

Assignment Number 10

Name: Deep Prajapati; Branch: I.T (T.E.); Roll Number: 96

17/10/2023

Aim: To create a Lambda function using Python for adding data to Dynamo DB database.

LO mapped: LO6

Theory:

Step 1: Set Up AWS Environment

1. **Create an AWS Account:** If you don't have an AWS account, sign up for one at [AWS Console](#).
2. **Access AWS Lambda Console:**

The screenshot shows the AWS Lambda console interface for creating a new function. The 'Table name' section has a text input field containing 'demoytfirst'. The 'Partition key' section has a text input field containing 'roll_id' and a dropdown menu set to 'String'. The 'Sort key - optional' section has a text input field and a dropdown menu set to 'String'. The 'Settings' section at the bottom has two radio buttons: 'Default settings' (selected) and 'Customize settings'.

- Go to the [AWS Lambda Console](#).
- Click on "Create function."

3. Create a New Lambda Function

The image shows two screenshots from the AWS Management Console. The top screenshot displays the 'DynamoDB' console with a notification that 'The demoyfirst table was created successfully.' Below this, the 'Tables (1)' section shows a table named 'demoyfirst' with a partition key of 'roll_no (Number)' and a status of 'Active'. The bottom screenshot shows the 'Create role' page, specifically the 'Attach permissions policies' step. It lists several policies, with 'AmazonDynamoDBFullAccess' selected. The 'Used as' column for this policy is 'None'.

Top Screenshot: DynamoDB Tables

Notification: The demoyfirst table was created successfully.

Tables (1) Info

Find tables by table name: Any table tag

Name	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode
demoyfirst	Active	roll_no (Number)	-	0	Provisioned with auto scaling (5)	Provisioned with

Bottom Screenshot: Create role

Attach permissions policies

Choose one or more policies to attach to your new role.

Create policy

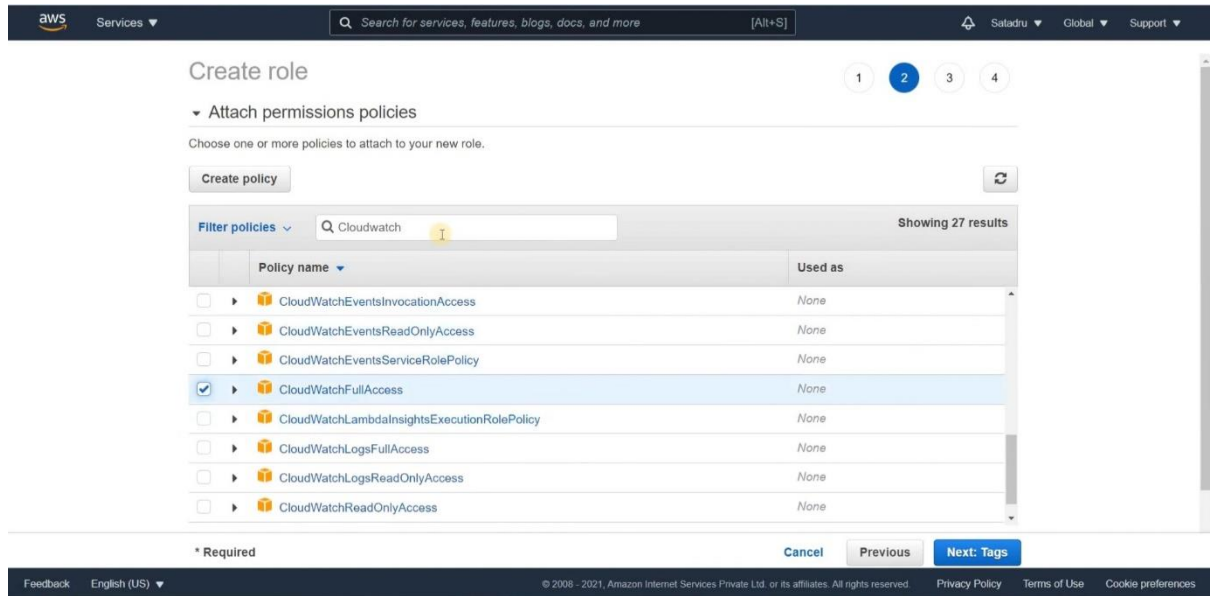
Filter policies: Q Dynamo Showing 8 results

Policy name	Used as
<input checked="" type="checkbox"/> AmazonDynamoDBFullAccess	None
<input type="checkbox"/> AmazonDynamoDBReadOnlyAccess	None
<input type="checkbox"/> AWSApplicationAutoscalingDynamoDBTablePolicy	Permissions policy (1)
<input type="checkbox"/> AWSLambdaDynamoDBExecutionRole	None
<input type="checkbox"/> AWSLambdaInvocation-DynamoDB	None
<input type="checkbox"/> DynamoDBCloudWatchContributorInsightsServiceRolePolicy	None
<input type="checkbox"/> DynamoDBKinesisReplicationServiceRolePolicy	None
<input type="checkbox"/> DynamoDBReplicationServiceRolePolicy	None

* Required Cancel Previous Next: Tags

- Choose "Author from scratch."
- Enter a name for your function (e.g., **AddToDynamoDBFunction**).
- Choose "Python" as the runtime.

