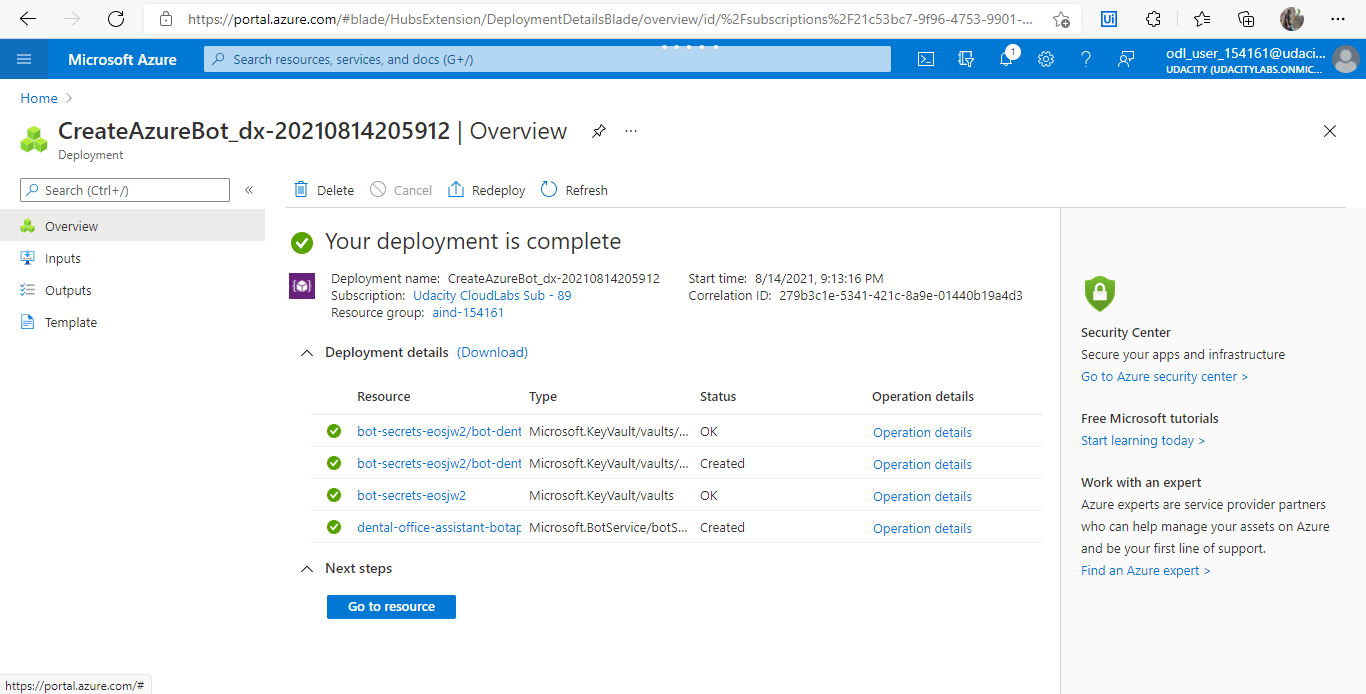
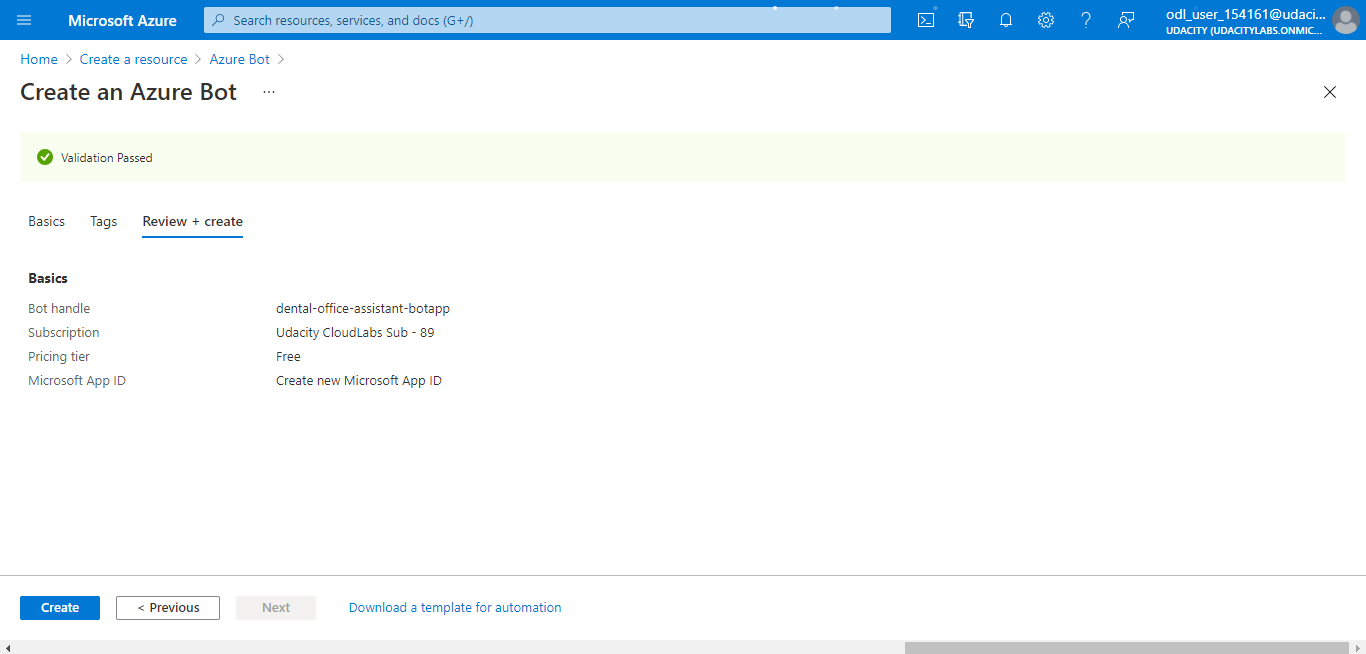
**STEP 1, Create a bot resource**

* Create a new resource, in Microsoft Azure Portal. From the first figure go to input , there you will find Microsoftapp password



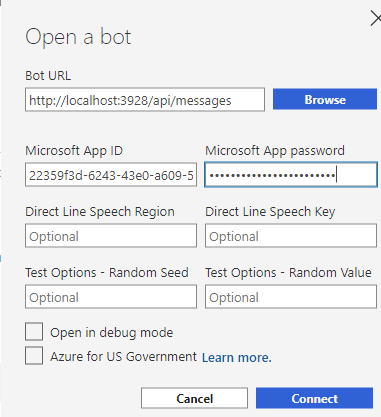
* Then create the resource, and get the MicrosoftAppId from the configuration item from the second screenshot below.



