Watson Conversation Service

Implementation Methodology

Lab Exercises

Course Code: **WCS201**
Version: **2.1**
Updated: 03/20/2017

Jazmin Capezza JazminCapezza@us.ibm.com
Ray Lopez Ray.Lopez@us.ibm.com
Mike Hervey mhervey@us.ibm.com

Table of Contents

Course Overview

Objectives

During this lab guide, you will learn all aspects of performing the steps required for a successful domain adaptation with Watson Conversation service.

The tasks and tools that are covered will allow you to perform the steps necessary to efficiently and effectively build and refine Ground Truth and to set up a Watson Conversation Service solution.

Lab Structure

The labs are divided into sections, subsections, and numbered steps. You will perform all numbered actions. Some exercises will be performed individually, some in groups, and some as class exercises.

Exercise Environment

Parts of this lab environment are staged in your public Bluemix account and require an Internet browser for access. The preferred IBM browser is Firefox; therefore, all instructions are based on the Firefox browser. For this lab guide, you will use an instance of Watson Conversation service that you have created in your personal Bluemix account. Please see the list of prerequisites for this class.

Other sections of this lab guide will run on your local system. Instructions for these sections are for both Windows and Mac unless specified for a specific platform.

Oracle Java is also a requirement for this lab guide to use the Conversation Test Tool. Instructions are given for how to properly download the correct version of Oracle Java (NOT IBM Java) in the WCS 101 lab guide.

Course Focus

This course focuses on tasks that lab services staff and client domain experts perform during WCS engagements. We do not cover specialized activities that Global Business Services (GBS), Dev Ops, or Watson Cloud Technology Services (WCTS) commonly perform to set up or manage instances.

Acronyms Used

WCS - Watson Conversation Service

Preparation

The files that you need for the WCS 201 labs are in the WCS 201 online course materials in Watson Academy under the Lab Resources section. You have received an email that contains the link to the Watson Academy course for your specific class and your enrollment key. **All of the documents are included in the zip file `WCS201_AllLabDocs.zip`, or you may download each file individually.**

This lab guide uses the following individual files:

WCS201Lab_EndUserExamples.xlsx	cf-cli-installer_6.22.1_osx.pkg
WCS201Lab_CoreIntentsGT.xlsx	cf-cli-installer_6.22.1_winx64.zip
WCS201Lab_IntentImport_v1.csv	Conversation-simple-master.zip
WCS201Lab_EntityImport.csv	ConversationTestTool.zip
WCS201Lab_ Workspacev1.json	defaultTemplate_CoreIntents_TestSet1.xlsx

1 Activity 3: Question Collection

Before starting your domain adaptation, you will work with the client to define the implentation's end-users, to identify the use case and to agree on the success criteria for the WCS solution.

The initial step in the domain adaptation is to collect end-user representative questions. End-user representative questions are questions that the end-user asks the production system. In a WCS implementation, end-user questions are used to train and test the system to prepare for production.

In a real-life question collection scenario, the user experience should mimic the final user experience as closely as possible. Production elements to focus on are:

- The device that will be used to interact with WCS.
- The look and feel of the user interface.
- The steps users take before asking a question (greeting and beginning of conversation).
- Any available information that will be a part of the end-user experience (for example, information on the screen).

The more accurately the client emulates the production environment and experience, the better the representative questions will be and the easier it will be to train the system.

Note: If the questions that are collected during question collection are not representative of the questions that users ask in production, or if the user experience changes before production, any work that has been done with intents, entities, dialog, or performance optimization may be negated.

1.1 Lab guide use case and end-users

Use case*:* A hotel concierge virtual assistant that is accessed via a kiosk in the hotel lobby.

End-users*:* Hotel customers

1.2 Question Management: Creating Question Sets

During this lab exercise, you will modify an Excel spreadsheet to organize and manage collected questions. If you do not have Excel, you may use another spreadsheet application that is on your computer, but all lab guide instructions are based on Excel software.

Once your client has collected end-user representative questions, the next step is to create question sets for training and testing.

After collecting the representative end-user questions, the questions will need to be randomly separated in train and test sets.

- Train: 1600 questions
- Test: Equally divided, randomized subsets of the training questions to test chat flow

Here are some guidelines you should follow when creating question sets:

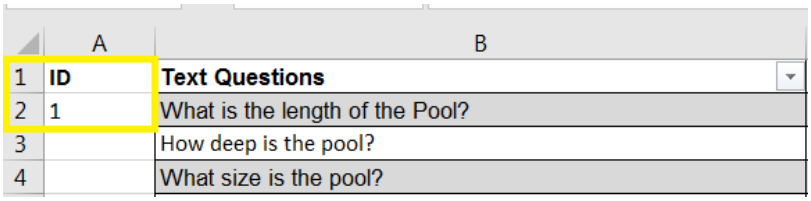
- Create question sets randomly. Never select questions to be part of a question set.
    - This lab guide will show you how to randomly obtain train and test sets using the rand() function in Excel.
- Keep track of each question and what you use it for.
    - Use question IDs

1.2.1 Create Train and Test Question Sets

The best tool for organizing question sets is spreadsheet software. For this lab, we will use Microsoft Excel to organize the questions that we have collected. The end-user examples used for this lab guide use case were collected from actual end-users for a hotel use case.

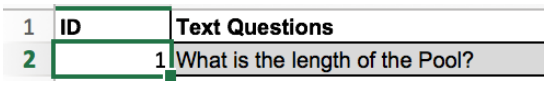
To create your question sets, you will begin by pasting your collected questions into a spreadsheet document. This has already been done for you for this lab exercise.

1. Open the `WCS201Lab_EndUserExamples.xlsx` file with Excel.
2. Click on the worksheet tab named **Question Set**.
3. In row 2 of the ID column (A2), type the number `1`.



	A	B
1	ID	Text Questions
2	1	What is the length of the Pool?
3		How deep is the pool?
4		What size is the pool?

4. In cell A2, click your cursor to create a green fill box.

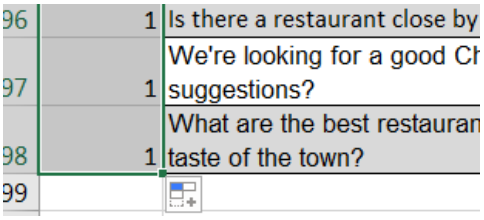


	ID	Text Questions
1	ID	Text Questions
2	1	What is the length of the Pool?

5. Use the **Fill Series** function to give each question an ID.

   Note: One way to do this is to put hover your cursor over the A2 box until your cursor turns into a black plus sign.

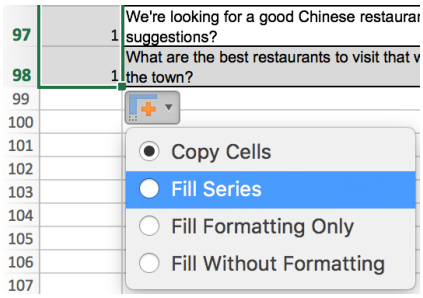
6. Left click and pull the green square down until the last question in the spreadsheet.



96	1	Is there a restaurant close by
97	1	We're looking for a good Cl suggestions?
98	1	What are the best restauran taste of the town?
99		

7. If the column fills with only 1s, then click on the menu button that appears at the bottom of the column.

8. Click **Fill Series**.



The numbers will now update to be in numerical order.

96	95	Is there a restaurant
97	96	We're looking for a suggestions?
98	97	What are the best r taste of the town?

To randomize the questions before you separate them into train and test partition sets, you will use the random function in the column labelled Random (C).

9. Enter the formula `=RAND()` in cell C2.

	A	B	C
1	ID	Text Questio	Random
2	1	What is the length of the	=RAND()

10. Press Enter.

Pressing Enter will fill all of the cells with a random number.

Note: If all cells in the column do not populate with numbers, double-click the lower right corner of the top cell. This will apply the formula to all rows.

Next, you will sort the entire table based on the random numbers that Excel generated.

11. Click on the drop down menu in cell C1. For Windows, choose **Sort Smallest to Largest** in the menu; For Mac, click on **Ascending**. This will sort all columns based on the random number from smallest to largest.

	A	B	C
1	I	Text Questions	Random
2	30	What time does the restaurant close?	0.5576839
3	64	How can I book a taxi?	0.0950708
4	23	Good sitdown reaturants in the area.	0.7660229
5	73	When are you serving dinner?	0.7174866
6	71	Hello I'm starving I'm looking for a place to eat	0.4557262

Note: Notice that the ID numbers in Column A are now rearranged. Your numbers will be different than the image.

Not that you have a randomized set of end-user examples, you are ready to cluster your ground truth.

2 Activity 4: Create Ground Truth

During this lab exercise, you will modify a Microsoft Excel spreadsheet to organize end-user examples after intents and entities have been determined. If you do not have Excel, you may use another spreadsheet application that is on your computer, but all lab guide instructions are based on Microsoft Excel software.

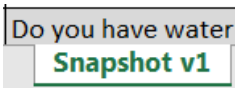
The document used for the lab guide use case is based on core intents. In class, we teach two methods to naming intents, core intents and full intents. In your lab guide documents, there is a spreadsheet document named `WCS201_Full_IntentsGT.xlsx`. This document is available as an example for how this Ground Truth would have been created if the Subject Matter Expert who created the Ground Truth decided to use a mixture of core intents and full intents.

2.1.1 Create Analyze Worksheet

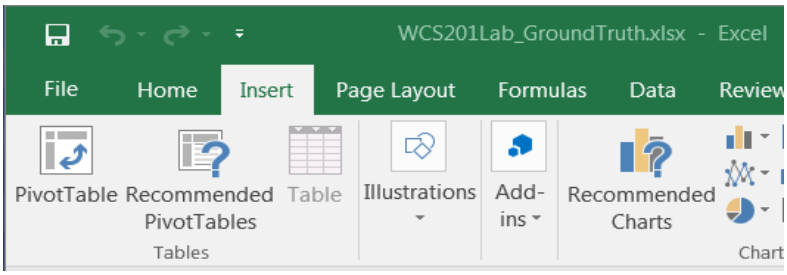
The best method to analyze the ground truth is to use a pivot table in Excel. This section will walk you through how to create a pivot table using the **Snapshot v1** worksheet in the `WCS201Lab_CoreIntentsGT.xlsx` document.

Note: There is a tab in this Excel document, Analyze v1, that has a completed pivot table for you to reference.

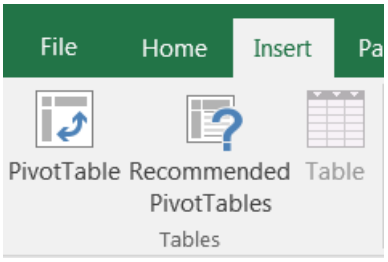
12. Click on the **Snapshot v1** worksheet in the `WCS201Lab_CoreIntentsGT.xlsx` spreadsheet.



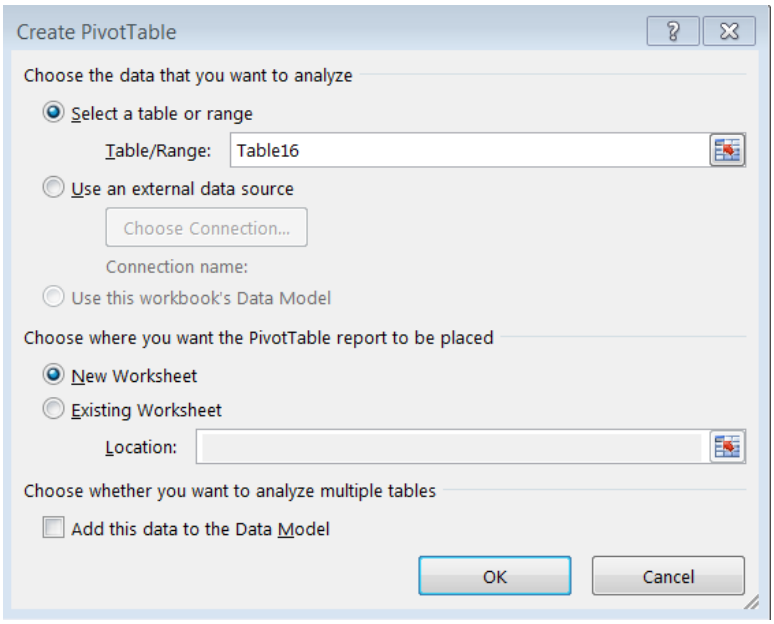
13. Click on the **Insert** tab.



14. Click on **PivtotTable**.



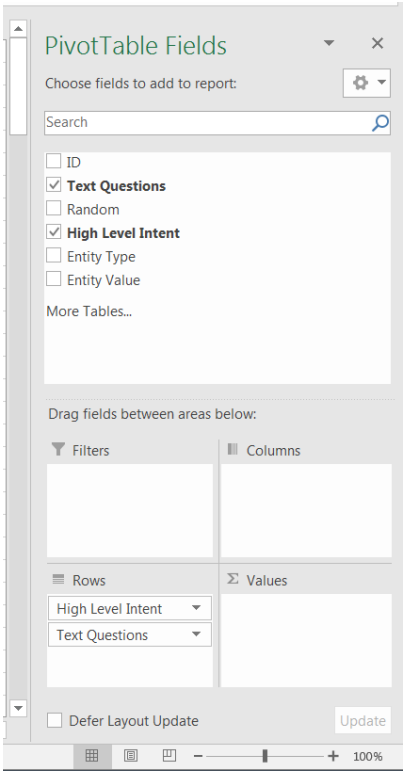
15. In the Create PivotTable window, select the **Select a table or range button**, do not change the default Table, and select **New Worksheet**.



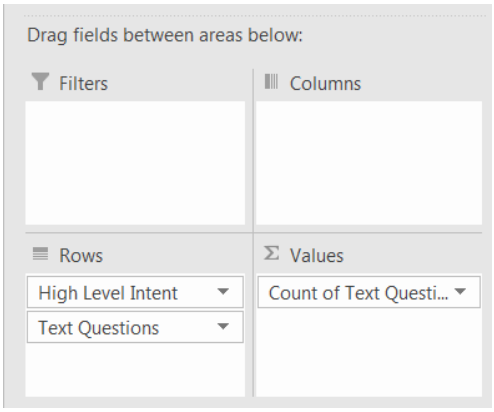
16. Click OK.

2.1.1.1 [Windows] PivotTable Instructions

17. [Windows] In the PivotTable Fields window on the right, select **High Level Intent** and **Text Questions** for the **Rows** field. If **High Level Intent** is not the first field in the **Rows** list, then drag it above **Text Questions**.



18. `[Windows]` Click on **Text Questions** in the Fields area to highlight it, hold your mouse button, and drag it into the **Values** box. The value in **Values** will read *Count of Text Questions*.



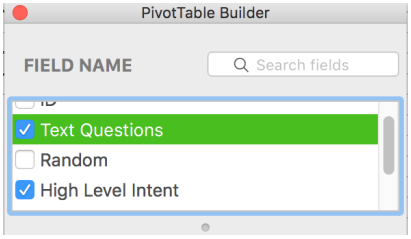
`[Windows]` Your Pivot Table should now look similar to this table:

Intents	Count of Text Questions
⊞ **hotel_info**	**12**
⊞ **hotel_hours**	**16**
⊞ **hotel_locations**	**28**
⊟ **hotel_procedure**	**11**
Hey can I get some new towels up to my room	1
How can I arrange transportation?	1
How can I book a cab?	1
How can I book a taxi?	2
How do I adjust the heat/AC in the guestroom?	1
How do I arrange a shuttle or a taxi to the airport?	1
How do I arrange a shuttle or taxi to the airport?	1
How do I book a large reservation in the restaurant?	1
How do I set the clock alarm in the guestroom?	1
how do I turn the ac off in my room	1
⊞ **local_recommend**	**35**
Grand Total	**102**

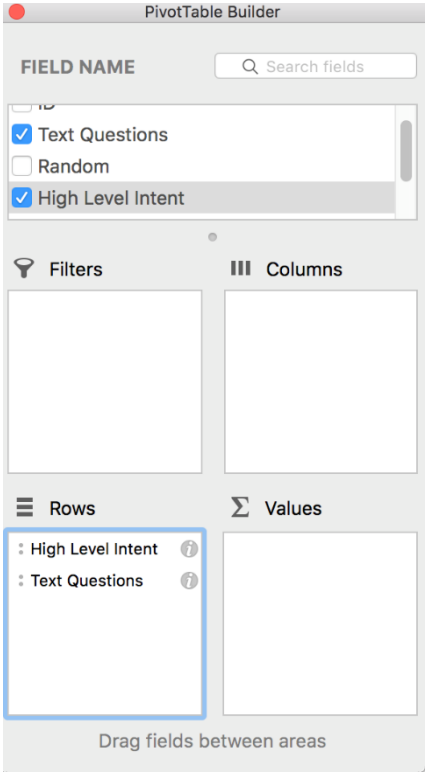
Note: Notice that you can minimize and maximize the high level intent sections of the pivot table to review all questions that have been categorized with each intent.

2.1.1.2 `[Mac]` PivotTable Instructions

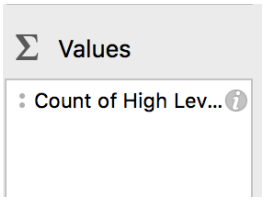
19. `[Mac]` In the PivotTable Builder popup window, select **Text Questions** and **High Level Intent**.



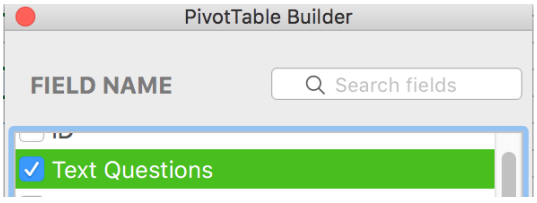
20. [Mac] In the **Rows** field, make sure that **High Level Intent** comes before **Text Questions** in the list. If **High Level Intents** does not show up in the list, simply click on the **Field Name** that you checked then drag and drop it into the **Rows** field. You can drag and drop the field names to change their order.



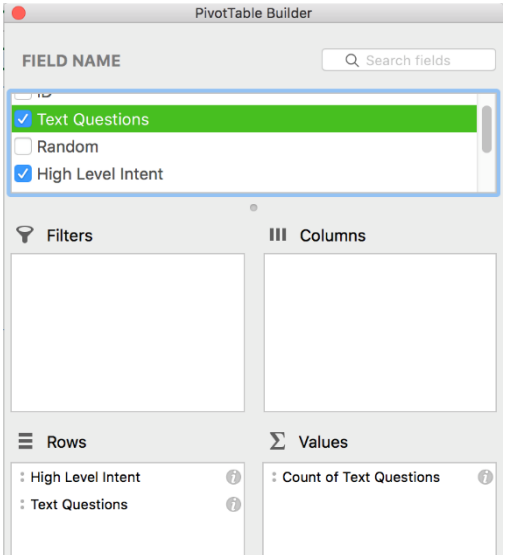
21. [Mac] If count of **High Level Intents** shows up in the **Values** field, click on it to highlight it, right click, and click on remove field.



22. [Mac] Click **Text Questions** in the **Field Name** field.



23. [Mac] Drag **Text Questions** into the **Values** field. It will now show up in the **Values** field as *Count of Text Questions.*



`[Mac]` Your PivotTable should now look similar to this table:

Intents	Count of Text Questions
hotel_info	**12**
hotel_hours	**16**
hotel_locations	**28**
hotel_procedure	**11**
Hey can I get some new towels up to my room	1
How can I arrange transportation?	1
How can I book a cab?	1
How can I book a taxi?	2
How do I adjust the heat/AC in the guestroom?	1
How do I arrange a shuttle or a taxi to the airport?	1
How do I arrange a shuttle or taxi to the airport?	1
How do I book a large reservation in the restaurant?	1
How do I set the clock alarm in the guestroom?	1
how do I turn the ac off in my room	1
local_recommend	**35**
Grand Total	**102**

Note: Notice that you can minimize and maximize the high level intent sections of the pivot table to review all questions that have been categorized with each intent.

