Back-end RAD Event API and Database Deliverables

Andrea

1.URL of the Publicly Accessible Endpoint:

REST API with one endpoint named: /events

- **read**: https://boee9mqtbe.execute-api.us-east-1.amazonaws.com/dev/events

- **delete**: https://boee9mqtbe.execute-api.us-east-1.amazonaws.com/dev/events/{id}

- **get**: https://boee9mqtbe.execute-api.us-east-1.amazonaws.com/dev/events/{id}

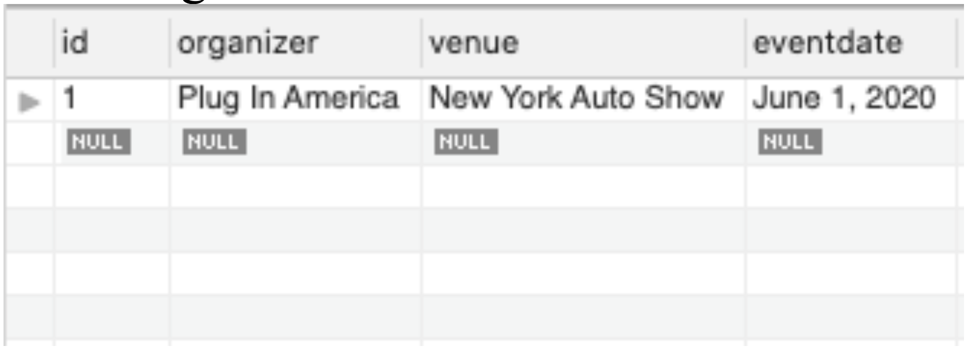
Testing in API proxy (postman) with Request Body

- **create**: https://boee9mqtbe.execute-api.us-east-1.amazonaws.com/dev/events

- **update**: https://boee9mqtbe.execute-api.us-east-1.amazonaws.com/dev/events/{id}

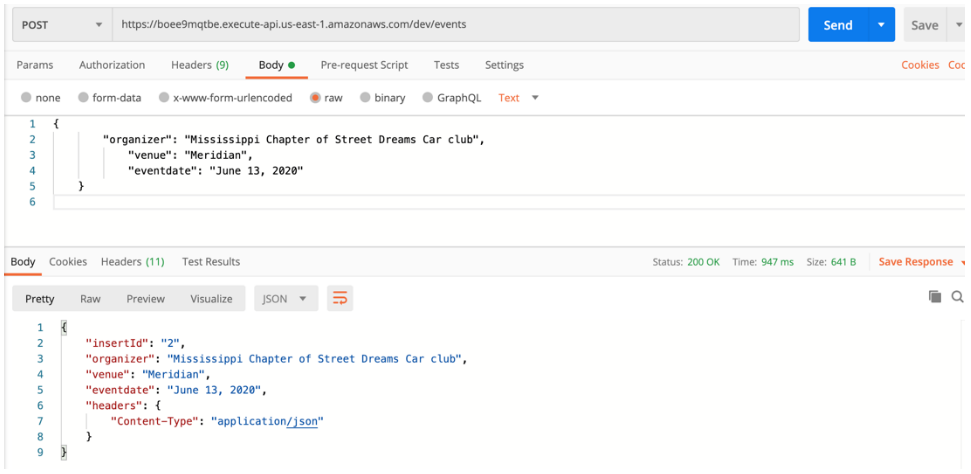
```
Request Body
{
        "organizer": "Mississippi Chapter of Street Dreams Car club",
        "venue": "Meridian",
        "eventdate": "June 13, 2020"
}
```

- Testing in Postman
    - Checking the data with database



	id	organizer	venue	eventdate
▶	1	Plug In America	New York Auto Show	June 1, 2020
	NULL	NULL	NULL	NULL

    - POST in Postman



POST https://boee9mqtbe.execute-api.us-east-1.amazonaws.com/dev/events

Body: raw, Text

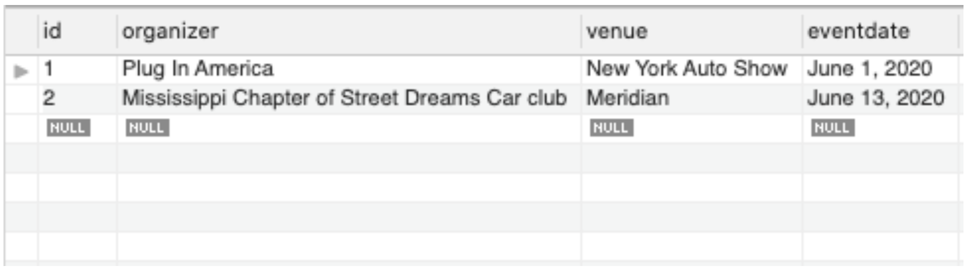
```
{
        "organizer": "Mississippi Chapter of Street Dreams Car club",
            "venue": "Meridian",
            "eventdate": "June 13, 2020"
    }

```

Status: 200 OK Time: 947 ms Size: 641 B

```
{
    "insertId": "2",
    "organizer": "Mississippi Chapter of Street Dreams Car club",
    "venue": "Meridian",
    "eventdate": "June 13, 2020",
    "headers": {
        "Content-Type": "application/json"
    }
}
```

    - Data insert into database



	id	organizer	venue	eventdate
▶	1	Plug In America	New York Auto Show	June 1, 2020
	2	Mississippi Chapter of Street Dreams Car club	Meridian	June 13, 2020
	NULL	NULL	NULL	NULL

Apply the Same way to the UPDATE method.

2. Environment Setup and Review Pointers

Assignment Approach:

- Using AWS user interface to implement the assignment for demonstrating AWS infrastructure.
- Selecting Javascript to implement Lambda function for lightweight web applications.
- Integrating AWS Lambda with AWS RDS MySQL query.
- Setting up API Gateway with mapping template and using AWS Lambda Proxy for integration.
- Implementing Fron-end service with React with css flexbox styling.
- Integrating AWS API Gateway endpoint with React by enabling CORS (Cross-origin resource sharing).
- Testing CORS with Curl(Client url) command-line tool by transfer data from or to a server, using supported protocols.
- Deploying App on AWS S3 bucket with Public Permission and Public Bucket Policy.

Future development:

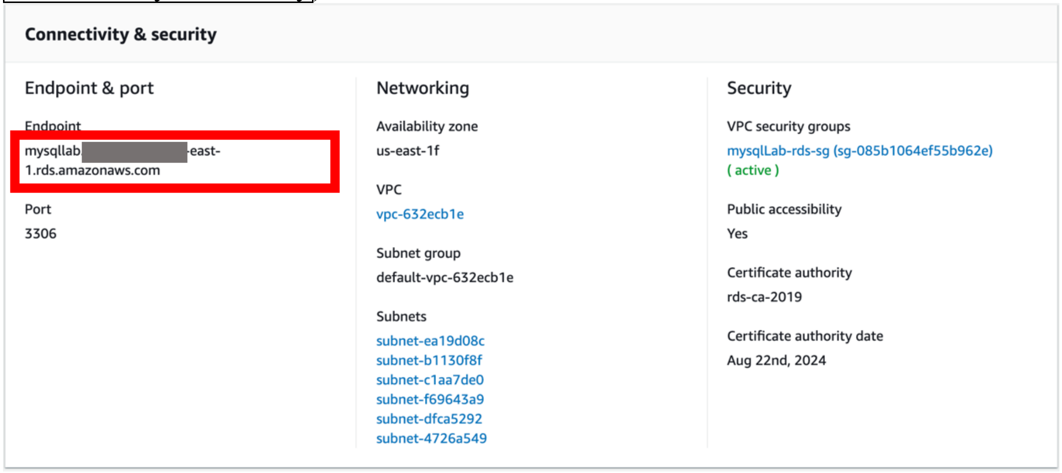
- Using Java with it's great performance to build scalable web apps with cross-platform functionality.
- Using Python with it's robust features provided by Django to build complex web applications.
- Using Swagger frame work to share a large REST API documentation.
- Using Serverless framwork to deploy AWS infrastructure resources(Lambda, RDS, API Gateway, S3, SNS, DynamoDB).
- Using React Redux for a better responsive single webpage.

