

REAL-TIME SENSOR DATA ANALYSIS PROJECT PROCESS STEPS

Project Purpose: This project aims to process and visualize real-time data from office building sensors (temperature, humidity, CO2, motion, etc.) and make ML predictions.

1. PROJECT STRUCTURE

Go to main directory

```
cd /home/train/dataops11/spark_class
```

Create project directory (If not exists)

```
mkdir -p spark_class
```

Create subdirectories (If not exists)

```
mkdir python_files
```

```
mkdir -p data/KETI
```

2. CONTAINER SETUP

Go to main directory

```
cd /home/train/dataops11/spark_class
```

Clean old services if exist

```
docker-compose down --volumes
```

Start new services

```
docker-compose up -d
```

Check service status

```
docker ps
```

Memory Map Limit Setting

```
sudo sysctl -w vm.max_map_count=262144
```

Description: Creating required directory structure for Docker containers.

3. PACKAGE INSTALLATION

Required packages for preprocess_data.py

```
docker exec -it spark_client pip3 install colorama tqdm
```

Required packages for dataframe_to_kafka_final.py

```
docker exec -it kafka pip3 install tqdm colorama
```

Required packages for spark processing and ML scripts

```
docker exec -it spark_client pip3 install colorama
```

Downgrade Elasticsearch client version

```
docker exec -it spark_client pip uninstall elasticsearch
```

```
docker exec -it spark_client pip install elasticsearch==7.12.1
```

Check version

```
docker exec -it spark_client pip list | grep elasticsearch
```

Install other required packages

```
docker exec -it spark_client pip install findspark kafka-python pandas
```

Description: Installing required packages for visual output and progress monitoring for each Python script.

4. VERSION COMPATIBILITY OPERATIONS

Description: Downgrading Elasticsearch client to version 7.12.1 due to compatibility issues with Elasticsearch 8.x. This ensures communication between Spark and Elasticsearch.

5. INTER-CONTAINER CONNECTION TESTS

Go to main directory

```
cd /home/train/dataops11/spark_class
```

Create test topic

```
docker exec -it kafka /kafka/bin/kafka-topics.sh --create \  
--topic test-topic \  
--bootstrap-server kafka:9092 \  
--partitions 1 \  
--replication-factor 1
```

Check topic list

```
docker exec -it kafka /kafka/bin/kafka-topics.sh --list \
--bootstrap-server kafka:9092
```

Create Test script

```
touch test_connection.py
```

paste codes

Paste all codes in this script

Copy test script

```
docker cp test_connections.py spark_client:/opt/spark/
```

Run test script

```
docker exec -it spark_client python3 /opt/spark/test_connections.py
```

Description: Testing connections between Kafka, Spark, and Elasticsearch. This step ensures that the system is working properly before starting the data flow.

6. SPARK CONTAINER PREPARATION

Create required directories

```
cd /home/train/dataops11/spark_class
```

Create key directories in Spark client container

```
docker exec-it spark_client mkdir-p /opt/spark/python_files
```

```
docker exec-it spark_client mkdir-p /opt/final_project/KETI
```

```
docker exec-it spark_client mkdir-p /opt/data-generator/input
```

```
docker exec-it spark_client mkdir-p /opt/spark/ml_model
```

Copy Python processing scripts

```
docker cp python_files/preprocess_data.py spark_client:/opt/spark/python_files/  
docker cp python_files/dataframe_to_kafka_final.py spark_client:/opt/spark/python_files/  
docker cp python_files/spark_to_elasticsearch_wo_functions.py spark_client:/opt/spark/python_files/  
docker cp python_files/model_training.py spark_client:/opt/spark/python_files/  
docker cp python_files/spark_ml_stream.py spark_client:/opt/spark/python_files/
```

Copy KETI data (data was manually moved to data/KETI first)

```
docker cp data/KETI/. spark_client:/opt/final_project/KETI
```

Verify file transfers

```
docker exec-it spark_client ls-l /opt/spark/python_files/  
docker exec-it spark_client ls-l /opt/final_project/KETI/
```

Description: Creating the necessary directory structure in the Spark container and copying our Python codes and data files to the container.

7. DATA PREPROCESSING

Download KETI sensor data

```
wget https://github.com/erkansirin78/datasets/raw/master/sensors_instrumented_in_an_office_building_dataset.zip
```

Extract dataset

unzip sensors_instrumented_in_an_office_building_dataset.zip

Move to KETI data directory

mv KETI data/

Run preprocessing script

docker exec-it spark_client python3 /opt/spark/python_files/preprocess_data.py

Verify processed data

docker exec-it spark_client ls-l /opt/data-generator/input/sensors.csv

docker exec-it spark_client head-n 5 /opt/data-generator/input/sensors

Description: We download and convert sensor data into a processable format. This step makes the raw data ready for streaming.

8. KAFKA PIPELINE SETUP

Description: In this section, we are implementing our Kafka pipeline setup.

