

Build a Pizza Bot using Microsoft Bot Framework

Open Lab | Digital Summit '19

Miracle Innovation Labs

Miracle Software Systems, Inc.

Build a Pizza Bot using Microsoft Bot Framework

Introduction

This document contains a step-by-step process to take you through the various options of Microsoft Bot Framework and will teach you how to create a NLU Bot which can be used to place pizza orders.

This guide was prepared by [Miracle's Innovation Labs](#).

Pre-Requisites and Installations

All attendees must have their workstation (with Internet) to participate in the lab (Both PC and MAC are compatible). The following pre-requisites will help you to make the Hands-on Lab experience easier.

- Create a **Microsoft Account** (Will be needed for using the Bot Framework and LUIS)
- Install the following items and ensure that you are able to access them from your command line
 - **Node JS** - <https://nodejs.org/en/download/>
 - **NPM** - Should come along with Node JS
 - **Git(Optional)** - <https://git-scm.com/downloads>
 - **ngrok** - <https://ngrok.com/download>
- Create Telegram Account to access bots - <https://web.telegram.org>

Technology Involved

- Server Side - Node JS
- NLP - LUIS
- Middleware - Microsoft Bot Framework

Lab Steps

So, let us get started with the bot!

The following steps will outline how you can create a Pizza Bot and integrate it into Telegram. Users will be able to directly converse with your bot and perform operations such as ordering a pizza of their choice.

Step #1 | Creating a Static Bot with MS Bot Framework

The first step will be to use the Bot Builder SDK for Node JS and build a static bot that can respond to user messages. We will then first simulate the bot with the Bot Emulator and then register your bot in MS Azure using Bot channel registration.

Create a folder name **Sample** for your bot backend and then navigate into that folder (Sample) in the command prompt. Initialize your node project using the [npm init](#) command.

```
npm
C:\Users\vnadendla\Desktop\Harshitha_Ds'18\Main Code\Bot>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help json' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (sample) sample
version: (1.0.0)
description:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\vnadendla\Desktop\Harshitha_Ds'18\Main Code\Bot\package.json:

{
  "name": "sample",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "botbuilder": "^3.14.0",
    "luis-sdk": "^1.0.1",
    "moment": "^2.22.2",
    "ping": "^0.2.2",
    "restify": "^7.2.3"
  }
}
```

Add the required node modules using the following command,

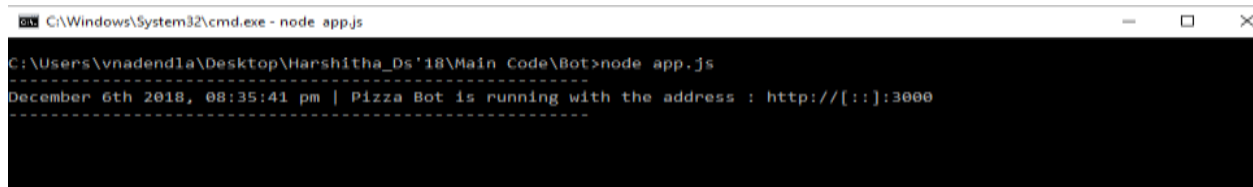
npm install --save botbuilder@3.13.1 restify moment

Create an **app.js** file in your **Sample** folder and paste the following code to handle the user's messages and send the responses.

```
var restify = require('restify');
var builder = require('botbuilder');
var server = restify.createServer();
server.listen(process.env.port || process.env.PORT || 3000, function () {
  console.log('%s listening to %s', server.name, server.url);
});
var connector = new builder.ChatConnector({
  appId : process.env.MICROSOFT_APP_ID,
  appPassword : process.env.MICROSOFT_APP_PASSWORD
});
```

```
server.post ('/api/messages', connector.listen ());  
var bot = new builder.UniversalBot(connector, function (session) {  
    session.send ("Hello world");  
})
```

To run your Node JS code, open the command prompt and move to the app directory and enter the command **node app.js**

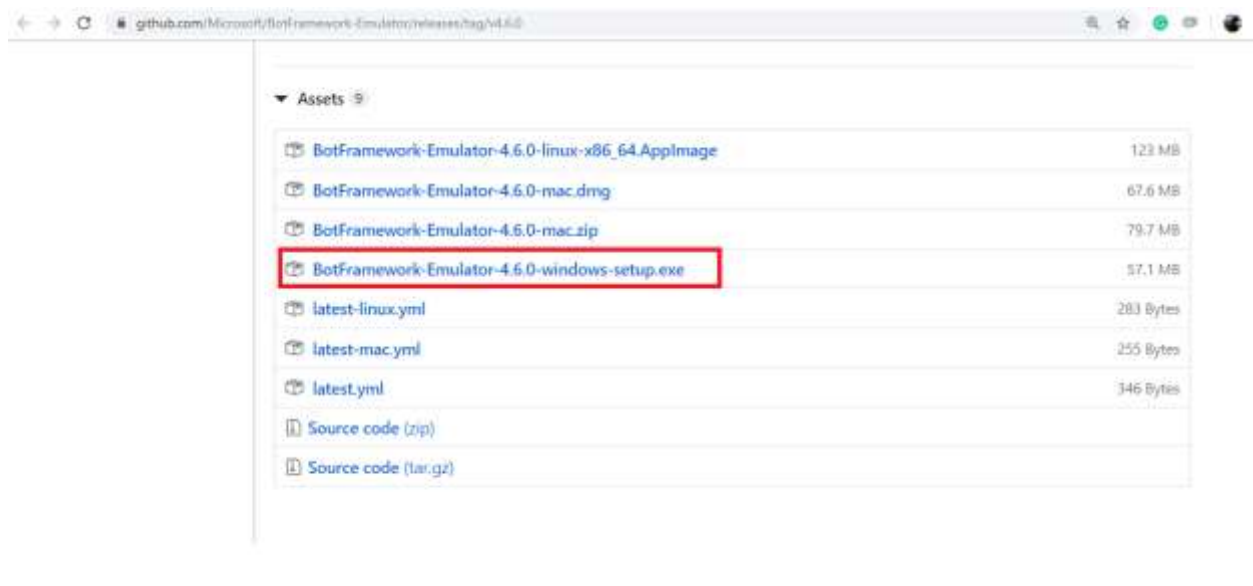


```
C:\Windows\System32\cmd.exe - node app.js  
C:\Users\vnadendia\Desktop\Harshitha_Ds'18\Main Code\Bot>node app.js  
December 6th 2018, 08:35:41 pm | Pizza Bot is running with the address : http://[::]:3000
```

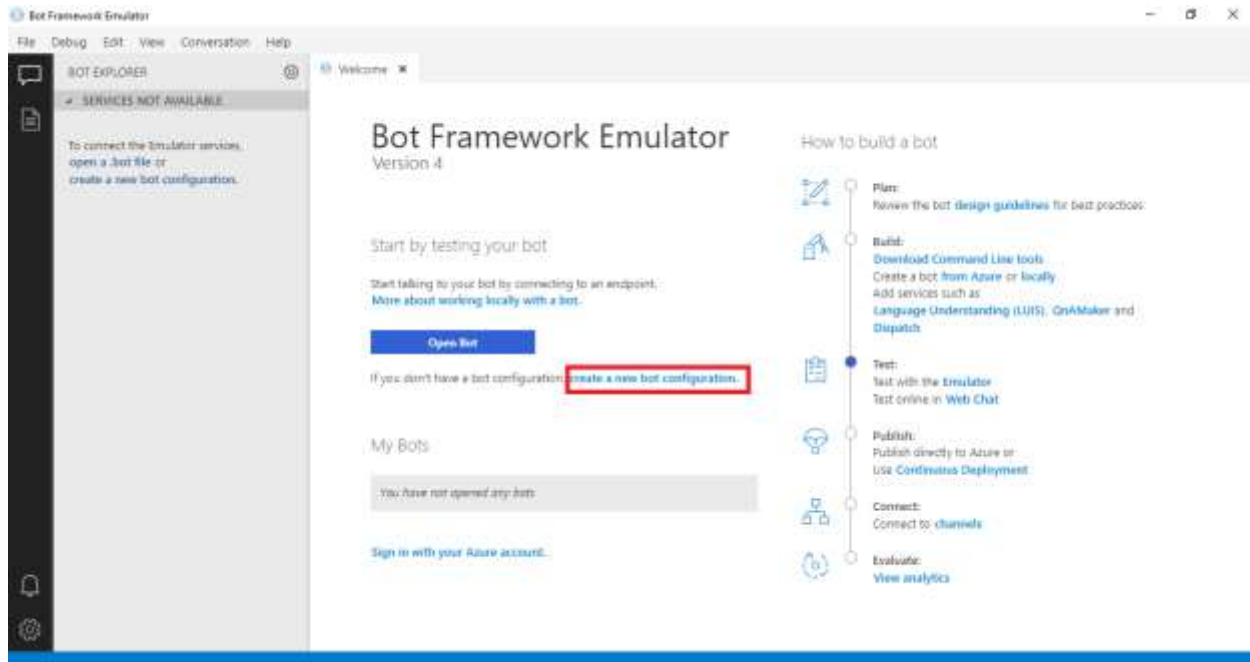
Download **Bot Emulator** - (Download .exe file from the below link)

