1.1

redhat

ENTERPRISE MESSAGING AND JBOSS A-MQ

Jakub Knetl

jknetl@redhat.com

LECTURE OUTLINE

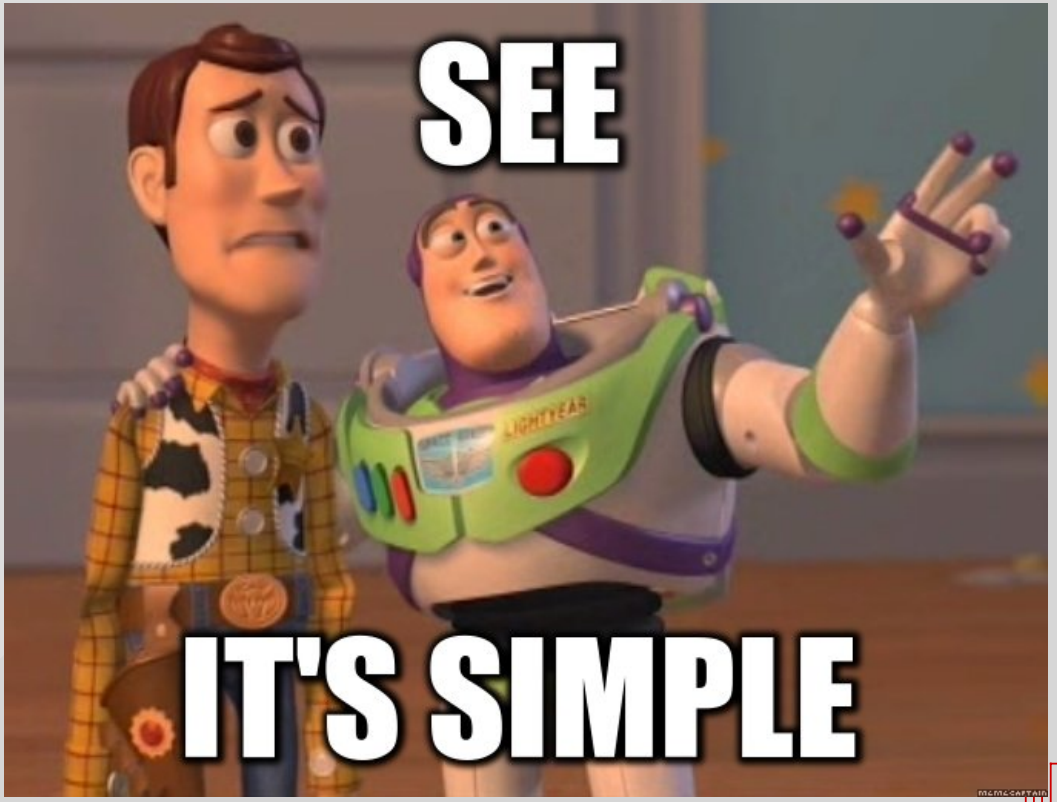
- Messaging systems
  - JMS specification
  - JMS API
- JBoss A-MQ
  - JBoss A-MQ and Apache ActiveMQ
  - Protocols
  - topologies
- Apache Artemis

WHAT IS ENTERPRISE MESSAGING

Method for asynchronous message (data) transfer between different systems which allows to integrate these systems.

