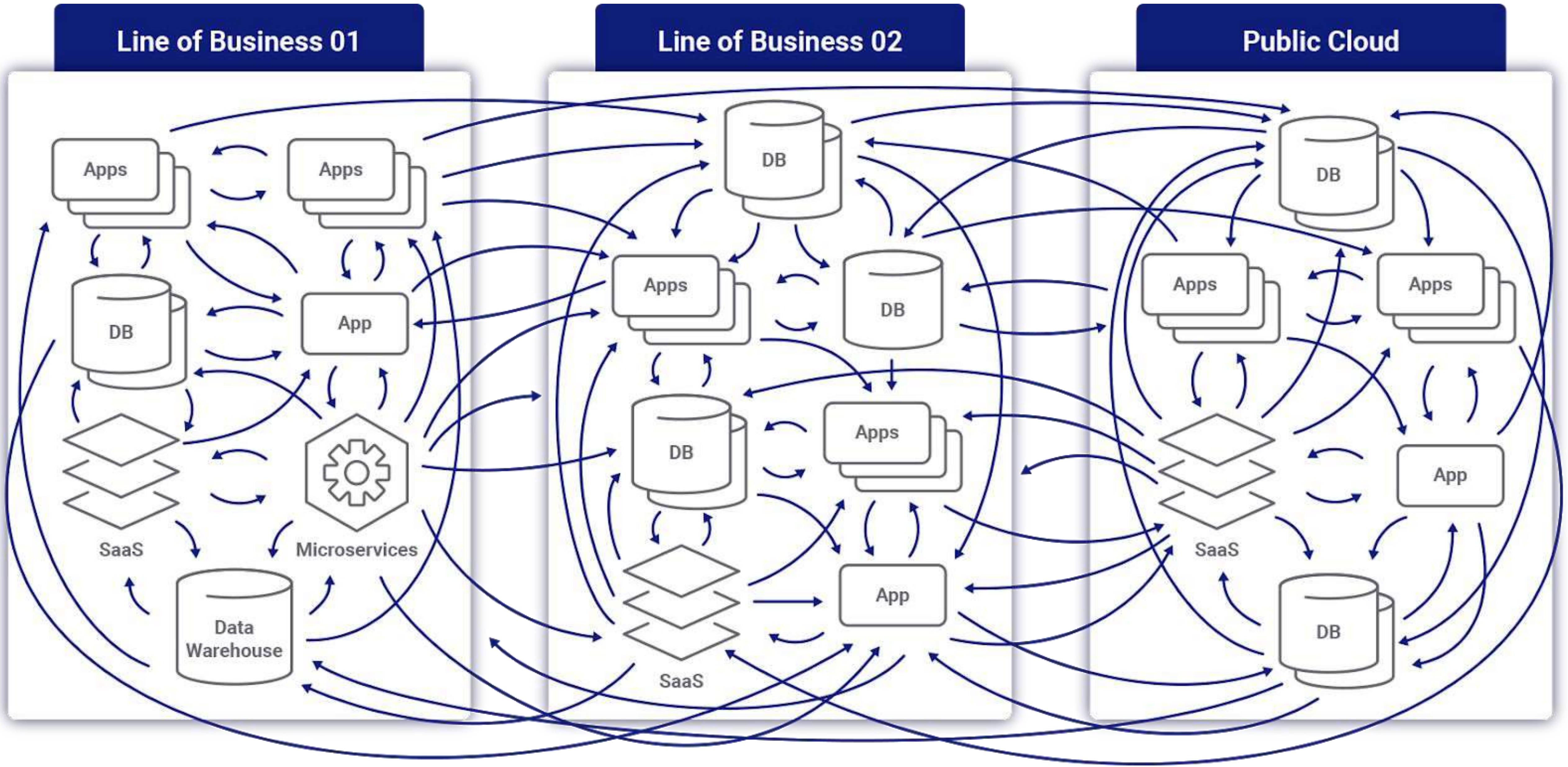
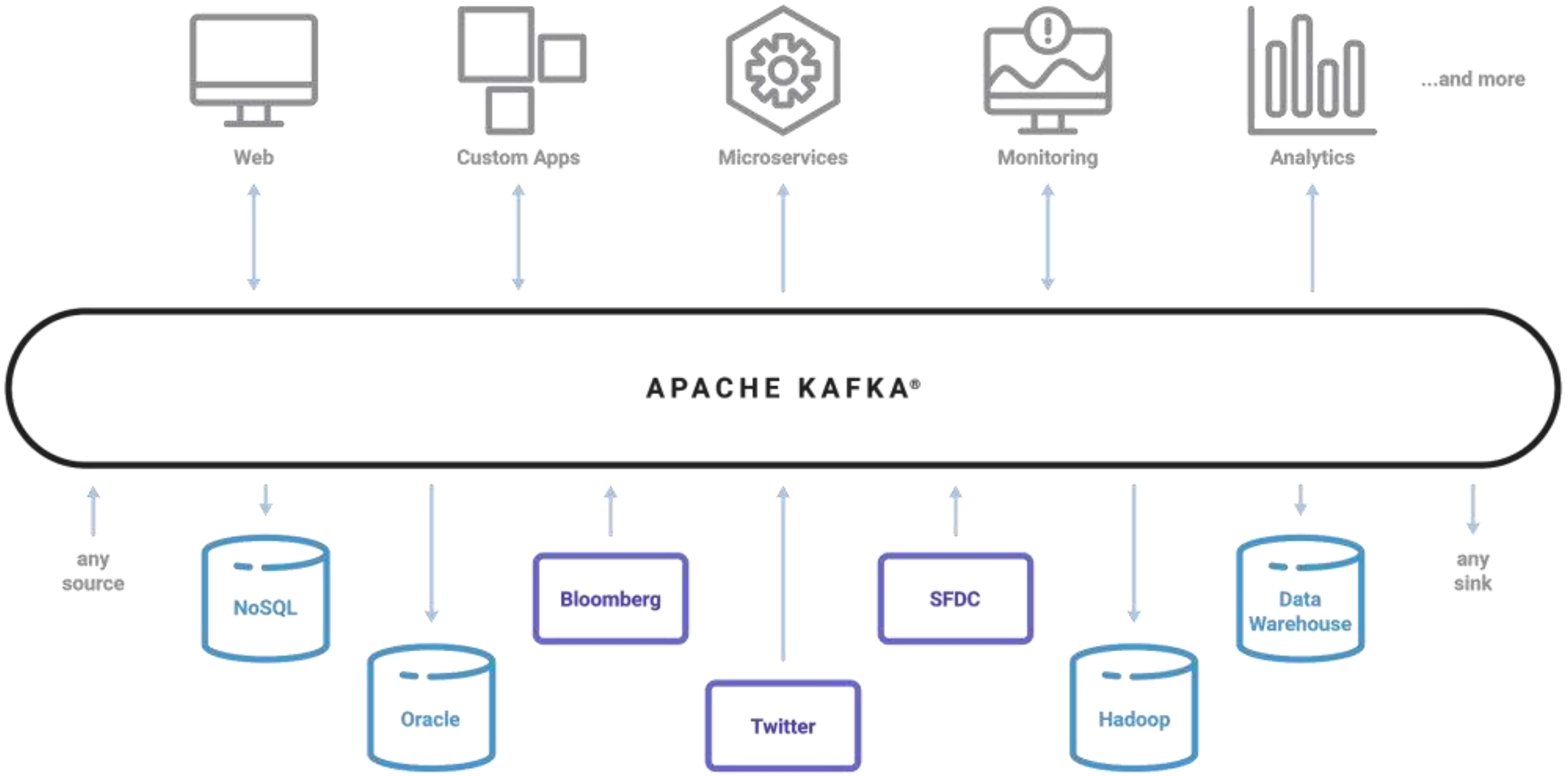
Confluent & Apache Kafka Patterns / Anti-patterns

Marcelo Manta and Jean Louis Boudart

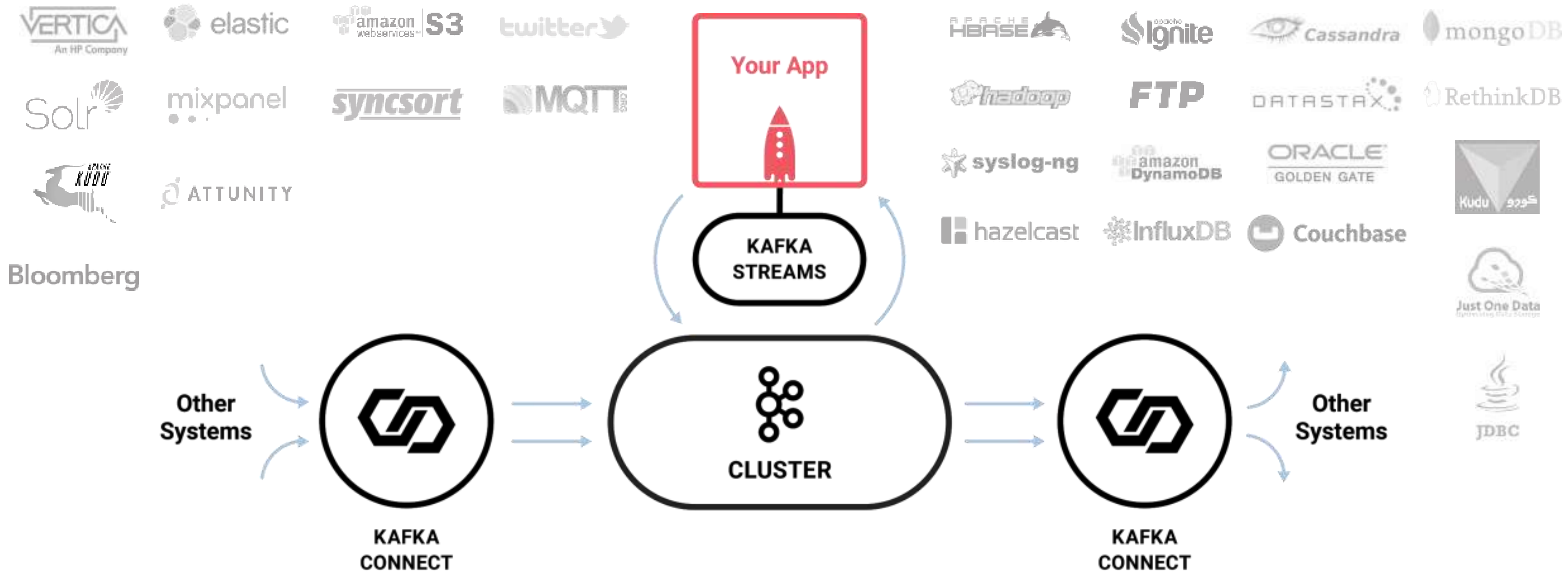
Problem ?


Line of Business 01
Line of Business 02
Public Cloud
Apps
Apps
DB
App
SaaS
Microservices
Data Warehouse
DB
Apps
DB
DB
Apps
App
SaaS
DB
Apps
Apps
App
SaaS
DB

With


Web
Custom Apps
Microservices
Monitoring
Analytics
...and more
APACHE KAFKA®
any source
NoSQL
Oracle
Bloomberg
Twitter
SFDC
Hadoop
Data Warehouse
any sink

How


Your App
KAFKA STREAMS
CLUSTER
Other Systems
KAFKA CONNECT
Other Systems
KAFKA CONNECT

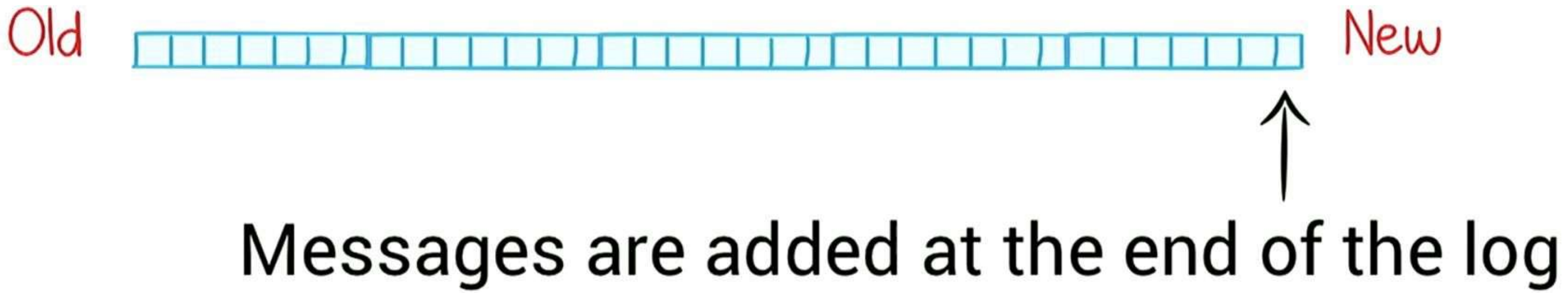
In short

Publish & Subscribe

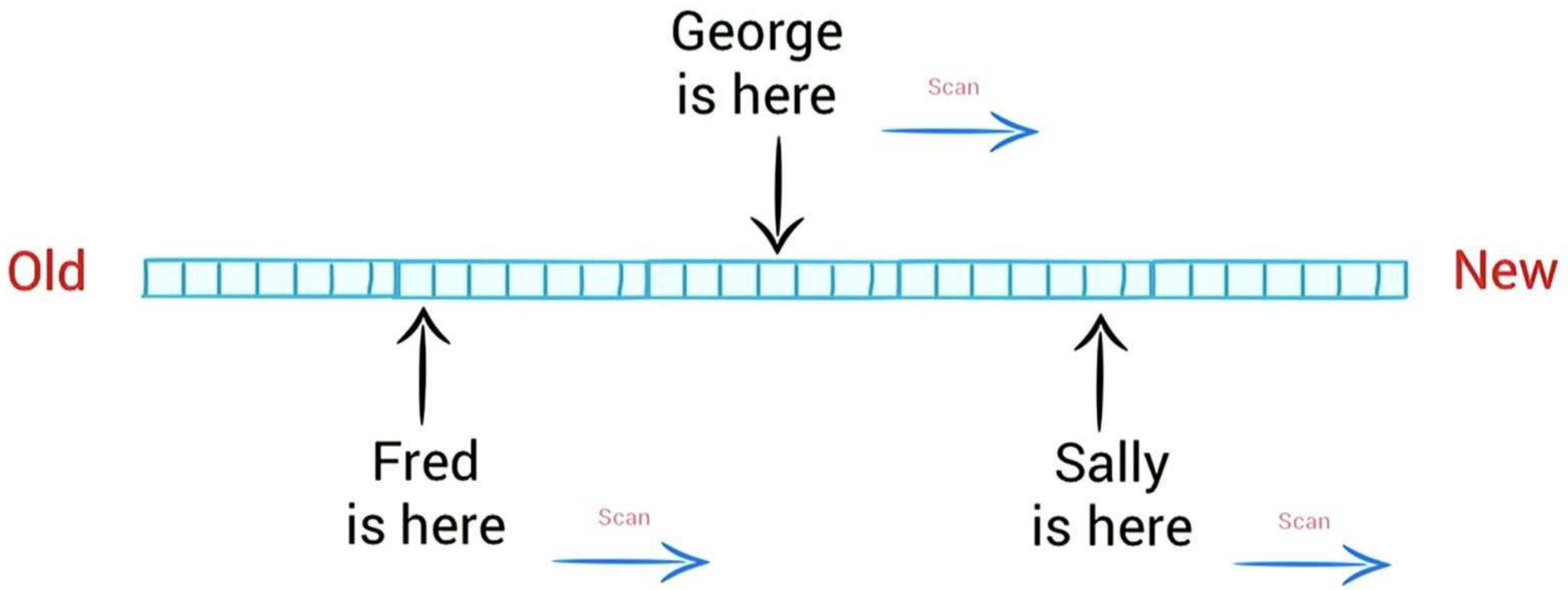
Store & ETL

Process

From a simple idea



From a simple idea

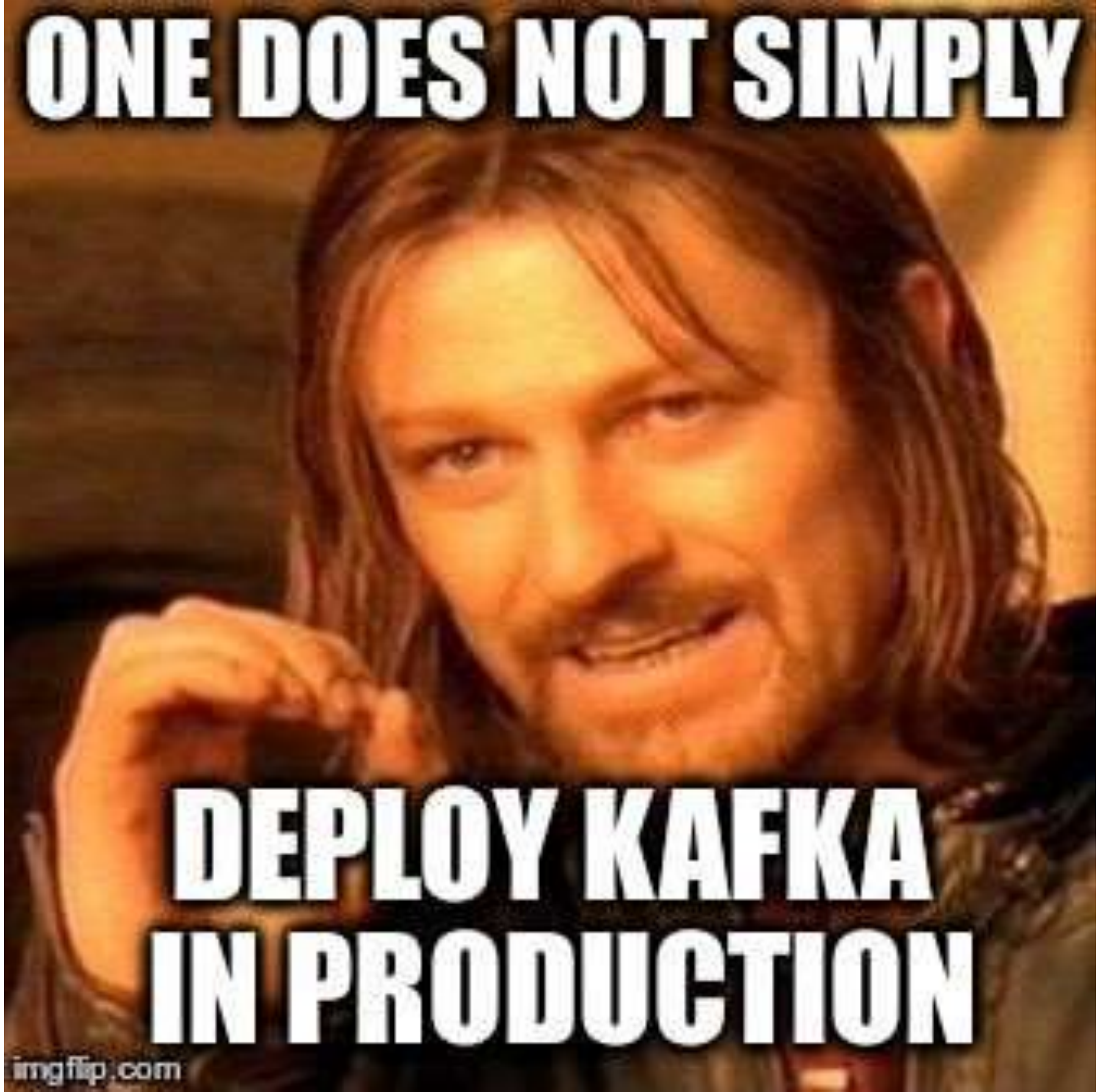


with great properties !

