AMAZON LEX

(Facebook Messenger Integration)

CHATBOT

**Table of Contents**

**[1.](#_Toc488162623)****[Introduction](#_Toc488162623)** [3](#_Toc488162623)

[1.1. Purpose 3](#_Toc488162624)

[1.2. Requirements 3](#_Toc488162625)

[**2.** **Amazon Lex** 3](#_Toc488162626)

[2.1. Bot and Intents 3](#_Toc488162627)

[**3.** **Lambda functions and Cloud-Watch Events** 4](#_Toc488162628)

[3.1. Lambda Function for Validation, Fulfillment and Notification. 4](#_Toc488162629)

[3.2. Cloud-watch events 5](#_Toc488162630)

[ **chatbot-change** :- Triggers Cloud watch event when the instance is in running state and a mail is sent to the user using SES. 5](#_Toc488162631)

[ **EC2Start\_Stop** :-Triggers Cloud-watch event when the Instance is in stopped state and a mail is sent to the user using SES. 5](#_Toc488162632)

[ **Chatbot\_termination**:-Triggers Cloud-watch event when the Instance is terminated and a mail is sent to the user using SES. 5](#_Toc488162633)

[**4.** **DynamoDB** 5](#_Toc488162634)

[**5.** **Facebook Application** 6](#_Toc488162635)

[ Created a Channel in Amazon Lex and published 6](#_Toc488162636)

[**6.** **Test your Bot** 7](#_Toc488162637)

[6.1. Tested in Amazon Lex console 7](#_Toc488162638)

[6.2. Tested in Facebook Messenger 8](#_Toc488162639)

[**7.** **Test Instructions** 9](#_Toc488162640)

[7.1. Facebook Credentials 9](#_Toc488162641)

[7.2. GitHub Credentials 9](#_Toc488162642)

[7.3. Video Details 9](#_Toc488162643)

[7.4. CreateInstance Intent 9](#_Toc488162644)

[7.5. Stop Instance Intent 9](#_Toc488162645)

[7.6. Terminate Instance Intent 10](#_Toc488162646)

# **Introduction**

## Purpose

Amazon Lex enables you to build applications using a speech or text interface powered by the same technology that powers Amazon Alexa. We are implementing a sample CHATBOT application using Amazon Lex. Following are the typical steps you perform when working with Amazon Lex:

* Create a bot and configure it with one or more intents that you want to support.
* Test the bot. You can use the test window client provided by the Amazon Lex console.
* Publish a version and create an alias.
* Deploy the bot - Facebook Messenger.
* Send Notification to end user.

## Requirements

To create and integrate Chatbot Application with Facebook Messenger you need below pre-requisites:

* Access to Amazon Lex, Lambda, RDS and EC2
* Create Lambda Functions for Validation, Fulfillment and Notification.
* Register an email address in Amazon Simple Email Service (SES).
* Create an Event rule in CloudWatch to trigger Notification.
* Facebook Developer Account.
* Facebook Application ID.

