**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

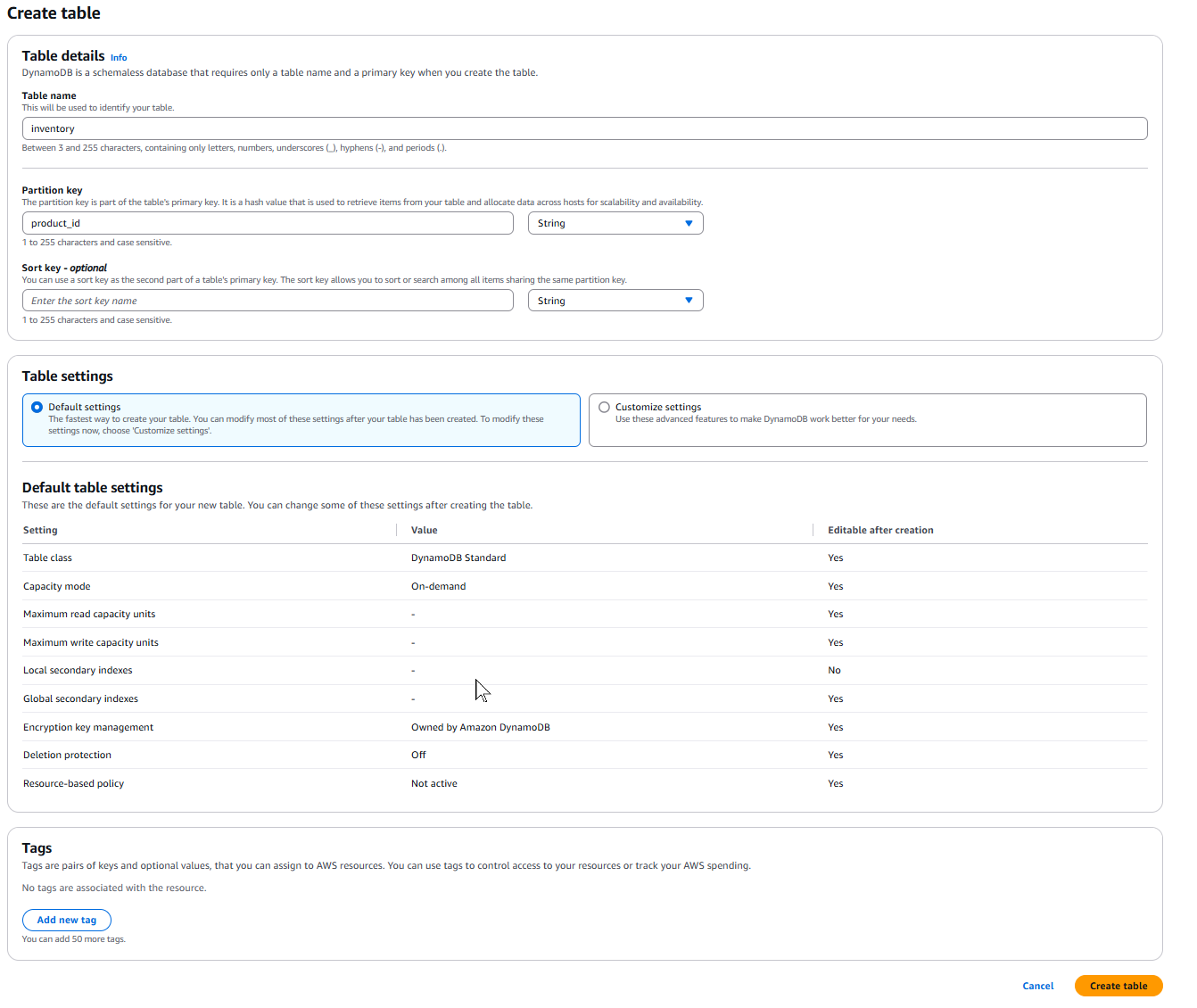
**Key Services:**

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

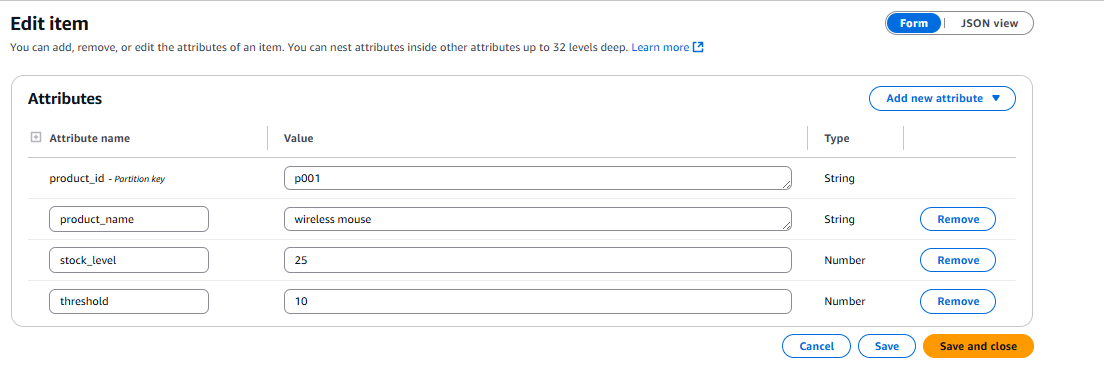
Dynamodb is serverless and automatically scales without provisioning infrastructure and it fits out use case because it is event driven architecture.

