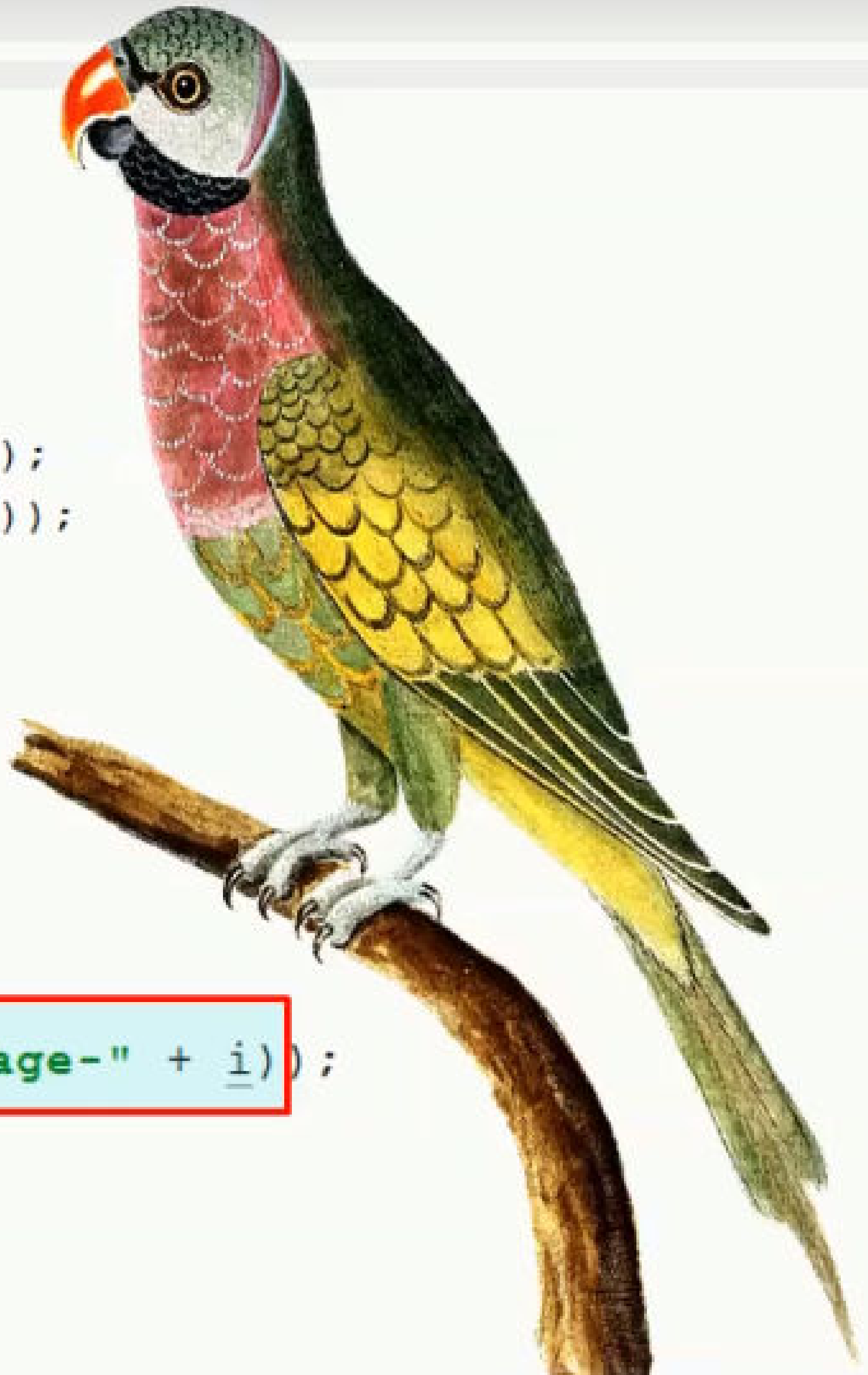
Kafka Producer API - Internals

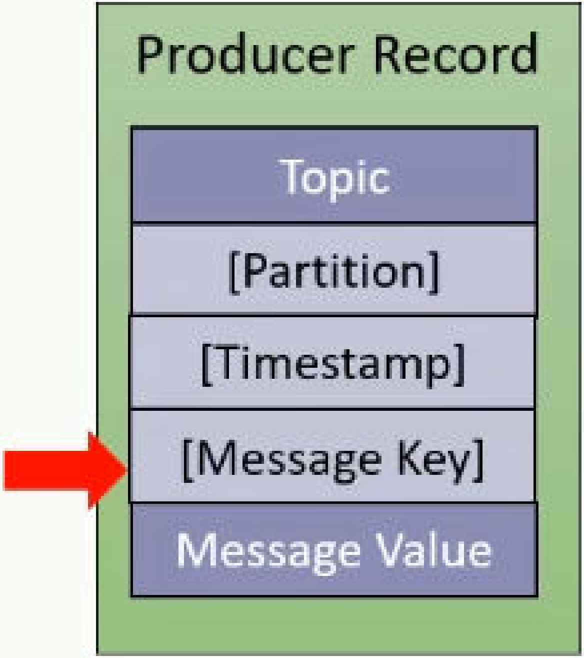
```
Properties props = new Properties();
props.put(ProducerConfig.CLIENT_ID_CONFIG, AppConfigs.applicationID);
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, AppConfigs.bootstrapServers);
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, IntegerSerializer.class.getName());
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());


KafkaProducer<Integer, String> producer = new KafkaProducer<>(props);


producer.send(new ProducerRecord<>(AppConfigs.topicName, i,  value: "Simple Message-" + i));


producer.close();
```





Producer Record

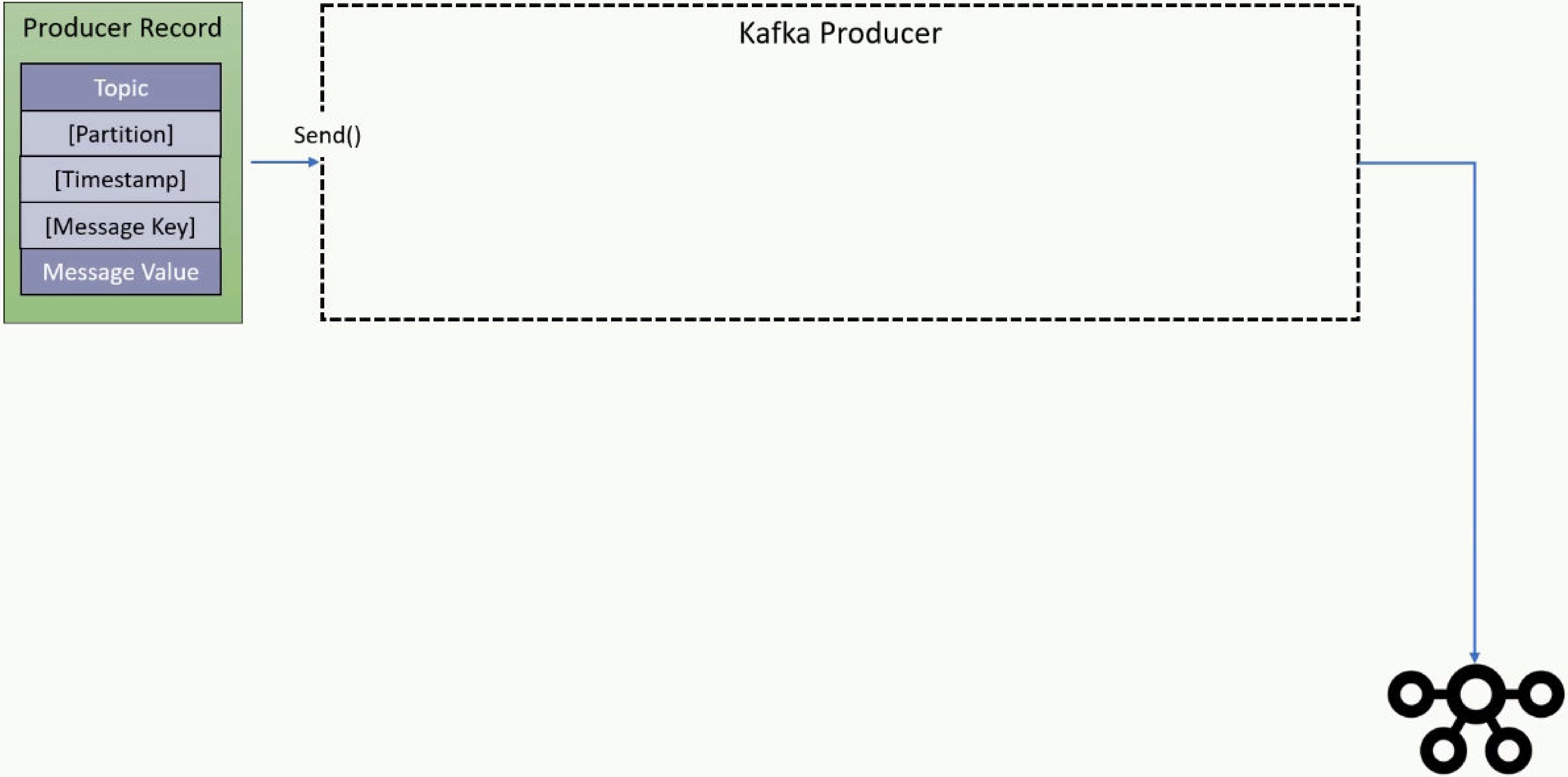
Topic

[Partition]

[Timestamp]

[Message Key]

Message Value



Producer Record

Topic

[Partition]

