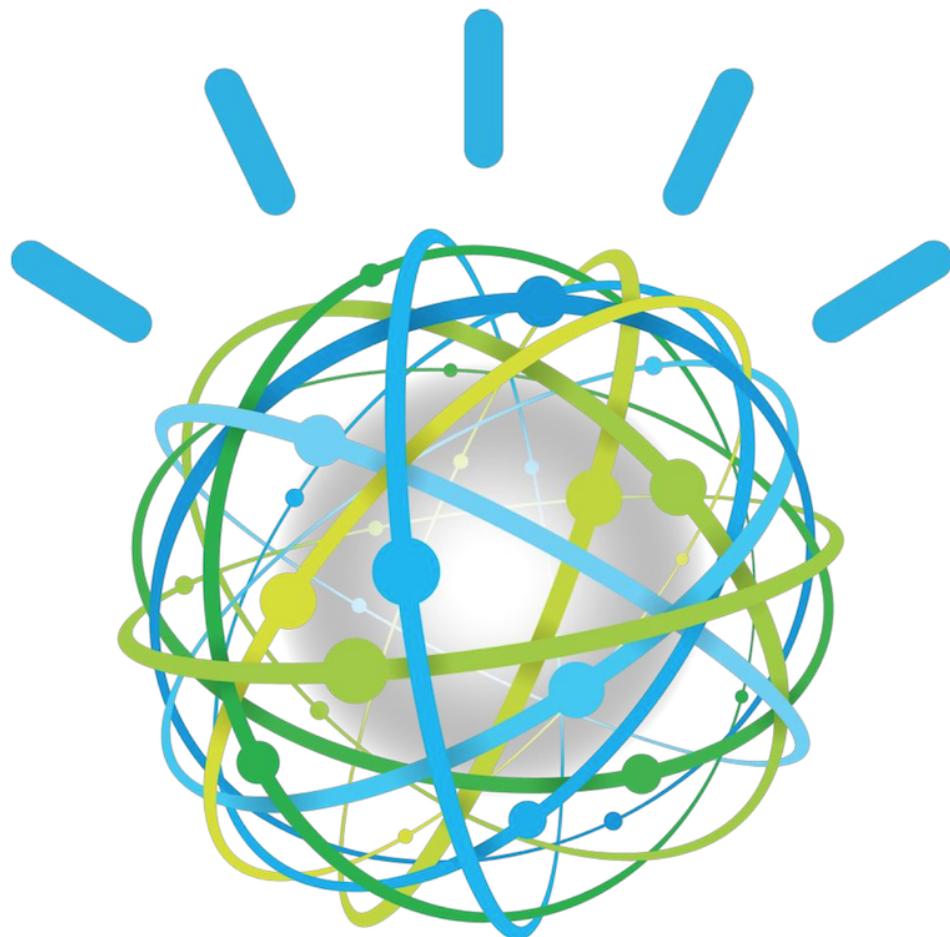


IBM Watson Solutions

Business and Academic Partners



Developing a Chatbot Using the IBM Watson Conversation Service

Prepared by Armen Pischdotchian

Version 5.1 November 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!

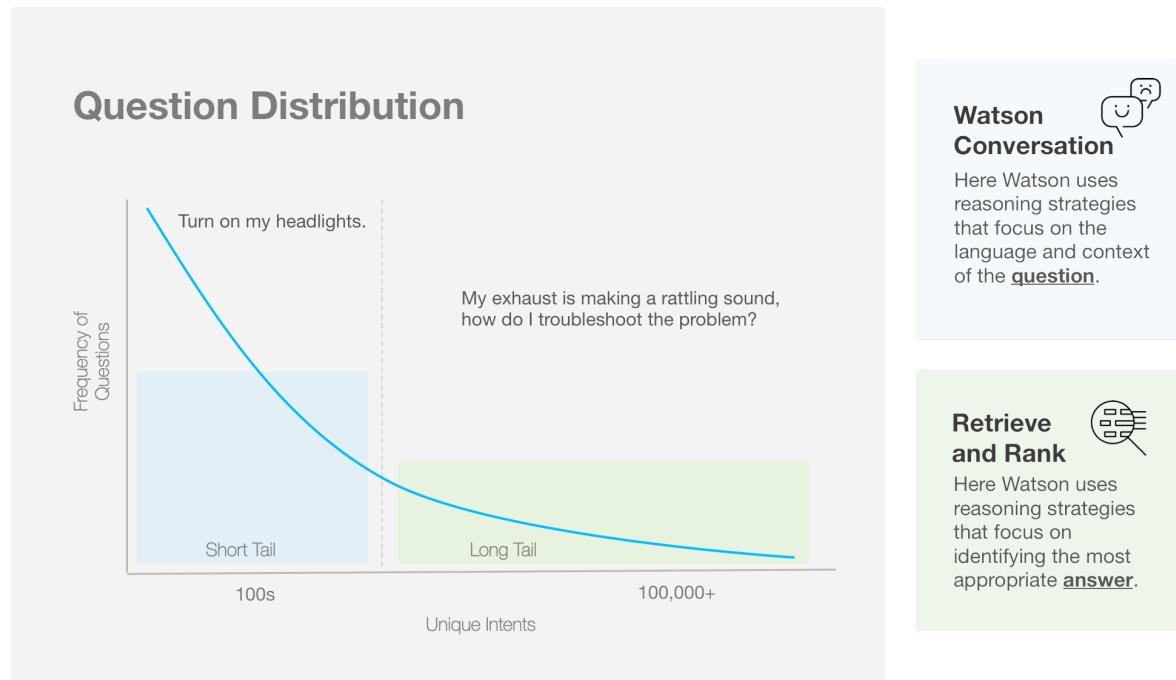
To get started, you will need to become an Academic Initiative member. Refer to this document for details:

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 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. If you are affiliated with a university, use your edu email.
3. Enter requested information: for **Org**, select the suggestions that are provided for you, for example, your email address; for **Space** name, you can also select the provided suggestions, such as dev; alternatively, you can specify your own values and click **Create Account**.
4. Look for the email confirmation. You will have to login once again into Bluemix after clicking the link from the email.

