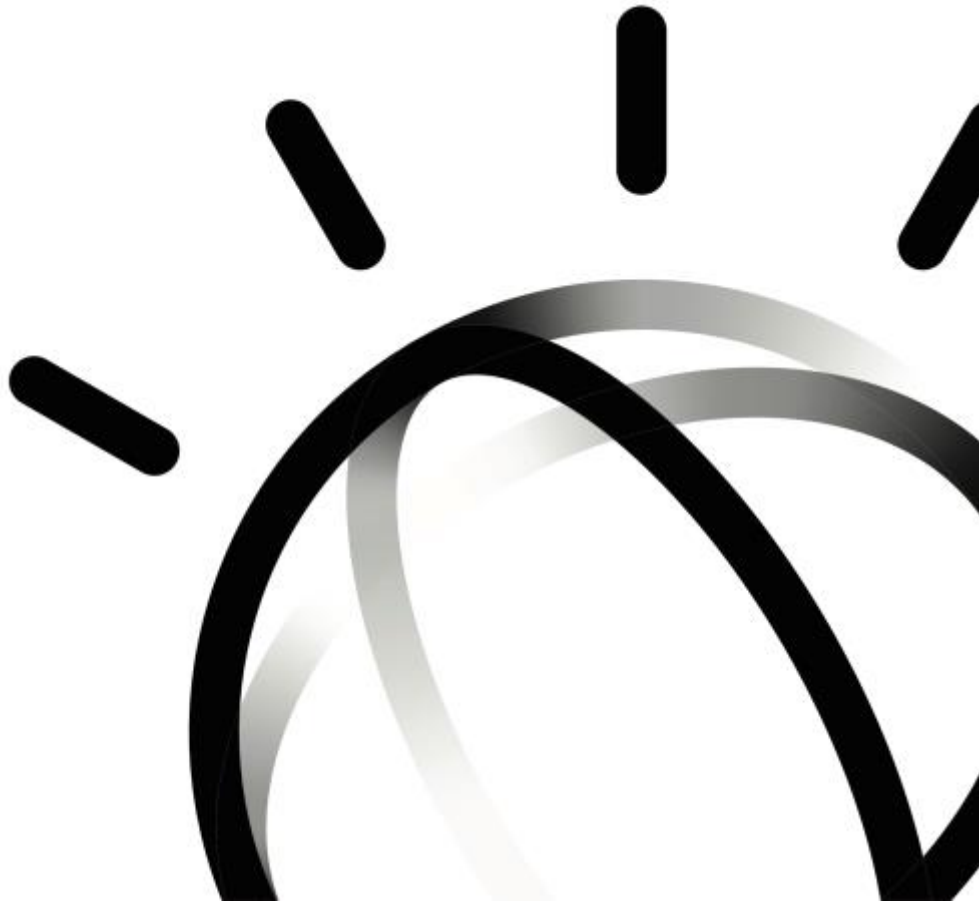# IBM Watson Solutions

Business and Academic Partners

Build a Virtual Agent by Combining the Conversation and the Discovery Services

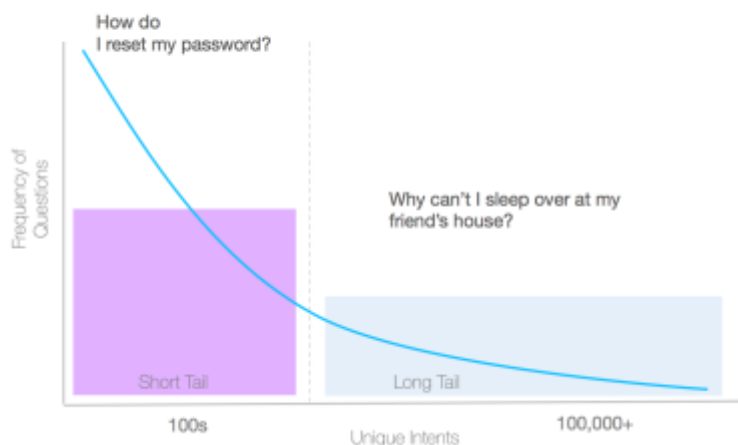## Prepared by Armen Pischdotchian

Version 2.0 August 2017

Overview

It is important to note, that to complete this workshop, you must have already completed the Conversation and the Discovery workshops. Some of the accounts and downloads from those labs are essential before you begin with these exercises. Suffice it to say, that the steps in this workshop are at a higher level and that is again, assuming that you have had ample detailed steps from prior experience.

