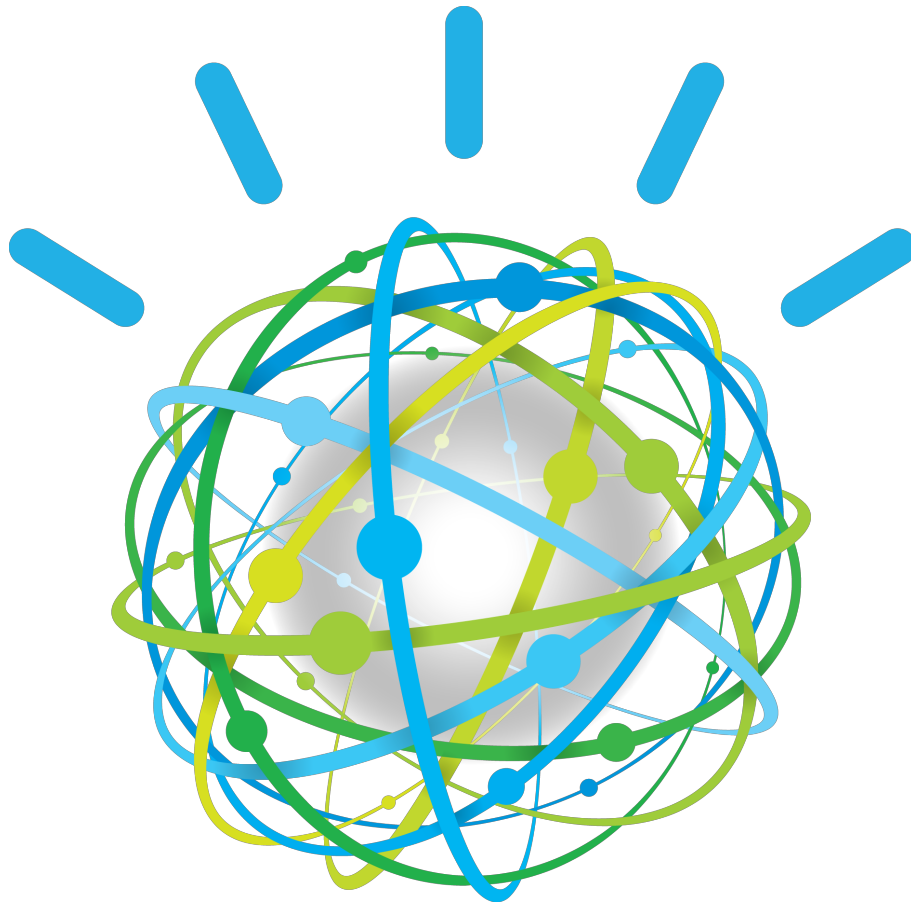


Session 4: Building a conversation

# IBM Watson Assistant



**Lab Instructions**

# Content

<b>Let's get started .....</b>	<b>3</b>
1. Overview .....	3
2. Objectives .....	3
3. Prerequisites .....	3
4. Scenario .....	3
5. What to expect when you are done .....	4
<b>Create an Assistant in IBM Cloud .....</b>	<b>5</b>
6. Create your Watson Assistant Service .....	5
7. Create a your first Skill .....	7
<b>Intents .....</b>	<b>9</b>
8. Create your first intent.....	9
9. Import intents .....	11
10. Test your Intents .....	12
11. Content Catalog .....	17
<b>Entities .....</b>	<b>18</b>
12. Create your first entity .....	18
13. Import entities .....	20
14. Test your entities.....	20
15. Contextual entities .....	22
16. Entity pattern.....	26
17. Test your pattern .....	26
18. Enable system entities .....	27
19. Test System Entities .....	28
<b>Conversation Dialog .....</b>	<b>29</b>
20. Create your first Dialog .....	29
21. Create your first nodes.....	34
22. Test your conversation.....	36
<b>Enrich your data by using context variables .....</b>	<b>38</b>
<b>Create your first Assistant .....</b>	<b>40</b>

# Let's get started

---

## 1. Overview

The [IBM Watson Developer Cloud](#) (WDC) offers a variety of services for developing cognitive applications. Each Watson service provides a Representational State Transfer (REST) Application Programming Interface (API) for interacting with the service. Some services, such as the Speech to Text service, provide additional interfaces.

The [Watson Assistant](#) service combines several cognitive techniques to help you build and train a bot - defining intents and entities and crafting dialog to simulate conversation. The system can then be further refined with supplementary technologies to make the system more human-like or to give it a higher chance of returning the right answer. Watson Conversation allows you to deploy a range of bots via many channels, from simple, narrowly focused bots to much more sophisticated, full-blown virtual agents across mobile devices, messaging platforms like Slack, or even through a physical robot.

The **illustrating screenshots** provided in this lab guide could be slightly different from what you see in the Watson Assistant service interface that you are using. If there are colour or wording differences, it is because there have been updates to the service since the lab guide was created.

## 2. Objectives

In this lab, you will:

- Learn how to use the IBM Cloud (Bluemix) web user interface to create and manage Watson services
- Learn how to train your chat bot to answer some asked questions

## 3. Prerequisites

Before you start the exercises in this guide, you will need to complete the following prerequisite tasks:

- Session 1 – Getting Started
- The instructor provided you the link to get labs content. You may download each file individually.

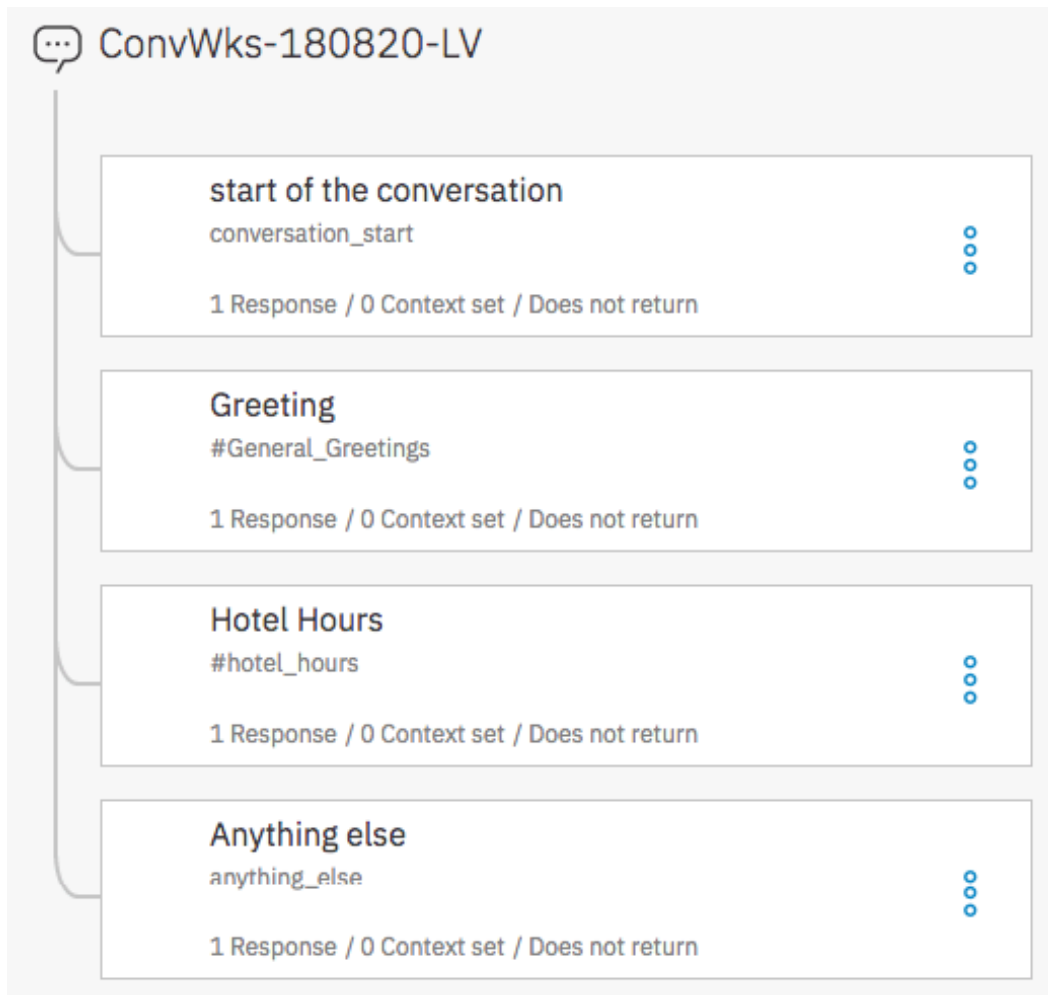
## 4. Scenario

**Use case:** A Hotel Concierge Virtual assistant that is accessed from the guest room and the hotel lobby.

**End-users:** Hotel customers

## 5. What to expect when you are done

At the end of session, you should have a simple dialog using  
23 intents,  
11 entities.



# Create an Assistant in IBM Cloud

## 6. Create your Watson Assistant Service

IBM Cloud offers services, or cloud extensions, that provide additional functionality that is ready to use by your application's running code.

You have two options for working with applications and services in IBM Cloud. You can use the IBM Cloud web user interface or the Cloud Foundry command-line interface. Today, you are using IBM Cloud user interfaces.

1. In a web browser, navigate to the following URL:

Location	URL
worldwide	<a href="https://console.bluemix.net/">https://console.bluemix.net/</a>

2. Log in with your IBM Cloud credentials. This should be your IBM ID.
3. You should start on your dashboard that contains a list of all your applications and services. On the top left click **Catalog** to display the list of available services
4. Enter *Watson Assistant* as Filter

Catalog

The screenshot shows the IBM Cloud Catalog interface. At the top, there is a search bar with the text 'watson assistant' and a 'Filter' button. Below the search bar, there is a list of categories on the left, including 'All Categories (2)', 'Compute', 'Containers', 'Networking', 'Storage', 'AI (1)', 'Analytics', 'Databases', 'Developer Tools', 'Integration', 'Internet of Things', 'Security and Identity', 'Starter Kits (1)', 'Web and Mobile', and 'Web and Application'. The 'AI (1)' category is selected. The main content area displays the search results for 'Watson Assistant'. Under the 'AI' section, there is a card for 'Watson Assistant (formerly Conversation) Lite • IBM'. The card description reads: 'Add a natural language interface to your application to automate interactions with your end users. Common applications include virtual agents and chat bots that can integrate and communicate on any channel or device.' Below this, under the 'Starter Kits' section, there is a card for 'Watson Assistant Basic IBM'. The card description reads: 'Simple application that demonstrates the Watson Assistant service in a chat interface simulating a car dashboard.'

