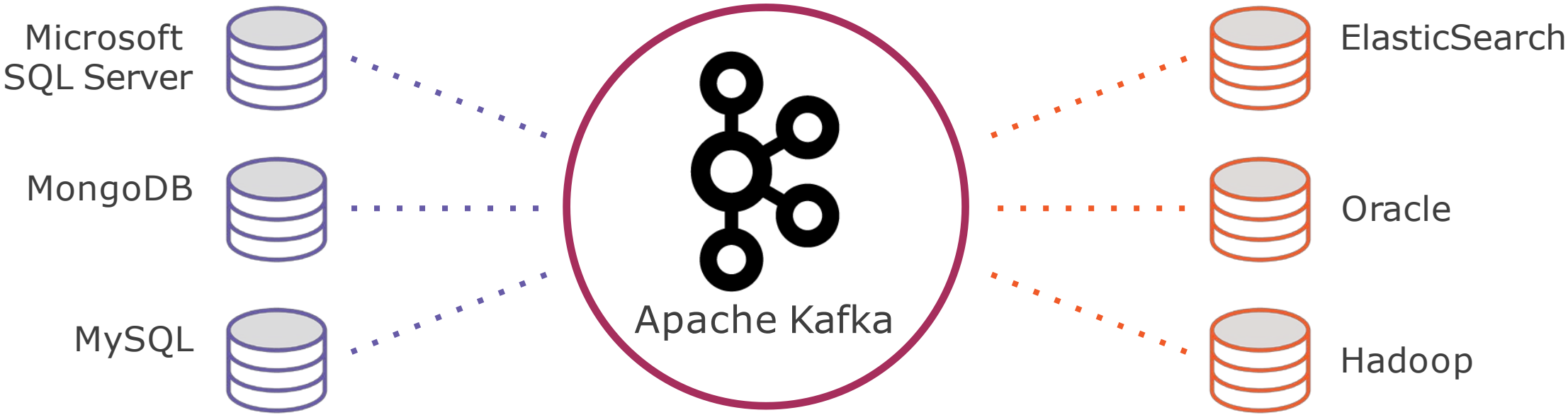
Getting Started with Apache Kafka

PluralSight Course
by Ryan Plant

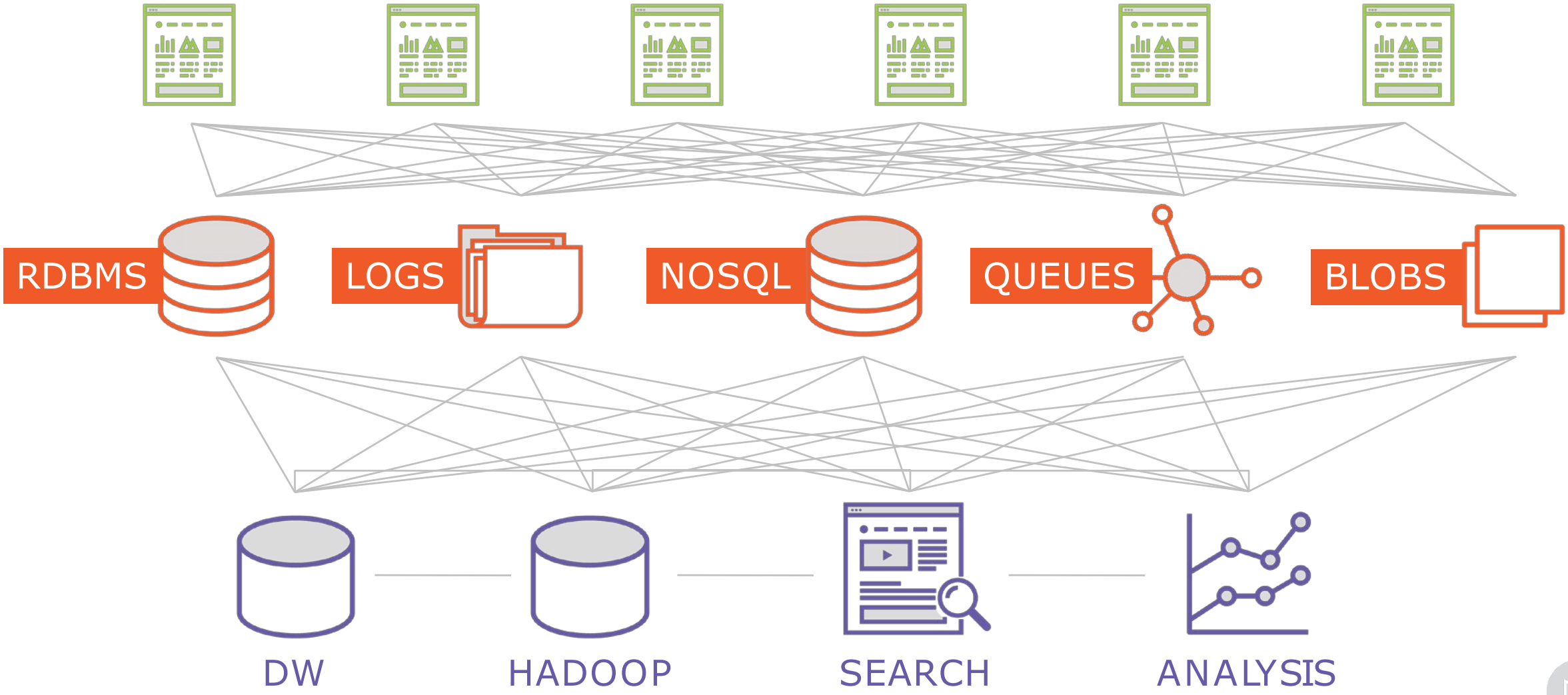
https://www.pluralsight.com/courses/apache-kafka-getting-started

What Is Apache Kafka?

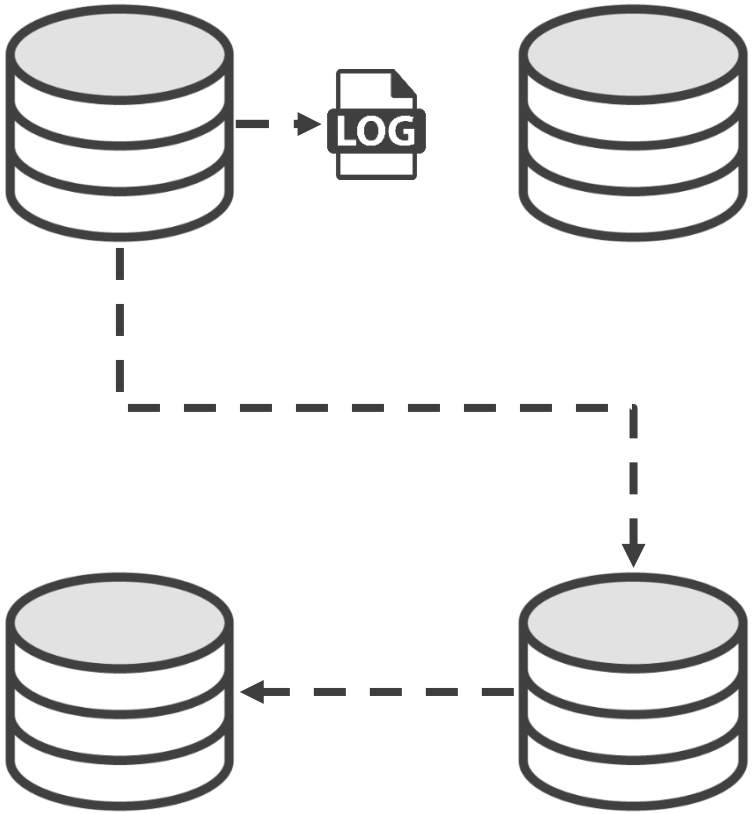


"A high-throughput distributed messaging system."