We will create three different Kafka topics:

"office-input" topic:

- Will receive CSV data created from data preprocessing
- Will be used as source data for both Elasticsearch stream and ML model

"office-activity" topic:

- Will be used for motion detection cases by ML model
- Data will be written to this topic when the model detects activity in a room

"office-no-activity" topic:

- Will be used for cases where ML model detects no motion
- Data will be written to this topic when the model detects no activity in a room

Go to main directory

```
cd /home/train/dataops11/spark_class
```

Create office-input topic

```
docker exec -it kafka /kafka/bin/kafka-topics.sh --create \  
--topic office-input \  
--bootstrap-server kafka:9092 \  
--partitions 1 \  
--replication-factor 1
```

Create office-activity topic

```
docker exec -it kafka /kafka/bin/kafka-topics.sh --create \  
--topic office-activity \  
--bootstrap-server kafka:9092 \  
--partitions 1 \  
--replication-factor 1
```

Create office-no-activity topic

```
docker exec -it kafka /kafka/bin/kafka-topics.sh --create \  
--topic office-no-activity \  
--bootstrap-server kafka:9092 \  
--partitions 1 \  
--replication-factor 1
```

Check Topic list

```
docker exec -it kafka /kafka/bin/kafka-topics.sh --list --bootstrap-server kafka:9092
```

Copy sensors.csv file to kafka (in two steps)

```
docker cp spark_client:/opt/data-generator/input/sensors.csv ./\  
docker cp sensors.csv kafka:/tmp/
```

Copy Producer code to Kafka container

```
docker cp python_files/dataframe_to_kafka_final.py kafka:/tmp/
```

Check files in Kafka container

```
docker exec -it kafka ls -l /tmp/
```


Run the Producer

```
docker exec-it kafka python3 /tmp/dataframe_to_kafka_final.py \  
-t office-input \  
-i /tmp/sensors.csv \  
-rst 0.5
```

Monitor data flow to Kafka

```
docker exec-it kafka /kafka/bin/kafka-console-consumer.sh \  
--bootstrap-server kafka:9092 \  
--topic office-input \  
--from-beginning
```

9. SPARK STREAMING AND ELASTICSEARCH

Start Streaming application

```
docker exec-it spark_client python3 /opt/spark/python_files/spark_to_elasticsearch_wo_functions.py
```

Check data written to Elasticsearch

```
curl-X GET "localhost:9200/office_input/_search?pretty&size=5&sort=event_ts_min:desc"
```

Check Elasticsearch status, list indices, get total record count

```
curl http://localhost:9200
```

```
curl http://localhost
```

```
curl -X GET "localhost:9200/office_input/_count"
```

Description: We are processing data from Kafka using Spark and writing it to Elasticsearch. We're checking if the data is being written successfully.

10. VISUALIZING GRAPHS IN KIBANA

1. Accessing Kibana:

- Go to <http://localhost:5601> in your web browser
- Select "Analytics" > "Dashboard" from left menu
- Click on "Sensors Real-time Dashboard"

2. Dashboard Settings:

- Set time range to "Last 15 minutes" from top right corner
- Check if auto-refresh is active (5 seconds)

3. Created Visualizations:

- Pie Chart: Distribution of motion/no-motion states
- Line Graph: CO2 levels in rooms
- Tag Cloud: Room motion intensity
- Arc Chart: Light levels based on motion status
- Bar Chart: Room-based PIR values
- Horizontal Bar: Temperature by motion status
- Area Graph: Hourly humidity trend

- Bar Chart: CO2 levels by motion status

4. Interactive Usage:

- Click on graphs to view details
- Filter data for specific rooms
- Modify time range
- Monitor real-time data flow with auto-refresh

Description: In this final step of our project, all sensor data is visualized and updated in real-time through a dashboard. This dashboard allows us to monitor motion, temperature, humidity, CO2, and light levels in the office building instantly.

11. ML MODEL TRAINING

Start model training

```
docker exec-it spark_client python3 /opt/spark/python_files/model_training.py
```

Description: We are training our machine learning model. This model will be trained to predict motion states in rooms using sensor data.

12. ML STREAM PROCESSING

Start ML stream processing

```
docker exec-it spark_client python3 /opt/spark/python_files/spark_ml_stream.py
```

Description: Using our trained model to analyze incoming sensor data in real-time and writing motion predictions to respective Kafka topics.

13. PREDICTION MONITORING

Monitor motion predictions

```
docker exec-it kafka /kafka/bin/kafka-console-consumer.sh \
--bootstrap-server kafka:9092 \
--topic office-activity \
```

--from-beginning

Monitor no-motion predictions

```
docker exec-it kafka /kafka/bin/kafka-console-consumer.sh \
```

```
--bootstrap-server kafka:9092 \
```

```
--topic office-no-activity \
```

```
--from-beginning
```

Description: We are monitoring our ML model's predictions. By listening to office-activity and office-no-activity topics, we track in real-time which rooms have detected motion or no motion.