Introduction to Apache Kafka

- Describe the evolution of event-driven architecture
- What is Apache Kafka?
- List some use cases for Kafka
- Describe basic Kafka concepts
- Work with the Kafka command-line interface

The event-driven evolution

87%

of companies are transforming to be more customer-centric

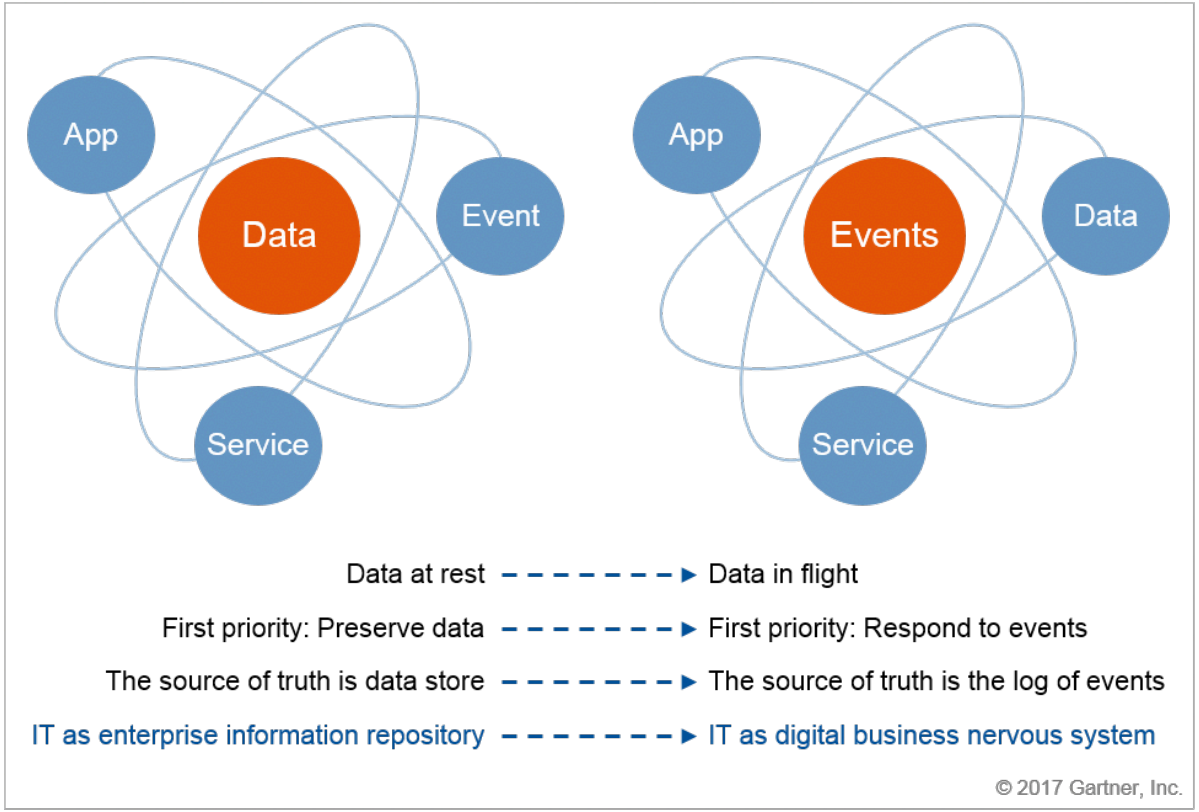
Source: A commissioned study conducted by Forrester Consulting on behalf of IBM, September 2016

Typical Event-driven Use Case | **Customer Satisfaction**

‘Zoom Air’ is a commercial airline

Re-accommodate passengers **before they realize** their journey has been disrupted

Data-centric to event-centric



Source: Gartner May 2017, “CIO Challenge: Adopt Event-Centric IT for Digital Business Success ”

Event-Driven in Action

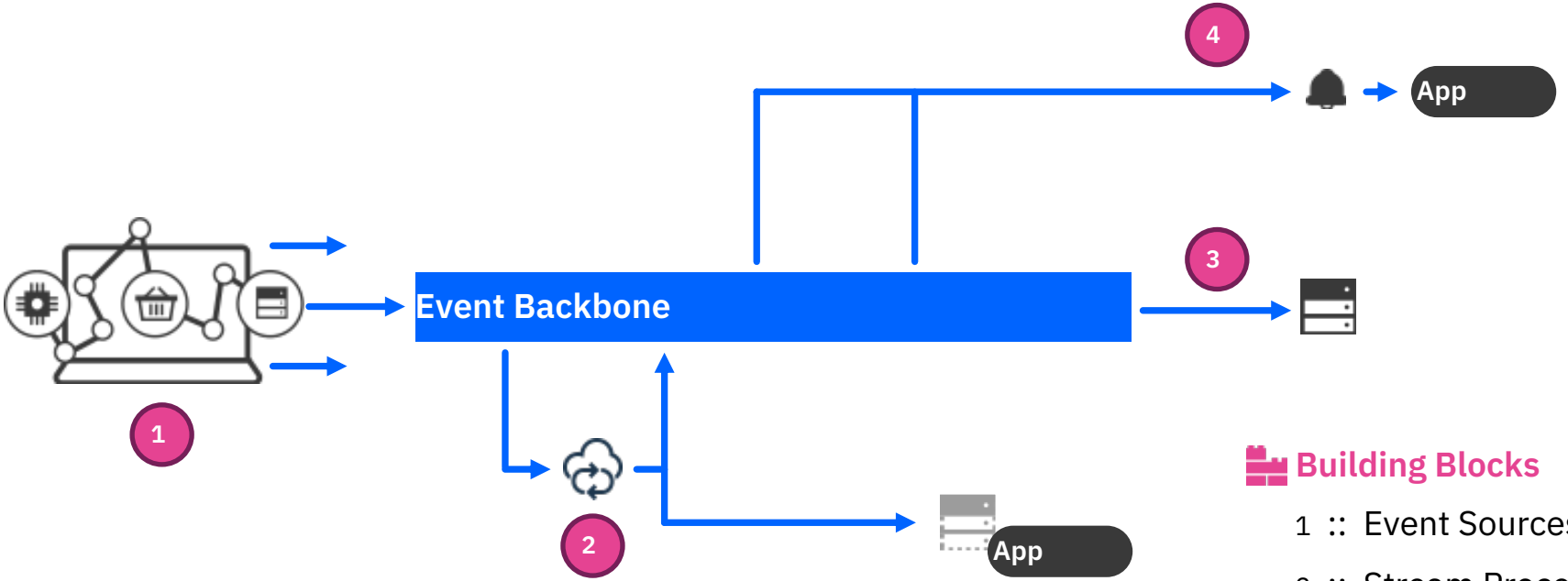
Getting data to where it's needed, before it's needed

Respond to events before the moment passes

Responsive & personalised customer experiences

Bring real time intelligence to your apps

Components of an event-streaming application



Building Blocks

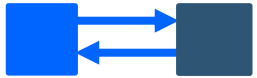
1 :: Event Sources

2 :: Stream Processing

3 :: Event Archive

4 :: Notifications

Two Styles of Messaging



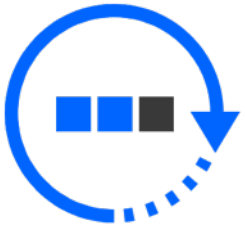
MESSAGE QUEUING

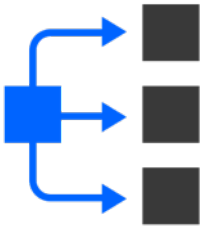
Transient data persistence

Request/reply

Targeted reliable delivery

EVENT STREAMING



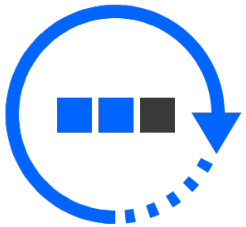


Stream history

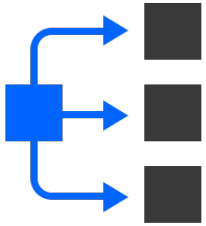
Scalable consumption

Immutable data

Properties of the Event Backbone

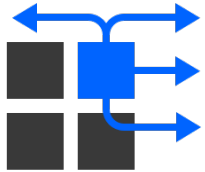


Stream history

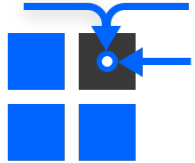


Scalable consumption

Immutable data



Scalable



Highly available

What is Apache Kafka?

What is Apache Kafka?

An open-source project, originally created by LinkedIn

Designed for real-time activity streams

Can handle hundreds of reads/writes per second from many clients

Distributed and highly scalable: designed for partitioning over a cluster

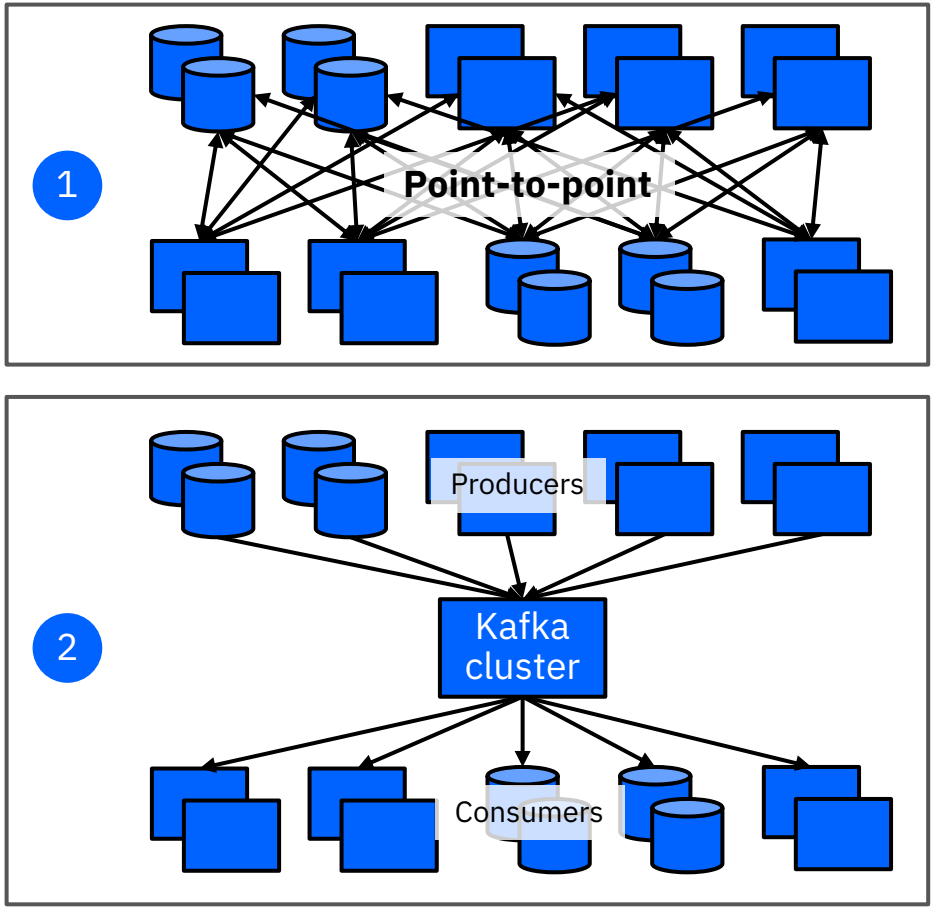
Can grow elastically and transparently with no downtime

Fault-tolerant: messages are persisted to disk and replicated within the cluster

Kafka decouples data pipeline

LinkedIn had dozens of data systems and repositories that connected in a geographically distributed point-to-point network.

Kafka was created as a centralized online data pipeline to decouple these systems.



Kafka use cases

- Messaging: publish/subscribe
- Real-time website user activity tracking
- Collecting operations metrics
- Collecting log data
- Processing data streams
- And more

Messaging: publish/subscribe

Publisher

Topic

Subscriber

Website activity tracking

Collecting operations metrics

Collecting log data

Processing data streams

Kafka basic concepts

Kafka basic concepts

Broker: a server in a Kafka cluster

Topic: a user-defined category to which messages are published

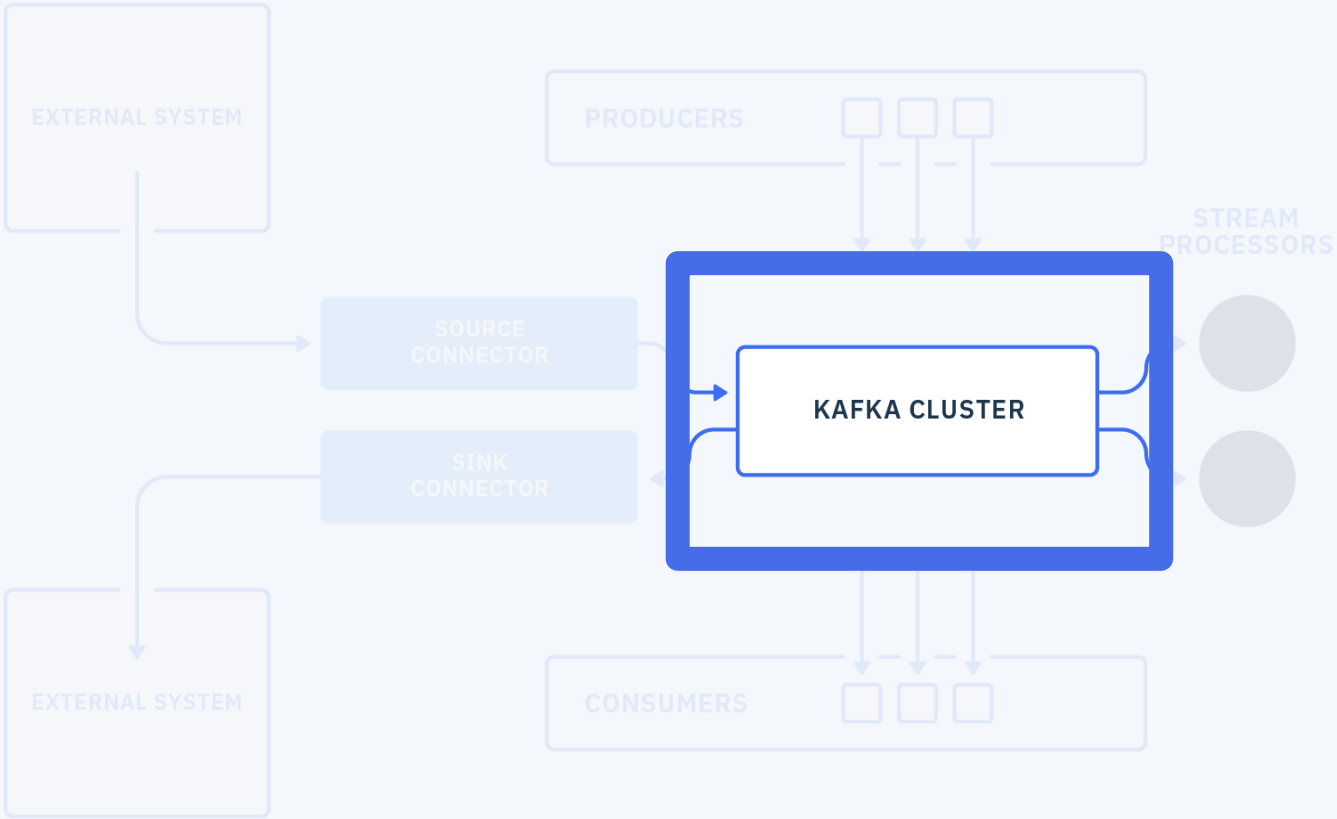
Partition: a topic is made up of one or more partitions

