

AWS Assignment 2 Screenshots

Adham Hisham Salah Hassan

10000480 | T-20

Created a DynamoDB table called orders:

The screenshot shows the AWS DynamoDB console in a web browser. The URL is https://us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#tables. On the left, there's a sidebar with 'DynamoDB' selected, showing options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Under 'Tables', it says 'Tables (1)'. A success message at the top right says 'The Orders table was created successfully.' The main table view shows one row for 'Orders': Name: Orders, Status: Active, Partition key: orderId (\$), Sort key: -, Indexes: 0, Replication Regions: 0, Deletion protection: Off, Favorite: Not favorited, Read capacity: On-demand. There are buttons for Actions, Delete, and Create table.

created an sns called OrderTopic:

The screenshot shows the AWS SNS console with the URL <https://us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/topic/arn:aws:sns:us-east-1:982534348264:OrderTopic>. The page displays a success message: "Topic OrderTopic created successfully. You can create subscriptions and send messages to them from this topic." Below this, the "OrderTopic" details are shown, including its Name (OrderTopic), Display name (-), ARN (arn:aws:sns:us-east-1:982534348264:OrderTopic), and Type (Standard). The "Subscriptions" tab is selected, showing "Subscriptions (0)". At the bottom, there are links for CloudShell, Feedback, and copyright information: "© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

Created Dead Letter Queue:

The screenshot shows the AWS SQS console with the URL <https://us-east-1.console.aws.amazon.com/sqs/v3/home?region=us-east-1#/queues/https%3A%2F%2Fsqs.us-east-1.amazonaws.com%2F982534348264%2FOrderQueueDLQ>. The page displays a success message: "Queue OrderQueueDLQ created successfully. You can now send and receive messages." Below this, the "OrderQueueDLQ" details are shown, including its Name (OrderQueueDLQ), Type (Standard), ARN (arn:aws:sqs:us-east-1:982534348264:OrderQueueDLQ), and Dead-letter queue (-). The "SNS subscriptions" tab is selected, showing "SNS subscriptions (0)". At the bottom, there are links for CloudShell, Feedback, and copyright information: "© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

Create Main Queue called OrderQueue:

The screenshot shows the 'Create queue' page for Amazon SQS. It includes sections for Dead-letter queue (disabled), Set this queue to receive undeliverable messages (enabled), Choose queue (arn:aws:sqs:us-east-1:982534348264:OrderQueueDLQ), Maximum receives (3), and Tags (optional). The page also features a CloudShell link.

The screenshot shows the 'OrderQueue' details page. It displays a success message: 'Queue OrderQueue created successfully. You can now send and receive messages.' Below this, the queue's details are listed: Name (OrderQueue), Type (Standard), ARN (arn:aws:sqs:us-east-1:982534348264:OrderQueue), URL (https://SQS.us-east-1.amazonaws.com/982534348264/OrderQueue), and Dead-letter queue (Enabled). The 'SNS subscriptions' tab is selected. The page also includes a CloudShell link.

Both queues:

The screenshot shows the AWS SQS console with the URL <https://us-east-1.console.aws.amazon.com/sqsv3/home?region=us-east-1#/queues>. The page displays two Standard queues:

Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication
OrderQueue	Standard	2025-05-08T18:43+03:00	0	0	Amazon SQS key (SSE-SQS)	-
OrderQueueDLQ	Standard	2025-05-08T18:41+03:00	0	0	Amazon SQS key (SSE-SQS)	-

At the bottom, there are links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Subscribe SQS Queue to SNS Topic

The screenshot shows the AWS SNS console with the URL <https://us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/create-subscription>. A blue banner at the top indicates "New Feature: Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)".

The "Create subscription" form is filled out as follows:

- Details**: Topic ARN: `arn:aws:sns:us-east-1:982534348264:OrderTopic`
- Protocol**: The type of endpoint to subscribe: `Amazon SQS`
- Endpoint**: Only Amazon SQS standard queues will be listed and can receive notifications from an Amazon SNS standard topic. Endpoint ARN: `arn:aws:sqs:us-east-1:982534348264:OrderQueue`
- Enable raw message delivery
- After your subscription is created, you must confirm it. [Info](#)

At the bottom, there is a section for "Subscription filter policy - optional" with a link to "Info".

