# kafka

## Event streaming

- Event streaming is the practice of capturing data in real-time from event sources like sensors, mobile devices, cloud services, and software applications
- Storing these event streams durably for later retrieval; manipulating, processing, and reacting to the event streams in real-time

## Use cases

- To collect and immediately react to customer interactions and orders, such as in retail, the hotel and travel industry, and mobile applications.
- To process payments and financial transactions in real-time
- To track and monitor cars, trucks, fleets, and shipments in real-time, such as in logistics and the automotive industry.

## Kafka combines three key capabilities

- **publish** (write) and **subscribe** (read)
- To **store** streams of events durably and reliably for as long as you want.
- To **process** streams of events

## Main Concepts and Terminology

- An **event** records the fact that "something happened" in the world or in your business.
- Event key: "Alice"
- Event value: "Made a payment of rs 200 to Bob"
- Event timestamp: "Jun. 25, 2020 at 2:06 p.m."

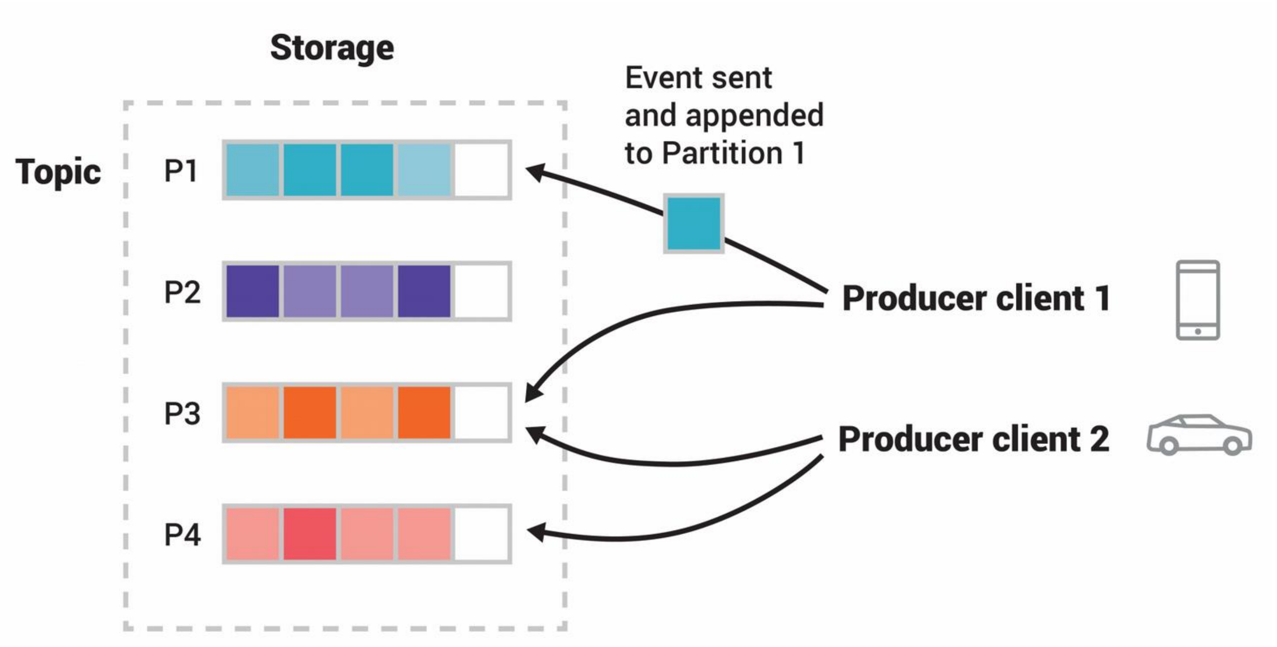
- **Producers** are those client applications that publish (write) events to Kafka, and **consumers** are those that subscribe to (read and process) these events.
- In Kafka, producers and consumers are fully decoupled and agnostic of each other, which is a key design element to achieve the high scalability that Kafka is known for. For example, producers never need to wait for consumers.
- Kafka provides various guarantees such as the ability to process events exactly-once.