Replication: each topic can be replicated to multiple brokers

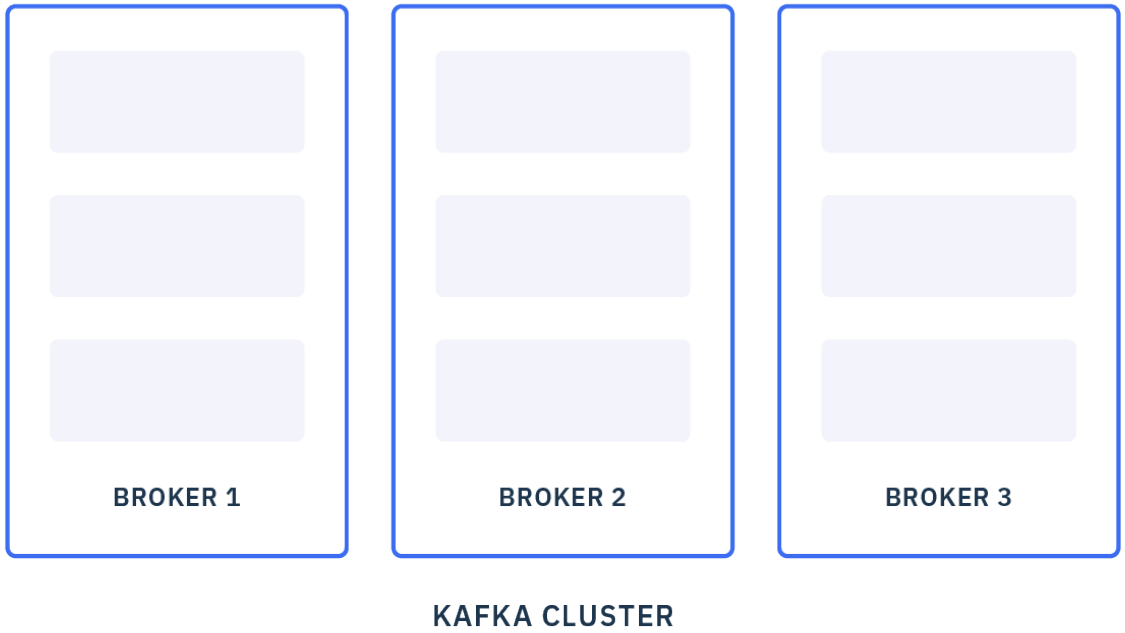
Consumers: read topics and consume messages from topic partitions

Producers: publish messages to a topic

Stream: an unbounded flow of data


EXTERNAL SYSTEM
PRODUCERS
STREAM PROCESSORS
SOURCE CONNECTOR
KAFKA CLUSTER
SINK CONNECTOR
EXTERNAL SYSTEM
CONSUMERS

Brokers



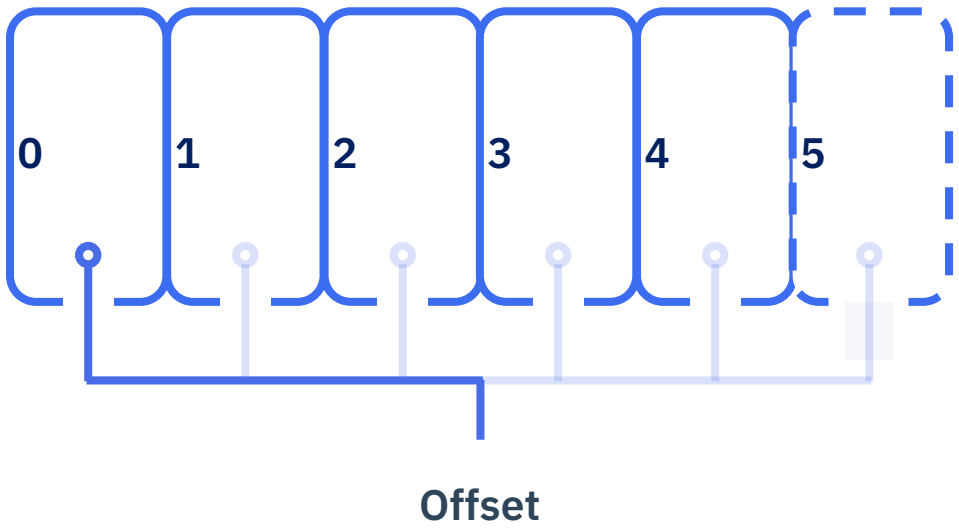
Topics

Topics are an immutable sequence of records

An individual record is made up of a key and a value

Messages are assigned a sequential ID number called an ***offset***

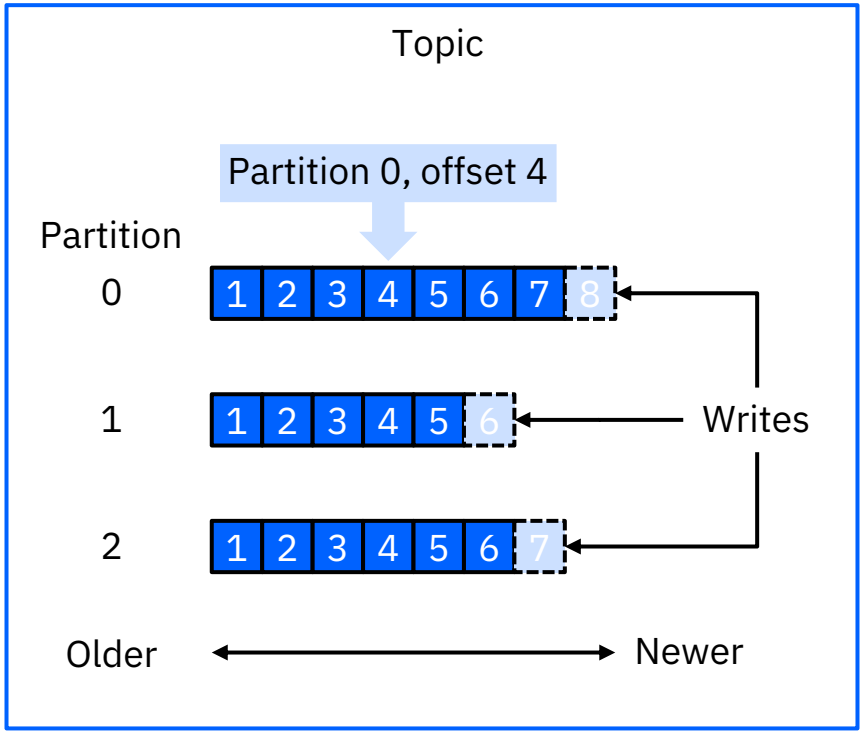
Messages are appended to the end; offset number increases



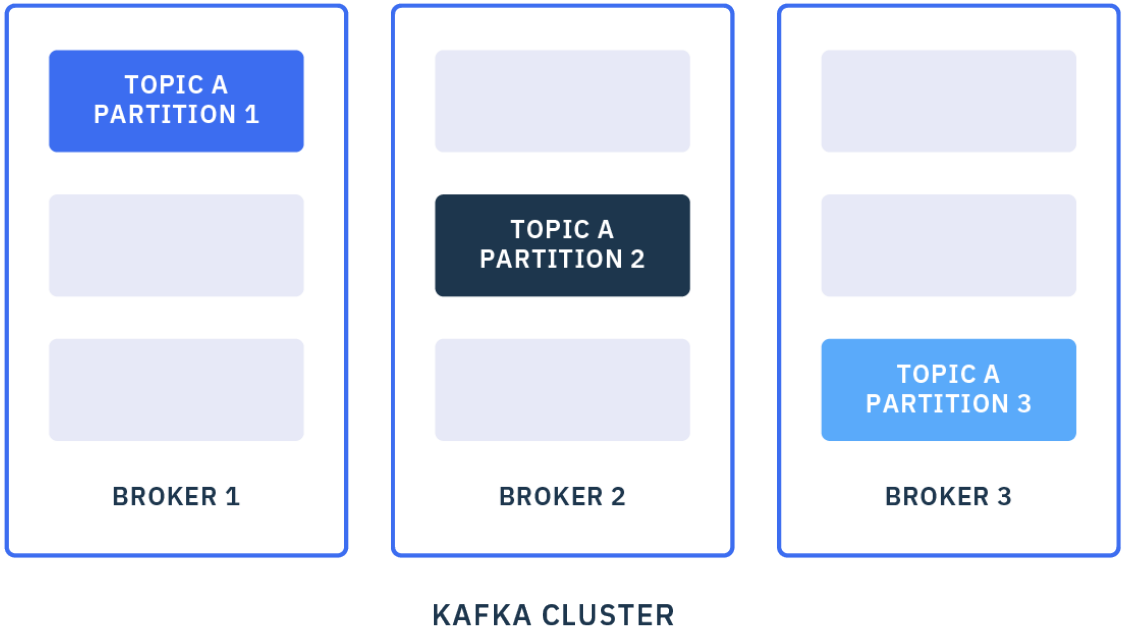
Partitions

A topic is made up of one or more partitions

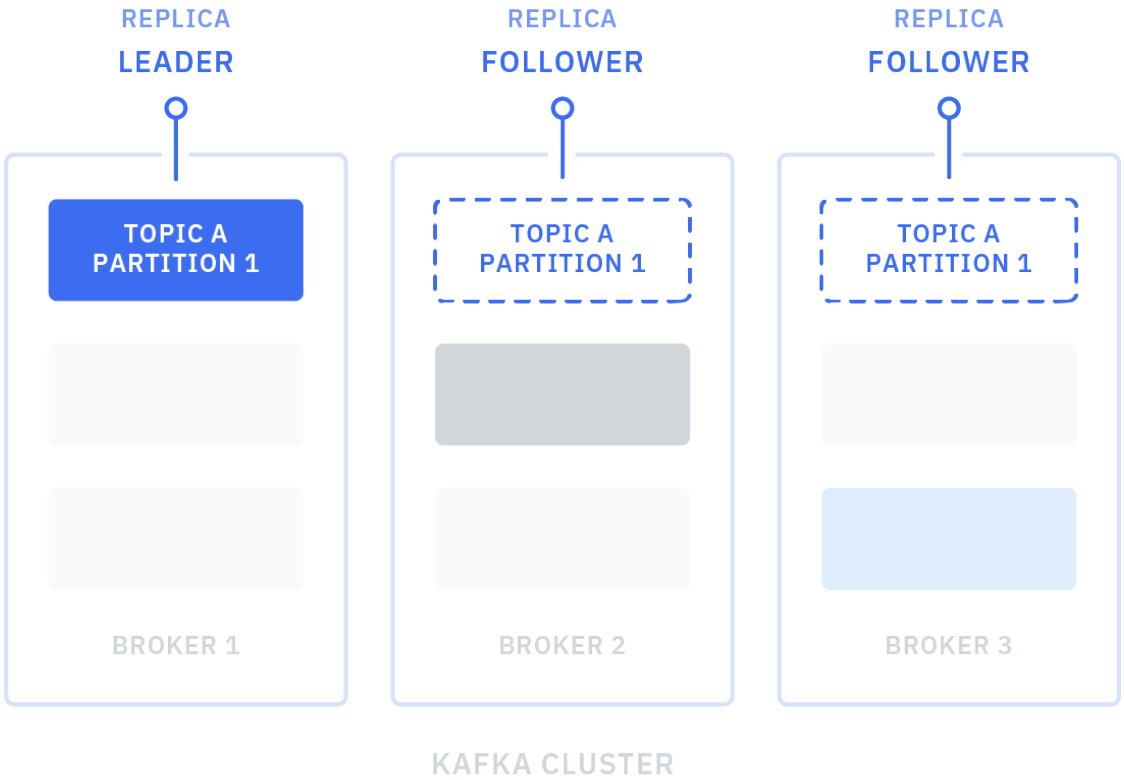
A partition is an ordered, immutable sequence of messages that is continually appended to