3 Activity 5: Configure Dialog Component

Watson Conversation Service applies cognitive computing techniques to return the best matching classes for a user's utterance. For example, you submit a question and the service returns keys to the best matching answers or next actions for your application.

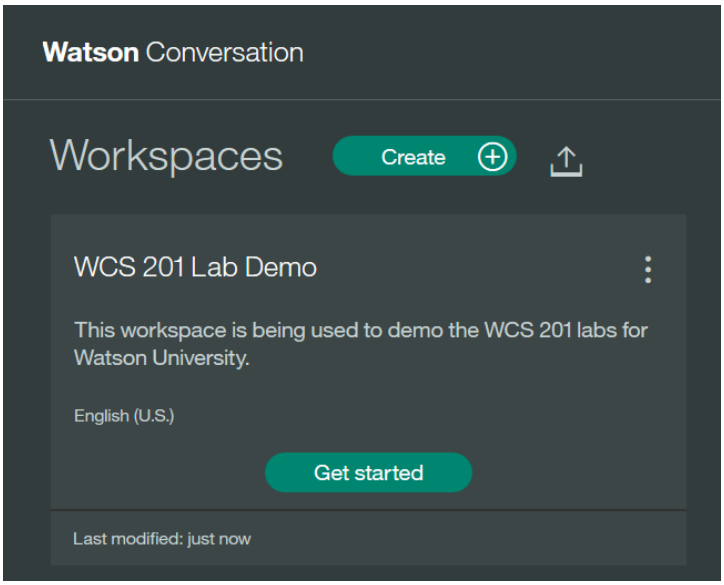
This lab guide is focused on giving you recommended best practices to support an enterprise implementation of Conversation.

To launch the WCS tooling, you may either sign in to your Bluemix account and click Launch Tool for your instance of WCS, or you may click on this link and sign in using your IBM ID and password: `www.ibmwatsonconversation.com`

Definitions for this section

`Workspace`: A container for all of the artifacts that define the behavior of your service. You will configure WCS by using the tooling within a workspace. To prepare for this section of the lab guide, you should create a workspace within your WCS instance. Give your workspace a name and description.

Note: If you need guidance for creating an instance of WCS, please see the `WCS 101 Lab Guide` in the WCS online course materials on Watson Academy.



The contents of a workspace can be downloaded and imported as a `JSON` file. In this lab, you will be shown how to import a workspace `JSON` file to create a new workspace.

`Training`: When you add intents, entities, or dialog nodes to your workspace, the service automatically trains your bot to use them.

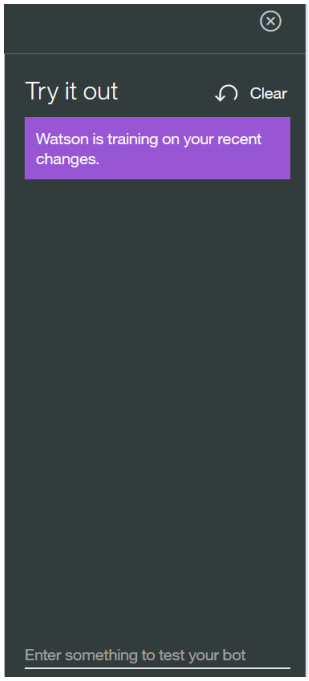
`Try it out`: From any panel in WCS, you can open the "Try it out" panel to enter a test input. The panel will display the intent and entity value that WCS recognized for the input and the response from the dialog node based on its condition.

Note: The "Try it out" panel will be used for testing in the lab guide. From any panel, click on  to open the panel. Also note that when you hoover your mouse over the icon, the popup text may read, *Ask Watson*.

This is the “Try it out” panel when WCS is training on recently added data. You must wait for the message highlighted with purple to clear before you can test newly added intents and entities.

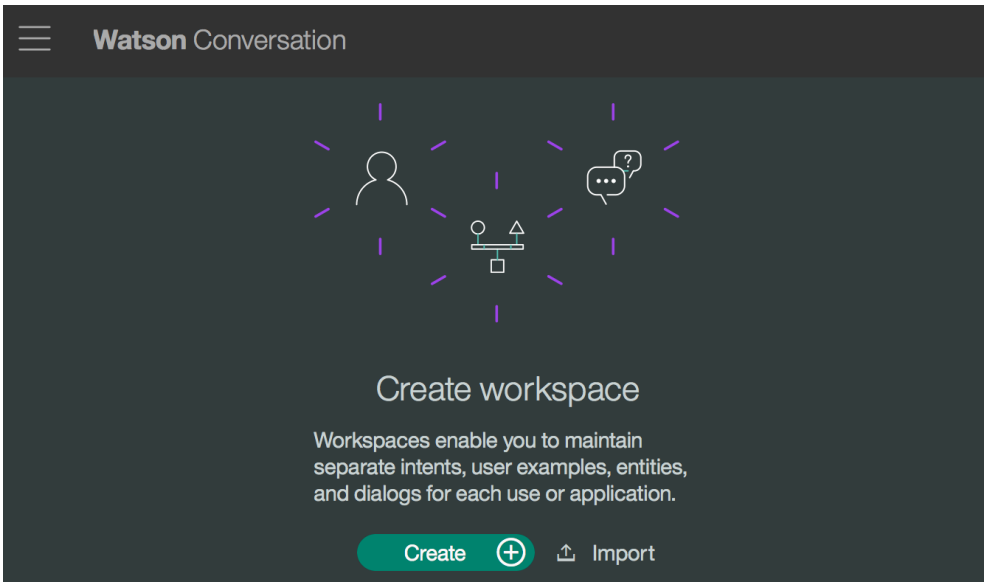
Watson will respond, but you will get unpredictable results until the training is complete.

To close the panel, click on the x in the upper right corner. This panel covers parts of the WCS screen, so if you cannot find a button that is referenced in this guide, you may need to close this panel.

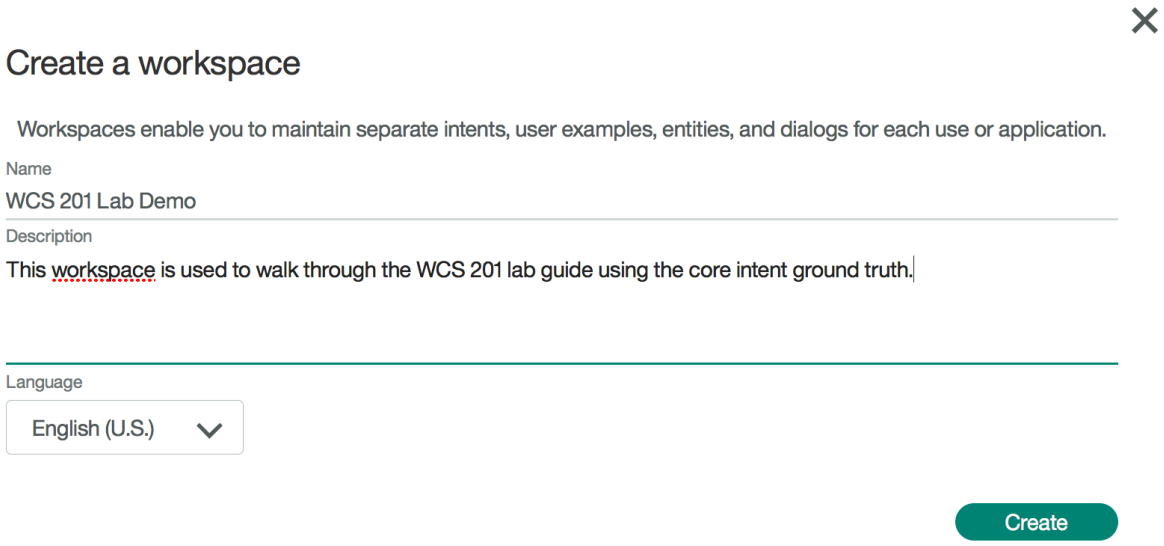


3.1 Create a Workspace

24. Click **Create.**



25. Type a name for your workspace, type in a description, and click **Create**.



Your new workspace should automatically open if it is the first workspace you created in your instance of Conversation.

3.2 Intents

An **intent** is the specific goal or idea of a user's input. Intents and end-user examples can be imported using UTF-8 text in a comma-separated value (`CSV`) file.

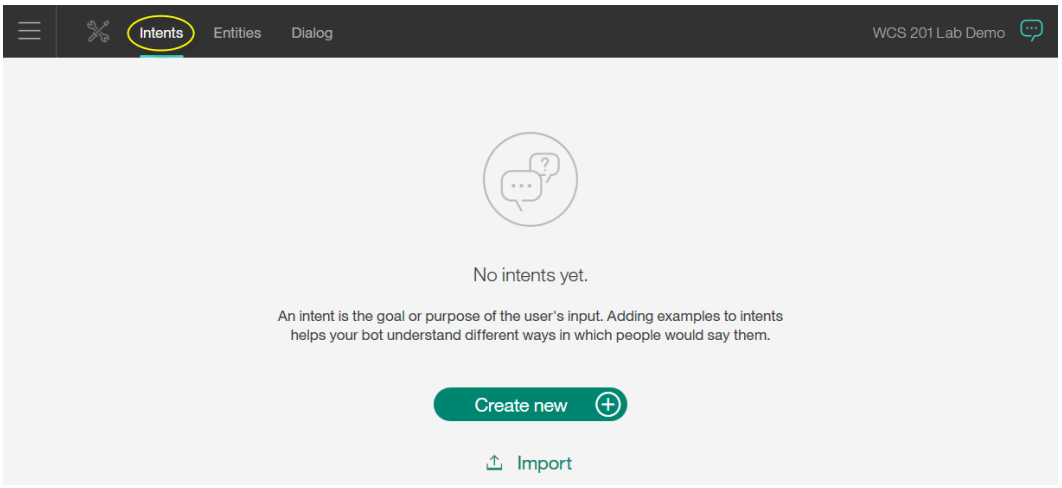
Facts about intents:

- Symbol used to refer to intents in dialog nodes: #
  - Do not include the # in the intent name when you create a new one.
- Intents are not case sensitive (i.e. `hotel_info == HOTEL_INFO`).
- Intent names can contain characters that are letters, numbers, underscores, hyphens, and dots.
- Intent names cannot contain spaces.
- Character limits for intents: 128.

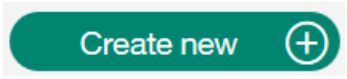
3.2.1 Add Intents Manually

You can use the WCS tooling to manually add intents and end-user examples.

26. Click on the Intent panel.

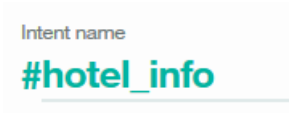


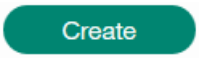
27. Click **Create new**.



28. Click in the **Intent** name field.

29. Enter `hotel_info` into the **Intent** name field (after the #).



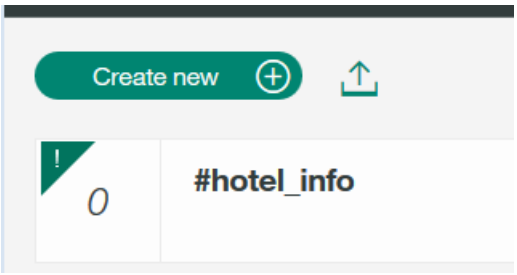
30. Click 

in the upper right corner of the screen. There is an *X* beside the create button.

Note: If you cannot see the **Create** button, close the "Try it out" panel. ☺

You should see this confirmation message at the top of your screen. You can click the **x** to close the message or wait for it to disappear.

Your Intent panel should now look like this image:



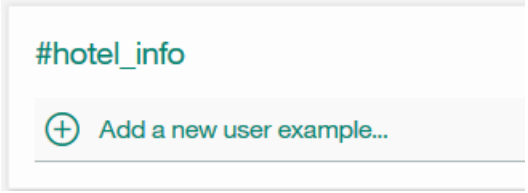
Note: To the left of the `#hotel_info` intent, there is a square the contains a zero and an exclamation point in the upper left corner highlighted in teal.

The digit is the number of user examples that have been entered for this intent. In our case, we have not yet added any user examples (we will do this next).

The exclamation point is a reminder that this intent has fewer than 5 user examples. It is the recommended best practice for every intent to have at least 5 user examples, so this exclamation point is just there as a reminder. For production use cases, IBMers have found that using 8-10 end-user examples for each intent yields higher confidence in recognizing intents for new end-user inputs.

Next, you will manually add user examples for the `#hotel_info` intent.

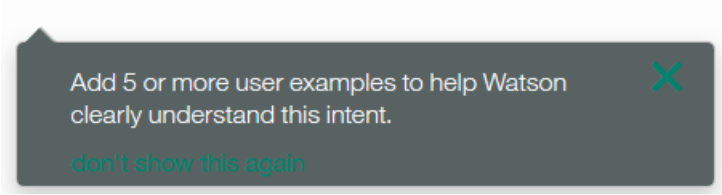
31. Click on the `#hotel_info` intent.
32. Click in the "Add a new user example…" field.



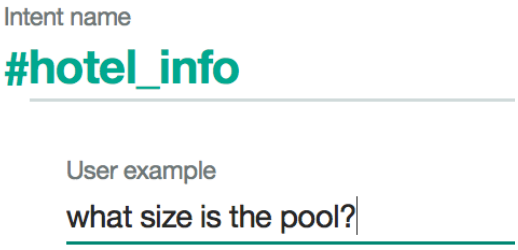
33. Type `what size is the pool?`

34. Click the ⊕ icon at the end of the user example field or press Enter to add the user example.

Note: You may get the below error after you click the plus symbol. This is normal. WCS is programmed to remind you that it needs at least 5 user examples for each intent.



Your Intent panel will now look similar to this:



To add more examples to an intent:

35. Click on the intent name.
36. Click ⊕ next to the "Add new user example…" field or click your cursor in the text field.
37. Type `can I bring food to the pool?`
38. Click the ⊕ or press Enter.
39. Repeat the previous 3 steps until you have added at least 5 examples. Here are three more questions from the lab's Ground Truth for this intent (misspellings and punctuation are intentional):

- `Do you have bikes?`
- `Do you have water in your ygm`
- `Is there a life guard?`

Note: Here are a few more things that you can do to work with intents:

- Rename an intent
- Delete an intent
- Add, edit, or delete end-user examples for an intent
- Move an end-user example to a different intent.

3.2.2 Import Intents

Intents can be imported into the Conversation tool using UTF-8 in a `CSV` file.

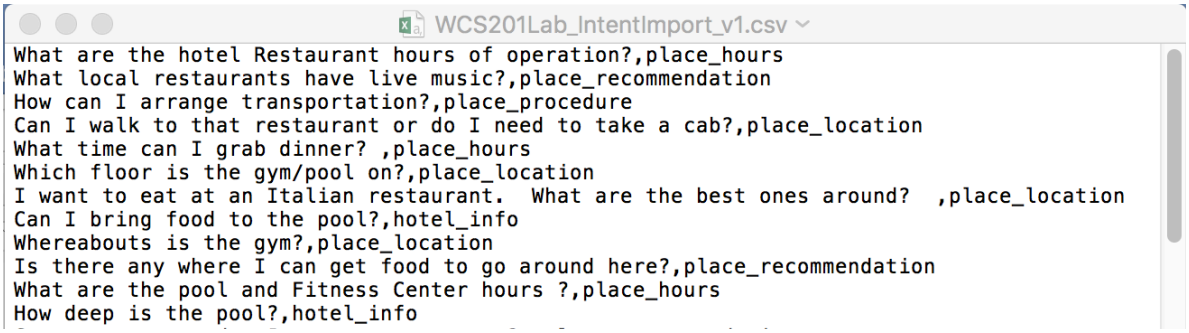
In order to import a file successfully, it must include only two columns: one for user examples and one for intents.

3.2.2.1 Prepare Ground Truth for Import

Note: This step has already been done for you. This section is for future reference only. Please proceed to the next numbered lab step to proceed with the lab guide exercises.

- Copy the Text column and the Intent column into a new spreadsheet workbook.
- Do not insert any column headings for the intent import file.
- Use a text editor to make sure that your file is UTF-8.

- This step is important because many text processing software programs add spaces and other foreign characters to text when it is copied and pasted into a new source file that are often hidden. When the files are uploaded into the Conversation tooling, these spaces can be converted to symbols that will negatively affect learning by adding text that the user did not, and in many cases, will not type or speak to Watson.
- Your questions may not be the same as the image below, but your columns should look similar with questions in one column and intents in the next column.



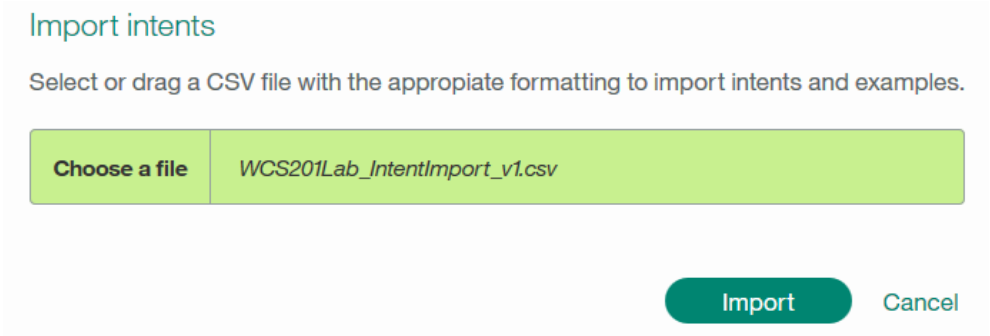
- Save the document with the File Format: Comma Separated Value (`.csv`)

- It is always best to double check your import document to make sure that all non-end-user-input symbols have been removed, since some foreign characters may have been added from saving your document to UTF-8. Also look for questions that have commas and remove the comma from the example text.

3.2.2.2 Import Intents and User Examples

To import an intent `CSV` file:

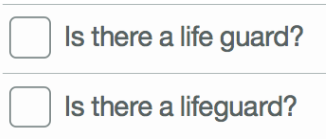
40. Open the Intents panel.
41. Click  .
42. Drag the `WCS201Lab_IntentImport_v1.csv` into the **Choose a file** box or click **Choose a file** to browse your computer.



43. Click **Import**.

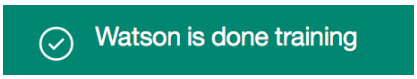
44. Click the intent that you created, `#hotel_info`, to view the end-user examples (there should be 15).

Note: Notice that Conversation did not import any questions that were exact duplicates of questions that you manually entered. Duplicates are not case sensitive, for example, *What size is the pool?* is the same as *what size is the pool?* . It did import questions that had semantic differences, such as the absence of punctuation, extra spaces between words, or misspelled words. For example, note that the only difference between the two questions below is the spacing between the words *life guard*.



