

How to build a web application with front end and backend components using AWS technology

This document outlines the steps required to build a web application using technology from AWS. Specifically, we will be using

- DynamoDB to store the data in the back end
- Lambda functions in the middle to read and write data
- API Gateway features to access the lambda functions

For this example, I will be setting up a database that stores data from the Ontario Government web site concerning Covid-19. The same approach could be used for any type of data.

The steps involved are

1. Create a table in DynamoDB for the data
2. Create the Lambda functions
3. Create the API Gateway

We are going to do things iteratively. We will do steps 1-3 and then repeat steps 2-3 for the addition of a second lambda function

Assumptions

For this, I assume:

- You have access to AWS and some familiarity with it
- You will be setting things up in US-east-1
- You have access to the curl command

Create a table in DynamoDB

To work with Dynamo, go here: <https://us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#tables>

Click the button: Create Table (this is a screen grab of my account. I already have some tables)

DynamoDB > Tables

Tables (4) Info

↺

Actions ▾

Delete

Create table

🔍 Find tables by table name

Any table tag ▾

< 1 > ⚙️

<input type="checkbox"/>	Name ▲	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode	Size
<input type="checkbox"/>	InternetSpeed	✔️ Active	ID (S)	-	0	Provisioned (5)	Provisioned (5)	804 kilobytes
<input type="checkbox"/>	Movies	✔️ Active	Id (S)	-	0	Provisioned (5)	Provisioned (5)	1 kilobyte
<input type="checkbox"/>	songs	✔️ Active	Artist (S)	SongTitle (S)	0	Provisioned (5)	Provisioned (5)	428 bytes
<input type="checkbox"/>	test	✔️ Active	email (S)	-	0	Provisioned (5)	Provisioned (5)	134 bytes

You will see the screen below. Fill it in this and then scroll down and click on Create Table button

DynamoDB > Tables > Create table

Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

CovidOntario

Between 3 and 255 characters, containing only letters, numbers, underscores (`_`), hyphens (`-`), and periods (`.`).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

theDate

String ▾

1 to 255 characters and case sensitive.

Sort key - optional

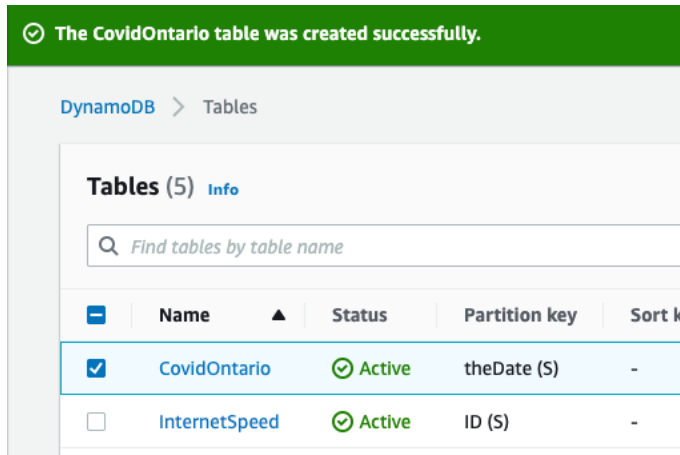
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Enter the sort key name

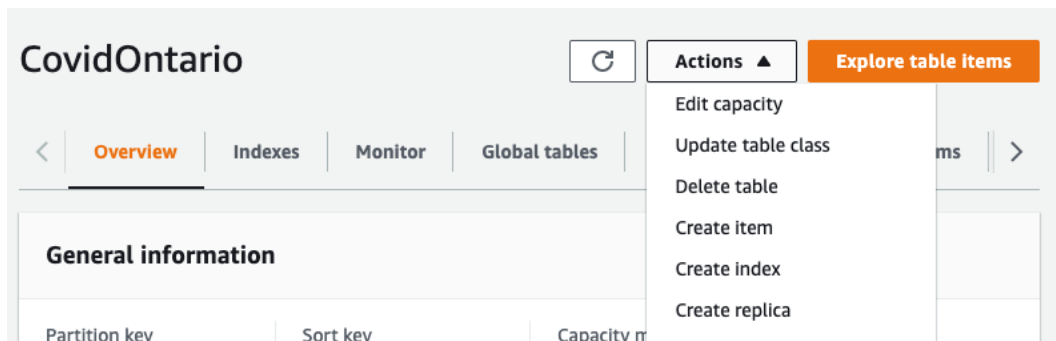
String ▾

1 to 255 characters and case sensitive.

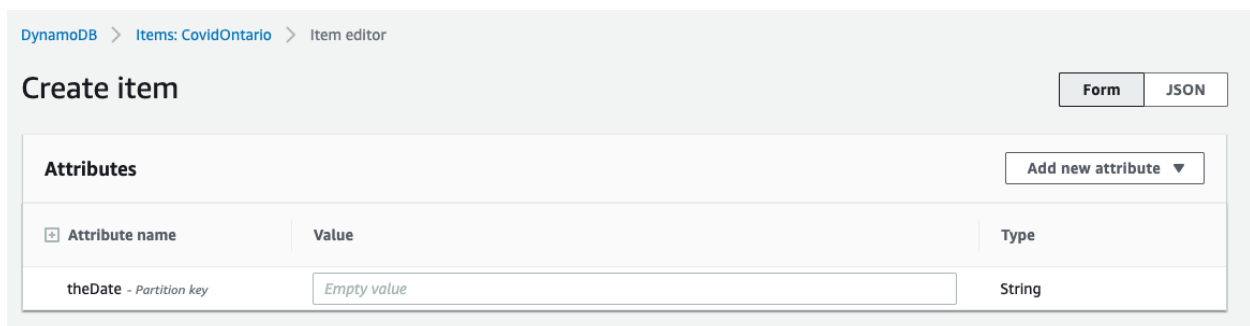
Wait until you see the following. Congrats, you have your table.



Now click on CovidOntario, then click on Actions > Create item



You want to be in Form mode, not Json mode. Now you are going to add attributes under theDate: hospitalized, inICU, cases. Each will be a number.



It should look like this. Now click Create Item. You will have to give theDate a value. Use 0.

Create item

Form

JSON

Attributes

Add new attribute ▼

Attribute name	Value	Type	
theDate - Partition key	Empty value	String	
hospitalized	0	Number	Remove
inICU	0	Number	Remove
cases	0	Number	Remove

Cancel

Create Item

Now that we have a DynamoDB table with one item in it (you can delete it later). Let’s make a Lambda function to talk to it.

Second, create a lambda function

To create Lambda functions, go here:
<https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions>

You will see a screen like this. Click on the button: Create function

Lambda > Functions

Functions (11)

Last fetched 50 seconds ago

Actions ▼

Create function

Filter by tags and attributes or search by keyword

< 1 2 > ⚙

	Function name ▼	Description ▼	Package type ▼	Runtime ▼	Code size ▼	Last modified ▼
<input type="checkbox"/>	ListMovies	List entries in movies db	Zip	Node.js 12.x	420.0 byte	3 hours ago
<input type="checkbox"/>	CreateMovie	Add an entry to a movie	Zip	Node.js 12.x	400.0 byte	3 hours ago

Select Author from scratch and go with defaults. You will have to give the function a name and choose a runtime like I did for this to work:

Lambda > Functions > Create function

Create function [Info](#)

Choose one of the following options to create your function.