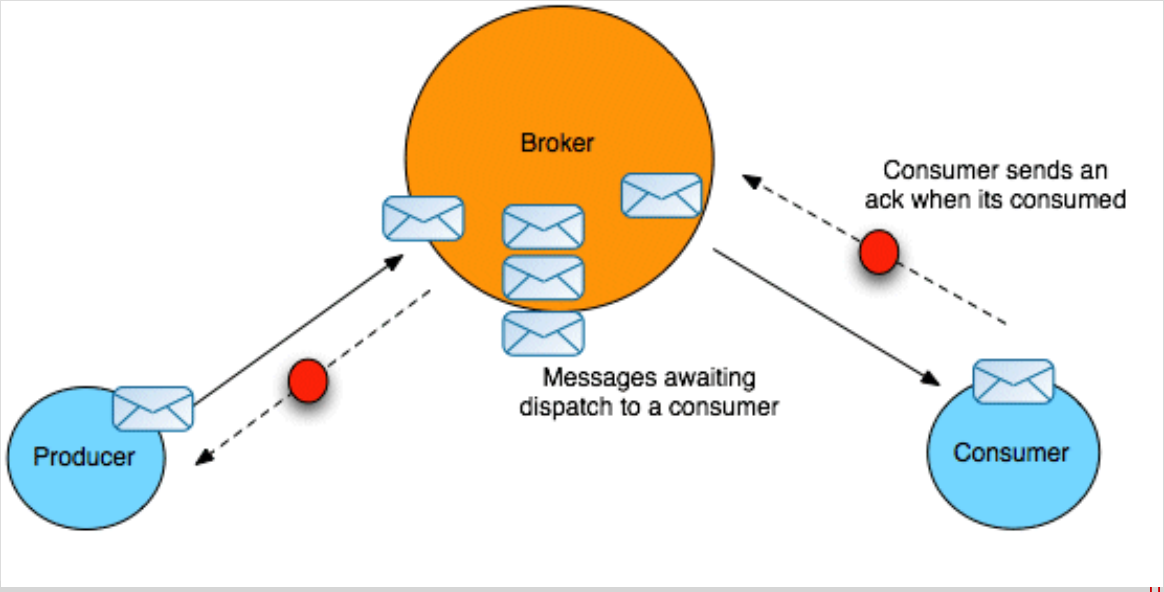
WHAT IS ENTERPRISE MESSAGING

Method for asynchronous message (data) transfer between different systems which allows to integrate these systems.



ENTERPRISE MESSAGING ARCHITECTURE

- data exchange between applications
- Message oriented middleware (MoM)
  - Message broker servers as mediator between communicating parties



WHY DO WE NEED ANOTHER COMMUNICATION METHOD?

Other methods:

- Webservices (REST, SOAP)
- JDBC and ORM
- RMI
- CORBA
- ...

BENEFITS OF MESSAGING SYSTEM

- Asynchronous communication
- Loose coupling
- Scalability
- Reliability
- Message routing and transformation

EXAMPLE USE CASE

Web music streaming service, which allows to upload and store your own music files. Music files are stored in the ogg format in some shared storage (NAS, cloud, whatever, ...). However it allows to upload music in many formats (mp3, flac, wav, aac, and others) which are converted by the service into ogg.

Where will we convert data?

EXAMPLE USE CASE

Web music streaming service, which allows to upload and store your own music files. Music files are stored in the ogg format in some shared storage (NAS, cloud, whatever, ...). However it allows to upload music in many formats (mp3, flac, wav, aac, and others) which are converted by the service into ogg.

Where will we convert data?

- server side?
- client side?

Usually none of them because of required CPU performance...

EXAMPLE USE CASE - SOLUTION

1. server will just simply store data into temporary place shared storage
2. It will sends message containing metadata about music file into a queue
3. There may be arbitrary numbers of running consumer, which will perform encoding and upload into final storage location in ogg format.

Benefits

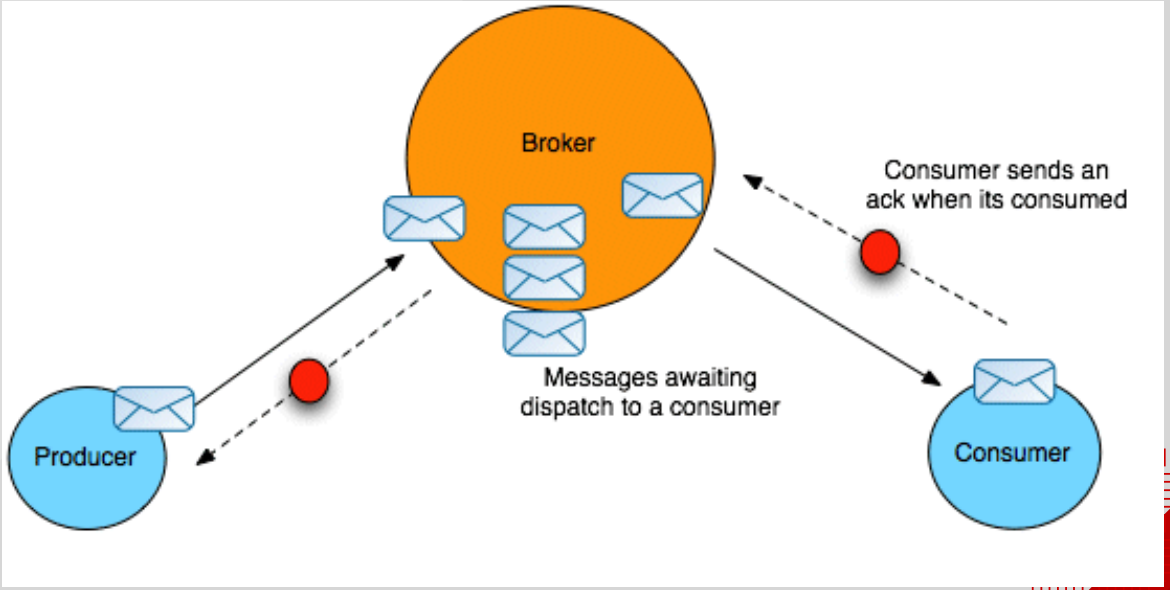
- **asynchronous communication** - server doesn't need to wait neither until data are encoded or sent
- **loose coupling** - server doesn't need to know anything about the encoder
- **scalability** - we may just start new processes as encoders
- **message routing and transformation -** message may be arbitrary routed between using metadata information

