AWS　Lambda演習(関数の作成と実行)1

クラス　　　　No. 氏名

NoSQLのデータベースであるDynamoDBを利用してLambda関数を実行する。

□DynamoDBの作成

1.テーブルの作成を行いなさい。(DynamoDB⇒テーブル)

テーブル名：クラス名番号-db　パーティションキー：pid(数値)

テーブルは作成できたか。［　作成できた　・　作成できない　］

2.次のように項目を追加しなさい。

項目の表示⇒項目の作成で次のように項目を作成しなさい。

・パーティションキー：pid 値：1001 タイプ：数値

・pname　値：ブッシュド・ノエル 　タイプ：文字列　　・price　値：200　　タイプ：数値

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール

自動的に生成された説明

　項目の作成はできたか確認しなさい。［　確認できた　・　確認できない　］

3.2と同じように次の内容の項目を作成しなさい。

・パーティションキー：pid 値：1002 タイプ：数値

・pname　値：シブースド・ノエル 　タイプ：文字列　　・price　値：250　　タイプ：数値

・パーティションキー：pid 値：1003 タイプ：数値

・pname　値：イチゴとシブースドケーキ 　タイプ：文字列　　・price　値：400　　タイプ：数値

4.「項目の検索」を選び、3項目が追加されているか確認しなさい。[　確認できた　・　確認できない　]

□関数の作成

1.関数を作成しなさい。詳細は以下のとおり。

　一から作成　関数名：クラス名番号-cake-function　ランタイム：Python3.9

　実行ロール：既存のロールを使用する(robomarker\_students**LabRole**)

　＊作成する前に「IAM コンソールで robomaker\_students ロールを表示します。」をクリックする。

　　「ポリシーをアタッチ」をクリックして、「AmazonDynamoDBFullAccess」を選ぶ。

＊ 「認証を許可」を選択するとポリシーを選ぶことができる。

関数は作成できたか。［　作成できた　・　作成できない　］

2.コードの入力

lambda\_functionの内容をすべて削除して、次のコードをコピーしなさい。(lambda-func01.txt)。

|  |
| --- |
| import boto3  from boto3.dynamodb.conditions import Key, Attr    # DynamoDBと接続  dynamodb = boto3.resource('dynamodb')    # テーブル内のIDを指定してデータを該当するitemを取得する  def lambda\_handler(event, context):  table\_name = "クラス名番号-db"  partition\_key = {"pid": event["pid"]}  dynamotable = dynamodb.Table(table\_name)  res = dynamotable.get\_item(Key=partition\_key)  item = res["Item"]    return item |

3.作成した関数をデプロイしなさい。［　デプロイできた　・　デプロイできない　］

4.作成した関数を実行する。

　今回はトリガーなしで関数を実行する。まずは「Test」をクリックしてテストイベントの設定を行いなさい。

　イベント名：クラス名番号event１

　「テストイベント」に次のコードを入れる(いらないものを削除して修正を行う)。

|  |
| --- |
| {  "pid": 1001  } |

　テストイベントは作成できたか。［　作成できた　・　作成できない　］

5.関数を実行しなさい。

「Execution result」が表示されるので「Function Logs」を確認しなさい。

「Error」がないか確認しなさい。［　確認できた　・　確認できない　］

6.次の内容が表示されるか確認しなさい。

|  |
| --- |
| Response  {  "pid": 1001,  "pname": "ブッシュド・ノエル",  "price": 250  } |

［　確認できた　・　確認できない　］

７.関数のコードをテキストファイルにコピーし、クラス名番号-lambda-function1.txtで保存してsv23に提出しなさ

い。

AWS　Lambda演習(関数の作成と実行)2

クラス　　　　No. 氏名

今回はLambdaの関数を作成し、実行する。

■作成する関数

　指定されたS3のバケットにファイルを保存する。保存ファイル名は「クラス名番号\_現在の日付時刻.txt」。

　<例>ie3a00\_2022-10-28-12-24-18.txt

□S3バケットの作成

1.ファイルを保存するS3バケットを作成しなさい。バケット名はクラス名番号-bg、リージョンは変更なし。

　［　作成できた　・　作成できない　］

2.作成したバケットにファイルがないことを確認しなさい。

　［　ファイルがない　・　ファイルがある　］＊もしあれば削除しなさい。

□Lambdaの設定(関数)

1.次の仕様で関数を作成しなさい。

　一から作成　関数名：クラス名番号-s3-function　ランタイム：Python3.9

　実行ロール：既存のロールを使用する(robomarker\_students**LabRole**)

関数は作成できたか。［　作成できた　・　作成できない　］

2.コードの入力

lambda\_functionの内容をすべて削除して、次のコードをコピーしなさい(lambda-func02.txt)。

|  |
| --- |
| #Pythonが使用できるようにする  import boto3  #時間モジュール  from datetime import datetime  #print('Loading function')  #S3オブジェクトの生成  s3 = boto3.resource('s3')  #Lambdaのメイン関数  def lambda\_handler(event, context):  #バケット名の指定  bucket = '作成したバケット名'  #ファイル名設定  key = 'クラス名番号\_' + datetime.now().strftime('%Y-%m-%d-%H-%M-%S') + '.txt'  #ファイルの内容  file\_contents = 'Hello AWS　Lambda!!'  #バケット名とパスを指定  obj = s3.Object(bucket,key)  #ファイルをバケットの格納  obj.put( Body=file\_contents )  return |

3.作成した関数をデプロイしなさい。［　デプロイできた　・　デプロイできない　］

4.作成した関数を実行しなさい。

　今回はトリガーなしで関数を実行しなさい。まずは「Test」をクリックしてテストイベントの設定を行いなさい。

　イベント名：クラス名番号event2

　テストイベントは作成できたか。［　作成できた　・　作成できない　］

5.関数を実行しなさい。

「Execution result」が表示されるので「Function Logs」を確認しなさい。

「Error」がないか確認しなさい。［　確認できた　・　確認できない　］

6.ファイルの確認

　作成したS3バケット内を確認し、ファイルが作成されているか確認しなさい。

［　確認できた　・　確認できない　］　作成されたファイル名：［ ］

7.6で作成されたファイルをダウンロードしてファイルの内容を確認しなさい。

　ファイルの内容：［　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　］

８.関数のコードをテキストファイルにコピーし、クラス名番号-lambda-function2.txtで保存してsv23に提出しなさ

い。