IBM Cloud

Building a Watson Chatbot

Workshop – Node-Red

Lab Guide

Notices and Disclaimers

Document Revision History

Rev #	File Name	Date
1.0	Building a Watson Chatbot	January 2018

Prepared & Revised by:
Chris Tyler – ctyler@us.ibm.com
Ashley Troggio – atroggio@us.ibm.com
Darrel Pyle – darrel.pyle@ibm.com

Table of Contents

Lab Environment Overview

Software and Tools

Software	Link
IBM Cloud	https://www.ibm.com/cloud/
IBM Watson Conversation Service	https://www.ibm.com/watson/services/conversation/
WolfPack Cogintiive Chatbot GitHub repository	https://ibm.biz/wolfpack-cognitive-chatbot

Module 1: Pre-Work

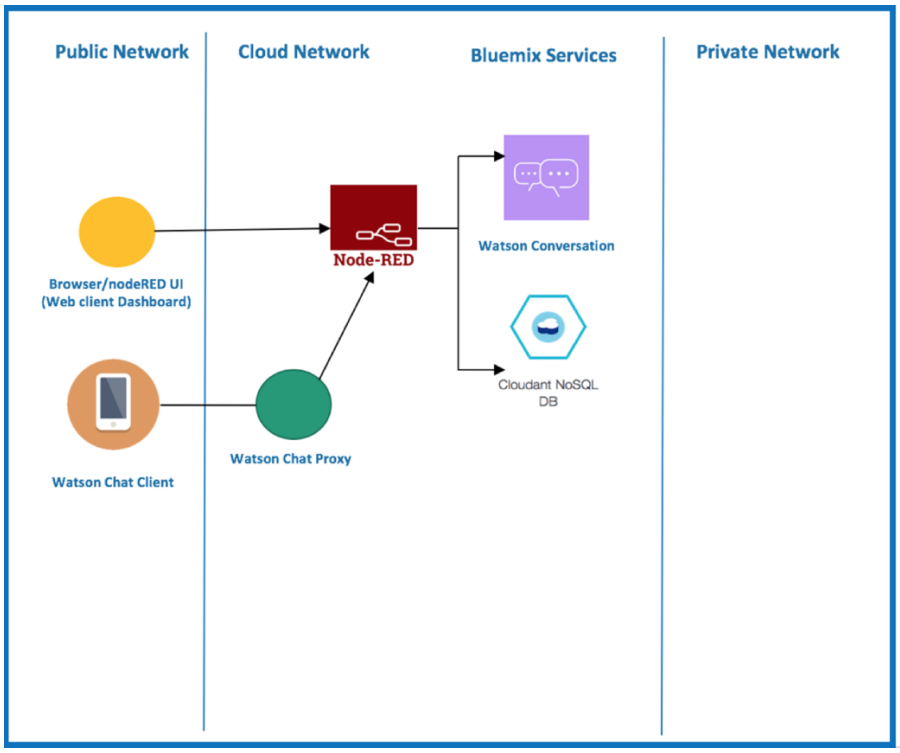
Purpose:	This lab introduces the Watson Conversation Service (WCS) workflow. After completing the lab, you should be able to understand: • WCS architecture for cloud native applications • Basics of the WCS API • WCS foundational terminology • WCS integration with applications

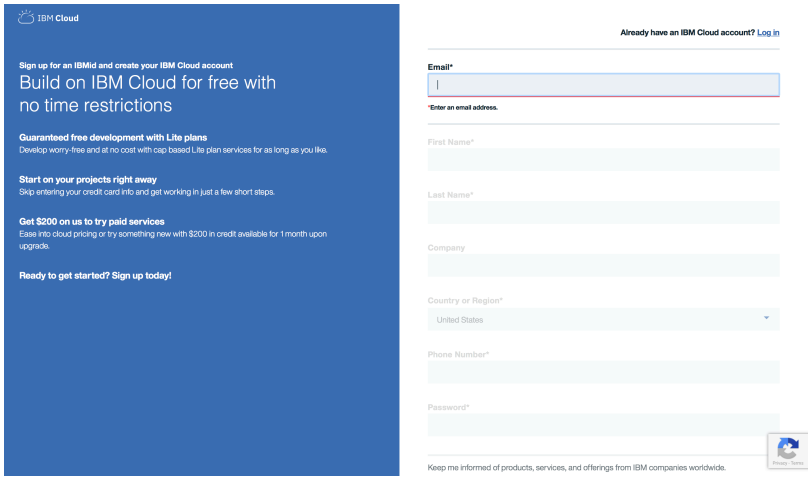
Tasks:	Tasks you will complete in this lab exercise include: • Signing up for an IBM Cloud account

Module 1: Pre-Work Overview

1. Architecture
2. Application Overview
3. Terminology
4. Prerequisites

Module 1: Pre-Work Information

Step	Action
1	**<u>Architecture</u>** Below is the architecture overview of the workshop, Watson Conversation Service (WCS). This architecture is consistent with the reference implementations of WCS for cloud native applications using microservices. 
2	**<u>Application Overview</u>** This workshop is intended to help you understand the basics of the Watson Conversation Service (WCS) as part of the Watson APIs. WCS is a question and answer system that focuses on providing a dialog type of experience between the user and the conversation system. This style of interaction is commonly called a bot. The intent of this lab is to leverage the WCS capabilities. We will enable through a dialog approach, WCS interacting with data from the weather service API and the ability to issue commands to change the color of our dialog background. Though the example is simple, it will provide you with a solid understanding of the core pieces of WCS

Step	Action
3	**<u>Terminology</u>** WCS has terms that are foundational for understanding the service. **Intent:** An *intent* represents the purpose of a user's input, such as a question about business locations or a bill payment. You define an intent for each type of user request you want your application to support. In the tool, the name of an intent is always prefixed with the # character. To train the workspace to recognize your intents, you supply lots of examples of user input and indicate which intents they map to. **Entities:** An *entity* represents a term or object that is relevant to your intents and that provides a specific context for an intent. For example, an entity might represent a city where the user wants to find a business location, or the amount of a bill payment. In the tool, the name of an entity is always prefixed with the @ character. To train the workspace to recognize your entities, you list the possible values for each entity and synonyms that users might enter. **Dialog:** A *dialog* is a branching conversation flow that defines how your application responds when it recognizes the defined intents and entities. You use the dialog builder in the tool to create conversations with users, providing responses based on the intents and entities that you recognize in their input.
4	**<u>Prerequisites</u>** You must have an [IBM Cloud account:](#) IBM.com/Cloud 

Module 1: Pre-Work Summary

Having completed this lab, you now understand Watson Conversation Service (WCS)'s architecture, purpose and terminology. You also have an active IBM Cloud account.

Module 2: Watson Conversation Service

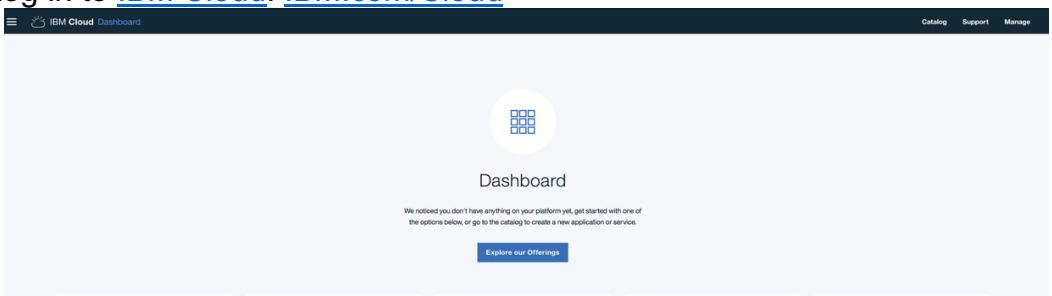
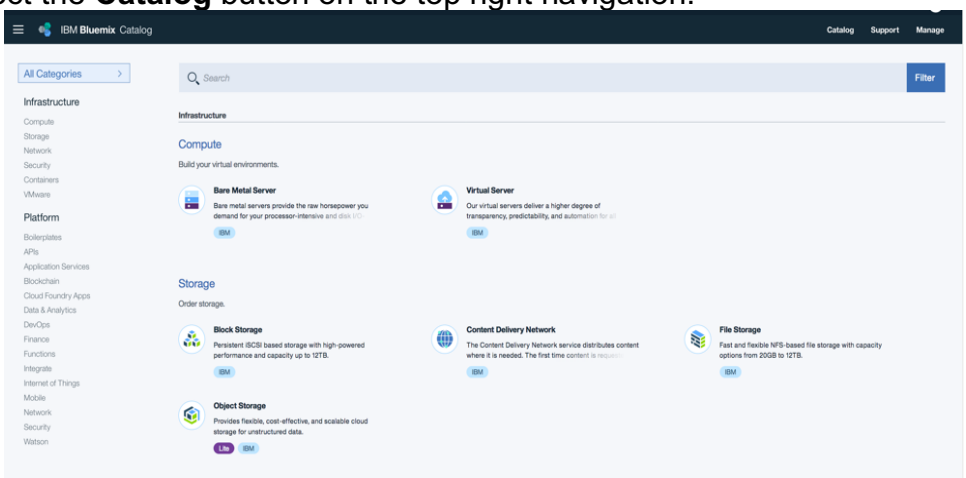
Purpose:	This lab introduces the subject of Watson Conversation Service (WCS). After completing the lab, you should be able to: • Build your own WCS