JMS SPECIFICATION

- goal is to:
  - provide messaging functionality to the java applications
  - maximizes portability between messaging products
  - define common messaging concepts
- JMS provides API for messaging products
- JMS **is not** messaging system!
- JMS **does not** address:
  - load balancing
  - fault tolerance
  - administration
  - wire protocol
  - security

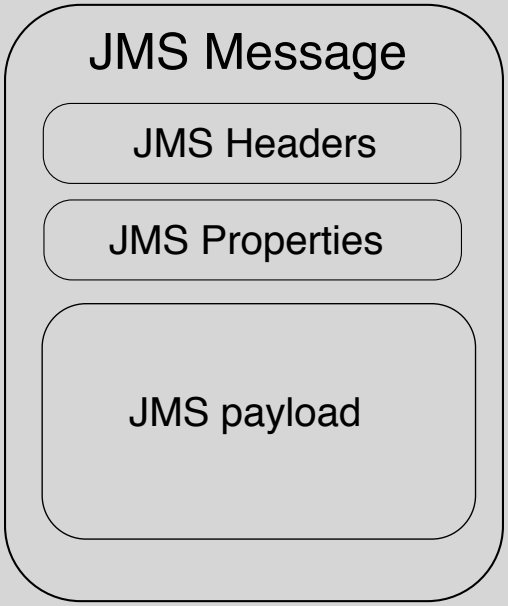
BASIC TERMS

- JMS provider
- JMS client
  - producer/consumer
- non JMS client
- JMS destinantion
- JMS domains
- JMS Message



MESSAGE STRUCTURE

- Headers
  - metadata about message
  - stored as key value pairs
  - two types (differs only semantically):
    - headers (same for all messages)
    - properties
      - arbitrary key-values (some of them are standardized)
- Payload
  - may contain different types of payload



MESSAGE HEADERS LIST

Header name	meaning
JMSDestination	destination on the broker
JMSDeliveryMode	persistent or nonperistent delivery mode
JMSExpiration	message will not be delivered after expiration
JMSMessageID	Identifiaction of the message
JMSPriority	Number 0 - 9 (0-4 low, 5-9 high priority). Advise only
JMSTimestamp	Time when the message is handed to provider for send
JMSCorrelationID	links message to another one
JMSReplyTo	Destination for reply
JMSRedelivered	Contains true if the message was likely redelivered

PREDIFINED PROPERTIES

Search through JMS specification!

MESSAGE SELECTORS

- Message filtering based on properties and headers
- Condition based on subset of SQL92

```
PRICE <= 1000 AND COLOR = 'RED'
```

```
FLIGHT_NUMBER LIKE 'N14%'
```

See examples!

MESSAGE CONTENT

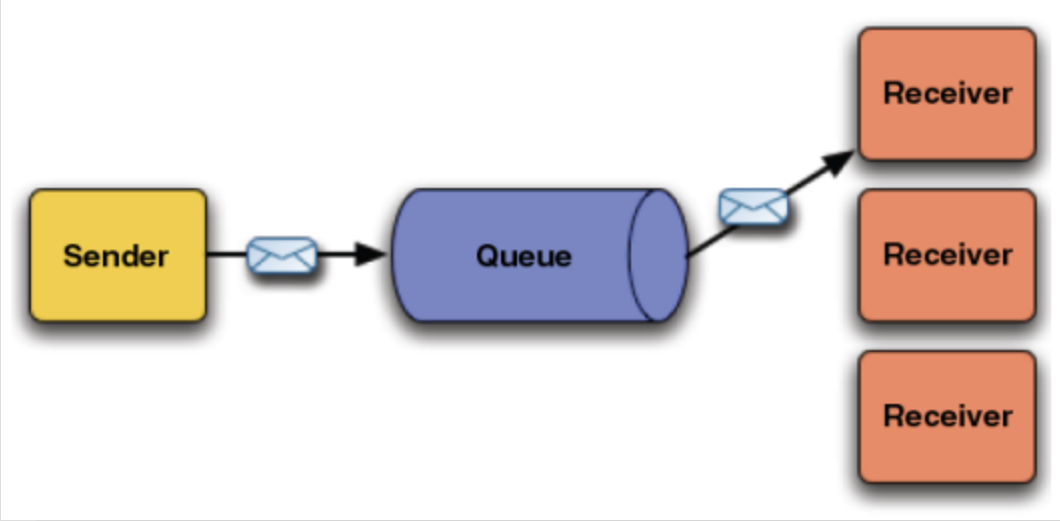
- There are several types of message defined in JMS:
  - TextMessage - String content
  - MapMessage - "body contains a set of name-value pairs where names are Strings and values are Java primitive type"
  - BytesMessage - stream of uninterpreted bytes bytes
  - StreamMessage - stream of java primitive values (read and filled sequentially)
  - ObjectMessage - Serializable object

