RED HAT SUMMIT

A-MQ 7 Sneak Preview

David Ingham
Senior Engineering Manager

June 30, 2016

Red Hat JBoss A-MQ 7

Standard protocols

Common tooling

Flexible, standards-based messaging for the enterprise, cloud and Internet of Things

Broker

- New broker core with modern async architecture
- Improved performance and scalability

Interconnect

- Message router
- High-performance direct messaging
- Distributed messaging backbone

Clients

- New set of AMQP 1.0 clients
- Support for A-MQ 6.x and HornetQ clients for backward compatibility

Standard Protocols

Advanced Message Queuing Protocol
AMQP v1.0

Open, standard messaging protocol

General-purpose message transfer protocol that enables cross-platform apps to be built using brokers, libraries and frameworks from different vendors

International Standard ISO/IEC 19464

ISO/IEC 19464:2014

Features

Efficient – binary, connection-oriented protocol

Reliable – range of guarantees, fire-&-forget to reliable, exactly-once delivery

Portable data representation – cross-platform, full-fidelity message exchange

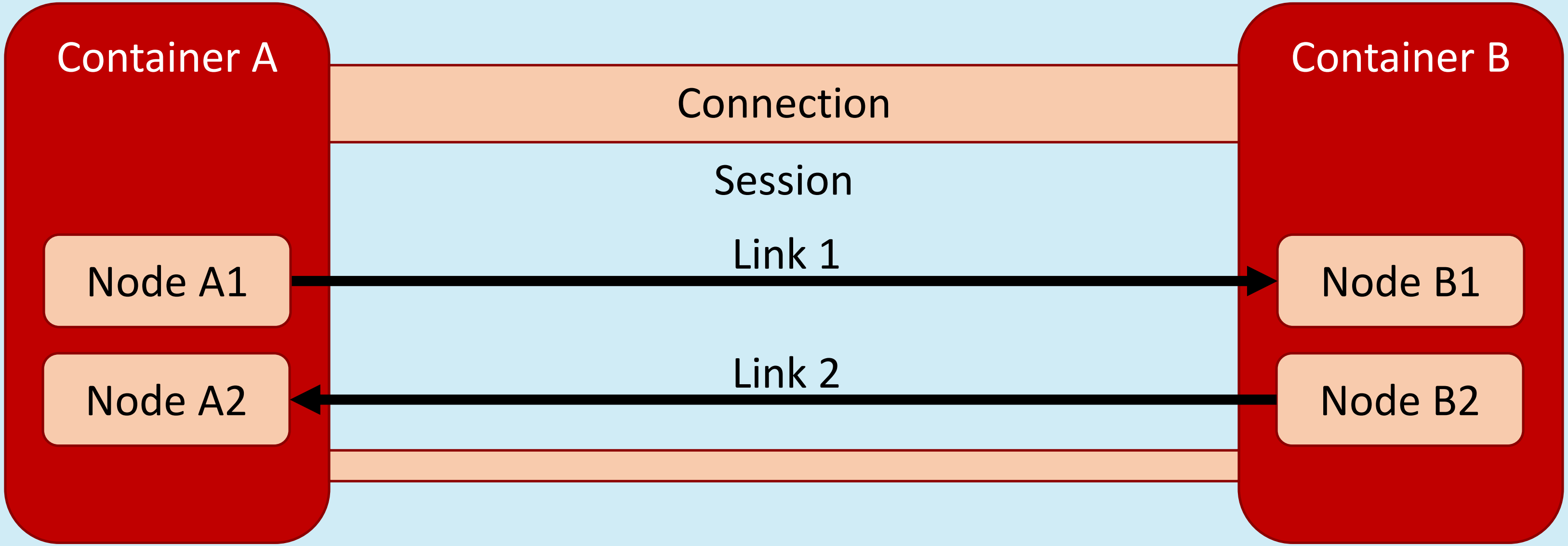
Sophisticated flow control – credit-based flow control & channel multiplexing

Flexible – client-client, client-broker & broker-broker topologies

Broker-model independent – no requirements on broker internals

http://www.amqp.org

AMQP Concepts



- Containers (clients, brokers, …) connect to other containers
- Connections manage transfer capacity (frame size, channel count, …)
- Bidirectional Sessions created over Connection. Byte-based flow control.
- Links are created over Sessions for unidirectional message transfer. Credit-based flow control.

Message Queuing Telemetry Transport
MQTT v3.1.1

Open, standard messaging protocol

Lightweight publish/subscribe messaging transport for the Internet of Things

International Standard ISO/IEC 20922

ISO/IEC 20922:2016

Features

Efficient – binary, connection-oriented protocol

Reliable – range of guarantees, fire-&-forget to reliable, exactly-once delivery

Lightweight – minimal overhead of business payload

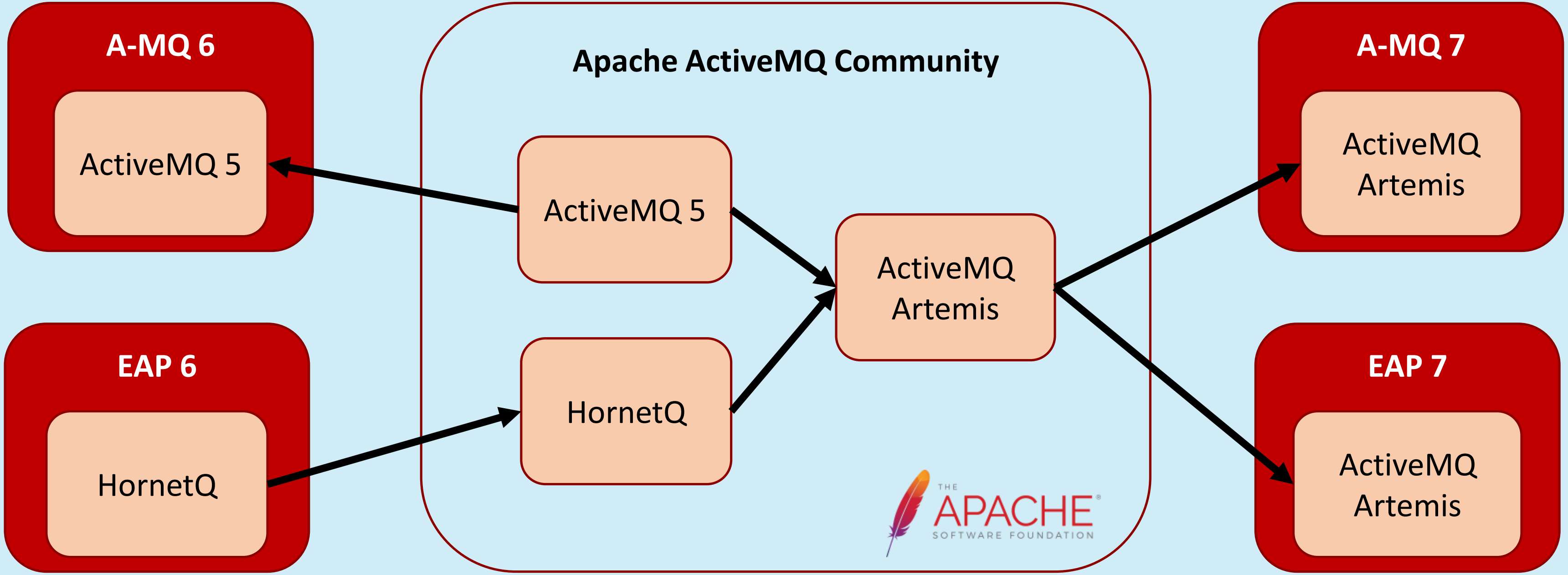
Simple message format – messages are simple arrays of bytes