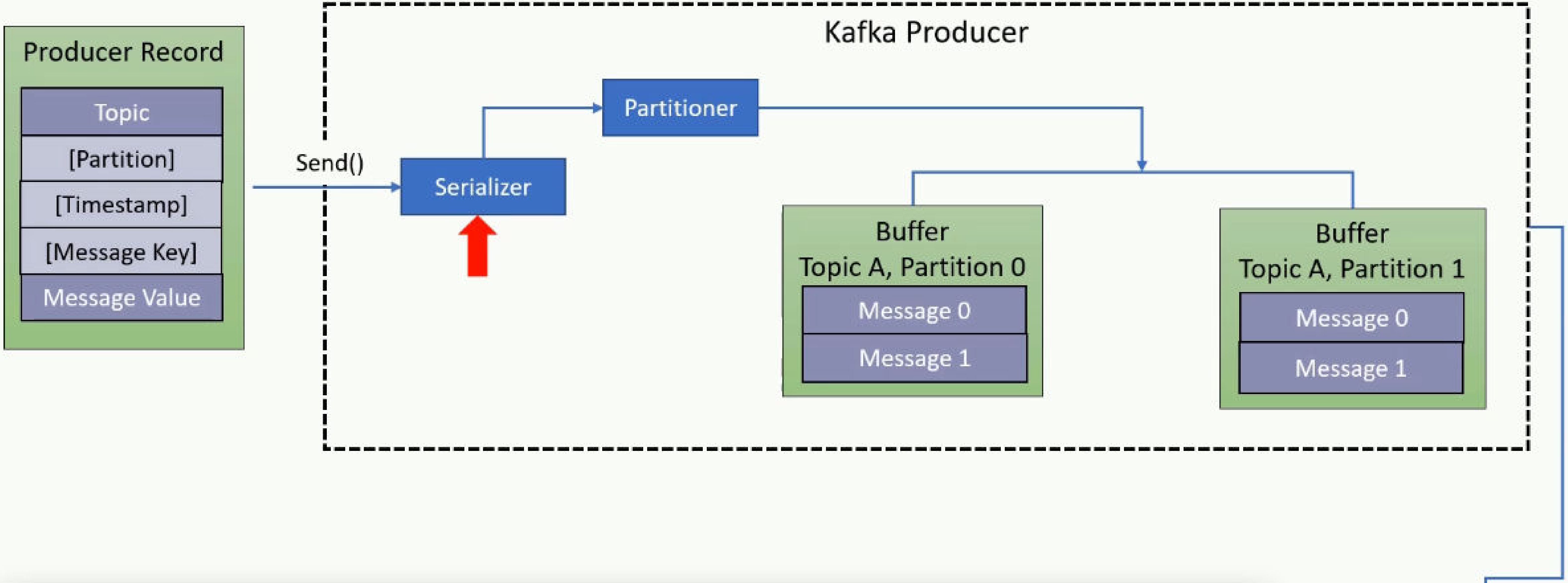
[Timestamp]

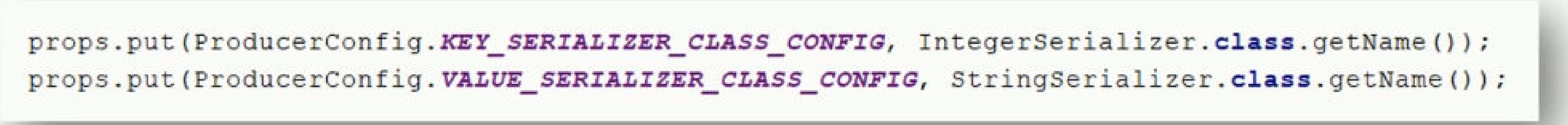
[Message Key]

Message Value

Send()

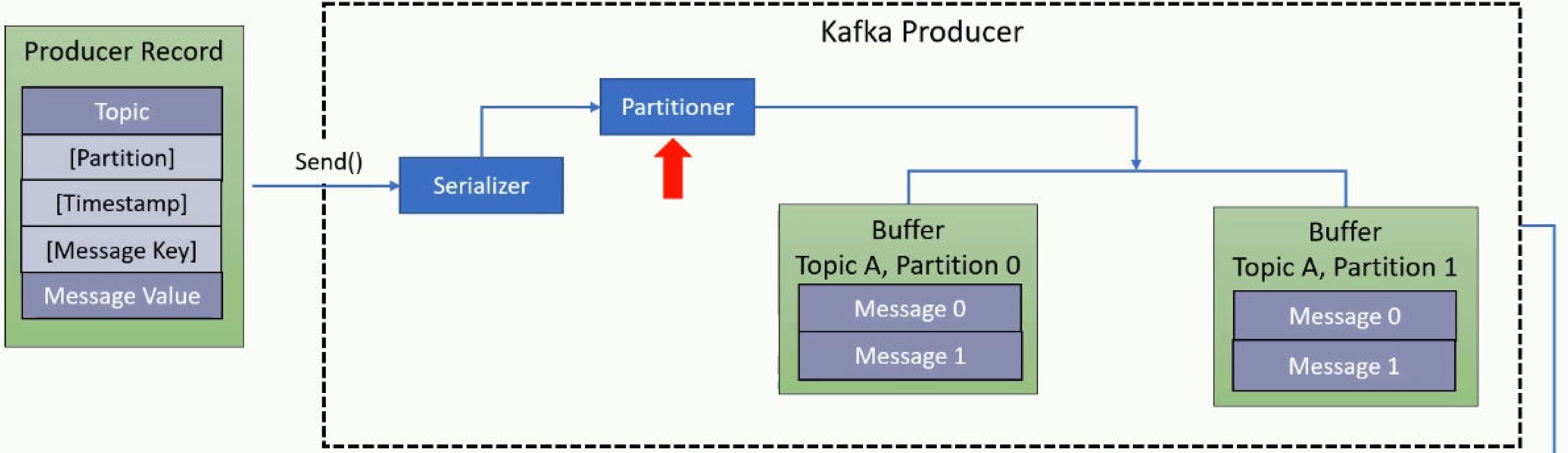
Kafka Producer

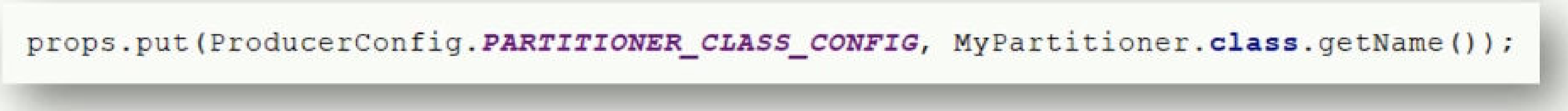




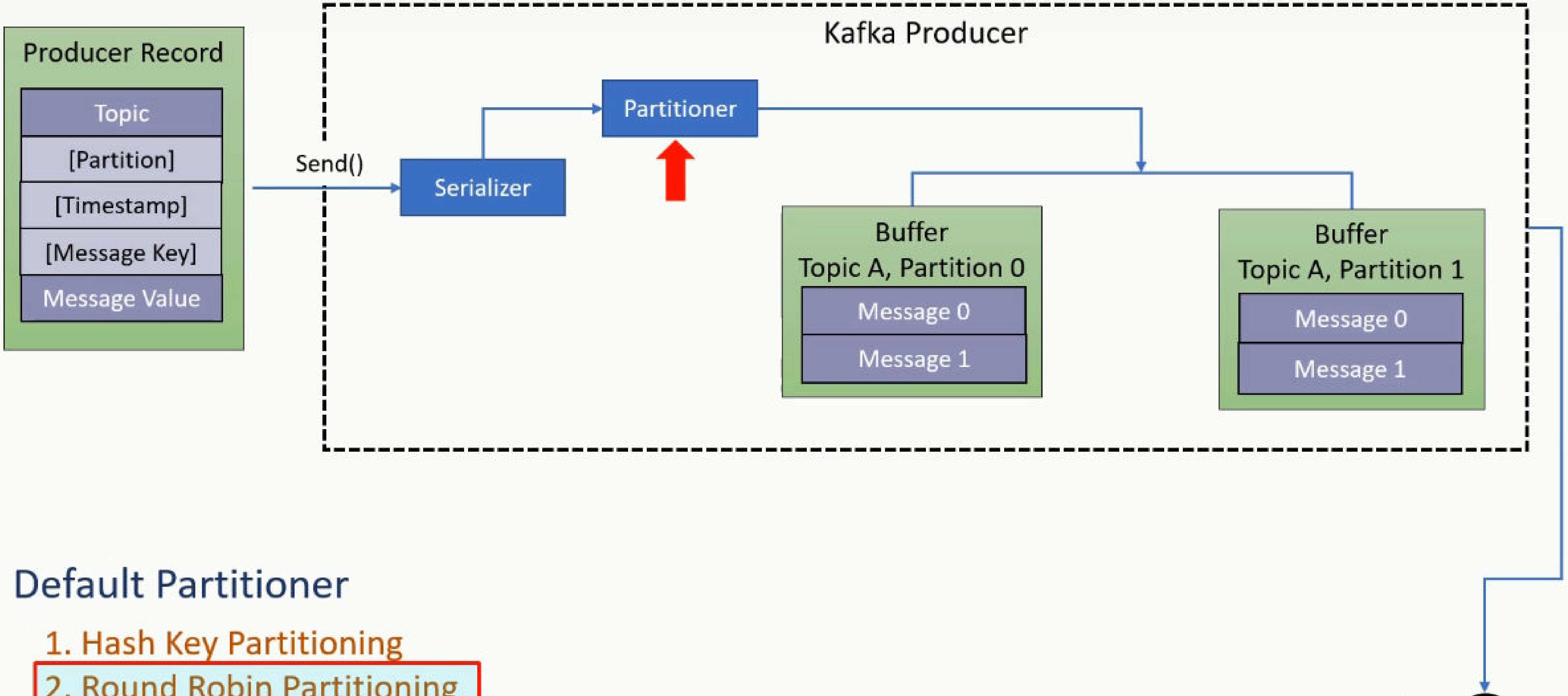
```
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, IntegerSerializer.class.getName());
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());
```

Kafka Producer API





```
props.put(ProducerConfig.PARTITIONER_CLASS_CONFIG, MyPartitioner.class.getName());
```



Producer Record

Topic

[Partition]

[Timestamp]

[Message Key]

Message Value

Send()

