Apache Kafka

Limitations Hadoop/HDFS

- High throughput but high latency: It can take a long time to process the data
- No real-time data processing
- Multiple readers but *single writer*

Introducing Apache Kafka

- Distributed event streaming platform
- Based on the publisher/consumer pattern

Applications:

- *Activity tracking*: Register user activities on a website. These activities are then used by other applications to generate reports, feed machine learning models, update search results, ...
- *Messaging*: Send emails asynchronously.
- *Metrics/logging*: Applications publish metrics later consumed by a monitoring system.
- *Commit log*: Publish database changes on Kafka. Then, it can be saved in a database, replicated, processed for other applications, ...
- *Stream processing*: Next lecture

Introducing Apache Kafka - Pros/Cons

Kafka:

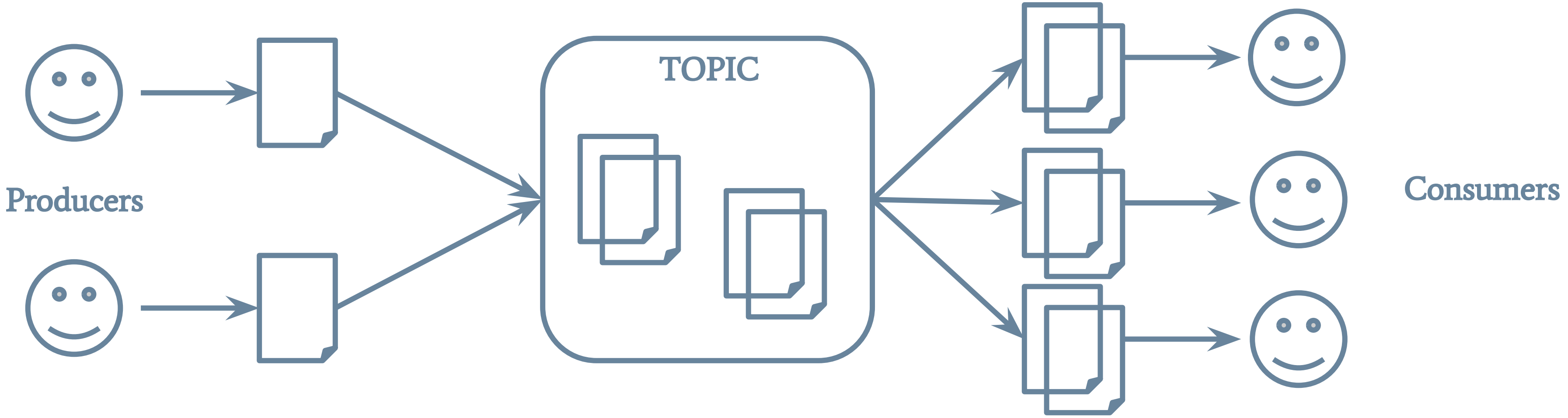
- Is *scalable*
- Has *low latency and high throughput*: Messages processed quickly
- Is *fault tolerant*
- Allows *real-time data pipeline*

But:

- *Fixed format messages*: Hard to modify the messages
- *Retention is expensive*: We have to use another system for long-term storage (HDFS)
- Readers must *read entire messages*: They cannot get a specific field.

