AWS Lambda

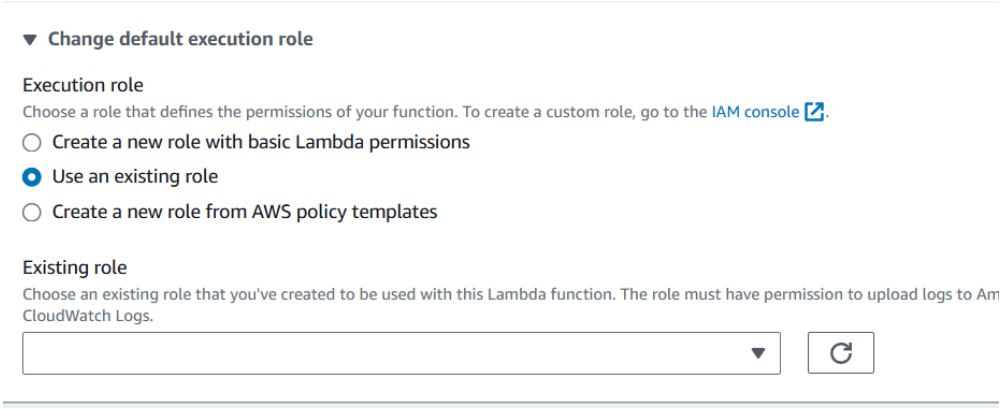
AWS Lambda is a serverless compute service that lets you run code in response to events without provisioning or managing servers. Key points:

- **Serverless**: No server management required.
- **Event-Driven**: Runs code in response to events (e.g., S3 uploads, API Gateway requests).
- **Automatic Scaling**: Scales automatically with workload.
- **Flexible Language Support**: Supports multiple programming languages.
- **Pay-as-You-Go**: Only pay for compute time used.
- **Integrates with AWS Services**: Easily connects with other AWS services.
- **Security**: Uses roles and policies for permissions.

Creating a Lambda function:

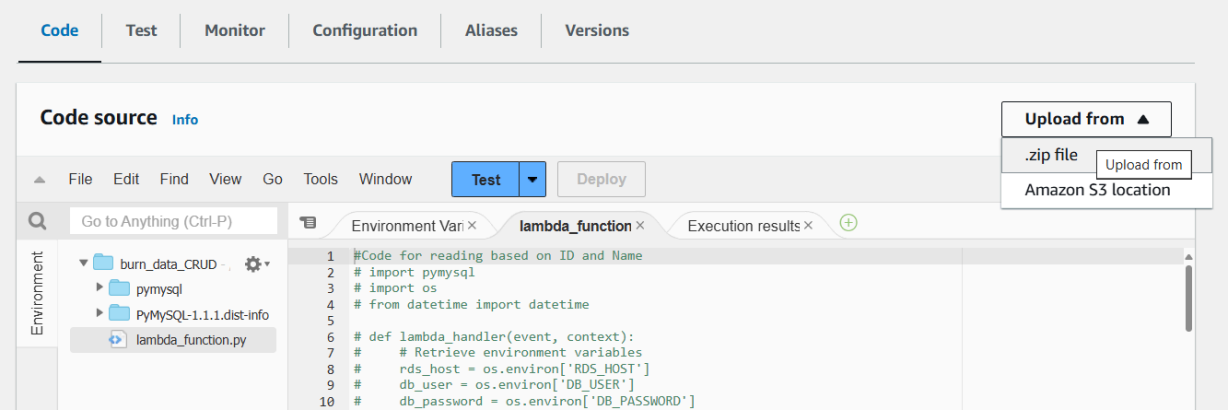
1. **IAM Role and Policy for Lambda function:** Before creating a lambda function we need to create an IAM role for it and attach policy to it. Create an IAM role with the necessary permissions for AWS Lambda to access your RDS instance. To create this follow these steps:
    - Search for “IAM” Service on AWS and select it.
    - Then select the “Roles” tab from the IAM dashboard.
    - Click on “Create Role”. In that fill in the details :
      - **Trusted Entity Type option**: Select AWS Service in this.
      - **Use case**: In this option select Lambda service.
    - After that click on the “Next” button in the bottom
    - Then next page will ask you the policies you want to attach to this Role. As we are going to attach this role with lambda function we are going to create for getting data from RDS for performing CRUD operations. Select these two Policies:
      ‘AWSLambdaBasicExecutionRole’
      A custom policy to access RDS- ‘`AmazonRDSFullAccess`’
    - After adding these again click on “Next” button
    - After this you will get a review page to see if everything is okay and then click on “create role” button
    - So we are done creating IAM Role for our lambda function
2. **Creating Lambda function:** We will create our lambda function. To do this follow these steps:
    - Search for “Lambda” Service on AWS and select it.
    - Click on “Create Function”. In that fill the details:
      Name
      Runtime: For this I am selecting Python as I am using Python
      Architecture

