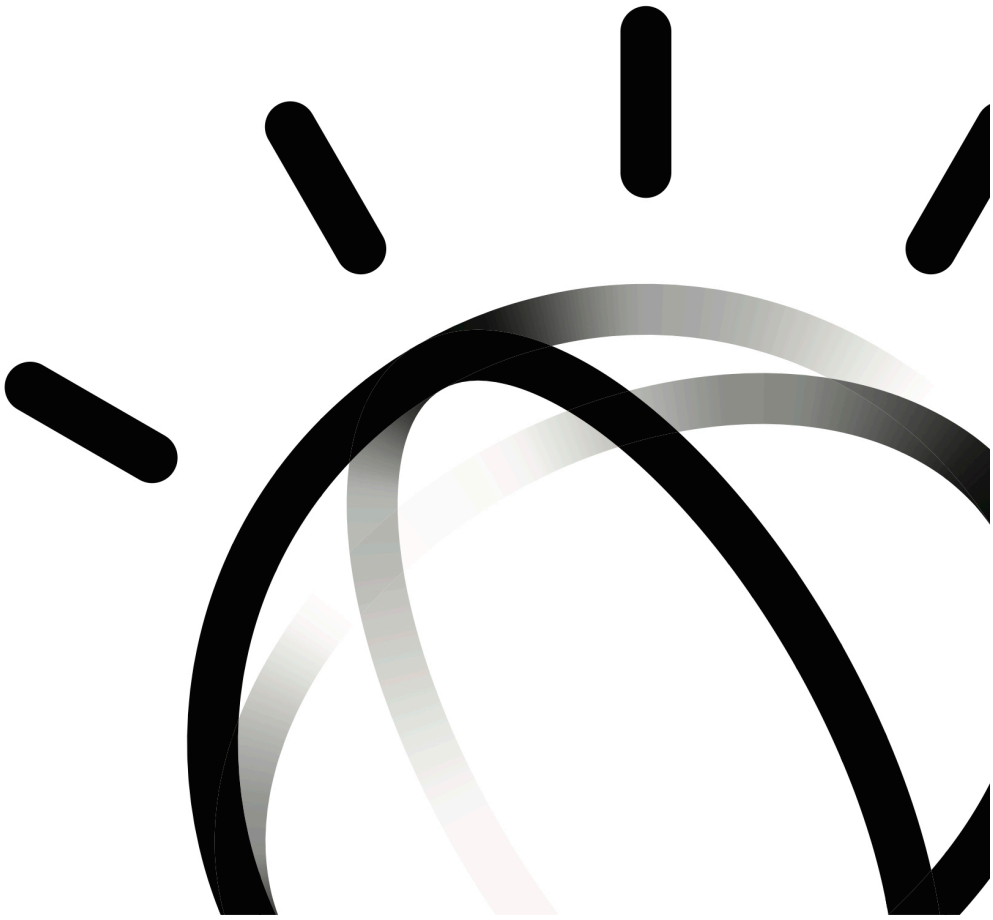
IBM Watson Solutions

Business and Academic Partners



Developing a Chatbot Using the IBM Watson Conversation Service

Prepared by Armen Pischdotchian

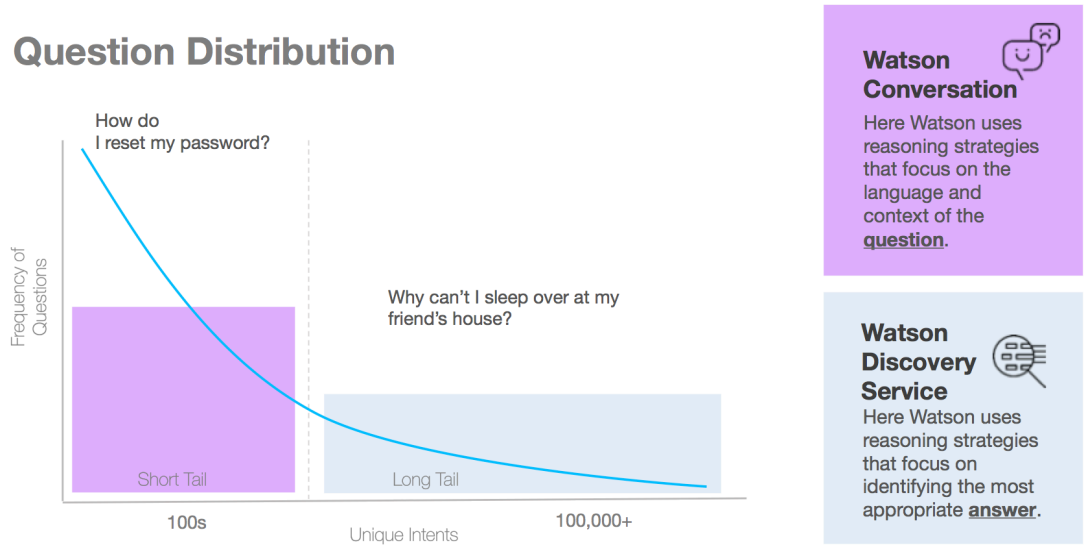
Version 8.0 November 2017

Overview

This guide is an instructional approach to working with the IBM Watson™ Conversation service where you can create virtual agents and bots that combine machine learning, natural language understanding, and integrated dialog tools to provide automated customer engagements. Watson Conversation provides an easy-to-use graphical environment to create natural conversation flows between your apps and your users. Creating your first conversation using the IBM Watson™ Conversation service entails the following steps:

1. Train Watson to understand your users' input with example utterances: Intents and Examples
2. Identify the terms that may vary in your users' input: Entities
3. Create the responses to your user's questions: Dialog Builder
4. Test and Improve

IBM Watson Conversation service is designed to answer the short tail of typical question distribution. The Discovery service, another API on IBM Cloud can address not so commonly asked questions, where the answer may reside in ingested documents that specifically answer esoteric questions that the system gets to learn over time. Combination of both services is a wonderful solution for a robust virtual agent.



Consider the following scenario used in this guide: You are driving a cognitive enabled car. You ask in a natural way, the way you speak, for the car to do things for you, such as turn on wipers, play music and so forth. At one point, you ask the about the weather. In this workshop, you will connect the Conversation service with an external service from the Weather Company (http://api.wunderground.com/?MR=1) to inform you of real time actual weather in the exact location that you are sitting and performing this lab. The service obtains your location using the browser's geo-location capabilities.

It is strongly recommended that you watch this 14-minute video: https://youtu.be/ELwWhJGE2P8

At the end of this workshop, spend some time and consider what other services you can use to augment a better approach for bringing further cognition to your app; for example, Retrieve and Rank to extract information from specific documents (the long tail), or, you may want to include Speech to Text upfront and Text to Speech for the returned responses. Enjoy the cognitive journey you are about to undertake.

Prerequisite

This section provides instructions to help you get started quickly with the IBM Watson™ Developer Cloud services using Node.js as your programming runtime environment. To make it easy to get up and running with a functional application that uses the REST Application Programming Interface (API) for any Watson service, IBM provides a Node.js package with wrappers that simplify application development. The package includes simple command-line example applications to let you experiment with any of the available services.

