Pest Management Chatbot for Lygus

This document will go over a detailed process of how to deploy the strawberry chatbot application to your own AWS environment. The application is composed of an Amazon Lex bot, four lambda functions, a Twilio callback URL, a weather API, and an RDS database.

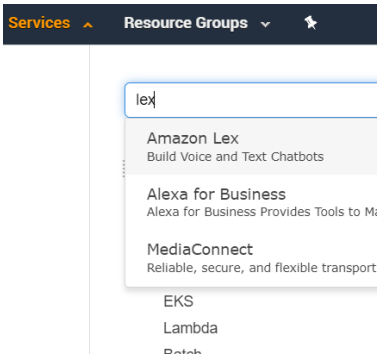
To start, download all the necessary files from the GitHub link provided:

https://github.com/cal-poly-dxhub/Pest-Management-Chatbot

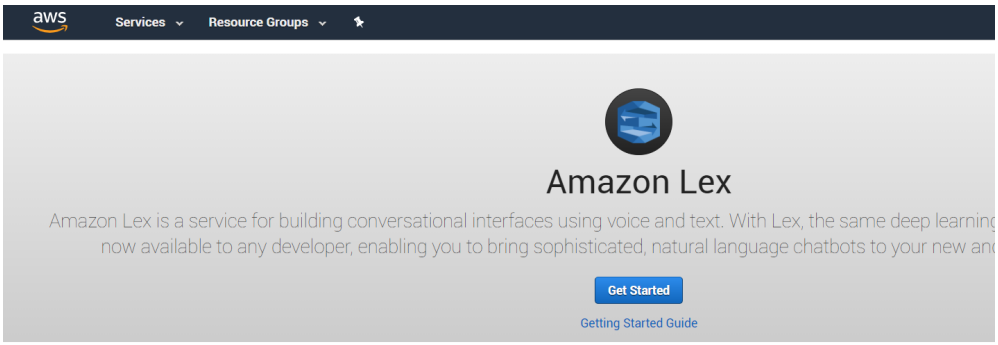
Amazon Lex

To allow for the application to communicate with the user and process natural language. We need to set up Amazon Lex.

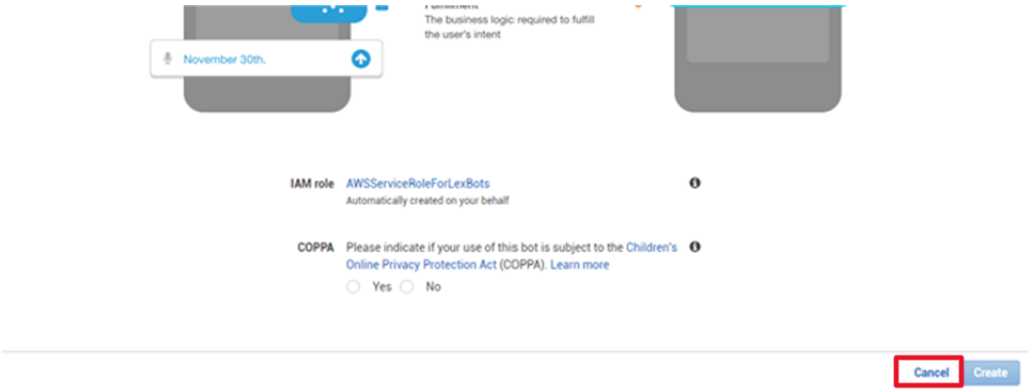
1. From the AWS console, hit “services” the type “lex” in the search box. You should see a suggestion for “Amazon Lex”, click it.



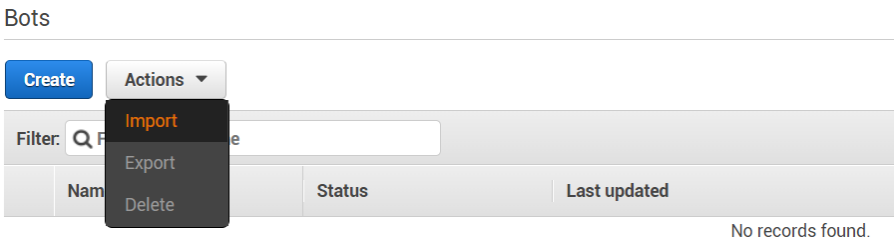
2. If this is your first-time using Lex, you will be prompted with a page that looks like this: Click on “Get Started”. (Skip to step #4 if you have used Lex before).



