

Amazon Connect Call statistics Solution

Contents

Amazon Connect Call statistics Solution	1
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 functions.....	9
Upload “AddCustomerContactFlowJourneyData“ lambda :	9
Configuration section:.....	13
Change the function handler	14
Test the lambda function section:	16
Upload “SendSMSFromContactFlowToCaller“ lambda :.....	17
Create SMS Project in Pinpoint Service:.....	20
Configuration section:.....	22
Change the function handler	24
Test the lambda function section:	26
Configure lambda to amazon connect instance.....	27
Upload “ConnectCallStatisticsDashboard“ Lambda : This lambda will be used to fetch the data from DynamoDB for displaying in the UI dashboard.	28
Configuration section:.....	31
Change Function handler section	33
Test the lambda function section:.....	34
Steps to import contact flow to Amazon Connect instance.....	35
Creating REST API in APIGateway to access lambda from public	37
Deploy dashboard code in S3 and make available to access from public	45
Creating CloudFront to access the S3 objects from public	51
Launch dashboard.....	53
Testing by associating call flow to the number and make test calls	53
SMS Deflect flow:	53
Agent Queue Flow:	53

Installation Instructions

Pre-requisites

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

https://github.com/ajvenkatesh/AmazonConnect_IVRCallStatistics

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"/> ▾
---------------------------------	---

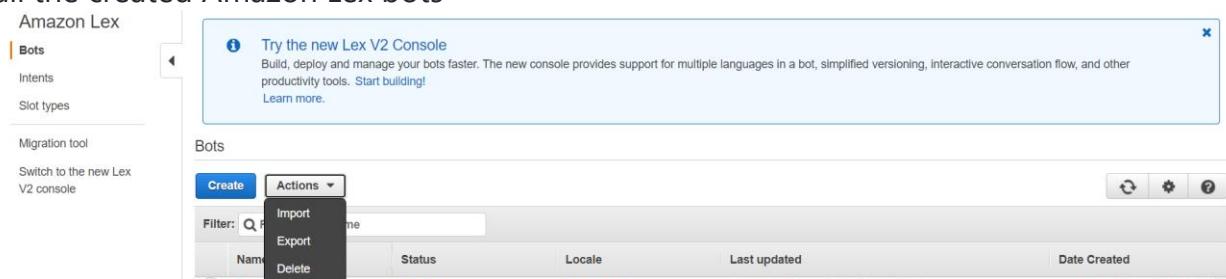
1 to 255 characters and case sensitive.

- ✓ 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. 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



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**

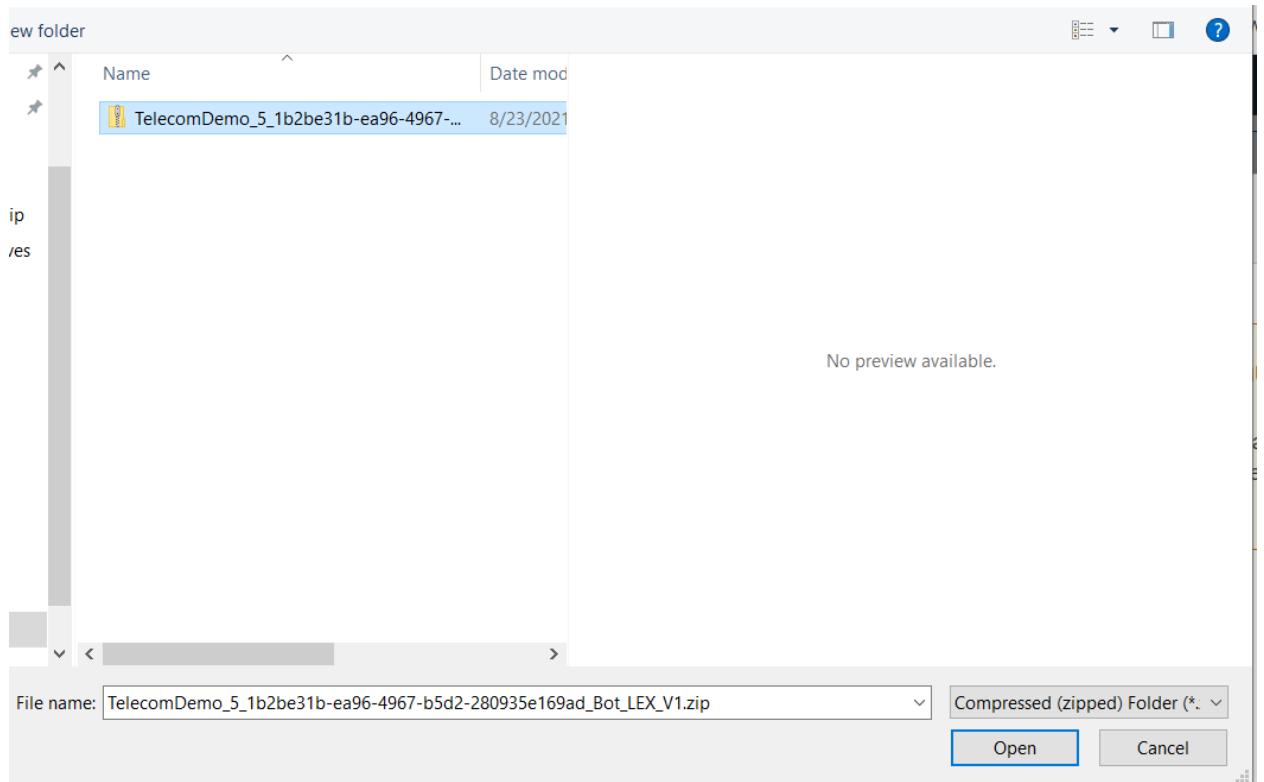
Upload a .zip file

▶ Tags

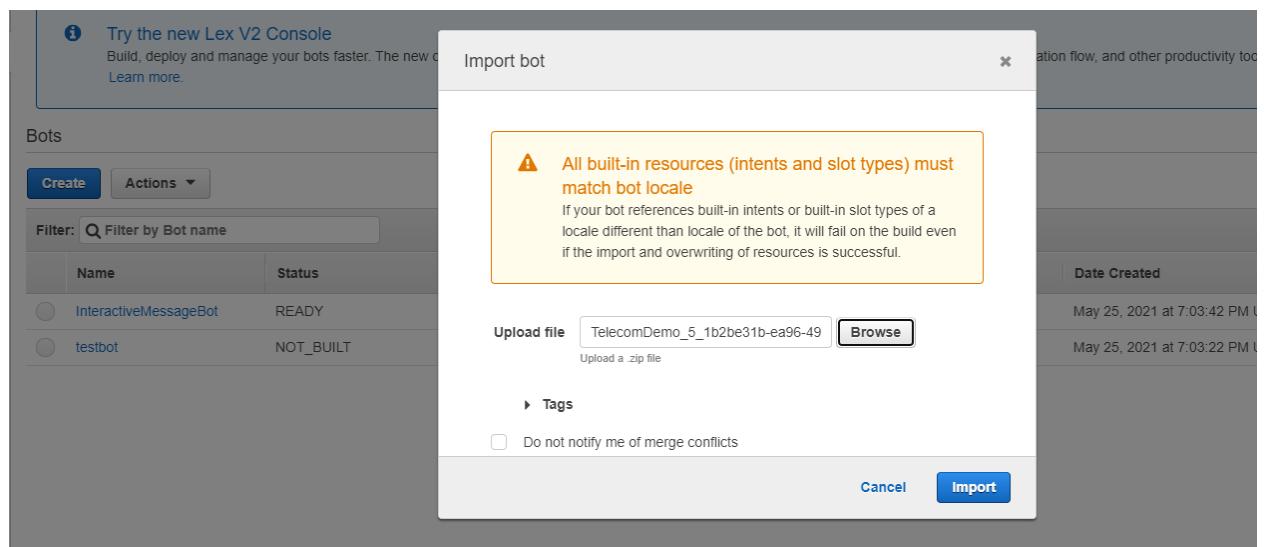
Do not notify me of merge conflicts

Cancel **Import**

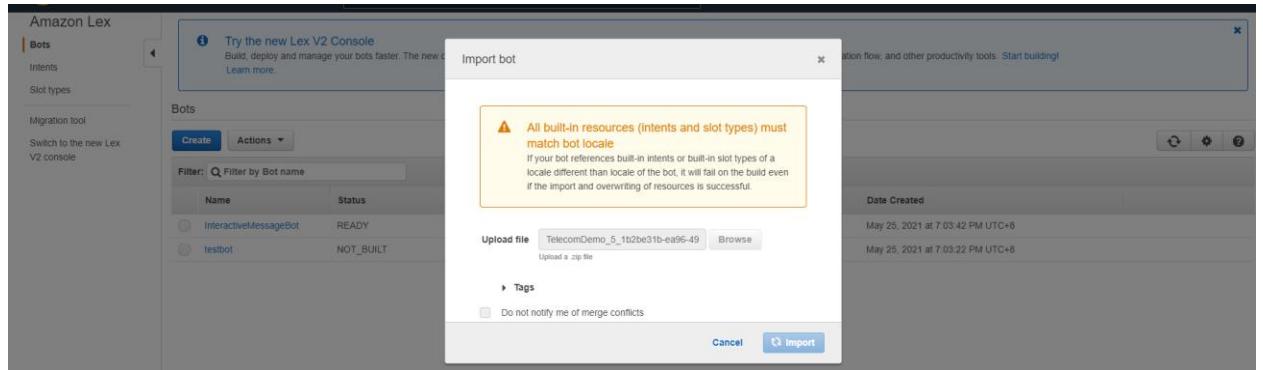
Click on browse to upload the Bot zip file.



Click on Open.



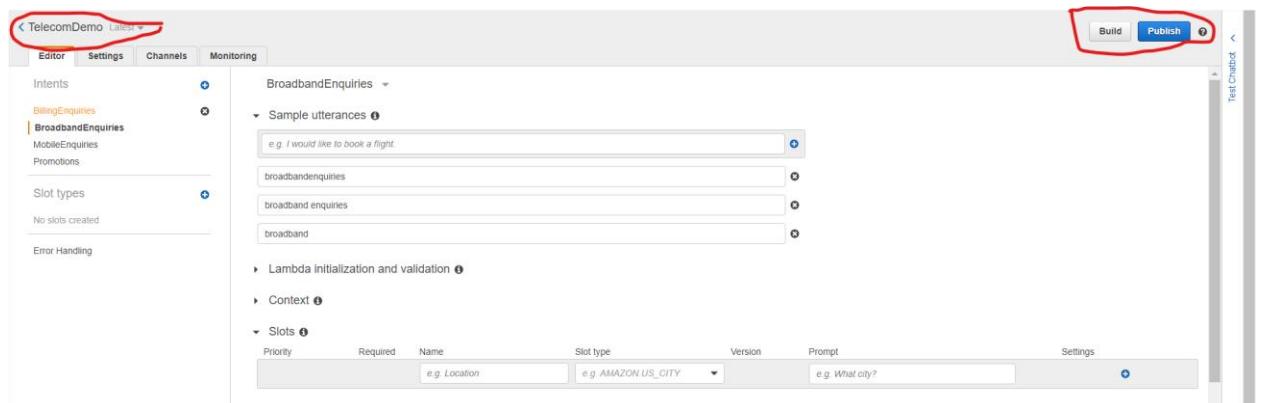
Click on Import button to load the Bot data to the console.