Pub/sub model – defines hierarchical pub/sub broker model

http://www.mqtt.org

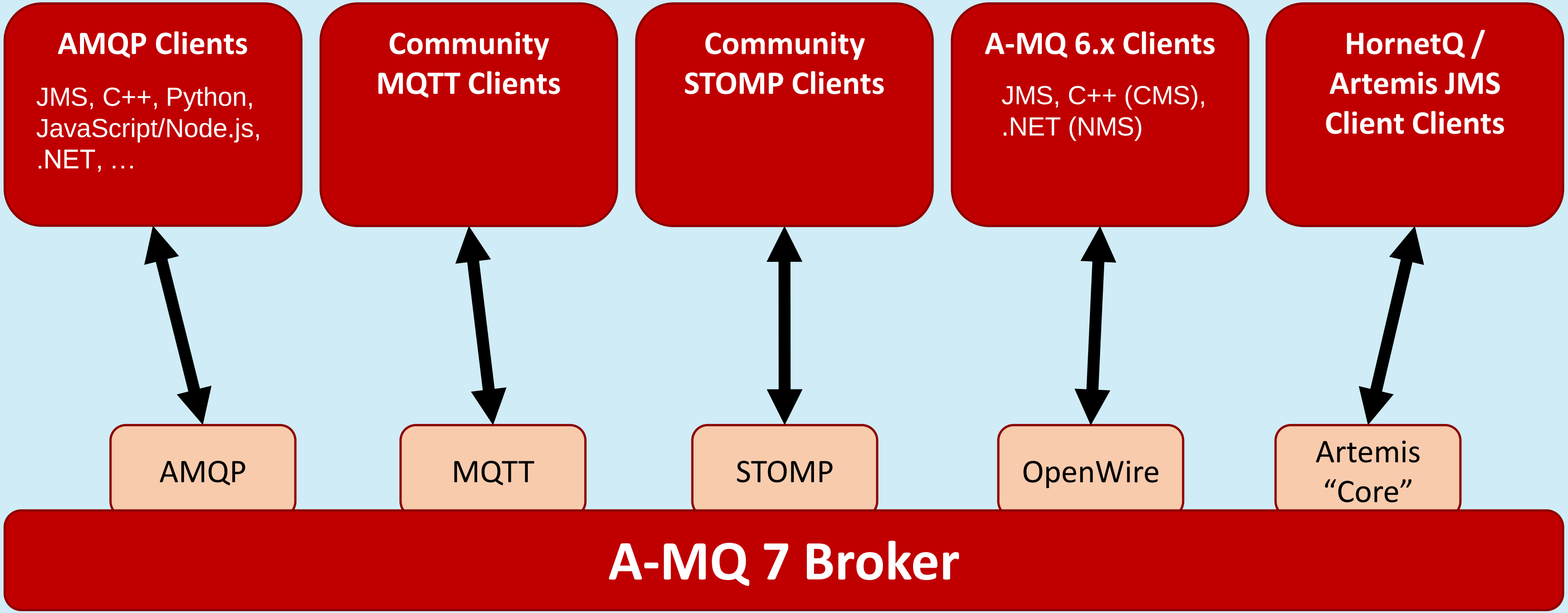
A-MQ 7 Broker

Broker consolidation



redhat

Protocols and clients



AMQP Clients

JMS, C++, Python, JavaScript/Node.js, .NET, …

Community MQTT Clients

Community STOMP Clients

A-MQ 6.x Clients

JMS, C++ (CMS), .NET (NMS)

HornetQ / Artemis JMS Client Clients

AMQP

MQTT

STOMP

OpenWire

Artemis “Core”

A-MQ 7 Broker

A-MQ 7 Broker vs. EAP 7 Broker

Feature	A-MQ 7	EAP 7
High performance broker based on ActiveMQ Artemis	✓	✓
JMS 2.0 client library using "Core" protocol	✓	✓
AMQP 1.0 protocol support	✓	✗
MQTT protocol support	✓	✗
OpenWire protocol support	✓	✗

A-MQ 7 Broker: headline features

- Pure Java, high-performance message broker
- Multi-protocol: AMQP 1.0, MQTT, STOMP, OpenWire, HornetQ Core
- Polyglot: Java JMS, C++, .NET, Python, JavaScript (inc. Node.js)
- Flexible persistence: high performance journal or JDBC
- Support for large messages
- Flexible clustering
- High availability: shared SAN or shared nothing

