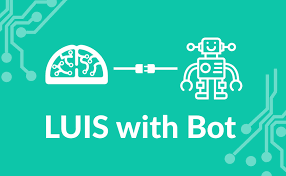
|  |
| --- |
| Introduction to Microsoft LUIS Chatbot Framework |
|  |
| April 2020  Authored by: Siddhartha Ranjan Bhattacharya |



Contents

[Setting the Context 3](#_Toc37936941)

[What is a Chatbot? 4](#_Toc37936942)

[Intro to Microsoft LUIS 5](#_Toc37936943)

[Building the BOT 5](#_Toc37936944)

[Problem Statement 5](#_Toc37936945)

[Architecture 5](#_Toc37936946)

[Training LUIS 7](#_Toc37936947)

[Subscribing with openweathermap 10](#_Toc37936948)

[Building the Flask App 10](#_Toc37936949)

[Sequence Diagram 11](#_Toc37936950)

[Config File Settings 12](#_Toc37936951)

[Testing the Flask app 12](#_Toc37936952)

[Deployment in Microsoft Azure 13](#_Toc37936953)

[Setting up Azure 13](#_Toc37936954)

[Setting up GIT 16](#_Toc37936955)

[BOT Channel Registration 17](#_Toc37936956)

[Third Party Integration 18](#_Toc37936957)

[Wrapping it up 18](#_Toc37936958)

[Appendix 19](#_Toc37936959)

# Setting the Context

*This document provides an introduction of the Microsoft LUIS chatbot framework. The intent is to cover the basics so that anyone can start building his/her own chatbot within a day. We have taken a simple use case and utilized LUIS as well as Python Flask framework to solve the same. Lastly, we have deployed the application in Azure. The steps are covered to the best possible manner. Have tried to kept it less technical and more functional in nature for easy understanding. 😊.*

# What is a Chatbot?

When I searched the internet, figured out almost thousand of definitions. So, thought of simplifying the same in my write-up. As the name suggests, there are 2 parts.

* Chat – which means capable of doing a conversation
* Bot – which means it is non-human being (often times a software application)

To put in context, a chatbot is something which will understand human language and reply back accordingly and carry on with the conversation till you are fulfilled. With the rapid advancement of AI & Machine Learning, conversational AI has improved a lot. Chatbot is nothing but conversational AI in action. While traditional chatbots used to be rule based, which is very cumbersome to be maintained, majority of the current chatbot frameworks are AI based.

*Caution- Please don’t expect that chatbots are emotional beings. These are applications which are trained to do some pre-defined conversation. So, it will not be able to fulfil all types of conversation as a normal human being can.*

Some of the key benefits of chatbots are –

* Customer Service – Almost the entire service sector (banks, insurance, college etc) have deployed chatbots in their commnuication channels as a 1st touchpoint to their customers.
* Cost Effective – Once build & deployed, running cost is almost neglible.
* Automation – zero human touch
* Time to launch – time to build & deploy is in days. Also these applications can be easily integrated with whole host of channels. (company website, facebook messenger, twitter etc)

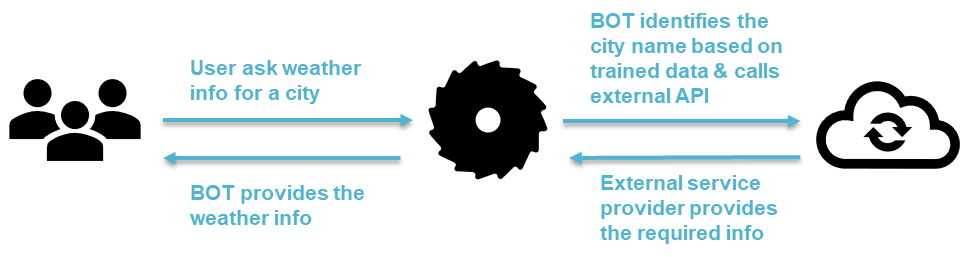
# Intro to Microsoft LUIS

Microsoft LUIS (Language Understanding Intelligence Service) is Microsoft’s cognitive AI product. It is a cloud-based API service that uses machine learning and natural language processing techniques. LUIS provides both prebuilt models as well as custom models.

# Building the BOT

## Problem Statement

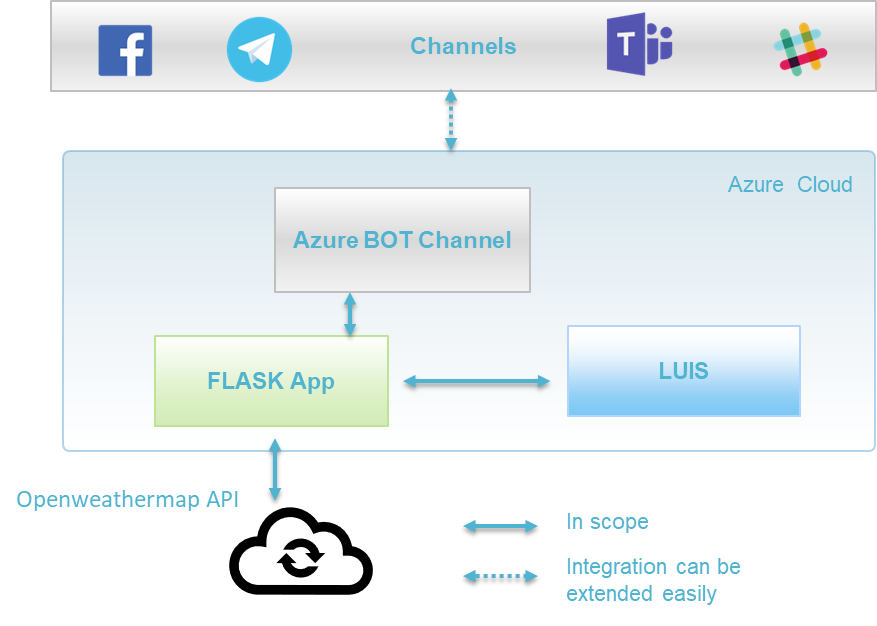
The following diagram depicts the user flow that we are trying to achieve in the BOT using LUIS.



Have taken up a very simple use case. There can be many possibilities. If BOT is unable to identify a valid city name, then it should have a fallback option. Fallback option is not shown in the diagram.

## Architecture

In order to achieve the above the following diagram depicts a schematic view of the underlying design. Every section of the design will then be discussed in a sequence.

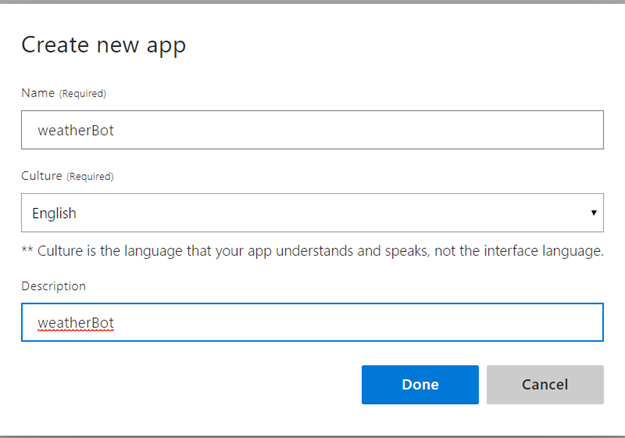


As depicted above the following are the core components of the architecture –

* LUIS - This application will be used to train on the user inputs.
* FLASK App – this is Python flask-based application (REST API) which performs the business logic of providing the necessary inputs to LUIS and external service provider
* Openweathermap - This is a 3rd party service provider which exposes API to the external world and provides information on weather
* Azure BOT channel – This is a channel that Azure provides which can be connect a flask REST API end point and other front-end communication channels
* Channels – this part is shown for illustrative purpose. Azure BOT channel has inbuilt capabilities of integrating with any other channels. Some of the channels are shown in the diagram only.
* Deployment - Lastly as depicted we will deploy the flask application in Microsoft Azure. The other components LUIS & BOT channel are already in Azure as both are Microsoft product.

## Training LUIS

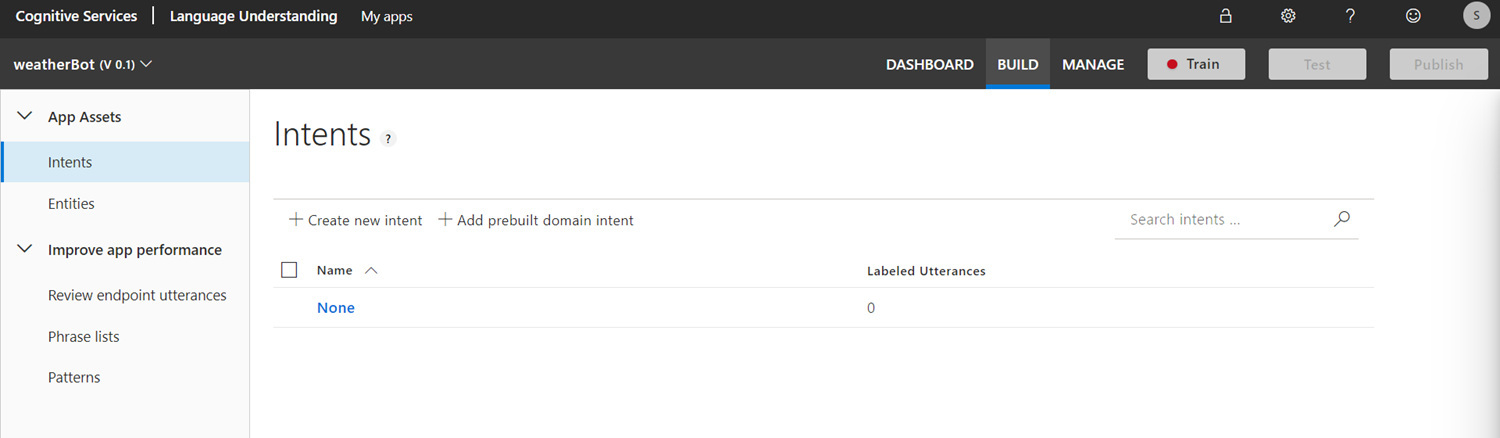
As a pre-requisite create an account at <https://www.luis.ai> before we can start working on LUIS



Once account is created please login to create a new app. As a 1st step click on create new app. Fill in the detail as depicted. Once the app creation is done it will be shown in the My App section. Click the app to proceed with the next set of actions.