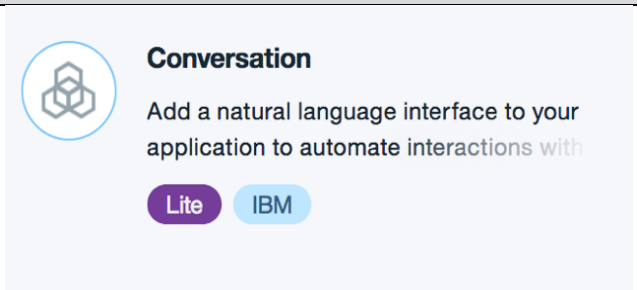
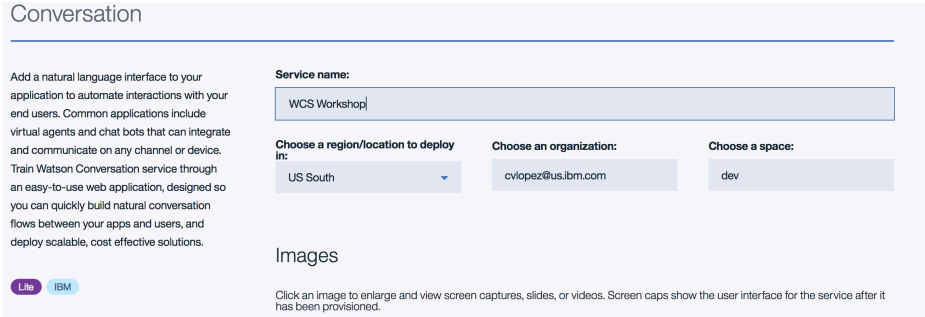
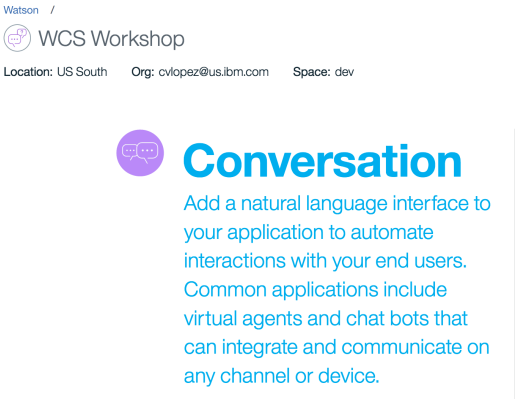
Tasks:	Tasks you will complete in this lab exercise include: • Provision a WCS instance • Configure a WCS workspace • Define Intents and Entities • Build a Dialog

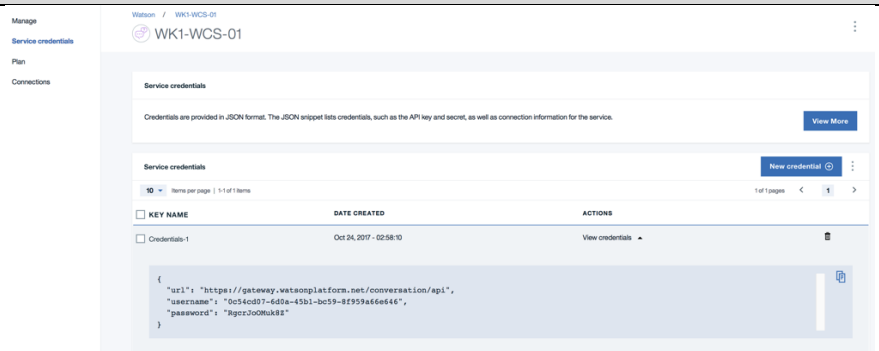
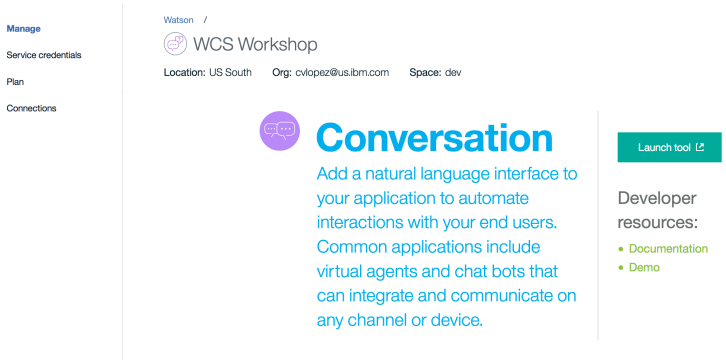
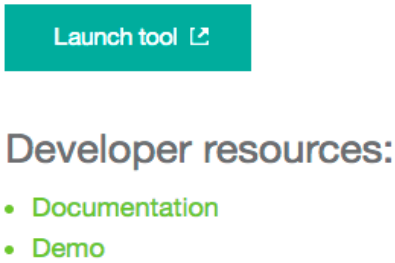
Module 2: Lab Workflow Overview

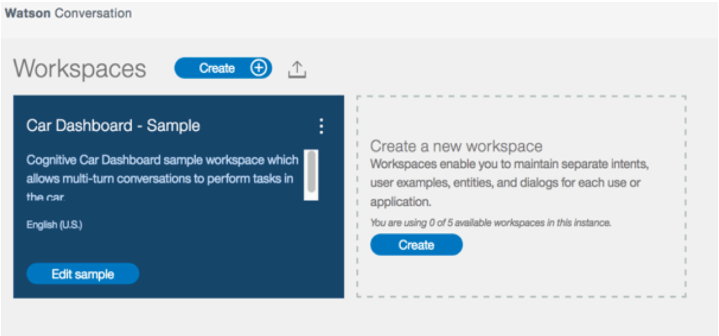
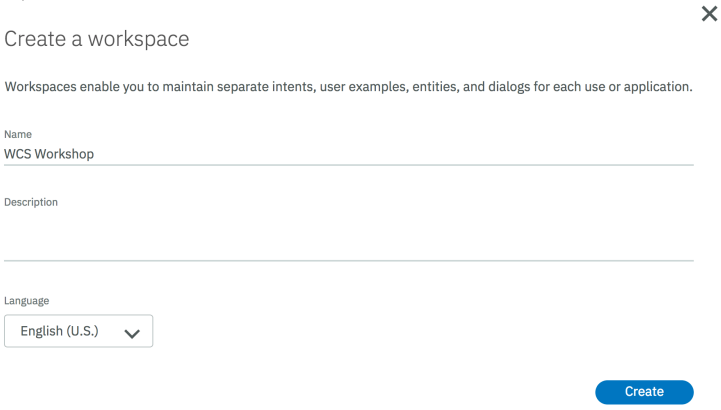
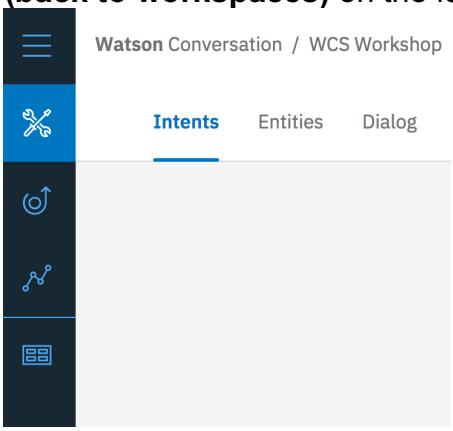
1. Create Watson Conversation Service
2. Launch WCS Tooling
3. Create Intents
4. Build a Dialog
5. Add Advanced Intents
6. Maintain Entities
7. Create Complex Dialog

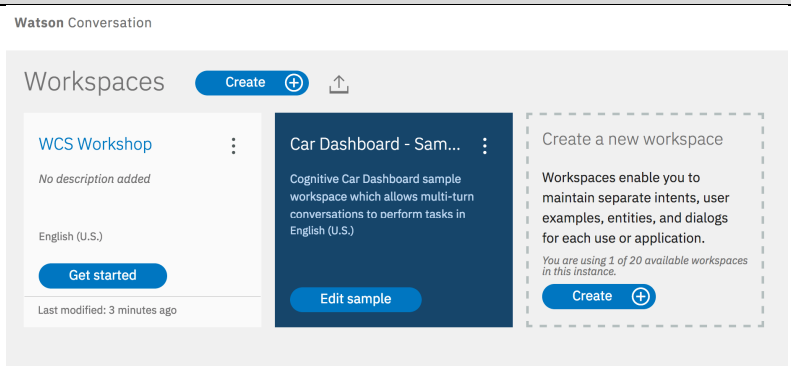
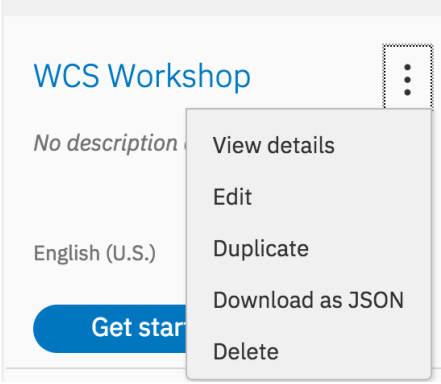
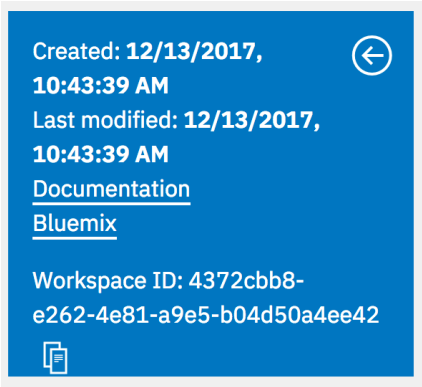
Module 2: Lab Instructions

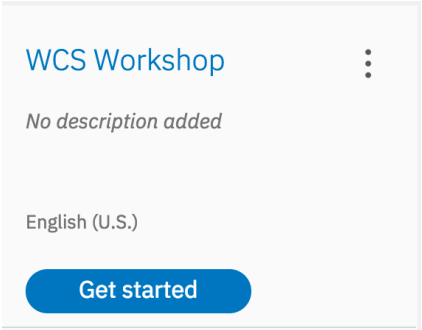
Step	Action
1	**<u>Create Watson Conversation Service</u>** 1. Log in to [IBM Cloud](): [IBM.com/Cloud]()  2. Select the **Catalog** button on the top right navigation.  3. Select **Watson**  on the top left navigation, under the Platform menu.  4. Select **Conversation.**

Step	Action
	 5. Type the project name (e.g., **WCS Workshop)**, an add a meaningful description.  6. Click create in the lower right corner. You should see the following:  7. You will need to cut and paste your service credentials. Click on **Service Credentials** on the left navigation, under "Manage". Once the page has loaded, click on **View Credentials** on the right side of the screen. A drop down will show with your specific credentials.

