Workshop: Lambda Functions 101

This workshop is intended to take you on a short tour of making your first Lambda functions. You will make two Lambda functions in this workshop. The first Lambda function will be a hello world function to simply give you experience in creating and testing a Lambda Function. The second Lambda function will read a single record from DynamoDB so that you can do something useful with Lambda other than printing Hello World. This workshop is divided in to 4 sections as follows:

> Section 1: Create an IAM role to be used with the two lambda functions that we will create in the Workshop
> Section 2: Create a Hello World Lambda Function
> Section 3: Setting up DynamoDB with a single record
> Section 4: Using Lambda to retrieve data from DynamoDB

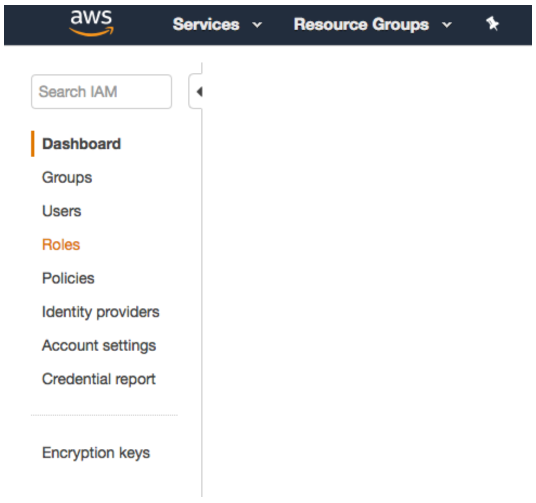
Important: Please note that this workshop should be done in your personal account using us-east-1 (N. Virginia) region. If you decide to use a different region, please note that you would have to update line 4 of the code in section 3. The same region should be used for all sections.

Section 1: Create an IAM role to be used with the two lambda functions that we will create in the Workshop

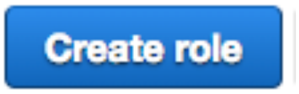
In this section you will make an IAM role that we will use for executing our Lambda function. This will NOT be a lesson on IAM so let's just make the role and we will agree to learn more about IAM on a later date unless you have already done so.

1. In the AWS console, go to the IAM service

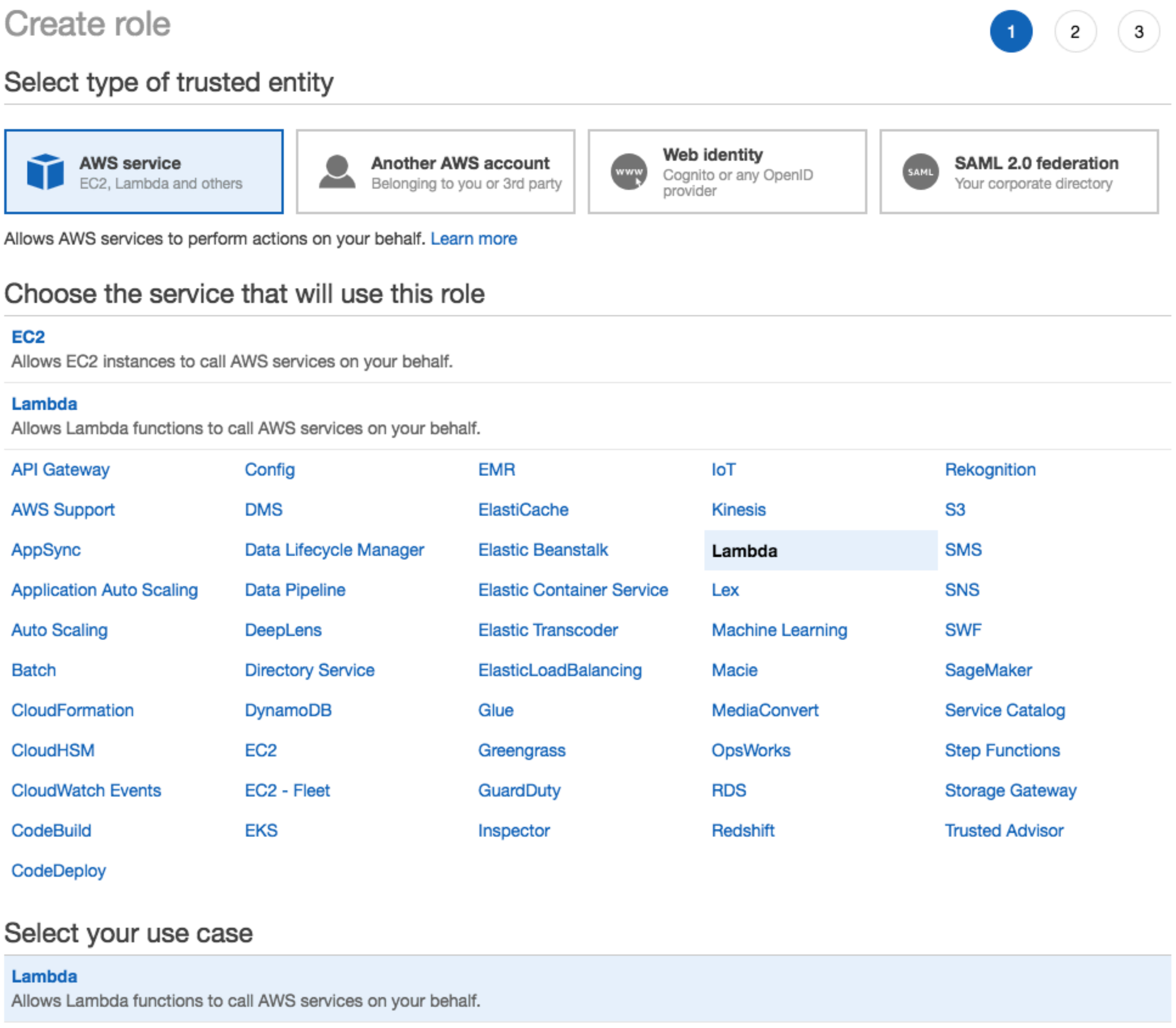
2. Select Roles from the left navigation bar:



3. Click the Create Role button:



4. Select the AWS Service Lambda to be used with this role



Create role

1 2 3

Select type of trusted entity

AWS service EC2, Lambda and others	**Another AWS account** Belonging to you or 3rd party	**Web identity** Cognito or any OpenID provider	**SAML 2.0 federation** Your corporate directory

Allows AWS services to perform actions on your behalf. Learn more

Choose the service that will use this role

EC2
Allows EC2 instances to call AWS services on your behalf.

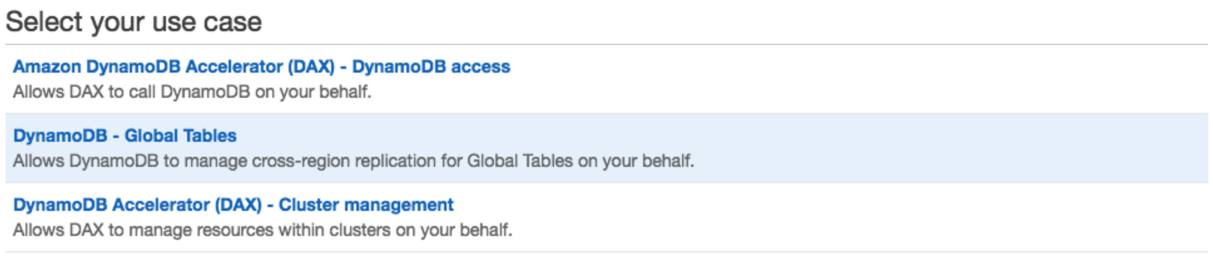
Lambda
Allows Lambda functions to call AWS services on your behalf.