COMMUNICATION DOMAINS

- Communication type:
  - Point-to-point (PTP)
  - Publish/Subscribe (Pub/Sub)

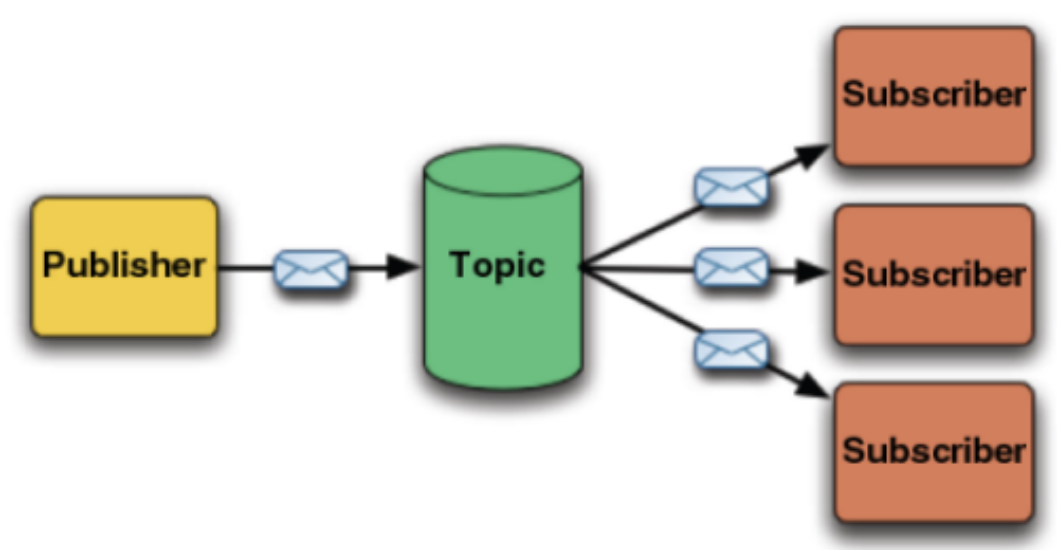
POINT TO POINT COMMUNICATION

- Destination is a queue
- one of multiple consumers gets message
- Load is distributed across consumers
- Message is stored until some consumers receives it



PUBLISH SUBSCRIBE DOMAIN

- Destination is a topic
- message is delivered to all subscribers
- message is thrown away if there is no subscription
- Durable subscriber
  - if durable subscriber disconnects broker is obliged to store all messages for later delivery



IT IS TIME FOR HISTORY...

JMS API

- JMS 1.0 (2001)
  - Different APIs for pub/sub and PTP communication
- **JMS 1.1** (2002)
  - Classic API - unified API for pub/sub and PTP
- JMS 2.0 (2013)
  - Classic API
    - it is not deprecated and will remain part of JMS indefinitely
  - Simplified API
    - less code needed
    - AutoCloseable resources -> Java 7 needed
    - no checked exceptions

CLASSES OF CLASSIC API

- ConnectionFactory
- Connection - heavyweight object (allocates resources outside JVM)
- Session - lightweight object
  - factory for MessageProducer, MessageConsumer and Destination
  - should not be shared between threads
  - defines serial order and transaction context
- MessageProducer
  - for producing messages
- MessageConsumer
  - for consuming messages
- Destination - administered object
- Message

RELIABILITY OF MESSAGE

- delivery mode
- acknowledgement mode
- transactional mode

DELIVERY MODE

- determines level of delivery reliability
  - Persistent (default)
    - provider should persist the message
    - message must be delivered once and only once even in case of provider failure
  - Nonpersistent
    - provider is instructed not ot presist the message
    - message must be delivered at most once
    - message is usually lost on provider failure
    - better performance