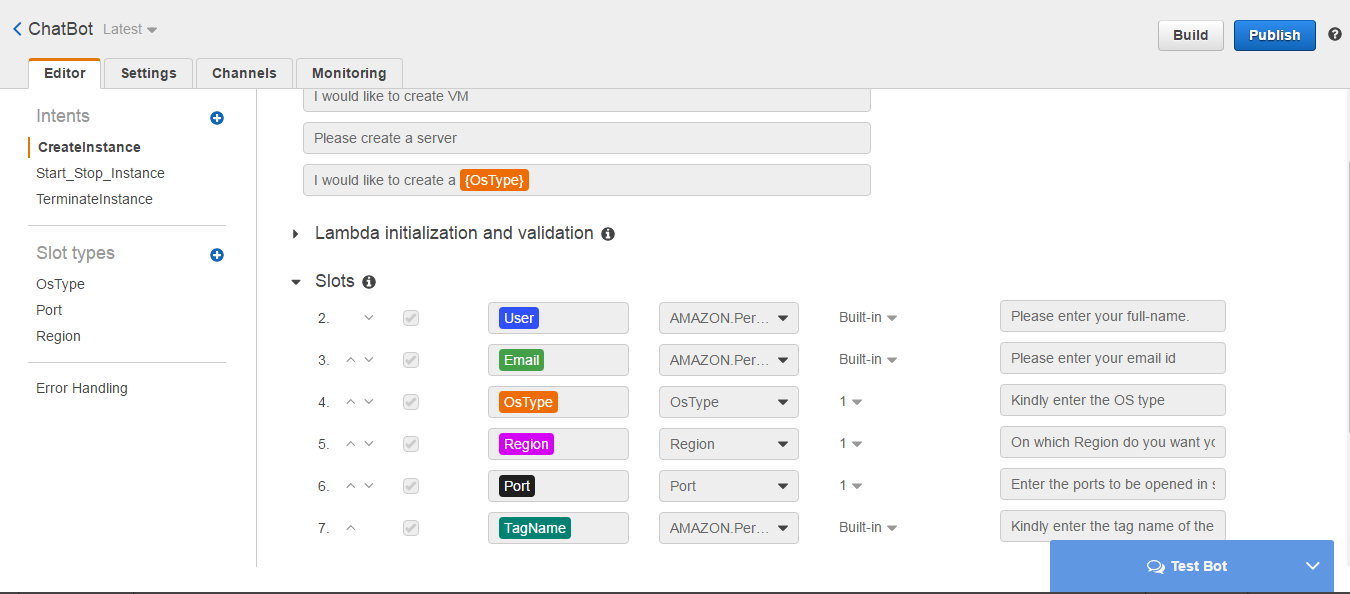
# **Amazon Lex**

## Bot and Intents

Created a Bot named **Chatbot** with following Intents:-

* + **CreateInstance:** - This Intent Creates an Instance.
  + **Start\_Stop\_Instance**:- This Intent Start or Stop an Instance based on the user input.
  + **TerminateInstance**: - This Intent Terminates the Instance.

Configured Intents with various Utterances and Slot types according to the requirements.



# **Lambda functions and Cloud-Watch Events**

## Lambda Function for Validation, Fulfillment and Notification.

Before configuring your Intent for utterances, slots and fulfillment your need to create Lambda Functions for Validation and Fulfillment. You will create two functions, one for Validation and other for Fulfillment.

Following are the lambda functions for Validation and fulfillment:-

* + - * **ChatBot\_Validation**:- This lambda function validates users input for creating Instance.
      * **Chatbot\_Fulfillment**:- This lambda function fulfills user Intent for creating Instance.
      * **ChatBot\_EC2Start\_Stop**:- This lambda function fulfills user Intent for Start and stopping the Instances based on the user input.
      * **Chatbot\_valdiationdeletion**:- This lambda function validates user input for deleting the instance.
      * **chatbot\_deletion**:- This lambda function fulfills user Intent to delete the instance.

## Cloud-watch events

## [**chatbot-change**](https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#rules:name=chatbot-change) :- Triggers Cloud watch event when the instance is in running state and a mail is sent to the user using SES.

## [**EC2Start\_Stop**](https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#rules:name=EC2Start_Stop) :-Triggers Cloud-watch event when the Instance is in stopped state and a mail is sent to the user using SES.

## [**Chatbot\_termination**](https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#rules:name=Chatbot_termination):-Triggers Cloud-watch event when the Instance is terminated and a mail is sent to the user using SES.

The event is triggered based on the following lambda function:-

* + - * [**EC2StateChange\_Logging**](https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions/EC2StateChange_Logging):- This lambda function triggers lambda function when the Instance is in running state.
      * **EC2Start\_Stop\_Notification**:- This lambda function triggers lambda function when the Instance is in stopped state.
      * [**EC2\_state\_change\_termination**](https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions/EC2_state_change_termination):- This lambda function triggers lambda function when the instance is terminated.

# **DynamoDB**

Created tables in DynamoDB for getting the Items and putting the items.