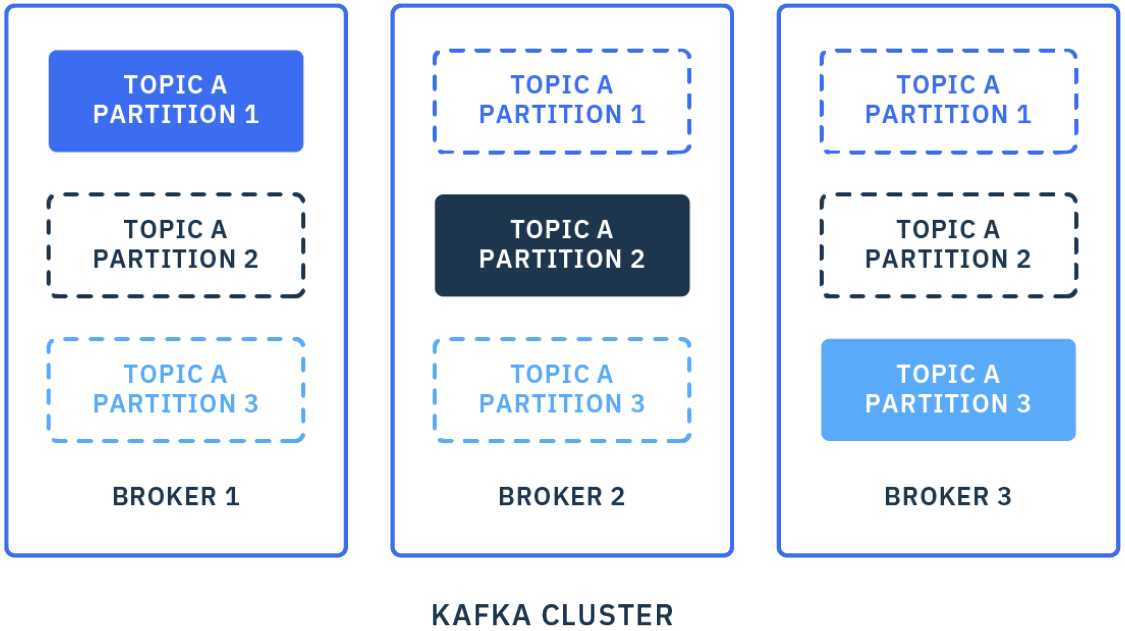
Partitions


TOPIC A
PARTITION 1
BROKER 1
TOPIC A
PARTITION 2
BROKER 2
TOPIC A
PARTITION 3
BROKER 3
KAFKA CLUSTER

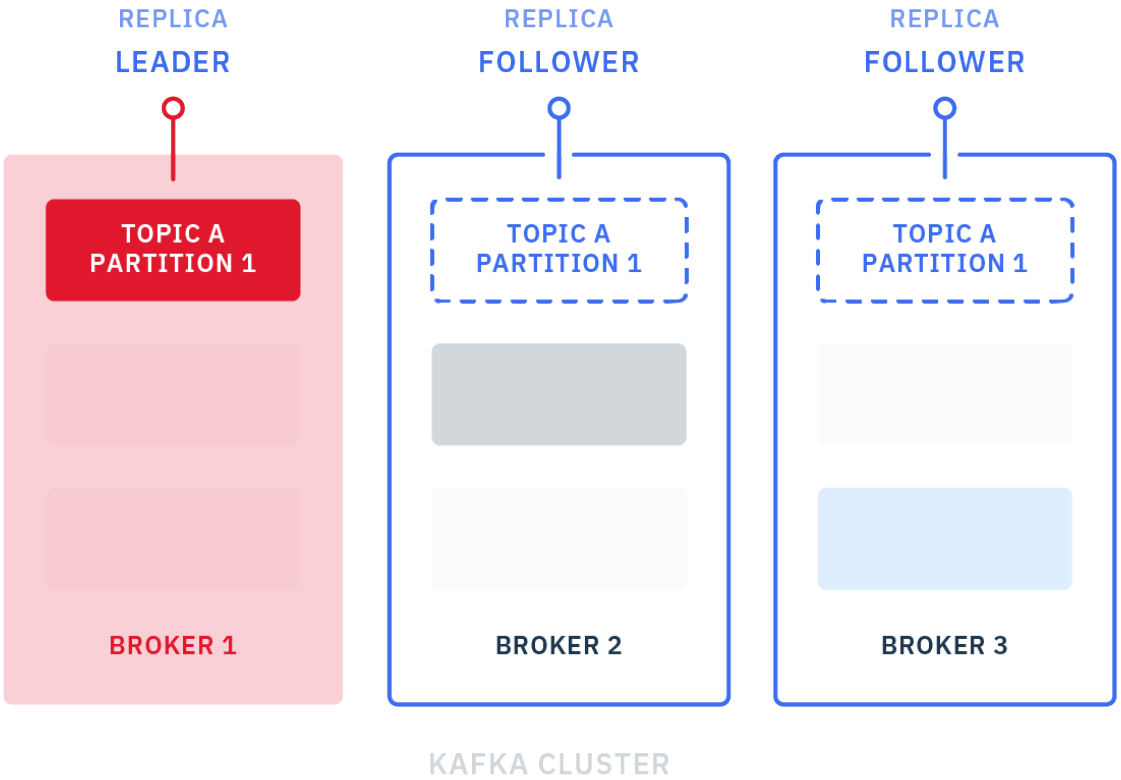
Replication



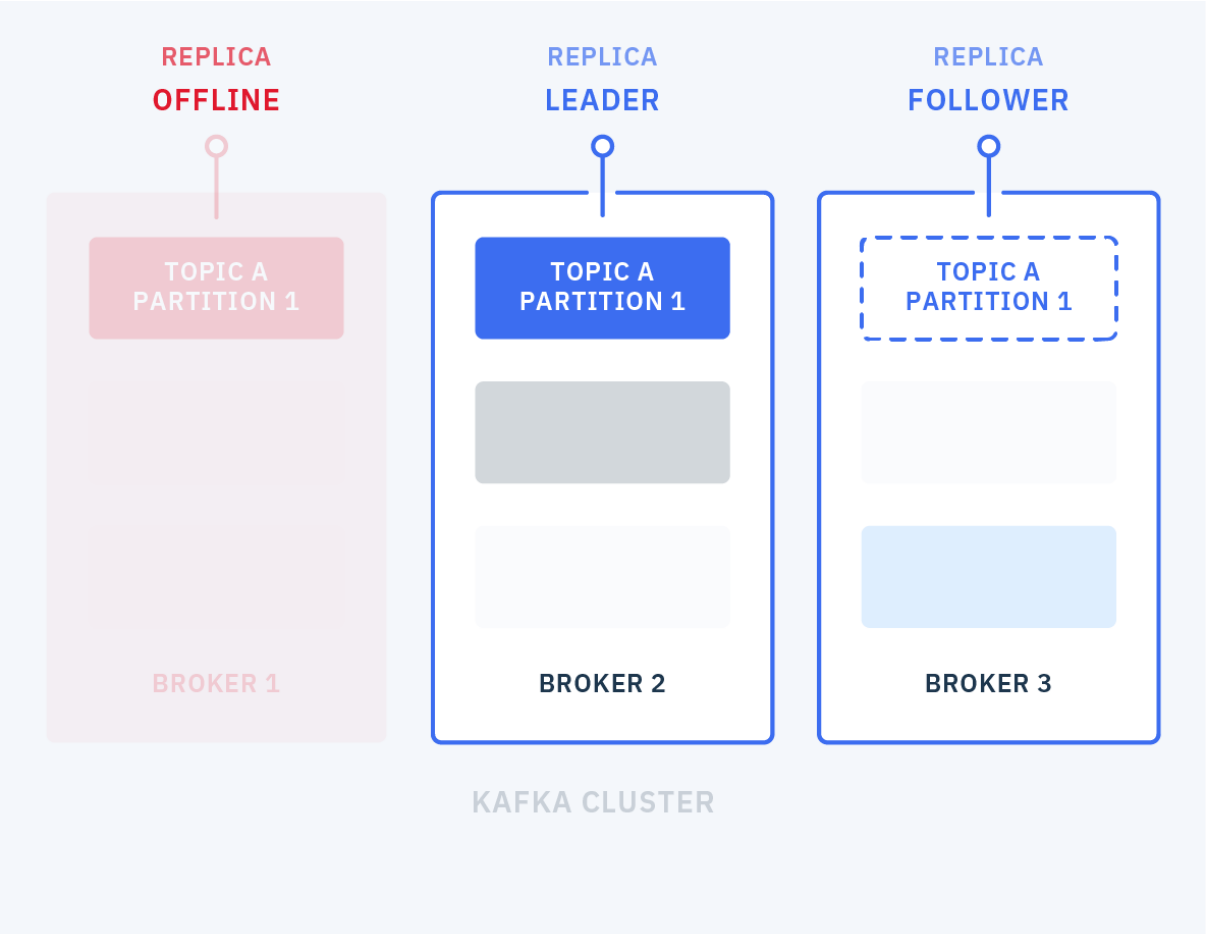
Replication



Replication



Replication

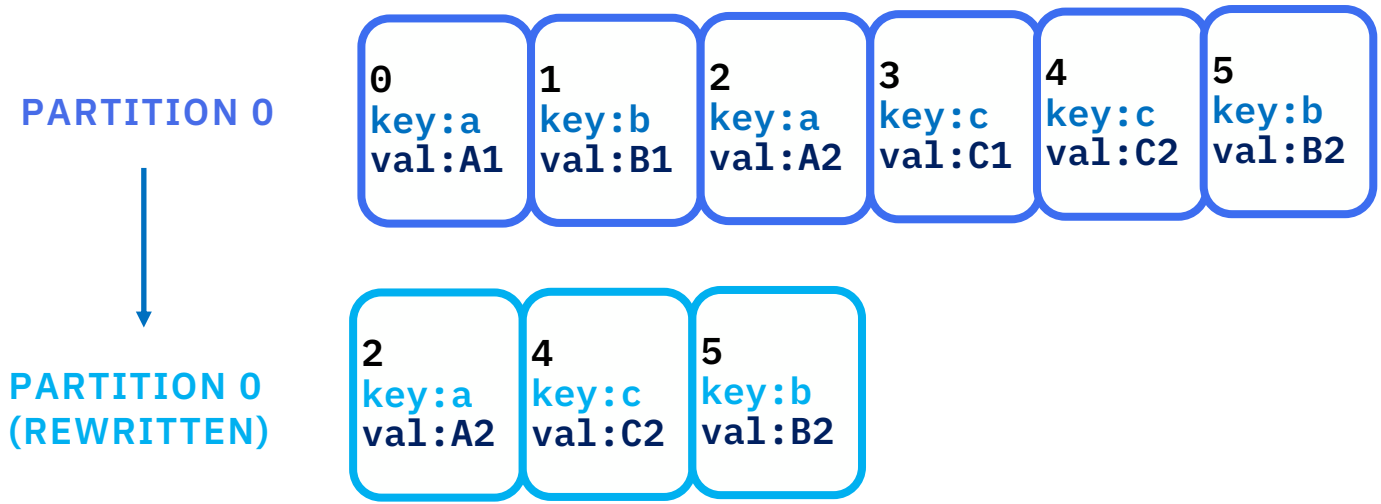


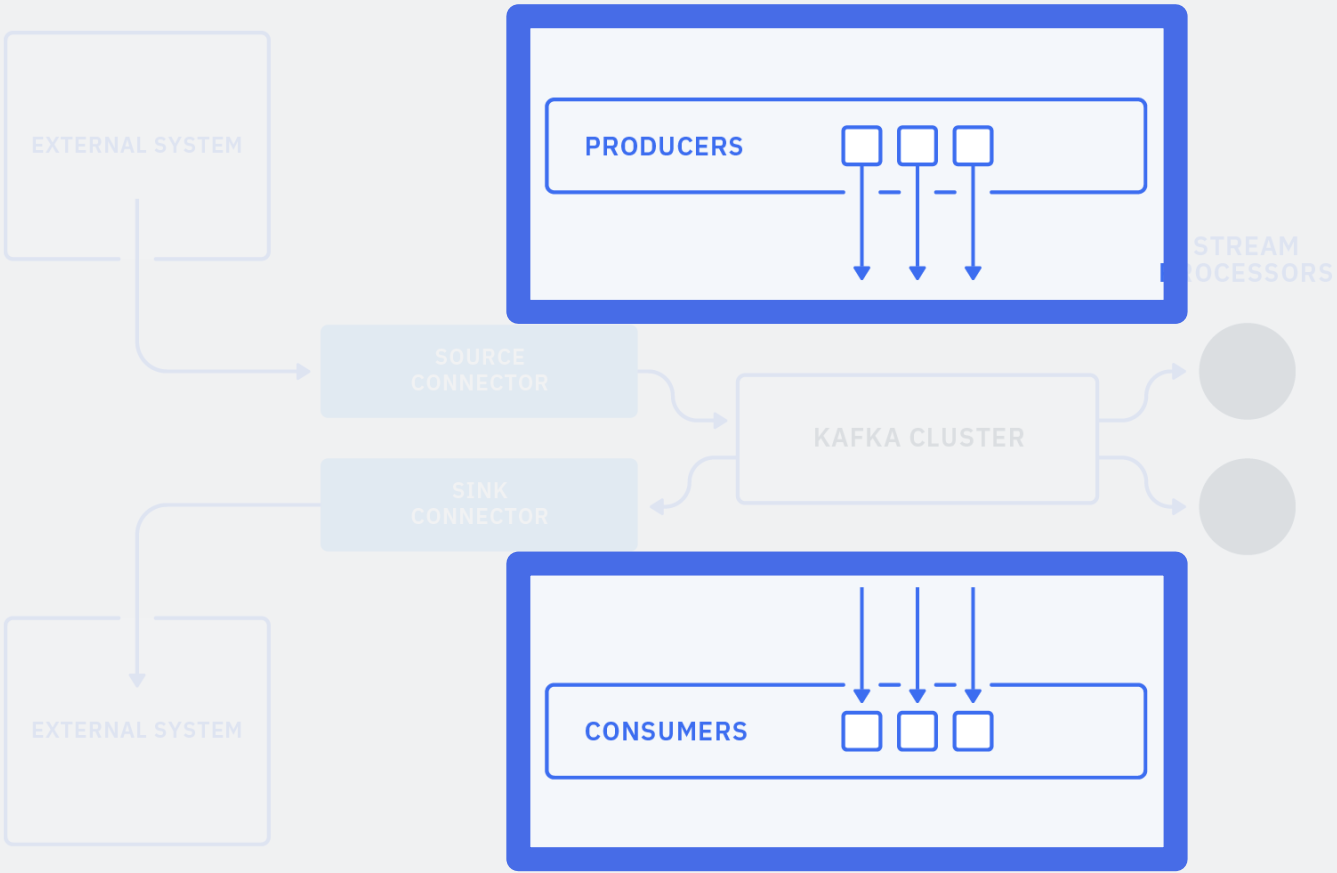
Retention period and compaction

Retention set in time or size

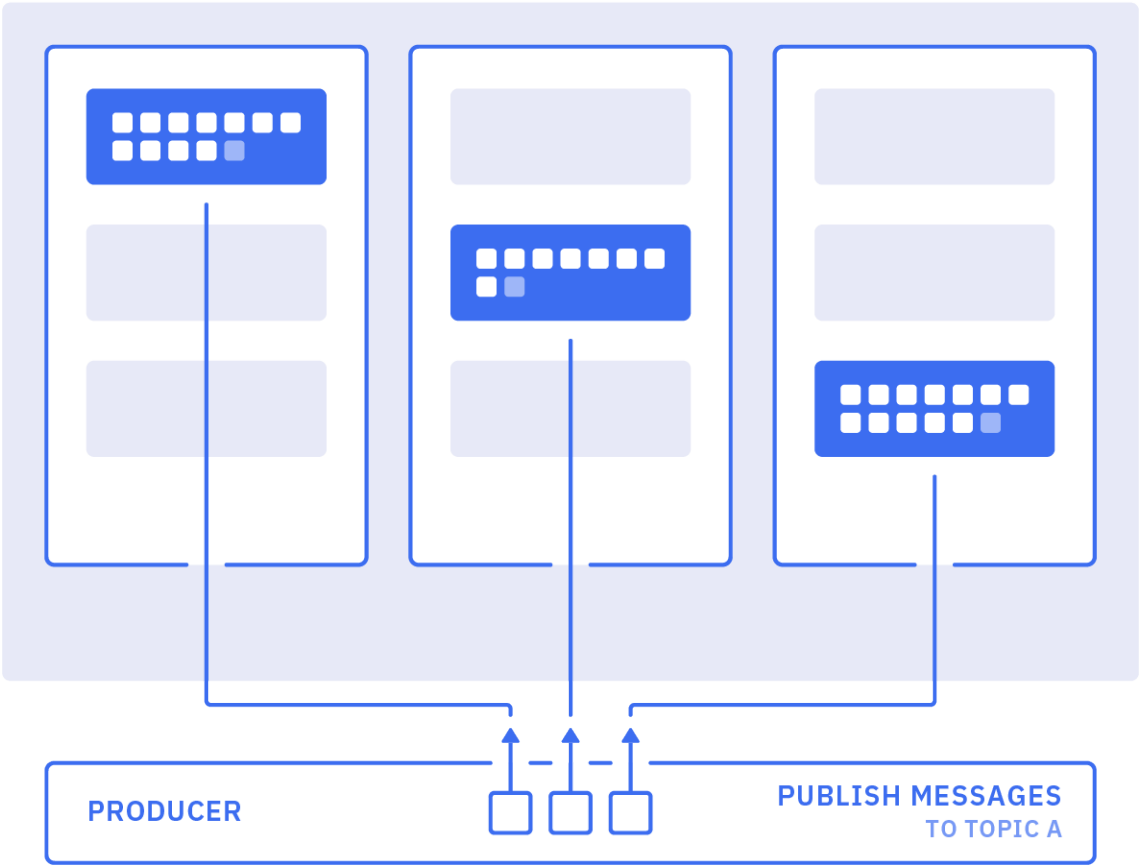
Compacted topics are evolving data stores

```
log.retention.minutes = 3000
log.retention.bytes   = 1048
```

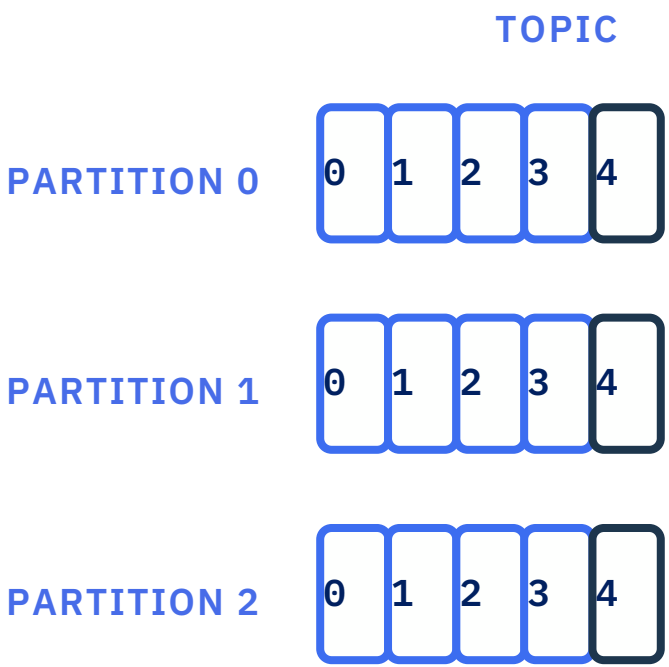



