Amazon Connect Call statistics Solution

Contents

Installation Instructions

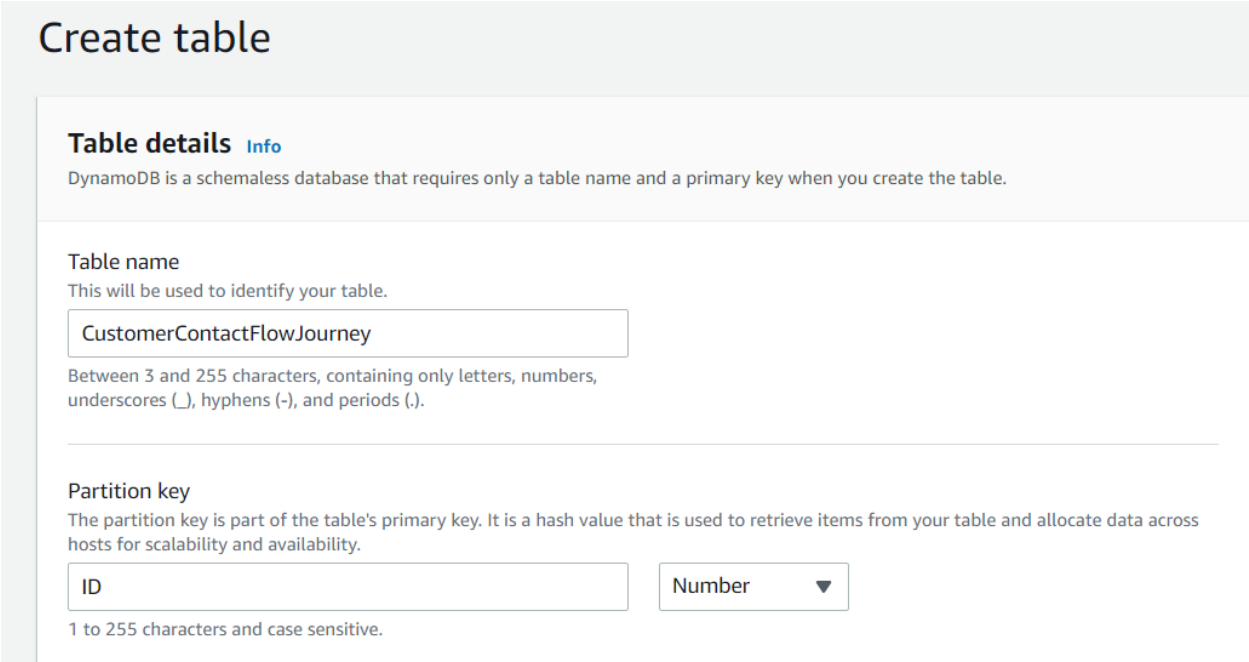
Pre-requisites

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

  https://github.com/ajvvenkatesh/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.

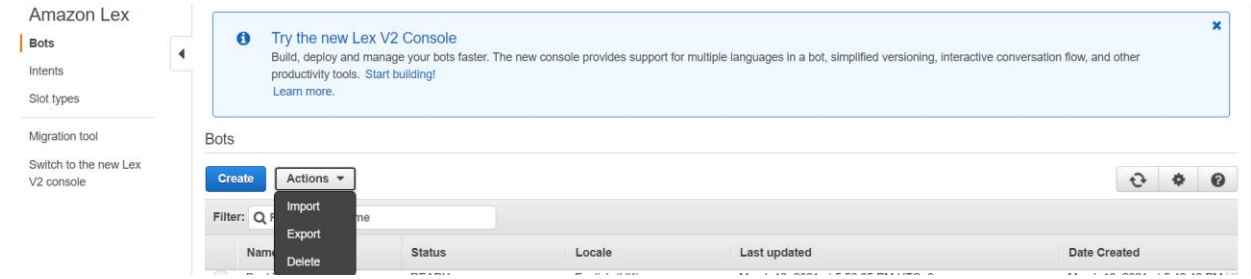


- ✓ Basic knowledge on C# coding to create server-less functions as lambdas. There are two lambda functions which we are using in this example

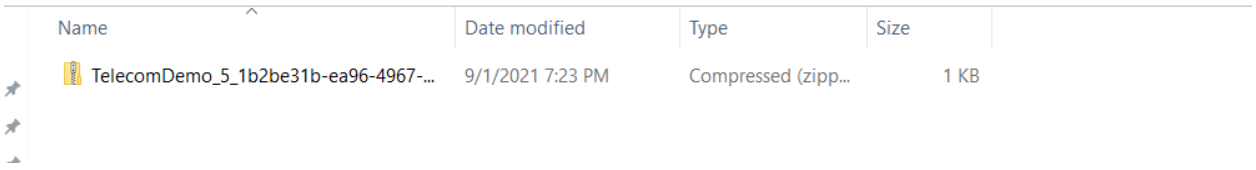
Steps to import TelecomDemo Amazon Lex Bot

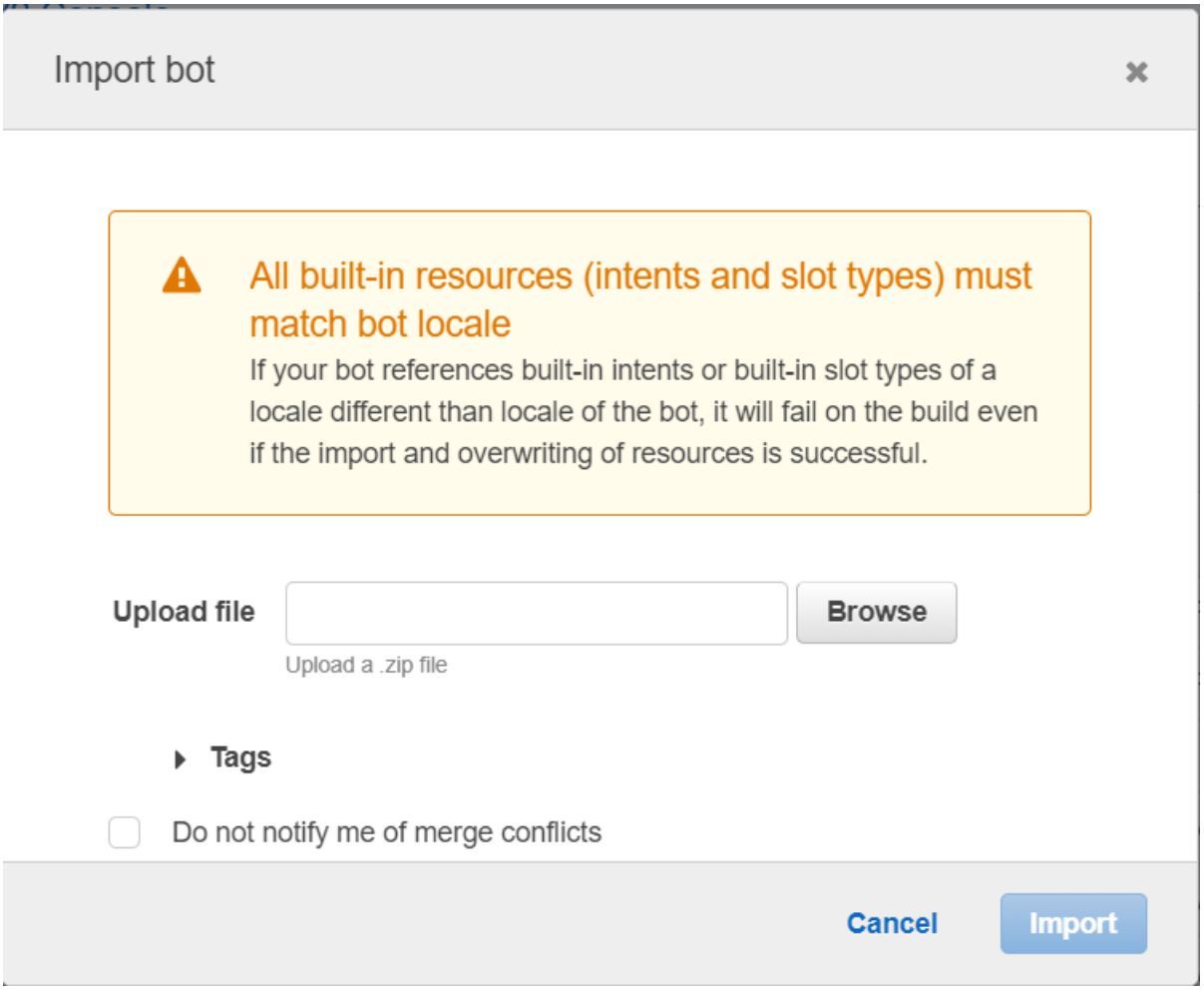
This bot is needed for the contact flow. This bot helps in branching the Amazon Connect contact flow decision making when the customer calling.

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

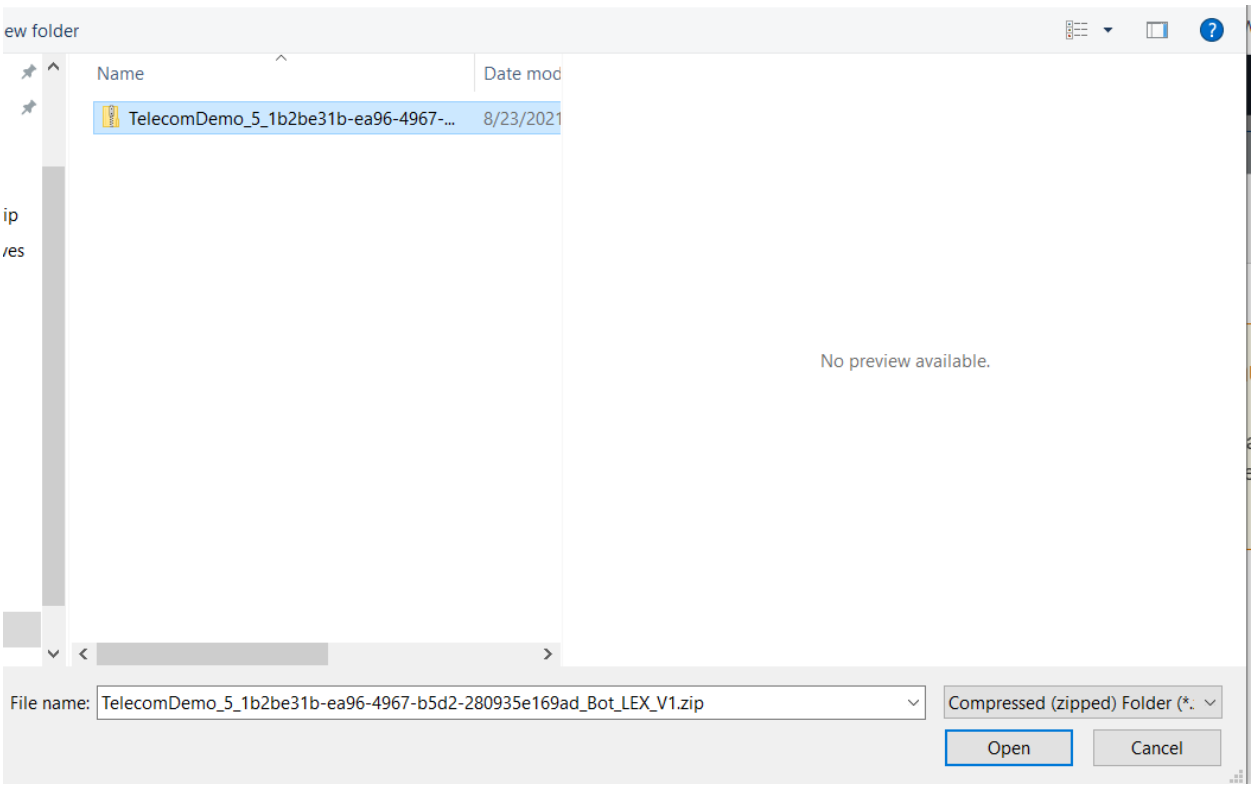


4. Click Actions → Import bot → Upload the zip file from `Unzipped Lex Folder`

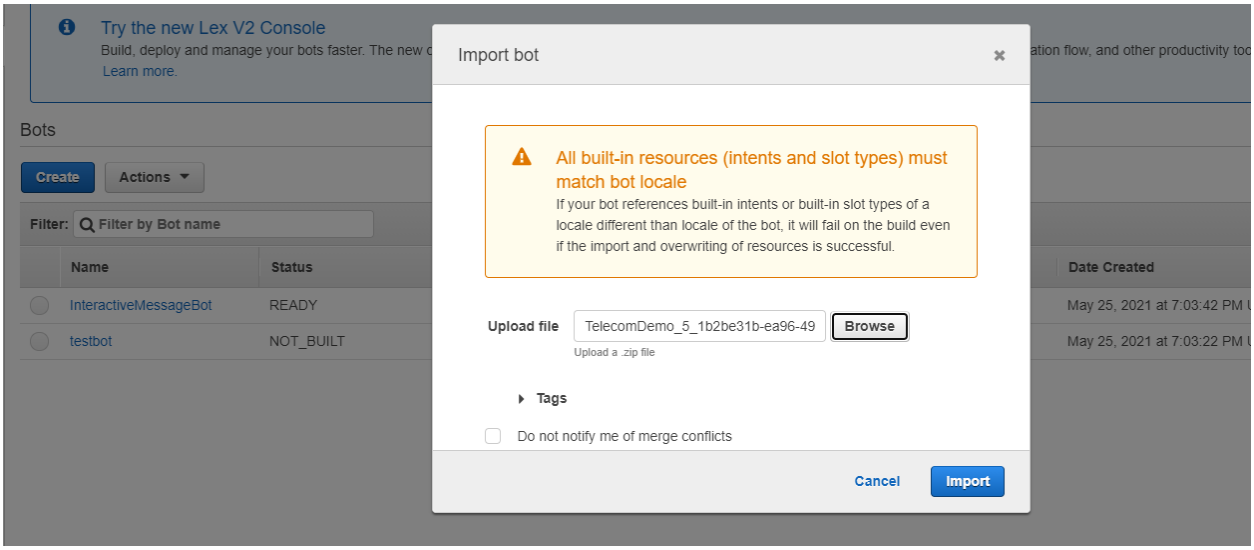




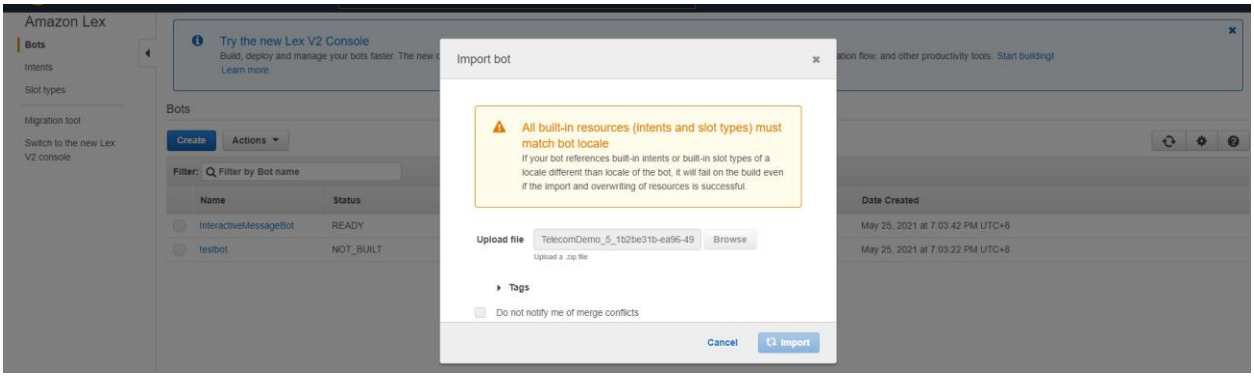
Click on browse to upload the Bot zip file.



Click on Open.

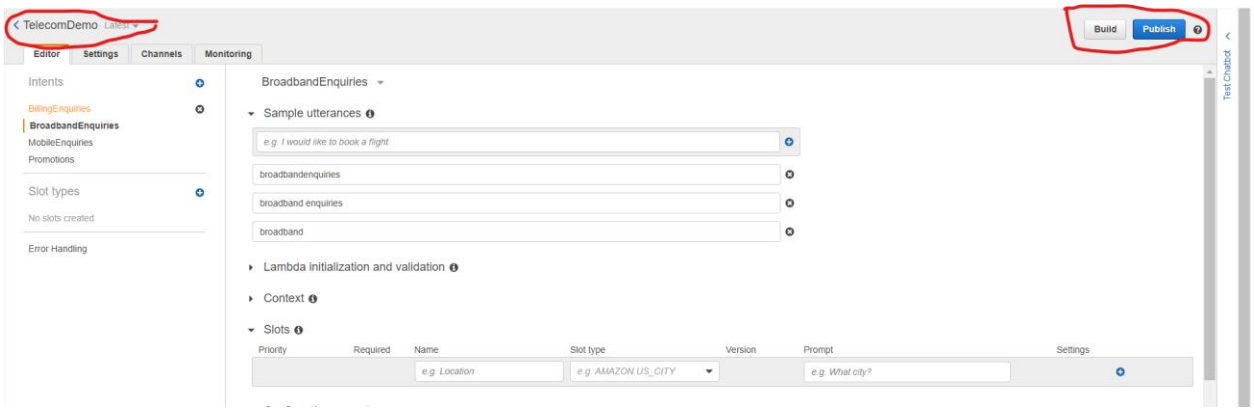


Click on Import button to load the Bot data to the console.

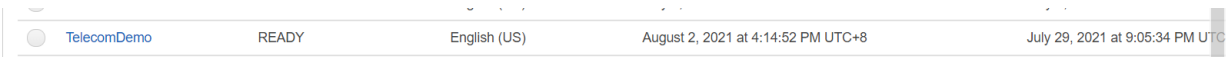


If it asks for overwriting intents click overwrite and the TelecomDemo Amazon Lex bot gets imported in NOT_BUILT status.

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

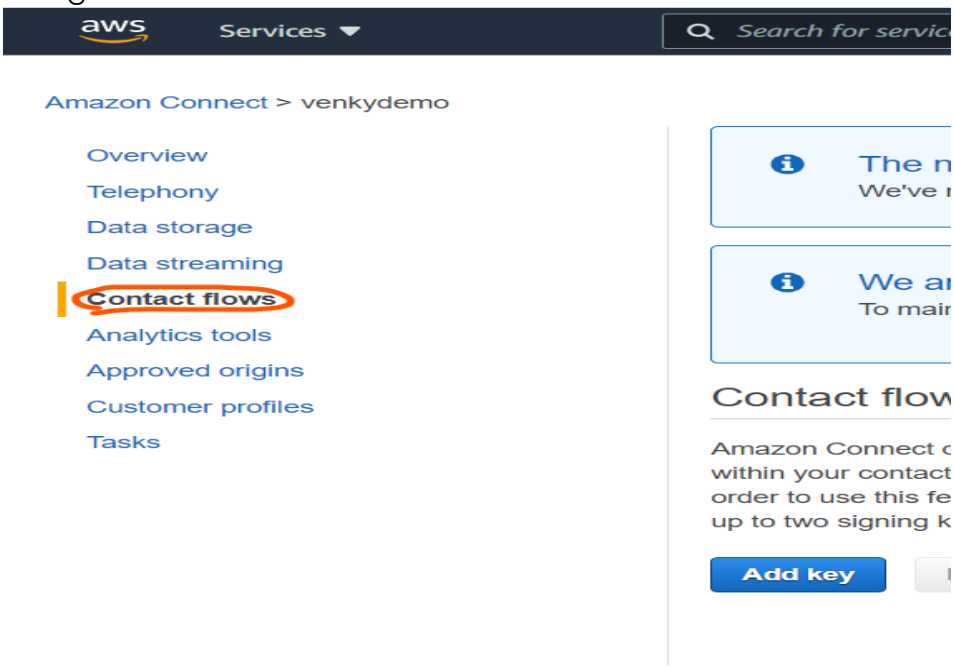


6. If it asks you for alias put TelecomDemo as alias and click Publish.
7. After successfully published, you can see the bot status as READY like mentioned below:

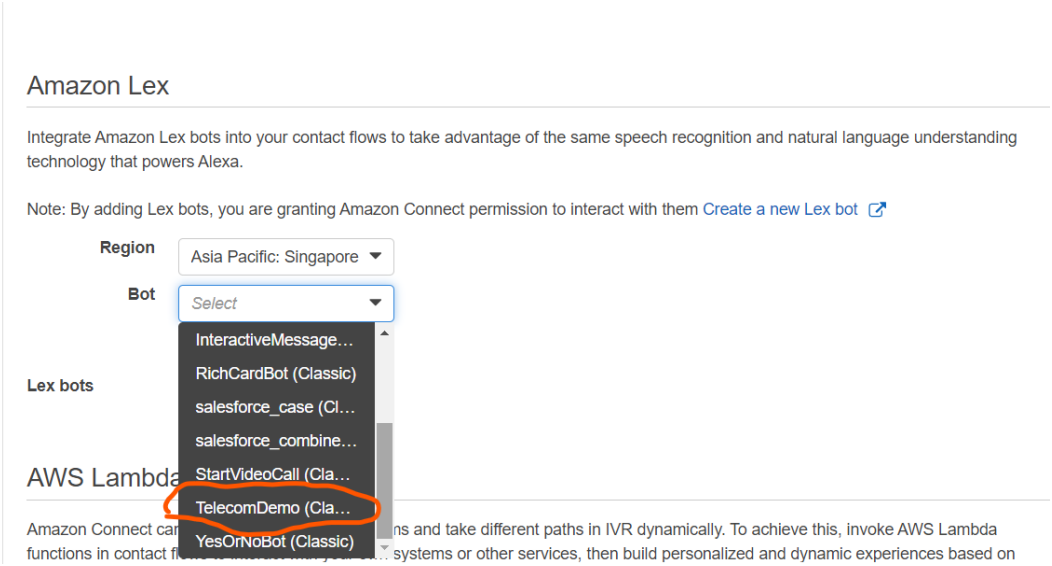


Step to add Lex bot to Amazon Connect Instance.

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:

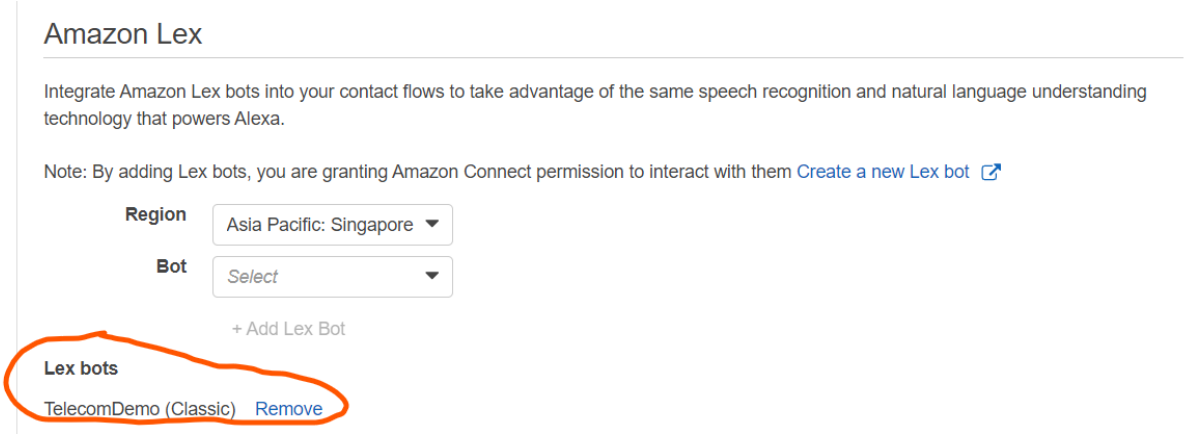


3. Go to Amazon Lex section from the right panel. Choose the region as "Asia Pacific:Singapore" or the region where connect instance is created. Select the drop down of Bot and then choose "Telecom Demo" bot which we have created in previous steps.



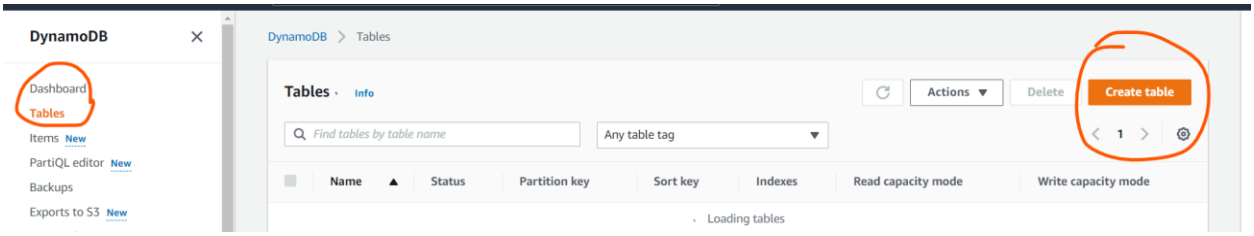
Once bot selected then click on "**+Add Lex Bot**" button to add the bot to the connect instance.