- Scalability
- Retention
- Durability
- Replication
- Security
- Resiliency
- Throughput
- Ordering
- Exactly Once Semantic
- Transaction
- Idempotency
- Immutability
- …

So goooooood

What could potentially go wrong ?


ONE DOES NOT SIMPLY
DEPLOY KAFKA
IN PRODUCTION
imgflip.com

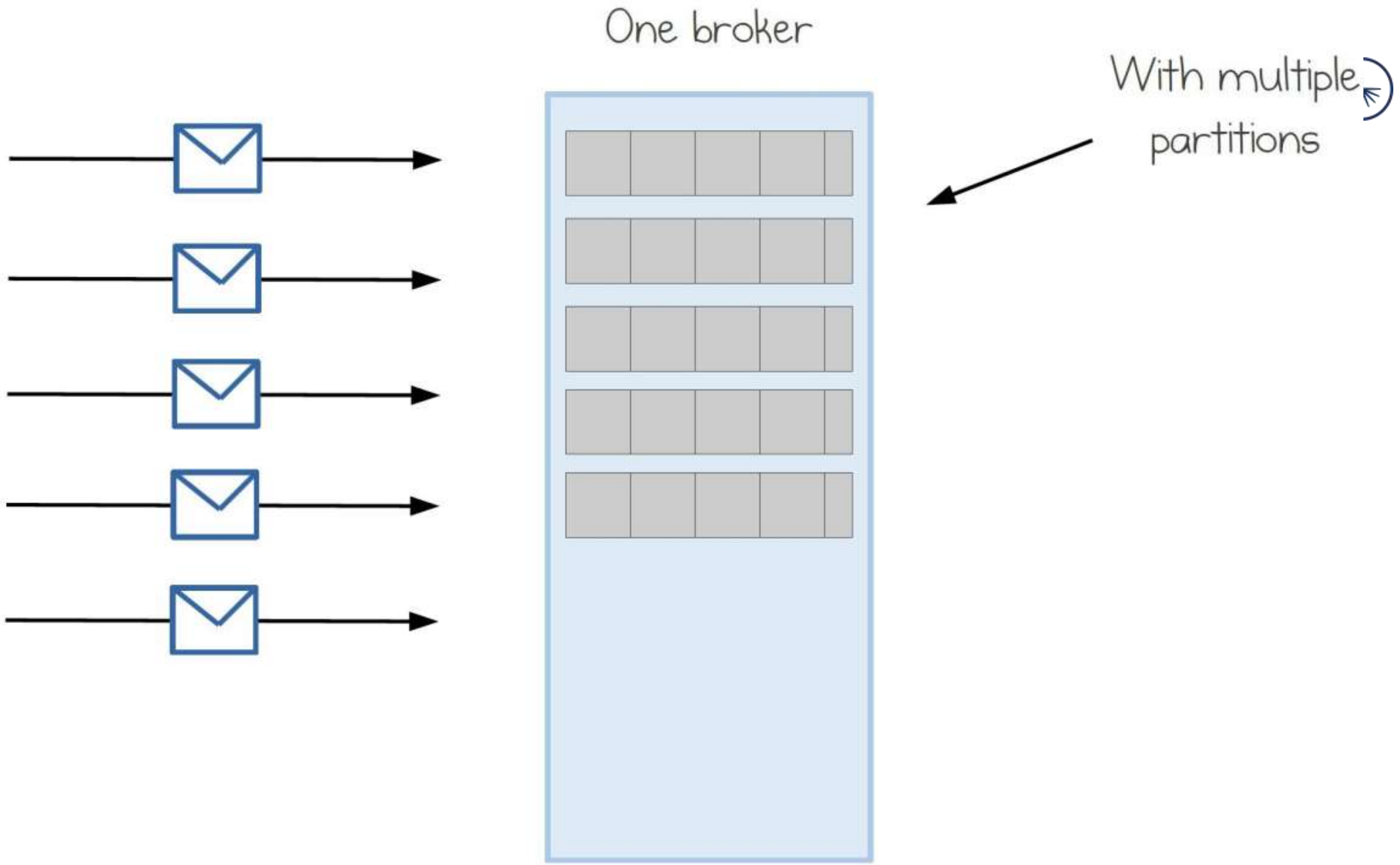
…which is true for any data systems

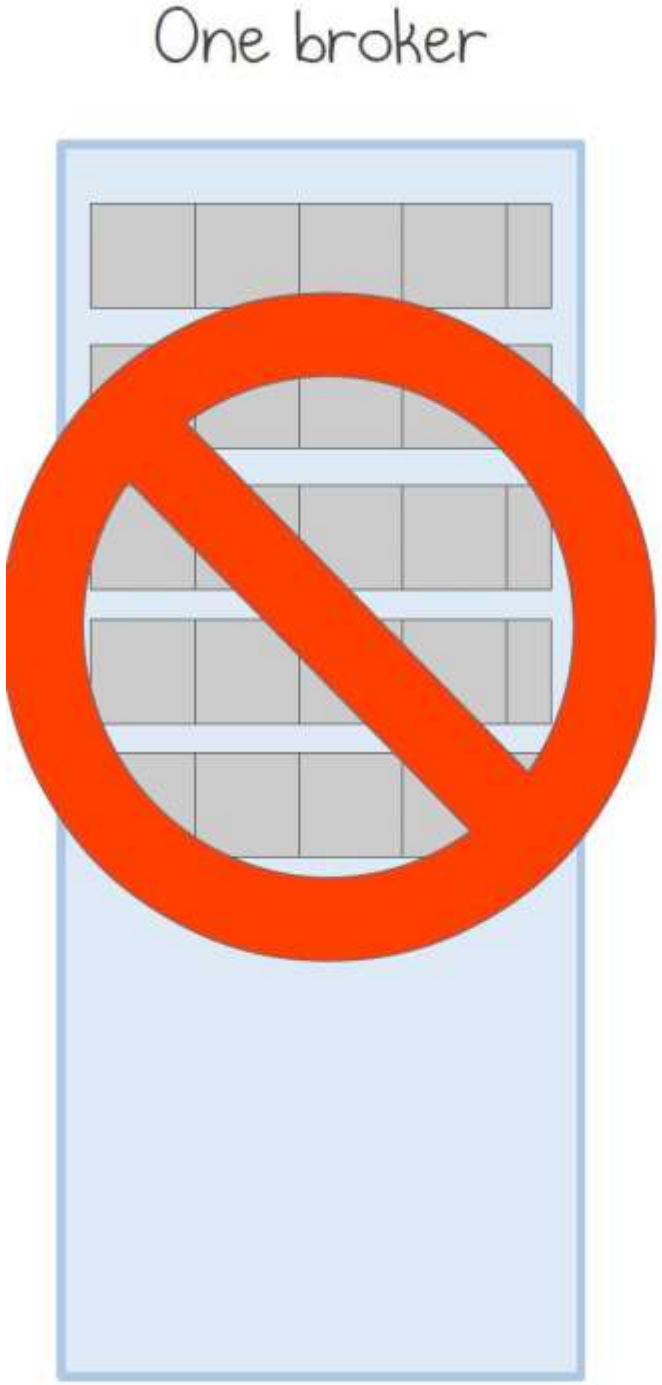
Not thinking about Durability

Data durability

If you didn’t think about it… it’s **not** durable!

OH MIGHTY KAFKA
PLEASE BRING OUR DATA BACK
imgflip.com


One broker
With multiple partitions



And my cluster is down....

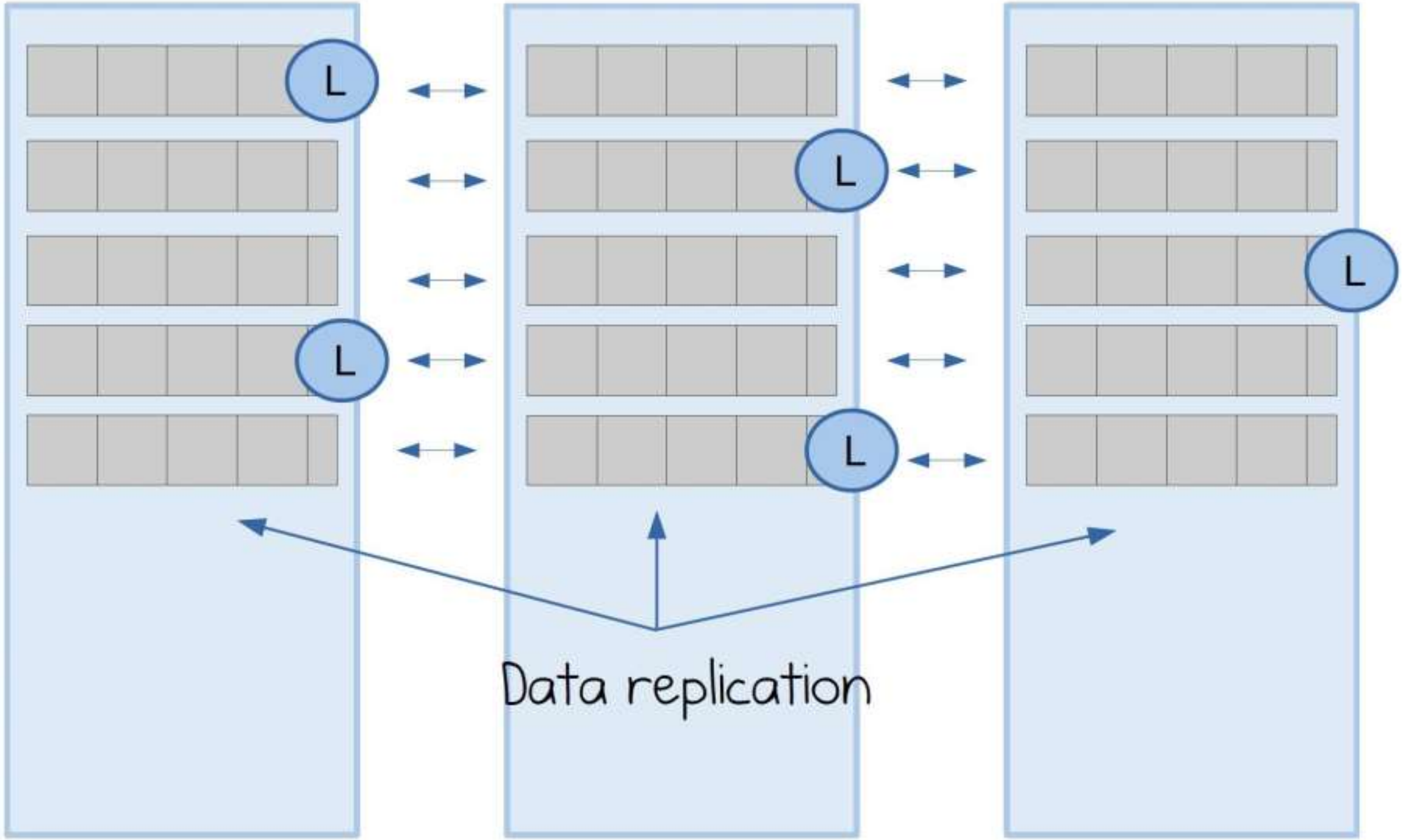
And you might have lost data!

Data durability

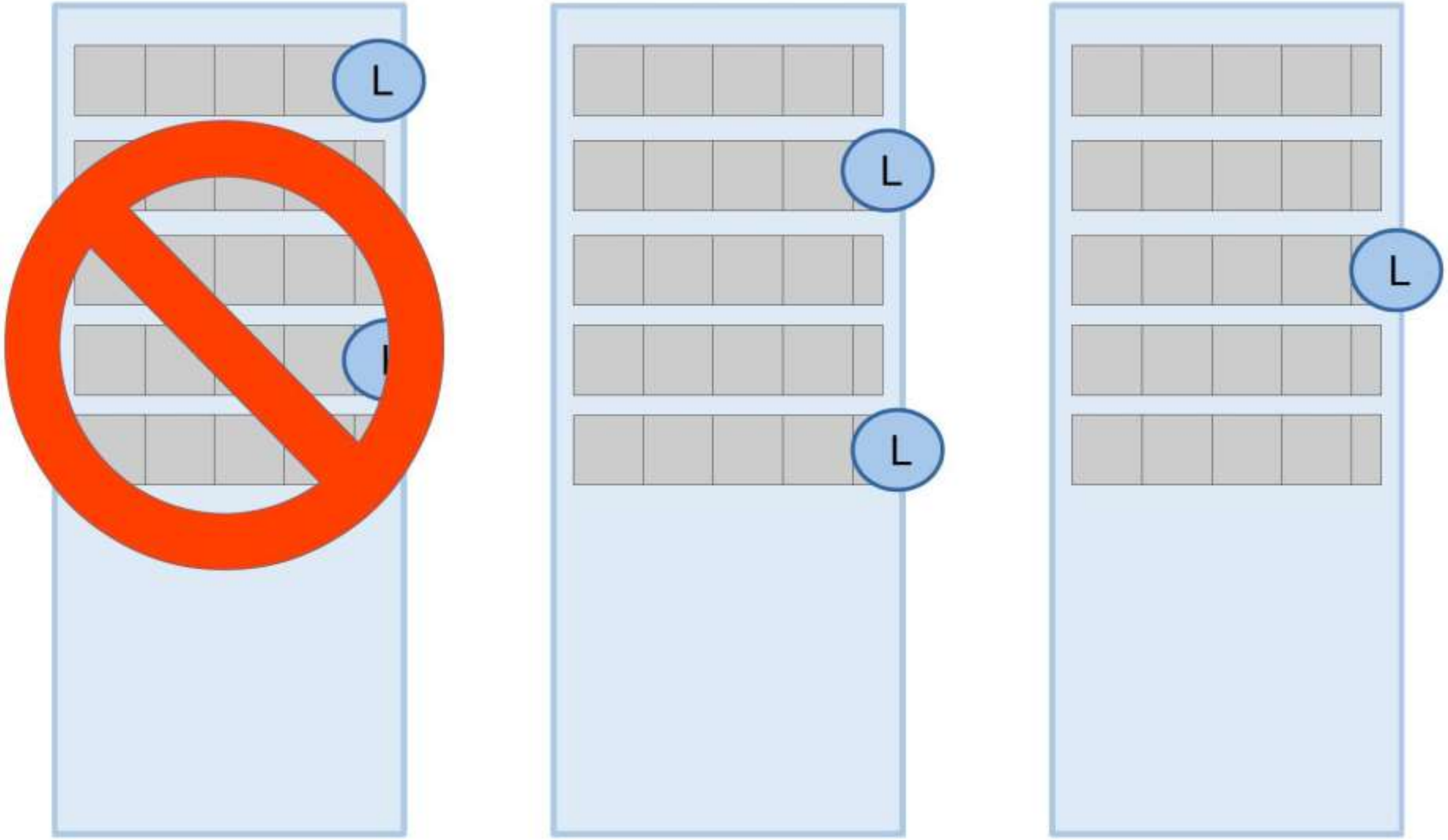
Kafka is **not** waiting for a disk flush by default.

Durability is achieved through **replication**.

