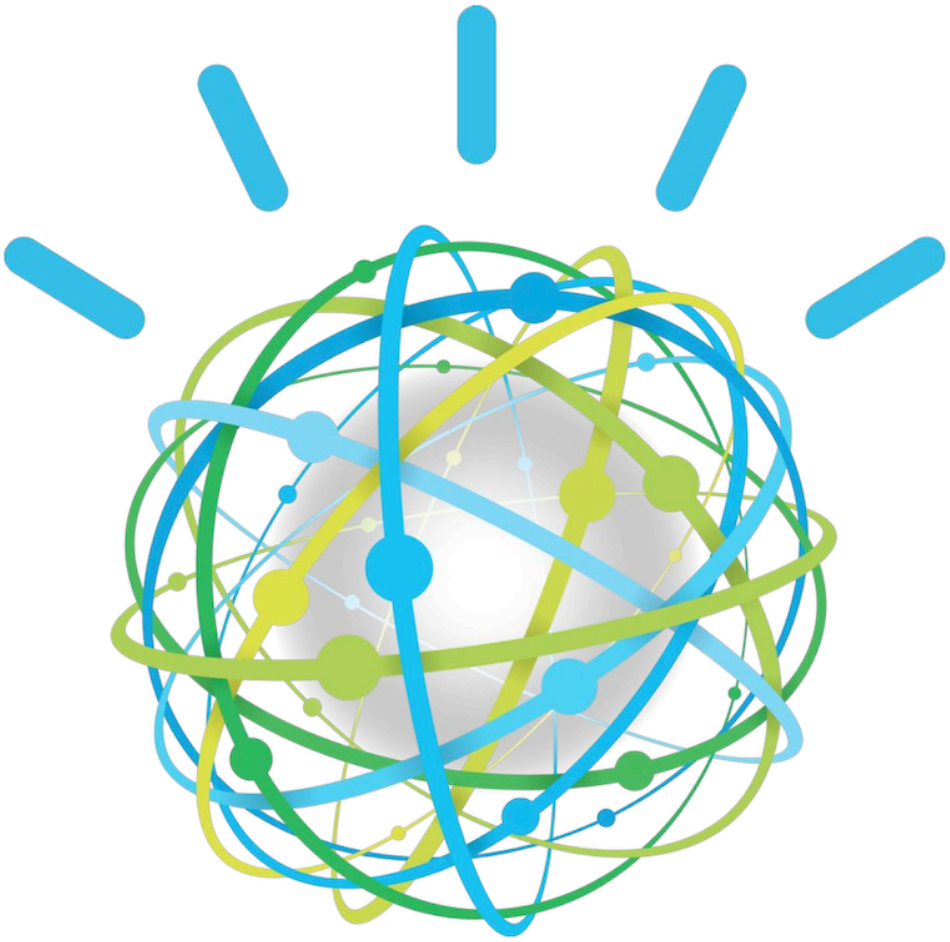
IBM Watson Solutions

Business and Academic Partners



Developing a Chatbot Using the IBM Watson Conversation Service

Prepared by Armen Pischdotchian

Version 2.1 October 2016

Overview

What is Bluemix you ask? **Bluemix** is an implementation of IBM's Open Cloud Architecture, leveraging Cloud Foundry to enable developers to rapidly build, deploy, and manage their cloud applications, while tapping a growing ecosystem of available services and runtime frameworks. You can view a short introductory video here: **http://www.ibm.com/developerworks/cloud/library/cl-bluemix-dbarnes-ny/index.html**

Additionally, for our academic partners, there are no-charge 12-month licenses for faculty and no-charge 6-month licenses for students - all renewable and NO CREDIT CARD required!

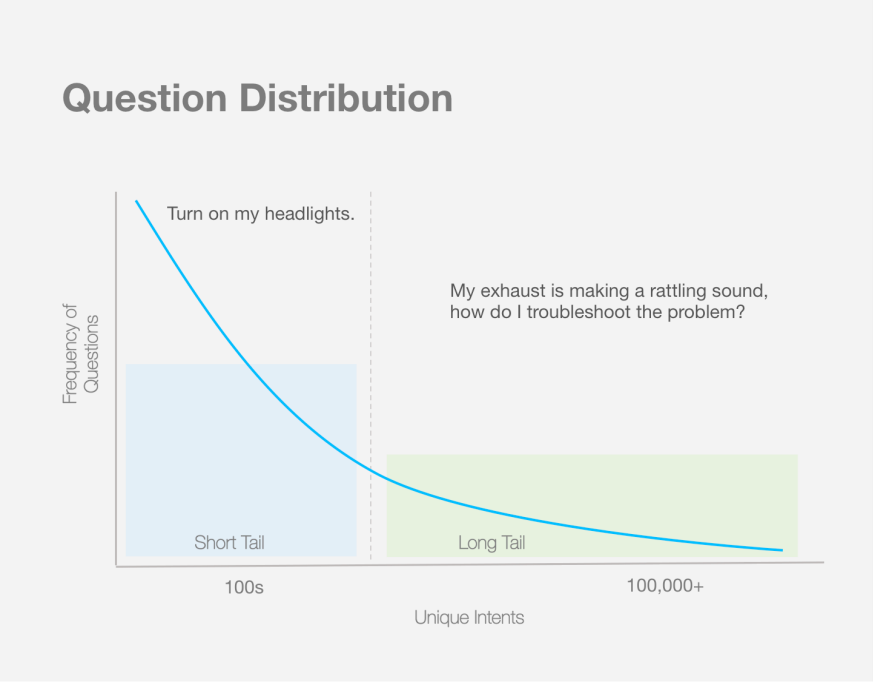
To get started, you will need to become an Academic Initiative member. Refer to this document for details:

http://it.husc.edu.vn/Media/TaiLieu/IBM_Academic_Initiative_for_Cloud_Process.pdf

The purpose of this guide is not to introduce you to Bluemix, that foundational knowledge is a prerequisite study on your part and you can obtain it from the links mentioned above. This guide is more of 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 per below.



