UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Entrega 8

Taller - 2022-1

Introducción a Cloud Computing con AWS

27 de mayo de 2022 - v1.0

Rodrigo Cayazaya Marín | 201773538-4
rodrigo.cayazaya@sansano.usm.cl

Índice

Índice de figuras

Instrucciones

El presente documento corresponde a una plantilla que incluye las informaciones que deben ser proveídas para evaluar la entrega.

La entrega se basa mayormente en capturas de pantalla de la consola, la mayoría de ellas vistas en clase más algunas acciones adicionales que deben ser descubiertas por cada estudiante.

Todos los textos en rojo a lo largo de la plantilla, junto con esta página de instrucciones, deben ser eliminadas antes de la compilación final que debe ser entregada por Moodle.

1. Funciones lambda

1.1. Nuevo Juego

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'sa-east-1'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB();

exports.handler = async ( event ) => {
    let tableContents;
    try{
        //get items from dynamo
        var params = {
            TableName: 'gato',
            ExpressionAttributeNames : {
            "#c": "gameid"
            },
            ProjectionExpression: "#c"
        };
        tableContents = await scanDB(params);
    }catch(err){
        console.log(err);
        return err;
    }

    var id = String(tableContents.length+1);
    var params2 = {
      TableName: 'gato',
      Item: {
        'gameid' : {S: id},
        'estado' : {S: '0'},
        'tablero' : {L: [
                    {S: ""},{S: ""},{S: ""},
                    {S: ""},{S: ""},{S: ""},
                    {S: ""},{S: ""},{S: ""}
                    ]},
        'turno' : {S: '0'},
        'posicion' : {S: '-'}
    }
    };

    // Call DynamoDB to read the item from the table
    await ddb.putItem(params2, function(err, data) {
        if (err) {
            console.log("Error", err);
        } else {
            console.log("Success", data);
        }
```

```
    }). promise();
    var respuesta = "Juego ";
    respuesta = respuesta.concat(id).concat(" creado\n");
    const response = {
        statusCode: 200,
        body: respuesta,
    };
    return response;
};

async function scanDB(params) {
    let dynamoContents = [];
    let items;
    do{
        items =  await ddb.scan(params).promise();
        items.Items.forEach((item) => dynamoContents.push(item));
        params.ExclusiveStartKey  = items.LastEvaluatedKey;
    }while(typeof items.LastEvaluatedKey != "undefined");
    return dynamoContents;
}
```

1.2. Estado

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'sa-east-1'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB();

exports.handler = async ( event ) => {
    let tableContents;
    try{
        //get items from dynamo
        var params = {
            TableName: 'gato',
            ExpressionAttributeNames : {
            "#b": "tablero",
            "#c": "gameid",
            "#d": "estado"
            },
            ProjectionExpression: "#b, #c, #d"
        };
        tableContents = await scanDB(params);
    }catch(err){
        console.log(err);
        return err;
    }
    const resp = [];
    const ids = [];
```

```
29      var respuesta = "";
30      var tableros = "";
31      tableContents.forEach(function(value){
32          tableros = tableros.concat("TABLERO ").concat(value["gameid"
                ].S).concat("\n");
33          tableros = tableros.concat(value["tablero"]["L"][0].S).
                concat(" | ").concat(value["tablero"]["L"][1].S).concat("
                 | ").concat(value["tablero"]["L"][2].S).concat("\n");
34          tableros = tableros.concat("---------\n");
35          tableros = tableros.concat(value["tablero"]["L"][3].S).
                concat(" | ").concat(value["tablero"]["L"][4].S).concat("
                 | ").concat(value["tablero"]["L"][5].S).concat("\n");
36          tableros = tableros.concat("---------\n");
37          tableros = tableros.concat(value["tablero"]["L"][6].S).
                concat(" | ").concat(value["tablero"]["L"][7].S).concat("
                 | ").concat(value["tablero"]["L"][8].S).concat("\n\n");
38          if(value["estado"].S == '0'){
39              ids.push(value["gameid"].S);
40              var q0 = value["tablero"]["L"][0].S;
41              var q1 = value["tablero"]["L"][1].S;
42              var q2 = value["tablero"]["L"][2].S;
43              var q3 = value["tablero"]["L"][3].S;
44              var q4 = value["tablero"]["L"][4].S;
45              var q5 = value["tablero"]["L"][5].S;
46              var q6 = value["tablero"]["L"][6].S;
47              var q7 = value["tablero"]["L"][7].S;
48              var q8 = value["tablero"]["L"][8].S;
49
50              if((q0 == q1) && (q1 == q2) && (q0 != "")){
51                  resp.push(q0);
52              }else if((q0 == q3) && (q3 == q6) && (q0 != "")){
53                  resp.push(q0);
54              }else if((q0 == q4) && (q4 == q8) && (q0 != "")){
55                  resp.push(q0);
56              }else if((q6 == q7) && (q7 == q8) && (q6 != "")){
57                  resp.push(q6);
58              }else if((q2 == q5) && (q5 == q8) && (q2 != "")){
59                  resp.push(q2);
60              }else if((q2 == q4) && (q4 == q6) && (q2 != "")){
61                  resp.push(q2);
62              }else{
63                  resp.push('0');
64                  respuesta = respuesta.concat("El juego n mero ").
                        concat(value["gameid"].S).concat(" no tiene
                        ganador\n");
65              }
66          }else{
67              respuesta = respuesta.concat("El juego n mero ").concat
                    (value["gameid"].S).concat(" tiene como ganador a ".
                    concat(value["estado"].S).concat("\n"));
68          }
69      });
70
71      for (var i = 0; i < ids.length; i++) {
```

```
72          const res = resp[i];
73          const id = ids[i];
74          if(res != '0') respuesta = respuesta.concat("El juego
                n mero ").concat(id).concat(" tiene como ganador a ".
                concat(res).concat("\n"));
75          var params2 = {
76              ExpressionAttributeValues: {
77                  ":newAttribute": {S: res}
78              },
79              Key: {
80                  "gameid": {S: id}
81              },
82              TableName: 'gato',
83              UpdateExpression: "SET estado = :newAttribute",
84          };
85          await ddb.updateItem(params2).promise();
86      }
87      tableros = tableros.concat(respuesta);
88      const response = {
89          statusCode: 200,
90          body: tableros,
91      };
92      return response;
93  };
94
95  async function scanDB(params) {
96      let dynamoContents = [];
97      let items;
98      do{
99          items =  await ddb.scan(params).promise();
100         items.Items.forEach((item) => dynamoContents.push(item));
101         params.ExclusiveStartKey  = items.LastEvaluatedKey;
102     }while(typeof items.LastEvaluatedKey != "undefined");
103     return dynamoContents;
104 }
```

