

# Assignment Number 10

Name: Soham Satam ; Branch: I.T (T.E.); Roll Number: 109

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:**

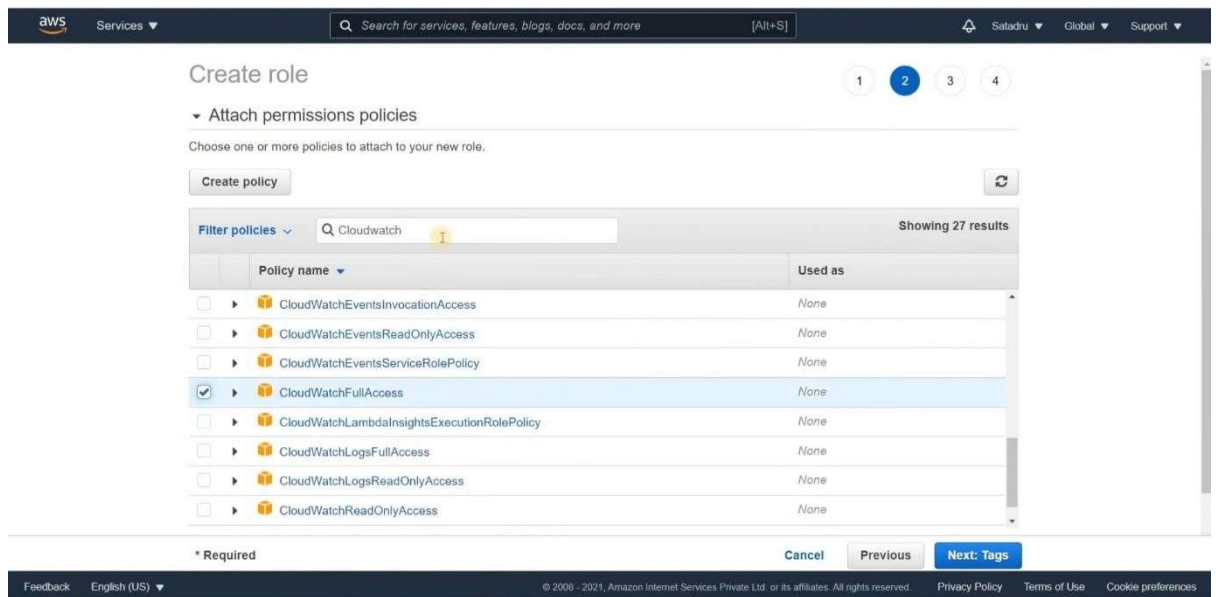
The screenshot shows the AWS Lambda console interface for creating a new function. The 'Table name' field is filled with 'demoytfirst'. The 'Partition key' field is filled with 'roll\_id' and the data type is set to 'String'. The 'Sort key - optional' field is empty. The 'Settings' section has 'Default settings' selected. The footer of the console shows 'Feedback', 'English (US)', '© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

- Go to the [AWS Lambda Console](#).
  - Click on "Create function."
3. **Create a New Lambda Function**

The top screenshot shows the AWS Management Console for DynamoDB. A green notification banner at the top states: "The demoytfirst table was created successfully." The left sidebar shows the navigation menu with options like Dashboard, Tables, Items, PartiQL editor, Backups, Exports to S3, Reserved capacity, DAX, Clusters, Subnet groups, Parameter groups, and Events. The main content area displays the 'Tables (1) Info' section, showing a table named 'demoytfirst' with an 'Active' status, a partition key of 'roll\_no (Number)', and a read capacity mode of 'Provisioned with auto scaling (5)'. The bottom screenshot shows the 'Create role' page, specifically the 'Attach permissions policies' step. It prompts the user to 'Choose one or more policies to attach to your new role.' A search filter 'Dynamo' is applied, showing 8 results. The first result, 'AmazonDynamoDBFullAccess', is selected with a checkmark. Other policies listed include 'AmazonDynamoDBReadOnlyAccess', 'AWSApplicationAutoscalingDynamoDBTablePolicy', 'AWSLambdaDynamoDBExecutionRole', 'AWSLambdaInvocation-DynamoDB', 'DynamoDBCloudWatchContributorInsightsServiceRolePolicy', 'DynamoDBKinesisReplicationServiceRolePolicy', and 'DynamoDBReplicationServiceRolePolicy'. The 'Used as' column for most policies is 'None', except for 'AWSApplicationAutoscalingDynamoDBTablePolicy' which is 'Permissions policy (1)'. The bottom of the page shows navigation buttons: 'Cancel', 'Previous', and '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` (with a refresh icon).
  - Link: [View the lambdaRoleDynamo role on the IAM console.](#)
- Advanced settings:** (collapsed)

