Event Streams using Apache Kafka
And how it relates to IBM MQ

Andrew Schofield
Chief Architect, Event Streams
STSM, IBM Messaging, Hursley Park
Event-driven systems deliver more engaging customer experiences

Phone company has existing data around customers’ usage and buying preferences.

Combined with events generated when phones connect to in-store wi-fi.

Enables a more engaging and personal customer experience.
But being event-driven is different

Event driven solutions require **different thinking**

Events form the **nervous system** of the digital business

Application infrastructure needs to provide event stream processing capabilities and **support emerging event-driven** programming models

This is an event-driven journey and will **underpin the next generation** of digital customer experiences

Source: Gartner - From data-centric to event-centric IT priority
How does this differ from “messaging”?
Apache Kafka is an Open-Source Streaming Platform

Use cases
- Pub/sub messaging
- Website activity tracking
- Metrics
- Log aggregation
- Stream processing
- Event sourcing
- Commit log
Kafka is built for scale

Topic

Partition 0

0 1 2 3 4 5
Kafka is built for scale
Kafka is built for scale

<table>
<thead>
<tr>
<th>Topic</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Partition 1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition 2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Kafka is built for scale
Kafka is built for scale
Kafka is built for scale
Kafka is built for scale
Kafka is built for availability
Kafka is built for availability
Topics are partitioned for scale

- **Partition 0**: 0 1 2 3 4 5 6 7
- **Partition 1**: 0 1 2 3
- **Partition 2**: 0 1 2 3 4 5

**Topic** → **Writes** → **New**

**Old**
Each partition is an immutable sequence of records

- Producer writes
- Consumer A (offset 6)
- Consumer B (offset 9)
Consumer groups share the partitions

```
Consumer Group A
  Consumer
    p0, offset 7
  Consumer
    p1, offset 3
  Consumer
    p2, offset 4
```

```
Consumer Group B
  Consumer
    p0, offset 6
  Consumer
    p1, offset 1
  Consumer
    p2, offset 5
```
# Reliability

**Producer can choose acknowledgement level**

<table>
<thead>
<tr>
<th>Value</th>
<th>Acknowledgement Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Fire-and-forget, fast but risky</td>
</tr>
<tr>
<td>1</td>
<td>Waits for 1 broker to acknowledge</td>
</tr>
<tr>
<td>All</td>
<td>Waits for all replica brokers to acknowledge</td>
</tr>
</tbody>
</table>

**Producer can choose whether to retry**

<table>
<thead>
<tr>
<th>Value</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not retry, lose messages on error</td>
</tr>
<tr>
<td>&gt;0</td>
<td>Retry, might result in duplicates on error</td>
</tr>
</tbody>
</table>

**Producer can also choose idempotence**

Means that retries can be used without risk of duplicates

**Consumer can choose how to commit offsets**

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
</tr>
<tr>
<td>Commits might go faster than processing</td>
</tr>
<tr>
<td>Manual, asynchronous</td>
</tr>
<tr>
<td>Fairly safe, but could re-process messages</td>
</tr>
<tr>
<td>Manual, synchronous</td>
</tr>
<tr>
<td>Safe, but slows down processing</td>
</tr>
</tbody>
</table>

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
Compacted topics are evolving data stores

Source change log

Producer

Old

New

Consumer takes latest values and builds own data store

Key | Value
--- | ---
a | A2
b | <deleted>
c | C2
Compacted topics are evolving data stores

Periodic compaction eliminates duplicate keys to minimize storage

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>A2</td>
</tr>
<tr>
<td>b</td>
<td>&lt;deleted&gt;</td>
</tr>
<tr>
<td>c</td>
<td>C2</td>
</tr>
</tbody>
</table>
Kafka Connect

Over 80 connectors
- HDFS
- Elasticsearch
- MySQL
- JDBC
- IBM MQ
- MQTT
- CoAP
- + many others

https://www.confluent.io/product/connectors/
It’s easy to connect IBM MQ to Apache Kafka