EXTERNAL SYSTEM
PRODUCERS
STREAM
OCESSORS
SOURCE
CONNECTOR
KAFKA CLUSTER
SINK
CONNECTOR
EXTERNAL SYSTEM
CONSUMERS

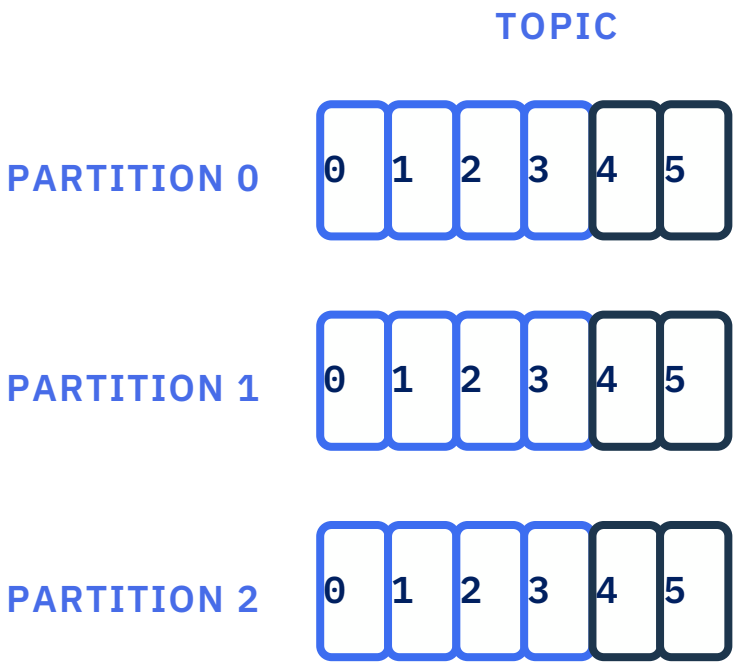
Producers



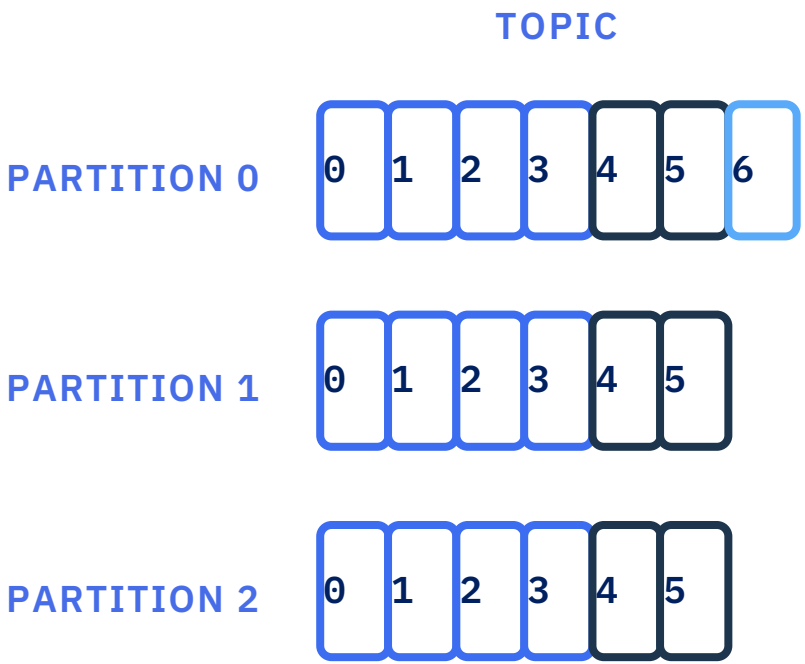
Producers and Keys



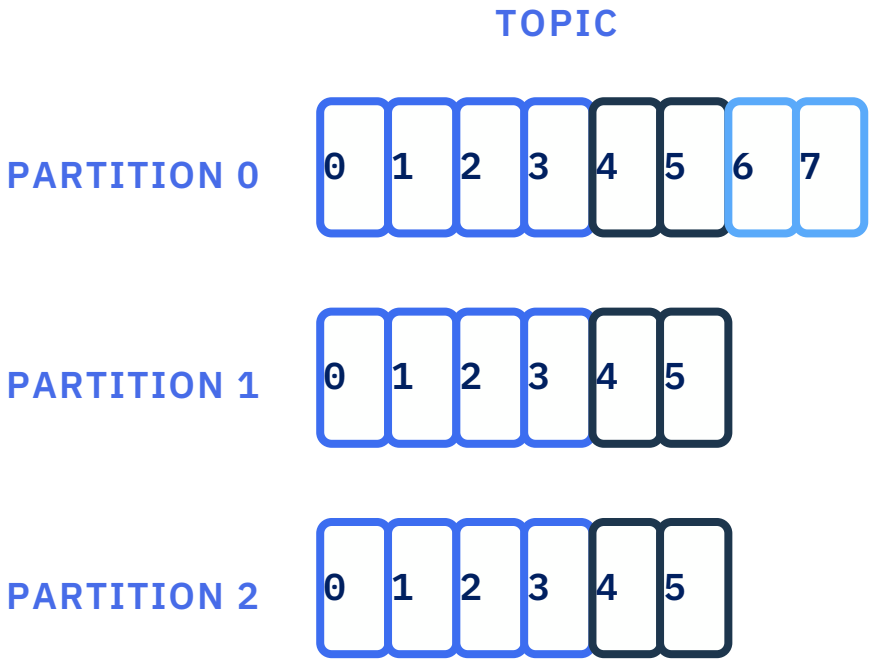
Producers and Keys



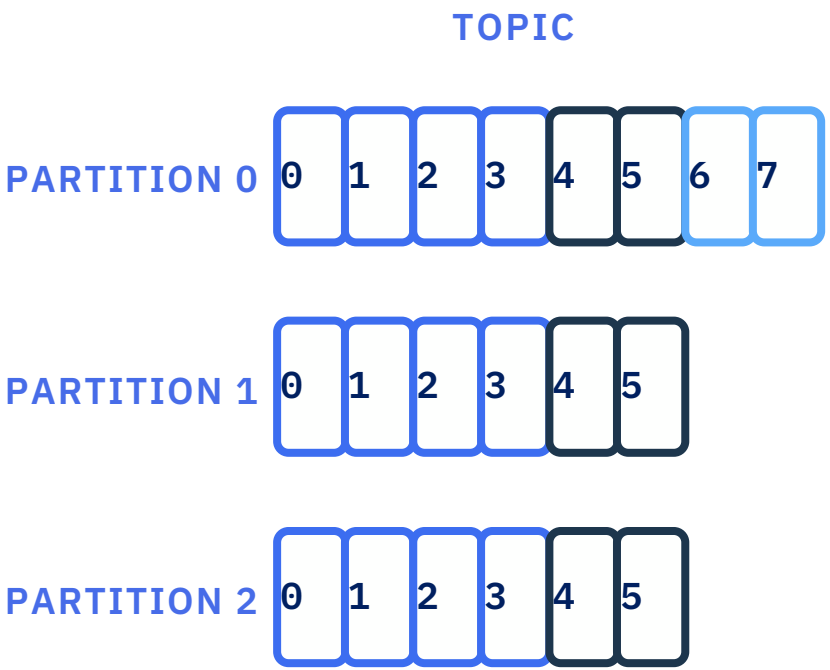
Producers and Keys



Producers and Keys



Producers



Producer can choose acknowledgement level:

- `0` Fire-and-forget
  Fast, but risky
- `1` Waits for 1 broker to acknowledge
- `ALL` Waits for all replica brokers to acknowledge

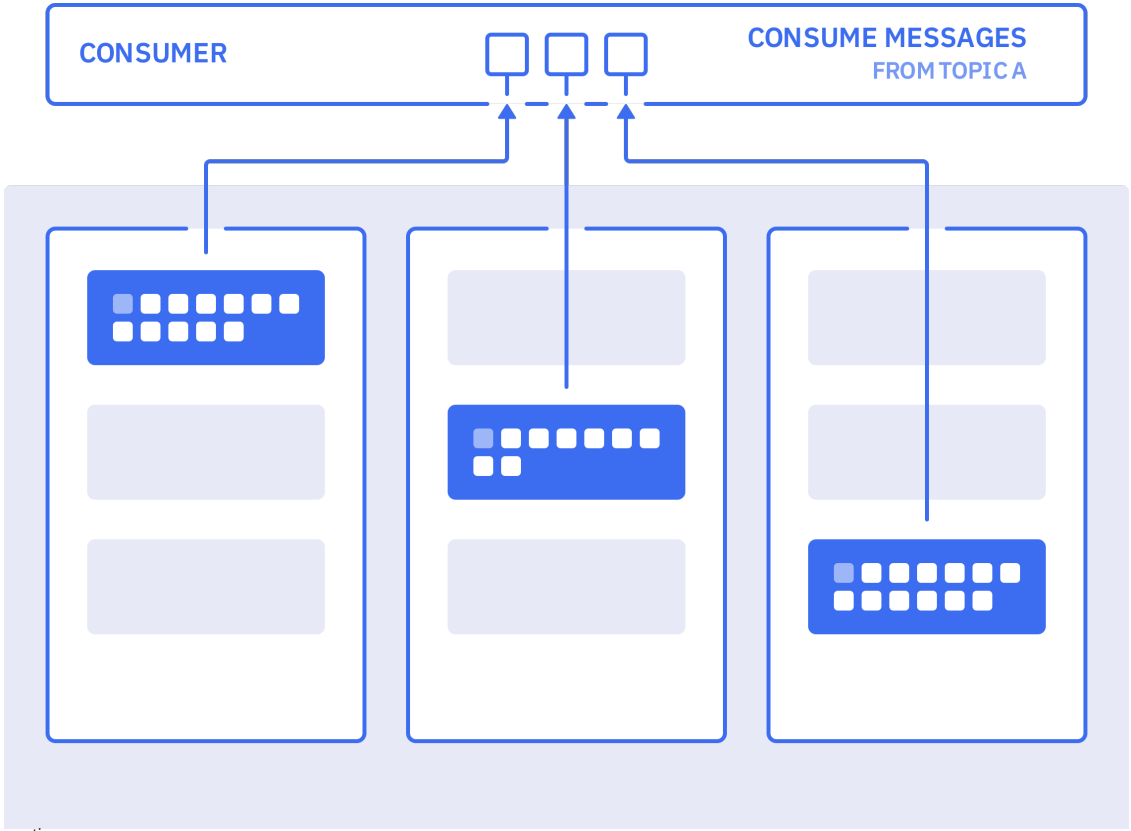
Producer can choose whether to retry:

- `0` Do not retry
  Loses messages on error
- `>0` Retry
  Retry, might result in duplicates on error

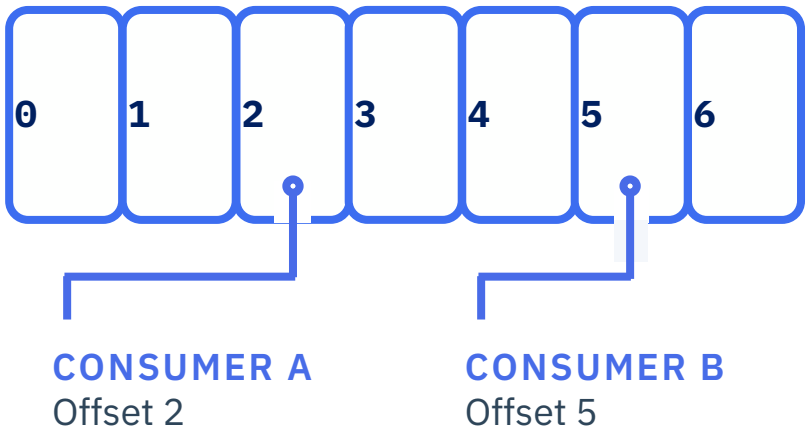
Producer can also choose idempotence

Can retry without risking duplicates

Consumers



Consumers



Consumer can choose how to commit offsets:

`Automatic`	Commits might go faster than processing
`Manual, asynchronous`	Fairly safe, but could re-process messages
`Manual, synchronous`	Safe, but slows down processing

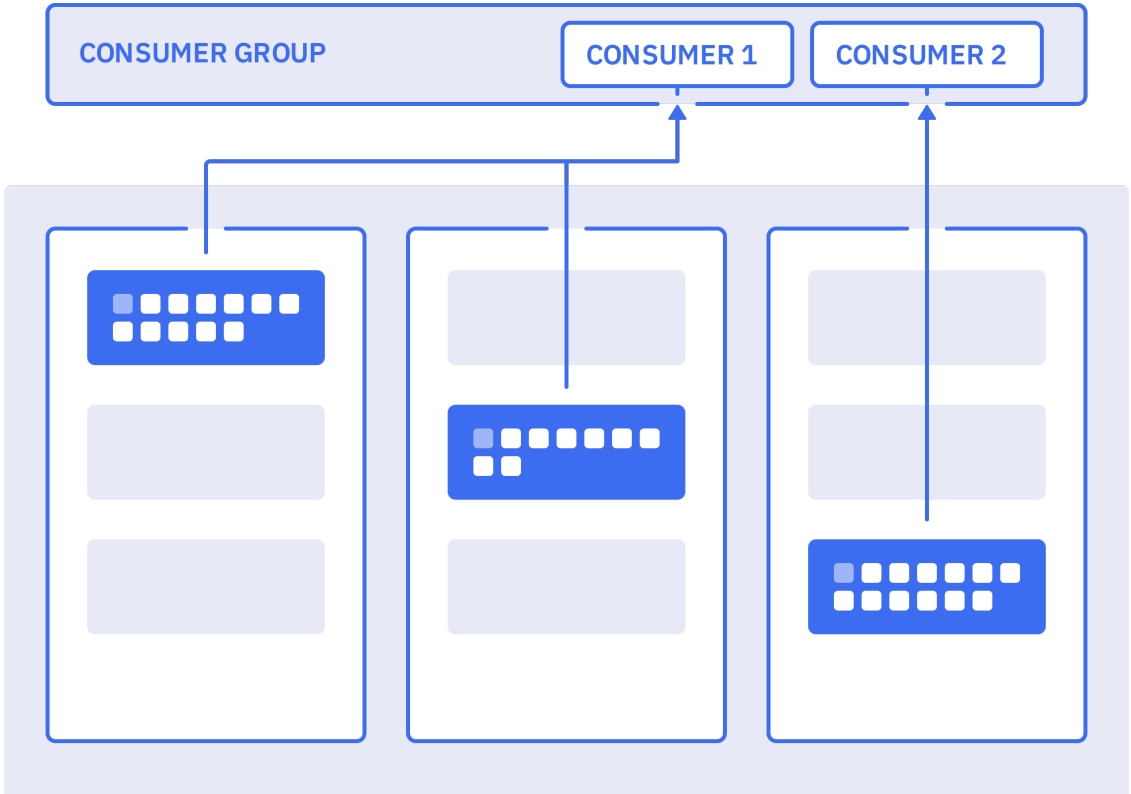
A common pattern is to commit offsets on a timer

Exactly once semantics

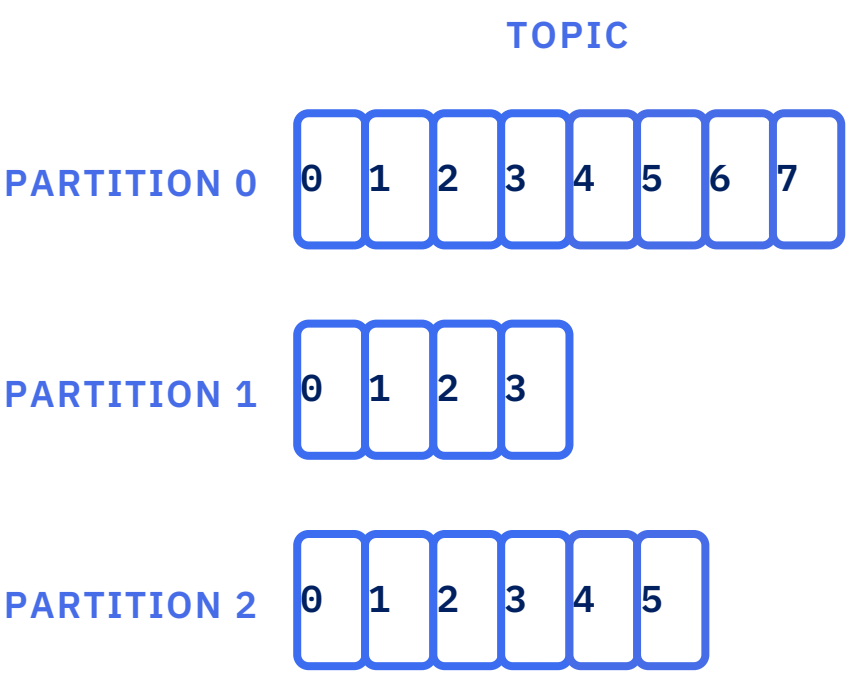
Can group sending messages and committing offsets into transactions

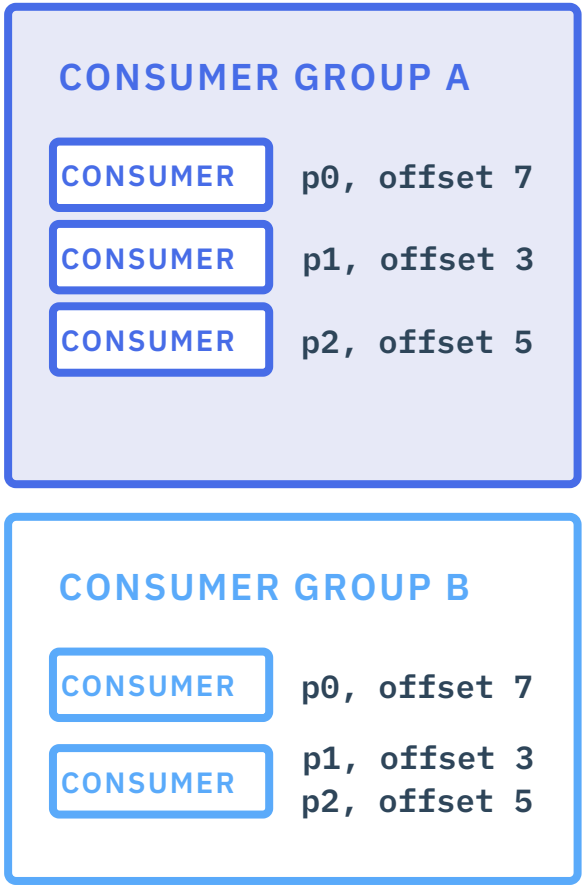
Primarily aimed at stream processing applications