API Gateway	Config	EMR	IoT	Rekognition
AWS Support	DMS	ElastiCache	Kinesis	S3
AppSync	Data Lifecycle Manager	Elastic Beanstalk	**Lambda**	SMS
Application Auto Scaling	Data Pipeline	Elastic Container Service	Lex	SNS
Auto Scaling	DeepLens	Elastic Transcoder	Machine Learning	SWF
Batch	Directory Service	ElasticLoadBalancing	Macie	SageMaker
CloudFormation	DynamoDB	Glue	MediaConvert	Service Catalog
CloudHSM	EC2	Greengrass	OpsWorks	Step Functions
CloudWatch Events	EC2 - Fleet	GuardDuty	RDS	Storage Gateway
CodeBuild	EKS	Inspector	Redshift	Trusted Advisor
CodeDeploy				

Select your use case

Lambda
Allows Lambda functions to call AWS services on your behalf.

5. Select your use case. For now, let's just use DynamoDB Global Tables:



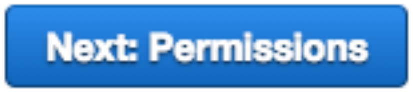
Select your use case

Amazon DynamoDB Accelerator (DAX) - DynamoDB access
Allows DAX to call DynamoDB on your behalf.

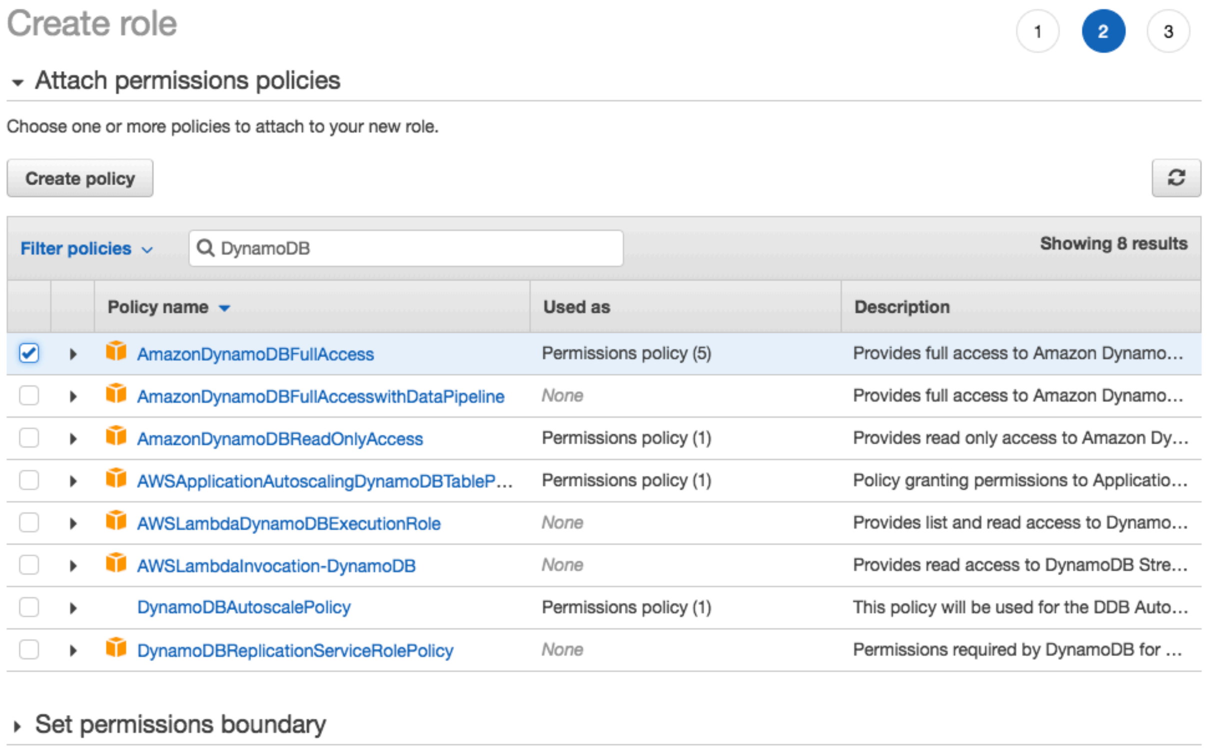
DynamoDB - Global Tables
Allows DynamoDB to manage cross-region replication for Global Tables on your behalf.

DynamoDB Accelerator (DAX) - Cluster management
Allows DAX to manage resources within clusters on your behalf.

6. Click Next: Permissions



7. Do a search for 'DynamoDB' and give your role Full Access to the database:



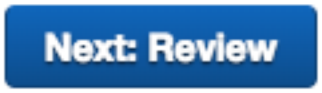
Choose one or more policies to attach to your new role.

DynamoDB

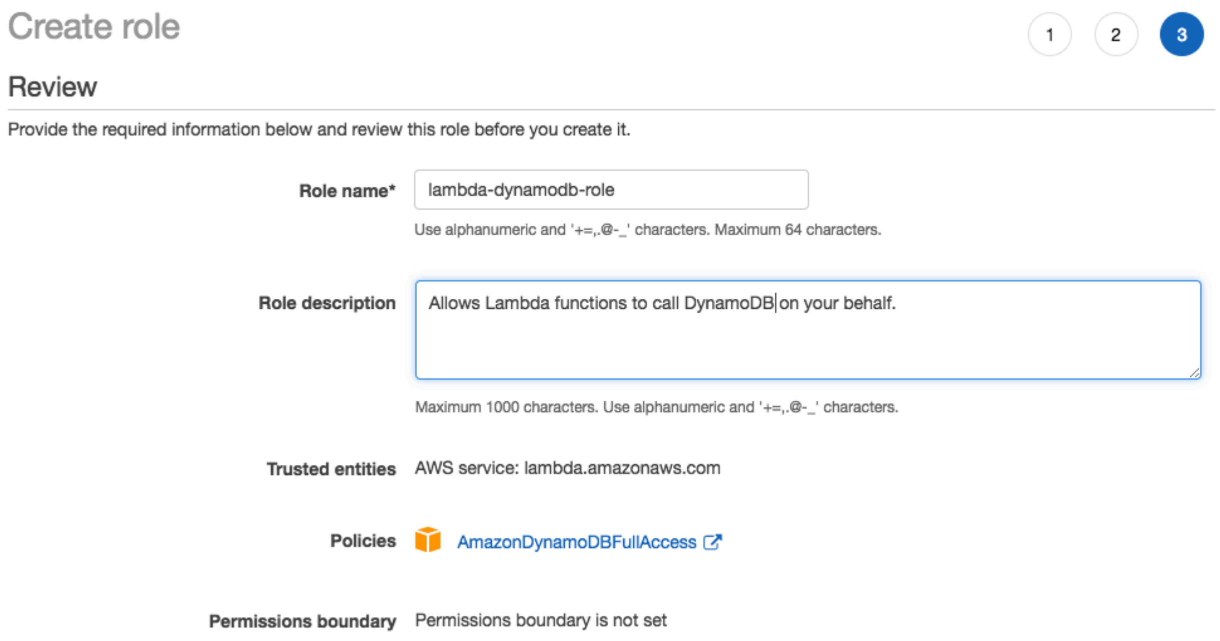
Showing 8 results

		Policy name	Used as	Description
✓	▸	AmazonDynamoDBFullAccess	Permissions policy (5)	Provides full access to Amazon Dynamo…
	▸	AmazonDynamoDBFullAccesswithDataPipeline	None	Provides full access to Amazon Dynamo…
	▸	AmazonDynamoDBReadOnlyAccess	Permissions policy (1)	Provides read only access to Amazon Dy…
	▸	AWSApplicationAutoscalingDynamoDBTableP…	Permissions policy (1)	Policy granting permissions to Applicatio…
	▸	AWSLambdaDynamoDBExecutionRole	None	Provides list and read access to Dynamo…
	▸	AWSLambdaInvocation-DynamoDB	None	Provides read access to DynamoDB Stre…
	▸	DynamoDBAutoscalePolicy	Permissions policy (1)	This policy will be used for the DDB Auto…
	▸	DynamoDBReplicationServiceRolePolicy	None	Permissions required by DynamoDB for …

