

**MODEL PEMBELAJARAN DAN LAPORAN AKHIR**  
**PROJECT-BASED LEARNING**  
**MATA KULIAH BIG DATA**  
**KELAS B**



**“Optimalisasi Pemantauan Data Cuaca *Real-Time* dari OpenWeatherMap  
Menggunakan Apache Kafka dan Kibana”**

**DISUSUN OLEH KELOMPOK I :**

- |                             |                           |
|-----------------------------|---------------------------|
| 1. NABILAH SELAYANTI        | ( 22083010013 ) - KETUA   |
| 2. DIAJENG SEKAR PRAMESWARI | ( 22083010003 ) - ANGGOTA |
| 3. INTAN ADIBA              | ( 22083010007 ) - ANGGOTA |
| 4. SHAFIRA AMANDA PUTRI     | ( 22083010008 ) - ANGGOTA |
| 5. LARASATI                 | ( 22083010018 ) - ANGGOTA |
| 6. DESI TRISTIANI           | ( 22083010037 ) - ANGGOTA |

**DOSEN PENGAMPU:**

TRESNA MAULANA FAHRUDIN, S.ST., MT ( 20219930501200 )

**PROGRAM STUDI SAINS DATA**  
**FAKULTAS ILMU KOMPUTER**  
**UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”**  
**JAWA TIMUR**  
**2024**

## KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Tuhan Yang Maha Esa, atas segala rahmat dan petunjuk-Nya yang senantiasa melimpahkan berkah dalam perjalanan penulisan ini. Dengan penuh keikhlasan dan semangat yang membara, penulis menghadirkan kata pengantar ini sebagai bagian dari Laporan Akhir yang berjudul “Optimalisasi Pemantauan Data Cuaca *Real-Time* dari OpenWeatherMap Menggunakan Apache Kafka dan Kibana”. Laporan ini merupakan hasil dari Project Based Learning yang dikerjakan dalam rangka memenuhi persyaratan mata kuliah Big Data.

Penulisan laporan ini tidak terlepas dari bimbingan serta arahan yang berharga dari Bapak Tresna Maulana Fahrudin S.S.T., M.T., yang dengan sabar memberikan dukungan dalam setiap tahapan penelitian dan penulisan. Beliau tidak hanya menjadi pembimbing akademis tetapi juga memberikan inspirasi dalam mengembangkan pemahaman terhadap teknologi Big Data. Keberhasilan penyelesaian laporan ini juga tidak terlepas dari dukungan moral dan teknis dari teman-teman seperjuangan, yang senantiasa memberikan dorongan dan masukan yang membangun.

Semoga laporan akhir ini tidak hanya memberikan manfaat bagi penulis pribadi dalam menambah pengetahuan dan keterampilan, tetapi juga dapat memberikan kontribusi yang berarti bagi pembaca yang mengikuti jejak penelitian ini. Dengan penuh harapan, penulis berharap bahwa hasil penelitian ini dapat memberikan inspirasi serta solusi yang dapat diimplementasikan dalam pengembangan teknologi informasi, khususnya dalam pengelolaan data cuaca secara *real-time*.

Surabaya. 13 Juni 2024

Tim Penulis Kelompok 1

## DAFTAR ISI

<b>KATA PENGANTAR.....</b>	<b>1</b>
<b>DAFTAR GAMBAR.....</b>	<b>4</b>
<b>DAFTAR TABEL.....</b>	<b>6</b>
<b>BAB I</b>	
<b>PENDAHULUAN.....</b>	<b>7</b>
1.1. Latar Belakang.....	7
1.2. Permasalahan.....	8
1.3. Tujuan.....	8
1.4. Manfaat.....	9
<b>BAB II</b>	
<b>TINJAUAN PUSTAKA.....</b>	<b>10</b>
2.1. Teori Penunjang.....	10
2.1.1. Apache Kafka.....	10
2.1.2. Kibana.....	10
2.1.3. Amazon Web Service (AWS).....	11
2.1.4. OpenWeatherMap.....	11
2.1.5. Application Programming Interface (API).....	12
2.1.6. Pemrograman Python.....	13
2.2. Penelitian Terkait.....	13
2.2.1. Implementasi Sistem Distribusi Pesan dan Proses Data Secara Real Time dengan Apache Kafka oleh Fezan Nabawi (2022).....	13
2.2.2. Penerapan Log Analyzer Log untuk Mengetahui Lalu Lintas Jaringan Berbasis Elasticsearch, Logstash, dan Kibana oleh Muhammad Jafier Rama Putra dan Henry Saptono (2022).....	14
2.2.3. Implementasi Sistem Real Time untuk Pendekripsi Dini Banjir berbasis ESP8266 dan Weather API oleh Shandi Sonna Mahardika, Wijaya Kurniawan, dan Fariz Andri Bakhtiar (2019).....	14
<b>BAB III</b>	
<b>METODOLOGI PENELITIAN.....</b>	<b>15</b>
3.1. Mendapatkan API Key dari OpenWeatherMap.....	16
3.2. Crawling Data Cuaca Real-time.....	16
3.3. Membuat Producer yang Mengirimkan Data Cuaca ke Topik Kafka.....	16
3.4. Mengintegrasikan Data Cuaca dengan Apache Kafka dan AWS.....	17
3.5. Membuat Consumer untuk Menerima Data Cuaca dari Topik Kafka.....	17

3.6. Cloud Storage Data pada Amazon S3.....	18
3.8. Visualisasi Data Cuaca dengan Apache Kibana.....	18
<b>BAB IV</b>	
<b>HASIL DAN PEMBAHASAN.....</b>	<b>20</b>
4.1. Implementasi Apache Kafka.....	20
4.1.1. Install dan Mengaktifkan Apache Kafka.....	20
4.1.2. Proses Crawling Data Cuaca dari OpenWeathermap.....	25
4.1.3. Menghubungkan Kafka dengan AWS.....	31
4.2. Implementasi Apache Kibana.....	44
4.2.1. Proses Instalasi.....	44
4.2.2 Menjalankan dan Memeriksa Aplikasi.....	46
4.2.3 Input Data ke dalam Kibana.....	49
4.2.4 Statistika Deskriptif pada Data.....	50
4.2.4 Visualisasi Data menggunakan Kibana.....	53
<b>BAB V: KESIMPULAN.....</b>	<b>63</b>
<b>DAFTAR PUSTAKA.....</b>	<b>65</b>
<b>LAMPIRAN.....</b>	<b>68</b>

## DAFTAR GAMBAR

<b>Gambar 1.</b> Logo Apache Kafka.....	10
<b>Gambar 2.</b> Logo Kibana.....	10
<b>Gambar 3.</b> Logo AWS.....	11
<b>Gambar 4.</b> Logo OpenWeatherMap.....	11
<b>Gambar 5.</b> Logo API.....	12
<b>Gambar 6.</b> Logo Python.....	13
<b>Gambar 8.</b> Install Kafka.....	20
<b>Gambar 9.</b> Memindahkan ke drive C.....	21
<b>Gambar 10.</b> Konfigurasi untuk install.....	21
<b>Gambar 11.</b> Membuka file server.....	22
<b>Gambar 12.</b> Ubah direktori kafka.....	22
<b>Gambar 13.</b> Ubah direktori zookeeper.....	23
<b>Gambar 14.</b> Mulai server zookeeper.....	24
<b>Gambar 16.</b> Pooling data real-time.....	30
<b>Gambar 18.</b> Hubungkan kafka ke AWS.....	32
<b>Gambar 19.</b> Memberi nama instance.....	33
<b>Gambar 20.</b> Pemberian nama telah berhasil.....	33
<b>Gambar 21.</b> Menyambungkan AWS dengan cmd.....	34
<b>Gambar 22.</b> Navigasi ke direktori penyimpanan.....	34
<b>Gambar 23.</b> Hubungkan ke instance EC2 AWS.....	34
<b>Gambar 25.</b> Data cuaca JSON.....	39
<b>Gambar 26.</b> Hasil data yang terkirim di bucket S3.....	40
<b>Gambar 28.</b> Hasil rata - rata suhu tiap kota.....	41
<b>Gambar 29.</b> Code hitung rata-rata kelembaban tiap kota.....	42
<b>Gambar 30.</b> Hasil rata-rata kelembaban tiap kota.....	42
<b>Gambar 31.</b> Code hitung suhu tertinggi dan terendah tiap kota.....	43
<b>Gambar 32.</b> Hasil suhu terendah dan tertinggi tiap kota.....	43
<b>Gambar 33.</b> Code hitung rata-rata tekanan udara tiap kota.....	44
<b>Gambar 34.</b> Hasil rata-rata tekanan udara tiap kota.....	44
<b>Gambar 35.</b> Code hitung rata-rata UV tiap kota.....	45
<b>Gambar 36.</b> Hasil rata-rata UV tiap kota.....	45
<b>Gambar 37.</b> Menginstal elasticsearch.....	46
<b>Gambar 38.</b> Menginstal kibana.....	46
<b>Gambar 39.</b> Mengekstrak file elasticsearch dan kibana.....	47
<b>Gambar 41.</b> Menjalankan cmd kibana.....	48
<b>Gambar 42.</b> Mengecek localhost elasticsearch.....	48
<b>Gambar 43.</b> Mengecek localhost kibana.....	49
<b>Gambar 44.</b> Import data.....	49

<b>Gambar 45.</b> Input file.....	50
<b>Gambar 47.</b> Statistik variabel V1.....	51
<b>Gambar 48.</b> Statistik variabel V2 - V5.....	52
<b>Gambar 50.</b> Statistik variabel V10 - V14.....	53
<b>Gambar 53.</b> Outputan data yang telah di import dan siap di visualisasikan.....	54
<b>Gambar 55.</b> Memasukkan variabel yang akan dilakukan visualisasi.....	56
<b>Gambar 56.</b> Memilih jenis visualisasi.....	56
<b>Gambar 57.</b> Memilih filter dan function yang akan digunakan.....	57
<b>Gambar 58.</b> Memilih beberapa variabel pada sumbu y.....	57
<b>Gambar 59.</b> Memilih function average untuk eksekusi visualisasi.....	58
<b>Gambar 61.</b> Visualisasi max/min suhu tiap kota.....	59
<b>Gambar 62.</b> Visualisasi max/min paparan UV tiap kota.....	60
<b>Gambar 63.</b> Menampilkan peta.....	61
<b>Gambar 64.</b> Visualisasi perbandingan cuaca tiap kota.....	61
<b>Gambar 65.</b> Visualisasi tren harian.....	62
<b>Gambar 66.</b> Hasil Akhir Visualisasi.....	63

## DAFTAR TABEL

<b>Tabel 1.</b> Buka command.....	23
<b>Tabel 2.</b> Aktifkan kafka.....	24
<b>Tabel 3.</b> Pooling data real-time.....	29
<b>Tabel 4.</b> Instalasi Apache kafka.....	35
<b>Tabel 5.</b> Menjalankan zookeeper.....	35
<b>Tabel 6.</b> Menjalankan server kafka.....	36
<b>Tabel 7.</b> Buat topik.....	37
<b>Tabel 8.</b> Start producer Kafka di cmd.....	37
<b>Tabel 9.</b> Start konsumen Kafka di cmd.....	37
<b>Tabel 10.</b> Membuat producer dengan python.....	39
<b>Tabel 11.</b> Membuat consumer dengan python.....	40

## **BAB I**

### **PENDAHULUAN**

#### **1.1. Latar Belakang**

Cuaca memiliki pengaruh besar terhadap aktivitas manusia sehari-hari karena sifatnya yang dinamis dan berubah-ubah. Menurut Kartasapoetra (2004), cuaca adalah kondisi atmosfer pada waktu tertentu yang berubah setiap saat, meliputi suhu, kelembaban, intensitas cahaya, serta kecepatan dan arah angin. Perubahan ini dapat dipantau dengan perangkat Stasiun Cuaca Otomatis (*Automatic Weather Station*), yang menekankan pentingnya data cuaca real-time untuk sektor-sektor seperti pertanian, transportasi, perencanaan kota, dan manajemen bencana. Data yang cepat dan akurat sangat diperlukan untuk mendukung pengambilan keputusan yang lebih baik dan tepat waktu dalam berbagai sektor tersebut.

Indonesia, dengan karakteristiknya sebagai negara kepulauan yang terletak di garis khatulistiwa, memiliki kondisi cuaca yang unik dan beragam. Dipengaruhi oleh perubahan iklim global, pola musiman, serta fenomena alam seperti El Niño dan La Niña. Indonesia mengalami dua musim utama, yaitu musim hujan dari Oktober hingga Maret dan musim kemarau dari April hingga September, dengan variasi curah hujan yang signifikan antar wilayah (Suhery et al., 2023). Perubahan iklim juga meningkatkan frekuensi dan intensitas cuaca ekstrem seperti banjir dan longsor. Pada tahun 2021, Indonesia mengalami 3.058 kasus bencana alam, sebagian besar disebabkan oleh cuaca ekstrem, yang menyebabkan kerugian ekonomi besar dan mempengaruhi jutaan orang di seluruh Indonesia (Farissa et al., 2021).

Menghadapi tantangan ini, optimalisasi pemantauan data cuaca real-time menjadi sangat penting. Sistem pemantauan yang andal dan efisien dapat meningkatkan ketepatan prediksi cuaca dan respons terhadap kondisi cuaca ekstrem. Dengan menggunakan data dari OpenWeatherMap, yang menyediakan informasi cuaca secara *real-time*, serta teknologi seperti Apache Kafka dan Kibana, pemrosesan dan visualisasi data dapat dilakukan dengan lebih efektif. Apache Kafka memungkinkan pengumpulan dan distribusi data dalam jumlah besar secara real-time (Permatahati et al., 2023), sementara Kibana menyediakan pencarian dan visualisasi data yang telah diindeks di dalam Elasticsearch (Pradana Aji & Budi Cahyono, n.d. 2020), membantu dalam analisis dan pengambilan keputusan yang cepat dan akurat.

Kemampuan untuk merespons perubahan cuaca dengan cepat dan akurat sangat penting untuk mengurangi dampak negatif pada berbagai sektor (Pancasakti Makassar et al., 2023). Sektor pertanian dapat merencanakan penanaman dan panen dengan lebih baik, transportasi dapat mengantisipasi gangguan perjalanan, perencanaan kota dapat mengelola tata ruang dengan lebih efektif, dan manajemen bencana dapat meningkatkan kesiapsiagaan dan respons terhadap bencana alam. Dalam konteks perubahan iklim yang meningkatkan frekuensi dan intensitas cuaca ekstrem, sistem pemantauan cuaca yang efektif adalah alat vital untuk mitigasi risiko dan adaptasi.

Oleh karena itu, projek ini dilakukan untuk mengembangkan sistem optimalisasi pemantauan data cuaca real-time dari OpenWeatherMap menggunakan Apache Kafka dan Kibana. Diharapkan, dengan kombinasi teknologi ini, dapat tercipta sistem pemantauan cuaca yang lebih efisien dan akurat. Kemudian dapat memastikan pengumpulan, pemrosesan, dan visualisasi data cuaca dilakukan secara efektif untuk mendukung analisis dan pengambilan keputusan. Sehingga manfaatnya tidak hanya untuk mendukung sektor-sektor kritis seperti pertanian, transportasi, perencanaan kota, dan manajemen bencana, tetapi juga untuk memberikan kontribusi penting dalam upaya mitigasi dampak perubahan iklim dan meningkatkan kesiapsiagaan terhadap bencana alam di Indonesia.

## 1.2. Permasalahan

Dengan latar belakang tersebut, maka peneliti mengambil perumusan masalah sebagai berikut :

1. Bagaimana cara mengoptimalkan data cuaca secara *real-time* dengan OpenWeatherMap?
2. Bagaimana cara melakukan visualisasi data cuaca secara *real-time*?

## 1.3. Tujuan

Tujuan utama dilaksanakannya penelitian ini diantaranya :

1. Untuk mengoptimalkan data cuaca secara real-time dengan OpenWeatherMap menggunakan Apache Kafka untuk mengumpulkan datanya.

- Untuk melakukan visualisasi data cuaca secara real-time dengan menggunakan kibana sehingga data dapat dengan mudah dipahami dan memberikan knowledge.

#### **1.4. Manfaat**

##### **1. Untuk Industri**

Dengan penelitian ini diharap Apache Kafka dan visualisasi Kabana dapat membantu industri untuk memantau cuaca secara *real-time* sehingga industri dapat memprediksi dan mengantisipasi perubahan cuaca yang dapat mempengaruhi kegiatan industri. Dengan memanfaatkan Apache Kafka, industri dapat memantau kondisi cuaca secara *real-time* dan mengintegrasikan data cuaca tersebut ke dalam sistem operasional mereka.

##### **2. Untuk Ilmu Pengetahuan**

Dengan dilakukannya penelitian diharap dapat menambah wawasan bagi pembaca terutama mahasiswa untuk memantau cuaca secara *real-time*, memberikan kesempatan kepada siswa untuk belajar dan melakukan prediksi cuaca. Dengan akses ke data cuaca yang akurat dari situs OpenWeatherMap diharap dapat meningkatkan keterampilan analisis dan prediksi cuaca. Selain itu, diharap dapat pemahaman teoritis mengenai bagaimana menerapkan pengetahuan ini pada kegiatan sehari - hari dan sebagai bahan uji coba penelitian selanjutnya.

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1. Teori Penunjang**

##### **2.1.1. Apache Kafka**



**Gambar 1.** Logo Apache Kafka

Apache Kafka merupakan sebuah platform untuk *streaming* data yang memungkinkan pengguna untuk mengumpulkan, mengirim, dan menganalisis aliran data secara *real-time* (Permatahati et al., 2023). Apache Kafka mampu beroperasi secara efisien dalam suatu kelompok atau *cluster*, yang memungkinkan penambahan mesin pada *cluster* tanpa perlunya melakukan peningkatan kapasitas secara vertikal ketika volume dan kecepatan aliran data yang harus diproses meningkat (Nabawi, 2022).

##### **2.1.2. Kibana**



**Gambar 2.** Logo Kibana

Kibana merupakan sebuah alat yang menjadi bagian dari tumpukan ELK, yang terdiri dari *Elasticsearch*, *Logstash*, dan *Kibana* yang dikembangkan oleh perusahaan Elastic. Kibana memiliki sebuah fungsi sebagai platform visualisasi yang dibangun di atas Elasticsearch dan memanfaatkan kemampuan Elasticsearch untuk menampilkan data secara visual (Putra M. J. R., et al., 2022). Kibana menyediakan pencarian dan visualisasi data yang telah diindeks di dalam

Elasticsearch (Pradana Aji & Budi Cahyono, n.d., 2020) membantu dalam analisis dan pengambilan keputusan yang cepat dan akurat.

#### 2.1.3. Amazon Web Service (AWS)



**Gambar 3.** Logo AWS

Amazon Web Services (AWS) adalah salah satu *platform cloud* paling komprehensif dan banyak digunakan di dunia. AWS menyediakan layanan infrastruktur sebagai layanan (IaaS) dan platform sebagai layanan (PaaS). Layanan AWS menawarkan solusi yang dapat disesuaikan untuk komputasi, penyimpanan, basis data, analitik, dan berbagai kebutuhan lainnya (Suprayogi et al., n.d., 2023). Dua layanan utama AWS yang sering digunakan dalam pengelolaan konten blog adalah Amazon Elastic Compute Cloud (EC2) dan Amazon Simple Storage Service (S3). Amazon EC2 adalah layanan cloud computing yang menyediakan kapasitas komputasi dalam lingkungan cloud. Sementara itu, Amazon S3 adalah layanan penyimpanan objek yang memungkinkan penyimpanan dan pengelolaan file seperti gambar, video, dan dokumen (Harimurti & Udariansyah, 2023).

#### 2.1.4. OpenWeatherMap



**Gambar 4.** Logo OpenWeatherMap

*OpenWeatherMap* adalah salah satu layanan *Weather API* yang menyediakan data cuaca. *Weather API* adalah antarmuka khusus yang dirancang untuk aplikasi yang membutuhkan informasi cuaca(Sonna Mahardika et al., 2019). Layanan ini merupakan sebuah layanan *online* yang menyajikan informasi mengenai kondisi cuaca saat ini, termasuk ramalan cuaca dan data historis, yang tersedia bagi para pengembang web dan aplikasi mobile (Setiawan et al., 2019).

