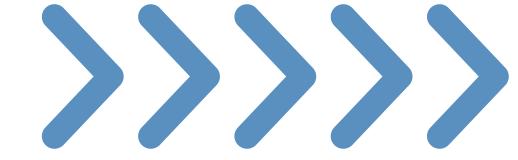


# **BIG DATA ARCHITECTURE FOR REALTIME COMMENT FILTERING AND ANALYSIS IN TWITCH LIVE STREAM**



# ANGGOTA KELOMPOK >>>>

01

adi putra hioe wiatma  
-2702244441

02

Rionaldho Limnardy -  
2702266764

03

Afif Belva Hadiat - 2702372920

Valentius Ryan Lukito -  
2702279211

04

Valerian Rivaldi -  
2702255571

05

Nicholas Hardianto  
Johari - 2702208594

06

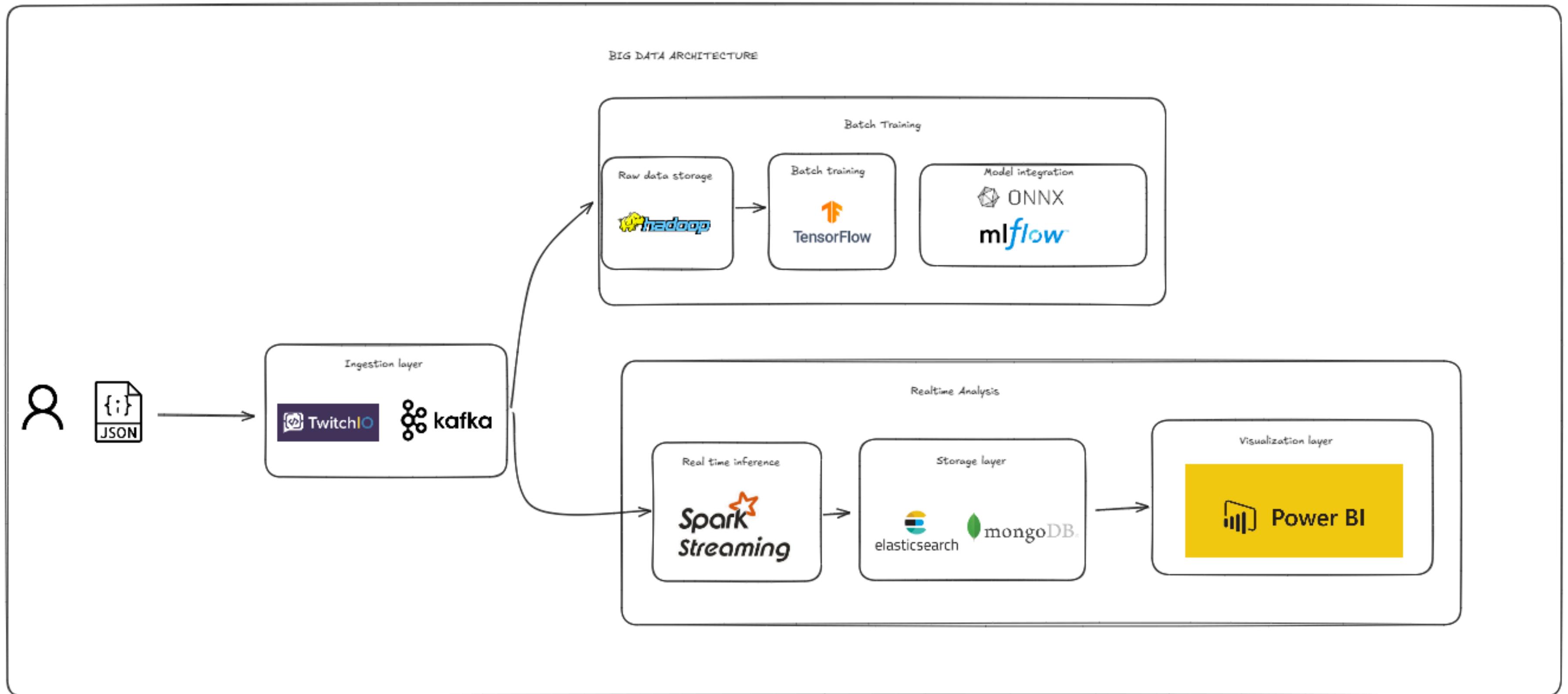
# LATAR BELAKANG

Platform live streaming seperti twitch kini banyak digunakan untuk hiburan, edukasi, dan interaksi pengguna secara real-time. Namun, kolom komentar sering disalahgunakan untuk menyebarkan ujaran kebencian, kata kasar, komentar rasis, dan konten tidak senonoh. Hal ini dapat mengganggu kenyamanan pengguna dan mencoreng reputasi platform.