1.3. Realizar Jugada

```
1   // Load the AWS SDK for Node.js
2   var AWS = require('aws-sdk');
3   // Set the region
4   AWS.config.update({region: 'sa-east-1'});
5
6   // Create the DynamoDB service object
7   var ddb = new AWS.DynamoDB();
8
9   exports.handler = async ( event ) => {
10      let items;
11      var resp = "";
12      var gameid = String(event.queryStringParameters.id);
13      var posicion = String(event.queryStringParameters.posicion);
```

```
14      var jugada = String(event.queryStringParameters.jugada);
15      var posicionInt = Number(posicion);
16      //var gameid = '10';
17      try{
18          //get items from dynamo
19          var params = {
20              TableName: 'gato',
21              Key: {
22                  "gameid": {S: gameid}
23              },
24              ExpressionAttributeNames : {
25              "#b": "tablero",
26              "#c": "gameid",
27              '#d': "turno",
28              '#e': "estado"
29              },
30              ProjectionExpression: "#b, #c, #d, #e"
31          };
32          items = await ddb.getItem(params).promise();
33      }catch(err){
34          console.log(err);
35          return err;
36      }
37      //console.log(items.Item.turno);
38      if(posicionInt>=0 && posicionInt<=8){
39          if(items.Item.estado.S == '0'){
40              if(items.Item.turno.S == '0'){
41                  if(jugada == 'X' || jugada == '0'){
42                      resp = resp.concat(await updateDB0(posicion,
                            gameid,items,jugada));
43                  }else{
44                      resp = resp.concat("Jugada incorrecta, debe ser
                            X o 0\n");
45                  }
46              }else if(items.Item.turno.S == 'X'){
47                  if(jugada != 'X'){
48                      resp = resp.concat("Jugada incorrecta, es turno
                            de X\n");
49                  }else{
50                      resp = resp.concat(await updateDB(posicion,
                            gameid,items));
51                  }
52              }else if(items.Item.turno.S == '0'){
53                   if(jugada != '0'){
54                      resp = resp.concat("Jugada incorrecta, es turno
                            de 0\n");
55                  }else{
56                      resp = resp.concat(await updateDB(posicion,
                            gameid,items));
57                  }
58              }
59          }else{
60              resp = resp.concat("Este tablero ya tiene ganador\n");
61          }
```

```
62      }else{
63          resp = resp.concat("Posici n incorrecta, debe ser entre 0 y
                8\n");
64      }
65      const response = {
66          statusCode: 200,
67          body: resp,
68      };
69      return response;
70
71  };
72
73  async function updateDB(posicion,gameid,items) {
74      var resp2;
75      if(items.Item.tablero.L[Number(posicion)].S == ""){
76          var params2 = {
77              ExpressionAttributeValues: {
78                  ":newAttribute": {S: posicion}
79              },
80              Key: {
81                  "gameid": {S: gameid}
82              },
83              TableName: 'gato',
84              UpdateExpression: "SET posicion = :newAttribute",
85          };
86          await ddb.updateItem(params2).promise();
87          resp2 = "Jugada correcta\n";
88      }
89      else{
90          resp2 = "Jugada incorrecta, ya existe un valor en esa
                posici n\n";
91      }
92      return resp2;
93  }
94
95  async function updateDB0(posicion,gameid,items,jugada) {
96      var resp2;
97      if(items.Item.tablero.L[Number(posicion)].S == ""){
98          var params2 = {
99              ExpressionAttributeValues: {
100                 ":newAttribute": {S: posicion},
101                 ":newTurno": {S: jugada}
102             },
103             Key: {
104                 "gameid": {S: gameid}
105             },
106             TableName: 'gato',
107             UpdateExpression: "SET posicion = :newAttribute, turno =
                    :newTurno"
108         };
109         await ddb.updateItem(params2).promise();
110         resp2 = "Jugada correcta\n";
111     }
112     else{
```

```
113         resp2 = "Jugada incorrecta, ya existe un valor en esa
                posici n\n";
114     }
115     return resp2;
116 }
```