Kafka Producer

Serializer

Partitioner

Buffer
Topic A, Partition 0

Message 0

Message 1

Buffer
Topic A, Partition 1

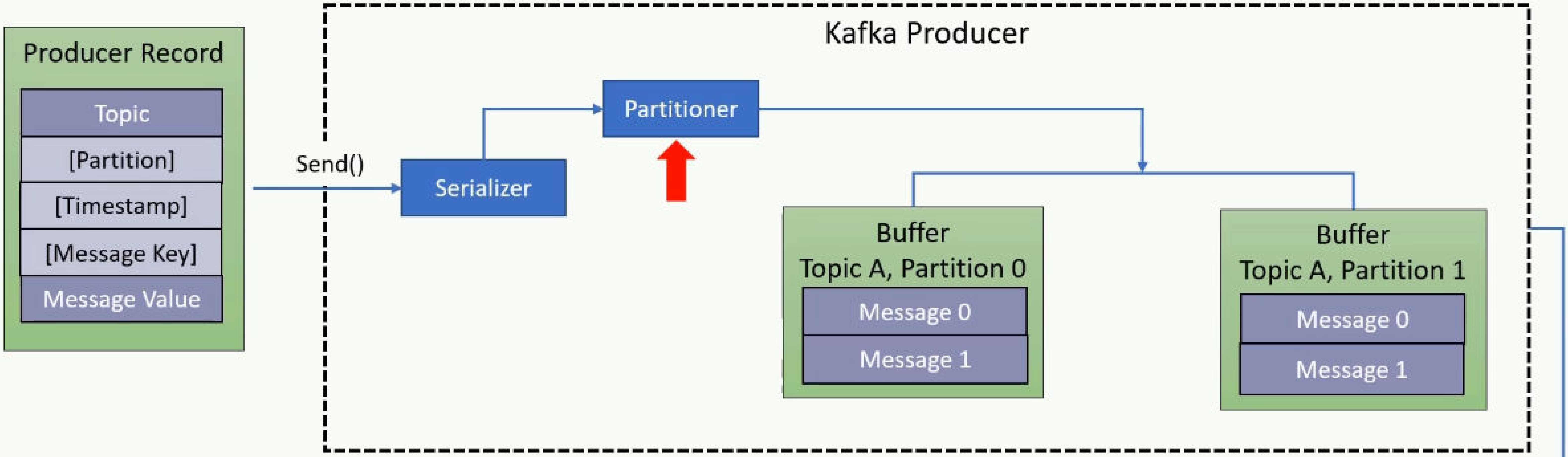
Message 0

Message 1

Default Partitioner

1. Hash Key Partitioning
2. Round Robin Partitioning

Kafka Producer API

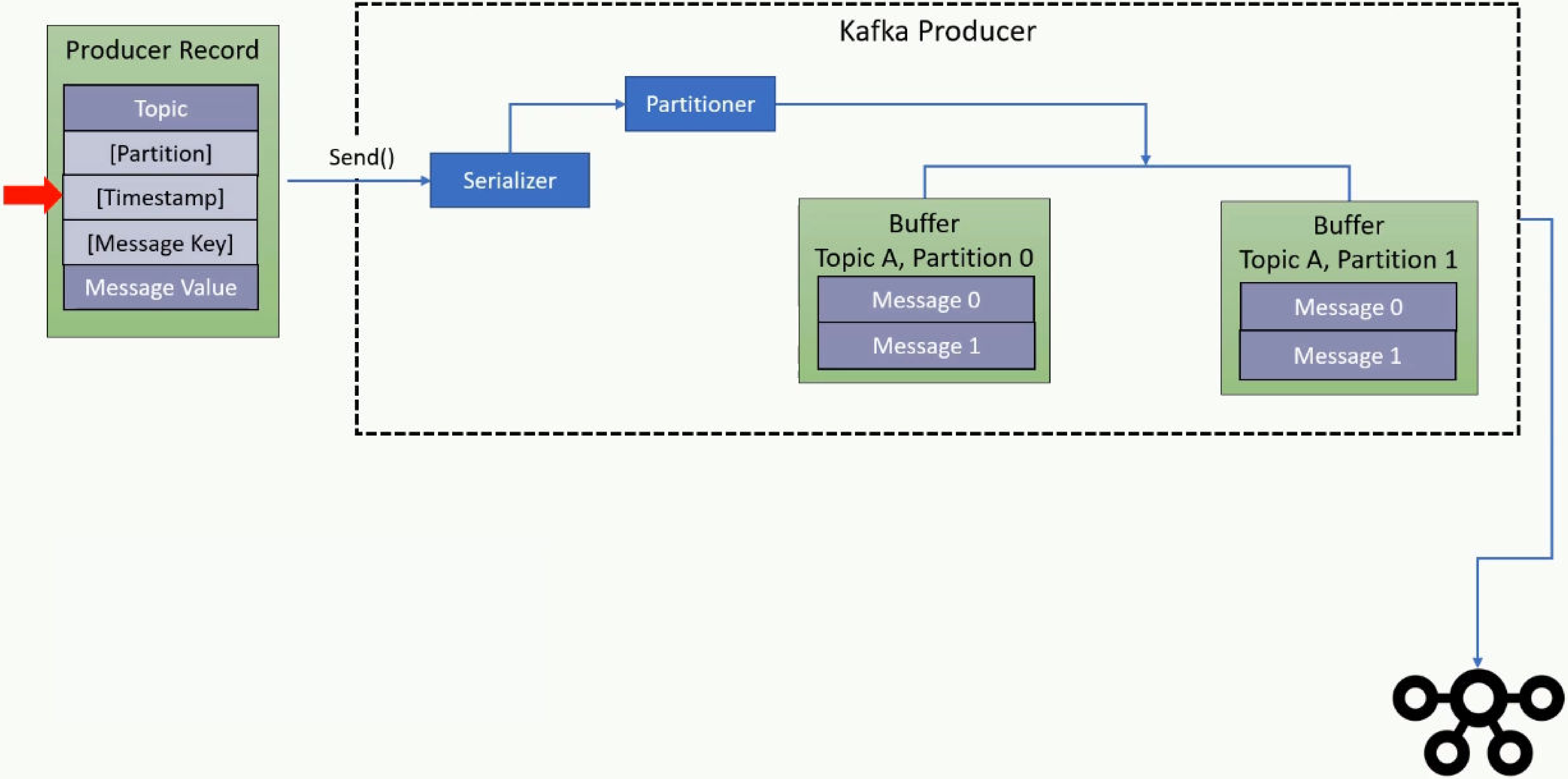


Default Partitioner

1. Hash Key Partitioning
2. Round Robin Partitioning

MyKey → hash(key) % #partitions → 3

Kafka Producer API



Producer Record

Topic

[Partition]

[Timestamp]

[Message Key]

Message Value

Send()

