

Use case

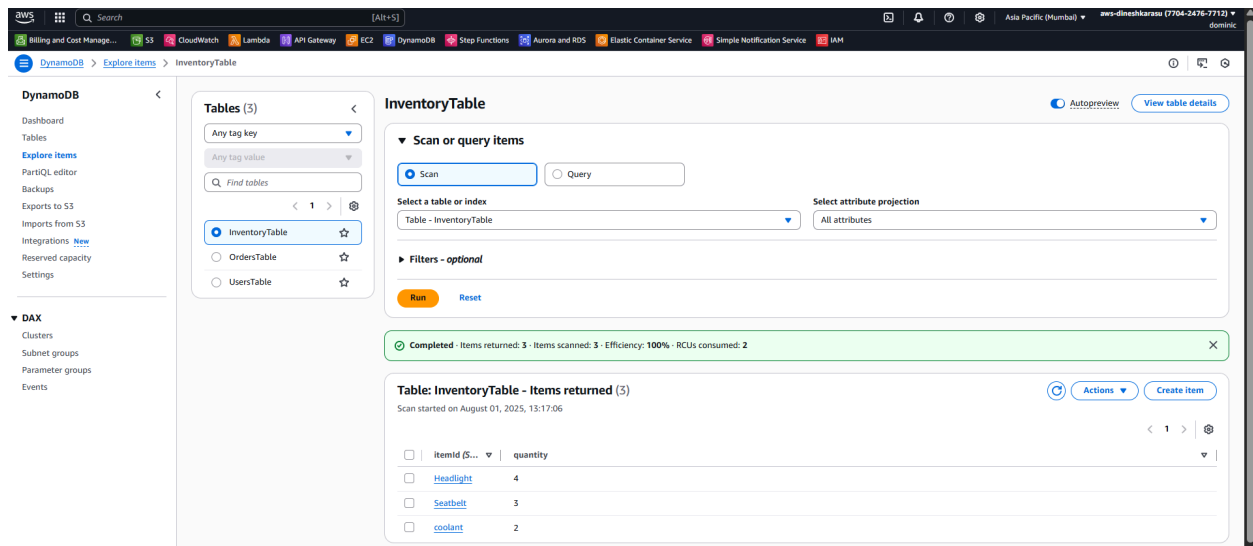
End-to-end microservice deployment with monitoring

Use case Description

Build a serverless microservices-based order management system using AWS Lambda, API Gateway, and DynamoDB. Integrated Step Functions for order workflows and SNS for real-time email notifications on order status updates.

Approach

1. Create Dynamo DB tables for inventory, users and orders
2. Write Lambda codes to
 - scan and check inventory
 - Create new user
 - Create Orders
3. Create step function to orchestrate the workflow
4. Create SNS topic to trigger emails
5. While creating user, read email from the input and subscribe to the topic
6. Create OrderStatus update Lambda to send updates through email notification
7. Create API Gateway endpoints and test through postman



The screenshot shows the AWS Management Console interface for the DynamoDB console. The left sidebar contains navigation links for various AWS services. The main content area displays the 'InventoryTable' details. The 'Scan or query items' section is active, showing the 'Scan' option selected. The 'Filters - optional' section is expanded, and the 'Run' button is clicked. The results are displayed in a table with the following data:

Itemid (S...	quantity
Headlight	4
Seatbelt	3
coolant	2

aws [Alt+S] Asia Pacific (Mumbai) aws-dineshkareem (7704-2476-7712) dominic

DynamoDB > Explore Items > UsersTable

DynamoDB

- Dashboard
- Tables
- Explore items
- PartiQL editor
- Backups
- Exports to S3
- Integrations **New**
- Reserved capacity
- Settings

▼ DAX

- Clusters
- Subnet groups
- Parameter groups
- Events

Tables (3)

Any tag key

Any tag value

Find tables

< 1 >

☐ InventoryTable ☆

☐ OrdersTable ☆

☒ UsersTable ☆

UsersTable

Autopreview View table details

▼ Scan or query items

☒ Scan ☐ Query

Select a table or index: Table - UsersTable

Select attribute projection: All attributes

Filters - optional

Run Reset

Completed - Items returned: 1 - Items scanned: 1 - Efficiency: 100% - RCUs consumed: 2

Table: UsersTable - Items returned (1)

Scan started on August 01, 2025, 12:37:28

Actions Create item

Userid (String)	email	name
467ebf85-c30b-4f17-...	dkarasu@e...	Dinesh

aws [Alt+S] Asia Pacific (Mumbai) aws-dineshkareem (7704-2476-7712) dominic

DynamoDB > Explore Items > OrdersTable

DynamoDB

- Dashboard
- Tables
- Explore items
- PartiQL editor
- Backups
- Exports to S3
- Integrations **New**
- Reserved capacity
- Settings