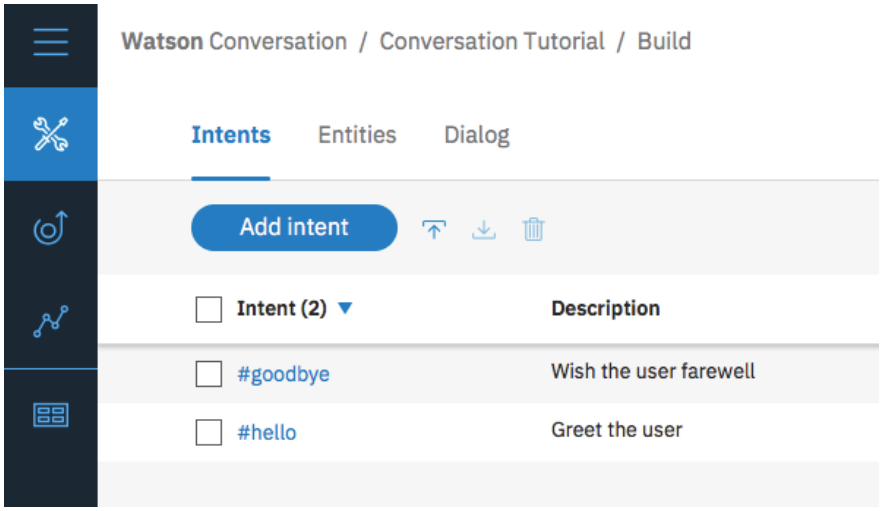
Step	Action
	 Your values will be different than shown. 8. Click the **copy icon** to copy the values to your clipboard. Paste the values in a text file for later use. 9. Go back to the Manage page, by clicking on the **Manage** link on the left side navigation.  You have now completed creating a WCS instance.
2	**<u>Launch WCS Tooling</u>** 1. Click on the **Launch tool** icon to take you to the WCS workspace  2. This will open another browser window, displaying the Watson Conversation

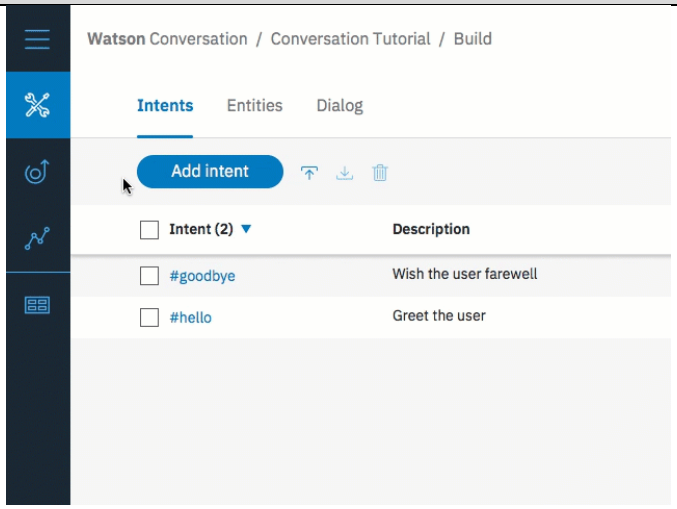
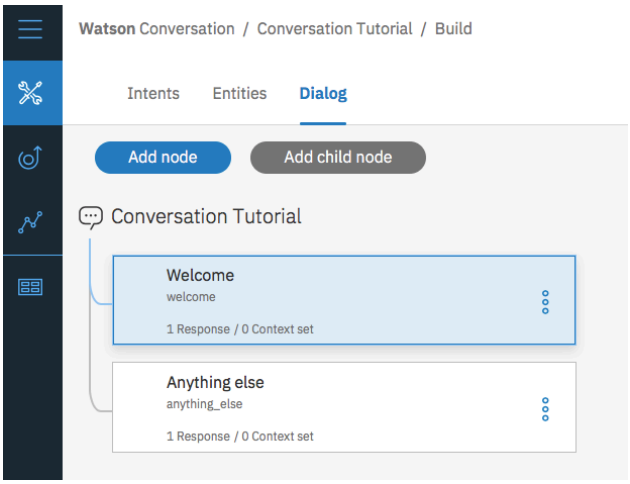
Step	Action
	Workspace.  3. Click on the **Create a new workspace** tile. 4. Enter a **workspace name** (e.g., WCS Workshop), add a meaningful description, and select **Create.**  5. Click the **last icon (back to workspaces)** on the left navigation.  6. You should now see your newly created workspace listed.

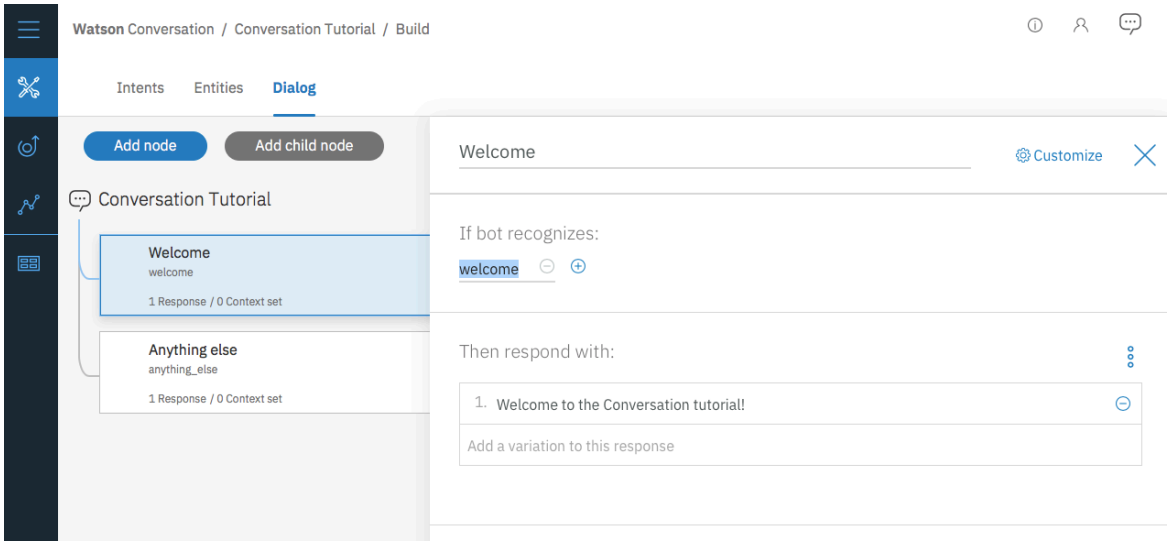
Step	Action
	 7. Click on the **three grey buttons** in the upper right corner of your workspace tile.  8. Click **view details** to find your creation date, documentation and Workspace ID.  9. Copy the **Workspace ID** and paste it in a text file. **You will need this later in the workshop.**

Step	Action
	10. Click on the **white back arrow** in the upper corner of the tile, and then **Get Started.** 

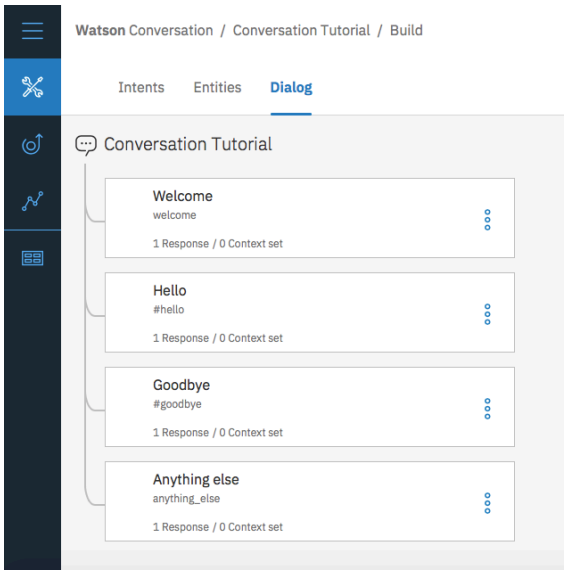
3	**<u>Create Intents</u>**
	
	An **intent** represents the purpose of a user's input. You can think of intents as the actions your users might want to perform with your application.
	
	For this example, we're going to keep things simple and define only two intents: one for saying hello, and one for saying goodbye.
	