The following are the tables created for getting the Items:-

***amidetails***:-Based on the OS type selected by the user, we get Imageid and Instance type from the table created.

***region\_details***:-Based on the region selected by the user, we get AZ and Subnet\_id from the table created.

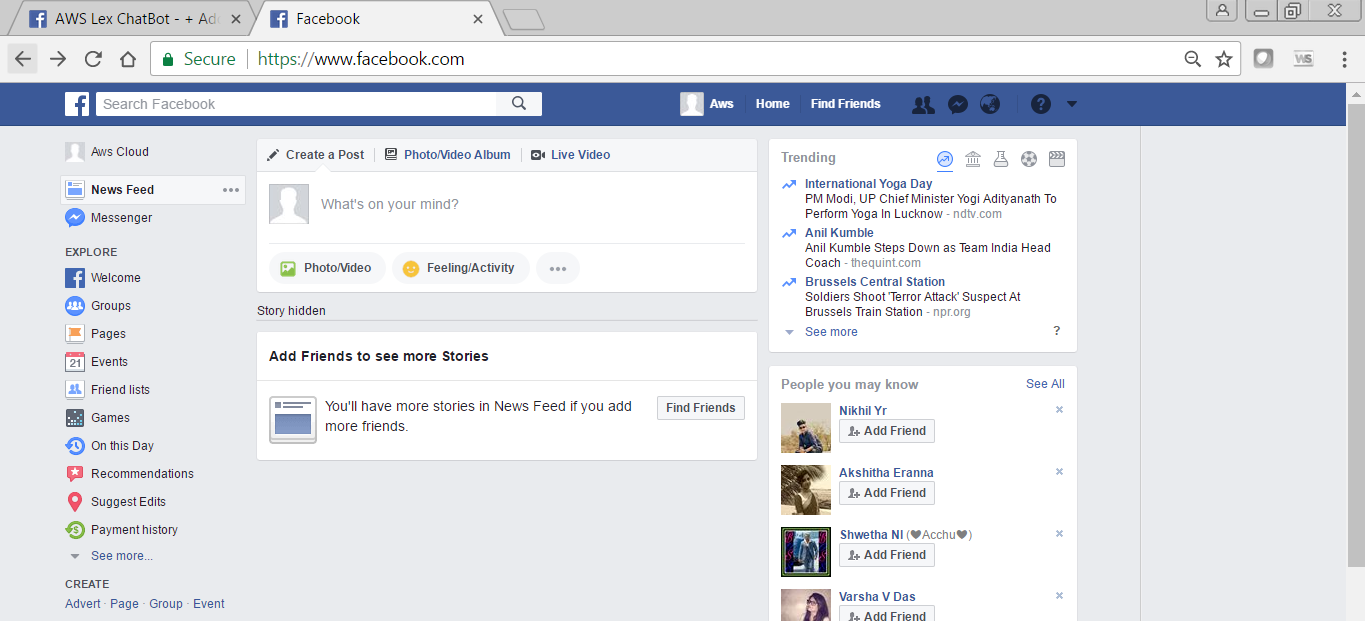
The following are the tables created for putting the Items:-

