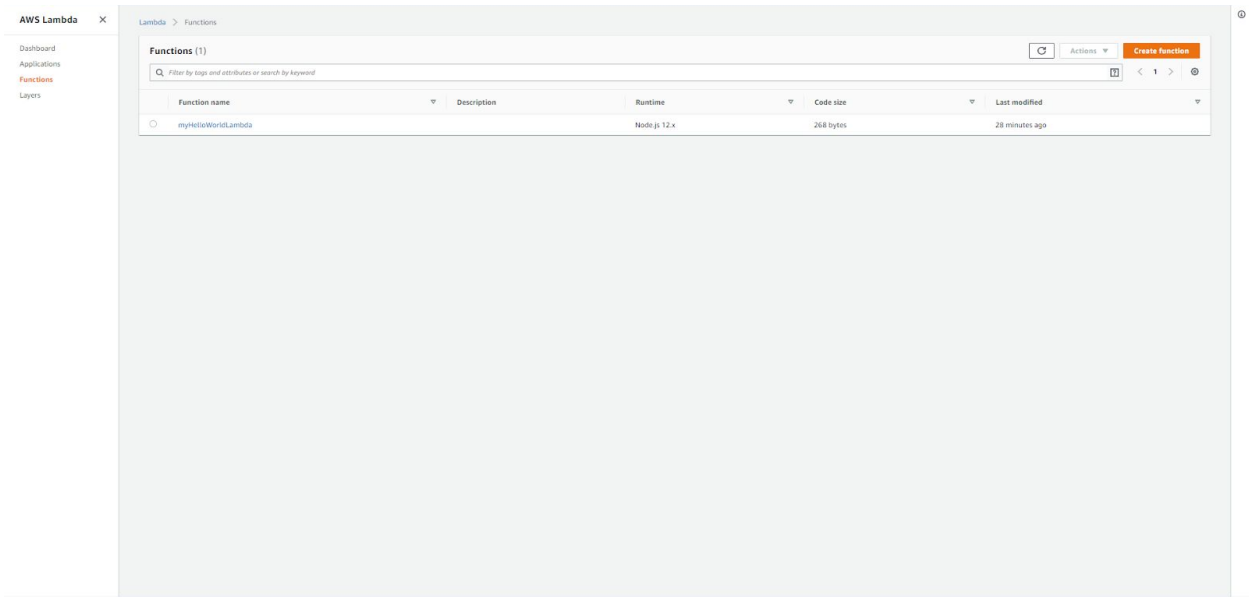
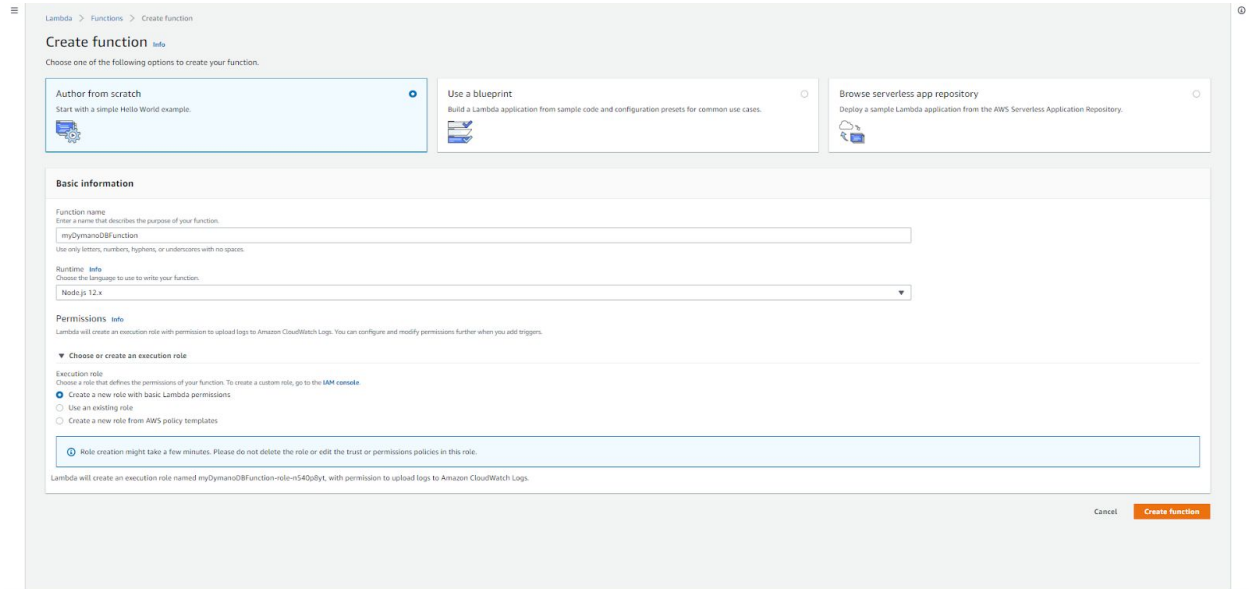
Module 6 Lab 2: DynamoDB Read

AWS Lambda



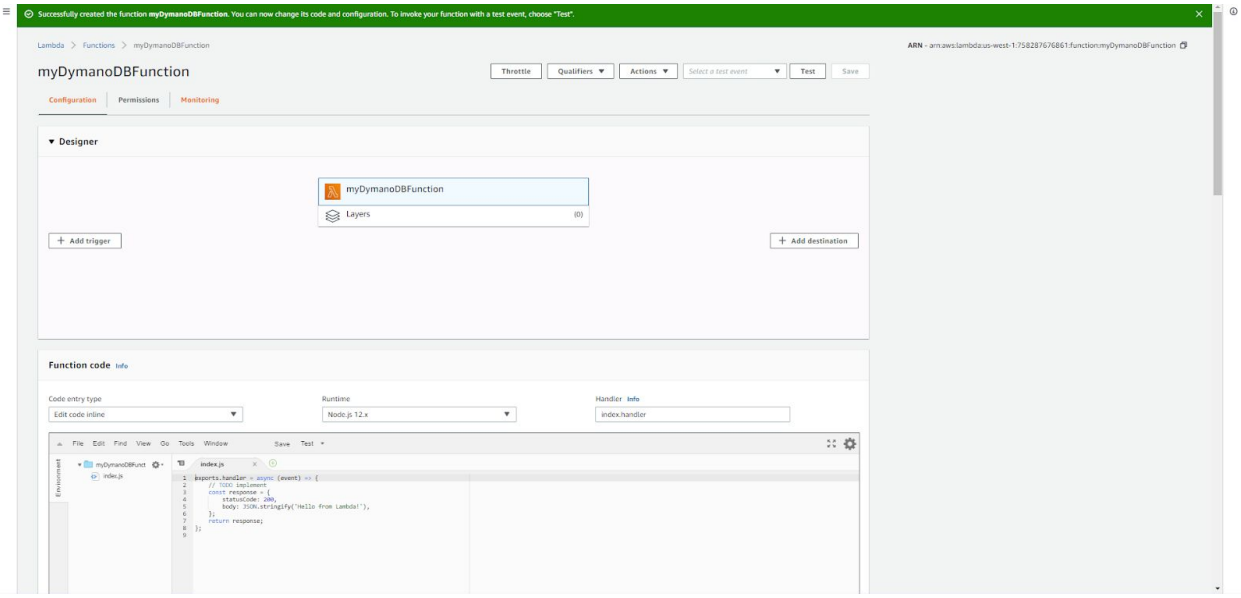
At the main menu for AWS Lambda. In this lab the following AWS services will be used: lambda, DynamoDB, and API Gateway.

Create function



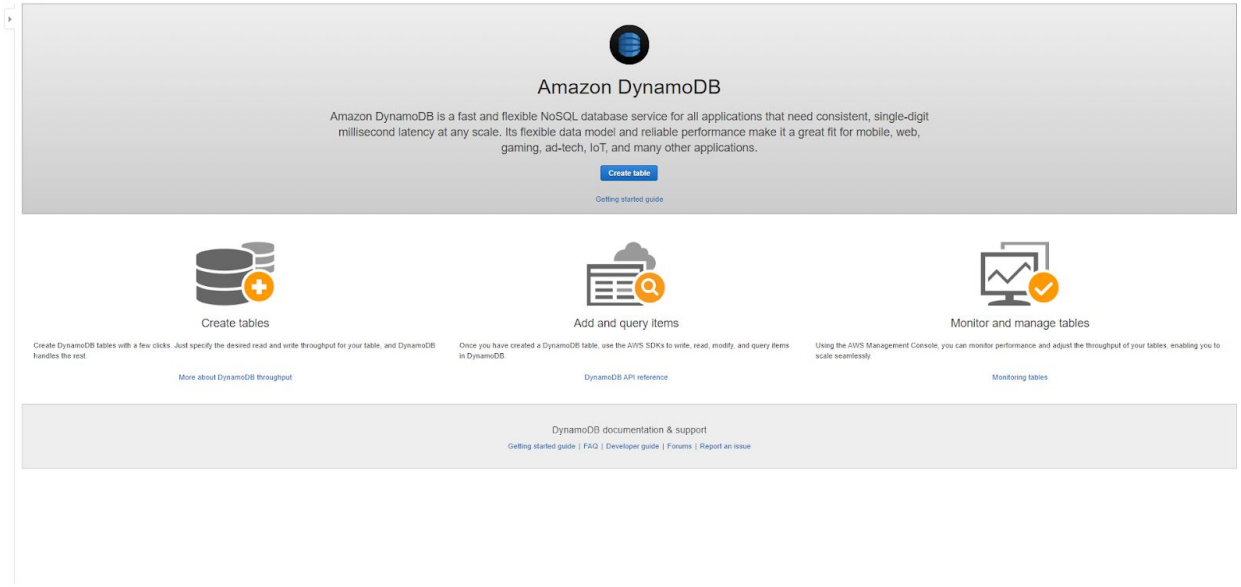
Name the new DynamoDB function. Note that basic lambda permissions have been selected but an update to the role will occur in the preceding steps.

myDymanoDBFunction



Successfully created and waiting code. Note that the handler name must match file, index.handler.

Amazon DynamoDB



A DynamoDB table creation.