5. Click on the **Watson Assistant** service tile.
6. Review the details for this service. At the top, there will be a description of the service. At the bottom, you can review the pricing plans. For this service, notice that the first 10 000 API calls per month are free. Enjoy your demo!
7. At the top, you can enter information for your new service. In the middle, you can click on a pricing plan to select it. Fill out the fields as follows, then click **Create** at the bottom.

Field	Value
Service name	<i>Conversation-XXXXXX-LV</i>
Region / location	<i>US South / Dallas</i>
space	dev
Selected Plan	<i>Lite</i>

Where **XXXXXX** must be replaced with the date of the creation with the format yymmdd and **LV** must be replaced with your initials.

IBM Cloud has created a new service instance. In order to use this specific instance in your application, you will need to obtain the credentials. There are display in the manage tab.

The screenshot displays the IBM Cloud console interface for a Watson Assistant service instance. On the left, a sidebar shows navigation options: Manage, Service credentials, Plan, and Connections. The main content area shows the service details for 'Conversation-181112-LV'. It indicates that 0.62% of the 9938 available API calls have been used. The location is set to Dallas. Below the service name, there are links for 'Launch tool', 'Getting started tutorial', and 'API reference'. A 'Launch tool' button is also present. The 'Credentials' section is highlighted with a red box, showing the 'API Key' (masked with dots) and the 'Url' (https://gateway.watsonplatform.net/assistant/api). A 'Show Credentials' link with an eye icon is also visible.

You should see the API Key and Url for your service. Later in this exercise, you will enter these values into a JSON configuration file for your Node.js and Node-Red application. Feel free to copy them to your clipboard, to a text file, or just return to this section of the IBM Cloud web interface when the credentials are needed.

## 7. Create a your first Skill

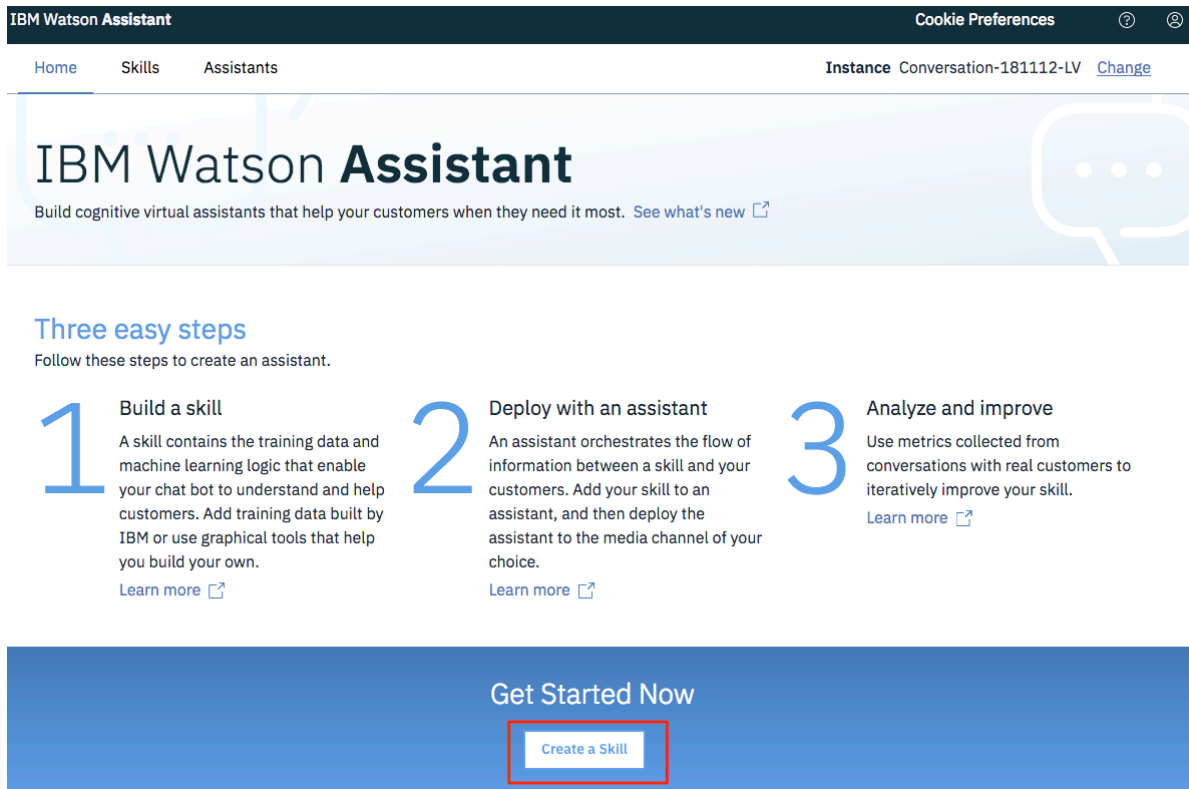
To use the new conversation, you will need to train it with the intents, entities, and/or dialog nodes relevant to your application's use case. You will create these items in a Skill. A Skill is a container for the artefacts that define a piece of the behaviour of your service instance.

1. On the left side of the page, click the **Launch tool** button.

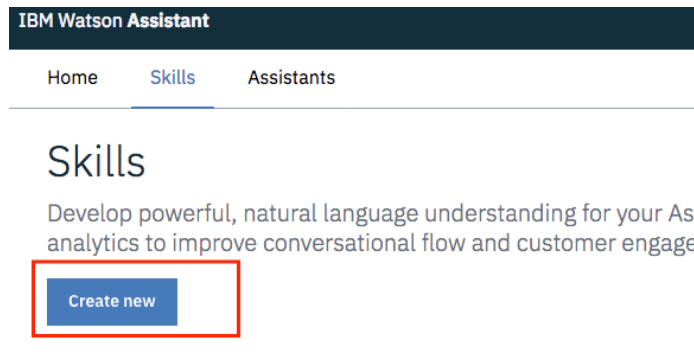
A green rectangular button with the text "Launch tool" and an external link icon (a square with an arrow pointing out).

Your first step in the Watson Assistant tool is to create a skill.

2. From the home page of the Watson Assistant tool, at the bottom, click **Create a Skill** tab

A screenshot of the IBM Watson Assistant web interface. At the top is a dark blue header with "IBM Watson Assistant" on the left, "Cookie Preferences" on the right, and a help icon. Below the header is a navigation bar with "Home", "Skills", and "Assistants" tabs. The main content area has a light blue background with the "IBM Watson Assistant" logo and a tagline: "Build cognitive virtual assistants that help your customers when they need it most." Below this is a section titled "Three easy steps" with three numbered steps: 1. Build a skill, 2. Deploy with an assistant, and 3. Analyze and improve. Each step has a brief description and a "Learn more" link. At the bottom is a blue banner with the text "Get Started Now" and a button labeled "Create a Skill" which is highlighted with a red rectangle.

3. Skill enables you to maintain separate intents, user examples, entities, and dialogs for each application. Click **Create new**.



4. In the *Create a skill* wizard, enter the following values and click **Create**.

Field	Value
Name	<i>ConvWks-XXXXXX-LV</i>
Description	<i>For demos only</i>
Language	<i>English (U.S.)</i>

Where **XXXXXX** must be replaced with the date of the creation with the format yymmdd and **LV** must be replaced with your initials.

## Add Dialog Skill

Create a new skill, add a sample, or import one

**Create skill**    Use sample skill    Import skill

---

**Name**

ConvWks-181112-LV

**Description (optional)**

For demos only

**Language**

English (US)

Create

Once the workspace has been created, your workspace should be displayed to start to create your first intent.

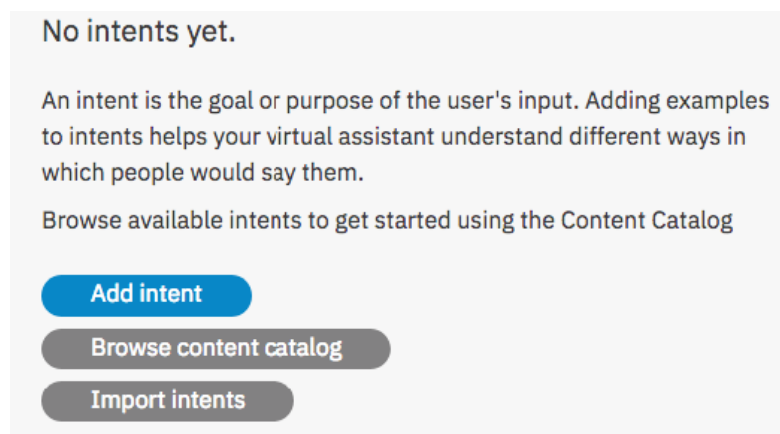


# Intents

## 8. Create your first intent

In order to use the new conversation, you will need to train it with the intents, entities, and/or dialog nodes relevant to your application's use case. An intent is the purpose or goal of a user's input.

1. First, you will need to define some *intents* to help control the flow of your dialog. An [intent](#) is the purpose or goal of a user's input. In other words, Watson will use natural language processing to recognize the intent of the user's question / statement to help it select the corresponding dialog branch for your automated conversation. If not already there, click the **Intents** tab at the top of your workspace.
2. On the *Intents* tab, click **Add intent** in the dialog window.



3. Enter [hotel\\_locations](#) as Intent name then click **Create intent**

A screenshot of the "Create new intent" form. At the top is a back arrow and the title "Create new intent". Below is a form with two fields: "Intent name" with the value "#hotel\_locations" and "Description" with the placeholder text "Add a description to this intent". Below the fields is a blue button labeled "Create intent" which is highlighted with a red rectangle. At the bottom of the form, it says "No examples yet." followed by the text "Train your virtual assistant with this intent by adding unique examples of what your users would say."

4. Enter the examples as defined below in the **Add user examples** field and click **Add example** button after each of them. The *user examples* are phrases that will help Watson recognize the new *intent*.

