**Azure BOT Development**

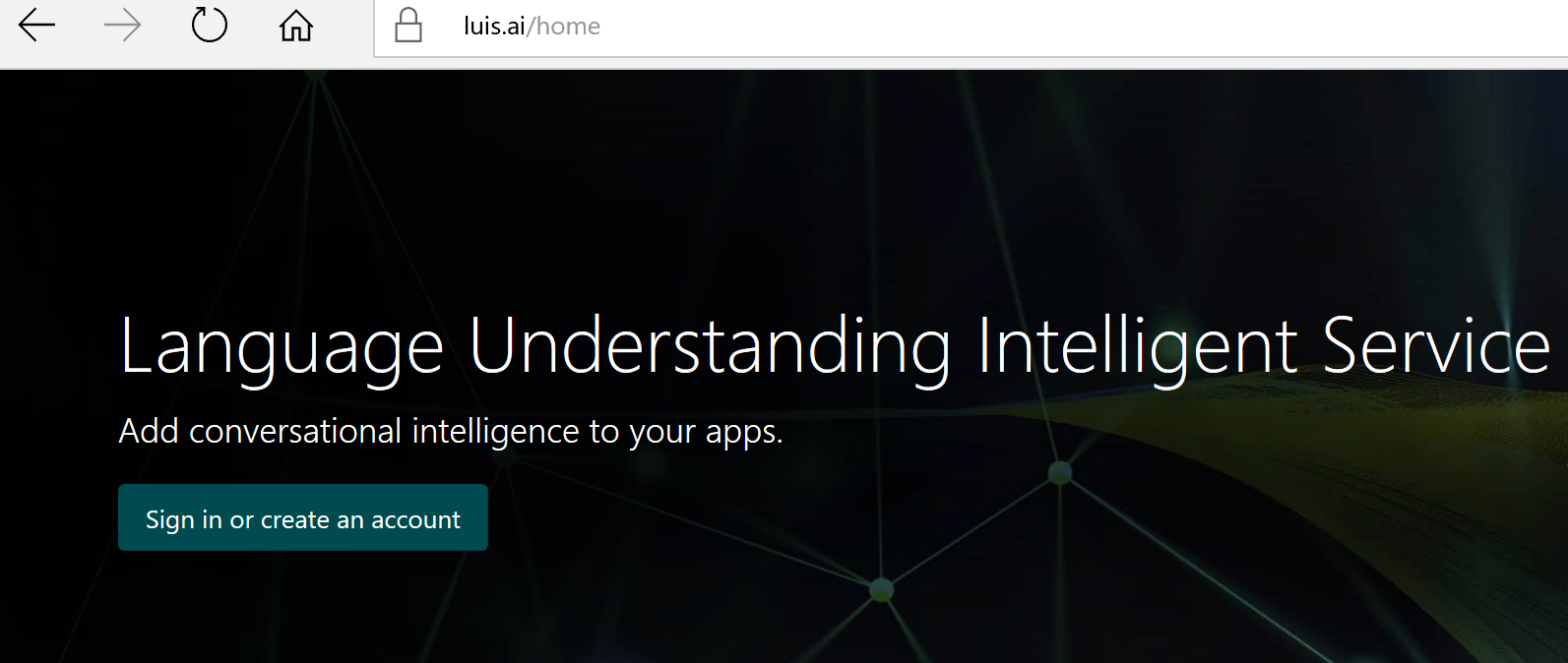
The purpose of the HOL is to help the developers to build a simple hotel reservation BOT with the help of Azure BOT Builder framework and Microsoft LUIS. When completed, you will be able to run the BOT in locally installed BOT emulator and ask questions in like “find hotels in Seattle” and “Show me the review for hotel”