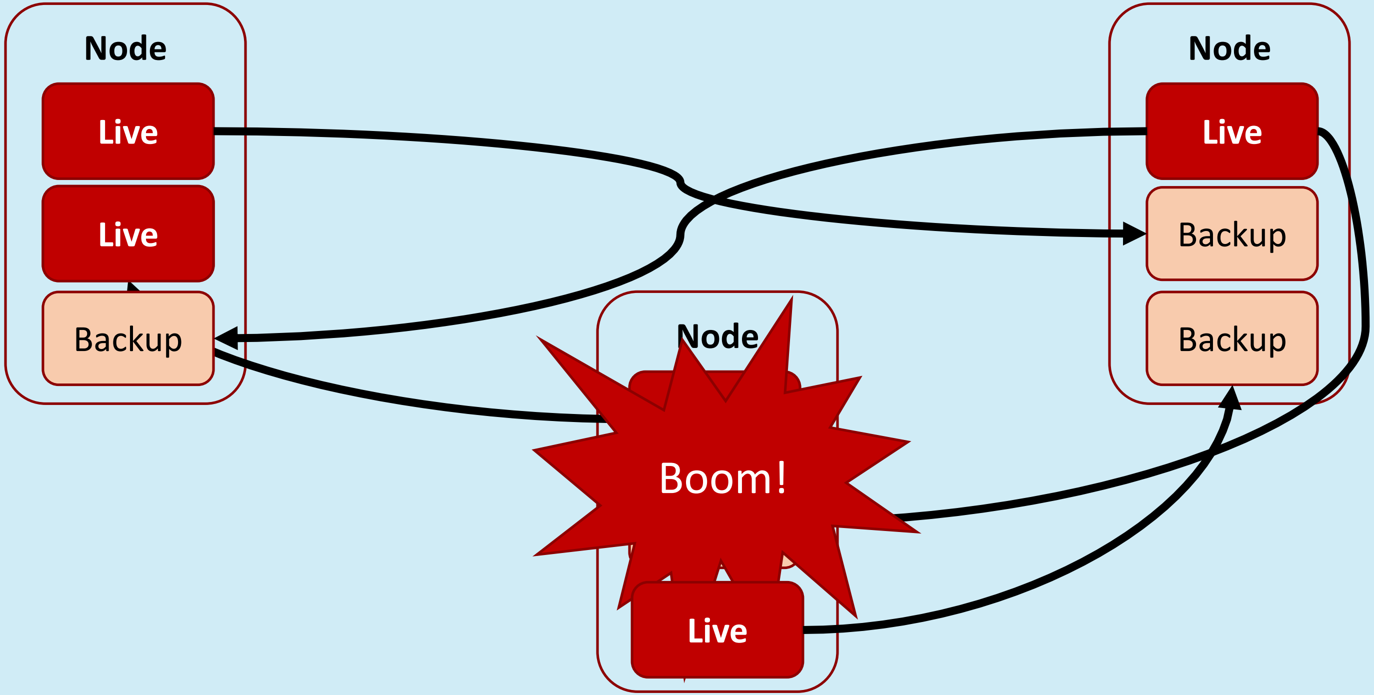
A-MQ 7 Broker: internal architecture

- Fully asynchronous (non-blocking) internal architecture
  - Developed using reactive patterns
  - Netty IO
- Thread pooling
  - Predictable thread usage (cf. 1 thread per client/queue)
  - Configurable thread pools
- High performance journal
  - Custom implementation using Linux asynchronous I/O (JNI to libaio)
  - Automatically switches to Java NIO implementation when not on Linux
  - Compaction

A-MQ 7 Broker: clustering

- Goal: balance message processing across several broker nodes
- Brokers can be grouped into clusters
  - Explicit configuration, discovery using UDP multicast, JGroups, …
- Messages arriving on cluster are balanced to different nodes to spread the load – default balancing is round robin
- Balancing takes into account selectors and consumers on each node

A-MQ 7 Broker: high availability

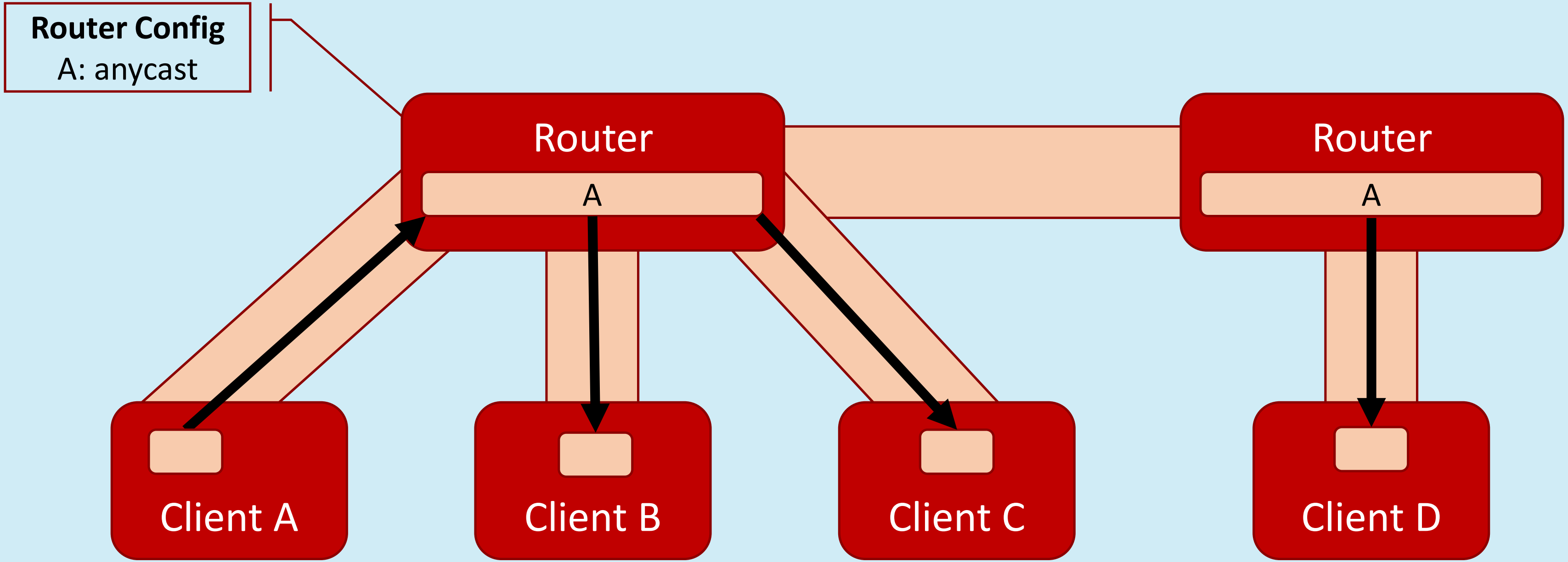


A-MQ 7 Interconnect

A-MQ 7 Interconnect

- AMQP-native message router
- Influenced by Red Hat MRG Messaging use cases
- Supports high performance direct messaging
- Best effort or end-to-end guaranteed delivery
- 1-to-1 or 1-to-many
- Can be used a single instance or connected as a network
- Network offers shortest-path routing with redundancy
- Can be used standalone or in conjunction with broker

