Apache Kafka

Table of Content

- Intro to Kafka
- How Kafka Works
- Role of Zookeeper
- Kafka Commands
- Spark Streaming
- Spark Streaming + Kafka

Intro to Kafka

What is real-time data?

- Real-time data is information that is delivered immediately after collection
- There is no delay in the timeliness of the information provided
- Such data is usually processed using real-time computing
- Examples: Log files, e-commerce purchases, weather events, utility service usage, geo-location of people and things, server activity, etc

What is real-time data?

- Many companies require real-time analytics to understand what's happening across their business units

- Typical industries that rely on real-time data analytics include:
  - Information technology
  - Financial services
  - Transportation
  - Healthcare
  - Advertising

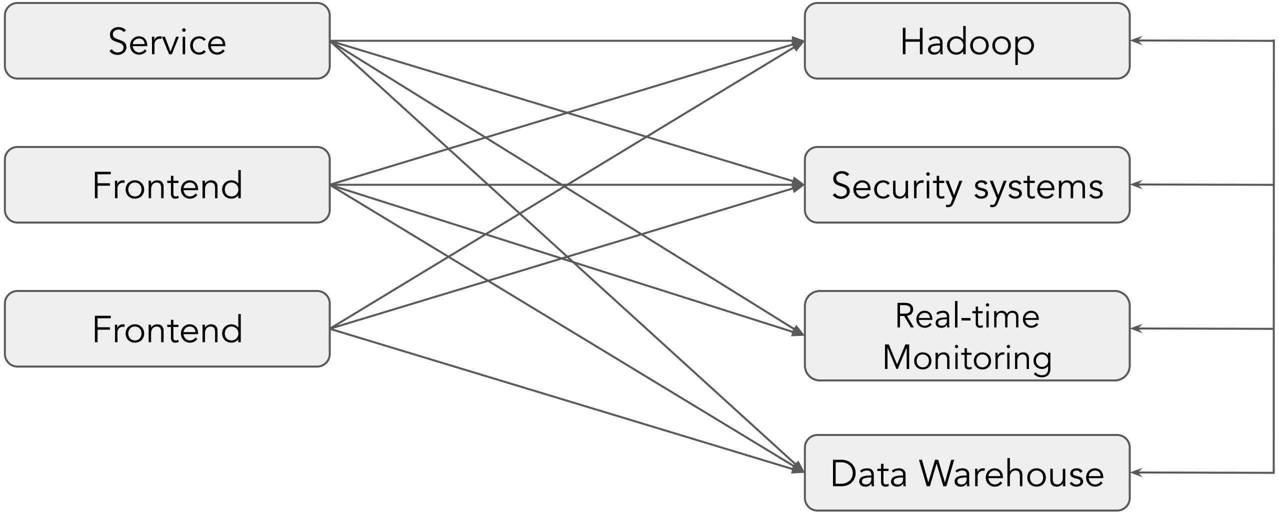
Challenges with real-time data

- Traditional batch distributed data processing methods require data to be downloaded as batches before it can be processed, stored, or analyzed

- Also, point-to-point data pipeline make a system more complicated

Challenges with real-time data

- This is how a typical point-to-point data pipeline looks like



- As you add more components the system gets more complicated

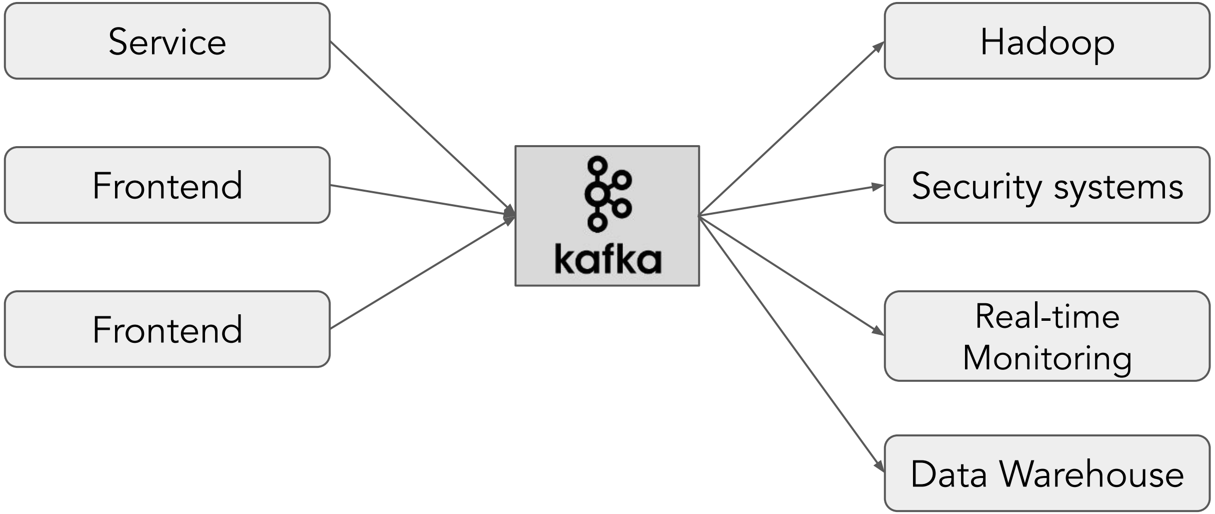
Solution

- This problem can be solved by using messaging system

- Messaging systems provides seamless integration among various distributed endpoints with the help of messages

Solution

- Using the publish-subscribe feature, kafka decouples data pipelines



Intro to Kafka

- Kafka is a distributed streaming platform that is used to publish and subscribe to streams of records

- Kafka is used:
  - to collect big data (real-time streams of data)
  - to do real time analysis or
  - both

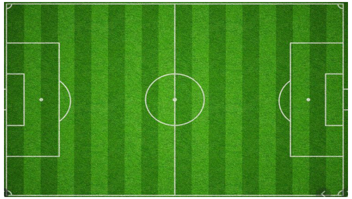
- It can work with Flume, Spark Streaming, Storm, HBase, Flink for real-time ingesting, analysis and processing of streaming data

Kafka is publish-subscribe messaging system and a robust queue that can handle a high volume of data and enables you to pass messages from one end-point to another

How Kafka Works?

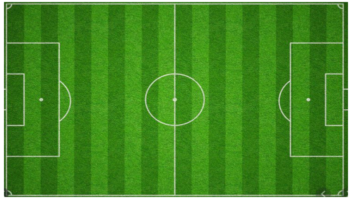
How Kafka Works

- Imagine you were to design a system that listens to various football game updates from various sources



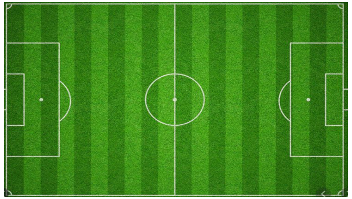
How Kafka Works

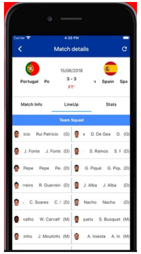
- Such updates might include game scores, countries, players, commentaries and timing information (half time, full-time etc.)



How Kafka Works

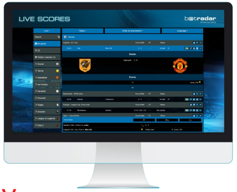
- It then displays the game status on various channels like mobile application, web browsers, etc





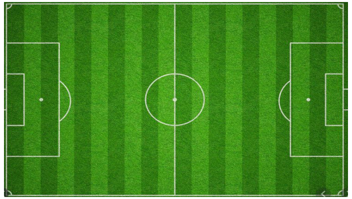
Portugal Po 15/06/2018 3 - 3 FT' n Spain Spa

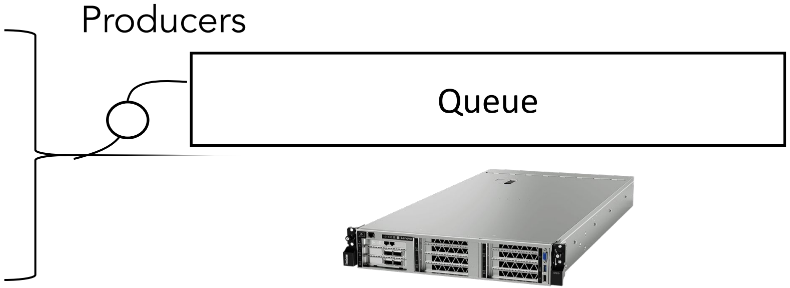
ício Rui Patrício (G) | a D. De Gea D. (G)
J. Fonte J. Fontє (D) | S. Ramos S. F (D)
Pepe Pepe Pe (D) | G. Piqué G. Piqu (D)
rreiro R. Guerrei (D) | J. Alba J. Alba (D)
C. Soares C. : (D) | Nacho Nacho (D)
valho W. Carvalh (M) | quets S. Busquet (M)
inho J. Moutinhc (M) | A. Iniesta A. In (M)

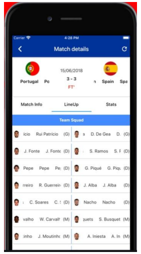


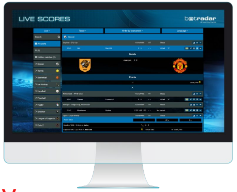
How Kafka Works

- In our architecture, we have a process that reads these updates and writes them in a queue, we call these process as producers since it is producing these updates