In this workshop you connect the IBM Watson™ Discovery service and the Conversation service where you using a confidence threshold setting within the app.js file that flags returned responses from the Conversation service; if the confidence threshold is below a certain limit, then it invokes the Discovery service.

IBM Watson Conversation service is designed to answer the short tail of typical question distribution. The Discovery service 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.

Prerequisite

The following prerequisites omit the necessary node.js and the CLI downloads plus a few other handy downloads and that is because you have already performed those tasks from the prior labs.

- **Download the Conv-Disc.zip package**

  Complete the following steps:

  1. Direct your browser to the Academia web page: http://academia.mybluemix.net/workshop.html

  2. Scroll down and click the Virtual Agent link.

  3. Download the updated PDF from here and select Conv-Disco_v6 document.

- **Download Git bash**

  Direct your browser to the following location and install the OS appropriate package:

  https://git-scm.com/downloads

Building the app

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

1. Create a top-level directory on your system.

2. Extract the conv_disc.zip package you downloaded earlier in that top level working directory.

3. Open a command window or a terminal and change directory (`cd`) to the working directory.

4. 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.

5. From the same terminal, run the node package manager (npm) to install dependencies that node.js relies on for further processing of the code. Run both commands below. Notice that you have a new folder named node_modules (you may have to install those using super user, which will prompt you to enter machine access password).
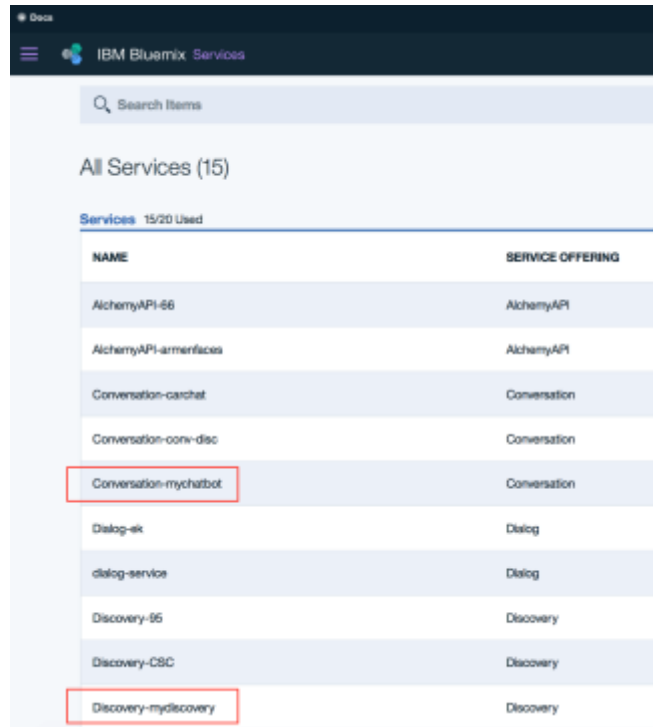
   ```
   Sudo npm install
   Sudo npm install –g gulp
   ```

