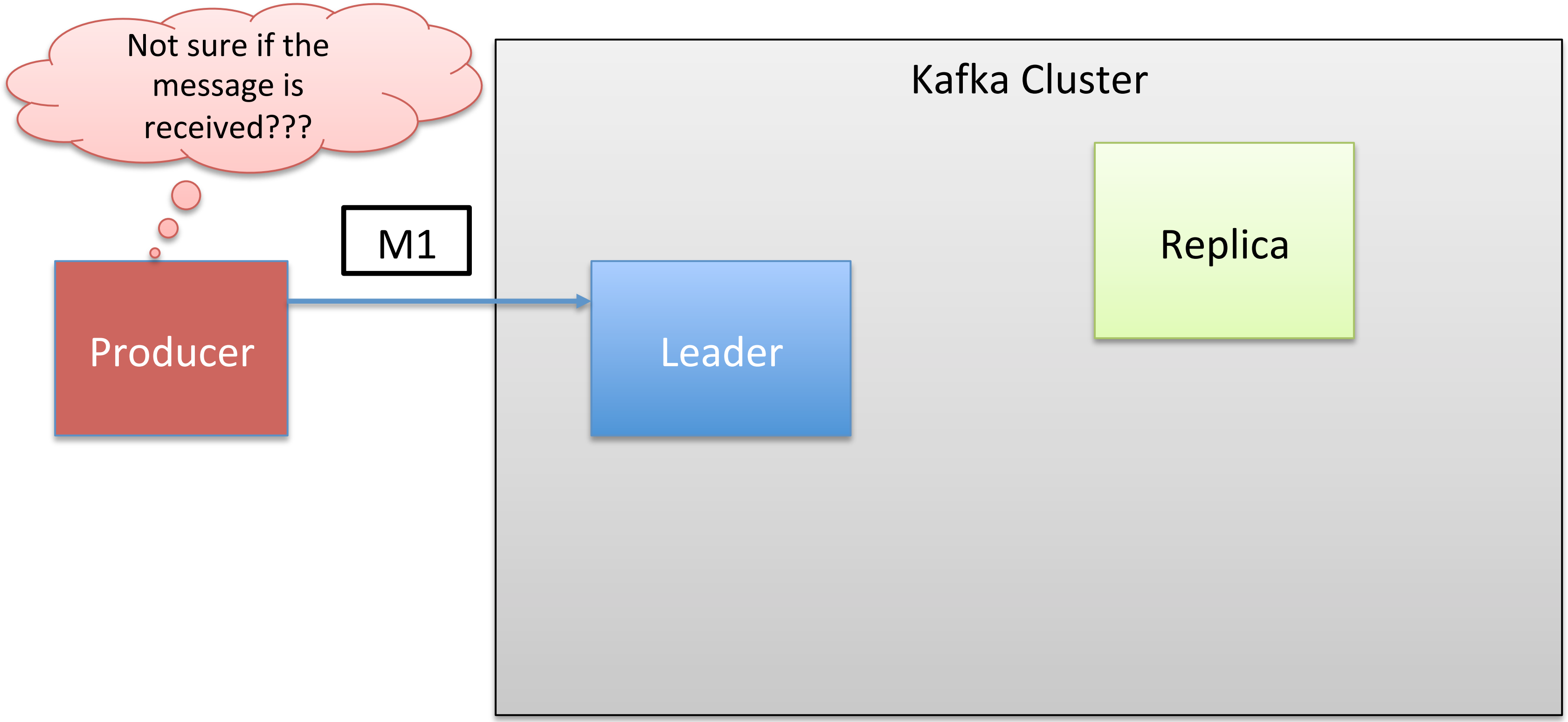
Kafka Delivery Guarantees

Introduction

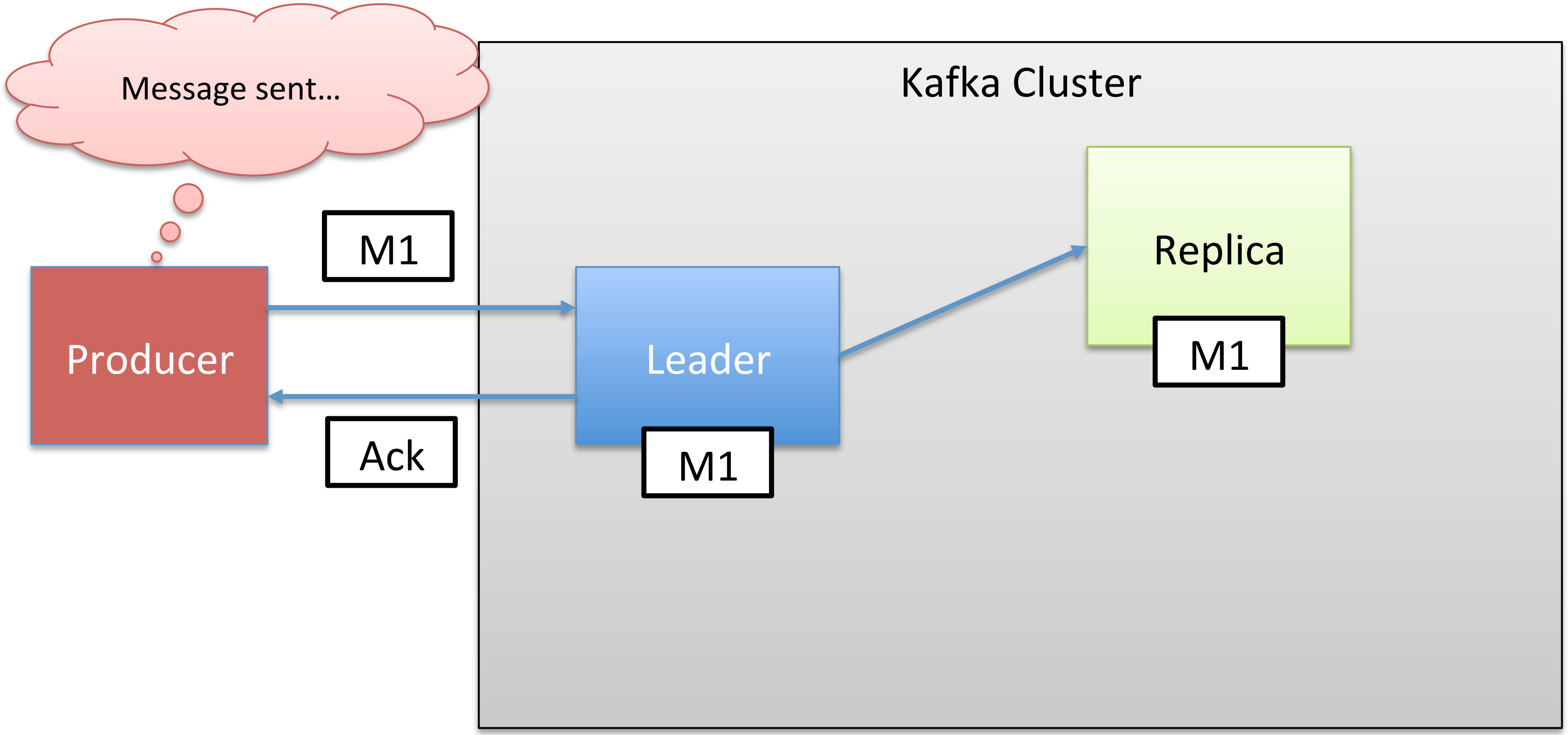
- Kafka’s delivery model leaves potential gaps in typical enterprise deployment scenarios
- Naïve programming may lead to
  - Messages being processed multiple times
  - Messages not being processed

- We will look at a few different scenarios where such failures may occur
- We’ll also look at some technique that can be used to minimize or avoid the failures
  - Idempotent services
  - Kafka’s new exactly once processing guarantee