	1. Make sure you're on the Intents tab. (You should already be there, if you just created the workspace.)
	
	2. Click **Add intent**.
	
	3. Name the intent hello, and then click **Create intent**.
	
	4. Type hello into the **Add user example** field, and then press **Enter**.
	
	Examples tell the Conversation service what kinds of user input you want to match to the intent. The more examples you provide, the more accurate the service can be at recognizing user intents.
	
	5. Add four more examples:
	
	  ○ good morning
	  ○ greetings
	  ○ hi
	  ○ howdy
	
	6. Click the **Close**  icon to finish creating the #hello intent.

Step	Action
	7. Create another intent named #goodbye with these five examples: ○ bye ○ farewell ○ goodbye ○ I'm done ○ see you later You've created two intents, #hello and #goodbye, and provided example user input to train Watson to recognize these intents in your users' input. 
4	**<u>Build a Dialog</u>** A **dialog** defines the flow of your conversation in the form of a logic tree. Each node of the tree has a condition that triggers it, based on user input. We'll create a simple dialog that handles our #hello and #goodbye intents, each with a single node. **Adding a start node**

Step	Action
	 1. In the Conversation tool, click the **Dialog** tab. 2. Click **Create**. You'll see two nodes: ○ **Welcome**: Contains a greeting that is displayed to your users when they first engage with the bot. ○ **Anything else**: Contains phrases that are used to reply to users when their input is not recognized.  3. Click the **Welcome** node to open it in the edit view.

Step	Action
	4. Replace the default response with the text, Welcome to the Conversation tutorial!.  5. Click  to close the edit view. You created a dialog node that is triggered by the welcome condition, which is a special condition that indicates that the user has started a new conversation. Your node specifies that when a new conversation stats, the system should respond with the welcome message. **Testing the start node** You can test your dialog at any time to verify the dialog. Let's test it now. Click the  icon to open the *Try it out* pane. You should see your welcome message.

Step	Action
	 Click to close the *Try it out* view. **Adding nodes to handle intents** Now let's add nodes to handle our intents between the Welcome node and the Anything else node. 1. Click the More icon on the **Welcome** node, and then select **Add node below**. 2. Give the node a name and type #hello in the **Enter a condition** field of this node. Then select the **#hello** option. 3. Add the response, Good day to you. 4. Click to close the edit view.

Step	Action
	5. Click the More icon on this node, and then select **Add node below** to create a peer node. In the peer node, specify #goodbye as the condition, and OK. See you later! as the response. 6. Click to close the edit view.  **Testing intent recognition** You built a simple dialog to recognize and respond to both hello and goodbye inputs. Let's see how well it works. 1. Click the icon to open the *Try it out* pane. There's that reassuring welcome message. 2. At the bottom of the pane, type Hello and press Enter. The output indicates that the #hello intent was recognized, and the appropriate response (Good day to you.) appears. 3. Try the following input: ○ bye ○ howdy ○ see ya ○ good morning ○ sayonara

Step	Action
	Watson can recognize your intents even when your input doesn't exactly match the examples you included. The dialog uses intents to identify the purpose of the user's input regardless of the precise wording used, and then responds in the way you specify.

5	**<u>Add Advanced Intents</u>**
	
	1. Go back to the Intents page and click **Add intent**.
	
	2. Add the following intent name, and then click **Create intent**. Create the intent #turn_on. The #turn_on intent indicates that the user wants to turn on an appliance such as the radio, windshield wipers, or headlights.
	
	3. In the **Add user example** field, type the following utterance, and then click **Add example**. Enter the following examples:
	  ○ I need
	  ○ Play
	  ○ Play some
	  ○ Start
	  ○ turn on
	  ○ Crank up
	
	4. Click the **Close**  icon to finish adding the #turn_on intent.
	
	You now have three intents, the #turn_on intent that you just added, and the #hello and #goodbye. Each intent has a set of example utterances that help train Watson to recognize the intents in user input.

6	**<u>Maintain Entities</u>** An **entity** definition includes a set of entity *values* that can be used to trigger different responses. Each entity value can have multiple *synonyms*, which define different ways that the same value might be specified in user input. Create entities that might occur in user input that has the #turn_on intent to represent what the user wants to turn on. 1. Click the **Entities** tab to open the Entities page. 2. Click **Add entity**.

Step	Action

3. Add the @appliance entity name, and then press Enter. The @appliance entity represents an appliance in the car that a user might want to turn on.

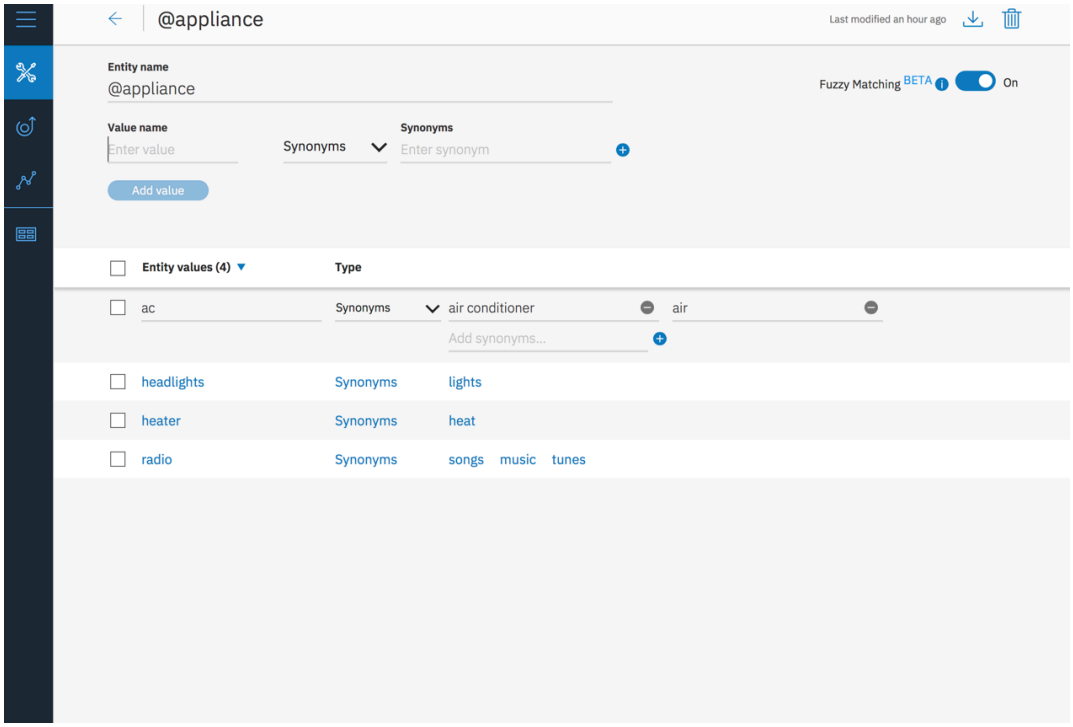
4. Add the following entities:

Entity value	Type	Values
radio	Synonym	music, tunes, songs
ac	Synonym	air, air conditioner
heater	Synonym	heat
headlight	Synonym	lights, headlamps

5. Click the toggle to turn fuzzy matching **On** for the @appliance entity. This setting helps the service recognize references to entities in user input even when the entity is specified in a way that does not exactly match the syntax you use here.

6. Click the **Close** icon to finish adding the @appliance entity.

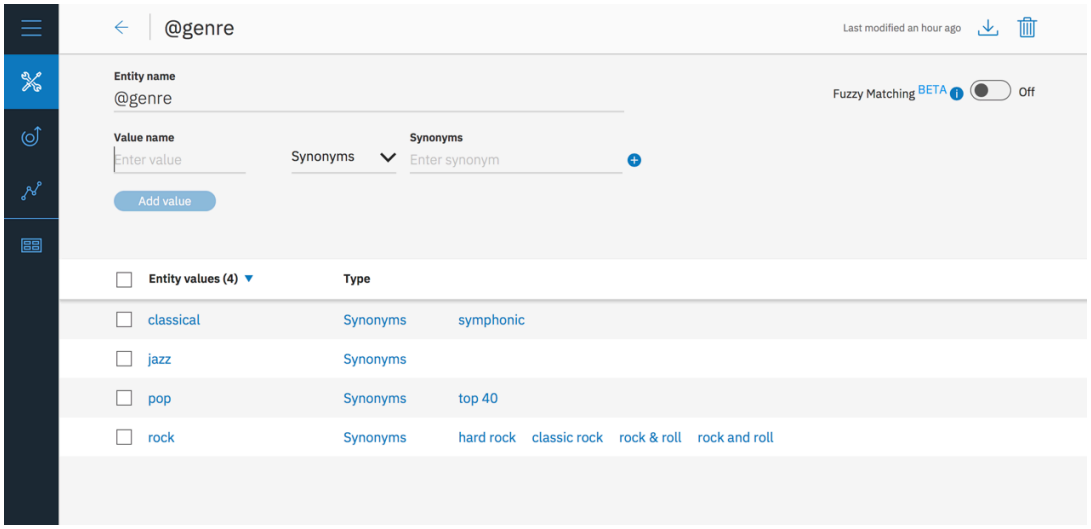
It should look like this:



Step	Action
	7. Repeat Steps 2-6 to create the @genre entity with fuzzy matching on, and these values and synonyms:

Entity Value	Type	Values
classical	Synonym	symphonic
jazz	Synonym	
pop	Synonym	top 40
rock	Synonym	rock & roll, rock and roll, hard rock

It should look like this:



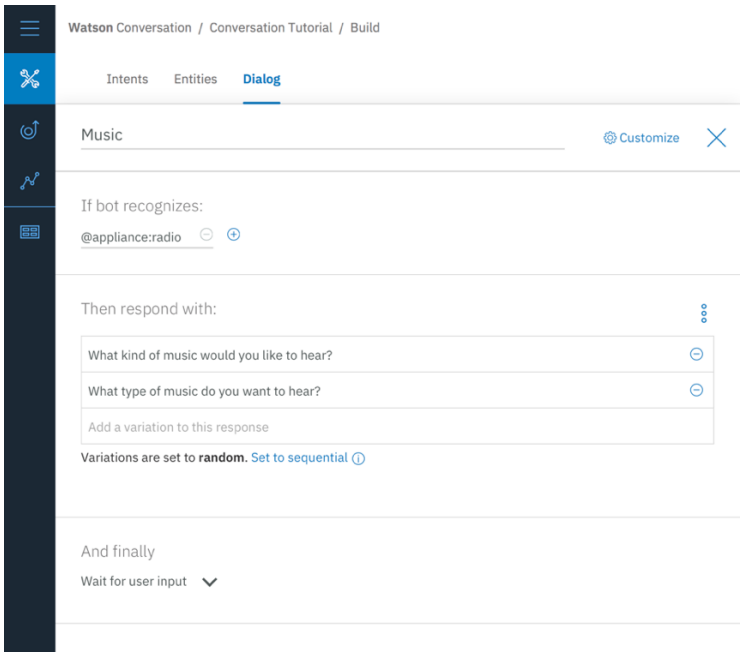
You defined two entities: @appliance (representing an appliance the bot can turn on) and @genre (representing a genre of music the user can choose to listen to).

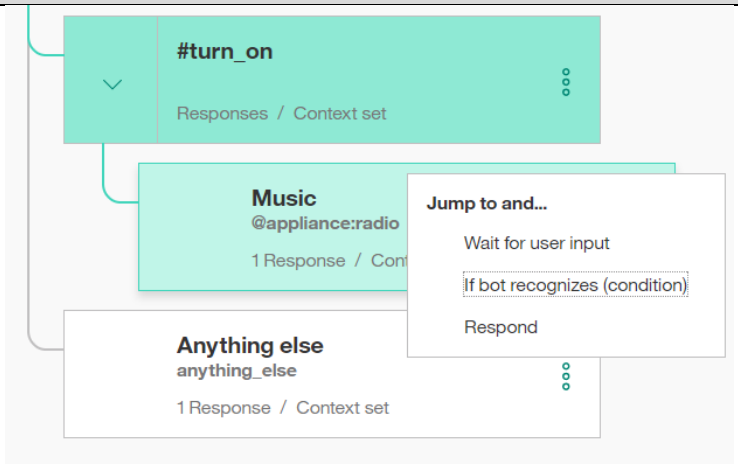
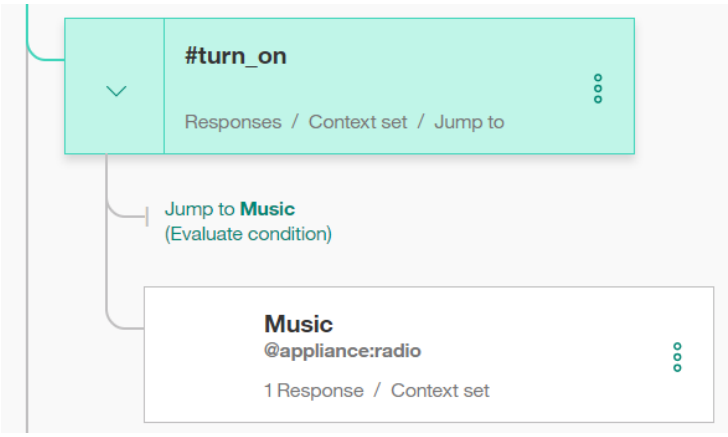
When the user's input is received, the Conversation service identifies both the intents and entities. You can now define a dialog that uses intents and entities to choose the correct response

.

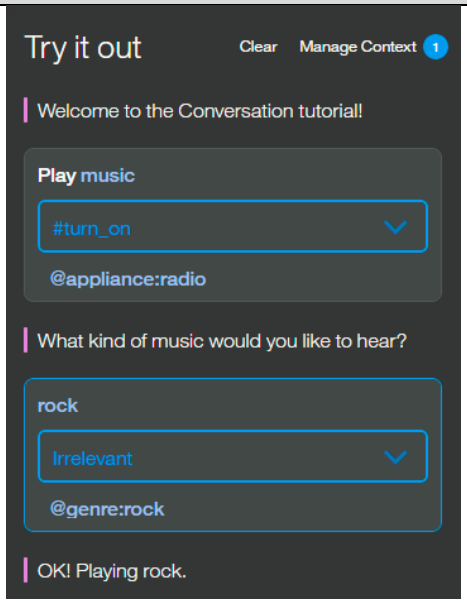
7	**<u>Create Complex Dialog</u>** In this complex dialog, you will create dialog branches that handle the #turn_on intent you defined earlier. **Add a base node for #turn_on**

Step	Action
	Create a dialog branch to respond to the #turn_on intent. Start by creating the base node: 1. Click the More icon on the **#hello** node, and then select **Add node below**. 2. Start typing #turn_on in the condition field, and then select it from the list. This condition is triggered by any input that matches the #turn_on intent. 3. Do not enter a response in this node. Click to close the node edit view. **Scenarios** The dialog needs to determine which appliance the user wants to turn on. To handle this, create multiple responses based on additional conditions. There are three possible scenarios, based on the intents and entities that you defined: ○ **Scenario 1**: The user wants to turn on the music, in which case the bot must ask for the genre. ○ **Scenario 2**: The user wants to turn on any other valid appliance, in which case the bot echos the name of the requested appliance in a message that indicates it is being turned on. ○ **Scenario 3**: The user does not specify a recognizable appliance name, in which case the bot must ask for clarification. Add nodes that check these scenario conditions in this order so the dialog evaluates the most specific condition first. **Address Scenario 1** Add nodes that address scenario 1, which is that the user wants to turn on the music. In response, the bot must ask for the music genre. Add a child node that checks whether the appliance type is music 1. Click the More icon on the **#turn_on** node, and select **Add child node**. 2. In the condition field, enter @appliance:radio. This condition is true if the value of the @appliance entity is radio or one of its synonyms, as defined on the Entities tab.

Step	Action
	3. In the response field, enter What kind of music would you like to hear? and add a second response of What type of music do you want to hear? 4. Set the variation to Random by clicking on the *Set to Random* link. 5. Name the node Music. 6. Click to close the node edit view. Your dialog for Music should look like this:  **Add a jump from the #turn_on node to the Music node** Jump directly from the #turn on node to the Music node without asking for any more user input. To do this, you can use a **Jump to** action. 1. Click the More icon on the **#turn_on** node, and select **Jump to**. 2. Select the **Music** child node, and then select **If bot recognizes (condition)** to indicate that you want to process the condition of the Music node.

Step	Action
	 Note that you had to create the target node (the node to which you want to jump) before you added the **Jump to** action. After you create the Jump to relationship, you see a new entry in the tree:  **Add a child node that checks the music genre** Now add a node to process the type of music that the user requests. 1. Click the More icon ⋮ on the **Music** node, and select **Add child node**. This child node is evaluated only after the user has responded to the question about the type of music they want to hear. Because we need a user input before this node, there is no need to use a **Jump to** action.

Step	Action
	2. Add @genre to the condition field. This condition is true whenever a valid value for the @genre entity is detected. 3. Enter OK! Playing @genre. as the response. This response reiterates the genre value that the user provides. **Add a node that handles unrecognized genre types in user responses** Add a node to respond when the user does not specify a recognized value for @genre. 1. Click the More icon on the *@genre* node, and select **Add node below** to create a peer node. 2. Enter true in the condition field. The true condition is a special condition. It specifies that if the dialog flow reaches this node, it should always evaluate as true. (If the user specifies a valid @genre value, this node will never be reached.) 3. Enter I'm sorry, I don't understand. I can play classical, rhythm and blues, or rock music. as the response. That takes care of all the cases where the user asks to turn on the music. **Test the dialog for music** 1. Select the icon to open the chat pane. 2. Type Play music. The bot recognizes the #turn_on intent and the @appliance:music entity, and it responds by asking for a musical genre. 3. Type a valid @genre value (for example, rock). The bot recognizes the @genre entity and responds appropriately.