Note LUIS support multiple languages. For this project we have chosen English.

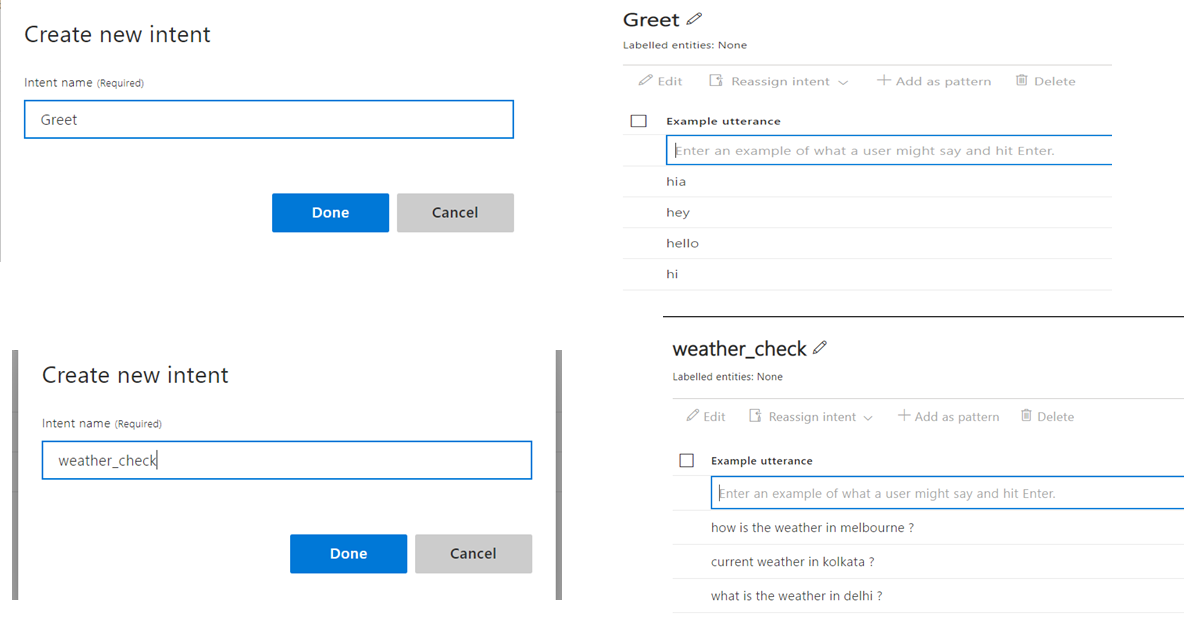
In the next step we will be training the app. Once you open the app the below view will come.



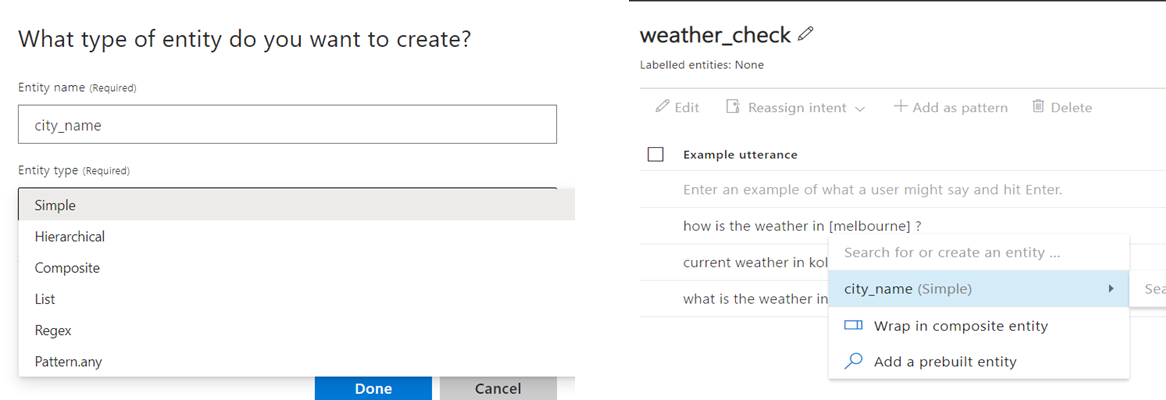
On the left-hand panel there are 2 key features. Intents & Entities. Intents are user utterances during the conversation cycle. Entities are keywords which will be utilized by the BOT to process further action.

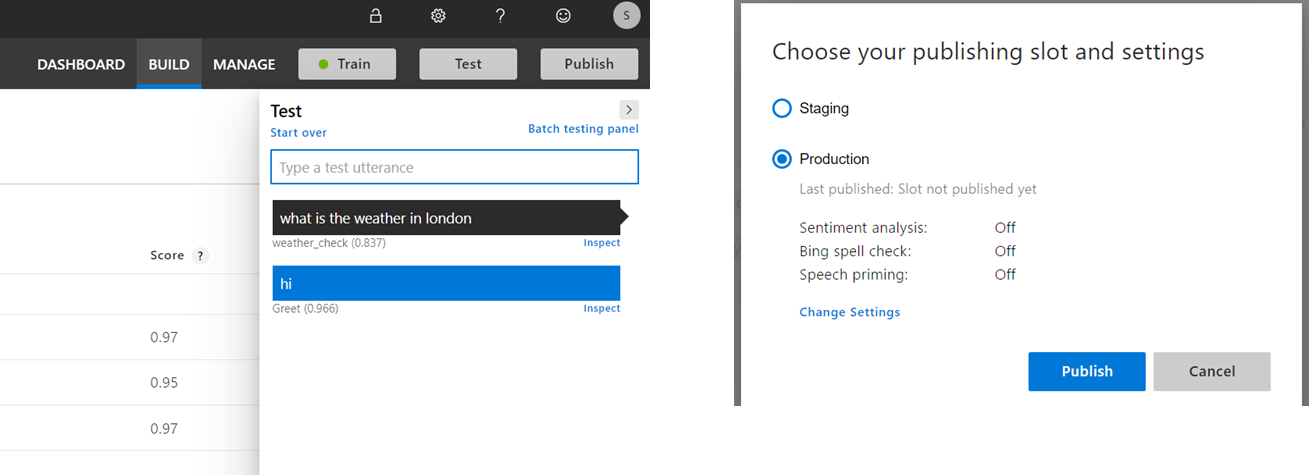
For our project we have created 2 intents as depicted below. As seen below for each of the intents we have created example utterances that an user might ask. It will be the task of the BOT to identify the intents based on the utterances.

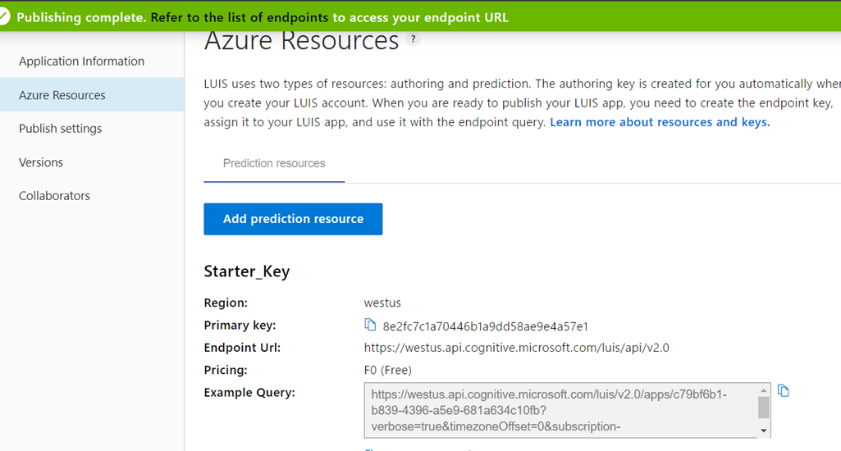
Note – In practical world we need to put a variety of utterances which matches the use case (intent)



As a next step we need to create Entity and co-relate the same to the user utterances of the Intents. In our project user is interested to get info on a particular city. So cityname is of critical importance. We will create city name as an entity and will map the city name in the utterances as shown below. This will help the machine to learn and identify the city name in a sentence. In the production scenario user can give any city name. So based on the machine learning, BOT will recognize the city name from the user request.

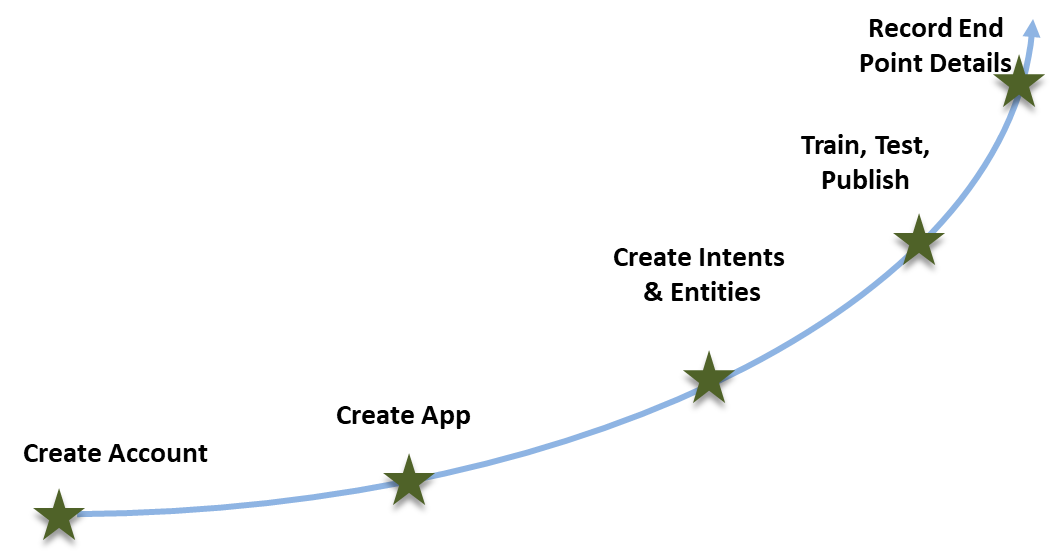


The next stage is to execute the train, test & publish. In the dashboard there are separate buttons for the same. The below figure shows the same. In the test, we see BOT is able to clearly identify the intents based on the user utterances. During publish select the production as shown.



Finally, we are at the end of the LUIS part. Once we have published, the resources tab will provide some valuable information around key, end-point URL details. The figure depicts the same. These details will come in handy when we build the Flask App.

