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:

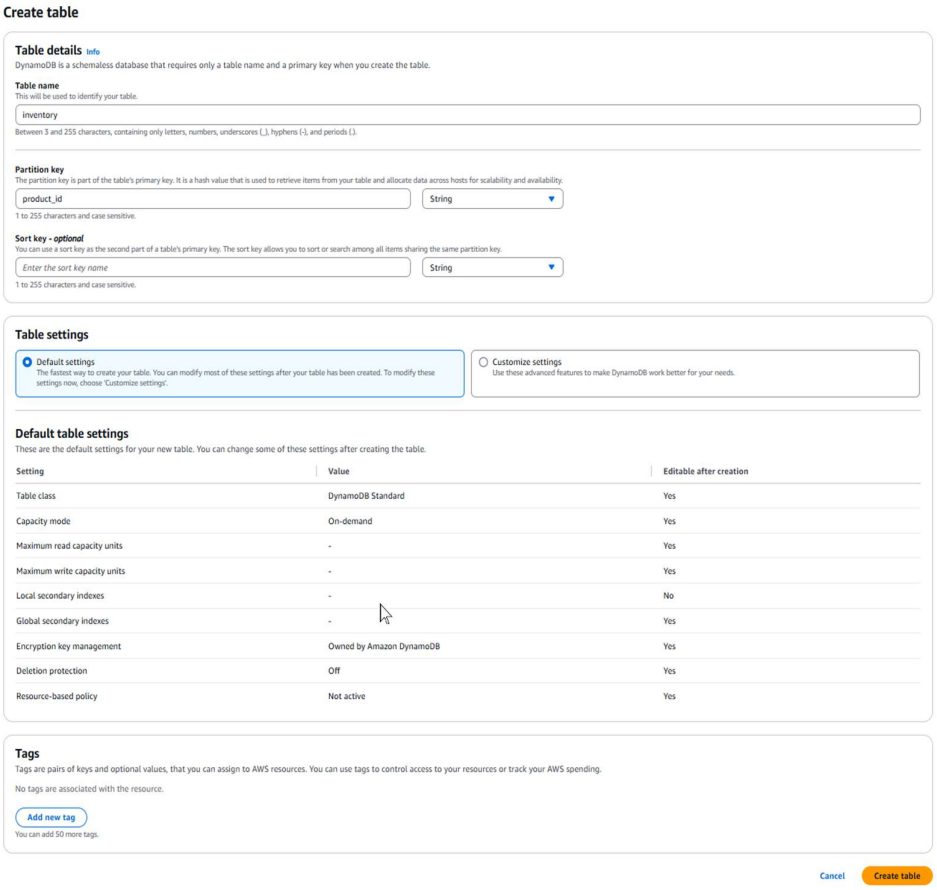
1. DynamoDB
2. Lambda
3. SNS
4. CloudWatch
5. API Gateway
6. IAM
7. Postman (For Sending Request)