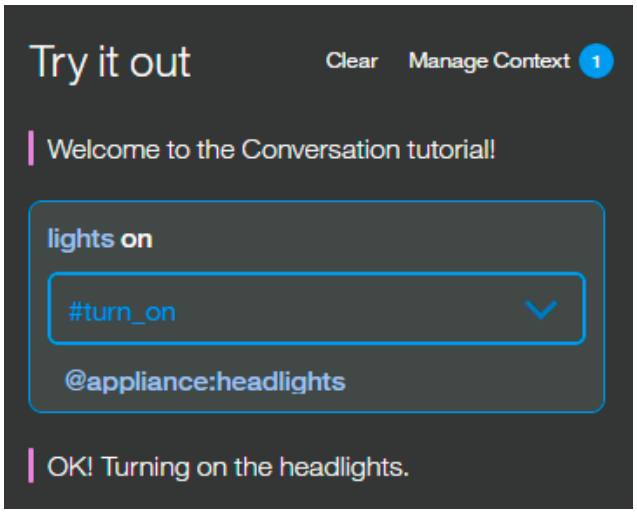
Step	Action
	 4. Type Play music again, but this time specify an invalid response for the genre. The bot responds that it does not understand. **Address Scenario 2** We will add nodes that address scenario 2, which is that the user wants to turn on another valid appliance. In this case, the bot echos the name of the requested appliance in a message that indicates it is being turned on. Add a child node that checks for any appliance Add a node that is triggered when any other valid value for @appliance is provided by the user. For the other values of @appliance, the bot doesn't need to ask for any more input. It just returns a positive response. 1. Click the More icon on the **Music** node, and then select **Add node below** to create a peer node that is evaluated after the @appliance:music condition is evaluated. 2. Enter @appliance as the node condition. This condition is triggered if the user input includes any recognized value for the @appliance entity besides music. 3. Enter OK! Turning on the @appliance. as the response. This response reiterates the appliance value that the user provided.

Test the dialog with other appliances

1. Select the icon to open the chat pane.

2. Type lights on.

The bot recognizes the #turn_on intent and the @appliance:headlights entity, and it responds with OK, turning on the headlights.



3. Type turn on the air.

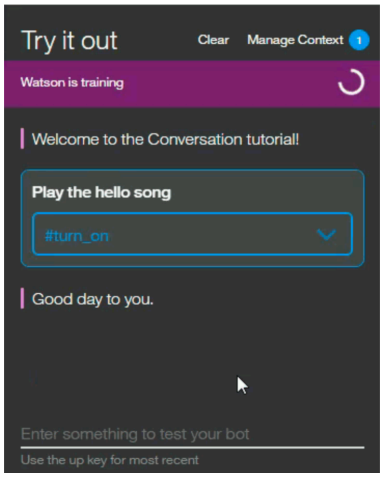
4. The bot recognizes the #turn_on intent and the @appliance:(air conditioning) entity, and it responds with OK, turning on the air conditioning.

5. Try variations on all of the supported commands based on the example utterances and entity synonyms you defined.

Address Scenario 3
Now add a peer node that is triggered if the user does not specify a valid appliance type.

1. Click the More icon on the **@appliance** node, and then select **Add node below** to create a peer node that is evaluated after the @appliance condition is evaluated.

2. Enter true in the condition field. (If the user specifies a valid @appliance value, this node will never be reached.)

Step	Action
	3. Enter I'm sorry, I'm not sure I understood you. I can turn on music, headlights, or air conditioning. as the response. **Test some more** 1. Try more utterance variations to test the dialog. If the bot fails to recognize the correct intent, you can retrain it directly from the chat window. Select the arrow next to the incorrect intent and choose the correct one from the list.  Optionally, you can review the **Car Dashboard - Sample** workspace to see this same use case fleshed out even more with a longer dialog and additional functionality. 1. Click the **Back to workspaces** button  from the navigation menu. 2. On the **Car Dashboard - Sample** tile, click **Edit sample**.

Module 2: Lab Summary

In this portion of the lab, Watson Conversation Services were explored and utilized. We began by creating the actual Watson Conversation Service within IBM Cloud. Next, the foundations for conversations were added starting with intents and dialogs. To enhance the conversation capabilities, entities were added which allow for synonyms of words to be picked up through the Watson Conversation Service. The lab concludes by creating and testing enhanced dialog capabilities.

Module 3: Deploy a Node-RED Web App

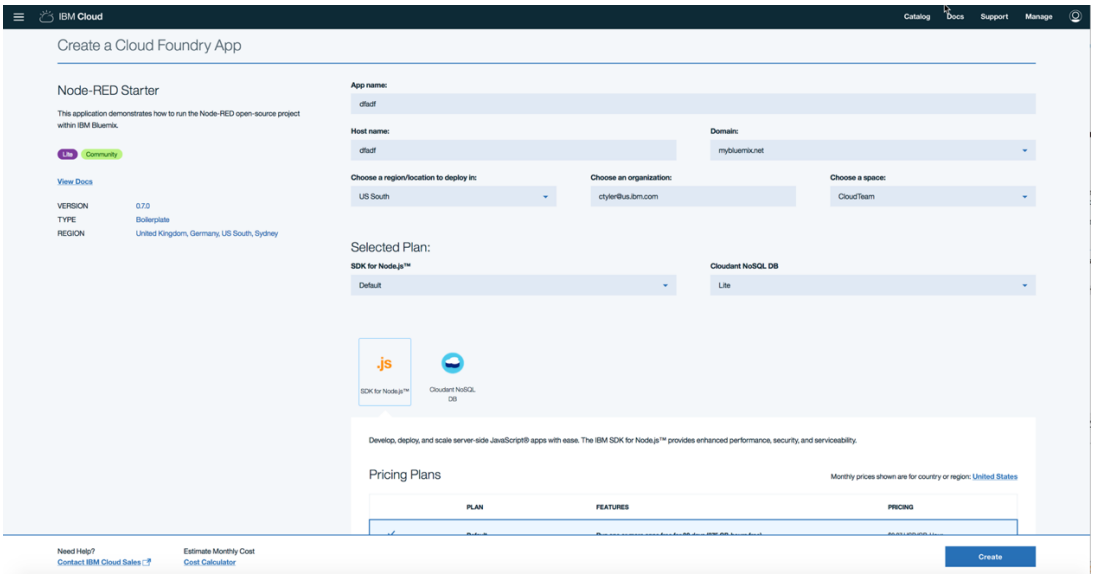
Purpose:	This lab introduces the subject of using Node-RED to deploy the Watson Conversation Services. After completing the lab, you should be able to: • Create a Node-RED instance • Understand how Node-RED flows work • Deploy Watson Conversation Services with Node-RED

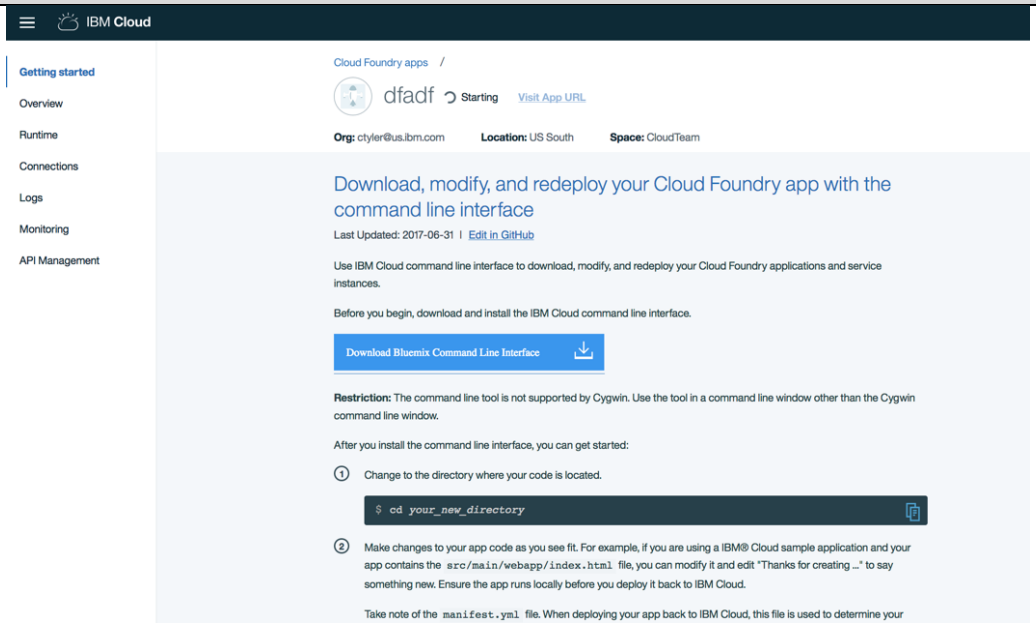
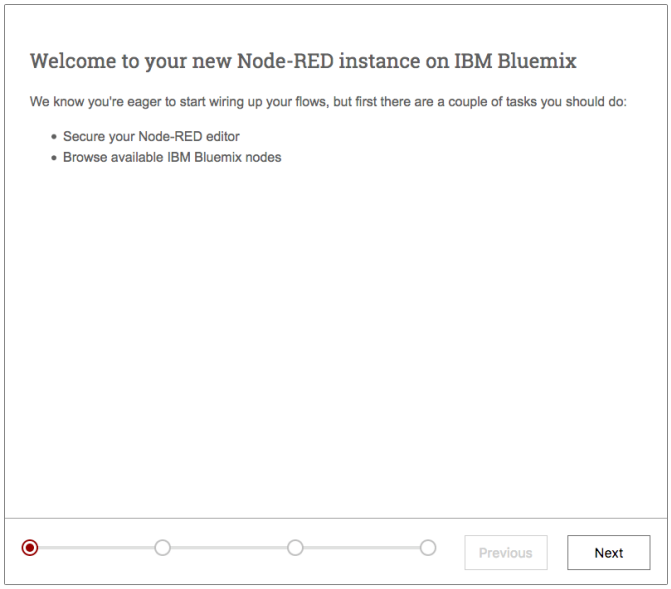
Tasks:	Tasks you will complete in this lab exercise include: • Create a Node-RED instance • Import flows into Node-RED • Edit a flow to add Watson Conversation Services • Deploy the Node-RED application

