#NDCPORTO @NDC_CONFERENCES

APACHE KAFKA EVENT-STREAMING PLATFORM FOR .NET DEVELOPERS

APRIL, 2020

@GAMUSSA | #NDCPORTO | @CONFLUENTINC


Streams

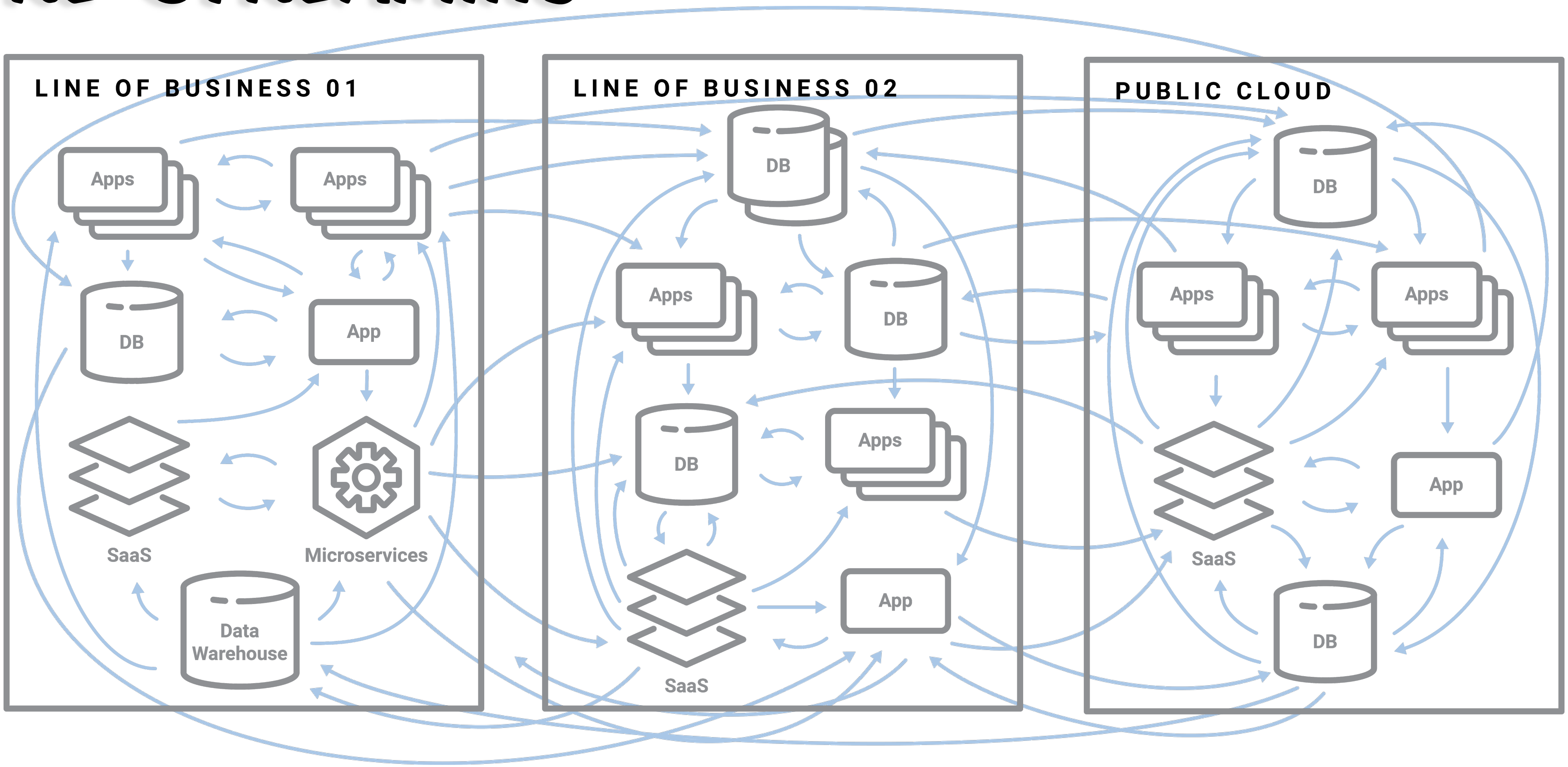
I BUILD HIGHLY SCALABLE

Hello World

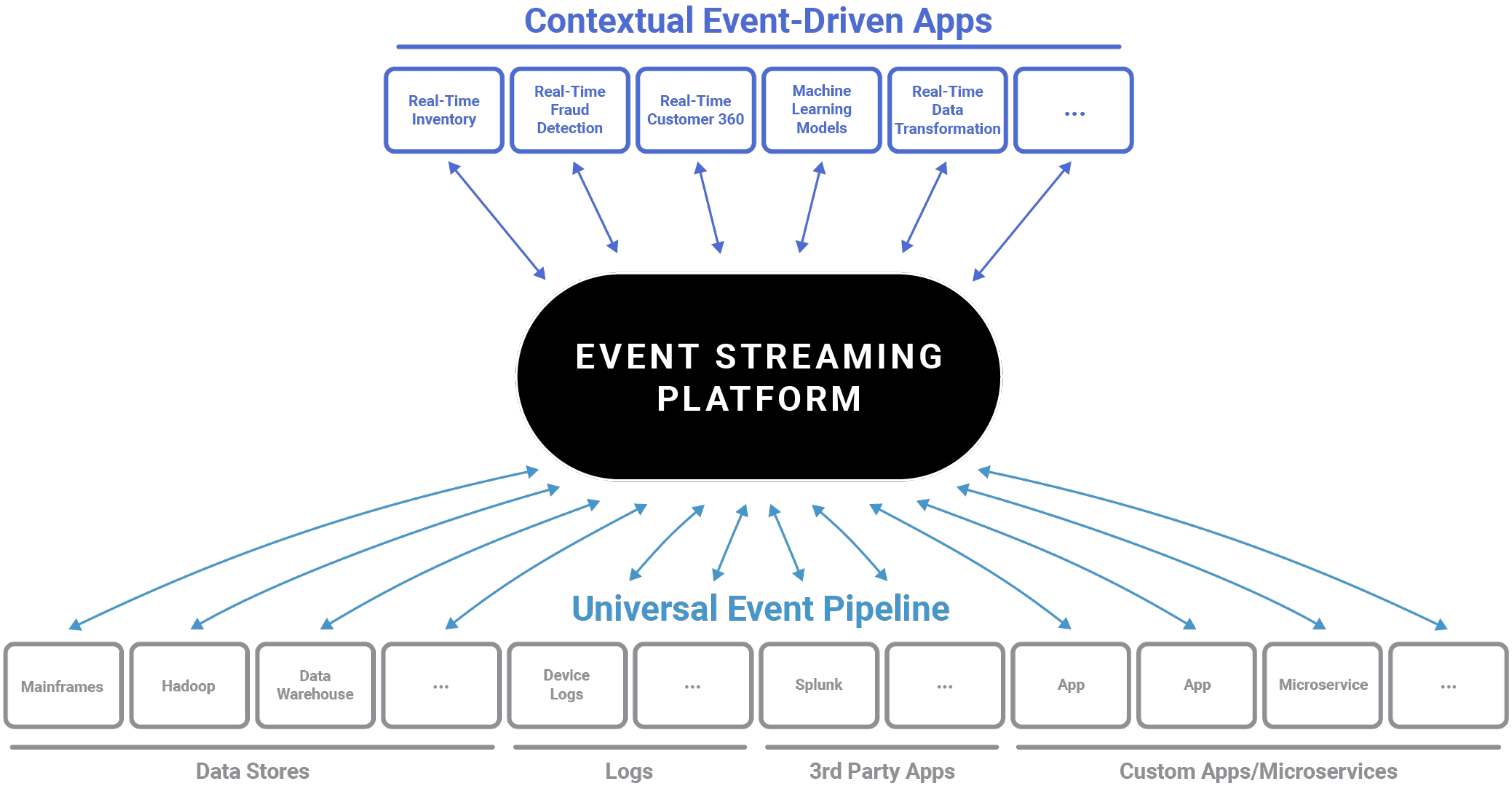
APPS

A COMPANY IS BUILD ON DATA FLOWS BUT ALL WE HAVE IS DATA STORES

PRE-STREAMING



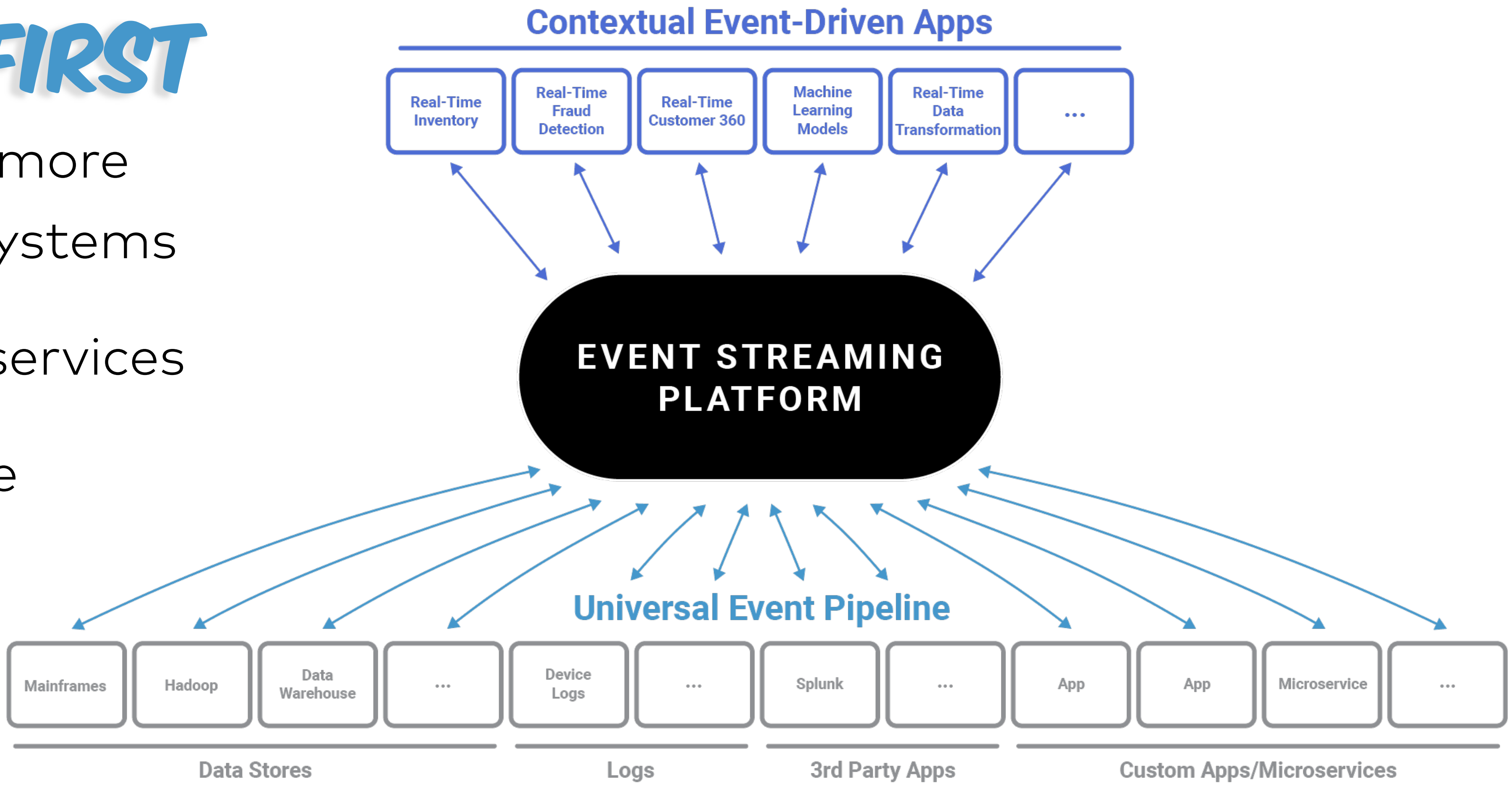

HIVE
hadoop
Map Reduce
STORM
kafka
APACHE
Spark
ZOO



Contextual Event-Driven Apps
Real-Time Inventory
Real-Time Fraud Detection
Real-Time Customer 360
Machine Learning Models
Real-Time Data Transformation
...
EVENT STREAMING PLATFORM
Universal Event Pipeline
Mainframes
Hadoop
Data Warehouse
...
Device Logs
...
Splunk
...
App
App
Microservice
...
Data Stores
Logs
3rd Party Apps
Custom Apps/Microservices