▼ DAX

- Clusters
- Subnet groups
- Parameter groups
- Events

Tables (3)

Any tag key

Any tag value

Find tables

< 1 >

☐ InventoryTable ☆

☒ OrdersTable ☆

☐ UsersTable ☆

OrdersTable

Autopreview View table details

▼ Scan or query items

☒ Scan ☐ Query

Select a table or index: Table - OrdersTable

Select attribute projection: All attributes

Filters - optional

Run Reset

Completed - Items returned: 1 - Items scanned: 1 - Efficiency: 100% - RCUs consumed: 2

Table: OrdersTable - Items returned (1)

Scan started on August 01, 2025, 13:17:40

Actions Create item

orderid (String)	userid (String)	email	itemid	status
ec6b14fa-d31a-4c85-...	467ebf85-c30b-4f17-aae8-f2...	dkarasu@e...	Headlight	Shipped

aws [Alt+S] Asia Pacific (Mumbai) aws-dineshkareem (7704-2476-7712) dominic

Lambda > Functions > InventoryCheckFunction

InventoryCheckFunction

Throttle Copy ARN Actions

Export to Infrastructure Composer Download

▼ Function overview

Diagram Template

InventoryCheckFunction

Layers (0)

+ Add trigger

+ Add destination

Description

Last modified 19 hours ago

Function ARN arn:aws:lambda:ap-south-1:770424767712:function:InventoryCheckFunction

Function URL Info

Code Test Monitor Configuration Aliases Versions

Code source

Open in Visual Studio Code Upload from

EXPLORER

INVENTORYCHECKFUNCTION

lambda_function.py

```
1 import json
2 import boto3
3
4 dynamodb = boto3.resource('dynamodb')
5 table = dynamodb.Table('InventoryTable')
6
```

OrderService

Function overview

Diagram

OrderService

API Gateway (2)

Code source

```
def update_order_status(event):
    try:
        return {
            'statusCode': 200,
            'body': 'Order status updated successfully'
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': 'Error updating order status: ' + str(e)
        }
```

UserService

Function overview

Diagram

UserService

API Gateway

Code source

```
import json
import boto3
import uuid

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('UserTable')
cors_headers = {
```

Amazon SNS

send-email

Details

Name: send-email

ARN: arn:aws:sns:ap-south-1:770424767712:send-email

Type: Standard

Display name: -

Topic owner: 770424767712

Subscriptions (2)

ID	Endpoint	Status	Protocol
804c8a11-abe0-4271-ad23-1e50af0234c	dkarasu@evoketechtechnologies.com	Confirmed	EMAIL
d7639f33-9db8-4797-9037-2af44cf2ae22	dineshkarasu137@gmail.com	Confirmed	EMAIL

Step Functions

State machines

Execution inspector

Activities

Developer resources

Online learning workshop

Local Development

Data flow simulator

Feature spotlight

Documentation

Join our feedback panel

Executions Monitoring Logging **Definition** Aliases Versions Tags

Definition

```
1 {
2   "Comment": "Check inventory and create order",
3   "StartAt": "CheckInventory",
4   "States": {
5     "CheckInventory": {
6       "Type": "Task",
7       "Resource": "arn:aws:lambda:ap-south-1:770424767712:function:InventoryCheckFunction",
8       "ResultPath": "$.InventoryResult",
9       "Next": "IsAvailable"
10    },
11    "IsAvailable": {
12      "Type": "Choice",
13      "Choices": [
14        {
15          "Variable": "$.InventoryResult.result",
16          "StringEquals": "Available",
17          "Next": "CreateOrder"
18        },
19        {
20          "Default": "OutOfStock"
21        }
22      ],
23      "CreateOrder": {
24        "Type": "Task",
25        "Resource": "arn:aws:lambda:ap-south-1:770424767712:function:OrderService",
26        "End": true
27      },
28      "OutOfStock": {
29        "Type": "Fail",
30        "Error": "ItemUnavailable",
31        "Cause": "The requested item is out of stock"
32      }
33    }
34  }
35 }
```

Start

AWS Lambda: Invoke

CheckInventory

Choice state

IsAvailable

\$.InventoryResult.result == "Available"

