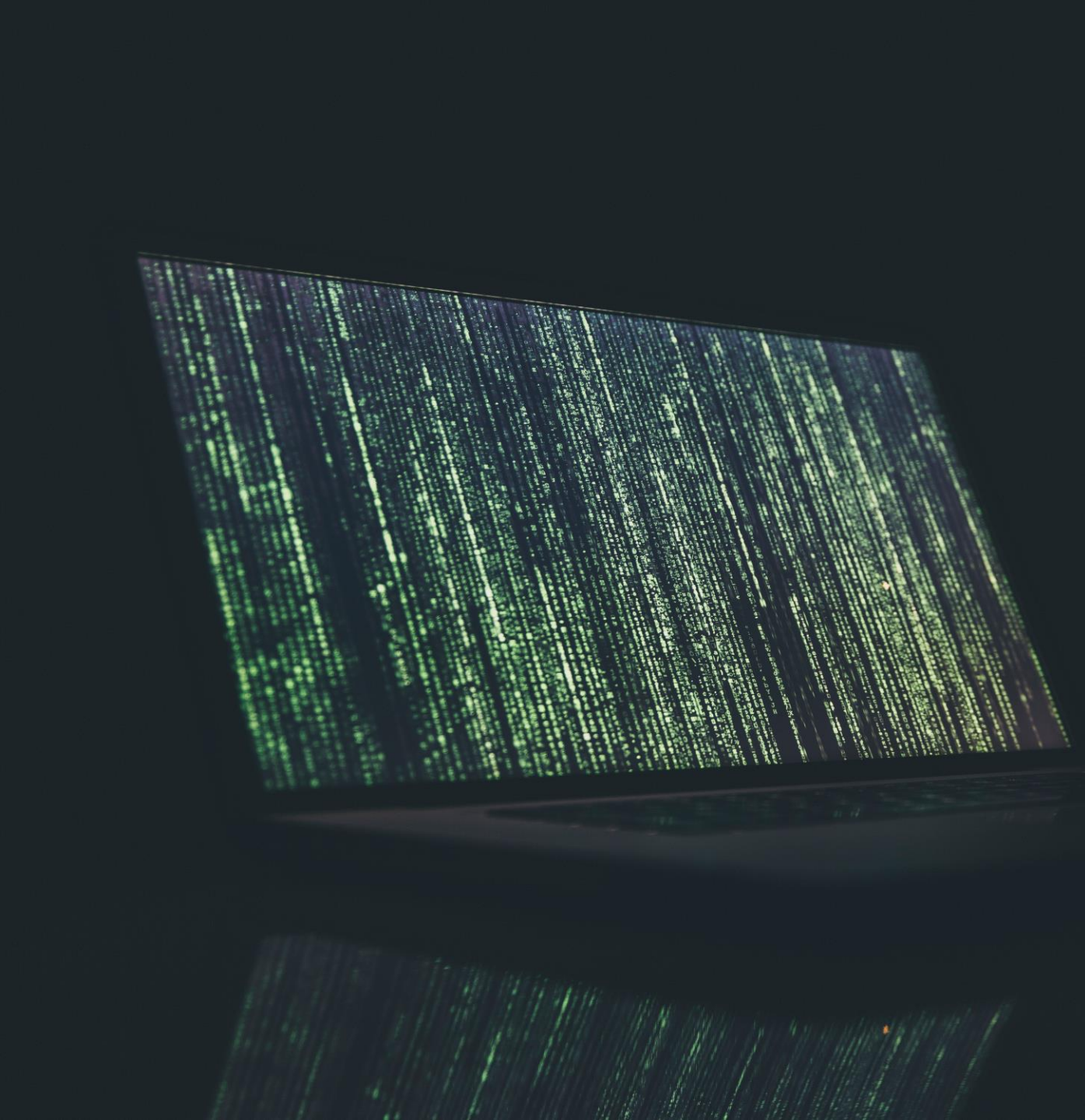


PERTEMUAN 5

TEKNOLOGI PEMROSESAN BIG DATA SECARA *REAL-TIME*

Mata Kuliah:

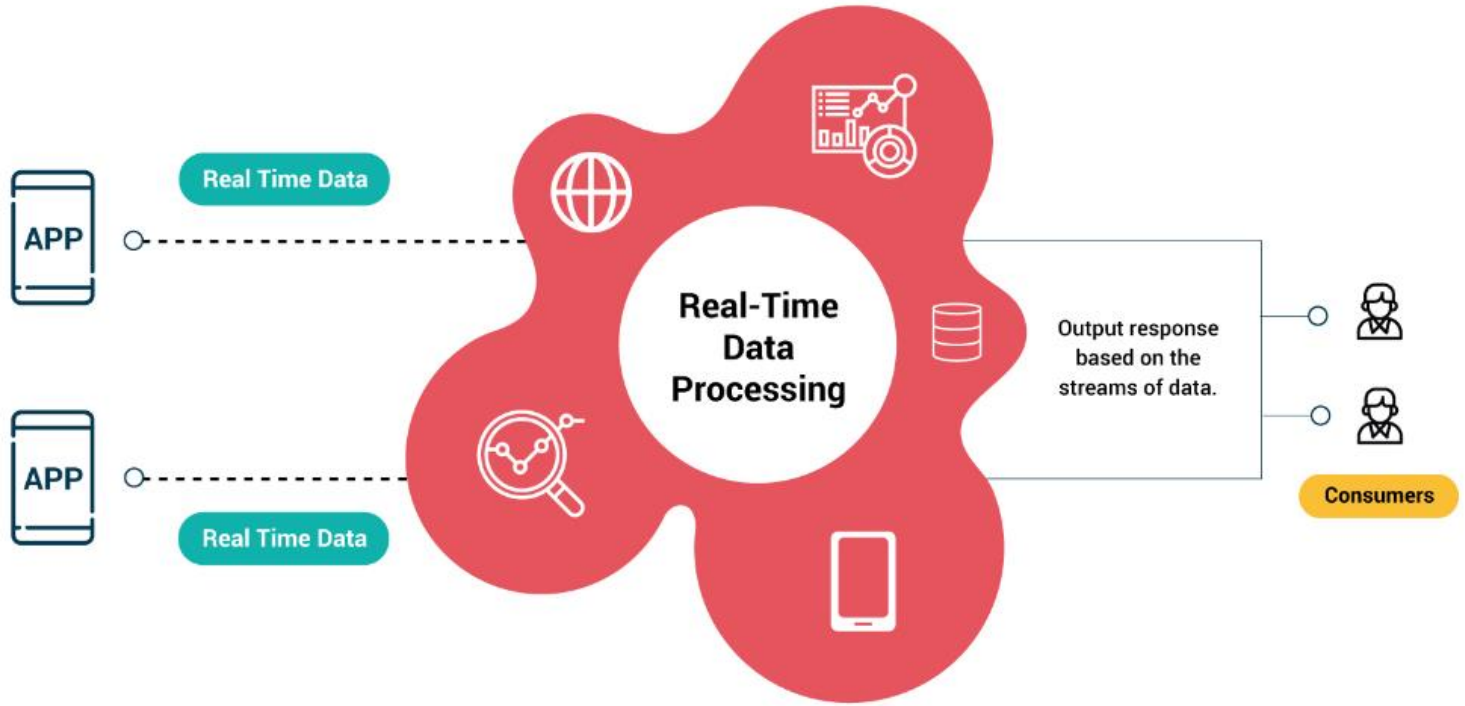
Infrastruktur Dan Teknologi Big Data

Dosen: Galih Hermawan, S.Kom., M.T.

Prodi Teknik Informatika. FTIK.

Universitas Komputer Indonesia

KONSEP PEMROSESAN *REAL-TIME* DALAM ANALISIS BIG DATA



Sumber gambar.
https://axual.com/top-things-to-know-about-real-time-data-processing/

- Definisi Pemrosesan *Real-time*:
  - Pemrosesan data secara instan saat data dihasilkan atau diterima.
- Keuntungan Pemrosesan *Real-time*:
  - Respons cepat terhadap perubahan data.
  - Mendukung pengambilan keputusan langsung.
  - Memungkinkan deteksi dan tindakan cepat.

