

Project Title – Smart Inventory System on AWS

Use Case: You run a e-commerce backend system that maintain product stock in Dynamodb table. Each time a product is purchased, the stock is decremented (VIA API Call). If the stock goes below a certain threshold the system will send an alert to the subscribed email.

Architecture: Event Driven, Serverless

Key Services:

1. DynamoDB
2. Lambda
3. SNS
4. CloudWatch
5. API Gateway
6. IAM
7. Postman (For Sending Request)

Dynamodb is serverless and automatically scales without provisioning infrastructure and it fits our use case because it is event driven architecture.

Steps

1. Create a Dynamo DB Table
2. Add the following items
 - a. Table Name: Inventory
 - b. Primary Key: product_id (string)
 - c. Create Table
 - d. Add the attributes after table creation
 - i. product_id: string
 - ii. product_name: string
 - iii. stock: number
 - iv. threshold: number

3. Create Table in DynamoDB

Create table

Table details [info](#)
DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

1 to 255 characters and case sensitive.

Table settings

☒ **Default settings**
The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose "Customize settings."

☐ **Customize settings**
Use these advanced features to make DynamoDB work better for your needs.

Default table settings
These are the default settings for your new table. You can change some of these settings after creating the table.

Setting	Value	Editable after creation
Table class	DynamoDB Standard	Yes
Capacity mode	On-demand	Yes
Maximum read capacity units	-	Yes
Maximum write capacity units	-	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.
No tags are associated with the resource.

You can add 50 more tags.

4. Select the Inventory Table
5. Click on Actions
6. Create Items
7. You can add more items. This is just a demo and to keep the cost minimum, only one item has been created.

Edit item Form JSON view

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn more](#)

Attributes Add new attribute

Attribute name	Value	Type	
product_id - Partition key	p001	String	
product_name	wireless mouse	String	Remove
stock_level	25	Number	Remove
threshold	10	Number	Remove

Cancel Save Save and close

8. Create a Simple Notification Service
9. Go to SNS
10. Create Topic

Create topic

Details

Type [info](#)
Topic type cannot be modified after topic is created.

☐ FIFO (first-in, first-out)
• Strictly guaranteed message ordering
• Exactly-once message delivery
• Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SNS, email, mobile application endpoints

☒ **Standard**
• Best-effort message ordering
• At least once message delivery
• Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SNS, email, mobile application endpoints

Name
LowStockAlert
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - optional [info](#)
To use this topic with SNS subscriptions, enter a display name. Only the first 10 characters are displayed in an SNS message.
LowStockAlert
Maximum 100 characters.

Encryption - optional
Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

Access policy - optional [info](#)
This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.

Data protection policy - optional [info](#)
This policy defines which sensitive data to monitor and to prevent from being exchanged via your topic.

Delivery policy (HTTP/S) - optional [info](#)
The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section.

Delivery status logging - optional [info](#)
These settings configure the logging of message delivery status to CloudWatch Logs.

Tags - optional
A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)

Active tracing - optional [info](#)
Use X-Ray's in-flight active tracing for this topic to view its traces and service map in Amazon CloudWatch. Additional costs apply.

Cancel Create topic

11. Click on Create Topic
12. Next Click on Create a Subscription
13. Select the Topic from the drop down
14. Select Protocol as EMAIL
15. Select Endpoint (Your Personal Email ID for testing)
16. Click on Create Subscription

Create subscription

Details

Topic ARN
arn:aws:sns:us-east-1:277707095024:LowStockLevelAlert

Protocol
Email

Endpoint
[Redacted]@gmail.com

After your subscription is created, you must confirm it.

Subscription filter policy - optional
This policy filters the messages that a subscriber receives.

Redrive policy (dead-letter queue) - optional
Send undeliverable messages to a dead-letter queue.

Cancel Create subscription

17. Go to your person email Inbox
18. Click on Confirm the Subscription

Subscription confirmed!
You have successfully subscribed.