Create DynamoDB table

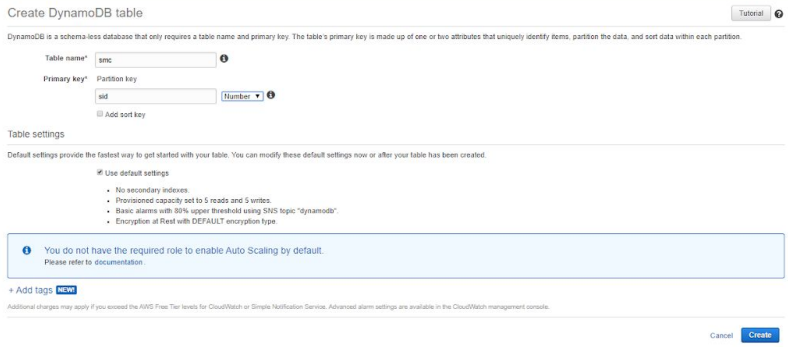
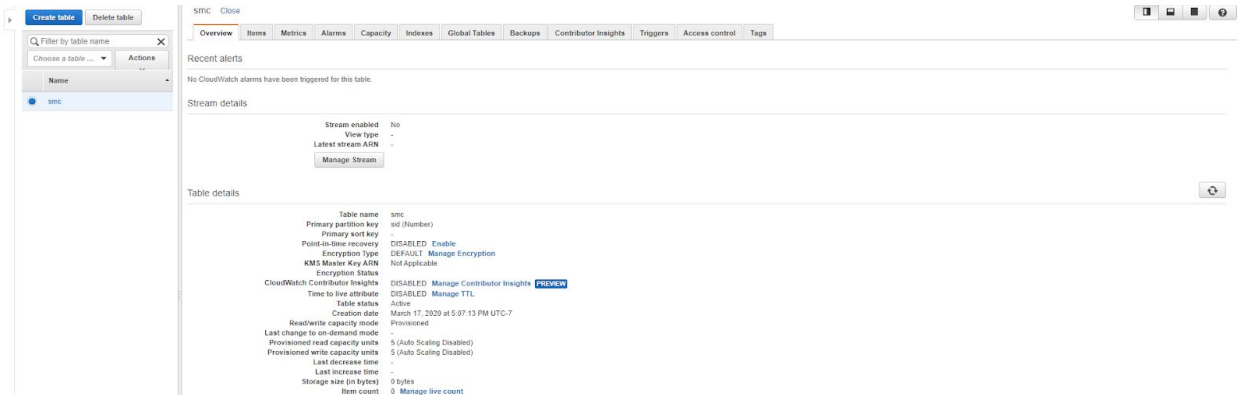


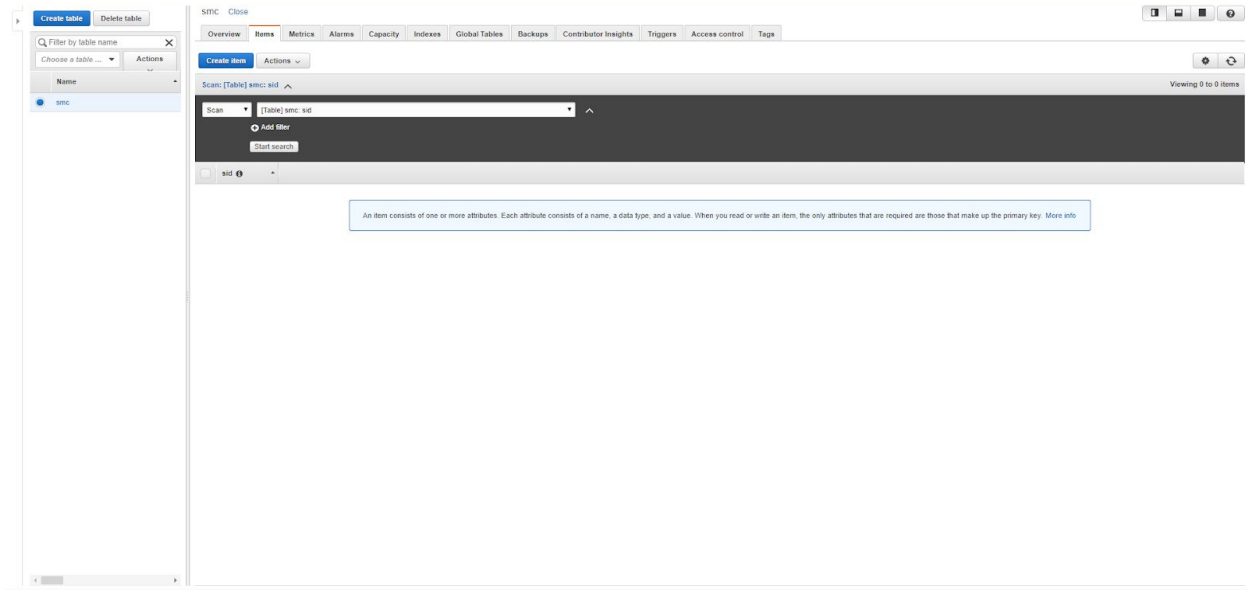
Table name, primary key and data type set

Table smc Overview



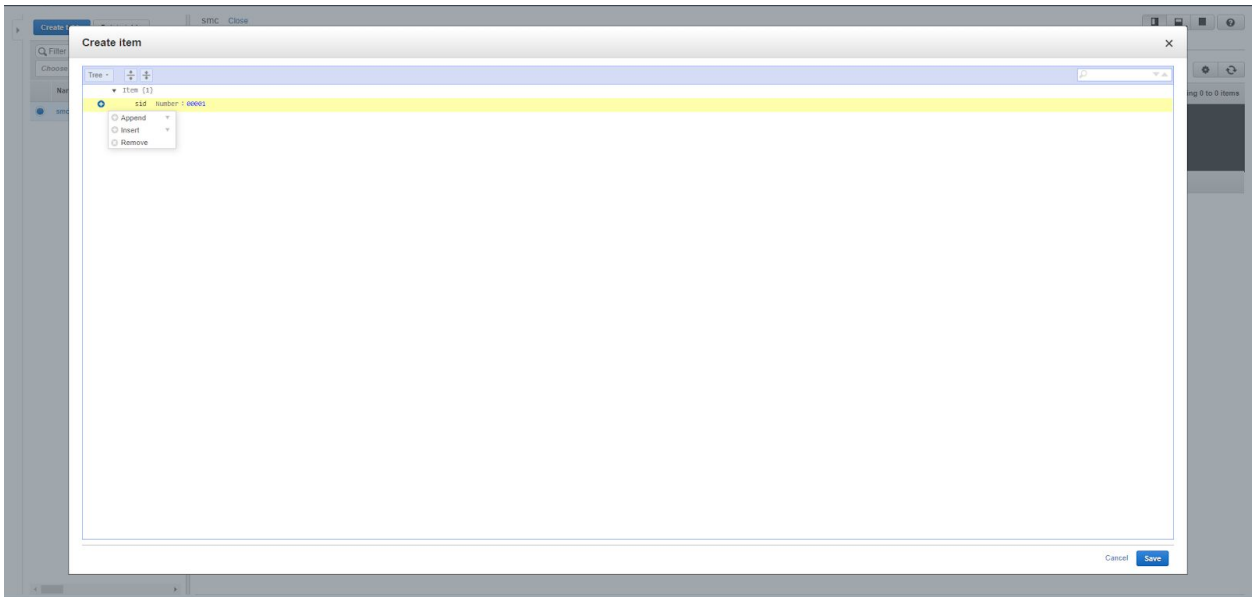
DynamoDB table has been created

SMC Items



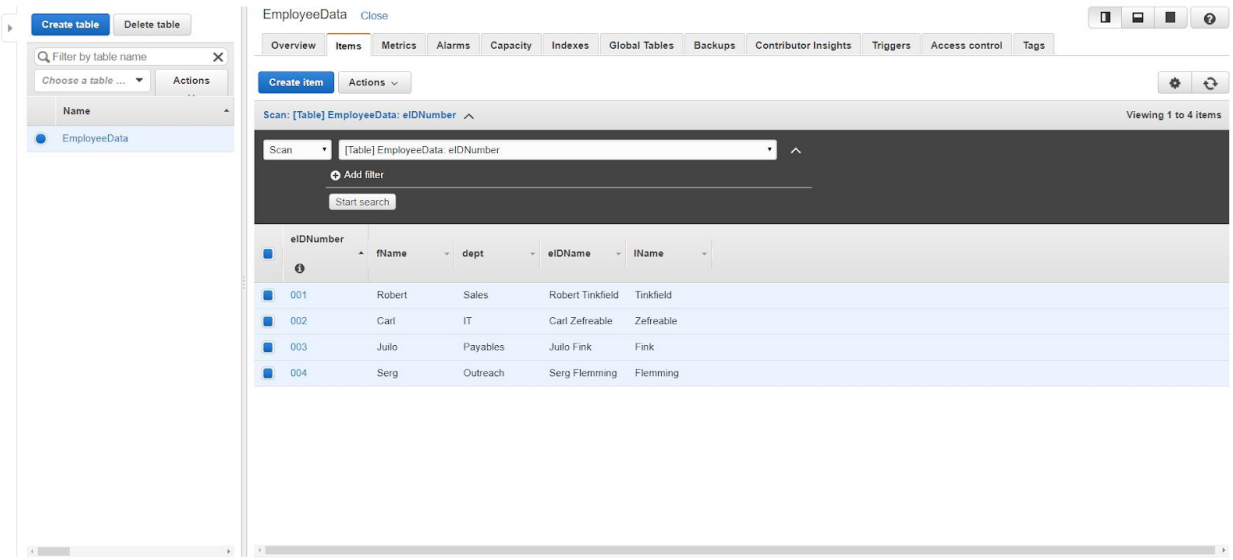
Under the tab “Items” new entries into the database can be made in TREE or JSON format.

Create item



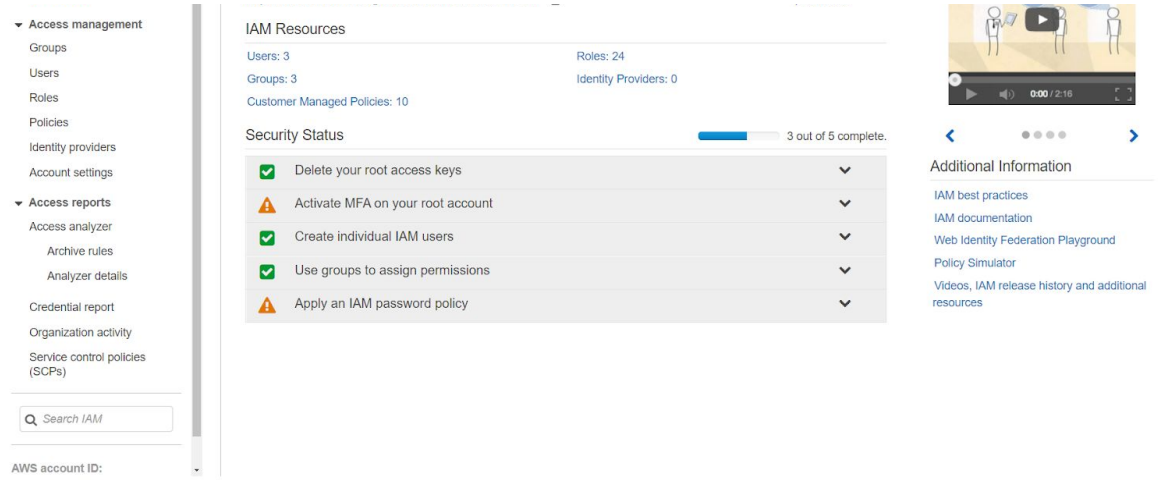
Here the item will Append to the list as a STRING datatype.

