Project 3 – Implementation for REST API Endpoints using AWS Lambda

Due Date: Monday, April 15 at 11:59 PM

Description:

In this project, you will develop a set of AWS lambda functions for a database backed web-based application. Assume you have a set of REST APIs that the frontend of your application uses to communicate with the backend. Refer to the following diagram for the deployment architecture of the APIs on AWS.

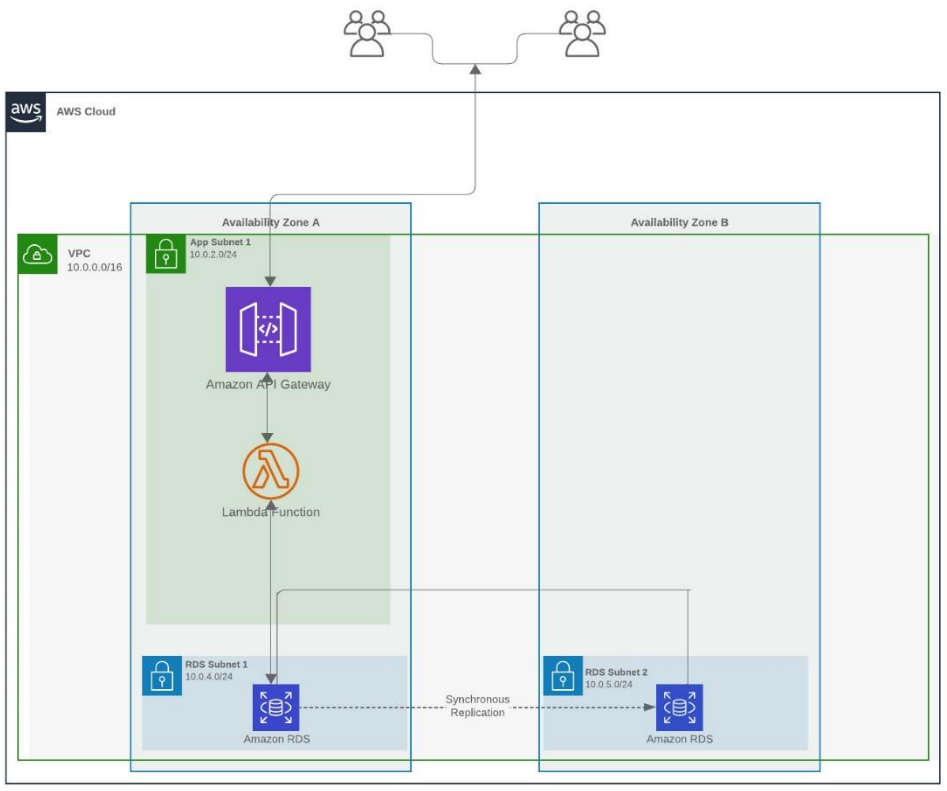

Fig 1: Deployment Architecture

You are going to define the following API end points using the AWS lambda functions.

- **getAllCategory** – fetch all the categories from the database
- **getCategoryName** - given the category id fetch the name of the category
- **getCategoryId** - fetch the ID of the category with the given category name
- **addCategory** - add the given category, with category Id and category name, in the database
- **getAllBook** - fetch all the books from the database

- **getBookById** – fetch the book given the book id
- **getBookByCategoryId** - fetch all books of the given category id
- **getBookByCategoryName** - fetch all books of the given category name
- **getRandomBook** - fetch 5 random books from the database
- **addBook** – add the book in the database given the book information

Note that all these lambda functions should be accessible through Amazon API Gateway all data should be in JSON format.

Starter Code

The schema of the database together with the tables with some sample records is provided for this project. You are free to add additional records in the tables.

Requirements

- All common dependencies for your lambda functions should be deployed as layers. Note that any library/dependency that is used by more than one function is a good candidate to be added as a layer.
- You have to define appropriate IAM policies and roles whenever necessary. The triggers need to have the proper authorization to invoke the lambda function and the lambda function needs the same to communicate with other AWS services such as RDS.
- You should not include any access key information directly in your code.
- You are free to use any programming language and the corresponding runtime for this project.
- You can also use any database management system that RDS supports.
- You are not required to develop the frontend, but if you do that will be ok.
- Your lambda functions should write appropriate information to the log.

Restriction

Before you deploy your functions on AWS, work on your solution using the programming language of your choice on your local machine. Once you are sure your solution works in your local machine, we will give you the authorization to create resources and use the necessary services on AWS. Please make sure you have a working solution before you ask for the AWS privilege.

You **MUST** follow the following rules strictly. **Failure to follow the rules will result in some points deductions from your grade**.

- Only one RDS database per student should be created at a time. Name your resources as p3-[yourName].
- All your functions should follow the naming convention p3-[nameofFunction]-[YourName]
- Make sure you give a reasonable timeout duration and memory size for your functions. AWS calculates the cost for your functions based on mainly these values.
- Use the region you were assigned for Project 1 to deploy your solution.
- DO NOT create a new VPC and Subnet. Use the default ones. There is no need of Elastic IP for this project.

Submission:

- Submission is via Canvas.
- A zip folder that contains:
    - A notepad file that contains the url to access your lambda functions from a browser
    - Your code

AWS

- All the functions, the database and the logs should be accessible on AWS.

Note: You are going to depend on the AWS documentation for this project and we recommend you to start working on the project early.

Creating your Database using RDS:

Go To RDS

- Create Database
- Select Standard Create
- Choose MySQL
- Select the Engine version (preferably the latest one)
- Templates
  - Choose Free Tier
- Setting
  - DB Instance identifier
    - Give your Database name - **[Group_Number]databasename**
  - Credential setting
  - Give username
  - Credential management
    - Choose self-manage
  - Give master password of your database
- Instance configuration
  - Use the given default values
- Storage
  - Use the given default values for storage type and size
  - Storage autoscaling
  - Deselect storage autoscaling
- Connectivity
  - Virtual Private Network (VPC)
    - Choose create a new VPC
  - Use the given default values for the rest
- Additional configuration
  - Make sure port 3306 is listed
- Tags
  - Give tags
- Authentication
  - Choose IAM and Password authentication