Bot will be assigned like mentioned below:

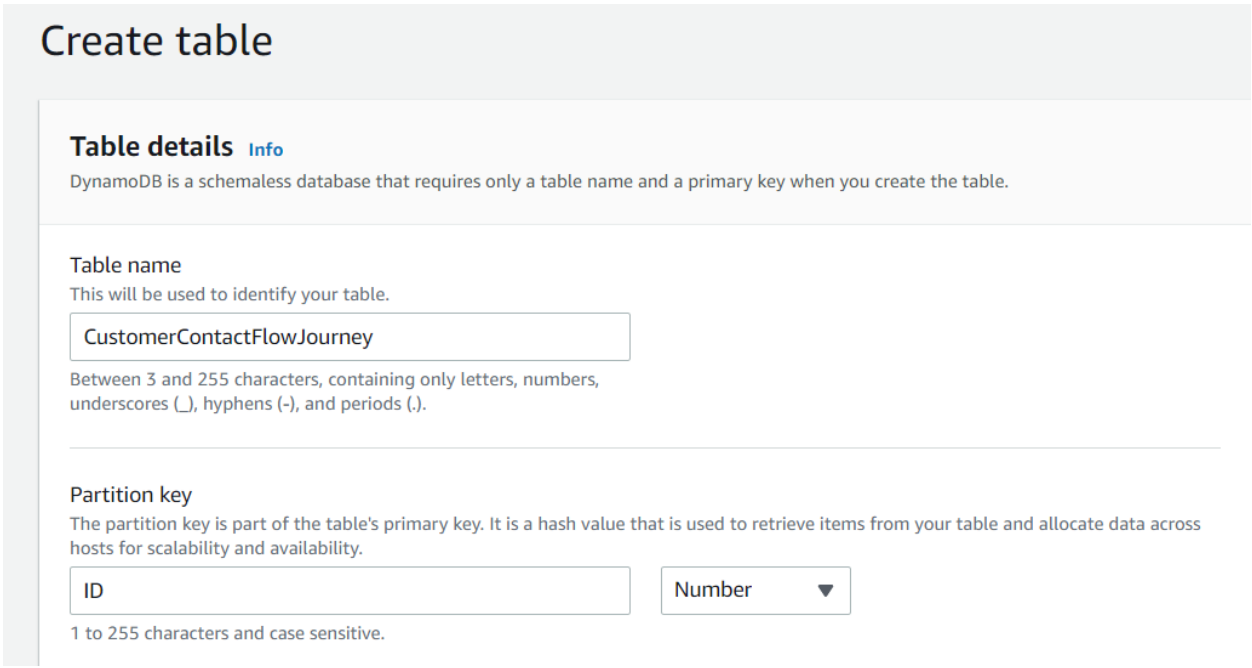


Steps to create DynamoDB table:

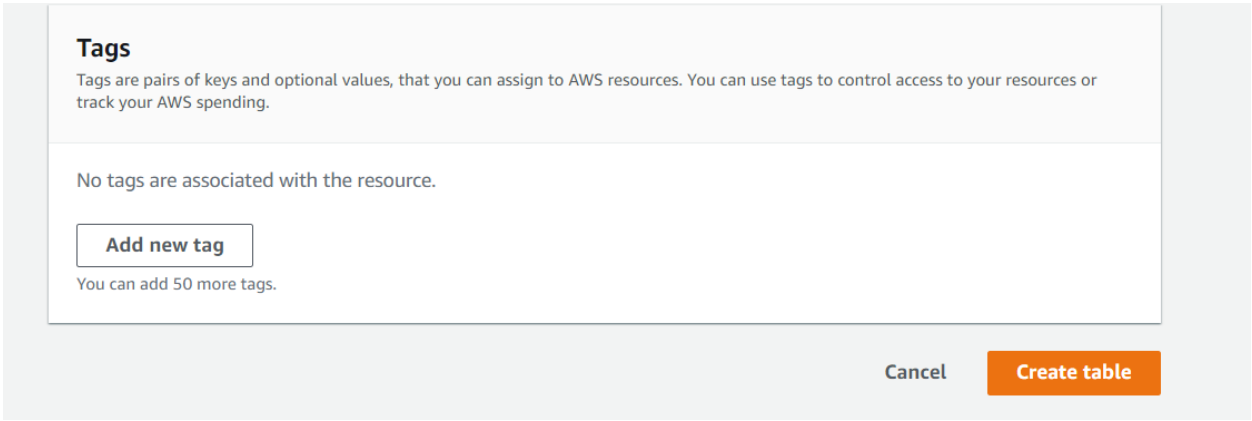
1. Go to [DynamoDB]
2. Navigate to Tables from the left menu and click on Create table button which appears on the top right corner like mentioned below:



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.



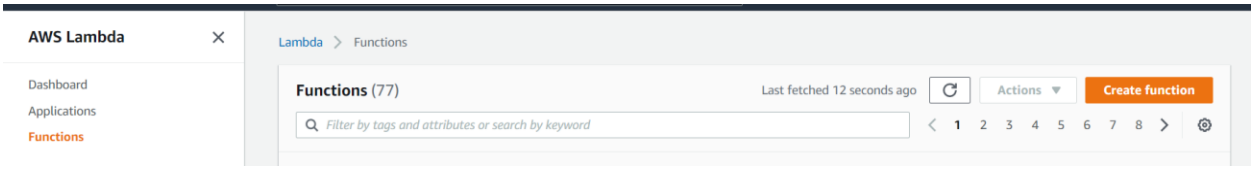
Leave other details default and proceed with creating table by clicking on “Create table” button.



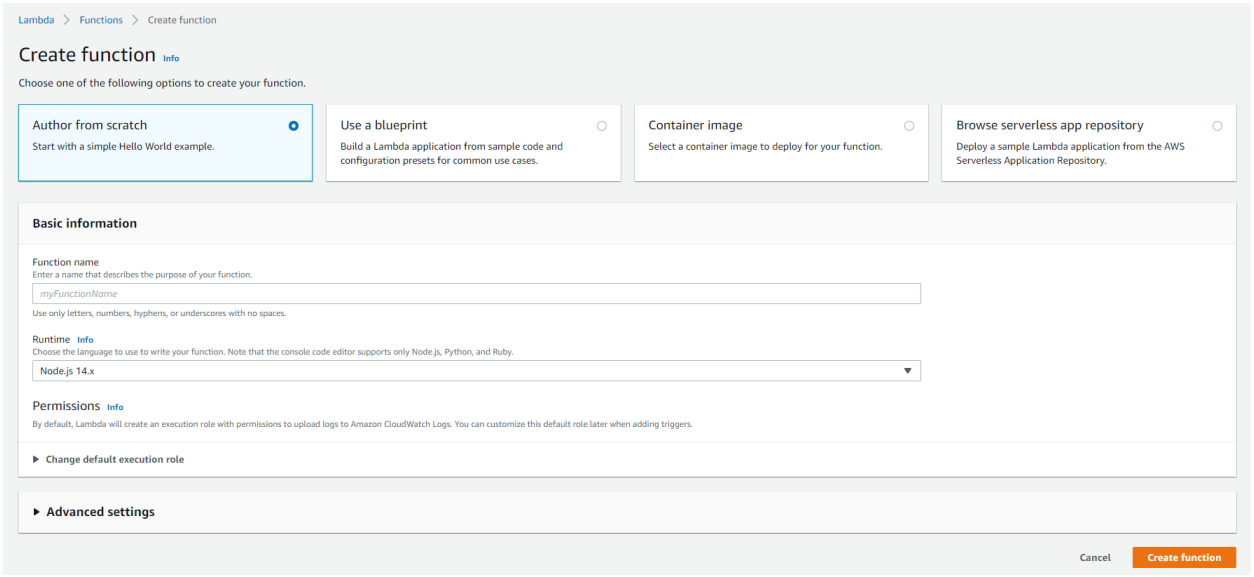
Steps to create or upload lambda functions.

Upload “AddCustomerContactFlowJourneyData“ lambda :

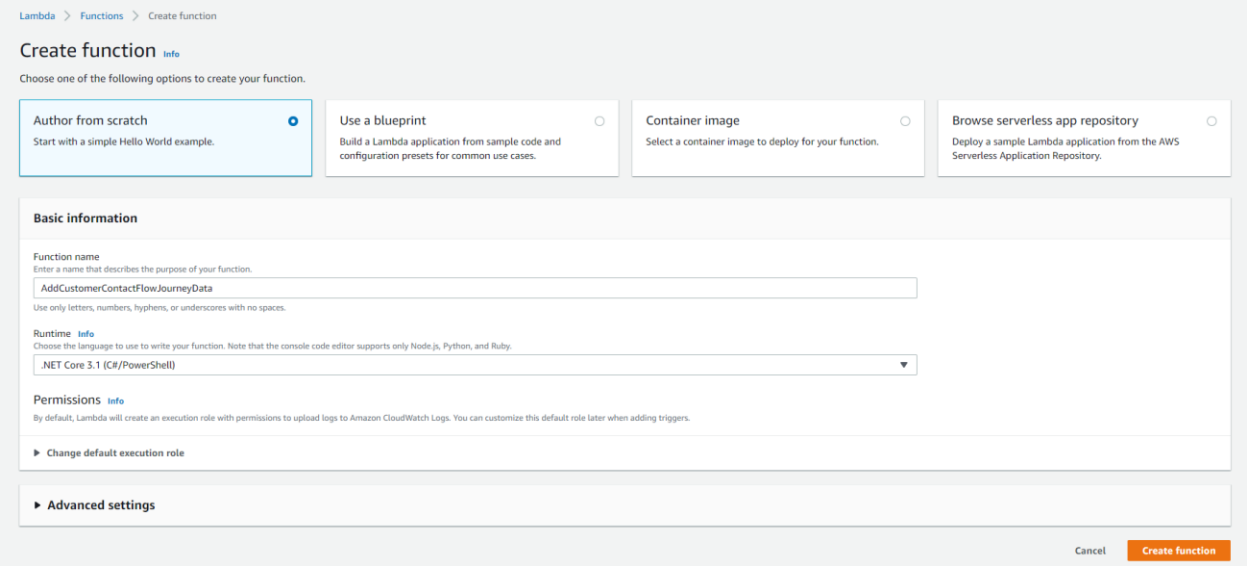
- ✓ Download the lambda zip file from the below github link.
- ✓ Navigate to lambda console
- ✓ Click on Functions from left menu



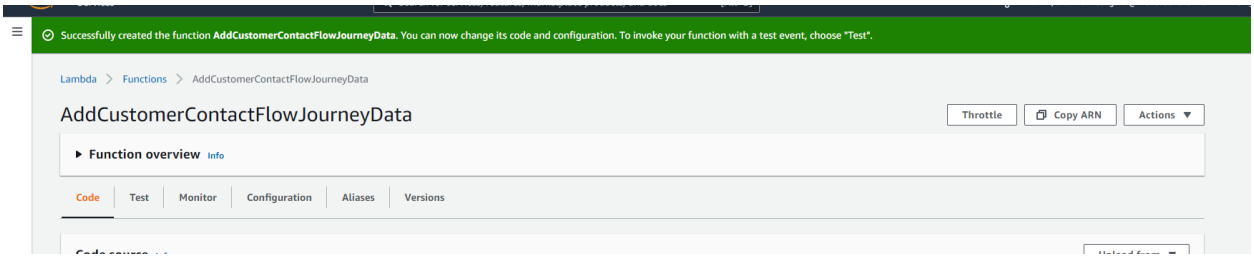
- Click on “Create Function” button at the right top corner to create function which will navigate to the page where you can provide details to create lambda function like mentioned below:



- ✓ Choose “Author from scratch”
- ✓ Provide function name as “`AddCustomerContactFlowJourneyData`”.
- ✓ Select Runtime as “.NET Core 3.1 (C#/PowerShell)” from the given list.
- ✓ Final options should be like mentioned below:

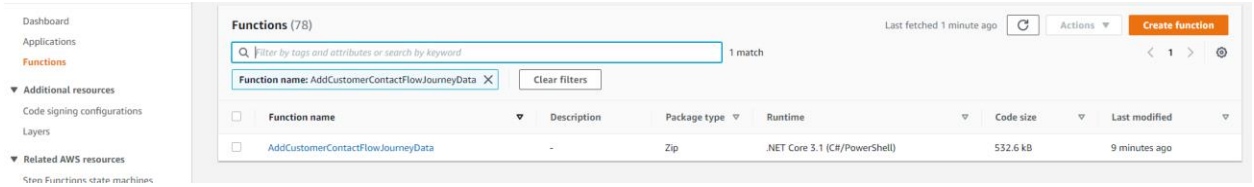


- Leave other values as default and click on “Create Function” from the bottom right corner to proceed.
- Once function is successfully created then you will be able to see below notification

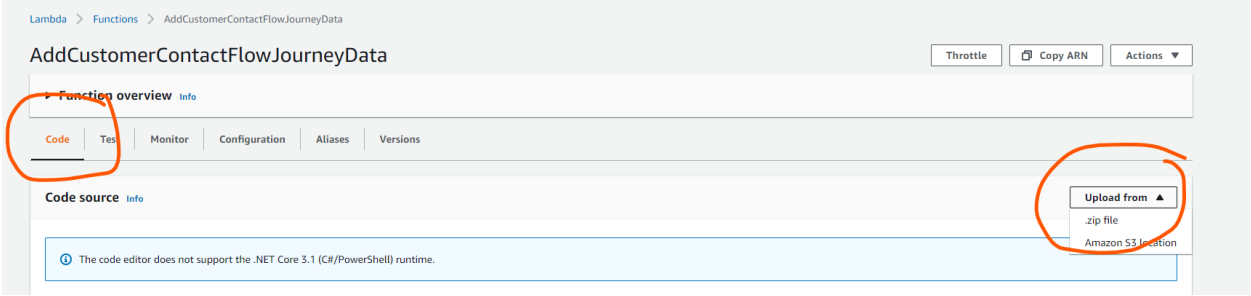


✓ Since we have successfully created lambda function. By default it will be empty function and now have to upload the code from our repository.

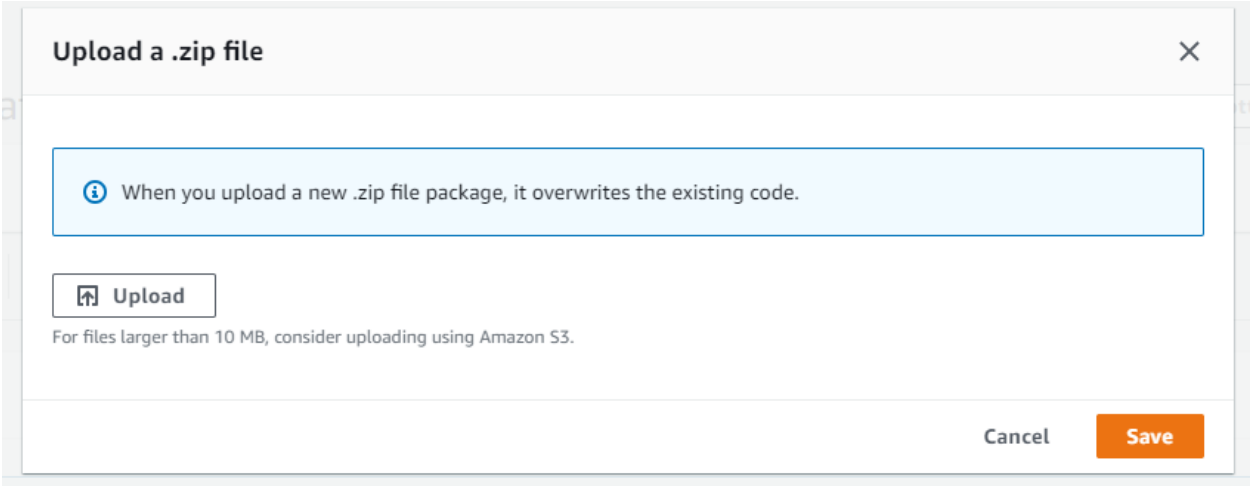
✓ Navigate to functions section from Lambda console and search for “`AddCustomerContactFlowJourneyData`” lambda function like mentioned below



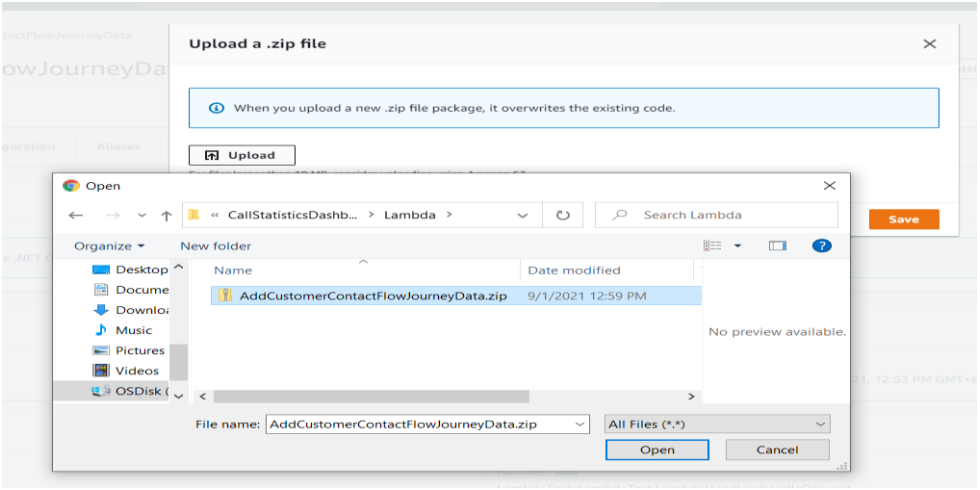
- Click on “`AddCustomerContactFlowJourneyData`” function which will navigate to the page where we can upload code.
- Click “.Zip file” from uploadfile option under the Code tab like mentioned below:



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



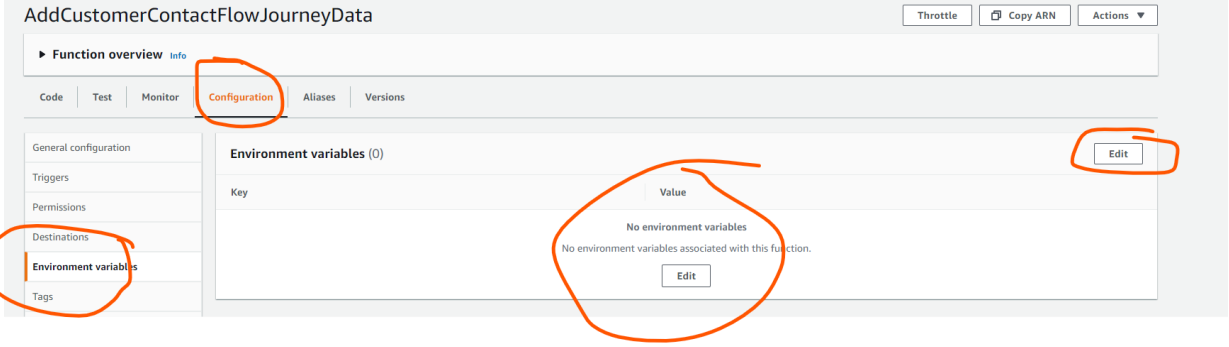
- Click “Upload” button and navigate to the location where lambda is downloaded earlier (Under Lambda folder from the zip file) . And choose the lambda and click on “Save” button to upload like mentioned below:



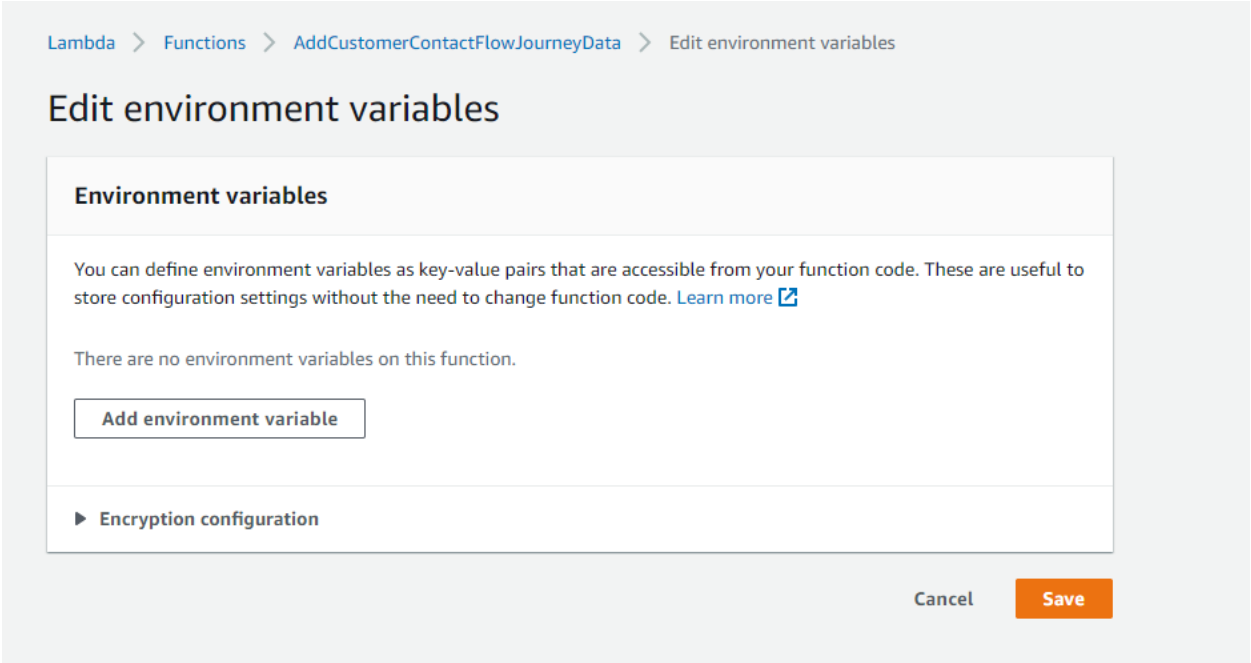
✓ After successfully loaded the zip file, we will get success notification. Please wait for some time to upload the function as it will take time.

Configuration section:

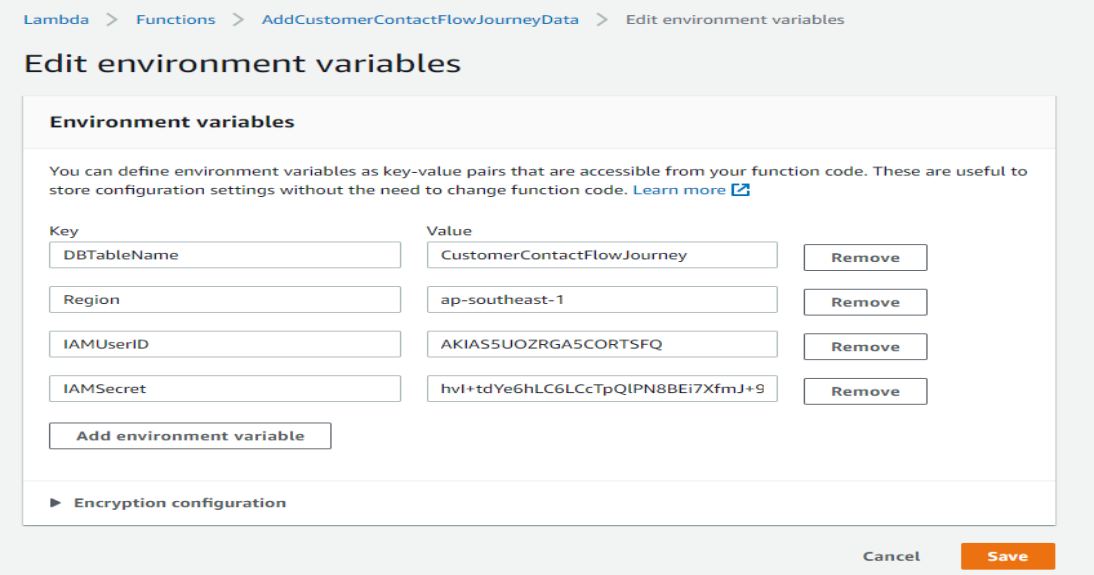
- Navigate to function, then click on Configuration tab and select “Environment Variables” like mentioned below:



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

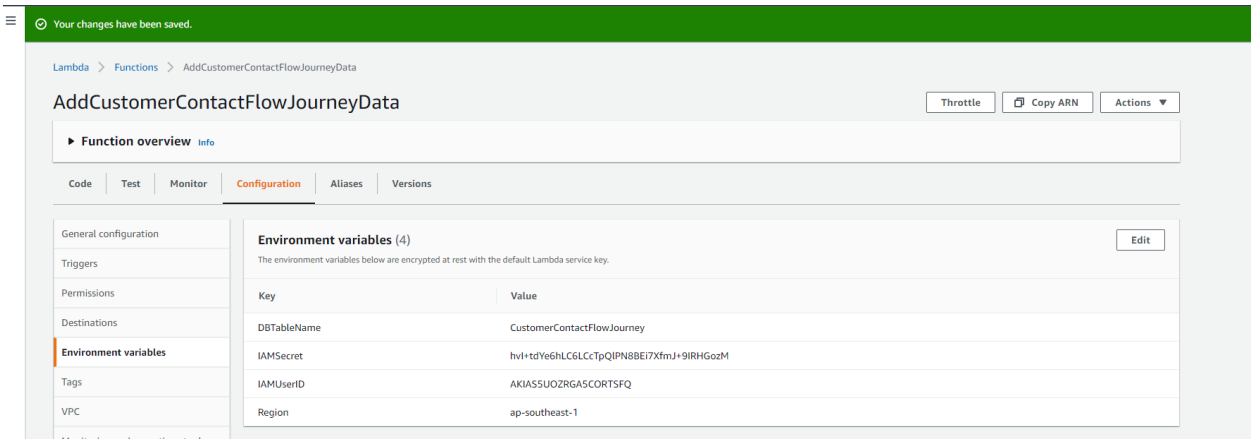


- Click on “Add environment variable” button to add the variables and values as key value pair like mentioned below



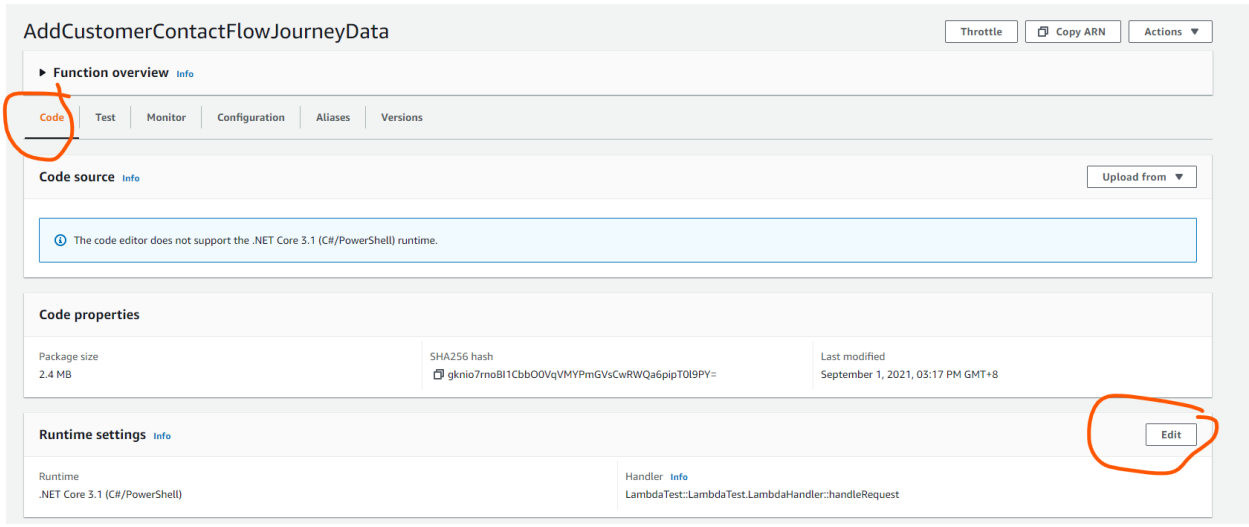
Add your IAM UserID & Secret key properly.

