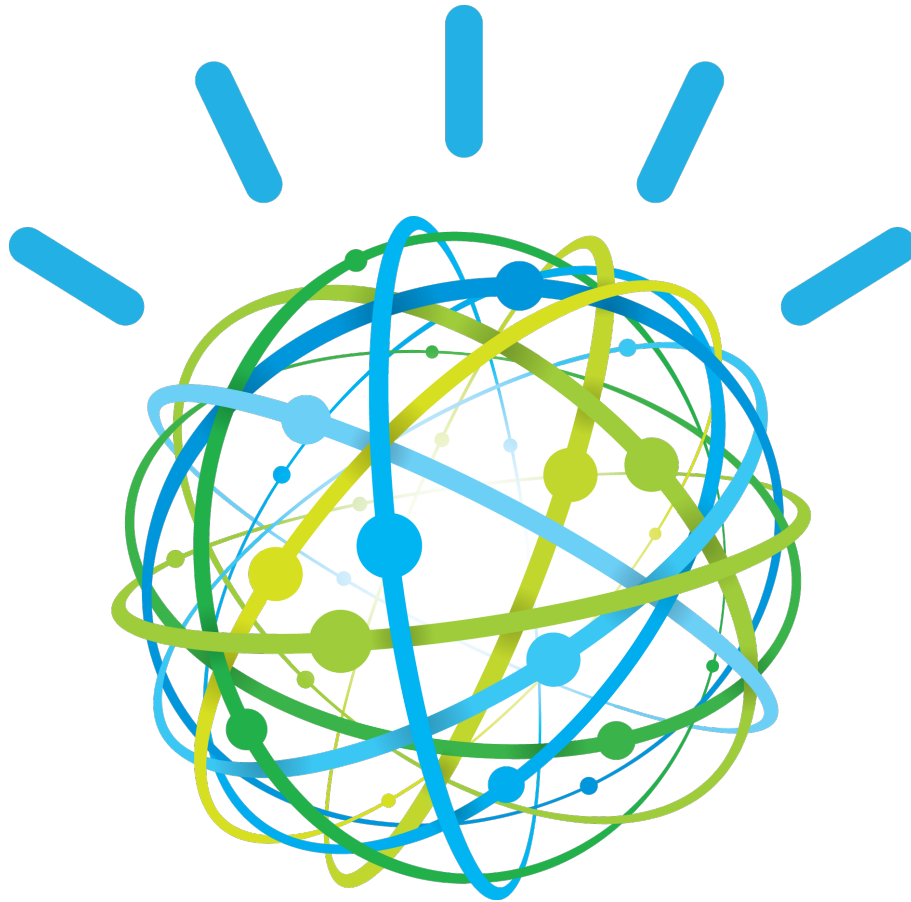


IBM Watson Assistant



**Build your first conversation
Lab Instructions**

Content

Let's get started	3
1. Overview	3
2. Objectives	3
3. Prerequisites	3
4. Scenario	3
5. What to expect when you are done	4
Create an Assistant in IBM Cloud	5
6. Create your Watson Assistant Service	5
7. Create your first Assistant	8
8. Create your first Skill	10
Intents	12
9. Create your first intent	12
10. Import intents	14
11. Test and improve your Intents	16
12. Content Catalog	21
Entities	22
13. Create your first entity	22
14. Import entities	24
15. Test your entities	25
16. Contextual entities	27
17. Entity pattern	32
18. Test your pattern	32
19. Enable system entities	33
20. Test System Entities	34
Conversation Dialog	35
21. Create your first Dialog	35
22. Create your first nodes	40
23. Test your conversation	42
Enrich your data by using context variables	44

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 Assistant 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 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:

- none
- The instructor provided you the link to get labs content. You may download each file individually.

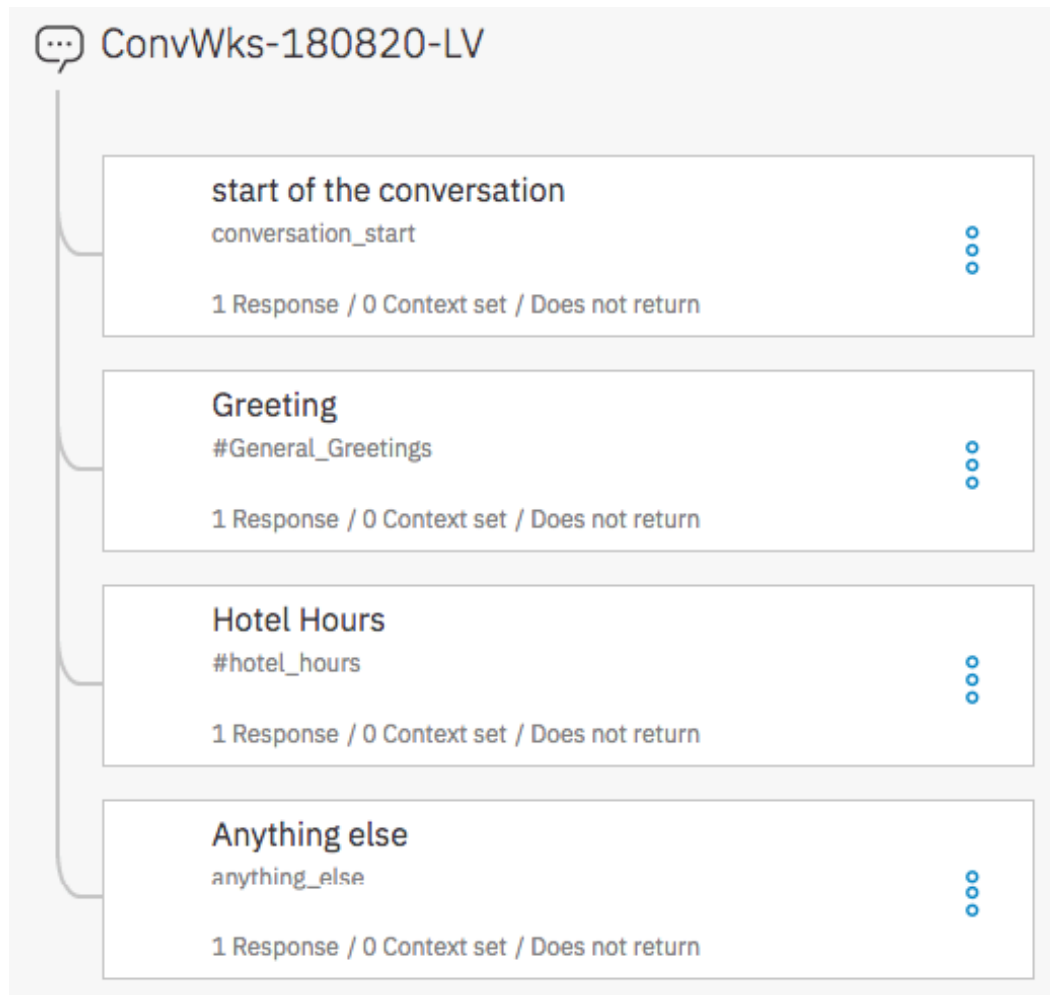
4. Scenario

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
around 23 intents,
around 11 entities.