Begin with the Pre-requisites document. At a minimum, you will need the following artifacts:

- **Obtain IBM Cloud credentials**
- **Install the Node.js runtime (select the Recommended for Most Users option)**
- **Download OS appropriate code-friendly editing tool (optional, but it is better than just Notepad)**
    - If you are using a PC, we recommend using **Notepad ++**
    - If you are using Mac, we recommend **Sublime Text**
- **Your browser must be geo-location enabled**

  You must have geo-location for your browser enabled. If you had turned it off, use the following steps to enable geo-locations services:

  For Mac users go to this link: https://support.apple.com/en-us/HT204690

  For Firefox browser complete these steps:

  1. Navigate to the site to that you want to enable the service
  2. Go to the *Tools* menu, then select *Page Info*
  3. Select the *Permissions* tab
  4. Change the setting for *Share Location*
- **You will need the API key from: http://api.wunderground.com/**

  You can register with the site and obtain your own key, or just use this key: `5aa2ca764b80f41a` when the time comes later in this document.

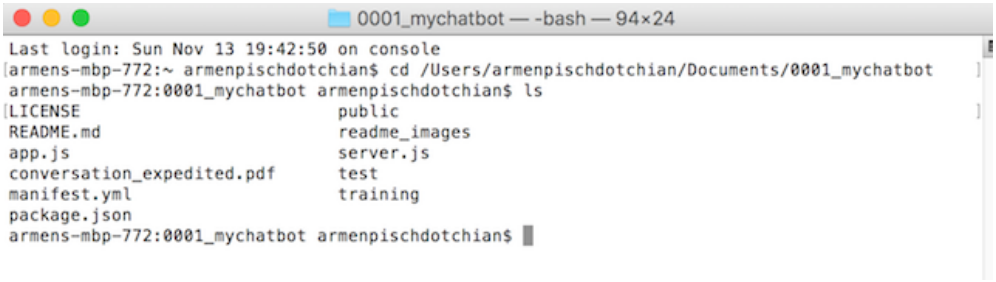
- **Install the cf command line (optional should you decide to deploy to IBM Cloud)**

  1. Direct your browser to a GitHub repository: https://github.com/cloudfoundry/cli/releases
  2. Download and install the most recent installer appropriate for your operating system.
  3. You may need to open Preferences → Security and Privacy → General tab (in Mac); unlock and change the Allow applications downloaded from **Anywhere**.

Using the Conversation service

Let's begin our journey with a few good housekeeping practices:

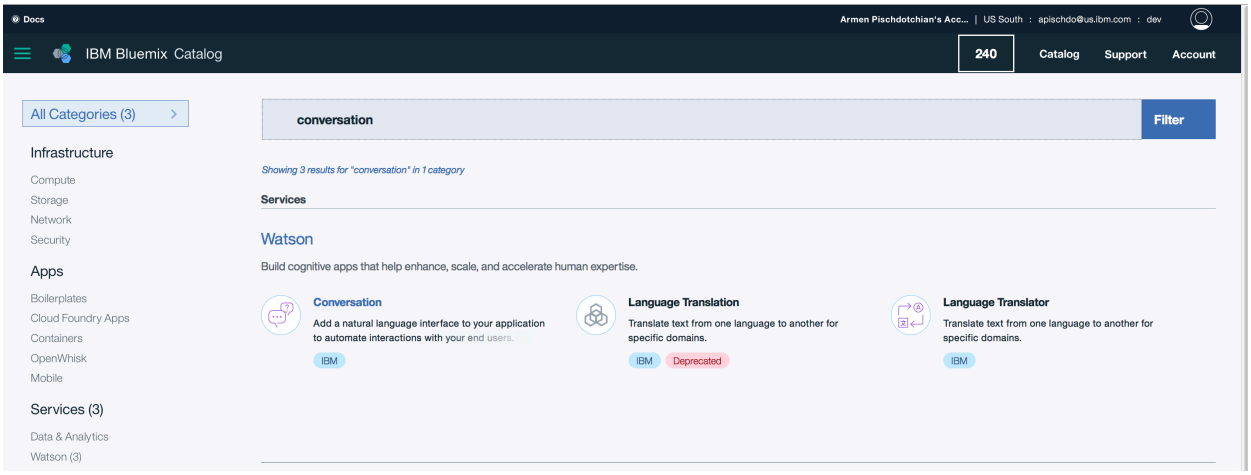
1. Create a top-level directory on your system.
2. Download the code from the Github Repository: https://github.com/apischdo/Bluemix-workshop-assets/blob/master/Conversation.zip
3. Extract the contents of the **Conversation.zip**. If it auto extracted in your Downloads folder, then copy the extracted contents to your top-level directory that you created in Step 1.
4. Open a command window or a terminal and change directory (`cd`) to the working directory (in this example, I named my top directory 0001_mychatbot (you can pick your own directory name).



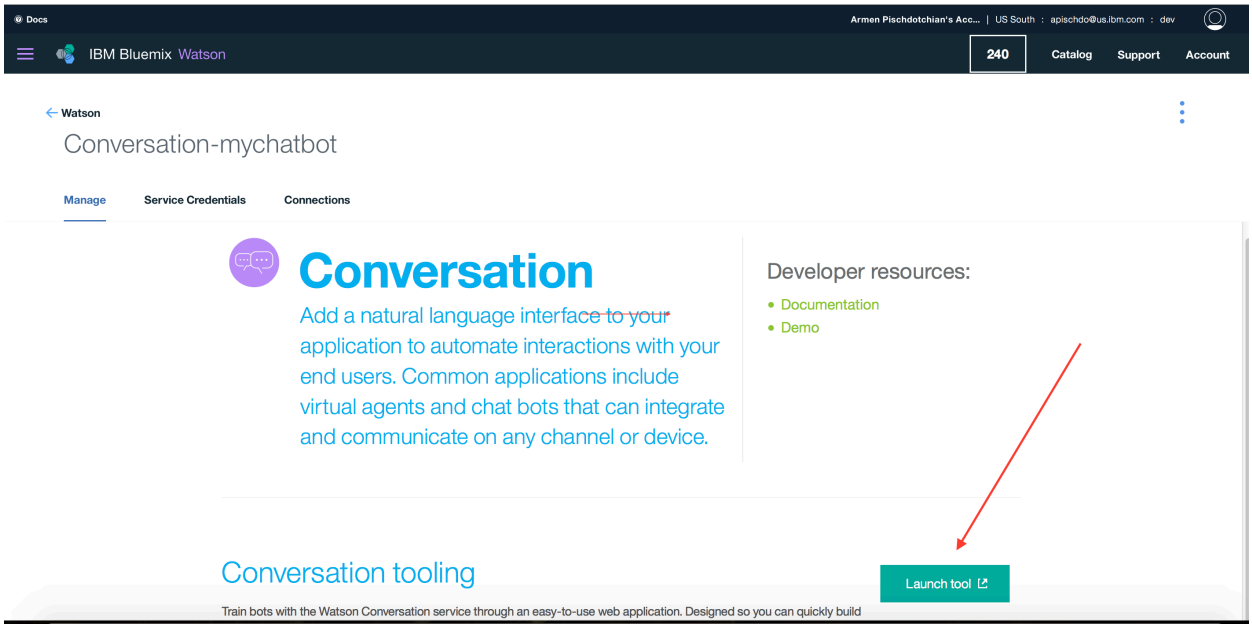
5. Press Enter and issue an `ls` (for Mac) or `dir` (for PC) command to ensure that all the artifacts that you extracted are visible. If you see the app.js, the main engine that does the interpreting, then you are working from the correct folder.
6. From the same terminal, run the node package manager (npm) to install dependencies that node.js relies on for further processing of the code. You might get some warning and one compile error message, ignore those; it has to do with version incompatibility. Notice that you have a new folder named node_modules.

