

Contents

| | |
|---|----|
| Installation Instructions | 2 |
| Pre-requisites | 2 |
| Steps to import TelecomDemo Amazon Lex Bot | 3 |
| Step to add Lex bot to Amazon Connect Instance..... | 7 |
| Steps to create DynamoDB table:..... | 8 |
| Steps to create or upload lambda. (If Visual Studio is available) | 9 |
| Upload “AddCustomerContactFlowJourneyData“ lambda | 9 |
| Steps to create or upload lambda. (If Visual Studio is not available) | 13 |
| Upload “AddCustomerContactFlowJourneyData“ lambda : | 13 |
| Configuration section:..... | 16 |
| Change the function handler | 18 |
| Test the lambda function section:..... | 19 |
| Upload “SendSMSFromContactFlowToCaller“ lambda : | 20 |
| Create SMS Project in Pinpoint Service:..... | 23 |
| Configuration section:..... | 25 |
| Change the function handler | 27 |
| Test the lambda function section: | 29 |
| Configure lambda to amazon connect instance..... | 30 |
| Upload “M1CallStatisticsDashboard“ Lambda : <i>This lambda will be used to fetch the data from DynamoDB for displaying in the UI dashboard.</i> | 31 |
| Configuration section:..... | 34 |
| Change Function handler section | 36 |
| Test the lambda function section:..... | 38 |
| Steps to import contact flow to Amazon Connect instance..... | 39 |
| Creating REST API in APIGateway to access lambda from public | 41 |
| Deploy dashboard code in S3 and make available to access from public | 48 |
| Creating CloudFront to access the S3 objects from public | 58 |
| Launch dashboard..... | 59 |
| Testing by associating call flow to the number and make test calls | 60 |
| SMS Deflect flow: | 60 |
| Agent Queue Flow: | 60 |
| DTMF Flow: | 60 |

Installation Instructions

Pre-requisites

- ✓ Download the statistics dashboard source code & other references from the below Github link.

<https://github.com/ajvvenkatesh/ConnectCallStatisticsDashboard.git>

Download and Unzip the zip file to your local folder.

- ✓ An AWS Account.
- ✓ An IAM user with the ability to create a server-less SAM application, create an IAM role, edit an Amazon Connect instance, and modify an S3 bucket's configuration.
- ✓ An Amazon Connect instance
- ✓ Basic knowledge on how to create dynamo DB table. Name it as "[CustomerContactFlowJourney](#)". While creating choose the partition key as "ID" and select data type as "Number" like mentioned below.

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.

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.

| | |
|---------------------------------|---|
| <input type="text" value="ID"/> | <input type="button" value="Number"/> ▼ |
|---------------------------------|---|

1 to 255 characters and case sensitive.

- ✓ Basic knowledge on C# coding to create server-less functions as lambdas. There are two lambda functions which we are using in this example

Steps to import TelecomDemo Amazon Lex Bot

This bot is needed for the contact flow (M1LexDashboard.json). This bot helps in branching the Amazon Connect contact flow decision making when the customer calling.

1. Goto [Amazon Lex](#)
2. If you have never created any Amazon Lex bot, then click on Get Started → Click Cancel (This is needed to go to home page)
3. You should see the Amazon Lex home page with Create and Actions buttons and all the created Amazon Lex bots

The screenshot shows the Amazon Lex home page. On the left, there's a sidebar with 'Amazon Lex' at the top, followed by 'Bots' (which is highlighted in orange), 'Intents', and 'Slot types'. Below that are 'Migration tool' and 'Switch to the new Lex V2 console'. The main area has a heading 'Try the new Lex V2 Console' with a sub-note about productivity tools. Below this is a table titled 'Bots' with columns for 'Name', 'Status', 'Locale', 'Last updated', and 'Date Created'. At the top of the table, there are buttons for 'Create' and 'Actions'. A dropdown menu for 'Actions' is open, showing options: 'Import' (which is highlighted in black), 'Export', and 'Delete'. There are also icons for refresh, settings, and help.

4. Click Actions → Import bot → Upload the zip file from Unzipped Lex Folder

| Name | Date modified | Type | Size |
|--|------------------|----------------------|------|
| 📁 TelecomDemo_5_1b2be31b-ea96-4967-... | 9/1/2021 7:23 PM | Compressed (zipp...) | 1 KB |

Import bot

⚠ All built-in resources (intents and slot types) must match bot locale

If your bot references built-in intents or built-in slot types of a locale different than locale of the bot, it will fail on the build even if the import and overwriting of resources is successful.

Upload file **Browse**

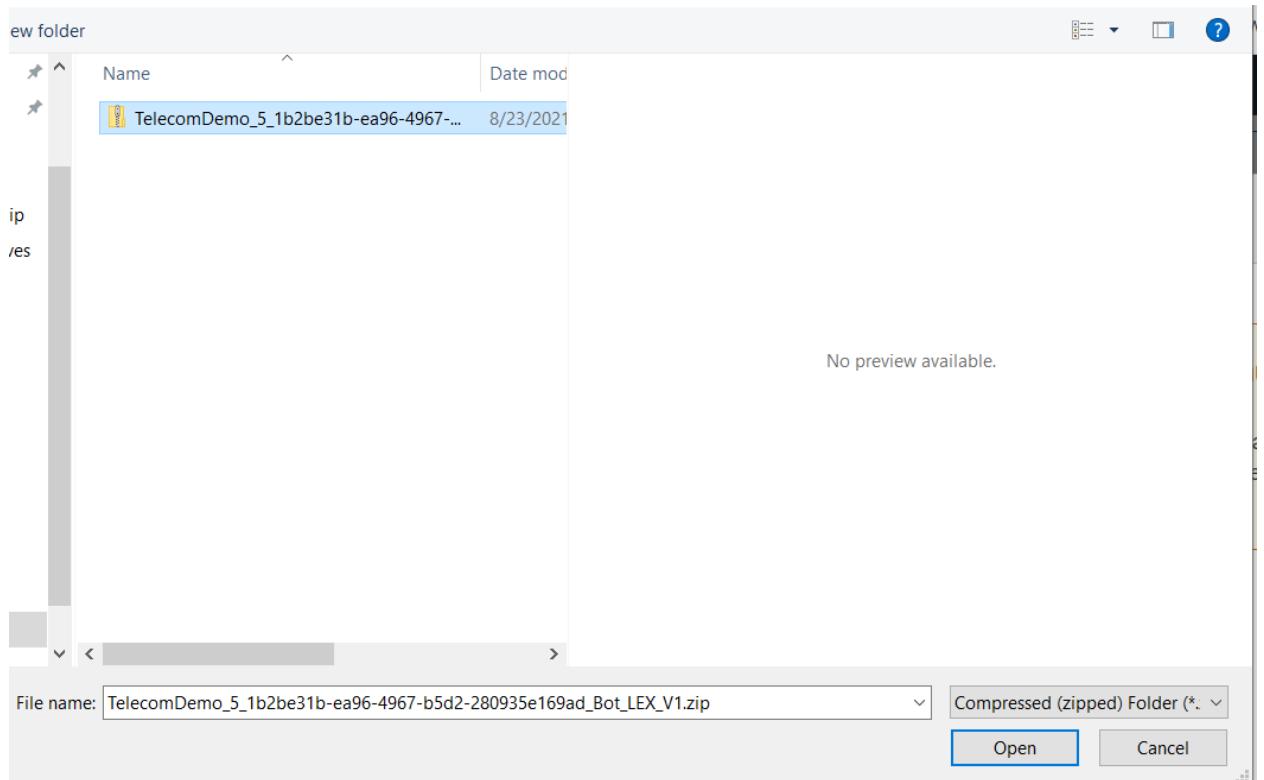
Upload a .zip file

▶ Tags

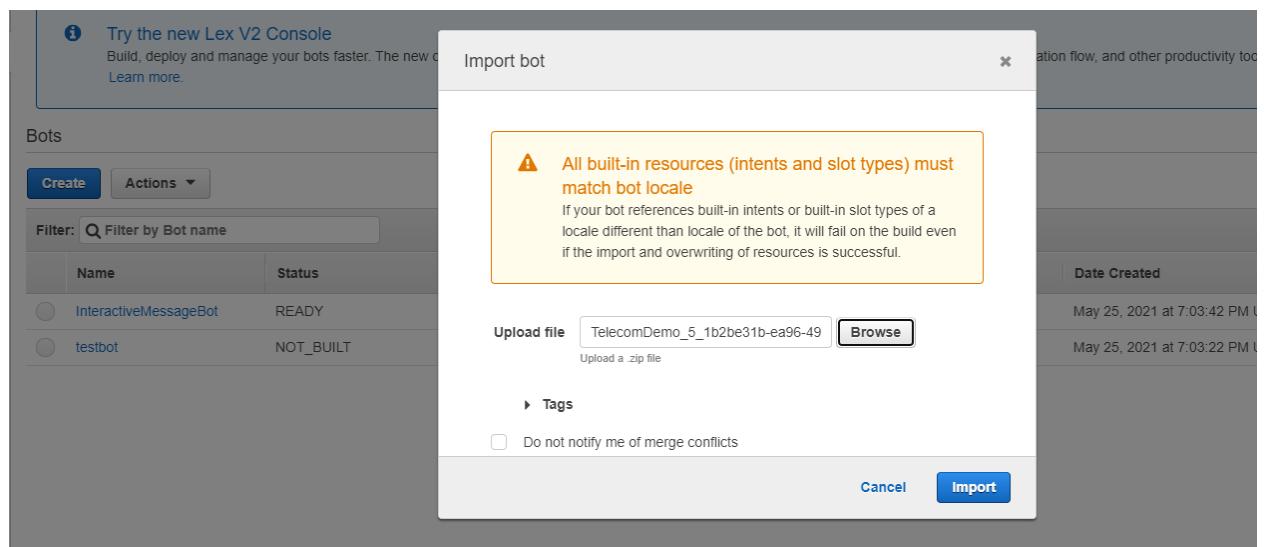
Do not notify me of merge conflicts

Cancel **Import**

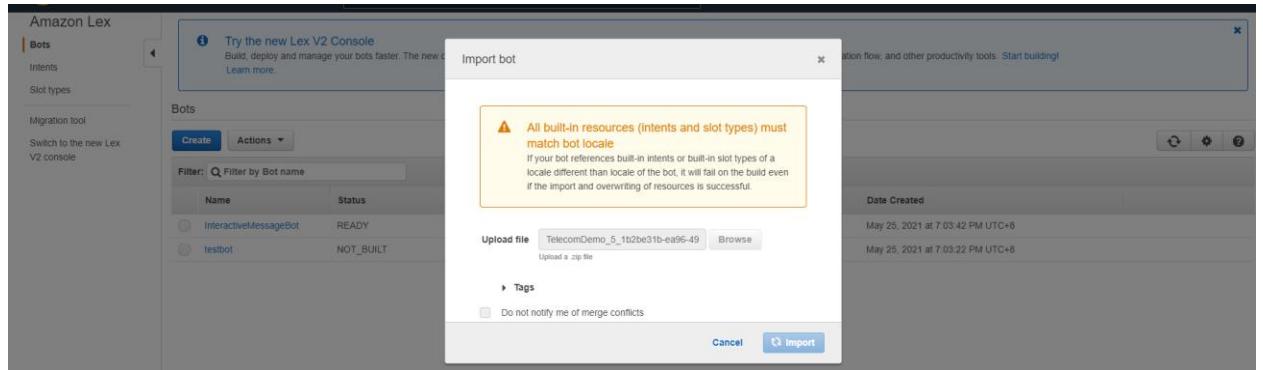
Click on browse to upload the Bot zip file.



Click on Open.



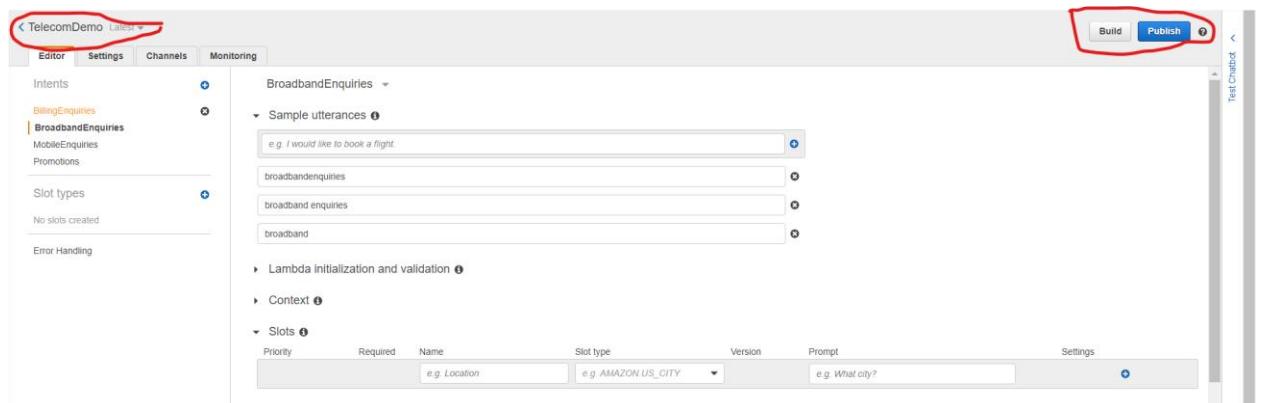
Click on Import button to load the Bot data to the console.



If it asks for overwriting intents click overwrite and the TelecomDemo Amazon Lex bot gets imported in NOT_BUILT status.



- Click on TelecomDemo Amazon Lex bot → Click Build and then Publish (In the top right corner of your page)



- If it asks you for alias put TelecomDemo as alias and click Publish.
- After successfully published, you can see the bot status as READY like mentioned below:

| | | | | |
|-------------|-------|--------------|------------------------------------|---------------------------------|
| TelecomDemo | READY | English (US) | August 2, 2021 at 4:14:52 PM UTC+8 | July 29, 2021 at 9:05:34 PM UTC |
|-------------|-------|--------------|------------------------------------|---------------------------------|

Step to add Lex bot to Amazon Connect Instance.

1. Navigate to Amazon connect console and choose the instance whichever you want bot to be assigned.
2. Navigate to Contact Flows like mentioned below:

The screenshot shows the Amazon Connect interface for a specific instance named 'venkydemo'. The left sidebar has a red circle around the 'Contact flows' link. The main content area displays two contact flow cards: 'The n' (with a note 'We've r') and 'We ar' (with a note 'To mail'). Below these is a section titled 'Contact flow' with a note: 'Amazon Connect c within your contact order to use this fe up to two signing k'. At the bottom are 'Add key' and 'Edit' buttons.

3. Go to Amazon Lex section from the right panel. Choose the region as "Asia Pacific:Singapore" or the region where connect instance is created. Select the drop down of Bot and then choose "Telecom Demo" bot which we have created in previous steps.

The screenshot shows the 'Amazon Lex' integration page. It includes a note about integrating Lex bots into contact flows. A 'Region' dropdown is set to 'Asia Pacific: Singapore'. A 'Bot' dropdown is open, showing a list of available bots: 'InteractiveMessage...', 'RichCardBot (Classic)', 'salesforce_case (Cl...)', 'salesforce_combine...', 'StartVideoCall (Cla...', and 'TelecomDemo (Cla...)' (which is circled in red). A note at the bottom explains how AWS Lambda functions can be used in contact flows.

Once bot selected then click on “**+Add Lex Bot**” button to add the bot to the connect instance.

Bot will be assigned like mentioned below:

Amazon Lex

Integrate Amazon Lex bots into your contact flows to take advantage of the same speech recognition and natural language understanding technology that powers Alexa.

Note: By adding Lex bots, you are granting Amazon Connect permission to interact with them [Create a new Lex bot](#)

Region

Bot

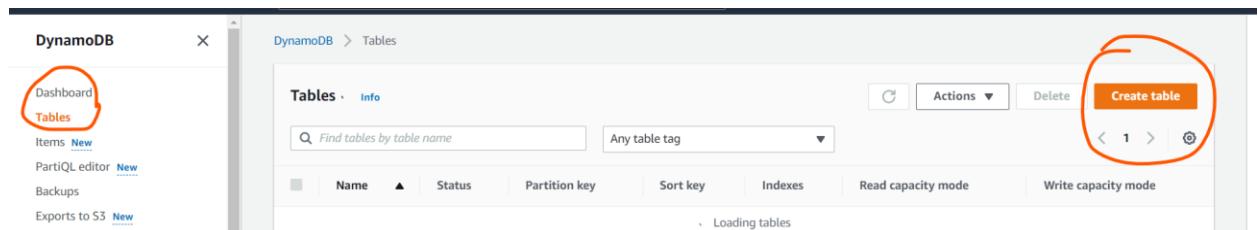
[+ Add Lex Bot](#)

Lex bots

TelecomDemo (Classic) [Remove](#)

Steps to create DynamoDB table:

1. Go to [DynamoDB](#)
2. Navigate to Tables from the left menu and click on Create table button which appears on the top right corner like mentioned below:



3. Add table name as “**CustomerContactFlowJourney**” & add primary key as “**ID**” with data type as **Number**. Then click on “Create” button which is appears on the right bottom corner.

