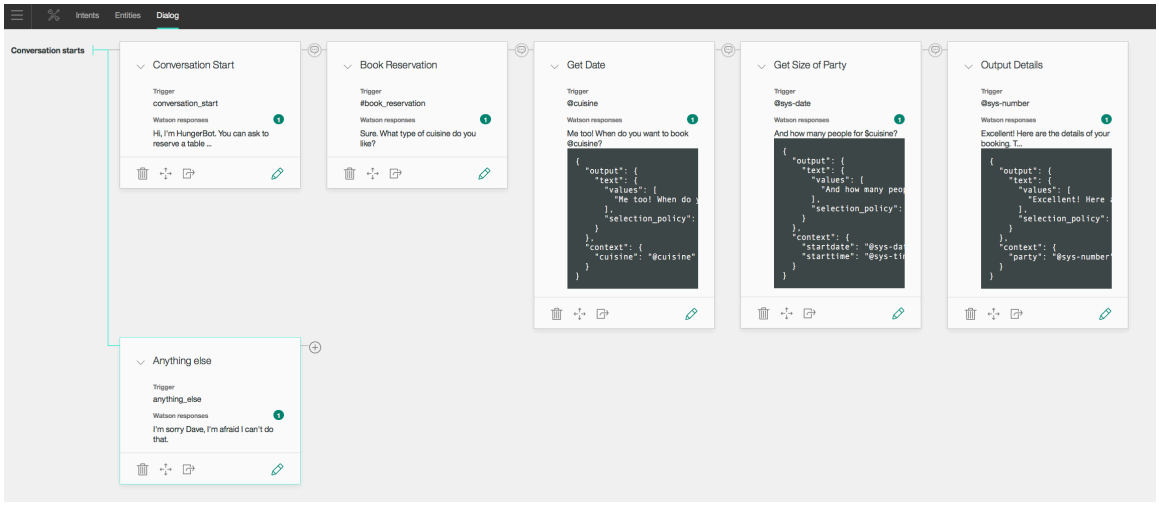
Watson Conversation Chatbot in Node.js

Hands-On Lab

JeanCarl Bisson | jbisson@us.ibm.com | @dothewww



Design intents and entities of a custom chatbot (pg. 2), to use to train the Watson Conversation service (pg.4), and deploy the sample Node.js application as a starter (pg. 12).

A digital copy of this lab and code snippets can be found at:
`http://ibm.biz/nodejs-conversation`

Designing Your Bot

Building a chatbot with Watson Conversation is so easy, some developers choose to dive right into the tooling. However, with a well-thought out, well-planned chatbot, the interaction with the user can lead to a much better experience that can handle edge cases. In this section, we will design the interaction between a user, Dave, and a chatbot named HungerBot.

A good question to ask yourself is, "*Who is my user and what problem do they have?*" Expand on the user's profile by determining what the user needs from this chatbot. Does the user have a need to book a reservation at a restaurant? Or an answer to a common question like "*Where's the bathroom?*" at a conference. Maybe a chatbot that handles tasks like turning on lights or other equipment. It might help to think of the chatbot as an automated version of an existing agent, such as a customer service agent. Look at existing processes that include repeated manual processes, which can sometimes be augmented with chatbots.

Training a chatbot is like training a human agent. You will train the chatbot with the knowledge of certain tasks (intents) and things that these tasks interact with (entities). These components are then combined to create a dialog tree that can take one or more paths to respond to the user's request.

In the following steps, we have provided a sample restaurant chatbot that handles reservations for a restaurant. On the right side, it's your turn to create your own chatbot. Fill in the blanks to design your chatbot.

1. Envision the user that interacts with the bot.

Example:

A user, named Dave, needs to book a table at the restaurant.

Your turn:

A user, named _____, needs to _____.

2. Now, let's give the chatbot a name and describe the overall function it can help with.

Example:

The chatbot, named HungerBot, can help the user with common tasks at a restaurant.

Your turn:

The chatbot, named _____, can help the user with _____.

3. It can be helpful to take a snapshot of an existing dialogue and then break it down into intents and entities. A sample conversation is shown below. Keep the conversation simple…you can always add more complex logic later.

Example:

HungerBot	Hi, I'm HungerBot. You can ask to reserve a table and more.
Dave	I would like to reserve a table
HungerBot	Sure. What type of cuisine do you like?
Dave	I like Chinese food
HungerBot	Me too! When do you want to book Chinese?
Dave	Tonight
HungerBot	And how many people for Chinese?
Dave	Five adults
HungerBot	Excellent! Here are the details of your booking.

Your turn:

Bot _____

User _____

Bot _____

User _____

Bot _____

User _____

Bot _____

User _____

Bot _____

4. Let's start with the action the user wants to do, which is referred to as an *intent*. Write a human-friendly description of the action the user is wanting to perform. List at least five ways the user might phrase this action. Lastly, add a label, like a variable name in code (alpha-numeric, underscores, etc.), that can be used later as a reference.

Example:

Intent: book a reservation

Variations:

1. Reserve a table
2. Book a reservation
3. Make a reservation
4. Secure a reservation
5. Schedule a reservation

Label: #book_reservation

Your turn:

Intent: _____

Variations:

1. _____
2. _____
3. _____
4. _____
5. _____

Label: #_____

If you find that you don't have many variations, invite a friend (or a real user!) to suggest how they would ask "*to book a reservation.*" In the real world, you could use customer interactions as a base of inspiration or use a thesaurus.

5. Another component to training a chatbot is recognizing objects, which is referred to as an *entity*. This example reservation system can differentiate different types of cuisine. We add a type of cuisine to booking a reservation.

Example:

Entity: type of cuisine

Variations:

1. Mexican
2. Chinese
3. American
4. Italian
5. Mediterranean

Label: @cuisine

Your turn:

Entity: _____

Variations:

1. _____
2. _____
3. _____
4. _____
5. _____

Label: @_____

We could add time and number entities, however, there are some built-in system entities provided by IBM, like numbers, dates, and times, that the HungerBot will use. If you have another entity, use a separate sheet of paper to define the additional entity.