A-MQ 7 Interconnect

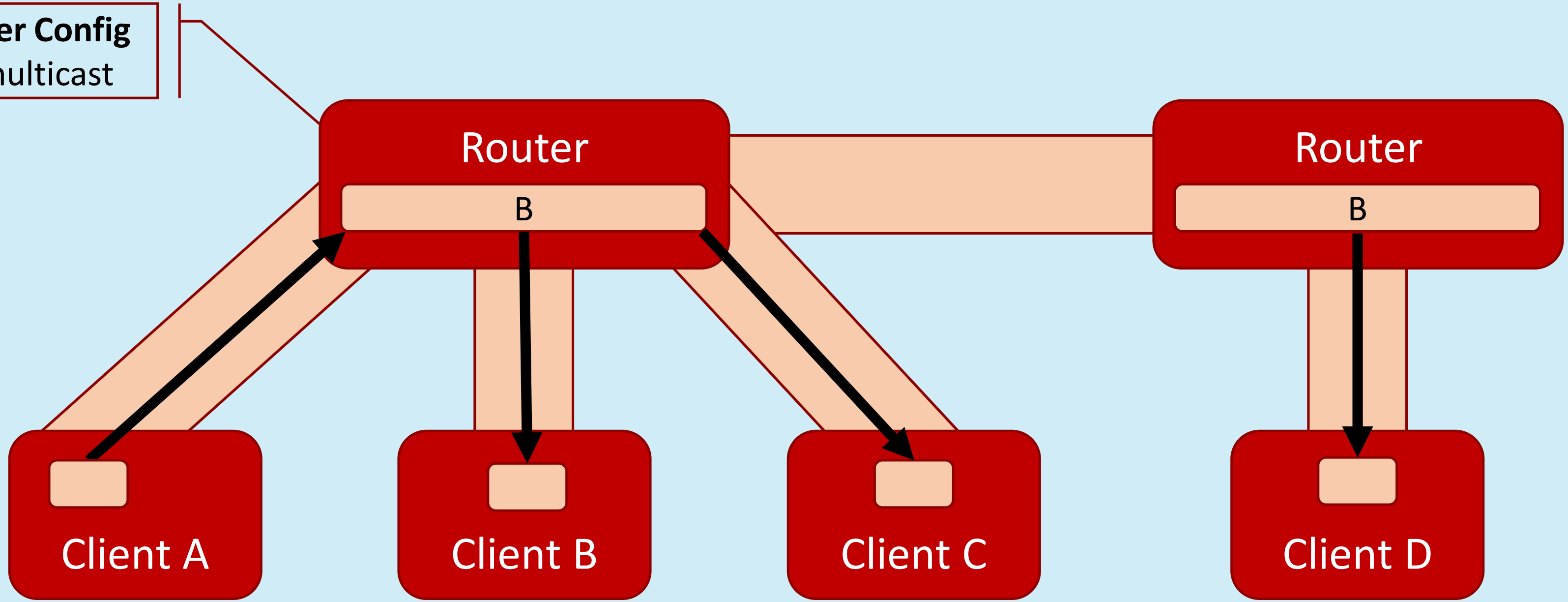


Router Config

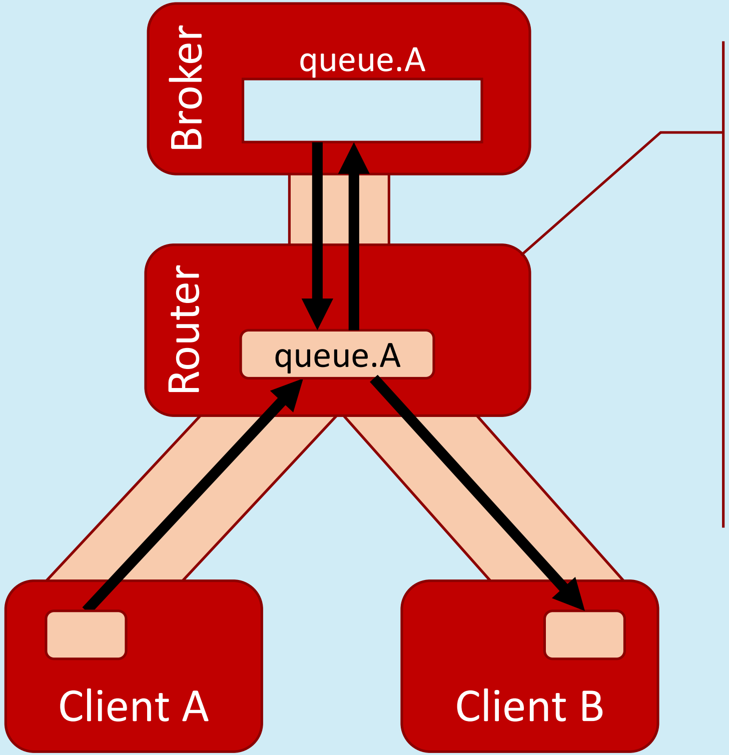
A: anycast

A-MQ 7 Interconnect

Router Config
B: multicast



A-MQ 7 Interconnect + Broker



Router Config

```
address:
   prefix: queue, waypoint = true
connector:
    host: bhost, port: amqp, name: broker1
autoLink
    addr: queue.A, connector: broker1, dir: in
autoLink
    addr: queue.A, connector: broker1, dir:out
```

Comparing broker vs. router

	A-MQ 7 Broker	A-MQ 7 Interconnect Router
Role	Intermediary between clients	Intermediary between clients
Protocol	Supports AMQP (and others)	Supports AMQP
Patterns	1:1 (queue) & 1:many (topic)	1:1 (anycast) & 1:many (multicast)
Delivery guarantees	Best-effort or guaranteed delivery	Best-effort or guaranteed delivery
Ownership	Takes ownership of messages	Does not take ownership of messages
Contracts	2: sender-broker & broker-receiver	1: sender-receiver
High availability	Provided by broker clustering	Provided by redundant topology

A-MQ 7 Clients

AMQP 1.0 client libraries

for A-MQ 7 Broker and Interconnect

- Java JMS 1.1 client (Apache Qpid JMS based on Qpid Proton)
- Reactive C++ client (Apache Qpid Proton)
- Reactive Python client (Apache Qpid Proton)
- Reactive pure JavaScript client w support for Node.js (GitHub Rhea)
- Fully-featured .NET library (GitHub AMQP .NET Lite)