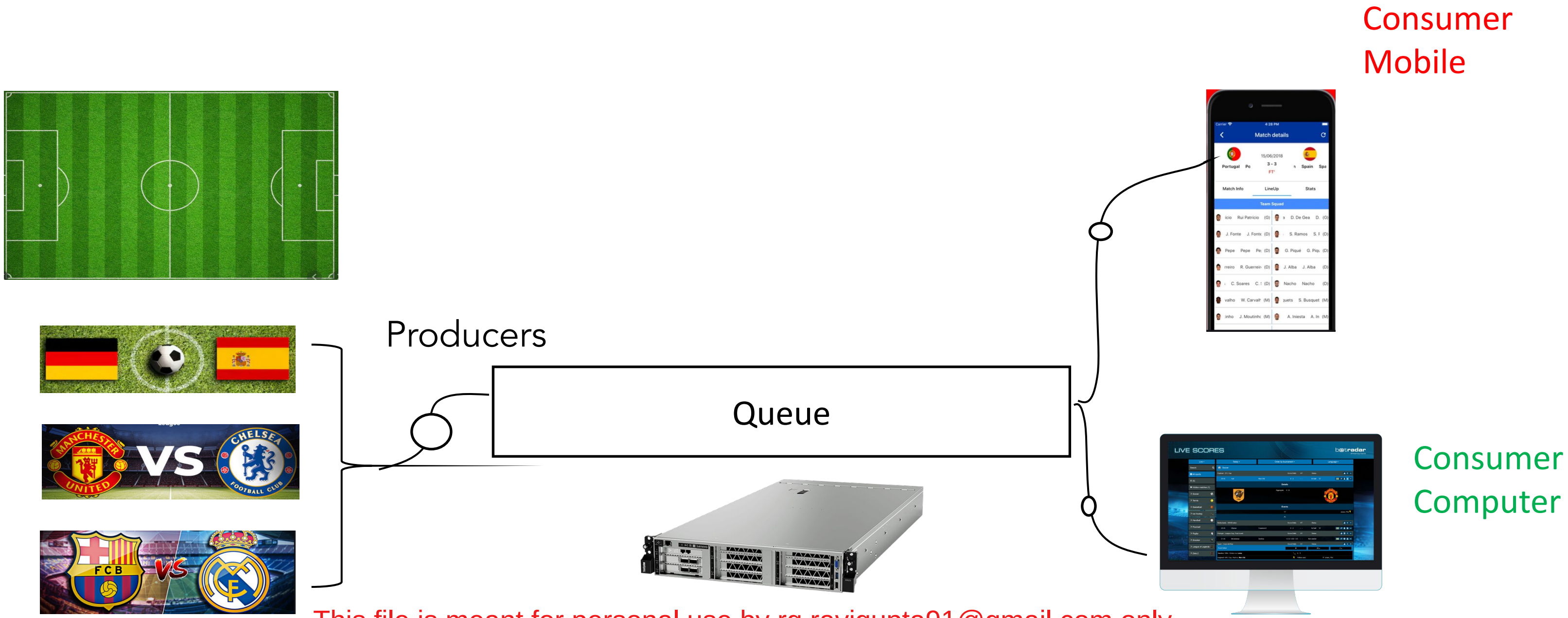






How Kafka Works

- At the head of these queue, we have platforms that consumes these updates to display, we call these platforms as consumers



Consumer Mobile

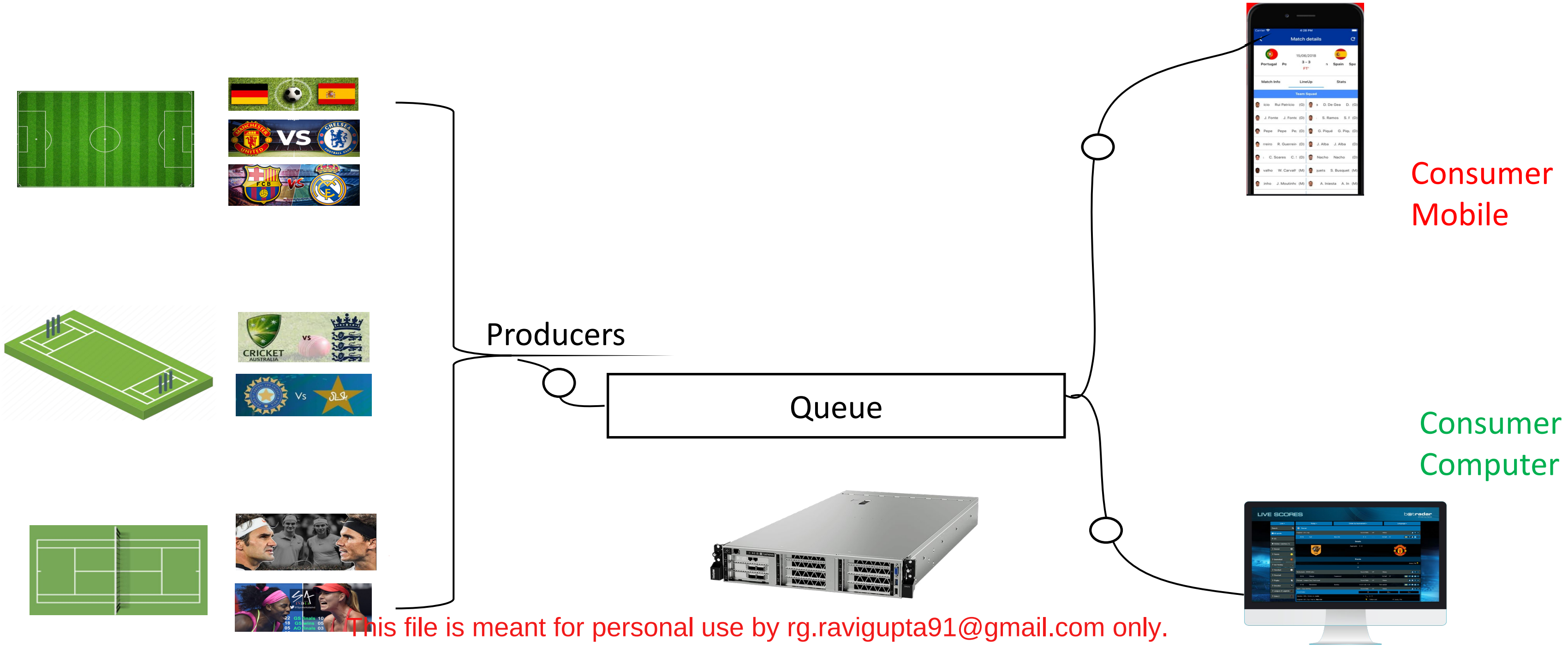
Producers

Queue

Consumer Computer

How Kafka Works

- Over time we decide to expand and start following more and more games

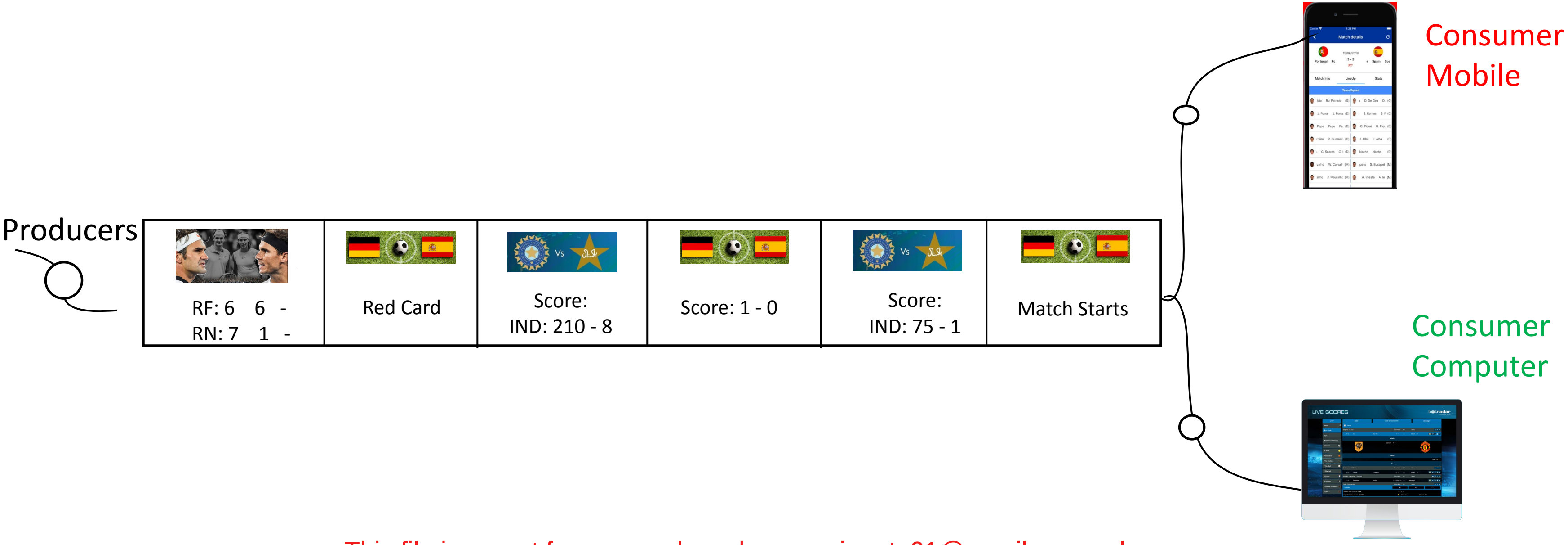


How Kafka Works

- The problem is that the servers are now struggling handling the load
- This is mainly because the queue is hosted on one server which is running out of memory and processing capacity
- At the same time the consumers are also struggling in a similar fashion
- How can we improve the computing power of this architecture?