<https://github.com/Microsoft/BotFramework-Emulator/releases/tag/v4.6.0>

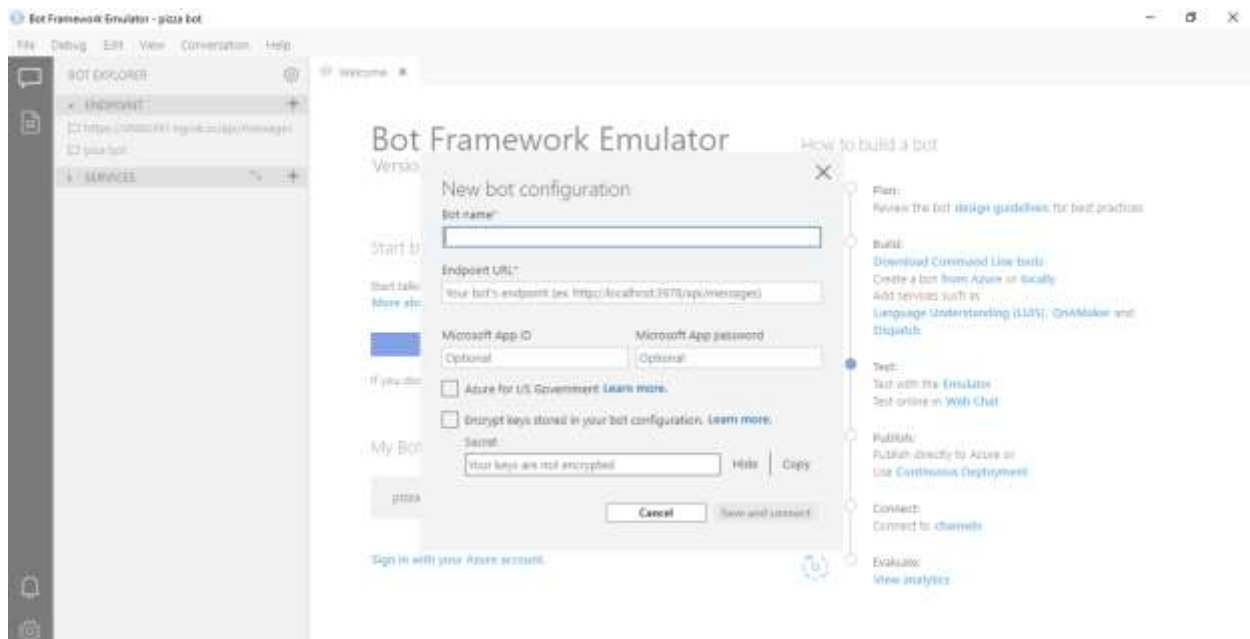
and then select the **BotFramework-Emulator-4.6.0-windows-setup.exe**



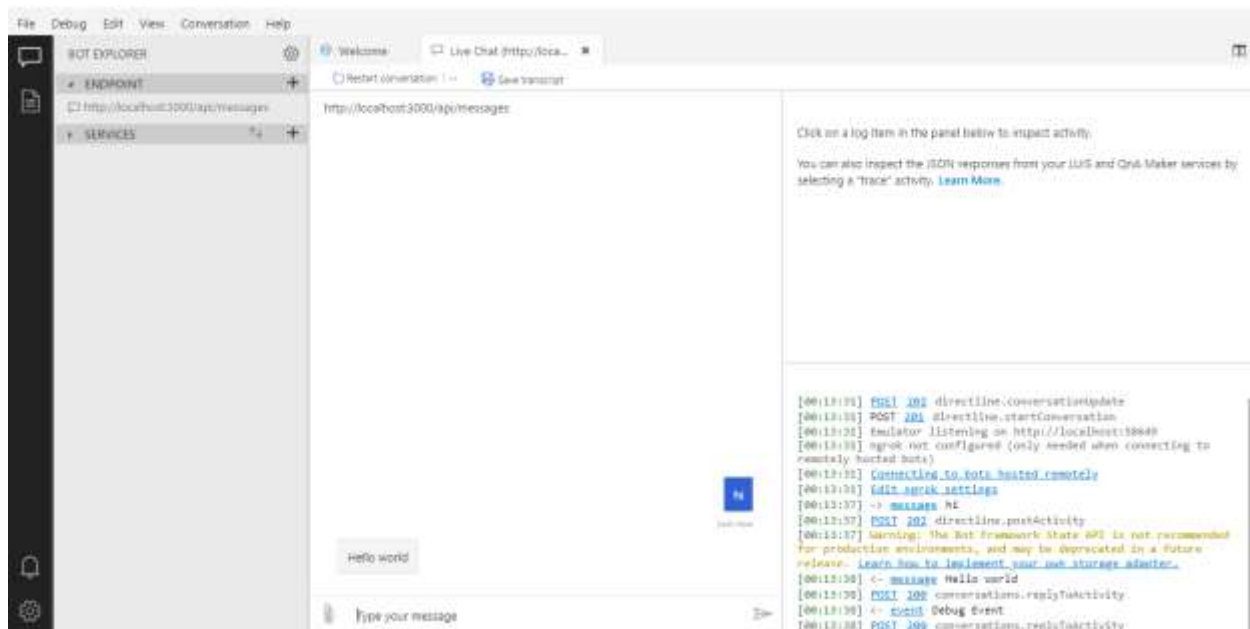
Open your Bot Emulator and click on **create a new bot configuration**.



Enter Bot name and Endpoint URL fields and click on **Save and connect**. You need to save the bot configuration file at root folder location.

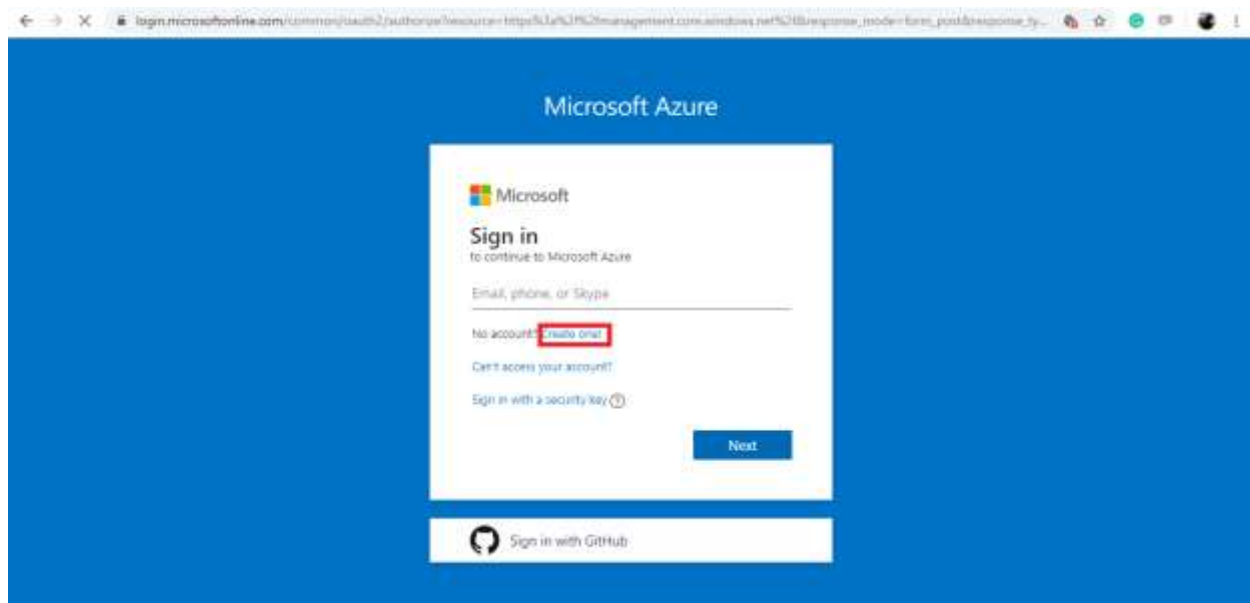


You can then send a message and receive the **“Hello World”** response.

