Messaging and Queuing

In the coffee shop, there are cashiers taking orders from the customers and baristas making the orders. Currently, the cashier takes the order, writes it down with a pen and paper, and delivers this order to the barista.

The barista then takes the paper and makes the order. When the next order comes in, the process repeats.

This works great as long as both the cashier and the barista are in sync. But what would happen if the cashier took the order and turned to pass it to the barista and the barista was out on break or busy with another order? Well, that cashier is stuck until the barista is ready to take the order. And at a certain point, the order will probably be dropped so the cashier can go serve the next customer.

This idea of **placing messages into a buffer is called messaging and queuing**. Just as our cashier sends orders to the barista, applications send messages to each other to communicate. If applications **communicate directly** like our cashier and barista previously, this is called being **tightly coupled**.

Monolithic applications and microservices :

Applications are made of multiple components. The components communicate with each other to transmit data, fulfil requests, and keep the application running.

Suppose that you have an application with tightly coupled components. These components might include databases, servers, the user interface, business logic, and so on. This type of architecture can be considered a **monolithic application**.

**In a microservices approach, application components are loosely coupled**. In this case, if a single component fails, the other components continue to work because they are communicating with each other.

The loose coupling prevents the entire application from failing.

When designing applications on AWS, you can **take a microservices approach** with services and components that fulfil different functions. **Two services facilitate application integration**: Amazon Simple Notification Service (Amazon SNS) and Amazon Simple Queue Service (Amazon SQS).

Amazon Simple Queue Service (Amazon SQS)

Amazon Simple Queue Service (Amazon SQS) is a **message queuing service**.

Using Amazon SQS, you can **send, store, and receive messages between software components**, without losing messages or requiring other services to be available. In Amazon SQS, an application sends messages into a queue. A user or service retrieves a message from the queue, processes it, and then deletes it from the queue.

SQS allows you to send, store, and receive messages between software components at any volume. This is without losing messages or requiring other services to be available. Think of messages as our coffee orders and the order board as an SQS queue. Messages have the person's name, coffee order, and time they ordered.

The **data contained within a message is called a payload, and it's protected until delivery**. SQS queues are where messages are placed until they are processed. AWS manages the underlying infrastructure for you to host those queues.

These scale automatically, are reliable, and are easy to configure and use.