Consumer Groups

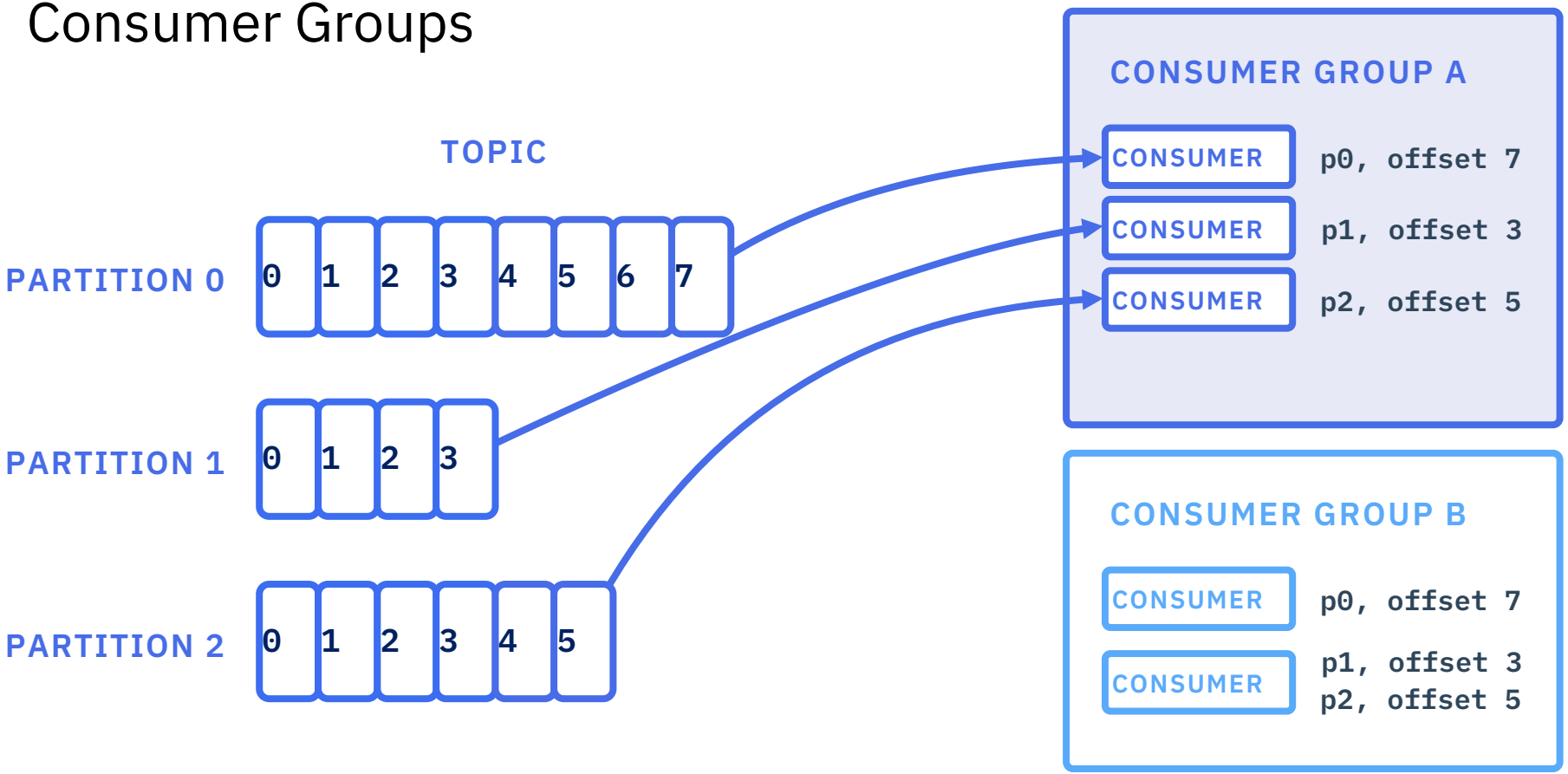


Consumer Groups

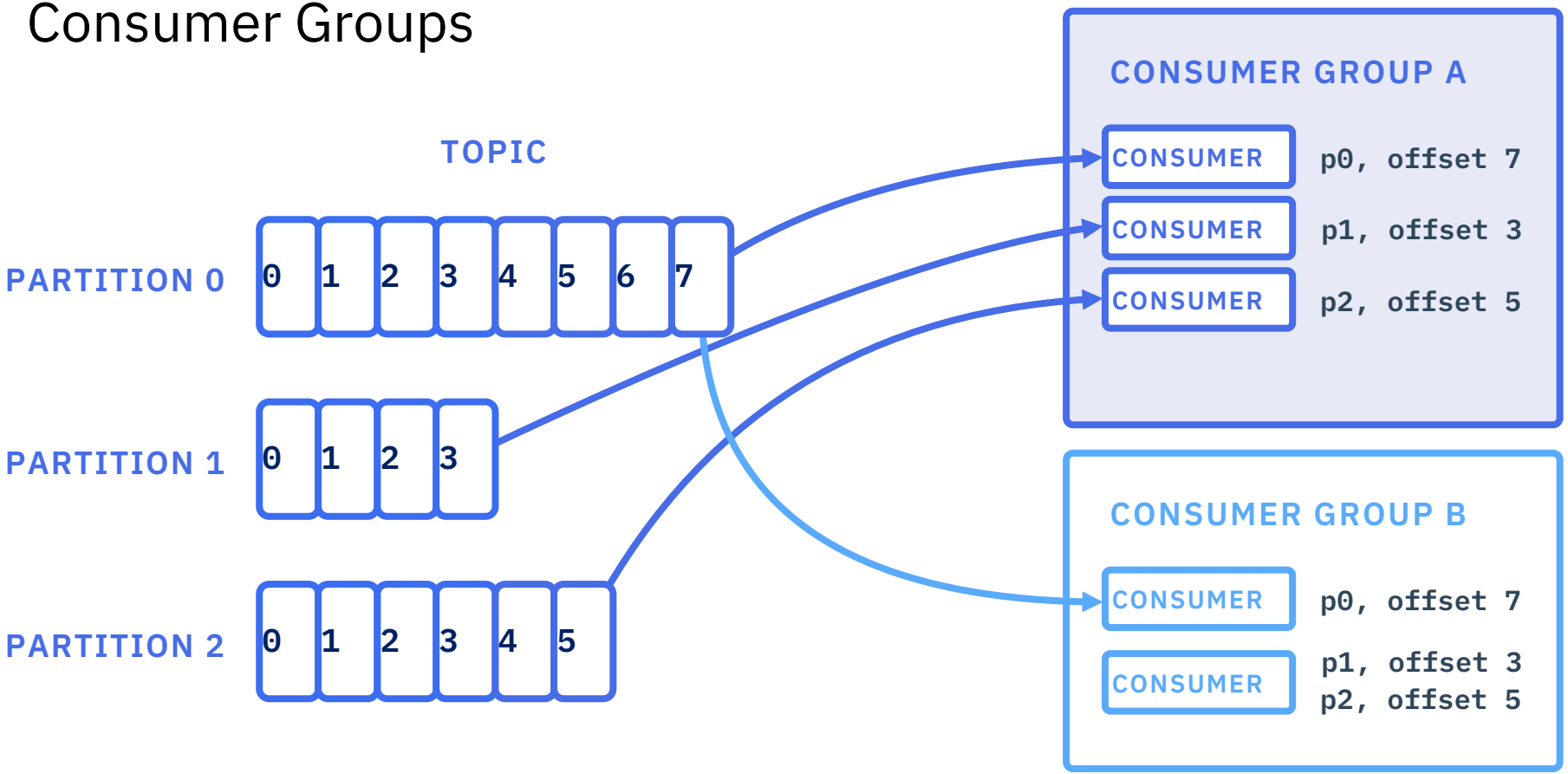




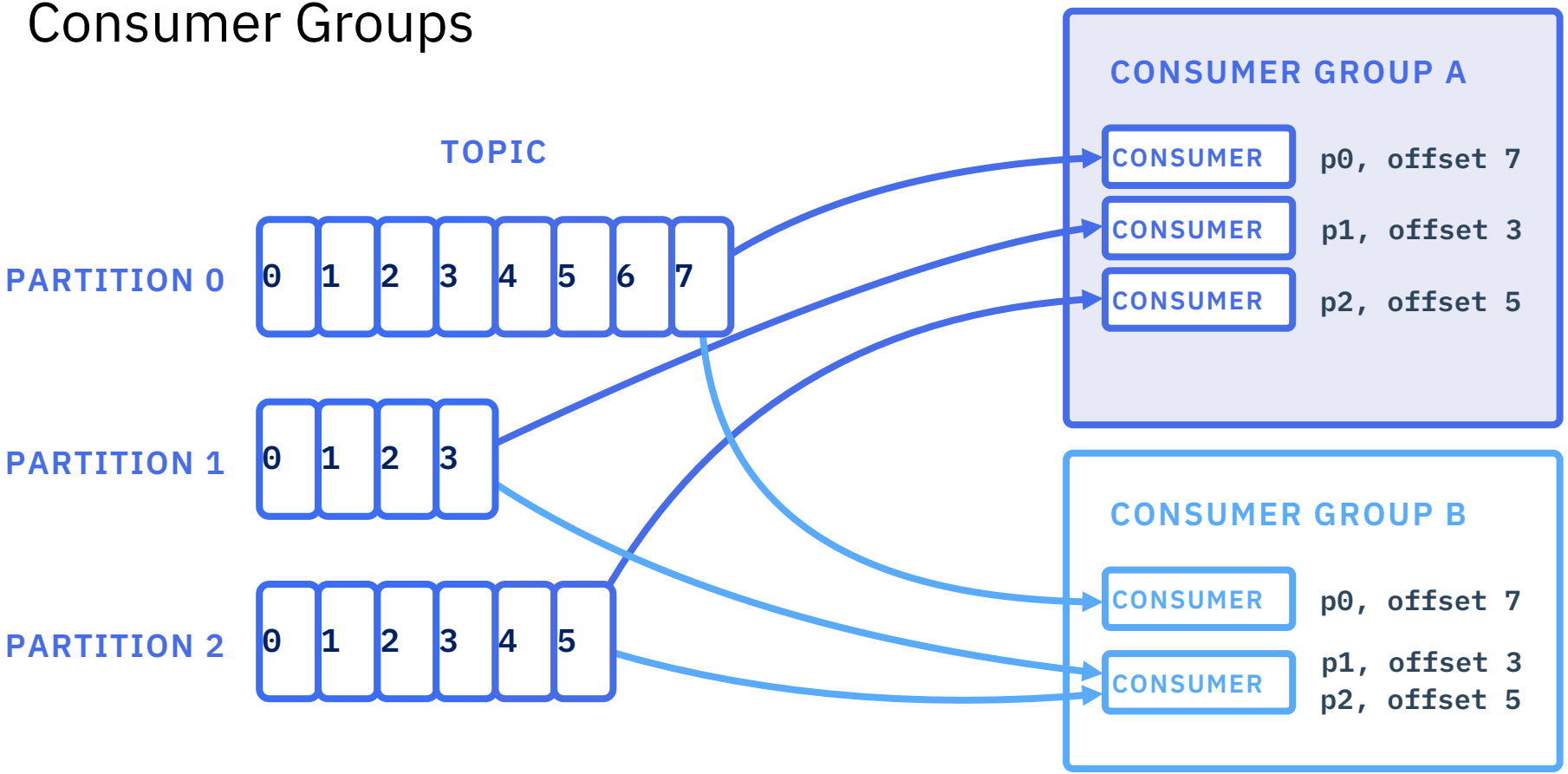
Consumer Groups



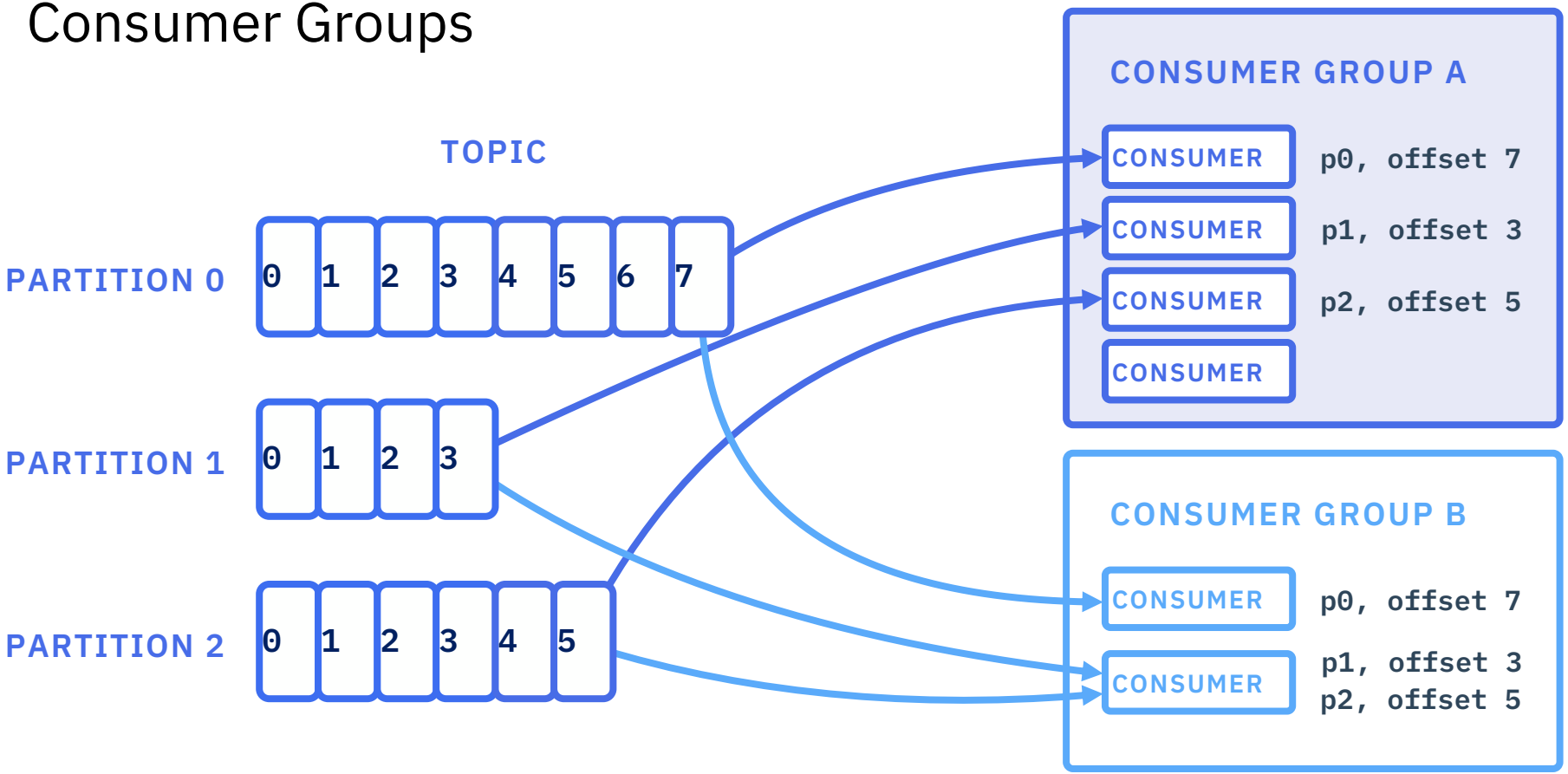
Consumer Groups

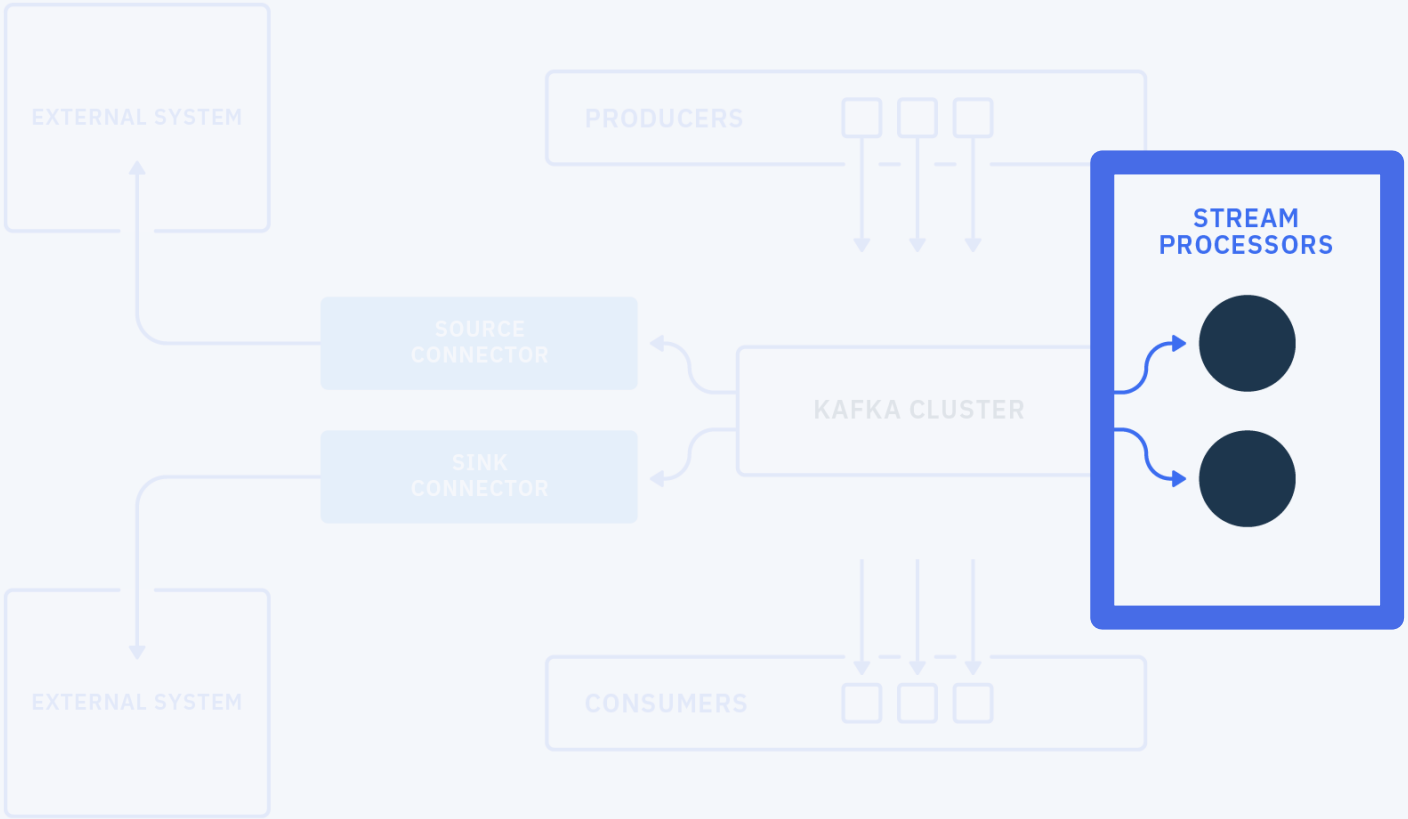


Consumer Groups



Consumer Groups



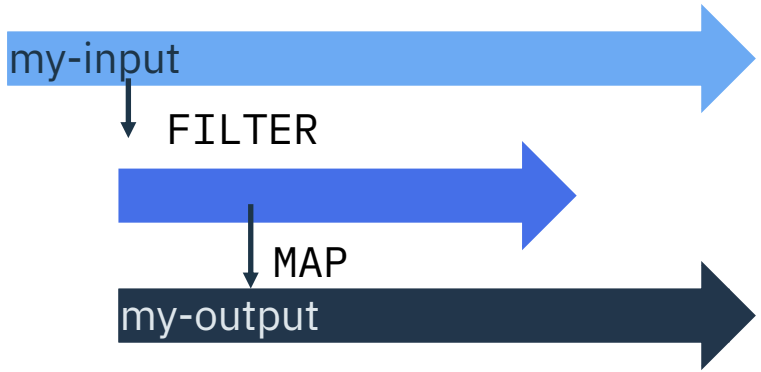

EXTERNAL SYSTEM
PRODUCERS
STREAM PROCESSORS
SOURCE CONNECTOR
KAFKA CLUSTER
SINK CONNECTOR
EXTERNAL SYSTEM
CONSUMERS

Kafka Streams

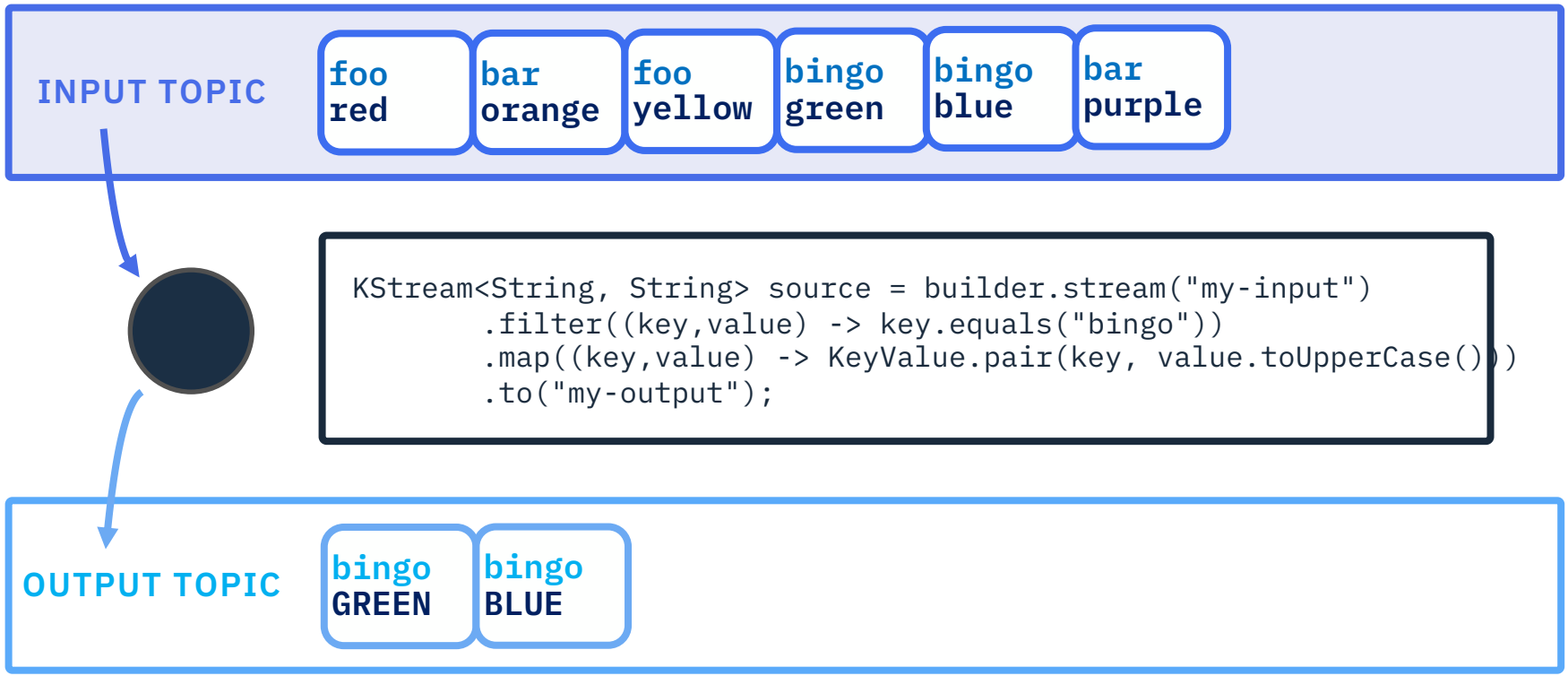
Client library for processing and analyzing data stored in Kafka