What a Typical Enterprise Looks Like



Database Replication and Log Shipping



RDBMS to RDBMS only

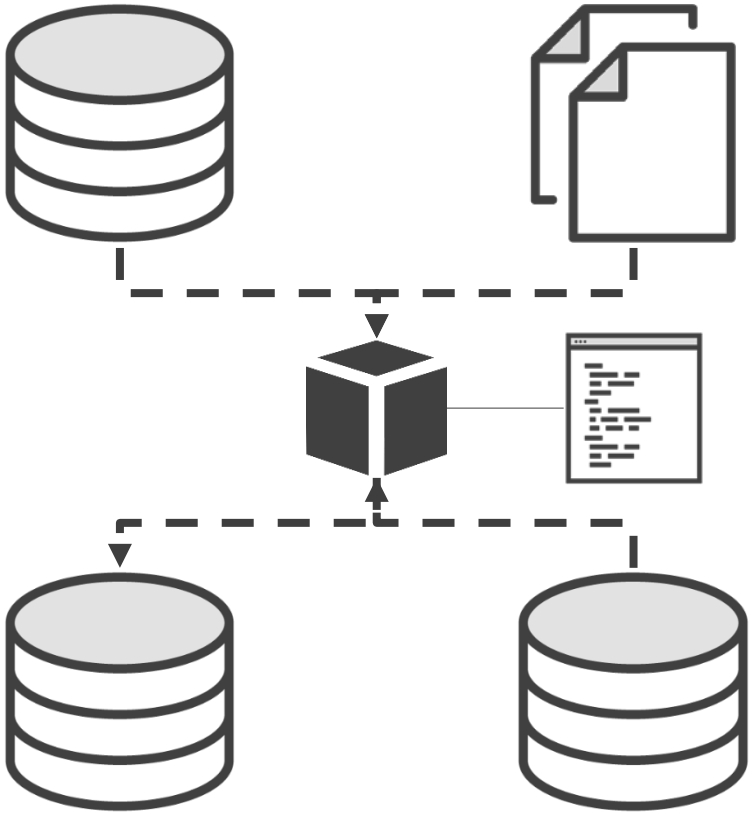
Database-specific

Tight coupling (schema)

Performance challenges (log shipping)

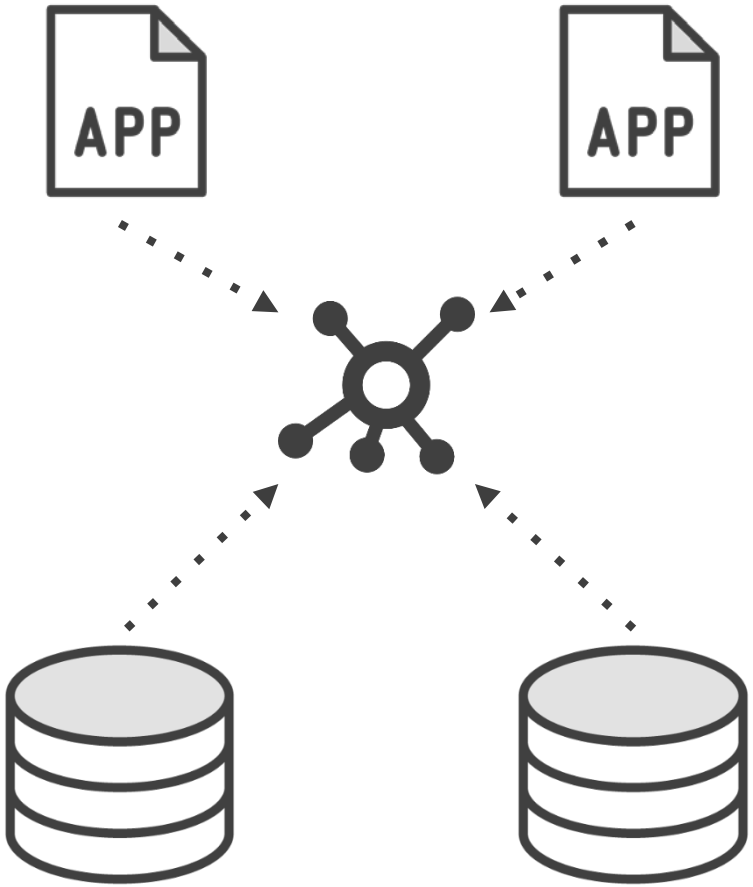
Cumbersome (subscriptions)

Extract, Transform, and Load (ETL)



- Typically proprietary and costly
- Lots of custom development
- Scalability challenged
- Performance challenged
- Often times requires multiple instances

Messaging



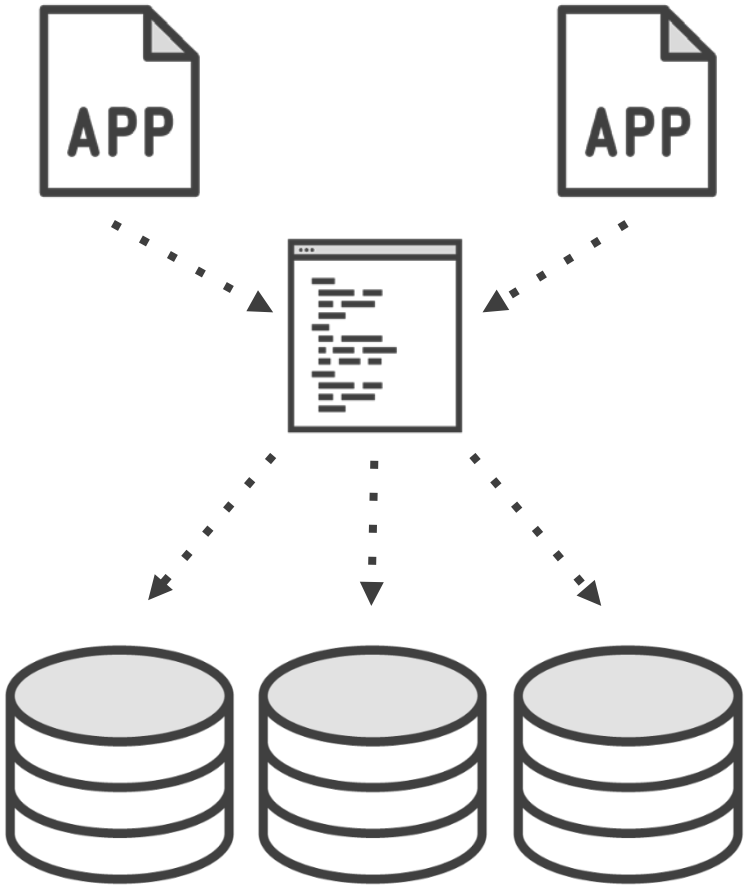
Limited scalability

Smaller messages

Requires rapid consumption

Not fault-tolerant (application)

Middleware Magic



Increasingly complex

Deceiving

Consistency concerns

Potentially expensive

Isn’t There a Better Way?