Kafka Producer

Serializer

Partitioner

Buffer
Topic A, Partition 0

Message 0

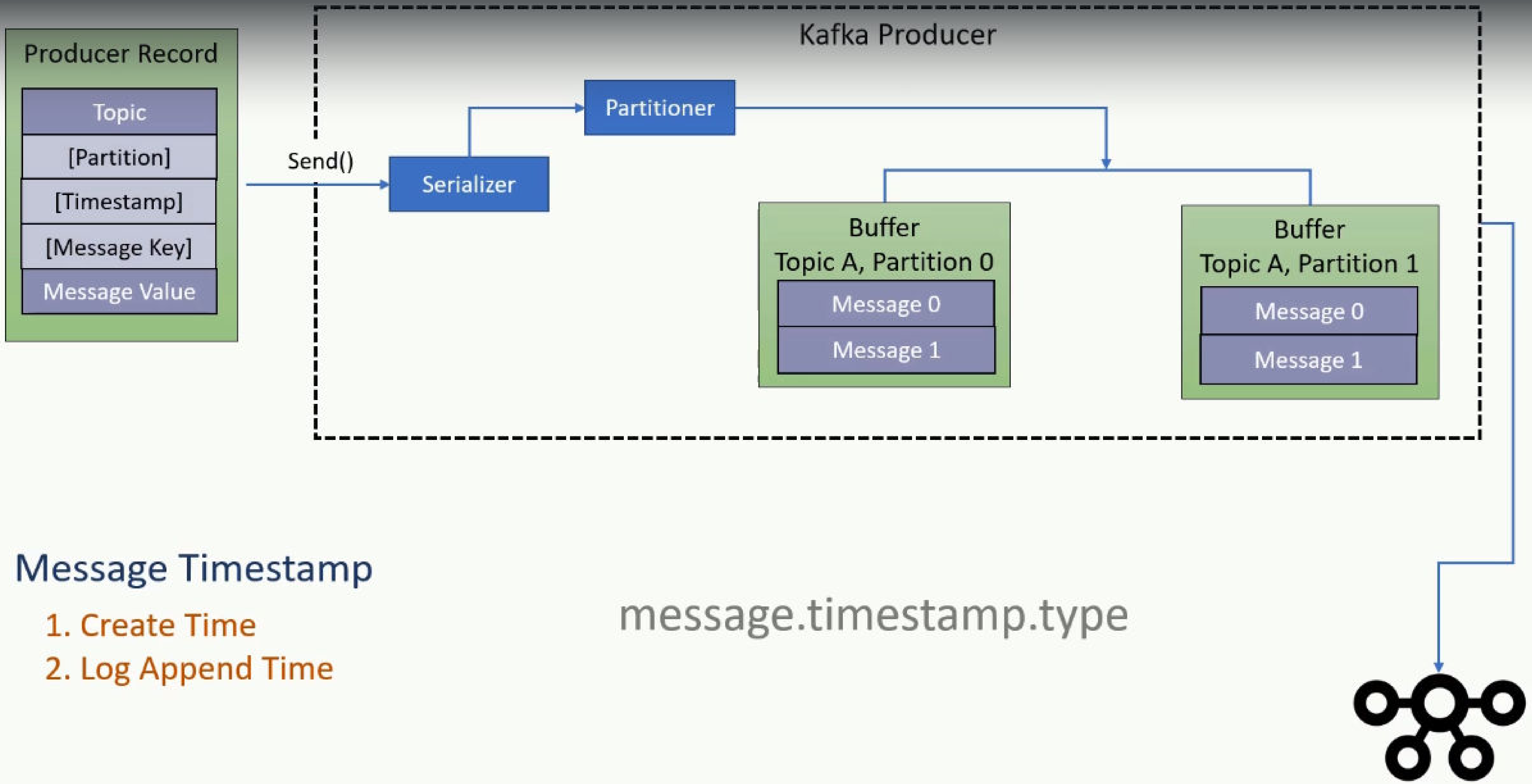
Message 1

Buffer
Topic A, Partition 1

Message 0

Message 1

Kafka Producer API

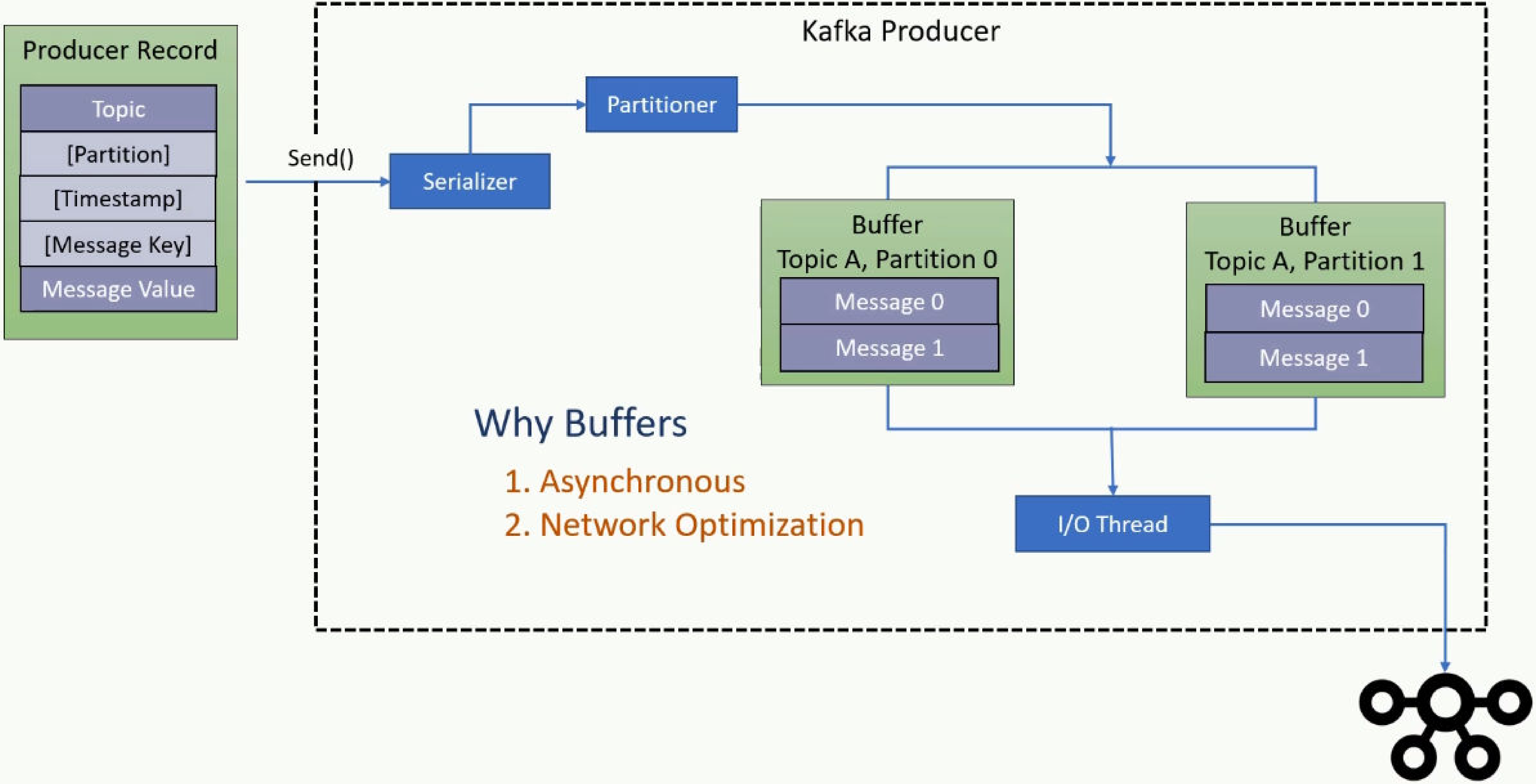


Message Timestamp

1. Create Time
2. Log Append Time

message.timestamp.type

Kafka Producer API



Producer Record

Topic

[Partition]

[Timestamp]

[Message Key]

Message Value

Send()