Finally, we design the flow of the conversation. Using the labels of the intents and entities, fill in each blank with one stage of the conversation that must be present before the chatbot can continue to the next blank to the right.

Example:

`conversation_start` -> #book_reservation -> @cuisine -> @sys-date -> @sys-number -> *fin*

Your turn:

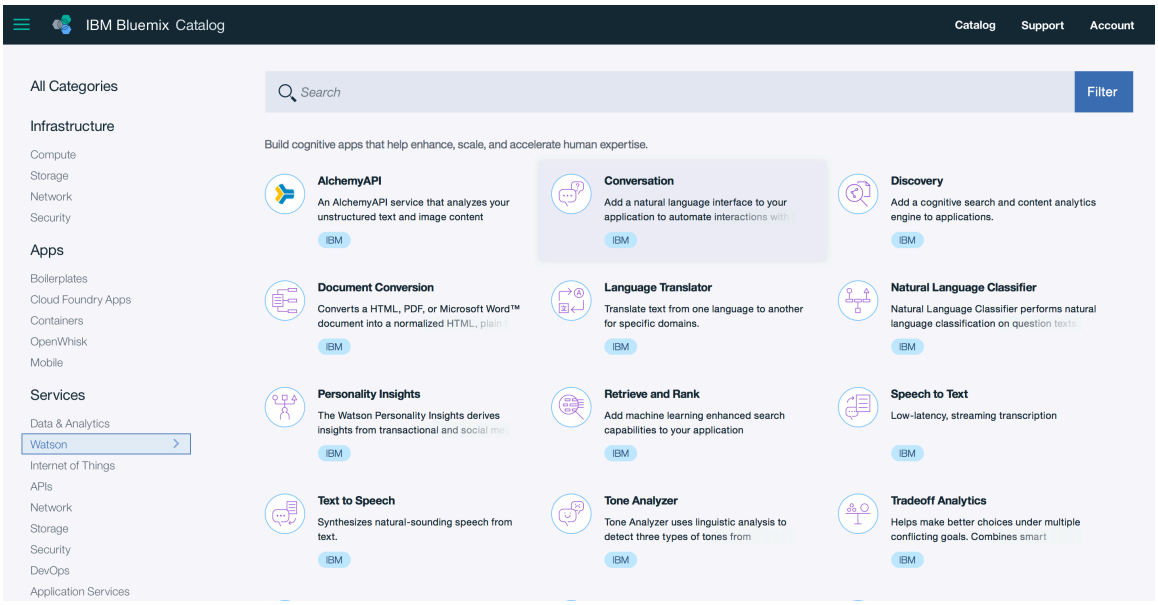
`conversation_start` -> _____ -> _____ -> _____ -> _____ -> *fin*

Notice how we have intents and entities used throughout the conversation. In the Dialog editor of Watson Conversation, we can now setup logic to step the user through the conversation. In the next section, we will use this design to train Watson Conversation.

Training Watson Conversation Service

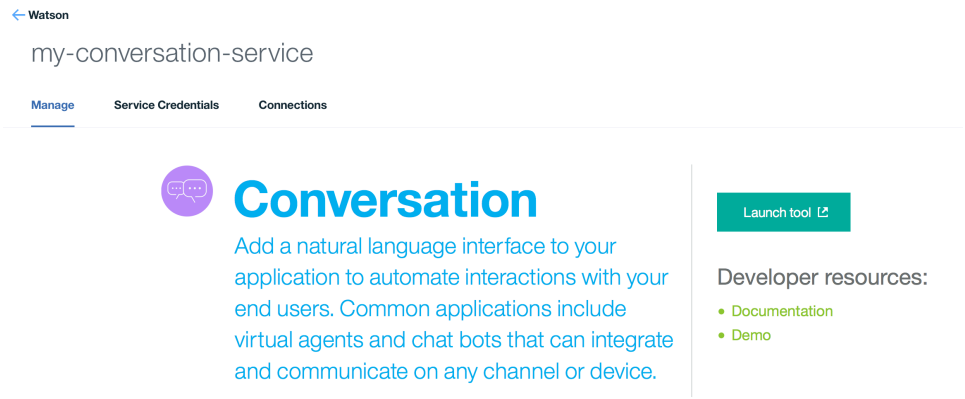
Now that we have designed the first dialogue between the chatbot and the user, we can train the Watson Conversation service. Sign up for an IBM Bluemix account at `bluemix.net`. If you already have an account, sign into your account.

1. Click on the **Catalog** link in the top-right corner of the Bluemix dashboard.
2. Select **Watson Conversation** tile under the section titled Watson.

