

04-11-2025

Task-2

Sri Perumbuduri D M S Satyanarayana

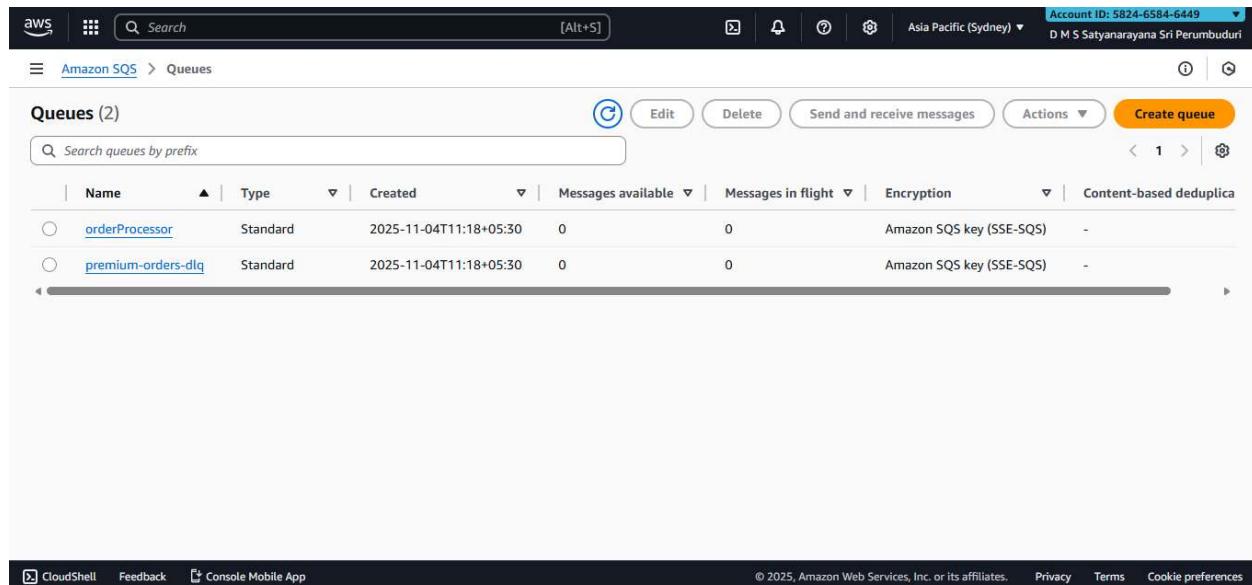
ENC\16800

Objective

Design and implement a real-time data processing pipeline for an e-commerce application that handles order events with the following requirements

Step1: Create an SQS Queue

Use Amazon SQS to create a queue (standard or FIFO depending on message ordering requirements). Standard queues: high throughput, at-least-once delivery. FIFO queues: exactly-once processing, preserve message order.



Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplica
orderProcessor	Standard	2025-11-04T11:18+05:30	0	0	Amazon SQS key (SSE-SQS)	-
premium-orders-dlq	Standard	2025-11-04T11:18+05:30	0	0	Amazon SQS key (SSE-SQS)	-

Step2: Implement DynamoDB Table

Enable DynamoDB streams

The screenshot shows the AWS DynamoDB console with the 'Tables' section selected. A specific table named 'Order' is being viewed. The left sidebar includes links for Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Under 'DAX', there are links for Clusters, Subnet groups, Parameter groups, and Events. At the bottom of the sidebar are CloudShell, Feedback, and Console Mobile App links. The main content area displays the 'Tables (1)' section with a search bar for 'Any tag key' and 'Any tag value', and a 'Find tables' input field. Below this is a table with one item, 'Order'. The 'Order' table details are shown in the right panel, including General information (Partition key: orderId, Sort key: -), Capacity mode (On-demand), Alarms (No active alarms), Table status (Active), and Point-in-time recovery (PITR) (Off). A note about PITR is present: 'Protect your DynamoDB table from accidental writes and deletes. When you turn on point-in-time recovery (PITR), DynamoDB backs up your table data automatically so that you can restore to any given second in the preceding 1 to 35 days. Additional charges apply.' Buttons for Actions, Explore table items, Settings, Indexes, Monitor, Global tables, Backups, and Export are also visible.

Step3: Lambda Function to process message body

The screenshot shows the AWS Lambda console with the 'Functions' section selected. The left sidebar includes links for Dashboard, Applications, Functions, Additional resources (Code signing configurations, Event source mappings, Layers, Replicas), and Related AWS resources (Step Functions state machines). The main content area displays the 'Functions (2)' list. The table shows two functions: 'orderProcessor' and 'premiumOrderProcessor'. Both functions are in Zip package type and Python 3.12 runtime, with last modified times of 10 hours ago. A 'Create function' button is located at the top right of the table. The bottom of the page includes CloudShell, Feedback, and Console Mobile App links, along with standard footer links for Privacy, Terms, and Cookie preferences.

