Red Hat A-MQ – 7.x

A lightweight, high-performance, robust messaging platform

– **Avadhut**

About Me

- I'm Avadhut!
- Working as Senior Integration Engineer
- Open-source enthusiast
- Active community member for couple of open-source projects
- Worked with Integration, Fuse, Camel, Karaf, Kafka and messaging platforms for quite a bit
- Did full production deployments, architecture review and performance tuning for couple of employers and lot of Red Hat customers

** You can find me on:

GitHub: https://github.com/kodtodya

GitLab: https://gitlab.com/kodtodya

LinkedIn: https://www.linkedin.com/in/kodtodya/

Twitter: https://twitter.com/kodtodya

Pre-Requisite for A-MQ Course

- Knowledge of Java or overall programming (We will use Java-8 for this course) [to create clients]
- Linux(Any flavor) and Mac are strongly preferred, avoid Windows if possible
- Knowledge of loosely coupled architecture is helpful..
- Willingness to learn an awesome technology... 

Who is this course for?

- **Developer:** who would like to learn how to write and run an application that integrates RH A-MQ-7 messaging servers

- **Architects:** who want to understand the role of A-MQ in the enterprise integration and messaging with loosely coupled systems

- **DevOps:** who want to understand how A-MQ works with regards to various systems, protocols, components and its infrastructural setup

Welcome...!!!

- A warm welcome to Red Hat A-MQ-7.x Rapid Track course..

Course Structure

- **Part – 1 : Fundamentals**
  - Traditional way of communication
  - Need of messaging broker
  - What is loosely coupled system?
  - Core Concepts (To be covered in subsequent slides)
  - What is A-MQ?
  - History of A-MQ
  - A-MQ Architecture
  - A-MQ Installation
  - A-MQ Eco-System
  - JMS Basics
  - A-MQ Management and Command Line Interface (Fuse CLI)
  - A-MQ Operations

Course Structure

Part – 2 : Core Concepts & Part – 3 : Practical

- Core Concepts (To be covered in subsequent slides)
- A-MQ Producer
- A-MQ Consumer
- A-MQ Topologies
- A-MQ DLQ & Message re-delievery
- A-MQ Persistence
- Master-Slave configuration

Exception

What I am not going to cover

- Your UAT and production issues
- Your enterprise application issues
- Your enterprise project architecture
- Advance clustering
- A-MQ migrations