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



Change the function handler

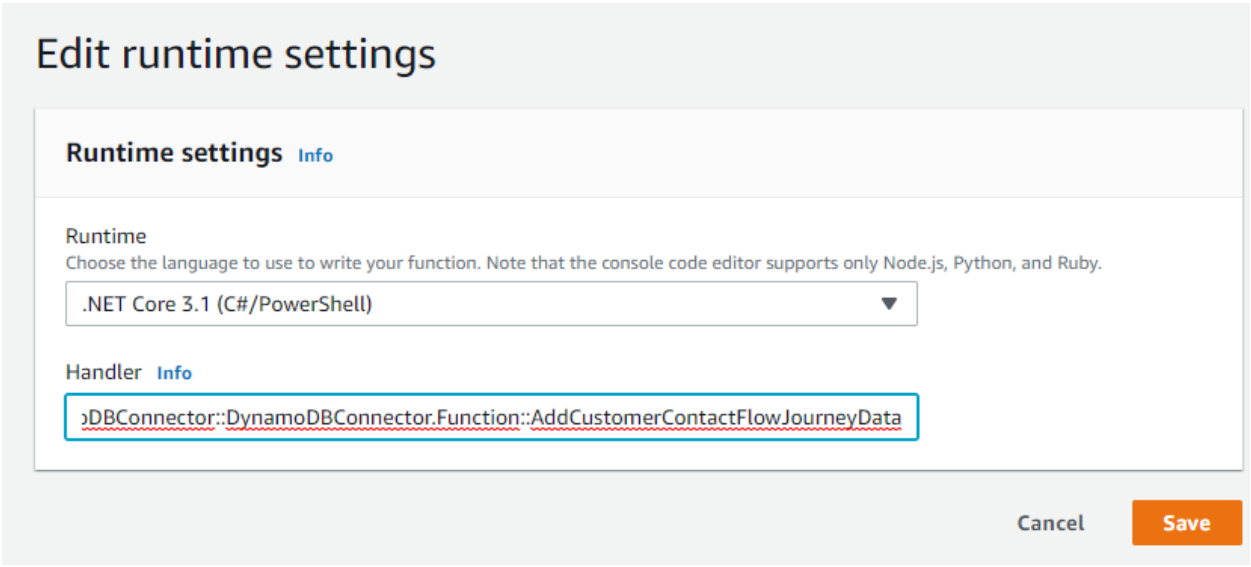
- Go to code tab and click on Edit button under Run time settings section like mentioned below



- Change the Handler from

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

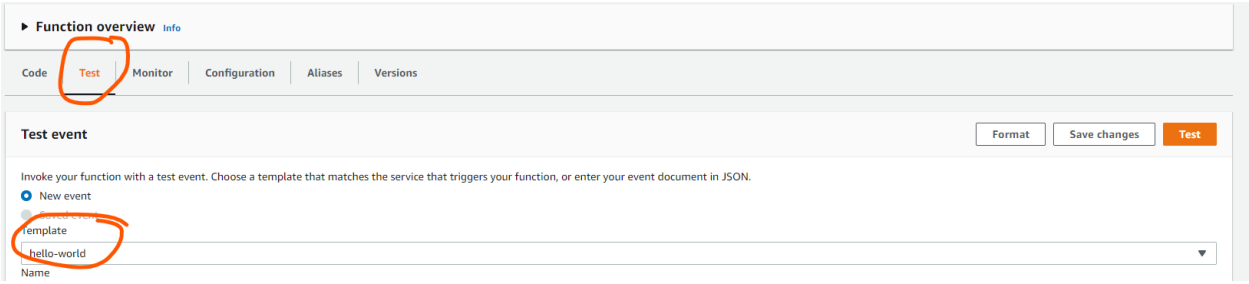
  ”DynamoDBConnector::DynamoDBConnector.Function::AddCustomerContactFlowJourneyData”



Then click on save button to save the handler settings.

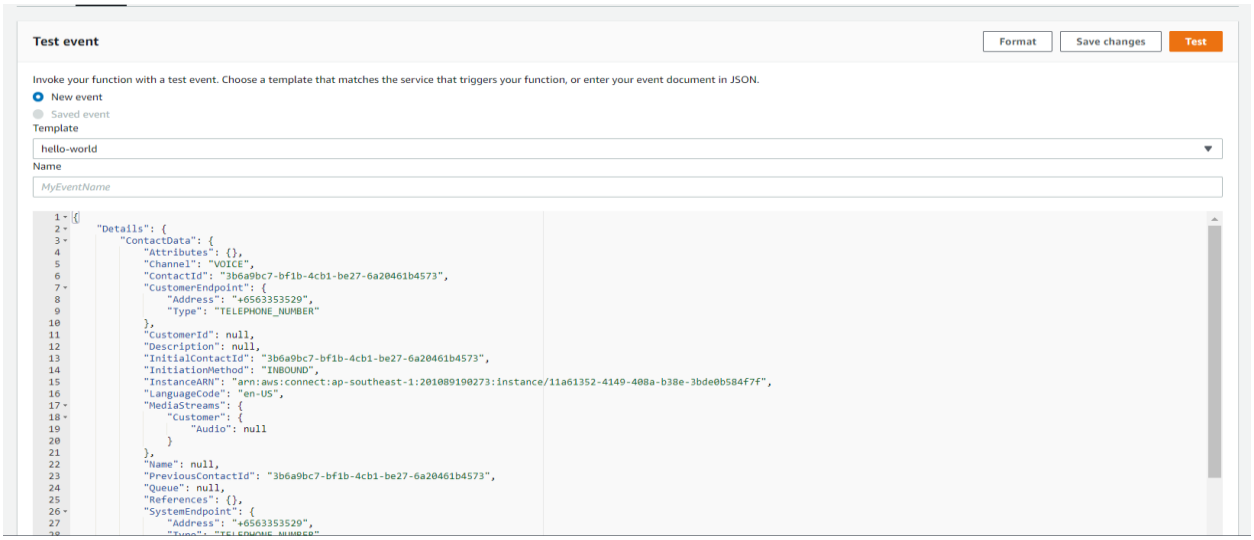
Test the lambda function section:

- Naviagate to Test tab under function overview -> Select template as “Hello-world”

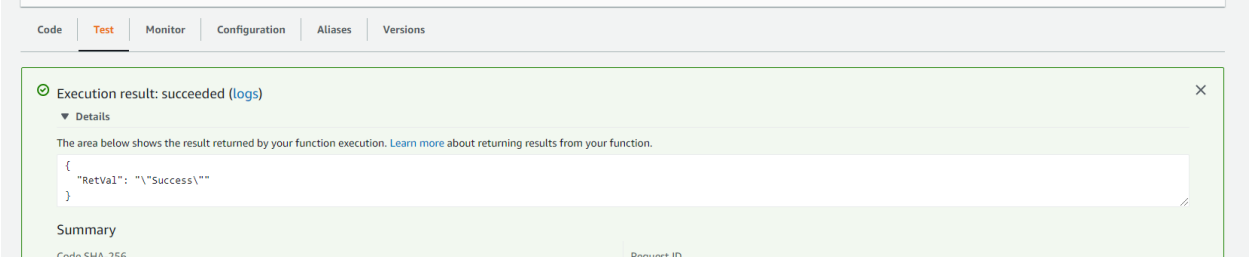


- Copy below json in the json field to test the function

```
{ "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" }
```

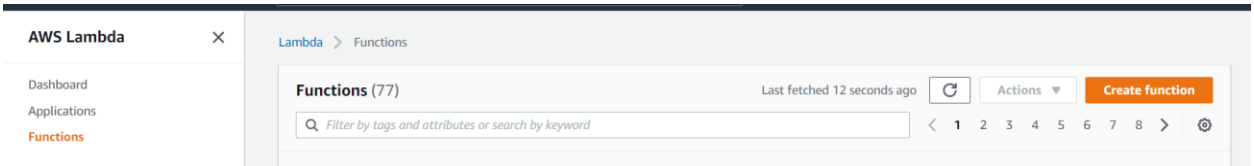


- Click on “Test” button to test the function
  If the function gets success then you will be able to see below success message.

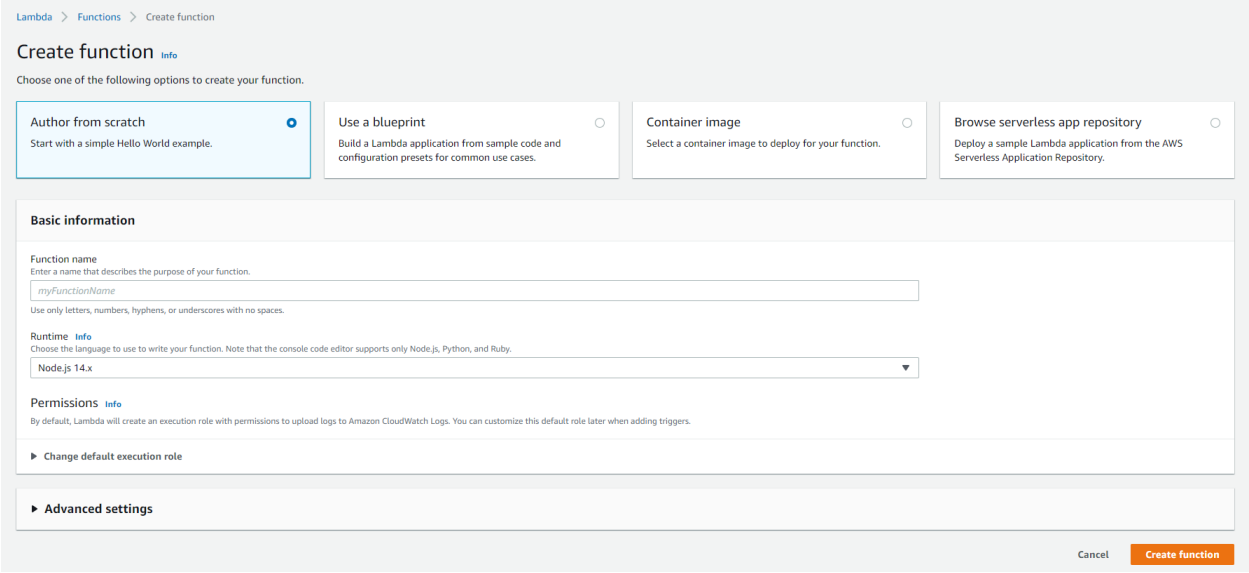


Upload “SendSMSFromContactFlowToCaller“ lambda :

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

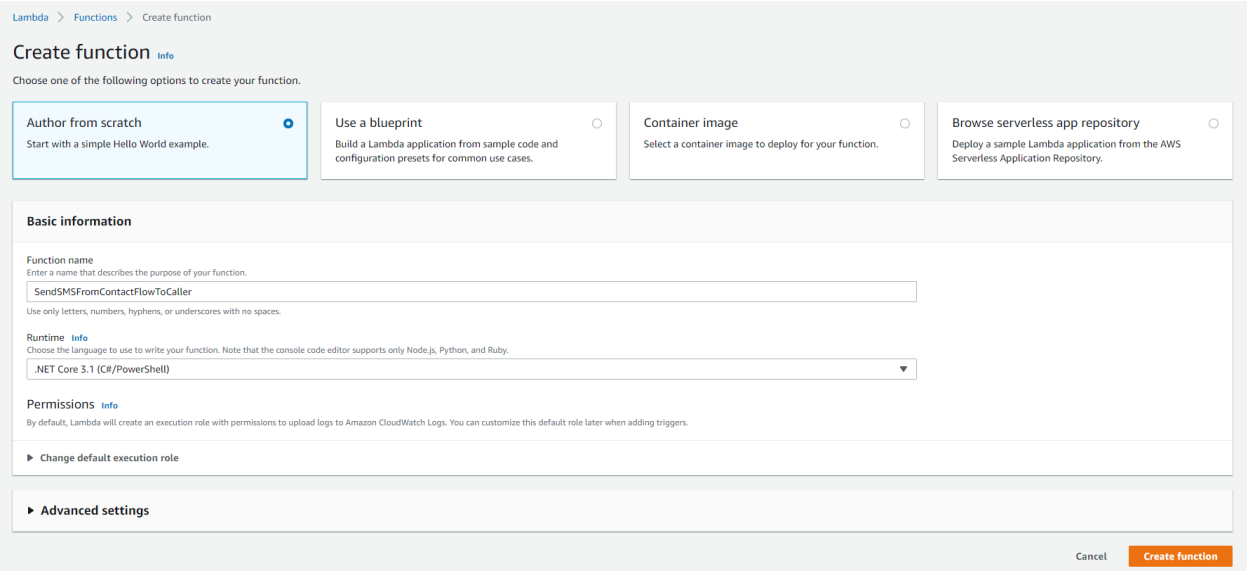


- Click on “Create Function” button at the right top corner to create function which will navigate to the page where you can provide details to create lambda function like mentioned below:

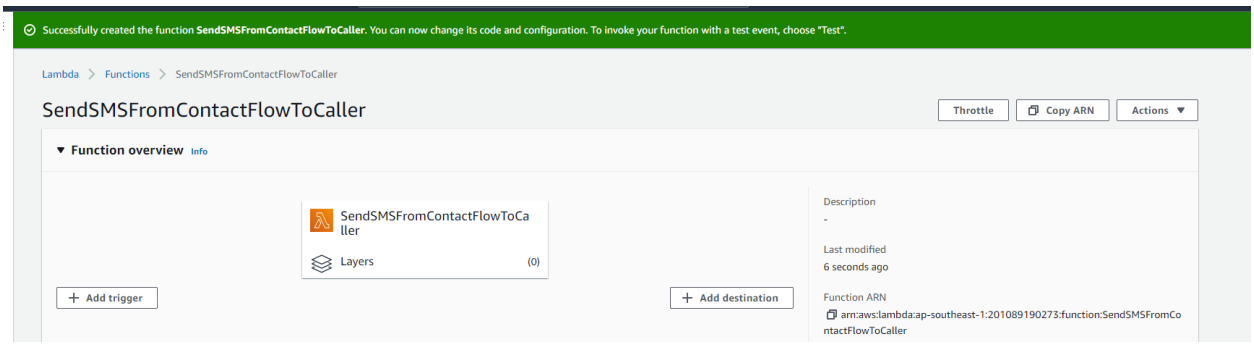


- ✓ Choose “Author from scratch”
- ✓ Provide function name as “`SendSMSFromContactFlowToCaller`”.
- ✓ Select Runtime as “.NET Core 3.1 (C#/PowerShell)” from the given list.

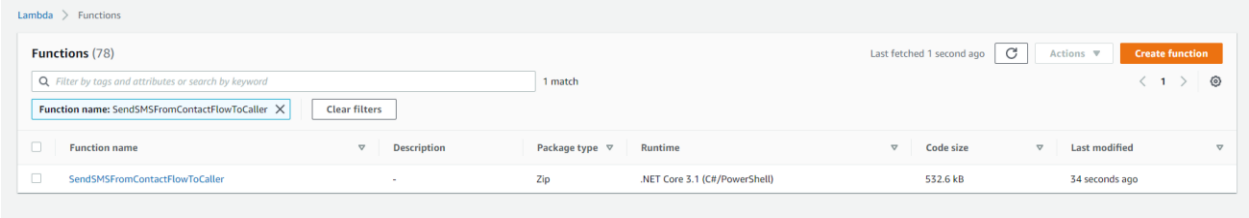
Final options should be like mentioned below:



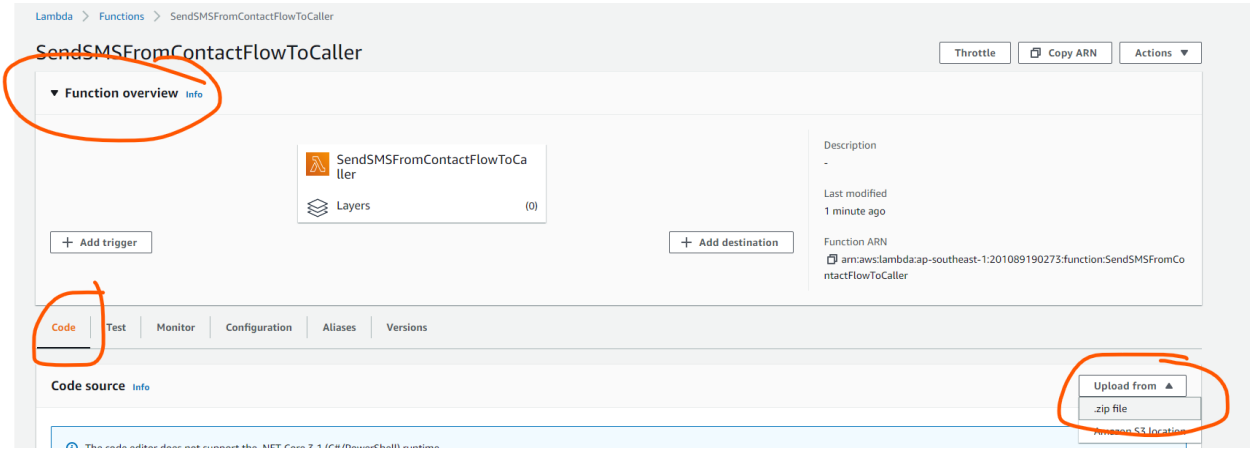
- Leave other values as default and click on “Create Function” from the bottom right corner to proceed.
- Once function is successfully created then you will be able to see below notification



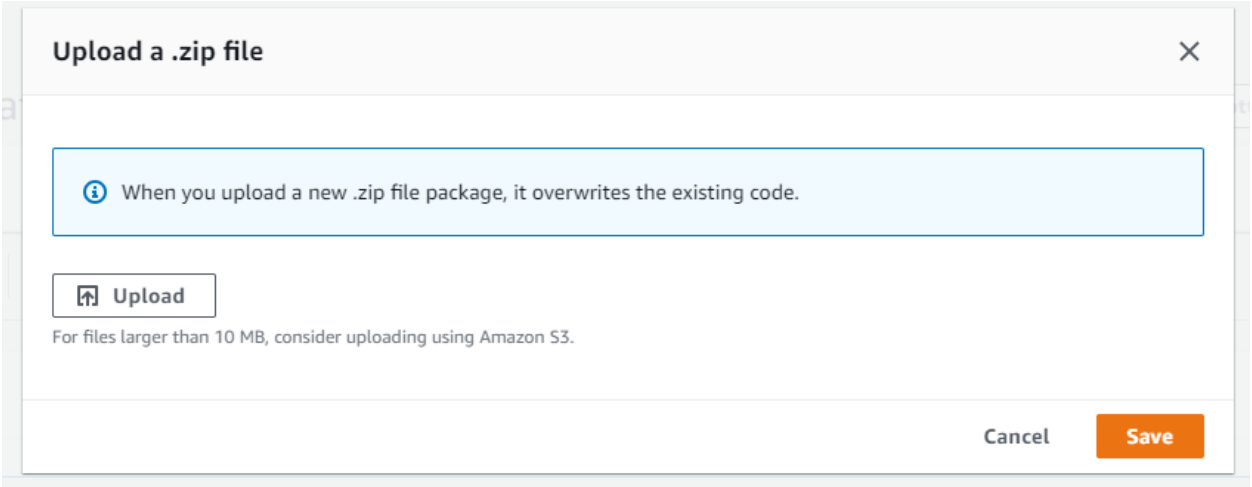
- ✓ Since we have successfully created lambda function. By default it will be empty function and now have to upload the code from our repository.
- ✓ Navigate to functions section from Lambda console and search for “`SendSMSFromContactFlowToCaller`” lambda function like mentioned below



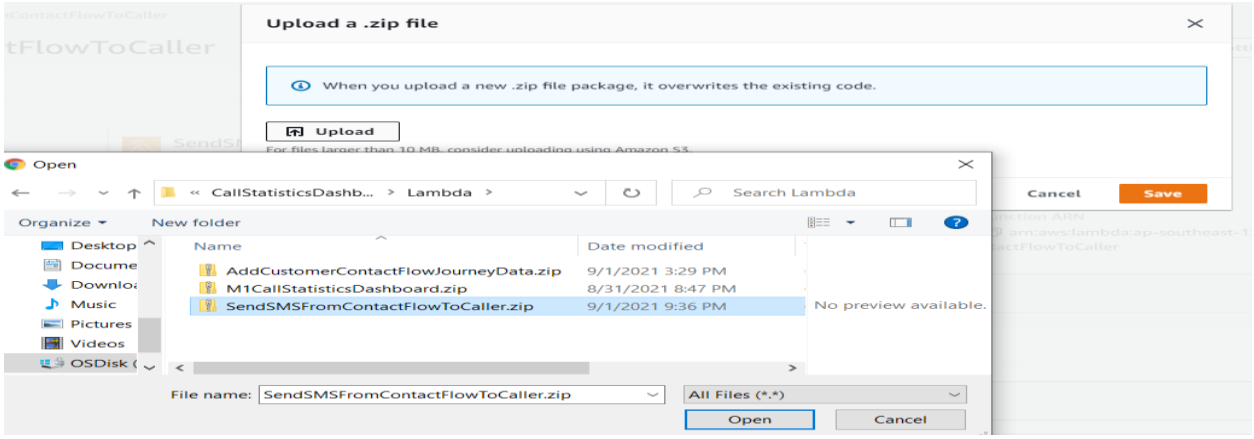
- Click on "`SendSMSFromContactFlowToCaller`" function which will navigate to the page where we can upload code.
- Click ".Zip file" from uploadfile option under the Code tab like mentioned below:



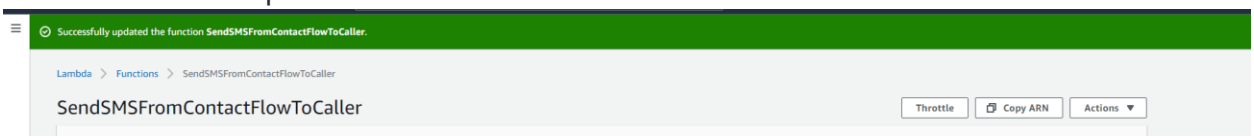
- It will open page where you can upload lambda source code.



- 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:

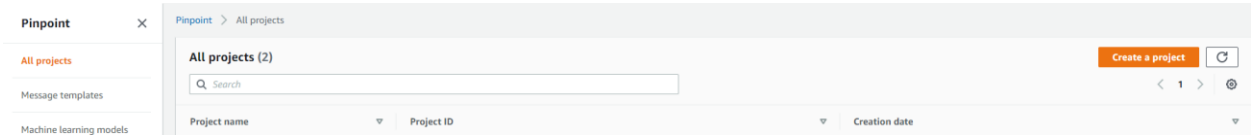


- ✓ After successfully loaded the zip file, we will get success notification. Please wait for some time to upload the function as it will take time.

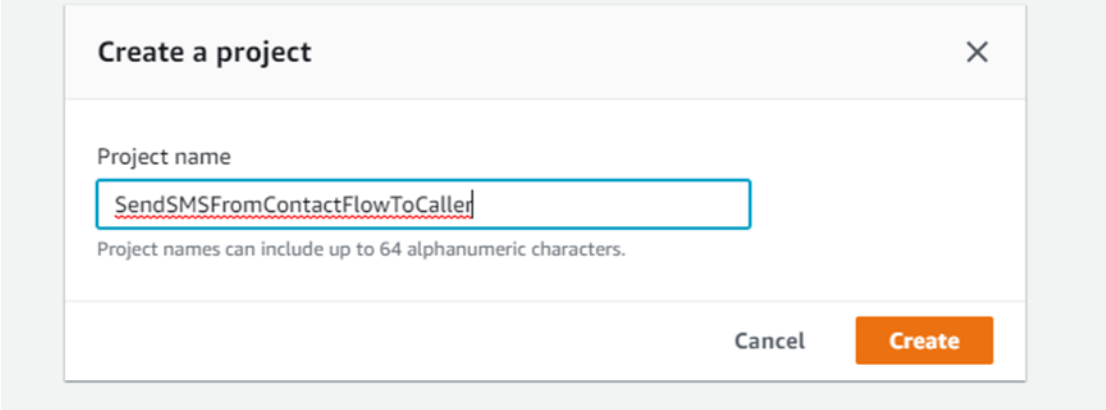


Create SMS Project in Pinpoint Service:

- ✓ 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

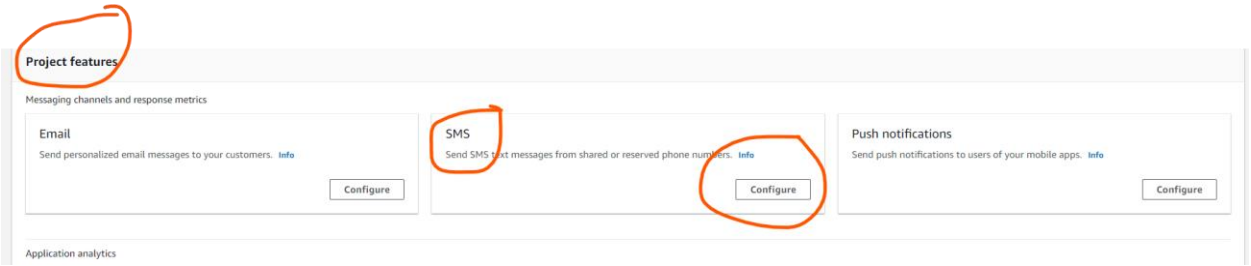


- ✓ Provide project name as "`SendSMSFromContactFlowToCaller`" then click on "Create" button