6. Login into Bluemix: https://console.ng.bluemix.net

7. Click the **Dashboard** link from the top right corner (the three lines).
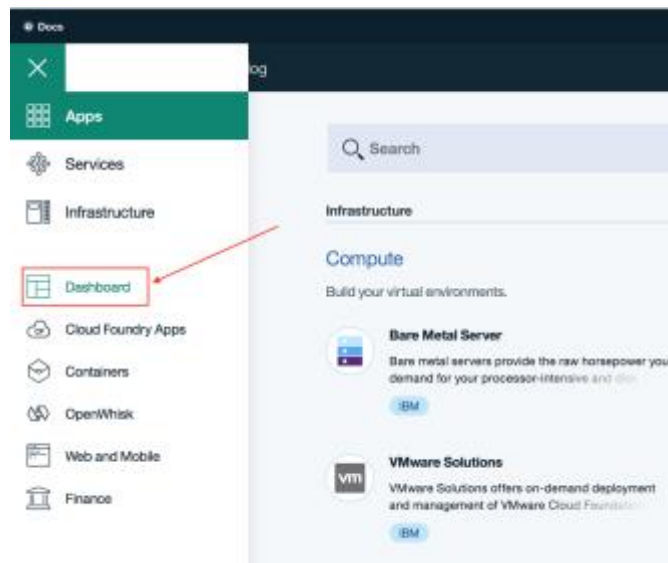
8. Scroll to the section where your services are listed. Below is an example from my Dashboard. You will be binding the Conversation and the Discovery services that you created in the prior workshops.
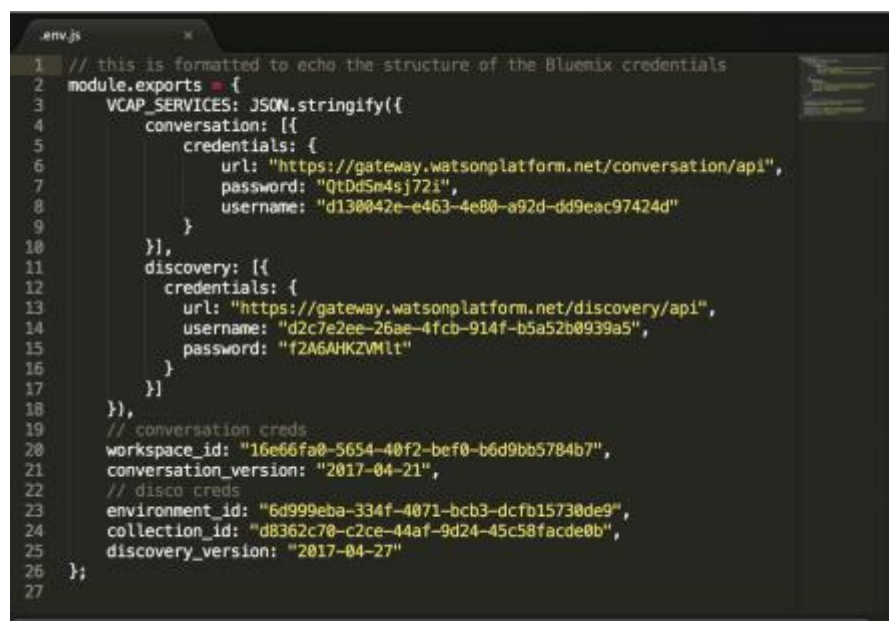
9. Click the **Conversations** service that you created from the Conversation workshop on your Dashboard and complete the following subtask:

   a. Click **Service credentials** and then **View credentials**.

   b. Copy and paste the username and password in a text editor

   c. Click the **Manage** link from the menu to the left.

   d. Click **Launch tool**.

   e. Click the three dots in the top right corner of the workspace and then select **View details**.

   f. Copy and paste the Workspace ID along with the service credentials in your text editor

10. Back to Bluemix, click the **Dashboard** link again and open your **Discovery** service. This is the service you created from the Discovery workshop that contains the AirBnb ground truth.

   a. Click **Service credentials** and then **View credentials**.

   b. Copy and paste the username and password in a text editor

   c. Click the **Manage** link from the menu to the left.

   d. Open the **AirBnB** collection.

   e. Copy and paste the `collection_id` and the `environment_id` in the same text editor.

You must now provide all parameters to the code that you had saved earlier in your text editor. Bear in mind that files that start with a dot "." are hidden system files. In a Mac OS, you can issue the command `ls -la` to view hidden files.

11. Open a new file with the text editor and copy-paste the below in that file; Or you can open the **.sample_env** file and populate it with the appropriate credentials where it says <enter here>.

