Kafka on OpenShift: make it easy with AMQ Streams
Event Streaming and reactive architectures

Paolo Patierno
Principal Software Engineer @ Red Hat
@ppatierno

#RedHatOSD
Messaging ≠ Messaging ≠ Messaging

- Low-latency pub/sub
- Cross-cloud backbone
- Replayable streams
- Load levelling
- IoT device connectivity
- Enterprise application integration
- Load balancing
- Temporal decoupling
- Geo-aware routing
- Database change data capture
- Message-driven beans
- Event-driven microservices
- Replayable streams
- Long-term message storage
- Event sourcing
AMQ Streams on OpenShift Container Platform

• Enterprise distribution of Apache Kafka
• Simplified deployment on OpenShift
• Based on OSS project called Strimzi
• Provides:
  • Container images for running Apache Kafka and Zookeeper
  • Operators for managing and configuring Apache Kafka clusters, topics and users
What is Apache Kafka?

A publish/subscribe messaging system?

A streaming data platform?

A distributed, horizontally-scalable, fault-tolerant, commit log?
Apache Kafka

Concepts

• Messages are sent to and received from a topic
  • Topics are split into one or more partitions (aka shards)
  • All actual work is done on partition level, topic is just a virtual object

• Each message is written only into a one selected partition
  • Partitioning is usually done based on the message key
  • Message ordering within the partition is fixed

• Retention
  • Based on size / message age
  • Compacted based on message key
Apache Kafka concepts

Topics & partitions

Partition 0
0 1 2 3 4 5 6 7 8 9 1 1

Partition 1
0 1 2 3 4 5 6

Partition 2
0 1 2 3 4 5 6 7 8 9 1 0

Producer

old new

#RedHatOSD
Apache Kafka concepts

Topics & partitions

Partition 0
0 1 2 3 4 5 6 7 8 9 1 1

Partition 1
0 1 2 3 4 5 6

Partition 2
0 1 2 3 4 5 6 7 8 9 1 0

#RedHatOSD
Apache Kafka concepts

High availability

Leaders and followers spread across the cluster
Apache Kafka concepts

High availability

If a broker with leader partition goes down, a new leader partition is elected on different node
Apache Kafka concepts

Reading and writing to leaders
Apache Kafka concepts

**Consumer Groups: partitions assignment**

- **Topic**
  - Partition 0
  - Partition 1
  - Partition 2
  - Partition 3

- **Group 1**
  - Consumer
  - Consumer

- **Group 2**
  - Consumer
  - Consumer
  - Consumer
Apache Kafka concepts

**Consumer Groups: rebalancing**

- **Topic**
  - Partition 0
  - Partition 1
  - Partition 2
  - Partition 3

- **Group 1**
  - Consumer
  - Consumer

- **Group 2**
  - Consumer
  - Consumer
  - Consumer

In the diagram, the partitions are distributed among the consumer groups. When rebalancing occurs, consumers are reassigned to ensure an even distribution of load. The crossed-out consumer in Group 2 indicates that it is being removed or reassigned during rebalancing.
Apache Kafka concepts