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. Complete the following steps:

- **Obtain Bluemix credentials:**
  1. Direct your browser to the Bluemix home page: https://console.ng.bluemix.net/home/
  2. Click **Sign Up** on the top right.
  3. Enter requested information and click **Create Account**.
  4. If you have received a promo code as a member of the Academic Initiative, enter it at this time; otherwise, you have 30 days to include the promo code.
- **Install the Node.js runtime:**

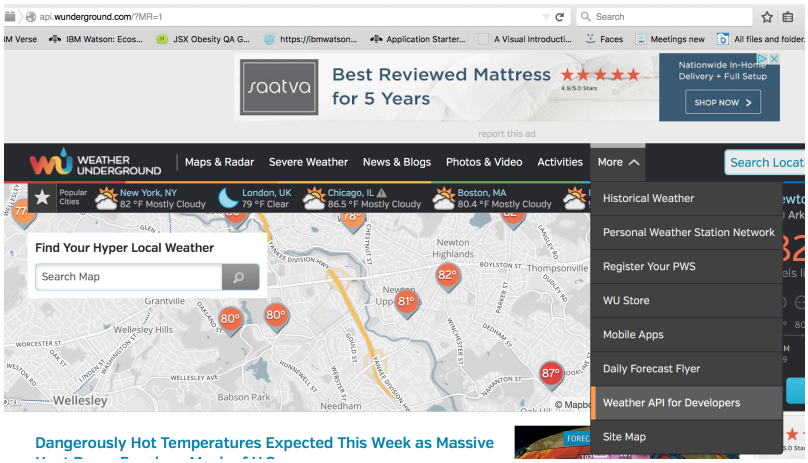
  The default installation includes both the runtime and package manager. Make sure to include the installed binaries on your PATH environment variable after installation (typically, the default installation locations that the installer selects does the inclusion).
  1. Direct your browser to the nodejs.org web site: https://nodejs.org
  2. Click **Downloads.**
  3. Select and install the installer (not the binary) appropriate for your operating system.
- **Install the cf command line**
  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**.
- **Download OS appropriate code-friendly editing tool:**
  - If you are using a PC, we recommend using **Notepad ++**
  - If you are using Mac, we recommend **Sublime Text**

- **Obtain the Weather Company API**

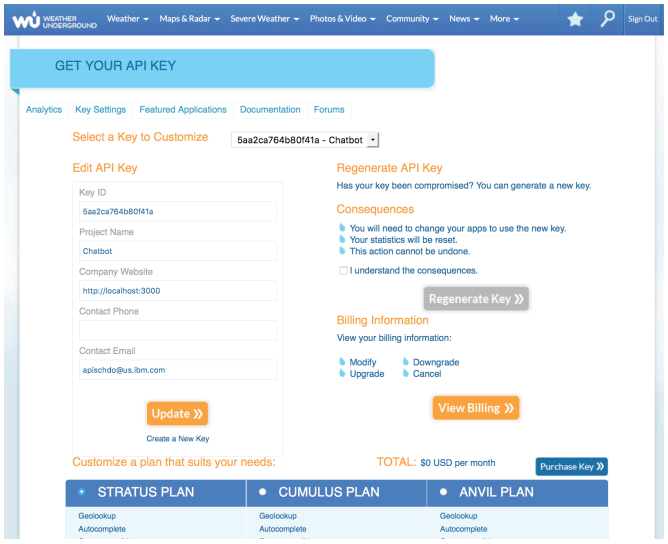
  Currently, the following desktop browsers support the W3C Geolocation API:

  - Firefox 3.5+
  - Chrome 5.0+
  - Safari 5.0+
  - Opera 10.60+
  - Internet Explorer 9.0+

1. Point your browser to the following site: http://api.wunderground.com/
2. Create an account and sign in.
3. From the **More** pull down menu, click **Weather API for Developers**.



4. Follow the on screen instructions to obtain the key. All fields must be filled.
5. Select any plan (default is STRATUS PLAN) and click **Purchase Key** (no cost).
6. Below is an example from my experience (use http://localhost:3000 for the purposes of this lab). You need your own key, which appears under the **Key Settings** tab. Refer to your email for verification.



- **Extract the ConversationMaster_expedited.zip package from Github**

  This package contains basic code that you will need to build your app.

  1. Direct your browser to a GitHub repository (no need to sign up): https://github.com/
  2. Search for **bluemix-workshop**.
  3. Scroll down and select: apischdo/*Bluemix-workshop*-assets
  4. Download just the **ConversationMaster_expedited.zip** package.
  5. Extract the contents of the **ConversationMaster_expedited.zip** file in a temporary location on your local system.

Overview of the steps required to complete the workshop

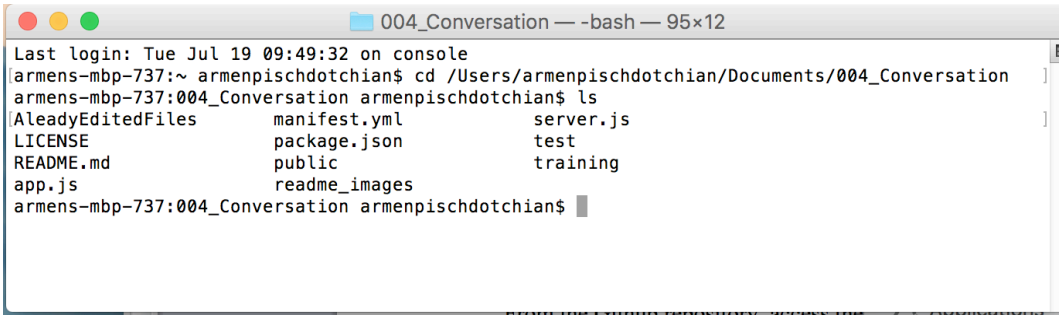
This table summarizes the steps that you need to complete to stand up an instance of the Watson Conversation service and integrate it an eternal service.