12. Keep the double quotes (but remove the < > brackets) and take a close look at your copy/pasted double quotes. Ensure that they are san serif (straight up and down) not squiggly double quotes, as the case often is when copying from a rich text format. The service version dates are valid, don't change those.

```
// this is formatted to echo the structure of the Bluemix credentials
module.exports = {
    VCAP_SERVICES: JSON.stringify({
        conversation: [{
            credentials: {
                url: "https://gateway.watsonplatform.net/conversation/api",
                password: "<enter password here>",
                username: "<enter username here>"
            }
        }],
        discovery: [{
          credentials: {
            url: "https://gateway.watsonplatform.net/discovery/api",
            username: "<enter username here>",
            password: "<enter password here>"
          }
        }]
    }),
    // conversation creds
    workspace_id: "<enter workspace_id here, from details section, back of the tile",
    conversation_version: "2017-04-21",
    // disco creds
    environment_id: "<enter environment id here>",
    collection_id: "<enter collection id here>",
    discovery_version: "2017-04-27"
};
```

13. Save the file as **.env.js** (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. In the prior workshops is was just .env with this package, it is **.env.js**
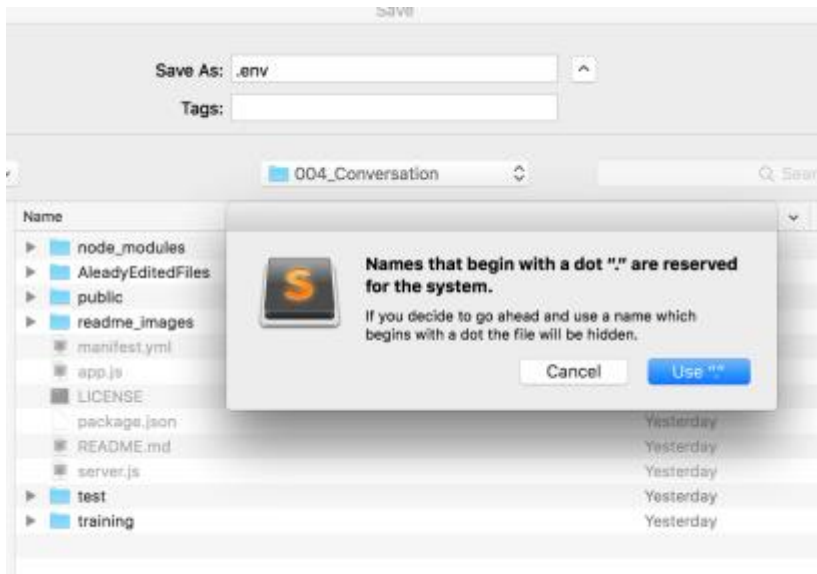
If you are on a PC, ensure that the file type is **All types**, not text or any other file extensions.

14. Go back to the terminal window and run the following command (notice the 75 percent confidence threshold).

```
node server.js gulp build && npm start
```

If for some reason you get errors, try running just: `npm start`

15. Open a new tab in your browser and enter **localhost:3000** for the URL address.

16. Run a conversation similar to the screen capture below (Watson's responses may vary depending on workspace dialog)

My Virtual Assistant

| Hi. What took you so long? What can I do for you my dear?

Hi how are you

| Well, you had me at hello; so, what can I do for you my dear?

what can you do for me?

| I'm trained to turn things on or off in the car, play music and find nearby locations, even show you the AirBnb Manhattan apartment reviews if interested?

find me a gas station

| There are a few gas stations nearby. Which one would you like to drive to?

Enter your question here...

17. The responses that appear in the screen capture (and likely on your screen as well) are from the Conversation service. At this point as the Virtual Agent the following question:

Find me a Manhattan apartment with great views of the Hudson

My Virtual Assistant

nearest
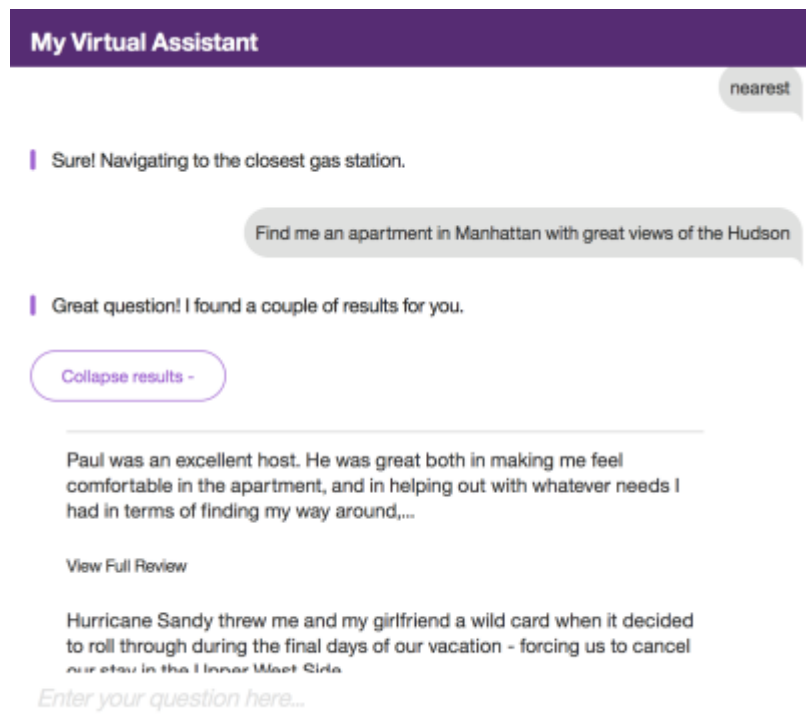
| Sure! Navigating to the closest gas station.

Find me an apartment in Manhattan with great views of the Hudson

| Great question! I found a couple of results for you.

Collapse results −

Paul was an excellent host. He was great both in making me feel comfortable in the apartment, and in helping out with whatever needs I had in terms of finding my way around,...

View Full Review

Hurricane Sandy threw me and my girlfriend a wild card when it decided to roll through during the final days of our vacation - forcing us to cancel our stay in the Upper West Side

Enter your question here...

Notice, it jumped to the Discovery service!

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, and click the Dashboard view from the left contents drop down link.
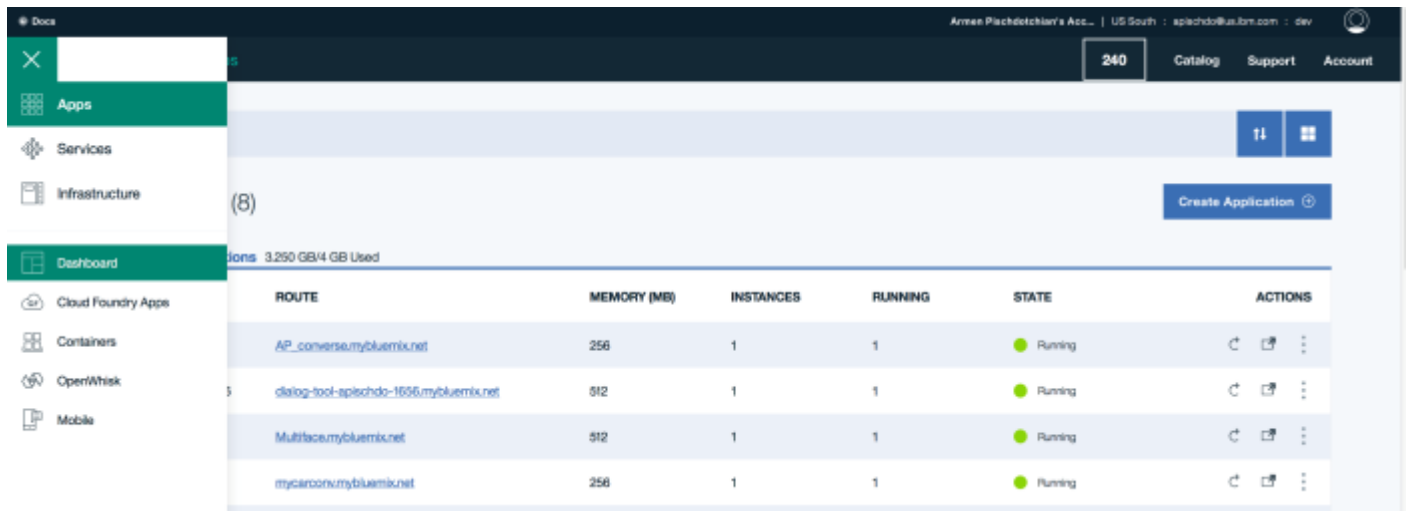

2. From the Dashboard view and click **Create App** (you may have to scroll down a bit; and yes, it used to say Create Application).

