Understanding Cloud-Based Messaging on AWS

A Beginner's Guide to SNS, SQS, and EventBridge

FIL

🤔 What Problem Are We Solving?

The Challenge of Monolithic Applications

Imagine one giant application where every component is tightly connected:

- A single bug in one part (e.g., payment processing) can bring down the entire system (e.g., user login, product search)
- Scaling one small feature means scaling the whole application, which is expensive and inefficient
- Making updates is slow and risky

The Solution: Decoupling with Messaging Services!

We break down the application into smaller, independent services that communicate with each other through messages instead of direct calls. This is where services like **SNS, SQS, and EventBridge** come in.

Introducing the Services

Let's meet the three core AWS services for building decoupled applications:

AWS SQS (Simple Queue Service)

A message queue. It holds messages from a producer service until a consumer service is ready to process them. **Think of it as a to-do list for your application.**

AWS SNS (Simple Notification Service)

A publish/subscribe (pub/sub) messaging service. It sends messages to multiple subscribers at once. **Think of it as a broadcast or a newsletter.**

AWS EventBridge

An event bus. It receives events from various sources (your apps, AWS services, etc.) and routes them to specific targets based on rules you define. **Think of it as an intelligent switchboard for events.**

📬 Deep Dive: AWS SQS

What is it?

SQS is a fully managed message queuing service that lets you decouple and scale microservices, distributed systems, and serverless applications.

How it Works:

- **Producer:** An application component sends a message to an SQS queue
- **SQS Queue:** The queue stores the message reliably
- **Consumer:** Another component polls the queue, retrieves the message, processes it, and then deletes it from the queue

Key Features:

- **Reliability:** Messages are stored across multiple servers
- **Prevents Message Loss:** The consumer must explicitly delete the message after processing
- **Standard:** High throughput, at-least-once delivery, best-effort ordering
- **FIFO:** Guarantees message order and exactly-once processing (crucial for financial transactions)

📢 Deep Dive: AWS SNS

What is it?

SNS is a pub/sub service that allows you to send notifications to a large number of subscribers.

How it Works:

- **Publisher:** An application publishes a message to an SNS Topic
- **SNS Topic:** A logical access point and communication channel
- **Subscribers:** Endpoints like SQS queues, Lambda functions, or email addresses that receive a copy of the message

Key Features:

- **Fan-out Architecture:** A single message can trigger multiple actions simultaneously
- **Multiple Protocols:** Subscribers can be SQS queues, AWS Lambda, HTTP/S endpoints, email, SMS, and mobile push notifications
- **Message Filtering:** Subscribers can define filter policies to receive only the messages they are interested in

SNS + SQS: The Powerful "Fan-Out" Pattern

This is one of the most common and powerful patterns in cloud architecture.

Scenario: E-commerce Order Placement

An order is placed. We need to:

- Update the inventory service
- Notify the shipping service
- Send a confirmation email to the user

```
                 Order Service
                       ↓
           SNS Topic: "Order Placed"
                  ↙    ↓    ↘
      SQS Queue     SQS Queue     SQS Queue
    (Inventory)    (Shipping)    (Notification)
         ↓             ↓             ↓
      Service       Service       Service
```

Benefit: The Order Service doesn't need to know about the other services. If the notification service is down, inventory and shipping are unaffected.

🌉 Deep Dive: AWS EventBridge

What is it?

EventBridge is a serverless event bus that makes it easy to connect applications together using data from your own applications, integrated SaaS applications, and AWS services.

How it's Different from SNS:

- **Event-Driven:** Designed around events, which are structured JSON objects that represent a change in a system
- **Advanced Filtering & Routing:** Uses rules to evaluate incoming events and routes them to specific targets
- **Many Sources:** Can natively receive events from over 200 AWS services and many SaaS partners

Core Components:

- **Events:** The data describing a change
- **Event Bus:** The router that receives events
- **Rules:** Match incoming events and route them to targets
- **Targets:** The endpoints that process the event (Lambda, SQS, etc.)

When to Use What? A Quick Summary

Service	Use Case	Analogy
SQS	**Task Processing:** When you have a single task that needs to be done reliably by one worker. Decouple a sender and a receiver.	A to-do list for one person.
SNS	**Notifications/Fan-out:** When you need to send the same message to many different subscribers simultaneously.	A broadcast radio station or a group chat announcement.
EventBridge	**Event Routing:** When you need to react to events from various sources and route them intelligently based on their content.	A smart postal service that reads the mail and delivers it to different departments based on what's inside.

🧠 Quick Questions!

Question 1:

You have a microservice that generates invoices. Another microservice processes these invoices to create PDF reports. The reporting service sometimes goes offline for maintenance. Which service would you use to ensure no invoices are lost while the reporting service is down?

Question 2:

A new user signs up for your application. You need to simultaneously trigger three different actions: send a welcome email, add them to a marketing list, and set up their user profile. What is the most efficient way to design this using the services we discussed?

Question 3:

Your application needs to react specifically when an S3 bucket receives a file with a .jpg extension but ignore all other file types. Which service offers the most powerful built-in filtering capabilities to achieve this?

✅ Solutions

Solution 1:

SQS (Simple Queue Service). The invoice service (producer) can place messages in an SQS queue. The reporting service (consumer) can then pull messages from the queue whenever it's online. The queue will hold the messages safely until they are processed and deleted.

Solution 2:

SNS (Simple Notification Service) with the Fan-Out pattern. The user signup service would publish a single "NewUser" message to an SNS topic. Three separate services (email, marketing, profile) would subscribe to this topic (likely via their own SQS queues) and receive the notification to perform their individual tasks.

Solution 3:

EventBridge. EventBridge is the ideal choice here. You can create a rule that listens for events from S3. The rule's event pattern can be configured to match only those events where the object key ends with .jpg, and then route them to a specific target like a Lambda function for processing. This is much more efficient than sending all events and filtering them in your own code.

Thank You

Questions?