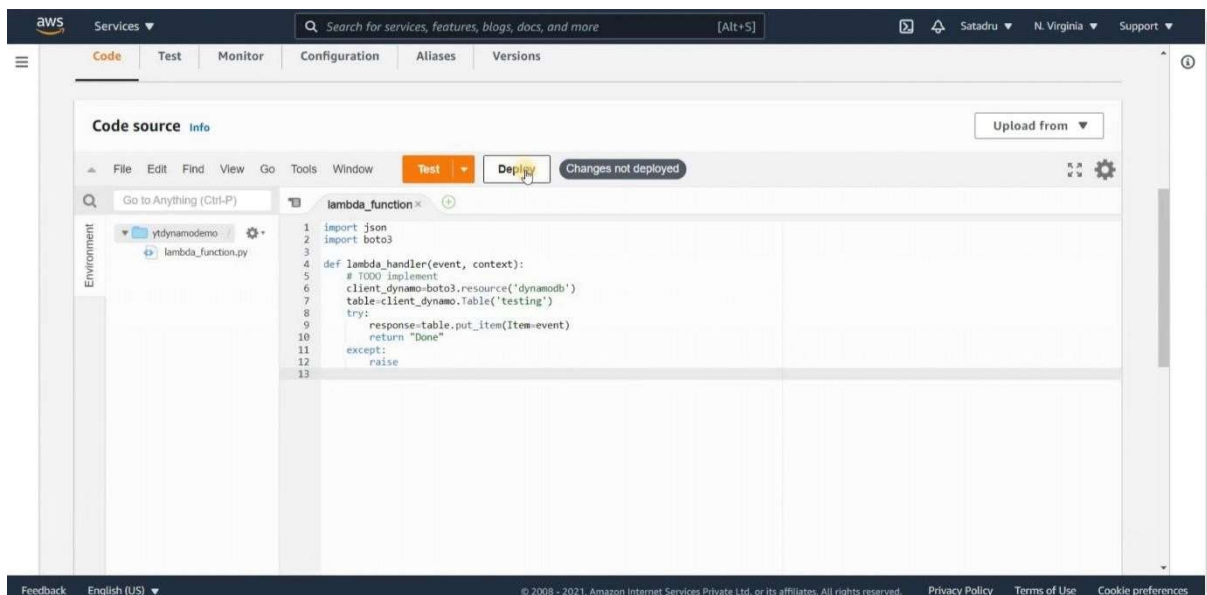
**Bottom Screenshot: Basic information Section**

- Function name:** ytdynamodemo (with a note: "Use only letters, numbers, hyphens, or underscores with no spaces.")
- Runtime:** Python 3.9 (selected from a dropdown menu)
- Architecture:** x86\_64 (selected), arm64.
- Permissions:** (same as the top screenshot)
- Change default execution role:** (collapsed)
- Advanced settings:** (collapsed)

