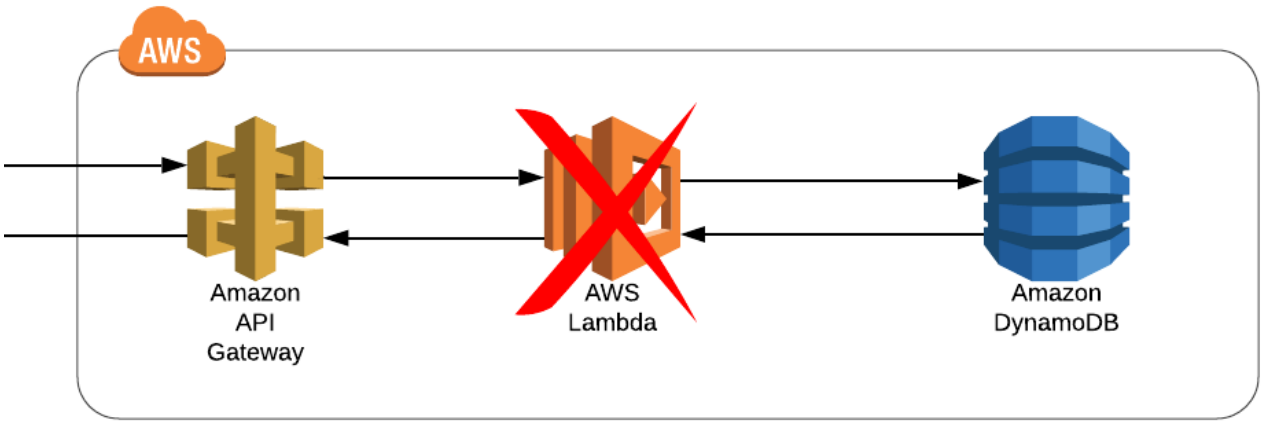
Published in BRLink · Follow

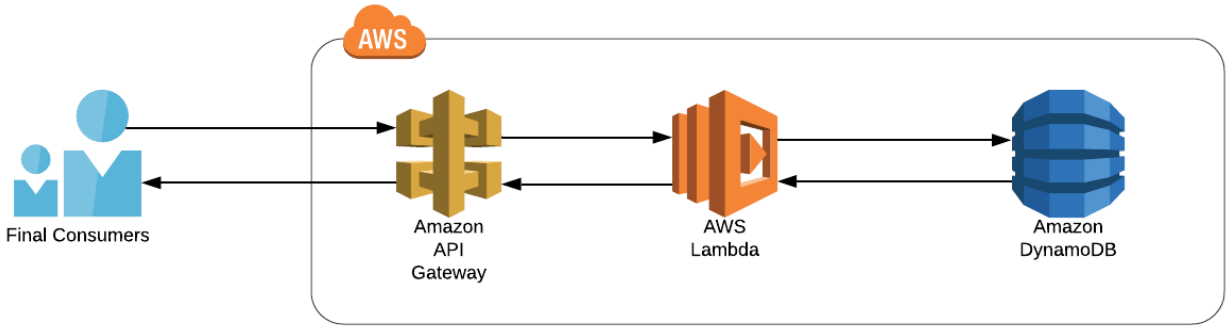
Rafael Campana · Follow
Aug 17, 2019 · 4 min read