The screenshot shows the AWS SNS console with a successful subscription creation message: "Subscription to OrderTopic created successfully. The ARN of the subscription is arn:aws:sns:us-east-1:982534348264:OrderTopic:60e994eb-93f1-428d-9cc5-a280e4470988." Below this, the "Subscription: 60e994eb-93f1-428d-9cc5-a280e4470988" details are displayed, including ARN, Endpoint, Topic, Subscription Principal, Status (Confirmed), Protocol (SQS), and Raw message delivery (Disabled). The "Subscription filter policy" tab is selected, showing "No filter policy configured for this subscription." The browser interface includes a search bar, navigation links, and a bottom status bar.

create lambda function

The screenshot shows the AWS Lambda console's "Create function" wizard. It starts with three options: "Author from scratch" (selected), "Use a blueprint", and "Container image". The "Basic information" section allows setting the function name (ProcessOrderLambda), runtime (Python 3.12), architecture (x86_64), and permissions. To the right, a "Tutorials" sidebar provides instructions for creating a simple web app, listing steps like building a Lambda function with a URL and invoking it. The browser interface includes a search bar, navigation links, and a bottom status bar.

The screenshot shows the AWS Lambda console. In the top navigation bar, there are links for Gmail, Outlook, LeetCode 75, Codeforces, Collaboratory, Developer Roadmaps, Kotlin, W3Schools, Overleaf, TypingClub, HTB, HackerRank, Free Code Camp, Cybrary, and a search bar. The location bar shows the URL: https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/ProcessOrderLambda?newFunction=true&tab=code. The top right corner shows the user's name: adham hisham barakat.

The main content area displays the 'ProcessOrderLambda' function. It includes a 'Function overview' section with tabs for 'Diagram' (selected), 'Template', 'Throttle', 'Copy ARN', and 'Actions'. Below this is a 'Description' section with 'Last modified' (9 seconds ago) and a 'Function ARN' (arn:aws:lambda:us-east-1:982534348264:function:ProcessOrderLambda). There is also a 'Function URL' link.

The 'Code' tab is selected, showing the code editor with the file 'lambda_function.py' containing:

```

import json
import boto3

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Orders')

def lambda_handler(event, context):
    for record in event['Records']:
        message = json.loads(record['body']) # Get message from SQS

        # If message is wrapped by SNS, unwrap it
        if 'Message' in message:
            message = json.loads(message['Message'])

        print(f'Processing order: {message["orderId"]}')

        # Save to DynamoDB
        table.put_item(Item=message)
    return {
        'statusCode': 200,
        'body': json.dumps('Order processed successfully')
    }

```

Below the code editor are buttons for 'Upload from' and 'CloudShell'. The right sidebar features a 'Create a simple web app' tutorial with steps to build a Lambda function with a function URL that outputs a webpage and invoke it through its function URL.

This screenshot shows the 'Code source' editor for the same function. The left sidebar has tabs for 'EXPLORER', 'DEPLOY', and 'TEST EVENTS [NONE SELECTED]'. Under 'DEPLOY', there are buttons for 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+)'. The main area shows the 'lambda_function.py' code with syntax highlighting and line numbers. A status bar at the bottom indicates 'Successfully updated the function ProcessOrderLambda.'

The right sidebar contains the same 'Create a simple web app' tutorial as the previous screenshot.

```

import json
import boto3

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Orders')

def lambda_handler(event, context):

```

```

for record in event['Records']:
    message = json.loads(record['body']) # Get message from SQS

    # If message is wrapped by SNS, unwrap it
    if 'Message' in message:
        message = json.loads(message['Message'])

    print(f"Processing order: {message['orderId']}")

    # Save to DynamoDB
    table.put_item(Item=message)

return {
    'statusCode': 200,
    'body': json.dumps('Order processed successfully')
}

```

Add to the lambda the needed permissions

The screenshot shows the AWS IAM Roles page. The URL is https://us-east-1.console.aws.amazon.com/iam/home#/roles/details/ProcessOrderLambda-role-fgcn5w8?section=permissions. The left sidebar shows 'Identity and Access Management (IAM)' with 'Access management' expanded, showing 'Roles'. The main content area shows a green success message: 'Policies have been successfully attached to role.' It lists three policies: 'AmazonDynamoDBFullAccess', 'AmazonSQSFullAccess', and 'AWSLambdaBasicExecutionRole-22502ccc-cd21-4ef2-b...'. There are tabs for 'Permissions', 'Trust relationships', 'Tags', 'Last Accessed', and 'Revoke sessions'. A 'Permissions policies' section shows three managed policies. A 'Permissions boundary' section indicates '(not set)'. A 'Generate policy based on CloudTrail events' section shows 'No requests to generate a policy in the past 7 days.'

Add a trigger:

Trigger configuration

SQS arn:aws:sqs:us-east-1:982534348264:OrderQueue

Event poller configuration

- Activate trigger**: Select to activate the trigger now. Keep unchecked to create the trigger in a deactivated state for testing (recommended).
- Enable metrics**: Monitor your event source with metrics. You can view those metrics in CloudWatch console. Enabling this feature incurs additional costs. [Learn more](#)
- Batch size - optional**: The maximum number of records in each batch to send to the function. 10
- Batch window - optional**: The maximum amount of time to gather records before invoking the function, in seconds. 0
- Maximum concurrency - optional**: The maximum number of concurrent function instances that the SQS event source can invoke. 2
- Report batch item failures - optional**: Allow your function to return a partial successful response for a batch of records.