Proyek ini bertujuan membangun arsitektur big data untuk mendeteksi dan mengklasifikasi komentar negatif dalam skala besar. Dengan memanfaatkan teknologi seperti Apache Kafka, Spark, dan machine learning, sistem ini dapat memproses komentar secara batch, menyimpan hasilnya, dan memisahkan komentar positif dan negatif berdasarkan kategori seperti rasis, judol, atau vulgar. Arsitektur ini mendukung pengambilan keputusan berbasis data untuk menjaga komunitas tetap sehat dan aman.



# Big Data Architecture



# **INGESTION**



# **LAYER**



# APPA ITU INGESTION LAYER ?

Tujuan Utama:

Gerbang pertama untuk semua data masuk.

Fungsi dalam Proyek Ini:

- Mengumpulkan data komentar pengguna secara real-time.
- Menyediakan data mentah untuk diproses lebih lanjut.

Karakteristik Ideal: Cepat, Andal (Reliable), Skalabel.

Teknologi Pilihan: Apache Kafka, Twitch IO

# DATA YANG DI INGEST

Sumber Data: Komentar dari pengguna (misalnya dari aplikasi web/mobile) di dalam twitch.

Format Data:

- JSON Fleksibel dan mudah dipahami mesin & manusia.
- Contoh Field: comment\_id, user\_id, text, timestamp.

Alur Masuk: TwitchIO (producer) akan mengambil komen secara realtime dari twitch dan akan dikirim ke kafka

# PERAN KAFKA DI INGESTION LAYER

Fokus Utama: Menerima & Menyimpan Data Mentah.

- Bukan untuk analisis isi komentar (deteksi negatif dilakukan di layer berikutnya).

Tugas Kafka di Sini:

1. Menerima pesan (komentar JSON) dari Producer.
2. Menyimpan pesan ke dalam Topic Kafka.
3. Memastikan data tersedia untuk diambil oleh Consumer (Spark Streaming, Hadoop).



# ARSITEKTUR & KOMPONEN INTI KAFKA

(Diagram Sederhana: Producer -> Kafka (Topic/Broker) -> Consumer)

Producer: Aplikasi/sistem yang mengirim data ke Kafka.

Broker: Server Kafka yang menyimpan data.

- Cluster: Kumpulan Broker untuk skalabilitas & fault tolerance.

Topic: Kategori atau nama stream data (e.g., komentar\_mentah).

- Partition: Topic dibagi menjadi partisi untuk paralelisasi.

Consumer: Aplikasi/sistem yang membaca data dari Kafka.





# KEBUTUHAN NODE KAFKA: BROKER

Jumlah Broker Node:

- Development: 1 Node (cukup untuk belajar, tanpa fault tolerance).
- Produksi: Minimal 3 Node.

Mengapa 3 Node untuk Produksi?

- Load Balancing: Beban kerja didistribusikan, tidak membebani satu node.
- Fault Tolerance/High Availability: Jika 1 node mati, data tetap aman & sistem terus berjalan (dengan replikasi data).

Fungsi Utama Broker Node: Menerima & menyimpan stream data dari producer.

- Menyimpan data dalam partisi topik Kafka.
- Melayani permintaan data dari consumer.
- Mengelola replikasi data.

# KEBUTUHAN NODE KAFKA: ZOOKEEPER (ATAU KRAFT)

Zookeeper (Pendekatan Tradisional):

- Jumlah Node: Minimal: 1 Node (Kafka butuh ini untuk berjalan).
- Produksi: 3 Node (untuk konsensus & fault tolerance Zookeeper).