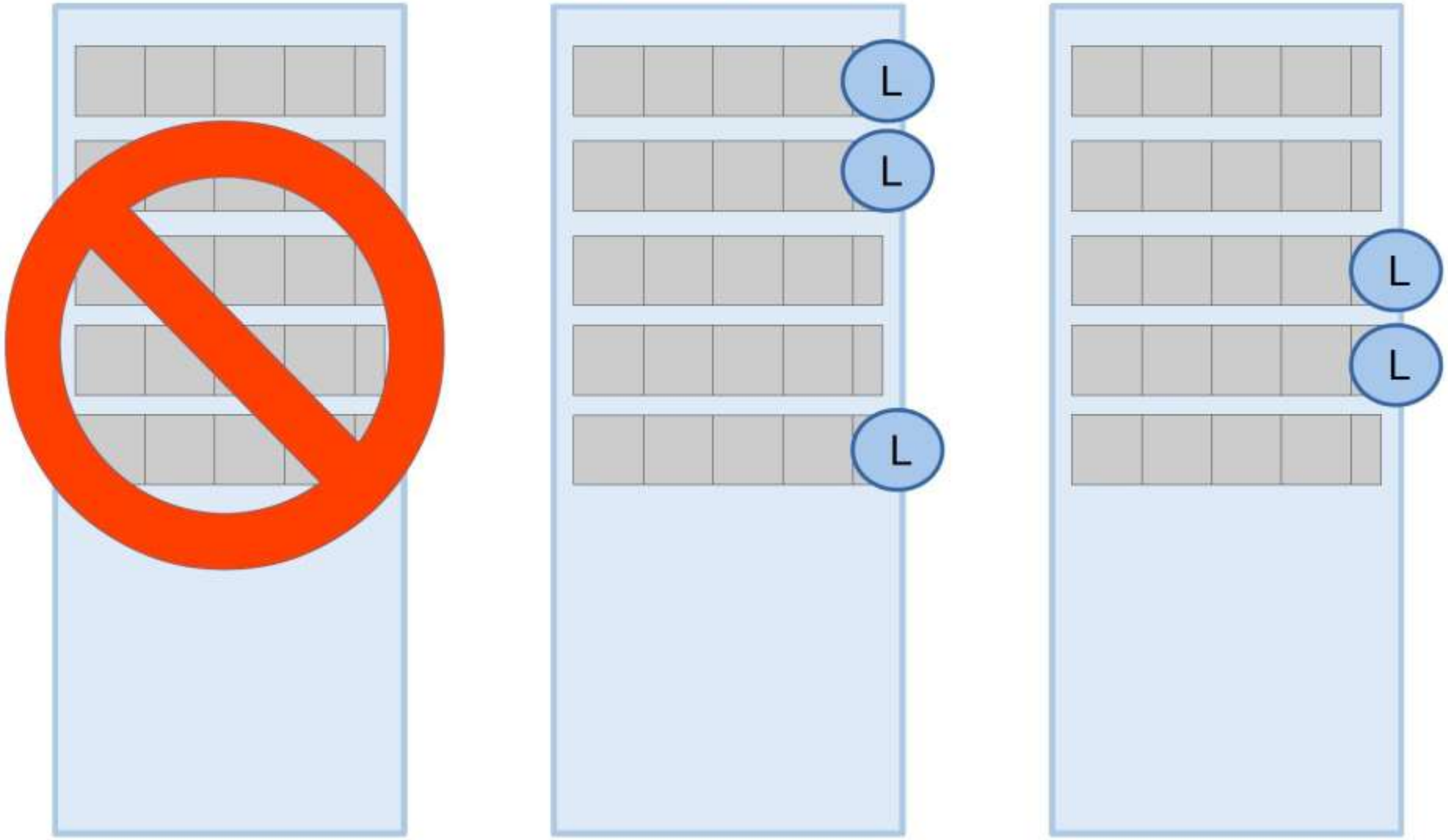
Cluster for High Availability



Cluster for High Availability



Leaders are automically moved

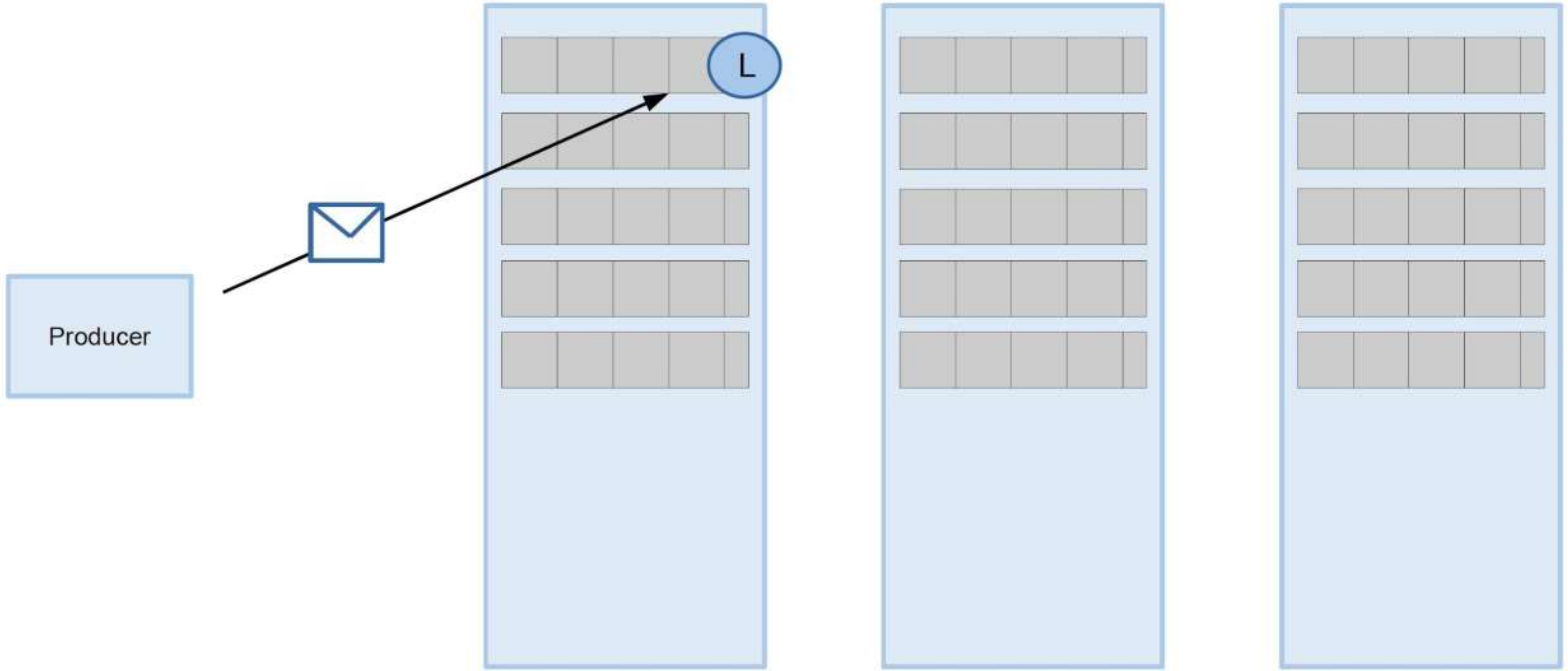


SO IS MY DATA SAFE?
imgflip.com

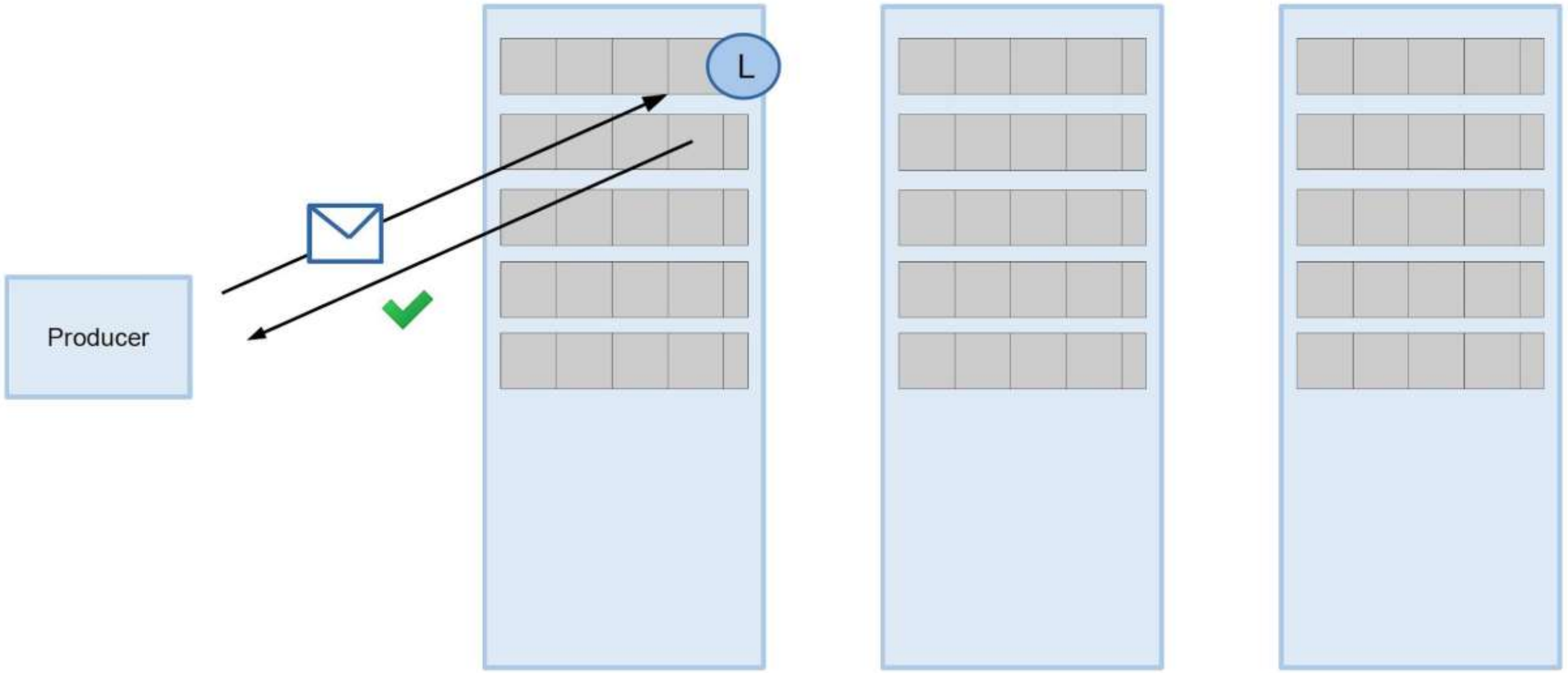
Data durability

It depends on your **configuration**...

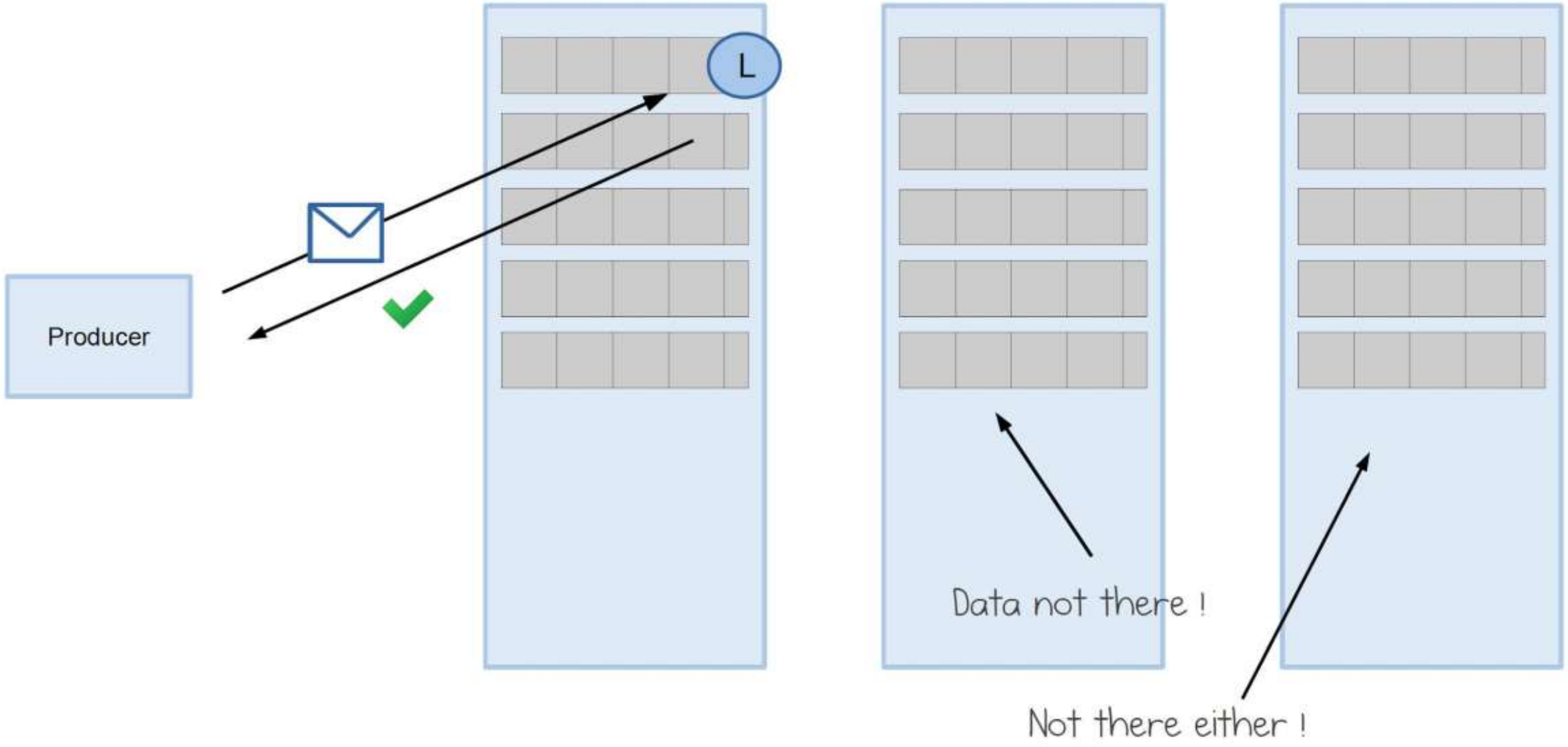
When will the cluster acknowlege ?

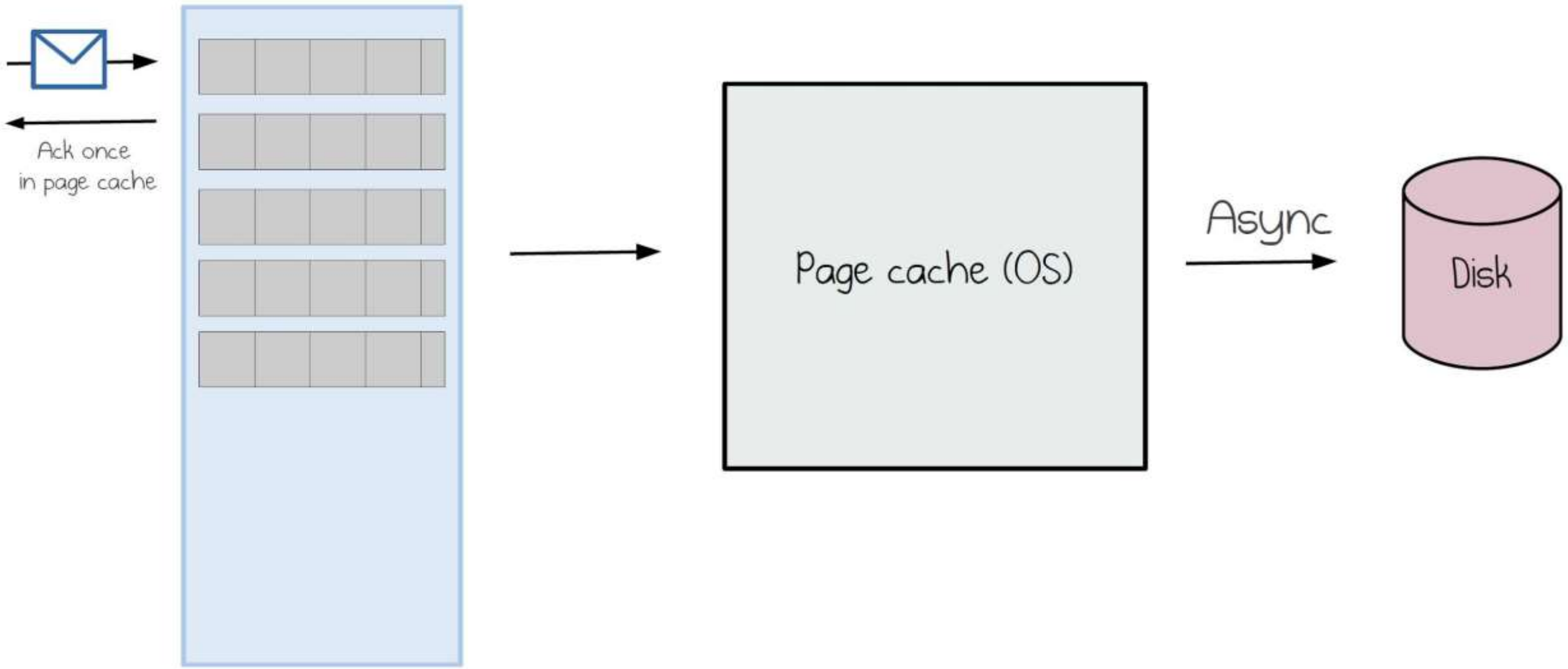


As soon as it is on the page cache of the leader...



As soon as it is on the page cache of the leader...



