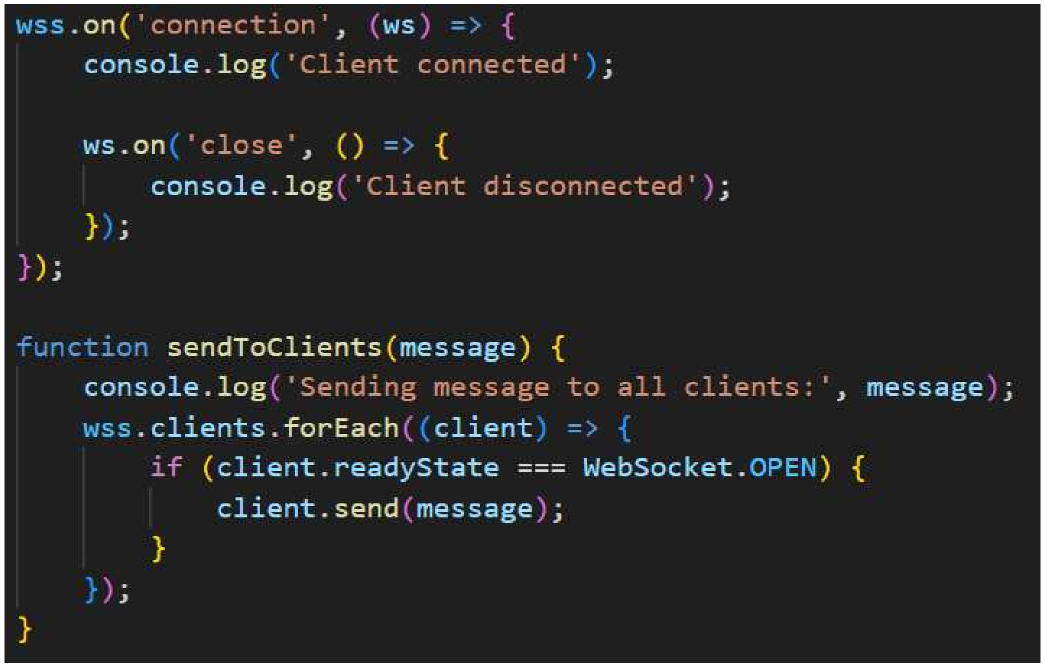


```
\"sql\":\"SELECT * FROM measurements_stream WHERE heartRate > 60;\",\"actions\"
```