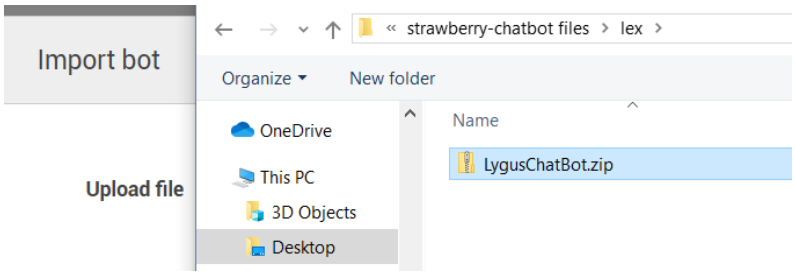
3. You will be prompted with a “Create your bot” page. Scroll down and click on cancel. (Skip to step #4 if you have used Lex before)



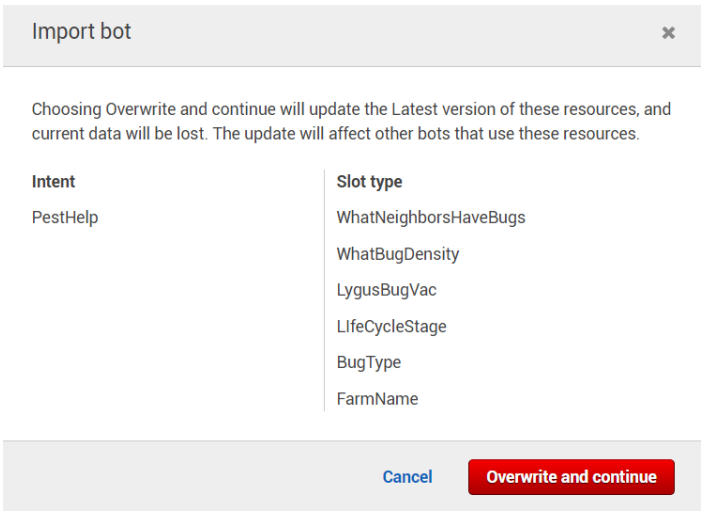
4. Now you will import the bot from the files you downloaded at the beginning. To do that, click on Actions, then import.



5. Click on “Browse”, then from the directory of files you downloaded, go to “Lex” then select the file named “LygusChatBot.zip”, then click on “import”.



6. You may be shown a warning about other bots being possibly affected. That won’t unless you have another bot with the same name as the one you are uploading (LygusChatBot). Click on “Overwrite and continue”.



7. Now you should be able to see your bot displayed on the list. Click on it to enter it. We will revisit the bot later to add the lambda functions and Twilio SMS.

Twilio SMS

For Lex to chat with a messaging system such as Facebook, Slack or SMS, you need to establish a channel for Lex to use to communicate with that system. This chatbot will be working with SMS which means that Twilio is the easiest way to integrate with Lex. Twilio is a third-party SMS platform that will handle the forwarding and receiving of text messages to the Lex chatbot. To get this setup, follow the instructions here after setting up the Lex chatbot in your account.

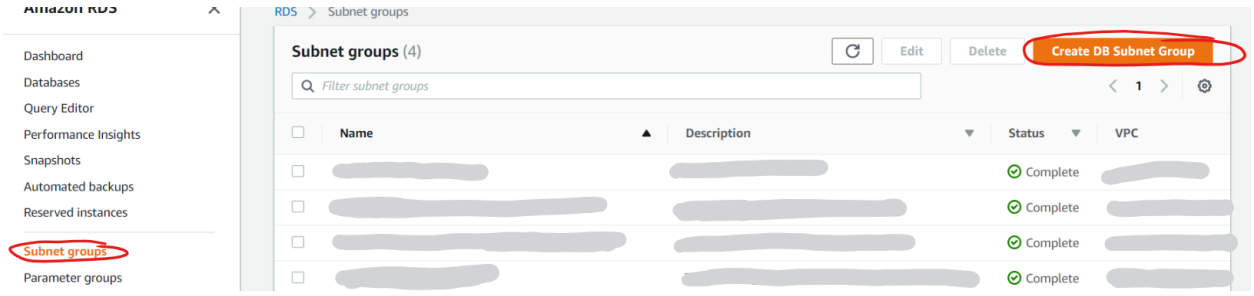
https://docs.aws.amazon.com/lex/latest/dg/twilio-bot-association.html