EmployeeData



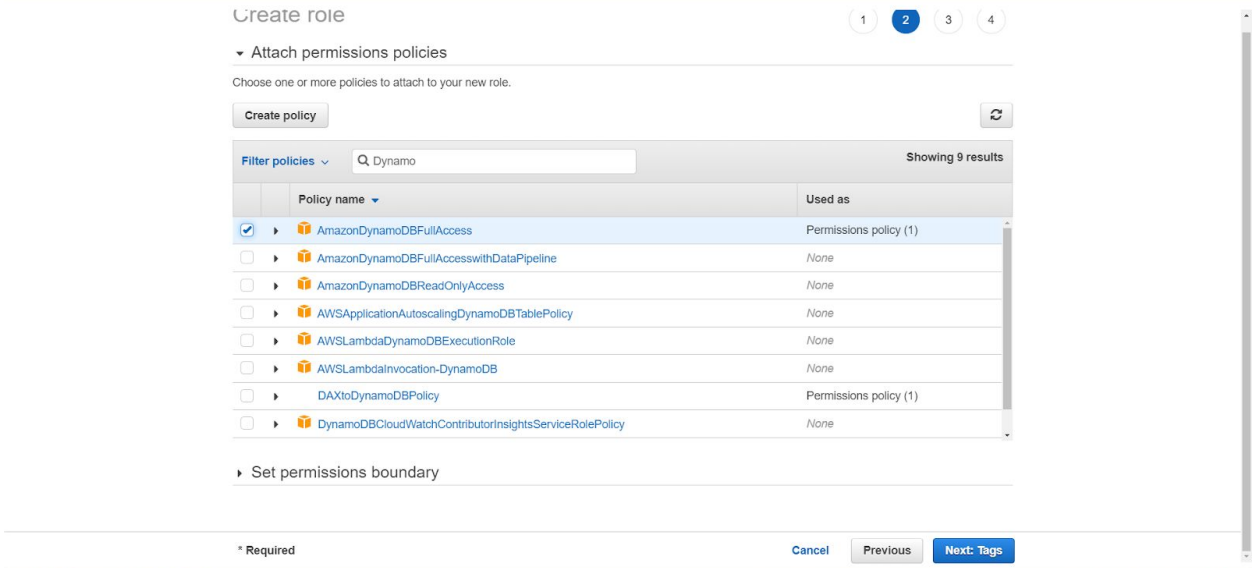
EmployeeData table created with four employees

Identity and Access Management (IAM)



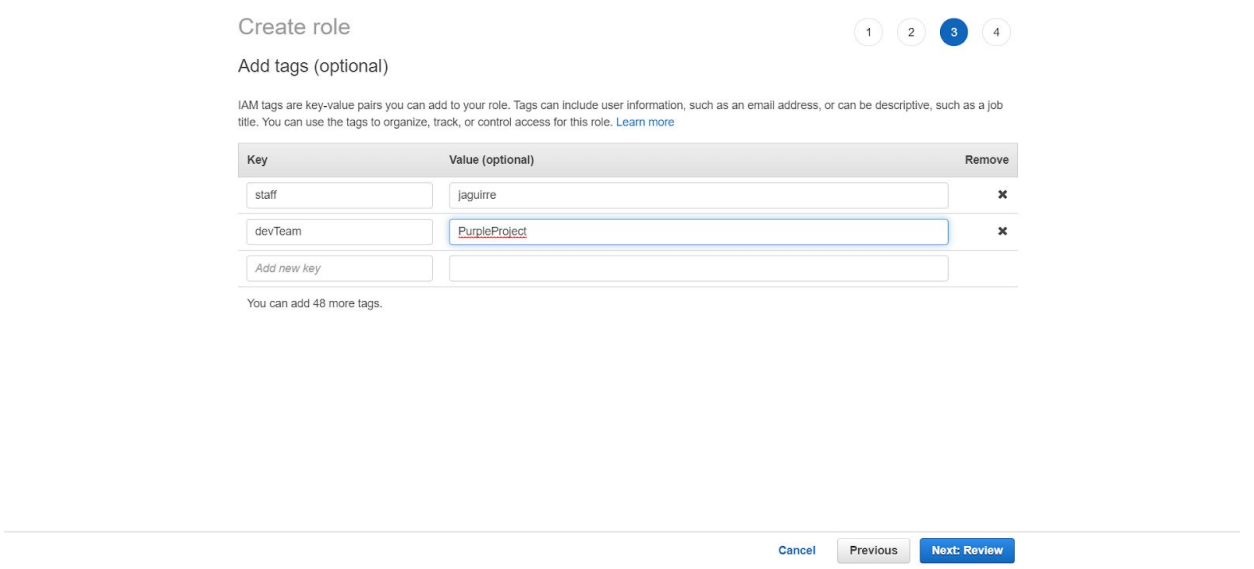
A new role will be created in IAM

Create role



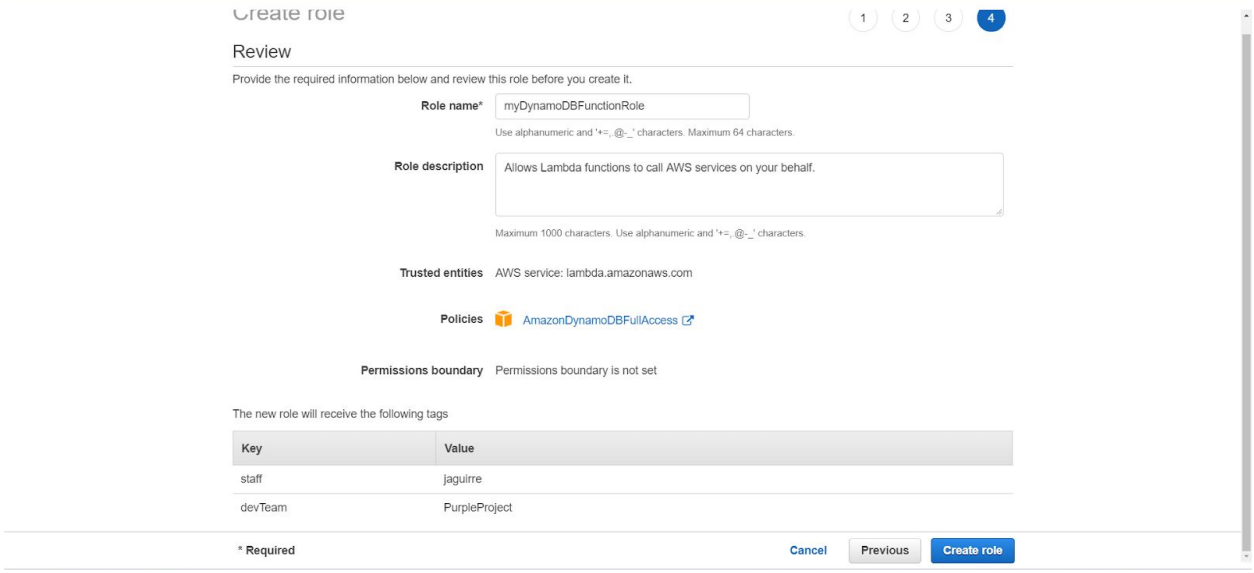
This role will not have DynamoDB access but will have Lambda access

Create role (tags)



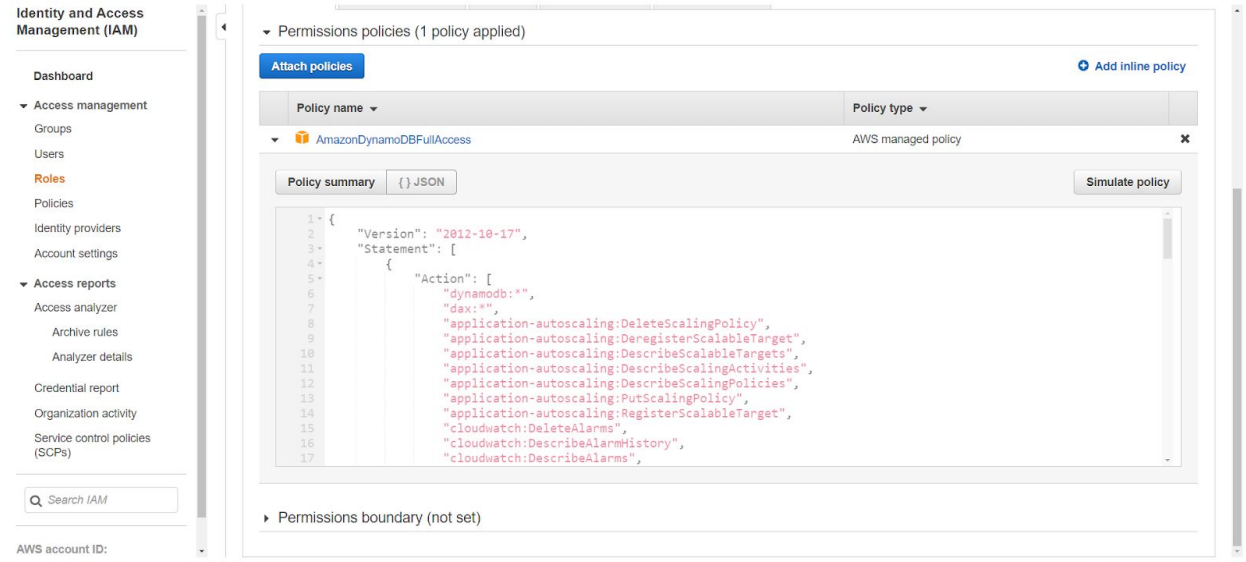
Tags created

Create role (review)



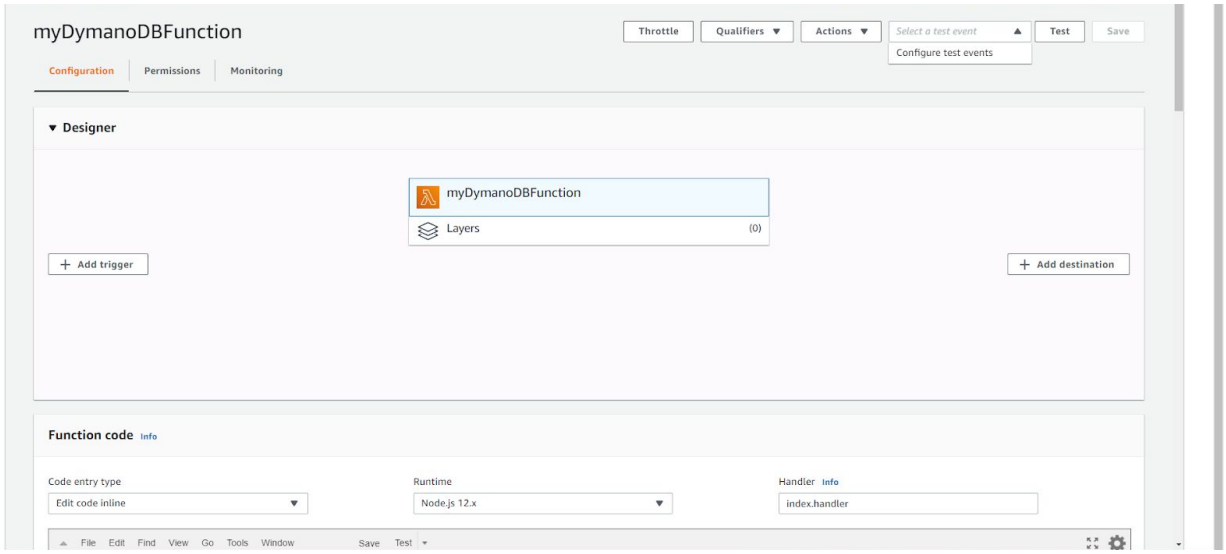
Role name includes a reference to the AWS service

Identity and Access Management (IAM)