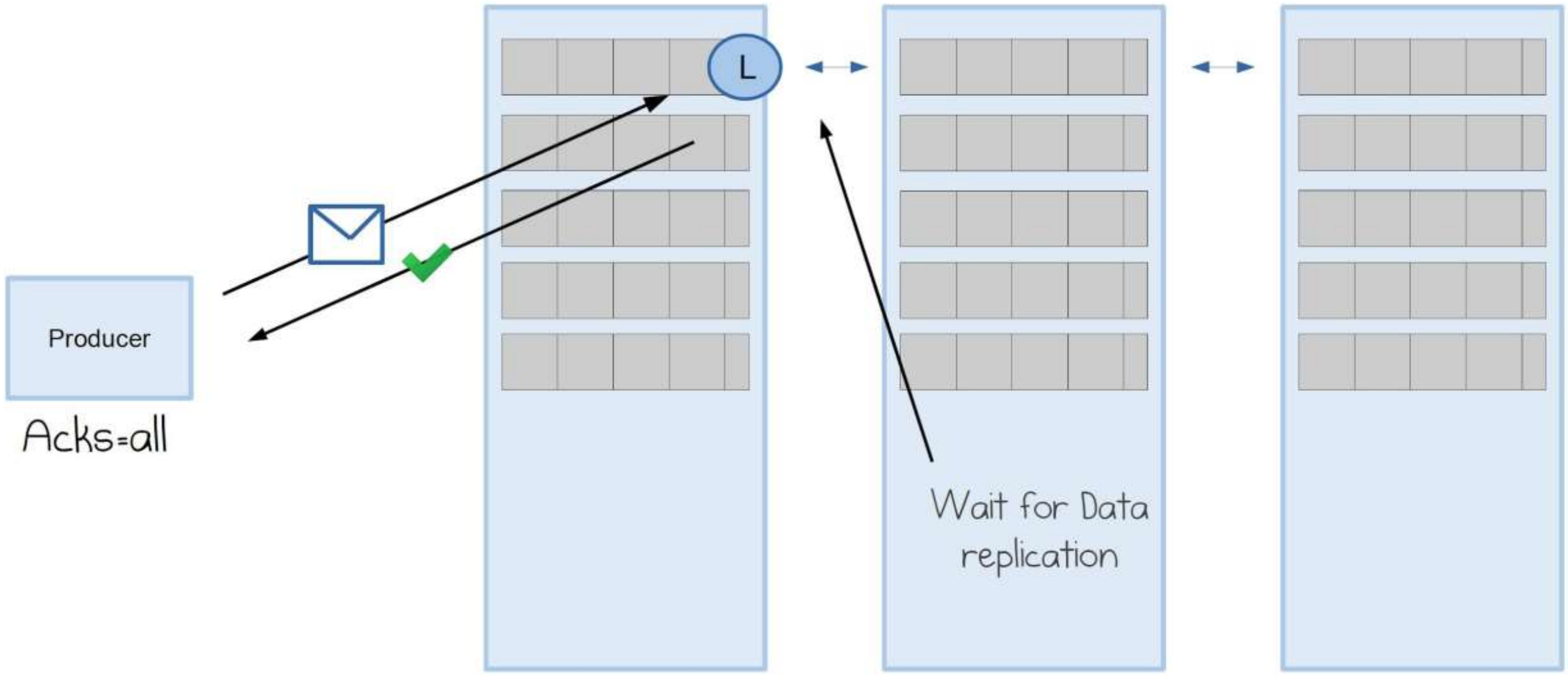

Ack once in page cache
Page cache (OS)
Async
Disk

Data durability

acks=1 (default value) good for **latency**

acks=all
good for **durability**

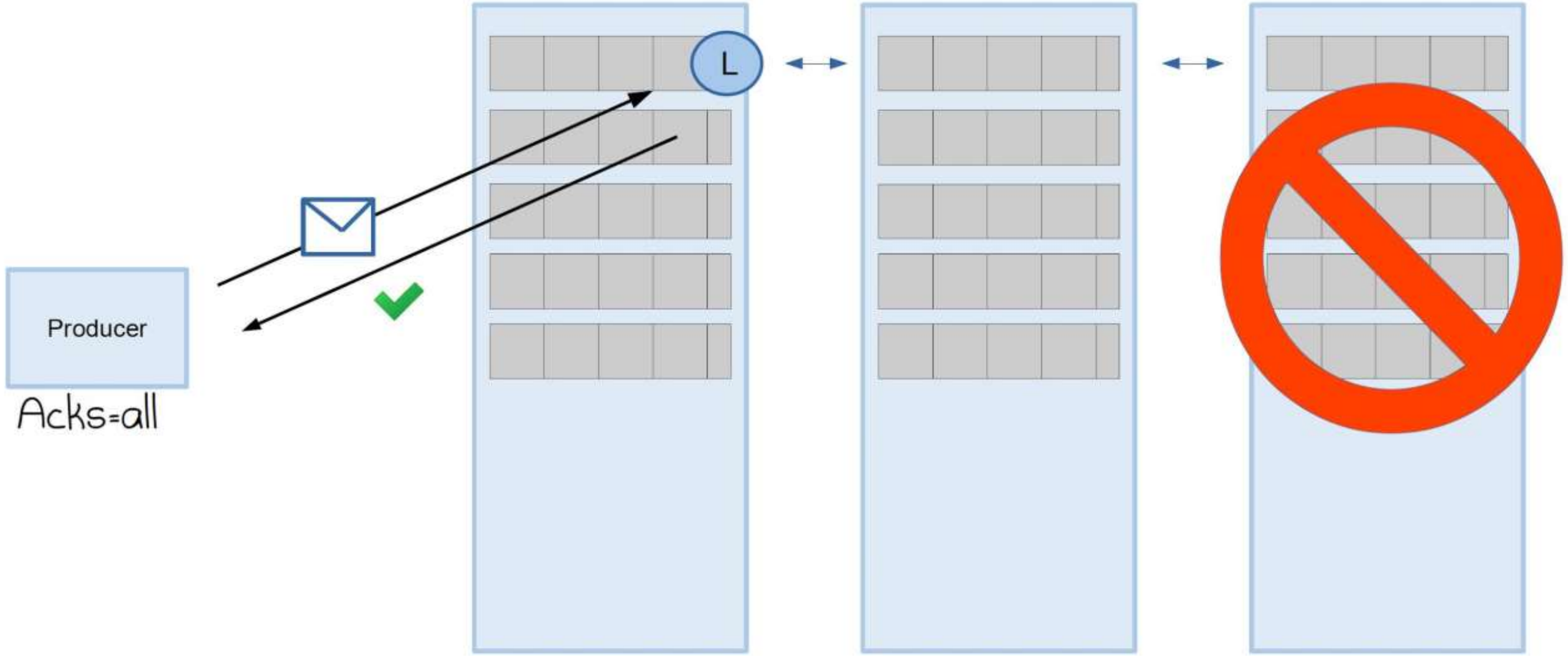
Replication before acknolowdging



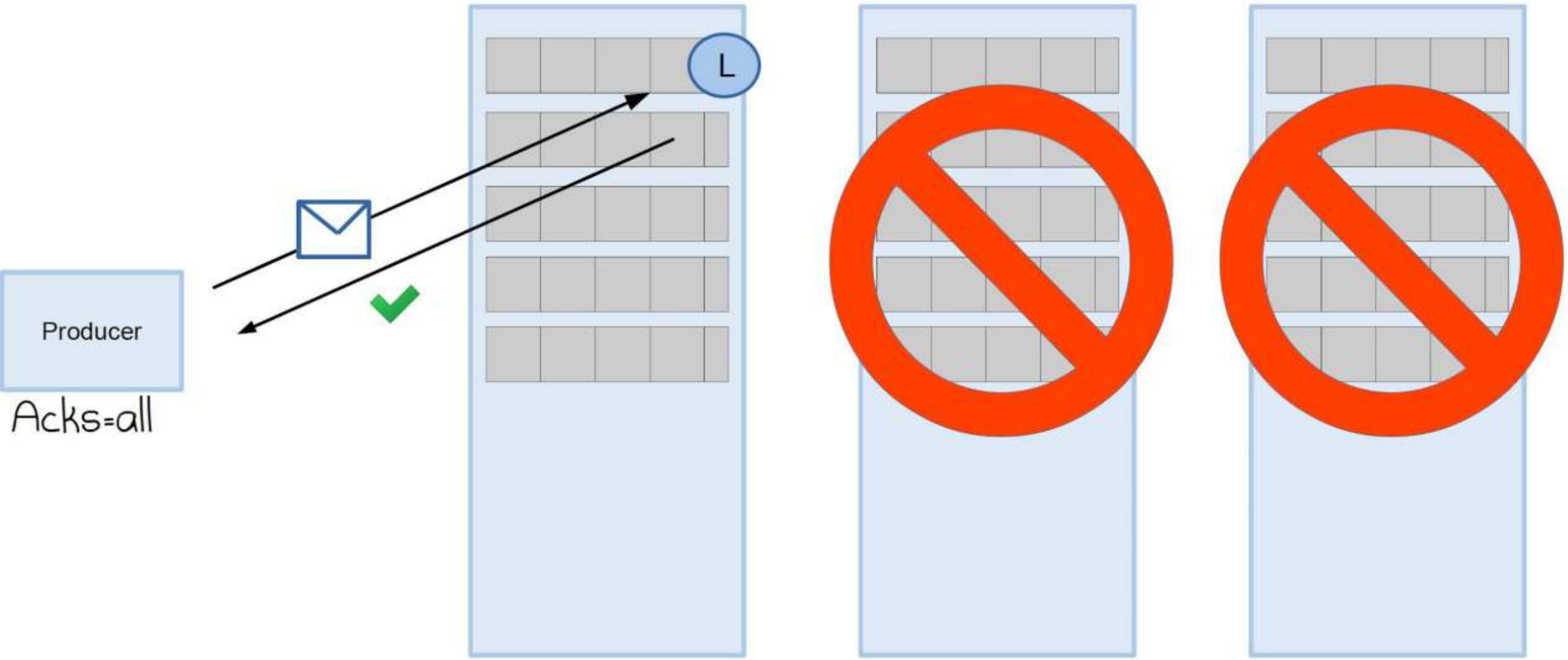
acks=all

The leader will wait for the full set of **in-sync replicas** to acknowledge the record.

But only to the In-Sync Replicas...



... which could be only one server



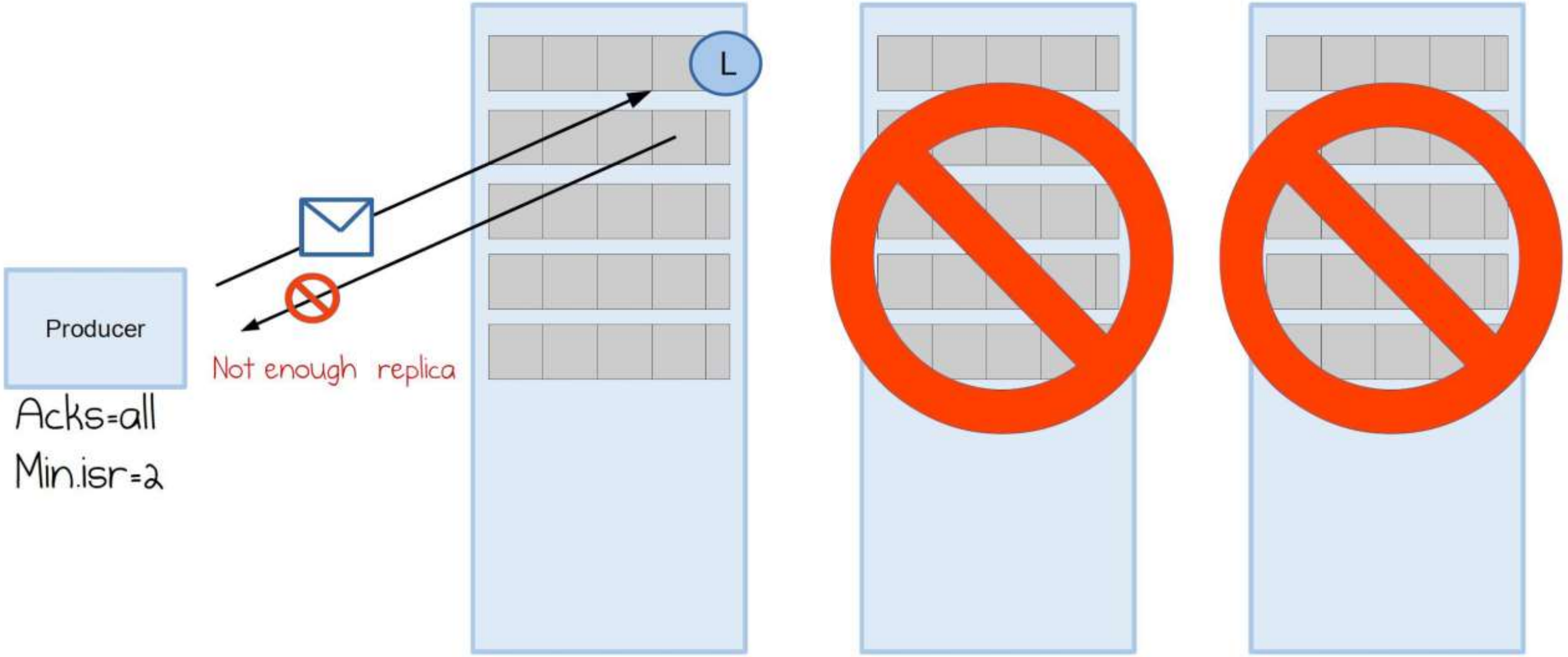
OH NO
MY DATA IS
ONLY ON ONE SERVER
imgflip.com

min.insync.replicas

minimum number of replicas that must acknowledge.

Default is 1.

... which could be only one server



Data Durability while Producing ?

Tune it with the parameters **acks** and **min.insync.replicas**

defaults

The default values are optimized for availability & latency.

If durability is more **important**, tune it!

Deploying on multi datacenters ?

Dubli
Lille

Multi-dc

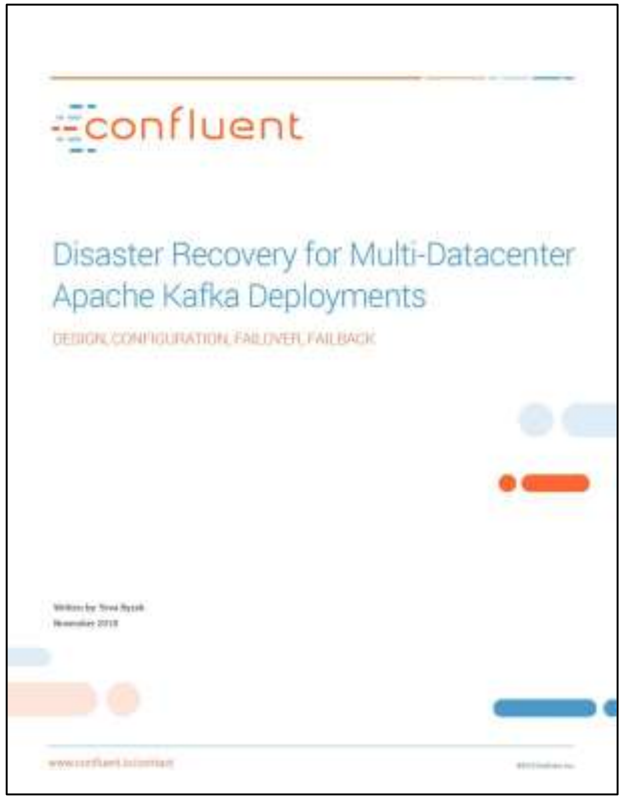
It’s quite complicated...

It’s easy to make it wrong on many levels.

It could be a **3h talk**.