RDS Database

The chatbot requires a database to store users' info for validation purposes. In this document a MsSQL database is used. Any other database can be used provided it has the same schema, however you will have to modify the lambda functions to use the proper libraries to access any other database.

Before we start, you need to create a subnet group.

1. To do so, navigate to RDS in the console.
2. In the left panel, click on "Subnet groups", then click on "Create DB Subnet Group".



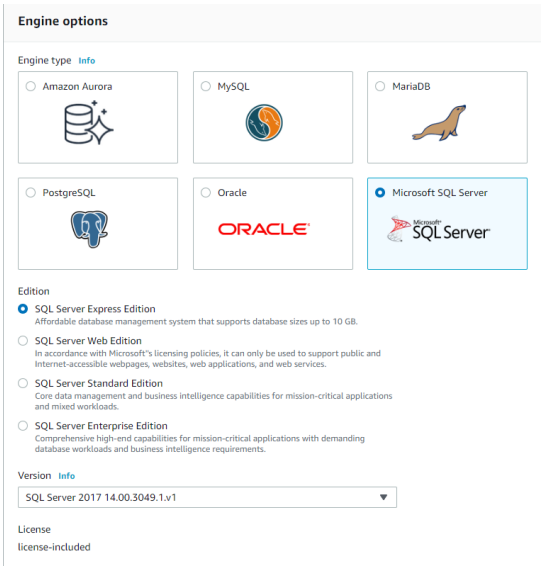
3. Fill out the “Name” and “Description” field. For the VPC field, select the VPC you plan for the project to reside in.

To Create the RDS Database Instance:

1 Navigate to RDS in the console. Click on create database.

2 You will be prompted with a setup page where the first option is to choose the creation method. Select “Standard Create”.

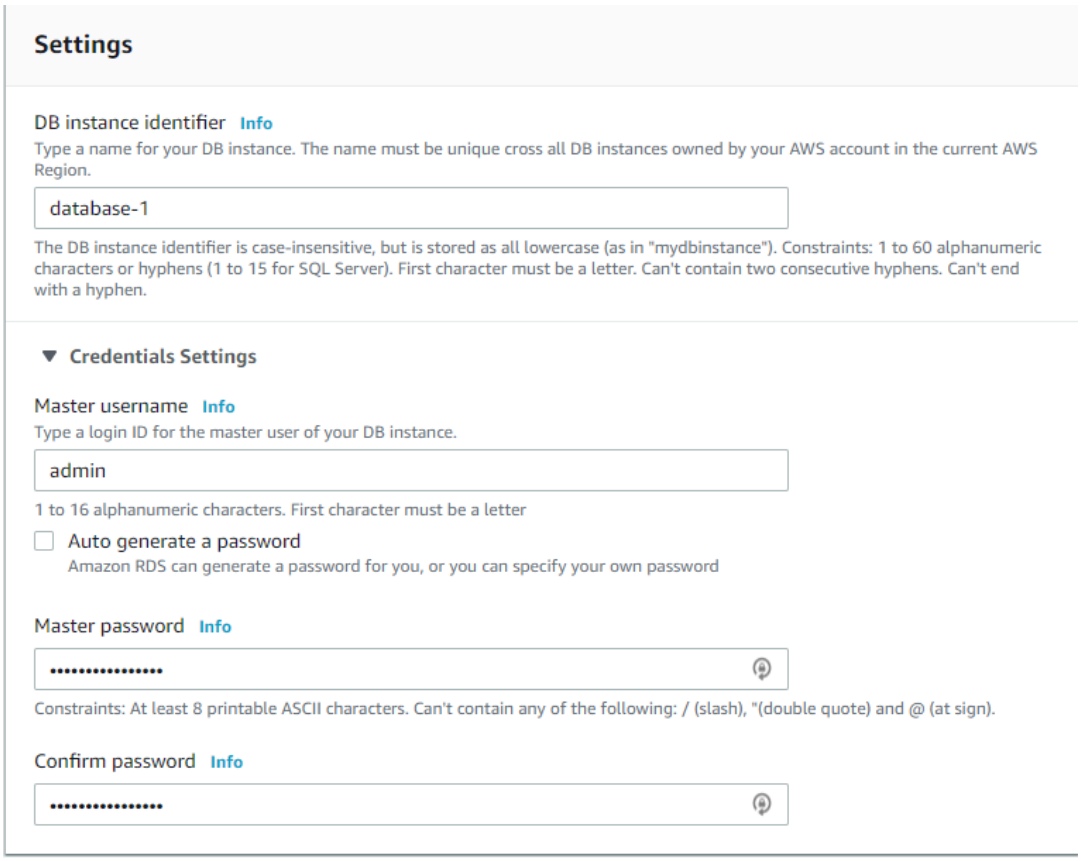
3 Under “Engine Options”, select “Microsoft SQL Server”. Under “Edition”, choose “SQL Server Express Edition”. Select the latest version.