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.

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.

 ▾

1 to 255 characters and case sensitive.

Leave other details default and proceed with creating table by clicking on "Create table" button.

Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

You can add 50 more tags.

Cancel

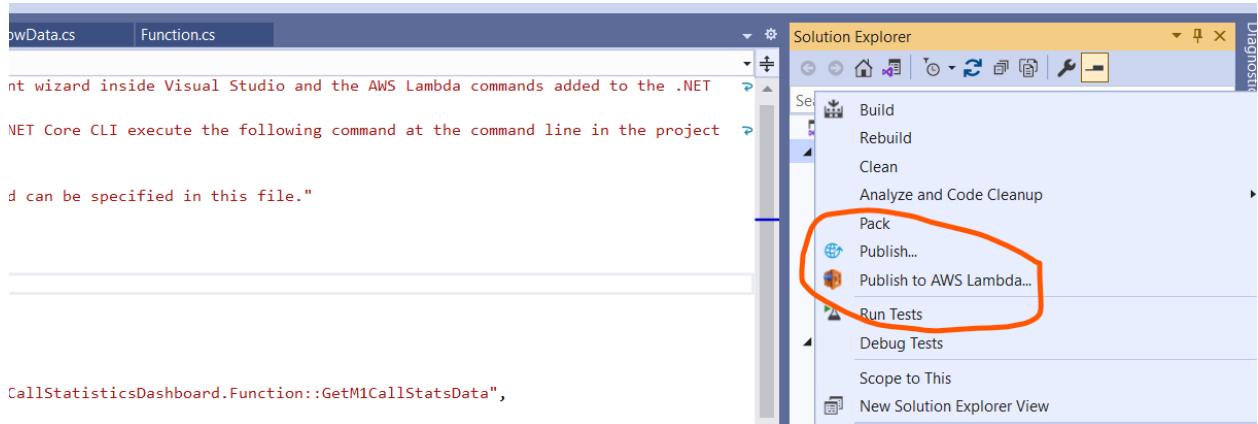
Create table

Steps to create or upload lambda. (If Visual Studio is available)

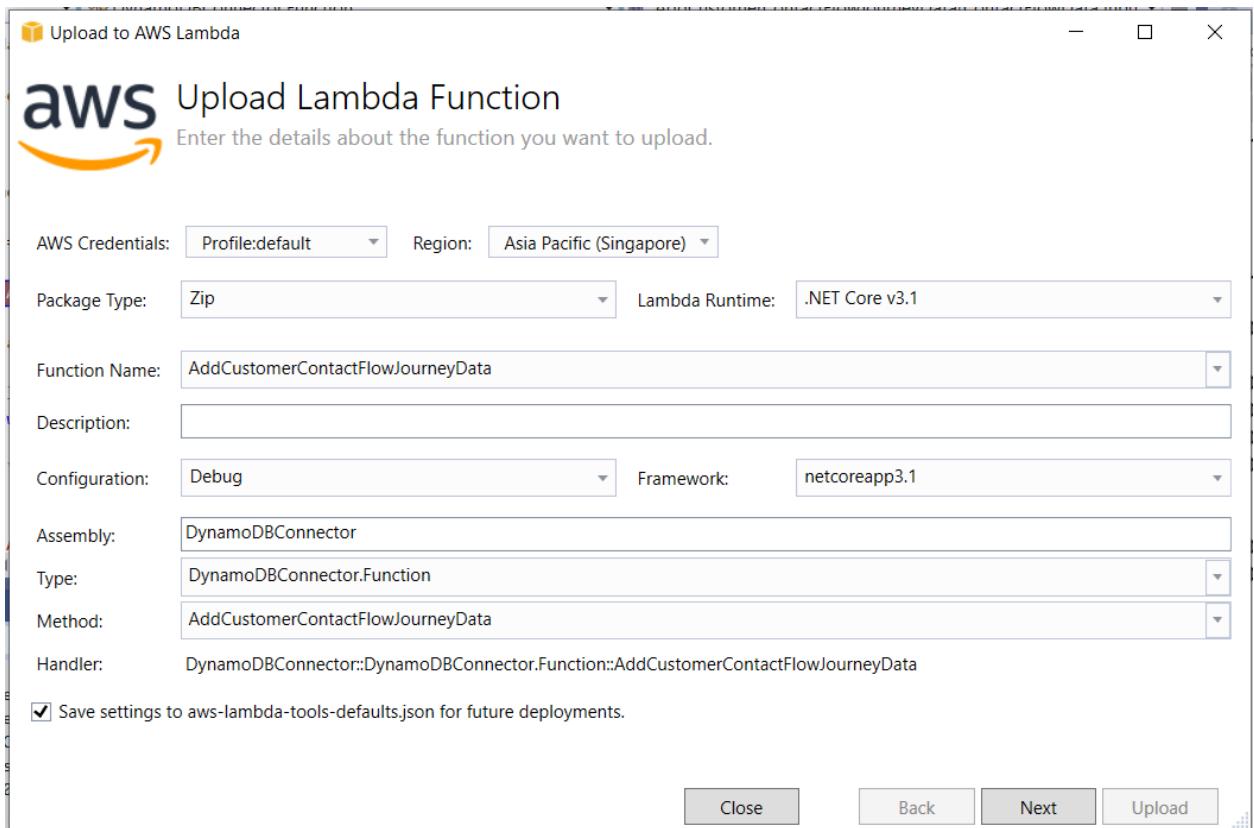
Upload “AddCustomerContactFlowJourneyData“ lambda

- ✓ Open Visual Studio.
- ✓ Load/Open the project of DynamoDBConnector lambda source to Visual Studio.
- ✓ Compile the project.

- ✓ After successful compilation, right click on project and select "Publish to AWSLambda" option from the list.



- ✓ Proceed with steps for uploading function



- ✓ Provide data for below fields properly before proceeding with uploading lambda.

1. **AWS Credentials** – Choose profile where AWS IAM credentials are created. If there is no profile created, follow below link to create profile using

aws configure command.

<https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html>

The following example shows sample values. Replace them with your own values as described in the following sections.

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJAlrXUtnFEMI/K7MDENG/bPxRfCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

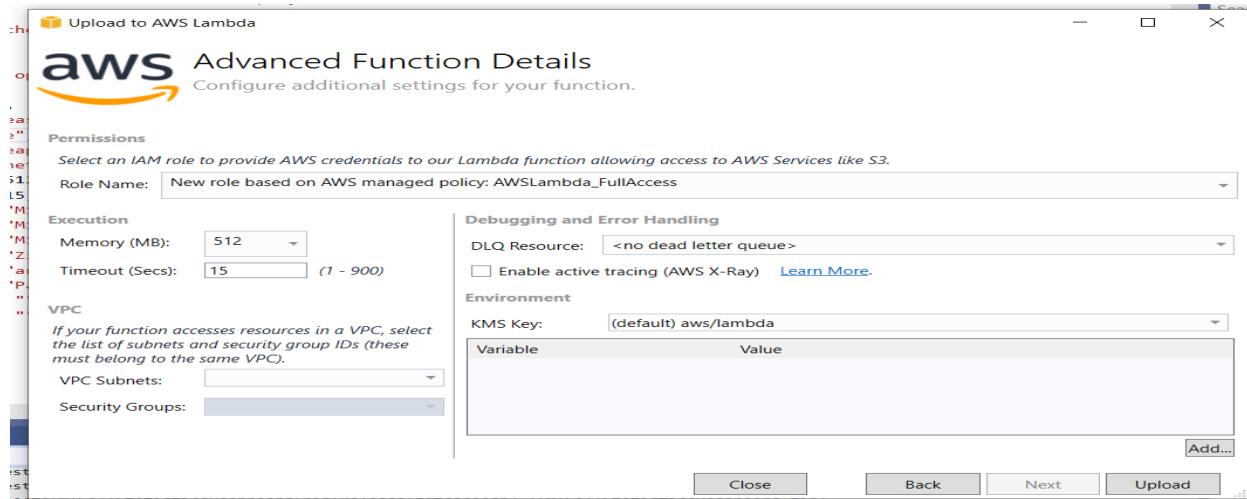
2. **Region** : once you have profile created then region will be displayed in the region field. It will be the same whatever we provide when running aws configure command.
3. **Package Type** : Zip
4. **Lambda RunTime** : .Net Core v3.1
5. **Function Name** : AddCustomerContactFlowJourneyData
6. **Configuration** : Debug/Release
7. **Framework** : netcoreapp3.1
8. **Assembly** : DynamoDBConnector
9. **Type**: DynamoDBConnector.Function
10. **Method** : AddCustomerContactFlowJourneyData

Once we fill all these details, it will display the handler as

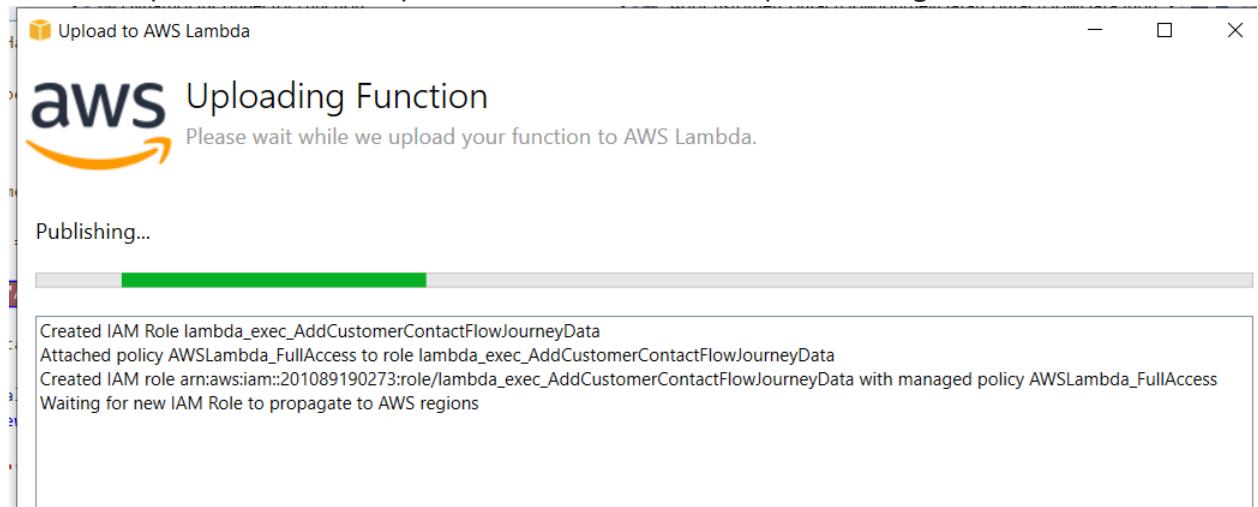
```
DynamoDBConnector::DynamoDBConnector.Function::AddCustomerContactFlowJourneyData
```

Please make sure it has the above handler created.

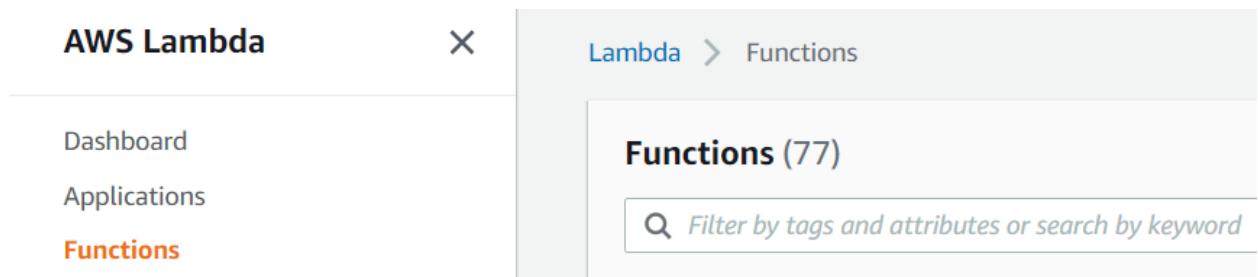
- ✓ In the next page choose the role as "New role based on AWS managed policy: AWSLambda_FullAccess"



- ✓ Click upload button to upload the lambda to the specified region.



- ✓ Navigate to [Lambda console](#) and selection Functions from the left menu then search for the lambda function which we have uploaded just now.



- ✓ It will appear like mentioned below

| Functions (1) | | | | | | |
|----------------------------------|-----------------------------------|-------------|--------------|-------------------------------|-----------|---------------|
| | | Description | Package type | Runtime | Code size | Last modified |
| <input checked="" type="radio"/> | AddCustomerContactFlowJourneyData | - | Zip | .NET Core 3.1 (C#/PowerShell) | 2.5 MB | 15 hours ago |

Steps to create or upload lambda. (If Visual Studio is not available)

Upload “AddCustomerContactFlowJourneyData“ lambda :

- ✓ Download the lambda zip file from the below github link.
- ✓ Navigate to [lambda console](#)
- ✓ Click on Functions from left menu

| AWS Lambda | | Lambda > Functions |
|-----------------------------|--|--|
| Dashboard | Applications | Functions |
| Functions (77) | | |
| Last fetched 12 seconds ago | <input type="button" value="Actions"/> | <input type="button" value="Create function"/> |

- Click on “Create Function” button at the right top corner to create function which will navigate to the page where you can provide details to create lambda function like mentioned below:

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. | Browse serverless app repository Deploy a sample Lambda application from the AWS Serverless Application Repository. |
|--|---|---|---|

Basic information

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

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.

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

▶ Advanced settings

Cancel **Create function**

- ✓ Choose “Author from scratch”
- ✓ Provide function name as “**AddCustomerContactFlowJourneyData**”.
- ✓ Select Runtime as “.NET Core 3.1 (C#/PowerShell)” from the given list.
- ✓

Final options should be like mentioned below:

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. | Browse serverless app repository Deploy a sample Lambda application from the AWS Serverless Application Repository. |
|--|---|---|---|

Basic information

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

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.

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

▶ Advanced settings

Cancel **Create function**

- Leave other values as default and click on “Create Function” from the bottom right corner to proceed.
- Once function is successfully created then you will be able to see below notification

The screenshot shows the AWS Lambda Functions page. At the top, a green banner indicates: "Successfully created the function AddCustomerContactFlowJourneyData. You can now change its code and configuration. To invoke your function with a test event, choose *Test*." Below the banner, the function name "AddCustomerContactFlowJourneyData" is displayed. A navigation bar includes "Function overview" and "Actions". Under the "Code" tab, there are tabs for "Code", "Test", "Monitor", "Configuration", "Aliases", and "Versions". A "Code source" section is present.

- ✓ Since we have successfully created lambda function. By default it will be empty function and now have to upload the code from our repository.
- ✓ Navigate to functions section from Lambda console and search for "**AddCustomerContactFlowJourneyData**" lambda function like mentioned below

The screenshot shows the AWS Lambda Functions list page. On the left, a sidebar lists "Dashboard", "Applications", "Functions", "Additional resources", "Related AWS resources", and "Step Functions state machines". The main area shows a table titled "Functions (78)". A search bar at the top of the table shows "Function name: AddCustomerContactFlowJourneyData" with "1 match". The table columns include "Function name", "Description", "Package type", "Runtime", "Code size", and "Last modified". The "AddCustomerContactFlowJourneyData" row shows "Zip" under Package type, ".NET Core 3.1 (C#/PowerShell)" under Runtime, "532.6 kB" under Code size, and "9 minutes ago" under Last modified.

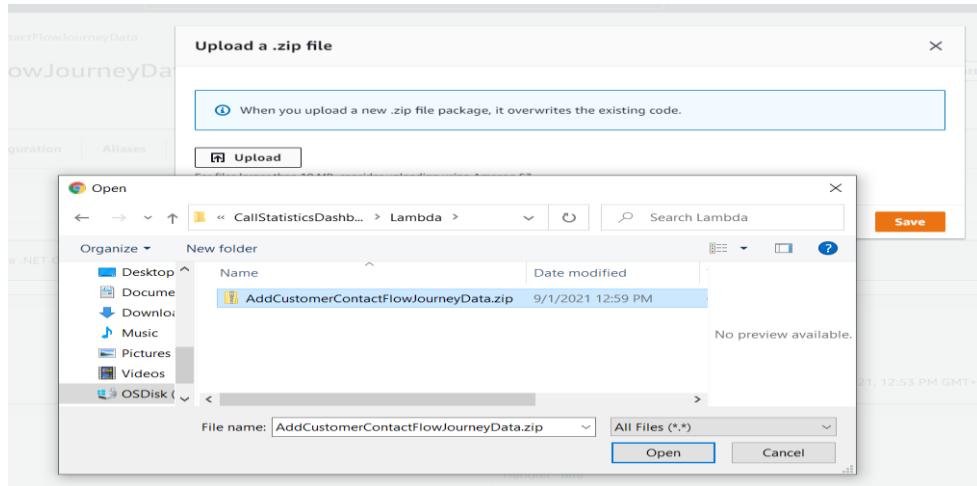
- Click on "**AddCustomerContactFlowJourneyData**" function which will navigate to the page where we can upload code.
- Click ".Zip file" from uploadfile option under the Code tab like mentioned below:

The screenshot shows the AWS Lambda Function details page for "AddCustomerContactFlowJourneyData". The "Code" tab is selected. A callout highlights the "Upload from" button in the "Code source" section, which has options for ".zip file" and "Amazon S3 location".