Kafka Producer

Serializer

Partitioner

Buffer
Topic A, Partition 0

Message 0

Message 1

Buffer
Topic A, Partition 1

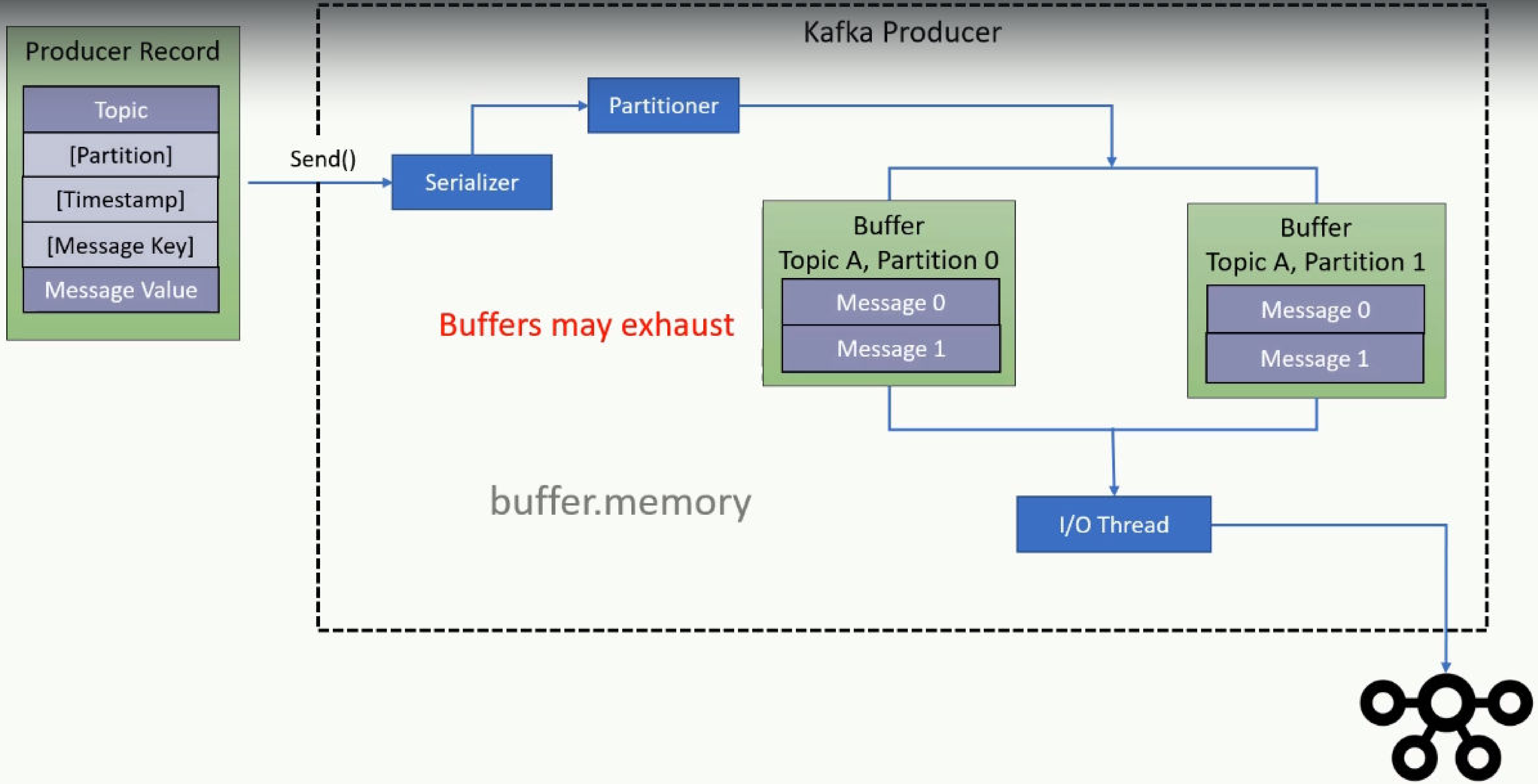
Message 0

Message 1

I/O Thread

Why Buffers

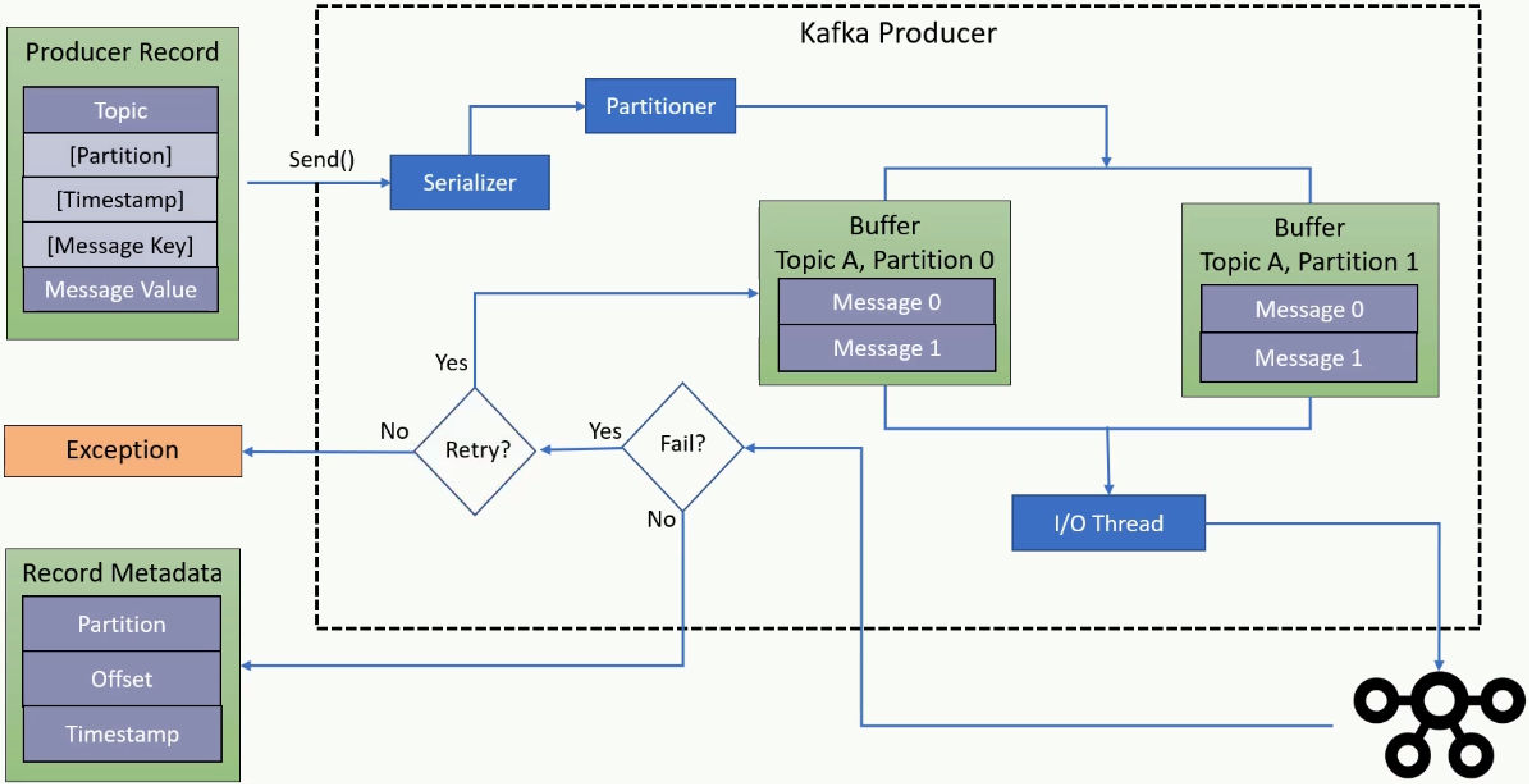
1. Asynchronous
2. Network Optimization



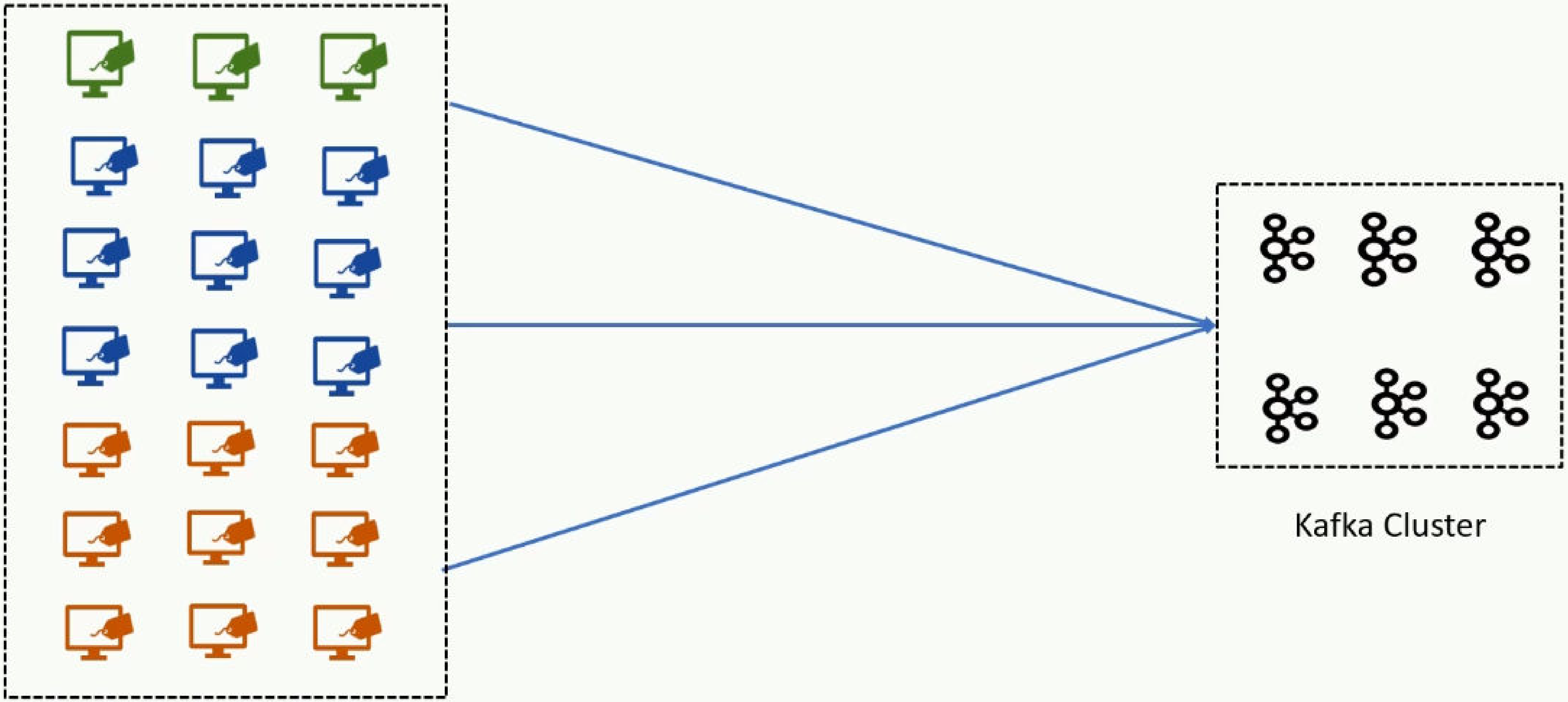
Buffers may exhaust

buffer.memory

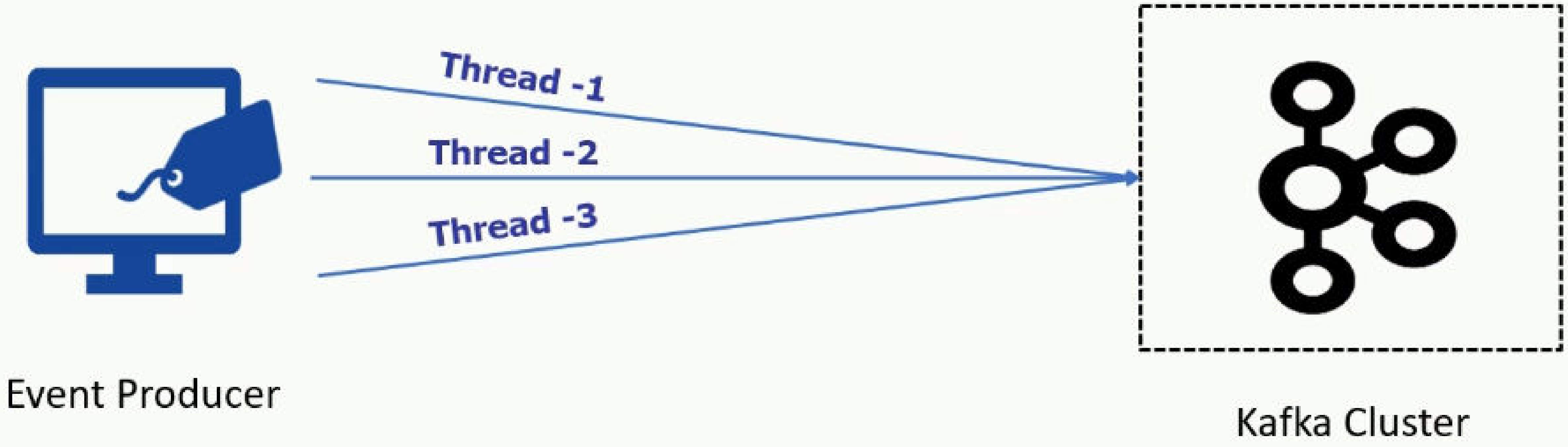
Kafka Producer API


Producer Record
Topic
[Partition]
[Timestamp]
[Message Key]
Message Value
Send()
Kafka Producer
Serializer
Partitioner
Buffer
Topic A, Partition 0
Message 0
Message 1
Buffer
Topic A, Partition 1
Message 0
Message 1
Yes
No
Retry?
Yes
Fail?
No
Exception
I/O Thread
Record Metadata
Partition
Offset
Timestamp

