Assignment 5

ICE191 Software Architecture / Cloud Coxmputing

1. Write a Lamba function to CRUD over the Students DynamoDB table. 30 points.

First able we need to understand a few things here.

- **CRUD:**
  - Meaning
    - It's an acronym of Create, Read, Update and Delete, we use it because are direct access to recourses on the system. What happened is that our REST API maps or linking CRUD operations to the ones from HTTP methods.
  - Why we want to use them.
    - Well, this set of basic operations can be worked on most database systems and is commonly used in the field.
  - Now I'm going to explain the meaning of each acronym.
    - Create -> refers to the operation to inserting new data from a database.
    - Read-> means to the operation of getting data form database.
    - Update -> This means to the operation of modifying existing data in a database.
    - Delete -> As logical as it sounds its removing data from a database.
- **AWS Lambda**
  - Meaning
    - It's a serverless tool, where you can run your code virtually without the computation that it needs, it works with any type of application or back-end service, and it's important to mention that you don't need to provide or manage those servers.
  - Pros
    - You don't need to manage any server. This means that you don't need to care about the infrastructure, so you only need to focus on your code.

        - It's automatically scale up or down to match the incoming request traffic.
        - You can join or combine this tool with the other ones from AWS services like AWS S3, DYNAMO DB, API GATEWAY, etc.
    - Cons
        - You imagine how hard it's to debug something that is serverless, because we are used to work with server-based architecture, and that happen because you can't see what is behind the infrastructure.
        - As we see previously, when we use previously a cold start it's the process when you use or invoked a new function so it's going to be delay for all the resources that need to be provided.
        - You have limited execution time and memory, you have limited time and memory to use, so this makes kind at complicated to run our resources-intensive applications.
- Now we need to create our CRUD functions in our py file, so for that we going to use Boto3, and then those function we going to adapt them to our hander function, but I will explain that later.
    - **Read Function**
        -
```
def read_operation(key:dict):
        response = table.get_item(Key=key)
        return response
```
        - With this function only allows a dict named key
        - response = table.get_item(Key=key) -> in response we are going to ask to get the item from table and confirm the there is a key which in our case is our 'id', we just return the response.
    - **Create Function**
        - It looks something like this.
        -
```
def create_operation(item: dict):
    response = table.put_item(Item=item)
    key = item.get("id")
    key = {"id": key}
    return response
```

- As you can see, we call our function create_operation, that takes a single ‘item’ which must be a dictionary.
- In our variable response I’m going to put a new item ‘Item’ the entire Item in the dictionary, which need to validate first, before to add it.
- key = item.get("id") -> This line retrieves the “id” in our new variable key, that we are getting from our ‘item’ dict.
- key = {"id": key} -> This create a new dict with a single-value pair which it’s our key.
- And finally, our response to know if everything run correctly.

- **Delete Function**
  - ```
    def delete_operation(key: dict):
        response = table.delete_item(Key=key)
        return response
    ```
  - Like our read function again our function only accepts a dictionary declare like ‘key’ in this case we are going to use the method delete_item from our key and then we return our variable ‘response.’
- **Update Function**
  - ```
    def update_operation(key: dict, values: str, expression_attributes: dict):
        table.update_item(Key=key,
                          UpdateExpression=values,
                          ExpressionAttributeValues=expression_attributes)
        response = table.get_item(Key=key)
        return response
    ```
  - In this function is a little bit different the input going to be a key dictionary, values its where we going to indicate the attribute that we are going to replace, expression attributes the new attribute that going to substitute the old one.
  - UpdateExpression=values -> This UpdateExpression this property it’s from the operations declare in boto3, so basically you need to use it exactly like that, this function is where we going to indicate the attribute that we going to change and the variable that we are going to use, important to add “SET” and the beginning of this string, this is to update or change over our item that already had.