Author from scratch ☒
Start with a simple Hello World example.

Use a blueprint ☐
Build a Lambda application from sample code and configuration presets for common use cases.

Container image ☐
Select a container image to deploy for your function.

Bridge CLI ☐
Deploy a function from a local development environment.

Basic information

Function name
Enter a name that describes the purpose of your function.

ListCOVIDdata

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Node.js 14.x ▼

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.

☒ x86_64
☐ arm64

The second half of this screen is below:

Architecture [Info](#)

Choose the instruction set architecture you want for your function code.

- ☒ x86_64
☐ arm64

Permissions [Info](#)

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default 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

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/mySimpleMicroservicepermissions ▼

↻

[View the mySimpleMicroservicepermissions role](#) on the IAM console.

► Advanced settings

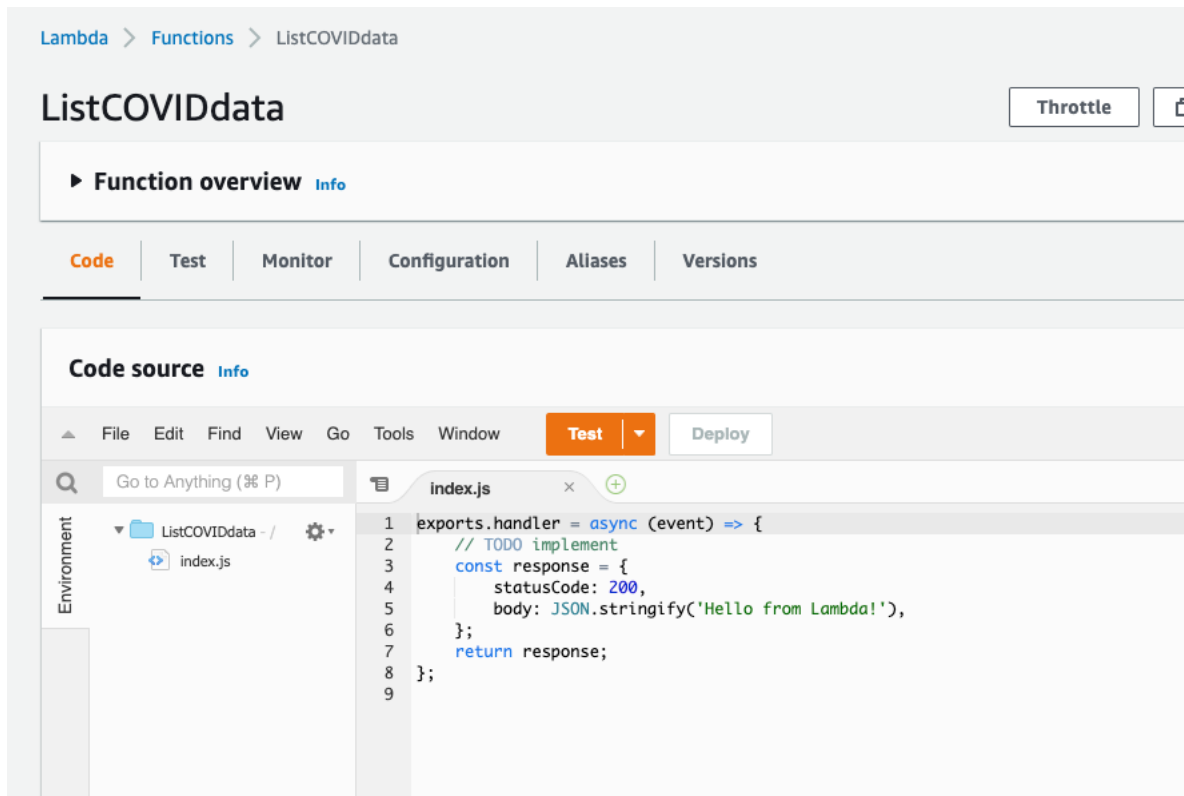
Cancel

Create function

Note in my case I went with an existing role. There's a section on setting up roles at the end of the document. I will show this later.

Click on Create Function

This is what you get:



Click Test to Test it. You will have to fill information on the Test Event:

Configure test event

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event ☐ Edit saved event

Event name

testEvent

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

And save it. Then run Test again. You should get this.

Code source

Upload from

File Edit Find View Go Tools Window Test Deploy

Go to Anything (% P)

index.js Execution result: x

Environment

- ListCOVIDdata - /
- index.js

Execution results Status: **Succeeded** Max memory used: 55 MB Time: 2.42 ms

Test Event Name
testEvent

Response

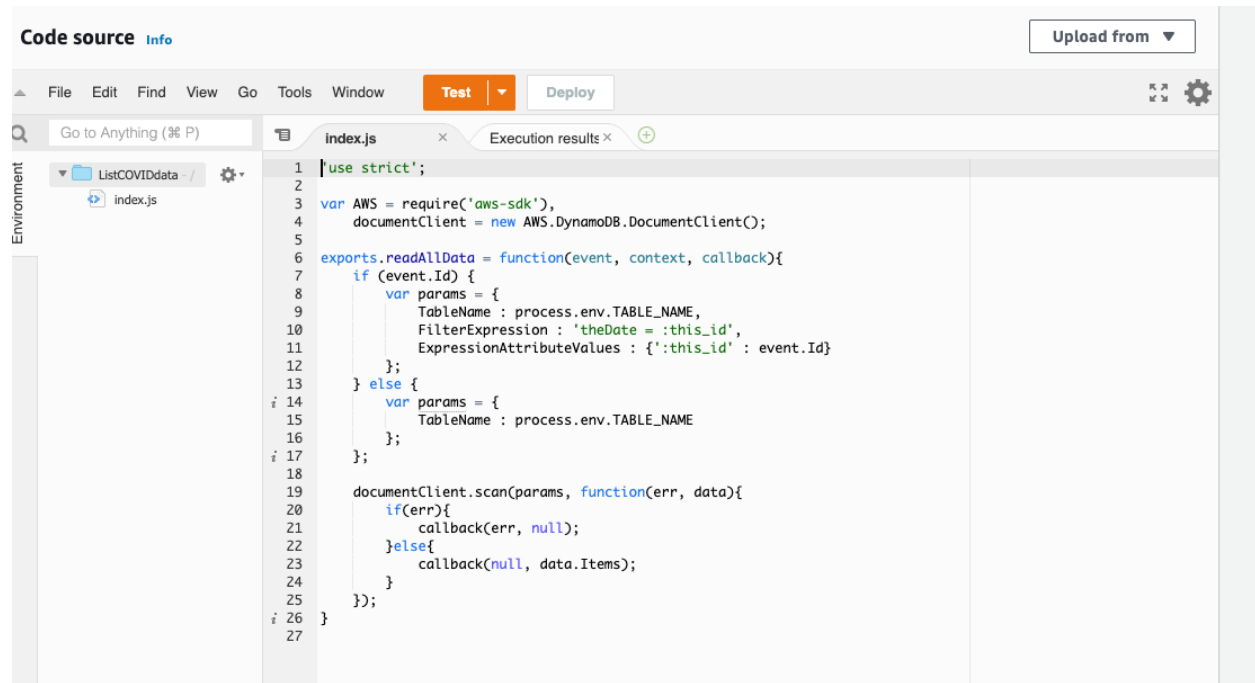
```
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}
```