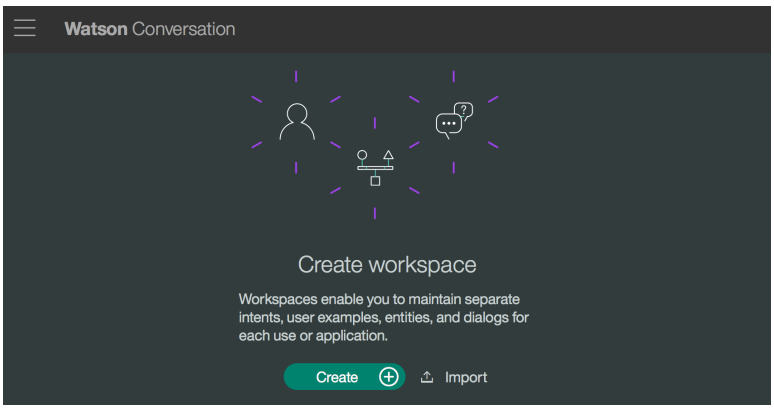


3. Enter `my-conversation-service` in the field labeled **Service name**. Click on **Create**.

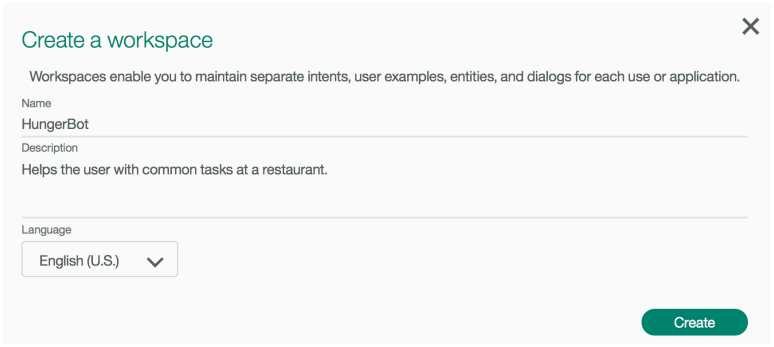
4. Click on the green **Launch tool** button.



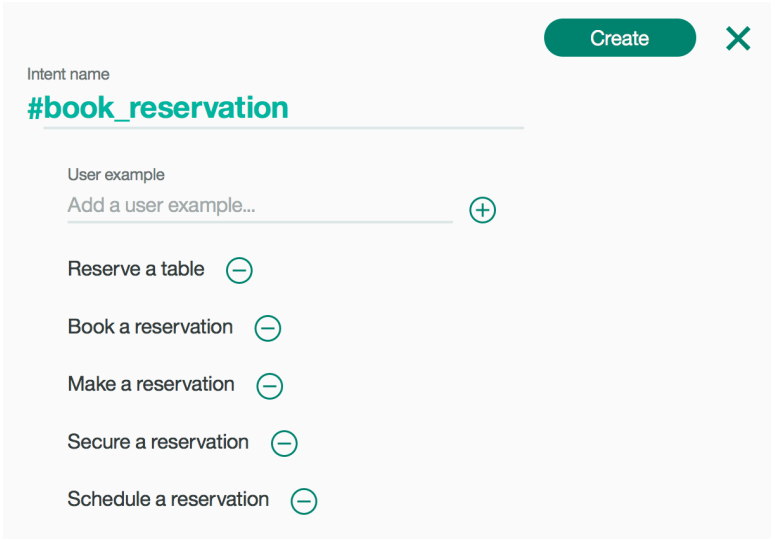
5. This is the Watson Conversation tooling where you can create workspaces and setup different chatbots dialogues and applications. Click on the green **Create** button to create a new workspace.



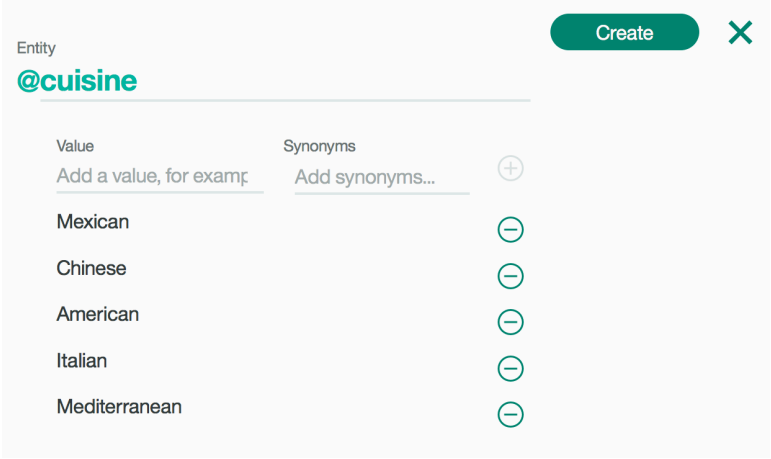
6. Enter a name for the chatbot and a description. Use the answers you wrote on page 2. Click **Create** when finished.



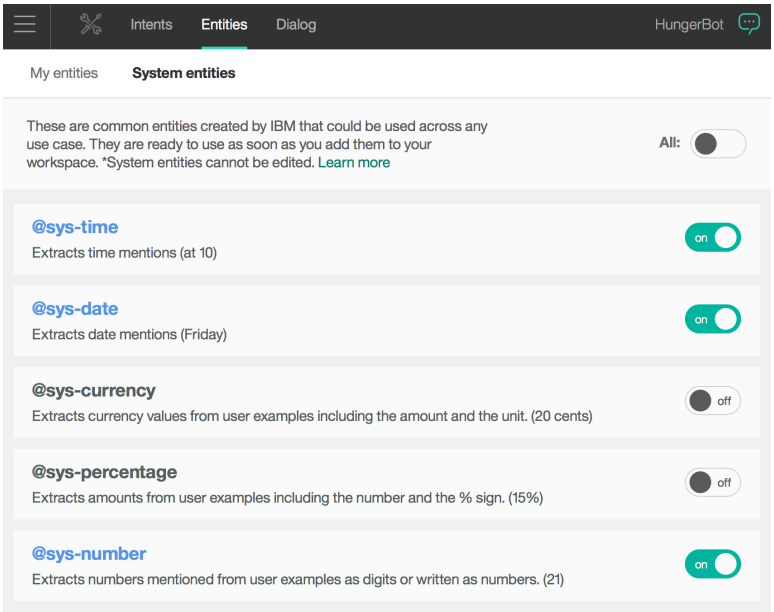
7. You will be redirected into a page with three tabs, **Intents**, **Entities**, and **Dialog**. Under the **Intents** tab, click on **Create** to create the first intent.
8. Use the answers you wrote on page 3 to create the first intent. Click on **Create** when finished.



