Sistem Monitoring Kualitas Udara Real-Time Berbasis Arsitektur Terdistribusi – Proyek Data Warehouse Kelompok 11

Deskripsi Singkat

Proyek ini membangun sistem ETL *(Extract, Transform, Load)* untuk memproses data kualitas udara secara terstruktur dan otomatis. Sistem dikembangkan menggunakan Python dan *Celery* untuk menjalankan proses secara terdistribusi, dengan RabbitMQ sebagai *message broker*. Data mentah yang berisi informasi dibaca dan diproses untuk menjalankan *tasks* yang diminta. Hasil transformasi kemudian disimpan ke dalam file CSV serta disimpan ke dalam database PostgreSQL agar dapat dianalisis lebih lanjut. Proses *tasks* dipantau menggunakan *Flower* sebagai dashboard monitoring yang memberikan visibilitas real-time terhadap status dan hasil tugas Celery. Proyek ini dirancang agar fleksibel, modular, dan siap dikembangkan ke tahap integrasi sistem atau visualisasi data.

Tujuan

Proyek ini bertujuan untuk membangun sistem monitoring kualitas udara secara real-time dengan memanfaatkan pendekatan ETL (Extract, Transform, Load) dan arsitektur komputasi terdistribusi. Sistem ini dirancang untuk mendukung pengambilan keputusan berbasis data dalam bidang kesehatan lingkungan, riset ilmiah, dan peringatan dini terhadap polusi udara.

Struktur Proyek

Proyek ini terdiri dari beberapa file sebagai berikut:

1. celery_app.py untuk Konfigurasi Celery.
2. Task Celery (ETL) yang terdapat pada beberapa file berikut:
    - tasks1.py yang berisi proses ETL untuk menghitung jumlah hari yang kualitas udara buruk.
    - tasks2.py yang berisi proses ETL untuk menghitung nilai rata-rata dan maksimum tiap polutan per stasiun.
    - tasks3.py yang berisi proses ETL untuk mencari hari dengan polutan tertinggi per stasiun.
    - tasks4.py berisi proses ETL untuk menghitunug rata-rata polutan bulanan per stasiun.
    - tasks5.py berisi proses ETL untuk melihat kategori kualitas udara terbanyak per stasiun.
    - tasks6.py berisi proses ETL untuk menghitung frekuensi polutan kritikal per stasiun.
3. Producer untuk mengirim tasks ke Celery. Setiap *tasks* memiliki 1 producer.
    - producer.py untuk mengirim tasks.py khususnya untuk mengambil beberapa data ke Celery.
4. load_rdbms.py" untuk koneksi ke PostgreSQL
5. Folder Output untuk menyimpan file csv hasil running semua tasks.

6. File requirements.txt berisi library apa yang perlu diunduh dalam proyek ini.

Persiapan

1. Python
2. PostgreSQL terinstal dan aktif
3. RabbitMQ terinstal dan aktif

Install Library

Karena library yang diperlukan sudah ada pada file requirements.txt, maka kita bisa langsung unduh dengan pip, dengan contoh sebagai berikut:

pip install -r requirements.txt

Konfigurasi PostgreSQL

Pastikan Anda memiliki database bernama cuaca_db. Jika belum, silakan membuatnya dengan:

CREATE DATABASE cuaca_db;

Kemudian ubah konfigurasi di file **load_rdbms.py** sesuai setting yang Anda miliki, sebagai contoh:

```
DB_USER = "postgres"
DB_PASSWORD = "postgres"
DB_HOST = "localhost"
DB_PORT = "5432"
DB_NAME = "cuaca_db"
```

Menjalankan Sistem

1. Jalankan RabbitMQ
   Disarankan untuk menggunakan Docker dalam menjalankan RabbitMQ. Buka docker dan jalankan RabbitMQ atau bisa dengan:

   docker run -d --hostname rabbit --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3-management

2. Akses dashboard di http://localhost:15672 (user: guest, password: guest)
3. Jalankan worker Celery

   celery -A tasks worker --loglevel=info –pool=solo

Pastikan untuk mengganti **tasks** menjadi nama file task yang ingin dijalankan. Misal ingin menjalankan tasks2.py, maka dapat diubah menjadi:

celery -A tasks2 worker --loglevel=info

4. Jalankan Flower untuk monitoring task

   celery -A tasks flower

   Setelah berhasil dijalankan, silakan akses di http://localhost:5555
5. Jalankan producer untuk memicu *task*

   python producer.py

   Setelah itu, hasil setiap task akan dibuat.
6. Untuk setiap file csv yang terbuat pindahkan ke database. Jalankan file **load_rdbms.py** untuk mengubah file csv menjadi sebuah database.

   python load_rdbms.py