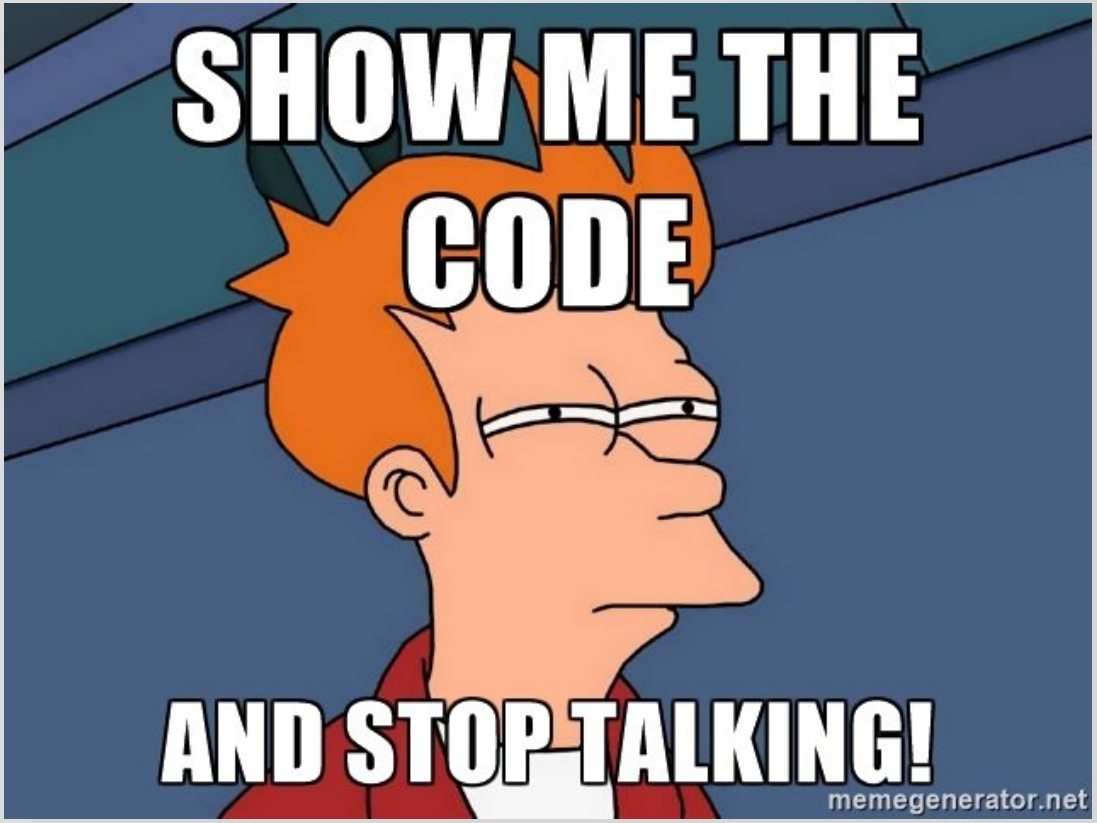
MESSAGE ACKNOWLEDGEMENT

- Delivery between broker and client is not considered successful until message is acknowledged.
- acknowledgement modes:
  - AUTO_ACKNOWLEDGE
  - DUPS_OK_ACKNOWLEDGE
  - CLIENT_ACKNOWLEDGE

TRANSACTIONS

- Session can be transacted
- multiple messages handled as atomic unit
- transaction is completed by calling commit() or rollback() on session
- commit also acknowledges message
- Support for distributed transaction is not required by JMS
  - but still many providers implement distributed transactions
  - JMS recommends support using JTA XAResource API