#### 2.1.5. *Application Programming Interface (API)*



**Gambar 5.** Logo API

API atau *Application Programming Interface* adalah pemrograman aplikasi yang dapat diakses oleh siapa saja, termasuk developer. Application Programming Interface atau singkatnya API adalah sebuah menyediakan konsep fungsionalitas untuk antarmuka pemrograman aplikasi. Dengan adanya API pengembang dapat membangun perangkat lunak tanpa harus memulainya dari awal. Selain itu, pemilik API dapat tetap menjaga kode mereka dan sekaligus mendapatkan lebih banyak perhatian untuk produk mereka. API ini memungkinkan aplikasi diakses dan digunakan oleh pihak lain serta memfasilitasi komunikasi antara sistem yang berbeda platform (Rakhmawati, N. A., et al. 2019). Web Service, sebagai salah satu bentuk API, memiliki peran penting dalam menyediakan sebuah terutama dalam proses pengambilan, pengiriman, dan pertukaran data antar sistem web.

Pengguna dapat mengambil data melalui sebuah Web Service yang berfungsi sebagai API yaitu penyedia akses pengambilan data di dalam web. Melalui arsitektur *Representational State Transfer* (ReST) yang diproses dan dijalankan menggunakan *Hypertext Transfer Protocol* (HTTP), Web Service menghasilkan output data dalam bentuk file Javascript Object Notation (JSON)

kepada pengguna saat mereka mengakses API tersebut (Warsito Budi A., et al., 2017).

#### 2.1.6. Pemrograman *Python*



**Gambar 6.** Logo *Python*

*Python* diciptakan oleh Guido van Rossum di Belanda pada tahun 1990. Bahasa pemrograman ini telah menjadi pilihan utama bagi banyak perusahaan besar dan pengembang untuk mengembangkan aplikasi desktop, web, dan mobile (Romzi & Kurniawan, 2020). Python merupakan bahasa pemrograman berbasis objek yang dapat interaksi secara langsung melalui mode interaktif(Ridho, & Niani., 2022). Bahasa pemrograman yang dirancang secara khusus untuk menyederhanakan proses pembuatan program bagi para pengembang, dengan fokus pada efisiensi waktu, kemudahan pengembangan, dan kompatibilitas dengan berbagai sistem. *Python* dapat digunakan untuk membuat berbagai jenis aplikasi, baik itu aplikasi mandiri maupun skrip pemrograman (Aqmila, 2022).

## 2.2. Penelitian Terkait

### 2.2.1. Implementasi Sistem Distribusi Pesan dan Proses Data Secara Real Time dengan Apache Kafka oleh Fezan Nabawi (2022).

Pada penelitian tersebut, penulis mengimplementasikan apache kafka dalam sistem distribusi pesan dan proses data secara *real-time*. Dalam penelitiannya disebutkan jika tempat penelitian dilakukan di PT. Adira Finance Jakarta, dan sistem operasi yang digunakan pada server Kafka adalah CentOS 7.0. Proses implementasi penelitian secara umum meliputi instalasi dan konfigurasi

Kafka, aktivasi server Kafka, pembuatan topik, serta pengujian aplikasi produsen dan konsumennya. Keseluruhan tahapan tersebut berhasil dilaksanakan dengan baik, sehingga memungkinkan distribusi pesan atau data secara real-time.

- 2.2.2. Penerapan Log Analyzer Log untuk Mengetahui Lalu Lintas Jaringan Berbasis Elasticsearch, Logstash, dan Kibana oleh Muhammad Jafier Rama Putra dan Henry Saptono (2022).

Pada penelitian tersebut, peneliti menerapkan log analyzer log untuk mengetahui lalu lintas jaringan berbasis elasticsearch, logstash, dan kibana. Penelitian ini bertujuan untuk menerapkan analisis log dan pemantauan lalu lintas jaringan menggunakan ELK Stack (Elasticsearch, Logstash, Kibana), yang meliputi pengiriman, pengumpulan, dan visualisasi log dari server. Dalam implementasi ini, desain yang digunakan untuk mengatur log dengan tepat menggunakan Elasticsearch, Logstash, dan Kibana. Dimana desain sistem pemantauan lalu lintas jaringan dengan *packetbeat* digunakan secara langsung ke Elasticsearch dan Kibana untuk hasilnya divisualisasinya.

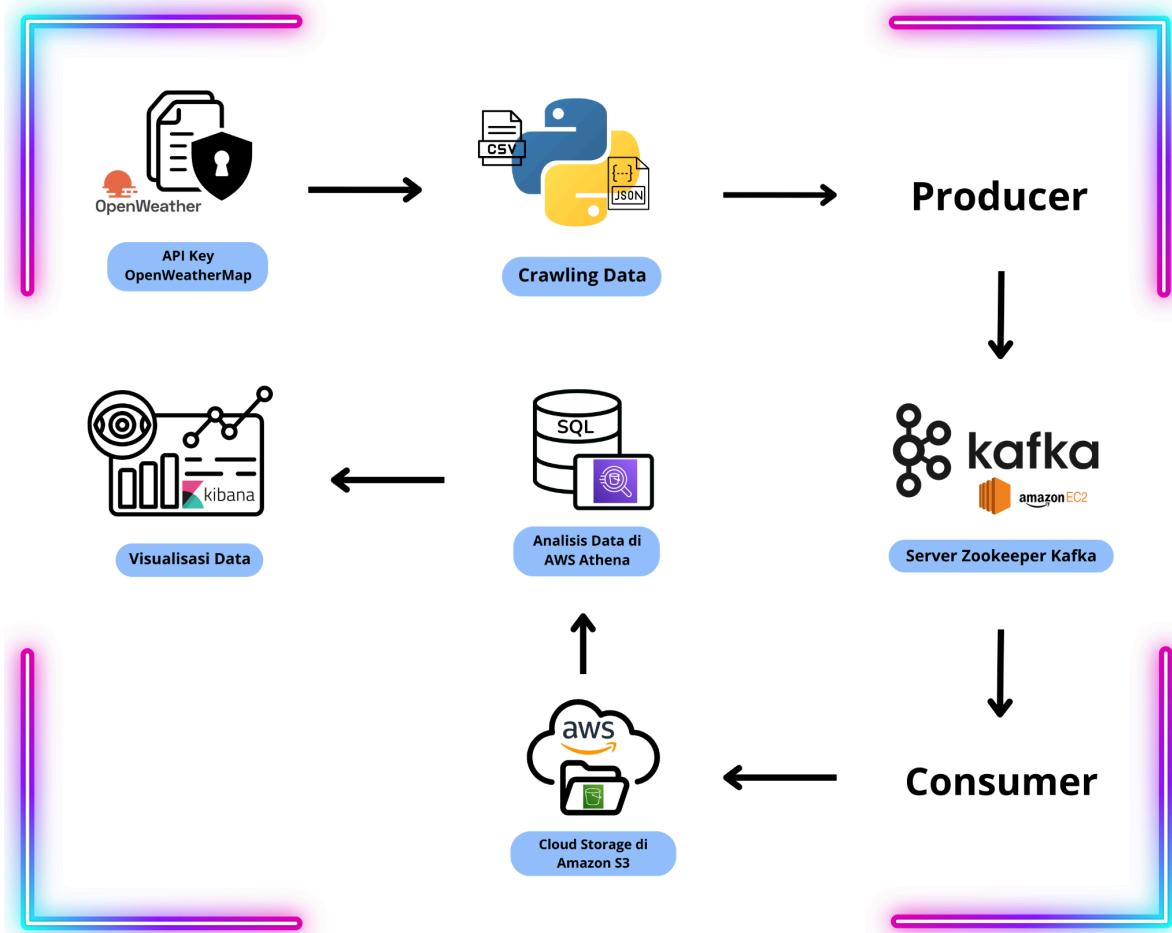
- 2.2.3. Implementasi Sistem Real Time untuk Pendekripsi Dini Banjir berbasis ESP8266 dan Weather API oleh Shandi Sonna Mahardika, Wijaya Kurniawan, dan Fariz Andri Bakhtiar (2019)

Pada penelitian tersebut, penulis mengimplementasikan sistem *real-time* untuk mendekripsi banjir berbasis ESP8266 dan Weather API. Dalam penelitiannya disebutkan jika sistem peringatan dini banjir menggunakan Weather API, sensor HC-SR04, modul FC-37, dan sensor YF-S201. Sensor dan Weather API dalam sistem ini harus beroperasi secara bersamaan. Sistem operasi real-time (RTOS) menyediakan kemampuan multitasking dan penjadwalan untuk menjalankan beberapa program secara simultan. Perancangan cuaca pada Weather API mengacu pada ID cuaca, di mana semakin rendah nilainya, semakin tinggi intensitas hujan, dan semakin tinggi nilainya, semakin rendah intensitas hujan.

## BAB III

### METODOLOGI PENELITIAN

Pada bab ini, akan dijelaskan mengenai metode yang digunakan dalam penelitian ini untuk mengoptimalkan pemantauan data cuaca real-time dari OpenWeatherMap menggunakan Apache Kafka dan Kibana. Setiap tahap dijelaskan secara rinci untuk memberikan gambaran yang jelas mengenai alur kerja dan implementasi dari keseluruhan sistem. Setiap tahap dalam metode penelitian ini dijelaskan secara detail untuk memberikan pemahaman yang komprehensif tentang proses dan teknologi yang digunakan dalam penelitian ini. Flowchart yang terlampir di bawah ini memberikan ilustrasi visual dari alur kerja penelitian ini, mulai dari pengambilan data hingga visualisasi akhir.



Gambar 7. Flowchart Alur Penelitian

### **3.1. Mendapatkan API Key dari OpenWeatherMap**

Langkah pertama yang harus dilakukan yaitu mendapatkan API key dari OpenWeatherMap yang akan digunakan untuk mengakses data cuaca secara real-time. API key adalah sebuah kode unik yang digunakan untuk mengidentifikasi dan mengautentikasi aplikasi atau pengguna yang mengakses sebuah Application Programming Interface (API). API key bertindak sebagai pengaman yang memastikan bahwa hanya aplikasi atau pengguna yang memiliki izin yang dapat mengakses data atau layanan yang disediakan oleh API. Untuk mendapatkan API key OpenWeatherMap, terlebih dahulu mendaftar pada situs OpenWeatherMap di <https://openweathermap.org/> lalu membuat akun baru jika belum memiliki akun OpenWeatherMap. Kemudian pada halaman API keys pilih opsi membuat kunci API baru dan berikan nama deskriptif untuk API key lalu klik tombol generate atau create. Setelah API key berhasil dibuat, salin API key karena akan digunakan dalam script Python untuk crawling data cuaca dari OpenWeatherMap.

### **3.2. Crawling Data Cuaca Real-time**

Setelah mendapatkan API key dari OpenWeatherMap, lalu mengembangkan script Python untuk melakukan crawling data cuaca secara real-time dari OpenWeatherMap. Sebelum script Python mulai dikembangkan, siapkan library-library yang diperlukan untuk melakukan crawling data seperti ‘request’ dan ‘json’ untuk mengirim permintaan HTTP dan memproses data yang diterima. Setelah itu mengakses API WeatherMap menggunakan endpoint cuaca saat ini, yang memungkinkan mengambil data cuaca secara real-time berdasarkan wilayah tertentu yang diinginkan. Script Python yang dibuat untuk mengcrawling data harus ditulis secara benar dan rinci agar data cuaca yang berhasil didapatkan dapat diolah dan diakses dengan benar

### **3.3. Membuat Producer yang Mengirimkan Data Cuaca ke Topik Kafka**

Pada tahap ini, tujuan utama adalah mengirimkan data cuaca yang telah di-crawling ke topik Kafka untuk pemrosesan lebih lanjut oleh berbagai aplikasi. Langkah pertama adalah instalasi Apache Kafka dan Zookeeper pada sistem yang digunakan, diikuti dengan konfigurasi Kafka untuk memastikan

efisiensi pengelolaan dan penyimpanan pesan. Setelah konfigurasi, dibuat topik baru di Kafka untuk menampung data cuaca. Langkah selanjutnya adalah mengembangkan producer dengan bahasa Python. Script Python ini akan membaca data cuaca yang di-crawling dan mengirimkannya sebagai pesan ke topik Kafka yang telah dibuat.

Implementasi pengiriman data dilakukan dengan menghubungkan producer ke broker Kafka, memformat data cuaca yang sesuai, dan mengirimkan data tersebut ke topik. Script ini juga menangani kesalahan selama pengiriman untuk memastikan keandalan sistem. Setelah pengembangan, dilakukan pengujian untuk memastikan bahwa data cuaca dikirim dengan benar ke topik Kafka dan dapat dikonsumsi oleh consumer, serta memantau kinerja producer untuk memastikan tidak ada pesan yang hilang atau tertunda.

### **3.4. Mengintegrasikan Data Cuaca dengan Apache Kafka dan AWS**

Pada tahap ini, fokus utamanya adalah mengintegrasikan data cuaca dengan Apache Kafka dan layanan cloud Amazon Web Services (AWS) untuk memastikan alur data yang efisien dari sumber data hingga penyimpanan dan analisis. Proses integrasi dimulai dengan konfigurasi AWS untuk menerima data dari Kafka, menggunakan layanan seperti AWS Lambda untuk memproses data secara real-time yang diterima dari topik Kafka. Setelah pemrosesan, data yang telah diolah dikirim ke Amazon S3 untuk penyimpanan yang aman dan mudah diakses. Integrasi ini memungkinkan data cuaca yang dikirimkan melalui Kafka dapat langsung diproses, disimpan, dan dianalisis menggunakan berbagai layanan AWS, memastikan ketersediaan data secara real-time untuk berbagai aplikasi dan pengguna akhir.

### **3.5. Membuat Consumer untuk Menerima Data Cuaca dari Topik Kafka**

Consumer adalah komponen yang bertanggung jawab untuk menerima data cuaca yang telah dikirimkan ke topik Kafka. Dalam konteks ini, consumer dapat diimplementasikan sebagai aplikasi yang berjalan di atas infrastruktur AWS, seperti menggunakan layanan EC2. Aplikasi ini akan terhubung ke topik Kafka yang telah dibuat sebelumnya dan akan membaca

pesan-pesan yang dikirimkan oleh produsen. Setelah menerima data cuaca, consumer dapat menjalankan berbagai tindakan, seperti menyimpan data ke penyimpanan awan (misalnya Amazon S3), melakukan pemrosesan lanjutan, atau mengirim data ke layanan lain untuk analisis lebih lanjut.

### **3.6. Cloud Storage Data pada Amazon S3**

Amazon S3 adalah layanan penyimpanan awan yang sangat *scalable* dan aman yang digunakan untuk menyimpan data cuaca yang telah diolah atau mentah. Setelah data cuaca diterima oleh consumer dari topik Kafka, langkah selanjutnya adalah menyimpan data tersebut di S3. Data cuaca dapat disimpan dalam berbagai format yang sesuai, seperti JSON, di dalam bucket S3 yang telah dibuat sebelumnya. Penyimpanan data cuaca di S3 memungkinkan akses yang mudah dan cepat untuk aplikasi lain yang membutuhkan data tersebut, serta memfasilitasi analisis dan pemrosesan lebih lanjut menggunakan layanan AWS lainnya.

### **3.7. Analisis Data dengan AWS Athena Dalam Bentuk Query SQL**

Pada tahap ini, tugas utamanya adalah menganalisis data cuaca yang sudah disimpan pada Amazon S3 menggunakan AWS Athena. AWS Athena merupakan layanan query interaktif yang memungkinkan analisis data di Amazon S3 menggunakan SQL. Prosesnya terdiri dari pembuatan tabel di Athena yang mengarah ke lokasi data di S3, kemudian menjalankan query SQL untuk mengekstrak dan menganalisis informasi cuaca yang diperlukan. Hasil dari analisis dapat membantu mengidentifikasi tren, pola, dan anomali cuaca.

### **3.8. Visualisasi Data Cuaca dengan Apache Kibana**

Pada tahap akhir dari metodologi penelitian ini, tujuan utama dari tahap ini adalah untuk membuat visualisasi data cuaca yang telah dianalisis sebelumnya menggunakan Kibana. Kibana merupakan alat visualisasi *open source* yang sudah terintegrasi dengan Elasticsearch. Proses ini melibatkan ElasticSearch untuk menyimpan hasil dari query dari AWS Athena, kemudian membuat berbagai dashboard dan visualisasi data cuaca menggunakan Kibana.

Visualisasi ini akan memberikan wawasan mendalam mengenai kondisi cuaca secara real-time dan pengambilan keputusan yang lebih baik berdasarkan data cuaca yang sudah dianalisis. Sebelum masuk kibana perlu menginstall paket elasticsearch, kemudian menjalankan elasticsearch pada command dan dilanjut dengan menjalankan kibana. Setelah berhasil masuk dalam kibana, maka untuk melakukan visualisasi, masukkan data terlebih dahulu kemudian di simpan pada kibana, setelah itu membuat dashboard baru. Langkah selanjutnya melakukan visualisasi dan menyimpannya dengan memberi judul dashboard.

## BAB IV

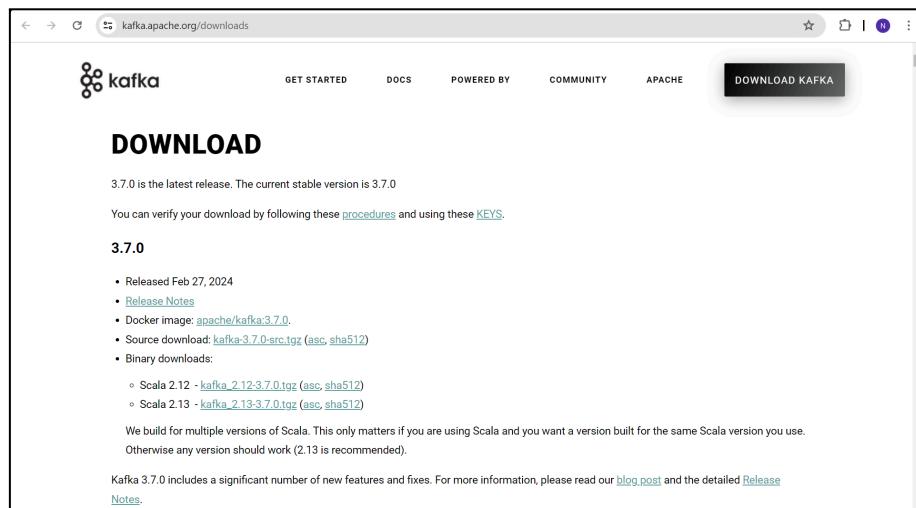
# HASIL DAN PEMBAHASAN

Dalam bab ini, akan dibahas hasil dan implementasi dari dua komponen utama dalam proyek ini, yaitu Apache Kafka dan Kibana. Pada bagian ini, akan diuraikan langkah-langkah yang dilakukan dalam menerapkan kedua teknologi tersebut dalam proyek ini, serta menganalisis hasil yang diperoleh dari penggunaannya. Dengan demikian, pembahasan ini diharapkan dapat memberikan pemahaman yang lebih mendalam tentang bagaimana teknologi-teknologi tersebut diimplementasikan dalam data cuaca secara *real-time* dari OpenWeatherMap.

### 4.1. Implementasi Apache Kafka

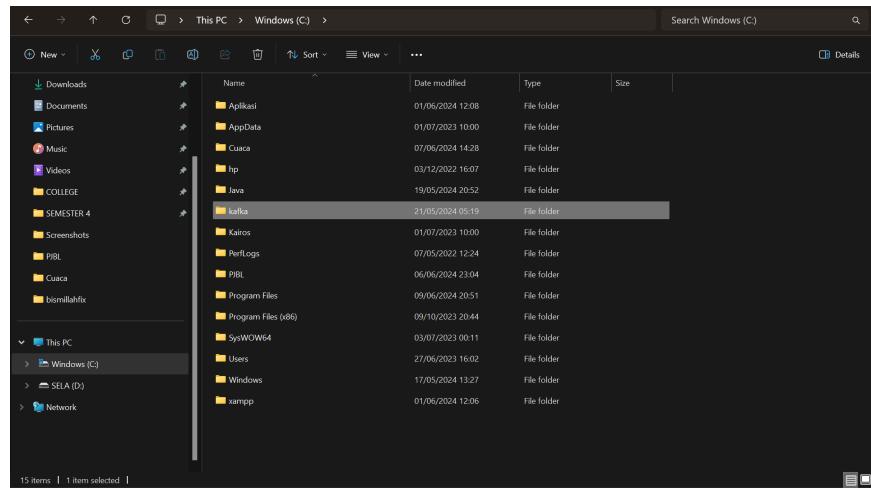
#### 4.1.1. Install dan Mengaktifkan Apache Kafka

Sebelum menginstal Apache Kafka, unduh versi terbaru **Scala 2.13** dari Apache Kafka melalui tautan berikut <https://kafka.apache.org/downloads>.