19. Create a lambda function
20. Go to Lambda > Create Function
21. Select Author From Scratch
22. Select Python as Run Time
23. Add the IAM roles to the lambda function
24. Provide a name for the lambda function(name:lambdainventoryrole)
25. Select Use Existing IAM Roles and Select the IAM Role Created
26. Refer the IAM Screenshot Role
 - a. AmazonDynamoDBFullAccess
 - b. AmazonSNSFullAccess
 - c. CloudWatchLogsFullAccess
27. Refer the lambdafunction creation screenshot

28. IAM Screenshot

Name, review, and create

Role details

Role name
 Permission Administrator (Full Control)

Full control permissions

Review the permissions for the permission set "Full Control".
 All roles inherit permissions from the role "Full Control".

Review
 All roles inherit permissions from the role "Full Control".

Review
 All roles inherit permissions from the role "Full Control".

Review the permissions for the permission set "Full Control".
 All roles inherit permissions from the role "Full Control".

Step 1: Select trusted entities

Step 1: Select trusted entities

Step 2: Add permissions

Step 2: Add permissions

Permission policy summary

Permission policy summary

Policy name	Type	Permissions
AmazonEC2FullControl	Full control	AmazonEC2FullControl
AmazonEC2ReadOnlyAccess	Full control	AmazonEC2ReadOnlyAccess
AmazonEC2ReadOnlyAccess	Full control	AmazonEC2ReadOnlyAccess

Step 3: Add tags

Step 3: Add tags

Add tag - optional

Type the value for the tag and click Add. The tag value must be unique.

Key: Value:

Add tag

Review the tag value.

Done Cancel Create

29. LAMBDA Screenshot

Create function NEW

Choose one of the following options to create your function.

☒ **Author from scratch**
 Start with a sample function or Lambda example.

☐ **Use a Blueprint**
 Build a custom application from sample code and configuration provided for common use cases.


☐ **Container image**
 Select a container image to deploy for your function.

Basic information

Function name
 Give your function the name of your function.

Function name must be 1 to 63 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, 0-9, hyphens (-), and underscores (_).

Runtime Info
 Choose a language to use to write your function. Note that the console code editor supports only Ruby, Python, and Java.



Architecture Info
 Select the execution architecture you want for your function code.

☒ x86_64
☐ arm64

Permissions Info
 By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding VPCs.


Change default execution role

Execution role
 Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☒ [Create a new role using AWS Lambda permissions](#)
Go to a new role page

☐ [Create a new role from AWS policy templates](#)

Execution role
 Select an existing role that you created to be used with this Lambda function. The role will have permission to upload logs to Amazon CloudWatch Logs.



[View the lambdaexecutionrole role](#) in the IAM console.

Additional Configurations

Use additional configurations to set up code signing, function URLs, tags, and Amazon VPC access for your function.

Enable Code signing

☐ Use the signing configuration to ensure that the code has been signed by an approved source and has not been altered since signing.

Enable recognition with an AWS IoT Core managed key

☐ If enabled, Lambda recognizes the key by the private key and the AWS IoT Core key.

Enable function URL

☐ Use function URLs to expose HTTP(S) endpoints to your Lambda function.

Enable tags

☐ A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and track your resources, track your AWS costs, and perform other actions across your account.

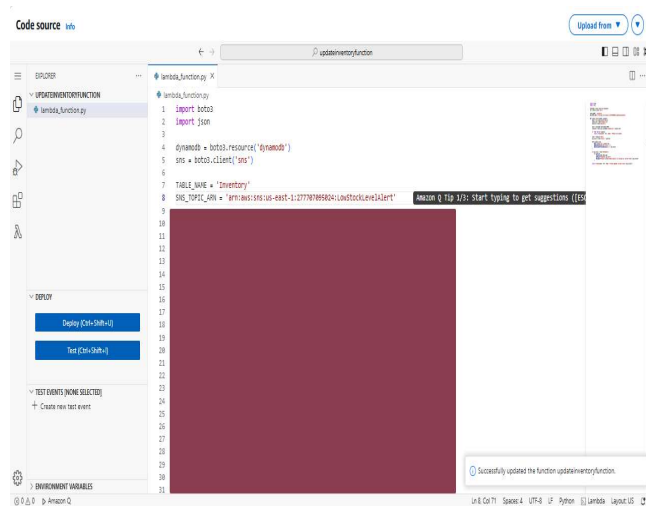
Enable VPC

☐ Connect your function to a VPC to access private resources during execution.