Red Hat A-MQ Theory

Fundamentals

History of Messaging

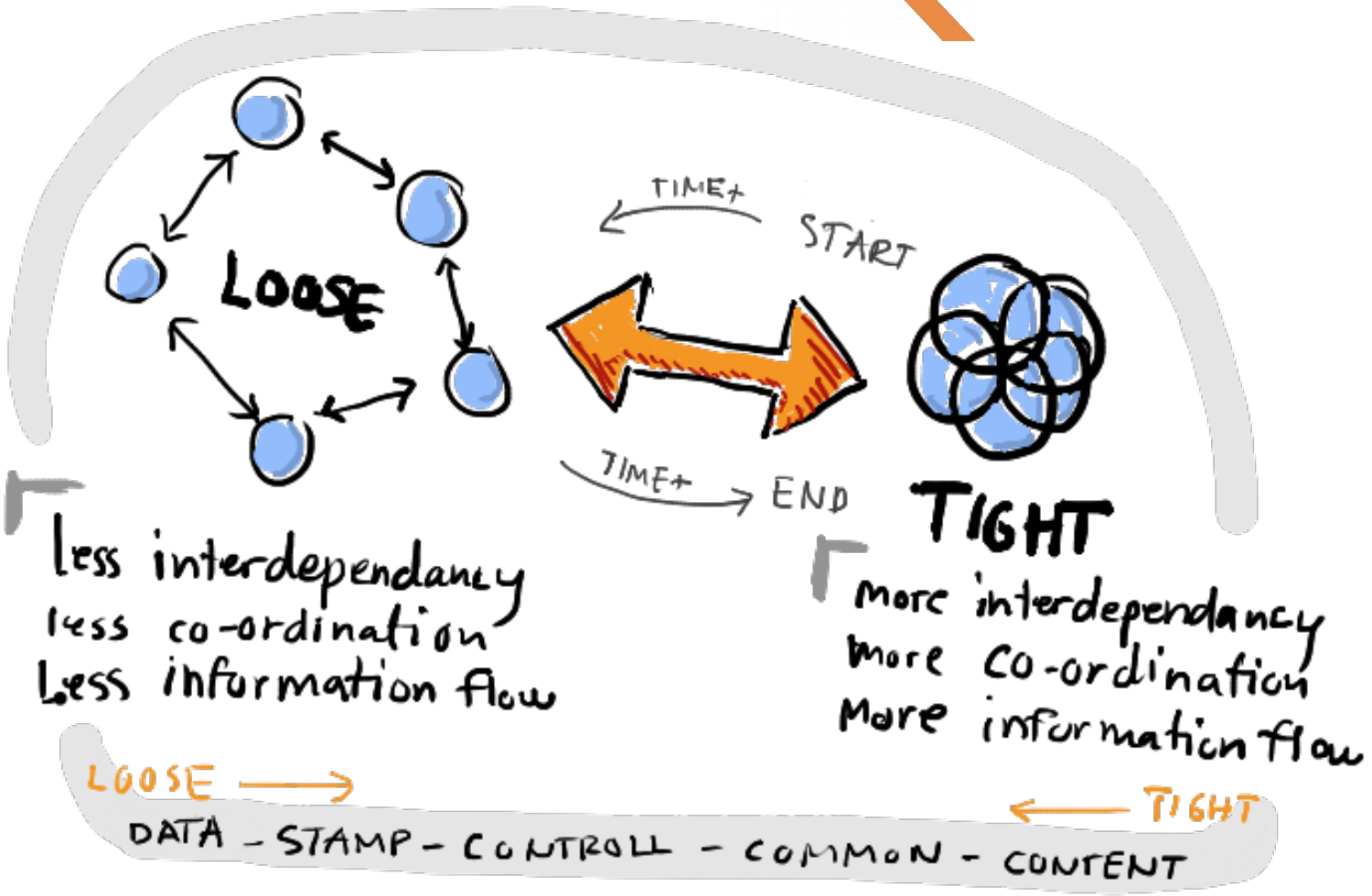
TIGHTLY COUPLED COMMUNICATION IN OLD ERA

FAILURE OF ONE OF THE ENDPOINT IS FAILURE OF COMMUNICATION

Problems

- If one of the system goes down, then entire communication system will collapse as there won’t be any listener to listen

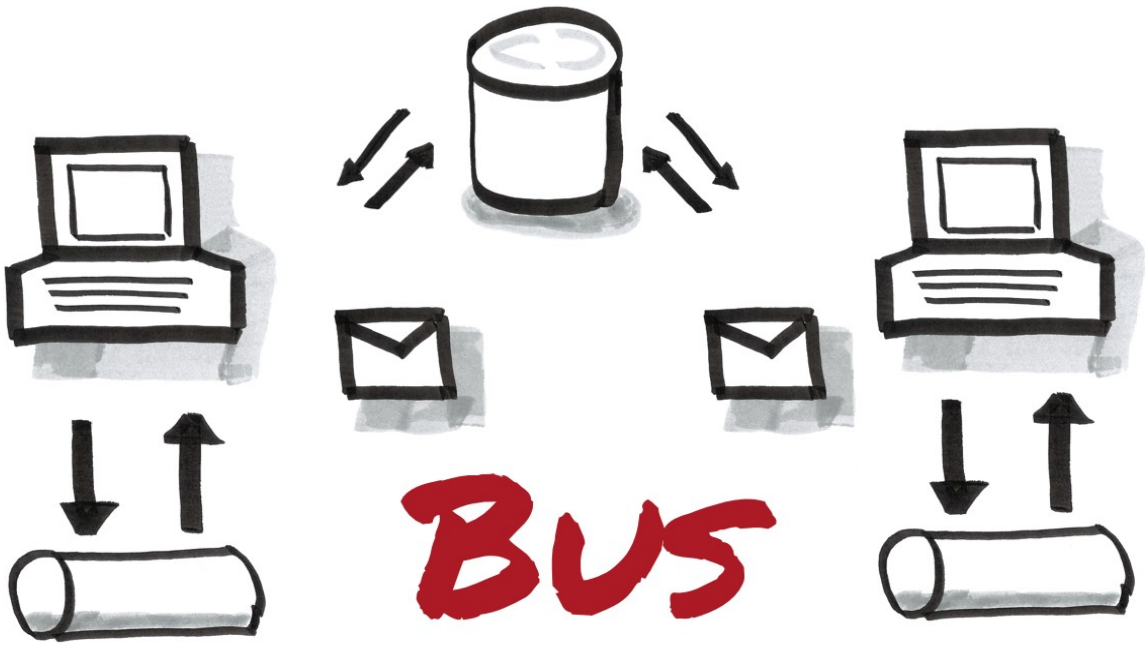
What is Loosely Coupled Approach?



Loosely Coupled Approach

Loosely Coupled Approach

LOOSELY COUPLED APPROACH FOR COMMUNICATION IN MESSAGING



Traditional Messaging Systems

Apache:
ActiveMQ
Qpid
Artemis
QDB (supports message replay by timestamp)

Red Hat:
HornetQ
JBoss Messaging (JBoss)
Fuse Message Broker/A-MQ (enterprise ActiveMQ)

