BUILDING A WATSON CHATBOT

Workshop: Node-RED lab guide

IBM

IBM Cloud

Notices and Disclaimers

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

This document is current as of the initial date of publication and may be changed by IBM at any time. Not all offerings are available in every country in which IBM operates.

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at ibm.com/legal/copytrade.shtml

Other company, product and service names may be trademarks or service marks of others

IBM Cloud

Table of Contents

An Introduction to Cloud and IBM Cloud Platform

Prerequisites

You must have an IBM Cloud account. If you do not have an IBM Cloud account (it is free) go to www.ibm.com/cloud and click "Sign up".

Purpose of this Lab

This lab introduces the Watson Conversation Service (WCS) along with how to deploy them using Node-RED.
After completing the lab, you should be able to:

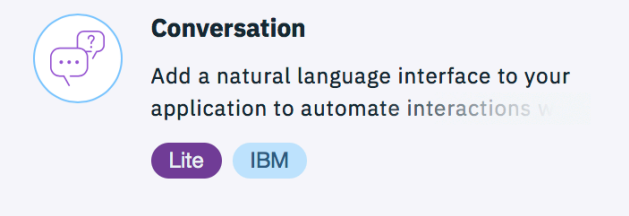
- Build your own Watson Conversation Service
- Create a Node-RED instance
- Understand how Node-RED flows work
- Deploy Watson Conversation Services with Node-RED

Tasks you will complete in this lab include:

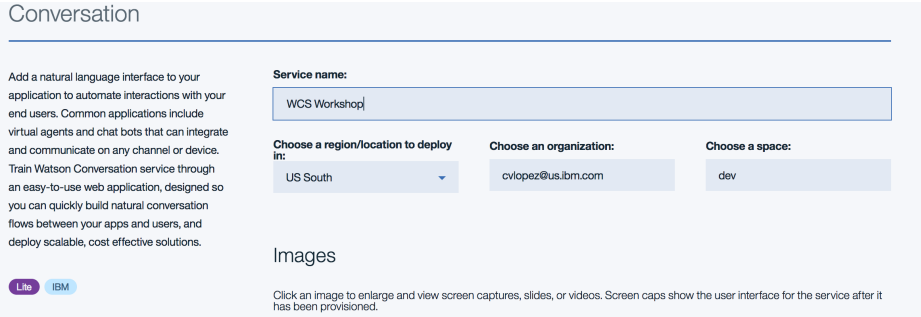
- Provision a Watson Conversation Service instance
- Configure a Watson Conversation Service workspace
- Define Intents and Entities within Watson Conversation Service
- Build a Dialog within Watson Conversation Service
- Create a Node-RED instance
- Import flows into Node-RED
- Edit a flow to add Watson Conversation Service
- Deploy the Node-RED application

Module 1: Watson Conversation Service

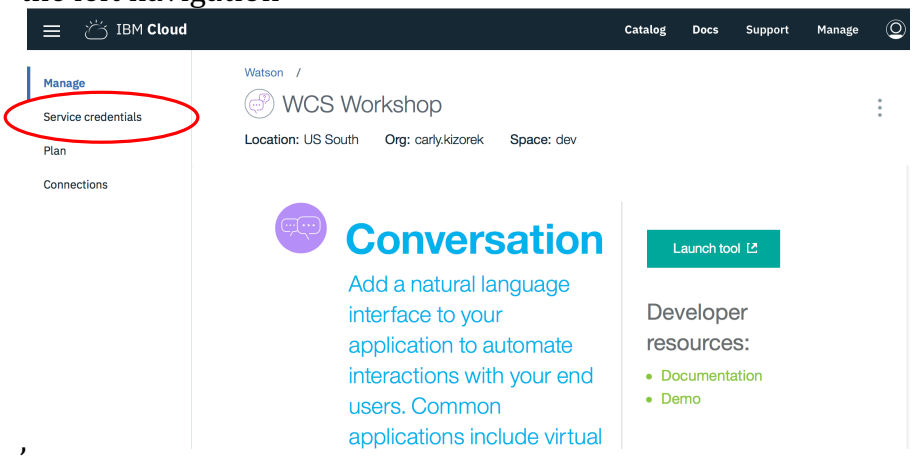
1. Log into IBM Cloud at https://console.bluemix.net/
2. Select the **Catalog** button from the navigation bar.
3. Type **Conversation** into the search bar.
4. Select **Conversation.**



5. For the Service name field, type the project name (e.g., **WCS Workshop)**.

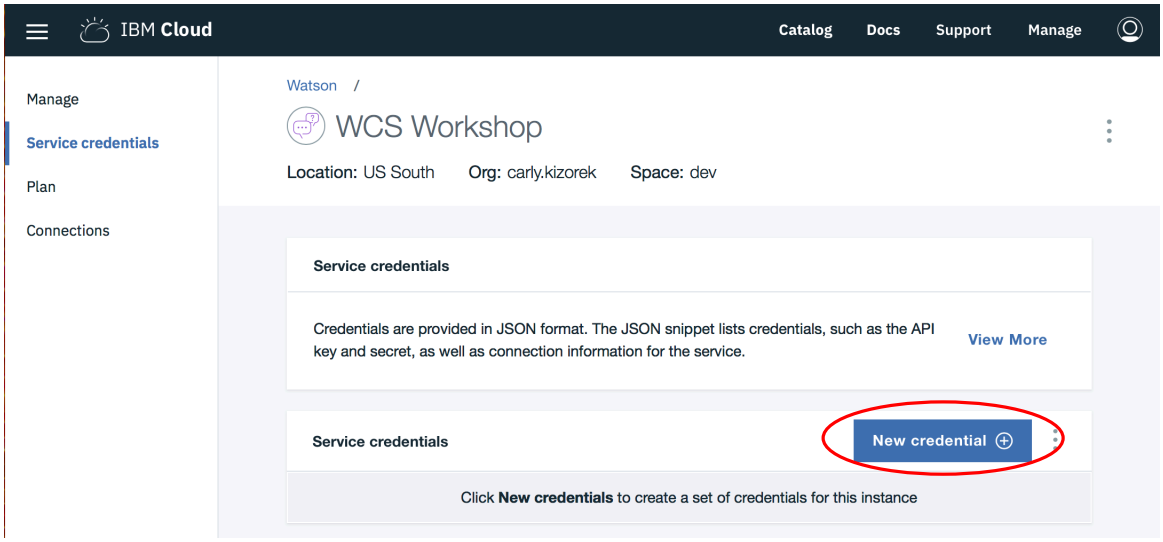


6. Click **Create** in the lower right corner.
7. You will need to add service credentials to use later. Click on **Service Credentials** on the left navigation