4 Under “Templates”, we will go with the “Free tier”.

5 Under “Settings”, type a name for your database in the “DB instance identifier” textbox.

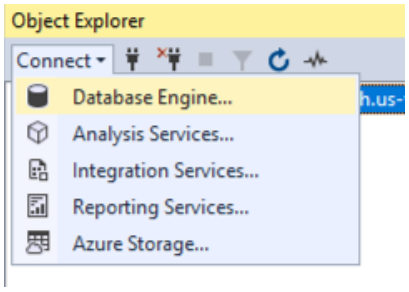
6 Click on “Credentials Settings” to expand it. There, you will be prompted to fill out master user information.



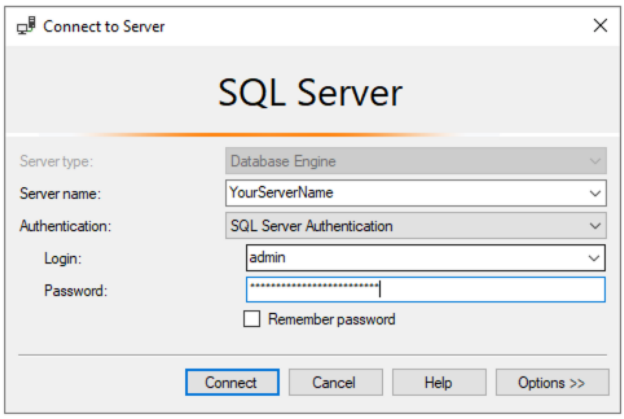
7 Keep the default settings on “DB instance size” and “Storage”.
8 In the “Connectivity” section, select the VPC you wish to use.
9 Click on “Additional connectivity configuration” to expand it, then:
    a. Select the subnet group you created earlier in this section.
    b. Select “No” for “Publicly accessible”.
    c. In the “VPC security group” section, select “Create new” then create a VPC security group.
10 Click “Create database”.

Create Database

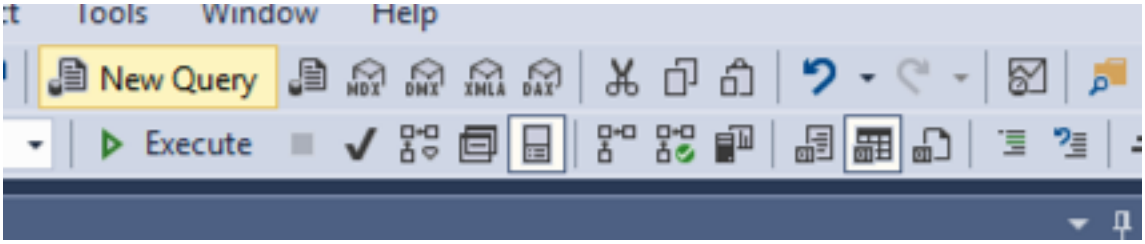
1. Open SQL Server Management Studio (Download SQL Management Studio) from an EC2 instance that has access to your RDS database instance.
2. In the top left, click on “Connect” and select “Database Engine”