Field	Value
Intent name	<i>hotel_locations</i>
User example	<i>Is the pool outside?</i> <i>Where is the gym?</i> <i>Do you have a sauna?</i>

The screenshot shows the IBM Watson Assistant interface for editing an intent. At the top, there is a back arrow and the intent name "#hotel\_locations". Below this, the "Intent name" field is set to "#hotel\_locations". The "Description" field contains the placeholder text "Add a description to this intent". The "Add user examples" section shows the placeholder text "Add user examples to this intent" and a blue "Add example" button. Below this, a list of user examples is displayed, each with a checkbox and a delete icon:

- ☐ User examples (3) ▼
- ☐ Do you have a sauna? ✎
- ☐ Is the pool outside? ✎
- ☐ Where is the gym? ✎

5. When finished, click on the arrow Icon (top left side of the window) to go back to list of existing intents

At this point, you have defined one *intent* for the application along with the example utterances that will help train Watson to recognize and control the conversation flow.

## 9. Import intents

Intents can be imported into the Conversation tool using a Comma Separated Values (CSV ) file saved with UTF-8 encoding.

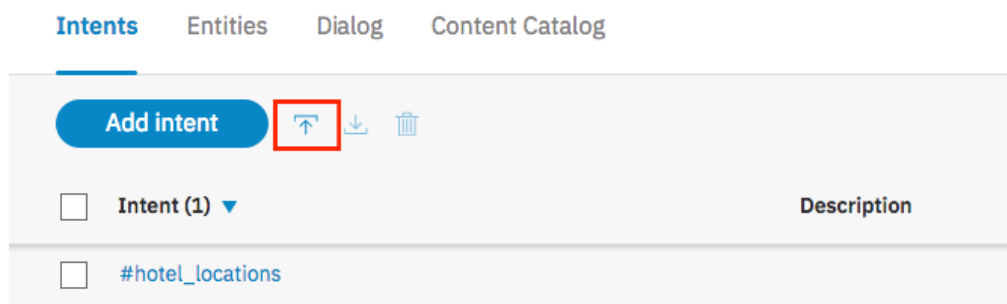
In order to import a file successfully, it must include only two columns: one for user examples and one for intents.

**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.

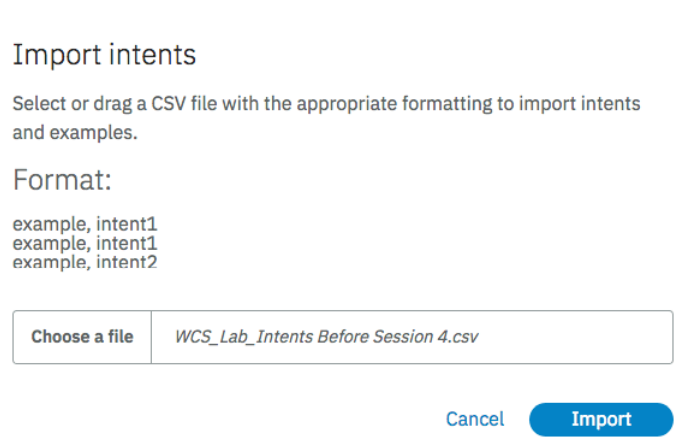
- Copy the end-user examples into a single column.
- Label each end-user example with its assigned intent in the column beside it.
- Do not insert any column headings for the intent import file.
- Use a text editor to make sure that your file is UTF-8 and saved as a .csv file. We suggest Numbers for Mac and Notepad++ for Windows.

**Note:** 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.

### 1. Click **Import**



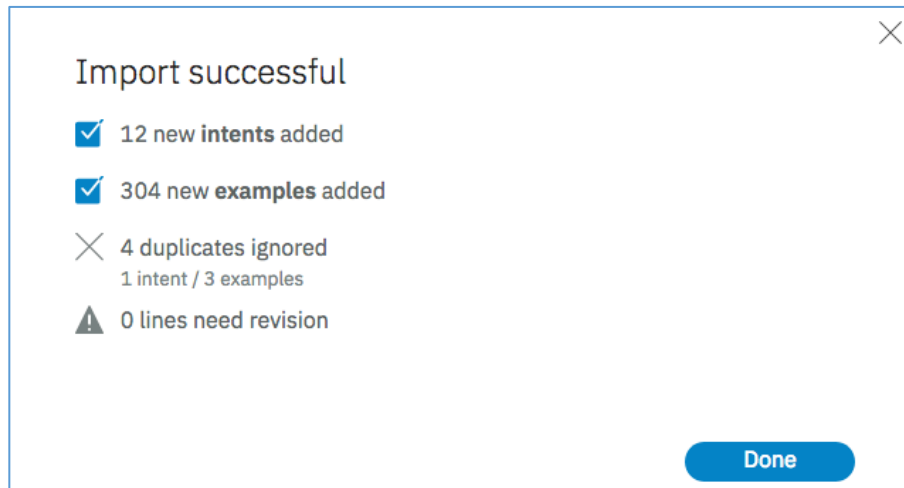
### 2. Drag [WCS\\_Lab\\_Intents Before Session 4.csv](#) into the **Choose a file** box or click **Choose a file** to browse your computer.



### 3. Click **Import**

**Note:** Notice that the Conversation tooling did not import any questions that were exact duplicates of questions that you manually entered. Duplicates are not case sensitive. For example, “Do you have a sauna?” is the same as “do you have a sauna”. It did import questions that had semantic differences, such as the absence of punctuation, extra spaces between words, or misspelled words.

So all existing utterances of hotel\_locations were ignored:



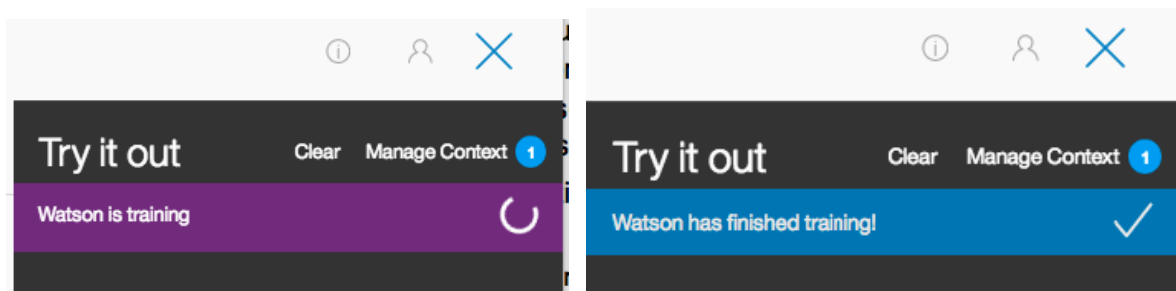
#### 4. Click **Done**

Your conversation should have 12 intents, with 307 utterances

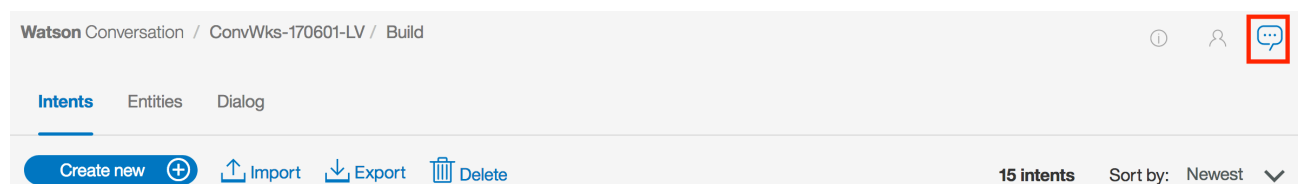
## 10. Test your Intents

**Try it** panel is used for testing in the lab guide.

When WA is training on recently added data. You must wait for the message highlighted with purple to clear before you can test newly added intents, utterances, entities. Watson will respond, but you will get unpredictable results until the training is complete.



#### 1. Open the **try it** panel by clicking on the following icon (upper right):



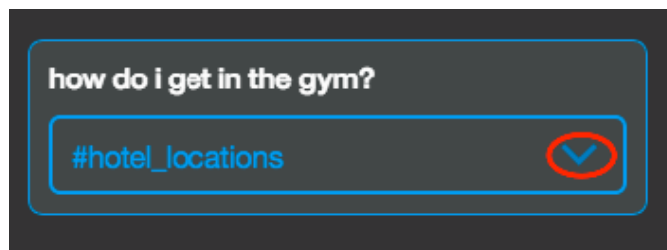
You cannot perform the **Try it out** activities in this section until Watson has finished training on your recent changes. When Watson is finished, the purple box with the text, “Watson is training” will no longer appear in your "Try it out" panel. You may also notice a message stating that, "Watson has finished training!".

2. Type *how do I get in the gym?*

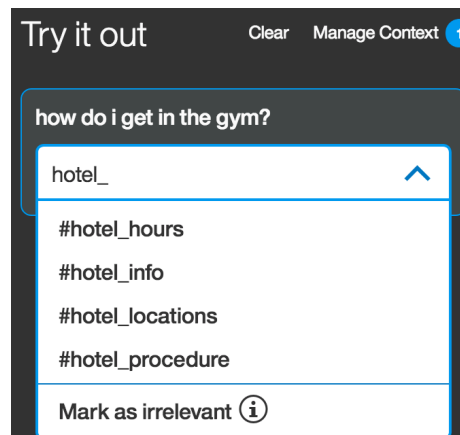
**Note:** Notice that Watson assigns this question to the intent *#hotel\_locations*. The *#hotel\_locations* intent is meant for questions that refer to finding directions for locations within the hotel. In this case, the user wants to know how to get in the gym, not how to get to the gym.

Let's change the intent for the user example.

3. Click the arrow next to *#hotel\_locations*



4. Type *hotel\_info*



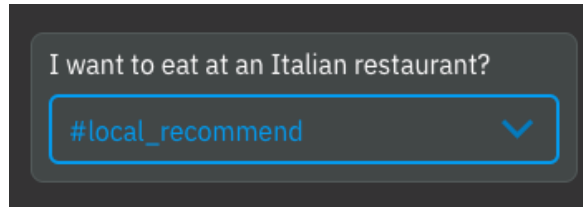
5. Select the intent *#hotel\_info* that appears in the drop-down box

Your try it out panel will show a message similar to “Watson is training”

**Note:** If you go to your Intents panel, and open the *#hotel\_info* intent, you will now see that your input has been added to this section.