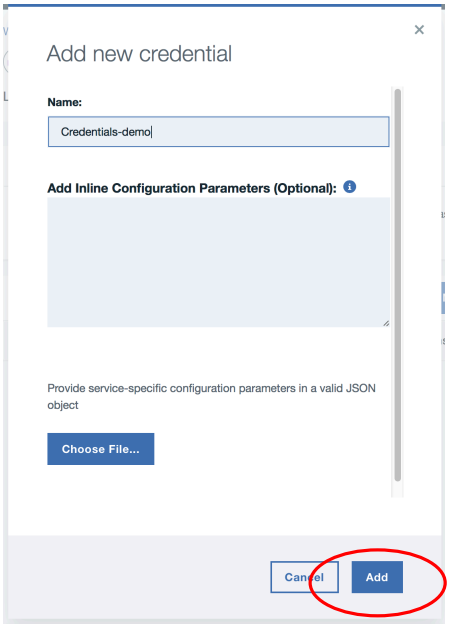


,

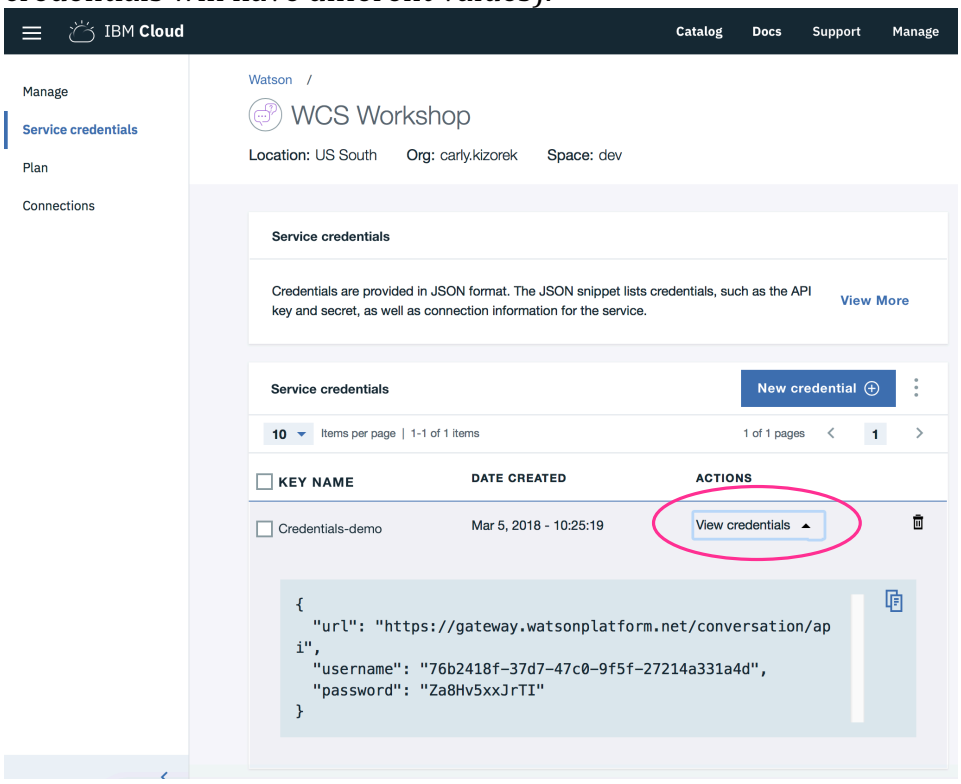
8. Click **New credential**



9. Add a meaningful name to the credentials you are creating. Then click **Add**



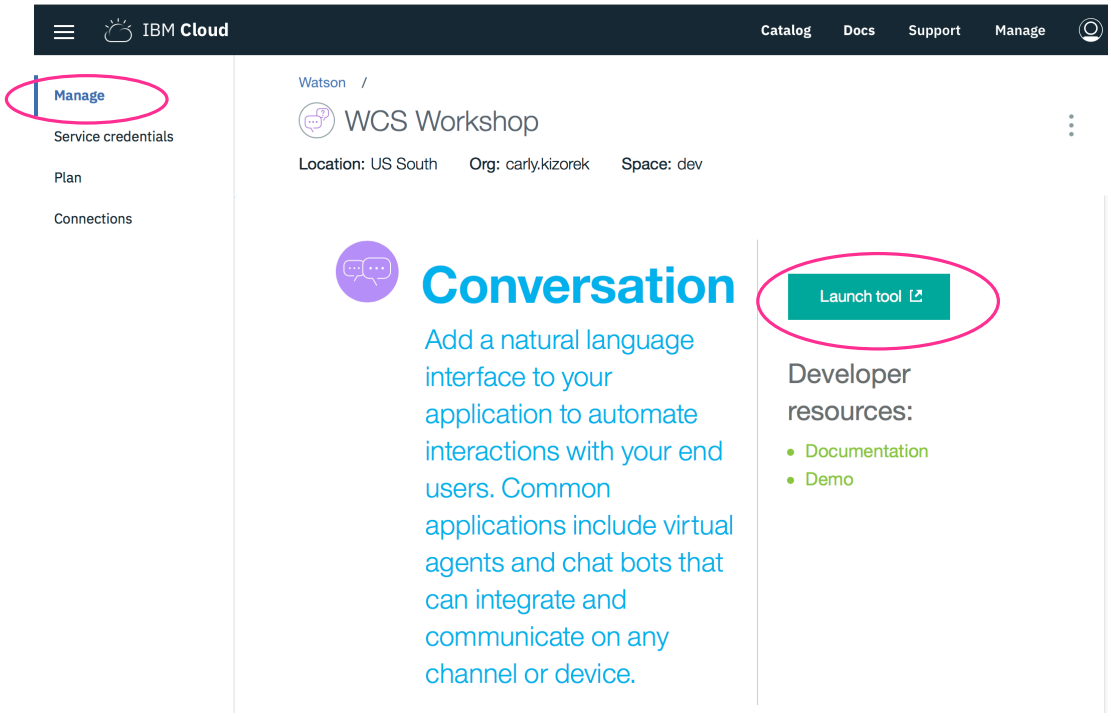
10. Now that your credentials have been added, select/expand **View credentials** (yours credentials will have different values).



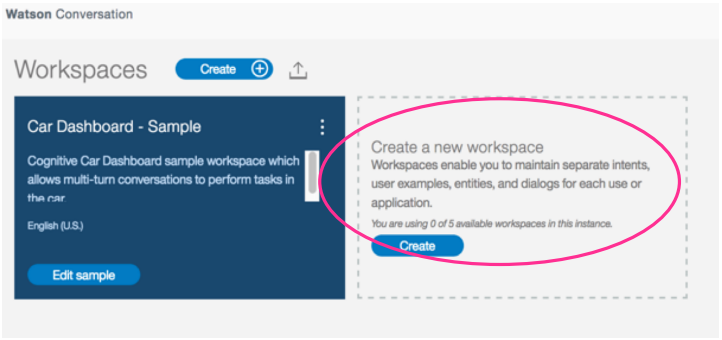
11. Copy these values (or click the **copy icon**) and paste these into a text file for later use.

Launch Watson Conversation service Tool

1. From the **Manage** page of your Watson Conversation service, click **Launch tool** located on the right

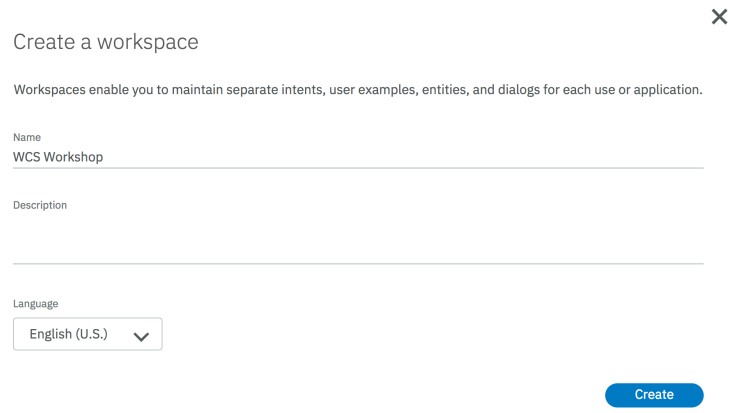


2. This will open another browser window, displaying the Watson Conversation Workspace:

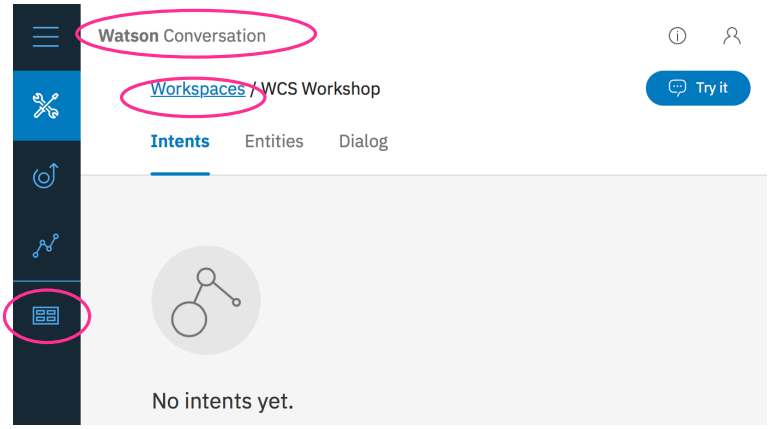


3. Click on the **Create a new workspace** tile.

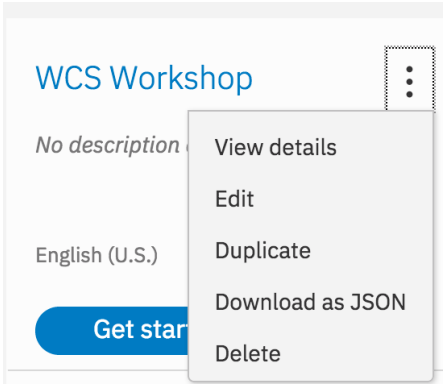
4. Enter a workspace **name** (e.g., WCS Workshop), add a meaningful description, and select **Create.**



