

Aftab0045 / AWS-Projects

Code Issues Pull requests Actions Projects Wiki Security

AWS-Projects / AWS-SNS-to-SQS-Message-Delivery /

aftab0045 add 48f1aee · 17 minutes ago

Name	Name	Last commit date
..		
img	add	17 minutes ago
README.md	add	17 minutes ago

README.md

# AWS SNS to SQS Message Delivery Project

## Project Overview

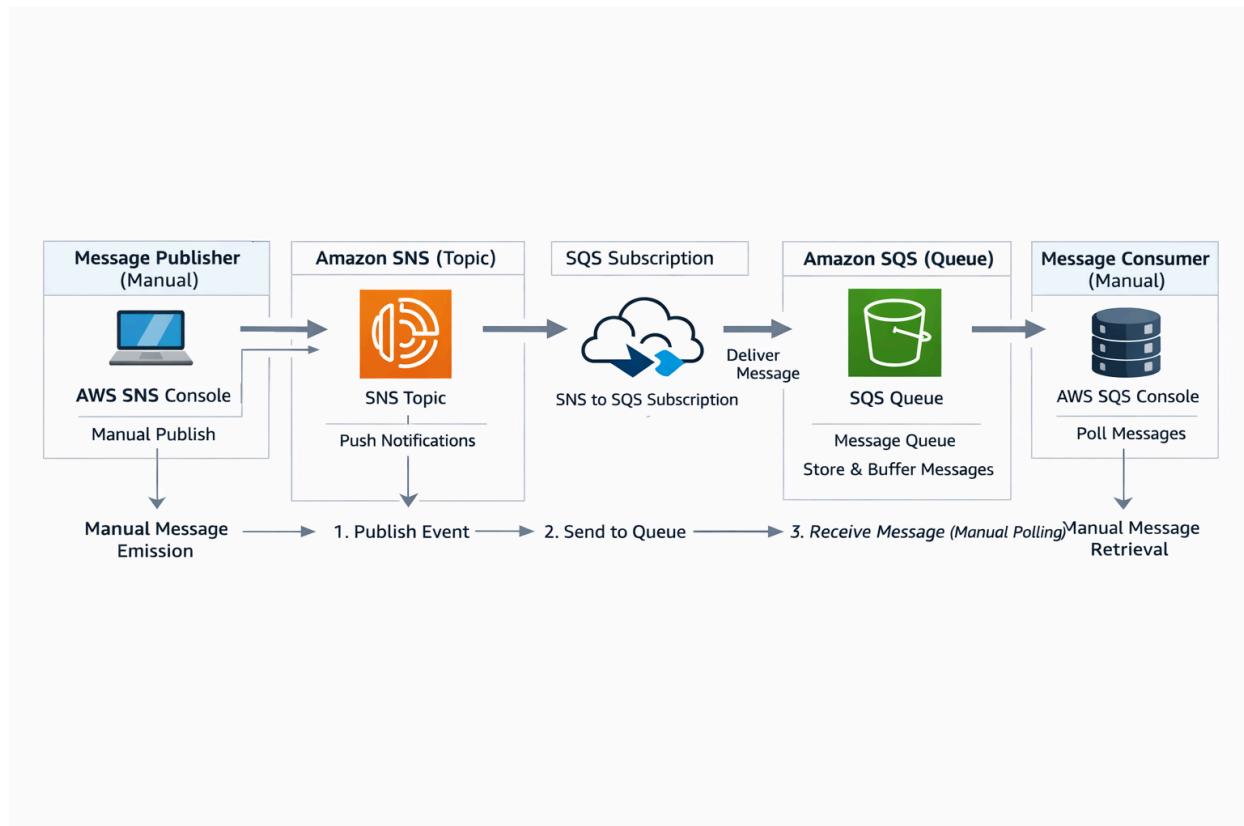
This project demonstrates how to use Amazon Simple Notification Service (SNS) and Amazon Simple Queue Service (SQS) together to build a decoupled, event-driven architecture.

