Create API Gateway for Backend Operation

(LAB-M07-01)

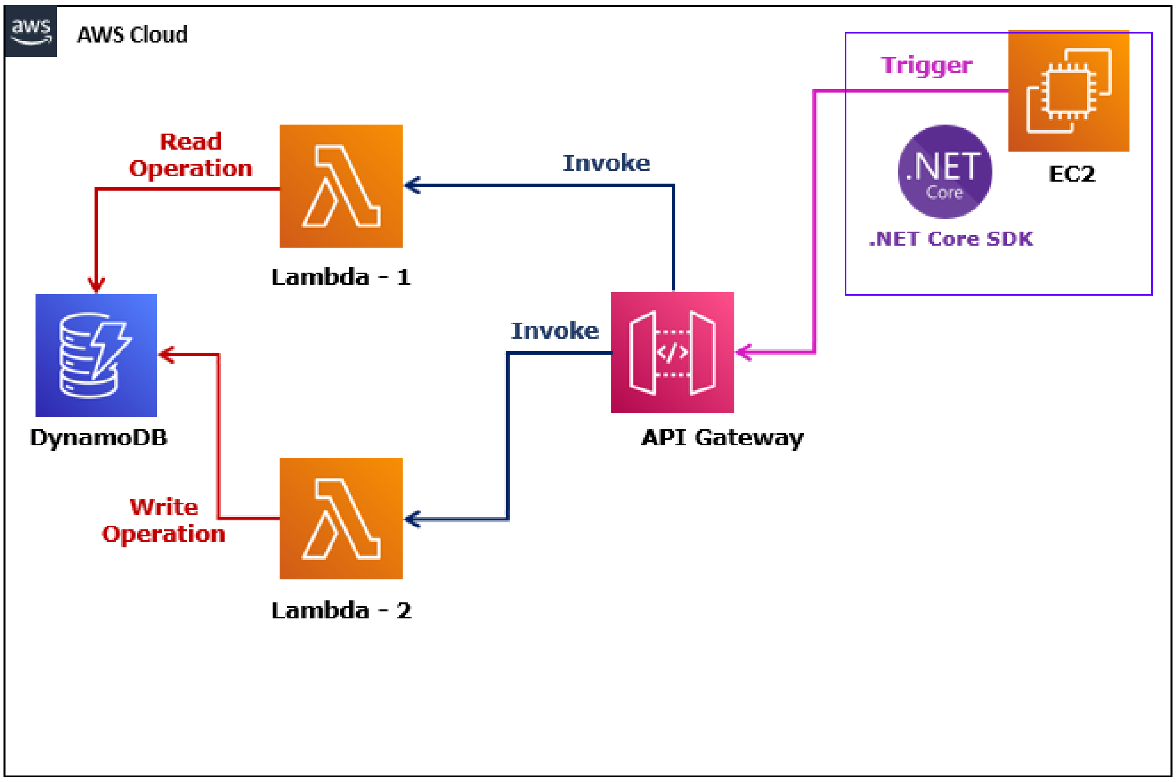
Lab scenario

In this lab, you will learn how to use AWS Lambda to trigger a Lambda function and update the DynamoDB. You will also integrate the Lambda function with API gateway and trigger a Lambda function via API Gateway.

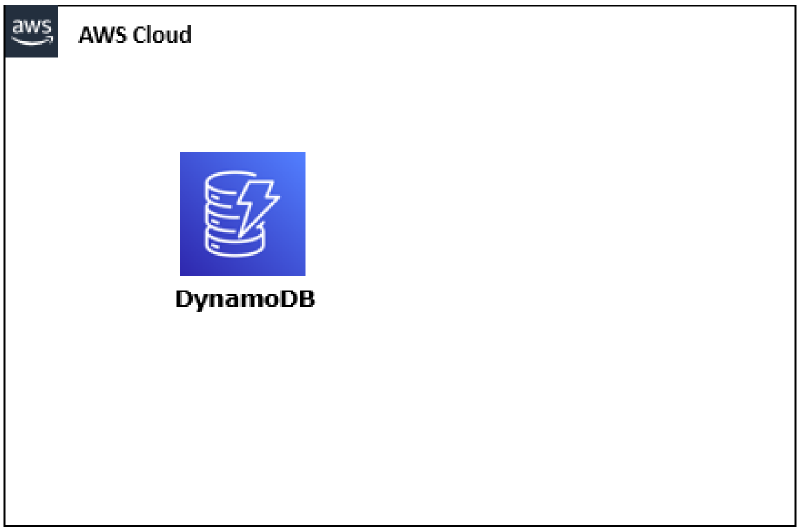
Objectives

After you complete this lab, you will be able to:

- Create new Lambda function.
- Add the data in the DynamoDB.
- Integrate the Lambda function with API gateway.



Task 1: Create Database



Step 1: Create a DynamoDB Table

1. Choose the **US East (N. Virginia)** region list to the right of your account information on the navigation bar.
2. In the **AWS Management Console**, on the **Services** menu, click **DynamoDB**.
3. Choose **Create Table** and **Configure**:
   a. **Table name**: Write **empdata**.
   b. **Primary key**: Write **empid**.
      i. Set the data type to **String**.

Note: **Write** the table name and primary key in the **lower case** only.

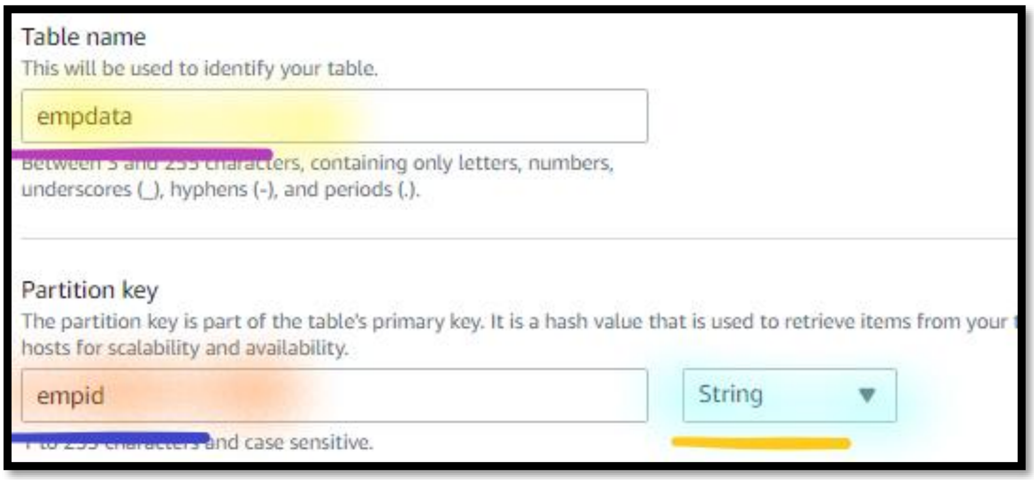


Table name
This will be used to identify your table.
empdata
Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

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 t
hosts for scalability and availability.
empid String
1 to 255 characters and case sensitive.

   c. Select **Use default settings** under **Settings**.

d. Select **Create**.

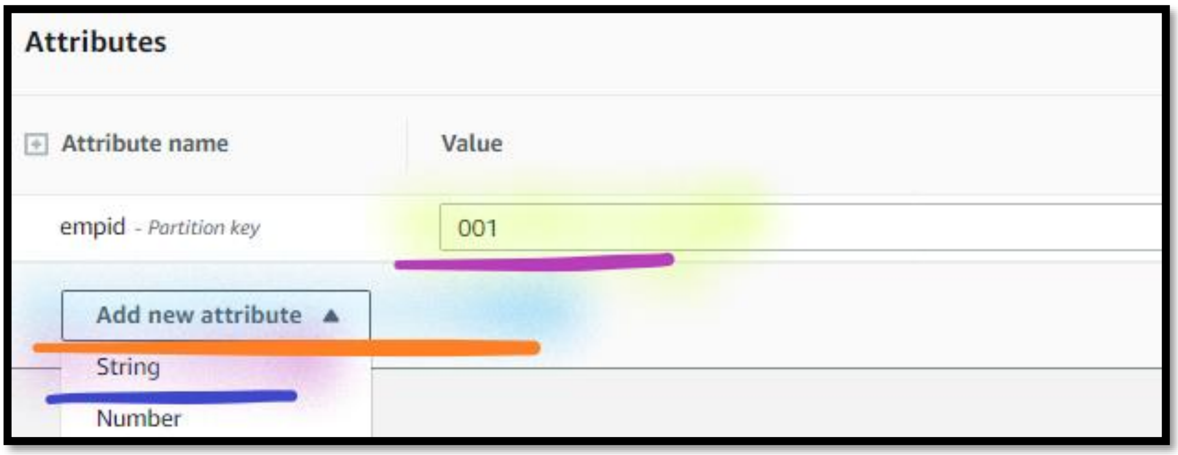
Note: **Wait** till DynamoDB table gets **created**.

Step 2: Add Items into DynamoDB Table

4. In the **AWS Management Console**, on the **Services** menu, click **DynamoDB**.
5. Select **Items**.
   a. Select **empdata**.
   b. Select **Create Item**.

Note: New window gets opened to entered the items details from UI.

a. **empid**: Write **001** (***in value field***).
   i. Select **Add new attribute**.
   ii. Select **String**.



b. **Attribute name**: Write **empfirstname**.
   i. **Value**: Write **Ajay**.
      1) Select **Add new attribute**.
      2) Select **String**.

c. **Attribute name**: Write **emplastname**.
   i. **Value**: Write **Kaushik**.

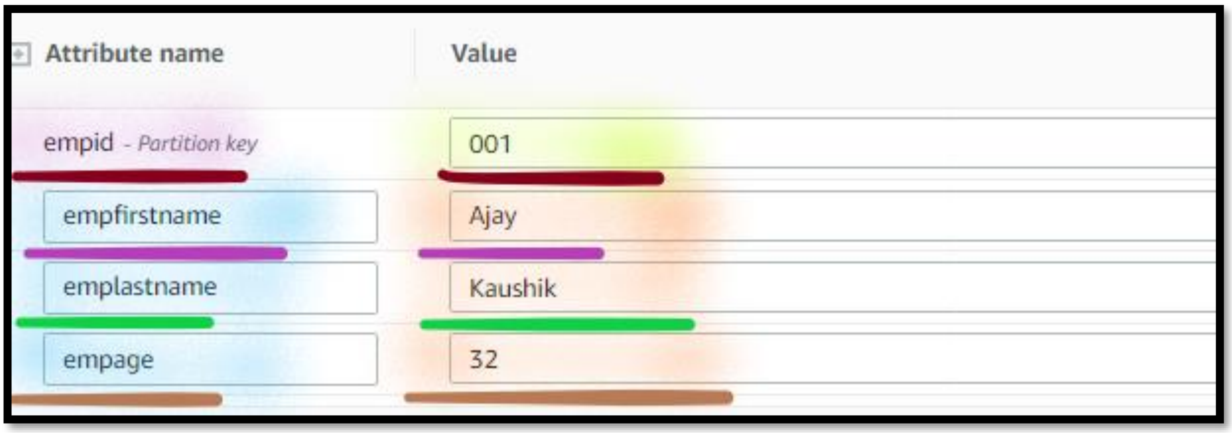
1) Select **Add new attribute**.

2) Select **String**.

d. **Attribute name**: Write **empage**.

i. **Value**: Write **32**.

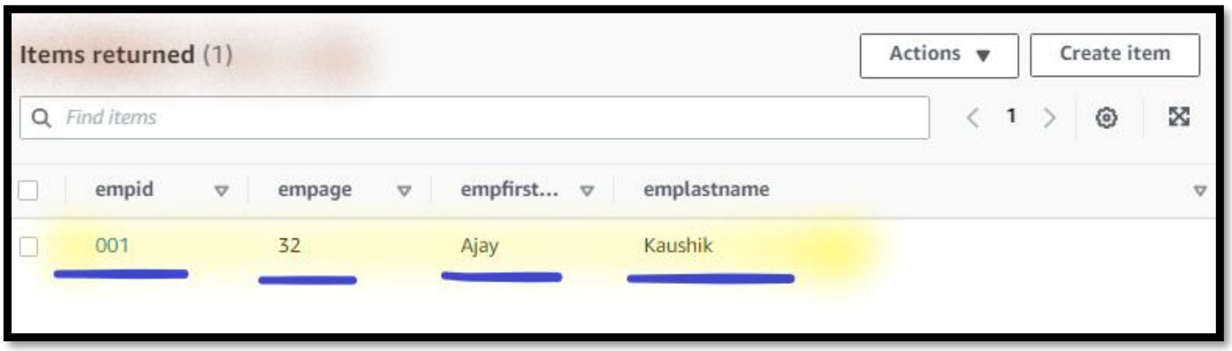
Note: **Write** the **empfirstname**, **emplastname** and **empage** in the **lower case** only.