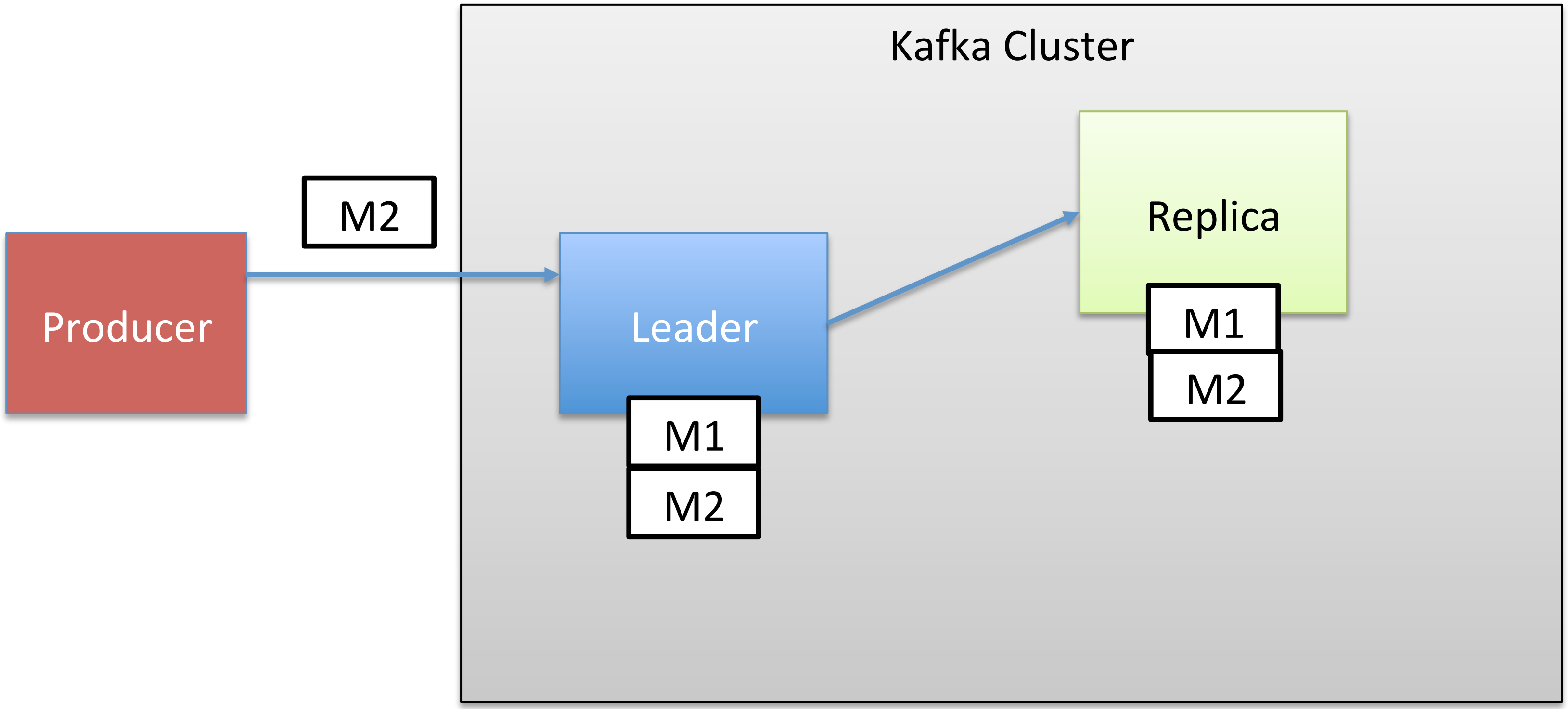
Message Duplication: 1. Send the message



