1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA UNIVERSIDADE DE COIMBRA FACULDADE DE CIÊNCIAS E TECNOLOGIA ***Departamento de Engenharia Informática***	**API Gateway Exercises** **Services Engineering** **Software Services Engineering** **2019/20 – 2nd Semester** MEI/MSE

Overview of API Gateway and Related Technologies

In this assignment, we will introduce Lambda and API Gateway and a couple of other AWS technologies.

Important NOTE: Some of the functionalities might not work properly under the restricted environment of AWS Educate. For example, students are restricted to use the North Virginia (us-east-1) region for the Lambda Functions.

To complete the exercises in this assignment, students are invited to read some of the excellent AWS documentation carefully, namely "Getting Started with REST APIs in Amazon API Gateway":

https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started.html

and "Use API Gateway Lambda Authorizers":

https://docs.aws.amazon.com/pt_br/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html

Building an API from an example is also relevant:

https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-create-api-from-example.html

This example creates an API specified in an Open API/Swagger document, and provides multiple resources accessible with different HTTP methods, whereas the latter explains how to connect a resource to a Lambda function.

Once you have implemented and deployed a service in the API Gateway, you are ready for interaction with the outside world. For this, in the left pane "Stages" you should now be able to see the URL to enter in the browser. For example (note that there is nothing to see in this link):

https://4pz82nt02bb.execute-api.eu-west-1.amazonaws.com/production

From the web page on the browser you can do additional GETs and you can see advice for using Postman, if you want to run a POST method. Curl in the command line is also an option, when you need to run a POST method and do not want to create a form for that purpose. For example, you could do as follows.

```
curl -d "{\"type\" : \"cat\", \"price\" : 123.11}" -X POST
https://4pz82nt02bb.execute-api.eu-west-
1.amazonaws.com/production/pets -H "Content-Type: application/json"
```

(although this will not actually add an extra animal to the list).

Exercises

1- Use the API Gateway to serve a static page saying ‘Viva!’’ to the invoker. Hint: look for the entry page “/” on the PetStore example, as the GET invocation returns a static HTML file. **Please keep in mind that the changes that you do to your API will only take place after deploying. Otherwise, they will be invisible.**

2- Now, use the same API and a new resource based on a Lambda function that outputs the following line:

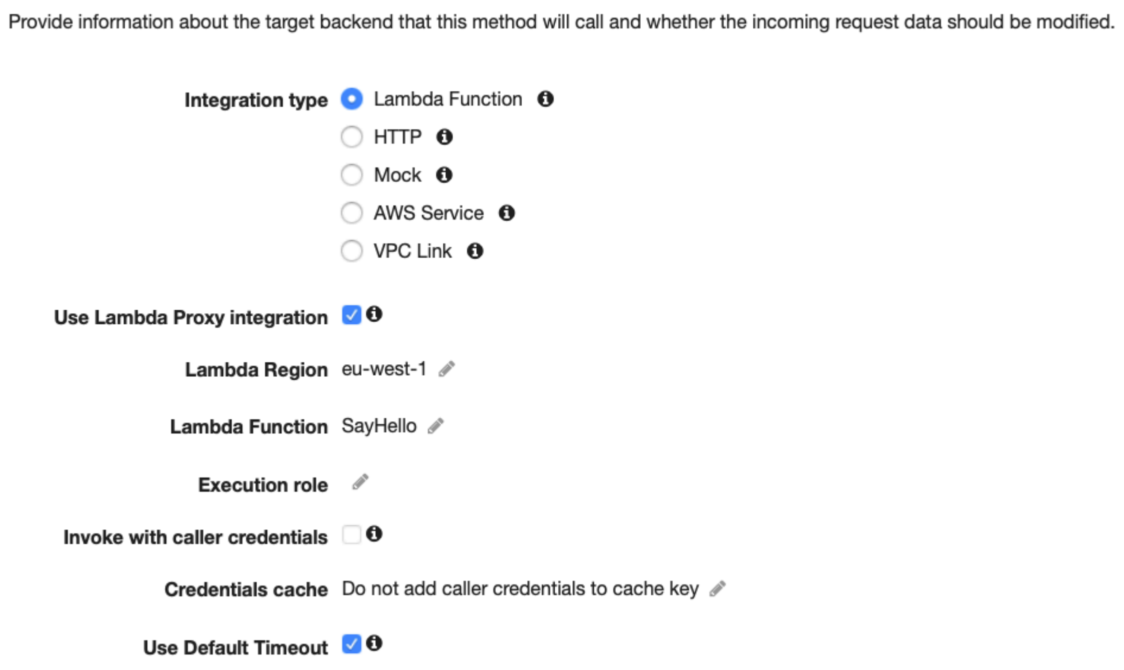
Hello <your name>, <your age>! Welcome to this <resource path>!