Multi-dc

[Disaster recovery for multi datacenter]


confluent

Disaster Recovery for Multi-Datacenter Apache Kafka Deployments

DESIGN, CONFIGURATION, FAILOVER, FAILBACK

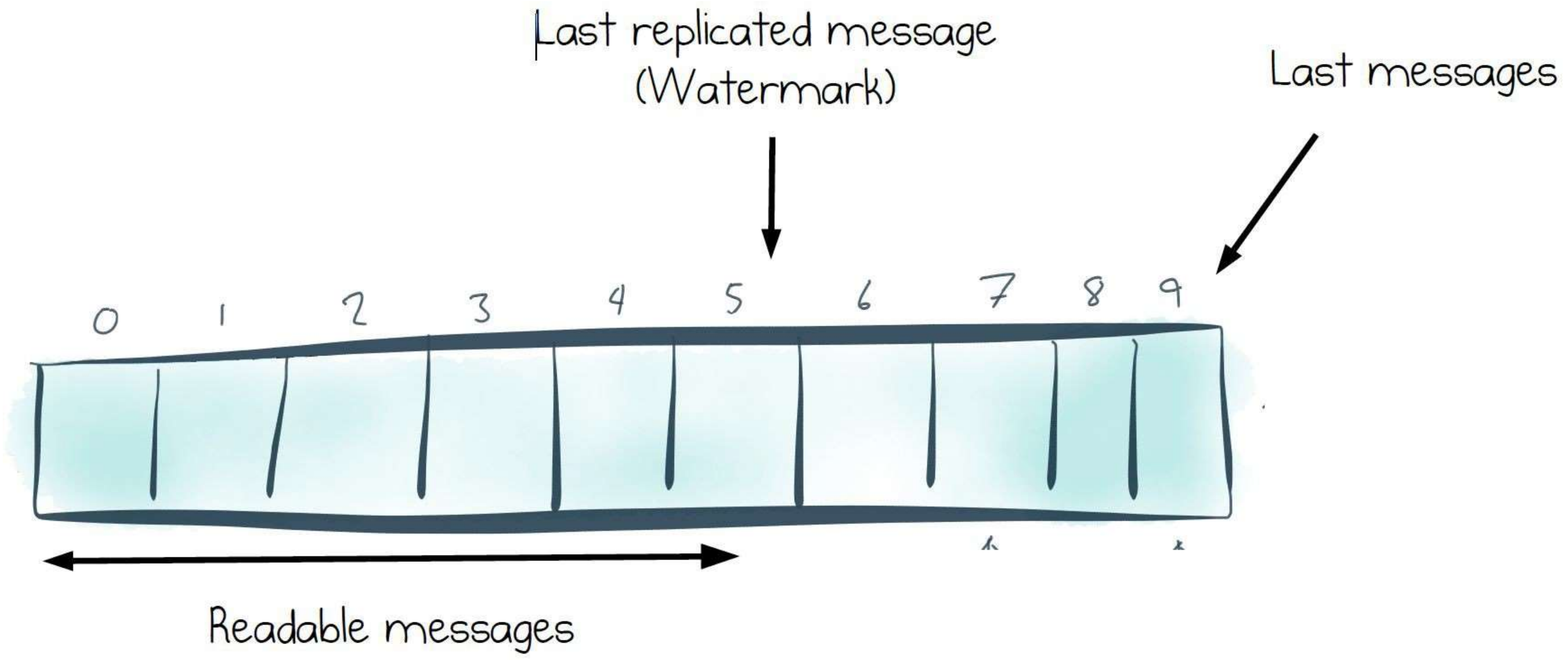
Written by Yeva Byzek
November 2018

www.confluent.io/contact

What about the consumers ?

consumers

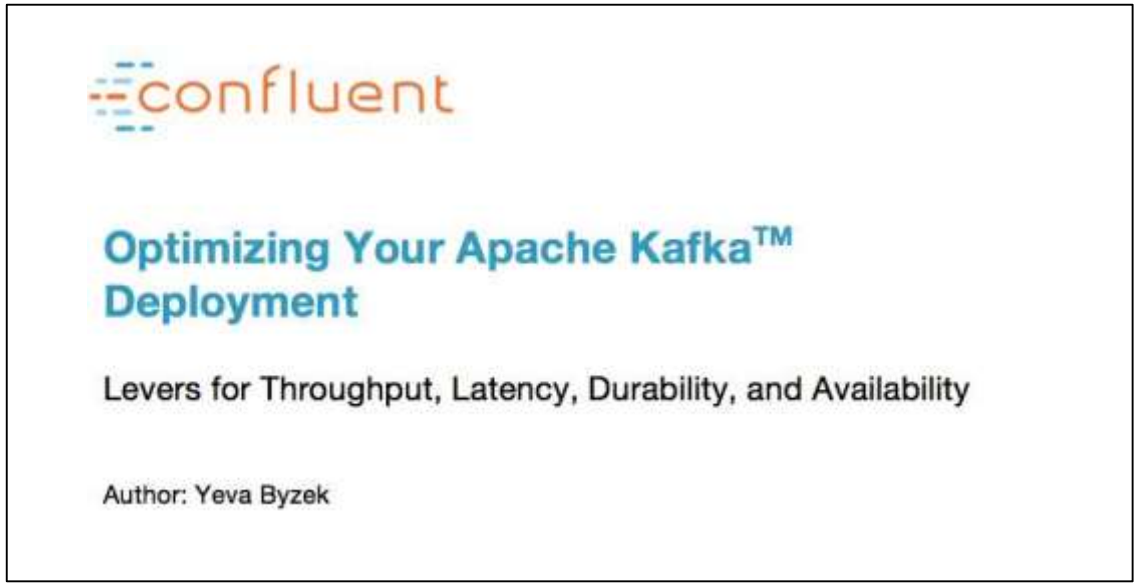
Consumer can read only **committed** data.


Last replicated message
(Watermark)
Last messages
0
1
2
3
4
5
6
7
8
9
Readable messages

Think about data durability and decide of the best trade-off for you

Throughput, latency, durability, availability

[Optimizing your Apache Kafka deployment]

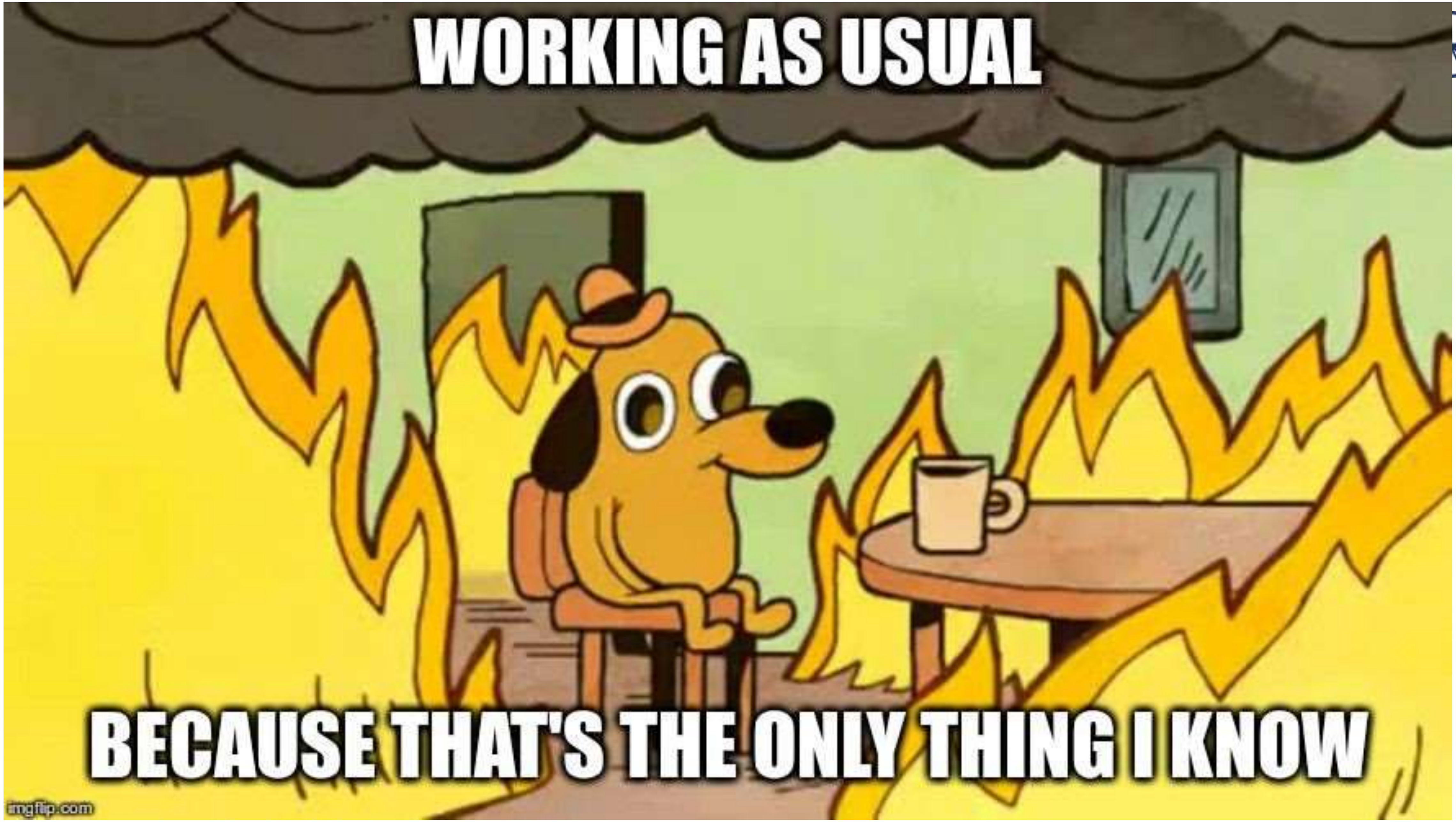

confluent

Optimizing Your Apache Kafka™ Deployment

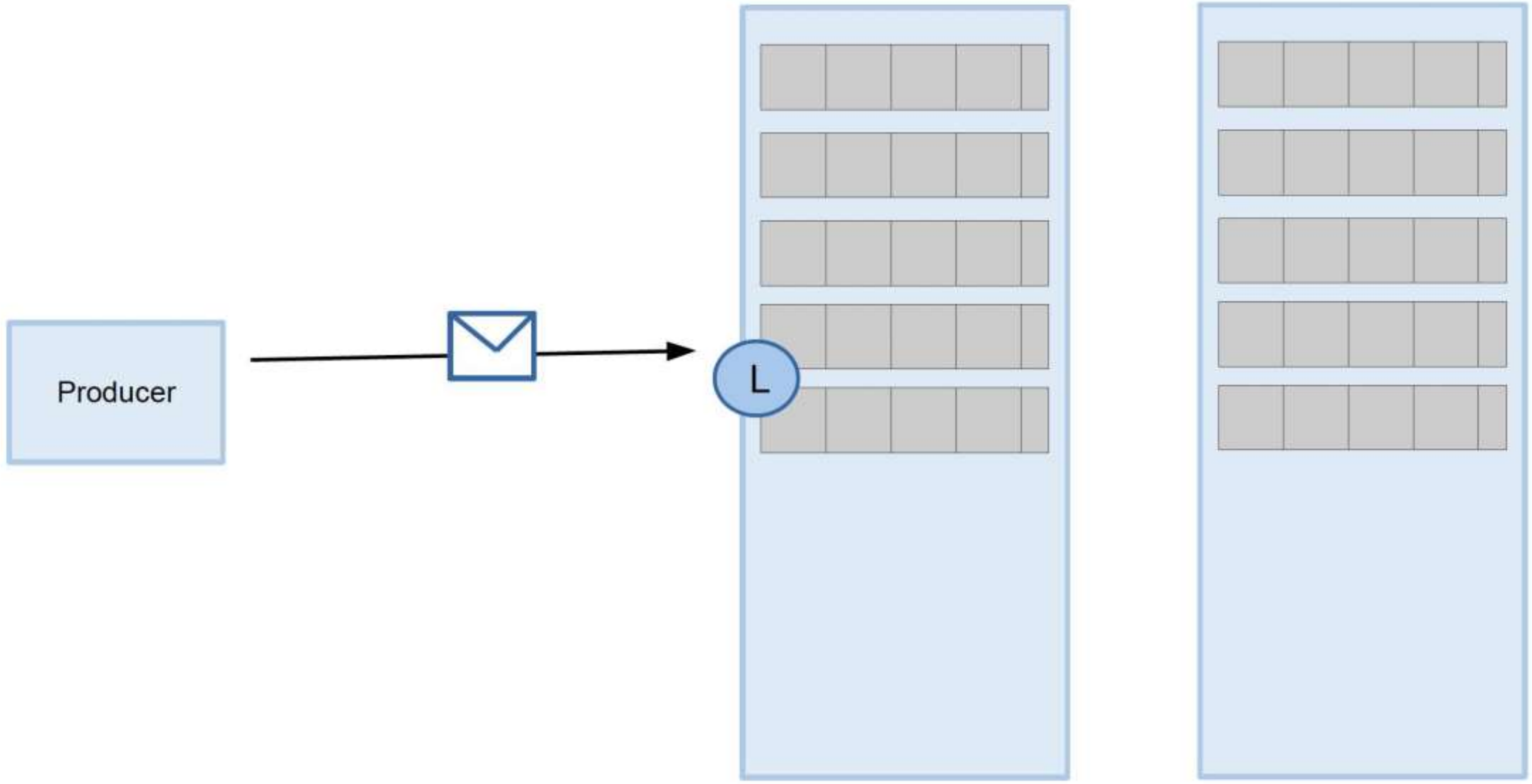
Levers for Throughput, Latency, Durability, and Availability

Author: Yeva Byzek

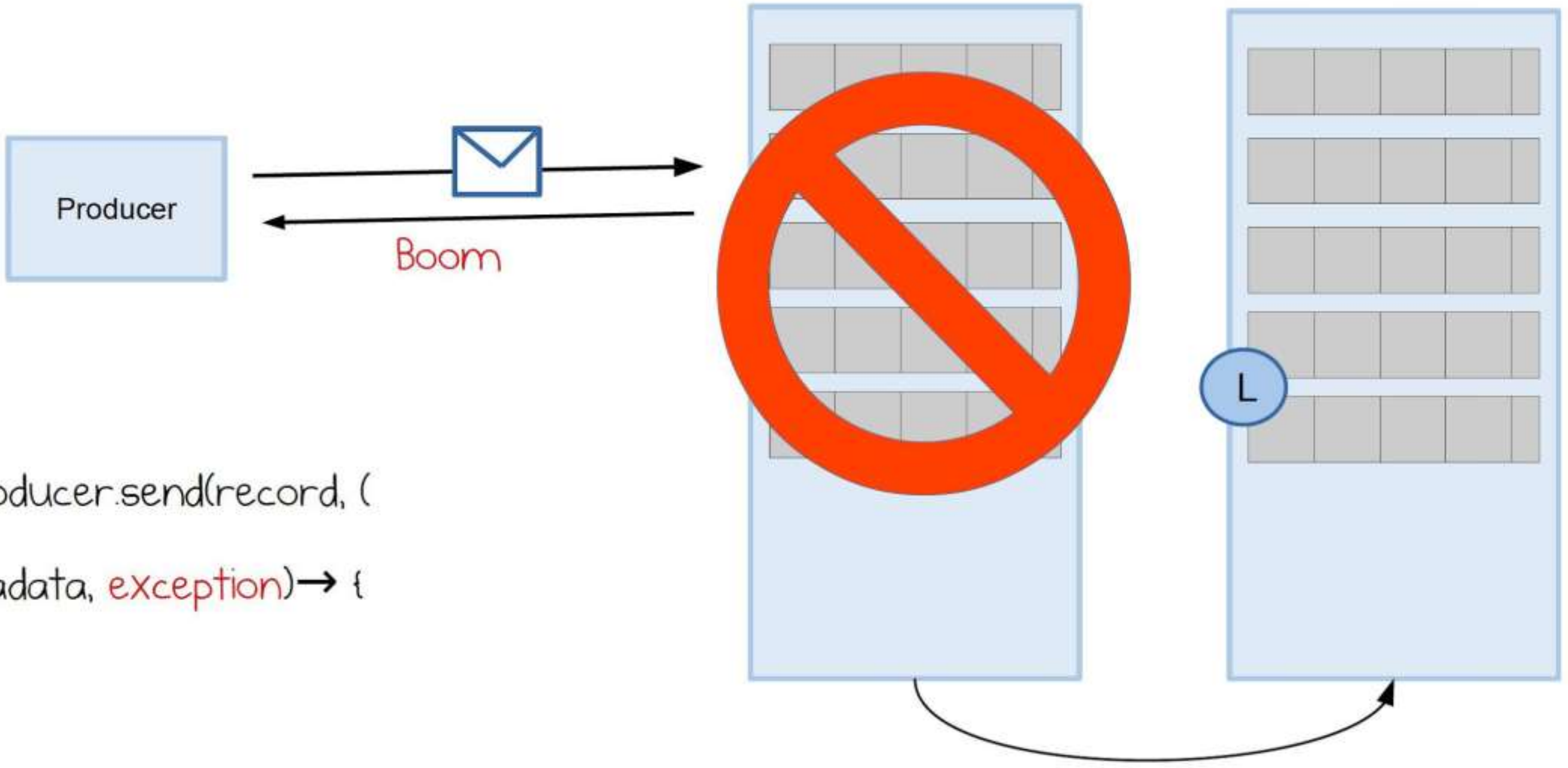
Focusing only on the happy path


WORKING AS USUAL
BECAUSE THAT'S THE ONLY THING I KNOW
imgflip.com

What's happening in case of issue ?



What's happening in case of issue ?



```
kafkaProducer.send(record, (
  (metadata, exception)→ {
     ...
  })
)
```

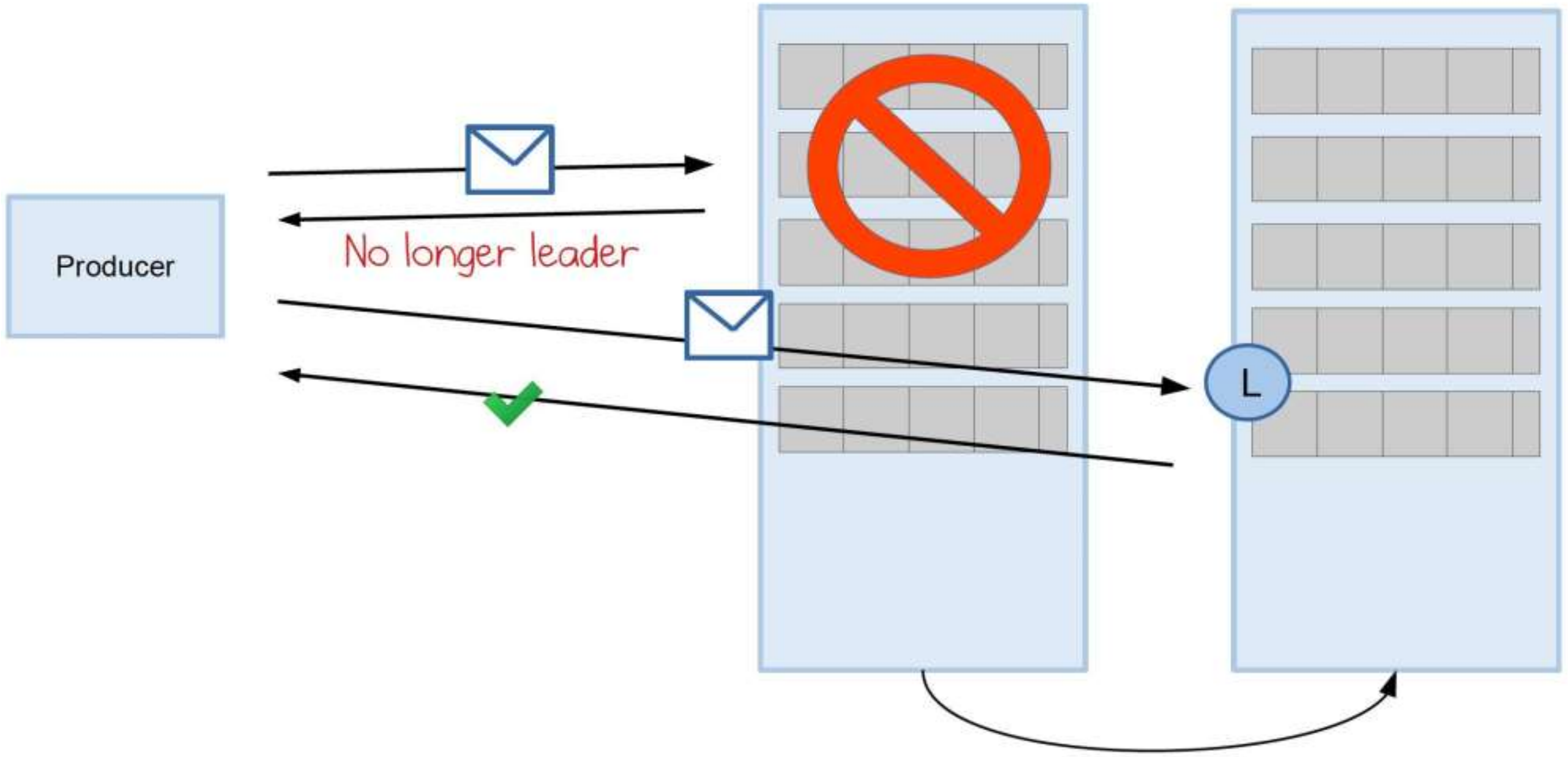
The leader moved to a different broker

retries

It will cause the client to resend any record whose send fails with a potentially transient error.