Environment Setup and Step:

Create RDS MySQL — AWS Serverless to MySQL

1. Select the MySQL Database and Select the Free Tire for MySQL
2. Setting on the DB cluster identifier, Master Username and Master Password

   For your local MySQL:
   DB cluster identifier as the database name,
   Master Username as Username,
   Master Password as password
   Endpoint as Hostname
   (After the Database is created, you will get an EndPoint at RDS Database Connectivity & security)



Connectivity & security

Endpoint & port

Endpoint
mysqllab[redacted]east-1.rds.amazonaws.com

Port
3306

Networking

Availability zone
us-east-1f

VPC
vpc-632ecb1e

Subnet group
default-vpc-632ecb1e

Subnets
subnet-ea19d08c
subnet-b1130f8f
subnet-c1aa7de0
subnet-f69643a9
subnet-dfca5292
subnet-4726a549

Security

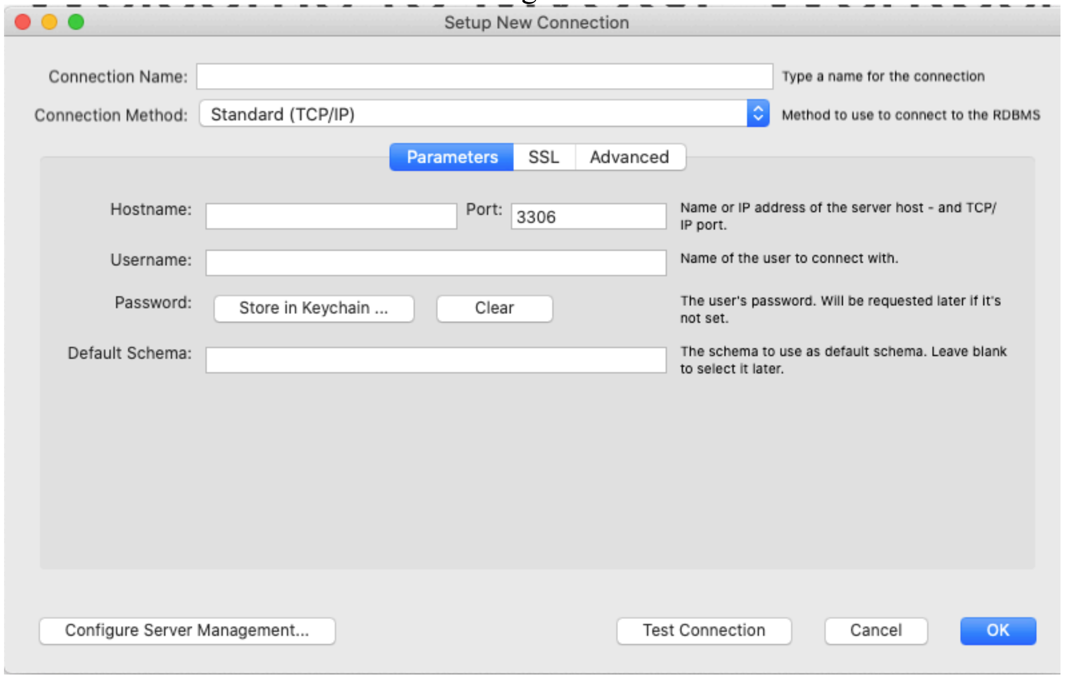
VPC security groups
mysqlLab-rds-sg (sg-085b1064ef55b962e)
(active)

Public accessibility
Yes

Certificate authority
rds-ca-2019

Certificate authority date
Aug 22nd, 2024

Fig. 1



Setup New Connection

Connection Name: [ ] Type a name for the connection

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Hostname: [ ] Port: 3306 Name or IP address of the server host - and TCP/IP port.

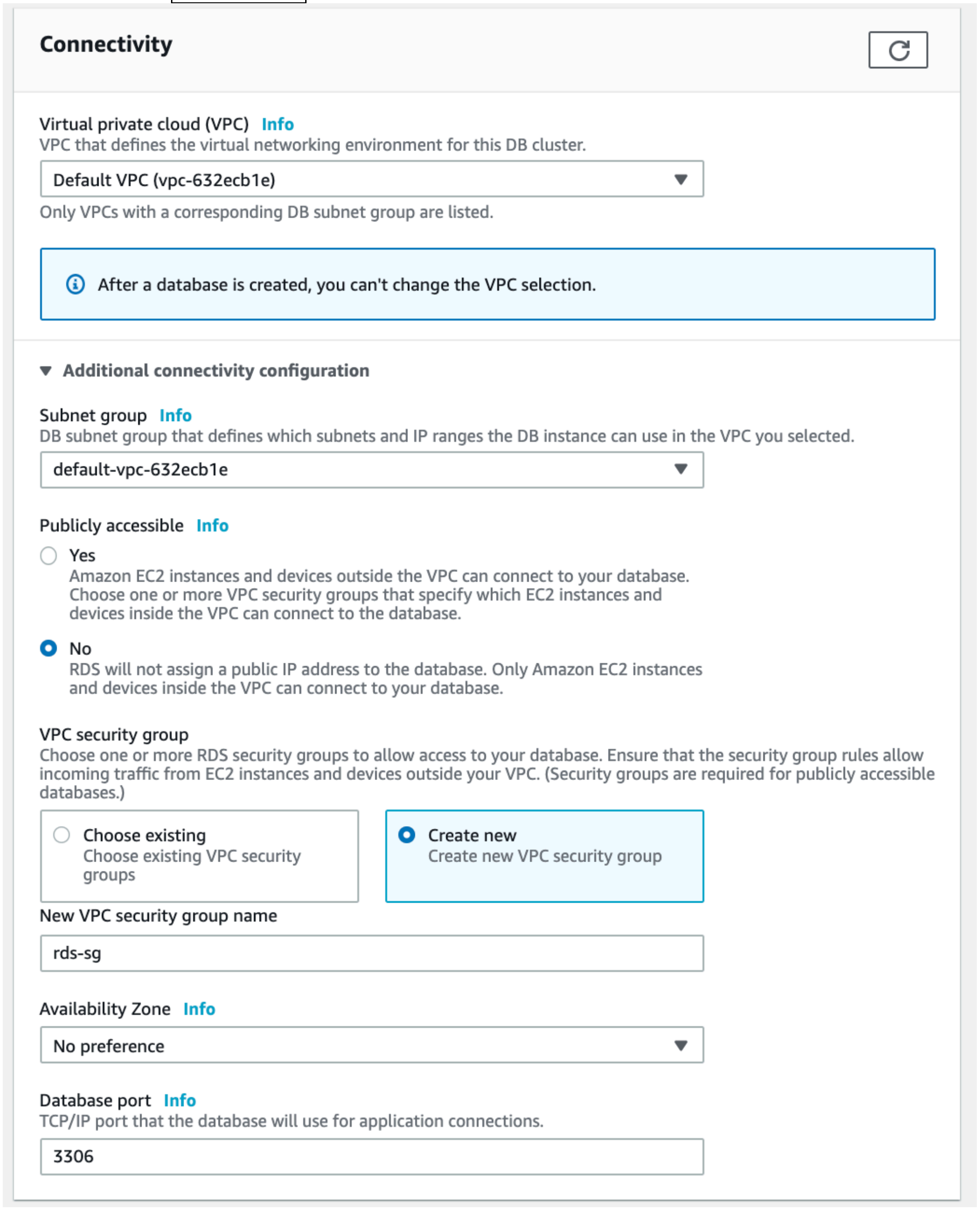
Username: [ ] Name of the user to connect with.

Password: Store in Keychain ... Clear The user's password. Will be requested later if it's not set.

Default Schema: [ ] The schema to use as default schema. Leave blank to select it later.

Fig. 2

3. Connectivity for MySQL
   At $\boxed{\text{Additional connectivity configuration}}$
   - Publicly accessible
     Select $\boxed{\text{Yes}}$
     (We can improve the security by setting VPC to connect your local MySQL with AWS RDS)
   - VPC security group
     Select $\boxed{\text{Create new}}$



Connectivity

Virtual private cloud (VPC) Info
VPC that defines the virtual networking environment for this DB cluster.
Default VPC (vpc-632ecb1e)
Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change the VPC selection.

