Assignment 6

ICE191 Software Architecture / Cloud Computing

1. **Modify your Lambda function to include the field City for each new record in the Students DynamoDB table. 30 points.**
    - Firstly we need to define what it's a lambda function, a lambda function helps to run any code virtually without the computation that it needs and it's a serverless tool, important that need to be mentioned that you don't need to provide or manage those servers.
    - In this case it's more easy to use a lambda function,but firstly we need to modify our lambda adding this function to add the field city.
    - So, basically we need to add these two functions in our lambda handler, well that's what I did, for that we modify this.
    -

```
if operation == "PUT":
    try:
        body = event.get("body")
        body = json.loads(body)
        try:
            if "city" not in body.keys():
                item = {"id": body["id"], "full_name": body["full_name"],
                        "personal_website": body["personal_website"],
                        "city": None}
            else:
                item = {"id": body["id"], "full_name": body["full_name"],
                        "personal_website": body["personal_website"],
                        "city": body["city"]}

            dynamo_resp = create_operation(item=item)
            return get_response(200, json.dumps(dynamo_resp))

        except KeyError:
            return get_response(400, json.dumps("Invalid item format"))

    except ClientError:
        return json.dumps("CREATE not successful")
```

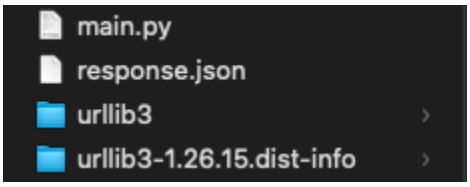
    - Inside in this operation we are creating a new item to add to our table, that going to check if our attribute city is in the attributes, and depending how it is, adding None or just select the type of value that comes from our body, besides that, we are just confirming the response and have dumps to know if everything is working fine.
- Adding to this function out of lambda_handler we need to add this function that we use.

```
def create_operation(item:dict):
    students_table.put_item(Item=item)
    key = item.get("id")
    key = {"id": key}
    response = students_table.get_item(Key=key)
    return response
```

-
- And finally just confirm if it's running how it should be, just printing our new item in my case was pruebita now with the city included. Basically now we need to think and keep in mind this need attribute in our next operations.

```
ystijuana.com'}
{'city': 'San Diego', 'full_name': 'Daniel Velazquez', 'id': '29896', 'personal_website': 'velazqu
ez.cetystijuana.com'}
{'full_name': 'Pruebat', 'id': '317503221', 'personal_website': 'prueabss.cetystijuana.com'}
{'city': 'Navojoa', 'full_name': 'pruebita', 'id': '55555', 'personal_website': 'pruebita.com'}
{'full_name': 'Gatubela', 'id': '111', 'personal_website': 'GatCat.com'}
{'city': 'Nogales,MX', 'full_name': 'Hasta', 'id': '322', 'personal_website': 'Pront.com'}
{'full_name': 'aerws', 'id': '123554', 'personal_website': 'asdf.com'}
{'full_name': 'ams', 'id': '12354', 'personal_website': 'asdf.com'}
{'city': 'Chula Vista', 'full_name': 'John Towers', 'id': '42069', 'personal_website': 'johntowers
.com'}
{'city': None, 'full_name': 'AlbertoTST2mod', 'id': '18821v3', 'personal_website': '12345test.com'
}
```

2. **Modify Read in your Lambda function to return the weather of the city assigned to the Students DynamoDB table record. 30 points.**
   Okay moving on, to create all the things we need to deploy our API.
    - We need to create a folder, where we are going to save everything
    - In this folder we are going to install the library that we need for our API, for that we need to use the next command.
        - pip3 install --target . urllib3

    - This command will install the library in the folder we are in the terminal, so you need to make sure that you are in the correct folder that we already create
- inside this folder we need to add our lambda function where we have our lamba handler in need to be compress in a zip, for that we can use simply right click and compress file. And i going to look like something like this.
    - 

- Once we have our folder ready now we can create our lambda our just update it, with the next command
    - aws lambda update-function-code --function-name weather_lambda_roldan --zip-file fileb://weahter_lamda_roldan.zip
- Now we already have everything until this, its time to move to the api gateway
- To create our api gateway we need to use the next command
    - aws apigateway create-rest-api --name roldan_weather_api
- We are going to receive an output like this, all the next output is important to save them


```
{
    "id": "6z1kxi2kg3",
    "name": "roldan_weather_api",
    "createdDate": 1679772644,
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
    "types": [
    "EDGE"
    ]
    },
    "disableExecuteApiEndpoint": false
}
```

- Now we need to understand how we are going to create resources with the next path like this /students and the /{id}, so we need to create the students resources and then id.
- To get to that we need to create the parent, for that we use this command
    - aws apigateway get-resources --rest-api-id 6z1kxi2kg3
    - This command help us to start creating our resources that we need.


```
{
    "items": [
    {
    "id": "si89mvnehd",
    "path": "/"
    }
    ]
}
```


    - Important to keep the id value.
- As I said we need to create students, for that we use this command
    - aws apigateway create-resource \
      --rest-api-id 6z1kxi2kg3 \
      --parent-id si89mvnehd \
      --path-part students
    - And we are going to have a output like this


```
{
    "id": "fuyxgn",
    "parentId": "si89mvnehd",
    "pathPart": "students",
    "path": "/students"
}
```