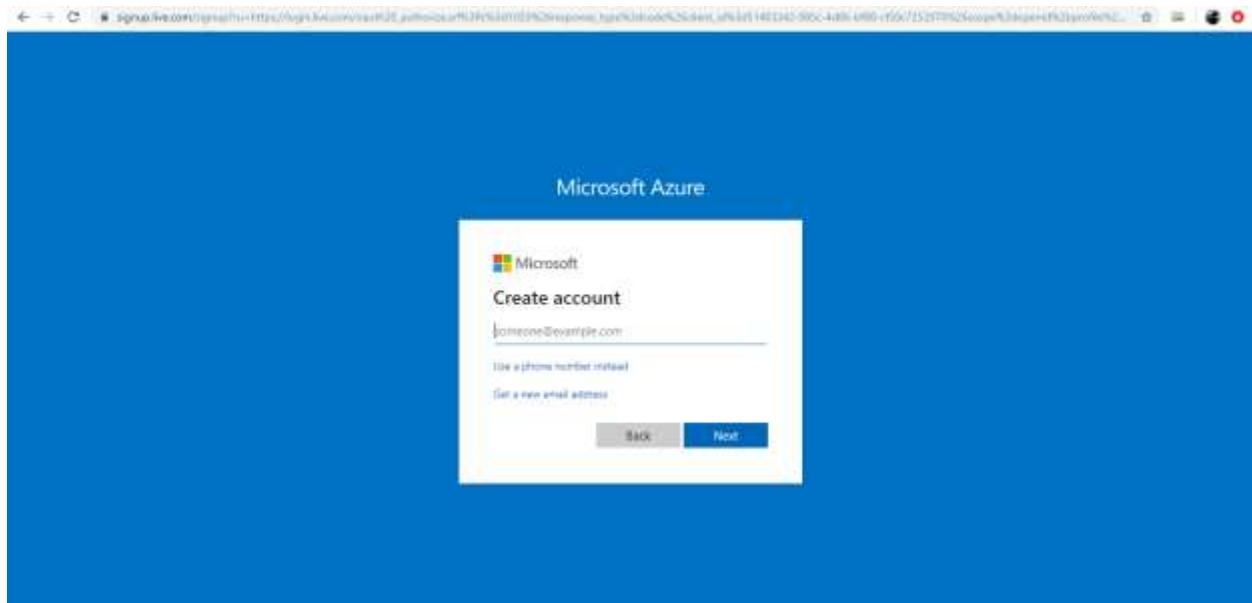


Step #2 | Register and Test with the Bot Framework

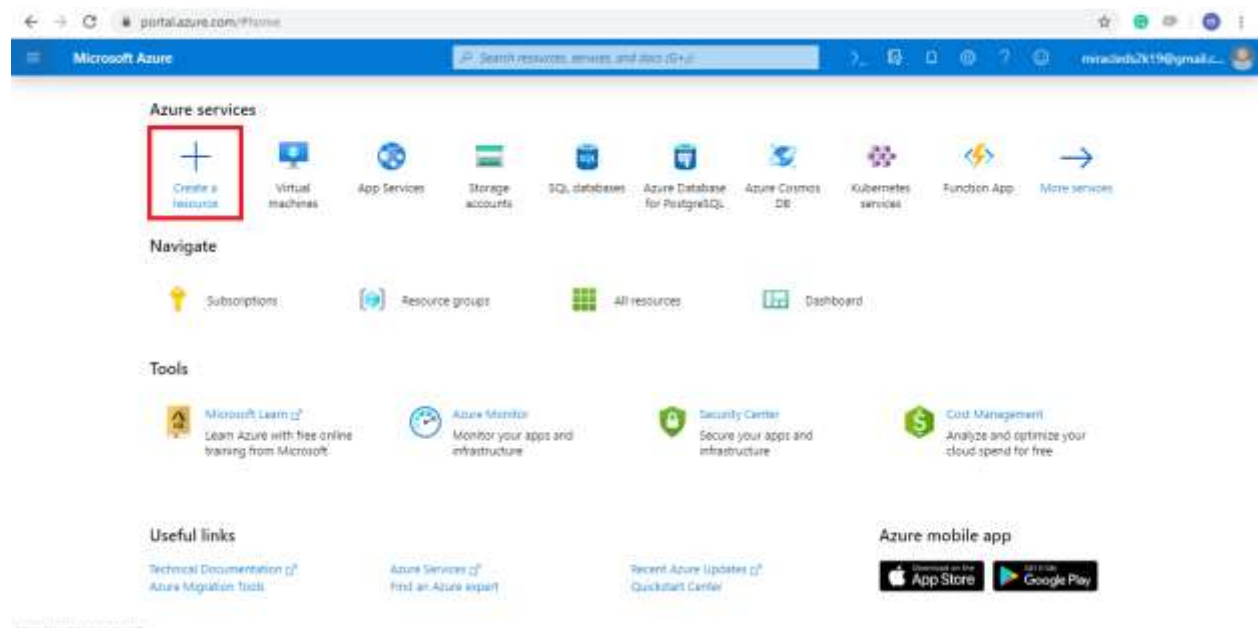
Log into <https://portal.azure.com> with your Microsoft account. If you don't have an account click on **Create one** and create your azure account.



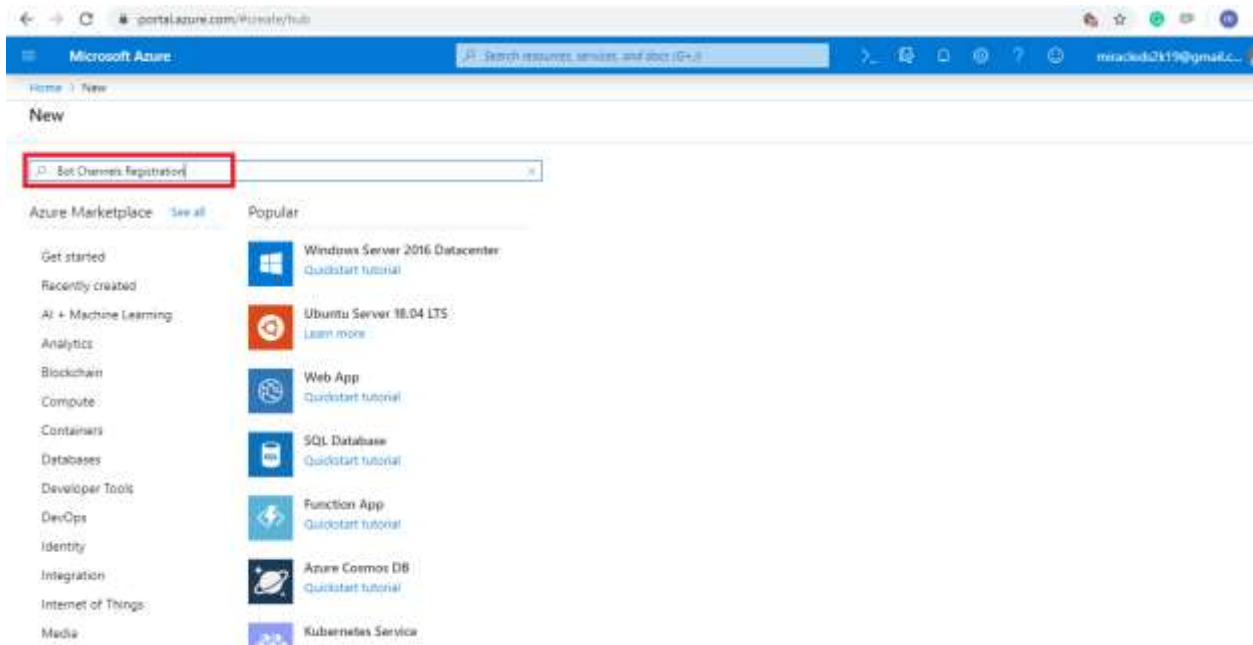
Provide your email and required details as well as you can set password for the new account,



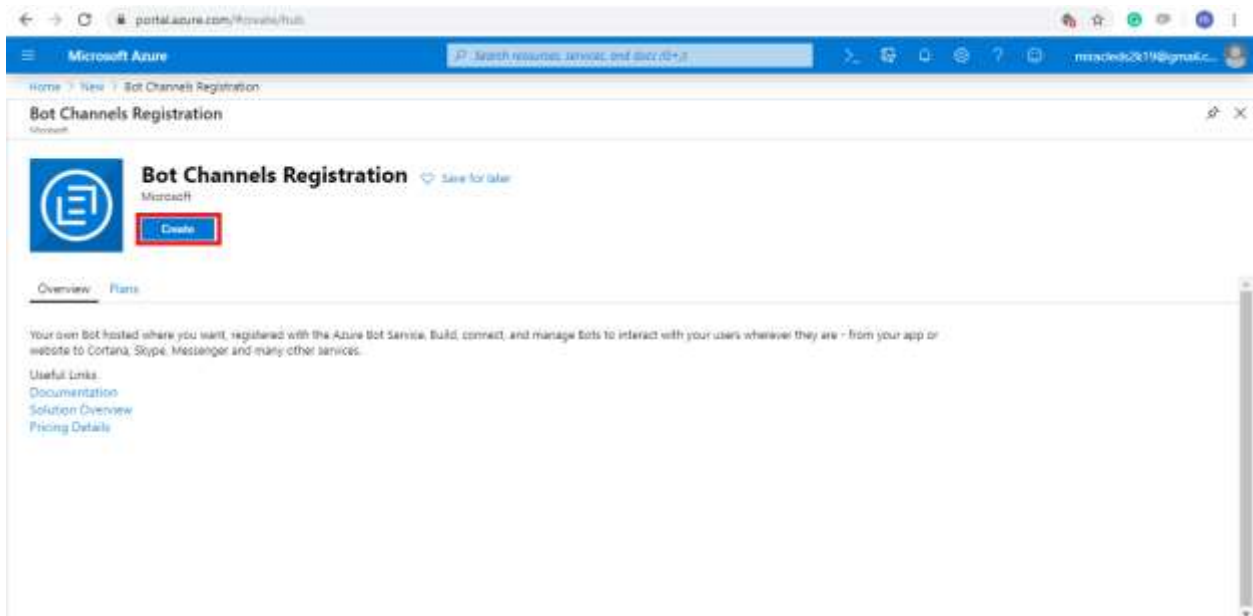
Once you created an account you can logged in to new account then you can find Azure dashboard. Click on **Create a resource**



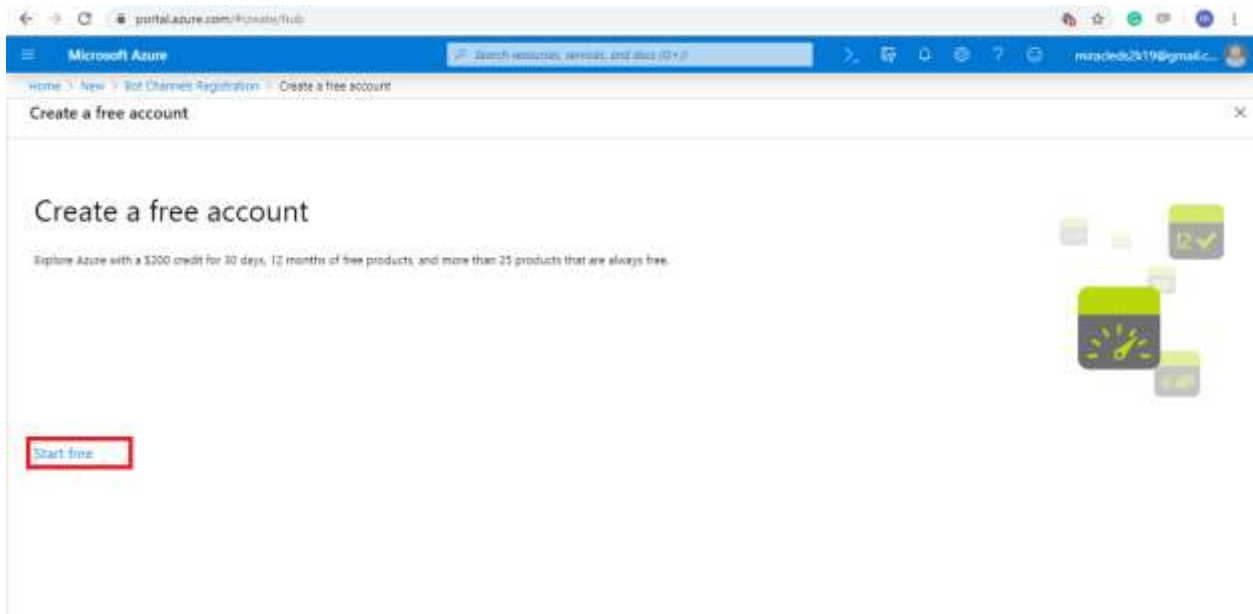
Now search for **Bot Channels Registration** service as shown in below,