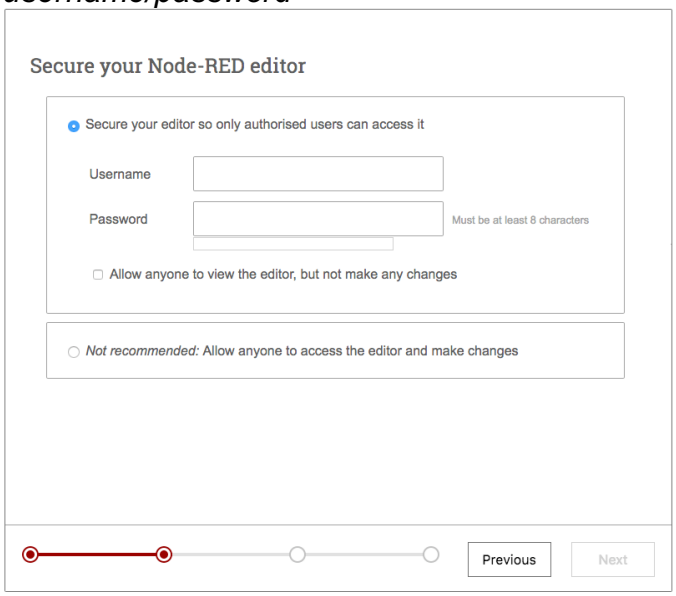
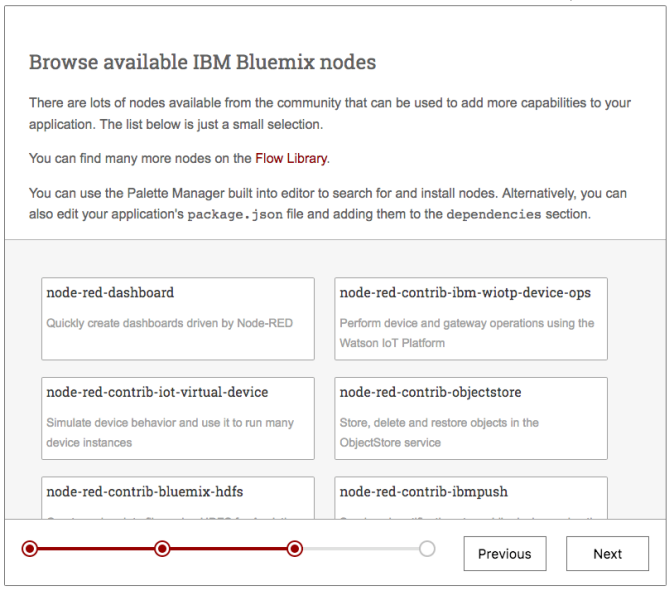
Module 3: Lab Workflow Overview

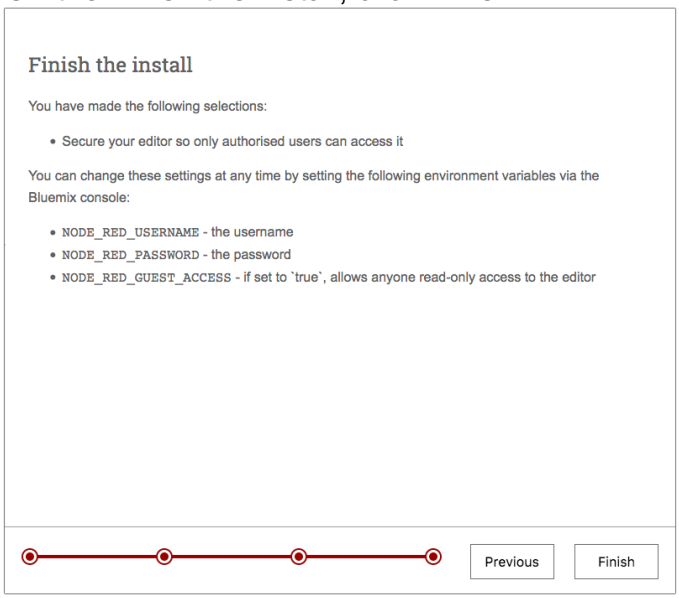
1. Create Node-RED Instance
2. Configure Node-RED App
3. Integrate Watson Conversation Services
4. Deploy Node-RED Application

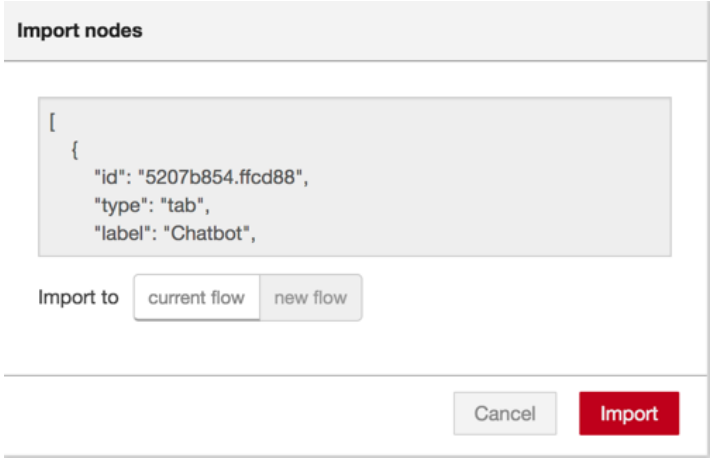
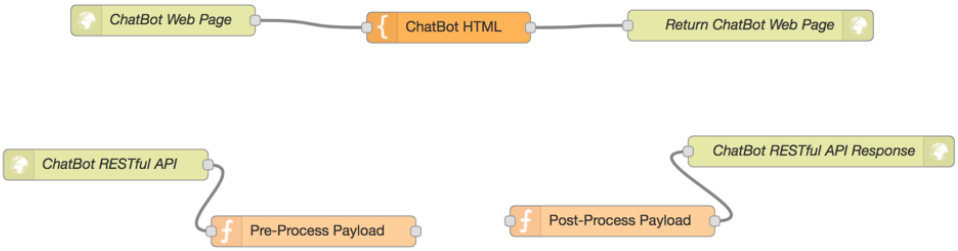
Module 3: Lab Instructions

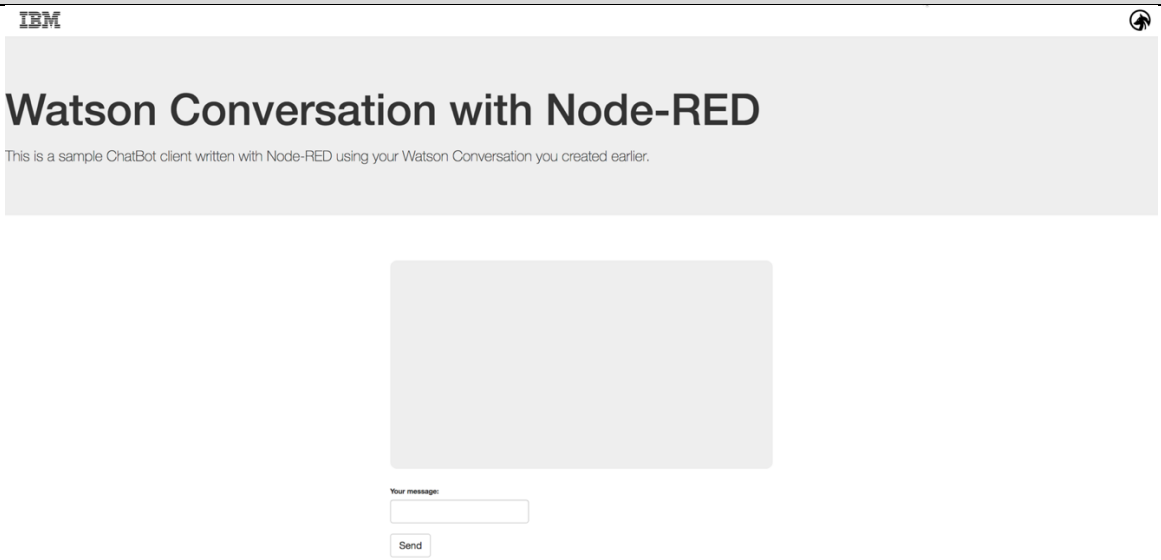
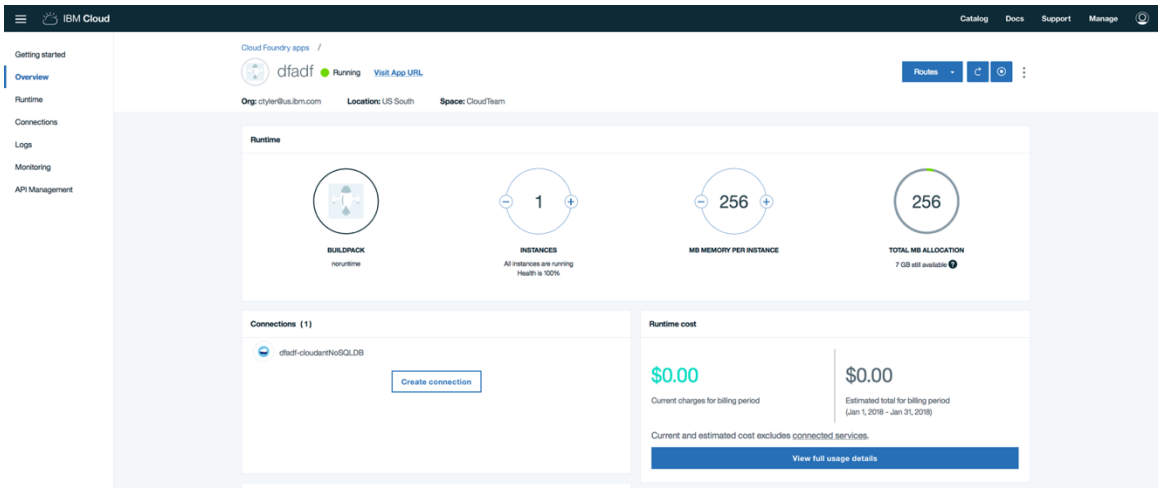
Step	Action
1	**<u>Create Node-RED Instance</u>** Now that you have built and tested your workspace, you can deploy it by connecting it to a user interface. Click the link below. 1. Login to IBM Cloud (https://bluemix.net/) 2. On the Dashboard, click the Create Resource button in the top left. 3. On the search page, search for Node-RED and click on the Node-RED Starter. 4. Give the application a name, choose the organization and region and click create. This will take a minute or two to create, deploy and start the Node-RED instance.  5. You will land on the Getting Started page for the Node-RED Starter instance and you will see the Starting icon rotating. Wait for the application to change to green and show Running, then click the Visit App URL link.