8. Click Next:Review



9. Give your new role a Name and Description



Review

Provide the required information below and review this role before you create it.

Role name*	lambda-dynamodb-role
	Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.
Role description	Allows Lambda functions to call DynamoDB on your behalf.
	Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Trusted entities	AWS service: lambda.amazonaws.com
Policies	AmazonDynamoDBFullAccess
Permissions boundary	Permissions boundary is not set

10. Click Create role:

We now have a role that will be used to execute our Lambda functions. We will make two Lambda Functions in this workshop. The first function will simply be a Hello World Lambda function so that we can demonstrate how to setup a Lambda function and how to test the Lambda Function

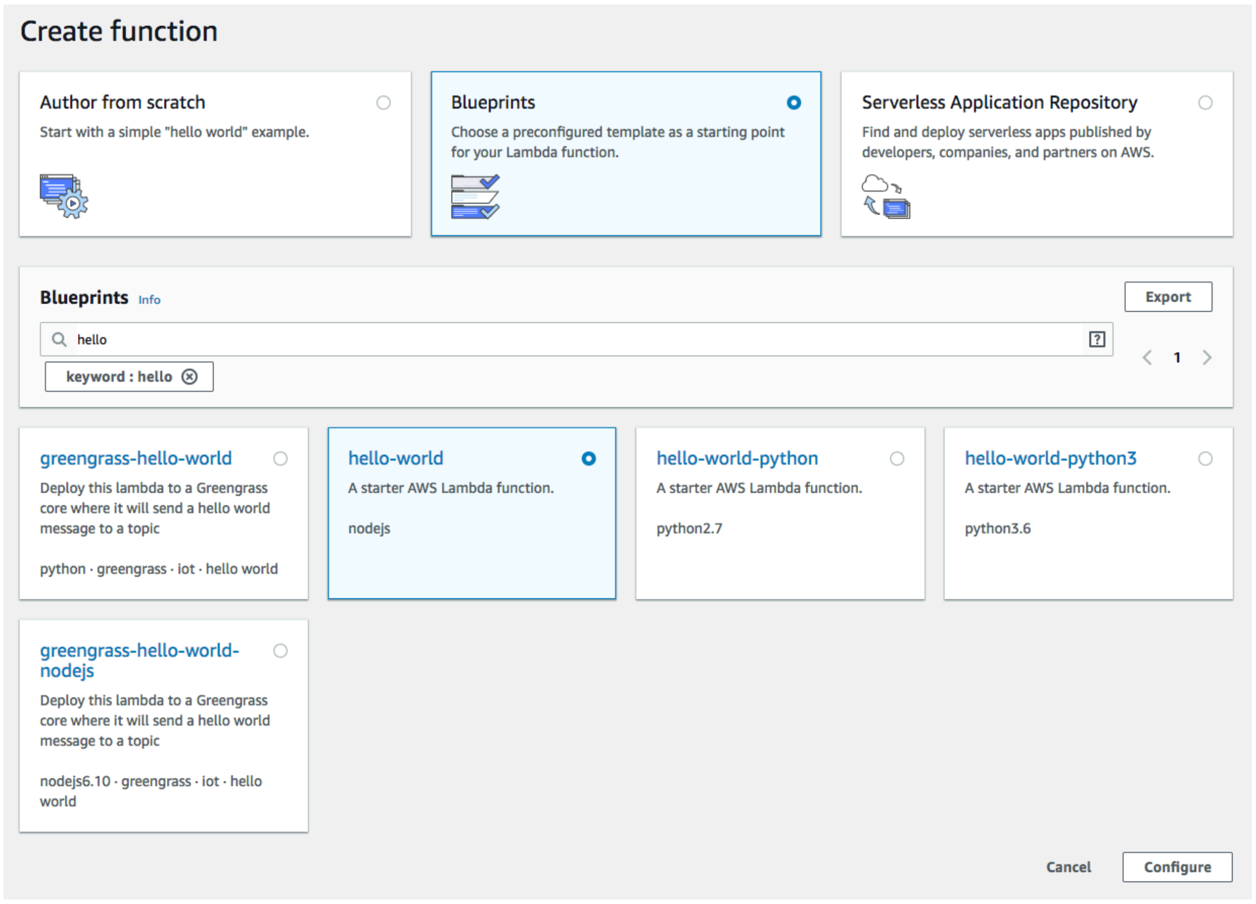
Section 2: Create a Hello World Lambda Function

In this section you will create a basic Node.js Lambda function. You will use the IAM role created in the previous section to execute the function. You will configure the function and run a test event to test the function. In addition, you will navigate to CloudWatch Logs so that you can see the logs from the lambda execution.

1. In the AWS console, go to the Lambda service

2. Click the Create function button in the upper right corner:

3. Select the Blueprints section. In the search box, type the word ‘hello’ and click return. We will be using the hello-world blueprint that says nodejs in the box. Select the hello-world blueprint and click Configure in the bottom right corner:



Create function

Author from scratch ○
Start with a simple "hello world" example.

Blueprints ●
Choose a preconfigured template as a starting point for your Lambda function.

Serverless Application Repository ○
Find and deploy serverless apps published by developers, companies, and partners on AWS.

Blueprints Info — Export

hello

keyword : hello

greengrass-hello-world ○
Deploy this lambda to a Greengrass core where it will send a hello world message to a topic
python · greengrass · iot · hello world