CODE EXAMPLES



PART II

APACHE ACTIVEMQ

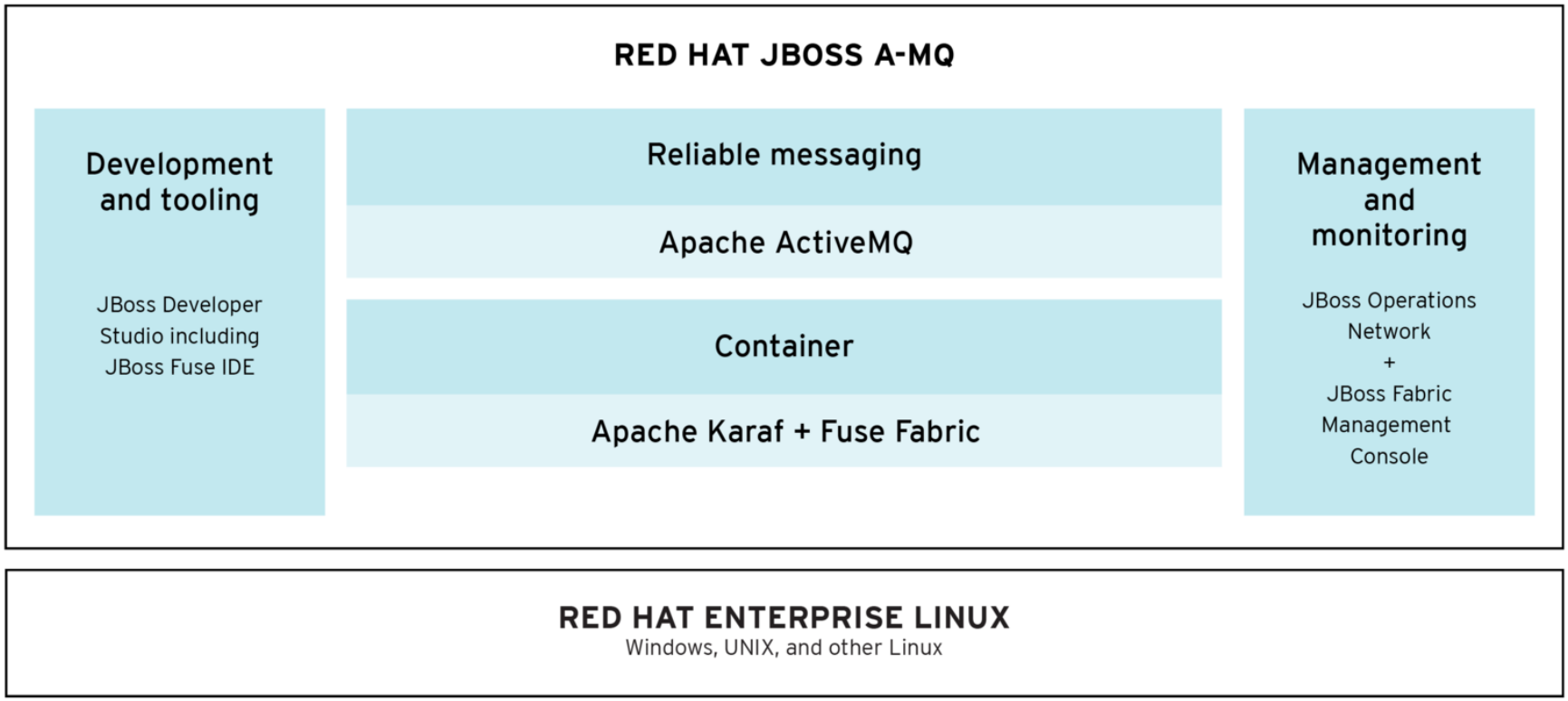
APACHE ACTIVEMQ

- Opensource MoM
- JMS 1.1 compliant
- Supports many protocols and clients
- other features:
  - High availability
  - scalibility
  - management
  - security

JBOSS A-MQ

- Open-source messaging platform
- Messaging system based on Apache ActiveMQ
- Runs on OSGI container
- Enable easy deployment
- Provides web based management console

JBOSS A-MQ


RED HAT JBOSS A-MQ
Development and tooling
JBoss Developer Studio including JBoss Fuse IDE
Reliable messaging
Apache ActiveMQ
Container
Apache Karaf + Fuse Fabric
Management and monitoring
JBoss Operations Network
+
JBoss Fabric Management Console
RED HAT ENTERPRISE LINUX
Windows, UNIX, and other Linux
JB0009

CONFIGURATION

- XML file
- most of things work out of the box
- configuration example

MESSAGE STORES

- kahaDB
- multi kahaDB
- levelDB
- JDBC

```
<persistenceAdapter>
     <kahaDB directory="${activemq.data}/kahadb"/>
</persistenceAdapter>
```

```
  <persistenceAdapter>
     <jdbcPersistenceAdapter dataSource="#derby-ds"/>
  </persistenceAdapter>


<!-- Embedded Derby DataSource Sample Setup -->
<bean id="derby-ds" class="org.apache.derby.jdbc.EmbeddedDataSource">
  <property name="databaseName" value="derbydb"/>
  <property name="createDatabase" value="create"/>
</bean>
```

CONNECTION TO BROKER

- Transport connectors
  - For client to broker connections
- Network connectors
  - For broker to broker connections
- Many transport protocols supported:
  - tcp, udp, nio, ssl, http/https, vm

WIRE PROTOCOLS

- Openwire
- STOMP
- AMQP
- MQTT

PROTOCOL URIS

- determines what:
  - protocol will be used
  - location of the broker (typically hostname and port)
- high level uris:
  - typically uses composite URI
  - failover
  - fabric

```
failover:(tcp://primary:61616,tcp://secondary:61616)?randomize=false
```

RUNTIME MANAGEMENT

There are three options for management:

- web console
- JMX
- Karaf OSGi console

DEMO!

HIGH AVAILABILITY (HA)

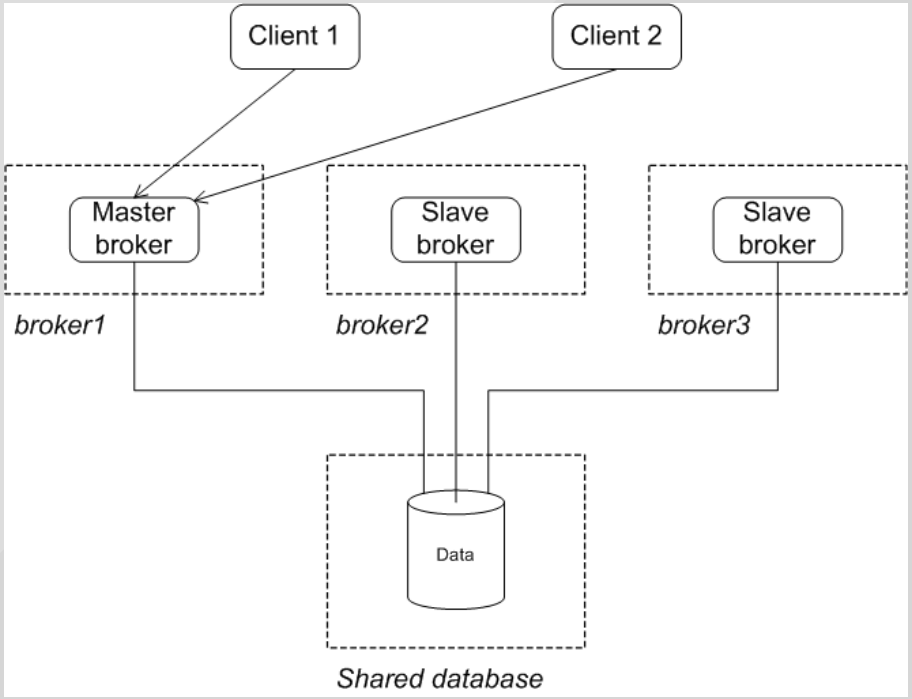
- Messaging systems usually processes business critical data
- broker must be accessible 24/7
- ActiveMQ provides various mechanisms to ensure HA