   ```
   npm install
   ```

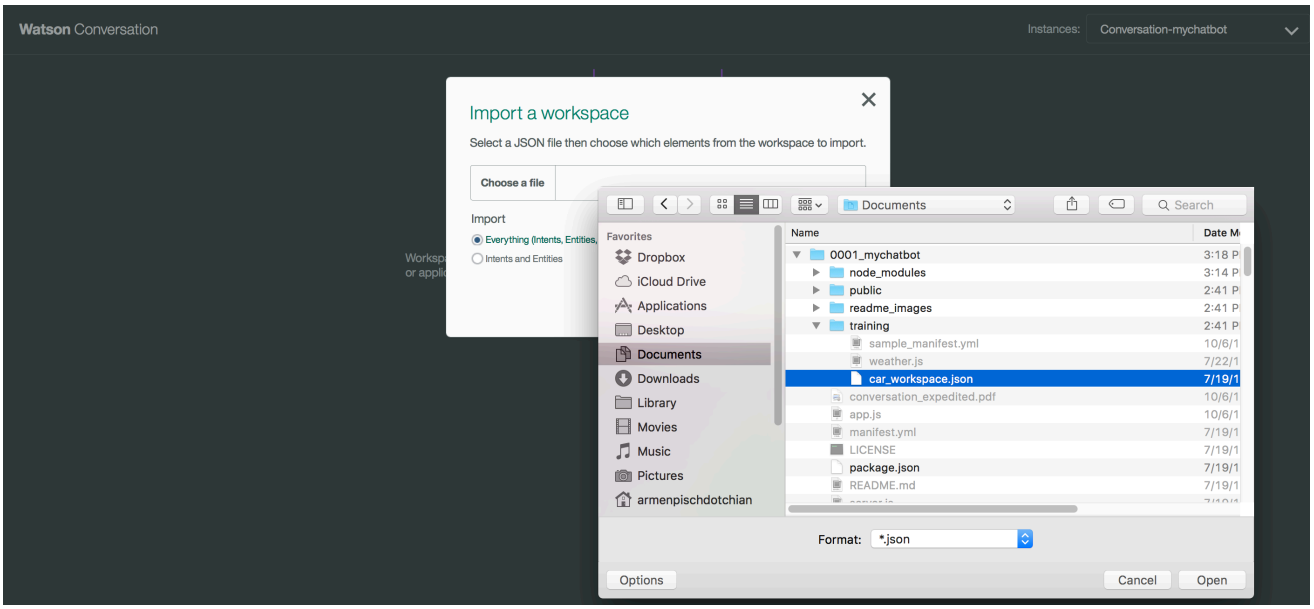
7. Login into IBM Cloud: https://console.bluemix.net/catalog/
8. Click the **Catalog** tab.
9. Search for the **Conversation** service and click that tile.



10. Edit the Service name to something meaningful to you (for example: Conversation-mychatbot) and click **Create** (If you have just created your account and accessed it from the confirmation email, you may need to log into IBM Cloud once again, then you can see the Create button in the bottom right corner).



11. Click the **Launch Tool** button. It may appear as an arrow in your interface 
12. Login once again using the same credentials that you use to login IBM Cloud.
13. Click **Import**.
14. Navigate to the **training/car_workspace.json** file and click **Open** to import the file.



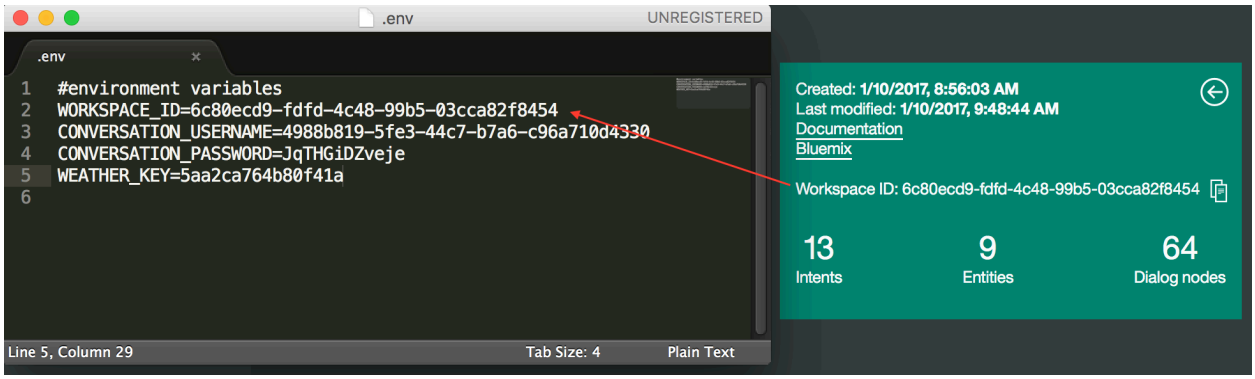
You must now provide four parameters to the code: conversation ID and the service credentials (username and password). Bear in mind that files that start with a dot "." are hidden system files, but with Sublime or Notepad++, by opening the folder, instead of just a file, you can see those hidden files. And if you just can't see internal files, then no worries just save the file and trust it is there.

