# Introduction to Message Queues

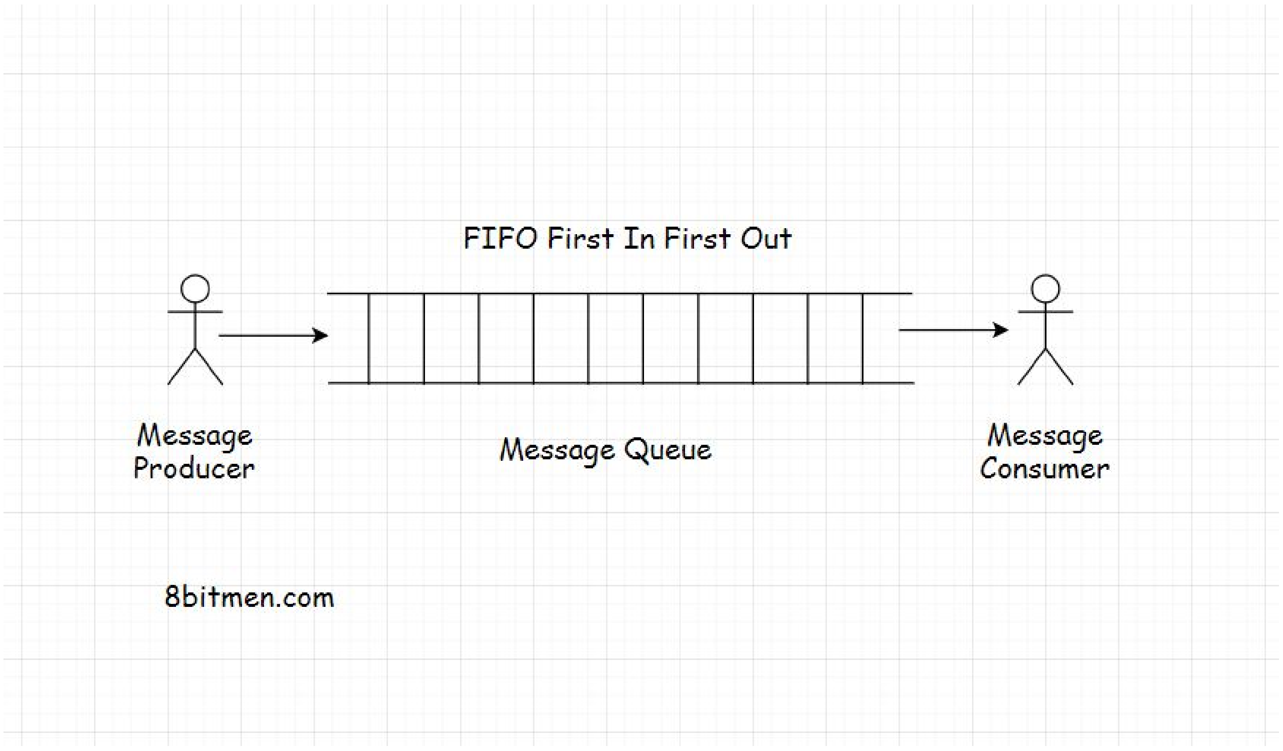
In this lesson, you will learn about the message queues and their functionalities.

**We'll cover the following**

## What is a message queue?#

> Message queue, as the name says, is a *queue* that routes messages from the source to the destination, or from the sender to the receiver.

Since it is a *queue*, it follows the *FIFO (First in First Out)* policy. The message that is sent first is delivered first. Although, messages do have a priority attached to them that makes the queue, a *priority queue* but for now, let's keep things simple.



## Features of a message queue#

Message queues facilitate asynchronous behavior. You have already learned what asynchronous behavior is in the *AJAX* lesson. Asynchronous behavior allows the modules to communicate with each other in the background without hindering their primary tasks.

You will understand the behavior of message queues with the help of an example in a short while. For now, let’s take a quick look at the features of the message queues.

Message queues facilitate *cross-module communication*, which is key in *service-oriented* and *microservices* architecture. It allows communication in a heterogeneous environment. They also provide temporary storage for storing messages until they are processed and consumed by the consumer.

## Real-world example of a message queue#

Take email for example. Both the sender and receiver of the email don’t have to be online at the same time to communicate with each other. The sender sends an email, and the message is temporarily stored on the message server until the recipient comes online and reads the message.

Message queues enable us to run background processes, tasks, and batch jobs. Speaking of background processes, you’ll understand this better with the help of a use case.

Think of a user signing up on a portal. After they sign up, they are immediately allowed to navigate to the application’s homepage, but the sign-up process isn’t complete yet. The system has to send a confirmation email to the user’s registered email ID. Then, the user has to click on the confirmation email for the confirmation of the sign-up event.

However, the website cannot keep the user waiting until it sends the email to the user. Either they are allowed to navigate to the home page, or they bounce off. So, this task is assigned as an asynchronous background process to a message queue. It sends an email to the user for confirmation while the user continues to browse the website.

This is how a message queue can be used to add asynchronous behavior to a web application. Message queues are also used to implement notification systems just like Facebook notifications. I’ll discuss this in the upcoming lessons.

## Message queue in running

## batch jobs#

Now on to the batch jobs. Do you remember the scenario from the previous caching lesson where I discussed how I used the cache to cut down the application deployment costs?

The batch job, which updated the stock prices at regular intervals in the database, was run by a message queue.

By now, I am sure you have an idea of what a message queue is and why we use it in applications.

Now, you also understand that there is a *queue*, there is a message sender called the *producer*, and there is a message receiver called the *consumer*.

Both the *producer* and *consumer* don't have to reside on the same machine to communicate. This is pretty obvious.

While routing messages through the *queue*, we can define several rules based on our business requirements. Adding priority to the messages is one I pointed out. Other important features of queuing include message acknowledgments, retrying failed messages, etc.

Speaking of the size of the queue, there is no definite size, and it can be an infinite buffer, depending on the business's infrastructure.

Now, we'll look into the messaging models widely used in the industry, beginning with the *publish-subscribe* message routing model, which is pretty popular in today's online universe and is how we consume information at large.

← Back
Caching Quiz

Next →
Publish-Subscribe Model

✓ Mark as Completed

Report an Issue