Rest Api with ApiGateway and DynamoDb

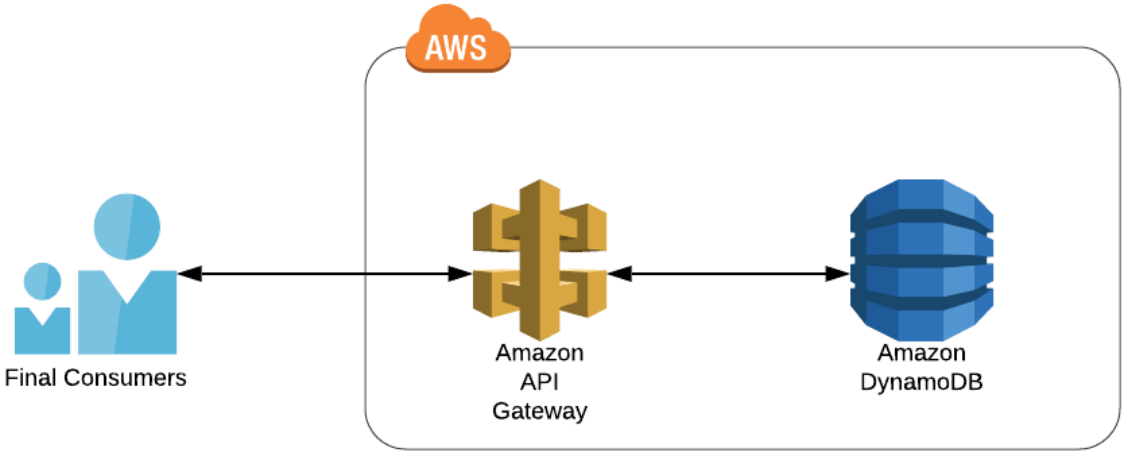
The common approach of software development is use a middleware function to manipulate the data between api and database.

The serverless way to achieve this in AWS world is use a lambda function to do this job as following sample:



Sometimes (a lot of times to be honest) we just use this to as proxy to database. What about remove this useless code and keep the things simple and clean as a possible?

With a ApiGateway using mapping templates you can implement a restfull api, with almost zero line codes and delivery a good solution in a record time with lower cost possible . The schema you can see bellow:



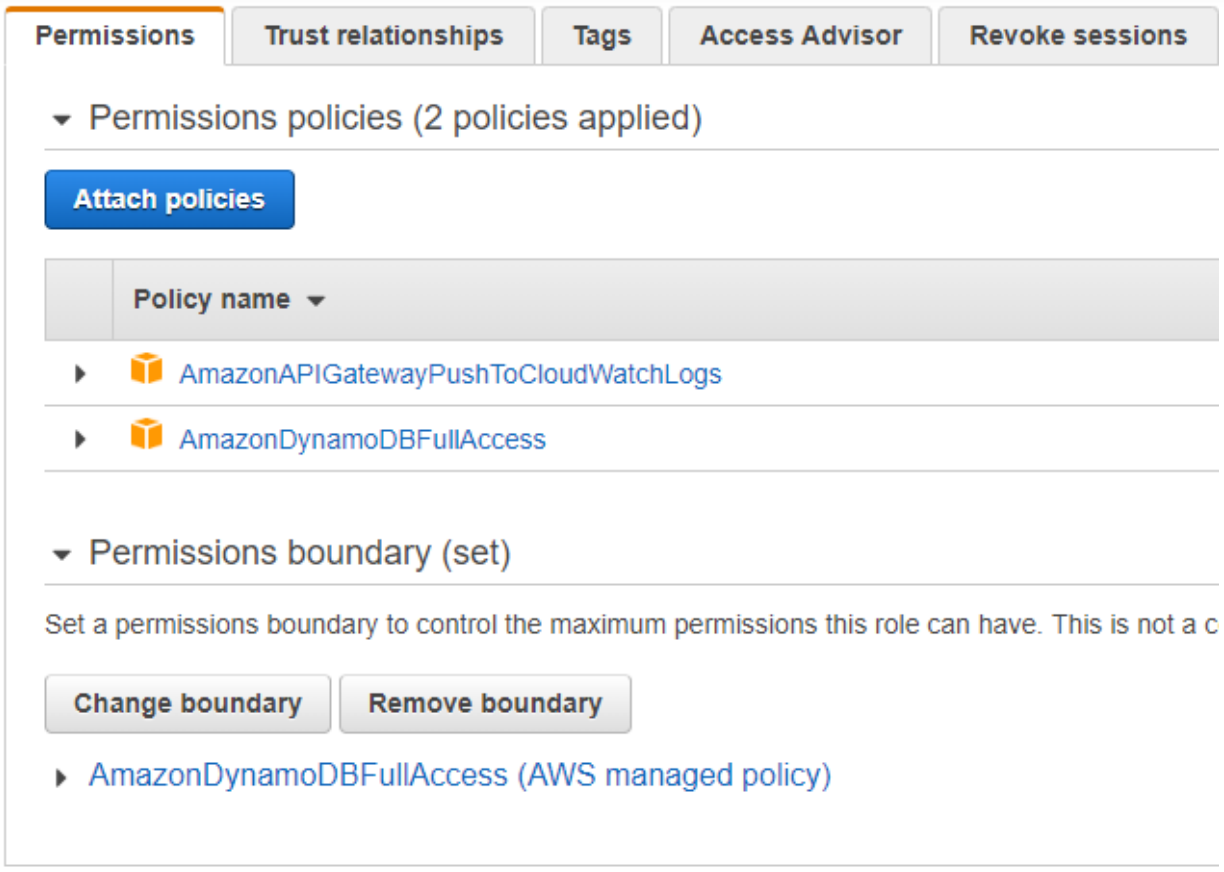
Once you convinced , let's to this

1- Create a DynamoDB table

To this sample, I've create a super simple table just with **id, name and userName**. I won't detail how to do this, but you can see the steps here: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.CreateTables.html

2- Create a Role

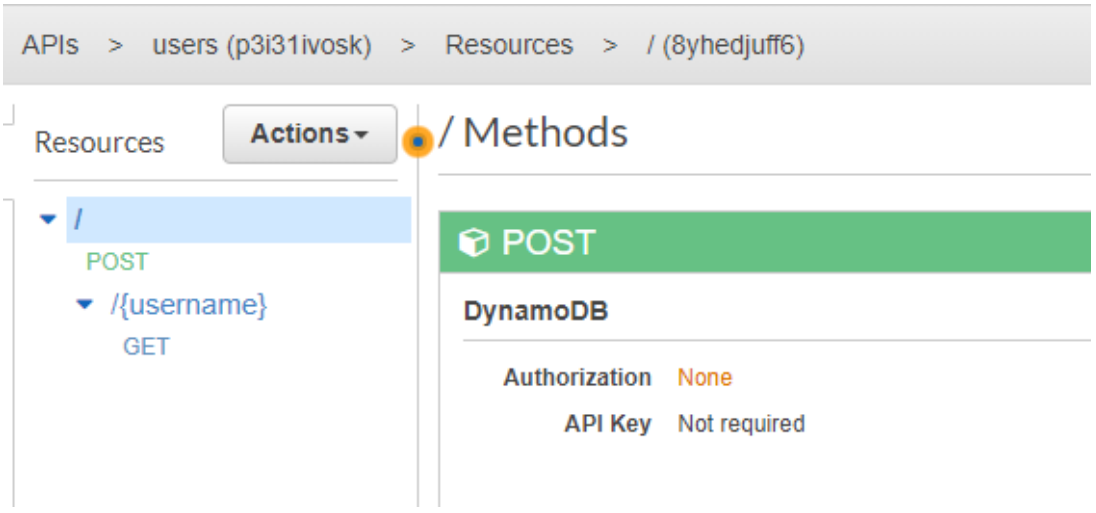
Your role should have access to your table, for this sample I've created a full dynamo db, sure in a real world is a good practice create a fine granulated access for each api,table and method