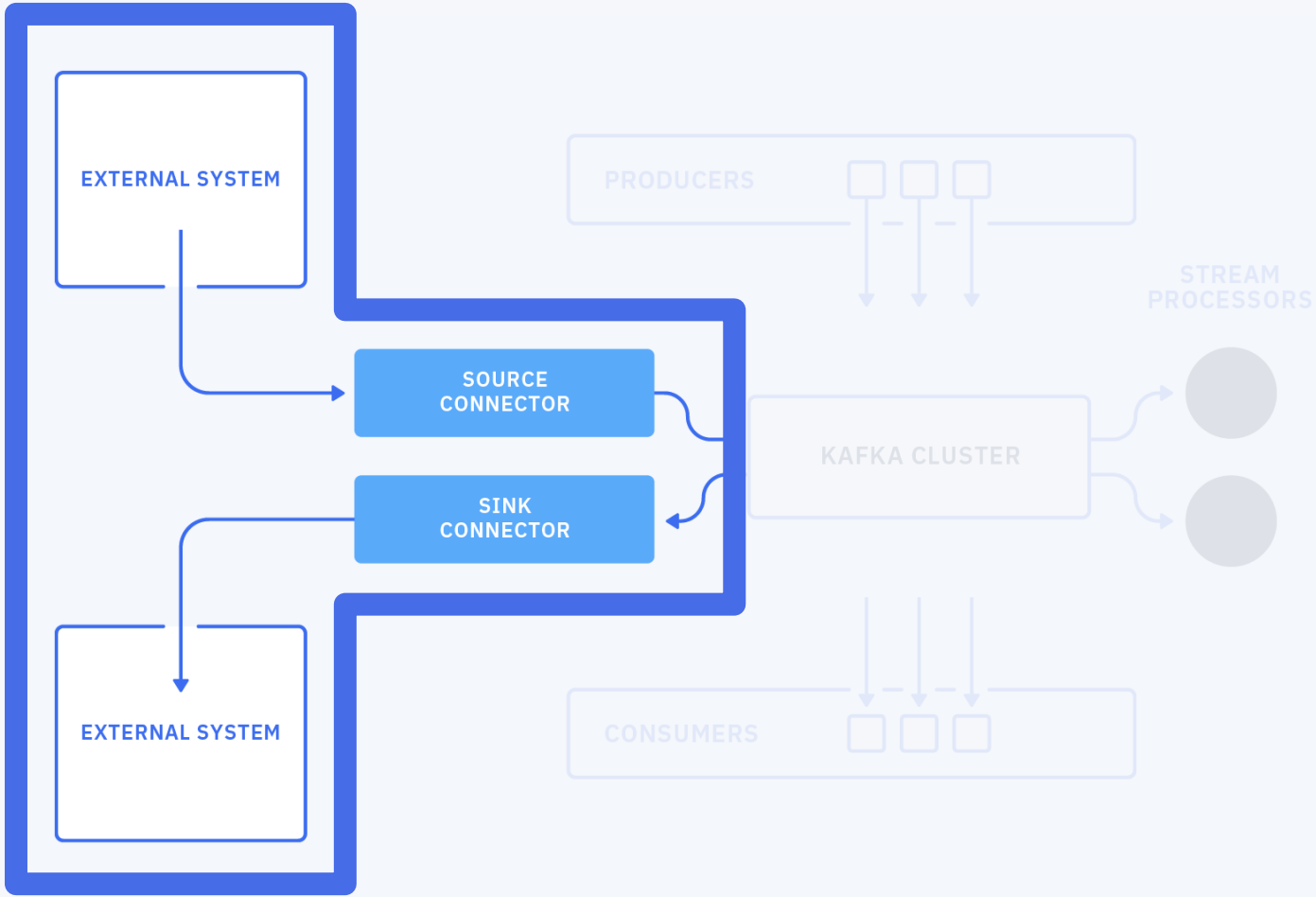
Processing happens in the app

Supports per-record processing – no batching

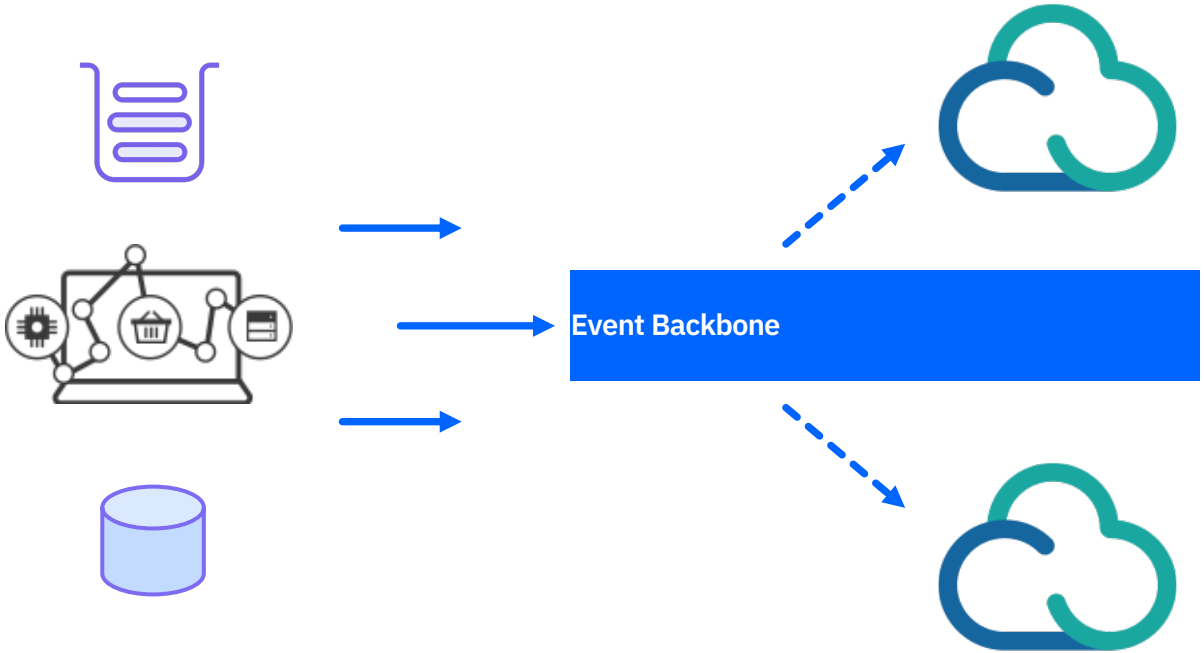


Kafka Streams

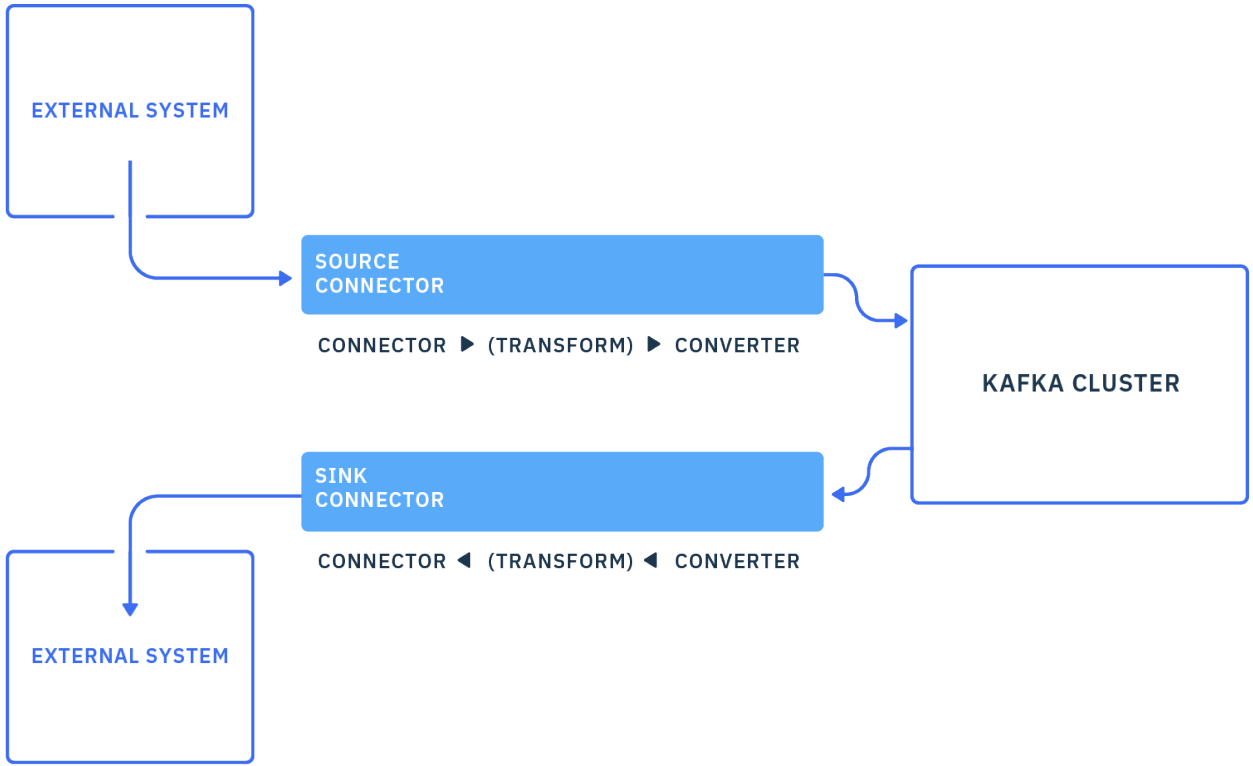



EXTERNAL SYSTEM
SOURCE CONNECTOR
SINK CONNECTOR
EXTERNAL SYSTEM
PRODUCERS
STREAM PROCESSORS
KAFKA CLUSTER
CONSUMERS

Kafka Connect – bridge to cloud-native apps



Kafka Connect



Over 80 connectors

HDFS

Elasticsearch

MySQL

JDBC

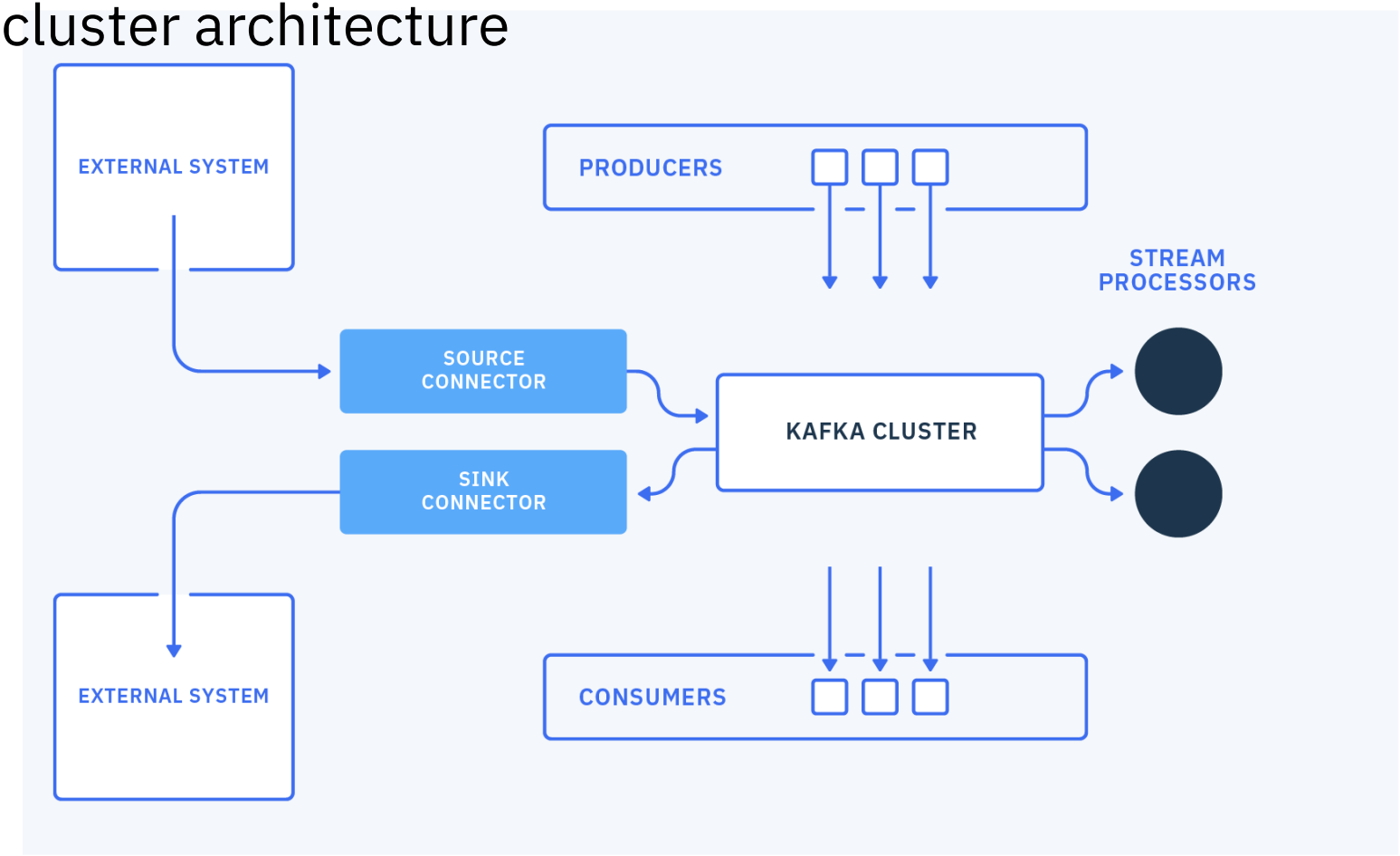
IBM MQ

MQTT

CoAP

+ many others

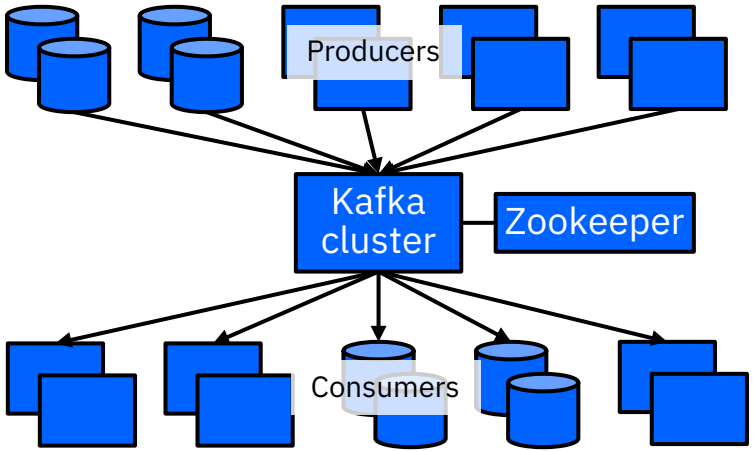
Kafka cluster architecture



Zookeeper

Kafka requires Apache Zookeeper

Manages cluster membership, electing leaders, topic configuration, and more



Kafka guarantees

Messages are appended to partitions in the order they are sent

Consumers see messages in the order they are stored in the log

Committed messages are not lost as long as at least one ISR exists at all times

For a topic with a replication factor of *N*, Kafka tolerates up to *N*-1 server failures without losing any messages that are committed to the log

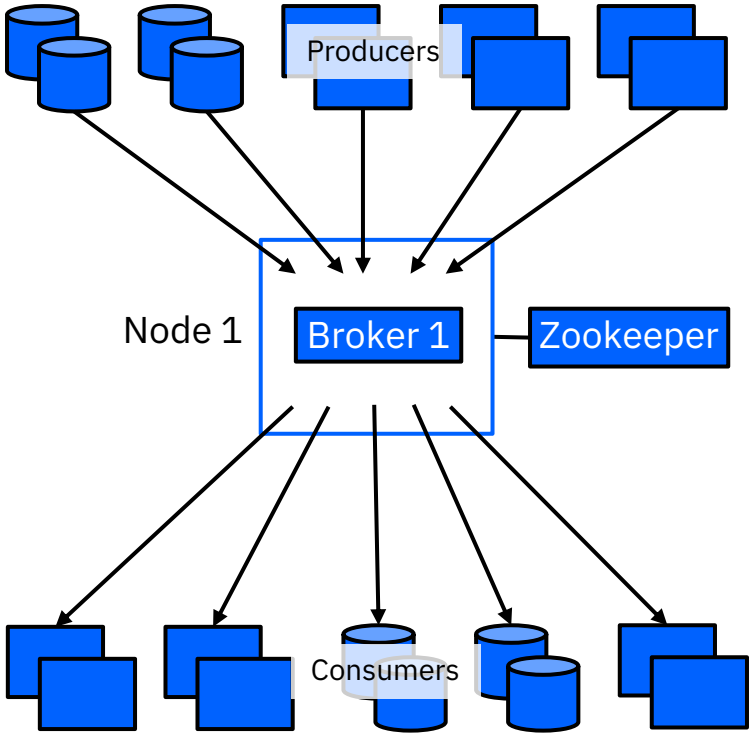
At-least-once message delivery guaranteed by default

You can implement at-most-once by disabling producer retries and committing offset prior to processing a batch of messages

Exactly-once requires cooperation with the destination storage system

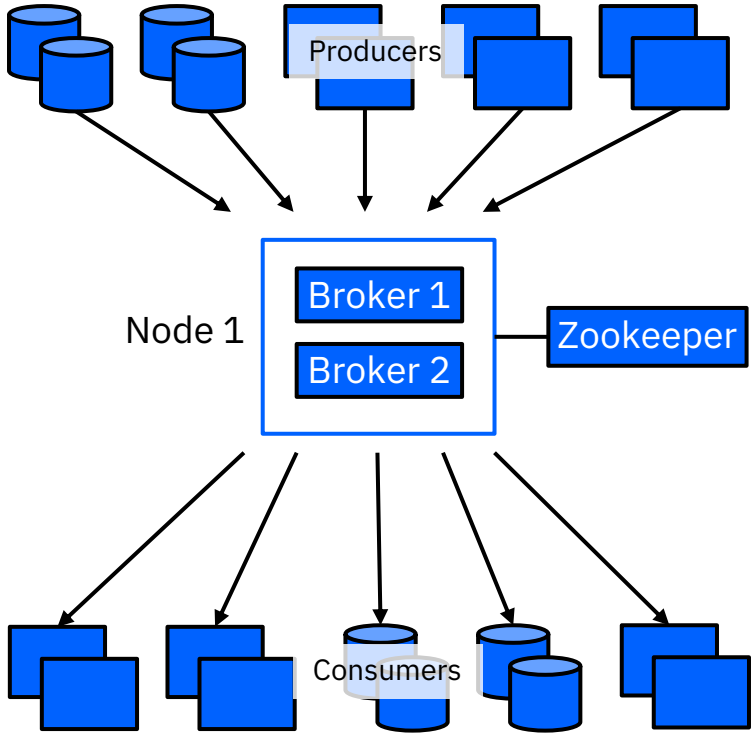
Kafka cluster configurations

Single node, single broker



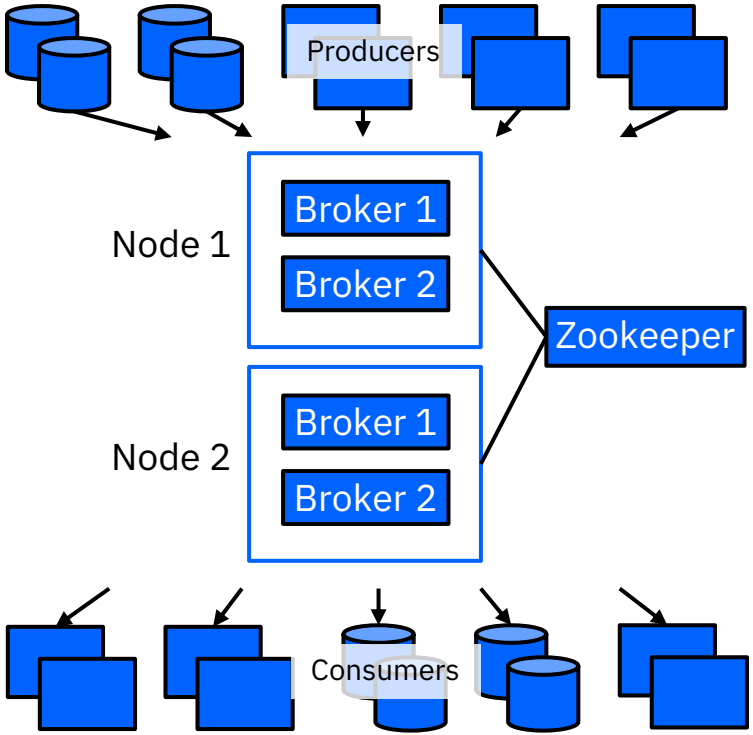
Kafka cluster configurations

Single node, multiple brokers



Kafka cluster configurations

Multiple nodes,
multiple brokers



Kafka commands

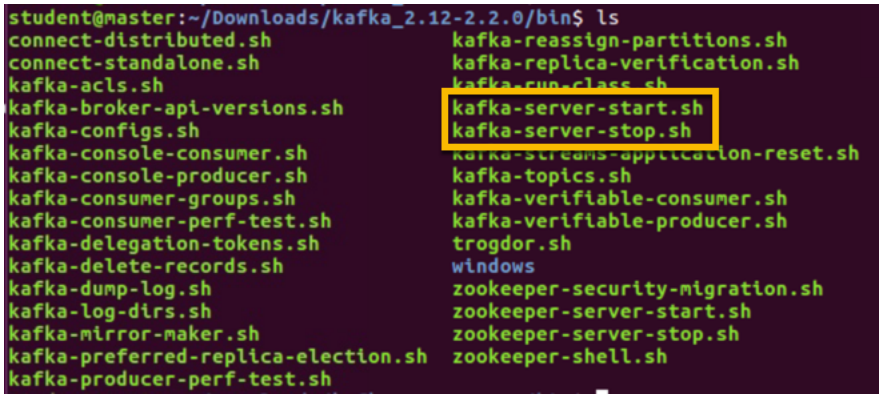
Commands in the Kafka `/bin` directory

Start and stop a Kafka broker server

Pass in a `server.properties` file

Kafka `/config` directory contains a default `server.properties` file

You can also use Kafka commands to create and manage topics



Kafka server.properties file

Contains several configuration properties for the broker, such as the broker ID, port, and socket that the broker listens on

You can copy and modify this file for each broker

```
# see kafka.server.KafkaConfig for additional details and defaults

############################## Server Basics ##############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0

############################## Socket Server Settings ##############################

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092

# Hostname and port the broker will advertise to producers and consumers. If not set,
# it uses the value for "listeners" if configured.  Otherwise, it will use the value
# returned from java.net.InetAddress.getCanonicalHostName().
#advertised.listeners=PLAINTEXT://your.host.name:9092

# Maps listener names to security protocols, the default is for them to be the same. See the config
documentation for more details
#listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SA

# The number of threads that the server uses for receiving requests from the network and sending
responses to the network
num.network.threads=3

# The number of threads that the server uses for processing requests, which may include disk I/O
num.io.threads=8

# The send buffer (SO_SNDBUF) used by the socket server
socket.send.buffer.bytes=102400

# The receive buffer (SO_RCVBUF) used by the socket server
socket.receive.buffer.bytes=102400

# The maximum size of a request that the socket server will accept (protection against OOM)
socket.request.max.bytes=104857600
```

Kafka console producer

Can be used for development and testing

Takes input from a file or the console and sends messages to the Kafka cluster

Each line is sent as a separate message

After entering the command, in the console, type a message and press Enter

Example:

```
# bin/kafka-console-producer.sh
--broker-list localhost:9092 --topic
mytopic
```

Kafka console consumer

Can be used for development and testing

Prints messages from topics to standard output

Each message is printed on a new line

Use `--from-beginning` to consume messages published before consumer start

Example:

```
# bin/kafka-console-consumer.sh
--zookeeper localhost:2181 --topic
mytopic –from-beginning
```

This is message one.

This is message two.

More resources

Apache Kafka documentation

https://kafka.apache.org/documentation/

Event Driven Reference Architecture

https://ibm-cloud-architecture.github.io/refarch-eda/

Practice quiz (1 of 2)

1. How are event-driven solutions different from other kinds of solutions?

   a. They look at IT as an information repository.

   b. They are more concerned with data at rest, rather than data in flight.

   c. They are more concerned with the data store, rather than the log of events.

   d. They are more concerned with responding to events, rather than preserving data.

2. What basic Kafka concept can be described as an "immutable sequence of records"?

   a. Topic

   b. Broker

   c. Producer

   d. Consumer

Practice quiz (2 of 2)

3. True/False: A producer decides which partition in a topic that a message is written to.

4. True/False: Consumers can start reading from any offset on a topic.

5. How is Kafka able to provide horizontal scaling and fault tolerance?

a. by replicating producers

b. by replicating consumers

c. by replicating partitions

d. by having an odd number of brokers

June 2019 Edition

Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation
North Castle Drive, MD-NC119 Armonk, NY 10504-1785
United States of America

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

IBM Cloud™

z/OS®

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware is a registered trademark or trademark of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.