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



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



6. If it asks you for alias put TelecomDemo as alias and click Publish.
7. 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 message 'We've r') and 'We ar' (with a message 'To mail'). Below these is a section titled 'Contact flow' with a note about signing keys and an 'Add key' button.

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 note about granting permission to interact with them, and a 'Create a new Lex bot' link. The 'Region' dropdown is set to 'Asia Pacific: Singapore'. The '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 to create dynamic paths in IVR based on user responses.

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

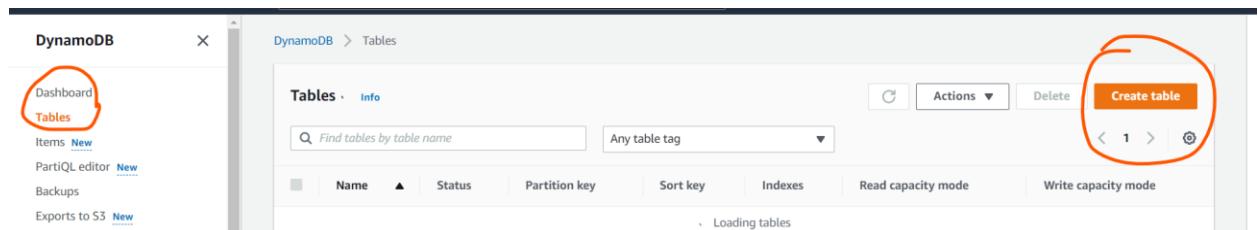
[+ Add Lex Bot](#)

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 functions.

Upload “AddCustomerContactFlowJourneyData“ 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 AWS Lambda Functions page. The left sidebar has links for Dashboard, Applications, and Functions. The main area shows a table with 77 entries, with a search bar at the top. The top right corner features a prominent orange '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:

The screenshot shows the 'Create function' wizard. It consists of five steps:

- Step 1: Choose options** (radio button selected): "Author from scratch" (selected), "Use a blueprint", "Container image", "Browse serverless app repository".
- Step 2: Basic information**: Function name: myFunctionName, Runtime: Node.js 14.x.
- Step 3: Permissions**: 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.
- Step 4: Advanced settings**: Options include "Change default execution role".
- Step 5: Review**: Buttons: "Cancel" and "Create function" (orange button).

- ✓ 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.

<input checked="" type="radio"/> Author from scratch Start with a simple Hello World example.	<input type="radio"/> Use a blueprint Build a Lambda application from sample code and configuration presets for common use cases.	<input type="radio"/> Container image Select a container image to deploy for your function.	<input type="radio"/> 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.
AddCustomerContactFlowJourneyData

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)

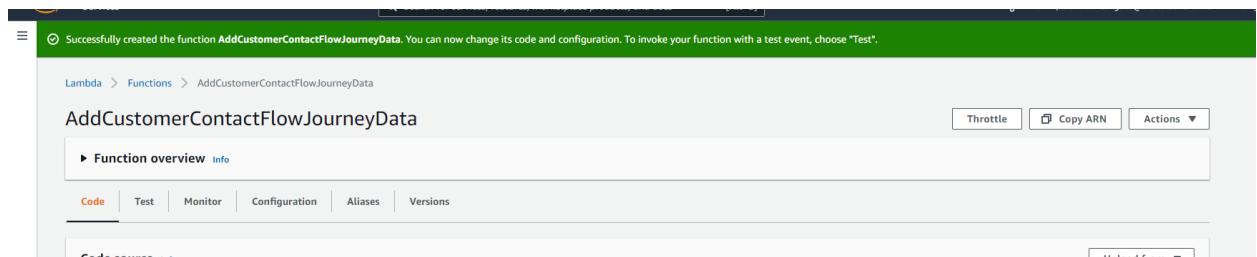
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▶ 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 “**AddCustomerContactFlowJourneyData**” lambda function like mentioned below

Dashboard Applications **Functions** Additional resources Related AWS resources

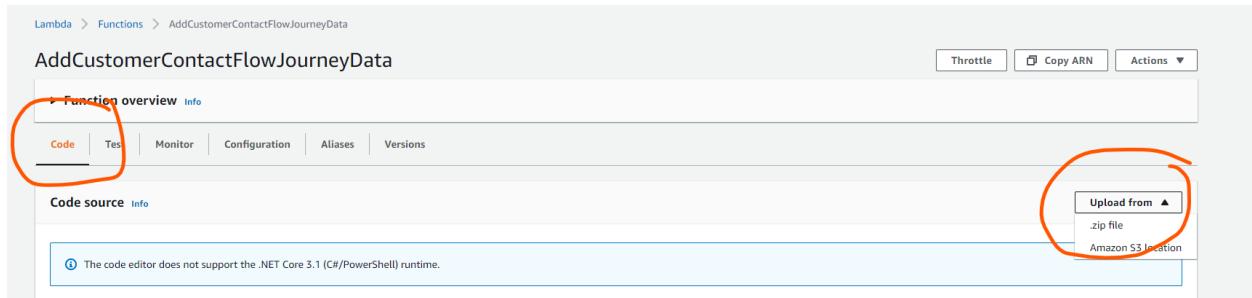
Functions (78) Last fetched 1 minute ago Actions **Create function**

Filter by tags and attributes or search by keyword 1 match

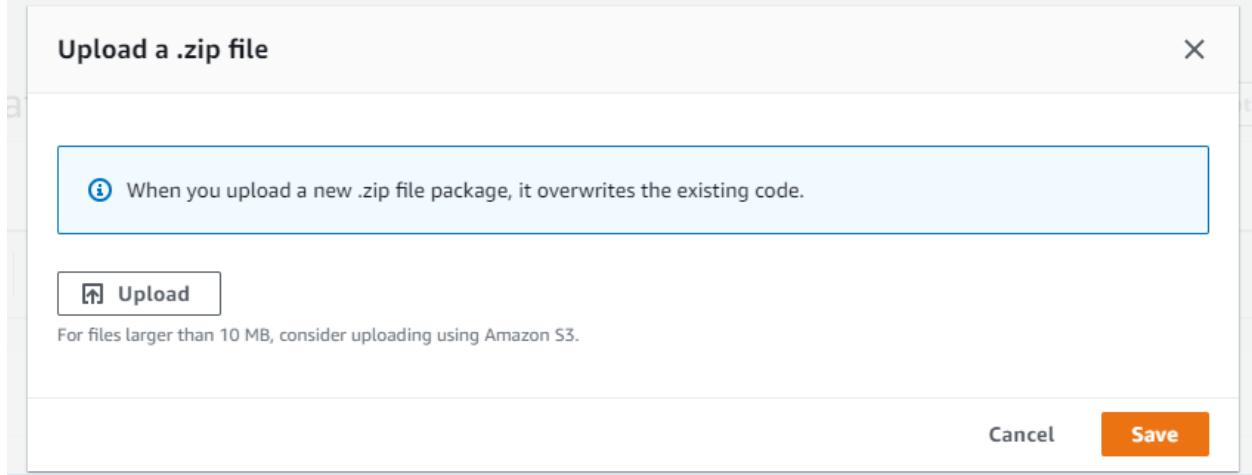
Function name: **AddCustomerContactFlowJourneyData**

Function name	Description	Package type	Runtime	Code size	Last modified
AddCustomerContactFlowJourneyData	-	Zip	.NET Core 3.1 (C#/PowerShell)	532.6 kB	9 minutes ago

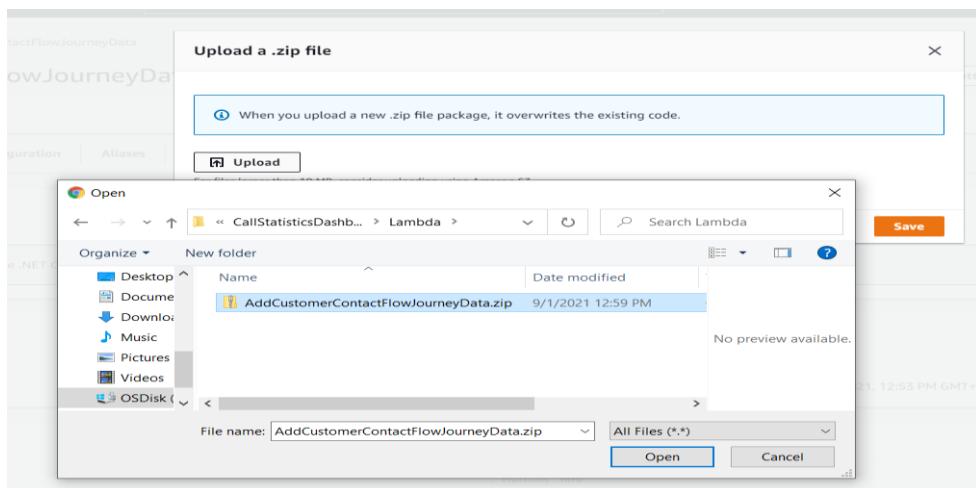
- Click on “**AddCustomerContactFlowJourneyData**” function which will navigate to the page where we can upload code.
- Click “.Zip file” from uploadfile option under the Code tab like mentioned below:



- 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) . 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:

The screenshot shows the AWS Lambda Function Overview page for a function named "AddCustomerContactFlowJourneyData". The "Configuration" tab is selected. On the left, there's a sidebar with options: General configuration, Triggers, Permissions, Destinations, Environment variables (which is highlighted with a red circle), and Tags. The main area shows "Environment variables (0)". It has two columns: "Key" and "Value". A large red circle highlights the entire "Environment variables" section. To the right of the table, there's a "No environment variables" message and an "Edit" button, which is also circled.

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

The screenshot shows the "Edit environment variables" page. At the top, there's a breadcrumb navigation: Lambda > Functions > AddCustomerContactFlowJourneyData > Edit environment variables. The main title is "Edit environment variables". Below it, there's a section titled "Environment variables" with a sub-instruction: "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](#)". A message below states: "There are no environment variables on this function." There is a prominent "Add environment variable" button, which is circled. 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

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	AKIASSUOZRGAS5CORTSFQ	Remove
IAMSecret	hvl+tdYe6hLC6LcTpQlPN8BEi7XfmJ+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.

Lambda > Functions > AddCustomerContactFlowJourneyData

AddCustomerContactFlowJourneyData

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
DBTableName	CustomerContactFlowJourney
IAMSecret	hvl+tdYe6hLC6LcTpQlPN8BEi7XfmJ+9IRHGozM
IAMUserID	AKIASSUOZRGAS5CORTSFQ
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. At the top, there's a header with 'AddCustomerContactFlowJourneyData' and three buttons: 'Throttle', 'Copy ARN', and 'Actions'. Below the header, there's a navigation bar with tabs: 'Function overview' (highlighted with a red circle), 'Code' (highlighted with a red circle), 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. Under the 'Code' tab, there's a 'Code source' section with a note: 'The code editor does not support the .NET Core 3.1 (C#/PowerShell) runtime.' and a 'Upload from' button. In the 'Runtime settings' section, there's a 'Runtime' dropdown set to '.NET Core 3.1 (C#/PowerShell)' and a 'Handler' dropdown set to 'LambdaTest::LambdaHandler::handleRequest'. An 'Edit' button is highlighted with a red circle.