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

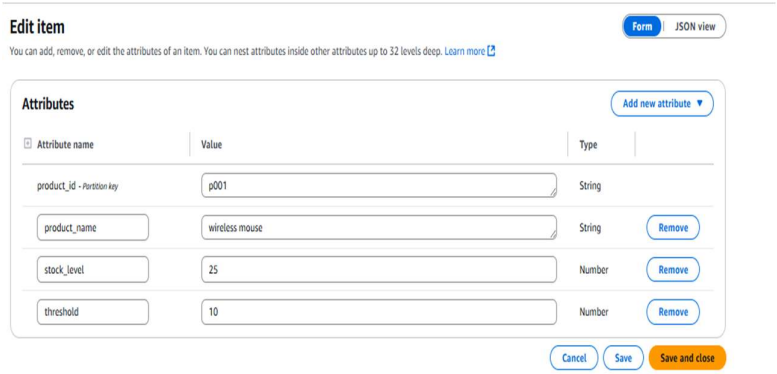
Steps

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

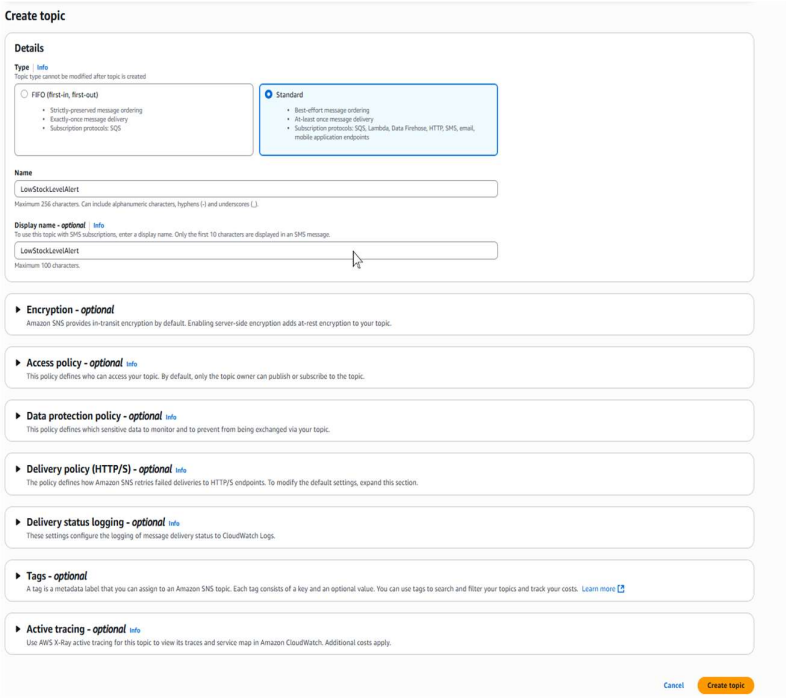
3. Create Table in DynamoDB



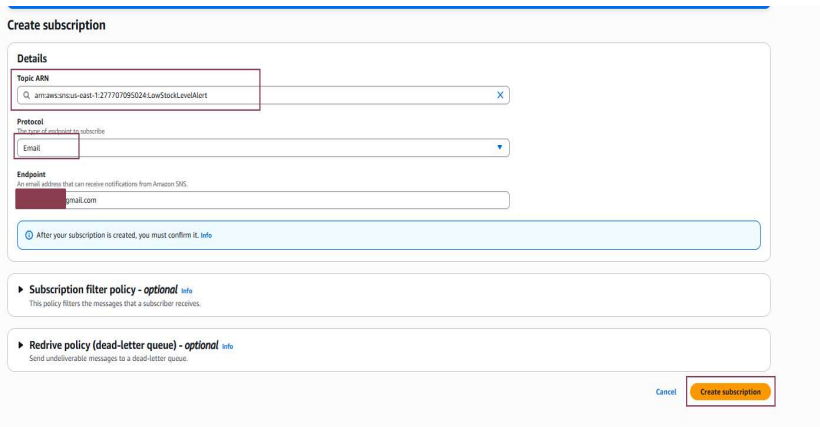
4. Select the Inventory Table
5. Click on Actions
6. Create Items
7. You can add more items. This is just a demo and to keep the cost minimum, only one item has been created.



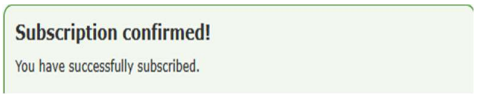
8. Create a Simple Notification Service
9. Go to SNS
10. Create Topic



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

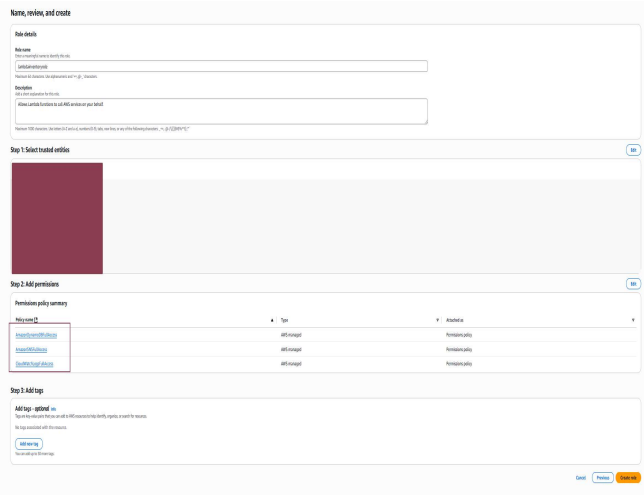


17. Go to your person email Inbox
18. Clicl on Confirm the Subscription

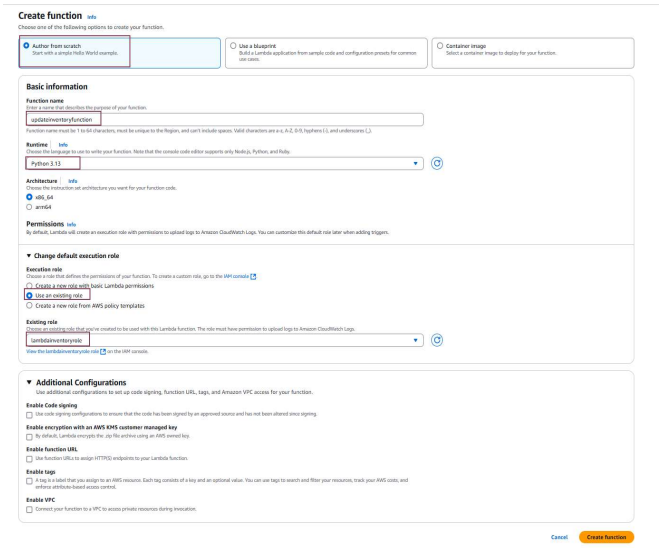


19. Create a lambda function
20. Go to Lambda > Create Function
21. Select Author From Scratch
22. Select Python as Run Time
23. Add the IAM roles to the lambda function
24. Provide a name for the lambda function(name:lambdainventoryrole)
25. Select Use Existing IAM Roles and Select the IAM Role Created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