Step	Where do I go?	What do I do?
1	Github: https://github.com/apischdo/Bluemix-workshop-assets	Download the ConversationMaster.zip (you can also download everything that you see there). Alternatively, can also use the following command from a command prompt or your terminal: `git clone https://github.com/apischdo/Bluemix-workshop-assets.git`
2	Your machine, the terminal window	Run the command `npm install` from the same directory where you extracted or cloned the package
3	Bluemix	Create a Conversation service and launch the tooling
4	Conversation service tooling	Import the `car_workspace.js` file
6	Your machine: create a new system internal file	This `.env` file will contain the conversation ID and the username/password for the Conversation service
7	Your machine, the terminal window	From the same directory, run the `node server.js` command
8	Your machine: use the provided weather.js file	Update the conversation.js and the app.js files. You include the API from the Weather Company in this step.
9	Your machine	From the same directory, run the `node server.js` command to reflect your changes.
10	Bluemix:	Specify custom credentials
11	Your machine	Run the `cf push` command
12	Bluemix	Invoke the Routes URL and test your app on Bluemix

Using the Conversation service

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

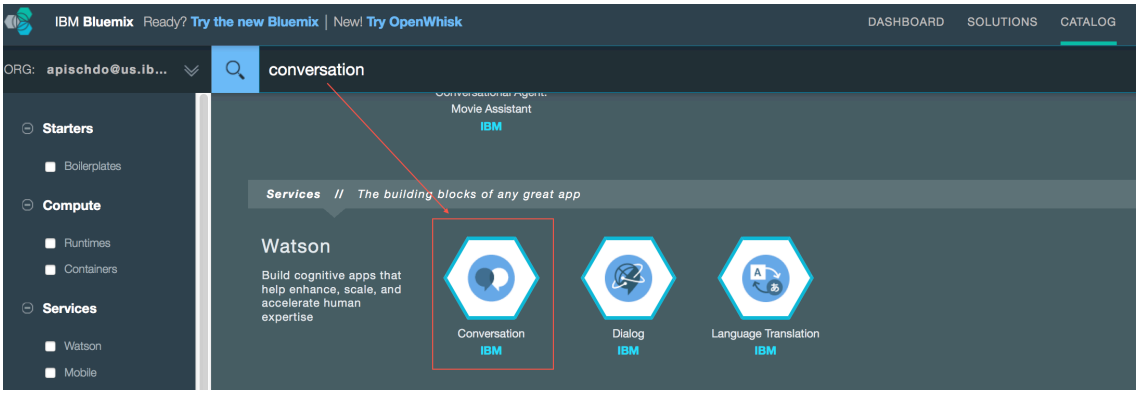
1. Create a top-level directory on your system.
2. Extract the contents of the **ConversationMaster.zip** file to your top level-working directory.
3. Open a command window or a terminal and change directory to the working directory (in this example, I named my top directory 004_Conversation (you pick your own directory name).



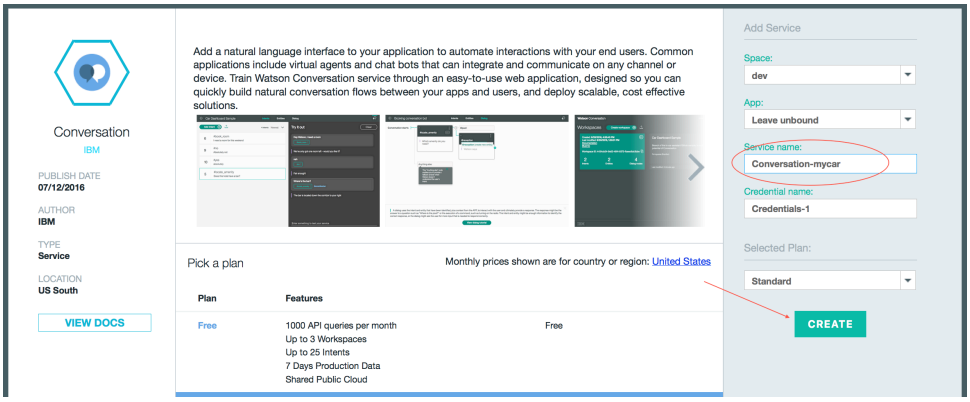
4. From the same terminal, run the node package manager (npm) to install dependencies that node.js relies on for further processing of the code. Notice that you have a new folder named node_modules.

   ```
   npm install
   ```

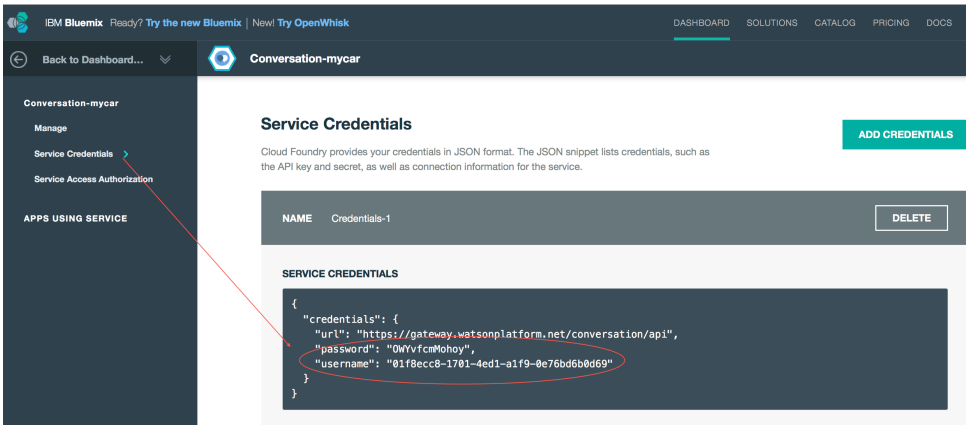
5. Login into Bluemix: https://console.ng.bluemix.net
6. Change to the **Classic** view. Notice your URL is slightly different that the one in the previous step. Edit the URL in your browser to reflect exactly this: **https://console.ng.bluemix.net**
7. Search for the **Conversation** service and click that tile.