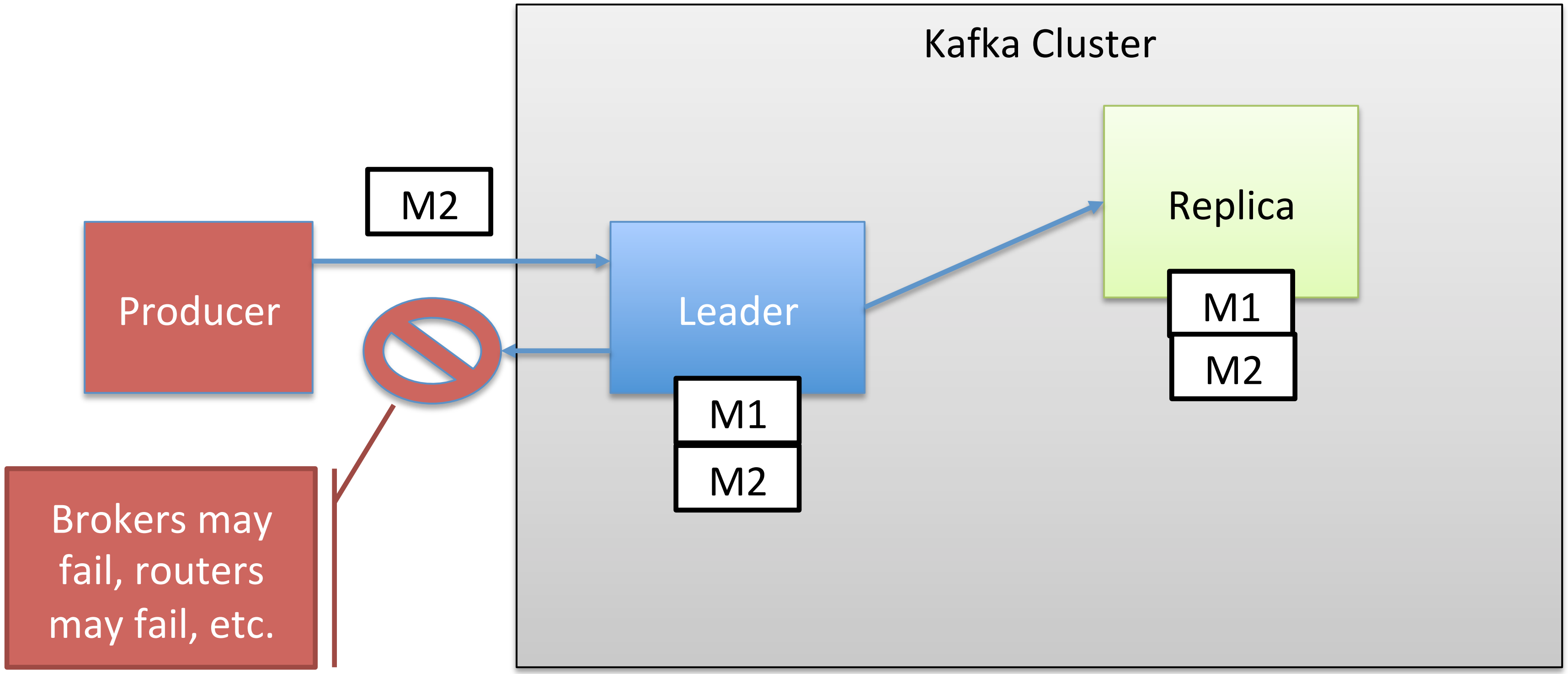
Message Duplication: 2. Message ack



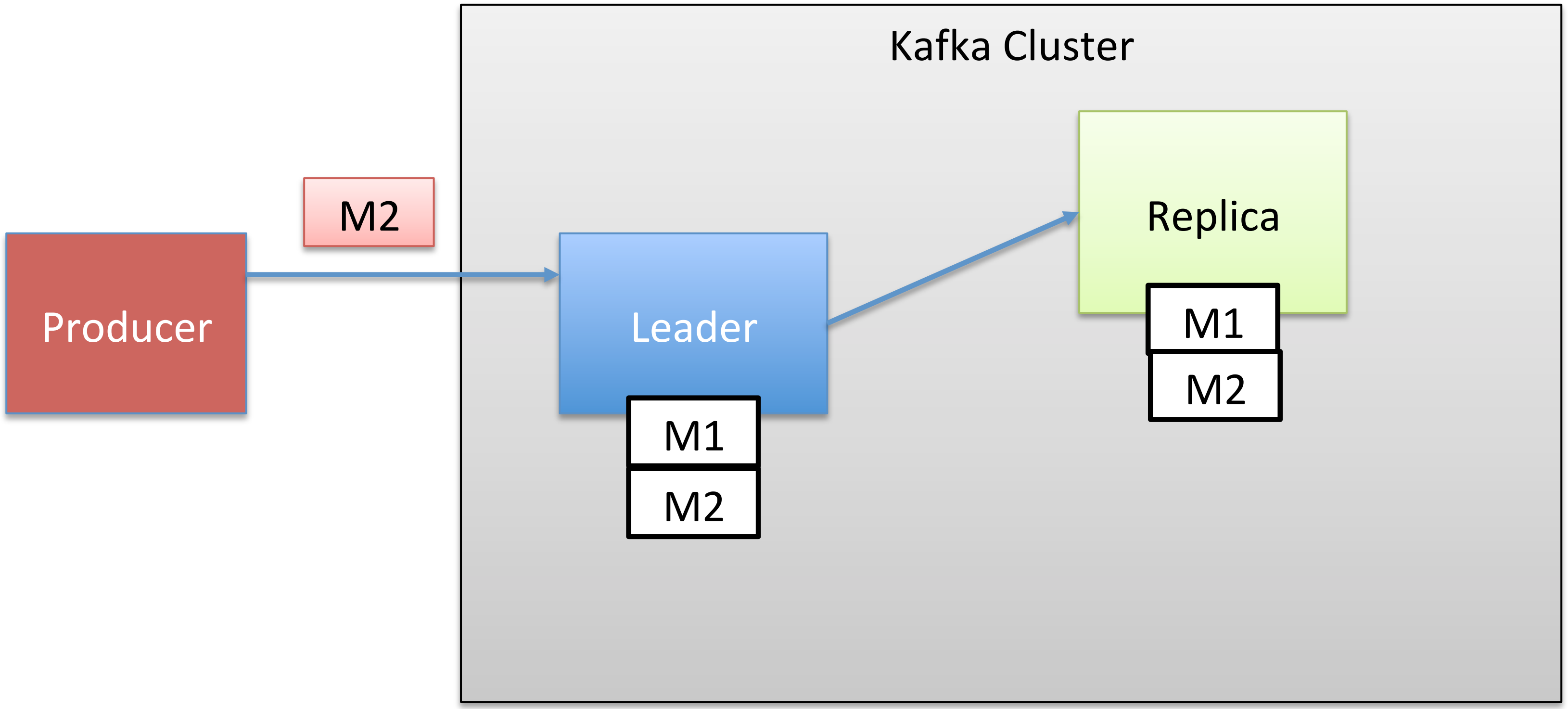
