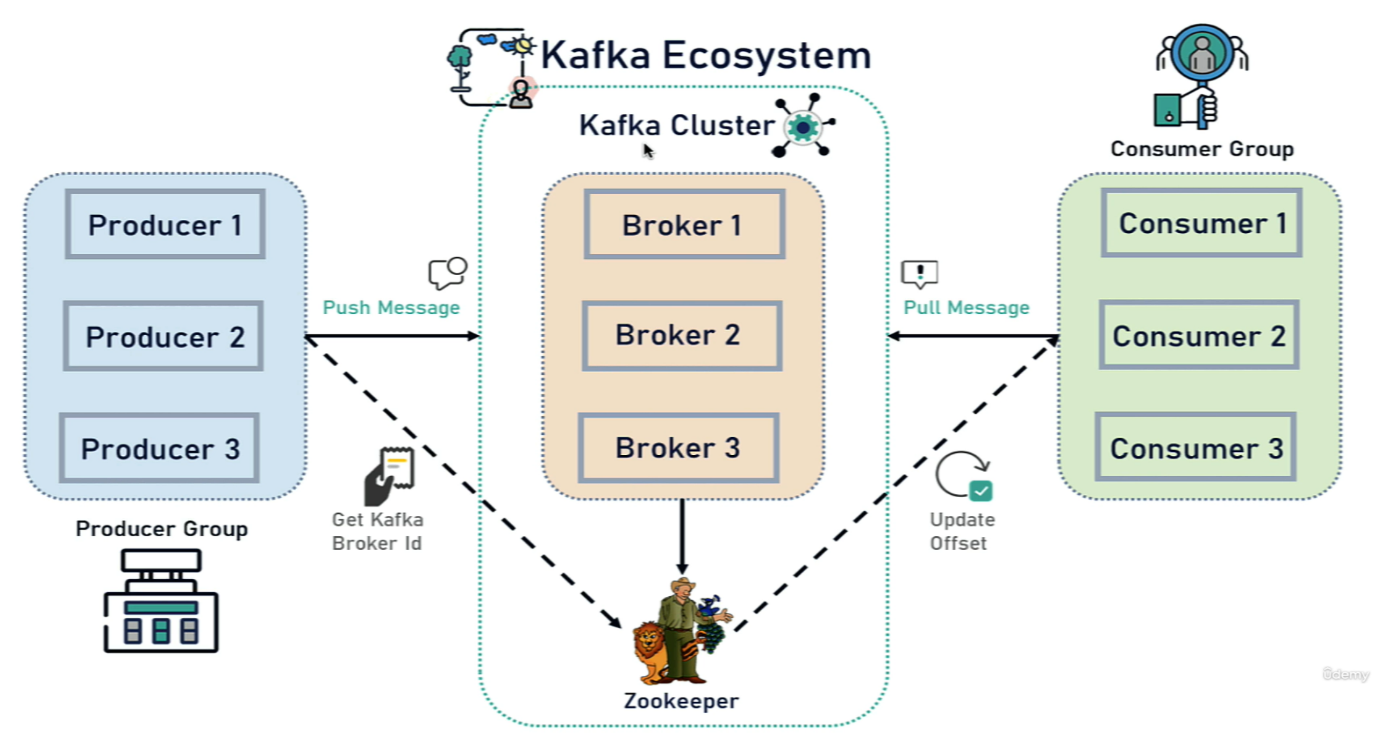
**Apache Kafka**

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

Kafka operates on a publish-subscribe model, where data producers publish records to topics, and data consumers subscribe to these topics to receive and process the data



**Zookeeper**

Zookeeper maintains the state of all the Kafka brokers in the Kafka cluster as well as Zookeeper

maintains the configuration of all the topics, producers, and consumers.

**Kafka Cluster**

A **Kafka cluster** is a distributed system. Kafka Cluster consist of set of brokers. A cluster has a minimum of three brokers. Set of Kafka brokers working together to provide high availability, fault tolerance, and scalability for real-time data streaming applications.

**Kafka broker**

Kafka broker is a Kafka server. The producer and consumer don't interact directly. They use Kafka server as agent or a broker to exchange messages.

**Kafka Topic**

Topics are logical channels that categorize the messages in a Kafka broker and we can create any number of topics in a Kafka broker.

Topic is like a table in a database or folder in a file system. It is identified by a name. Each topic contains a unique name so that Consumers can easily consume the data from that particular topic. You can have any number of topics in the Kafka cluster or Kafka broker.

**Kafka Partitions**

Kafka topic is divided into a number of partitions, which contains records in an unchangeable sequence.

Kafka brokers will store messages for a topic, the capacity of the data can be enormous and it may not possible to store in a single computer. Therefore, it will be partitioned into multiple parts and distributed among multiple computers since Kafka is a distributed system.

**Offsets**

Offset is a sequence of ids given to the messages as they arrived at the partition. Once the Offset is assigned it will never be changed. The first message gets an Offset zero. The next message receives an offset one and so on.

**PRODUCER**

The producer is nothing but application that produces the message and sends it to Kafka

Broker.