1.4. Comprometer Jugada

```
1  // Load the AWS SDK for Node.js
2  var AWS = require('aws-sdk');
3  // Set the region
4  AWS.config.update({region: 'sa-east-1'});
5
6  // Create the DynamoDB service object
7  var ddb = new AWS.DynamoDB();
8
9  exports.handler = async ( event ) => {
10     let items;
11     var gameid = String(event.queryStringParameters.id);
12     //var gameid = '10';
13     try{
14         //get items from dynamo
15         var params = {
16             TableName: 'gato',
17             Key: {
18                 "gameid": {S: gameid}
19             },
20             ExpressionAttributeNames : {
21             "#b": "tablero",
22             "#c": "gameid",
23             '#d': "turno",
24             '#e': "posicion"
25             },
26             ProjectionExpression: "#b, #c, #d, #e"
27         };
28         items = await ddb.getItem(params).promise();
29     }catch(err){
30         console.log(err);
31         return err;
32     }
33     var resp = "";
34     var pos = items.Item.posicion.S;
35     if(pos != "-"){
36         var turno = items.Item.turno.S;
37         var tab = items.Item.tablero.L;
38         var nuevoTurno;
39         if(turno == 'X') nuevoTurno = 'O';
40         if(turno == 'O') nuevoTurno = 'X';
41         tab[Number(pos)].S = turno;
42         var params2 = {
43             ExpressionAttributeValues: {
```

```
                ":newTab": {L: tab},
                ":newPos": {S: '-'},
                ":newTurno": {S: nuevoTurno}
            },
            Key: {
                "gameid": {S: gameid}
            },
            TableName: 'gato',
            UpdateExpression: "SET tablero = :newTab, posicion = :
                newPos, turno = :newTurno"
        };
        await ddb.updateItem(params2).promise();
        resp = resp.concat("Jugada realizada\n");
    }else{
        resp = resp.concat("Debes realizar una jugada primero\n");
    }
    const response = {
        statusCode: 200,
        body: resp,
    };
    return response;

};
```