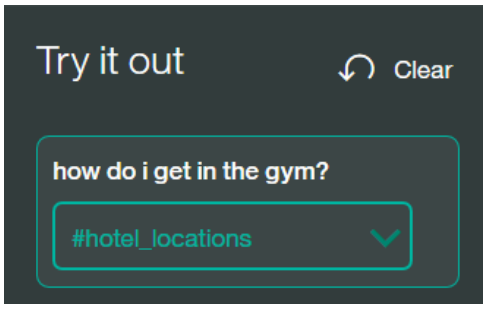
3.2.3 Test and Refine Intents

You cannot perform the **`Try it out`** activities in this section until Watson has finished training on your recent changes. When Watson is finished, the purple box with the text, "Watson is training on your recent changes" will no longer appear in your Try it out panel or you will get a message stating that:




Try it out

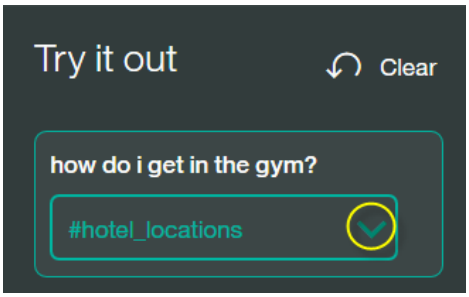
45. Type `how do I get in the gym?`



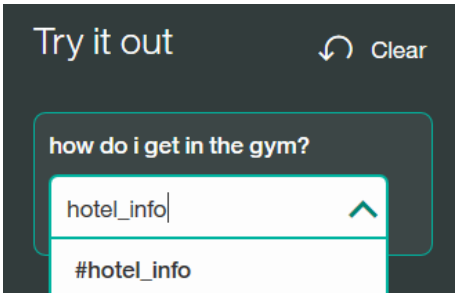
Note: Notice that Watson assigns this question the intent `#hotel_locations`. The `#hotel_locations` intent is meant for questions that refer to finding directions for locations within the hotel. In this case, the user wants to know how to get in the gym, not how to get to the gym.

Let's change the intent for this user example.

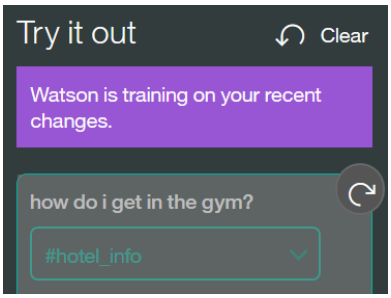
46. Click on the arrow next to `#hotel_location`



47. Type `hotel_info`



48. Click the intent name `#hotel_info` that appears in the drop down box.
49. Your "Try it out" panel will now show the message "Watson is training on your recent changes."



Note: If you go to your Intents panel, and open the `#hotel_info` intent, you will now see that your input has been moved to this section.

3.3 Entities

Entities are inputs that alter the way Watson responds to the user's intent.

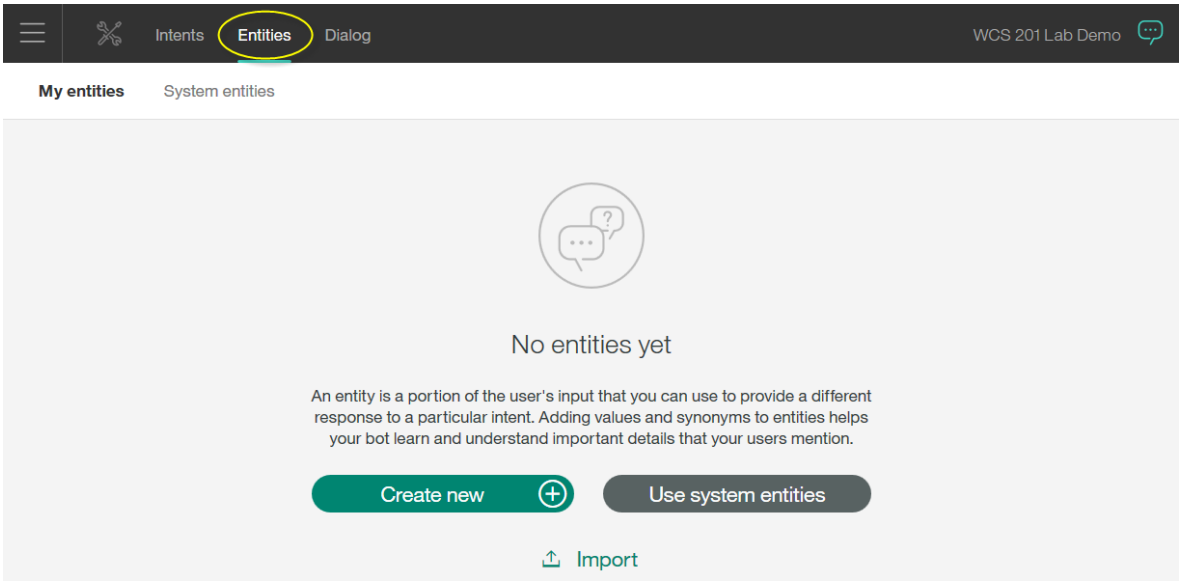
Facts about entities:

- **Entity Name**: 64 character limit: letters (Unicode), numbers, underscores, and hyphens.
  - Cannot begin with the string *sys-* (reserved for system entity names).
- **Entity Value**: 64 character limit: any string
- **Synonyms**: 64 character limit: any string
- Symbol used to refer to entities in dialog nodes: @
  - Do not add the @ symbol when creating a new entity name.
- Entities are not case sensitive.

- If your entity values or synonyms have a space in the name, i.e. alarm clock, then when you refer to your entity in a dialog node, you must enclose the entity name in `()`.
    - Example: `#hotel_amenity:(alarm clock)`
- **Important**: Do not include sensitive or personal information in entity names or values. The names and values can be exposed in URLs in an app.

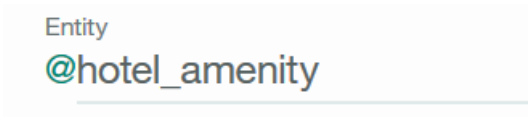
3.3.1 Add Entities Manually

50. Click on the Entities panel and make sure the **My entities** is bold.

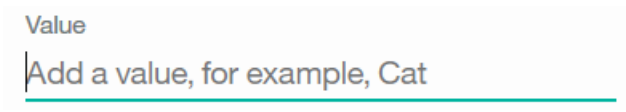


51. Click **Create new**.
52. Click in the **Entity** field.

53. Type the entity name, `hotel_amenity` (after the @).



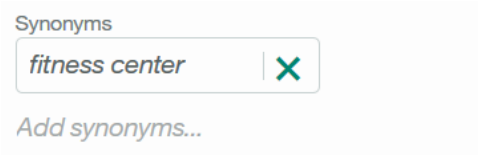
54. Click in the **Value** field.



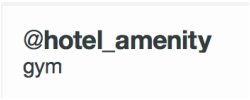
55. Type `gym`.

56. Click in the **Synonyms** field.

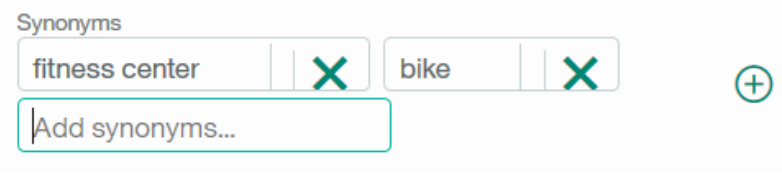
57. Type `fitness center`.



58. Press Enter or click on the **Create** button to the right.
59. If this closes the entity field, click on the name for the entity that you just created.

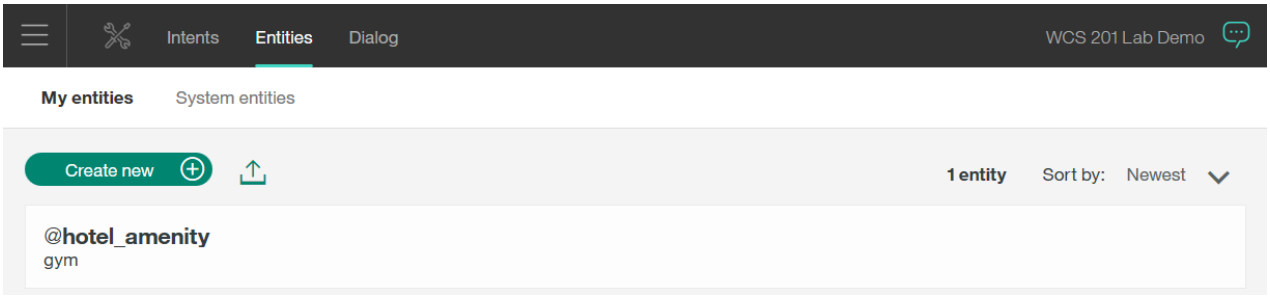


60. Click on the synonym `fitness center` to open the **Add synonyms…** text field.
61. Click in the "Add synonyms" text field, type `bike`, and press Enter on your keyboard.



62. Click out of the field and WCS will automatically save and update training based on your changes.

The Entities screen will now look like this:



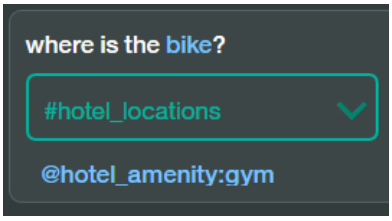
Here are a few more things that you can do to work with entities (from WDC documentation on Bluemix):

- To rename an entity, select the entity and type the new name in the input field.

- To delete an entity, select the entity and then select . Confirm that you want to delete the entity, including its values and synonyms.
- To sort your entities, open the drop-down list and select the method that you want to use: Newest, Oldest, Name (A-Z), or Name (Z-A).

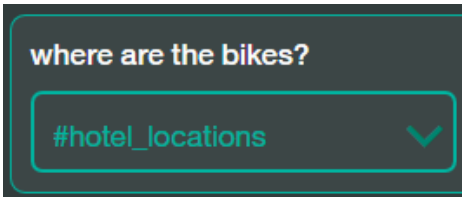
Try it out

63. Open the "Try it out" panel.
64. Wait until Watson is done training.
65. Type `where is the bike?`



Note: Notice that Watson has identified the term *bike* with the correct intent, entity, and entity value.

66. Type `where are the bikes?`



Note: Notice that Watson does not identify *bikes* the same as it identified *bike.* This is because entity matching is brute-force rules-based matching, therefore the *s* added to the end of *bikes* makes it unrecognizable by entity matching.

3.3.2 Enable System Entities

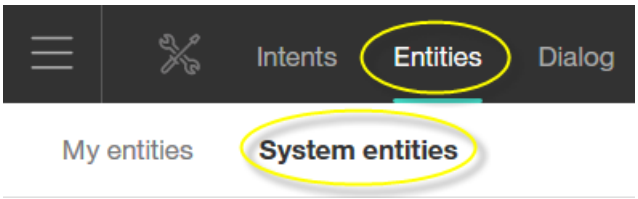
System entities are common entities created by IBM that can be used across any use case. They are ready to be used as soon as you add them to your workspace.

The following system entities are currently supported:

- **@sys-time:** Extracts time mentions, for example, *at 10*.
- **@sys-date:** Extracts date mentions, for example, *Friday*.
- **@sys-currency:** Extracts currency values from user examples including the amount and the unit, for example, *20 cents*.
- **@sys-percentage:** Extracts amounts from user examples including the number and the % sign, for example, *15%*.

- **@sys-number:** Extracts numbers mentioned from user examples as digits or written as numbers, for example, *21*.

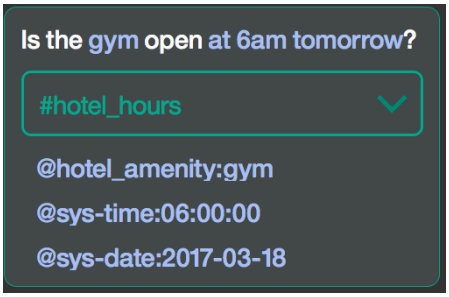
67. Click the Entities panel and "System entities" subpanel.



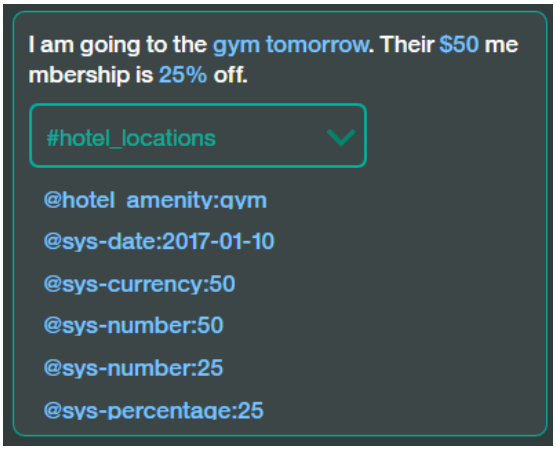
68. Click the **All** switch to turn on support for all system entities.

Try it out

69. Open the "Try it out" panel.
70. Wait until Watson is done training.
71. Type `Is the gym open at 6am tomorrow?`



72. Type `I am going to the gym tomorrow. Their $50 membership is 25% off.`



Note: Notice how these entities are identified. You can use these system entities in your dialog node calculations the same way you would use your personally defined entities.

3.3.3 Import Entities

Importing entities works the same as importing intents, and the same rules apply. Entity import files should use UTF-8 encoding in a CSV file. WCS will not import duplicates of entities that have already been entered and saved in the system.

3.3.3.1 Create the entity import file

Note: The import file has been created for you. This section is for reference and guidance only. Please proceed to the next numbered lab step to proceed with the lab guide exercises.

You may use either a text editor or a spreadsheet application to create your entity file for import.

In your text editor document, entities should be organized in this order:

entity1, value1, synonym1, synonym2, synonym3, synonym4

entity1, value2, synonym1, synonym2

entity2, value1, synonym1

entitiy2, value2, synonym1, synonym2, synonym3, synonym4

In your spreadsheet document, entities should be organized in this order:

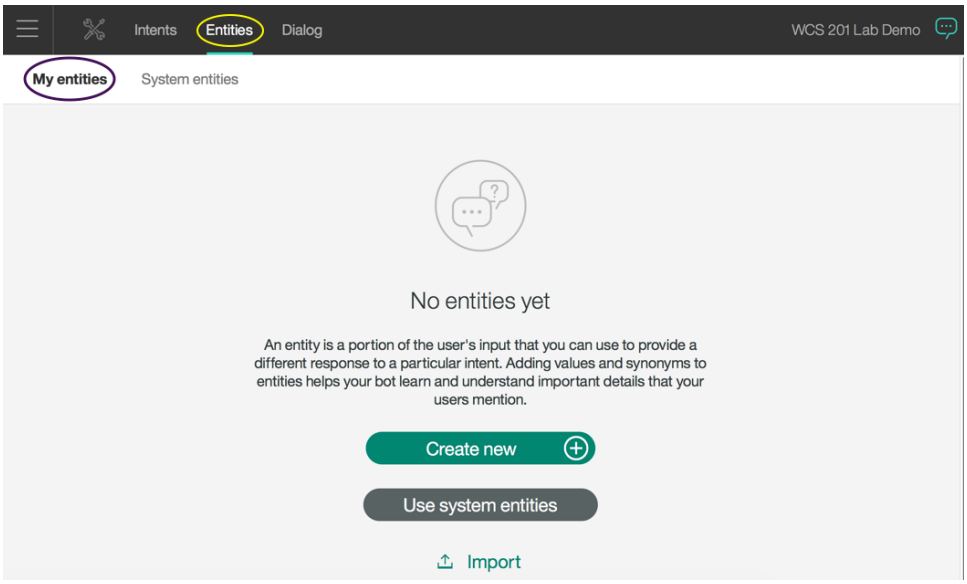
places	breakfast						
places	fast_food	fast food					
places	food_to_go	take away	carry out	to go food	food to go		
places	lunch						
places	chinese	chinese restaurant					
places	coffee_shop	coffee place	coffee shop	dunkin	dunking	starbucks	
places	dinner	supper					
places	italian_restaurant	italian cuisine	italian food	italian place	italian restaurant		
places	japanese	japanese food	japanese restaurant	sushi			
places	pizza	pizza restaurant	pizza pie				
places	sitdown	sit down	eat in	eat-in			
hotel_amenity	gym	bike	elliptical	fitness center	lift weights	weights	
hotel_amenity	hotel restaurant	food place	hotel lounge	hotel restaurant	lounge	your restaurant	
hotel_amenity	pool	lifeguard	swimming	swimming pool			
hotel_amenity	sauna						
transportation	shuttle						
transportation	taxi	cab					
guestroom	alarm_clock	alarm	clock	radio	radio clock	alarm clock	
guestroom	heat_ac	ac	cooler	heater	room temperature	temperature	heat
guestroom	towels	towel	wash cloth	linens	linen		

Note: Documents that are imported require special preparations. When you are finished creating your entity import file, be sure that it is saved using UTF-8 encoding as a CSV file. If you have issues with importing your entity file, it is most likely a problem with UTF-8 encoding. In Windows, it is recommended that you use an application similar to Notepad++ since the default text editor in Windows does not save with UTF-8 encoding. If you are using a spreadsheet application, such as Excel, and you have issues with your import, you can open your file with a text editor to ensure that you have UTF-8 encoding. Mac users can use the Numbers spreadsheet application for UTF-8 encoding.

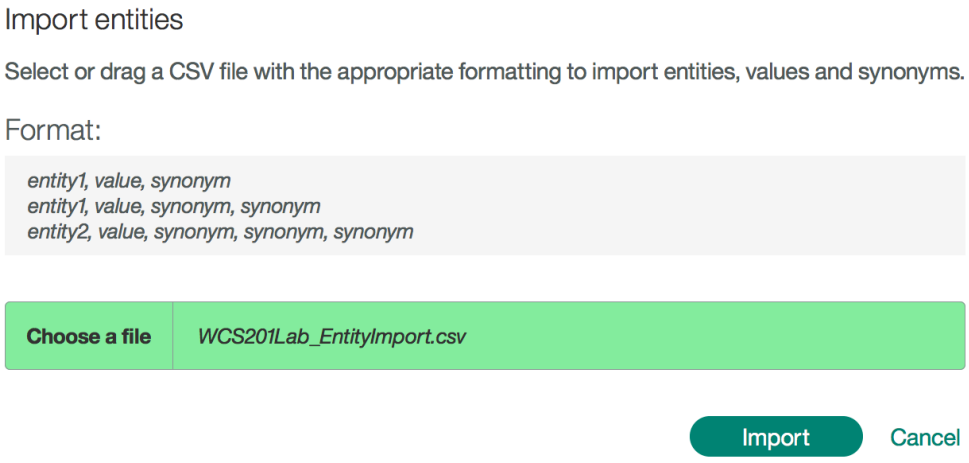
3.3.3.2 Import entities to your workspace

To import an entity CSV file:

73. Open the Entity panel and the “My entities” panel.

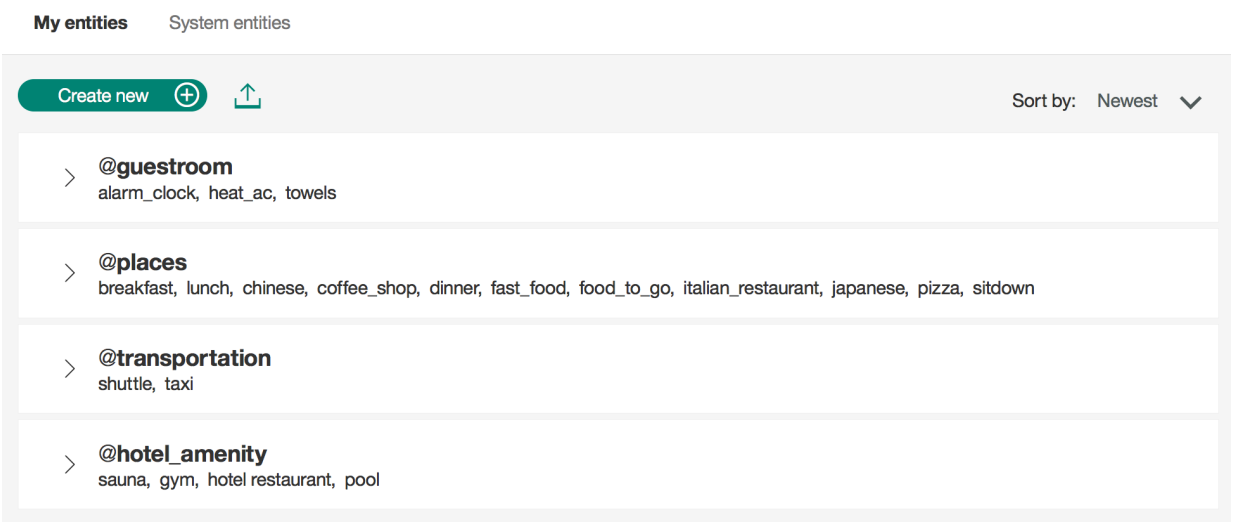


74. Click  .
75. Drag the `WCS201Lab_EntityImport.csv` file into the **Choose a file box** or click **Choose a file** to browse your computer.



