* **API Gateway** → GET /products
* **Lambda (Python)** → Fetch products from DynamoDB + send SNS notification
* **DynamoDB** → Holds sample product data
* **SNS** → Sends email whenever products are fetched

## **Step 1 — Create the DynamoDB Table (same as before)**

* Table name: ProductsTable
* Partition key: productId (String)
* Add sample items:
* { "productId": "p1", "name": "Blue T-shirt", "price": 19.99 }
* { "productId": "p2", "name": "Red Mug", "price": 9.99 }
* { "productId": "p3", "name": "Sneakers", "price": 59.99 }

## **Step 2 — Create the SNS Topic**

* Topic name: ProductFetchTopic
* Create a subscription → Protocol: **Email**, Endpoint: your email → confirm it from your inbox.

## **Step 3 — Create Lambda Function**

1. Go to **Lambda** → **Create function**.
   * Function name: GetProductsPython
   * Runtime: **Python 3.12**
   * Permissions: Create a new role with basic Lambda permissions
   * Create function

Replace the code in **Code source** with:  
import boto3

import json

from decimal import Decimal

dynamodb = boto3.resource("dynamodb")

sns = boto3.client("sns")

TABLE\_NAME = "ProductsTable"

TOPIC\_ARN = "arn:aws:sns:REGION:ACCOUNT\_ID:ProductFetchTopic"

# Custom encoder to convert Decimal → float

class DecimalEncoder(json.JSONEncoder):

    def default(self, obj):

        if isinstance(obj, Decimal):

            return float(obj)  # or int(obj) if you prefer

        return super(DecimalEncoder, self).default(obj)

def lambda\_handler(event, context):

    try:

        table = dynamodb.Table(TABLE\_NAME)

        response = table.scan()

        items = response.get("Items", [])

# Publish notification to SNS

sns.publish(

TopicArn=TOPIC\_ARN,

Subject="Products Fetched",

Message=f"Products fetched successfully. Count: {len(items)}"

)

        return {

            "statusCode": 200,

            "headers": {

                "Content-Type": "application/json",

                "Access-Control-Allow-Origin": "\*"

            },

            "body": json.dumps(items, cls=DecimalEncoder)

        }

    except Exception as e:

        print("Error fetching products:", str(e))

        return {

            "statusCode": 500,

            "body": json.dumps({"error": str(e)})

        }

1. Click **Deploy**.

## **Step 4 — Attach Permissions**

Go to Lambda → **Configuration → Permissions** → click role name → **Attach policies**:

* AmazonDynamoDBReadOnlyAccess
* AmazonSNSFullAccess

## Step 5 — Create API Gateway

1. Go to **API Gateway** → Create **REST API** → Build.
2. Add a new resource /products.
3. Add a **GET method** → Integration type: **Lambda function** → Select GetProductsPython.
4. Enable **Lambda proxy integration**.

## Step 6 — Deploy API

1. **Actions → Deploy API** → Stage name: prod.
2. Copy the Invoke URL, e.g.:

https://abc123xyz.execute-api.ap-south-1.amazonaws.com/prod/products

## **Step 7 — Test in Postman**

* Method: **GET**
* URL:

https://abc123xyz.execute-api.ap-south-1.amazonaws.com/prod/products