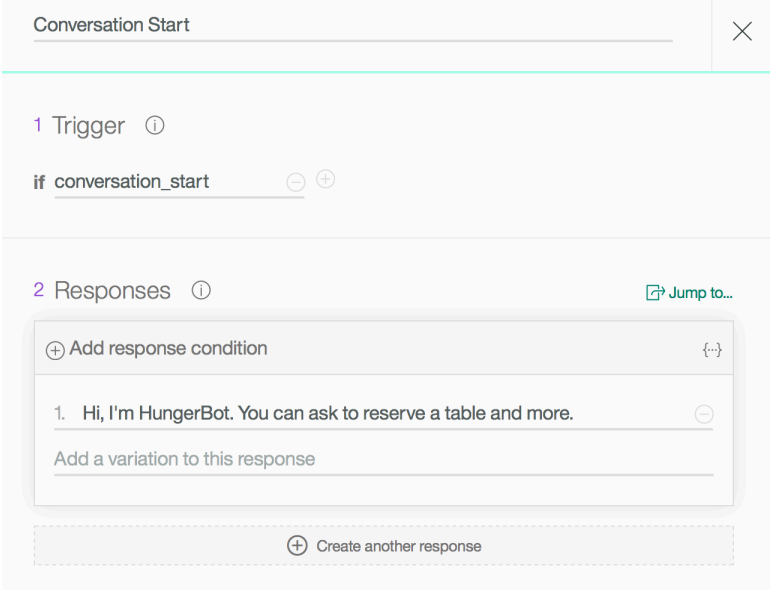
9. Click on the **Entities** tab in the top menu bar. This is where you can add entities. Add the entity you have on page 3. Click **Create** when finished.



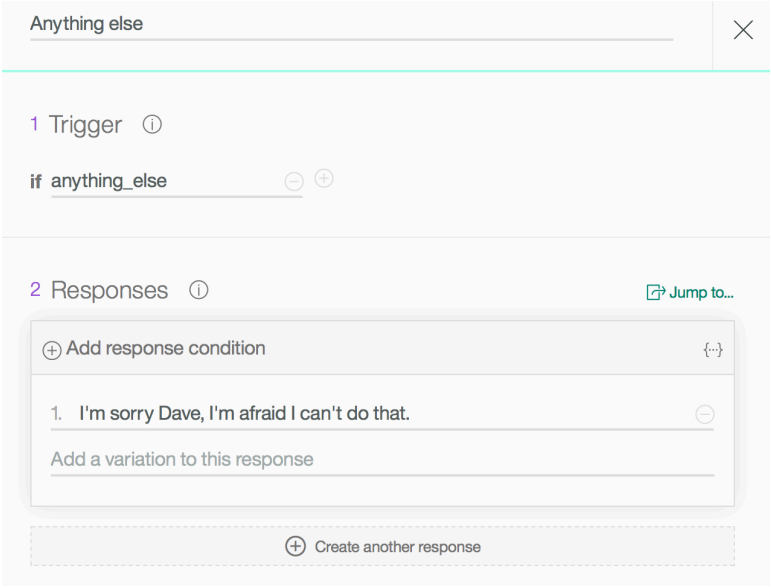
10. The Watson Conversation has a handful of common entities created by IBM that can be used across any use case. These entities include: date, time, currency, percentage, and numbers. Click on **System entities**. Toggle on the switch for `@sys-time`, `@sys-date`, and `@sys-number` to enable the entities.



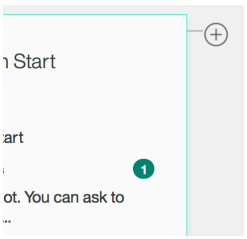
11. Click on the **Dialog** tab in the top menu bar. Click **Create**. There are two nodes you may choose to add by default. The `conversation_start` condition is triggered when the chatbot is initially started. This is a good place to introduce the bot and suggest actions the user can ask of this chatbot.



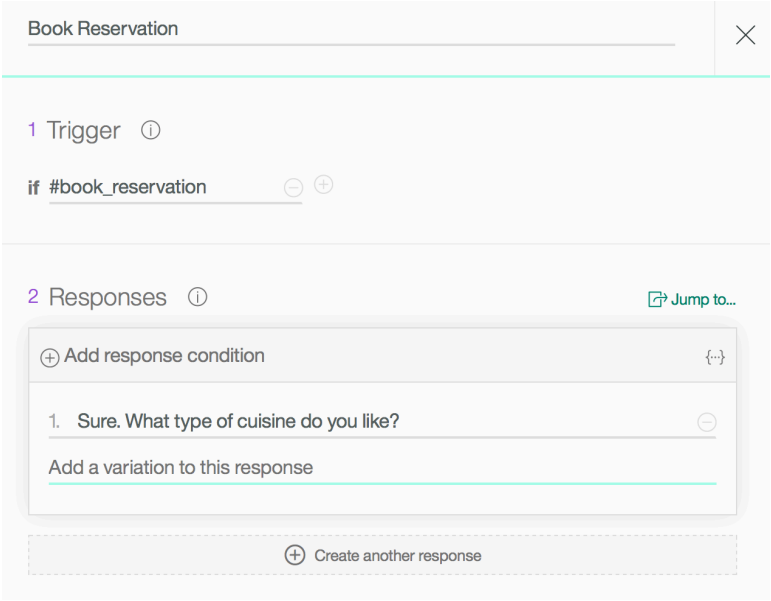
12. Click on the + sign below the previous node. Add a node that matches the condition `anything_else`. In the event the user enters something that wasn't expected, the service will return this response. Ideally, it should convey a way for the user to recover, such as example phrases.