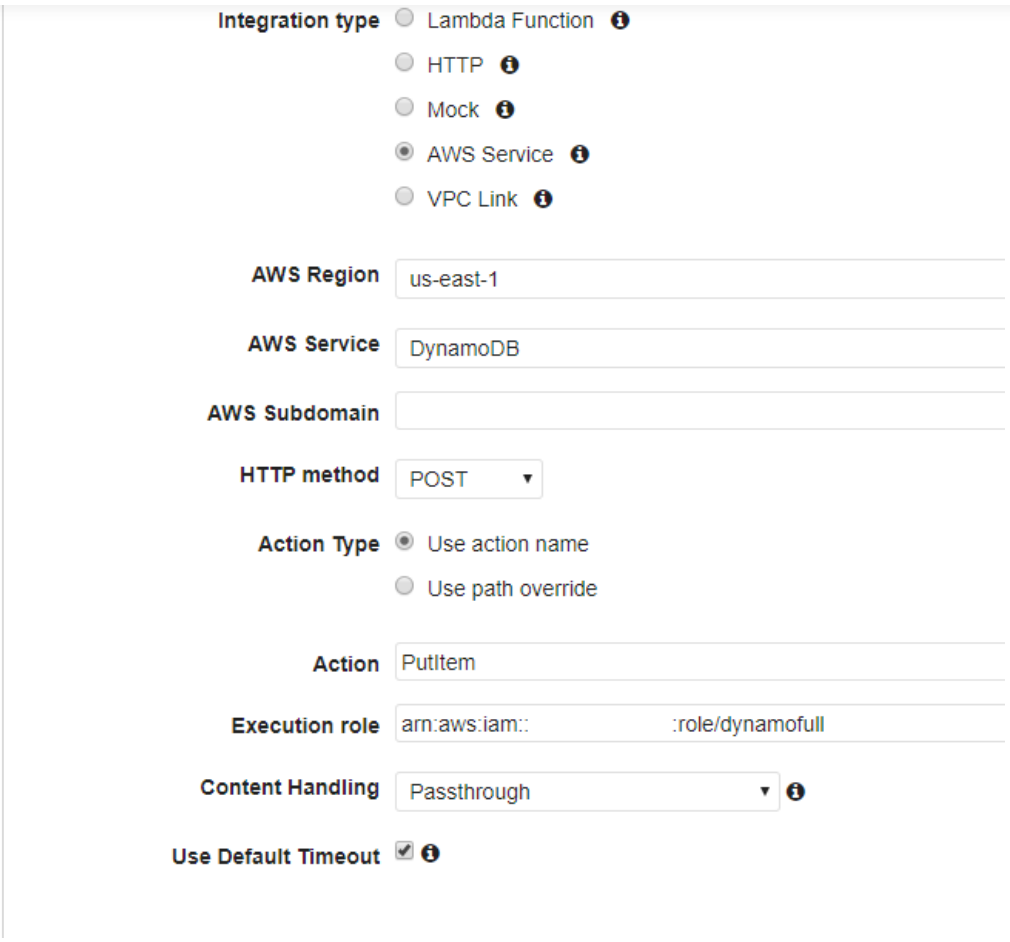
Polices of the Role

3- Create Your Api Gateway

Create a API Gateway with the following structure :



On method post, you should point the Dynamo db and indicate your role created in the

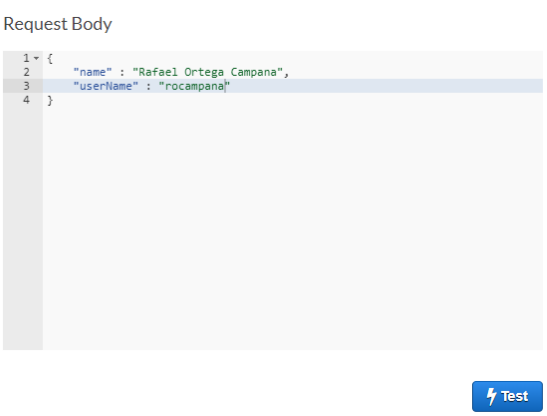


Once created, you should create a mapping template to convert a body content from the post rest request to a PutItem DynamoDb request, in the **Integration Request > Mapping Templates** add a new item with content type application/json with the following content:

```
{
    "TableName": "users",
    "Item": {
     "uid": {
            "S": "$context.requestId"
        },
        "name": {
            "S": "$input.path('$.name')"
            },
        "userName": {
            "S": "$input.path('$.userName')"
        }
    }
}
```

with a requestId unique generated by api gateway request.