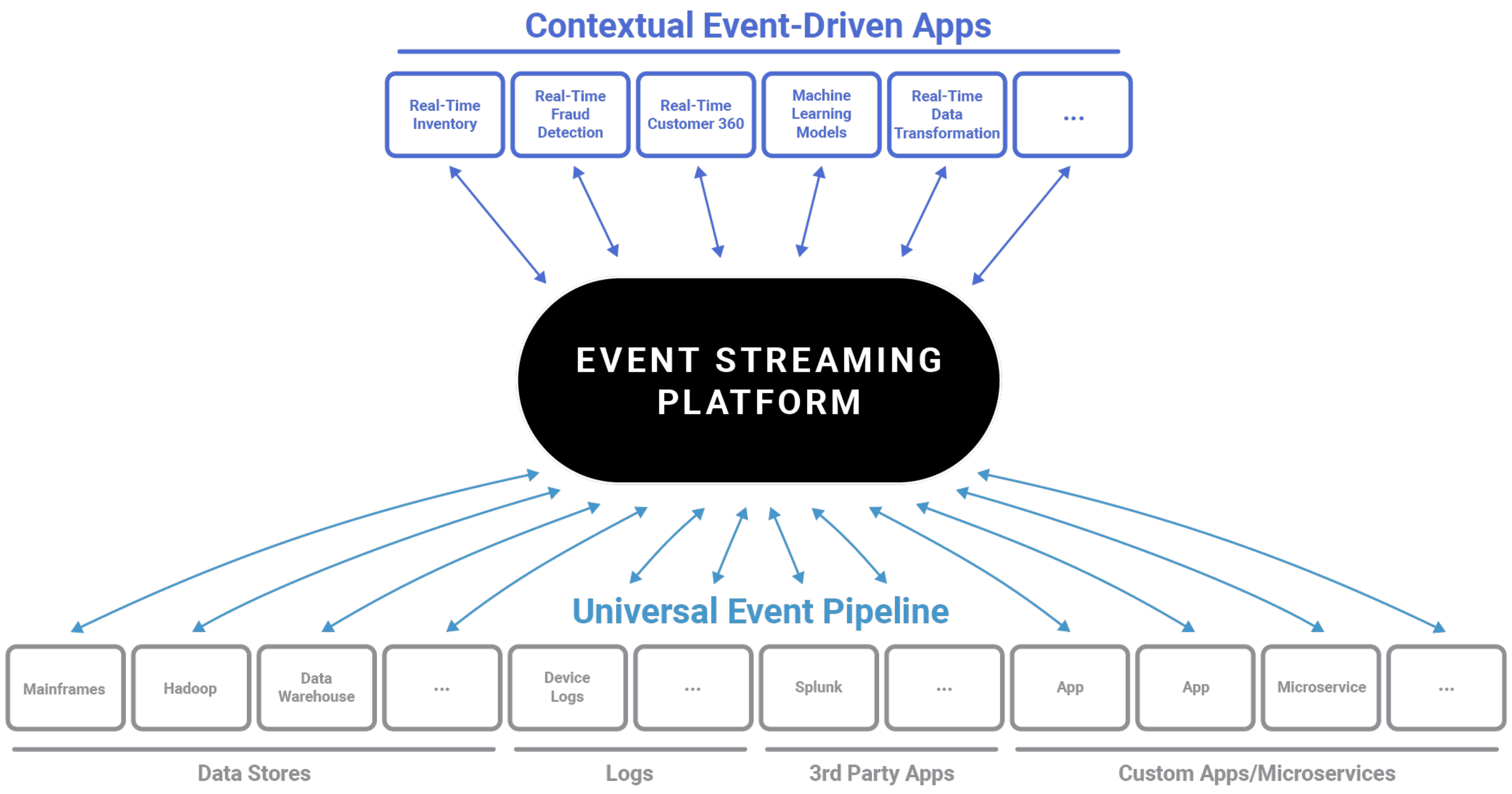
NEW WORLD STREAMING FIRST

- DB/DWH + Many more distributed data systems
- Monolith -> Microservices
- Batch -> Real-time



STREAMING PLATFORM

- STORAGE
- PUB / SUB
- PROCESSING



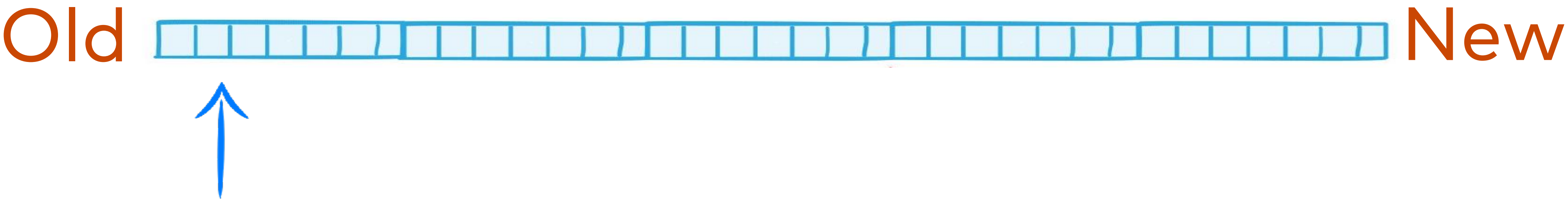
STORAGE

CORE ABSTRACTION

- DB - table
- Hadoop - file
- Kafka - ?

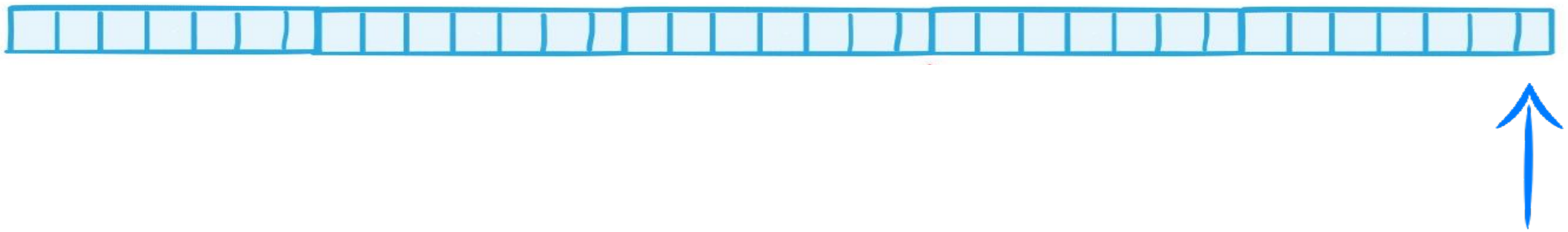
LOG

THE LOG IS A SIMPLE IDEA



Messages are added at the end of the log

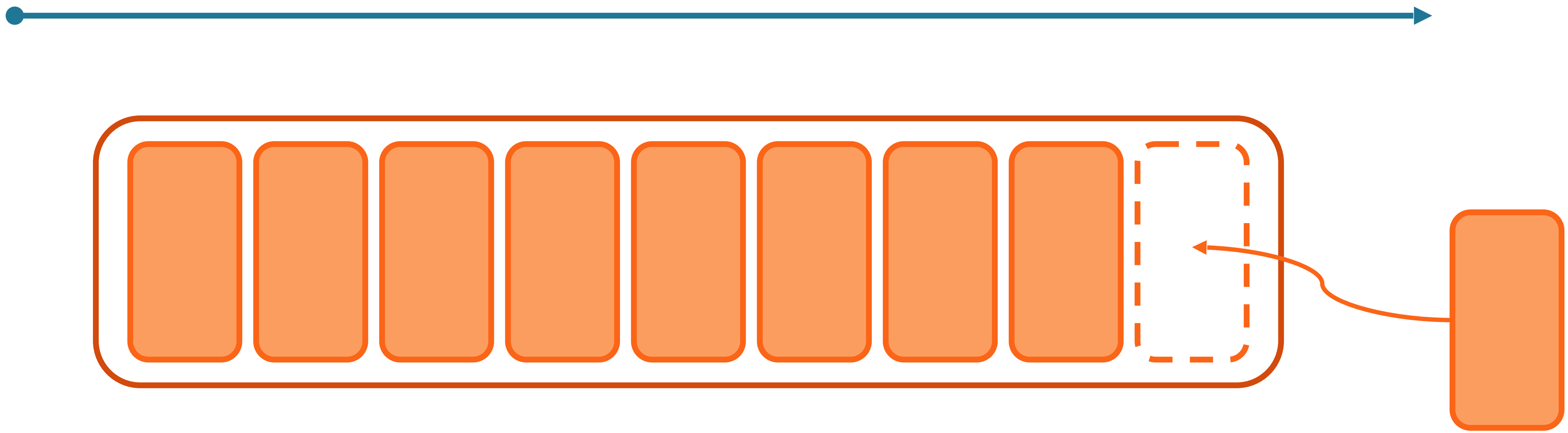
THE LOG IS A SIMPLE IDEA

Old  New

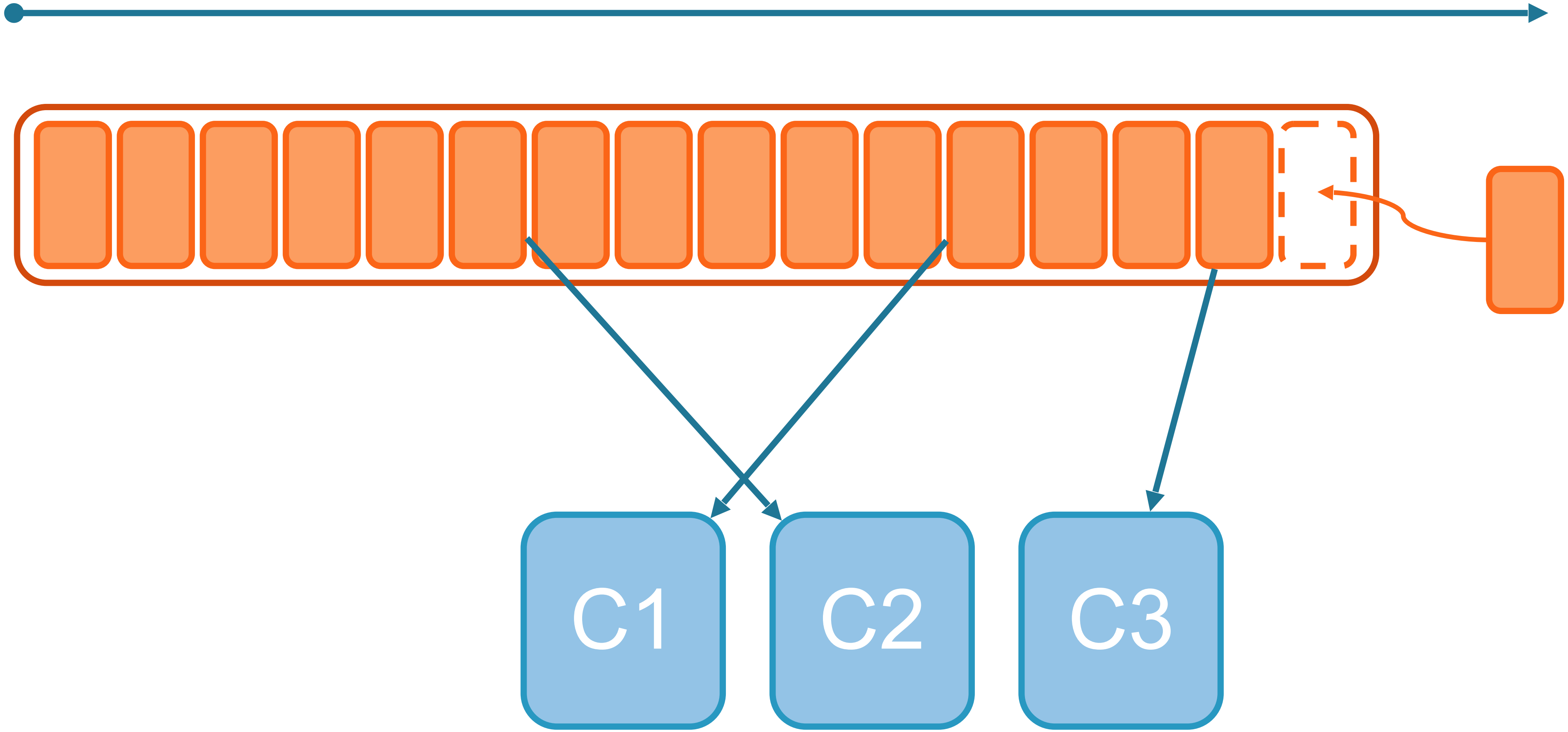
Messages are added at the end of the log