AWS Lambda: Invoke

CreateOrder

Fail state

OutOfStock

End

Step Functions

State machines

Execution inspector

Activities

Developer resources

Online learning workshop

Local Development

Data flow simulator

Feature spotlight

Documentation

Join our feedback panel

OrderProcessingStateMachine Execution: 99e39d3a-05b8-4614-a1de-fcf488413ee0

Details Execution input and output Definition

Execution status: Succeeded

Execution type: Standard

Execution ARN: arn:aws:statesap-south-1:770424767712:execution:OrderProcessingStateMachine:99e39d3a-05b8-4614-a1de-fcf488413ee0

IAM role ARN: arn:aws:iam:770424767712:role/state-machine-permissions

State transitions: 5

Start time: Jul 31, 2025, 17:04:21.092 (UTC+05:30)

End time: Jul 31, 2025, 17:04:22.828 (UTC+05:30)

Duration: 00:00:01.736

Alias: -

Version: -

Graph view Table view

Graph view

Start

AWS Lambda: Invoke

CheckInventory

Choice state

IsAvailable

\$.InventoryResult.result == "Available"

AWS Lambda: Invoke

CreateOrder

Fail state

OutOfStock

End

Step details

Choose a step to view its details.

API Gateway

APIs

Custom domain names

Domain name access associations

VPC links

API: MicroservicesAPI (72zh1fadg)

Resources

Stages

Authorizers

Gateway responses

Models

Resource policy

Documentation

Dashboard

API settings

Usage plans

API keys

Client certificates

Settings

Stages

Dev

/

/inventory

GET

OPTIONS

/orders

POST

OPTIONS

/users

PUT

OPTIONS

POST

Stage details

Stage name: Dev

Cache cluster: Inactive

Default method-level caching: Inactive

Invoke URL: https://72zh1fa4gl.execute-api.ap-south-1.amazonaws.com/Dev

Active deployment: ce86wj on July 31, 2025, 18:26 (UTC+05:30)