- It will open page where you can upload lambda source code.

The screenshot shows the "Upload a .zip file" dialog box. It contains a message: "When you upload a new .zip file package, it overwrites the existing code." Below this is an "Upload" button with a file icon. A note says: "For files larger than 10 MB, consider uploading using Amazon S3." At the bottom right are "Cancel" and "Save" buttons.

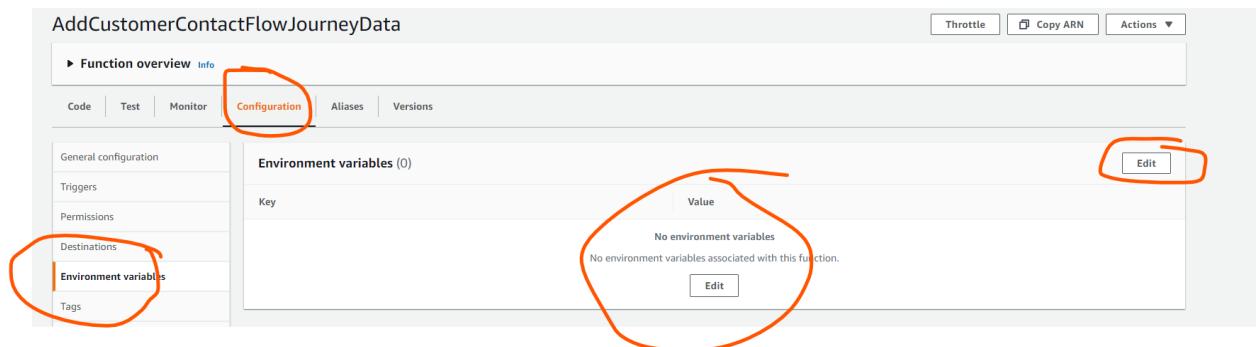
- Click “Upload” button and navigate to the location where lambda is downloaded earlier (Under Lambda folder from the zip file) . And choose the lambda and click on “Save” button to upload like mentioned below:



- After successfully loaded the zip file, we will get success notification. Please wait for some time to upload the function as it will take time.

Configuration section:

- Navigate to function, then click on Configuration tab and select “Environment Variables” like mentioned below:



- By default no variables will be configured. Click on Edit button to add variables.

Lambda > Functions > AddCustomerContactFlowJourneyData > Edit environment variables

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

There are no environment variables on this function.

[Add environment variable](#)

▶ [Encryption configuration](#)

[Cancel](#) [Save](#)

- Click on “Add environment variable” button to add the variables and values as key value pair like mentioned below

Lambda > Functions > AddCustomerContactFlowJourneyData > Edit environment variables

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

| Key | Value | Remove |
|-------------|----------------------------------|------------------------|
| DBTableName | CustomerContactFlowJourney | Remove |
| Region | ap-southeast-1 | Remove |
| IAMUserID | AKIASSUOZRGAS5CORTSFQ | Remove |
| IAMSecret | hvl+tdYe6hLC6LcTpQIPN8BEi7XfmJ+9 | Remove |

[Add environment variable](#)

▶ [Encryption configuration](#)

[Cancel](#) [Save](#)

- Add your IAM UserID & Secret key properly.
- Once you have provided variables, click on save button to save the configuration.
 - After successfully saved you will get notification like mentioned below:

Your changes have been saved.

AddCustomerContactFlowJourneyData

Function overview [Info](#)

Code Test Monitor Configuration Aliases Versions

General configuration

Triggers

Permissions

Destinations

Environment variables (4)

The environment variables below are encrypted at rest with the default Lambda service key.

| Key | Value |
|-------------|--|
| DBTableName | CustomerContactFlow.Journey |
| IAMSecret | hvl+tdYe6hLC6LcTpQIPN8BEi7XfmJ+9RHGozM |
| IAMUserID | AKIASUOZRGAS5CORTSFQ |
| Region | ap-southeast-1 |

Tags

VPC

Throttle [Copy ARN](#) Actions ▾

Change the function handler

- Go to code tab and click on Edit button under Run time settings section like mentioned below

AddCustomerContactFlowJourneyData

Function overview [Info](#)

Code [Edit](#) Test Monitor Configuration Aliases Versions

Code source [Info](#)

The code editor does not support the .NET Core 3.1 (C#/PowerShell) runtime.

Code properties

| | | |
|------------------------|--|--|
| Package size 2.4 MB | SHA256 hash gknio7rnoB1Cb00VqVMYPrmGVsCwRWQa6pipT0l9PY= | Last modified September 1, 2021, 03:17 PM GMT+8 |
|------------------------|--|--|

Runtime settings [Info](#)

| | |
|--|---|
| Runtime .NET Core 3.1 (C#/PowerShell) | Handler Info LambdaTest::LambdaTest.LambdaHandler::handleRequest |
|--|---|

Throttle [Copy ARN](#) Actions ▾

- Change the Handler from

“`LambdaTest::LambdaTest.LambdaHandler::handleRequest`” to

“`DynamoDBConnector::DynamoDBConnector.Function::AddCustomerContactFlowJourneyData`”

Edit runtime settings

Runtime settings [Info](#)

Runtime
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

.NET Core 3.1 (C#/PowerShell)

Handler [Info](#)

DynamoDBConnector::DynamoDBConnector.Function::AddCustomerContactFlowJourneyData

Cancel **Save**

Then click on save button to save the handler settings.

Your changes have been saved.

Lambda > Functions > AddCustomerContactFlowJourneyData

AddCustomerContactFlowJourneyData

Throttle Copy ARN Actions ▾

Test the lambda function section:

- Navigate to Test tab under function overview -> Select template as "Hello-world"

▶ Function overview [Info](#)

Code **Test** Monitor Configuration Aliases Versions

Test event

Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON.

New event

Template

hello-world

Name

Format Save changes **Test**

- Copy below json in the json field to test the function

```
{ "Details": { "ContactData": { "Attributes": {}, "Channel": "VOICE", "ContactId": "3b6a9bc7-bf1b-4cb1-be27-6a20461b4573", "CustomerEndpoint": { "Address": "+6563353529", "Type": "TELEPHONE_NUMBER" }, "CustomerId": null, "Description": null, "InitialContactId": "3b6a9bc7-bf1b-4cb1-be27-6a20461b4573", "InitiationMethod": "INBOUND", "InstanceARN": "arn:aws:connect:ap-southeast-
```

```
1:201089190273:instance/11a61352-4149-408a-b38e-3bde0b584f7f", "LanguageCode": "en-US",
"MediaStreams": { "Customer": { "Audio": null } }, "Name": null, "PreviousContactId":
"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573", "Queue": null, "References": {},
"SystemEndpoint": { "Address": "+6563353529", "Type": "TELEPHONE_NUMBER" } },
"Parameters": { "flowType": "Non-Lex", "Intent": "Non-Lex -> Pressed 1" } }, "Name":
>ContactFlowEvent" }
```

The screenshot shows the 'Test event' configuration page in the AWS Lambda console. At the top, there are buttons for 'Format', 'Save changes', and a large orange 'Test' button. Below these are tabs for 'New event', 'Saved event', and 'Template'. The 'Template' tab is selected, showing a dropdown menu with 'hello-world' and 'MyEventName'. The main area contains a JSON code editor with a line number sidebar on the left. The JSON content is identical to the one shown above, representing a test event for a contact flow function.

- Click on “Test” button to test the function

If the function gets success then you will be able to see below success message.

The screenshot shows the 'Test' results page in the AWS Lambda console. The top navigation bar includes 'Code', 'Test' (which is highlighted in orange), 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The main content area displays a success message: 'Execution result: succeeded (logs)'. It includes a 'Details' section with a link to 'View logs'. Below this, a summary of the execution result is shown: '{ "RetVal": "\\"Success\\"}'.

Upload “SendSMSFromContactFlowToCaller“ lambda :

- ✓ Download the lambda zip file from the below github link.
- ✓ Navigate to [lambda console](#)
- ✓ Click on Functions from left menu

The screenshot shows the 'Functions' list page in the AWS Lambda console. The top navigation bar includes 'AWS Lambda' (with a close button), 'Lambda > Functions', 'Dashboard', 'Applications', and 'Functions' (which is highlighted in orange). The main content area shows a table with 77 rows, titled 'Functions (77)'. A search bar at the bottom allows filtering by tags and attributes. Navigation controls at the bottom right include page numbers (1-8) and a 'Create function' button.

- Click on “Create Function” button at the right top corner to create function which will navigate to the page where you can provide details to create lambda function like mentioned below:

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.

Browse serverless app repository Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

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

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.

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

Advanced settings

Cancel **Create function**

- ✓ Choose “Author from scratch”
- ✓ Provide function name as “**SendSMSFromContactFlowToCaller**”.
- ✓ Select Runtime as “.NET Core 3.1 (C#/PowerShell)” from the given list.

Final options should be like mentioned below:

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.

Browse serverless app repository Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

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

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

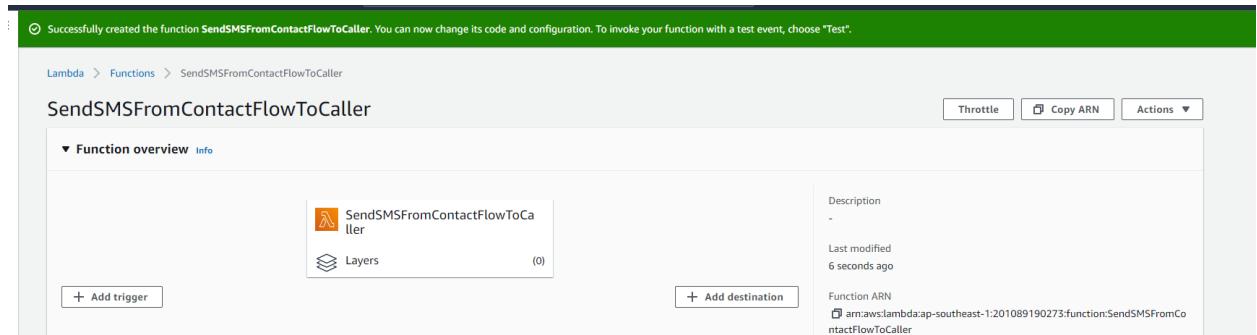
Runtime info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

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

Advanced settings

Cancel **Create function**

- Leave other values as default and click on “Create Function” from the bottom right corner to proceed.
- Once function is successfully created then you will be able to see below notification



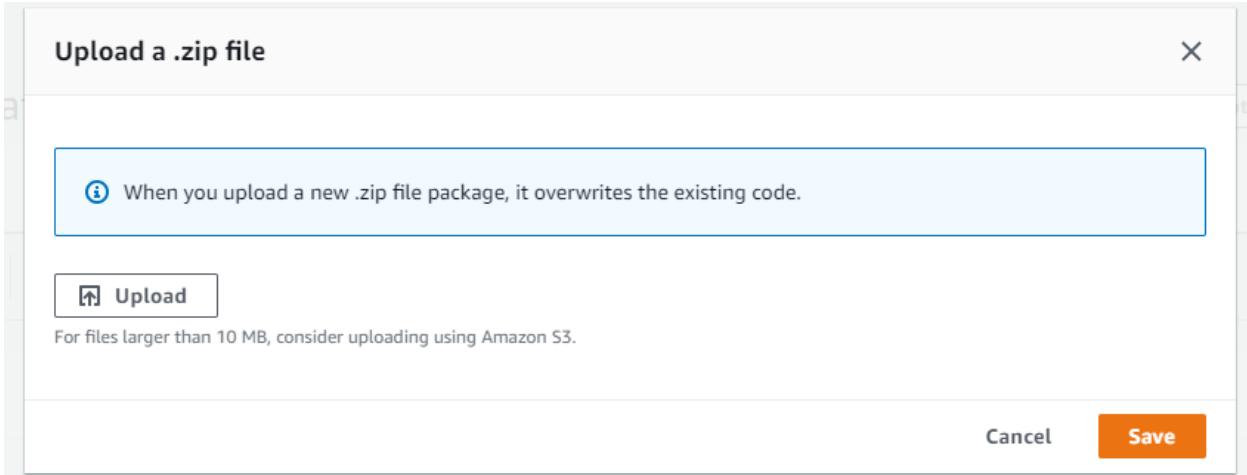
- ✓ Since we have successfully created lambda function. By default it will be empty function and now have to upload the code from our repository.
- ✓ Navigate to functions section from Lambda console and search for “**SendSMSFromContactFlowToCaller**” lambda function like mentioned below

The screenshot shows the Lambda Functions list page with a search bar containing 'SendSMSFromContactFlowToCaller'. The results table shows one match: 'SendSMSFromContactFlowToCaller' with a description of '-' and a runtime of '.NET Core 3.1 (C#/PowerShell)'. The last modified time is 34 seconds ago. The 'Actions' button is visible at the top right of the table.

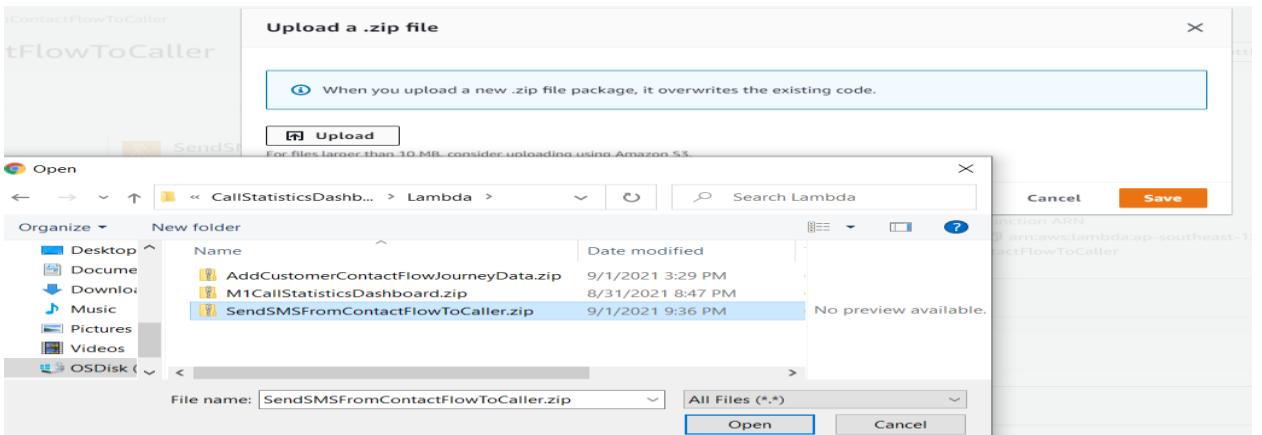
- Click on “**SendSMSFromContactFlowToCaller**” function which will navigate to the page where we can upload code.
- Click “.Zip file” from uploadfile option under the Code tab like mentioned below:

The screenshot shows the 'SendSMSFromContactFlowToCaller' function details page. The 'Code' tab is active. The 'Function overview' section is highlighted with a red circle. The 'Code source' tab is also highlighted with a red circle. A third red circle highlights the 'Upload from' dropdown menu, which contains options for 'zip file' and 'Amazon S3 location'.

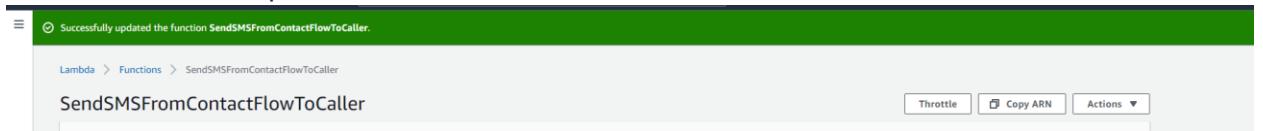
- It will open page where you can upload lambda source code.



- Click “Upload” button and navigate to the location where lambda is downloaded earlier (Under Lambda folder from the zip file choose “SendSMSFromContactFlowToCaller.zip” file) . And choose the lambda and click on “Save” button to upload like mentioned below:



- ✓ After successfully loaded the zip file, we will get success notification. Please wait for some time to upload the function as it will take time.



