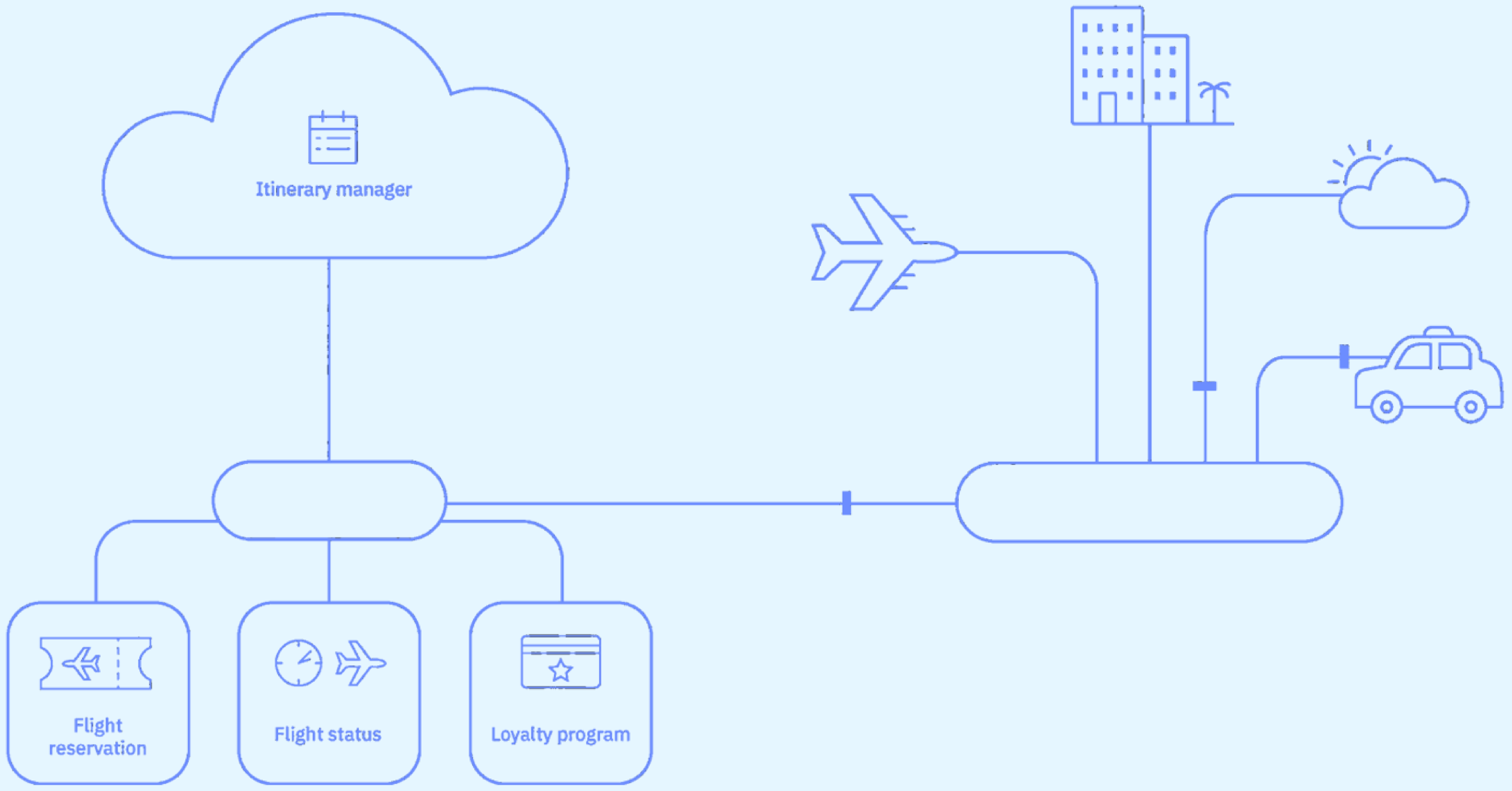
Unlock events in mission-critical systems

Connect IBM MQ and Kafka to create insights from core system data

Dorothy Quincy
Client Technical Specialist

Dorothy.quincy@ibm.com

Messaging is essential for building fully connected, efficient and scalable solutions. More now than ever before



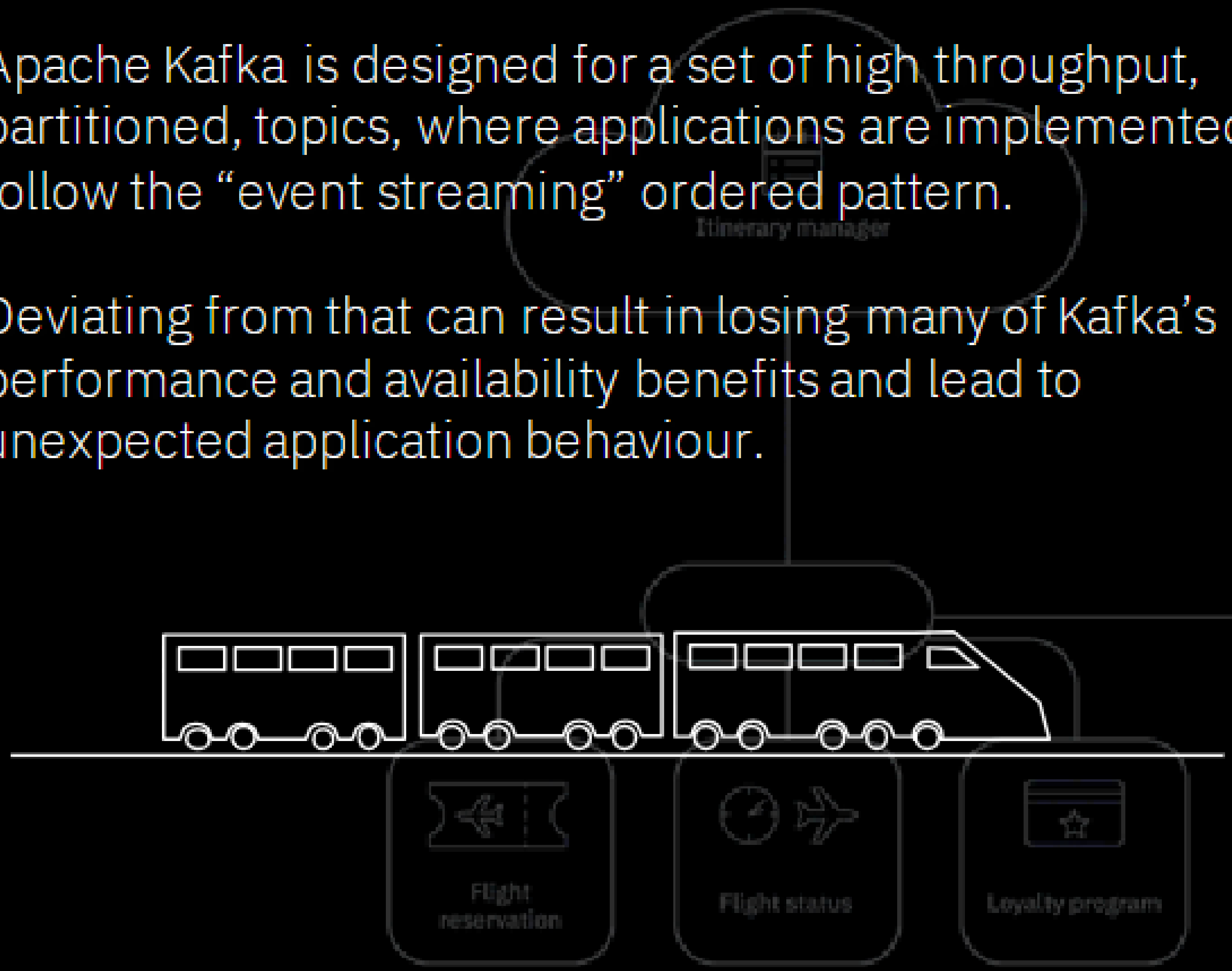
Messaging technologies

Different approaches, satisfy different requirements

Apache Kafka

Apache Kafka is designed for a set of high throughput, partitioned, topics, where applications are implemented to follow the “event streaming” ordered pattern.

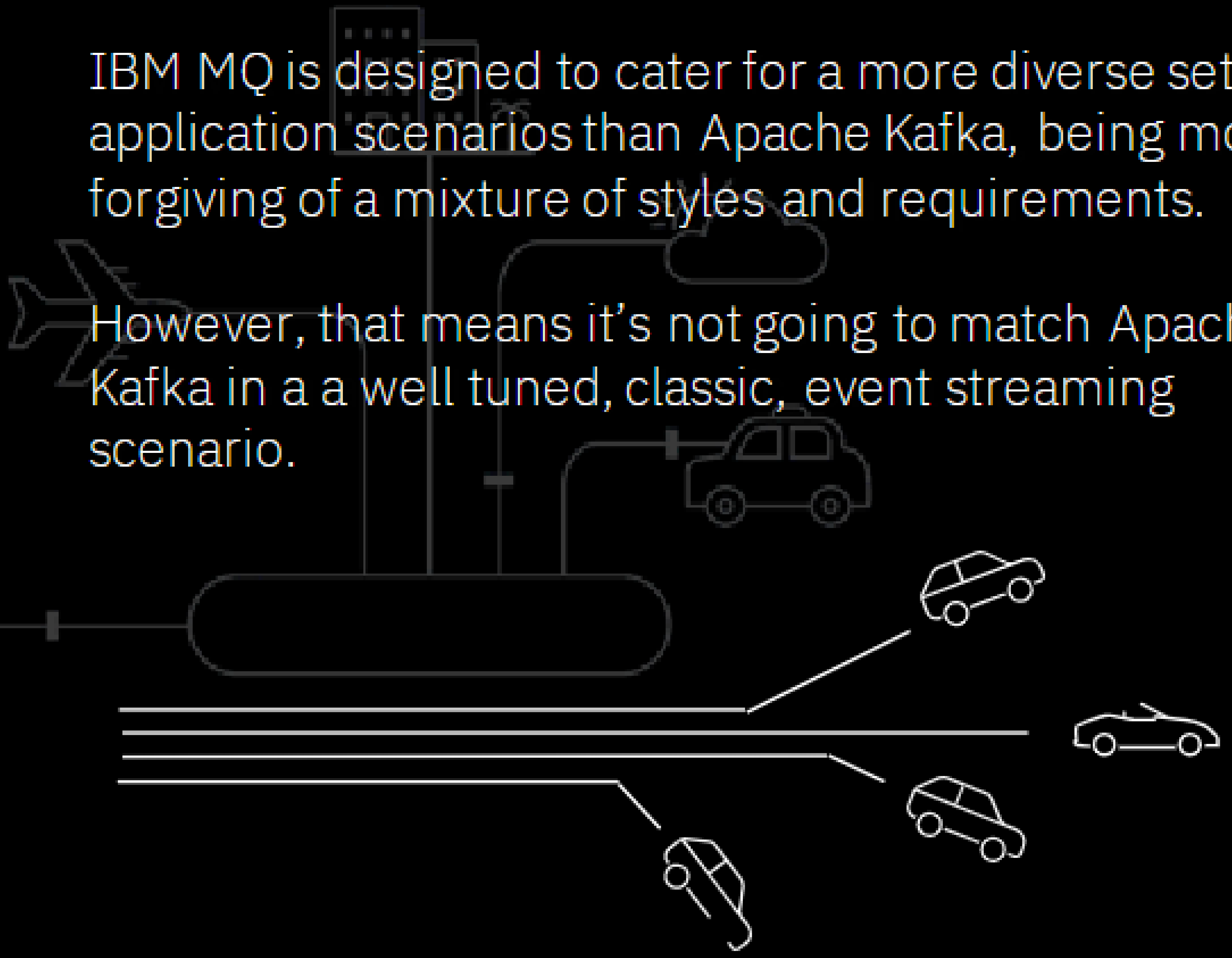
Deviating from that can result in losing many of Kafka’s performance and availability benefits and lead to unexpected application behaviour.



IBM MQ

IBM MQ is designed to cater for a more diverse set of application scenarios than Apache Kafka, being more forgiving of a mixture of styles and requirements.

However, that means it’s not going to match Apache Kafka in a a well tuned, classic, event streaming scenario.



Understand the application’s style and behaviour and pick the technology based on that

Messaging patterns

Commands

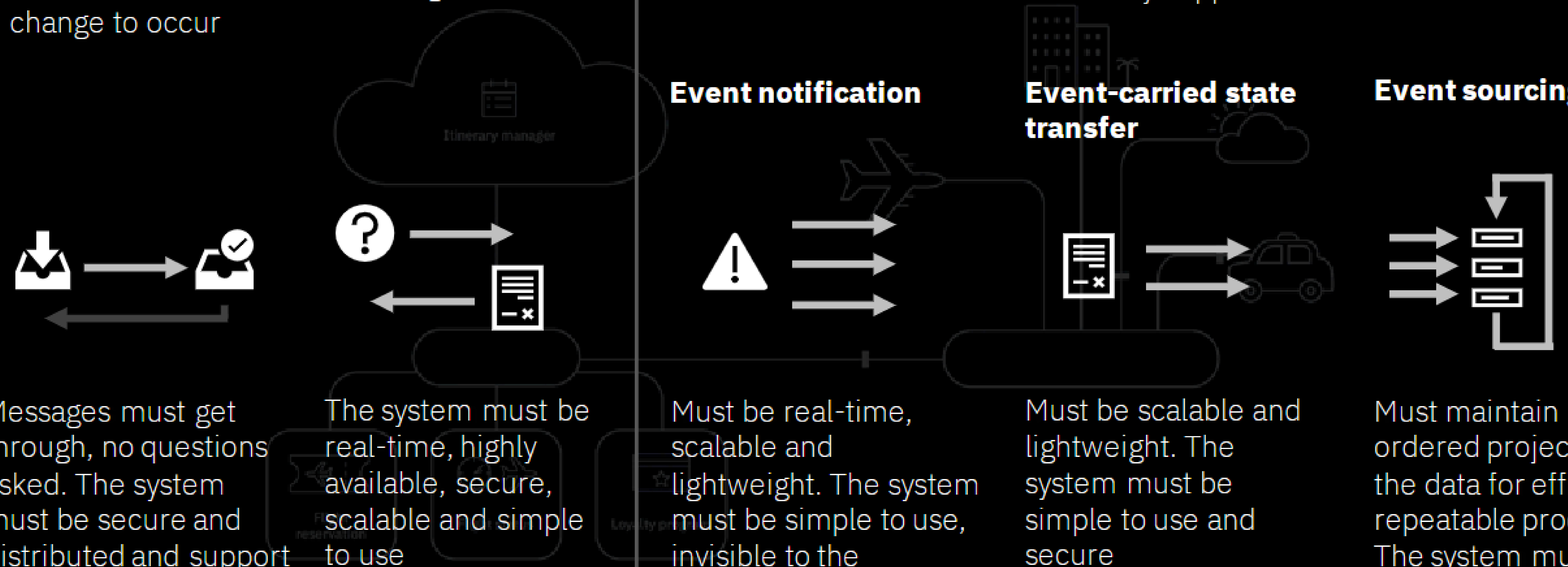
A command instructs a state change to occur

Queries

A query returns existing state

Events

An event reports the state of something that has already happened



Commands: Messages must get through, no questions asked. The system must be secure and distributed and support queued workload

Queries: The system must be real-time, highly available, secure, scalable and simple to use

Event notification

Must be real-time, scalable and lightweight. The system must be simple to use, invisible to the application and highly available