Default value : 0

What's happening in case of issue with retry ?



The leader moved to a different broker

retries

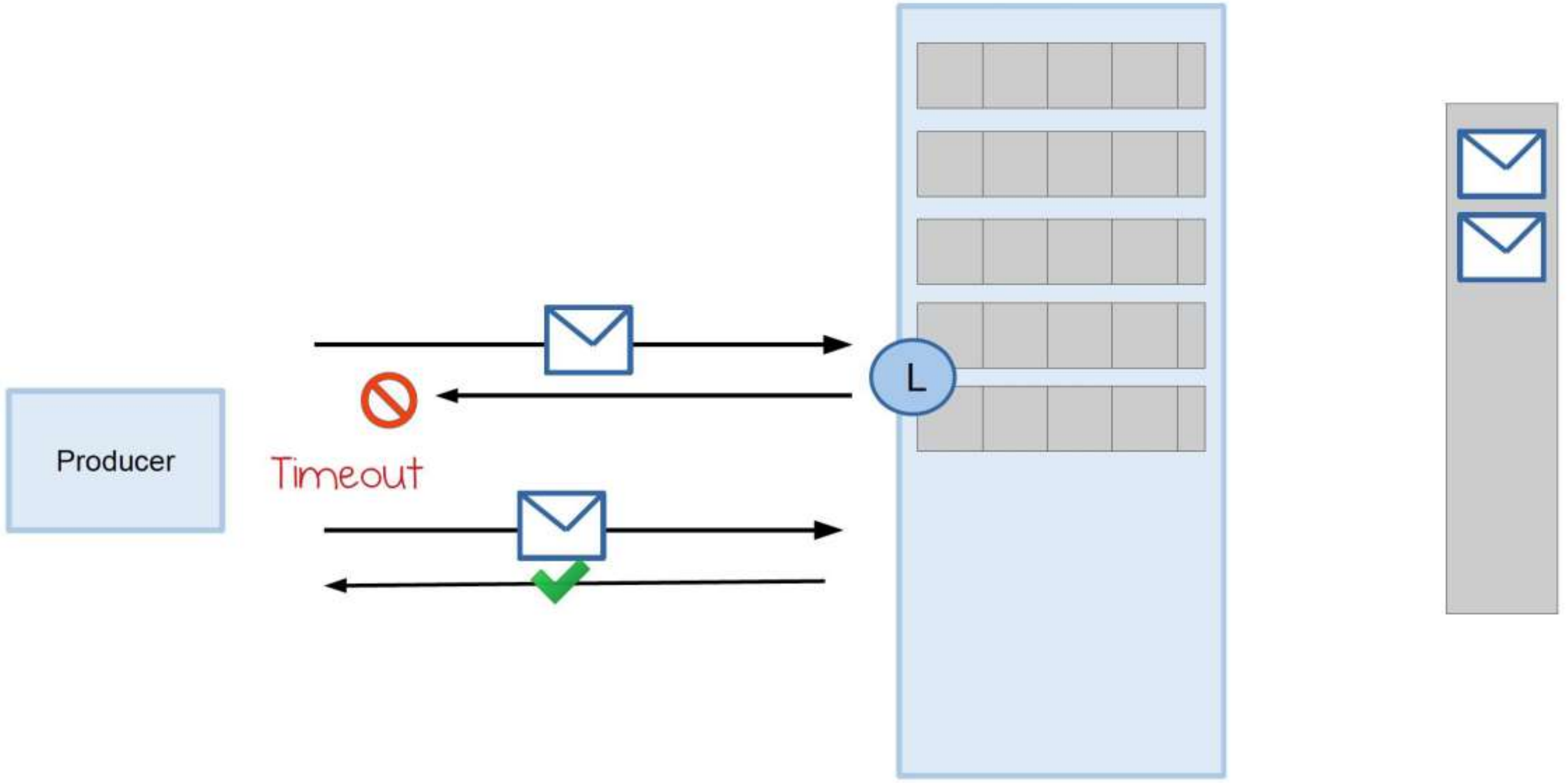
Use built in retries !

Bump it from 0 to **infinity**!

retries

But you are exposed to a different kind of issue…

Message duplication

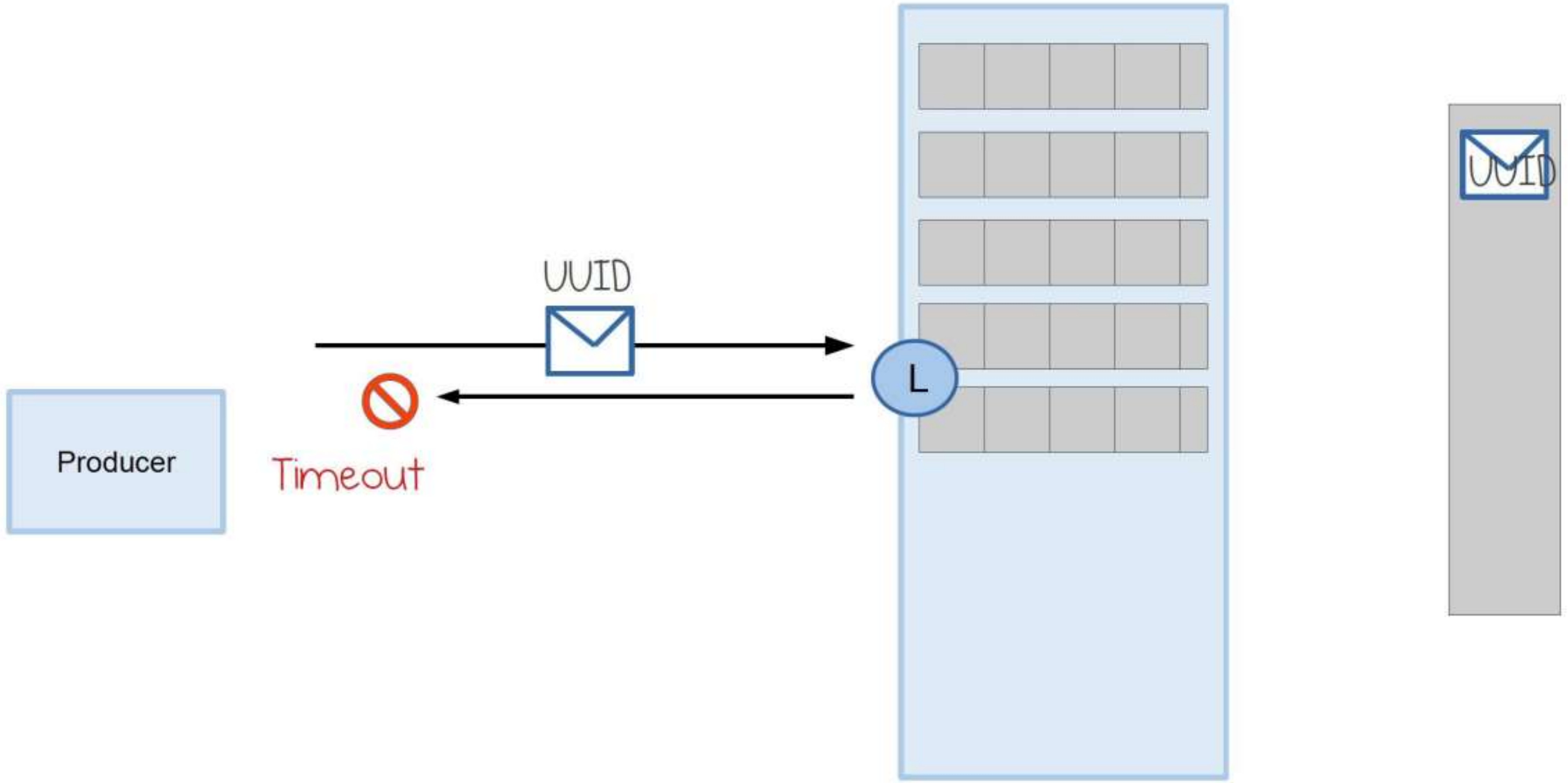


enable.idempotence

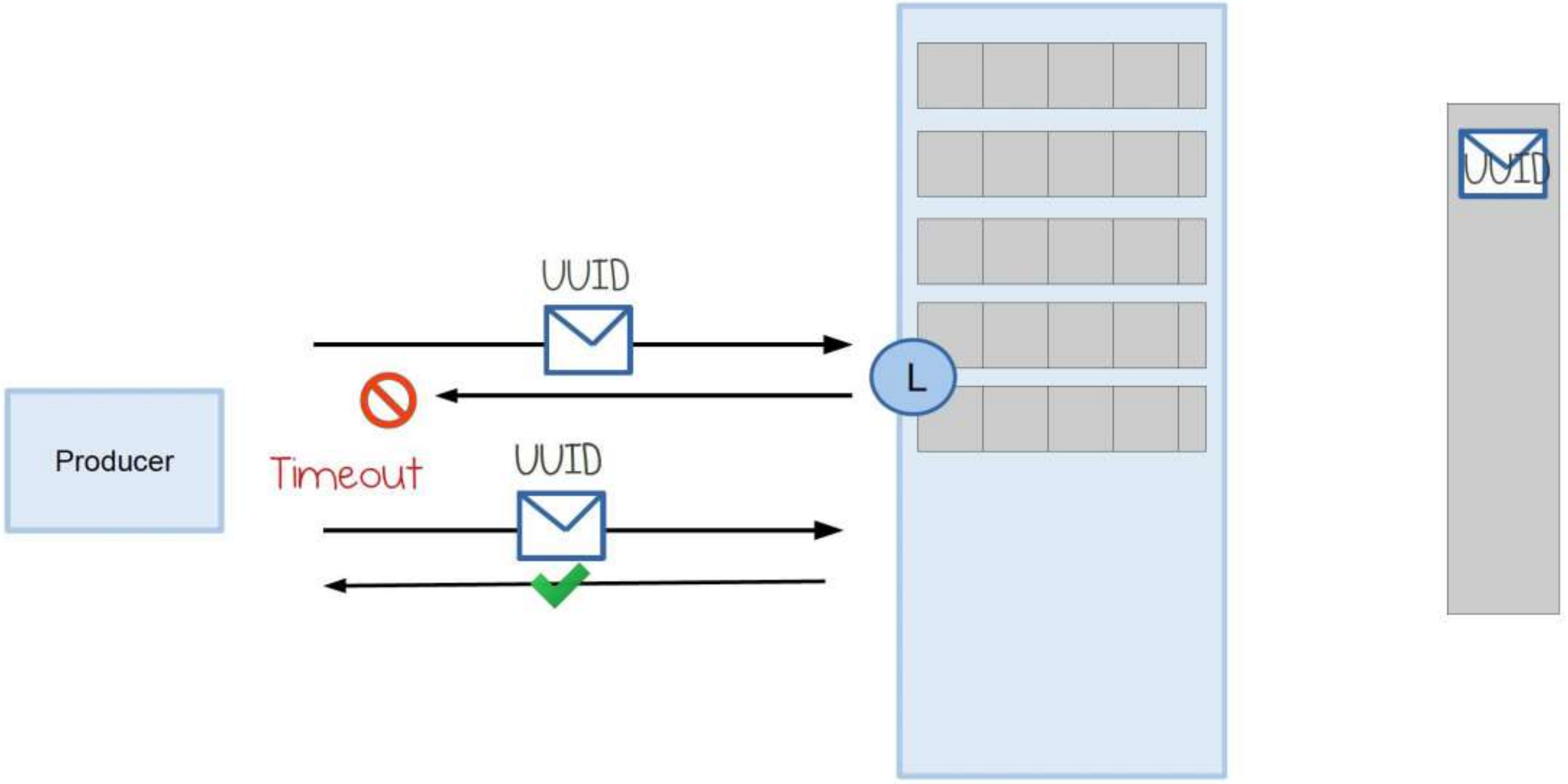
When set to 'true', the producer will ensure that exactly **one** copy of each message is written.

Default value: **false**

With Retries and Idempotency



With Retries and Idempotency



Use built in idempotency!

But it does not save you from

- Managing exception and failure
- Developing Idempotent consumer

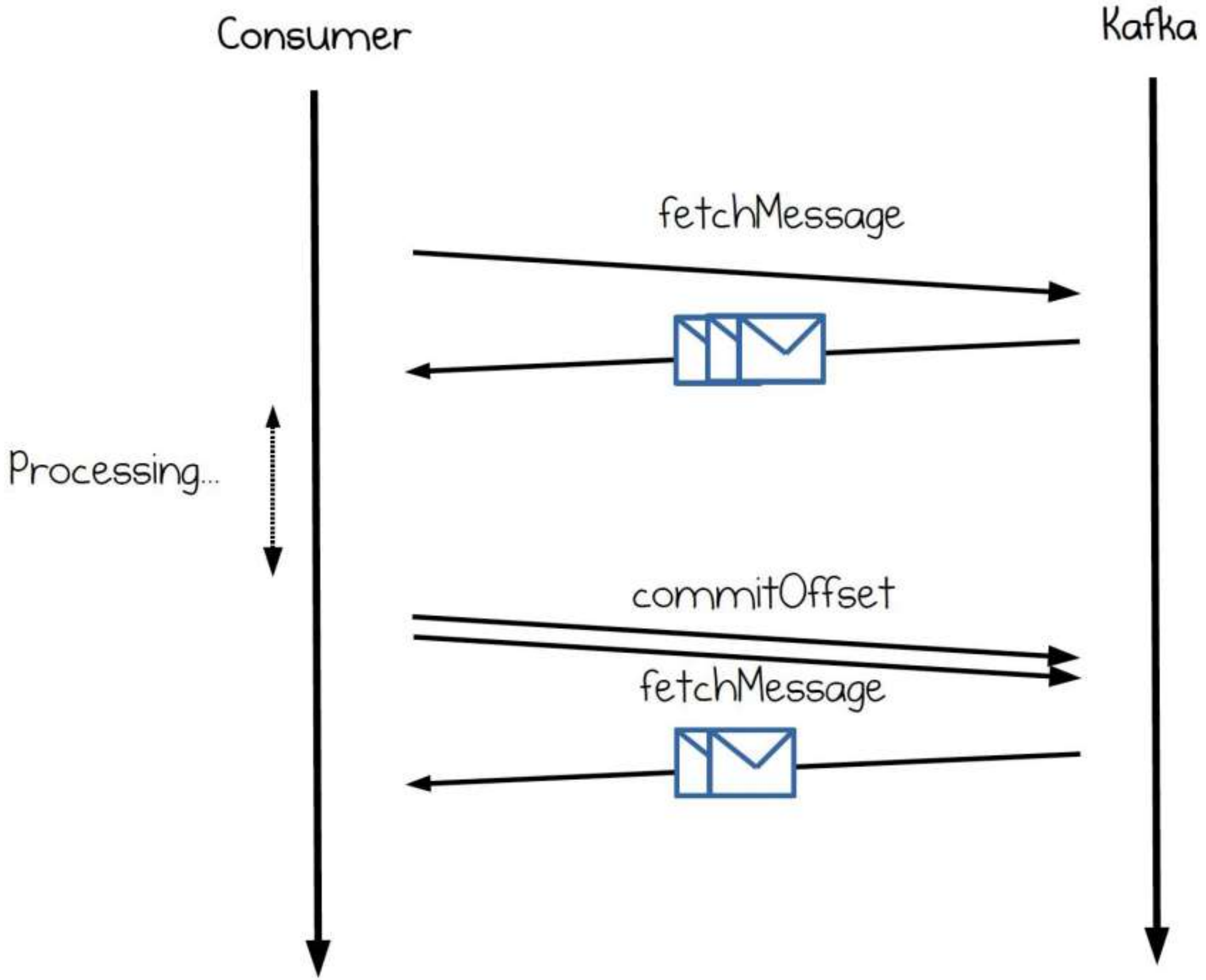
No Idempotent consumer

IF YOU'RE USING KAFKA AND YOUR MESSAGES AREN'T IDEMPOTENT
INSTRUCTOR
YOU'RE GONNA HAVE A BAD TIME
memegenerator.net

At *least* once (default)