Create SMS Project in Pinpoint Service:

- ✓ Download the lambda zip file from the below github link.
- ✓ Navigate to **pinpoint console**

- ✓ Click on “Create Project” button which appears in the top-right corner

The screenshot shows the Pinpoint interface with the title 'Pinpoint > All projects'. On the left, there's a sidebar with 'All projects' (which is orange), 'Message templates', and 'Machine learning models'. In the main area, it says 'All projects (2)' with a search bar. At the bottom, there are filters for 'Project name' and 'Creation date'. In the top right, there's a prominent orange 'Create a project' button.

- ✓ Provide project name as “`SendSMSFromContactFlowToCaller`” then click on “Create” button

This screenshot shows a modal dialog titled 'Create a project'. Inside, there's a 'Project name' input field containing 'SendSMSFromContactFlowToCaller', which is underlined with a red wavy line. Below the input field is a note: 'Project names can include up to 64 alphanumeric characters.' At the bottom right of the dialog are two buttons: 'Cancel' and a large orange 'Create' button.

- ✓ Upon successfully creation of project, you will get notification like mentioned below

This screenshot shows a green notification bar at the top of the Pinpoint interface. It says 'You've created SendSMSFromContactFlowToCaller. Now add features to your project.' Below the bar, the navigation path is shown as 'Pinpoint > All projects > SendSMSFromContactFlowT... > Configure features'.

- ✓ Under project features -> SMS -> Click configure button

This screenshot shows the 'Configure features' page. At the top, there's a section titled 'Project features' with a red circle around it. Below it, there are three categories: 'Messaging channels and response metrics'. The first category, 'Email', has a 'Configure' button. The second category, 'SMS', has a 'Configure' button and is also circled in red. The third category, 'Push notifications', also has a 'Configure' button. At the bottom, there's a section for 'Application analytics'.

- ✓ Follow the configuration as mentioned in below screenshot and click save changes

Set up SMS

General settings

Enable the SMS channel for this project

▼ Account-level settings

The settings in this section apply to all SMS messages that you send from your AWS account, including messages that you send using other AWS services.

Default message type
The type of messages you plan to send from this project. [Info](#)

Transactional
Time-sensitive content, such as one-time passcodes.

Promotional
Non-critical content, such as marketing messages.

Account spending limit
The maximum amount of money, in USD, that you want to spend sending SMS messages each month. The limit for accounts in the sandbox is 1 (\$1.00). [Info](#)

1

The spend limit that you specify can't include decimals. The minimum value is 0, and the maximum value is 200.

(Optional) Account sender ID
The identity that appears on recipients' devices when they receive this message. Support varies by country or region. [Info](#)

Enter a valid sender ID

Your sender ID can contain up to 11 alphanumeric or hyphen (-) characters. It has to contain at least one letter, and it can't consist only of numbers. It has to start and end with an alphanumeric character. Some countries and regions may have additional restrictions.

► Advanced configurations - optional

[Cancel](#) [Save changes](#)

- ✓ Navigate to projects and copy the Project ID which we are going to use in lambda function later in this section

| Project name | Project ID | Creation date |
|--------------------------------|----------------------------------|-----------------------------------|
| Campaign SMS | 6765fe03ba6a46fd91057bd3e4bf5fb5 | March 11th 2021, 04:57 AM, UTC |
| testproject | 941e3e343a074e27ae4f6be2f6fb55d | September 1st 2021, 01:33 PM, UTC |
| SendSMSFromContactFlowToCaller | 3b458df1da345ff4738faeff4f6031 | September 1st 2021, 02:21 PM, UTC |

Configuration section:

- Navigate to function, then click on Configuration tab and select “Environment Variables” like mentioned below:

The screenshot shows the AWS Lambda Function Overview page for a function named "SendSMSFromContactFlowToCaller". The "Configuration" tab is active. At the top left, there is a "Function overview" link with a red circle around it. In the main content area, under "Environment variables (0)", there is an "Edit" button with a red circle around it.

- By default no variables will be configured. Click on Edit button to add variables.

The screenshot shows the "Edit environment variables" page for a function named "AddCustomerContactFlowJourneyData". At the top, the navigation path is: Lambda > Functions > AddCustomerContactFlowJourneyData > Edit environment variables. The main section is titled "Environment variables" and contains the following text: "You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more". Below this, it says "There are no environment variables on this function." There is a "Add environment variable" button with a red circle around it. At the bottom right, there are "Cancel" and "Save" buttons.

- Click on “Add environment variable” button to add the variables and values as key value pair like mentioned below

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

| Key | Value | Remove |
|-------------------|-----------------------------------|------------------------|
| IAMSecret | hvl+tdYe6hLC6LCcTpQIPN8BEi7XfmJ+9 | Remove |
| IAMUserID | AKIAS5UOZRGA5CORTSFQ | Remove |
| MessageType | PROMOTIONAL | Remove |
| PinpointProjectID | 941e3e343a074e27ae4f6be2bf6ab35d | Remove |
| Region | ap-southeast-1 | Remove |

[Add environment variable](#)

[▶ Encryption configuration](#)

[Cancel](#) [Save](#)

Add your IAM UserID & Secret key properly. PinpointProjectID – enter the value which you have saved earlier when we were creating pinpoint project.

- Once you have provided variables, click on save button to save the configuration.
- After successfully saved you will get notification like mentioned below:

The screenshot shows the AWS Lambda Function Overview page for the function "SendSMSFromContactFlowToCaller". The "Configuration" tab is selected. In the "Environment variables" section, there are five entries:

| Key | Value |
|-------------------|--|
| IAMSecret | hvl+tdYe6hLC6LCcTpQIPN8BEi7XfmJ+9IRHGoZM |
| IAMUserID | AKIAS5UOZRGA5CORTSFQ |
| MessageType | PROMOTIONAL |
| PinpointProjectID | 941e3e343a074e27ae4f6be2bf6ab35d |
| Region | ap-southeast-1 |

Change the function handler

- Go to code tab and click on Edit button under Run time settings section like mentioned below

The screenshot shows the AWS Lambda function configuration interface. The top navigation bar has tabs: Code (highlighted with a red circle), Test, Monitor, Configuration, Aliases, and Versions. Below the tabs, there's a 'Code source' section with an 'Upload from' button. A note says 'The code editor does not support the .NET Core 3.1 (C#/PowerShell) runtime.' Under 'Code properties', it shows Package size (2.4 MB), SHA256 hash (8UgFMIOgz0uNBq7vJ3ipQSeN2hTYdOcgZlakBxcx0=), and Last modified (September 1, 2021, 09:58 PM GMT+8). In the 'Runtime settings' section, the Runtime is set to '.NET Core 3.1 (C#/PowerShell)'. The Handler is set to 'LambdaTest::LambdaTest.LambdaHandler::handleRequest'. An 'Edit' button is located next to the Handler field, which is also highlighted with a red circle.

- Change the Handler from
“`LambdaTest::LambdaTest.LambdaHandler::handleRequest`” to
“`SendSMSFromContactFlowToCaller::SendSMSFromContactFlowToCaller.Function::SendSMSFromContactFlowToCaller`”

The screenshot shows the 'Edit runtime settings' dialog. At the top, it says 'Lambda > Functions > SendSMSFromContactFlowToCaller > Edit runtime settings'. The main title is 'Edit runtime settings'. The 'Runtime settings' section includes a 'Runtime' dropdown set to '.NET Core 3.1 (C#/PowerShell)' and a 'Handler' input field containing 'SendSMSFromContactFlowToCaller::SendSMSFromContactFlowToCaller.Function::Sen...'. At the bottom right, there are 'Cancel' and 'Save' buttons, with 'Save' highlighted with a red circle.

Then click on save button to save the handler settings.

The screenshot shows the AWS Lambda function configuration interface again. A green success message at the top says 'Your changes have been saved.' Below it, the navigation path is 'Lambda > Functions > SendSMSFromContactFlowToCaller'.

Test the lambda function section:

- Navigate to Test tab under function overview -> Select template as "Hello-world"

The screenshot shows the AWS Lambda Function Overview page. At the top, there are tabs for 'Function overview' and 'Info'. Below these are tabs for 'Code', 'Test' (which is highlighted with an orange circle), 'Monitor', 'Configuration', 'Aliases', and 'Versions'. Under the 'Test' tab, there is a section titled 'Test event'. It contains a dropdown menu labeled 'Template' with 'hello-world' selected (also highlighted with an orange circle). At the bottom right of this section are buttons for 'Format', 'Save changes', and 'Test'.

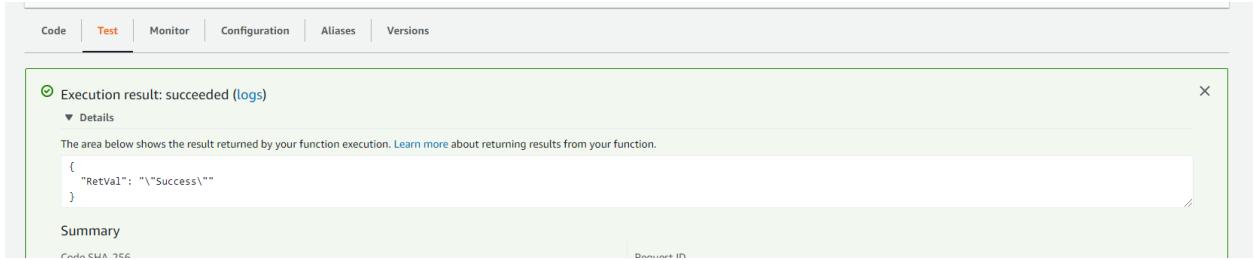
- Copy below json in the json field to test the function. (change your mobile number in two places where highlighted)

```
{
  "Details": {
    "ContactData": {
      "Attributes": {},
      "Channel": "VOICE",
      "ContactId": "c9edb5bb-68ea-45c1-919c-1b823b02c1ef",
      "CustomerEndpoint": {
        "Address": "+6563353529",
        "Type": "TELEPHONE_NUMBER"
      },
      "CustomerId": null,
      "Description": null,
      "InitialContactId": "c9edb5bb-68ea-45c1-919c-1b823b02c1ef",
      "InitiationMethod": "INBOUND",
      "InstanceARN": "arn:aws:connect:ap-southeast-1:201089190273:instance/11a61352-4149-408a-b38e-3bde0b584f7f",
      "LanguageCode": "en-US",
      "MediaStreams": {
        "Customer": {
          "Audio": null
        },
        "Name": null,
        "PreviousContactId": "c9edb5bb-68ea-45c1-919c-1b823b02c1ef",
        "Queue": null,
        "References": {}
      },
      "SystemEndpoint": {
        "Address": "+6563353529",
        "Type": "TELEPHONE_NUMBER"
      }
    },
    "Parameters": {
      "flowType": null,
      "message": "Message from M1 : Please use below weblink to enjoy with our chat service. https://ec2-52-77-232-11.ap-southeast-1.compute.amazonaws.com:9000/"
    },
    "Name": "ContactFlowEvent"
  }
}
```

The screenshot shows the AWS Lambda Function Overview page with the 'Test' tab selected. In the 'Test event' section, there is a 'MyEventName' input field and a large text area containing the JSON code shown above. The text area has line numbers from 1 to 27 on the left. The 'Test' button is located at the top right of the 'Test event' section.

- Click on "Test" button to test the function

If the function gets success then you will be able to see below success message.

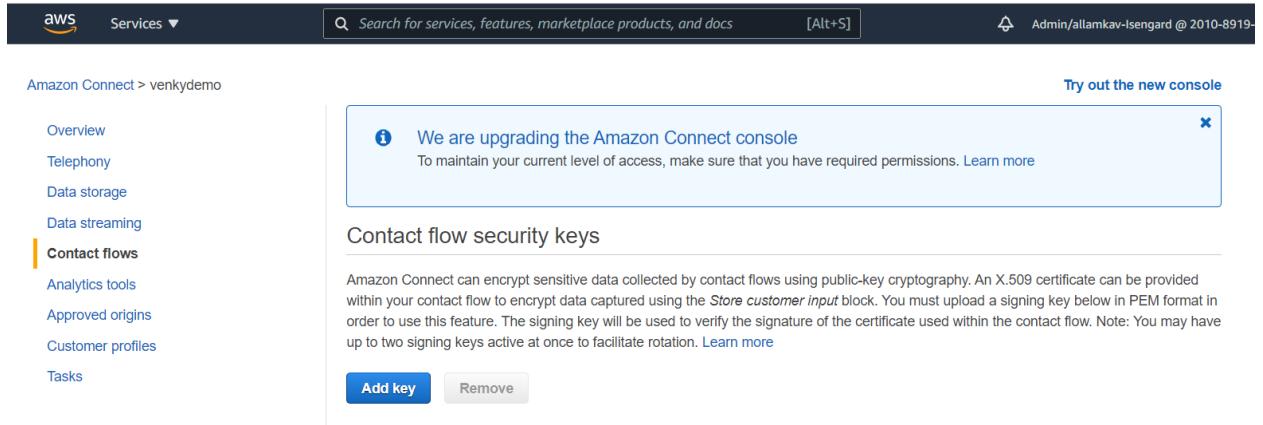


The screenshot shows the AWS Lambda function test interface. The 'Test' tab is selected. In the main area, there is a green box indicating a successful execution with the message: "Execution result: succeeded (logs)". Below this, a 'Details' section shows the returned JSON object: {"RetVal": "\"Success\""}.

You should be able to receive SMS.

Configure lambda to amazon connect instance

- ✓ Go to Amazon Connect
- ✓ open the instance which you are using for contact flow
- ✓ Navigate to Contact flows from the left menu like mentioned below:



The screenshot shows the Amazon Connect Contact flows page. The left sidebar has a highlighted 'Contact flows' option under the 'Contact flows' heading. The main content area displays a message: "We are upgrading the Amazon Connect console. To maintain your current level of access, make sure that you have required permissions. Learn more". Below this message, there is a section titled "Contact flow security keys" with a "Add key" button and a "Remove" button.

- ✓ Go to AWS Lambda section from the right menu and choose the function (**AddCustomerContactFlowJourneyData**) from the list and then click on “+Add Lambda Function” button to add to connect instance.

AWS Lambda

Amazon Connect can interact with your own systems and take different paths in IVR dynamically. To achieve this, invoke AWS Lambda functions in contact flows to interact with your own systems or other services, then build personalized and dynamic experiences based on data returned.

Note: By adding Lambda functions, you are granting Amazon Connect permission to invoke them [Create a new Lambda function](#)

The screenshot shows the AWS Lambda Functions list. At the top, there is a search bar labeled "Function" with "Select" and a dropdown arrow, followed by a "+ Add Lambda Function" button. Below the search bar, the heading "Lambda Functions" is displayed. A single function entry is listed:

| AddCustomerContactFlowJourneyData | arn:aws:lambda:ap-southeast-1:201089190273:function:AddCustomerContactFlowJourneyData | | |
|-----------------------------------|---|--|--|
| a | | | |

- ✓ Follow and repeat the same above instructions to add "SendSMSFromContactFlowToCaller" lambda to connect instance.

The screenshot shows the AWS Lambda Functions list. At the top, there is a search bar labeled "Function" with "Select" and a dropdown arrow, followed by a "+ Add Lambda Function" button. Below the search bar, the heading "Lambda Functions" is displayed. Four function entries are listed:

| AddCustomerContactFlowJourneyData | arn:aws:lambda:ap-southeast-1:201089190273:function:AddCustomerContactFlowJourneyData | | |
|---|---|--|--|
| a | | | |
| SendSMSToMyMobileNumber | arn:aws:lambda:ap-southeast-1:201089190273:function:SendSMSToMyMobileNumber | | |
| serverlessrepo-AmazonConnectSalesforce-sfInvokeAPI-dnuVvW8keJcw | arn:aws:lambda:ap-southeast-1:201089190273:function:serverlessrepo-AmazonConnectSalesforce-sfInvokeAPI-dnuVvW8keJcw | | |
| SendSMSFromContactFlowToCaller | arn:aws:lambda:ap-southeast-1:201089190273:function:SendSMSFromContactFlowToCaller | | |
| | | | |

Upload “M1CallStatisticsDashboard” Lambda : This lambda will be used to fetch the data from DynamoDB for displaying in the UI dashboard.

- ✓ Download the lambda zip file from the below github link.
- ✓ Navigate to [lambda console](#)
- ✓ Click on Functions from left menu

The screenshot shows the AWS Lambda Functions list. At the top, there is a search bar labeled "Lambda" with "Functions" and a dropdown arrow, followed by a "+ Add Lambda Function" button. Below the search bar, the heading "Functions (77)" is displayed. A "Last fetched 12 seconds ago" message is shown. On the right, there are "Actions" and a "Create function" button. Below the heading, there is a search bar with the placeholder "Filter by tags and attributes or search by keyword". At the bottom, there is a pagination bar with numbers 1 through 8 and a refresh icon.

- ✓ Click on “Create Function” button at the right top corner to create function which will navigate to the page where you can provide details to create lambda function like mentioned below:

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.
- Browse serverless app repository** Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

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

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

▶ Advanced settings

Cancel **Create function**

- ✓ Choose “Author from scratch”
- ✓ Provide function name as “**M1CallStatisticsDashboard**”.
- ✓ Select Runtime as “.NET Core 3.1 (C#/PowerShell)” from the given list.