Event-carried state transfer

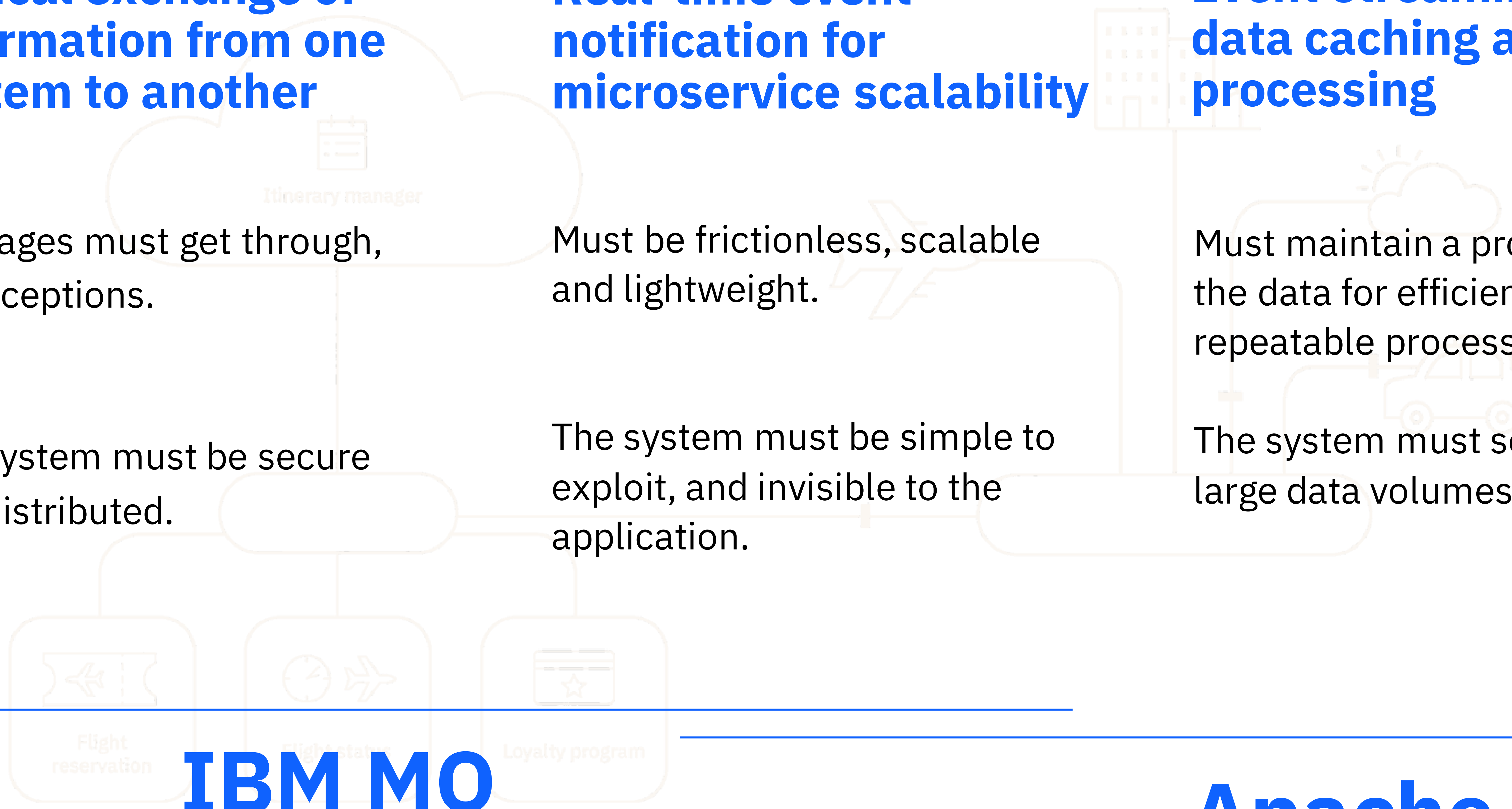
Must be scalable and lightweight. The system must be simple to use and secure

Event sourcing

Must maintain an ordered projection of the data for efficient repeatable processing. The system must be secure and scale for high data retention

Messaging patterns

Messaging is essential for building fully connected, efficient and scalable solutions. More now than ever before



Critical exchange of information from one system to another

Messages must get through, no exceptions.

The system must be secure and distributed.

Real-time event notification for microservice scalability

Must be frictionless, scalable and lightweight.

The system must be simple to exploit, and invisible to the application.

Event streaming for data caching and processing

Must maintain a projection of the data for efficient repeatable processing.

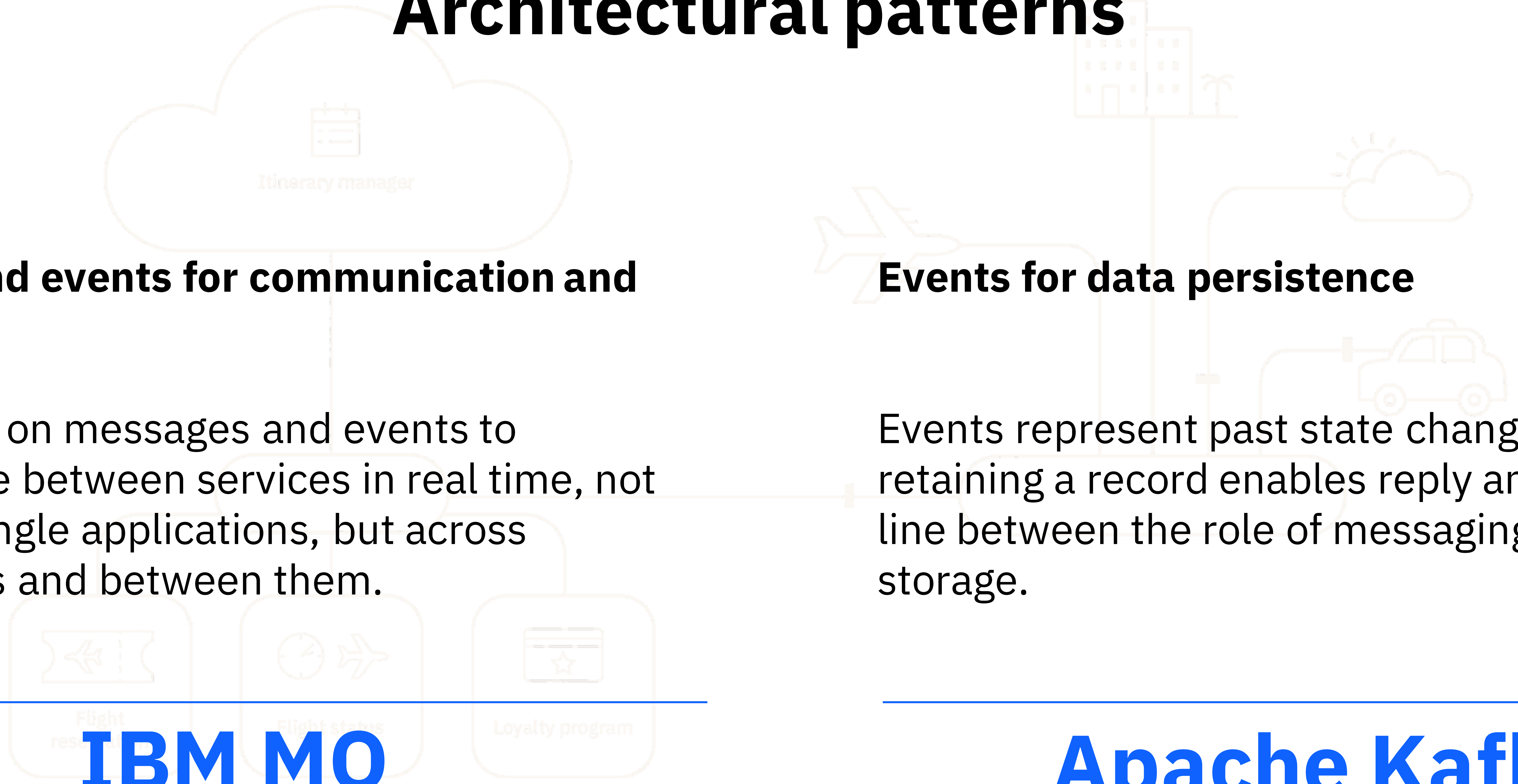
The system must scale for large data volumes.

IBM MQ

Apache Kafka

Messaging patterns

Messaging is essential for building fully connected, efficient and scalable solutions. More now than ever before



Architectual patterns

Messages and events for communication and notification

Systems rely on messages and events to communicate between services in real time, not just within single applications, but across organisations and between them.

IBM MQ

Events for data persistence

Events represent past state changes, retaining a record enables reply and blurs the line between the role of messaging and data storage.

Apache Kafka

Event-driven fuels the real-time enterprise

Provides continuous awareness

Build a better picture of your current business

Drives automation

Enable automation in response to events to any given situation

Enables adaptability

Adapt the way your business operates based on situations you detect

Unlock events in mission-critical data

Data from the core systems that IBM MQ connects must flow uninterrupted around your organization to help ensure **seamless customer experiences** and **support strategic business processes**.

Connect

IBM MQ & Apache Kafka

Fully IBM-supported MQ sink and source connectors are now included to make it easier than ever to connect core business systems to Apache Kafka.

Unlock

Events

Use streaming queues or topics to expose events in disparate systems without interrupting the flow of data between mission-critical apps.

Discover

Insights

Streams of valuable data from across the business enable you to drive new insights and act on events as they happen.

Available in v9.3.3 - IBM MQ Advanced & MQ Advanced for zOS Value Unit Edition

Connecting IBM MQ with Apache Kafka

Key Vocab

Brokers

A Kafka **cluster** consists of a set of brokers. A cluster has a minimum of three brokers.

Topics and partitions

Each topic is a named stream of messages. A **topic** is made up of one or more partitions. The messages on a partition are ordered by a number that is called the **offset**.

Why would a topic have more than one partition?

- It allows data to be fed through in parallel to increase throughput by distributing the partitions across the cluster.
- The number of partitions also influences the balancing of workload among consumers.

Producer/consumer

A producer publishes messages to one or more topics. A producer can publish to one or more topics and can optionally choose the partition that stores the data.

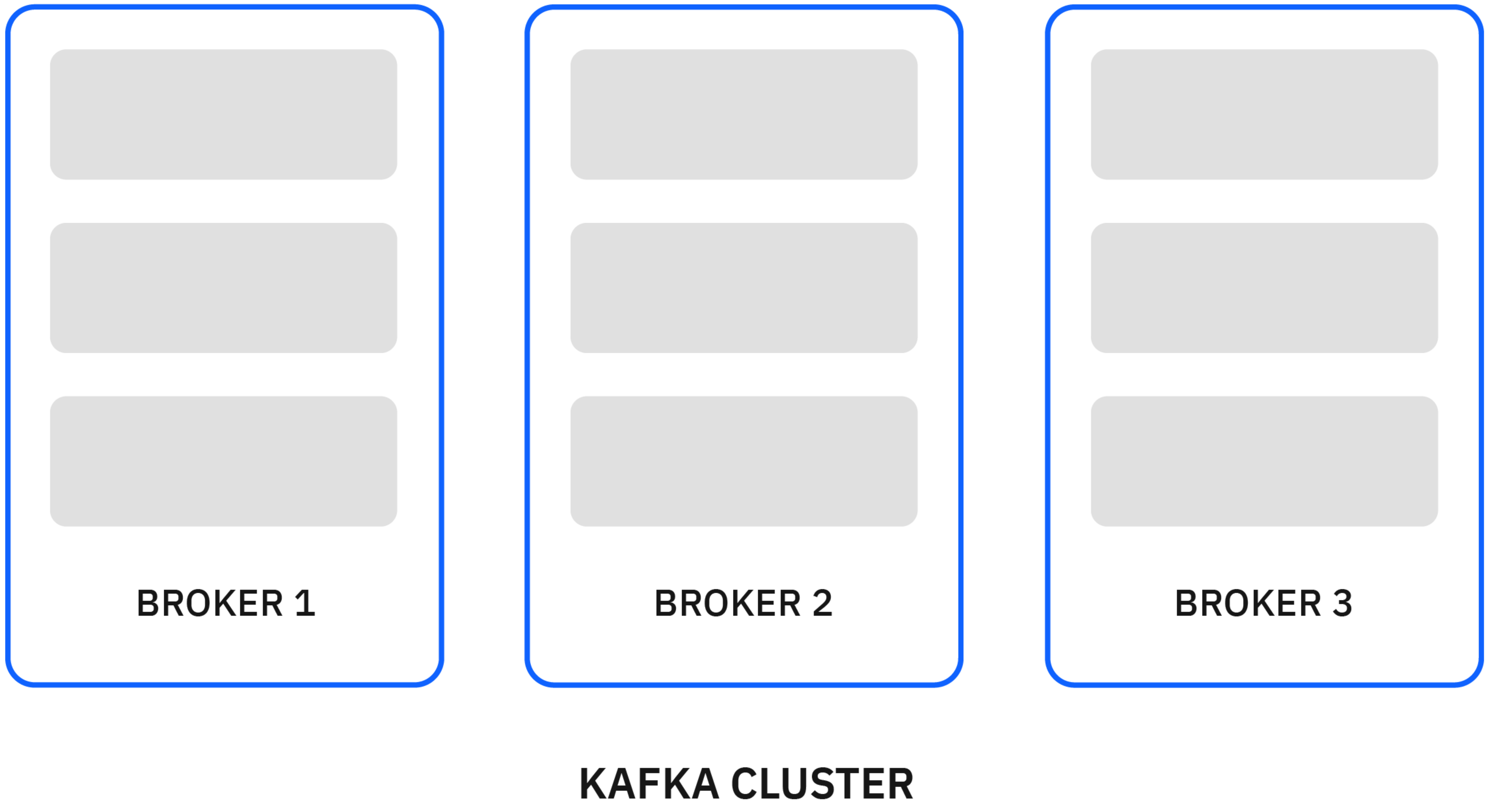
A consumer reads messages from one or more topics and processes them. The difference between a consumer's current position and the newest message on a partition is known as the offset lag.

Replication

To improve availability, each topic can be replicated onto multiple brokers. For each partition, one of the brokers is the leader, and the other brokers are the followers.

Brokers

A Kafka **cluster** consists of a set of brokers. A cluster has a minimum of three brokers.

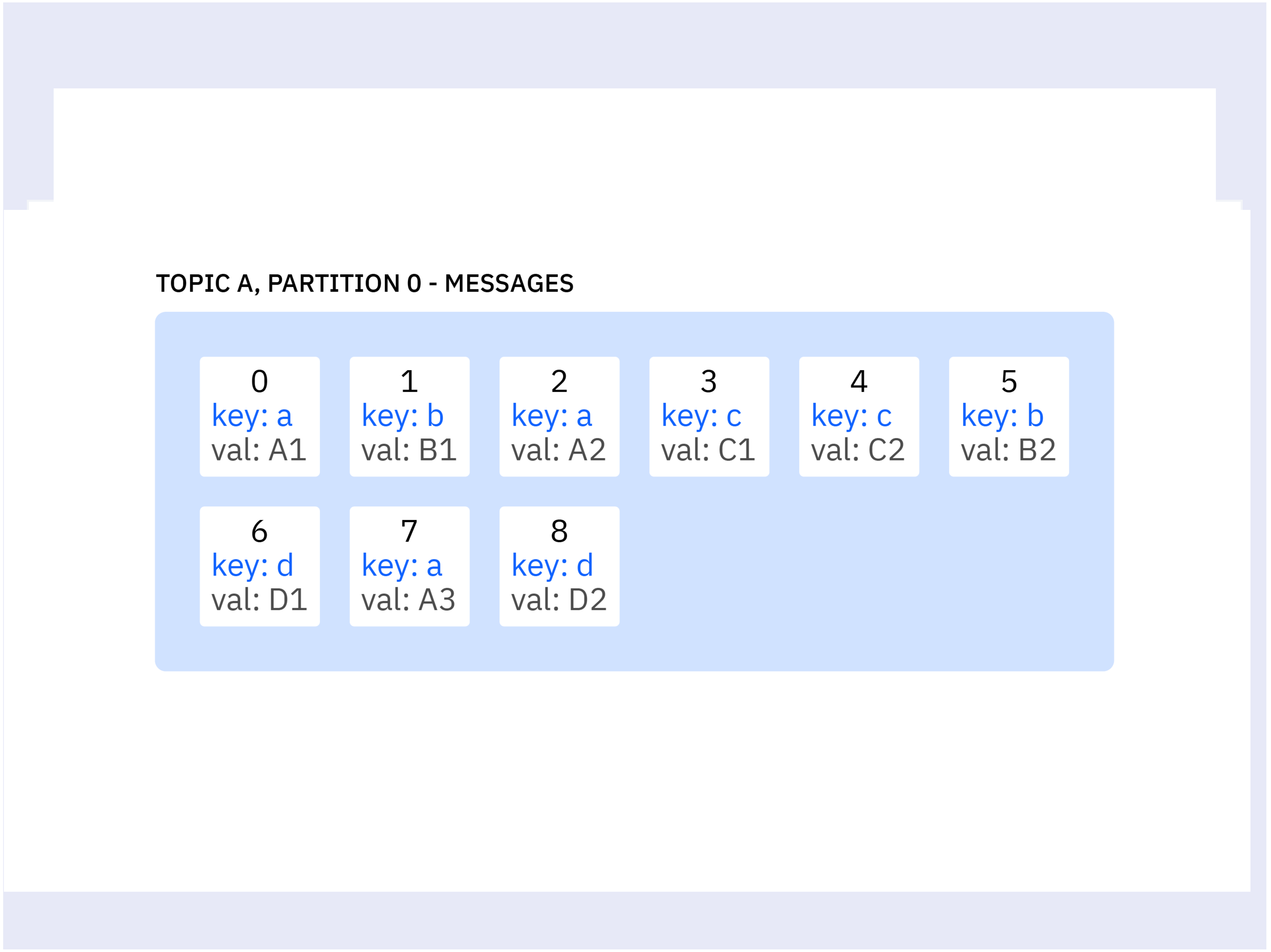


Topics and Partitions

Each topic is a named stream of messages. A **topic** is made up of one or more partitions. The messages on a partition are ordered by a number that is called the **offset**.