- o Change the Handler from

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

“`DynamoDBConnector::DynamoDBConnector.Function::AddCustomerContactFlowJourn
eyData`”

Edit runtime settings

The screenshot shows the 'Edit runtime settings' dialog. It has a title 'Runtime settings' with an 'Info' link. Below it is a 'Runtime' section with a dropdown menu set to '.NET Core 3.1 (C#/PowerShell)'. Underneath is a 'Handler' section with a dropdown menu set to '`DynamoDBConnector::DynamoDBConnector.Function::AddCustomerContactFlowJourn
eyData`'. At the bottom right are 'Cancel' and 'Save' buttons, with 'Save' being highlighted with a red border.

Then click on save button to save the handler settings.

The screenshot shows the AWS Lambda function configuration interface again. At the top, there's a green banner with the message 'Your changes have been saved.' Below the banner, there's a header with 'Lambda > Functions > AddCustomerContactFlowJourneyData' and three buttons: 'Throttle', 'Copy ARN', and 'Actions'. The main content area shows the function name 'AddCustomerContactFlowJourneyData'.

Test the lambda function section:

- Naviagate 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's a section titled 'Test event'. It says '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.' There are two radio buttons: 'New event' (selected) and 'Saved event'. A dropdown menu labeled 'Template' has 'hello-world' selected (also circled in orange). At the bottom right of the 'Test event' section are buttons for 'Format', 'Save changes', and 'Test'.

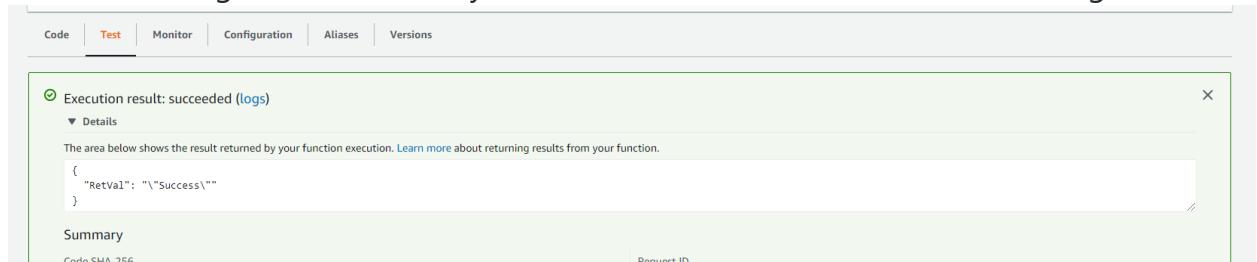
- Copy below json in the json field to test the function

```
{
  "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"
  }
}
```

This screenshot is identical to the one above, showing the AWS Lambda Function Overview page with the 'Test' tab selected. The 'Template' dropdown still shows 'hello-world'. The 'Test event' section now contains the JSON template code from the previous step. At the bottom right of the 'Test event' section are buttons for 'Format', 'Save changes', and 'Test'.

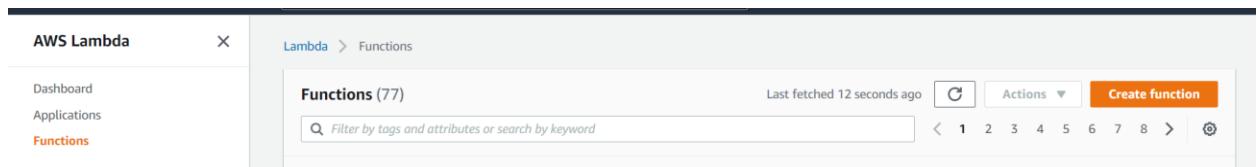
- Click on “Test” button to test the function

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

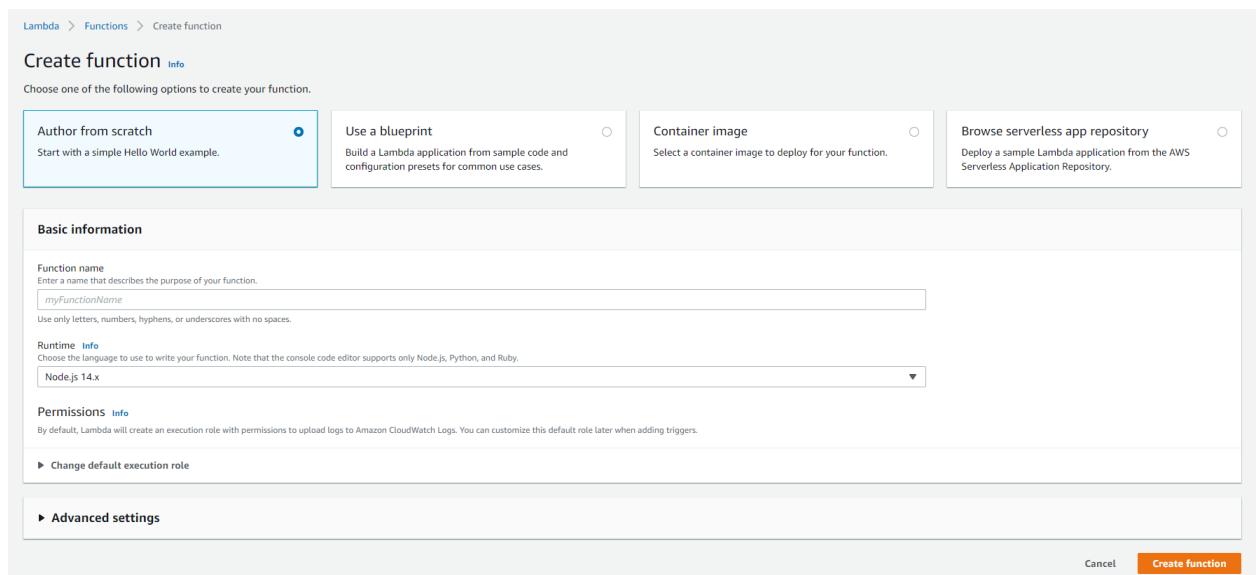


Upload “SendSMSFromContactFlowToCaller“ lambda :

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



- 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:



- ✓ 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.
`SendSMSFromContactFlowToCaller`
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use for writing 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.

Advanced settings

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

Lambda > Functions > SendSMSFromContactFlowToCaller

SendSMSFromContactFlowToCaller

Function overview Info

 <code>SendSMSFromContactFlowToCaller</code>	 Layers (0)	Description -
Add trigger		Add destination
Last modified 6 seconds ago		
Function ARN <code>arn:aws:lambda:ap-southeast-1:201089190273:function:SendSMSFromContactFlowToCaller</code>		

- ✓ 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

Functions (78)						
Function name:		Description	Package type	Runtime	Code size	Last modified
<input type="checkbox"/>	Function name	-	Zip	.NET Core 3.1 (C#/PowerShell)	532.6 kB	34 seconds ago
<input type="checkbox"/>	SendSMSFromContactFlowToCaller	-				

- 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:

Lambda > Functions > SendSMSFromContactFlowToCaller

SendSMSFromContactFlowToCaller

Function overview

Add trigger

Add destination

Layers (0)

Description

Last modified 1 minute ago

Function ARN arn:aws:lambda:ap-southeast-1:201089190273:function:SendSMSFromContactFlowToCaller

Code Test Monitor Configuration Aliases Versions

Code source

The code editor does not support the .NET Core 3.1 PowerShell runtime.

Upload from .zip file

Amazon S3 location

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

Upload a .zip file

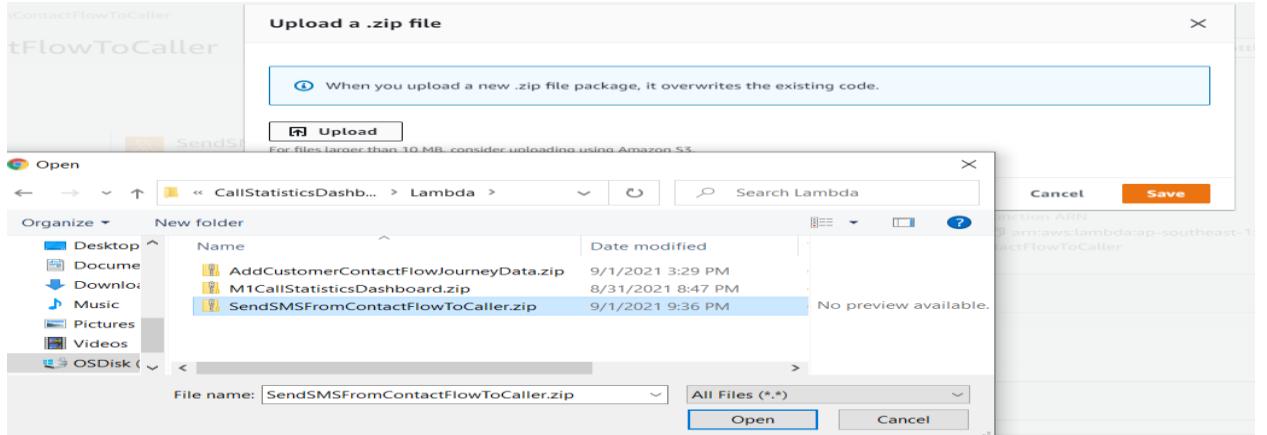
When you upload a new .zip file package, it overwrites the existing code.

Upload

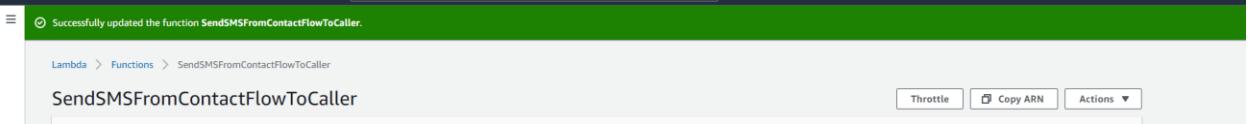
For files larger than 10 MB, consider uploading using Amazon S3.

Cancel Save

- 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:

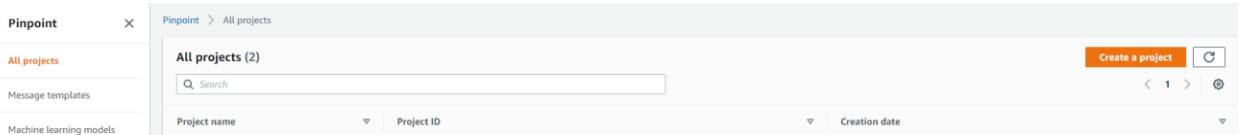


- ✓ After successfully loaded the zip file, we will get success notification. Please wait for some time to upload the function as it will take time.