6. Type *I want to eat at an Italian restaurant?*

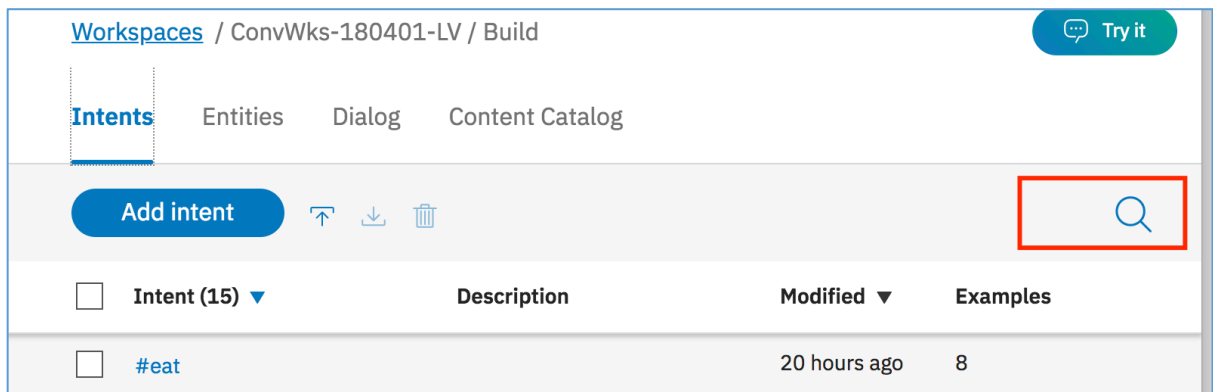
**Note:** Notice that Watson assigns this question the intent `#Local_recommend`. This intent is meant for questions that refer to finding recommendation for sightseeing or restaurant. In this case, the user wants to eat something.



Let's change the intent for the user example, this time by using the search capability of the engine.

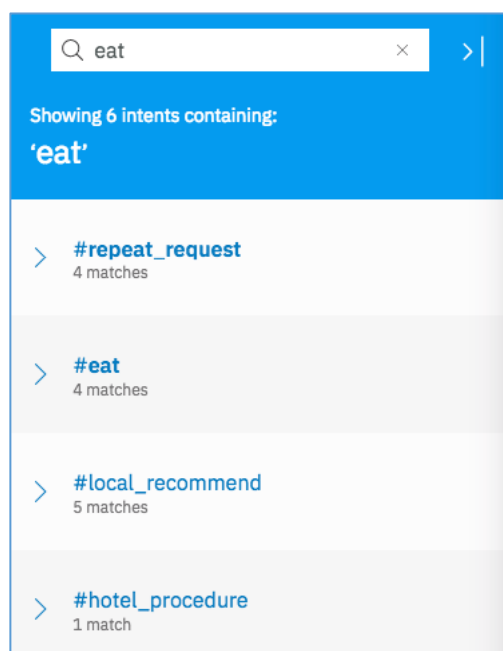
7. Close the **Try it out** Panel

8. ON the intent tab click on the search icon

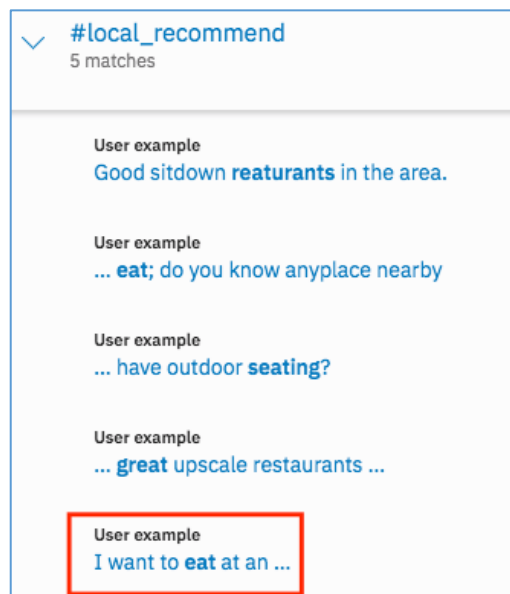


9. Enter *eat* in the search field (retry this search if indexing is needed)

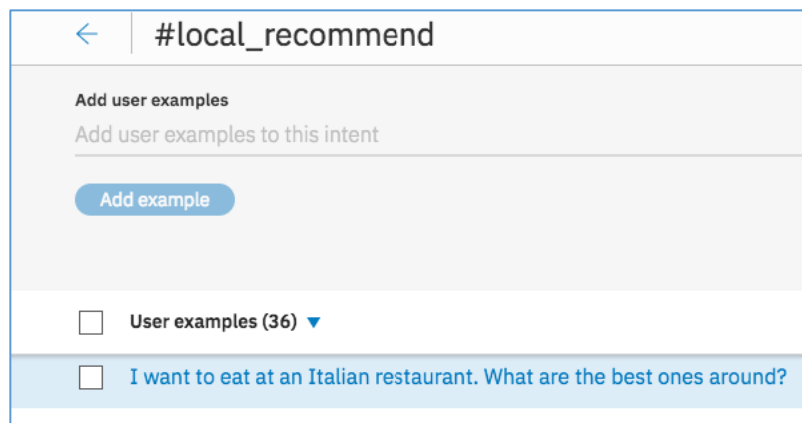
The solution returns all intents including the 'eat' word.



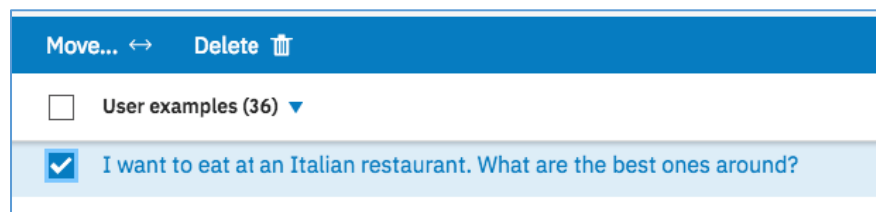
10. As you are looking for a user example which belongs to #local\_recommend, expand it.



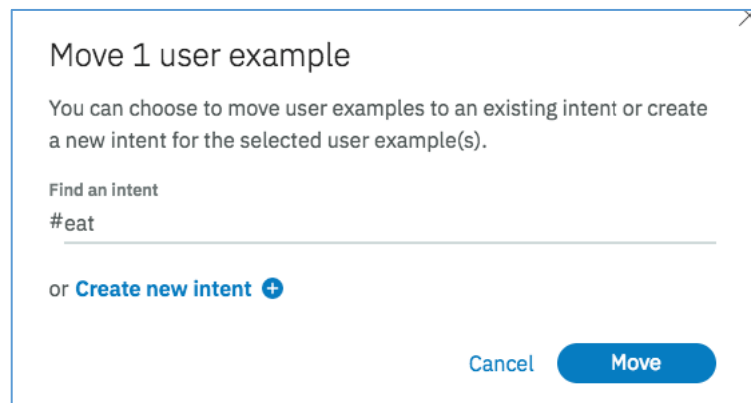
11. The last one looks like the one we used during the test, click on it  
Watson open a right page and highlight the right user example.



12. Select it, click on **Move**



13. Enter *eat* as intent



Move 1 user example

You can choose to move user examples to an existing intent or create a new intent for the selected user example(s).

Find an intent

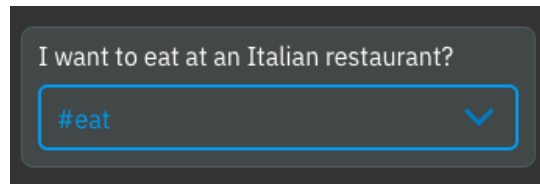
#eat

or [Create new intent](#) +

Cancel Move

14. Click on **Move**

**Note:** Now, your user example is move on the right intent and if you will do a new test the system returns the right intent.



I want to eat at an Italian restaurant?

#eat ✓

You have seen two options to improve the Watson classification

- Add a new user example in an existing intent

- Move an existing user example to the right intent.

You are going to review several other options during this workshop.



## 11. Content Catalog

**Content Catalogs** provide an easy way to add common intents to your Watson Assistant service workspace.

You are going to add general conversation topics intents.

1. On the **Content Catalog** tab, click Add to workspace to add General category intents.

Intents	Entities	Dialog	Content Catalog
Get started faster by adding existing intents from the content catalog. These intents are trained on common questions that users may ask.			
Category	Description	Intents	
Banking	Basic transactions for a banking use case.	13	+ Add to workspace
Bot Control	Functions that allow navigation within a conversation.	9	+ Add to workspace
Customer Care	Understand and assist customers with information about themselves and their account.	18	+ Add to workspace
eCommerce	Payment, billing, and basic management tasks for orders.	14	+ Add to workspace
General	General conversation topics most users ask.	10	+ Add to workspace
Insurance	Issues related to insurance policies and claims.	12	+ Add to workspace
Telco	Questions and issues related to a user's telephony service, device, and plan.	21	+ Add to workspace
Utilities	Help a user with utility emergencies and their utility service.	10	+ Add to workspace

2. Go back to **Intents** tab, we will find new intents prefix by #general\_

<input type="checkbox"/>	#General_About_You	Request generic personal attributes.	5 mir
<input type="checkbox"/>	#General_Agent_Capabilities	Request capabilities of the bot.	5 mir
<input type="checkbox"/>	#General_Connect_to_Agent	Request a human agent.	5 mir
<input type="checkbox"/>	#General_Ending	End the conversation.	5 mir
<input type="checkbox"/>	#General_Greetings	Greet the bot.	5 mir
<input type="checkbox"/>	#General_Human_or_Bot	Ask if speaking to a human or a bot.	5 mir
<input type="checkbox"/>	#General_Jokes	Request a joke.	5 mir
<input type="checkbox"/>	#General_Negative_Feedback	Express unfavorable feedback.	5 mir
<input type="checkbox"/>	#General_Positive_Feedback	Express positive sentiment or gratitude.	5 mir
<input type="checkbox"/>	#General_Security_Assurance	Express concerns about the security of the	5 mir

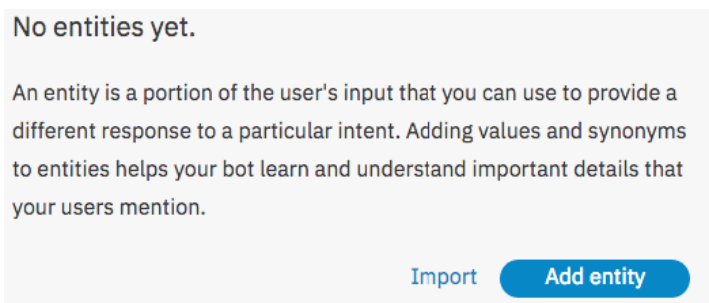
# Entities

## 12. Create your first entity

To use the new conversation, you will need to train it with the intents, entities, and/or dialog nodes relevant to your application's use case. An entity is the portion of the user's input that you can use to provide a different response or action to an intent.

