AWS LAMBDA

AWS Lambda Integration Patterns with SQS and SNS

Asynchronous communication in AWS lets different parts of a system talk to each other without having to wait for a response, making the system more efficient and able to handle more tasks at the same time.

SQS (Simple Queue Service) and **SNS** (Simple Notification Service) are widely used for managing asynchronous communication in AWS.

SQS holds messages until they're ready to be processed, while SNS broadcasts messages to multiple places instantly. This setup helps keep the system responsive and flexible.

While using SQS, it's also important to understand the existence of SQS FIFO, which is designed for specific use cases.

How they differ from each other:

Feature	SQS Standard	SQS FIFO
Message Order	No guaranteed order	Messages are processed in the exact order they are sent (first-in, first-out)
Duplicates	Allows potential duplicates	Exactly-once processing to avoid duplicates
Throughput	Higher throughput, scales more easily	Lower throughput due to ordering and exactly-once processing constraints
Use Case	Best for large volumes of messages where order doesn't matter (e.g., **background tasks**)	Ideal for scenarios where the exact order of operations is crucial (e.g., **financial transactions**)

In short,

SQS is ideal for general messaging where high throughput is needed, and strict order isn't critical.

SQS FIFO is better suited for applications where it's important that messages are processed in the exact order they are sent, and no duplicates are allowed.

In this context, you should also be aware of the **DLQ** 💀

and ensure that problematic messages can be addressed without affecting the overall system.

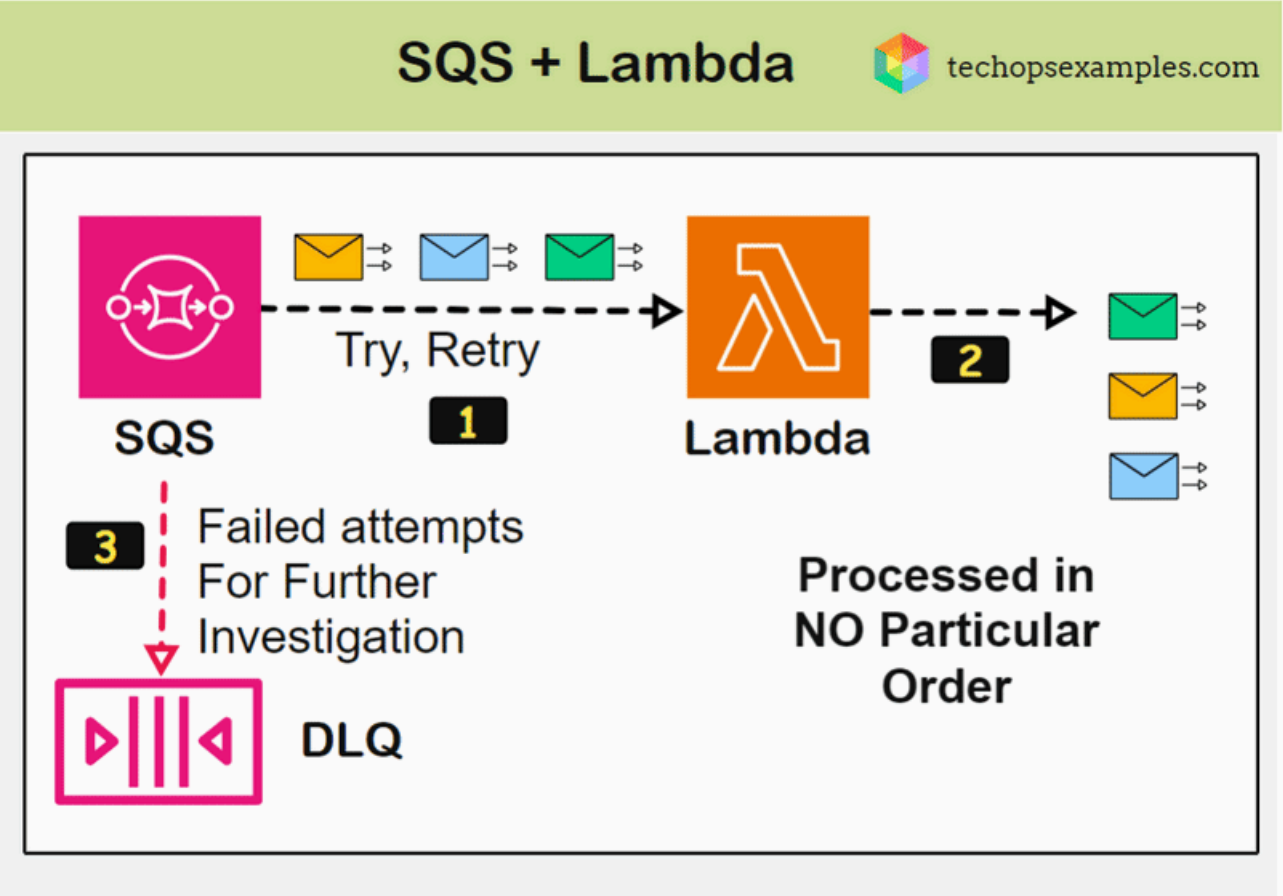
Let's look at how Lambda can be integrated for asynchronous communication.