How Kafka Works

- We need to distribute our architecture



Producers

RF: 6 6 -
RN: 7 1 -

Red Card

Score:
IND: 210 - 8

Score: 1 - 0

Score:
IND: 75 - 1

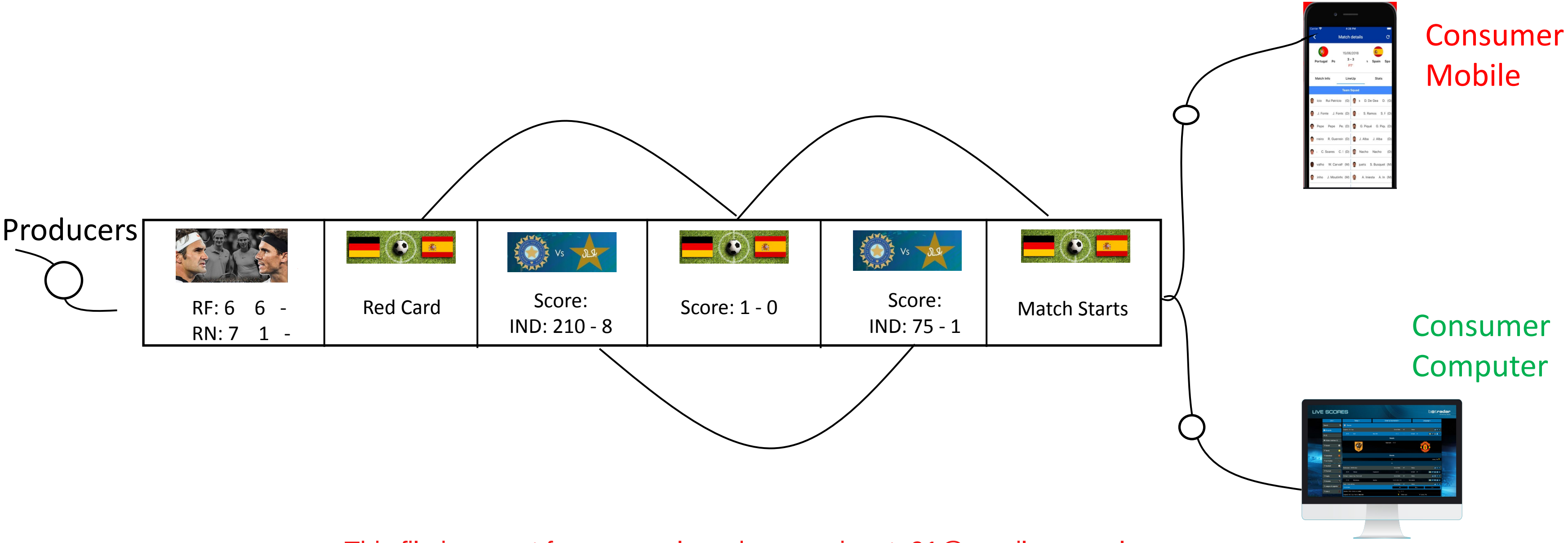
Match Starts

Consumer Mobile

Consumer Computer

How Kafka Works

- By its nature the item in the queue follow specific ordering



Producers

RF: 6 6 -
RN: 7 1 -

Red Card

Score:
IND: 210 - 8

Score: 1 - 0

Score:
IND: 75 - 1

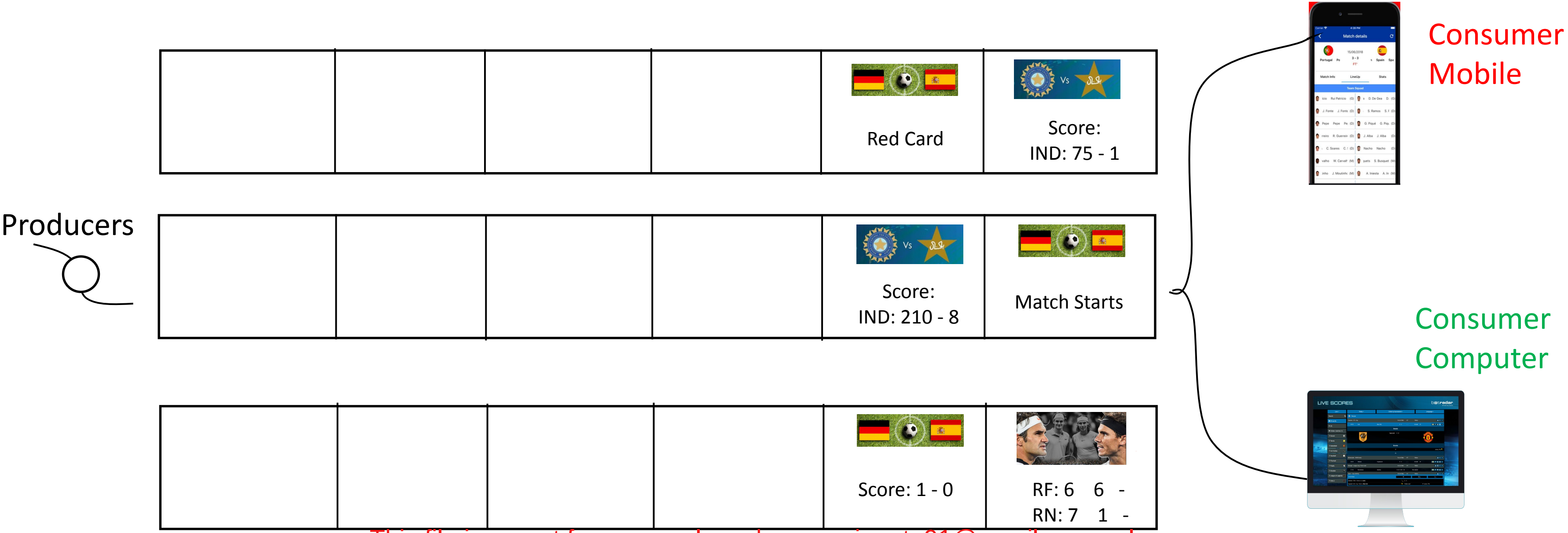
Match Starts

Consumer
Mobile

Consumer
Computer

How Kafka Works

- We could randomly distribute the contents of the queue on to multiple queues



How Kafka Works

- If we do this our consumers might consume the updates in the wrong order
- This would result in inconsistencies. For example the wrong scores being displayed across the channels

How Kafka Works

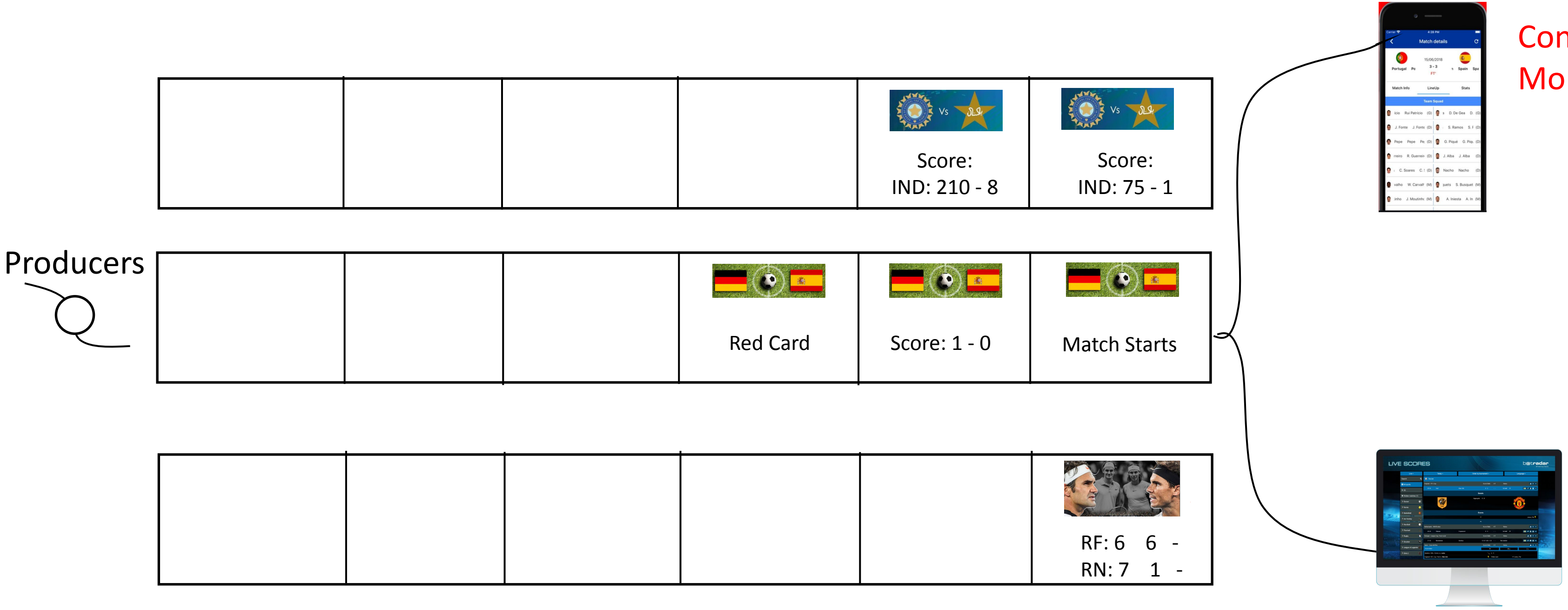
- One solution is to let the application specify to distribute the items in the queue

- In our example we could distribute the item using the match name

- Meaning that the update coming from the same match would be on the same queue

- This strategy would maintain an ordering for football match. This is the basic fundamental difference between kafka and other messaging system i.e the items sent and received by kafka requires a distributed strategy

How Kafka Works



Producers

Score:
IND: 210 - 8

Score:
IND: 75 - 1

Red Card

Score: 1 - 0

Match Starts

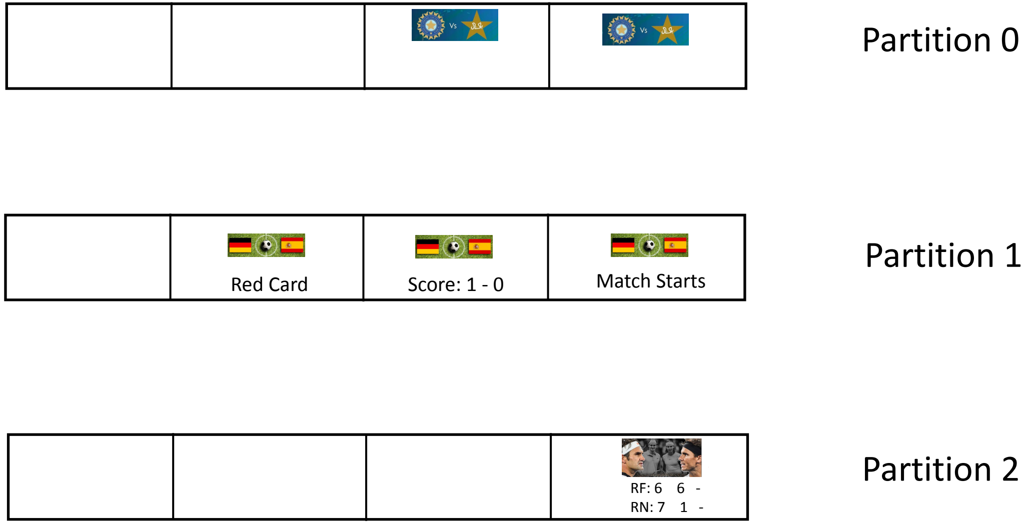
RF: 6 6 -
RN: 7 1 -

Consumer
Mobile

Consumer
Computer

How Kafka Works

- In Kafka, each one of these is called Partitions and the total number of Partitions is called as Partition Counts