**Note:** you will need to create some *entities*. An [entity](#) is the portion of the user's input that you can use to provide a different response or action to an intent. These entities can be used to help clarify a user's questions/phrases. You should only create *entities* for things that matter and might alter the way a bot responds to an *intent*.

1. If not already there, click the **Entities** tab at the top of your workspace.
2. Click **Add entity**.



3. As you did for intent create your first entity with the following information

In these labs, several intents will indicate that the user are looking for a restaurant. So, you will need to create a new *entity* representing restaurant categories. You will then provide *values* (and possibly synonyms) for the various types. (Enter multiple *examples* by pressing "Enter" or clicking the **plus sign** at the end of the line.)

A form for creating a new entity. It has a section for "Entity name" with the text "@restaurant". Below that is a table with two columns: "Value name" and "Synonyms". The "Value name" column has the text "pizza\_restaurant". The "Synonyms" column has the text "pizza restaurant" and a blue plus sign icon in a red square. At the bottom left, there is a blue button labeled "Add value".

Field	Value	Synonym
Entity name	<i>restaurant</i>	<< N/A >>
Value	<i>pizza_restaurant</i>	<i>pizza restaurant, pizza, pizza pie, pizzeria</i>
	<i>chinese_restaurant</i>	<i>chinese restaurant, spring roll, noodles</i>
	<i>french_restaurant</i>	<i>french restaurant, brasserie, frog legs, macaron</i>

←

@restaurant

La

Entity name

@restaurant

Fuzzy Matching BETA Off 🔍

Value name

Enter value

Synonyms

Synonyms

Add synonym...

+

Add value

Show recommendations

Dictionary

Annotation BETA

Entity values (3) ▼

Type

chinese\_restaurant

Synonyms

chinese, chinese food, noodles, spring roll, chinese resta

french\_restaurant

Synonyms

french, french restaurant, brasserie, frog legs, macaron, fi

pizza\_restaurant

Synonyms

pizza restaurant, pizzeria, pizza pie, pizza

4. Enable the **Fuzzy Matching** option
5. When finished, click on the arrow icon

## 13. Import entities

Entities like intents can be imported into the Conversation tool using a Comma Separated Values (CSV ) file saved with UTF-8 encoding.

In order to import a file successfully, it must include a minimum of two columns: one for entity names and one for entity values which can be followed by the synonyms.

1. Click **Import**
2. Drag [WCS\\_Lab\\_Entities.csv](#) into the **Choose a file** box or click **Choose a file** to browse your computer.

**Import entities**

Select or drag a CSV file with the appropriate formatting to import entities, values and synonyms.

Format:

```
entity1, value, synonym
entity1, value, synonym, synonym
entity2, value, synonym, synonym, synonym
```

Choose a file	WCS_Lab_Entity.csv
---------------	--------------------

**Import** **Cancel**

3. Click **Import**

**Import successful** ✕

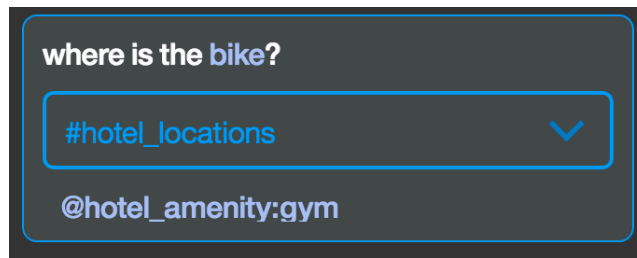
- ✓ 10 new **entities** added
- ✓ 55 new **values** added
- ✓ 76 new **synonyms** added
- ✓ 0 new **patterns** added
- ✕ 19 duplicates ignored  
1 entity / 3 values / 15 synonyms/ 0 patterns
- ⚠ 0 lines need revision

**Done**

**Note:** Notice that the Conversation tooling did not import any questions that were exact duplicates of questions that you manually entered. So the existing restaurant entity name were ignored.

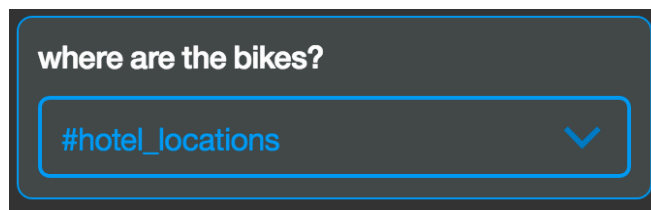
## 14. Test your entities

1. Open the **try it out** panel, wait until Watson is done training.
2. Type [where is the bike?](#)



**Note:** Notice that Watson has identified the term *bike* with the correct intent, entity, and entity value (*@entity\_name:entity\_value*).

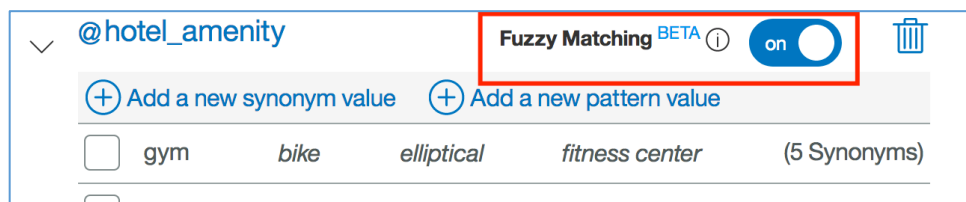
Type *where are the bikes?*



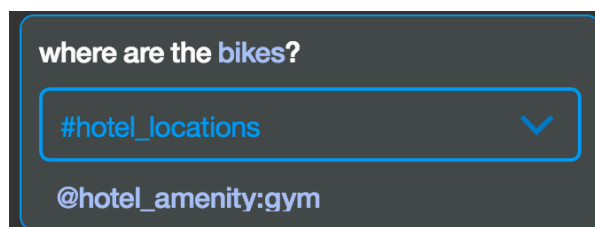
**Note:** Notice that Watson does not identify *bikes* the same as it identified *bike*. This is because entity matching is brute-force rules-based string matching without fuzzy matching.

Let's turn on fuzzy matching to see how it improves entity recognition.

3. Open the *Hotel\_amenity* entity and turn on **Fuzzy Matching**.



4. When Watson is done training, type *where are the bikes?*



**Note:** notice that the correct *@entity\_name:entity\_value* pair is now recognized for *bikes*.

5. Enables the **Fuzzy Matching** for the entities *@pizza-toppings* and *@pizza\_notoppings*.

## 15. Contextual entities

When you define specific values for an entity, the service finds entity mentions only when a term in the user input exactly matches (or closely matches if fuzzy matching is enabled) a value or synonym defined. When you define a contextual entity, a model is trained on both the entity *value* and the *context* in which the entity is used in sentences that you annotate. This new contextual entity model enables the service to calculate a confidence score that identifies how likely a word or phrase is to be an instance of an entity, based on how it is used in the user input

1. Open **Try it out** panel.
2. Enter *I want a pizza without anchovy and more olives* at the bottom of the chat window.

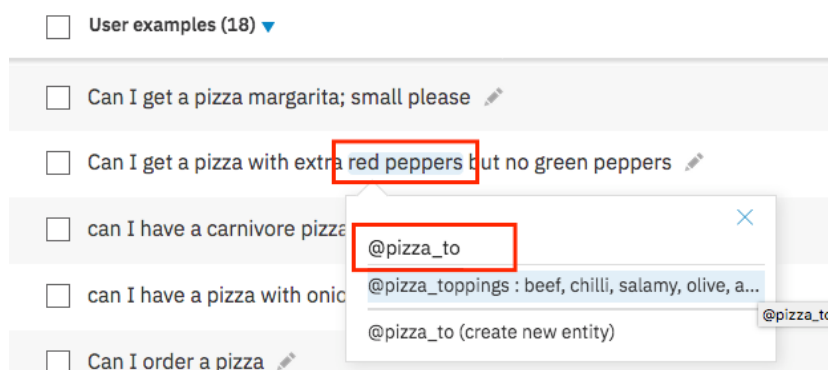
The bot understands the #order\_pizza intend and the entities @pizza\_toppings and @pizza\_notoppings but it can't determine what the user want and don't want!



Watson needs to know the context.

We are going to annotate the user's example to identify where the entities are used.

3. Go to **intents** tab and select *#Order\_pizza*
4. In the 4<sup>th</sup> example, highlight the group *red peppers* the enter *@pizza\_toppings*



We identify the toppings requested by the user.

5. Annotate the other entities like below.

☐ User examples (18) ▼

☐ Can I get a pizza margarita; small please ✎

☐ Can I get a pizza with extra red peppers but no green peppers ✎

☐ can I have a carnivore pizza but no chicken ✎

☐ can I have a pizza with onions, beef, chicken and extra cheese ✎

☐ Can I order a pizza ✎

☐ Can I order a pizza with more cheese ✎

☐ Can I order a pizza without pineapple ✎

☐ Can you deliver a vegetarian pizza ✎

☐ I'd like a large hawaiian with extra pineeapple ✎

☐ I'd like a small margherita but please no anchovies ✎

☐ I 'd like pepperoni pizza ✎

☐ I don't want red peppers or onions on my vegetarian pizza ✎

☐ I love anchovies so please send me a pizza full of them ✎

☐ I want to order a large pizza ✎

☐ I want to order a pizza ✎

☐ medium pizza please and I don't want any red peppers ✎

Now, we are going to identify the toppings not requested by the user.