Here are three main patterns we'll cover:

1. SQS + Lambda
2. SQS FIFO + Lambda
3. SNS + Lambda

Here's how each pattern works:

1. SQS + Lambda :



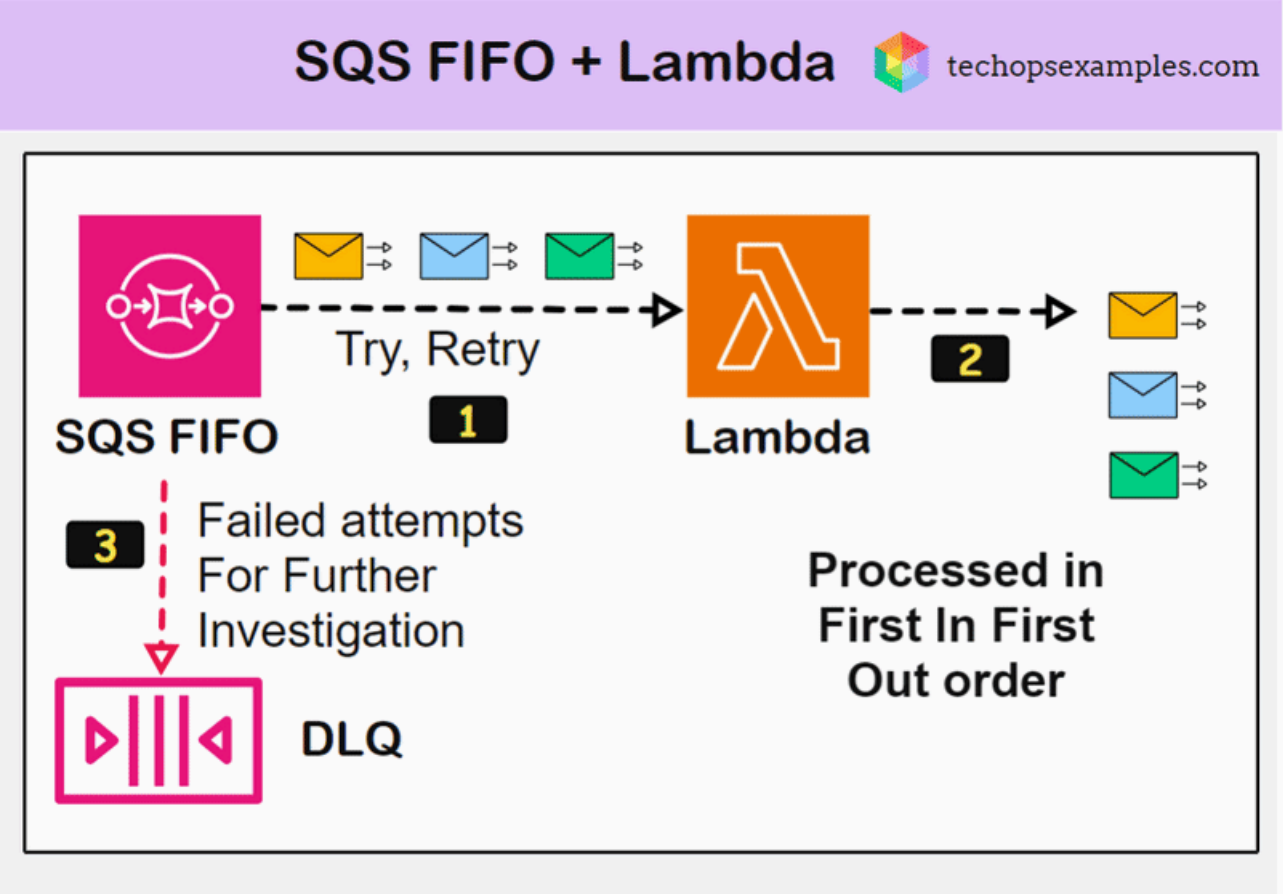
Step 1: SQS stores messages in a queue, and Lambda checks the queue to process each message.

Step 2: If something goes wrong during processing, SQS retries the message.

Step 3: If the message keeps failing, it gets moved to a Dead Letter Queue (DLQ) for further investigation.