IBM:
IBM Integration Bus
IBM Message Queues

Microsoft:
Microsoft Azure Service Bus
Microsoft BizTalk Server

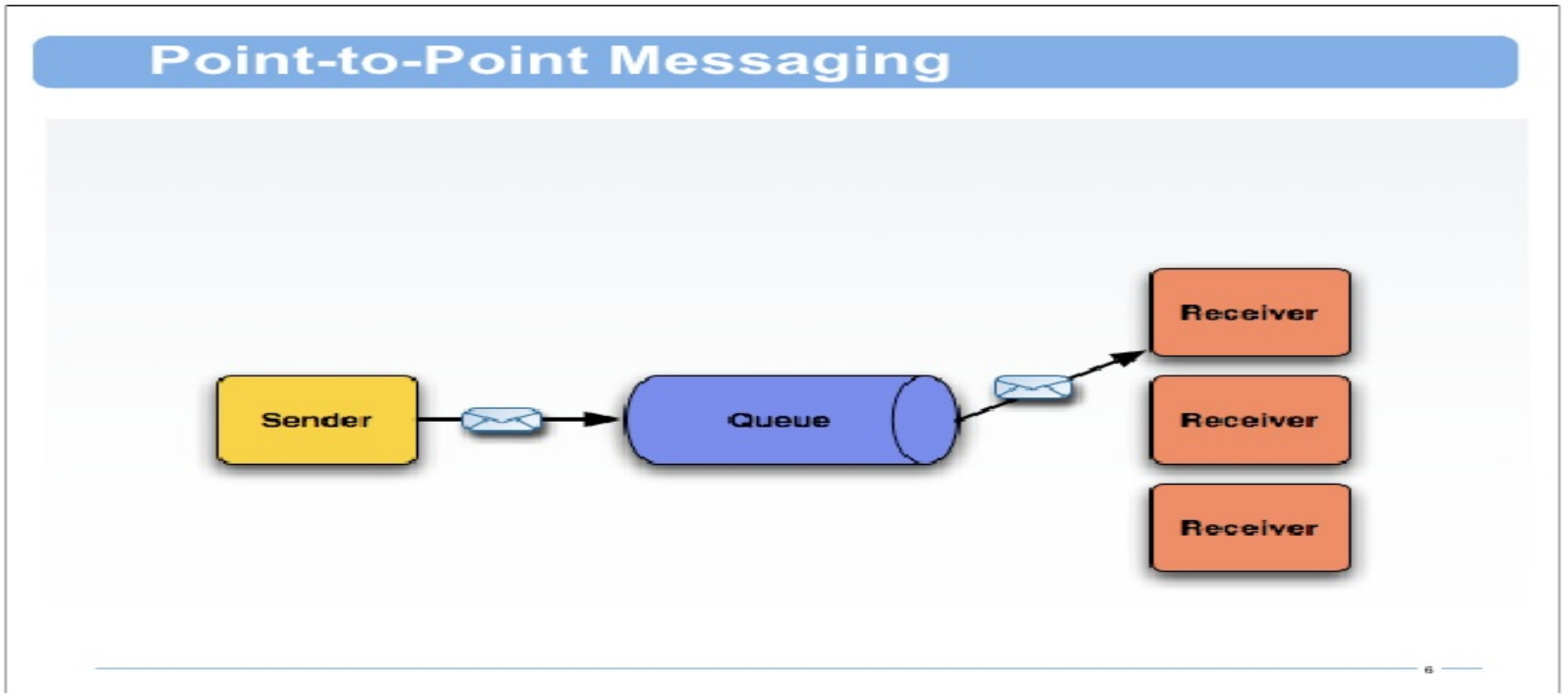
Tibco EMS

Oracle:
Oracle Message Broker

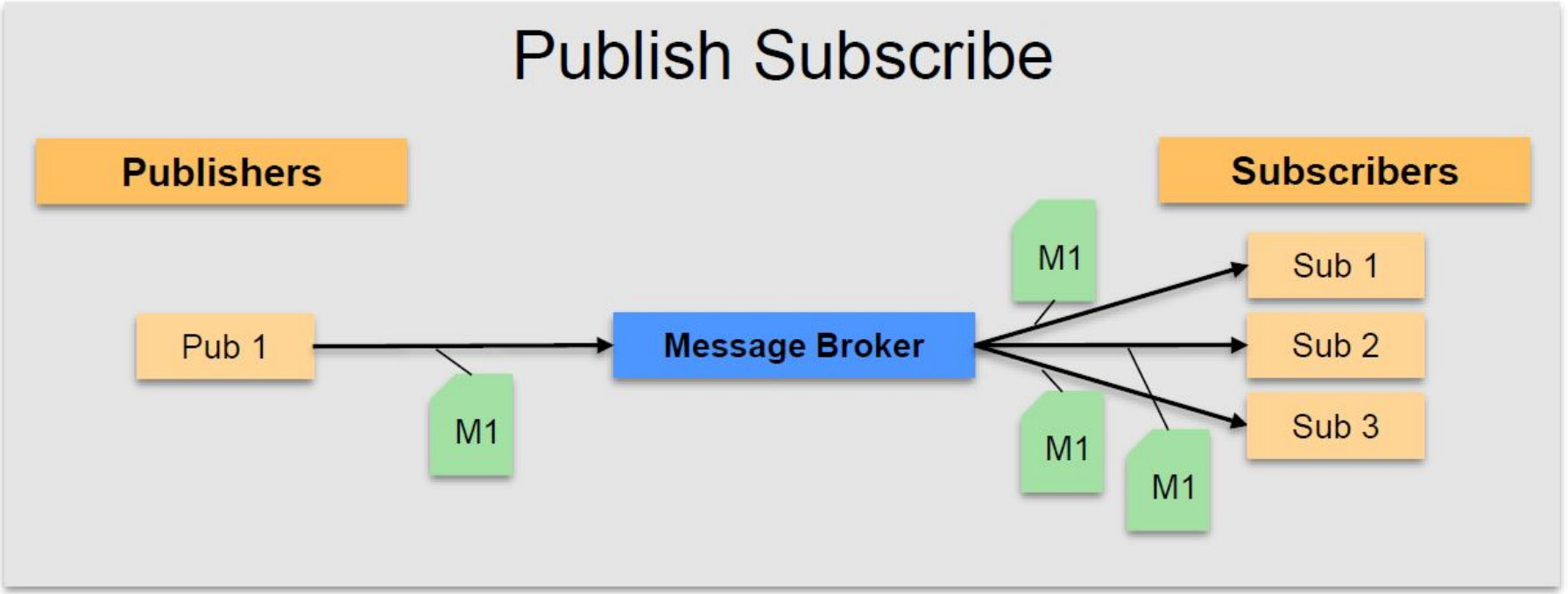
RabbitMQ (Mozilla Public License, written in Erlang)