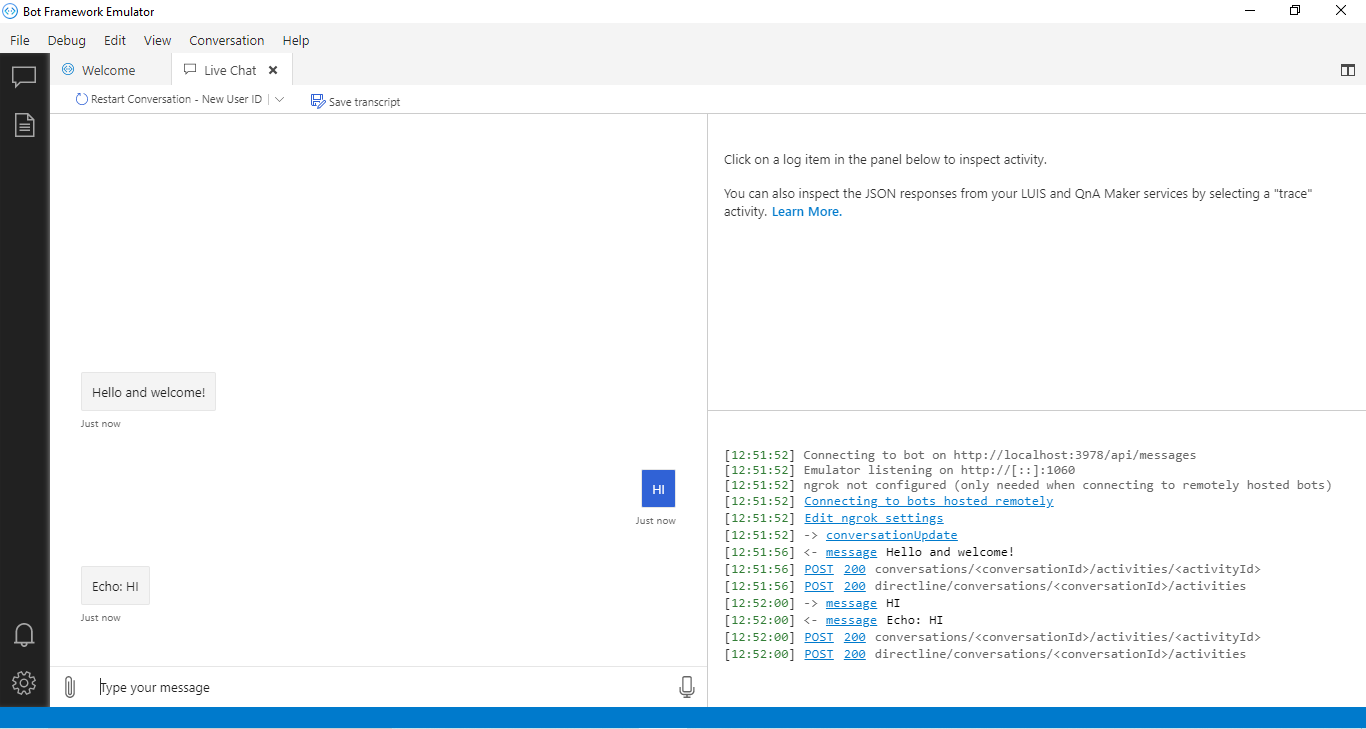
* Create or open the .env file in Visual studio code and enter the appID and app app password
* Make sure you have index.js file.
* Next step is to install all the dependency from pakage.json file.
* First run **npm install** in terminal. It will install all dependencies required.
* Then run **npm start** in terminal. It will start the bot, and it will start listening to a port.



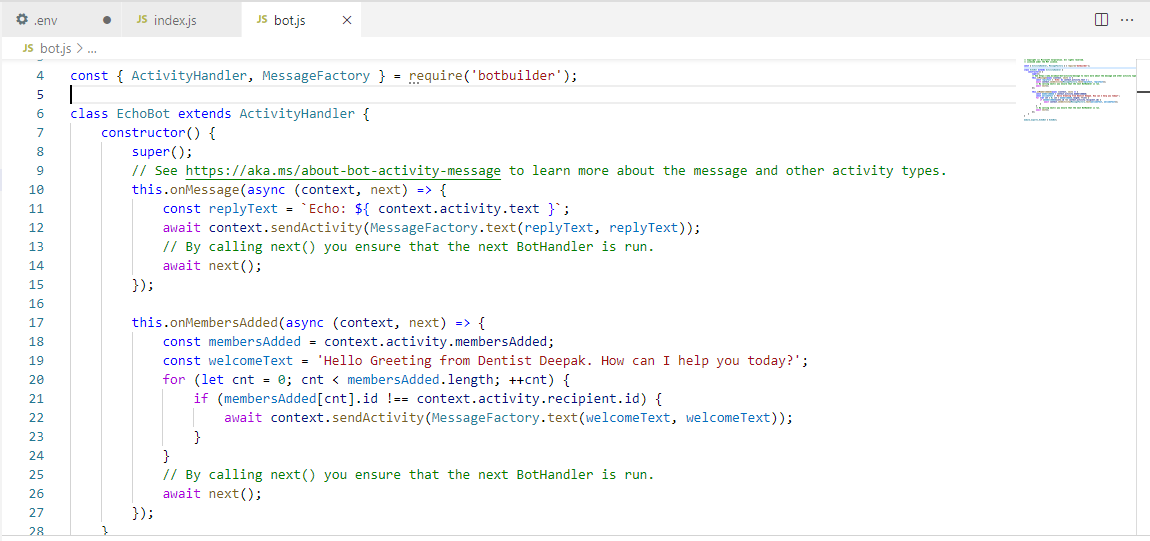
* Open the Bot Framework Emulator Application, enter the localhost address, appId and app password
* Make sure you enter correct, port number, In below screenshot correct addess is not been entered.
* Click on connect button, it should give 200 message if configured correctly.



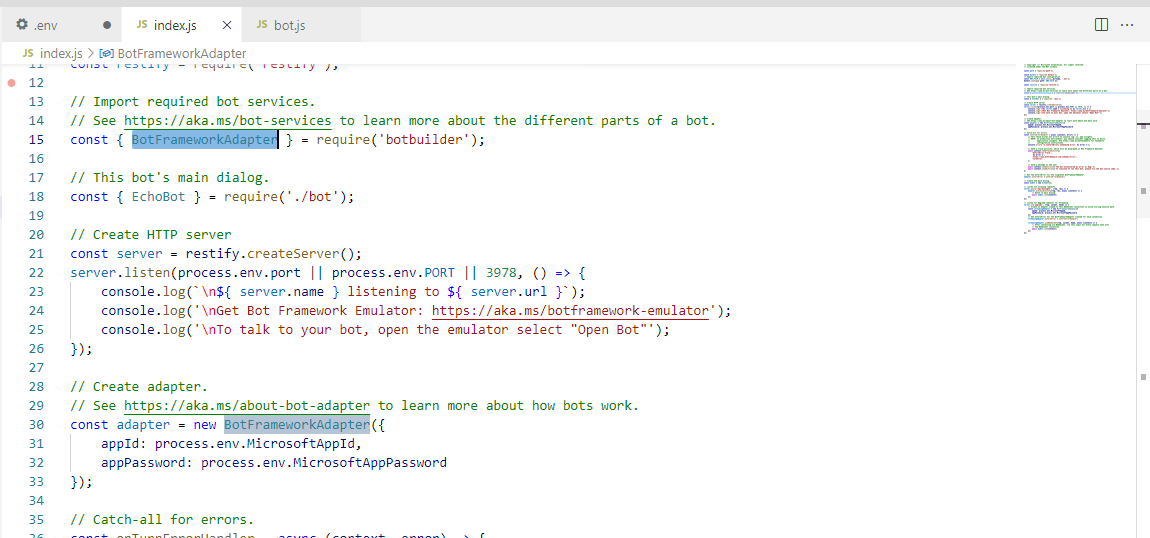
* At this stage bot will revert back the message entered.



* The bot.js file will extend from ActivityHandler class will handles the messages in our bot.
* The onMemberAdded function will show the new message once a user is added.
* onMessage function will handle the incoming and outgoing text.
* Context object will contain the function to interact with the bot.



