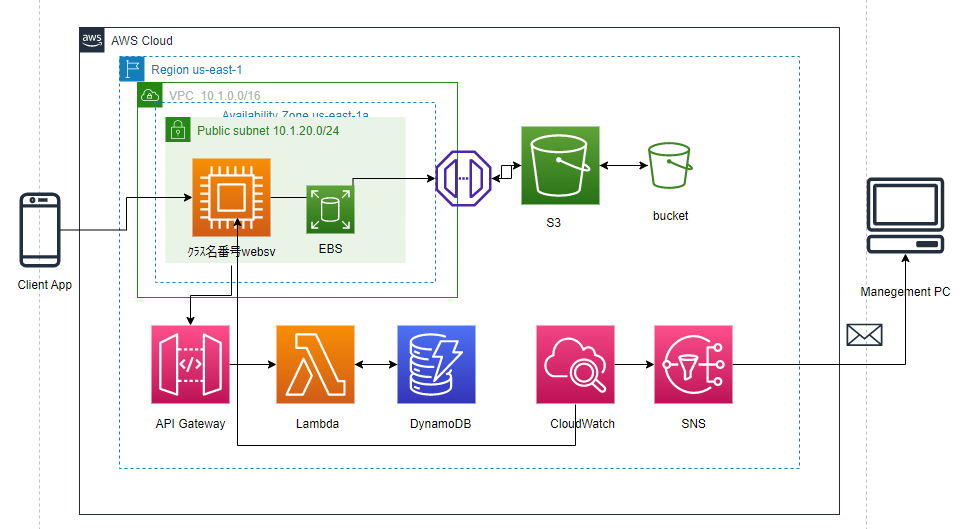
設計パターン(EC2/Dynamo)演習2-1

クラス名　　　　No. 氏名

今回の演習では、Lambda/APIGateway、DynamoDBを利用したWebシステムを構築します。EC2上のWebサーバからAPIを利用してDynamoDBの内容を表示します。

＊設定情報はWebサイト設定シートファイルに入力していくこと。



■キーペアの作成

1.キーペアがない場合、クラス名-key.pemで作成しておくこと。

■ネットワークの構築

VPC、サブネットなどネットワークの構築を行いなさい。

1.VPCを作成しなさい。

　VPC名：クラス名番号\_vpc CIDR: 10.1.0.0/16

2.次のサブネットを作成しなさい(VPC：クラス名番号\_vpc)。

①パブリックサブネット用・・・サブネット名：クラス名番号\_pub　AZ：us-east-1a　CIDR：10.10.20.0/24

3.インターネットゲートウェイを作成しなさい。

　名前：クラス名\_igw　アタッチ：クラス名番号\_vpc

4.次のルートテーブルを作成しなさい。＊VPCはクラス名番号\_vpc

　①クラス名番号\_pub用・・・ルートテーブル名：クラス名番号\_rt

5.次のようにサブネットのルートテーブルの関連付けを行いなさい。

　①クラス名番号\_rt・・・クラス名番号\_pub

6.次のセキュリティグループを作成しなさい。＊設定はすべて「インバウンドルール」で行うこと。

　①EC2用(クラス名番号\_websv)

グループ名：クラス名番号\_websv\_sg　説明：クラス名番号\_websv\_sg 　VPC：クラス名番号\_vpc

タイプ：SSH ソース：、タイプ：HTTP ソース：0.0.0.0)

7.ルートテーブル(クラス名番号\_pub\_route)にルートを追加しなさい。

　送信先：0.0.0.0 ターゲット：インターネットゲートウェイ(クラス名\_igw)

■EC２の設定(Webサーバ)

1.次の仕様でインスタンスを起動しなさい。

・キーペア　・・・　クラス名-key.pem

・VPC：クラス名番号\_vpc　サブネット： 名前：クラス名番号\_websv

パブリックIPの自動割り当て：有効化　 セキュリティグループ：クラス名番号\_websv\_sg

■接続確認/設定

1.Tera termでWebサーバ(クラス名番号\_websv)に接続しなさい。＊パブリックアドレスでアクセスする

2.Webサーバ(Apache)とPHPをインストールしなさい。

|  |
| --- |
| sudo yum install -y httpd  sudo amazon-linux-extras install -y php8.0  sudo systemctl start httpd  sudo systemctl enable httpd |

3.次の内容のPHPファイルを作成して、アップロードしてPHPの動作確認をしなさい。

|  |
| --- |
| <?php  phpinfo();  ?> |

■DynamoDBの設定

1.テーブルの作成(DynamoDB＞テーブル)

　次の仕様に従いテーブルを作成しなさい。

　テーブル名：クラス名番号-cake-table　パーティションキー：pid(数値)

タグ：キー(Name)、値(クラス名番号-db)

＊状態が「Active」になったらOK。

＊テーブルがすでになる場合は、内容の変更のみでもOK。

2.データの追加

　「項目を表示」を選び、「項目の作成」からデータを追加しなさい。

＊項目のタイプ・・・pname(文字列)、category(文字列)、price(数値)

|  |  |  |  |
| --- | --- | --- | --- |
| pid | pname | category | price |
| 1 | ブッシュド・ノエル | ショートケーキ | 250 |
| 2 | シブースド・ノエル | ショートケーキ | 200 |
| 3 | イチゴとシブースドケーキ | ショートケーキ | 2000 |

設計パターン(Lambda/APIGateway)演習2-2

クラス名　　　　No. 氏名

■Lambda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：-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(LaboRole)