Function Logs

```
START RequestId: b7a6da47-e2d1-4efe-a07b-0f80eb3cbbce Version: $LATEST
END RequestId: b7a6da47-e2d1-4efe-a07b-0f80eb3cbbce
REPORT RequestId: b7a6da47-e2d1-4efe-a07b-0f80eb3cbbce Duration: 2.42 ms Billed Duration: 3 ms Memory Size: 128 MB
```

Request ID
b7a6da47-e2d1-4efe-a07b-0f80eb3cbbce

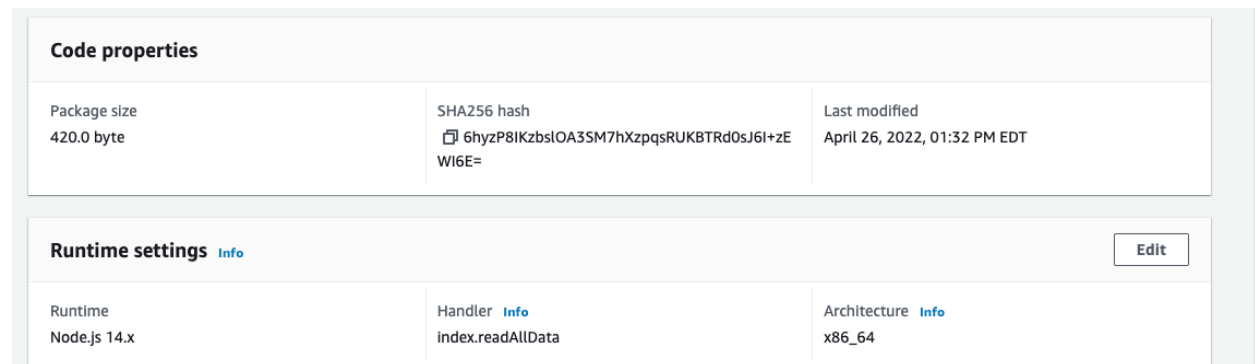
You now want to replace the code with the code below. You can find it in this repo.
After you replace the code, make sure you click Deploy



The screenshot shows the AWS Lambda console's 'Code source' tab. At the top right, there is an 'Upload from' button. Below the menu bar (File, Edit, Find, View, Go, Tools, Window), there are 'Test' and 'Deploy' buttons. The left sidebar shows the 'Environment' section with a folder 'ListCOVIDdata' and a file 'index.js'. The main area displays the code for 'index.js' with line numbers 1 through 27. The code is as follows:

```
1 'use strict';
2
3 var AWS = require('aws-sdk'),
4     documentClient = new AWS.DynamoDB.DocumentClient();
5
6 exports.readAllData = function(event, context, callback){
7     if (event.Id) {
8         var params = {
9             TableName : process.env.TABLE_NAME,
10            FilterExpression : 'theDate = :this_id',
11            ExpressionAttributeValues : {':this_id' : event.Id}
12        };
13    } else {
14        var params = {
15            TableName : process.env.TABLE_NAME
16        };
17    }
18
19    documentClient.scan(params, function(err, data){
20        if(err){
21            callback(err, null);
22        }else{
23            callback(null, data.Items);
24        }
25    });
26 }
27
```

Important thing to note. See how it says “exports.readAllData”? Scroll down this screen



The screenshot shows the 'Code properties' and 'Runtime settings' sections of the AWS Lambda console. The 'Code properties' section contains a table with the following information:

Code properties		
Package size 420.0 byte	SHA256 hash 6hyzP8IKzbslOA3SM7hXzpqSRUKBTRd0sJ6I+zE WI6E=	Last modified April 26, 2022, 01:32 PM EDT

The 'Runtime settings' section contains a table with the following information:

Runtime settings		
Runtime Node.js 14.x	Handler index.readAllData	Architecture x86_64

There is an 'Edit' button in the top right corner of the 'Runtime settings' section.

See the Handler? It needs to read: index.readAllData

Now you need to change the configuration of the function. Right now the Description is blank.

ListCOVIDdata Throttle Copy ARN Actions

► **Function overview** Info

Code Test Monitor **Configuration** Aliases Versions

General configuration Info Edit

Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout		
0 min 3 sec		

AWS Compute Optimizer
Opt in to see memory recommendations for your Lambda functions. [View details](#)

Click on the Edit button and change the Description and Save it.

Then go to Environment variables and edit them so it looks like this

Lambda > Functions > ListCOVIDdata

ListCOVIDdata Throttle Copy ARN

► **Function overview** Info

Code Test Monitor **Configuration** Aliases Versions

Environment variables (1)
The environment variables below are encrypted at rest with the default Lambda service key.

Key	Value
TABLE_NAME	CovidOntario

The case is very important here for both the Key and the Value.

Change the test to this. You can see the Event JSON is simply {} and the event name is testEvent.

Function overview [Info](#)

[Code](#) | **Test** | [Monitor](#) | [Configuration](#) | [Aliases](#) | [Versions](#)

Test event Delete Save Test

To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.

Test event action

☐ Create new event

☒ Edit saved event

Event name

testEvent ↕ ↻

Event JSON Format JSON

1 {}

Now run the test. You should see get back a JSON response with all the data in the CovidOntario table. Nice!

Code source [Info](#) Upload from ▾

File Edit Find View Go Tools Window **Test** ▾ Deploy

Go to Anything (% P)

Environment

index.js

Execution results

Status: **Succeeded** Max memory used: 78 MB Time: 635.64 ms

Test Event Name
testEvent

Response
[
 {
 "inICU": 1,
 "theDate": "04262022",
 "hospitalized": 2,
 "cases": 3
 },
 {
 "inICU": 0,
 "theDate": "0",
 "hospitalized": 0,
 "cases": 0
 }
]

Function Logs
START RequestId: c288002f-a776-410b-a8d7-faba20f589ed Version: \$LATEST
END RequestId: c288002f-a776-410b-a8d7-faba20f589ed
REPORT RequestId: c288002f-a776-410b-a8d7-faba20f589ed Duration: 635.64 ms Billed Duration: 636 ms Memory Size: 128 MB