15. In this example, select the top-level directory (0001_mychatbot) and open it with Sublime.
16. Open a new file with sublime and copy paste the below in that file; you may have to enter the line breaks after each equal sign so it appears as below in four separate lines:

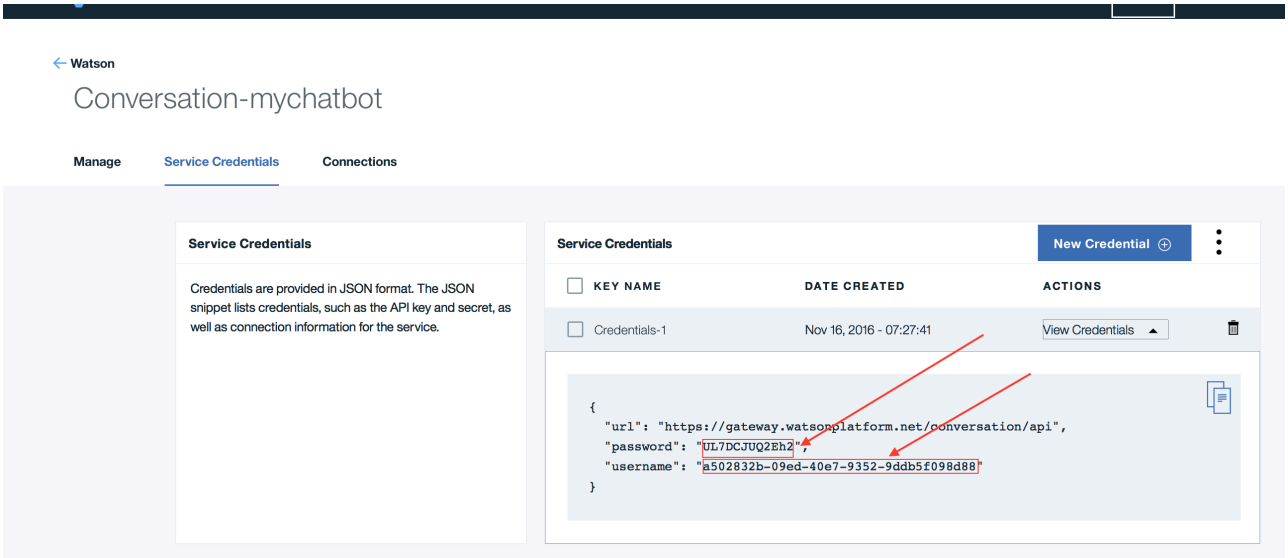
```
#environment variables
WORKSPACE_ID=
CONVERSATION_USERNAME=
CONVERSATION_PASSWORD=
WEATHER_KEY=5aa2ca764b80f41a
```

You are now ready to populate the environment variables:

17. Obtain the workspace ID by clicking the three dots in Workspace box and then click **View Details**.

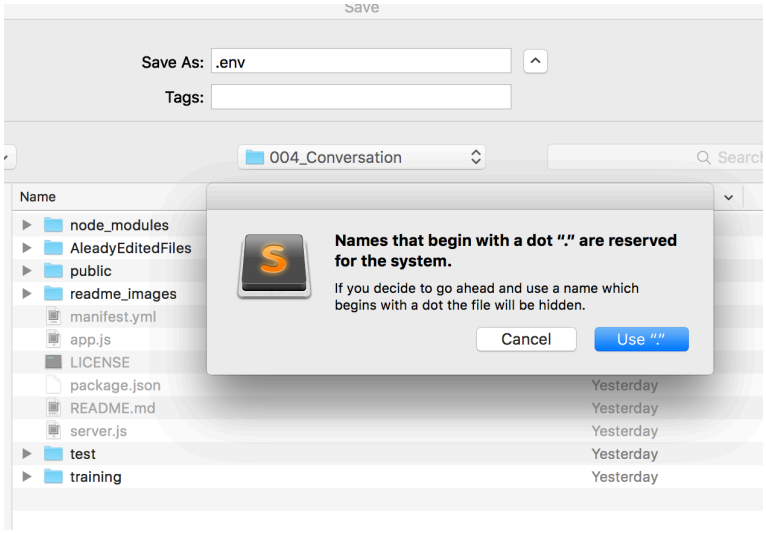


18. Go to the Conversation service on IBM Cloud, and obtain the username and password of the Conversation service from the **Service Credential** link and then click **View Credentials**.



19. The Weather API key is: `5aa2ca764b80f41a`
20. Enter all four parameters in a new file in Sublime or Notepad ++.

21. Save the file as **.env** (notice, there is a dot in front of the file name, this is a system internal file, typically hidden) and click **Use "."** Ensure that you save this file in the top-working directory where all other files, same level as app.js reside.



If you are using a PC, for file type ensure to select **All types**. You don't want to save it as a txt file.

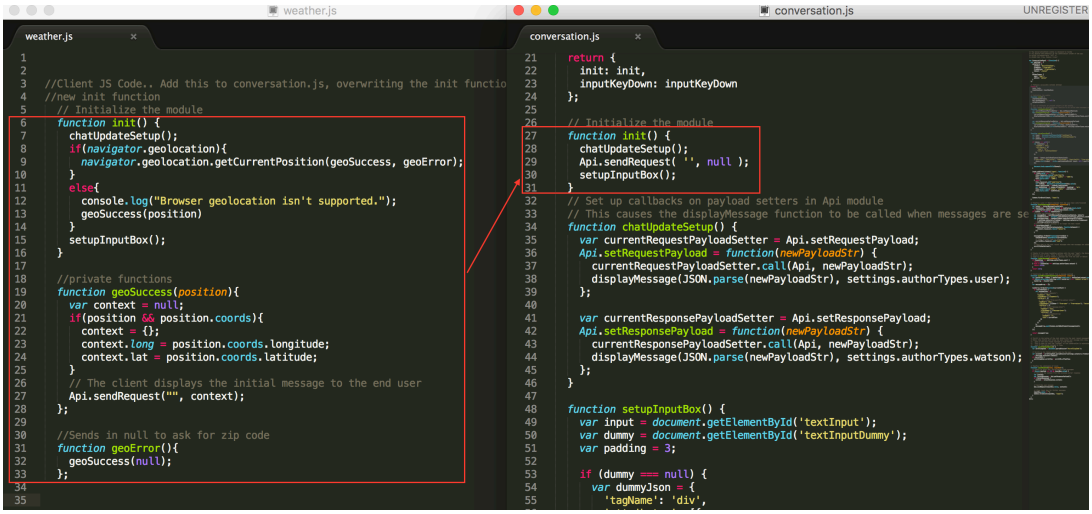
22. Go back to the terminal window and run the following command:

```
Node server.js
```

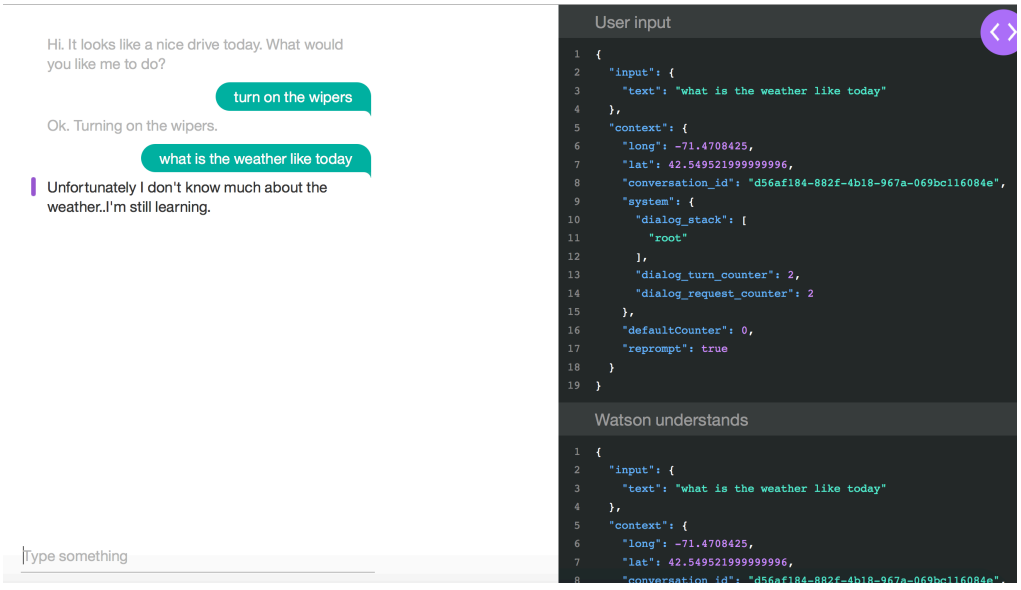
Binding the Conversation service with an external API