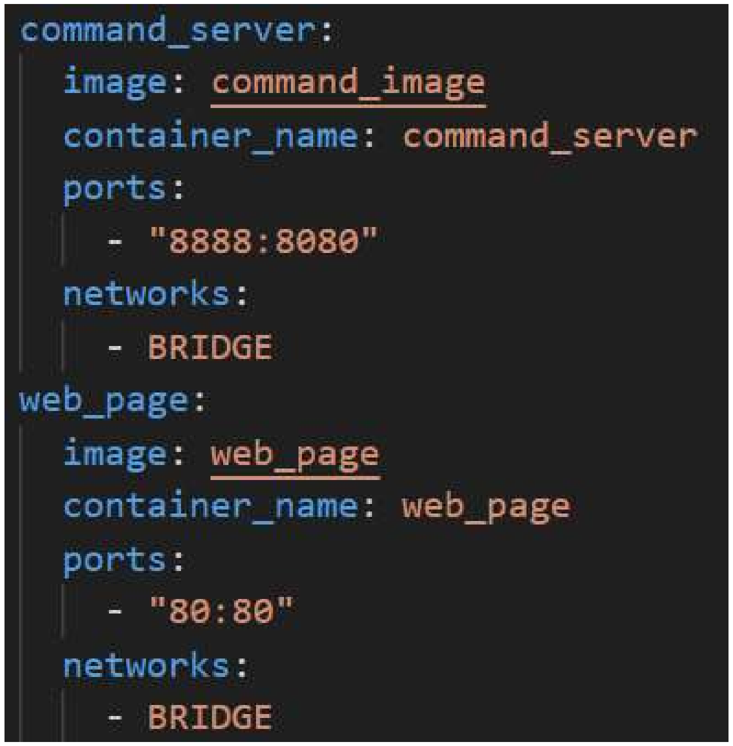
Command



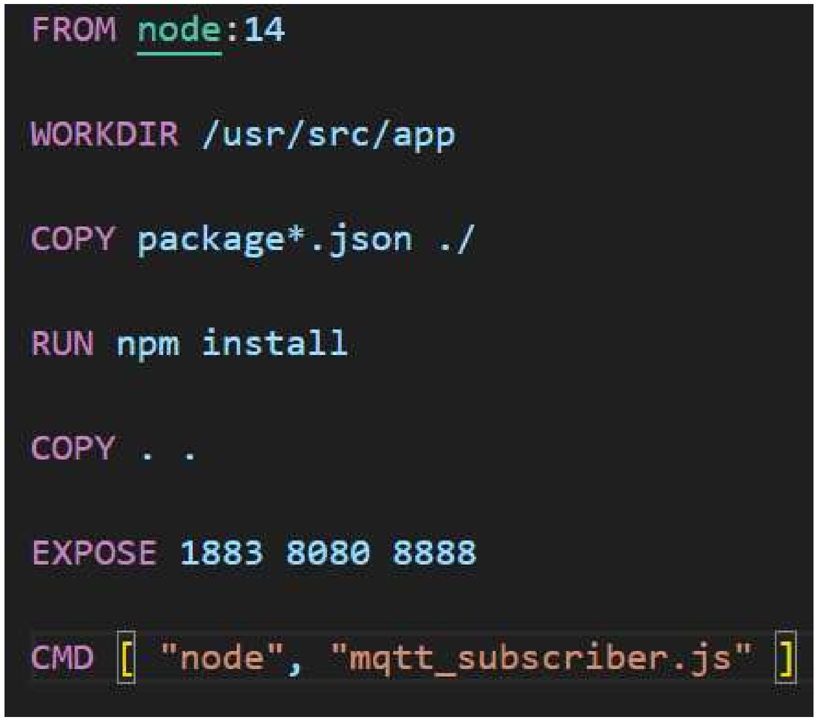
```javascript
const mqtt = require('mqtt');
const WebSocket = require('ws');

// MQTT connection
const mqttClient = mqtt.connect('mqtt://mosquitto:1883');
// const mqttClient = mqtt.connect('mqtt://localhost:1883');
// WebSocket server
const wss = new WebSocket.Server({ port: 8080 });

mqttClient.on('connect', () => {
    console.log('Connected to MQTT broker');
    mqttClient.subscribe('filtered');
});

mqttClient.on('message', (topic, message) => {
    console.log('Received message:', message.toString());
    sendToClients(message.toString());
});
```



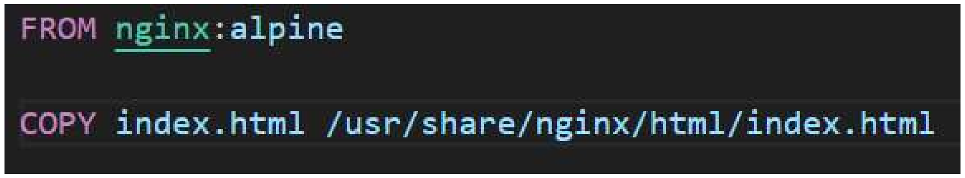
```javascript
wss.on('connection', (ws) => {
    console.log('Client connected');

    ws.on('close', () => {
        console.log('Client disconnected');
    });
});

function sendToClients(message) {
    console.log('Sending message to all clients:', message);
    wss.clients.forEach((client) => {
        if (client.readyState === WebSocket.OPEN) {
            client.send(message);
        }
    });
}
```