4. Configure Execution Role



- Create a new role with basic Lambda permissions and additional permissions to interact with DynamoDB.
- Attach the role to your Lambda function.

Step 2: Set Up DynamoDB

1. Access DynamoDB Console:

- Go to the [AWS DynamoDB Console](#).

The image displays two screenshots of the AWS Lambda console interface.

Top Screenshot: Permissions Section

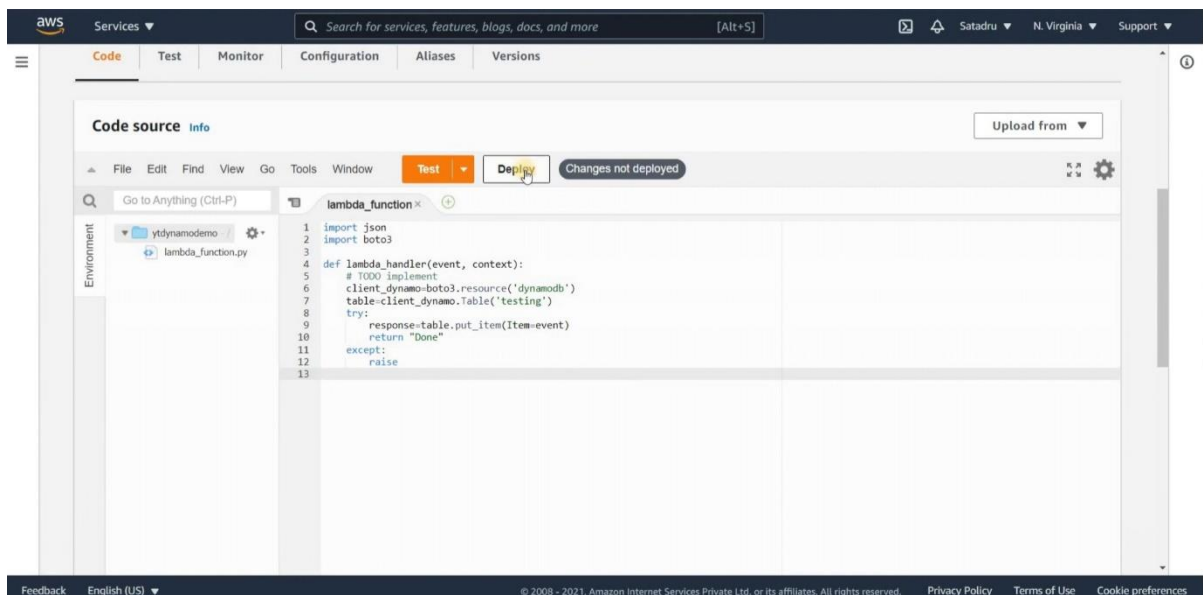
- Architecture:** x86_64 (selected), arm64.
- Permissions:**
 - By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.
 - Change default execution role:**
 - Execution role: Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).
 - ☐ Create a new role with basic Lambda permissions
 - ☒ Use an existing role
 - ☐ Create a new role from AWS policy templates
 - Existing role: Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
 - Selected role: `lambdaRoleDynamo`
 - [View the lambdaRoleDynamo role on the IAM console.](#)
- Advanced settings:** (collapsed)
- Buttons: Cancel, Create function