76. Click **Import**.

Your entity panel should now look like this with 4 entities:



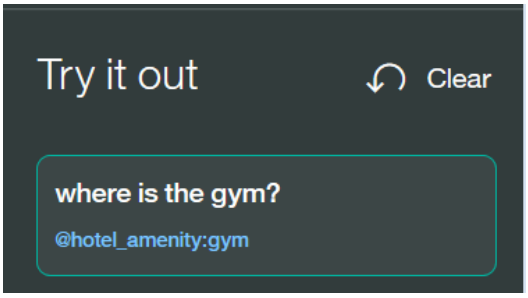
Note: Entities that we entered manually and that were also in the entity import file (duplicates) were not entered twice into the entity field.

3.3.4 Test and Refine Entities

Try it out

77. Open the "Try it out" panel.
78. Wait until Watson is done training.
79. Type `where is the gym?`

WCS should return that your question is matched to the `entityType:value` pair, `@hotel_amenity:gym`.



Note: You cannot update the entity from this screen, but if you feel like there was an error, or more information is needed, click on the Entities tab to update your entities. At the end of this lab guide, once you have your UI app created and connected to your WCS workspace, you will also be able to update entities using the Improve function.

3.4 Dialog

The dialog component of WCS provides responses to users based on the intents and entities identified.

A dialog chat flow is made up of nodes. **Nodes** define the steps in the conversation or chat flow. Dialog nodes are chained together in a tree structure, and each node can be defined by two parts: a condition and a response.

A **condition** is the portion of the dialog node that determines whether the node is used in the conversation. Conditions can be defined using intents, entities, context variables, and special conditions. The Spring Expression (SpEL) language is used to write valid expressions for conditions.

A **response** is activated if the node's condition matches the user input. A response is returned to the end-user and can be either simple text or a more advanced process (such as walking the user through the process of changing their password).

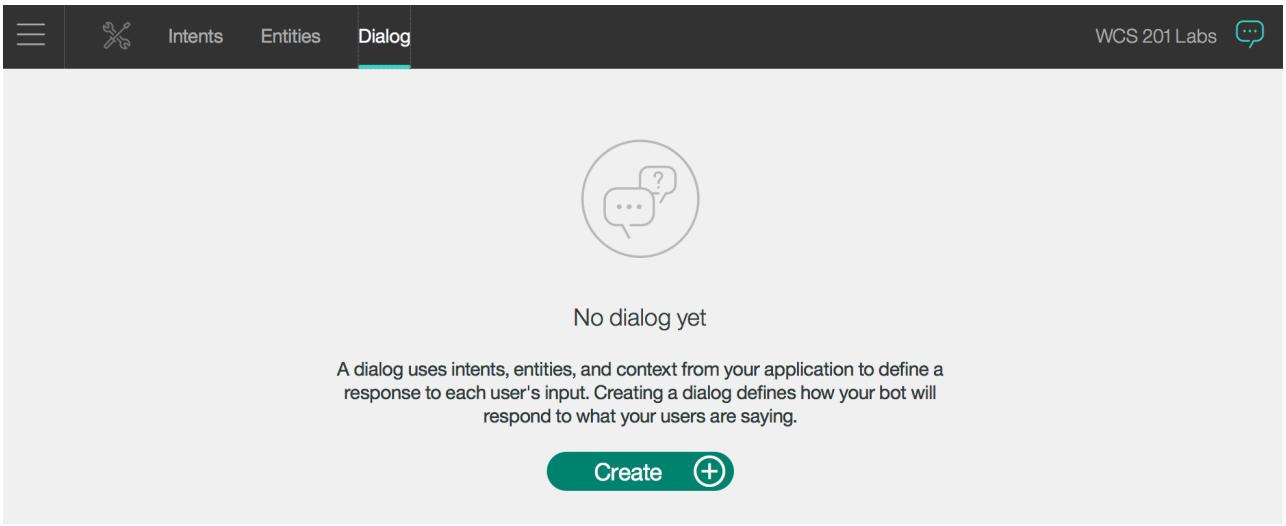
Every node has a node menu. This enables you to choose advanced features or delete the node (which will delete all of the node's child nodes).

3.4.1 Build Dialog Nodes

Remember to review Glenys McLaughlin's course **Designing Conversational Solutions** in Watson Academy to prepare your conversational design before creating your dialog chat flow. Her course includes a lecture with slides for how to present this information to your clients.

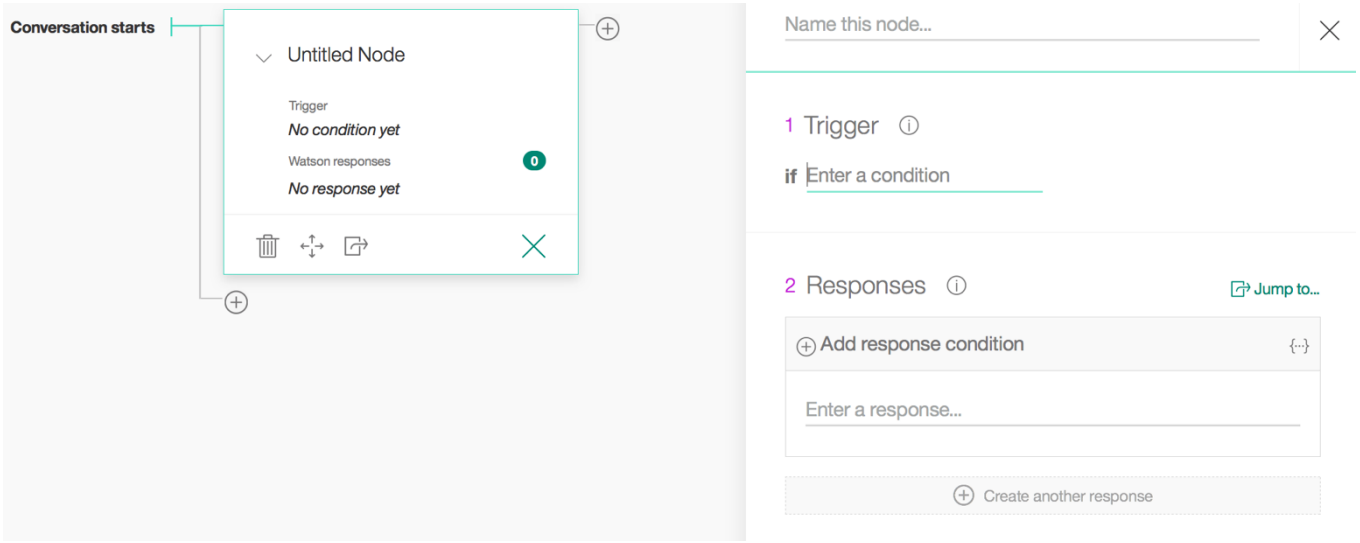
Review the tab in the **Responses v1** worksheet in the `WCS201Lab_CoreIntentsGT.xlsx` document for an example of how to plan your chat flow before you begin working with dialog nodes.

80. Click the Dialog panel.



81. Click on the **Create** button.

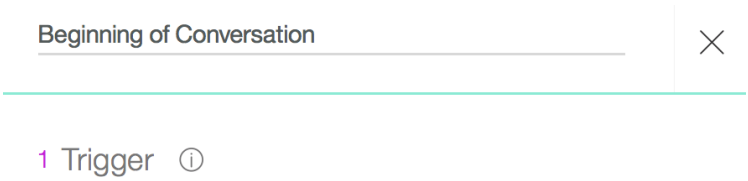
Your screen will look similar to this:



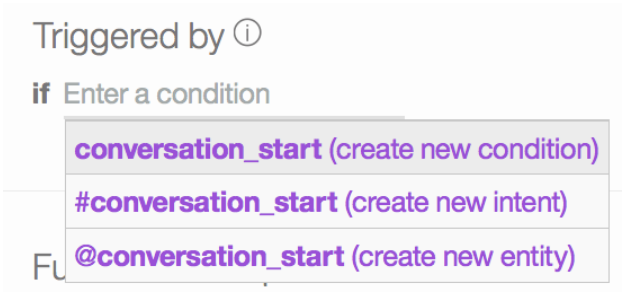
Note: The panel that appears on the right side of the screen is the editing panel for the node. When you create a new node or when you click on a node to make edits, this panel will appear for the node that you click on. This is where all editing is done for the node. The "Trigger" section is where you enter the condition and the "Responses" section is where you will edit the response that WCS will perform if the condition evaluates to true.

In the node editor:

82. Click in the "Name this node…" field and type `Beginning of Conversation`.

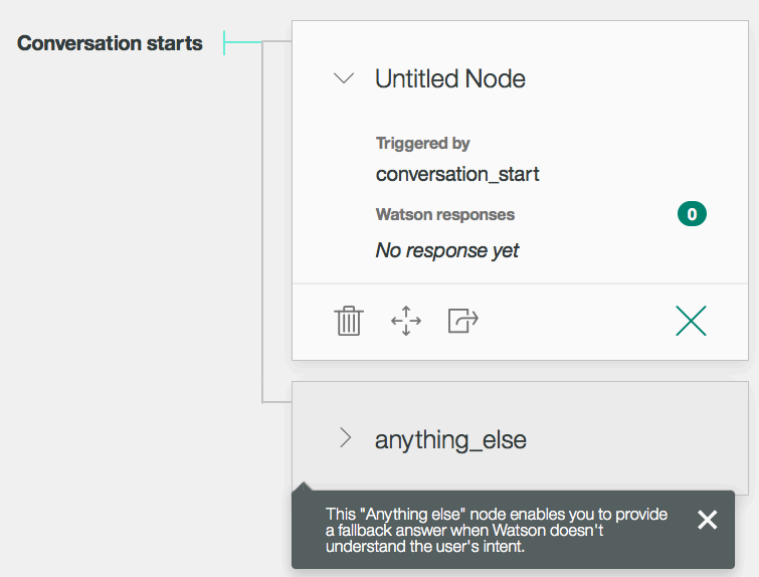


83. Click in the "Trigger" text field to Enter a condition.
84. Type `conversation_start` in the text field and in the drop down box that appears, click on **conversation_start** (create new condition).

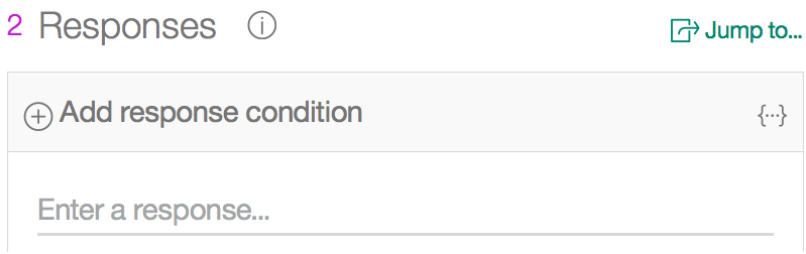


`conversation_start` is a system-defined condition that triggers this node automatically at the beginning of every conversation. Conversation service will only search this node once when the conversation begins with the user. After the first round or turn in the conversation, WCS will not search for this node a second time.

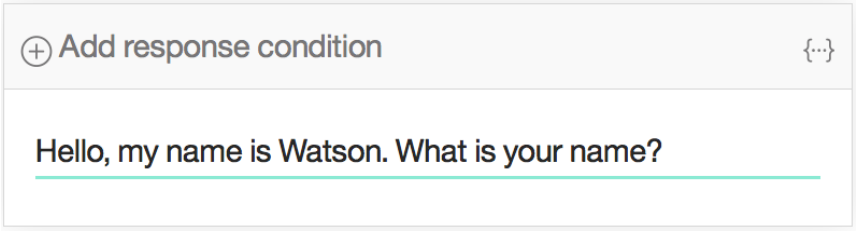
When you create a **conversation_start** node, an **anything_else** node will automatically appear at the end of your dialog tree. The anything_else node is used to provide an last-resort answer when every other parent node in the dialog tree has evaluated to false for an utterance.



85. Click in the **Responses** field under **Add response condition**.



86. Type `Hello, my name is Watson. What is your name?`

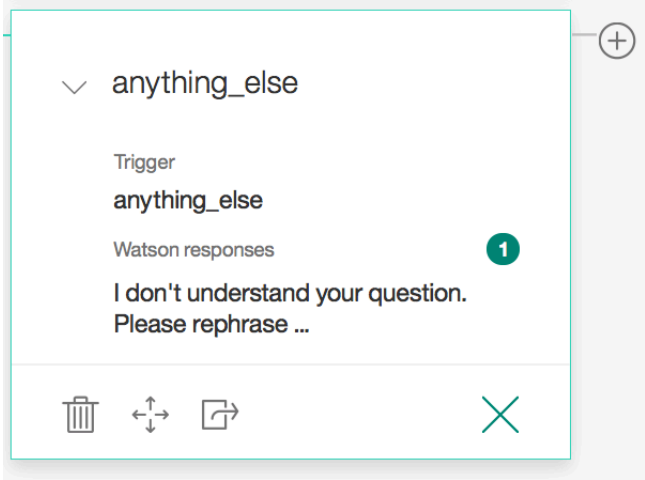


87. Click the **anything_else** node to open it.

88. Name your node `anything_else`.

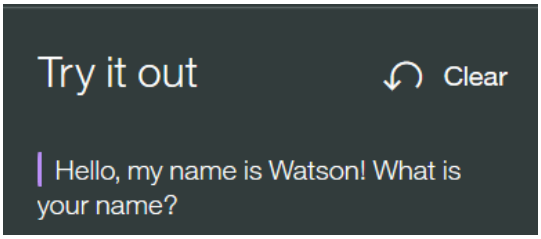
89. Type `I don't understand your question. Please rephrase your request.` in the Responses field.

Your node should now look like this:

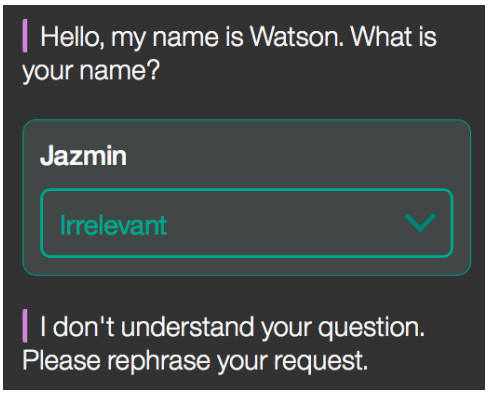


Try it out

Notice that when you open the "Try it out" panel, the conversation now begins with the Hello message you entered into the **conversation_start** node. If it does not, click **Clear** to clear your previous conversations.



90. Type your name.



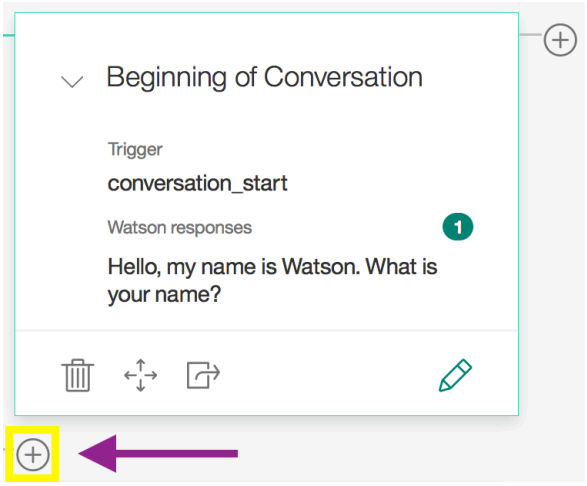
Note: Watson should not recognize your name. We will use context variables in a later section to store the name you enter and use your name in the conversation. There is no need to change the intent for your name at this time.

Pay attention to two things here:

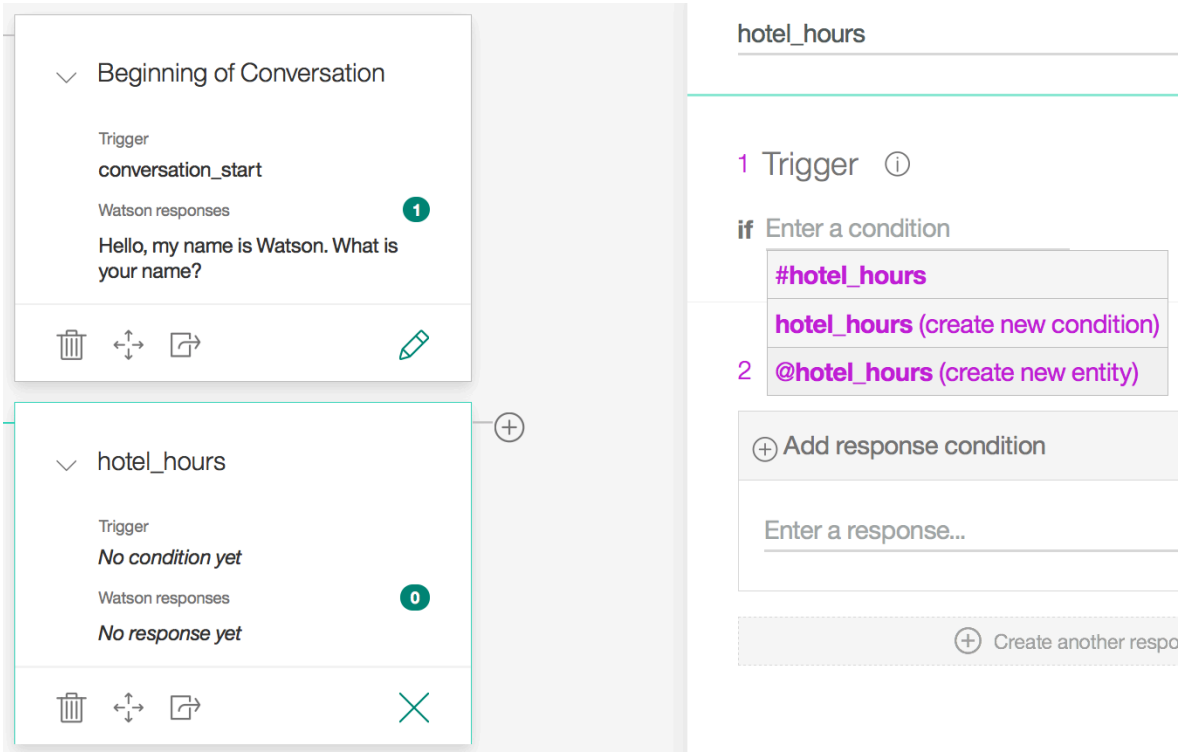
1) Watson paired my input, *Jazmin*, with the Irrelevant intent. The Irrelevant intent is a system intent that can be used to classify end-user examples that are not relevant to the use case. Be careful using this intent, though, because once an end-user example is marked as irrelevant, it is hidden in the system. You will not be able to see it during any cases in which you may want to review conversations and user inputs for ITTC.
2) Watson returned the answer that you typed into the **anything_else** node. This is because there are no other dialog nodes to search. When you entered your name, the only node left in the dialog tree that could evaluate to true is the anything-else node. We need to build dialog nodes with responses for each of the intents. You will do this in later sections of the lab guide.