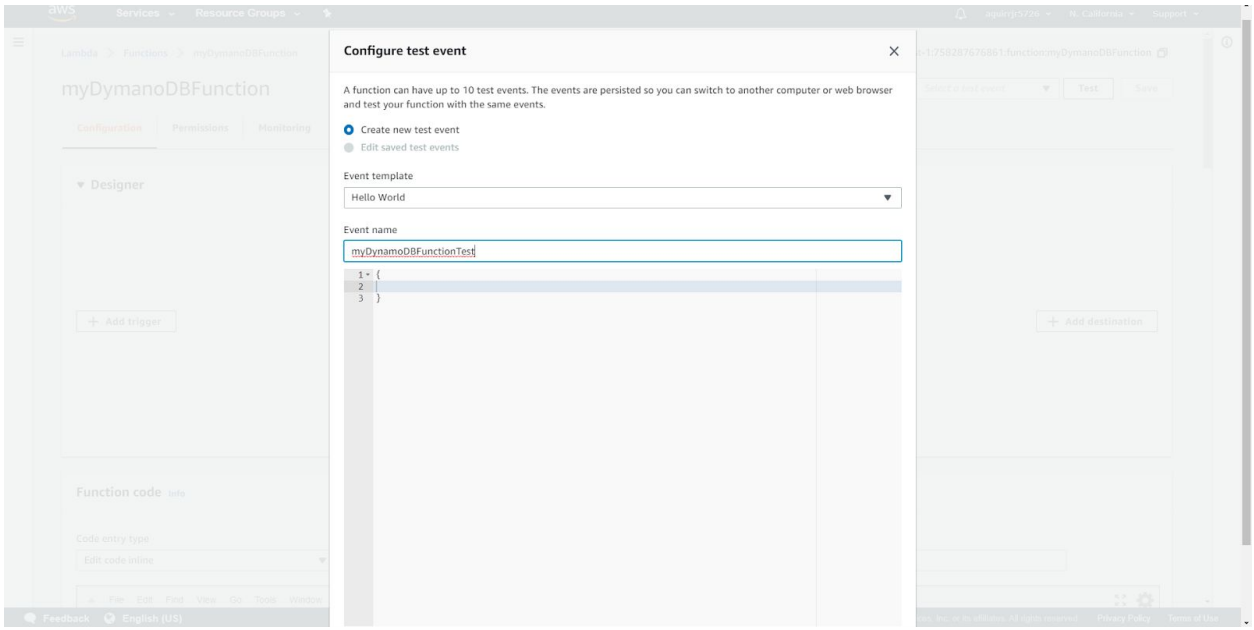
Role created with JSON policy summary.

myDymanoDBFunction



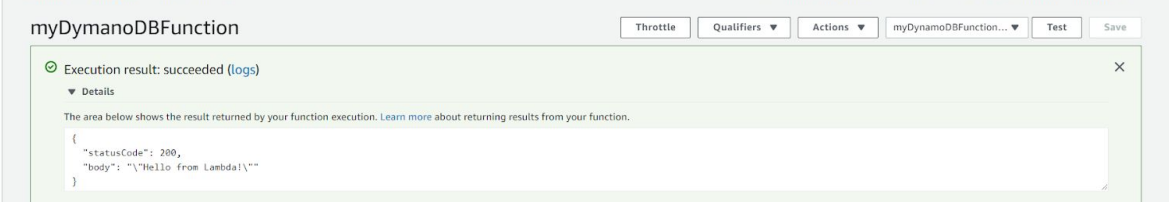
Test event for the lambda function to be used next

Configure test event



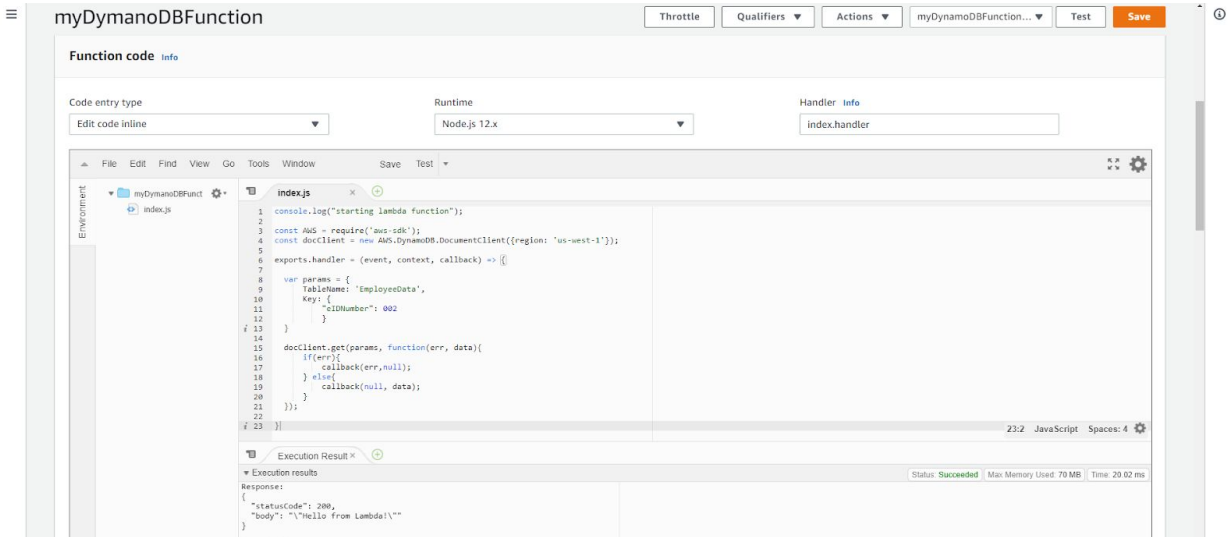
Empty to show that the code will run.

myDynamoDBFunction



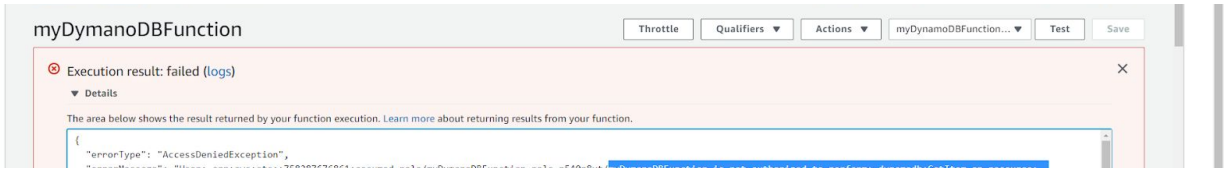
The hello world code works, but now a DynamoDB test is required.

myDynamoDBFuncion -Scan



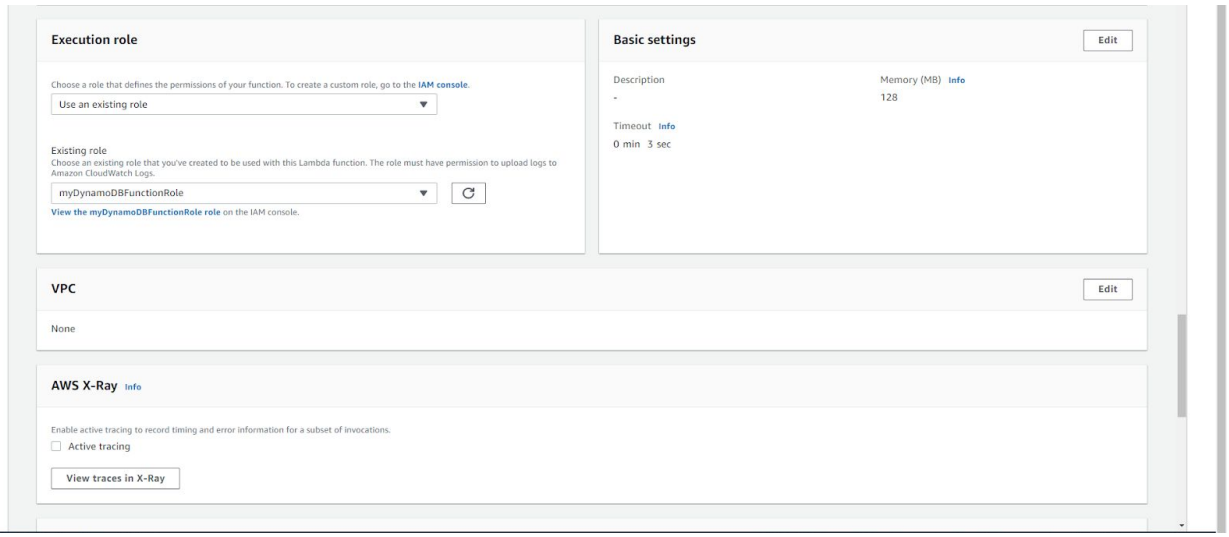
This code snippet will return EmployeeData where eIDNumber is equal to ‘002’. Saved then Tested. However the Execution role is not updated, therefore no return from the database should occur.

myDynamnoDBFunction table GETITEM fail



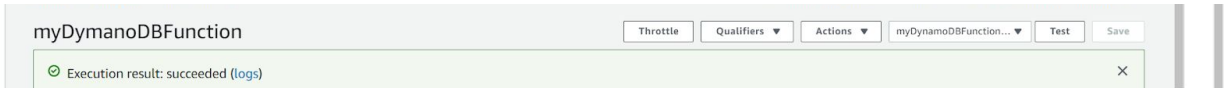
As expected, the function has no access to the table and returns an Execution failure.

myDynamoDBFunction



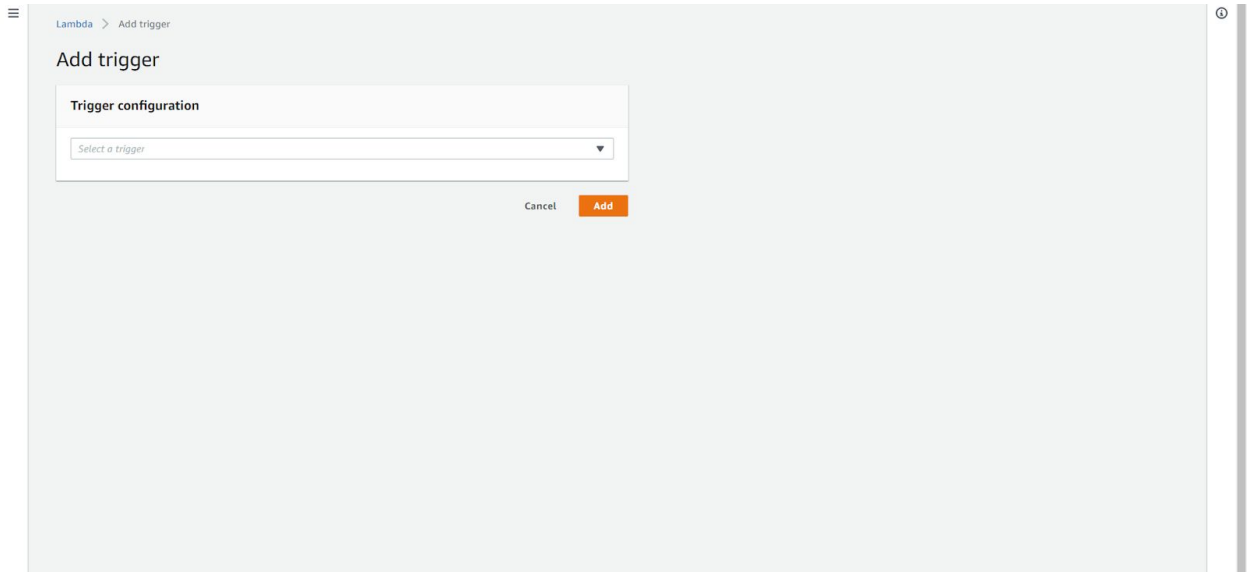
Existing role updated to role created to allow DynamoDB full access.