Additional connectivity configuration

Subnet group Info
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.
default-vpc-632ecb1e

Publicly accessible Info
Yes
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.
[✓ No]
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group
Choose one or more RDS security groups to allow access to your database. Ensure that the security group rules allow incoming traffic from EC2 instances and devices outside your VPC. (Security groups are required for publicly accessible databases.)
Choose existing
Choose existing VPC security groups
[✓ Create new]
Create new VPC security group

New VPC security group name
rds-sg

Availability Zone Info
No preference

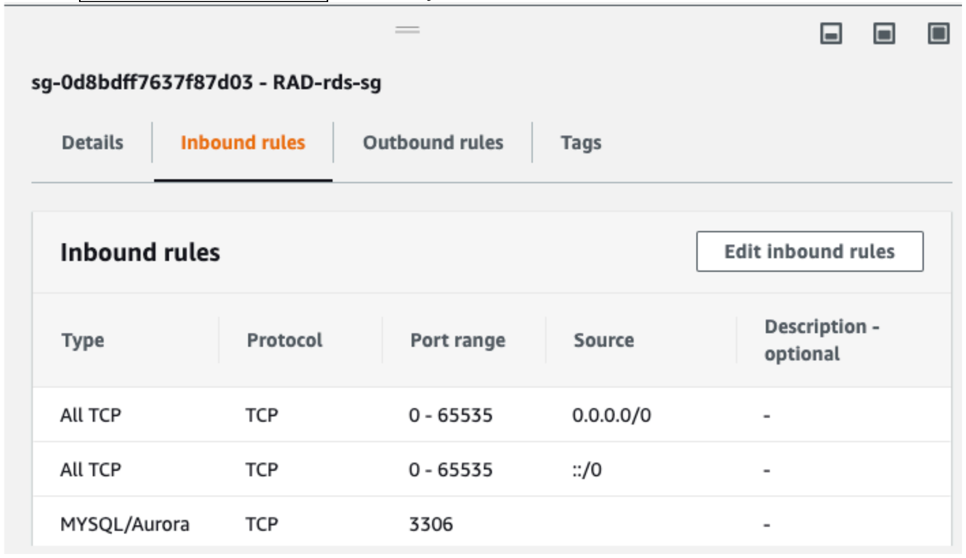
Database port Info
TCP/IP port that the database will use for application connections.
3306

Fig. 3

4. Click "Create database"
5. After the Database is created, checking the details of your MySQL.

Connect to Your Local MySQL (from your computer)

- At AWS RDS (as show in Fig. 1) Select Connectivity & security and Click VPC security groups
- Click Edit Inbound rules, beside your default Inbound roles.



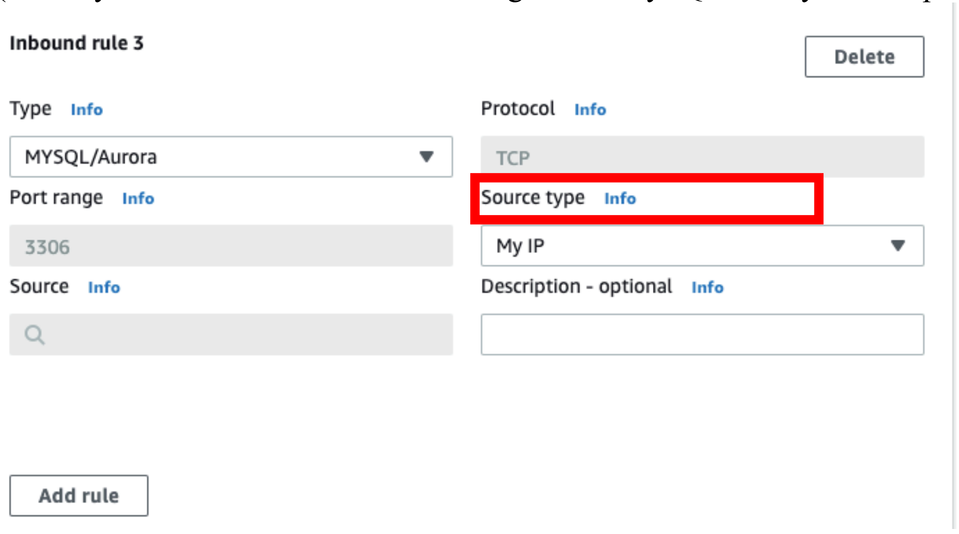
sg-0d8bdff7637f87d03 - RAD-rds-sg

Inbound rules

Type	Protocol	Port range	Source	Description - optional
All TCP	TCP	0 - 65535	0.0.0.0/0	-
All TCP	TCP	0 - 65535	::/0	-
MYSQL/Aurora	TCP	3306		-

Fig 4.

- Then Click Add rule , Select My IP For Source type.
  (Allow you to connect AWS RDS through local MySQL from your computer)

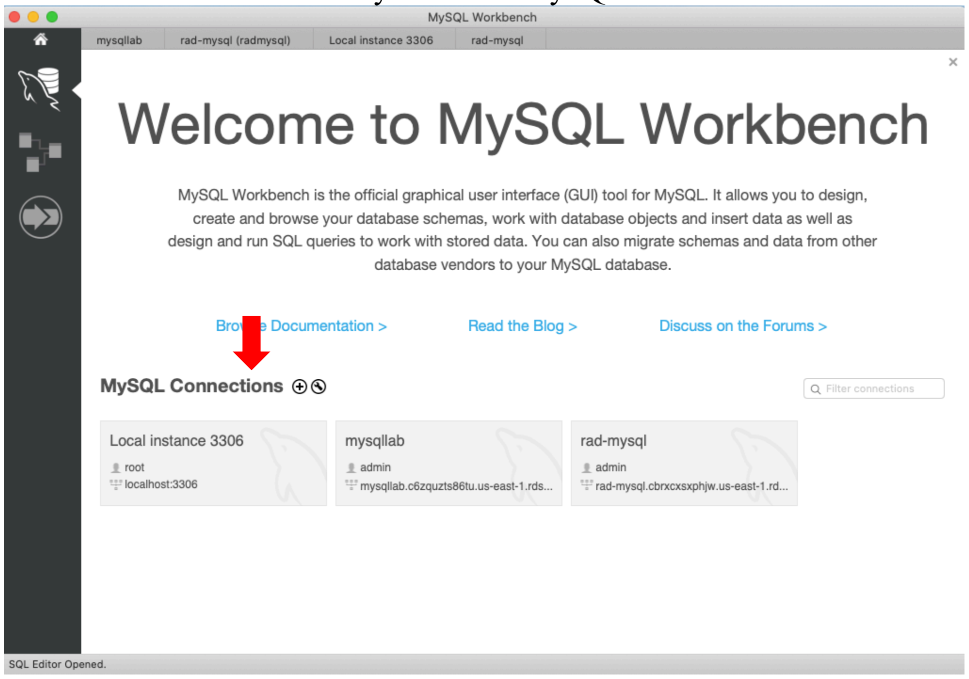


Inbound rule 3

Type: MYSQL/Aurora
Protocol: TCP
Port range: 3306
Source type: My IP
Source:
Description - optional:

Fig. 5

- Create the connections from your Local MySQL Workbench



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Connection	User	Host
Local instance 3306	root	localhost:3306
mysqllab	admin	mysqllab.c6zquzts86tu.us-east-1.rds...
rad-mysql	admin	rad-mysql.cbrxcxsxphjw.us-east-1.rd...

SQL Editor Opened.

Fig. 6

- Sign in with your AWS RDS Service in Fig. 1 and Fig. 2.

Select the database and Create the table

- Query For creating table and data

```
USE radmysql;

#DROP TABLE EventsTable;

CREATE TABLE EventsTable (
EventID int PRIMARY KEY NOT NULL AUTO_INCREMENT,
Organizer varchar(255) NULL,
Venue varchar(255) NULL,
EventDate varchar(255) NULL
);

SELECT * FROM EventsTable;

INSERT INTO EventsTable (Organizer, Venue, EventDate)
VALUE ('Plug In America','New York Auto Show', 'June 1, 2020');

SELECT * FROM EventsTable;
```

Create Lambda and API Gateway (Nodejs) — AWS Serverless to RDS MySQL

1. *Create NodeJS Code*

- Create package.json

```
npm init
```

- The package.json for your reference.

```
{
  "name": "mysql-lambda",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Andrea Chang",
  "license": "ISC",
  "dependencies": {
    "mysql": "^2.18.1"
  }
}
```

- Install the “mysql” library for connecting the MySQL

```
npm install --save mysql
```

- Create Lambda function in index.js
  (example for the lambda update function, more examples for implementing REST api function at Github link)

```
const mysql = require('mysql');

// connect Lambda with AWS RDS MySQL
const db = mysql.createConnection({
  host     : process.env.RDS_HOSTNAME,
  user     : process.env.RDS_USERNAME,
  password : process.env.RDS_PASSWORD,
  port     : process.env.RDS_PORT,
  database:'radmysql'
});

exports.handler = (event, context, callback) => {

  context.callbackWaitsForEmptyEventLoop = false;

  // get the dynamic parameters from the routing input
  // passing the default value to
  // prevent the error 'Internal server error' for
  // cannot destructure property 'undefind' or 'null'
  const id = event.queryStringParameters.EventID || 1;
  const Organizer = event.queryStringParameters.Organizer|| "Plug In America";
  const Venue = event.queryStringParameters.Venue|| "New York Auto Show";
  const EventDate = event.queryStringParameters.EventDate|| "June 1, 2020";

  // create the MySQL query for updating data in MySQL from Node.js
  const sql = `UPDATE EventsTable SET Organizer= ?, Venue= ?, EventDate= ? WHERE Ev
entID = ?`;

  db.query(sql, [`${Organizer}`, `${Venue}`, `${EventDate}`, `${id}`], function (err, result) {

    if (err) throw err;

    // check the output is matching
    // the id we pass in, for
    // updating it with the new values (development purpose)
    console.log('Update EventID : ' + `${id}` + ', Organizer : ' + `${Organizer}` + ', Venue : ' +
 `${Venue}` + ', EventDate : ' + `${EventDate}`);

    // have to return the status code and the body with 'response' for
    // configuring it with GatewayAPI
    callback(null,{
      statusCode: 200,
      body: JSON.stringify({response: result})
    });
  });
```

- Zip all the node_modules, index.js and package.json
  Upload the zip file
  Save the zip file
  The library and source code are uploaded.