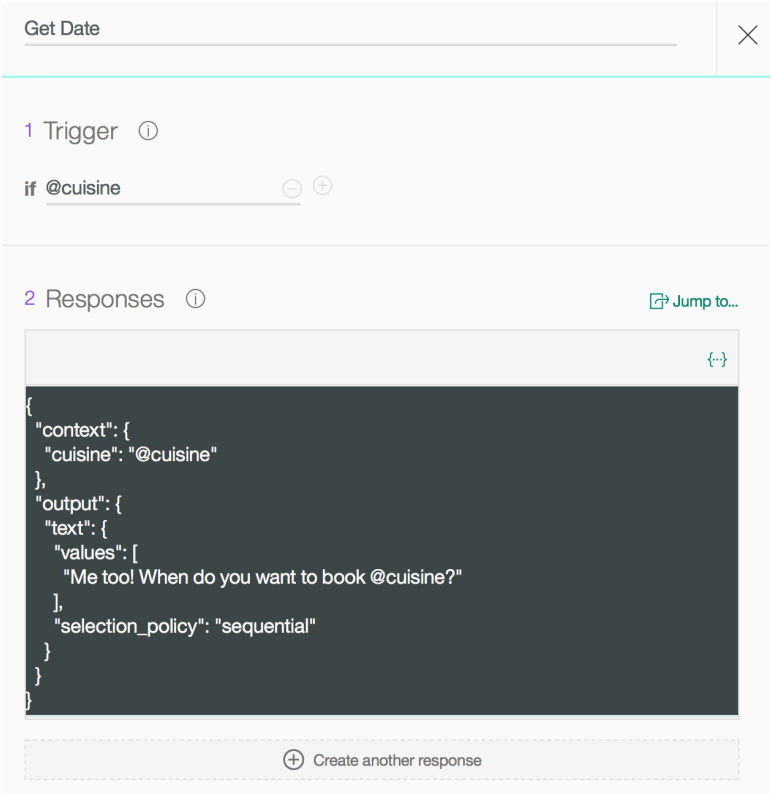
13. Return back to the `conversation_start` node, and click on the + sign to the right on the node. If it doesn't appear, select the `conversation_start` node.



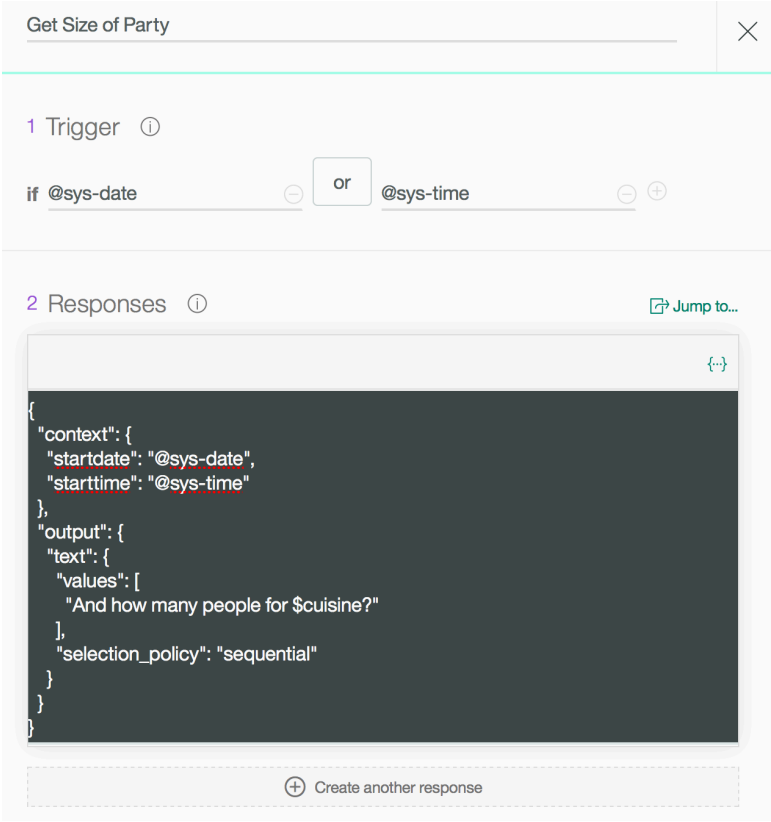
14. Add a node to test the condition of the first intent, `#book_reservation`, as shown below.



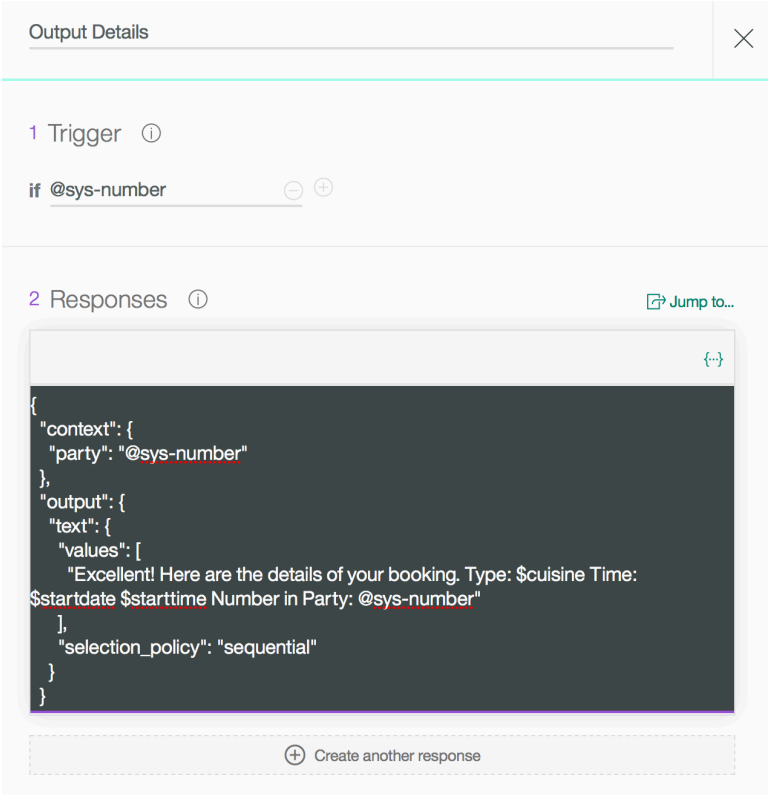
15. Add a node to the right, by clicking on the + sign to the right of this box, to test for the condition `@cuisine`. This time, click on the {...} symbol in the responses section to expand the advanced editor. This box lets you edit the JSON with the context, output, and other elements of the response object. The `context` property is used to keep track of the user's choices so we can use them later.