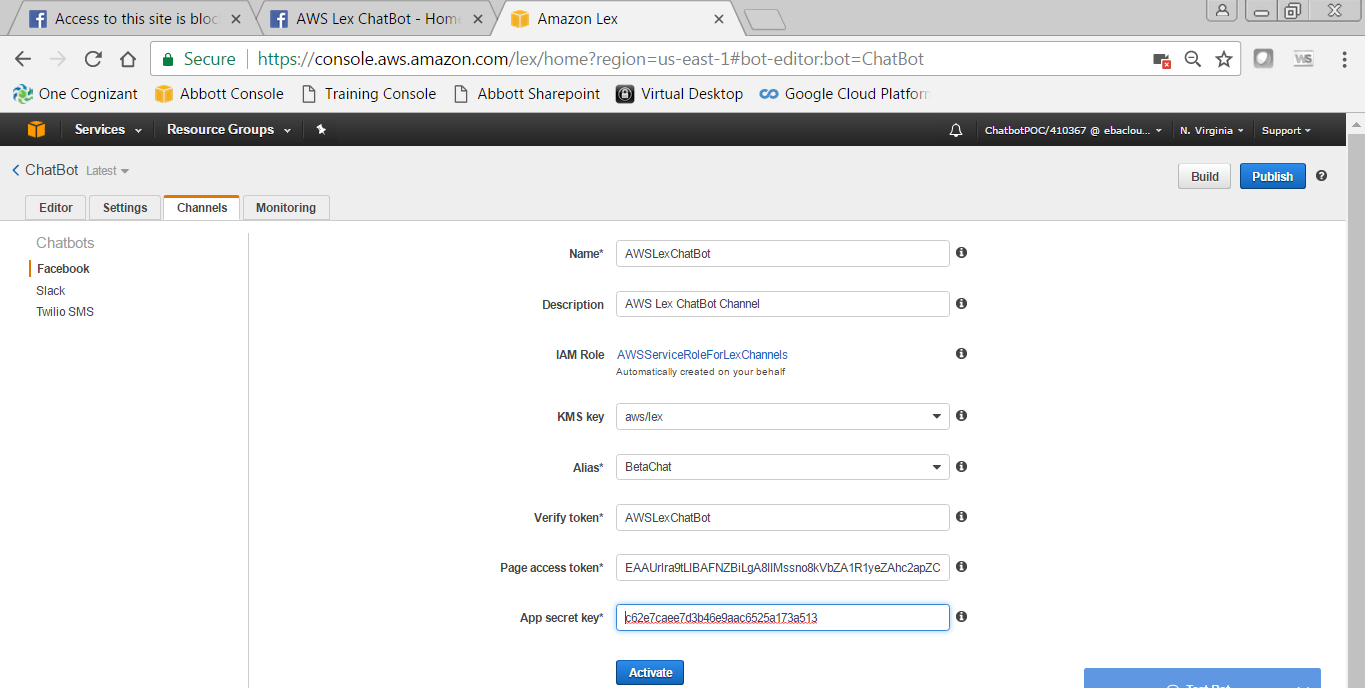
***Instanceiddetails***:-We put the Instance ID when the instance is up and running accordingly the username, email-id, tagname and securitygroup\_id.

# **Facebook Application**

* + Created a Facebook Account, Developer Account and a Facebook page
  + Configured Messenger Platform

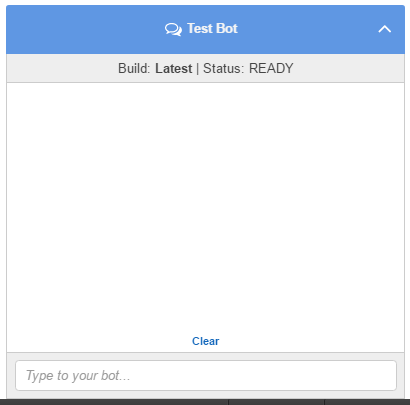
## Created a Channel in Amazon Lex and published