To solve this exercise, follow the steps in this URL: https://docs.aws.amazon.com/pt_br/apigateway/latest/developerguide/getting-started-with-lambda-integration.html

The example is written in Node.js. You may either use Node.js or some other language, like Python, but in that case, you need to do the appropriate transformations.

Please note that you should use “Lambda Proxy Integration” to provide the request data in the “event” variable of your Lambda function.

Note also that the region should be us-east-1.

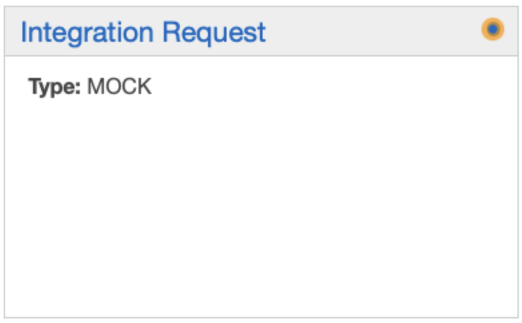

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type	● Lambda Function ○ HTTP ○ Mock ○ AWS Service ○ VPC Link
Use Lambda Proxy integration	✓
Lambda Region	eu-west-1
Lambda Function	SayHello
Execution role	
Invoke with caller credentials	☐
Credentials cache	Do not add caller credentials to cache key
Use Default Timeout	✓

3- In a Lambda function, explain the difference between the context and event parameters.

4- In the API Gateway explain the difference between Method Request, Method Response, Integration Request and Integration Response.

Click the blue and orange circles on the API Gateway service to know the answer to this question:


Integration Request

Type: MOCK

5- Now, you should create a custom authorizer with a Lambda function. This authorizer must be created in the "Authorizers" link and refer to a Lambda function that you must create. Then, you should configure the authorizer in the "Method Request" Box of the Resource. Note that this can only be seen after deployed into production.

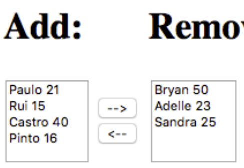
For instructions on the Lambda function you need to create refer to:

https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html

Once in place you will be unable to see the contents of the method you protected with the browser. You can use curl, as follows, for that:

```
curl -X GET 'https://2y31rjalxll.execute-api.eu-
west/production/helloworld' -H 'playauth: allow'
```

6- In this exercise you should use the example from the previous assignment, where you had to create two listboxes and adapt it to run on AWS resources.



The overall solution involves the following resources:

- An S3 object with names and ages (i.e., a file with a pair name, age in each line).
- A Lambda function that reads that object and exposes the results as JSON, as follows:

```
[{"name": "Bryan Tevez", "age": 35}, {"name": "Manuel", "age":
41}, {"name": "Alcides Pinto", "age": 28}, {"name": "Paula
Santos", "age": 20}, {"name": "Artur Bastos", "age": 21},
{"name": "Paulino Rocha", "age": 55}, {"name": "Jean Yves",
"age": 18}, {"name": "Tiago Borges", "age": 101}]
```

  You may use the following function as a base for your solution. This function is incomplete, as it misses the appropriate Cross-Origin Resource Sharing (CORS) headers (see below).

```
import json
import boto3

s3 = boto3.resource('s3')

def lambda_handler(event, context):
    #ugly solution with hard-coded names
    obj = s3.Object('es-exercises-bucket', 'ages.txt')
    body = obj.get()['Body'].read()

    result = []

    print(body)

    elements = body.decode("utf-8").split('\n');
    for e in elements:
        name, a = e.split(',')
        age = int(a)
        result.append({'name': name, 'age': age})

    return {
        'statusCode': 200,
        'body': json.dumps(result)
    }
```

- An API Gateway that exposes the Lambda function.

- Two additional objects with the HTML and JavaScript of the solution in the S3 bucket. Look for these files in the support materials of the course. You need to change the URL on the bottom of the JavaScript file to match the URL exposed by API Gateway. The idea is to expose these files via browser; this, however, requires some modifications: https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteAccessPermissionsReqd.html

  Note that you may want to adapt the solution to completely match this exercise.

- Finally, you need to set up an API Gateway to access the Lambda function. This will immediately raise a CORS problem. You may solve this in your Lambda function. Check this:

  https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-cors.html