Partition 0

Red Card

Score: 1 - 0

Match Starts

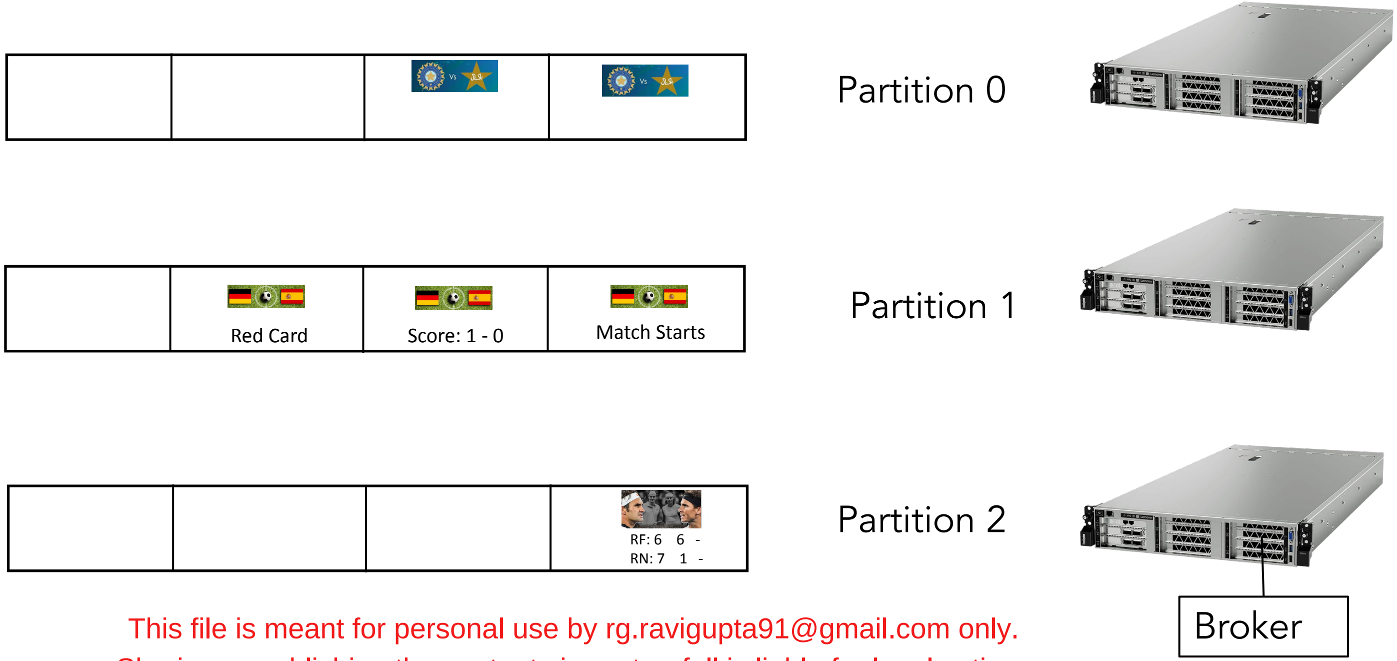
Partition 1

RF: 6 6 -
RN: 7 1 -

Partition 2

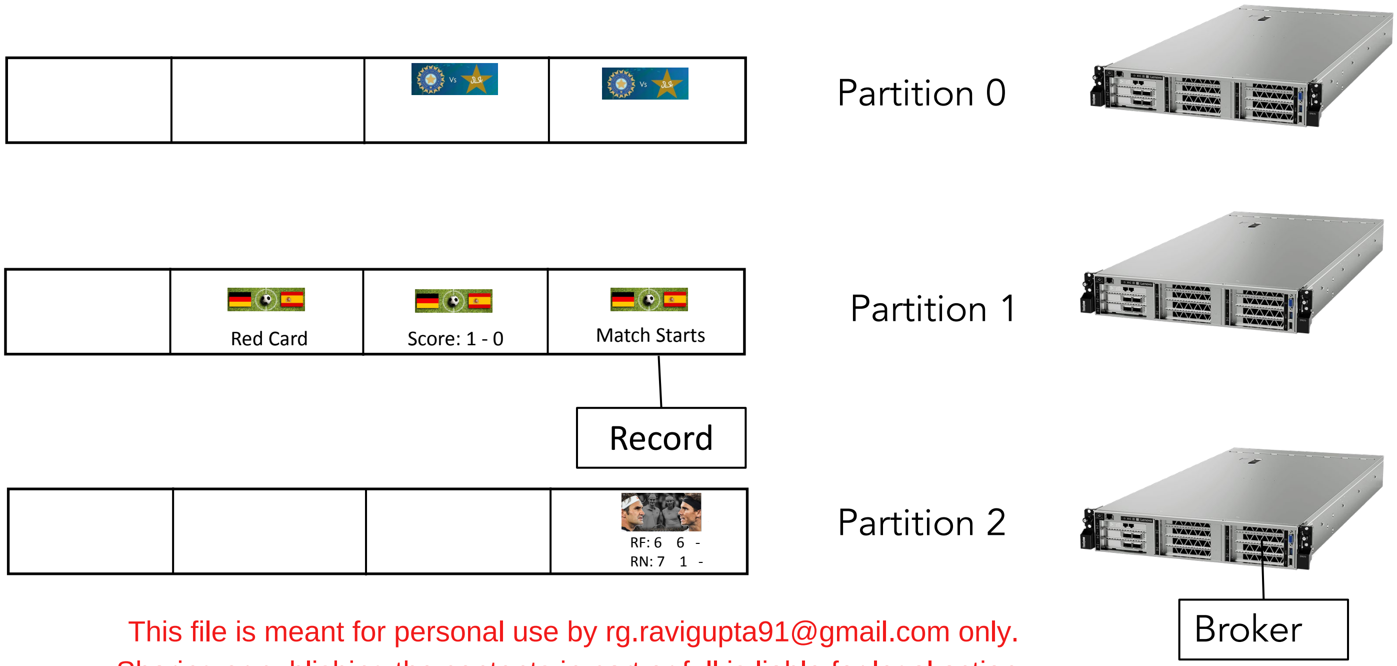
How Kafka Works

- Each server holding one or more of these partitions is called as Broker



How Kafka Works

- Each item in the partition is called as Record



Partition 0

Red Card

Score: 1 - 0

Match Starts

Partition 1

Record

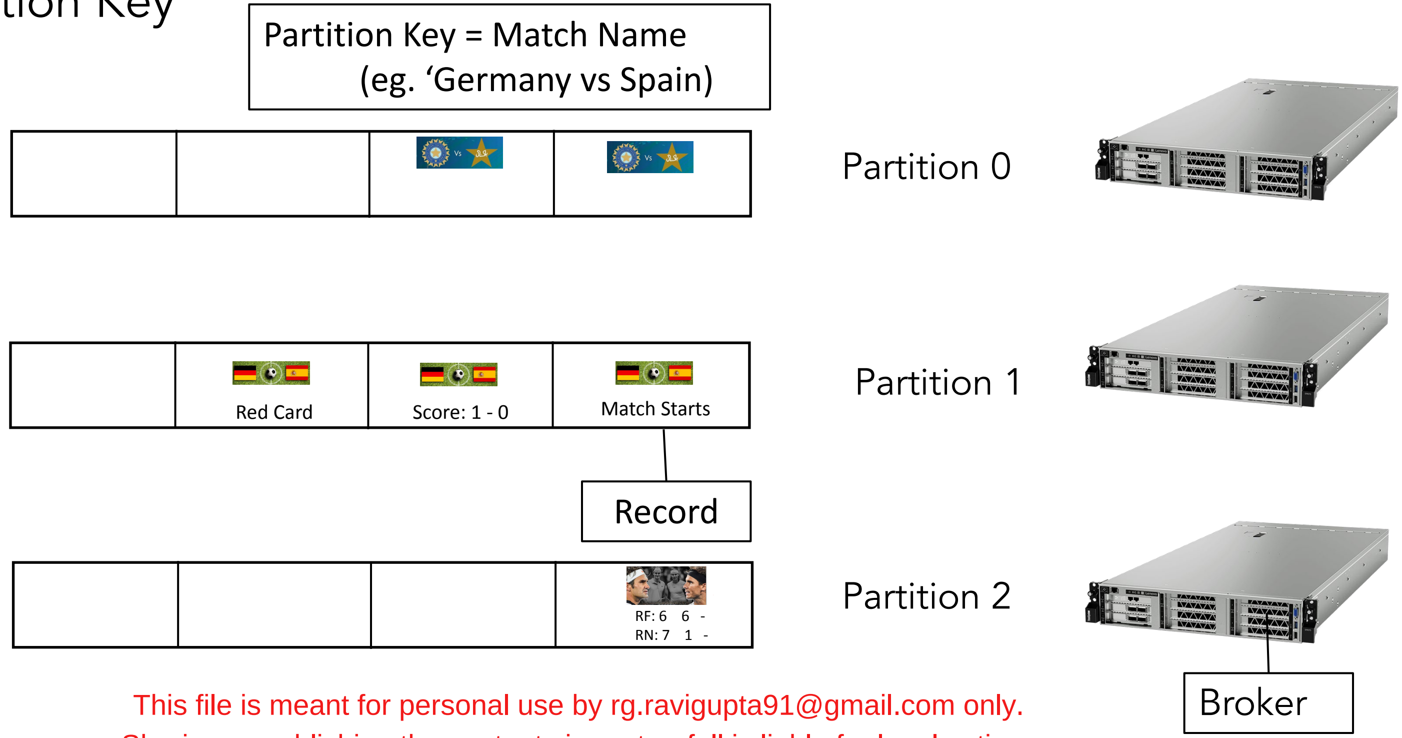
RF: 6 6 -
RN: 7 1 -

Partition 2

Broker

How Kafka Works

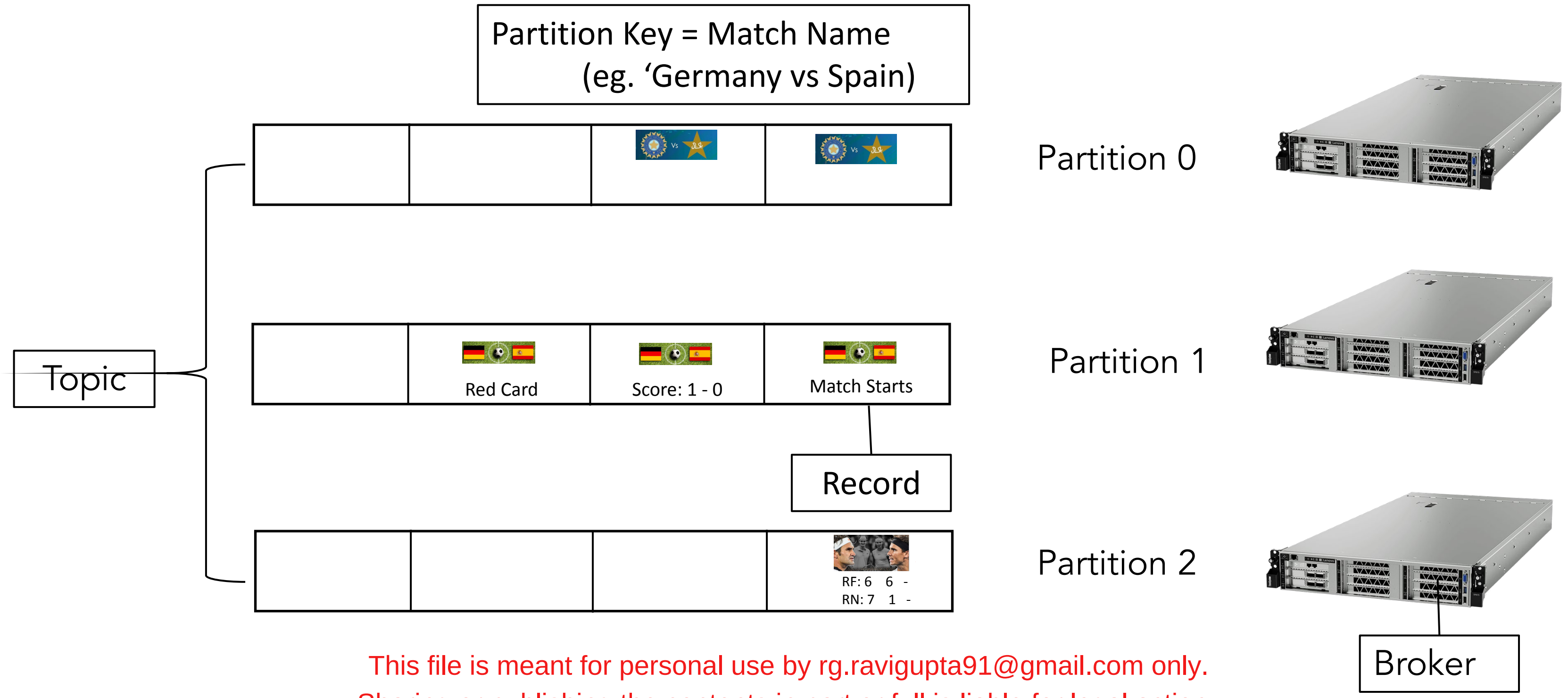
- The field used to decide which partition the record should be stored is called the Partition Key



If no key is specified Kafka simply assigns a random partition

How Kafka Works

- A grouping of partitions handling the same type of data is called a topic