Backward compatibility client libraries

for A-MQ 7 Broker

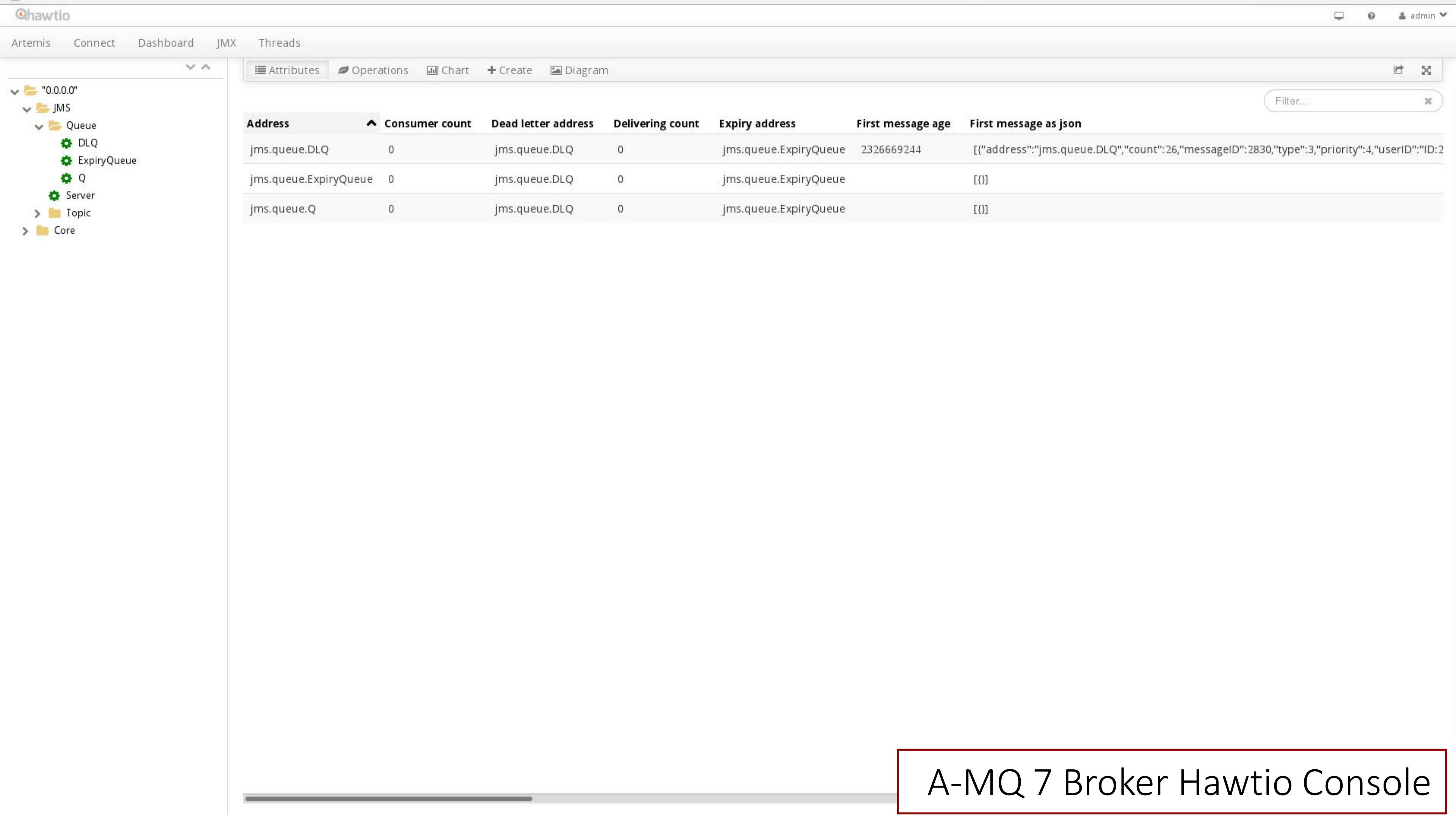
A-MQ 6.x (OpenWire)

- Java JMS 1.1 client
- C++ CMS client (ActiveMQ-CPP)
- .NET NMS client (ActiveMQ-NMS)

HornetQ (Artemis ‘Core’ protocol)

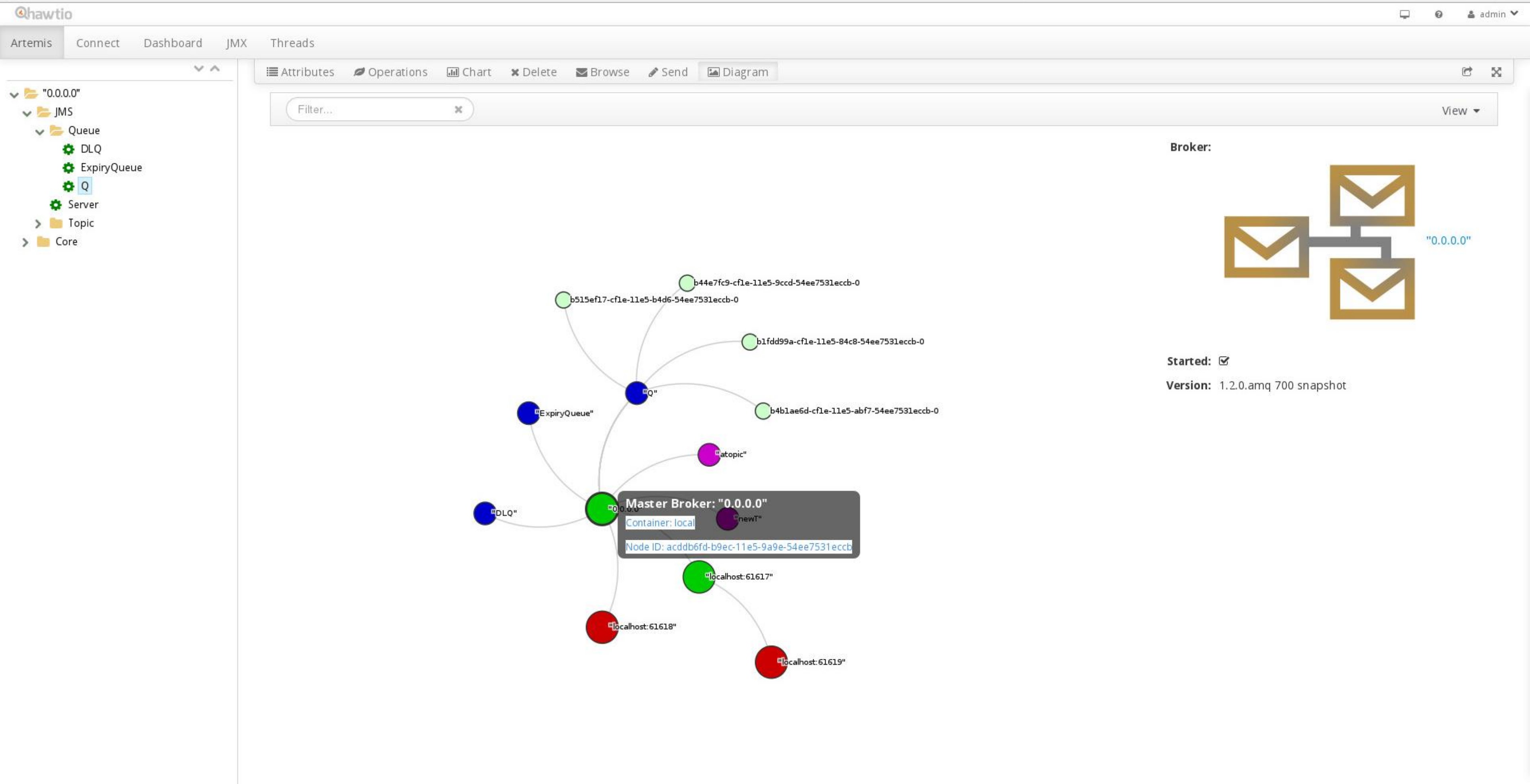
- Java JMS 2.0 client

Common Tooling



Address	Consumer count	Dead letter address	Delivering count	Expiry address	First message age	First message as json
jms.queue.DLQ	0	jms.queue.DLQ	0	jms.queue.ExpiryQueue	2326669244	[{"address":"jms.queue.DLQ","count":26,"messageID":2830,"type":3,"priority":4,"userID":"ID:2
jms.queue.ExpiryQueue	0	jms.queue.DLQ	0	jms.queue.ExpiryQueue		[{}]
jms.queue.Q	0	jms.queue.DLQ	0	jms.queue.ExpiryQueue		[{}]