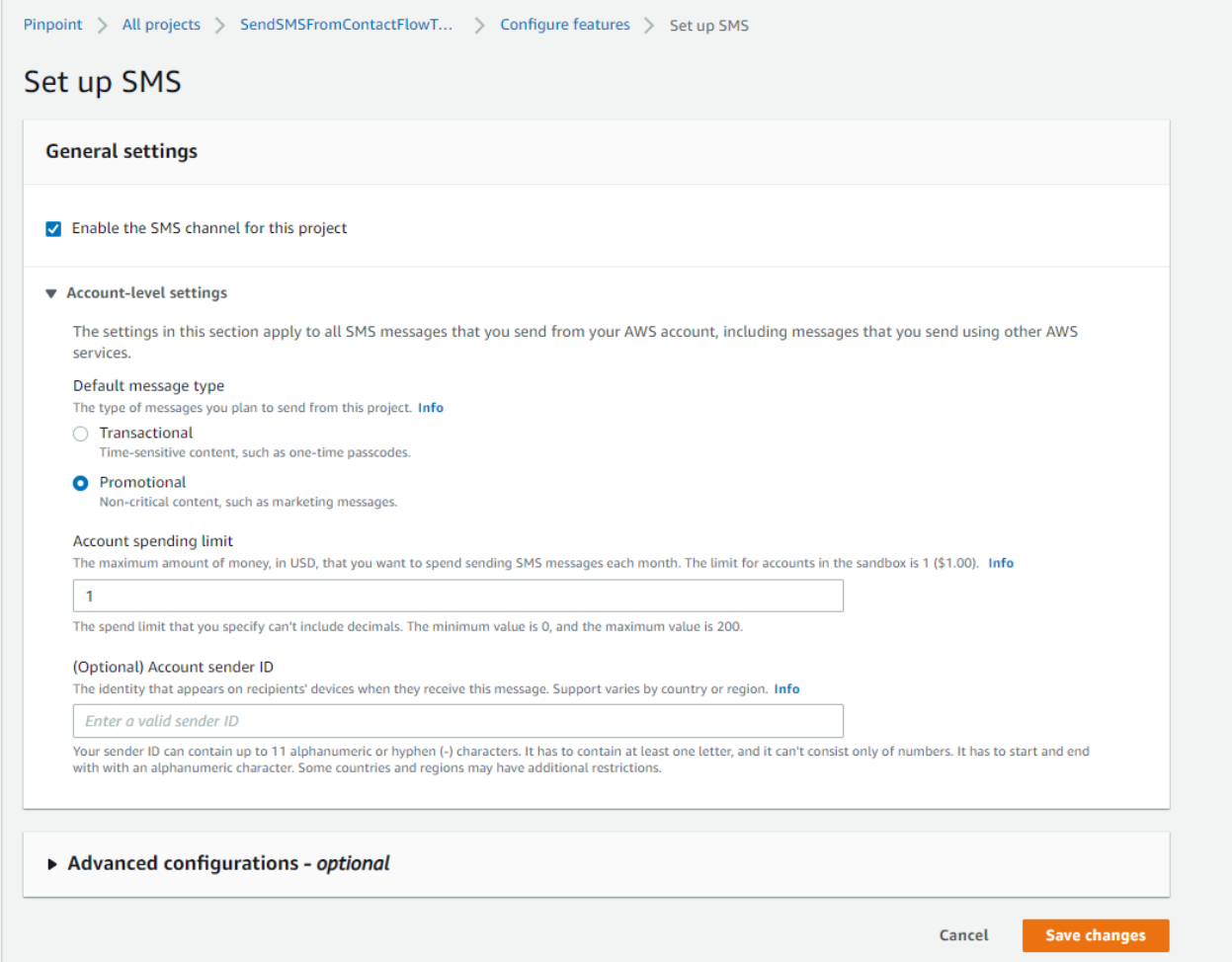


✓ Upon successfully creation of project, you will get notification like mentioned below

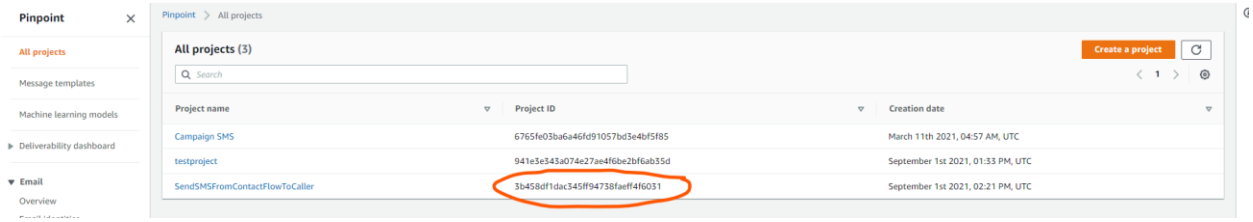
✓ Under project features -> SMS -> Click configure button



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

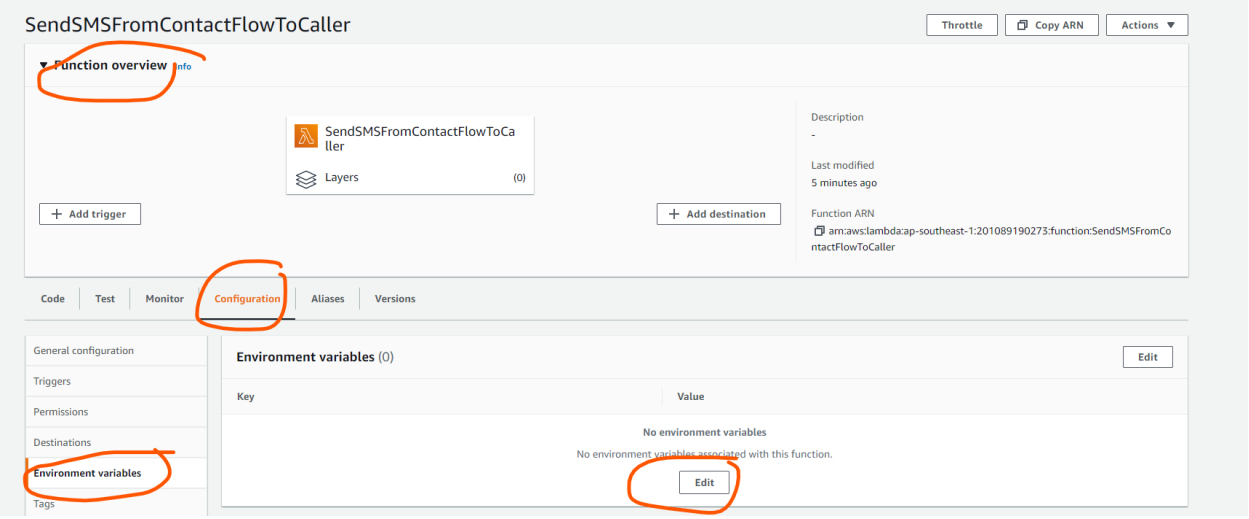


✓ Navigate to projects and copy the Project ID which we are going to use in lambda function later in this section

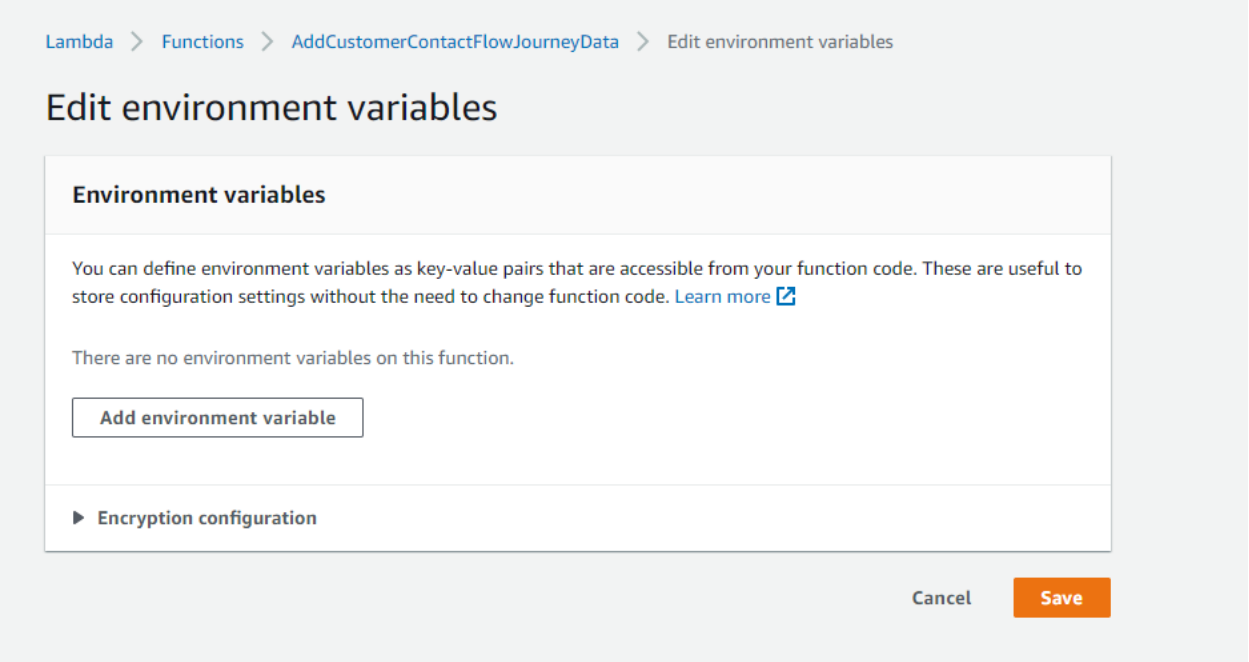


Configuration section:

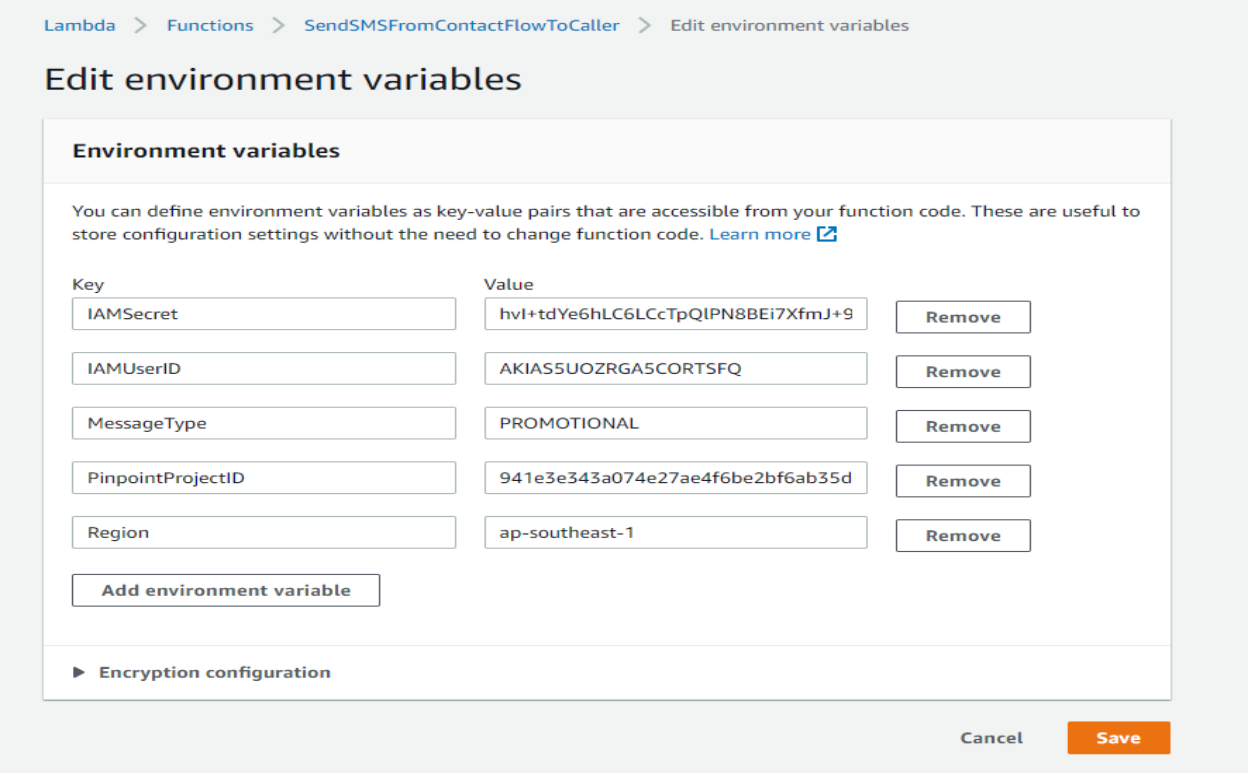
- Navigate to function, then click on Configuration tab and select “Environment Variables” like mentioned below:



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

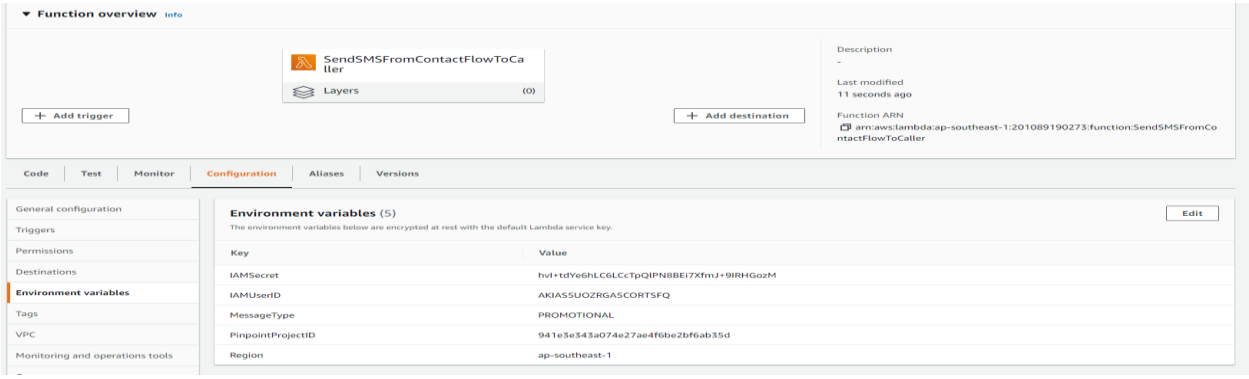


- Click on “Add environment variable” button to add the variables and values as key value pair like mentioned below



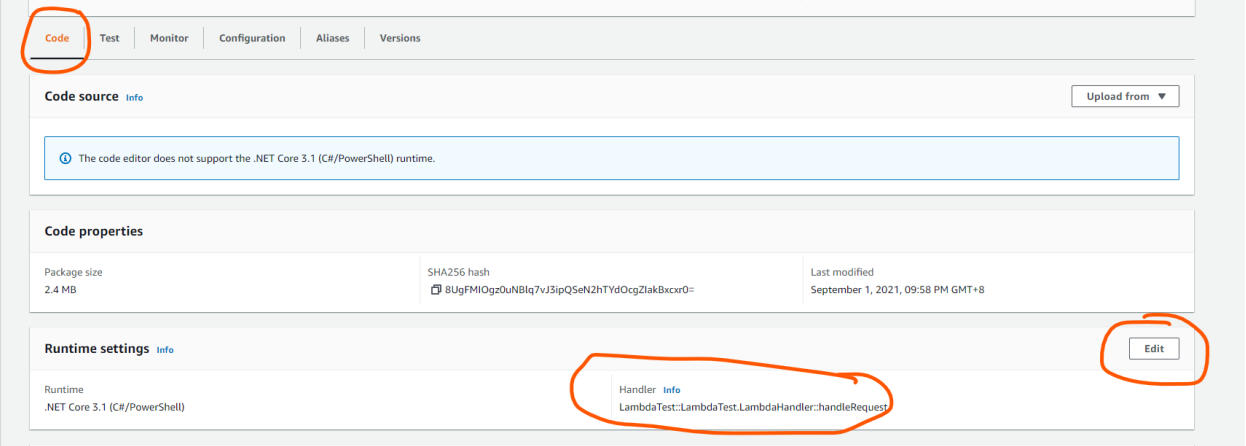
Add your IAM UserID & Secret key properly. PinpointProjectID – enter the value which you have saved earlier when we were creating pinpoint project.

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



Change the function handler

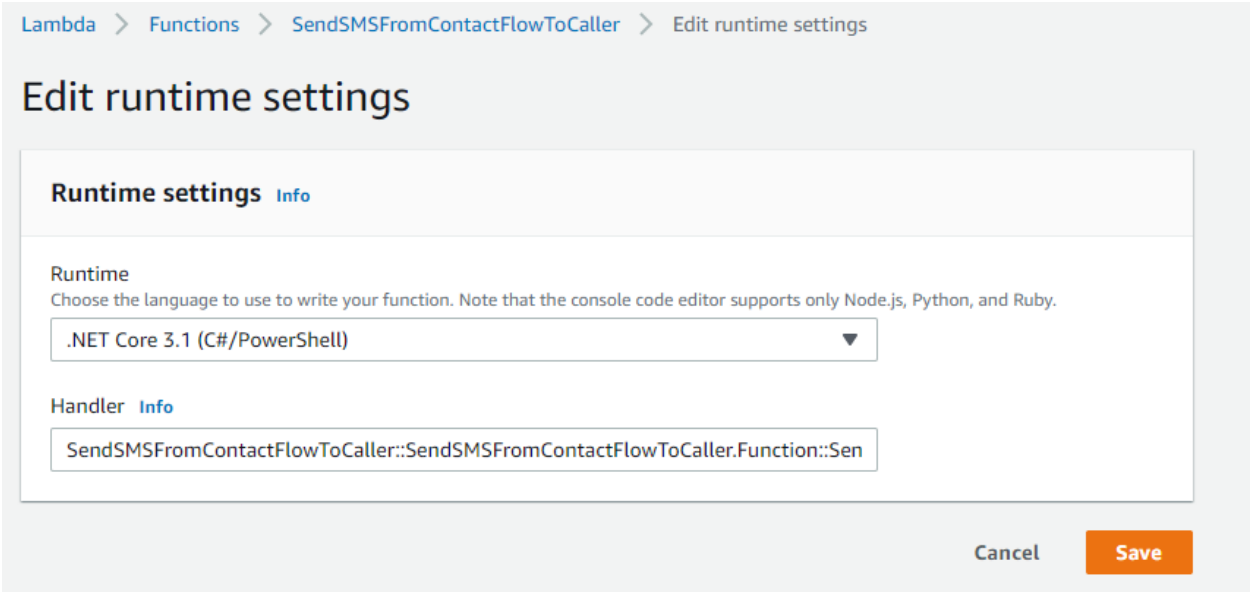
- Go to code tab and click on Edit button under Run time settings section like mentioned below



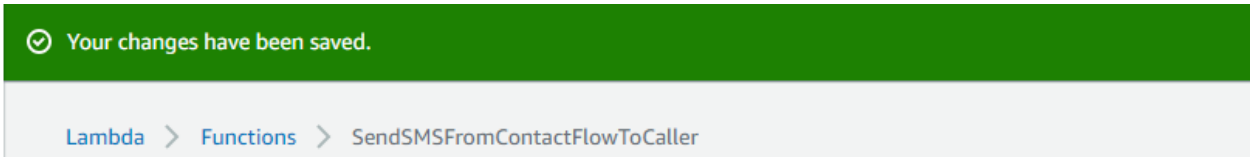
- Change the Handler from

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

  “`SendSMSFromContactFlowToCaller::SendSMSFromContactFlowToCaller.Function::SendSMSFromContactFlowToCaller`”

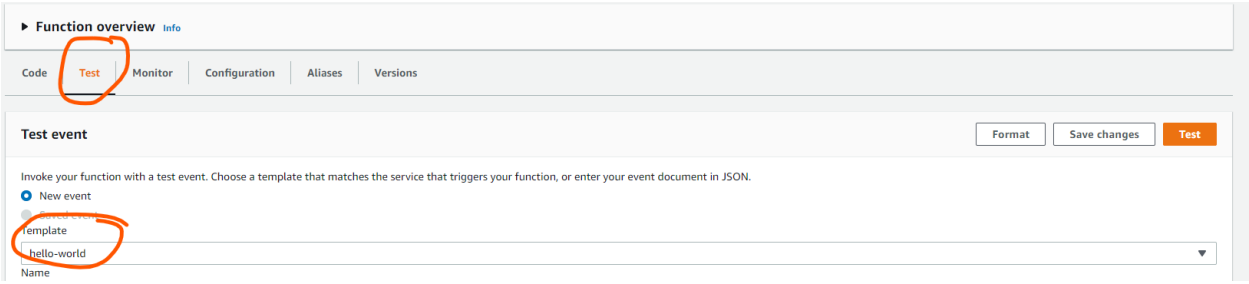


Then click on save button to save the handler settings.



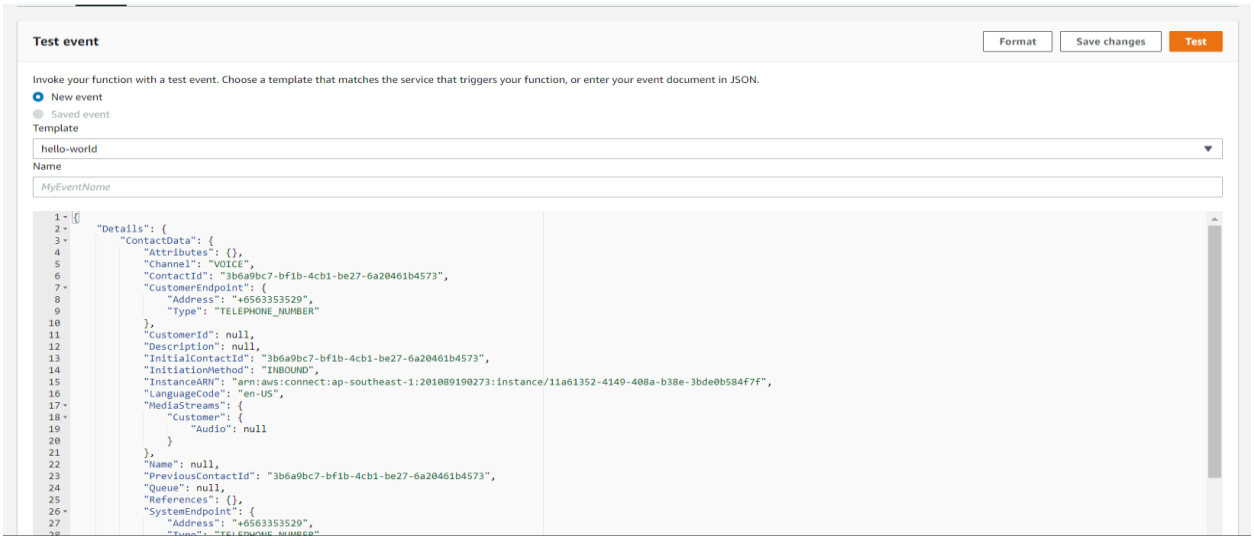
Test the lambda function section:

- Naviagate to Test tab under function overview -> Select template as “Hello-world”



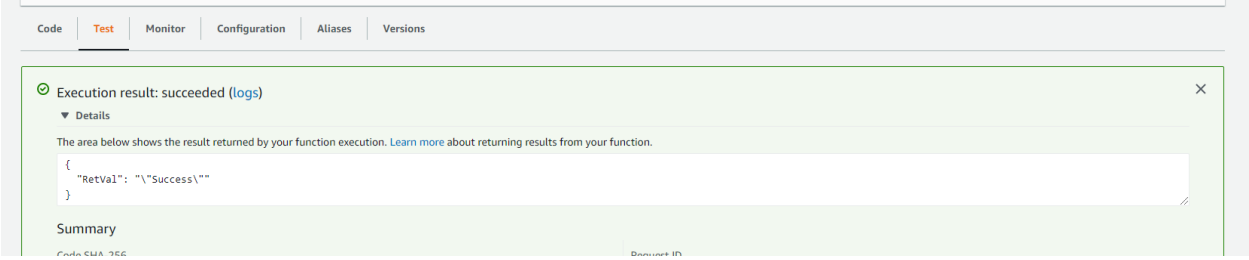
- Copy below json in the json field to test the function. (change your mobile number in two places where highlighted)

```
{ "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" }
```



- Click on “Test” button to test the function

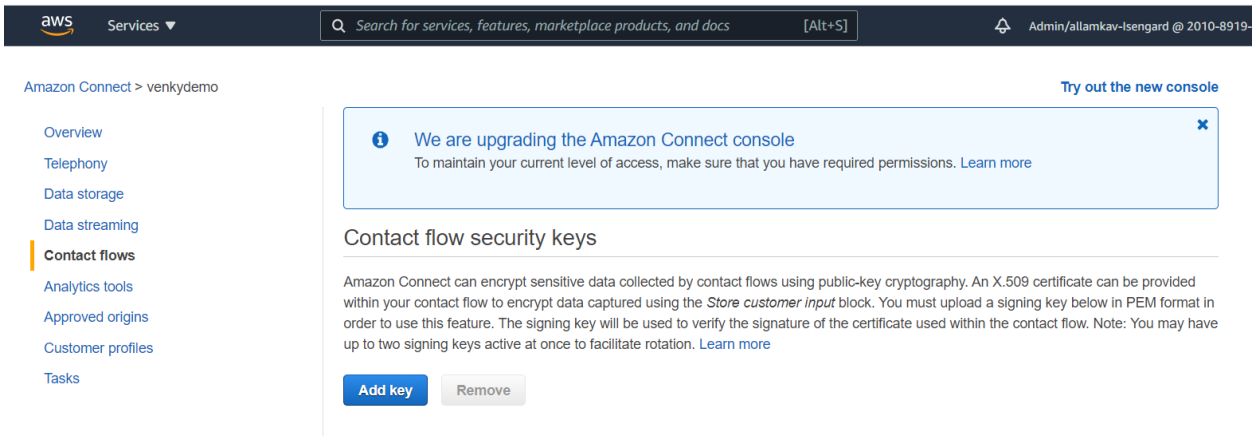
If the function gets success then you will be able to see below success message.



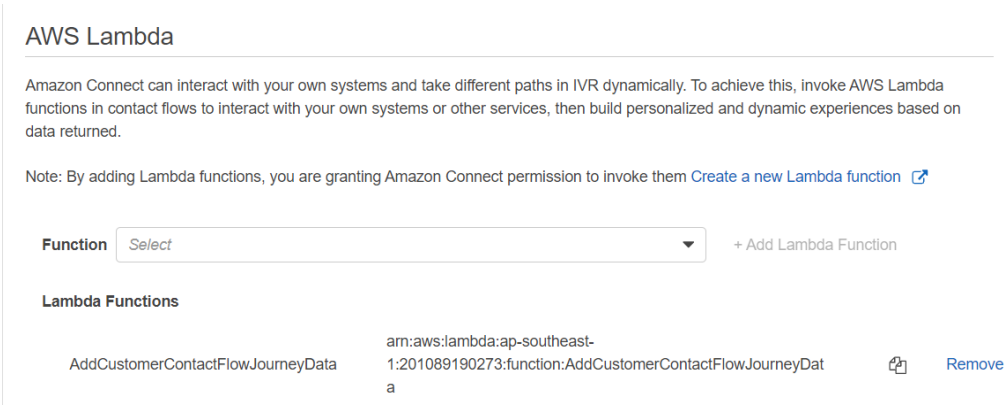
You should be able to receive SMS.

