### **Домашнє завдання: Реалізація DAG у Apache Airflow**

#### **Результати виконання DAG**

* Назва DAG: viktor\_svertoka\_dag2
* Кількість запусків: 3
  + Успішні запуски: 2 (позначено зеленим кольором)
  + Неуспішні запуски: 1 (позначено червоним кольором)
* Дати та час запусків:
  + Перший запуск: 2024-12-03, 12:20:37 UTC
  + Останній запуск: 2024-12-03, 12:25:51 UTC
* Тривалість виконання DAG:
  + Максимальна: 00:01:36
  + Середня: 00:01:07
  + Мінімальна: 00:00:50

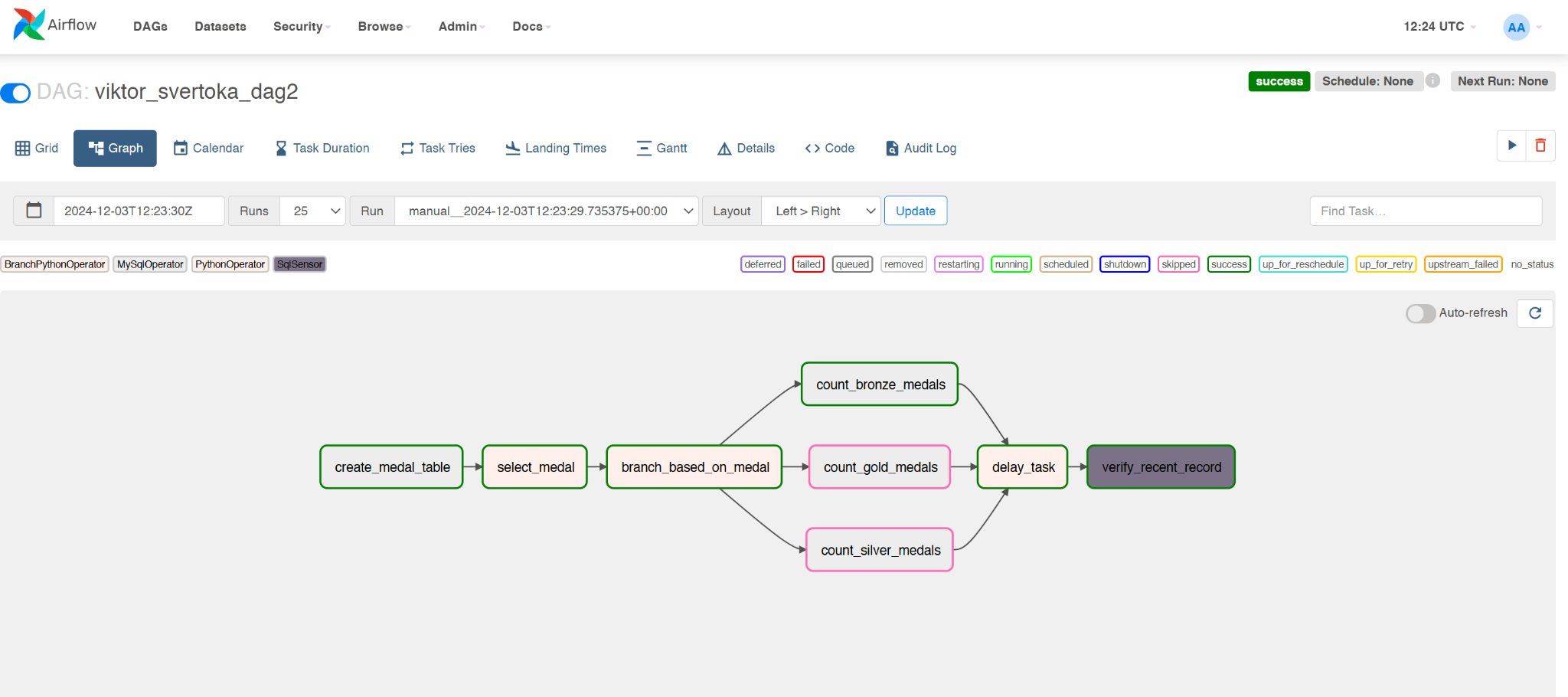
#### **Огляд компонентів DAG**

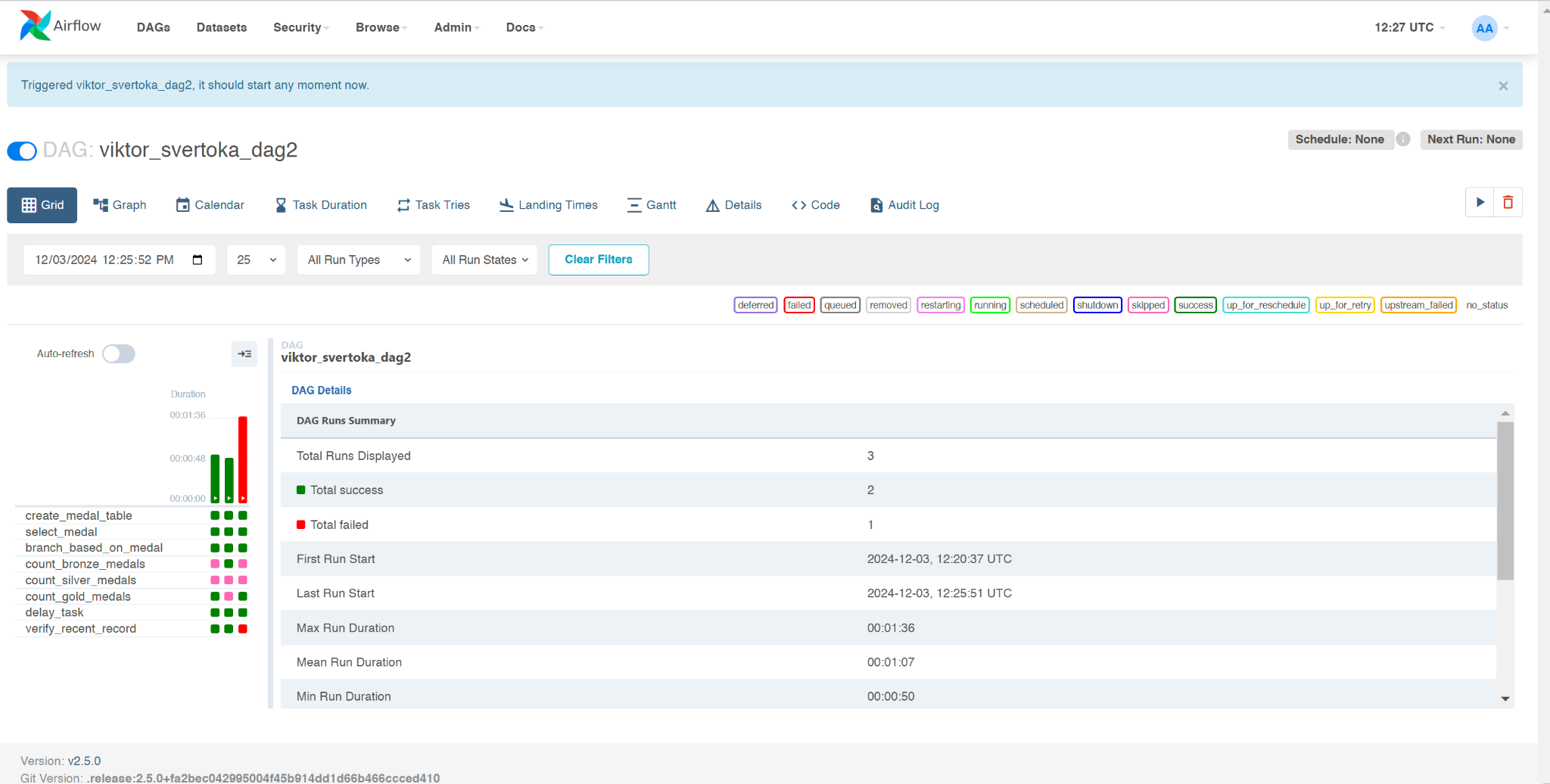
1. Кількість задач: 8
   * Оператори:
     + MySqlOperator: 4
     + PythonOperator: 2
     + BranchPythonOperator: 1
     + SqlSensor: 1
2. Зв’язок задач:
   * Логіка DAG починається зі створення таблиці, після чого відбувається випадковий вибір типу медалі.
   * Вибір типу медалі (Gold, Silver або Bronze) визначає наступну гілку виконання.
   * Після виконання підрахунку медалей виконується затримка в 35 секунд, а далі сенсор перевіряє актуальність запису в таблиці.

#### **Реалізовані кроки**

1. Створення таблиці
   * Використовується MySqlOperator для створення таблиці neo\_data.viktor\_svertoka\_medal\_counts із полями:
     + id (автоінкремент, PRIMARY KEY)
     + medal\_type (тип медалі)