myDynamoDBFunction



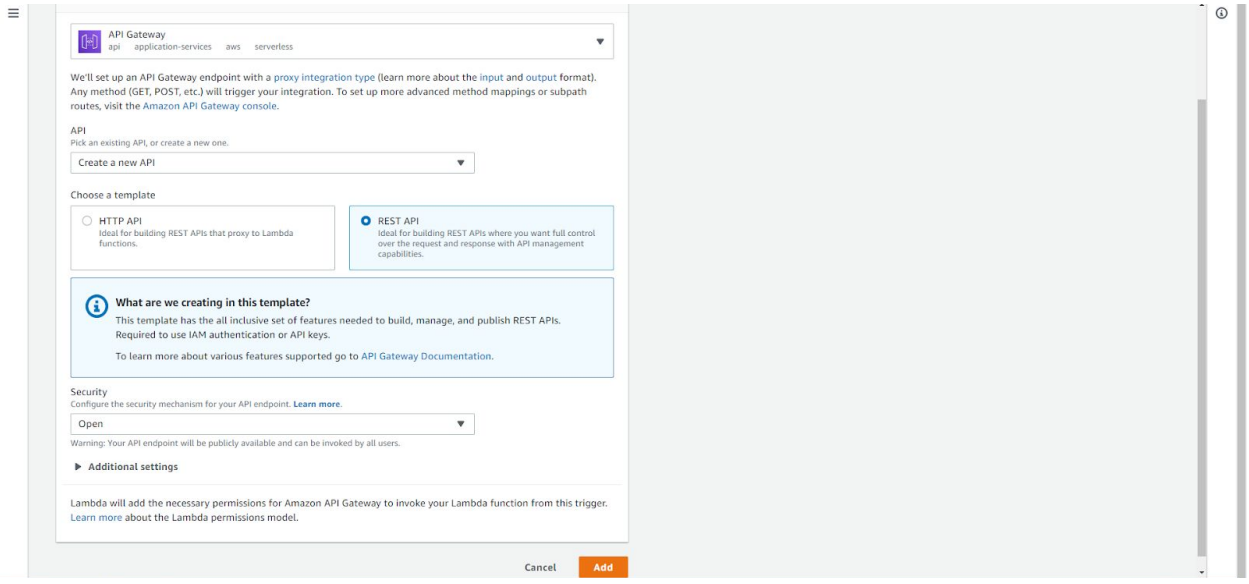
Success! We have a working code base to retrieve data from DynamoDB. Note the fields brought back.

Add trigger



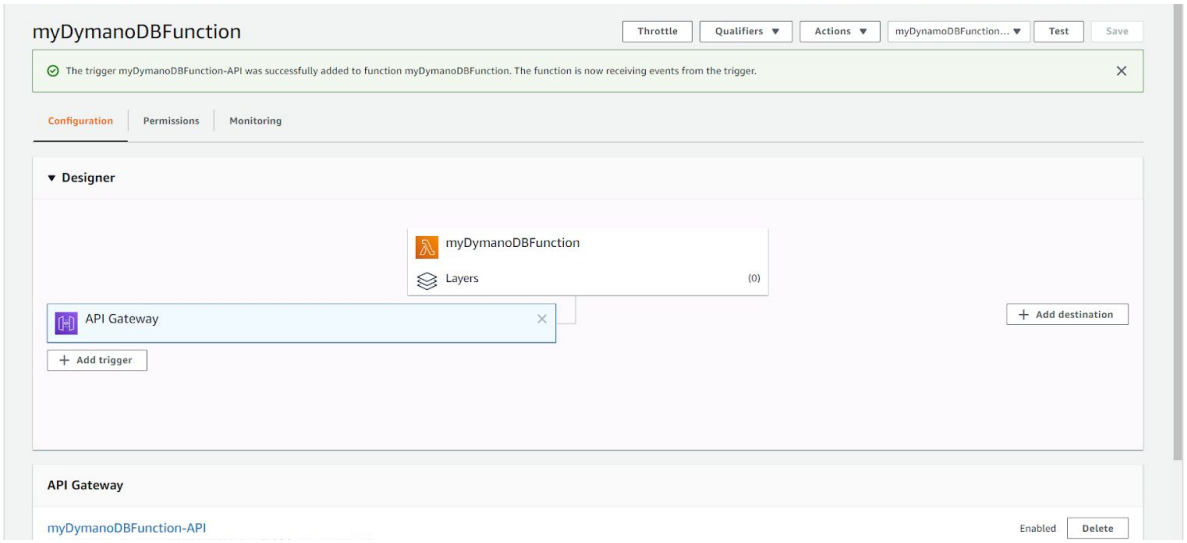
A new trigger for the lambda read function.

Add trigger



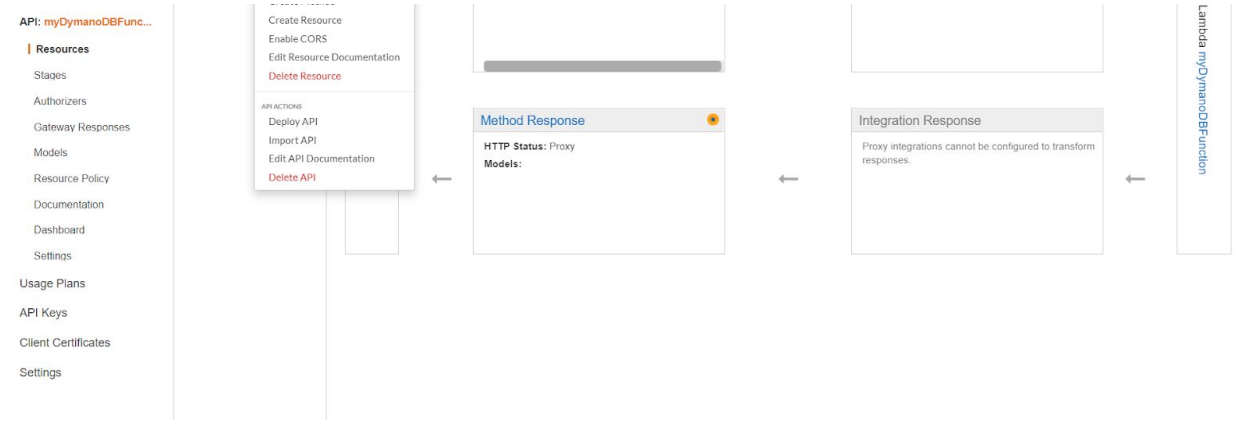
A new REST API with security set to Open. Choosing Open is NOT recommended for production use.

myDynamoDBFunction



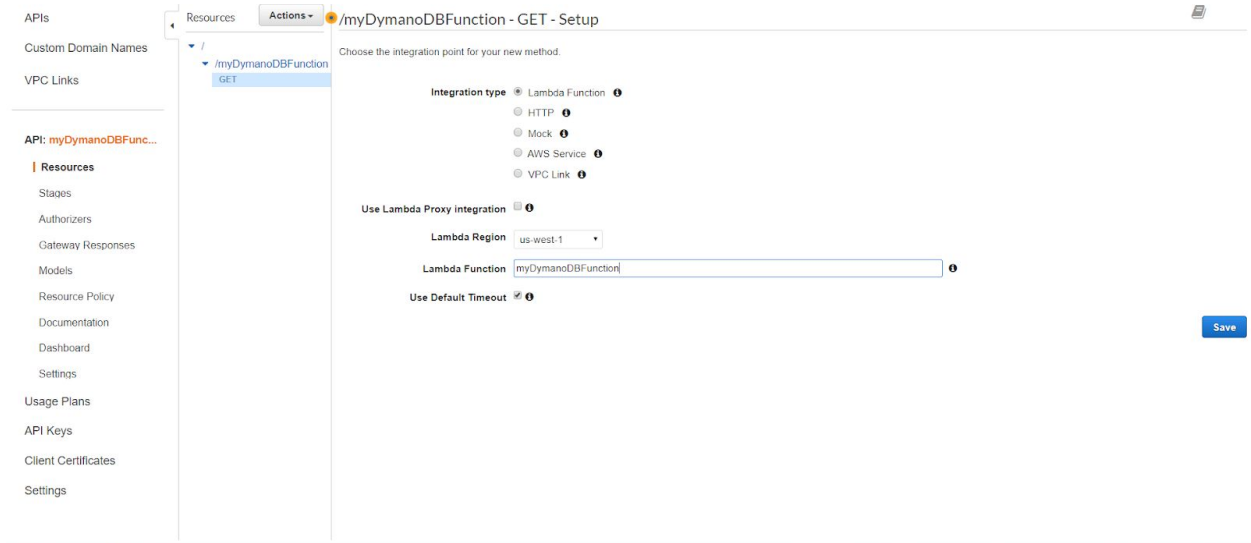
Trigger has been added successfully

.Amazon API



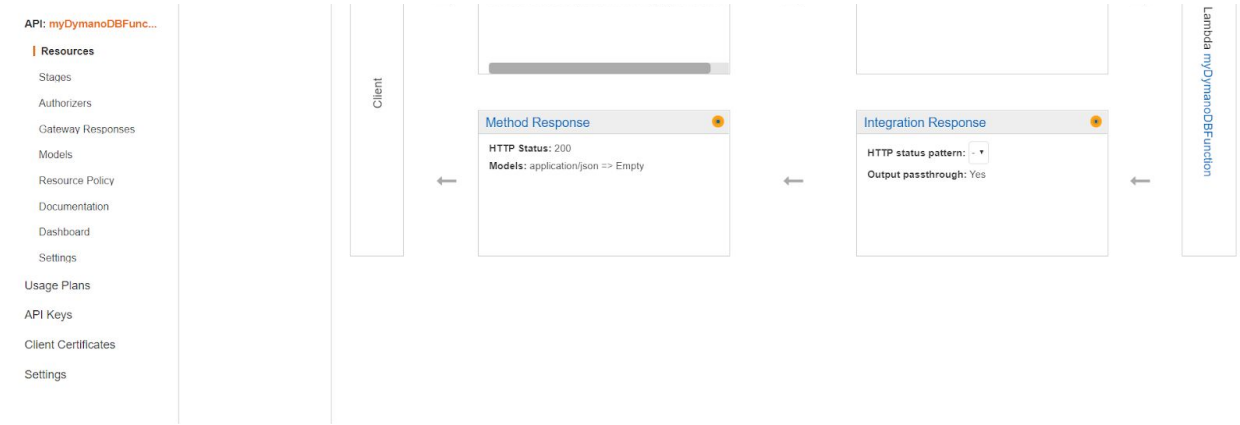
Deleting the ANY method.

