■ Serverless Notification System using DynamoDB + Lambda + SNS

This project demonstrates how to build a simple, real-world serverless application on AWS using DynamoDB, Lambda, and SNS. The goal is to store data (toy information) in a DynamoDB table and send an email notification every time a new item is added.

■ Table of Contents

■ Project Idea

Build a toy collection system where every time a new toy is added, it is saved in a DynamoDB table and a notification email is sent using SNS, triggered by a Lambda function.

■ AWS Services Used

Service	Purpose
DynamoDB	Store toy data in a fast, serverless NoSQL database
Lambda	Run a function to write data and send email notification
SNS	Send email alerts when new data is added
IAM	Manage permissions between services

■ Objective

Create a system that:

Accepts toy details (ToyID, ToyName) via Lambda

Stores them in DynamoDB

Sends an email notification via SNS

■■ Architecture Diagram

```
+---------+ +-----------+ +-----------+
| | | | | |
| User +-------> | Lambda +------> | DynamoDB |
| | | Function | | Table |
| | +-----+-----+ +-----------+
| | |
| | v
| | +----------+
| | | SNS |
| | | Topic |
| | +----+-----+
| | |
| | v
| | Email Notification
+---------+ ■
```

■ Step-by-Step Setup

1■■ Create DynamoDB Table

Name: ToyCollection

Partition Key: ToyID (String)

Billing Mode: Pay-per-request (Free Tier)

2■■ Create Lambda Function

Name: AddToyToDynamoDB

Runtime: Python 3.12

Role Permissions:

AmazonDynamoDBFullAccess

AmazonSNSFullAccess

Lambda Code:

```
import json
import boto3

dynamodb = boto3.resource('dynamodb')
sns = boto3.client('sns')

table = dynamodb.Table('ToyCollection')
TOPIC_ARN = 'arn:aws:sns:ap-south-1::ToyAlerts' # Replace this
```

```
def lambda_handler(event, context):
toy_id = event['ToyID']
toy_name = event['ToyName']

table.put_item(Item={"ToyID": toy_id, "ToyName": toy_name})

sns.publish(
TopicArn=TOPIC_ARN,
Subject="New Toy Added",
Message=f"A new toy was added: {toy_name} (ID: {toy_id})"
)

return {
'statusCode': 200,
'body': json.dumps("Toy added and notification sent!")
}
```

3■■ Create SNS Topic

Name: ToyAlerts

Subscribe with your email

Confirm the subscription via email link

■ Testing

Go to Lambda → Click Test

Input Test JSON:

```
{
"ToyID": "102",
"ToyName": "LEGO Blocks"
}
```

Result:

Data is added to DynamoDB

Email notification is received with subject New Toy Added

■ Cleanup (Free Tier Safety)

Use these CLI commands to remove resources after testing:

Delete Lambda:

```
aws lambda delete-function --function-name AddToyToDynamoDB
```

Delete DynamoDB Table:

```
aws dynamodb delete-table --table-name ToyCollection
```

Delete SNS Topic:

```
aws sns delete-topic --topic-arn arn:aws:sns:ap-south-1::ToyAlerts
```