For this next section you are going to use the geo-location capabilities of your browser to check the actual weather forecast and for that you will need to include the API key from the Weather Company web site that you obtained from the Pre-requisites section.

1. From your top-level directory navigate to the **training** folder and open the **weather.js** file and position it to the left of your screen.
2. Navigate to the **public/js/conversation.js** folder and open the **conversation.js** file and position it to the right of your screen; just like the image below, you want them side-by-side.



3. Copy the **init( )** function, **geoSuccess( )** and **getError( )** functions (lines 6 through 33) from the **weather.js** file *replacing* the init( ) function in the **conversation.js** file, *over writing* lines 27 through 31.
4. Save the **conversation.js** file and close it.
5. Run the `node server.js` command again and if need be, (Ctrl+C) to end the current terminal session.
6. Back to your localhost:3000 tab and refresh the browser.
7. Ask the following questions: **Turn on my wipers** and **What is the weather like today?**



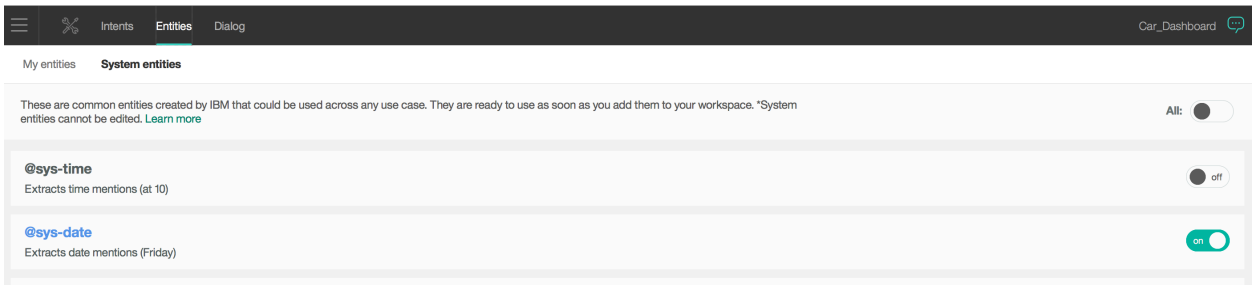
Notice that it understands the concept of weather, but the app cannot fetch that data at this point.

You are now ready to incorporate these changes in the dialog structure.

Updating Dialog

Working with Intents, Entities and Dialog is covered in a separate workshop that elaborates on all facets of creating a robust dialog, for the purposes of this lab, you will edit the Entities and the Dialog.

1. Go to the **Car_Dashboard** conversation and click the **Entity** tab.
2. Turn on the **@sys-date** system entity.



3. Click the **Dialog** tab.
    a. Select and click the **Weather** Dialog to open it for editing.
    b. Delete all three child nodes to the right of it.
    c. Make sure the condition is "**#weather"** and the response field is blank.
    d. Add two child nodes of **#weather**, named "Weather date specified" and "*Weather date not specified*".
    e. For "*Weather date specified*" include a condition of "`@sys-date`" and a response of

```
The weather for <?@sys-date.reformatDateTime('EEEEEEEE')?> in {0} is {1}
```

        i. Ensure that the single quotes around `EEEEEEEE` are straight, san serif, when you enter it in the node, not curved as it appears in this rich text format below. Type over it in the code.

    f. For "*Weather date not specified*" include a condition of "`!@sys-date`" and a response of

```
For which day would you like to know the weather?
```

    g. Return to the **#weather** node and change the *And finally* to be a Jump to *"Weather date specified"*.

~~h. In the Triggered by section, add an **or** function (change the and to or) and specify it as: **@sys-date**~~

~~i. Add two new responses: the true statement: **@sys-date**, which contains the following date format~~

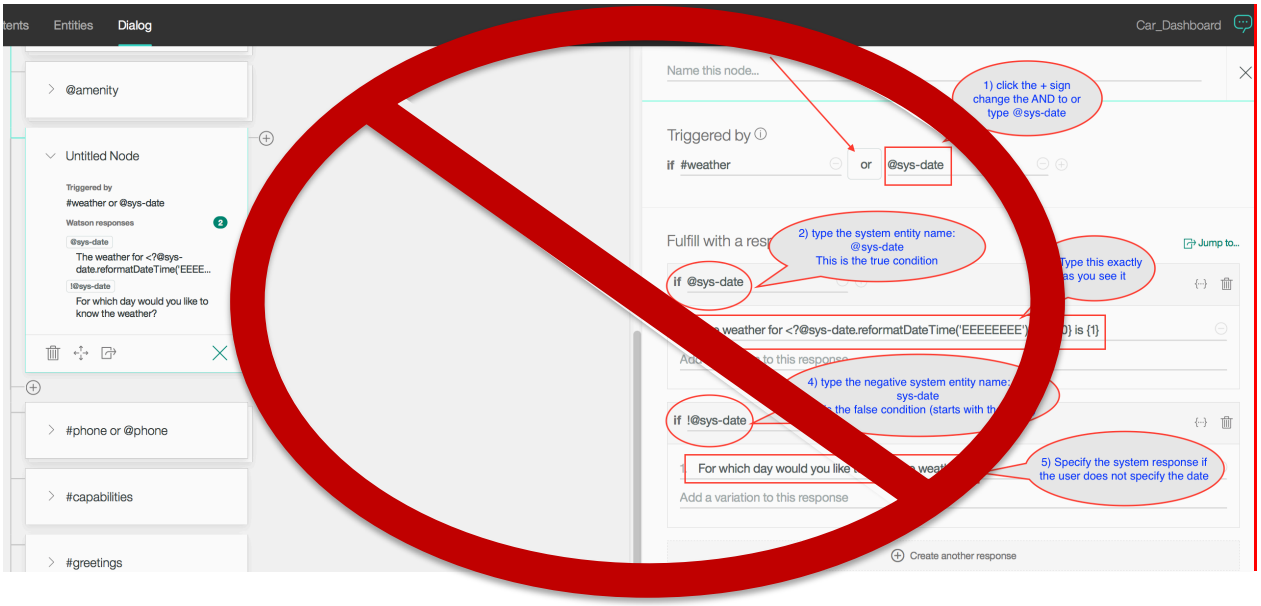
~~j. Ensure that the single quotes around `EEEEEEEE` are straight, san serif, when you enter it in the node, not curved as it appears in this rich text format below. Type over it in the code.~~

```
~~The weather for <?@sys-date.reformatDateTime('EEEEEEEE')?> in {0} is {1}~~
```