6. Annotate the intent using @pizza\_notoppings entities like below.

☐ User examples (18) ▼

☐ Can I get a pizza margarita; small please ✎

☐ Can I get a pizza with extra red peppers but no green peppers ✎

☐ can I have a carnivore pizza but no chicken ✎

☐ can I have a pizza with onions, beef, chicken and extra cheese ✎

☐ Can I order a pizza ✎

☐ Can I order a pizza with more cheese ✎

☐ Can I order a pizza without pineapple ✎

☐ Can you deliver a vegetarian pizza ✎

☐ I'd like a large hawaiian with extra pineeapple ✎

☐ I'd like a small margherita but please no anchovies ✎

☐ I 'd like pepperoni pizza ✎

☐ I don't want red peppers or onions on my vegetarian pizza ✎

☐ I love anchovies so please send me a pizza full of them ✎

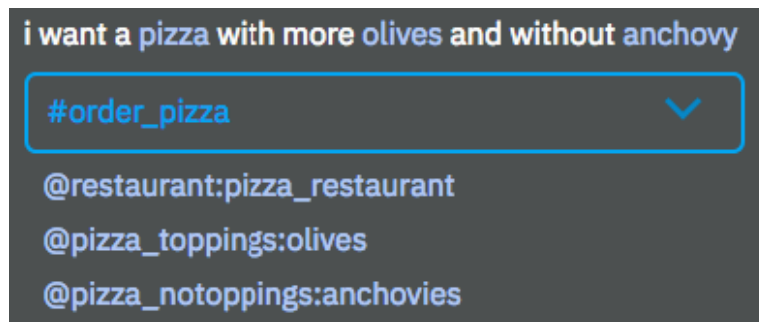
☐ I want to order a large pizza ✎

☐ I want to order a pizza ✎

☐ medium pizza please and I don't want any red peppers ✎



7. Open **Try it out** panel.
8. Enter *I want a pizza with more olives and without anchovy*



So, pretty cool!

If we open the entity [@pizza\\_notoppings](#), and the **Annotation** tab, we retrieve the annotated examples.

Entity name  
@pizza\_notoppings

Dictionary **Annotation BETA**

<input type="checkbox"/> User examples (7)	Intent ▼
<input type="checkbox"/> Can I get a pizza with extra red peppers but no <b>green peppers</b>	#order_pizza
<input type="checkbox"/> can I have a carnivore pizza but no <b>chicken</b>	#order_pizza
<input type="checkbox"/> Can I order a pizza without <b>pineapple</b>	#order_pizza
<input type="checkbox"/> I'd like a small margherita but please no <b>anchovies</b>	#order_pizza
<input type="checkbox"/> I don't want red peppers or <b>onions</b> on my vegetarian pizza	#order_pizza
<input type="checkbox"/> I don't want <b>red peppers</b> or onions on my vegetarian pizza	#order_pizza
<input type="checkbox"/> medium pizza please and I don't want any <b>red peppers</b>	#order_pizza

## 16. Entity pattern

1. Click **Add entity**.
2. Enter  
as entity name : *@pattern*
3. Click **Create entity**
4. Enter  
as value : *email address*  
as pattern: *\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b*  
as type select *Patterns*

Entity name  
@pattern

Value name  
email address

Patterns  
Patterns ▼ \b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b

Add value

5. When finished, click **Add value**, then click on arrow icon  
That the way to extract an e mail address from the user input

## 17. Test your pattern

1. Open the **try it out** panel, wait until Watson is done training.
2. Type: *it is johndoes@gmail.com*

it is johndoes@gmail.com

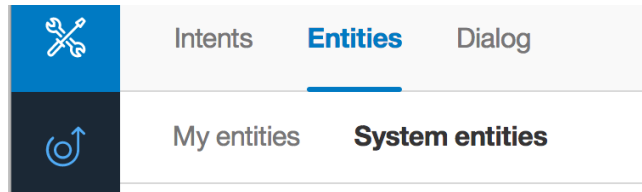
#greeting ▼

@pattern:email address

Watson identify johndoes@gmail.com as an entity email address.

## 18. Enable system entities

1. You will need to enable system entity. These entities can be used to help clarify a user's questions/phrases. If not already there, click the **Entities** and **System entities** tabs at the top of your workspace.



you will get

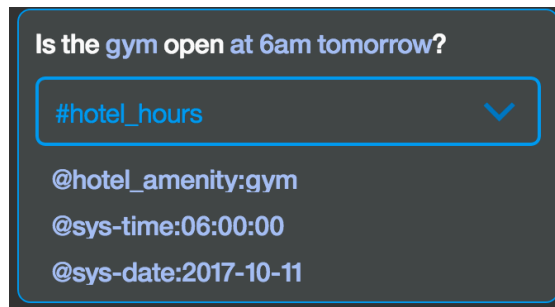
- **@sys-number:** detects numbers that are written using either numerals or words. In either case, a numeric value is returned.
- **@sys-currency:** detects monetary currency values that are expressed in an utterance with a currency symbol or currency-specific terms. A numeric value is returned.
- **@sys-percentage:** detects percentages that are expressed in an utterance with the percent symbol or written out using the word percent. In either case, a numeric value is returned.
- **@sys-date, @sys-time and \$timezone:** Mentions of a date range such as the weekend, next week, or from Monday to Friday are extracted as a pair of @sys-date entity mentions that show the start and end of the range
- **@sys-location :** extracts place names (country, state/province, city, town, etc.) **BETA, English-only**
- **@sys-person :** extracts names from the user's input. **BETA, English-only**

2. Switch on all system entities.

> @sys-currency	Extracts currency values from user examples including the amount and the unit. (20 cents)	<input checked="" type="checkbox"/> On
> @sys-date	Extracts date mentions (Friday)	<input checked="" type="checkbox"/> On
> @sys-location <sup>BETA</sup>	The @sys-location system entity extracts place names (country, state/province, city, town, etc.) from the user's input. (Boston)	<input checked="" type="checkbox"/> On
> @sys-number	Extracts numbers mentioned from user examples as digits or written as numbers. (21)	<input checked="" type="checkbox"/> On
> @sys-percentage	Extracts amounts from user examples including the number and the % sign. (15%)	<input checked="" type="checkbox"/> On
> @sys-person <sup>BETA</sup>	The @sys-person system entity extracts names from the user's input. (Anna)	<input checked="" type="checkbox"/> On
> @sys-time	Extracts time mentions (at 10)	<input checked="" type="checkbox"/> On

## 19. Test System Entities

1. Open the **try it out** panel, wait until Watson is done training.
2. Type *Is the gym open at 6am tomorrow?*

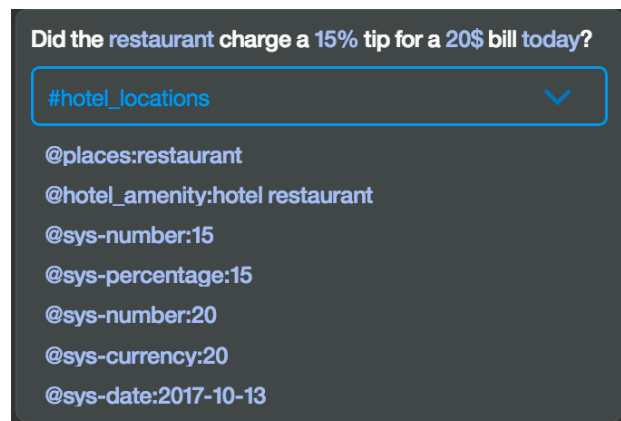


**Note:** notice that the time and date are recognized

**6am** has been captured **06:00:00** which is the standard time format

**tomorrow** has been captured **2017-10-11** (yyyy-mm-dd) which is the format the current date is Oct the 10<sup>th</sup>.

3. Type *Did the restaurant charge a 15% tip for a 20\$ bill today?*



**Note:** notice that the percentage, currency, number and date are recognized. You can use these system entities in your dialog node calculations the same way you would use your custom defined entities.

# Conversation Dialog

---

## 20. Create your first Dialog

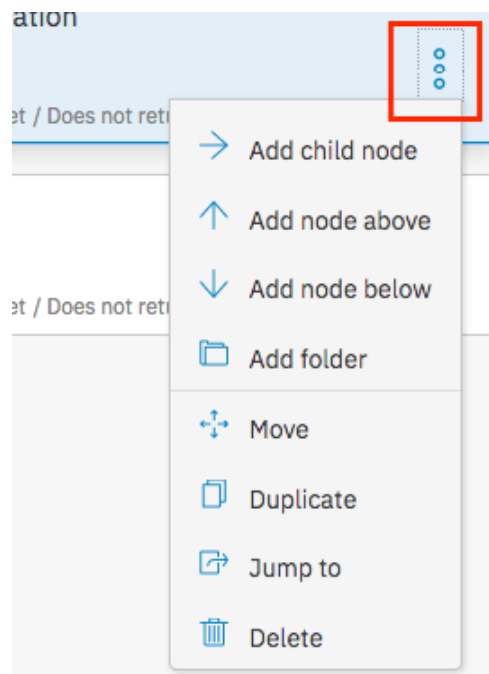
To use the new conversation, you will need to train it with the intents, entities, and/or dialog nodes relevant to your application's use case. A dialog uses the intent and entity that have been identified, plus context from the application, to interact with the user and provide a response.