RINGKASAN PEMROSESAN REAL-TIME

- Proses menganalisis data segera setelah data tersebut tersedia di dalam sistem.
- Mengaplikasikan logika dan matematika untuk memberikan wawasan yang lebih cepat dan akurat tentang data tersebut, sehingga dapat mendukung pengambilan keputusan yang lebih baik dan tepat.
- Pemrosesan *real-time* berbeda dengan pemrosesan *batch*, yang menunggu data terkumpul dalam jumlah tertentu sebelum diproses.
- Pemrosesan *real-time* juga berbeda dengan pemrosesan *near-real-time*, yang memiliki sedikit jeda atau latensi antara data masuk dan data keluar.

MENGAPA PEMROSESAN *REAL-TIME* PENTING UNTUK BIG DATA

- Big Data adalah data yang memiliki volume, variasi, dan kecepatan yang sangat besar, sehingga membutuhkan metode pemrosesan dan analisis yang berbeda dengan data tradisional.
- Big Data dapat memberikan banyak manfaat bagi bisnis, seperti meningkatkan efisiensi, produktivitas, inovasi, dan keunggulan kompetitif.
- Namun, Big Data juga memiliki tantangan, seperti data yang kompleks, heterogen, inkonsistensi, dan tidak lengkap.

KEGUNAAN

- Memanfaatkan data yang paling baru dan relevan untuk mendapatkan wawasan yang lebih akurat dan mendalam.
- Merespon perubahan dan kejadian yang terjadi secara cepat dan tepat, sehingga dapat mengurangi risiko, biaya, dan kerugian.
- Meningkatkan kualitas dan kepuasan pelanggan dengan memberikan layanan yang lebih responsif, personalisasi, dan rekomendasi.
- Mendorong inovasi dan diferensiasi produk dengan menggabungkan data dari berbagai sumber dan melakukan eksperimen secara cepat.

KERANGKA

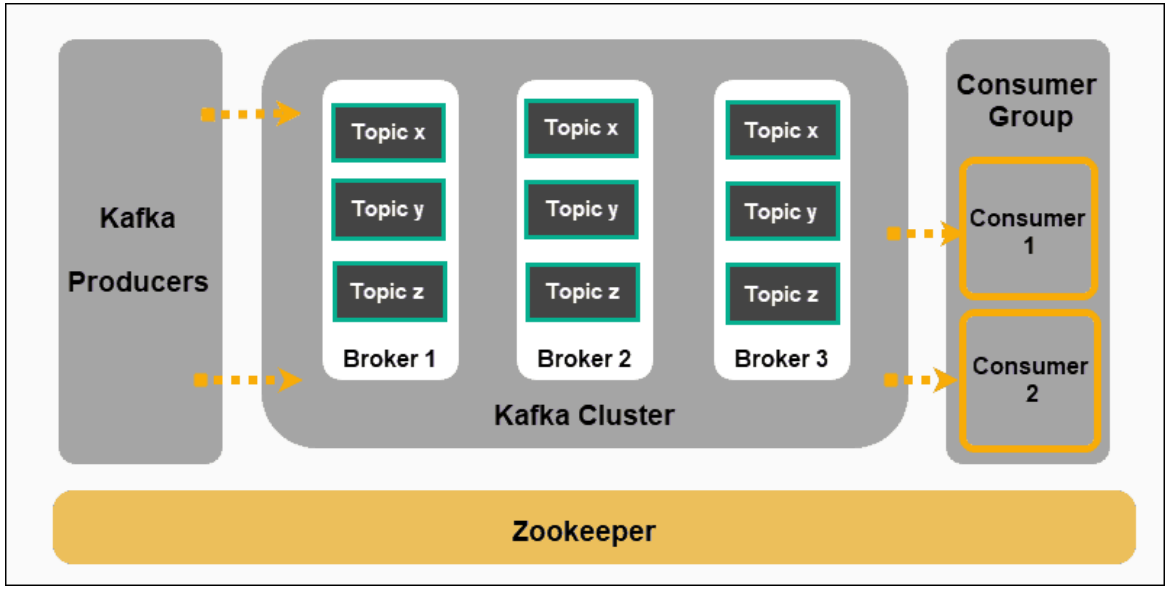
- Apache Kafka
  - Kerangka pemrosesan aliran data yang dapat digunakan untuk memproses data yang mengalir secara terus-menerus.
  - Merupakan sistem pesan terdistribusi yang dapat menangani data dengan volume dan kecepatan yang sangat tinggi.
  - Mengandalkan konsep topik dan partisi untuk menyimpan dan mengirimkan data.
  - Dapat digunakan sebagai antarmuka antara berbagai sumber data dan aplikasi pemrosesan data, seperti *streaming* media, analitik *real-time*, dan IoT.
- Apache Storm
  - Kerangka pemrosesan aliran data yang dapat digunakan untuk menjalankan tugas-tugas kompleks secara paralel.
  - Mengandalkan konsep *spout* dan *bolt* untuk menerima dan memanipulasi data.
  - Dapat digunakan untuk melakukan operasi seperti agregasi, filter, join, dan interaksi dengan sumber data dan basis data seperti analisis *streaming*, pembelajaran mesin, dan deteksi anomali.