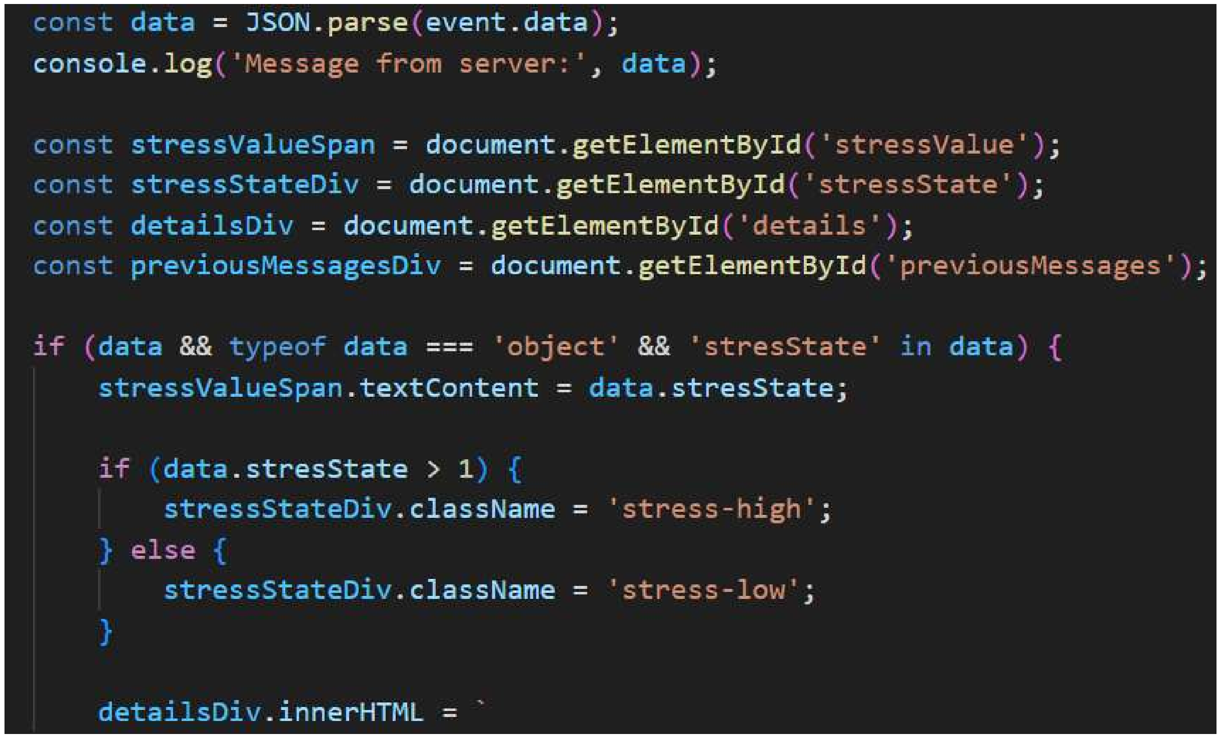
```yaml
command_server:
  image: command_image
  container_name: command_server
  ports:
    - "8888:8080"
  networks:
    - BRIDGE
web_page:
  image: web_page
  container_name: web_page
  ports:
    - "80:80"
  networks:
    - BRIDGE
```