Configure lambda to amazon connect instance

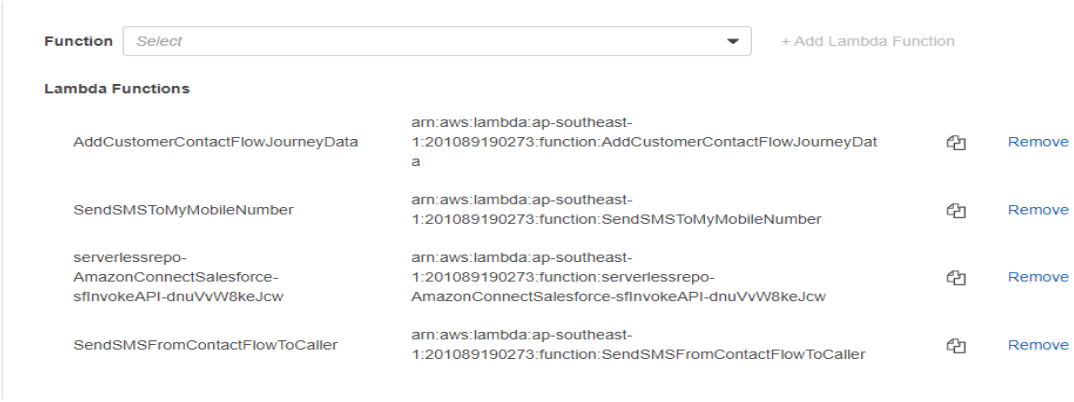
- ✓ Go to Amazon Connect
- ✓ open the instance which you are using for contact flow
- ✓ Navigate to Contact flows from the left menu like mentioned below:



- ✓ 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.

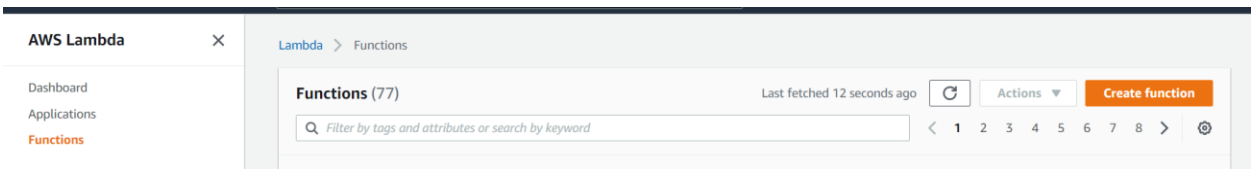


- ✓ Follow and repeat the same above instructions to add "SendSMSFromContactFlowToCaller" lambda to connect instance.

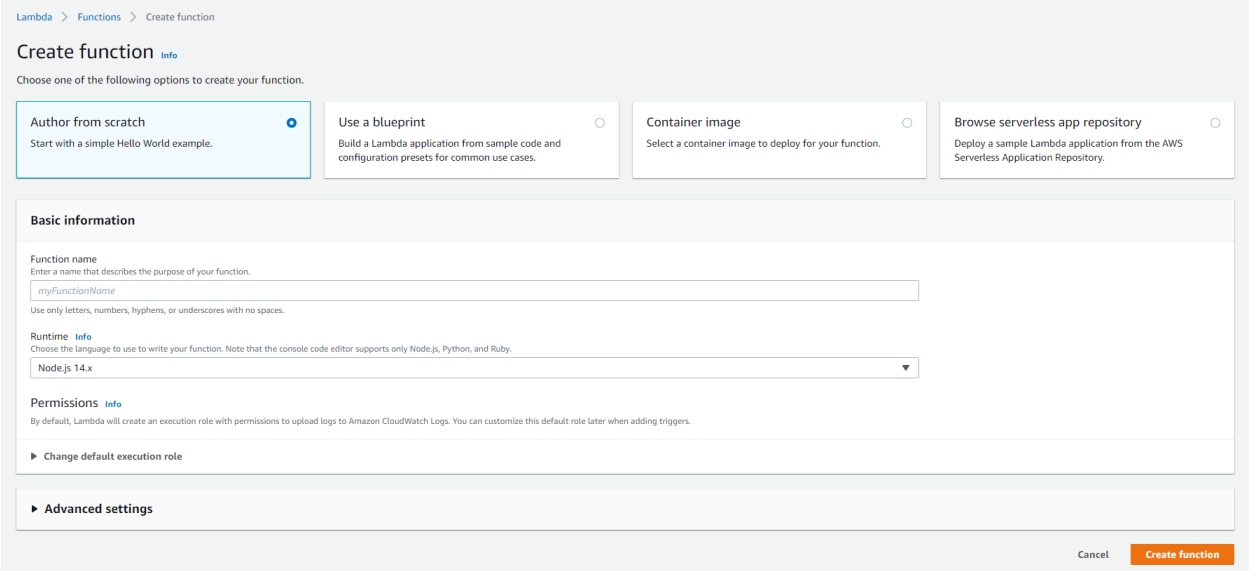


Upload "ConnectCallStatisticsDashboard" Lambda : This lambda will be used to fetch the data from DynamoDB for displaying in the UI dashboard.

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

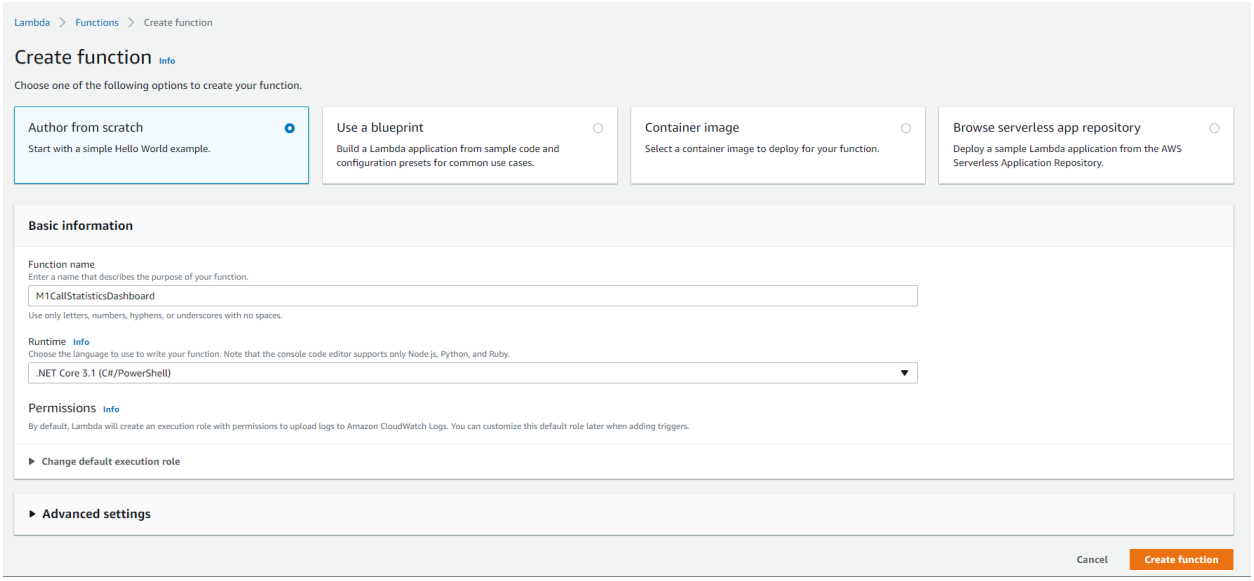


✓ Click on “Create Function” button at the right top corner to create function which will navigate to the page where you can provide details to create lambda function like mentioned below:

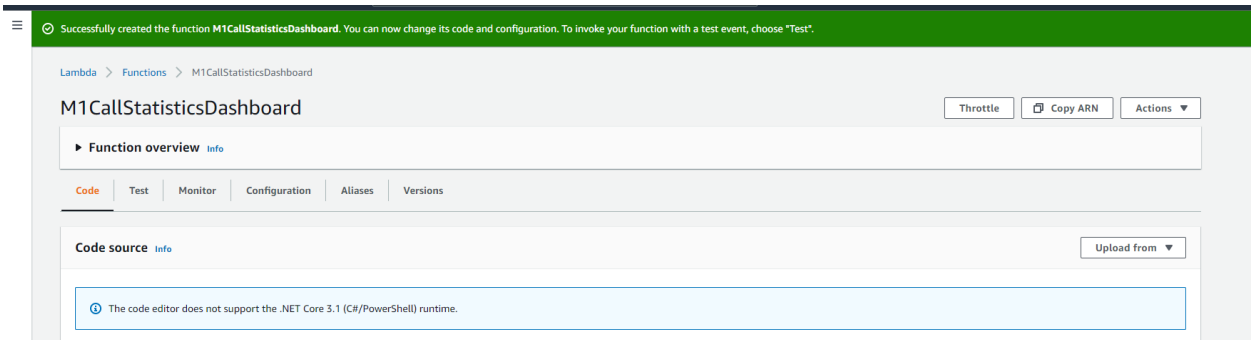


✓ Choose “Author from scratch”
✓ Provide function name as “`ConnectCallStatisticsDashboard`”.
✓ Select Runtime as “.NET Core 3.1 (C#/PowerShell)” from the given list.

Final options should be like mentioned below:

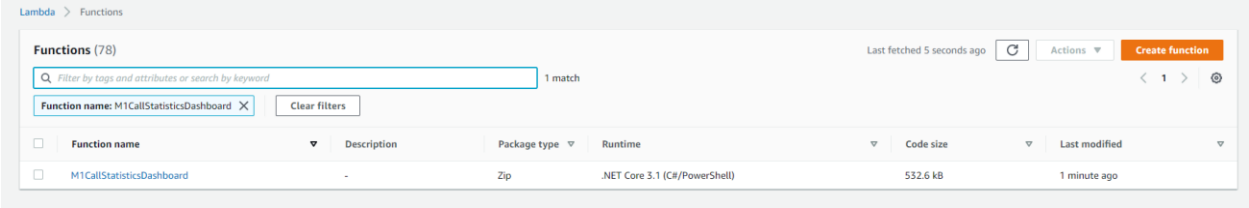


- Leave other values as default and click on “Create Function” from the bottom right corner to proceed.
- Once function is successfully created then you will be able to see below notification

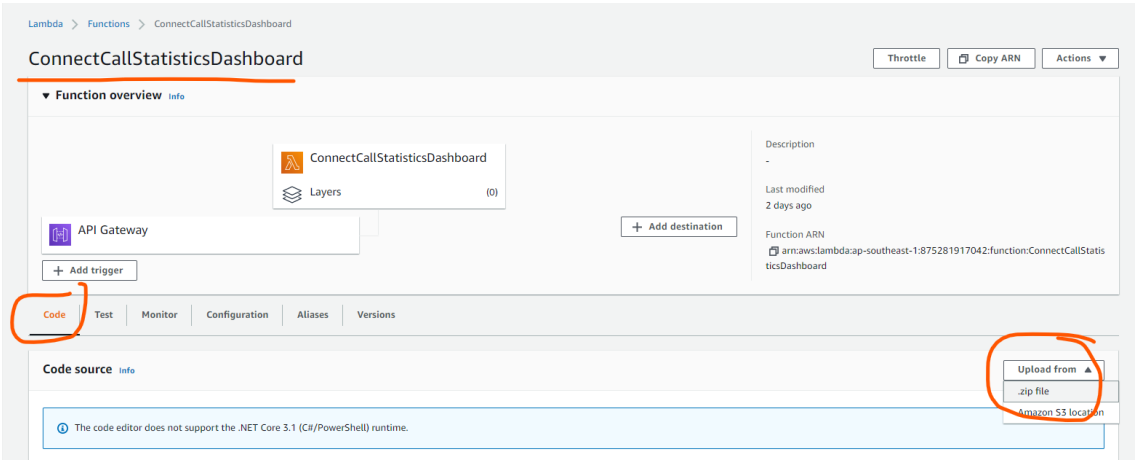


✓ Since we have successfully created lambda function. By default it will be empty function and now have to upload the code from our repository.

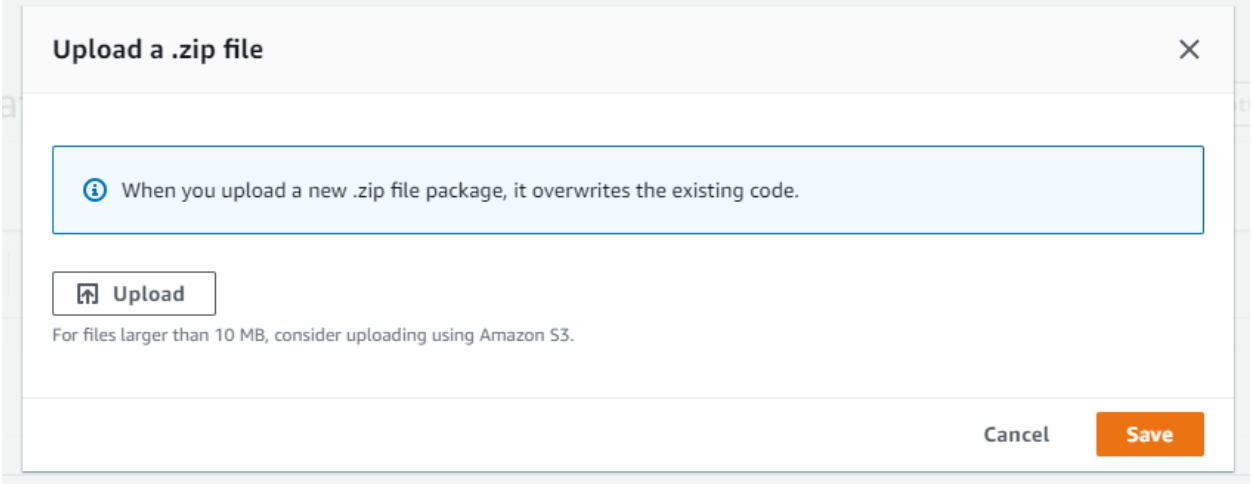
✓ Navigate to functions section from Lambda console and search for “`ConnectCallStatisticsDashboard`” lambda function like mentioned below



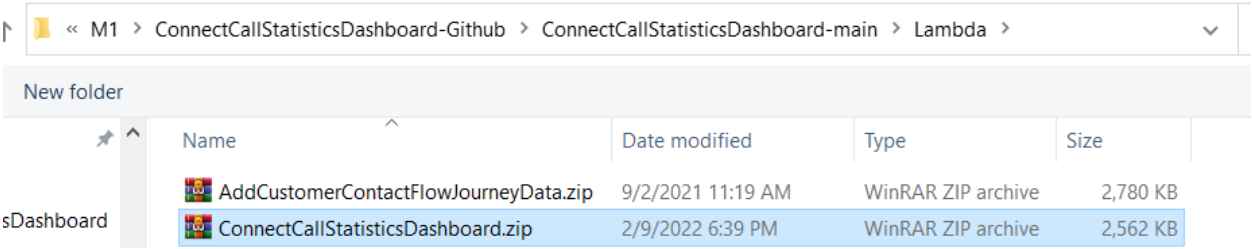
- Click on “`ConnectCallStatisticsDashboard`” function which will navigate to the page where we can upload code.
- Click “.Zip file” from uploadfile option under the Code tab like mentioned below:



- 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:

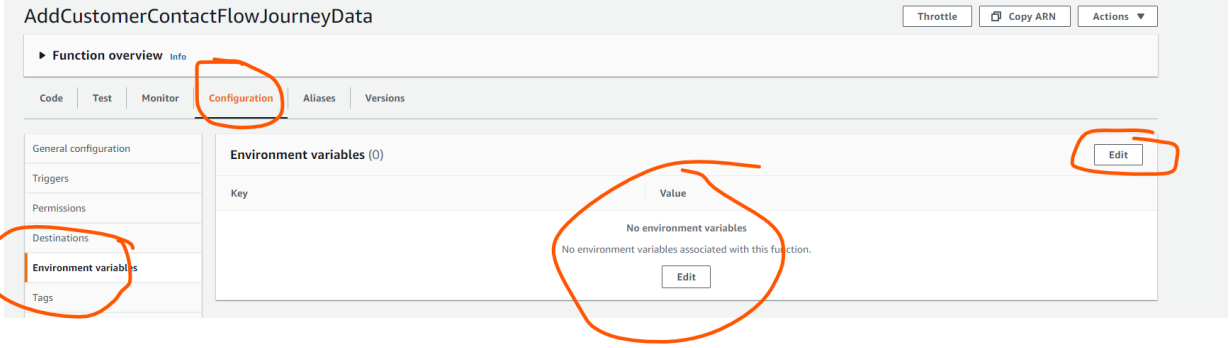


✓ After successfully loaded the zip file, we will get success notification. Please wait for some time to upload the function as it will take time.

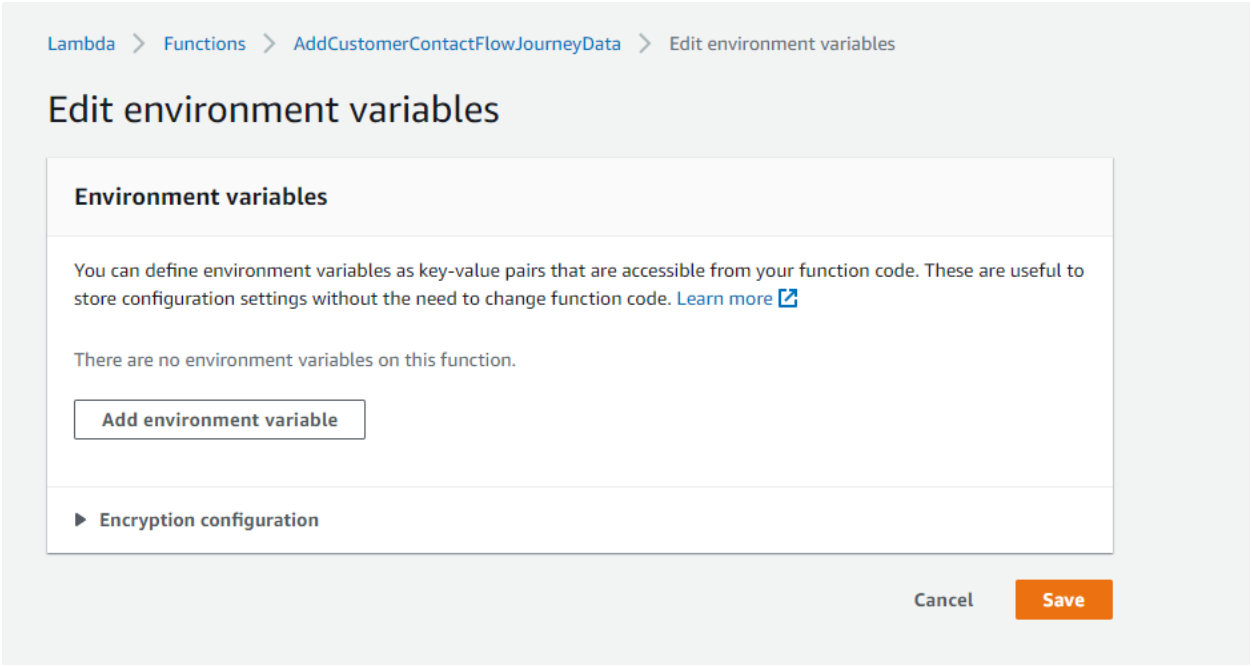
✓

Configuration section:

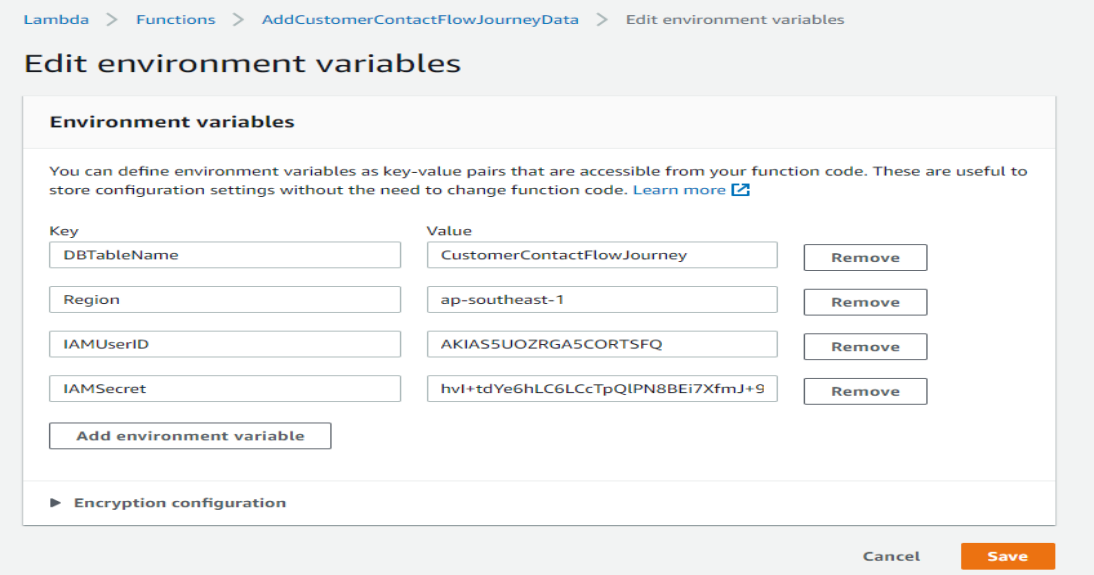
- Navigate to function, then click on Configuration tab and select “Environment Variables” like mentioned below:



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

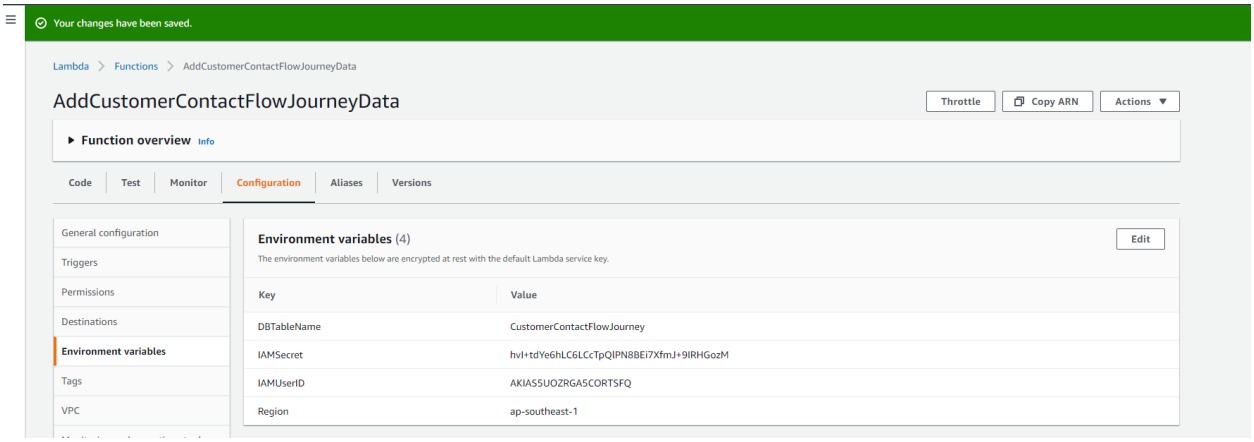


- Click on “Add environment variable” button to add the variables and values as key value pair like mentioned below



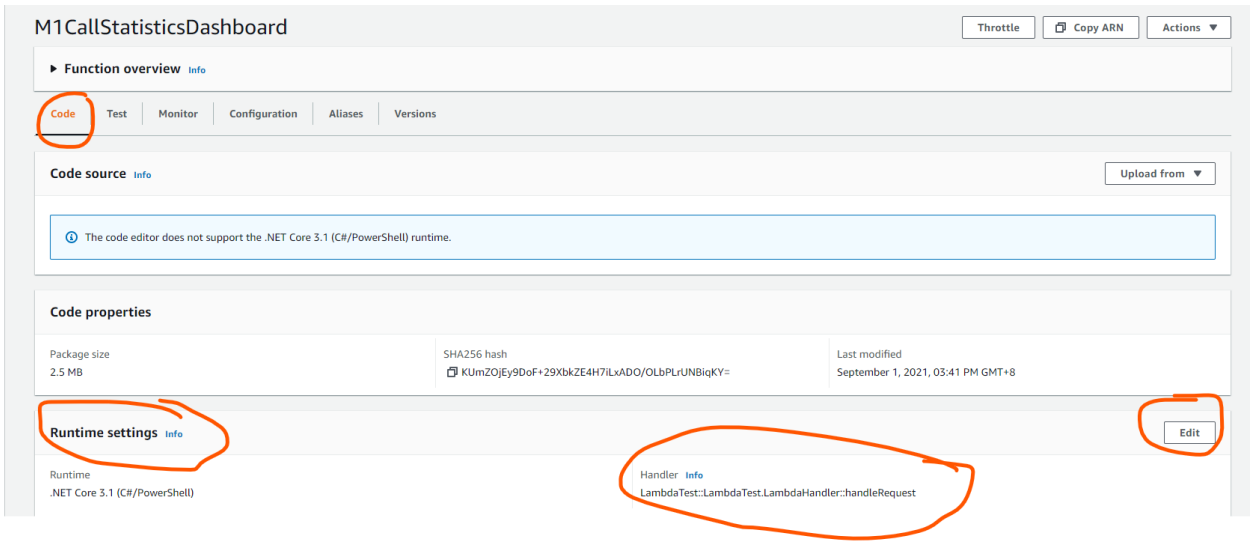
Configure your IAM UserID & Secret key properly.

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



Change Function handler section

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

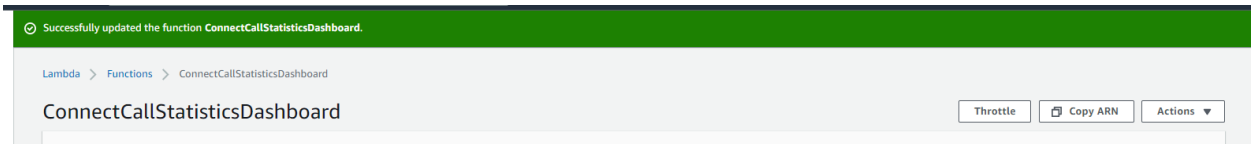


- Change the Handler from

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

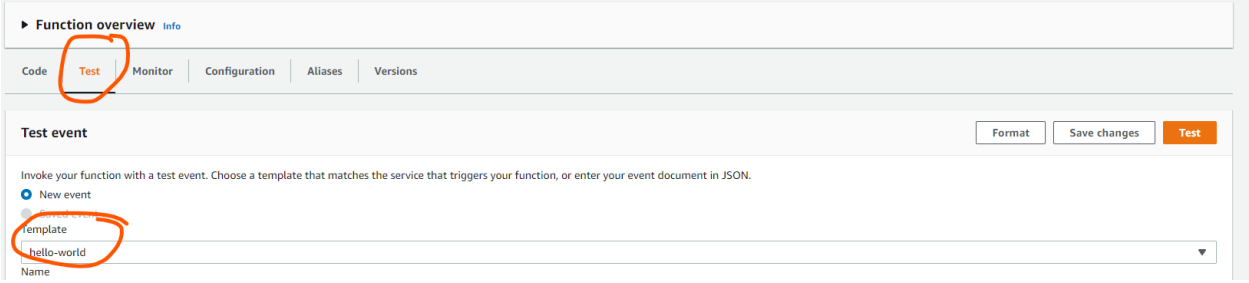
  “ConnectCallStatisticsDashboard::ConnectCallStatisticsDashboard.Function::GetCallStatsData”

Then click on save button to save the handler settings.