[CloudShell](#) [Feedback](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

ProcessOrderLambda

The trigger OrderQueue was successfully added to function ProcessOrderLambda. The trigger is in a disabled state.

Function overview

Triggers (1)

Trigger	ARN	State
SQS: OrderQueue	arn:aws:sqs:us-east-1:982534348264:OrderQueue	Creating

[Code](#) [Test](#) [Monitor](#) [Configuration](#) [Aliases](#) [Versions](#)

[CloudShell](#) [Feedback](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Publish Test Message to SNS

The screenshot shows the 'Publish message to topic' page for an 'OrderTopic'. In the 'Message details' section, the 'Topic ARN' is listed as 'arn:aws:sns:us-east-1:982534348264:OrderTopic'. The 'Subject - optional' field contains 'NewOrder'. The 'Time to Live (TTL) - optional' field is set to 0 seconds. In the 'Message body' section, the 'Message structure' dropdown is set to 'Identical payload for all delivery protocols'. The message body is a JSON object:

```
1 {
2   "orderId": "O1234",
3   "userId": "U123",
4   "itemName": "Laptop",
5   "quantity": 1,
6   "status": "new",
7   "timestamp": "2025-05-03T12:00:00Z"
8 }
```

The screenshot shows the 'OrderTopic' details page. It displays the topic's name, ARN, and type (Standard). A green banner at the top indicates that 'Message published to topic OrderTopic successfully.' Below the details, there are tabs for 'Subscriptions', 'Access policy', 'Data protection policy', 'Delivery policy (HTTP/S)', 'Delivery status logging', 'Encryption', 'Tags', and 'Integrations'. The 'Subscriptions' tab shows one subscription entry:

ID	Endpoint	Status	Protocol
60e994eb-93f1-428d-9cc5-a280e4470988	arn:aws:sqs:us-east-1:982534348264:Order...	Confirmed	SQS

check dynamodb for the order as it has been placed

DynamoDB

Tables (1)

Orders

Scan or query items

Select a table or index: Table - Orders

Select attribute projection: All attributes

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

	orderId (String)	itemName	quantity	status	timestamp	userId
	01234	Laptop	1	new	2025-05-08T12:34:56Z	U123

check cloudwatch group logs for the order 01234

CloudWatch

Log events

Timestamp | Message

2025-05-08T16:16:42.651Z INIT START Runtime Version: python3.12.v65 Runtime Version ARN: arn:aws:lambda:us-east-1:1:runtime:74b4a691bc20ef933c2b521f88092b9287c36c73d31d47774c5e8b8428b563d

2025-05-08T16:16:43.125Z START RequestId: 91768743-1245-5f5a-9d80-e7dc9636162 Version: \$LATEST

2025-05-08T16:16:43.126Z Processing order: 01234

2025-05-08T16:16:43.422Z END RequestId: 91768743-1245-5f5a-9d80-e7dc9636162 Duration: 296.23 ms Billed Duration: 297 ms Memory Size: 128 MB Max Memory Used: 85 MB Init Duration: 470.46 ms

2025-05-08T16:16:43.422Z REPORT RequestId: 91768743-1245-5f5a-9d80-e7dc9636162 Duration: 296.23 ms Billed Duration: 297 ms Memory Size: 128 MB Max Memory Used: 85 MB Init Duration: 470.46 ms

cloudformation template (Bonus)

CloudFormation < OrderProcessingStack

CloudFormation

- Stacks
- Stack details**
- Drifts
- StackSets
- Exports
- Infrastructure Composer
- IaC generator
- Hooks overview
- Hooks
- Registry
- Public extensions
- Activated extensions
- Publisher
- Spotlight
- Feedback

OrderProcessingStack

Events (1)

Timestamp	Logical ID	Status	Detailed status	Status reason	Hook info
2025-05-08 19:44:24 UTC+0300	OrderProcessingStack	CREATE_IN_PROGRESS	-	User Initiated	-

CloudFormation < OrderProcessingStack

CloudFormation

- Stacks
- Stack details**
- Drifts
- StackSets
- Exports
- Infrastructure Composer
- IaC generator
- Hooks overview
- Hooks
- Registry
- Public extensions
- Activated extensions
- Publisher
- Spotlight
- Feedback

Events (15)