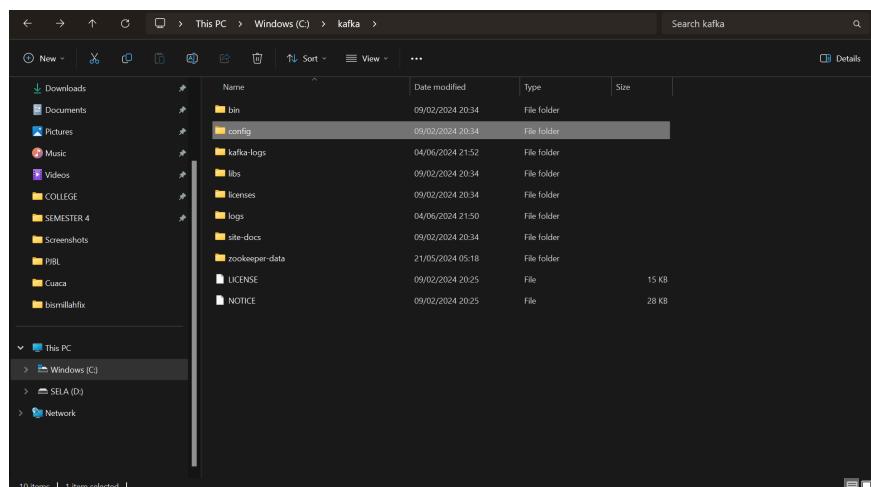
Gambar 8. Install Kafka

Setelah selesai mengunduh, buka direktori tempat file unduhan disimpan. Ekstrak file zip tersebut dan ganti nama folder hasil ekstrak menjadi **kafka** untuk mempermudah. Kemudian, pindahkan folder **kafka** tersebut ke drive 'C'.



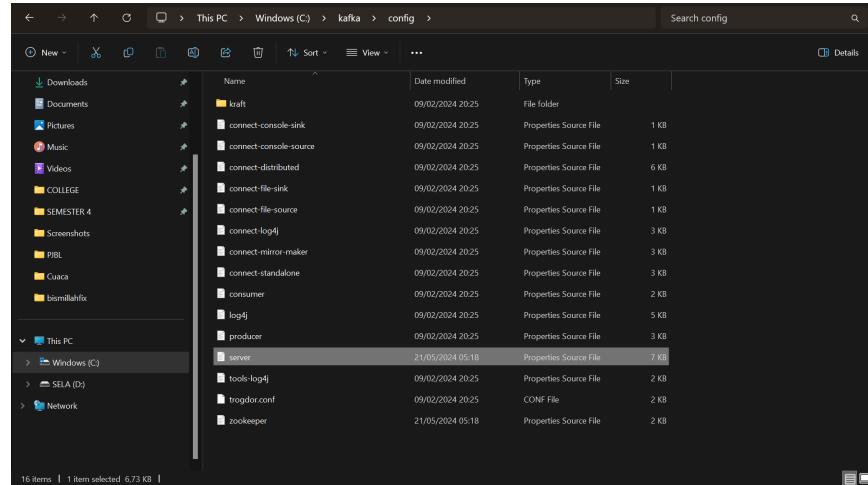
Gambar 9. Memindahkan ke drive C

Selanjutnya, melakukan beberapa langkah konfigurasi untuk menginstal Kafka di Windows. Arahkan ke folder **config** di dalam folder **kafka** seperti yang ditunjukkan di bawah ini.



Gambar 10. Konfigurasi untuk install

Langkah selanjutnya, buka file **server** menggunakan editor teks Notepad. Caranya, klik kanan pada file tersebut, pilih opsi 'Buka dengan', lalu pilih Notepad sebagai editor.



**Gambar 11.** Membuka file server

Untuk mengubah direktori log Kafka, temukan baris '`log.dirs=/tmp/kafka-logs`' dalam file **server**. Ganti '`/tmp/kafka-logs`' menjadi '`C:/kafka/kafka-logs`'. Setelah itu, simpan perubahan yang telah dilakukan.

```

# The receive buffer (SO_RCVBUF) used by the socket server
socket.receive.buffer.bytes=102400
# The maximum size of a request that the socket server will accept (protection against OOM)
socket.request.max.bytes=104857600

#####
# A comma separated list of directories under which to store log files
log.dirs=C:/kafka/kafka-logs

# The default number of log partitions per topic. More partitions allow greater
# parallelism for consumption, but this will also result in more files across
# the brokers.
num.partitions=1

# The number of threads per data directory to be used for log recovery at startup and flushing at shutdown.
# This value is recommended to be increased for installations with data dirs located in RAID array.
num.recovery.threads.per.data.dir=1

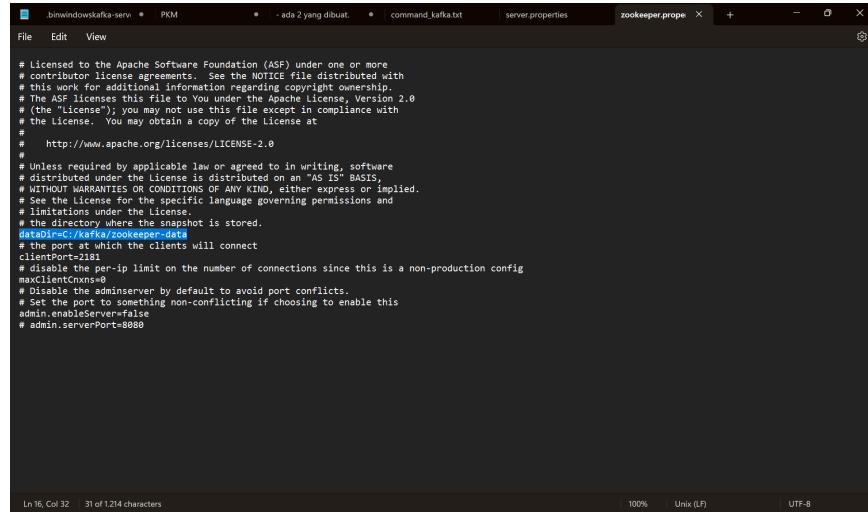
#####
# Internal Topic Settings #####
# The replication factor for the group metadata internal topics "_consumer_offsets" and "_transaction_state"
# For anything other than development testing, a value greater than 1 is recommended to ensure availability such as 3.
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.lsr=1

#####
# Log Flush Policy #####
# Messages are immediately written to the filesystem but by default we only fsync() to sync
# the OS cache briefly. The following configurations control the flush of data to disk.
# These settings fall into three categories:
# 1. Durability: Unflushed data may be lost if you are not using replication.
# 2. Latency: Very large flush intervals may lead to latency spikes when the flush does occur as there will be a lot of data to flush.
# 3. Throughput: The flush is generally the most expensive operation, and a small flush interval may lead to excessive seeks.
# The settings below allow one to configure the flush policy to flush data after a period of time or
# a specific number of bytes.

```

**Gambar 12.** Ubah direktori kafka

Sama seperti sebelumnya, buka file **zookeeper**. menggunakan Notepad. Cari baris yang berisi '`dataDir=/tmp/zookeeper`' dan diubah menjadi '`dataDir=C:/kafka/zookeeper-data`'. Lalu simpan perubahan yang telah dilakukan.



```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor licenses. See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License. You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the license.
dataDir=C:/kafka/zookeeper-data
# the port at which the clients will connect
port=2181
# disable the per-ip limit on the number of connections since this is a non-production config
maxClientCnns=0
# Disable the adminserver by default to avoid port conflicts.
# Set the port to something non-conflicting if choosing to enable this
admin.enableServer=false
# admin.serverPort=8080
```

**Gambar 13.** Ubah direktori zookeeper

Langkah selanjutnya adalah mengaktifkan Zookeeper melalui terminal cmd. Pertama, membuka Command Prompt (cmd). Kemudian, arahkan ke direktori Kafka. Setelah itu, perintah berikut dimasukkan untuk menjalankan Zookeeper.

```
cd C:\kafka
.\bin\windows\zookeeper-server-start.bat
.\config\zookeeper.properties
```

**Tabel 1.** Buka command

Server Zookeeper akan dimulai setelah menjalankan perintah di atas. Setelah dijalankan akan terlihat proses startup Zookeeper di Command Prompt, yang menandakan bahwa Zookeeper telah berhasil dijalankan. Output yang muncul terlihat pada gambar berikut.

**Gambar 14.** Mulai server zookeeper

Selanjutnya, buka jendela Command Prompt (cmd) baru untuk mengaktifkan server Kafka. Arahkan ke folder **kafka** dan perintah berikut dijalankan untuk memulai server Kafka.

```
cd C:\kafka  
  
.\\bin\\windows\\kafka-server-start.bat  
  .\\config\\server.properties
```

**Tabel 2.** Aktifkan kafka

Server Kafka akan mulai berjalan, dan output di Command Prompt akan menunjukkan bahwa server Kafka sedang aktif, seperti pada gambar di bawah ini.

**Gambar 15.** Kafka dimulai

#### 4.1.2. Proses Crawling Data Cuaca dari OpenWeathermap

a. Mendapatkan API key dari OpenWeatherMap

Untuk memulai penggunaan layanan OpenWeatherMap, langkah pertama yang perlu dilakukan adalah mendaftar di situs web <https://openweathermap.org/api>. Proses pendaftaran ini melibatkan pembuatan akun pengguna di platform OpenWeatherMap. Setelah mendaftar, diberikan sebuah kunci API yang unik dikirim lewat email. Kunci API ini mirip dengan kunci yang digunakan untuk membuka pintu, karena memberikan akses ke layanan cuaca mereka. API key ini adalah cara utama bagi OpenWeatherMap untuk mengidentifikasi dan mengotorisasi permintaan data cuaca.

Dengan menggunakan API key yang diperoleh saat mendaftar, akan dapat membuat permintaan ke server OpenWeatherMap dan mengambil data cuaca dari berbagai lokasi di seluruh dunia. API key ini juga membantu OpenWeatherMap melacak penggunaan layanan mereka dan memberlakukan batasan dan kebijakan yang sesuai. Dengan demikian, langkah pertama yang penting sebelum mengakses data cuaca melalui OpenWeatherMap adalah mendaftar dan mendapatkan API key mereka.

b. Membuat Kode Python Untuk Mengambil Data Cuaca dari OpenWeatherMap dan Mengirimkannya Ke Kafka

Kode ini digunakan untuk melakukan polling data cuaca secara real time dari OpenWeatherMap menggunakan API mereka, kemudian mengirim data tersebut ke Kafka dan menyimpannya dalam file CSV.

```
import requests
import json
import time
import csv
import os
from kafka import KafkaProducer

# Fungsi untuk inisialisasi Kafka Producer
def inisialisasi_kafka_producer():
    try:
        producer = KafkaProducer(
            bootstrap_servers='localhost:9092',
            value_serializer=lambda v:
            json.dumps(v).encode('utf-8')
        )
        print("Kafka Producer berhasil
diinisialisasi.")
        return producer
    except Exception as e:
        print(f"Gagal menginisialisasi Kafka
Producer: {e}")
        return None

# API Key dan URL untuk API Cuaca Realtime
WeatherAPI
api_key = '81c93728c6cf49f496052543242005'
base_url =
"http://api.weatherapi.com/v1/current.json"

# Fungsi untuk mengambil data cuaca dari
WeatherAPI
def ambil_data_cuaca(lokasi):
    try:
        url =
f"{base_url}?key={api_key}&q={lokasi}"
        response = requests.get(url)
        response.raise_for_status()  #
Memunculkan error untuk status kode yang buruk
        data = response.json()
        return data
    except requests.exceptions.RequestException
```

```

as e:
    print(f"Gagal mengambil data dari
WeatherAPI: {e}")
    return None
# Fungsi untuk mengirim data ke Kafka dan
menyimpan ke CSV
def kirim_ke_kafka(producer, data, topik,
csv_writer):
    if producer and data:
        try:
            producer.send(topik, data)
            print(f"Terkirim ke Kafka: {data}")

            # Simpan data ke CSV
            location = data['location']
            current = data['current']
            csv_writer.writerow([
                location['name'],
                location['region'],
                location['country'],
                location['lat'],
                location['lon'],
                location['tz_id'],
                location['localtime'],
                current['last_updated'],
                current['temp_c'],
                current['temp_f'],
                current['is_day'],
                current['condition']['text'],
                current['condition']['code'],
                current['wind_mph'],
                current['wind_kph'],
                current['wind_degree'],
                current['pressure_in'],
                current['precip_in'],
                current['humidity'],
                current['cloud'],
                current['feelslike_c'],
                current['uv'],
                current['gust_mph'],
                current['gust_kph']
            ])
        except Exception as e:
            print(f"Gagal mengirim data ke
Kafka: {e}")

# Loop utama untuk polling data secara berkala
def main():
    topik = 'cuaca-realtime' # Ganti dengan
    nama topik Kafka Anda

```

```

lokasi_surabaya = 'Surabaya' # Lokasi
Surabaya
lokasi_malang = 'Malang' # Lokasi Malang
max_iterasi = 500 # Tentukan jumlah iterasi
maksimum sebelum berhenti
interval_polling = 60 # Interval polling
dalam detik
iterasi = 0

producer = inisialisasi_kafka_producer()
if not producer:
    print("Kafka Producer tidak dapat
diinisialisasi. Keluar.")
    return

# Nama file CSV
csv_file = 'cuaca_realtime_sby_mlg.csv'

# Periksa apakah file CSV sudah ada
file_exists = os.path.isfile(csv_file)

# Buka file CSV untuk ditulis
with open(csv_file, mode='a', newline='') as
file:
    csv_writer = csv.writer(file)

    # Tulis header jika file baru
    if not file_exists:
        csv_writer.writerow([
            'name', 'region', 'country',
            'lat', 'lon', 'tz_id', 'localtime',
            'last_updated', 'temp_c',
            'temp_f', 'is_day', 'condition', 'code',
            'wind_mph', 'wind_kph',
            'wind_degree', 'pressure_in',
            'precip_in', 'humidity', 'cloud', 'feelslike_c',
            'uv', 'gust_mph', 'gust_kph'
        ])

    while iterasi < max_iterasi:
        start_time = time.time() # Waktu
awal sebelum polling

        # Ambil data cuaca untuk Surabaya
        data_surabaya =
ambil_data_cuaca(lokasi_surabaya)
        if data_surabaya:
            kirim_ke_kafka(producer,
data_surabaya, topik, csv_writer)

```

```

        # Ambil data cuaca untuk Malang
        data_malang =
ambil_data_cuaca(lokasi_malang)
        if data_malang:
            kirim_ke_kafka(producer,
data_malang, topik, csv_writer)

        # Hitung waktu yang dibutuhkan untuk
polling
        elapsed_time = time.time() -
start_time
            sleep_time = max(0, interval_polling
- elapsed_time) # Pastikan sleep_time tidak
negatif
            time.sleep(sleep_time)

        iterasi += 1

    print("Polling selesai. Menutup Kafka
Producer.")
    producer.close()

if __name__ == "__main__":
    main()

```

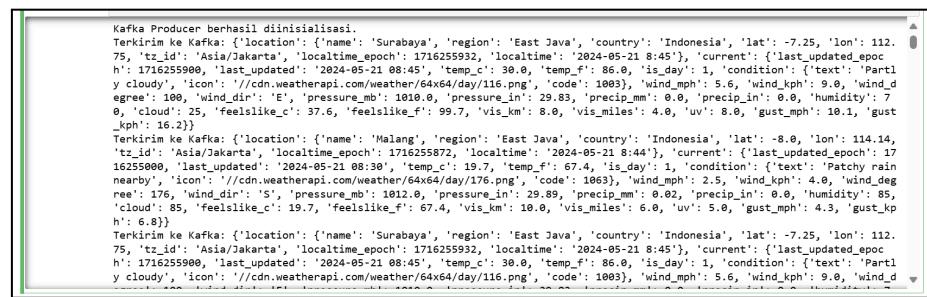
**Tabel 3.** Pooling data real-time

Kode Python di atas dapat dijelaskan secara rinci sebagai berikut :

1. Program mengimpor beberapa modul Python yang diperlukan, seperti **requests**, **json**, **time**, **csv**, **os**, dan **KafkaProducer** dari modul kafka.
2. Fungsi **inisialisasi\_kafka\_producer()** digunakan untuk menginisialisasi Kafka Producer dengan pengaturan yang diberikan, seperti server bootstrap dan serialisasi nilai ke format JSON.
3. Konstanta yang mendefinisikan API key OpenWeatherMap, URL untuk permintaan API, nama topik Kafka, lokasi (Surabaya dan Malang), jumlah iterasi maksimum, dan interval polling.
4. Fungsi **ambil\_data\_cuaca(lokasi)** bertanggung jawab untuk melakukan permintaan HTTP ke API OpenWeatherMap dan mengambil data cuaca untuk lokasi Surabaya dan Malang.

5. Fungsi **`kirim_ke_kafka(producer, data, topik, csv_writer)`** mengirim data cuaca ke Kafka menggunakan produsen Kafka yang telah diinisialisasi sebelumnya. Selain itu, data juga disimpan ke file CSV menggunakan objek penulis CSV yang diberikan.
6. Fungsi **`main()`** adalah loop utama program yang melakukan polling data cuaca secara berkala untuk Surabaya dan Malang, lalu mengirimkan data ke Kafka dan menyimpannya ke file CSV.
7. Selama iterasi, program mengambil data cuaca untuk Surabaya dan Malang, mengirimkan data ke Kafka, dan menunggu interval polling sebelum melakukan polling berikutnya.
8. Setelah selesai, program menutup produsen Kafka dan mengakhiri eksekusi.

Berikut adalah hasil output dari proses crawling data menggunakan kode di atas.



```

Kafka Producer berhasil diinisialisasi.
Terkirim ke Kafka: {"location": {"name": "Surabaya", "region": "East Java", "country": "Indonesia", "lat": -7.25, "lon": 112.75, "tz_id": "Asia/Jakarta"}, "localtime_epoch": 1716255932, "localtime": "2024-05-21 08:45"}, "current": {"last_updated_epoch": 1716255900, "last_updated": "2024-05-21 08:45", "temp_c": 30.0, "temp_f": 86.0, "is_day": 1, "condition": {"text": "Partly cloudy", "icon": "//cdn.weatherapi.com/weather/64x64/day/116.png", "code": 1003}, "wind_mph": 5.6, "wind_kph": 9.0, "wind_deg": 100, "wind_dir": "E", "pressure_mb": 1010.0, "pressure_in": 29.83, "precip_mm": 0.0, "precip_in": 0.0, "humidity": 70, "cloud": 25, "feelslike_c": 37.6, "feelslike_f": 99.7, "vis_km": 8.0, "vis_miles": 4.0, "uv": 8.0, "gust_mph": 10.1, "gust_kph": 16.2}}
Terkirim ke Kafka: {"location": {"name": "Malang", "region": "East Java", "country": "Indonesia", "lat": -8.0, "lon": 114.14, "tz_id": "Asia/Jakarta"}, "localtime_epoch": 1716255872, "localtime": "2024-05-21 08:44"}, "current": {"last_updated_epoch": 1716255900, "last_updated": "2024-05-21 08:30", "temp_c": 19.7, "temp_f": 67.4, "is_day": 1, "condition": {"text": "Patchy rain nearby", "icon": "//cdn.weatherapi.com/weather/64x64/day/176.png", "code": 1063}, "wind_mph": 2.5, "wind_kph": 4.0, "wind_deg": 176, "wind_dir": "S", "pressure_mb": 1012.0, "pressure_in": 29.89, "precip_mm": 0.02, "precip_in": 0.0, "humidity": 85, "cloud": 85, "feelslike_c": 19.7, "feelslike_f": 67.4, "vis_km": 10.0, "vis_miles": 6.0, "uv": 5.0, "gust_mph": 4.3, "gust_kph": 6.8}}
Terkirim ke Kafka: {"location": {"name": "Surabaya", "region": "East Java", "country": "Indonesia", "lat": -7.25, "lon": 112.75, "tz_id": "Asia/Jakarta"}, "localtime_epoch": 1716255932, "localtime": "2024-05-21 08:45"}, "current": {"last_updated_epoch": 1716255900, "last_updated": "2024-05-21 08:45", "temp_c": 30.0, "temp_f": 86.0, "is_day": 1, "condition": {"text": "Partly cloudy", "icon": "//cdn.weatherapi.com/weather/64x64/day/116.png", "code": 1003}, "wind_mph": 5.6, "wind_kph": 9.0, "wind_deg": 100, "wind_dir": "E", "pressure_mb": 1010.0, "pressure_in": 29.83, "precip_mm": 0.0, "precip_in": 0.0, "humidity": 70, "cloud": 25, "feelslike_c": 37.6, "feelslike_f": 99.7, "vis_km": 8.0, "vis_miles": 4.0, "uv": 8.0, "gust_mph": 10.1, "gust_kph": 16.2}}

```

Gambar 16. Pooling data real-time

Setelah melakukan crawling data, hasilnya juga disimpan dalam format CSV. Berikut adalah hasil dari penyimpanan data bentuk CSV.