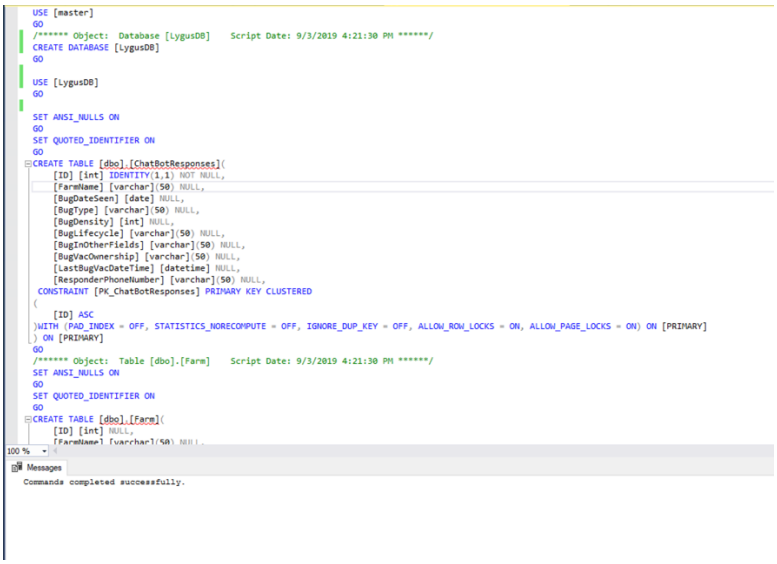
3. In the dialog box that comes up, connect to your RDS instance using the master user you created in the previous steps.



4. Click the “New Query” button in the top left to open up a new query window.



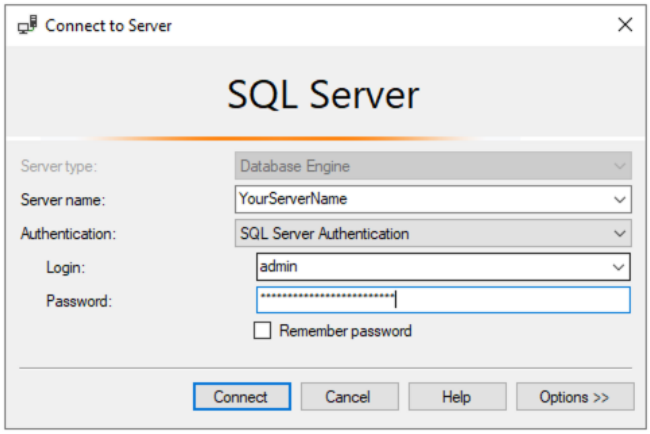
5. Copy and paste the code from the [DatabaseCreateScript.sql] file that was located in the github repository.



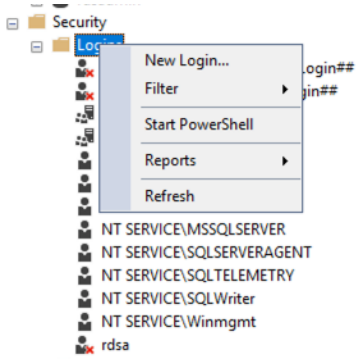
6. In the top left click Execute to create the database and tables needed for the Chat Bot to operate.

Create SQL User

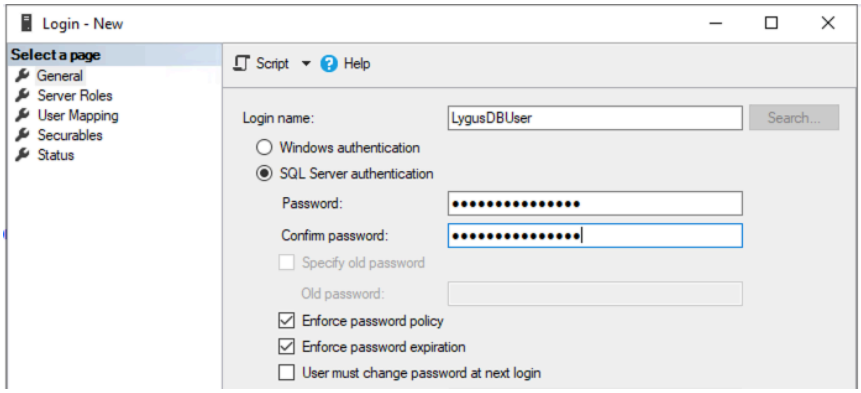
1. Again, connect to your SQL Server Instance.