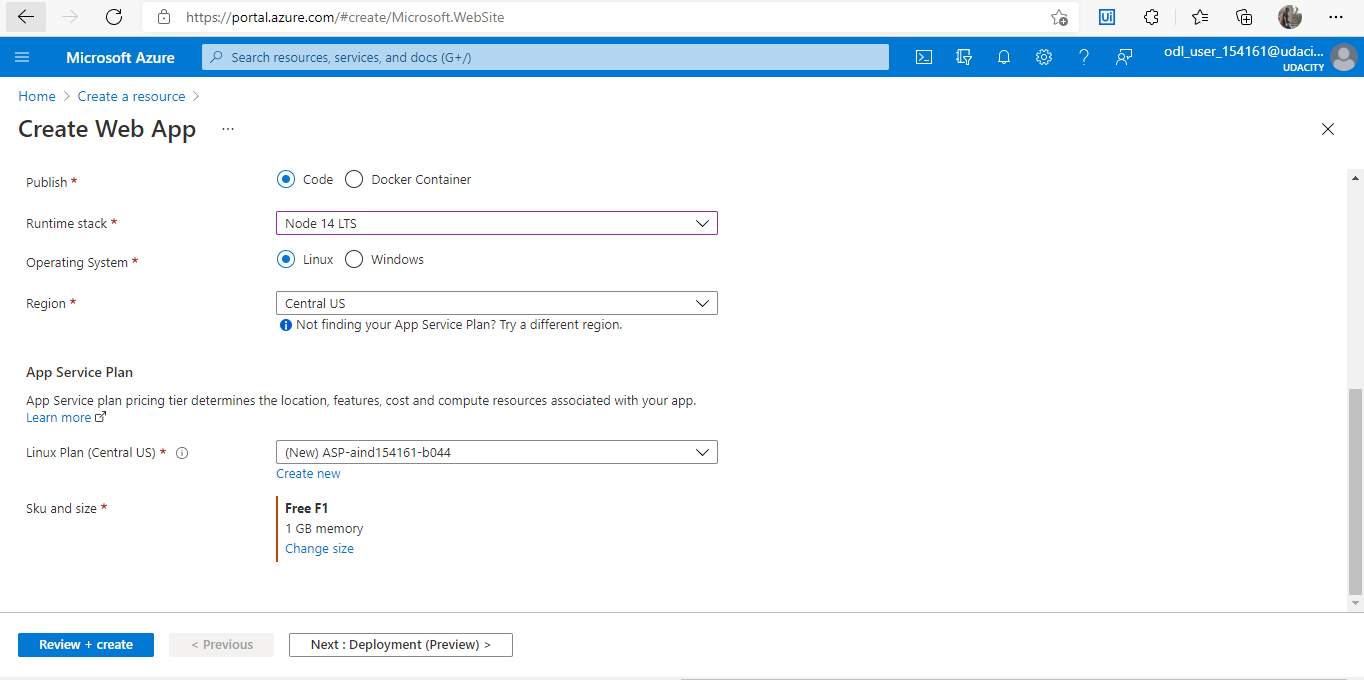
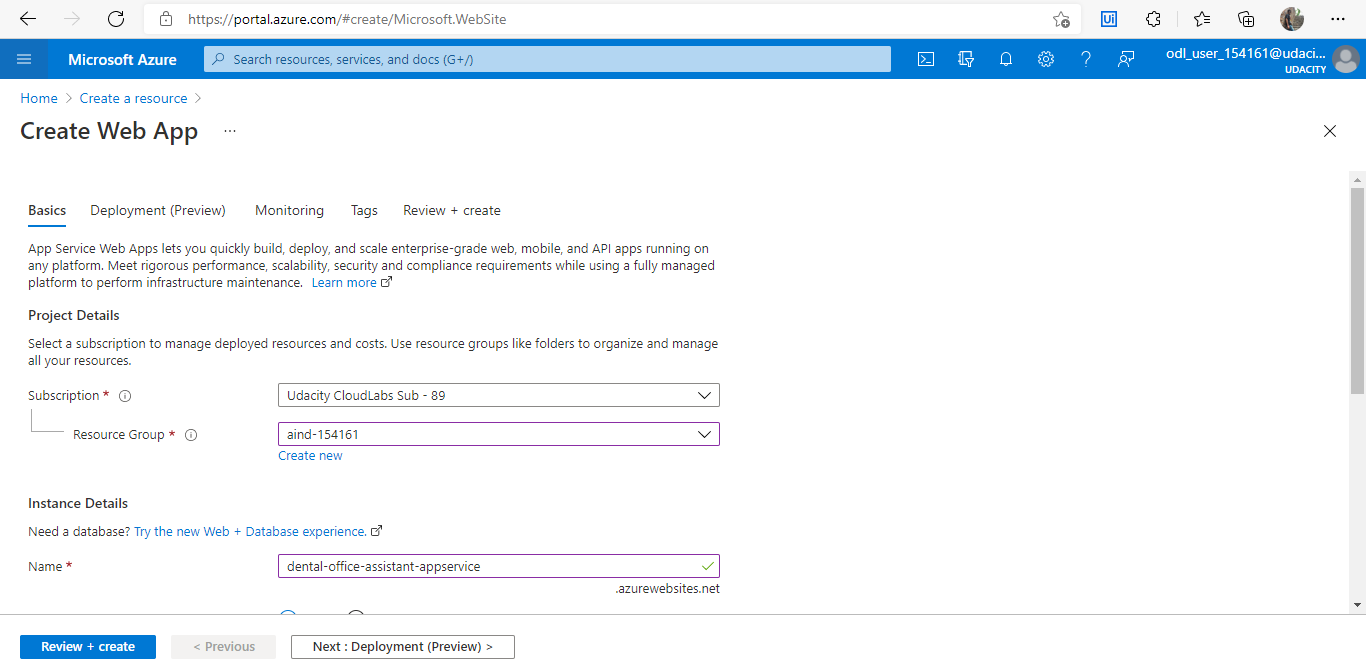
* The idex.js will contain the configurations used in our application. For example, it created and registered the BotFrameWorkAdapter from the Microsoft bot builder SDK, linked with the appId and Password variables created in .env file.
* Created Echobot from bot.js file.
* The restify server is created which will listen to 3978 port