Why would a topic have more than one partition?

- It allows data to be fed through in parallel to increase throughput by distributing the partitions across the cluster.
- The number of partitions also influences the balancing of workload among consumers.

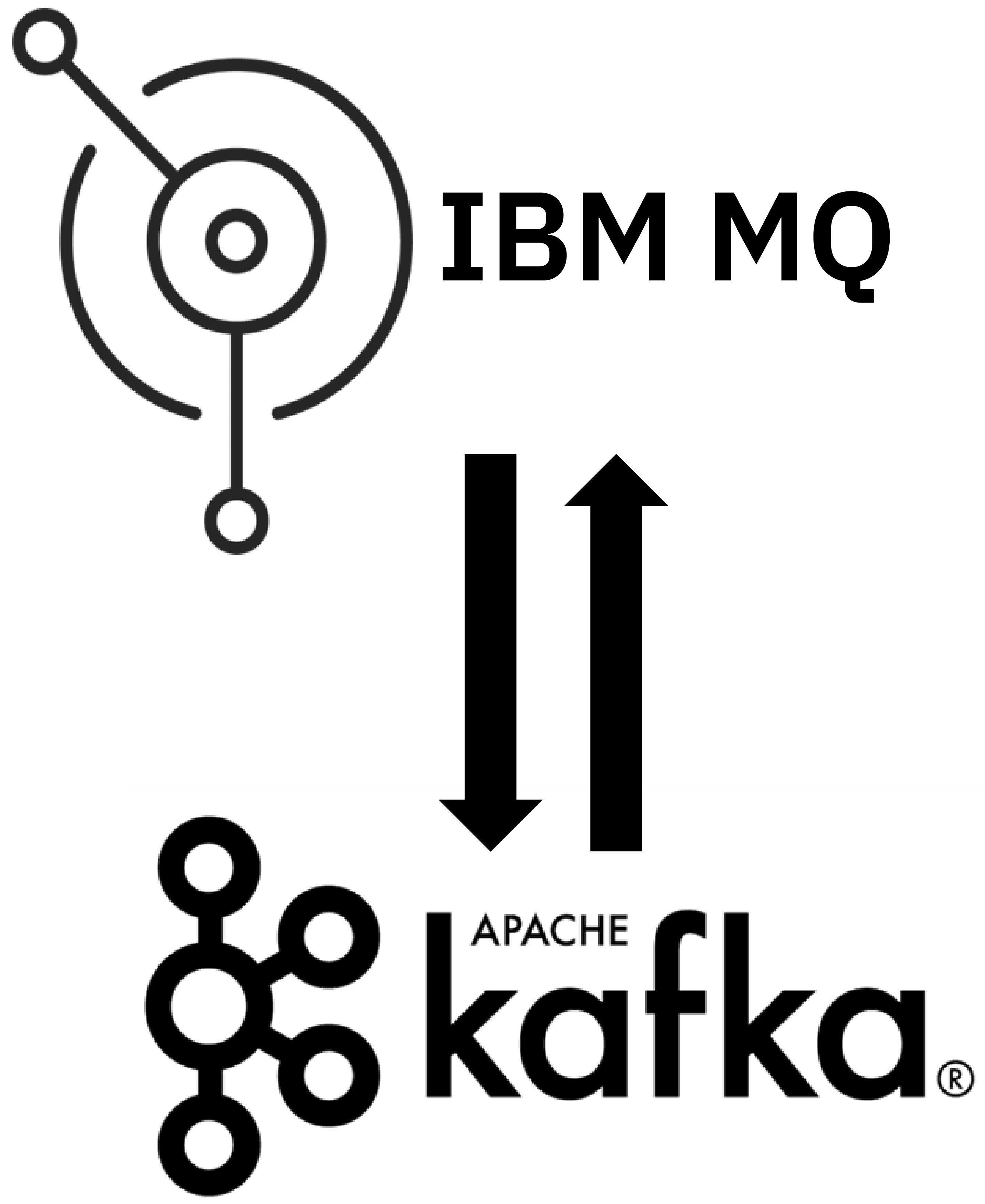


Connecting MQ and Kafka: why?

With IBM MQ and Apache Kafka specialising in different aspects of the messaging spectrum, one on connectivity and the other on data, solutions often require messages to flow between the two.

Common scenarios:

- Core banking system with MQ used as connectivity backbone. Business needs to take a copy of messages from MQ and push them into Kafka for analytics.
- Business needs to extend core banking system to emit data into Kafka, but doesn’t want to add network latency that might affect SLAs, so uses local queue manager as a bridge.
- Business needs to get data into Kafka from z/OS and wants to accelerate delivery so uses in-house MQ experience.
- Customer needs to get data into z/OS from distributed. Distributed development team has experience with Kafka, z/OS team want to exploit MQ integration with CICS / IMS.



Kafka Connect

As Kafka is commonly used for analytics there is a need to be able to get data into it from a wide range of sources

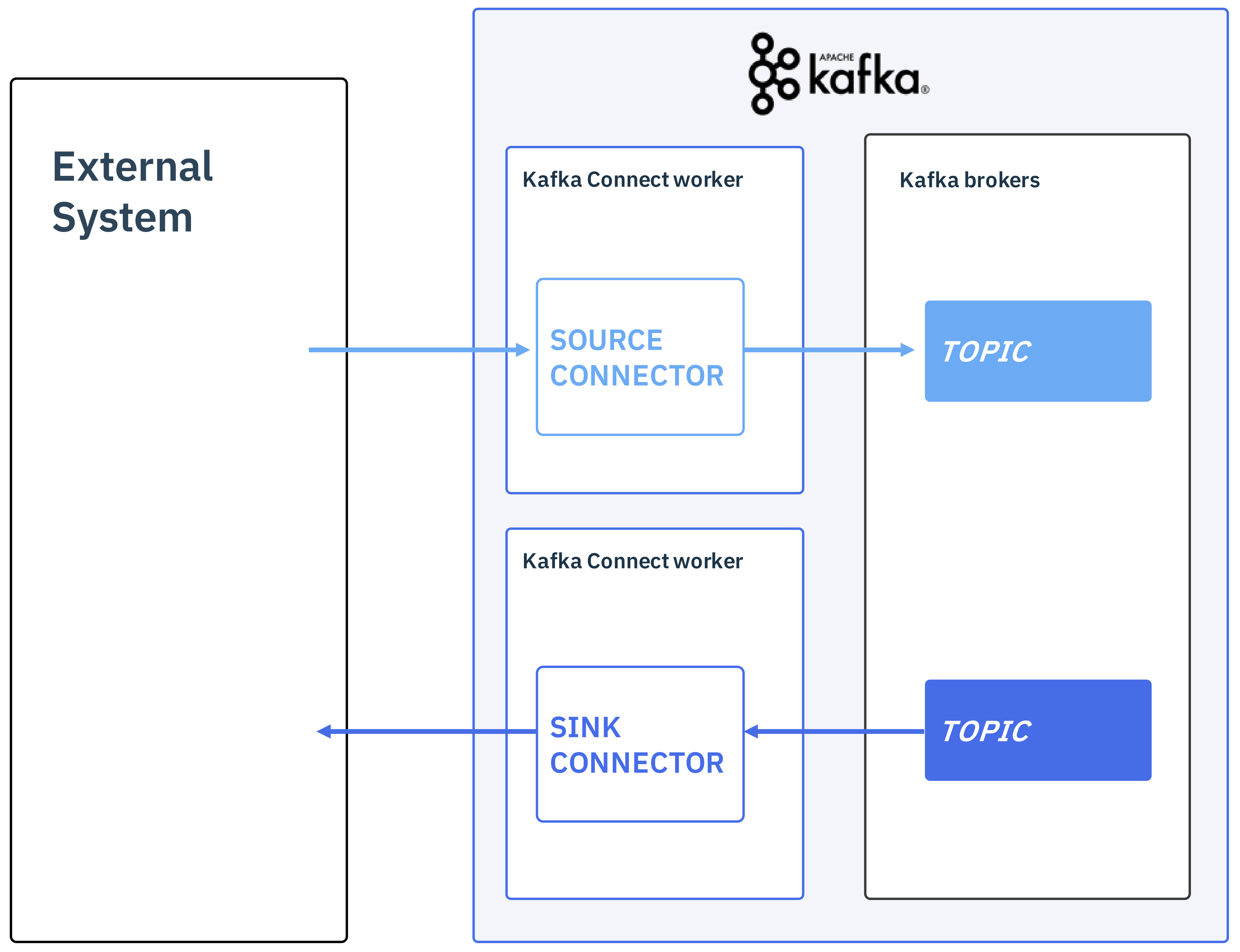
Kafka Connect provides a framework for this which makes it easy to build connectors

Source connectors: data from external system into Kafka

Sink connectors: data into external system from Kafka

Over 120 connectors exist today: IBM MQ, JDBC, ElasticSearch, Splunk…

Some supported, some not



NB: a connector is actually divided up into a connector which deals with configuration, and configuration changes, and a set of tasks which do the copying of data. For this presentation I will just use ‘connector’ to refer to both.

IBM MQ - Kafka Connector

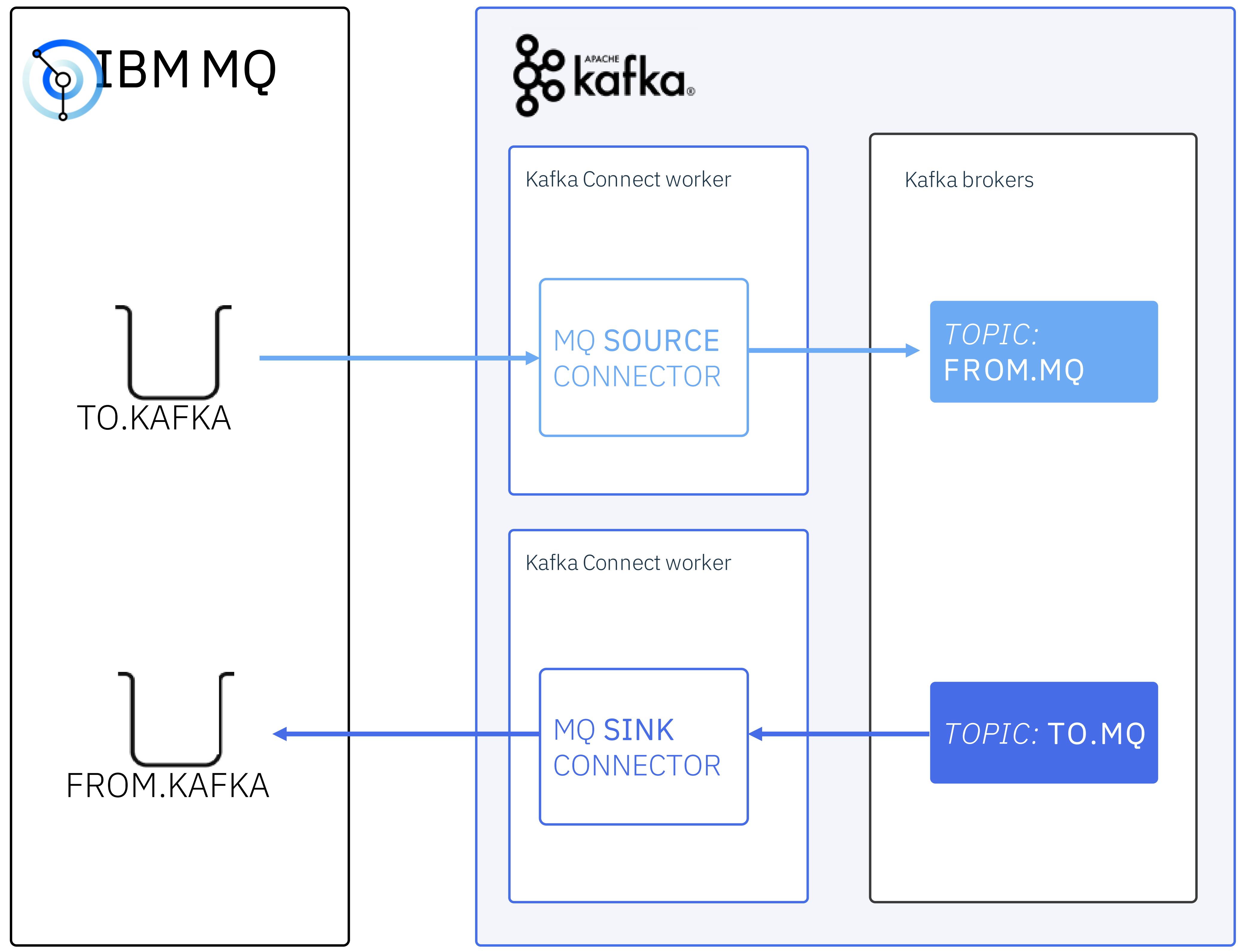
Several connectors exist for connecting MQ **queues** to Kafka **topics**

Both source and sink connectors exist. The source one is by far the most commonly used one today.

IBM provides functionally identical connectors in two forms:

– Unsupported open source
– Supported with IBM MQ Advanced
– Supported with CP4I (MQ base or MQ Advanced)