Partition Key = Match Name
(eg. ‘Germany vs Spain)

Topic

Red Card

Score: 1 - 0

Match Starts

Record

RF: 6 6 -
RN: 7 1 -

Partition 0

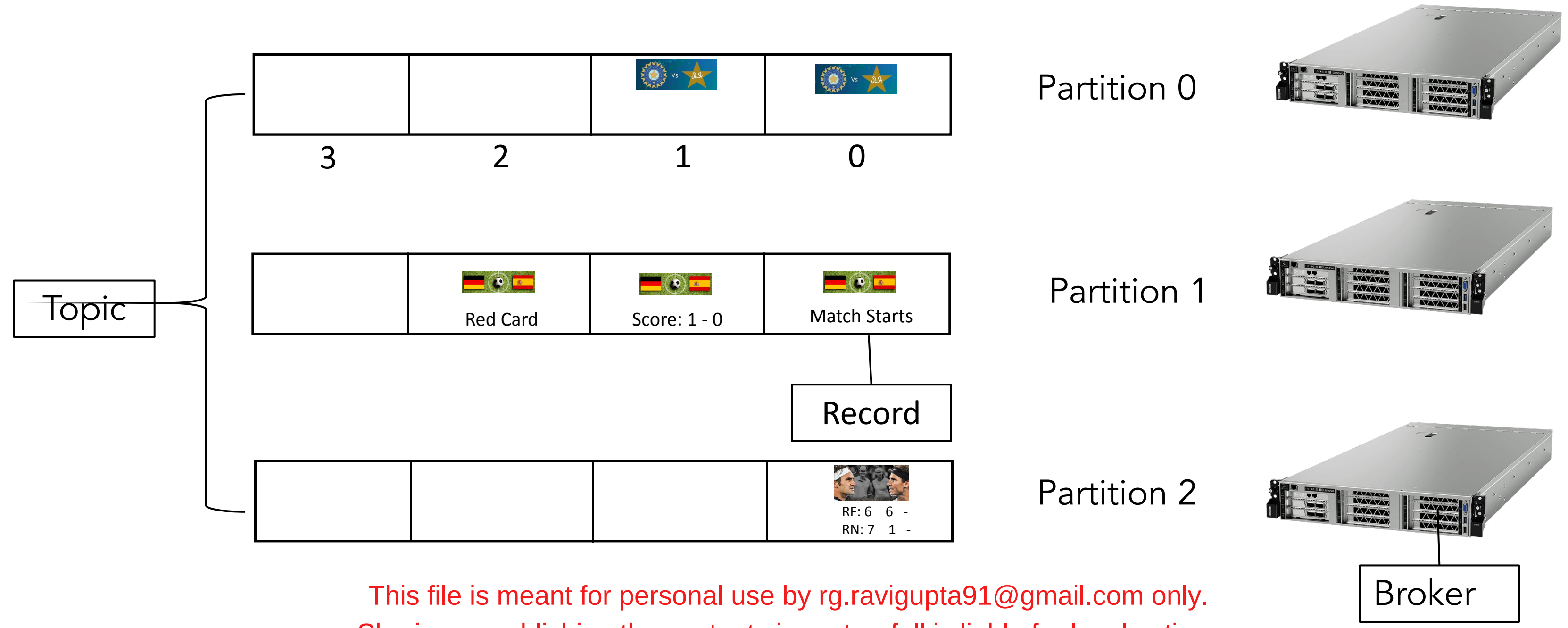
Partition 1

Partition 2

Broker

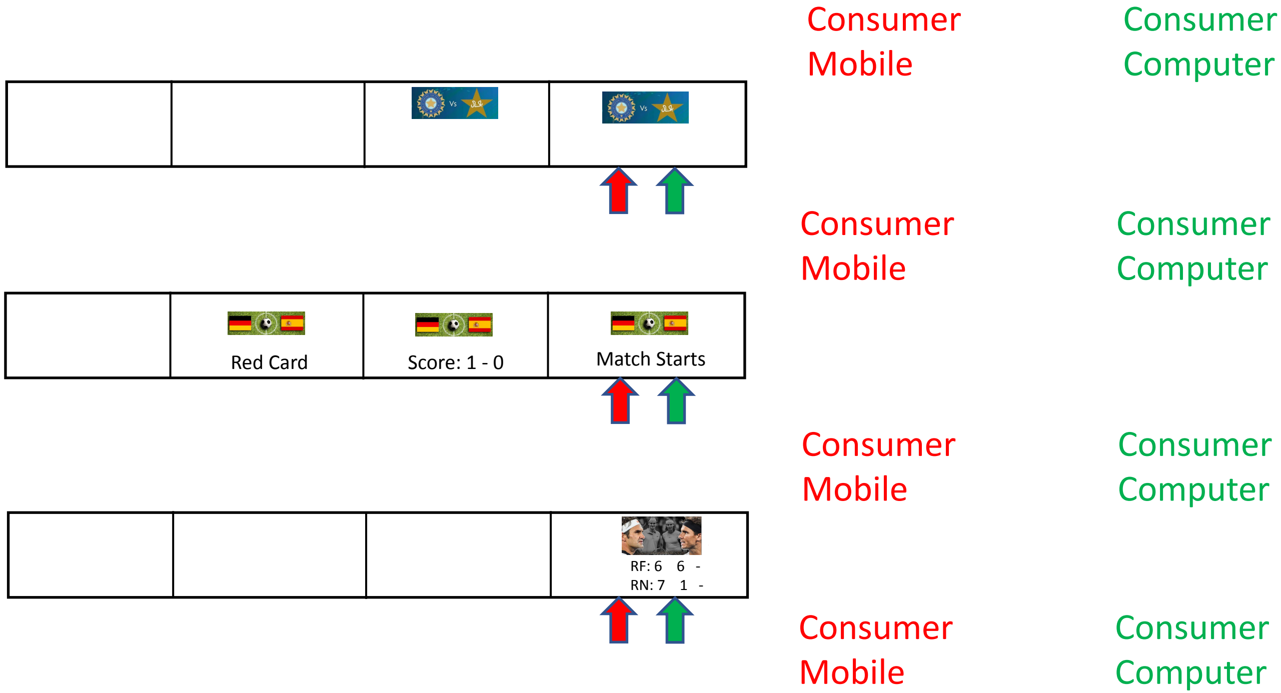
How Kafka Works

- In order to identify each record uniquely, Kafka provides a sequential number to each record which is called an offset



You can also parallelize the consumer applications having one consumer per partition guarantees ordering a game. Consumers can live on one machine or distributed among multiple ones.

One important concept in kafka is that the consumers are very light weight, we can create many of them without affecting performance. This is mainly because Kafka only needs to maintain the latest offset read by each consumer.