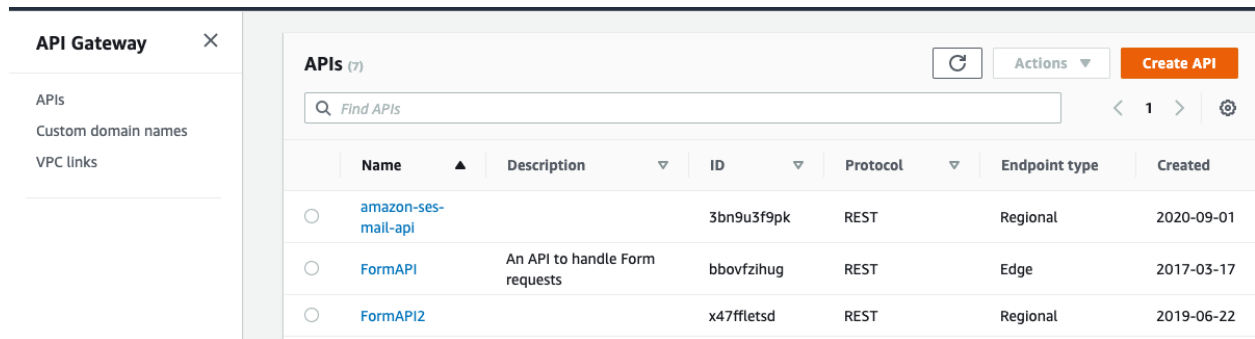
Request ID
c288002f-a776-410b-a8d7-faba20f589ed

Third, create an API to call the function. Go here:

To create an API, go here:

<https://us-east-1.console.aws.amazon.com/apigateway/main/apis?region=us-east-1>

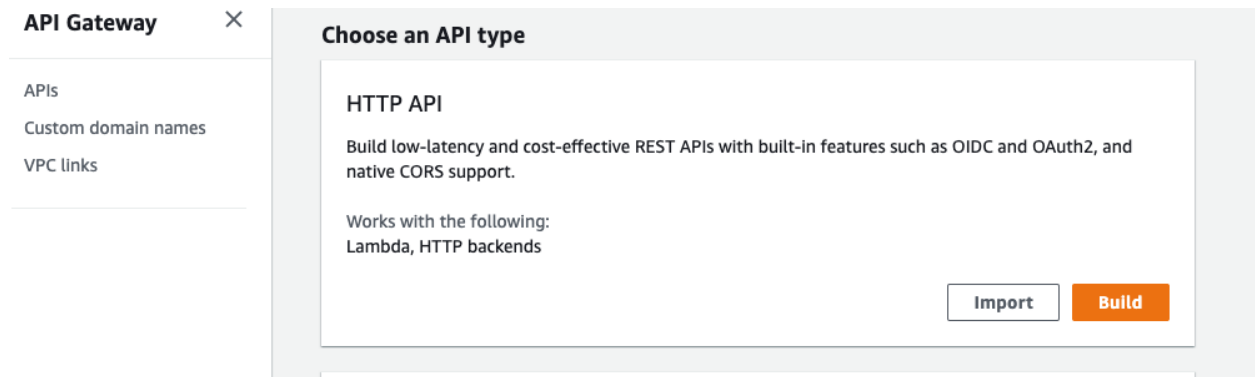
Click on the button Create API



The screenshot shows the AWS API Gateway console. On the left is a sidebar with 'API Gateway' and a search icon, and a list of links: 'APIs', 'Custom domain names', and 'VPC links'. The main panel is titled 'APIs (7)' and contains a search bar with the placeholder 'Find APIs'. Below the search bar is a table with the following columns: Name, Description, ID, Protocol, Endpoint type, and Created. The table lists three APIs: 'amazon-ses-mail-api' (ID: 3bn9u3f9pk, Protocol: REST, Endpoint type: Regional, Created: 2020-09-01), 'FormAPI' (ID: bbovfzihug, Protocol: REST, Endpoint type: Edge, Created: 2017-03-17), and 'FormAPI2' (ID: x47ffletsd, Protocol: REST, Endpoint type: Regional, Created: 2019-06-22). In the top right corner of the main panel, there are buttons for 'Create API' and 'Actions'.

Name	Description	ID	Protocol	Endpoint type	Created
amazon-ses-mail-api		3bn9u3f9pk	REST	Regional	2020-09-01
FormAPI	An API to handle Form requests	bbovfzihug	REST	Edge	2017-03-17
FormAPI2		x47ffletsd	REST	Regional	2019-06-22

Let's create a simple one for now with a type of HTTP API:



The screenshot shows the 'Choose an API type' dialog in the AWS API Gateway console. The dialog has a title 'Choose an API type' and a description: 'HTTP API. Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.' Below the description, it says 'Works with the following: Lambda, HTTP backends'. At the bottom right of the dialog are two buttons: 'Import' and 'Build'.

Choose an API type

HTTP API

Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.

Works with the following:
Lambda, HTTP backends

Import **Build**

Click on button, Add integration:

Create an API

Create and configure integrations

Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Integrations [Info](#)

Add integration

Fill out the data. You want to link to your lambda function, as follows:

Create and configure integrations

Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Integrations [Info](#)

Lambda ▼ Remove

AWS Region

us-east-1 ▼

Lambda function

×

Version [Learn more.](#)

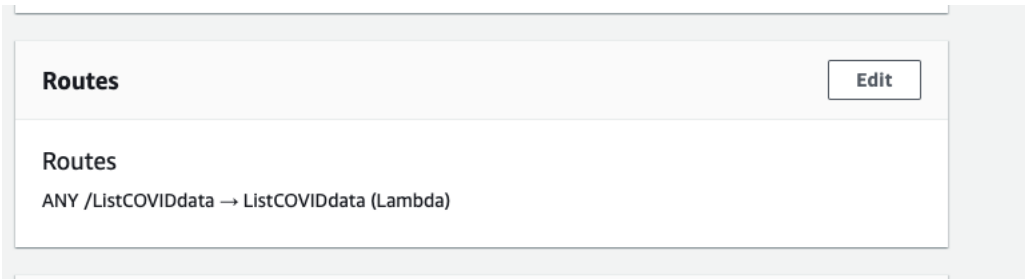
2.0 ▼

Add integration

API name
An HTTP API must have a name. This name is cosmetic and does not have to be unique; you will use the API's ID (generated later) to programmatically refer to this API.

Cancel Review and Create Next

Click Review and Create. Note the Route. You will need that later to call the Lambda function via the API.



and then Create then Deploy your API:

Stages for ListCOVIDdataAPI				
<input type="text" value="Find resources"/>				
Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
\$default	https://qpx025ptl3.execute-api.us-east-1.amazonaws.com	5nm6u7	enabled	2022-04-26

Tags (0)

Note the invoke URL.

If you combine the invoke URL with the route, you can access the function. For example, you can curl it like this:

```
{ "message": "Not Found" }
berniemichalik@Bernies-MacBook-Air charts % curl https://qpx025ptl3.execute-api.us-east-1.amazonaws.com/ListCOVIDdata
[{"inICU":1,"theDate":"04262022","hospitalized":2,"cases":3},{"inICU":0,"theDate":"0","hospitalized":0,"cases":0}]
berniemichalik@Bernies-MacBook-Air charts %
```

Or you can type that into your browser

You will get back the data in the DynamoDB table in the form of JSON output.

If you forgot the route, you could go here and check it.