3.4.2 Build the Chat Flow

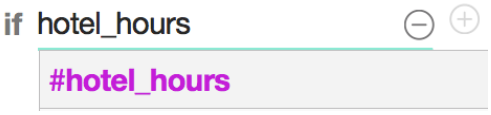
91. Click on the **conversation_start** node to highlight it.
92. Click on the ⊕ at the bottom of the node to create a new parent node.



93. Click in the “Trigger” text field and type `#hotel_hours`



94. Click on **#hotel_hours** in the drop down box. Be sure to select the option with the **#** to choose the intent.

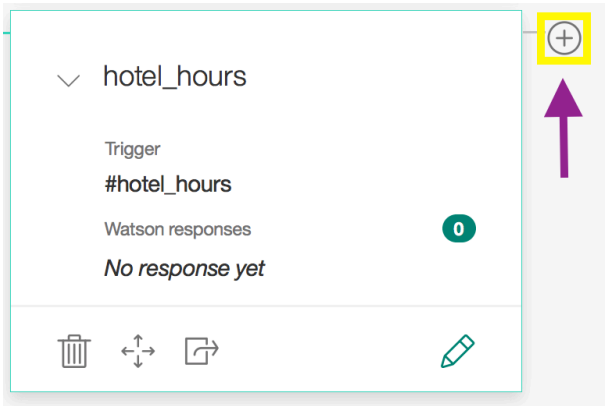


Note: Pushing enter after typing the intent name will not work for this step. You must chose the correct intent from the drop down box.

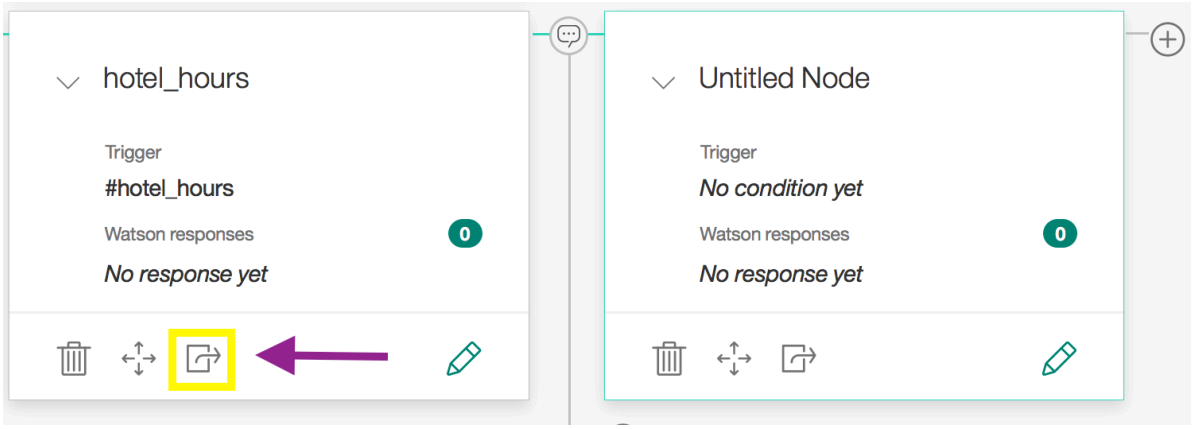
Next, you will create a child node of the **hotel_hours** node by using "Jump To".

95. Click on the **hotel_hours** node to highlight it.

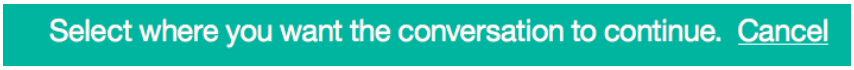
96. Click on the ⊕ to the right of the node to create a child node.



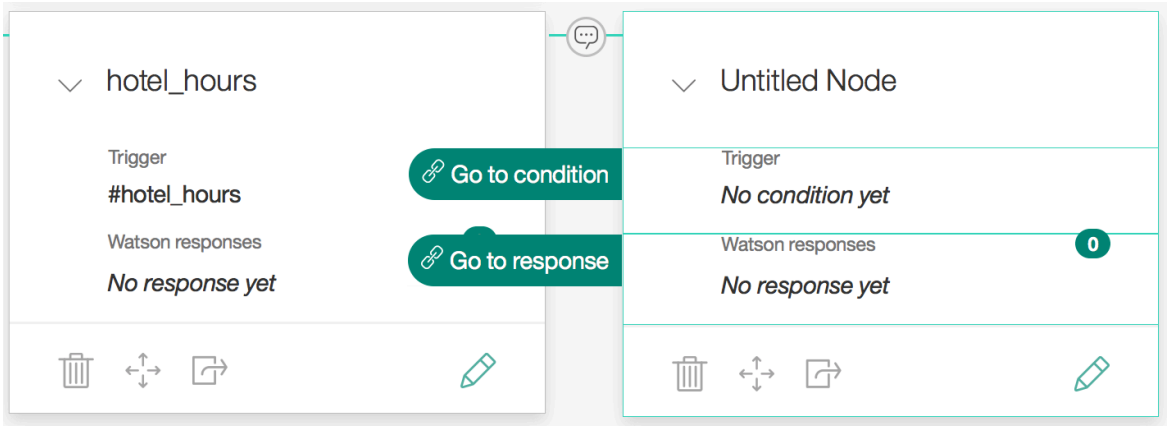
97. Click on the **Jump To** button at the bottom of the hotel_hours node.



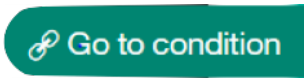
This message will appear at the top of your screen.



98. Click on the blank node to the right of the **#hotel_hours** node. Your node should now look like this:

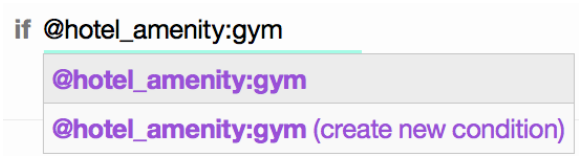


99. Click the **Go to condition** tab that appears to the left of the new node.

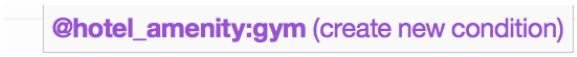


100. Click in the **Name this node…** field and Type `Gym Hours`

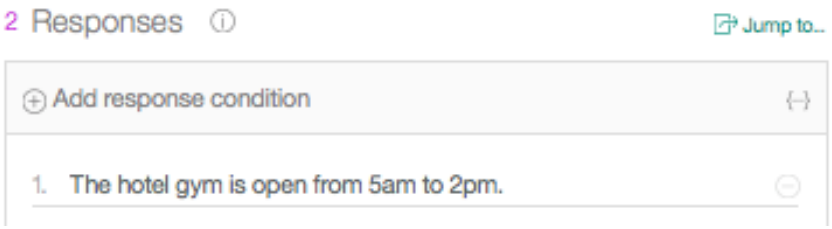
101. Click in the **Enter a condition** field and type `@hotel_amenity:gym`



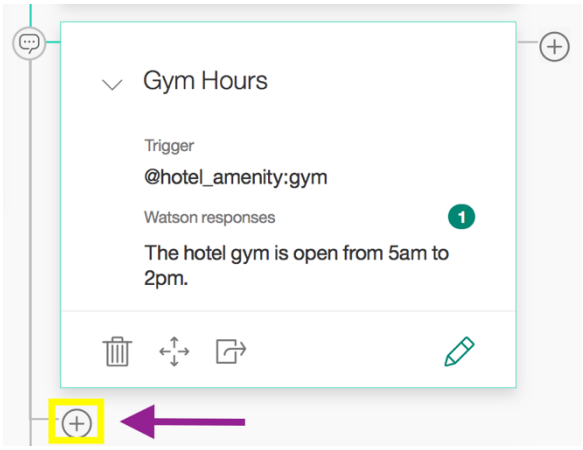
102. Select **@hotel_amenity:gym (create new condition)** in the drop down box.



103. Click in the **Enter a response…** text field and type `The hotel gym is open from 5am to 2pm.`



104. Click the ⊕ on the bottom of the **Gym Hours** node.



105. Continue the previous 4 steps to create 3 more nodes.
    1. Node Name: Pool Hours

       Condition: `@hotel_amenity:pool`

       Response: `The hotel pool is open from 5am to 8pm.`
    2. Node Name: Hotel Restaurant Hours

       Condition: `@hotel_amenity:(hotel restaurant)`

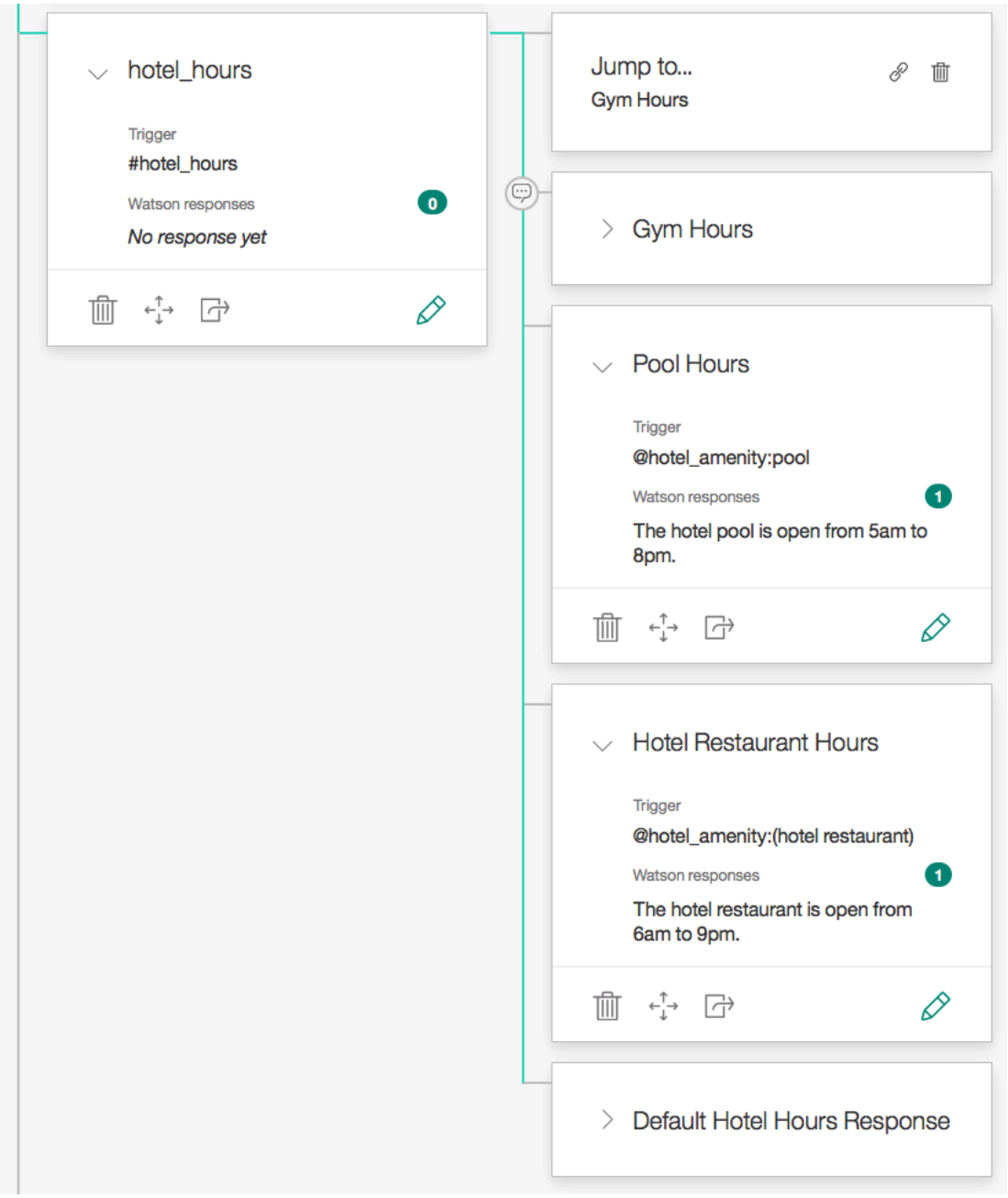
       Response: `The hotel restaurant is open from 6am to 9pm.`
    3. Node Name: Default Hotel Hours Response

       Condition: `true`

       Note: After typing in *true*, make sure to choose **true (create new condition)** from the drop down box.

       Response: `I don't understand your question. Please rephrase your request.`

When you are finished, your chat flow should look like this:



3.4.3 Context Variables ($)

Dialog context is a mechanism for passing information between your bot and the application that submits the user input.

To store information, you will need to modify the context part of the dialog node definition using Advanced Response editor. In the editor, you can define or modify variables that will persist in the dialog context.

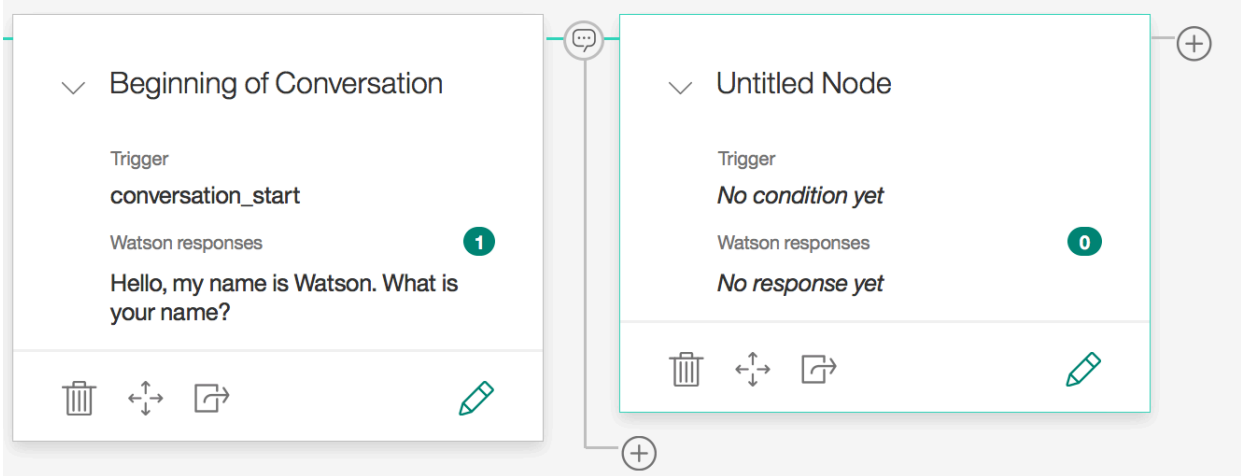
There are two methods to access a context variable: shorthand or full syntax.

- Shorthand: `$variable_name`
- Full Syntax: `context.variable_name`

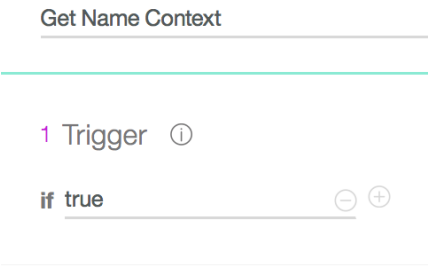
For more details about using context variables, see the Context Variables section of the WCS documentation on the Watson Developer Cloud website
`www.ibm.com/watson/developercloud/conversation.html`

106. Click the **Beginning of Conversation (conversation_start)** node to highlight it.

107. Click the ⊕ to the right of the node to create a new child node of the **Beginning of Conversation** node.

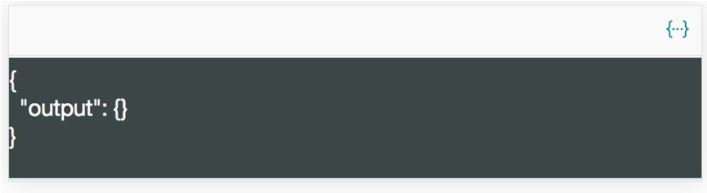


108. Name this node `Get Name Context` and Type `true` in the **Enter a condition** text field and select **true (create new condition)** in the drop down menu.



109. In the **Add response condition** text box, click on the {···} symbol to the right of box.

This will open the advanced editor for entering code.

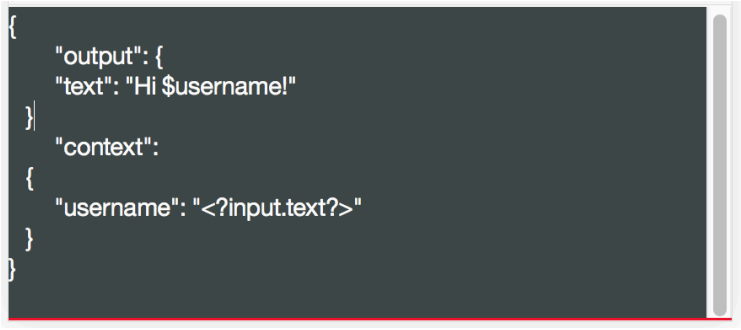


110. Type the code below into the code editor. Your code will automatically save.

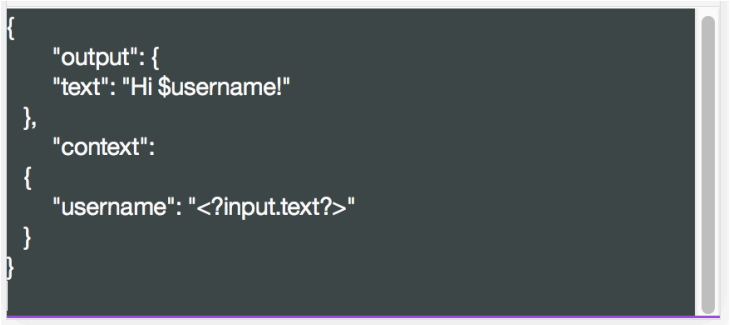
```
{
      "output": {
      "text": "Hi $username!"
   },
      "context":
   {
      "username": "<?input.text?>"
   }
}
```

Note: There is a comma after the first } in this code.