Final options should be like mentioned below:

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.
- Browse serverless app repository** Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.
`M1CallStatisticsDashboard`
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.
`.NET Core 3.1 (C#/PowerShell)`

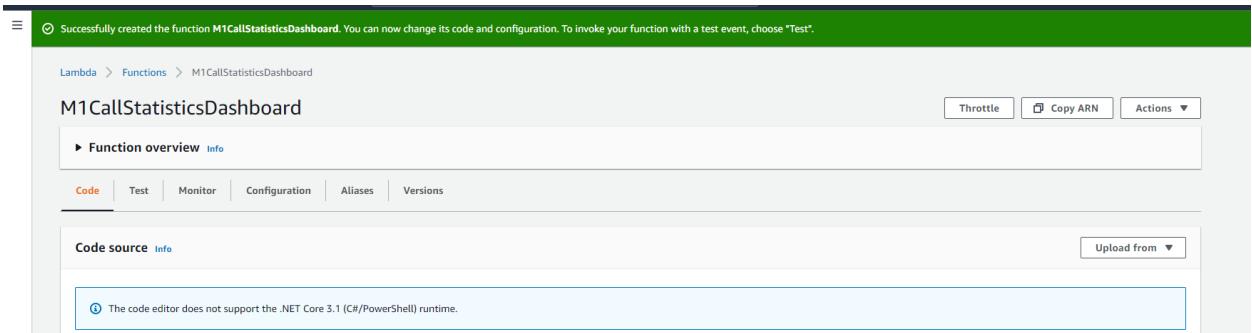
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▶ Change default execution role

▶ Advanced settings

Cancel **Create function**

- Leave other values as default and click on “Create Function” from the bottom right corner to proceed.
- Once function is successfully created then you will be able to see below notification

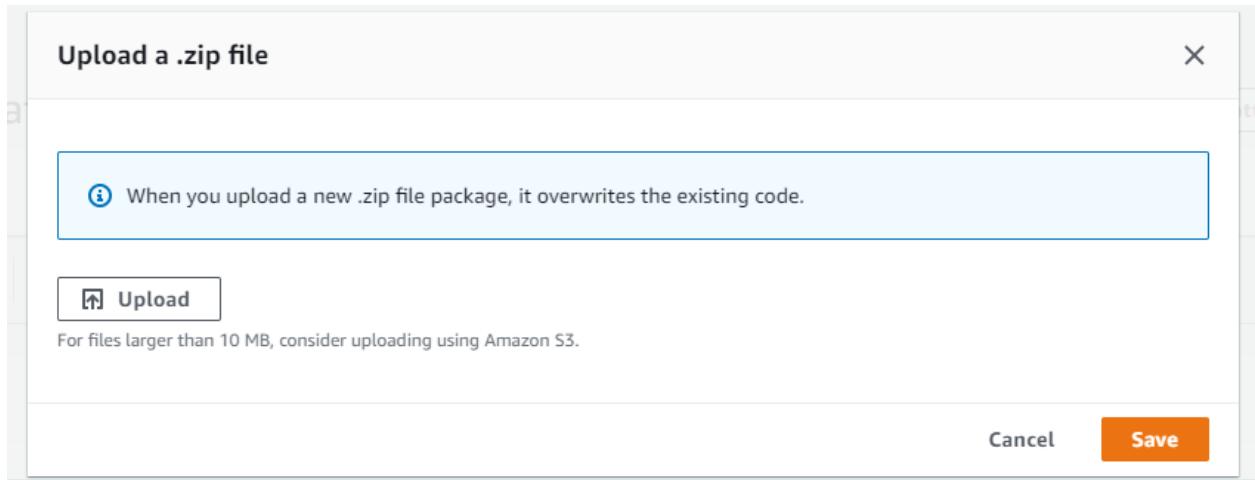


- ✓ Since we have successfully created lambda function. By default it will be empty function and now have to upload the code from our repository.
- ✓ Navigate to functions section from Lambda console and search for “**M1CallStatisticsDashboard**” lambda function like mentioned below

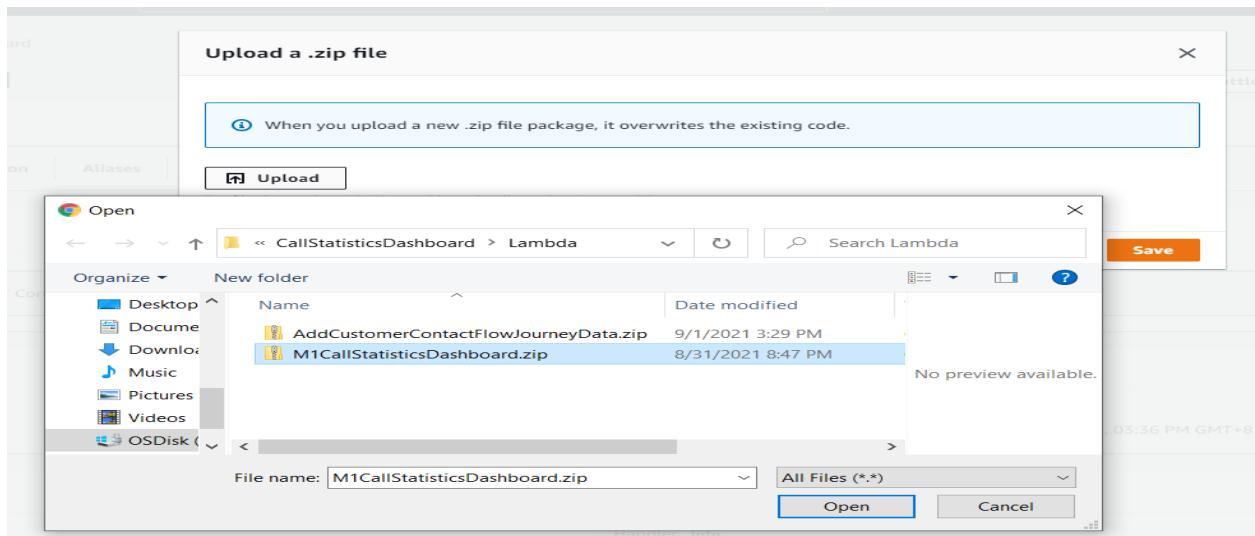
| Function name | Description | Package type | Runtime | Code size | Last modified |
|---------------------------|-------------|--------------|-------------------------------|-----------|---------------|
| M1CallStatisticsDashboard | - | Zip | .NET Core 3.1 (C#/PowerShell) | 532.6 kB | 1 minute ago |

- Click on “**M1CallStatisticsDashboard**” function which will navigate to the page where we can upload code.
- Click “.Zip file” from uploadfile option under the Code tab like mentioned below:

- It will open page where you can upload lambda source code.



- Click "Upload" button and navigate to the location where lambda is downloaded earlier (Under Lambda folder). And choose the lambda and click on "Save" button to upload like mentioned below:



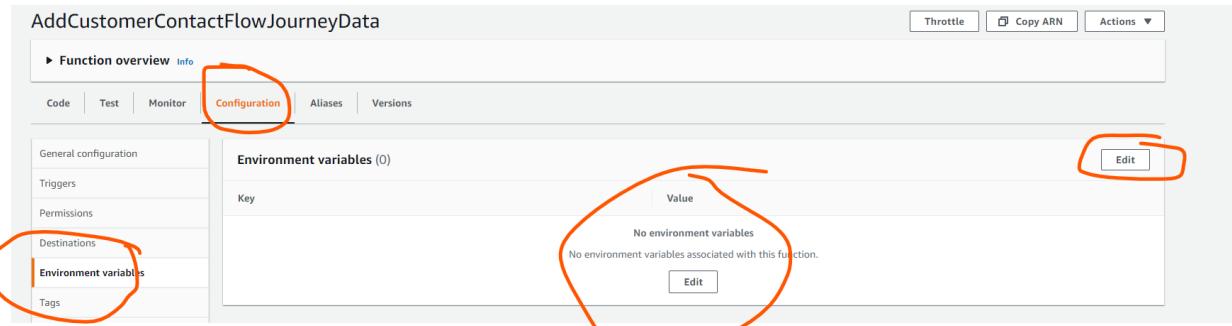
- After successfully loaded the zip file, we will get success notification. Please wait for some time to upload the function as it will take time.

✓



Configuration section:

- Navigate to function, then click on Configuration tab and select "Environment Variables" like mentioned below:



- By default no variables will be configured. Click on Edit button to add variables.

Lambda > Functions > AddCustomerContactFlowJourneyData > Edit environment variables

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

There are no environment variables on this function.

[Add environment variable](#)

▶ Encryption configuration

[Cancel](#) [Save](#)

- Click on “Add environment variable” button to add the variables and values as key value pair like mentioned below

Lambda > Functions > AddCustomerContactFlowJourneyData > Edit environment variables

Edit environment variables

| Environment variables | | |
|--|-----------------------------------|---|
| You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more | | |
| Key | Value | |
| DBTableName | CustomerContactFlowJourney | Remove |
| Region | ap-southeast-1 | Remove |
| IAMUserID | AKIASSUOZRGA5CORTSFQ | Remove |
| IAMSecret | hv1+tdYe6hLC6LccTpQlPN8BEi7XfmJ+9 | Remove |
| Add environment variable | | |
| ▶ Encryption configuration | | |
| | | Cancel Save |

Configure your IAM UserID & Secret key properly.

- Once you have provided variables, click on save button to save the configuration.
- After successfully saved you will get notification like mentioned below:

Your changes have been saved.

Lambda > Functions > AddCustomerContactFlowJourneyData

AddCustomerContactFlowJourneyData

Throttle Copy ARN Actions ▾

Function overview [Info](#)

Code Test Monitor Configuration Aliases Versions

General configuration Triggers Permissions Destinations Environment variables Tags VPC

Environment variables (4)

The environment variables below are encrypted at rest with the default Lambda service key.

| Key | Value | Edit |
|-------------|--|----------------------|
| DBTableName | CustomerContactFlowJourney | |
| IAMSecret | hv1+tdYe6hLC6LccTpQlPN8BEi7XfmJ+9IRHGozM | |
| IAMUserID | AKIASSUOZRGA5CORTSFQ | |
| Region | ap-southeast-1 | |

Change Function handler section

- Go to code tab and click on Edit button under Run time settings section like mentioned below

M1CallStatisticsDashboard

Function overview Info

Code Test Monitor Configuration Aliases Versions

Code source Info

The code editor does not support the .NET Core 3.1 (C#/PowerShell) runtime.

Upload from ▼

Code properties

| | | |
|------------------------|--|--|
| Package size 2.5 MB | SHA256 hash KUmZOjEy9DoF+29XbkZE4H7iLxADO/OlbPLUNBiqKY= | Last modified September 1, 2021, 05:41 PM GMT+8 |
|------------------------|--|--|

Runtime settings Info

Runtime .NET Core 3.1 (C#/PowerShell)

Handler Info
LambdaTest:::LambdaTest.LambdaHandler::handleRequest

Edit

- Change the Handler from
“LambdaTest:::LambdaTest.LambdaHandler::handleRequest” to
“M1CallStatisticsDashboard::M1CallStatisticsDashboard.Function::GetM1CallStatsData”

Lambda > Functions > M1CallStatisticsDashboard > Edit runtime settings

Edit runtime settings

Runtime settings Info

Runtime
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
.NET Core 3.1 (C#/PowerShell)

Handler Info
M1CallStatisticsDashboard::M1CallStatisticsDashboard.Function::GetM1CallStatsData

Cancel Save

Then click on save button to save the handler settings.

Your changes have been saved.

Lambda > Functions > M1CallStatisticsDashboard

M1CallStatisticsDashboard

Test the lambda function section:

- Navigate to Test tab under function overview -> Select template as "Hello-world"

Function overview Info

Code Test Monitor Configuration Aliases Versions

Test event

Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON.

New event

Saved event

Template

hello-world

Name

Format Save changes Test

- No need to change the json

Test event

Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON.

New event

Saved event

Template

hello-world

Name

MyEventName

```
1- [{}  
2-   "key1": "value1",  
3-   "key2": "value2",  
4-   "key3": "value3"  
5- ]
```

Format Save changes Test

- Click on "Test" button to test the function

If the function gets success then you will be able to see below success message.

Execution result: succeeded (logs)