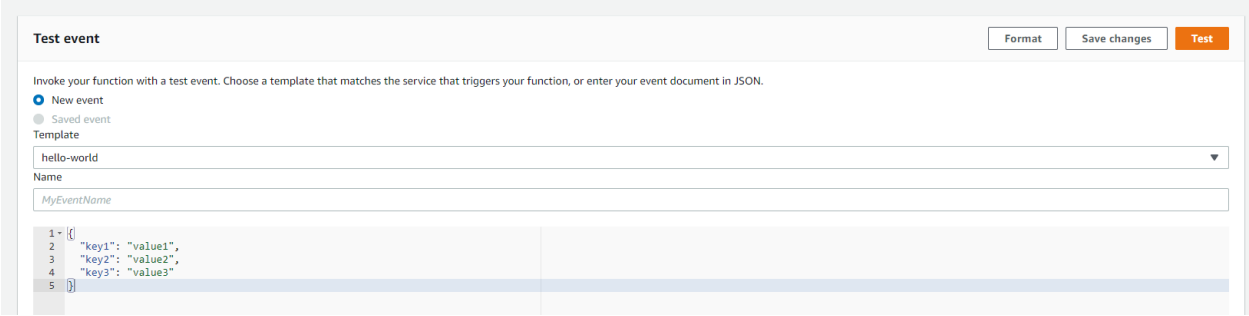


Test the lambda function section:

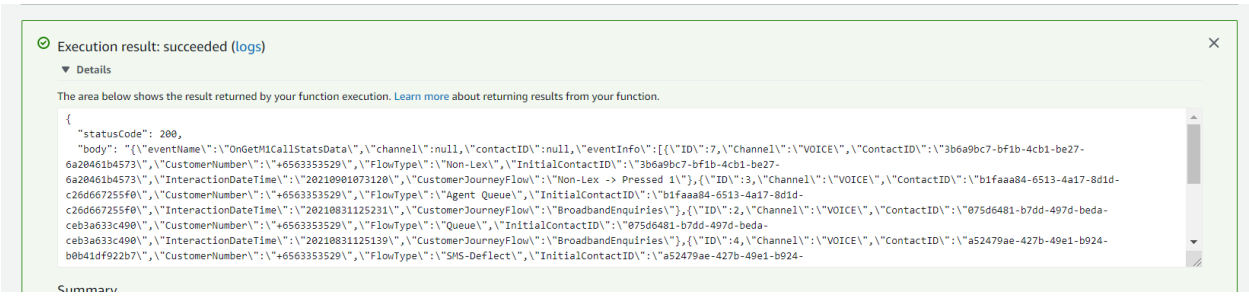
- Naviagate to Test tab under function overview -> Select template as “Hello-world”



- No need to change the json

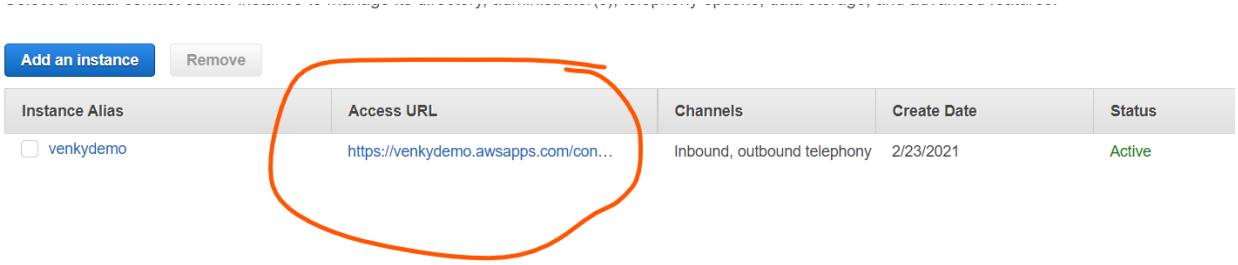


- Click on “Test” button to test the function
  If the function gets success then you will be able to see below success message.



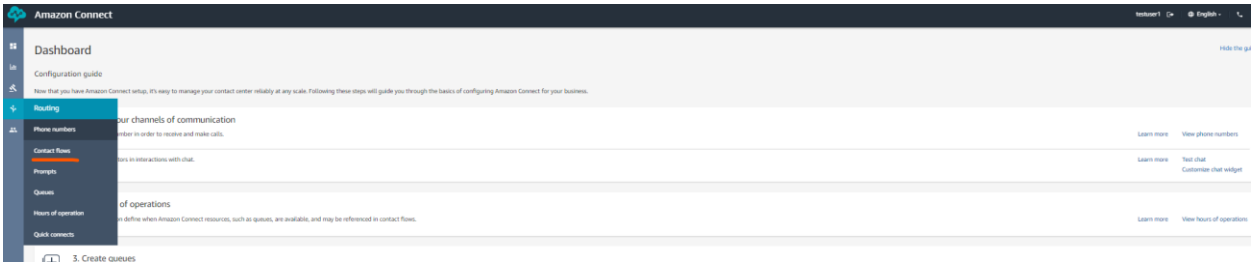
Steps to import contact flow to Amazon Connect instance.

1. Go to the Amazon Connect console, open the instance URL you are using like mentioned below:

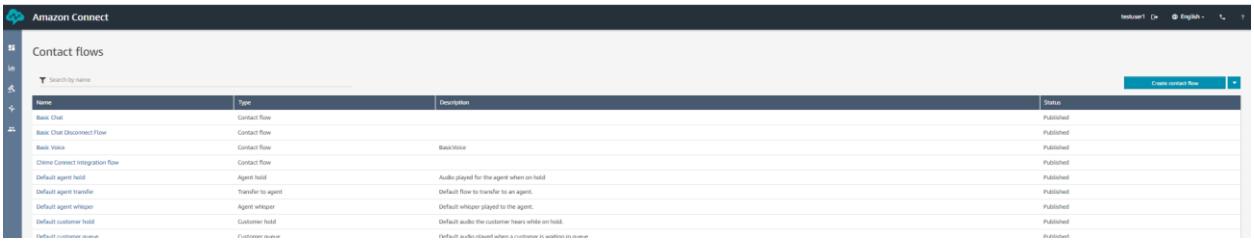


2. Login to Amazon connect dashboard with the user which you have created while setting up connect instance.

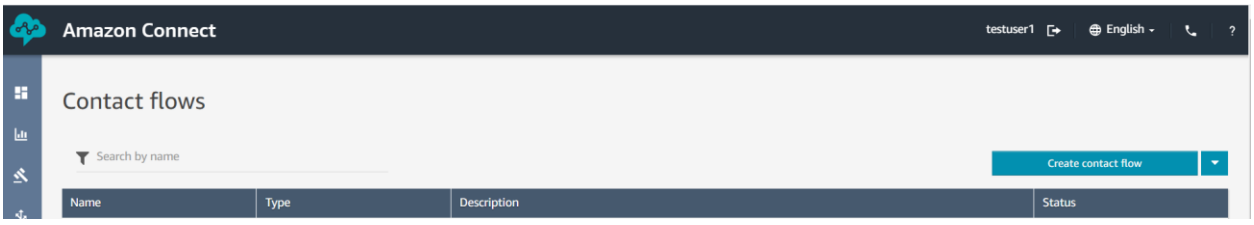
3. Navigate to Routing -> Contact Flows from the left menu.



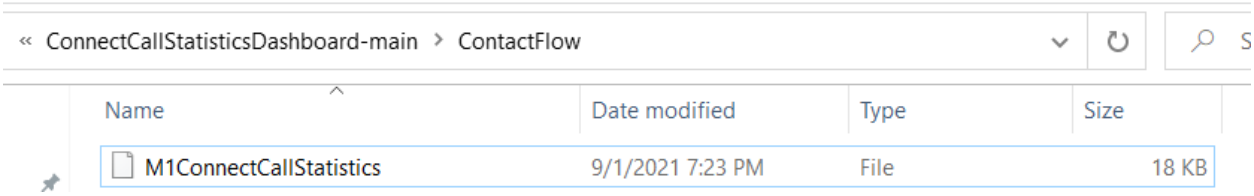
You will be able to see the existing default contact flows which comes along with connect instance.



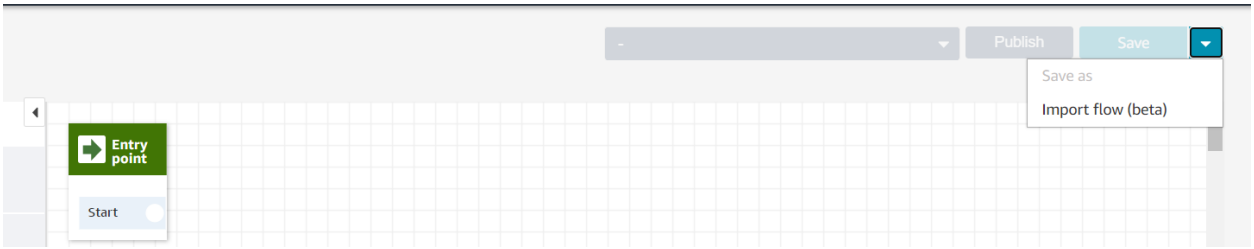
4. Click on the “Create contact flow” button which appears on the top-right corner.



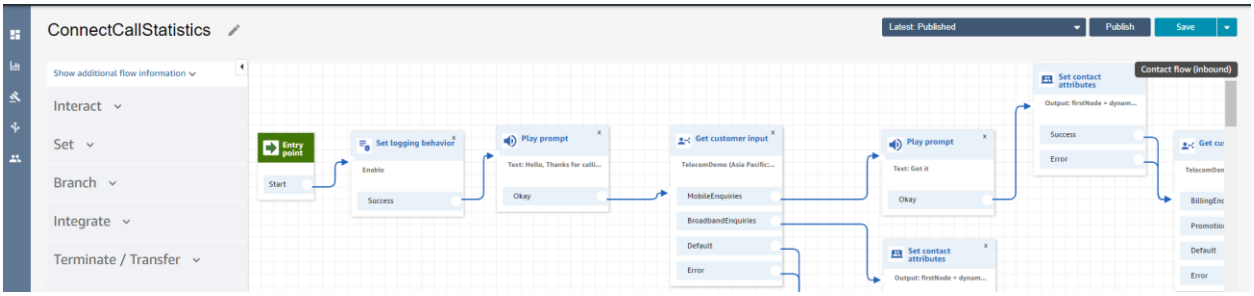
5. Open the unzipped folder and navigate to ContactFlow folder to see contactFlow file



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

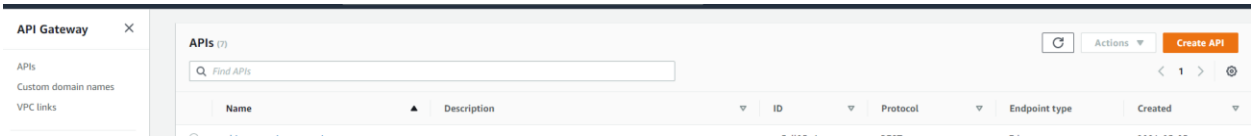


7. Click on Import flow and choose the “ConnectCallStatistics” call flow from ContactFlow folder. Then call flow will imported. Click on “Save” to save the call flow & click on “Publish” to publish the changes for your testing.
8.

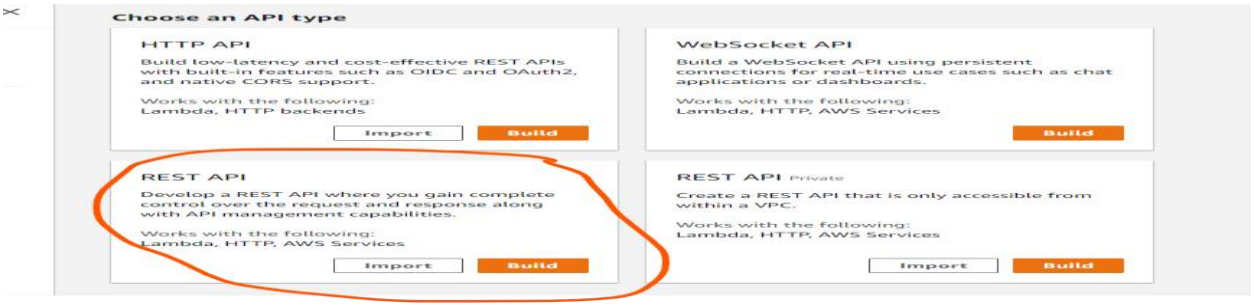


Creating REST API in APIGateway to access lambda from public

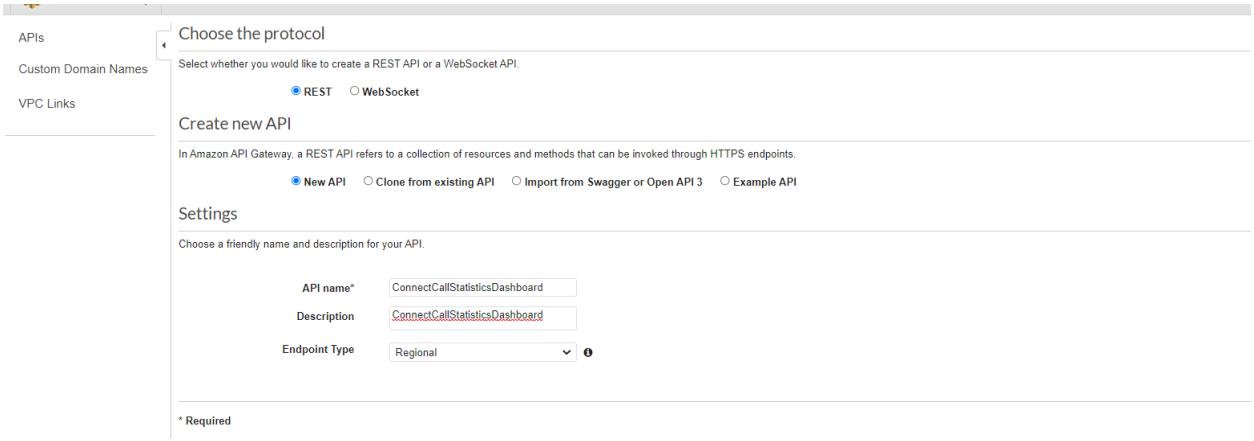
- Navigate to [Api gateway] in console
- Click on “Create API” buttton which appears in the top right corner



- Choose API type as REST API



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



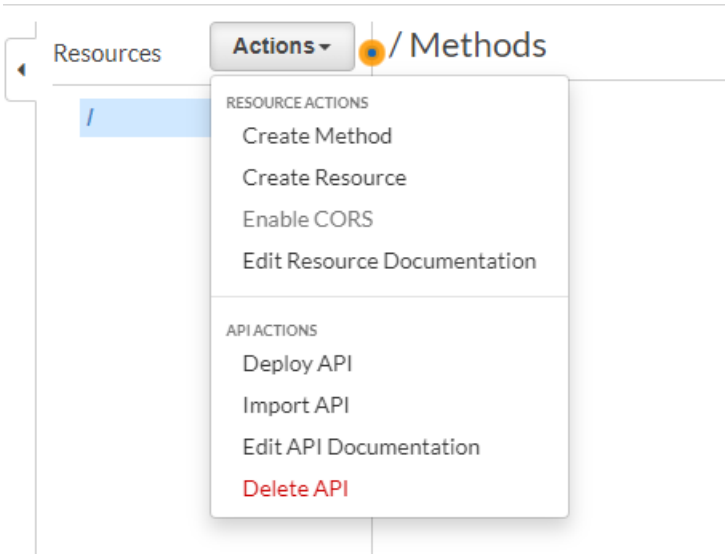
Choose REST API

Under settings

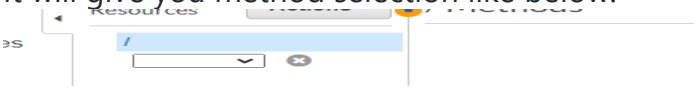
- ✓ API Name : ConnectCallStatisticsDashboard
- ✓ Description : ConnectCallStatisticsDashboard
- ✓ Endpoint Type : Regional

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

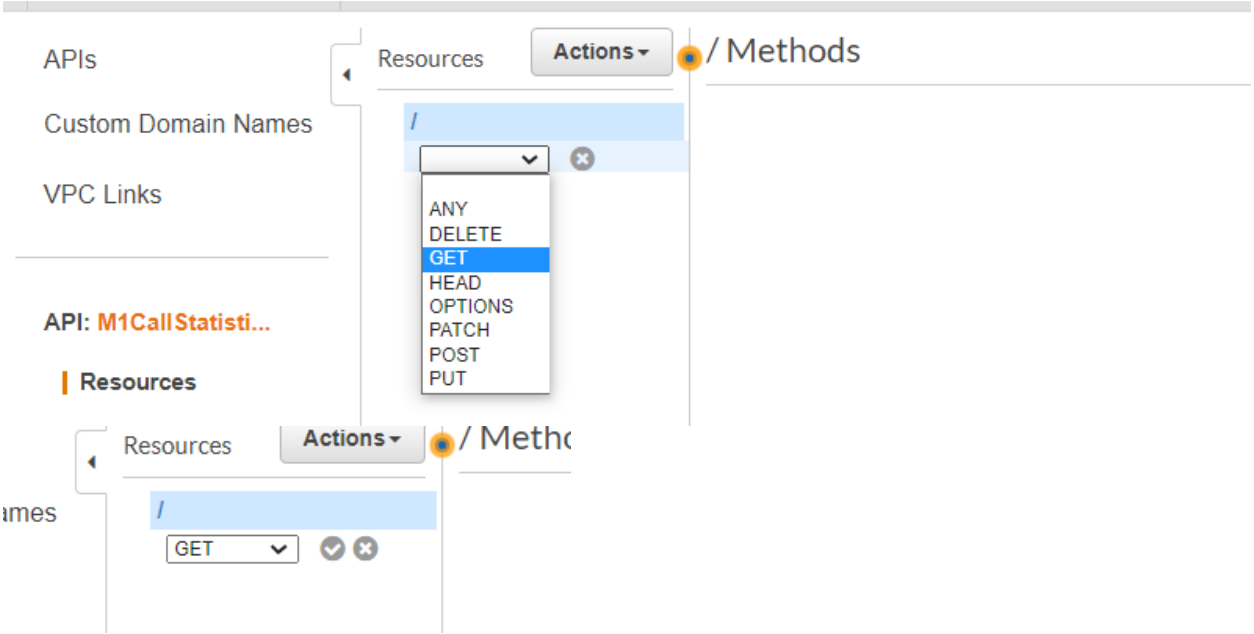
- Now we need to create Method. Click on Actions -> Create Method



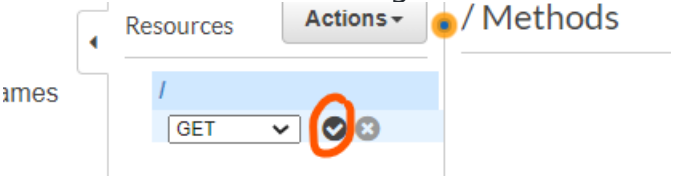
- It will give you method selection like below:



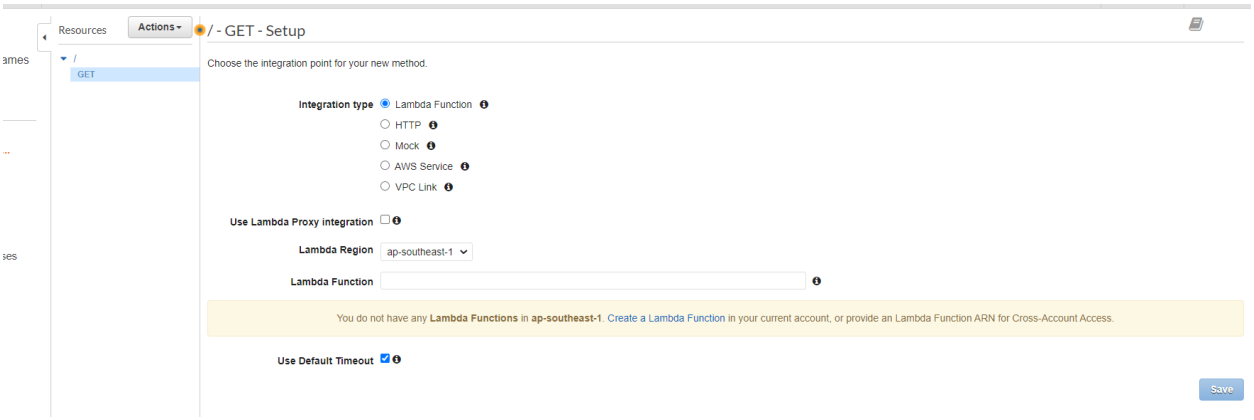
- Select “GET” from the list



- Click “tick” to save the changes

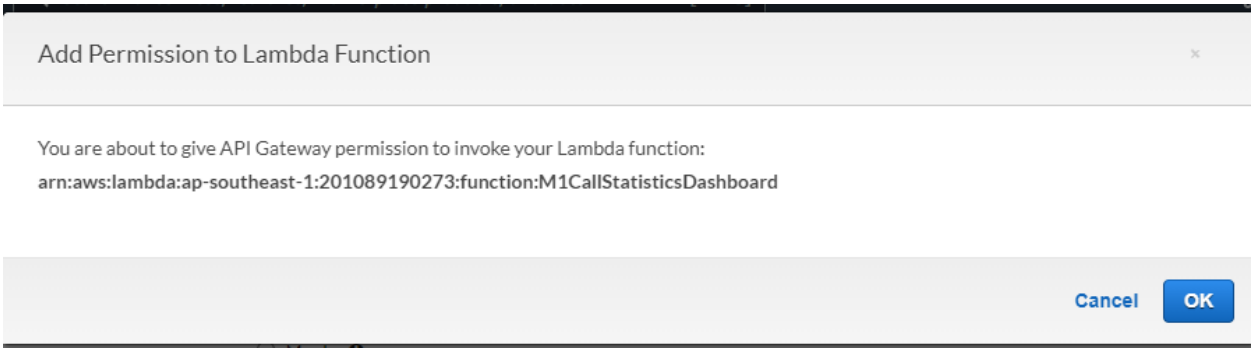


- Once method is created, you will be able to see below page

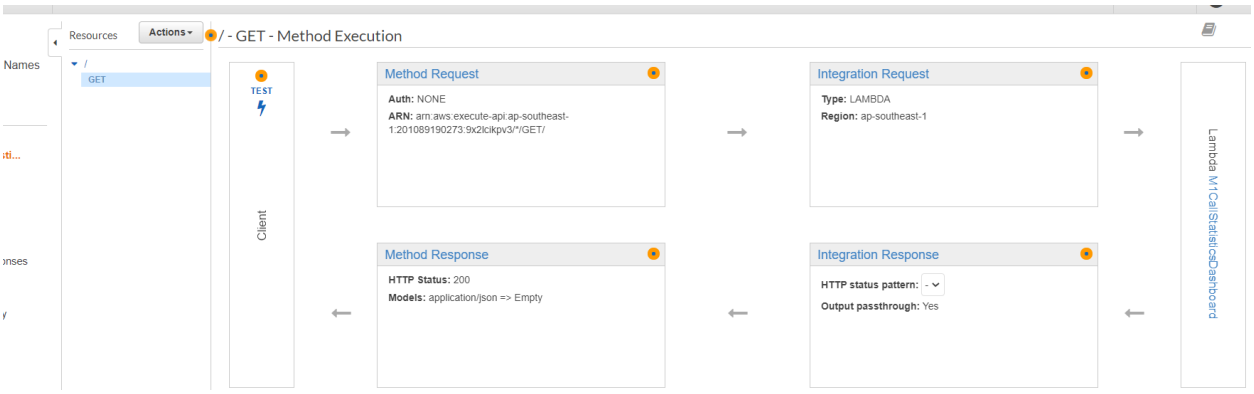


- Lambda integration with REST API
    - Choose Integration type as “Lambda Function”
    - Use lambda proxy integration – do not uncheck and leave blank
    - Lambda region : please choose Singapore region if you are located in Singapore, otherwise choose the region where lambda function is located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

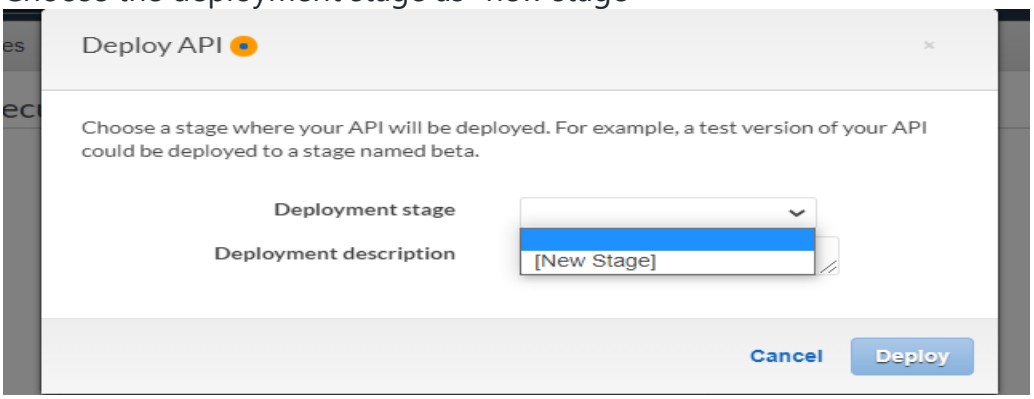
    You will get confirmation to provide permissions, please click ok.