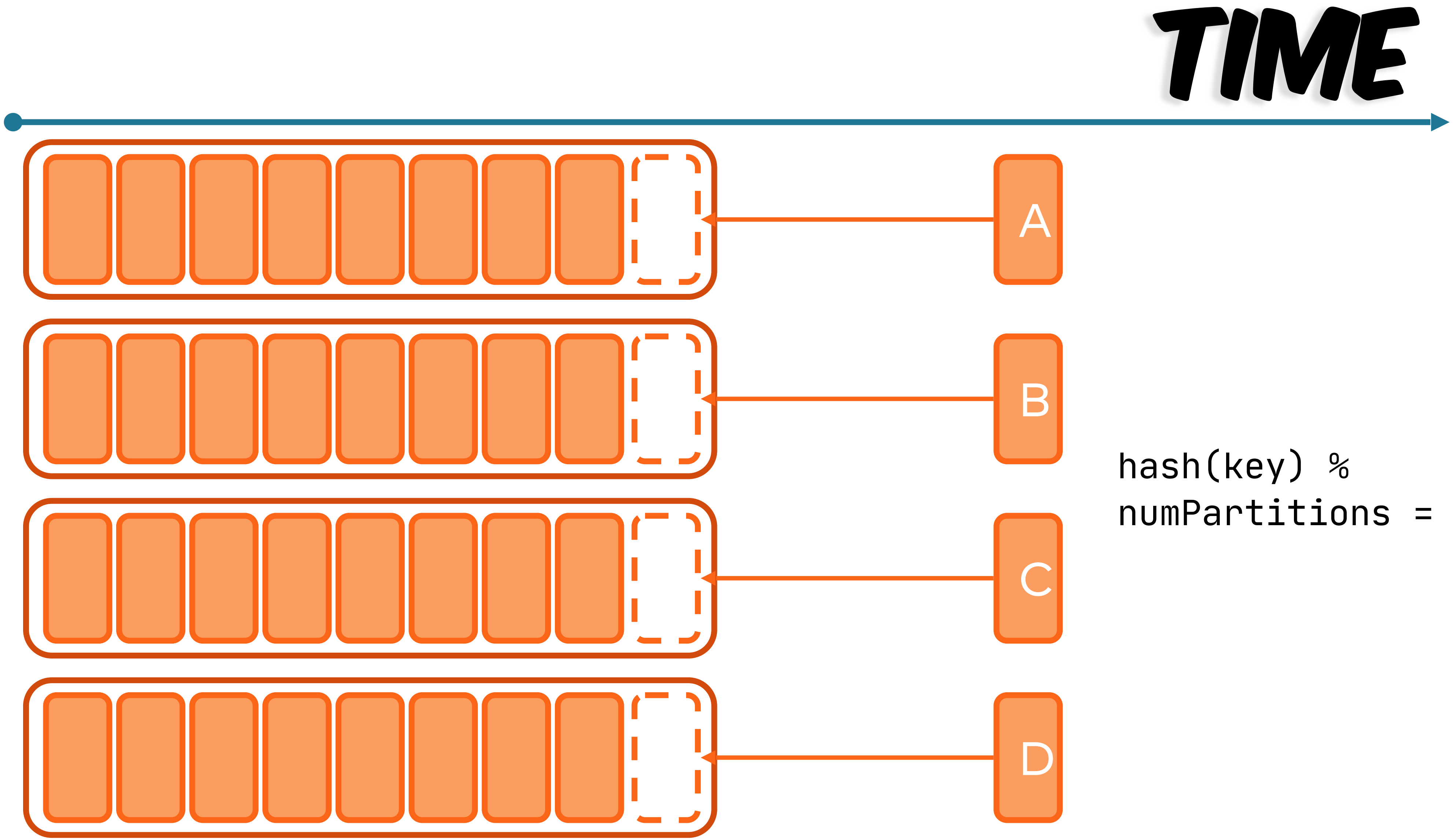
PUB / SUB

TIME

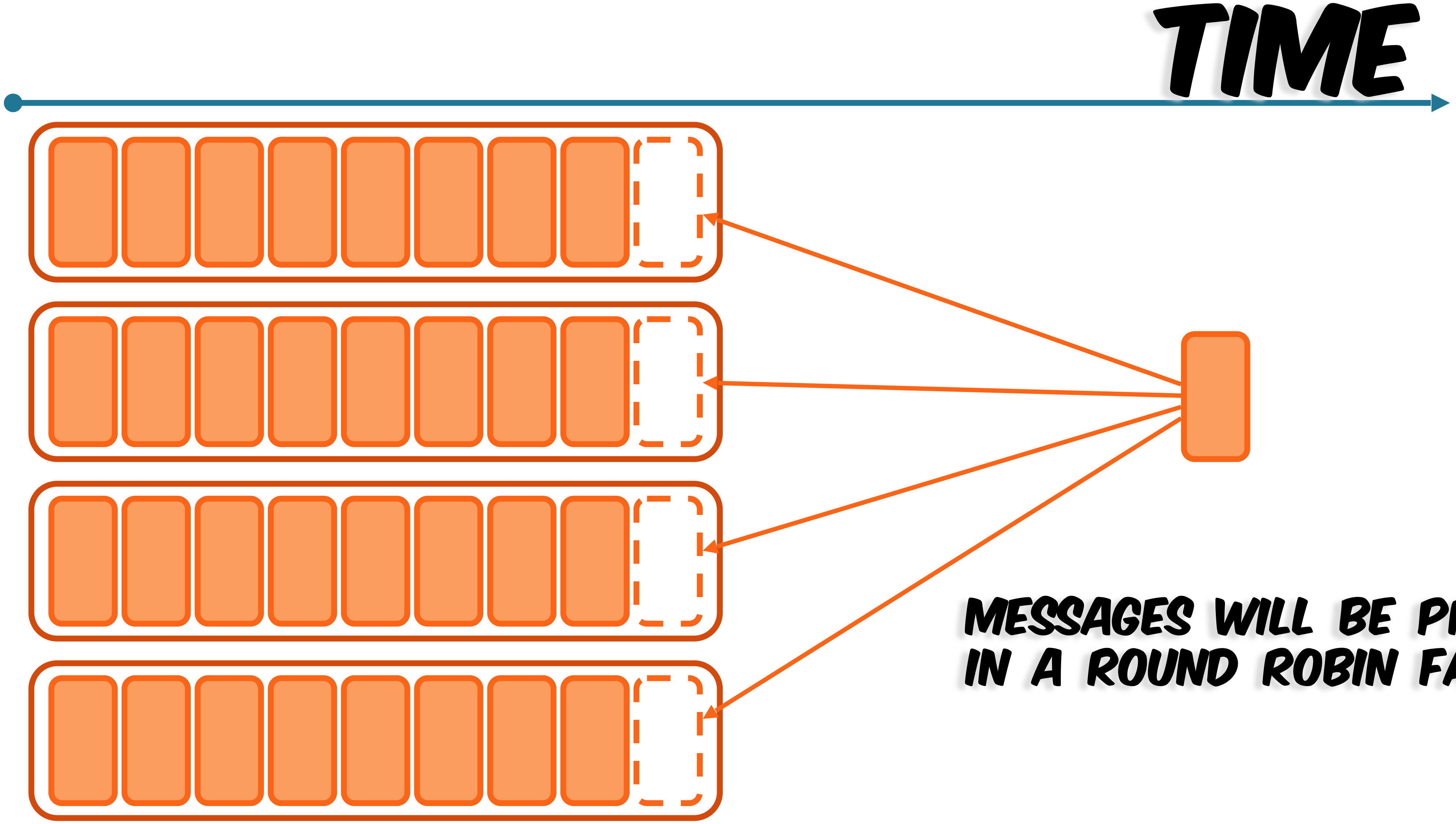


TIME



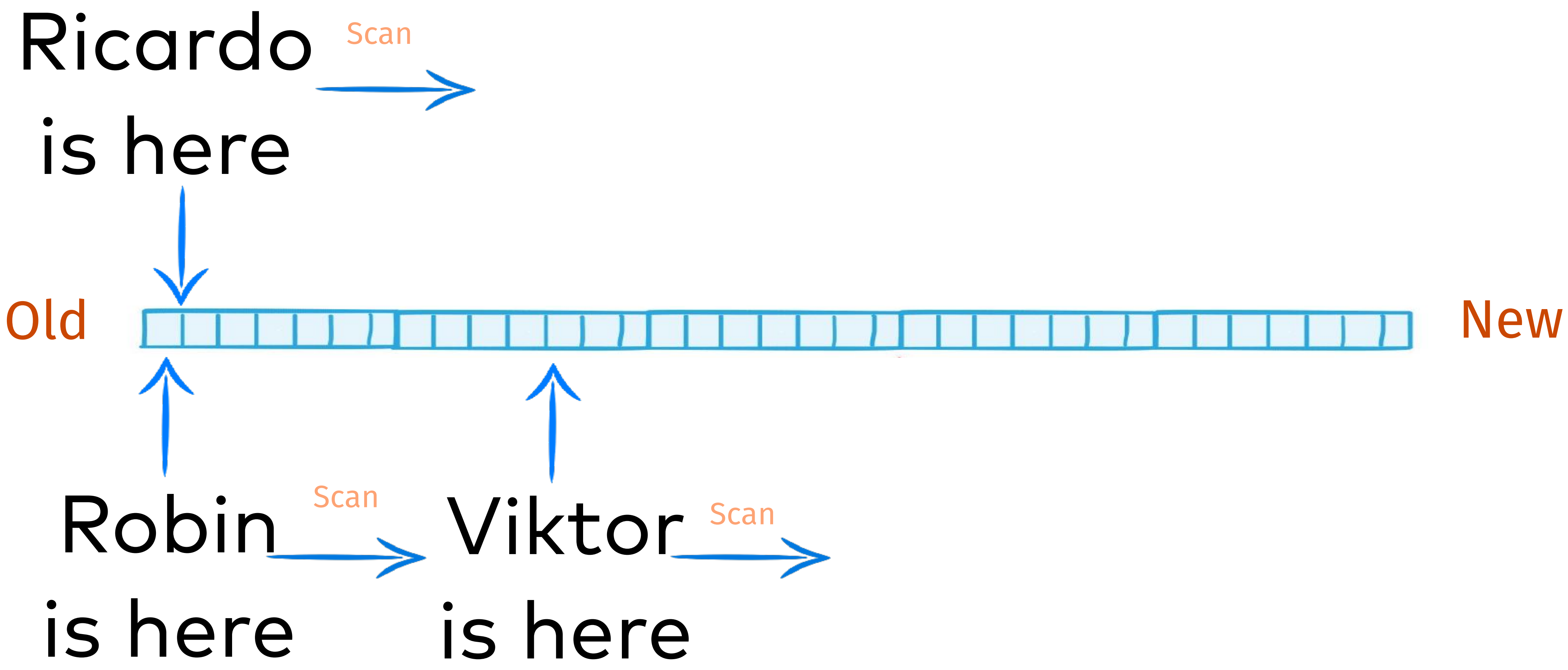


`hash(key) % numPartitions = N`

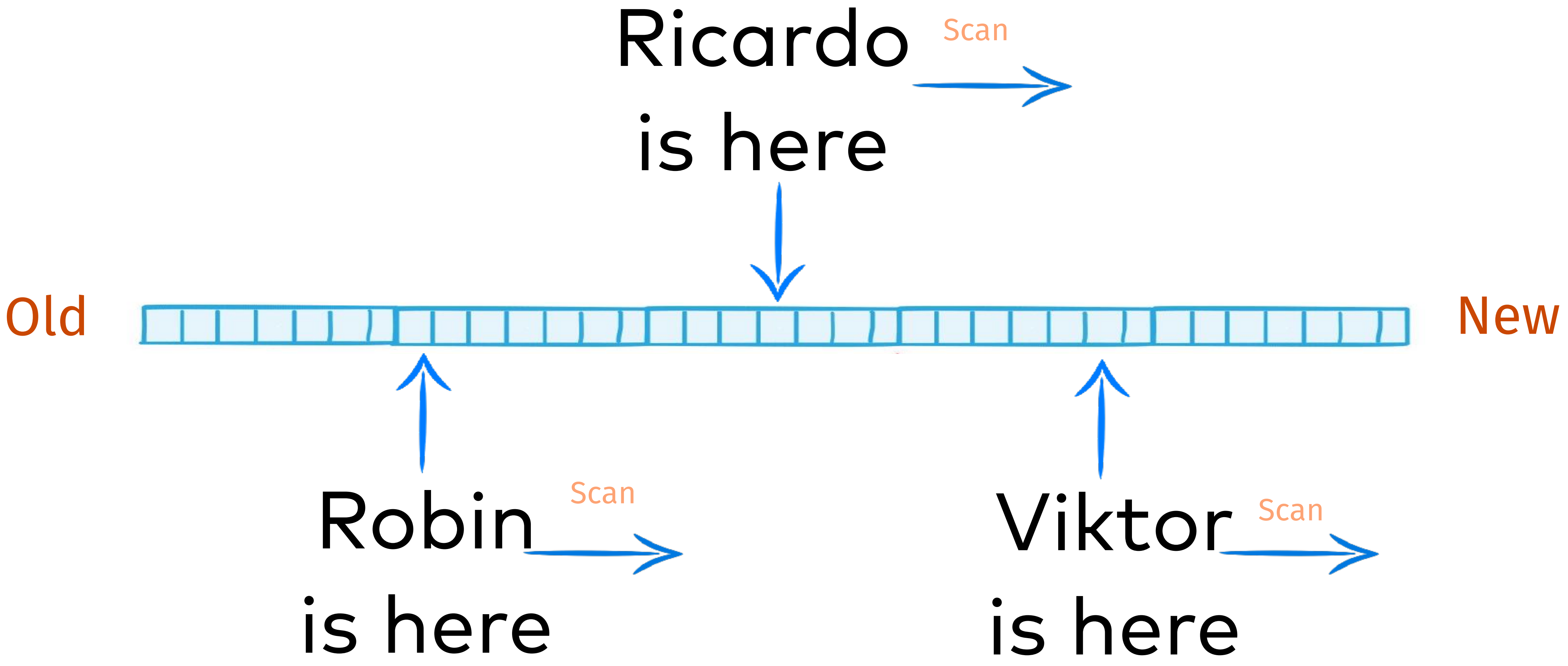


MESSAGES WILL BE PRODUCED IN A ROUND ROBIN FASHION

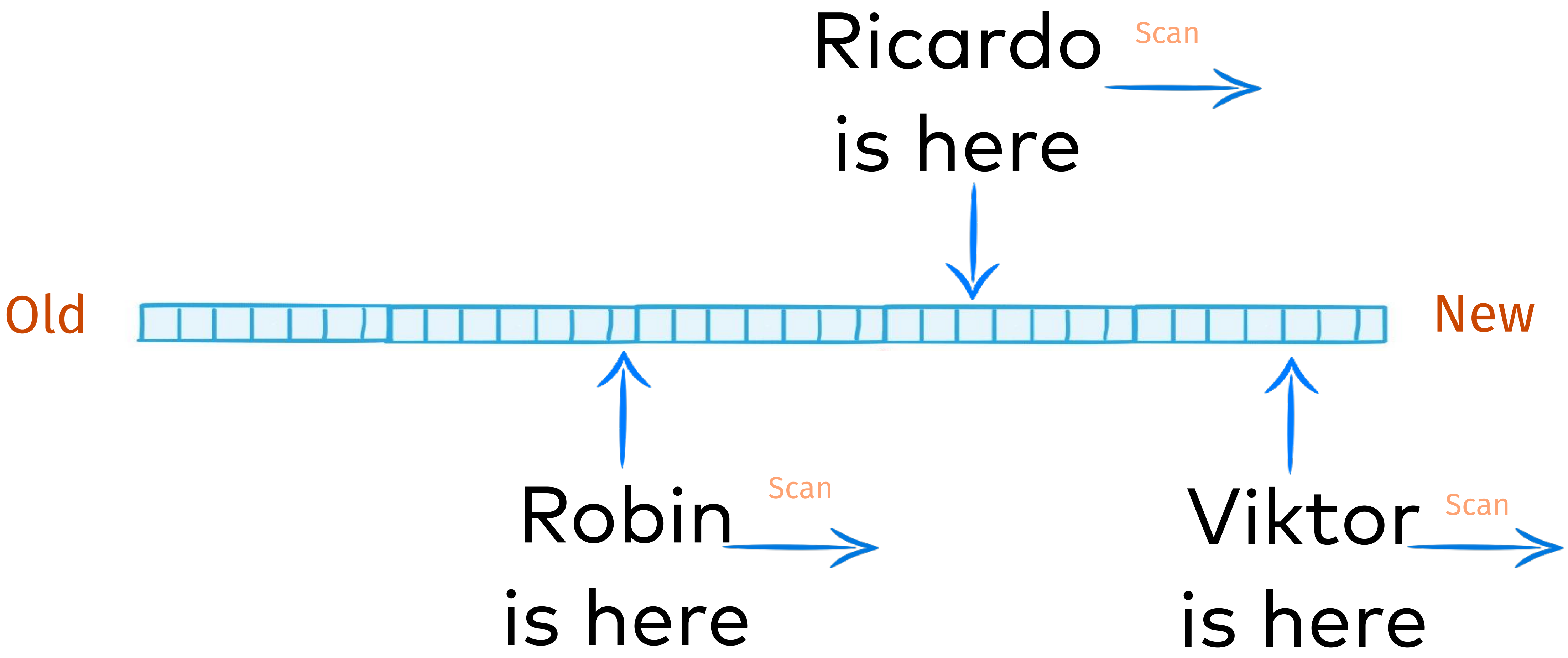
CONSUMERS HAVE A POSITION ALL OF THEIR OWN