Details

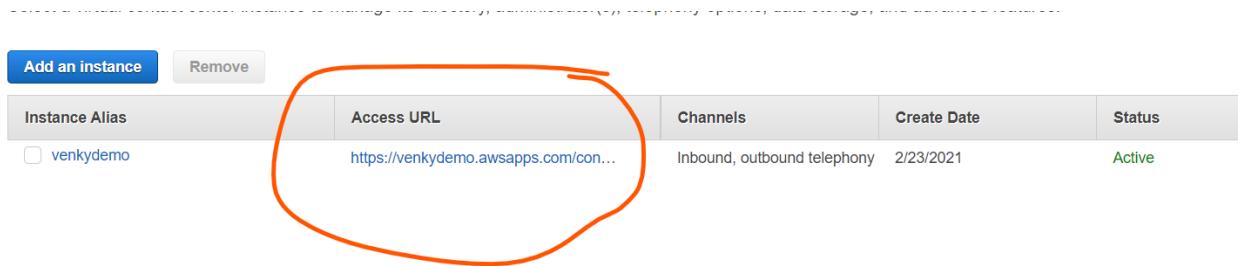
The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
{
  "statusCode": 200,
  "body": "{\"eventName\":\"OnGetM1CallStatsData\", \"channel\":null, \"contactID\":null, \"eventInfo\":[{\"ID\":7, \"Channel\":\"VOICE\", \"ContactID\":\"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\", \"CustomerNumber\":\"+6563353529\", \"FlowType\":\"Non-Lex\", \"InitialContactID\":\"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\", \"InteractionDateTime\":\"202109081073120\", \"CustomerJourneyFlow\":\"Non-Lex - Pressed 1\"}, {\"ID\":3, \"Channel\":\"VOICE\", \"ContactID\":\"b1faaa84-6513-4a17-8d1d-c26d667255f0\", \"CustomerNumber\":\"+6563353529\", \"FlowType\":\"Agent Queue\", \"InitialContactID\":\"b1faaa84-6513-4a17-8d1d-c26d667255f0\", \"InteractionDateTime\":\"20210831125231\", \"CustomerJourneyFlow\":\"BroadbandEnquiries\"}, {\"ID\":2, \"Channel\":\"VOICE\", \"ContactID\":\"075d6481-b7dd-497d-beda-ceb3a63c490\", \"CustomerNumber\":\"+6563353529\", \"FlowType\":\"Queue\", \"InitialContactID\":\"075d6481-b7dd-497d-beda-ceb3a63c490\", \"InteractionDateTime\":\"20210831125139\", \"CustomerJourneyFlow\":\"BroadbandEnquiries\"}, {\"ID\":4, \"Channel\":\"VOICE\", \"ContactID\":\"a52479ae-427b-49e1-b924-b0b41df922b7\", \"CustomerNumber\":\"+6563353529\", \"FlowType\":\"SMS-Deflect\", \"InitialContactID\":\"a52479ae-427b-49e1-b924-
```

Summary

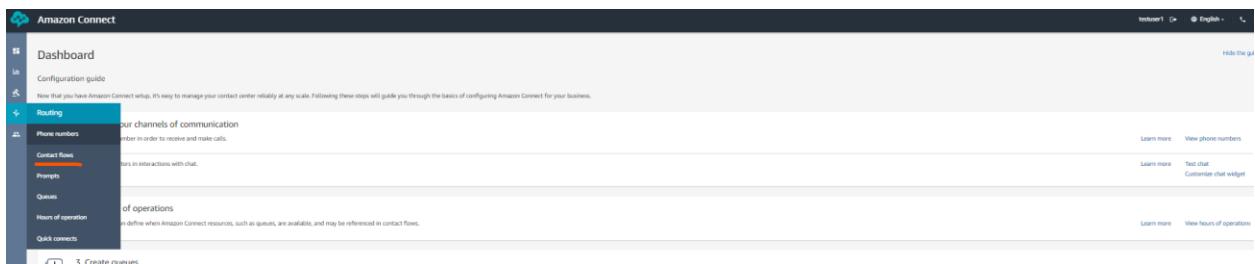
Steps to import contact flow to Amazon Connect instance.

1. Go to the [Amazon Connect console](#), open the instance URL you are using like mentioned below:

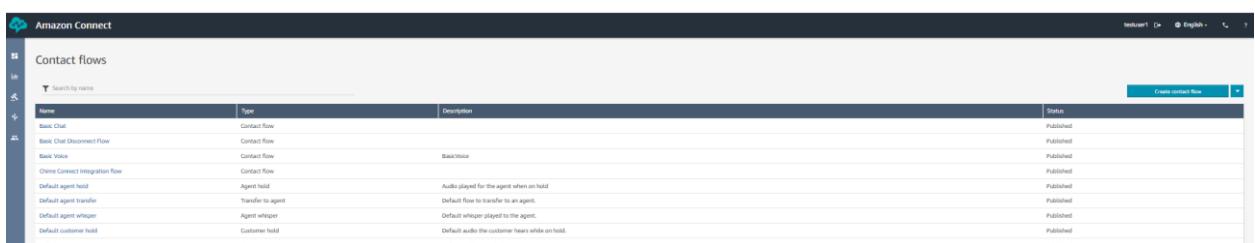


| Add an instance | Remove | Instance Alias | Access URL | Channels | Create Date | Status |
|-----------------|--------|----------------|--------------------------------------|-----------------------------|-------------|--------|
| | | venkydemo | https://venkydemo.awsapps.com/con... | Inbound, outbound telephony | 2/23/2021 | Active |

2. Login to Amazon connect dashboard with the user which you have created while setting up connect instance.
3. Navigate to Routing -> Contact Flows from the left menu.



You will be able to see the existing default contact flows which comes along with connect instance.



| Name | Type | Description | Status |
|----------------------------------|-------------------|---|-------------|
| Basic Chat | Contact Flow | | Published |
| Basic Chat Disconnect Flow | Contact Flow | | Published |
| Basic Voice | Contact Flow | Basic voice | Published |
| Connect Connect Integration Flow | Contact Flow | | Published |
| Default agent hold | Agent hold | Audio played for the agent when on hold. | Published |
| Default agent transfer | Transfer to agent | Default flow to transfer to an agent. | Published |
| Default agent whisper | Agent whisper | Default whisper played to the agent. | Published |
| Default customer hold | Customer hold | Default audio the customer hears while on hold. | Published |
| Default customer transfer | Customer transfer | Default audio played when a customer is transferred to another. | Unpublished |

4. Click on the "Create contact flow" button which appears on the top-right corner.

The screenshot shows the Amazon Connect Contact flows interface. At the top, there's a header with the Amazon Connect logo, user information (testuser1, English), and a help icon. Below the header is a search bar labeled "Search by name". A blue button on the right says "Create contact flow". The main area is titled "Contact flows" and contains a table with columns: Name, Type, Description, and Status. There are several contact flows listed, including "M1ConnectCallStatistics" which is highlighted.

5. Open the unzipped folder and navigate to ContactFlow folder to see contactFlow file

The screenshot shows a file explorer interface with a breadcrumb navigation bar at the top: "« ConnectCallStatisticsDashboard-main > ContactFlow". Below is a table with columns: Name, Date modified, Type, and Size. A single file, "M1ConnectCallStatistics", is listed with the details: Date modified 9/1/2021 7:23 PM, Type File, and Size 18 KB.

6. In the new contact flow designer, select icon next to save button to import the call flow.

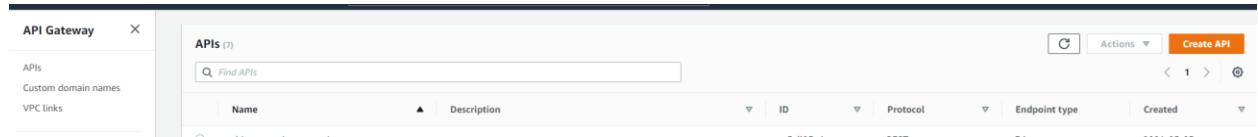
The screenshot shows the Amazon Connect Contact flow designer interface. On the right, there's a toolbar with buttons for Publish, Save, and Import flow (beta). A dropdown menu is open over the Save button, showing options "Save as" and "Import flow (beta)". The main workspace shows a partial contact flow with nodes like "Entry point", "Start", and "Set logging behavior".

7. Click on Import flow and choose the "M1ConnectCallStatistics" call flow from ContactFlow folder. Then call flow will imported. Click on "Save" to save the call flow & click on "Publish" to publish the changes for your testing.

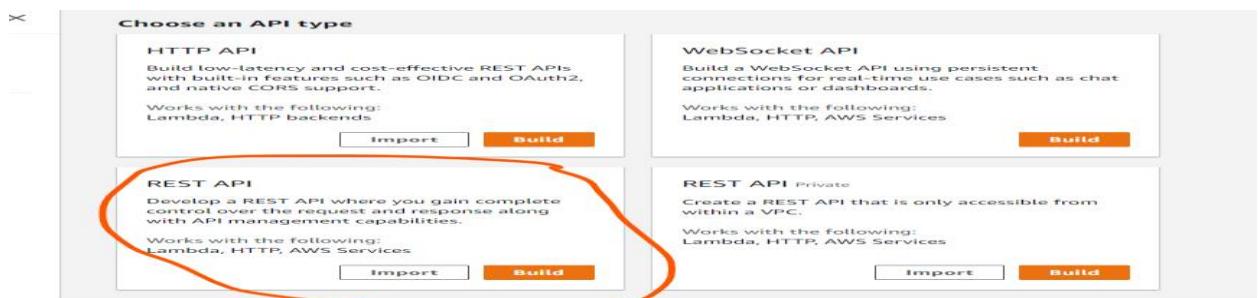
The screenshot shows the Amazon Connect Contact flow designer interface with the imported "M1ConnectCallStatistics" flow. The flow consists of several nodes: "Entry point", "Start", "Set logging behavior", "Play prompt", "Get customer input", "Play prompt", "Set contact attributes", and "Get customer input". The "Get customer input" node has a dropdown menu showing options like "MobileEnquiries", "BroadbandEnquiries", and "Default". The "Set contact attributes" node also has a dropdown menu. On the right, there's a toolbar with "Latest Published", "Publish", and "Save" buttons. The status bar at the bottom right shows "Latest Published" and "Save".

Creating REST API in APIGateway to access lambda from public

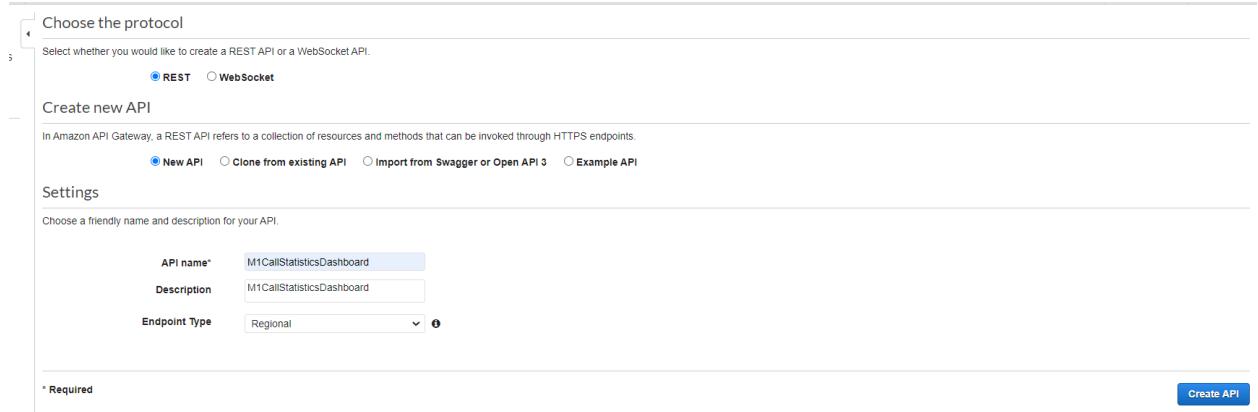
- Navigate to [Api gateway](#) in console
- Click on “Create API” button which appears in the top right corner



- Choose API type as REST API



- Click on build button
- Provide values like mentioned in the screenshot



Choose REST API

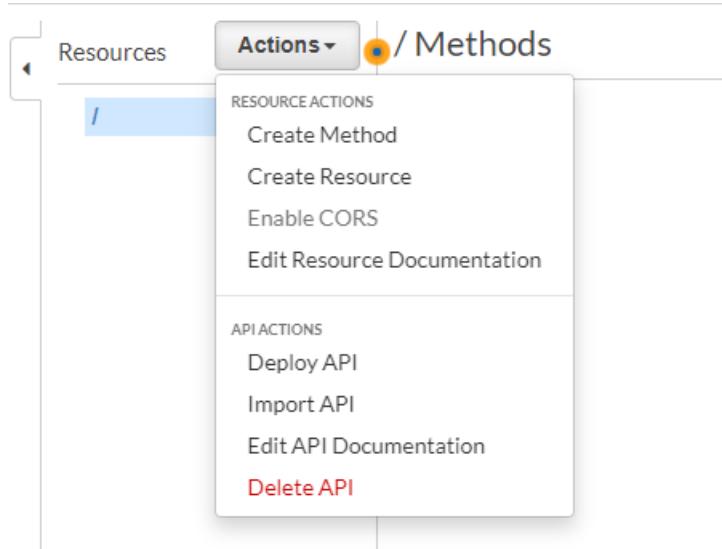
Under settings

- ✓ API Name : M1CallStatisticsDashboard
- ✓ Description : M1CallStatisticsDashboard
- ✓ Endpoint Type : Regional

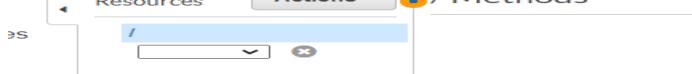
- Click on Create API button
- Once API is created you will navigate to below page



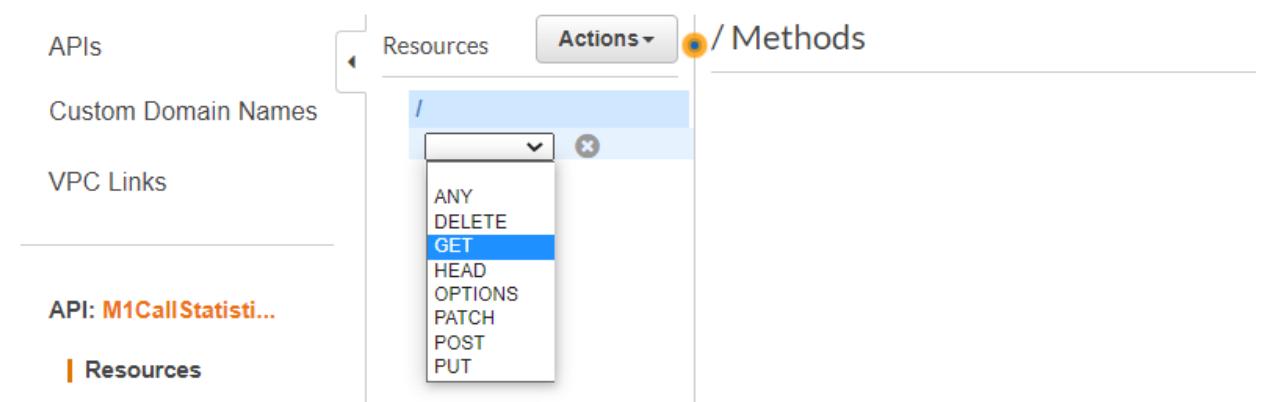
- Now we need to create Method. Click on Actions -> Create Method

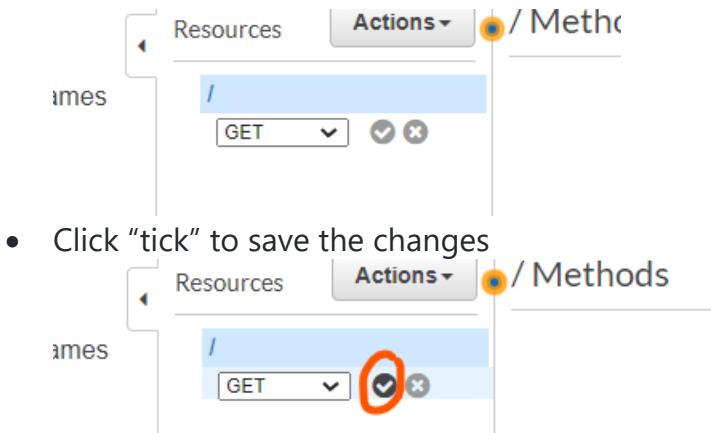


- It will give you method selection like below:



- Select "GET" from the list



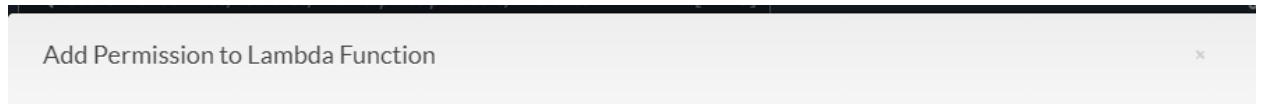


- Click "tick" to save the changes
- Once method is created, you will be able to see below page

The screenshot shows the configuration for a new API method. The 'Integration type' is set to 'Lambda Function'. The 'Lambda Region' is set to 'ap-southeast-1'. The 'Lambda Function' field is empty. A note at the bottom states: 'You do not have any Lambda Functions in ap-southeast-1. Create a Lambda Function in your current account, or provide an Lambda Function ARN for Cross-Account Access.' A 'Save' button is visible at the bottom right.

- Lambda integration with REST API
 - Choose Integration type as "Lambda Function"
 - Use lambda proxy integration – do not uncheck and leave blank
 - Lambda region : please choose Singapore region if you are located in Singapore, otherwise choose the region where lambda function is located
 - Lambda function : **M1CallStatisticsDashboard**
 - Use default timeout : leave ticked as is
 - Click on save button

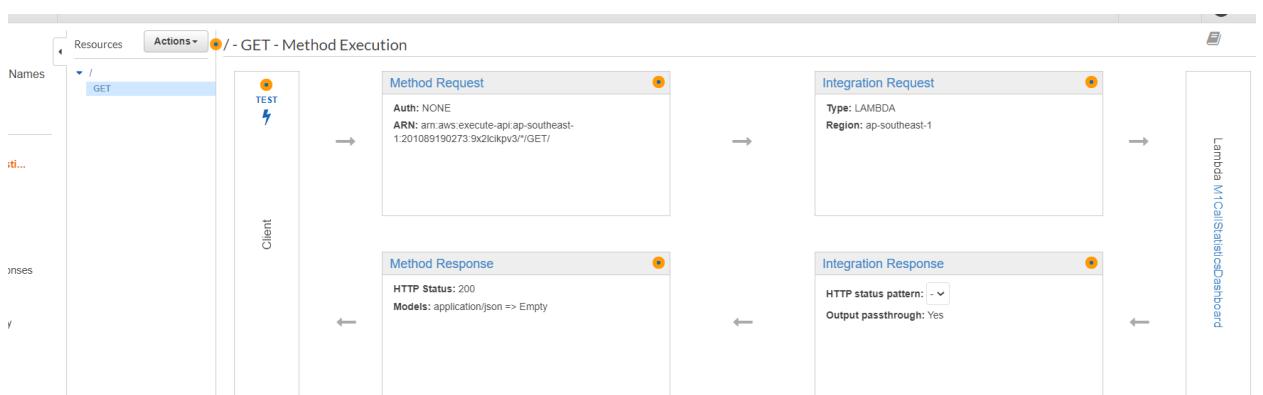
You will get confirmation to provide permissions, please click ok.



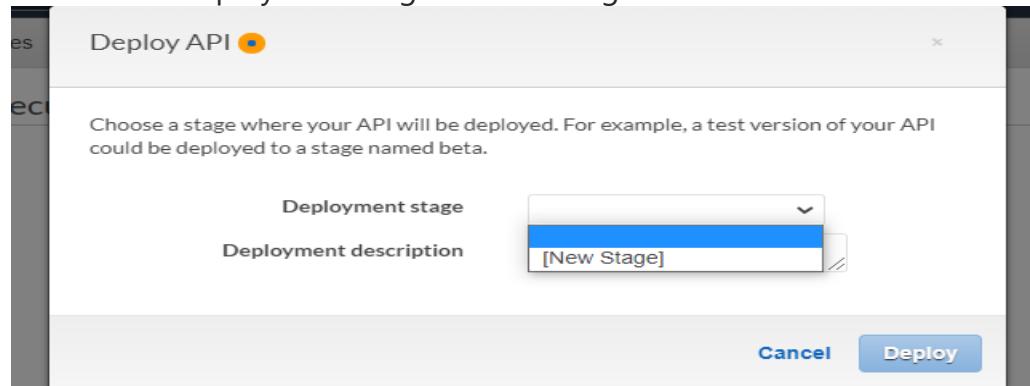
You are about to give API Gateway permission to invoke your Lambda function:
arn:aws:lambda:ap-southeast-1:201089190273:function:M1CallStatisticsDashboard



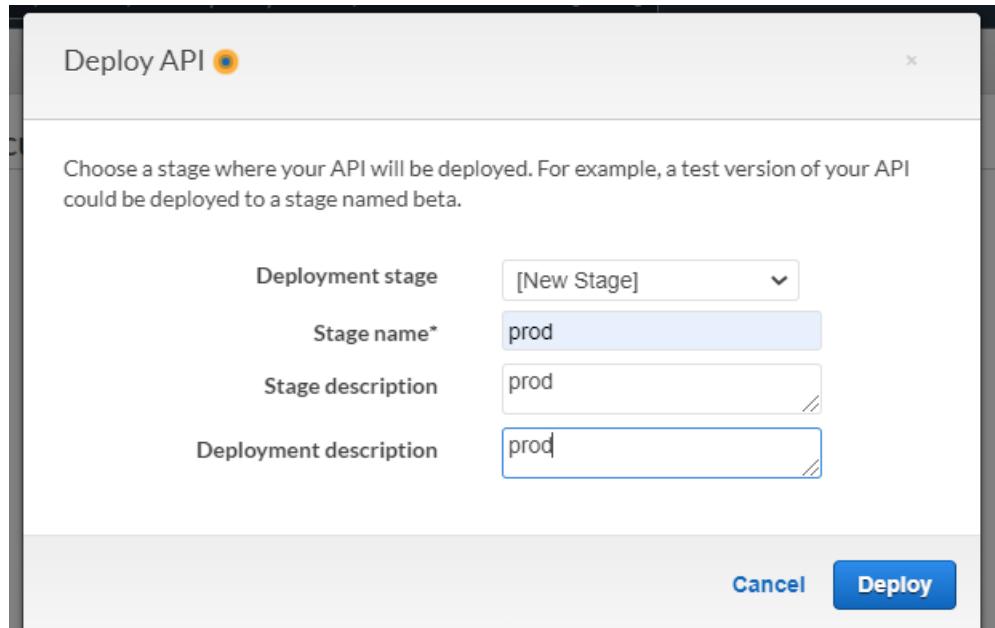
- Once done, you will navigate to below page



- Deploy API Section
 - Click on Actions -> Deploy API
 - Choose the deployment stage as "new stage"



- Provide the staging details like mentioned below:



- Then click on "Deploy" button to deploy the API
- Then you will be navigated to below page where API URL will be created.

- We need to enable CORS
 - Click on "Resources" from left menu
 - Click Actions -> Enable CORS

The screenshot shows the AWS Lambda interface. On the left, there's a sidebar with 'APIs', 'Custom Domain Names', and 'VPC Links'. Below that is a section for the 'API: M1CallStatisticsDashboard' with 'Resources', 'Stages', and 'Authorizers'. The main area shows a 'Resources' list with a single item: '/ (GET)'. An 'Actions' dropdown menu is open over this item, with 'Enable CORS' highlighted by a red oval. Other options in the menu include 'Create Method', 'Create Resource', 'Edit Resource Documentation', 'Deploy API', 'Import API', 'Edit API Documentation', and 'Delete API'. To the right of the menu, there's a status bar with 'a:ap-southeast-1:201089190273:...' and a note 'None'.

- Select "Default 4xx" and "Default 5xx" from the configuration like mentioned below and click on "Enable CORS and replace existing CORS Headers" button

This screenshot shows the 'Enable CORS' configuration dialog for the GET method of the '/ resource'. It includes sections for 'Gateway Responses for M1CallStatisticsDashboard API', 'Methods' (set to GET and OPTIONS), and 'Access-Control-Allow-Methods' (set to GET, OPTIONS). The 'Access-Control-Allow-Headers' field contains 'Content-Type X-Amz-Date Authorization'. A link 'Advanced' is visible at the bottom. At the bottom right is a blue button labeled 'Enable CORS and replace existing CORS headers'.

- Next it will ask you to confirm with the changes

This screenshot shows a confirmation dialog titled 'Confirm method changes'. It lists the changes that will be made: creating an OPTIONS method, adding 200 Method Response with Empty Response Model to OPTIONS method, adding Mock Integration to OPTIONS method, adding 200 Integration Response to OPTIONS method, adding Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Method Response Headers to OPTIONS method, adding Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Integration Response Header Mappings to OPTIONS method, adding Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Response Headers to DEFAULT 4XX Gateway Response, adding Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Response Headers to DEFAULT 5XX Gateway Response, adding Access-Control-Allow-Origin Method Response Header to GET method, and adding Access-Control-Allow-Origin Integration Response Header Mapping to GET method. At the bottom are 'Cancel' and 'Yes, replace existing values' buttons.

- Click on "Yes, replace existing values" button to enable CORS for this API
- Below is the confirmation on Enabling CORS

The screenshot shows the AWS Lambda function configuration interface. The 'Actions' dropdown menu is open, and the 'Enable CORS' step has been selected. A list of successful steps is displayed, including creating an OPTIONS method, adding 200 Method Response with Empty Response Model to OPTIONS method, and adding various Access-Control-Allow-Headers and Access-Control-Allow-Methods. A note at the bottom states: 'Your resource has been configured for CORS. If you see any errors in the resulting output above please check the error message and if necessary attempt to execute the failed step manually via the Method Editor.'

- Re-deploy API
 - Under Resources -> Click Actions -> Deploy API

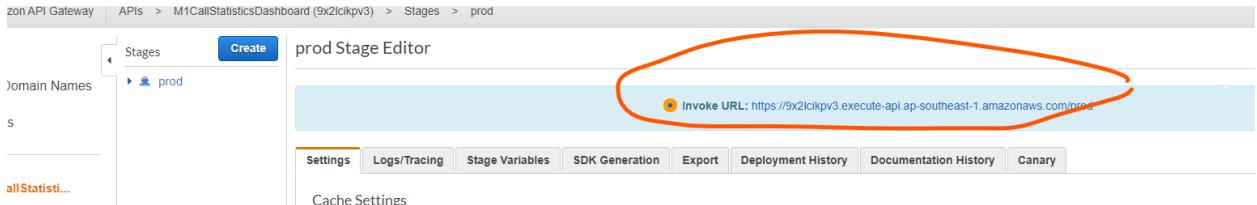
The screenshot shows the AWS API Gateway Resources page for the 'M1CallStatisticsDashboard' API. The 'Actions' dropdown menu is open, and the 'Deploy API' option is highlighted with a red circle. Other options like Create Method, Create Resource, and Edit Resource Documentation are also visible.

- Select the "prod" instance which we have created earlier:

The screenshot shows the 'Deploy API' dialog box. The 'Deployment stage' dropdown is set to 'prod'. The 'Deployment description' field contains '[New Stage]'. At the bottom are 'Cancel' and 'Deploy' buttons.

The screenshot shows the 'Deploy API' dialog box again. The 'Deployment stage' dropdown is set to 'prod'. The 'Deployment description' field contains 'prod - redeploying'. At the bottom are 'Cancel' and 'Deploy' buttons.

- Click on “Deploy” button
- Copy the API gateway REST URL which will be created after deploying API. This we need to configure back in dashboard source code to communicate from public.



- Test the lambda invocation from API Gateway URL
 - Launch above copied URL in browser which should communicate with lambda and should give response like mentioned below