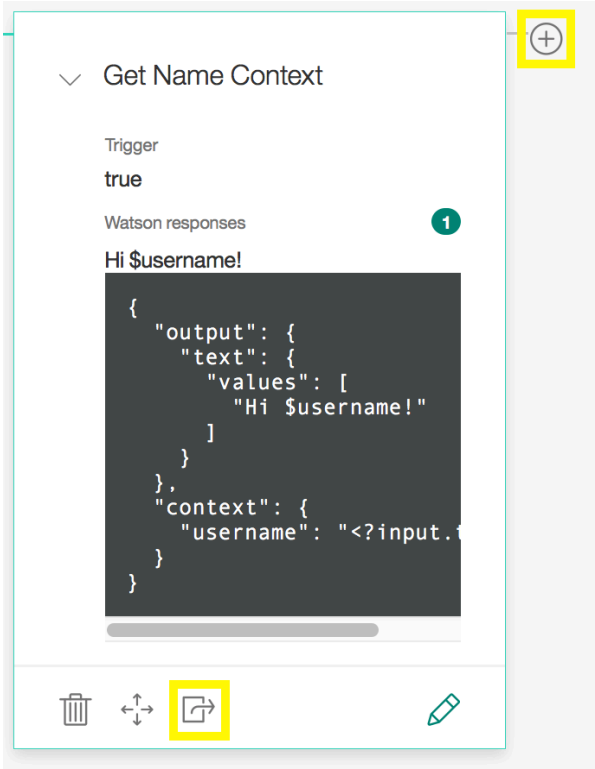
Note: You will get a slight red line at the bottom of the code editor field if your code is typed incorrectly. In this image, the code is missing a comma:



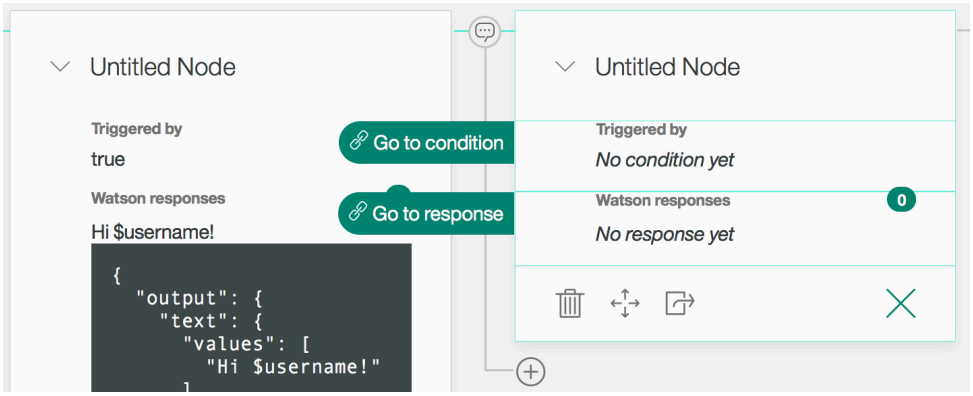
You will get a purple line if the code is correct.



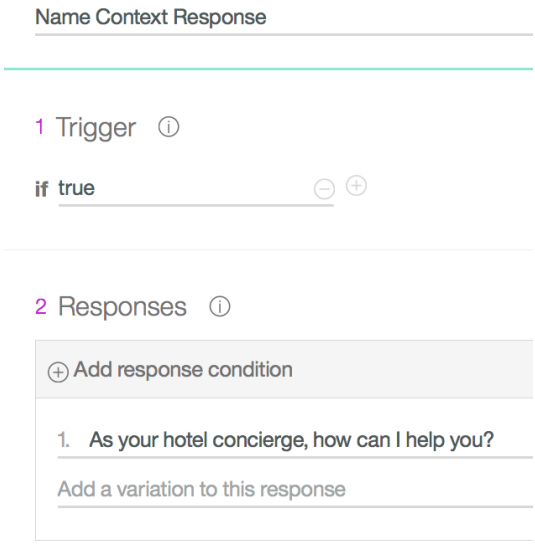
111. Click the ⊕ to the right of the Get Name Context node to create a new node and click the button at the bottom.



112. Click on the new untitled node to highlight it.
113. Select **Go to condition**.

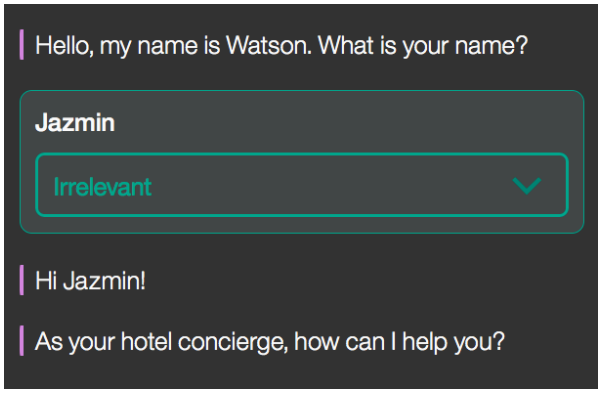


114. Name the node `Name Context Response` and In the **Enter a condition** field, type `true`, and select **true (create new condition)** in the drop down menu.
115. Type `As your hotel concierge, how can I help you?` in the **Add a response** condition text field.



Try it out

116. Open the "Try it out" panel and click Clear if necessary.
117. Wait for Watson to finish training.
118. Type your name.



Note: Notice that Watson is now using the name that you typed in its response to you.

In the image above, WCS recognizes the name as `Irrelevant`. In the full workspace that you will import, name will be recognized as `#name_request`. This is a node that will appear in the full workspace that you will import later in the labs. This node has been added to provide a response in case the end-user types in their name at a later point in the conversation.

3.4.4 Multiple condition Responses

In this lab, you have created nodes that have one response per condition. It is now possible for a single dialog node to have multiple conditions that trigger different responses, called a multiple condition response. This approach enables you to simplify your dialog tree.

For this section of the lab, you will create a new dialog node with a multiple condition response.

Instead of giving you specific step-by-step instructions for this section, you will be given a list of conditions with the paired responses. There is a screen shot of the final dialog node after the instructions.

119. Create a new dialog node after the **hotel_hours** node (this should be a parent node in the main dialog tree, beneath the **hotel_hours** node. When you put your cursor over the ⊕, the text *create alternative conversation* should pop up).

Here are the parameters of your node:

Node name: `hotel_locations`

Trigger condition: `#hotel_locations`

Response 1:

Add response condition: `@hotel_amenity:pool`

Enter a response: `The pool is on the second floor.`

Response 2:

Add response condition: `@hotel_amenity:gym`

Enter a response: `The gym is in the front lobby.`

Response 3:

Add response condition: `@hotel_amenity:(hotel restaurant)`

Enter a response: `The hotel restaurant is located on the ground floor of the hotel.`

Response 4:

Add response condition: `@hotel_amenity:sauna`

Enter a response: `This hotel location does not have a sauna.`

Response 5:

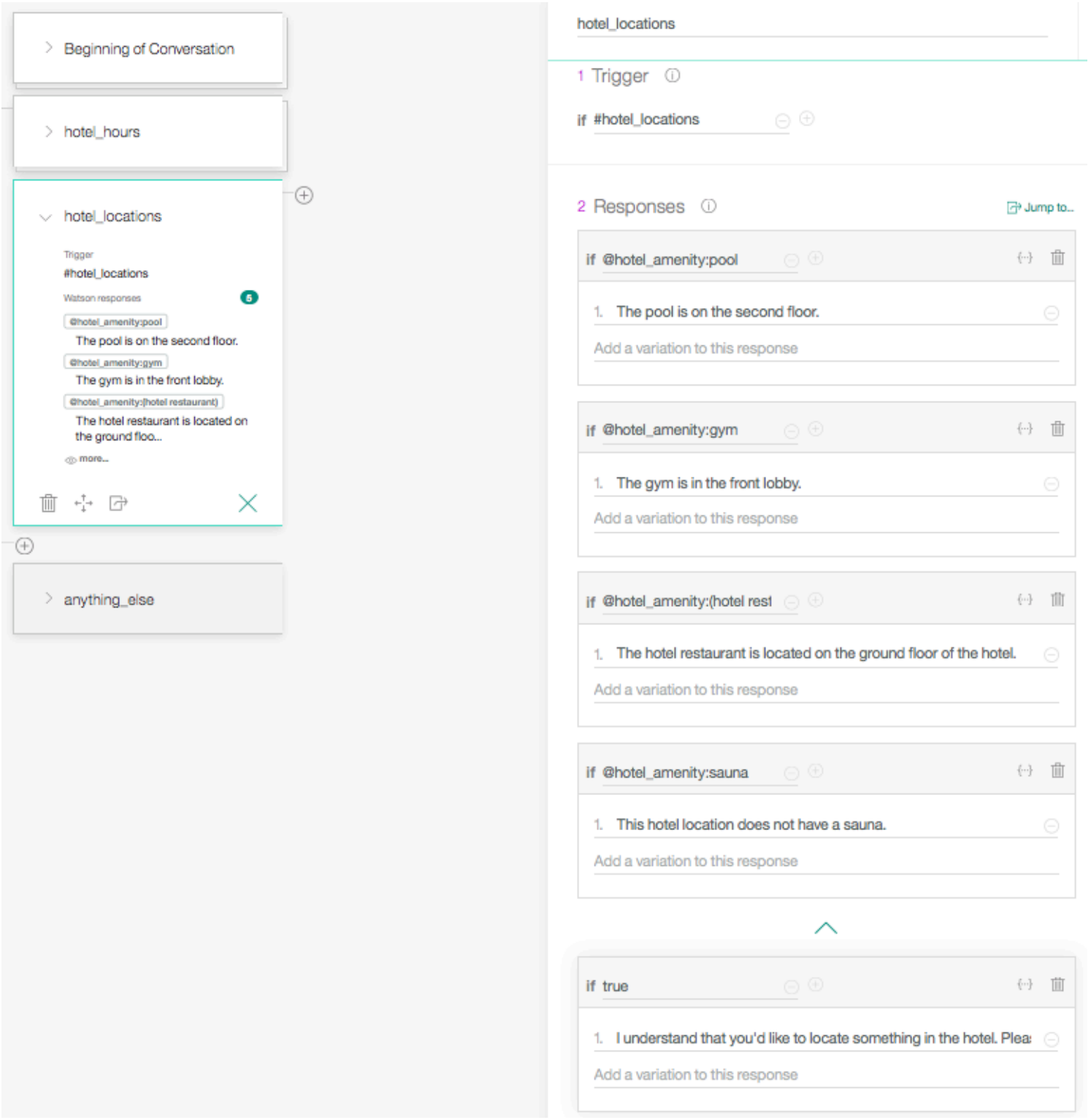
Add response condition: `true`

Enter a response: `I understand that you'd like to locate something in the hotel. Please ask me for the specific amenity you are looking for, or you may call the front desk for directions.`

Note: This true node is used to catch end-user inputs that match to the #hotel_locations intent, but do not match any of the entities used as a condition in this node. Therefore, this answer should reflect that the user's intent was correctly identified with the #hotel_locations intent.

Remember, `true` is a special condition that will always evaluate to true in the search flow.

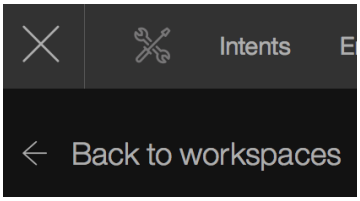
When you are finished, your multiple response node should look like this:



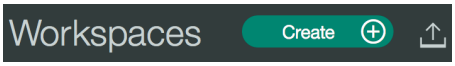
3.4.5 Upload Workspace

At this point in the labs, to save time from rebuilding the entire dialog chat flow that we will use for the rest of the lab guide, you will create a new workspace in your instance of WCS by uploading the `WCS201Lab_Workspacev1.json` file to create a new workspace.

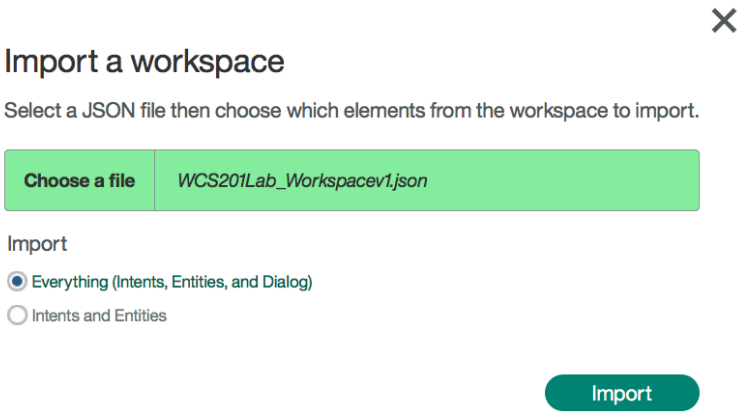
120. Open your instance of WCS (or go Back to Workspaces from the side menu).



121. Click the **import** button, next to the **Create** button.

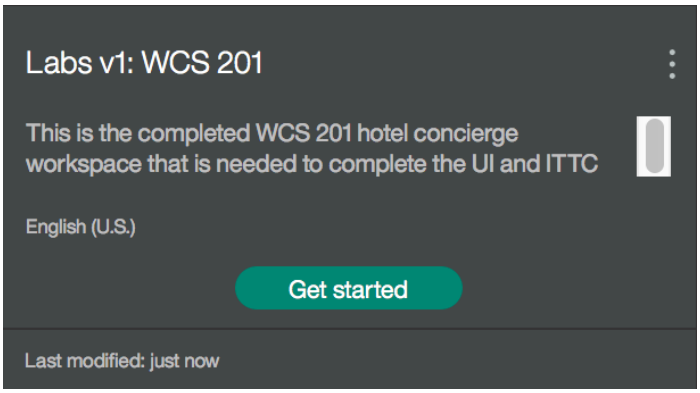


122. Drag the `WCS201Lab_Workspacev1.json` file that you downloaded for this section into the **Choose a file** box or click the **Choose a file** button to browse your computer.



123. Click **Import**.

You will now have a new workspace (although WCS may automatically open the new workspace that you imported). The imported workspace will retain the name of the original workspace **Labs v1: WCS 201**. WCS will allow you to have two workspaces with the same name, so be careful to name workspaces in a way that will not be confusing about which workspace that you are working in.



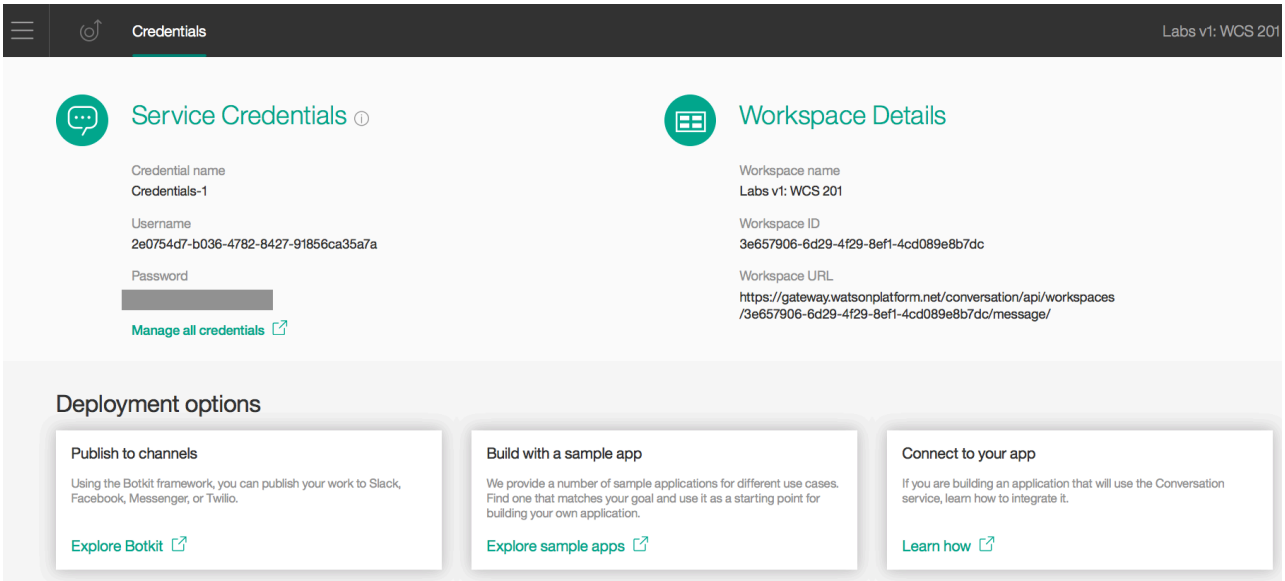
Note: Your new workspace uses the Multiple Condition Response method to create nodes.

3.5 Improve

In December 2016, the Improve panel was updated in WCS, but as of March 2017, the Improve functions only work with the Conversation service is paired with a UI. We will discuss how to use this section in class, after the UI Design section of the lab guide. When this section is updated in the lab guide, it will be made available in the **WCS 201 Alumni** community and updated in the online WCS 201 course information. If you are taking this class as an online course instead of in-person, email Jazmin Capezza, Mike Hervey, or Ray Lopez to be added to the WCS 201 Alumni community.

3.6 Deploy

In March 2017, the Deploy panel was added to Conversation service. As of March, this panel only includes the Credentials tab. The credentials tab gives the Service Credentials and Workspace Details that are needed to connect your Conversation service to other apps and APIs directly within the tooling. The Deployment options section has links to help with each of the topics given. In the future, this section will offer more functionality, including the ability to integrate with other services, such as Slack.



4 Activity 2: UI Design

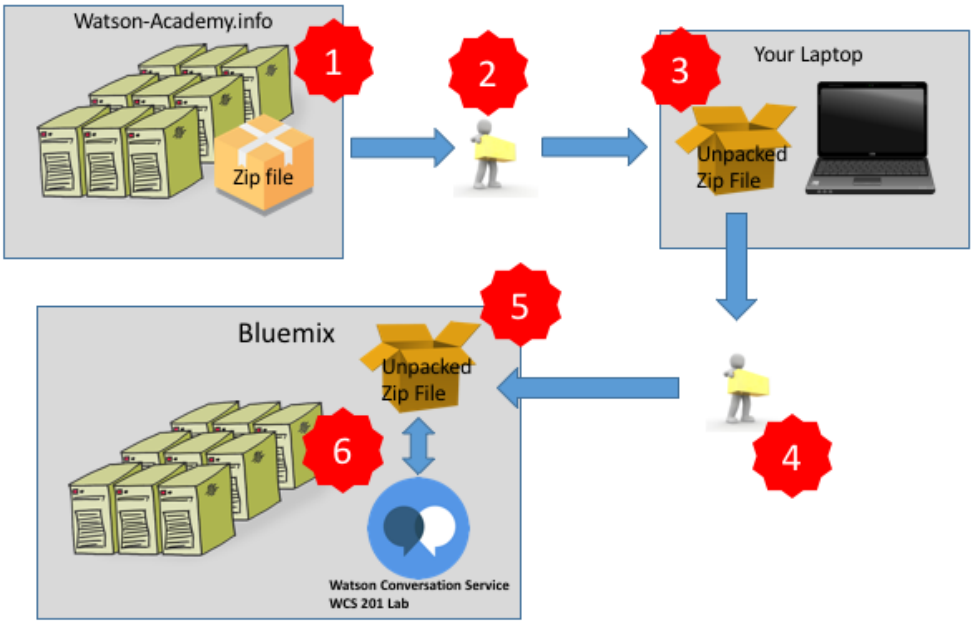
In this section, you will learn how to deploy a user interface application for WCS. The UI is necessary for testing your dialog build and for seeing confidence levels that are returned for each intent.

The steps for this lab exercise are:

1. Download the user interface code package.
2. Unpack the user interface code package, modify the `manifest.yml` file in the code package.
3. Push the modified code package to Bluemix.
4. Code package is automatically deployed to Bluemix
5. Code is executed and automatically bound to the Watson Conversation service instance.

When you complete this exercise, you will have a functional conversation application that uses the WCS workspace that you imported in the last section.

The figure below summarizes the main activities in this lab section.



Summary of the main steps involved in the current lab exercise:

1. The zip file is downloaded to your local computer. (This file is located in the online course lab materials and on GitHub. The GitHub link is included in the online course and it is linked in the Practitioner Assistant community).
2. The zip file is unpacked and you use a text editor to modify the file `manifest.yml`
3. All of the files are pushed to Bluemix using the cf command line tool.
4. The code is unpacked and installed to Bluemix.
5. The code is bound to the workspace in WCS and started.

4.1 Installing the cf command line tool on Mac OSX

Before you begin to work with the code, you will need to install the cf command line tool on your computer. Please note the instructions below are different based on your operating system. Please use the appropriate section for your system.

