Certified Tech Developer
The Ultimate Degree

Infraestructura III

API REST INTEGRADA

- Actividad individual
- Nivel de complejidad: medio 🔥🔥

Objetivo

Crear una API de REST que se integre a un microservicio.

Consigna

Usaremos el servicio Lambda de AWS para crear un simple microservicio back end que pueda recibir peticiones y que devuelva un saludo según los parámetros pasados. Además, desplegaremos una API en API Gateway y vincularemos la misma al microservicio.

Utilizaremos un código escrito en Node.js de ejemplo, pero recordemos que el objetivo no es centrarse en el código fuente de la Lambda sino en su uso mediante APIS.

Nuestra función de back end devolverá un JSON con la forma:
`"greeting": "Good {time}, {name} of {city}.[ Happy {day}!]"`

Desarrollo:
Ingresamos a la consola de AWS y vamos a los servicios Lambda:
https://console.aws.amazon.com/lambda

Pasos:

1. Crear la función de back end.
   Elegir Create function.
   Elegir Author from scratch.
   En Nombre colocar: GetStartedLambdaProxyIntegration.
   Elegir runtime node 14.
   En Permissions expandir Choose or Create an execution role y luego Create new role from AWS policy templates.

```
    console.log("response: " + JSON.stringify(response))
    return response;
};
```

En Rolename poner GetStartedLambdaBasicExecutionRole y Create function
En Code function, dentro del index.js, poner:

```
'use strict';
console.log('Loading hello world function');
exports.handler = async (event) => {
    let name = "you";
    let city = 'World';
    let time = 'day';
    let day = '';
    let responseCode = 200;
    console.log("request: " + JSON.stringify(event));
    if (event.queryStringParameters && event.queryStringParameters.name) {
        console.log("Received name: " + event.queryStringParameters.name);
        name = event.queryStringParameters.name;
    }
    if (event.queryStringParameters && event.queryStringParameters.city) {
        console.log("Received city: " + event.queryStringParameters.city);
        city = event.queryStringParameters.city;
    }
    if (event.headers && event.headers['day']) {
        console.log("Received day: " + event.headers.day);
        day = event.headers.day;
    }
    if (event.body) {
        let body = JSON.parse(event.body)
        if (body.time)
        time = body.time;
    }
    let greeting = `Good ${time}, ${name} of ${city}.`;
    if (day) greeting += ` Happy ${day}!`;
    let responseBody = {
        message: greeting,
        input: event
    };
    // The output from a Lambda proxy integration must be
    // in the following JSON object. The 'headers' property
    // is for custom response headers in addition to standard
    // ones. The 'body' property must be a JSON string. For
    // base64-encoded payload, you must also set the 'isBase64Encoded'
    // property to 'true'.
    let response = {
        statusCode: responseCode,
        headers: {
            "x-custom-header" : "my custom header value"
        },
```

```
        console.log("response: " + JSON.stringify(response))
        return response;
    };


        body: JSON.stringify(responseBody)
    };
```

Elegir Deploy

2. Creación de la API.
   Vamos al servicio de APIS: https://console.aws.amazon.com/apigateway
   En Create new API (crear nueva API), elegir New API (nueva API).
   En Settings (configuración).
   En API name (nombre de la API), escribir LambdaSimpleProxy.
   Seleccionar Create API (crear API).
   Creamos el recurso:
   Elegir el recurso raíz (/) en el árbol Resources (recursos).
   En el menú desplegable Actions (acciones), elegir Create Resource (crear recurso).
   Dejar Configure as proxy resource (configurar como recurso de proxy) sin marcar.
   En Resource Name escribir helloworld.
   Dejar Resource Path establecido en /helloworld.
   Dejar Enable API Gateway CORS sin marcar.
   Elegir Create Resource.
   Configuramos un método ANY.
   En la lista Resources elegir /helloworld.
   En el menú Actions elegir Create method.
   Seleccionar ANY en el menú desplegable y, a continuación, el icono de marca de verificación.
   Dejar Integration type establecido en Lambda Function.
   Seleccionar Use Lambda Proxy integration.
   Desde el menú desplegable, Lambda Region, seleccione la región en la que ha creado la función de Lambda GetStartedLambdaProxyIntegration.
   En el campo Lambda Function escriba cualquier carácter y elija GetStartedLambdaProxyIntegration en el menú desplegable.
   Dejar la casilla Use Default Timeout (usar tiempo de espera predeterminado) desactivada.
   Seleccionar Save.
   Elegir OK (aceptar) cuando se solicite Add Permission to Lambda Function.
3. Deploy de la API.
   Realizar un deploy como en el ejercicio asincrónico y darle Name: test
4. Prueba de la API:
   Podemos utilizar Curl o Postman. Las consultas GET también las podemos probar desde un Navegador web.

```
curl -X POST
'https://e44rzrbz6b.execute-api.us-east-1.amazonaws.com/test/helloworld?n
```

```
        console.log("response: " + JSON.stringify(response))
        return response;
    };
```

```
ame=Marce&city=Cordoba' -H 'content-type: application/json' -H 'day:
Martes' -d '{ "time": "evening"}'
curl -X GET
'https://e44rzrbz6b.execute-api.us-east-1.amazonaws.com/test/helloworld?n
ame=Luis&city=Cordoba' -H 'content-type: application/json' -H 'day:
Martes'
```