- **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**

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>

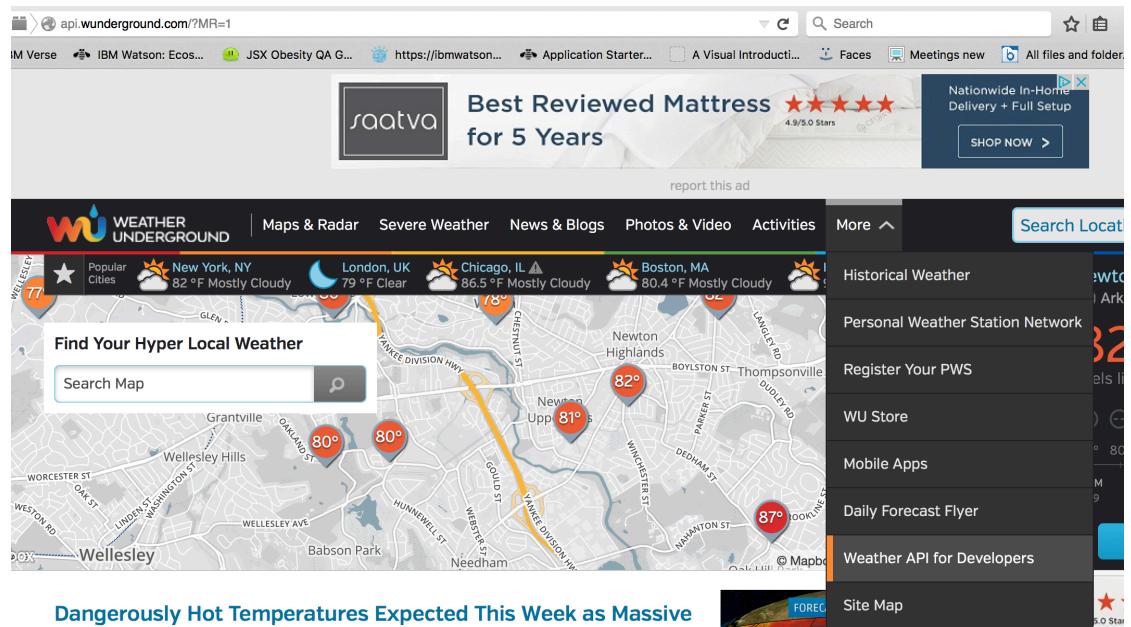
For Firefox browser complete these steps:

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*

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.

The screenshot shows the Weather Underground API key management interface. At the top, there's a navigation bar with links like Weather, Maps & Radar, Severe Weather, Photos & Video, Community, News, and More. On the right side of the header are icons for a star, a gear, and Sign Out. Below the header, a blue banner says "GET YOUR API KEY". Underneath, there are tabs for Analytics, Key Settings (which is selected), Featured Applications, Documentation, and Forums. A dropdown menu labeled "Select a Key to Customize" shows "5aa2ca764b80f41a - Chatbot". The main content area is titled "Edit API Key" and contains fields for Key ID (5aa2ca764b80f41a), Project Name (Chatbot), Company Website (<http://localhost:3000>), Contact Phone, Contact Email (apischdo@us.ibm.com), and an "Update" button. To the right of these fields is a "Regenerate API Key" section with a link to regenerate a new key if the current one has been compromised. It also includes a "Consequences" section with three bullet points: "You will need to change your apps to use the new key.", "Your statistics will be reset.", and "This action cannot be undone." There's a checkbox for "I understand the consequences." followed by a "Regenerate Key" button. Below this is a "Billing Information" section with a "View Billing" button and links for Modify, Upgrade, Downgrade, and Cancel. At the bottom, there's a section for "Customize a plan that suits your needs:" with three plans: STRATUS PLAN (selected), CUMULUS PLAN, and ANVIL PLAN. Each plan has a list of features: STRATUS PLAN includes Geolookup and Autocomplete; CUMULUS PLAN includes Geolookup and Autocomplete; ANVIL PLAN includes Geolookup and Autocomplete. A "TOTAL: \$0 USD per month" label is next to the plans, and a "Purchase Key" button is at the far right.

- **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

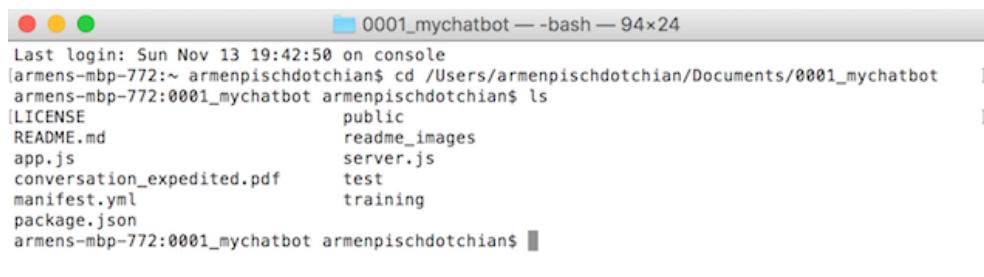
This table summarizes the steps that you need to complete to stand up an instance of the Watson Conversation service and integrate it an eternal service.