Attribute name	Value
empid - *Partition key*	001
empfirstname	Ajay
emplastname	Kaushik
empage	32

e. Select **Create Item**.

Note: You can view the newly created item details under Items.



Items returned (1)

empid	empage	empfirst...	emplastname
001	32	Ajay	Kaushik

Task 2: Create IAM Role

Step 1: Create IAM Role

6. In the **AWS Management Console**, on the **Services** menu, click **IAM**.

7. Select **Roles**, click on **Create role**.

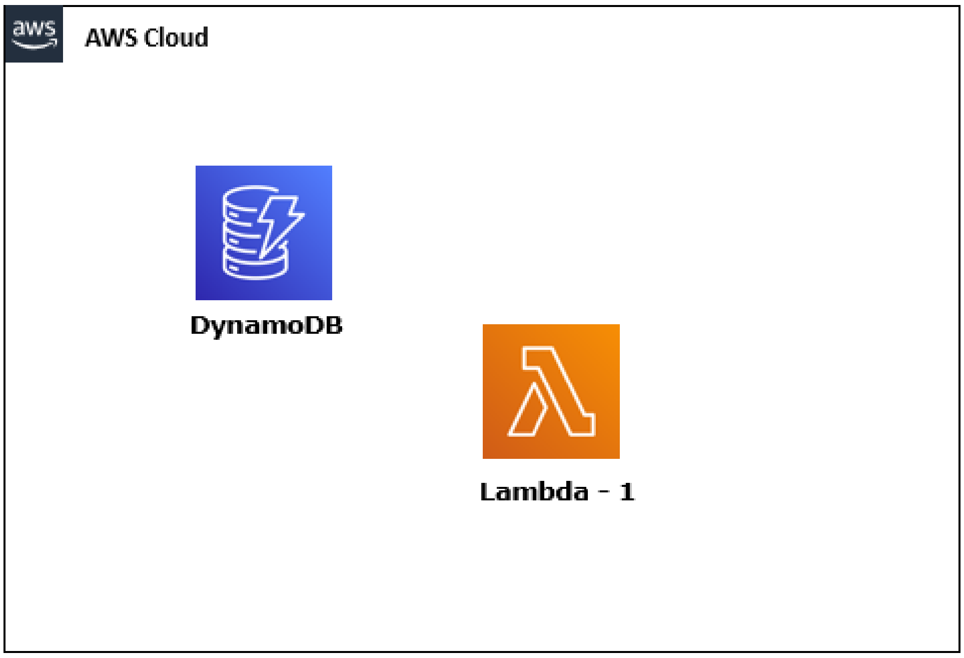
8. Select **Lambda**, under a **use case**.
   a. Select **Next: Permissions**.
      i. Search and Select **AmazonDynamoDBFullAccess**.
      ii. Search and Select **AWSLambdaBasicExecutionRole**.
   b. Select **Next: Tags**.
   c. Select **Next: Review**.

Note: Here you will see **DynamoDB Policy** under policies.

      i. **Role name**: Write **Lambda-DynamoDB-Role**.
   d. Click **Create role**.

Note: You get the message, the role **Lambda-DynamoDB-Role** been created.

Task 3: Create Lambda Function to Read the Items



Step 1: Create Lambda Function to Read the Items

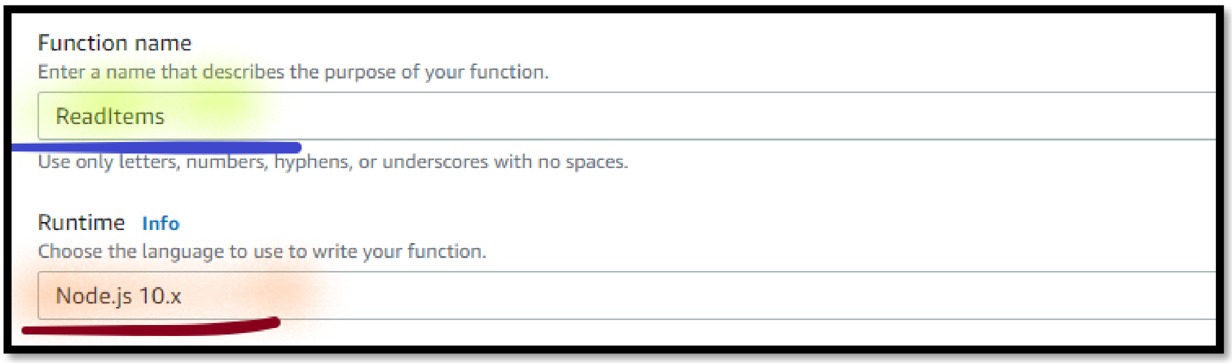
9. In the **AWS Management Console**, on the **Services** menu, click **Lambda**.

10. Click **Create a function**.

11. Select **Author from scratch** and configure:

    a. **Name**: Write **ReadItems**.

    b. **Runtime**: Dropdown and Select **Node.js [*Latest version*]**.



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

Runtime Info
Choose the language to use to write your function.
Node.js 10.x

    c. **Expand** **Change default execution role**.

    d. **Role**: Select **Use an existing role**.

        i. **Existing role**: Dropdown and Select **Lambda-DynamoDB-Role**.

    e. Select **Create function**.

Note: ReadItems function gets open the configuration section.

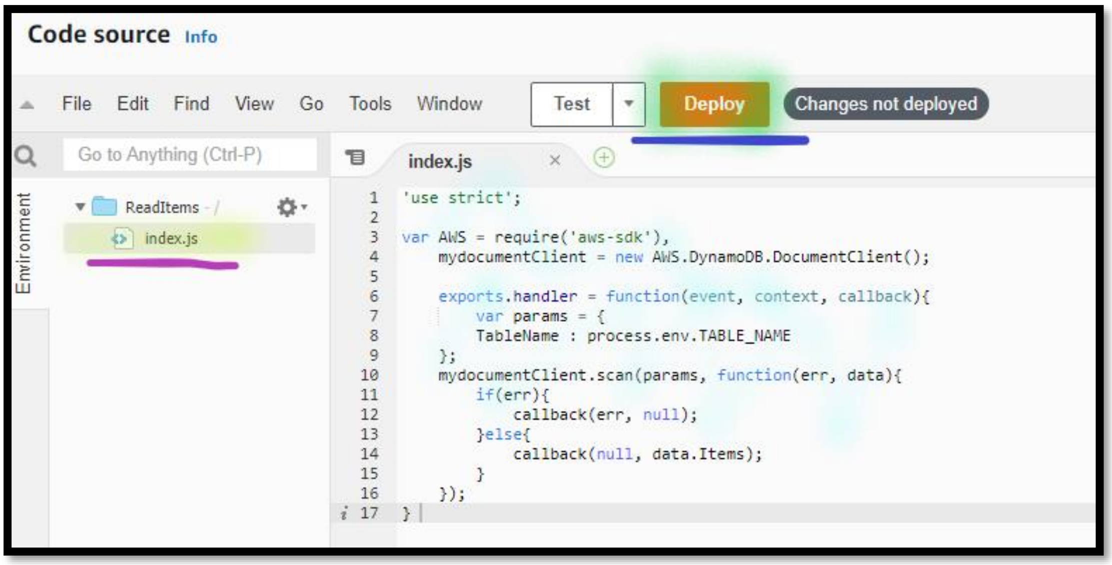
12. **Select** the **Code** section:

    a. Click on **index.js**.

        i. **Replace** the ***existing code*** and **copy** the code into the **editor** from **Read-Items-Lambda-Function-Code**.txt file.

Note: **Code** for **Read-Items-Lambda-Function-Code.txt** is available with the Lab manual.

    b. Select **Deploy**.

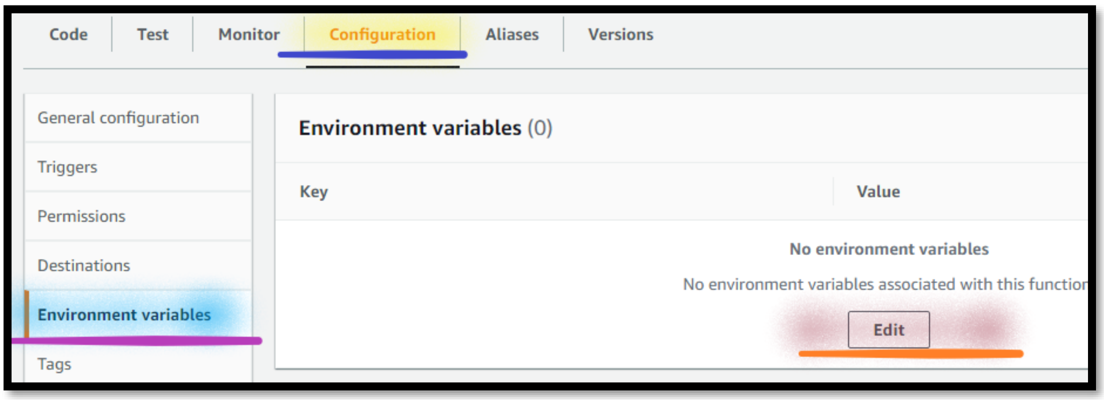


```
'use strict';

var AWS = require('aws-sdk'),
    mydocumentClient = new AWS.DynamoDB.DocumentClient();

    exports.handler = function(event, context, callback){
        var params = {
        TableName : process.env.TABLE_NAME
    };
    mydocumentClient.scan(params, function(err, data){
        if(err){
            callback(err, null);
        }else{
            callback(null, data.Items);
        }
    });
}
```

13. Select the Configuration section:

   a. Select **Environment variables**.

   b. Select **Edit**.



Environment variables (0)

Key	Value

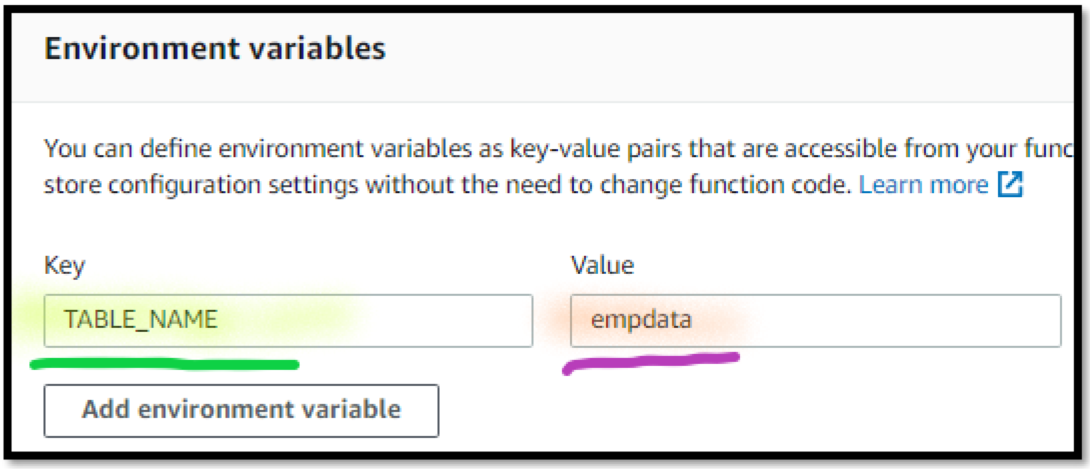
No environment variables

No environment variables associated with this function

   c. Select **Add environment variables**.

      i. **Key**: Write **TABLE_NAME**.

      ii. **Value**: Write **empdata** (DynamoDB table name).



Environment variables

You can define environment variables as key-value pairs that are accessible from your func
store configuration settings without the need to change function code. Learn more

Key	Value
TABLE_NAME	empdata

d. Select **Save**.