A-MQ 7 Broker Hawtio Console



Master Broker: "0.0.0.0"

Container: local

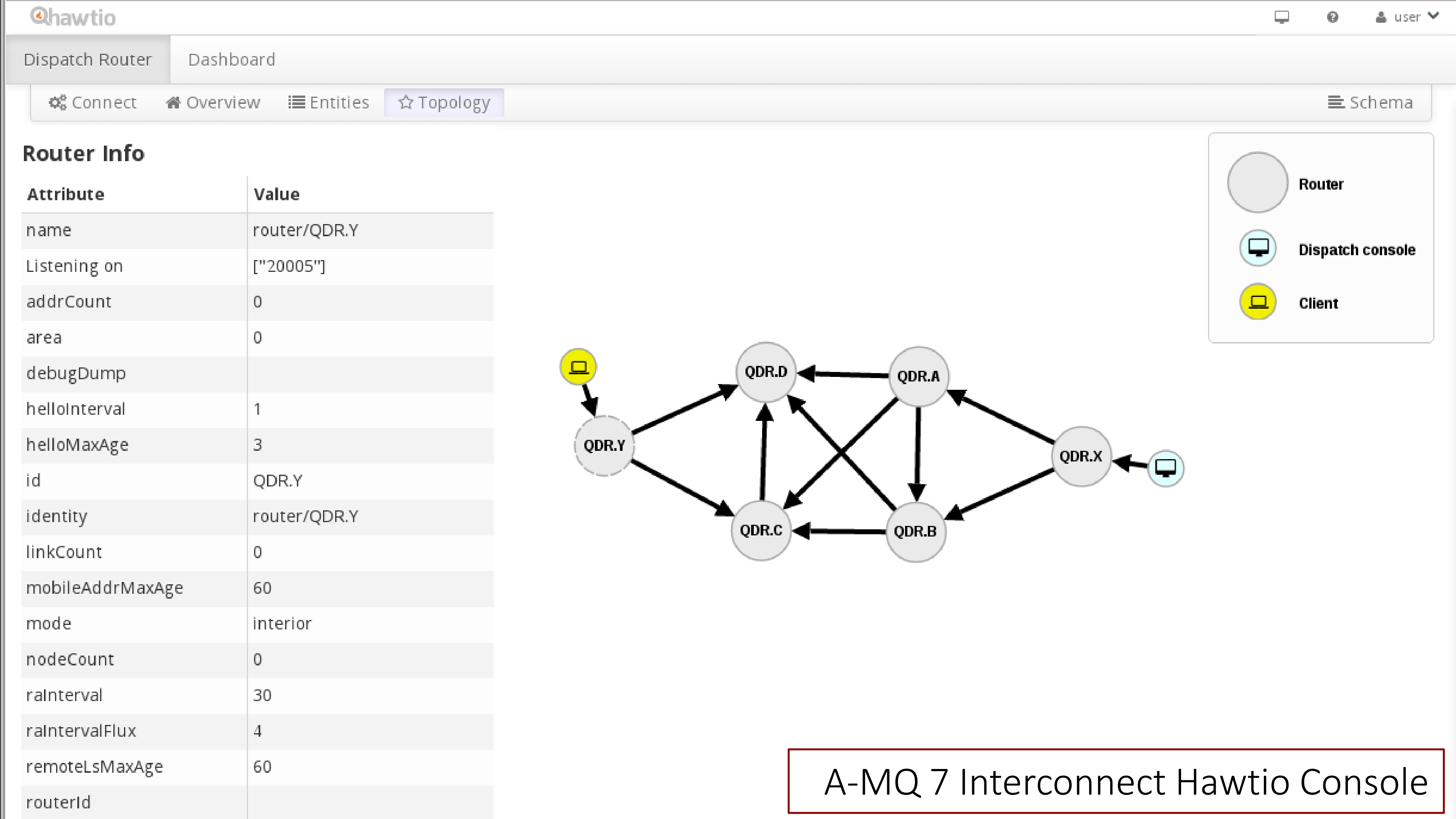
Node ID: acddb6fd-b9ec-11e5-9a9e-54ee7531eccb

Broker:

Started: ☑

Version: 1.2.0.amq 700 snapshot

A-MQ 7 Broker Hawtio Console



Router Info

Attribute	Value
name	router/QDR.Y
Listening on	["20005"]
addrCount	0
area	0
debugDump	
helloInterval	1
helloMaxAge	3
id	QDR.Y
identity	router/QDR.Y
linkCount	0
mobileAddrMaxAge	60
mode	interior
nodeCount	0
raInterval	30
raIntervalFlux	4
remoteLsMaxAge	60
routerId	