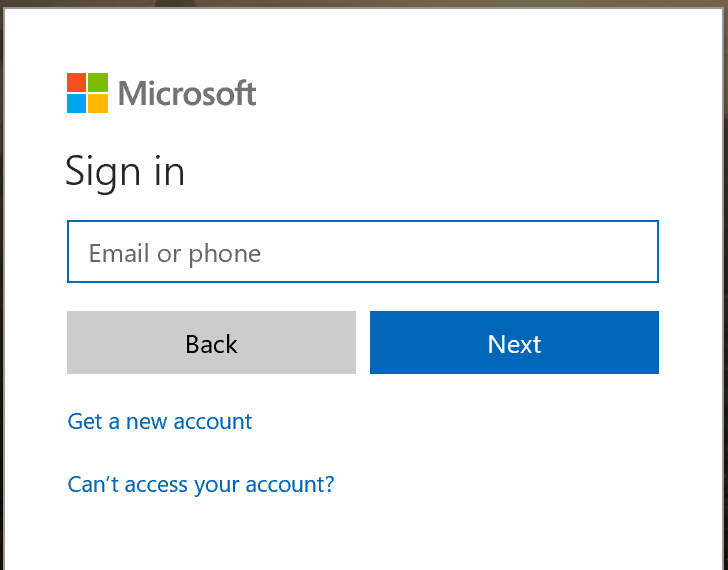
**Prerequisites**

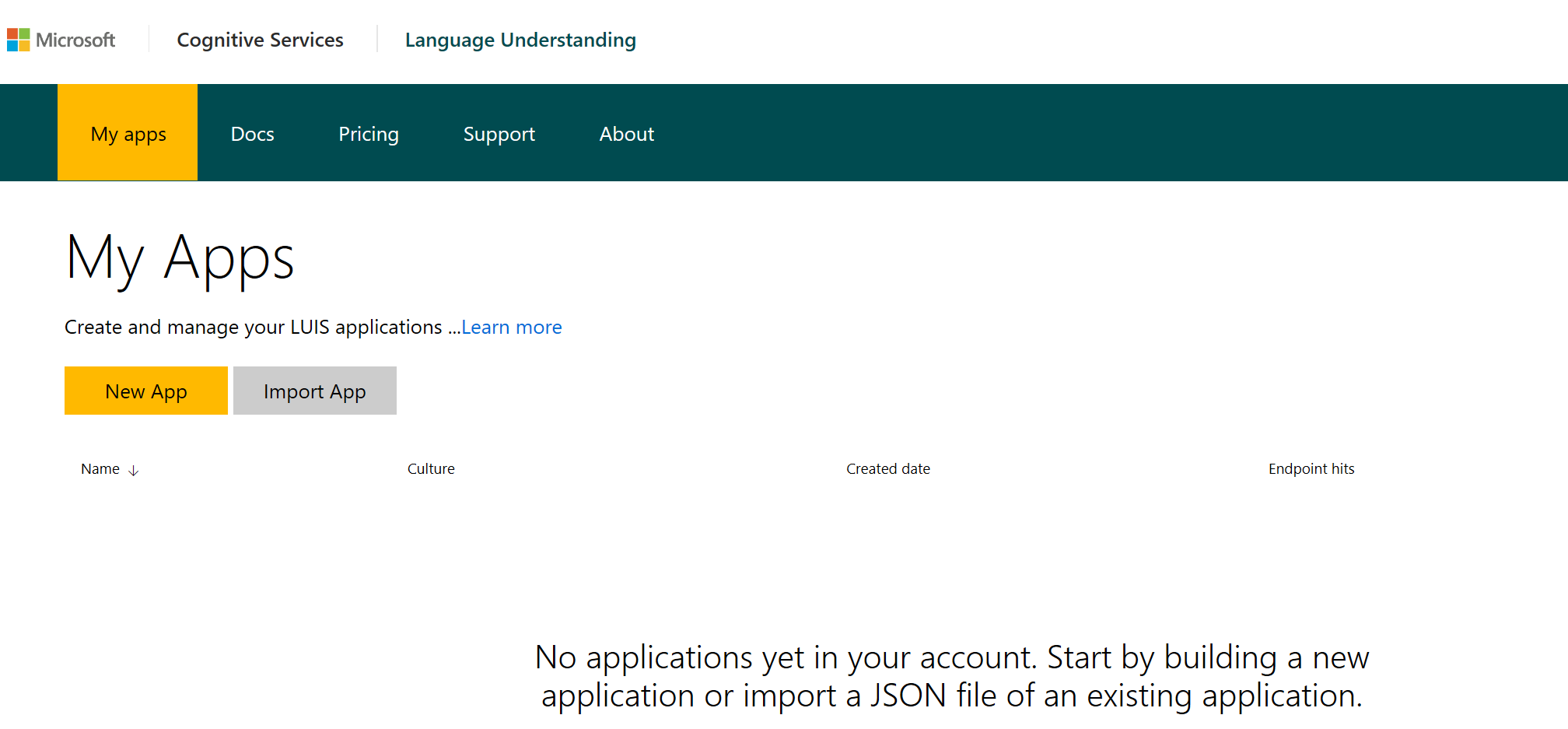
The minimum prerequisites to run this sample are given below, please ensure your developer workstation has the following components pre-installed.

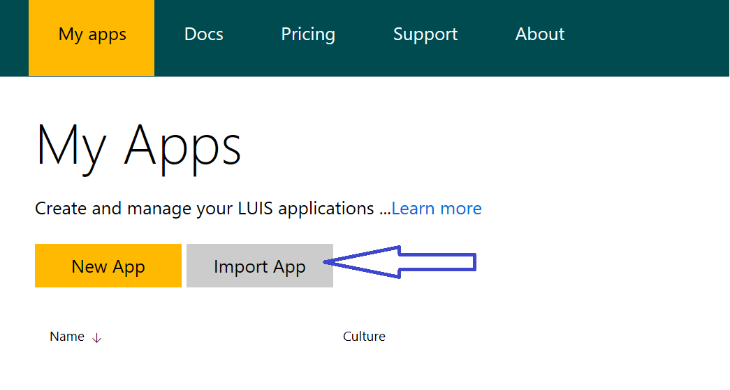
1. Latest Node.js with NPM. Download it from [here](https://nodejs.org/en/download/).
2. The Bot Framework Emulator. To install the Bot Framework Emulator, download it from [here](https://emulator.botframework.com/). Please refer to [this documentation article](https://github.com/microsoft/botframework-emulator/wiki/Getting-Started) to know more about the Bot Framework Emulator.
3. **[Recommended]** Visual Studio Code for IntelliSense and debugging, download it from [here](https://code.visualstudio.com/) for free.

**Step 1 – Building LUIS Model**

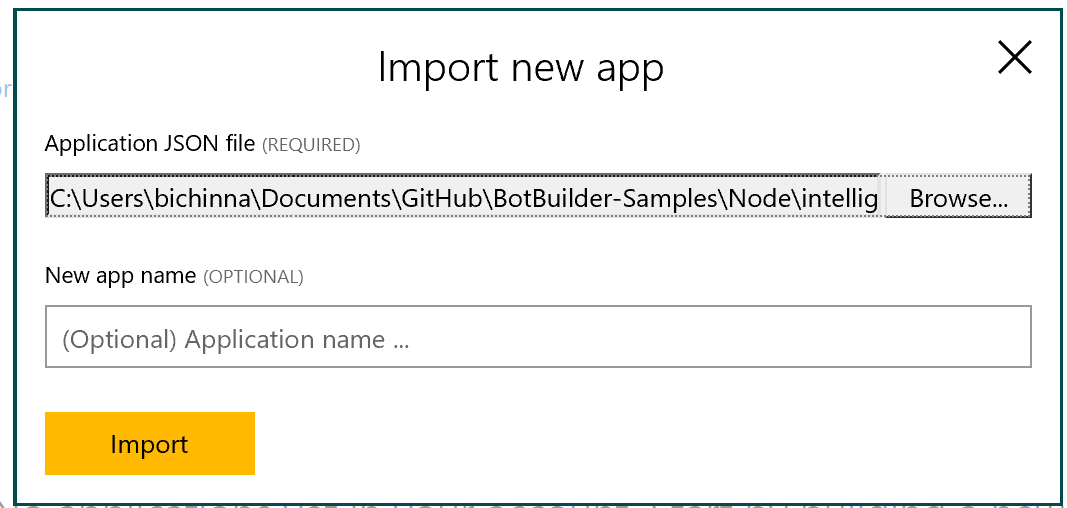
1. The first step is create an account in LUIS and create or import an existing model. Go to <https://www.luis.ai/> and click on “Sign-in or create an account” as shown below 
2. Sign-in with your Microsoft Live account if you already have one, otherwise sign-up for a Microsoft Live Account