Step 2: Validate Your Implementation

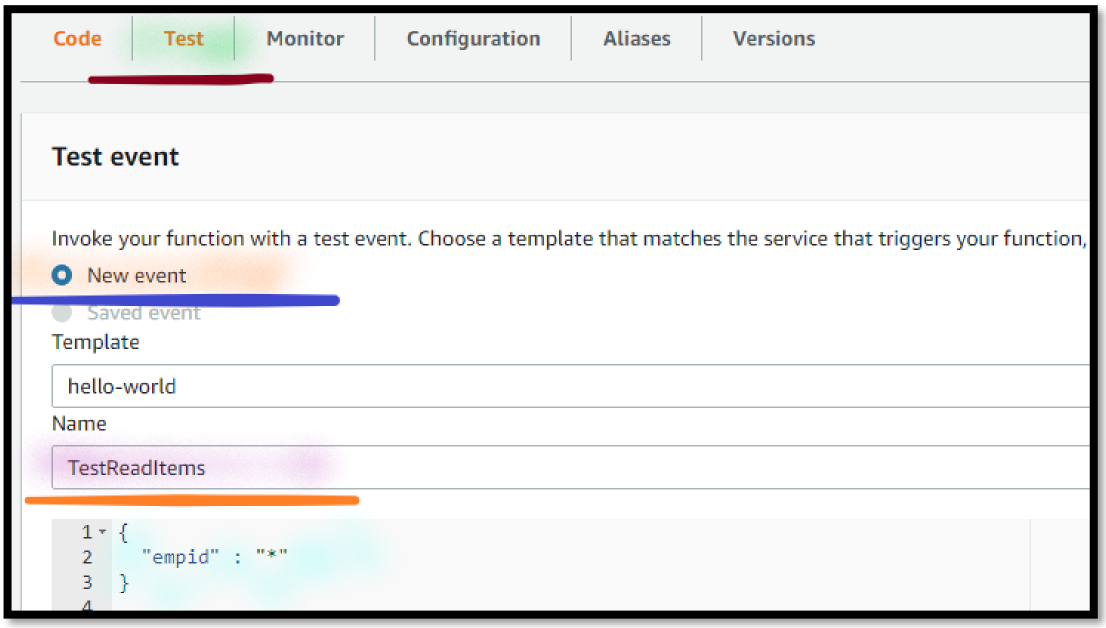
14. **Select** the **Test** section:

a. **Template**: Dropdown and Select **hello-world**.

b. **Name**: Write **TestReadItems**.

c. In the **Event**, **Remove** the existing events and **copy** the below event:

```
{
  "empid": "*"
}
```



Test event

Invoke your function with a test event. Choose a template that matches the service that triggers your function,

New event

Saved event

Template: hello-world

Name: TestReadItems

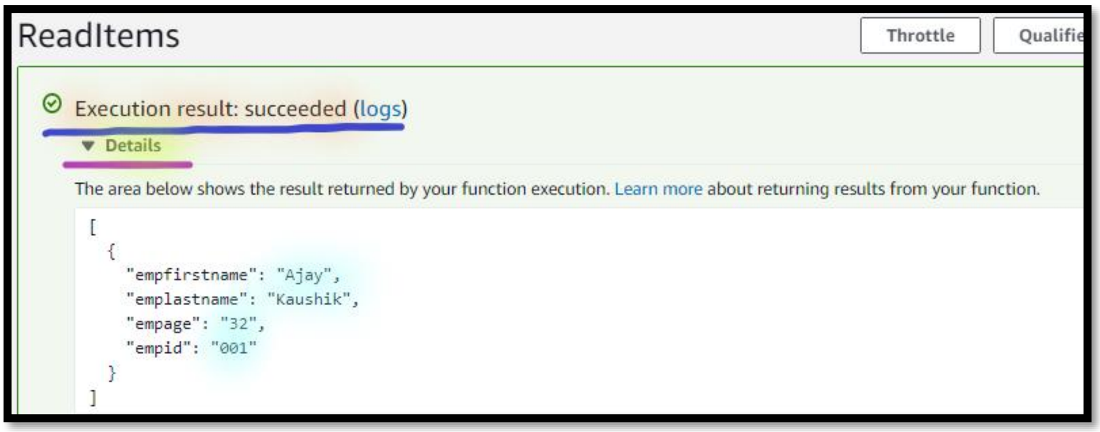
```
{
  "empid" : "*"
}
```

d. Select **Test**.

Note: Once you Test the function and code executed succesfully you can see the execution result as **succeeded**.

15. **Expand** the **Details** section of the **execution result** section.

Note: You can view the **Items**, which you have added in the DynamoDB in the Previous Step.



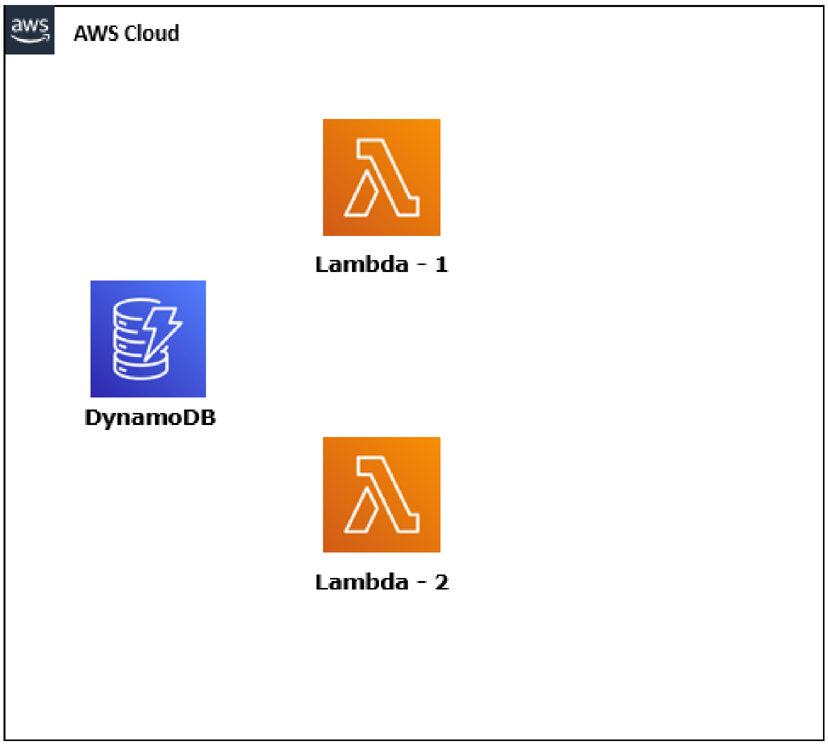
Execution result: succeeded (logs)

Details

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

```
[
  {
    "empfirstname": "Ajay",
    "emplastname": "Kaushik",
    "empage": "32",
    "empid": "001"
  }
]
```

Task 4: Create Lambda Function to Write the Items



Step 1: Create Lambda Function to Write the Items

16. In the **AWS Management Console**, on the **Services** menu, click **Lambda**.
17. Click **Create a function**.
18. Select **Author from scratch** and configure:
    a. **Name**: Write **WriteItems**.
    b. **Runtime**: Dropdown and Select **Node.js [*Latest version*]**.
    c. **Expand** **Change default execution role**.
    d. **Role**: Select **Use an existing role**.
        i. **Existing role**: Dropdown and Select **Lambda-DynamoDB-Role**.
    e. Select **Create function**.

Note: WriteItems function gets open the configuration section.

19. **Select** the **Configuration** section:
    a. Select **Environment variables**.
    b. Select **Edit**.
    c. Select **Add environment variables**.
        i. **Key**: Write **TABLE_NAME**.
        ii. **Value**: Write **empdata** (DynamoDB table name).
    d. Select **Save**.
20. Select the **Code** section:
    a. Click on **index.js**.
        i. **Replace** the ***existing code*** and **copy** the code into the **editor** from **Write-Items-Lambda-Function-Code**.txt file.

Note: **Code** for **Write-Items-Lambda-Function-Code.txt** is available with the Lab manual.

b. Select **Deploy**.

Step 2: Validate Your Implementation

21. **Select** the **Test** section:

    a. **Template**: Dropdown and Select **hello-world**.

    b. **Name**: Write **TestWriteItems**.

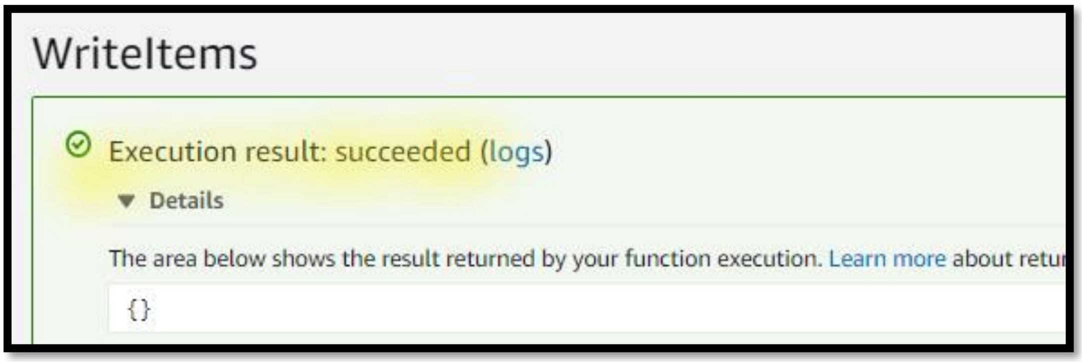
    c. In the **Event**, **Remove** the existing events and **copy** the below event:

Note: You can add the new items now.

```
{
  "empid": "002",
  "empfirstname": "Sana",
  "emplastname": "Yusuf",
  "empage": "21"
}
```

    d. Select **Test**.

Note: Once you Test the function and code executed succesfully you can see the Execution result as **succeeded**.


WriteItems

Execution result: succeeded (logs)

Details

The area below shows the result returned by your function execution. Learn more about retur

{}

Step 3: View the DynamoDB Data

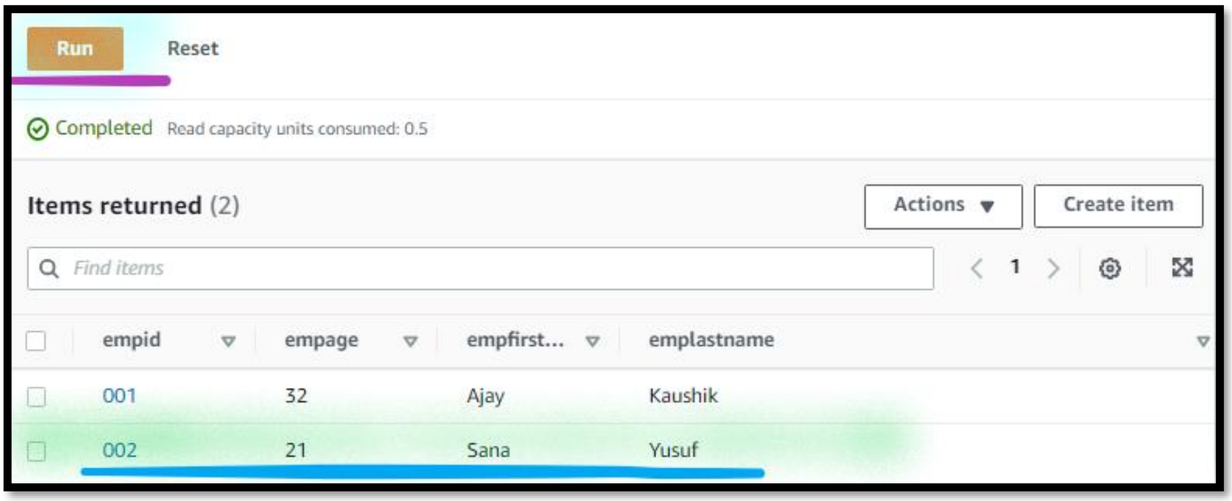
22. In the **AWS Management Console**, on the **Services** menu, click **DynamoDB**.

23. Select **Items**.

a. Select **empdata** DynamoDB table.

b. Select **Run**.

Note: You can view the **Added Items**, which you have added in the DynamoDB via the **Lambda function**.