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

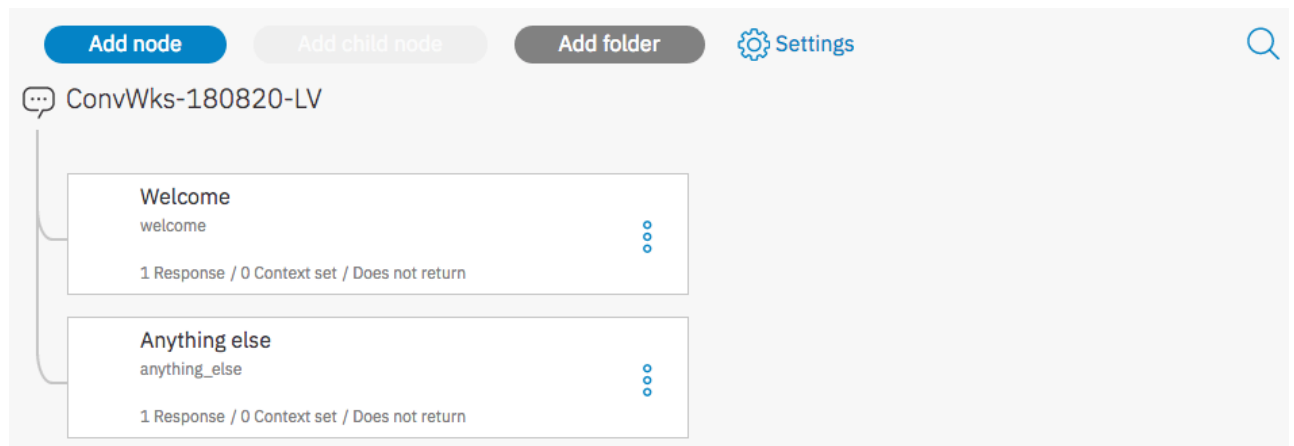
A **condition** is the portion of the dialog node that determines whether the node is used in the conversation. Conditions can be defined using intents, entities, context variables, and special conditions. The Spring Expression (SpEL) language is used to write valid expressions for conditions.

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).

Each node has a node menu. This guide will go through most of the actions that can be completed from the node menu, but be sure to note that **delete** will remove the node and all of its child nodes.



1. Click on **Dialog** tab
2. Click on **Create**



You can notice that Watson has created 2 nodes a “*Welcome*” and “*anything\_else*” nodes.

**Note:**

- The **Welcome** node is automatically added to your dialog chat flow. Welcome is a special condition in Conversation that only evaluated as true during the first dialog turn when the application does not contain any user input. If the initial request from the application contains user input, it will not be triggered. Typically, welcome is used when you always want to display a message or a greeting at the beginning of the conversation.
- The **Anything else** node is automatically added to you dialog chat flow. The Anything else node is set to always evaluate to true. Its purpose is to always provide a response to the user if the user’s input does not evaluate to true for any other parent node in the tree.

**Click on the Anything else** node to review the pre-trained responses. You may edit these if you wish.

- The panel that appears on the **right side of the screen** is the editor 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 image below explains the elements of the node editor.

Name this node...
Node Name: Name your node in this field.

Customize
Customize Widget:
Turn on Slots & multiple responses.

If bot recognizes:

Enter an intent, entity or context variable...
Condition: Enter the main condition in this field.

Then respond with:

Advanced editor menu: Delete a response or open the Json editor for the response field.

Text
Move:

Enter response text
Response: Enter a response for the condition in this field.

Response variations are set to sequential. Set to random

Add response type

And finally

Wait for user input
Next action menu: Drop down menu to choose which action Watson will take next after it returns a response to the user.

3. Select **Welcome** node to edit it and fill it as below

Field	Value
Name of the node	<i>start of the conversation</i>
If bot recognizes	<i>conversation_start</i>
Watson responses	<i>Hi! I am Watson, nice to meet you</i>

start of the conversation

 Customize





If bot recognizes:




conversation\_start




Then respond with:




 Text 

Move:   

Hi! I am Watson, nice to meet you 

Enter response variation

Response variations are set to **sequential**. Set to [random](#) | [multiline](#) 

 Add response type



4. Select the **anything\_else** node, 3 responses are pre-created.

Anything else

Customize

If bot recognizes:

anything\_else

Then respond with:

Text

Move:

I didn't understand. You can try rephrasing.

Can you reword your statement? I'm not understanding.

I didn't get your meaning.

Enter response variation

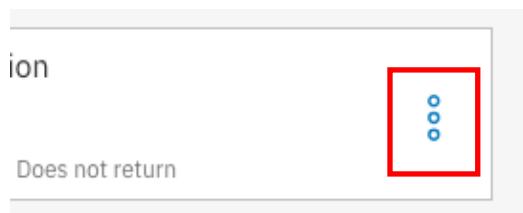
Response variations are set to **sequential**. Set to [random](#) | [multiline](#) ⓘ

+ Add response type

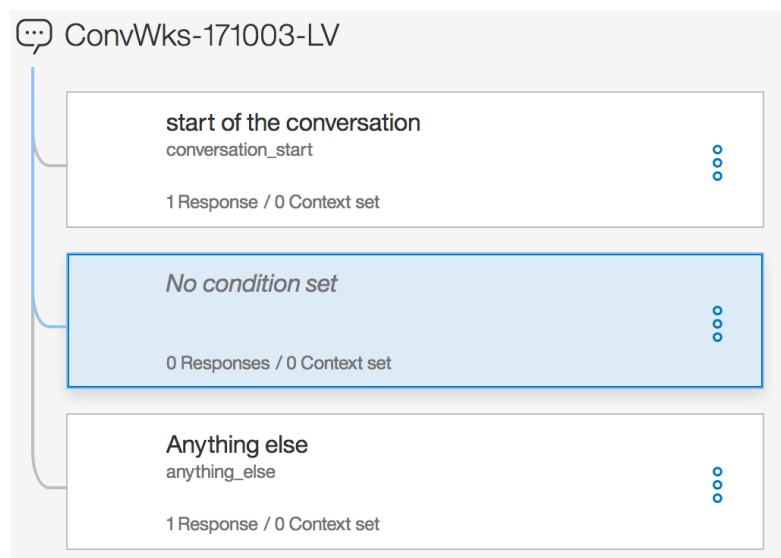
## 21. Create your first nodes

You can create a dialog branch for each of the *intents* you identified as well as the start & end of the conversation. You should also greet the end user as appropriate. For the greeting, you can create a new dialog node to respond to a *greeting* intent. To do this you are going to leverage the intent *#General\_Greetings* that you have imported in earlier steps.

**Note:** When you click on add node button at the top of the tab. The node will be added below the selected node. So best way to add correctly a node is to use the **node menu**.

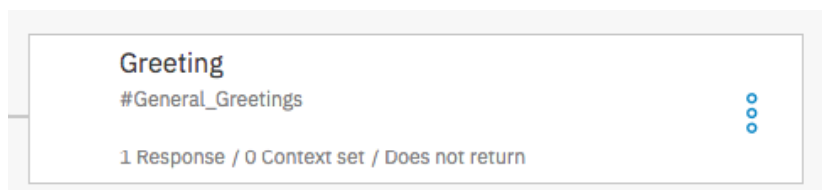


1. Select the **start of the conversation** node
2. Click **node menu**
3. Click **Add node below**



4. In this new node, enter the following values. By setting the condition to an *intent*, you are indicating that this node will be triggered by any input that matches the specified *intent*.

Field	Value
Name of the node	<i>Greeting</i>
If bot recognizes	<i>#General_Greetings</i>
Watson responses	<i>Hi! What can I do for you?</i>



**Note:** The figure above indicates that there is only one condition that is evaluated, one response and zero variable context used.

You will create another dialog branch to respond to the *#hotel\_hours* intent. Because there are multiple possibilities to manage this intent, this branch could be more complex. Right now, we will keep it simple

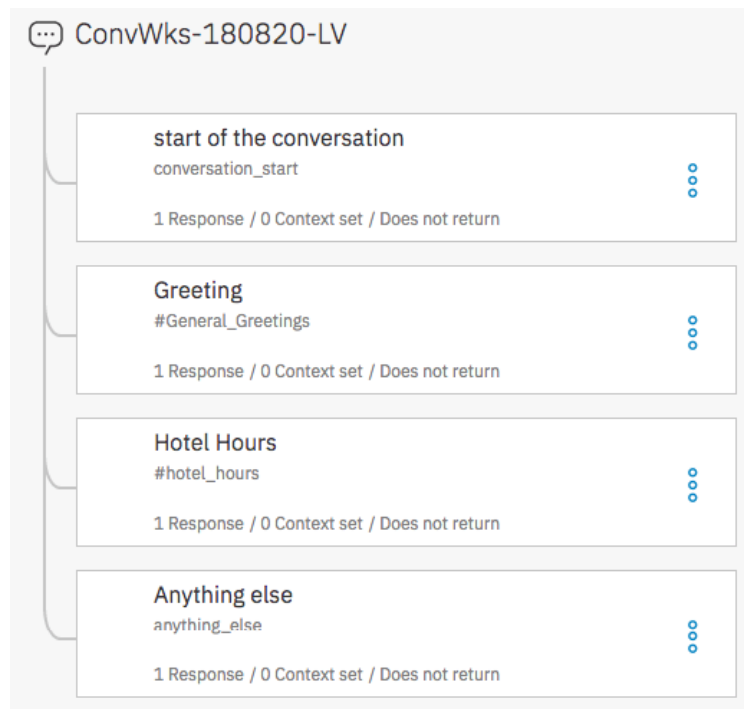
5. Add a node below the greeting node as below

Field	Value
Name of the node	<i>Hotel Hours</i>
Triggered by	<i>#hotel_hours</i>
Watson responses	<i>The @hotel_amenity is open from 6am to 9pm</i>

@hotel\_amenity is the entity captured by Watson. The bot will return this information and demonstrate that it understood the request.

## 22. Test your conversation

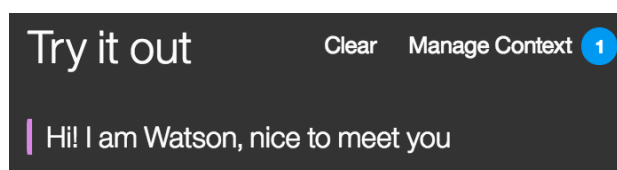
Now, you should have a dialog with four nodes.



it's time to test it.

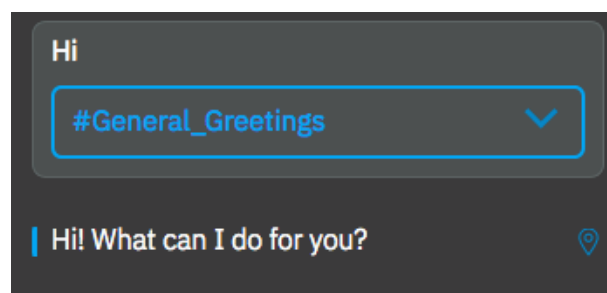
9. Open **Try it out** panel.

A test user interface will immediately launch and, based off the root node you just created, provide a greeting to the end user. (You may see a message that Watson is being trained.)