Create an Assistant in IBM Cloud

6. Create your Watson Assistant Service

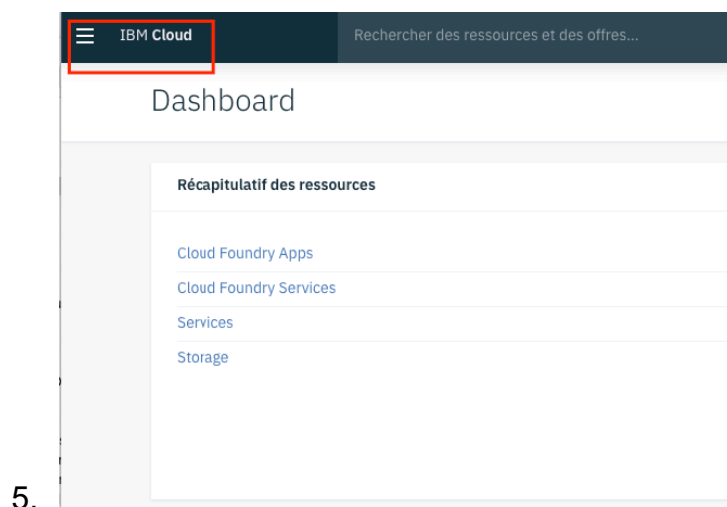
IBM Cloud offers services, or cloud extensions, that provide additional functionality that is ready to use by your application's running code.

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

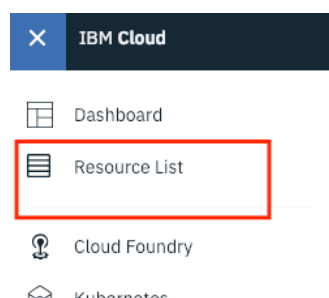
1. In a web browser, navigate to the following URL:

Location	URL
worldwide	https://cloud.ibm.com/login

2. Log in with your IBM Cloud credentials. This should be your IBM ID.
3. You should start on your dashboard that contains metrics related to your IBM Cloud activities.
4. Click on the hamburger menu (up right of the page)



6. Then select **Resource List**

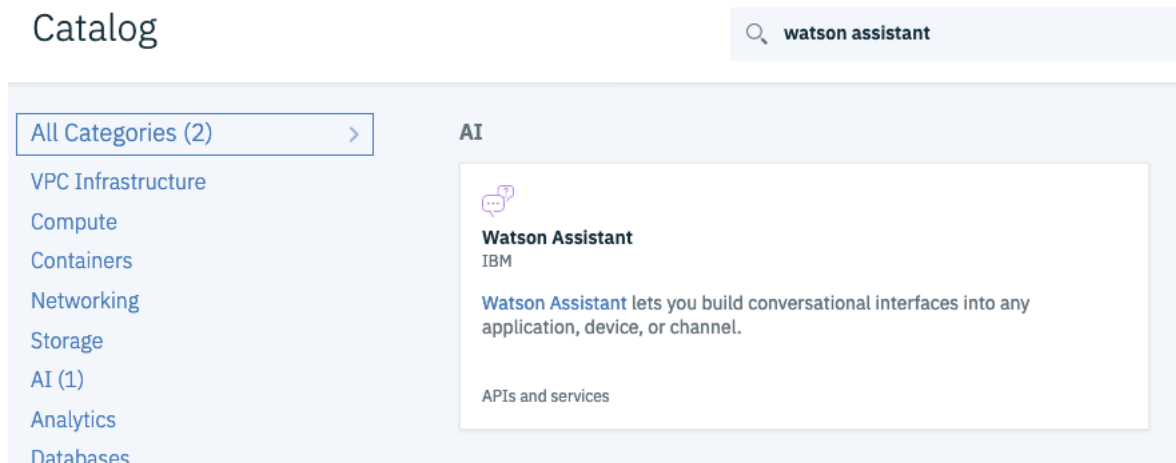


Then you should get the list of all your applications and services. It could be empty.

7. Click on **Create resource** (left right of the page)

Create resource

1. Enter *Watson Assistant* as Filter



2. Click on the **Watson Assistant** service tile.
3. 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!
4. At the top, you can enter information for your new service. Select the location *Dallas*.
5. In the middle, you can click on a pricing plan 0 *Plus Trial* to select it
6. At the bottom, fill out the fields as follows, then click **Create** at the bottom.

Field	Value
Service name	<i>Conversation-XXXXXX-LV</i>
Resource group	default

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

7. Then click **Create** (top right).

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 shows the IBM Watson Assistant console interface. On the left is a sidebar with navigation links: **Manage**, Service credentials, Plan, and Connections. The main content area has a header with 'Watson Services / Assistant /' and a breadcrumb. Below this, it says 'Conversation-181112-LV' with a status bar showing '0.62% Used | 9938 Api calls available' and a 'Details' link. Underneath, it lists 'Resource Group: default' and 'Location: Dallas'. The main section is titled 'Get started by launching the tool.' and includes a 'Launch tool' button, a 'Getting started tutorial' link, and an 'API reference' link. To the right of this section, it says 'Plan: Lite' with an 'Upgrade' link. Below this is a 'Credentials' section with a 'Show Credentials' link and an eye icon. A red rectangle highlights the 'API Key' and 'Url' fields. The 'API Key' field contains a masked value (dots) and a copy icon. The 'Url' field contains the value 'https://gateway.watsonplatform.net/assistant/api' and a copy icon.

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.

Note :

You can provision only one plan Lite or Plus Trial Watson Assistant service. So, if you have already a Lite plan, go ahead and delete the Lite version.

8. On the left side of the **Manage** Tab, click the **Launch Watson Assistant** button.

Launch Watson Assistant

Your first step in the Watson Assistant tool is to create an Assistant or a Skill.

7. Create your first Assistant

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

1. On the landing page of the service, click **Create Assistant**
2. Fill assistant as below

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

Create Assistant

Create an assistant to deploy the skill that addresses your customers' goals.

Name

Name your assistant, for example **Banking** or **Customer Care**.

Concierge

Description (optional)

For concierge demo

Preview Link ⓘ

☒ Enable Preview Link

Create assistant

3. Click **Create Assistant**


Concierge

For concierge demo

Skills

A dialog skill provides specific responses you've created. You can use a search skill to provide answers from linked documents or web pages. Choose one or both skills for your assistant. [Learn more](#)

Dialog




Add a dialog skill to design your conversation flow

Dialog skills use Watson natural language processing and machine learning technologies to understand user requests and respond appropriately.

[Add dialog skill](#)

Search Plus



Extend your conversation flow with a search skill


Once you've created a dialog skill, you can add a search skill. Search skills find relevant information from external data sources and provide responses to user queries that the dialog is not designed to handle.

[Add search skill](#)

Integrations

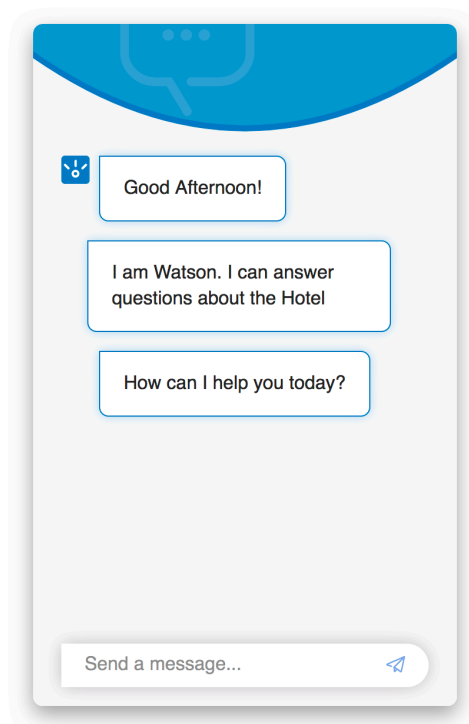
Choose a channel to deploy your Assistant.