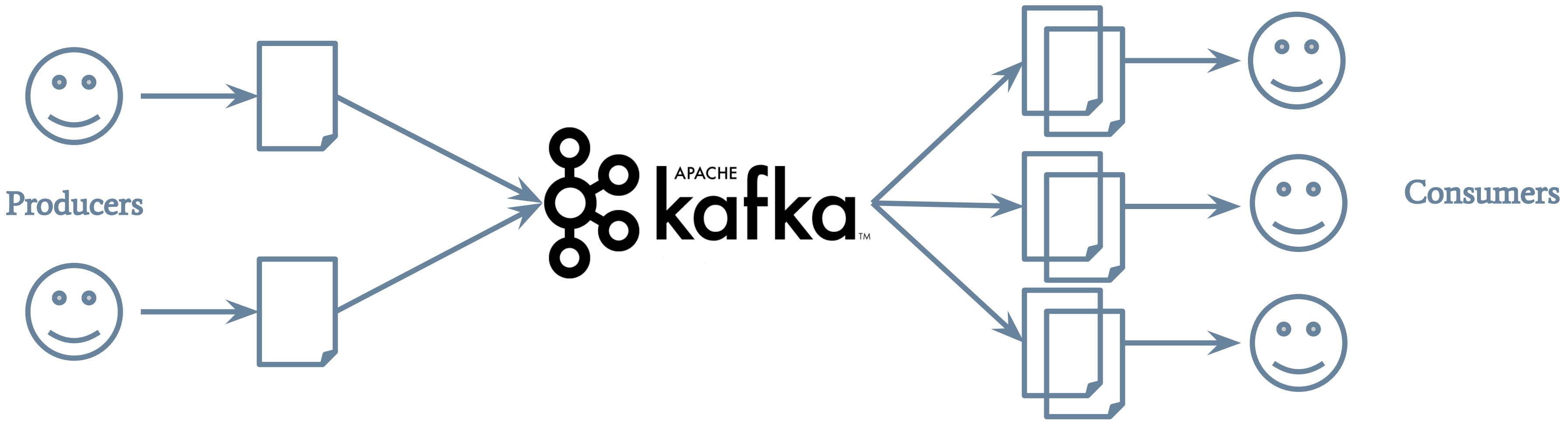
Producer/Consumer Pattern

- Also known as publish/subscribe
- Two kinds of actors:
  - The publishers/producers: They can write messages on a topic
  - The subscribers/consumers: They are notified when there are new messages and they can read them



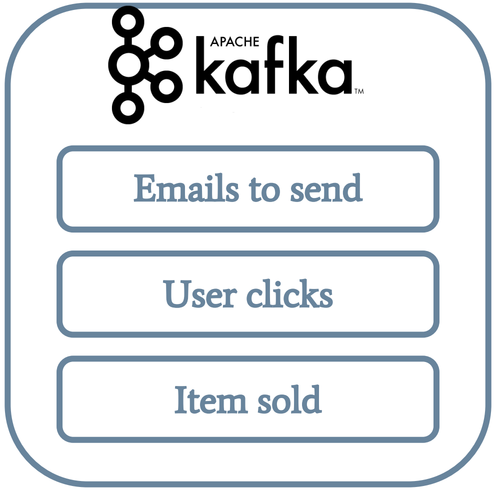
Producer/Consumer Pattern

- Also known as publish/subscribe
- Two kinds of actors:
  - The publishers/producers: They can write messages on a topic
  - The subscribers/consumers: They are notified when there are new messages and they can read them



Kafka Topic

- A topic is a category of messages to which producers can write and consumers subscribe



Kafka Topic - Operations

- Creation

```
$ bin/kafka-topics.sh --create --topic my-topic --bootstrap-server KAFKA_ADDRESS
```

- Describe

```
$ bin/kafka-topics.sh --describe --topic my-topic --bootstrap-server
KAFKA_ADDRESS
```

- Delete

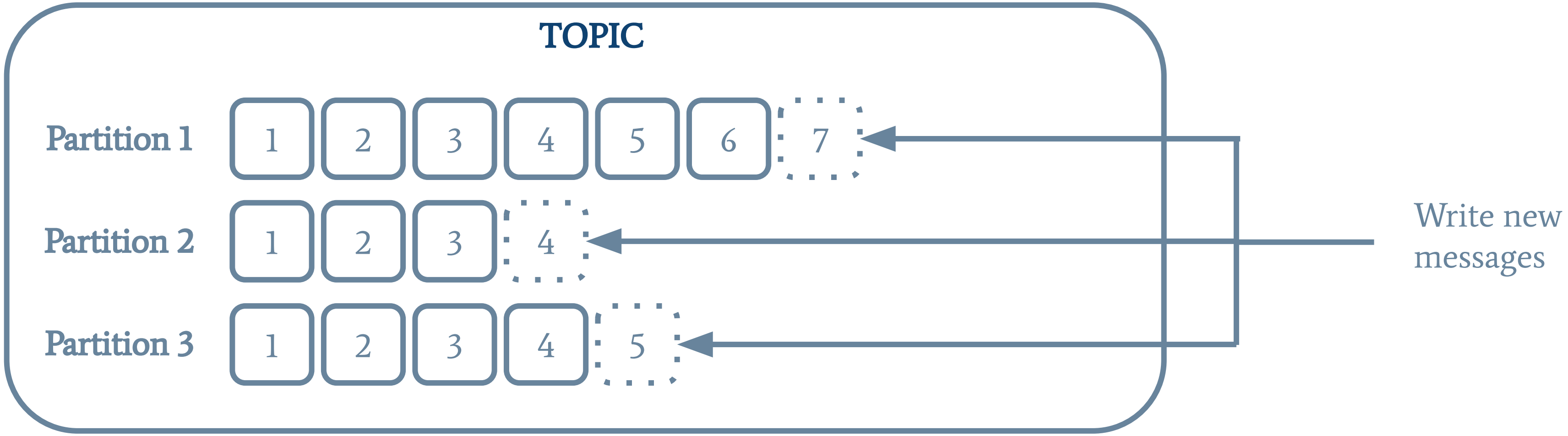
```
$ bin/kafka-topics.sh --delete --topicmy-topic --bootstrap-server
KAFKA_ADDRESS
```

