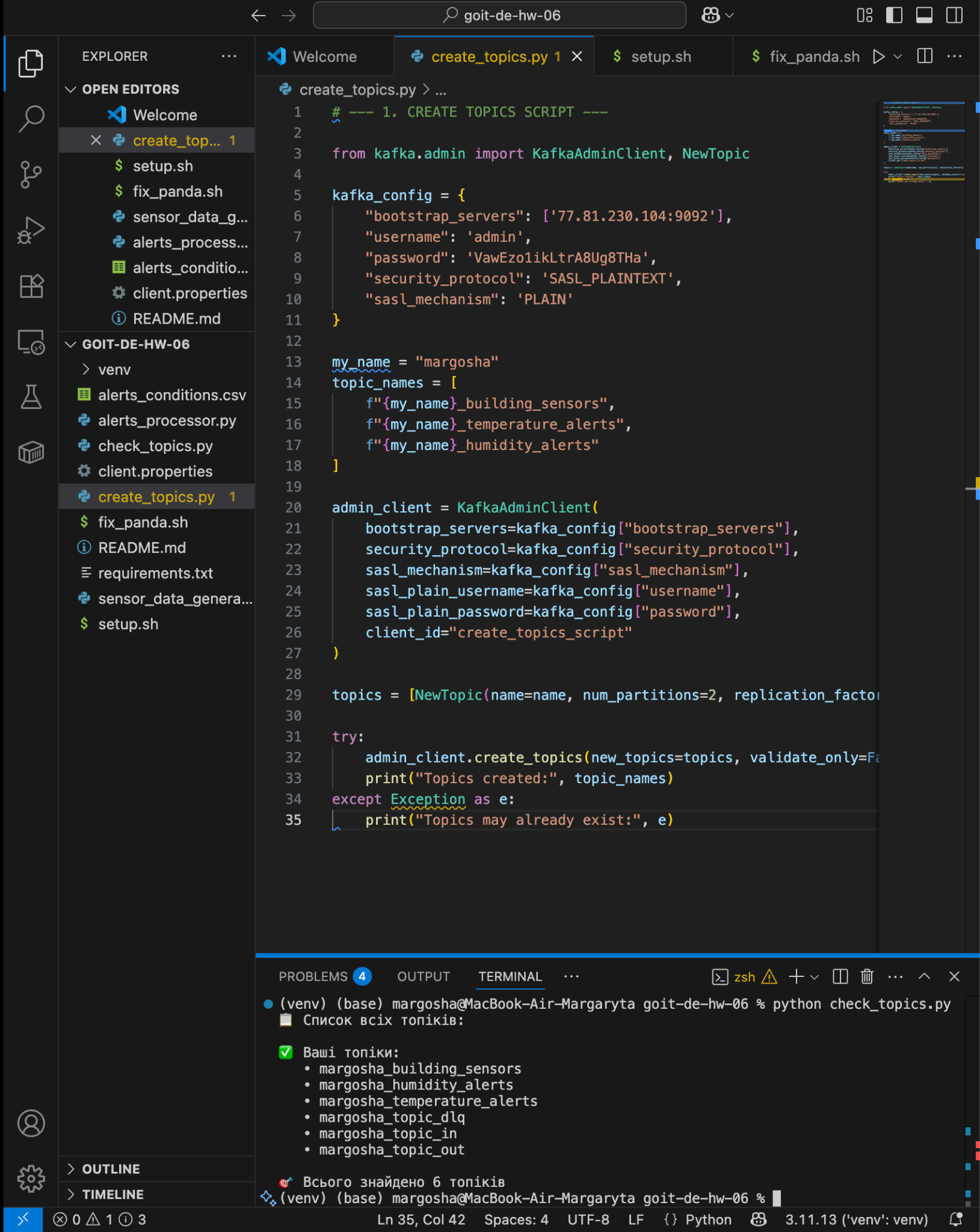
## **Скриншоти виконання**

### **Скриншот 1: Створення Kafka топіків**

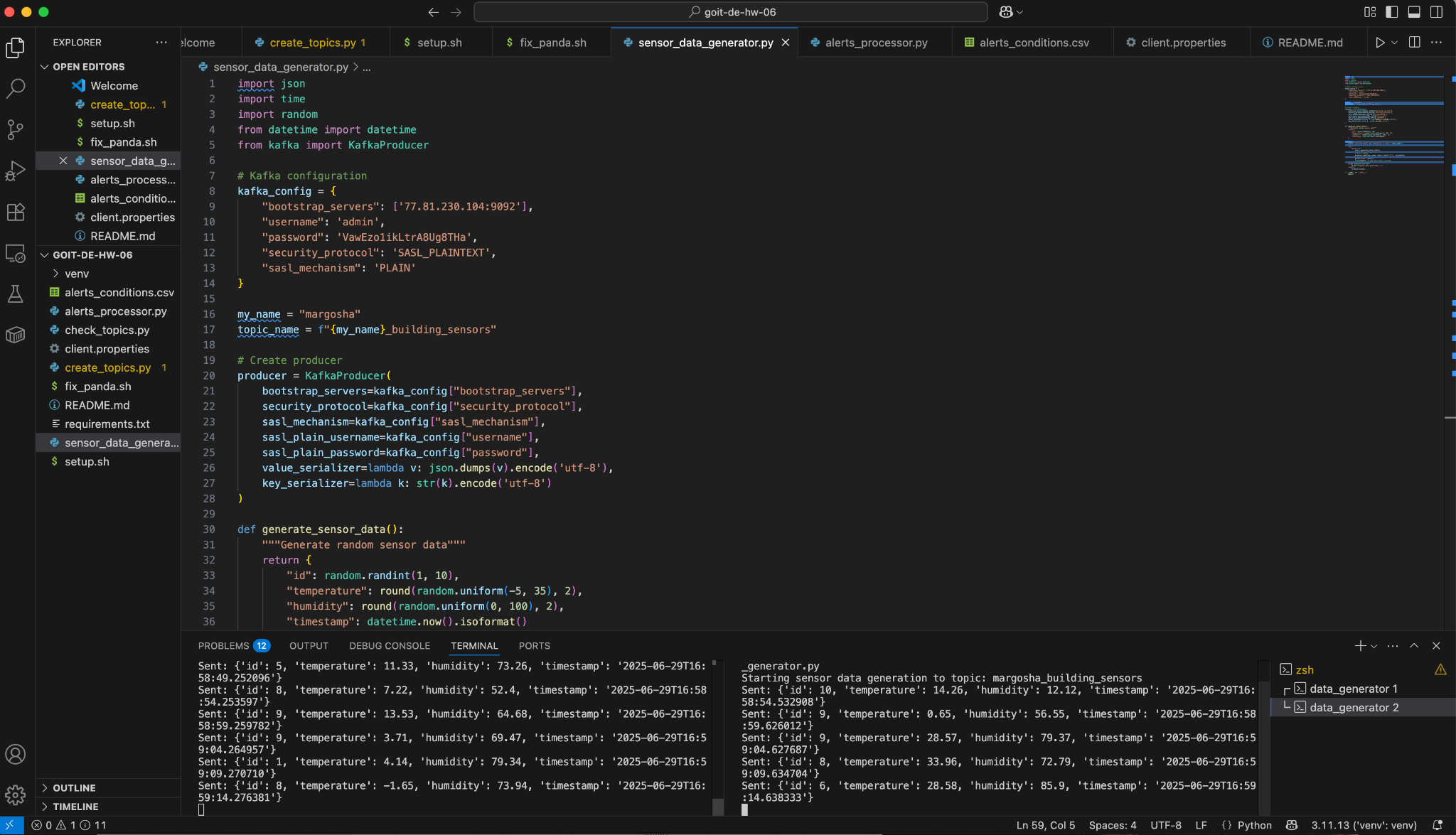


**Опис:** Успішне створення трьох Kafka топіків за допомогою скрипта create\_topics.py:

* margosha\_building\_sensors - для вхідних даних сенсорів
* margosha\_temperature\_alerts - для алертів температури
* margosha\_humidity\_alerts - для алертів вологості

Виведено список існуючих топіків, що підтверджує успішне створення.