Message Duplication: 3. M2 being sent



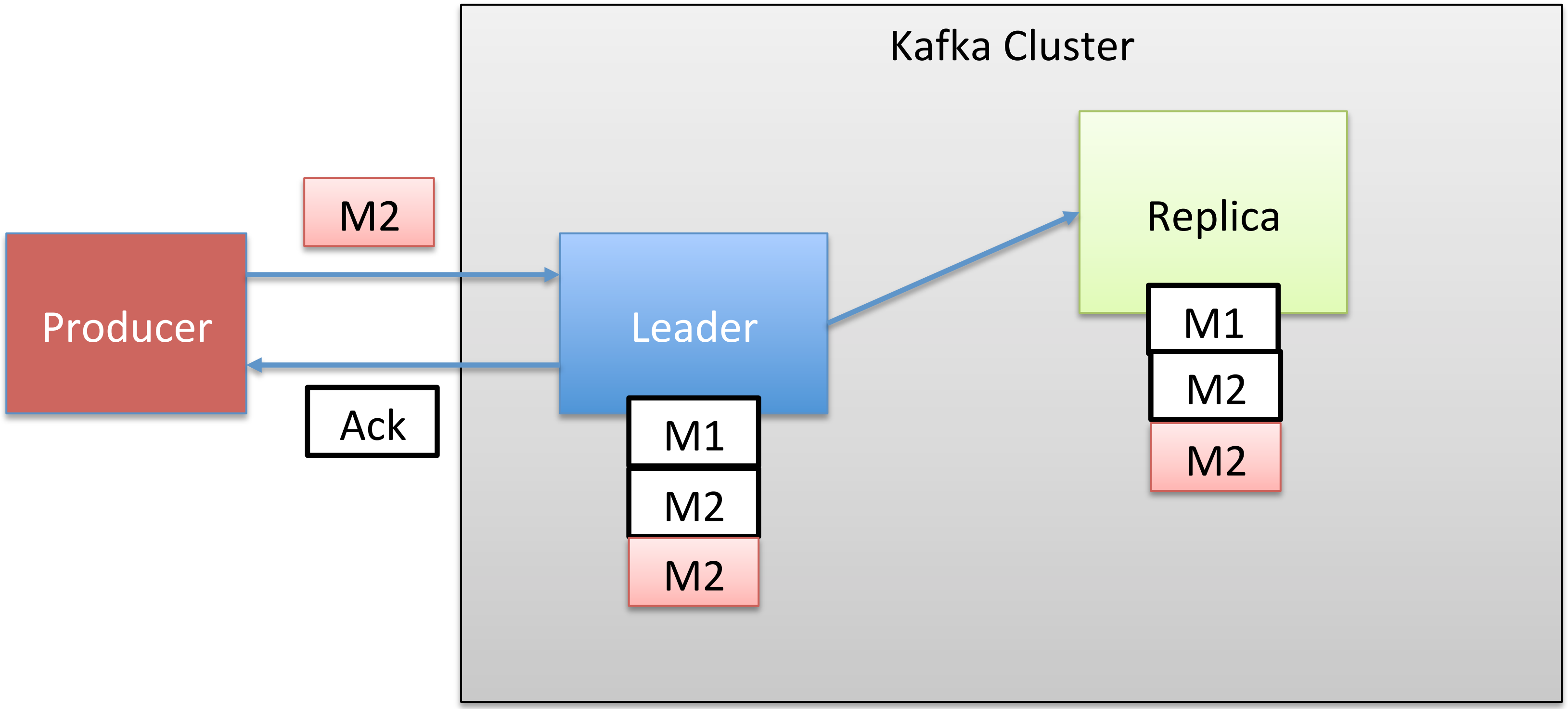
Message Duplication: 4. Message never ack-ed