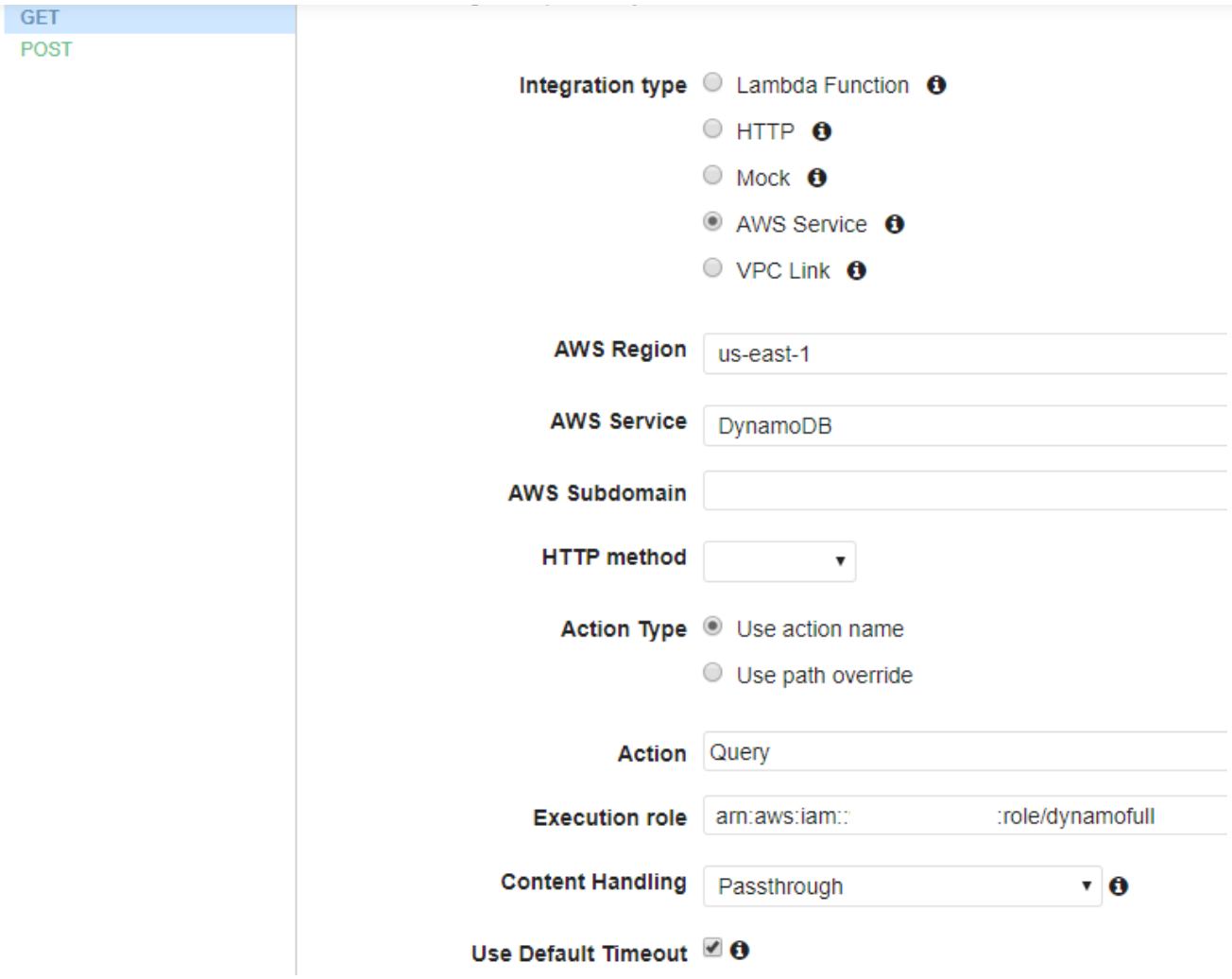
Once this done, let's do our first test and analyse the response:



As you can see, the rest call is converted to putitem invoke to dyname and the response was 200, the item has been created and you can check on dynamo db console:)

4- Create a Resource and Get Method

Create a **Resource** with resource path {username} and a get method associated. Configure the get method almost as the post, but changing the "PutItem" to "Query", once the action now needed is that query in the database.