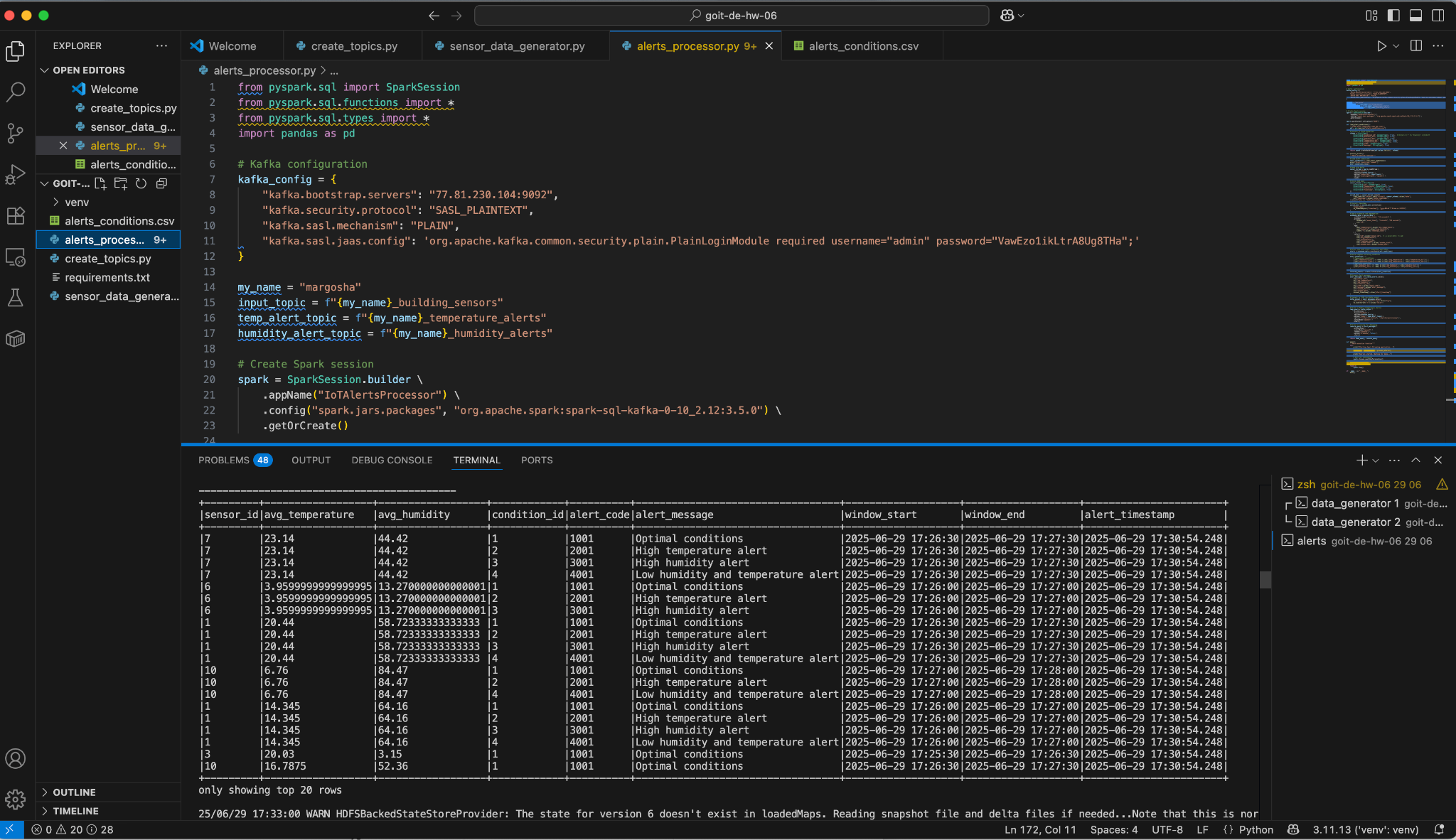
### **Скриншот 2: Генерація даних сенсорів**



**Опис:** Демонстрація роботи двох одночасних генераторів даних (data\_generator 1 та data\_generator 2). Кожен генератор відправляє дані сенсорів кожні 5 секунд у топік margosha\_building\_sensors. Показано успішну відправку даних з різними значеннями температури та вологості.