- List

```
$ bin/kafka-topics.sh --list --bootstrap-server KAFKA_ADDRESS
```

Partitioning

- Messages are stored inside a topic into append-only partitions
- Helps with grouped message processing, replication, fault tolerance, parallel writing, throughput, …
- The partition of a message can be chosen automatically by Kafka or using a given function based on a key



Partitioning - Operations

- Change the number of partitions (while application is running)

```
$ bin/kafka-topics.sh --topic my-topic --bootstrap-server
KAFKA_ADDRESS --alter --partitions NEW_NUMBER_PARTITIONS
```

Producers

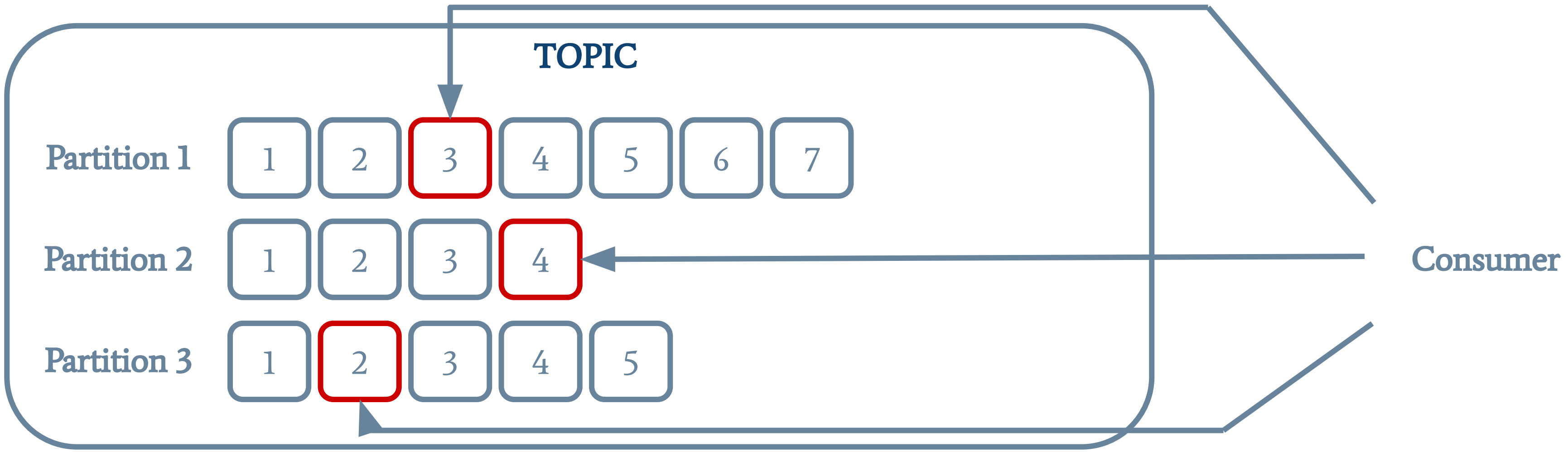
- Producers write “messages” to a topic

What are messages?

