# REAL-TIME SENSOR DATA ANALYSIS PROJECT PROCESS STEPS

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

## 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
```

**Description:** Creating the necessary directory structure for Docker containers.

## 2. STARTING CONTAINERS

```
# Go to main directory

cd /home/train/dataops11/spark_class

# Clean old services

docker-compose down--volumes

# Restart services

docker-compose up-d

# Check services status

docker ps
```

```
# Memory Map Limit Setting
password for train: Ankara06
sudo sysctl-w vm.max_map_count=262144
```

**Description:** Starting Docker containers cleanly and configuring system settings for Elasticsearch.

## 3. PACKAGE INSTALLATION

```
# Required package installations
docker exec-it spark_client pip3 install colorama tqdm
docker exec-it kafka pip3 install tqdm colorama
docker exec-it spark_client pip3 install colorama
```

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

## 4. VERSION COMPATIBILITY OPERATIONS

```
# Downgrade Elasticsearch client
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:** Downgrading to Elasticsearch 7.12.1 due to compatibility issues with 8.x. This ensures communication between Spark and Elasticsearch.

## 5. INTER-CONTAINER CONNECTION TESTING

**Explanation:** In the next step, we will prepare the test script file to test our system. By running the following code in the bash terminal in the directory we have navigated to, we will create our script named test_connection.py.

```
# 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 scprit

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. DATA DOWNLOAD

```
# Download KETI data

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

# Extract zip file

unzip sensors_instrumented_in_an_office_building_dataset.zip

# Move to data/KETI folder

mv KETI data/
```

**Description:** Downloading and converting sensor data into a processable format. This step makes the raw data ready for streaming.

## 7. KAFKA TOPIC CREATION

```
# 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
```

```
# Check topic

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

**Description:** Creating and verifying the office-input topic where produced messages will be stored in Kafka.

## 8. SPARK CONTAINER PREPARATION

```
# Create required directories

cd /home/train/dataops11/spark_class

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/spark/ml_model

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

# Copy Python files

docker cp python_files/spark_to_elasticsearch_wo_functions.py spark_client:/opt/spark/python_files/

docker cp python_files/stream_reader.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 (Before performing this step, I manually moved the data under data/KETI. In the data download section, this process is handled at the code level.)

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

# Check files exist

docker exec-it spark_client ls-l /opt/spark/python_files/

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

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

## 9. START DATA FLOW

```
# Run stream reader

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

# Open new terminal

# Run Elasticsearch writer

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

# Start the model training once the training data is ready

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

# Start the ML stream process

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

**Description:** Running stream_reader.py and spark_to_elasticsearch_wo_functions.py files sequentially to start the pipeline system.

## 10. DATA FLOW MONITORING

```
# Check data in Kafka

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

--bootstrap-server kafka:9092 \

--topic office-input \

--from-beginning


# Check data in Elasticsearch

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

```
curl http://localhost:9200/_cat/indices

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

```
# Check the ML predictions

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

--bootstrap-server kafka:9092 \

--topic office-activity \

--from-beginning
```

```
# Check the ML predictions

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

--bootstrap-server kafka:9092 \

--topic office-no-activity \

--from-beginning
```

**Description:** Monitoring data flow in both Kafka and Elasticsearch systems.

## 11. KIBANA VISUALIZATION

### 1. Accessing Kibana:

- Go to http://localhost:5601 in web browser
- Select "Analytics" > "Dashboard" from left menu

### 2. Dashboard Settings:

- Set time range to "Last 15 minutes"
- Enable auto-refresh (5 seconds)

**3. Visualization Types:**

- Real-time sensor metrics
- Activity tracking
- Environmental monitoring
- Room-specific analytics
- Trend analysis
- Motion detection status

**Description:** Setting up real-time visualization dashboard in Kibana for monitoring sensor data, activity detection, and environmental metrics.