- Once we have students, we need to create id
    - aws apigateway create-resource \
      --rest-api-id 6z1kxi2kg3 \
      --parent-id fuyxgn --path-part {id}
    - Output

{
"id": "b39s7s",
"parentId": "fuyxgn",
"pathPart": "{id}",
"path": "/students/{id}"
}

- Now, that we have all the resources correctly, its time to add a method to it, for that we use this command
    - aws apigateway put-method \
      --rest-api 6z1kxi2kg3 \
      --resource-id b39s7s \
      --http-method GET --authorization-type NONE
    - We should get a output like this

{
"httpMethod": "GET",
"authorizationType": "NONE",
"apiKeyRequired": false
}

- If everything its correct now we need to put integration, its like combine this two thing, before deploy,

```
aws apigateway put-integration \
--rest-api-id 6z1kxi2kg3 \
--resource-id b39s7s \
--http-method GET \
--integration-http-method POST \
--type AWS_PROXY \
--uri
arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions
/arn:aws:lambda:us-east-1:292274580527:function:weather_lam
bda_roldan/invocations
```

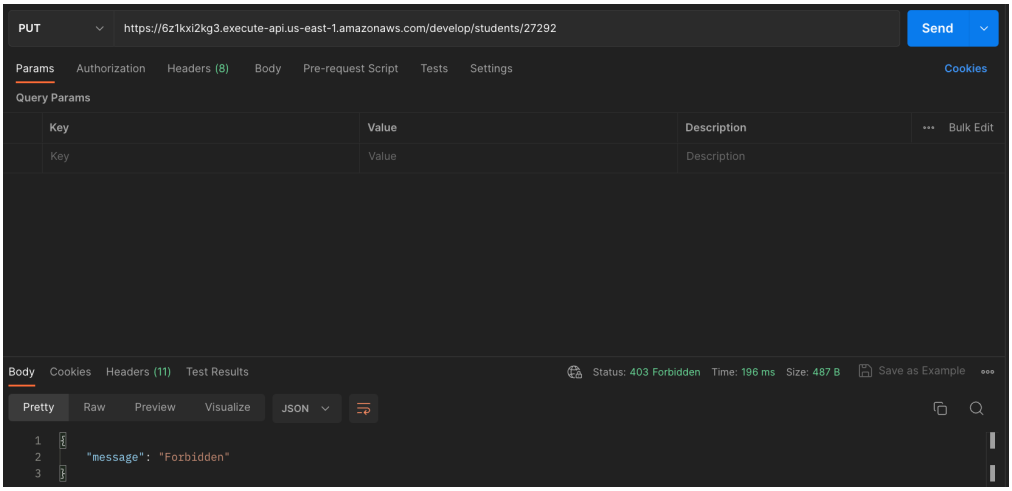
- Output that we are goig to receive

{
"type": "AWS_PROXY",

"httpMethod": "POST",
"uri":
"arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:292274580527:function:weather_lambda_roldan/invocations",
"passthroughBehavior": "WHEN_NO_MATCH",
"timeoutInMillis": 29000,
"cacheNamespace": "b39s7s",
"cacheKeyParameters": []
}

- Now, it's time to deploy our api to try it, for that we use the next command
    - aws apigateway create-deployment \
      --rest-api-id 6z1kxi2kg3 \
      --stage-name dev \
      --description "Deploying 32th api"
    - We need to receive an output like this

{
"id": "f6scoq",
"description": "Deploying 32th api",
"createdDate": 1679774804
}

- Now, we can you test it our API, for example with postman
    - 

3. **Add authorization to your API Gateway API. Only valid user is admin and password abc123!@#. 30 points.**
   I couldn't do it.

4. **Read the Test Driven Development is the best thing that has happened to software design article and write a summary and opinions about it. 10 points.**

- How he sees or it is the TDD, how a living organism evolves and adapts itself, because it is constantly changing and adaptive to every environment.
- If was interesting to understand different methods in the industry to try your code, or trying to test the best you could.
- Important to emphasize that you need to write your test before you start coding, well, that's what he suggests.
- It was interesting to see different implementations of codes and understand it, even though they are small but with a lot of information.
- For me the mocking assimilates to what is a programming principle, of this kind of overwrites enormous.
- And why it's important to try to predict the future even if you want to start coding, like most of us.

A summary of this article, it's about -Test-Driven-Deployment even it sounds something related with cars it doesn't, and he suggest us 4 ideas of how to implement in our projects or jobs, the articles its complement with a part of code and another of the explanation of it, very important to mention that all this is to try to reduce complexity in our test setups. Because most of the time they are complex, and the article end with why to use basically because this case is going to give us the feedback so now, we can start moving forward with big steps. Another thing that mention is why isn't correct to try of predict futures scenarios, well this happen because most of the cases we are not good trying to predict the future and with all the time involved with this, the only thing that happen it's to get in a cycle of overthinking and you can just get to the conclusion that basically you can predict it, most of the cases.