[Add integration](#)

 [Preview Link](#)

In the integration panel, WA has created a web page (preview link) to test your conversation. Below a copy of the web page.

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



4. Click **Add Dialog Skill**

8. Create your first Skill

To use the new Assistant, 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. Select **Create skill** tab
2. In the *Create a skill* wizard, enter the following values and click **Create dialog skill**

Field	Value
Name	<i>My Concierge Skill</i>
Description	<i>For demos only</i>
Language	<i>English (U.S.)</i>

Add Dialog Skill

Add an existing skill, or create a new dialog skill to add to your assistant.

Add existing skill

Create skill

Use sample skill

Import skill

Name

Name your skill, for example **Account application** or **Personal banking**.

My Concierge Skill

Description (optional)

For demos only

Language

English (US)

Create dialog skill

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

Concierge

For concierge demo

Skills

A dialog skill provides specific responses you've created. You can use a search skill to provide answers from linked documents or web pages. Choose one or both skills for your assistant. [Learn more](#)

Dialog

My Concierge Skill

For demos only

LANGUAGE:	TRAINED DATA:	VERSION:	CREATED:	UPDATED:
English (US)	0 Intents 0 Entities 0 Dialog Nodes	Development	Sep 24, 2019 2:24 PM CEST	Sep 24, 2019 2:24 PM CEST

LINKED ASSISTANTS (1): Concierge

3. To open the skill page, click on the tile **My Concierge Skill**

Intents

9. 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 skill.
2. On the *Intents* tab, click **Create intent** in the dialog window.

What is an intent?

An intent is a collection of user statements that have the same meaning. By creating intents, you train your assistant to understand the variety of ways users express a goal. [Learn more](#)

You will find some pre-made intents in the content catalog. [Browse content catalog](#)

Create intent View intent recommendations Import intents

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

← | Create intent

Intent name
Name your intent to match a customer's question or goal






hotel_locations

Description (optional)
Add a description to this intent

Create intent

- 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>

 #hotel_locations
 Last updated: a few seconds ago
 




Intent name
 Name your intent to match a customer's question or goal

Description (optional)

User example
 Add unique examples of what the user might say. (*Pro tip:* Add at least 5 unique examples to help Watson understand)

☒ Annotate entities
 BETA


<input type="checkbox"/> User examples (3) ▲	Added	Conflicts (0) ▼
<input type="checkbox"/> Do you have a sauna?	a few seconds ago	
<input type="checkbox"/> Is the pool outside ?	a few seconds ago	
<input type="checkbox"/> Where is the gym?	a few seconds ago	

- 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.

10. Import intents

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

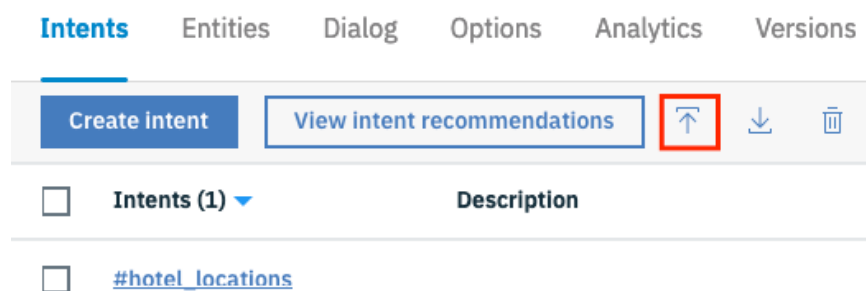
In order to import a file successfully, it must include only two columns: one for user examples and one for intents.

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.

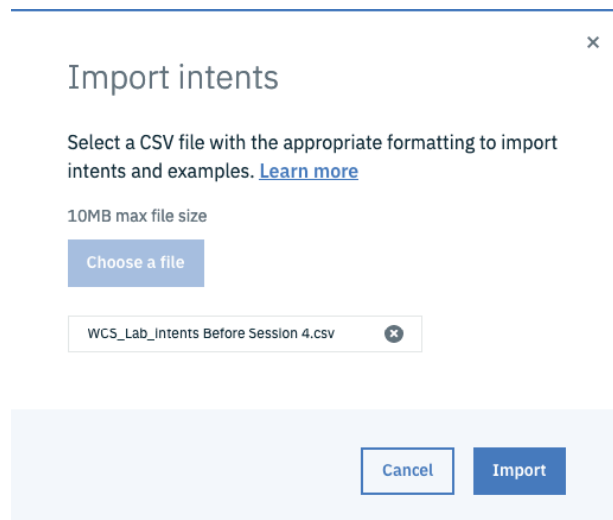
- Copy the end-user examples into a single column.
- Label each end-user example with its assigned intent in the column beside it.
- Do not insert any column headings for the intent import file.
- Use a text editor to make sure that your file is UTF-8 and saved as a .csv file. We suggest Numbers for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



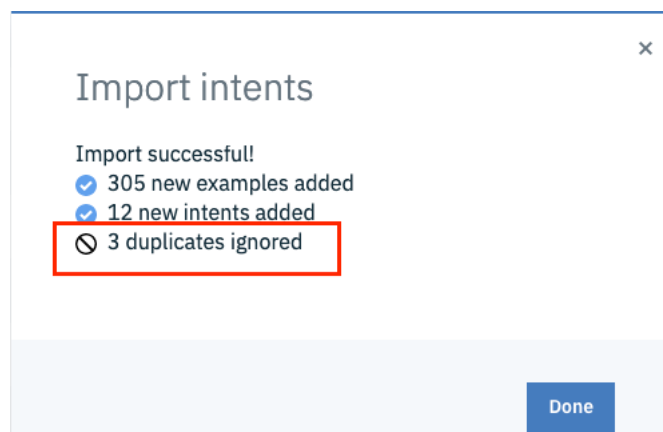
2. Drag [WCS_Lab_Intents Before Session 4.csv](#) into the **Choose a file** box or click **Choose a file** to browse your computer.