A-MQ 7 Interconnect Hawtio Console

Messaging-as-a-Service

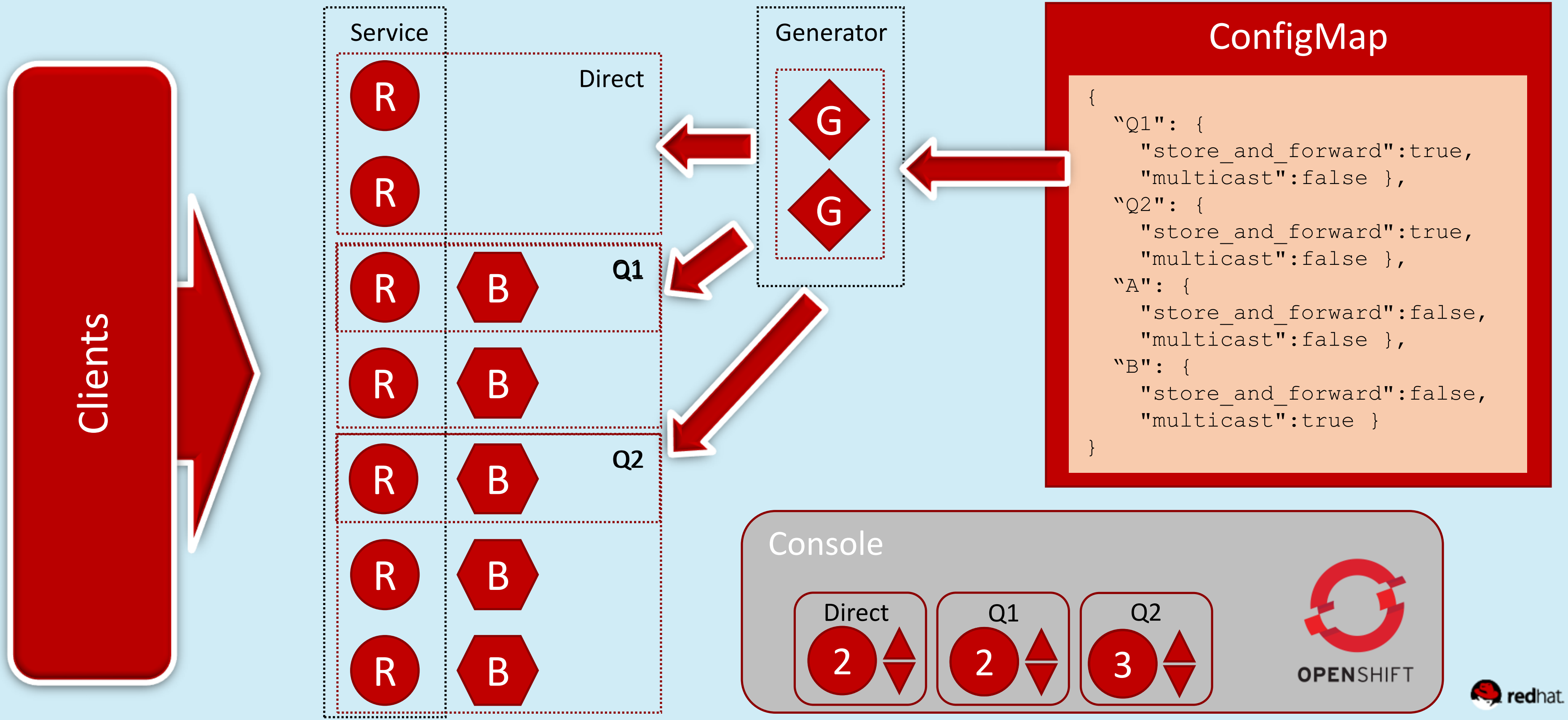
Messaging-as-a-Service (MaaS)

- Elastic-scale messaging utility built using A-MQ 7 on OpenShift
- Separation of concerns
  - Application provisioning and monitoring vs. infrastructure administration
- Multi-tenancy
  - Different namespaces
  - Permissions, quotas and limits
- Support a variety of communication patterns
  - Request-response (point-to-point & service pool)
  - Pub-sub, events
  - Store-and-forward
- Support a variety of different protocols
  - AMQP, MQTT, HTTP(1.1& 2)/CoAP, STOMP

Messaging-as-a-Service (MaaS)

- Handle scale in various dimensions
  - Number of communicating endpoints (i.e. connections)
  - Aggregate message volume
  - Number of distinct addresses (or paths)
  - Number of producers and consumers on given address
  - Examples:
    - Inflow of large volume of sensor events for processing
    - Making requests on a large number of connected devices
    - Broadcasting events to a large number of subscribers
- Elasticity
  - Add and remove capacity without disrupting communication

A-MQ 7 on OpenShift

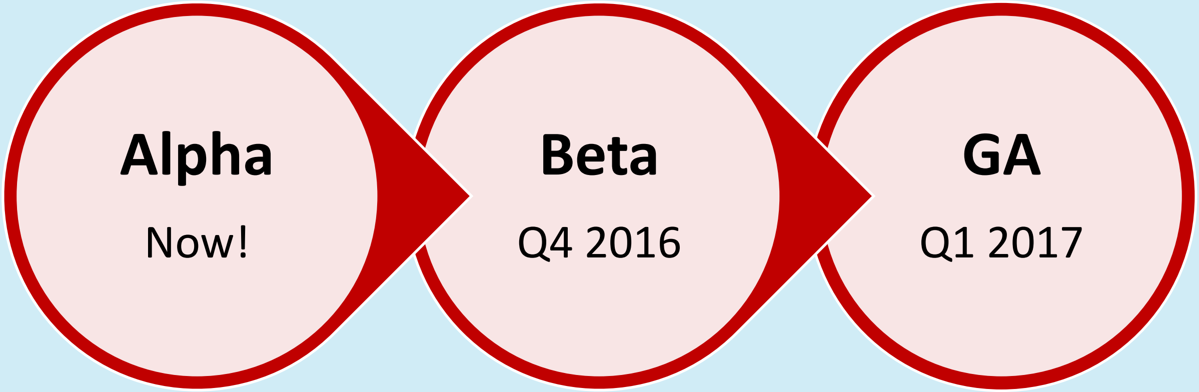


```
{
  “Q1": {
    "store_and_forward":true,
    "multicast":false },
  “Q2": {
    "store_and_forward":true,
    "multicast":false },
  “A": {
    "store_and_forward":false,
    "multicast":false },
  “B": {
    "store_and_forward":false,
    "multicast":true }
}
```

Timeline & Summary

A-MQ 7 Timeline

TENTATIVE



Alpha

Now!