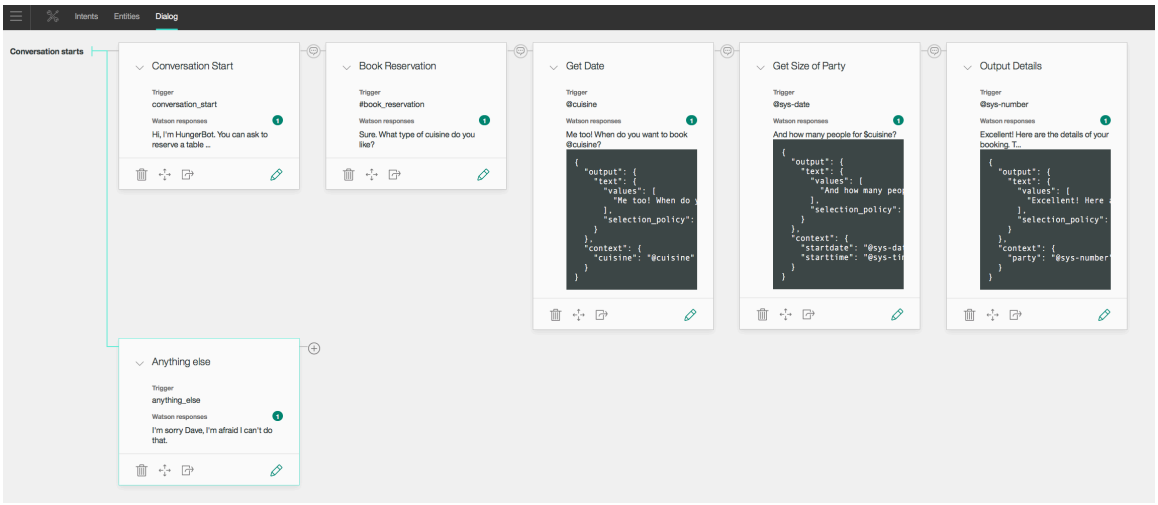
16. Add another node to the right to prompt for the number of people in the party as shown below.



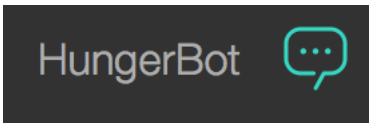
17. Finally, add a node to the right to display the choices as shown below.



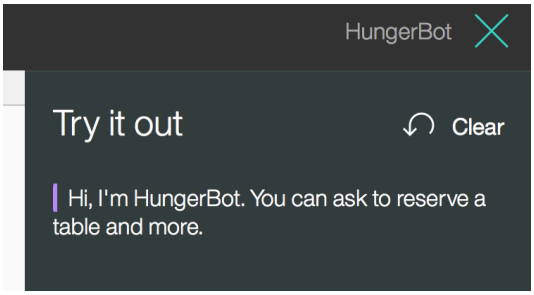
The completed dialog is shown below.



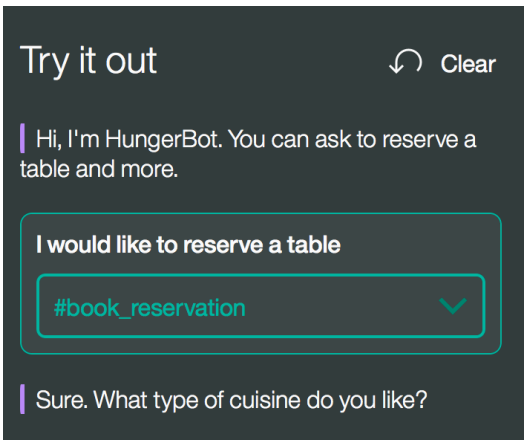
18. To test the bot, click on the Ask Watson icon in the top-right corner of the tooling.



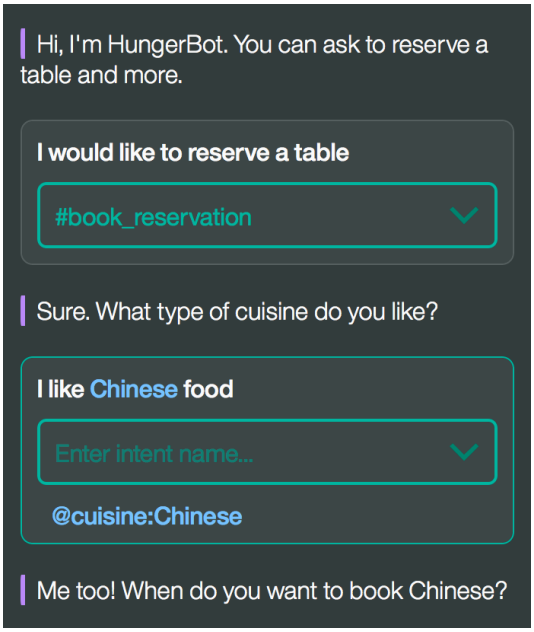
19. A sidebar appears and shows the contents of the node that matches `conversation_start`. Enter a message that triggers the first intent. In the example bot, we can ask "`I want to book a table.`"



20. Notice that the intent `#book_reservation` was recognized. The `#book_reservation` node was triggered and the output includes the response from the `Book Reservation` node.