Gambar 17. Data Dalam bentuk CSV

Proses crawling data berhasil mengumpulkan total 996 data.

Berikut adalah penjelasan untuk setiap variabel yang terdapat dalam data tersebut.

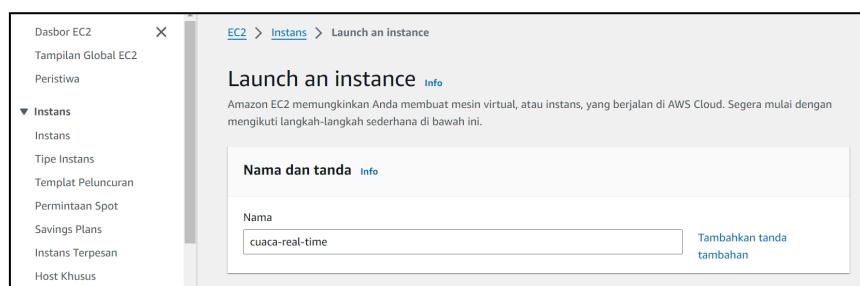
1. **name** : Nama lokasi, dalam hal ini adalah "Surabaya" dan "Malang".
2. **region** : Wilayah atau provinsi tempat lokasi berada, yaitu "East Java".
3. **country** : Negara tempat lokasi berada, yaitu "Indonesia".
4. **lat** : Lintang geografis lokasi.
5. **lon** : Bujur geografis lokasi.
6. **tz\_id** : Zona waktu lokasi.
7. **localtime** : Waktu lokal dalam format yang mudah dibaca, yaitu "21/05/2024 08:45:00".
8. **last\_updated\_epoch** : Waktu terakhir data diperbarui.
9. **last\_updated** : Waktu terakhir data diperbarui dalam format yang mudah dibaca, yaitu "21/05/2024 08:45:00".
10. **temp\_c** : Suhu saat ini dalam derajat Celcius.
11. **temp\_f** : Suhu saat ini dalam derajat Fahrenheit.
12. **is\_day** : Indikator apakah saat ini siang hari (1) atau malam hari (0).
13. **condition** : Objek yang berisi kondisi cuaca saat ini:
14. **text** : Deskripsi kondisi cuaca.
15. **code** : Kode kondisi cuaca.
16. **wind\_mph** : Kecepatan angin dalam mil per jam.
17. **wind\_kph** : Kecepatan angin dalam kilometer per jam.

18. **wind\_degree** : Arah angin dalam derajat.
19. **pressure\_in** : Tekanan atmosfer dalam inci.
20. **precip\_in** : Curah hujan dalam inci.
21. **humidity** : Kelembaban relatif dalam persen.
22. **cloud** : Persentase cakupan awan.
23. **feelslike\_c** : Suhu yang terasa dalam derajat Celcius.
24. **uv** : Indeks UV.
25. **gust\_mph** : Kecepatan hembusan angin dalam mil per jam.
26. **gust\_kph** : Kecepatan hembusan angin dalam kilometer per jam.

#### 4.1.3. Menghubungkan Kafka dengan AWS

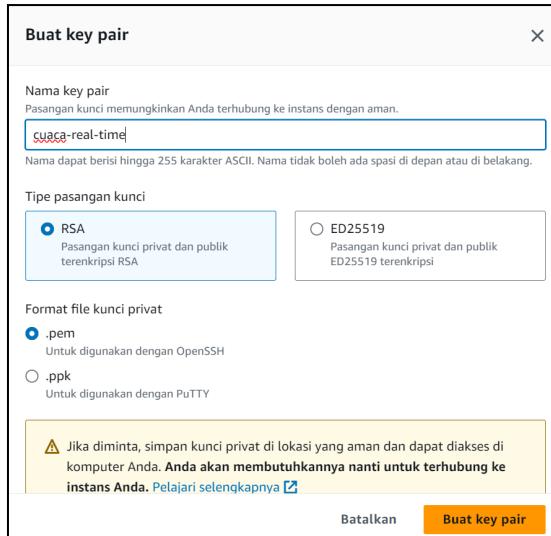
- a. Membuat instance dalam akun AWS yang sudah dibuat

Menghubungkan Kafka dengan AWS melibatkan beberapa langkah, termasuk pembuatan instance EC2 di akun AWS. Untuk memulai, perlu mendaftar akun AWS. Selanjutnya, membuka console EC2 AWS dan navigasi ke bagian **Instances**. Di sana, dapat memilih opsi **Launch Instance** untuk membuat instance baru.



**Gambar 18.** Hubungkan kafka ke AWS

Selanjutnya, memberikan nama instance, yaitu **cuaca-real-time**. Disarankan untuk menggunakan tipe instance T2 Micro karena tidak dikenakan biaya. Selama proses pembuatan instance, akan diminta untuk membuat key pair baru.



**Gambar 19.** Memberi nama instance

Disarankan untuk menamainya sesuai dengan nama instance yang sudah dibuat, yaitu **cuaca-real-time**, karena kunci ini akan digunakan untuk mengakses mesin EC2 dari komputer lokal. Setelah membuat key pair, sebuah file dalam format PEM akan terunduh secara otomatis. File ini akan digunakan untuk menyambungkan ke mesin EC2 dari komputer lokal.

Instans (1) Info		Hubungkan	Status instans	Tindakan	Luncurkan instans
<input type="text"/> Temukan Instans berdasarkan atribut atau tanda (case-sensitive)			Semua status		
ID Instans = i-09bba03c7bc70ffdc	X	Hapus filter			
Name	ID Instans	Status instans	Tipe instans	Pemeriksaan stati	Status alarm
cuaca-real-time	i-09bba03c7bc70ffdc	Menjalankan	t2.micro	Menginisialisasi...	Lihat alarm +

**Gambar 20.** Pemberian nama telah berhasil

Selanjutnya, klik **Launch Instance** dan memilih **ID Instance**, periksa apakah statusnya sedang **menunggu**. Setelah status berubah menjadi **menjalankan**, dapat melanjutkan dan melakukan eksekusi sebenarnya. Artinya, instance sudah aktif dan siap digunakan. Klik pada **ID instance**, lalu klik **Connect**. Dapat menggunakan perintah ‘ssh’ untuk terhubung ke instance EC2. Jika menggunakan Windows, buka CMD dan navigasikan ke folder tempat menyimpan key pair .PEM yang telah diunduh sebelumnya.

- b. Menyambungkan AWS dengan terminal

```
Microsoft Windows [Version 10.0.22621.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Selayanti>cd C:\PJBL
```

**Gambar 21.** Menyambungkan AWS dengan cmd

Buka Command Prompt (CMD) di Windows. Navigasikan ke direktori tempat menyimpan file key pair PEM.

```
C:\PJBL>ssh -i "cuaca-real-time.pem" ec2-user@ec2-3-107-41-163.ap-southeast-2.compute.amazonaws.com
# 
# Amazon Linux 2023
# 
# https://aws.amazon.com/linux/amazon-linux-2023
# 
# /m/
```

**Gambar 22.** Navigasi ke direktori penyimpanan

Perintah ini digunakan untuk menghubungkan ke instance EC2 di AWS dengan menggunakan key pair PEM untuk autentikasi, nama pengguna default `ec2-user`, dan hostname instance diberikan oleh AWS.

```
[ec2-user@ip-172-31-41-250 ~]$ |
```

**Gambar 23.** Hubungkan ke instance EC2 AWS

Jika autentikasi berhasil, pengguna akan masuk ke terminal instance EC2 dan dapat menjalankan perintah serta mengelola instance tersebut langsung dari terminal lokal.

- c. Instalasi Apache Kafka pada sebuah instance EC2 di layanan AWS.

```
.cd C:\PJBL
.ssh -i "cuaca-real-time.pem"
ec2-user@ec2-3-107-41-163.ap-southeast-2.compute.amazonaws.com
.wget
https://downloads.apache.org/kafka/3.7.0/kafka_2
```

```

.13-3.7.0.tgz

.ls

.tar -xvf kafka_2.13-3.7.0.tgz

.sudo yum install java-1.8.0-openjdk

.sudo yum update

.sudo yum install java-1.8.0-amazon-corretto

.cd kafka_2.13-3.7.0/

```

**Tabel 4.** Instalasi Apache kafka

Baris kode tersebut bertujuan untuk melakukan persiapan dan instalasi Apache Kafka pada sebuah instance EC2 di layanan AWS. Langkah-langkahnya dimulai dengan mengubah direktori kerja ke folder "PJBL" pada komputer lokal, kemudian terhubung ke instance EC2 menggunakan key pair PEM untuk autentikasi. Setelah itu, dilakukan pengunduhan file arsip Apache Kafka versi 3.7.0 dan dilakukan ekstraksi file arsip tersebut. Setelah berhasil diekstrak, dilakukan instalasi Java OpenJDK 1.8 pada instance EC2, diikuti dengan pembaruan semua paket yang diinstal di sistem. Selanjutnya, dilakukan instalasi Amazon Corretto 1.8 sebagai distribusi Java yang didukung oleh Amazon. Terakhir, direktori kerja diubah ke folder instalasi Kafka yang baru saja diekstrak untuk melanjutkan proses konfigurasi dan menjalankan server Kafka.

d. Menjalankan Zookeeper dan Kafka Server

- Menjalankan Zookeeper

Mengaktifkan ZooKeeper yang disertakan dengan Kafka untuk kemudahan konfigurasi awal yang akan dijalankan di terminal cmd dengan script sebagai berikut.

```

.bin/zookeeper-server-start.sh
config/zookeeper.properties

```

**Tabel 5.** Menjalankan zookeeper

- Menjalankan Kafka server

Setelah ZooKeeper berjalan, selanjutnya menjalankan Kafka server menggunakan script di bawah yang dijalankan di terminal cmd.

```
.cd C:\PJBL
.ssh -i "cuaca-real-time.pem"
ec2-user@ec2-3-107-41-163.ap-southeast-2.compute.amazonaws.com
.export KAKFA_HEAP_OPTS="-Xmx256M -Xms128M"
.cd kafka_2.13-3.7.0/
.bin/kafka-server-start.sh
config/server.properties
```

**Tabel 6.** Menjalankan server kafka

Kode tersebut digunakan untuk mengatur dan menjalankan server Kafka pada instance EC2. Langkah-langkah ini dimulai dengan mengubah direktori kerja ke folder proyek di komputer lokal. Kemudian, menggunakan perintah SSH untuk menghubungkan ke instance EC2 di AWS menggunakan kunci SSH untuk autentikasi, memastikan koneksi aman dengan menggunakan key pair yang telah diunduh sebelumnya. Setelah terhubung ke instance EC2, menetapkan ukuran heap memory Kafka, dengan batas maksimal 256 MB dan ukuran awal 128 MB. Selanjutnya, mengubah direktori kerja ke folder instalasi Kafka, memastikan bahwa semua perintah berikutnya dijalankan di lokasi yang benar. Terakhir, server Kafka dijalankan, memulai server Kafka sesuai dengan pengaturan yang tercantum dalam file konfigurasi, termasuk pengaturan seperti port dan parameter lainnya.

#### e. Membuat dan Memulai Producer dan Consumer dalam Kafka

- Membuat topic

bin/kafka-topics.sh	-create	-topic
cuaca-real-time		-bootstrap-server

```
3.107.41.163:9092      -replication-factor      1  
-partitions 1
```

**Tabel 7.** Buat topik

Membuat topik baru dalam sistem Kafka yang siap digunakan untuk memproduksi dan mengkonsumsi pesan terkait cuaca real-time.

- Start Producer Kafka dalam linux melalui CMD

```
bin/kafka-console-producer.sh      --topic  
cuaca-real-time                  --bootstrap-server  
3.107.41.163:9092
```

**Tabel 8.** Start producer Kafka di cmd

Dengan menggunakan perintah ini, dapat mengirimkan pesan ke topik **cuaca-real-time** yang terdapat dalam server Kafka. Setelah menjalankan perintah ini, konsol produser akan aktif dan dapat memasukkan pesan-pesan secara langsung untuk dipublikasikan ke topik yang ditentukan.

- Start Consumer Kafka dalam linux melalui CMD

```
bin/kafka-console-consumer.sh      --topic  
cuaca-real-time                  --bootstrap-server  
3.107.41.163:9092
```

**Tabel 9.** Start konsumen Kafka di cmd

Memulai sebuah konsol konsumen dalam Kafka untuk mengonsumsi pesan dari topik **cuaca-real-time** yang terdapat dalam server Kafka. Setelah menjalankan perintah ini, konsol konsumen akan aktif dan dapat digunakan untuk membaca pesan-pesan yang diproduksi ke topik tersebut.

The image shows two terminal windows side-by-side. The left window is titled 'C:\PJBL>' and the right window is titled 'C:\PJBL>'. Both windows show command-line interfaces for interacting with a Kafka cluster.

```

C:\PJBL> ssh -i "cuaca-real-time.pem" ec2-user@ec2-3-107-41-163.ap-southeast-2.compute.amazonaws.com
Last login: Thu Jun  6 16:47:26 2024 from 139.195.182.182
[ec2-user@ip-172-31-41-250 ~]$ cd kafka_2.13-3.7.0/
[ec2-user@ip-172-31-41-250 kafka_2.13-3.7.0]$ bin/kafka-topics.sh --create --topic cuaca-real-time --bootstrap-server 3.107.41.1:9092 --replication-factor 1 --partitions 1
Created topic cuaca-real-time.
[ec2-user@ip-172-31-41-250 kafka_2.13-3.7.0]$ bin/kafka-console-producer.sh --topic cuaca-real-time --bootstrap-server 3.107.41.1:9092
>Bismillah
>Alhamdulillah
>PJBL Big Data Kelompok 1
>

C:\PJBL>ssh -i "cuaca-real-time.pem" ec2-user@ec2-3-107-41-163.ap-southeast-2.compute.amazonaws.com
Last login: Thu Jun  6 16:53:38 2024 from 139.195.182.182
[ec2-user@ip-172-31-41-250 ~]$ cd kafka_2.13-3.7.0/
[ec2-user@ip-172-31-41-250 kafka_2.13-3.7.0]$ bin/kafka-console-consumer.sh --topic cuaca-real-time --bootstrap-server 3.107.41.1:9092
Bismillah
Alhamdulillah
PJBL Big Data Kelompok 1

```

**Gambar 24.** Producer dan Consumer dalam Kafka

Dalam gambar tersebut, Kafka telah dijalankan melalui terminal, produsen (*producer*) mengirimkan pesan dan konsumen (*consumer*) untuk menerima output dari produsen. Dengan Kafka aktif, dapat menggunakan terminal untuk mengirimkan pesan melalui produsen dan menerima output dari produsen tersebut melalui konsumen.

#### f. Membuat Producer dan Consumer menggunakan Python

Setelah membuat producer dan consumer di terminal cmd, langkah berikutnya adalah mengimplementasikan producer menggunakan Python. Producer akan membaca data cuaca dari file CSV dan mengirimkannya ke topik Kafka '**'cuaca-real-time'**'. Data ini kemudian akan diunggah ke Amazon S3 untuk analisis lebih lanjut.

- Producer

```

from kafka import KafkaConsumer, KafkaProducer
from time import sleep
from json import dumps, loads
import json
from s3fs import S3FileSystem
import pandas as pd

def produce_weather_data():
    producer =
KafkaProducer(bootstrap_servers='3.107.68.101:9092')
    csv_file = 'cuaca_realtime_sby_mlg.csv'
    df = pd.read_csv(csv_file)

```

```
for index, row in df.iterrows():
    data = row.to_dict()
    producer.send('data-cuaca-real-time',
value=str(data).encode('utf-8'))
    producer.flush()

producer.close()

produce_weather_data()
```

**Tabel 10.** Membuat producer dengan python

Setelah kode tersebut dijalankan, data cuaca dikirimkan ke consumer yang telah dibuat di cmd sebelumnya dan diterima dalam bentuk JSON. Proses ini dimulai ketika producer membaca data dari file CSV dan mengirimkannya ke topik Kafka yang ditentukan. Consumer di cmd mendengarkan topik tersebut dan menerima data dalam format JSON, yang terlihat pada gambar dibawah ini.

**Gambar 25.** Data cuaca JSON

- Consumer

```
from kafka import KafkaConsumer, KafkaProducer
from time import sleep
from json import dumps, loads
import json
from s3fs import S3FileSystem
import pandas as pd
import s3fs

s3 = s3fs.S3FileSystem(anon=False,
```

```

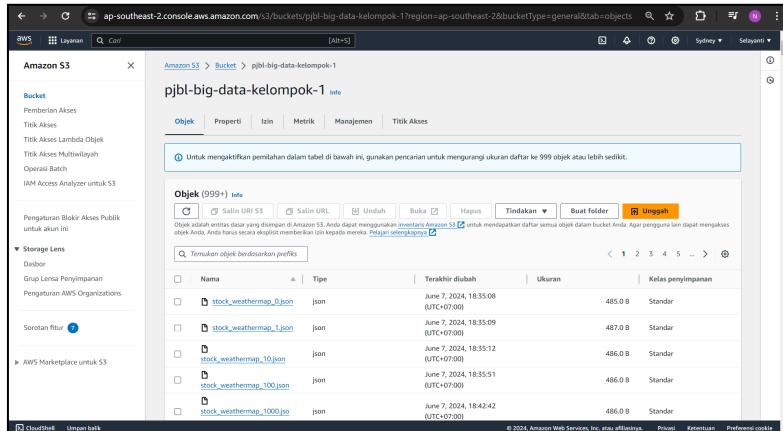
key='AKIA3FLD5UE7EDE663SG',
secret='y9xEVmFSObJOaB+u/ZPrMB2EAGm1NS5ns5HVKC
Da')
consumer =
KafkaConsumer("data-cuaca-real-time",
    bootstrap_servers=["3.107.68.101:9092"],
    value_deserializer=lambda x:
json.loads(x.decode("utf-8")),
)

for count, i in enumerate(consumer):
    with
s3.open("s3://pjbl-big-data-kelompok-1/stock_w
eathermap_{}.json".format(count), 'w') as
file:
    json.dumps(i.value, file)

```

**Tabel 11.** Membuat consumer dengan python

Kode consumer tersebut dijalankan untuk mengambil data dari topik Kafka dan menyimpannya ke Amazon S3. Data yang diterima oleh consumer akan disimpan dalam format JSON di bucket S3 telah ditentukan. berikut tampilan data yang berhasil terkirim di bucket S3.