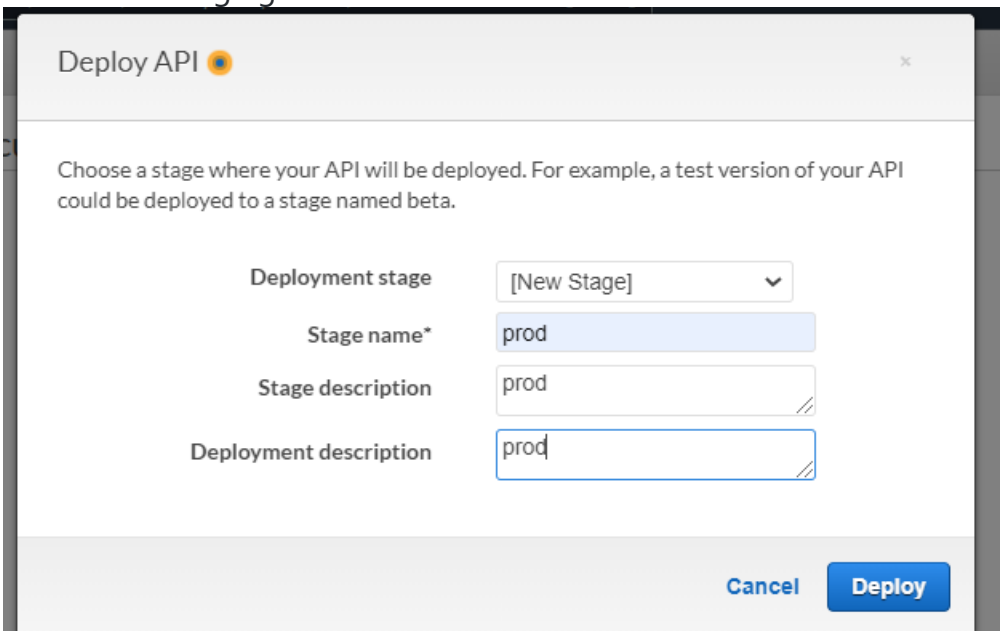
- Once done, you will navigate to below page



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

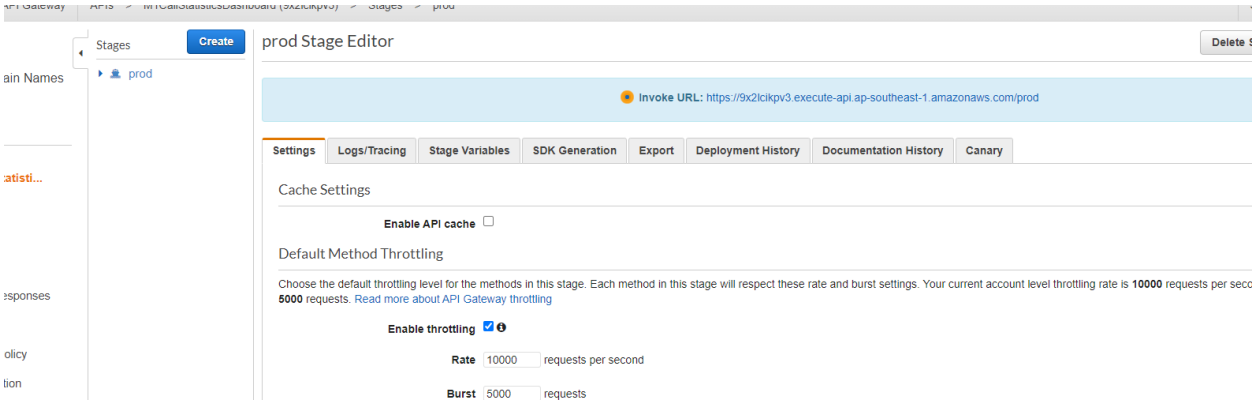


  - Provide the staging details like mentioned below:

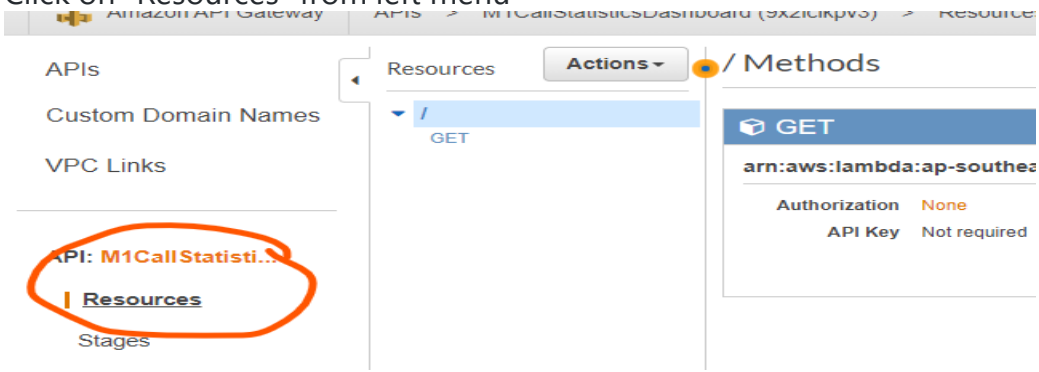


  - Then click on “Deploy” button to deploy the API

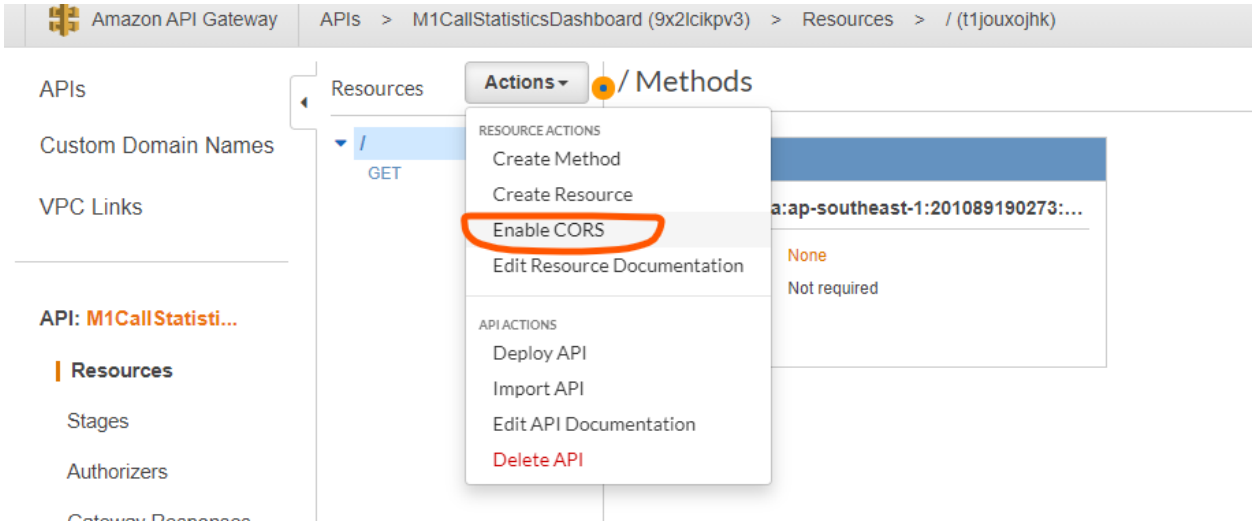
- Then you will be navigated to below page where API URL will be created.



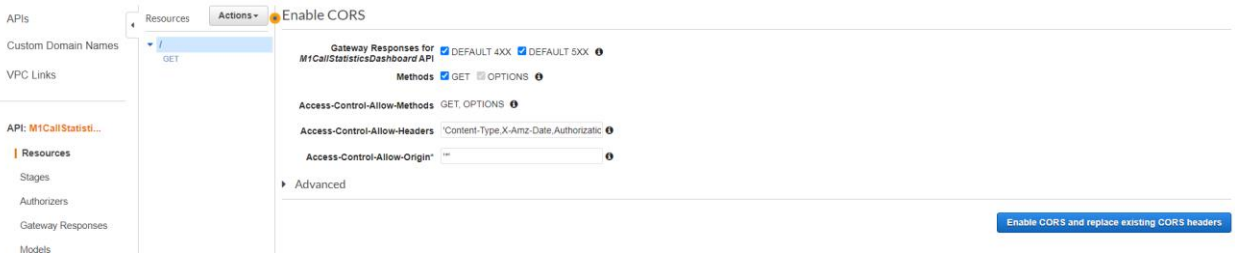
- We need to enable CORS
  - Click on "Resources" from left menu



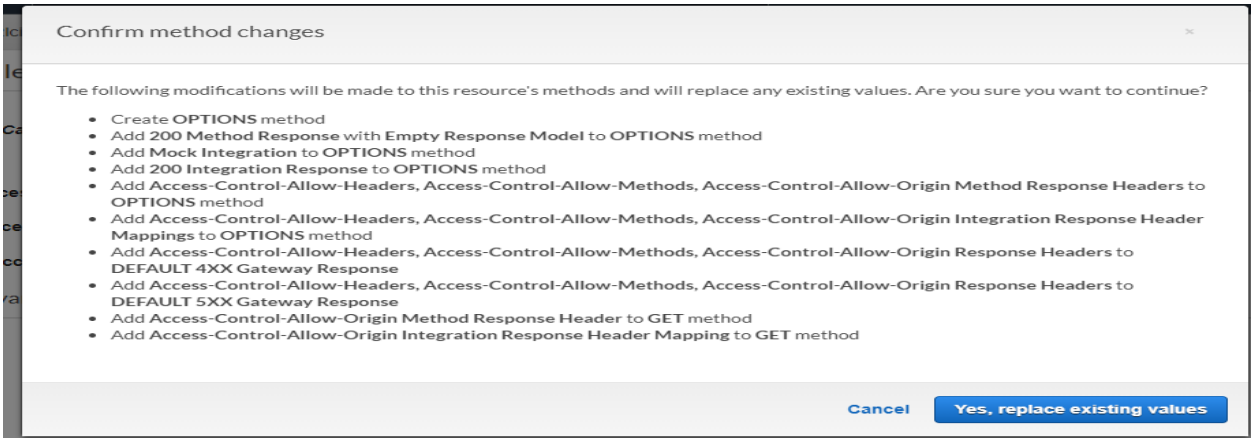
  - Click Actions -> Enable CORS



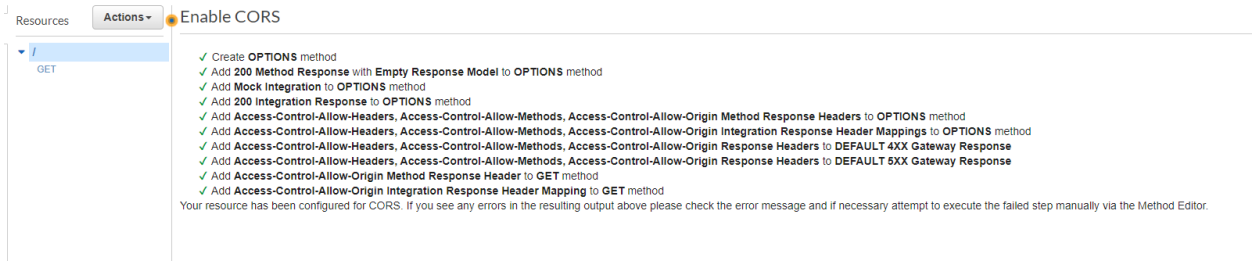
  - Select "Default 4xx" and "Default 5xx" from the configuration like mentioned below and click on "Enable CORS and replace existing CORS Headers" button



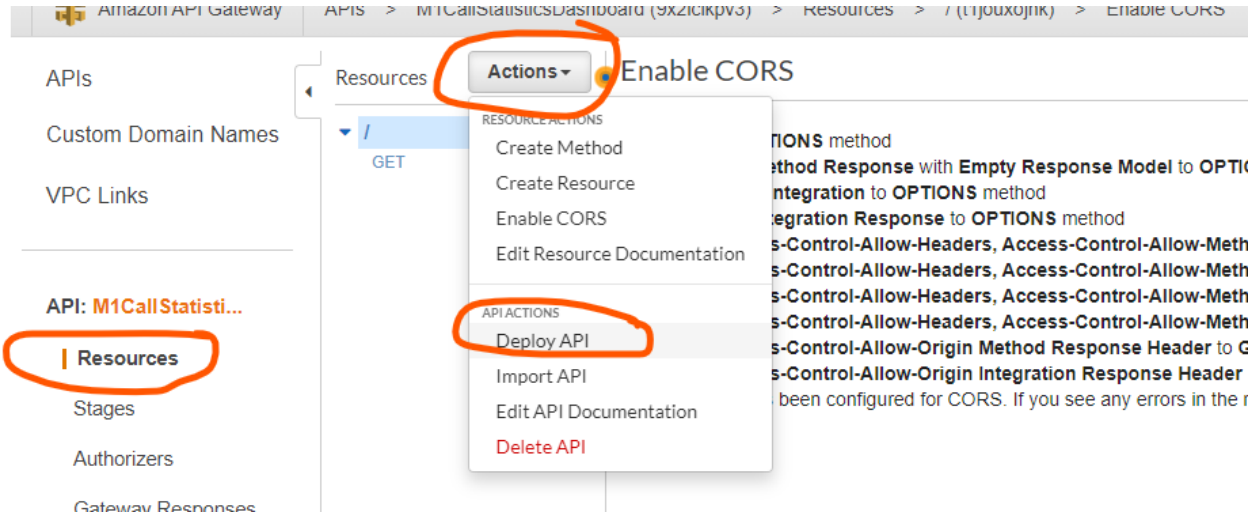
- Next it will ask you to confirm with the changes



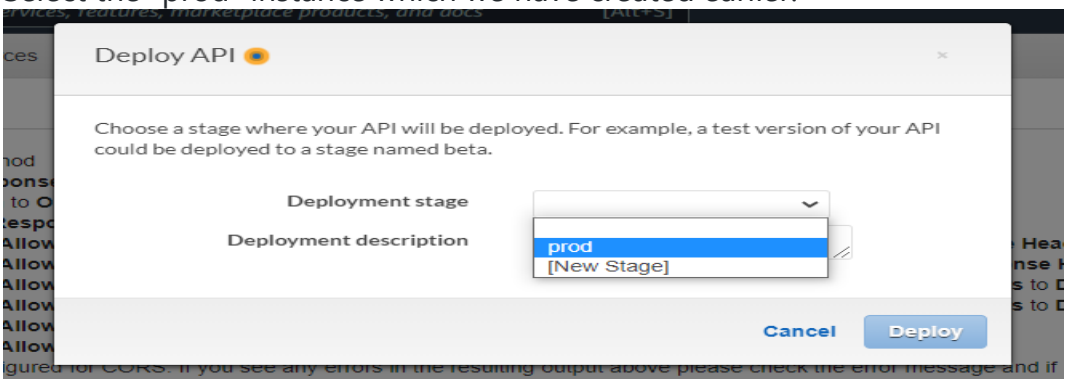
- Click on "Yes, replace existing values" button to enable CORS for this API
- Below is the confirmation on Enabling CORS

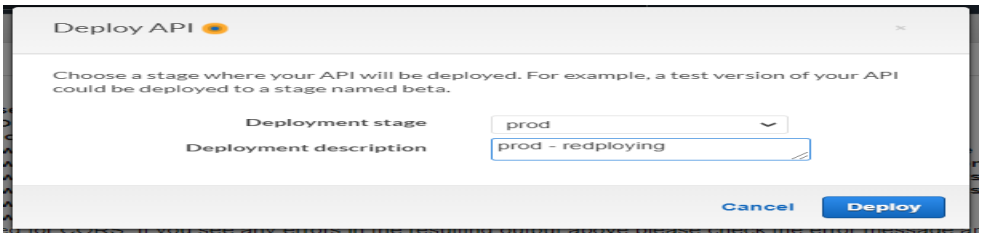


- Re-deploy API
  - Under Resources -> Click Actions -> Deploy API

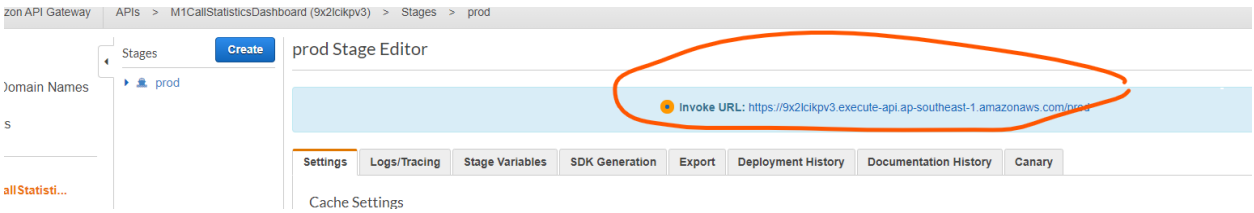


- Select the “prod” instance which we have created earlier:





  - 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

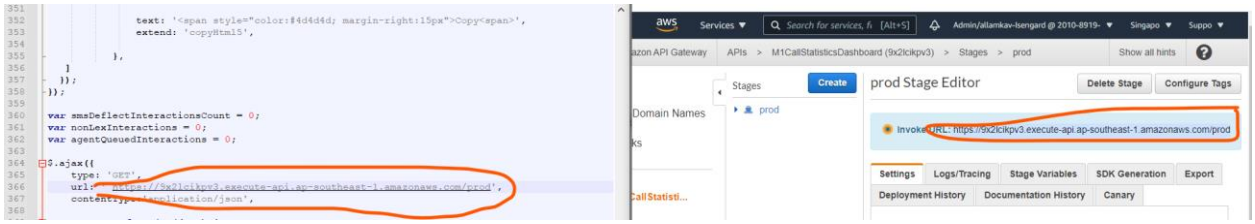

```
{"statusCode":200,"body":"{\"eventName\":\"OnGetM1CallStatsData\",\"channel\":null,\"contactID\":null,\"eventInfo\":[{\"ID\":7,\"Channel\":\"VOICE\",\"ContactID\":\"3b6a9bc7-bf1b-4cb1-be27-
6a20461b4573\",\"CustomerNumber\":\"+6563353529\",\"FlowType\":\"Non-Lex\",\"InitialContactID\":\"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\",\"InteractionDateTime\":\"20210901073120\",\"CustomerJourneyFlow\":\"Non-
Lex -> Pressed 1\"},{\"ID\":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-
ceb3a633c490\",\"CustomerNumber\":\"+6563353529\",\"FlowType\":\"Queue\",\"InitialContactID\":\"075d6481-b7dd-497d-beda-
ceb3a633c490\",\"InteractionDateTime\":\"20210831125139\",\"CustomerJourneyFlow\":\"BroadbandEnquiries\"},{\"ID\":4,\"Channel\":\"VOICE\",\"ContactID\":\"a52479ae-427b-49e1-b924-
b0b41df922b7\",\"CustomerNumber\":\"+6563353529\",\"FlowType\":\"SMS-Deflect\",\"InitialContactID\":\"a52479ae-427b-49e1-b924-
b0b41df922b7\",\"InteractionDateTime\":\"20210831125335\",\"CustomerJourneyFlow\":\"MobileEnquiries -> BillingEnquiries\"},{\"ID\":6,\"Channel\":\"VOICE\",\"ContactID\":\"3b6a9bc7-bf1b-4cb1-be27-
6a20461b4573\",\"CustomerNumber\":\"+6563353529\",\"FlowType\":\"Non-Lex\",\"InitialContactID\":\"3b6a9bc7-bf1b-4cb1-be27-6a20461b4573\",\"InteractionDateTime\":\"20210901073026\",\"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\",\"InteractionDateTime\":\"20210831124908\",\"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\",\"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 **"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

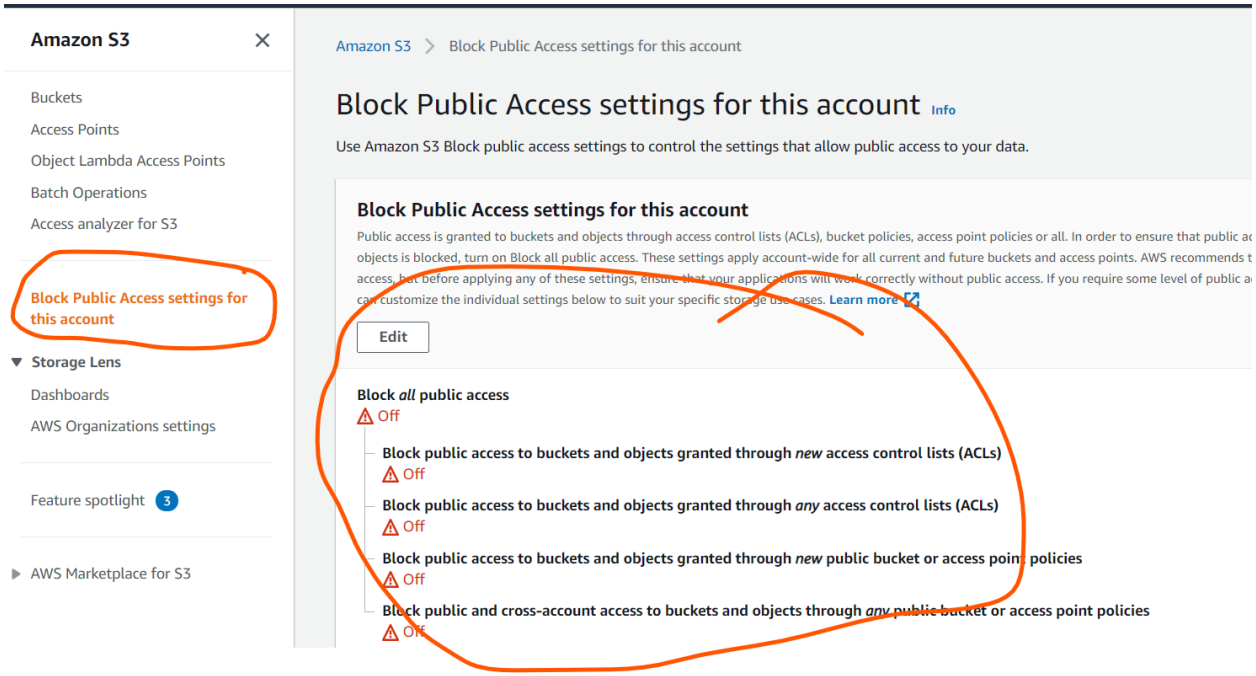
```
360  var smsDeflectInteractionsCount = 0;
361  var nonLexInteractions = 0;
362  var agentQueuedInteractions = 0;
363
364  $.ajax({
365      type: 'GET',
366      url: 'https://ycasu7jsd3.execute-api.ap-southeast-1.amazonaws.com/prod',
367      contentType:'application/json',
368
369      success: 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

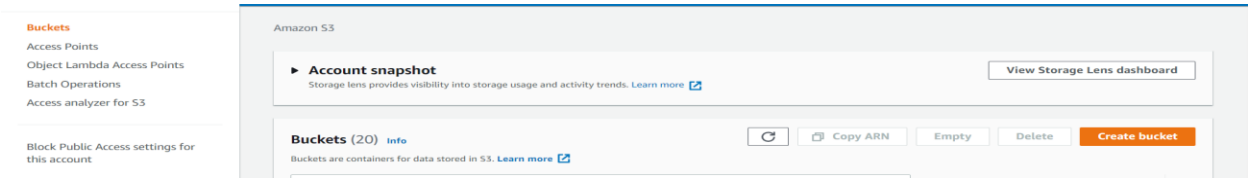


- Save the file

- 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”

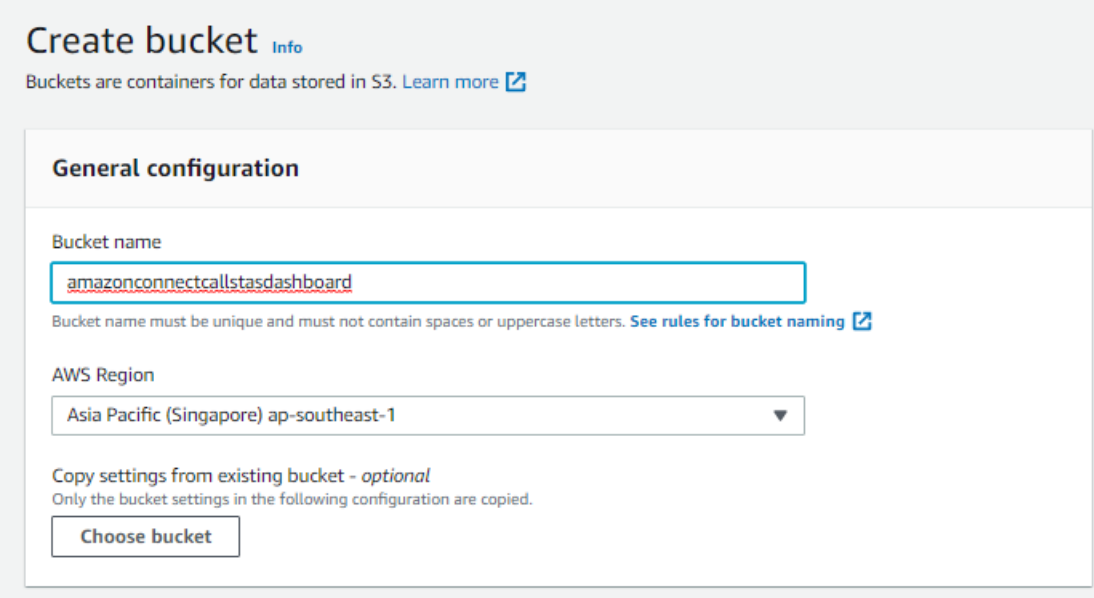


    - Click on buckets section from the left menu
    - Click on “Create Bucket” button

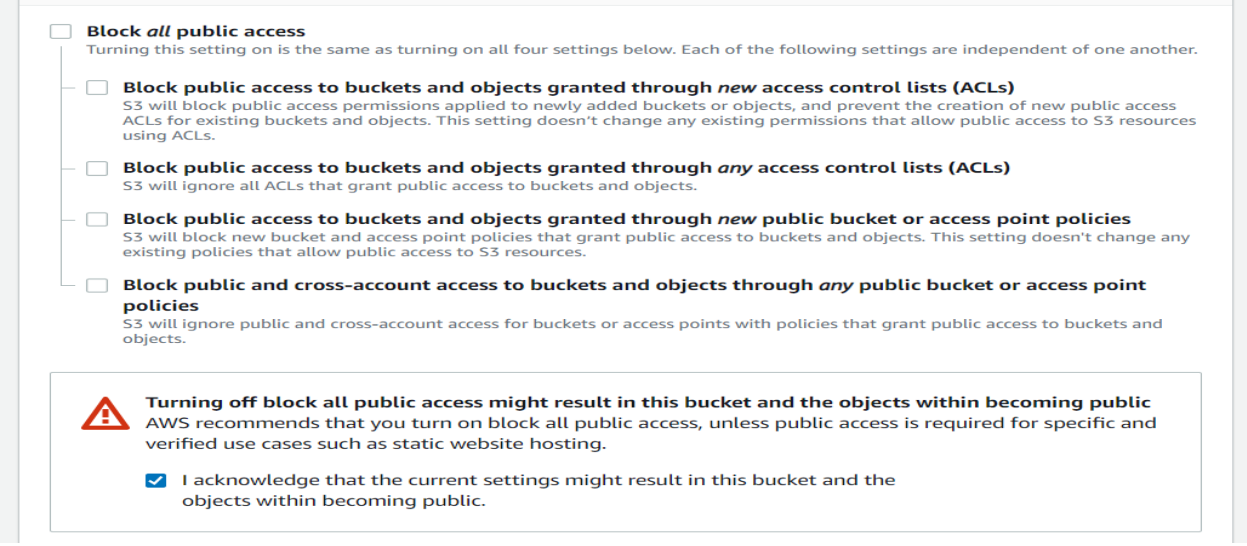


    - Under Create bucket configuration, provide as like below
        - Bucket name : **amazonconnectcallstasdashboard** should be unique in global

- AWS Region : ap-southeast-1

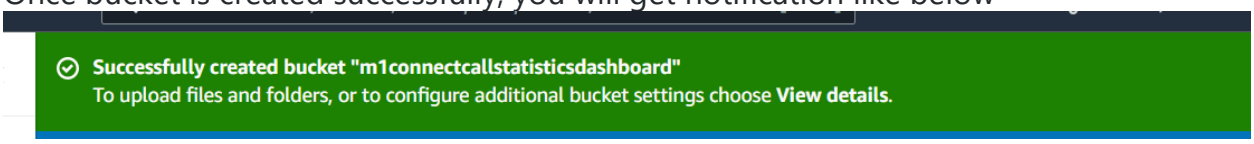


- Disable block access by unchecking the tick mark

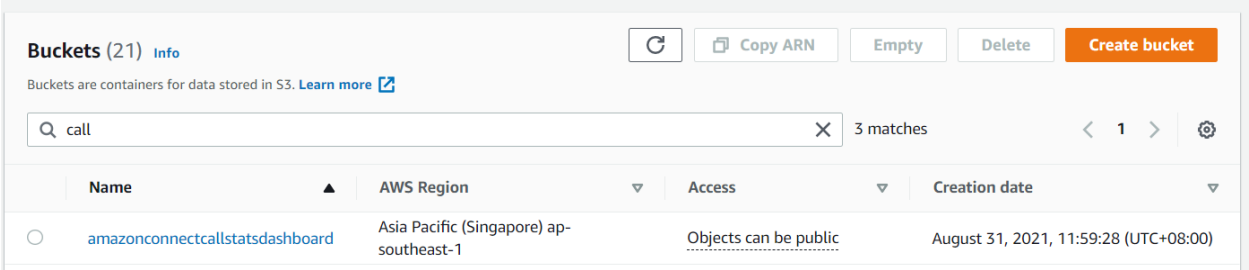


- Do not forget to acknowledge change to disable block access
- Leave the other options as is and click on “Create bucket” button.