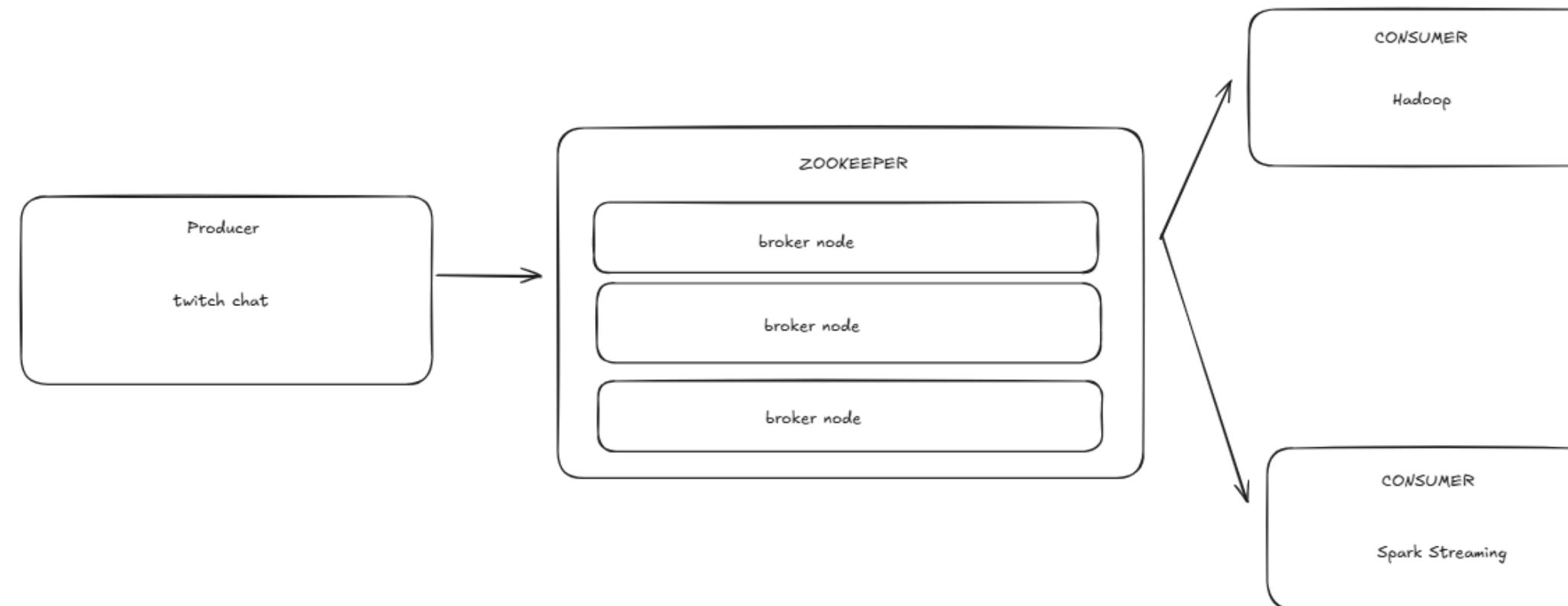
Fungsi Utama:

- Koordinasi Cluster Kafka: Pemilihan leader partisi, status broker.
- Penyimpanan Metadata: Info topics, konfigurasi.

KRaft (Kafka Raft metadata mode):

- Alternatif Modern: Kafka tanpa Zookeeper.
- Keuntungan: Menyederhanakan arsitektur & operasional.
- Rekomendasi: Pertimbangkan untuk proyek baru.

# DIAGRAM SEDERHANA KAFKA



# **RAW DATA**

**STORAGE**

# APA ITU RAW DATA STORAGE ?

Raw data storage adalah layer penyimpanan awal untuk data mentah yang belum diproses sama sekali. di sinilah semua data dikumpulkan dari berbagai sumber lalu disimpan dalam bentuk aslinya alias raw

Tujuan layer ini :

- Menyimpan data komentar mentah (raw)
- Menyediakan storage yang scalable dan kuat untuk data dari kafka
- Data yang disimpan dalam storage ini akan digunakan untuk melatih model machine learning

Tools yang digunakan :

- Apache HDFS (Hadoop distributed file system)

# KENAPA PAKE HDFS ?

## Scalable

Bisa simpan data dalam jumlah besar (terdistribusi ke banyak node)

## Fault-tolerant

Data direplikasi otomatis ke beberapa node

## Terintegrasi

Support berbagai tools big data kayak Spark, Hive dan Kafka.

# BENTUK DATA YANG DISIMPAN

## Jenis data

- Komentar mentahan dari user
- Metadata tambahan

## Format file

- JSON – fleksibel dan mudah dibaca
- parquet – efisien untuk proses analisis

# ARSITEKTUR HDFS

## NameNode

- Mengelola metadata file system
- Tidak menyimpan data langsung

## DataNode

- Menyimpan blok data komentar mentah dari kafka
- Handle permintaan read/write dari client dan replikasi data antar node

# BATCH TRAINING PROCESS



# MELATIH MODEL DENGAN TENSORFLOW

Setelah data disimpan di dalam hadoop data akan digunakan untuk melatih model machine learning di Tensorflow. Proses training akan dilakukan secara batch dengan data per 3 hari.

model tensorflow ini akan dilatih untuk mendekripsi jenis komentar menjadi komentar baik atau buruk.