The cf command line tool is used to work with IBM Bluemix from the command line and is the preferred method for working with Bluemix.

The downloads needed for this section are located both in the Lab Resources section of the online course or in the zipped `WCS201_AllLabDocs.zip` file.

For more information on command line interface and dev tools, including updates to the cf installer packages used in this guide, refer to the IBM Bluemix docs at:
`https://console.ng.bluemix.net/docs/cli/index.html#cli`

4.1.1 [Mac] Download Installer Package

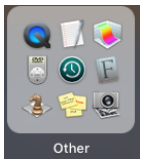
The installer package that you will need for Mac is `cf-cli-installer_6.22.1_osx.pkg`

4.1.2 [Mac OSX] Install the cf application

124. [Mac] Locate the installer file `cf-cli-installer_6.22.1_osx.pkg`
125. [Mac] Double click on the file to begin the installation.
126. [Mac] Click **Continue** in the Install Cloud Foundry CLI window to begin the installation process.
127. [Mac] Click through the instructions in the installation wizard to complete installation.

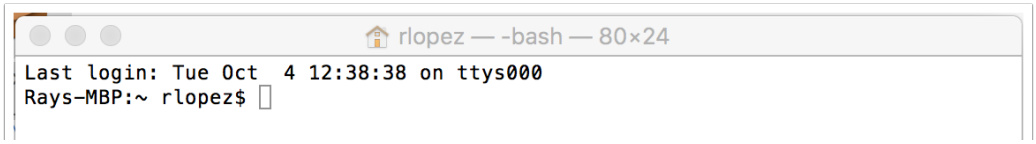
Next, you will open a terminal window.

128. [Mac] Open your **Launch Pad** 
129. [Mac] Click the **Other** container in the Launchpad view



130. [Mac] Click the Terminal icon  .

A terminal similar to the one shown below should now be on your screen:



4.1.3 [Mac] Test the cf tool to ensure it is properly installed.

131. [Mac] Click inside the terminal window.
132. [Mac] Type the following command into the terminal window: `cf -v`

133. [Mac] Press Enter.

You should see the following output (or something similar) displayed on your terminal screen:

```
cf version 6.22.1+6b7af9c-2016-09-24
```

You are now ready to start working with the UI code. Proceed to the section: Working with the UI.

4.2 Installing the cf command line tool on Windows

In this step, you will go through the process of installing the `cf command line tool`. This tool is used to work with IBM Bluemix directly from the command line and is the most preferred method for working with Bluemix.

The downloads needed for this section are located both in the Lab Resources section of the online course or in the zipped `WCS201_AllLabDocs.zip` file.

4.2.1 [Windows] Download Installer Package

The installer package that you will need for Windows is `cf-cli-installer_6.22.1_winx64.zip`

Save the downloaded file to your hard drive in a place that you can remember. The instructions below assume that you will save the file to your **Downloads** folder.

4.2.2 [Windows] Install the cf application

134. [Windows] Locate the downloaded installer file `cf-cli-installer_6.22.1_winx64.zip`

135. [Windows] Double click on the icon to open the archive file.

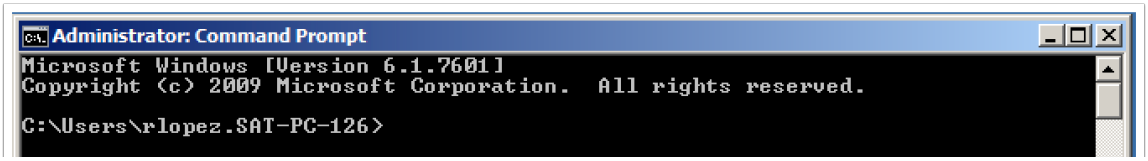
136. [Windows] Double click on `cf_installer.exe` to start the installation.

137. [Windows] Click through the instructions in the installation wizard to complete installation.

138. [Windows] Open a command prompt window  .

139. [Windows] Click on **Start → All Programs → Accessories →Command Prompt**

The Windows Command Prompt will appear in a window similar to the image below.



4.2.3 [Windows] Test the cf tool to ensure it is properly installed.

140. [Windows] Click your mouse cursor inside the Command Prompt window.

141. [Windows] Type the command in the command prompt: `cf -v`

142. [Windows] Press Enter.

You should see the following output (or something similar) displayed on your terminal screen:

```
cf version 6.22.1+6b7af9c-2016-09-24
```

You are now ready to start working with the UI code. Proceed to the section: Working with the UI.

4.3 Working with the UI Code

In this step, you will go through the process of obtaining the UI code. This is the code you will upload into Bluemix using the `cf command line tool`. The downloads needed for this section are located both in the Lab Resources section of the online course or in the zipped `WCS201_AllLabDocs.zip` file.

Please note that the instructions below are similar for both Windows and Mac.

To access the credentials for this section, you will need to sign in to your Conversation service. You can either sign in through Bluemix or you can access your Conversation instances directly using `https://www.watsonconversation.com`

4.3.1 Locate and Download the file containing the code.

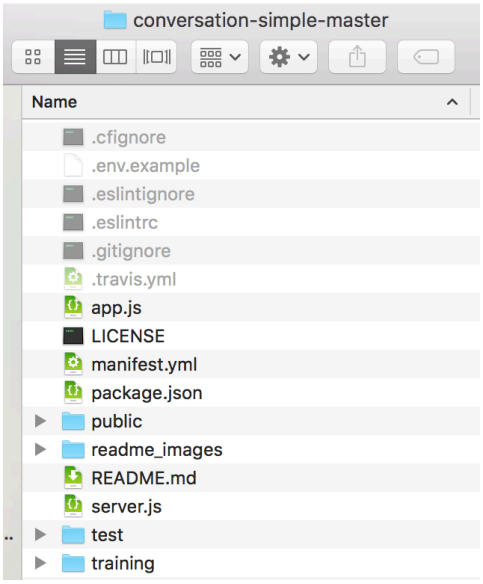
The zip file that you will need for both Mac and Windows is `conversation-simple-master.zip`

Save the downloaded file to your hard drive in a place that you can remember. The instructions below assume that you will save the file to your **Downloads** folder.

4.3.2 Unpack the user interface code package

Locate `conversation-simple-master.zip`. Use the Windows Explorer or Mac Finder to extract the contents of the zip archive (this can usually be done by right-clicking on the icon for the archive file).

When you open the folder containing the extracted files, you should see the files shown in the screen shot below.



Make a note of the full path name of the directory in which these files are stored. Do not remove the `manifest.yml` file from this folder. Your code will not push if the manifest file is not located in the **conversation-simple-master** folder.

On Windows, the path is likely (unless you downloaded to your desktop, then use *desktop* instead of *downloads*): `Downloads\conversation-simple-master`

On Mac, the path will probably be: `Downloads/conversation-simple-master`

4.3.3 Modify the `manifest.yml` file

In the directory you verified in the previous step, you should see a file named `manifest.yml`.

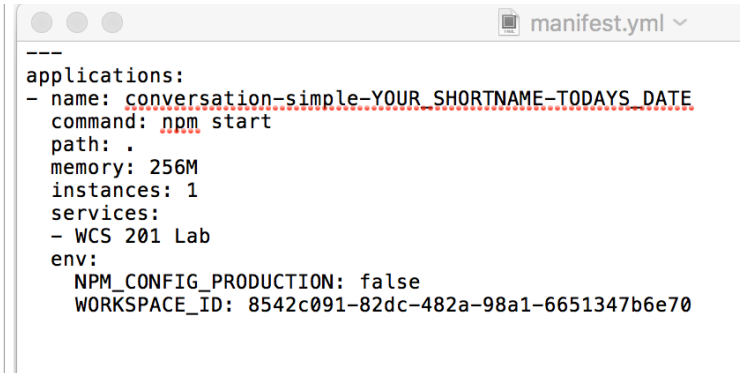
143. Use a text editor on your computer to open this file.

    Suggested text editors:

    `[Mac]` TextEdit

    `[Windows]` Notepad or Notepad++.

144. Open the `manifest.yml` file with either TextEdit or Notepad++. You should see the file contents similar to the screenshot below:



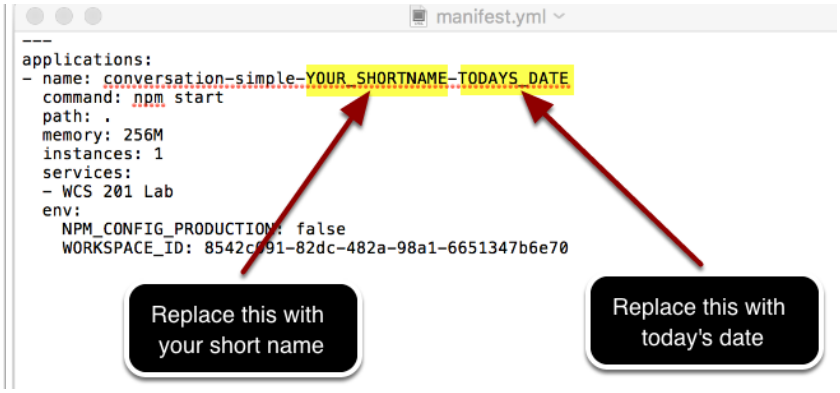
145. Modify the line `- name conversation-simple-YOUR_SHORTNAME-TODAYS_DATE` by replacing `YOUR_SHORTNAME` with your Lotus Notes short name and replacing `TODAYS_DATE` with today's date.

For example (if you were Gini Rometty editing the file on 10 October 2016):

This line: `- name conversation-simple-YOUR_SHORTNAME-TODAYS_DATE`

Might become this: `- name conversation-simple-grometty-2016oct10`

Note: Do not put spaces in your short name or in the date.



146. Save the file. You are now ready to push your code to Bluemix.

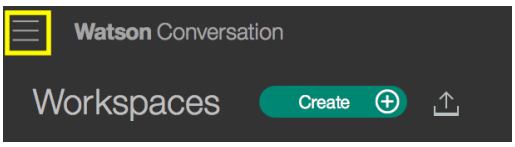
Next, you will change the service name and workspace ID in the `manifest.yml` file. This is how you will point your UI code to the correct WCS instance and to the correct workspace inside the WCS instance.

Under **Services**, replace the text `-REPLACE_WITH_NAME_OF_WCS_INSTANCE` with the name of the WCS instance that you wish to bind to your UI code. In our example, we are binding our code to the WCS 201 Lab instance. There are two way to get the name of your WCS instance, from the Bluemix dashboard or from the Watson Conversation service menu.

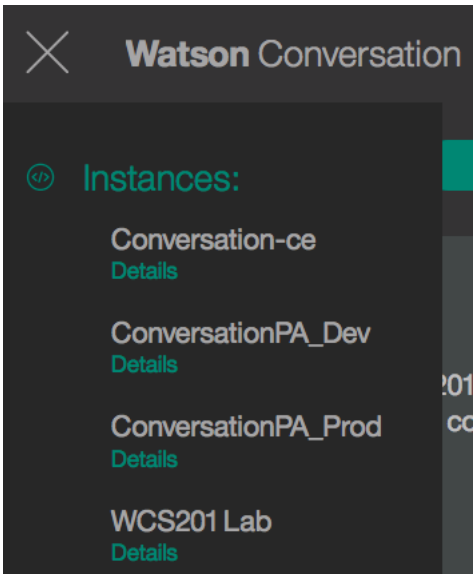
To get your service name from Bluemix, go to your dashboard and look under **All Services**. For example, the name of a WCS instance `WCS 201 Lab` listed in a Bluemix dashboard, under **All Services** is shown below.

Note: The name of your Conversation instance may be different than the image.

If you have your Conversation service tooling open, you can view your instance name in the Watson Conversation menu bar to the left of the UI.



Note: In this menu image, the user has 4 Instances of the Conversation service.

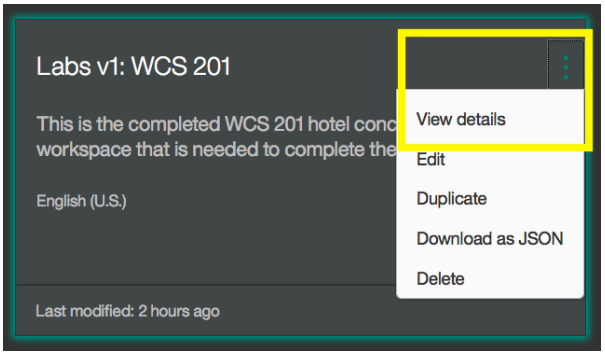


147. Return to your text editor and paste the service name into the `manifest.yml` as shown below:

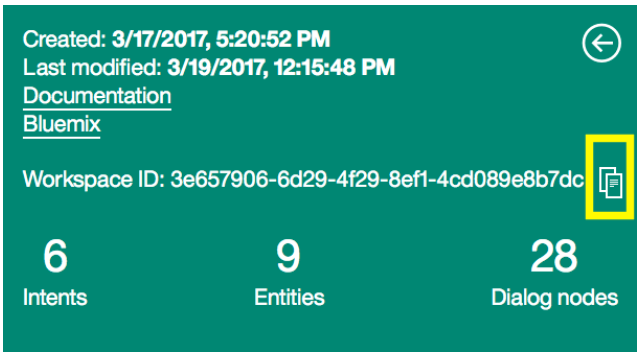
Next, you will access your service credentials to get your workspace ID.

You will be using the workspace that you uploaded into your instance of Conversation in section 3 of the lab guide. The name of this workspace is `Labs v1: WCS 201`. Your service credentials are located in two places, in the menu for the workspace and in the Deploy panel, under the Credentials tab. Instructions are given for accessing your workspace ID using the workspace menu.

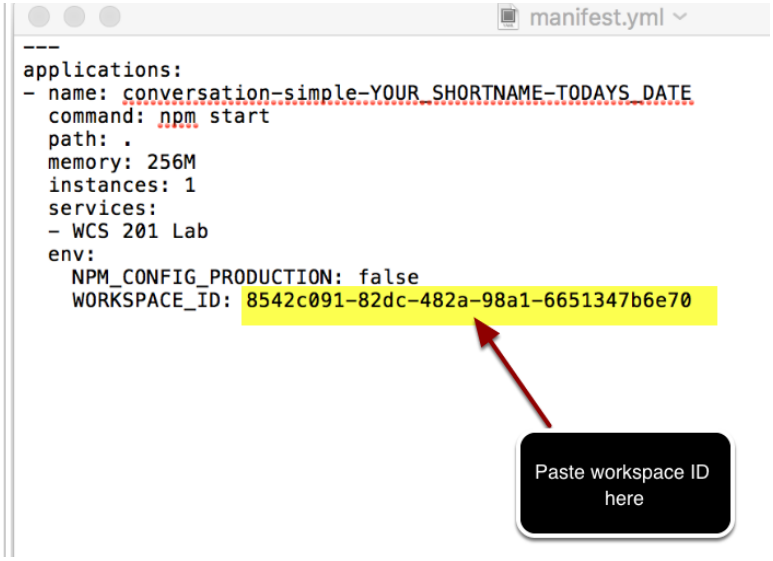
148. For the `Labsv1: WCS 201` workspace, click the menu icon in the upper right corner (three dots) and select **View Details** as shown below:



149. Click on the copy icon next to the workspace ID to copy the workspace ID to your clipboard as shown below:



150. Return to your text editor and paste the workspace ID into the `manifest.yml` as shown below. Afterwards, save the file. You have now finished modifying the `manifest.yml` file.



Note: Your workspace ID will be different than the one shown above.

4.3.4 Push the modified code package to Bluemix

Open a terminal window on your Mac or a command prompt window on Windows. If you don't remember how to do this, see the instructions you followed above for `[Mac]` "Open a Terminal Window" or `[Windows]`"Open a Command Prompt Window"

Once you have opened a terminal or command prompt window, type the following commands into the window:

151. `[Windows]` Type `cd Downloads\conversation-simple-master`

152. `[Mac]` Type `cd Downloads/conversation-simple-master`

```
Jazmins-MacBook-Pro:~ jlcapezz$ cd desktop/conversation-simple-master
```

153. `[BOTH]` Press Enter.

You should get the command prompt immediately and no error should be indicated. If you do see an error, it means that you need to locate the directory where the code is stored. You must be in the `conversation-simple-master` directory and your `manifest.yml` file must be in the `conversation-simple-master` directory to proceed with the rest of this section.

For the next step, you will connect to a data center, i.e. Endpoint.

`<ENDPOINT>` will be one of the following, depending on your geography:

US South: `https://api.ng.bluemix.net`

Sydney: `https://api.au-syd.bluemix.net`

UK: `https://api.eu-gb.bluemix.net`

154. Type `cf api <ENDPOINT>`

Example: If you are connecting to the US South, use the command:

```
cf api https://api.ng.bluemix.net
```

155. Press Enter.

The computer will respond with something similar to below:

```
Setting api endpoint to https://api.ng.bluemix.net...
OK
API endpoint:   https://api.ng.bluemix.net (API version: 2.54.0)
User:           ray.lopez@us.ibm.com
Org:            WatsonU_Labs_WCS
Space:          WCS 201 Labs
```

If you are prompted to sign in to Bluemix:

156. Type `cf login -u you@xx.ibm.com` where `you@xx.ibm.com` is your email address you have registered as your login id for Bluemix.

157. Press Enter. The computer will respond with something similar to below:

```
API endpoint: https://api.ng.bluemix.net
Password>
```

158. Type your Bluemix password at the prompt then press Enter. The computer may respond with something similar to below:

```
Authenticating...
OK
Select an org (or press enter to skip):
1. My Bluemix Org
2. WatsonU_Labs_WCS
Org>
```

You may not see an `Org>` prompt! If you do not, then skip the steps on selecting an org.

159. If you see the Org prompt as shown above, select the number corresponding to the Bluemix org where your Conversation instance is running.

160. Type the number for the option then press Enter.

You may then see output like this:

```
Targeted org WatsonU_Labs_WCS
Targeted space WCS 201 Labs
API endpoint: https://api.ng.bluemix.net (API version: 2.54.0)
```

```
User:           ray.lopez@us.ibm.com
Org:            WatsonU_Labs_WCS
Space:          WCS 201 Labs
```

If you see something similar to the above output, you have successfully logged into Bluemix with the cf command line tool.

161. Type `cf push` into the command window then press Enter.

Note: If you are asked to type in the app name, please ask your instructor for assistance.

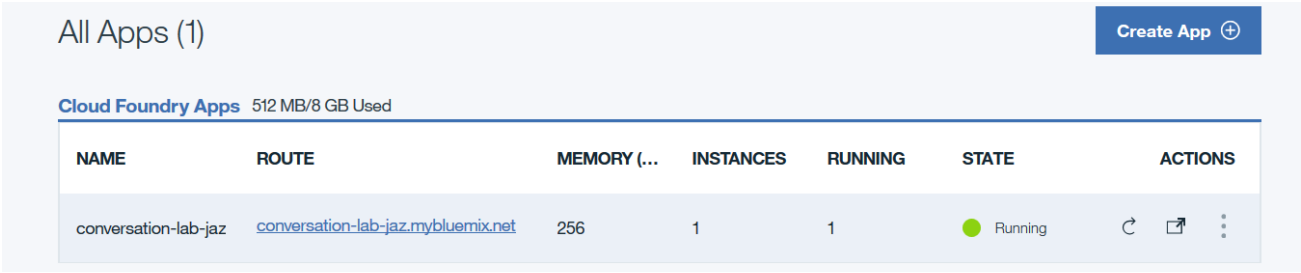
You will see a very long stream of text appear on your screen as the cf tool pushes your code to Bluemix. This process may take anywhere from 1 to 10 minutes. Once this process completes, the code has been installed and executed on Bluemix, and it will be using the Watson Conversation Service you indicated in the `manifest.yml` file.