APACHE KAFKA - PENGENALAN

- Definisi:
  - Apache Kafka adalah platform distribusi *streaming open-source* yang dirancang untuk menangani aliran data secara *real-time*.
- Tujuan:
  - Memungkinkan publikasi dan langganan data *real-time*.
  - Memberikan toleransi kesalahan dan skalabilitas horizontal.
- Arsitektur:
  - Terdiri dari produsen (*producer*), makelar (*broker*), dan konsumen (*consumer*).
  - Data diorganisir dalam topik dan dapat dibagi menjadi partisi.

Sumber gambar.
https://kafka.apache.org/

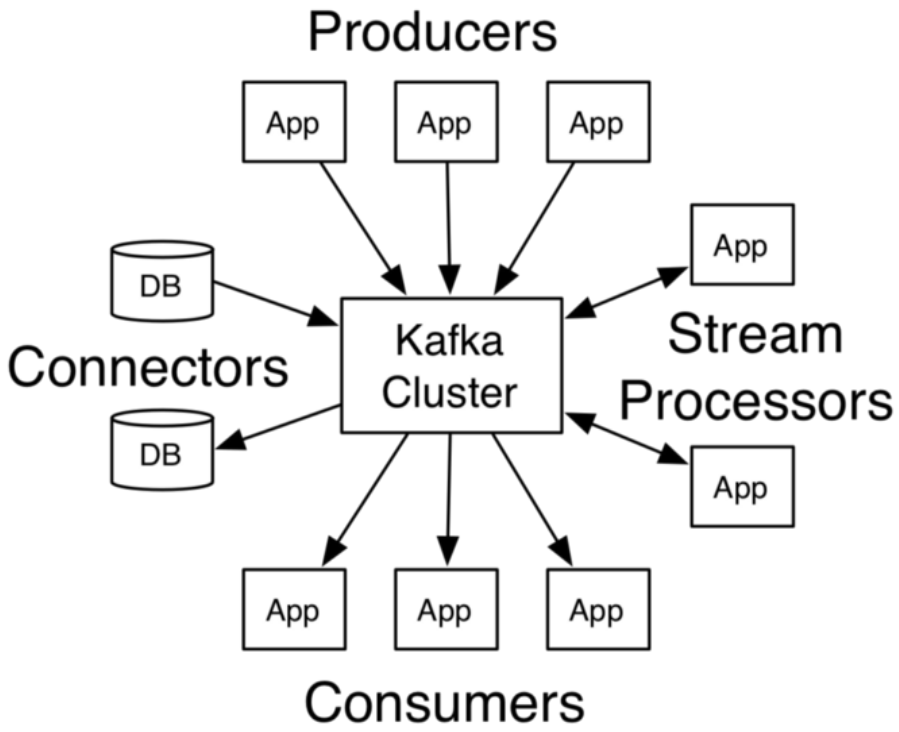
APACHE KAFKA - KONSEP



- Producer:
  - Entitas yang menghasilkan dan mengirimkan data ke topik Kafka.
- Consumer:
  - Entitas yang berlangganan dan mengonsumsi data dari topik Kafka.
- Broker:
  - Server Kafka yang bertanggung jawab untuk menyimpan dan mengelola data.
  - Mengkoordinasi antara produsen dan konsumen.
- Topik:
  - Kategori atau saluran yang digunakan untuk mengorganisir data dalam Kafka.
  - Semua pesan dikelompokkan dalam satu atau lebih topik.
- Partisi:
  - Pembagian fisik dari topik yang memungkinkan data terdistribusi secara efisien.
  - Setiap partisi dikelola oleh broker yang berbeda.