WSO2 Message Broker

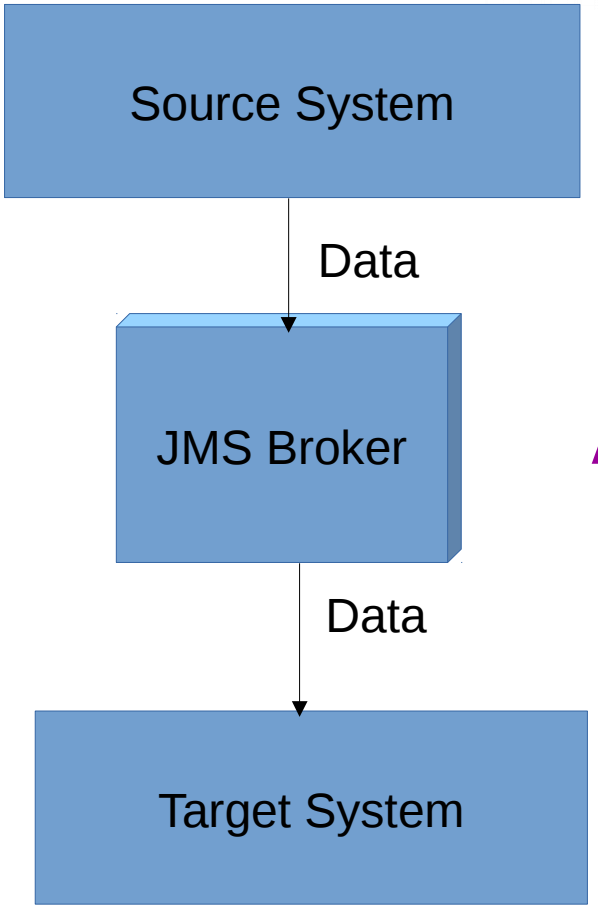
What does they provide?



What does they provide?