3. Click **Import**

Note: Notice that the Watson Assistant 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:

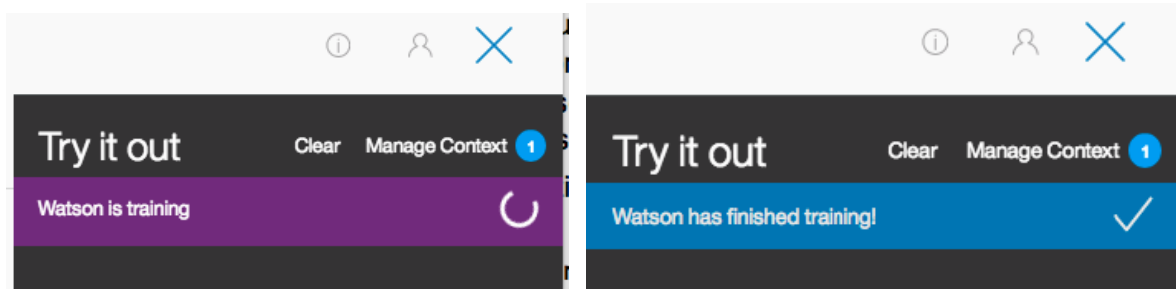


4. Click **Done**

11. Test and improve 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):



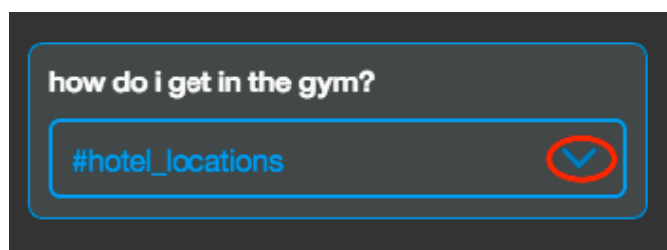
You cannot perform the **Try it out** activities in this section until Watson has finished training on your recent changes. When Watson is finished, the purple box with the text, "Watson is training" will no longer appear in your "Try it out" panel. You may also notice a message stating that, "Watson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

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

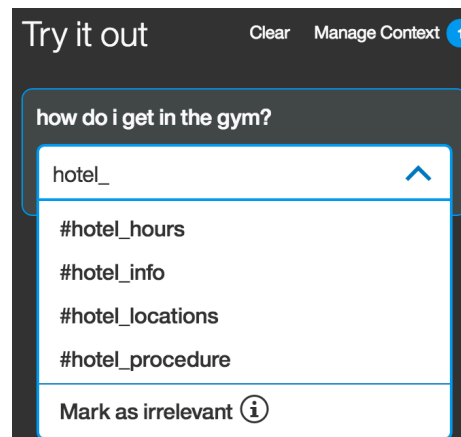
Note: Notice that Watson assigns this question to the intent `#hotel_locations`. The `#hotel_locations` intent is meant for questions that refer to finding directions for locations within the hotel. In this case, the user wants to know how to get in the gym, not how to get to the gym.

Let's change the intent for the user example.

3. Click the arrow next to `#hotel_locations`



4. Type *hotel_info*



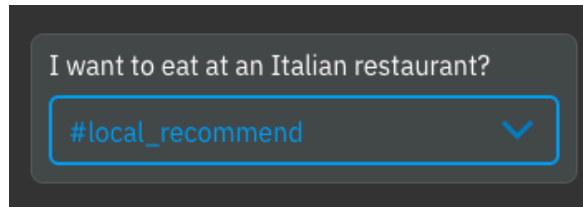
5. Select the intent *#hotel_info* that appears in the drop-down box

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. As this user's example has been used in the intent *#hotel_locations* this update should cause a conflict between the 2 intents. In another lab you are going to use 'Watson Conflict Resolution' feature to fix it.

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

Note: Notice that Watson assigns this question the intent `#Local_recommend`. (the classification could differ). This intent is meant for questions that refer to finding recommendation for sightseeing, restaurant in the hotel. 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.

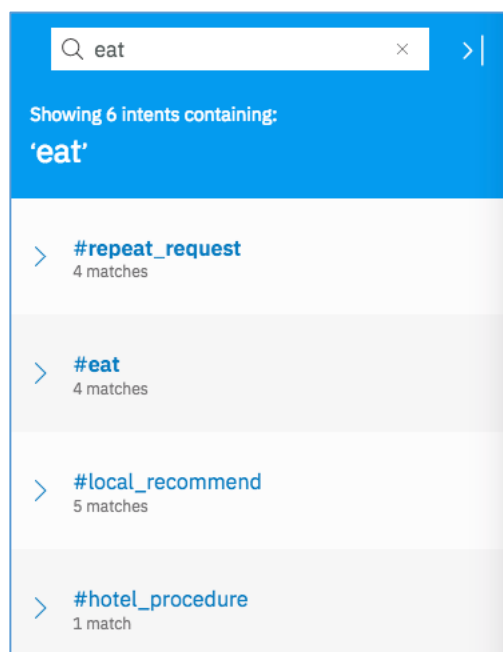
7. Close the **Try it out** Panel

8. Top right, click on the search icon

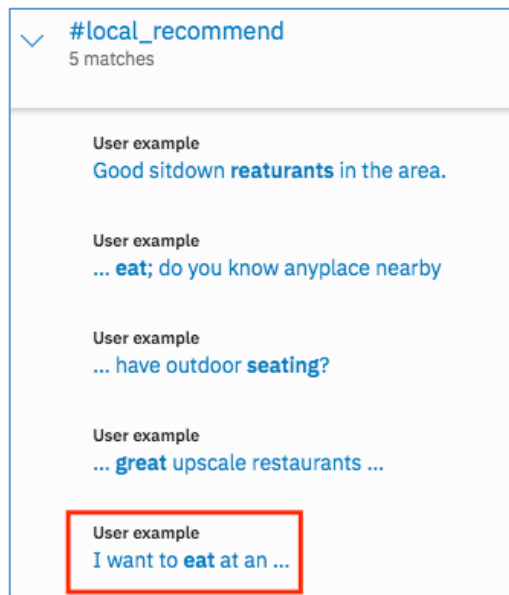


9. Enter *eat* in the search field (retry this search if indexing is needed)

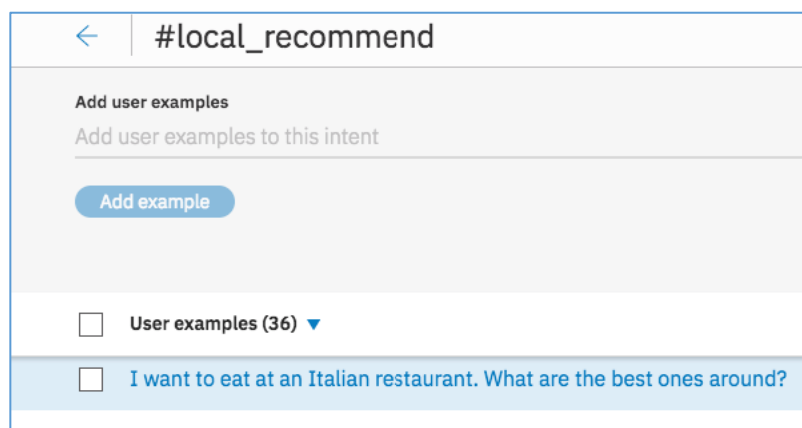
The solution returns all intents including the 'eat' word.



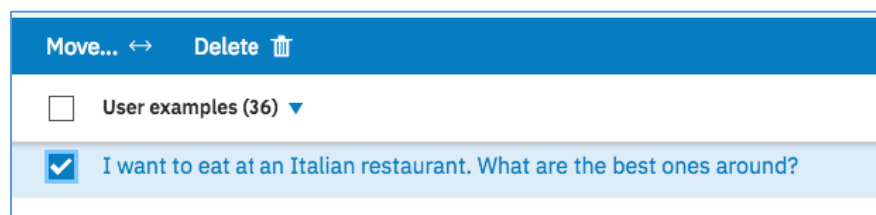
10. As you are looking for a user example which belongs to `#local_recommend`, expand it.



11. The last one looks like the one we used during the test, click on it. On the left, Watson opens the right page and highlights the right user example.



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



13. Enter *eat* as intent

Do you want to move 1 example?

You are about to move 1 example.