Message Duplication: 5. M2 is resent



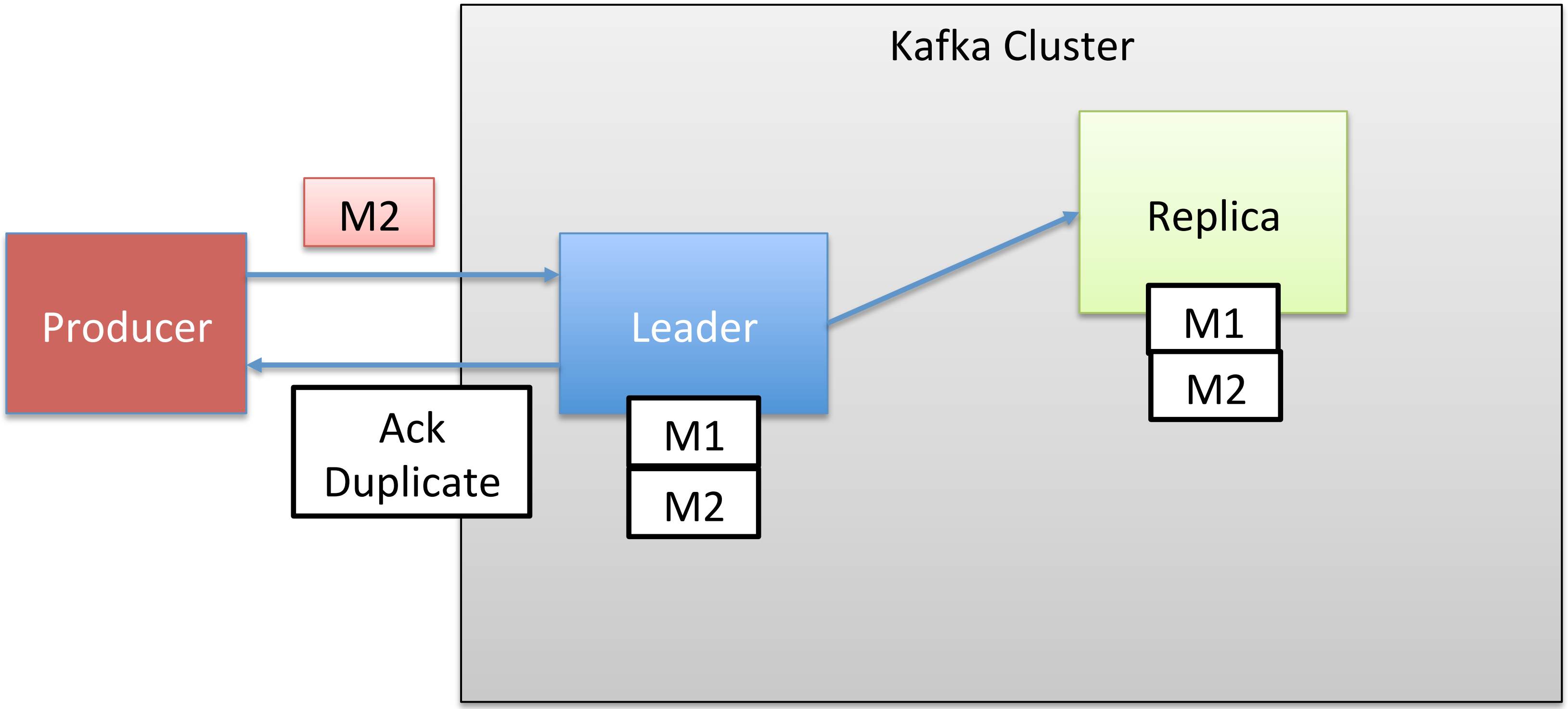
Message Duplication: 5. M2 is resent and stored again



Kafka Exactly Once Producer Solution

- Idempotent producer (per partition)
  - Exactly once
  - In order
- Transactions
  - Atomic writes across partitions