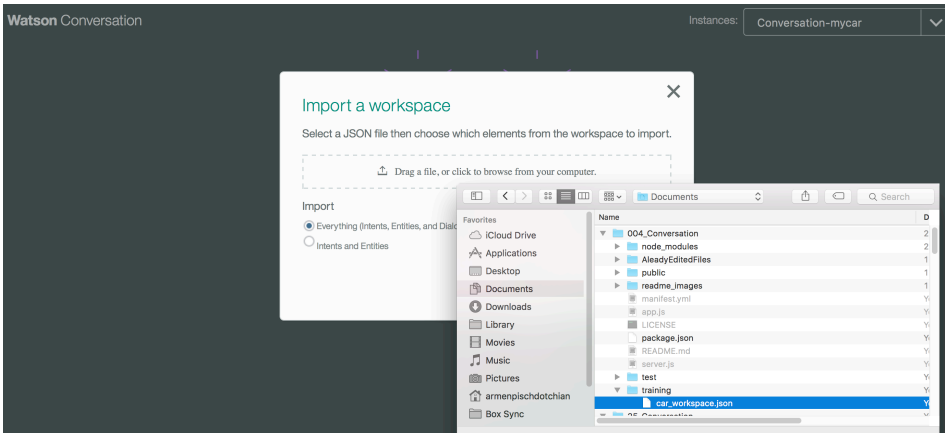
8. Edit the Service name to something meaningful to you (for example: Conversation-mycar) and click **CREATE**.



9. Creation of the service may take up to a minute or two. If seems to be spinning for a while, click the **Service Credentials** link in the left pane and copy and paste the username and password in your text pad.



10. Click **Manage** just above the Service Credentials and you will see the front page of the service. Note that creating the service may take up to 2 minutes or so.
11. Click the **Launch Tool** button.
12. Login (if you had not logged in earlier) using the same credentials that you use to login Bluemix.
13. Click **Import**.
14. Navigate to the **car_workspace.json** file in the **training** folder and click **Open** to import the file.



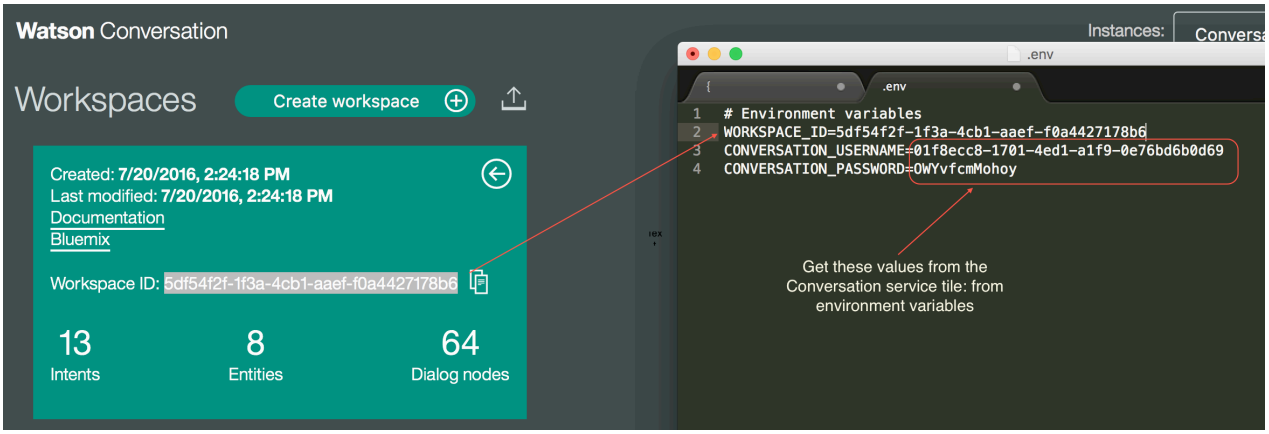
You must now provide three parameters to the code: conversation ID and the service credentials (username and password). Bear in mind that files that start with a dot "." are hidden system files, but with Sublime or Notepad++, by opening the folder, instead of just a file, you can see those hidden files. 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 (004_Conversation) 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:

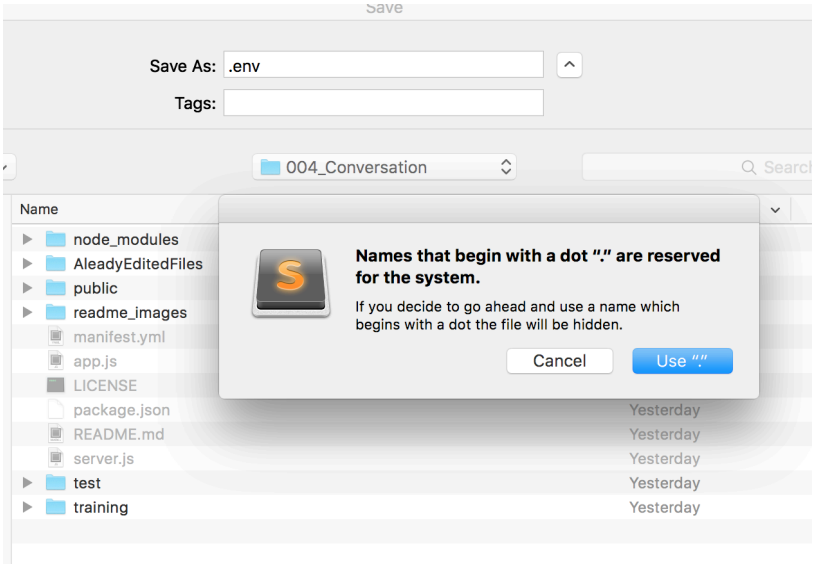
```
#environment variables
WORKSPACE_ID=
CONVERSATION_USERNAME=
CONVERSATION_PASSWORD=
```