HA IN ACTIVEMQ

- Group of brokers forms logically one broker
- Master broker
  - communicates with clients
- Slave brokers
  - Passive (all connectors are stopped)
- election mechanisms
- client reconnects in case of failure (failover)
- Message acknowledgment after the message is stored safely

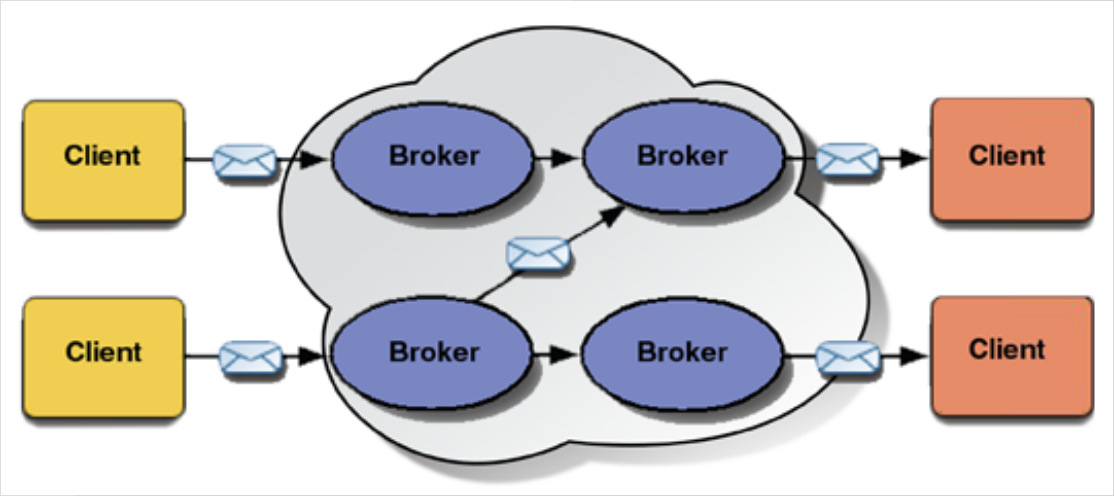
MASTER SLAVE FOR HA

- Shared JDBC master/slave
- Shared file system master/slave
- Replicated levelDB master/slave



SCALABILITY: NETWORK OF BROKERS

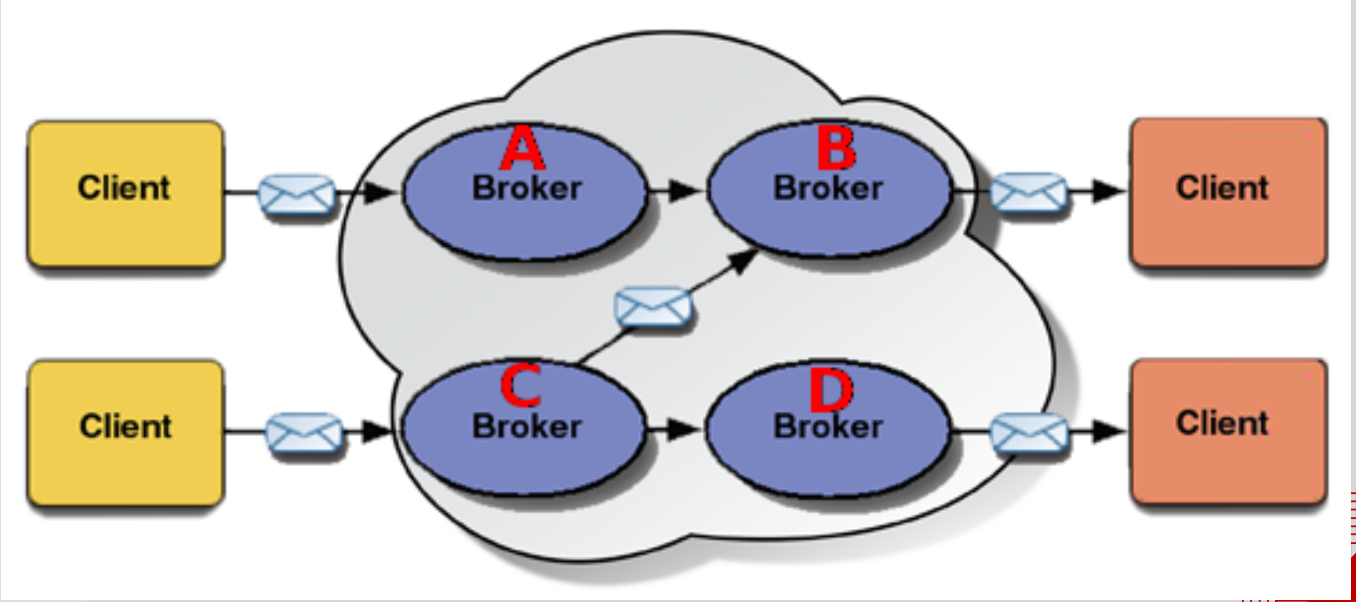
- connections between broker
- message forwarding
- enables massive scalability
- requires careful configuration