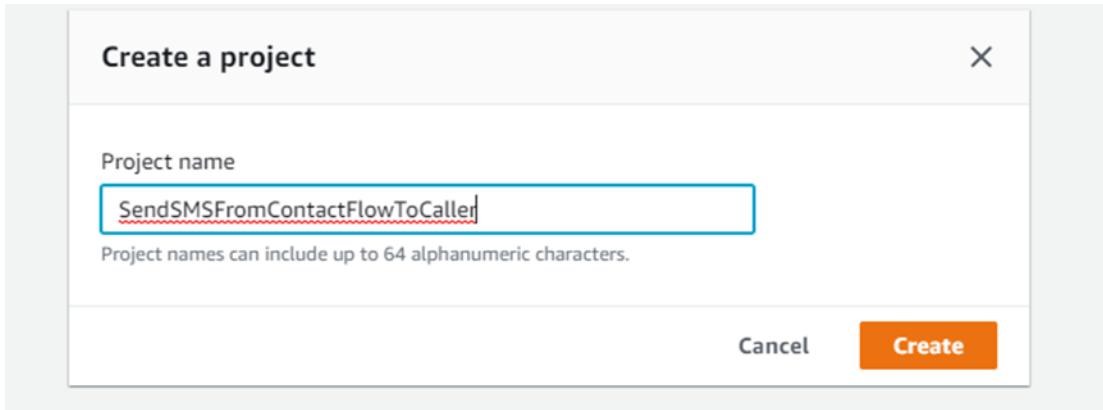


Create SMS Project in Pinpoint Service:

- ✓ 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



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



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

You've created SendSMSFromContactFlowToCaller. Now add features to your project.

Pinpoint > All projects > SendSMSFromContactFlowT... > Configure features

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

Project features

Messaging channels and response metrics

Email

SMS

Push notifications

Configure

Configure

Configure

- ✓ 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](#)

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

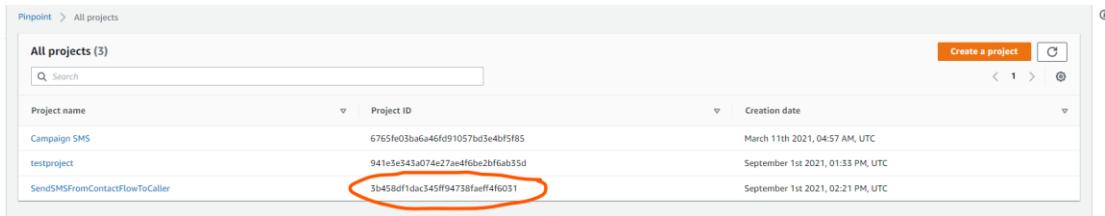
(Optional) Account sender ID
 The identity that appears on recipients' devices when they receive this message. Support varies by country or region. [Info](#)

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 'Function overview' section is highlighted with a red circle. The 'Configuration' tab in the navigation bar is also highlighted with a red circle. Below the configuration tab, there is a section titled 'Environment variables (0)' which contains a table with columns 'Key' and 'Value'. A red circle highlights the 'Edit' button next to this section.

- 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 breadcrumb navigation shows: Lambda > Functions > AddCustomerContactFlowJourneyData > Edit environment variables. The main section is titled 'Environment variables' and contains a message: '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.' Below this, it says 'There are no environment variables on this function.' A red circle highlights the 'Add environment variable' button. At the bottom right, there are 'Cancel' and 'Save' buttons, with 'Save' being highlighted by a red circle.

- 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::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::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 banner at the top says 'Your changes have been saved.' Below it, the navigation path is 'Lambda > Functions > SendSMSFromContactFlowToCaller'.

Test the lambda function section:

- Naviagate 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 that, a navigation bar has tabs for 'Code', 'Test' (which is highlighted with an orange circle), 'Monitor', 'Configuration', 'Aliases', and 'Versions'. Under the 'Test' tab, there's a section titled 'Test event'. It says '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.' There are two radio buttons: 'New event' (selected) and 'Saved event'. A dropdown menu labeled 'Template' shows 'hello-world' selected (also highlighted with an orange circle). At the bottom right of the 'Test event' section are buttons for 'Format', 'Save changes', and 'Test'.

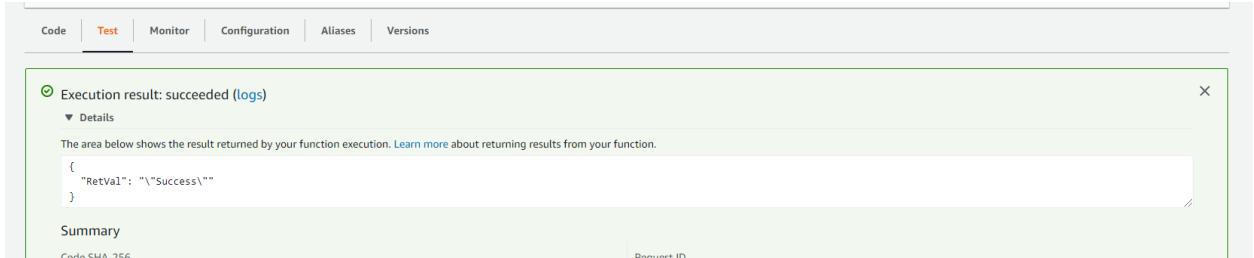
- Copy below json in the json field to test the function. (change your mobile number in two places where highlighted)

```
{
  "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 Amazon : 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, the 'Template' dropdown shows 'hello-world'. Below it, there's a text input field labeled 'MyEventName' containing the JSON code from the previous step. The code is identical to the one above, with the mobile number '+6563353529' highlighted in yellow. At the bottom right of the 'Test event' section are buttons for 'Format', 'Save changes', and 'Test'.

- Click on "Test" button to test the function

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

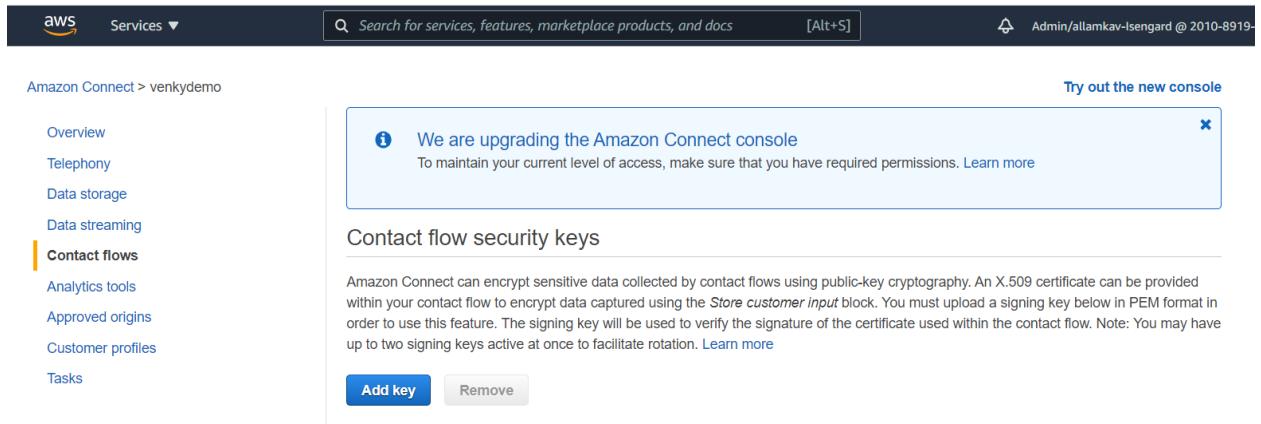


The screenshot shows the AWS Lambda Test interface. At the top, there are tabs for 'Code', 'Test' (which is selected), 'Monitor', 'Configuration', 'Aliases', and 'Versions'. Below the tabs, a green box indicates a successful execution with the message 'Execution result: succeeded (logs)'. A 'Details' section shows the returned JSON object: { "RetVal": "\"Success\"" }. There is also a 'Summary' section at the bottom.

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. At the top, there is a navigation bar with the AWS logo, 'Services ▾', a search bar ('Search for services, features, marketplace products, and docs'), and user information ('Admin/allamkav-lsengard @ 2010-8919-'). Below the navigation bar, the URL 'Amazon Connect > venkydemo' is shown. On the left, a sidebar menu includes 'Overview', 'Telephony', 'Data storage', 'Data streaming', 'Contact flows' (which is highlighted with an orange border), 'Analytics tools', 'Approved origins', 'Customer profiles', and 'Tasks'. 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, the heading 'Contact flow security keys' is visible, along with a note about encrypting sensitive data using X.509 certificates and a 'Add key' 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" dropdown and a "Select" button. Below it is a "Lambda Functions" section. A single function is listed: "AddCustomerContactFlowJourneyData". The ARN is "arn:aws:lambda:ap-southeast-1:201089190273:function:AddCustomerContactFlowJourneyData". To the right of the function name are three icons: a copy icon, a "Remove" link, and another copy icon.

- ✓ 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" dropdown and a "Select" button. Below it is a "Lambda Functions" section. Four functions are listed:

- "AddCustomerContactFlowJourneyData" (ARN: arn:aws:lambda:ap-southeast-1:201089190273:function:AddCustomerContactFlowJourneyData)
- "SendSMSToMyMobileNumber" (ARN: arn:aws:lambda:ap-southeast-1:201089190273:function:SendSMSToMyMobileNumber)
- "serverlessrepo-AmazonConnectSalesforce-sfInvokeAPI-dnuVvW8keJcw" (ARN: arn:aws:lambda:ap-southeast-1:201089190273:function:serverlessrepo-AmazonConnectSalesforce-sfInvokeAPI-dnuVvW8keJcw)
- "SendSMSFromContactFlowToCaller" (ARN: arn:aws:lambda:ap-southeast-1:201089190273:function:SendSMSFromContactFlowToCaller)

Each function entry includes an "Edit" icon, a "Remove" link, and another "Edit" icon.

Upload “ConnectCallStatisticsDashboard” 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 > Functions". Below it is a "Functions (77)" section. The ARN "Last fetched 12 seconds ago" is shown. There is a "Create function" button. Below the main list, there is a pagination bar with numbers 1 through 8 and a "Filter by tags and attributes or search by keyword" input field.

- ✓ 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 for 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 “**ConnectCallStatisticsDashboard**”.
- ✓ 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 for 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

The screenshot shows the AWS Lambda console with a green header bar indicating "Successfully created the function M1CallStatisticsDashboard. You can now change its code and configuration. To invoke your function with a test event, choose "Test". Below this, the function details are shown:

- M1CallStatisticsDashboard**
- Function overview** (Info)
- Code** (selected), **Test**, **Monitor**, **Configuration**, **Aliases**, **Versions**
- Code source** (Info)
- Upload from** (button)
- A note: "The code editor does not support the .NET Core 3.1 (C#/PowerShell) runtime."

- ✓ 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 “**ConnectCallStatisticsDashboard**” lambda function like mentioned below

The screenshot shows the AWS Lambda Functions list page with the following details:

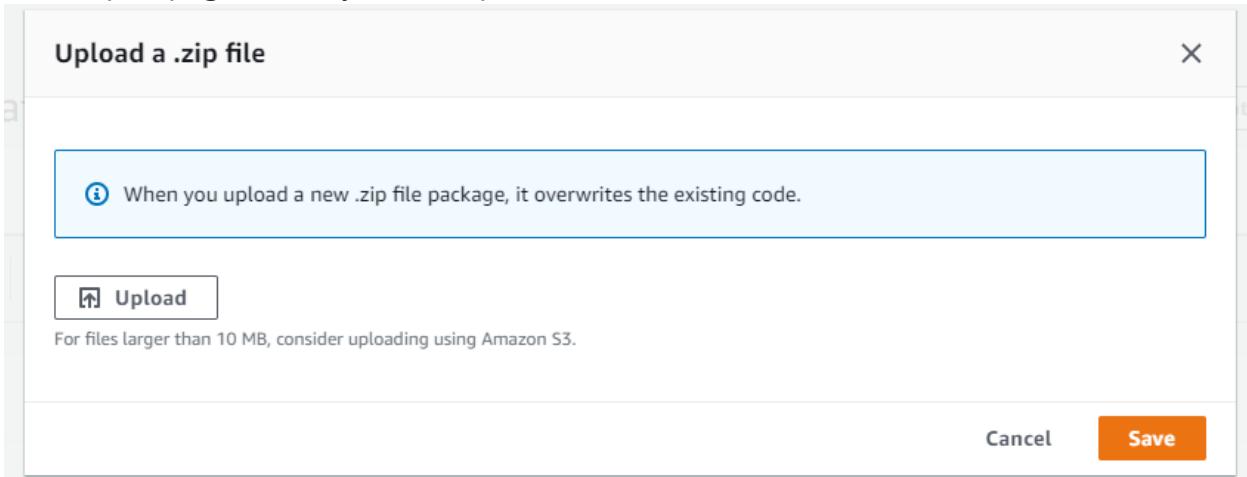
- Functions (78)**
- Filter by tags and attributes or search by keyword**: M1CallStatisticsDashboard (1 match)
- Last fetched**: 5 seconds ago
- Create function** button
- Actions** dropdown
- Function name**: M1CallStatisticsDashboard
- Description**: -
- Package type**: Zip
- Runtime**: .NET Core 3.1 (C#/PowerShell)
- Code size**: 532.6 kB
- Last modified**: 1 minute ago

- Click on “**ConnectCallStatisticsDashboard**” 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 ConnectCallStatisticsDashboard:

- ConnectCallStatisticsDashboard**
- Function overview** (Info)
- Code** (selected), **Test**, **Monitor**, **Configuration**, **Aliases**, **Versions**
- Code source** (Info)
- Upload from** dropdown menu:
 - zip file (highlighted with a red circle)
 - Amazon S3 location
- A note: "The code editor does not support the .NET Core 3.1 (C#/PowerShell) runtime."

- 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:

M1 > ConnectCallStatisticsDashboard-Github > ConnectCallStatisticsDashboard-main > Lambda >					
New folder					
	Name	Date modified	Type	Size	
sDashboard	AddCustomerContactFlowJourneyData.zip	9/2/2021 11:19 AM	WinRAR ZIP archive	2,780 KB	
	ConnectCallStatisticsDashboard.zip	2/9/2022 6:39 PM	WinRAR ZIP archive	2,562 KB	

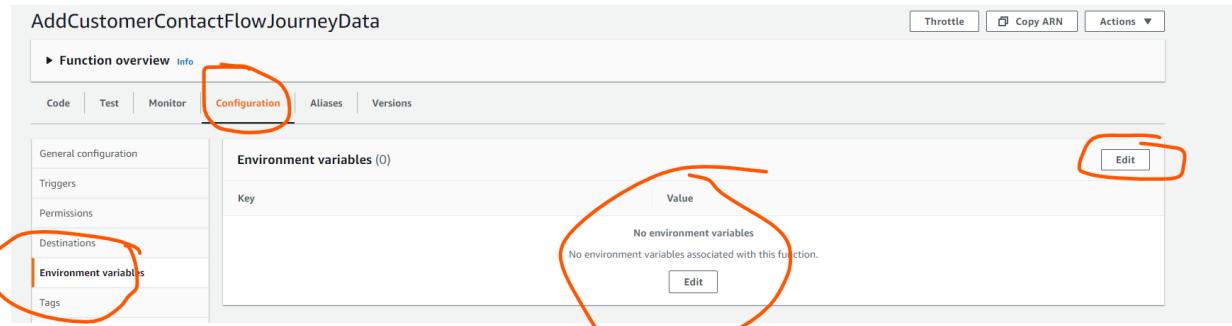
- ✓ After successfully loaded the zip file, we will get success notification. Please wait for some time to upload the function as it will take time.

✓



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	AKIASSUOZRGA5CORTSFQ	Remove
IAMSecret	hvl+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	hvl+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

The screenshot shows the AWS Lambda function configuration page for 'M1CallStatisticsDashboard'. The 'Code' tab is selected. The 'Runtime settings' section is highlighted with a red circle, showing the runtime as '.NET Core 3.1 (C#/PowerShell)' and the handler as 'LambdaTest::LambdaTest.LambdaHandler::handleRequest'. An 'Edit' button is also circled.

- Change the Handler from
“LambdaTest::LambdaTest.LambdaHandler::handleRequest” to
“ConnectCallStatisticsDashboard::ConnectCallStatisticsDashboard.Function::GetCallStatsData”

Edit runtime settings

The screenshot shows the 'Edit runtime settings' dialog. It shows the 'Runtime' section with '.NET Core 3.1 (C#/PowerShell)' selected and the 'Handler' section with 'ConnectCallStatisticsDashboard::ConnectCallStatisticsDashboard.Function::GetCallStat' entered.

Then click on save button to save the handler settings.

The screenshot shows the AWS Lambda function configuration page. A green banner at the top says 'Successfully updated the function ConnectCallStatisticsDashboard.'. Below it, the function details are shown: Lambda > Functions > ConnectCallStatisticsDashboard. The 'Actions' menu is visible at the bottom right.

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 'Code', 'Test' (which is highlighted with a red circle), 'Monitor', 'Configuration', 'Aliases', and 'Versions'. Below the tabs, there's a section titled 'Test event'. It says '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.' There are two radio button options: 'New event' (selected) and 'Saved event'. A dropdown menu labeled 'Template' has 'hello-world' selected (highlighted with a red circle). At the bottom right are 'Format', 'Save changes', and 'Test' buttons.

- No need to change the json

The screenshot shows the 'Test event' configuration page. The 'Template' dropdown is set to 'hello-world'. The 'Name' field contains 'MyEventName'. The 'Event Content' text area shows the following JSON:

```

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

```

- 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 'Execution result: succeeded (logs)' modal. It displays the following log output:

```

{
  "statusCode": 200,
  "body": "{\"eventName\":\"OnGetOrCreateStsData\",\"channel\":null,\"contactID\":null,\"eventInfo\":{},\"ID\":7,\"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-c266d67255f0\", \"CustomerNumber\":\"+6563353529\", \"FlowType\":\"Agent Queue\", \"InitialContactID\":\"b1faaa84-6513-4a17-8d1d-c266d67255f0\", \"InteractionDateTime\":\"20210831125231\", \"CustomerJourneyFlow\":\"BroadbandEnquiries\", \"ID\":2, \"Channel\":\"VOICE\", \"ContactID\":\"075d6481-b7dd-497d-beda-ceb3a633c490\", \"CustomerNumber\":\"+6563353529\", \"FlowType\":\"Queue\", \"InitialContactID\":\"075d6481-b7dd-497d-beda-ceb3a633c490\", \"InteractionDateTime\":\"20210831125139\", \"CustomerJourneyFlow\":\"BroadbandEnquiries\", \"ID\":4, \"Channel\":\"VOICE\", \"ContactID\":\"a52479ae-427b-49e1-b924-b0b41df92b7\", \"CustomerNumber\":\"+6563353529\", \"FlowType\":\"SMS-Deflect\", \"InitialContactID\":\"a52479ae-427b-49e1-b924-"
}

```

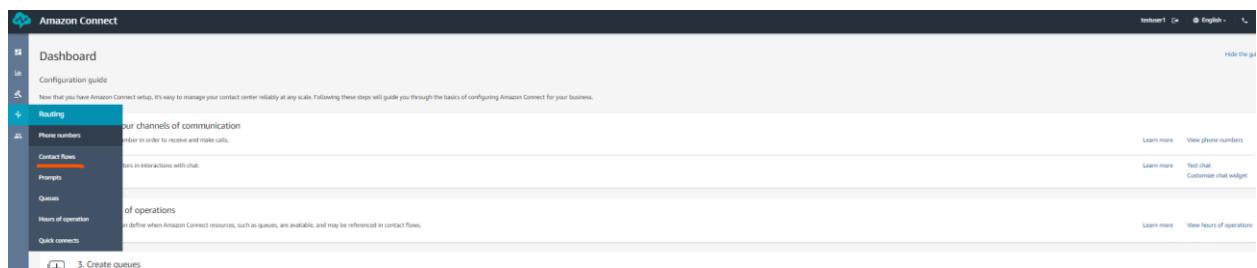
Steps to import contact flow to Amazon Connect instance.

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



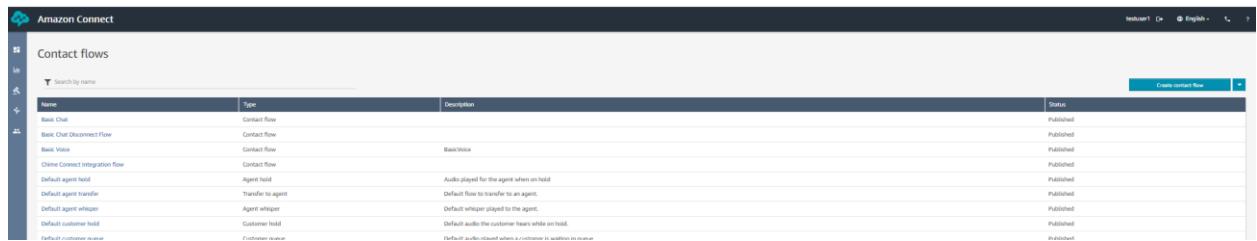
Add an instance	Remove	Instance Alias	Access URL	Channels	Create Date	Status
<input type="checkbox"/> venkydemo			https://venkydemo.awsapps.com/con...	Inbound, outbound telephony	2/23/2021	Active

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.



The screenshot shows the Amazon Connect Routing interface. On the left, there's a sidebar with links: Dashboard, Configuration guide, Routing (which is selected and highlighted with an orange box), Phone numbers, Contact flows (selected), Prompts, Queues, Hours of operation, and Quick consults. The main content area has sections for 'Your channels of communication' (Phone numbers, Contact flows, Prompts, Queues, Hours of operation, Quick consults) and a 'Create queues' button at the bottom.

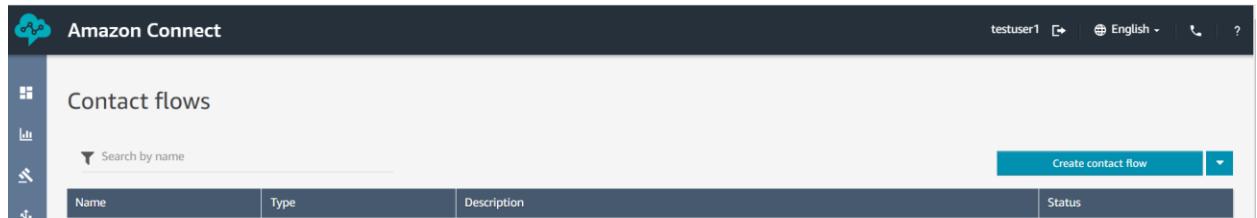
You will be able to see the existing default contact flows which comes along with connect instance.



The screenshot shows the Contact flows page. The sidebar has 'Contact flows' selected. The main area displays a table of existing contact flows:

Name	Type	Description	Status
Basic Chat	Contact flow		Published
Basic Chat Disconnect Flow	Contact flow		Published
Basic Voice	Contact flow	Basic voice	Published
Chime 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 intent	Customer intent	Default audio played when a customer is waiting on queue.	Published

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



The screenshot shows the Contact flows page again. The 'Create contact flow' button in the top right corner is highlighted with a blue box.

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

« ConnectCallStatisticsDashboard-main > ContactFlow

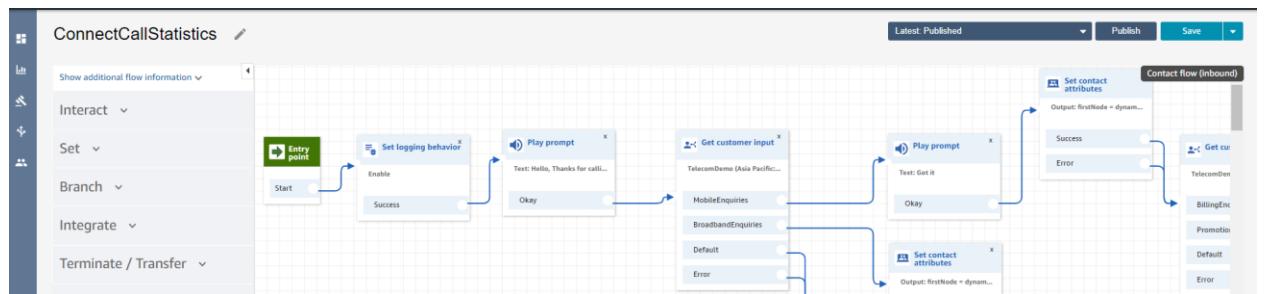
Name	Date modified	Type	Size
M1ConnectCallStatistics	9/1/2021 7:23 PM	File	18 KB

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



7. Click on Import flow and choose the "ConnectCallStatistics" 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.

8.

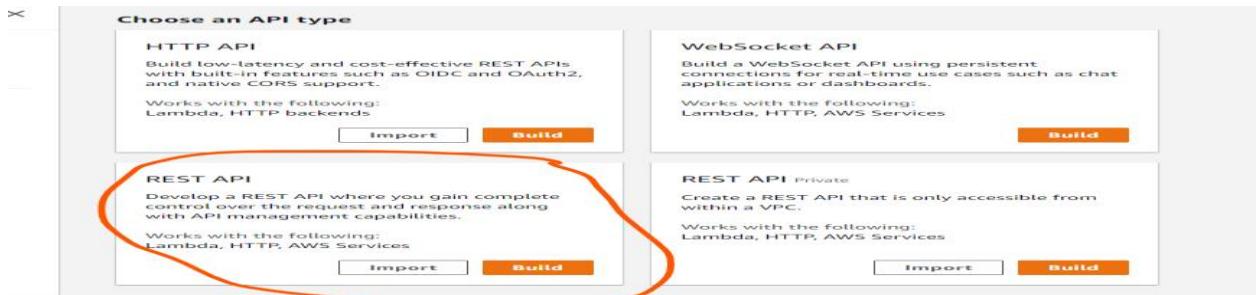


Creating REST API in APIGateway to access lambda from public

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

API Gateway		APIs (1)						
Actions		Create API						
APIs								
Custom domain names								
VPC links								

- Choose API type as REST API