**Consumer Groups: max parallelism & idle consumer**

![Diagram of Consumer Groups](image-url)
# AMQ Broker & AMQ Streams

## Key differences

<table>
<thead>
<tr>
<th></th>
<th>AMQ Broker (ActiveMQ Artemis)</th>
<th>AMQ Streams (Kafka)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td>“Smart broker, dumb clients”</td>
<td>“Dumb broker, smart clients”</td>
</tr>
<tr>
<td><strong>Durability</strong></td>
<td>Volatile or durable storage</td>
<td>Durable storage</td>
</tr>
<tr>
<td><strong>Storage duration</strong></td>
<td>Temporary storage of messages</td>
<td>Potential long-term storage of messages</td>
</tr>
<tr>
<td><strong>Message retention</strong></td>
<td>Retained until consumed</td>
<td>Retained until expired or compacted</td>
</tr>
<tr>
<td><strong>Consumer state</strong></td>
<td>Broker managed</td>
<td>Client managed (can be stored in broker)</td>
</tr>
<tr>
<td><strong>Selectors</strong></td>
<td>Yes, per consumer</td>
<td>No</td>
</tr>
<tr>
<td><strong>Stream replay</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>High-availability</strong></td>
<td>Replication</td>
<td>Replication</td>
</tr>
<tr>
<td><strong>Protocols</strong></td>
<td>AMQP, MQTT, OpenWire, Core, STOMP</td>
<td>Kafka protocol</td>
</tr>
<tr>
<td><strong>Delivery guarantees</strong></td>
<td>Best-effort or guaranteed</td>
<td>Best-effort or guaranteed</td>
</tr>
</tbody>
</table>
Apache Kafka is *stateful* which means we require …
○ … a stable broker identity
○ … a way for the brokers to discover each other on the network
○ … durable broker state (i.e., the messages)
○ … the ability to recover broker state after a failure
• All the above are true for Apache Zookeeper as well
• StatefulSets, PersistentVolumeClaims, Services can help but …
It’s not easy!
Simplifying the Apache Kafka deployment on OpenShift

Using the OpenShift native mechanisms for...
  ○ Provisioning the cluster
  ○ Managing the topics and users

... thereby removing the need to use Kafka command-line tools

Providing a better integration with applications running on OpenShift
  ○ microservices, data streaming, event-sourcing, etc.
AMQ Streams on OCP

The “Operator” model

- An application used to create, configure and manage other complex applications
  - Contains specific domain / application knowledge
- Operator works based on input from Config Maps or Custom Resource Definitions
  - User describes the desired state
  - Controller applies this state to the application
- It watches the *desired* state and the *actual* state …
  - ... taking appropriate actions
AMQ Streams on OCP
The Operators

- **Zookeeper**: Deploys & manages cluster
- **Kafka**: Manages topics & users
- **User Operator**: Manages topics & users
- **Topic Operator**: Deploys & manages cluster
- **Cluster Operator**: Deploys & manages cluster
- **User CR**: Deploys & manages cluster
- **Kafka CR**: Manages topics & users
- **Topic CR**: Manages topics & users

#RedHatOSD
AMQ Streams on OCP
Cluster Operator

- Responsible for managing clusters
  - Kafka brokers (including Zookeeper)
  - Kafka Connect clusters
  - Kafka Mirror Maker
- Responsible for
  - Deployment
  - Scale-up / Scale-down
  - Re-configuration
AMQ Streams on OCP

Topic Operator

- Responsible for managing Kafka topics
  - You can create, update and delete topics “the Kubernetes way”
  - No need to know Kafka commands
  - Applications can still create topics directly in Kafka
    - Topic operator synchronizes the topics bi-directionally
    - For topics created in Kafka, it will create Custom Resources
    - In case of conflicts, it will use 3-way-diff to resolve them
AMQ Streams on OCP

User Operator

- Responsible for managing users
  - Allows to create, update and delete users
  - Currently two supported authentication mechanisms
    - TLS client certificates
    - SASL SCRAM-SHA-512 (username and password based authentication)
  - Authorization manages using Kafka ACL plugin
    - Allowed / Denied operations can be defined together with the user
AMQ Streams on OCP

Main features

- Mirroring
- Storage
- Encryption
- Metrics
- Scale Up
- Source2Image
- Tolerations
- High Availability
- Authentication
- Logging
- Authorization
- JVM Configuration
- Memory and CPU resources
- Affinity
- Scale Down
- Healthchecks
- Configuration
AMQ Streams on OCP Operator

- OCP 3.11 provides a few operators
  - Prometheus, etcd, ...
- AMQ Streams 1.0 available
AMQ Streams on OpenShift Container Platform is GA!

#RedHatOSD
DEMO TIME
Resources

- AMQ Streams: https://access.redhat.com/products/red-hat-amq-streams
- Strimzi: http://strimzi.io/ - @strimziio
- Apache Kafka: https://kafka.apache.org/
- Demo: https://github.com/ppatierno/rh-osd-2018
GRAZIE PER L’ATTENZIONE

Paolo Patierno
Principal Software Engineer @ Red Hat
@ppatierno