# Introduction to Message queues

AARHUS UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

## What is a message queue?

- A message queue is a component of messaging middleware solutions that enables independent applications and services to exchange information.
- Message queues store “messages” (packets of data) that applications create for other applications to consume—in the order they are transmitted until the consuming application can process them.
- This enables messages to wait safely until the receiving application is ready, so if there is a problem with the network or receiving application, the messages in the message queue are not lost.
- This is also known as **asynchronous messaging.**

## Why use message queues?

- Many modern cloud architectures are a collection of loosely coupled microservices that communicate via a message queue.

- One of the advantages of a message queue is that **it makes your data temporarily persistent**, ***reducing the chance of errors*** that may occur when different parts of the system are offline.

## The role of a message queue in a microservice architecture

- Think of a message broker like a delivery person who takes mail from a sender and delivers it to the correct destination.
- In a microservice architecture, there typically are cross-dependencies that mean no single service can perform without getting help from other services.
- This is where it is crucial for systems to have a mechanism in place, allowing services to keep in touch with each other with no blocked responses.
- Message queuing fulfils this purpose by providing a means for services to push messages to a queue asynchronously and ensure they are delivered to the correct destination.
- To implement message queuing, a message broker is required.

## Benefits

- **Reliable message delivery**
  - Using a message queue can ensure that business-critical messages between applications will not be lost and that they will be only be delivered to the recipient once.
- **Inter-application connectivity**
  - Some message queue solutions can handle message encryption, transactionality, and other communication aspects between applications and services.
  - This simplifies application development and enables disparate architectures to work together.
- **Resilience**
  - Asynchronous messaging ensures application-specific faults won’t impact the system.
  - If one component in the system stalls, all others can continue interacting with the queue and processing messages.

## Benefits

- **Versatility**
  - Message queue solutions can support multiple languages, such as Java, Node.js, COBOL, C/C++, Go, .NET, Python, Ruby, and C#.
  - They can also support numerous application programing interfaces (APIs) and protocols, including MQTT, AMQP, and REST, as well as others.
- **Improved security**
  - A message queue may be able to identify and authenticate all messages, and in some message queue solutions, they can be set to encrypt messages at rest, in transit, or end-to-end.
  - This can contribute to the overall security of the applications and/or infrastructure.
- **Integrated file transfers**
  - Some message queue solutions include additional features, such as the ability to transfer files.
  - This can be used as an alternative to FTP within enterprises where such solutions are in use.

## Message queue vs. pub/sub

- Message queues use a point-to-point messaging pattern, in which one application (called the sender) submits a message to the queue and another application (called the receiver) gets the message from the queue and consumes it.
  - There should be a tightly coupled one-to-one relationship between the sender and consumer, **and each message should be consumed only once**.
  - If your applications require messages to be distributed to multiple parties, either multiple message queues can be combined or a publish/subscribe (pub/sub) messaging model can be used.
- In pub/sub messaging, the application producing the message is called a publisher and applications consuming it are the subscribers. Each message is published to a topic, and every application that subscribes to that topic gets a copy of all messages published to it.

## Message queue vs. message bus

- A message bus allows services ubiquitous access to data while ensuring they remain decoupled and independently functional within a distributed system architecture.
- When you employ a message bus, all the services or applications must share common data types, a common command set, and common communication protocols (although, they may be written in different languages).
  - Consumers can determine how they use messages.
- If decoupled applications are to communicate via a message bus, the messages must be transformed so that they are all of the same type.
- In contrast, **message queues transport messages, no matter whether they are of the same or different types**.