At *most* once

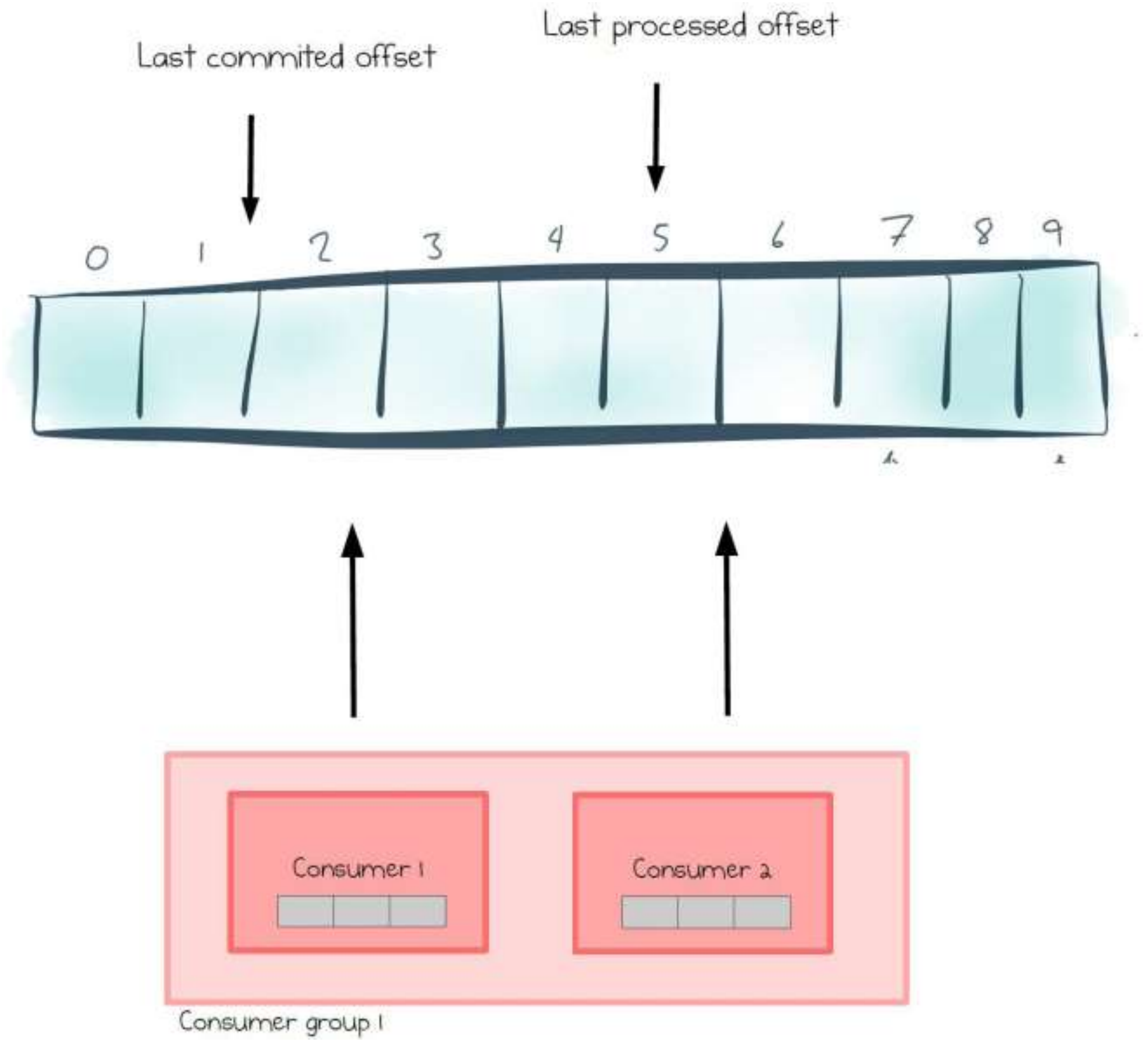
Exactly Once

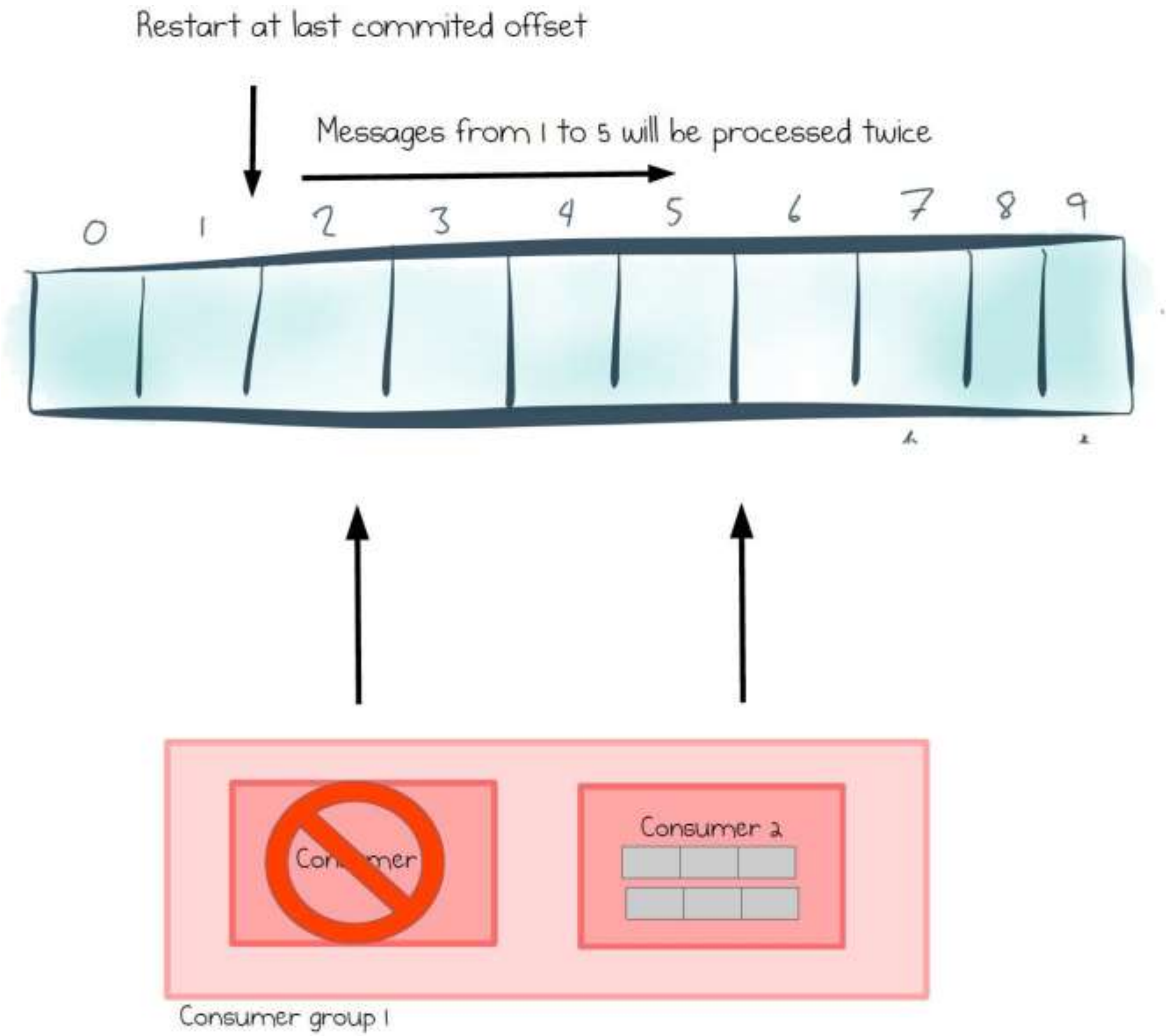

Consumer
Kafka
fetchMessage
Processing...
commitOffset
fetchMessage

```
while (this.getRunning()) {
    var consumerRecords = consumer.poll(1000);
    for (var record: records) {
        /*
         * Doing my business logic here
         */
    }
}
```

```
while (this.getRunning()) {
    var consumerRecords = consumer.poll(1000);
    for (var record: records) {
        /*
         * Doing my business logic here
         */
    }
}
```

Poll might commit the consumer offset (by default every 5 seconds)


Last commited offset
Last processed offset
0
1
2
3
4
5
6
7
8
9
Consumer 1
Consumer 2
Consumer group 1


Restart at last commited offset
Messages from 1 to 5 will be processed twice
0
1
2
3
4
5
6
7
8
9
Con mer
Consumer 2
Consumer group 1

```
while (this.getRunning()) {
    var consumerRecords = consumer.poll(1000);
    for (var record: records) {
        /*
         * Doing my business logic here
         */
        consumer.commitSync(…)
    }
}
```

commit

Manually committing aggressively...

Add a huge workload on Apache Kafka

```
while (this.getRunning()) {
    var consumerRecords = consumer.poll(1000);
    for (var record: records) {
        /*
         * Doing my business logic here
         */
        --------------------------------------------
        consumer.commitSync(...)
    }
}
```

What if you fail here ?

commit

Manually committing aggressively...

Does **not** provide exactly once semantic

Embrace at least once

Rely on Kafka Streams with Exactly Once !

No exception handling

YOU GET AN EXCEPTION
AND YOU GET AN EXCEPTION!
AND YOU GET AN EXCEPTION!!

```
Future<RecordMetadata> send(ProducerRecord<K, V> record);
```

```
Future<RecordMetadata> send(ProducerRecord<K, V> record,
                            Callback callback);
```

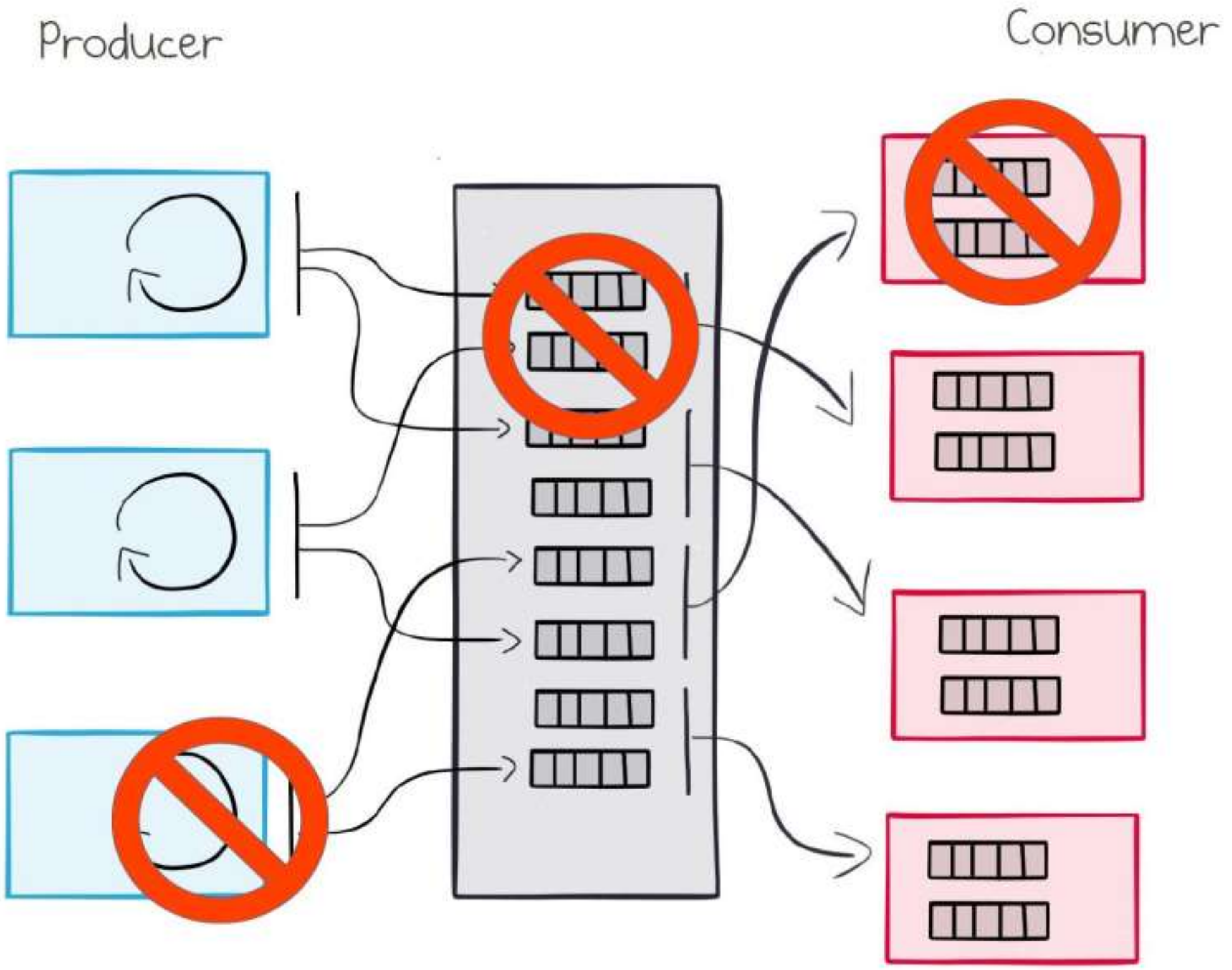
```
producer.send(record, (metadata, exception) -> {

});
```

error handling

We don’t expect the unexpected until the unexpected is expected.

What to do in case of an error ?



error handling

A message can **not** be processed

error handling

A message can **not** be processed

A message doesn’t have the **expected** schema

Retry

```
while (this.getRunning()) {
    try {
        var consumerRecords = consumer.poll(1000) ;
    } catch (Exception e) {
        Logger.error(e);
        continue ;
    }
    for (var record : consumerRecords) {
      try {
          /* Processing messages */
      } catch (Exception e) { … }
   }
}
```

Infinite retry

```
properties.put(ProducerConfig.RETRIES_CONFIG, Integer.MAX_VALUE);
```

Write to a dead letter queue and continue

```
while (this.getRunning()) {
    var consumerRecords = consumer.poll(1000) ;
    for (var record : consumerRecords) {
        try {
            /* Processing messages */
        } catch (Exception e) {
            producer.send(« dead-my-topic », new ProducerRecord(…)) ;
            Logger.error(e);
        }
    }
}
```

Ignore and continue

```
kafkaProducer.send(record, (
    (metadata, exception)→ {
        if (exception != null) {
            /* Something bad happened */
            /* But those are ephemaral data anyway */
            Logger.error(exception) ;
        }
    })
);
```

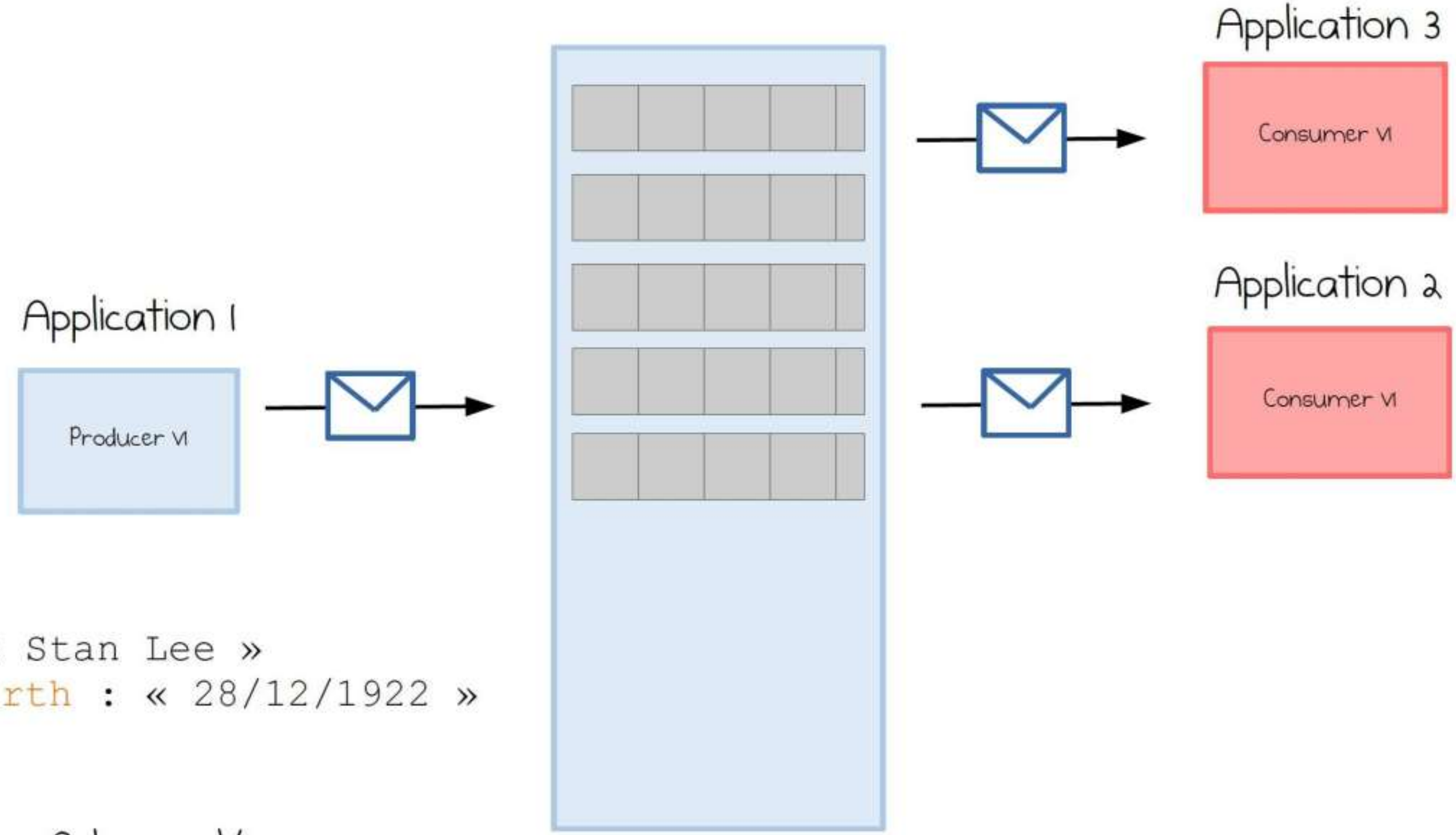
No silver bullet

Handle the exceptions !