■ Resources

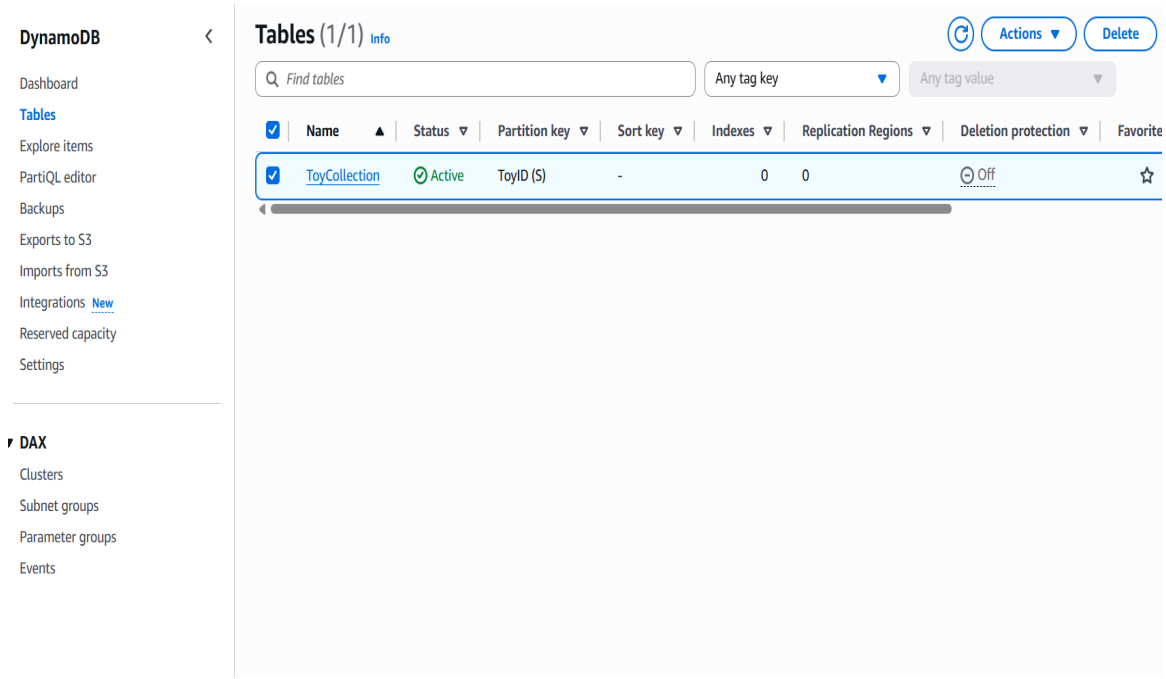
AWS DynamoDB Docs

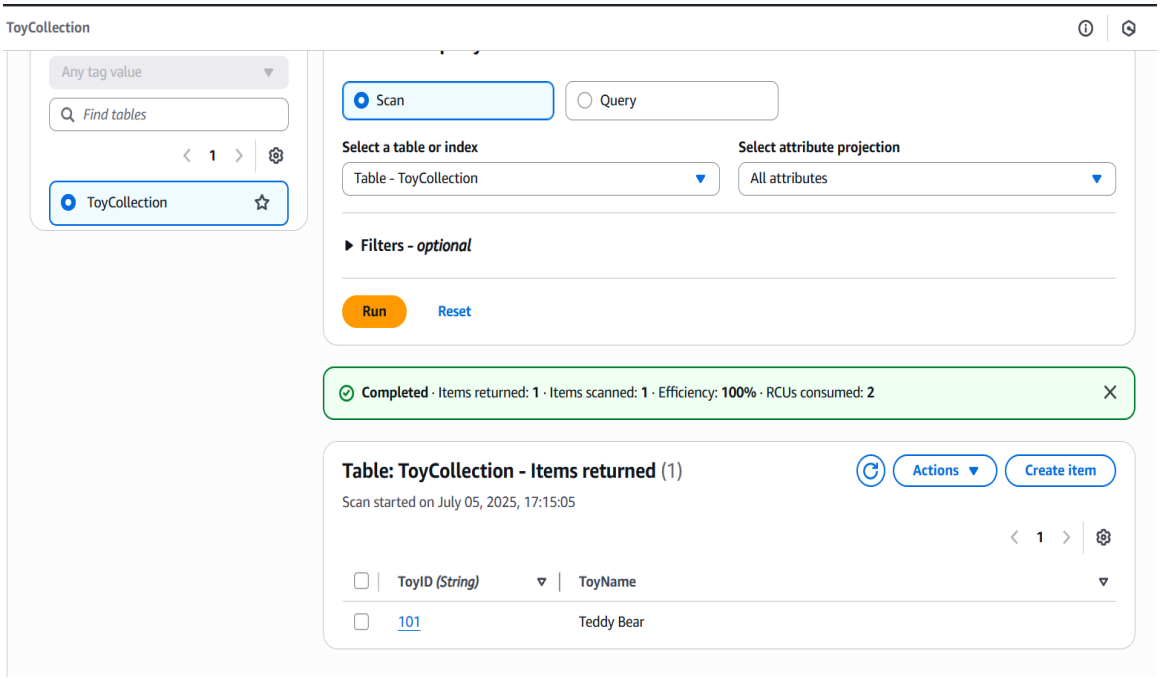
AWS Lambda Docs

AWS SNS Docs

AWS Free Tier Info

■ This project is part of my AWS hands-on journey. Uploaded to GitHub and shared on LinkedIn to document my learning. #AWS #Serverless #CloudProjects #DevOps #LearningByDoing


DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New
Reserved capacity
Settings
DAX
Clusters
Subnet groups
Parameter groups
Events
Tables (1/1) Info
Actions
Delete
Find tables
Any tag key
Any tag value
Name
Status
Partition key
Sort key
Indexes
Replication Regions
Deletion protection
Favorite
ToyCollection
Active
ToyID (S)
-
0
0
Off


ToyCollection
Any tag value
Find tables
1
ToyCollection
Scan
Query
Select a table or index
Table - ToyCollection
Select attribute projection
All attributes
Filters - optional
Run
Reset
Completed · Items returned: 1 · Items scanned: 1 · Efficiency: 100% · RCUs consumed: 2
Table: ToyCollection - Items returned (1)
Scan started on July 05, 2025, 17:15:05
Actions
Create item
1
ToyID (String)
ToyName
101
Teddy Bear

Learn to create AI applications with AWS Lambda
From agentic AI and chatbots to intelligent document processing and content generation.

