Messaging with Spring

Active MQ vs. Rabbit MQ

Speaker:

Milan Gogic

#creatingtomorrow

Agenda

1. About the Viesure Innovation centre
2. Introduction
3. JMS – What? Why?
4. Active MQ – Terminology and architecture
5. Active MQ – Spring example
6. Rabbit MQ – Terminology and architecture
7. Rabbit MQ – Spring example
8. Differences between Active MQ and RabbitMQ
9. Use cases
10. Other solutions

viesure
innovation center

About
viesure Innovation centre

#creatingtomorrow

This is us

viesure

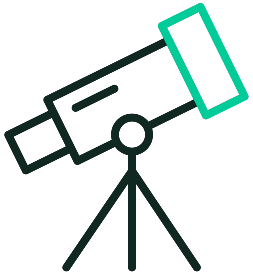
[vee-sure] ***noun***

Definitions

1. the innovation center of Wiener Städtische Versicherung, one of the largest insurance companies in Austria.
2. a playground for innovators, creatives and pioneers, who shape the future together.

Our Vision

We reimagine insurance with customer-focused innovations through creativity, empathy and technology.



Who we are

★ 3 → 57

♀ 16 ♂ 41

🌍 15

A tech savvy team with more than 30 engineers.

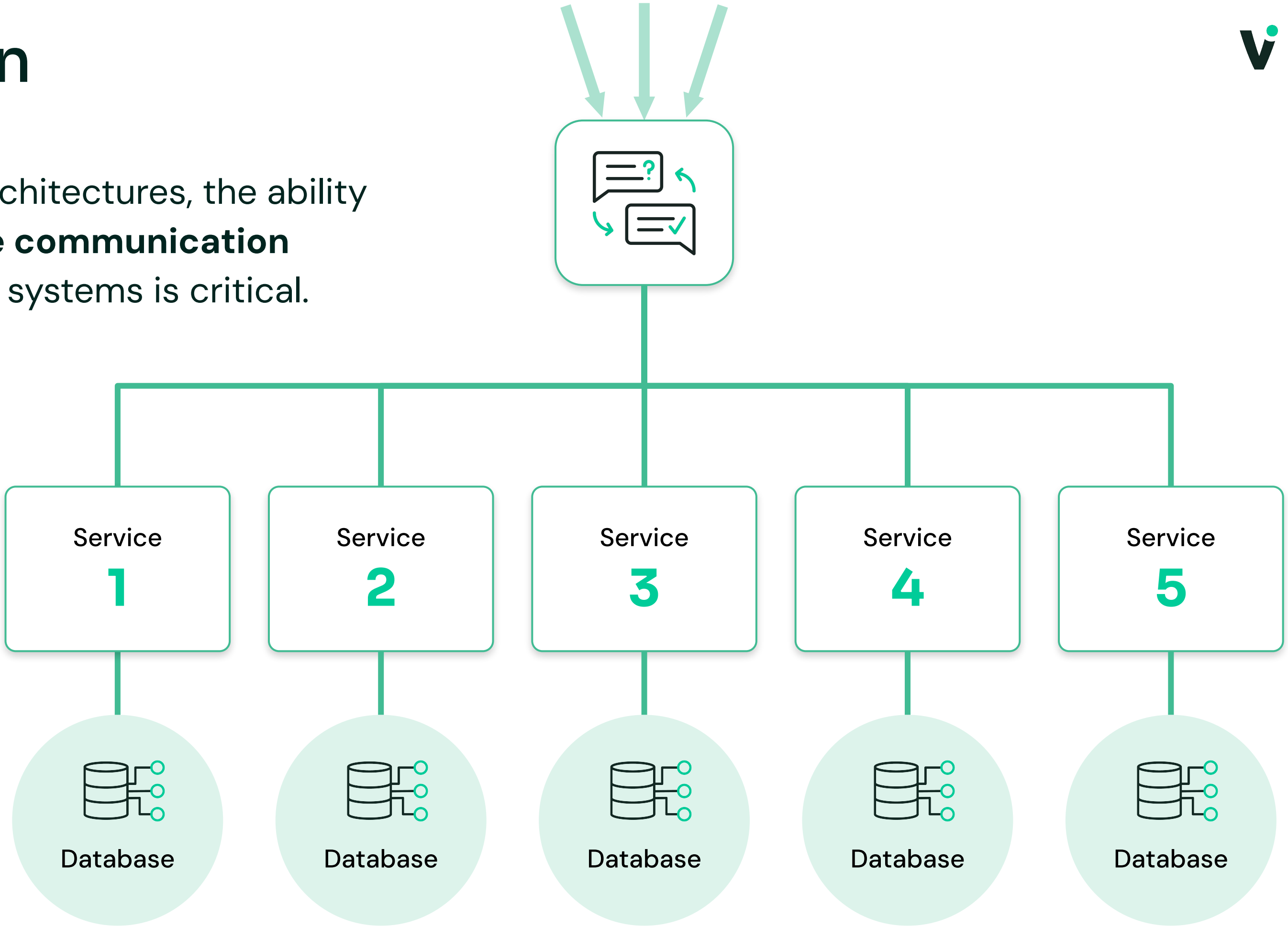
We have backend, mobile, frontend, dev ops and AI expertise.