- Any byte array
- In general, in a key/value format
- We try to structure the value using a simple format (JSON, CSV, XML) or a more advanced library
- The data needs to be transferred and therefore serialized/deserialized
- Messages are indexed using a counter called the *offset* for each partition

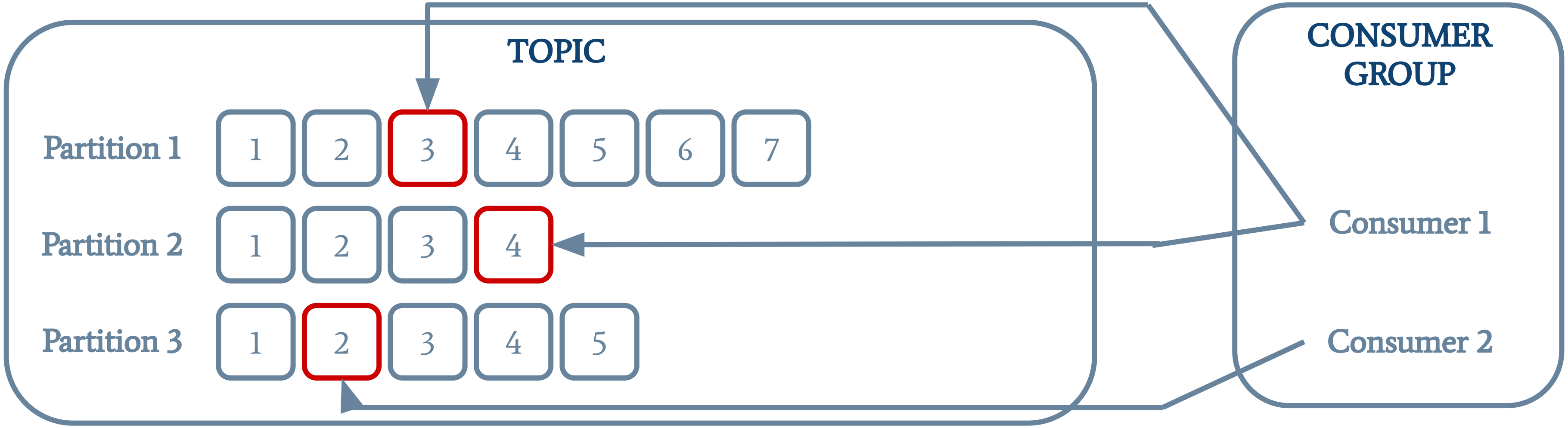
Consumers

- Consumers subscribe to a given topic and read messages
- A consumer keeps tracks of its current offset for each partition. It can continue the processing later if it crashes.



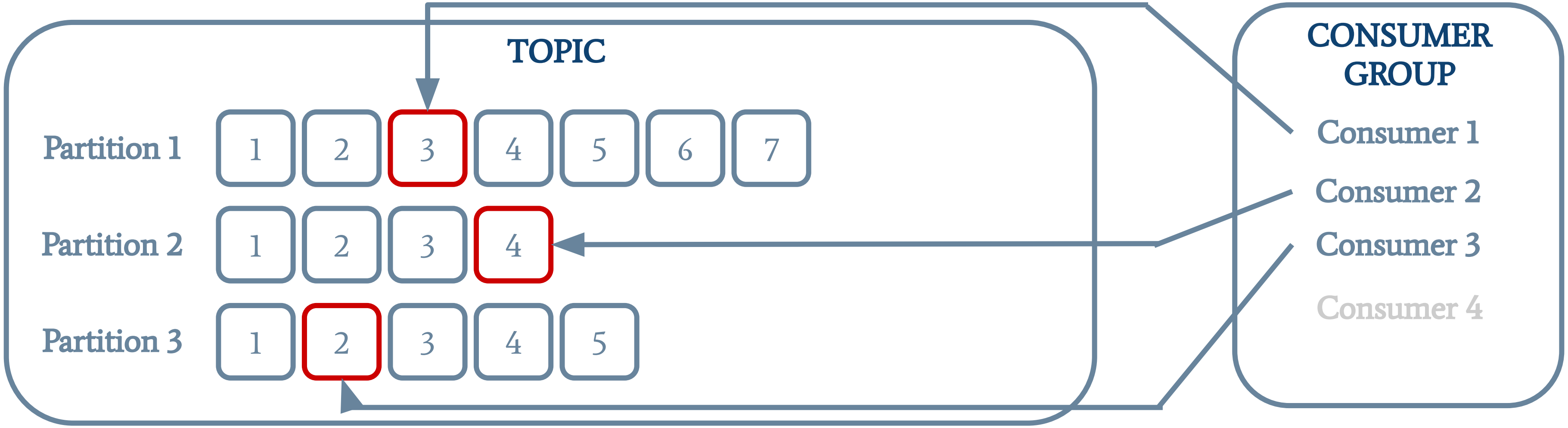
Consumer Groups

- A group of consumers is composed of several consumers that share the same offsets
- A partition is always assigned to *only one* consumer
- If there are more consumers than partitions, the additional consumers wait