2. Creación de API Gateway

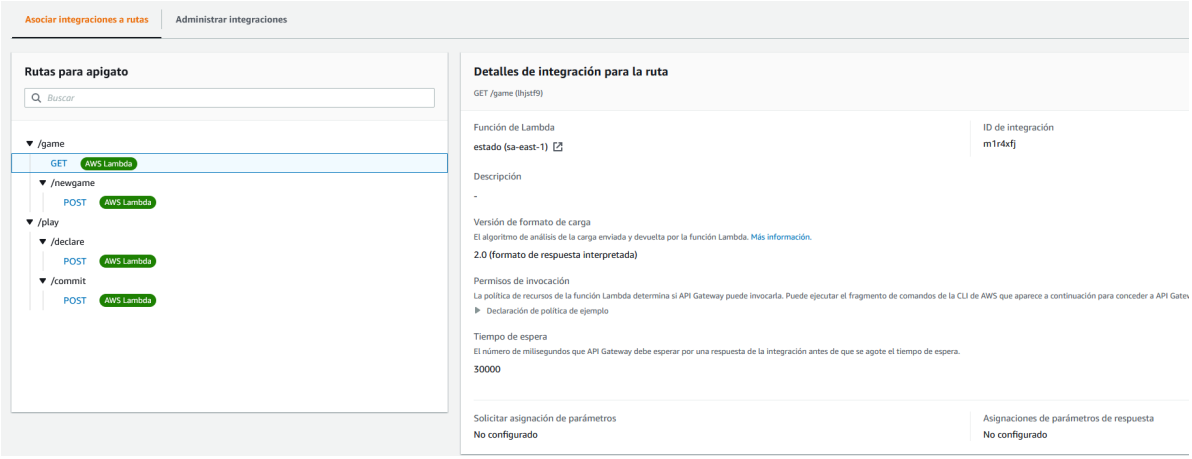

Figura 1: Captura de pantalla de la consola de API Gateway mostrando las rutas de la API

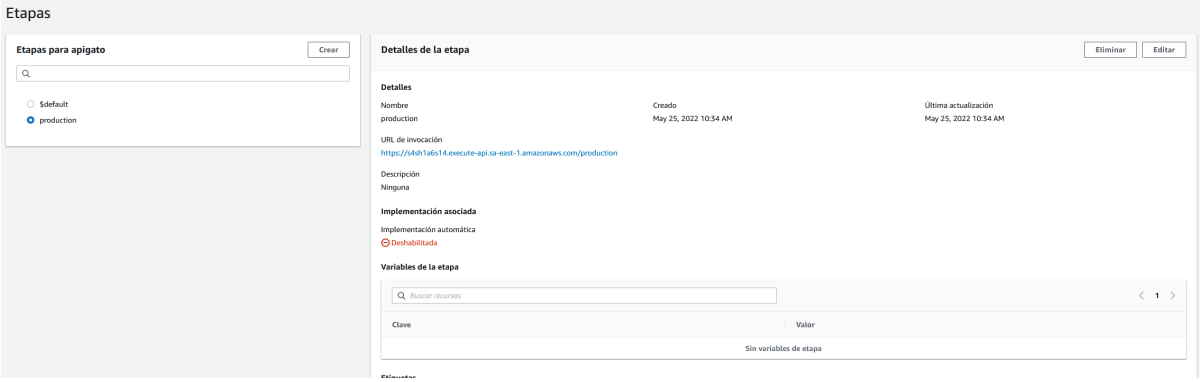

Figura 2: Captura de pantalla de la Stage “production”

3. Prueba de la API

Procedimiento CURL utilizado que demuestra el correcto funcionamiento de la API:

```
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/game/newgame"
curl -X GET "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/game"
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/play/declare?id=1&posicion=0&jugada=X"
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/play/commit?id=1"
curl -X GET "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/game"
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/play/declare?id=1&posicion=4&jugada=O"
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/play/commit?id=1"
curl -X GET "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/game"
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/play/declare?id=1&posicion=4&jugada=X"
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/play/declare?id=1&posicion=2&jugada=X"
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/play/commit?id=1"
curl -X GET "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/game"
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/play/declare?id=1&posicion=7&jugada=O"
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/play/commit?id=1"
curl -X GET "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/game"
```

```
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/play/declare?id=1&posicion=1&jugada=X"
curl -X POST "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/play/commit?id=1"
curl -X GET "https://s4sh1a6s14.execute-api.sa-east-1.amazonaws.com/game"
```