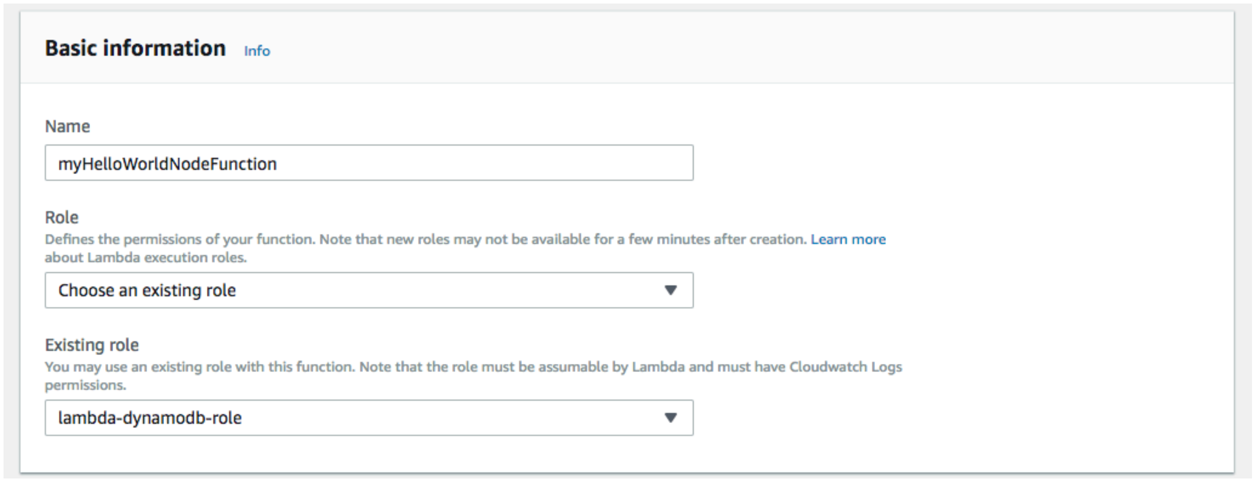
hello-world ●
A starter AWS Lambda function.
nodejs

hello-world-python ○
A starter AWS Lambda function.
python2.7

hello-world-python3 ○
A starter AWS Lambda function.
python3.6

greengrass-hello-world-nodejs ○
Deploy this lambda to a Greengrass core where it will send a hello world message to a topic
nodejs6.10 · greengrass · iot · hello world

4. Give your Lambda function a name and configure it to use the role we created in section 1 of this workshop:



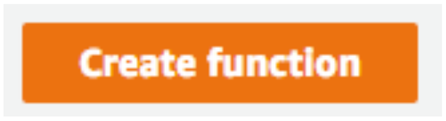
Basic information Info

Name
myHelloWorldNodeFunction

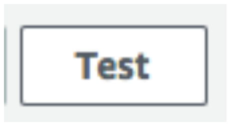
Role
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. Learn more about Lambda execution roles.
Choose an existing role

Existing role
You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Logs permissions.
lambda-dynamodb-role

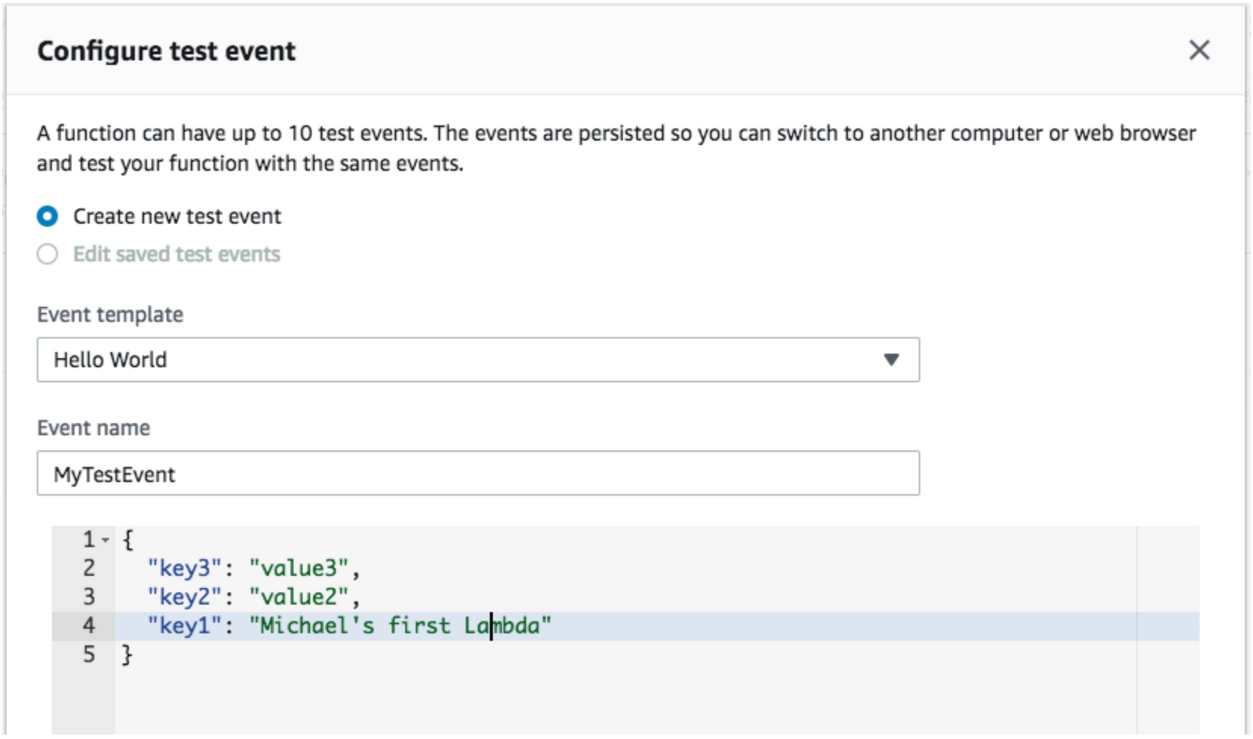
5. Click Create function:



6. Click the Test Button in the upper right corner



7. Give your event a name and change value1 of key1 to something unique to you:



Configure test event

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

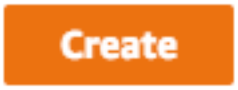
- (●) Create new test event
- ( ) Edit saved test events

Event template: Hello World

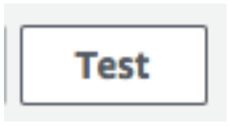
Event name: MyTestEvent

```
{
  "key3": "value3",
  "key2": "value2",
  "key1": "Michael's first Lambda"
}
```

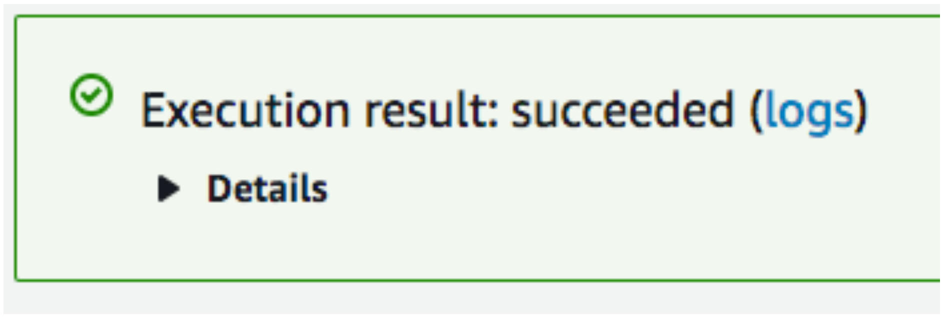
8. Click the Create Button:



9. Click the test button again:



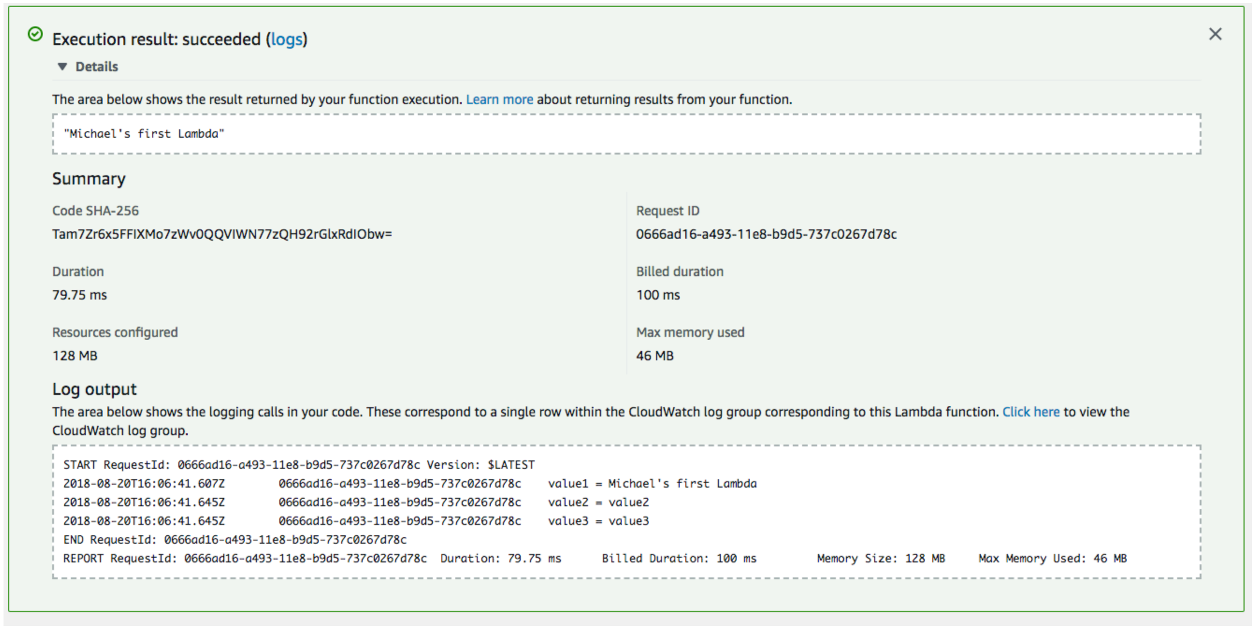
10. In the Execution result section, expand the Details by click on it:



Execution result: succeeded (logs)

▶ Details

11. Your custom message should appear along with details about the execution of the function:



Execution result: succeeded (logs)

▼ Details

The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
"Michael's first Lambda"
```

Summary

Code SHA-256
Tam7Zr6x5FFlXMo7zWv0QQVIWN77zQH92rGlxRdIObw=

Request ID
0666ad16-a493-11e8-b9d5-737c0267d78c

Duration
79.75 ms

Billed duration
100 ms

Resources configured
128 MB

Max memory used
46 MB

Log output

The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. Click here to view the CloudWatch log group.