NETWORK CONNECTOR

Broker A :

```
<networkConnectors>
    <networkConnector uri="static:(tcp://B:61617)"/>
</networkConnectors>
```

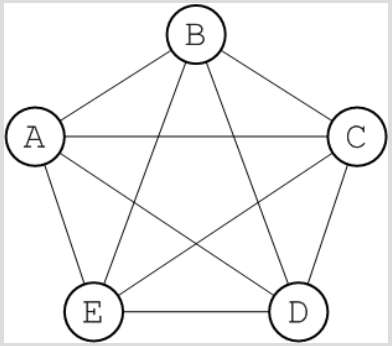


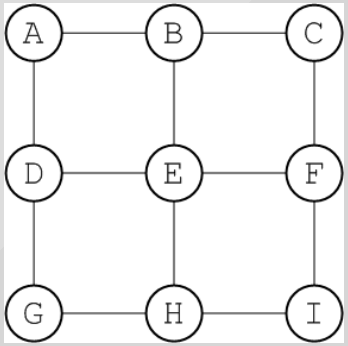
NETWORK OF BROKERS

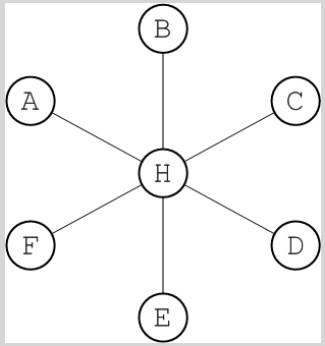
- duplex connections
- destination filtering
- dynamic vs static forwarding
- AdvisoryMessages
- network consumer priority
- networkTTL

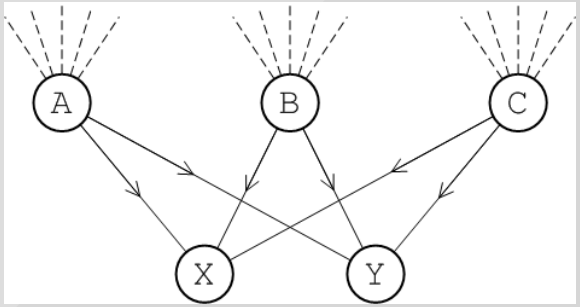
HIERARCHIES OF NETWORKS

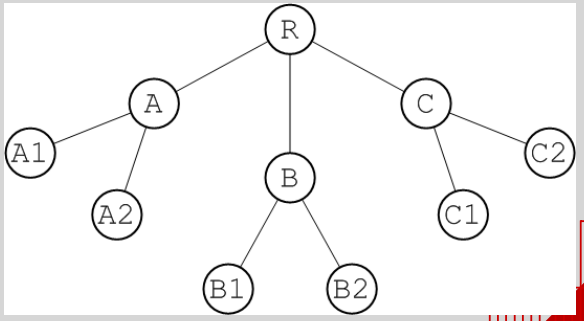
- concentrator topology
- hub and spokes topology
- tree topology
- mesh topology
- complete graph











OTHER ACTIVEMQ FEATURES

- exclusive consumers
- message groups
- composite destinations
- wildcards (. * >)
- virtual destinations

See http://activemq.apache.org/features.html

APACHE ACTIVEMQ ARTEMIS

- new Apache MoM
- non-blocking architecture => great performance
- merges codebase with JBoss HornetQ
- JMS 2.0 compliant
- Support for:
  - ActiveMQ clients
  - AMQP
  - STOMP
  - HornetQ clients

More details on Artemis website.

redhat®

THANK YOU!