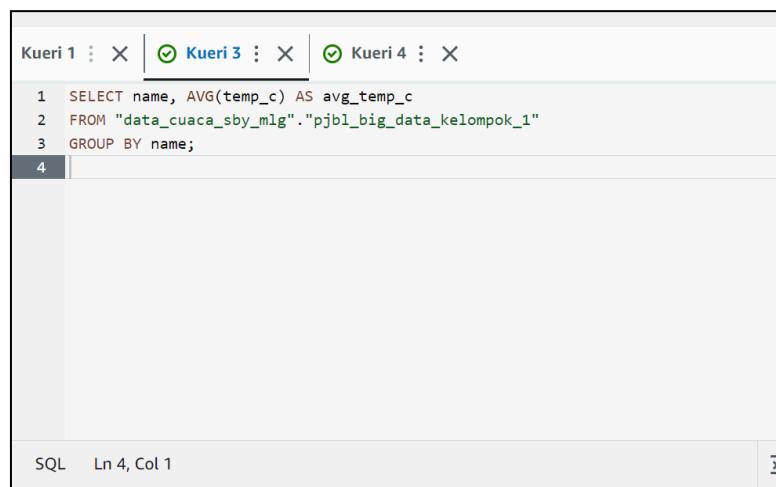


**Gambar 26.** Hasil data yang terkirim di bucket S3

#### g. Analisis data dengan query di AWS

Setelah berhasil mengirim data ke AWS, selanjutnya dapat menganalisisnya menggunakan query dengan memanfaatkan AWS Athena. Berikut beberapa analisis yang dilakukan terhadap data tersebut.

- Rata-rata Suhu (Celcius) di Setiap Kota



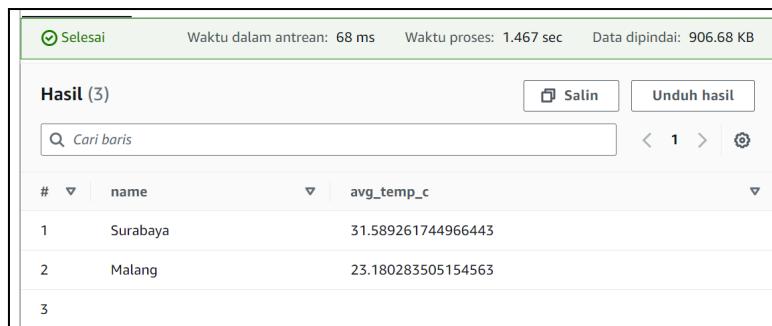
```

Kueri 1 : X | Kueri 3 : X | Kueri 4 : X
1 SELECT name, AVG(temp_c) AS avg_temp_c
2 FROM "data_cuaca_sby_mlg"."pjbl_big_data_kelompok_1"
3 GROUP BY name;
4
SQL  Ln 4, Col 1

```

**Gambar 27.** Code hitung rata-rata suhu tiap kota

Dari output di bawah ini, dapat dilihat bahwa rata-rata suhu di Surabaya adalah 31°C, sedangkan di Malang adalah 23°C. Dari perbandingan ini, dapat disimpulkan bahwa Surabaya umumnya memiliki suhu yang lebih tinggi daripada Malang.



Selesai      Waktu dalam antrean: 68 ms      Waktu proses: 1.467 sec      Data dipindai: 906.68 KB

**Hasil (3)**

Cari baris

#	name	avg_temp_c
1	Surabaya	31.589261744966443
2	Malang	23.180283505154563
3		

**Gambar 28.** Hasil rata - rata suhu tiap kota

- Kelembaban Rata-rata di Setiap Kota



```

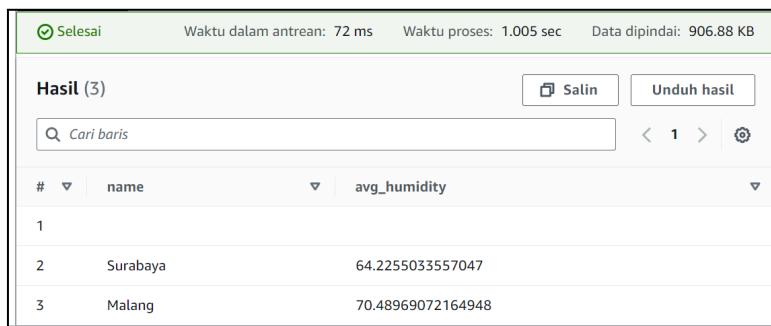
Kueri 1 : X | Kueri 3 : X
1  SELECT name, AVG(humidity) AS avg_humidity
2  FROM "data_cuaca_sby_m1g"."pjbl_big_data_kelompok_1"
3  GROUP BY name;
4
5

```

SQL Ln 4, Col 1

**Gambar 29.** Code hitung rata-rata kelembaban tiap kota

Dari hasil output di bawah ini, dapat dilihat bahwa Surabaya memiliki rata-rata kelembaban udara sebesar 64, sementara Malang memiliki rata-rata kelembaban udara sebesar 70. Dari perbandingan ini, dapat disimpulkan bahwa kota dengan kelembaban rata-rata yang lebih tinggi mungkin terasa lebih lembab dan kurang nyaman dibandingkan dengan kota yang memiliki kelembaban rata-rata lebih rendah.



Selesai Waktu dalam antrean: 72 ms Waktu proses: 1.005 sec Data dipindai: 906.88 KB

Hasil (3)

Cari baris

#	name	avg_humidity
1		
2	Surabaya	64.2255033557047
3	Malang	70.48969072164948

**Gambar 30.** Hasil rata-rata kelembaban tiap kota

- Suhu Tertinggi dan Terendah di Setiap Kota

```

Kueri 1 : X | Kueri 3 : X
1   SELECT name, MAX(temp_c) AS max_temp_c, MIN(temp_c) AS min_temp_c
2   FROM "data_cuaca_sby_mig"."pjbl_big_data_kelompok_1"
3   GROUP BY name;
4
5
6

```

SQL Ln 4, Col 1

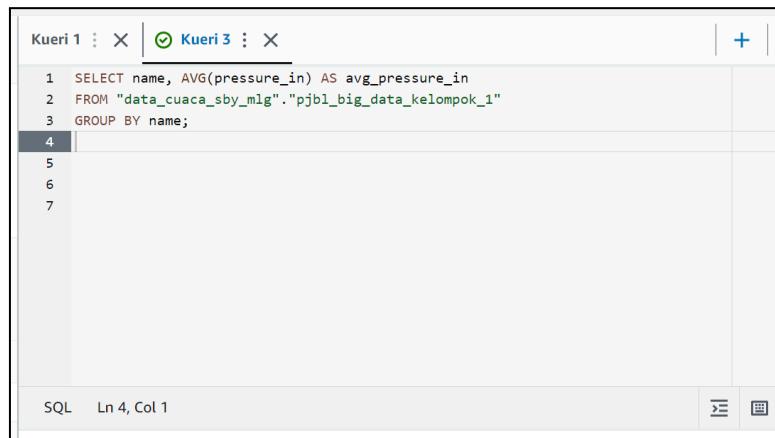
**Gambar 31.** Code hitung suhu tertinggi dan terendah tiap kota

Dari hasil output di bawah ini, terlihat bahwa suhu maksimum di Surabaya adalah 32°C dengan suhu minimum sebesar 30°C. Sedangkan di Malang, suhu maksimum adalah 25°C dan suhu minimum adalah 20°C. Dari perbandingan ini, dapat disimpulkan bahwa Surabaya memiliki rentang suhu yang lebih sempit dibandingkan dengan Malang, yang mengindikasikan variasi suhu harian atau musiman yang lebih besar di Malang.

#	name	max_temp_c	min_temp_c
1	Surabaya	32.0	30.0
2	Malang	24.9	19.7
3			

**Gambar 32.** Hasil suhu terendah dan tertinggi tiap kota

- Rata-rata Tekanan Udara di Setiap Kota



```

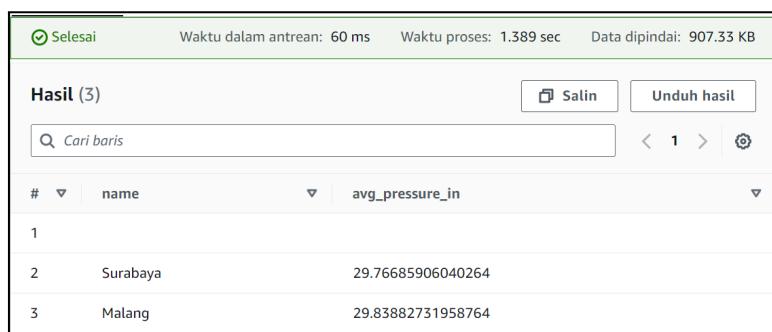
Kueri 1 : X | Kueri 3 : X +
1 SELECT name, AVG(pressure_in) AS avg_pressure_in
2 FROM "data_cuaca_sby_mlg"."pjbl_big_data_kelompok_1"
3 GROUP BY name;
4
5
6
7

```

SQL Ln 4, Col 1

**Gambar 33.** Code hitung rata-rata tekanan udara tiap kota

Analisis tekanan udara rata-rata memberikan informasi tentang kondisi atmosfer di setiap kota. Dari hasil output di bawah ini, terlihat bahwa rata-rata tekanan udara di Surabaya adalah 29.77, sedangkan di Malang adalah 29.84. Dari perbandingan ini, dapat disimpulkan bahwa Malang memiliki rata-rata tekanan udara yang sedikit lebih tinggi dibandingkan dengan Surabaya. Hal ini menunjukkan kondisi atmosfer yang berbeda di kedua kota tersebut, yang dapat berkaitan dengan ketinggian kota atau kondisi cuaca umum seperti tekanan udara tinggi atau rendah.



Selesai      Waktu dalam antrian: 60 ms      Waktu proses: 1.389 sec      Data dipindai: 907.33 KB

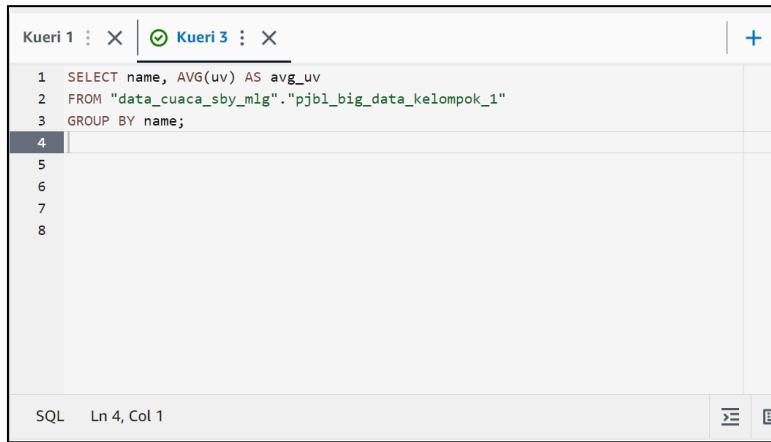
**Hasil (3)**

Cari baris

#	name	avg_pressure_in
1		
2	Surabaya	29.76685906040264
3	Malang	29.83882731958764

**Gambar 34.** Hasil rata-rata tekanan udara tiap kota

- Rata-rata Indeks UV di Setiap Kota



```

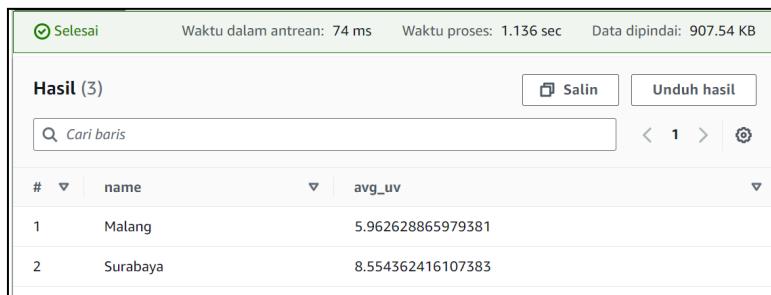
Kueri 1 : X | Kueri 3 : X
+-----+
1   SELECT name, AVG(uv) AS avg_uv
2   FROM "data_cuaca_sby_mlg"."pjbl_big_data_kelompok_1"
3   GROUP BY name;
4
5
6
7
8

```

SQL Ln 4, Col 1

**Gambar 35.** Code hitung rata-rata UV tiap kota

Analisis rata-rata indeks UV memberikan informasi tentang tingkat radiasi ultraviolet di setiap kota. Dari hasil output di bawah ini, terlihat bahwa rata-rata indeks UV di Surabaya adalah 8.55, sedangkan di Malang adalah 5.96. Dari perbandingan ini, dapat disimpulkan bahwa kota dengan rata-rata indeks UV yang tinggi mungkin memiliki risiko lebih besar terhadap dampak kesehatan yang terkait dengan paparan UV, seperti sunburn atau kanker kulit.



Selesai      Waktu dalam antrian: 74 ms      Waktu proses: 1.136 sec      Data dipindai: 907.54 KB

Hasil (3)

Cari baris

#	name	avg_uv
1	Malang	5.962628865979381
2	Surabaya	8.554362416107383

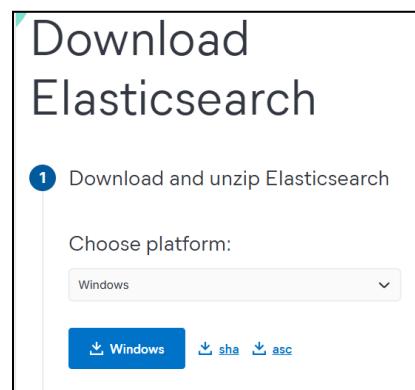
**Gambar 36.** Hasil rata-rata UV tiap kota

## 4.2. Implementasi Kibana

### 4.2.1. Proses Instalasi

#### a. Install dan Mengaktifkan Elasticsearch

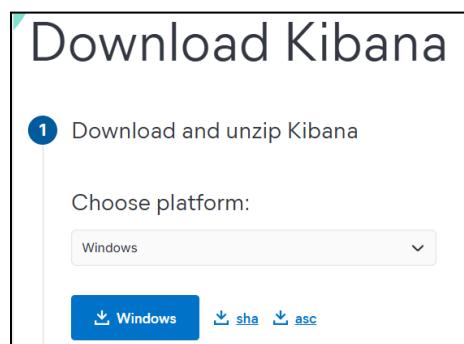
Sebelum menginstall Kibana, perlu menginstal elasticsearch terlebih dahulu melalui tautan berikut <https://www.elastic.co/downloads/elasticsearch> menggunakan versi lama yaitu **versi 8.7.0**.



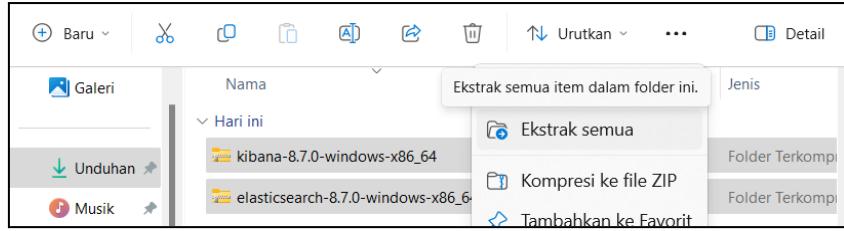
Gambar 37. Menginstal elasticsearch

#### b. Install dan Mengaktifkan Kibana

Setelah menginstall elasticsearch tidak lupa install kibana melalui tautan berikut <https://www.elastic.co/downloads/kibana>. Menggunakan versi lama yaitu **versi 8.7.0** karena jika menggunakan versi baru terkendala banyak error maka peneliti menggunakan versi lamas. Kemudian kedua file tersebut dilakukan ekstraksi file karena file yang didownload sebelumnya merupakan file zip.



Gambar 38. Menginstal kibana



**Gambar 39.** Mengekstrak file elasticsearch dan kibana

Setelah proses unduhan selesai, kedua file tersebut perlu diekstrak karena dalam keadaan awal keduanya berupa file berbentuk zip. Langkah ini dilakukan untuk mengakses isi dari setiap file, yang terdiri dari berbagai dokumen dan folder yang terkompresi di dalamnya. Proses ekstraksi memungkinkan pengguna untuk mengekstrak semua konten yang ada di dalam file zip sehingga dapat diakses dan digunakan.

#### 4.2.2. Menjalankan dan Memeriksa Aplikasi

Langkah pertama yang dilakukan untuk menjalankan aplikasi Elasticsearch dan Kibana adalah membuka terminal atau command prompt pada komputer. Di dalam direktori instalasi Elasticsearch, jalankan perintah `elasticsearch.bat` untuk memulai Elasticsearch. Setelah itu, di direktori yang sama atau direktori instalasi Kibana, jalankan perintah `kibana.bat` untuk memulai Kibana. Pastikan untuk menunggu hingga kedua proses tersebut selesai, sehingga dapat menggunakan kedua aplikasi tersebut untuk melakukan analisis dan visualisasi data sesuai kebutuhan.

```
C:\Windows\System32\cmd.exe + - Microsoft Windows [Version 10.0.22621.3672]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp\Downloads\elasticsearch-8.7.0>cd bin
C:\Users\hp\Downloads\elasticsearch-8.7.0>elasticsearch.bat
[2024-06-06T21:05:13,893][INFO ][o.e.n.Node] [DESKTOP-L2Q1IE5] version[8.7.0], pid[22280], build[zip/0952
0b59b6c1057340b5570186466ea715e30e/2023-03-27T16:31:09.816451435Z], OS[Windows 11/10.0/amd64], JVM[Oracle Corporation/
OpenJDK 64-Bit Server VM/19.0.2/19.0.2+4]
[2024-06-06T21:05:13,898][INFO ][o.e.n.Node] [DESKTOP-L2Q1IE5] JVM home [C:\Users\hp\Downloads\elasti
rch-8.7.0\jdk], using bundled JDK [true]
[2024-06-06T21:05:13,899][INFO ][o.e.n.Node] [DESKTOP-L2Q1IE5] JVM arguments [-Des.networkaddress.cache.t
tl=60, -Des.networkaddress.cache.negative.ttl=10, -Djava.security.manager=allow, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.
.awtHeadless=true, -Dfile.encoding=UTF-8, -Djava.nosys=true, -XX:-omitStackTraceInFastThrow, -Dio.netty.noUnsafe=true, -Dio.
.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dlog4j.shutdownHookEnabled=false, -Dlog4j2.
.disable.jmx=true, -Dlog4j.formatMsgNoLookups=true, -Djava.locale.providers=SPI,COMPAT, --add-opens=java.base/java.io=A
LL-UNNAMED, -XX:+UseG1GC, -Djava.io.tmpdir=C:\Users\hp\AppData\Local\Temp\elasticsearch, -XX:+HeapDumpOnOutOfMemoryError,
-XX:+ExitOnOutOfMemoryError, -XX:HeapDumpPath=data, -XX:ErrorFile=logs/hadoop_pid%p.log, -Xlog:gc*,g+age+trace,safepr
int:file=logs/gc.log:utctime,level,pid,tags:fileCount=32,fileSize=64m, -Xms3929m, -Xmx3929m, -XX:MaxDirectMemorySize=206
0451840, -XX:G1HeapRegionSize=4m, -XX:InitiatingHeapOccupancyPercent=30, -XX:G1ReservePercent=15, -Des.distribution.type
=zip, --module-path=C:\Users\hp\Downloads\elasticsearch-8.7.0\lib, --add-modules=jdk.net, -Djdk.module.main=org.elastics
earch_server]
[2024-06-06T21:05:15,732][INFO ][l.c.a.c.i.j.JacksonVersion] [DESKTOP-L2Q1IE5] Package versions: jackson-core=2.13.4, jac
son-dataformat-bind=2.13.4-2, jackson-dataformat-xml=2.13.4, jackson-datatype-jsr310=2.13.4, azure-core=1.34.0, Troubleshootin
g version conflicts: https://aka.ms/azsdk/java/dependency/troubleshoot
[2024-06-06T21:05:17,647][INFO ][o.e.p.PluginsService] [DESKTOP-L2Q1IE5] loaded module [aggregations]
[2024-06-06T21:05:17,662][INFO ][o.e.p.PluginsService] [DESKTOP-L2Q1IE5] loaded module [analysis-common]
[2024-06-06T21:05:17,670][INFO ][o.e.p.PluginsService] [DESKTOP-L2Q1IE5] loaded module [apm]
[2024-06-06T21:05:17,672][INFO ][o.e.p.PluginsService] [DESKTOP-L2Q1IE5] loaded module [blob-cache]
[2024-06-06T21:05:17,680][INFO ][o.e.p.PluginsService] [DESKTOP-L2Q1IE5] loaded module [constant-keyword]
```