To move data around:

- Cleanly
- Reliably
- Quickly
- Autonomously

That's What LinkedIn Asked in 2010…

High Volume:

- Over 1.4 trillion messages per day
- 175 terabytes per day
- 650 terabytes of messages consumed per day
- Over 433 million users

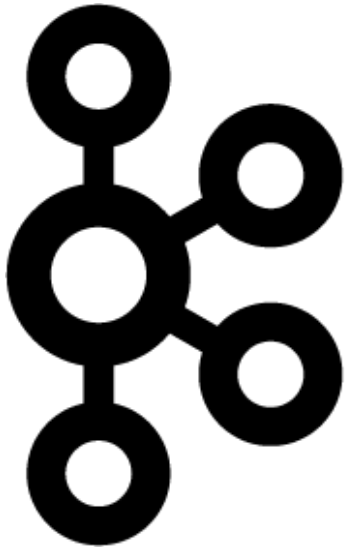
High Velocity:

- Peak 13 million messages per second
- 2.75 gigabytes per second

High Variety:

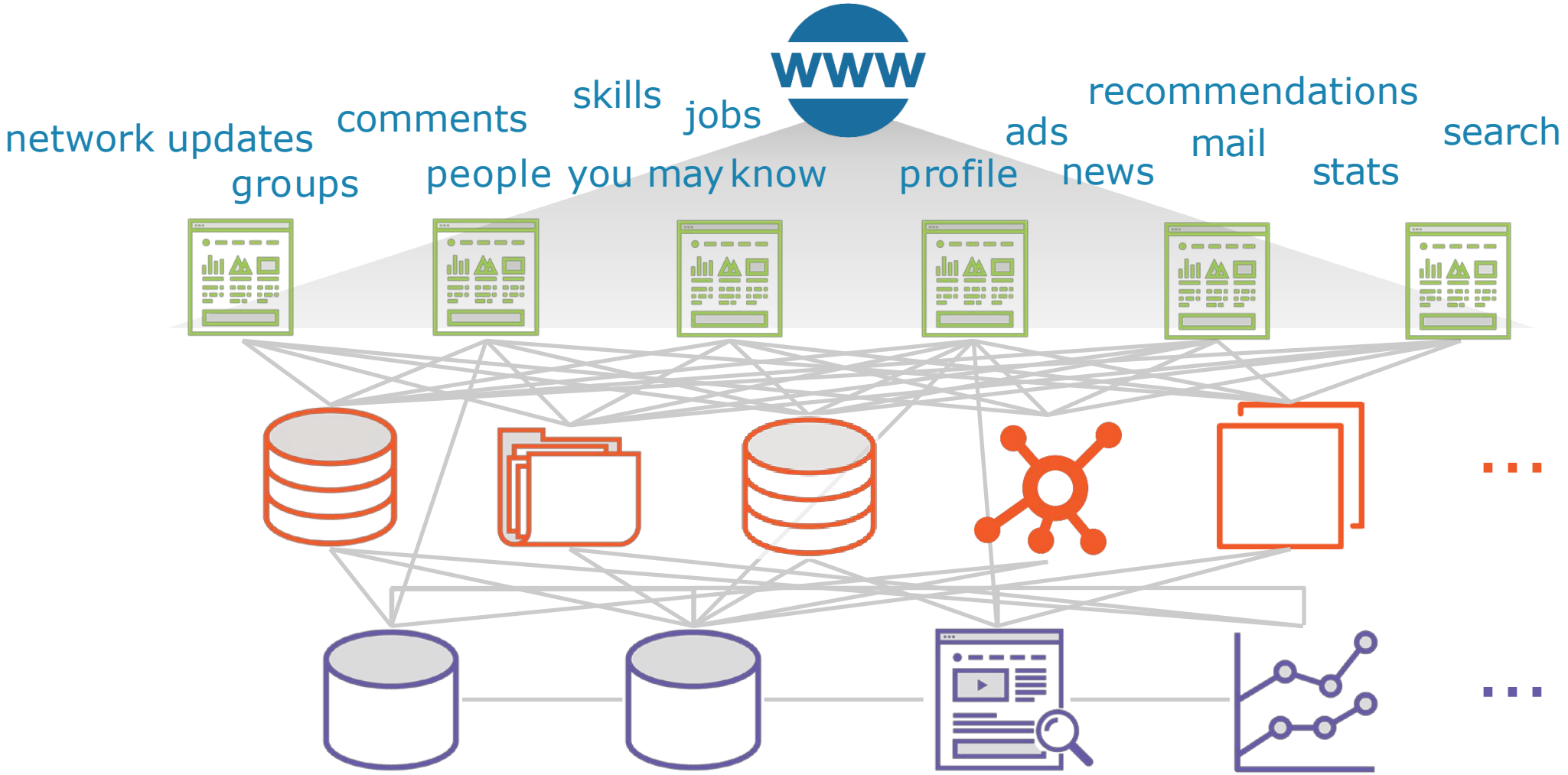
- Multiple RDBMS (Oracle, MySQL, etc.)
- Multiple NoSQL (Espresso, Voldemort)
- Hadoop, Spark, etc.