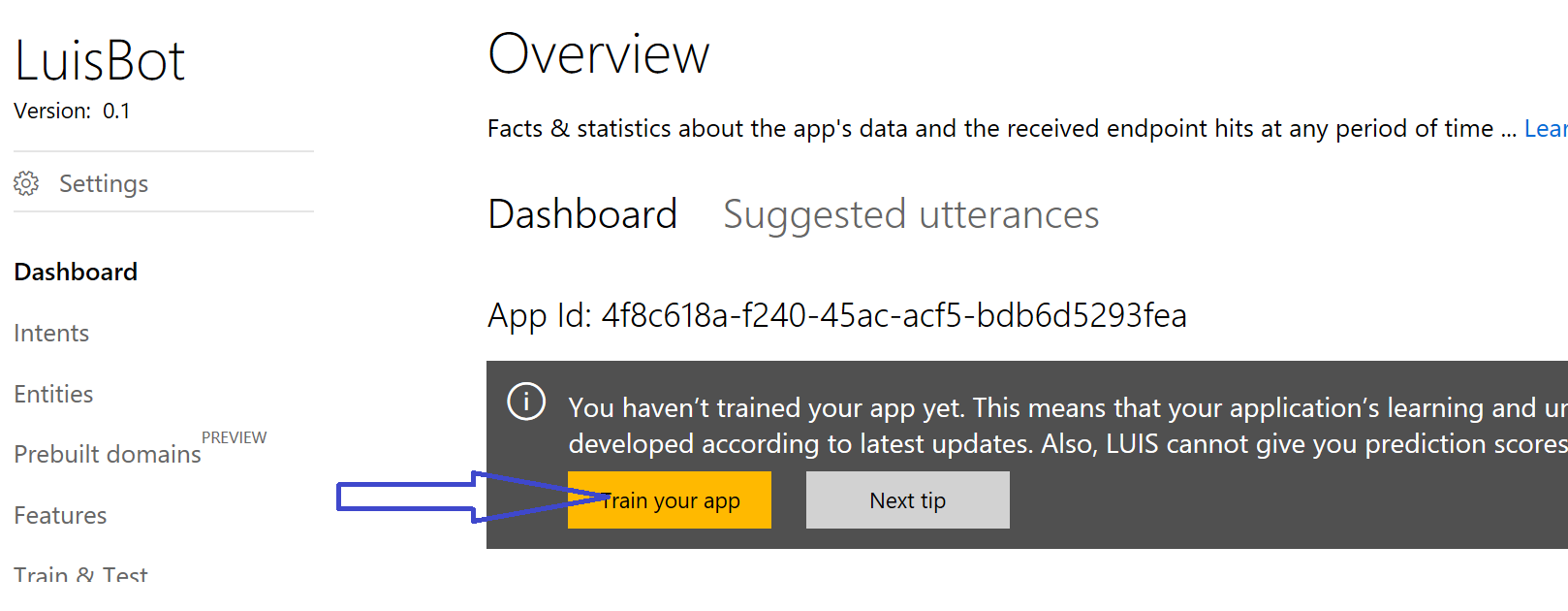


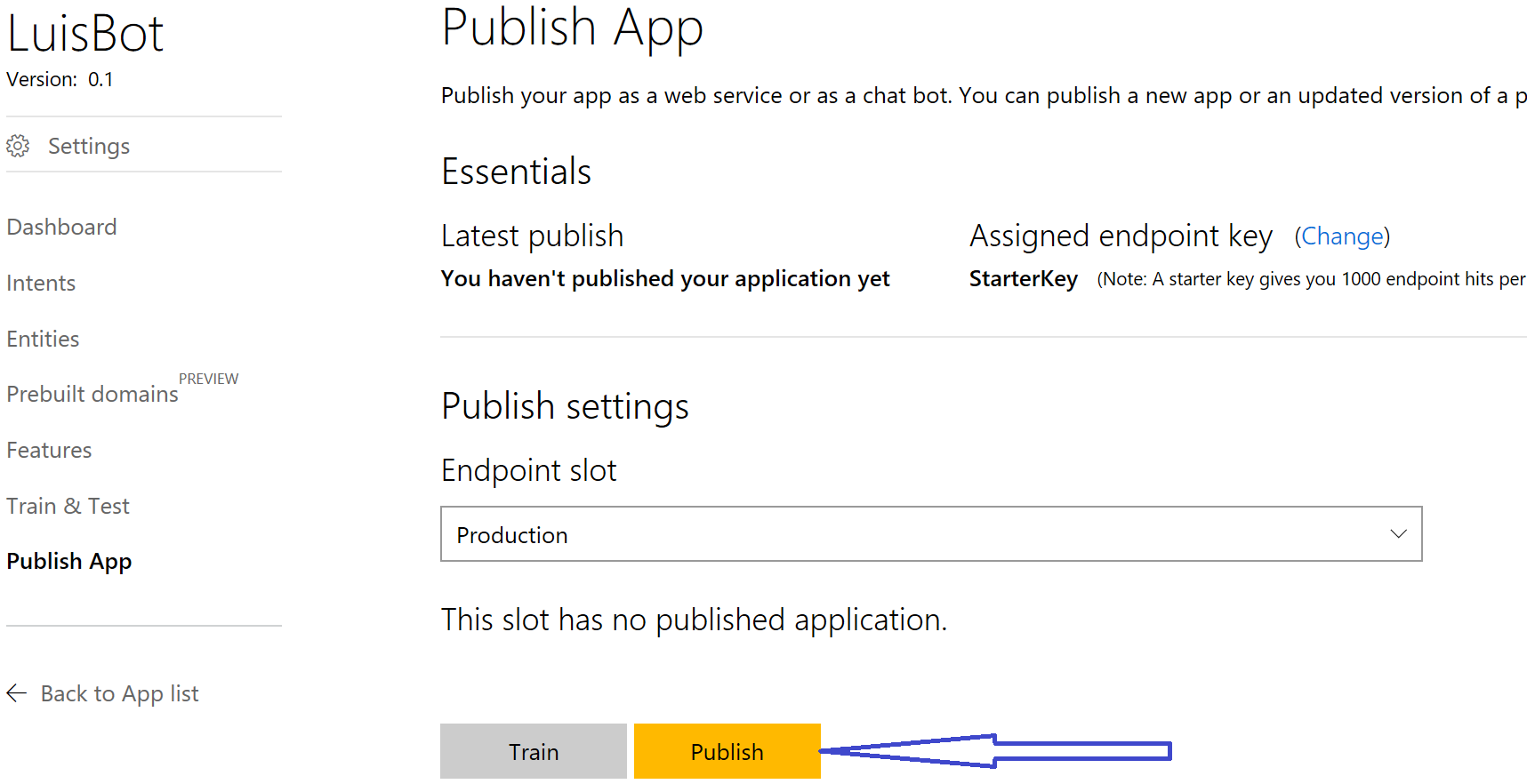
1. Once you complete the sign-up and/or sign-in process, you should be reaching the LUIS welcome page (if you are using LUIS for the first time), in the welcome page provide your Country, Company/Organization and other required fields and read and agree the terms and conditions and click “Continue”, once all done you should reach LUIS home page which will look similar to the image show below 
2. You can build your own LUIS model or import an existing model, for the purpose of demonstration we will be importing the already trained LUIS model, Click on “**Import APP**” and select “LuisBot.json” given in the Git repository