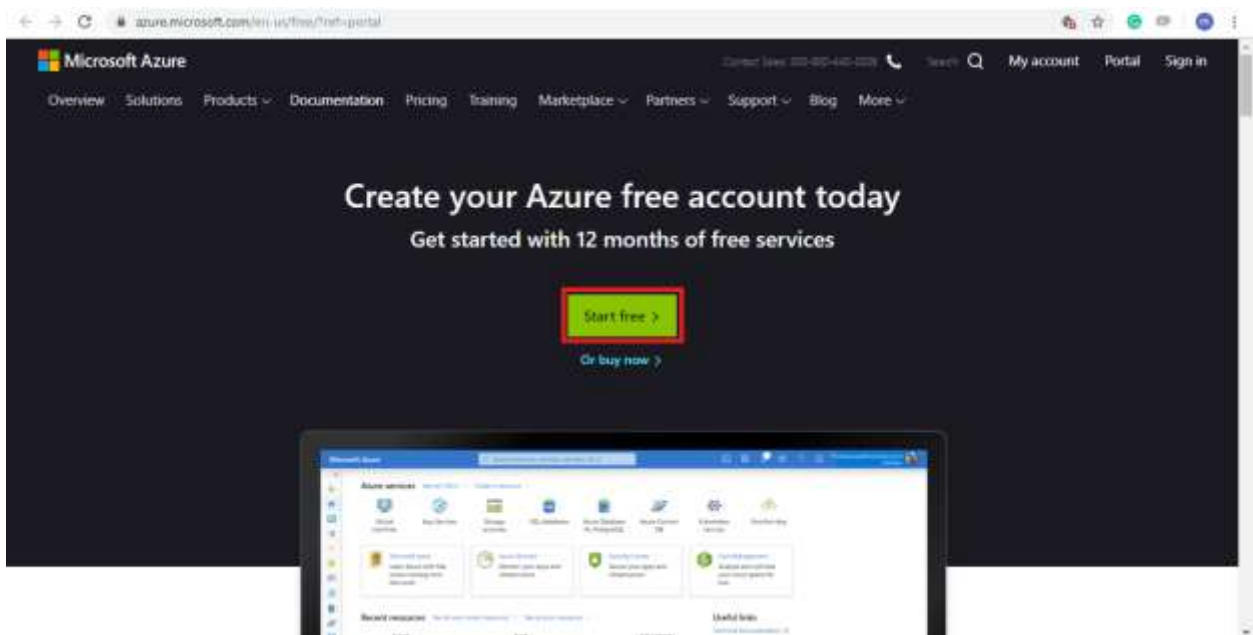
Click on **create** to create the service,



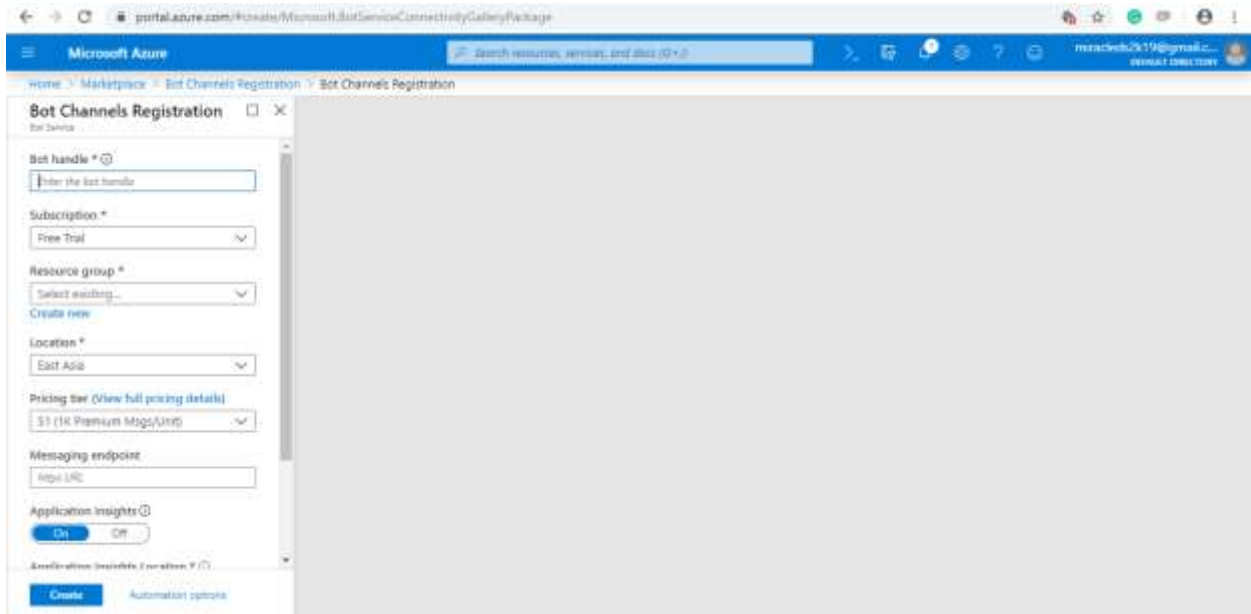
Now you have to create a free account to enable a service,



Now you will be redirect to the page as follows, click on **start free** button.

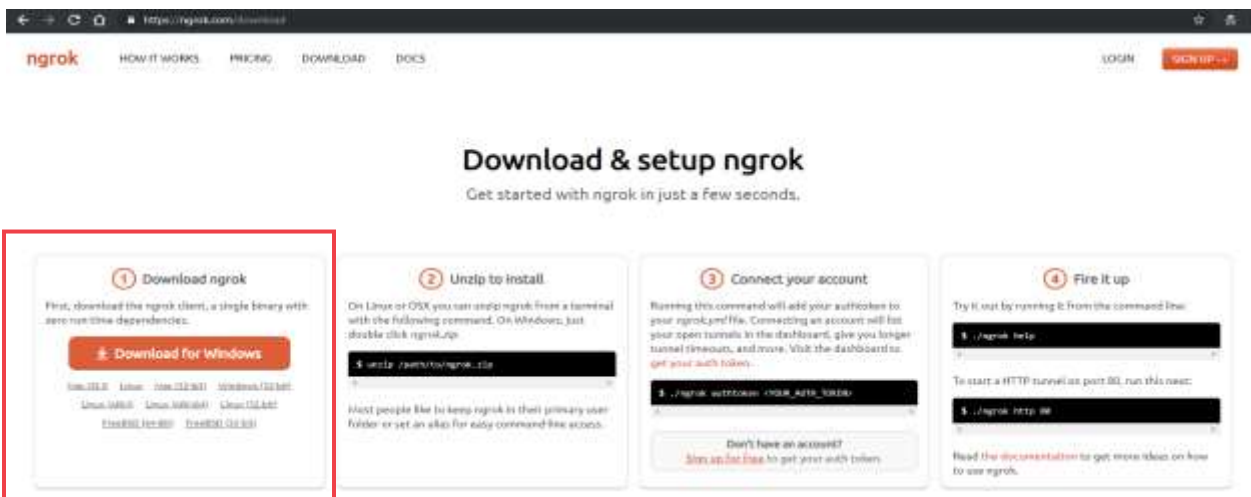


Now fill the required details and then go to the **Bot Channels Registration** by providing your credit or debit card details.



In the place of messaging endpoint URL, you need to place **ngrok** URL. For that open your browser and download the ngrok by using the below link <https://ngrok.com/download>

You need to install based on system requirement shown below,



After downloading the ngrok, extract the downloaded folder and run the **ngrok.exe** file. Now, enter the following command and click Enter button, **ngrok http <your-application-port-number>**

Example : ngrok http 3000

```
C:\Windows\System32\cmd.exe - ngrok http 3000
ngrok by @inconshreveable (Ctrl+C to quit)

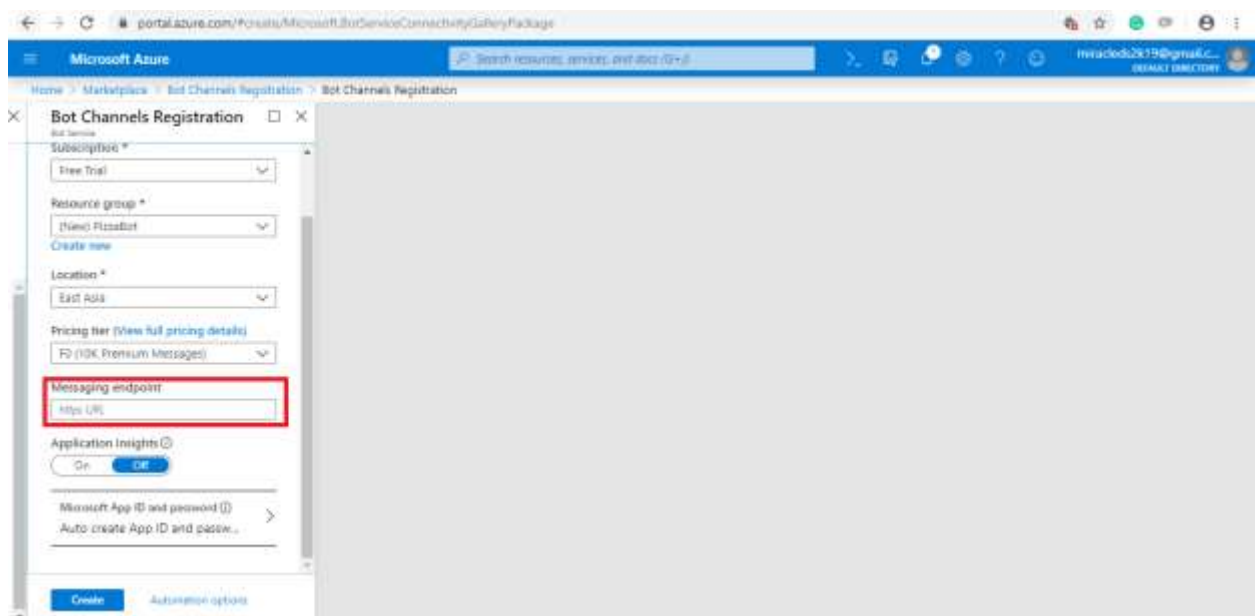
Session Status      online
Session Expires     7 hours, 59 minutes
Version             2.2.8
Region              United States (us)
Web Interface        http://127.0.0.1:4041
Forwarding           http://239e16e6.ngrok.io -> localhost:3000
                    https://239e16e6.ngrok.io -> localhost:3000

Connections
  ttl    opn    rt1    rt5    p50    p90
   0      0     0.00   0.00   0.00   0.00
```