        - ExpressionAttributeValues=expression_attributes -> Same as the last point ExpressionAttributeValues come from boto3, where we need to declare our variable and add the new value for that.
- Once we had our 4 functions of CRUD, our destination now it's to create our lambda and then try it for each function.
- To create our lambda, we can use the next command.
    - `aws lambda create-function --function-name lambdaRoldanV10 --zip-file fileb:///Users/davidroldanmachado/Desktop/ICC/6/CLOUD_COMPUTER/imp_docs/compress_crud.py.zip --handler CRUD_FUNCT_ROLDAN.handler --runtime python3.9 --role arn:aws:iam:292274580527:role/lambda_ice191.`

    - create-function -> For create the lambda.
    - –function-name -> The name we want to give it, this name we are going to use it later.
    - –zip-file -> Were we need to indicate the path where our file its saved, important to and 'fileb://' and our file need to be compress in a zip file.
    - –handle -> Here we need to specify the name of our file then and "." + your function name, for example "CRUD_FUNCT_ROLDAN.hadler" my file its named CRUD_FUNCT_ROLDAN and my function inside my file it declared like handler.
    - –runtime -> The language that we going to use I going to use python 3.9
    - –role -> the ARN with the permission to create a lambda
- The output that you are going to receive will look, something like this,

```
{
    "FunctionName": "lambdaRoldanV9",
    "FunctionArn": "arn:aws:lambda:us-east-1:292274580527:function:lambdaRoldanV9",
    "Runtime": "python3.9",
    "Role": "arn:aws:iam::292274580527:role/lambda_ice191",
    "Handler": "CRUD_FUNCT_ROLDAN.handler",
    "CodeSize": 1832,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2023-03-06T06:07:01.563+0000",
    "CodeSha256": "/Z9rkfm6XNvBlThpsQ8CIPPdXIwNsK43ZM2PykVRzfM=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "6bc3265b-5ad4-4b3b-8b0a-8b8188f7f336",
    "State": "Pending",
    "StateReason": "The function is being created.",
    "StateReasonCode": "Creating",
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ],
    "EphemeralStorage": {
        "Size": 512
    },
    "SnapStart": {
        "ApplyOn": "None",
        "OptimizationStatus": "Off"
    },
    "RuntimeVersionConfig": {
        "RuntimeVersionArn": "arn:aws:lambda:us-east-1::runtime:07a48df201798d627f2b950f03bb227aab4a655a1d019c3296406f95937e2525"
    }
}
```