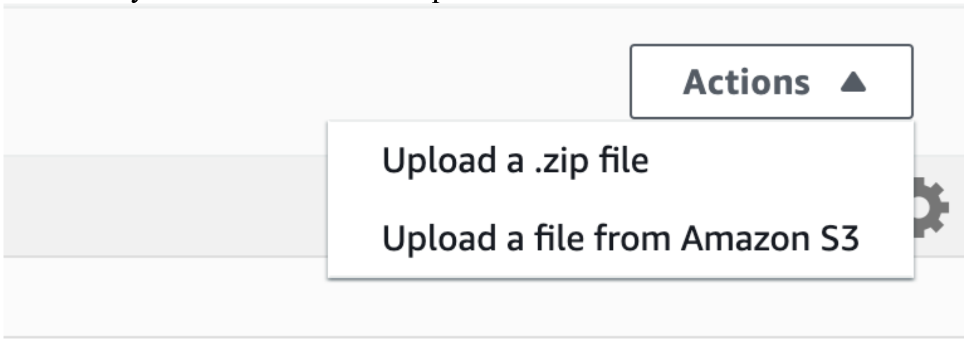
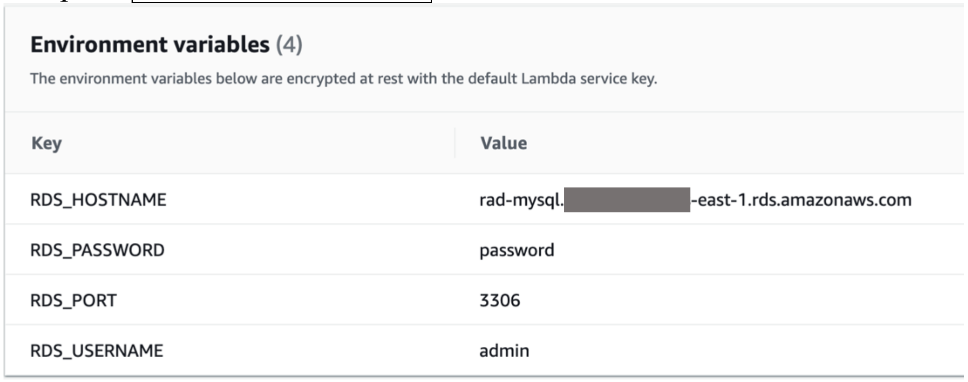


Fig. 7

- Setup the Environment Variables and Save



Environment variables (4)

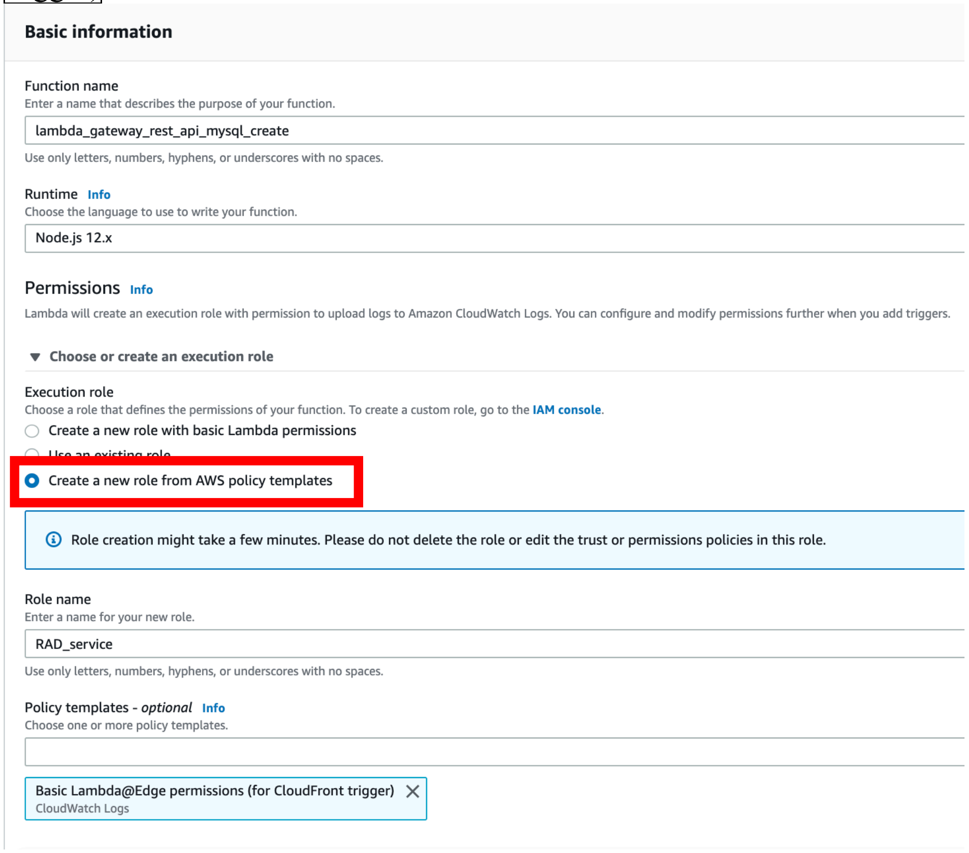
The environment variables below are encrypted at rest with the default Lambda service key.

Key	Value
RDS_HOSTNAME	rad-mysql.[redacted]-east-1.rds.amazonaws.com
RDS_PASSWORD	password
RDS_PORT	3306
RDS_USERNAME	admin

Fig. 8

2. *Create Lambda*

- Click Create function and select the "Author from scratch" and follow the setting. For Permission:
- Select Create a new role from AWS policy templates (Role name as you IAM, Identity and Access Management, roles name)
- Policy templates Select Basic Lambda@Edge permissions (for CloudFront trigger)



Basic information

Function name
Enter a name that describes the purpose of your function.
lambda_gateway_rest_api_mysql_create
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function.
Node.js 12.x

Permissions Info
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ Choose or create an execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
○ Create a new role with basic Lambda permissions
○ Use an existing role
● Create a new role from AWS policy templates

ⓘ Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

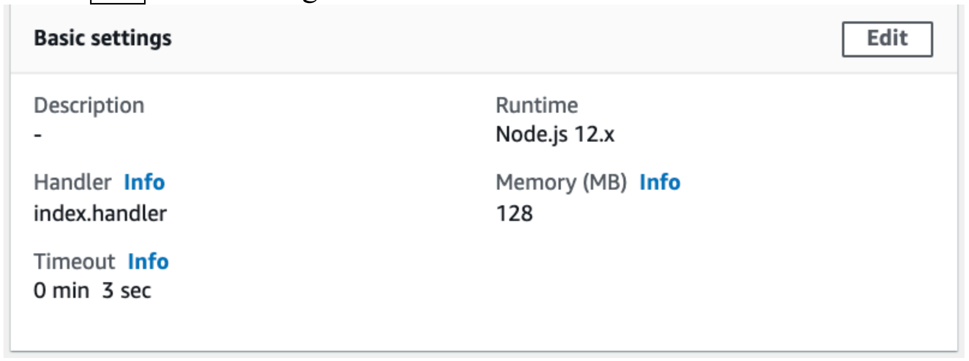
Role name
Enter a name for your new role.
RAD_service
Use only letters, numbers, hyphens, or underscores with no spaces.

Policy templates - *optional* Info
Choose one or more policy templates.