Timestamp	Logical ID	Status	Detailed status	Status reason	Hook info
2025-05-08 19:46:35 UTC+0300	OrderQueueDLQ	DELETE_COMPLETE	-	-	-
2025-05-08 19:44:31 UTC+0300	ProcessOrderLambdaRole	DELETE_COMPLETE	-	-	-
2025-05-08 19:44:30 UTC+0300	ProcessOrderLambdaRole	DELETE_IN_PROGRESS	-	The role with name ProcessOrderLambdaRole cannot be found. [Error ID: 81cb794a-1518-4efab6d5-88835e0635b0] (SDK Attempt Count: 1)	-
2025-05-08 19:44:30 UTC+0300	OrdersTable	DELETE_COMPLETE	-	-	-
2025-05-08 19:44:30 UTC+0300	ProcessOrderLambdaRole	DELETE_IN_PROGRESS	-	The following resource(s) failed to create: [OrdersTable, OrderTopic, ProcessOrderLambdaRole, OrderQueueDLQ]. Rollback requested by user.	-
2025-05-08 19:44:28 UTC+0300	OrderProcessingStack	ROLLBACK_IN_PROGRESS	-	Resource creation cancelled	-
2025-05-08 19:44:28 UTC+0300	ProcessOrderLambdaRole	CREATE_FAILED	-	Resource creation cancelled	-

Script .yaml file for bonus Exported CloudFormation course i did all using the console manually:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Event-Driven Order Notification System
```

Resources:

OrdersTable:

Type: AWS::DynamoDB::Table

Properties:

 TableName: Orders

 AttributeDefinitions:

 - AttributeName: orderId

 AttributeType: S

 KeySchema:

 - AttributeName: orderId

 KeyType: HASH

 BillingMode: PAY_PER_REQUEST

OrderTopic:

Type: AWS::SNS::Topic

Properties:

 TopicName: OrderTopic

OrderQueueDLQ:

Type: AWS::SQS::Queue

Properties:

 QueueName: OrderQueueDLQ-CF # added suffix to avoid name conflict

OrderQueue:

Type: AWS::SQS::Queue

Properties:

 QueueName: OrderQueue-CF # added suffix to avoid name conflict

RedrivePolicy:

 deadLetterTargetArn: !GetAtt OrderQueueDLQ.Arn

 maxReceiveCount: 3

OrderQueueSubscription:

Type: AWS::SNS::Subscription

Properties:

 TopicArn: !Ref OrderTopic

 Protocol: sqs

 Endpoint: !GetAtt OrderQueue.Arn

 RawMessageDelivery: false

```
OrderQueuePolicy:  
  Type: AWS::SQS::QueuePolicy  
  Properties:  
    Queues:  
      - !Ref OrderQueue  
  PolicyDocument:  
    Version: "2012-10-17"  
    Statement:  
      - Effect: Allow  
        Principal: "*"  
        Action: "SQS SendMessage"  
        Resource: !GetAtt OrderQueue.Arn  
    Condition:  
      ArnEquals:  
        aws:SourceArn: !Ref OrderTopic  
  
ProcessOrderLambdaRole:  
  Type: AWS::IAM::Role  
  Properties:  
    # No RoleName → auto-generated by CloudFormation to avoid naming conflict  
    AssumeRolePolicyDocument:  
      Version: "2012-10-17"  
      Statement:  
        - Effect: Allow  
          Principal:  
            Service: lambda.amazonaws.com  
          Action: sts:AssumeRole  
    ManagedPolicyArns:  
      - arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole  
      - arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess  
      - arn:aws:iam::aws:policy/AmazonSQSFullAccess  
  
ProcessOrderLambda:  
  Type: AWS::Lambda::Function  
  Properties:  
    FunctionName: ProcessOrderLambda  
    Runtime: python3.12
```

```
Handler: index.lambda_handler
Role: !GetAtt ProcessOrderLambdaRole.Arn
Code:
ZipFile: |
    import json
    import boto3

    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('Orders')

    def lambda_handler(event, context):
        for record in event['Records']:
            message = json.loads(record['body'])
            if 'Message' in message:
                message = json.loads(message['Message'])
                print(f"Processing order: {message['orderId']}")"
                table.put_item(Item=message)
        return {
            'statusCode': 200,
            'body': json.dumps('Order processed successfully')
        }
```

LambdaSQSTrigger:

Type: AWS::Lambda::EventSourceMapping

Properties:

EventSourceArn: !GetAtt OrderQueue.Arn

FunctionName: !Ref ProcessOrderLambda

Enabled: true

BatchSize: 1

Outputs:

OrdersTableName:

Description: DynamoDB table name

Value: !Ref OrdersTable

OrderTopicArn:

Description: SNS topic ARN

Value: !Ref OrderTopic

OrderQueueURL:

Description: SQS Queue URL

Value: !Ref OrderQueue

OrderQueueDLQURL:

Description: SQS Dead Letter Queue URL

Value: !Ref OrderQueueDLQ

LambdaFunctionName:

Description: Lambda function name

Value: !Ref ProcessOrderLambda