As always let’s do a recap of what all we have done in LUIS. The below diagram depicts the same.



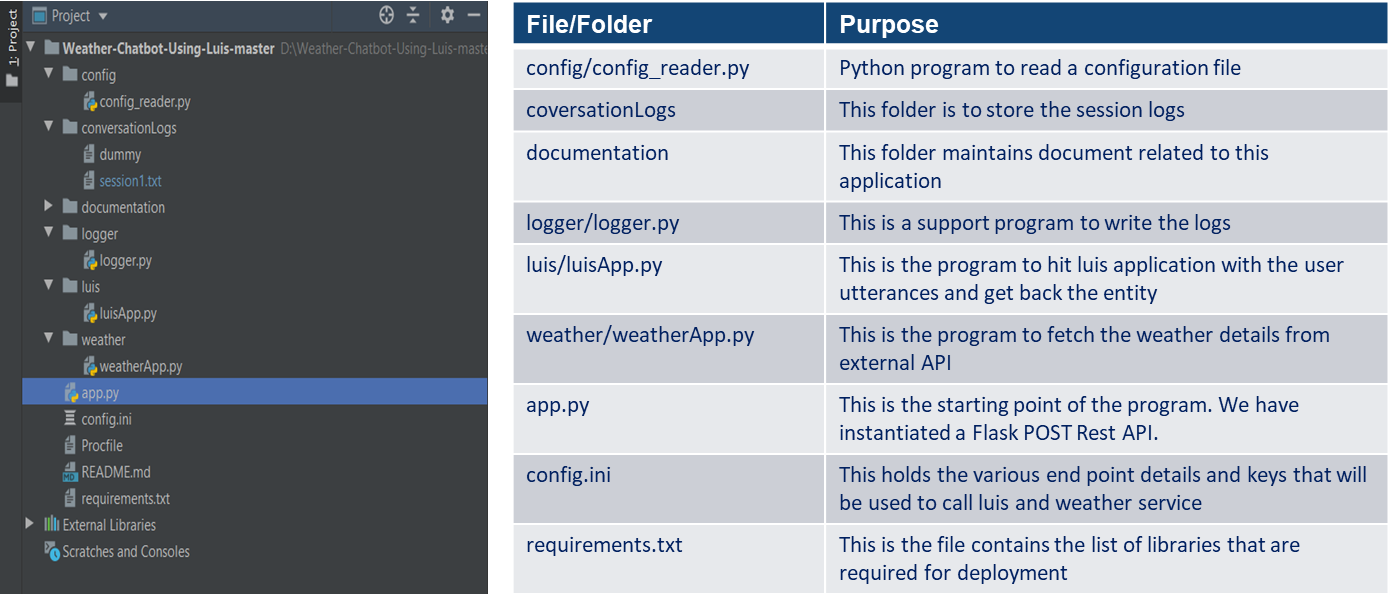
Creation of the relevant intents & entities is the most critical of this journey. Key focus should be there for this part.

## Subscribing with openweathermap

Before we jump in building the flaks app, we need to register in openweathermap. Login to the URL. <https://home.openweathermap.org/>. Create an account and click on the API Keys section. There will be a default key already generated. This will also come handy in our flask app.

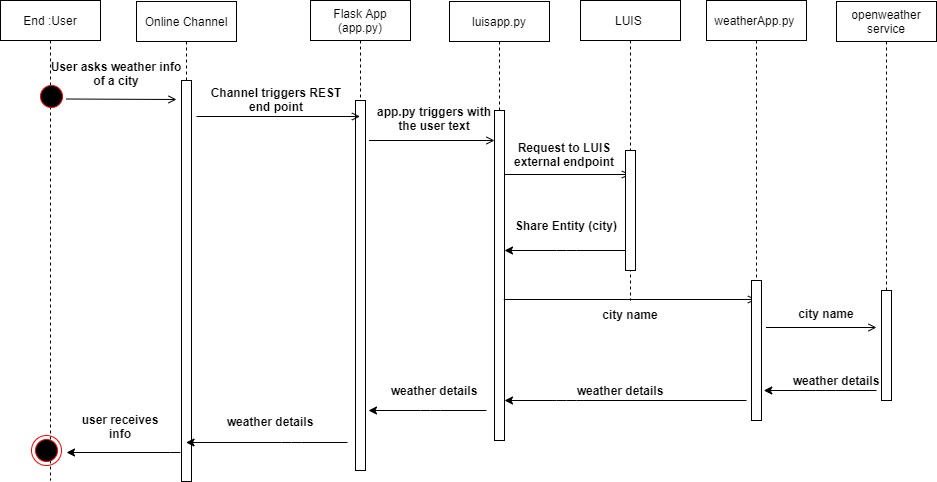
## Building the Flask App

This is the heart of the overall BOT application. The full code is available in the [Github account.](https://github.com/sidbhat1979/weatherchatbotusingLUIS/tree/master/Weather-Chatbot-Using-Luis-master) The below is the list of files created for the project. Have used PyCharm as the IDE for python. Please refer to the appendix on how to install PyCharm. It is recommended to create a new conda env for this project. You can create a new environment using PyCharm easily. Please note this program requires quite a lot of libraries. All are listed in the requirements.txt file. All are simple pip installations.



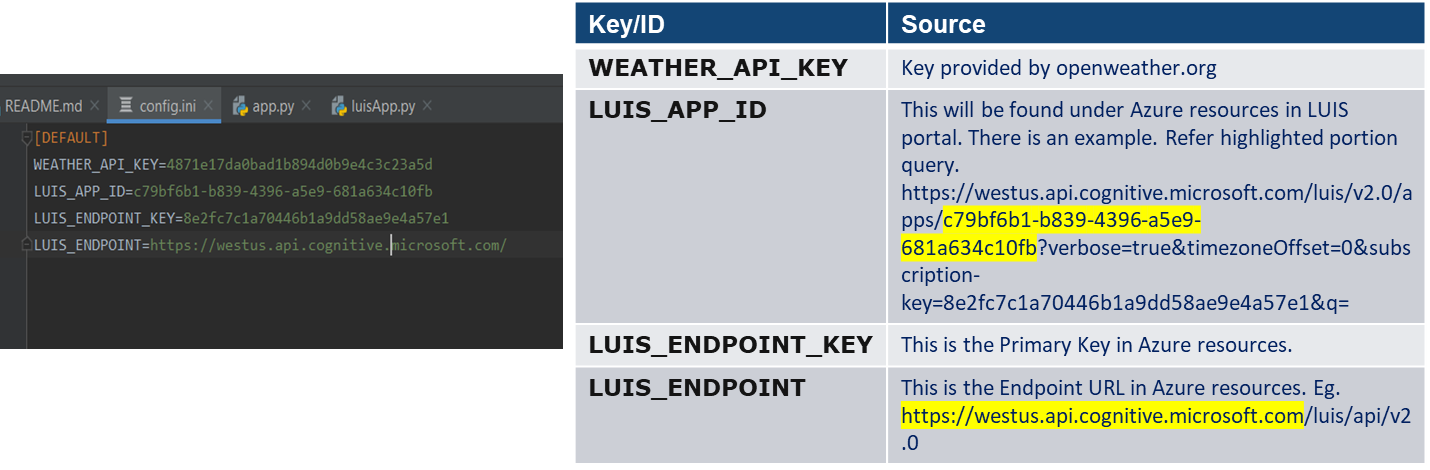
### Sequence Diagram

The below diagram depicts the sequence of action. Please note the online channel is shown for illustrative purpose. The flask app will provide a Rest end point (POST Method) which can be consumed by any application.



