



# Hey ... How Can We Help You?

**AI for a Better Customer Support Experience**

## **TEAM #2**

Vijay R Dhulipala

Samira Arnodekar

Akhil Kodali

Anisha Mula

Vijay K Ragupathi

Michael Thompson

## External Links

To view our presentation on the business value and general needs of the format of our project and its execution check out our voice thread.

[Team 2's Voice Thread](#)

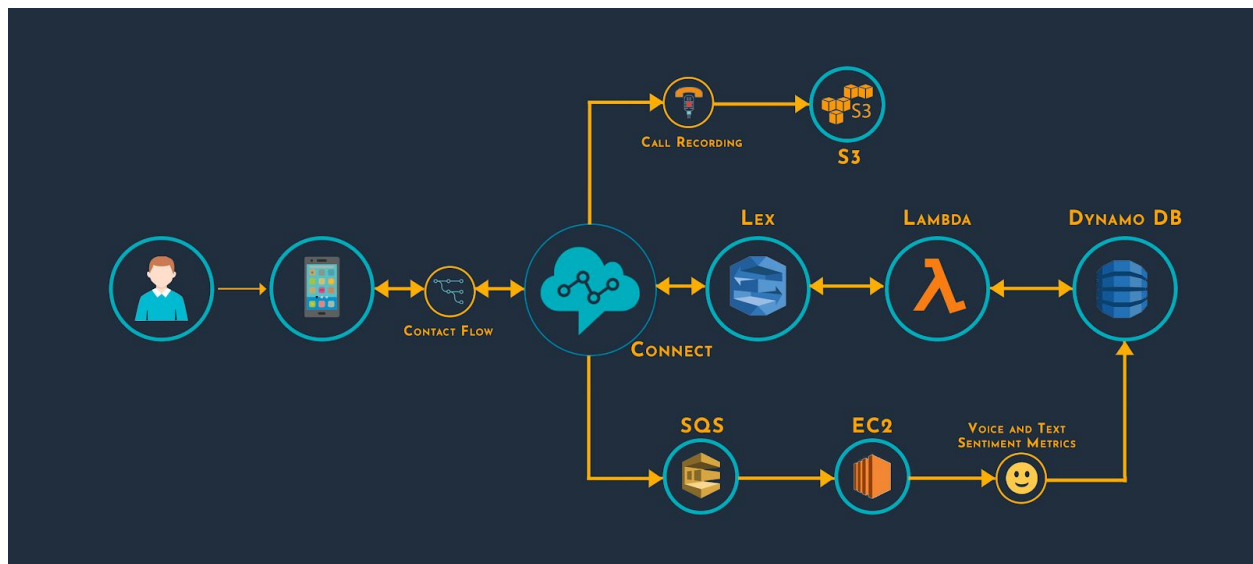
To view our work, details about the author, and additional resources check out our webpage.

[Team 2's Website](#)

If you want further detail on the coding execution and a repository of our customer code executions for each step reference our GitHub page.

[Team 2's GitHub Page](#)

## Technology Used



### Infrastructure overview

When a customer places a call to the customer support centre, the call is answered by Amazon Connect which is a cloud based contact centre infrastructure. It prompts the user for details. Then Amazon Lex is used to convert the identifying details spoken by the caller to text format. Lex is like a chatbot that talks to the customer based on our configuration. It runs NLTK at the backend to detect customer intents and invoke a response. It tries to extract the relevant information from the user's voice input required for further processing.

Depending on the nature of customer service requests, there might be multiple transactions between Connect and Lex. Following this, Amazon Lambda is triggered by Lex, and it is used to query the database for customer and order details of the caller. AWS Lambda provides an environment to run codes without managing servers and infrastructure. The database used here is Amazon DynamoDB. It is a key-value and document database with very low latency which is ideal to support the real time conversations for a customer support centre.

Once required details are fetched, the call is transferred to an agent and the conversation is recorded. The complete call recording is stored in Amazon S3 which is an easily scalable storage service offering the ability to query in-place. As a new voice file is loaded into S3, an event message is sent to Amazon SQS which stands for Simple Queue Service and offers a fully managed message queuing service. Each new file in S3 results in a message in the queue.

Further Amazon EC2 is used to run a script that polls the message queue for new requests. EC2 provides cloud computing services and is highly customizable, allowing us to set up new instances quickly and scale up as required. The script running on EC2 fetches the new voice file and identifies emotions based on both voice signals as well as text.