> 💡 **This pattern is particularly beneficial for processing user uploads or**

2. SQS FIFO + Lambda :



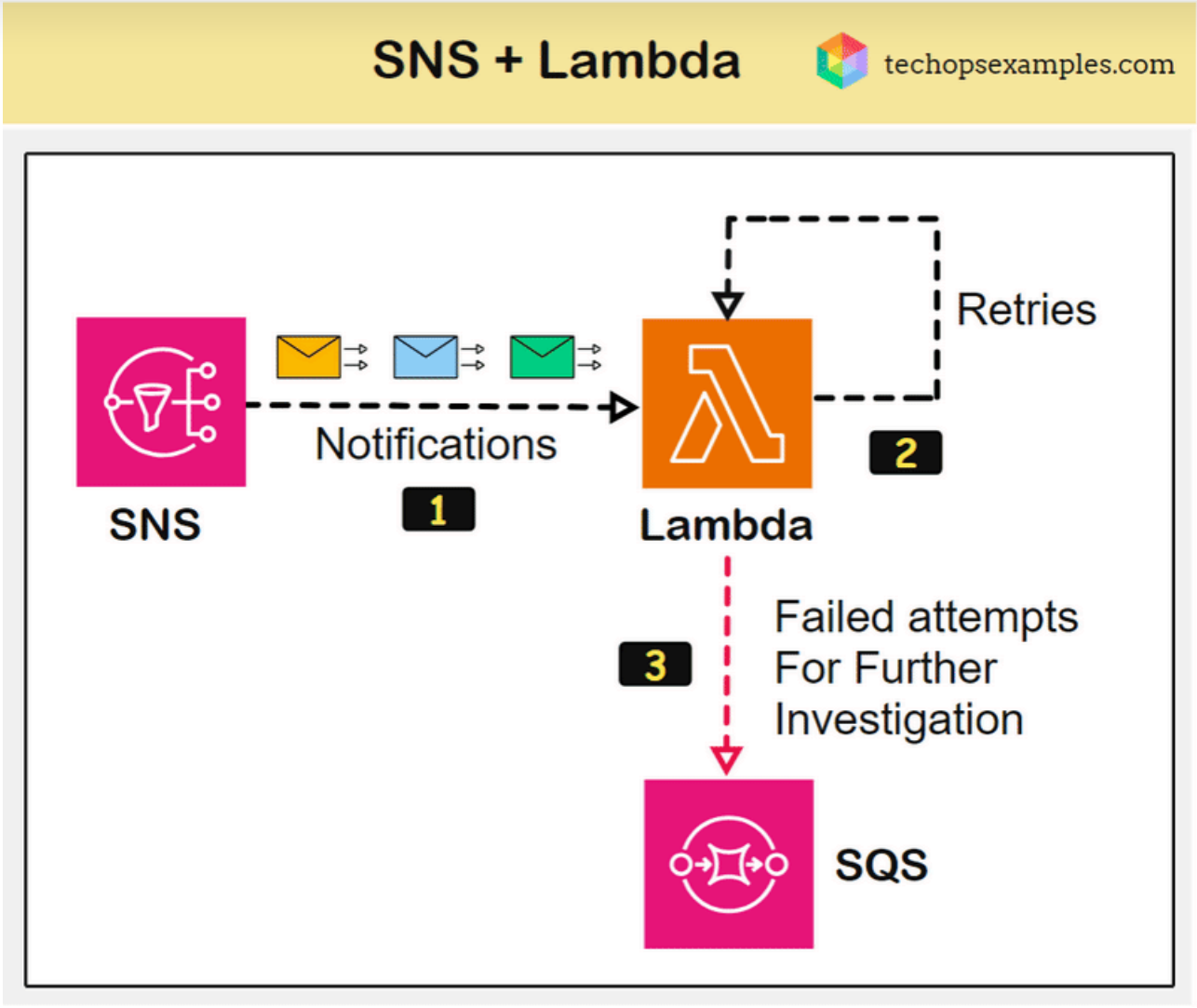
Step 1: SQS FIFO makes sure messages are processed in the exact order they were sent.

Step 2: Lambda processes each message one by one in that order.

Step 3: If a message fails after several attempts, it is moved to a DLQ for further action.

> 💡 **This pattern is vital for applications like e-commerce order processing, where processing orders in the exact sequence they are received is essential to maintaining consistency and accuracy.**

3. SNS + Lambda :



Step 1: SNS sends notifications to Lambda function(s) as soon as something happens.

Step 2: The Lambda functions handle these notifications in real-time.

Step 3: If something goes wrong, Lambda retries the task, and if it still fails, the message is sent to an SQS queue or DLQ for further action.

> 💡 **This pattern is especially effective for real-time monitoring systems, where immediate notification and processing are crucial, such as alerting multiple teams about a system failure.**

I believe now you've got how to choose the right Lambda pattern for your needs!

Tool Of The Day