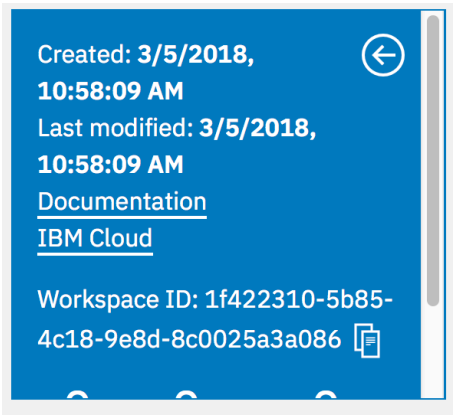
5. Return to your workspaces by clicking any of the following:



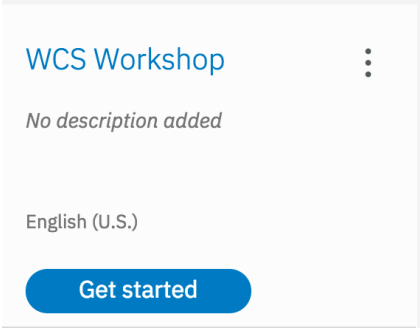
6. Click on the **More** icon ⋮ in the top right corner of your new workspace tile.



7. Click **view details** to find your creation date, documentation and Workspace ID.



8. Copy the **Workspace ID** and paste it in a text file. **You will need this later in the workshop.**
9. Click on the **white back arrow** in the upper corner of the tile, and then **Get Started.**



Create Intents

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**.

IBM Cloud

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.

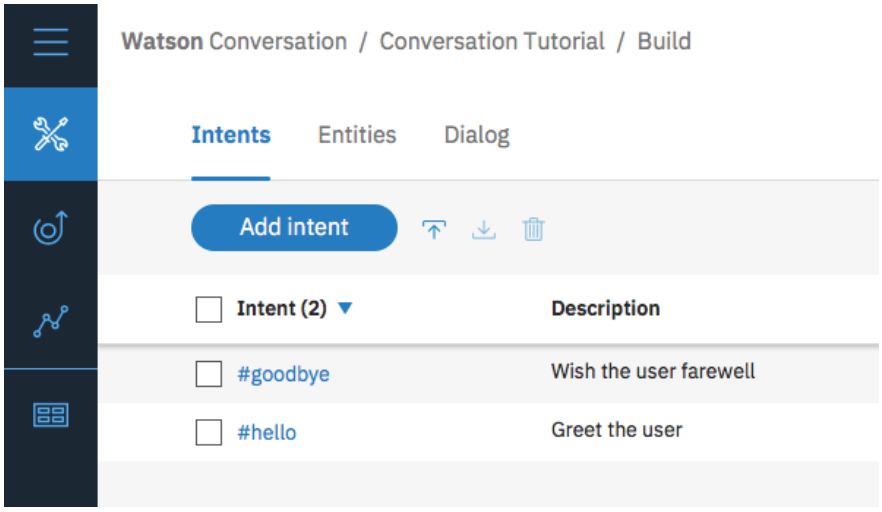
4. Add four more examples:

    - good morning
    - greetings
    - hi
    - howdy

5. Click the **Close** icon to finish creating the #hello intent.
6. 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.



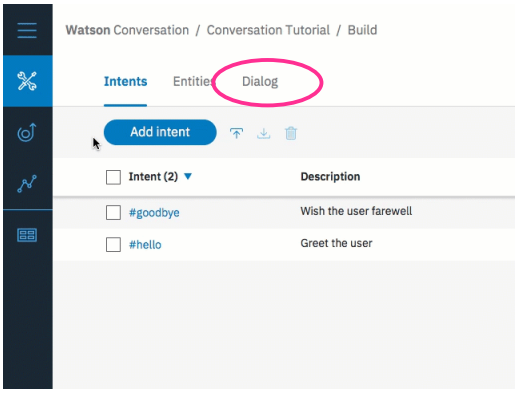
IBM Cloud

Build a Dialog

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.

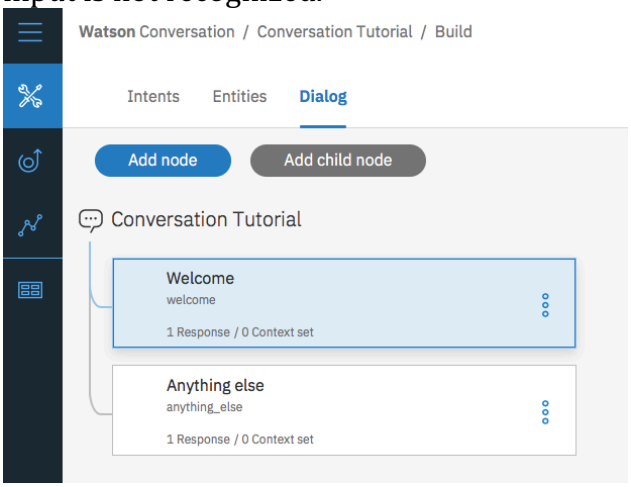
Adding a start node



1. In the Conversation tool, click the **Dialog** tab.

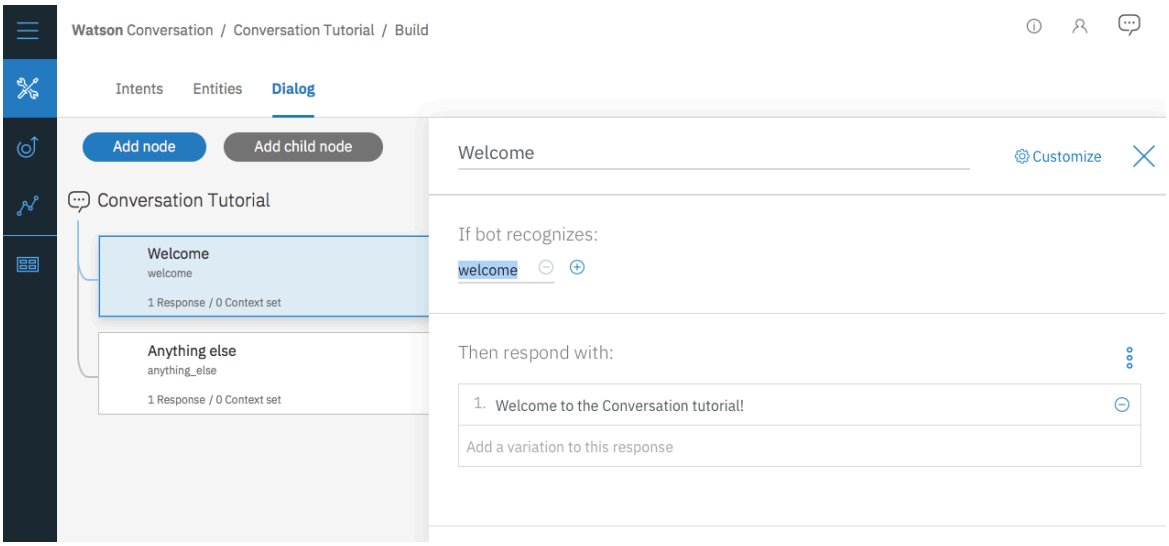
2. Click **Create**. This creates two nodes:

   - **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.