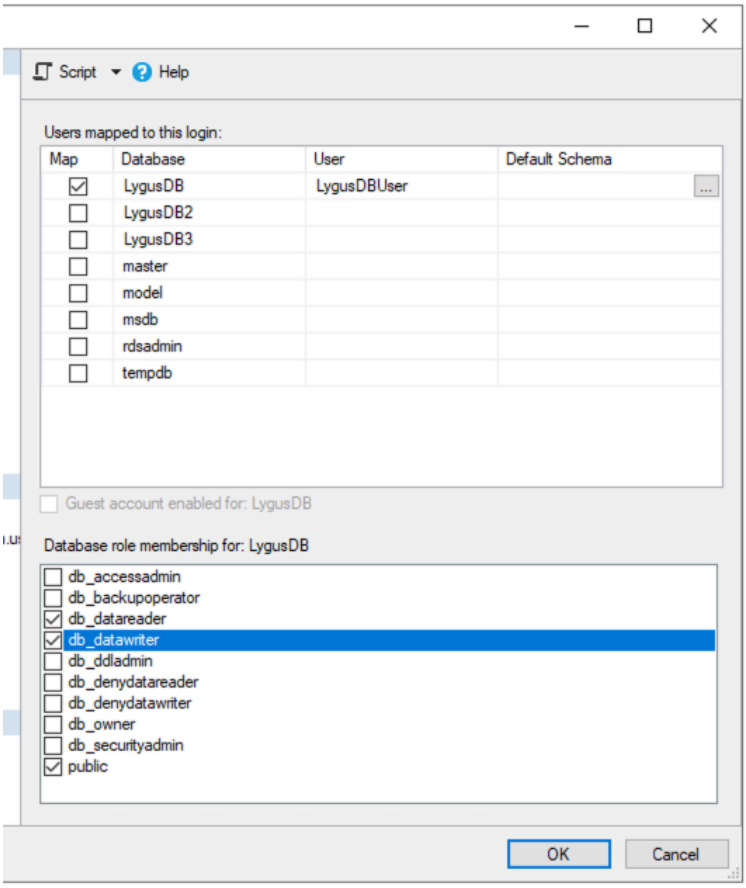
2. On the left-hand side in the object explorer, right click on “Login” under the “Security” folder and click “New Login.”



3. Give your user a name. Select SQL Server Authentication and type in the password you want to use for this user. Also, make sure to uncheck “User must change password at next login” since this user will be used by our application.



4. Click on User Mapping on the left-hand side of the dialog box. Select the database you created earlier and ensure that your user has the db_datareader and db_datawriter role membership.



5. Click OK to create the user.

Secrets Manager

This section will go over how to set up a secrets manager to securely store your RDS database information and make it easily accessible from your lambda functions.

1. From the top panel, select “services” then “Secrets Manager”.
2. Click on “Store a new secret”.
3. From there, choose “Credentials for RDS database”, then insert the username and password of the user you created for your RDS database.
4. Scroll down and select the database you just created in the last section, then click next.
5. Under “Secret name”, you could use whatever name you like. You will use this name later in the Lambda functions to access the RDS database. Click Next.
6. Configure rotation to your liking, then click next.
7. Finally click on Store in the review page.

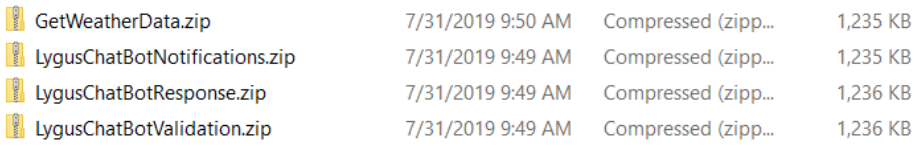
8. What you will need to from this section are: the secret name that you created in step 5 and the region name.

LAMBDA FUNCTIONS

In this section, I will go over how to set up the lambda functions responsible for running the logic behind the whole application.

