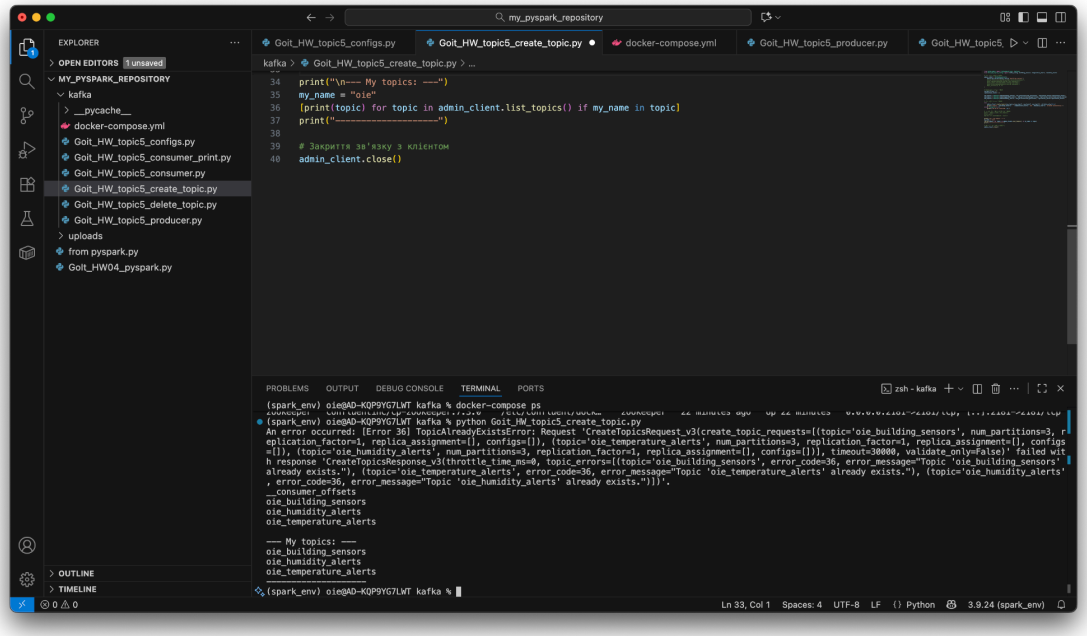
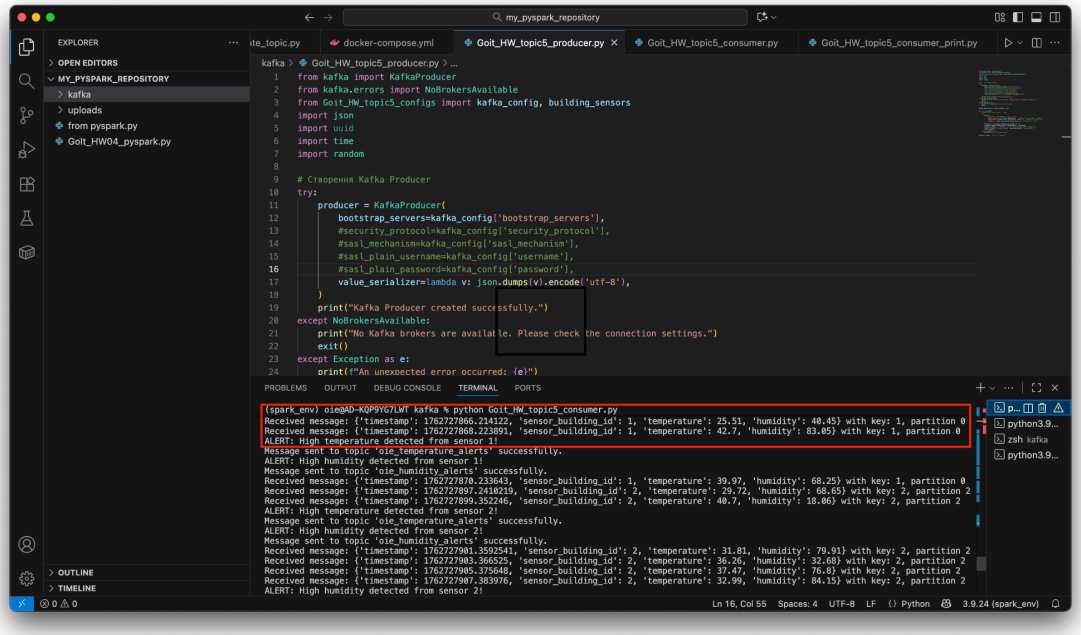
Кроки виконання завдання:

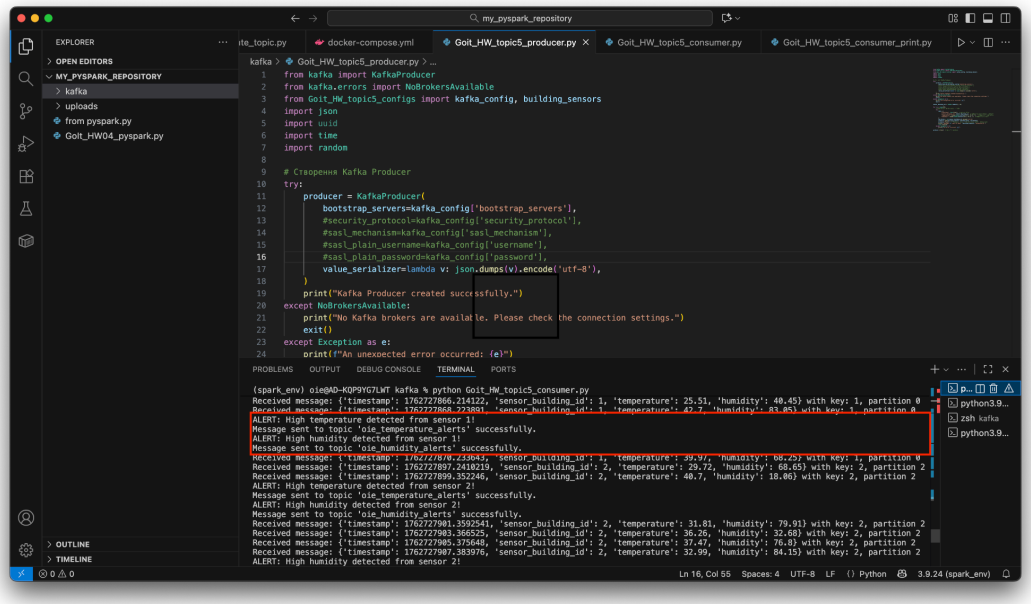
1. Створення topics. Було створено три топіка: oie_building_sensors, oie_humidity_alerts, oie_temperature_alerts на локальному сервері. Для цього створено контейнер kafka через файл docker-compose.yml



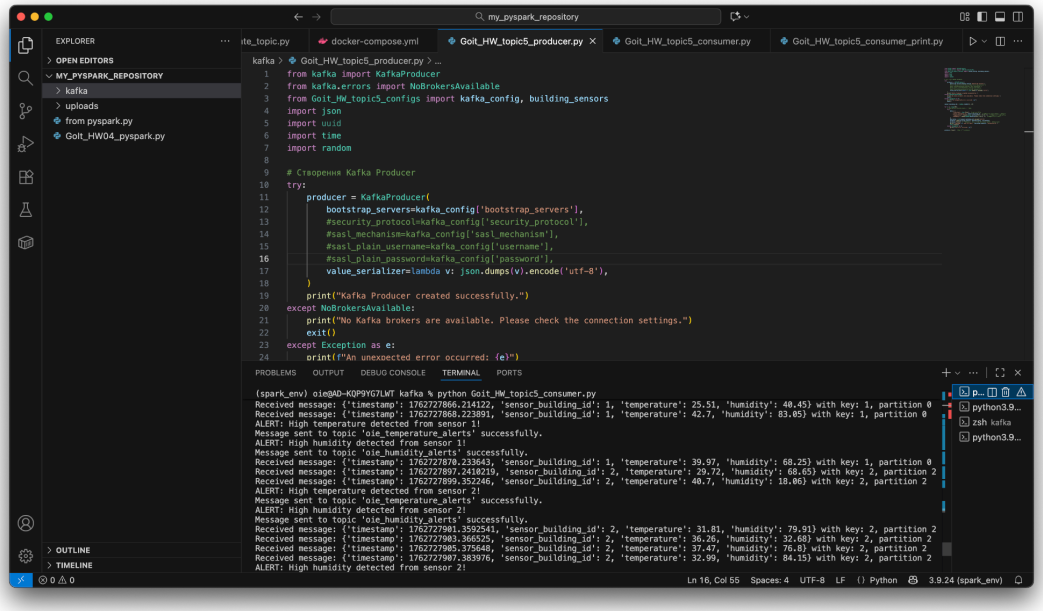
2. Створено та запущено в роботу consumer (підписаний на oie_building_sensors). Що приймає сповіщення від provider з даними, отриманими від датчиків (у даному випадку два датчика sensor_building_1, sensor_building_2) та обробляє отримані дані.