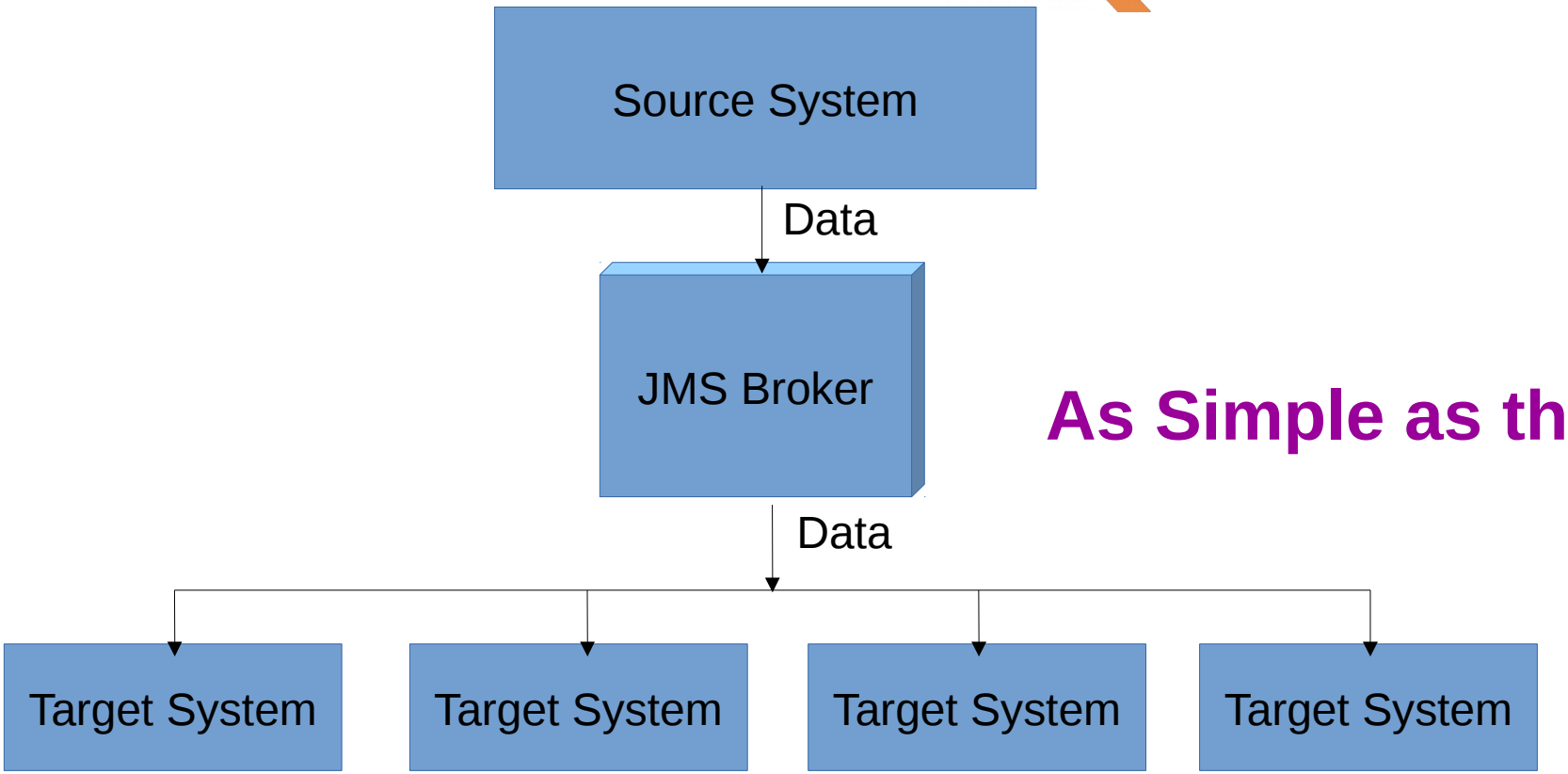


Normal Situation



As Simple as that !

Normal Situation

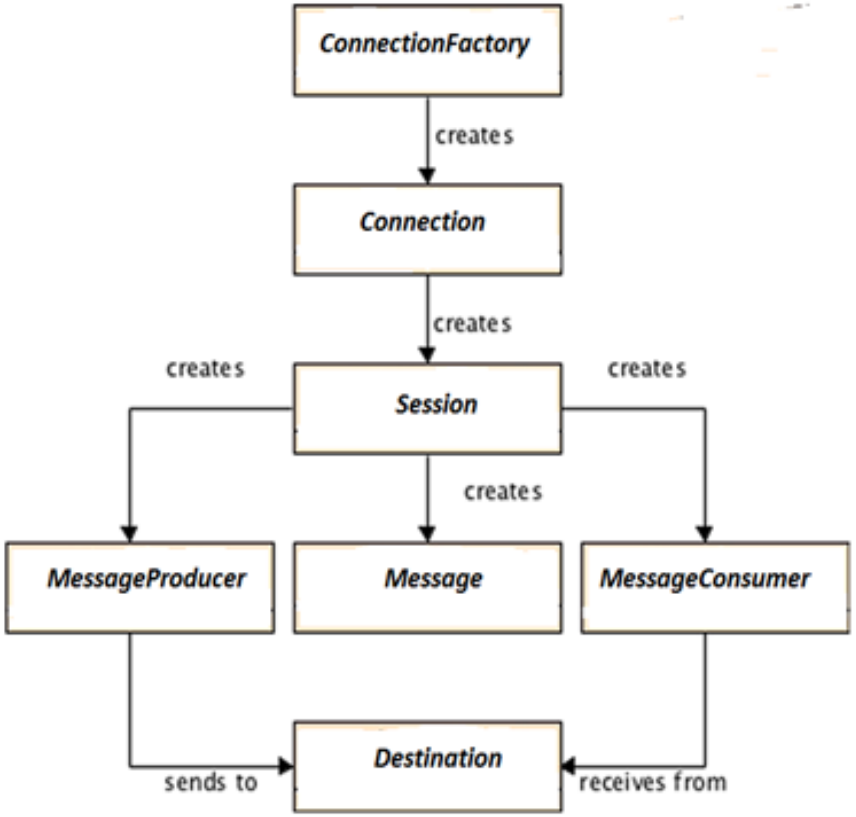


As Simple as that !

What is JMS?

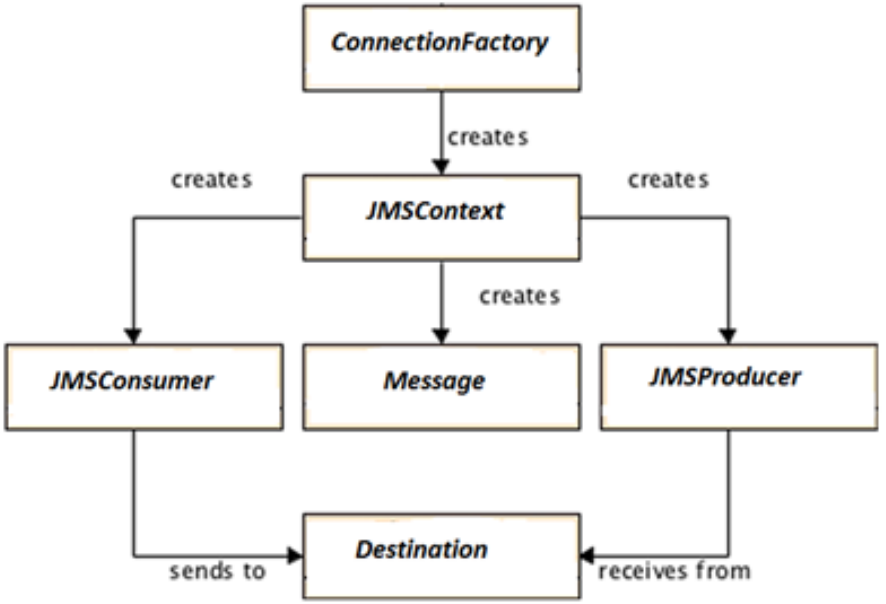
- JMS : Java Messaging Server
- A message broker that is written in Java
- JMS servers can be called as brokers, nodes or messaging systems
- JMS brokers follows JMS Specifications
- JMS specifications are nothing but Java interfaces released by Oracle
- Different companies implement these specifications and develop broker

Difference in JMS Specification



JMS V.1.1 Classic API Overview

JMS-1.x



JMS V2.0 API Overview

JMS-2.x

What is AMQ?