1. Select the “LuisBot.json” from your local folder (where you cloned this GIT repository) and click “**import**”

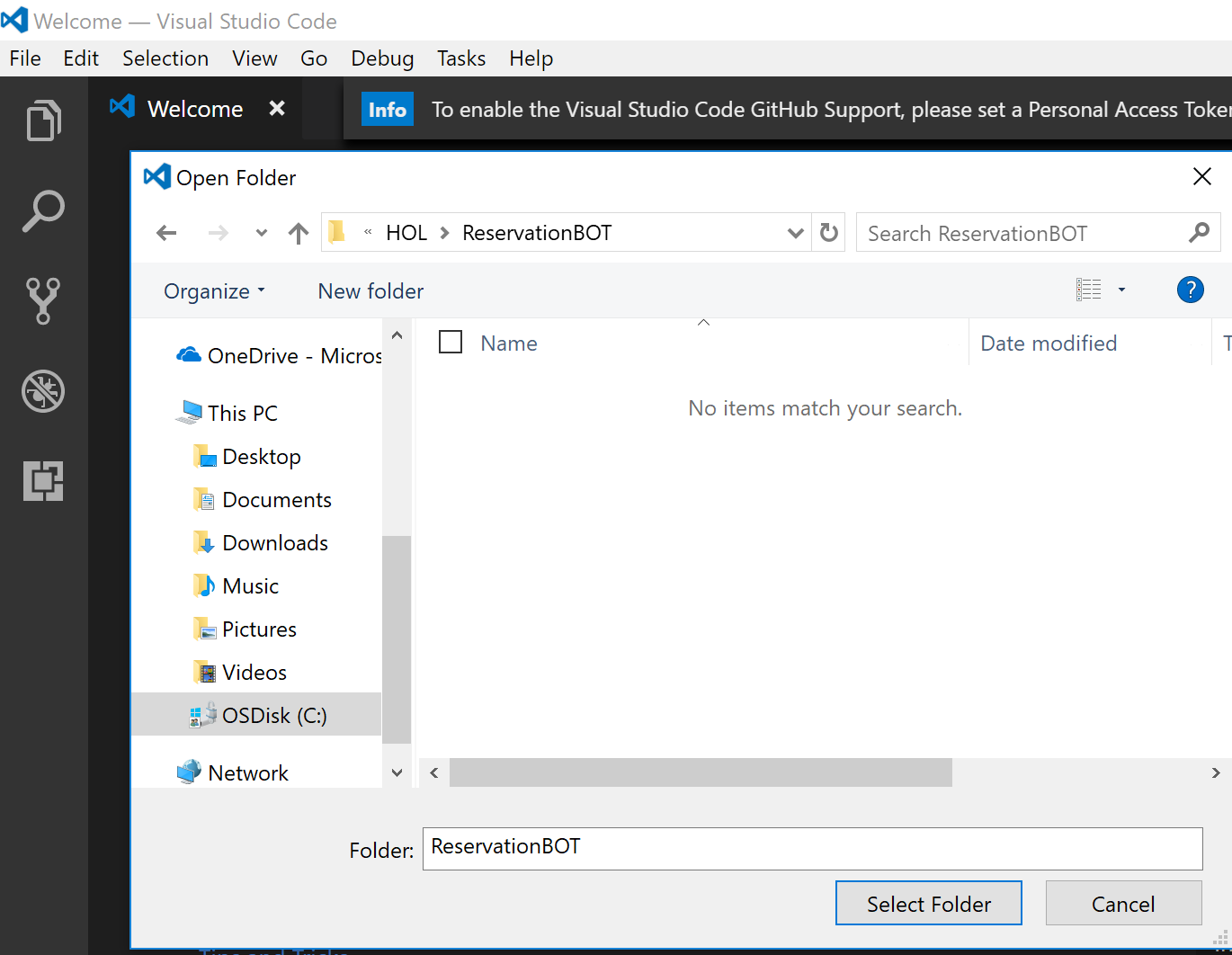
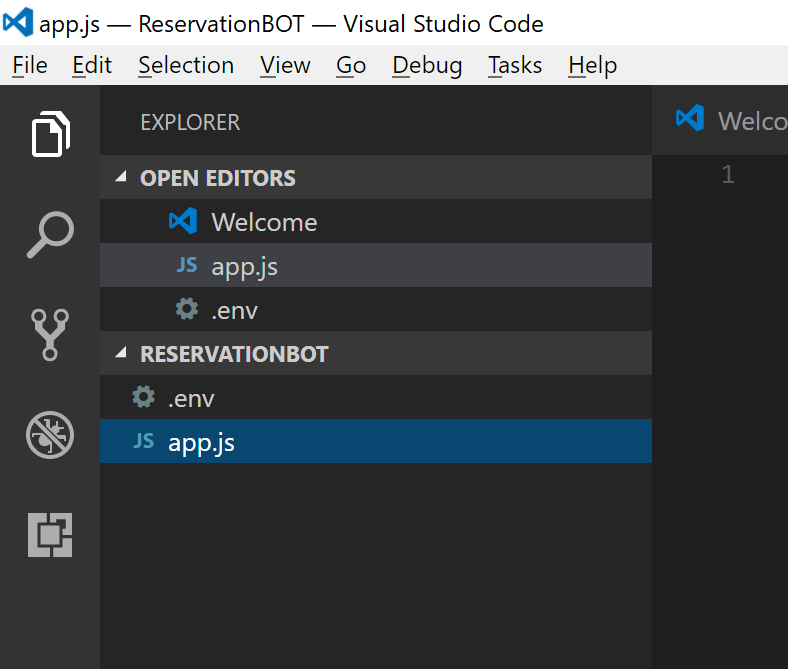


1. Go to LUIS dashboard and click on “**Train your app**” and your LUIS model will be qued for training 
2. In the LUIS application's dashboard, click the "**Publish App**" button in the left side bar, select an Endpoint Key and then click the "**Publish**" button. After a couple of moments, you will see a url that makes your models available as a web service.