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。

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  } |

5.関数を実行しなさい。

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  } |

■トリガーの作成(Lambda>関数)

1.Lambda関数(クラス名番号-cake-function)を選び、「トリガーを追加」をクリックする。ソースの選択から「API

Gateway」を選択しなさい。

2.以下の使用でAPIを作成しなさい(Create a new APIを選ぶ)。

　　APIタイプ：REST API セキュリティ：IAM　　API名：cake-function-API

3.「トリガー」に追加された確認しなさい。

■API Gatewayの作成(API Gatewayで実施)

作成したAPI Gatewayの設定を行いなさい。

1.トリガーにある作成したAPI GatewayをクリックしてAPI Gatewayに移動。メソッドの「ANY」を選択して、「アクション」から「メソッドの削除」しなさい。

2.メソッドの追加を行いなさい。「アクション」の「メソッドの作成」から「GETメソッド」を作成しなさい。

3.2で作成したGETメソッドに「Lambda関数」を設定しなさい。設定する関数は「cake-function」。設定後に「保

存」を行うこと。

4.API Gatewayのテスト次の仕様に従い行いなさい。

　①「統合リクエスト」を選択して、次の設定をおこないなさい。ここでは、実際に送信するデータなどの指定を行う。

　　　・マッピングテンプレートの追加をクリック

　　　　Content-Type：application/json ＊設定後「作成」をクリックする。

　②テンプレート欄に次のコードを入力しなさい。入力後に「保存」しなさい。

|  |
| --- |
| {  "pid" : $input.params('pid')  } |

　③「メソッドの実行」のリンクをクリックして、メソッドの実行画面に戻る。

　④「テスト」ボタンをクリックして次の設定を行いなさい。

　　　・クエリ文字列・・・pid=1001

入力後に「テスト」ボタンをクリックして、テストしなさい。

　⑤レスポンス本文に次のような表示ができれば、テストは成功。

|  |
| --- |
| "pid": 1001, "pname": "\u30d6\u30c3\u30b7\u30e5  \u30c9\u30fb\u30ce\u30a8\u30eb", "price": 200}　＊修正 |

　⑥「メソッドの実行」のリンクで戻り、「アクション」の「APIのデプロイ」を選ぶ。「デプロイされるステージ」を「default」に設

定してデプロイしなさい。

設計パターン(Lambda/APIGateway/S3)演習2-3

クラス名　　　　No. 氏名

■APIGatewayの確認(ブラウザで確認する)

1.Lambdaに戻り、「トリガー」の「API Gateway」を選び、「API endpoint」のURLをコピーする。

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

自動的に生成された説明

2.ブラウザで1でコピーしたURLを貼り付け、?pid=1001を追加して実行しなさい。

　　＊https://**～?pid=1001**

3.次のような表示がされた確認しなさい。[　確認できた　・　確認できない　]



■S3のバケット作成

１.次の仕様に従いバケットを作成しなさい。

　バケット名：クラス名番号-bucket リージョン：バージニア北部(us-east-1) ACL有効

バケットのバージョニング：有効にする　パブリックアクセスをすべてブロック：チェックを外す

2.1で作成したバケットにフォルダ「images」を作成し、cake01.jpg〜cake03.jpgをアップロードしなさい。

3.アップロードした3つのファイルとimagesフォルダを「ACLを使用して公開」を選び公開するように設定すること。

4.設定シートのS3の箇所に情報を記入しなさい。

5.作成したバケット内の「images」フォルダを選び、「URLをコピー」して一旦テキストファイルにペーストしておく。

　＊ここでコピーしたものは、画像表示の時にsrcに指定する。

■VPCエンドポイントの作成

プライベートサブネット内のEC2からS3のバケットにアクセスするためのエンドポイントを作成しなさい。

1.次の仕様に従いVPCエンドポイントを作成しなさい。

　名前タグ：クラス名番号-s3-endpoint サービスカテゴリ：AWSのサービス

　サービス名：com.amazonaws-us-east-1.s3(タイプ：Gateway) VPC：クラス名番号\_vpc

ルートテーブル：クラス名番号\_rt ポリシー：フルアクセス

■PHPファイルの変更

1.画像ファイルが表示されるようにPHPファイル(part-db2.php)を修正しなさい。

|  |
| --- |
| <img src=エンドエンドポイント(コピーしたURL)/images/%s> |

　　　↓ コピーしたURLを設定

<設定例>

|  |
| --- |
| <img src=https://ie3a000-bucket.s3.amazonaws.com/images/%s> |

■PHPで作成したAPIからデータを取得して、実行する。

　＊ここでは以前に作成したEC2の「クラス名番号\_websv」にPHPファイルをアップロードして確認すること。

1.PHPファイル(part-db2.php)を修正しなさい。「API GatewayのURL (endopoint)」の部分を自分のAPI

GatewayのURLに変更しなさい。

|  |
| --- |
| $url = 'API GatewayのURL(endopoint)?pid=' . $number; |

2.変更後、Tera TermでEC2にアクセスして、part-db2.phpをアップロードしなさい。

3.アップロード後、ファイルを「/var/www/html」へ移動しなさい。

4.ブラウザで次のURLにアクセスして次のような表示がされるか確認しなさい。

http://IPアドレス/part-db2.php

グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明

＊商品番号を選択し、「選択」ボタンを

　　　クリックした結果

5.作成したPHPファイルを、クラス名番号-part-db2.phpで保存してsv23に提出しなさい。