3. Scroll down and from the Cloud Foundry Apps, select the **Node.js Cloudant DB Web Starter**
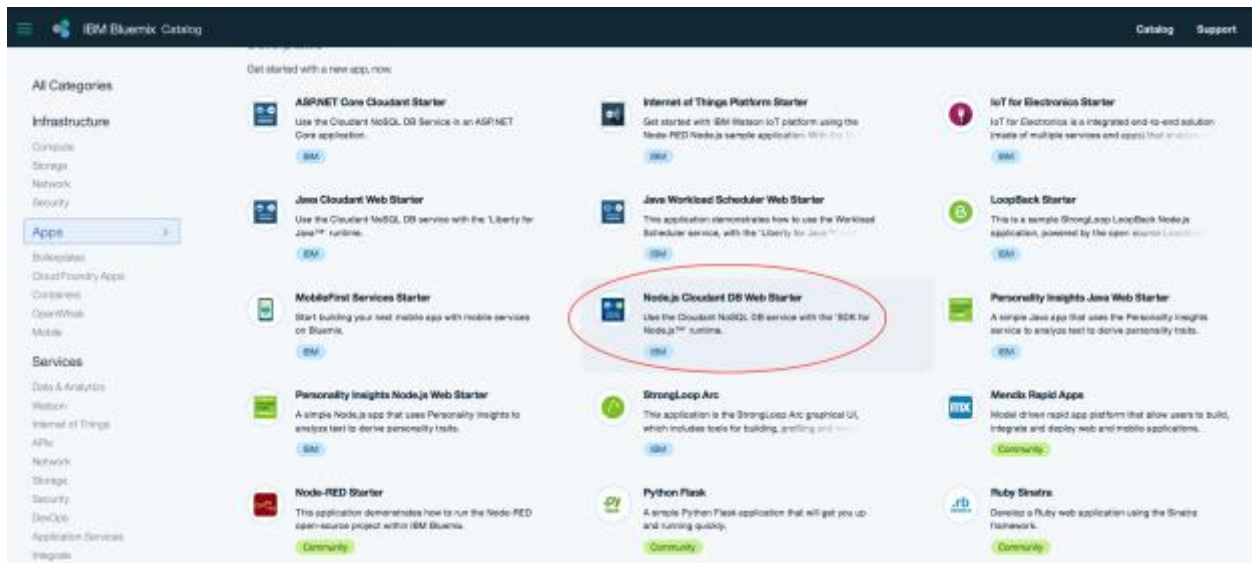
4. Specify a unique app name and host name and accept all other settings.

5. Click **Create**. Allow the necessary time for the app to start.

6. Click **Connections**.

7. Add an existing Conversations service: click **Existing service** and select the conversation service from the ensuing page.