Rate info: 10000

Band info: 5000

Web ACL: -

Client certificate: -

Logs and tracing

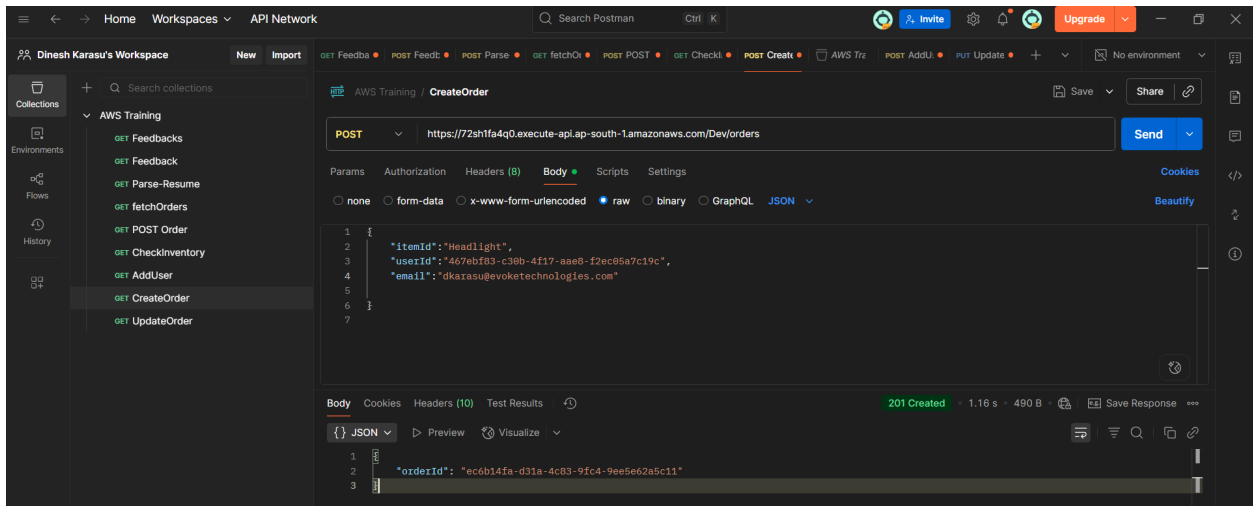
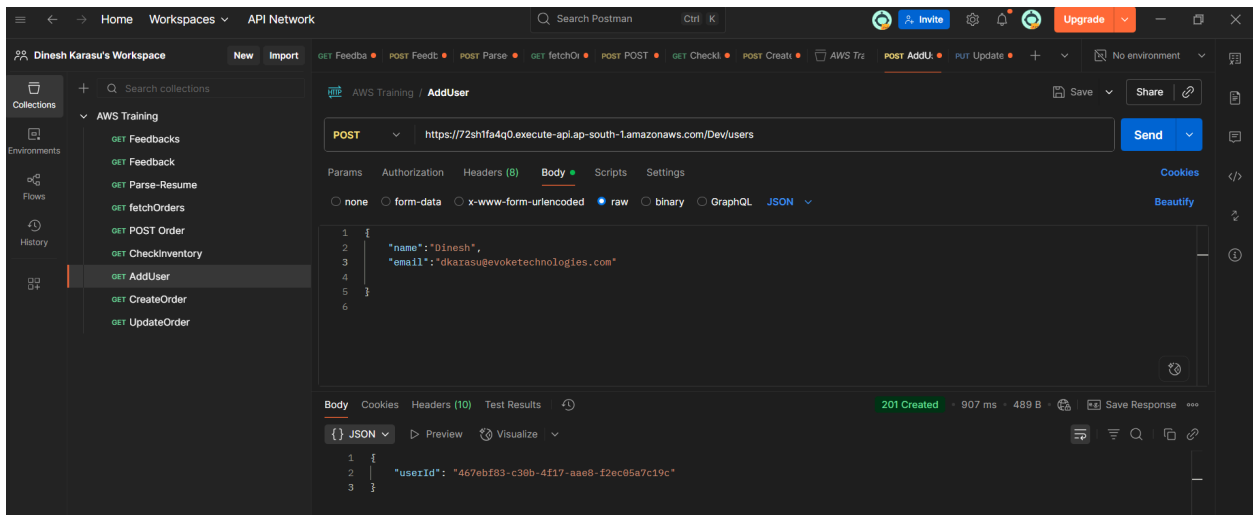
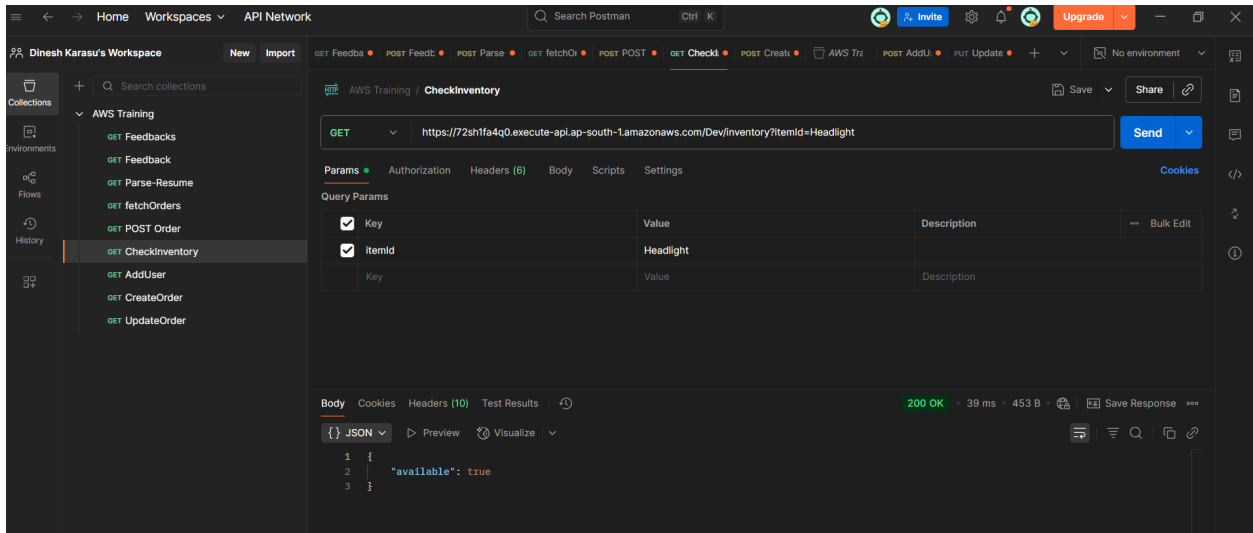
CloudWatch logs: Inactive

X-Ray tracing: Inactive

Custom access logging: Inactive

Detailed metrics: Inactive

Data tracing: Inactive



Postman interface showing a PUT request to `https://72sh1fa4q0.execute-api-south-1.amazonaws.com/Dev/orders` with a JSON body:

```
{  "orderId": "ec6b14fa-d31a-4c83-9fc4-9ee5e62a5c11",  "userId": "467ebf83-c38b-4f17-aae8-f2ec85a7c19c",  "status": "shipped"}
```

The response is a 200 OK status with a JSON body:

```
{  "message": "Order updated and customer notified."}
```

The interface includes a sidebar with collections (AWS Training) and environments, and a main panel for editing requests and viewing responses.