1. Once the model gets published, you will get “**Endpoint URL**”, make a not of it, you will need it to use in the BOT application

**Step 2 – Building BOT using NodeJS**

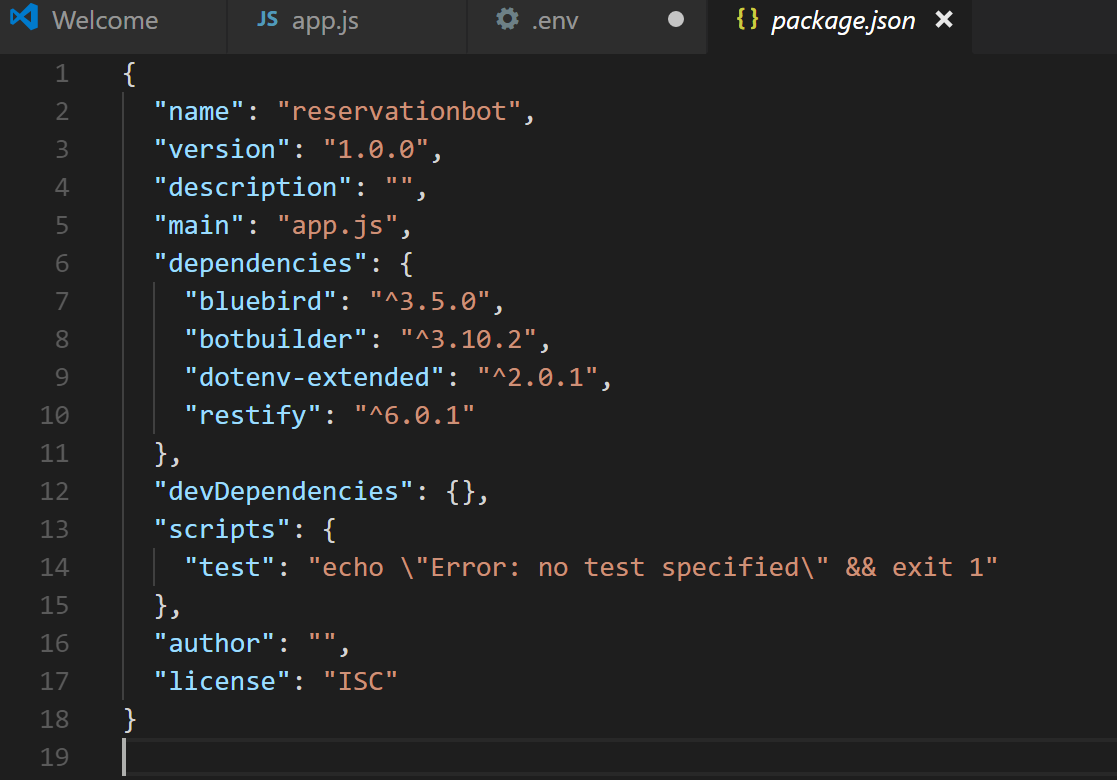
1. Create a folder in your computer where you would like to build the BOT application [**Example** : C:\HOL\ReservationBOT] 
2. Open this folder in Visual Studio Code (or any editor of your choice) – **File -> Open Folder** -> Select the folder you have created (Example C:\HOL\ReservationBOT) 
3. Create **app.js** and **.env** files in the folder as shown below 
4. Copy the content from **.env** file from the GIT repository to .env file in your project. And update the LUIS model URL you have obtained in the earlier step