https://viesure.io/careers/

https://www.linkedin.com/company/viesure/

Introduction

In modern software architectures, the ability to **efficiently manage communication** between services and systems is critical.



viesure
innovation center

Jakarta Messaging (JMS)

#creatingtomorrow

JMS (Jakarta Messaging)

JMS *(Jakarta Messaging, earlier Java Message Service)* is a messaging standard that allows Java-based applications to create, send, receive, and consume messages

JMS is an API that provides the facility to **create, send, and read messages.**

It provides loosely coupled, reliable, and asynchronous communication.

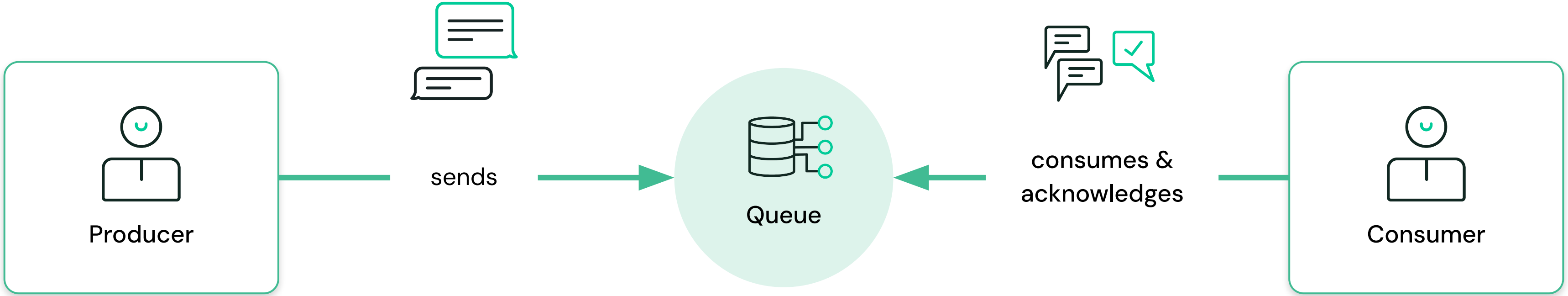
Advantages of JMS

Reliable

Asynchronous