Confluent also provides a supported connector

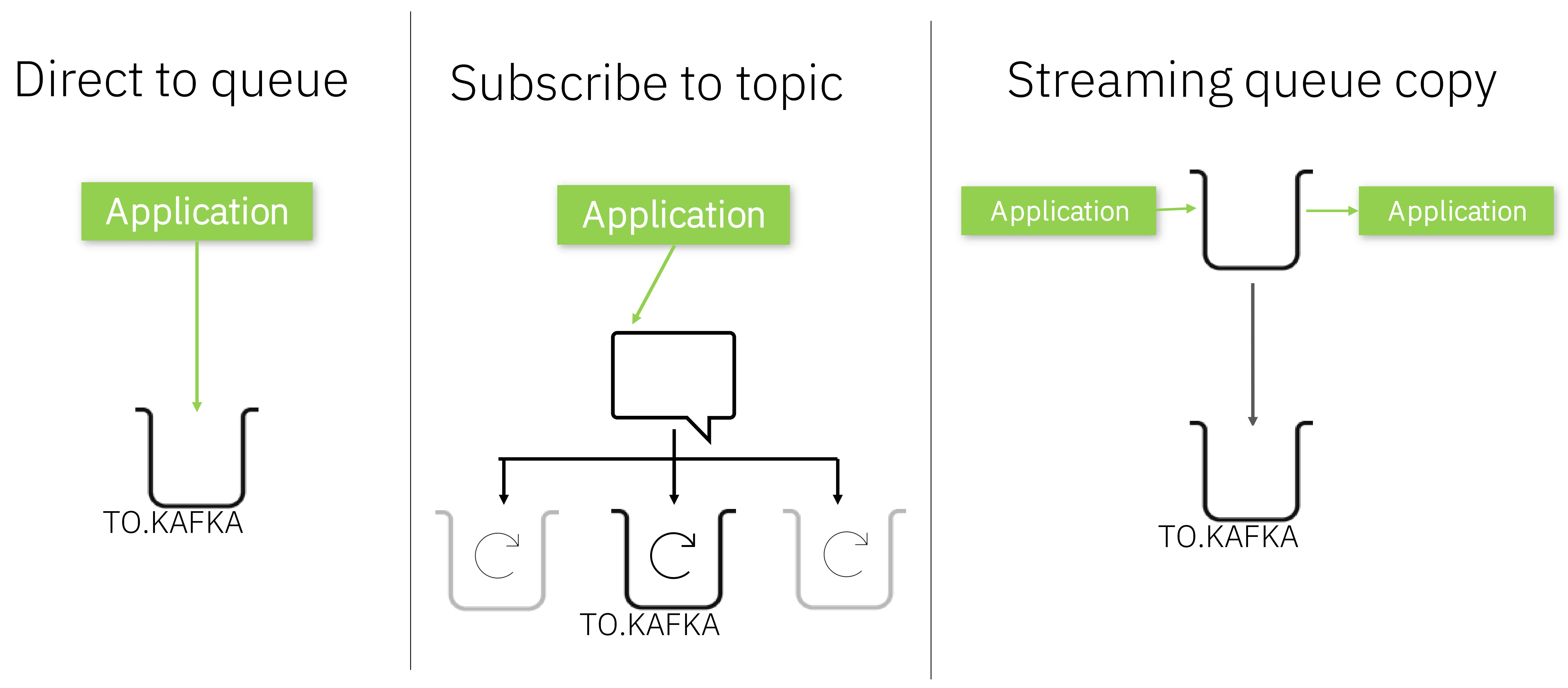


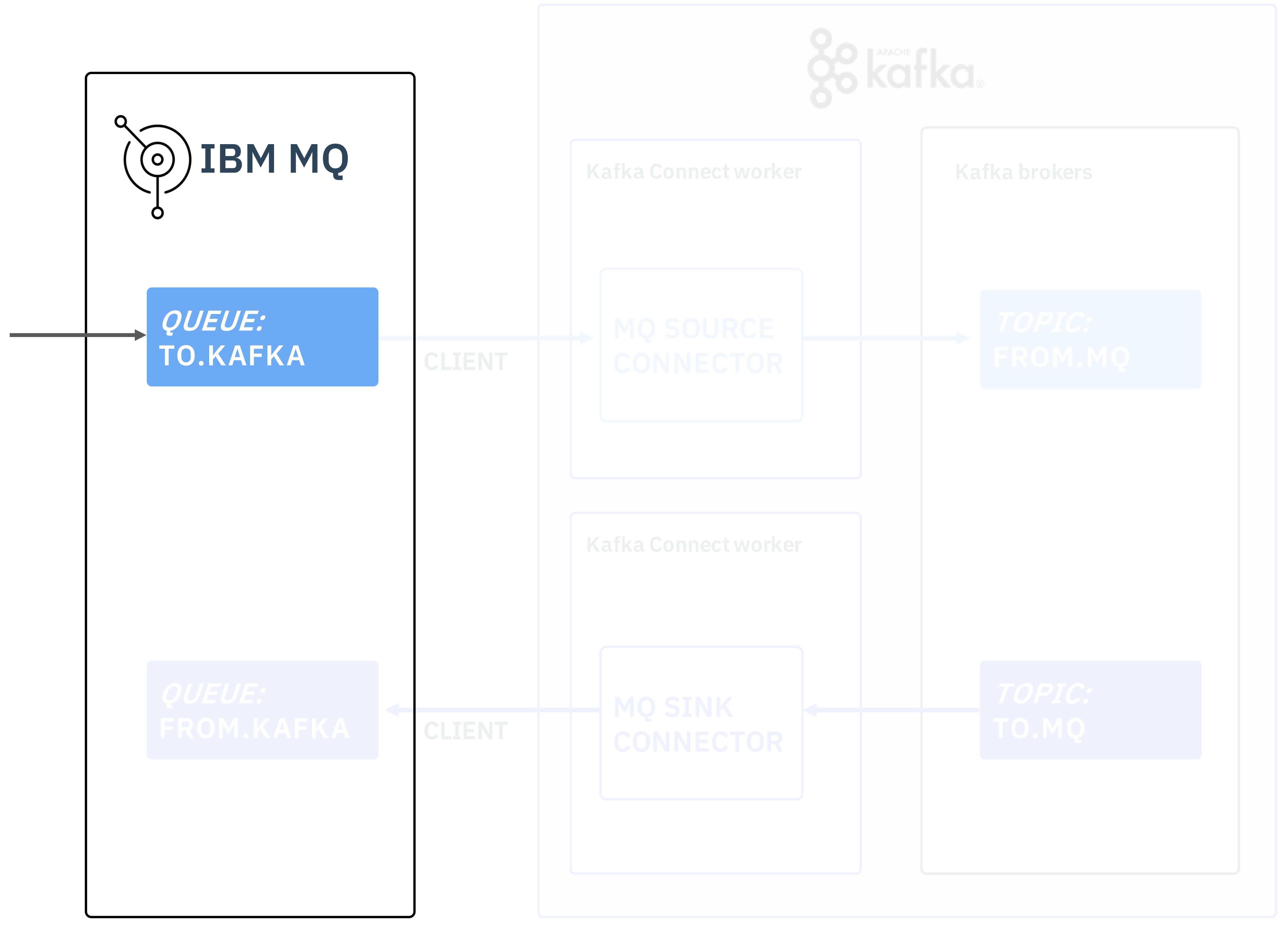
https://github.com/ibm-messaging/kafka-connect-mq-source
https://github.com/ibm-messaging/kafka-connect-mq-sink

Application options

The IBM source / sink connectors both require a queue name

With the source connector there are three different approaches that can be used to get messages onto the queue to be consumed by the connector…



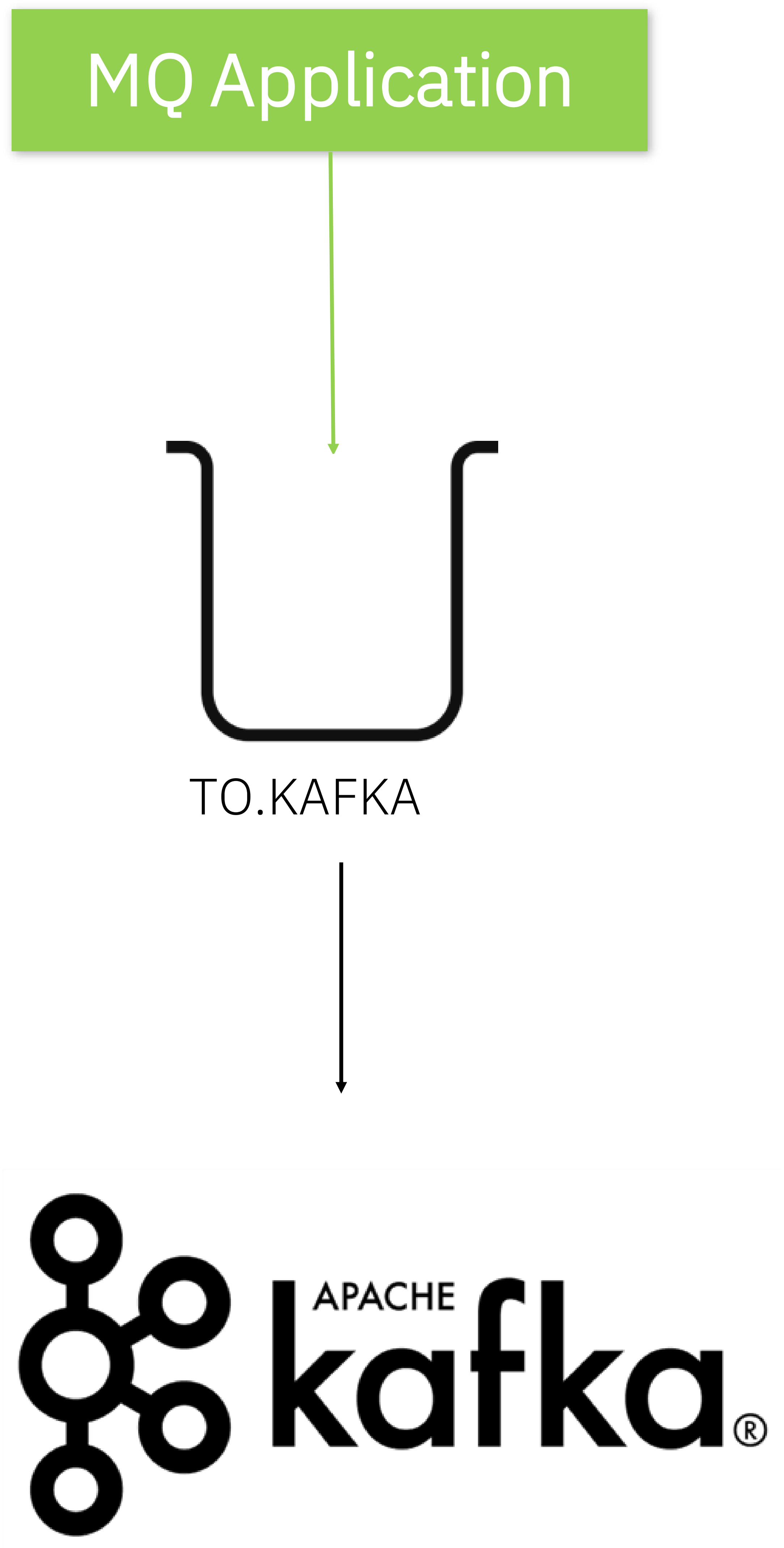


Direct to queue

New applications, or applications that are being changed, can just put the relevant data to the queue used by the connector

Tends to be used when you don’t want to generate a copy of existing messages

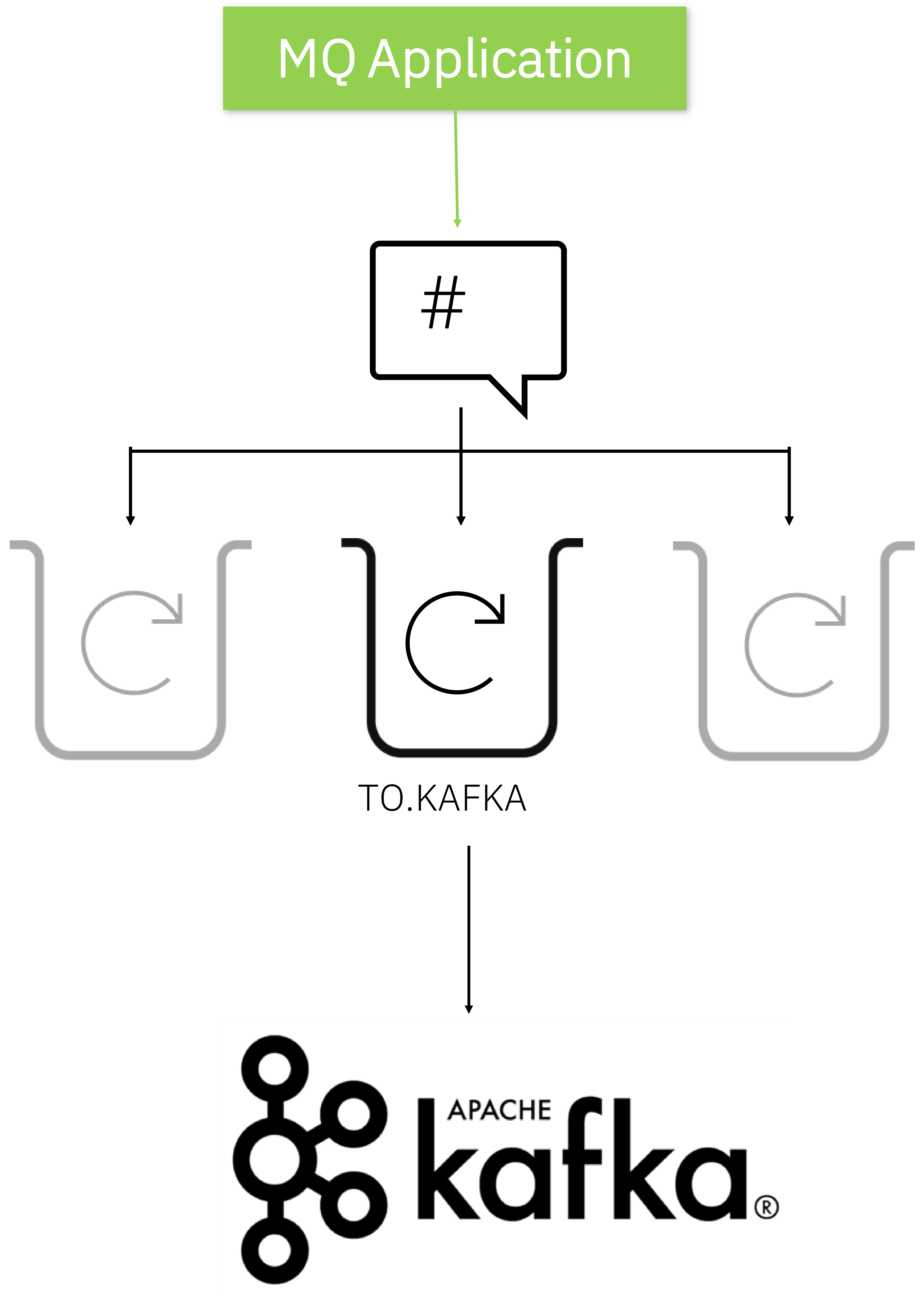
Often used on z/OS to remove the need for network latency associated with a direct connection to Kafka via REST, or for having to make use of more complicated approaches like exploiting the Kafka Java API in CICS



Subscribe to topic

If messages are already being published to a topic, its trivial to generate another copy of them by using an administrative subscription pointing to the queue used by the connector

This approach is transparent to existing applications



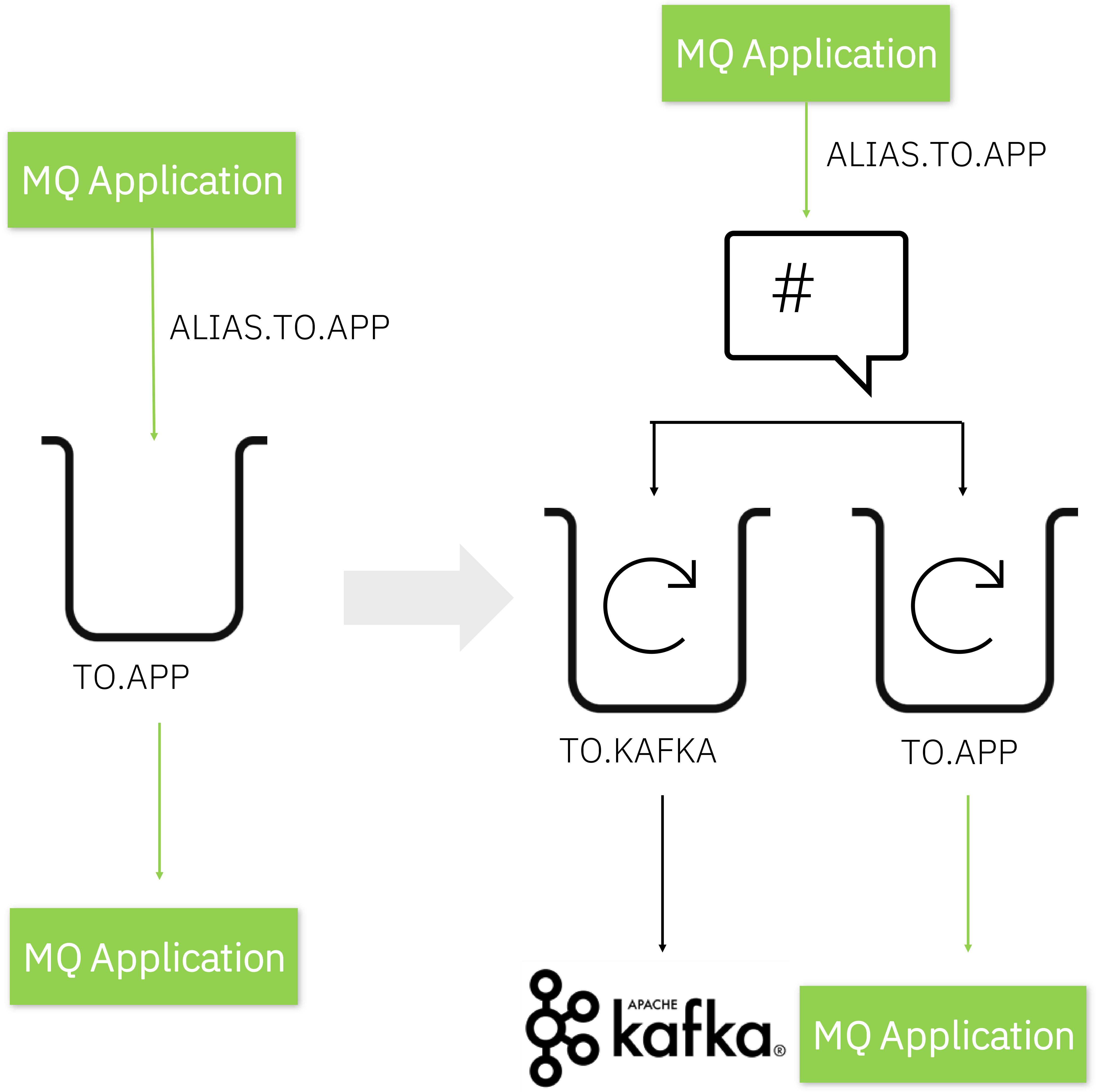
Subscribe to topic

This approach can also be used to take a copy of messages that are being sent to a queue

The queue is changed into a queue alias pointing to a topic. Two administrative subscriptions are created, one for application that used to consume from the queue, the other for the connector

This approach might not be viable in all cases depending on whether queue aliases have been used, or whether the application can be adjusted to use them.

