HACKTIV8

Assignment 1

HACKTIV8

Studi Kasus

Case

- Apache Kafka adalah platform streaming terdistribusi yang memungkinkan pengelolaan data real-time dengan throughput tinggi dan toleransi kesalahan. Dalam tugas ini, Anda diminta untuk mengeksplorasi konsep-konsep dasar Kafka, termasuk arsitektur, API Producer & Consumer, manajemen topik, dan integrasi dengan sistem eksternal. Tujuan Anda adalah untuk mengimplementasikan solusi Kafka untuk streaming dan pemrosesan data real-time.

Requirement Kafka Infrastructure

HACKTIV8

Buat topic berikut:

- orders-topic (partisi: 3, replication factor: 2)
- logs-topic (TTL: 7 hari, log compaction enabled)
- Implementasikan idempotent producer pada service pengirim order.

Requirement Service Springboot

HACKTIV8

Order Producer (Spring Boot):

- REST API menerima payload order (JSON) → kirim ke orders-topic

Order Consumer (Kafka Consumer):

- Konsumsi dari orders-topic → simpan ke MySQL table transactions
- Log setiap proses ke logs-topic (format: timestamp, service_name, status, error_message)

Kafka Streams App:

- Hitung total transaksi per jam dari orders-topic → output ke topic hourly-transaction-topic

Requirement Integration

HACKTIV8

Kafka Connect:

- Setup JDBC Sink Connector untuk sinkron hourly-transaction-topic → MySQL table hourly_summary
- Setup Elasticsearch Sink Connector untuk sinkron logs-topic → Elasticsearch index shopstream-logs

Monitoring:

- Buat Kibana dashboard menampilkan:
- Error rate dari log
- Throughput transaksi per jam

Task

Task 1: Local Setup

- Docker compose untuk Kafka cluster (1 zookeeper, 2 brokers), MySQL, Elasticsearch
- Konfigurasi topic via kafka-topics.sh

Task 2: Spring Boot Producer

Code snippet:

```
@RestController
public class OrderController {
    @Autowired
    private KafkaTemplate<String, String> kafkaTemplate;

    @PostMapping("/order")
    public String sendOrder(@RequestBody Order order) {
        kafkaTemplate.send("orders-topic", order.toJSON());
        return "Order received!";
    }
}
```

Task

Task 3: Kafka Streams Aggregation

Code snippet untuk windowed aggregation:

```
KStream<String, Order> stream = builder.stream("orders-topic");
stream.groupByKey()
      .windowedBy(TimeWindows.of(Duration.ofHours(1)))
      .count()
      .toStream()
      .to("hourly-transaction-topic");
```

Task

Task 4: Connector Configuration

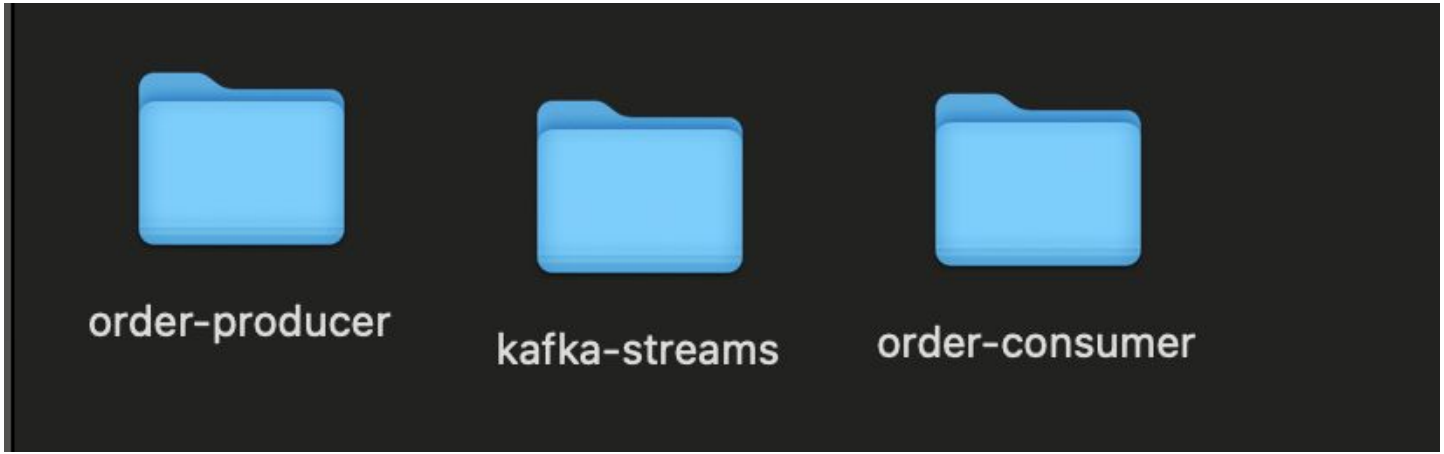
Contoh config JDBC Sink (connector-mysql.json):

```
{
  "name": "mysql-sink",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
    "connection.url": "jdbc:mysql://mysql:3306/shopstream",
    "topics": "hourly-transaction-topic",
    "table.name.format": "hourly_summary"
  }
}
```

Task

Task 5:

Silahkan gunakan starter code springboot yang ada untuk membantu



Deliverables

Endpoint yang dibuat kurang lebih.