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. Obtain the username and password of the Conversation service from the **Service Credential** link from **Bluemix.**



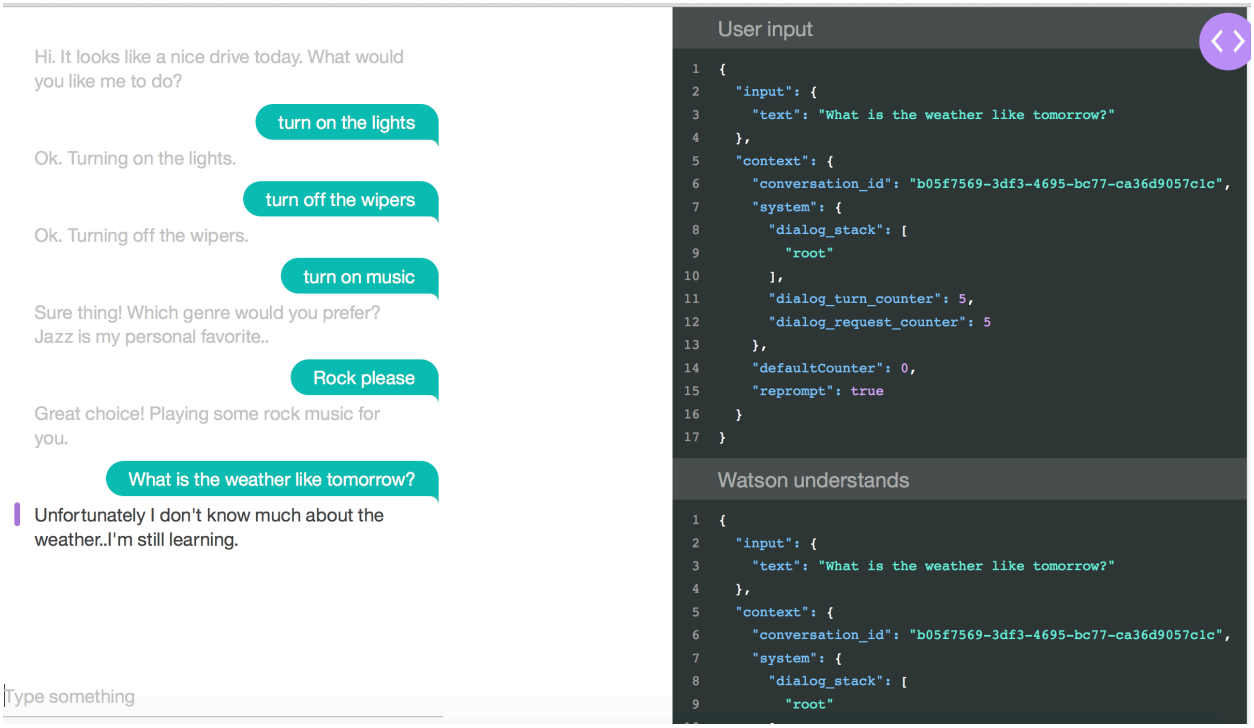
19. Save the file as **.env** (notice, there is a dot in front of the file name, this is a system internal file, typically hidden) and click **Use "."**



20. You are now ready to edit the app.js file. Open the **app.js** file.
21. Include the API key from the Weather Company in line 165:
    var key =//"enter API key from wunderground here";
    Ensure to remove the //comment forward slashes; you need the double quotes and the semicolon at the end.
22. Save the **app.js** file.
23. Go back to the terminal window and run the following command:

    ```
    Node server.js
    ```

24. Open a new browser tab and enter **localhost:3000** in the URL field and run a conversation similar to what you se in the screen capture below:

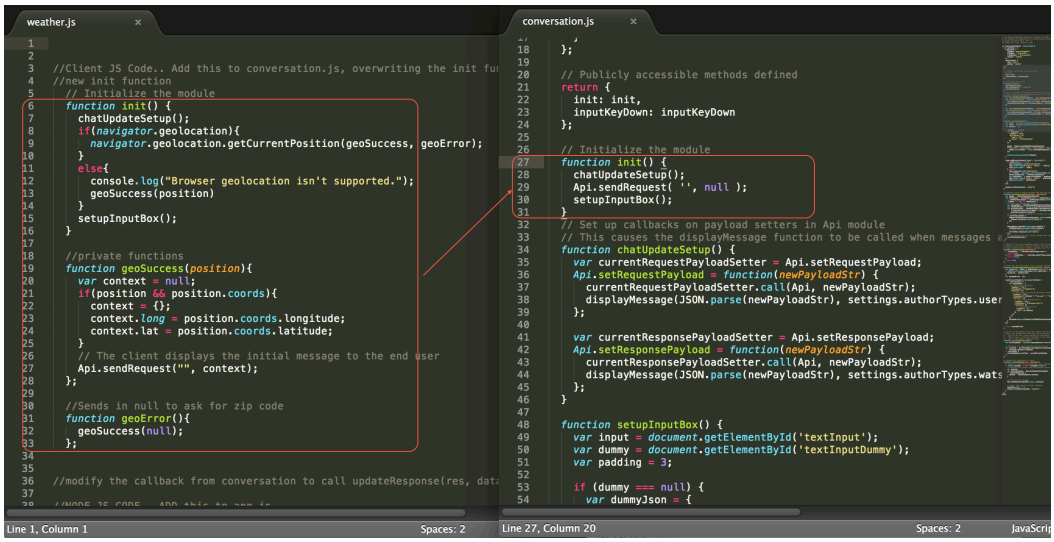


Notice that you have some work to do for the service to tell you how the weather is for your location, actual forecast exactly where you are sitting right now (well, rather where you will be when the next part of the workshop is complete).