**Gambar 40.** Menjalankan cmd elasticsearch

```

C:\Windows\System32\cmd.exe + Microsoft Windows [Version 10.0.22621.3672]
(C) Microsoft Corporation. All rights reserved.

C:\Users\hp\Downloads\kibana-8.7.0\bin>kibana.bat
[2024-06-06T22:20:53.494+07:00][INFO ][node] Kibana process configured with roles: [background_tasks, ui]
[2024-06-06T22:22:21.418+07:00][INFO ][plugins-service] Plugin "cloudChat" is disabled.
[2024-06-06T22:22:21.432+07:00][INFO ][plugins-service] Plugin "cloudExperiments" is disabled.
[2024-06-06T22:22:21.434+07:00][INFO ][plugins-service] Plugin "cloudFullStory" is disabled.
[2024-06-06T22:22:21.436+07:00][INFO ][plugins-service] Plugin "cloudGainsight" is disabled.
[2024-06-06T22:22:21.443+07:00][INFO ][plugins-service] Plugin "profiling" is disabled.
[2024-06-06T22:22:21.503+07:00][INFO ][http server Preboot] http server running at http://localhost:5601
[2024-06-06T22:22:21.539+07:00][INFO ][plugins-system preboot] Setting up [1] plugins: [interactiveSetup]
[2024-06-06T22:22:21.543+07:00][INFO ][preboot] "interactiveSetup" plugin is holding setup: Validating Elasticsearch connection configuration...
[2024-06-06T22:22:21.569+07:00][INFO ][root] Holding setup until preboot stage is completed.
[2024-06-06T22:22:21.659+07:00][WARN ][config deprecation] The default mechanism for Reporting privileges will work differently in future versions, which will affect the behavior of this cluster. Set "xpack.reporting.roles.enabled" to "false" to adopt the future behavior before upgrading.
[2024-06-06T22:22:21.839+07:00][INFO ][plugins-system standard] Setting up [132] plugins: [translations, monitoringCollection, licensing, globalSearch, globalSearchProviders, features, mapsEms, licenseApiGuard, customBranding, usageCollection, taskManager, cloud, guidedOnboarding, telemetryCollectionManager, telemetryCollectionXpack, kibanaUsageCollection, share, screenshotMode, banners, newsfeed, ftrApis, fieldFormats, expressions, screenshots, dataViews, charts, esUiShared, customIntegrations, home, searchProfiler, painlessLab, grokdebugger, management, cloudDataMigration, advancedSettings, spaces, security, snapshotRestore, lists, encryptedSavedObjects, telemetry, licenseManagement, files, eventLog, actions, notifications, console, contentManagement, bfetch, data, watcher, fileUpload, ingestPipelines, ecsDataQualityDashboard, alerting, unifiedSearch, unifiedFieldList, savedSearches, savedObjects, graph, savedObjectsManagement, eventAnnotation, embeddableReporting, uiActionsEnhanced, presentationUtil, expressionShape, expressionRevealImage, expressionRepeatImage, expressionMetric, expressionImage, controls, dataViewFieldEditor, triggersActionsUi, transform, stackConnectors, stackAlerts, ruleRegistry, visualizations, canvas, visTypeXY, visTypeVislib, visTypeVega, visTypeTimeSeries, visTypeTimeline, visTypeTagCloud, visTypeTable, visTypeMetric, visTypeHeatmap, vis

```

**Gambar 41.** Menjalankan cmd kibana

Untuk memeriksa koneksi ke Elasticsearch, dapat membuka peramban web dan masukkan 'localhost:9200' di bilah alamat. Ini akan menampilkan informasi terkait status Elasticsearch, termasuk versi yang sedang aktif dan detail lainnya.

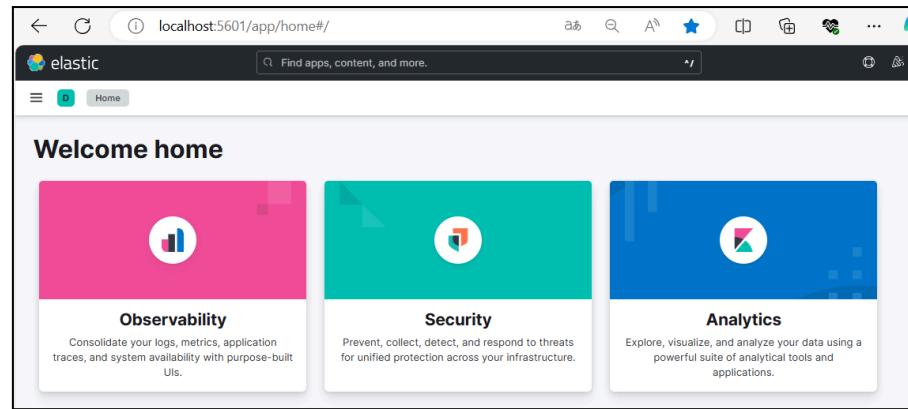
```

{
  "name": "DESKTOP-L2QI1E5",
  "cluster_name": "elasticsearch",
  "cluster_uuid": "PeeD7LTOR90Mf-muDBx4lg",
  "version": {
    "number": "8.7.0",
    "build_flavor": "default",
    "build_type": "zip",
    "build_hash": "09520b59b6bc1057340b55750186466ea715e30e",
    "build_date": "2023-03-27T16:31:09.816451435Z",
    "build_snapshot": false,
    "lucene_version": "9.5.0",
    "minimum_wire_compatibility_version": "7.17.0",
    "minimum_index_compatibility_version": "7.0.0"
  },
  "tagline": "You Know, for Search"
}

```

**Gambar 42.** Mengecek localhost elasticseach

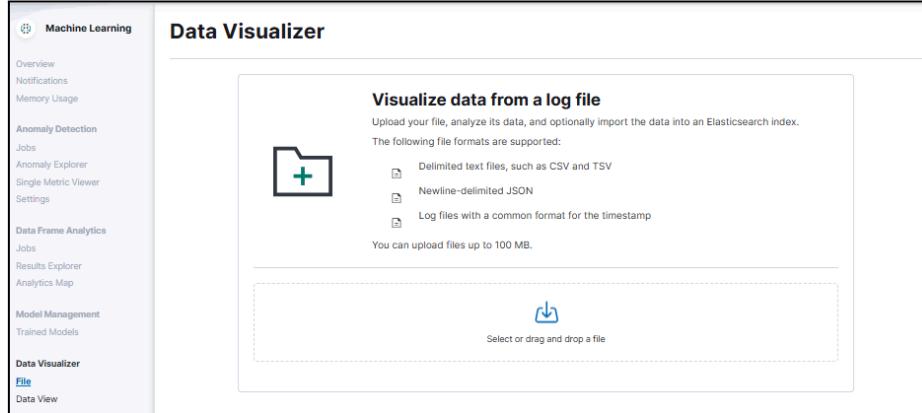
Selanjutnya, untuk memeriksa Kibana, cukup ketik localhost:5601 di bilah alamat peramban web. Kibana akan memuat dashboard dan antarmuka pengguna yang memungkinkan untuk melakukan visualisasi data dan analisis menggunakan Elasticsearch sebagai backend.



**Gambar 43.** Mengecek localhost kibana

Setelah berhasil di localhost kibana, maka proses selanjutnya yaitu pengolahan data yang meliputi statistik deskripsi hingga visualisasi data. Pastikan bahwa kedua aplikasi ini berjalan dengan lancar dan dapat diakses melalui peramban web sebelum melanjutkan ke langkah-langkah berikutnya dalam mengelola dan menggunakan data dengan Elasticsearch dan Kibana.

#### 4.2.3. Input Data ke dalam Kibana



**Gambar 44.** Import data

Data Visualizer	
<b>cuaca_realtime_sby_ml.csv</b>	
<b>File contents</b> First 997 lines	
<pre> 23 Malang;East Java;Indonesia;-8.0;114.14;Asia/Jakarta;21/05/2024 08:45:19.7;6:29.89;0.85;85;19.7;5;0;4;3;6.8 24 Surabaya;East Java;Indonesia;-7.25;112.75;Asia/Jakarta;21/05/2024 08:51:21/05/2024 08:45:30.0;6.0;6.0;1;Partly cloudy;1003;5.6;9.0;100;29.83;0.0;70;25;37.6;8;0;10;1;16.2 25 Malang;East Java;Indonesia;-8.0;114.14;Asia/Jakarta;21/05/2024 08:55:21/05/2024 08:45:19.7;6:29.89;0.85;85;19.7;5;0;4;3;6.8 26 Surabaya;East Java;Indonesia;-7.25;112.75;Asia/Jakarta;21/05/2024 08:55:21/05/2024 08:45:19.7;6:29.89;0.85;85;19.7;5;0;4;3;6.8 27 Malang;East Java;Indonesia;-8.0;114.14;Asia/Jakarta;21/05/2024 08:55:21/05/2024 08:45:19.7;6:29.89;0.85;85;19.7;5;0;4;3;6.8 28 Surabaya;East Java;Indonesia;-7.25;112.75;Asia/Jakarta;21/05/2024 08:55:21/05/2024 08:45:19.7;6:29.89;0.85;85;19.7;5;0;4;3;6.8 29 Malang;East Java;Indonesia;-8.0;114.14;Asia/Jakarta;21/05/2024 08:55:21/05/2024 08:45:30.0;6.0;6.0;1;Partly cloudy;1003;5.6;9.0;100;29.83;0.0;70;25;37.6;8;0;10;1;16.2 30 Surabaya;East Java;Indonesia;-7.25;112.75;Asia/Jakarta;21/05/2024 08:55:21/05/2024 08:45:30.0;6.0;6.0;1;Partly cloudy;1003;5.6;9.0;100;29.83;0.0;70;25;37.6;8;0;10;1;16.2 31 Malang;East Java;Indonesia;-8.0;114.14;Asia/Jakarta;21/05/2024 08:55:21/05/2024 08:45:30.0;6.0;6.0;1;Partly cloudy;1003;5.6;9.0;100;29.83;0.0;70;25;37.6;8;0;10;1;16.2 32 Surabaya;East Java;Indonesia;-7.25;112.75;Asia/Jakarta;21/05/2024 09:01:21/05/2024 09:00:30.0;6.0;6.0;1;Partly cloudy;1003;5.6;9.0;100;29.83;0.0;70;25;35.6;8;0;10;1;16.2 33 Malang;East Java;Indonesia;-8.0;114.14;Asia/Jakarta;21/05/2024 08:55:21/05/2024 08:45:19.7;6:29.89;0.85;85;19.7;5;0;4;3;6.8 34 Surabaya;East Java;Indonesia;-7.25;112.75;Asia/Jakarta;21/05/2024 08:55:21/05/2024 08:45:19.7;6:29.89;0.85;85;19.7;5;0;4;3;6.8 35 Malang;East Java;Indonesia;-8.0;114.14;Asia/Jakarta;21/05/2024 08:55:21/05/2024 08:45:19.7;6:29.89;0.85;85;19.7;5;0;4;3;6.8 36 Surabaya;East Java;Indonesia;-7.25;112.75;Asia/Jakarta;21/05/2024 09:01:21/05/2024 09:00:30.0;6.0;6.0;1;Partly cloudy;1003;5.6;9.0;100;29.83;0.0;70;25;35.6;8;0;10;1;16.2 </pre>	
<b>Summary</b>	
Number of lines analyzed	997
Format	delimited
Delimiter	:
Has header row	true
<a href="#">Override settings</a> <a href="#">Analysis explanation</a>	

Gambar 45. Input file

Proses penginputan data di dalam Kibana, untuk mengimpor data manual ke Kibana langkah pertamanya adalah membuka aplikasi Kibana melalui peramban web menggunakan localhost:5601 atau URL lain yang sesuai. Setelah berhasil masuk, navigasikan ke bagian "Management" di sidebar sebelah kiri. Di halaman manajemen Kibana, pilih "Saved Objects" untuk mengakses fungsi impor. Pilih opsi "Import" dan unggah file data yang ingin digunakan, seperti file JSON atau CSV. Kibana akan mengolah data yang diunggah, dan setelah proses selesai, pastikan untuk memeriksa bahwa data telah berhasil diimporkan dan siap digunakan untuk melakukan visualisasi dan analisis lebih lanjut di Kibana.

#### 4.2.4. Statistika Deskriptif pada Data

Kemudian melakukan statistik deskriptif dengan memilih opsi "Visualize" dan buat visualisasi baru untuk menghasilkan statistika deskriptif. Pilih jenis visualisasi yang sesuai seperti tabel data atau histogram, tergantung pada jenis data yang dimiliki. Konfigurasikan visualisasi dengan memilih indeks pattern yang tepat dan kolom-kolom data yang ingin dianalisis, seperti rata-rata, median, atau nilai minimum dan maksimum. Setelah proses selesai, hasil statistika deskriptif tersebut untuk mendapatkan wawasan yang lebih mendalam dari data yang akan dikelola, memungkinkan pengambilan keputusan yang lebih baik berdasarkan informasi yang diberikan. Kemudian memunculkan hasil statistika deskriptif dari variabel-variabelnya.

File stats						
All fields 24 of 24 total		Number fields 17 of 17 total		Field name 24 ▾ Field type 2 ▾ ⓘ		
Type	Name ↗	Documents (%) ⓘ	Distinct values	Distributions ⓘ		
>	cloud	996 (100%)	12	min 25	median 38	max 88
>	condition_code	996 (100%)	4	min 1003	median 1033	max 1240
>	condition_text	996 (100%)	4	4 categories		
>	country	996 (100%)	1	1 category		
>	feelslike_c	996 (100%)	23	min 19.7	median 30.5	max 38.5
>	gust_kph	996 (100%)	10	min 4	median 13.5	max 27.4
>	gust_mph	996 (100%)	10	min 2.5	median 8.4	max 17
>	humidity	996 (100%)	12	min 59	median 65	max 85
>	is_day	996 (100%)	1	min 1	median 1	max 1
>	last_updated	996 (100%)	35	top 10 or 25 categories		
>	lat	996 (100%)	2	min -8	median -7.62	max -7.25
>	localtime	996 (100%)	207	top 10 or 25 categories		
>	lon	996 (100%)	2	min 112.75	median 113.45	max 114.14
>	name	996 (100%)	2	2 categories		
>	precip_in	996 (100%)	2	min 0	median 0	max 0.01
>	pressure_in	996 (100%)	10	min 29.71	median 29.8	max 29.9
>	region	996 (100%)	1	1 category		
>	temp_c	996 (100%)	13	min 19.7	median 27.45	max 32
>	temp_f	996 (100%)	14	min 67.4	median 81.45	max 89.6
>	tz_id	996 (100%)	1	1 category		
>	uv	996 (100%)	4	min 5	median 7	max 9
>	wind_degree	996 (100%)	17	min 60	median 129	max 187
>	wind_kph	996 (100%)	6	min 3.6	median 6.5	max 20.2
>	wind_mph	996 (100%)	6	min 2.2	median 4.05	max 12.5

Gambar 46. Statistik Data Setiap variabel

File stats						
All fields 24 of 24 total		Number fields 17 of 17 total		Field name 24 ▾ Field type 2 ▾ ⓘ		
Type	Name ↗	Documents (%) ⓘ	Distinct values	Distributions ⓘ		
>	cloud	996 (100%)	12	min 25	median 38	max 88
	DOCUMENTS STATS	SUMMARY	TOP VALUES			
count	996	min 25	498 (< 0.1%)			
percentage	100%	median 38	76	60 (< 0.1%)		
distinct values	12	max 88	81	60 (< 0.1%)		
			75	59 (< 0.1%)		
			82	59 (< 0.1%)		
			51	58 (< 0.1%)		
			88	58 (< 0.1%)		
			55	57 (< 0.1%)		
			66	53 (< 0.1%)		
			85	17 (< 0.1%)		
				Calculated from 996 records.		

Gambar 47. Statistik variabel V1

**Machine Learning**

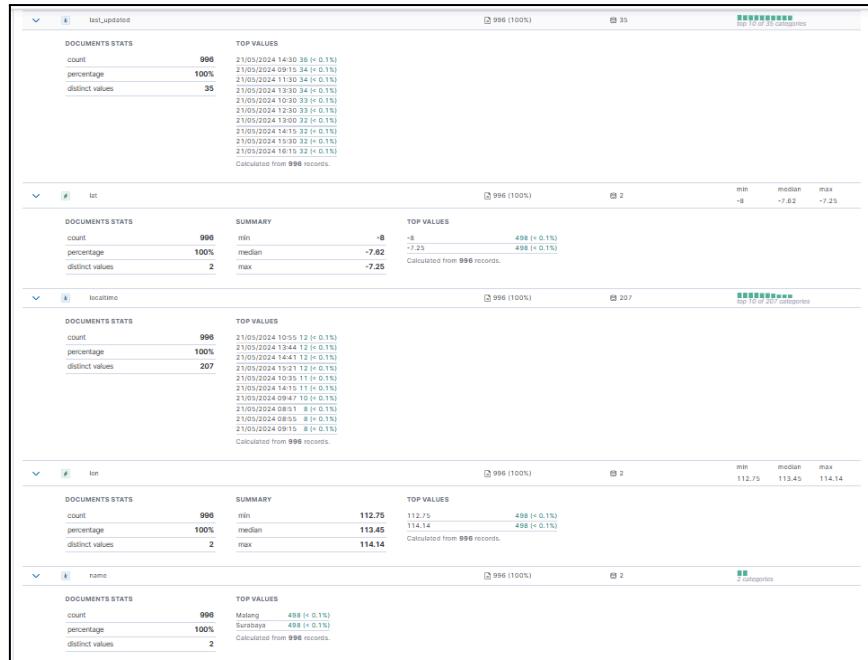
- Overview
- Notifications
- Memory Usage
- Anomaly Detection
  - Jobs
  - Anomaly Explorer
  - Single Metric Viewer
  - Settings
- Data Frame Analytics
  - Jobs
  - Results Explorer
  - Analytics Map
- Model Management
  - Trained Models
- Data Visualizer
  - File
  - Data View
- AllOps Labs
- Explain Log Rate Spikes
- Log Pattern Analysis
- Change Point Detection

**Gambar 48.** Statistik variabel V2 - V5

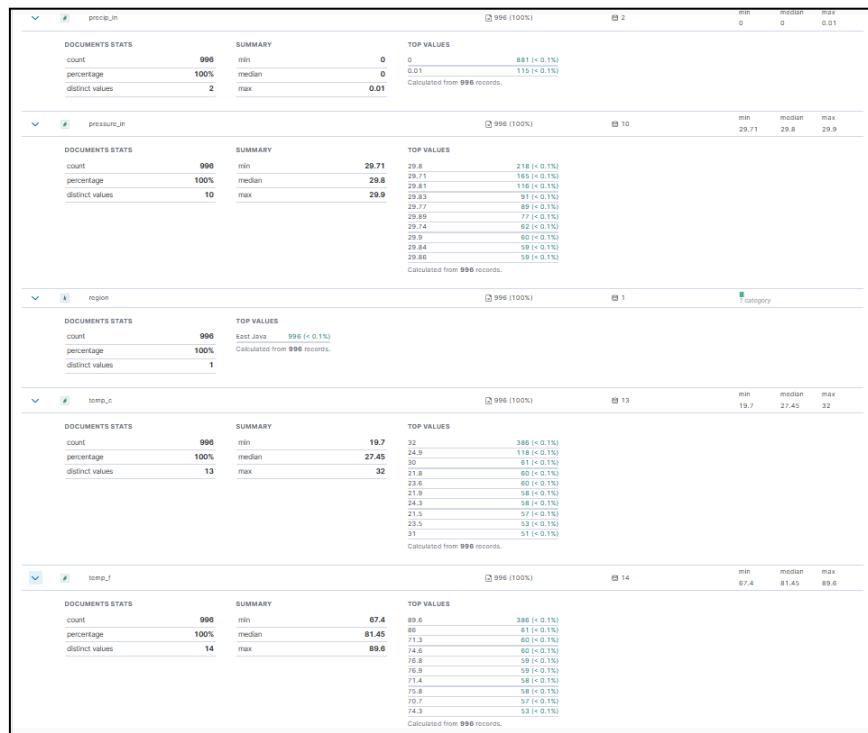
**Machine Learning**

- Overview
- Notifications
- Memory Usage
- Anomaly Detection
  - Jobs
  - Anomaly Explorer
  - Single Metric Viewer
  - Settings
- Data Frame Analytics
  - Jobs
  - Results Explorer
  - Analytics Map
- Model Management
  - Trained Models
- Data Visualizer
  - File
  - Data View
- AllOps Labs
- Explain Log Rate Spikes
- Log Pattern Analysis
- Change Point Detection