# Bot Framework Variables

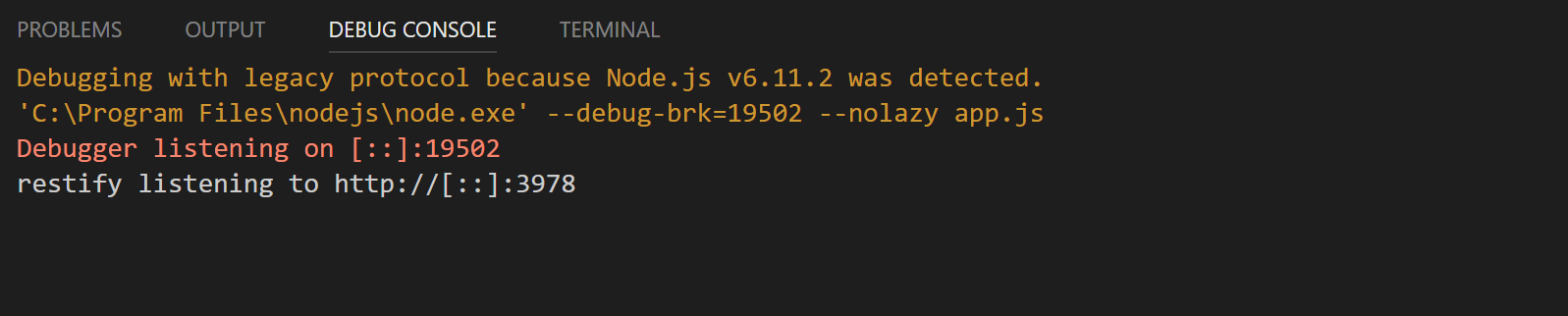
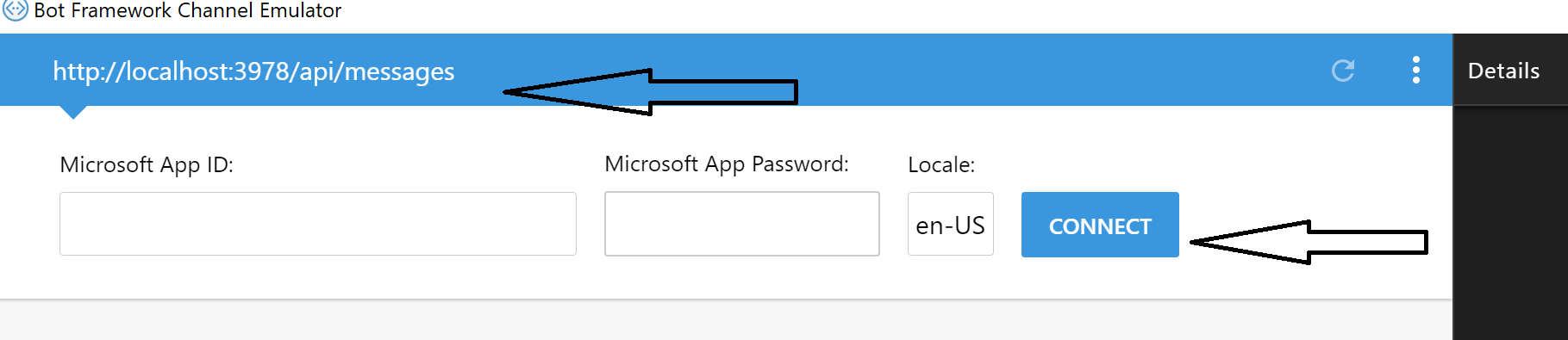
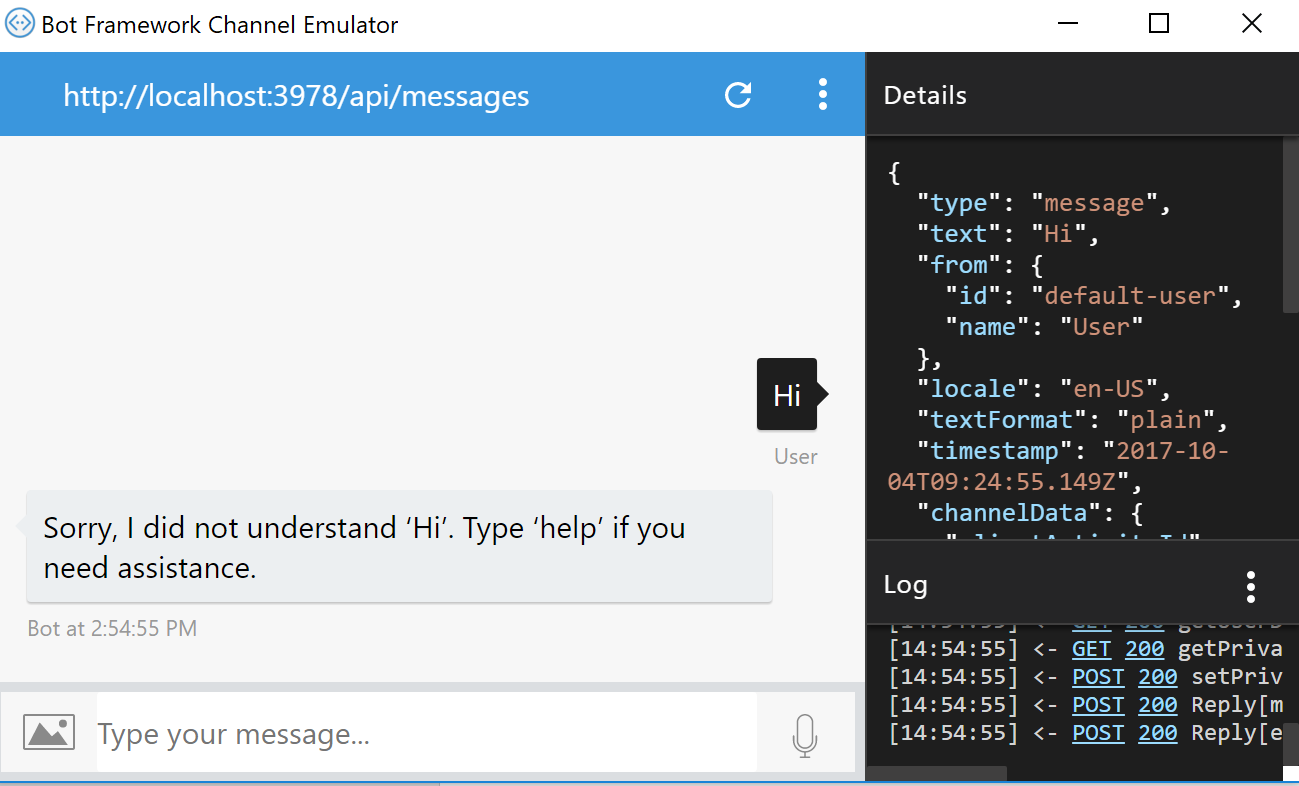