CONSUMERS HAVE A POSITION ALL OF THEIR OWN

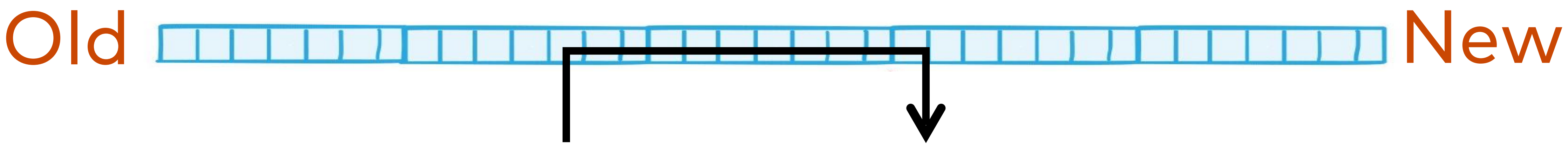


CONSUMERS HAVE A POSITION ALL OF THEIR OWN

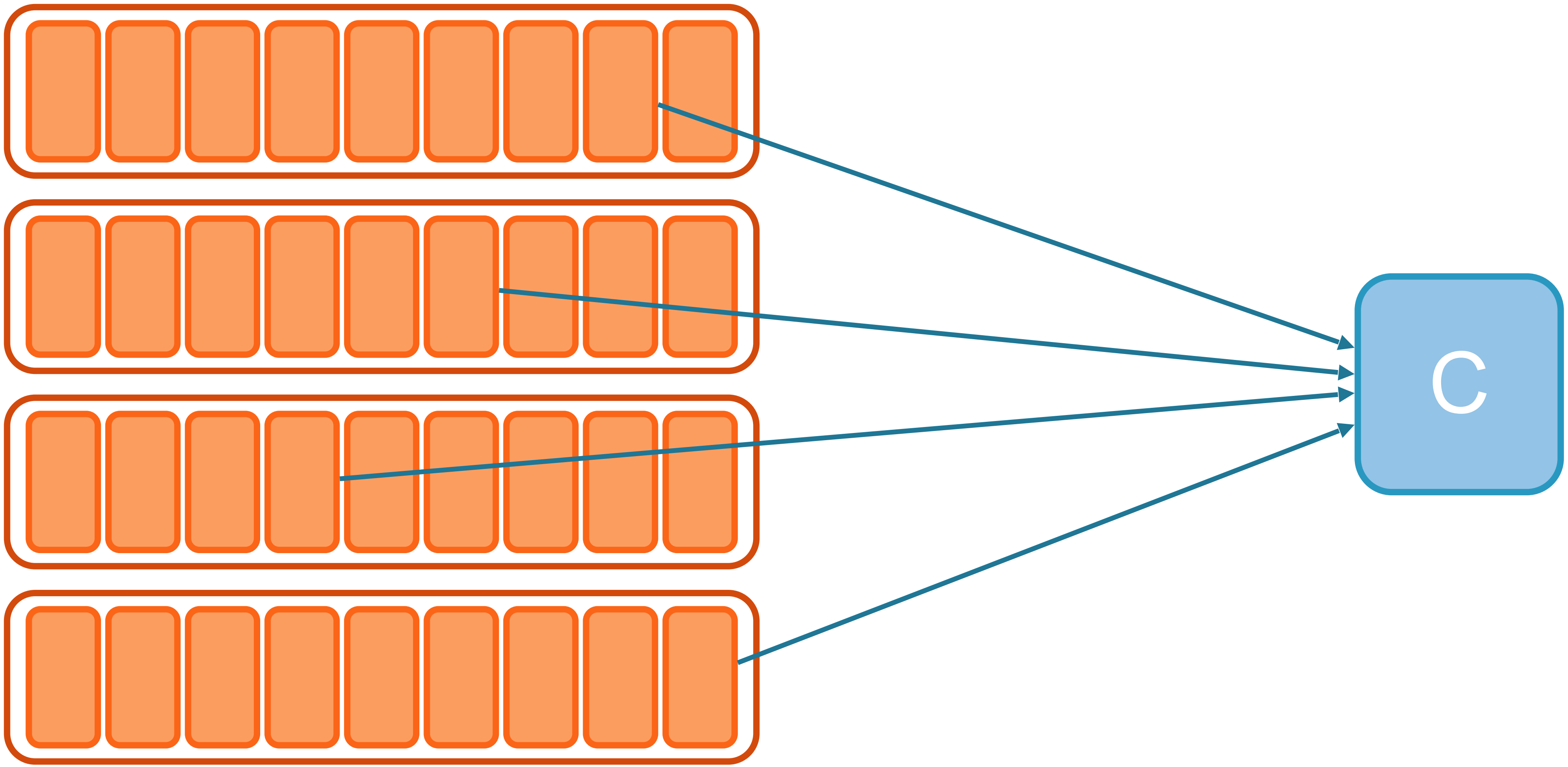


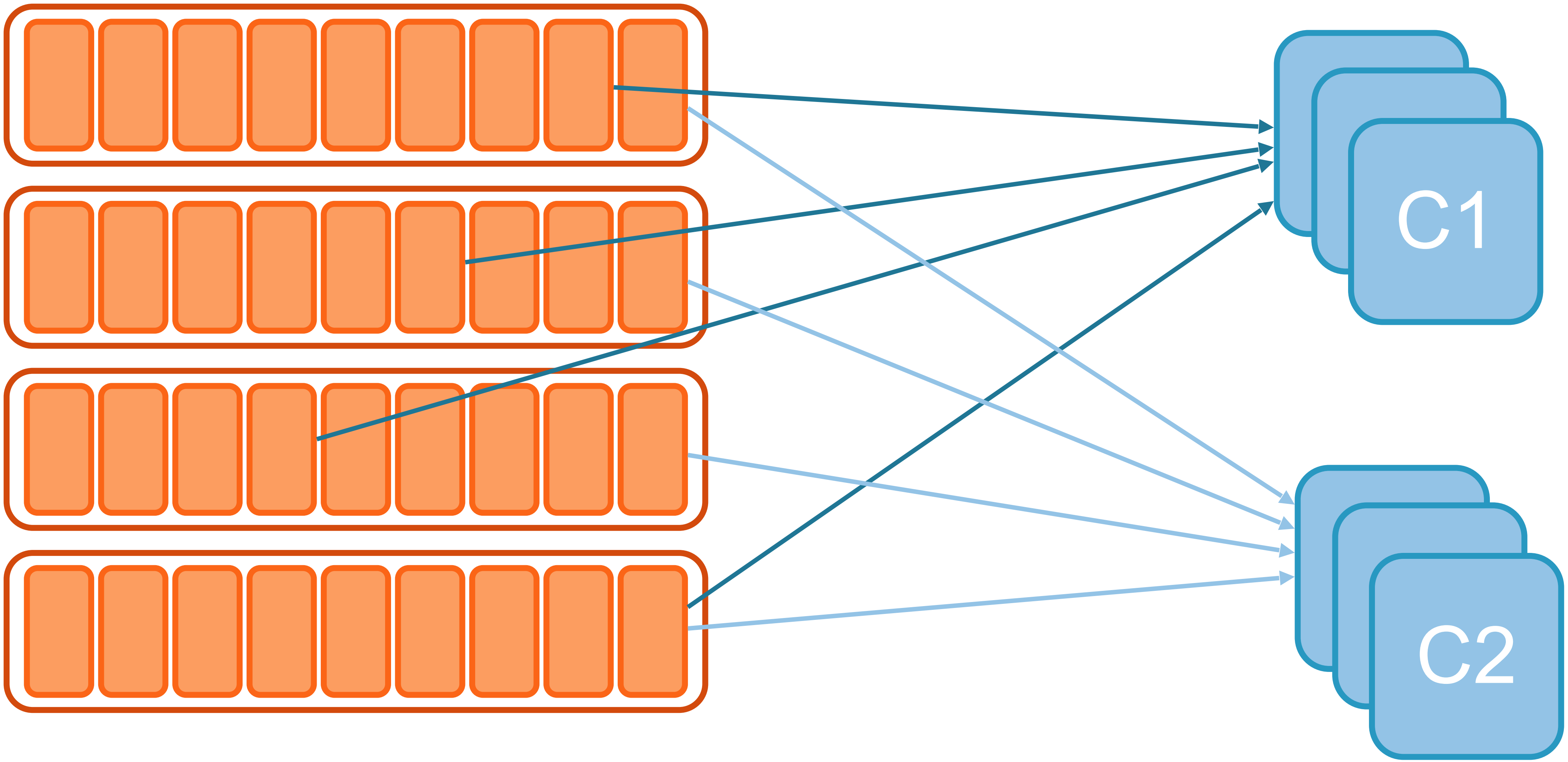
ONLY SEQUENTIAL ACCESS

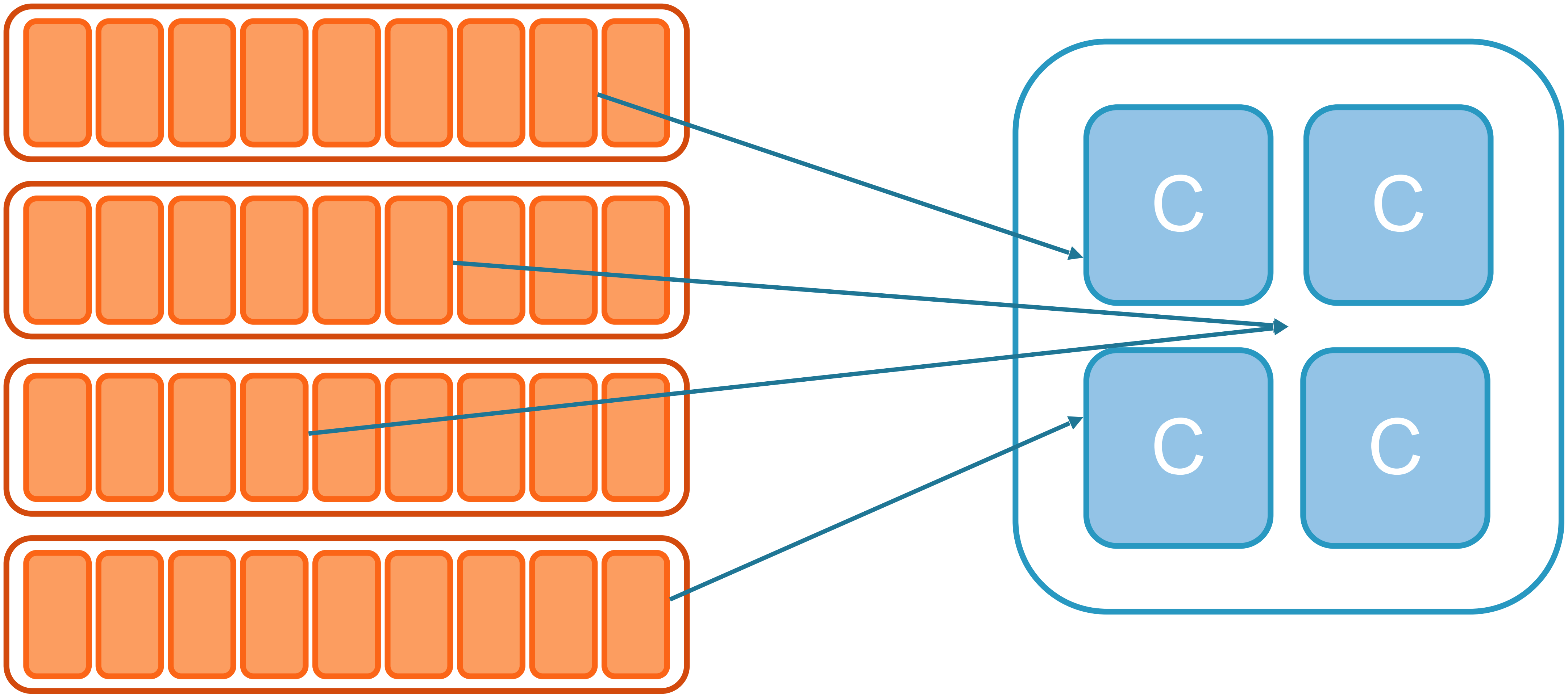
Read to offset & scan

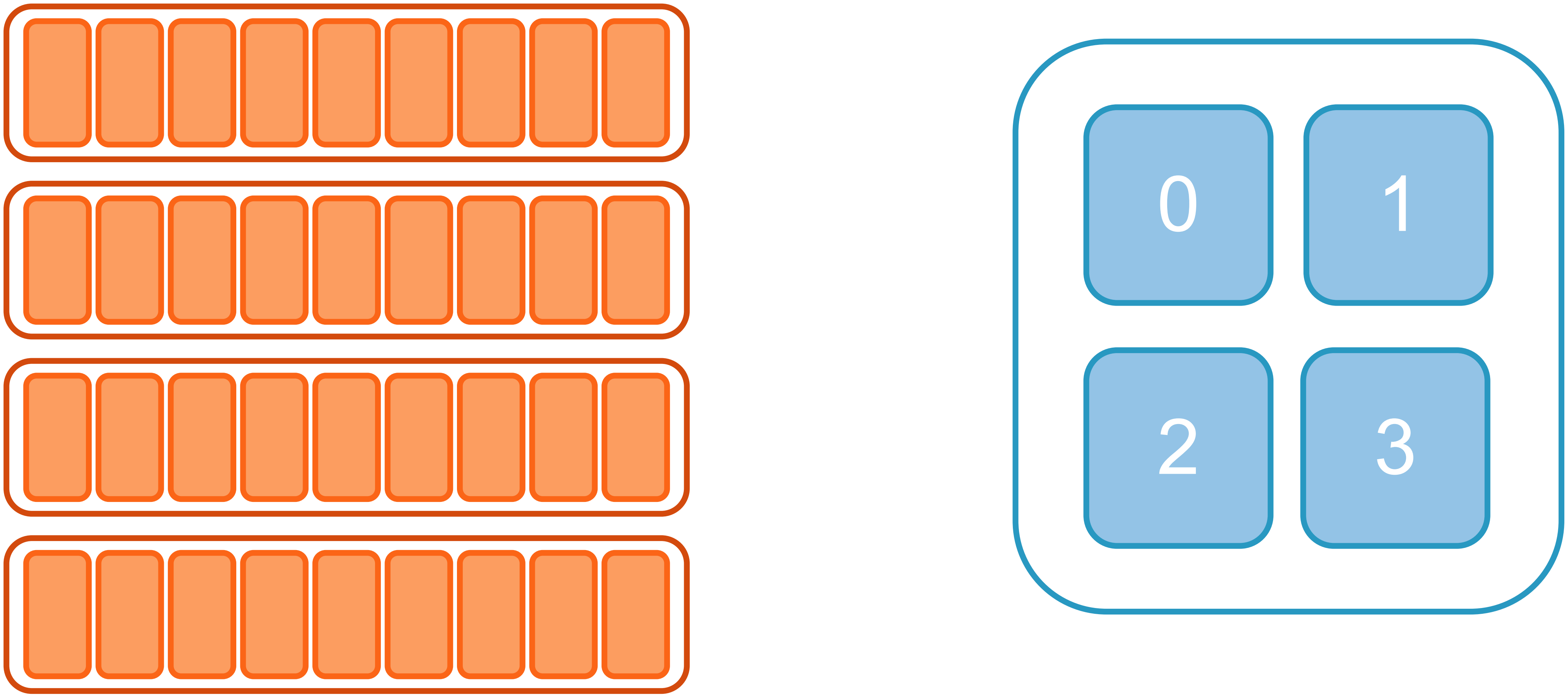


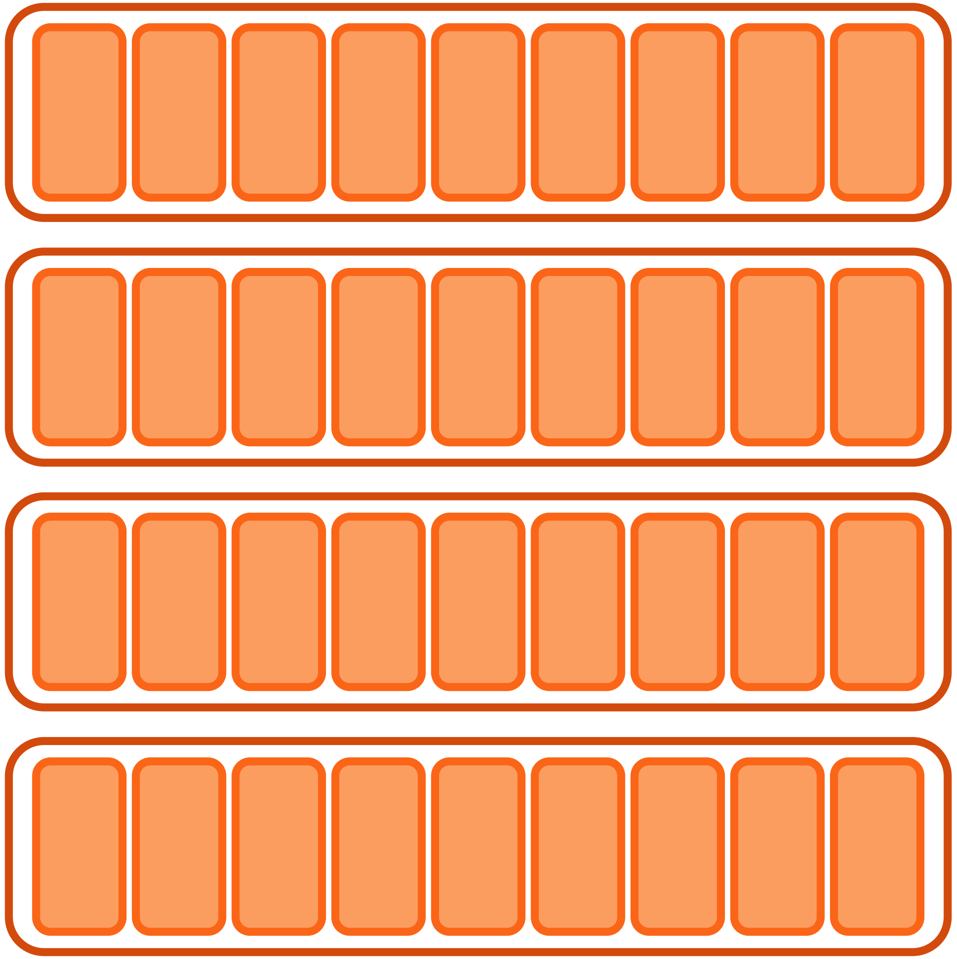

CONSUMER GROUP
COORDINATOR
CONSUMERS
CONSUMER GROUP

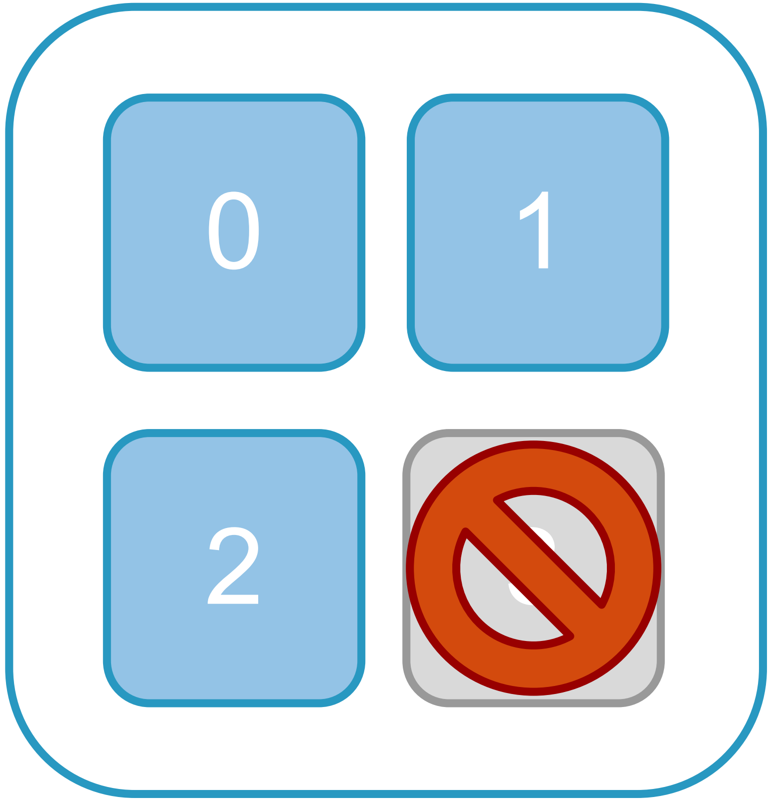

C

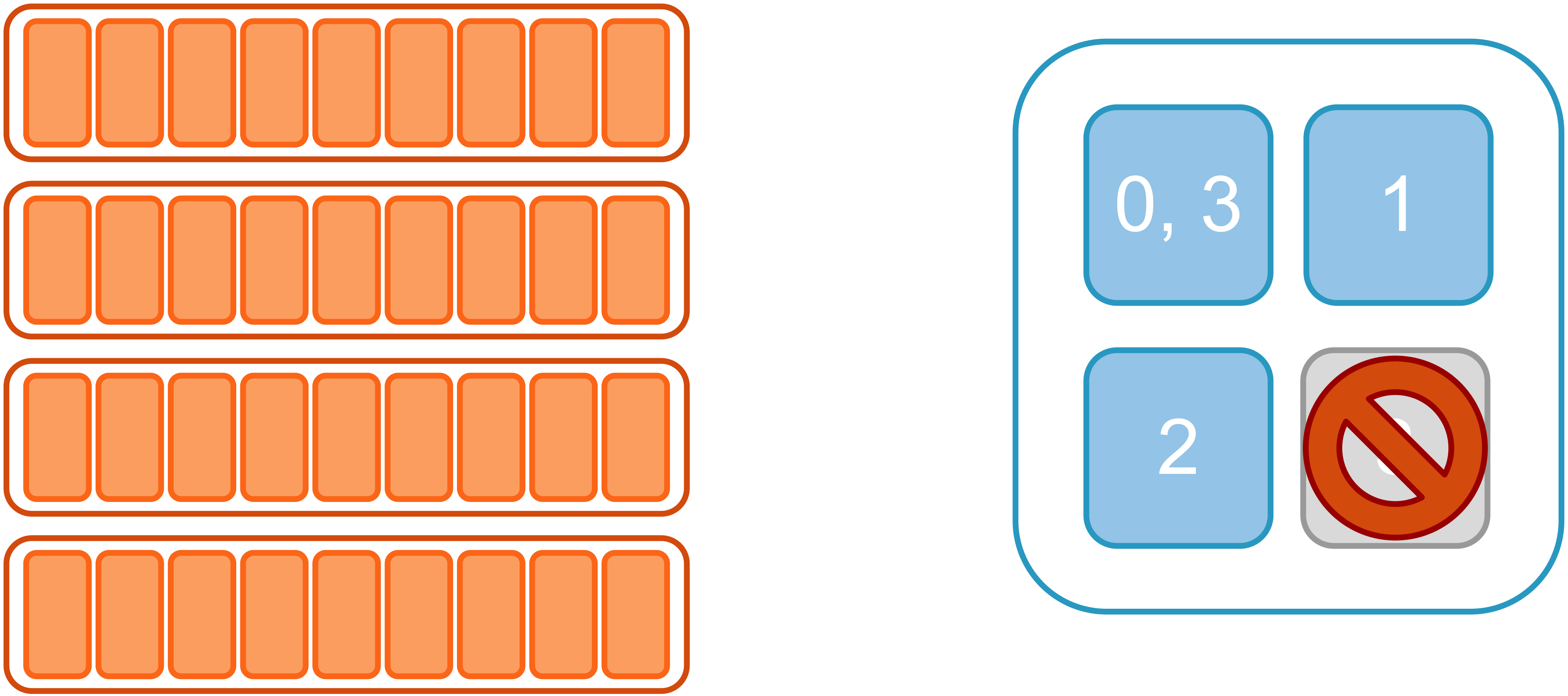