**Steps**

1. Create a Dynamo DB Table
2. Add the following items
   1. Table Name: Inventory
   2. Primary Key: product\_id (string)
   3. Create Table
   4. Add the attributes after table creation
      1. product\_id: string
      2. product\_name: string
      3. stock: number
      4. threshold: number
3. Create Table in DynamoDB



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

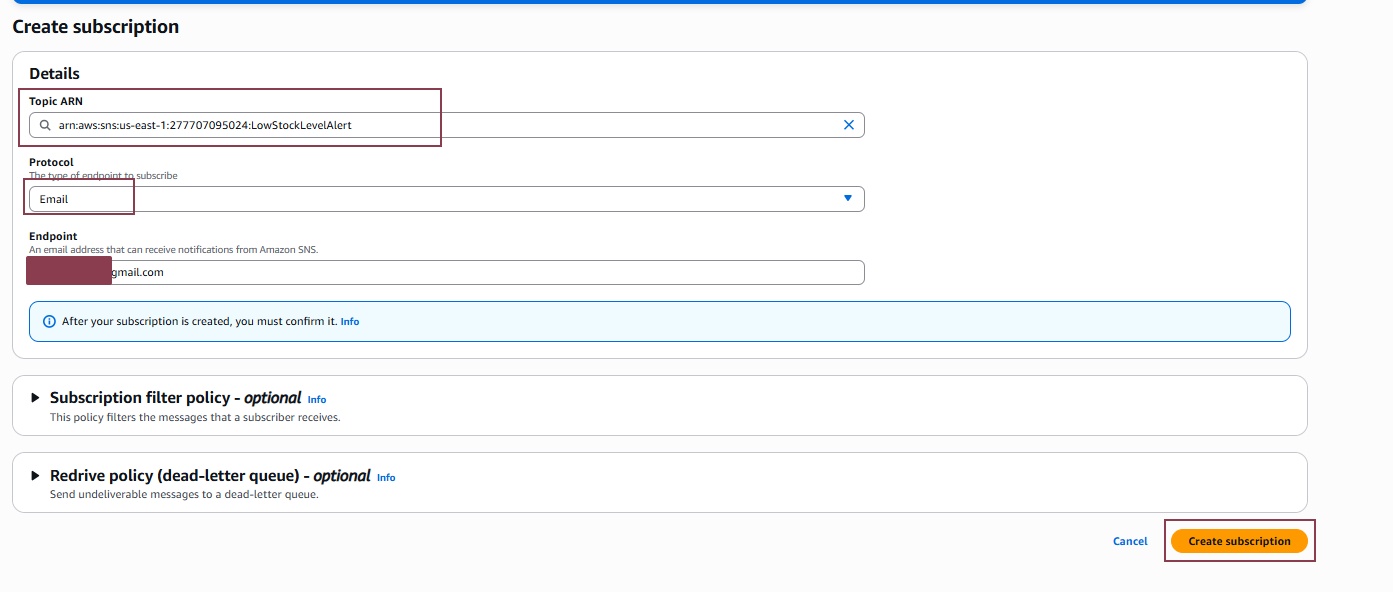


1. Create a Simple Notification Service
2. Go to SNS
3. Create Topic

A screenshot of a computer

AI-generated content may be incorrect.

1. Click on Create Topic
2. Next Click on Create a Subscription
3. Select the Topic from the drop down
4. Select Protocol as EMAIL
5. Select Endpoint (Your Personal Email ID for testing)
6. Click on Create Subscription



1. Go to your person email Inbox
2. Clicl on Confirm the Subscription

A computer screen shot of a computer screen

AI-generated content may be incorrect.

1. Create a lambda function
2. Go to Lambda > Create Function
3. Select Author From Scratch
4. Select Python as Run Time
5. Add the IAM roles to the lambda function
6. Provide a name for the lambda function(name:lambdainventoryrole)
7. Select Use Existing IAM Roles and Select the IAM Role Created
8. Refer the IAM Screenshot Role
   1. AmazonDynamoDBFullAccess
   2. AmazonSNSFullAccess
   3. CloudWatchLogsFullAccess
9. Refer the lambdafunction creation screenshot
10. IAM Screenshot

A screenshot of a computer

AI-generated content may be incorrect.

1. LAMBDA Screenshot

A screenshot of a computer

AI-generated content may be incorrect.

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

A screenshot of a computer

AI-generated content may be incorrect.

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

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

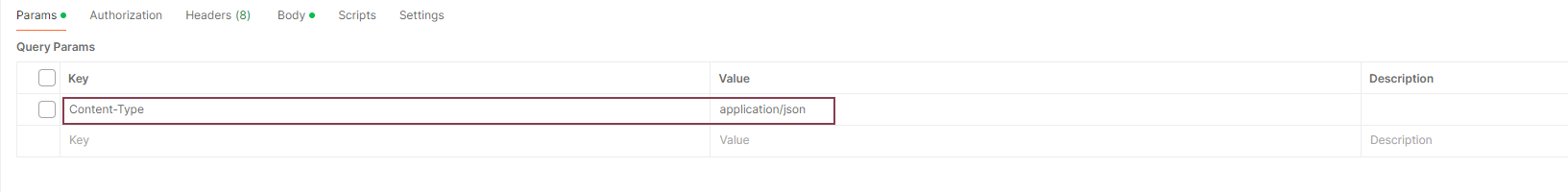
AI-generated content may be incorrect.

1. Go to Stage Details
2. Copy the InvokeURL
3. Go to POSTMAN (web version or if installed)
4. Change the method to POST
5. URL – Paste the API Invoke URL add /updateinventory at the end
6. Select Body
7. Select Raw
8. Select JSON
9. Enter the code to test
10. Select Params
11. In the key enter Content-Type and application/json
12. Click on Send
13. In the DynamoDB the Current Stock will Show as 13
14. Change the Raw code to 13
15. The Stock will be Zero and you will get a Notification Alert
16. Refer the Screen Shot



A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.