Idempotent Producer



Transaction API

- Atomic writes across multiple partitions
- Either all records are visible, or none

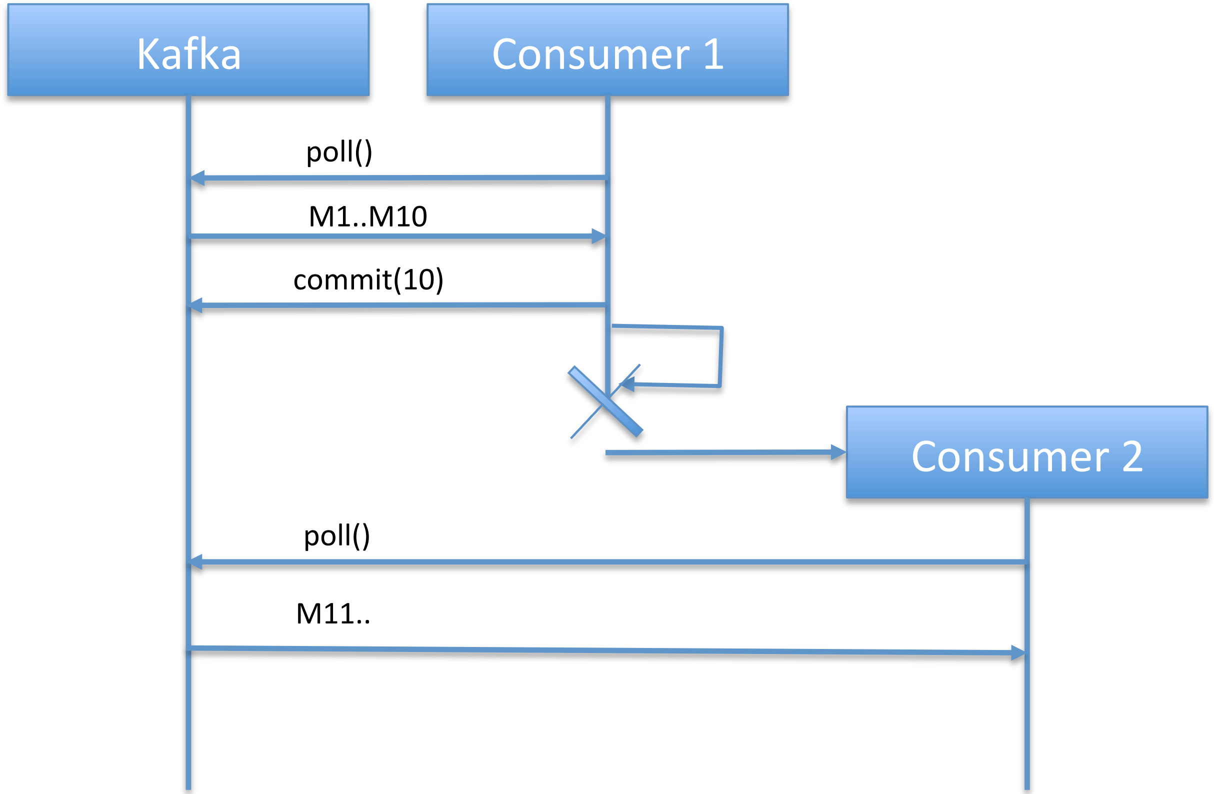
```
producer.initTransaction();
try {
    producer.beginTransaction();
    producer.send(record1);
    producer.send(record2);
    …
    producer.commitTransaction();
}
catch (KafkaException e) {
    producer.abortTransaction();
}
```

Consumer Problems

- Messages missed
  - Offset is committed prior to processing the messages
  - No or only partial processing of incoming messages
  - Consumer failure
- Messages double processed
  - Processing of messages prior to commit
  - Consumer failure before commit
- There is now real failsafe way to ensure that messages are processed exactly once
  - Requires distributed transaction