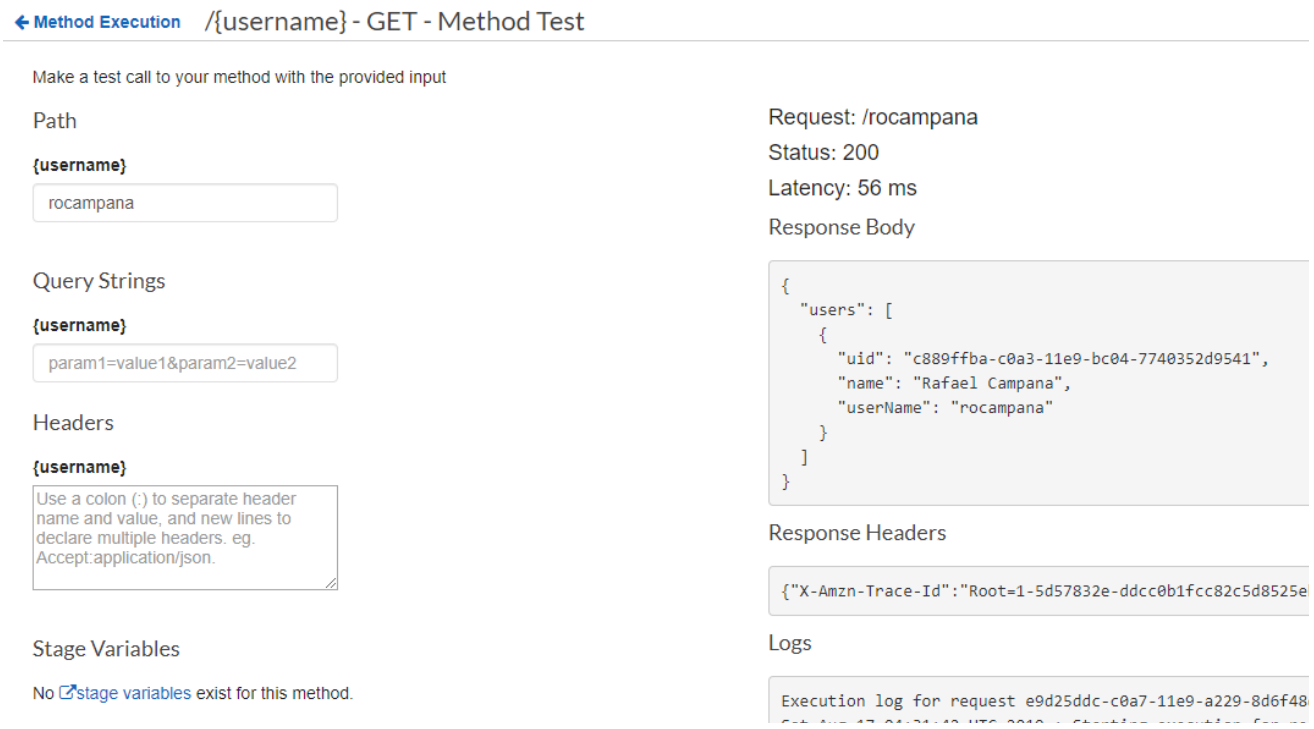
Now we need to create again the **Mapping Templates** on **Integration Request**, but now with the following translation:

```
{
    "TableName": "users",
    "IndexName": "userName-index",
    "KeyConditionExpression": "userName = :val",
    "ExpressionAttributeValues": {
        ":val": {
            "S": "$input.params('username')"
        }
    }
}
```

This maps the resource param **username** to dynamoDb with the condition “userName” and does the query.

```
#set($inputRoot = $input.path('$'))
{
    "users": [
        #foreach($elem in $inputRoot.Items) {
            "uid": "$elem.uid.S",
            "name": "$elem.name.S",
            "userName": "$elem.userName.S"
        }#if($foreach.hasNext),#end
 #end
    ]
}
```

Let`s test and analyse the response:



We have passed the username created early “rocampana” witch contains a array (with just one item) with all users with this profile registered on database.

And now we have **two methods**:

> **post** — *users/*
> **get** — *users/{username}*

Now, ensure that your api gateways resources is accessed securely and be happy