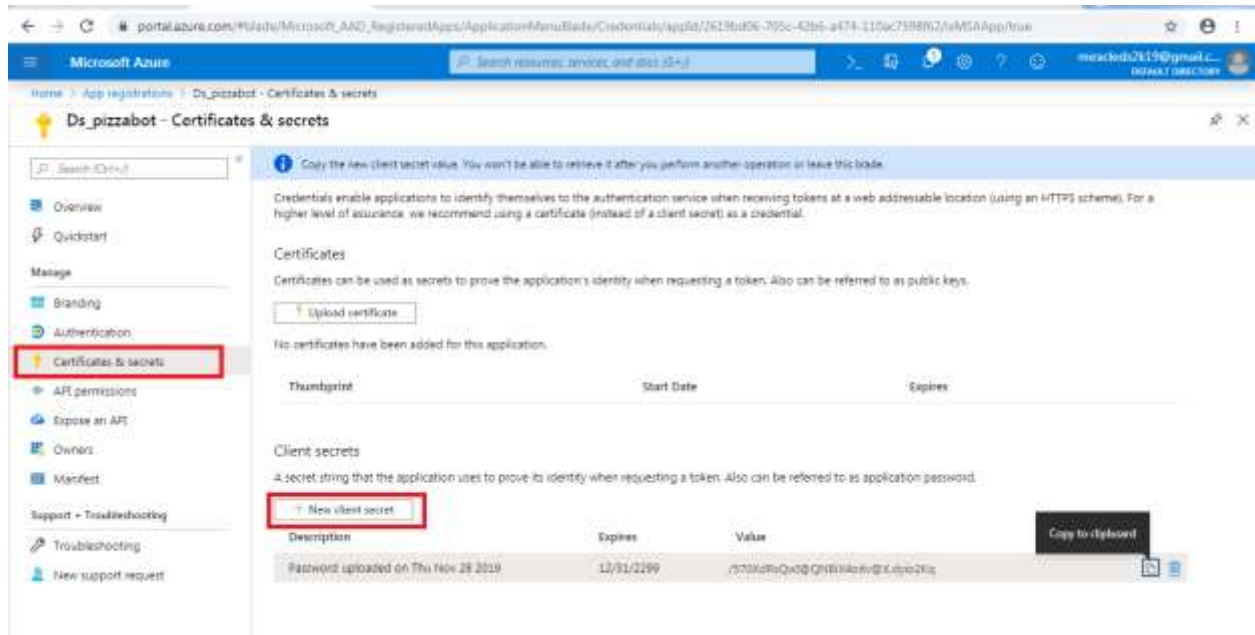
Take the above endpoint URL and place at messaging endpoint field in portal as shown below. Now, you need to attach [/api/messages](#) to the end point URL.

Example : <https://<random-code>.ngrok.io/api/messages>

After configuring the messaging endpoint, click on create button.

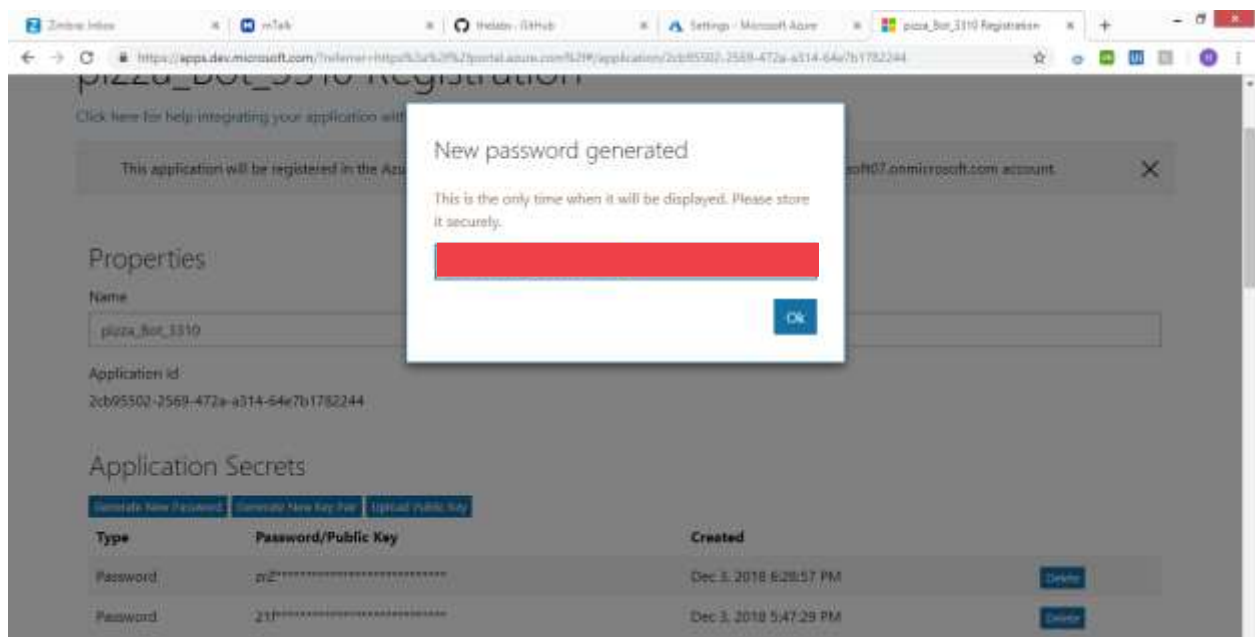


You have successfully registered your bot. In Settings, you will get **App ID**. Save this for later use. Now, go to **App registrations** and then find the service which you have created. In that, select **certificates & secrets** to generate a client secret by clicking on **New client secret**



Go to **Application Secrets** and you will find **Generate New Password**. Click on that button.

Note : Please save the password on your notepad for later use.



Now, go ahead and run your server once again with command,
node app.js

```
C:\Windows\System32\cmd.exe - node app.js  
C:\Users\vnadendia\Desktop\Harshitha_Ds'18\Main_Code\Bot>node app.js  
December 6th 2018, 08:35:41 pm | Pizza Bot is running with the address : http://[::]:3000
```

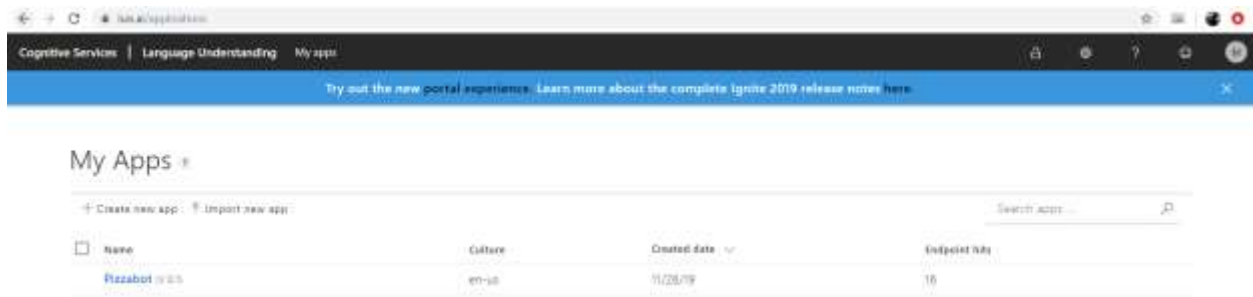
Step #3 | Creating your LUIS Model

Our bot is pretty cool now, but it would be cooler if we can add Natural Language Understanding capabilities. In this lab, you will add intents and entities to your LUIS Model and test them out.

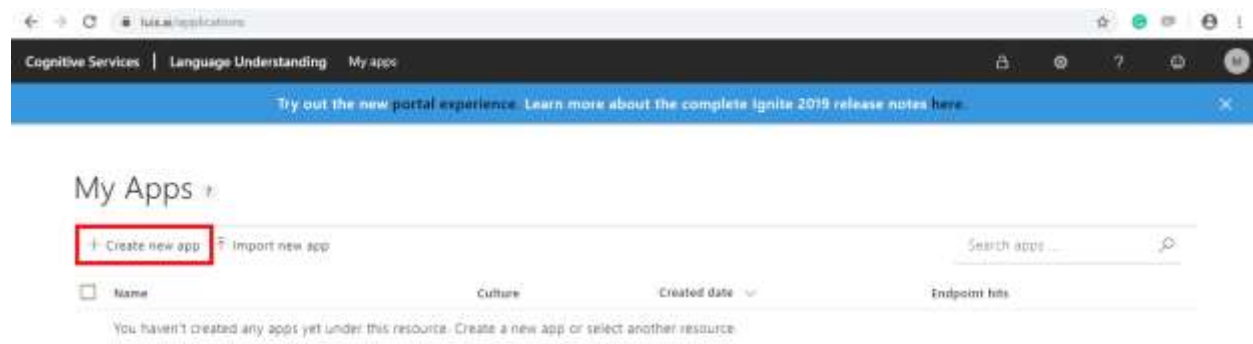
First, head over to the LUIS homepage at <http://www.luis.ai> where you will be making your dialog model. Click on **Sign In** to navigate into LUIS dashboard



Once you click on **Sign In** it will be redirected using your Microsoft account. Now you can see the trained apps in the dashboard as below,