Change default execution role: Click on this and select “Use an Existing role” option and then select the Role you created

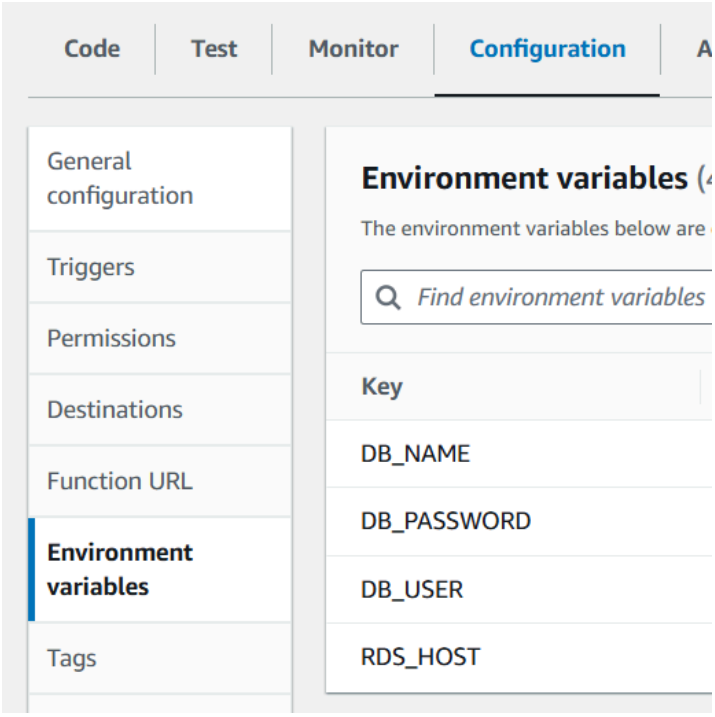


- Now, click on the “Create function” button and the bottom after filling in all the details.

3. **Install MySQL connector:** Lambda functions need to have the MySQL connector installed to interact with your RDS instance. Since Lambda does not come with this package by default, you need to create a deployment package. Follow these steps to do this:
    - Create a folder locally in your PC and navigate into it:
      Code:
      ```
      mkdir lambda_mysql
      cd lambda_mysql
      ```
    - Install the MySQL connector in it:
      Code
      ```
      pip install pymysql -t
      ```
    - Now you can create a Python file containing Code for Lambda functions for your CRUD operations in this folder **OR** you can also do this in the Code section of the Lambda console also by creating a Python file in there (but if you are doing this keep in mind when creating a python file make sure you create in the folder you uploaded which contains MYSQL connector.)
    - I created a Python file containing my code locally in my PC in the folder.
    - Next, zip this folder. While zipping the folder make sure, to double check that when you unzip the folder is should directly contain the pymysql file, it should not be like after unzipping the folder there is another folder inside it and inside that folder there is pymysql file.
      The folder after unzipping structure should be: Lambda_sql -> Pythonfile, pymsql file, etc.
    - Lastly, upload this zip file in Lambda function you created in Lambda console. Under the "Code" section, choose "Upload from" and select ".zip file".