4. Replace the default response with the text, `Welcome to the Conversation tutorial!`.

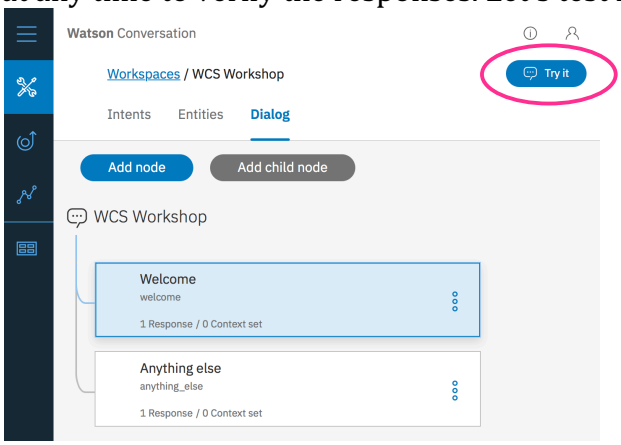


5. Click × to close the edit view.

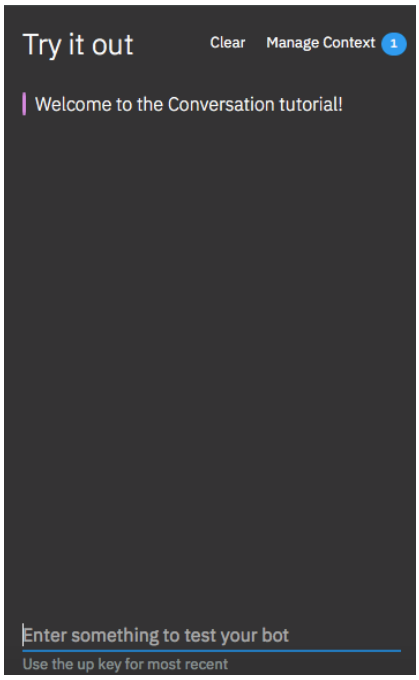
You created a dialog node that is triggered by the welcome condition, which is a special condition that indicates that the user has started a new conversation. Your node specifies that when a new conversation starts, the system should respond with the welcome message.

Testing the start node

You can test your dialog at any time to verify the responses. Let's test it now.



Click the Try it icon to open the *Try it out* pane. You should see your welcome message.

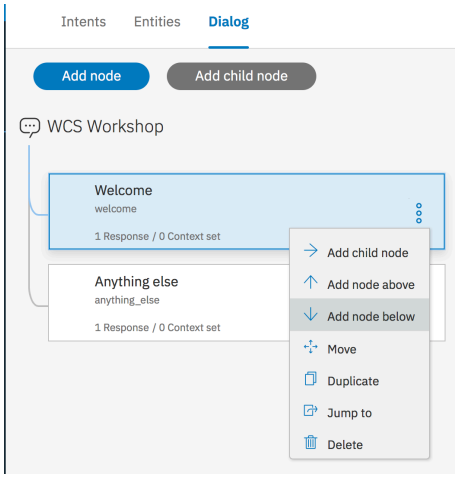


Click × to close the *Try it out* view.

Adding nodes to handle intents

Now let's add nodes to handle our intents between the Welcome node and the Anything else node.

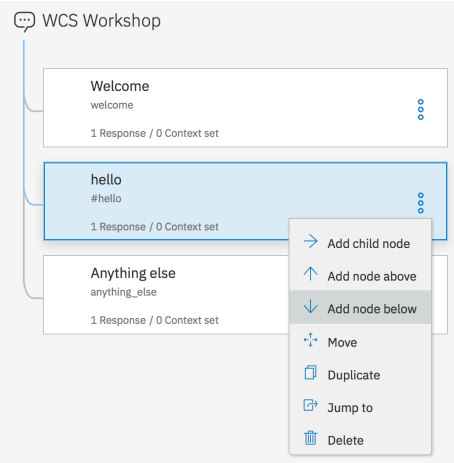
1. Click the More icon ⋮ on the **Welcome** node, and then select **Add node below**.



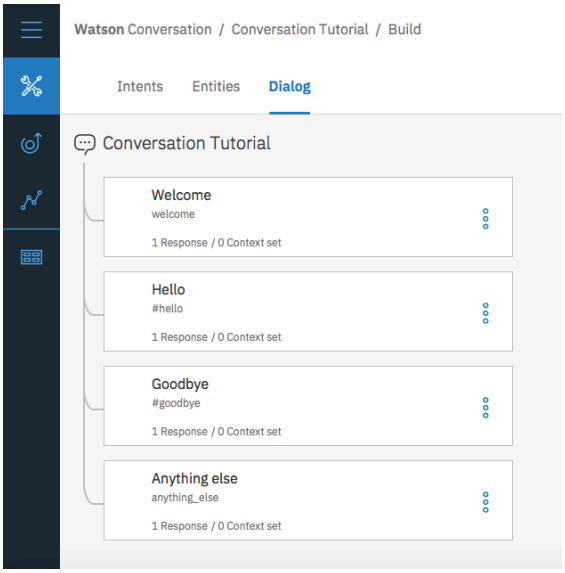
2. Give the node a name (eg hello) and in the **If bot recognizes** field, type `#hello`.

3. Add the response, `Good day to you`.

4. Click × to close the edit view.
5. Click the More icon ⋮ on this node, and then select **Add node below** to create a peer node.



6. In the peer node, specify `#goodbye` as the condition, and `OK. See you later!` as the response.
7. 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

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.

Add Advanced Intents

1. Go back to the Intents page and click **Add intent**.

2. Name the new 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. Click **Create.**

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 to the #turn_on intent.

You now have three intents, #turn_on which you just added, and #hello and #goodbye. Each intent has a set of example utterances that help train Watson to recognize the intents in user input.

Maintain Entities

IBM Cloud

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**.

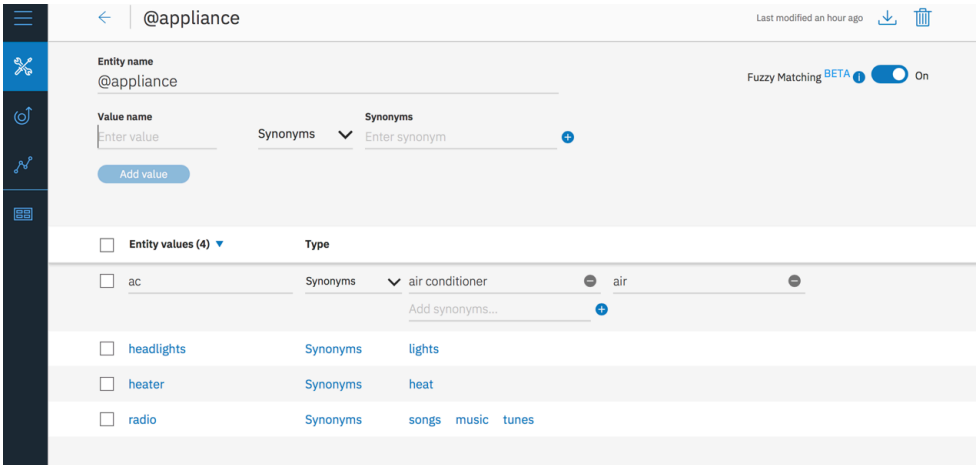
3. Add the @appliance entity name, and then press Enter. The @ appliance entity represents an appliance in the car that a user might want to turn on.