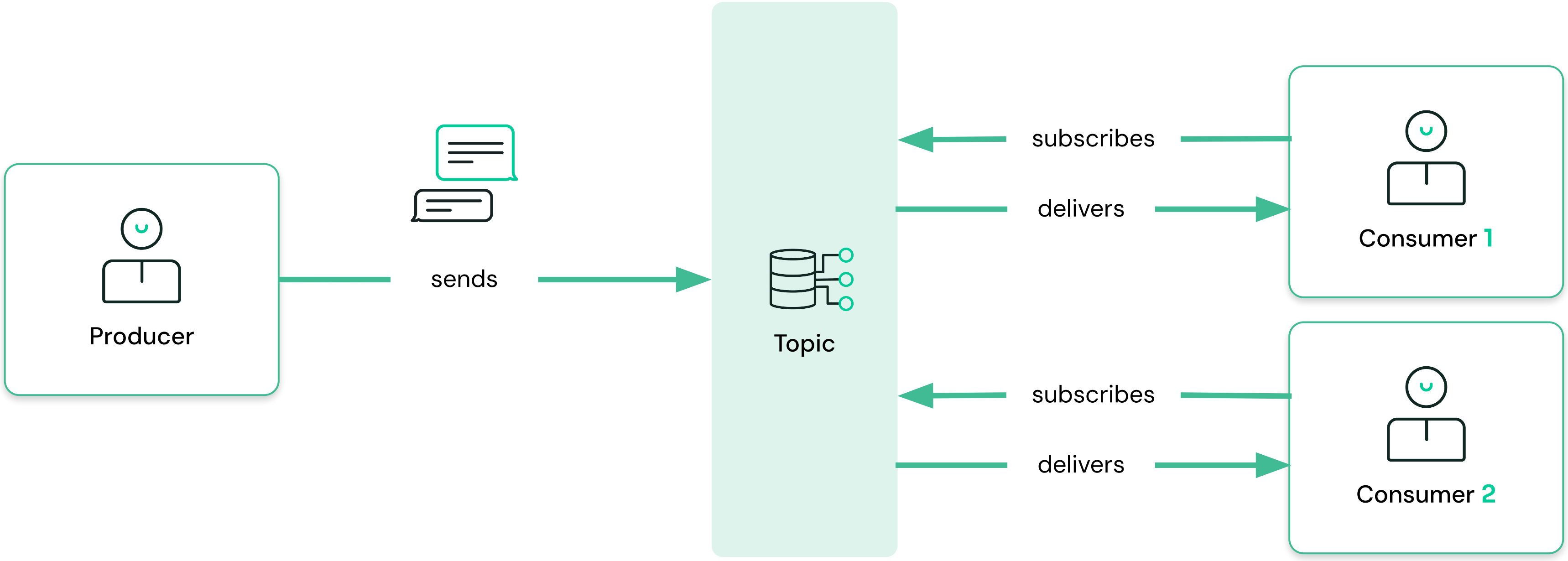
JMS - Messaging domains

Point to Point Messaging Domain



JMS - Messaging domains

Publisher/Subscriber (Pub/Sub) Messaging Domain



Active MQ 🏋️‍♀️

Active MQ as a JMS supporting messaging Broker

How ActiveMQ Messages Work

Messages are arranged into two patterns:
Queues
Topics

Queues – FIFO (first-in, first-out) pipelines of messages produced and consumed by brokers and clients

Topics – subscription-based message broadcast channels

A topic implements a publish and subscribe workflow.

A queue implements a load-balancing workflow.

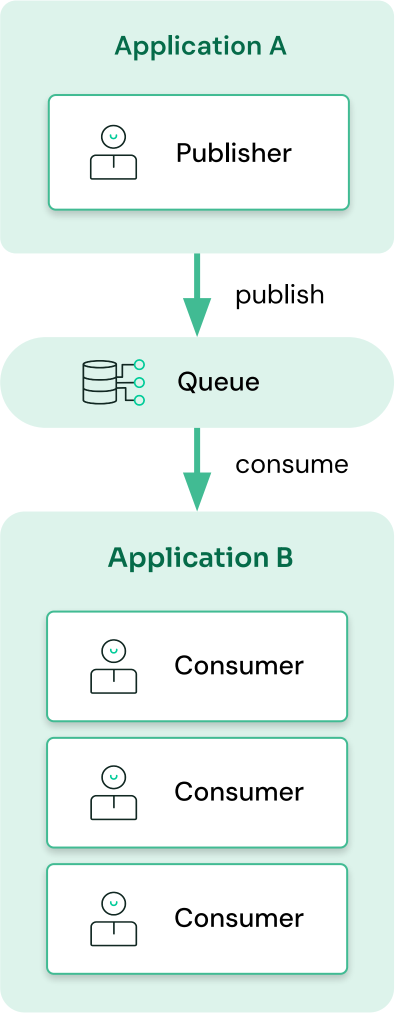
Active MQ as a JMS supporting messaging Broker

Key Components of ActiveMQ:

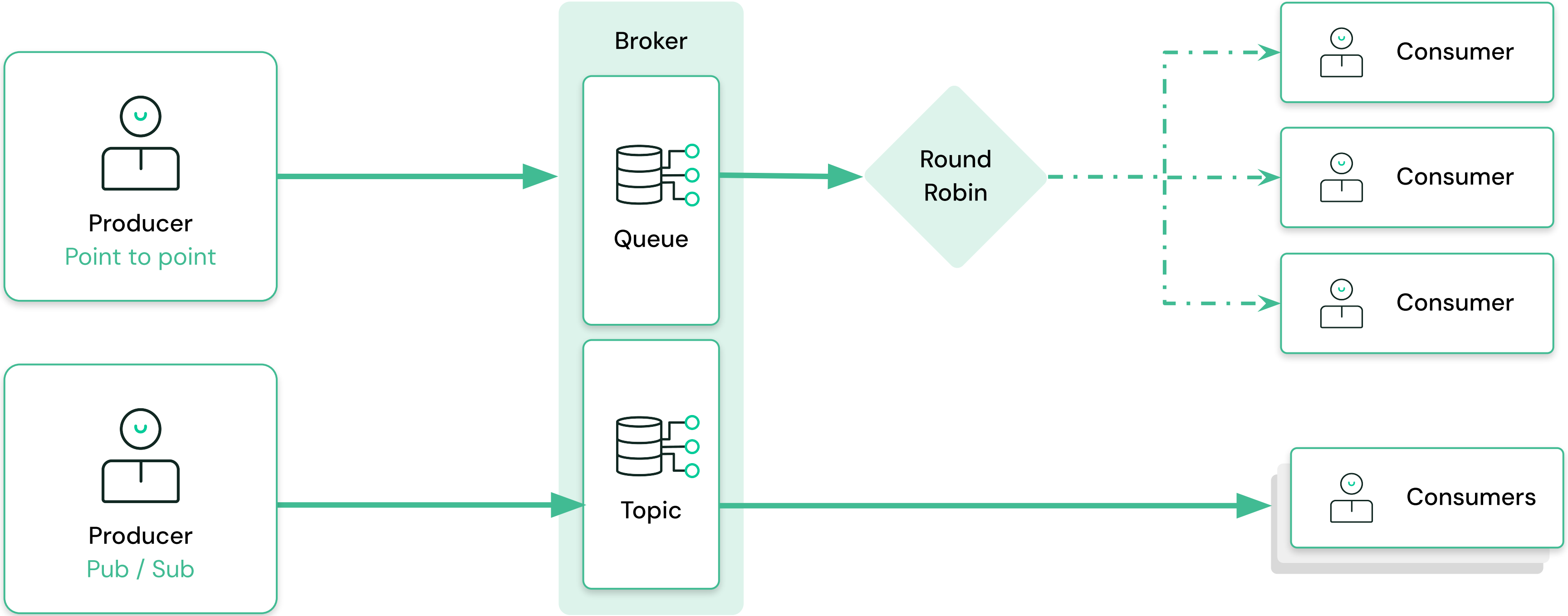
- **Message Broker:**
  ActiveMQ serves as a message-oriented middleware, facilitating communication
  between various components of a distributed system through messaging.

- **JMS (Java Message Service):**
  ActiveMQ adheres to the JMS standard, providing a reliable and standardized way for Java applications to produce and consume messages.

- **Queue and Topic Support:**
  It supports both queues for point-to-point communication and topics for publish-subscribe models, allowing users to choose the messaging paradigm that suits their application's needs.



Active MQ - detailed overview



Active MQ - Message overview

Header	Properties	Messagebody
JMSExpires	JMSXUserID	
JMSMessageID	JMSXAppID	“Payload for consumer”
JMSTimeStamp	JMSXDeliveryCount	
…	…	

Active MQ - Transactional support

ActiveMQ has strong built-in support for **transactional messaging,** which is particularly useful for scenarios that require guaranteed message delivery with full **"exactly once" semantics.**

JMS Transactions: ActiveMQ, being a JMS-compliant broker, supports **JMS transactions.** A JMS transaction groups a set of message operations (sending and/or receiving) into a single atomic unit. Either all operations within the transaction are successfully completed, or none are (in the case of failure). This ensures **atomicity.**

Active MQ - Durability

Durability in ActiveMQ ensures that messages are not lost, even if the broker, producers, or consumers experience failures or go offline.

Persistent Messaging

- **Queue Persistence:** Messages sent to a queue are persisted in ActiveMQ until they are consumed.
- **Topic Durability:** For topics (publish/subscribe model), ActiveMQ supports **durable subscriptions.** This ensures that subscribers receive messages even if they were disconnected or offline when the messages were originally sent.

Broker-Level Durability

Active MQ - Non-Destructive Queues

- ActiveMQ standard behavior -When the consumer acknowledges the messages, the messages are removed from the queue.