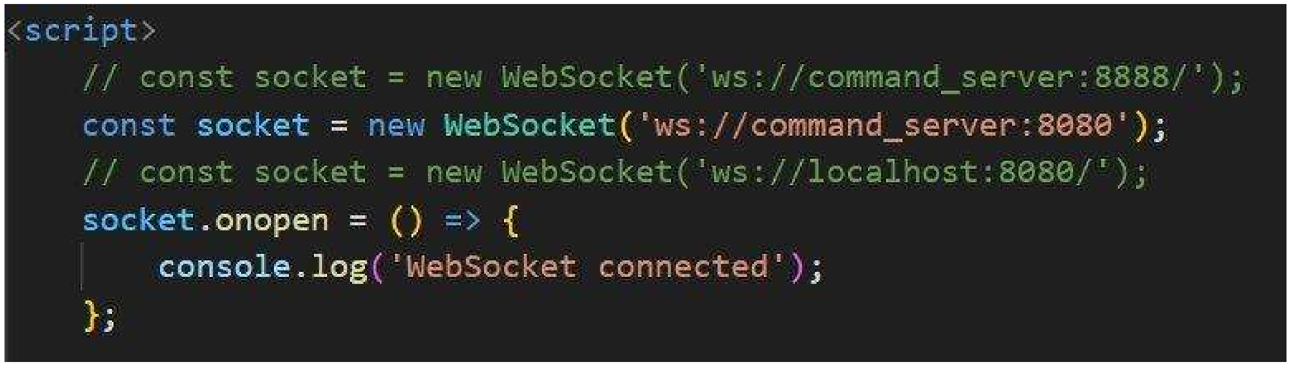
```dockerfile
FROM node:14

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 1883 8080 8888

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



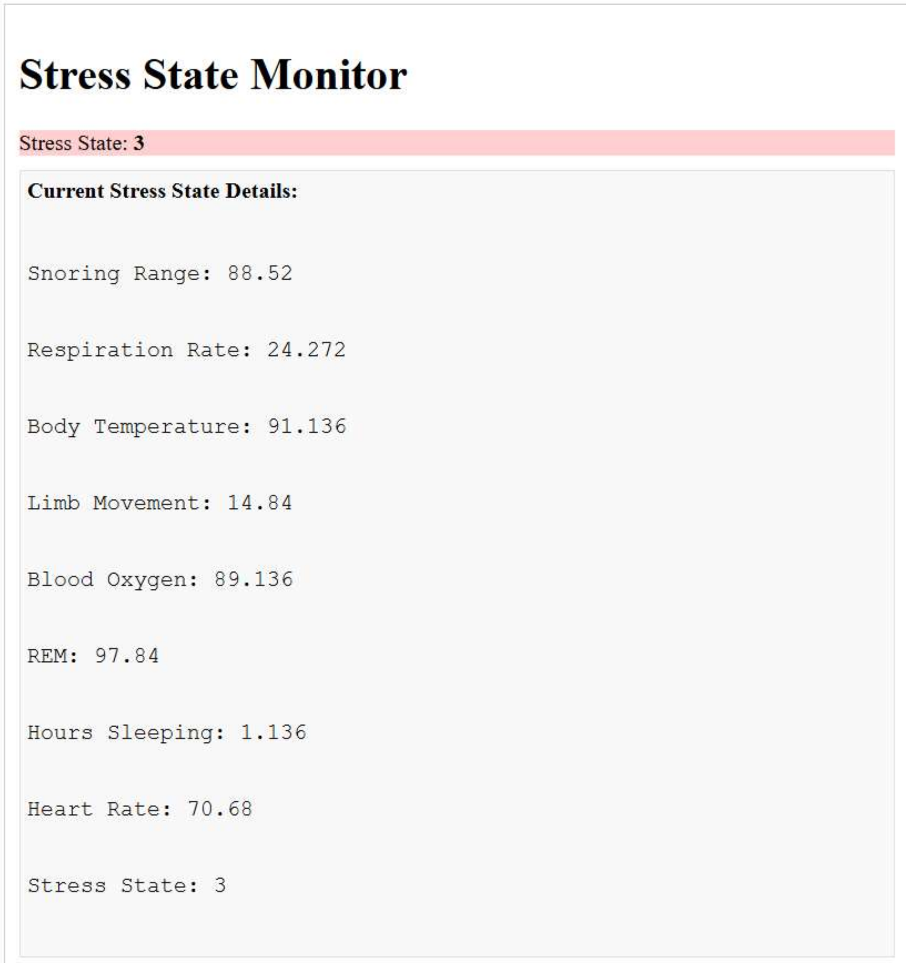
```dockerfile
FROM nginx:alpine