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.

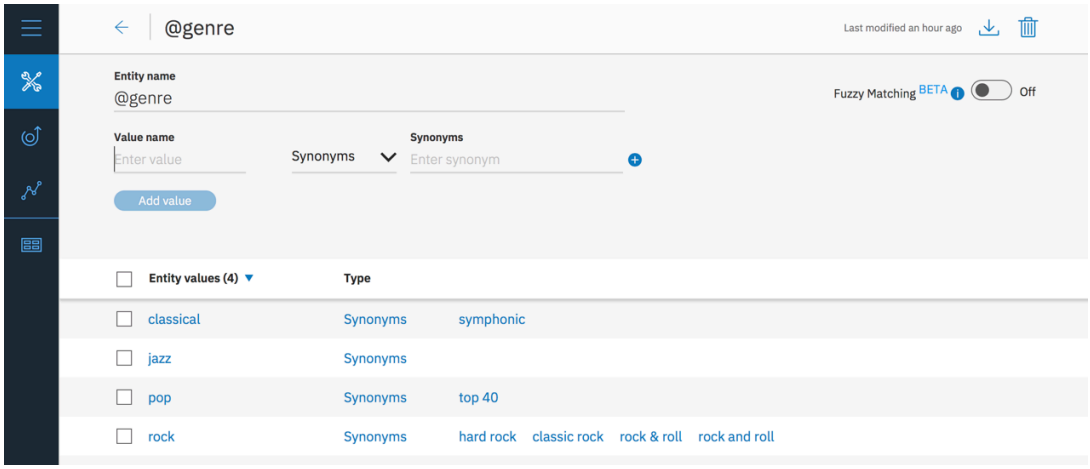
It should look like this:



7. Repeat Steps 2-6 to create the `@genre` entity with fuzzy matching on for 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 create the correct response.

Create Complex Dialog

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
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 echoes 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.

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. Name the node `Music`.

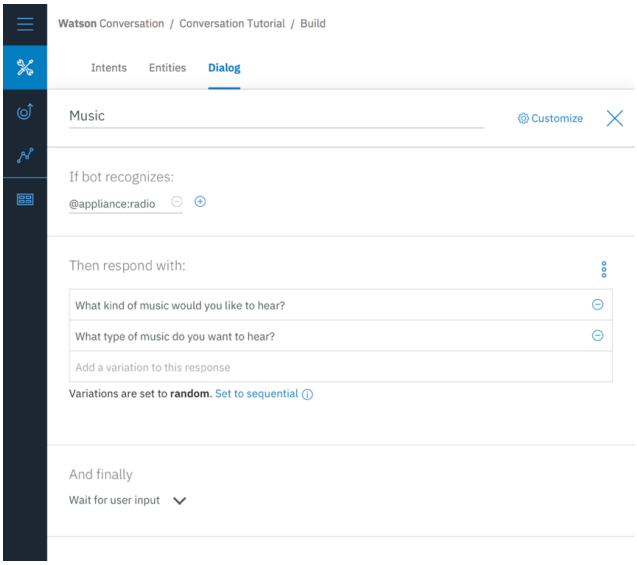
3. In the condition field, enter `@appliance:radio`. This condition is true if the value of the @appliance entity is `radio` or one of its synonyms, as defined on the Entities tab.

4. 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?`

5. Set the variation to Random by clicking on the *Set to Random* link.

6. Click × to close the node edit view.

Your dialog for Music should look like this:

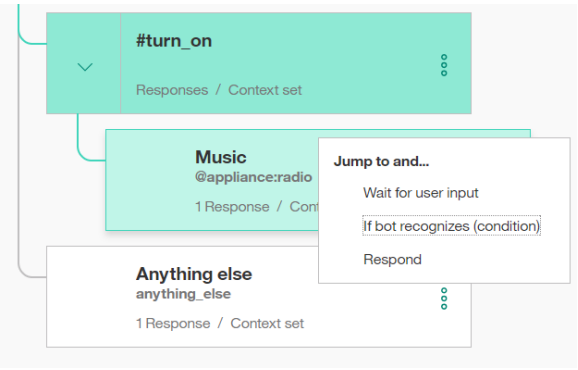


Add a jump from the #turn_on node to the Music node

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.

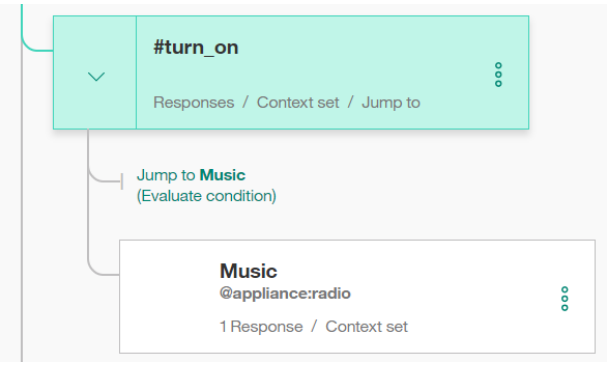
1. Click the More icon ⋮ on the **#turn_on** node, and select **Jump to**.

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.



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.

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.

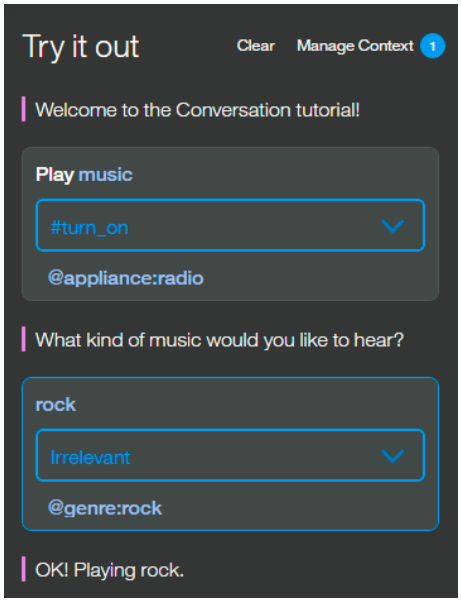
That takes care of all the cases where the user asks to turn on the music.

Test the dialog for music

1. Select the Try it icon to open the chat pane.

2. Type `Play music`. The bot recognizes the #turn_on intent and the @appliance:radio 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.



4. Type Play music again, but this time specify an invalid response for the genre. The bot responds that it does not understand.

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 echoes 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:radio 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.

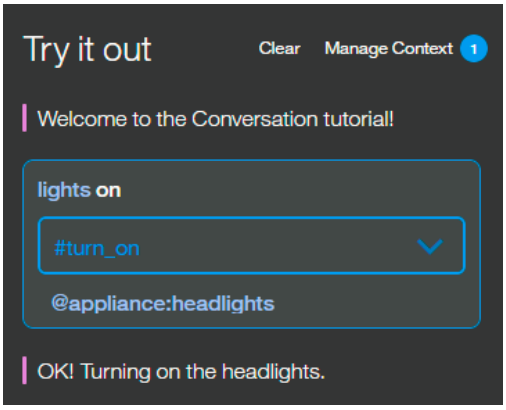
3. Enter `OK! Turning on the @appliance` as the response. This response reiterates the appliance value that the user provided.

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.

Scenario 3

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.

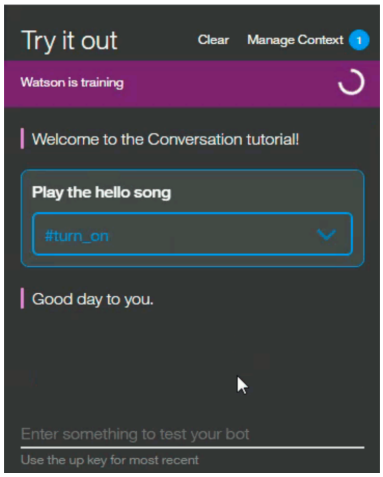
2. Enter `true` in the condition field. (If the user specifies a valid @appliance value, this node will never be reached.)