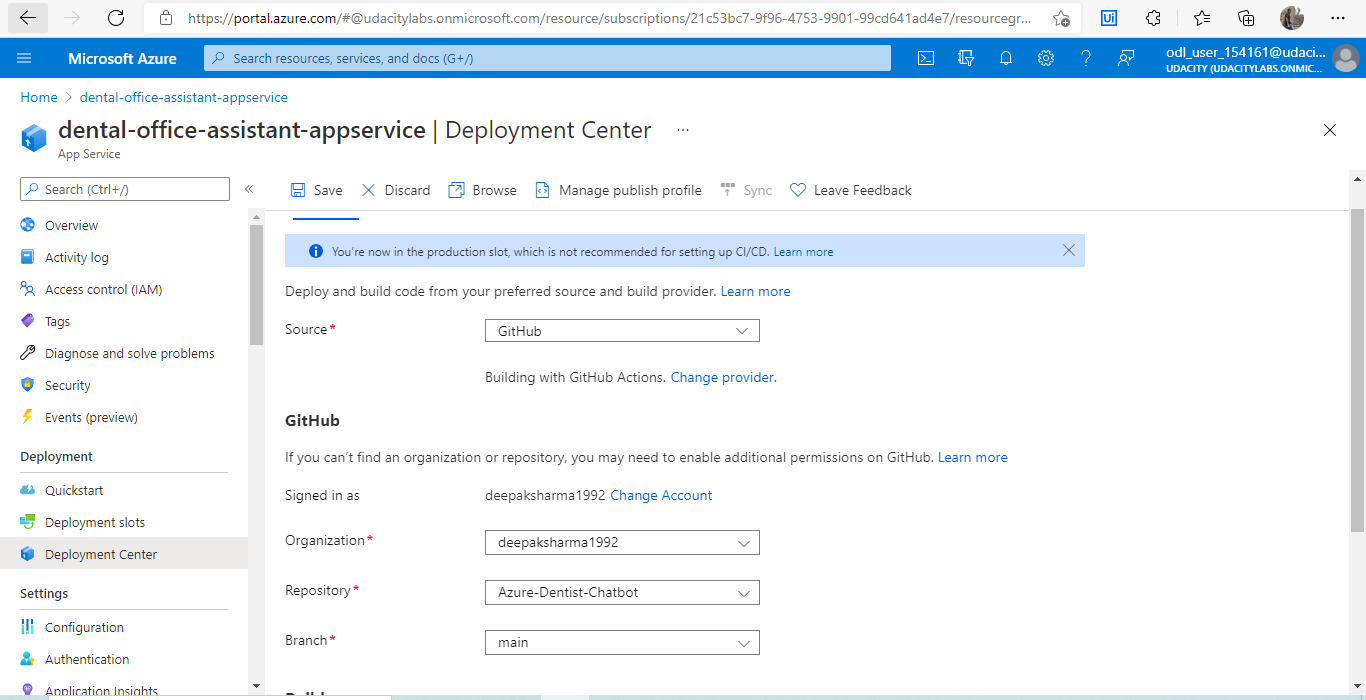
**STEP 2 APP SERVICES**

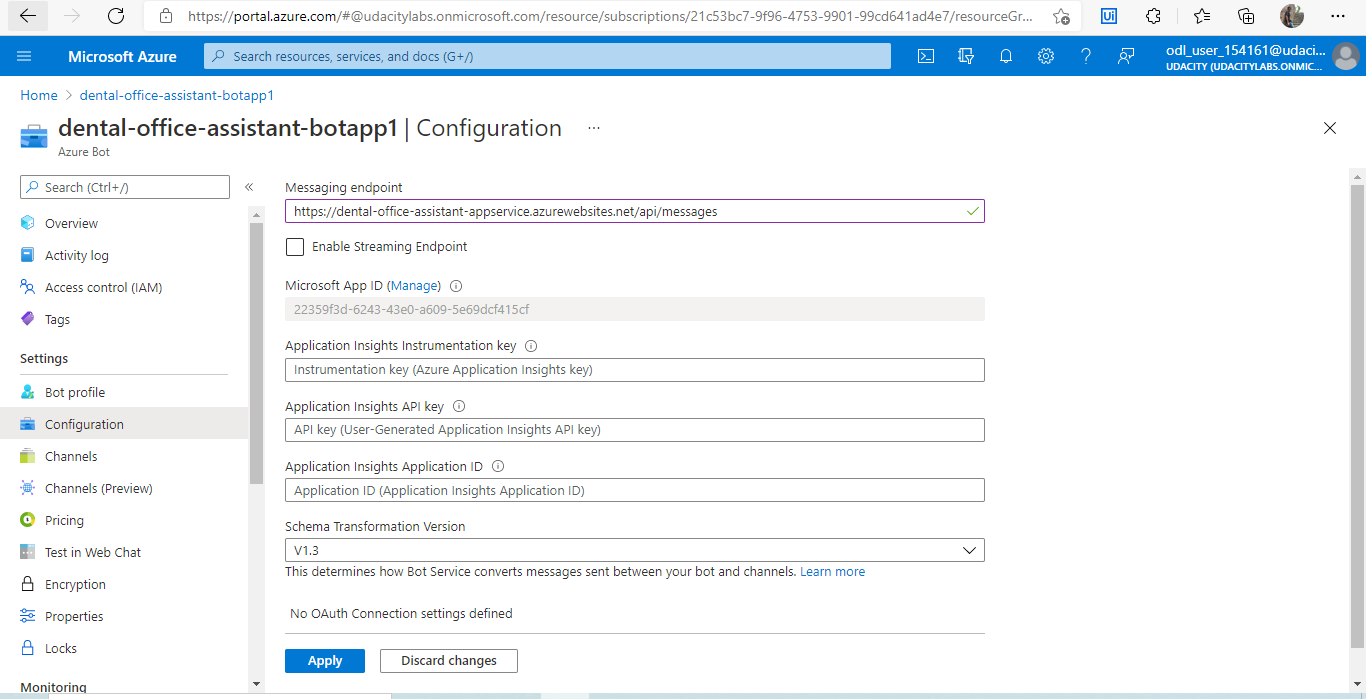
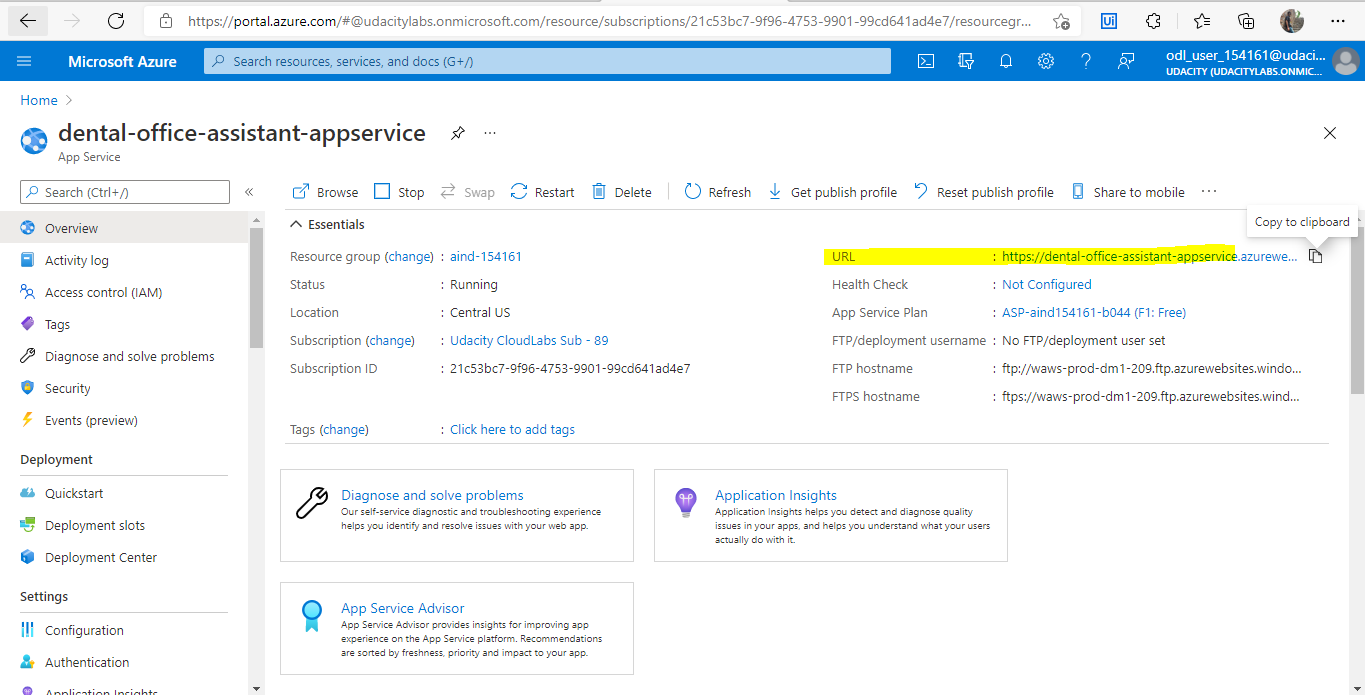
**Main purpose is to link the new created web app service with Bot Service created in step 1**