```
START RequestId: 0666ad16-a493-11e8-b9d5-737c0267d78c Version: $LATEST
2018-08-20T16:06:41.607Z        0666ad16-a493-11e8-b9d5-737c0267d78c    value1 = Michael's first Lambda
2018-08-20T16:06:41.645Z        0666ad16-a493-11e8-b9d5-737c0267d78c    value2 = value2
2018-08-20T16:06:41.645Z        0666ad16-a493-11e8-b9d5-737c0267d78c    value3 = value3
END RequestId: 0666ad16-a493-11e8-b9d5-737c0267d78c
REPORT RequestId: 0666ad16-a493-11e8-b9d5-737c0267d78c  Duration: 79.75 ms      Billed Duration: 100 ms         Memory Size: 128 MB     Max Memory Used: 46 MB
```

There are several things to note here. First, line 8 of your code has a return statement. That value is shown in the top section of the result details. But, we should also note that lines 5-8 have console.log statements. Those appear in the log output at the bottom of the details. This is just a snippet from the CloudWatch log group specific to your this Lambda execution.

There is much to explore on your functions page. Most notably, you will explore the following in more detail as you use Lambda more:

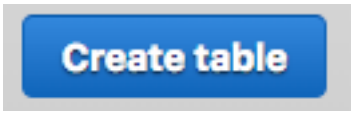
1. Triggers: Using other AWS services to launch your Lambda
2. Monitoring: Visual analysis of your functions usage and performance
3. Environment Variables: abstracting variables outside of your function for flexibility
4. Tags: Used for billing, tracking, and organizing your AWS services
5. Timeout: control the code execution performance and costs for your Lambda function
6. VPC: Allowing Lambda to access resources in your custom VPC
7. Using CloudWatch and AWS X-Ray to troubleshoot and diagnose your code

Section 3: Setting up DynamoDB with a single record

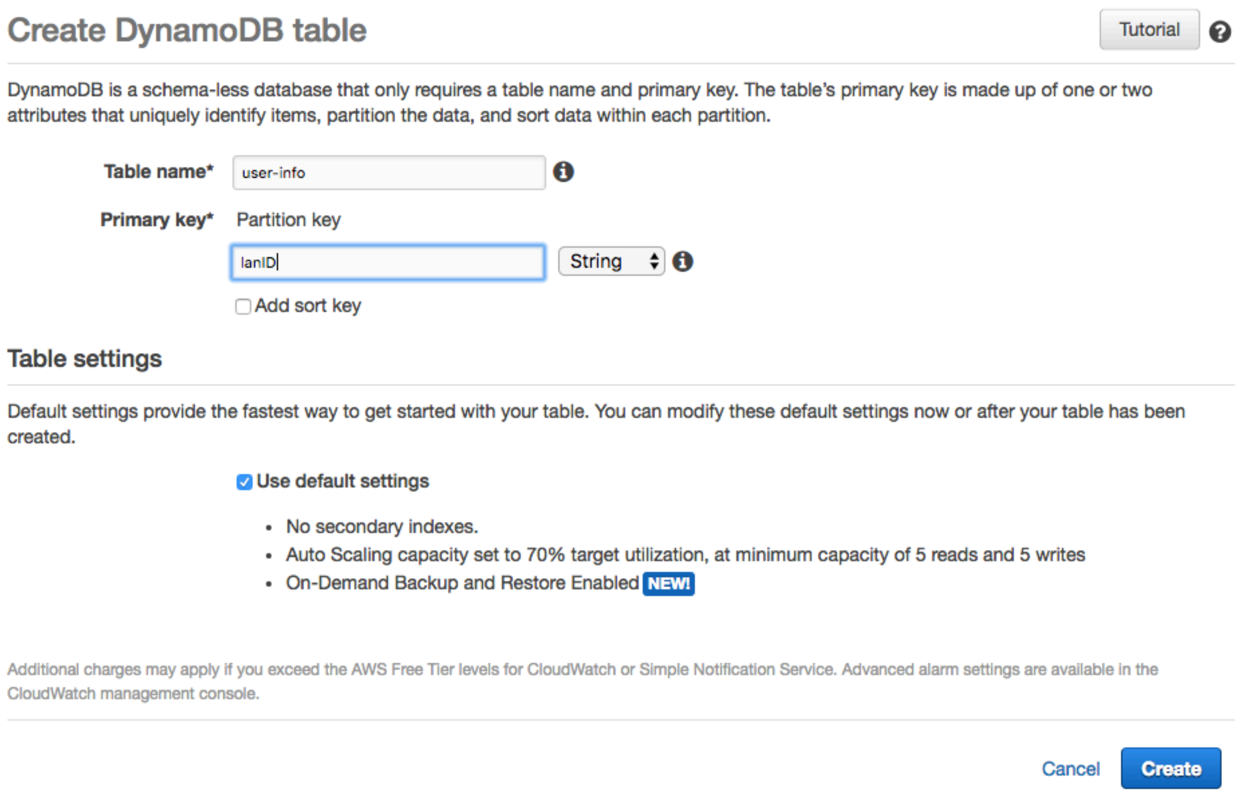
In this section we are NOT going to learn DynamoDB. However, we will create a table and add a single record to the table.

1. In the AWS console, select the DynamoDB service

2. Select the Create table button:



3. Give the table a name of 'user-info' and set the Partition key to 'lanID'. Then, click Create:



Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* user-info

Primary key* Partition key lanID String

☐ Add sort key

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

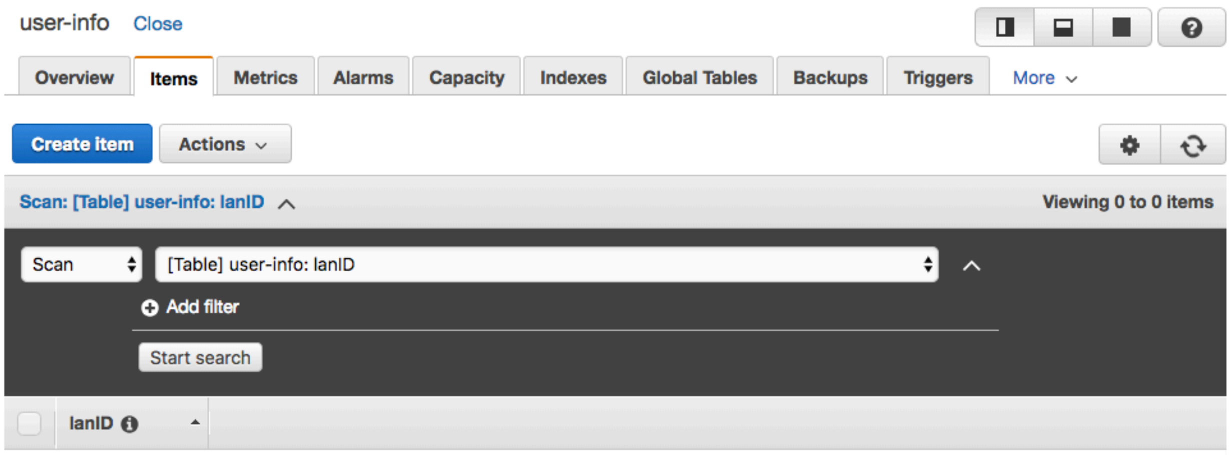
☑ Use default settings

- No secondary indexes.
- Auto Scaling capacity set to 70% target utilization, at minimum capacity of 5 reads and 5 writes
- On-Demand Backup and Restore Enabled NEW!

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

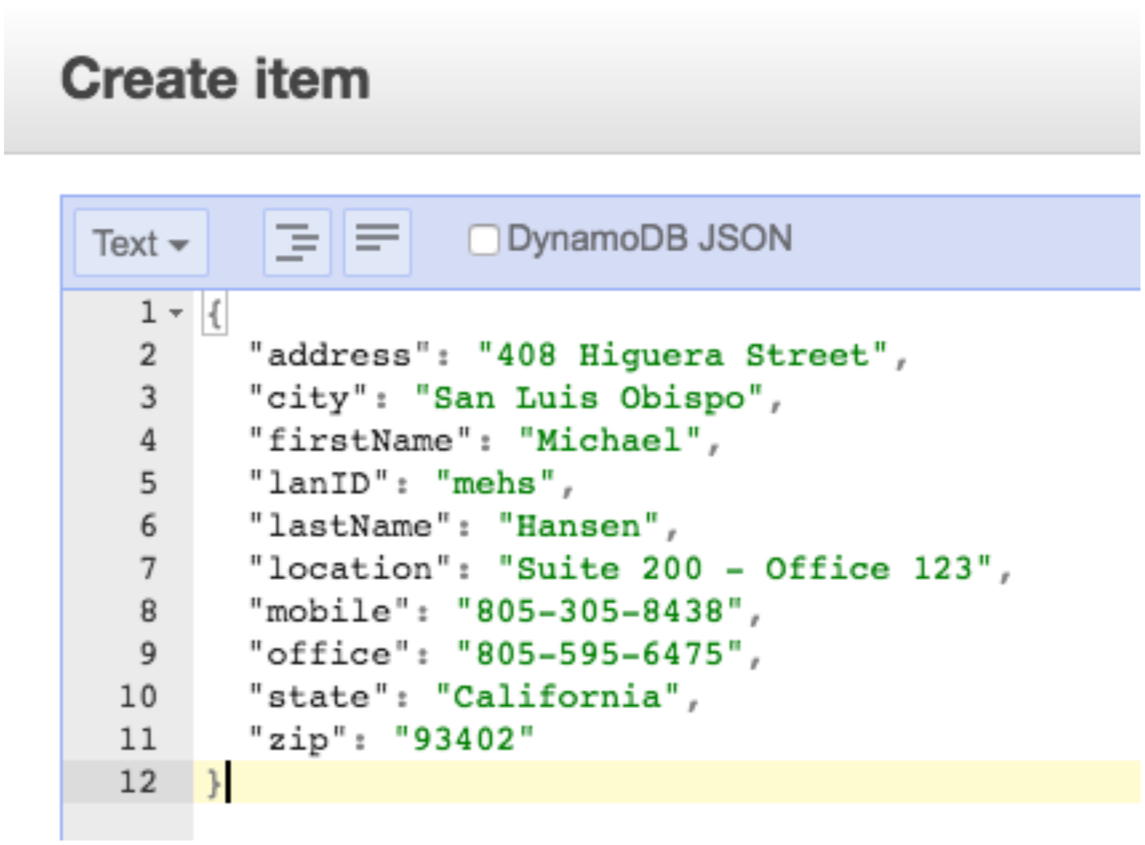
Note that we are not going to worry at this point about partition keys, sort keys, provisioned capacity, streams, or any of the other details of DynamoDB. Lets just get some data in the DB for the example.

4. Click on the Items tab and click the Create item button:



5. In the upper left corner, change the view from tree to text. Then, place the following XML as the record:

```
{
    "lanID": "mehs",
    "firstName": "Michael",
    "lastName": "Hansen",
    "mobile": "805-305-8438",
    "office": "805-595-6475",
    "address": "408 Higuera Street",
    "location": "Suite 200 - Office 123",
    "city": "San Luis Obispo",
    "state": "California",
    "zip": "93402"
}
```