Beta

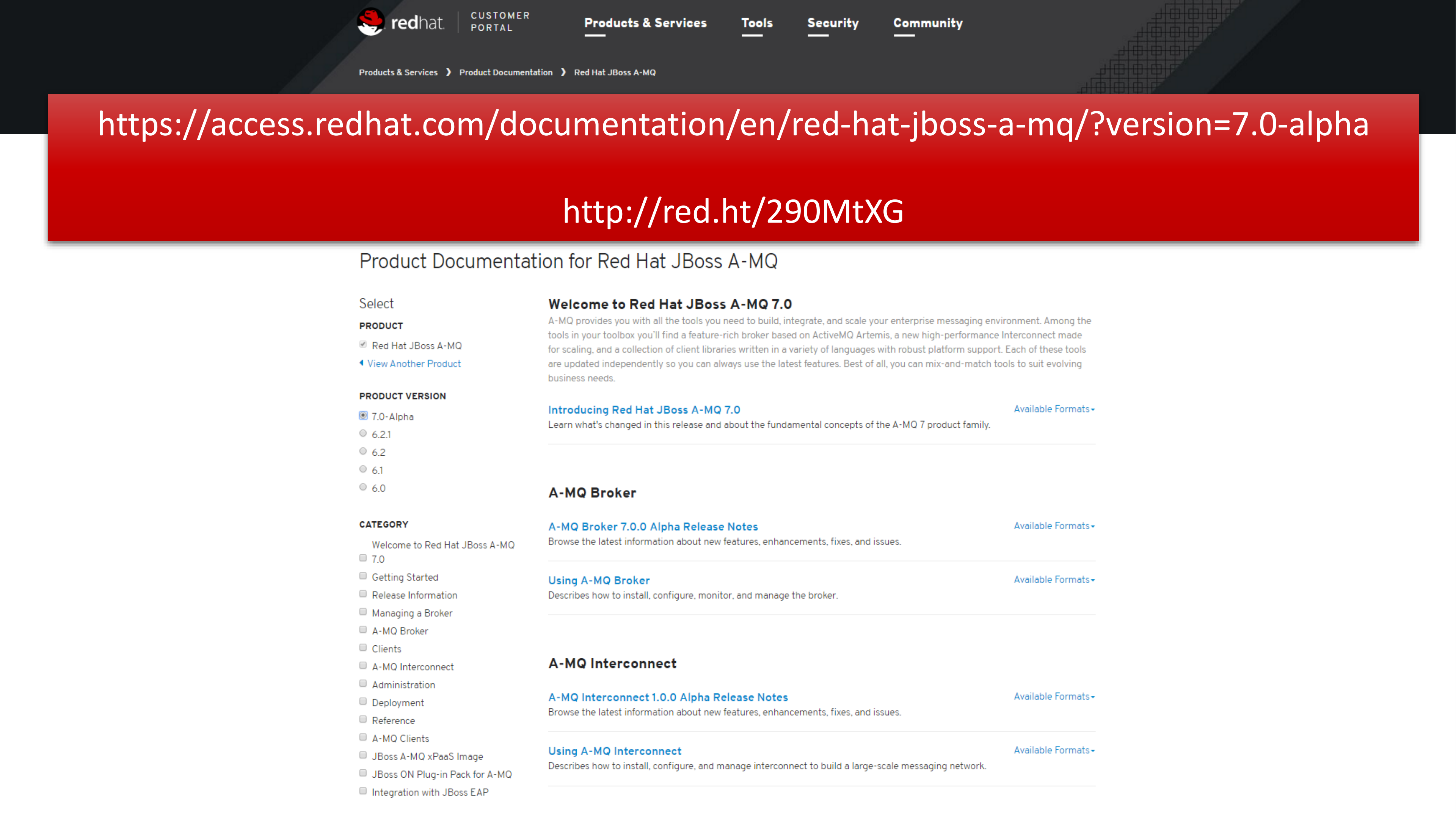
Q4 2016

GA

Q1 2017

https://access.redhat.com/documentation/en/red-hat-jboss-a-mq/?version=7.0-alpha

http://red.ht/290MtXG



redhat. CUSTOMER PORTAL
Products & Services
Tools
Security
Community
Products & Services ❯ Product Documentation ❯ Red Hat JBoss A-MQ

Product Documentation for Red Hat JBoss A-MQ

Select

PRODUCT

☑ Red Hat JBoss A-MQ

◂ View Another Product

PRODUCT VERSION

● 7.0-Alpha
○ 6.2.1
○ 6.2
○ 6.1
○ 6.0

CATEGORY

□ Welcome to Red Hat JBoss A-MQ 7.0
□ Getting Started
□ Release Information
□ Managing a Broker
□ A-MQ Broker
□ Clients
□ A-MQ Interconnect
□ Administration
□ Deployment
□ Reference
□ A-MQ Clients
□ JBoss A-MQ xPaaS Image
□ JBoss ON Plug-in Pack for A-MQ
□ Integration with JBoss EAP

Welcome to Red Hat JBoss A-MQ 7.0

A-MQ provides you with all the tools you need to build, integrate, and scale your enterprise messaging environment. Among the tools in your toolbox you'll find a feature-rich broker based on ActiveMQ Artemis, a new high-performance Interconnect made for scaling, and a collection of client libraries written in a variety of languages with robust platform support. Each of these tools are updated independently so you can always use the latest features. Best of all, you can mix-and-match tools to suit evolving business needs.

Introducing Red Hat JBoss A-MQ 7.0 Available Formats▾
Learn what's changed in this release and about the fundamental concepts of the A-MQ 7 product family.

A-MQ Broker

A-MQ Broker 7.0.0 Alpha Release Notes Available Formats▾
Browse the latest information about new features, enhancements, fixes, and issues.

Using A-MQ Broker Available Formats▾
Describes how to install, configure, monitor, and manage the broker.

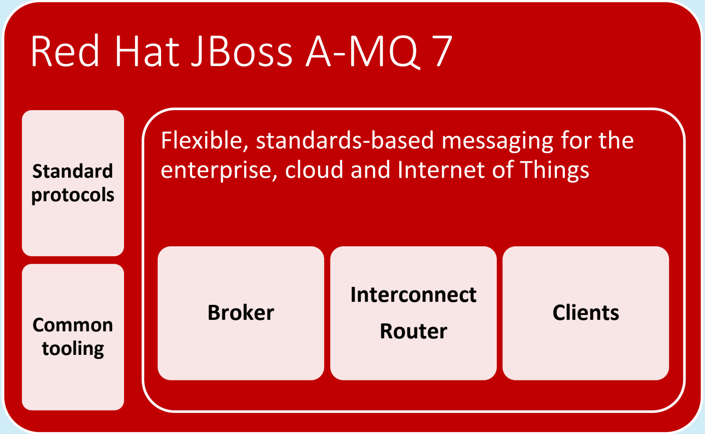
A-MQ Interconnect

A-MQ Interconnect 1.0.0 Alpha Release Notes Available Formats▾
Browse the latest information about new features, enhancements, fixes, and issues.

Using A-MQ Interconnect Available Formats▾
Describes how to install, configure, and manage interconnect to build a large-scale messaging network.

Summary

- A-MQ 7 is the upcoming next major release of the popular messaging platform
  - Improved broker performance and scalability
  - New Interconnect router
  - New AMQP 1.0 clients



Red Hat JBoss A-MQ 7

Standard protocols

Common tooling

Flexible, standards-based messaging for the enterprise, cloud and Internet of Things

Broker

Interconnect Router

Clients

- A-MQ 7 on OpenShift = Messaging-as-a-service (MaaS)
- A-MQ 7 Alpha Release is available today – http://red.ht/290MtXG
- A-MQ 7 GA planned for Q1 CY 2017 (tentative)

http://red.ht/290MtXG

Thank You! Any Questions?

@dingha

RED HAT SUMMIT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.