* Create new app resource of web service and configure like below



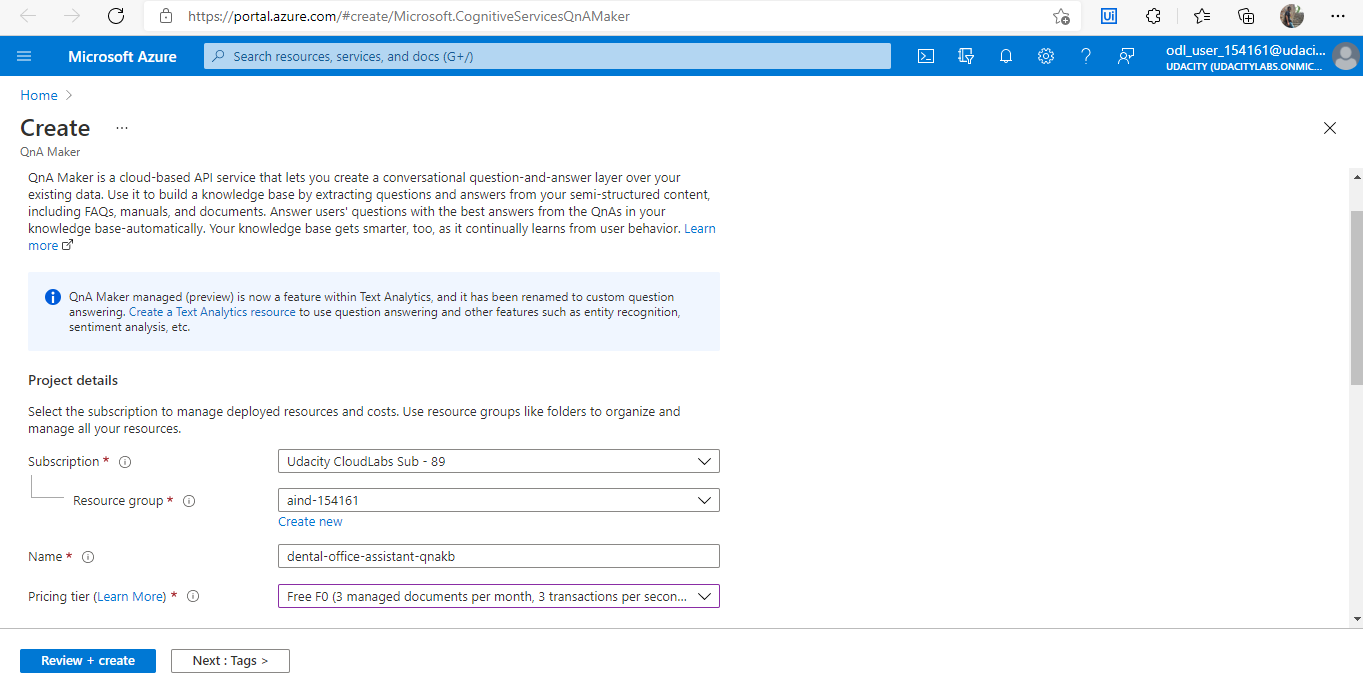
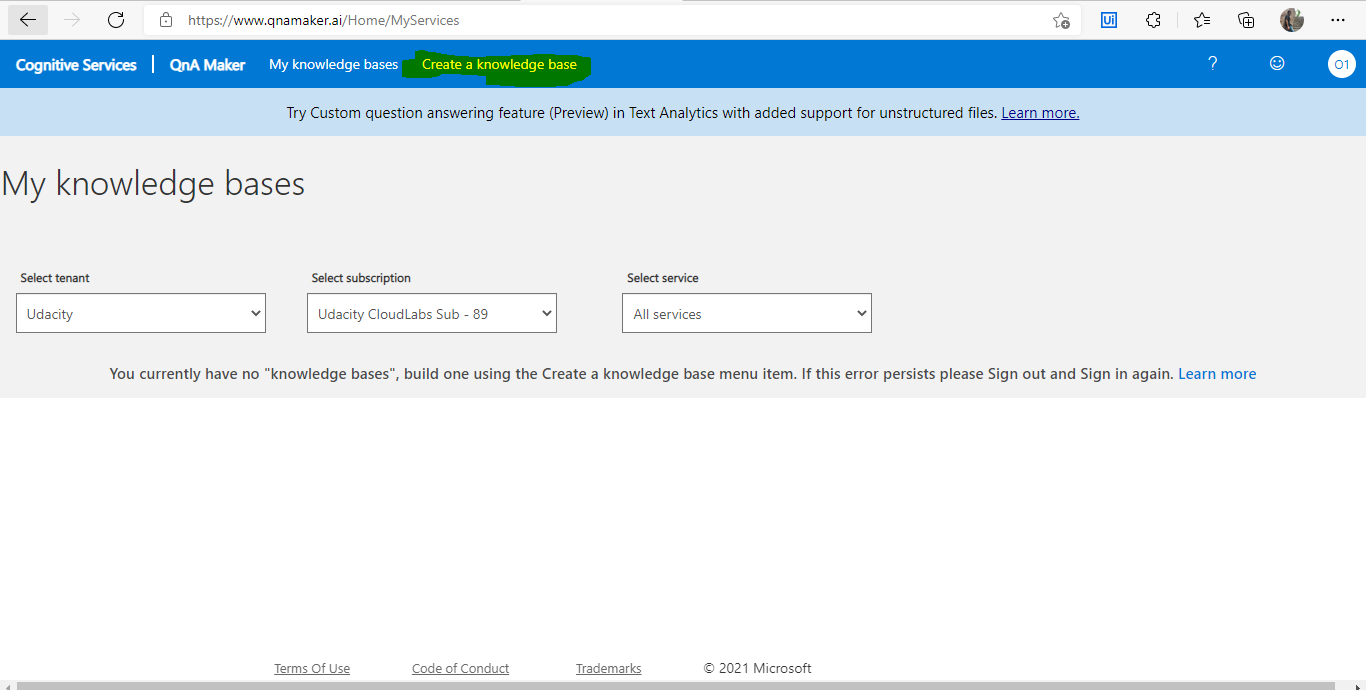
* Link your app with your github account for continuous deployment.



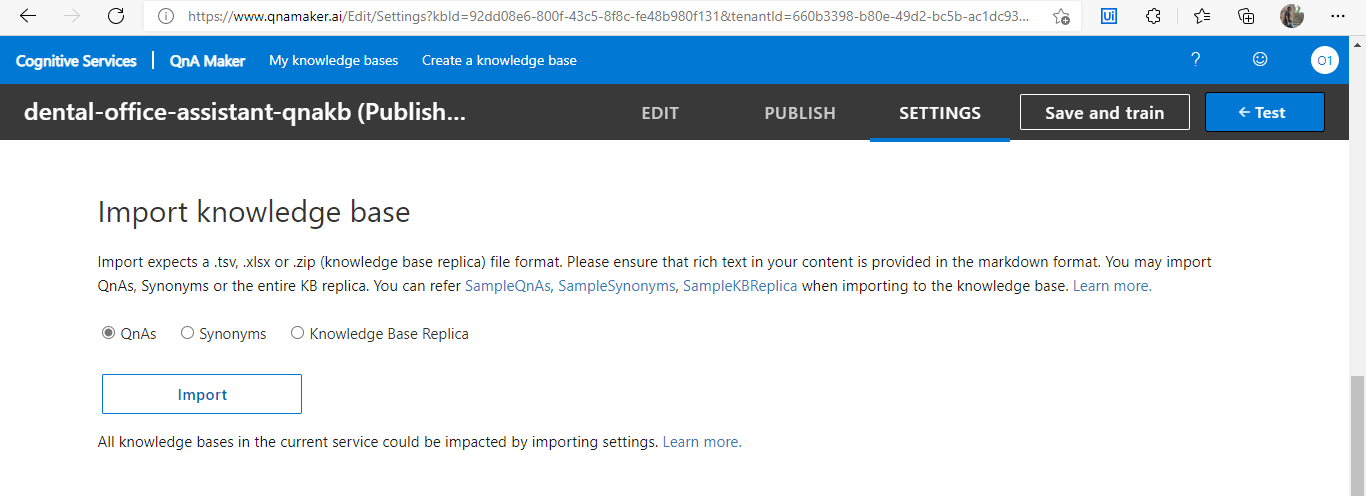
* Copy the url created and link it with the Bot resource created as stated below. Add api/messages at the end of the URL.

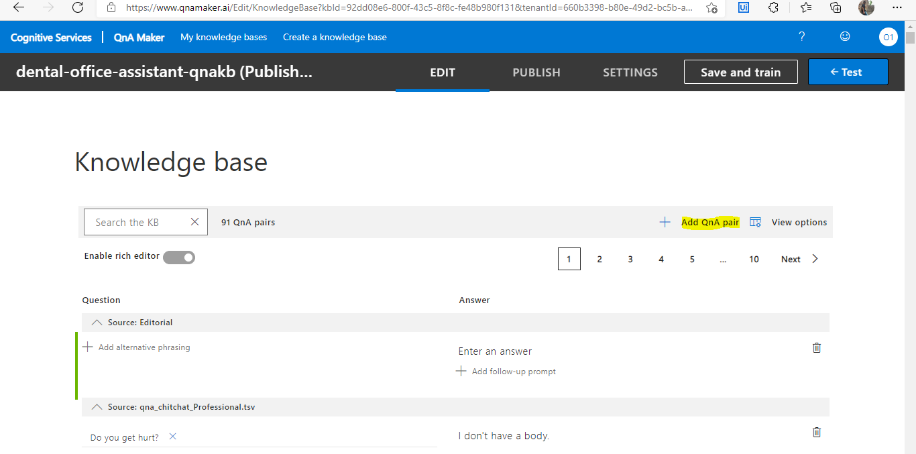
**Step 3 QNA Service Integration**

* Go to <https://www.qnamaker.ai/>
* Click on Create a Knowledge base
* Create a QNA service in azure portal
* Create the resource