It also relies on existing applications being able to tolerate the changes to message ID, correlation ID, etc when using pub / sub



Streaming queue copy

From 9.2.3 on distributed, there is an alternative option: use streaming queues

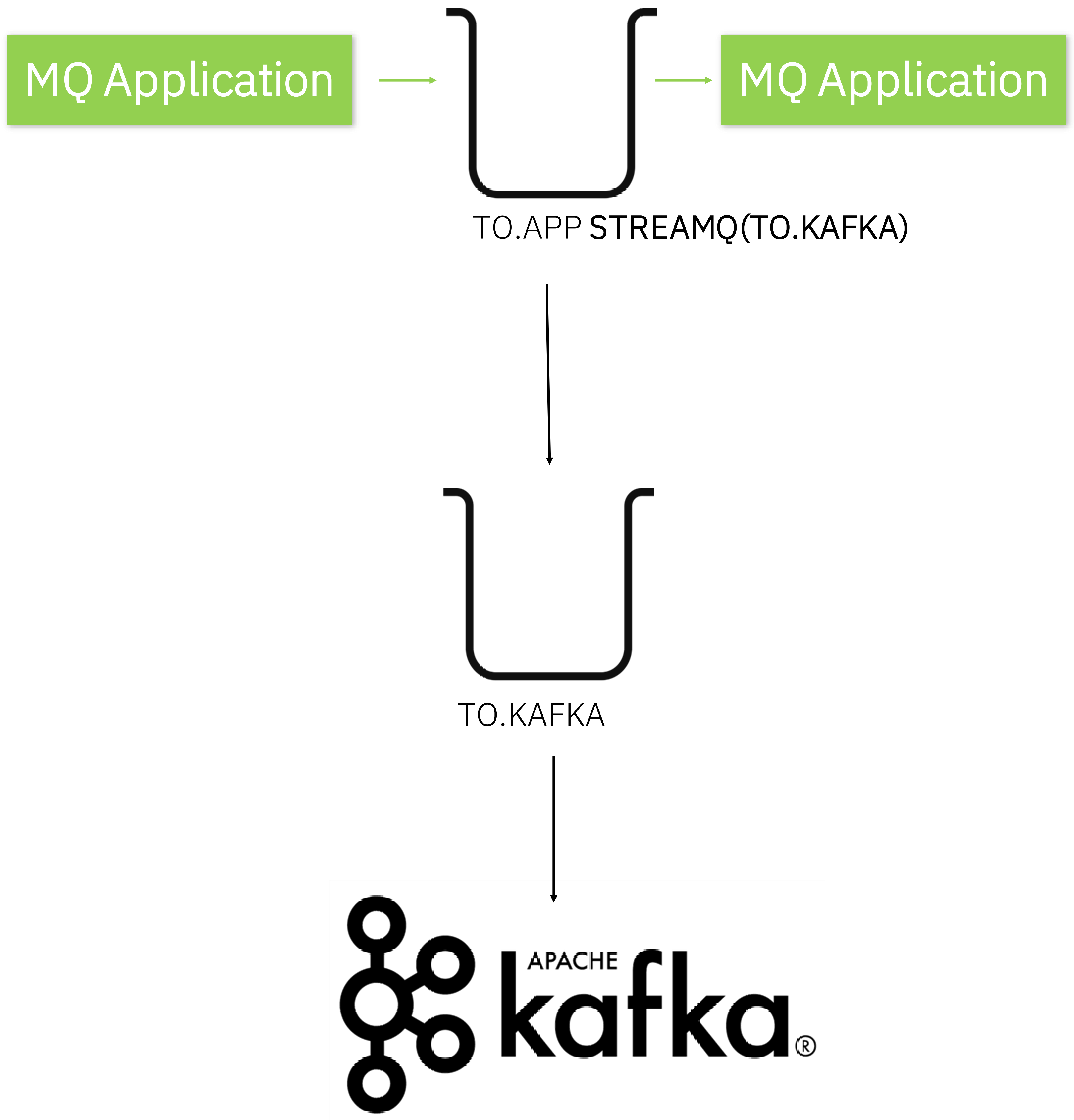
Streaming queues allows messages being put to one queue to be copied to a second queue without affecting the applications using the first queue

For example:

```
DEF QL(TO.APP) STREAMQ(TO.KAFKA) STRMQOS(MUSTDUP)
DEF QL(TO.KAFKA)
```

Says, when a message is sent to TO.APP a copy of that message must be sent to TO.KAFKA

Enabling streaming queues has no effects on existing applications as the original message doesn't change, the message sent to the second queue is identical (*) to the original message: same payload, message & correlation ID, etc



The source connector in detail

The connector is Java based and so uses the MQ JMS API to interact with MQ

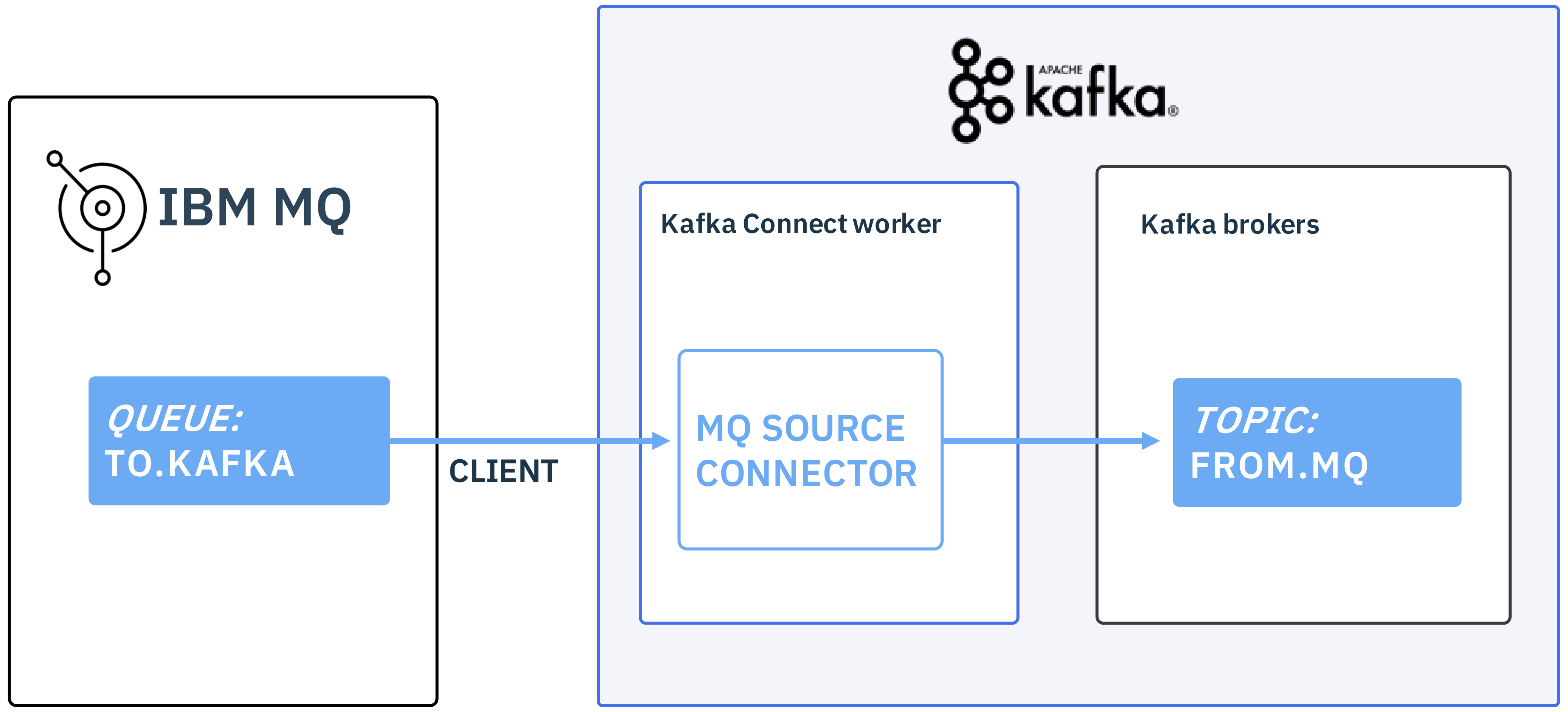
Provided in the form of a jar file. If using the open source connector, you build this yourself, otherwise you download it from CP4I

The connector is installed into Kafka Connect and run with a properties file containing configuration

Lots of flexibility in configuration:

- Client / bindings mode
- TLS including mutual auth
- User id and password
- JMS and MQ format messages
- Message properties

Client mode connections are the default



Example properties

```
# The name of the target Kafka topic
topic=FROM.MQ

# The name of the MQ queue manager mq.queue.manager=MQ1
mq.connection.mode=bindings

# The name of the source MQ queue
mq.queue=TO.KAFKA
```

Data conversion

Messages read in from MQ go through a number of conversions before being sent to Kafka

If the connector receives a message it can't convert it will stop to prevent throwing the message away, so understanding the data format is important!

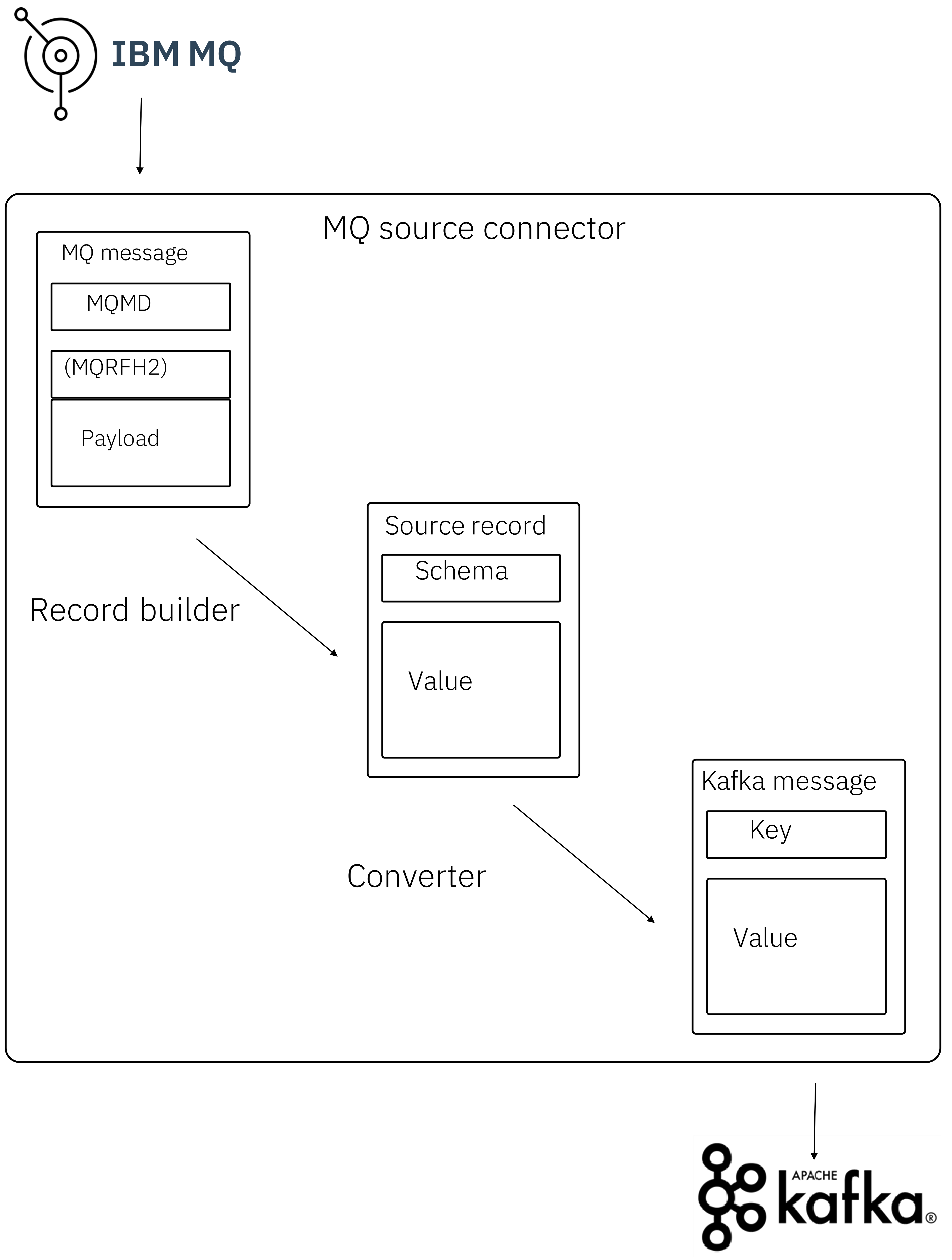
When a message is read from MQ it's payload is converted to an internal format known as a source record

This conversion is done via a record builder. Two record builders are provided with the connector (default, JSON) or you can write your own

The source record is then converted into a Kafka message

This conversion is done via a converter provided by Kafka (byte array, string, JSON)

The documentation for the connector provides recommendations for common use cases



Partitioning

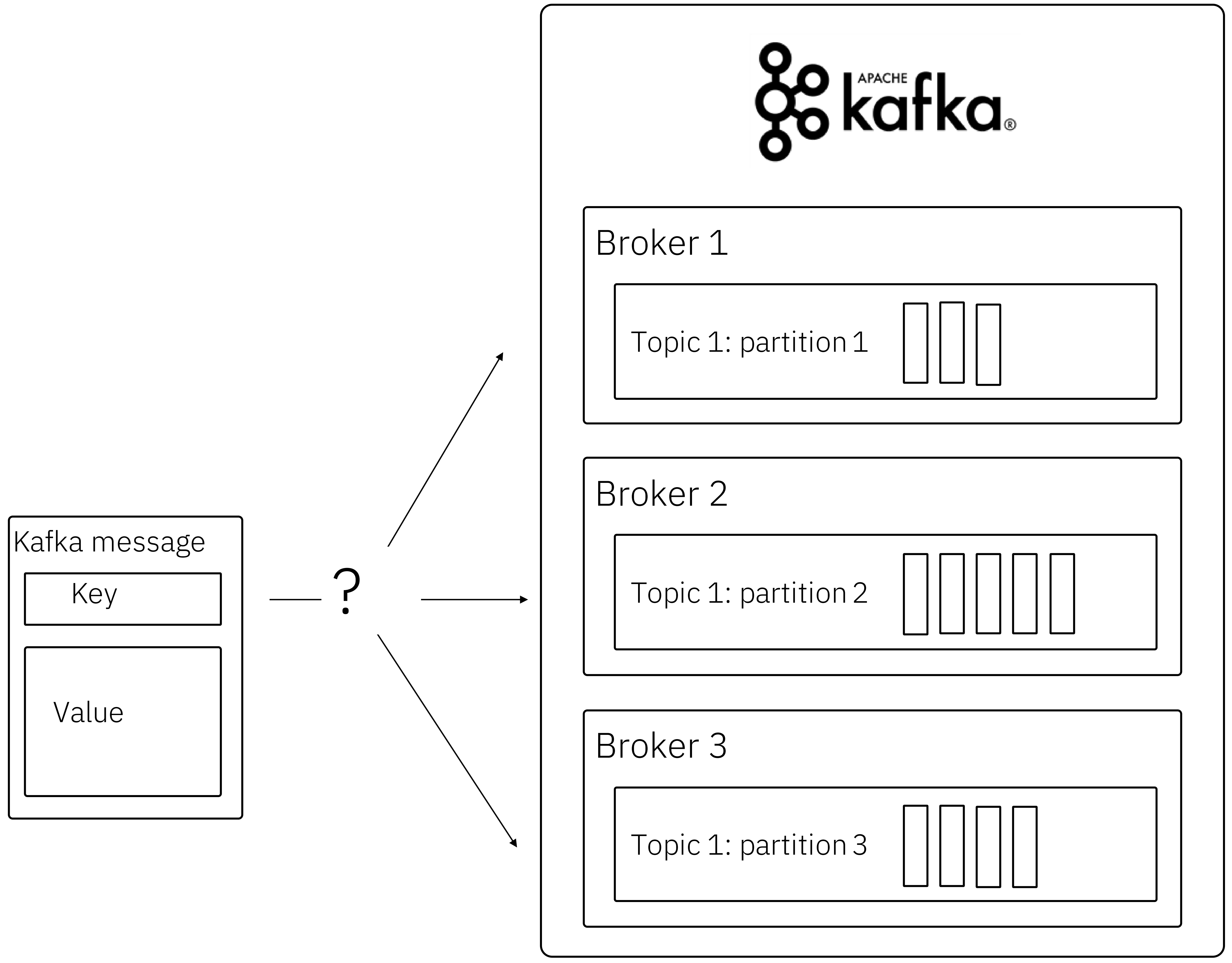
When messages are published to a Kafka topic they need to be spread across the different partitions of the topic

This is either done by a round-robin process, or if the message contains a key, the hash of the key is used to select a partition (same key => same partition)

Its also possible to write your own partitioning implementation

By default the MQ source connector doesn't specify a key

However it can be configured to use the MQ message ID, correlation ID, or the queue name as a key