## Message queue vs. web services

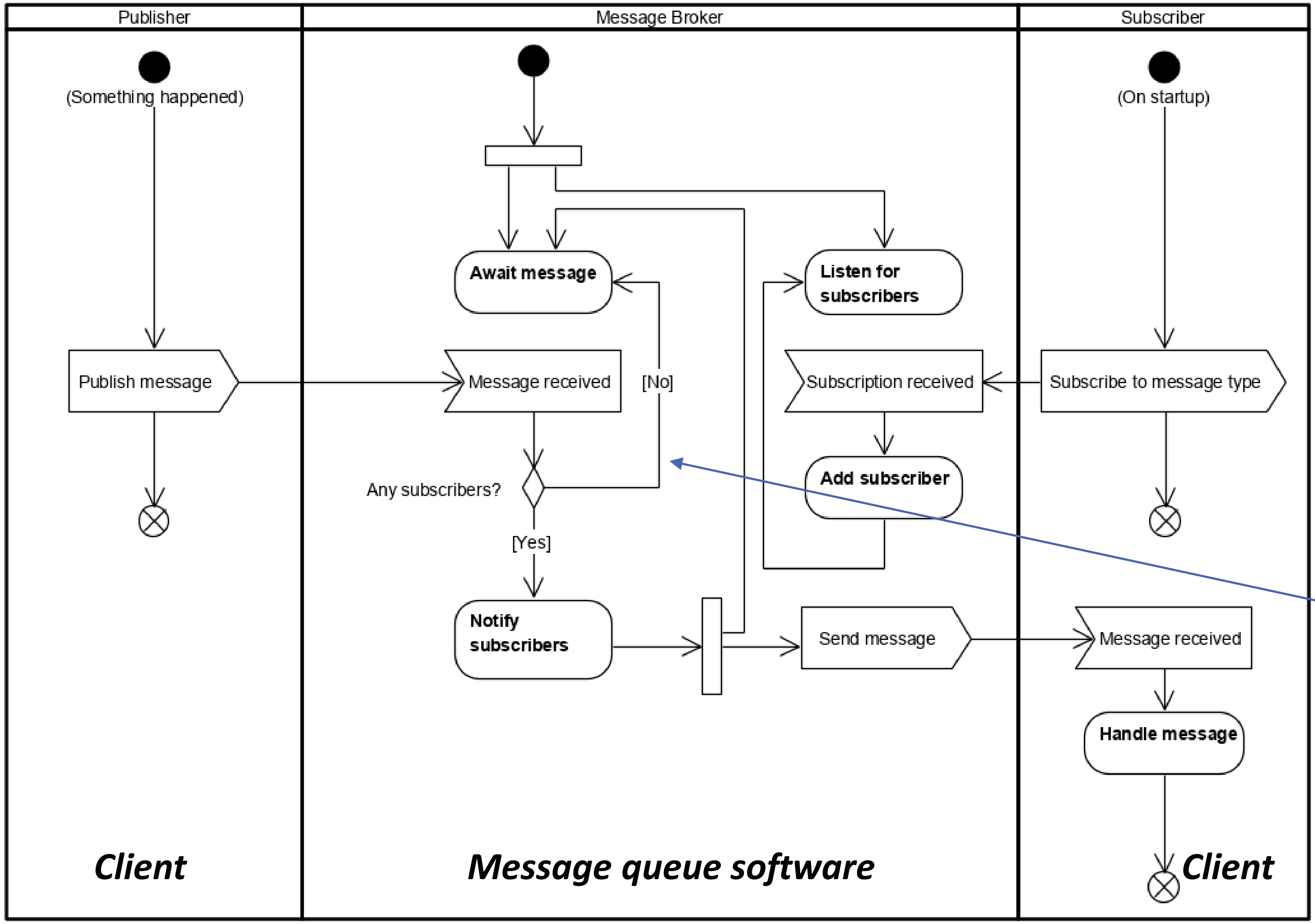
- Applications can communicate directly through web services.
- Web services are widely used in distributed systems, relatively simple, and easy to implement, making them a viable alternative to message queues in certain use cases and scenarios.
- But web services cannot guarantee message delivery.
  - If the server or connection fails, you must build the capability to handle the error within the client.
- Web services also lack pub/sub-distribution models.
- Messaging middleware offers greater fault tolerance and better ability to handle heavy traffic or activity bursts.

## Message queue vs. databases

- Databases can be used as an alternative to message queues in certain situations, but most of the time they serve different purposes and are not readily interchangeable.
- Databases are most commonly used for storage, and they allow you to access the same information over and over again.

- Message queues cannot be used for storage purposes. Once a message has been consumed, it is deleted from the queue.

# Message Broker service



If no subscribers then store the message in a queue

## Examples of Message Queue Software

- RabbitMQ
- Apache Kafka
- Apache ActiveMQ
- Amazon MQ
- IBM MQ
- Azure Queue Storage
- Google Cloud Pub/Sub
- MuleSoft Anypoint Platform
- ZeroMQ
- KubeMQ

## References & Links

- Message Queues
  https://www.ibm.com/cloud/learn/message-queues
- **Using Apache Kafka with .NET**
  https://www.red-gate.com/simple-talk/dotnet/net-development/using-apache-kafka-with-net