**CONSUMER**

Consumer is an application that reads or consumes a message from the Kafka server.

**Consumer Group**

A consumer group contains one or more consumers working together to process the messages.

**Achieving Scalability and Fault Tolerance**

**Scalability**

Kafka achieves scalability through data partitioning, which allows parallel processing of messages across multiple brokers.

**Fault Tolerance**

Fault tolerance in Kafka is achieved by creating multiple copies (replicas) of data across different brokers. If one broker fails, another broker with a copy of the data takes over, ensuring the system keeps running without interruptions. Kafka can also quickly move data around when brokers fail or new ones are added, helping the system recover fast.

**Kafka Commands**

STEP 1: DOWNLOAD AND INSTALL KAFKA

<https://kafka.apache.org/downloads>

STEP 2: START THE KAFKA ENVIRONMENT

# Start the ZooKeeper service

C:\2025\application\kafka>.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

# Start the Kafka broker service

C:\2025\application\kafka>.\bin\windows\kafka-server-start.bat .\config\server.properties

STEP 3: CREATE A TOPIC TO STORE YOUR EVENTS

C:\2025\application\kafka>.\bin\windows\kafka-topics.bat --create --topic topic\_demo --bootstrap-server localhost:9092

STEP 4: WRITE SOME EVENTS INTO THE TOPIC

C:\2025\application\kafka>.\bin\windows\kafka-console-producer.bat --topic topic\_demo --bootstrap-server localhost:9092

>hello world

>topic demo

STEP 5: READ THE EVENTS

C:\2025\application\kafka>.\bin\windows\kafka-console-consumer.bat --topic topic\_demo --from-beginning --bootstrap-server localhost:9092

>hello world

>topic demo

**Spring Boot + Kafka Producer and Consumer**

**#Kafka Project Setup**

Add dependency -> **Spring for Apache Kafka Messaging (spring-kafka)** Publish, subscribe, store, and process streams of records.

**#1#Configure Kafka Producer and Consumer**

**application.properties**

spring.kafka.consumer.group-id: myGroup  
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  
spring.kafka.consumer.value-deserializer: org.springframework.kafka.support.serializer.JsonDeserializer  
spring.kafka.consumer.properties.spring.json.trusted.packages=\*  
  
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  
spring.kafka.producer.value-serializer: org.springframework.kafka.support.serializer.JsonSerializer

**#2#Create Kafka Topic**

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 KafkaTopicConfig {  
  
 @Bean  
 public NewTopic javaguides\_json(){  
 return TopicBuilder.*name*("javaguides\_json").build()**;** }  
}

**#3#Create Kafka Producer**

import net.javaguides.springboot.payload.User**;**import org.slf4j.Logger**;**import org.slf4j.LoggerFactory**;**import org.springframework.kafka.core.KafkaTemplate**;**import org.springframework.kafka.support.KafkaHeaders**;**import org.springframework.messaging.Message**;**import org.springframework.messaging.support.MessageBuilder**;**import org.springframework.stereotype.Service**;**@Service  
public class JsonKafkaProducer {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(JsonKafkaProducer.class)**;** private KafkaTemplate<String**,** User> kafkaTemplate**;** public JsonKafkaProducer(KafkaTemplate<String**,** User> kafkaTemplate) {  
 this.kafkaTemplate = kafkaTemplate**;** }  
  
 public void sendMessage(User data){  
  
 *LOGGER*.info(String.*format*("Message sent -> %s"**,** data.toString()))**;** /\*  
 kafkaTemplate.send("javaguides", "String message");  
 \*/  
  
 Message<User> message = MessageBuilder  
 .*withPayload*(data)  
 .setHeader(KafkaHeaders.*TOPIC***,** "javaguides-json")  
 .build()**;** kafkaTemplate.send(message)**;** }  
}

**#4#Create REST API to Send Message**

import net.javaguides.springboot.kafka.JsonKafkaProducer**;**import net.javaguides.springboot.payload.User**;**import org.springframework.http.ResponseEntity**;**import org.springframework.web.bind.annotation.\***;**@RestController  
@RequestMapping("/api/v1/kafka")  
public class JsonMessageController {  
  
 private JsonKafkaProducer kafkaProducer**;** public JsonMessageController(JsonKafkaProducer kafkaProducer) {  
 this.kafkaProducer = kafkaProducer**;** }  
  
 @PostMapping  
 public ResponseEntity publish(@RequestBody User user){  
 kafkaProducer.sendMessage(user)**;** return ResponseEntity.*ok*("message set to topic")**;** }  
}

**#5#Create Kafka Consumer**

import net.javaguides.springboot.payload.User;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.kafka.annotation.KafkaListener;  
import org.springframework.stereotype.Service;  
  
  
@Service  
public class JsonKafkaConsumer {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(JsonKafkaProducer.class);  
  
 @KafkaListener(topics = "javaguides-json", groupId = "myGroup")  
 public void consume(User user){  
 *LOGGER*.info(String.*format*("Json message recieved -> %s", user.toString()));  
 }  
}