Scaling Kafka Producer



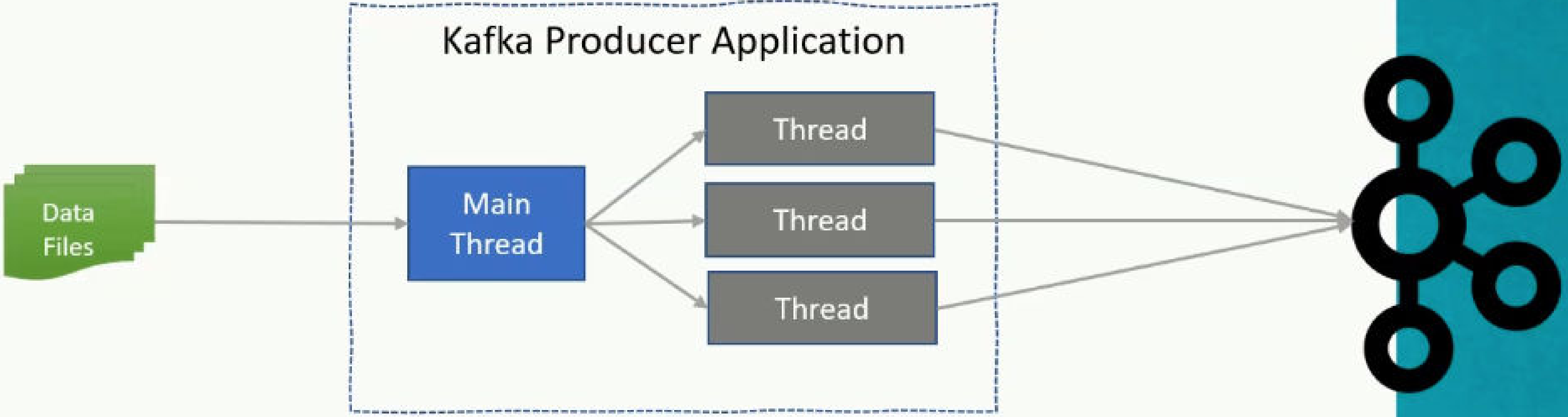
Scaling Kafka Producer

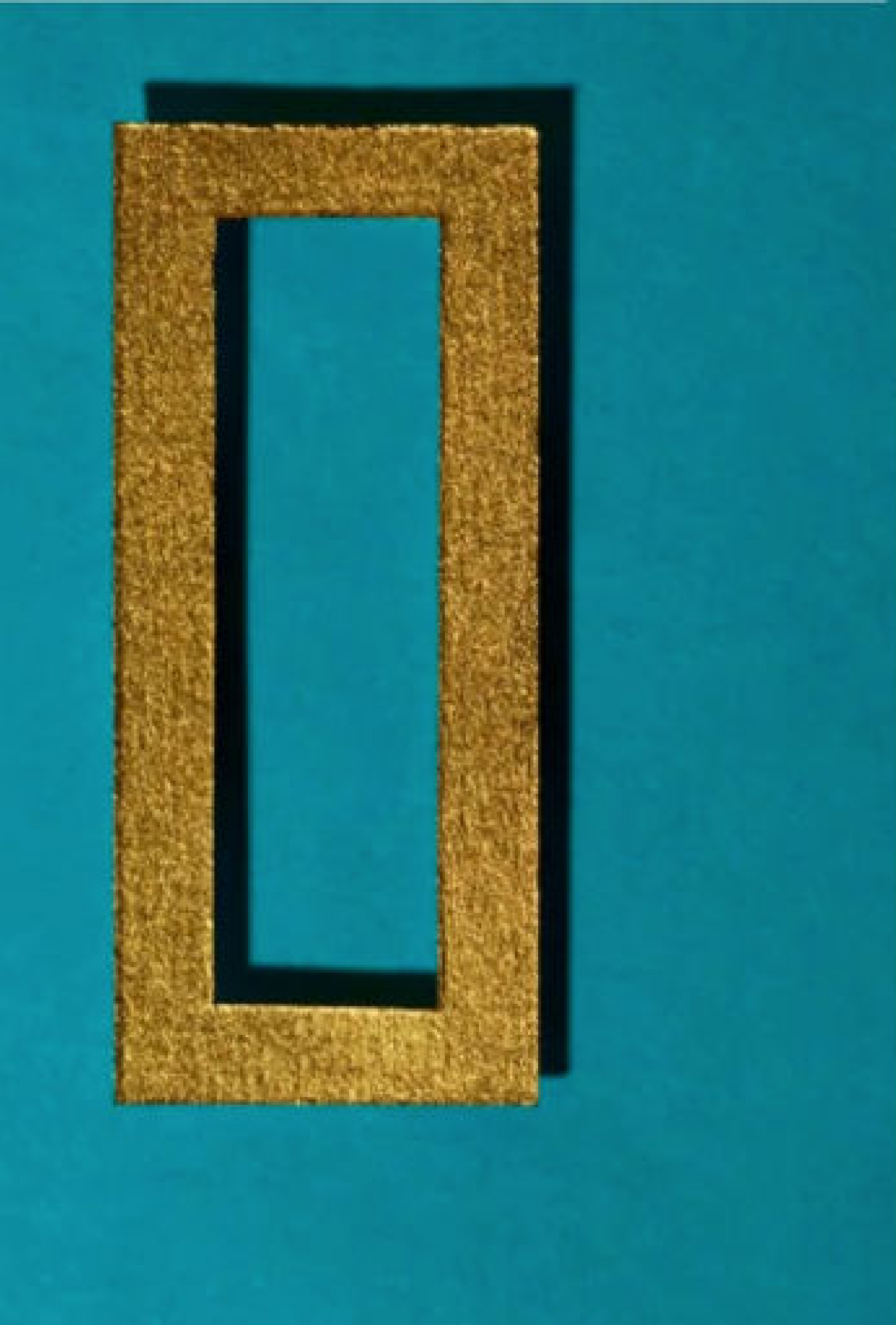


Scaling Kafka Producer

Problem Statement

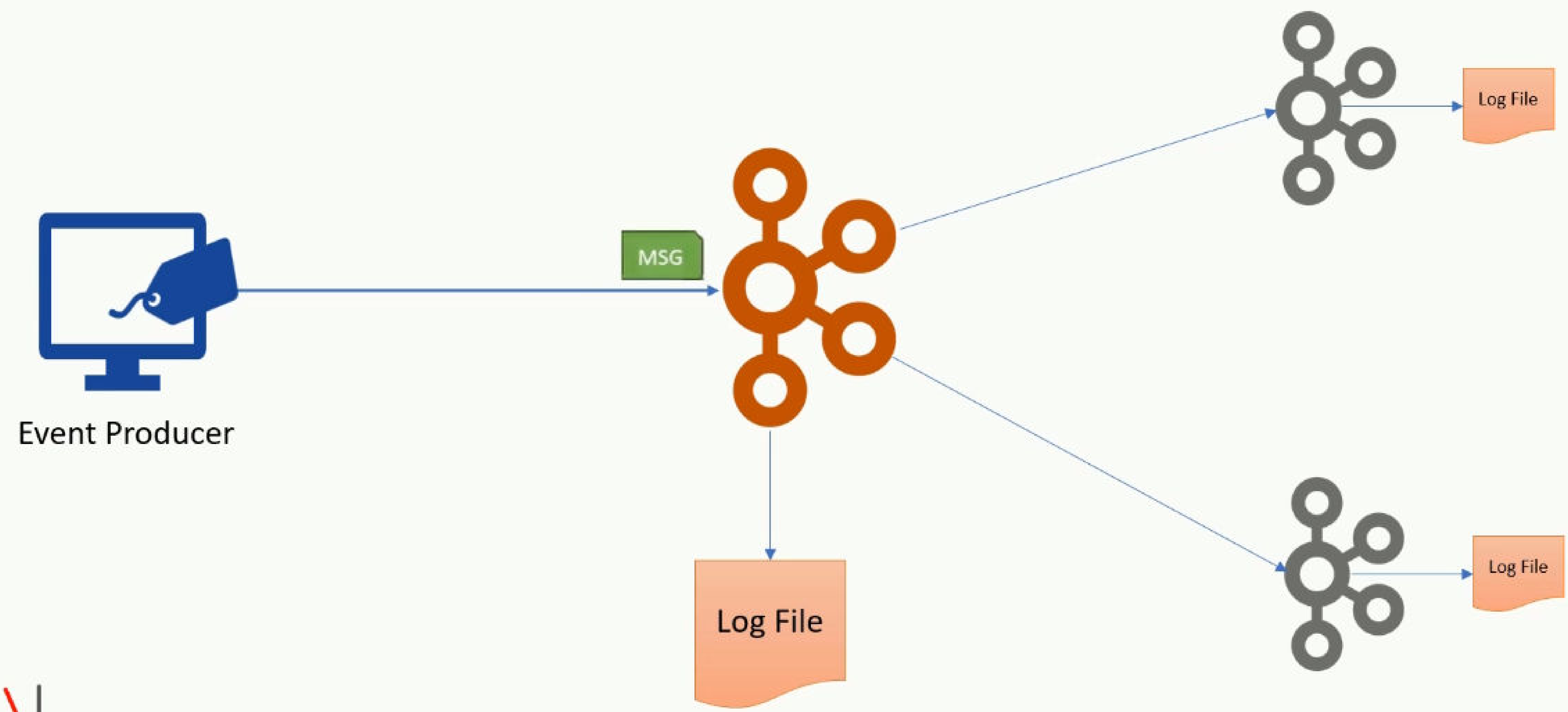
Create a multi-threaded Kafka Producer that sends data from a list of files to a Kafka topic such that independent thread streams each file.