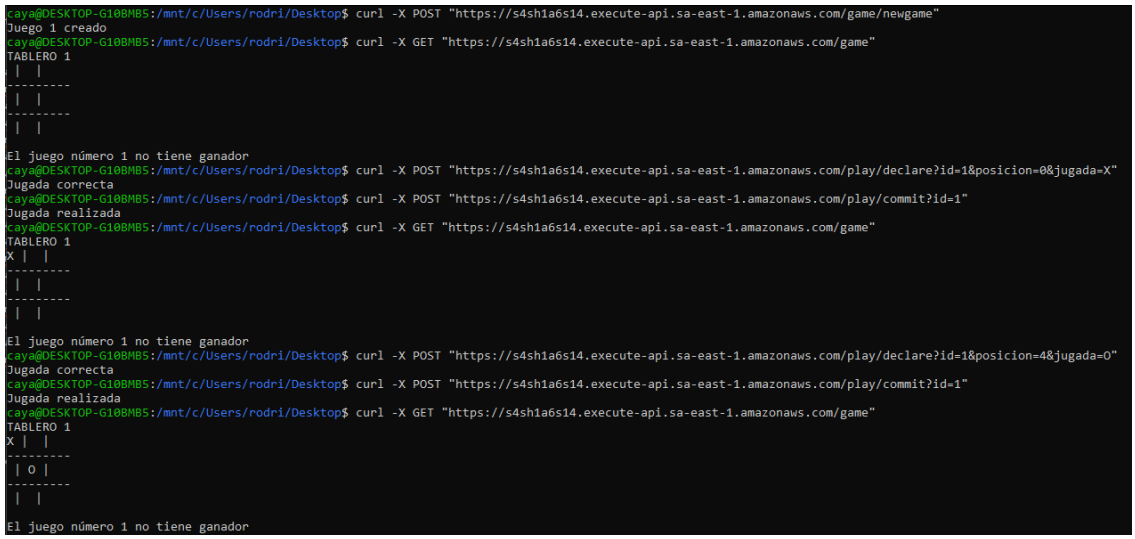

Figura 3: Captura de pantalla mostrando el procedimiento de prueba en ejecución

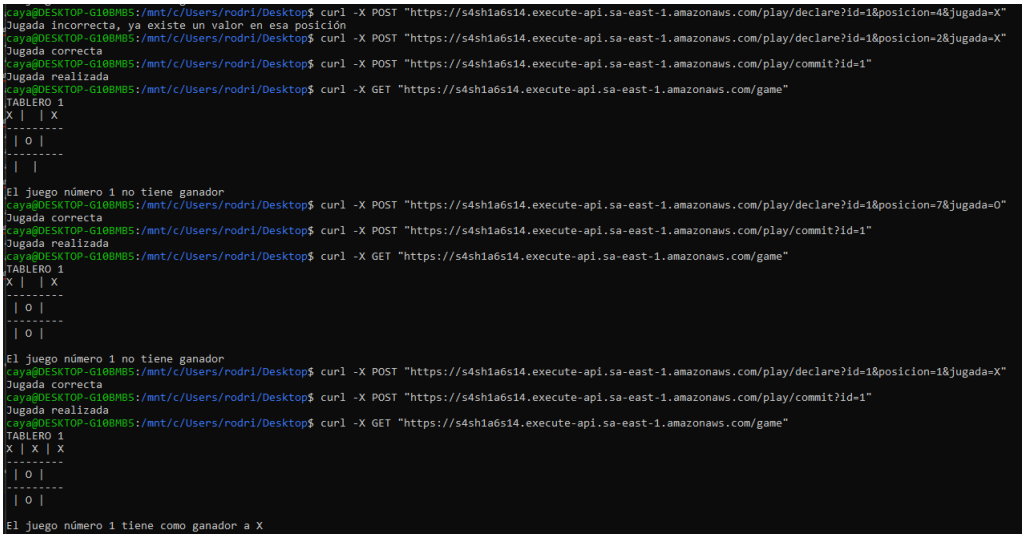

Figura 4: Captura de pantalla mostrando el procedimiento de prueba en ejecución

4. Dominio personalizado

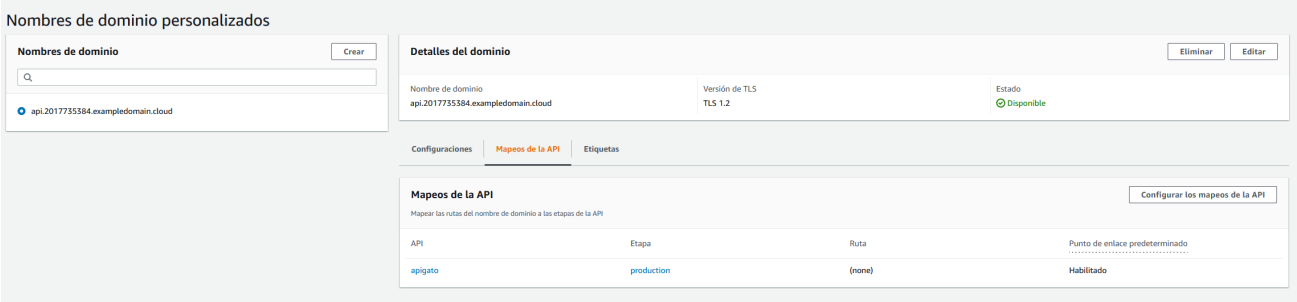

Figura 5: Captura de pantalla del API Mapping en la sección Custom Domain Names