RED HAT®
AMQ

fabric8

Apache
ACTIVEMQ

- One of the leading open-source messaging broker
- Community Apache ActiveMQ Artemis is the upstream product of (RH)AMQ-7
- This is provided as embedded broker in Fuse

History of AMQ?

- ActiveMQ was initially created by its founders from LogicBlaze in 2004, as an open source message broker, hosted by CodeHaus
- The code and ActiveMQ trademark were donated to the Apache Software Foundation in 2007, where the founders continued to develop the codebase with the extended Apache community
- Apache ActiveMQ was build with further few new features and made available by Red Hat which called as Red Hat AMQ
- In October 2014, Apache ActiveMQ Artemis project was started from the code from Apache ActiveMQ and HornetQ code donated by Red Hat to Apache
- Latest version of AMQ internally uses Apache ActiveMQ Artemis (here onwards referred as Artemis)
- It was already container based and now a days we have capability to deploy it on openshift hybrid cloud environment

What additional features AMQ provides?

- AMQP protocol support
- MQTT support
- STOMP protocol support
- JMS 2.0 and 1.1 support
- Flexible Clustering
- Queue memory limitation
- SSL support
- Management over JMX, JMS and core protocol
- Large message support
- Diverts
- OpenWire support for ActiveMQ 5 clients
- HornetQ Core protocol support for HornetQ 2.4,2.5 clients
- High availability with shared store and non shared store (replication)
- High performance journal for message persistence
- Producer flow control
- Topic hierarchies
- Consumer flow control
- Message Groups
- OSGi support

AMQ Architecture

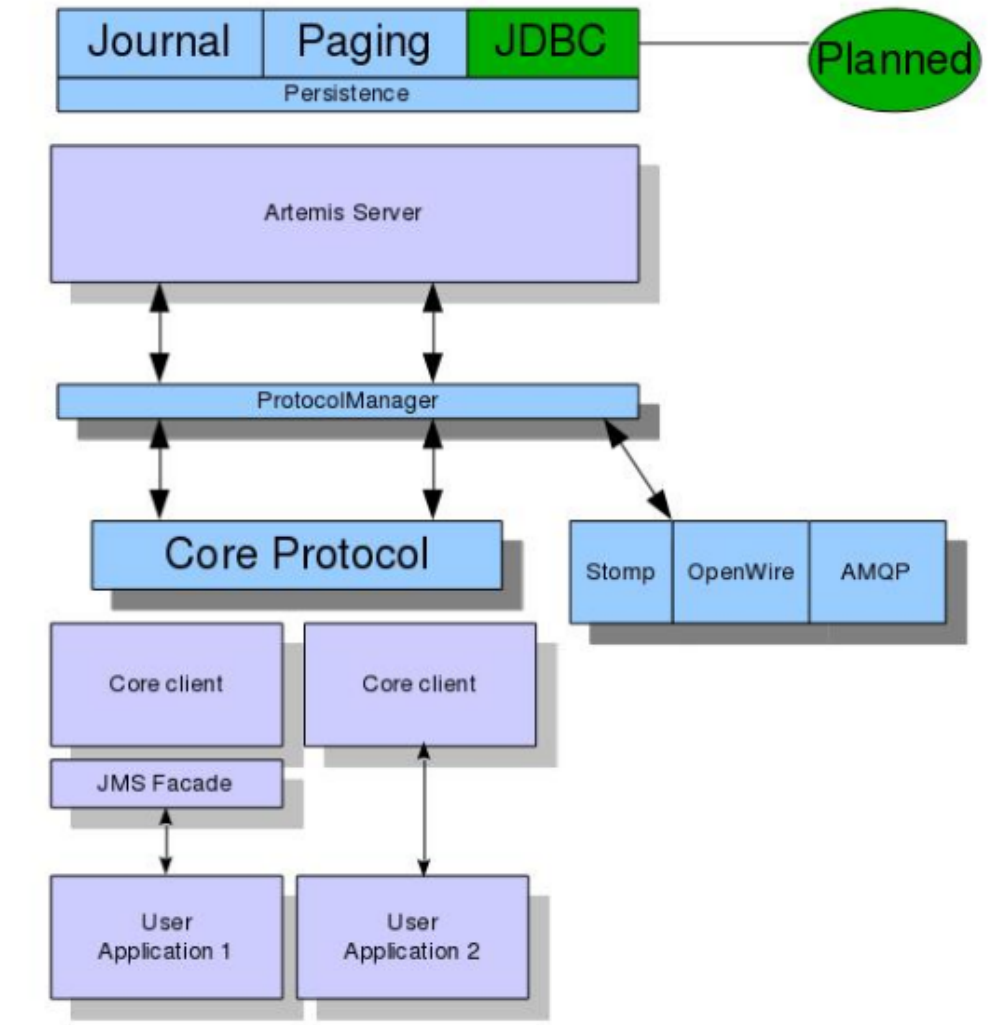


Figure 3.1 Artemis High Level Architecture

Red Hat A-MQ

Installation

AMQ Management using CLI

- **Create broker**

$artemis create mybroker

--user=admin

--default role=amq

--anonymous access=Y

- **Start Broker**

$artemis run

- **start broker in background**

$artemis start

- **Produce messages**

$artemis producer --destination helloworld --message-count 100 --url tcp://localhost:61616