Next-generation Messaging Goals

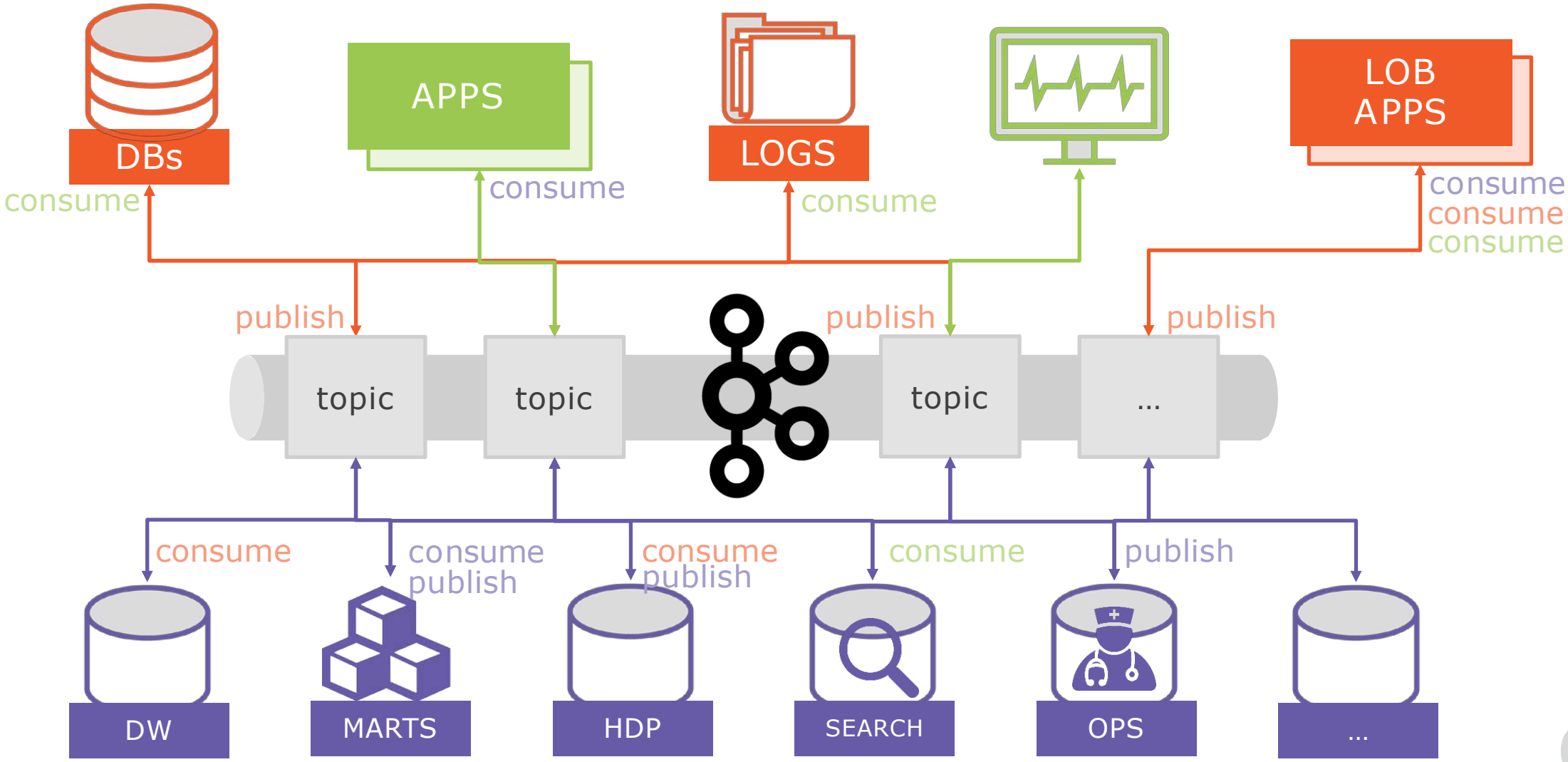


- High throughput
- Horizontally scalable
- Reliable and durable
- Loosely coupled Producers and Consumers
- Flexible publish-subscribe semantics

Pre-2010 LinkedIn Data Architecture



Post-2010 LinkedIn Data Architecture



Timeline of Events

2003
LinkedIn Launch

2009
Kafka Inception
Development begins

2010
Initial Kafka
Deployment @
LinkedIn

2011
Kafka Open Sourced
Apache Software
Foundation

Today
1.1Trillion
messages per
day @ LinkedIn

Apache Kafka Adoption

7X since 2015

Yahoo
Etsy
Microsoft
Bing
Mailchimp

Uber
Oracle
Goldman Sachs
Netflix
PayPal

Square
Coursera
IBM
Pintrest
Twitter

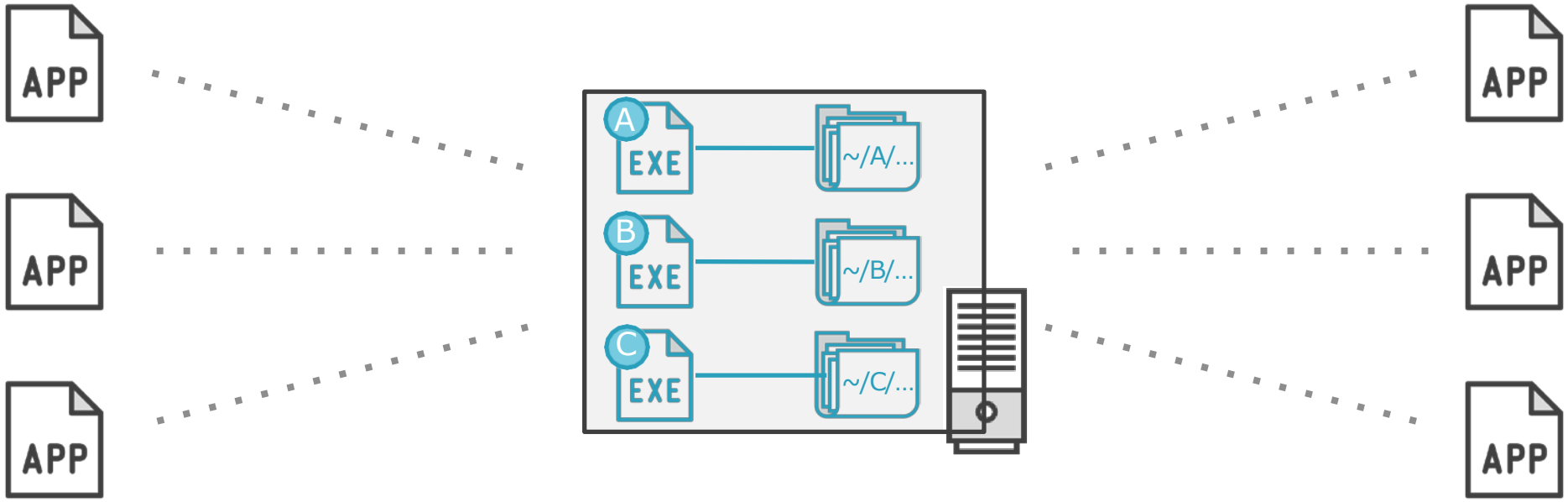
Airbnb
Spotify
Ancestry
LinkedIn
Hotels.com