- Now we have our lambda ready we can move on to try our functions.
- For that I use the next command.
    - `aws lambda invoke --function-name lambdaRoldanV9 --payload file:///Users/davidroldanmachado/Desktop/ICC/6/CLOUD_COMPUTER/imp_docs/test_invoke.json output6.txt`
        - lambda -> We are using the benefits from lambda.
        - invoke -> Function to invoke or basically try our function.
        - --function-name -> Indicate your function name of your lambda
        - --payload -> The JSON that you want you want to provide to your Lambda function as input
        - Important to say that my output.txt it's to generate a file in my system instead the typical output.
- Okay, now we already understand how to invoke your lambda lets test it with our functions.
    - **Trying my Read**
        - This going to be the payload file for the create.

-
```
{
    "operation":"READ",
    "key":{"id":"27292"
    }
}
```
-
```
davidroldanmachado@Davids-MacBook-Air ~ % aws lambda invoke --function-name lambdaRoldanV6 --pa
yload file:///Users/davidroldanmachado/Desktop/ICC/6/CLOUD_COMPUTER/imp_docs/test_invoke.js out
put2.txt
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```

- **Trying my update**
  - This is our payload file to do the invoke function.
  -
```
{
    "operation":"UPDATE",
    "key":{"id":"55555"},
    "values": "SET full_name = :val1",
    "expression_attributes": {":val1": "Don Camote"}
}
```
  -
```
davidroldanmachado@Davids-MacBook-Air ~ % aws lambda invoke --function-name lambdaRoldanV7 --pa
yload file:///Users/davidroldanmachado/Desktop/ICC/6/CLOUD_COMPUTER/imp_docs/test_invoke.js out
put.txt
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```
  - If everything it correct this need to be your input
  - {"body": "{\"Item\": {\"full_name\": \"Don Camote\", \"id\": \"55555\", \"personal_website\": \"www.elmilanios.com\"}, \"ResponseMetadata\": {\"RequestId\": \"6P0VD603PSRK8JDSMM4SKTVNJFVV4KQNSO5AEMVJF66Q9ASUAAJG\", \"HTTPStatusCode\": 200, \"HTTPHeaders\": {\"server\": \"Server\", \"date\": \"Tue, 07 Mar 2023 22:27:53 GMT\", \"content-type\": \"application/x-amz-json-1.0\", \"content-length\": \"106\", \"connection\": \"keep-alive\", \"x-amzn-requestid\": \"6P0VD603PSRK8JDSMM4SKTVNJFVV4KQNSO5AEMVJF66Q9ASUAAJG\", \"x-amz-crc32\": \"3745846114\"}, \"RetryAttempts\": 0}}"}
- **delete function.**
  - This is our payload file to do the invoke function.

- ```
  {
      "operation":"DELETE",
      "key":{"id":"55555"}
  }
  ```
- The form to correctly invoke our lambda.
- ```
  davidroldanmachado@Davids-MacBook-Air ~ % aws lambda invoke --function-name lambdaRoldanV7 --pa
  yload file:///Users/davidroldanmachado/Desktop/ICC/6/CLOUD_COMPUTER/imp_docs/test_invoke.js out
  put1.txt
  {
      "StatusCode": 200,
      "ExecutedVersion": "$LATEST"
  }
  ```
- If runs correctly the input need to be something like this
- ```
  {"body": "{\"ResponseMetadata\": {\"RequestId\":
  \"TBTHRAAF04JC61S8FJIQ6K45GNVV4KQNSO5AEMVJF66Q9ASUAAJG
  \", \"HTTPStatusCode\": 200, \"HTTPHeaders\":
  {\"server\": \"Server\", \"date\": \"Tue, 07 Mar 2023
  22:31:27 GMT\", \"content-type\": \"application/x-amz-
  json-1.0\", \"content-length\": \"2\", \"connection\":
  \"keep-alive\", \"x-amzn-requestid\":
  \"TBTHRAAF04JC61S8FJIQ6K45GNVV4KQNSO5AEMVJF66Q9ASUAAJG
  \", \"x-amz-crc32\": \"2745614147\"},
  \"RetryAttempts\": 0}}"}
  ```

- **Create function.**
  - This going to be the payload file for the create.
  - ```
    {
        "operation":"CREATE",
        "item": {
            "id": "92922",
            "full_name": "Don Julio",
            "personal_website": "donjulio.com"
        }
    }
    ```
  - And our output
  - ```
    davidroldanmachado@Davids-MacBook-Air ~ % aws lambda invoke --function-name lambdaRoldanV9 --pa
    yload file:///Users/davidroldanmachado/Desktop/ICC/6/CLOUD_COMPUTER/imp_docs/test_invoke.json o
    utput6.txt
    {
        "StatusCode": 200,
        "ExecutedVersion": "$LATEST"
    }
    ```
  - The other output.txt, it's going to look like this.
    ```

        - {"body": "{\"ResponseMetadata\": {\"RequestId\": \"NJ1URATGP0T75C046Q2CE15N9VVV4KQNSO5AEMVJF66Q9ASUAAJG\", \"HTTPStatusCode\": 200, \"HTTPHeaders\": {\"server\": \"Server\", \"date\": \"Mon, 06 Mar 2023 17:58:21 GMT\", \"content-type\": \"application/x-amz-json-1.0\", \"content-length\": \"2\", \"connection\": \"keep-alive\", \"x-amzn-requestid\": \"NJ1URATGP0T75C046Q2CE15N9VVV4KQNSO5AEMVJF66Q9ASUAAJG\", \"x-amz-crc32\": \"2745614147\"}, \"RetryAttempts\": 0}}"}
- Another important thing, you can know, if in your output of your terminal doesn't appear nothing about Unhandled, you have error invoking your lambda or in your payload file, so you need to check it out.
    - 

- So that's it with your lambda its already created and test it, ready to move on.