Binding the Conversation service with an external API

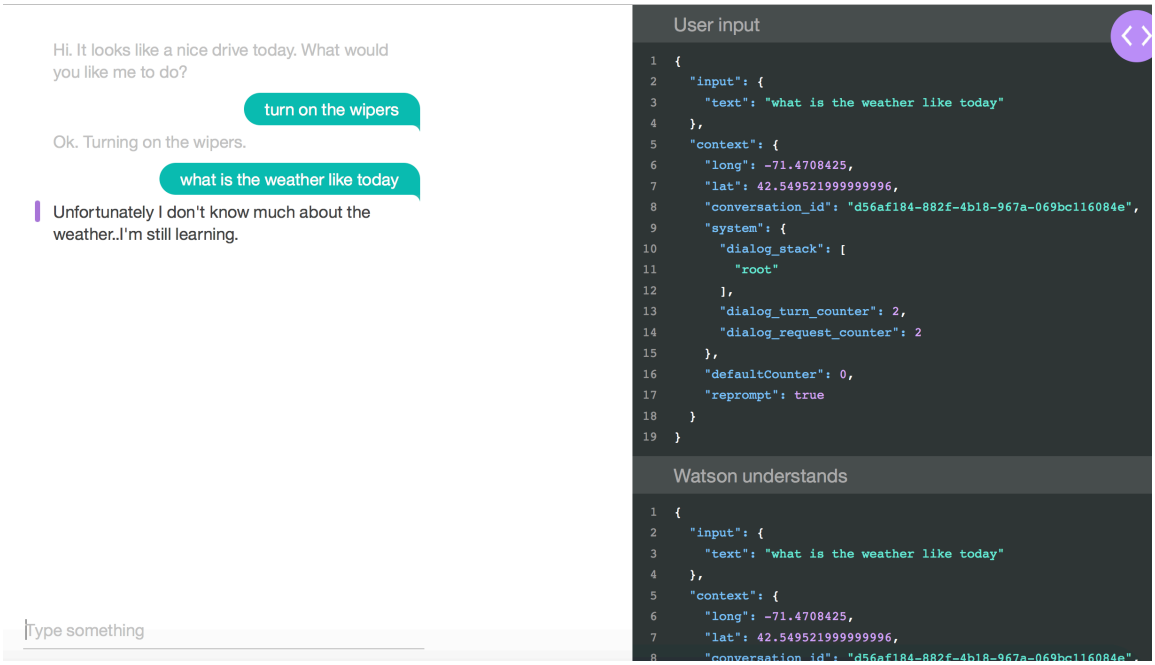
For this next section you are going to use the geo-location capabilities of your browser to check the actual weather forecast and for that you will need to include the API key from the Weather Company web site that you obtained from the Pre-requisites section.

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 (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. Ask the following questions:
    - Turn on my wipers
    - What is the weather like today?



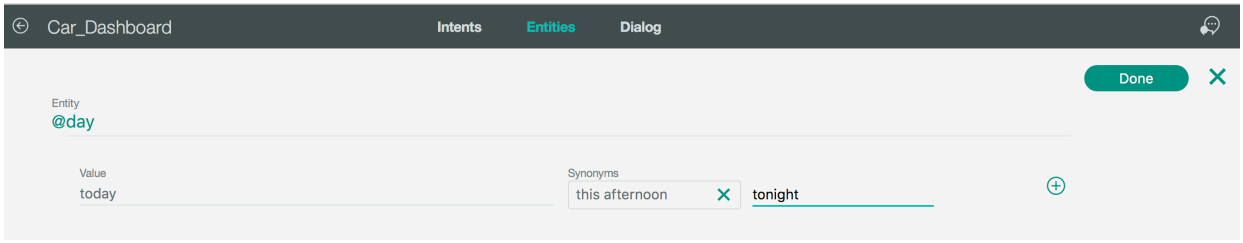
Notice that it understands the concept of weather, but the app cannot fetch that data at this point.

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

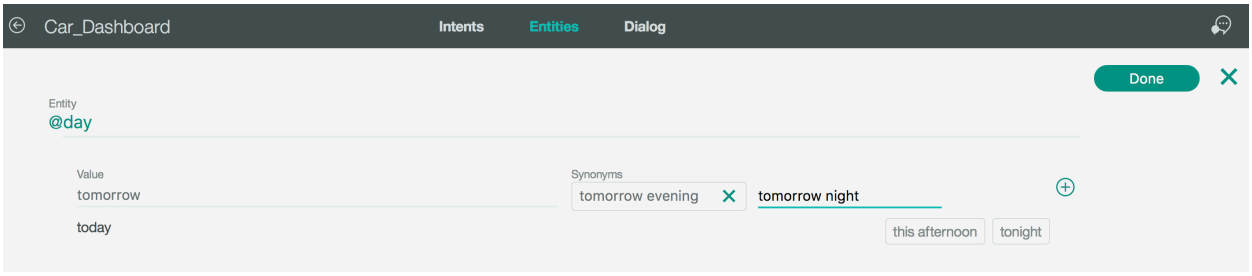
Updating Dialog

Working with Intents, Entities and Dialog is entirely separate discipline and subsequent workshops will elaborate on all facets of creating a robust dialog, for the purposes of this lab, you will edit the Entities and the Dialog.