Apache Kafka as a Messaging System

Producers

Broker

Consumers

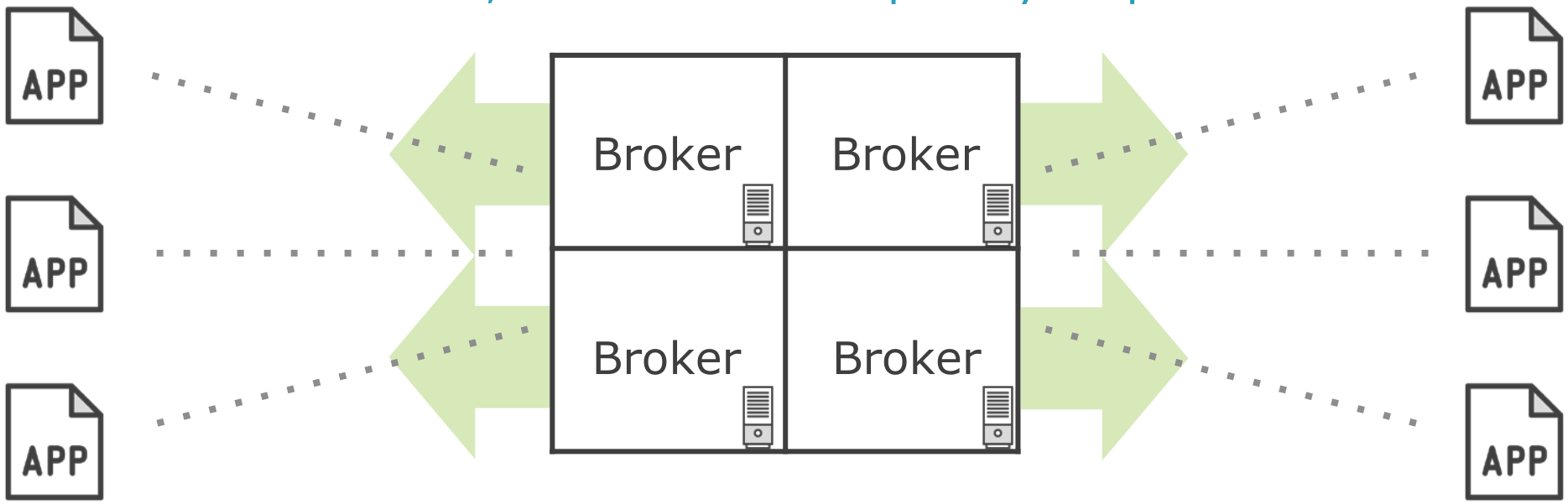


How Apache Kafka Starts to Differentiate

Producers

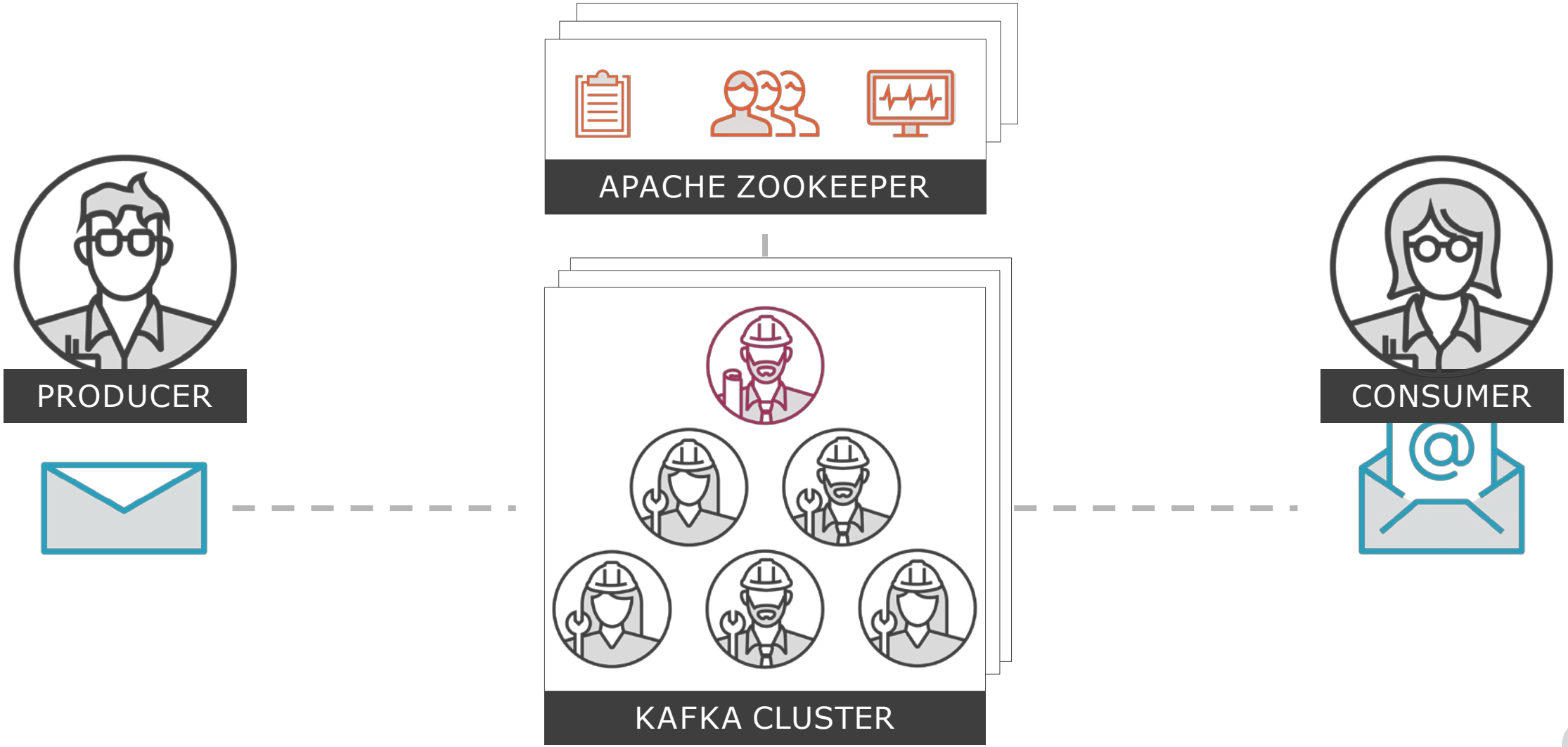
Consumers

LinkedIn: 1,400 brokers =>2 petabytes per week



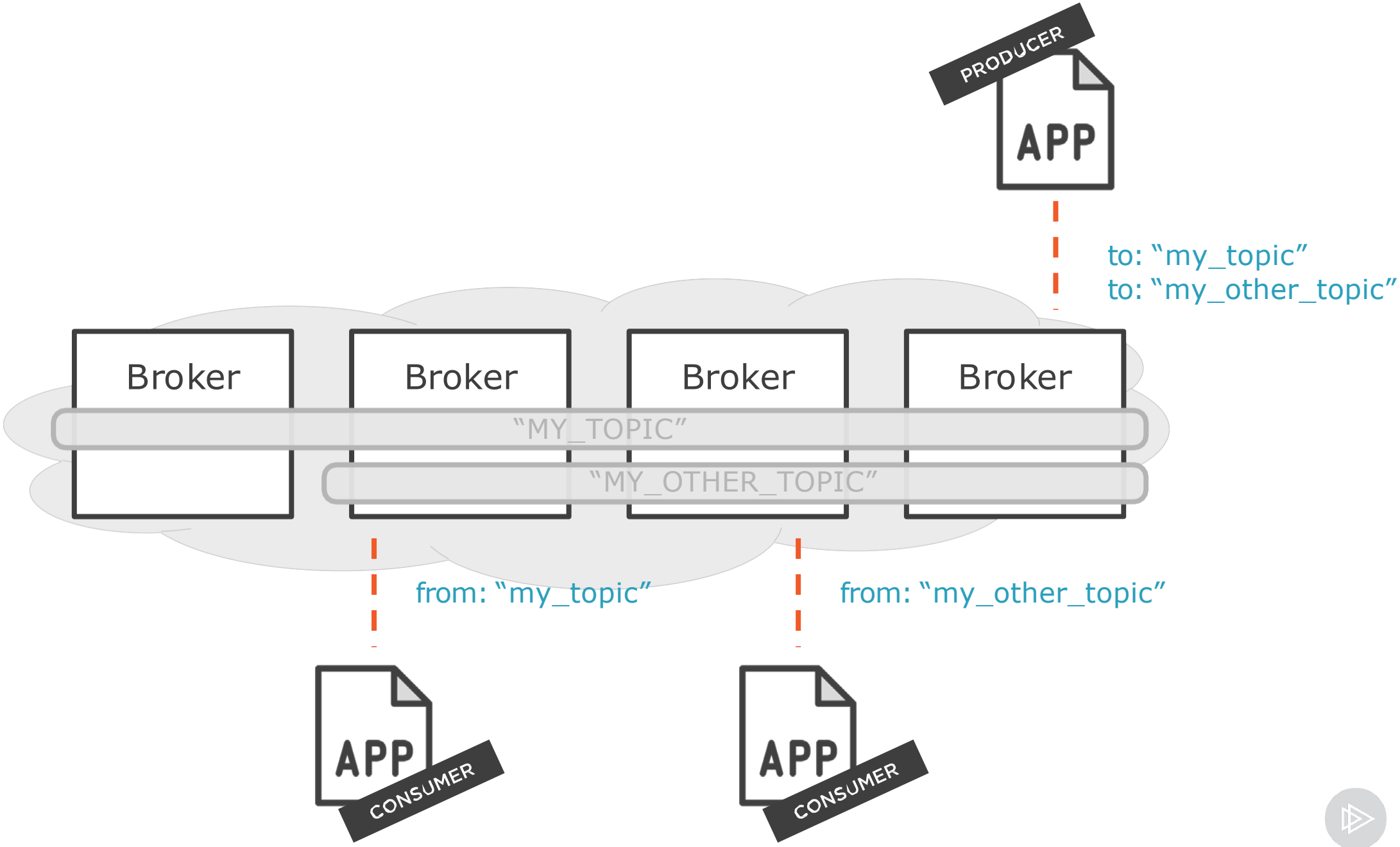
"A high-throughput distributed messaging system."