Amazon API Gateway



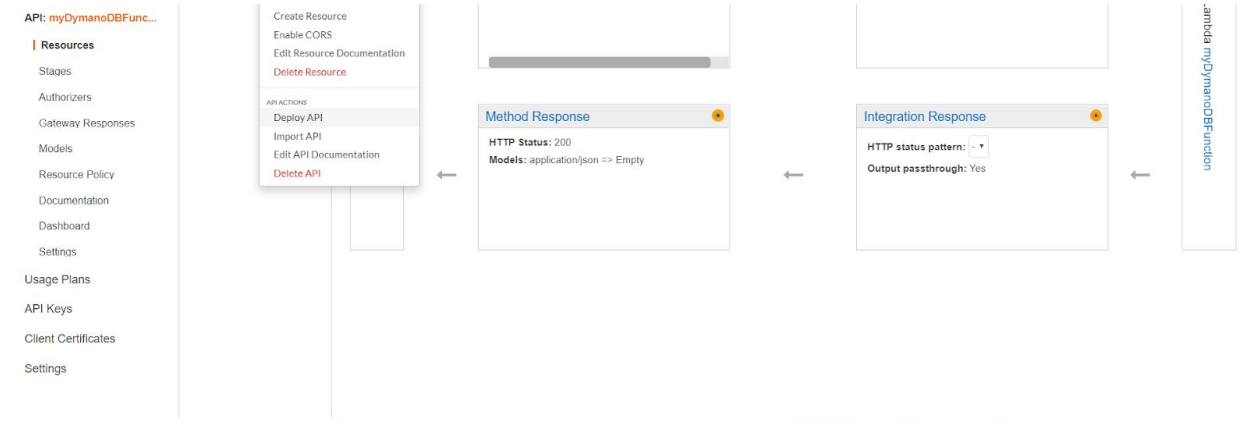
When there's an API call to this gateway, the lambda function will be called.

/myDynamoDBFunction - GET - Method Execution



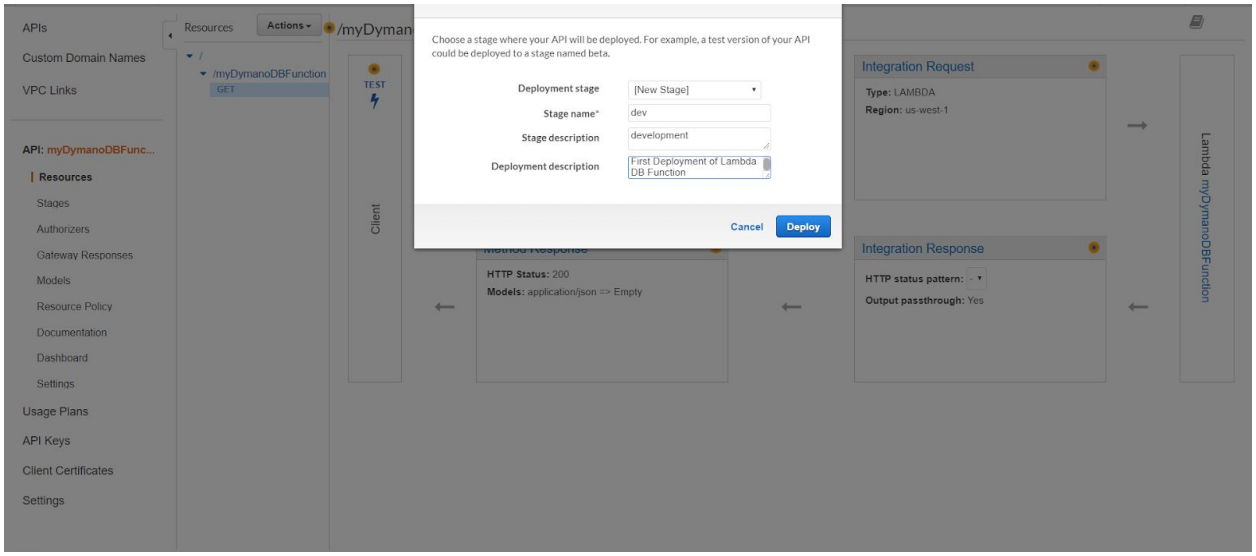
The GET Method Execution is defined.

/myDynamoDBFunction - GET - Method Execution



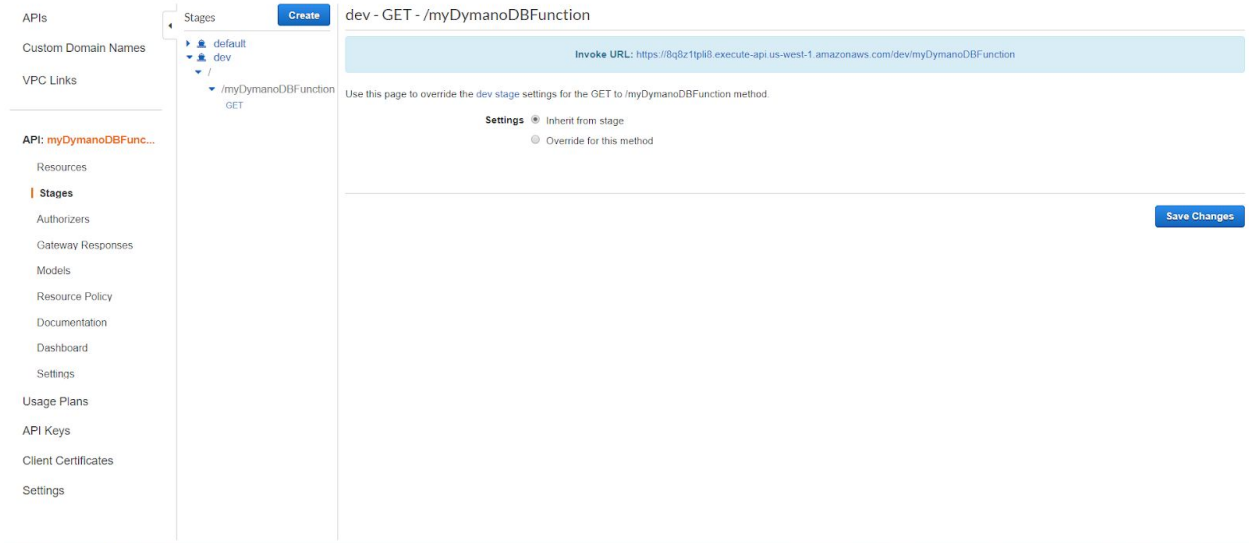
With the GET request selected choosing Deploy API

Deploy API



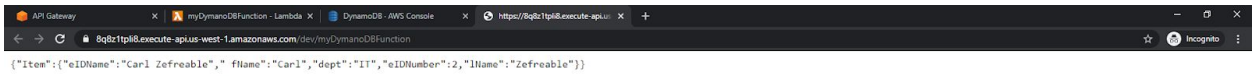
Deployment ready.

dev - GET - /myDynamoDBFunction



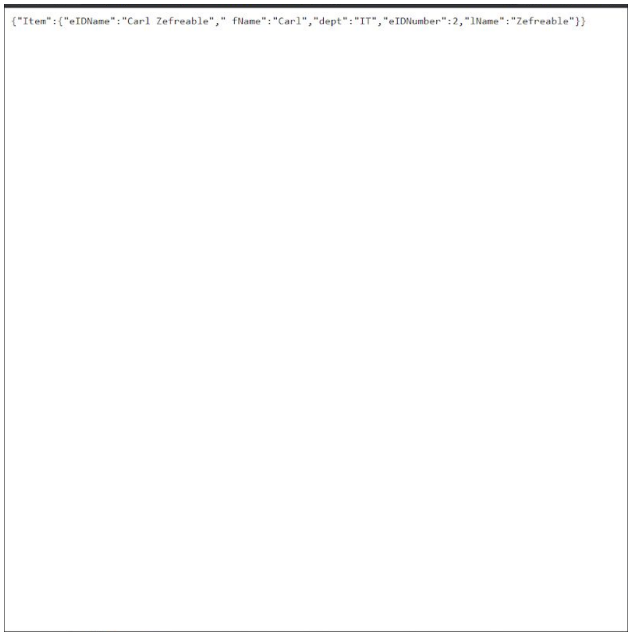
The invoke URL is ready to be tested.

Invoke URL



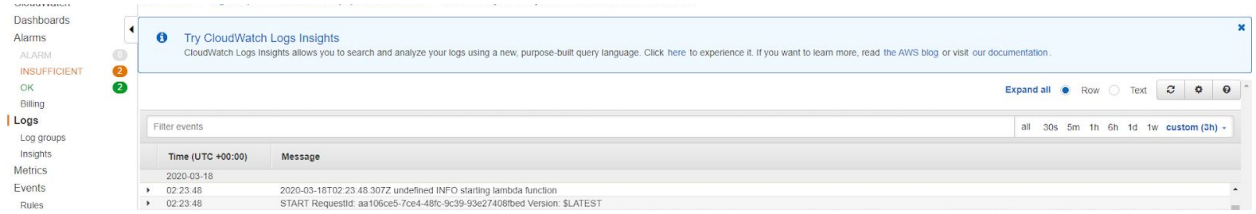
As expected from the previous test, the correct items “eIDNumber:2”are returned.

Screen comparison of Invoke URl



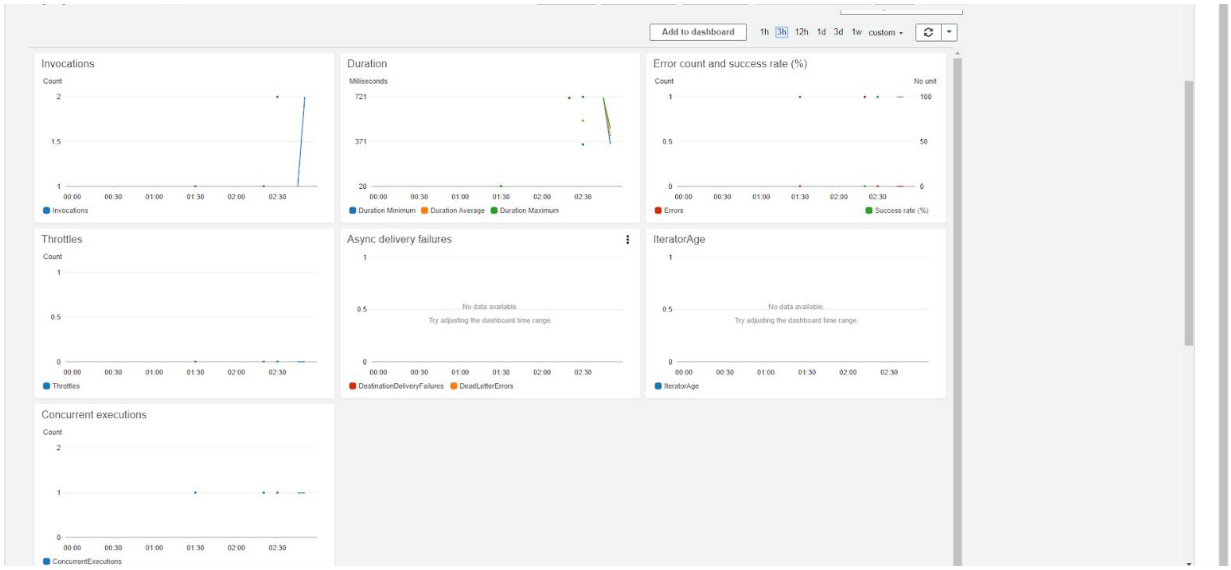
On the left the items returned with the correct Invoke URL.