## Configuration details:

### Amazon Connect:

Amazon Connect is an easy to use omnichannel cloud contact center that helps companies provide superior customer service at a lower cost. Over 10 years ago, Amazon's retail business needed a contact center that would give our customers personal, dynamic, and natural experiences. We couldn't find one that met our needs, so we built it. We've now made this available for all businesses, and today thousands of companies ranging from 10 to tens of thousands of agents use Amazon Connect to serve millions of customers daily. We used connect to configure our call routing processes.

Step to launch and configure the lex connect instance:

1. Login to your console.aws.amazon.com
2. Search for connect in the services section
3. Click on add an instance
4. Check on Store users within Amazon Connect and give a name for the instance ex: 'trendsmsba6320' (This name should be unique)
5. Check the new admin box and provide all the details like name, email etc
6. Keep default settings and click next and finally click create instance.
7. Go to the following link and login : <https://trendsmsba6320.awsapps.com/>

8. Click on the 3rd icon on the left pane which has options like routing, contact flow and click on contact flow
9. Create a new contact flow and give a name
10. Now to configure your network follow the below steps:
  - 10.1 Pull the set voice block in set section and click on set voice and change the desired voice
  - 10.2 Pull the play prompt block from the interact section and click on the block and check the text to speech option and in the text box write your welcome message. Connect set voice to this block
  - 10.3 Pull the get customer input block from interact section, for setting in text to speech enter a prompt and click on amazon lex below, select the amazon lex bot and give the intent name for us it was 'OrderNum'
  - 10.4 next pull in set customer attributes block and we have set 4 attributes as a key value pair, values are set from the lex outputs
  - 10.5 next the check contact attributes block, set type to user defined, attributes to flag and condition equal to 1 and 0
  - 10.6 next a prompt to get the output from lex to the user.
  - 10.7 After this block, next block should be set call recording behavior and enable customer to record the customers voice
  - 10.8 next pull in the set working queue and set the agent which had been configured before
  - 10.9 then use the transfer to queue block and finally a disconnect or hang up block
  - 10.10 for all the error or default options a play prompt can be set
  - 10.11 Save and publish this flow
- 11 Next go to dashboard, first option in the left pane, click on view phone numbers
12. Click on the phone number and in the contact flow option select the contact flow created above.

▼ Slots ⓘ

Priority	Required	Name	Slot type	Version	Prompt	Settings
		<input type="text" value="e.g. Location"/>	<input type="text" value="e.g. AMAZON.US_CITY"/>		<input type="text" value="e.g. What city?"/>	
1. ▼	<input checked="" type="checkbox"/>	<input type="text" value="OrderNumber"/>	<input type="text" value="AMAZON.NUMBER"/>	Built-in ▼	<input type="text" value="Please State your order Number"/>	
2. ▲	<input checked="" type="checkbox"/>	<input type="text" value="CardNumber"/>	<input type="text" value="AMAZON.NUMBER"/>	Built-in ▼	<input type="text" value="Please confirm the last 4 digits of the card used"/>	

► Confirmation prompt ⓘ

▼ Fulfillment ⓘ

☒ AWS Lambda function ☐ Return parameters to client

Lambda function


[View in Lambda console](#)


Version or alias

Amazon Lex is a service for building conversational interfaces into any application using voice and text. Amazon Lex provides the advanced deep learning functionalities of automatic speech recognition (ASR) for converting speech to text, and natural language understanding (NLU) to recognize the intent of the text, to enable you to build applications with highly engaging user experiences and lifelike conversational interactions.<sup>1</sup>