COPY index.html /usr/share/nginx/html/index.html
```



```
const data = JSON.parse(event.data);
console.log('Message from server:', data);

const stressValueSpan = document.getElementById('stressValue');
const stressStateDiv = document.getElementById('stressState');
const detailsDiv = document.getElementById('details');
const previousMessagesDiv = document.getElementById('previousMessages');

if (data && typeof data === 'object' && 'stresState' in data) {
    stressValueSpan.textContent = data.stresState;

    if (data.stresState > 1) {
        stressStateDiv.className = 'stress-high';
    } else {
        stressStateDiv.className = 'stress-low';
    }

    detailsDiv.innerHTML = `
```



```
<script>
    // const socket = new WebSocket('ws://command_server:8888/');
    const socket = new WebSocket('ws://command_server:8080');
    // const socket = new WebSocket('ws://localhost:8080/');
    socket.onopen = () => {
        console.log('WebSocket connected');
    };
```

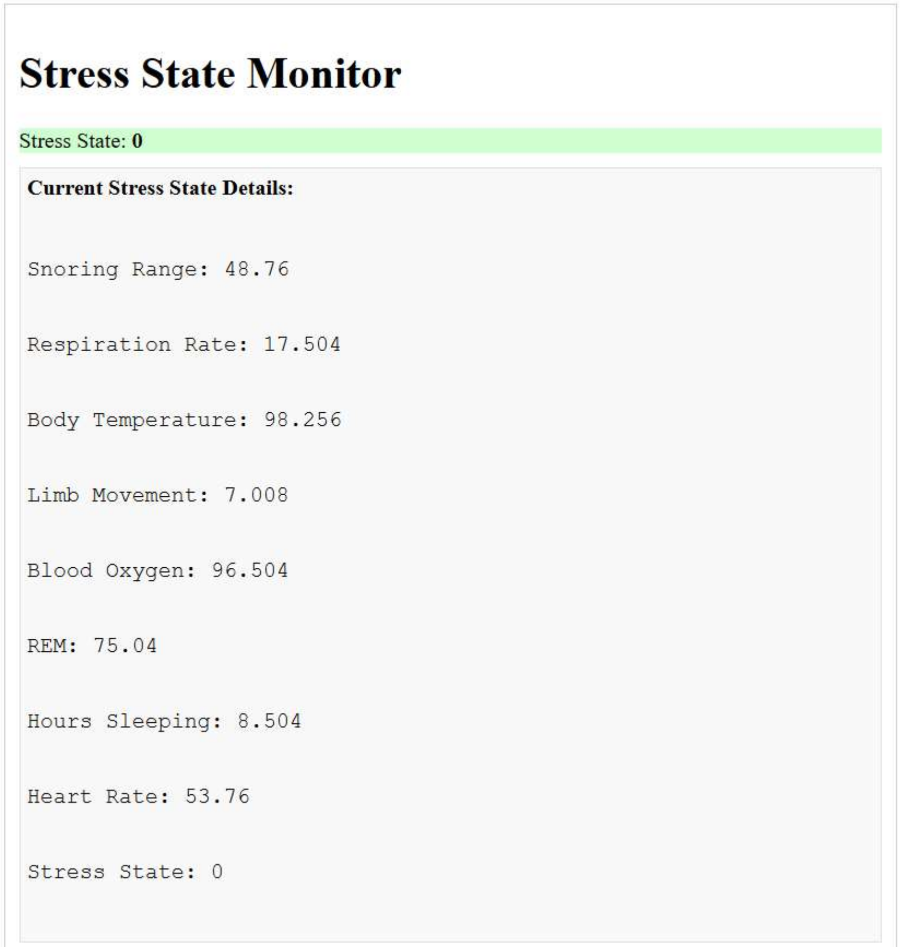


Stress State Monitor

Stress State: **3**

Current Stress State Details:

```
Snoring Range: 88.52

Respiration Rate: 24.272

Body Temperature: 91.136

Limb Movement: 14.84

Blood Oxygen: 89.136

REM: 97.84

Hours Sleeping: 1.136

Heart Rate: 70.68

Stress State: 3
```

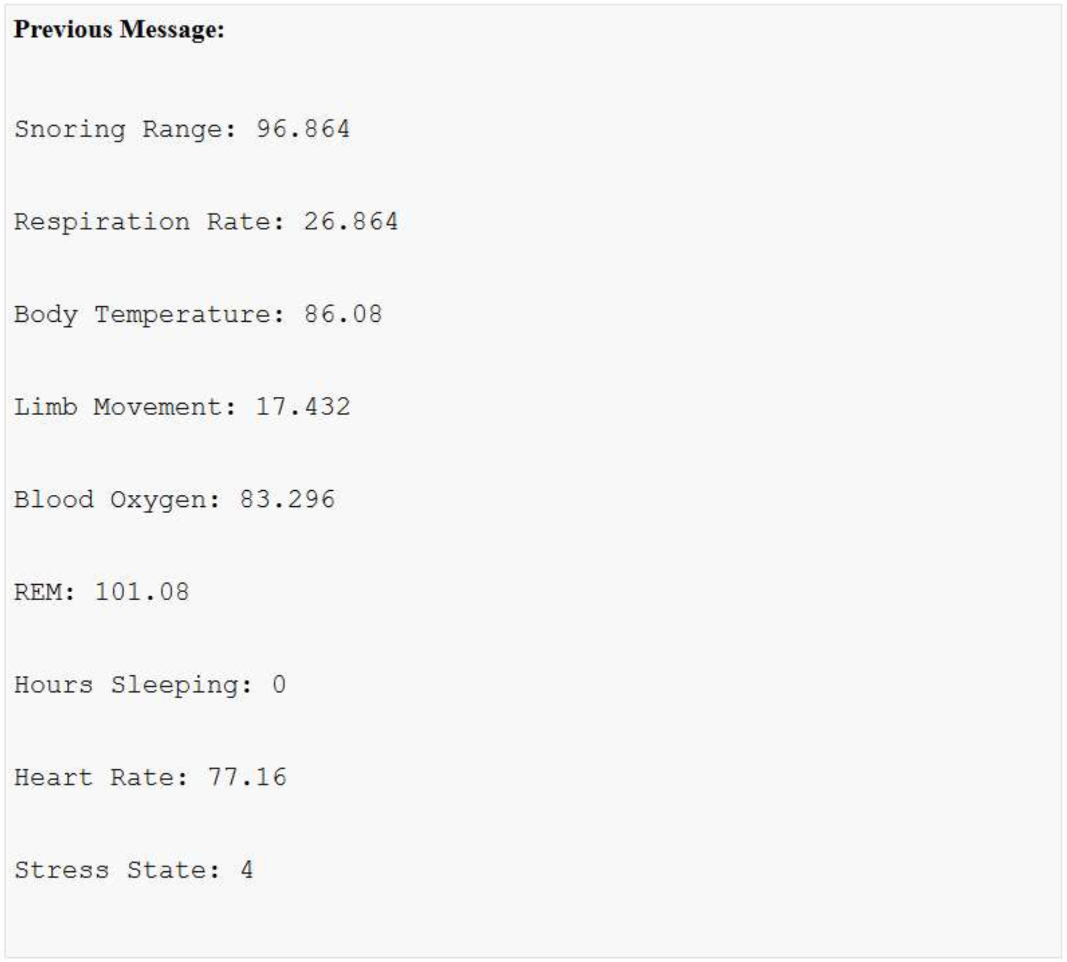


Stress State Monitor

Stress State: **0**

Current Stress State Details:

```
Snoring Range: 48.76

Respiration Rate: 17.504

Body Temperature: 98.256

Limb Movement: 7.008

Blood Oxygen: 96.504

REM: 75.04

Hours Sleeping: 8.504

Heart Rate: 53.76

Stress State: 0
```



Previous Message:

```
Snoring Range: 96.864

Respiration Rate: 26.864

Body Temperature: 86.08

Limb Movement: 17.432

Blood Oxygen: 83.296

REM: 101.08

Hours Sleeping: 0

Heart Rate: 77.16

Stress State: 4
```