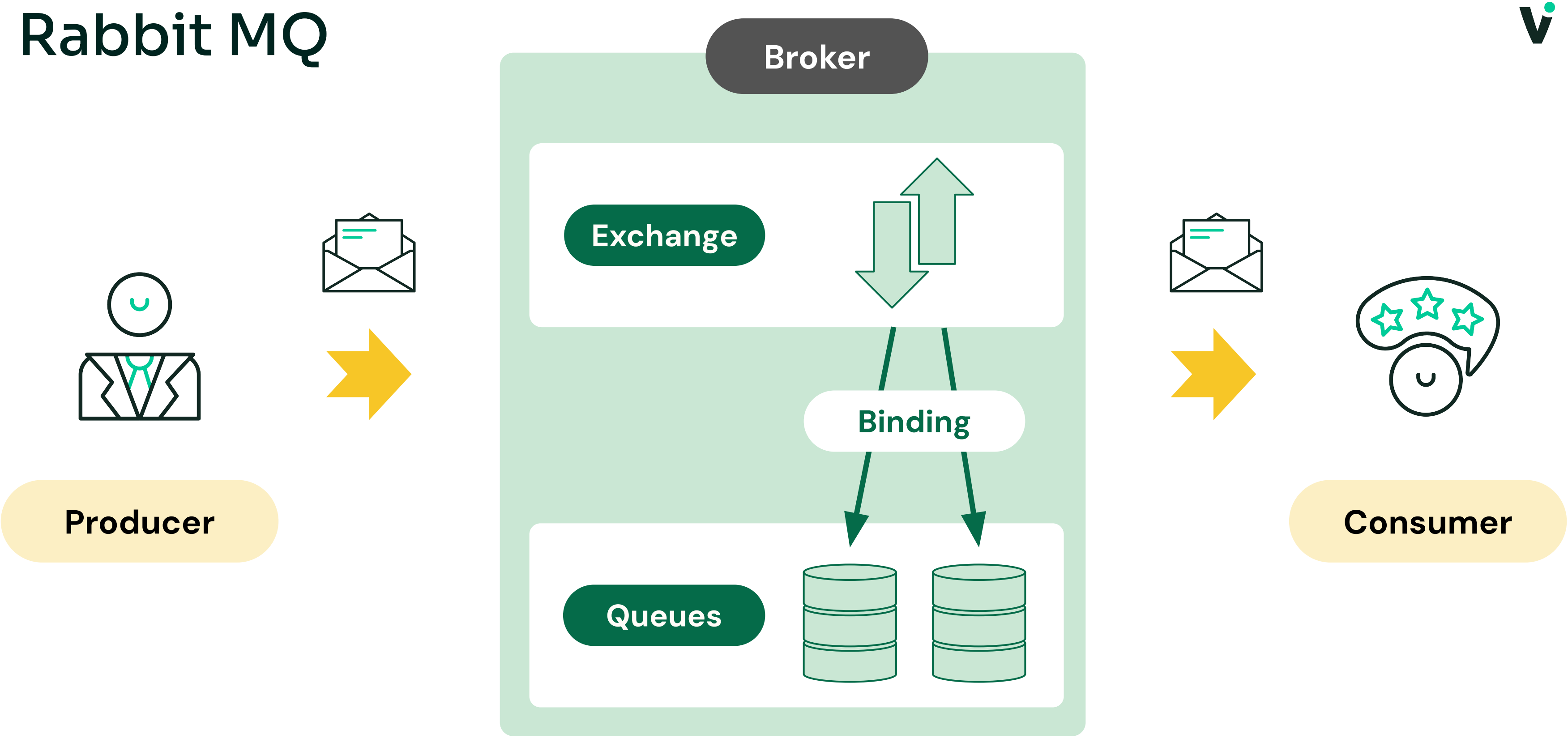
- In cases where messages are not removed when read, then instances of such queues are non-destructible queues.