~~k. Add the false statement: **!sys-date**....notice the ! sign up front...that means NOT.~~

l. ~~Include a system response for users who do not indicate for when they want to know the weather:~~

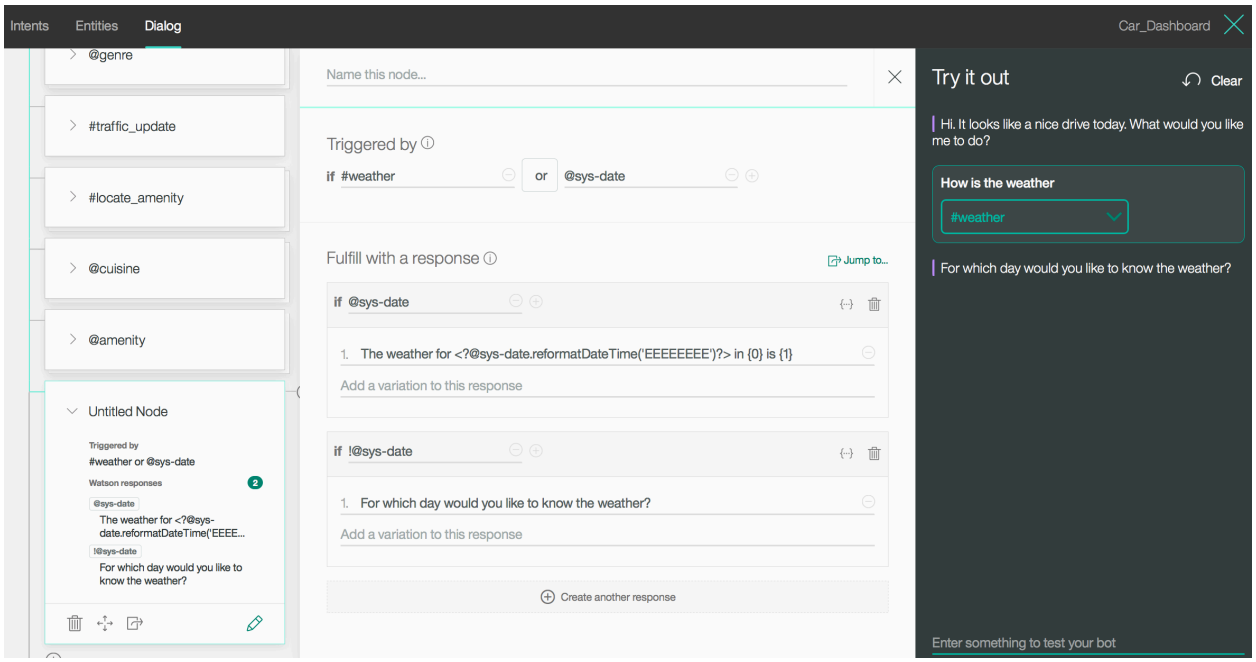
~~For which day would you like to know the weather?~~



Now let's test the conversation

4. Click **Try it out** in the upper right corner.

5. Ask: **How is the weather?**



6. The system asks you to specify a date. Type a day, for example **Friday**.

   Notice it understood the intent and returned: The weather for Friday in {0} is {1}

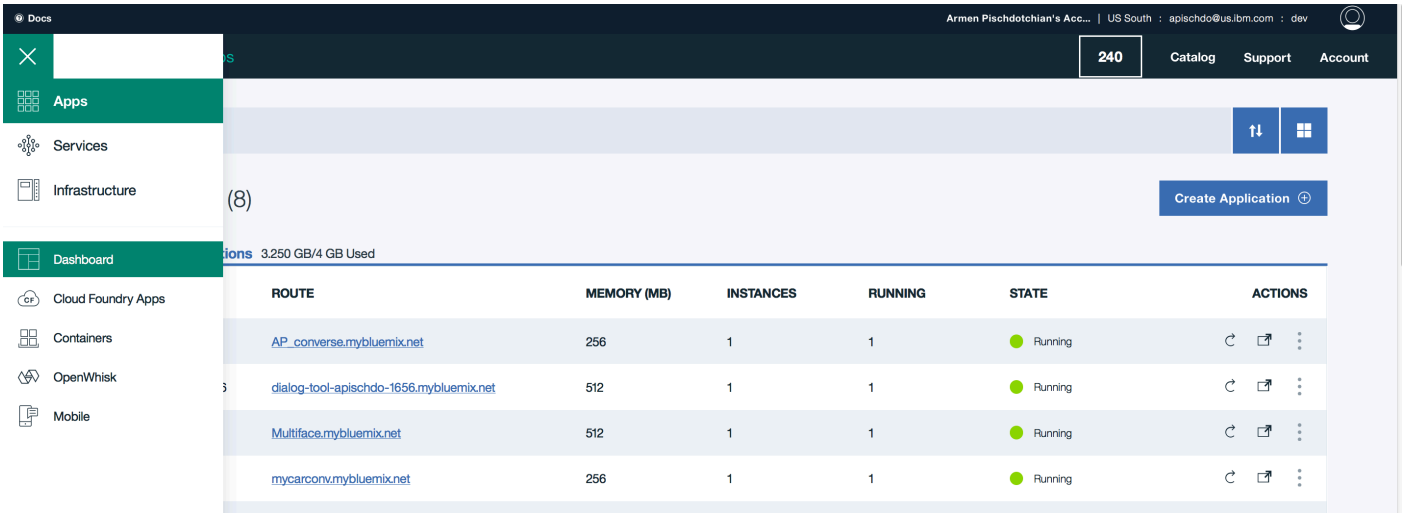
7. Try it in **http://localhost:3000**
8. Ask the question: “How is the weather today?” or just “Friday”. Notice the accurate weather forecast at your location.

If time permits, the following exercises are optional.