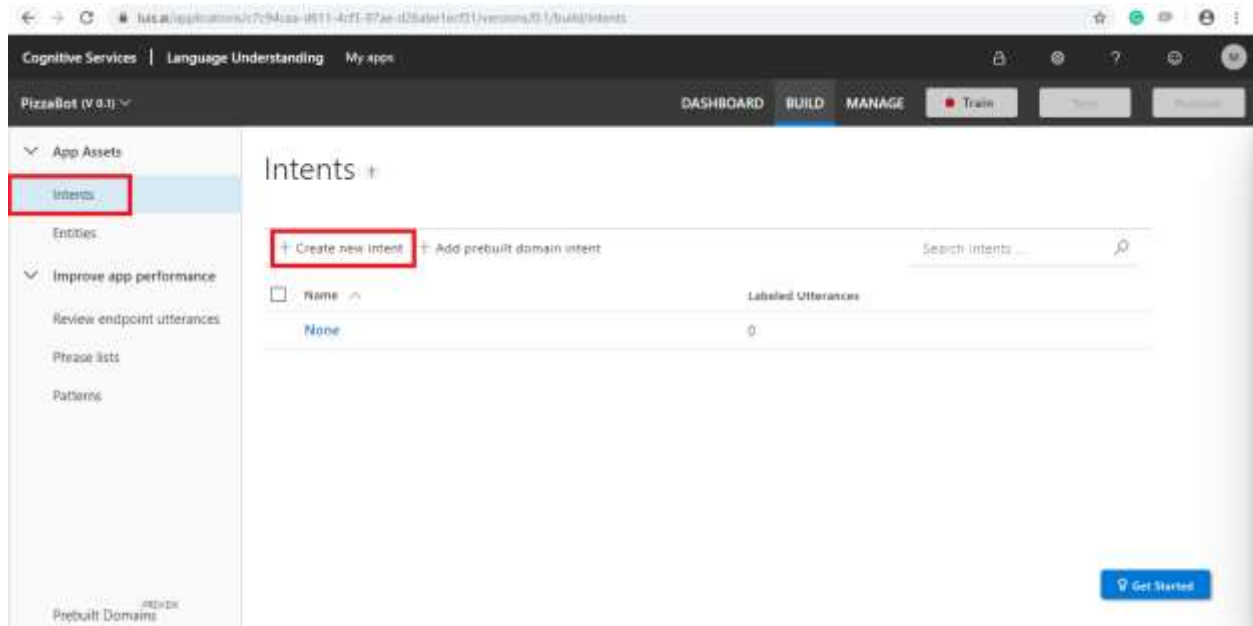


If you are not having any previous apps, then you can create new app for your Natural Language dialog model. Under My Apps you can find **Create new app**. Click on **+Click new app**

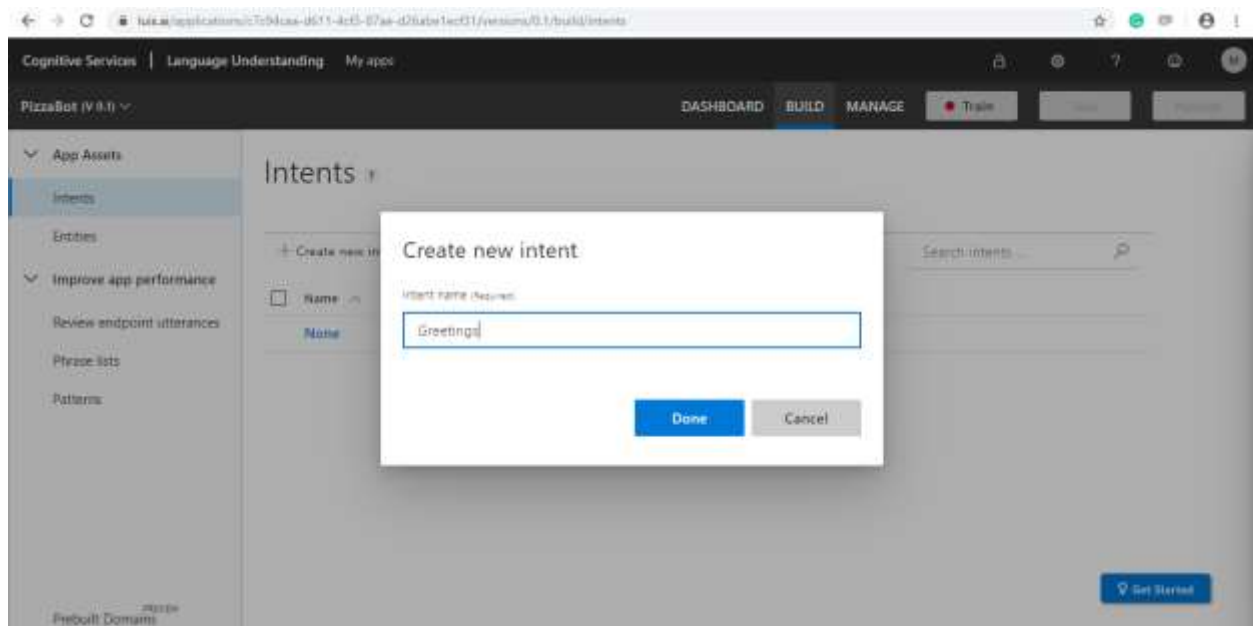


After you have create an app providing a name, your first step will be to create and train an intent. You can think of an intent as an action your end user wishes to carry out that is processed and executed by your bot, either by replying to the user with relevant dialog and information, or triggering your backend application to perform certain task.

To begin creating intent, click on the **Intents** tab under your apps dashboard, and select **+Create new Intent**.



We will begin by adding a simple intent that responds with the Greetings of the bot when the end user is facing an issue or has an off topic question. Give the Intent name - “Greetings” to describe what the intent does, and click on **done**.



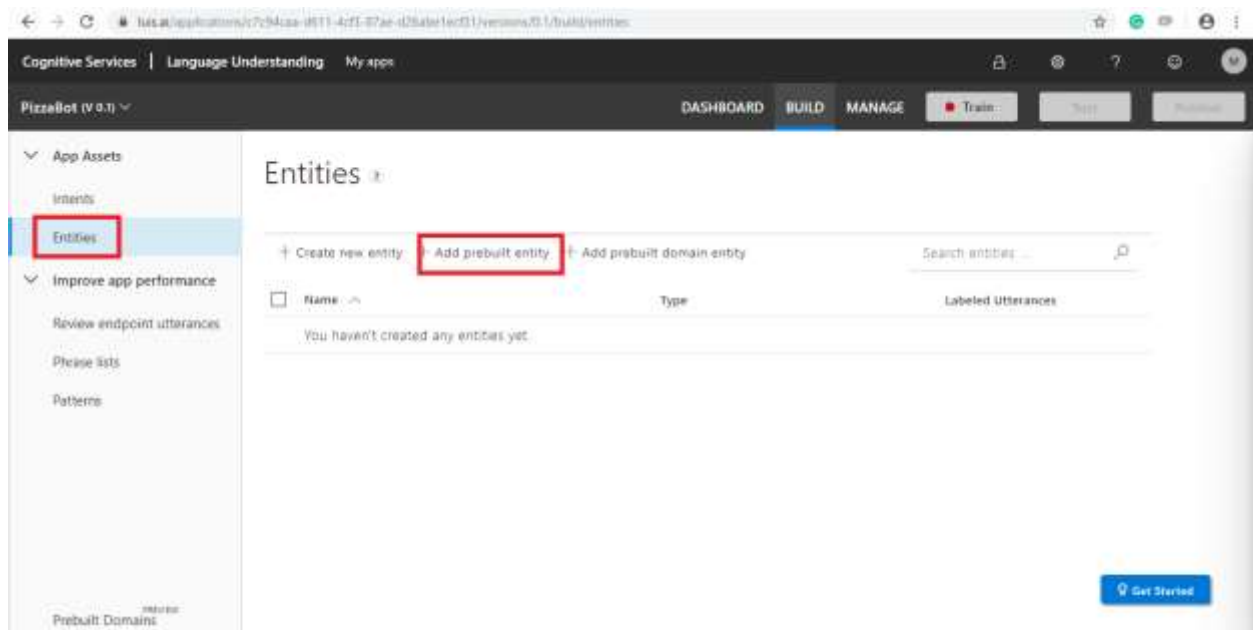
You will then need to train your Greetings intent on example inputs or utterances the user might enter when wanting to know what your bot can do.

Try adding a few examples like,

- Hi
- Hey
- Hello
- Hola

Now it is time to create your first entity. An entity can be thought of as a label for an object, topic, or key identifier for something specific an end user wants to perform an action on.

We will be using prebuilt entity that contains number which is related to single entity type. Next, click **+Add prebuilt entity**.



Now select **number** from the list of prebuilt entities. To finish your prebuilt entity, click on **Done** and LUIS will add it to your application.

Add prebuilt entities

When you add a built-in entity, its predictions will be available to you while labeling utterances.

☒

number
A cardinal number in numeric or text form
ten, forty two, 3.141, 10K

☐

ordinal
An ordinal number in numeric or text form
first, second, tenth, 1st, 2nd, 10th

☐

temperature
A temperature in celsius or fahrenheit
32F, 34 degrees celsius, 2 deg C

☐

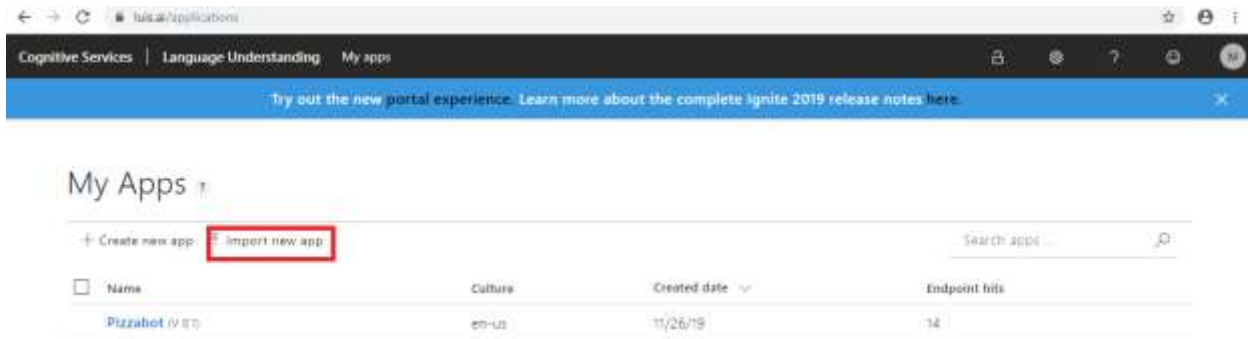
dimension
.....