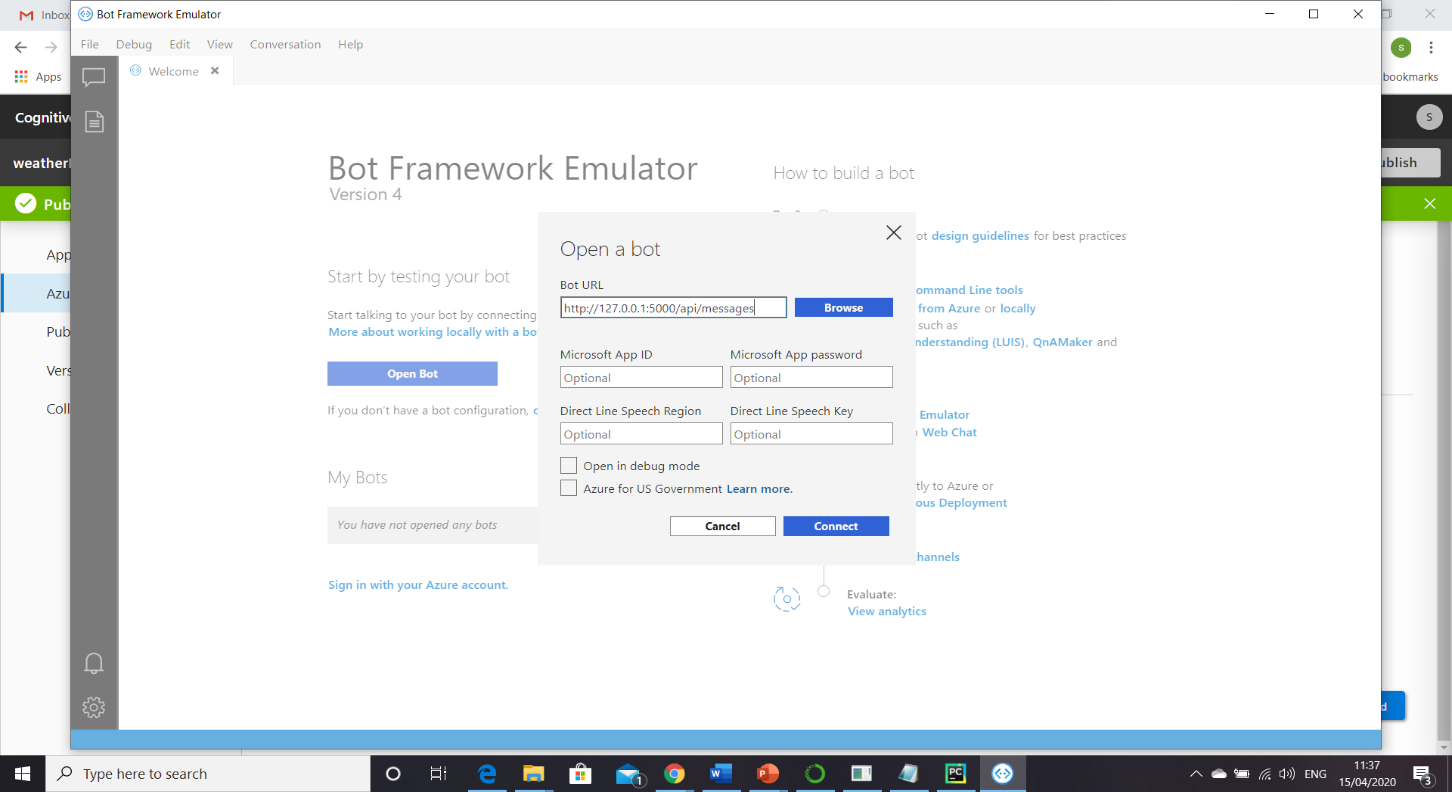
### Config File Settings

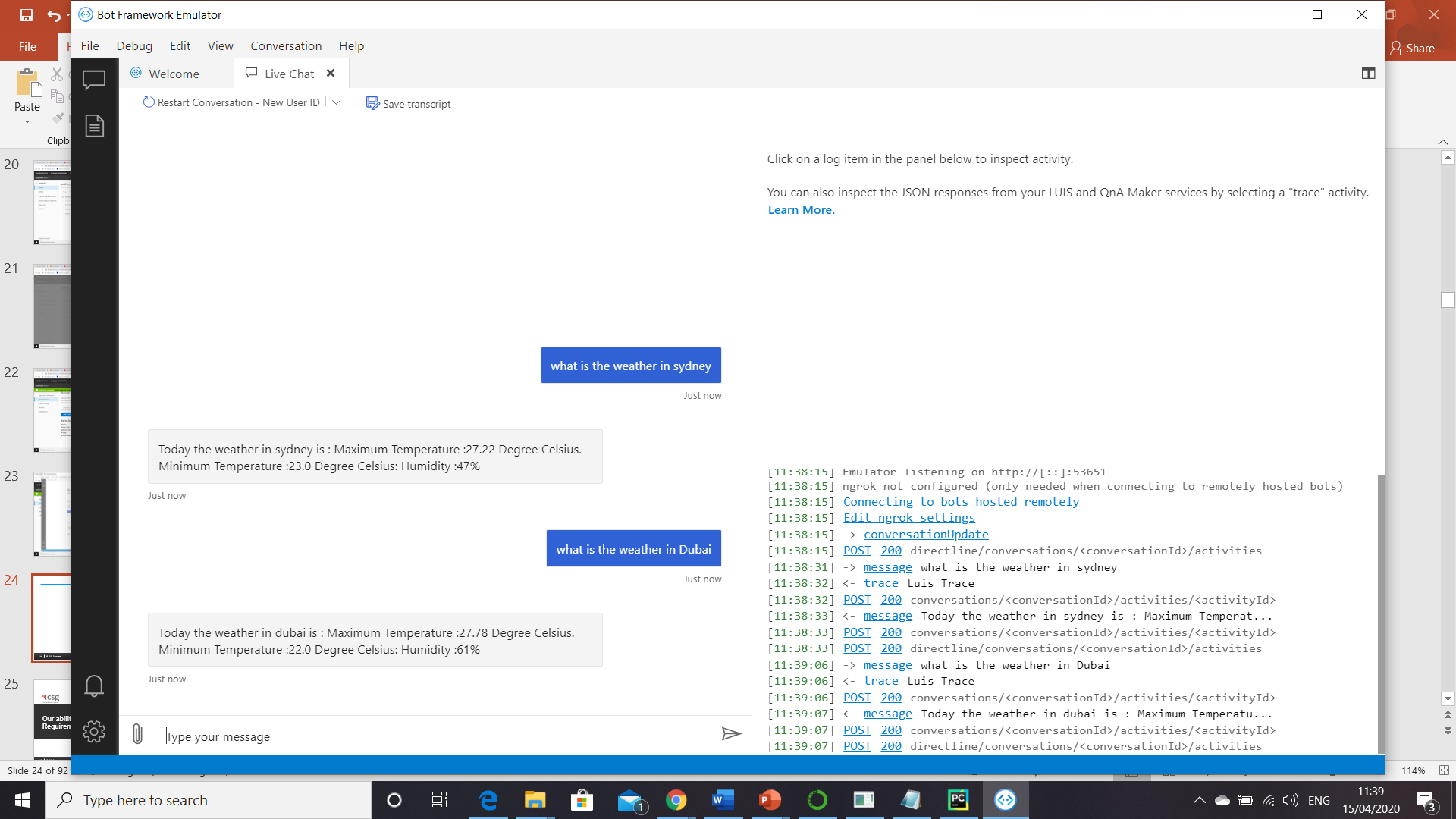
The below table illustrates the config.ini settings. This file is very key and helps in all the external integrations.



### Testing the Flask app

Before we deploy the application, it is better to test the flask app locally. In order to do the test, we need to install Microsoft BOT emulator. Follow the steps below.

1. Visit <https://github.com/Microsoft/BotFramework-Emulator/releases> and download the Bot Emulator setup file based on your computer settings.
2. Once downloaded install the file
3. Open the BOT emulator. As shown in the figure provide the BOT URL. When you execute the app.py file in PyCharm, it will provide an URL. Add api/messages to that URL. (this is the URL wherein the REST end point is listening)
4. Finally hit connect. You can now chat with the BOT. As you see BOT is able to fetch weather info for different cities which are not part of the original training of LUIS. So LUIS is able to identify the entity (city\_name) properly. BOT is ready for deployment.