2. Write an API Gateway API to CRUD over your Lambda function. 30 points.

- First able we need to create a new Lamba or use the one we already have.
- To check out lambdas that we have.
    - `aws lambda list-functions`
- A summary of the steps to create an API Gateway version 1, very important to know that are this one:
    - Create our API.
    - Get resources of our API
    - Create resources.
    - Put method.
    - Put integration.
    - Add permission.
    - Deploy our API.
- Create our api.
    - The command to create our API it's the next one.
        - `aws apigateway create-rest-api --name RoldanApiV2`
            - apigateway -> The subfunction from aws, we are using the version 1.

        - create-rest-api -> Command to create the rest api.
        - --name RoldanApiV2 -> The name that we are used for our api
    - The output it's going to look something like this.
    - `aws apigateway create-rest-api --name RoldanApiV2 { "id": "rgbqroeat3", "name": "RoldanApiV2", "createdDate": 1678088770, "apiKeySource": "HEADER", "endpointConfiguration": { "types": [ "EDGE" ] }, "disableExecuteApiEndpoint": false`

- Now it's time to Get resources of our API.
    - aws apigateway get-resources --rest-api-id rgbqroeat3
    - get-resources ->
    - --rest-api-id -> Here we need to specify the id of our apigateway that we receive of output.
    - The output example:
    - `% aws apigateway get-resources --rest-api-id rgbqroeat3 { "items": [ { "id": "3fgabk3r8i", "path": "/" } ] }`
- Create resource here It's like the form what to give to our project and management of our folder, if we call it something like that.

  ```
  % aws apigateway create-resource --rest-api-id rgbqroeat3 --
  parent-id 3fgabk3r8i --path-part students
  ```

    - Create-resource -> Command to create a resource of our api.
    - --rest-api-id -> The id of our rest-api-id, we need to use our id
    - --parent-id -> Here we need to specify our parent id for example if we want to create another folder inside students in the new create-resource you use the id from the resource of students
    - --path-part ->You add the path that you are going to use to add another file for example, you need to add even another file o attribute something like this "/students/get_method" or "/students/"{id}"
    - Output

      ```
      {
          "id": "xi0izx",
          "parentId": "3fgabk3r8i",
          "pathPart": "students",
          "path": "/students"
      ```

}

- If we have done with the resource and you are sure of the distribution, we want to put-method because is where we specify more information about the method that you are trying to create in this case 'GET', it's time to put-method, for that we use the next command.
    - aws apigateway put-method --rest-api-id rgbqroeat3 --resource-id xi0izx --http-method GET --authorization-typ "NONE"
        - rest-api-id -> Our original API id
        - resource-id ->As a explain before this resource-id, tell us where we this method it's going to be applied.
        - --http-method -> The http method that you want to use for example "GET", "PUT","DELETE", "PATCH."
        - `--autorization-typ ->` If you want to be public or private, for public access you use "NONE", for private use "CUSTOM".
    - Output

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

- Moving forward, we need integrate this method to everything for that we use the next command.
    - aws apigateway put-integration --rest-api-id rgbqroeat3 --resource-id xi0izx --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:lambdaRoldanV10/invocations
        - rest-api-id ->Again our API id
        - --resource-id -> Where it's going to be applied
        - http-method ->Which http method are you using.
        - --integration-http-method -> Where it's important to specify 'POST', this action it posted in the documentation, so every '--integration-http-method' needs to be 'POST.'
        - --type -> Can be AWS, MOCK, HTTP_PROXY, AWS_PROXY, in our case we are going to use ' AWS_PROXY' because it specifies the

type of integration you want to create between your API Gateway and our lambda

- --uri -> This is one of the most important steps because you need to specify everything about the url, but I'm going to explain it how to create it. *arn:aws:apigateway:us-east-1:lambda:path/2015-03-31* this part comes in your lambda then add */function/* and now *arn:aws:lambda:us-east-1:292274580527* our special arn role to create lambda, *:function:lambdaRoldanV10/invocations* ->:function: and your lambdafunction with */invocations.*

- Output

```
{
"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:lambdaRoldanV10/invocations",
"passthroughBehavior": "WHEN_NO_MATCH", "timeoutInMillis": 29000,
"cacheNamespace": "xi0izx", "cacheKeyParameters": []
}
```