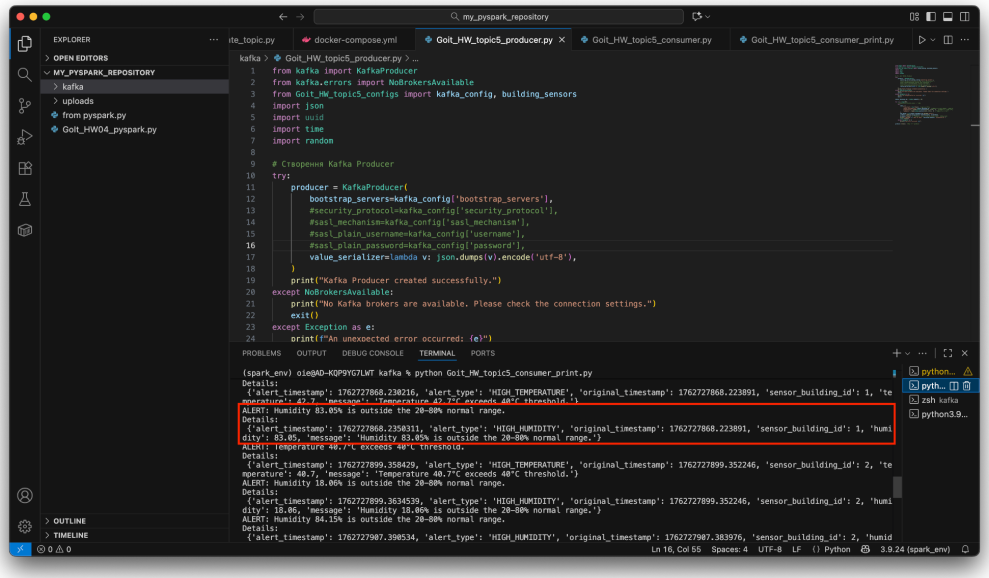
У разі відповідності параметрів заданим умовам (перевищення певного порогу значень) вбудований provider формує alert сповіщення та відправляє його в інші топіки (oie_humidity_alerts, oie_temperature_alerts) в залежності від того, який з параметрів відповідає заданим умовам.



3. Створено та запущено в роботу provider. Що імітує роботу двох датчиків (у даному випадку два датчика sensor_building_1, sensor_building_2), генерує сповіщення з даними, отриманими від датчиків , та відправляє сповіщення в топік oie_building_sensors, який слухає consumer (підписаний на oie_building_sensors).



4. Створено та запущено в роботу ще один consumer (підписаний на oie_humidity_alerts, oie_temperature_alerts), що приймає сповіщення від provider з alert сповіщеннями (містять оригінальні показники датчика), та виводять попередження у консоль, що містить показники датчика та причину сповіщення.


my_pyspark_repository
EXPLORER
OPEN EDITORS
MY_PYSPARK_REPOSITORY
kafka
uploads
from pyspark.py
Goit_HW04_pyspark.py
ite_topic.py
docker-compose.yml
Goit_HW_topic5_producer.py
Goit_HW_topic5_consumer.py
Goit_HW_topic5_consumer_print.py
kafka > Goit_HW_topic5_producer.py > ...
from kafka import KafkaProducer
from kafka.errors import NoBrokersAvailable
from Goit_HW_topic5_configs import kafka_config, building_sensors
import json
import uuid
import time
import random
# Створення Kafka Producer
try:
producer = KafkaProducer(
bootstrap_servers=kafka_config['bootstrap_servers'],
#security_protocol=kafka_config['security_protocol'],
#sasl_mechanism=kafka_config['sasl_mechanism'],
#sasl_plain_username=kafka_config['username'],
#sasl_plain_password=kafka_config['password'],
value_serializer=lambda v: json.dumps(v).encode('utf-8'),
)
print("Kafka Producer created successfully.")
except NoBrokersAvailable:
print("No Kafka brokers are available. Please check the connection settings.")
exit()
except Exception as e:
print(f"An unexpected error occurred: {e}")
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
(spark_env) oie@AD-KQP9YG7LWT kafka % python Goit_HW_topic5_consumer_print.py
Details:
{'alert_timestamp': 1762727868.230216, 'alert_type': 'HIGH_TEMPERATURE', 'original_timestamp': 1762727868.223891, 'sensor_building_id': 1, 'temperature': 42.7, 'message': 'Temperature 42.7°C exceeds 40°C threshold.'}
ALERT: Humidity 83.05% is outside the 20–80% normal range.
Details:
{'alert_timestamp': 1762727868.2350311, 'alert_type': 'HIGH_HUMIDITY', 'original_timestamp': 1762727868.223891, 'sensor_building_id': 1, 'humidity': 83.05, 'message': 'Humidity 83.05% is outside the 20–80% normal range.'}
ALERT: Temperature 40.7°C exceeds 40°C threshold.
Details:
{'alert_timestamp': 1762727899.358429, 'alert_type': 'HIGH_TEMPERATURE', 'original_timestamp': 1762727899.352246, 'sensor_building_id': 2, 'temperature': 40.7, 'message': 'Temperature 40.7°C exceeds 40°C threshold.'}
ALERT: Humidity 18.06% is outside the 20–80% normal range.
Details:
{'alert_timestamp': 1762727899.3634539, 'alert_type': 'HIGH_HUMIDITY', 'original_timestamp': 1762727899.352246, 'sensor_building_id': 2, 'humidity': 18.06, 'message': 'Humidity 18.06% is outside the 20–80% normal range.'}
ALERT: Humidity 84.15% is outside the 20–80% normal range.
Details:
{'alert_timestamp': 1762727907.390534, 'alert_type': 'HIGH_HUMIDITY', 'original_timestamp': 1762727907.383976, 'sensor_building_id': 2, 'humid
python...
pyth...
zsh kafka
python3.9...
OUTLINE
TIMELINE
Ln 16, Col 55
Spaces: 4
UTF-8
LF
Python
3.9.24 (spark_env)