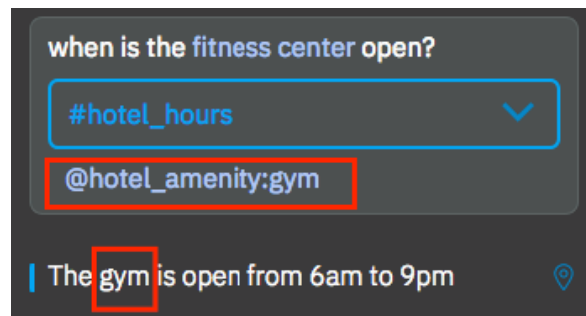


10. Enter **Hi** at the bottom of the chat window.

The bot understands the **#General\_greetings** intent and send you back the right response.

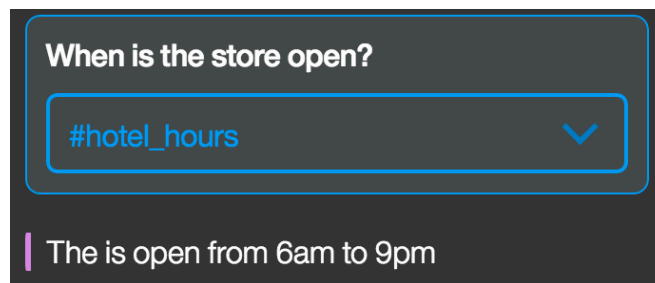


11. Enter *When is the fitness center open?*



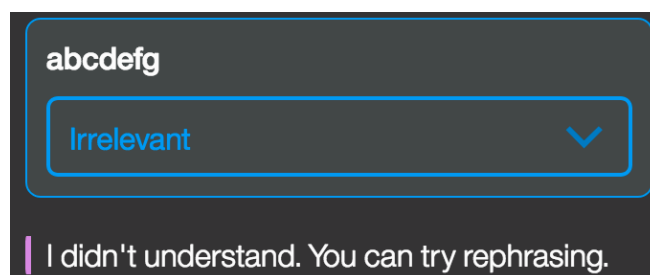
**Note:** Watson should return the response that is matched to #hotel\_hours and @hotel\_amenity:gym

12. Enter *When is the store open?*



**Note:** Watson understood the intent #hotel\_hours. But as *store* is not defined as value of any entities, the system returns the answer without @hotel\_amenity value. During the next lab, we are going to understand how to improve the WA behaviour to manage such behaviour.

13. Enter *abcdefg*



Watson cannot identify anything and executes the **anything\_else** node.

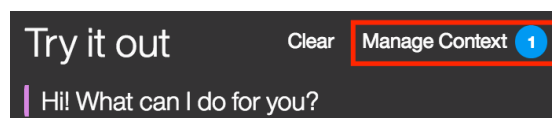
# Enrich your data by using context variables

Watson Assistant allows you to store information. This is done using the 'context'. For example, when asked where something is, you may wish to show a map. So, your dialog can store the location in the context, and your application can use it to show the location on a map. You can set variables in the context using the advanced response editor.

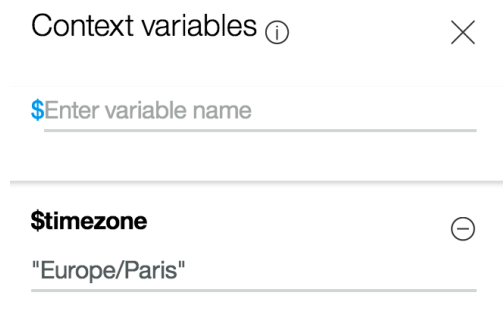
There are two methods to access a context variable:

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

1. To view the context variables that have been set for a conversation, open the **try it out** panel, and click on **Manage Context**



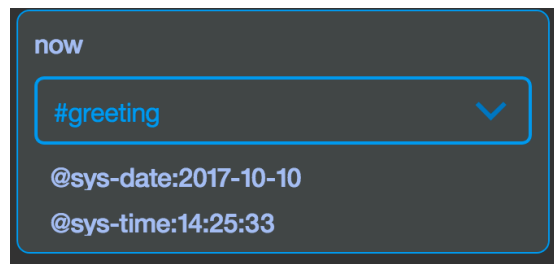
This will open the **Context variables** panel.



The number beside Manage Context tells you how many context variables are set; By default, as you enable `sys_date` and `sys_time` system entities, Watson sets the user's time zone in the `$timezone` context variable.

2. Close **Manage Context**

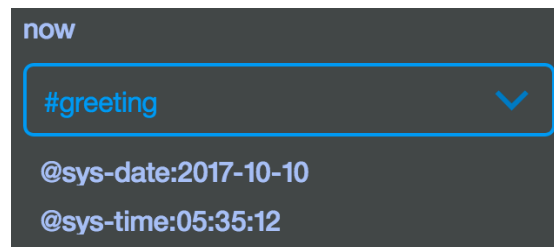
3. In the **try it out** panel, enter *now*



4. Go back to **manage context** panel and update *\$timezone*
5. Replace *Europe/Paris* with *America/Los\_Angeles*

**\$timezone** ⊖  
"America/Los\_Angeles"

6. In the **try it out** panel, enter *now*



You can see that WA applied the new time zone.

7. To avoid any confusion in the time zone, reuse *Europe/Paris* as context variable value for *\$timezone* or your time zone.

### **Important Note:**

Save your workspace to your laptop (Click on Download as JSON on the workspace tile)

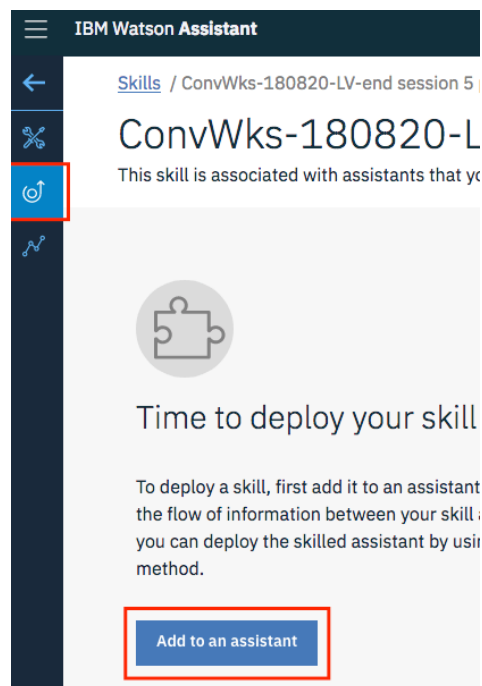
# Create your first Assistant

A Assistant is a virtual container to which you add your skills that enables it to interact with your customers in useful way.

There are two methods to access a context variable:

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

1. Select **Deploy** menu then click **Add to an assistant**



2. Click **Create New**

3. Fill assistant as below

Field	Value
Name	<i>Concierge</i>
Description	<i>For concierge demo</i>
Enable preview link	Keep it selected

4. Click **Create**



## 5. Click **Add Dialog Skill**

Home Skills **Assistants** Instance Conversation-181112-LV [Change](#)


< Assistants

### Concierge

*For concierge demo*

[View API Details](#) [Rename](#) [Delete](#)

#### Time to give your Assistant a Skill [Learn more about skills](#)





**New Dialog Skill**  
Dialog uses Watson natural language processing and machine learning technologies to understand user questions and requests, and address them.

[Add Dialog Skill](#)

#### Integrations

Choose a channel to deploy your Assistant.

[Add integration](#)

 [Preview Link](#) 

6. Select **Add existing skill** then select the skill you have previously created  
Now you assistant to use.

7. You are going to review the sample UI to test your assistant. Click **Preview Link** tile.


< Assistants

### Concierge

*For concierge demo*

[View API Details](#) [Rename](#) [Delete](#)

#### Skill

**ConvWks-181112-LV** 

For demos only



LANGUAGE:	TRAINED DATA:	DATE CREATED:	DATE MODIFIED:
English (US)	0 Intents   0 Entities   0 Dialog Nodes	Mon Nov 12 2018	Mon Nov 12 2018

**LINKED ASSISTANTS:**  
Concierge

#### Integrations

Choose a channel to deploy your Assistant.

[Add integration](#)

 [Preview Link](#) 

8. You can use the URL link to test your Assistant. Click on the hyper link

## Preview Link Integration

**Name**

Preview Link

**Description**

A public link you can share to test your assistant outside of the tooling.

### Try it out and share the link

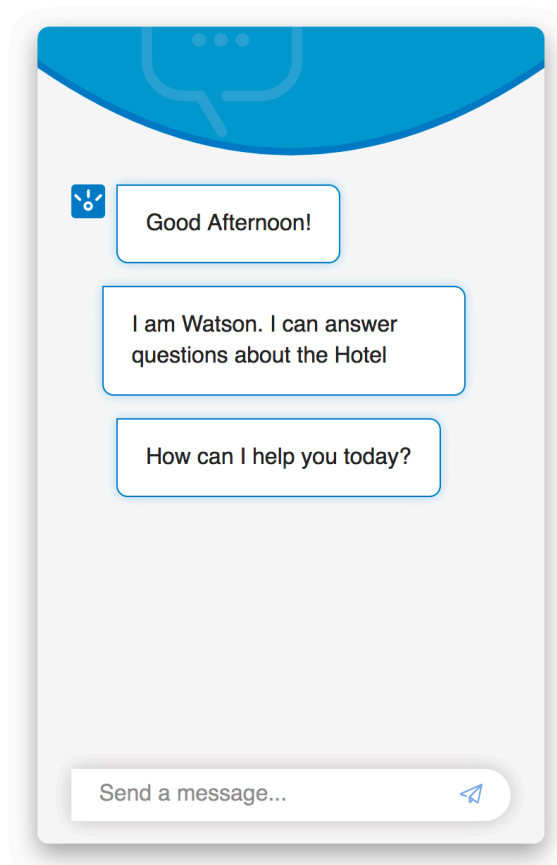
Use of the assistant embedded in this web page incurs billing charges.

- Each message you send to the service through the chat window in the preview web page counts as an API call to the service, and is logged and billed as such.

<https://assistant-chat-us-south.watsonplatform.net/web/public/b39409ae-5734-4721-adb9-50ce47934001>

9. You can use the page to test your Assistant.

Build your own assistant using [IBM Watson Assistant](#)



# END OF LAB