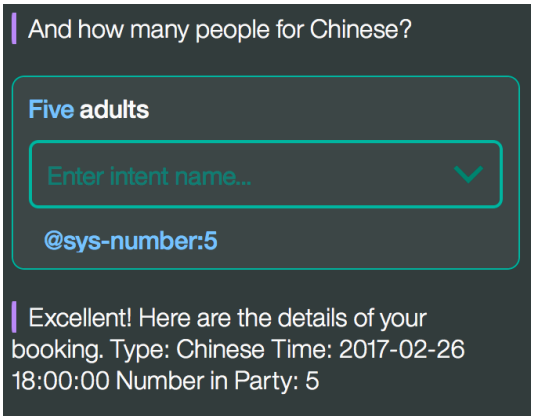


21. When the user enters a cuisine, the `@cuisine` entity is recognized. This allows a more refined dialog tree that could have other branches based on a specific value of the entity.



22. When the user enters a date or time, Watson extracts out the value using the system entities `@sys-date` and `@sys-time`.

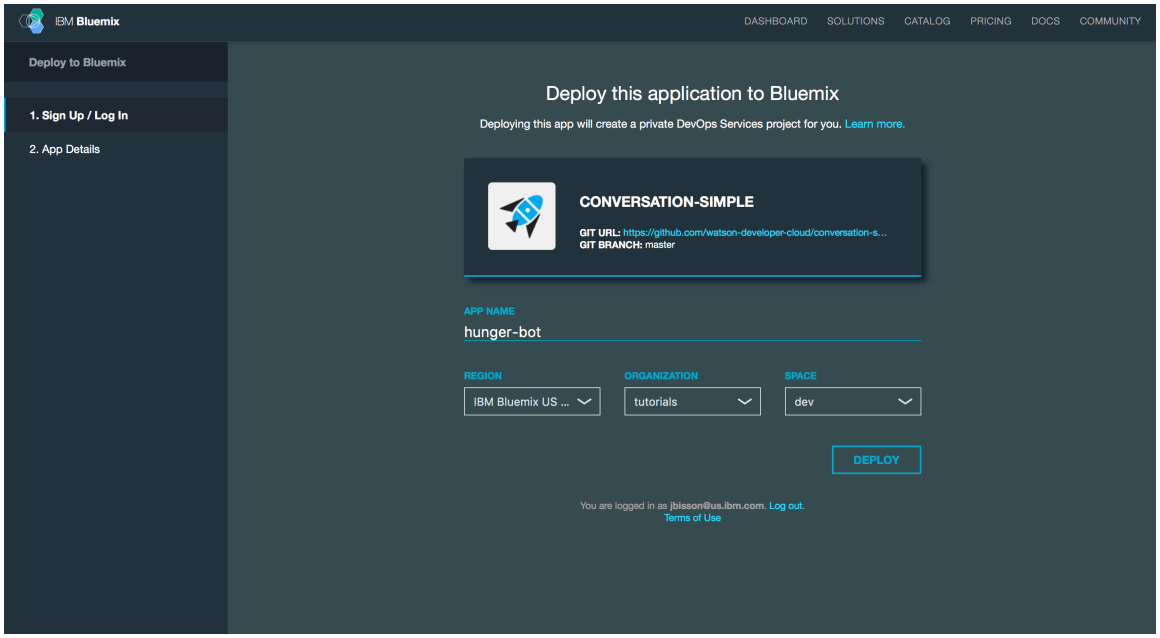
23. Finally, when the user enters a number (even spelled out) for the number of people in the reservation, Watson extracts out the number using the system entity `@sys-number`.



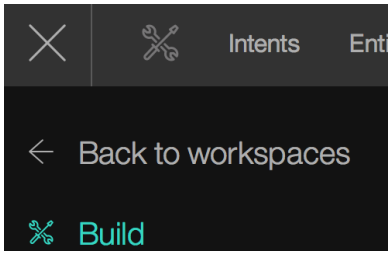
Node.js Sample Application

Now that we have a workspace created, we can use a sample Node.js application from the Watson Developer Cloud GitHub repository to create a web-based interface (`ibm.biz/conversation-simple`).

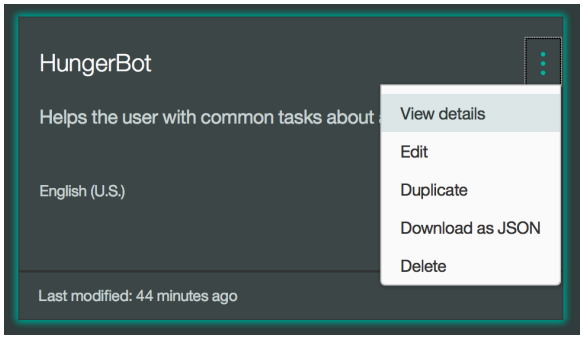
1. Open a new browser window and deploy the sample application via IBM Bluemix DevOps services by visiting `ibm.biz/deploy-conversation-simple`
2. Enter an application name. Click on **Deploy**.



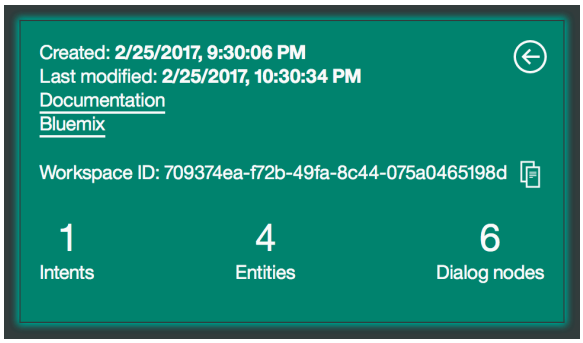
3. Wait for the four stages to complete. Return to the Watson Conversation tooling and click on the menu icon in the top-left corner. Click on **Back to workspaces**.



