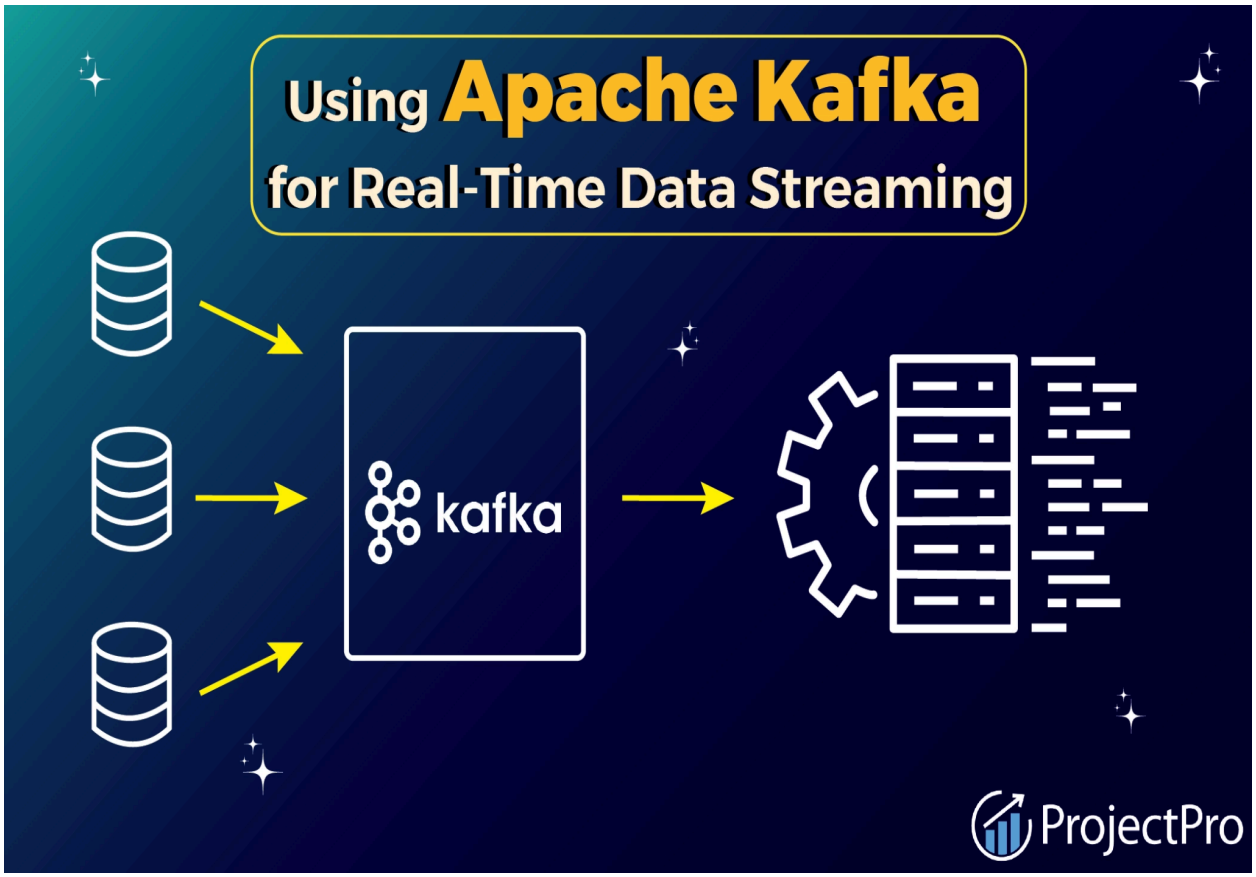
KAFKA

Your Name:- Lokesh Rajendra Khadse
Your Company :- Eidiko Systems Integrators
EmpID :- 1177

What is kafka?

Kafka is a publish-subscribe messaging system that retrieves data from various sources and makes it available in real time to target systems when they are ready.