Bottom Screenshot: Basic information Section

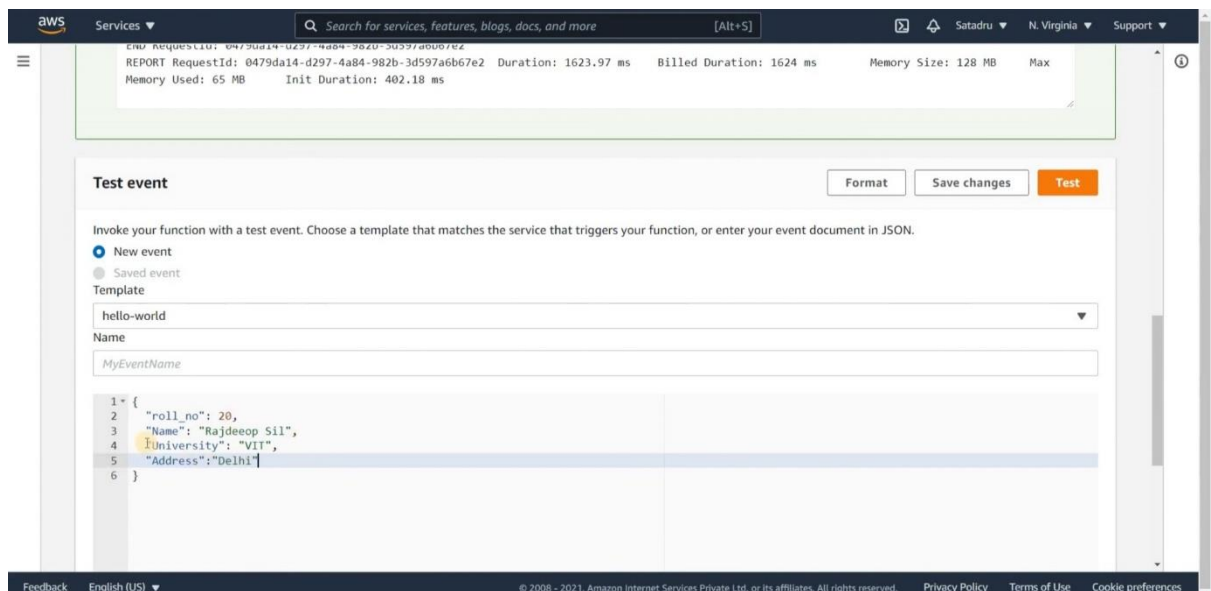
- Function name:** ytdynamodemo (with instructions: Enter a name that describes the purpose of your function. Use only letters, numbers, hyphens, or underscores with no spaces.)
- Runtime:** Python 3.9 (with note: Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.)
- Architecture:** x86_64 (selected), arm64.
- Permissions:** (same as top screenshot)
- Change default execution role:** (collapsed)
- Advanced settings:** (collapsed)
- Buttons: Cancel, Create function

2. Create a DynamoDB Table:

- Click on "Create table."
- Enter a table name and a primary key (e.g., **ID**).
- Configure other settings as needed and create the table.



Step 3: Write Lambda Function Code



1. Ensure to replace '**YourDynamoDBTableName**' with the actual name of your DynamoDB table.
2. **Configure Lambda Handler:**
 - In the Lambda function configuration, set the handler to **filename.lambda_handler** (e.g., **yourfilename.lambda_handler**).

Step 4: Configure Environment Variables

1. **Add DynamoDB Table Name:**
 - In the Lambda function configuration, add an environment variable (e.g., **DYNAMODB_TABLE**) with the value set to your DynamoDB table name.

Step 5: Test Locally

1. Test Lambda Function Locally:

- Create a test event with sample data.
- Test the Lambda function using the "Test" button in the Lambda console.

Step 6: Deploy Lambda Function

1. Deploy the Lambda Function:

- Click on the "Deploy" button in the Lambda console.

Step 7: Test in AWS Lambda Console

1. Test in AWS Lambda Console:

- Use the Lambda console to test the function by creating a test event with sample data.
- Verify that the function executes successfully.

Step 8: Monitor and Troubleshoot

1. Check CloudWatch Logs:

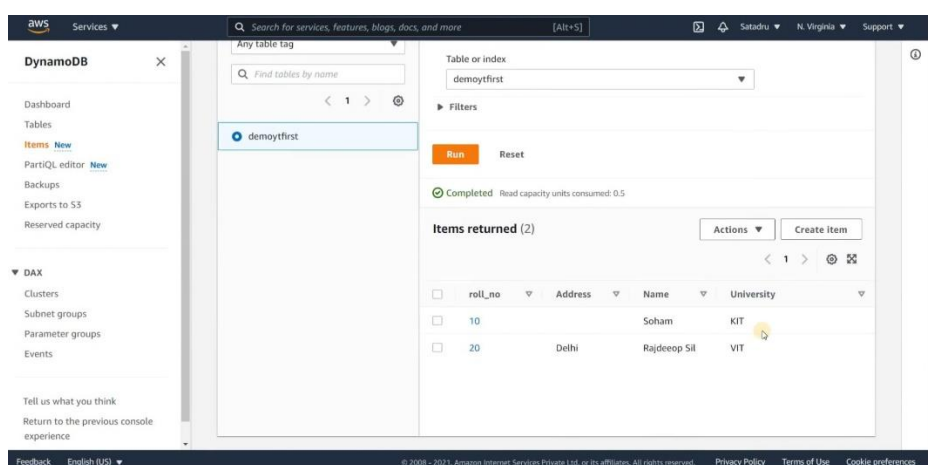
- Use CloudWatch Logs to check logs generated by your Lambda function for troubleshooting.

Step 9: Documentation

1. Create Documentation:

- Document the purpose, input parameters, and expected output of your Lambda function.
- Explain any challenges faced during development and how they were resolved.

By following these steps, you can create a Lambda function using Python to add data to a DynamoDB database in the AWS environment.



Conclusion: By this assignment we learn how to create a Lambda function using Python for adding data to Dynamo DB database.