Step	Where do I go?	What do I do?
1	Github: https://github.com/apischdo/Bluemix-workshop-assets	Download the ConversationMaster_expedited.zip (you can also download everything that you see there). Alternatively, can also use the following command from a command prompt or your terminal: <code>git clone https://github.com/apischdo/Bluemix-workshop-assets.git</code>
2	Your machine, the terminal window	Run the command <code>npm install</code> 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 <code>car_workspace.js</code> file
6	Your machine: create a new system internal file	This <code>.env</code> 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 <code>node server.js</code> command
8	Your machine: use the provided <code>weather.js</code> file	Update the <code>conversation.js</code> and the <code>app.js</code> files. You include the API from the Weather Company in this step.
9	Your machine	From the same directory, run the <code>node server.js</code> command to reflect your changes.
10	Bluemix:	Create a “placeholder” app in Bluemix, connect it to your existing conversation service, add custom credentials and download the Starter Kit. This action will serve two purposes: <ul style="list-style-type: none"> You will use the resulting <code>manifest.yml</code> file in your local app You can now see and use the commands to upload your app to Bluemix that will include the new <code>manifest.yml</code> file.
11	Your machine	Run the commands that you see in Bluemix on your local system to upload the file. This will replace the “placeholder” app that you created earlier.
12	Bluemix	View your app in Bluemix and test it.

Using the Conversation service

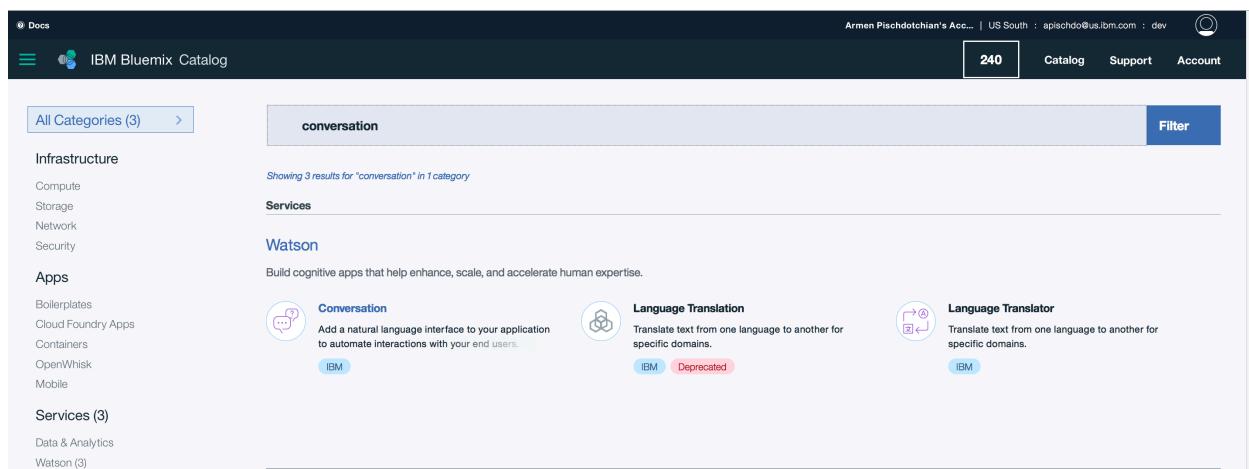
Let's begin our journey with a few good housekeeping practices:

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/ConversationMaster_expedited.zip
3. Extract the contents of the **ConversationMaster_expedited.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)).



```
0001_mychatbot — -bash — 94x24
Last login: Sun Nov 13 19:42:50 on console
[armens-mbp-772:~ armenpischdotchian$ cd /Users/armenpischdotchian/Documents/0001_mychatbot
armens-mbp-772:0001_mychatbot armenpischdotchian$ ls
[LICENSE           public
 README.md         readme_images
 app.js            server.js
 conversation_expedited.pdf   test
 manifest.yml      training
 package.json
armens-mbp-772:0001_mychatbot armenpischdotchian$ ]
```

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 Bluemix: <https://console.ng.bluemix.net>
8. Click the **Catalog** tab.
9. Search for the **Conversation** service and click that tile.



The screenshot shows the IBM Bluemix Catalog interface. At the top, there is a navigation bar with links for 'Docs', 'IBM Bluemix Catalog', '240', 'Catalog', 'Support', and 'Account'. A user profile is shown on the right. Below the navigation, a search bar contains the text 'conversation'. On the left, there is a sidebar with categories: 'All Categories (3)', 'Infrastructure' (with sub-options: Compute, Storage, Network, Security), 'Watson' (with sub-option: Watson), 'Apps' (with sub-options: Boilerplates, Cloud Foundry Apps, Containers, OpenWhisk, Mobile), and 'Services (3)' (with sub-options: Data & Analytics, Watson (3)). The main content area displays search results for 'conversation' in the 'Watson' category. It shows three services: 'Conversation' (represented by a speech bubble icon), 'Language Translation' (represented by a hexagon icon), and 'Language Translator' (represented by a circular icon). The 'Conversation' service is described as adding a natural language interface to an application to automate interactions with end users. The 'Language Translation' service is described as translating text from one language to another for specific domains. The 'Language Translator' service is marked as 'Deprecated'. The entire interface has a dark-themed design.

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 Bluemix once again, then you can see the Create button in the bottom right corner).

Watson

Conversation-mychatbot

Manage Service Credentials Connections

Conversation

Add a natural language interface to your application to automate interactions with your end users. Common applications include virtual agents and chat bots that can integrate and communicate on any channel or device.

Developer resources:

- Documentation
- Demo

Conversation tooling

Train bots with the Watson Conversation service through an easy-to-use web application. Designed so you can quickly build

Launch tool

11. Click the **Launch Tool** button.
12. Login once again using the same credentials that you use to login Bluemix.
13. Click **Import**.
14. Navigate to the **training/car_workspace.json** file and click **Open** to import the file.

Watson Conversation

Instances: Conversation-mychatbot

Import a workspace

Select a JSON file then choose which elements from the workspace to import.