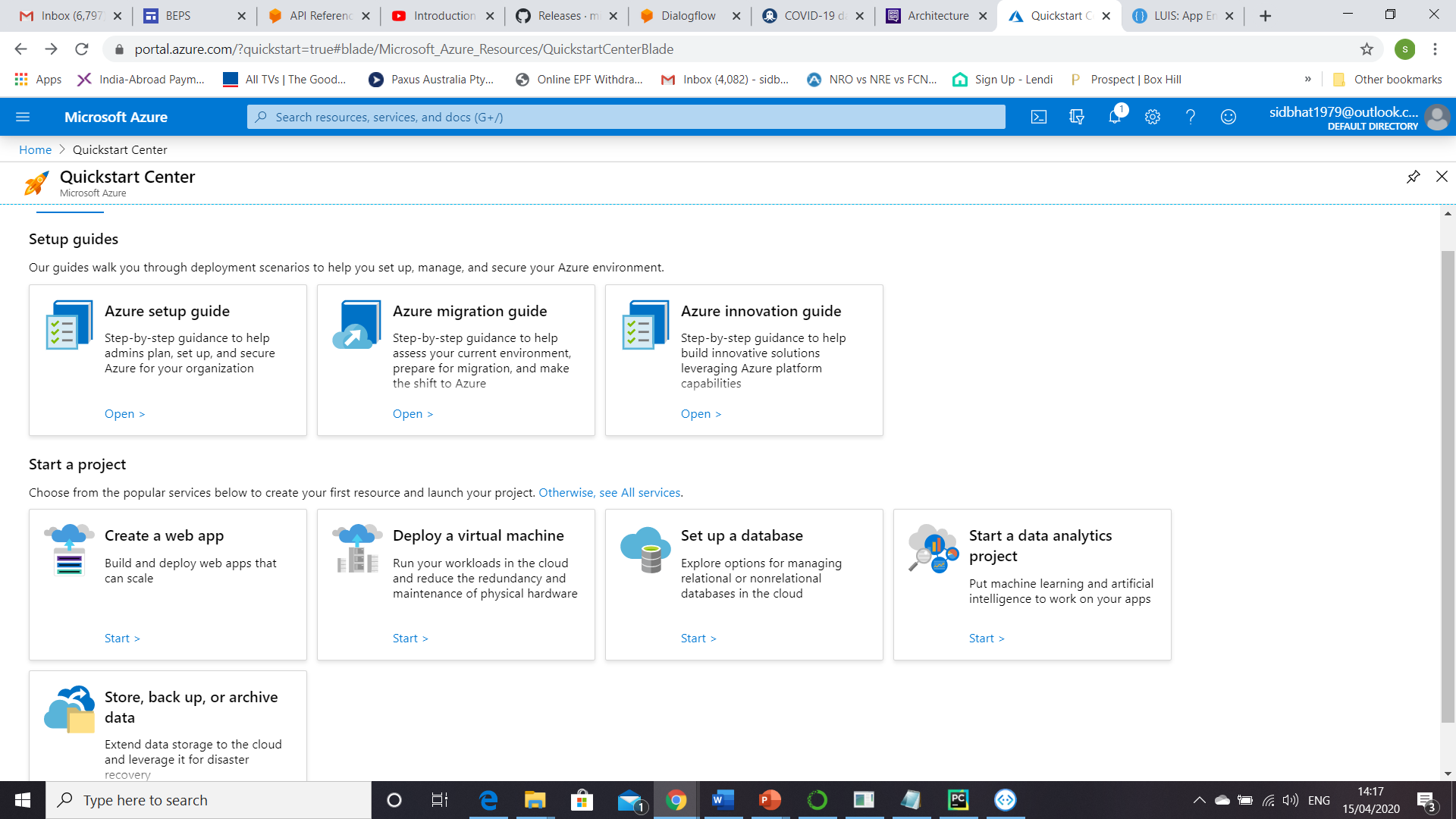


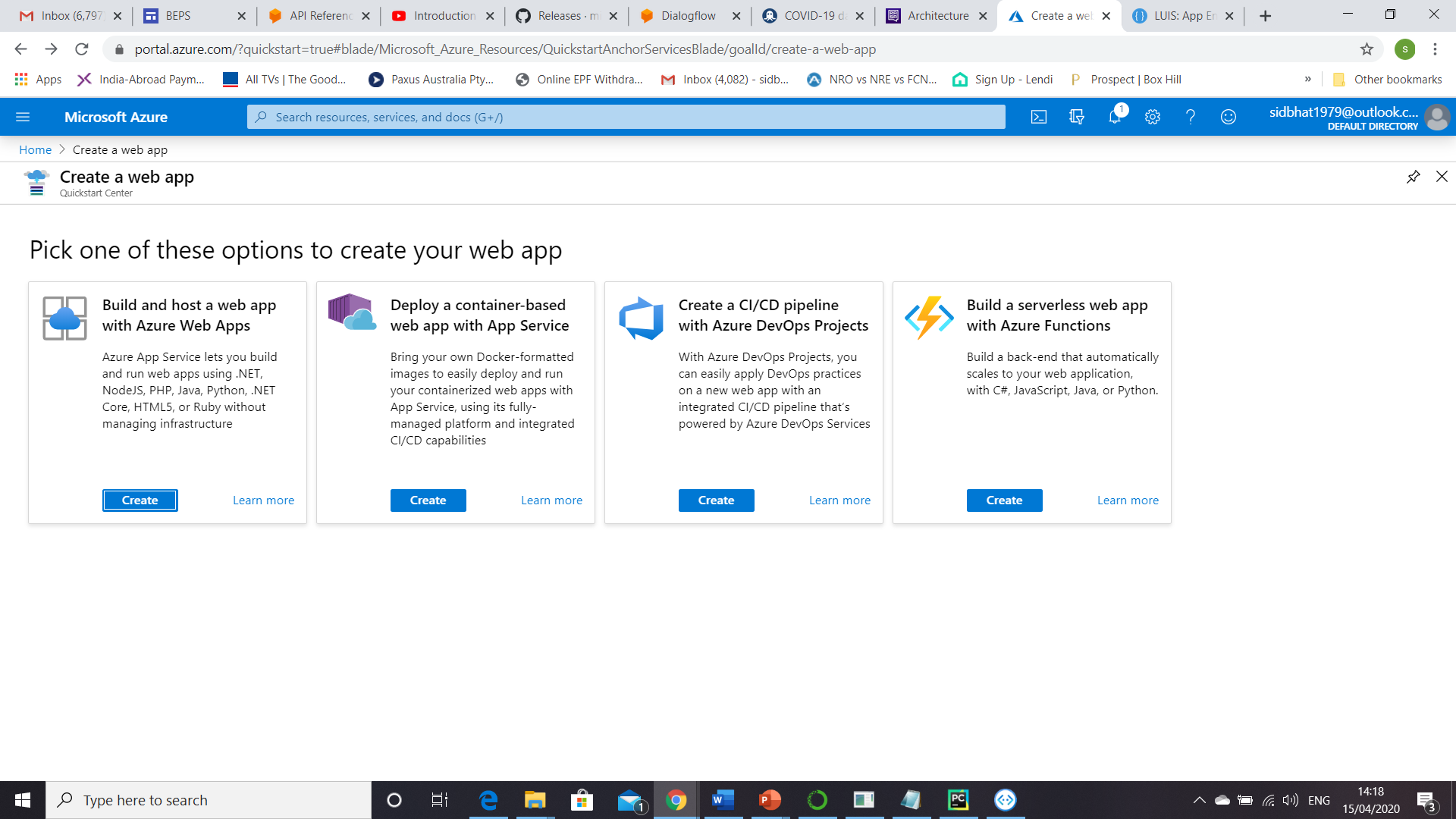
## Deployment in Microsoft Azure

The next logical step is to deploy the application in cloud so that it can be exposed to the external world.

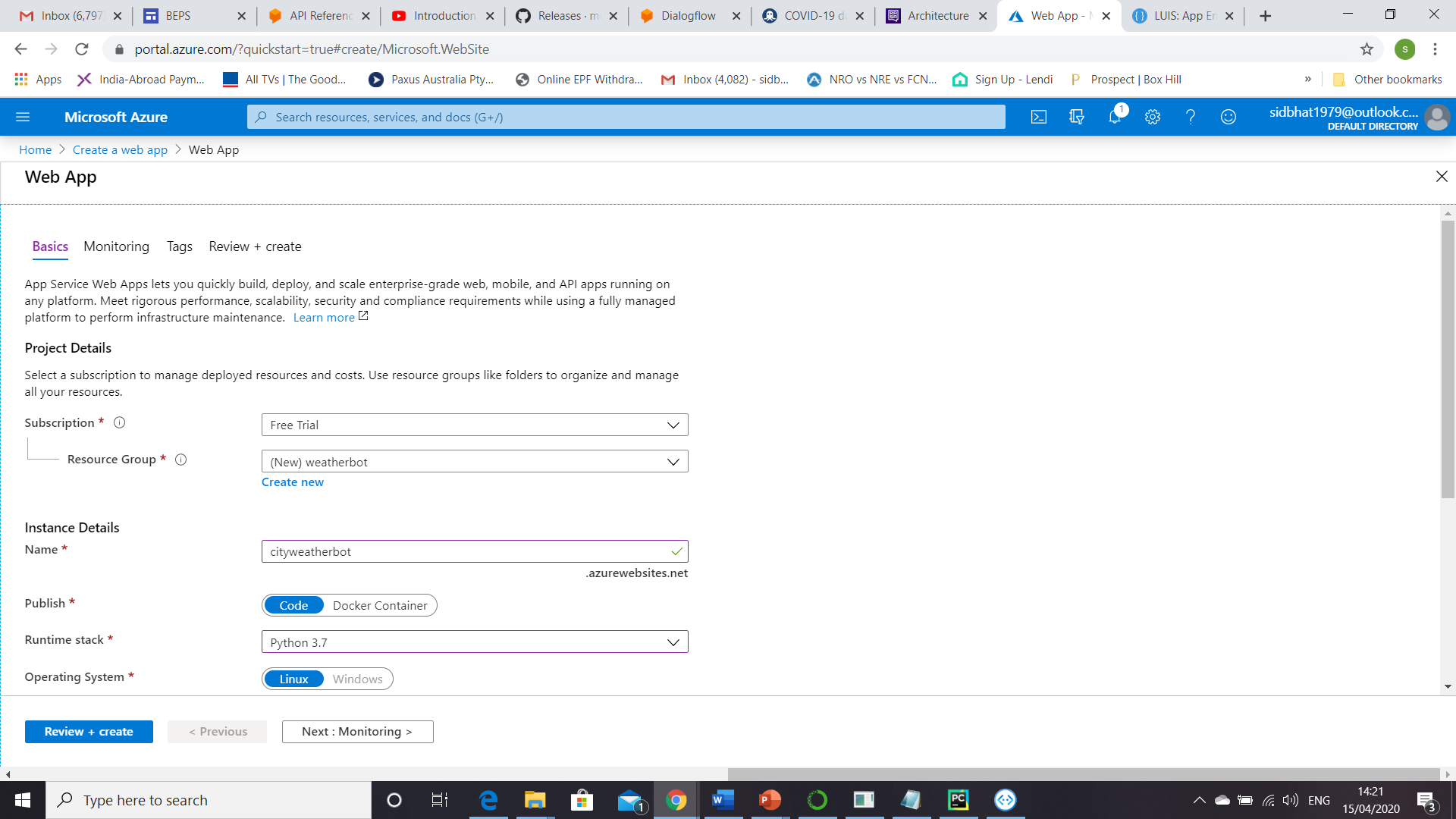
Follow the following steps to perform the deployment. There are 2 parts to the deployment. One is setting the Azure part. The other one is using git to push the code to the Azure.

### Setting up Azure

Login in to the Azure portal and create a web app.

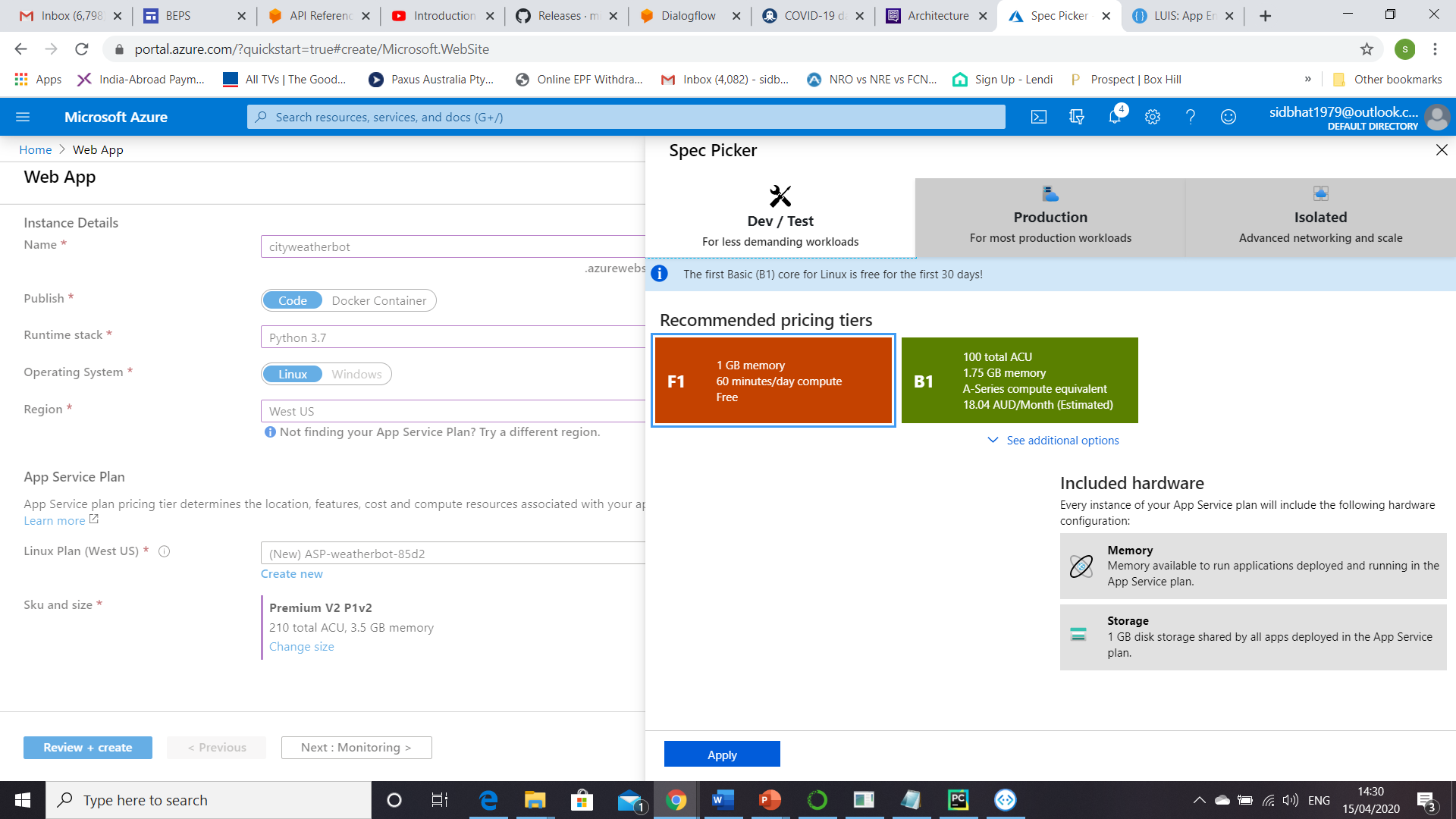
In the next screen select “Build and host a web app with Azure Web Apps”

As a next screen we need to fill up the details related to the web app. It is quite simple. We will go for a free trial option. Refer to the screen shots for the details.