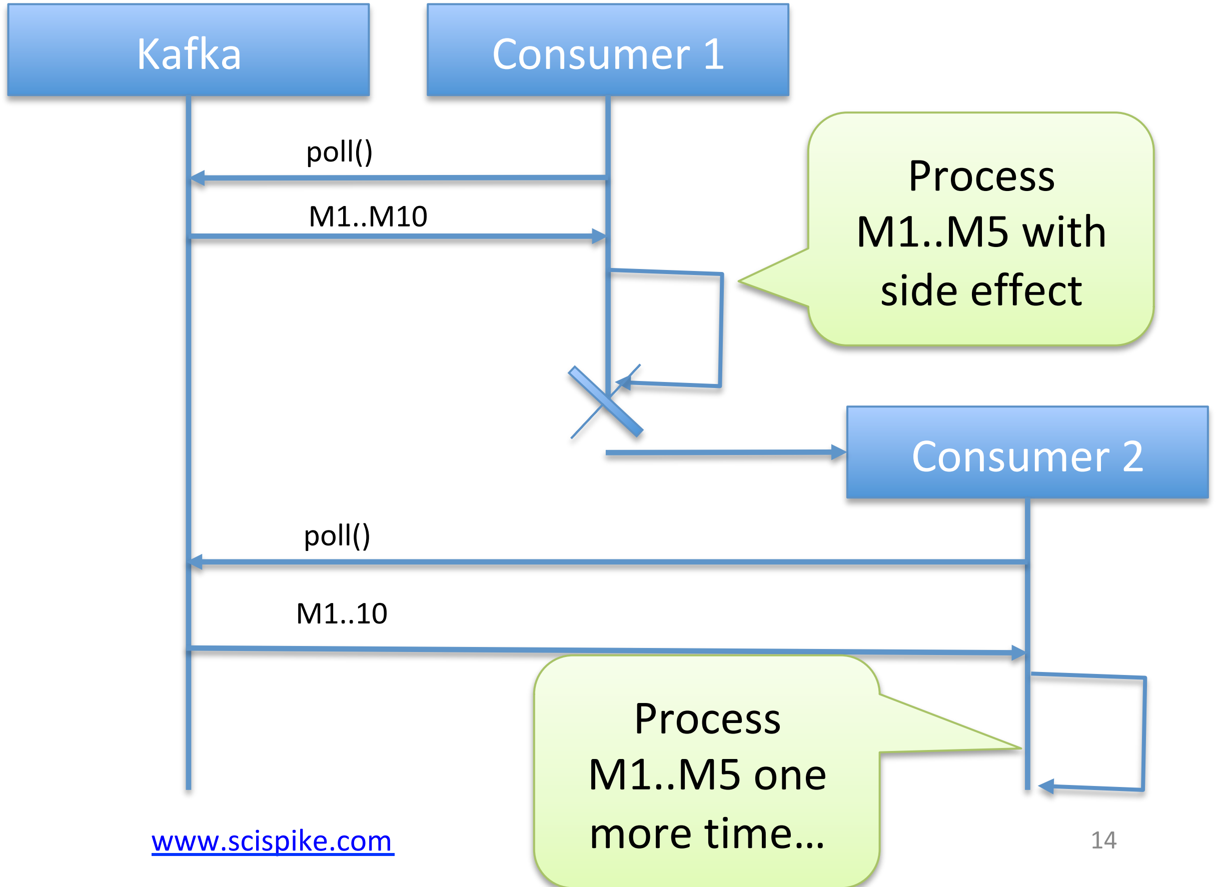
Consumer Message Loss

1. Messages received (M1…M10)
2. Offset committed (M10)
3. Processing M1..M5
4. Consumer crashes
5. Consumer resumes
6. Message M11.. is delivered



Consumer Double Processing

1. Messages received (M1…M10)
2. Processing M1..M5
3. Consumer crashes
4. Consumer resumes
5. Message M11.. is delivered


Kafka
Consumer 1
poll()
M1..M10
Process M1..M5 with side effect
Consumer 2
poll()
M1..10
Process M1..M5 one more time…

Kafka Exactly Once Stream Processing

- Kafka provides an exactly once guarantee for stream processing
- Important to understand the guarantee
  - The effect on the Kafka stream state is AS IF each message was processed exactly once
    - The message may actually be processed multiple times
    - However, the offset is moved with the state changes
- This means:
  - If the processing have side effect OUTSIDE KAFKA this side effect may have to be idempotent!

Configuration of Exactly Once

Producer Configuration

enable.idempotence=true

Also recommended

max.inflight.requests.per.connection=1

acks="all"

retries= MAX_INT

Consumer Configuration

isolation.level = read_committed

or

isolation.level = read_uncommitted

processing.mode = "exactly_once"

Summary

- Kafka supports an exactly once message delivery guarantee

- Solved with
  - Idempotent producer
  - Transactions
  - Stream processing

- It’s important to know that the exactly once guarantee on the consumer is only guaranteed within the context of Kafka!