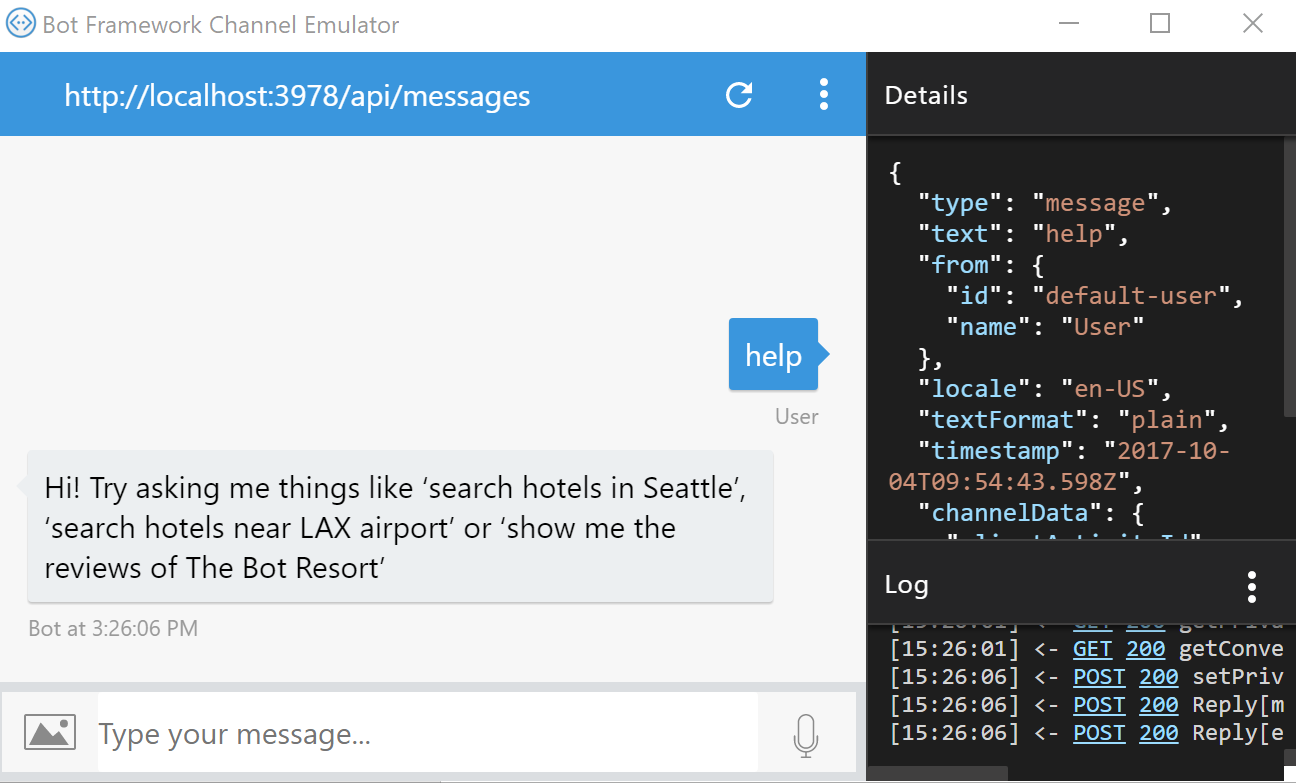
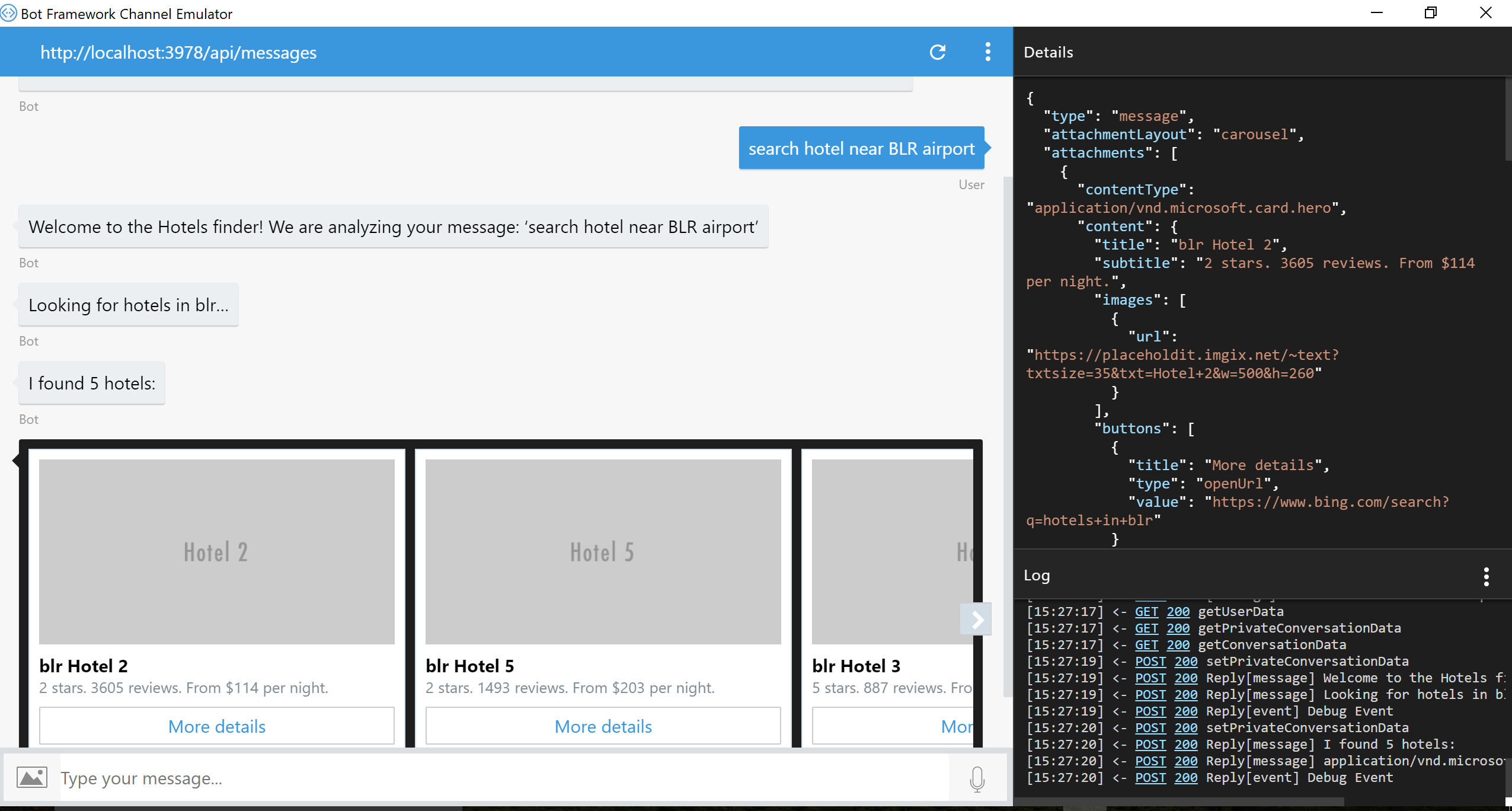
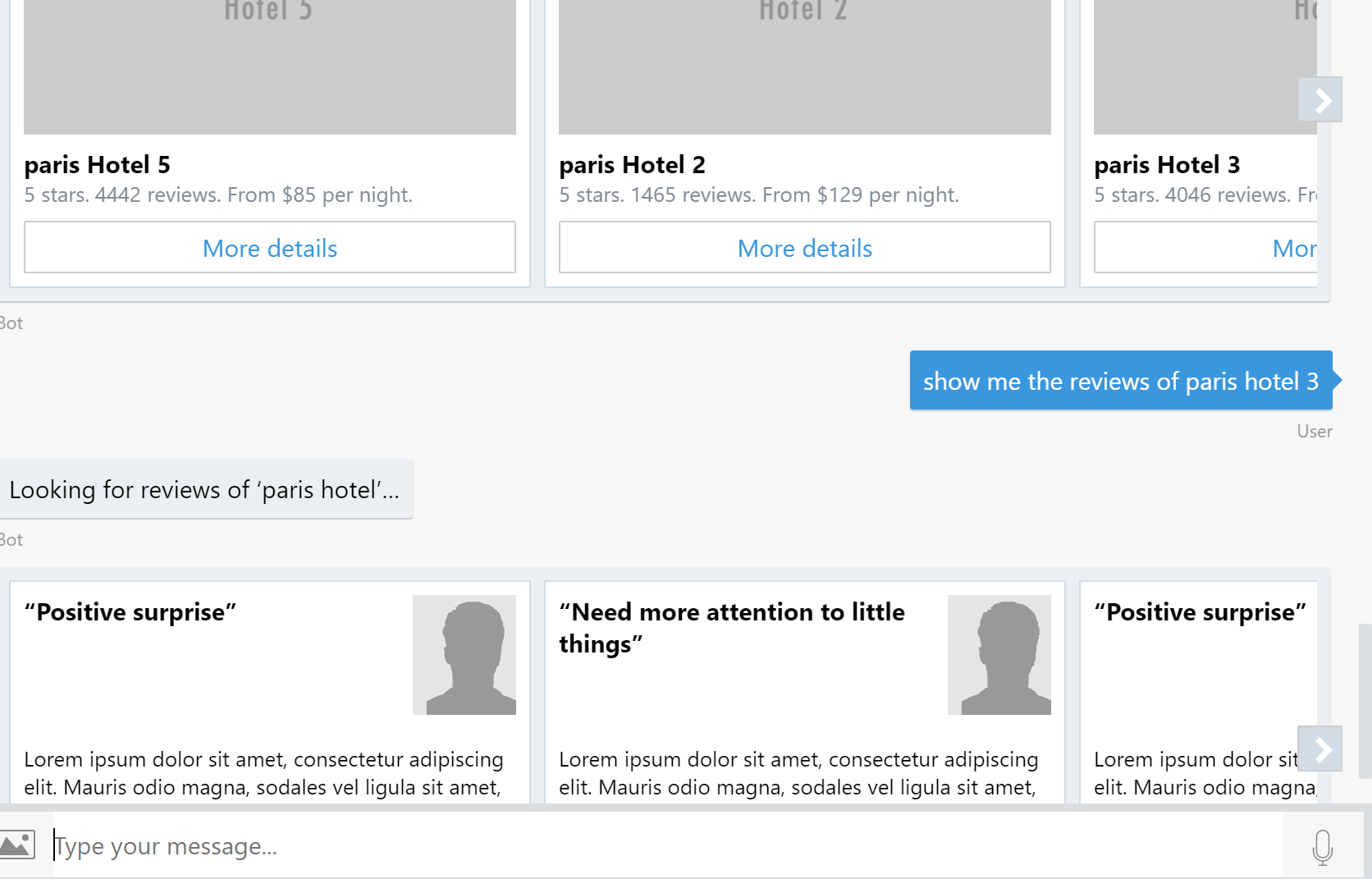
MICROSOFT\_APP\_ID=