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

Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

REST WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

New API Clone from existing API Import from Swagger or Open API 3 Example API

Settings

Choose a friendly name and description for your API.

API name*	ConnectCallStatisticsDashboard
Description	ConnectCallStatisticsDashboard
Endpoint Type	Regional

* Required

Choose REST API

Under settings

- ✓ API Name : [ConnectCallStatisticsDashboard](#)
- ✓ Description : [ConnectCallStatisticsDashboard](#)
- ✓ Endpoint Type : Regional

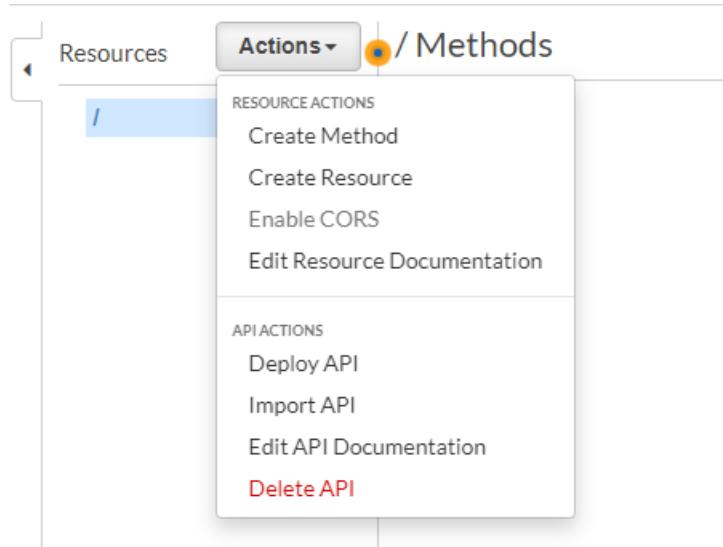
- Click on Create API button
- Once API is created you will navigate to below page

Amazon API Gateway APIs > M1CallStatisticsDashboard (9x2lckpv3) > Resources > / (t1jouxjhk)

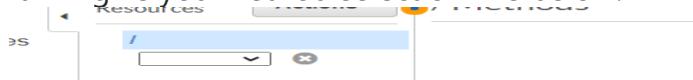
Actions > / Methods

No methods defined for the resource.

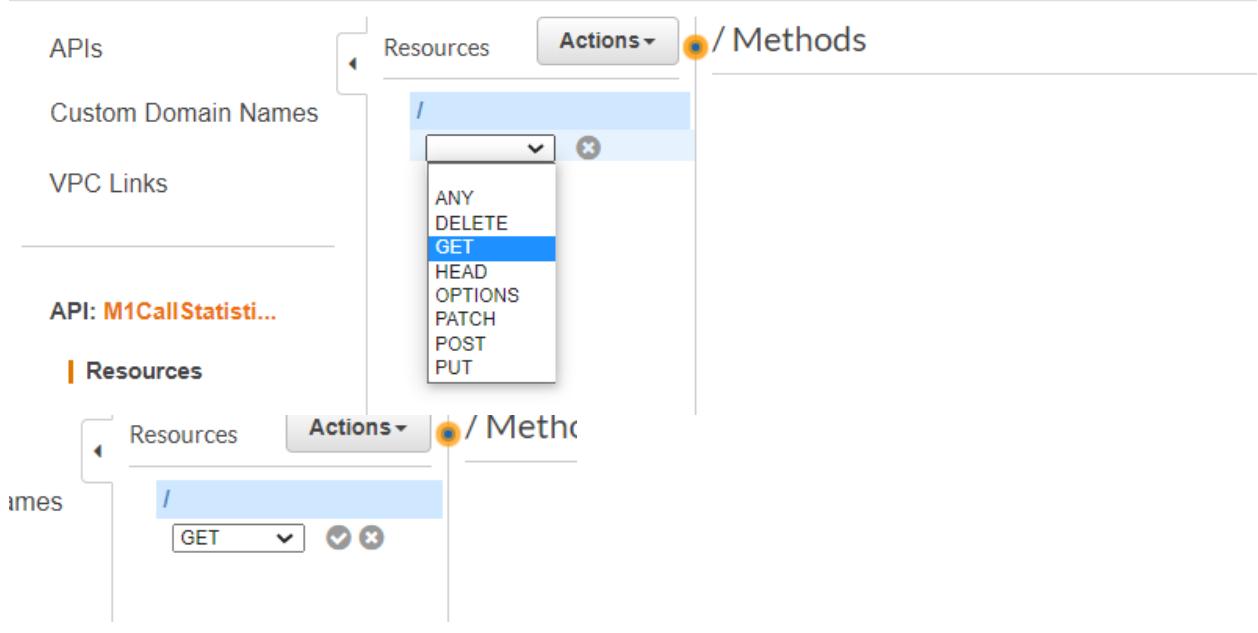
- Now we need to create Method. Click on Actions -> Create Method



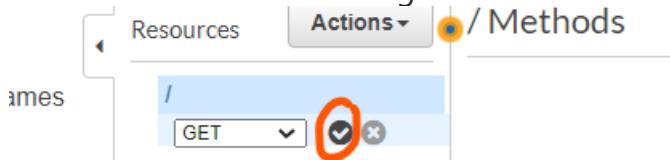
- It will give you method selection like below:



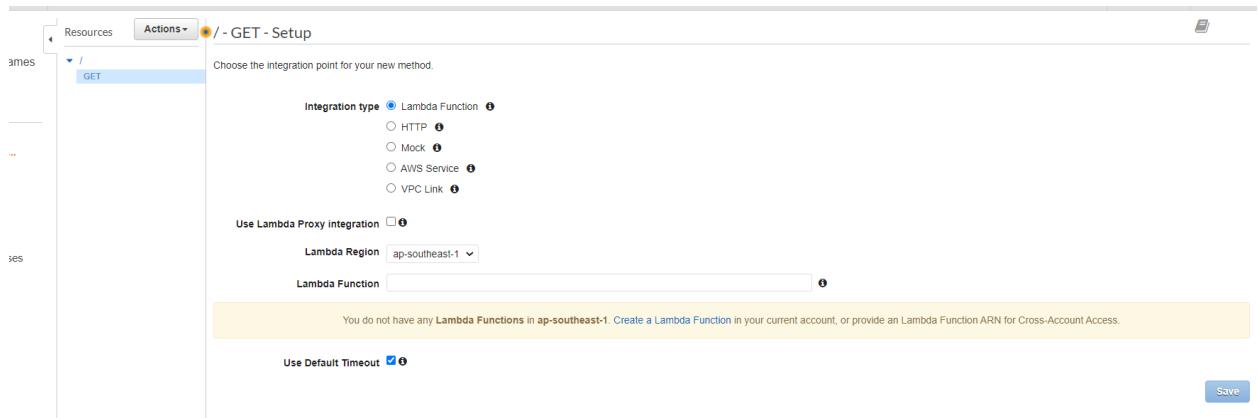
- Select "GET" from the list



- Click "tick" to save the changes

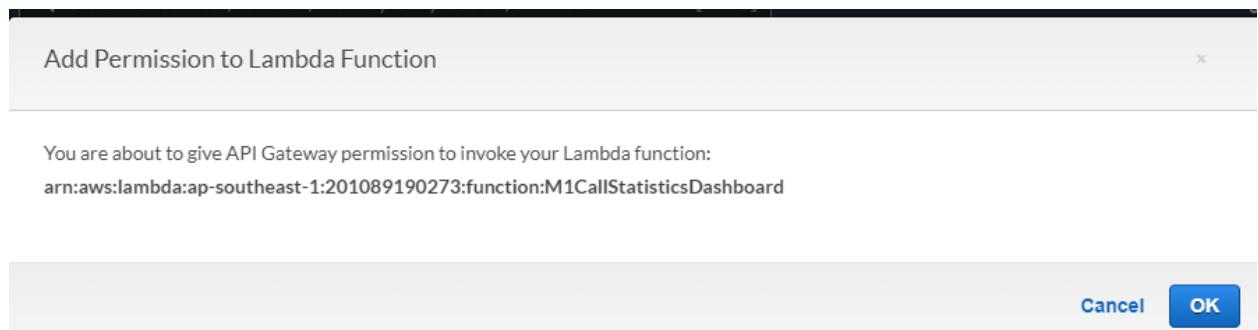


- Once method is created, you will be able to see below page

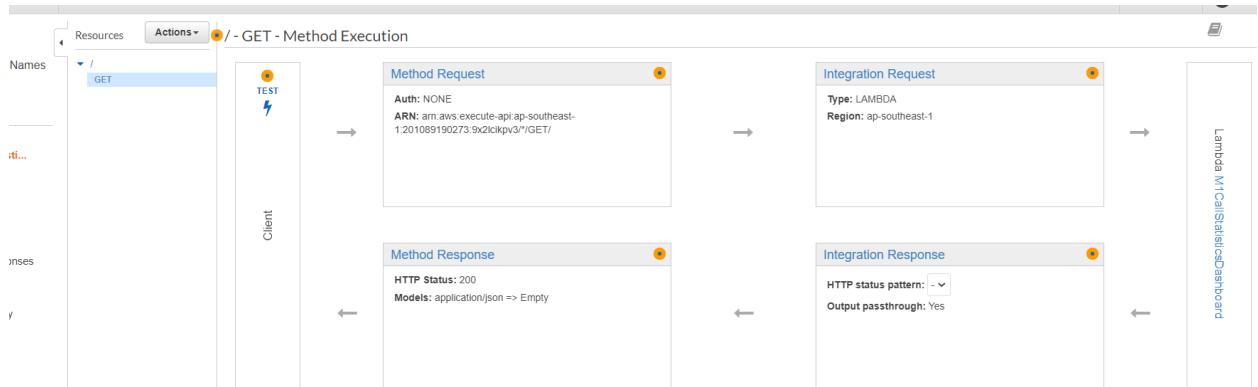


- Lambda integration with REST API
 - Choose Integration type as "Lambda Function"
 - Use lambda proxy integration – do not uncheck and leave blank
 - Lambda region : please choose Singapore region if you are located in Singapore, otherwise choose the region where lambda function is located
 - Lambda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

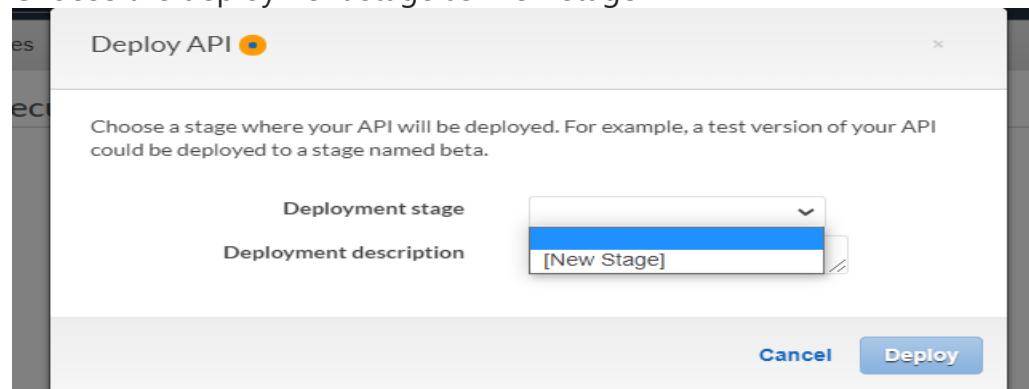
You will get confirmation to provide permissions, please click ok.



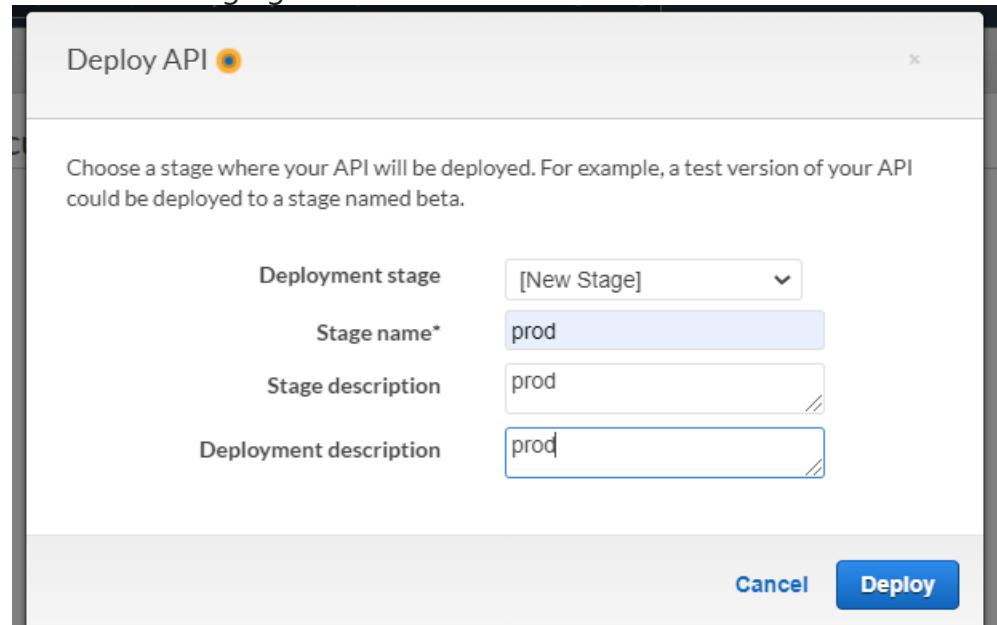
- Once done, you will navigate to below page



- 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.

The screenshot shows the 'prod Stage Editor' in the AWS API Gateway console. At the top, there's a navigation bar with 'APIs' and a stage name 'prod'. Below it, a sidebar lists 'Stage Names' and 'Responses'. The main area is titled 'prod Stage Editor' with a 'Create' button. A blue box highlights the 'Invoke URL' field, which contains the value 'https://9x2lcikpv3.execute-api.ap-southeast-1.amazonaws.com/prod'. Below this, the 'Settings' tab is active, showing 'Cache Settings' and 'Default Method Throttling' sections. Under 'Default Method Throttling', it says 'Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is 10000 requests per second' and '5000 requests'. There are checkboxes for 'Enable API cache' and 'Enable throttling' (which is checked), with input fields for 'Rate' (10000) and 'Burst' (5000).

- We need to enable CORS

- Click on "Resources" from left menu

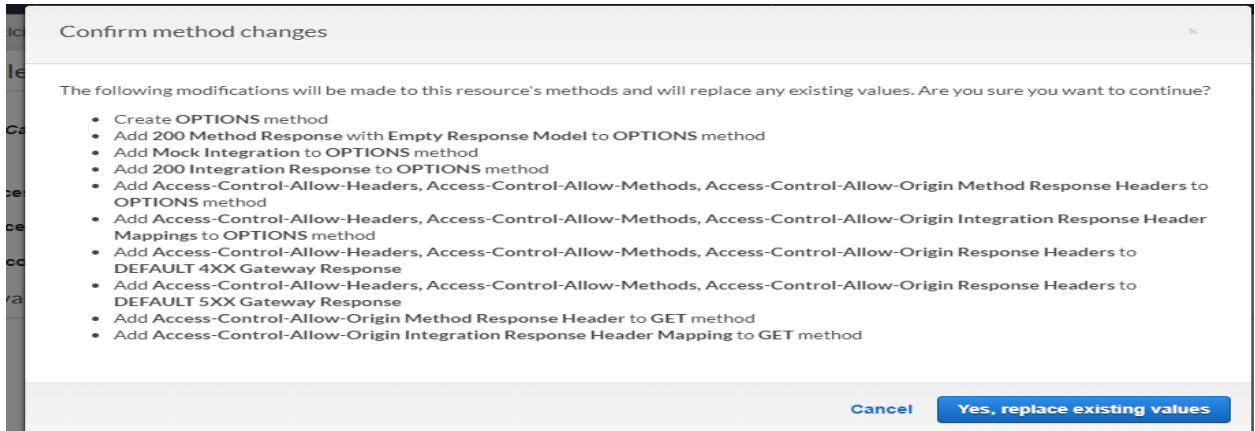
The screenshot shows the 'Resources' section in the AWS API Gateway console. The left sidebar has links for 'APIs', 'Custom Domain Names', 'VPC Links', 'API: M1CallStatistics...', 'Resources' (which is highlighted with a red circle), and 'Stages'. On the right, a 'Methods' section for the root resource '/' is shown. It lists a 'GET' method with the ARN 'arn:aws:lambda:ap-southeast-1:201089190273:...'. Under 'Authorization', it says 'None' and 'API Key Not required'.

- Click Actions -> Enable CORS