Search for a destination Intent or create a new one:

#eat
#exit
#feedback
#hotel_hours
#hotel_info

Move examples

14. Click on **Move examples**

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.

12. Content Catalog

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

You are going to add general conversation topics intents.

1. On the **Content Catalog** tab, click Add to skill to add *General* and Bot Control categories into your training set.

Intents Entities Dialog Options Analytics Versions Content Catalog

Get started faster by adding existing intents from the content catalog. These intents are trained on questions that customers commonly ask.

Category	Description	Intents	
Banking	Basic transactions for a banking use case.	13	+ Add to skill
Bot Control	Functions that allow navigation within a conversation.	9	+ Add to skill
Customer Care	Understand and assist customers with information about themselves and your business.	18	+ Add to skill
eCommerce	Payment, billing, and basic management tasks for orders.	14	+ Add to skill
General	General conversation topics most users ask.	10	+ Add to skill
Insurance	Issues related to insurance policies and claims.	12	+ Add to skill
Mortgage	Common questions related to the mortgage industry	20	+ Add to skill
Telco	Questions and issues related to a user's telephony service, device, and plan.	21	+ Add to skill
Utilities	Help a user with utility emergencies and their utility service.	10	+ Add to skill

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

13. 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 on **Entities** tab at the top of your skill.
2. Click **Create entity**.

What is an entity?

Entities are like nouns or keywords. By building out your business terms in entities your assistant can provide targeted responses to queries. [Learn more](#)

You can also enable pre-built system entities to capture common phrases such as dates, times and numbers.

Create entityImport entities

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.)

←

@restaurant

Entity name

Name your entity to match the category of values that it will detect.

@ restaurant

Value

pizza_restaurant

Synonyms

pizza restaurant

+

Add value

Recommend synonyms

Dictionary (0)

Annotation (0) BETA

☐

Values (0) ▲

Type

Field	Value	Synonym
Entity name	<i>restaurant</i>	<< N/A >>
Value	<i>pizza_restaurant</i>	<i>pizza restaurant, pizza, pizzeria</i>
	<i>french_restaurant</i>	<i>french restaurant, brasserie</i>

←

@restaurant

Last updated: a few seconds ago

↓

🗑

🔍

Try it

Entity name

Name your entity to match the category of values that it will detect.

@ restaurant

Value

Type value here, e.g. Checking

Synonyms

Synonyms

Type synonym here, e.g Deposit

Add value

Recommend synonyms

Dictionary (2)

Annotation (0) BETA

☐

Values (2) ▲

Type

☐

french_restaurant

Synonyms

french restaurant, brasserie

☐

pizza_restaurant

Synonyms

pizza restaurant, pizzeria

- Fuzzy Matching** must be on
- When finished, click on the arrow icon

14. 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** (icon)
2. Drag [WCS_Lab_Entities.csv](#) into the **Choose a file** box or click **Choose a file** to browse your computer.

Import entities

Select a CSV file with the appropriate formatting to import entities, values and synonyms. [Learn more](#)

10MB max file size

Choose a file

WCS_Lab_entities.csv

Cancel

Import

3. Click **Import**

Import entities

Import successful!

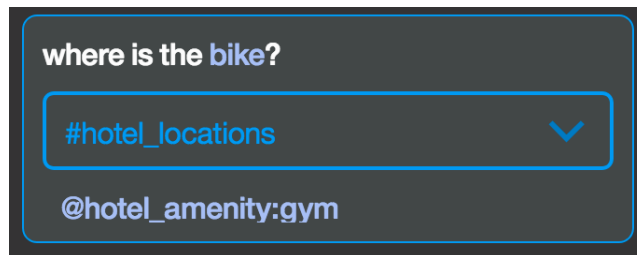
- ✓ 10 new entities added
- ✓ 0 new patterns added
- ✓ 87 new synonyms added
- ✓ 54 new values added
- ✗ 7 duplicates ignored

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, values, synonyms were ignored.

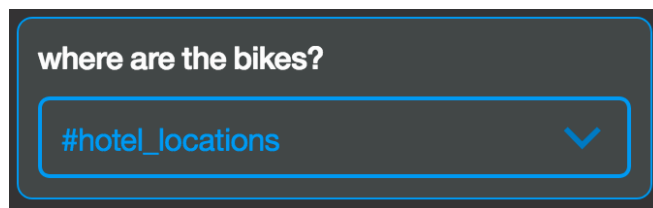
15. Test your entities

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*).

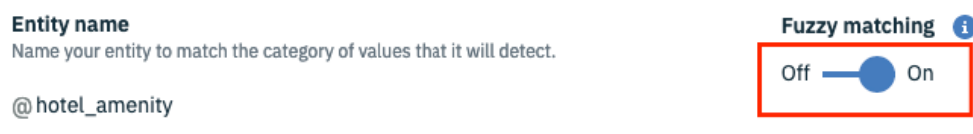
Type *where are the bikes?*



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.

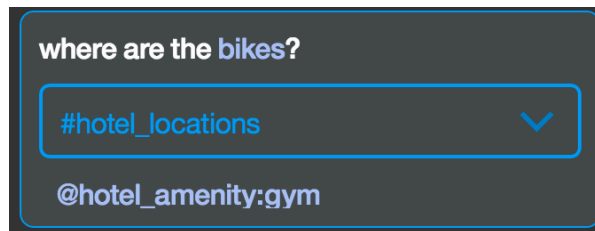
Let's turn on fuzzy matching to see how it improves entity recognition.

3. Open the *Hotel_amenity* entity and turn on **Fuzzy Matching**.



4. To complete the training, enable the **Fuzzy Matching** for the entities *@pizza-toppings* and *@pizza_notoppings*

5. 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*.

16. 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 with more olives and without anchovy* at the bottom of the chat window.

The bot understands the #order_pizza intent 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.

- Go to **intents** tab and select [#Order_pizza](#)
- Enable the option **Annotate entities**

<input checked="" type="checkbox"/> Annotate entities BETA i		
<input type="checkbox"/> User examples (17) ▲	Added	Conflicts (0) ▼
<input type="checkbox"/> a large vegetarian without onions	20 hours ago	
<input type="checkbox"/> Can I get a hawaiian pizza with no pineapple	20 hours ago	
<input type="checkbox"/> Can I get a pizza margarita ; small please	20 hours ago	
<input type="checkbox"/> Can I get a pizza with extra red peppers but no green peppers	20 hours ago	

- In the 4th example, highlight the group *red peppers* the enter [@pizza_toppings](#)
-

<input type="checkbox"/> Can I get a pizza with extra red peppers but no green peppers	20 I
<input type="checkbox"/> can I have a carnivore pizza	20 I
<input type="checkbox"/> can I have a pizza with oni	20 I
<input type="checkbox"/> Can I order a pizza with m	20 I

Enter a search to filter by entity name

@pizza_to pizza_toppings

@pizza_toppings (add new value)

@pizza_to (create new entity)

We identify the toppings requested by the user.