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. Click the **Back to workspaces** button  from the navigation menu. On the **Car Dashboard - Sample** tile, click **Edit sample**.

.

IBM Cloud

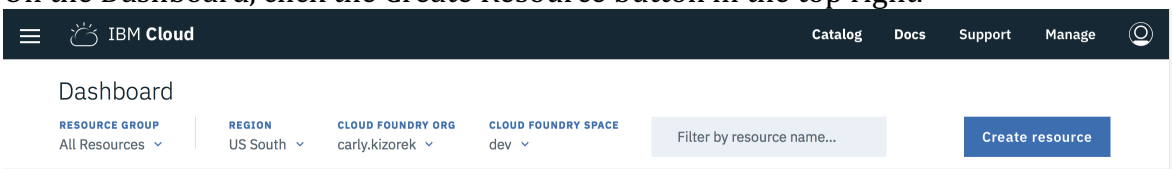
Module 2: Deploy a Node-RED web app

Create Node-RED Instance

Now that you have built and tested your Watson Conversation workspace, you can deploy it by connecting it to a user interface. Click the link below.

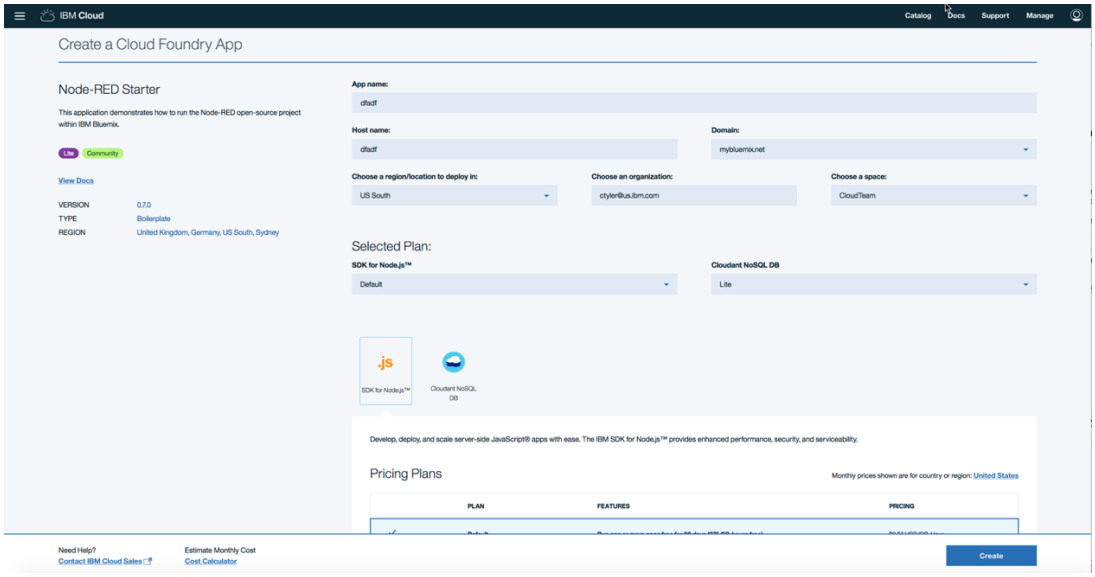
1. Login to IBM Cloud (https://bluemix.net/)

2. On the Dashboard, click the Create Resource button in the top right.

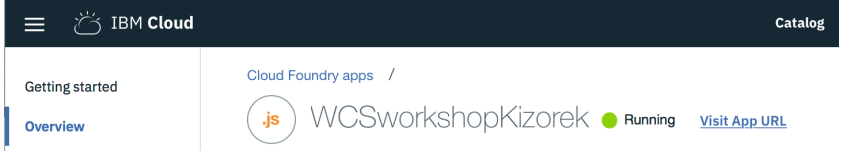


3. On the search page, search for Node-RED and click on the **Node-RED Starter**.

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.

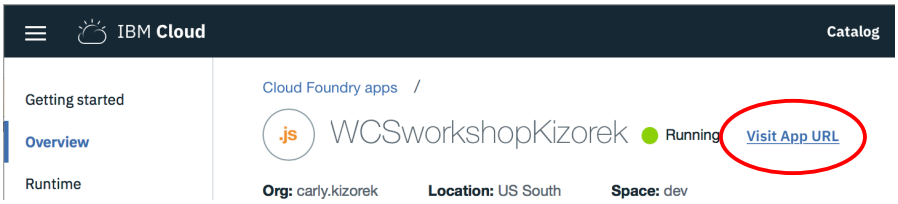


5. You will land on the **Getting Started** page for the Node-RED Starter instance and you will see the Starting icon rotating. Wait for this icon to turn green and display "Running". Once you get the green dot and "Running", click the **Visit App URL** link.



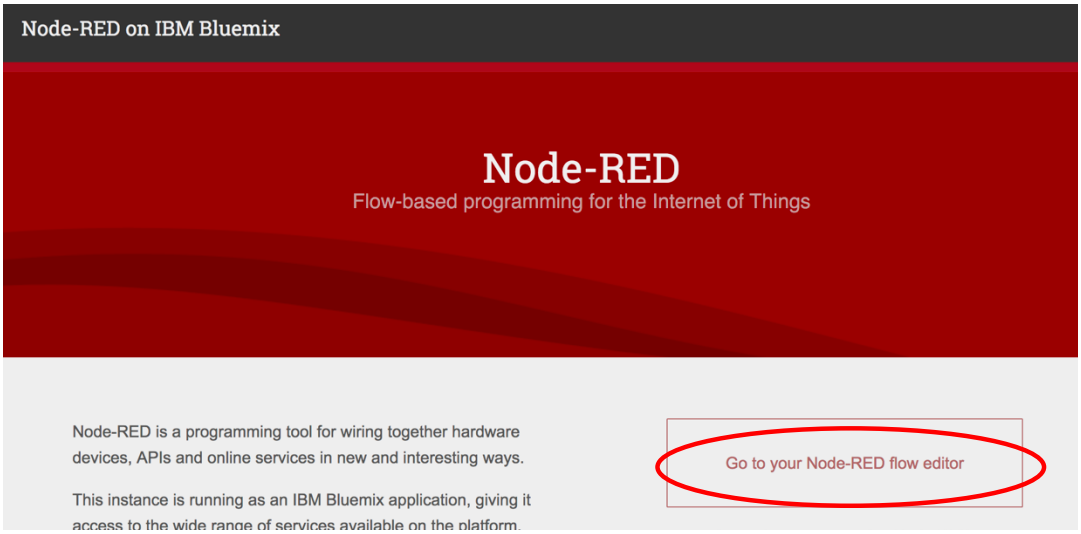
Configure Node-RED App

1. Once the app has restarted, and is Running, click **Visit App URL**.

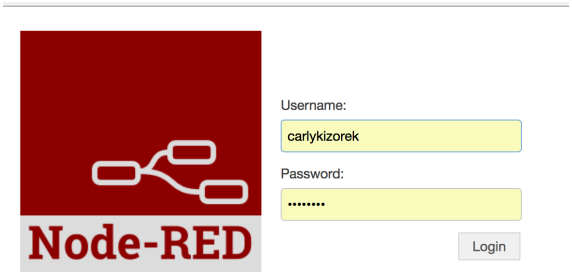


When you open your Node-RED app for the first time you will be prompted with options to secure the editor, etc. This username and password are completely independent of any other credentials so you will have to create them. Click "Next" through the screens then click "Finish".