The screenshot shows the 'Actions' dropdown menu in the AWS API Gateway console. The left sidebar shows 'APIs', 'Custom Domain Names', 'VPC Links', 'API: M1CallStatistics...', 'Resources' (highlighted with a red circle), 'Stages', and 'Authorizers'. The right side shows a 'Methods' section for the root resource '/'. The 'Actions' dropdown is open, showing options like 'Create Method', 'Create Resource', 'Enable CORS' (highlighted with a red circle), 'Edit Resource Documentation', 'Deploy API', 'Import API', 'Edit API Documentation', and 'Delete API'. A tooltip for 'Enable CORS' shows 'a:ap-southeast-1:201089190273:...'. Below the dropdown, there are sections for 'None' and 'Not required'.

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

- Next it will ask you to confirm with the changes

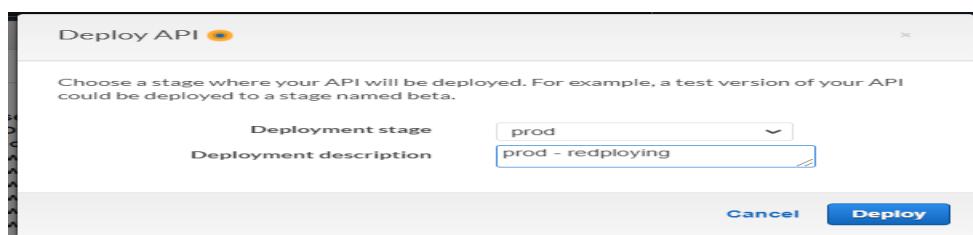
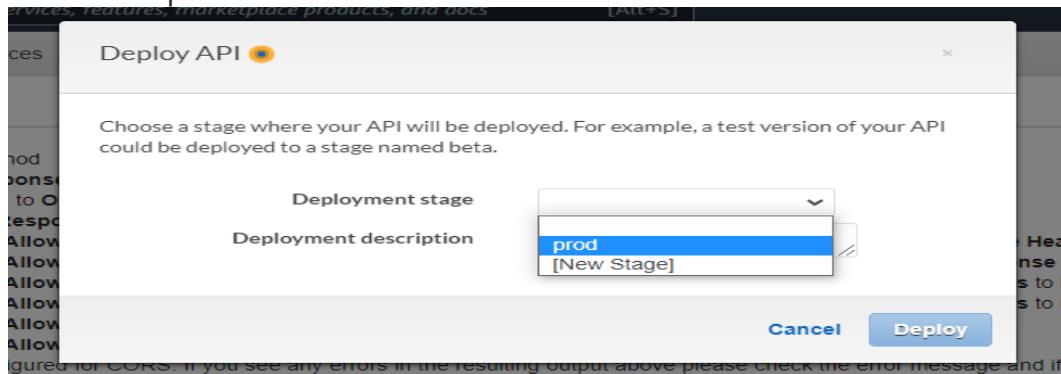


- Click on "Yes, replace existing values" button to enable CORS for this API
- Below is the confirmation on Enabling CORS

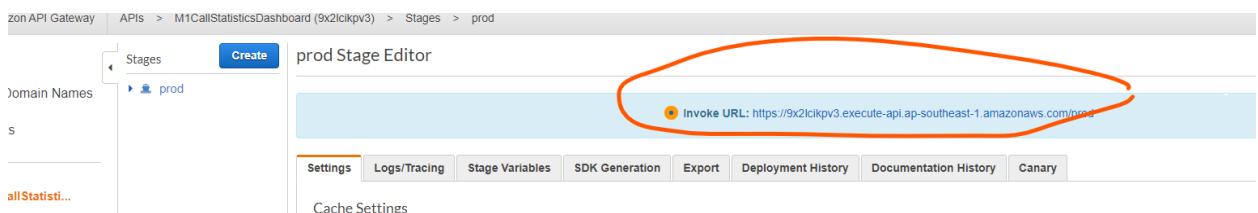
- Re-deploy API
 - Under Resources -> Click Actions -> Deploy API

The screenshot shows the 'Actions' dropdown menu for a specific API resource. The 'Enable CORS' option is highlighted with a red circle. Below it, under 'API ACTIONS', the 'Deploy API' option is also highlighted with a red circle.

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



- 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": "{\"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\":\"\\\"20210901073120\\\"\", \"CustomerJourneyFlow\":\"\\\"Non-Lex -> Pressed 1\\\"\", \"ID\":1, \"Channel\":\"\\\"VOICE\\\"\", \"ContactID\":\"\\\"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\\\"\", \"CustomerNumber\":\"\\\"+6563353529\\\"\", \"FlowType\":\"\\\"Agent Queue\\\"\", \"InitialContactID\":\"\\\"b1faaaa8d-6513-4a17-8d1d-c26667255f0\\\"\", \"InteractionDateTime\":\"\\\"2021083125231\\\"\", \"CustomerJourneyFlow\":\"\\\"BroadbandEnquiries\\\"\", \"ID\":2, \"Channel\":\"\\\"VOICE\\\"\", \"ContactID\":\"\\\"a52479ae-427b-49e1-b924-ceb3a33c490\\\"\", \"CustomerNumber\":\"\\\"+6563353529\\\"\", \"InteractionDateime\":\"\\\"20210831125139\\\"\", \"CustomerJourneyFlow\":\"\\\"InitialContact\\\"\", \"ID\":4, \"Channel\":\"\\\"VOICE\\\"\", \"ContactID\":\"\\\"a52479ae-427b-49e1-b924-bb041df922b\\\"\", \"CustomerNumber\":\"\\\"+6563353529\\\"\", \"InteractionDateime\":\"\\\"20210831125335\\\"\", \"CustomerJourneyFlow\":\"\\\"MobileEnquiries\\\"\", \"ID\":6, \"Channel\":\"\\\"VOICE\\\"\", \"ContactID\":\"\\\"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\\\"\", \"CustomerNumber\":\"\\\"+6563353529\\\"\", \"FlowType\":\"\\\"SMS Deflect\\\"\", \"InitialContactID\":\"\\\"a52479ae-427b-49e1-b924-bb041df922b\\\"\", \"CustomerNumber\":\"\\\"+6563353529\\\"\", \"FlowType\":\"\\\"Non-Lex\\\"\", \"InitialContactID\":\"\\\"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\\\"\", \"CustomerJourneyFlow\":\"\\\"Non-Lex -> Pressed 1\\\"\", \"ID\":1, \"Channel\":\"\\\"VOICE\\\"\", \"ContactID\":\"\\\"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\\\"\", \"CustomerNumber\":\"\\\"+6563353529\\\"\", \"FlowType\":\"\\\"Non-Lex\\\"\", \"InitialContactID\":\"\\\"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\\\"\", \"CustomerJourneyFlow\":\"\\\"2021083124908\\\"\", \"CustomerJourneyFlow\":\"\\\"Non-Lex -> Pressed 1\\\"\", \"ID\":5, \"Channel\":\"\\\"VOICE\\\"\", \"ContactID\":\"\\\"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\\\"\", \"CustomerNumber\":\"\\\"+6563353529\\\"\", \"FlowType\":\"\\\"Non-Lex\\\"\", \"InitialContactID\":\"\\\"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\\\"\", \"CustomerJourneyFlow\":\"\\\"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 **“ConnectCallStatsDashboard.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 & “ConnectCallStatsDashboard.html” file