CloudWatch Log Output

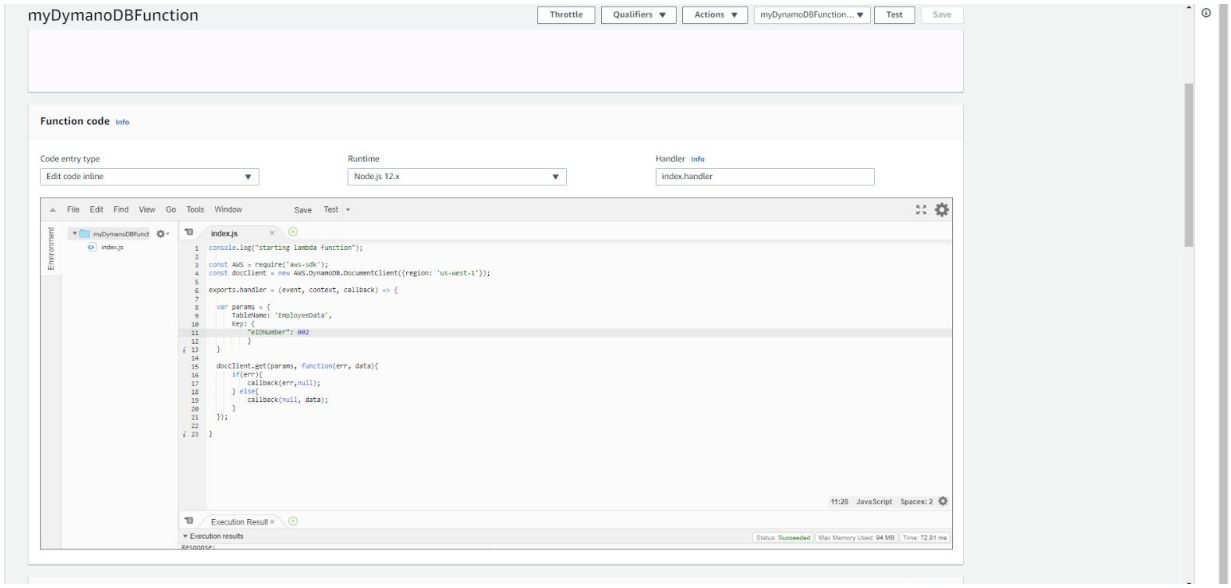


The ERROR timestamp references the incorrect Role Permissions.

myDynamoDBFunction CloudWatch metrics

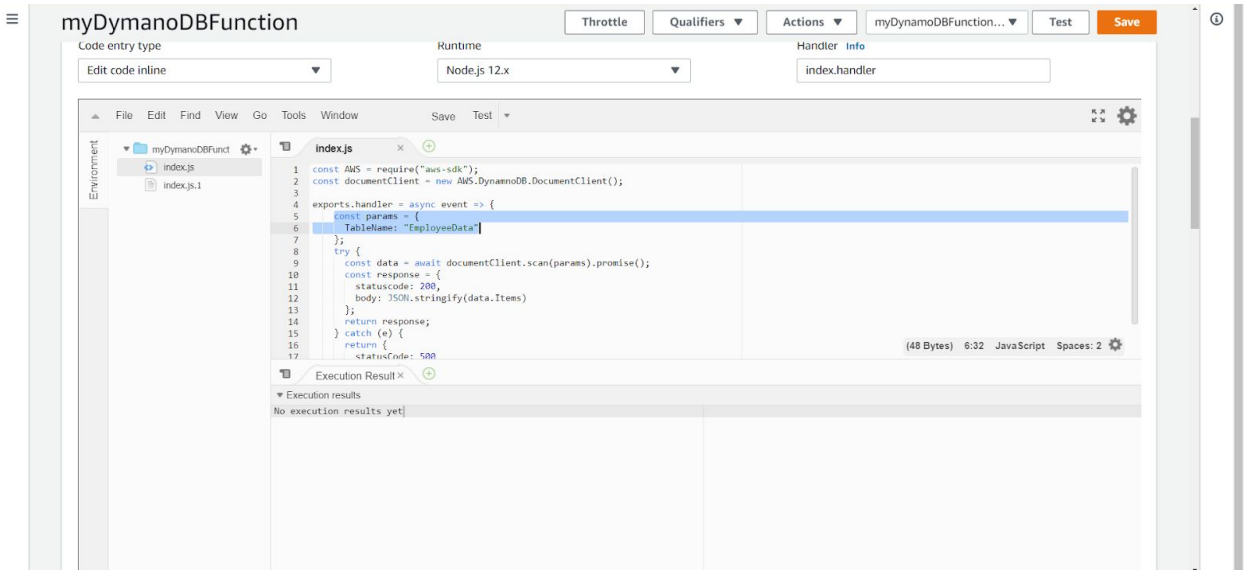


Various CloudWatch Metrics.



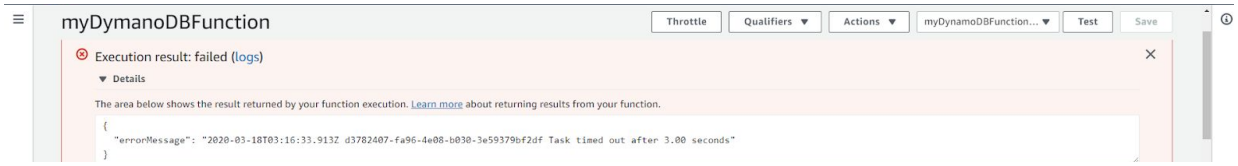
If I wanted to return the whole table (EmployeeData), I'd have to change the code I entered previously to reflect my new needs. Will I have to change anything with the API Gateway for it to work?

myDynamoDBFunction



A duplicate of the first code snippet was made and named “index.js.1”. Line 5 of the current code notes that all all data items will be returned if successful.

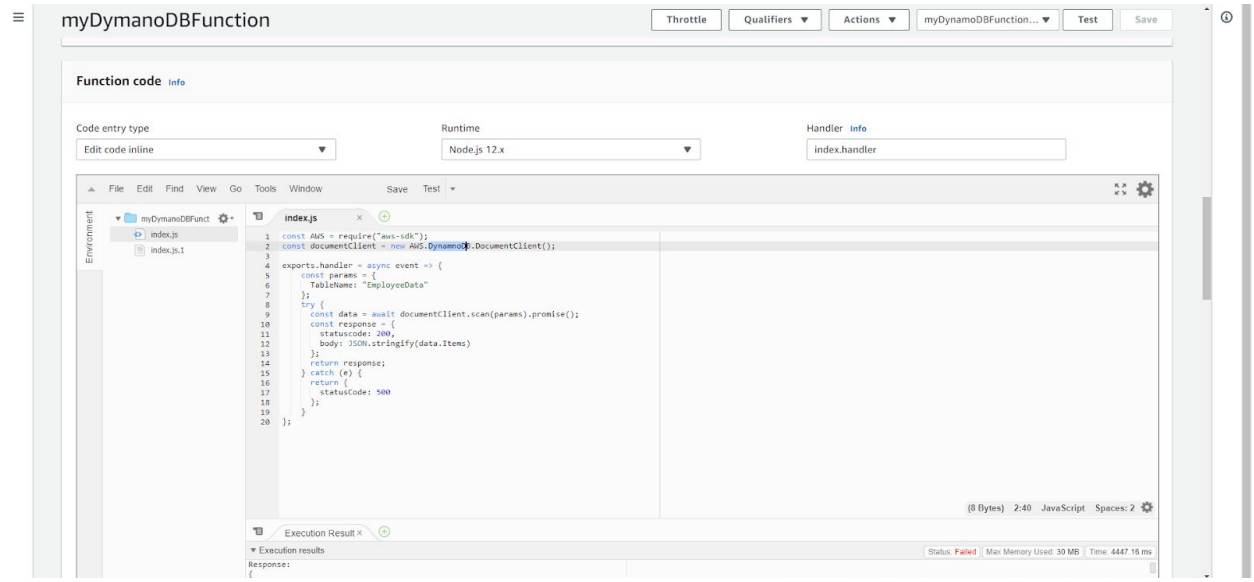
myDymanoDBFunction Error log



An error occurred. Another go may help.

myDymanoDBFunctionError log

This time a different error log.

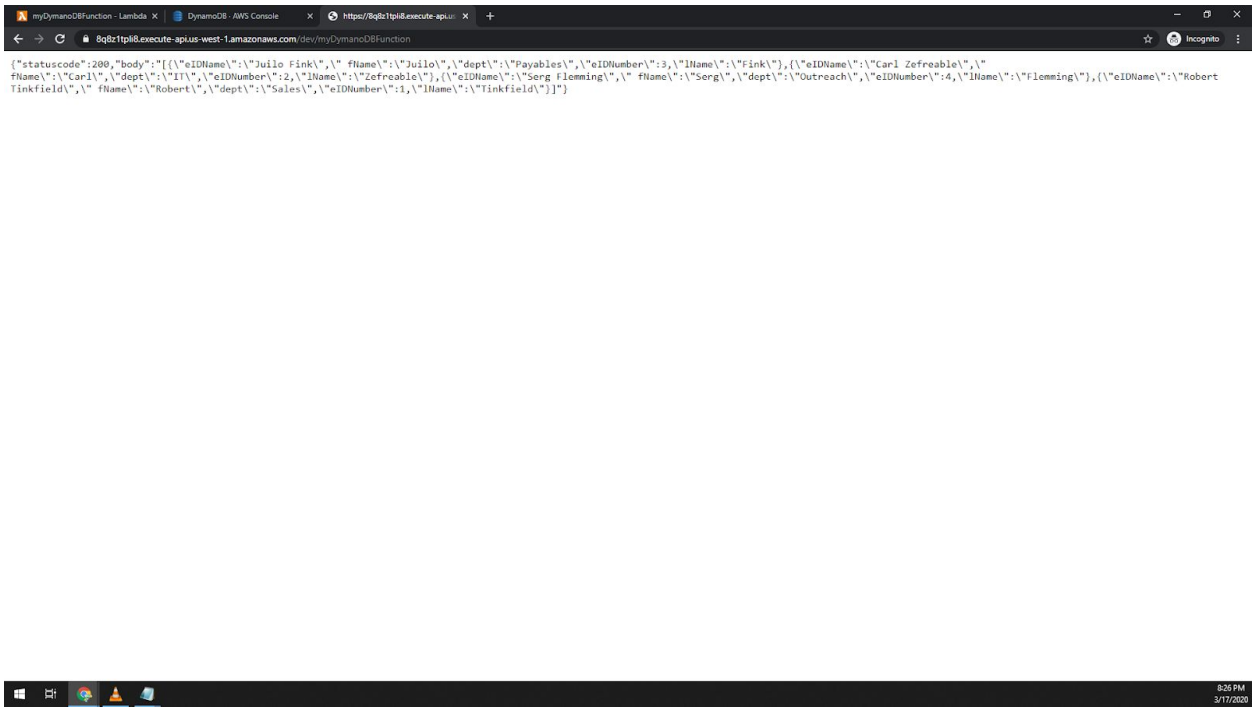


A misspelling of DynamoDB.