C1
C2


C
C
C
C


0
1
2
3

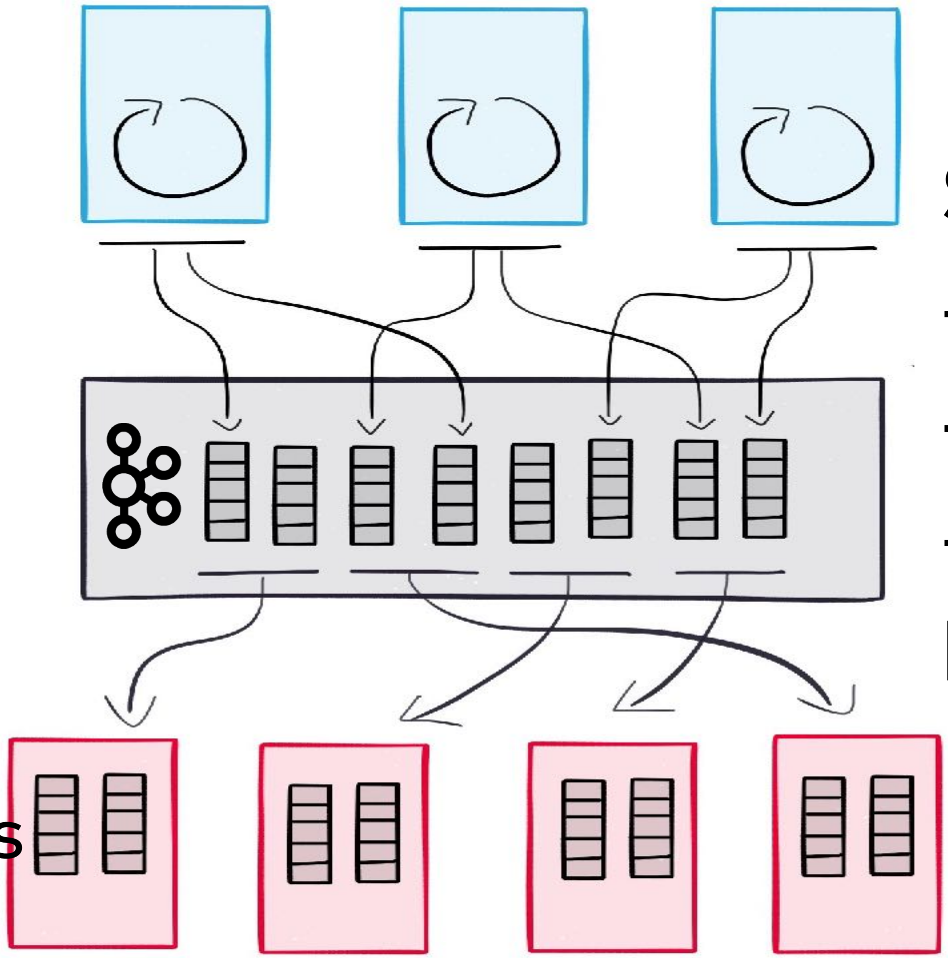



0
1
2


0, 3
1
2

LINEARLY SCALABLE ARCHITECTURE

Producers



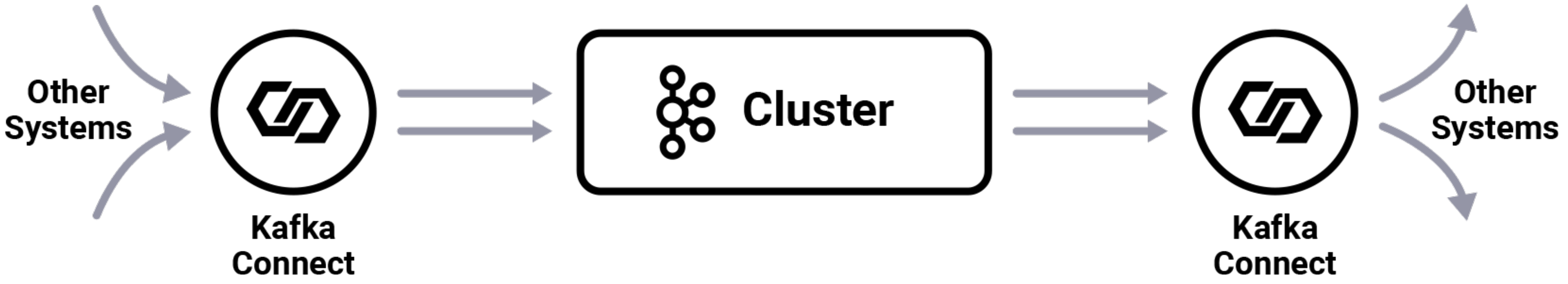
Consumers

Single topic:
- Many producers machines
- Many consumer machines
- Many Broker machines

No Bottleneck!!

From/to other systems: Kafka Connect

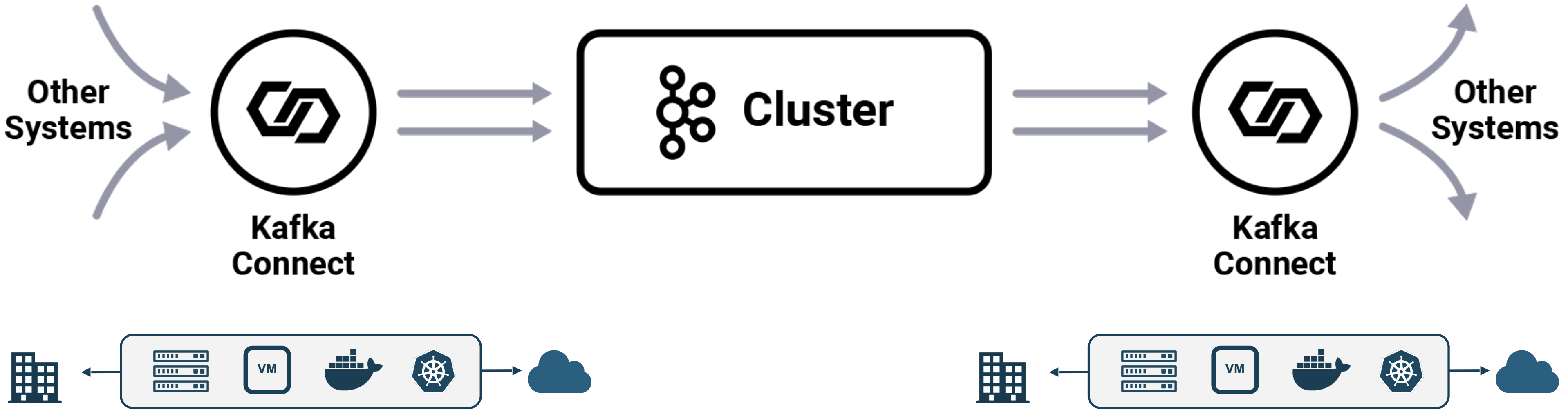
and more



Tip: Great option to gradually move workloads to Kafka while keeping production

Kafka Connect