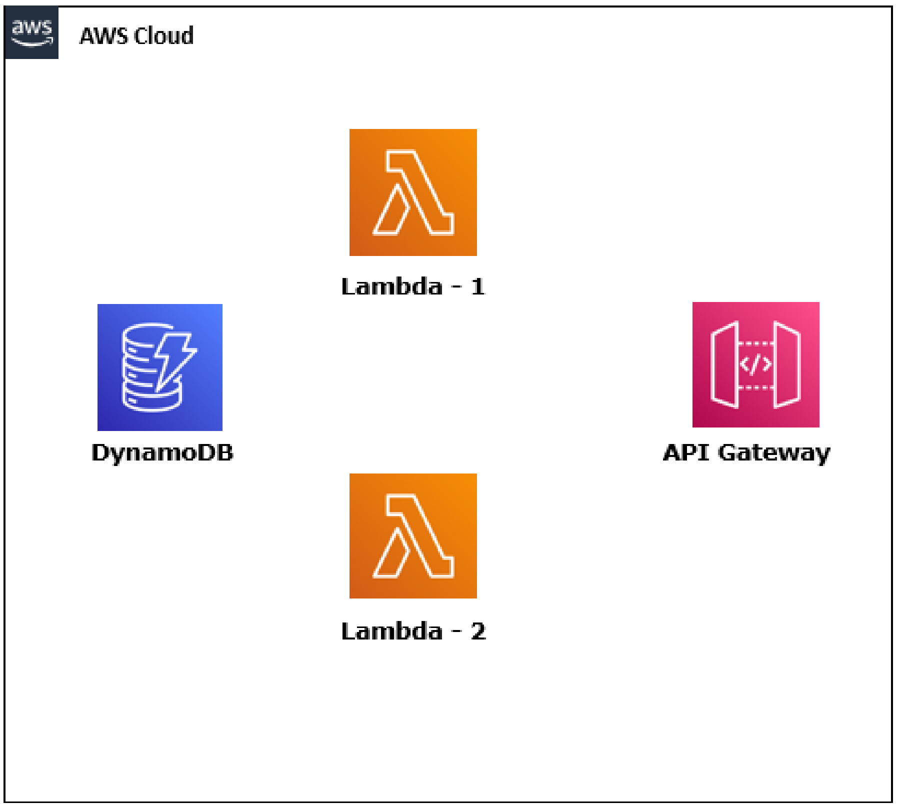


Completed Read capacity units consumed: 0.5

Items returned (2)

empid	empage	empfirst...	emplastname
001	32	Ajay	Kaushik
002	21	Sana	Yusuf

Task 5: Create a RESTful API



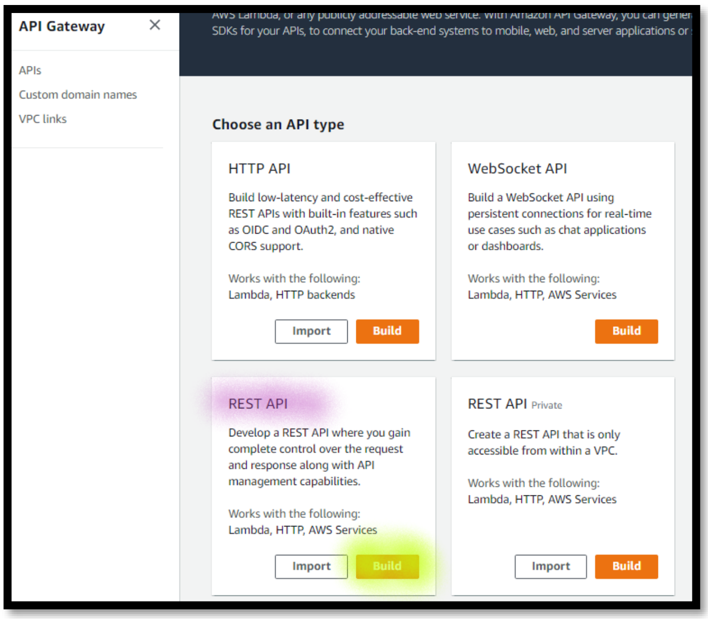
Step 1: Create REST API

24. In the **AWS Management Console**, on the **Services** menu, click **API Gateway**.

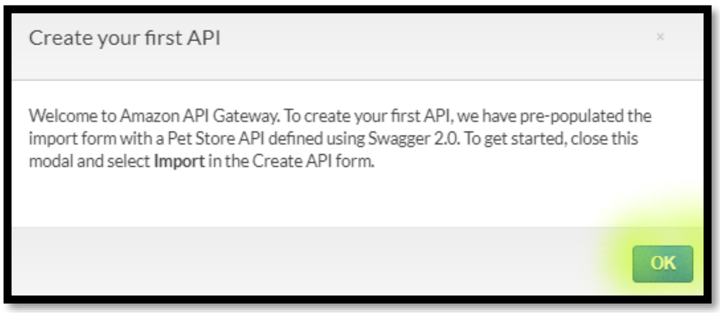
25. Choose the **US East (N. Virginia)** region list to the right of your account information on the navigation bar.

26. Select **Rest API** (***Don't select Rest API private***).

27. Select **Build**.


API Gateway
APIs
Custom domain names
VPC links
AWS Lambda, or any publicly addressable web service. With Amazon API Gateway, you can gener
SDKs for your APIs, to connect your back-end systems to mobile, web, and server applications or
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
WebSocket API
Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.
Works with the following: Lambda, HTTP, AWS Services
Build
REST API
Develop a REST API where you gain complete control over the request and response along with API management capabilities.
Works with the following: Lambda, HTTP, AWS Services
Import
Build
REST API Private
Create a REST API that is only accessible from within a VPC.
Works with the following: Lambda, HTTP, AWS Services
Import
Build

28. Select **OK**, when **Create your first API** window prompt.



Create your first API

Welcome to Amazon API Gateway. To create your first API, we have pre-populated the import form with a Pet Store API defined using Swagger 2.0. To get started, close this modal and select **Import** in the Create API form.

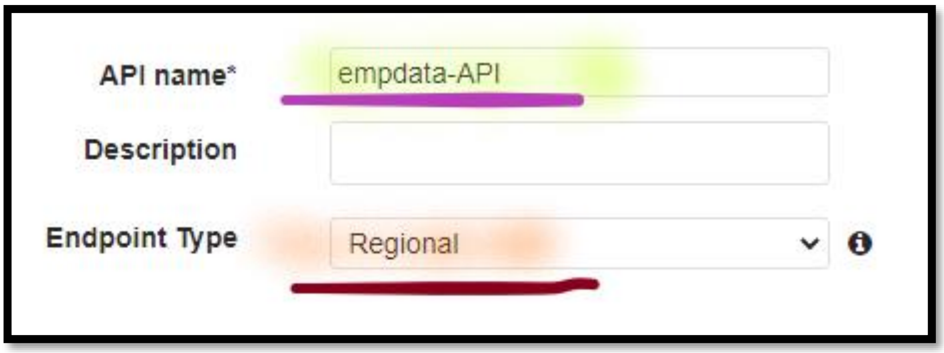
OK

29. Select **New API** under ***Create new API***.

30. In the **Settings**, provide the following:

    a. **API name**: Write **empdata-API**.

    b. **Endpoint type**: Dropdown and select **Regional**.


API name* empdata-API

Description

Endpoint Type Regional

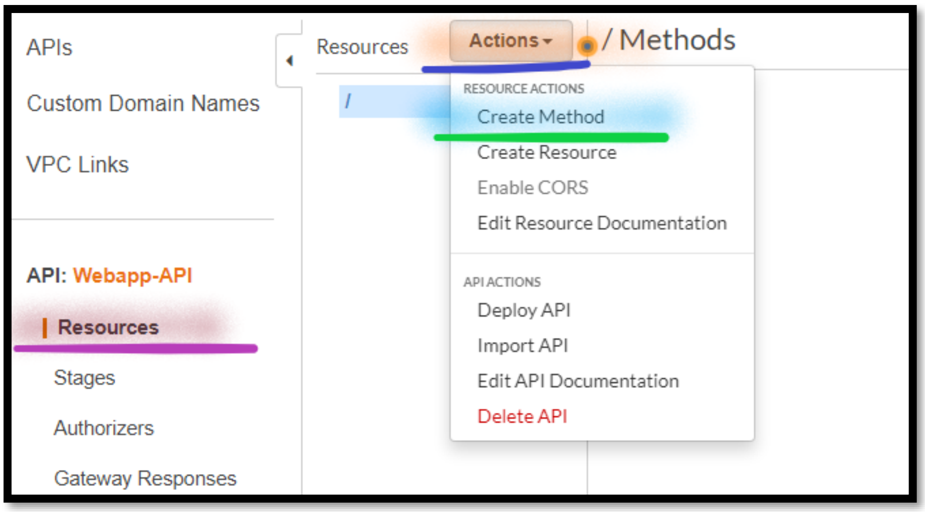
c. Select **Create API**.

Task 6: Create API Method to Read the Data

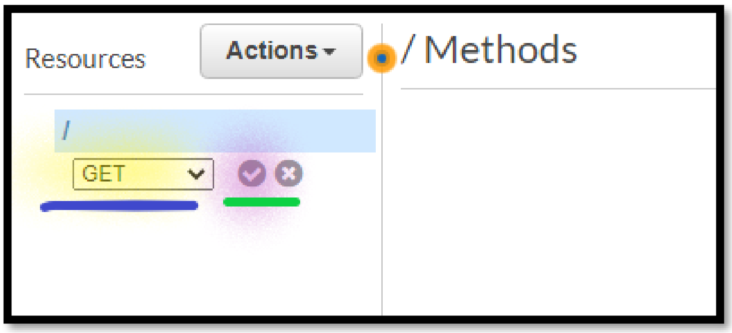
Step 1: Create Method to Read the Items

31. **Go to left**, choose **Resources**.

a. From the **Actions** dropdown select **Create Method**.



b. Select **Get** from the new dropdown that appears, then click the **checkmark**.

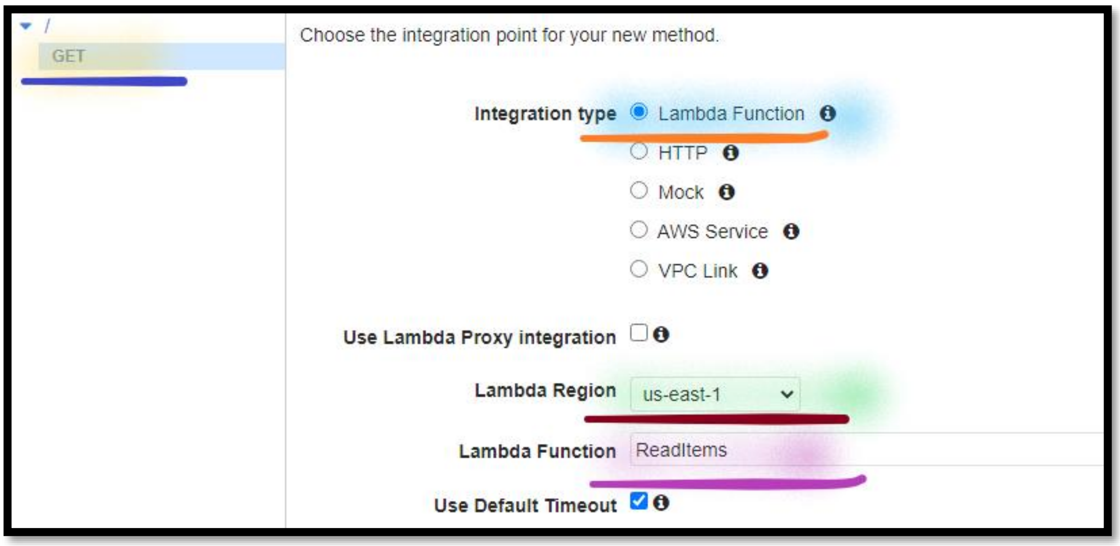


c. **Integration type**: Select **Lambda Function**.

d. **Lambda region**: Dropdown and Select the **us-east-1** region.

e. **Lambda function**: Write **ReadItems**, the lambda function you created in the previous lab, that read the data from DynamoDB table.

Note: Leave other details as default.



Choose the integration point for your new method.

Integration type [✓ Lambda Function] HTTP Mock AWS Service VPC Link

Use Lambda Proxy integration ☐

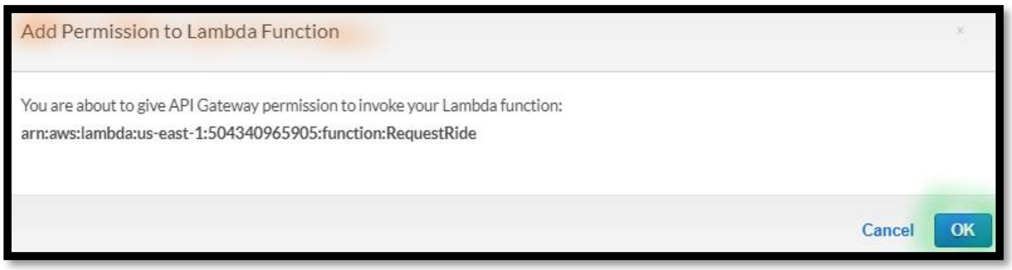
Lambda Region: us-east-1

Lambda Function: ReadItems

Use Default Timeout ✓

f. Select **Save**.

g. When **prompted** to give Amazon API Gateway permission to invoke your function, choose **OK**.