Done

Cancel

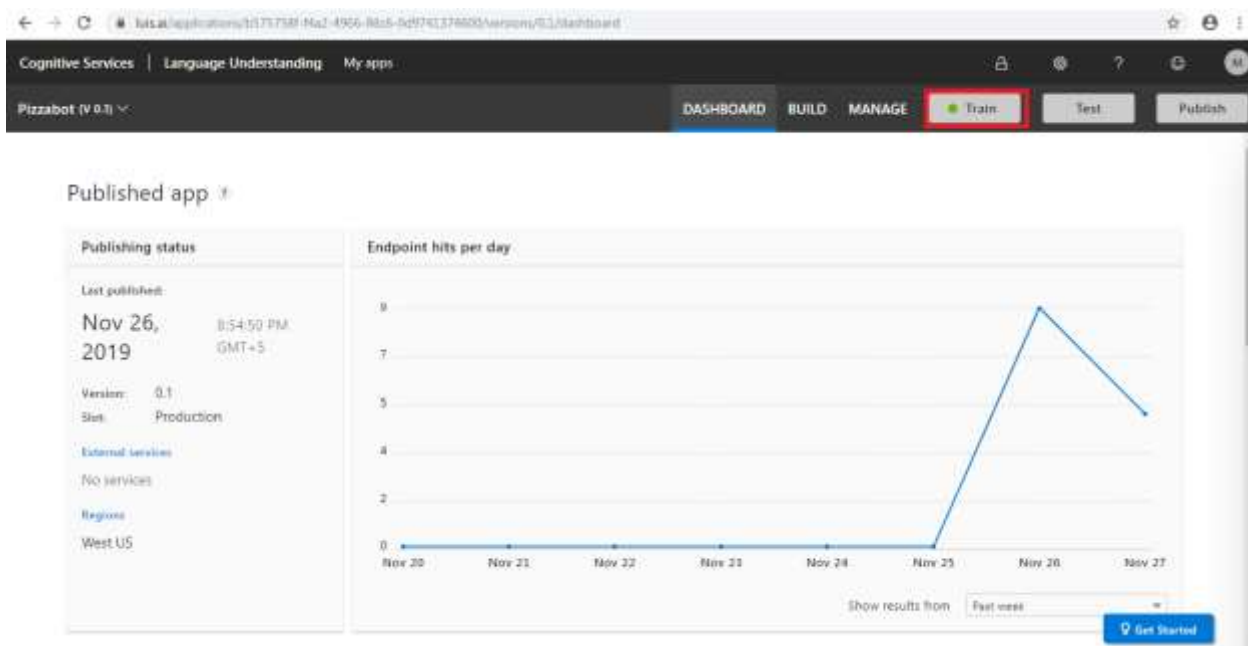
Now that you are familiar with creating an Intent and Entity, import our finished dialog model to LUIS, where you will train and publish the finished app.

To add the finished model, open the `dialog_model.jsonfile` which is available in the downloaded folder.



Step #4 | Train, Test and Publish your LUIS App

Now that you have a completed LUIS model, you will need to train your application on your example utterances to create the model used for identifying your Intents. To do this, select **Train** at the top-right menu, and click on **Train Application**.



LUIS will then create the model used for handling Natural Language. Make a note that you will need to re-train your model when entering new intents, entities, or example utterances to have your bots dialog reflect your new changes.

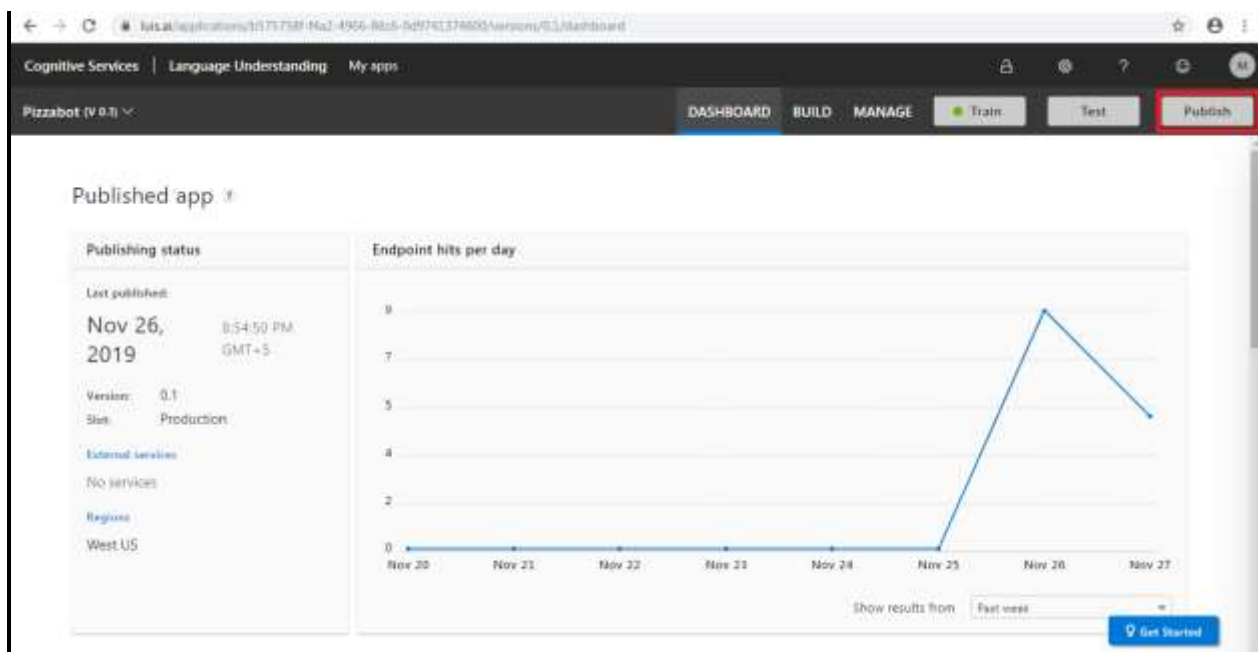
Once the training is completed, try entering the below sentences in the Interactive Testing window to see how LUIS classifies intents and labels your entities,

- Hi, how are you?
- What are your capabilities?

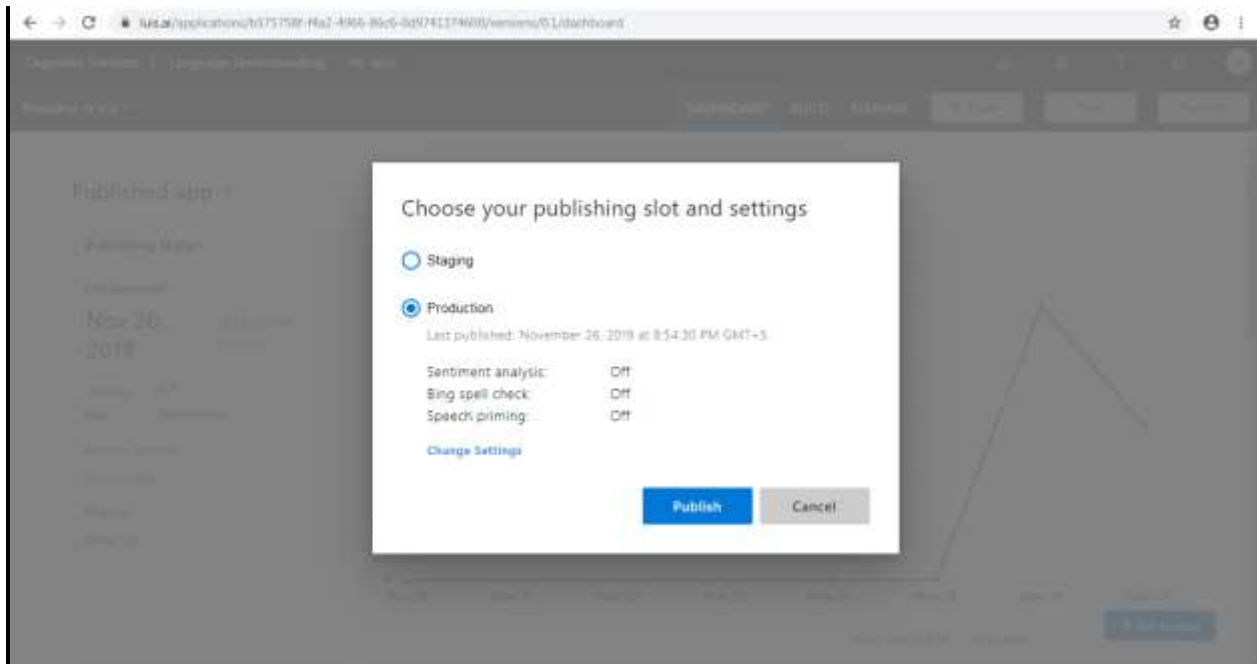
You should see the top scoring intent for each utterance and your entity keywords labeled with the entity name they fall under.

The last step in creating your LUIS model is to publish your App and create the endpoint URL for integrating into your backend.

To do this, click on **Publish** at the top-right in the menu bar.



The next step is to publish the app. Click on **Publish** button, when pop-up appears.



After a few moments your application should be published, and you will be given an Endpoint URL. Make note of your URL as later lab will need it for integrating your LUIS dialog model.

Step #5 | Adding Basic Dialog and Matching Capabilities

Navigate back to your project directory in the command prompt and run the following command,

npm install - save luis-sdk

Now, open your app.js file in your sample folder and copy the code from app.js file (/example/app.js) present in the GitHub repository and replace <bot-app-id> with your APP ID at line **#30**, replace <bot-app-pwd> with your Password at line **#31** and <luis-model-publish-url> with LUIS model url at line **#38** which you copied from previous steps.