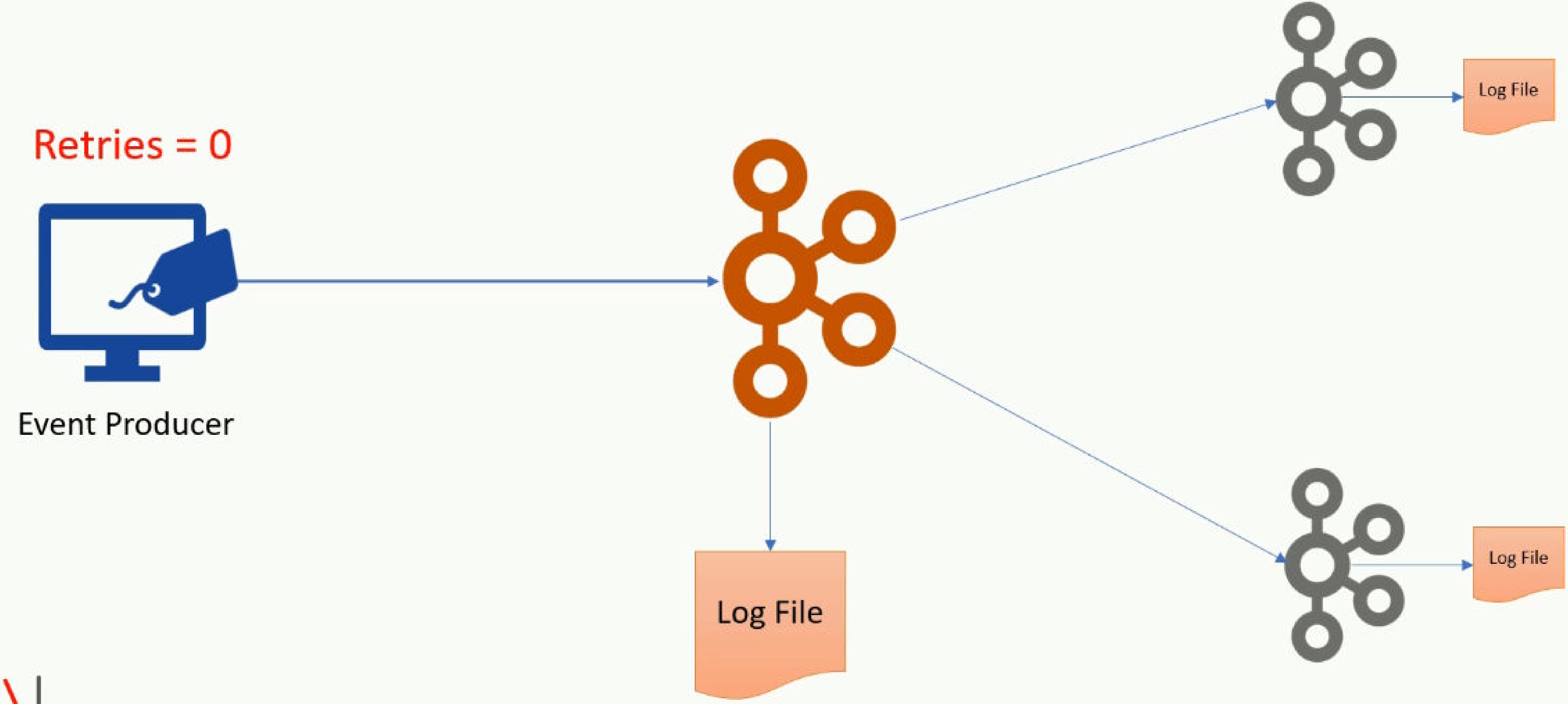
Once

At Least Once



At Most Once

Retries = 0



Event Producer

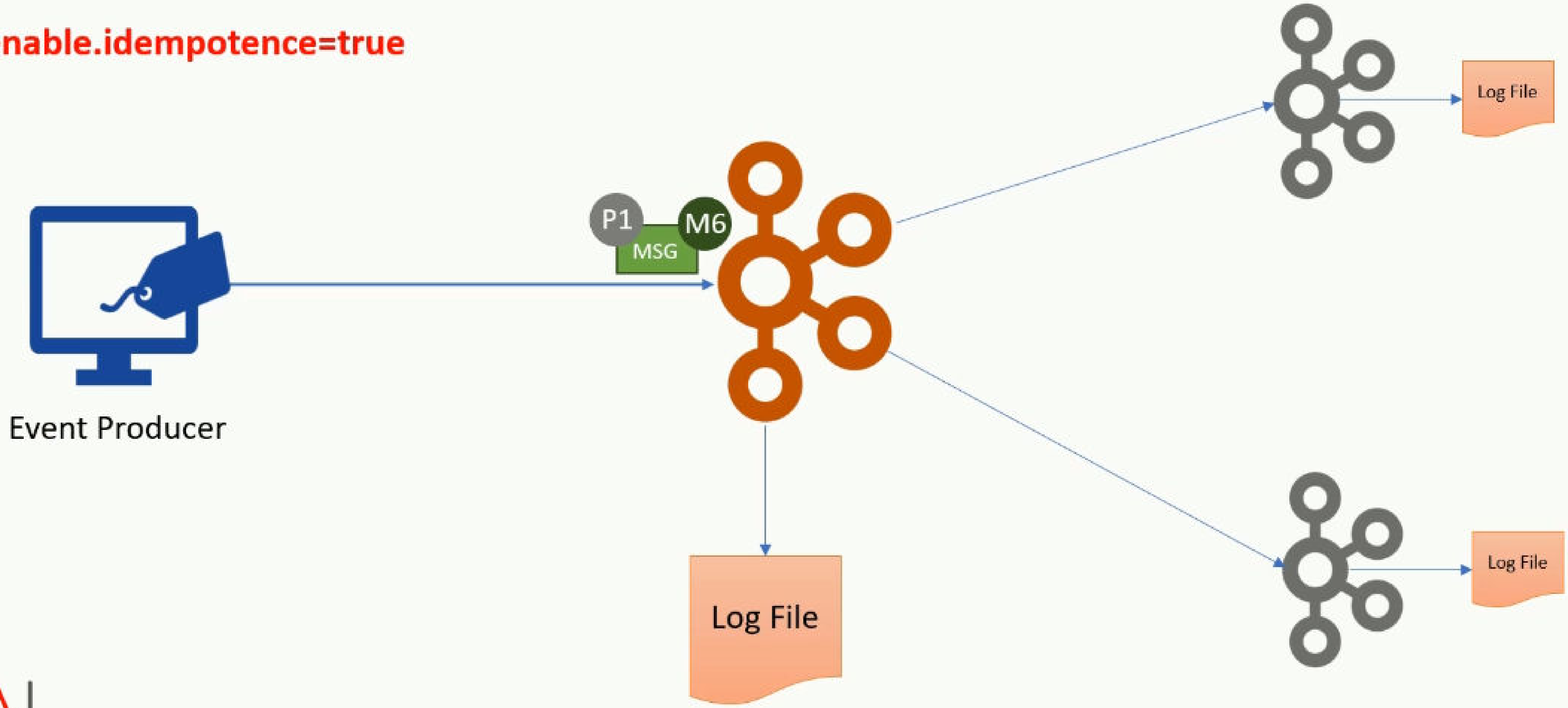
Log File

Log File

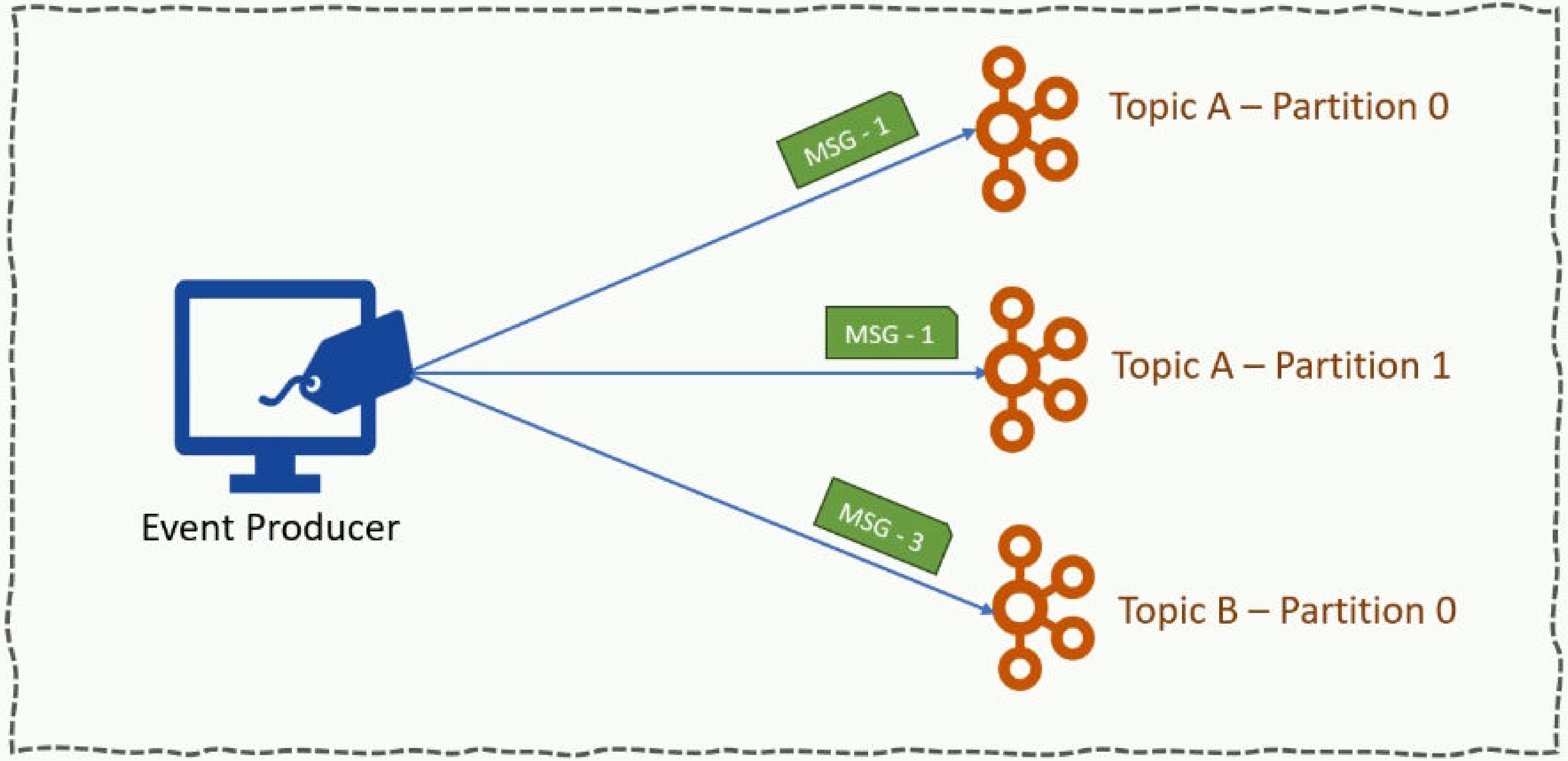
Log File

Exactly Once

enable.idempotence=true



Implementing Transactions



MSG - 1

MSG - 1

MSG - 3

Event Producer

Topic A – Partition 0

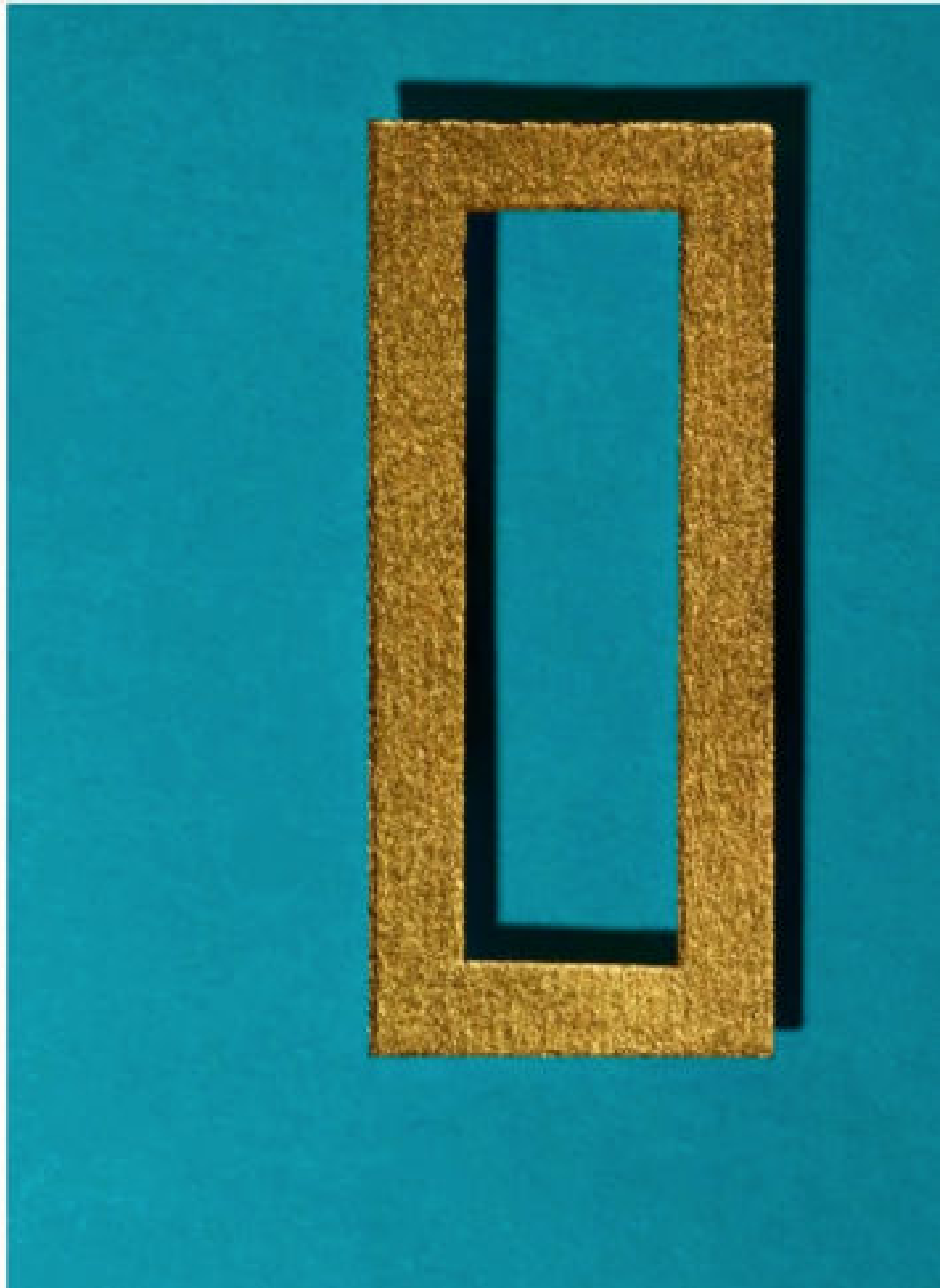
Topic A – Partition 1

Topic B – Partition 0

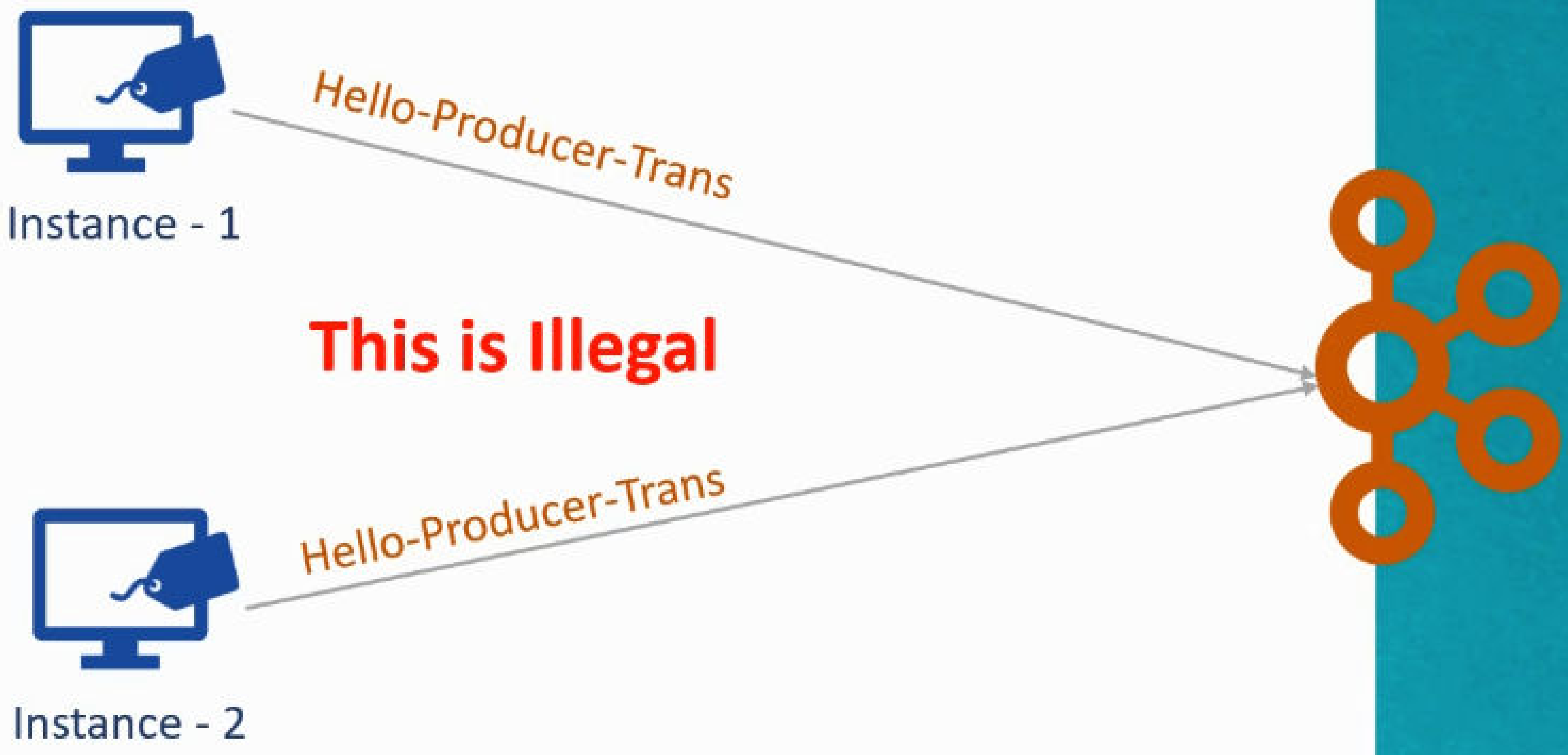