4.3.5 Verifying your new app on Bluemix

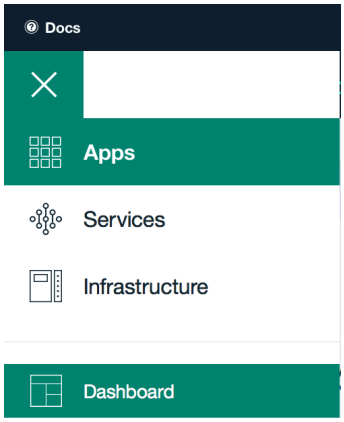
In this section, you will check the status of your application in Bluemix. You will also use your Web browser to test your new app.

162. Log into Bluemix.

After logging in, you should see your dashboard. You should see your new app under Cloud Foundry Apps.

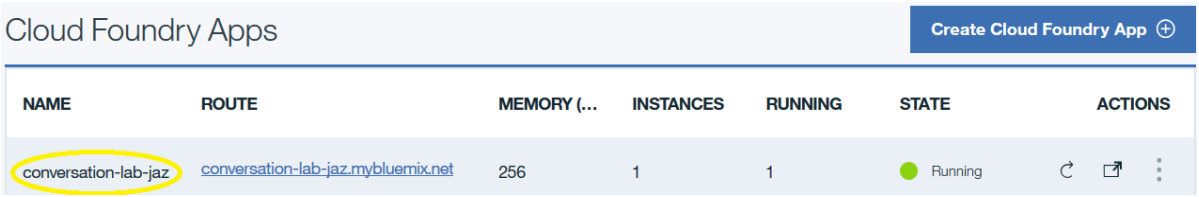


Note: If you do not see your dashboard, click the menu icon on the left of the title bar to navigate to Cloud Foundry Apps.

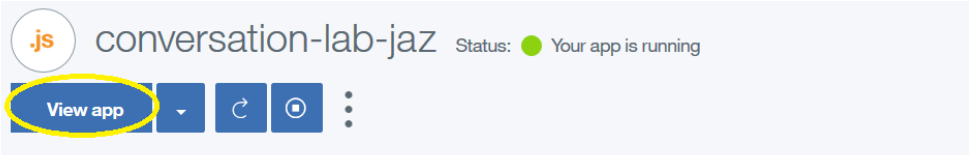


To check the status of your app:

163. Click on the name of your app to view it (under Name, not the URL highlighted in blue. The name of this app is LabUI).

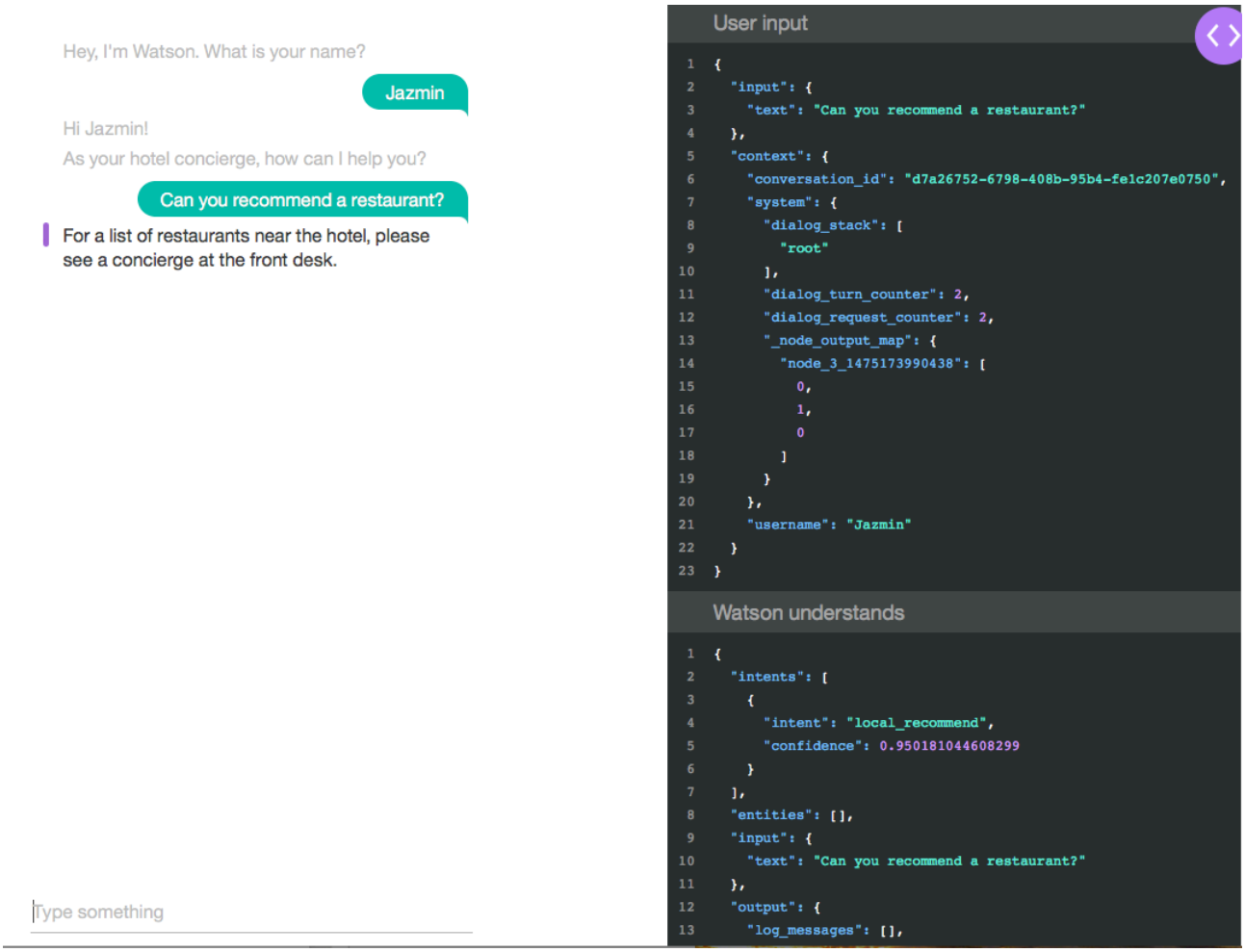


164. Click on View App. This will open a Web page for accessing your conversation application.



To test your conversation app:

165. Open your conversation app.
166. Type your name (If you are using the **Labsv1: WCS 201** workspace, you will be prompted for your name).
167. Type `Can you recommend a restaurant?`


Hey, I'm Watson. What is your name?
Jazmin
Hi Jazmin!
As your hotel concierge, how can I help you?
Can you recommend a restaurant?
For a list of restaurants near the hotel, please see a concierge at the front desk.
Type something
User input
{
"input": {
"text": "Can you recommend a restaurant?"
},
"context": {
"conversation_id": "d7a26752-6798-408b-95b4-fe1c207e0750",
"system": {
"dialog_stack": [
"root"
],
"dialog_turn_counter": 2,
"dialog_request_counter": 2,
"_node_output_map": {
"node_3_1475173990438": [
0,
1,
0
]
}
},
"username": "Jazmin"
}
}
Watson understands
{
"intents": [
{
"intent": "local_recommend",
"confidence": 0.950181044608299
}
],
"entities": [],
"input": {
"text": "Can you recommend a restaurant?"
},
"output": {
"log_messages": [],

5 Activity 6: Iterative Teach, Test, and Calibrate

In this section, you will learn how to set up and use the Conversation Test Tool (CTT). The CTT is a Java application that connects with the conversation service that you want to test and submits a series of inputs to the service. This tool was created by Simon O'Doherty/Ireland/IBM. There is a link to his blog in the online course materials to point you to more information about this tool.

Note: If you have signed out of Bluemix from your terminal, you will need to sign in again before proceeding with this step. The instructions to sign in are in the section titled, "Push the modified code package to Bluemix".

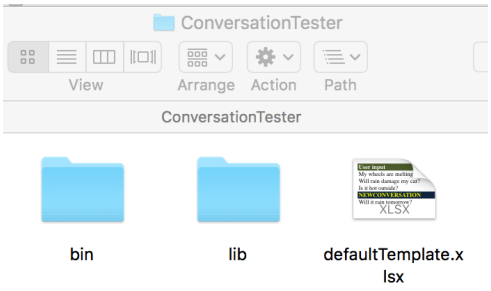
5.1 Conversation Test Tool

5.1.1 Download the Conversation Test Tool

The downloads needed for this section are located both in the Lab Resources section of the online course or in the zipped `WCS201_AllLabDocs.zip` file.

168. Unzip the file `ConversationTestTool.zip`. It should unpack into a folder titled **ConversationTester.**

169. Open the **ConversationTester** folder. You should see the following content in the folder:

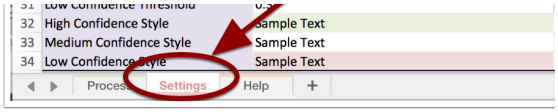


5.1.2 Configure the Conversation Test Tool

This step uses Microsoft Excel to enter and edit configuration values for the Conversation Test Tool.

170. Open `defaultTemplate.xlsx`

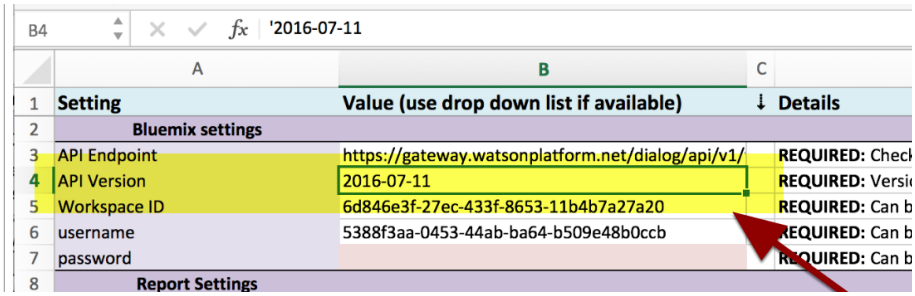
171. Click on the **Settings** tab.



On this page, you will enter the information needed by the Conversation Test Tool to connect to the Conversation service. This page also gives you many options to configure the final report.

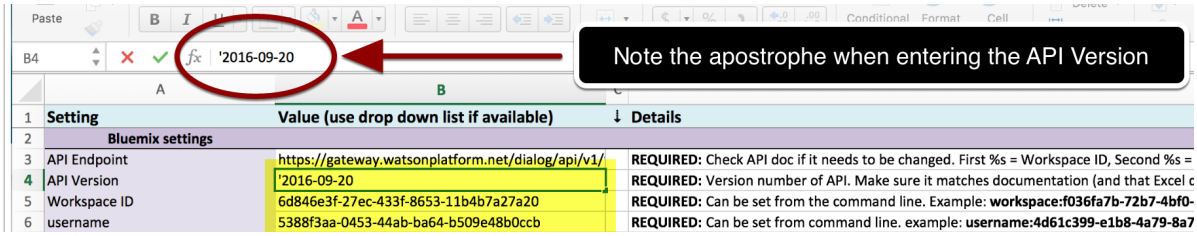
172. Locate the API Version cell.

173. Click on the cell to the right of the API Version cell that has the value `2016-07-11`. This should be cell B4.



174. Enter `'2016-09-20` for the API version number into cell B4.

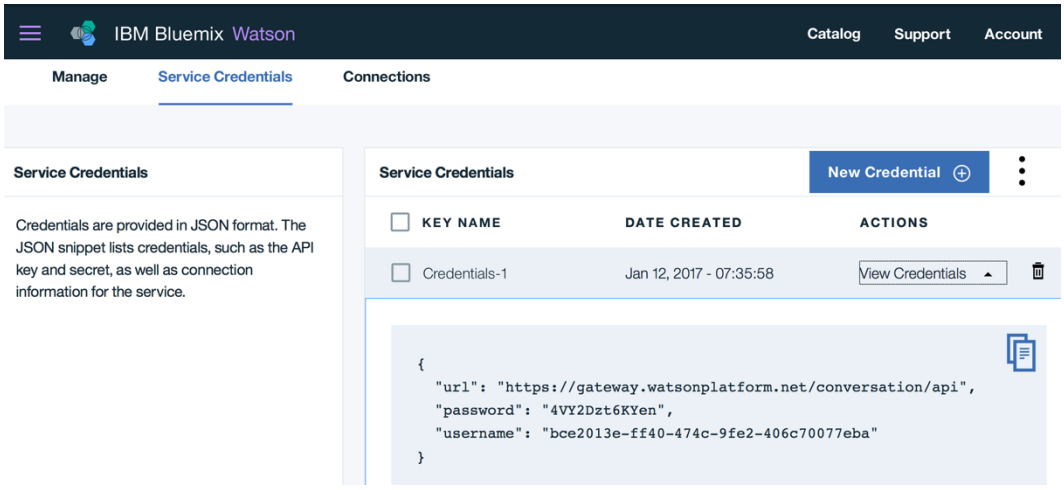
Note: There is an apostrophe at the beginning of the version entry.



Note: If the api version number does not work, use this link to obtain the most recent version number: https://www.ibm.com/watson/developercloud/conversation/api/v1/#versioning

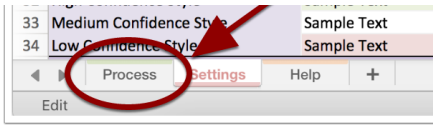
175. Enter the Workspace ID for your instance of WCS in cell B5 (this is the same workspace ID that you used for the UI code).

176. Enter your WCS username into cell B6. (This is your username for your instance of WCS, located in the Service Credentials tab of WCS. Click on View Credentials to get your service URL, username, and password).



177. Enter your WCS instance password into cell B7. Click Save.

178. Click on the **Process** tab in the worksheet.



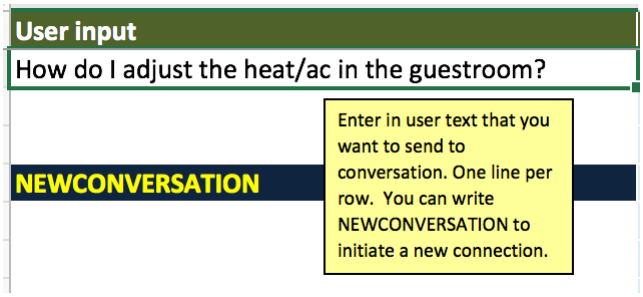
For the purposes of this lab exercise, you will only test a few of the intents included in the lab guide's ground truth.

You will need to enter the User Input and Match Intent information in your spreadsheet to match the information in the lab guide use case. If you are not sure what to enter, or want to enter something other than the image below, be sure to reference the `WCS201Labs_CoreIntentsGT.xlsx` document for the correct ground truth information for the workspace that you are connected to.

Each line in the spreadsheet represents one user example that will be submitted to the Conversation service. The actual end-user example is entered in the User Input column.

The entries in the **Match Intent** column are the intents that we expect to get back from the Conversation Service.

Note: When you put your cursor in a cell on this worksheet, a message will appear to give you more information.



179. Enter the information into the Process worksheet as shown below. You may choose to enter a train set of your choice, or you can use the `defaultTemplate_CoreIntents_TestSet1.xslx` document in your lab resources zip folder for this step.

1	User input	Match Output	Match Intent
2	How do I adjust the heat/ac in the guestroom?		hotel_procedure
3	What time does the food place open?		hotel_hours
4	When is the pool open?		hotel_hours
5	Is there a local full service gym?		hotel_locations
6	Which floor is the gym/pool on?		hotel_locations
7	Where can I get sushi?		hotel_locations

180. Save the spreadsheet.

5.1.3 Execute the Conversation Test Tool

181. Open a Terminal `[Mac]` or a Command Prompt `[Windows]`.

182. Type the command to go to the directory where the defaultTemplate.xlsx file is located:

```
[Mac]      cd Downloads/ConversationTester
[Windows]  cd Downloads\ConversationTester
```

If you are having problems finding the directory for your file, please ask the instructor for help.

183. Type the command to start the test (Note that the commands are case sensitive):

```
[Mac]      bin/conversationtester.sh defaultTemplate.xlsx

[Windows]  bin\conversationtester.cmd defaultTemplate.xlsx
```

Note: If you are using the document that has been prepared for you with train set 1, use these commands:

```
[Mac]

bin/conversationtester.sh defaultTemplate_CoreIntents_TestSet1.xlsx

[Windows]

bin\conversationtester.cmd defaultTemplate_CoreIntents_TestSet1.xlsx
```

184. Press Enter.

After a few seconds, you should see a similar output to your screen:

```
Conversation Tester. Version: 1.0.20160918
IBM Internal tool only. Report can be shared with customer.

Reading: defaultTemplate.xlsx
-----------------------------

In     Out    Match  Text
-----  -----  -----  ------
00001->00004 :     : Hi!  What's the quickest way to get from point A to point B
.
00002->00006 :     : How do I get in the gym?
00003->00008 :     : Do you have an outdoor indoor pool?
00004->00010 :     : How deep is the pool?
00005->00012 :     : Can you make a dinner recommendation - we are thinking Ital
ian.
00006->00014 :     : Can you recomend a resturant?
00007->00016 :     : Can you recommend a good Italian restaurant?

Writing to: SampleOutput.xlsx
DONE.
```

5.1.4 Obtain and View the Conversation Test Tool Report

The CTT report is saved in a file called `SampleOutput.xlsx` in the **ConversationTester** folder. This file is located in the same directory as the `defaultTemplate.xlsx` spreadsheet you configured in the steps above.

Take note that the file name `sampleOutput.xlsx` is defined in the `defaultTemplate.xlsx` configuration, in the Settings tab. You may change this file name to anything you need, but it must be saved with an `*.xlsx` suffix.

185. Locate `SampleOutput.xlsx` in the ConversationTester directory and open the file. The file should appear similar to the screen shot below:

User Input	Output Text	Ma	Mat	Mat	Intent	Intent Co	Entity	Entity Value
	Hey, I'm Watson. What is your name?							
How do I adjust the heat/ac in the gu	Hi How do I adjust the heat/ac in the gue		Yes		hotel_procedure	1	guestroom	heat_ac
	As your hotel concierge, how can I help y							
What time does the food place open	I don't understand what you are asking. '		Yes		hotel_hours	0.9944113	hotel_amenity	hotel restaurant
When is the pool open?	The hotel pool is open from 5am to 8pm		Yes		hotel_hours	1	hotel_amenity	pool
Is there a local full service gym?	The gym is in the front lobby.		Yes		hotel_locations	1	hotel_amenity	gym
Which floor is the gym/pool on?	The pool is on the second floor.		Yes		hotel_locations	1	hotel_amenity	gym

5.1.5 Understanding the data in the Conversation Test Tool Report

There are many different types of data captured in this report. Some of the most interesting include:

1. **User Input**: The input that was submitted to the Conversation Service.
2. **Output Text**: The output that the Conversation Service responded with.
3. **Matched Output/Intent/Entity**: Each of these 3 columns will be Yes or No, depending on whether or not the actual output of the Conversation Service matched the expected output that was built into the dialog flow. In the example above we see that the "Matched Intent" column shows "Yes" for each user input. This means that the actual intent that was output by the Conversation Service matched the expected intent.

Thank you for attending WCS 201!

Please check your email for a Watson University brief student survey. Let us know what you liked about this course and what suggestions you have to help us improve.

Updates to this lab guide will be made available in the **WCS 201 Alumni** community and in the online version of the WCS 201 course. If you are taking this class as an online course instead of in-person, email Jazmin Capezza, JazminCapezza@us.ibm.com, to be added to the WCS 201 Alumni community.