API Gateway

×

APIs

Custom domain names

VPC links

API: ListCOVIDdataAP...
(qpx025ptl3)

Develop

Routes

Authorization

Integrations

CORS

Reimport

Export

Deploy

Stages

Protect

Successfully created API ListCOVIDdataAPI (qpx025ptl3)

API Gateway > Routes

Routes

Routes for ListCOVIDdataAPI

Create

Search

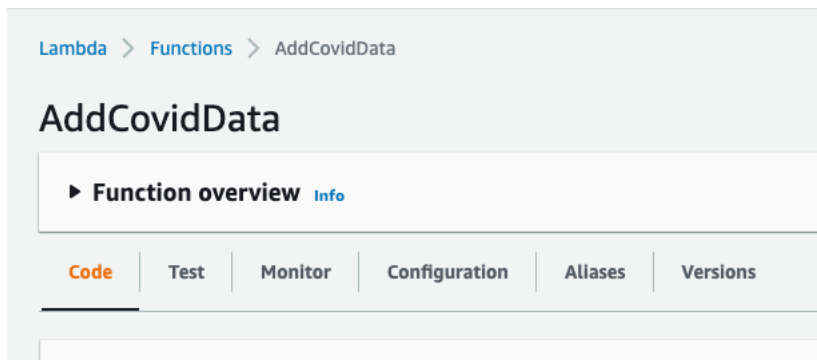
▼ /ListCOVIDdata

ANY

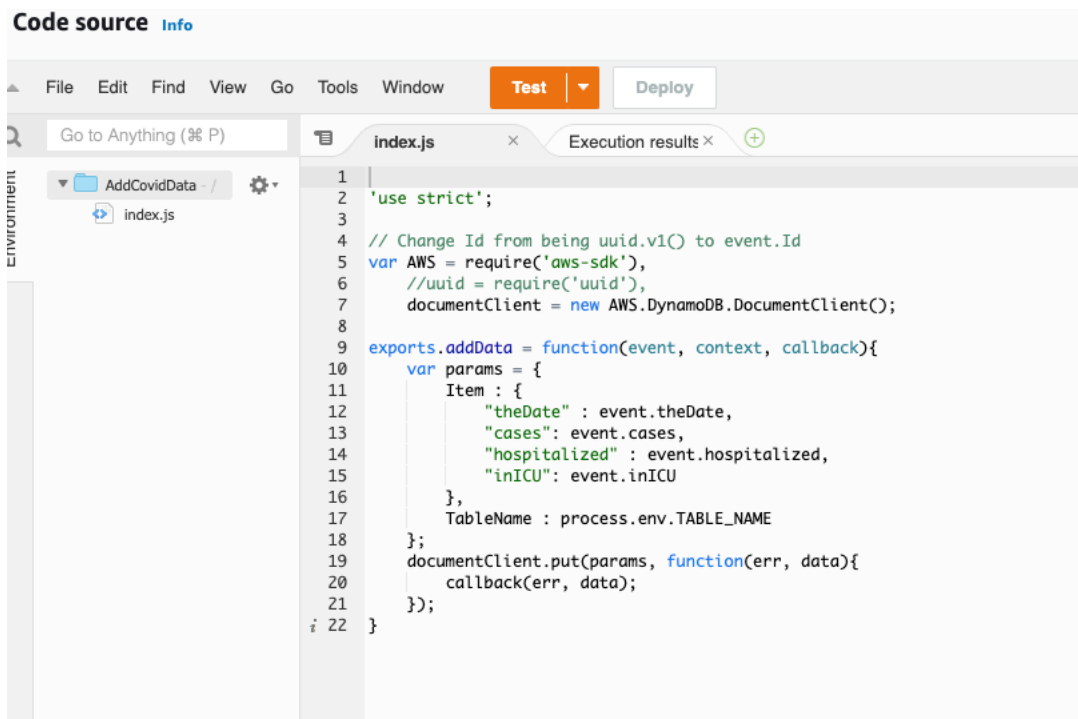
Create a second function to add data to the database

Now that you can list data in the database via our API, you may want to a function to add to it. Here's how you do that.

Here is the function name you will use for this example:



Here is the source code. Note the names of the values in Item



Make sure we set the handler properly:

Runtime settings Info			Edit
Runtime Node.js 14.x	Handler Info Index.addData	Architecture Info x86_64	

Put the table name in the environment variables:

Code	Test	Monitor	Configuration	Aliases	Versions				
<div>General configuration</div> <div>Triggers</div> <div>Permissions</div> <div>Destinations</div> <div>Function URL - <i>new</i></div> <div>Environment variables</div>									
<h3>Environment variables (1)</h3> <p>The environment variables below are encrypted at rest with the default Lambda service key.</p> <table><thead><tr><th>Key</th><th>Value</th></tr></thead><tbody><tr><td>TABLE_NAME</td><td>CovidOntario</td></tr></tbody></table>						Key	Value	TABLE_NAME	CovidOntario
Key	Value								
TABLE_NAME	CovidOntario								

Create a test event. Note the data in the Event JSON. It's no longer {}.

► Function overview Info					
Code	Test	Monitor	Configuration	Aliases	Versions
<h3>Test event</h3> <div>Delete Save Test</div> <p>To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.</p> <p>Test event action</p> <div><input type="radio"/> Create new event</div> <div><input checked="" type="radio"/> Edit saved event</div> <p>Event name</p> <div>addTestData ▼ ↺</div> <div><h4>Event JSON</h4><div>Format JSON</div><pre>1 { 2 "theDate": "04272022", 3 "cases": "10", 4 "hospitalized": "100", 5 "inICU": "50" 6 }</pre></div>					

Run the test:

The screenshot shows a code editor interface with a menu bar (File, Edit, Find, View, Go, Tools, Window) and buttons for 'Test' and 'Deploy'. The 'Test' button is highlighted. Below the menu bar, there's a search bar and a file explorer showing 'AddCovidData' and 'index.js'. The main area displays the 'Execution results' for 'index.js'. The status is 'Succeeded', with 'Max memory used: 78 MB' and 'Time: 604.36 ms'. The 'Test Event Name' is 'addTestData'. The 'Response' is an empty object '{}'. The 'Function Logs' show the start and end of the request, and the 'Request ID' is 'a3b28ad2-b0aa-40dc-8224-05c3e3790035'.

Voila! The test JSON data that you used above is now in the table:

The screenshot shows a table with 4 columns: 'theDate', 'cases', 'hospitalized', and 'inICU'. There are 3 rows of data. The first row has '04262022', '3', '2', and '1'. The second row has '04272022', '10', '100', and '50'. The third row has '0', '0', '0', and '0'. The table is titled 'Items returned (3)' and has a 'Read capacity units consumed: 0.5' status.

