### **Task 1: Configuring the Development Environment**

### **Access AWS Cloud9**

### **From the Services menu, search for and select Cloud9.**

### **Choose Open for the existing Cloud9 instance.**

### **Download and Extract Files**

### **Run the following command in the Cloud9 terminal to download the required files: wgethttps://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/05-lab-lambda/code.zip -P /home/ec2-user/environment**

### **Extract the files: unzip code.zip**

### **Run the Setup Script**

### **Run the setup script to recreate the lab resources: chmod +x ./resources/setup.sh && ./resources/setup.sh**

### **When prompted for an IP address, use your IPv4 address.**

### **Verify SDK Installation**

### **Verify Python SDK (boto3) installation: pip show boto3**

### **Confirm Resources**

### **S3 Bucket: Go to the S3 console, confirm the bucket, and check that the website displays.**

### **DynamoDB: Go to the DynamoDB console, confirm the table contains menu data and an index named special\_GSI.**

### **API Gateway: Go to the API Gateway console, confirm the ProductsApi API has the expected methods.**

### **Update Website Config**

### **Edit resources/website/config.js to replace null with the copied Invoke URL from API Gateway.**

### **Save the file.**

### **Update and Run the Configuration Script**

### **In python\_3/update\_config.py, replace <FMI\_1> with the S3 bucket name.**

### **Run the script: cd ~/environment/python\_3**

### **python update\_config.py**

### 

### **Task 2: Creating a Lambda Function to Retrieve Data from DynamoDB**

### **Edit the Lambda Function Code**

### **Open python\_3/get\_all\_products\_code.py.**

### **Replace <FMI\_1> and <FMI\_2> with the appropriate values from DynamoDB.**

### **Test Code Locally**

### **Run the following command in the Cloud9 terminal: cd ~/environment/python\_3**

### **python3 get\_all\_products\_code.py**

### **The output should show DynamoDB data in JSON format.**

### **Modify the Code for Testing**

### **Temporarily modify the condition: if offer\_path\_str is None:**

### **Run the code again and observe that fewer items are returned.**

### **Reverse the change to restore the original functionality.**

### **Prepare Lambda Code**

### **Open python\_3/get\_all\_products\_wrapper.py and update it with the IAM Role ARN.**

### **Save and package the Lambda code: zip get\_all\_products\_code.zip get\_all\_products\_code.py**

### **Upload Code to S3**

### **Upload the .zip file to your S3 bucket: aws s3 cp get\_all\_products\_code.zip s3://<bucket-name>**

### **Create the Lambda Function**

### **Run the wrapper code to create the Lambda function: python3 get\_all\_products\_wrapper.py**

### **Test the Lambda Function.**

### 

### **Task 3: Configuring the REST API to Invoke the Lambda Function**

### **Test the Existing API**

### **In API Gateway, test the /products GET method and confirm it returns mock data.**

### **Integrate Lambda with the API**

### **Replace the mock endpoint with the Lambda function in the GET /products method.**

### **Save the changes and test the API again.**

### **Enable CORS for /products**

### **Re-enable CORS on the /products GET method by following the instructions to add the Access-Control-Allow-Origin header.**

### **Update /on\_offer Method**

### **Configure the /on\_offer GET method to call the same Lambda function.**

### **Enable CORS for this method as well.**

### **Configure Mapping Templates**

### **For /on\_offer, add a mapping template to pass the path variable to the Lambda function: {**

### **"path": "$context.resourcePath"**

### **}**

### 

### **Deploy the API**

### **In the Resources panel, select the root /.**

### **Choose Deploy API, select prod, and click Deploy.**

### 

### **Task 4: Creating a Lambda Function for Report Requests**

### **Open the Lambda Code**

### **In AWS Cloud9, browse to python\_3/create\_report\_code.py.**

### **Run the Lambda Code Locally**

### **Run the following command in the terminal: python3 create\_report\_code.py**

### **Comment Out the Last Line of Code**

### **Comment out the last line: # print(lambda\_handler(None, None))**

### **Save the changes.**

### **Edit the Wrapper Code**

### **Open python\_3/create\_report\_wrapper.py and replace <FMI\_1> with the LambdaAccessToDynamoDB Role ARN value.**

### **Save the changes.**

### **Package and Upload the Lambda Code**

### **Zip the code: zip create\_report\_code.zip create\_report\_code.py**

### **Upload the zip file to your S3 bucket: aws s3 cp create\_report\_code.zip s3://<bucket-name>**

### **Create the Lambda Function**

### **Run the following command to create the Lambda function: python3 create\_report\_wrapper.py**

### **Test the Lambda Function**

### **In the Lambda console, select the create\_report function.**

### **Open the create\_report\_code.py file.**

### **Choose Test, name the event ReportTest, and save it.**

### 

### **Task 5: Configuring the REST API to Invoke the Lambda Function for Reports**

### **Test the Existing POST Method**

### **In API Gateway, go to the ProductsApi API and select the POST method for /create\_report.**

### **Replace the Mock Endpoint with Lambda**

### **With the POST method selected, choose Integration Request and click Edit:**

### **Integration type: Lambda Function**

### **Lambda Region: us-east-1**

### **Lambda Function: create\_report**

### **Click Save.**

### **Test the POST Method**

### **Deploy the API**

### **In the Resources panel, select the root /.**

### **Choose Deploy API, select prod, and click Deploy.**

### 

### **Task 6: Testing the Integration Using the Café Website**

### **Load the Café Website**

### **Return to the browser tab with the café website, and refresh the page.**

### **Alternatively, go to the S3 console, select the bucket, and copy the Object URL for index.html.**

### **Test the Menu Filter**

### **Scroll to the Browse Pastries section, click view all, and verify that more menu items appear.**

### **Update Menu Item Price in DynamoDB**

### **In the DynamoDB console, locate a menu item, and change its price\_in\_cents value.**

### **Save the change and reload the café website to verify that the price update is reflected.**

### 

### **Completion**

### **Congratulations! You’ve successfully created the Lambda function, updated the REST API, and tested the integration through the café website.**

### 