Single Transaction – All or Nothing

Implementing Transactions

1. Transaction depends on Idempotence
2. transactional_id_config must be unique



Implementing Transactions



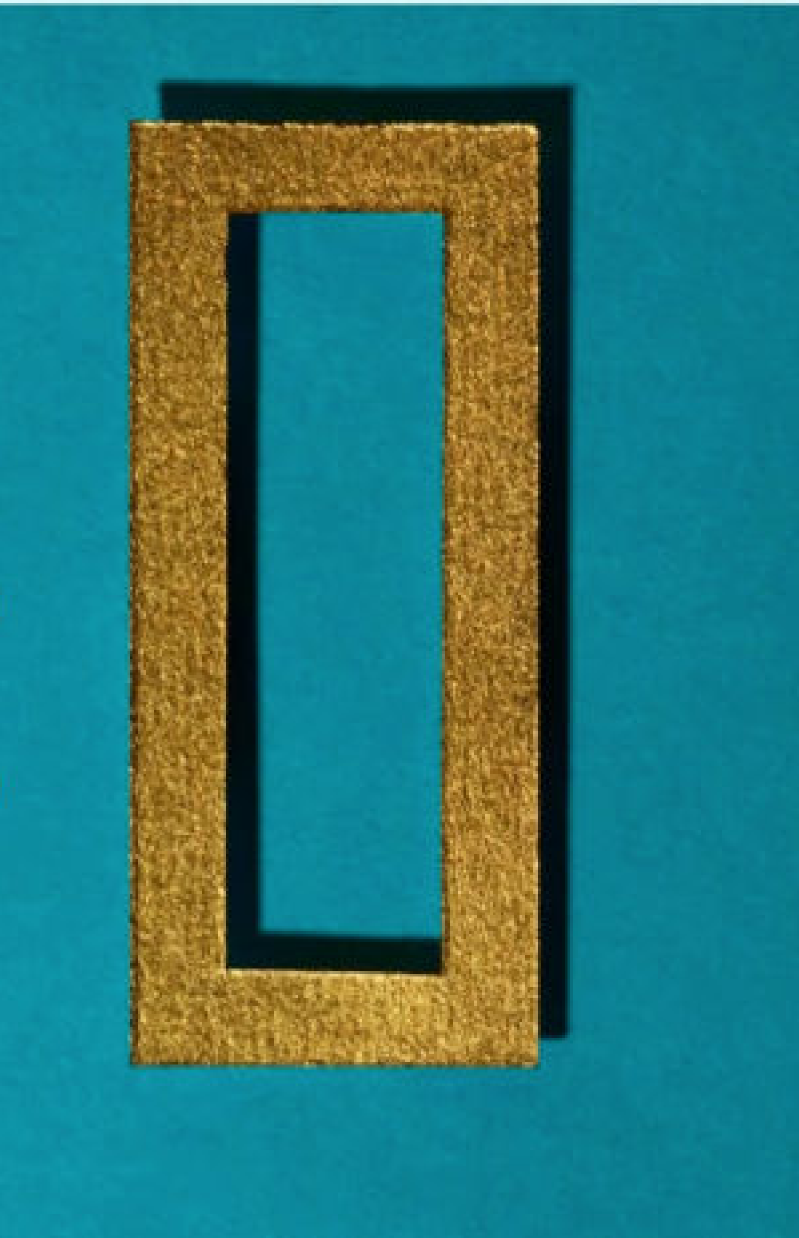
Instance - 1

Hello-Producer-Trans

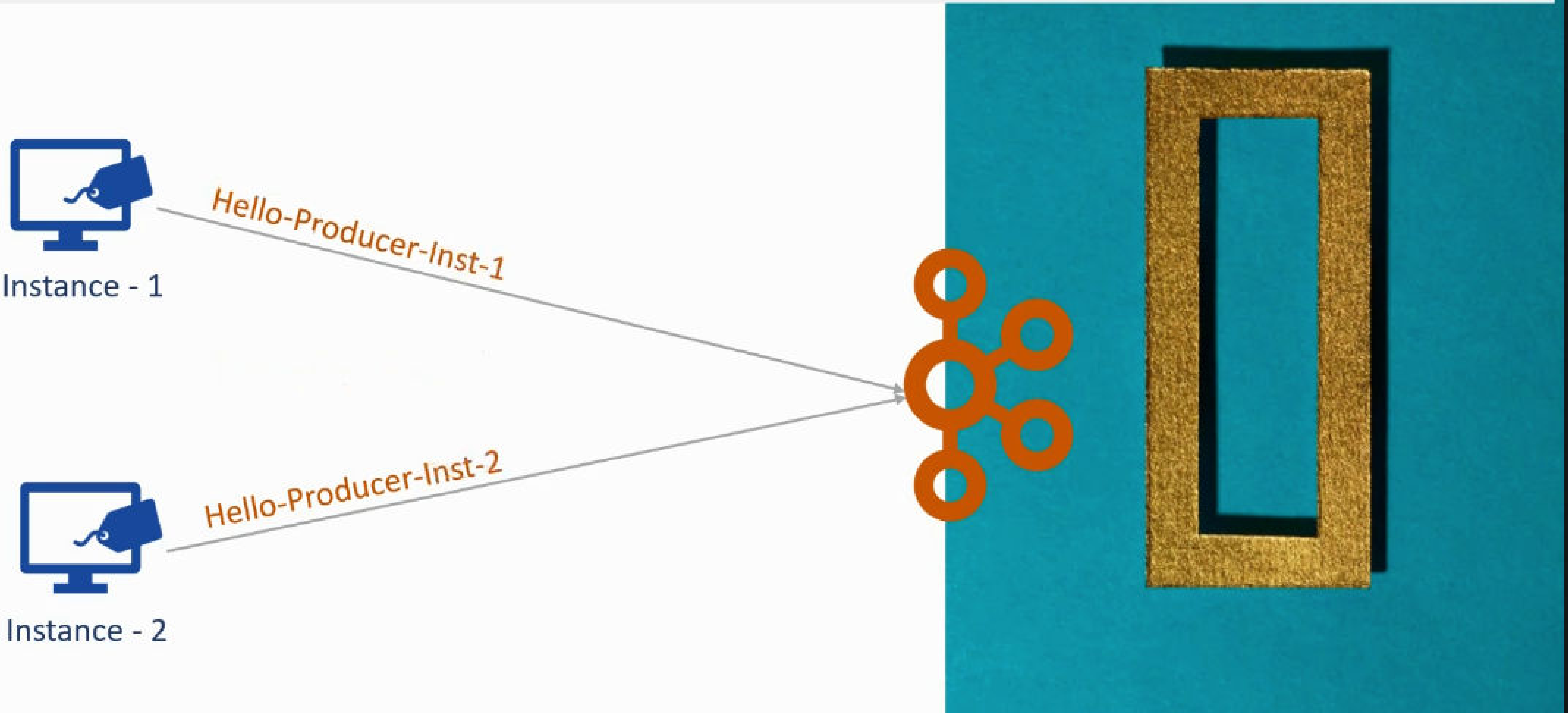
This is Illegal

Hello-Producer-Trans

Instance - 2



Implementing Transactions


Hello-Producer-Inst-1
Instance - 1
Hello-Producer-Inst-2
Instance - 2