Consumer Mobile

Consumer Computer

Consumer Mobile

Consumer Computer

Red Card

Score: 1 - 0

Match Starts

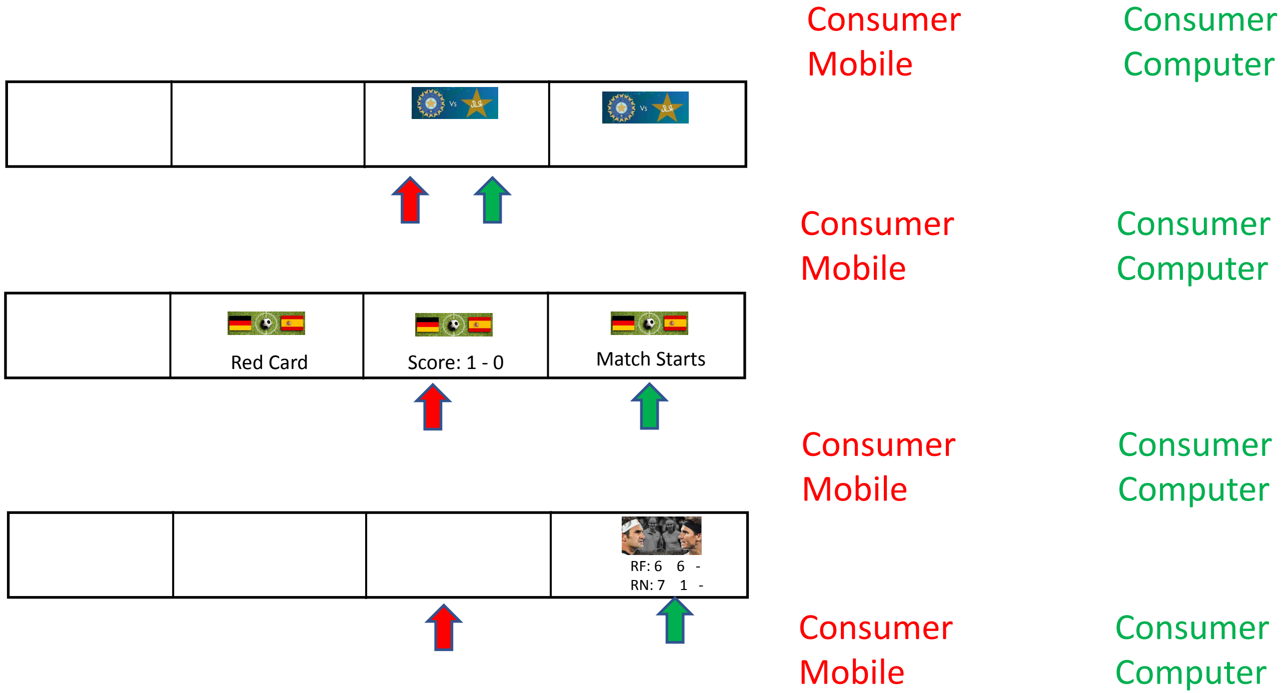
Consumer Mobile

Consumer Computer

RF: 6 6 -
RN: 7 1 -

Consumer Mobile

Consumer Computer



Consumer Mobile

Consumer Computer

Consumer Mobile

Consumer Computer

Red Card

Score: 1 - 0

Match Starts

Consumer Mobile

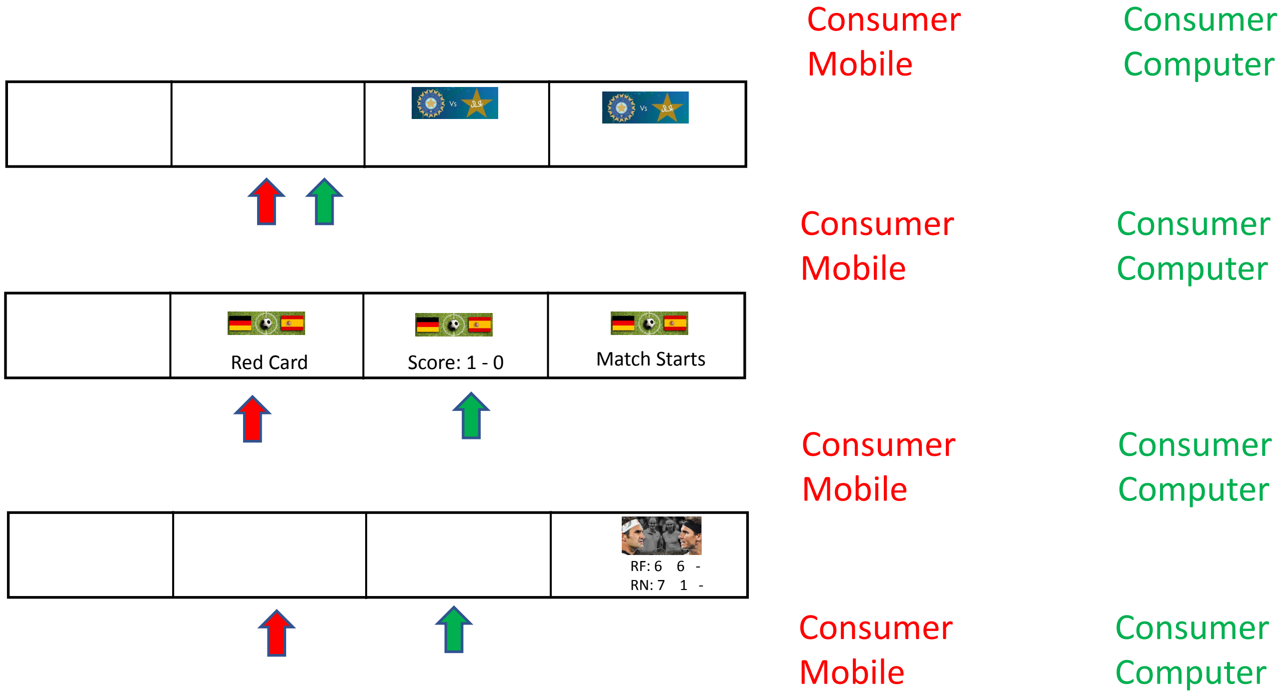
Consumer Computer

RF: 6 6 -

RN: 7 1 -

Consumer Mobile

Consumer Computer



Consumer Mobile

Consumer Computer

Consumer Mobile

Consumer Computer

	Red Card	Score: 1 - 0	Match Starts

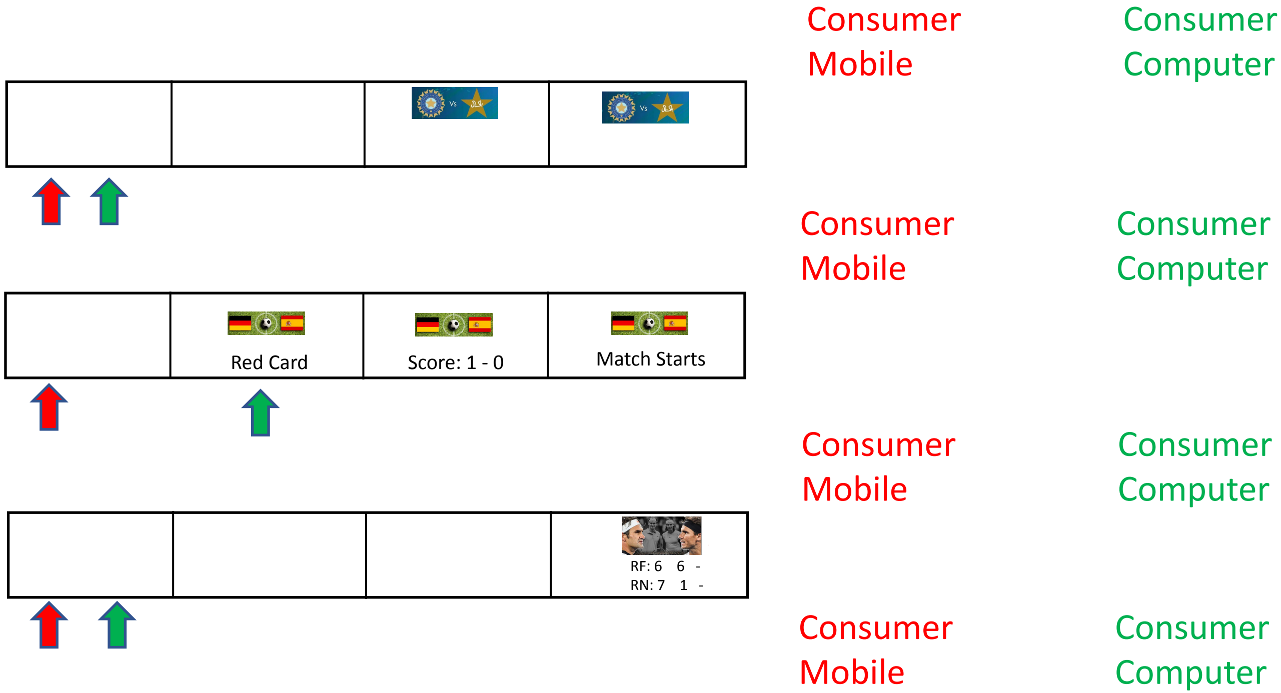
Consumer Mobile

Consumer Computer

			RF: 6 6 - RN: 7 1 -

Consumer Mobile

Consumer Computer



Consumer Mobile

Consumer Computer

Consumer Mobile

Consumer Computer

Red Card

Score: 1 - 0

Match Starts

Consumer Mobile

Consumer Computer

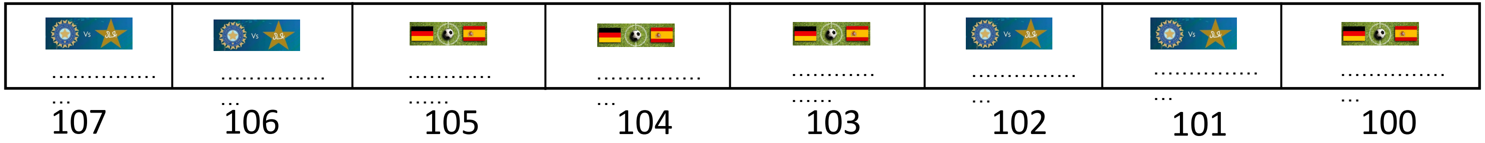
RF: 6 6 -
RN: 7 1 -

Consumer Mobile

Consumer Computer

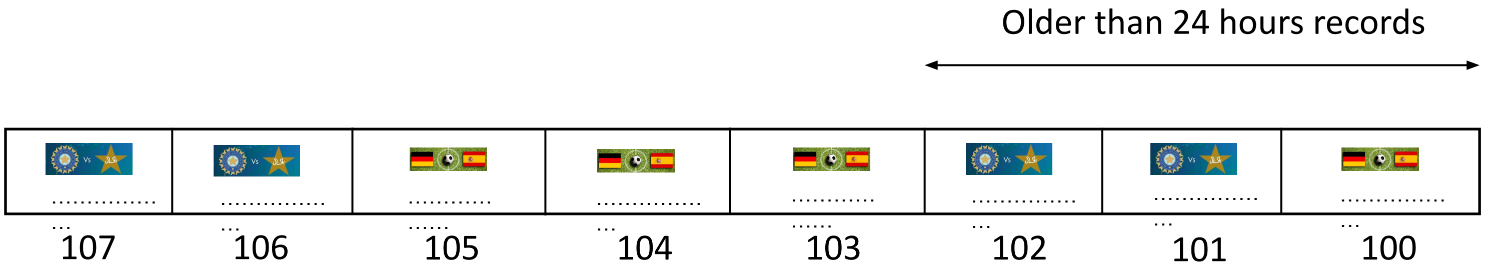
How Kafka Works

- How can kafka determine the record has been consumed and it can safely deleted so it can free up space?