Note:-

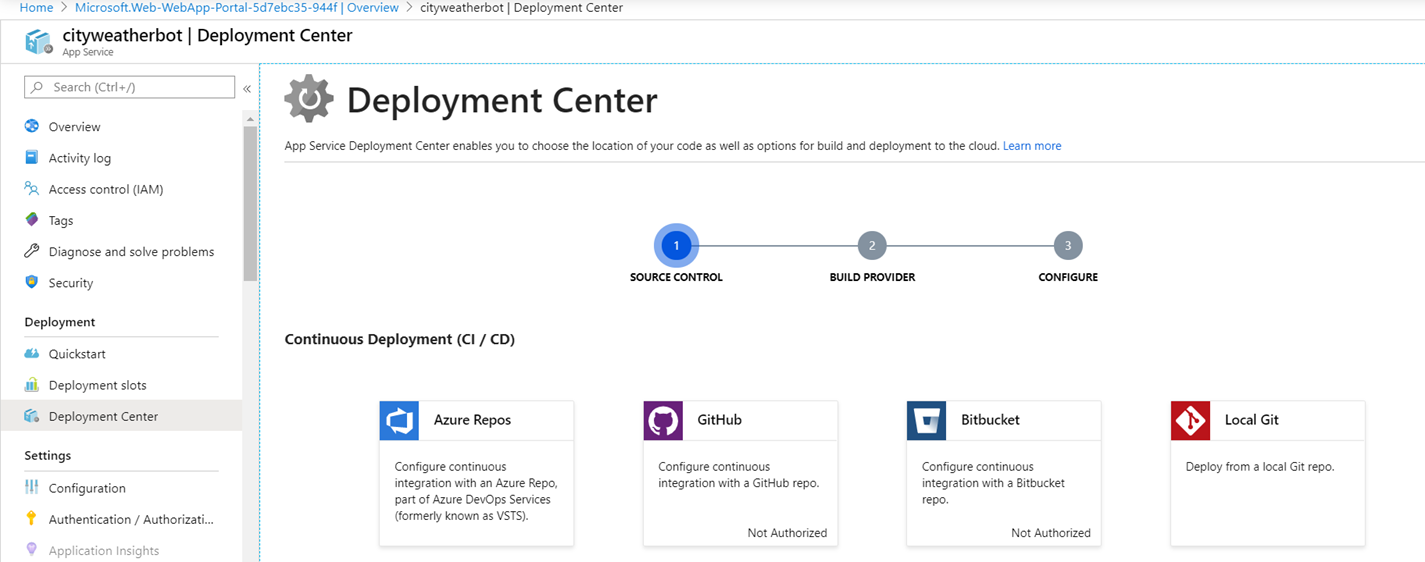
* Need to create a new resource group if not created.
* Runtime stack – This will be Python 3.7 as our flask app is on that version

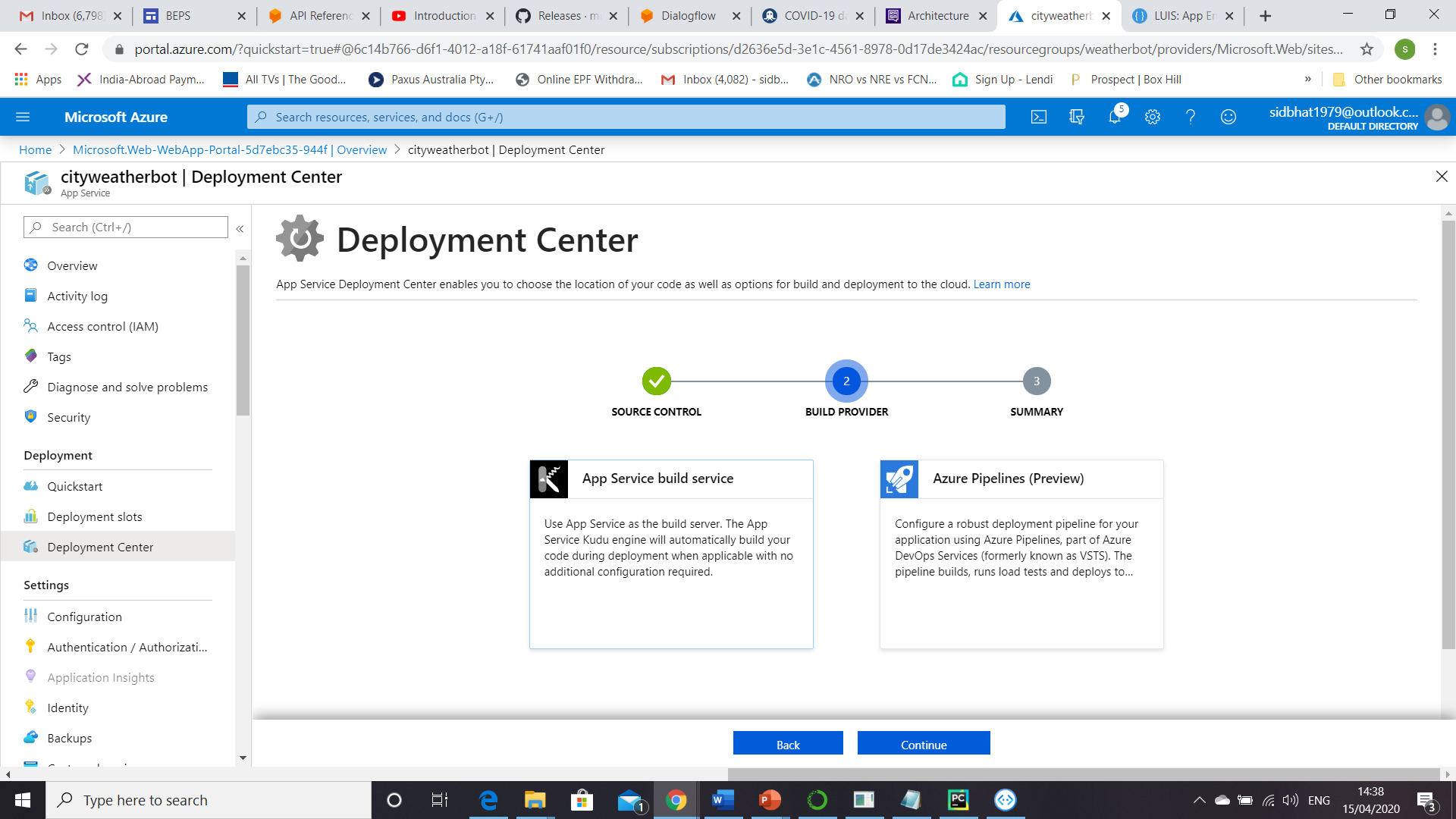


Note:- In the Sku and size pls change it to 1 GB memory as it is free.

Post this click review+ create to create the app. It will take some time to create. Once the app is created, we need to set the deployment settings for this app.

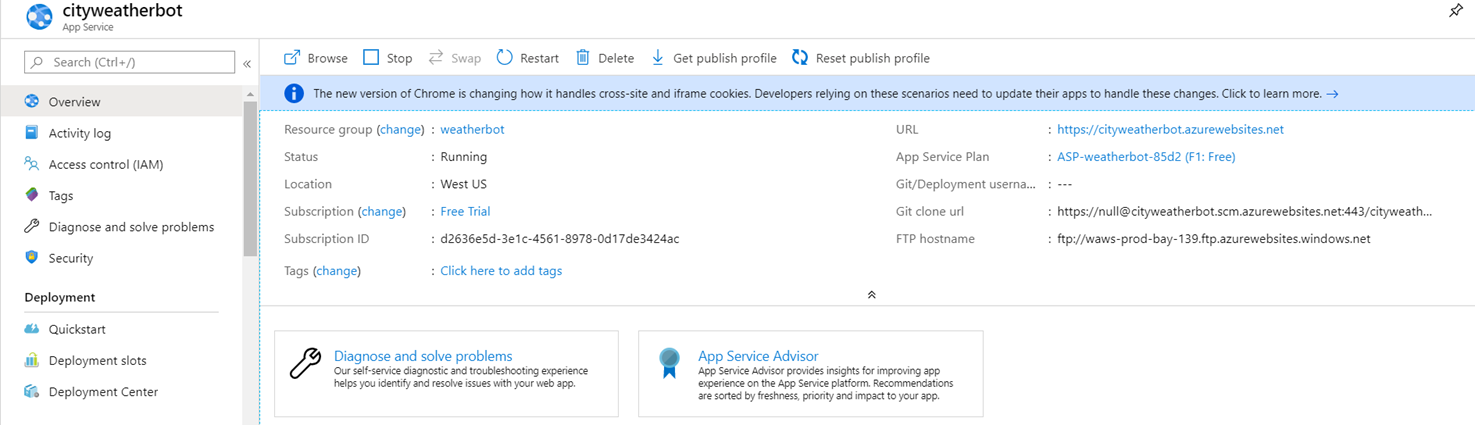
Please go to “deployment center” as shown below. Opt for ‘Local Git’ option & move.