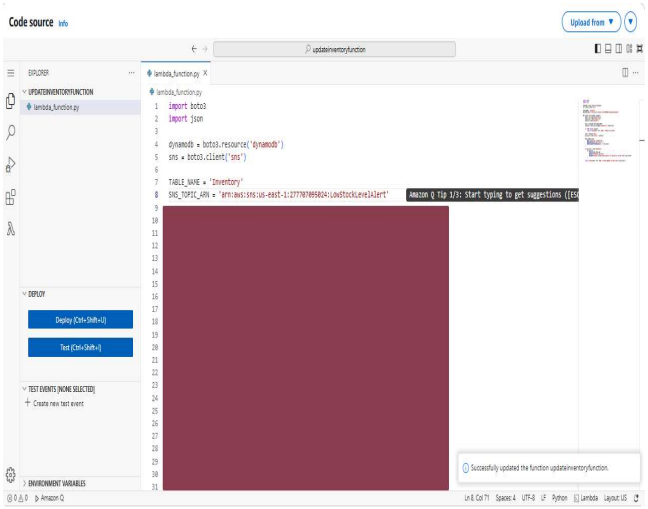
28. IAM Screenshot



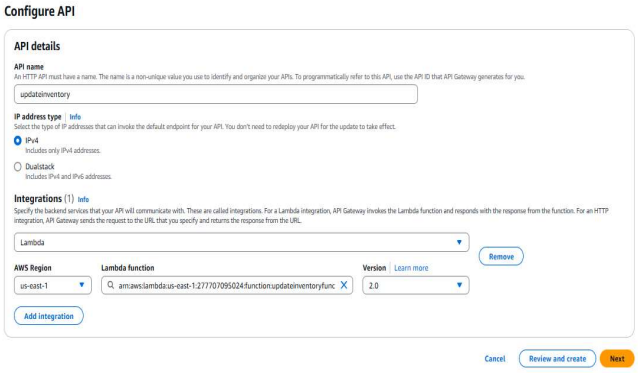
29. LAMBDA Screenshot

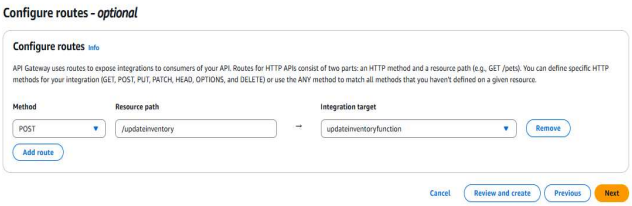


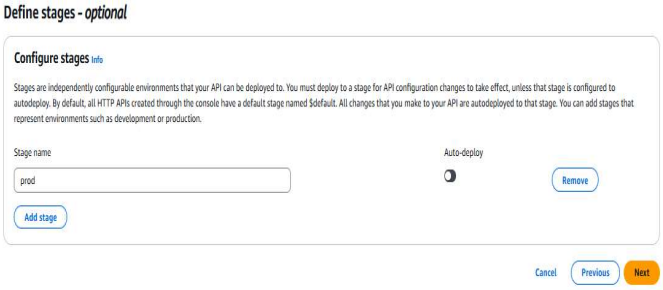
30. The Python Code is updated in the console and deployed
31. You can test using the json code or do it at the end
32. Do not forget to paste the SNS topic ARN

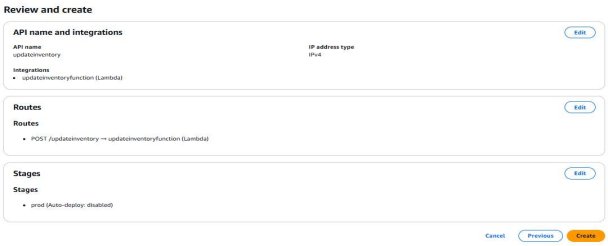


1. Create an API Gateway
2. Choose HTTP API
3. Click on Build
4. Provide a API Name
5. Select IPV4
6. Select Integration as Lambda
7. Select the Lambda Function
8. Configure the Route
9. Select Method as POST
10. Resource Path /updateinventory (to be consistent and not forget I use the same naming convention)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









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

POST https://36nlbaj3l5.execute-api.us-east-1.amazonaws.com/prod/updateinventory

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
  "product_id": "p001",
  "quantity": 12
}
```

Params Authorization Headers (8) Body Scripts Settings

Query Params

Key	Value	Description
Content-Type	application/json	
Key	Value	Description

Table: Inventory - Items returned (1)

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

product_id *(String)*	product_name	stock	threshold
p001	Wireless Mouse	13	10

Body Cookies Headers (5) Test Results

Raw Preview Visualize

```
Stock updated. Current stock: 13
```