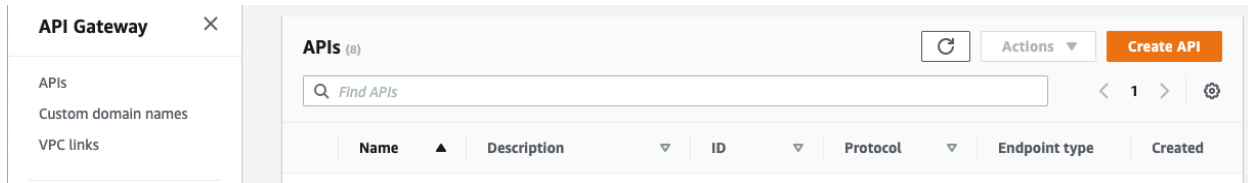
	theDate	cases	hospitalized	inICU
<input type="checkbox"/>	04262022	3	2	1
<input type="checkbox"/>	04272022	10	100	50
<input type="checkbox"/>	0	0	0	0

If we curl the data, we now get the new information:

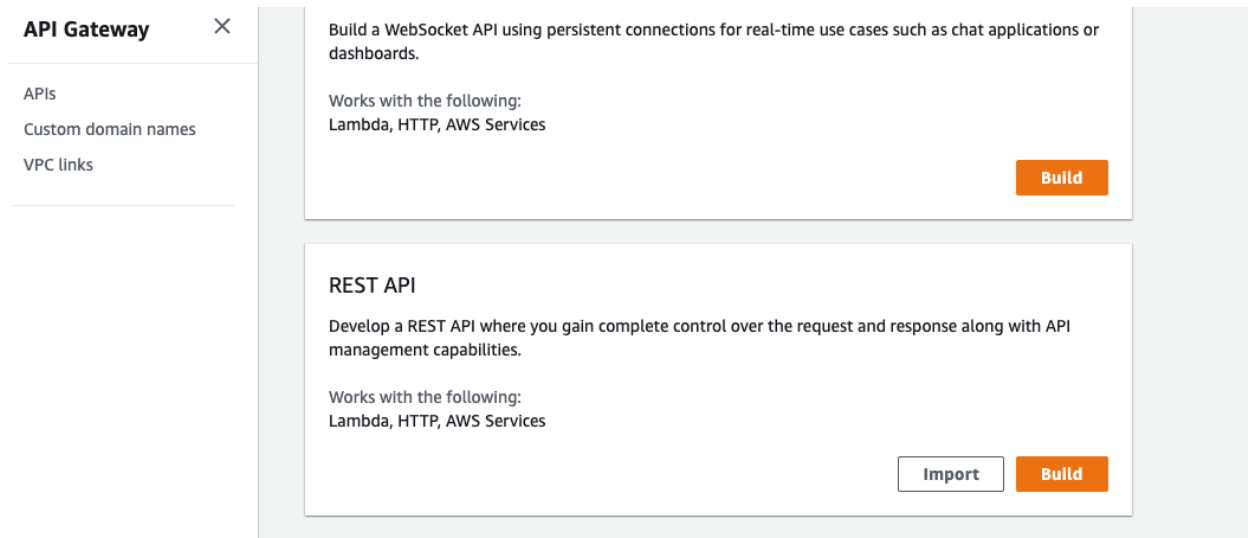
```
berniemichalik@Bernies-MacBook-Air charts % curl https://qpx025pt13.execute-api.us-east-1.amazonaws.com/ListCOVIDdata
[{"inICU":1,"theDate":"04262022","hospitalized":2,"cases":3}, {"inICU":50,"theDate":"04272022","hospitalized":100,"cases":10}, {"inICU":0,"theDate":"0","hospitalized":0,"cases":0}]
berniemichalik@Bernies-MacBook-Air charts %
```

Create a second API to invoke the second function

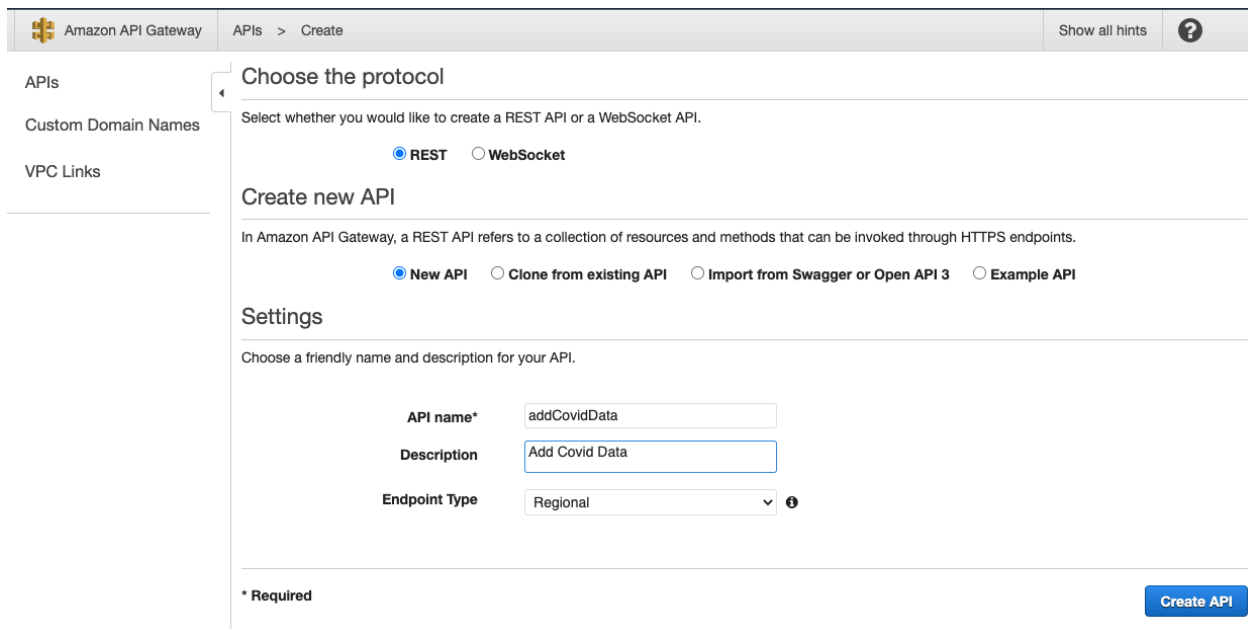
Now that we have the second function, we are going to create a new API to access it. Click Create API:



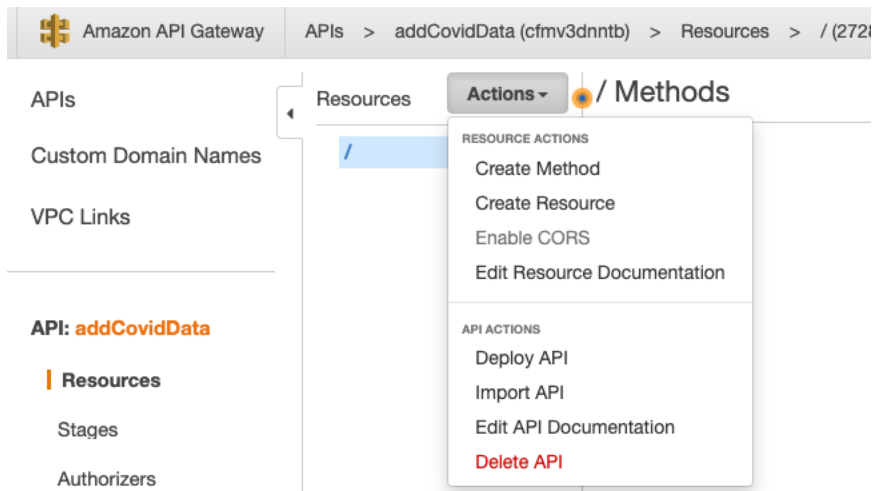
This time you are going to go with REST API, not HTTP API. Click Build in the REST API box:



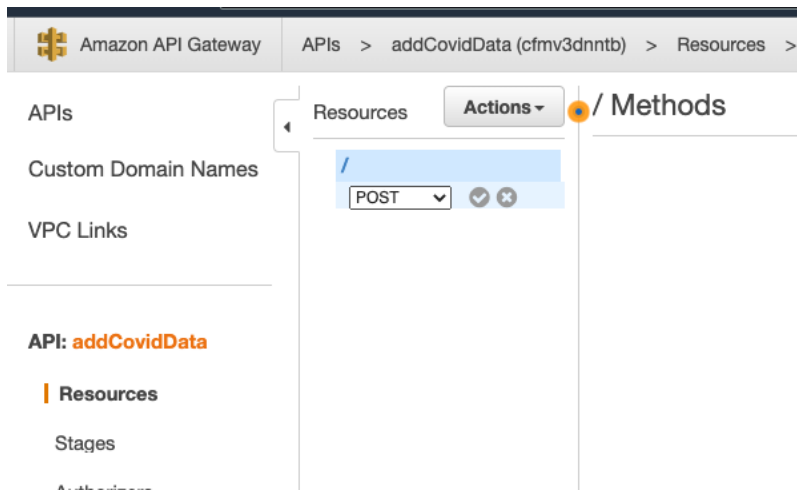
Stick with the defaults, but add the API name, description. Click Create API:



Now you are going to click on Actions > Create Method:



The method we want is POST



We want to integrate with Lambda Function “AddCovidData”. Make sure the values are as you see here and click Save.

The screenshot shows the AWS API Gateway console. The breadcrumb trail is: Amazon API Gateway > APIs > addCovidData (cfmv3dnntb) > Resources > / (2728m0eo3h) > POST. The left sidebar shows the API name 'addCovidData' and the 'Resources' section. The main panel is titled '/ - POST - Setup' and contains the following configuration:

- Integration type:** ☒ Lambda Function ⓘ
☐ HTTP ⓘ
☐ Mock ⓘ
☐ AWS Service ⓘ
☐ VPC Link ⓘ
- Use Lambda Proxy integration:** ☐ ⓘ
- Lambda Region:** us-east-1
- Lambda Function:** AddCovidData ⓘ
- Use Default Timeout:** ☒ ⓘ

A blue 'Save' button is located at the bottom right of the configuration panel.

Click OK for this window:

The screenshot shows a dialog box titled 'Add Permission to Lambda Function'. The text inside reads: 'You are about to give API Gateway permission to invoke your Lambda function: arn:aws:lambda:us-east-1:606955541489:function:AddCovidData'. At the bottom right, there are 'Cancel' and 'OK' buttons.

To test your API, click on Test in the box labelled Client (with the lightning bolt):

Amazon API Gateway

APIs > addCovidData (cfmv3dnntb) > Resources > / (2728m0eo3h) > POST

Show all hints ?

APIs

Custom Domain Names

VPC Links

API: **addCovidData**

Resources

Stages

Authorizers

Gateway Responses

Models

Resource Policy

Documentation

Dashboard

Settings

Usage Plans

Resources

POST

Actions

POST - Method Execution

Client

Method Request

Auth: NONE

ARN: arn:aws:execute-api:us-east-1:606955541489:cfmv3dnntb/*/*

Integration Request

Type: LAMBDA

Region: us-east-1

Integration Response

HTTP status pattern: -

Output passthrough: Yes

Method Response

HTTP Status: 200

Models: application/json => Empty

Lambda AddCovidData

9 Scroll data and enter some data. (It's similar to the JSON data used elsewhere). Then click on Test:

Stage Variables

No stage variables exist for this method.

Request Body

```
1 {  
2   "theDate": "999",  
3   "cases": "99",  
4   "hospitalized": "19990",  
5   "inICU": "51"  
6 }
```



Go check the database. You should see the JSON test data above in your database:

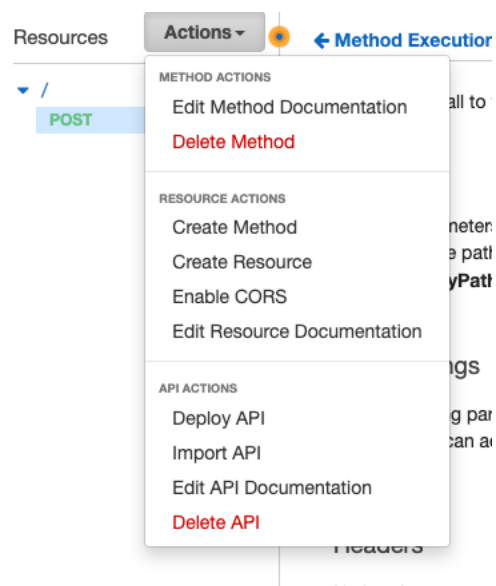
✔ **Completed** Read capacity units consumed: 0.5

Items returned (6)

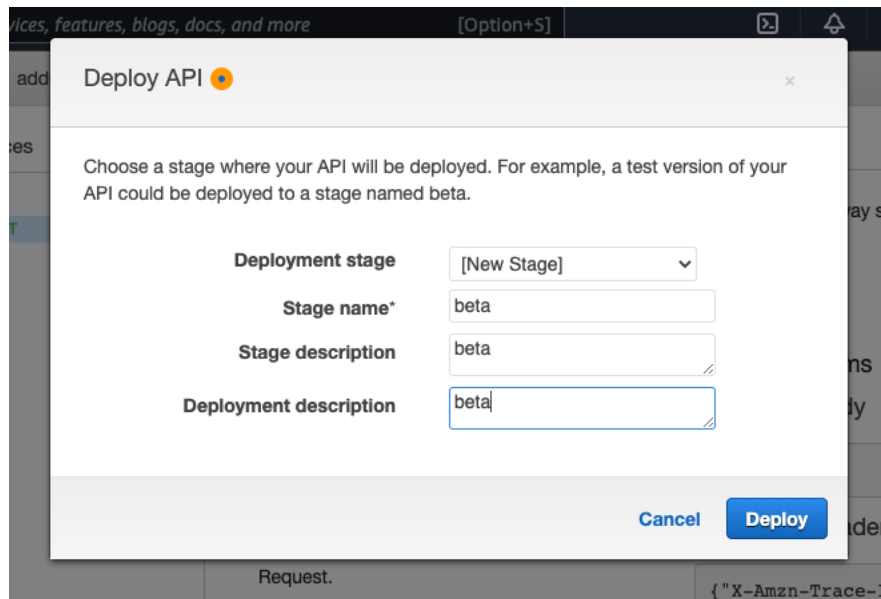
< 1 > ⚙️ 🔍

<input type="checkbox"/>	theDate ▼	cases ▼	hospitali... ▼	inICU
<input type="checkbox"/>	999	99	19900	51
<input type="checkbox"/>	04262022	3	2	1
<input type="checkbox"/>	04272022	10	100	50
<input type="checkbox"/>	9	10	100	51
<input type="checkbox"/>	0	0	0	0
<input type="checkbox"/>	10	10	100	51