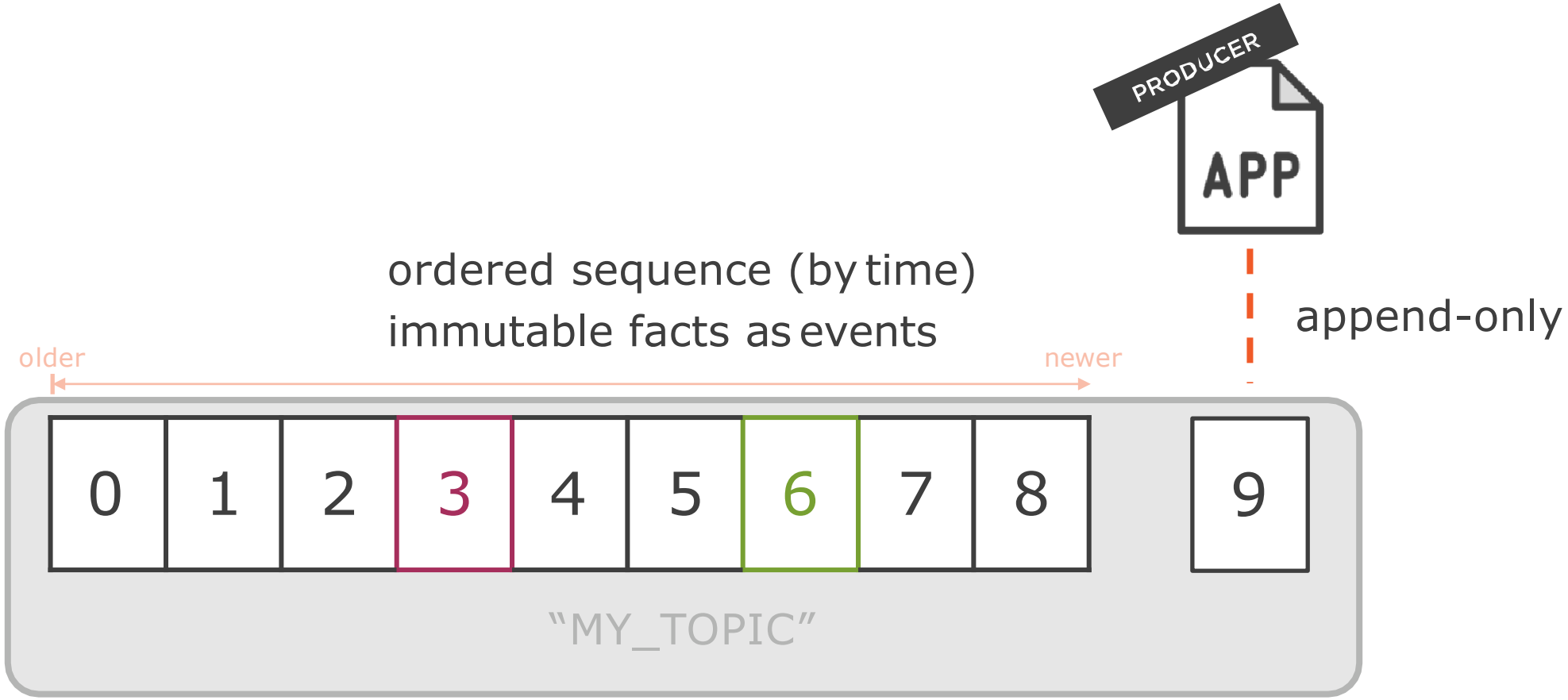
Apache Kafka's Distributed Architecture



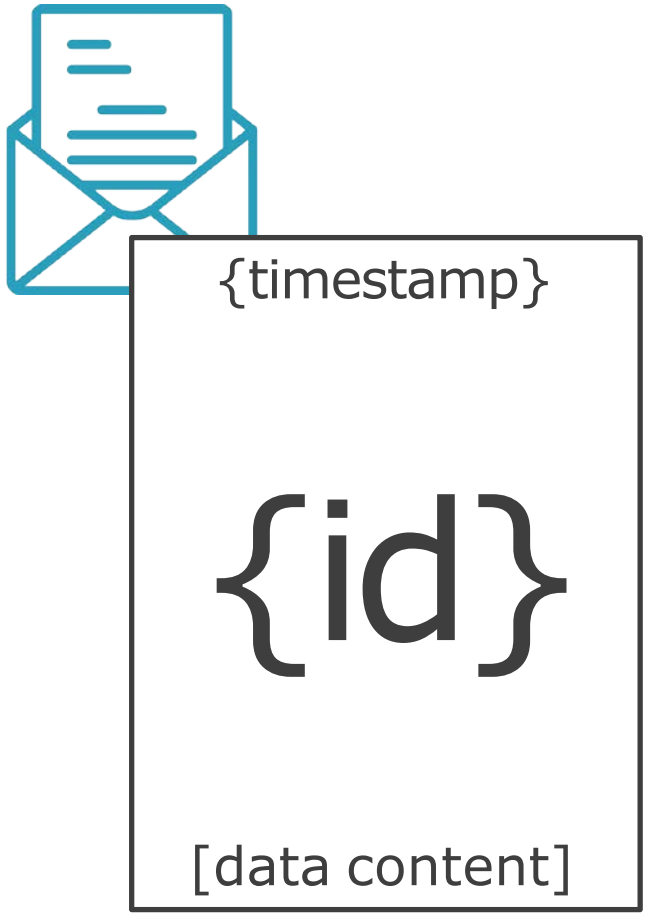
Event Sourcing

An architectural style or approach to maintaining an application's state by capturing all changes as a sequence of time-ordered, immutable events.