7. Annotate the other entities like below.

☐ User examples (17) ▲

☐ a large vegetarian without onions

☐ Can I get a hawaiian pizza with no pineapple

☐ 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 with more cheese

☐ Can I order a pizza without pineapple

☐ Can you deliver a vegetarian pizza

☐ 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

☐ I'd like a large hawaiian with extra pineapple

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

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

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

8. Annotate the intent using @pizza_notoppings entities like below.

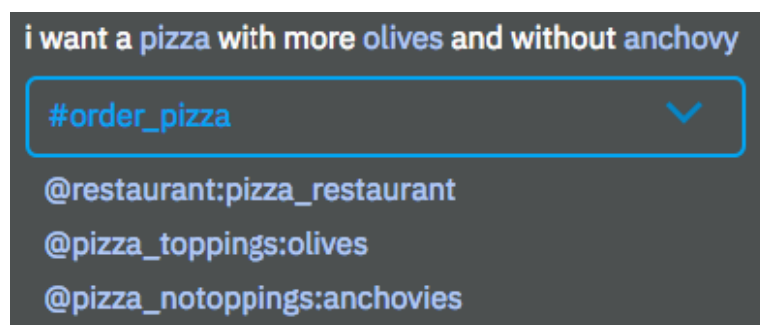
<input type="checkbox"/>	User examples (17) ▲
<input type="checkbox"/>	a large vegetarian without onions
<input type="checkbox"/>	Can I get a hawaiian pizza with no pineapple
<input type="checkbox"/>	Can I get a pizza margarita; small please
<input type="checkbox"/>	Can I get a pizza with extra red peppers but no green peppers
<input type="checkbox"/>	can I have a carnivore pizza but no chicken
<input type="checkbox"/>	can I have a pizza with onions, beef, chicken and extra cheese
<input type="checkbox"/>	Can I order a pizza with more cheese
<input type="checkbox"/>	Can I order a pizza without pineapple
<input type="checkbox"/>	Can you deliver a vegetarian pizza
<input type="checkbox"/>	I'd like pepperoni pizza
<input type="checkbox"/>	I don't want red peppers or onions on my vegetarian pizza
<input type="checkbox"/>	I love anchovies so please send me a pizza full of them
<input type="checkbox"/>	I want to order a large pizza
<input type="checkbox"/>	I want to order a pizza
<input type="checkbox"/>	I'd like a large hawaiian with extra pineapple
<input type="checkbox"/>	I'd like a small margherita but please no anchovies
<input type="checkbox"/>	medium pizza please and I don't want any red peppers

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

Dictionary (11)	Annotation (9) <small>BETA</small>
<input type="checkbox"/> User Examples (9)	Intent
<input type="checkbox"/> Can I get a pizza with extra red peppers but no green peppers	#order_pizza
<input type="checkbox"/> I'd like a small margherita but please no anchovies	#order_pizza
<input type="checkbox"/> I don't want red peppers or onions on my vegetarian pizza	#order_pizza
<input type="checkbox"/> can I have a carnivore pizza but no chicken	#order_pizza
<input type="checkbox"/> Can I order a pizza without pineapple	#order_pizza
<input type="checkbox"/> I don't want red peppers or onions on my vegetarian pizza	#order_pizza
<input type="checkbox"/> Can I get a hawaiian pizza with no pineapple	#order_pizza
<input type="checkbox"/> a large vegetarian without onions	#order_pizza

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

10. Enter *I want a pizza with more olives and without anchovy*



So, pretty cool!

17. Entity pattern

1. Go back to the **entities** tab, Click **Create entity**.
2. Enter *@pattern* as entity name
3. Click **Create entity**
4. Enter
as value: *email address*
as type select **Patterns**
as pattern: *\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b*
5. Don't forget to disable the Fuzzy matching as we want WA to return the exact text defined by the regular expression.

Entity name
Name your entity to match the category of values that it will detect.

@ pattern

Fuzzy matching Off On

Value
email address

Patterns
Patterns

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

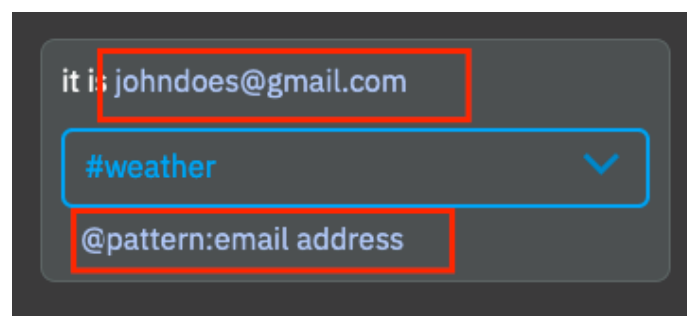
Type pattern here, e.g. `{\d{3}}`

Add value

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

18. Test your pattern

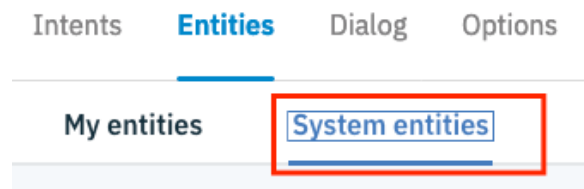
1. Open the **try it out** panel, wait until Watson is done training.
2. Type: *it is johndoes@gmail.com*



Watson extracts *johndoes@gmail.com* as an entity email address.

19. 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 skill.



you will get

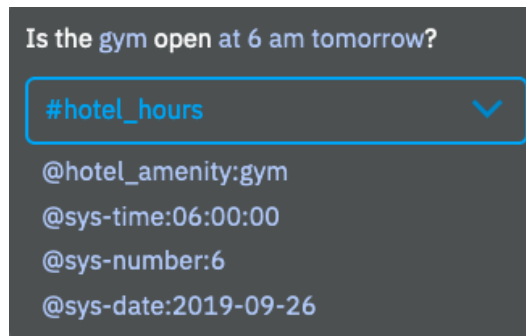
- **@sys-number:** detects numbers that are written using either numerals or words. In either case, a numeric value is returned.
- **@sys-currency:** detects monetary currency values that are expressed in an utterance with a currency symbol or currency-specific terms. A numeric value is returned.
- **@sys-percentage:** 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 ^{BETA}	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 ^{BETA}	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

20. Test System Entities

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

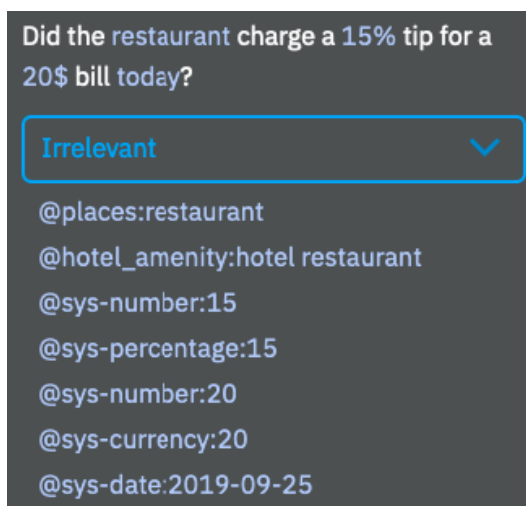


Note: notice that the time and date are recognized

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

tomorrow has been captured **2019-09-26** (yyyy-mm-dd) which is the format the current date is Sept the 25th.

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.

Note: Now, you have the possibility to evaluate new system entities classification. To do this; just go to the **Options** tab, select **system entities** and enables **Try beta**. Then you can retry the previous tests to see the differences. The most valuable interests of this new system entities are the list of methods you can leverage to extract useful details such as the day of the week for an object date.