Get started

Functions (1)

Last fetched 0 seconds ago

Actions

Create function

Filter by attributes or search by keyword

1

Function name	Description	Package type	Runtime	Last modified
AddToyToDynamoDB	-	Zip	Python 3.12	5 minutes ago

Lambda > Functions > AddToyToDynamoDB

AddToyToDynamoDB

EXPLORER

ADDTOYTODYNAMODB

lambda_function.py

DEPLOY

Deploy (Ctrl+Shift+U)

Test (Ctrl+Shift+I)

TEST EVENTS [SELECTED: ADDTEDDY]

Create new test event

Private saved events

AddTeddy

ENVIRONMENT VARIABLES

lambda_function.py

```python
import json
import boto3

dynamodb = boto3.resource('dynamodb')
sns = boto3.client('sns')

table = dynamodb.Table('ToyCollection')

TOPIC_ARN = 'arn:aws:sns:ap-south-1:531504455335:ToyAlerts'  # Replace this

def lambda_handler(event, context):
    toy_id = event['ToyID']
    toy_name = event['ToyName']

    # Write to DynamoDB
    table.put_item(
        Item={
            "ToyID": toy_id,
            "ToyName": toy_name
        }
    )

    # Send email via SNS
    message = f"A new toy was added: {toy_name} (ID: {toy_id})"

    sns.publish(
        TopicArn=TOPIC_ARN
```

0 0 Amazon Q

DEPLOY

Deploy (Ctrl+Shift+U)

Test (Ctrl+Shift+I)

TEST EVENTS [SELECTED: ADDTEDDY]

Create new test event

Private saved events

AddTeddy

ENVIRONMENT VARIABLES

```
29        Message=message
30    )
31
32    return {
33        'statusCode': 200,
34        'body': json.dumps("Toy added and notification sent!")
35    }
36
```

PROBLEMS OUTPUT CODE REFERENCE LOG TERMINAL

```
Status: Succeeded
Test Event Name: AddTeddy

Response:
{
  "statusCode": 200,
  "body": "\"Toy added and notification sent!\""
}
```

0 0 Amazon Q

AddToyToDynamoDB-role-v6vihon7

AddToyToDynamoDB-role-v6vihon7 Info

Delete

Summary

Edit

Creation date
July 05, 2025, 16:50 (UTC+05:30)

ARN
arn:aws:iam::531504455335:role/service-role/AddToyToDynamoDB-role-v6vihon7

Last activity
18 minutes ago

Maximum session duration
1 hour

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (3) Info

You can attach up to 10 managed policies.

Simulate | Remove | Add permissions

Filter by Type

Search | All types

1

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	1
AmazonSNSFullAccess	AWS managed	2
AWSLambdaBasicExecutionRole-8ab55516...	Customer managed	1

aws | Search [Alt+S] | Asia Pacific (Mumbai)

Amazon SNS > Topics > ToyAlerts

Amazon SNS

- Dashboard
- Topics
- Subscriptions
- Mobile
  - Push notifications
  - Text messaging (SMS)

Amazon SNS now supports High Throughput FIFO topics. Learn more

ToyAlerts

Edit | Delete

Details

Name ToyAlerts	**Display name** ToyAlerts
ARN arn:aws:sns:ap-south-1:531504455335:ToyAlerts	**Topic owner** 531504455335
Type Standard	

Subscriptions | Access policy | Data protection policy | Delivery policy (HTTP/S) | Delivery status logging | Encry

Subscriptions (1)

Edit | Delete | Request confirmation | Confirm subscription

Search

ID	Endpoint	Status	Protocol
24c2e065-6d2d-4d5f-a85f-75710...	ronymitra117@gmail.com	Confirmed	EMAIL

New Toy Added

Inbox ×

ToyAlerts <no-reply@sns.amazonaws.com>
to me

A new toy was added: Teddy Bear (ID: 101)

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.ap-south-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:ap-south-1:531504455335:ToyAlerts:a117@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at htt