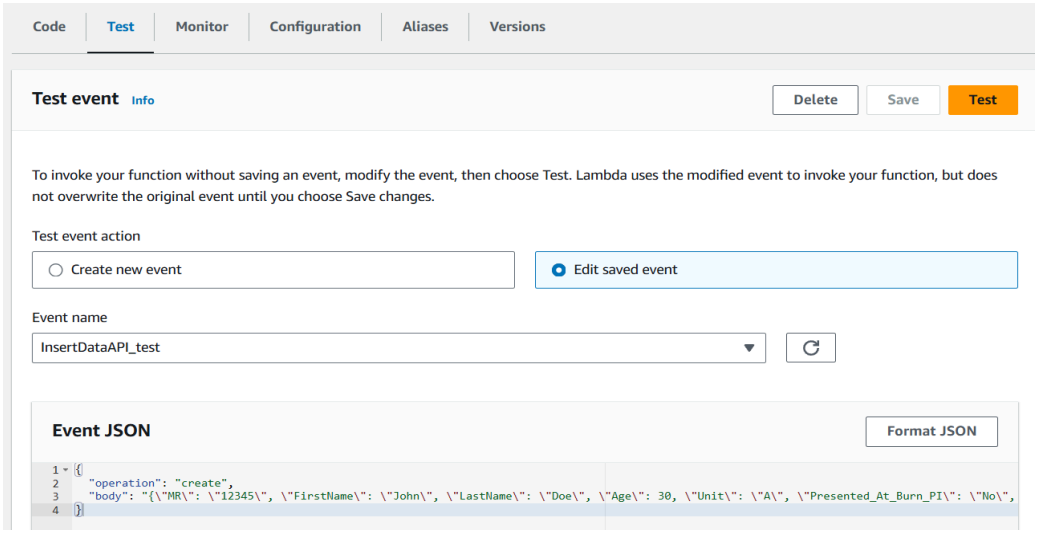
4. **RDS Connection:** In the AWS Lambda console, configure environment variables for your database connection details (`DB_HOST`, `DB_USER`, `DB_PASSWORD`, `DB_NAME`).
   - For this go to the “Configuration” tab in the Lambda Console.
   - In that go to “Environment variables” option and Click on Edit and then add your db connection details.



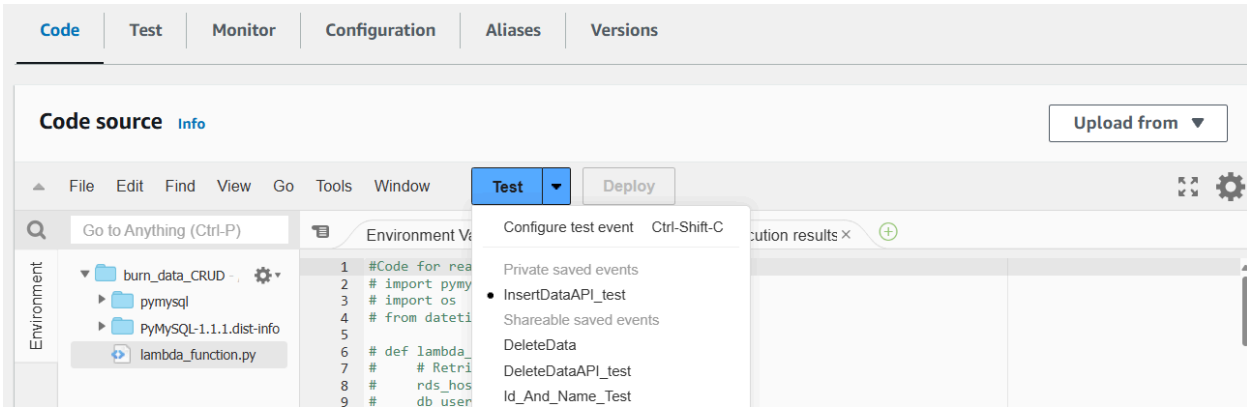
5. **Handler Configuration:** Before testing make sure the handler is properly configured. The handler is specified in the format `filename.function_name`. For example, if you want `unified.py` to be your entry point, and your function inside `unified.py` is named `lambda_handler`, you would set the handler to `unified.lambda_handler` Under the **Function code** section, there is a field called **Handler**. If your main file is `unified.py` and the main function is

`lambda_handler`, you would enter `unified.lambda_handler` into this field.
This tells AWS Lambda to use the `lambda_handler` function inside the `unified.py` file as the entry point whenever the Lambda function is invoked.

6. **Test your Function**: Use the AWS Lambda console to create test events and trigger the functions. Ensure that each function works as expected by performing the corresponding CRUD operations on your RDS instance. To do this:
   - Select “Test” Option form Lambda Console
   - In that create test event for each of your CRUD Operation



   - After this Go the “Code” Tab.
   - Open your Python code file in that. Then select the Test event you want to test and click on Test.



So you have created the lambda function and once you have tested your Lambda functions, you can create a frontend (using HTML/CSS/JavaScript) and integrate it with AWS API Gateway to

call these Lambda functions. This will allow you to perform CRUD operations through your website.

Next we will create API Integration for this Using AWS API Gateway.