Step4 : Create Event Bridge Pipes

The screenshot shows the Amazon EventBridge Pipes list page. On the left, there's a navigation sidebar with links like Dashboard, Developer resources, Buses, Pipes (selected), and Scheduler. The main area has a header "Pipes (2) Info" with buttons for Edit, Delete, Stop, Start, CloudFormation Template, and Create pipe. A search bar says "Find pipes" and a dropdown says "Any Status". There are two pipes listed:

Name	Status	Source	Enrichment	Target	Last modified
OrdersPipe	Running	Order/stream/2025-11-04T05:46:30.494	--	orderProcessor	Nov 4, 2025, 11:42 AM GMT+5:30
PremiumOrdersPipe	Running	Order/stream/2025-11-04T05:46:30.494	--	premiumOrderProcessor	Nov 4, 2025, 11:47 AM GMT+5:30

Filters in Event Bridge pipes

The screenshot shows the Amazon EventBridge OrdersPipe Filtering tab. The sidebar is identical to the previous one. The main area has tabs for Source, Filtering (selected), and Target. A sub-header "Filtering Info" says "Define an event pattern to filter the events that are sent through the pipe." Below is a code editor with the following JSON event pattern:

```
1 {
2   "eventName": ["INSERT"],
3   "dynamodb": {
4     "NewImage": {
5       "status": {
6         "S": ["pending", "shipped"]
7       },
8       "customerEmail": {
9         "S": [
10           {"S": "anything-but": "test.com"}
11         ]
12       }
13     }
14   }
15 }
```

Lambda Function for processing messages

```

1  def lambda_handler(event, context):
2      # EventBridge Pipes sends a list of records
3      if isinstance(event, list):
4          records = event
5      else:
6          records = [event]
7
8
9
10     for record in records:
11         # Extract DynamoDB NewImage
12         new_image = record.get('dynamodb', {}).get('NewImage', {})
13
14             # Extract fields
15             order_id = new_image.get('orderId', {}).get('S')
16             status = new_image.get('status', {}).get('S')
17             amount_str = new_image.get('amount', {}).get('N')
18             email = new_image.get('customerEmail', {}).get('S')
19
20             # Skip if amount is missing or not numeric
21             try:

```

Table contents

Attributes View DynamoDB JSON

```

1 [{}]
2   "orderId": {
3     "S": "001"
4   },
5   "amount": {
6     "N": "150"
7   },
8   "customerEmail": {
9     "S": "user@example.com"
10 },
11  "status": {
12    "S": "pending"
13 }
14 []

```

Cloud Watch Logs

CloudWatch > Log groups > /aws/lambda/orderProcessor > 2025/11/04/[\${LATEST}]7895939f10fb426dba4600eb63ae5dd4

CloudWatch

- Favorites and recents
- Alarms ▲ 0
- Logs
- Log groups
- Log Anomalies
- Live Tail
- Logs Insights
- Contributor Insights
- Metrics New
- Application Signals New (APM)
- GenAI Observability New
- Network Monitoring
- Insights

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

Filter events - press enter to search

Actions ▾ Start tailing Create metric filter

1m 1h UTC timezone

Display ▾

Timestamp	Message
2025-11-04T06:13:57.237Z	START RequestId: aefa31b4-c258-4fea-b70d-0ccdff8b9848 Version: \${LATEST}
2025-11-04T06:13:57.238Z	Processing order 001 Status=pending Amount=150.0 Email=user@example.com
2025-11-04T06:13:57.246Z	END RequestId: aefa31b4-c258-4fea-b70d-0ccdff8b9848
2025-11-04T06:13:57.247Z	REPORT RequestId: aefa31b4-c258-4fea-b70d-0ccdff8b9848 Duration: 12.38 ms Billed Duration: 90 ms Memory ...
2025-11-04T06:18:05.124Z	START RequestId: 5935b839-d688-42bb-a6e2-abbc5f208fd0 Version: \${LATEST}
2025-11-04T06:18:05.125Z	Processing order 002 Status=pending Amount=2000.0 Email=user@example.com
2025-11-04T06:18:05.144Z	END RequestId: 5935b839-d688-42bb-a6e2-abbc5f208fd0
2025-11-04T06:18:05.144Z	REPORT RequestId: 5935b839-d688-42bb-a6e2-abbc5f208fd0 Duration: 19.69 ms Billed Duration: 20 ms Memory ...

No newer events at this moment. Auto retry paused. [Resume](#)

Back to top ^