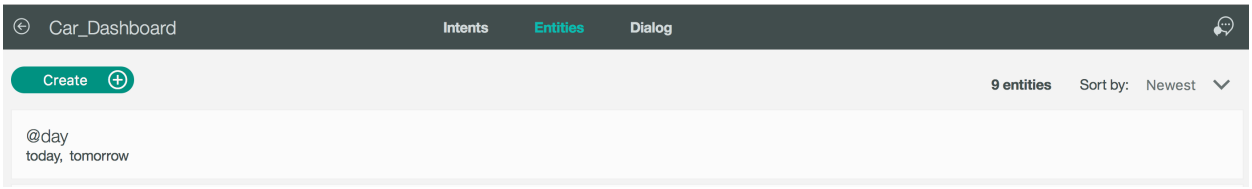
1. Go to the **Car_Dashboard** conversation and click the **Entity** tab.
2. Click the plus sign and add an *Entity* of **day** (case matters, use lower case).
3. Add the value of **today** for the Entity of day (case matters, use lower case).
4. Add synonyms such as **this afternoon** and **tonight** (case does not matter here, pick your own synonyms).



5. Click the plus sign and add another entity *value* of **tomorrow** (case matters, use lower case)
6. Add synonyms such as **tomorrow evening** and **tomorrow night**.

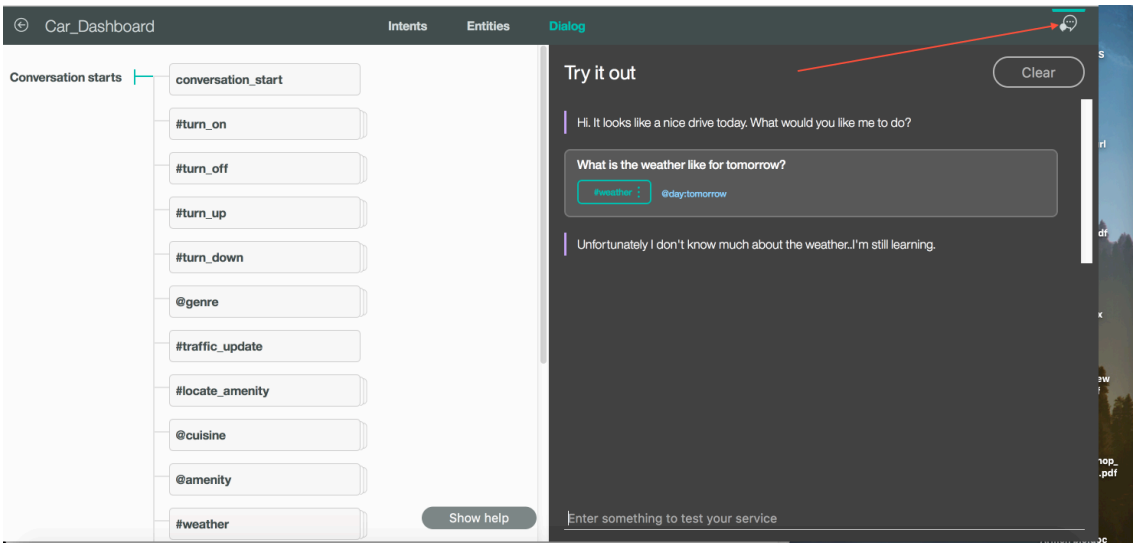


7. Click **Done**.
8. Note, # sign signifies Intent and @ sign signifies Entity. The end result looks like the image below:

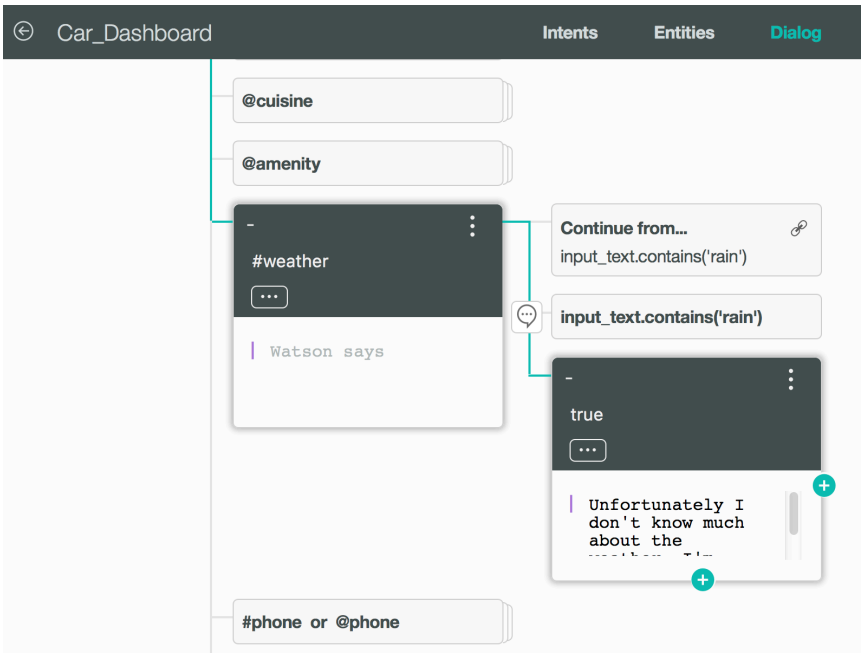


9. Let's update the Dialog flow. Click the **Dialog** tab.

10. Click the chat icon to the top right corner of the page and do a trial run and ask the system “how is the weather tomorrow?” The result should appear as below:



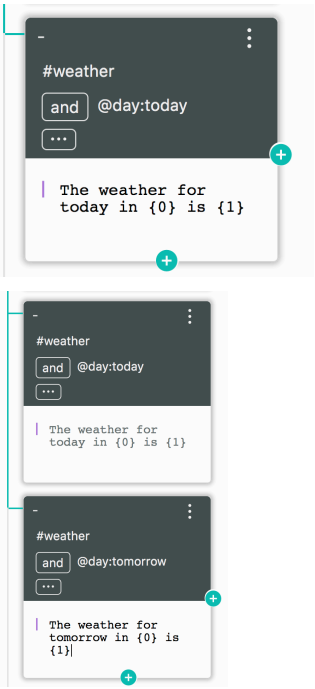
11. Close the chat window and click the #weather Dialog node and expand the tree, notice where the chat got it’s response!



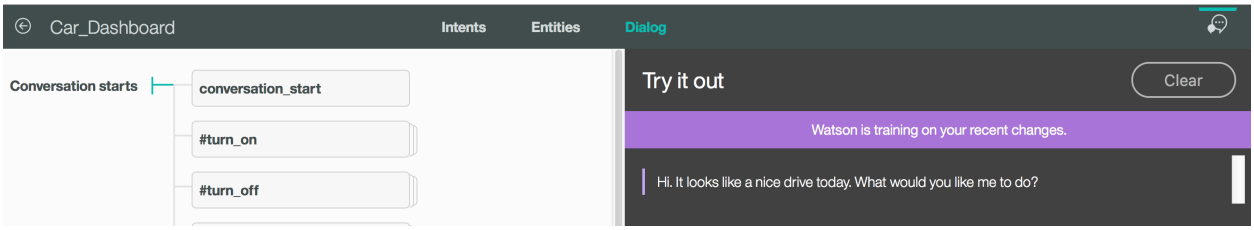
12. Click the three dots in the #weather node and delete the #weather node.
13. Click the amenity node (or any node) and then click the plus sign in the bottom of the node, to add another Dialog node.

14. Type and then select the **#weather** for Intent.
15. Click the plus sign *inside the green* node and specify the condition of **@day:today**
16. Add the dialog: `The weather for today in {0} is {1}`
17. There is the concept of conditions, if the condition is true then the dialog code gets executed.
18. Click the plus sign *underneath the dialog* node you just created and add another node: #weather
19. Add the condition of tomorrow: @day:tomorrow
20. Include the same script:

```
21. The weather for tomorrow in {0} is {1}
```



Keep an eye on the message where it says **Watson is training on your recent changes**. It will become green and state that training has finished and disappears shortly thereafter.



You are now ready to deploy your app to Bluemix.

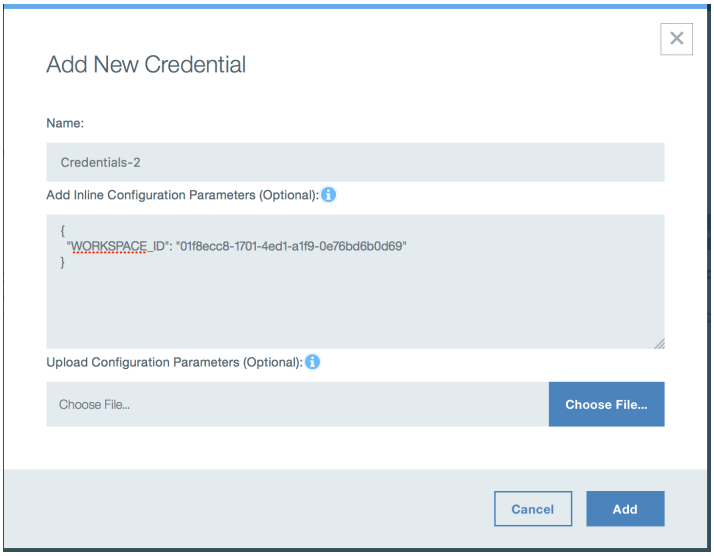
Deploying your app to Bluemix

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 Bluemix. However, this guide will make greater use of Bluemix capabilities in ensuring that you app is deployed properly.

1. Back to Bluemix, from the Dashboard view and click **Create Application** (top right corner).
2. Scroll down and from the Cloud Foundry Runtimes, select the **SDK for Node.js**
3. Follow the onscreen instructions until your placeholder app fully stages and is available, that means the Route link is live. This may take a minute or two.
4. Click the **Connection** link in the left panel.
5. Click **Existing Service**
6. Select the conversation service you created in the beginning of this workshop. Allow the restaging process to complete and click **Connect**.
7. Click **Restage**.

8. Click the **Service Credentials** link in the left pane.
9. Click **New Credetials**.
10. For name, accept default (Credentials-2) and for **Add Inline Configuration Parameters (Optional)**, copy and paste the WORKSPACE_ID from the Conversation tooling UI, using JSON syntax:

```
{
  "WORKSPACE_ID": "01f8ecc8-1701-4ed1-a1f9-0e76bd6b0d69"
}
```

11. 

12. Navigate back to your newly created application.
13. Click the **Getting Started** link from the left panel.
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. Try a short cut and from the terminal console or the command prompt (of course from your working directory) type: `cf push` and allow enough time for the app local to your machine to upload to Bluemix.
18. If the prompt is asking you to login, then follow the instructions from Step 6 onward and you perform these in the same terminal or Command prompt where the app.js and the server.js files reside.
19. Allow enough time for the app to upload and restage
20. From within your application click the **Overview** and then **View App** (you may need to Restart the app again from Bluemix).

Congratulations, you just deployed a chatbot onto Bluemix.