Basic Lambda@Edge permissions (for CloudFront trigger) ✕
CloudWatch Logs

Fig. 9

- Click "Create function"
- After the lambda is created, we need to Attach new policies for your IAM role("service-role/RAD_service)
- Select Edit Basic settings



Basic settings | Edit

Description
-

Runtime
Node.js 12.x

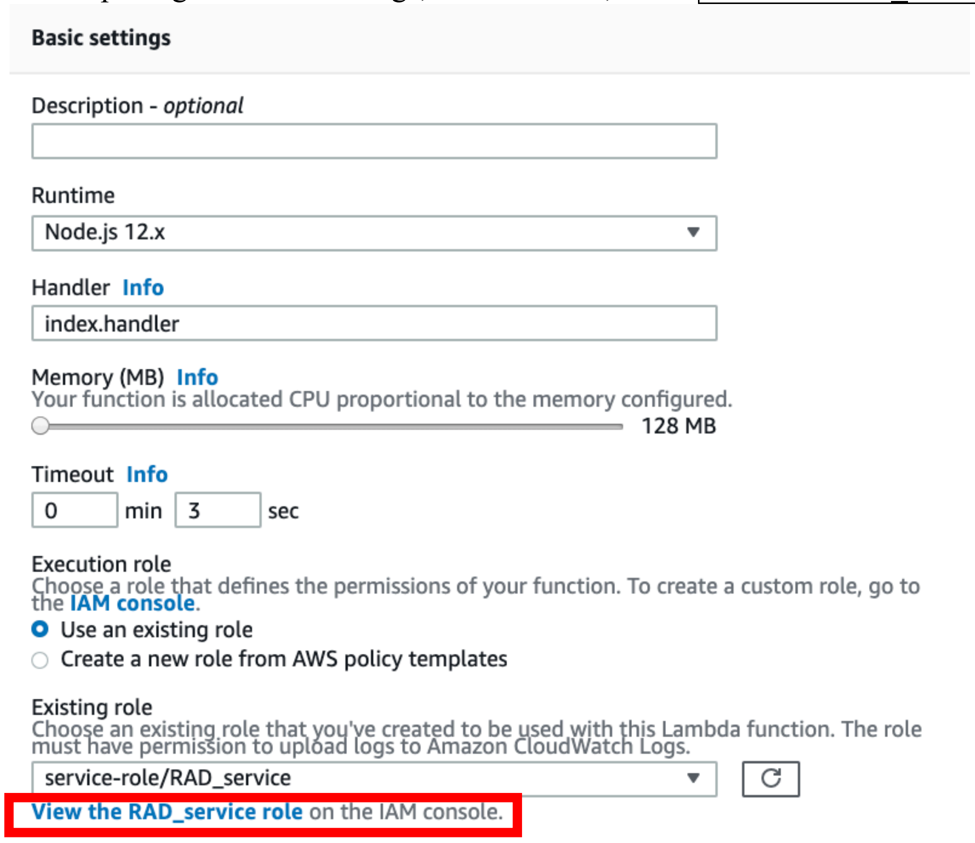
Handler Info
index.handler

Memory (MB) Info
128

Timeout Info
0 min 3 sec

Fig.10

- After opening the Basic settings, at the bottom, Click View the RAD_service role



Basic settings

Description - *optional*

Runtime: Node.js 12.x

Handler Info: index.handler

Memory (MB) Info
Your function is allocated CPU proportional to the memory configured.
128 MB

Timeout Info: 0 min 3 sec

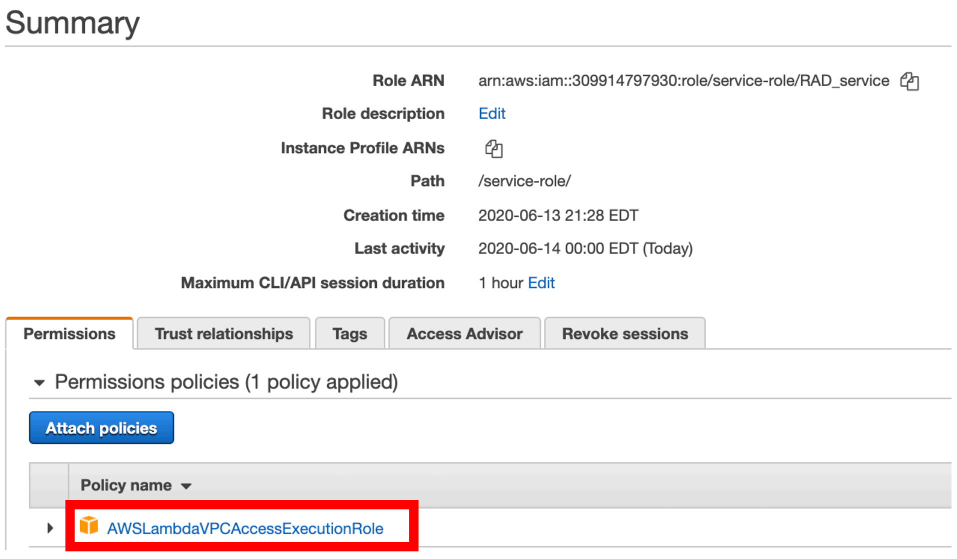
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
[✓ Use an existing role]
Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
service-role/RAD_service

View the RAD_service role on the IAM console.

Fig. 11

- Select Attach polices, Search AWSLambdaVPCAccessExecutionRole



Summary

Role ARN	arn:aws:iam::309914797930:role/service-role/RAD_service
Role description	Edit
Instance Profile ARNs	
Path	/service-role/
Creation time	2020-06-13 21:28 EDT
Last activity	2020-06-14 00:00 EDT (Today)
Maximum CLI/API session duration	1 hour Edit

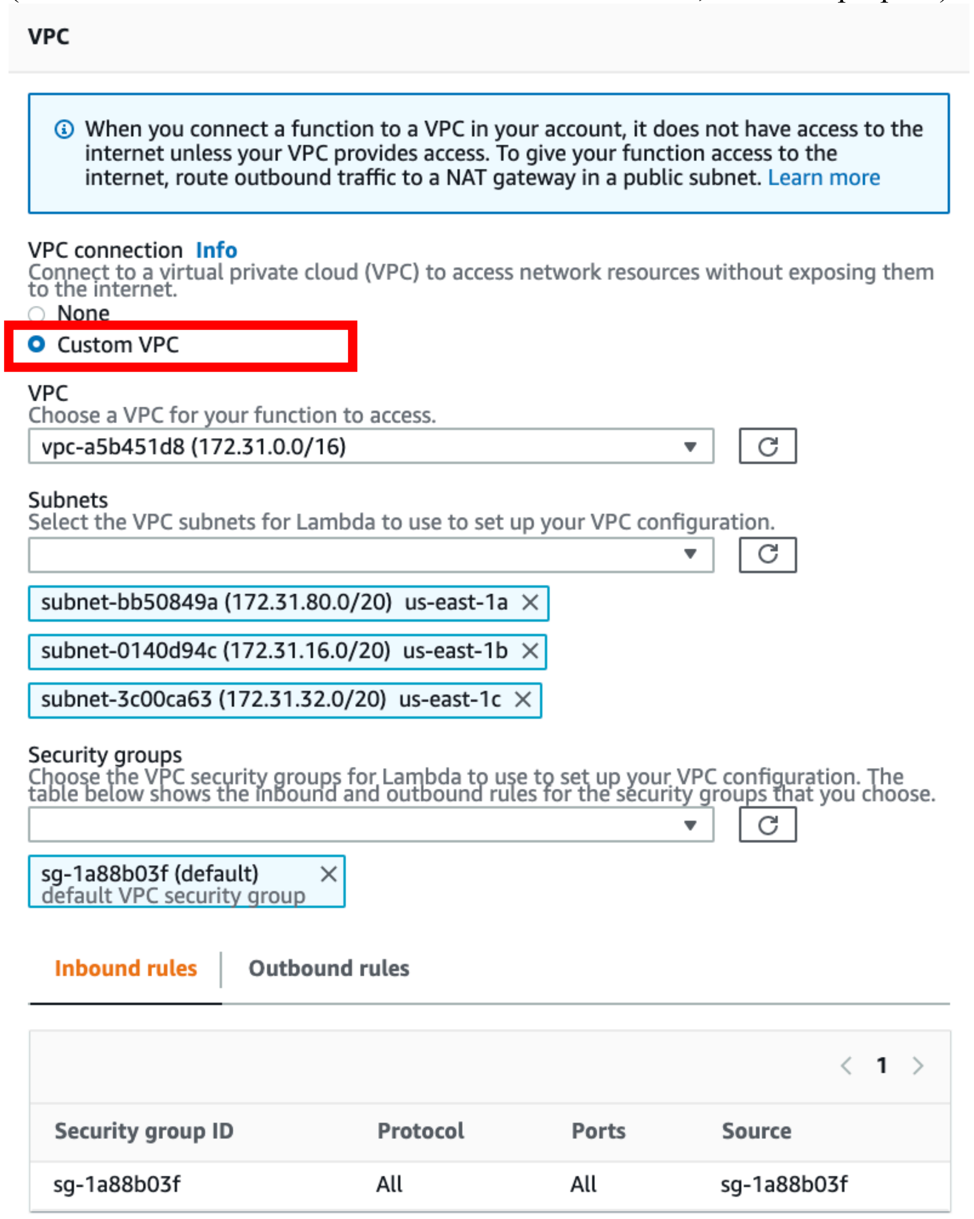
Permissions policies (1 policy applied)