- **Consume messages**

$artemis consumer --destination helloworld --message-count 50 --url tcp://localhost:61616

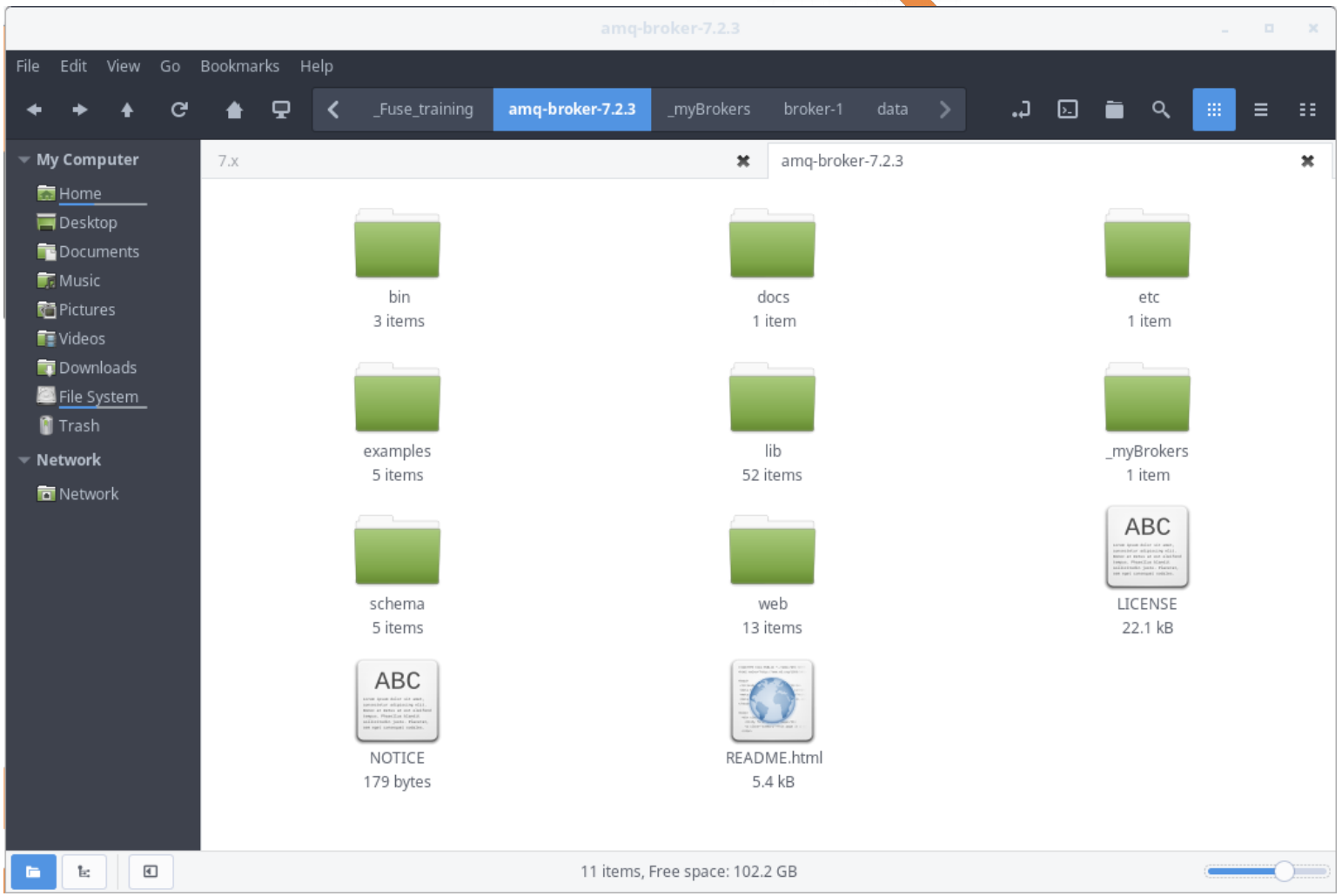
- **Stop broker**

$artemis stop

Red Hat A-MQ

Eco-System

A-MQ Eco-System



Red Hat A-MQ

JMS Basics

JMS Basics

- JMS Provider
- JMS Client
- Producer and it’s example
- Consumer and it’s example
- Connection Factory and JMSContext
- Message
- Queue and Topic
- Inflight messages
- Synchronous and asynchronous behavior