8. Click **Cancel,** no need to restage now, you'll restage the app after adding both services.

9. Repeat the same procedure and add the Discovery service to the app.

10. Now click **Restage**.

11. Click **Runtime**

12. Click the **Environment variables** link in the center.

13. Scroll down and in the **User defined** section and click **Add**.
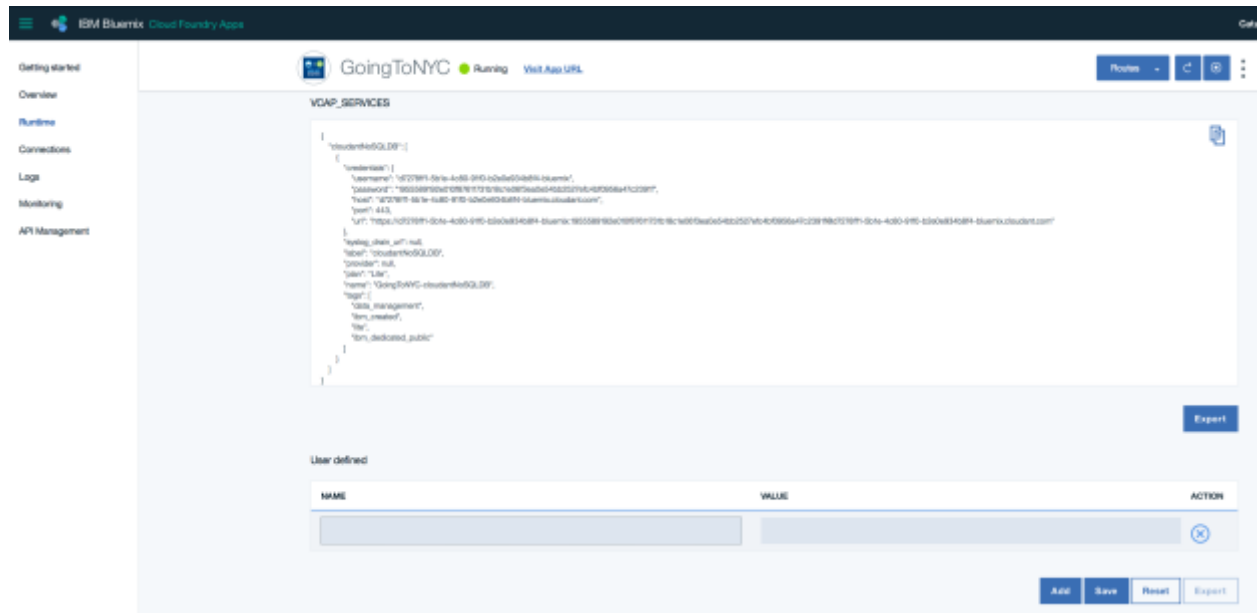
14. Add the following **Environment Variables** and Save them each time (case matters, all lower case for this lab). Refer to your .env file to get these values or the text editor where you have been saving these values:

Variable name	Value
workspace_id	From the Conversations service
collection_id	From Discovery
environment_id	From Discovery
intent_confidence	From app.js, line 92 (it is 0.75)
conversation_version	2017-04-21
discovery_version	2017-04-27

15. 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 Bluemix. 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.

16. Click **DOWNLOAD STARTER CODE**.

17. Extract the contents of the downloaded code in a temporary location.

18. 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.

19. One other edit that you must make is within the **dialog-constants.js** file.

20. Navigate inside your working directory as such: ui → modules → dialog-constants.js

21. Open the **dialog-constants.js** file and refer to lines 39 and 40.

22. Comment out line 39, and uncomment line 40, also specify your app name (the one that appears in the screen capture is an example I used) in the getResponse message call.

```
35
36                    },
37                    url: {
38                        //getWCSResponse: window.location.href.split('#')[0] + 'api/messa
39                        //getWCSResponse: "http://localhost:3000/api/message"
40                        getWCSResponse: 'https://almostthere.mybluemix.net/api/message'
41                    },
42                    ibmIdCookie: 'ibmid',
43                    usernameCookie: 'username',
44                    skipAuth: true,
45                    customGreeting: false
46
47                });
48    }());
49
```

23. Save the file.

Pushing the app to Bluemix

Now that you have edited your local copy with all necessary edits, you are ready to push it up to Bluemix.

24. Refer back to your Dashboard on Bluemix and open the app link (not the URL).

25. Click **Getting Started**.

26. Start from Step 3 onward from the **Getting Started** page on Bluemix 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)

```
% cf api https://api.ng.bluemix.net
% cf login -u apischdo@us.ibm.com -o apischdo@us.ibm.com -s dev
% cf push <the name you gave to your app in the manifest>
```

The above is just an example of my credentials and settings. Yours would have your email address, org name and space name. When logging in, use the same Bluemix password when you created the account.

Additional Self Paced Exercises

Let's get creative. Consider the following undertakings as you customize your virtual agent.

1. Change the conversation dialog within the Conversation service tooling. Have it greet you differently.

2. Change the confidence level from line 92 in the app.js file. Try other values, refresh the node and test the results on your local system (localhost:3000)

3. In the Discovery part of your customization work, replace the collection_id and the environment_id with the **Watson News** collection. Or perhaps, bind a new Discovery collection, your own corpus.

   Each time you change any code, to view it locally, remember to run the command: `npm install -g gulp`

4. Refresh the browser tab to see your changes.

Use the documentation frequently by clicking Learn more from the tooling interface and explore the contents of the doc from the left panel. Enjoy your discovery journey.