- **Servers**: Kafka is run as a cluster of one or more servers that can span multiple datacenters or cloud regions. Some of these servers form the storage layer, called the brokers.

- **Clients**: They allow you to write distributed applications and microservices that read, write, and process streams of events in parallel, at scale

- Events are organized and durably stored in **topics**. Very simplified, a topic is similar to a folder in a filesystem, and the events are the files in that folder

- Topics are **partitioned**, meaning a topic is spread over a number of "buckets" located on different Kafka brokers.

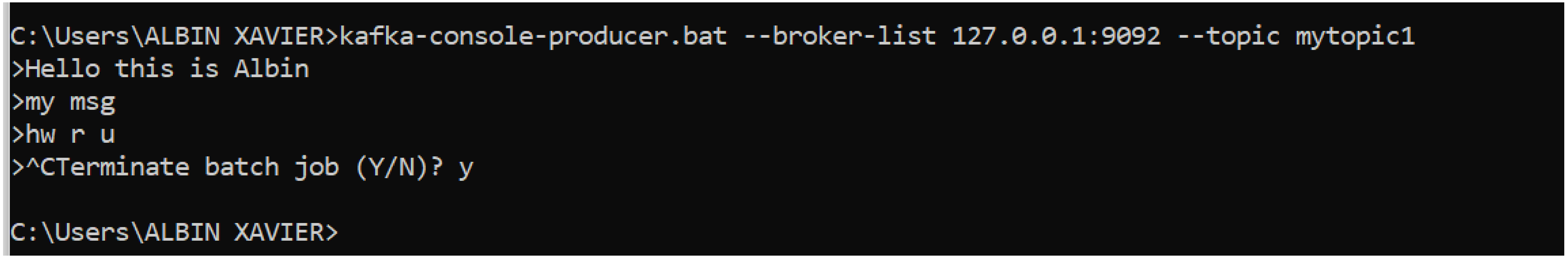


- enabled listener in server.properties as below,
- listeners=PLAINTEXT://localhost:9092

- Started zookeeper server: zookeeper-server-start.bat config\zookeeper.properties
- Started kafka server: kafka-server-start.bat config\server.properties
- Command to list topics:

  kafka-topics --bootstrap-server localhost:**9092**  --list

- To send msgs :
- kafka-console-producer.bat --broker-list 127.0.0.1:9092 --topic mytopic1



```
C:\Users\ALBIN XAVIER>kafka-console-producer.bat --broker-list 127.0.0.1:9092 --topic mytopic1
>Hello this is Albin
>my msg
>hw r u
>^CTerminate batch job (Y/N)? y

C:\Users\ALBIN XAVIER>
```

- **CREATE A TOPIC TO STORE YOUR EVENTS**
- kafka-topics --create --topic mytopic1 --bootstrap-server localhost:9092

- **WRITE SOME EVENTS INTO THE TOPIC**
- kafka-console-producer --topic mytopic1 --bootstrap-server localhost:9092

## To Send KEY

- kafka-console-producer --topic mytopic1 --bootstrap-server localhost:9092 --property "parse.key=true" --property "key.separator=:"

## Topics with partition

- kafka-topics --create --topic mytopic66 --bootstrap-server localhost:9092 --partitions 3

- To describe the partition
- kafka-topics --describe --topic mytopic66 --bootstrap-server localhost:9092
- To get logs such as size
- kafka-log-dirs --describe --bootstrap-server localhost:9092 --topic-list mytopic1

- **READ THE EVENTS**
- kafka-console-consumer --topic quickstart-events --from-beginning --bootstrap-server localhost:9092

- You can stop the consumer client with Ctrl-C at any time.

## Describe abt the topics

- kafka-topics --describe --topic quickstart-events --bootstrap-server localhost:9092