Orchestrating Lambda Functions Using AWS Step Functions

Introduction

AWS Lambda functions can be integrated within Step Functions state machines to orchestrate workflow and create conditional logic. In this AWS hands-on lab, you'll be building two Lambda functions and orchestrating the functions using Step Functions. Step Functions use numerical scores provided by students to determine the appropriate rewards to provide to the students by invoking the respective Lambda function. The orchestration of the workflow will rely on Step Functions calling the appropriate Lambda function depending on the inputted score.

Solution

Log in to the AWS Management Console using the credentials provided on the lab instructions page. Make sure you're using the `us-east-1` Region.

Set Up S3

1. Navigate to the S3 console by searching for and selecting **S3** in the top search bar, or selecting it from **Recently Visited**.
2. On the right, click **Create bucket**.
3. Under **Bucket name**, enter *guru-rewards-datafeed* followed by some characters to make it unique.
4. Leave the other settings at default, and click **Create bucket**.
5. Under **Buckets**, click on the **guru-rewards-datafeed** bucket.
6. Under **Objects**, click **Create folder**.
7. Under **Folder name**, enter *guru-premium-courses*.
8. Click **Create folder**.
9. Click **Create folder** again; under **Folder name**, enter *guru-premium-lessons*.
10. Click **Create folder**.

Populate the Buckets with CSV Files

1. To populate the buckets with the CSV files, navigate to the GitHub repository and download the CSV files.

   > **Note:** If necessary, review the How to Download a CSV from GitHub page.

2. Click on the **guru-premium-courses** folder.
3. In the upper right corner, click **Upload**.
4. Next to **Files and folders**, click **Add files**.
5. Select the **premium-courses.csv** file, and click **Open**.
6. Click **Upload**.
7. In the upper right corner, click **Close**.
8. In the upper left breadcrumb trail, click on your bucket name.
9. Under **Objects**, click **guru-premium-lessons**.

10. In the upper right corner, click **Upload**.
11. Next to **Files and Folders**, click **Add files**.
12. Select the **premium-lessons.csv** file, and click **Open**.
13. Click **Upload**.

Create the Lambda Functions

Create the First Lambda Function

1. In the search bar at the top, search for and select **Lambda**.
2. In the upper right corner, click **Create function**.
3. Under **Create function**, ensure **Author from scratch** is selected.
4. Under **Basic information**, set the following values:
   - **Function name**: Enter *UnlockPremiumCourses*.
   - **Runtime**: Select **Python 3.9**.
   - **Architecture**: Select **x86_64**.
5. Leave the other settings at default, and click **Create function**; this may take a few minutes.
6. Once the function is created, under **Code source** > **lambda_function**, delete the code and paste in the code found in the GitHub repository under `unlock_premium_courses_lambda_function`.
7. Update the bucket name on line `4` to match your bucket name (e.g., guru-rewards-datafeed).
8. At the top, click **Deploy** to save the changes.
9. Toward the top, under **Function overview**, click **Layers**.
10. Next to **Layers**, click **Add a layer**.
11. Under **Choose a layer**, set the following values:
    - **Layer source**: Ensure **AWS layers** is selected.
    - **AWS layers**: Select **AWSSDKPandas-Python39**.
    - **Version**: Select the available version.
12. Click **Add**.
13. Under **Function overview**, on the right, copy the function ARN and paste it in a separate text file for use later.

Create the Second Lambda Function

1. In the upper left breadcrumb trail, click **Functions**.
2. In the upper right corner, click **Create Function**.
3. Under **Create function**, ensure **Author from scratch** is selected.
4. Under **Basic Information**, set the following values:
   - **Function name**: Enter *UnlockPremiumLessons*.
   - **Runtime**: Select **Python 3.9**.
   - **Architecture**: Select **x86_64**.
5. Leave the other settings at default, and click **Create function**; this may take a few minutes.
6. Toward the top, under **Function overview**, click **Layers**.
7. Next to **Layers**, click **Add a layer**.
8. Under **Choose a layer**, set the following values:
   - **Layer source**: Ensure **AWS layers** is selected.
   - **AWS layers**: Select **AWSSDKPandas-Python39**.
   - **Version**: Select the available version.

9. Click **Add**.
10. Under **Code source** > **lambda_function**, delete the code and paste in the code found in the GitHub repository under `unlock_premium_lessons_lambda_function.py`.
11. Update the bucket name on line `4` to match your bucket name.
12. At the top, click **Deploy** to save the changes.
13. Under **Function overview**, on the right, copy the function ARN and paste it in a separate text file for use later.