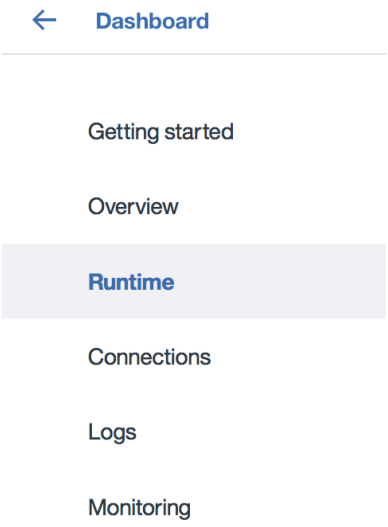
4. Click on the three dots in the card for the workspace. Select **View details** from the menu.



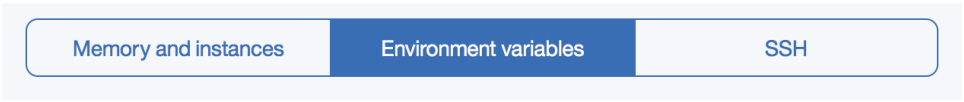
5. Copy the **Workspace ID**.



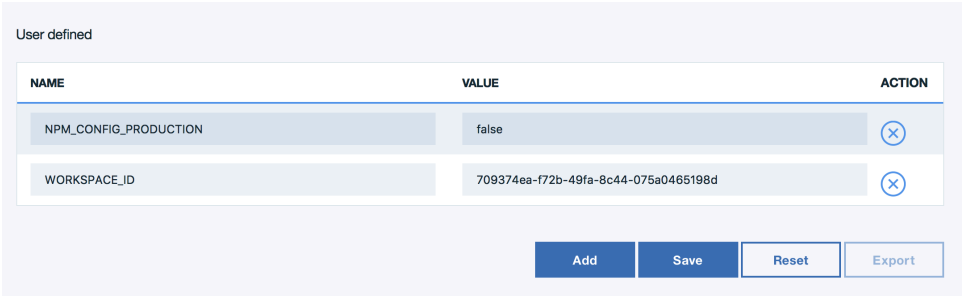
6. Open another tab and visit your IBM Bluemix console at `bluemix.net`.
7. Click on the application to view the overview of the application.
8. In the menu on the left side, click on **Runtime**.



9. Click on **Environment variables**.

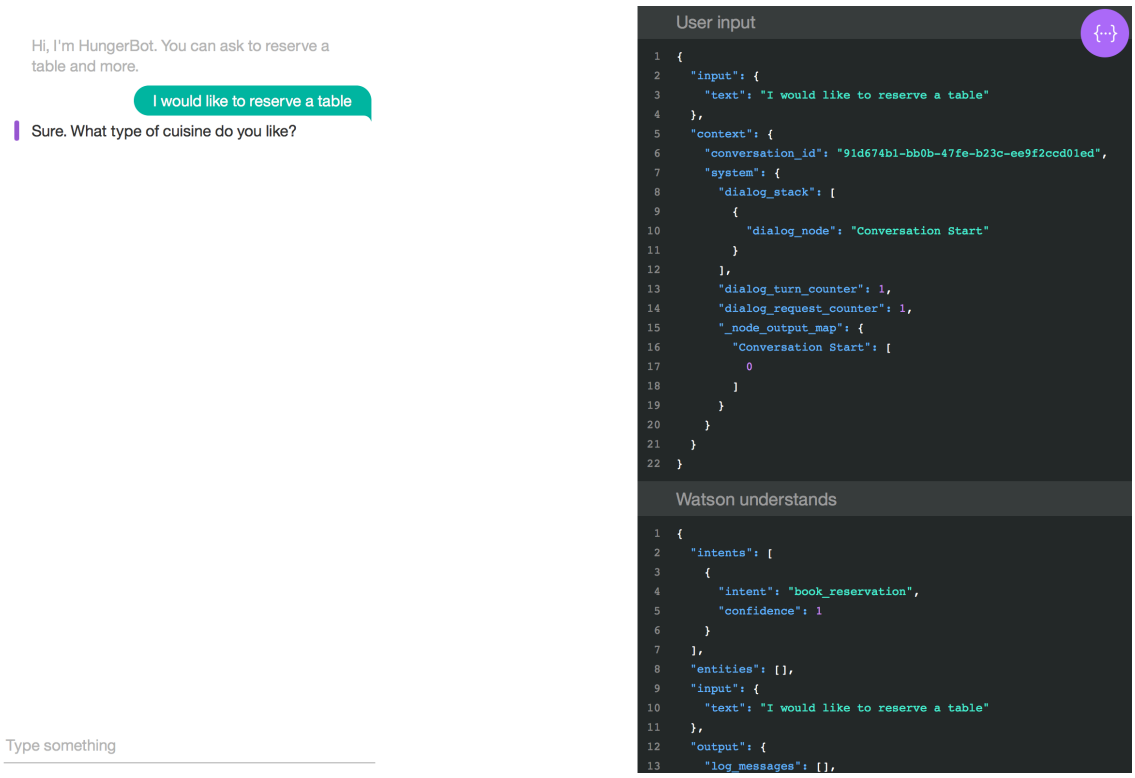


10. At the bottom of the page, click **Add** and add the environment variable named `WORKSPACE_ID` with the value from step #5.



11. Click **Save. The application will** restart and include this new variable in the environment. The application uses this environment variable when processing the messages in the chat window and calling the Watson Conversation API.

12. Visit the application's URL by taking the hostname you chose in Step #2 (`hunger-bot`), appended with `.mybluemix.net` . This example's chatbot URL would be http://hunger-bot.mybluemix.net
13. This application has a chat window on the left, and the user input and JSON response that updates after every user input.



14. Explore how the intents, entities, input and output JSON changes as the user proceeds through the conversation. You can also view the context values, which can be used in the application source code.

Invite a friend to test out the chatbot and find new phrases/use cases that you haven't thought of. It is common to update and improve the training as new phrases are discovered. This was a very basic introduction to the Watson Conversation service. For more information on advanced topics, please visit `ibm.biz/conversation-docs` .