```
{
  "address": "408 Higuera Street",
  "city": "San Luis Obispo",
  "firstName": "Michael",
  "lanID": "mehs",
  "lastName": "Hansen",
  "location": "Suite 200 - Office 123",
  "mobile": "805-305-8438",
  "office": "805-595-6475",
  "state": "California",
  "zip": "93402"
}
```

6. Click the Save button:

We should now have a table in DynamoDB and a single record in Dynamo DB. In our next section, we will write a Lambda function that will read this record.

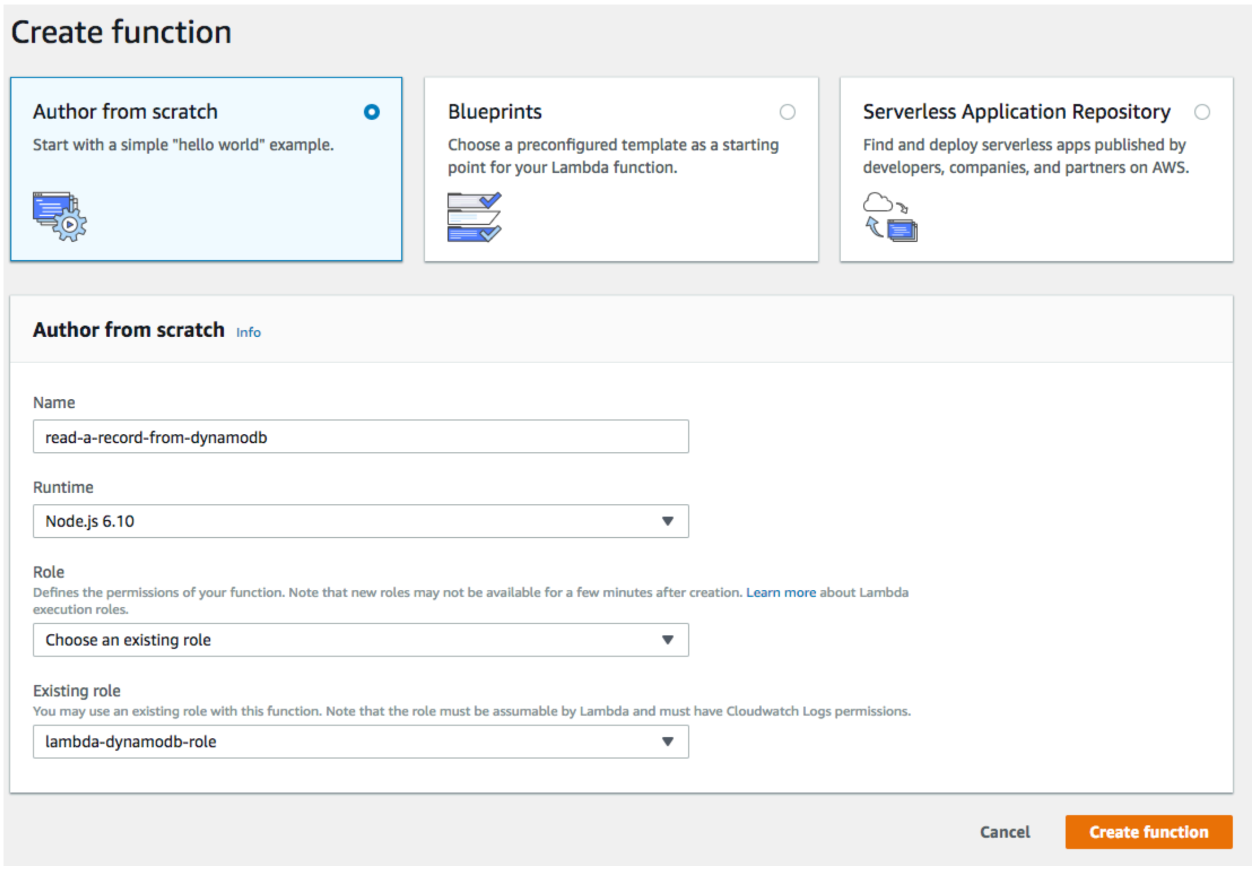
Section 4: Using Lambda to retrieve data from DynamoDB

In this section we will create a blank Nodejs function and will write some code that will access DynamoDB and return a single record. We will not use a trigger event for the Lambda. We will simply run a test event to exercise the Lambda function and will view our record as a result returned from the function as well as a console.log entry sent to CloudWatch Logs.

1. In the AWS console, go to the Lambda service

2. Click the Create function button in the upper right corner:

3. Leave 'Author from scratch' selected and give your Lambda function a name. Also, configure the function to use the role that we created in section 1. Then click Create function:



Create function

Author from scratch
Start with a simple "hello world" example.

Blueprints
Choose a preconfigured template as a starting point for your Lambda function.

Serverless Application Repository
Find and deploy serverless apps published by developers, companies, and partners on AWS.

Author from scratch Info

Name
read-a-record-from-dynamodb

Runtime
Node.js 6.10

Role
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. Learn more about Lambda execution roles.
Choose an existing role

Existing role
You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Logs permissions.
lambda-dynamodb-role

4. Normally, we would write our Lambda function in an IDE and import the code in to Lambda. My recommendation for coding in Lambda is to use the AWS Cloud9 service. However, for this example we are just going to edit the Function code directly in the Lambda editor on the function page. So, in the "Function code" section, make sure index.js is select in the code tree and replace the code with the following:

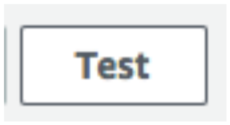
```
var AWS = require("aws-sdk");

AWS.config.update({
    region: 'us-east-1'
});
var docClient = new AWS.DynamoDB.DocumentClient();
```

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

    // get the entered user from the query string
    var currentUser = 'mehs'; //req.params.lanID;

    // get the user info if it exits
    var params = {
        TableName: 'user-info',
        KeyConditionExpression: "lanID = :value",
        ExpressionAttributeValues: {
            ":value": currentUser
        },
        "ProjectionExpression": "lanID, firstName, lastName, mobile"
    };


    docClient.query(params, function(err, data) {
        console.log("data: " + JSON.stringify(data));
        console.log("error: " + err);
        if (err) {
            callback(err, null);
        }
        else {
            callback(null, data);
        }

    });
};
```

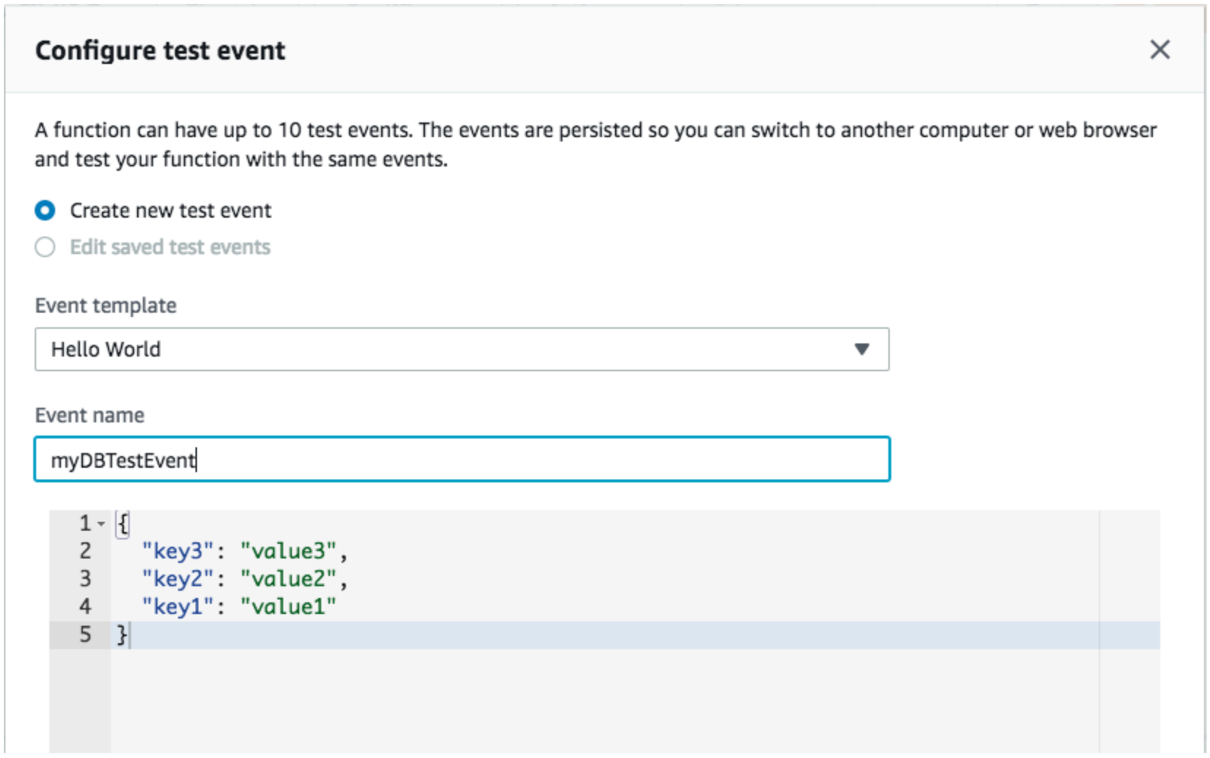
Note that we have hardcoded the user name instead of getting the value from a query string. Using query strings provided by the API Gateway is a different workshop. However, I wanted to illustrate that this Lambda could be used to back an API.