PRODUCER
APP
to: “my_topic”
to: “my_other_topic”
Broker
Broker
Broker
Broker
“MY_TOPIC”
“MY_OTHER_TOPIC”
from: “my_topic”
from: “my_other_topic”
APP
CONSUMER
APP
CONSUMER


PRODUCER
APP
ordered sequence (by time)
immutable facts as events
append-only
older
newer
0
1
2
3
4
5
6
7
8
9
“MY_TOPIC”

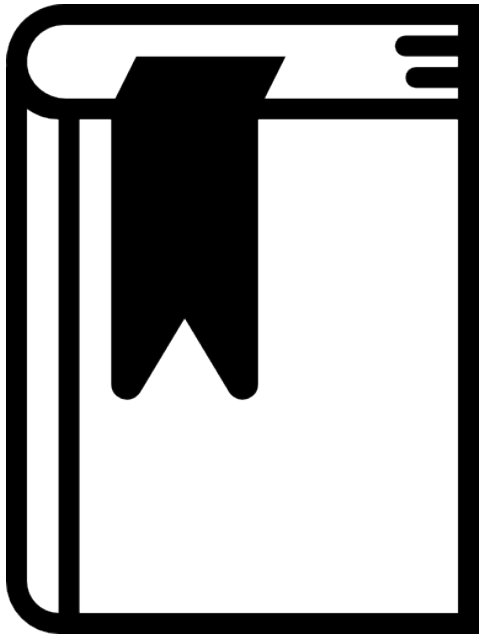
Message Content



Each message has a:

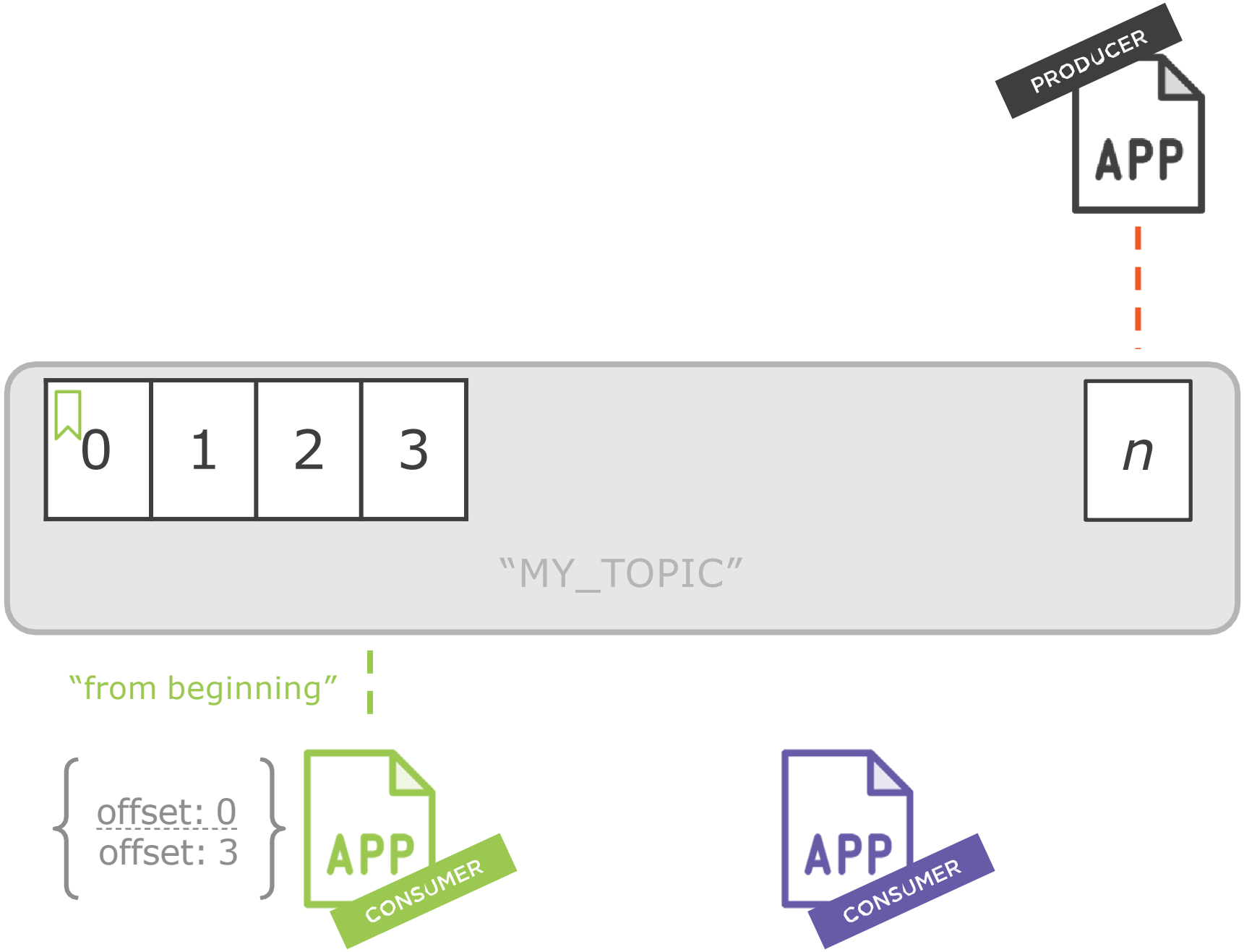
- Timestamp
- Referenceable identifier
- Payload (binary)