```
{"statusCode":200,"body":"{ \"eventNumber\": \"\" ,\"channel\": \"null\", \"contactID\": \"null\", \"eventInfo\": [{\"ID\":7,\"Channel\": \"VOICE\", \"ContactID\": \"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\"}, {\"CustomerNumber\": \"+6563353529\", \"FlowType\": \"Non-Lex\", \"InitialContactID\": \"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\"}, {\"InteractionDateTime\": \"20210901073120\"}, {\"CustomerJourneyFlow\": \"Non-Lex - Pressed 1\", \"ID\": 3, \"Channel\": \"VOICE\", \"ContactID\": \"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\"}, {\"CustomerNumber\": \"+6563353529\", \"FlowType\": \"Agent Queue\", \"InitialContactID\": \"bifaaa84-6513-4a17-8d1d-c2dd6d725f6017\"}, {\"CustomerNumber\": \"+6563353529\", \"FlowType\": \"Agent Queue\", \"InitialContactID\": \"bifaaa84-6513-4a17-8d1d-c2dd6d725f6017\"}, {\"CustomerNumber\": \"+6563353529\", \"InteractionDateTime\": \"20210901125231\"}, {\"CustomerJourneyFlow\": \"BroadbandEnquiries\"}], {\"ID\":2, \"Channel\": \"VOICE\", \"ContactID\": \"075d6481-b7dd-497d-beda-ceba3a93c490\", \"CustomerNumber\": \"+6563353529\", \"FlowType\": \"Queue\", \"InitialContactID\": \"075d6481-b7dd-497d-beda-ceba3a93c490\", \"InteractionDateTime\": \"20210831125139\", \"CustomerJourneyFlow\": \"BroadbandEnquiries\"}, {\"ID\":4, \"Channel\": \"VOICE\", \"ContactID\": \"a52479ae-427b-49e1-b924-bb041df922b7\", \"CustomerNumber\": \"+6563353529\", \"FlowType\": \"SMS Deflect\", \"InitialContactID\": \"a52479ae-427b-49e1-b924-bb041df922b7\"}, {\"CustomerNumber\": \"+6563353529\", \"FlowType\": \"MobileInquiries - BillingEnquiries\"}, {\"ID\":6, \"Channel\": \"VOICE\", \"ContactID\": \"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\", \"CustomerNumber\": \"+6563353529\", \"FlowType\": \"Non-Lex\", \"InitialContactID\": \"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\"}, {\"CustomerNumber\": \"+6563353529\", \"FlowType\": \"Non-Lex\", \"InitialContactID\": \"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\"}, {\"InteractionDateTime\": \"20210831125335\", \"CustomerJourneyFlow\": \"MobileInquiries - BillingEnquiries\"}, {\"ID\":8, \"Channel\": \"VOICE\", \"ContactID\": \"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\", \"CustomerNumber\": \"+6563353529\", \"FlowType\": \"Non-Lex\", \"InitialContactID\": \"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\"}, {\"InteractionDateTime\": \"20210831124908\", \"CustomerJourneyFlow\": \"Non-Lex - Pressed 1\"}, {\"ID\":15, \"Channel\": \"VOICE\", \"ContactID\": \"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\", \"CustomerNumber\": \"+6563353529\", \"FlowType\": \"Non-Lex\", \"InitialContactID\": \"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\"}, {\"CustomerNumber\": \"+6563353529\", \"FlowType\": \"Non-Lex\", \"InitialContactID\": \"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\"}, {\"InteractionDateTime\": \"20210901072946\", \"CustomerJourneyFlow\": \"Non-Lex - Pressed 1\"}, {\"faceComparePercentage\": \"0.0\"}, \"isBase64Encoded\": false}
```

Deploy dashboard code in S3 and make available to access from public

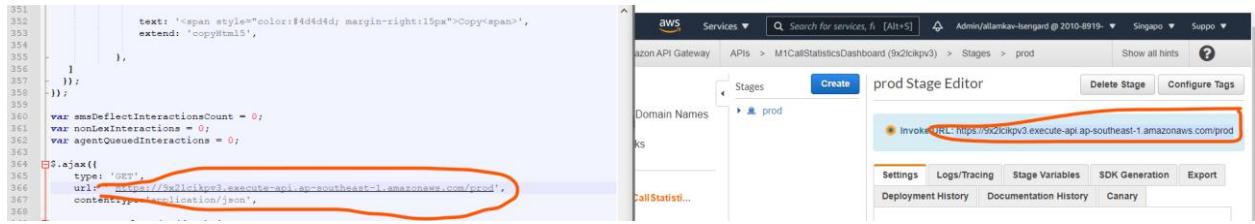
- Navigate to the folder where you have downloaded source files from github
- Open the **CallStatsDashboard** folder from the unzipped folder
- Open **“M1CallStatsDashboard.html”** file in text editor either notepad or notepad++
- Go to the line number 366 and update the API gateway REST URL which you have created earlier and stored in your local