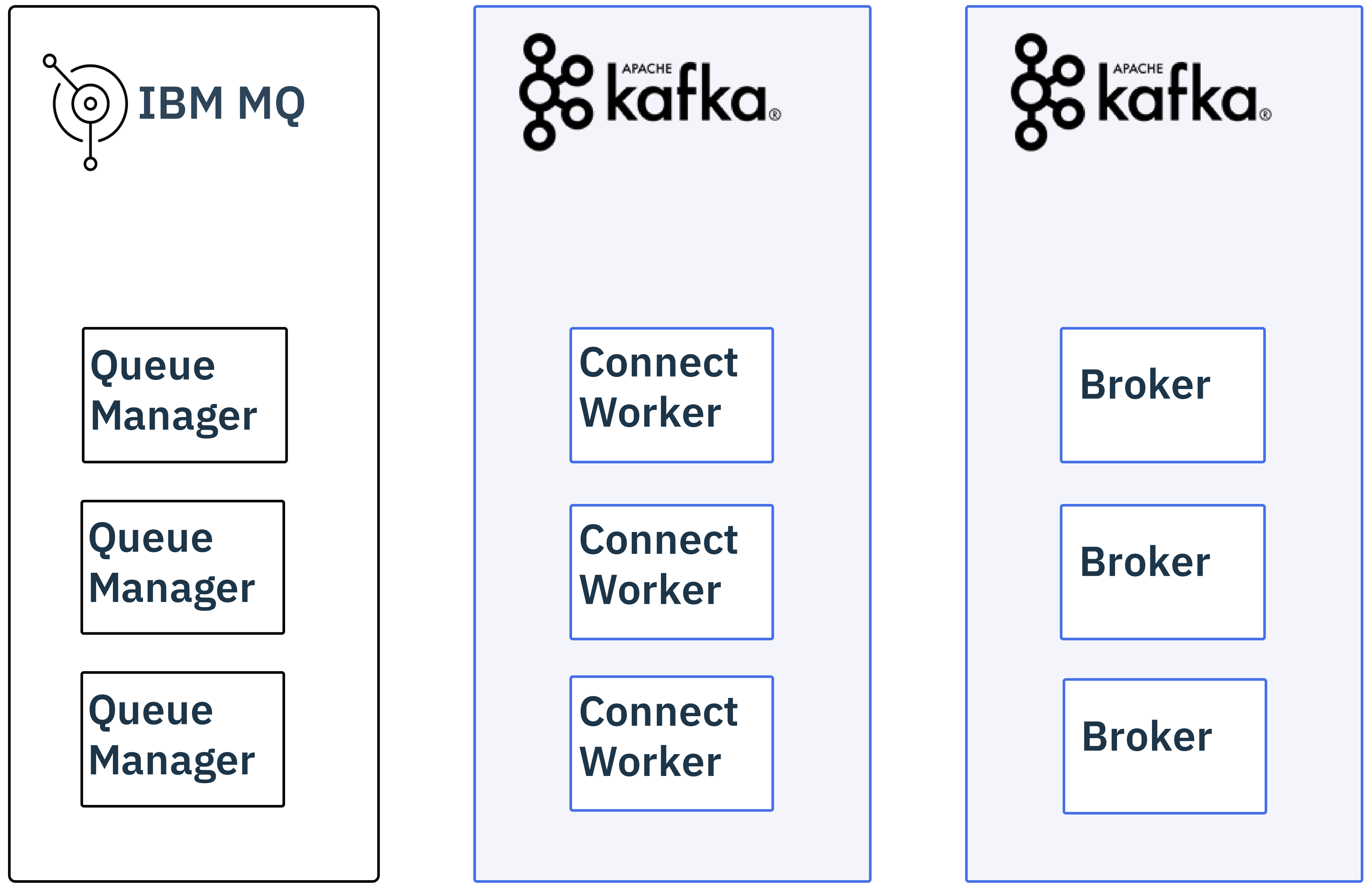
Fault tolerance and scalability

Both MQ and Kafka are highly fault tolerant and scalable. This extends to Kafka Connect and the MQ connectors

Kafka Connect can be run in two modes:

Standalone: a single Kafka Connect worker process runs the connector. This is useful for getting started with the connectors, or if you want guaranteed message ordering, but is a single point of failure, and can be a scalability bottle neck

Distributed: multiple Kafka Connect worker processes run across a set of machines and form a Kafka Connect cluster. Connectors are run across a number of these workers depending on configuration. If a worker process running a connector fails, the connector can be restarted on a different worker in the cluster. The workers collaborate to ensure they each have about the same amount of work



Fault tolerance and scalability

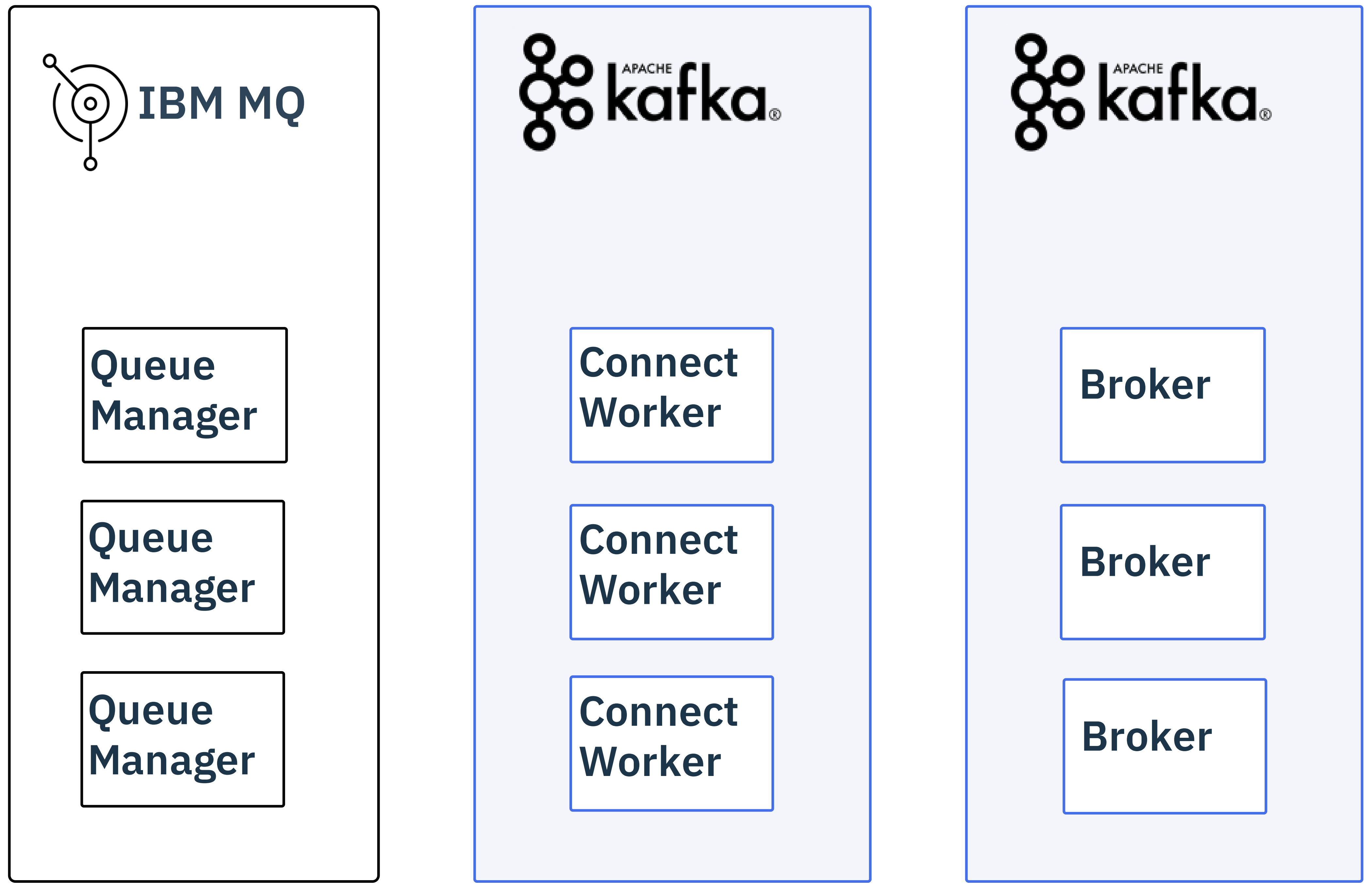
Messages are received from MQ in batches using a JMS transacted session. I.e using a transaction coordinated by the queue manager

If there is any failure in converting any MQ messages then the transaction is rolled back and the messages can be reprocessed later

The connector automatically deals with reconnections to MQ if needed

Similarly the Kafka Connect framework automatically deals with reconnections to Kafka if needed, depending on configuration

Kafka doesn't have the ability to take part in two phase commit transactions. Therefore some failure scenarios might end up with MQ messages being written to the Kafka topic multiple times



Using the connector on z/OS

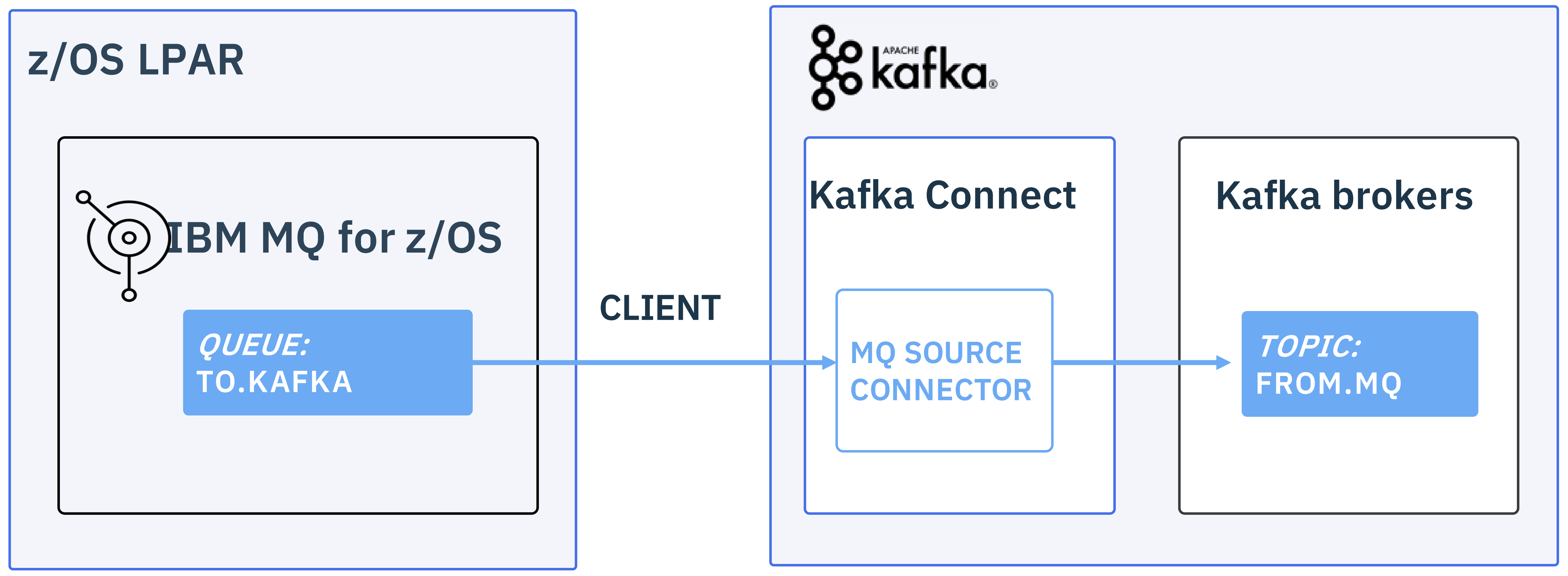
Lots of customers on z/OS have MQ and use it to communicate between z/OS LPARs as well as between z/OS and distributed

There has been **a lot** of interest in the MQ connector on z/OS

It's possible to run Kafka Connect on distributed and connect to z/OS as a client

This is the same model you would likely use with distributed MQ

However …



Using the connector on z/OS

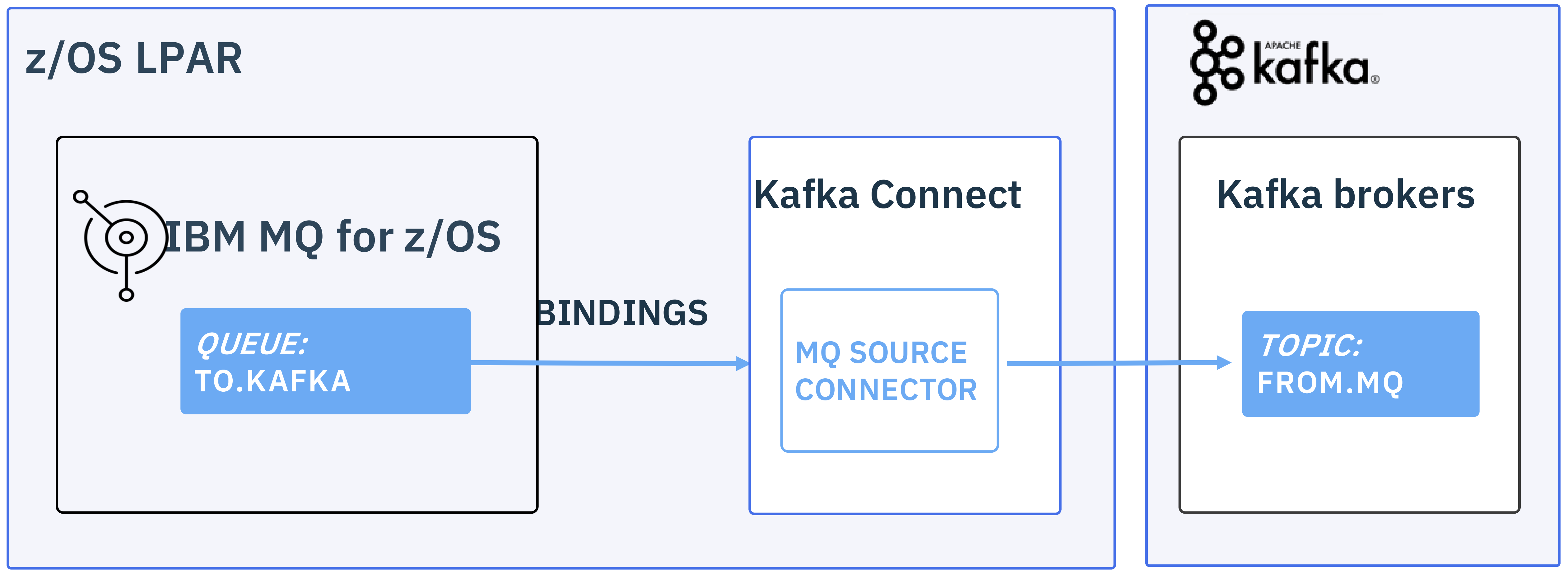
An alternative is to run the connector on z/OS in USS and connect to the queue manager in bindings

The connector then connects to the remote Kafka cluster over the network. Connections to Kafka are always network based

This model uses less MQ CPU, and because Kafka connect is Java based it is zIIP eligible making costs very completive

Kafka Connect works fine on z/OS. However various properties files and shell scripts need to be converted to EBCDIC first

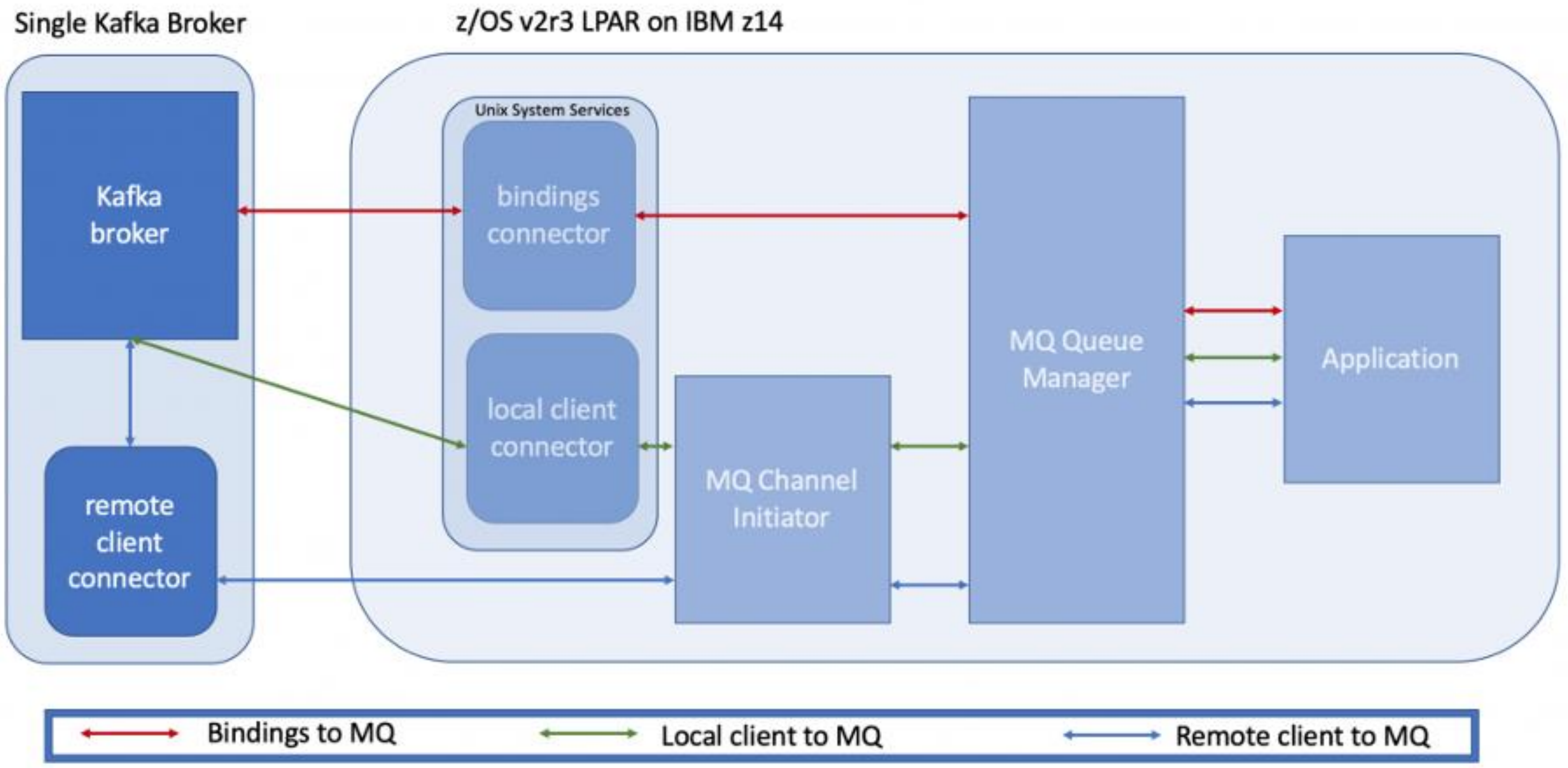
The conversion is documented here: https://ibm.github.io/event-streams/connecting/mq/zos/