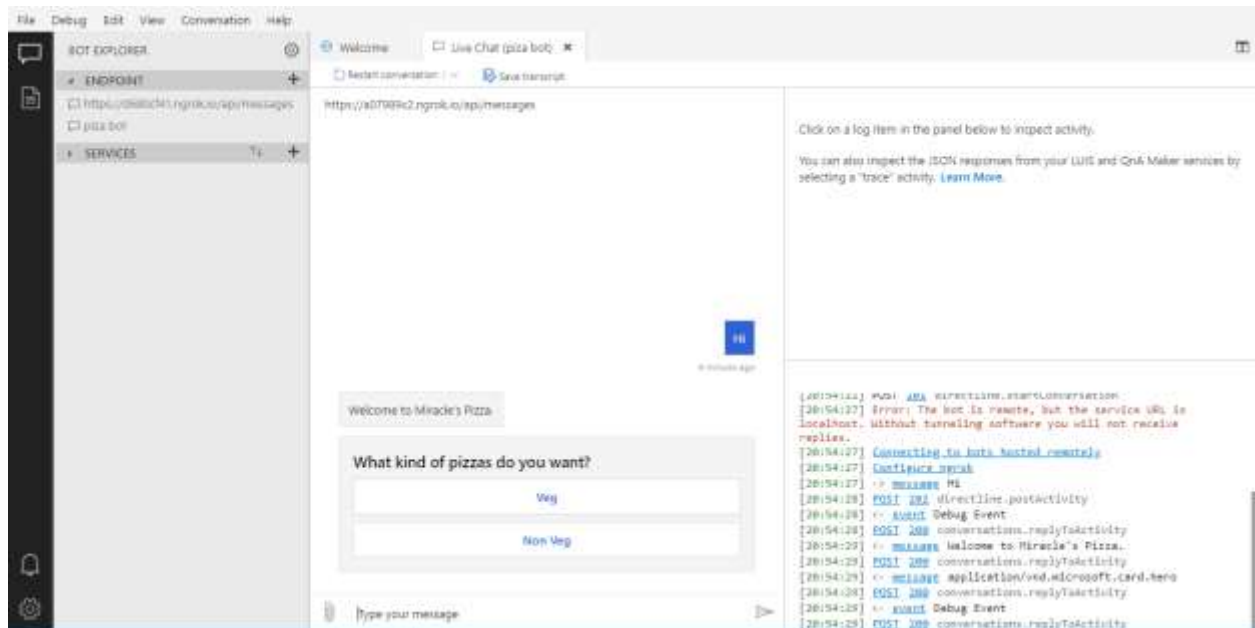
Now, save the app.js file and run your server by using command **node app.js** in your command prompt as shown below,

```
C:\Windows\System32\cmd.exe - node app.js

C:\Users\vnadendla\Desktop\Harshitha_Ds\18\Main Code\Bot>node app.js

December 6th 2018, 08:35:41 pm | Pizza Bot is running with the address : http://[::]:3000
```

Open the emulator and test the bot as shown below,

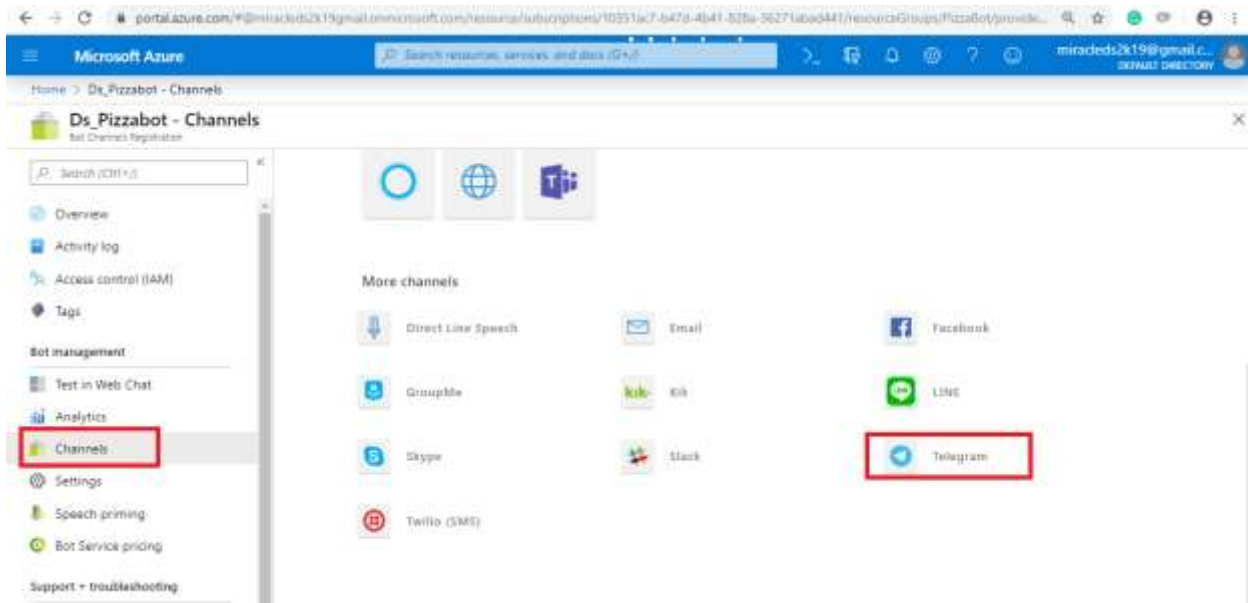


Step #6 | Integrating your Bot with Telegram

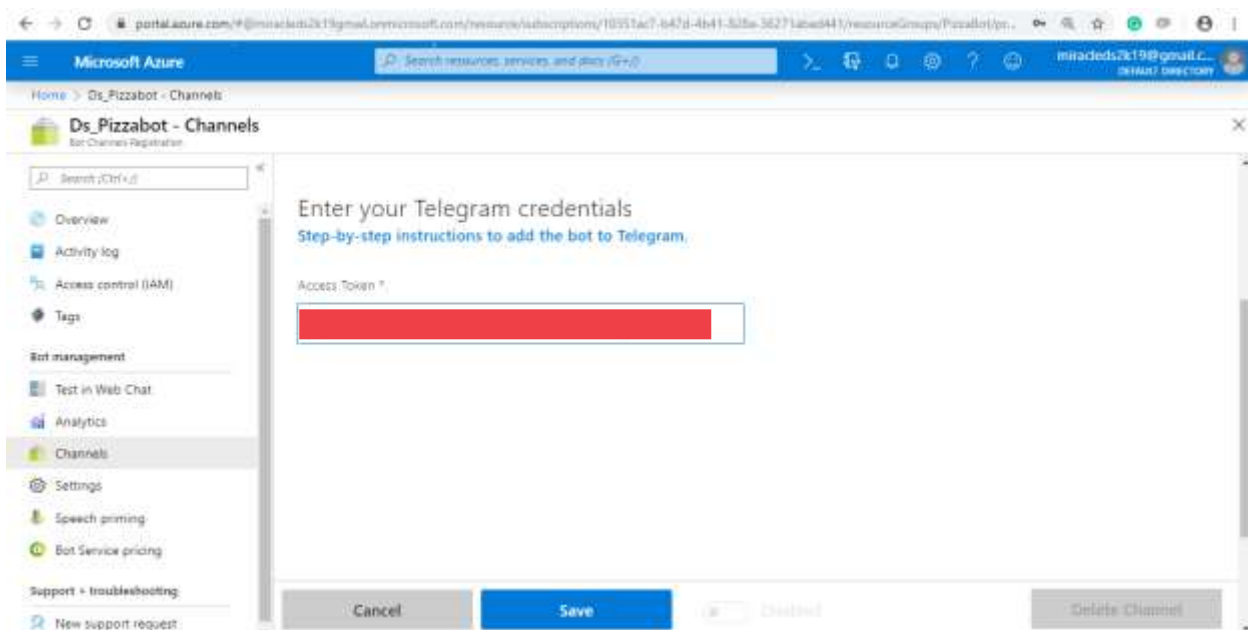
Once we have a sample bot up and running, we can now integrate with the channel of our choice - in this lab we will see how you can integrate your bot with Telegram.

To integrate your bot with Telegram, follow the below document, <https://docs.microsoft.com/en-us/azure/bot-service/bot-service-channel-connect-telegram?view=azure-bot-service-4.0>

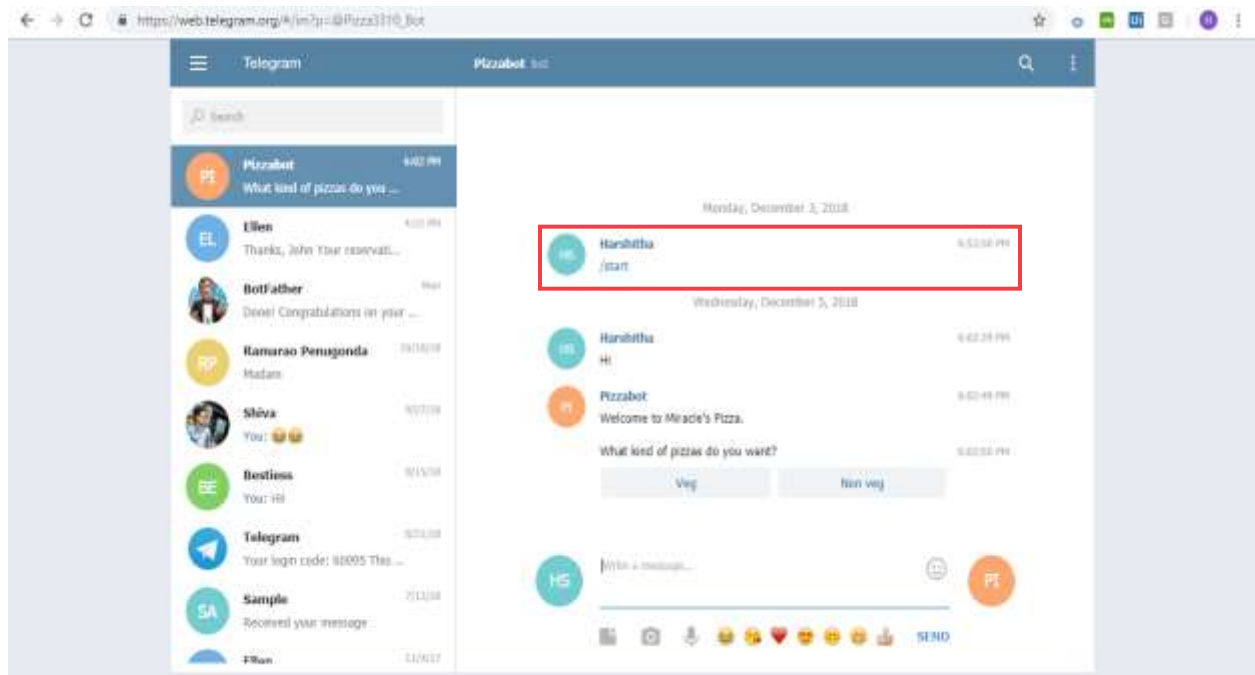
- Go to <https://portal.azure.com> and select your bot channel registration
- Click on channels, in the more channels section. Select **Telegram** icon



- Now open your telegram app and open the **Bot Father** and give **/start** and then give the bot name.
- Paste the access token that you get from the **Bot Father**(Check the above link) and click save
- There you go, your bot is ready!



Now before you start chatting with your bot send the message “**/start**” which is used to start the conversation with the bot. After sending that message you can start your conversation with the bot.



For any questions regarding the lab, please feel free to reach out to innovation@miraclesoft.com. We hope you enjoy creating bots with us 😊