Diagram



Component of Kafka

1. Producer :- publishes events or messages to kafka.
2. Consumer :- Consumes or retrieves messages or events from Kafka
3. Broker :- A Kafka server that acts as a middleman between producers and consumers, facilitating data exchanges.
4. Clusters:- Multiple Brokers called clusters that are working for a common purpose.
5. Topic:- It is present inside the topic , it is used to categorize different types of messages.
6. Partitions : - It is inside the topic , it breaks kafka topics into multiple parts called partitions. Each part is called a partition.
7. Offset:-A unique identifier for each message within a partition, used to track the position of consumers. It helps keep track of which messages have been consumed.
8. Zookeeper :- This all process done or managed by zookeeper .Manages and coordinates Kafka brokers, maintaining metadata

Steps

1 . start zookeeper

:- bin\windows\zookeeper-server-start.bat config\zookeeper.properties

2. Start Kafka Server

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

3. Create topic

:- bin\windows\kafka-topics.bat –create –topic topic1 topic2 –partitions3 –replication-factor -–bootstrap-server localhost:9092

4. Create producer

:- bin\windows\kafka-console-producer.bat –topic topic1 topic2 -–bootstrap-server localhost:9092

5 create consumer

:- :- bin\windows\kafka-console-consumer.bat –topic topic1 topic2 –from-beginning -–bootstrap-server localhost:9092

Default port:

1 . zookeeper :- 2181

2. KafkaServer:- 9092

Coding part

Dependency:

```
<dependency>
   <groupId>org.springframework.kafka</groupId>
   <artifactId>spring-kafka</artifactId>
</dependency>
```

Application properties for producer

```
spring.kafka.producer.bootstrap-servers=localhost:9092
spring.kafka.producer.key-serializer=org.apache.kafka.common.serialization.StringSerializer
spring.kafka.producer.value-serializer=org.apache.kafka.common.serialization.StringSerializer
```

Kafka config

```
package com.DeliveryBoy.config;


import org.apache.kafka.clients.admin.NewTopic;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.kafka.config.TopicBuilder;
```

```
@Configuration
public class KafkaConfig {

   @Bean
   public NewTopic topic(){                    //(step1)topic create here



       return TopicBuilder
               .name(AppConstants.LOCATION_TOPIC_NAME)    //
"location-update-topic"
//                .partitions()
//                .replicas()
               .build();


   }
}
```

Producer service

```
package com.DeliveryBoy.Service;


import com.DeliveryBoy.config.AppConstants;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.stereotype.Service;


@Service
public class KafkaService {       //this is service is used to send msg
```

```
   @Autowired
   private KafkaTemplate<String,String>kafkaTemplate;  //by using this we can
send msg

   private Logger logger = LoggerFactory.getLogger(KafkaService.class);


   public boolean updateLocation(String location){

       this.kafkaTemplate.send(AppConstants.LOCATION_TOPIC_NAME,location);
//topic,data
       this.logger.info("message/location produced");
       return true;    //if exception not occur then return true

   }
}
```

Producer controller

```
package com.DeliveryBoy.Controller;

import com.DeliveryBoy.Service.KafkaService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.Map;

@RestController
```

```
@RequestMapping("/location")
public class LocationController {


   @Autowired
   private KafkaService kafkaService;


   @PostMapping("/update")
   public ResponseEntity<?> updateLocation(){


       this.kafkaService.updateLocation("( " + Math.round(Math.random() * 100)
+ " , " + Math.round(Math.random() * 100) + " )");
      return new ResponseEntity<>(Map.of("message","location updated"),
HttpStatus.OK);
   }
}
```

Consumer

Dependency

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
   <groupId>org.springframework.kafka</groupId>
   <artifactId>spring-kafka</artifactId>
</dependency>
```

Application properties

```
spring.application.name=EndUser


//Consumer configuration

server.port=8082

spring.kafka.consumer.bootstrap-servers=localhost:9092

spring.kafka.consumer.group-id-group-1

spring.kafka.consumer.auto-offset-reset=earliest

spring.kafka.consumer.key-deserializer=org.apache.kafka.common.serialization.StringDeserializer

spring.kafka.consumer.value-deserializer=org.apache.kafka.common.serialization.StringDeserializer
```

Config

```
package com.enduser.EndUser;


import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.kafka.annotation.KafkaListener;


@Configuration

public class KafkaConfig {



   @KafkaListener(topics=AppConstants.LOCATION_UPDATE_TOPIC , groupId = AppConstants.GROUP_ID )

   public void updatedLocation(String value){


       System.out.println(value);

   }

}
```

```
package com.enduser.EndUser;


public class AppConstants {


   public static final String LOCATION_UPDATE_TOPIC="location-update-topic";


   public  static final String GROUP_ID="group-1";
}
```

OUTPUT

(45, 78)

(44, 80)

(40, 78) .....