Attach policies

Policy name
AWSLambdaVPCAccessExecutionRole

Fig. 12

- And then configure the security group For VPC
- Select Custom VPC , Select sg-1a88b03f (default) For Security group (We allow all the TCP access to our Lambda function, for demo purpose)



VPC

When you connect a function to a VPC in your account, it does not have access to the internet unless your VPC provides access. To give your function access to the internet, route outbound traffic to a NAT gateway in a public subnet. Learn more

VPC connection Info
Connect to a virtual private cloud (VPC) to access network resources without exposing them to the internet.
None [✓ Custom VPC]

VPC
Choose a VPC for your function to access.
vpc-a5b451d8 (172.31.0.0/16)

Subnets
Select the VPC subnets for Lambda to use to set up your VPC configuration.
subnet-bb50849a (172.31.80.0/20) us-east-1a
subnet-0140d94c (172.31.16.0/20) us-east-1b
subnet-3c00ca63 (172.31.32.0/20) us-east-1c

Security groups
Choose the VPC security groups for Lambda to use to set up your VPC configuration. The table below shows the inbound and outbound rules for the security groups that you choose.
sg-1a88b03f (default)
default VPC security group

Inbound rules | Outbound rules

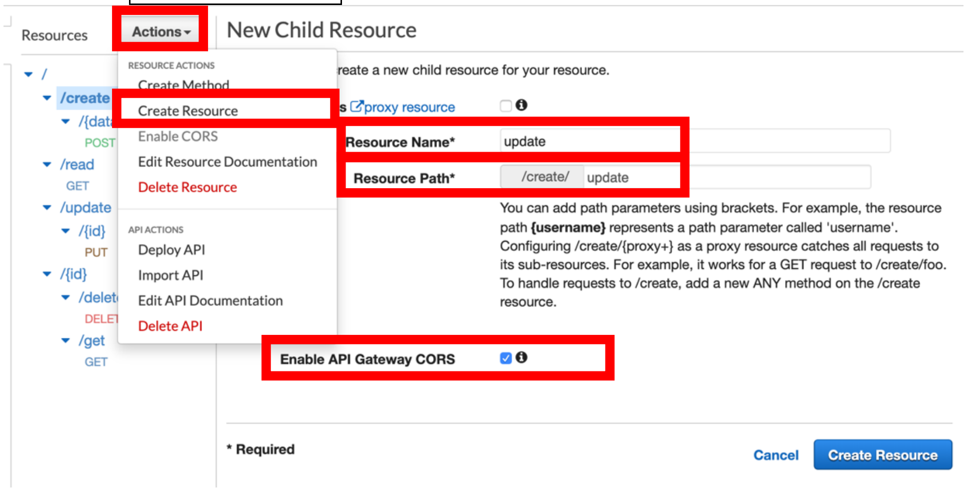
Security group ID	Protocol	Ports	Source
sg-1a88b03f	All	All	sg-1a88b03f

Fig. 13

3. *Create Gateway API*

Create API and create Resource For events

- Click Actions, Select Create Resource
- Type update For Resource Name and Resource Path
- Click Enable API Gateway CORS
- Click Create Resource



Create a new child resource for your resource.

Configure as proxy resource ☐

Resource Name* update

Resource Path* /create/ update

You can add path parameters using brackets. For example, the resource path **{username}** represents a path parameter called 'username'. Configuring /create/{proxy+} as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /create/foo. To handle requests to /create, add a new ANY method on the /create resource.

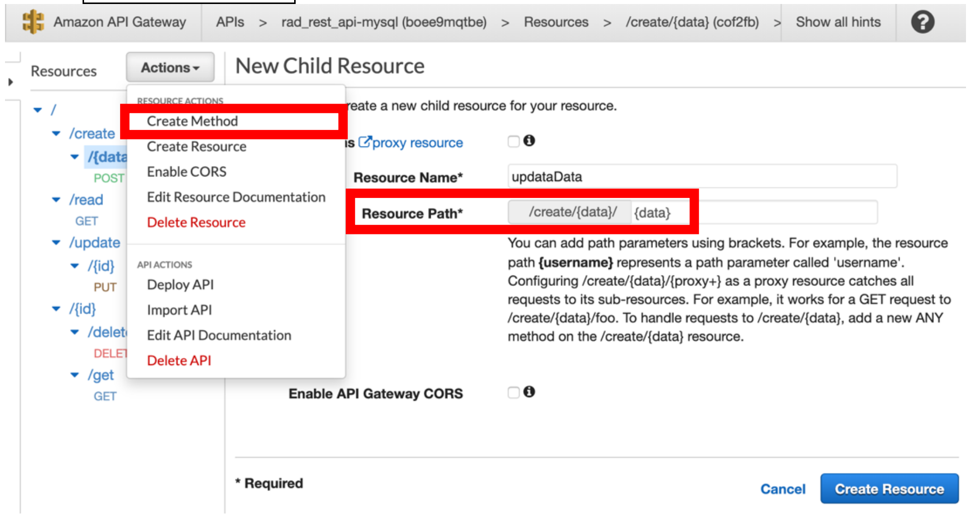
Enable API Gateway CORS ☑

* Required

Fig. 14

Create request body For update method

- Click Actions, Select Create Method
- Select update For Method Name and {data}For Resource Path
- Click Enable API Gateway CORS
- Click Create Resource



Create a new child resource for your resource.

Configure as proxy resource ☐

Resource Name* updataData

Resource Path* /create/{data}/ {data}

You can add path parameters using brackets. For example, the resource path **{username}** represents a path parameter called 'username'. Configuring /create/{data}/{proxy+} as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /create/{data}/foo. To handle requests to /create/{data}, add a new ANY method on the /create/{data} resource.

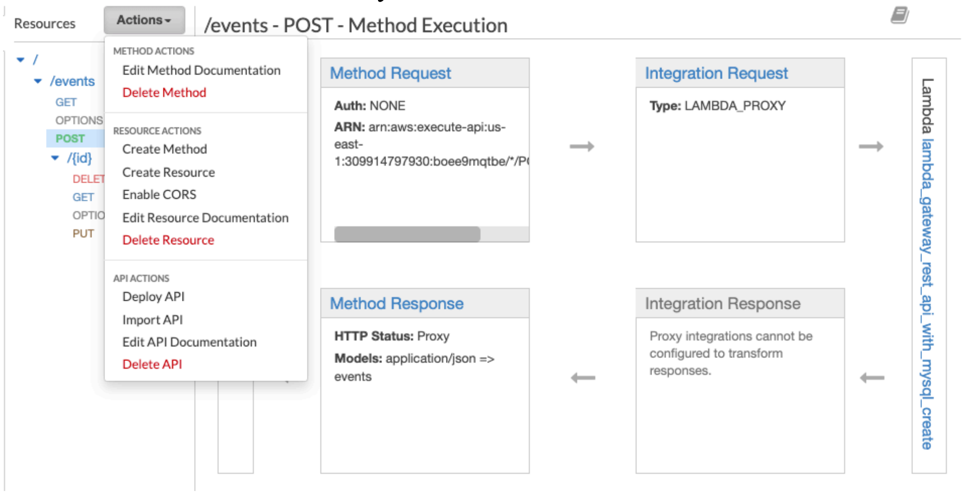
Enable API Gateway CORS ☐

* Required

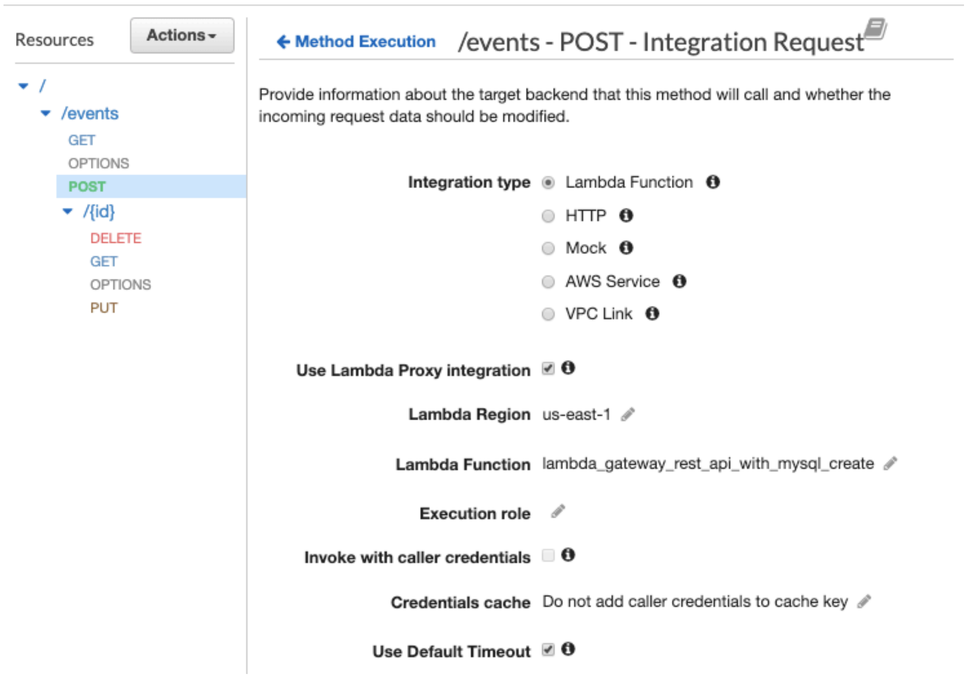
Fir. 15

Create POST Method (REST API end point)