Sumber gambar.
https://phoenixnap.com/kb/kafka-on-kubernetes

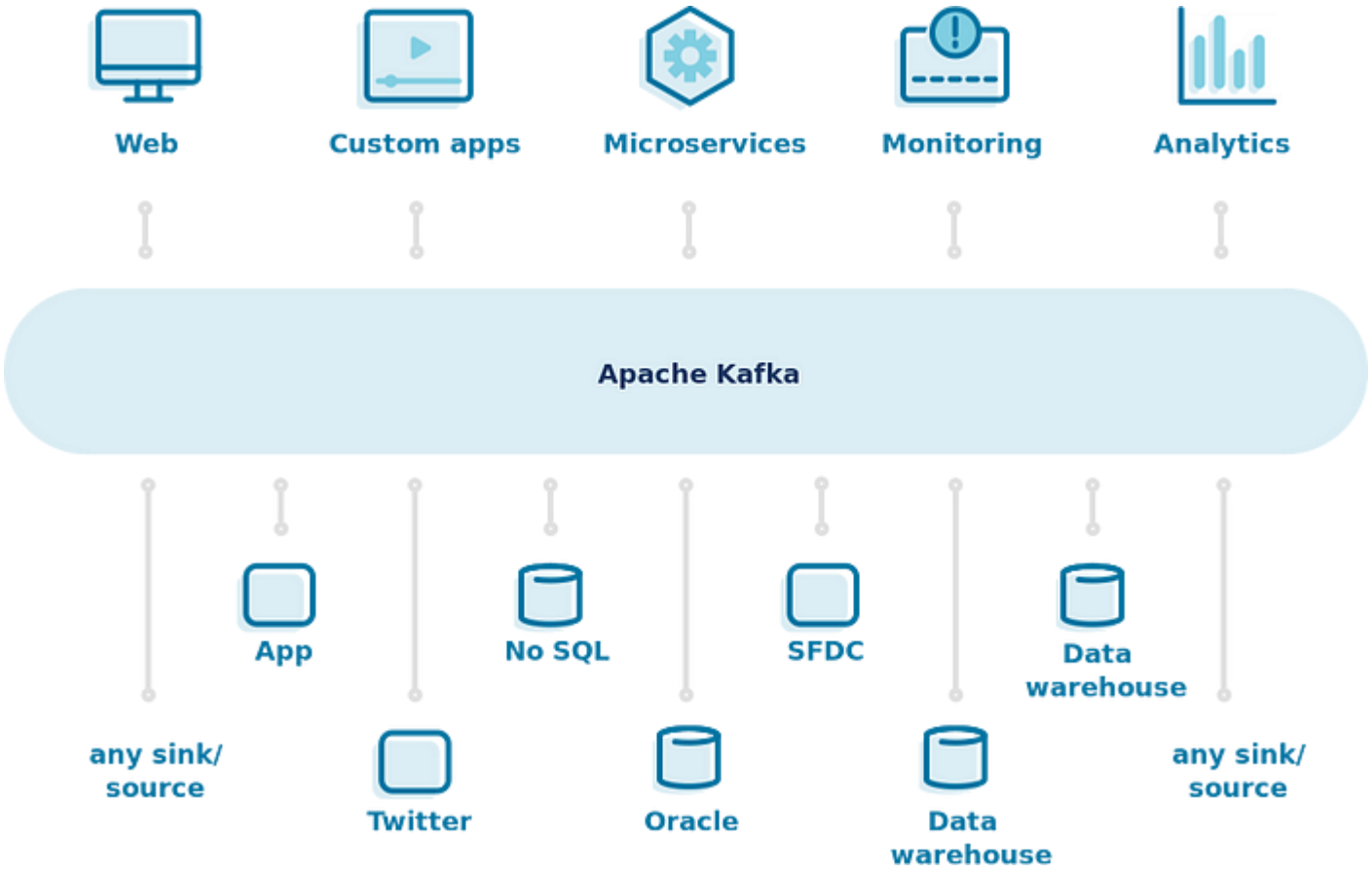
APACHE KAFKA - FITUR



Sumber gambar.
https://kafka.apache.org/

- **Producers**, aplikasi yang mengirimkan data ke *topic*
- **Consumers**, aplikasi yang mendengarkan data dari *topic*
- **Connectors**, Integrasi dengan aplikasi eksternal, sebagai contoh basis data
- **Stream Processors**, mengolah data yang berbentuk streams (data yang mengalir terus, sebagai contoh *metric*)