**Підтвердження критерію:** ✅ Два або більше одночасних запусків програми генерації даних.

### **Скриншот 3: Обробка алертів Spark Streaming**



**Опис:** Spark Streaming додаток IoTAlertsProcessor успішно:

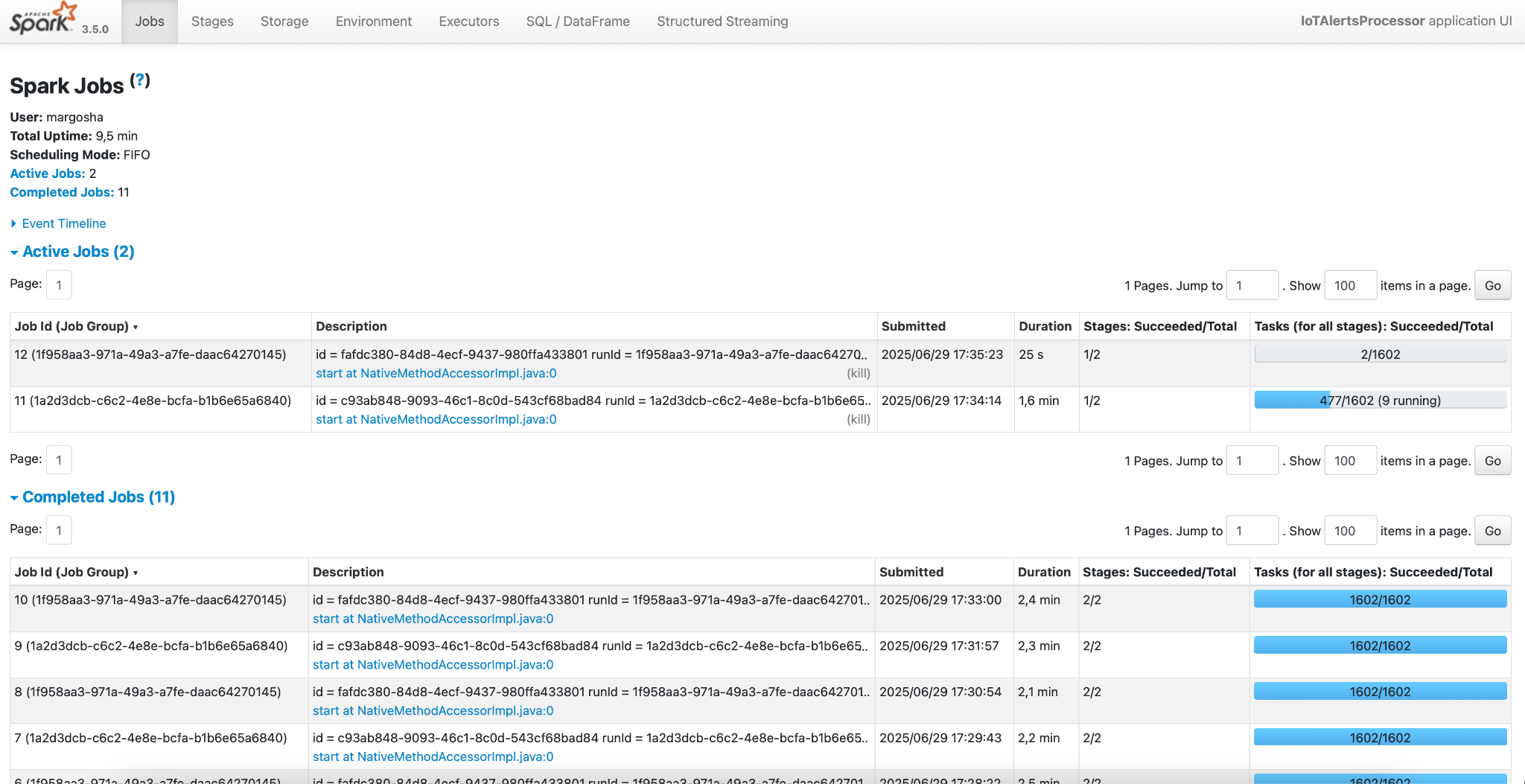
* Завантажив умови алертів з CSV файлу
* Обробляє потік даних з Kafka
* Генерує алерти на основі sliding window агрегації
* Виводить результати в консоль

Показано успішну генерацію різних типів алертів:

* "Optimal conditions" (код 1001)
* "High temperature alert" (код 2001)
* "High humidity alert" (код 3001)
* "Low humidity and temperature alert" (код 4001)

**Підтвердження критерію:** Відфільтровані дані успішно обробляються та відправляються у відповідні топіки.

### **Скриншот 4: Spark UI Dashboard**



**Опис:** Веб-інтерфейс Spark ([http://localhost:4040](http://localhost:4040/)) показує:

* **Active Jobs:** 2 (потоки для консольного виводу та запису в Kafka)
* **Completed Jobs:** 11 (попередні обробки батчів)
* **User:** margosha
* **Total Uptime:** 9.5 хвилин

Демонструє стабільну роботу Spark Streaming додатку з обробкою потоків даних.