### **CRM RDS database**

#### Make a separate Lambda function for CRM database and add this code to your index.js file

'use strict';

const mysql = require('mysql');

const connectToDatabase = () => {

return mysql.createConnection({

host: process.env.db\_host,

user: process.env.db\_username,

password: process.env.db\_password,

database: process.env.DB\_NAME

});

};

const executeQuery = (connection, query, values = []) => {

return new Promise((resolve, reject) => {

connection.query(query, values, (error, results) => {

connection.end(); // Close the connection

if (error) {

reject(error);

} else {

resolve(results);

}

});

});

};

// Create Customer Handler

exports.createCustomerHandler = async (event) => {

console.log('Received event for createCustomer:', JSON.stringify(event, null, 2));

const body = JSON.parse(event.body);

const { first\_name, last\_name, email, phone, address, city, state, postal\_code, country } = body;

try {

const connection = connectToDatabase();

const query = 'INSERT INTO crm\_customers (first\_name, last\_name, email, phone, address, city, state, postal\_code, country) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)';

const values = [first\_name, last\_name, email, phone, address, city, state, postal\_code, country];

const results = await executeQuery(connection, query, values);

console.log('Query executed successfully:', results);

return {

statusCode: 201,

body: JSON.stringify({ customer\_id: results.insertId, ...body })

};

} catch (error) {

console.error('Error executing query:', error);

return {

statusCode: 500,

body: JSON.stringify({ error: error.message })

};

}

};

// Get Customers Handler

exports.getCustomersHandler = async (event) => {

console.log('Received event for getCustomers:', JSON.stringify(event, null, 2));

try {

const connection = connectToDatabase();

const query = 'SELECT \* FROM crm\_customers';

const results = await executeQuery(connection, query);

console.log('Query executed successfully:', results);

return {

statusCode: 200,

body: JSON.stringify(results)

};

} catch (error) {

console.error('Error executing query:', error);

return {

statusCode: 500,

body: JSON.stringify({ error: error.message })

};

}

};

exports.updateCustomerHandler = async (event) => {

console.log('Received event:', JSON.stringify(event, null, 2));

const body = JSON.parse(event.body);

const { first\_name, last\_name, email, phone, address, city, state, postal\_code, country } = body;

const { customer\_id } = event.pathParameters;

try {

const connection = connectToDatabase();

const query = 'UPDATE crm\_customers SET first\_name = ?, last\_name = ?, email = ?, phone = ?, address = ?, city = ?, state = ?, postal\_code = ?, country = ? WHERE customer\_id = ?';

const values = [first\_name, last\_name, email, phone, address, city, state, postal\_code, country, customer\_id];

const results = await executeQuery(connection, query, values);

console.log('Query executed successfully:', results);

return {

statusCode: 200,

body: JSON.stringify({ customer\_id, ...body })

};

} catch (error) {

console.error('Error executing query:', error);

return {

statusCode: 500,

body: JSON.stringify({ error: error.message })

};

}

};

#### This is your deploy.yml file

name: Deploy Serverless Project

on:

push:

branches:

- main

workflow\_dispatch:

schedule:

- cron: '0 0 \* \* \*' # Daily at midnight

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Set up Node.js

uses: actions/setup-node@v2

with:

node-version: '18.x' # Updated to match the version used in serverless.yml

- name: Install dependencies

run: npm install

- name: Install MySQL and AWS SDK

run: npm install mysql @aws-sdk/client-secrets-manager

- name: Deploy to AWS

run: npx serverless deploy

env:

AWS\_ACCESS\_KEY\_ID: ${{ secrets.AWS\_ACCESS\_KEY\_ID }}

AWS\_SECRET\_ACCESS\_KEY: ${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

SERVERLESS\_ACCESS\_KEY: ${{ secrets.SERVERLESS\_ACCESS\_KEY }}

#### **Serverless installations should be there**

#### **Connect your RDS instance with the Lambda function**

#### **Add this IAM inline policy to your Lambda function**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"secretsmanager:GetSecretValue",

"secretsmanager:PutSecretValue",

"secretsmanager:UpdateSecretVersionStage",

"secretsmanager:DescribeSecret"

],

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": [

"ssm:GetParameters",

"ssm:GetParameter",

"ssm:GetParametersByPath"

],

"Resource": [

"arn:aws:ssm:ap-south-1:815469512242:parameter/project/aws-node/\*"

]

},

{

"Effect": "Allow",

"Action": [

"rds:ModifyDBInstance",

"rds-db:connect"

],

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": [

"lambda:InvokeFunction"

],

"Resource": [

"arn:aws:lambda:ap-south-1:815469512242:function:aws-node-project-crm-service-createCustomer",

"arn:aws:lambda:ap-south-1:815469512242:function:aws-node-project-crm-service-getCustomers",

"arn:aws:lambda:ap-south-1:815469512242:function:aws-node-project-crm-service-updateCustomer"

]

},

{

"Effect": "Allow",

"Action": [

"logs:CreateLogStream",

"logs:CreateLogGroup",

"logs:PutLogEvents"

],

"Resource": [

"arn:aws:logs:ap-south-1:815469512242:log-group:/aws/lambda/aws-node-project-crm-service-createCustomer:\*",

"arn:aws:logs:ap-south-1:815469512242:log-group:/aws/lambda/aws-node-project-crm-service-getCustomers:\*",

"arn:aws:logs:ap-south-1:815469512242:log-group:/aws/lambda/aws-node-project-crm-service-updateCustomer:\*"

]

}

]

}