5. Click the Save button:

6. Click the Test Button in the upper right corner



7. Give your event a name:



Configure test event

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

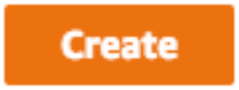
● Create new test event
○ Edit saved test events

Event template: Hello World

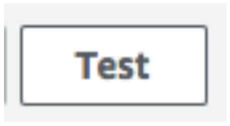
Event name: myDBTestEvent

```
{
  "key3": "value3",
  "key2": "value2",
  "key1": "value1"
}
```

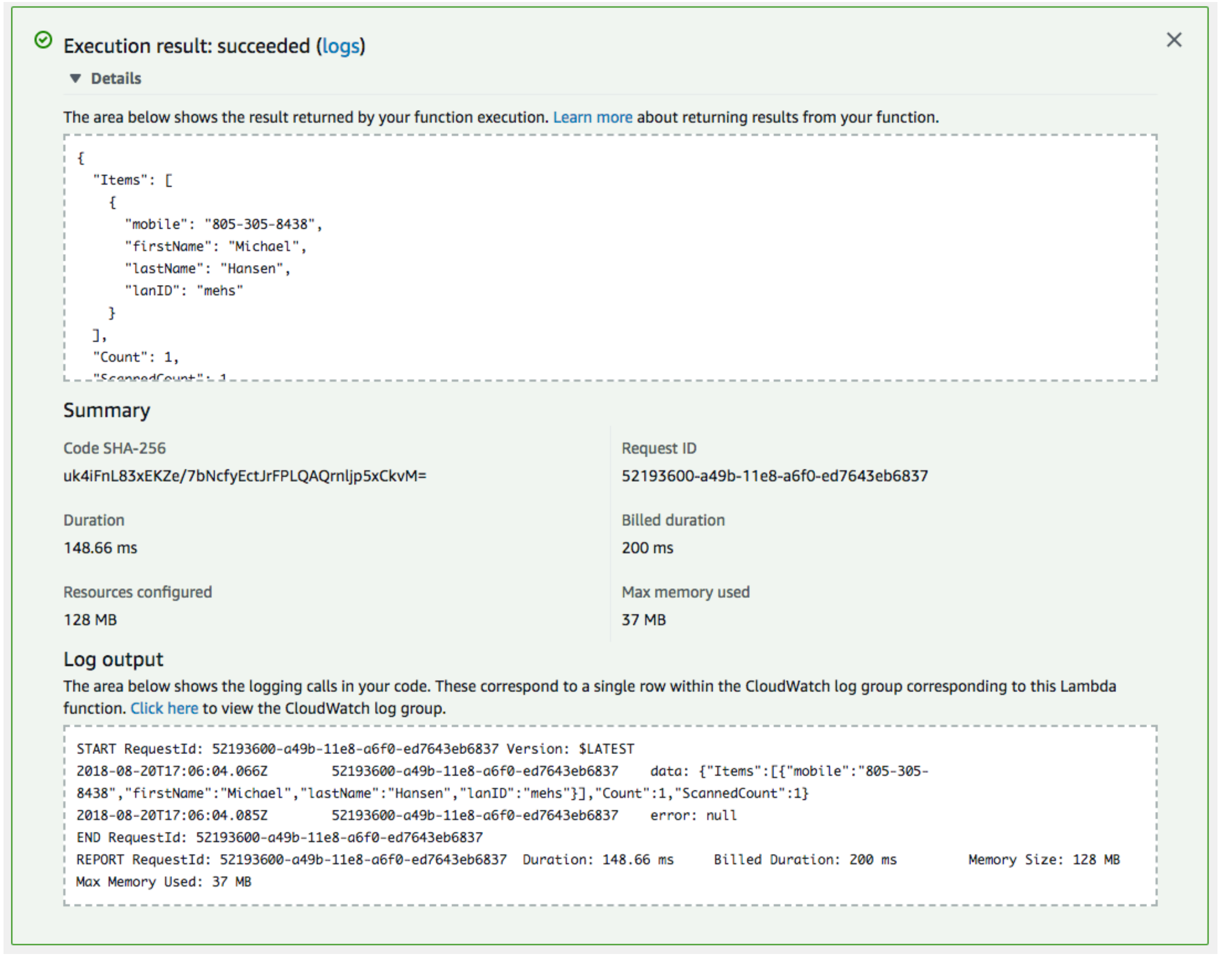
8. Click the Create Button:



9. Click the test button again:



10. Running the test should output the following result:



Execution result: succeeded (logs)

Details

The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
{
  "Items": [
    {
      "mobile": "805-305-8438",
      "firstName": "Michael",
      "lastName": "Hansen",
      "lanID": "mehs"
    }
  ],
  "Count": 1,
  "ScannedCount": 1
```

Summary

Code SHA-256	Request ID
uk4iFnL83xEKZe/7bNcfyEctJrFPLQAQrnljp5xCkvM=	52193600-a49b-11e8-a6f0-ed7643eb6837
Duration	Billed duration
148.66 ms	200 ms
Resources configured	Max memory used
128 MB	37 MB

Log output

The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. Click here to view the CloudWatch log group.

```
START RequestId: 52193600-a49b-11e8-a6f0-ed7643eb6837 Version: $LATEST
2018-08-20T17:06:04.066Z        52193600-a49b-11e8-a6f0-ed7643eb6837    data: {"Items":[{"mobile":"805-305-
8438","firstName":"Michael","lastName":"Hansen","lanID":"mehs"}],"Count":1,"ScannedCount":1}
2018-08-20T17:06:04.085Z        52193600-a49b-11e8-a6f0-ed7643eb6837    error: null
END RequestId: 52193600-a49b-11e8-a6f0-ed7643eb6837
REPORT RequestId: 52193600-a49b-11e8-a6f0-ed7643eb6837  Duration: 148.66 ms     Billed Duration: 200 ms         Memory Size: 128 MB
Max Memory Used: 37 MB
```

There are several things to note here. First note that the output from DynamoDB that is returned as the function result is in XML. Also note that it has the items returned (in this case a single item) in the *Items[]* array. Also note that the result includes the count as well as the number of records scanned in the DB. Second, note that we did return the data as well as the error message (in this case null) in the CloudWatch log. Third, note that I did not return all values found in the DB in the result. Line 20 in the code identifies the Projection Expression which identifies what fields will be returned. Feel free to edit this list and run a few more tests.