**Gambar 49.** Statistika Variabel V6-V9



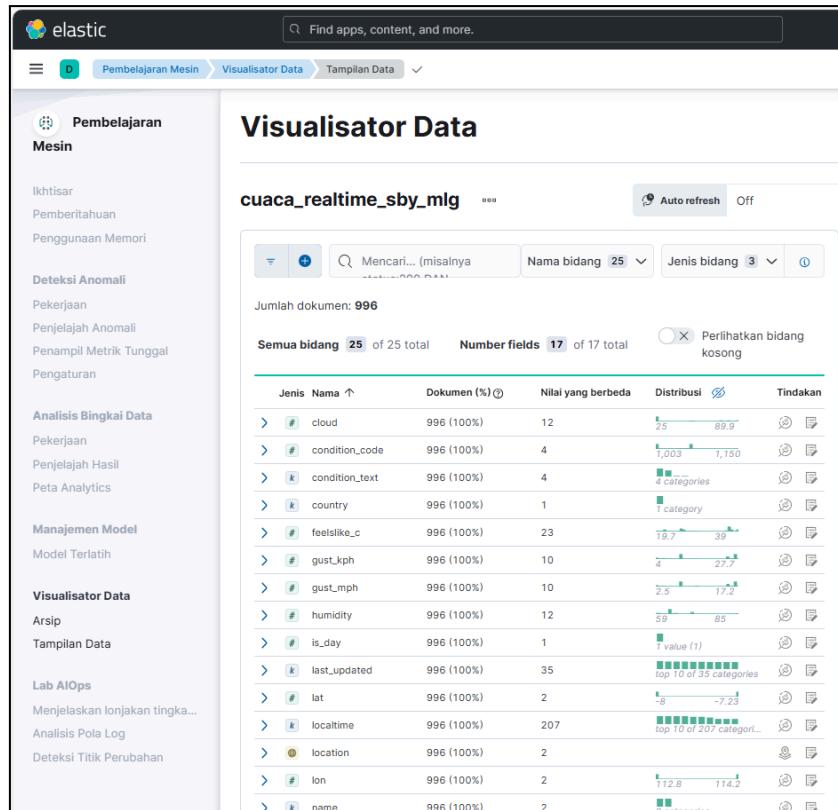
Gambar 50. Statistik variabel V10 - V14



Gambar 51. Statistik variabel V15 - V19

DOCUMENTS STATS		TOP VALUES	
count 996		Asia/Jakarta 996 (< 0.1%)	
percentage 100%		Calculated from 996 records.	
distinct values 1			
▼	uv	(996 (100%))	Category
DOCUMENTS STATS		SUMMARY	TOP VALUES
count	996	min	5
percentage	100%	median	7
distinct values	4	max	9
		(996 (100%))	min median max
			5 7 9
Calculated from 996 records.			
▼	wind_degree	(996 (100%))	Category
DOCUMENTS STATS		SUMMARY	TOP VALUES
count	996	min	80
percentage	100%	median	129
distinct values	17	max	187
		(996 (100%))	min median max
			80 129 187
Calculated from 996 records.			
▼	wind_kph	(996 (100%))	Category
DOCUMENTS STATS		SUMMARY	TOP VALUES
count	996	min	3.6
percentage	100%	median	6.5
distinct values	6	max	20.2
		(996 (100%))	min median max
			3.6 6.5 20.2
Calculated from 996 records.			
▼	wind_mph	(996 (100%))	Category
DOCUMENTS STATS		SUMMARY	TOP VALUES
count	996	min	2.2
percentage	100%	median	4.05
distinct values	6	max	12.5
		(996 (100%))	min median max
			2.2 4.05 12.5
Calculated from 996 records.			

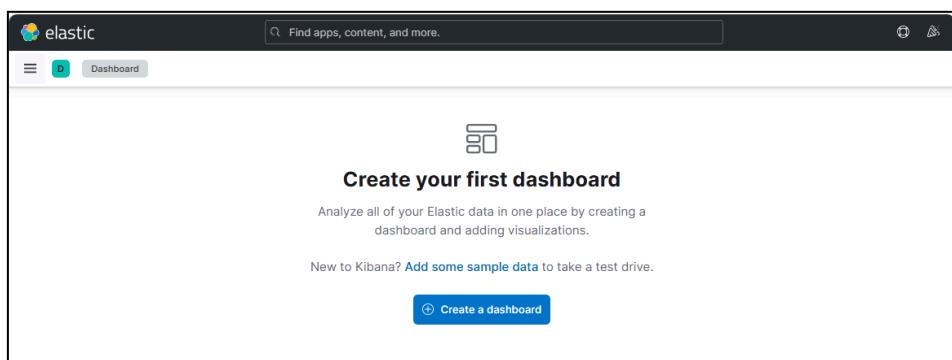
Gambar 52. Statistik variabel V20 - V25



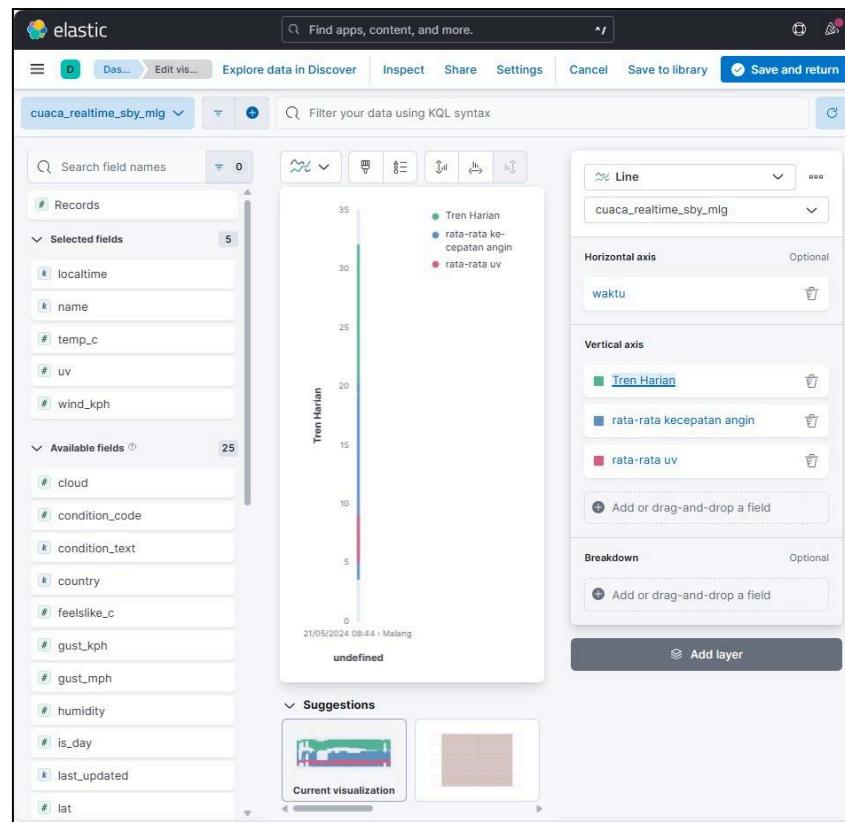
Gambar 53. Outputan data yang telah di import dan siap di visualisasikan

#### 4.2.5. Visualisasi Data menggunakan Kibana

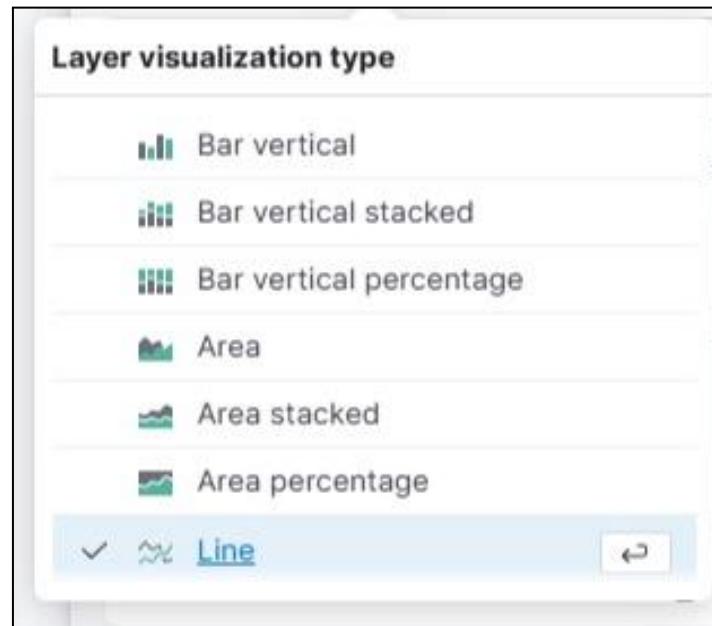
Setelah itu, navigasikan ke menu "Dashboard" di sidebar untuk memilih jenis visualisasi seperti bar chart, line chart, atau pie chart sesuai kebutuhan analisis data. Konfigurasikan visualisasi dengan memilih indeks pattern dan kolom-kolom data yang ingin visualisasikan, serta tambahkan metrik seperti rata-rata atau jumlah. Kibana akan menampilkan visualisasi secara interaktif, memungkinkan untuk menyesuaikan pengaturan visualisasi, menerapkan filter, dan menyimpan hasil untuk digunakan dalam dashboard atau laporan lebih lanjut. Langkah-langkah ini memudahkan dalam mengeksplorasi data dan mendapatkan pemahaman yang lebih dalam dari informasi yang terkandung dalam dataset.



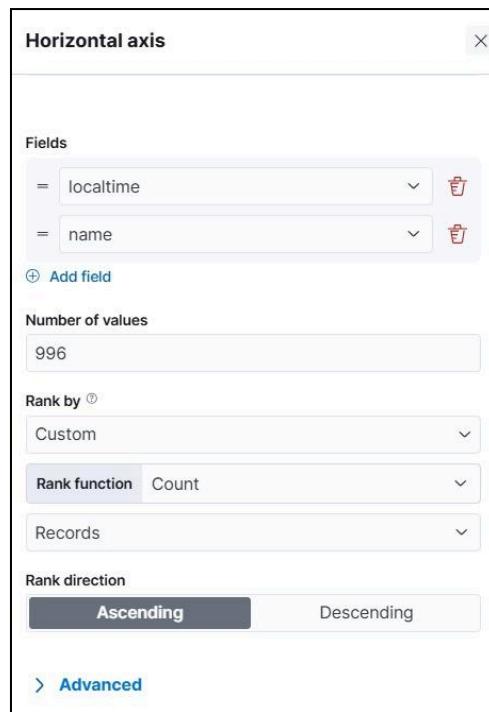
**Gambar 54.** Membuat dashboard



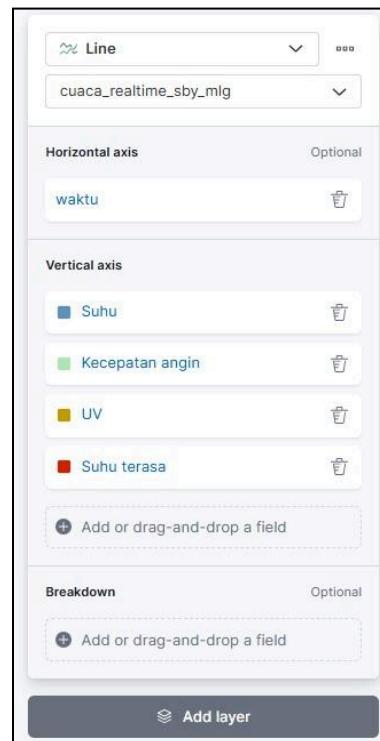
**Gambar 55.** Memasukkan variabel yang akan dilakukan visualisasi



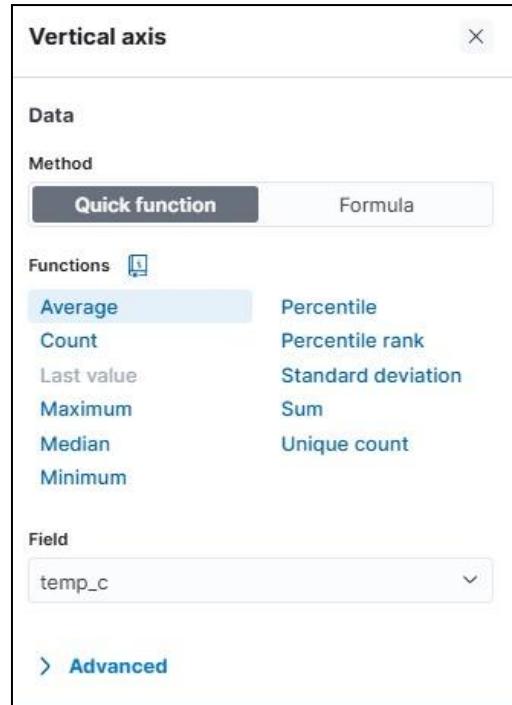
**Gambar 56.** Memilih jenis visualisasi



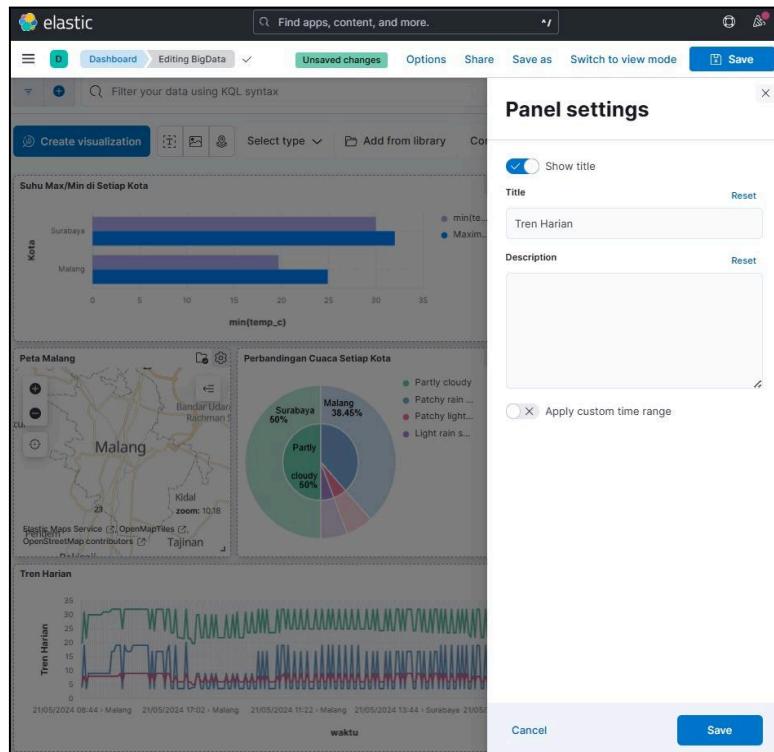
**Gambar 57.** Memilih filter dan function yang akan digunakan



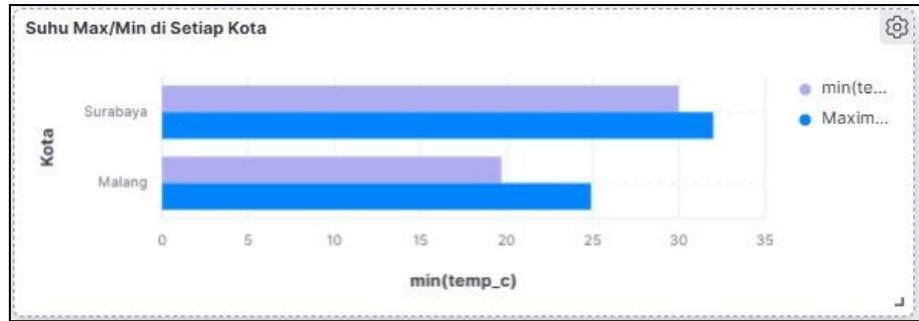
**Gambar 58.** Memilih beberapa variabel pada sumbu y



**Gambar 59.** Memilih function average untuk eksekusi visualisasi



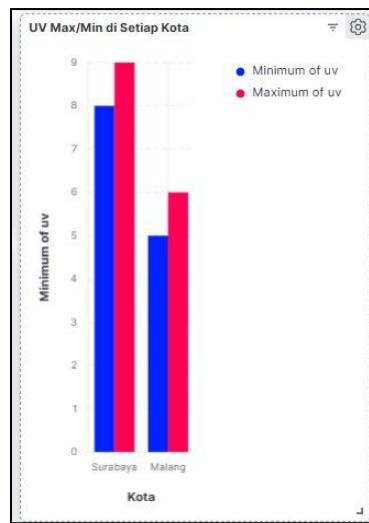
**Gambar 60.** Memberikan judul pada setiap hasil visualisasi



**Gambar 61.** Visualisasi max/min suhu tiap kota

Berdasarkan grafik perbandingan suhu antara Surabaya dan Malang, dapat diamati bahwa Malang memiliki rentang suhu yang lebih rendah dibandingkan Surabaya. Suhu minimum di Malang adalah 19.7°C, sedangkan suhu maksimumnya mencapai 24.9°C. Di sisi lain, Surabaya memiliki suhu yang lebih tinggi dengan suhu minimum mencapai 30°C dan suhu maksimum mencapai 32°C. Perbandingan ini menunjukkan bahwa Malang memiliki iklim yang lebih sejuk dibanding Surabaya, yang dapat berpengaruh pada kondisi lingkungan dan kehidupan sehari-hari di kedua kota tersebut.

Perbedaan suhu antara Malang dan Surabaya memiliki dampak yang pada aktivitas luar ruangan, konsumsi energi untuk pendinginan, dan kesehatan masyarakat. Malang, dengan suhu yang lebih rendah, memungkinkan kondisi termal yang lebih nyaman, mendorong aktivitas luar ruangan yang lebih aktif, dan mengurangi kebutuhan akan pendinginan udara. Di Surabaya, yang memiliki suhu lebih tinggi, mungkin diperlukan upaya lebih besar untuk mengatasi panas, seperti penggunaan AC atau perlindungan dari sinar matahari langsung. Solusi yang bisa dipertimbangkan meliputi penyesuaian aktivitas dengan kondisi cuaca, penggunaan teknologi efisien untuk mengatur suhu dalam ruangan, serta peningkatan kesadaran akan pentingnya adaptasi terhadap perubahan suhu.



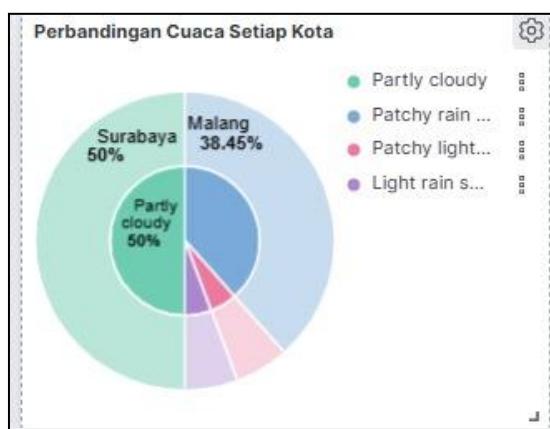
**Gambar 62.** Visualisasi max/min paparan UV tiap kota

Berdasarkan perbandingan UV Index antara Surabaya dan Malang, terlihat bahwa Malang memiliki tingkat UV Index yang lebih rendah dibandingkan Surabaya. Dalam grafik, tercatat bahwa Malang memiliki UV Index minimum sebesar 5 dan maksimum sebesar 6, sementara Surabaya memiliki UV Index minimum 8 dan maksimum sebesar 9. Perbedaan ini mengindikasikan bahwa Malang cenderung memberikan paparan sinar UV yang lebih rendah dibandingkan Surabaya. Perbedaan UV karena terdapat faktor-faktor geografis seperti Malang terletak di dataran tinggi, mungkin mendapatkan beberapa keuntungan dalam mengurangi intensitas sinar UV karena efek penurunan intensitas sinar matahari dengan ketinggian. Sebaliknya, Surabaya yang terletak di dataran rendah dan dekat dengan pantai mungkin mengalami paparan sinar UV yang lebih tinggi karena sinar matahari yang langsung dan intens di daerah tersebut.