Choose a file

Import

Everything (Intents, Entities)

Intents and Entities

Documents

Format: *.json

Cancel Open

Name	Date M
0001_mychatbot	3:18 P
node_modules	3:14 P
public	2:41 P
readme_images	2:41 P
training	2:41 P
sample_manifest.yml	10/6/1
weather.js	7/22/1
car_workspace.json	7/19/1
conversation_expedited.pdf	10/6/1
app.js	10/6/1
manifest.yml	7/19/1
LICENSE	7/19/1
package.json	7/19/1
README.md	7/19/1

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 (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=
```

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 Bluemix, and obtain the username and password of the Conversation service from the **Service Credential** link and then click **View Credentials**.

KEY NAME	DATE CREATED	ACTIONS
Credentials-1	Nov 16, 2016 - 07:27:41	View Credentials

```
{
  "url": "https://gateway.watsonplatform.net/conversation/api",
  "password": "UL7DCJUQ2Eh2",
  "username": "A502832b-09ed-40e7-9352-9ddb5f098d88"
}
```

19. Enter all three parameters in a new file in Sublime or Notepad ++.

Watson Conversation

Workspaces

Created: 7/20/2016, 2:24:18 PM
Last modified: 7/20/2016, 2:24:18 PM
[Documentation](#)
[Bluemix](#)

Workspace ID: 5df54f2f-1f3a-4cb1-aaef-f0a4427178b6

13 Intents 8 Entities 64 Dialog nodes

.env

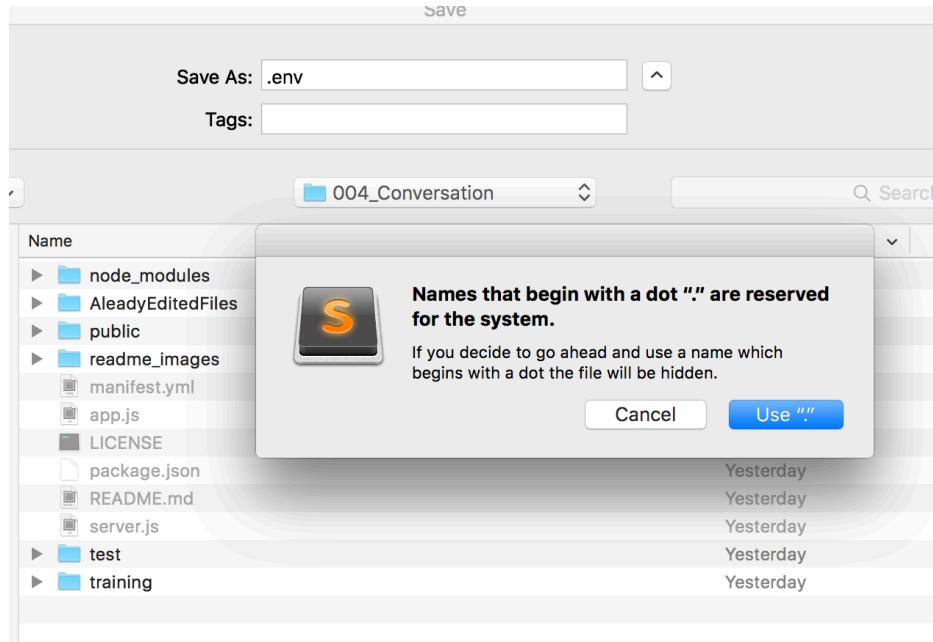
```

1 # Environment variables
2 WORKSPACE_ID=5df54f2f-1f3a-4cb1-aaef-f0a4427178b6
3 CONVERSATION_USERNAME=01f8ecc8-1701-4ed1-a1f9-0e76bd6b0d69
4 CONVERSATION_PASSWORD=0WYvfcMhoy

```

Get these values from the Conversation service tile: from environment variables

20. 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.



21. You are now ready to edit the app.js file. Open the **app.js** file.
 22. 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.
 23. Save the **app.js** file.
 24. Go back to the terminal window and run the following command:

```
Node server.js
```

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

The screenshot shows a Watson Conversation interface. On the left, a user conversation is displayed with messages from both the user and the system. On the right, the raw JSON input and output payloads are shown.

User input