IBM has created two open-source connectors available on GitHub

**Source Connector**
From MQ queue to Kafka topic
https://github.com/ibm-messaging/kafka-connect-mq-source

**Sink Connector**
From Kafka topic to MQ queue
https://github.com/ibm-messaging/kafka-connect-mq-sink

Detailed instructions for running them:
Kafka Connect source connector for IBM MQ

Open-source – build it yourself
Use any supported MQ release
Uses JMS client internally
Client connections
Supports TLS, authentication
MQ queue->Kafka topic
Support for binary, text, JSON
Easy to extend

https://github.com/ibm-messaging/kafka-connect-mq-source
Configuration of MQ Source connector

Configuration is provided in a properties file

Required:
- `mq.queue.manager` – MQ QMgr name
- `mq.connection.name.list` – MQ client conname
- `mq.channel.name` – MQ svrconn channel name
- `mq.queue` – MQ source queue
- `topic` – Kafka target topic

Optional:
- `mq.user.name` – MQ user name for client
- `mq.password` – MQ password for client
- `mq.message.body.jms` – native MQ or JMS
- `mq.ssl.cipher.suite` – MQ SSL cipher suite
- `mq.ssl.peer.name` – MQ SSL peer name

Sample file provided in GitHub

Conversion parameters:
- `mq.record.builder`
- `value.converter`

Example:
```yaml
name=mq-source
connector.class=com.ibm.mq.kafkaconnect.MQSourceConnector
tasks.max=1
mq.queue.manager=MYQM
mq.connection.name.list=localhost:1414
mq.channel.name=MYSVRCONN
mq.queue=TO.KAFKA.Q
topic=FROM.MQ.TOPIC
mq.user.name=alice
mq.password=passw0rd
mq.record.builder=com.ibm.mq.kafkaconnect.builders.DefaultRecordBuilder
value.converter=org.apache.kafka.connect.converters.ByteArrayConverter
```
Kafka Connect sink connector for IBM MQ

Open-source – build it yourself
Use any supported MQ release
Uses JMS client internally
Client connections
Supports TLS, authentication
Kafka topic -> MQ queue
Support for binary, text, JSON
Easy to extend

https://github.com/ibm-messaging/kafka-connect-mq-sink
Configuration of MQ Sink connector

Configuration is provided in a properties file

Required:
- topics – Kafka source topic list
- mq.queue.manager – MQ QMgr name
- mq.connection.name.list – MQ client conname
- mq.channel.name – MQ svrconn channel name
- mq.queue – MQ target queue

Optional:
- mq.user.name – MQ user name for client
- mq.password – MQ password for client
- mq.message.body.jms – native MQ or JMS
- mq.ssl.cipher.suite – MQ SSL cipher suite
- mq.ssl.peer.name – MQ SSL peer name
- mq.time.to.live – MQ message expiration
- mq.persistent – MQ message persistence

Sample file provided in GitHub

Conversion parameters:
- mq.message.builder
- value.converter

Example:
name=mq-sink
connector.class=com.ibm.mq.kafkaconnect.MQSinkConnector
tasks.max=1
topics=TO.MQ.TOPIC
mq.queue.manager=MYQM
mq.connection.name.list=localhost:1414
mq.channel.name=MYSVRCONN
mq.queue=FROM.KAFKA.Q
mq.user.name=alice
mq.password=passw0rd
mq.message.builder=com.ibm.mq.kafkaconnect.builders.DefaultMessageBuilder
value.converter=org.apache.kafka.connect.converters.ByteArrayConverter
Ubuntu 14.04 virtual machine

MQ Source connector

RecordBuilder

MQ Message
MQMD
(MQRFH2)
Payload

Converter

SourceRecord
Schema
Value (may be complex)

Kafka Record
Null key
Value byte[]

TSOURCE

Kafka Record

DEMO
Introducing IBM Event Streams

React to events in real-time to deliver more engaging experiences for your customers

Deploy production-ready Apache Kafka onto IBM Cloud Private in minutes

Rely on disaster recovery & security designed for mission-critical use

Build intelligent apps on Kafka with the confidence IBM is supporting you

Exploit existing data to become a real Event-Driven Enterprise
IBM Event Streams builds on the open standards of IBM Cloud Private