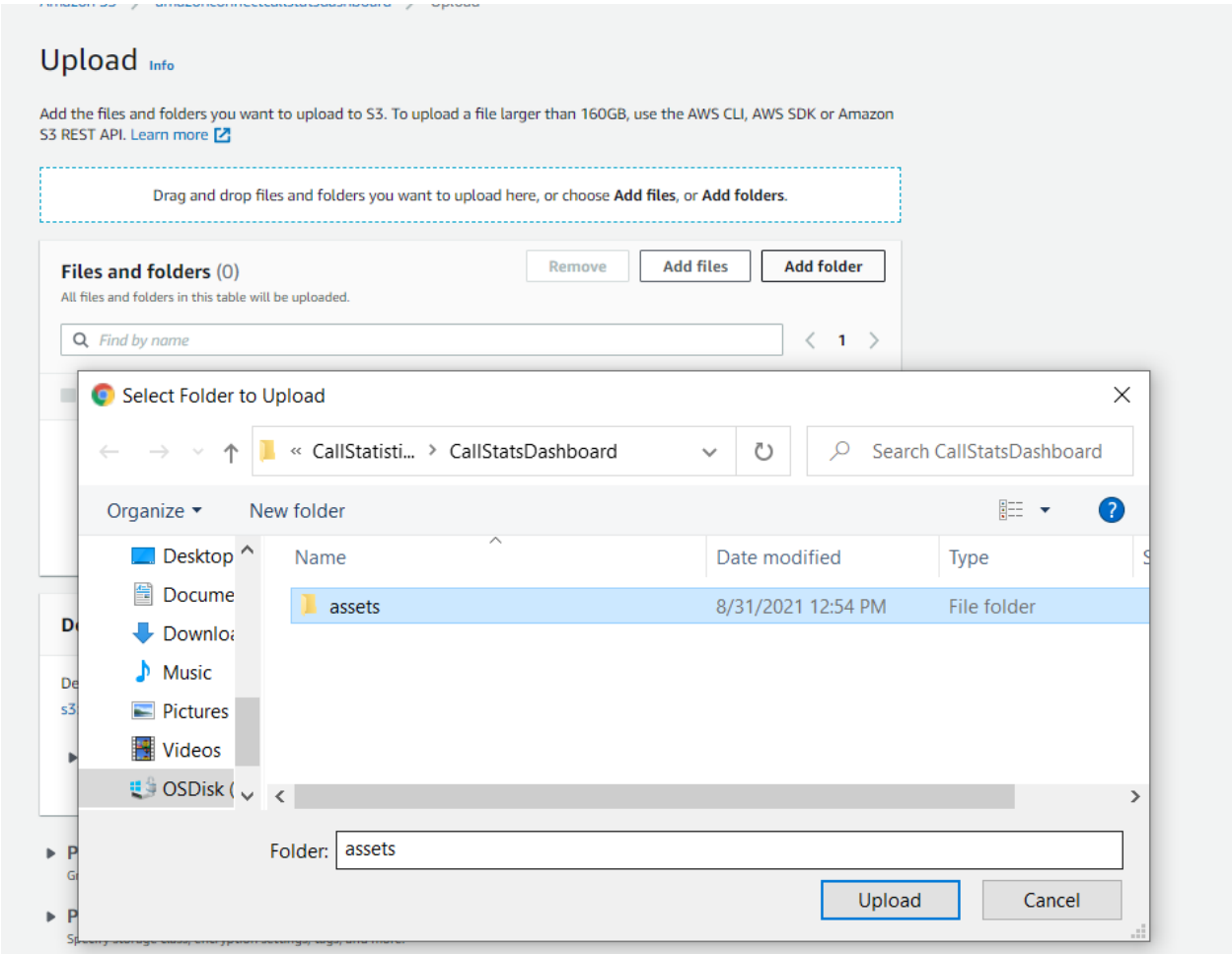
- Once bucket is created successfully, you will get notification like below



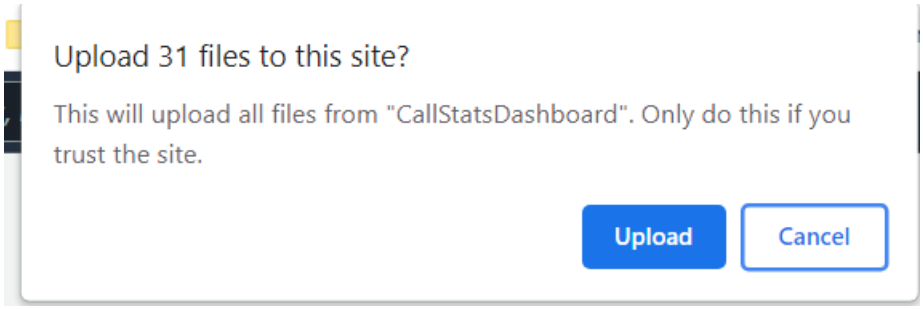
- Open the bucket by clicking on hyperlink provided to the bucket name



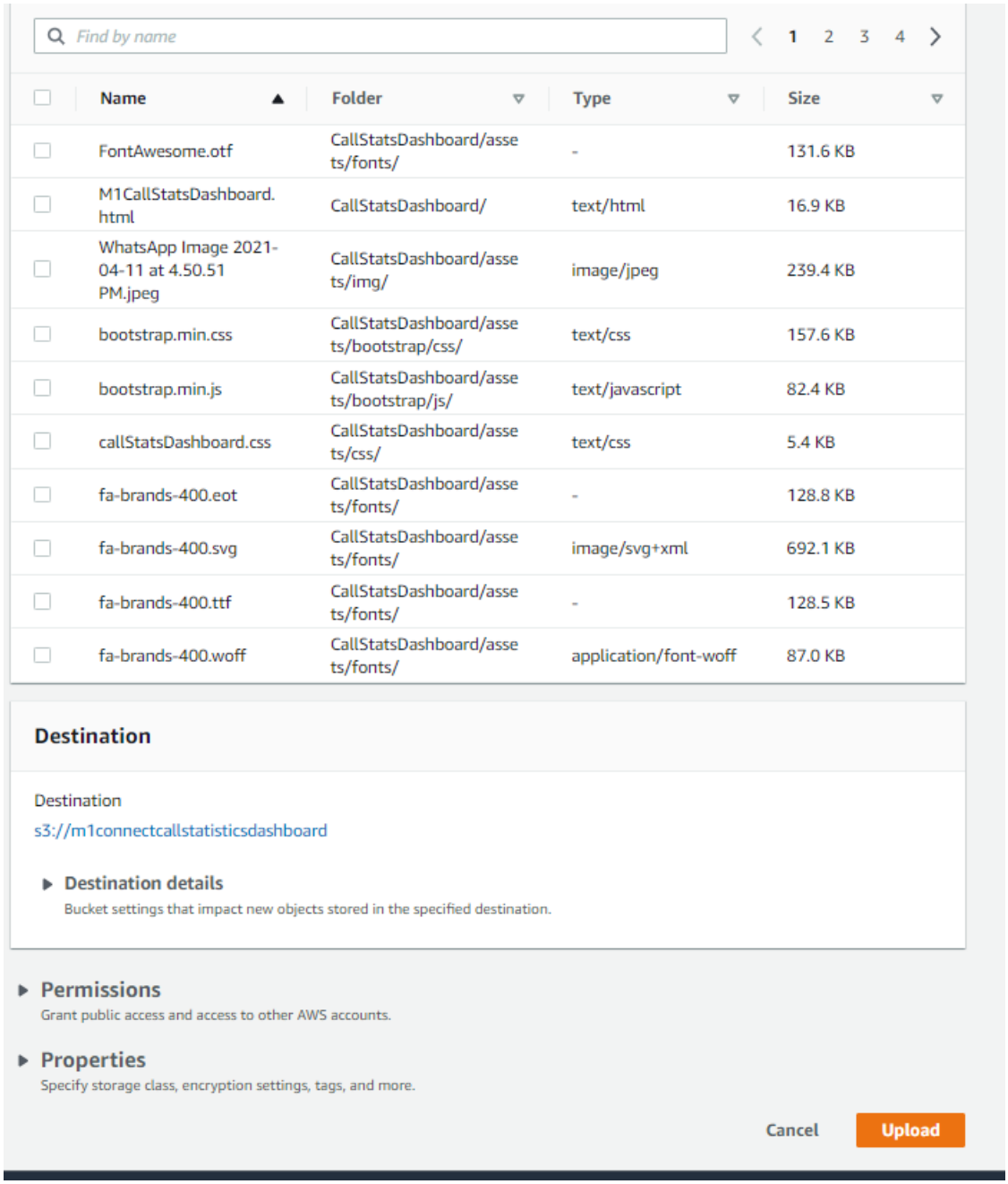
- Under Objects, Click on “Upload” button to upload the static files which will navigate to below page:
- Click on “Add folder” button to upload files. Which will open browse window to upload the folder. Navigate to the downloaded local folder (CallStatsDashboard) and choose uploading “assets” folder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



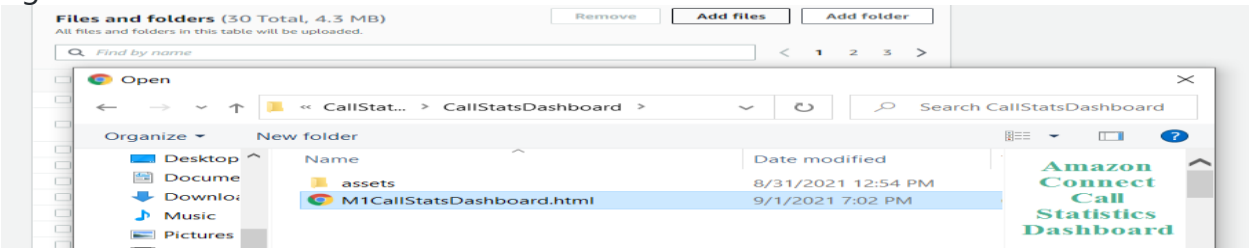
- Once you choose the folder, click on “Upload” button to upload files
- Accept the confirmation dialog box by clicking on “Upload” button



- Then files will be loaded to console.

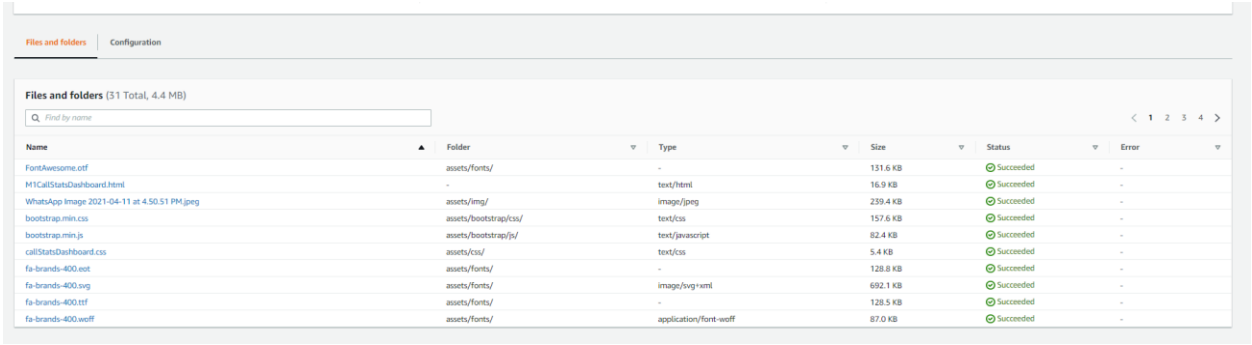


- Again click on Add Files button to add “ConnectCallStatsDashboard.html” file

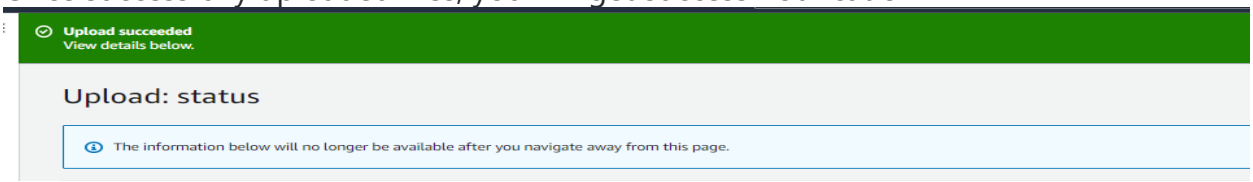


- Then click on “Upload” button to upload all the files to S3 bucket

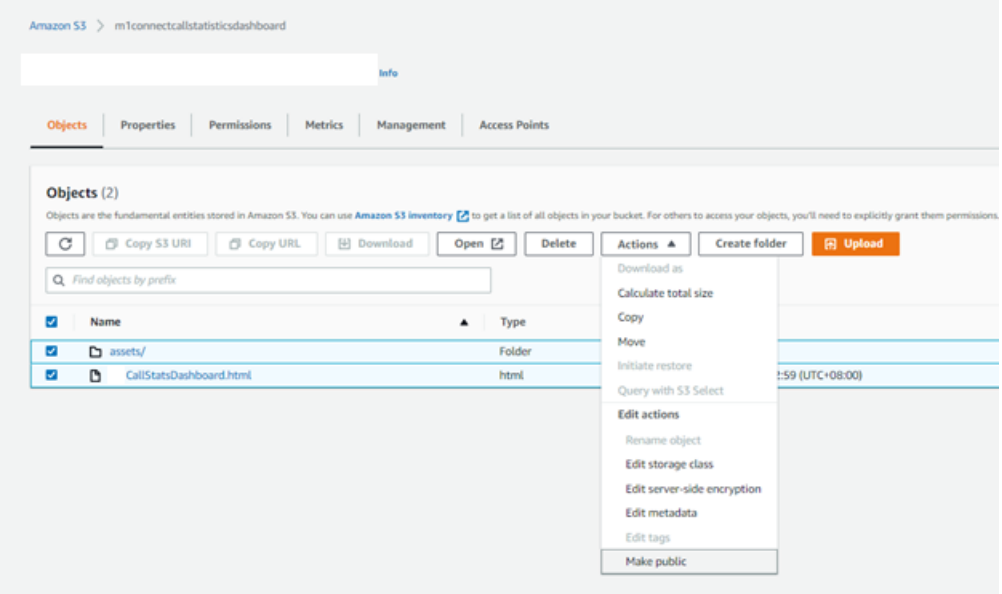
- You can see upload status like below



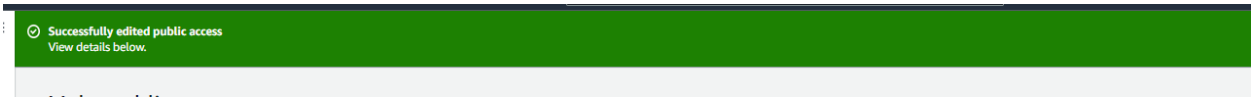
- Once successfully uploaded files, you will get success notification



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

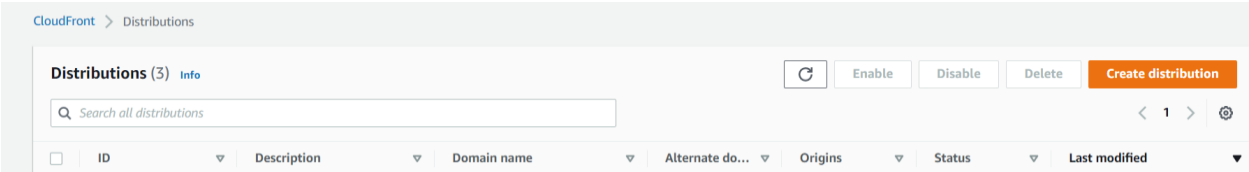


- Once successfully enabled public access, you will get below notification

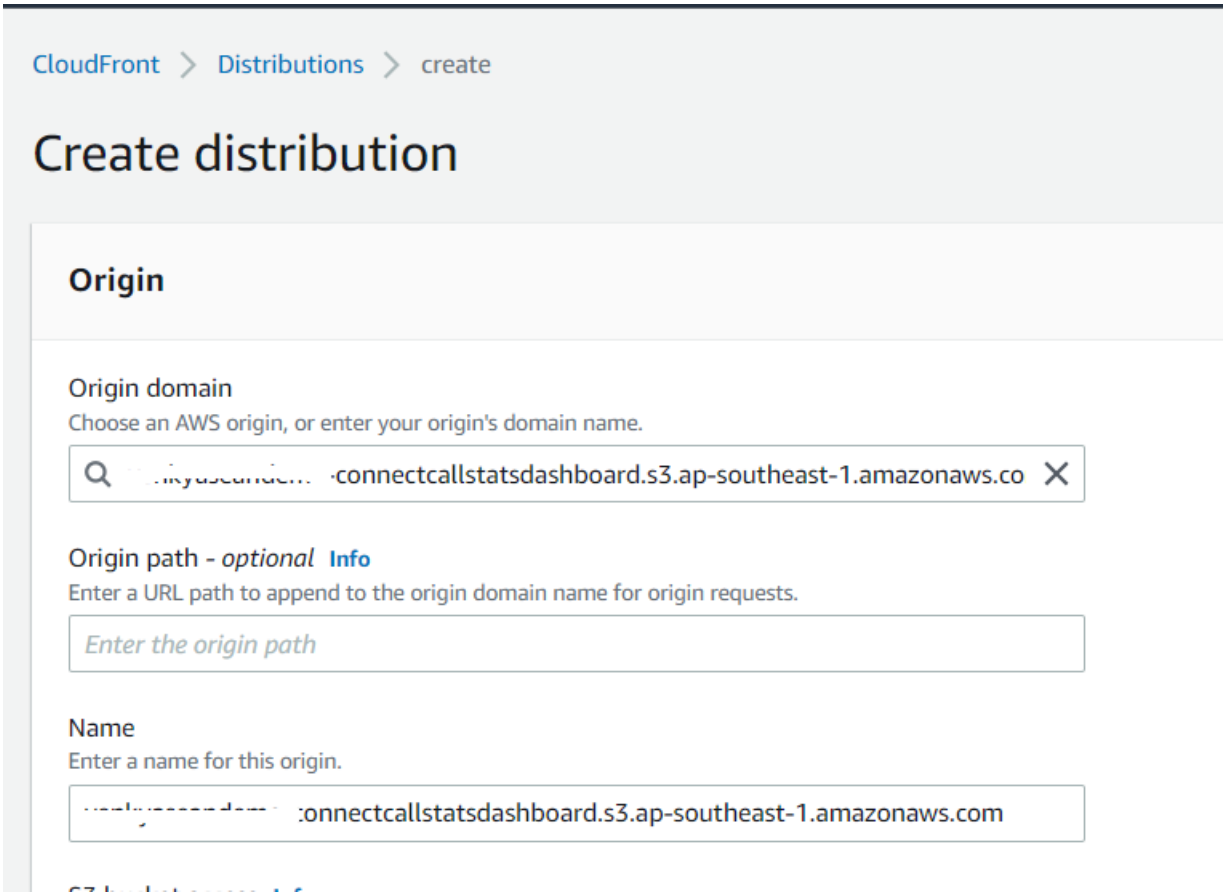


Creating CloudFront to access the S3 objects from public

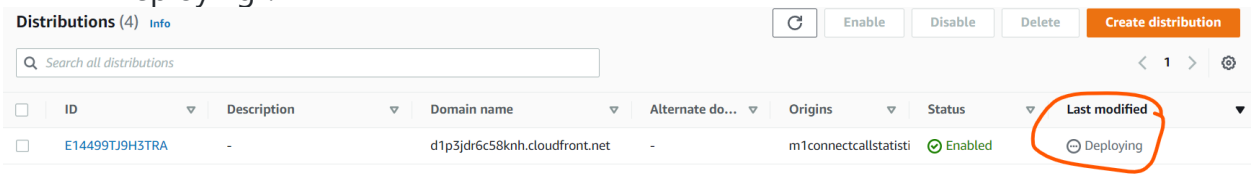
- Navigate to Cloudfront in console
- Click on “Create Distribution”



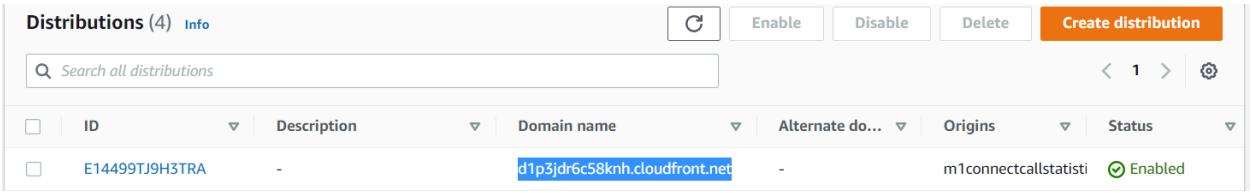
- Choose the options as mentioned below:
  - Origin domain : select s3 bucket name from the dropdown as “connectcallstatisticsdashboard” or the one which you have created.



- 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".

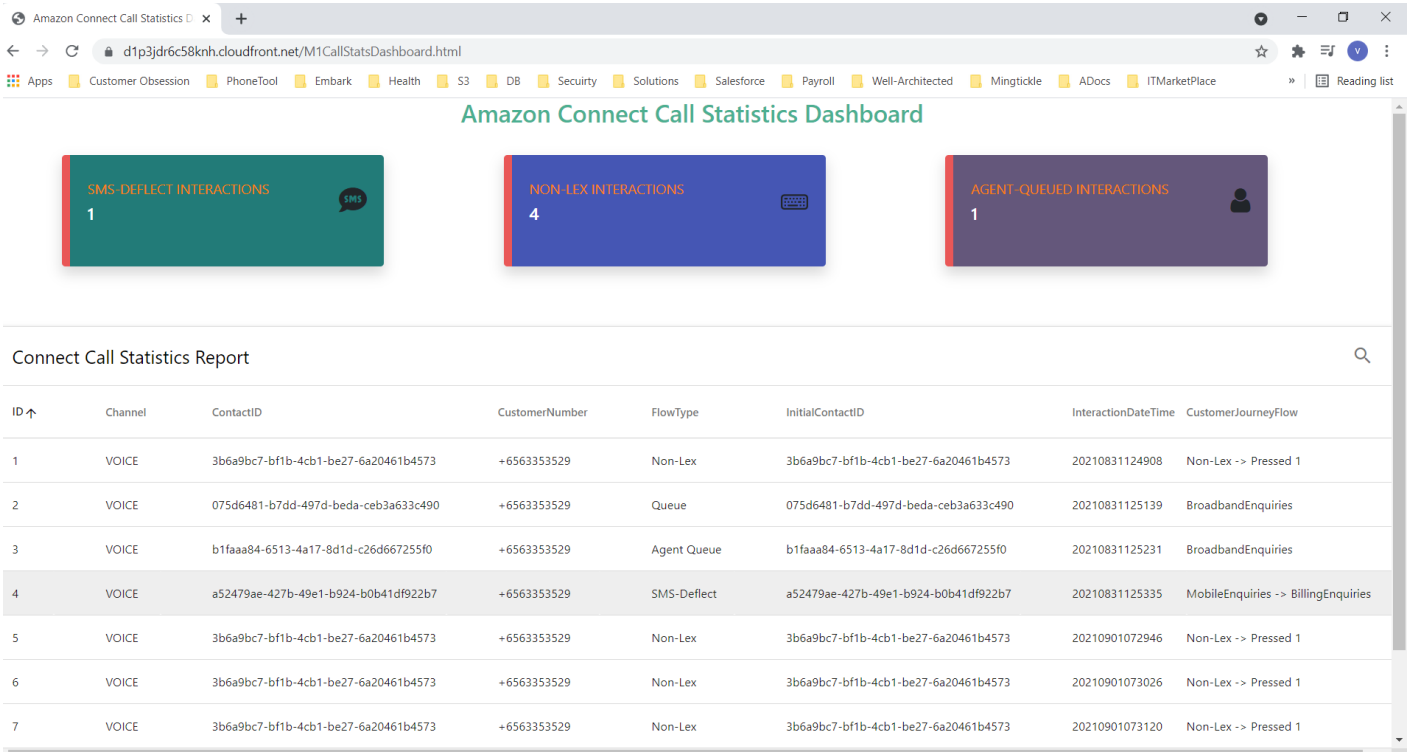


- Wait for some time until it is completely deployed.
- Once it is successfully deployed, get domain name which we can use to access our dashboard.



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