```
1 {
2   "input": {
3     "text": "What is the weather like tomorrow?"
4   },
5   "context": {
6     "conversation_id": "b05f7569-3df3-4695-bc77-ca36d9057c1c",
7     "system": {
8       "dialog_stack": [
9         "root"
10      ],
11      "dialog_turn_counter": 5,
12      "dialog_request_counter": 5
13    },
14    "defaultCounter": 0,
15    "reprompt": true
16  }
17 }
```

Watson understands

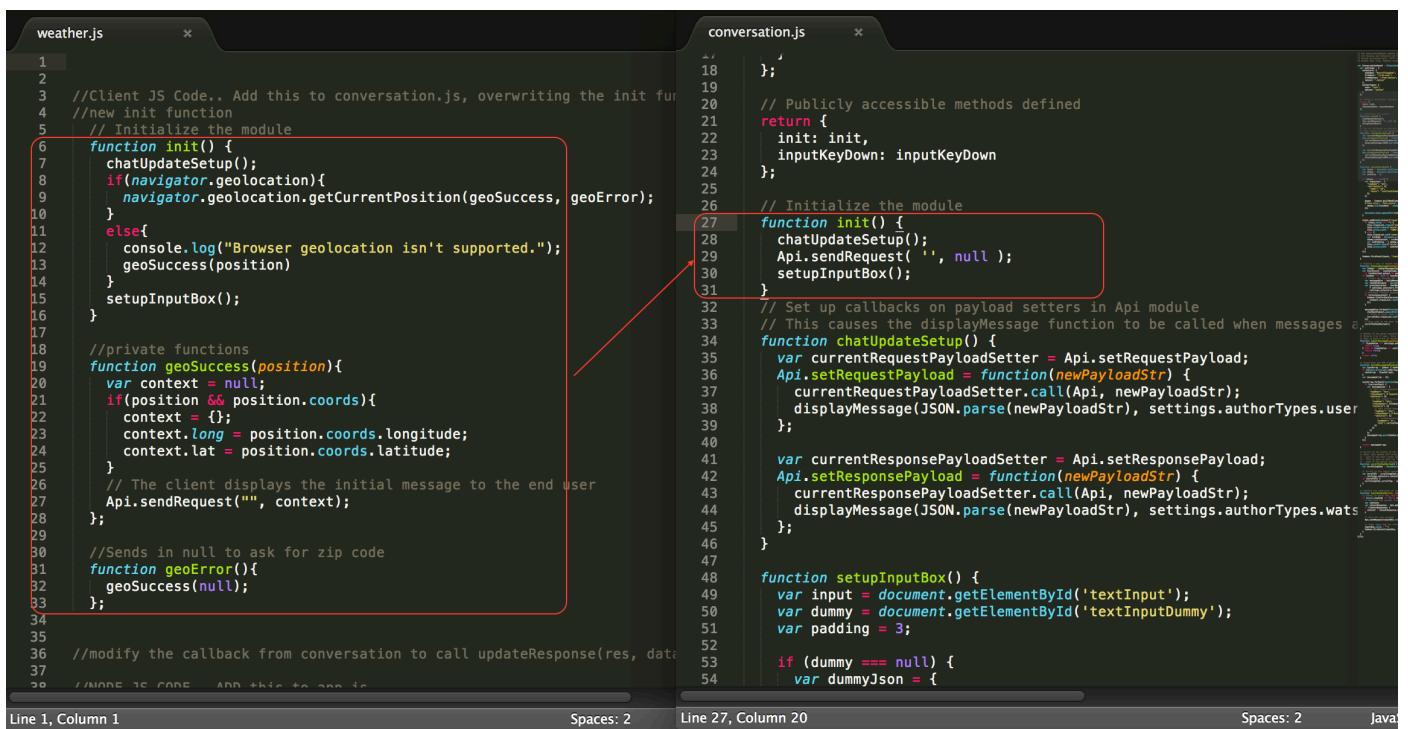
```
1 {
2   "input": {
3     "text": "What is the weather like tomorrow?"
4   },
5   "context": {
6     "conversation_id": "b05f7569-3df3-4695-bc77-ca36d9057c1c",
7     "system": {
8       "dialog_stack": [
9         "root"
10      ],
11      "dialog_turn_counter": 5,
12      "dialog_request_counter": 5
13    },
14    "defaultCounter": 0,
15    "reprompt": true
16  }
17 }
```

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.

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.



```
weather.js
1
2
3 //Client JS Code.. Add this to conversation.js, overwriting the init fun
4 //new init function
5 // Initialize the module
6 function init() {
7   chatUpdateSetup();
8   if(navigator.geolocation){
9     navigator.geolocation.getCurrentPosition(geoSuccess, geoError);
10  }
11  else{
12    console.log("Browser geolocation isn't supported.");
13    geoSuccess(position)
14  }
15  setupInputBox();
16}
17
18 //private functions
19 function geoSuccess(position){
20   var context = null;
21   if(position && position.coords){
22     context = {};
23     context.long = position.coords.longitude;
24     context.lat = position.coords.latitude;
25   }
26   // The client displays the initial message to the end user
27   Api.sendRequest("", context);
28 }
29
30 //Sends in null to ask for zip code
31 function geoError(){
32   geoSuccess(null);
33 }
34
35
36 //modify the callback from conversation to call updateResponse(res, data)
37 //MORE JS CODE... ADD THIS TO APP.js
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

conversation.js
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
```

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?

The screenshot shows a Watson Assistant interface. On the left, there is a text input field containing "type something". Above it, a message from the user says "Hi. It looks like a nice drive today. What would you like me to do?". Below this, two messages from the user are shown in rounded green bubbles: "turn on the wipers" and "what is the weather like today". To the right, the "User input" section displays the JSON payload for each message. The first message's payload is:

```
1 {
2   "input": {
3     "text": "turn on the wipers"
4   },
5   "context": {
6     "long": -71.4708425,
7     "lat": 42.549521999999996,
8     "conversation_id": "d56af184-882f-4b18-967a-069bc116084e",
9     "system": {
10       "dialog_stack": [
11         "root"
12       ],
13       "dialog_turn_counter": 2,
14       "dialog_request_counter": 2
15     },
16     "defaultCounter": 0,
17     "reprompt": true
18   }
19 }
```

The second message's payload is:

```
1 {
2   "input": {
3     "text": "what is the weather like today"
4   },
5   "context": {
6     "long": -71.4708425,
7     "lat": 42.549521999999996,
8     "conversation_id": "d56af184-882f-4b18-967a-069bc116084e"
9   }
10 }
```

Below the user input, the "Watson understands" section shows the same JSON payloads with minor differences in the "reprompt" field.

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. Click **Create new+** 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).

The screenshot shows the 'Entities' tab selected in the Car_Dashboard conversation. A new entity named '@day' has been created. It has a value 'today' and two synonyms: 'this afternoon' and 'tonight'. There is a 'Done' button at the top right.

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**.

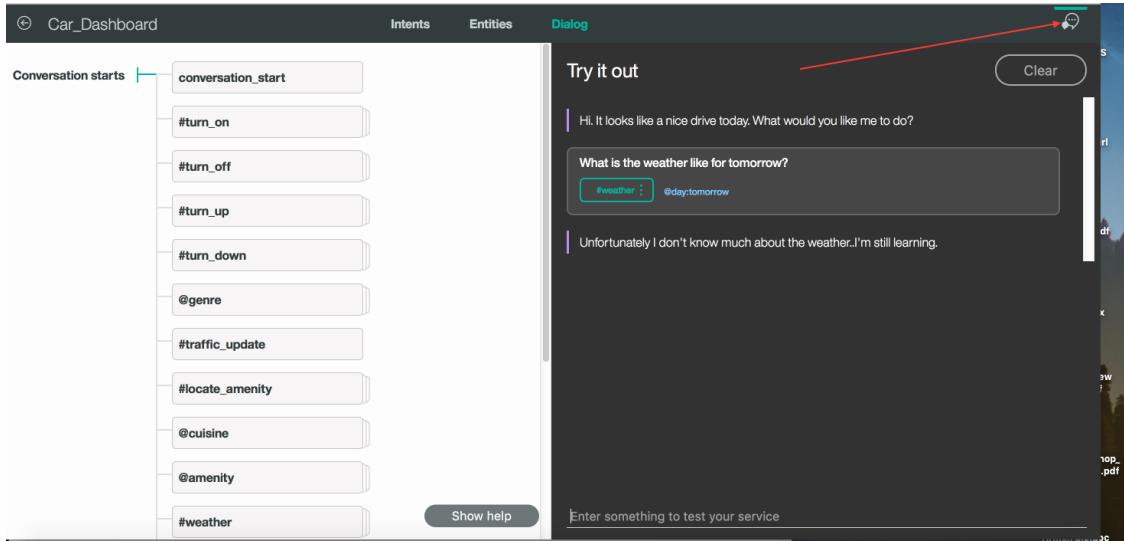
The screenshot shows the 'Entities' tab selected in the Car_Dashboard conversation. The entity '@day' now includes a value 'tomorrow'. It also has two new synonyms: 'tomorrow evening' and 'tomorrow night'. Other entities like 'this afternoon' and 'tonight' are visible in the list below.

7. Click **Create** (or Done...the UI may have changed).
8. Note, # sign signifies Intent and @ sign signifies Entity. The end result looks like the image below:

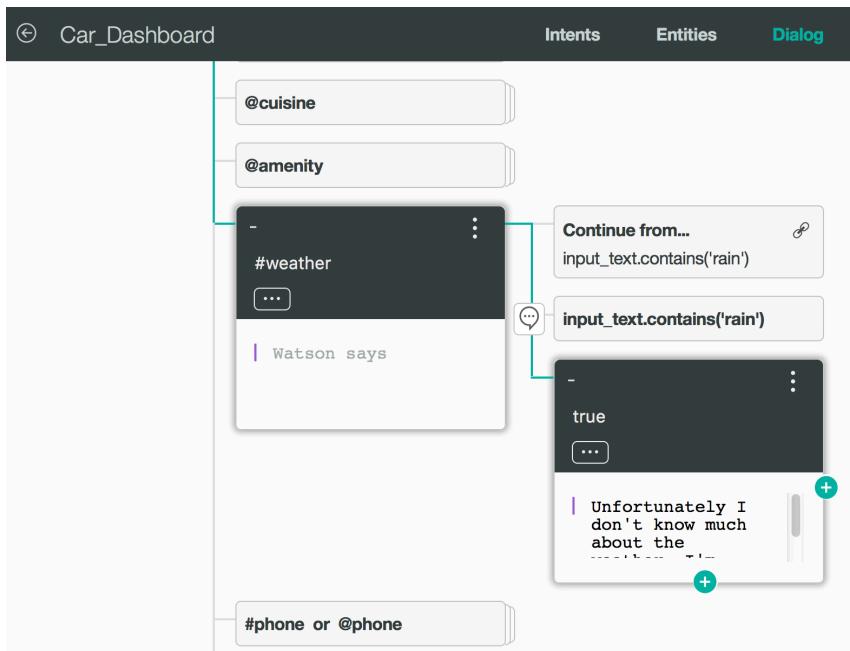
The screenshot shows the 'Entities' tab selected in the Car_Dashboard conversation. The entity '@day' is listed along with its values 'today' and 'tomorrow'. At the top, there is a 'Create' button and a '+' sign. The interface indicates 9 entities and allows sorting by newest.

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:

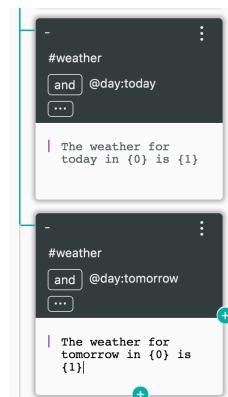


11. Close the chat window and click the #weather Dialog node and expand the tree, notice where the chat got its response!



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 sibling Dialog node (if it goes across, it is a child node). Nodes that are siblings (they are stacked on top of each other), are new conversations. Nodes that traverse across are child nodes (pertain to the same conversation) and are conditioned on the parent 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}
(Type the syntax, don't copy/paste it from here).
There is the concept of conditions, if the condition is true then the dialog code gets executed.
17. Click the plus sign *underneath the dialog node* you just created and add another node: **#weather**
18. Add the condition of tomorrow: **@day:tomorrow**
19. Include the same script:
The weather for tomorrow in {0} is {1}



If you have the chat open or if you open it right away, notice 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.

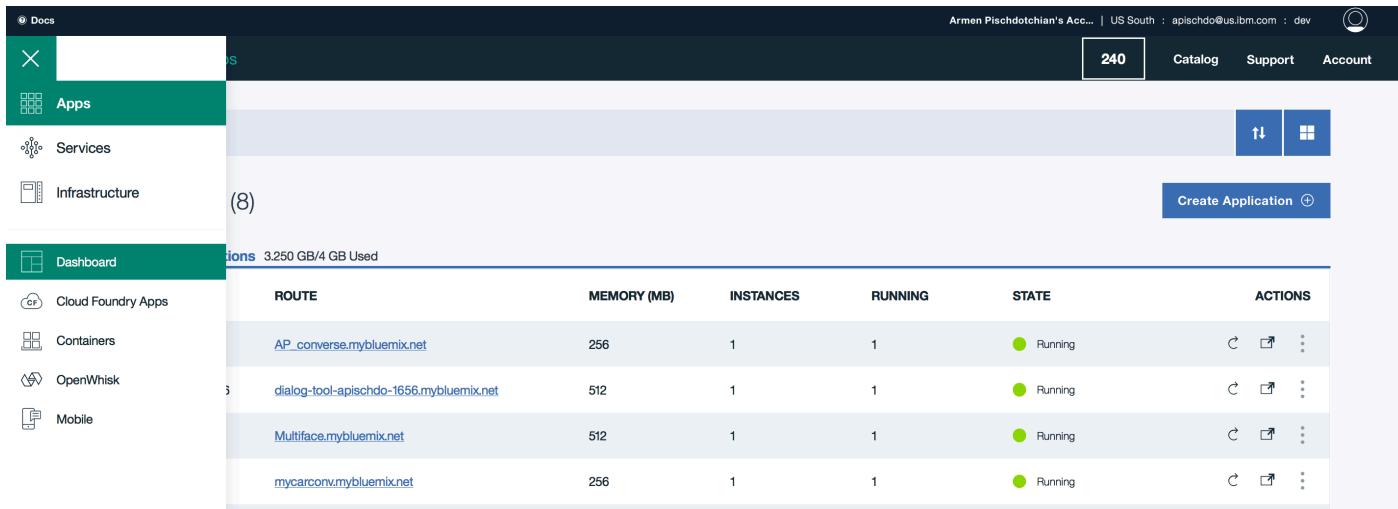
20. At this point, go back to your command console and run the `node server.js` command again (you may have to Control+C to stop the current process).
21. Ask the question: "How is the weather today?" and notice the correct weather forecast for your location.

But if you ask the same question from the Conversation service, it will give you the syntax answer: `The weather for tomorrow in {0} is {1}` and that is because the service on Bluemix does not know about your local app; not yet anyway. Your next step is 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 your app is deployed properly.

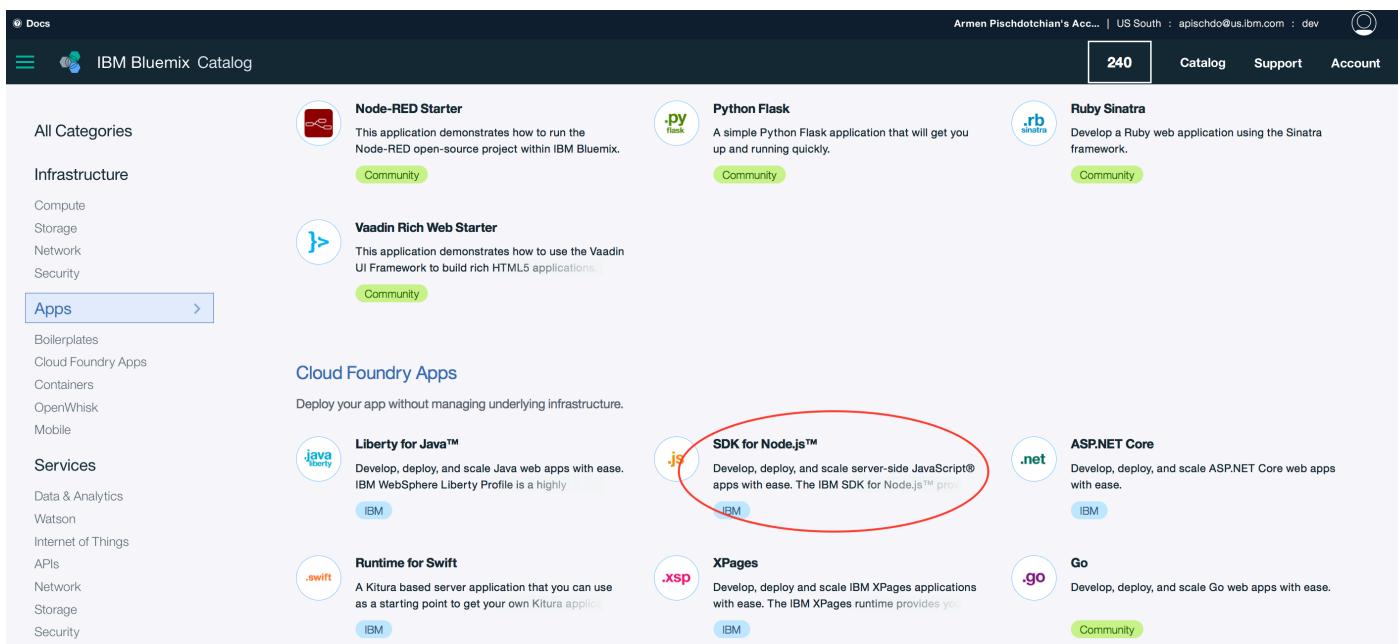
1. Back to Bluemix, and click the Dashboard view from the left contents drop down link.



The screenshot shows the IBM Bluemix dashboard with the 'Dashboard' link selected in the left sidebar. The main area displays a table of four running Cloud Foundry Apps:

ROUTE	MEMORY (MB)	INSTANCES	RUNNING	STATE	ACTIONS
AP_converse.mybluemix.net	256	1	1	Running	⋮
dialog-tool-apischdo-1656.mybluemix.net	512	1	1	Running	⋮
Multiface.mybluemix.net	512	1	1	Running	⋮
mycarconv.mybluemix.net	256	1	1	Running	⋮

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



The screenshot shows the IBM Bluemix Catalog with the 'Cloud Foundry Apps' section selected. The 'SDK for Node.js™' option is highlighted with a red oval.

Cloud Foundry Apps
Deploy your app without managing underlying infrastructure.

SDK for Node.js™
Develop, deploy, and scale server-side JavaScript® apps with ease. The IBM SDK for Node.js™ provides you with a simple API to interact with the IBM Cloud environment.

4. Specify a unique name, you may have to include your name, for example, I used Armenchatbot.
5. Click **Connections** link from the left panel.
6. Click **Existing** and find the very Conversation service that you created earlier in this lab (for example: mychatbot)

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

The screenshot shows the IBM Bluemix Cloud Foundry Apps dashboard. On the left, there's a sidebar with links: Docs, Dashboard, Getting Started, Overview, Runtime (which is highlighted in blue), Connections, Logs, and Monitoring. In the main area, there's a card for an app named 'Armenchatbot'. The card has a 'js' icon, a status message 'Status: Your app is restaging' (circled in red), and a 'Conversation' section with a 'Conversation-mychatbot' entry. At the bottom of the card are 'View Credentials' and 'Docs' buttons.

- Click the **Runtime** link in the left panel.
- 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).

The screenshot shows the IBM Bluemix Cloud Foundry Apps dashboard with the 'Runtime' tab selected. In the top right, there are buttons for View App, Catalog, Support, and Account. Below the tabs, there are three buttons: Memory and Instances, Environment Variables (which is highlighted in blue), and Files. The 'VCAP_SERVICES' section contains a JSON object for the Conversation service:

```
{
  "conversation": [
    {
      "credentials": {
        "uri": "https://gateway.watsonplatform.net/conversationV1",
        "password": "0dx4wXK9mH",
        "username": "7ef055af-5c98-4ed5-ba8d-5b8b7f444cc7"
      },
      "label": "conversation",
      "provider": null,
      "plan": "free",
      "name": "Conversation-mychatbot",
      "tags": [
        "assistant",
        "bot"
      ],
      "version": "2017-05-22"
    }
  ]
}
```

The 'User Defined' section shows a table with one row:

NAME	VALUE	ACTION
WORKSPACE_ID	f1e202bc-0a87-4f3a-83de-2ecbfcc91dd99	X

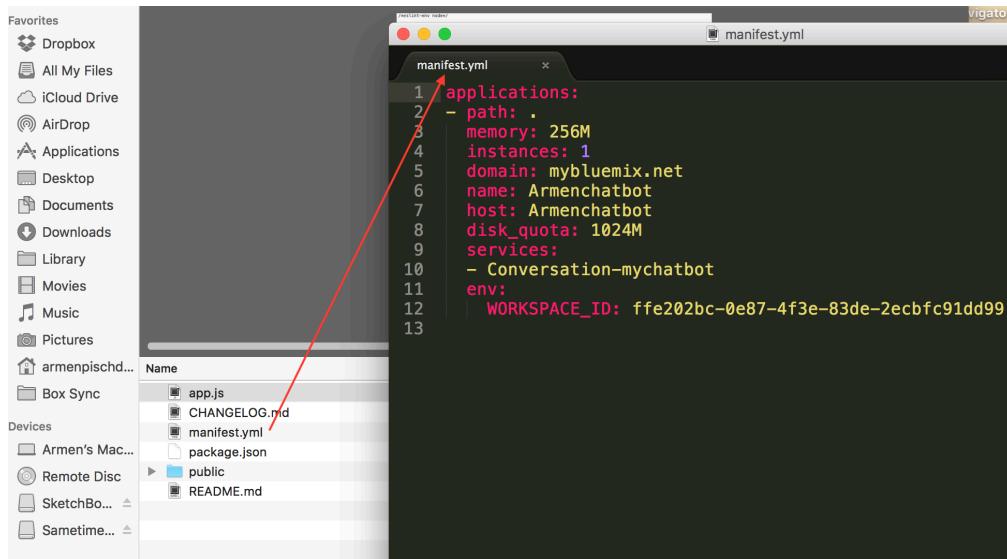
At the bottom right of the table are buttons for Add, Save, Reset, and Export. A red arrow points from the 'Value' field of the 'WORKSPACE_ID' row to the 'Save' button.

- Click **Save**.
- 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.

- Click **DOWNLOAD STARTER CODE**.
- Extract the contents of the downloaded code in a temporary location.

14. 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.



15. Start from Step 4 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 "Armenchatbot"
```