https://eng.uber.com/reliable-reprocessing/

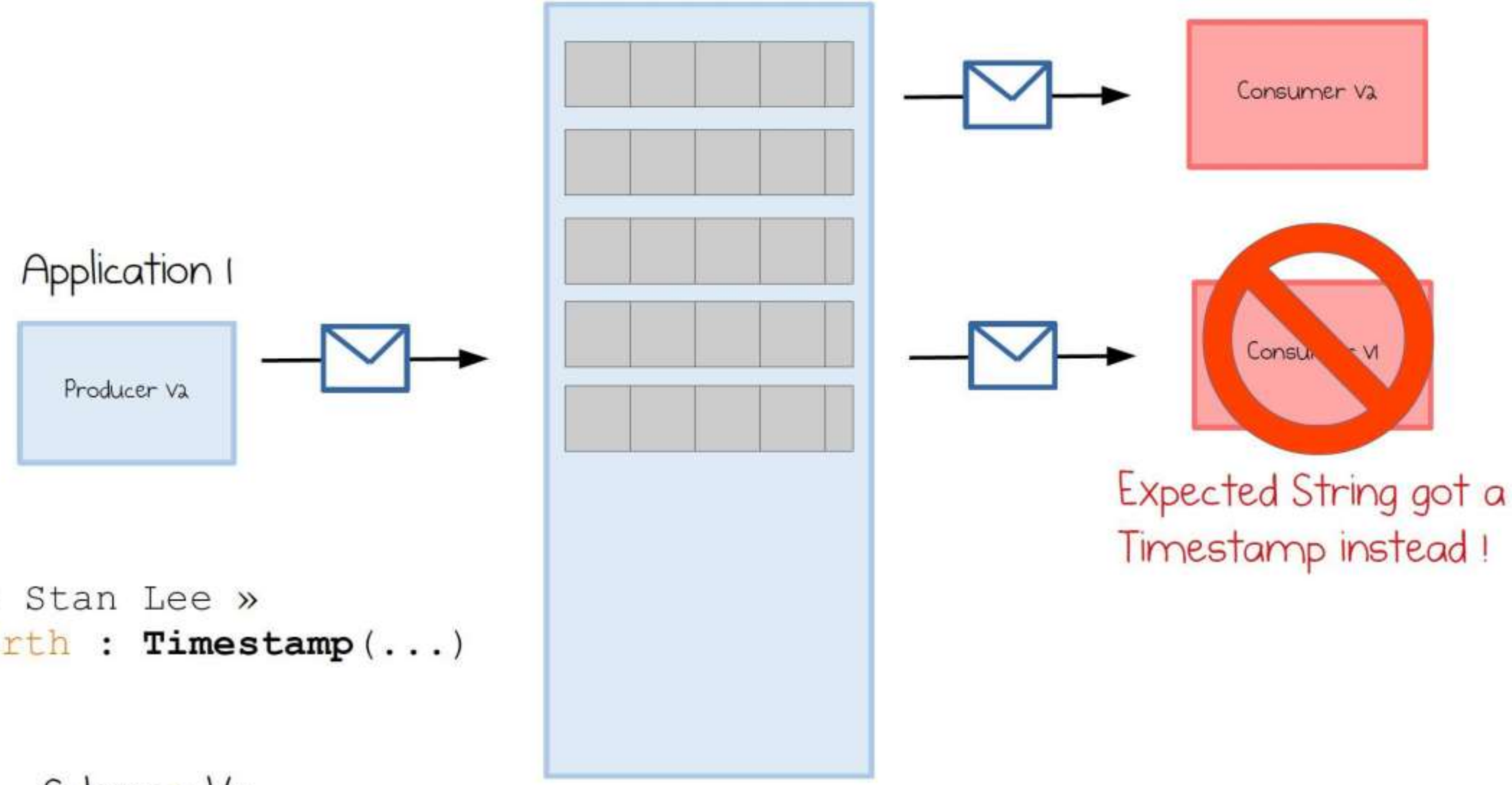
No data governance

TEAMWORK
IN A NUTSHELL



```
{
  Name : « Stan Lee »
  DateOfBirth : « 28/12/1922 »
}
```

Schema V1



```
{
  Name : « Stan Lee »
  DateOfBirth : Timestamp(...)
}
```

Schema V2

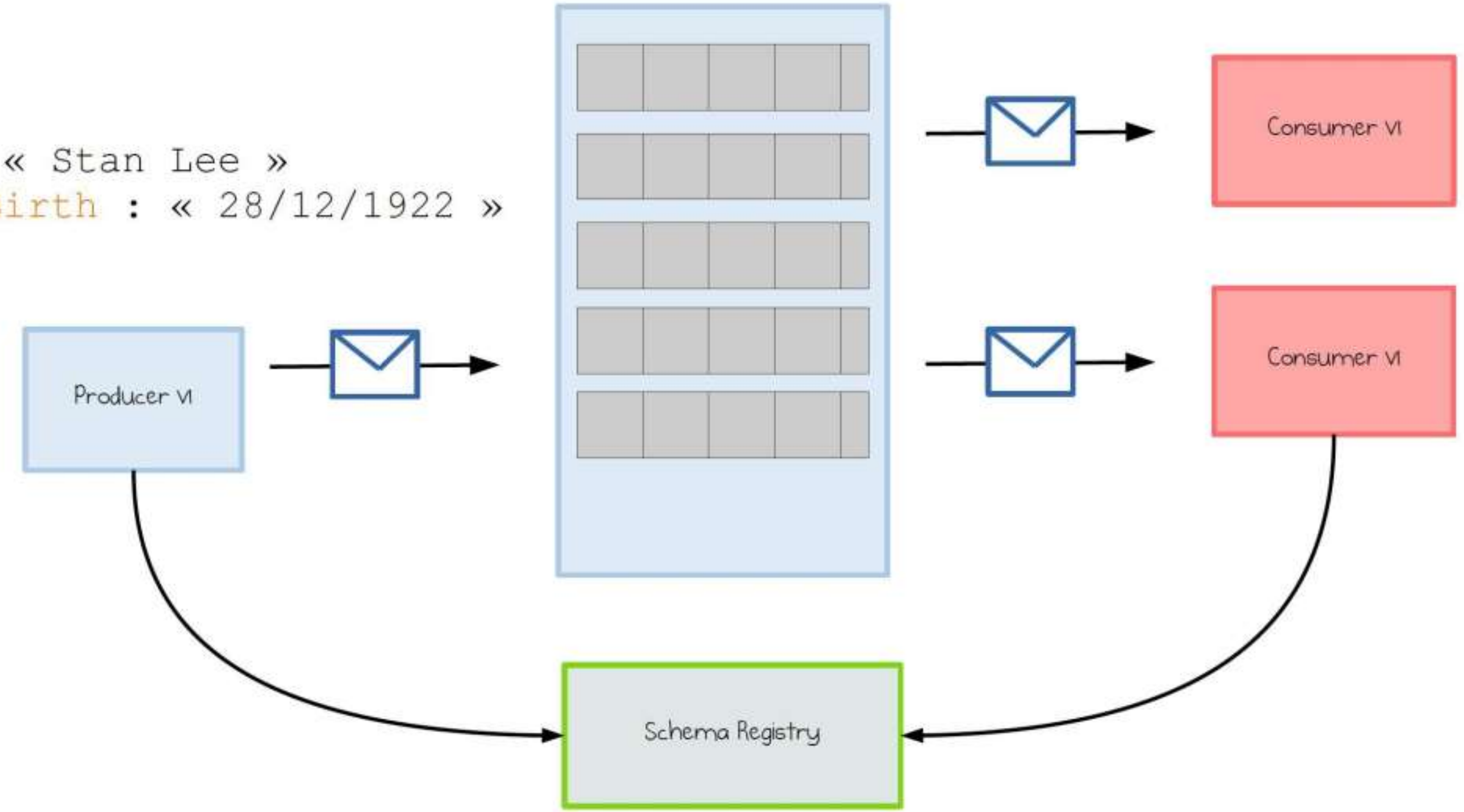
governance

Changes in producers might impact consumers

governance

Schema registry

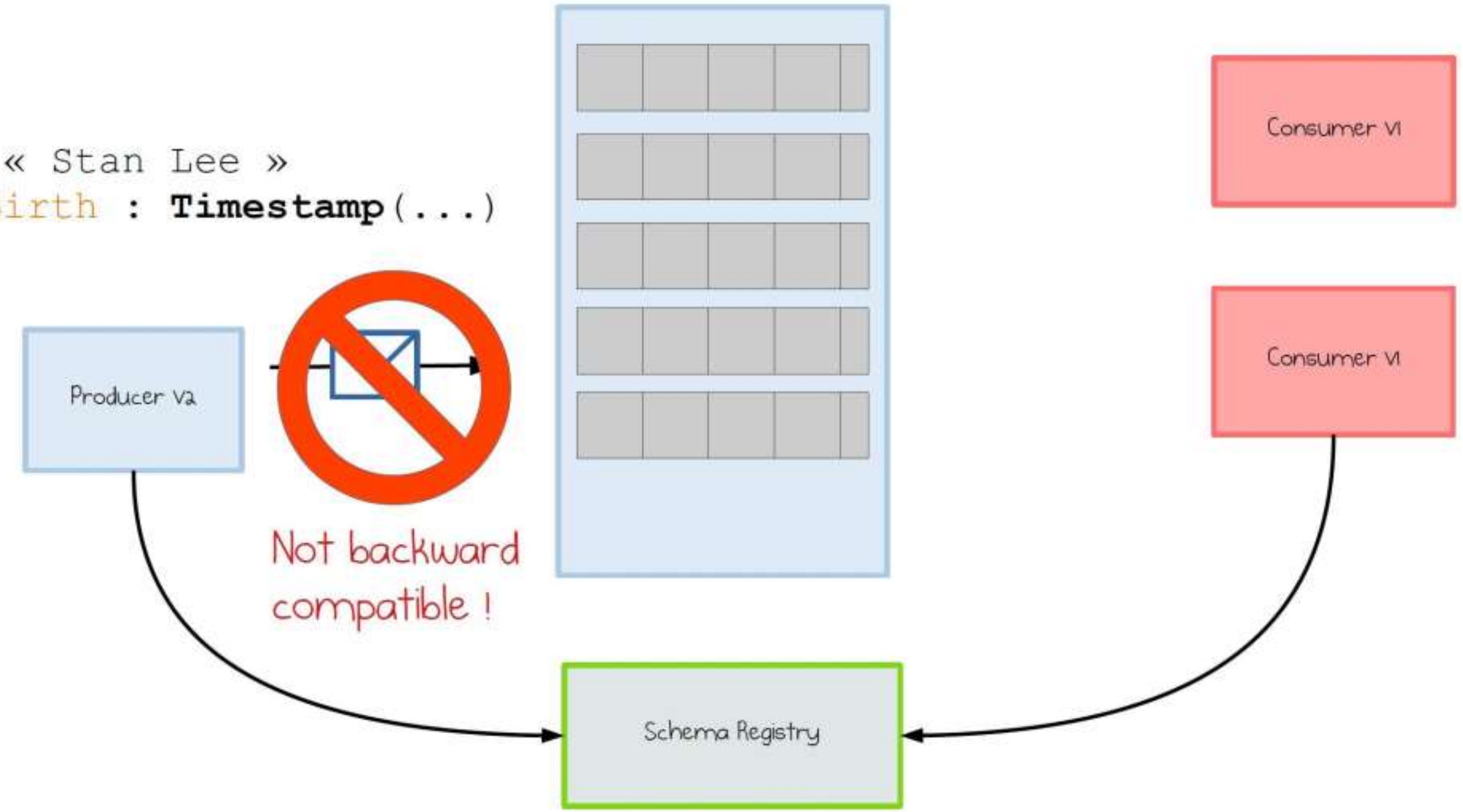
```
{
  Name : « Stan Lee »
  DateOfBirth : « 28/12/1922 »
}
```



Schema V1

```
{
  Name : { type : « string » }
  DateOfBirth : { type : « string » }
}
Compatibility : « Backward »
```

```
{
  Name : « Stan Lee »
  DateOfBirth : Timestamp(...)
}
```



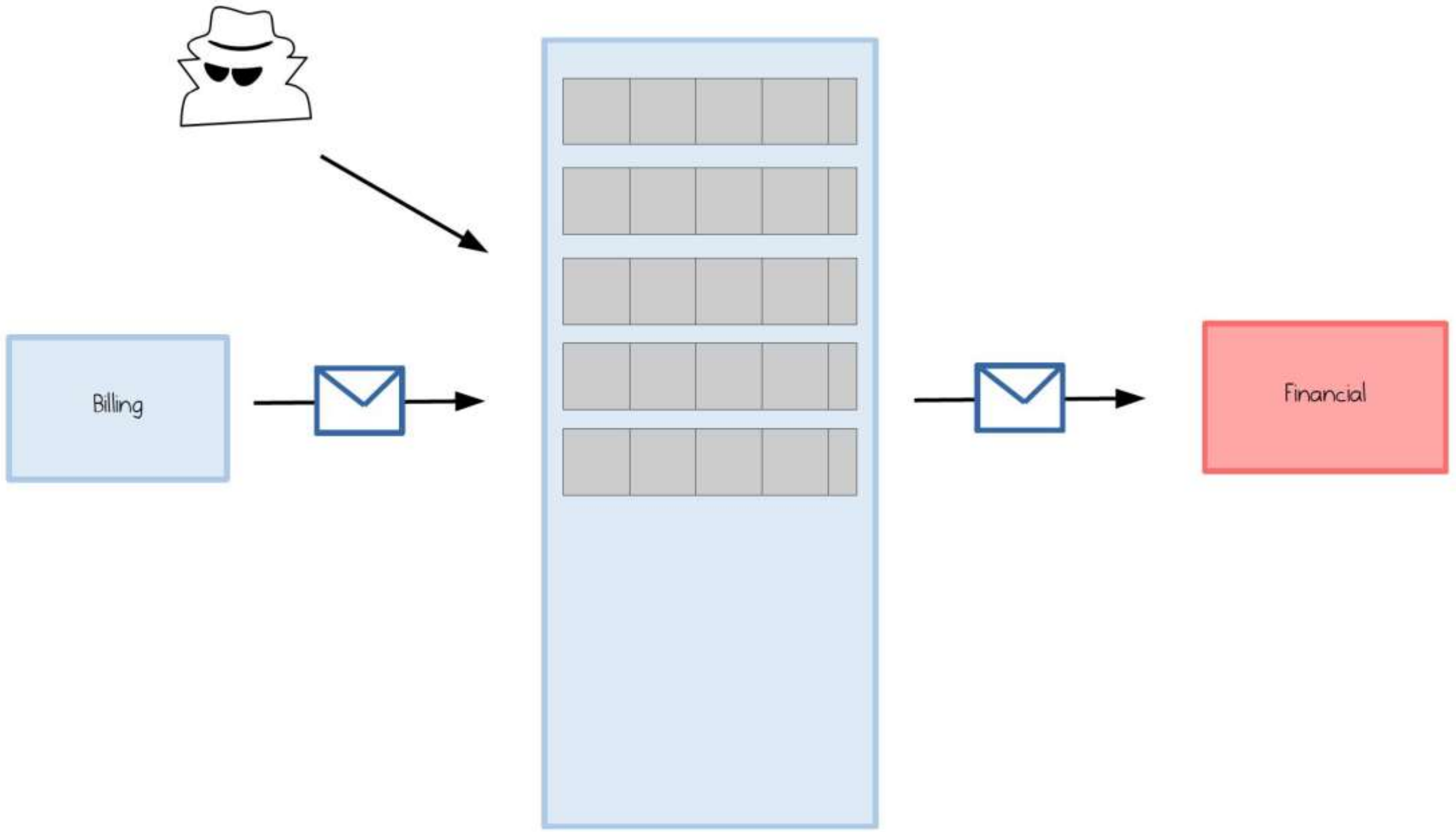
Schema V1

```
{
  Name : { type : « string » }
  DateOfBirth : { type : « string » }
}
Compatibility : « Backward »
```

Share Schemas

Let bad citizens wander around

FAIL


Billing
Financial

Leverage Security, ACL and Quota

Security

Authorization and ACLs

Enforcing Client Quotas

Installing prod on Sunday night

I HAVE TO GO
I NEED TO INSTALL
OUR PRODUCTION ENVIRONMENT
imgflip.com

configuration

If you use the default configuration…

You will have issues!

Please read the doc

[Running Kafka in Production]

[Running ZooKeeper in Production]

Not configuring your OS

```
/var/lib/confluent/kafka/
├── cleaner-offset-checkpoint
├── mytopic-0
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── 00000000000000000007.snapshot
│   └── leader-epoch-checkpoint
├── mytopic-1
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── 00000000000000000005.snapshot
│   └── leader-epoch-checkpoint
├── mytopic-2
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── 00000000000000000005.snapshot
│   └── leader-epoch-checkpoint
├── mytopic-3
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── 00000000000000000005.snapshot
│   └── leader-epoch-checkpoint
├── mytopic-4
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── 00000000000000000005.snapshot
│   └── leader-epoch-checkpoint
```

OS

Tune at least your open file descriptors and mmap count.

Configure your os

Running Kafka in Production

Disregarding Apache Zookeeper

Not understanding Ordering

No monitoring

Too much partitions

Not enough partitions

Partition key choice

Topics vs Partitions

Call external services in Kafka Streams

Questions