     + medal\_count (кількість медалей)
     + created\_at (дата створення, за замовчуванням поточний час)
2. Генерація випадкового значення
   * За допомогою PythonOperator обирається один із трьох типів медалей: Gold, Silver, Bronze.
3. Розгалуження DAG
   * Залежно від типу медалі використовується BranchPythonOperator для вибору відповідної гілки виконання:
     + Підрахунок кількості золотих медалей
     + Підрахунок кількості срібних медалей
     + Підрахунок кількості бронзових медалей
4. Підрахунок медалей
   * Кожне завдання підрахунку реалізоване через MySqlOperator, який виконує SQL-запит для підрахунку записів у таблиці olympic\_dataset.athlete\_event\_results та записує результат у таблицю viktor\_svertoka\_medal\_counts.
5. Затримка виконання
   * Завдання із затримкою реалізовано через PythonOperator. Виконується затримка тривалістю 35 секунд, що моделює процес складної обробки.
6. Перевірка актуальності запису
   * Використовується SqlSensor для перевірки, чи є записи у таблиці, створені не пізніше ніж 30 секунд тому. У разі затримки понад 30 секунд завдання позначається як невдале.

**Скриншоти:**

**Структура DAG**: Граф виконання завдань.  
  


**Результати запусків**: Лог та статуси виконання DAG.  
  


**Скриншоти з таблиці MySQL**: Із записаними результатами.  
  