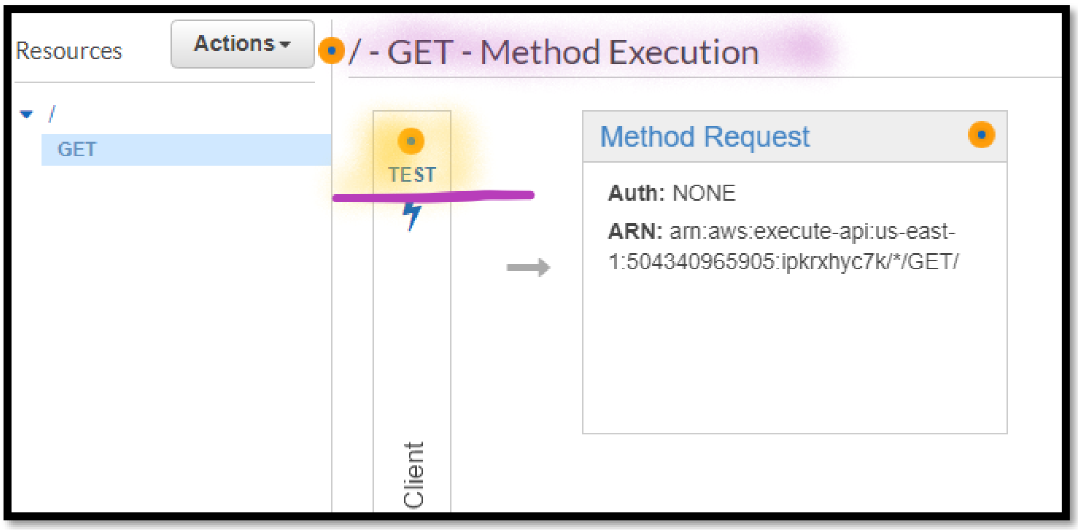


Add Permission to Lambda Function

You are about to give API Gateway permission to invoke your Lambda function:
arn:aws:lambda:us-east-1:504340965905:function:RequestRide

Step 2: Test the API Gateway to Read the Items

32.Click on **Test** under ***GET - Method Execution***.



Auth: NONE

ARN: arn:aws:execute-api:us-east-1:504340965905:ipkrxhyc7k/*/GET/

a. Click the **Test**.

Note: If request executed **succesfully**, you can see the Request status as **200**.

Note: In the Response body, you can see the **Items**, which you have added in the DynamoDB in the previous step.

Task 7: Create API Method to Write the Data

Step 1: Create Method to Write the Items

33.**Go to left**, choose **Resources**.

34.From the **Actions** dropdown select **Create Method**.

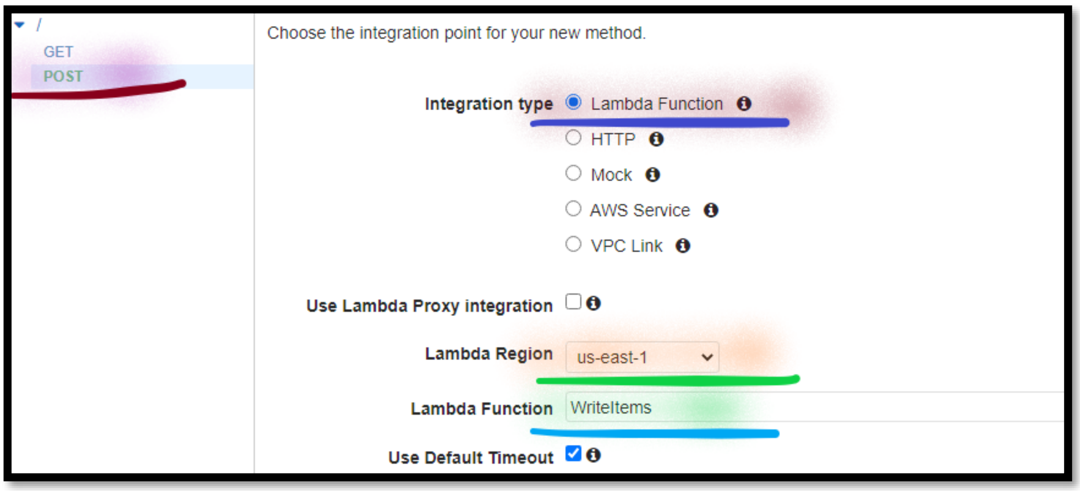
35.Select **Post** from the new dropdown that appears, then click the **checkmark**.

a. **Integration type**: Select **Lambda Function**.

b. **Lambda region**: Dropdown and Select the **us-east-1** region.

c. **Lambda function**: Write **WriteItems**, the lambda function you created in the previous lab, that write the data into DynamoDB table.

Note: Leave other details as default.



Choose the integration point for your new method.

Integration type: (● Lambda Function) ○ HTTP ○ Mock ○ AWS Service ○ VPC Link

Use Lambda Proxy integration ☐

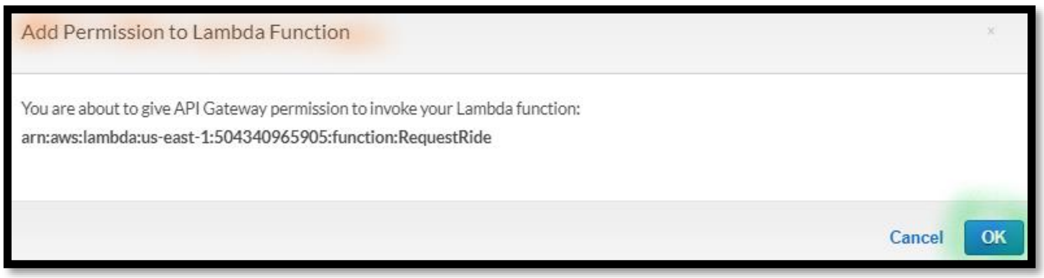
Lambda Region: us-east-1

Lambda Function: WriteItems

Use Default Timeout ☑

d. Select **Save**.

e. When **prompted** to give Amazon API Gateway permission to invoke your function, choose **OK**.



Add Permission to Lambda Function

You are about to give API Gateway permission to invoke your Lambda function:
arn:aws:lambda:us-east-1:504340965905:function:RequestRide

Step 2: Test the API Gateway to Write the Items

36.Click on **Test** under ***Post - Method Execution***.

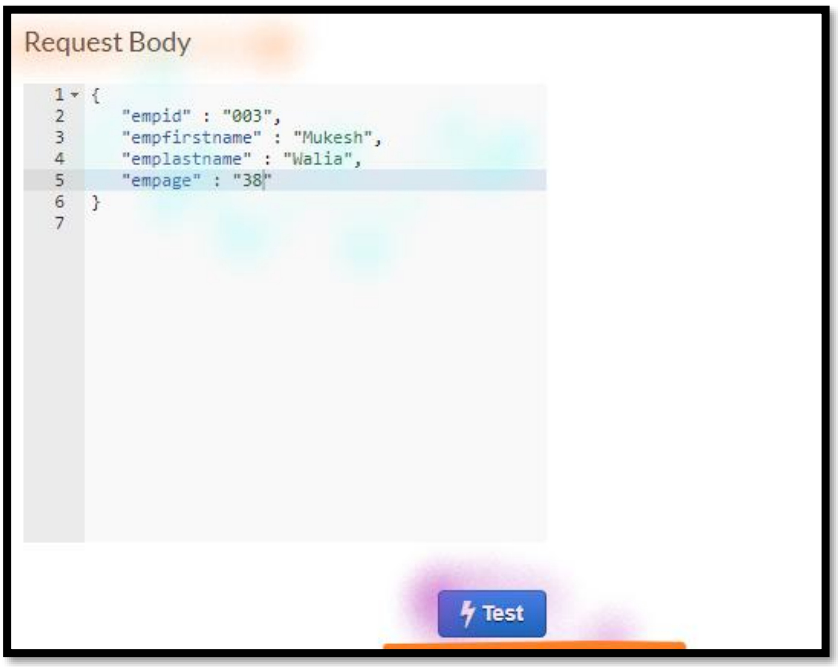
a. Click the **Test**.

b. In the **Request Body**, **Copy** the below event:

Note: You can now add the new items via API Gateway.

```
{
   "empid": "003",
   "empfirstname": "Mukesh",
   "emplastname": "Walia",
   "empage" : "38"
}
```

c. Click the **Test**.



```
{
    "empid" : "003",
    "empfirstname" : "Mukesh",
    "emplastname" : "Walia",
    "empage" : "38"
}
```

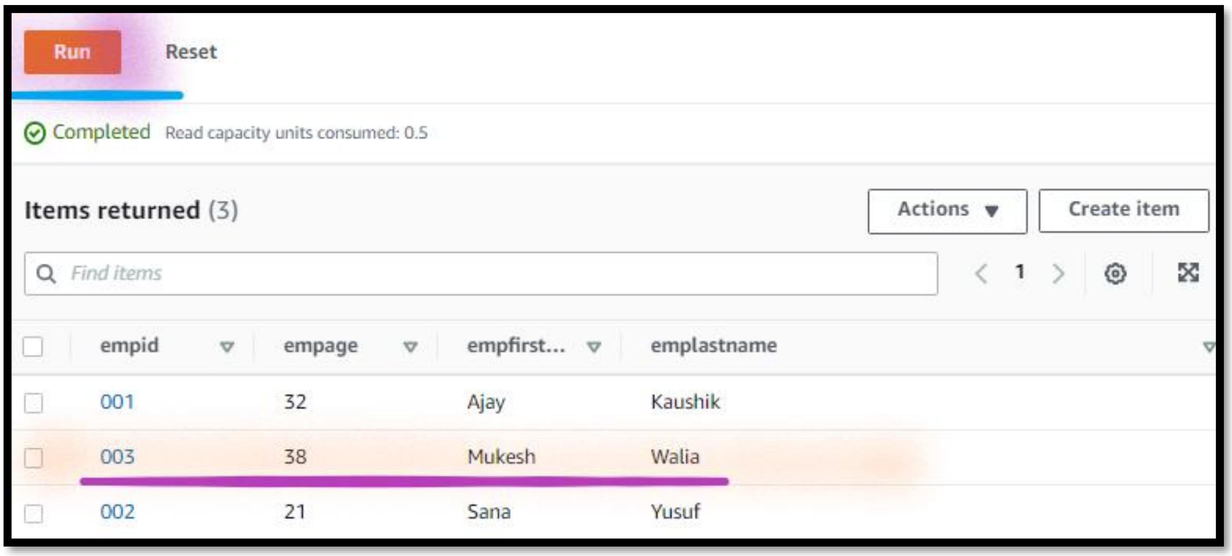
Note: If request executed **succesfully**, you can see the Request status as **200**.

Note: You can view the **Items** in DynamoDB, which you have added in the DynamoDB via the **API gateway**.

Step 3: View the DynamoDB Data

37. In the **AWS Management Console**, on the **Services** menu, click **DynamoDB**.
38. Select **Items**.
    a. Select **empdata** DynamoDB table.
    b. Select **Run**.

Note: You can view the **Added Items**, which you have added in the DynamoDB via the **API gateway**.



Completed Read capacity units consumed: 0.5

empid	empage	empfirst...	emplastname
001	32	Ajay	Kaushik
003	38	Mukesh	Walia
002	21	Sana	Yusuf

Task 4: Deploy API

Step 1: Deploy API

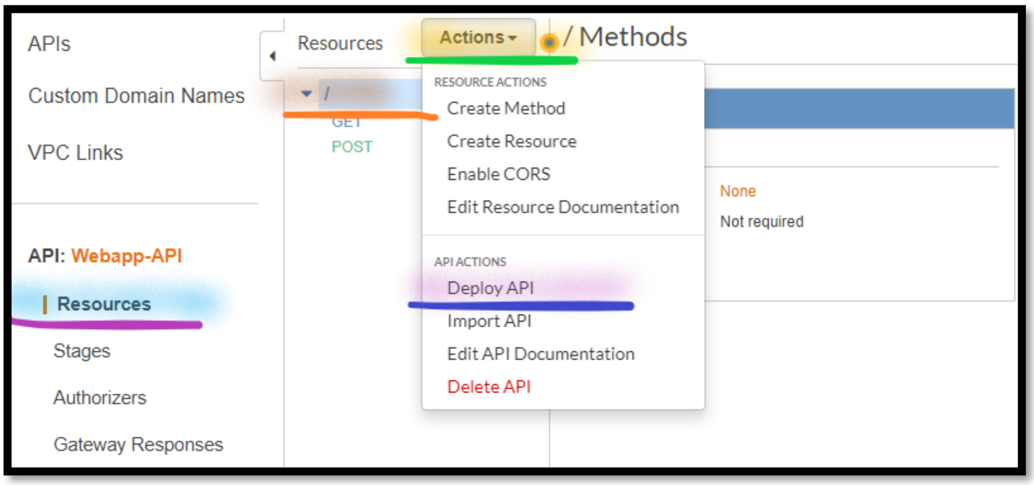
39. In the **AWS Management Console**, on the **Services** menu, click **API Gateway**.