In this setup:

- A message is published to an **SNS Topic**
- The SNS topic delivers the message to an **SQS Queue**
- The message is then available for processing by a consumer

This architecture is commonly used in **microservices**, **event notifications**, and **asynchronous processing systems**.

# Architecture Diagram



## AWS Services Used

- **Amazon SNS** – Message publishing and fan-out service
- **Amazon SQS** – Fully managed message queue
- **AWS IAM** – Permissions (handled automatically by AWS)

## Project Objectives

- Create an SNS topic
- Create an SQS queue
- Subscribe SQS to SNS
- Publish messages to SNS
- Receive messages in SQS
- Understand SNS → SQS message flow

## Architecture Flow

1. Publisher sends a message to SNS
2. SNS receives the message
3. SNS pushes the message to subscribed SQS queue
4. SQS stores the message safely
5. Consumer polls the queue and processes the message

## Step-by-Step Implementation

### Step 1: Create an SQS Queue

1. Open AWS Console
2. Go to Amazon SQS
3. Click **Create queue**
4. Select:
  - Queue type: **Standard**
  - Queue name: **sqs-Demo**
5. Keep default settings
6. Click **Create queue**

 SQS queue created successfully

### Step 2: Create an SNS Topic

1. Go to Amazon SNS
2. Click **Create topic**
3. Select:
  - Type: **Standard**
  - Name: **global**
4. Click **Create topic**

 SNS topic created successfully

### Step 3: Subscribe SQS Queue to SNS Topic

1. Open the **SNS Topic**
2. Go to **Subscriptions**

### 3. Click Create subscription

#### 4. Configure:

- o Protocol: Amazon SQS
- o Endpoint: Select SQS Queue ARN

### 5. Click Create subscription

The screenshot shows the AWS SNS 'Subscription' details page for a specific subscription. The subscription ARN is listed as `arn:aws:sns:us-east-1:003380494838:global:e0675f63-753b-4c78-80bd-7c44aca8f938`. The status is confirmed, and the protocol is set to SQS. The endpoint is `arn:aws:sqs:us-east-1:003380494838:SQS-Demo`. The topic is `global`. The subscription principal is `arn:aws:iam::003380494838:root`. The 'Subscription filter policy' section indicates 'No filter policy configured for this subscription.' A blue 'Edit' button is visible in the top right corner.

SNS and SQS are now connected

## 🛠 Step 4: Verify SQS Access Policy (Important)

SNS must have permission to send messages to SQS.

1. Open Amazon SQS
2. Select your queue
3. Go to Access Policy
4. Ensure policy allows SNS to send messages

Example policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow-SNS-SendMessage",
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "SQS:SendMessage",
      "Resource": "YOUR_SQS_QUEUE_ARN",
    }
  ]
}
```

```

    "Condition": {
        "ArnEquals": {
            "aws:SourceArn": "YOUR_SNS_TOPIC_ARN"
        }
    }
}
]
}

```

## Step 5: Publish a Message to SNS

1. Open SNS Topic
2. Click Publish message
3. Enter:
  - Subject: Order Notification
  - Message:

New order has been placed successfully.



4. Click Publish

Message sent to SNS topic

## Step 6: Receive Message from SQS

1. Open Amazon SQS
2. Select the queue
3. Click Send and receive messages
4. Click Poll for messages

ID	Sent	Size	Receive count
47dd12c1-4bfe-4506-8685-456b18390679	2025-12-26T19:29+05:30	9 bytes	3
e6b46a69-bdfd-4434-80b0-ef955eeb1630	2025-12-26T19:38+05:30	963 bytes	1

 Message published from SNS is now visible in SQS

## Conclusion

This project demonstrates how Amazon SNS and Amazon SQS work together to create a reliable, scalable, and decoupled messaging system. It is a foundational pattern used in many real-world AWS architectures.