- Deployed standalone (development) or as a **distributed cluster (production)**
- Elastic service that works on bare-metal, VMs, containers, Kubernetes, ...
- The individual 'Connector' determines delivery guarantees, e.g., exactly-once



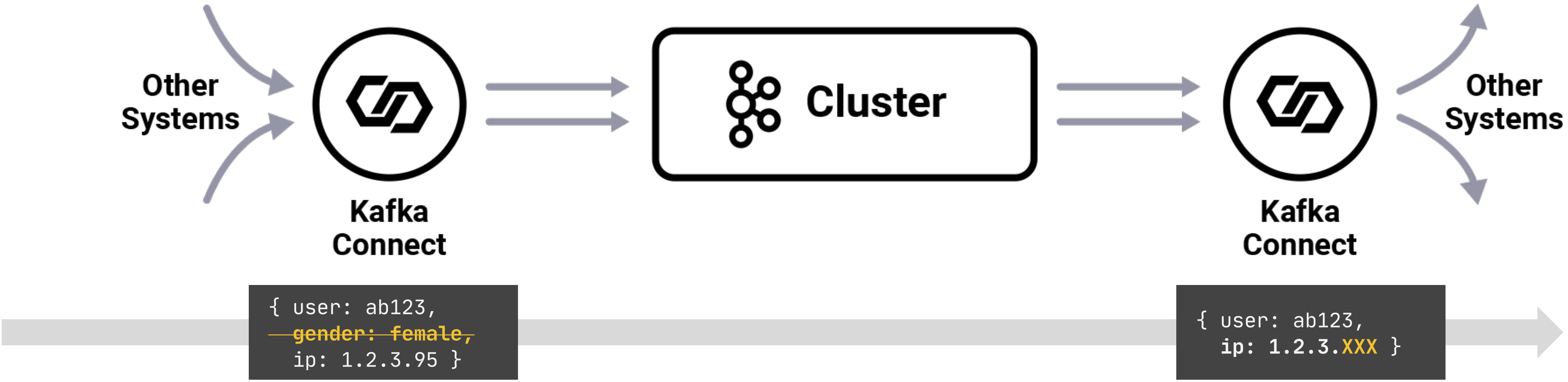
Single Message Transforms for real-time ETL

Ingress: modify an Event before storing

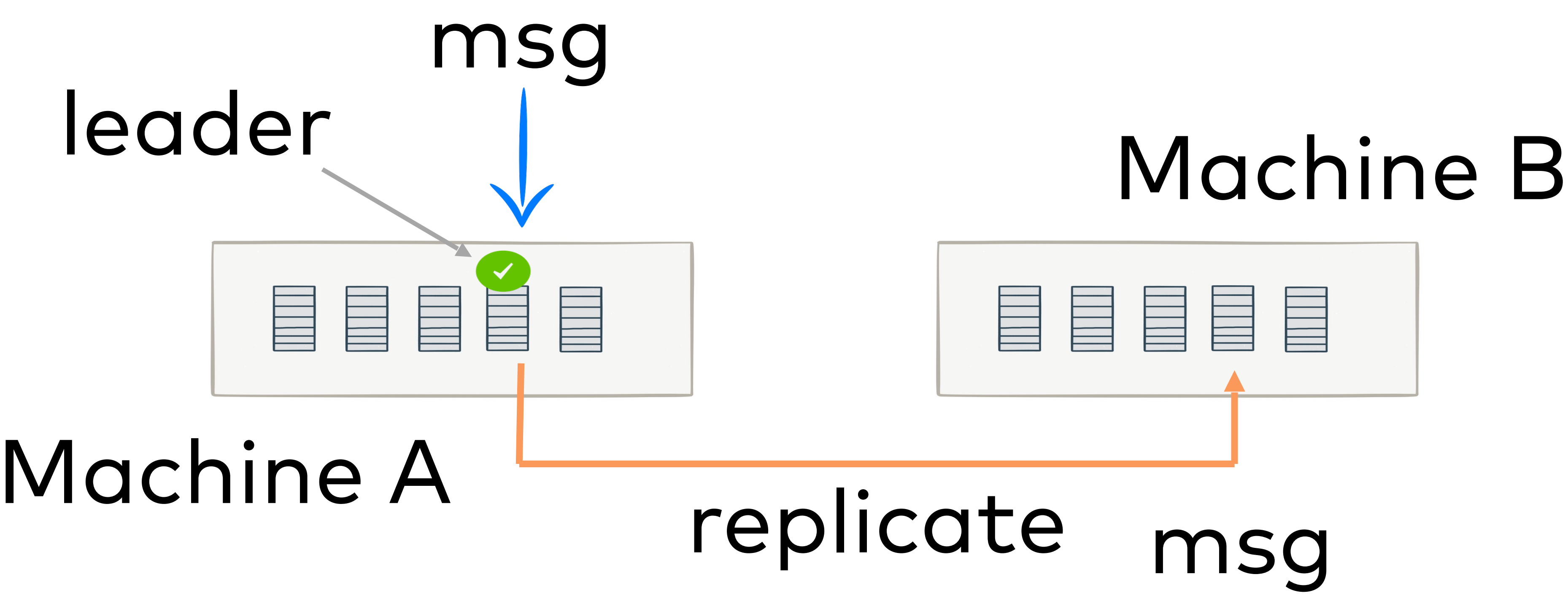
- Obfuscate sensitive information, e.g. PII
- Add origin of event for lineage tracking
- Remove unnecessary data fields
- … and more