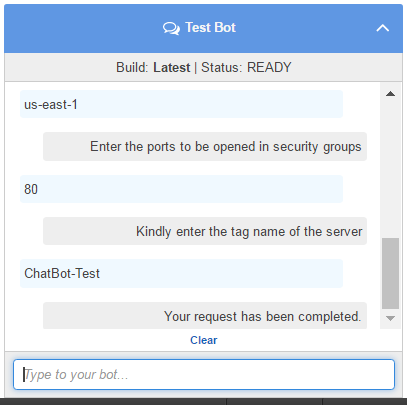


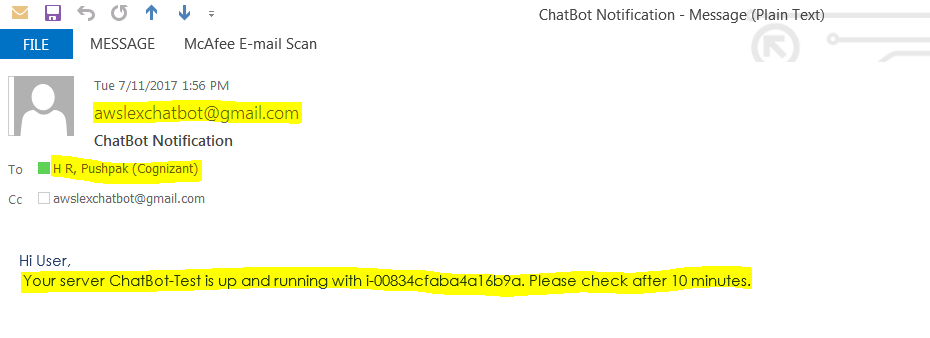


# **Test your Bot**

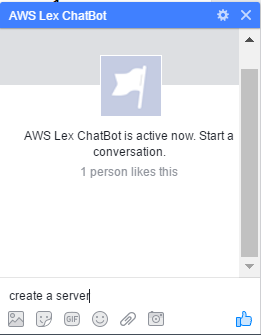
## Tested in Amazon Lex console

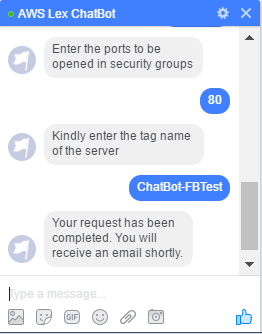


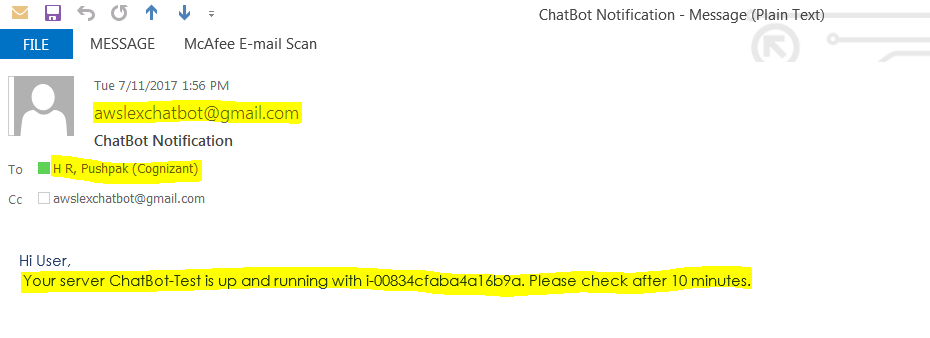




## Tested in Facebook Messenger







# **Test Instructions**

## Facebook Credentials

* User Name – [awslexchatbot@gmail.com](mailto:awslexchatbot@gmail.com)
* Password – Aws@123

## GitHub Credentials

* User Name – AwsChatBotPoc / [awslexchatbot@gmail.com](mailto:awslexchatbot@gmail.com)
* Password – Aws@123

***GitHub Link***: <https://github.com/AwsCloudPoc/CCS-ChatBot>

## Video Details

* User Name – [awslexchatbot@gmail.com](mailto:awslexchatbot@gmail.com)
* Password – AwsLex@123

***YouTube Link***: <https://youtu.be/9AhV4pmdE1A>

## CreateInstance Intent

Step1: Login to Facebook Account and open a new chat “***AWS Lex ChatBot***”.

Step2: Give utterance – “**create a server**”.

Step3: Give any username and remember it.

Step4: Give your valid email address *(you receive notification email to it).*

Step5: Give OS Type as – **“AmazonLinux/Windows2016Base/ RHEL7.3”**.

Step6: Give Region as – **“us-east-1”**.

Step7: Give Port as –

**“For AmazonLinux &** **RHEL7.3 - 22”**

**“For Windows2016Base - 3389”**

Step8: Give TagName as – Name you want your to be named.

## Stop Instance Intent

Step1: Give utterance – “**Stop a server**”.

Step2: Give UserName – given while create server.

Step3: Give TagName as – given while create server.

Step4: Give InstanceID as – Instance ID received in Notification Email.

## Terminate Instance Intent

Step1: Give utterance – “**delete a server**”.

Step2: Give UserName – given while create server.

Step3: Give TagName as – given while create server.

Step4: Give InstanceID as – Instance ID received in Notification Email.

\*\*\*\* End \*\*\*\*