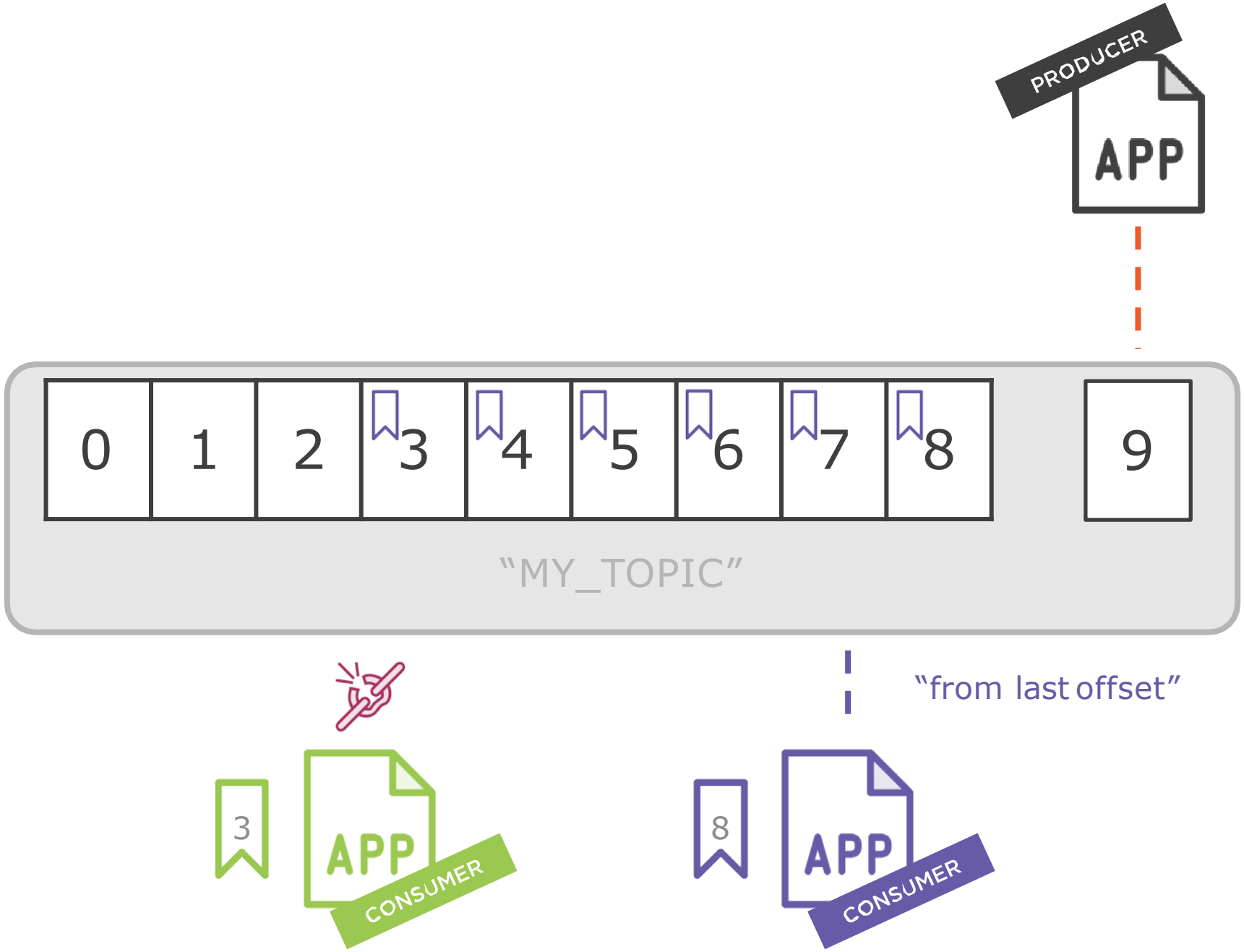
The Offset



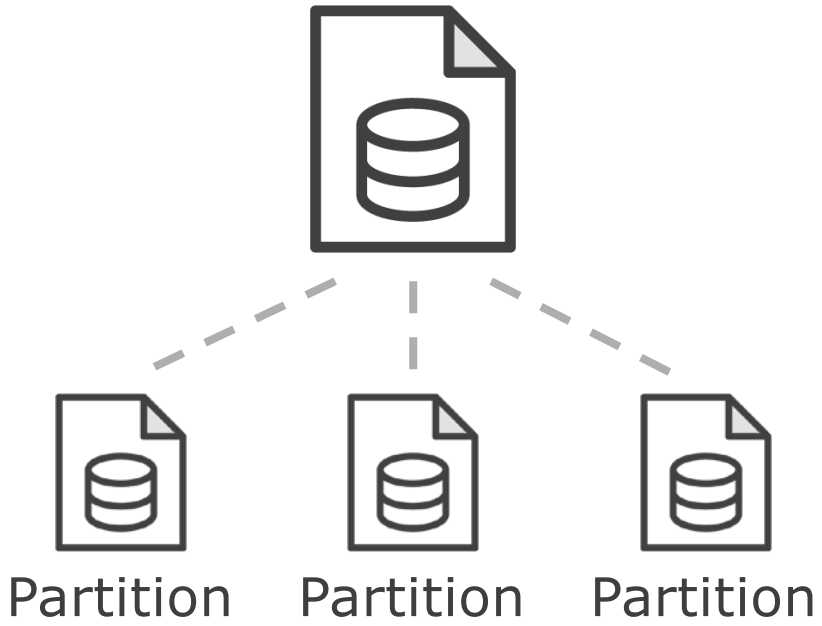
A placeholder:

- Last read message position
- Maintained by the Kafka Consumer
- Corresponds to the message identifier


PRODUCER
APP
0
1
2
3
n
"MY_TOPIC"
"from beginning"
offset: 0
offset: 3
APP
CONSUMER
APP
CONSUMER


PRODUCER
APP
0
1
2
3
4
5
6
7
8
9
"MY_TOPIC"
"from last offset"
3
APP
CONSUMER
8
APP
CONSUMER

Kafka Partitions



(Partition == Log)

Each topic has one or more partitions

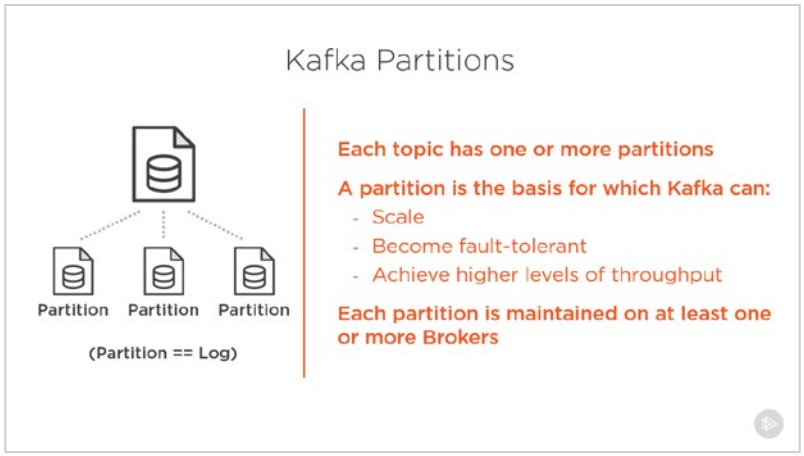
A partition is the basis for which Kafka can:

- Scale
- Become fault-tolerant
- Achieve higher levels of throughput

Each partition is maintained on at least one or more Brokers

In general, the scalability of Apache Kafka isdetermined by the number of partitions being managed by multiple broker nodes.

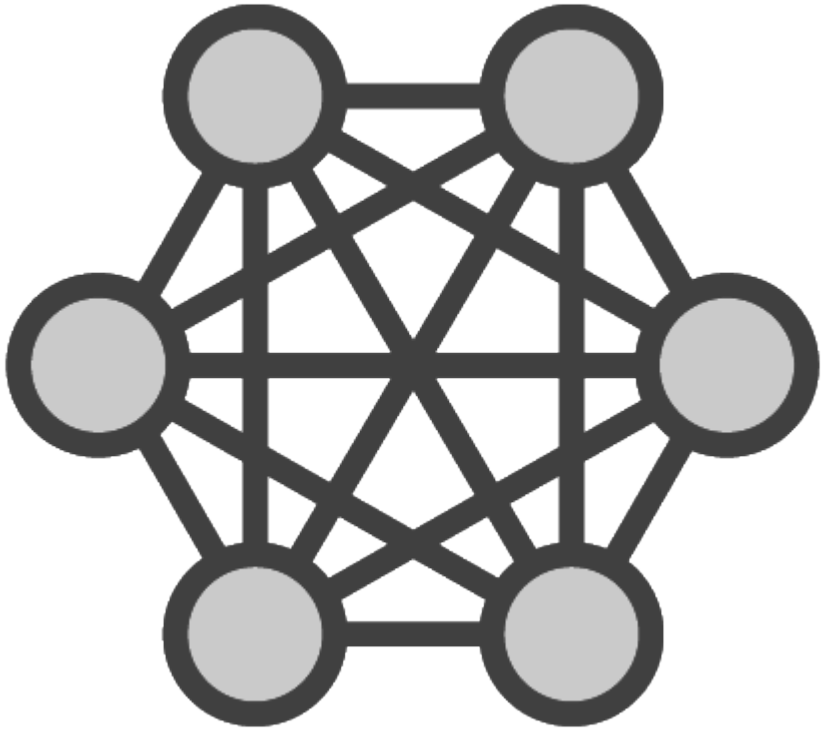
Something Is Missing



What about fault-tolerance?

- Broker failure
- Network issue
- Disk failure

Replication Factor



Reliable work distribution

- Redundancy of messages
- Cluster resiliency
- Fault-tolerance

Guarantees

- N-1 broker failure tolerance
- 2 or 3 minimum

Configured on a per-topic basis

Multiple Replica Sets

```
~$ bin/kafka-topics.sh --create --topic my_topic \
> --zookeeper localhost:2181 \
> --partitions 3 \
> --replication-factor 3
```