Egress: modify an Event on its way out

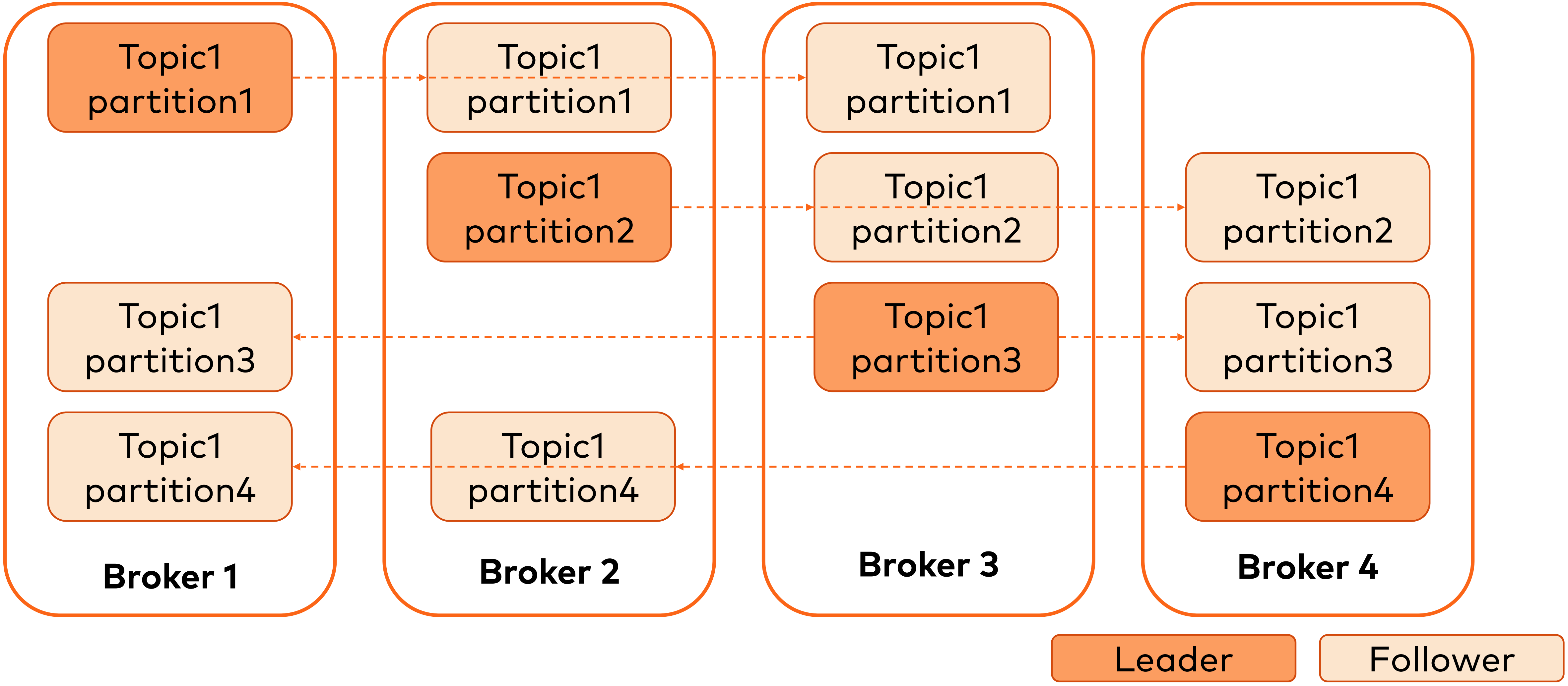
- Route high-priority events to faster stores
- Direct events to different Elasticsearch indexes
- Cast data types to match destination
- … and more



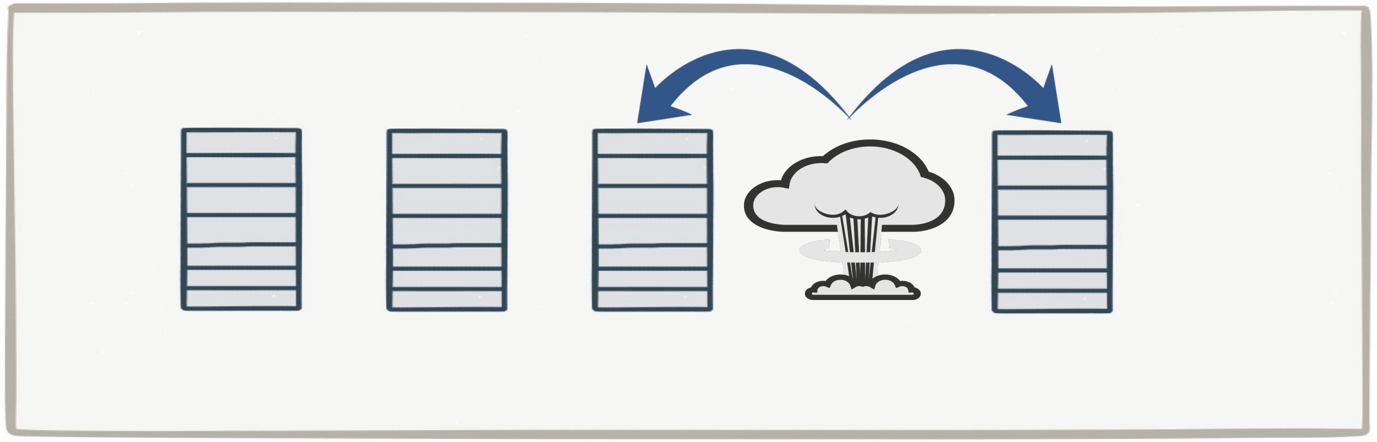
REPLICATE TO GET FAULT TOLERANCE



PARTITION LEADERSHIP AND REPLICATION

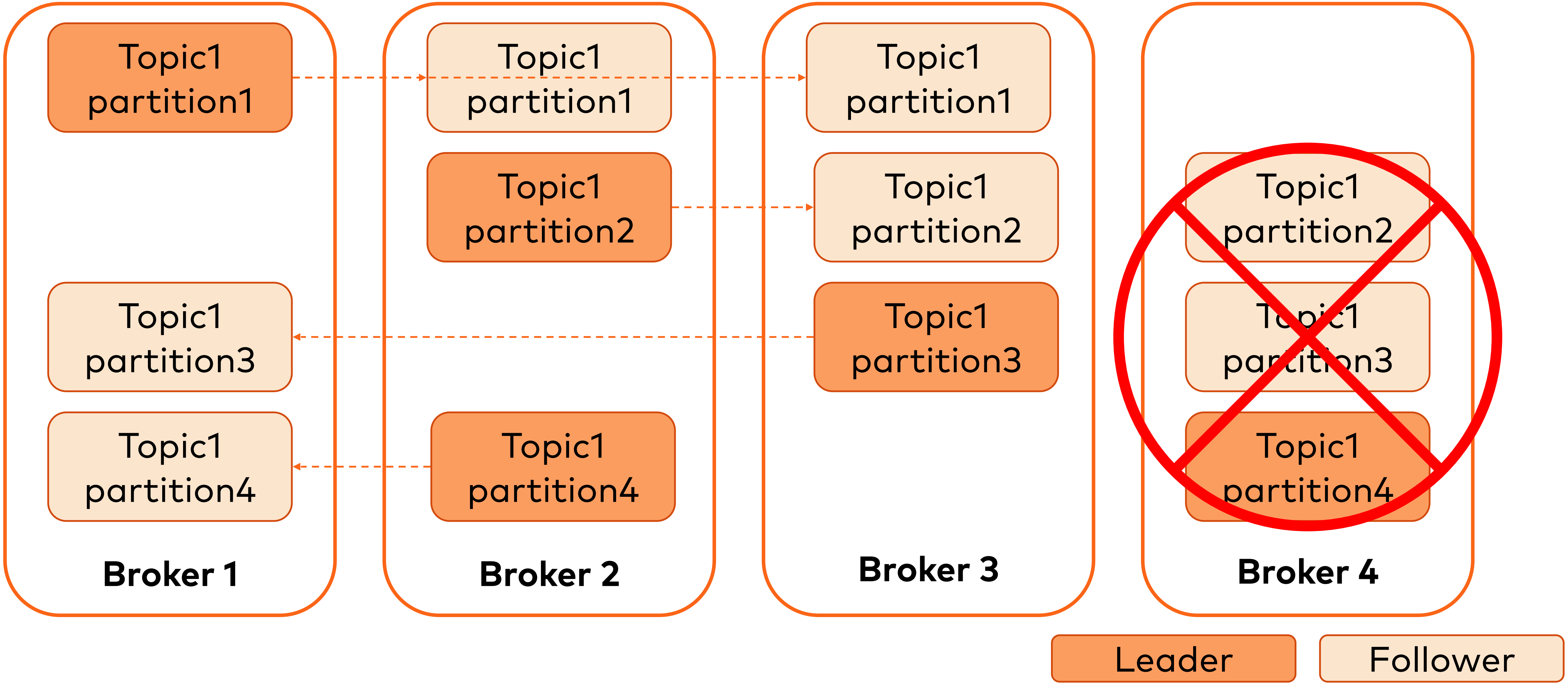


REPLICATION PROVIDES RESILIENCY



A replica takes over on machine failure

PARTITION LEADERSHIP AND REPLICATION - NODE FAILURE



THE LOG IS A TYPE OF DURABLE MESSAGING SYSTEM

Similar to a traditional messaging system (ActiveMQ, Rabbit etc) but with:

(a) Far better scalability

(b) Built in fault tolerance / HA

(c) Storage

STOP! DEMO TIME!

PROCESSING

Streaming

is the **toolset** for dealing with **events**

as **they move!**

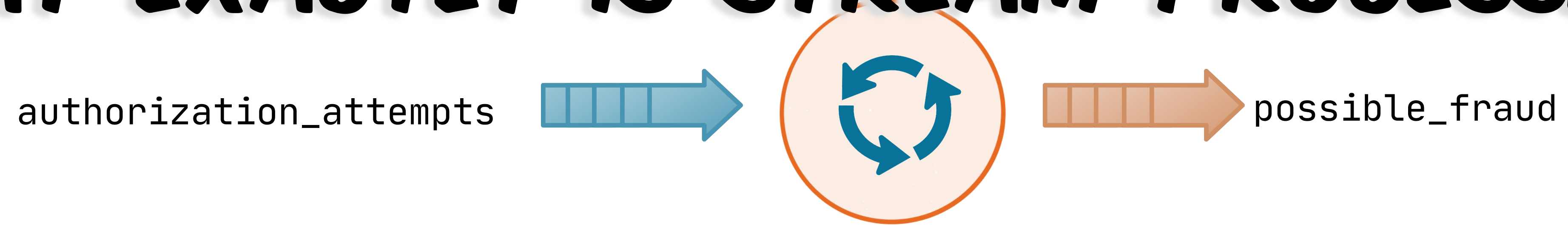
WHAT EXACTLY IS STREAM PROCESSING?

authorization_attempts

possible_fraud

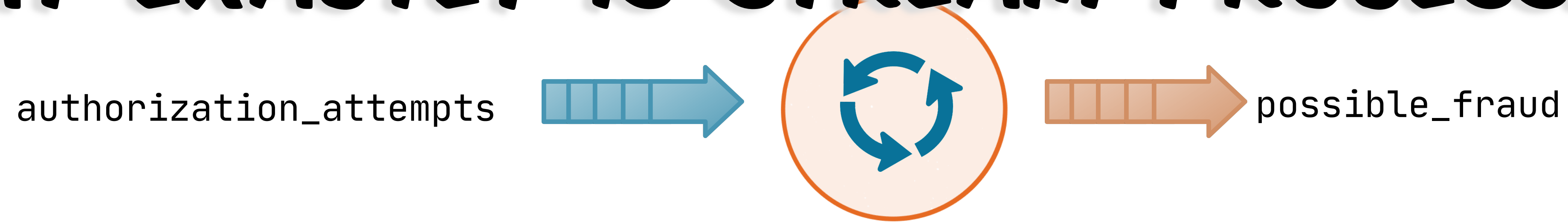
?

WHAT EXACTLY IS STREAM PROCESSING?



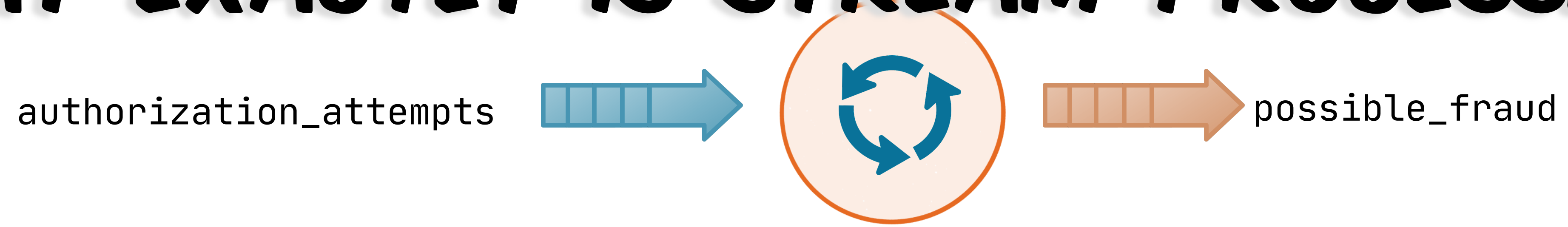

```
CREATE STREAM possible_fraud AS
SELECT card_number, count(*)
FROM authorization_attempts
WINDOW TUMBLING (SIZE 5 MINUTE)
GROUP BY card_number
HAVING count(*) > 3;
```

WHAT EXACTLY IS STREAM PROCESSING?