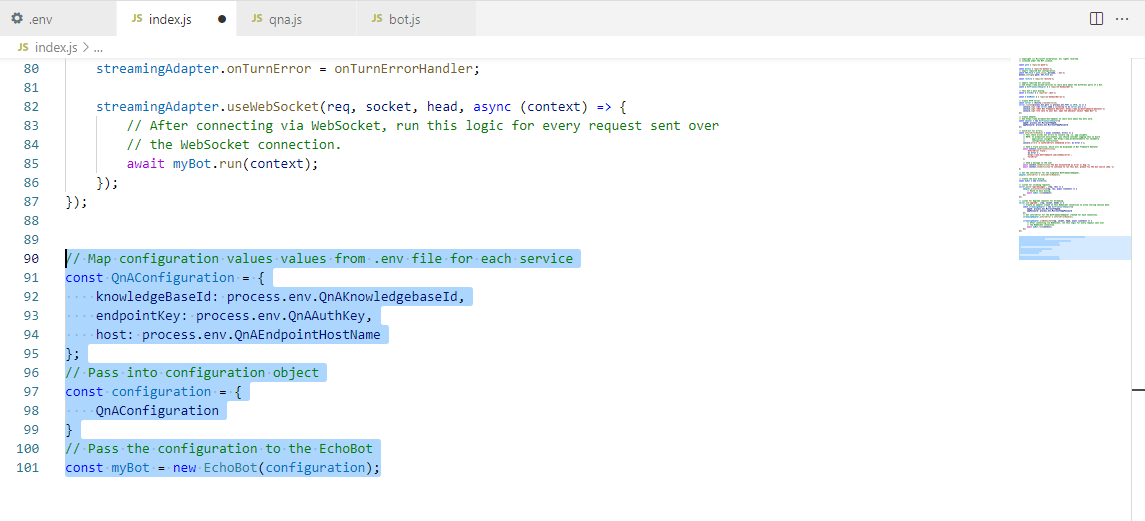
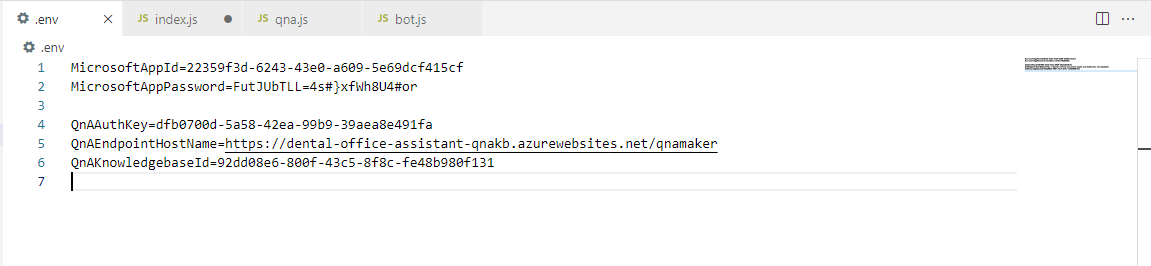


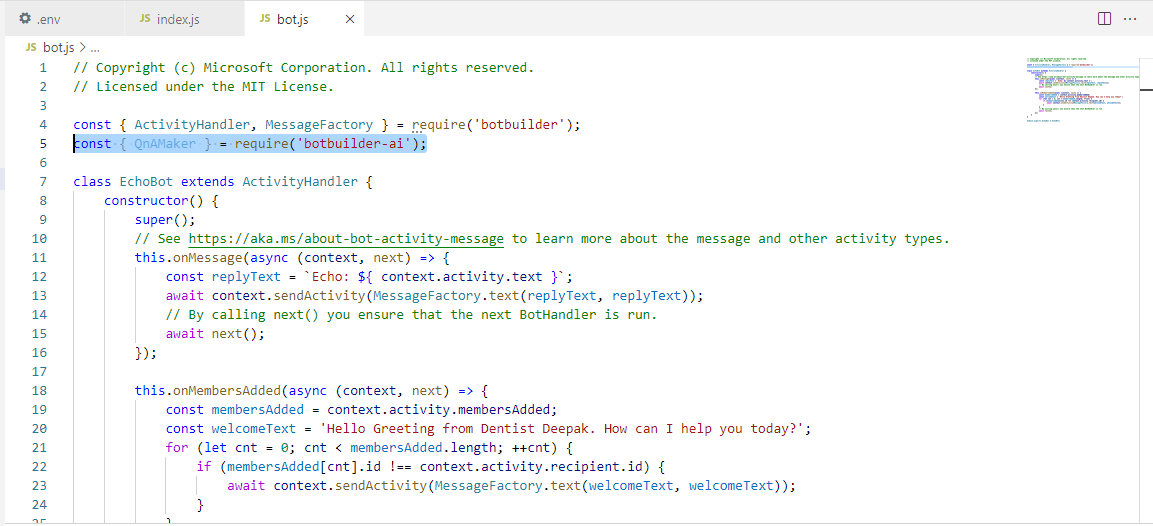
* Create your QNA questions OR,
* Import QNA tsv file.

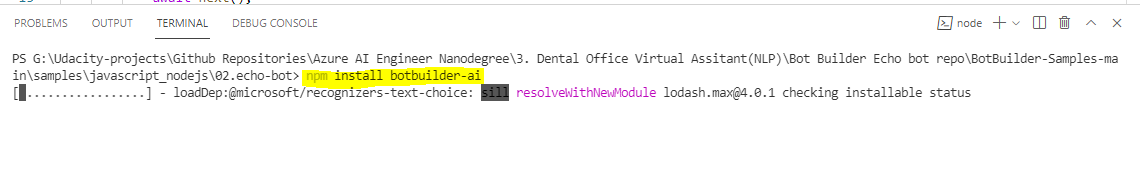




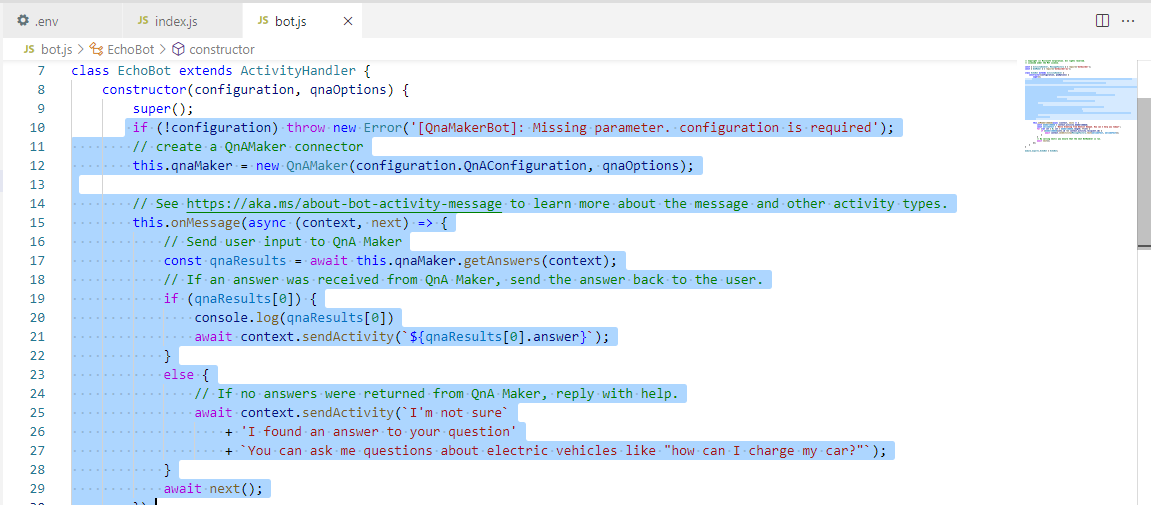
* First change the .env file by adding the configuration details.
* Then add the configuration adapter for QNA service to index.js file.
* Then pass this configuration to the EchoBot variable.
* Add the QNAMaker const to bot.js file.
* To add botbuilder-ai sdk, **perform** npm install botbuilder-ai using the terminal