Red Hat A-MQ

Management

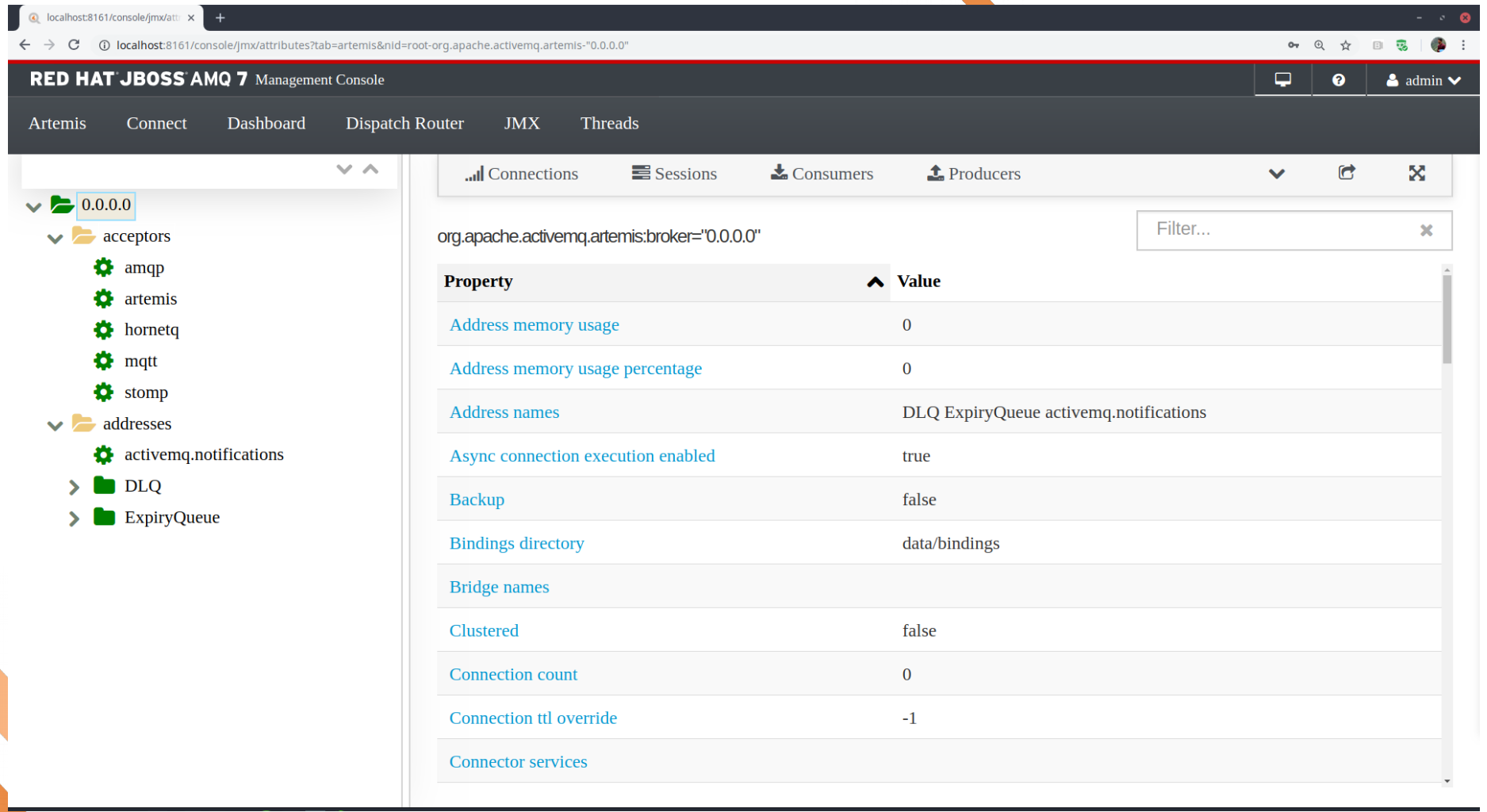
A-MQ Management

- Using Hawtio
- Using CLI

A-MQ Management - HawtIO

- **HawtIO**
  - **A modular web console for managing your Java stuffs**
  - **Composed of a collection of plug-ins, each of which is an AngularJS module**
  - **HawtIO has lot of plug-ins such as JMX, JVM, OSGi, Logs, ActiveMQ, Apache Camel and Spring Boot**
  - **Designed by considering micro-services strategy in enterprise applications**
  - **Cloud ready**
  - **Available at port number 8161 for Red Hat A-MQ**

A-MQ Management - HawtIO



A-MQ Management - CLI

- A-MQ CLI
  - A-MQ CLI is nothing but command line interface used for A-MQ
  - You can use the ./artemis script to run your commands and try to get your work done
  - Difference between setup ./artemis script and broker ./artemis script
  - You can search valid option to get your task done

Red Hat A-MQ

Operations

A-MQ Operations

address	Address tools group (create|delete|update|show) (example ./artemis address create)
browser	It will browse messages on an instance
consumer	It will consume messages from an instance
data	data tools group (print|imp|exp|encode|decode|compact) (example ./artemis data print)
help	Display help information
kill	Kills a broker instance started with --allow-kill
mask	mask a password and print it out
perf-journal	Calculates the journal-buffer-timeout you should use with the current data folder
producer	It will send messages to an instance
queue	Queue tools group (create|delete|update|stat) (example ./artemis queue create)
run	runs the broker instance
stop	stops the broker instance
user	default file-based user management (add|rm|list|reset) (example ./artemis user list)

Red Hat A-MQ

Practicals

A-MQ CLI Practicals

Please refer this repository for sample A-MQ commands:

https://github.com/kodtodya/amq-7-training

A-MQ Part-1 revision

- Part – 1 : Fundamentals
  - Traditional way of communication
  - Need of messaging broker
  - What is loosely coupled system?
  - Core Concepts (To be covered in subsequent slides)
  - What is A-MQ?
  - History of A-MQ
  - A-MQ Architecture
  - A-MQ Installation
  - A-MQ Eco-system
  - JMS Basics
  - A-MQ Management and Command Line Interface (A-MQ CLI)
  - A-MQ Operations

Questions?

Thank you..!!!

LinkedIn, GitHub, GitLab, Twitter: @kodtodya