APACHE KAFKA - IMPLEMENTASI



Sumber gambar.
https://www.confluent.io

- Produksi dan Konsumsi Pesan *Real-time*:
  - Produsen mengirimkan pesan ke topik secara *real-time*.
  - Konsumen dapat mengonsumsi pesan tersebut segera setelah tersedia.
- Kelebihan:
  - Skalabilitas tinggi karena dapat menangani banyak produsen dan konsumen.
  - Toleransi kesalahan yang baik melalui replikasi partisi.
- Kekurangan:
  - Memerlukan konfigurasi dan manajemen yang cermat.
  - Kompleksitas dalam pengelolaan partisi dan replikasi.

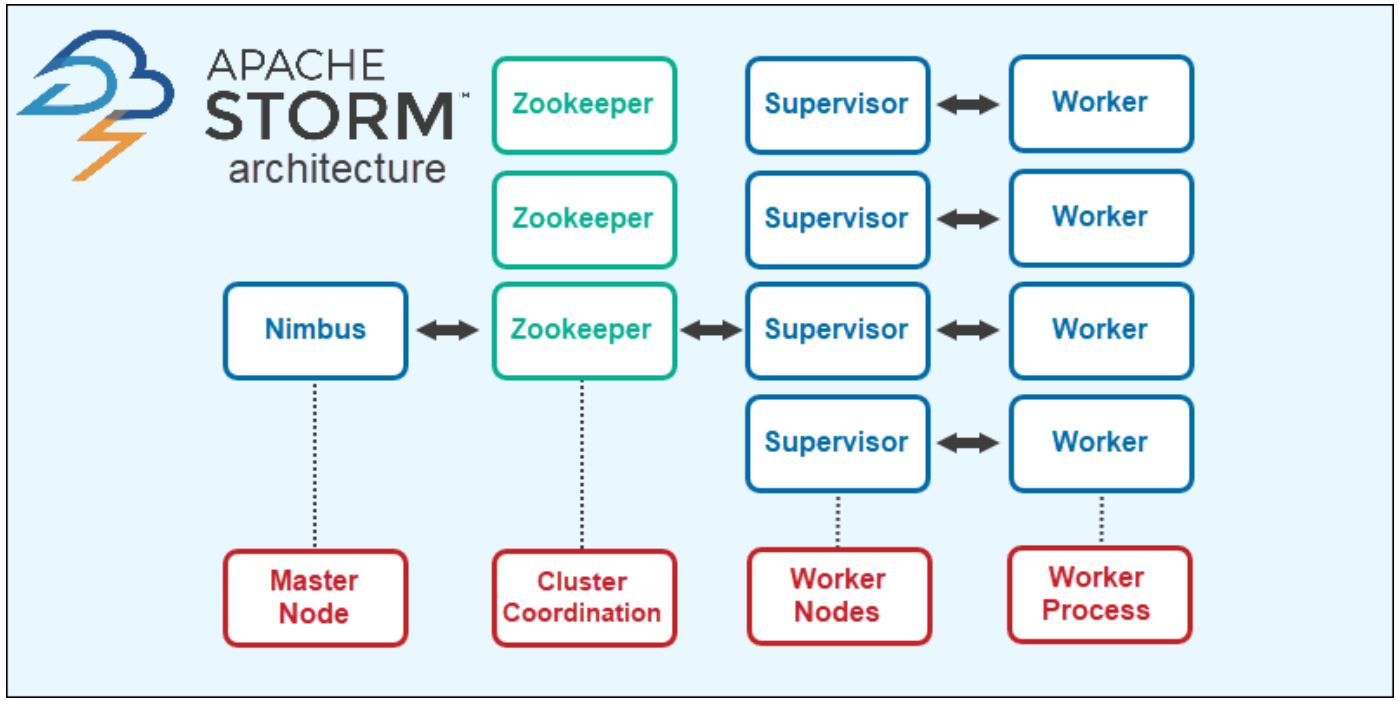
APACHE STORM - PENGENALAN

- Definisi:
  - Apache Storm adalah sistem pemrosesan *streaming open-source* yang dirancang untuk mengolah data secara *real-time* dan mendukung analisis data secara distribusi.
- Tujuan:
  - Menangani aliran data secara cepat dan dapat diandalkan.
  - Memberikan skalabilitas horizontal dan toleransi kesalahan.
- Arsitektur:
  - Terdiri dari *Nimbus* (pengelola kluster), *Supervisor* (menjalankan pekerjaan), dan *Zookeeper* (koordinasi).