myDymanoDBFunction

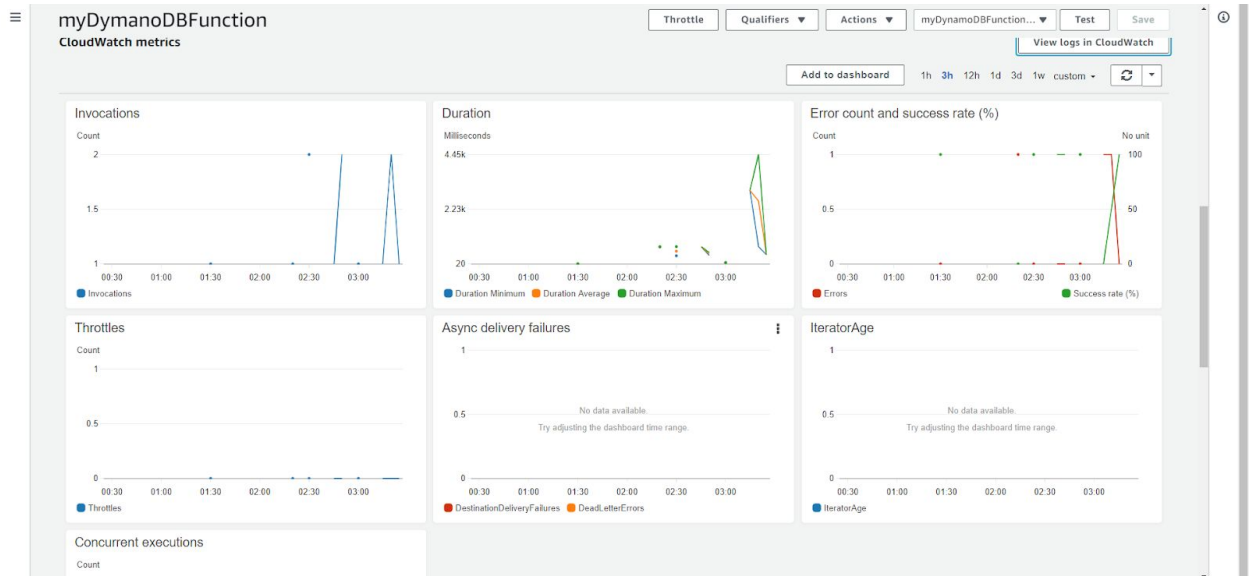
Success! Now to see if the Invoke URL is working and showing a similar output.

Invoke URL



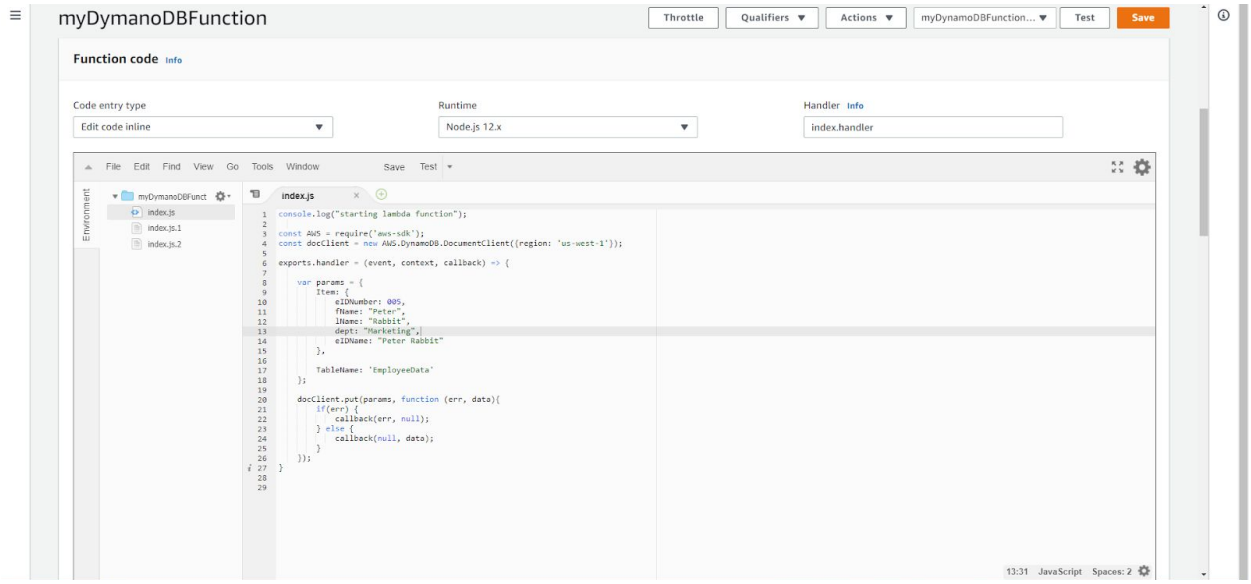
Success! All the entries are returned without any changes to the Amazon API Gateway.

myDynamoDBFunction CloudWatch metrics

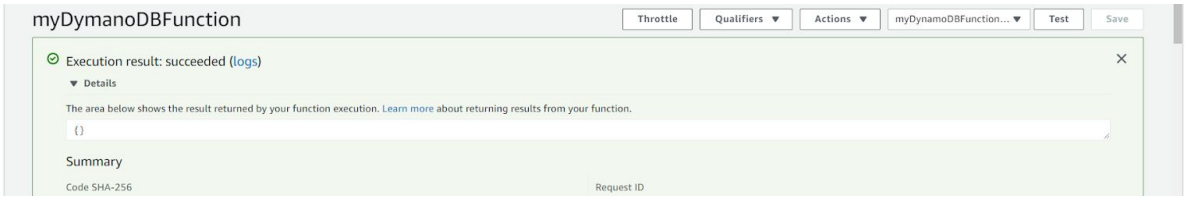


The increase of Invocations reference the updated codebase.

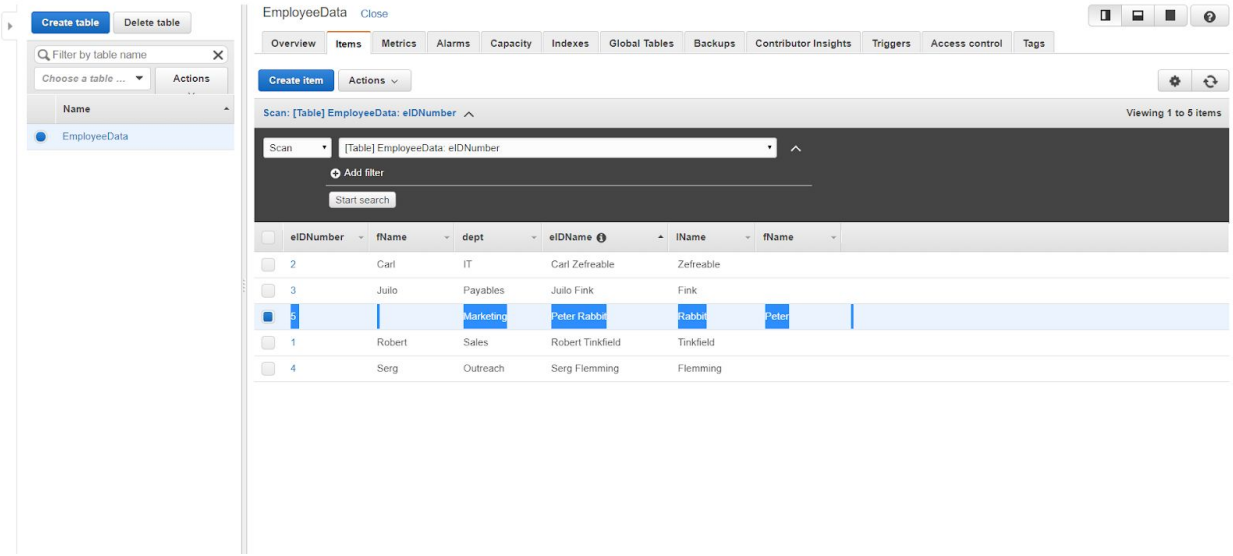
myDymanoDBFunction



Next, inserting an entry is tested. But before that another duplication named “index.js.2”. The code displayed here reflects the new requirement to add the new employee “Peter Rabbit”.

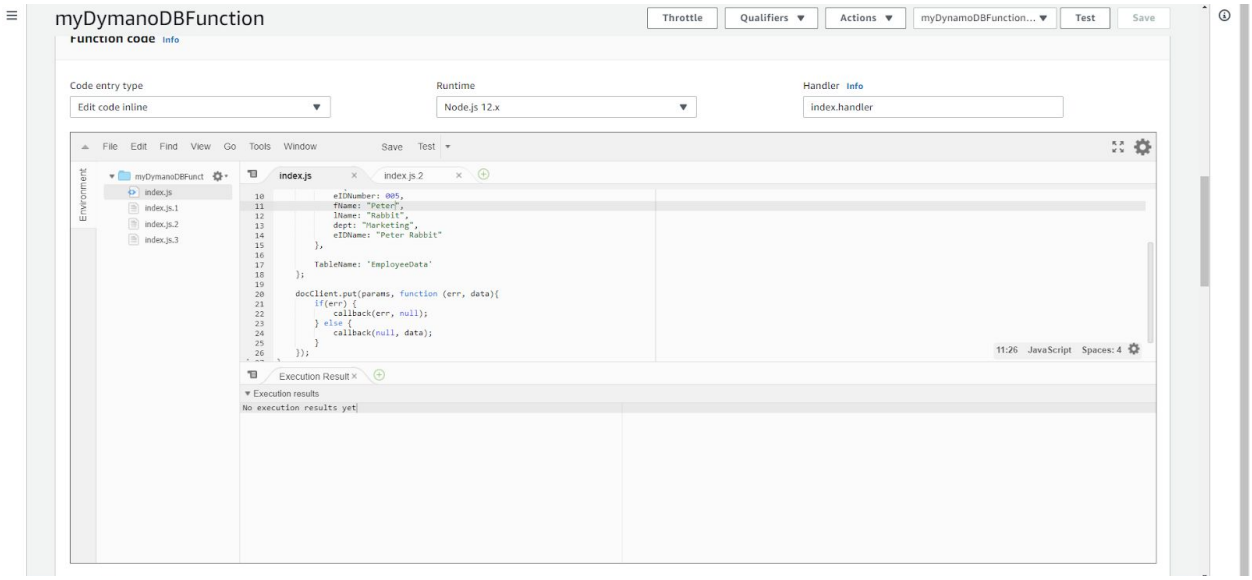


The function seems to have worked. DynamoDB may contain the new entry.



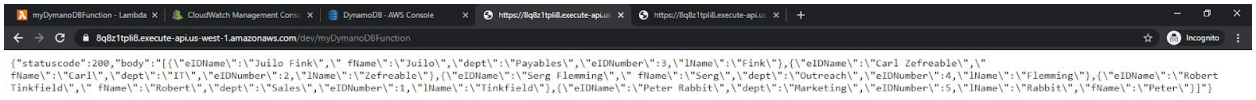
Success! But something unexpected occurred. A duplicate field (fName) was created to hold Peter.

myDymanoDBFunction



There now exists three unique function scripts for each requirement.

Invoke URL



Switching back to the GETITEMS script, we can see “Peter Rabbit” exists.