viesure
innovation center

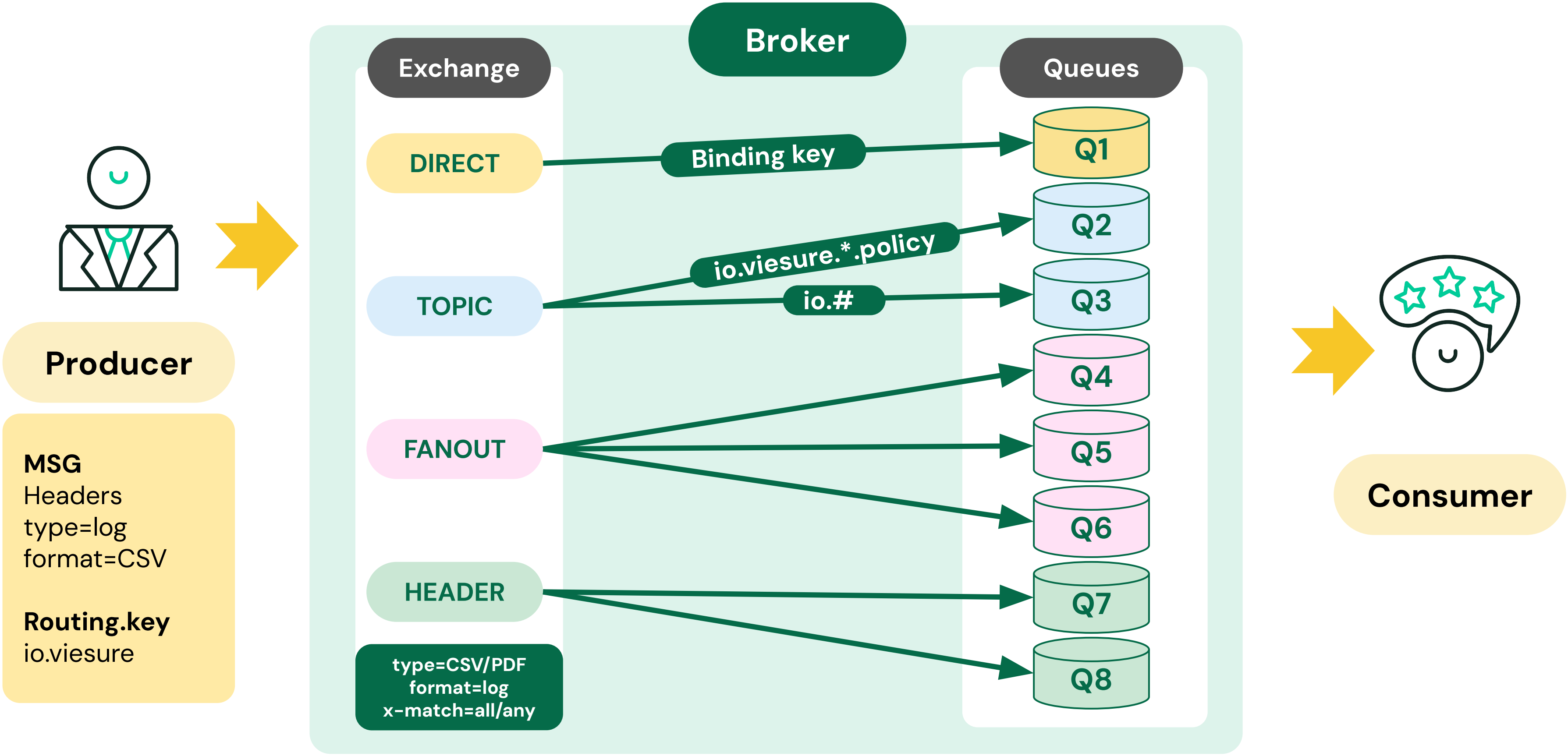
Active MQ – Example 😎

Rabbit MQ 🐰

Rabbit MQ



Types of Exchange



Producer

MSG
Headers
type=log
format=CSV

Routing.key
io.viesure

Broker

Exchange

DIRECT

TOPIC

FANOUT

HEADER

type=CSV/PDF
format=log
x-match=all/any

Binding key

io.viesure.*.policy

io.#

Queues

Q1

Q2

Q3

Q4

Q5

Q6

Q7

Q8

Consumer

RabbitMQ Transactional support

- RabbitMQ supports transactional messaging using **AMQP transactions**, where you can group a series of message-publishing or acknowledgment operations into a single unit of work.