```

360  var smsDeflectInteractionsCount = 0;
361  var nonLexInteractions = 0;
362  var agentQueuedInteractions = 0;
363
364  $.ajax({
365    type: 'GET',
366    url: 'https://ycasu7jsd3.execute-api.ap-southeast-1.amazonaws.com/prod',
367    contentType:'application/json',
368
369    success: function(data) {
370      console.log(data);
371      var jsonData = JSON.parse(data.body);
372      var eventInfo = eval(jsonData.eventInfo);
373      $('#datatable').DataTable().clear();
374

```

- Do not need to change anything except the URL
- Make sure you have the same URL in both API Gateway & "M1CallStatsDashboard.html" file



The screenshot shows the AWS API Gateway interface. In the top navigation bar, 'APIs' is selected. Below it, 'M1CallStatisticsDashboard (9x2iclkpv3)' is chosen, followed by 'Stages' and 'prod'. On the right, there's a 'prod Stage Editor' tab. Under 'Invoke URL', the value 'https://9x2iclkpv3.execute-api.ap-southeast-1.amazonaws.com/prod' is displayed. The bottom of the screen shows tabs for 'Settings', 'Logs/Tracing', 'Stage Variables', 'SDK Generation', and 'Export', along with links for 'Deployment History', 'Documentation History', and 'Canary'.

- Save the file
- Create Static Webpage in S3 and make publicly available to access
 - Navigate to the [S3](#) console
 - Make sure you have disabled block public access under "Block public access setting for this account"

Amazon S3

Buckets
Access Points
Object Lambda Access Points
Batch Operations
Access analyzer for S3

Block Public Access settings for this account

▼ Storage Lens
Dashboards
AWS Organizations settings

Feature spotlight (3)

► AWS Marketplace for S3

Block Public Access settings for this account

Use Amazon S3 Block public access settings to control the settings that allow public access to your data.

Edit

Block all public access

Off

- Block public access to buckets and objects granted through *new* access control lists (ACLs) Off
- Block public access to buckets and objects granted through *any* access control lists (ACLs) Off
- Block public access to buckets and objects granted through *new* public bucket or access point policies Off
- Block public and cross-account access to buckets and objects through *any* public bucket or access point policies Off

- Click on buckets section from the left menu
- Click on "Create Bucket" button

Buckets

Access Points
Object Lambda Access Points
Batch Operations
Access analyzer for S3

Block Public Access settings for this account

Amazon S3

► **Account snapshot**
Storage lens provides visibility into storage usage and activity trends. Learn more

Buckets (20) Info
Buckets are containers for data stored in S3. Learn more

Create bucket

- Under Create bucket configuration, provide as like below
 - Bucket name : **m1connectcallstatisticsdashboard** should be unique in global
 - AWS Region : ap-southeast-1

Create bucket Info

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

- Disable block access by unchecking the tick mark

Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

Block public access to buckets and objects granted through new access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

Block public access to buckets and objects granted through any access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

Block public access to buckets and objects granted through new public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

Block public and cross-account access to buckets and objects through any public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

⚠ Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

- Do not forget to acknowledge change to disable block access
- Leave the other options as is and click on "Create bucket" button.

- Once bucket is created successfully, you will get notification like below

⌚ Successfully created bucket "m1connectcallstatisticsdashboard"

To upload files and folders, or to configure additional bucket settings choose [View details](#).

- Open the bucket by clicking on hyperlink provided to the bucket name

Buckets (21) [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

Search: call X 3 matches

| Name | AWS Region | Access | Creation date |
|---------------------------------|---|-----------------------|---------------------------------------|
| amazonconnectcallstatsdashboard | Asia Pacific (Singapore) ap-southeast-1 | Objects can be public | August 31, 2021, 11:59:28 (UTC+08:00) |

- Under Objects, Click on “Upload” button to upload the static files which will navigate to below page:

Amazon S3 > m1connectcallstatisticsdashboard > Upload

Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

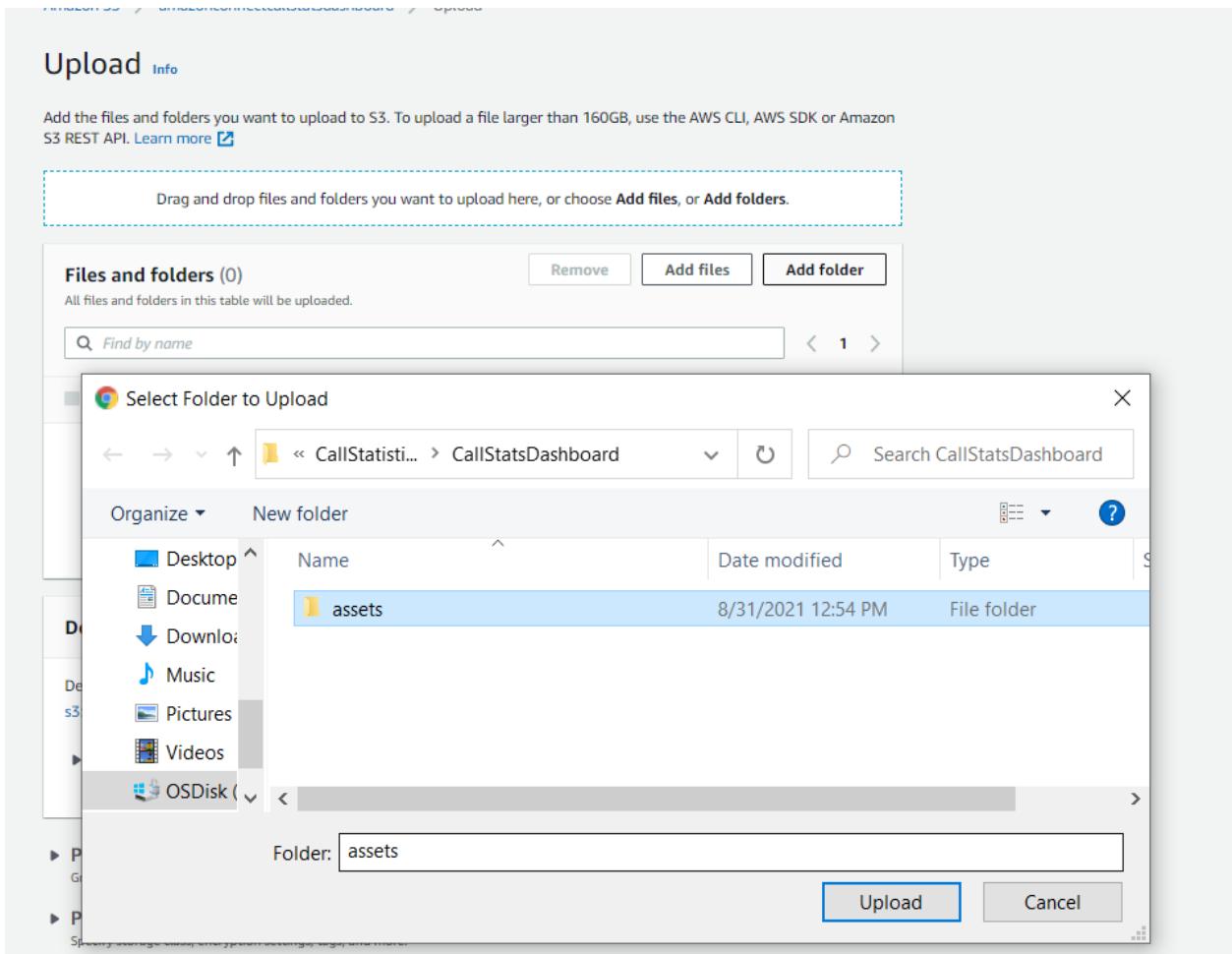
Drag and drop files and folders you want to upload here, or choose **Add files**, or **Add folders**.

| Files and folders (0) | | Remove | Add files | Add folder |
|---|------|------------------------|---------------------------|----------------------------|
| All files and folders in this table will be uploaded. | | | | |
| <input type="text"/> Find by name | | | | |
| <input type="checkbox"/> | Name | Folder | Type | Size |

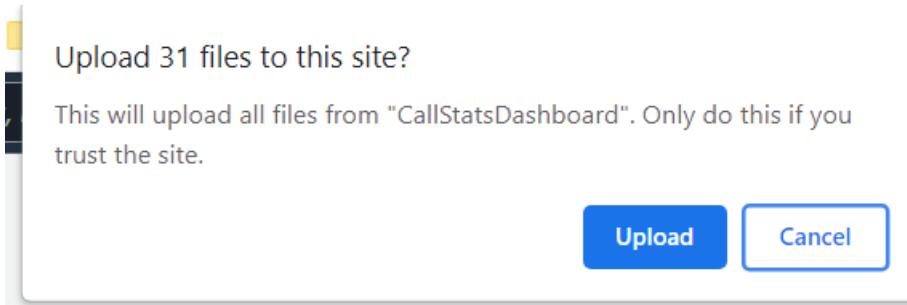
No files or folders

You have not chosen any files or folders to upload.

- Click on “Add folder” button to upload files. Which will open browse window to upload the folder. Navigate to the downloaded local folder (CallStatsDashboard) and choose uploading “assets” folder like mentioned below:



- Once you choose the folder, click on “Upload” button to upload files
- Accept the confirmation dialog box by clicking on “Upload” button



- Then files will be loaded to console.

Find by name

| <input type="checkbox"/> | Name | Folder | Type | Size |
|--------------------------|--|--|-----------------------|----------|
| <input type="checkbox"/> | FontAwesome.otf | CallStatsDashboard/assets/fonts/ | - | 131.6 KB |
| <input type="checkbox"/> | M1CallStatsDashboard.html | CallStatsDashboard/ | text/html | 16.9 KB |
| <input type="checkbox"/> | WhatsApp Image 2021-04-11 at 4.50.51 PM.jpeg | CallStatsDashboard/assets/img/ | image/jpeg | 239.4 KB |
| <input type="checkbox"/> | bootstrap.min.css | CallStatsDashboard/assets/bootstrap/css/ | text/css | 157.6 KB |
| <input type="checkbox"/> | bootstrap.min.js | CallStatsDashboard/assets/bootstrap/js/ | text/javascript | 82.4 KB |
| <input type="checkbox"/> | callStatsDashboard.css | CallStatsDashboard/assets/css/ | text/css | 5.4 KB |
| <input type="checkbox"/> | fa-brands-400.eot | CallStatsDashboard/assets/fonts/ | - | 128.8 KB |
| <input type="checkbox"/> | fa-brands-400.svg | CallStatsDashboard/assets/fonts/ | image/svg+xml | 692.1 KB |
| <input type="checkbox"/> | fa-brands-400.ttf | CallStatsDashboard/assets/fonts/ | - | 128.5 KB |
| <input type="checkbox"/> | fa-brands-400.woff | CallStatsDashboard/assets/fonts/ | application/font-woff | 87.0 KB |

Destination

Destination
<s3://m1connectcallstatisticsdashboard>

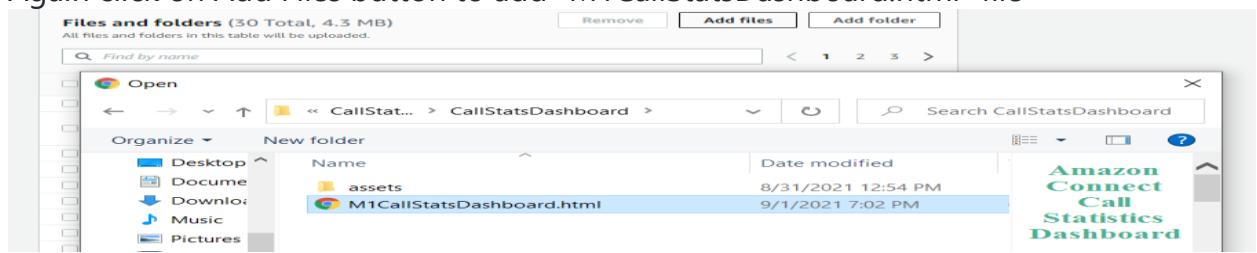
▶ **Destination details**
Bucket settings that impact new objects stored in the specified destination.

▶ **Permissions**
Grant public access and access to other AWS accounts.

▶ **Properties**
Specify storage class, encryption settings, tags, and more.

Cancel **Upload**

- Again click on Add Files button to add "M1CallStatsDashboard.html" file



- Then click on “Upload” button to upload all the files to S3 bucket
- You can see upload status like below

The screenshot shows the AWS S3 "Uploading" progress bar at 100%. Below it, a table lists the uploaded files and folders:

| Name | Folder | Type | Size | Status | Error |
|--|-----------------------|-----------------------|----------|-----------|-------|
| FontAwesome.otf | assets/fonts/ | - | 131.6 KB | Succeeded | - |
| MTCallStatsDashboard.html | - | text/html | 16.9 KB | Succeeded | - |
| WhatsApp Image 2021-04-11 at 4.50.51 PM.jpeg | assets/img/ | image/jpeg | 239.4 KB | Succeeded | - |
| bootstrap.min.css | assets/bootstrap/css/ | text/css | 157.6 KB | Succeeded | - |
| bootstrap.min.js | assets/bootstrap/js/ | text/javascript | 82.4 KB | Succeeded | - |
| callStatsDashboard.css | assets/css/ | text/css | 5.4 KB | Succeeded | - |
| fa-brands-400.eot | assets/fonts/ | - | 128.8 KB | Succeeded | - |
| fa-brands-400.svg | assets/fonts/ | image/svg+xml | 692.1 KB | Succeeded | - |
| fa-brands-400.ttf | assets/fonts/ | - | 128.5 KB | Succeeded | - |
| fa-brands-400.woff | assets/fonts/ | application/font-woff | 87.0 KB | Succeeded | - |

- Once successfully uploaded files, you will get success notification

The screenshot shows a success message: "Upload succeeded" and a link "View details below". Below it, a section titled "Upload: status" contains the message: "The information below will no longer be available after you navigate away from this page."

- Enable static website option
 - Navigate to properties

The screenshot shows the "Bucket overview" section of the AWS S3 console. It includes fields for "AWS Region" (Asia Pacific (Singapore) ap-southeast-1) and "Amazon Resource Name (ARN)" (arn:aws:s3:::m1connectcallstatisticsdashboard). At the bottom, there is a "Static website hosting" link.

- Scroll all the way down and then you will find “Static web hosting” option.

Static website hosting
Use this bucket to host a website or redirect requests. [Learn more](#) [Edit](#)

Static website hosting
Disabled

- Click edit button to enable webhosting
- Then follow below screenshot to enable the setting & provide index document as "M1CallStatsDashboard.html"

Static website hosting
Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting

Disable

Enable

Hosting type

Host a static website
Use the bucket endpoint as the web address. [Learn more](#)

Redirect requests for an object
Redirect requests to another bucket or domain. [Learn more](#)

For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#)

Index document
Specify the home or default page of the website.

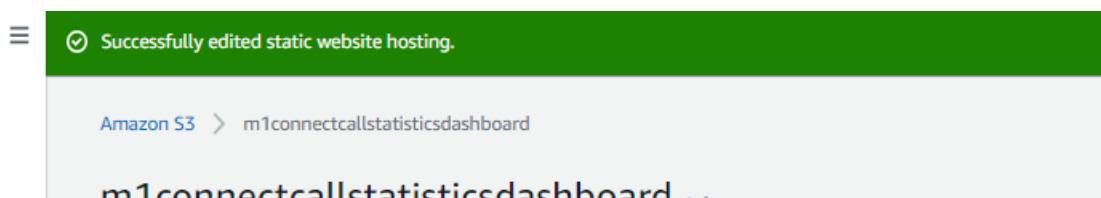
M1CallStatsDashboard.html

Error document - optional
This is returned when an error occurs.

error.html

Redirection rules - optional
Redirection rules, written in JSON, automatically redirect webpage requests for specific content. [Learn more](#)

- Then click on save changes.



☰ Successfully edited static website hosting.

Amazon S3 > m1connectcallstatisticsdashboard

m1connectcallstatisticsdashboard ...

- Make objects public:
 - Select “objects” under objects tab and Click actions -> Make public

Amazon S3 > m1connectcallstatisticsdashboard

m1connectcallstatisticsdashboard [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions.

| <input checked="" type="checkbox"/> | Name | Type |
|-------------------------------------|---------------------------|--------|
| <input checked="" type="checkbox"/> | assets/ | Folder |
| <input checked="" type="checkbox"/> | M1CallStatsDashboard.html | html |

[Actions](#) [Create folder](#) [Upload](#)

Download as
Calculate total size
Copy
Move
Initiate restore
Query with S3 Select
2:59 (UTC+08:00)
Edit actions
Rename object
Edit storage class
Edit server-side encryption
Edit metadata
Edit tags
Make public

- Once successfully enabled public access, you will get below notification

Successfully edited public access
View details below.

- Navigate to Properties tab and scroll all the way down till “Static website hosting” section. Copy the bucket website endpoint URL in browser

Static website hosting
Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting
Enabled

Hosting type
Bucket hosting

Bucket website endpoint
What if you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)

<http://m1connectcallstatisticsdashboard.s3-website-ap-southeast-1.amazonaws.com>

- Open the URL in browser. You should be able to see the page loaded.
"HURRAY!!!!.... but please wait, we need to access this page from cloud front which we are going to do that exercise in next section"

Creating CloudFront to access the S3 objects from public

- Navigate to [Cloudfront](#) in console
- Click on "Create Distribution"

- Choose the options as mentioned below:
 - Origin domain : select s3 bucket name from the dropdown as "m1connectcallstatisticsdashboard" or the one which you have created.

- Name field automatically will populate as "m1connectcallstatisticsdashboard.s3.ap-southeast-1.amazonaws.com"
- Do not need to change any other settings and scroll all the way down and click "Create Distribution" button to create the distribution

- Then distribution will start creating and you can see the status as "Deploying".

| Distributions (4) <small>Info</small> | | | | | | | | | <input type="button" value="C"/> | <input type="button" value="Enable"/> | <input type="button" value="Disable"/> | <input type="button" value="Delete"/> | <input type="button" value="Create distribution"/> |
|---------------------------------------|----------------|-------------|-------------------------------|-----------------|-----------------------|---|---|--|----------------------------------|---------------------------------------|--|---------------------------------------|--|
| | ID | Description | Domain name | Alternate do... | Origins | Status | Last modified | | < 1 > | ⚙️ | | | |
| | E14499TJ9H3TRA | - | d1p3jdr6c58knh.cloudfront.net | - | m1connectcallstatisti | <input checked="" type="button"/> Enabled | <input checked="" type="button"/> Deploying | | < 1 > | ⚙️ | | | |

- Wait for some time until it is completely deployed.
- Once it is successfully deployed, get domain name which we can use to access our dashboard.

| Distributions (4) <small>Info</small> | | | | | | | | | <input type="button" value="C"/> | <input type="button" value="Enable"/> | <input type="button" value="Disable"/> | <input type="button" value="Delete"/> | <input type="button" value="Create distribution"/> |
|---------------------------------------|----------------|-------------|-------------------------------|-----------------|-----------------------|---|---------------|--|----------------------------------|---------------------------------------|--|---------------------------------------|--|
| | ID | Description | Domain name | Alternate do... | Origins | Status | Last modified | | < 1 > | ⚙️ | | | |
| | E14499TJ9H3TRA | - | d1p3jdr6c58knh.cloudfront.net | - | m1connectcallstatisti | <input checked="" type="button"/> Enabled | | | < 1 > | ⚙️ | | | |

Launch dashboard

Launch the CloudFront URL followed by dashboard html file (<https://d1p3jdr6c58knh.cloudfront.net/M1CallStatsDashboard.html>) and then you will be able to see the dashboard loaded with data.

| ID ↑ | Channel | ContactID | CustomerNumber | FlowType | InitialContactID | InteractionDateTime | CustomerJourneyFlow |
|------|---------|--------------------------------------|----------------|-------------|--------------------------------------|---------------------|-------------------------------------|
| 1 | VOICE | 3b6a9bc7-bf1b-4cb1-be27-6a20461b4573 | +6563353529 | Non-Lex | 3b6a9bc7-bf1b-4cb1-be27-6a20461b4573 | 20210831124908 | Non-Lex -> Pressed 1 |
| 2 | VOICE | 075d6481-b7dd-497d-beda-ceb3a633c490 | +6563353529 | Queue | 075d6481-b7dd-497d-beda-ceb3a633c490 | 20210831125139 | BroadbandEnquiries |
| 3 | VOICE | b1faaa84-6513-4a17-8d1d-c26d667255f0 | +6563353529 | Agent Queue | b1faaa84-6513-4a17-8d1d-c26d667255f0 | 20210831125231 | BroadbandEnquiries |
| 4 | VOICE | a52479ae-427b-49e1-b924-b0b41df922b7 | +6563353529 | SMS-Deflect | a52479ae-427b-49e1-b924-b0b41df922b7 | 20210831125335 | MobileEnquiries -> BillingEnquiries |
| 5 | VOICE | 3b6a9bc7-bf1b-4cb1-be27-6a20461b4573 | +6563353529 | Non-Lex | 3b6a9bc7-bf1b-4cb1-be27-6a20461b4573 | 20210901072946 | Non-Lex -> Pressed 1 |
| 6 | VOICE | 3b6a9bc7-bf1b-4cb1-be27-6a20461b4573 | +6563353529 | Non-Lex | 3b6a9bc7-bf1b-4cb1-be27-6a20461b4573 | 20210901073026 | Non-Lex -> Pressed 1 |
| 7 | VOICE | 3b6a9bc7-bf1b-4cb1-be27-6a20461b4573 | +6563353529 | Non-Lex | 3b6a9bc7-bf1b-4cb1-be27-6a20461b4573 | 20210901073120 | Non-Lex -> Pressed 1 |

Testing by associating call flow to the number and make test calls

SMS Deflect flow:

- After greeting say "Mobile" for mobile enquiries
- Then in the next menu, say "Billing" for billing enquiries
- This will take us to the SMS deflect flow

Agent Queue Flow:

- After greeting say "Broadband" for broadband related queries
- This will take us to Agent queue flow

DTMF Flow:

- After greeting, say "TV packages" or do not say anything for some time. Here lex should not recognize intent to shift to DTMF flow
- After some time, then you will hear that DTMF flow is activated.
- Press1 to know for television packages.
- This will take us to DTMF flow