Connector location	Connection to MQ	Total CPU US	MQ CPU US	Connector CPU US
USS	Bindings	100.6 (4.9)	2.4	98.2 (2.5)
USS	Client (Advanced VUE only)	152.9 (60.9)	58.4	94.5 (2.5)
Distributed	Client	55.7	55.7	N/A

Values in brackets are if maximum zIIP offload is achieved

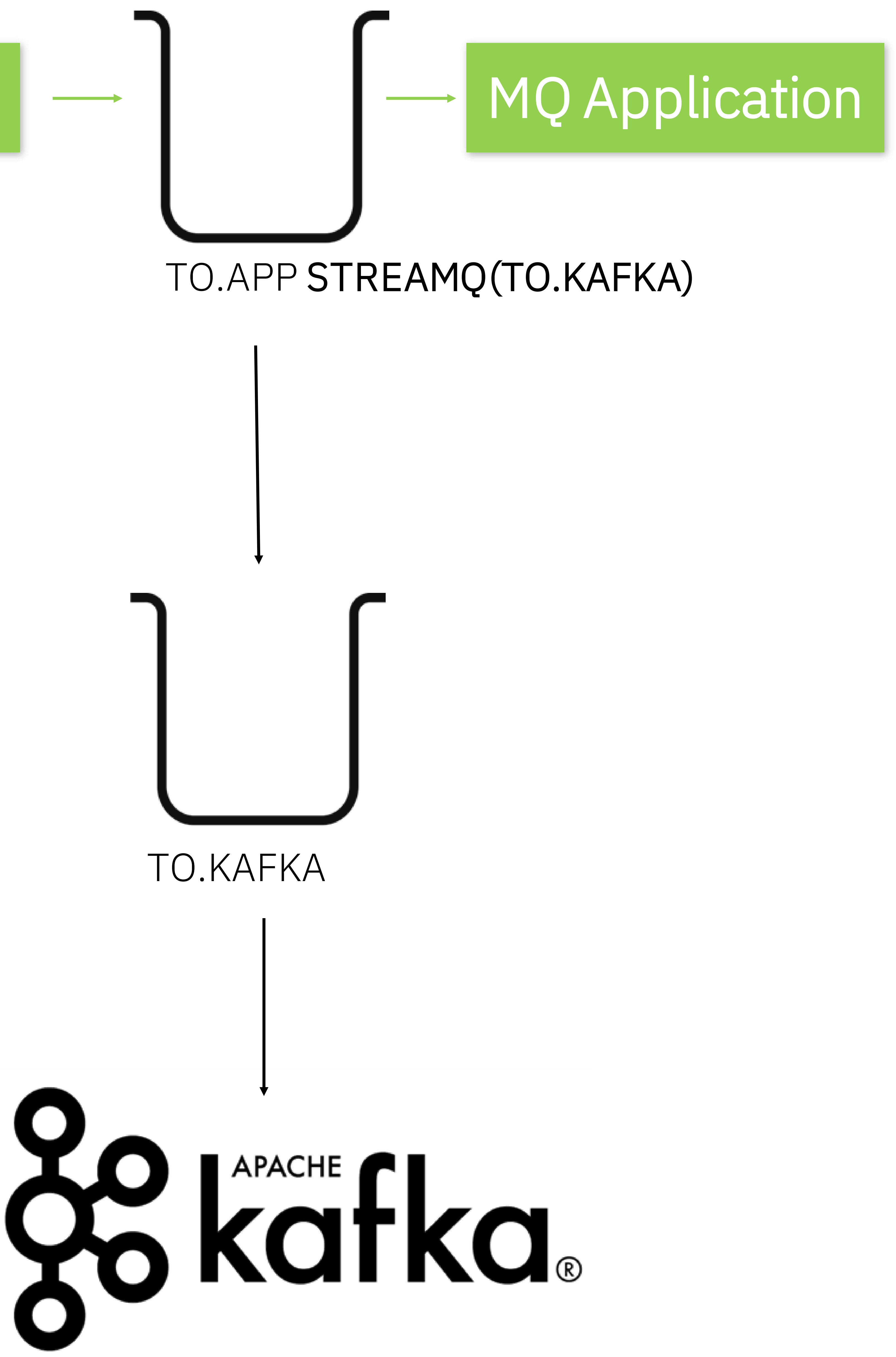
https://www.imwuc.org/HigherLogic/System/DownloadDocumentFile.ashx?DocumentFileKey=179ebb75-7be2-42aa-6f0e-818aeef805f2


Single Kafka Broker
z/OS v2r3 LPAR on IBM z14
Kafka broker
Unix System Services
bindings connector
local client connector
remote client connector
MQ Channel Initiator
MQ Queue Manager
Application
Bindings to MQ
Local client to MQ
Remote client to MQ

Demo

Starting with the following on Linux:

MQ 9.2.3 installed
Latest Kafka installed
MQ source connector built and installed



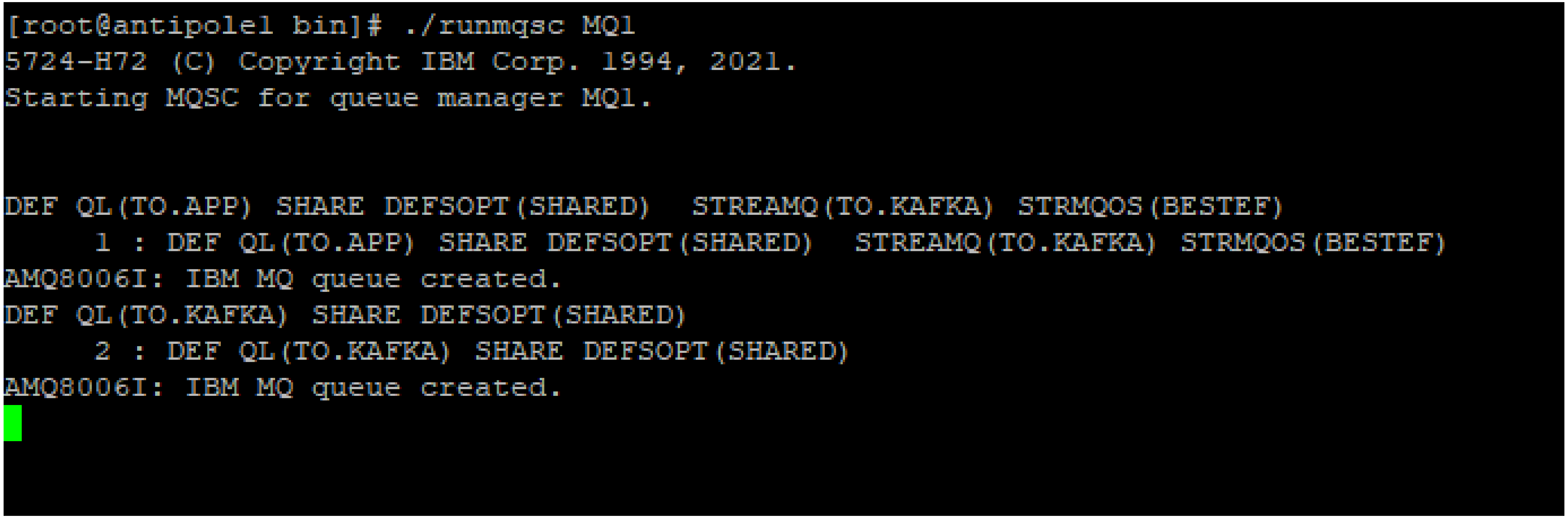
Start Zookeeper

```
[root@antipole1 kafka_2.13-3.0.0]# bin/zookeeper-server-start.sh config/zookeeper.properties
[2021-11-09 01:49:03,177] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,183] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,197] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,197] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,197] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,197] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,200] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2021-11-09 01:49:03,200] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2021-11-09 01:49:03,200] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2021-11-09 01:49:03,200] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2021-11-09 01:49:03,205] INFO Log4j 1.2 jmx support found and enabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2021-11-09 01:49:03,220] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,220] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,222] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,222] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,222] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,222] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2021-11-09 01:49:03,224] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2021-11-09 01:49:03,248] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@52af6cff (org.apache.zookeeper.server.ServerMetrics)
[2021-11-09 01:49:03,253] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2021-11-09 01:49:03,270] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,270] INFO   ______                  _                                          (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,270] INFO  |___  /                 | |                                         (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,270] INFO     / /    ___     ___   | | __   ___    ___   _ __     ___   _ __   (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,270] INFO    / /    / _ \   / _ \  | |/ /  / _ \  / _ \ | '_ \   / _ \ | '__| (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,270] INFO   / /__  | (_) | | (_) | |   <  |  __/ |  __/ | |_) | |  __/ | |     (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,270] INFO  /_____|  \___/   \___/  |_|\_\  \___|  \___| | .__/   \___| |_| (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,270] INFO                                               | |                     (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,270] INFO                                               |_|                     (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,270] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,272] INFO Server environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,273] INFO Server environment:host.name=antipole1.fyre.ibm.com (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,273] INFO Server environment:java.version=1.8.0_312 (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,273] INFO Server environment:java.vendor=Red Hat, Inc. (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,273] INFO Server environment:java.home=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-1.el8_4.x86_64/jre (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-09 01:49:03,291] INFO Server environment:java.class.path=/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/activation-1.1.1.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/aopa
afka_2.13-3.0.0/bin/../libs/audience-annotations-0.5.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/commons-cli-1.4.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/commons-lang
bin/../libs/connect-basic-auth-extension-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/connect-file-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/connect-json-3.0.
/../libs/connect-mirror-client-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/connect-runtime-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/connect-transforms-3.0.0
```

Start single Kafka broker

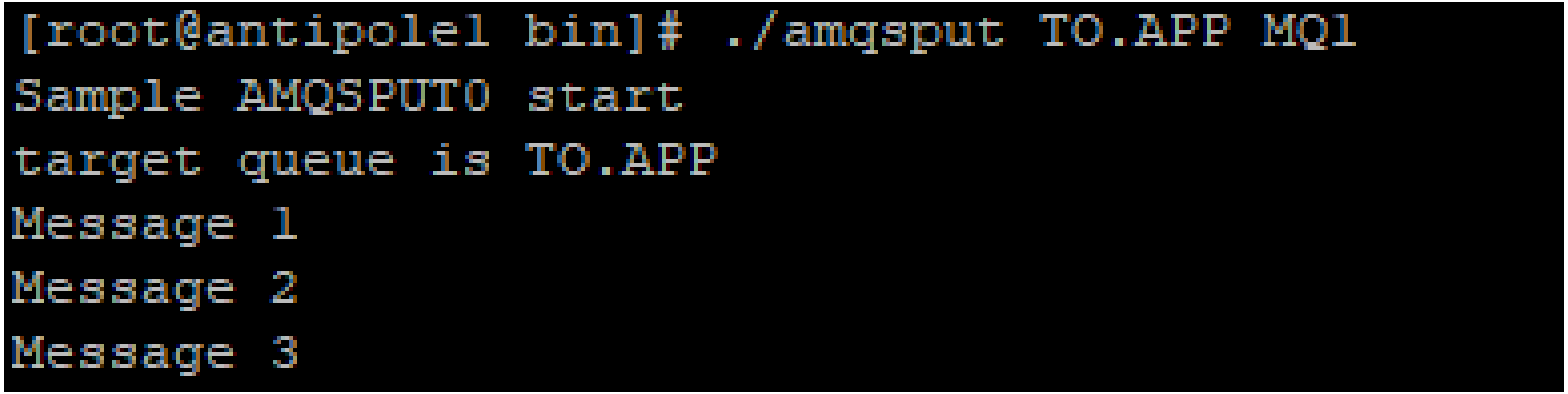
```
[root@antipolel kafka_2.13-3.0.0]# bin/kafka-server-start.sh config/server.properties
[2021-11-09 01:50:41,186] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2021-11-09 01:50:41,922] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.
[2021-11-09 01:50:42,123] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)
[2021-11-09 01:50:42,156] INFO starting (kafka.server.KafkaServer)
[2021-11-09 01:50:42,157] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2021-11-09 01:50:42,186] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2021-11-09 01:50:42,197] INFO Client environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (org.apache
[2021-11-09 01:50:42,197] INFO Client environment:host.name=antipolel.fyre.ibm.com (org.apache.zookeeper.ZooKeeper)
[2021-11-09 01:50:42,197] INFO Client environment:java.version=1.8.0_312 (org.apache.zookeeper.ZooKeeper)
[2021-11-09 01:50:42,197] INFO Client environment:java.vendor=Red Hat, Inc. (org.apache.zookeeper.ZooKeeper)
[2021-11-09 01:50:42,197] INFO Client environment:java.home=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-1.el8_4.x86_64/jre (org.apache.zookeeper.ZooKeeper)
[2021-11-09 01:50:42,198] INFO Client environment:java.class.path=/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/activation-1.1.1.jar:/root/gse/kafkadrive
afka_2.13-3.0.0/bin/../libs/audience-annotations-0.5.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/commons-cli-1.4.jar:/root/gse/kafkadriver/kafka_
bin/../libs/connect-basic-auth-extension-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/connect-file-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-
/../libs/connect-mirror-client-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/connect-runtime-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/b
/hk2-locator-2.6.1.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/hk2-utils-2.6.1.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jackson-annot
d-2.12.3.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jackson-dataformat-csv-2.12.3.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jackson-d
kson-jaxrs-json-provider-2.12.3.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jackson-module-jaxb-annotations-2.12.3.jar:/root/gse/kafkadriver/kafka_
driver/kafka_2.13-3.0.0/bin/../libs/jakarta.annotation-api-1.3.5.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jakarta.inject-2.6.1.jar:/root/gse/kaf
/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jakarta.xml.bind-api-2.3.2.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/javassist-3.27.0-GA.jar:/root/gse/
kadriver/kafka_2.13-3.0.0/bin/../libs/jaxb-api-2.3.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jersey-client-2.34.jar:/root/gse/kafkadriver/kafka
_2.13-3.0.0/bin/../libs/jersey-container-servlet-core-2.34.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jersey-hk2-2.34.jar:/root/gse/kafkadriver/ka
fka_2.13-3.0.0/bin/../libs/jetty-continuation-9.4.43.v20210629.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jetty-http-9.4.43.v20210629.jar:/root/gs
:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jetty-server-9.4.43.v20210629.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jetty-servlet-9.4.43.
til-9.4.43.v20210629.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jetty-util-ajax-9.4.43.v20210629.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/..
.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/kafka-clients-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/kafka-log4j-appender-3.
ot/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/kafka-server-common-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/kafka-shell-3.0.0.jar:/root/g
fkadriver/kafka_2.13-3.0.0/bin/../libs/kafka-streams-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/kafka-streams-examples-3.0.0.jar:/root/gse/k
r:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/kafka-tools-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/log4j-1.2.17.jar:/root/gse/kafka
2.13-3.0.0/bin/../libs/metrics-core-2.2.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/metrics-core-4.1.12.1.jar:/root/gse/kafkadriver/kafka_2.13-3.
3.0.0/bin/../libs/netty-common-4.1.62.Final.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/netty-handler-4.1.62.Final.jar:/root/gse/kafkadriver/kafka_
r/kafka_2.13-3.0.0/bin/../libs/netty-transport-native-epoll-4.1.62.Final.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/netty-transport-native-unix-co
anamer-2.8.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/plexus-utils-3.2.1.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/reflections-0.9.12
.4.4.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/scala-java8-compat_2.13-1.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/scala-library
13.6.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/slf4j-api-1.7.30.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/slf4j-log4j12-1.7.30.jar:/
```