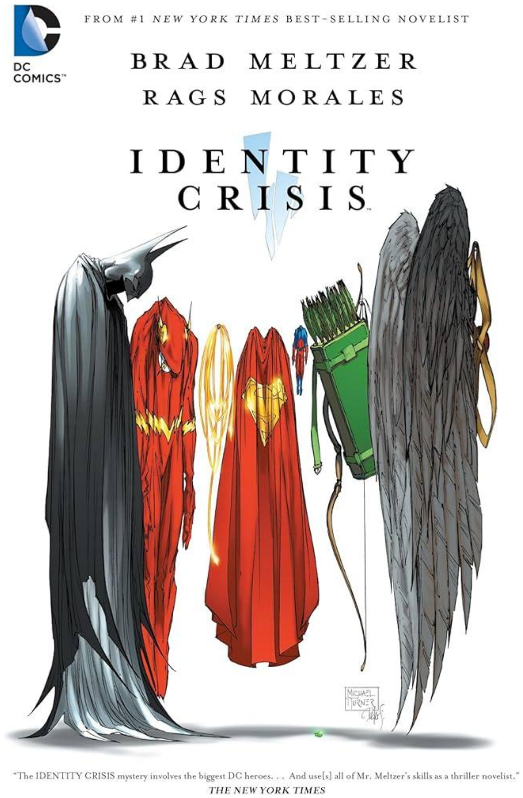
- Problem – RabbitMQ transactions introduce significant overhead because every message in a transaction requires disk writes and locks to ensure atomicity.

- Preferred approach – **Publisher Confirms**
- It provides **asynchronous acknowledgment** from the broker, ensuring that messages have been received and persisted by the broker without the overhead of full transactions

RabbitMQ Non-Destructive queues

Streams! – RabbitMQ goes Kafka? :)

- Streams differ from queues in two important ways: how messages are stored and consumed.
- Streams model an append-only log of messages that can be repeatedly read until they expire. Streams are always persistent and replicated.
- One or more consumers subscribe to it and read the same messages as many times as they want.


DC COMICS
FROM #1 NEW YORK TIMES BEST-SELLING NOVELIST
BRAD MELTZER
RAGS MORALES
IDENTITY CRISIS
"The IDENTITY CRISIS mystery involves the biggest DC heroes. . . And use[s] all of Mr. Meltzer's skills as a thriller novelist."
THE NEW YORK TIMES

Rabbit MQ – *Example* 😎

Comparison ⚖️

ActiveMQ vs RabbitMQ - Comparison

	Active MQ	Rabbit MQ
Protocol	**JMS**, AMQP, STOMP, **OpenWire**, MQTT	AMQP, MQTT, STOMP, HTTP
Architecture	Queue/Topic-based (P2P, Pub/Sub)	Exchange-based (direct, topic, etc.)
Message Handling	Simpler acknowledgment models, JMS transactions	Granular control, manual acknowledgments
Performance	Sufficient for enterprise, higher latency	High throughput, low latency
Use Cases	Enterprise integration, JMS-based system	Microservices, real-time systems
Routing	Simple routing (queue, topic)	Advanced routing (exchanges, routing patterns)
Setup	Easier setup, especially for JMS	More configuration, flexible
Monitoring	JMX support, basic web console	Rich UI, plugins, and tools

When is ActiveMQ preferred

- ✓ Enterprise Integration and Legacy Systems
- ✓ Transactional Messaging
- ✓ Durable Pub / Sub
- ✓ Simpler Routing and Setup
- ✓ Transactional Enterprise Systems
- ✓ Integration with Legacy Systems

When is RabbitMQ preferred

- ✓ Complex routing
- ✓ High Throughput and Real-Time Processing
- ✓ Microservices Communication
- ✓ Advanced Control over Message Delivery
- ✓ Cross Platform Integration
- ✓ Cloud-Native and Containerized Applications

Exploring the horizons 🚀

Exploring the Horizons | Preferring something else

When to look for other solutions:

- If you require **Kafka-style** distributed event streaming, with high-volume event storage and processing, then **Apache Kafka** might be a better fit. Kafka is built for large-scale, real-time data streaming.

- If your application demands **serverless** messaging, consider **AWS SQS**, **Google Pub/Sub**, or **Azure Service Bus**, which offer managed messaging services with minimal infrastructure concerns.

Exploring the Horizons - preferring something else

Kafka:

- Kafka - Event Streaming
- Event-driven microservices
- High Throughput and Low Latency, scalability, etc.

AWS SQS:

- Cloud-native apps
- Simple queueing
- Fully managed by AWS

Google Pub/Sub:

- Cloud-native apps
- Event Driven
- Fully Managed by GC

Reading List

- RabbitMQ Docs https://www.rabbitmq.com/docs
- Cloud AMQP blog: https://www.cloudamqp.com/blog/index.html
- Active MQ: https://activemq.apache.org/components/classic/documentation/features
- JMS: https://medium.com/@gaganjain9319/jms-java-message-service-detailed-explanation-50bc5ba6e3ef

Time for questions 🙋

viesure
innovation center

Thank you!

#creatingtomorrow