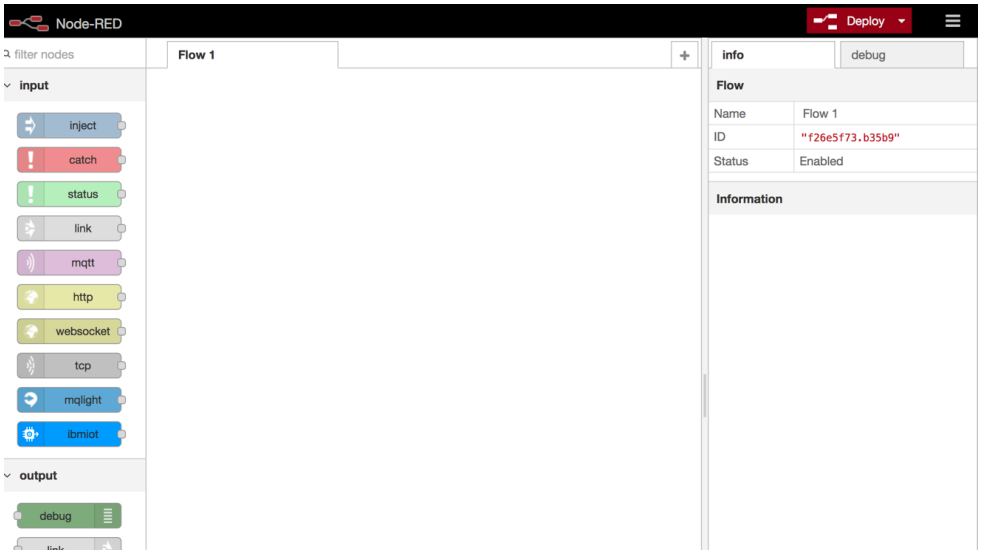
2. Click on **Go to your Node-RED flow editor.**



3. If you secured the editor with a username and password you will be brought to the login screen.

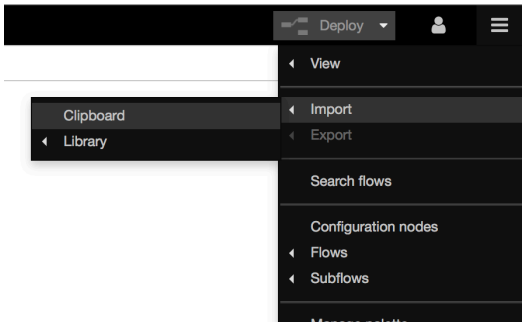


4. When the flow editor opens, on the left you will see the palette which contains nodes available that can contribute to a flow.

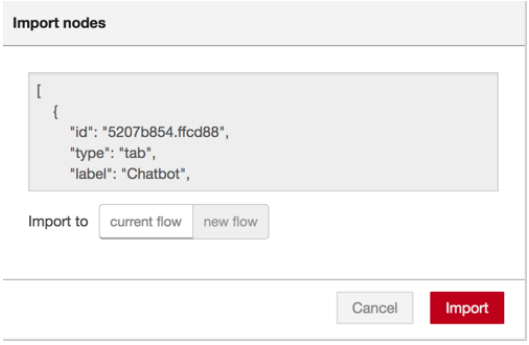


5. In the GitHub repository, copy the contents of the Node-RED-Flow Starter.json file to the clipboard (or a text file if you prefer).

6. Back in Node-RED, click the hamburger menu in the top right, and select **Import** -> **Clipboard**.



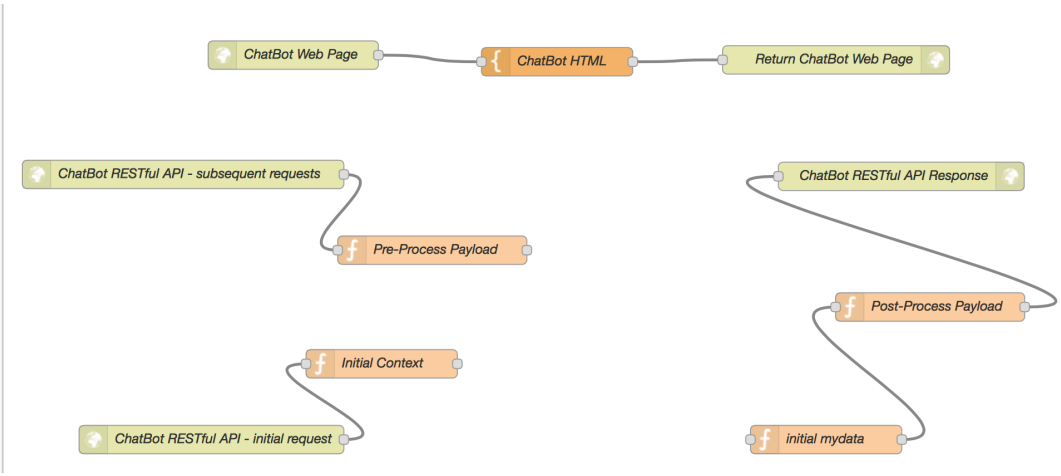
7. Paste the contents of the clipboard into the editor window. Choose to import to a **new flow** and click **Import**.



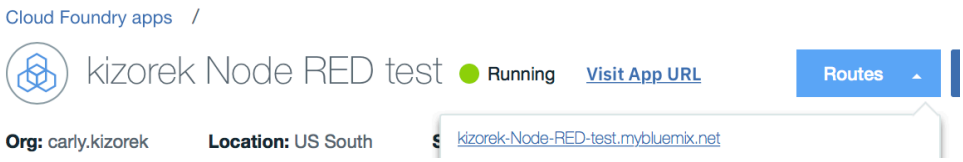
8. You should see two flows. 

You can delete the empty flow (Flow 1) by double-clicking its tab and when the **Edit Flow** dialog box appears select **Delete.**

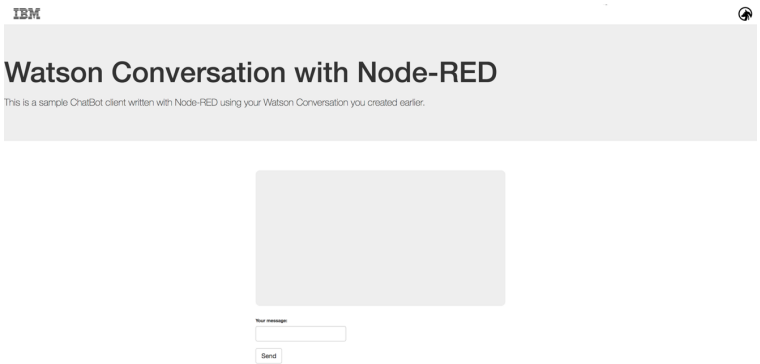
9. On the flow that you just imported, you should see a flow that looks like the following.



10. When you click **Deploy** in the top right, it will create a new web page for you which is https://<*your-Node-RED-route*>.mybluemix.net/bot.
Note: Change <*your-Node-RED-route*> to match the route of your Node-RED instance. You can verify the route of your Node-RED instance in IBM Cloud by expanding the **Routes** button next to **Visit App URL** as shown below.