Define MQ queues



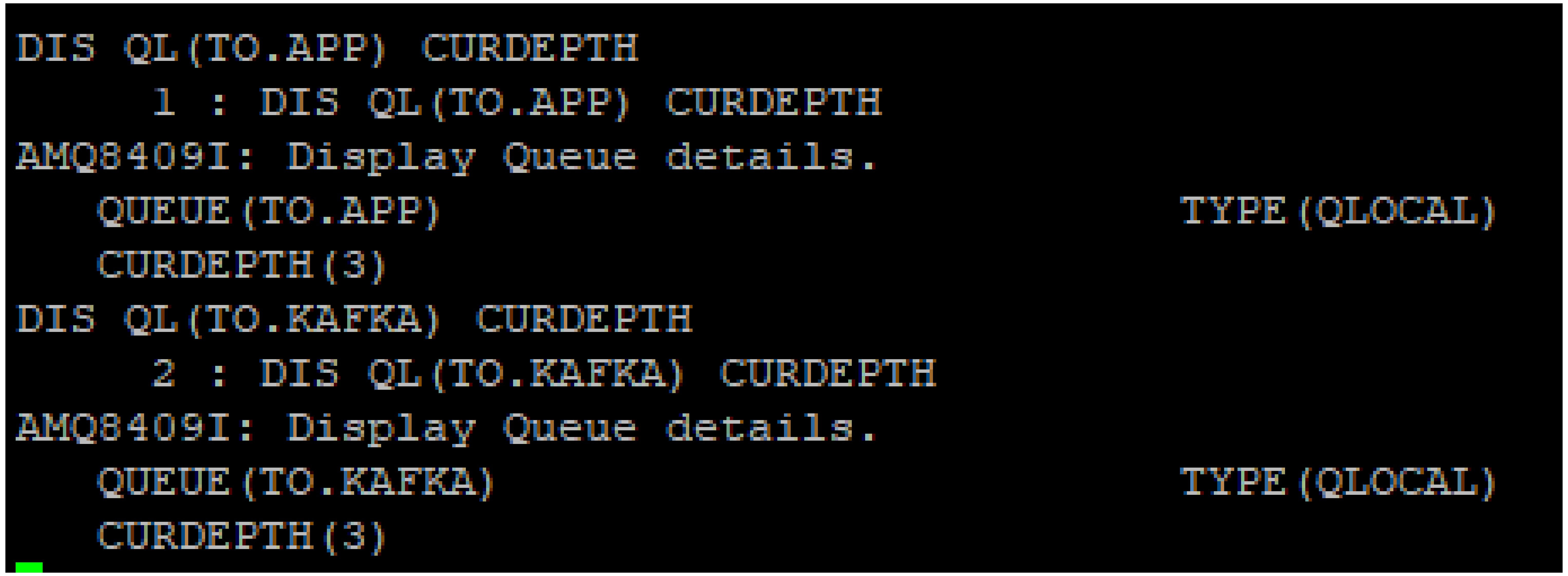
```
[root@antipole1 bin]# ./runmqsc MQ1
5724-H72 (C) Copyright IBM Corp. 1994, 2021.
Starting MQSC for queue manager MQ1.


DEF QL(TO.APP) SHARE DEFSOPT(SHARED)  STREAMQ(TO.KAFKA) STRMQOS(BESTEF)
     1 : DEF QL(TO.APP) SHARE DEFSOPT(SHARED)  STREAMQ(TO.KAFKA) STRMQOS(BESTEF)
AMQ8006I: IBM MQ queue created.
DEF QL(TO.KAFKA) SHARE DEFSOPT(SHARED)
     2 : DEF QL(TO.KAFKA) SHARE DEFSOPT(SHARED)
AMQ8006I: IBM MQ queue created.
```

Send a couple of messages



```
[root@antipole1 bin]# ./amqsput TO.APP MQ1
Sample AMQSPUT0 start
target queue is TO.APP
Message 1
Message 2
Message 3
```

Check they are there

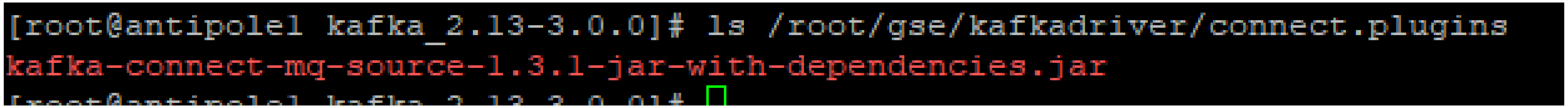


```
DIS QL(TO.APP) CURDEPTH
     1 : DIS QL(TO.APP) CURDEPTH
AMQ8409I: Display Queue details.
   QUEUE(TO.APP)                            TYPE(QLOCAL)
   CURDEPTH(3)
DIS QL(TO.KAFKA) CURDEPTH
     2 : DIS QL(TO.KAFKA) CURDEPTH
AMQ8409I: Display Queue details.
   QUEUE(TO.KAFKA)                          TYPE(QLOCAL)
   CURDEPTH(3)
```

Define the Kafka topic

```
[root@antipole1 kafka_2.13-3.0.0]# bin/kafka-topics.sh --create --topic KAFKA.TOPIC --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Created topic KAFKA.TOPIC.
```

Check connector is installed and configured



```
[root@antipole1 kafka_2.13-3.0.0]# ls /root/gse/kafkadriver/connect.plugins
kafka-connect-mq-source-1.3.1-jar-with-dependencies.jar
[root@antipole1 kafka_2.13-3.0.0]#
```

```
[root@antipole1 kafka_2.13-3.0.0]# cat config/mq-source.properties
name=mq-source
connector.class=com.ibm.eventstreams.connect.mqsource.MQSourceConnector

# You can increase this for higher throughput, but message ordering will be lost
tasks.max=1

# The name of the target Kafka topic - required
topic=KAFKA.TOPIC

# The name of the MQ queue manager - required
mq.queue.manager=MQ1

# The connection mode to connect to MQ - client (default) or bindings - optional
# mq.connection.mode=client
mq.connection.mode=bindings

# The name of the source MQ queue - required
mq.queue=TO.KAFKA
```

Start a Kafka consumer, nothing there...

```
[root@antipole1 kafka_2.13-3.0.0]# bin/kafka-console-consumer.sh --topic KAFKA.TOPIC --from-beginning --bootstrap-server localhost:9092
```

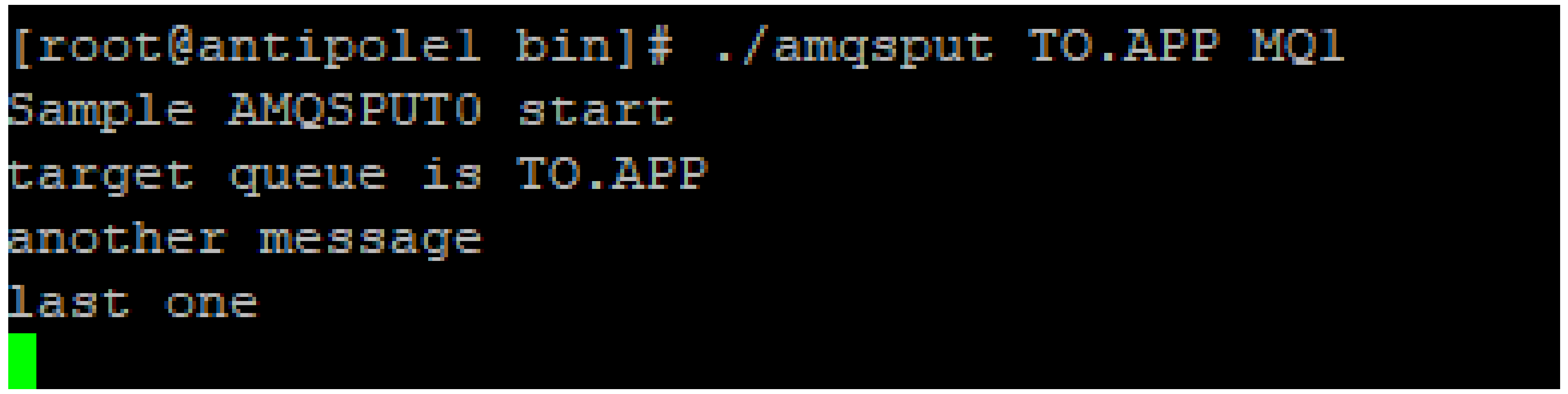
Start the connector

```
[root@antipolel kafka_2.13-3.0.0]# export LD_LIBRARY_PATH=/opt/mqm/java/lib64
[root@antipolel kafka_2.13-3.0.0]# bin/connect-standalone.sh config/connect-standalone.properties config/mq-source.properties
[2021-11-09 02:00:56,504] INFO Kafka Connect standalone worker initializing ... (org.apache.kafka.connect.cli.ConnectStandalone:68)
[2021-11-09 02:00:56,516] INFO WorkerInfo values:
        jvm.args = -Xms256M, -Xmx2G, -XX:+UseG1GC, -XX:MaxGCPauseMillis=20, -XX:InitiatingHeapOccupancyPercent=35, -XX:+ExplicitGCInvokesCo
alse, -Dkafka.logs.dir=/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../logs, -Dlog4j.configuration=file:bin/../config/connect-log4j.propertie
        jvm.spec = Red Hat, Inc., OpenJDK 64-Bit Server VM, 1.8.0_312, 25.312-b07
        jvm.classpath = /root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/activation-1.1.1.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/
ations-0.5.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/commons-cli-1.4.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/
.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/connect-file-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/conne
root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/connect-runtime-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/connect-tran
er/kafka_2.13-3.0.0/bin/../libs/hk2-utils-2.6.1.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jackson-annotations-2.12.3.jar:/root
.13-3.0.0/bin/../libs/jackson-dataformat-csv-2.12.3.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jackson-datatype-jdk8-2.12.3.jar
gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jackson-module-jaxb-annotations-2.12.3.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/
a.annotation-api-1.3.5.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jakarta.inject-2.6.1.jar:/root/gse/kafkadriver/kafka_2.13-3.0
jakarta.xml.bind-api-2.3.2.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/javassist-3.27.0-GA.jar:/root/gse/kafkadriver/kafka_2.13-
-api-2.3.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jersey-client-2.34.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs
ervlet-core-2.34.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jersey-hk2-2.34.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../l
ion-9.4.43.v20210629.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jetty-http-9.4.43.v20210629.jar:/root/gse/kafkadriver/kafka_2.1
n/../libs/jetty-server-9.4.43.v20210629.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jetty-servlet-9.4.43.v20210629.jar:/root/gse
iver/kafka_2.13-3.0.0/bin/../libs/jetty-util-ajax-9.4.43.v20210629.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/jline-3.12.1.jar:
3.0.0/bin/../libs/kafka-clients-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/kafka-log4j-appender-3.0.0.jar:/root/gse/kafka
/libs/kafka-server-common-3.0.0.jar:/root/gse/kafkadriver/kafka_2.13-3.0.0/bin/../libs/kafka-shell-3.0.0.jar:/root/gse/kafkadriver/kafka_2.
```

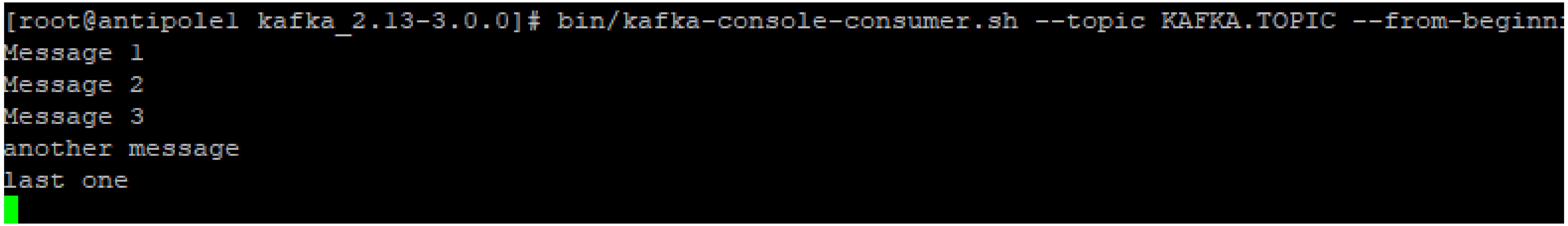
Messages have now arrived

```
[root@antipole1 kafka_2.13-3.0.0]# bin/kafka-console-consumer.sh --topic KAFKA.TOPIC --from-beginning --bootstrap-server localhost:9092
Message 1
Message 2
Message 3
```

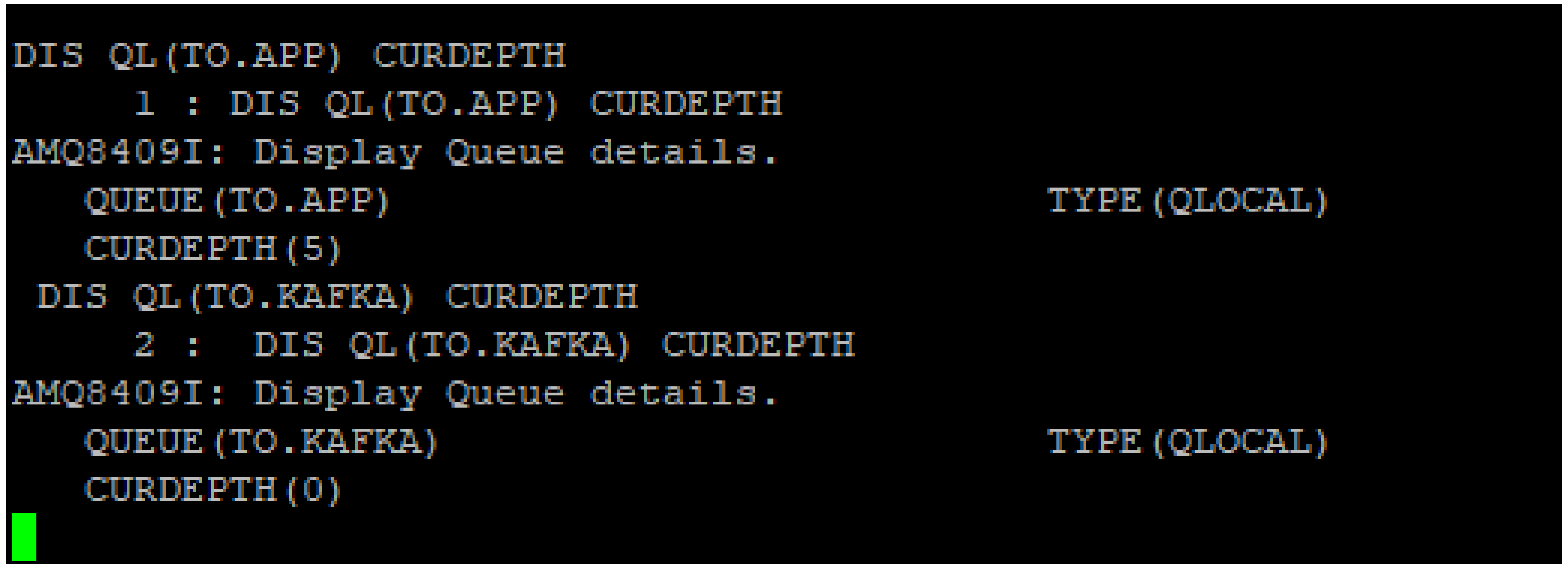
Send some more...



```
[root@antipole1 bin]# ./amqsput TO.APP MQ1
Sample AMQSPUT0 start
target queue is TO.APP
another message
last one
```



```
[root@antipole1 kafka_2.13-3.0.0]# bin/kafka-console-consumer.sh --topic KAFKA.TOPIC --from-beginn
Message 1
Message 2
Message 3
another message
last one
```



```
DIS QL(TO.APP) CURDEPTH
     1 : DIS QL(TO.APP) CURDEPTH
AMQ8409I: Display Queue details.
   QUEUE(TO.APP)                                  TYPE(QLOCAL)
   CURDEPTH(5)
 DIS QL(TO.KAFKA) CURDEPTH
     2 :  DIS QL(TO.KAFKA) CURDEPTH
AMQ8409I: Display Queue details.
   QUEUE(TO.KAFKA)                                TYPE(QLOCAL)
   CURDEPTH(0)
```

Thank you

Dorothy Quincy
Client Technical Specialist

Email: dorothy.quincy@ibm.com

More resources

[Kafka fundamentals - IBM Developer]

[Kafka Connectors for IBM MQ – a MQ for zOS perspective – Integration]

[How IBM MQ on z/OS works together with Kafka – YouTube]

Your feedback is important!

Submit a session evaluation for each session you attend:

www.share.org/evaluation

46