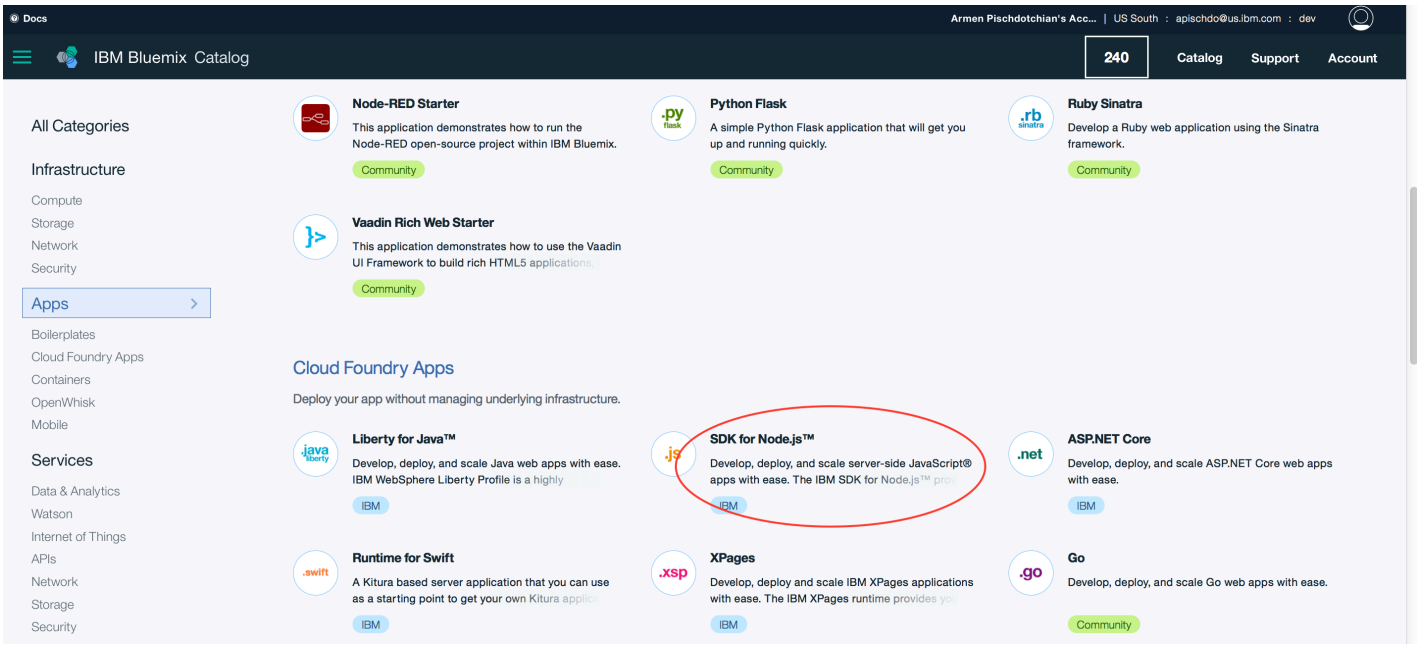
Deploying your app to IBM Cloud

An easy way to deploy your app is by including an app name in the manifest.yml file and then using the cf push command to deploy the app onto IBM Cloud. However, this guide will make greater use of IBM Cloud capabilities in ensuring that you app is deployed properly.

1. You will need to install the CLI command line from here: https://github.com/cloudfoundry/cli/releases
2. Back to IBM Cloud, and click the **Dashboard** view from the left contents drop down link.



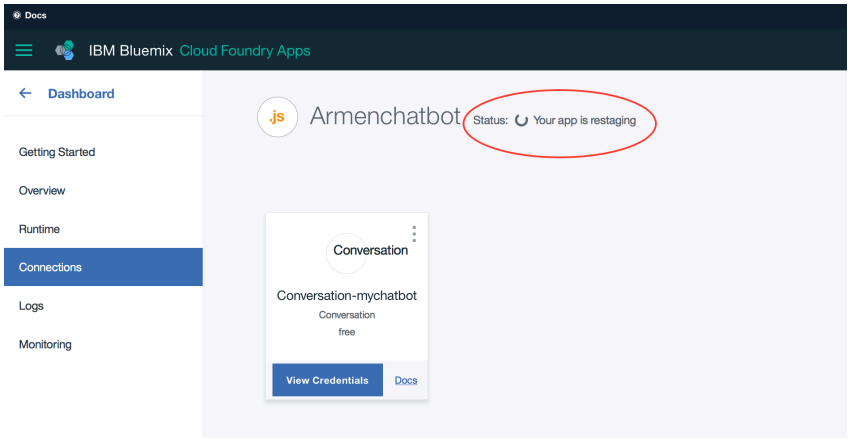
3. From the Dashboard view and click **Create Application** (you may have to scroll down a bit).
4. Scroll down and from the Cloud Foundry Apps, select the **SDK for Node.js**



5. Specify a unique name, you may have to include your name, for example, I used Armenchatbot.
6. Click **Connections** link from the left panel.

7. Click **Existing** and find the very Conversation service that you created earlier in this lab (for example: mychatbot)

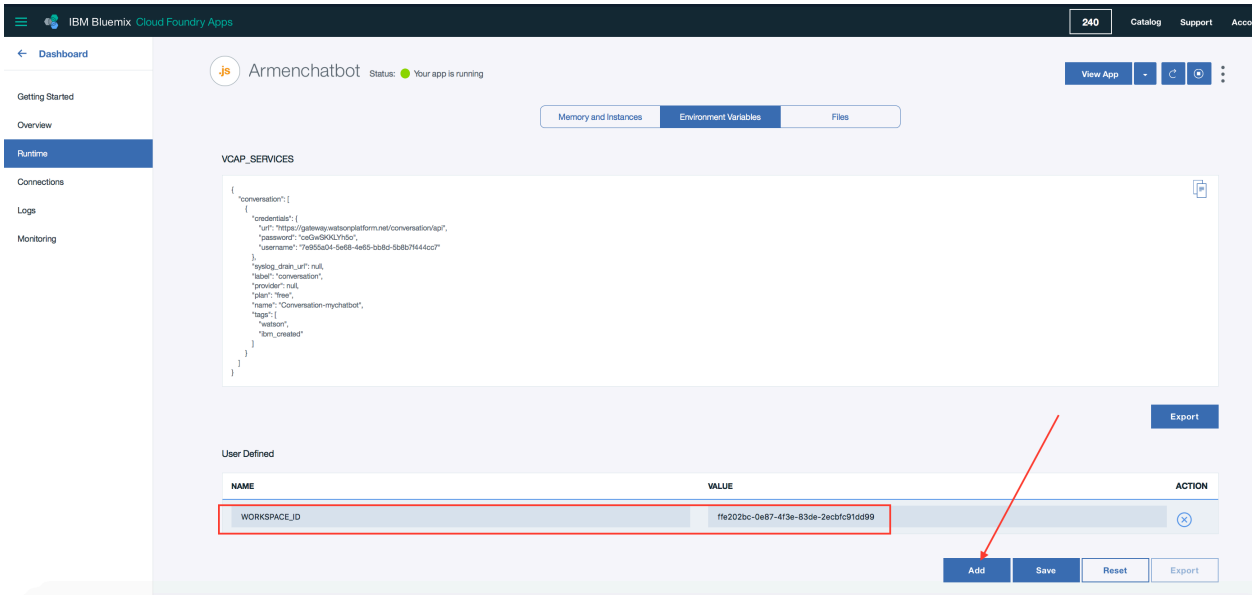
8. Click **Connect** and then click **Restage**. Allow enough time for the restaging to complete.