How Kafka Works

- Kafka provide various policies that allow it to do a record cleanup for example, using a retention policy you can provide a record age limit, say, 24 hours

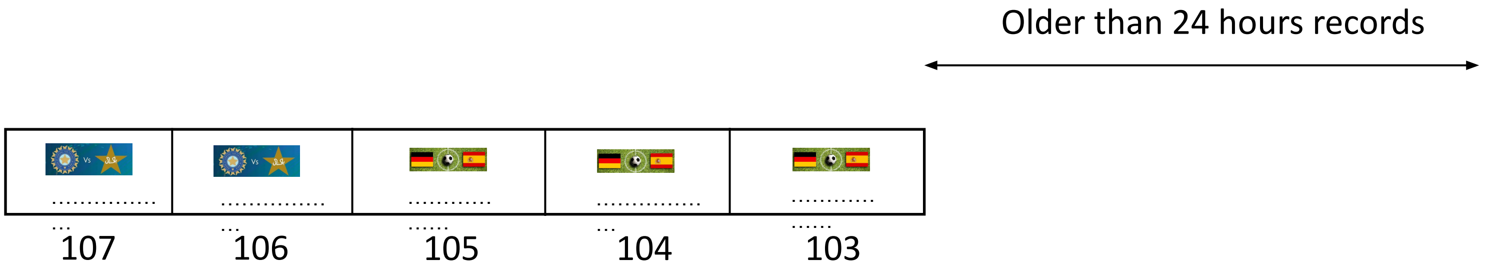


Older than 24 hours records

107 106 105 104 103 102 101 100

How Kafka Works

- If retention is configured for n days, then messages once published it is available for consumption for configured days and thereafter they are discarded

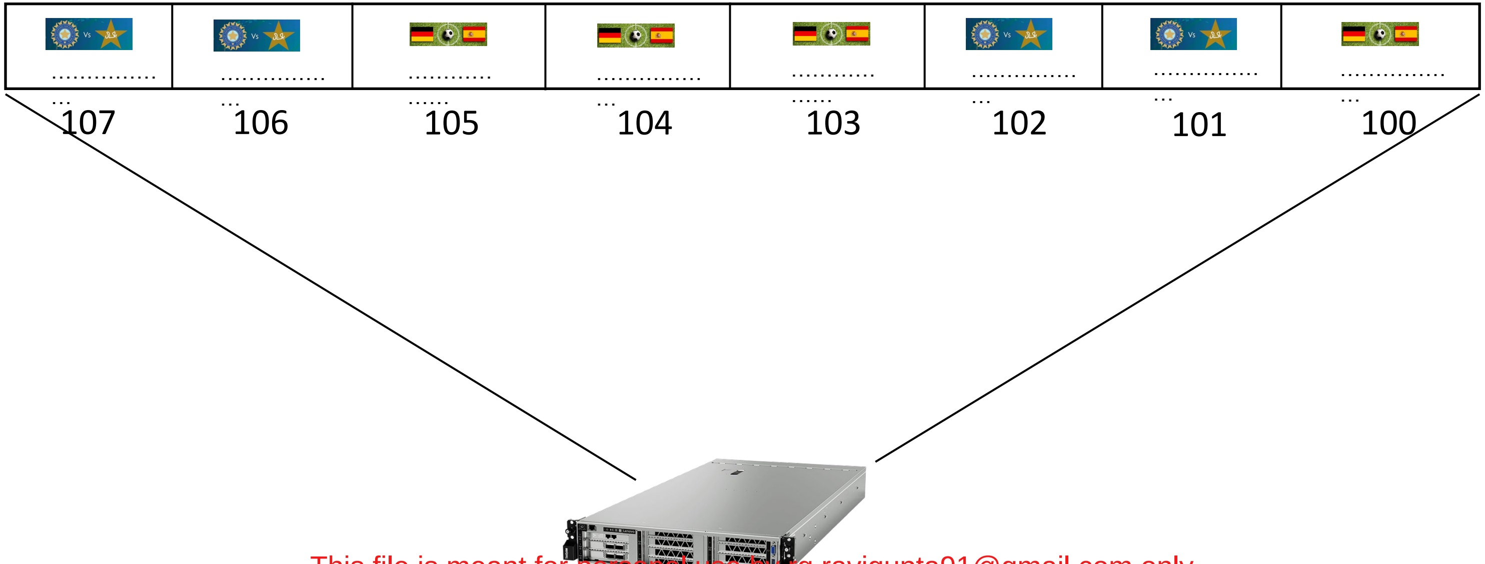


- After which the records are automatically deleted. Using this policy, your consumer is never down for more than this age limit - no messages are lost

Fault Tolerance and Durability

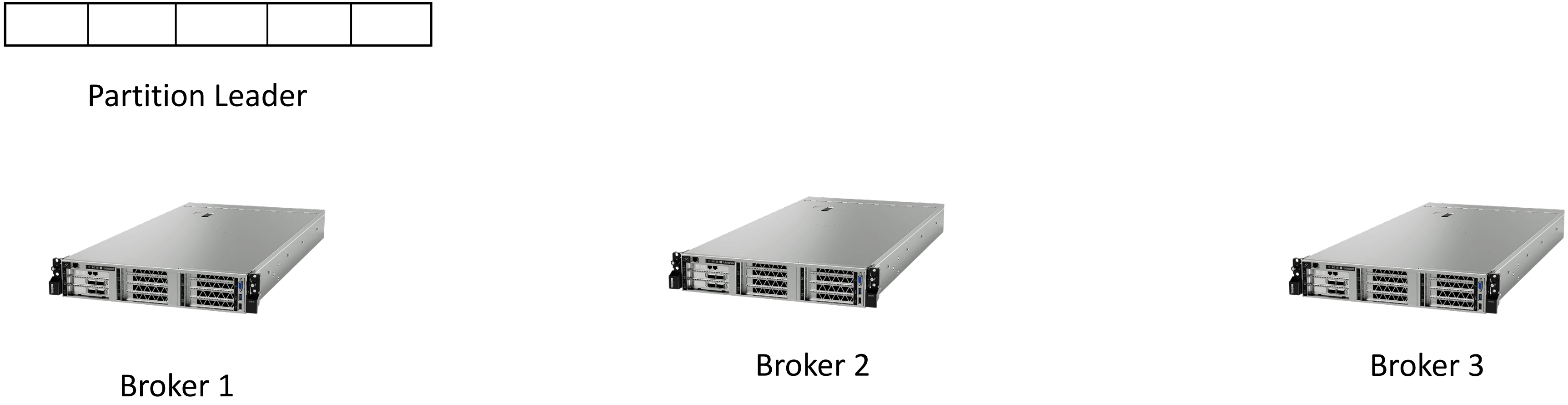
Fault Tolerance and Durability

- Each record is stored on a persistent storage so that if a broker goes down, it can recover when it comes back up



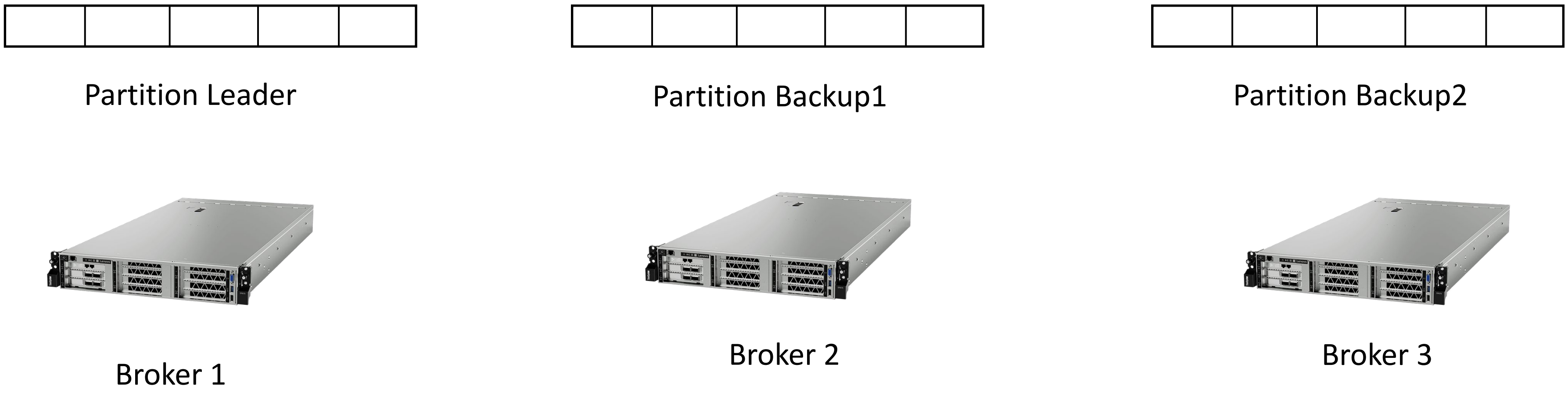
Fault Tolerance and Durability

- Kafka replicates partitions so that when a broker goes down a backup partition takes over and processing can resume



Fault Tolerance and Durability

- This replication is configured using a replication factor. For example, a replication factor of 3 leads to 3 copies of the partition