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.

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

```
0001_mychatbot — cf push Armenchatbot — 94x49
[armens-mbp-772:0001_mychatbot armenpischdotchian$ cf api https://api.ng.bluemix.net
Setting api endpoint to https://api.ng.bluemix.net...
OK

API endpoint: https://api.ng.bluemix.net (API version: 2.54.0)
User: apischdo@us.ibm.com
Org: apischdo@us.ibm.com
Space: dev
[armens-mbp-772:0001_mychatbot armenpischdotchian$ cf login -u apischdo@us.ibm.com -o apischdo@us.ibm.com -s dev
API endpoint: https://api.ng.bluemix.net

>Password>
Authenticating...
OK

Targeted org apischdo@us.ibm.com
Targeted space dev

API endpoint: https://api.ng.bluemix.net (API version: 2.54.0)
User: apischdo@us.ibm.com
Org: apischdo@us.ibm.com
Space: dev
[armens-mbp-772:0001_mychatbot armenpischdotchian$ cf push "Armenchatbot"
Using manifest file /Users/armenpischdotchian/Documents/0001_mychatbot/manifest.yml

Updating app Armenchatbot in org apischdo@us.ibm.com / space dev as apischdo@us.ibm.com...
OK

Using route Armenchatbot.mybluemix.net
Uploading Armenchatbot...
Uploading app files from: /Users/armenpischdotchian/Documents/0001_mychatbot
Uploading 65.2M, 22185 files
Done uploading
OK
Binding service Conversation-mychatbot to app Armenchatbot in org apischdo@us.ibm.com / space dev as apischdo@us.ibm.com...
OK

Stopping app Armenchatbot in org apischdo@us.ibm.com / space dev as apischdo@us.ibm.com...
OK

timeout connecting to log server, no log will be shown
Starting app Armenchatbot in org apischdo@us.ibm.com / space dev as apischdo@us.ibm.com...
```

17. From within your application click **View App** (you may need to Restart the app again from Bluemix).

18. Accept geo-location sharing request.

19. Ask away some typical car questions and then hit it with the weather questions.

Congratulations, you just deployed a chatbot onto Bluemix.