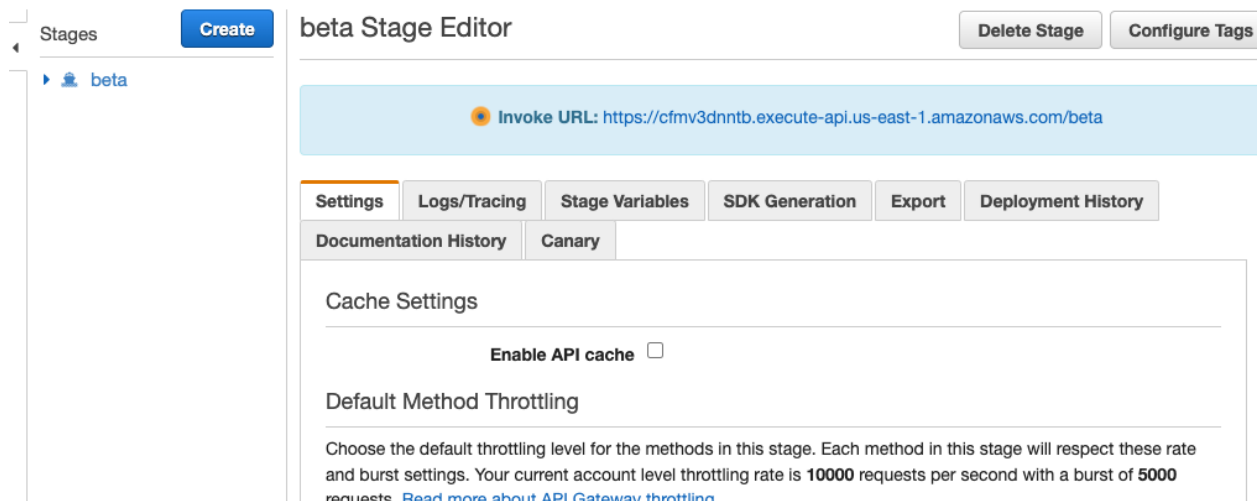
Now go back and click on Actions > Deploy API:



You need to assign a stage. It can be anything “test”, “alpha”, “beta”, “v1” etc. I went with beta:



After I clicked on Deploy, I got this. Copy the Invoke URL:



Now you want to make this curl command. Note the data we are going to pass: it's the JSON data you've used before. Also note the invoke URL with the "beta" in it:

```
curl -X POST -H "Content-Type: application/json"
-d '{"theDate": "11", "cases": "11", "hospitalized": "11", "inICU": "11"}'
https://cfmv3dnntb.execute-api.us-east-1.amazonaws.com/beta
```

Make the curl command a single line and run it. You get this!


```
berniemichalik@Bernies-MacBook-Air charts % curl -X POST -H "Content-Type: application/json" -d '{"theDate": "11", "cases": "11", "hospitalized": "11", "inICU": "11"}' https://cfmv3dnntb.execute-api.us-east-1.amazonaws.com/beta
{}
berniemichalik@Bernies-MacBook-Air charts %
```

Did it work? Check the database: at the bottom is your new record.

Items returned (7)				
< 1 > ⚙️ 🔍				
<input type="checkbox"/>	theDate ▾	cases ▾	hospitali... ▾	inICU
<input type="checkbox"/>	999	99	19900	51
<input type="checkbox"/>	04262022	3	2	1
<input type="checkbox"/>	04272022	10	100	50
<input type="checkbox"/>	9	10	100	51
<input type="checkbox"/>	0	0	0	0
<input type="checkbox"/>	11	11	11	11

Roles

Most of my Lambda functions work with a role I created with the name:
mySimpleMicroservicpermissions (I know, it has a typo). It looks like this:

[Permissions](#) | [Trust relationships](#) | [Tags](#) | [Access Advisor](#) | [Revoke sessions](#)

Permissions policies (5)
You can attach up to 10 managed policies.

Simulate

Remove

Add permissions ▼

< 1 >

<input type="checkbox"/>	Policy name	Type	
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-b6af6298-4932-41af-97b9-8075ae8e92a2	Customer managed	
<input type="checkbox"/>	AWSLambdaMicroserviceExecutionRole-db517260-2148-43a3-9eca-93166990c40c	Customer managed	
<input type="checkbox"/>	AmazonDynamoDBFullAccess	AWS managed	
<input type="checkbox"/>	CloudWatchLogsFullAccess	AWS managed	
<input type="checkbox"/>	CloudWatchEventsFullAccess	AWS managed	

Permissions boundary - (not set)
Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting but can be used to delegate

The first two items in the Policy name list were autogenerated. The third I added to write to the DynamoDB tables. And the last two were likely for some logging I was doing. It does the job.

When I create a Lambda function, it will ask me for a role. I use that one. You can make a similar one before you start making lambda functions. Or you can also create your own when you make a lambda function, like this

Edit basic settings

Basic settings [Info](#)

Description - *optional*

Add Covid data to CovidOntario

Memory [Info](#)

Your function is allocated CPU proportional to the memory configured.

128

MB

Set memory to between 128 MB and 10240 MB

Ephemeral storage [Info](#)

You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)

512

MB

Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

Timeout

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

For example, for one function I created this:

blmDefaultRole

Delete

Summary

Edit

Creation date

February 27, 2017, 15:41 (UTC-05:00)

ARN

arn:aws:iam::606955541489:role/service-role/blmDefaultRole

Last activity

50 minutes ago

Maximum session duration

1 hour

Permissions

Trust relationships

Tags

Access Advisor

Revoke sessions

Permissions policies (1)

You can attach up to 10 managed policies.

Refresh

Simulate

Remove

Add permissions

Filter policies by property or policy name and press enter

< 1 >

Settings

<input type="checkbox"/>	Policy name	Type	Description
<input type="checkbox"/>	<div><div>+</div><div>AWSLambdaBasicExecutionRole-79ef3d1b-182a-4443-9af2-3a6a2dda8c...</div></div>	Customer managed	

It's fine for running Lambda functions, but I can't get it to work with DynamoDB. So, I created this role and it works because I added an additional policy:

IAM > Roles > myTestRole2

myTestRole2

Delete

Summary

Edit

Creation date

April 26, 2022, 16:54 (UTC-04:00)

ARN

arn:aws:iam::606955541489:role/service-role/myTestRole2

Last activity

None

Maximum session duration

1 hour

Permissions

Trust relationships

Tags

Access Advisor

Revoke sessions

Permissions policies (2)

You can attach up to 10 managed policies.



Simulate

Remove

Add permissions ▼

Filter policies by property or policy name and press enter

< 1 >



<input type="checkbox"/>	Policy name	Type	Description
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-36296822-74e2...	Customer managed	
<input type="checkbox"/>	AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazc

When it comes to creating roles, use the KISS principle. Keep the number of roles limited and the number of policies limited until your code works. I have one role that works for all my Lambda code. That may work for you, or you may need more.