MICROSOFT\_APP\_PASSWORD=

LUIS\_MODEL\_URL=

1. Install pre-requisite Node packages. Open windows command prompt or Node.js command prompt and navigate to the working folder
   1. ***npm init***
      1. This command will create package.json file in your project, provide a valid name (all in lower cases) when promoted and other parameters can be left empty
   2. ***npm install botbuilder --save***
   3. ***npm install dotenv-extended --save***
   4. ***npm install restify --save***
   5. ***npm install bluebird –save***
2. Once you complete all the above steps, you should have package.json file looks something similar to the one shown below 
3. Copy **Store.js and spell-service.js** files **from the GIT repository** >> <https://github.com/rawatsudhir1/OSI2017/tree/master/MBF/Code> << and paste then in your working folder
4. Open app.js file and paste the following content (You will find the file **app.js** as well in the GIT repository for your reference)

|  |  |
| --- | --- |
|  | // This loads the environment variables from the .env file |
|  | require('dotenv-extended').load(); |
|  |  |
|  | var builder = require('botbuilder'); |
|  | var restify = require('restify'); |
|  | var Store = require('./store'); |
|  | var spellService = require('./spell-service'); |
|  |  |
|  | // Setup Restify Server |
|  | var server = restify.createServer(); |
|  | server.listen(process.env.port || process.env.PORT || 3978, function () { |
|  | console.log('%s listening to %s', server.name, server.url); |
|  | }); |
|  | // Create connector and listen for messages |
|  | 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('Sorry, I did not understand \'%s\'. Type \'help\' if you need assistance.', session.message.text); |
|  | }); |

1. The above line of Node code will create a BOT application and host it in your local host port number 3978, When you run the application at this stage, you should see that restify listening to port 3978 
2. Now launch “botframework-emulator” desktop application and feed-in <http://localhost:3978/api/messages>and click on “**CONNECT**” 
3. At this stage, when you type something in the bot, it should give you response ***“Sorry, I did not understand ‘Hi’. Type ‘help’ if you need assistance.”*** We have not implemented the conversation dialog yet, we will be building the same in the subsequent steps. 
4. Now **delete** all the content from app.js, and **copy** all the content from app.js from GIT repository and **paste** it in your working folder in app.js Alternatively, you can just **copy** the app.js file from GIT repository (from the cloned folder) and **replace** app.js file you have in your working folder
5. Open the .env file and update value for configuration element **LUIS\_MODEL\_URL =**
6. Now run the application once again (F5 in Visual Studio Code), and re-connect to BOT from the BOT Emulator for the URL <http://localhost:3978/api/messages>
7. Type “Help” in the BOT and hit enter, you should get response as “***Hi! Try asking me things like ‘search hotels in Seattle’, ‘search hotels near LAX airport’ or ‘show me the reviews of The Bot Resort’*** as shown below 
8. Now try posting question “***search hotel near BLR airport***” and you should get the following response 
9. Try posting questions such as
   1. ***Hotels in Seattle***
   2. ***Best hotels in New York***
   3. ***Look for hotels in Paris***
10. Once you get the result for hotels (for example – results for “Look for hotels in Paris”), you can also ask reviews of hotel – such as “**Show me the reviews of Paris Hotel 5**” 

**Conclusion**

We have completed developing “Hotel Search” BOT using Azure BOT development framework and testing locally using Azure BOT Emulator. This complete the Hands on Lab, you can further learn about publishing BOT to Azure and configuring BOT to be used with various channel in the [Microsoft article](https://docs.microsoft.com/en-us/bot-framework/deploy-bot-overview)