- **Containers**: Executable package of software that includes everything needed to run it
- **Orchestration**: Automate deployment, scaling, and management of containerized applications
- **Management**: Define, install, and upgrade Kubernetes applications
- **Provisioning**: Infrastructure as code to provision public cloud and on-premises environments
Benefit from the Core Services of IBM Cloud Private

**Enterprise Content Catalog**
Open Source and IBM Middleware, Data, Analytics, and AI Software

**Core Operational Services**
Log Management, Monitoring, Metering, Security, Alerting

**Kubernetes Container Orchestration Platform**

Choose your infrastructure:

- Power Systems
- openstack
- IBM Spectrum
- IBM Z
- VMware
- Intel

**Strategic Value:**
Self-service catalog
Agility, scalability, and elasticity
Self-healing
Enterprise security
No vendor lock-in
IBM Event Streams is packaged as a Helm chart

A 3-node Kafka cluster, plus ZooKeeper, UI, network proxies and so on is over 20 containers

Kubernetes and Helm brings this all under control

Install a Kafka cluster with a few clicks from the IBM Cloud Private catalog

It comes
- Highly available
- Secure
- Ready for production
High Availability, Scaling and Configuration with Ease

- Highly available by design
  - Brokers are spread across ICP worker nodes using anti-affinity policies
  - Minimizes the risk of down-time in the event of a node outage

- Scale the Kafka cluster up with one command
  - Safely grows the stateful set, reconfigures the network interfaces and gives you more capacity

- Roll out Kafka cluster configuration changes easily
  - Make a single configuration change and Event Streams rolls it out across the brokers in the cluster
  - Broker availability is managed using health checks to ensure that availability is maintained
Safe, Planned Upgrade of Apache Kafka

Upgrade Kafka versions safely and without hassle

- First, upgrade the Helm chart to a newer version of IBM Event Streams
  - Rolling update of the Kafka brokers minimizes disruption

- As a separate step, upgrade the broker data and protocol version to complete the upgrade
  - Until this point, you can roll back
IBM Event Streams | Making Apache Kafka Intuitive and Easy

Visualisation of your topic data

Simple deploy with just 3 clicks
IBM Event Streams | Making Apache Kafka Intuitive and Easy

Monitor status at a glance

Integrated feedback and support
Security – Authentication and Access Control

- User and group information controlled centrally
  - Integrate with your corporate LDAP through IBM Cloud Private

- Control access to Event Streams resources using role-based access control policies
  - Assign roles to users: Viewer, Editor, Operator, Administrator
  - Optionally, specify access for individual resources, such as topic T

- Application access control using service IDs
  - Same concepts as controlling user access
  - Can restrict application access to exactly the resources they need
  - Prevent accidental or intentional interference between applications

Example policy

**Permit Bob to write to topic T**

<table>
<thead>
<tr>
<th>User</th>
<th>bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role</td>
<td>Editor</td>
</tr>
<tr>
<td>Service</td>
<td>Event Streams instance R</td>
</tr>
<tr>
<td>Type</td>
<td>topic</td>
</tr>
<tr>
<td>Resource</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service action</th>
<th>Roles</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>topic.read</td>
<td>Viewer and above</td>
<td>Read messages or config</td>
</tr>
<tr>
<td>topic.write</td>
<td>Editor and above</td>
<td>Write messages</td>
</tr>
<tr>
<td>topic.manage</td>
<td>Operator and Administrator</td>
<td>Delete or change config</td>
</tr>
</tbody>
</table>
IBM Event Streams | Enterprise-Grade Reliability

Integrated geo-replication for disaster recovery
Geo-Replication makes Disaster Recovery simple

Target is take-over of workload on the destination cluster by business applications within 15 minutes

Easy configuration using the Event Streams UI from the origin cluster sets up the replicator and security credentials

At-least-once reliability so messages are not lost
IBM Event Streams | **Connects to existing MQ backbone**

*Kafka Connect source connector for IBM MQ*

Fully supported by IBM for customers with support entitlement for IBM Event Streams
IBM Event Streams | Ready for Mission-Critical Workloads

All with IBM 24x7 worldwide support

IBM has years of experience running Apache Kafka across the globe