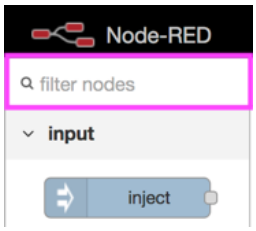
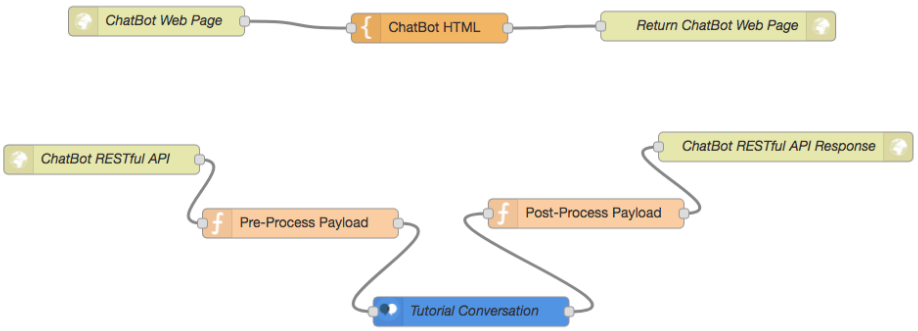
Step	Action
	
2	**<u>Configure Node-RED App</u>** 1. On the App URL page, you will walk through a series of steps to configure your Node-RED instance. - On the Welcome step, Click Next 

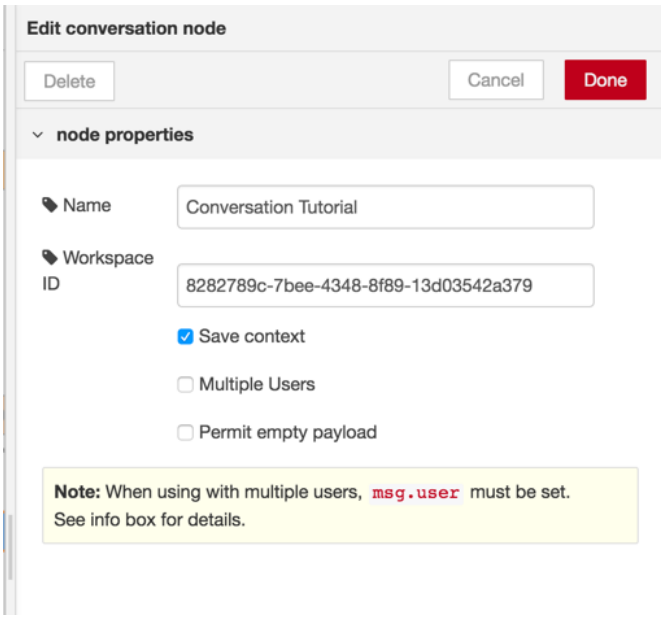
Step	Action
	○ Enter a username/password to secure your Node-RED Flow Editor (recommended) and do not check Allow anyone to view the editor or Allow anyone to access the editor then click Next. *Note: Remember this username/password*  ○ On the Browse available Bluemix nodes, click Next 

Step	Action
	○ On the Finish the install, click Finish  2. After the configuration has been applied, click on the **Go to your Node-RED Flow Editor** Button 3. Login with the username/password you just entered on the configuration. This will take you to a blank flow editor canvas.  4. In the GitHub repository, copy the contents of the Node-RED-Flow Starter.json file to the clipboard.

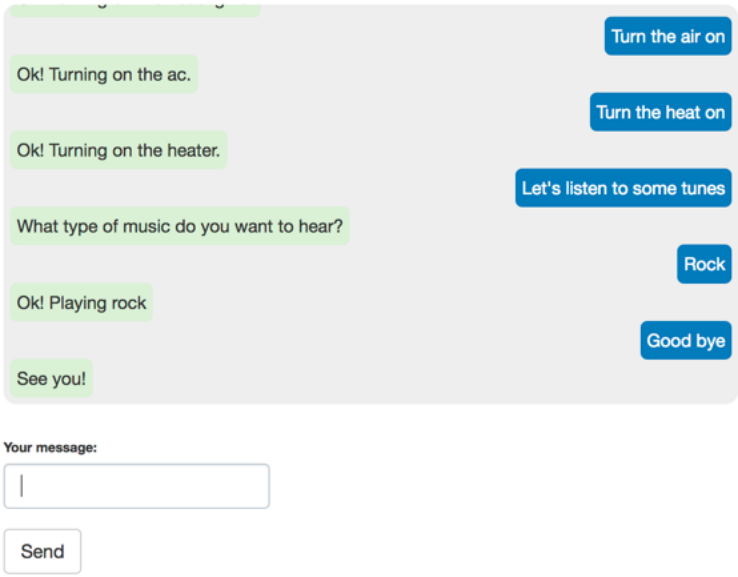
Step	Action
	5. Click the hamburger menu in the top right, and select Import -> Clipboard and then paste the contents of the clipboard into the editor window. Choose to import to a new flow and click Import.  6. You should see two flows. You can delete the blank flow by clicking its tab and using the hamburger menu and choosing Flows -> Delete 7. On the flow that you just imported, called Chatbot, you should see a flow that looks like the following.  8. If you click Deploy, in the top right, it will create a new web page for you which is your https://<*Node-RED domain*>.mybluemix.net/bot. **Note:** *Change the <Node-RED domain> to be yours*. 9. Open a new browser tab and go to your web app url. You should see a page that looks like this.

Step	Action
	
3	**<u>Integrate Watson Conversation Services</u>** 1. Within the Node-RED flow, we are missing the connection to the Conversation service, so let's add it in. 2. First, go to the Node-RED Starter service Overview page which looks like this.  3. In the Connections section, click on the Create Connection button. Find the Conversation Service you created earlier and click the connect button on it. 4. This will cause your Node-RED application to be Re-Staged. Allow it to be re-

Step	Action
	staged. This will take a few minutes for the process to stop, re-stage, and re-start your application. Wait for the status of your Node-RED application to show Running. 5. Go back to your Node-RED Flow editor (**Note:** *you may need to re-login*). 6. Now, we need to add the Conversation connection in our Flow. 7. In the top left of the Flow Editor, you can search for Nodes. Enter Conversation in the search box and you'll see the Nodes filtered down to just those with Conversation in the name. Click and drag the Conversation node into the canvas.  8. To connect nodes within Node-RED, you click the first dot, drag it to the second dot, and release the mouse button. Using your mouse, connect the dots on the outbound side of the Pre-Process Payload node to the inbound side of the new Conversation node. And connect the dots on the outbound side of the new Conversation Node with the inbound side of the Post-Process Payload node. Your end result will look similar to the following:  9. Double-click on the new Conversation node to edit it. Add a name and put the Workspace ID from your Conversation Workspace into the proper fields and click done. To get the Workspace ID, do the following: o Go into the Watson Conversation tool by opening the IBM Cloud dashboard (https://console.bluemix.net) o Click on the IBM Cloud hamburger menu in the top left and choose

Step	Action
	Watson ○ Click on the Conversation service you created earlier ○ Click on the Launch Tool button ○ If needed, log in with your IBM ID ○ Click on the More icon ⋮ and choose View Details ○ You can copy the Workspace ID and paste it into the Workspace ID field in Node-RED ○ Click Done 
4	**<u>Deploy Node-RED Application</u>** 1. Click the Deploy button at the top. 2. Go back to the tab with your Chatbot Web Page and refresh the page. 3. You should be able to enter some text into the Message button. Some of the messages you can send your chatbot include: • Hello • Turn the air on • Turn the heat on • Turn on the headlamps

Step	Action

	• Let's listen to some tunes
	  ○ This will ask you a follow-up question of what type of music
	  ○ You can enter genres such as Rock, Jazz, Classical
	
	 
	
	  ○ Good bye
	
	4. That concludes the workshop

Module 3: Lab Summary

Within this lab, a Node-RED application is deployed on IBM Cloud. The Node-RED instance that is deployed is then secured by specifying a username and password. Node-RED editor is launched and a base flow is imported using a file located on GitHub as the source. Next, the Watson Conversation Service that was created within previous modules is integrated into the Node-RED application by adding a conversation node and specifying the necessary parameters. The Node-RED application is then deployed and tested.

Congratulations, you have successfully deployed a Node-RED based chatbot powered by IBM Watson Conversation Services!