Consumer Groups

- A group of consumers is composed of several consumers that share the same offsets
- A partition is always assigned to *only one* consumer
- If there are more consumers than partitions, the additional consumers wait



Consumer Groups - Operations

- List

```
$ bin/kafka-consumer-groups.sh --bootstrap-server
KAFKA_ADDRESS --list
```

- Describe

```
$ bin/kafka-consumer-groups.sh --bootstrap-server
KAFKA_ADDRESS --describe --group MyGroup
```

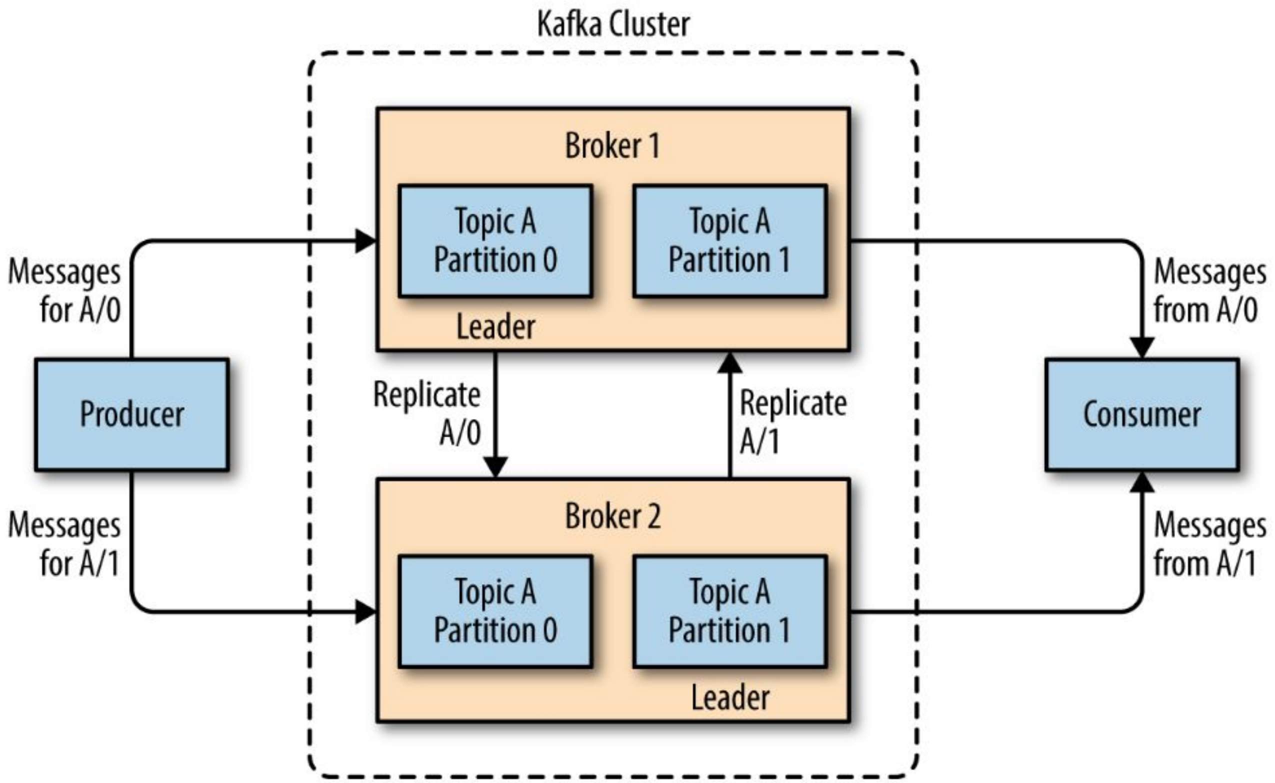
- Delete

```
$ bin/kafka-consumer-groups.sh --bootstrap-server
KAFKA_ADDRESS --describe --group MyGroup
```

Brokers

- A single Kafka server is called a *broker*
- A broker is in charge of answering requests from the producers and consumers and maintain its partitions
- A cluster is composed of several brokers
    - One of them is the controller responsible for administrative tasks like partition assignment
    - A partition is owned by only one broker (the leader) but can be replicated on several brokers

Brokers



From Kafka - The definitive guide

Let's go to the lab