40. Open **empdata-API** API.

41. **Go to left**, choose **Resources**.

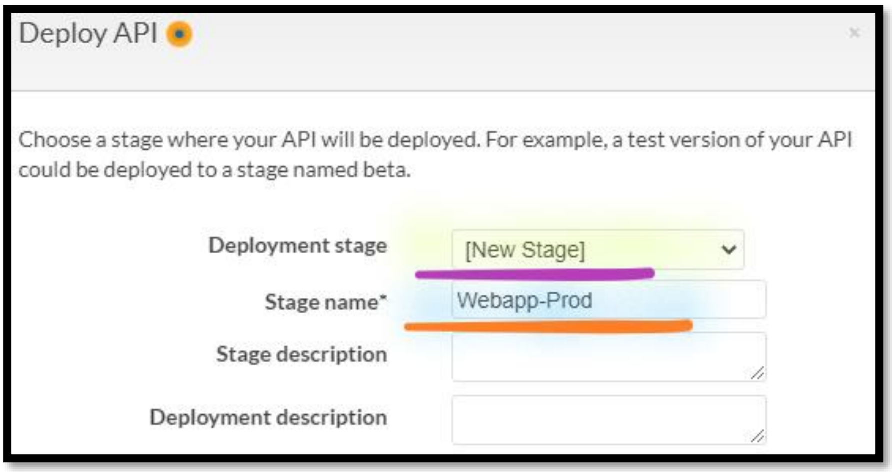
    a. Select **/** **resource**.

    b. From the **Actions** dropdown select **Deploy API**.



i. **Deployment stage**: Dropdown and Select **[New Stage]**.

ii. **Stage name**: Write **ReadWrite-API**.



Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage: [New Stage]

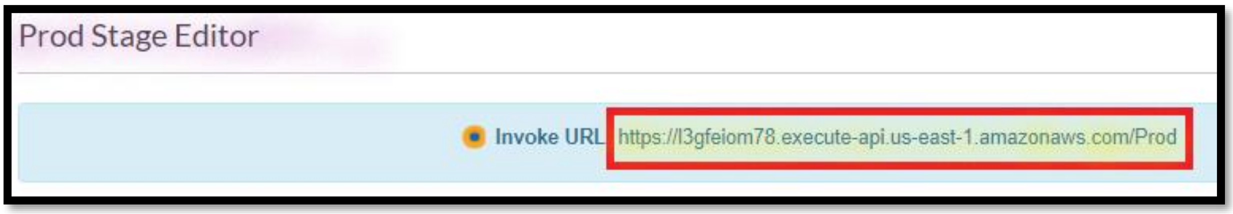
Stage name*: Webapp-Prod

Stage description:

Deployment description:

c. Select **Deploy**.

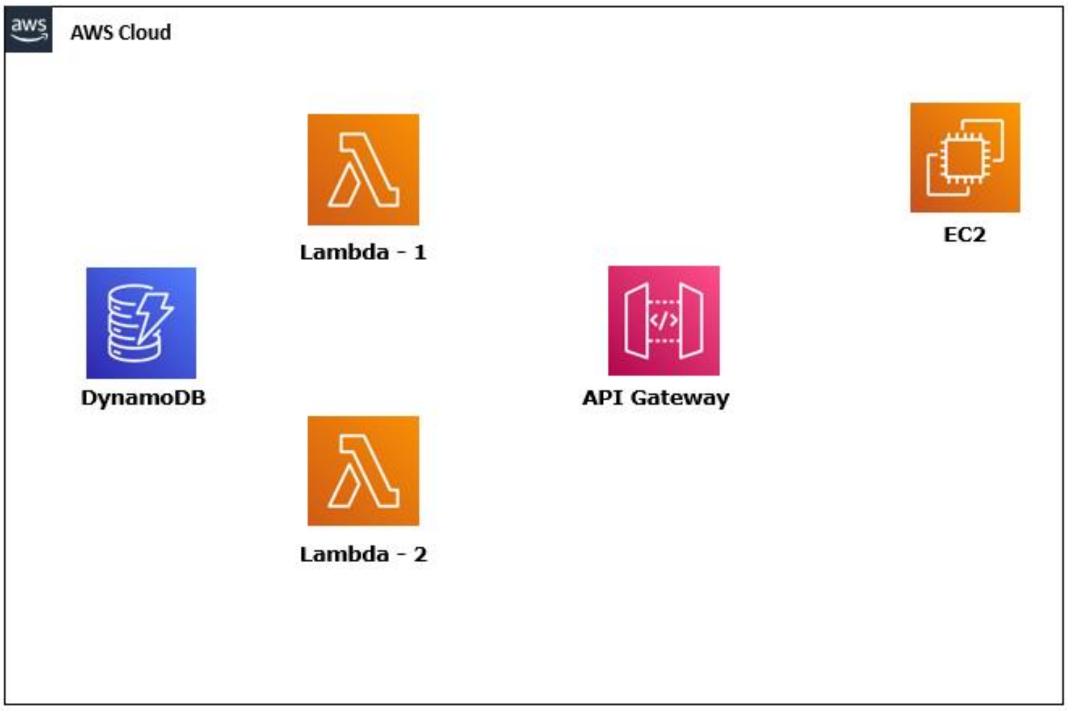
42. **Copy** the **Invoke URL** in the **Notepad**.



Prod Stage Editor

Invoke URL https://l3gfeiom78.execute-api.us-east-1.amazonaws.com/Prod

Task 5: Validate the Solution using Httprepl



Step 1: Create EC2 Instance

43. In the **AWS Management Console**, on the **Services** menu, search and select **EC2**.

44. Choose the **US East (N. Virginia)** region list to the right of your account information on the navigation bar.

45. Select **Instances**.

46. Select **Launch Instances**.
    a. In the **Name and tags** section:
        i. **Name**: Write **Dev-API-Instance**.
    b. In the In the **Application and OS Images** section:
        i. In the **Search box**:
            a) Type **Microsoft Windows Server 2019 Base**.
            b) Press **Enter** ***key***.

Note: You can see the **Choose an Amazon Machine Image** page.

            c) **From** the **Choose an Amazon Machine Image** page:
                1) Select **Microsoft Windows Server 2019 Base**.

Note: You can see the **Launch an Instance** page.

    c. In the **Instance Type** section:
        i. **Instance type**: Dropdown and in the **Search box**:
            a) Type **t2.micro**.
            b) Select **t2.micro**.
    d. In the **Key pair (login)** section:
        i. **Key pair name**: Dropdown and select **My-Dev-LAB-KP**.

e. In the **Network settings** section:

i. Click on **Select Create security group**.

a) Click on **Select Allow RDP traffic from**.

1) Dropdown and select **Anywhere**.

Note: Leave the other details as default.

f. In the **Summary** section:

i. Select **Launch Instances**.

Note: **Wait**, till you can see the **message** "**Successfully initiated launch of instance**".

g. Select **View all instances**

Note: **Wait**, till you can see the **Dev-API-Instance** Instance **State** is **Running**.

Note: **Wait**, till you can see the **Dev-API-Instance**Instance **Status check** is **2/2 check passed**.

Step 2: Copy the IP Address of Instance

47. **From** the **EC2** console.

48. Select the **Dev-API-Instance**.

a. Select the **Details**.

Note: **Copy** the **Public IP address** of **Dev-API-Instance** in the **Notepad**.

Step 3: Generate the Password of Instance

49. **From** the **Dev-API-Instance** console.

a. Select **Actions**.

i. Select **Security**.

ii. Select **Get Windows Password**.

a) From the **Get Windows Password** console:

1) **Browse**: **Click**, **Navigate** and **select** the **My-Dev-LAB-KP**.pem key pair (which you have downloaded in the previous step).

2) Click on **Decrypt Password**.

Note: **Copy** the **Dev-API-Instance Password** in the **Notepad**.

3) Select **Ok**.

Step 4: Connect to Instance

50. From the **Local Desktop/ Laptop** (Windows server 2019), right click on **Start** & **Run**.

a. In the **Open**, write **mstsc**.

b. Select **Ok**.

i. **From** the **Remote Desktop Connection**:

1. **Computer**: Write the **Public IP Address** of the **Dev-API-Instance**.

2. Select **Connect**.

Note: You can **get the prompt** to enter the **Username** and **Password**.

1) **Username:** Write **Administrator**.

2) **Password**: Write the **Password** (*which you have copied in the previous step*).

3) Select **Ok**.

Step 5: Update the Security Settings

51. From the Dev-API-Instance (Windows server 2019), right click on Start & Run.

    a. In the Open, write servermanager.

    b. Select Ok.

       i. **From the Server Manager:**

          a) **Select the Local Server.**

             1) **IE Enhanced Security Configuration: Select On.**

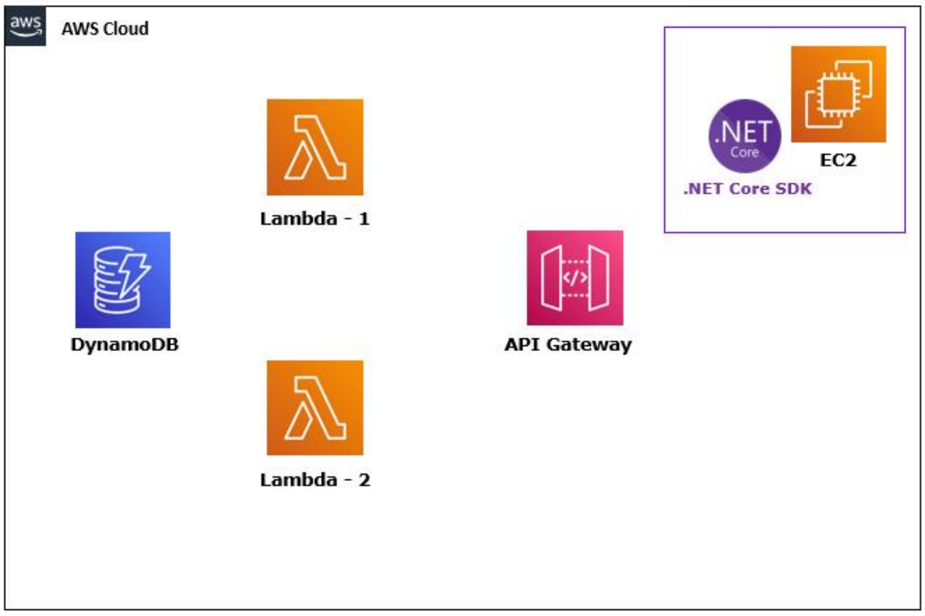
Note: You can see the **Internet Explorer Enhanced Security Configuration** page.

I. **Administrators: Select Off.**

II. **Select Ok.**

ii. **Select Cross to close the Server manager.**

Step 4: Install the Dot Net Core SDK



52. From the Dev-API-Instance (Windows server 2019).

    a. Download and Install the *.Net Core SDK* for Windows x64.

Note: Use the below URL to download the **.Net Core SDK 3.1**.

https://download.visualstudio.microsoft.com/download/pr/4e88f517-196e-4b17-a40c-2692c689661d/eed3f5fca28262f764d8b650585a7278/dotnet-sdk-3.1.301-win-x64.exe

Note: **Wait**, till **.NET Core SDK 3.1** install **succesfully**.