**Error handling**


☒ Clarification prompts


*e.g. Sorry, can you please repeat that?* 

*Sorry I didn't get that! could you please state your Order ID* 

Maximum number of retries

Hang-up phrase

*e.g. Sorry, I could not understand. Please contact customer support.* 

*Sorry! I could not get your details properly. Disconnecting the call!* 

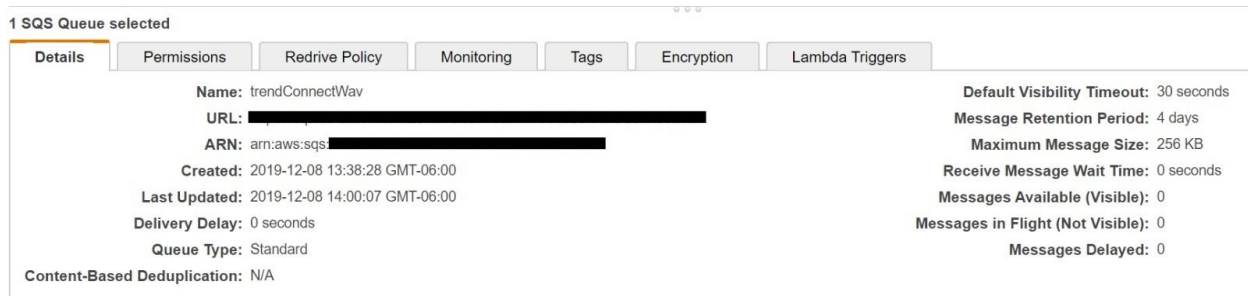
We used Lex as our interface for having conversations with the customers. With Lex, our goal was to be able to capture the order ID of the customer along with the last 4 digits of their credit card that they had used for the transaction to authenticate the user. We achieved this using the following steps:

- 1) Create an intent called 'OrderNum' and write sample utterances that a customer might generally say.
- 2) Create 2 slots - OrderNumber and CardNumber, to capture the numbers for both these items when uttered by the customer.
- 3) Write a Lambda script to verify fetch details from the database (DynamoDB) and attach the necessary permission roles such as complete DynamoDB access.
- 4) Under Fulfillment section in the Lex console, select AWS Lambda function and choose the Lambda script that was written from the dropdown.
- 5) Once Lex recognizes the intent and captures the slots, it triggers this Lambda function for verifying these details with the data in the DynamoDB database.
- 6) When the user is authenticated, the Python function (in Lambda) returns a message with all the user details such as his/her name and the product name. The Python script should pass along some attributes such as date of purchase and the cost of the item in the JSON file in the return statement of the Lambda function.
- 7) Once the intent is fulfilled and the conversation is over, the Lex bot transfers the control back to AWS connect.

---

<sup>1</sup> "Amazon Lex – Build Conversation Bots - AWS." <https://aws.amazon.com/lex/>. Accessed 11 Dec. 2019.

# Amazon SQS



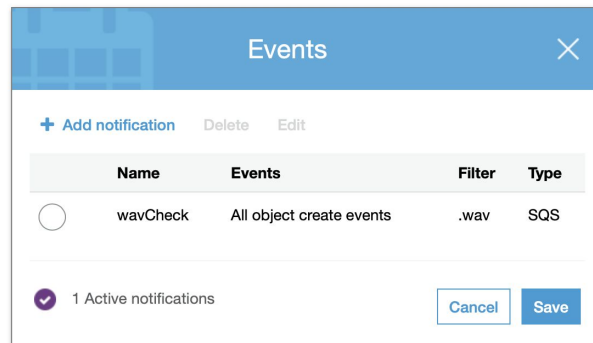
## Amazon S3

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics.

We used Amazon S3 to store our call data. Further S3 is connected with the build SQS.

Step to enable and configure event notifications for an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In the Bucket name list, choose the name of the bucket that you want to enable events for.
3. Choose Properties.
4. Under Advanced settings, choose Events.
5. Choose Add notification.
6. In Name, enter a descriptive name for your event configuration.
7. Under Events, Select all object create events
8. Enter an object name Prefix or a Suffix to filter the event notifications by the prefix or suffix.
9. Choose the type of destination to have the event notifications sent to.
10. Choose Save. Amazon S3 sends a test message to the event notification destination.



## Amazon lambda

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume.

With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

We used this service to set up triggers which were used to fetch details from the dynamo DB database once the user gives his order number through lex and also trigger to put data into dynamo DB when a transaction file is dropped in S3.

## Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

EC2's simple web service interface allows you to obtain and configure capacity with minimal friction.

For our project we have used Amazon Deep Learning AMI (Amazon Linux) Version 26.0 which contains MXNet-1.6.0rc0, Tensorflow-2.0 & 1.15, PyTorch-1.3.1, Keras-2.2, & other frameworks, configured with Elastic Inference, NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker & NVIDIA-Docker. For fully managed experience. It runs on the Amazon Linux platform. Other configurations can be found in the figure XX.