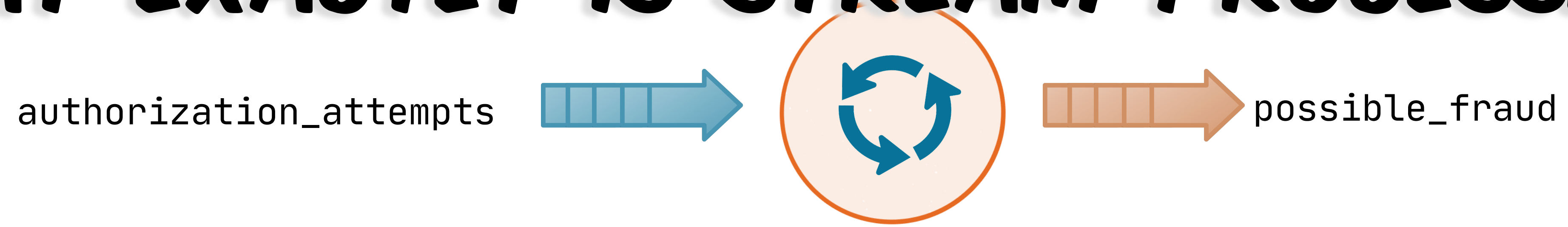

```
CREATE STREAM possible_fraud AS
SELECT card_number, count(*)
FROM authorization_attempts
WINDOW TUMBLING (SIZE 5 MINUTE)
GROUP BY card_number
HAVING count(*) > 3;
```

WHAT EXACTLY IS STREAM PROCESSING?



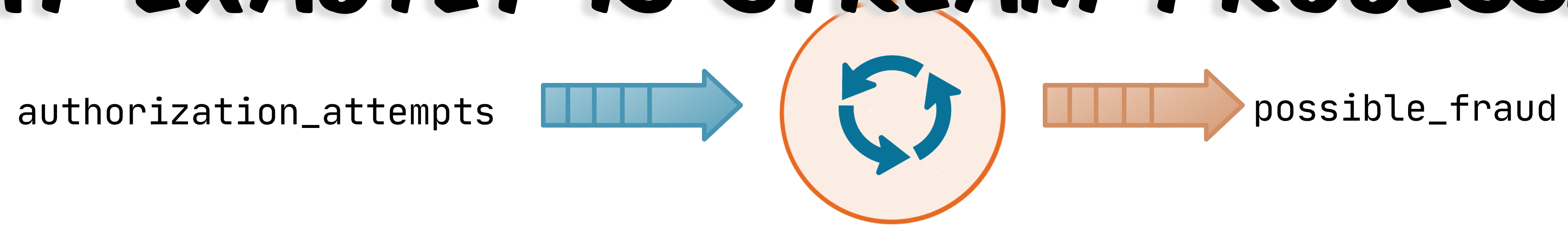

```
CREATE STREAM possible_fraud AS
SELECT card_number, count(*)
FROM authorization_attempts
WINDOW TUMBLING (SIZE 5 MINUTE)
GROUP BY card_number
HAVING count(*) > 3;
```

WHAT EXACTLY IS STREAM PROCESSING?



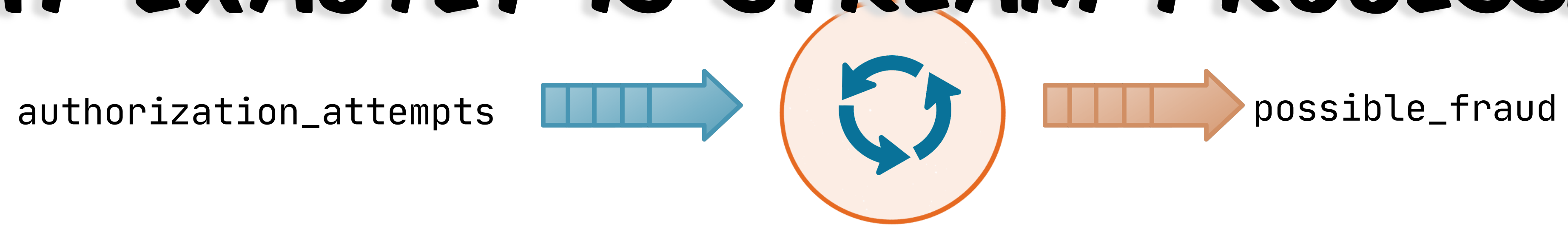

```sql
CREATE STREAM possible_fraud AS
SELECT card_number, count(*)
FROM authorization_attempts
WINDOW TUMBLING (SIZE 5 MINUTE)
GROUP BY card_number
HAVING count(*) > 3;
```

WHAT EXACTLY IS STREAM PROCESSING?



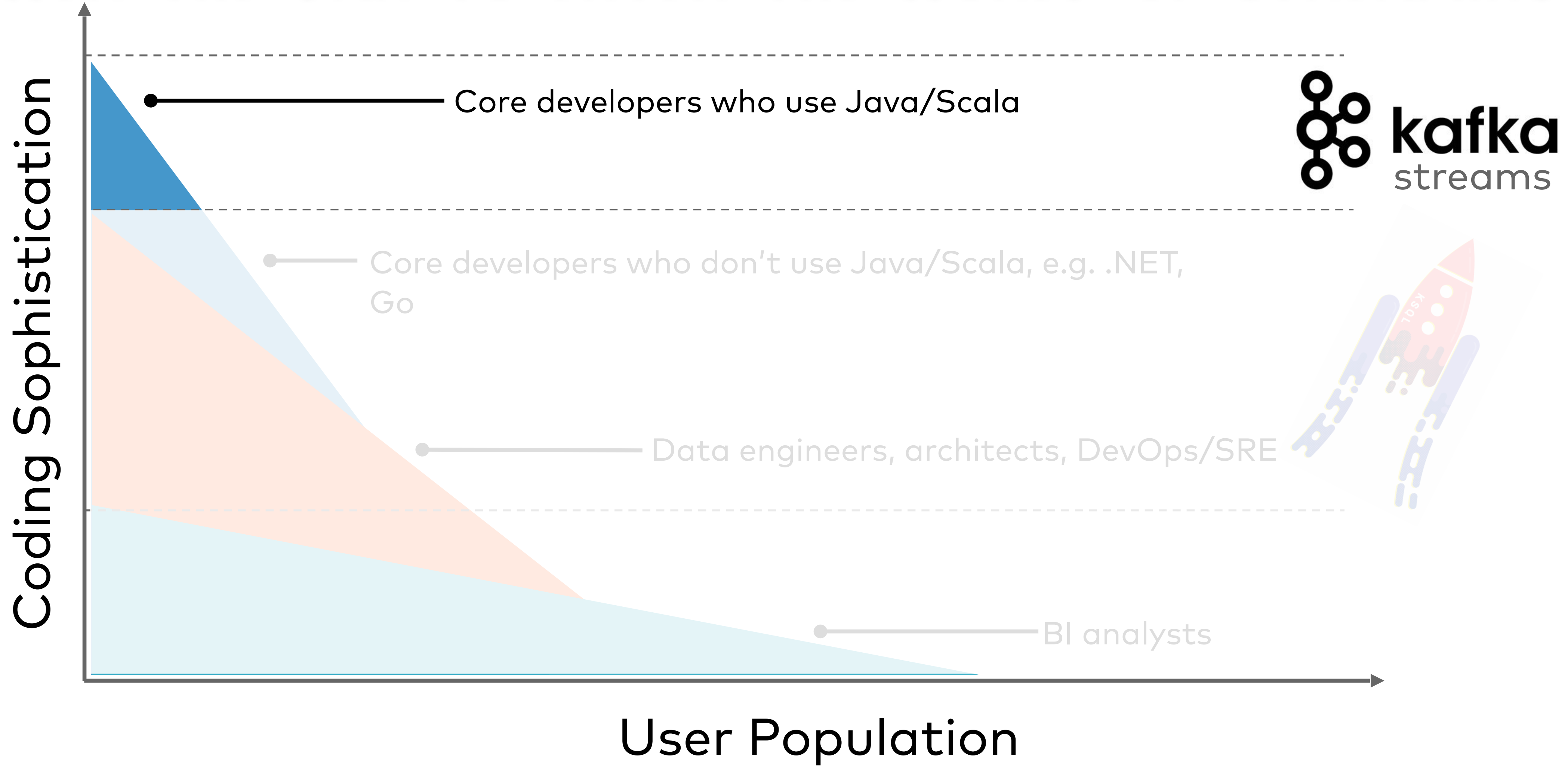

```
CREATE STREAM possible_fraud AS
SELECT card_number, count(*)
FROM authorization_attempts
WINDOW TUMBLING (SIZE 5 MINUTE)
GROUP BY card_number
HAVING count(*) > 3;
```

WHAT EXACTLY IS STREAM PROCESSING?

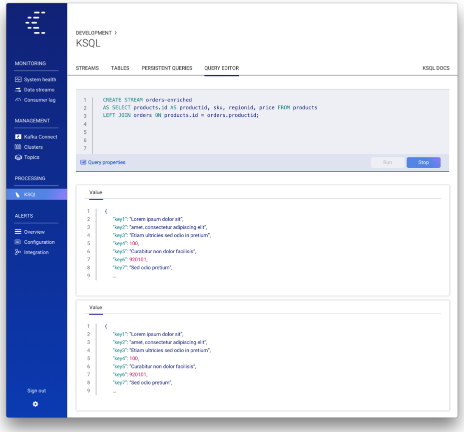



```
CREATE STREAM possible_fraud AS
SELECT card_number, count(*)
FROM authorization_attempts
WINDOW TUMBLING (SIZE 5 MINUTE)
GROUP BY card_number
HAVING count(*) > 3;
```

LOWER THE BAR TO ENTER THE WORLD OF STREAMING



KSQL #FTW



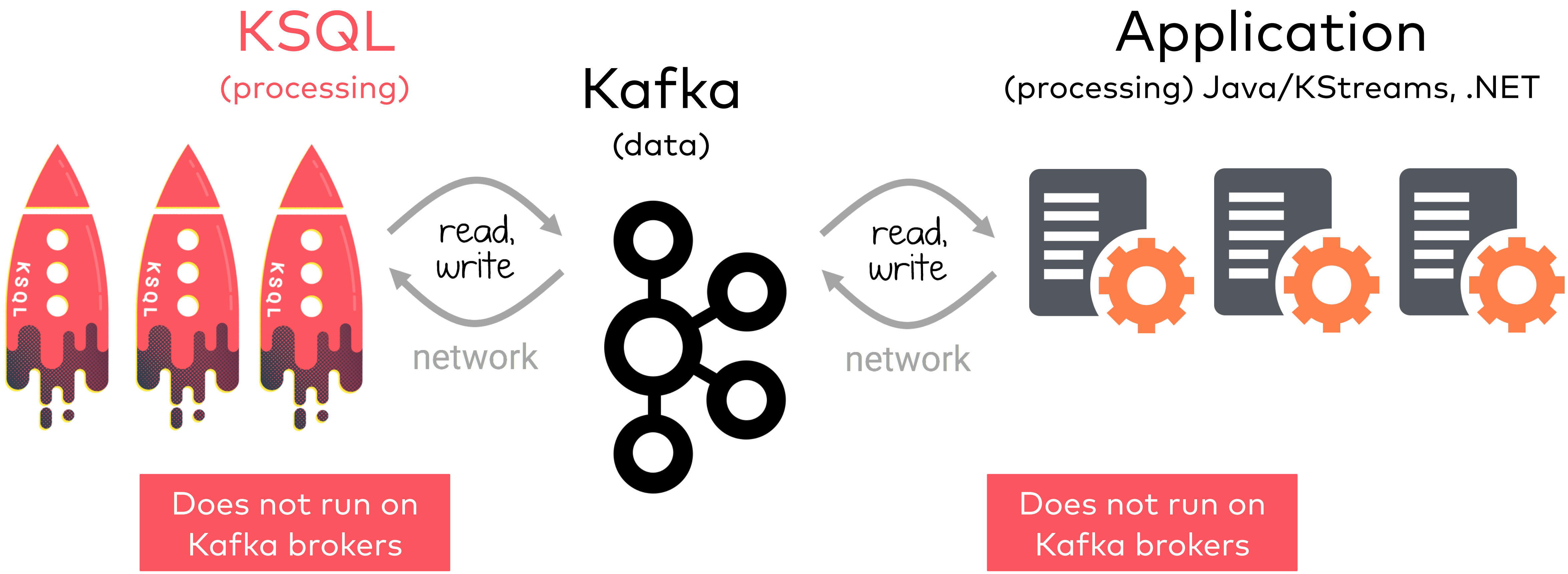
1 UI

2 CLI

3 REST

4 Headless

INTERACTION WITH KAFKA



Find your local Meetup Group

https://cnfl.io/kafka-meetups

Grab Stream Processing books

https://cnfl.io/book-bundle

Join us in Slack

http://cnfl.io/slack

THANKS!

@gamussa
viktor@confluent.io