APACHE STORM - KONSEP UTAMA

- Topologi:
  - Rangkaian aliran data yang didefinisikan untuk pemrosesan.
  - Mencakup sumber daya (*Spout*) dan operasi (*Bolt*).
- Spout:
  - Komponen yang mengambil data dari sumber eksternal dan mengirimkannya ke dalam topologi Storm.
- Bolt:
  - Komponen yang melakukan operasi pemrosesan pada data yang diterima dari *Spout* atau *Bolt* sebelumnya.
- Garis Waktu Pemrosesan Storm:
  - Menunjukkan seberapa cepat data dapat diproses melalui topologi Storm.
  - Penting untuk memahami keterlambatan data dalam sistem.

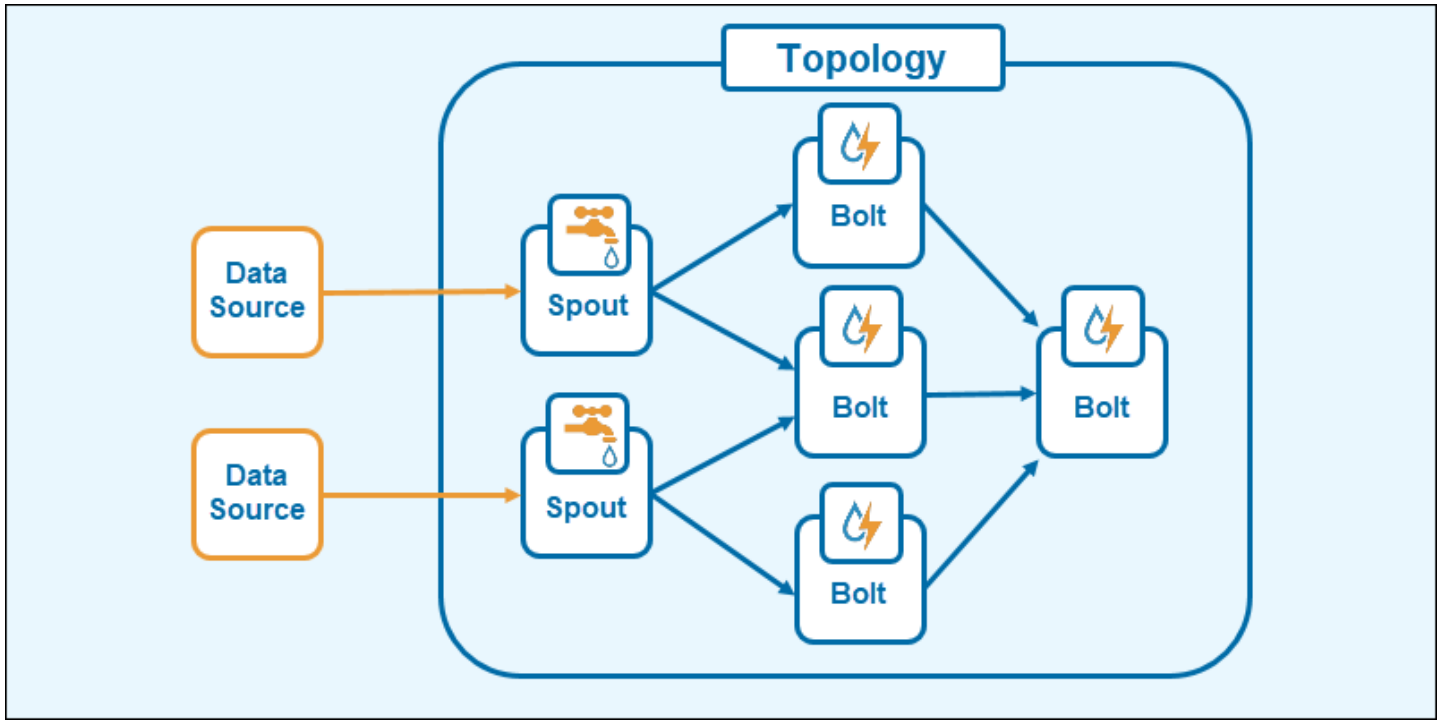
APACHE STORM – ARSITEKTUR BADAI APACHE



Sumber gambar.
https://phoenixnap.com/kb/apache-storm

- Apache Storm menggunakan arsitektur master-slave dengan komponen berikut:
  - *Nimbus* adalah server yang berada pada satu node master.
  - *Supervisor* adalah layanan yang berjalan di setiap node pekerja.
  - *Pekerja* adalah satu atau beberapa proses pada setiap node yang dimulai oleh supervisor. Para pekerja menjalankan penanganan input data paralel dan mengeluarkan data ke database atau sistem file.
  - *Zookeeper* mengoordinasikan dan mengelola proses data yang didistribusikan.

APACHE STORM – TOPOLOGI BADAI APACHE



- Topologi Apache Storm mirip dengan pekerjaan MapReduce di Hadoop. Topologinya terdiri dari:
  - *Spouts* adalah titik masuk aliran data dalam topologi. *Spout* terhubung ke sumber data, mengambil data secara terus-menerus, mengubah informasi menjadi aliran tupel, dan mengirim data ke baut.
  - *Baut* menyimpan logika pemrosesan. Baut menjalankan berbagai fungsi, agregasi, penggabungan aliran, pemfilteran tuple, dll. Outputnya membuat aliran baru untuk pemrosesan tambahan melalui baut lain atau menyimpan data dalam database.

Sumber gambar.
https://phoenixnap.com/kb/apache-storm

APACHE STORM - IMPLEMENTASI

- Pembuatan Topologi Pemrosesan:
  - Mendefinisikan Spout dan Bolt dalam topologi untuk pemrosesan data.