Instance: [REDACTED] Private IP: [REDACTED]			
Description			
Instance ID	[REDACTED]	Public DNS (IPv4)	[REDACTED]
Instance state	stopped	IPv4 Public IP	[REDACTED]
Instance type	t2.xlarge	IPv6 IPs	[REDACTED]
Finding	Opt-in to AWS Compute Optimizer for recommendations. <a href="#">Learn more</a>	Elastic IPs	
Private DNS	ip-172[REDACTED]	Availability zone	us-east-1d
Private IPs	172[REDACTED]	Security groups	launch-wizard-6, all, view inbound rules, view outbound rules
Secondary private IPs		Scheduled events	-
VPC ID	vpc-fa81780	AMI ID	Deep Learning AMI (Amazon Linux) Version 26.0 (ami-02bd97932dabc037b)
Subnet ID	subnet-57a4a80b	Platform	-
Network interfaces	eth0	IAM role	-
Source/dest. check	True	Key pair name	ec2instance
T2/T3 Unlimited	Disabled		
EBS-optimized	False	Owner	967717004393
Root device type	ebs	Launch time	December 10, 2019 at 1:47:00 PM UTC-6 (9 hours)
Root device	/dev/xvda	Termination protection	False
Block devices	/dev/xvda	Lifecycle	normal
Elastic Graphics ID	-	Monitoring	basic
Elastic Inference accelerator ID	-	Alarm status	None
		Kernel ID	-

## Step to set up EC2

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Choose Launch Instance.
3. In Step 1: Choose an Amazon Machine Image (AMI), find an Amazon Linux AMI at the top of the list and choose Select.
4. In Step 2: Choose an Instance Type, choose Next: Configure Instance Details.
5. Choose Next: Add Storage.
6. Choose Next: Add Tags.
7. Name your instance and choose Next: Configure Security Group. (Use your IPs)
8. In Step 6: Configure Security Group, set Assign a security group to Select an existing security group. (You can create an open Security Group and assign it to the instance)
9. Choose Review and Launch.
10. Choose Launch.
11. Select the check box for the key pair that you created, and then choose Launch Instances.

We used Tensorflow iPython notebook to run a python script and run our CNN

## Step to set up Ipython Environment

1. SSH into EC2 using your key pair
  - a. `(ssh -i "ec2instance.pem" your@instance`
2. Open jupyter notebook
  - a. `jupyter notebook --port=8889`
  - b. `ssh -i ~/ec2instance.pem -L 8000:localhost:8889 ec2-user@ec2-54-80-254-90.compute-1.amazonaws.com`
3. Install the required libraries using pip install in the same environment
  - a. Libraries used: librosa, pandas, numpy, textblob and pydub
4. Use the ipython notebook available in the git with appropriate location for model weight and model.

5. Name your SQS in the mentioned placeholder
6. Run the script

## Amazon DynamoDB

Order\_table

Close

Overview

Items

Metrics

Alarms

Capacity

Indexes

Global Tables

More

Table details

Table name	Order_table
Primary partition key	OrderId (String)
Primary sort key	-
Point-in-time recovery	DISABLED <a href="#">Enable</a>
Encryption Type	DEFAULT <a href="#">Manage Encryption</a>
KMS Master Key ARN	Not Applicable
Encryption Status	
CloudWatch Contributor Insights	DISABLED <a href="#">Manage Contributor Insights</a>
	<a href="#">PREVIEW</a>
Time to live attribute	DISABLED <a href="#">Manage TTL</a>
Table status	Active
Creation date	December 8, 2019 at 10:45:10 PM UTC-6
Read/write capacity mode	Provisioned
Last change to on-demand mode	-
Provisioned read capacity units	5 (Auto Scaling Disabled)
Provisioned write capacity units	5 (Auto Scaling Disabled)
Last decrease time	-
Last increase time	-

## Voice Prediction

We know that emotions have the ability to capture many more subtle nuances than text. Also it is quite intuitive to know that a human cannot express the same emotion for more than a few seconds. Thus for our use-case, we wanted to capture the nuances in the customer's emotions and feelings throughout the call in the most effective way. So we chose to analyze the audio in short bursts using a rolling window to capture the variation in emotion. We used "pydub" for splitting the voice into smaller fixed size segments. Therefore, it is not quite right to output a single emotion for the entire call duration, rather we wanted to capture how the emotion was varying throughout the call. Since we were analyzing short bursts of speech, we trained the neural network model on the RAVDESS dataset which has a collection of short speech segments in a variety of emotions for 12 female and 12 male actors.

"librosa" which is a popular python package for speech and audio analysis, was used to generate features from the speech. We extracted Mel-frequency cepstral coefficients (MFCCs) from the voice, which have widespread applications in human speech-recognition and music retrieval systems. Finally, a neural network was trained on the frequency cepstrum to predict whether the emotion was positive, negative or neutral for the given audio sample. For our use case, we used a pre-trained model publicly available on GitHub.