Kami memilih menggunakan Tensorflow dikarenakan Tensorflow dapat melatih model yang kompleks yang cocok digunakan untuk mengkategorikan komentar

# MENYIMPAN MODEL

Setelah model dilatih model akan disimpan dan digunakan disini kami menggunakan 2 tools yaitu ONNX dan MLFlow. Setelah model disimpan model akan dapat digunakan dalam spark streaming

Fungsi ONNX: ONNX memungkinkan kita untuk menyimpan model dan mengkonversi model ke bentuk ONNX. Ini menyelesaikan masalah framework lock in jadi kita tidak terlalu terpaku untuk menggunakan satu framework. Di dalam kasus ini model yang kita latih di dalam Tensorflow akan kita konversi ke bentuk ONNX untuk mempermudah deployment dan digunakan di realtime inference di dalam spark streaming.

# ML FLOW

Kami juga menggunakan ML Flow. ML Flow kami gunakan untuk tracking model yang sudah kami latih. Dengan ML Flow kami dapat menyimpan result training, metrik, dan juga mempermudah deployment model untuk digunakan.

# **PROCESSING LAYER**



P

Daya Saing

# APA ITU PROCESSING LAYER ?

Processing Layer adalah bagian dari arsitektur data atau machine learning yang bertanggung jawab untuk mengolah data mentah menjadi data yang siap digunakan baik untuk analisis, pelatihan model machine learning, dan lain-lain.

Tujuan layer ini :

- Mengolah data mentah menjadi data bersih (Siap digunakan)
- Transformasi dan Enrichment data
- Inference (Prediksi Model)
- Efisiensi dan Skalabilitas

Tools yang digunakan :

- Apache Spark streaming



# KENAPA MENGGUNAKAN SPARK?

1. Kecepatan eksekusi yang tinggi
  2. Dapat menangani data dalam skala besar
  3. Terdapat library machine learning dan streaming
  4. Bisa diprogram dalam multi-bahasa
  5. Bisa diintegrasikan dengan kafka
- 

# CONTOH HASIL PROCESSING LAYER

Data Awal

```
{  
  "user_id": "12345",  
  "comment": "Dasar monyet kamu!",  
  "timestamp": "2025-05-30T14:03:10Z"  
}
```

Data setelah processing

```
{  
  "user_id": "12345",  
  "comment": "Dasar monyet kamu!",  
  "timestamp": "2025-05-30T14:03:10Z",  
  "category": "racist",  
  "confidence_score": 0.93  
}
```

# MODEL MACHINE LEARNING

Model machine learning yang kami gunakan adalah model machine learning yang kami latih dalam batch training. Model tersebut akan di deploy/dihubungkan dengan MLFlow yang sudah di konversi ke ONNX.

# ESTIMASI KEBUTUHAN SPARK NODE

- Master Node → 1, untuk menjadwalkan dan mengatur eksekusi job ML dan mengelola cluster spark.
- Worker Nodes → 3-5, untuk menjalankan task dan stage ML, preprocessing teks, dan inference model. Memproses komentar secara paralel.

# **STORAGE DATA LAYER >>>**

P

Daya Saing

# APА ITU STORAGE LAYER?

## 1.Fungsi utama storage layer

Storage layer bergungsi untuk menyimpan data yang sudah diolah di processing layer (spark streaming) untuk disimpan.

Storage Layer adalah lapisan dalam arsitektur sistem data yang berfungsi untuk menyimpan data, baik data mentah (raw), data yang telah diproses, maupun data hasil analisis.

tools yang dipakai : MongoDB ,  
Elastisearch

# MENGAPA MENGGUNAKAN MONGODB?

## 1. Stabil dalam penyimpanan data

- MongoDB dirancang sebagai database yang mampu menangani beban kerja operasional yang tinggi, termasuk insert (penyimpanan), update, dan query data secara terus-menerus dan cepat.

## 2. sharding

- MongoDB dirancang untuk scale-out dengan sharding. ini artinya mongo db dapat menambah node untuk mendistribusikan beban penyimpanan

## 3. query dan analisis ringan

- MongoDB menggunakan query language yang mensupport data JSON Ini sangat intuitif bagi pengembang yang bekerja dengan data JSON karena struktur kueri mencerminkan struktur data yang disimpan.