```

351
352
353
354
355
356
357
358
359
360 var smsDeflectInteractionsCount = 0;
361 var nonLexInteractions = 0;
362 var agentQueuedInteractions = 0;
363
364 S.ajax({
365   type: 'GET',
366   url: 'https://9x2lckpv3.execute-api.ap-southeast-1.amazonaws.com/prod',
367   content-type: 'application/json',
368   ...

```

- 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

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

Buckets (20) **Create bucket**

- Under Create bucket configuration, provide as like below
 - Bucket name : **amazonconnectcallstashboard** 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.

- 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

The screenshot shows the AWS S3 Buckets list page. At the top, there are buttons for 'Create bucket', 'Copy ARN', 'Empty', and 'Delete'. A search bar contains the text 'call' with a result count of '3 matches'. The table below lists one bucket named 'amazonconnectcallstatsdashboard' located in 'Asia Pacific (Singapore) ap-southeast-1'. The bucket has 'Objects can be public' access and was created on '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:
- 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:

The screenshot shows the AWS S3 'Upload' interface. It includes a message to add files or folders, a 'Files and folders (0)' table with 'Add files' and 'Add folder' buttons, and a search bar. A modal window titled 'Select Folder to Upload' shows a file browser with the path 'CallStatisti... > CallStatsDashboard'. The 'assets' folder is selected. The modal also has 'Upload' and 'Cancel' buttons.

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

Upload 31 files to this site?

This will upload all files from "CallStatsDashboard". Only do this if you trust the site.

Upload

Cancel

- Then files will be loaded to console.

<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.

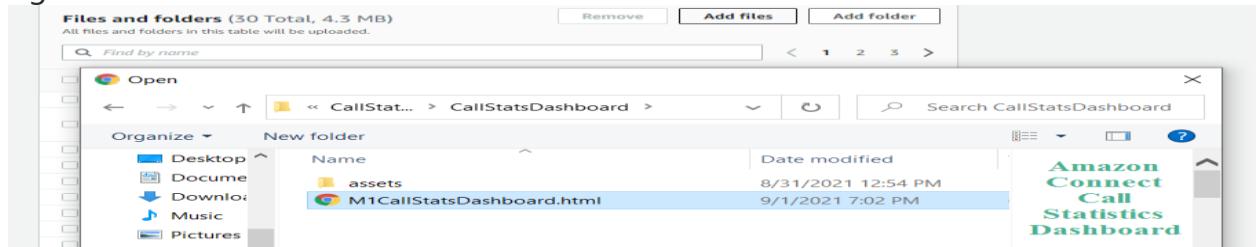
► Properties

Specify storage class, encryption settings, tags, and more.

Cancel

Upload

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



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



Files and folders (31 Total, 4.4 MB)							
Name	Folder	Type	Size	Status	Error		
FontAwesome.otf	assets/fonts/	-	131.6 kB	Succeeded	-		
M1CallStatsDashboard.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



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

Amazon S3 > m1connectcallstatisticsdashboard

Info

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.

Actions ▾

Upload

Find objects by prefix

Name	Type
assets/	Folder
CallStatsDashboard.html	html

Download as

Calculate total size

Copy

Move

Initiate restore

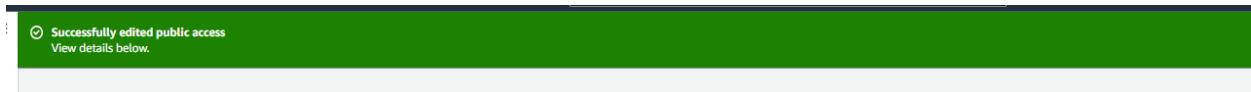
Query with S3 Select

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



Creating CloudFront to access the S3 objects from public

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

CloudFront > Distributions

Distributions (3) Info

Search all distributions

ID	Description	Domain name	Alternate do...	Origins	Status	Last modified
connectcallstatisticsdashbo...		connectcallstatisticsdashbo...		connectcallstatisticsdashbo...	Enabled	2023-09-11 10:59:59 UTC+08:00
connectcallstatisticsdashbo...		connectcallstatisticsdashbo...		connectcallstatisticsdashbo...	Enabled	2023-09-11 10:59:59 UTC+08:00
connectcallstatisticsdashbo...		connectcallstatisticsdashbo...		connectcallstatisticsdashbo...	Enabled	2023-09-11 10:59:59 UTC+08:00

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

Create distribution

Origin

Origin domain

Choose an AWS origin, or enter your origin's domain name.

Origin path - optional Info

Enter a URL path to append to the origin domain name for origin requests.

Name

Enter a name for this origin.

- Name field automatically will populate as "connectcallstatisticsdashboard.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>							
Create distribution <input type="button" value="C"/> <input type="button" value="Enable"/> <input type="button" value="Disable"/> <input type="button" value="Delete"/>							
<input type="text"/> Search all distributions							
ID	Description	Domain name	Alternate do...	Origins	Status	Last modified	
E14499TJ9H3TRA	-	d1p3jdr6c58knh.cloudfront.net	-	m1connectcallstatisti	<input checked="" type="checkbox"/> Enabled		

- 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>							
Create distribution <input type="button" value="C"/> <input type="button" value="Enable"/> <input type="button" value="Disable"/> <input type="button" value="Delete"/>							
<input type="text"/> Search all distributions							
ID	Description	Domain name	Alternate do...	Origins	Status	Last modified	
E14499TJ9H3TRA	-	d1p3jdr6c58knh.cloudfront.net	-	m1connectcallstatisti	<input checked="" type="checkbox"/> Enabled		

Launch dashboard

Launch the CloudFront URL followed by dashboard html file (<https://dtsqj0923a720.cloudfront.net/ConnectCallStatsDashboard.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