Conversation Dialog

21. Create your first Dialog

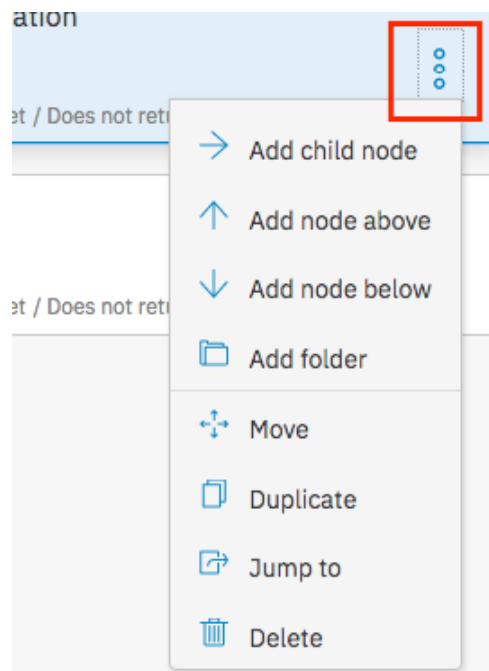
To use the new conversation, you will need to train it with the intents, entities, and/or dialog nodes relevant to your application's use case. A dialog uses the intent and entity that have been identified, plus context from the application, to interact with the user and provide a response.

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

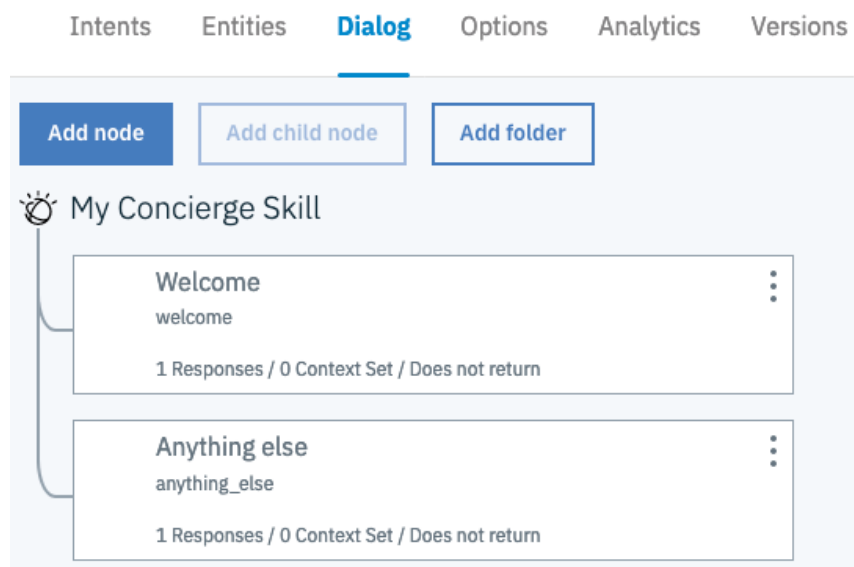
A **condition** is the portion of the dialog node that determines whether the node is used in the conversation. Conditions can be defined using intents, entities, context variables, and special conditions. The Spring Expression (SpEL) language is used to write valid expressions for conditions.

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



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.

- 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.

The image shows a screenshot of the Watson Assistant node editor interface. It consists of several sections with callout boxes explaining their functions:

- Node Name:** A field at the top labeled "Node Name: Name your node in this field." with a "Customize" button and a gear icon to its right.
- Condition:** A section titled "If assistant recognizes:" with a callout box stating "Condition: Enter the main condition in this field." and an input field labeled "Enter condition" with a clear button (X).
- Advanced editor menu:** A callout box pointing to a menu icon (three dots) next to the "Then respond with" label, stating "Advanced editor menu: Delete a response or open the Json editor for the response field."
- Response:** A section titled "Then respond with" with a dropdown menu currently set to "Text". A callout box states "Response: Enter a response for the condition in this field." and shows the input area for "Enter response text". Below the input area, it says "Response variations are set to **sequential**. Set to [random](#) | [multiline](#) [Learn more](#)". At the bottom of this section is a button "Add response type" with a plus icon.
- Next action menu:** A section titled "And finally:" with a dropdown menu currently set to "Wait for user input". A callout box states "Next action menu: Drop down menu to choose which action Watson will take next after it returns a response to the user."

Optionally, add a node purpose summary that can be displayed to users. ⓘ

Disambiguation : text display to the user.

🔒 Status: this node will not be used for features that require displaying an external node name unless an external node name is provided.

2. 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 assistant recognizes:

conversation_start ⊗ ⊕

Then respond with ⋮

Text ⌵ ⌶ 🗑️

Hi. I am Watson, nice to meet you ⊖



Enter response variation

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


Add response type ⊕


3. Select the **Anything else** node, 3 responses are pre-created.


Anything else


Customize  


If assistant recognizes:




anything_else 



Then respond with 



Text 

I didn't understand. You can try rephrasing.

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

I didn't get your meaning.

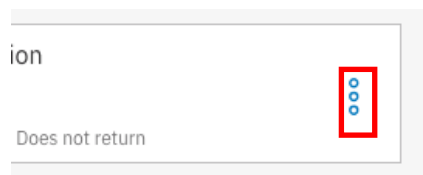
Enter response variation

You may edit these if you wish.

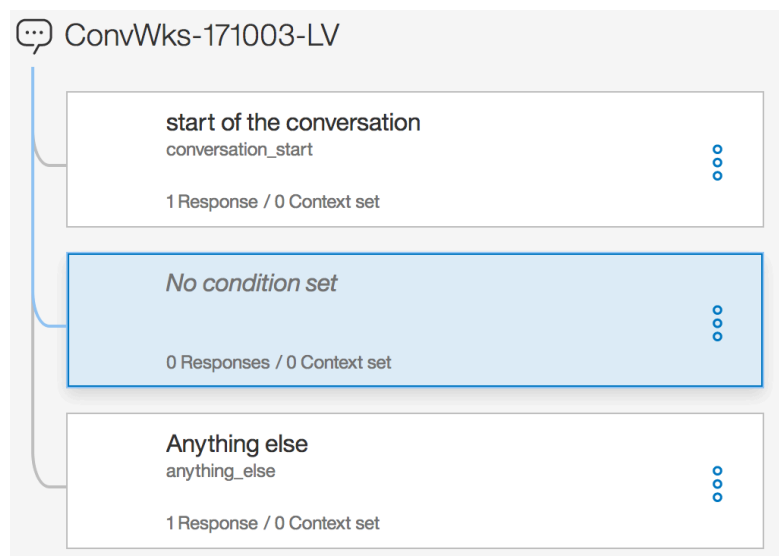
22. Create your first nodes

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**.