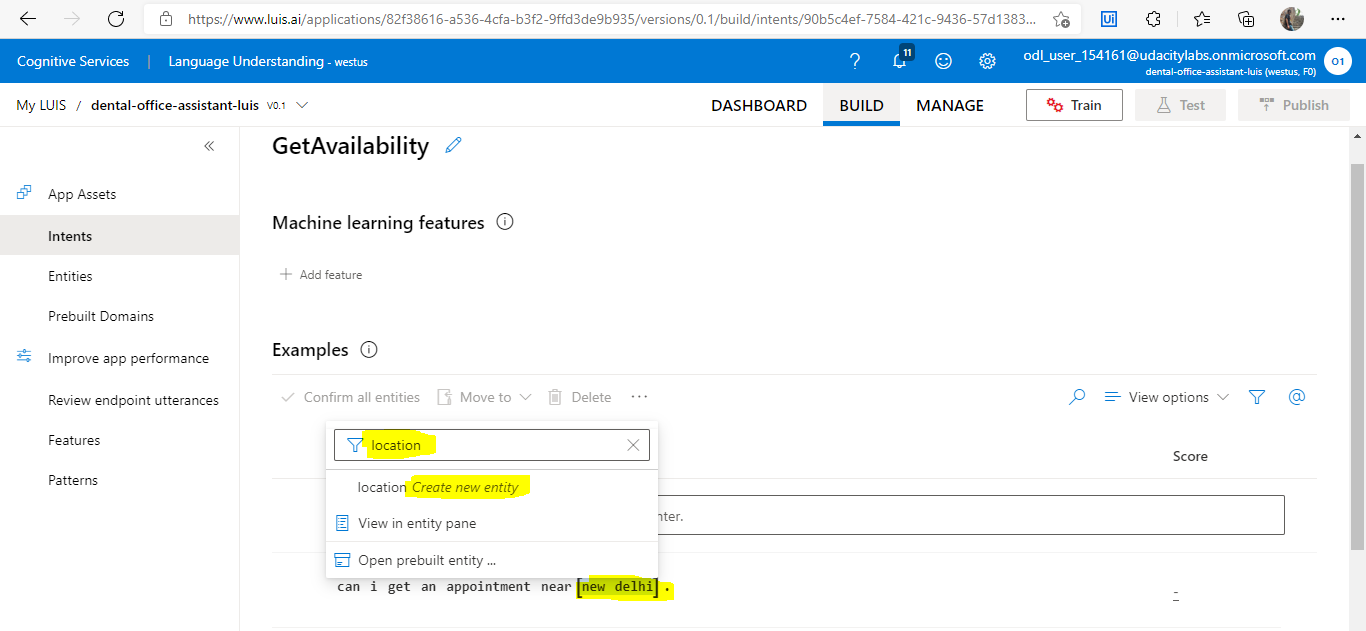
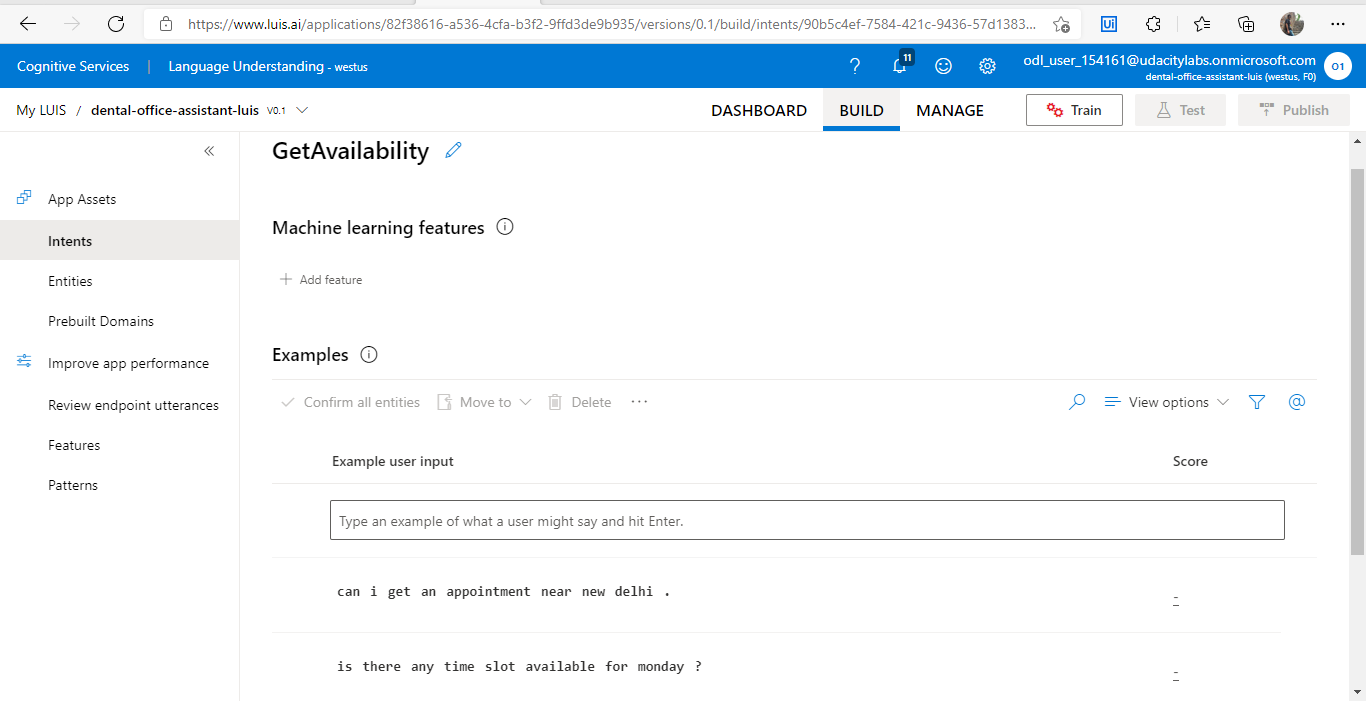