Order Producer Service (Spring Boot):

POST /api/orders

Menerima payload order dalam format JSON dan mengirimkannya ke Kafka topic orders-topic.

Deliverables

Order Consumer: Menggunakan Kafka Consumer API untuk membaca dari orders-topic dan menyimpan ke MySQL.

Kafka Streams App: Memproses data dari orders-topic secara otomatis tanpa endpoint.

Kafka Connect: Menggunakan endpoint REST Kafka Connect bawaan (misal: POST /connectors) untuk konfigurasi sinkronisasi JDBC dan Elasticsearch.

Monitoring: Kibana akan mengakses data langsung dari Elasticsearch dan MySQL.

Deliverables

Buatlah Diagram Arsitektur (Gunakan draw.io/LucidChart) yang menampilkan:

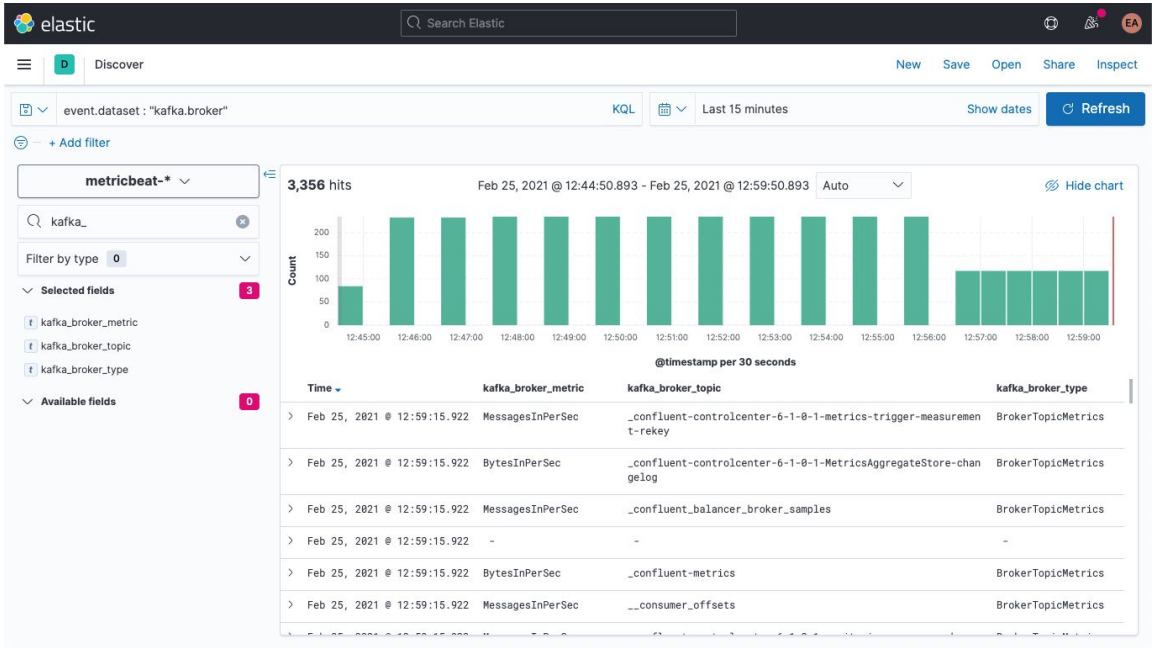
- Alur data dari API → Kafka → Sistem eksternal
- Posisi Kafka Connect dalam pipeline

Verifikasi Hasil:

- Screenshot Kibana dashboard
- Log consumer (tampilkan order masuk + error handling)

Deliverables

- Contoh tampilan Kibana



Deliverables

- Contoh consumer log jika berhasil

```
[2023-10-05 14:30:15,123] INFO [Consumer clientId=consumer-group1-1, groupId=group1]
Subscribed to topic(s): test-topic (org.apache.kafka.clients.consumer.KafkaConsumer)

[2023-10-05 14:30:15,456] INFO [Consumer clientId=consumer-group1-1, groupId=group1]
Discovered coordinator kafka1:9093 (id: 2147483647 rack: null) (org.apache.kafka.clients.consum
er.internals.ConsumerCoordinator)

[2023-10-05 14:30:15,789] INFO [Consumer clientId=consumer-group1-1, groupId=group1]
Successfully joined group with generation Generation{generationId=123, memberId='consumer-group
1-1-abc123', protocol='range'} (org.apache.kafka.clients.consumer.internals.ConsumerCoordinato
r)

[2023-10-05 14:30:15,890] INFO [Consumer clientId=consumer-group1-1, groupId=group1]
Assigned partitions: [test-topic-0, test-topic-1, test-topic-2, test-topic-3, test-topic-4, tes
t-topic-5] (org.apache.kafka.clients.consumer.internals.ConsumerCoordinator)

[2023-10-05 14:30:16,000] INFO [Consumer clientId=consumer-group1-1, groupId=group1]
Successfully authenticated using SSL (org.apache.kafka.common.network.SslChannelBuilder)
```

Thank you