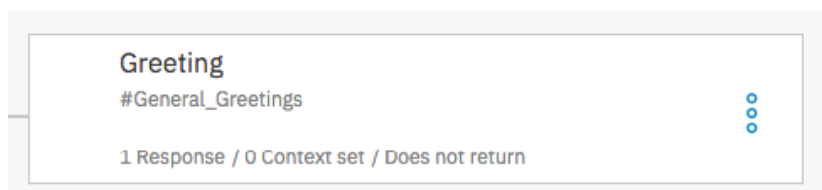


1. On the **start of the conversation** node, click 3 node **node menu**
2. Click **Add node below**



3. 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 should be more complex. Right now, we will keep it simple

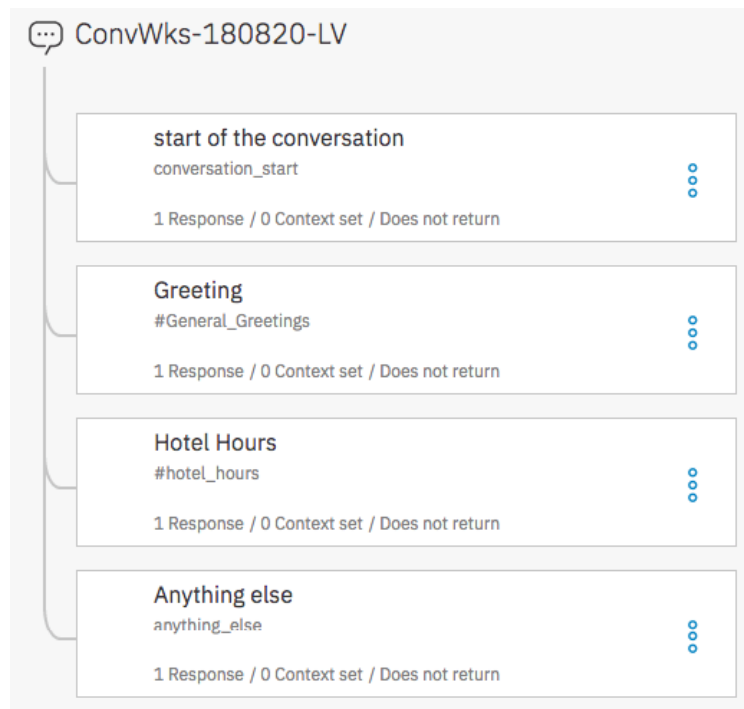
4. Add a node below the greeting node as below

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 6 am to 9 pm</i>

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

23. Test your conversation

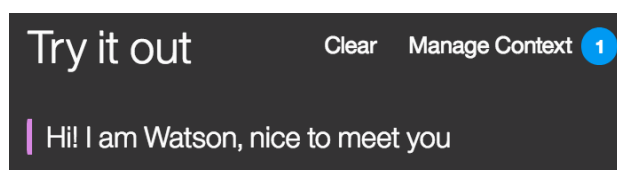
Now, you should have a dialog with four nodes.



it's time to test it.

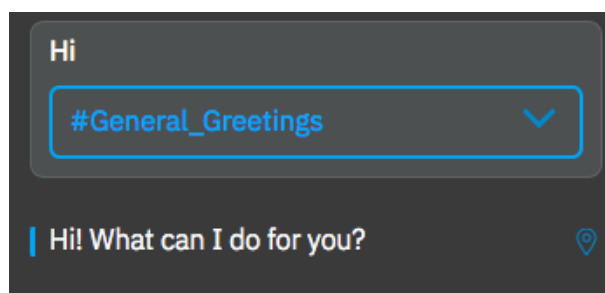
11. Open **Try it out** panel.

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.)

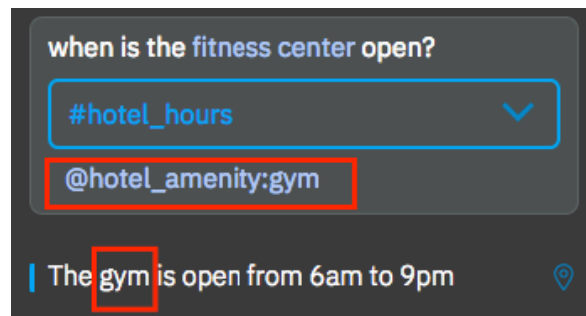


12. Enter **Hi** at the bottom of the chat window.

The bot understands the #General_greetings intend and send you back the right response.

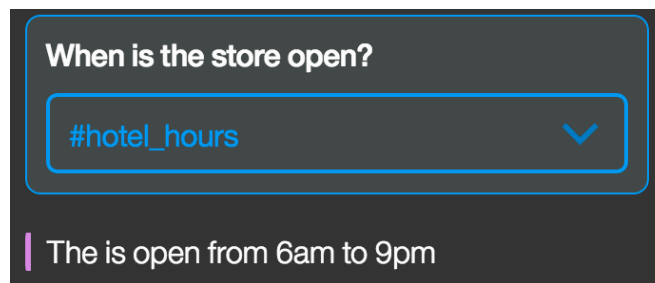


13. Enter *When is the fitness center open?*



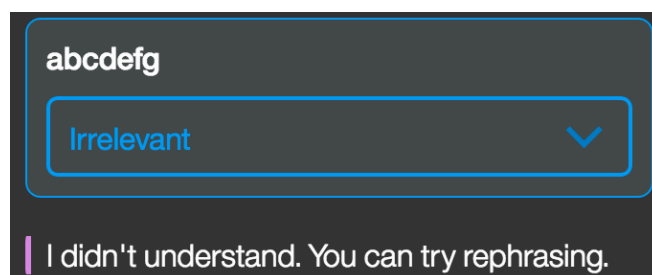
Note: Watson should return the response that is matched to #hotel_hours and @hotel_amenity:gym

14. 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.

15. Enter *abcdefg*



Watson cannot identify anything and executes the **anything_else** node.

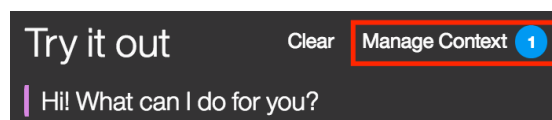
Enrich your data by using context variables

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.

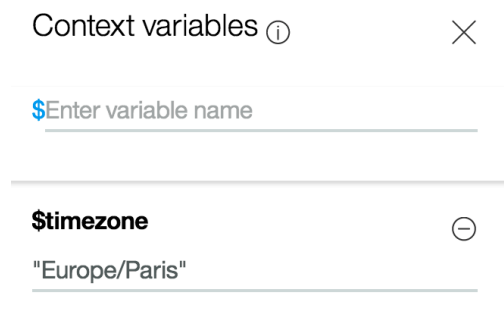
There are two methods to access a context variable:

- 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**



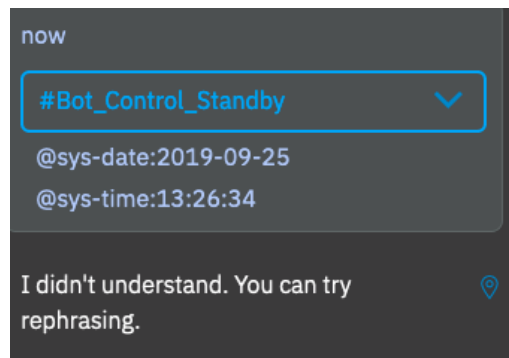
This will open the **Context variables** panel.



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**

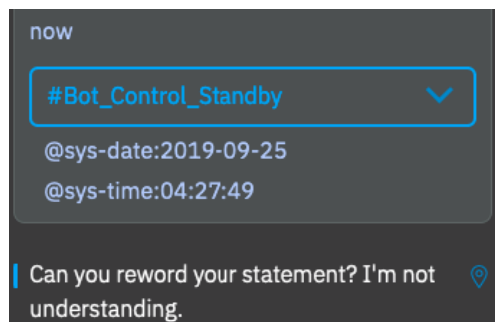
3. In the **try it out** panel, enter *now*



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

END OF LAB