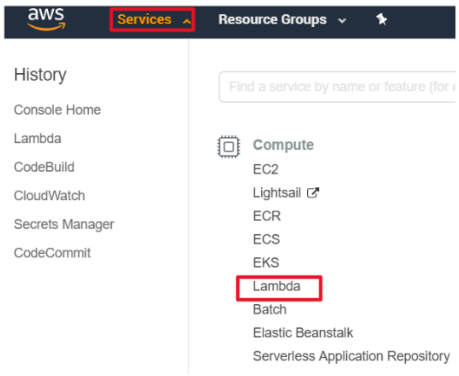
NOTE: The functions in this example use PyMsSQL to connect to a MsSQL database.

To start, you will need to have open your file you downloaded earlier in this tutorial. In that folder, go to /functions. You will see four zipped files like this:

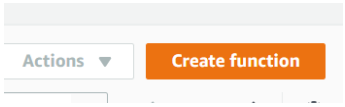


Each of these files will be deployed into a lambda function. They contain the code and libraries necessary to running the application. So, let's get started:

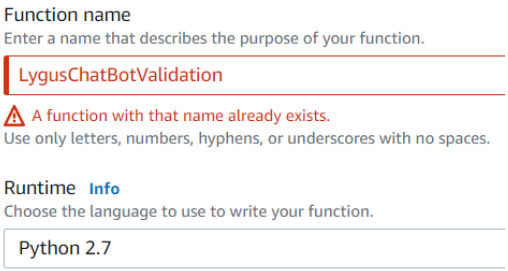
1. From the AWS console, hit "services" and click on "lambda".



2. Click on create function.

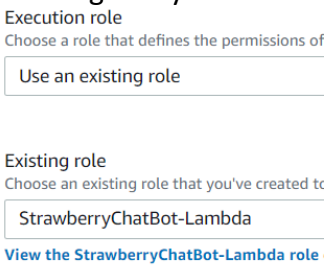


3. Choose "Author from scratch". Your function name should match the name of the file that corresponds to the function you are creating. For runtime, select Python 2.7.



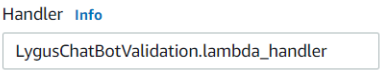
NOTE: I have already created a function with the name “LygusChatBotValidation”, so you won’t be getting this error.

4. Under “Permissions”, select the existing role you created earlier in this Tutorial.

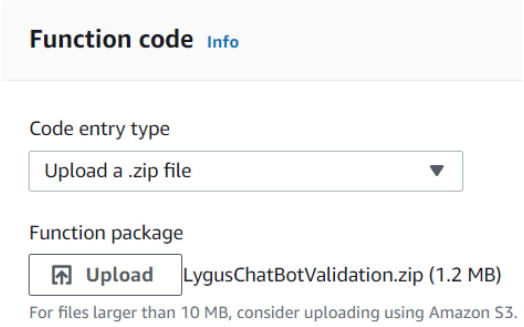


5. Click on create function.

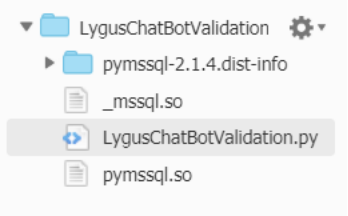
6. In the “Handler” box, copy and past the name of the function in the following format: FUNCTIONNAME.lambda_handler.



7. From “Code entry type”, click on “Upload a .zip file”, then click on Upload. Select the zip file that you named this function after. Once that is done, click on save in the top right corner.



8. From the code editors left panel, open the .py file that was extracted from the zip.



9. You should have the code open in the lambda code editor now. What you need to do now is fill in your secret manager information to the variables in the beginning of the code. Once you are done, click save.

```
8  from botocore.exceptions import ClientError
9
10 #INSERT YOUR INFORMATION HERE##########################
11 #SECRET MANAGER INFO
12 secret_name = "<<INSERT SECRET NAME HERE>>"
13 region_name = "<<INSERT REGION HERE>>"
14 ########################################################
15
```

10. NOTE: The .py file for "**GetWeatherData**" contains an extra variable slot for the **weather API**. You will get the weather API in the next section, so skip it for now.

11. Scroll down to "Network" then select the Virtual Private Cloud (VPC) and subnets that are appropriate to your project. It must be the same VPC in which the RDS resides in.