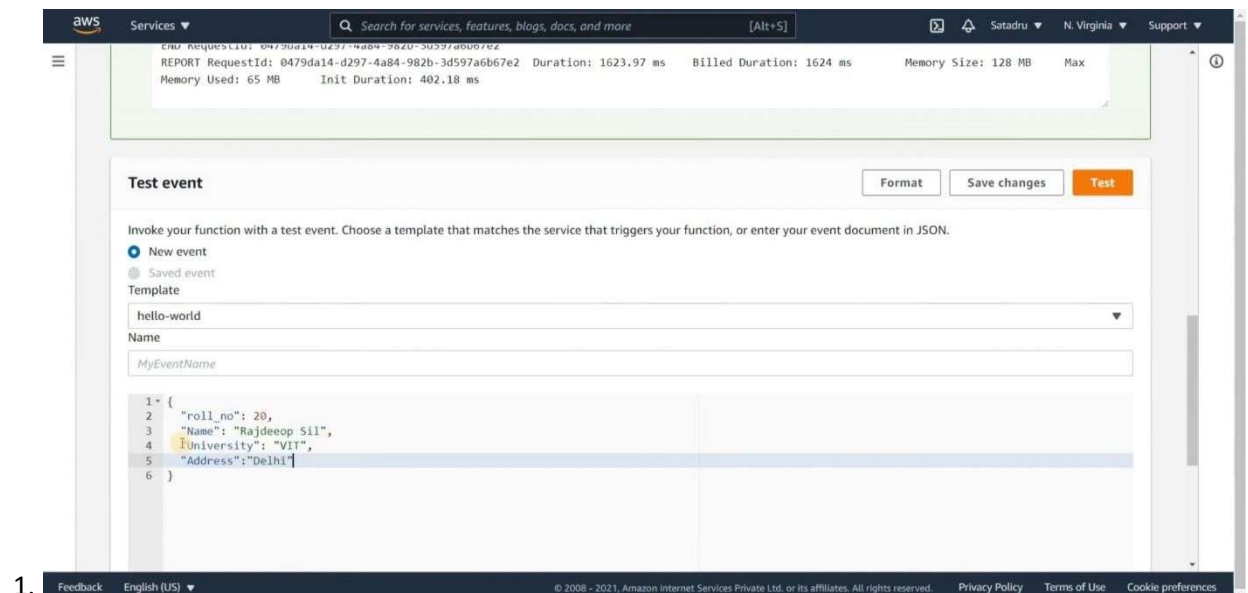
Both screenshots show the "Create function" button at the bottom right.

## 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:

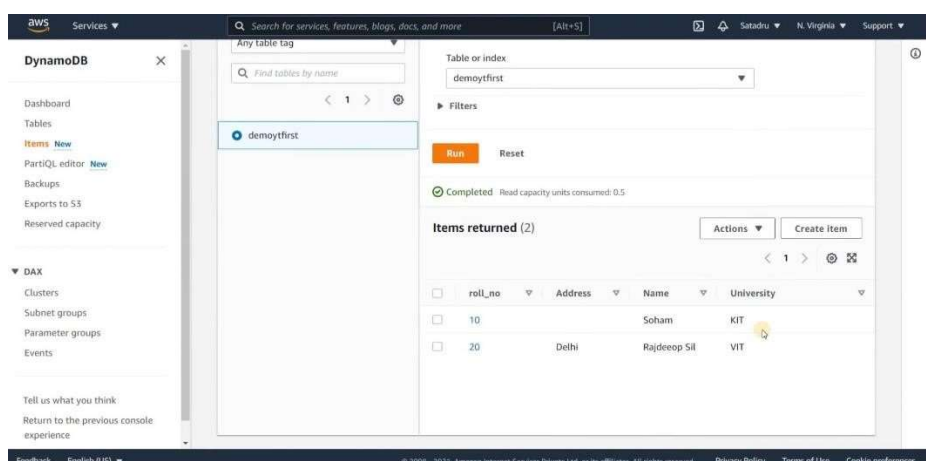
- Use CloudWatch Logs to check logs generated by your Lambda function for troubleshooting.

## 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.