- Click Actions, Select Create Method
- Select POST Method
- Select Integration type AWS Lambda
- Click Use Lambda Proxy Integration
  (Configuring API Gateway Request with AWS Lambda and send back the response, by using mapping template and lambda proxy integration)
- Select create Lambda function you initiate in the AWS Lambda



Auth: NONE
ARN: arn:aws:execute-api:us-east-1:309914797930:boee9mqtbe/*/P
Type: LAMBDA_PROXY
HTTP Status: Proxy
Models: application/json => events
Proxy integrations cannot be configured to transform responses.



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 us-east-1

Lambda Function lambda_gateway_rest_api_with_mysql_create

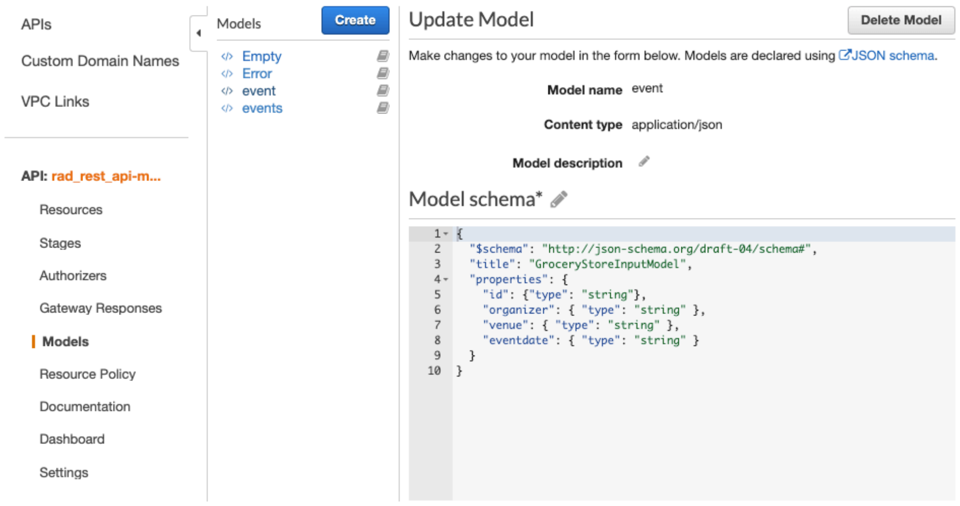
Execution role

Invoke with caller credentials ☐

Credentials cache Do not add caller credentials to cache key

Use Default Timeout ✓

Create event Model as mapping template For Request Body



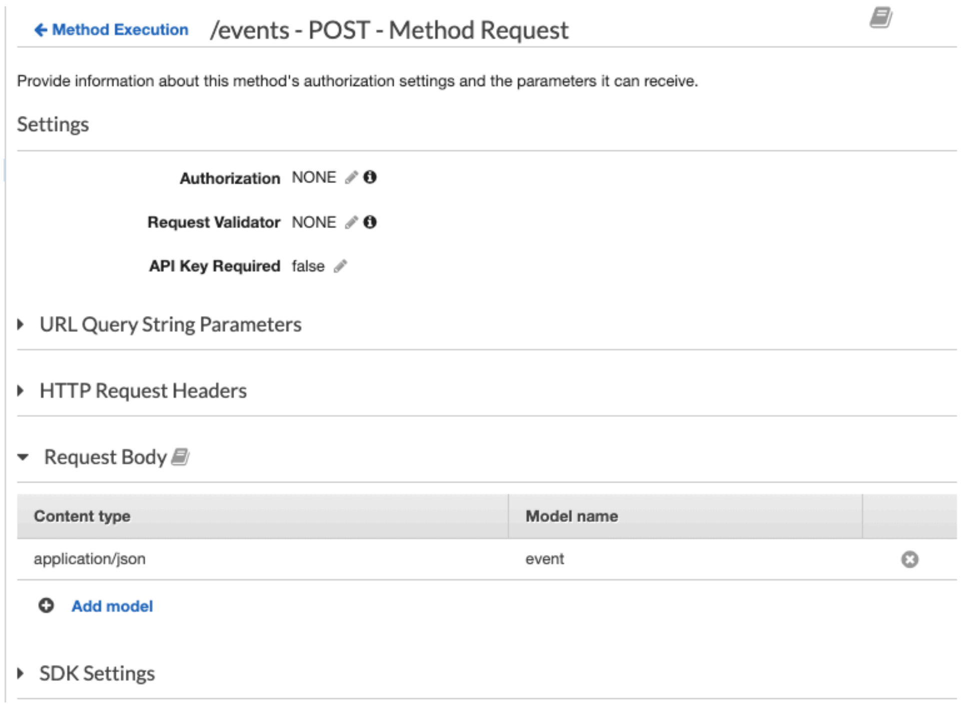
Make changes to your model in the form below. Models are declared using JSON schema.

Model name	event
Content type	application/json
Model description	

Model schema*

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "GroceryStoreInputModel",
  "properties": {
    "id": {"type": "string"},
    "organizer": { "type": "string" },
    "venue": { "type": "string" },
    "eventdate": { "type": "string" }
  }
}
```

Importing event Model into POST-Method Request



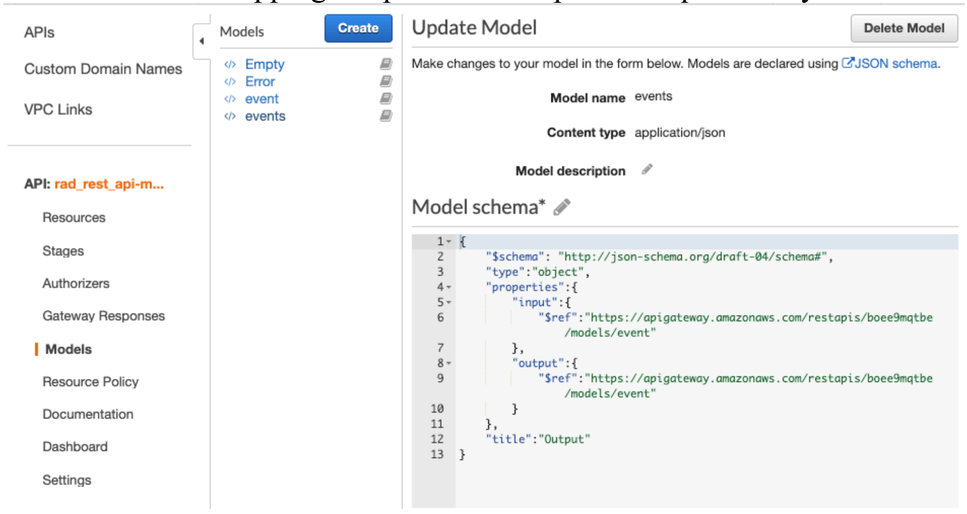
Provide information about this method's authorization settings and the parameters it can receive.

Settings

Authorization	NONE
Request Validator	NONE
API Key Required	false

Content type	Model name
application/json	event

Create events Model as mapping template For Response Request Body



Update Model

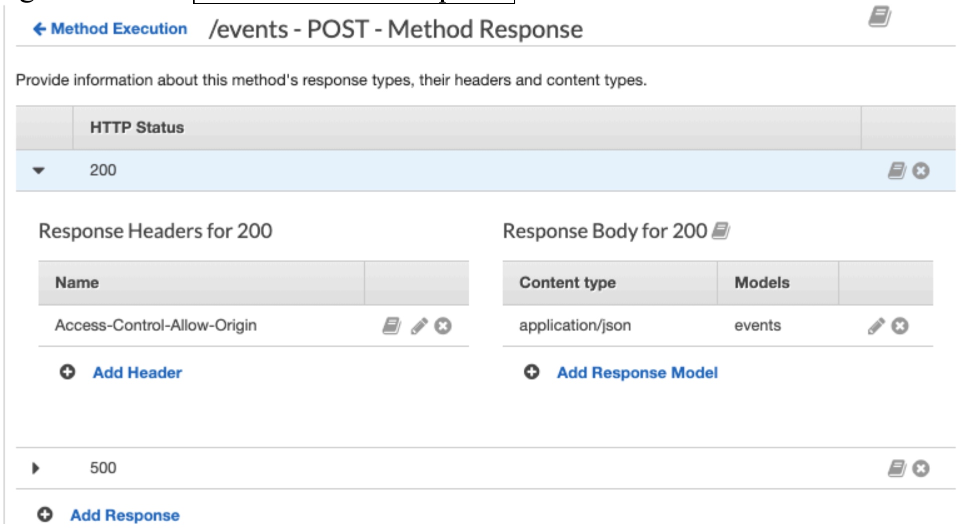
Make changes to your model in the form below. Models are declared using JSON schema.

Model name	events
Content type	application/json
Model description	

Model schema*

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type":"object",
    "properties":{
        "input":{
            "$ref":"https://apigateway.amazonaws.com/restapis/boee9mqtbe
                /models/event"
        },
        "output":{
            "$ref":"https://apigateway.amazonaws.com/restapis/boee9mqtbe
                /models/event"
        }
    },
    "title":"Output"
}
```

Importing Model into POST-Method Response



/events - POST - Method Response

Provide information about this method's response types, their headers and content types.