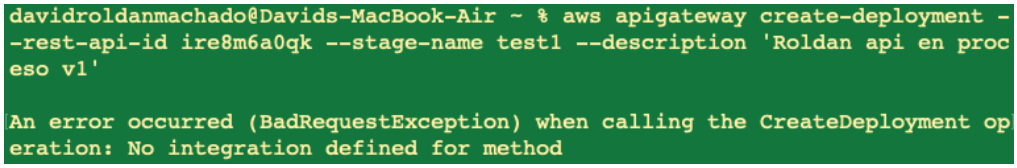
- Lambda Permissions
  - To moves to deploy our API, first able we need to give permission to our lambda that we have, for that we use the next.

    aws lambda add-permission --function-name lambdaRoldanV10 --source-arn "arn:aws:execute-api:us-east-1:292274580527:rgbqroeat3/students" --principal apigateway.amazonaws.com --statement-id roldan_get-1 --action lambda:InvokeFunction

    - --function-name -> Our lambda name in my case "lambdaRoldanV10".
    - --source-arn ->The source arn from our Lamba.
    - --principal -> Were we need to specify if we are going to use. SourceArn or SourceAccount.
    - --statement-id -> This is a relative name that you want to add.
    - **--**action -> There is two options lambdas:InvokeFunction or lambda:InvokeGetFunction, we are going to use InvokeFunction.
  - Output

```
{
    "Statement": "{\"Sid\":\"api-
roldan\",\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"apigateway.
```

```
amazonaws.com\"},\"Action\":\"lambda:InvokeFunction\",\"Resource\":\"a
rn:aws:lambda:us-east-1:292274580527:function:lambdaRoldanV11\"}"
}
```

- If you create your permissions correctly, it's time to do you deploy.
  ```
  aws apigateway create-deployment --rest-api-id ire8m6a0qk --
  stage-name test1 --description 'Roldan api en proceso v1.'
  ```
    - Create-deployment -> The function that we want to create in our case create-deployment.
    - –rest-api-id -> Once again or API id.
    - --stage-name -> The name we want to give it.
    - –description -> A message that you want to show.
- Output
    - Unfortunately, this time there is no successful output.
        - I get stuck with this error even I create everything from 0 a mean the api of check every aspect and attribute that was typed like 4 times and ask for help with 3 of my colleagues, but I couldn't so I only can say I'm sorry.
        - 
          ```
          davidroldanmachado@Davids-MacBook-Air ~ % aws apigateway create-deployment -
          -rest-api-id ire8m6a0qk --stage-name test1 --description 'Roldan api en proc
          eso v1'

          An error occurred (BadRequestException) when calling the CreateDeployment op
          eration: No integration defined for method
          ```

3. Implement 404 HTTP response when an invalid id is passed to the Read in your APIGateway API/Lambda. 30 points.

To create our 404 http response when an id it's not found, we can create something like this code. Which it's going to response. For that I create another function, in summary basically the check_in its going to search for all the items in the table 'Students' and return a Boolean and if for that the read function it's going to compare the input and check if continue or given a json.dups to alert from the error.

The function calls the check_id function to check if the extracted "id" exists in the table. If the "id" is found in the table, the function then uses the "key" dictionary to retrieve the corresponding item from the table using the table.get_item method and returns the item. If the "id" is not found in the table, the function returns a JSON with an error alert indicating that the "id" not found.

The check_id function takes in a string argument id_str and returns a boolean indicating whether the string exists as an "id" in the table object. The function first calls the table.scan method to scan through all items in the table. It then iterates through each item and checks if the "id" key exists in the item and if the value of the "id" key matches the id_str argument. If a matching "id" is found in the table, the function returns True or False

```
def read_operation(json_data):
    key = json.loads(json_data)
    search_id = key["key"]["id"]
    if check_id(search_id):
        response = table.get_item(Key=key)
        return response
    else:
        return json.dumps("StatusCode: Error 404 , ID not found")

def check_id(id_str):
    response = table.scan()
    for item in response['Items']:
        if 'id' in item and item['id'] == id_str:
            return True
    return False
```

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 is a living organism evolves and adapts itself, because its real its 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 a understand it, even 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 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 involve 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.