Partition Leader

Partition Backup1

Partition Backup2

Broker 1

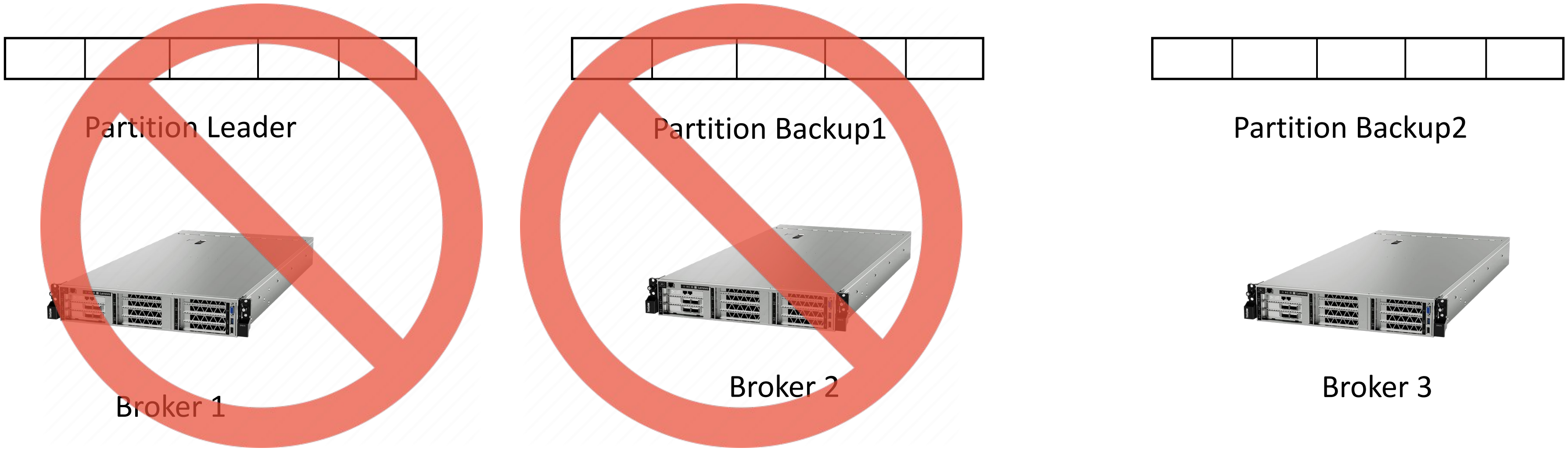
Broker 2

Broker 3

Replication Factor = 3

Fault Tolerance and Durability

- This means it can tolerate up to two broker servers going down at the same time



Replication Factor = 3

For a topic with replication factor N, Kafka can tolerate up to n-1 server failures without losing any messages.

Types of Kafka Cluster

- Types of Kafka Cluster:
  - Single Node – Single Broker Cluster:
  - Single Node – Multiple Broker Cluster
  - Multiple Nodes – Multiple Broker Cluster

- Kafka Cluster can run against the following broker model:
  - Single Broker Cluster
  - Multi Broker Cluster

Role of Zookeeper

Role of Zookeeper

- Apache Zookeeper is an open source project that provides a centralized infrastructure that enables synchronization across cluster

- It is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services

- All of these kinds of services are used in some form or another by distributed applications

Role of Zookeeper

- Each Kafka broker coordinates with other Kafka brokers using Zookeeper

- Producers and consumers are notified by Zookeeper service about the presence of new broker in Kafka system or failure of the broker in Kafka system

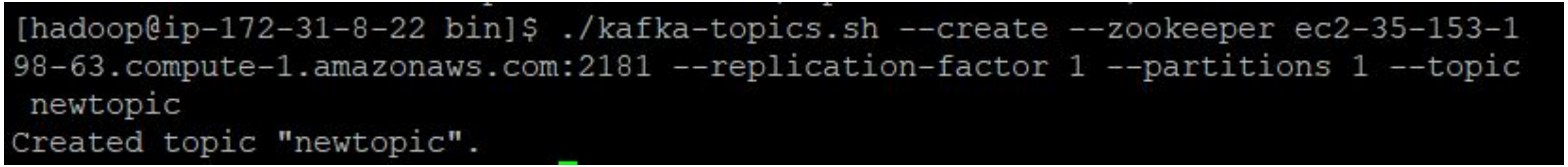
- Zookeeper is mainly used to track status of nodes present in Kafka cluster and also to keep track of Kafka topics, messages, etc

Kafka Commands

Create Kafka Topic

- Kafka command to create topic

```
kafka-topics --create zookeeper ip-address:2181
--replication-factor n --partition n --topic topic_name
```

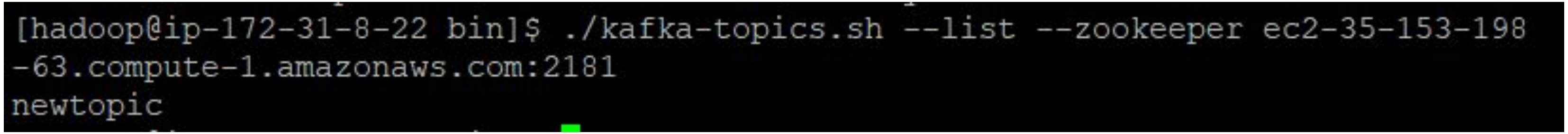


```
[hadoop@ip-172-31-8-22 bin]$ ./kafka-topics.sh --create --zookeeper ec2-35-153-1
98-63.compute-1.amazonaws.com:2181 --replication-factor 1 --partitions 1 --topic
 newtopic
Created topic "newtopic".
```

List Kafka Topics

- Kafka command to list created topics

```
kafka-topics --list --zookeeper ip-address:2181
```

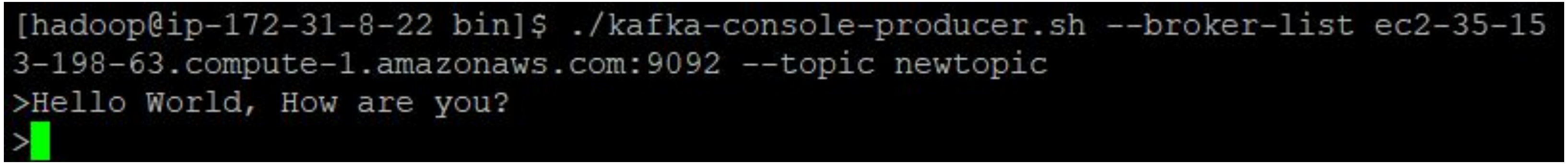


```
[hadoop@ip-172-31-8-22 bin]$ ./kafka-topics.sh --list --zookeeper ec2-35-153-198
-63.compute-1.amazonaws.com:2181
newtopic
```

Producer

- Kafka command to create a producer and publishing to topic

```
Kafka-console-producer --broker-list ip-address:9092 --topic
topic_name
```

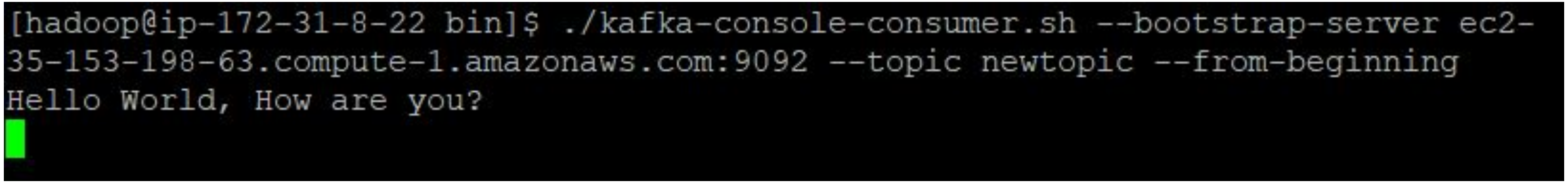


```
[hadoop@ip-172-31-8-22 bin]$ ./kafka-console-producer.sh --broker-list ec2-35-15
3-198-63.compute-1.amazonaws.com:9092 --topic newtopic
>Hello World, How are you?
>
```

Consumer

- Kafka command to create a consumer and subscribing to a topic

```
Kafka-console-consumer --bootstrap-server ip-address:9092
--topic topic_name --from-beginning
```



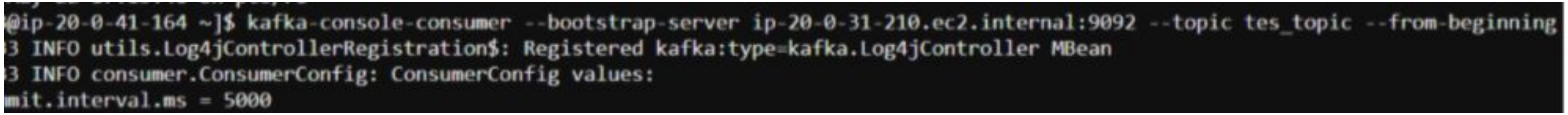
```
[hadoop@ip-172-31-8-22 bin]$ ./kafka-console-consumer.sh --bootstrap-server ec2-
35-153-198-63.compute-1.amazonaws.com:9092 --topic newtopic --from-beginning
Hello World, How are you?
█
```

If no key is specified Kafka simply assigns a random partition

For Multi Broker Kafka Cluster

- Kafka command to create a consumer and subscribing to a topic

```
Kafka-console-consumer --bootstrap-server ip-address:9092
--topic topic_name --from-beginning
```



```
@ip-20-0-41-164 ~]$ kafka-console-consumer --bootstrap-server ip-20-0-31-210.ec2.internal:9092 --topic tes_topic --from-beginning
3 INFO utils.Log4jControllerRegistration$: Registered kafka:type=kafka.Log4jController MBean
3 INFO consumer.ConsumerConfig: ConsumerConfig values:
mit.interval.ms = 5000
```

Summary

- Real-time data is information that is delivered immediately after collection
- Messaging systems provides seamless integration among various distributed endpoints with the help of messages in real-time
- Kafka is a distributed streaming platform that is used to publish and subscribe to streams of records
- Kafka provide various policies that allow it to do a record cleanup using a retention policy
- Kafka is a durable and fault-tolerant system
- Each Kafka broker coordinates with other Kafka brokers using Zookeeper

Thank You