9. Click the **Runtime** link in the left panel.

10. Click **Environment Variables** (it takes a minute or two for the VCAP service to appear; no worries if you don't see them right away; you are ready to include the user-defined variables, which is your WORKSPACE_ID.

```
  WORKSPACE_ID          01f8ecc8-1701-4ed1-a1f9-0e76bd6b0d69
(use your ID number from the Car_Dashboard workspace……the details view).
```



11. Repeat the previous step and add the weather variables:

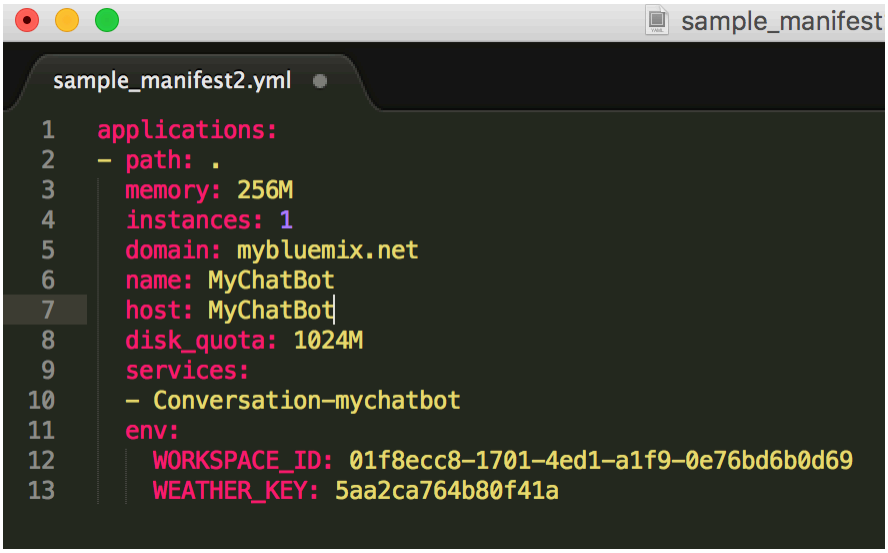
```
WEATHER_KEY           5aa2ca764b80f41a
```

12. Click **Save**.

13. Click the **Getting Started** link from the left panel.

The reason you created a dummy file is two fold: The Starter Code, which you will download in the next step contains the manifest file that you will replace with the manifest.yml file in your working directory. It has the proper URL and all the credentials that you need; the second reason is so you can see the commands needed to upload your app (with the new manifest.yml file) to IBM Cloud. You can copy/paste the commands that appear in the Getting Started page, except replace bluemix in the command line, with cf. We did not use the bluemix plugin, we used the cf (Cloud Foundry) plugin.

14. Click **DOWNLOAD STARTER CODE**.
15. Extract the contents of the downloaded code in a temporary location.
16. Copy the **manifest.yml** file that has an app name and the user defined variable value, replacing the **manifest.yml** file in your working directory. Alternatively, you could have just changed the Manifest file with app name and the custom env variable, but it's good to see where and how it's generated.



17. Start from Step 4 onward from the **Getting Started** page on IBM Cloud and you perform these in the same terminal or Command prompt where the app.js and the server.js files reside (start with `cf` not bluemix). The command below is one line. You org and space name may differ.

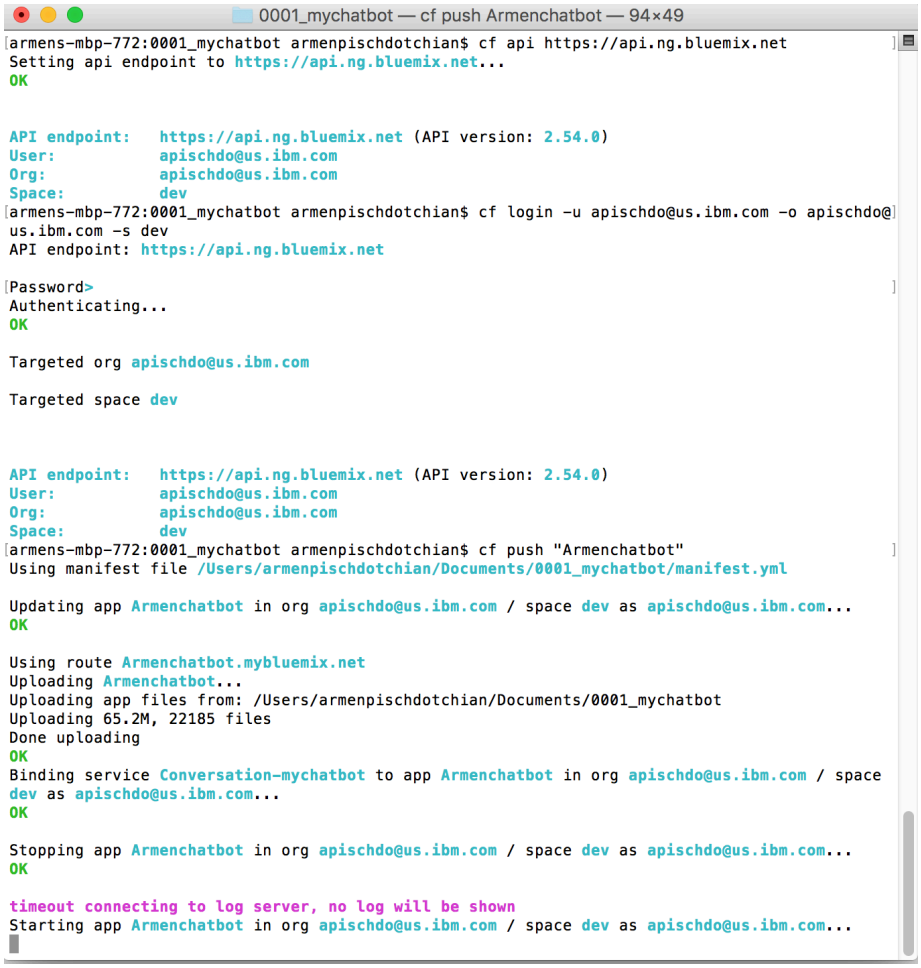
```
% cf login -u apischdo@us.ibm.com -o apischdo@us.ibm.com -s dev -sso API endpoint: https://api.ng.bluemix.net
```

18. Copy and paste the resulting URL in your browser to obtain the One Time Code https://login.ng.bluemix.net/UAALoginServerWAR/passcode)>
19. Enter the code from the URL back into the command prompt or terminal as the password.
20. Push the app to IBM Cloud:

```
%cf push <your_app_name_as_it_appears_in_the_manifest_file>
```

The above is just an example of my credentials and settings. Yours would have your email address, org name and space name.

21. Allow enough time for the app to upload and restage. This takes a few minutes. Watch the console.



22. From within your application click **View App** (you may need to Restart the app again from IBM Cloud).
23. Accept geo-location sharing request.
24. Ask away some typical car questions and then hit it with the weather questions.

Congratulations, you just deployed a chatbot onto IBM Cloud.