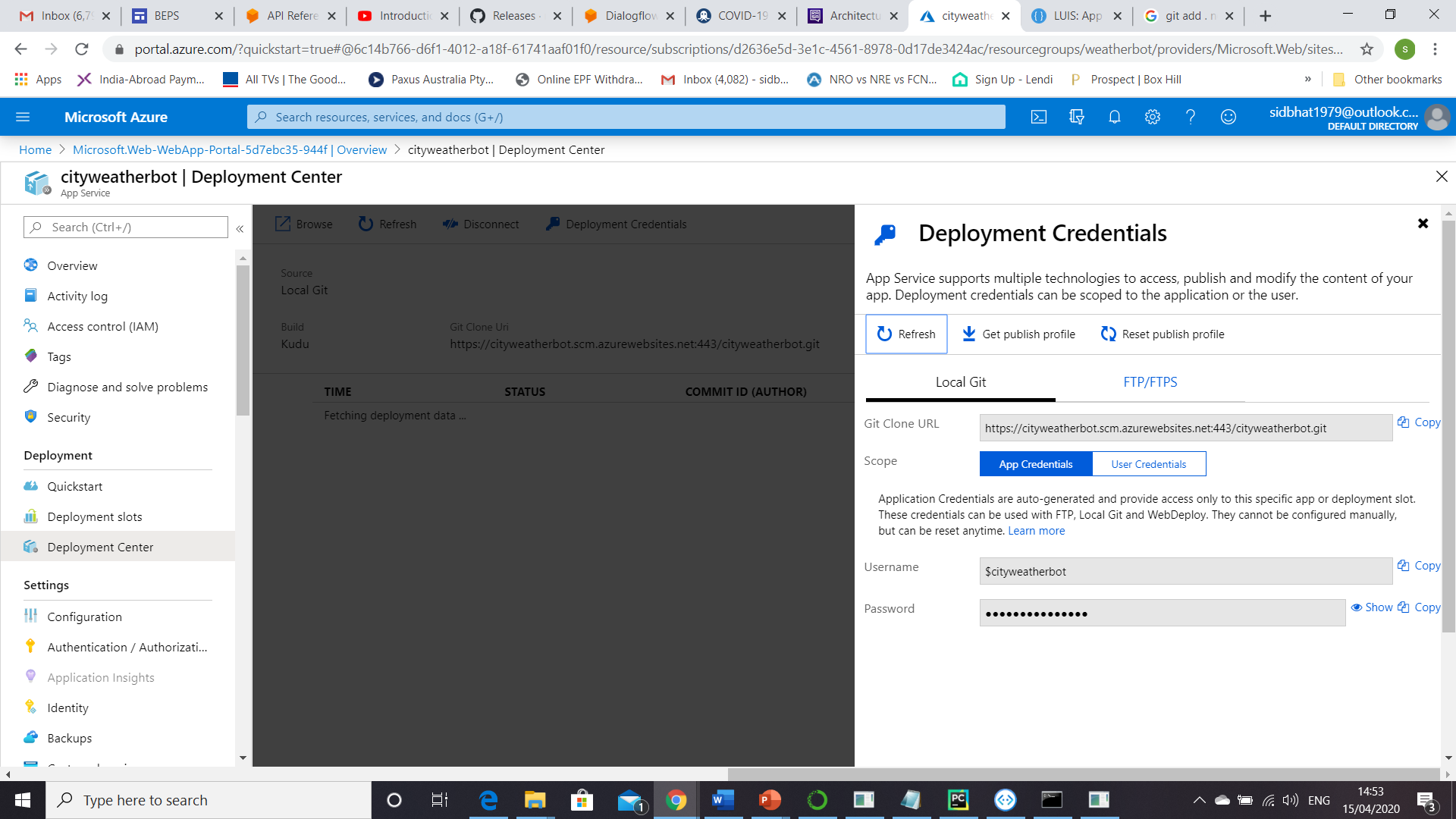
In the next screen opt ‘App service build service’. Click finish to end the deployment settings.

Once the setting is complete the overview section will be seen as below. The Git clone url will be important for the next step.



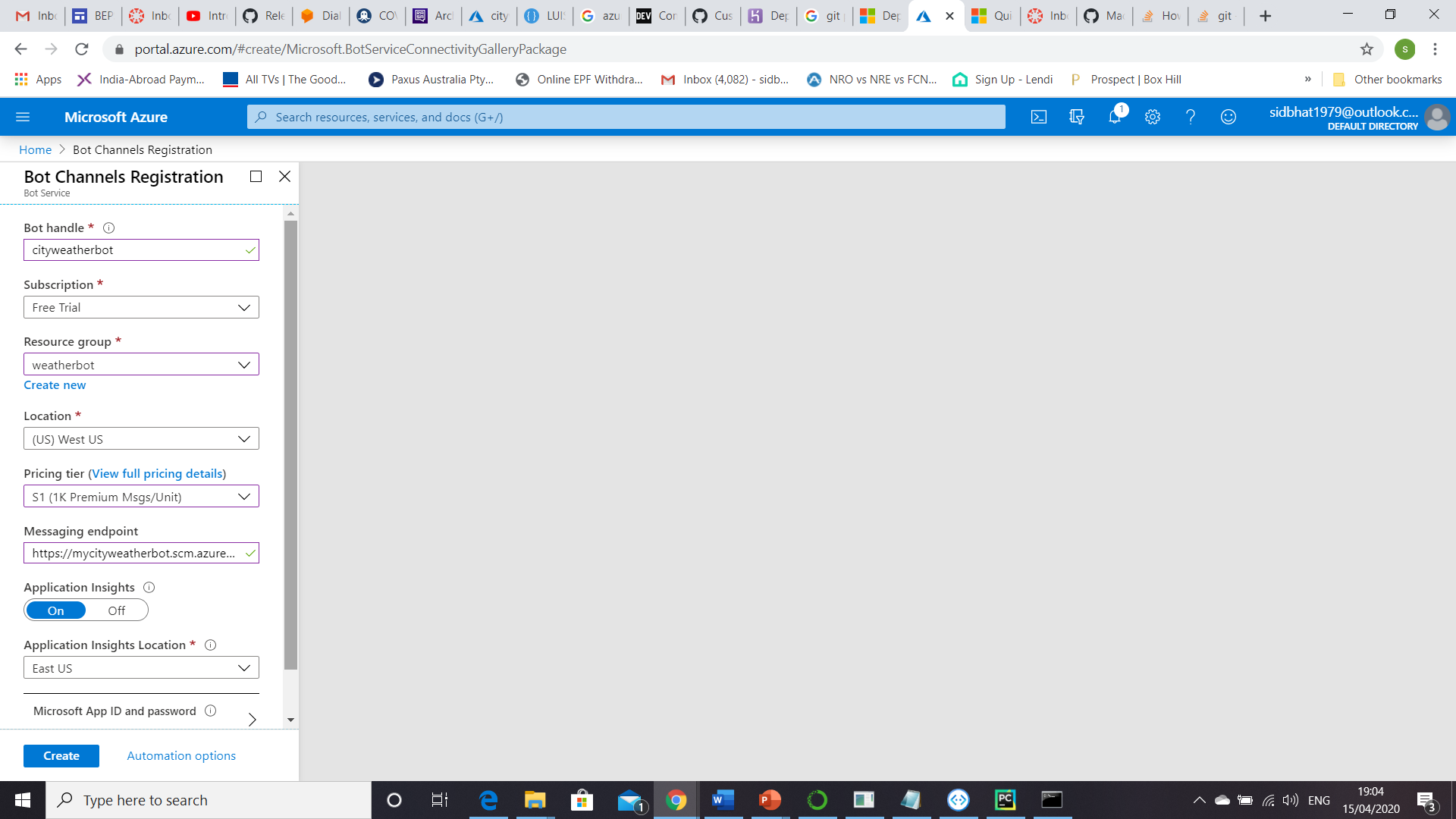
### Setting up GIT

You need to install GIT in local machine before executing the below steps

1. Open a command prompt in your local machine
2. Change directory to the top folder of your project
3. Run git init to initialize an empty git repository
4. Create a new remote git alias using the command: git remote <alias> <git clone url> git clone url is the same as we have seen in the overview section above.
5. Use git add . to add all the files to the local git repository.
6. Use git commit -m “my first commit” to commit the code to the git repo.
7. Push the code to the remote repo using git push <alias> master -f
8. This prompts for a username and password. Go to the ‘Deployment Credentials’ section and copy the username and password to enter in the prompt. Once the credentials are correctly entered, the app deployment to azure is completed.

## BOT Channel Registration

This is an independent step of the deployment. You can not we can deploy the flask app to any cloud environment and then do the BOT channel registration in Azure.

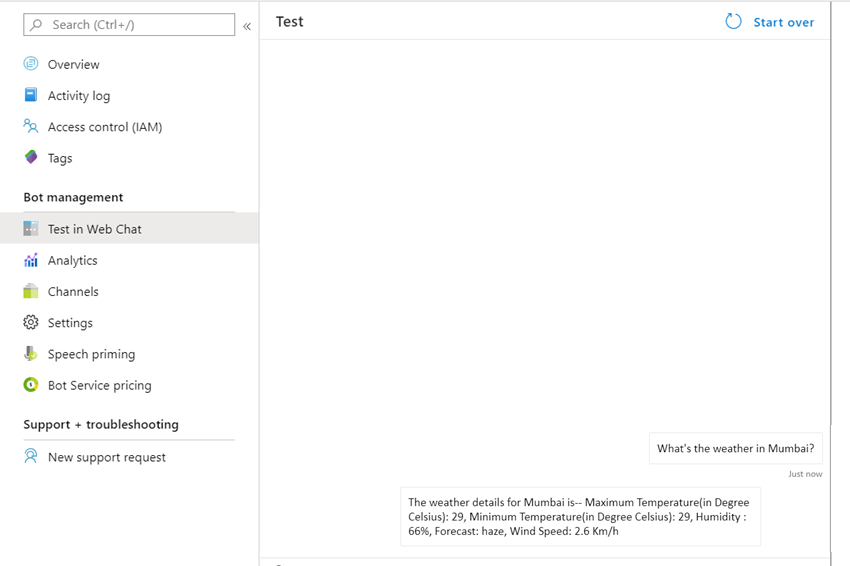


Step 1 – In the Azure portal, create a bot channel registration. Search for bot channel registration. The page will open up.

Step 2 – Need to provide the details as requested.