# BAGAIMANA DATA DIPROSES KE MONGODB?

1. Apache spark streaming membada data dari realtime dari kafka
  - spark akan melakukan data cleaning untuk menghapus noise
  - menginferensi data dengan model machine learning yang sudah dilatih
  
2. menyimpan data
  - Data yang sudah di kategorikan di spark akan dimasukan ke mongoDB dengan mongo-spark connector. Ini adalah konektor resmi yang dikembangkan oleh MongoDB itu sendiri.

# BAGAIMANA DATA DISIMPAN DALAM MONGODB

data dalam mongoDB akan disimpan di dalam bentuk BSON atau binary JSON. Ini adalah format serialisasi data biner yang dirancang untuk menjadi ringan, dapat dilalui dengan cepat, dan efisien.

## Contoh data json yang disimpan

```
{  
  "user_id": "12345",  
  "comment": "Dasar monyet kamu!",  
  "timestamp": "2025-05-30T14:03:10Z",  
  "category": "racist",  
  "confidence_score": 0.93  
}
```

# MENGAPA HARUS MENGGUNAKAN ELASTIC SEARCH?

## 1. Pencarian data yang sangat cepat

- elasticsearch sangat fleksibel untuk melakukan pencarian data karena mempunyai kemampuan pencarian full-text search misal untuk filtering komentar negatif dalam hitungan detik

## 2. indexing data sangat fleksibel

- data disimpan dalam format JSON dan di-index dengan sangat fleksibel dan mampu menangani struktur data kompleks (nested fields, array ,dll)

## 3. integrasi mudah dengan spark dan kafka

- spark dapat mengirim langsung hasil infrensi ke elasticsearch
- elastic search mempunyai API yang kuat

# VISUALITATION LAYER



P

Mr Daya Saing

# APA ITU VISUALIZATION LAYER?

Layer ini merupakan tahap akhir dalam arsitektur big data yang bertugas untuk:

- Menganalisis hasil klasifikasi komentar (misalnya: positif, negatif, spam, toxic, dll)
- Menyajikan data dalam bentuk visual secara real-time agar mudah dipahami pengguna akhir
- Memberikan insight berdasarkan hasil proses machine learning dan data pipeline.

Pada layer ini Kami Menggunakan Power BI sebagai Toolsnya. Power BI digunakan untuk mengolah, menganalisis, dan memvisualisasikan data dalam bentuk grafik, tabel, dashboard interaktif, dan laporan dinamis.

Mengapa Memakai Tools Power BI?

- Cocok digunakan untuk menyajikan hasil filter komentar kepada pihak non-teknis
- Mendukung koneksi ke berbagai sumber data seperti MongoDB dan Elasticsearch.
- Memudahkan dalam membuat dashboard interaktif dan real-time.

# CARA KERJA POWER BI

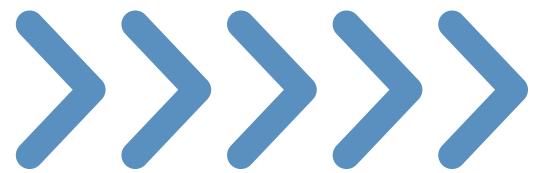
untuk membuat dashboard realtime dari power BI kami menggunakan pendekatan streaming dataset. Streaming Dataset di Power BI adalah jenis dataset khusus yang memungkinkan pengguna mengirimkan data secara real-time langsung ke Power BI melalui REST API.

streaming dataset menggunakan metode pendorongan data (push), sehingga sangat cocok untuk menampilkan data yang terus berubah, seperti data sensor, komentar live, atau transaksi. Dengan memanfaatkan fitur ini, Power BI mampu menampilkan visualisasi yang terus diperbarui seiring dengan masuknya data baru tanpa perlu melakukan refresh manual.

# APA HASIL DARI VISUALIZATION LAYER

- Dashboard Interaktif untuk melihat:
  1. Jumlah komentar berdasarkan kategori (positif, negatif, spam, dll).
  2. Trend data over time.
  3. Distribusi komentar berdasarkan waktu, pengguna, platform, dll.
  4. melihat user mana yang sering berkata kasar agar bisa di ban
- Insight Bisnis atau Operasional seperti:
  1. Kapan waktu komentar spam paling banyak muncul?
  2. Seberapa efektif model ML dalam memfilter komentar tidak relevan?
  3. Visualisasi performa model klasifikasi (berapa persen komentar spam berhasil difilter).
- Insight perilaku pengguna, seperti:
  1. Waktu paling sering munculnya komentar toxic.
  2. Topik/kata kunci yang sering muncul di komentar negatif.

# PENUTUP



# TERIMA KASIH