Grant Functions Access to S3

1. Click on the **Configuration** tab.
2. In the left navigation menu, under **General configuration**, click **Permissions**.
3. Under **Execution role**, click on the listed role name.
4. Next to **Permissions policies**, click on the **Add permissions** dropdown menu and select **Attach policies**.
5. In the upper right corner, click **Create policy**.
6. In the **Visual editor** tab, click on **Service** to expand the menu.
7. In the search bar, search for and select **S3**.
8. Expand the **Actions** menu. Under **Access level**, expand the **List** sub-menu, and select **ListBucket**.
9. Expand the **Read** sub-menu, and select **GetObject**.
10. Expand the **Resources** menu.
11. Next to **bucket**, click on the **Add ARN** hyperlink.
12. In the **Add ARN(s)** pop-up menu, next to **Bucket name**, enter the name of your bucket.
13. Copy your bucket name, then click **Add**.
14. Next to **object**, click on the **Add ARN** hyperlink and set the following values:
    - **Bucket name**: Paste in your bucket name.
    - **Object name**: Select **Any**.
15. Click **Add**.
16. Click **Next** until you get to **Create policy**.
17. Next to **Name**, enter *ReadS3RewardsDatafeed*; copy **ReadS3RewardsDatafeed** for later.
18. Click **Create policy**.

Attach Policies

1. In the left navigation menu under **Access management**, click **Roles**.
2. In the **Roles** search bar, enter *courses*.
3. Click on the **UnlockPremiumCourses** role.
4. Next to **Permissions policies**, click on the **Add permissions** dropdown menu and select **Attach policies**.
5. In the **Other permissions policies** search bar, paste in *ReadS3RewardsDatafeed* and press **Enter**.
6. Click on the checkbox next to **ReadS3RewardsDatafeed**.
7. Click **Attach policies**.
8. In the upper left breadcrumb trail, click **Roles**.
9. In the **Roles** search bar, enter *lesson*, and select the **UnlockPremiumLessons** role.
10. Next to **Permissions policies**, click on the **Add permissions** dropdown menu and select **Attach policies**.
11. In the **Other permissions policies** search bar, paste in **ReadS3RewardsDatafeed** and press **Enter**.

12. Click on the checkbox next to **ReadS3RewardsDatafeed**.
13. Click **Attach policies**.

Set Up IAM Roles

1. In the left navigation menu, under **Access management**, click **Roles**.
2. In the upper right corner, click **Create role**.
3. Under **Trusted entity type**, ensure **AWS service** is selected.
4. Under **Use case**, beneath **Use cases for other AWS services**, click on the dropdown menu and select **Step Functions** (in the search bar, you can search *Step Functions*).
5. Underneath the **Step Functions** dropdown menu, click on the radio button next to **Step Functions**.
6. Click **Next**.
7. Under **Permissions policies**, click on the **+** icon next to **AWSLambdaRole** to confirm the Step Functions can invoke Lambda.
8. Click **Next**.
9. Under **Role details** > **Role name**, enter *StepFunctionsInvokeLambdaRole*.
10. Leave the other settings at default, scroll down, and click **Create role**.

Set Up AWS Step Functions

1. In the search bar at the top, search for and select **Step Functions**.
2. Under **Get started** on the right, click **Get started**.
3. In the line of text under **Review Hello World example**, click on the **here** hyperlink.
4. Under **Choose authoring method**, select **Write your workflow in code**.
5. Under **Definition**, delete the code and paste in the code found in the GitHub repository under `state-machine-code.json`.
6. Scroll down to line `23`, and replace `Insert the ARN of your UnlockPremiumCoursesFunction` with the corresponding ARN you previously pasted in the separate text file.
7. On line `29`, replace `Insert the ARN of your UnlockPremiumLesson` with the corresponding ARN you previously pasted in the separate text file.
8. Click **Next**.
9. Under **Name**, enter *RewardsProcessorStateMachine*.
10. Under **Permissions**, select **Choose an existing role**.
11. Leave the other settings at default, and click **Create state machine**.

Test the Step Functions

1. In the upper right corner, click **Start execution**.
2. In the **Start execution** pop-up menu, under **Input**, update the following values:
   - Highlight and delete `Comment`, then type `Score`.
   - Highlight and delete `"Insert your JSON here"`, then type `30`.
3. Click **Start execution**.
4. Under **Graph view**, click on the **UnlockPremiumLessons** button.
5. To the right of **Graph view**, ensure the **Input and output** provides the correct output (i.e., the state machine triggers the Lambda function that returns the premium lessons).
6. In the upper right corner, click **New execution**.
7. In the **Start execution** pop-up menu, replace `30` with `80`.

8. Click **Start execution**.
9. Under **Graph view**, click on the **UnlockPremiumCourses** button.
10. To the right of **Graph view**, ensure the **Input and output** provides the correct output (i.e., the state machine triggers the Lambda function that returns the premium courses).

Conclusion

Congratulations — you've completed this hands-on lab!