HTTP Status
200

Response Headers for 200

Name
Access-Control-Allow-Origin

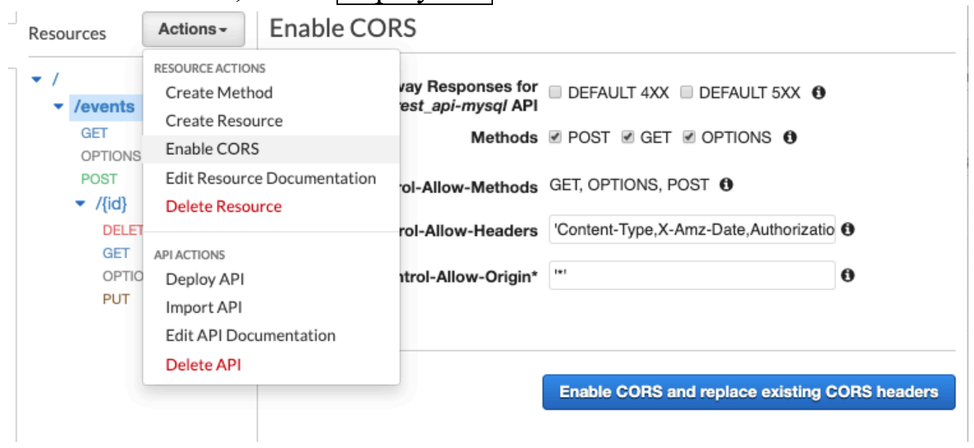
Response Body for 200

Content type	Models
application/json	events

HTTP Status
500

- Following the fig. to create the rest of the routing path

- Click Action, Select Enable CORS
- Click Enable CORS and replace existing CORS headers
- After enable CORS, Select Deploy API



vay Responses for est_api-mysql API ☐ DEFAULT 4XX ☐ DEFAULT 5XX

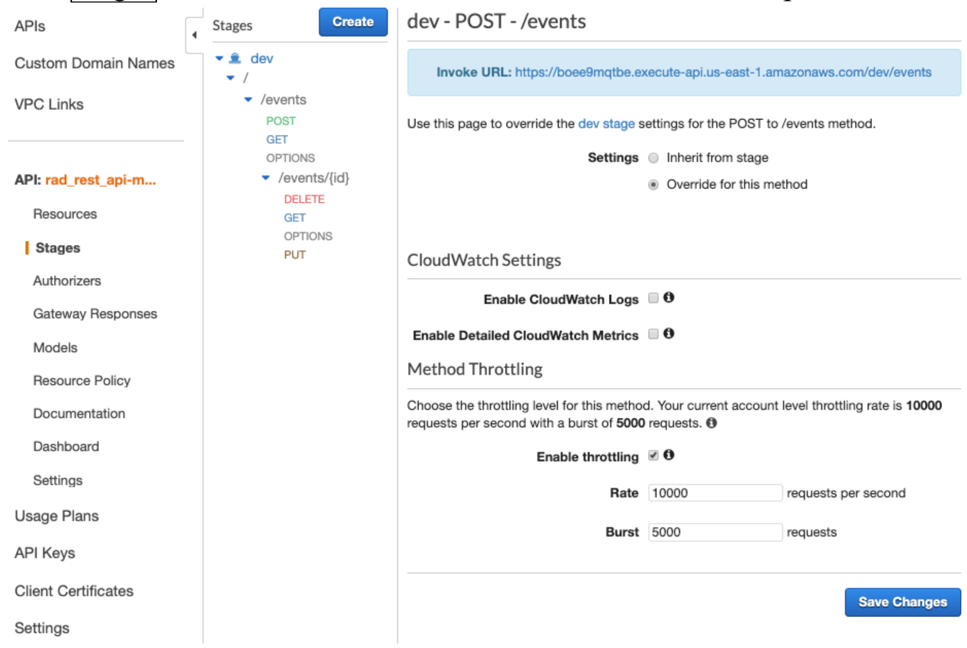
Methods ✓ POST ✓ GET ✓ OPTIONS

ol-Allow-Methods GET, OPTIONS, POST

rol-Allow-Headers 'Content-Type,X-Amz-Date,Authorizatio

ntrol-Allow-Origin* '*'

- Select Stages at the left side bar, Click the Invoke URL as endpoint



dev - POST - /events

Invoke URL: https://boee9mqtbe.execute-api.us-east-1.amazonaws.com/dev/events

Use this page to override the dev stage settings for the POST to /events method.

Settings ○ Inherit from stage
● Override for this method

CloudWatch Settings

Enable CloudWatch Logs ☐

Enable Detailed CloudWatch Metrics ☐

Method Throttling

Choose the throttling level for this method. Your current account level throttling rate is 10000 requests per second with a burst of 5000 requests.

Enable throttling ✓

Rate 10000 requests per second

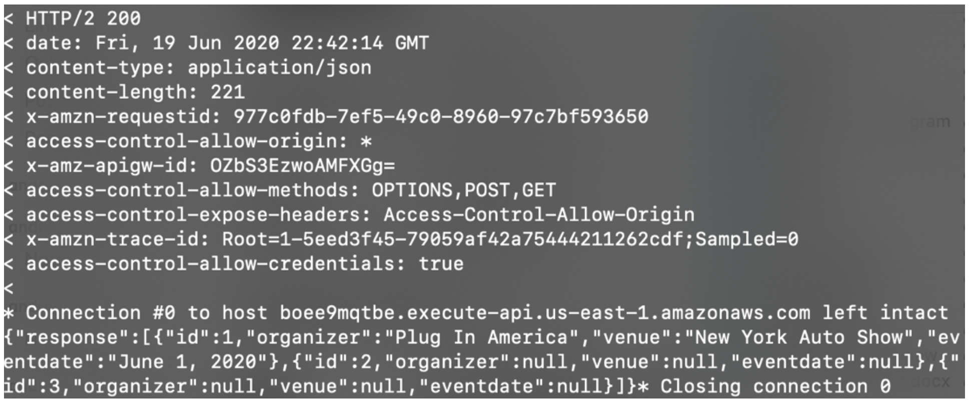
Burst 5000 requests

4. ***Curl comman-line tool for testing CORS(Cross-origin resource sharing)***

For instance,

- GET(ALL)

```
curl -X GET -d '{}' -H "Content-Type: application/json" \ -H "Origin: http://localhost:3000" --verbose https://boee9mqtbe.execute-api.us-east-1.amazonaws.com/dev/events
```



```
< HTTP/2 200
< date: Fri, 19 Jun 2020 22:42:14 GMT
< content-type: application/json
< content-length: 221
< x-amzn-requestid: 977c0fdb-7ef5-49c0-8960-97c7bf593650
< access-control-allow-origin: *
< x-amz-apigw-id: OZbS3EzwoAMFXGg=
< access-control-allow-methods: OPTIONS,POST,GET
< access-control-expose-headers: Access-Control-Allow-Origin
< x-amzn-trace-id: Root=1-5eed3f45-79059af42a75444211262cdf;Sampled=0
< access-control-allow-credentials: true
<
* Connection #0 to host boee9mqtbe.execute-api.us-east-1.amazonaws.com left intact
{"response":[{"id":1,"organizer":"Plug In America","venue":"New York Auto Show","ev
entdate":"June 1, 2020"},{"id":2,"organizer":null,"venue":null,"eventdate":null},{"
id":3,"organizer":null,"venue":null,"eventdate":null}]}* Closing connection 0
```