Step 5: Check the .NET Core SDK version

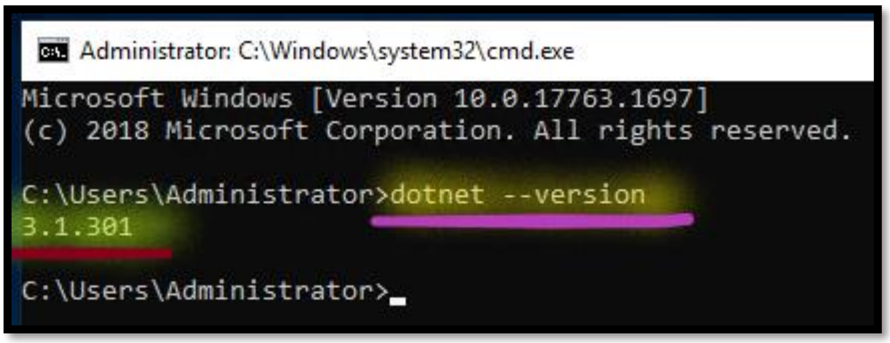
53. From the **Dev-API-Instance** (Windows server 2019), right click on **Start** & **Run**.

    a. In the **Open**, write **cmd**.

    b. Select **Ok**.

       i. From the **command line interpreter**, write **dotnet --version**, press **Enter** key.

Note: You can see the **Dotnet** installed **version**.



```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.1697]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>dotnet --version
3.1.301

C:\Users\Administrator>_
```

Step 6: Install the HTTP REPL

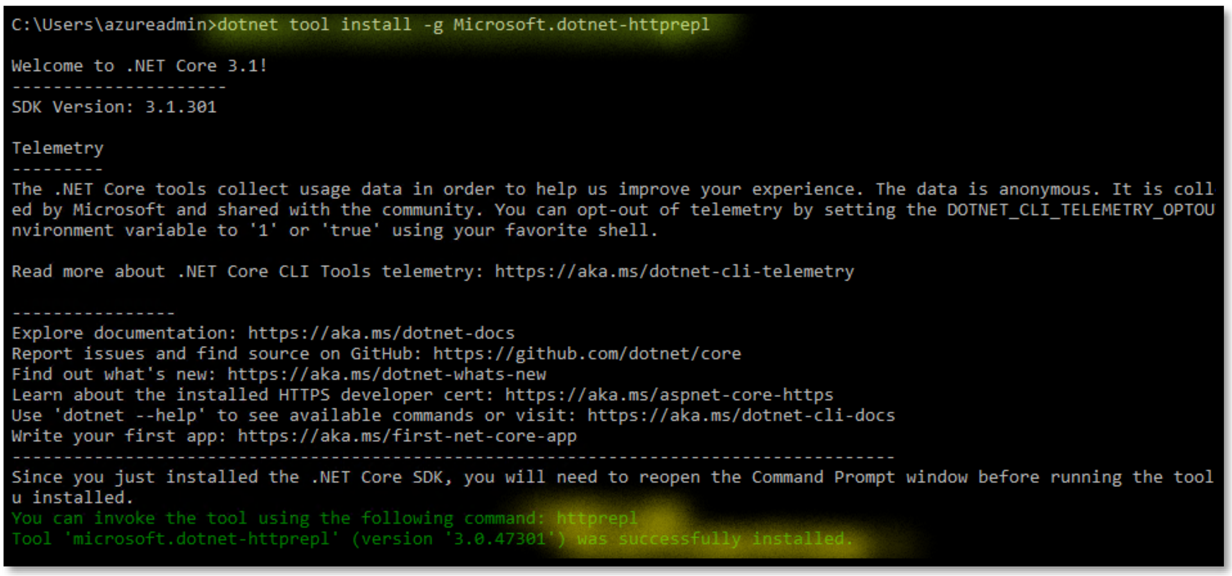
54. From the **Dev-API-Instance** (Windows 2019), right click on **Start** & **Run**.

    a. In **Open**, write **cmd** run the following command:

    b. From **CLI** **Install** the **HTTP REPL**.

    dotnet tool install -g Microsoft.dotnet-httprepl

Note: You can see the output, httprepl installed succesfully.



```
C:\Users\azureadmin>dotnet tool install -g Microsoft.dotnet-httprepl

Welcome to .NET Core 3.1!
---------------------
SDK Version: 3.1.301

Telemetry
---------
The .NET Core tools collect usage data in order to help us improve your experience. The data is anonymous. It is coll
ed by Microsoft and shared with the community. You can opt-out of telemetry by setting the DOTNET_CLI_TELEMETRY_OPTOU
nvironment variable to '1' or 'true' using your favorite shell.

Read more about .NET Core CLI Tools telemetry: https://aka.ms/dotnet-cli-telemetry

----------------
Explore documentation: https://aka.ms/dotnet-docs
Report issues and find source on GitHub: https://github.com/dotnet/core
Find out what's new: https://aka.ms/dotnet-whats-new
Learn about the installed HTTPS developer cert: https://aka.ms/aspnet-core-https
Use 'dotnet --help' to see available commands or visit: https://aka.ms/dotnet-cli-docs
Write your first app: https://aka.ms/first-net-core-app
--------------------------------------------------------------------------------------
Since you just installed the .NET Core SDK, you will need to reopen the Command Prompt window before running the tool
u installed.
You can invoke the tool using the following command: httprepl
Tool 'microsoft.dotnet-httprepl' (version '3.0.47301') was successfully installed.
```

Info: The HTTP Read-Eval-Print Loop (REPL) is A lightweight, cross-platform command-line tool, used for making HTTP requests to test web APIs and view their results.

c. Close the **cmd**.

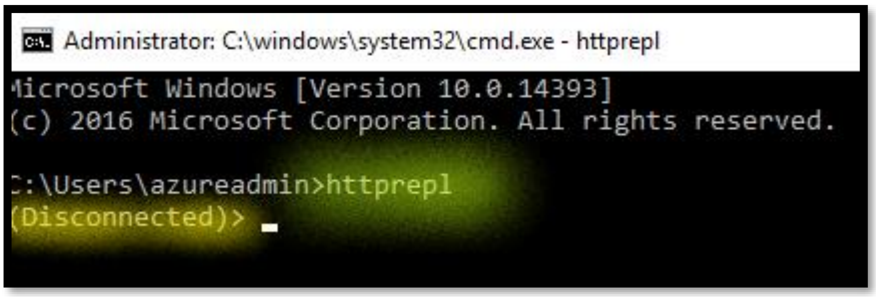
55. From the **Dev-API-Instance** (Windows 2019), right click on **Start** & **Run**.

a. In **Open**, write **cmd**.

b. **Test** the **HTTPREPL** from **CLI**.

httprepl

Note: You can see the output, shown as **disconnected**.



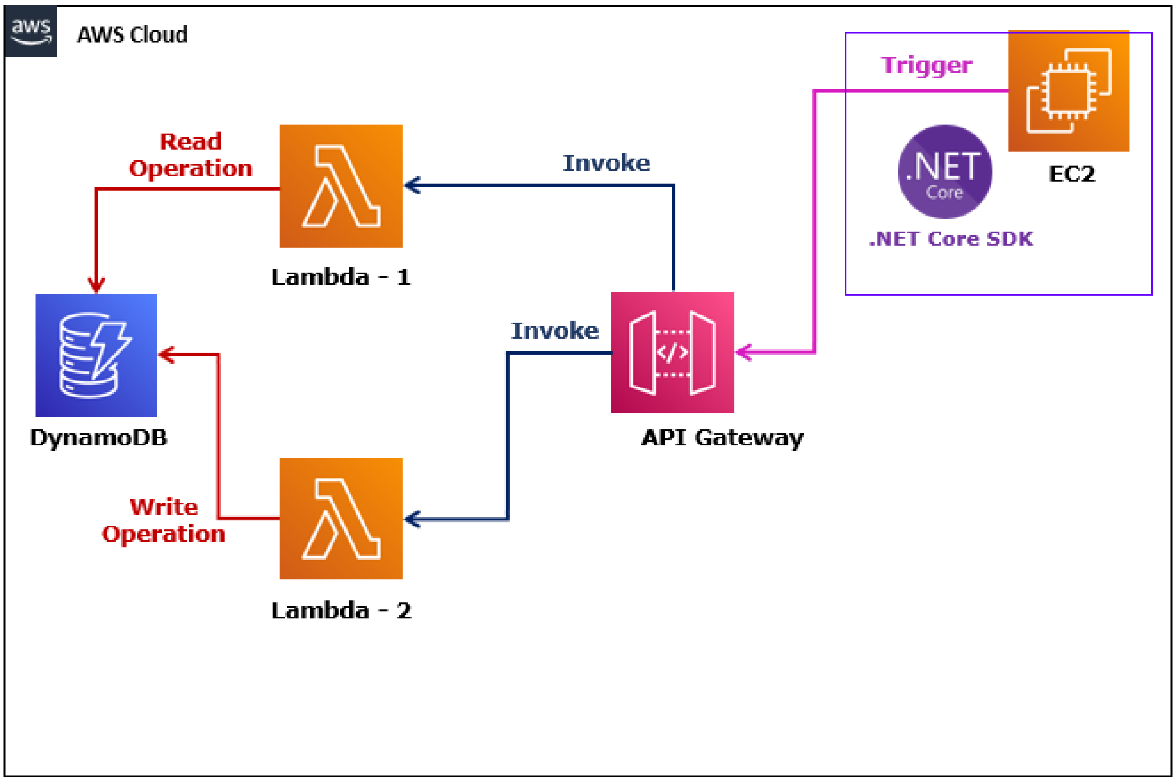
Administrator: C:\windows\system32\cmd.exe - httprepl

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\azureadmin>httprepl
(Disconnected)> _
```

c. **Exit** the **HTTPREPL** from **CLI**.

exit

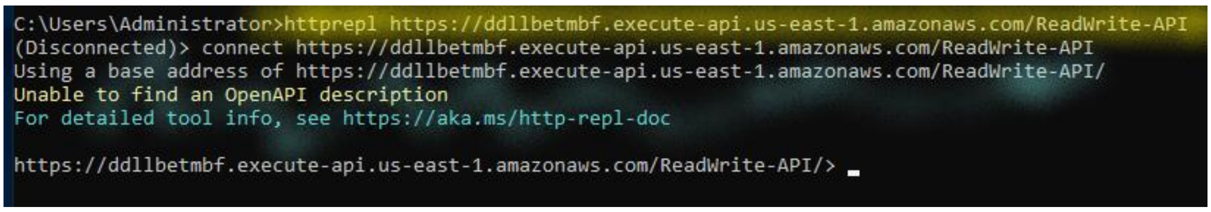
Step 6: Test API by using HTTPREPL



56. To **Start** the **HTTPREPL** tool and set the base Uniform Resource Identifier (URI) to the value of the Request URL for the API operation run the following command from **CLI**:

httprepl API-Invoke-URL

Note: **Replace** the **API-Invoke-URL**, with the API Invoke URL which you have copied in the previous step.



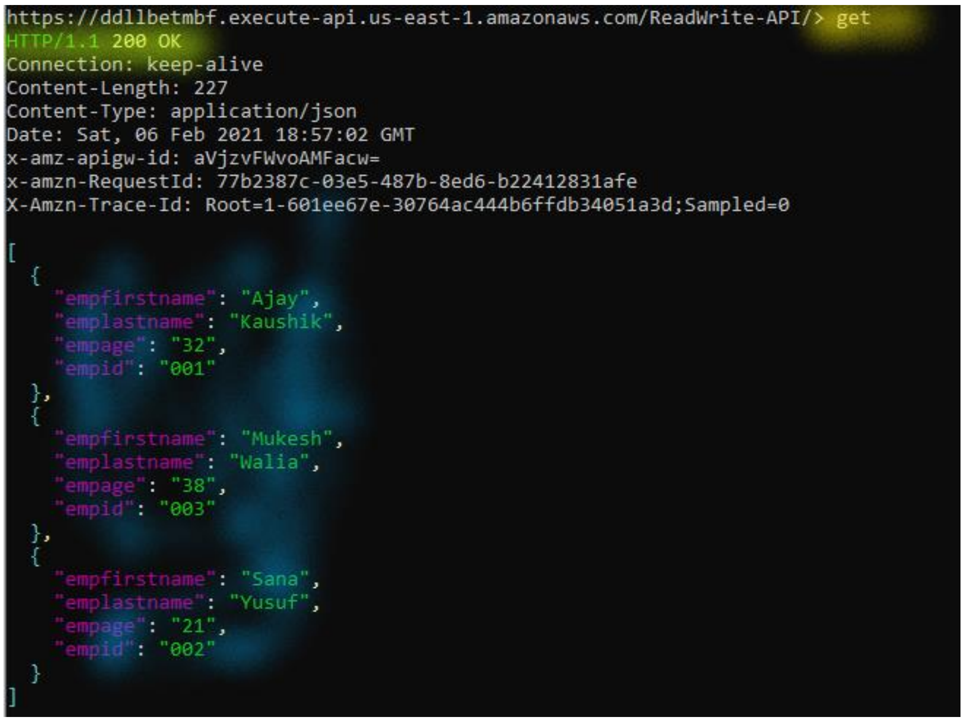
```
C:\Users\Administrator>httprepl https://ddllbetmbf.execute-api.us-east-1.amazonaws.com/ReadWrite-API
(Disconnected)> connect https://ddllbetmbf.execute-api.us-east-1.amazonaws.com/ReadWrite-API
Using a base address of https://ddllbetmbf.execute-api.us-east-1.amazonaws.com/ReadWrite-API/
Unable to find an OpenAPI description
For detailed tool info, see https://aka.ms/http-repl-doc

https://ddllbetmbf.execute-api.us-east-1.amazonaws.com/ReadWrite-API/> _
```

a. Within the **tool prompt**, **run** the get **command** against the API endpoint from **CLI**:

get

Note: Observe the JSON response content.

Note: In the Response body, you can see the **Items**, which you have added in the DynamoDB table.

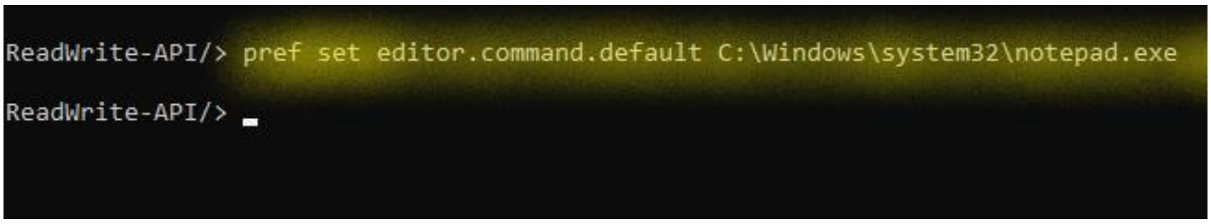