Untuk mengurangi risiko dari paparan sinar UV yang tinggi di Surabaya, penting untuk mempertimbangkan langkah-langkah perlindungan diri yang tepat. Ini termasuk penggunaan tabir surya dengan SPF tinggi, penggunaan pakaian pelindung seperti topi dan kacamata matahari, serta menghindari kegiatan di luar ruangan pada jam-jam di mana sinar UV paling intens, yaitu antara pukul 10 pagi hingga 4 sore. Edukasi masyarakat juga penting dalam meningkatkan kesadaran akan pentingnya perlindungan dari sinar UV untuk mencegah kerusakan kulit jangka panjang dan risiko kanker kulit.



Gambar 63. Menampilkan peta

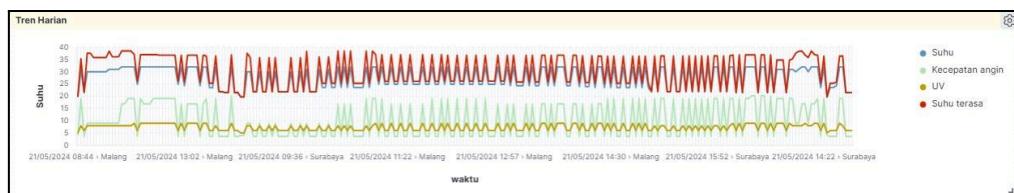


Gambar 64. Visualisasi perbandingan cuaca tiap kota

Berdasarkan diagram pie untuk perbandingan cuaca antara Surabaya dan Malang, terlihat bahwa Malang menunjukkan variasi cuaca yang lebih beragam dibandingkan Surabaya. Di Malang, dari total 498 data cuaca yang tercatat, kondisi "patchy rain nearby" mendominasi dengan persentase 38.35%, diikuti oleh "patchy light drizzle" sebesar 5.82% dan "light rain shower" sebesar 5.72%. Sementara itu, di Surabaya, seluruh 498 data cuaca menunjukkan cuaca "partly cloudy" secara konsisten. Perbedaan ini mungkin dipengaruhi oleh karakteristik geografis dan iklim kedua kota. Malang, yang terletak di dataran tinggi dengan kondisi iklim pegunungan, cenderung mengalami variasi cuaca yang lebih luas seperti hujan ringan dan kondisi berawan di sekitarnya. Di sisi lain, Surabaya yang terletak di dataran rendah dan dekat pantai, cenderung memiliki cuaca yang lebih stabil dengan dominasi cuaca berawan sebagian besar waktu.

Untuk menghadapi berbagai perubahan cuaca di Malang, diperlukan tindakan seperti monitoring cuaca secara teratur, merencanakan aktivitas sesuai

kondisi cuaca, serta menyiapkan perlengkapan seperti payung atau jas hujan. Meningkatkan kesadaran masyarakat akan pentingnya kesiapsiagaan dalam menghadapi perubahan cuaca juga merupakan faktor krusial untuk mengurangi dampak negatif terhadap kegiatan sehari-hari. Dengan demikian, langkah-langkah ini diharapkan dapat membantu penduduk Malang agar lebih siap dan adaptif menghadapi variasi cuaca yang terjadi secara berkala.



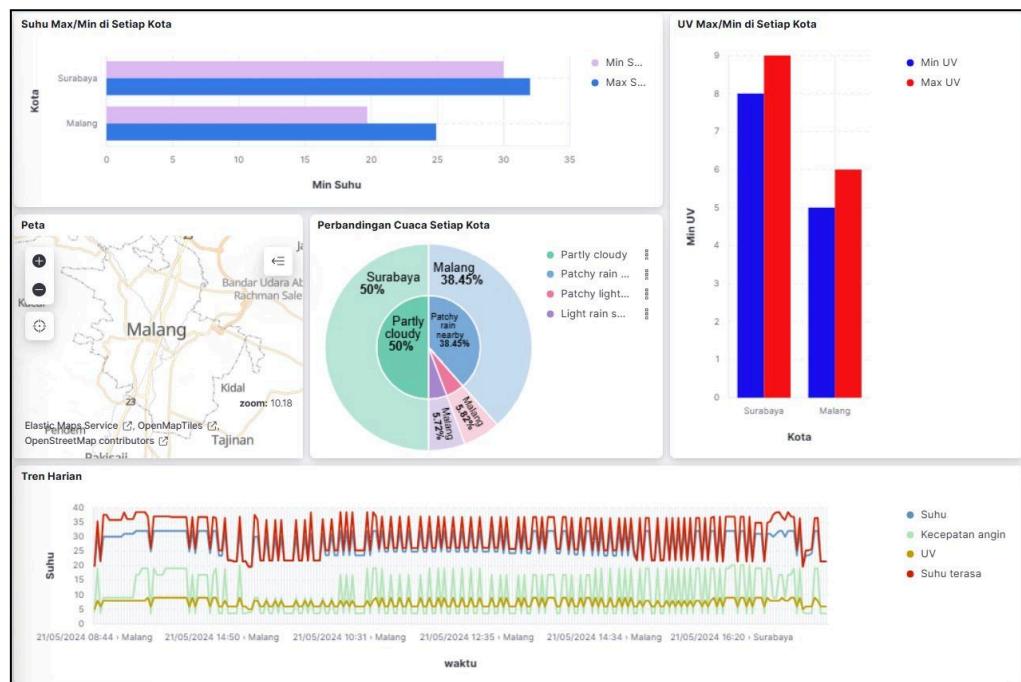
**Gambar 65.** Visualisasi tren harian

Berdasarkan gambar 65 dari tren suhu dapat dilakukan analisis dan perbandingan pada kedua waktu di daerah Malang dan Surabaya. Pada pukul 11.37 21/05/2024 di Surabaya, suhu mencapai 32°C dengan kecepatan angin 16.9 km/jam dan UV Index 9, tetapi suhu terasa mencapai 37°C, menunjukkan potensi risiko panas yang tinggi. Pada waktu yang sama di Malang, suhu 24.9°C dengan kecepatan angin 3.6 km/jam dan UV Index 6, dengan suhu terasa sekitar 26.1°C, menunjukkan kondisi yang lebih sejuk dibandingkan Surabaya.

Kemudian terlihat pada waktu berikutnya yaitu pukul 08.47 pada hari yang sama, Surabaya mencatat suhu 30°C dengan kecepatan angin 9 km/jam dan UV Index 8, namun suhu terasa mencapai 37.6°C, menunjukkan peningkatan risiko panas yang signifikan. Di Malang pada waktu yang sama, suhu 19.7°C dengan kecepatan angin 4 km/jam dan UV Index 5, dengan suhu terasa sama dengan suhu aktual, menunjukkan kondisi lebih sejuk.

Saat sore hari penulis melihat pada pukul 17.02, Surabaya mencatat suhu 31°C dengan kecepatan angin 19.1 km/jam dan UV Index 8, namun suhu terasa sekitar 35.3°C, menunjukkan pengaruh tinggi suhu udara dan kelembaban yang tinggi. Di Malang pada jam yang sama, suhu 21.6°C dengan kecepatan angin 3.6 km/jam dan UV Index 6, dengan suhu terasa sekitar 21.6°C, menunjukkan stabilitas suhu terasa meskipun suhu aktual lebih rendah. Perbedaan ini menunjukkan pentingnya memantau dan memahami kondisi cuaca lokal untuk mengelola risiko kesehatan dan aktivitas sehari-hari.

Untuk melindungi dari perubahan cuaca yang terus berubah kawasan industri yang terdampak oleh perubahan cuaca seperti sektor pertanian, konstruksi, dan pariwisata perlu mempertimbangkan penggunaan perlindungan cuaca yang tepat dan melakukan pemantauan kondisi cuaca secara teratur. Hal ini bertujuan untuk mengurangi risiko terhadap kesehatan dan meningkatkan produktivitas tenaga kerja dalam berbagai kondisi cuaca yang mungkin ekstrem. Di sisi lain, masyarakat perlu lebih menyadari pentingnya melindungi diri dari sinar UV yang tinggi dan risiko panas yang dapat berdampak negatif pada kesehatan. Disarankan untuk menggunakan perlengkapan pelindung seperti tabir surya, topi, dan menjaga asupan cairan yang cukup untuk mengurangi risiko terkena penyakit akibat cuaca panas. Dengan mempertimbangkan variasi kondisi cuaca di Surabaya dan Malang, langkah-langkah preventif ini diharapkan dapat membantu mengurangi dampak negatif dari perubahan cuaca yang ekstrim serta meningkatkan kesiapsiagaan dalam menghadapinya.



**Gambar 66.** Hasil Akhir Visualisasi

Setelah semua visualisasi yang diperlukan selesai, hasilnya akan disimpan dalam dashboard yang terdiri dari empat visualisasi berbeda dan satu peta kota. Perbandingan antara Surabaya dan Malang dalam hal suhu, UV Index, cuaca, dan tren kondisi cuaca menunjukkan perbedaan yang signifikan antara kedua

kota tersebut. Malang memiliki suhu yang lebih rendah dengan rentang antara 19.7°C hingga 24.9°C, sedangkan Surabaya memiliki suhu yang lebih tinggi, berkisar antara 30°C hingga 32°C. Perbedaan ini tidak hanya mempengaruhi pengaturan aktivitas luar ruangan, tetapi juga konsumsi energi untuk pendinginan dan kesehatan masyarakat secara keseluruhan. Selain itu, Malang juga menunjukkan UV Index yang lebih rendah, antara 5-6, dibandingkan Surabaya yang berkisar antara 8-9, sehingga berpotensi mengurangi risiko terhadap kerusakan kulit dan kanker kulit.

Selain itu, Malang memiliki variasi cuaca yang lebih beragam seperti hujan ringan dan hujan deras di sekitarnya, sementara Surabaya cenderung memiliki cuaca yang lebih stabil, sebagian besar cerah sebagian. Analisis tren suhu dan kondisi cuaca pada waktu tertentu menunjukkan bahwa Surabaya seringkali memiliki suhu dan suhu terasa yang lebih tinggi daripada Malang pada waktu yang sama, sedangkan Malang menunjukkan stabilitas suhu terasa yang lebih baik, mungkin disebabkan oleh kondisi geografisnya yang lebih sejuk. Dengan memahami perbedaan ini, penduduk kedua kota dapat mempersiapkan diri dengan lebih baik dalam menghadapi perubahan cuaca dan potensi risiko kesehatan yang terkait.

Penyimpanan dalam bentuk dashboard memungkinkan akses yang terpusat dan mempermudah pembagian informasi kepada tim atau pihak lain yang terlibat dalam pengambilan keputusan, selain juga memfasilitasi analisis data yang lebih efisien dan konsisten. Pentingnya mengelola perbedaan iklim lokal untuk meningkatkan kualitas hidup dan kesejahteraan masyarakat. Dengan memanfaatkan karakteristik iklim yang berbeda antara keduanya, seperti suhu dan tingkat UV Index yang bervariasi, dapat dilakukan pengaturan aktivitas dan perlindungan kesehatan yang lebih efisien. Peningkatan kesadaran masyarakat terhadap perlunya melindungi diri dari paparan sinar UV tinggi serta adaptasi terhadap perubahan cuaca yang ekstrem juga menjadi kunci. Langkah-langkah preventif yang diterapkan dengan tepat, seperti penggunaan peralatan pelindung dan penyesuaian aktivitas sesuai dengan kondisi cuaca, diharapkan dapat mengurangi dampak negatif dari perubahan cuaca ekstrem di kedua kota tersebut.

## **BAB V**

### **KESIMPULAN**

Proyek "Optimalisasi Pemantauan Data Cuaca *Real-Time* dari OpenWeatherMap Menggunakan Apache Kafka dan Kibana" telah berhasil mencapai tujuannya yaitu untuk meningkatkan efisiensi dalam mengumpulkan, memproses, dan memvisualisasikan data cuaca secara real-time. Dengan memanfaatkan teknologi Apache Kafka dan AWS, sistem ini dapat mengumpulkan data cuaca secara real-time dan menyimpannya di Amazon S3 untuk analisis lebih lanjut menggunakan AWS Athena. Hasil analisis ini kemudian diintegrasikan ke dalam Kibana, yang menyediakan alat visualisasi kuat untuk membangun dashboard yang informatif. Dashboard ini tidak hanya membantu dalam memahami tren cuaca, tetapi juga mendukung pengambilan keputusan yang lebih baik di berbagai sektor seperti pertanian, transportasi, dan manajemen bencana.

Hasil analisis AWS dan visualisasi di Kibana memperlihatkan perbedaan signifikan antara Surabaya dan Malang dalam kondisi lingkungan mereka. Surabaya memiliki suhu rata-rata yang lebih tinggi dan rata-rata indeks UV yang lebih tinggi dibandingkan dengan Malang, menunjukkan potensi risiko kesehatan yang lebih besar akibat sinar UV di kota ini. Di sisi lain, Malang menunjukkan variasi suhu harian yang lebih besar dan kelembaban udara rata-rata yang lebih tinggi, yang menggambarkan kondisi atmosfer yang berbeda. Visualisasi Kibana menyoroti kebutuhan untuk adaptasi yang lebih baik terhadap perubahan cuaca, khususnya di Malang yang memiliki cuaca lebih bervariasi dengan kondisi seperti hujan ringan sporadis, yang memerlukan kesiapan lebih dalam menghadapi dampaknya terhadap kehidupan sehari-hari dan industri.

Secara keseluruhan, proyek ini tidak hanya mengoptimalkan pemantauan dan analisis data cuaca, tetapi juga memberikan wawasan penting bagi mitigasi risiko dan adaptasi terhadap perubahan iklim di dua kota tersebut. Dengan memanfaatkan teknologi untuk memperoleh dan memproses data secara efisien, harapannya proyek ini dapat menjadi model bagi implementasi sistem pemantauan data real-time dalam berbagai konteks lainnya, mendukung perbaikan kualitas hidup dan kesiapan menghadapi tantangan lingkungan di masa depan.

## DAFTAR PUSTAKA

- Anirwan, & Haris A. (2023). Upaya Pemerintah Kota Makassar dalam Mewujudkan Ketahanan Kota Pascabencana Banjir. In Journal of Governance and Local Politics (Vol. 5, Issue 2). <https://doi.org/https://doi.org/10.47650/jglp.v5i2.999>
- Dewia, C., & Chen, R.-C. (2019). Integrating real-time weather forecasts data using OpenWeatherMap and Twitter. International Journal of Information Technology and Business (IJITEB), 1(2).
- Farissa, R. A., Mayasari, R., & Umaidah, Y. (2021). Perbandingan Algoritma K-Means dan K-Medoids Untuk Pengelompokkan Data Obat dengan Silhouette Coefficient. In Journal of Applied Informatics and Computing (JAIC) (Vol. 5, Issue 2). <http://jurnal.polibatam.ac.id/index.php/JAIC>
- Fathin, M. R. F., Basuki, A., & Bhawiyuga, A. (2022). Penerapan Elastic Stack sebagai Platform Visualisasi dan Analisis Trafik pada Jaringan Riset dan Edukasi. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 6(6), 2664-2672.
- Harimurti, Y., & Udariansyah, D. (2023). Implementasi Service EC2 & S3 Amazon Web Service Pada Niche Blog Menggunakan Metode SDLC. KLIK: Kajian Ilmiah Informatika dan Komputer, 4(2), 675-685. <https://www.djournals.com/klik/article/view/1192/735>
- Indrarto, S. A., & Basuki, A. (2022). Penerapan Platform Visualisasi dan Analisis Trafik Jaringan menggunakan Elastic Stack. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 6(9), 4562-4570.
- Khan, F. R. (2021). Apache Kafka with real-time data streaming. Retrieved from <https://www.researchgate.net/publication/348575301>
- Mahardika, S. S., Kurniawan, W., & Bakhtiar, F. A. (2019). Implementasi Sistem Real Time untuk Pendekripsi Dini Banjir berbasis ESP8266 dan Weather API. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 3(8), 8238-8247.

Nabawi, F. (2022). Implementasi Sistem Distribusi Pesan dan Proses Data Secara Real Time dengan Apache Kafka. *Jurnal Teknologi Informatika dan Komputer MH. Thamrin*, 8(1).

Permadi, R. B., Tibyani, & Arwani, I. (2020). Implementasi Teknologi AWS Cloud Dalam Pengembangan Aplikasi Ujian Online Berbasis Website Menggunakan Framework Codeigniter (Studi Kasus: SMAN 1 Jombang dan MAN 9 Jombang). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 4(7), 1933-1942.

Permatahati, I., Perdana, M. N., Apriadi, N., Amanda, T. P., & Maharani, Z. (2023). Analisis Dataset TOP 1000 IMDb Movies Menggunakan Hadoop. 2(2).  
<https://jurnal.netplg.com/>

Pradana Aji, R., & Budi Cahyono, A. (2020). Pengembangan Dasbor Sistem Pencatatan Log Server Menggunakan Elasticsearch-Fluentd-Kibana (EFK) Stack.  
<https://journal.uii.ac.id/AUTOMATA/article/view/15406>

Putra, M. J. R., & Saptono, H. (2022). Penerapan Log Analyzer Log untuk Mengetahui Lalu Lintas Jaringan Berbasis Elasticsearch, Logstash, dan Kibana. *Jurnal Informatika Terpadu*, 8(1), 21-25.

Rakhmawati, N. A., Suryawan, S. H., Furqon, M. A., & Hermansyah, D. (2019). Indonesia's Public Application Programming Interface (API). *Jurnal Penelitian Pos dan Informatika*, 85-96. <https://jurnal-ppi.kominfo.go.id/index.php/jppi/article/view/090201>

Suhery, N., Mahendra Jaya, M., Tri Khikmawati, L., Sarasati, W., Estmirar Tanjov, Y., Fitria Larasati, R., Arkam Azis, M., Purwanto, A., Purnama Sari, I., Mainnah, M., & Muda Satyawan, N. (2023). Keterkaitan Musim Hujan Dan Musim Angin Dengan Musim Penangkapan Ikan Lemuru Yang Berbasis Di PPN Pengambangan. *Jurnal Teknologi Dan Manajemen Perikanan Laut*, 14(Vol. 14 No. 1 (2023)), 77–90.  
<https://doi.org/https://doi.org/10.29244/jmf.v14i1.44383>

Suprayogi, R., Rizal, S., & Zuhriyadi, I. (2023). IMPLEMENTASI CLOUD COMPUTING AMAZON WEB SERVICES (AWS) PADA WEB PEMBELAJARAN WAHASAN NUSANTARA. *Jurnal Ilmiah Betrik*, 14(03 DESEMBER), 468-478.  
<https://ejournal.pppmitpa.or.id/index.php/betrik/article/view/138/104>

Tambunan, H. A., & Saputra, D. (2022). Rancang Bangun Aplikasi Prediksi Cuaca Berbasis Android. *Jurnal Bisantara Informatika (JBI)*, 6(2).

Tarigan, C., Engel, V. J. L., & Angela, D. (2018). Sistem Pengawasan Kinerja Jaringan Server Web Apache dengan Log Management System ELK (Elasticsearch, Logstash, Kibana). *Jurnal Telematika*, 7. Edisi Industrial Engineering Seminar and Call for Paper (IESC) 2018.

Warsito, A. B., Ananda A., Triyanjaya, D. (2017). Penerapan Data JSON Untuk Mendukung Pengembangan Aplikasi Pada Perguruan Tinggi Dengan Teknik Restfull Dan Web Service. in Journal Technomedia Journal (TMJ), 26-36  
267827-penerapan-data-json-untuk-mendukung-peng-b1a9128a.pdf (neliti.com)

## **LAMPIRAN**

- Link Youtube: <https://youtu.be/kTTpc6v8WcQ>
- Dashboard : <http://localhost:5601/app/r/s/5ivna>