  - Konfigurasi jumlah dan distribusi komponen untuk skalabilitas.
- Integrasi dengan Apache Kafka:
  - Menggunakan Spout untuk mengonsumsi data dari topik Kafka.
  - Menerapkan Bolt untuk operasi pemrosesan atau transformasi data.

TEKNIK PEMROSESAN REAL-TIME

- *Streaming analytics*
  - Teknik analisis data yang dilakukan secara terus-menerus terhadap aliran data.
  - Dapat digunakan untuk mengidentifikasi pola atau tren dalam data yang mengalir.
- *Machine learning*
  - Dapat diterapkan untuk pembelajaran mesin pada data *real-time*.
  - Dapat digunakan untuk memprediksi perilaku atau kejadian di masa depan.
- *Event-driven architecture*
  - Arsitektur yang dirancang untuk merespons peristiwa yang terjadi secara *real-time*.
  - Dapat digunakan untuk membangun aplikasi yang responsif terhadap perubahan data.

CONTOH PENERAPAN

- **Deteksi penipuan**: Pemrosesan real-time dapat digunakan untuk menganalisis transaksi keuangan secara cepat dan mendeteksi pola yang mencurigakan, sehingga dapat menghentikan atau mencegah penipuan sebelum terjadi.
- **Rekomendasi produk**: Pemrosesan real-time dapat digunakan untuk menganalisis perilaku dan preferensi pelanggan secara cepat dan memberikan rekomendasi produk yang sesuai dengan kebutuhan dan minat mereka.
- **Optimisasi harga**: Pemrosesan real-time dapat digunakan untuk menganalisis permintaan dan penawaran pasar secara cepat dan menyesuaikan harga produk sesuai dengan kondisi pasar yang berubah-ubah.
- **Analisis sentimen**: Pemrosesan real-time dapat digunakan untuk menganalisis data media sosial secara cepat dan mengukur sentimen publik terhadap merek, produk, atau topik tertentu, sehingga dapat meningkatkan reputasi dan loyalitas pelanggan.
- **Prediksi permintaan**: Pemrosesan real-time dapat digunakan untuk menganalisis data historis dan data real-time secara cepat dan memprediksi permintaan produk atau layanan di masa depan, sehingga dapat meningkatkan efisiensi dan produktivitas.

TANYA JAWAB

Terima Kasih