11. Open a new browser tab and go to your new web app url. You should see a page that looks like this.

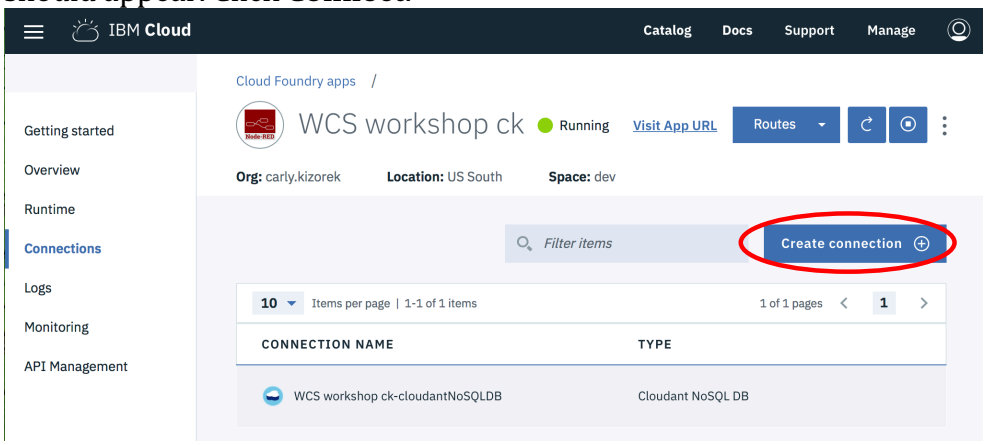


Integrate Watson Conversation Services

1. Within the Node-RED flow, we are missing the connection to the **Conversation** service, so let's add it in.

2. Return to IBM Cloud (should still be an open tab) and go back to the **Node-RED Starter** service page.

3. On the left select the **Connections** section, click on the **Create Connection** button. Find the Conversation Service you created earlier. When you hover over it, a **connect** button should appear. Click **Connect**.



4. This will cause your Node-RED application to be Re-Staged. Allow it to be re-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.

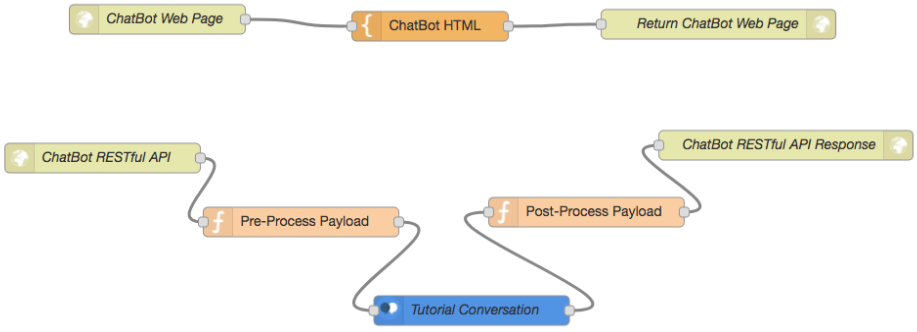
5. Go back to your Node-RED flow editor (**Note:** *you may need to re-login*).

6. Now, we need to add the **Conversation** node to our flow.

7. Within Node-RED, to the left of your flow editor are your nodes. Here you have the ability to filter nodes. Type **Conversation** into the search box. Click and drag the **conversation** node into the canvas. 

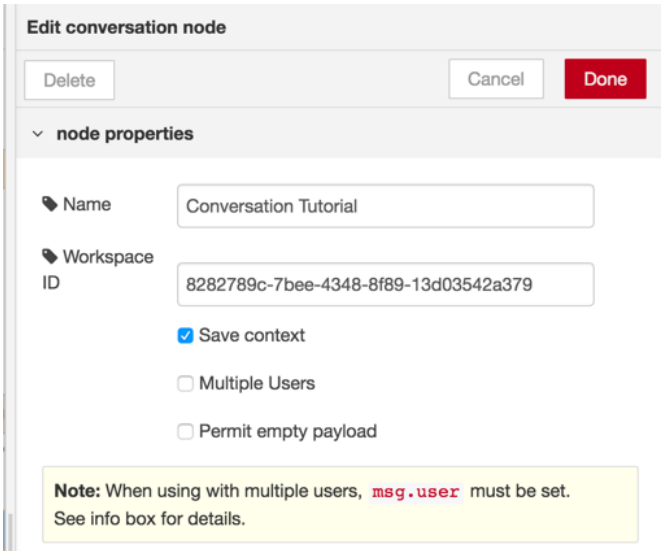
8. To connect nodes within Node-RED, you click the first dot, drag it to the second dot, and release the mouse button. Using your mouse, connect the following dots:

a. the outbound side of the Pre-Process Payload node to the inbound side of the new Conversation node.
b. the outbound side of the new Conversation node to the inbound side of the Post-Process Payload node.
Your flow should look similar to the following:



9. Double-click the **Conversation** node. Earlier in this lab you should have copied the Workspace ID into a text file after we created the Conversation Workspace. If you no longer have the Workspace ID, here are instructions on how to retrieve it:

   - Go into the Watson Conversation tool by opening the IBM Cloud dashboard (https://console.bluemix.net)
   - Click on the Conversation service you created earlier
   - Click on the **Launch Tool** button
   - Within Watson Conversation you should see your workspace. Click on the **More** icon ⋮ and choose **View Details**
   - Copy the Workspace ID then paste it into the Workspace ID field in Node-RED
   - Click Done



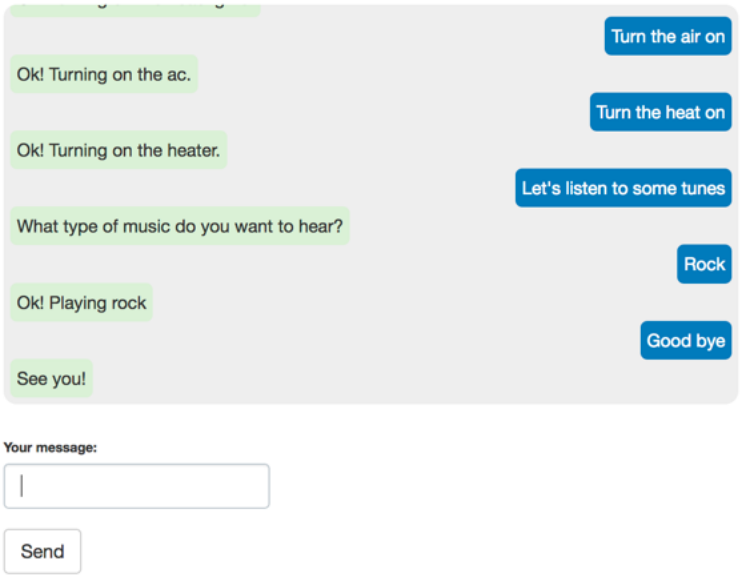
Deploy Node-RED Application

1. Click the Deploy button at the top.

2. Go back to the tab with your Chatbot Web Page and refresh the page.
   https://<*your-Node-RED-route*>.mybluemix.net/bot.

3. Enter text into the **Your message** field.
   Some of the messages you can send your chatbot include:

   - Hello
   - Turn the air on
   - Turn the heat on
   - Turn on the headlamps
   - 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
   - Lastly, say Good bye!



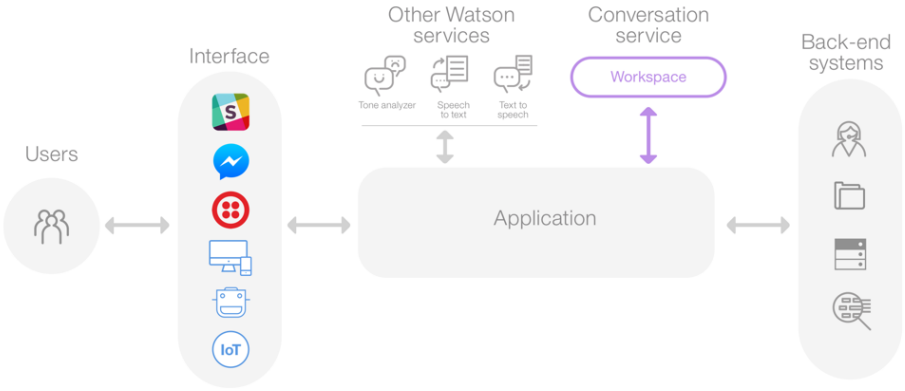
You have just built a chatbot! Congratulations! If you enjoyed this and want to play with it more, there is a **Car Dashboard - Sample** workspace in your Watson Conversation Service that is the same use case but fleshed out even more with a longer dialog and additional functionality.

IBM Cloud

Appendix: Watson Conversation Architecture and terminology

Architecture

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.



Application Overview

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.

Terminology

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.