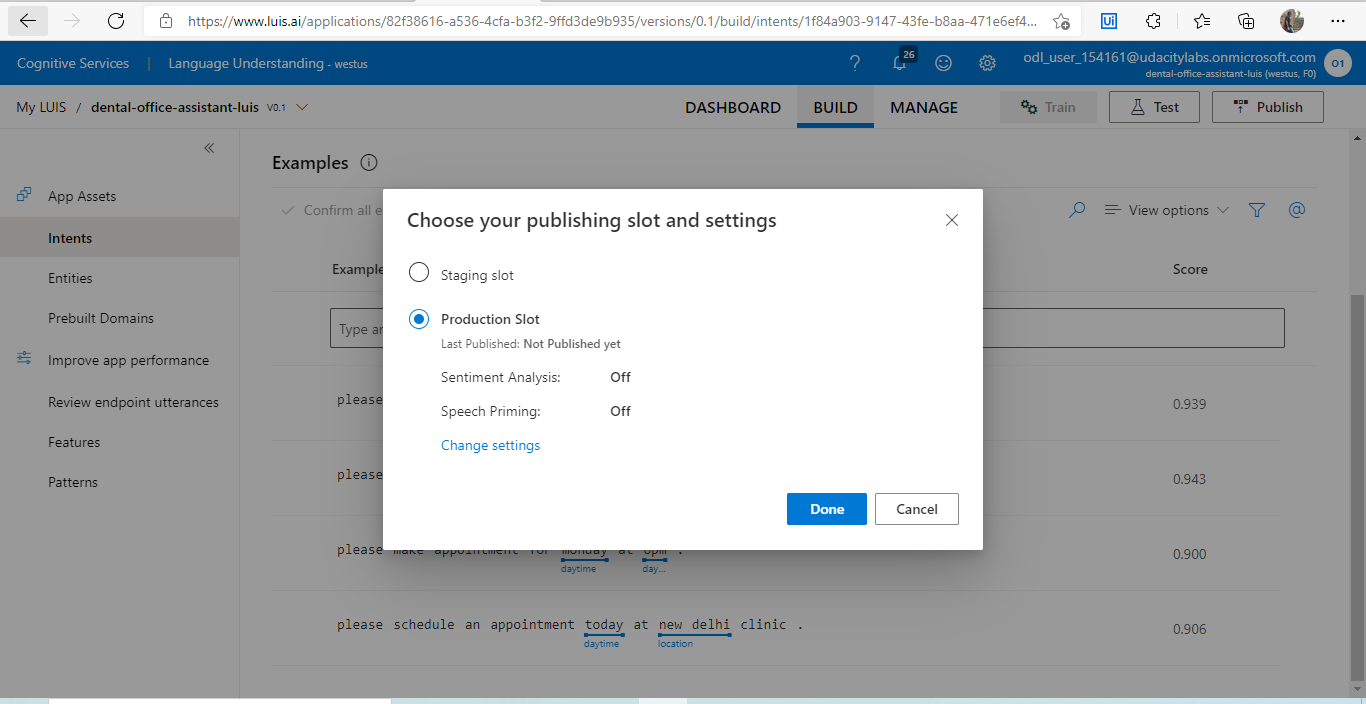
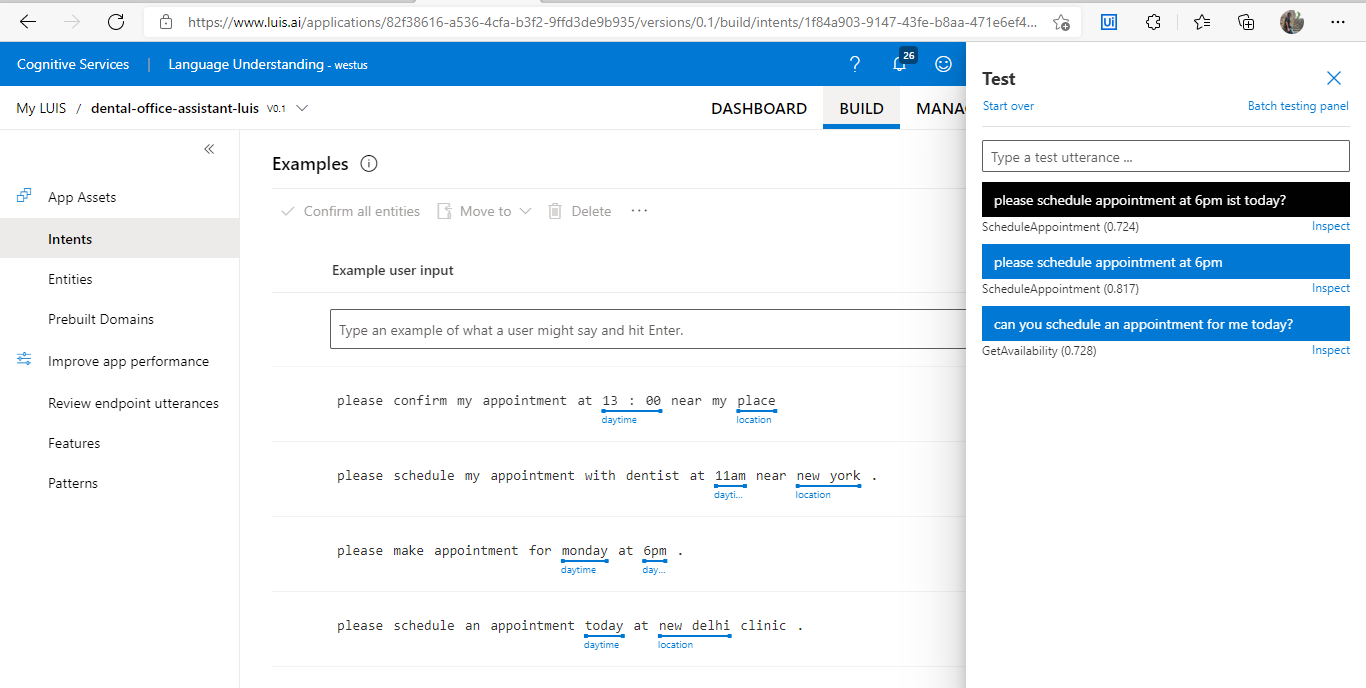
* Create QNA maker object in bot.js file.
* Modfy onMessage() function to receive answers and send them to server.

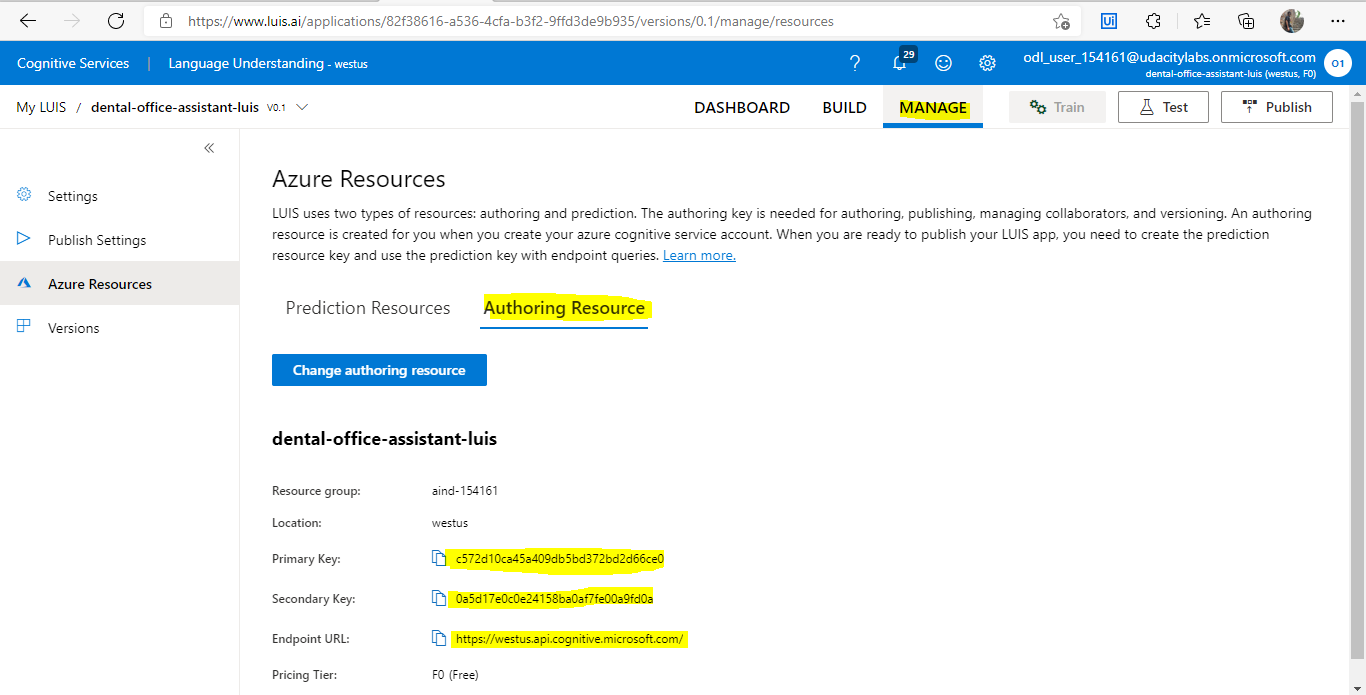


**STEP 4 LUIS Integration**

* Go to [LUIS (Language Understanding) - Cognitive Services - Microsoft](https://www.luis.ai/)
* Create the LUIS resource in above link.
* Create Intent by naming and click on create new (e.g GetAvailability)
* Then create Utterances in the Example menu.
* Then create Entities from these utterances.
* Finally publish in **production slot**.
* Click on authoring resource from manage tab to get the info for luis app.







* The above information will be place in env variable of our app.