```
https://ddllbetmbf.execute-api.us-east-1.amazonaws.com/ReadWrite-API/> get
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 227
Content-Type: application/json
Date: Sat, 06 Feb 2021 18:57:02 GMT
x-amz-apigw-id: aVjzvFWvoAMFacw=
x-amzn-RequestId: 77b2387c-03e5-487b-8ed6-b22412831afe
X-Amzn-Trace-Id: Root=1-601ee67e-30764ac444b6ffdb34051a3d;Sampled=0

[
  {
    "empfirstname": "Ajay",
    "emplastname": "Kaushik",
    "empage": "32",
    "empid": "001"
  },
  {
    "empfirstname": "Mukesh",
    "emplastname": "Walia",
    "empage": "38",
    "empid": "003"
  },
  {
    "empfirstname": "Sana",
    "emplastname": "Yusuf",
    "empage": "21",
    "empid": "002"
  }
]
```

b. To **Set** the **default text editor**, **run** the following **command** from **CLI**:

pref set editor.command.default C:\Windows\system32\notepad.exe

Note: By default, the HttpRepl has no text editor configured for use. To test web API methods requiring an HTTP request body, a default text editor must be set. The HttpRepl tool launches the configured text editor for the sole purpose of composing the request body.



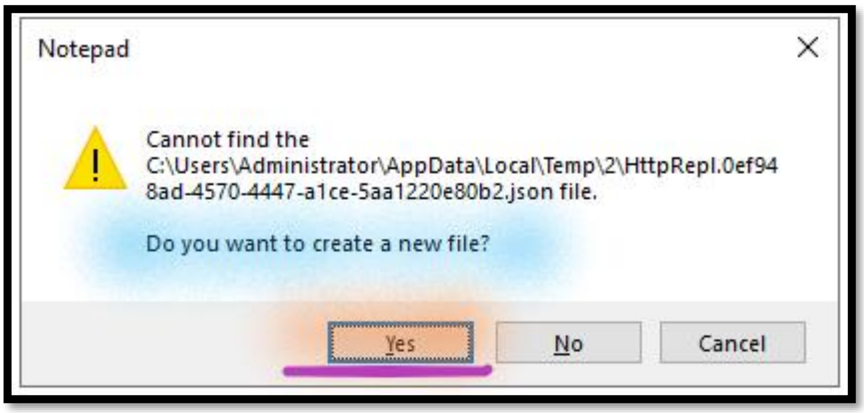
```
ReadWrite-API/> pref set editor.command.default C:\Windows\system32\notepad.exe

ReadWrite-API/> _
```

c. Within the **tool prompt**, **run** the get **command** against the API endpoint from **CLI**:

post -h Content-Type=application/json

Note: In the preceding command, the HTTP request header is set to indicate a request body media type of JSON. The default text editor opens a *.tmp* file.

d. Select **Yes**, when you **prompt** to ***create new file***.

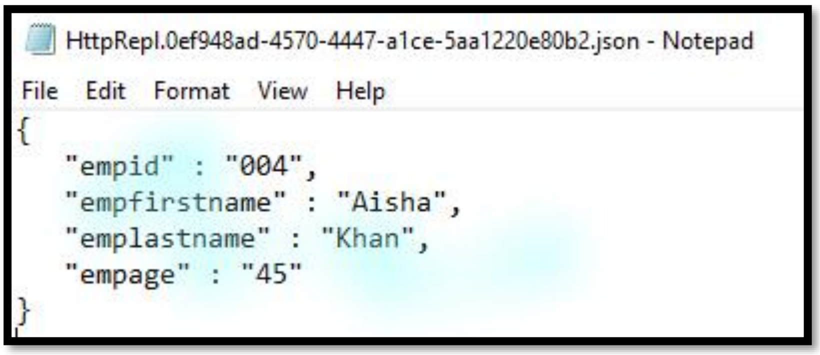

Notepad

Cannot find the C:\Users\Administrator\AppData\Local\Temp\2\HttpRepl.0ef948ad-4570-4447-a1ce-5aa1220e80b2.json file.

Do you want to create a new file?

i. In the **Body**, **Copy** the below details:

```
{
  "empid": "004",
  "empfirstname": "Aisha",
  "emplastname": "Khan",
  "empage" : "45"
}
```

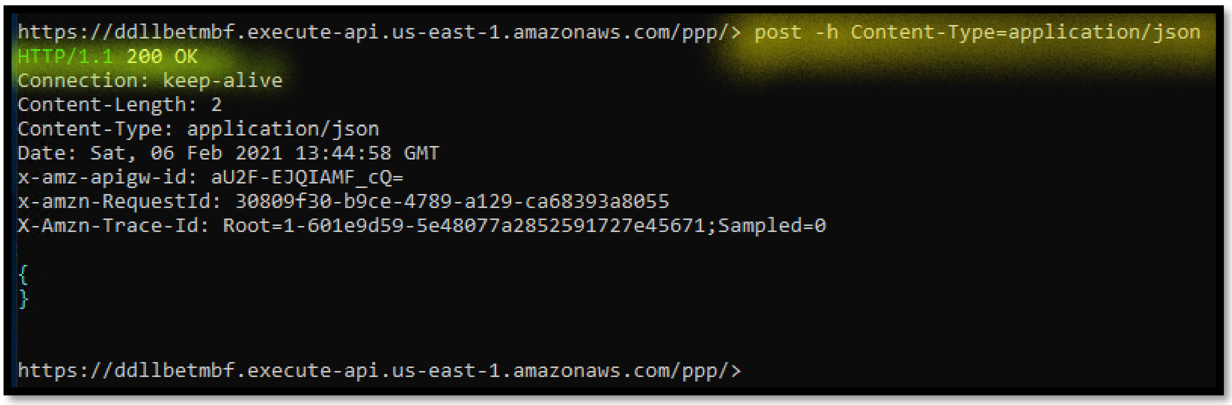

HttpRepl.0ef948ad-4570-4447-a1ce-5aa1220e80b2.json - Notepad

```
{
   "empid" : "004",
   "empfirstname" : "Aisha",
   "emplastname" : "Khan",
   "empage" : "45"
}
```

ii. From the **Notepad**, Select **File** and Select **Save**.

iii. From the **Notepad**, Select **File** and Select **Exit**.

e. Once you **exit** the notepad. The **following output** appears in the **command shell**:



```
https://ddllbetmbf.execute-api.us-east-1.amazonaws.com/ppp/> post -h Content-Type=application/json
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 2
Content-Type: application/json
Date: Sat, 06 Feb 2021 13:44:58 GMT
x-amz-apigw-id: aU2F-EJQIAMF_cQ=
x-amzn-RequestId: 30809f30-b9ce-4789-a129-ca68393a8055
X-Amzn-Trace-Id: Root=1-601e9d59-5e48077a2852591727e45671;Sampled=0

{
}


https://ddllbetmbf.execute-api.us-east-1.amazonaws.com/ppp/>
```

f. Within the **tool prompt**, run the get command against the API endpoint:

get

Note: In the Response body, you can see the **newly added items**, which you have added in the previous step.

g. Write the following command to **exit**:

exit

Note: You can view the **Items** in DynamoDB, which you have added in the DynamoDB via **Invoke URL**.

Step 7: View the DynamoDB Data

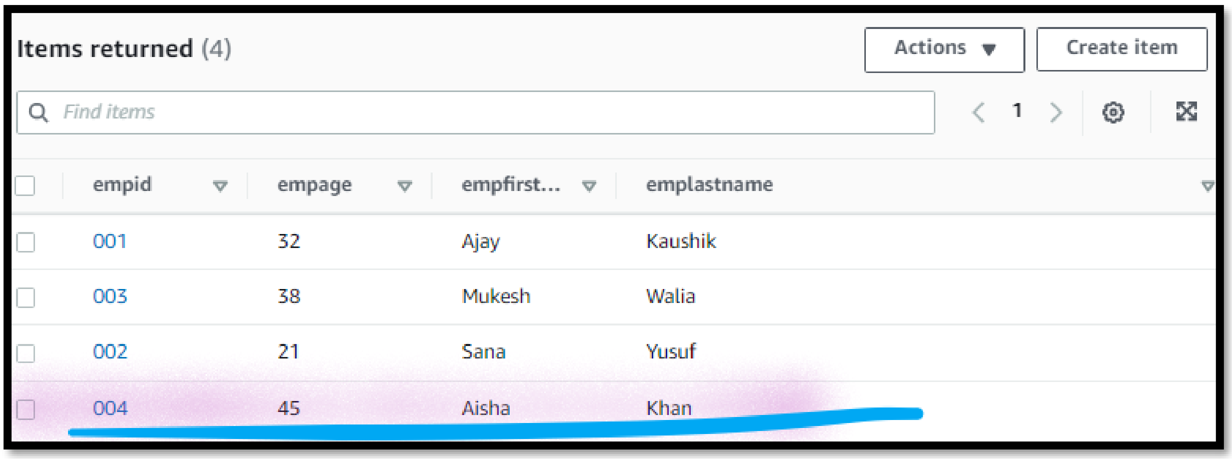
57.In the **AWS Management Console**, on the **Services** menu, click **DynamoDB**.

58.Select **Items**.

a. Select **empdata** DynamoDB table.

b. Select **Run**.

Note: You can view the **Added Items**, which you have added in the DynamoDB via the **Invoke URL**.



empid	empage	empfirst...	emplastname
001	32	Ajay	Kaushik
003	38	Mukesh	Walia
002	21	Sana	Yusuf
004	45	Aisha	Khan

Task 4: Delete the Environment

Step 1: Delete the DynamoDB Table

59. In the **AWS Management Console**, on the **Services** menu, click **DynamoDB**.

60. Click the **Tables**.

    a. Select the **emdpdata**.

    b. Select **Delete table**.

Step 2: Delete Lambda Function

61. In the **AWS Management Console**, on the **Services** menu, click **Lambda**.

62. Click the **Functions**.

    a. Select the **ReadItems**.

    b. Select **Actions**.

    c. Select **Delete**.

63. Click the **Functions**.

    a. Select the **WriteItems**.

    b. Select **Actions**.

    c. Select **Delete**.

Step 3: Delete the API Gateway

64. In the **AWS Management Console**, on the **Services** menu, click **API Gateway**.

65. Select the **empdata-API**.

    a. Click on the **Actions**.

    b. Select the **Delete**.

    c. When ***prompted to delete***, Select the **Delete**.

Step 4: Terminate EC2 Instances

66. In the **AWS Management Console**, on the **Services** menu, click **EC2**.

67. Click **Instances**.

68. Select **HTTPREPL Server**.

    i. Click on **Instance state**.

    ii. Select **Terminate instance**.

    iii. Select **Terminate**.