30. The Python Code is updated in the console and deployed

31. You can test using the json code or do it at the end

32. Do not forget to paste the SNS topic ARN



1. Create an API Gateway
2. Choose HTTP API
3. Click on Build
4. Provide a API Name
5. Select IPV4
6. Select Integration as Lambda
7. Select the Lambda Function
8. Configure the Route
9. Select Method as POST
10. Resource Path /updateinventory (to be consistent and not forget I use the same naming convention)
11. Provide a stage name
12. Disable automatic deploy
13. Manually deploy after review

Configure API

API details

API name
An HTTP API must have a name. The name is a non-unique value you use to identify and organize your APIs. To programmatically refer to this API, use the API ID that API Gateway generates for you.

updateinventory

IP address type Info
Select the type of IP addresses that can invoke the default endpoint for your API. You don't need to redeploy your API for the update to take effect.

☒ **IPv4**
Includes only IPv4 addresses.

☐ **Dualstack**
Includes IPv4 and IPv6 addresses.

Integrations 1 Info
Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For an HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Integration	Backend service	Method	URI	Version	Actions
Lambda	Q. amaws.lambda.us-east-1:27707096024:function:updateinventoryfunc	POST	/updateinventory	2.0	Remove

AWS Region **Lambda function** **Version**

us-east-1 Q. amaws.lambda.us-east-1:27707096024:function:updateinventoryfunc 2.0

[Add integration](#)

[Cancel](#) [Review and create](#) [Next](#)

Configure routes - optional

Configure routes info
API Gateway uses routes to expose integrations to consumers of your API. Routes for HTTP APIs consist of two parts: an HTTP method and a resource path (e.g., GET /pets). You can define specific HTTP methods for your integration (GET, POST, PUT, PATCH, HEAD, OPTIONS, and DELETE) or use the ANY method to match all methods that you haven't defined on a given resource.

Method
POST

Resource path
/updateinventory

→

Integration target
updateinventoryfunction

Remove

Add route

Cancel Review and create Previous Next

Define stages - optional

Configure stages info
Stages are independently configurable environments that your API can be deployed to. You must deploy to a stage for API configuration changes to take effect, unless that stage is configured to auto-deploy. By default, all HTTP APIs created through the console have a default stage named \$default. All changes that you make to your API are auto-deployed to that stage. You can add stages that represent environments such as development or production.

Stage name
prod

Auto-deploy
☒

Remove

Add stage

Cancel Previous Next

Review and create

API name and integrations info

API name
updateinventory
Integrations
• updateinventoryfunction (Lambda)

IP address type
IPv4

Edit

Routes info

Routes
• POST /updateinventory → updateinventoryfunction (Lambda)

Edit

Stages info

Stages
• prod (Auto-deploy: disabled)

Edit

Cancel Previous Create

14. Go to Stage Details

15. Copy the InvokeURL

16. Go to POSTMAN (web version or if installed)

17. Change the method to POST

18. URL – Paste the API Invoke URL add /updateinventory at the end

19. Select Body

20. Select Raw

21. Select JSON

22. Enter the code to test

23. Select Params

24. In the key enter Content-Type and application/json

25. Click on Send

26. In the DynamoDB the Current Stock will Show as 13

27. Change the Raw code to 13

28. The Stock will be Zero and you will get a Notification Alert

29. Refer the Screen Shot

Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {  
2   "product_id": "p001",  
3   "quantity": 12  
4 }  
5
```

Params

Authorization

Headers (8)

Body

Scripts

Settings

Query Params

<input type="checkbox"/> Key	Value	Description
<input checked="" type="checkbox"/> Content-Type	application/json	
<input type="checkbox"/> Key	Value	Description

Table: Inventory - Items returned (1)

Scan started on May 13, 2025, 14:01:01

<input type="checkbox"/>	product_id (String) ▾	product_name ▾	stock ▾	threshold
<input type="checkbox"/>	p001	Wireless Mouse	13	10

[illegible]