12. Under security groups, you need to add the security group you created in the RDS section earlier and create a new security group with the following permissions:

    - Access to the RDS database created earlier

Type	Protocol	Port Range	Destination	Description
MS SQL	TCP	1433	[redacted]	

    - Internet access

Type	Protocol	Port Range	Destination	Description
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	::/0	
HTTPS	TCP	443	0.0.0.0/0	

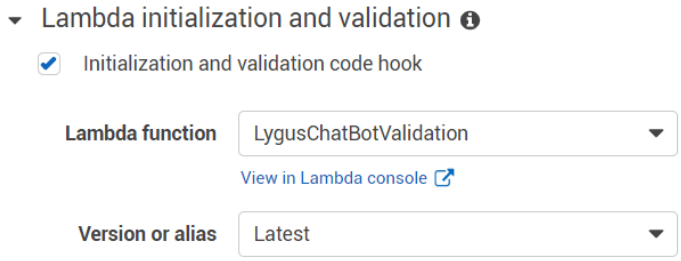
    Note: All the lambda functions require these permissions.

13. Repeat this process to create the three other lambda functions, one for each .zip file. Make sure to name the functions exactly how they are named on the .zip files

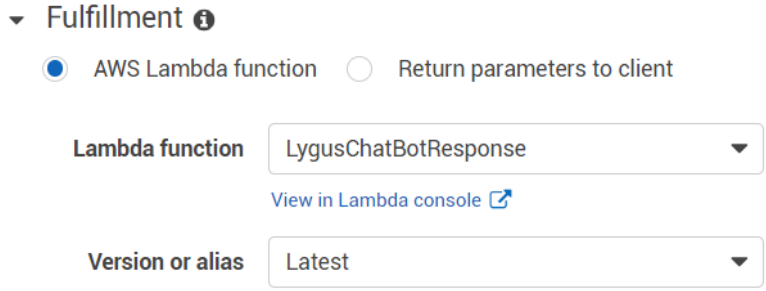
14. Once you have all the functions deployed and ready to go, we need to add the functions to your Amazon Lex bot. Go to your Lex bot that you created earlier in this tutorial.

15. Expand "Lambda initialization and validation", then check "Initialization and validation code hook". Under "Lambda function", select the "LygusChatBotValidation" function

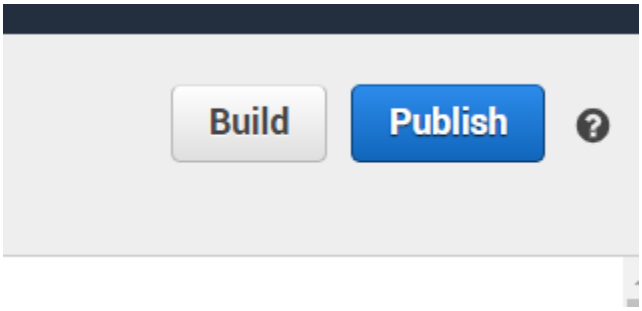
that you just created. You may see an “Add permission to Lambda Function” alert, click ok if you do.



16. Scroll down to “Fulfillment”, then select “AWS Lambda function”. You will be prompted with a Lambda function box, from there select “LygusChatBotResponse” function that you just created.



17. From the top panel, click on “Build”.



Weather API

This app makes use of a weather API to perform necessary functions. You will need to use your own weather API for this app.

We used Open Weather API (openweathermap.org/api), if you decide to use it there won’t be a lot that you need to change. However, if you are to choose a different API, there will be some minor changes you need to do.

1. Paste your API URL into the “weatherAPI” variable in the GetWeatherData lambda function. (Skip to #4 if you used Open Weather API).

2. Scroll down to line 31

```
30
31      cursor.execute('insert into WeatherData(ForecastDateTime, Speed, Direction) Values(%d, %s, %s)', (item["dt_txt"], item["wind"]["speed"], item["wind"]["deg"]))
32      conn.commit()
```

3. Modify the second part of the SQL statement to match the actual key values from your API.

```
Date,          Wind Speed ,          Wind Degree

, (item["dt_txt"], item["wind"]["speed"], item["wind"]["deg"]))
```

4. Click on save