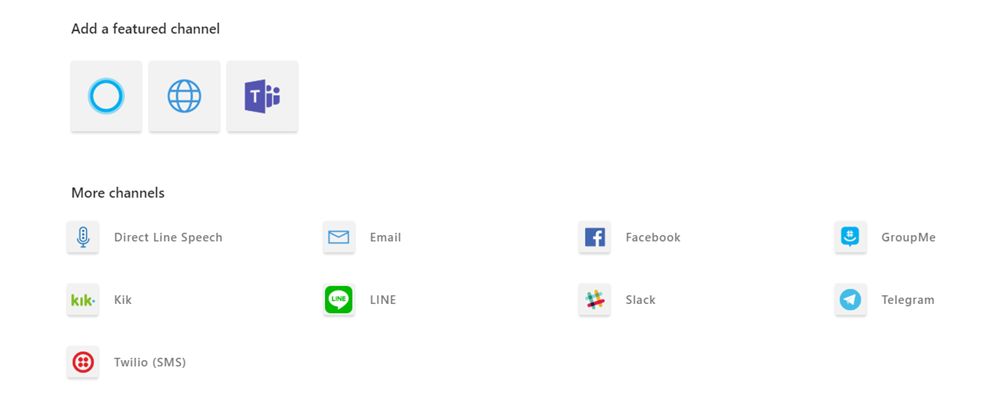
For Message endpoint, provide: <URL from the web app created through GIT push> + ‘api/messages’.

Step 3 - Click Create to create the bot.

Step 4 - Once your web channel registration gets done, open the bot and then click ‘test in web chat.’ If the chat works fine, our deployment is a success. Below is a quick snapshot of the test. It is working. So deployment is a success.

# Third Party Integration

Azure BOT channel can be easily integrated with multiple front-end channels. PFB a view of integration channels which are provided by Azure.



You can refer to the azure docs of how we can integrate with each of these applications.

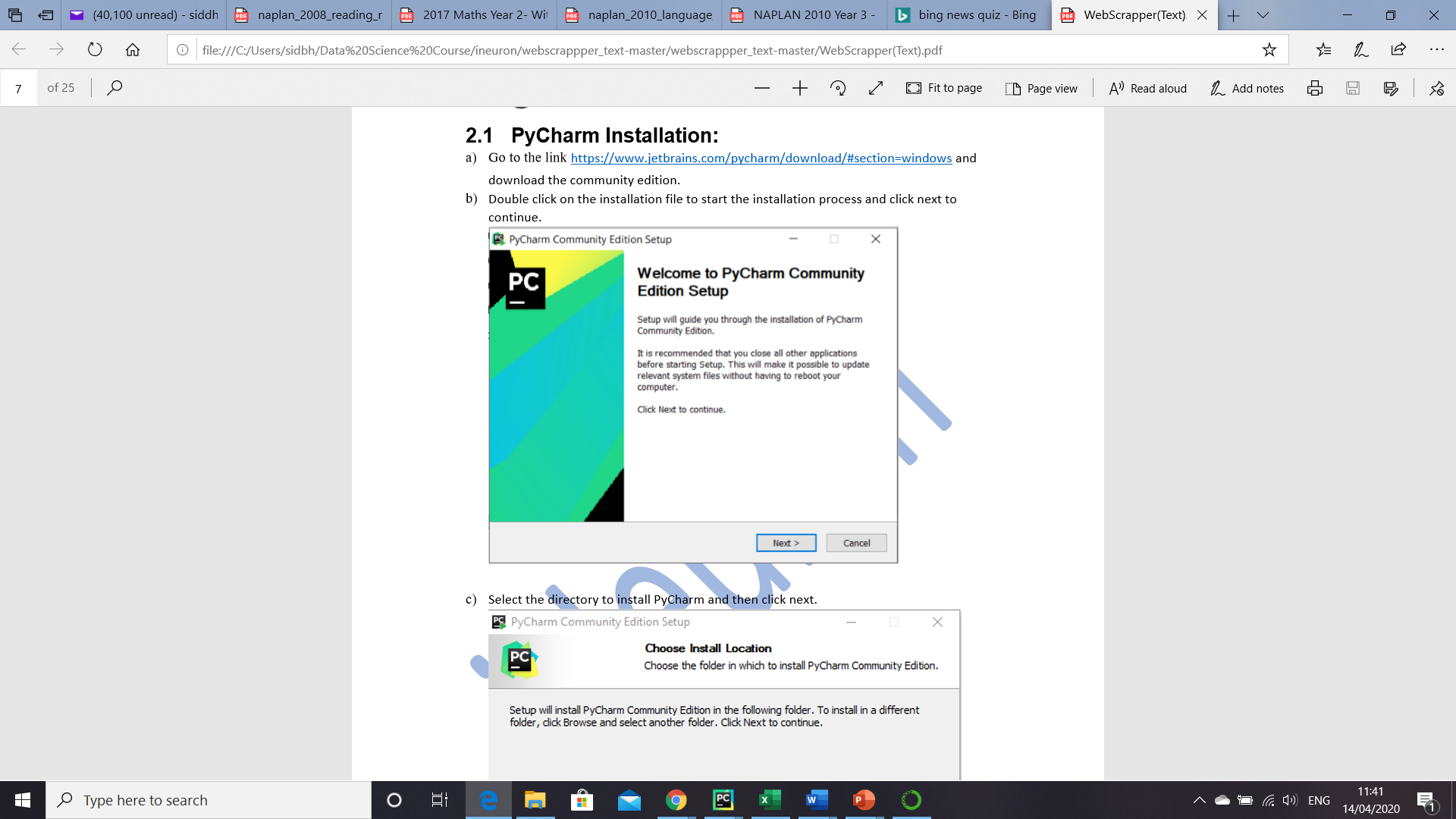
# Wrapping it up

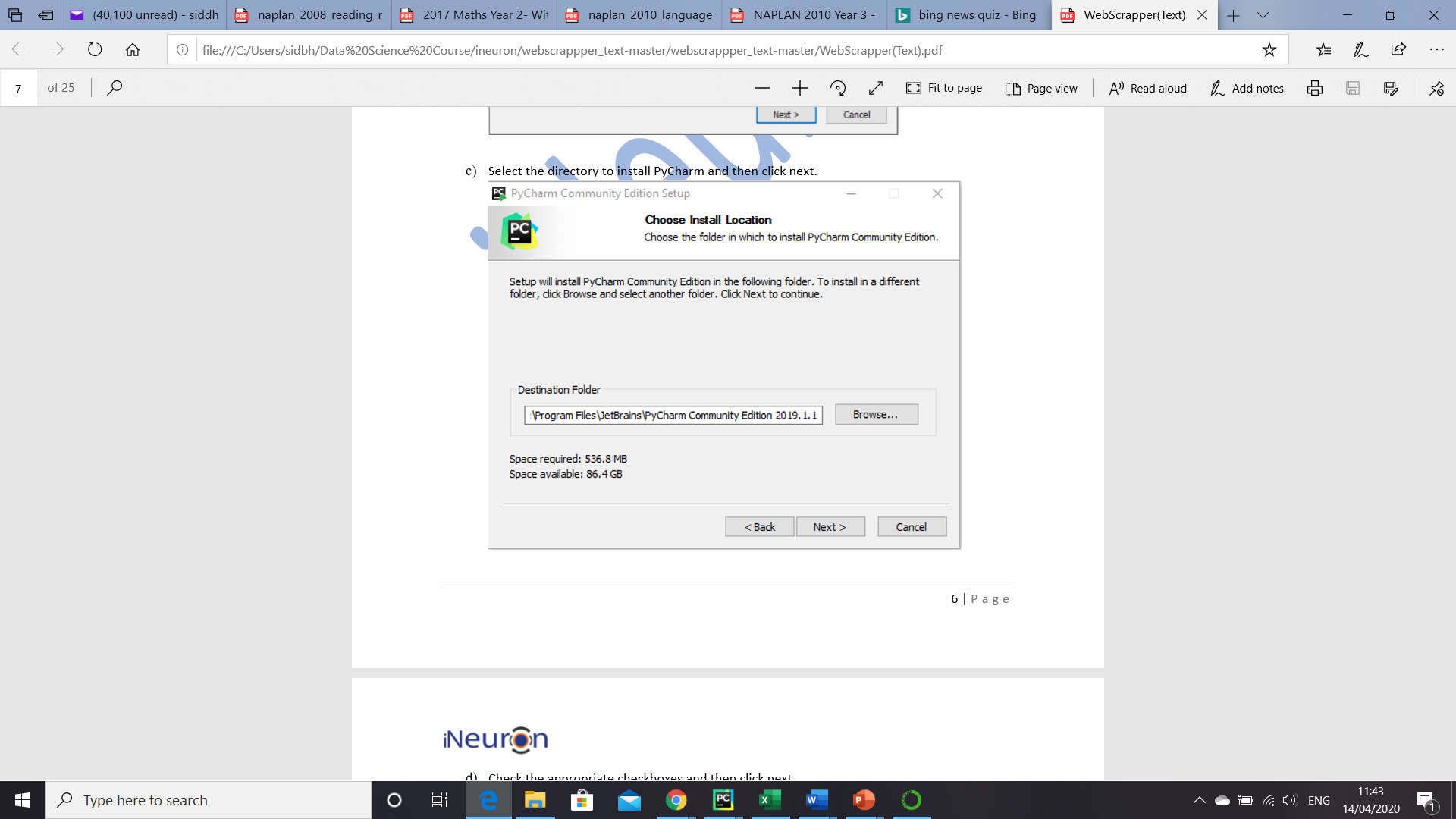
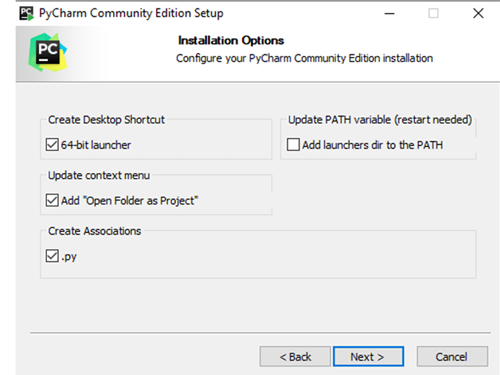
Finally, to end let’s have a look at the key steps to build the BOT & deploy. Hope you have enjoyed reading and got some directions to build your own BOT. Signing off.



Building the flask app is the key portion of this entire journey as it ties up all the parties. Special emphasis should be given on this milestone as this is the heart of the entire process.

# Appendix

Quick steps to install PyCharm.

1. Go to the [link](https://www.jetbrains.com/pycharm/download/#section=windows and download the community edition.)
2. Double click on the installation file to start the installation process and click next to continue.
3. Select the directory to install PyCharm and then click next
4. Select the appropriate checkboxes as shown below. Don’t miss to check the option “Open Folder as Project”
5. Choose the name of the start menu folder and then click on install to finish the installation.