IBM Bluemix

Using Watson Conversation Service to build chatbot
Lab Exercise

Build Help Desk Assistant chatbot using Watson Conversation Service

Prerequisites

To complete the steps in this lab, be sure you have these prerequisites:
_ Access to a Bluemix account
_ Basic knowledge of Bluemix
_ Basic knowledge of the IBM Watson Conversation service.

Installables

1) Cloud Foundary Plugin – https://github.com/cloudfoundry/cli/releases
2) Git - https://git-scm.com/download

Step-by-step implementation

Implementing this use case involves the following steps:
1. Creating a new Conversation workspace
2. Adding intents
3. Adding entities
4. Creating the dialog
5. Testing the dialog
6. Creating the Help Desk Assistant chatbot application in Bluemix(Node js application) and setting up the Application with Conversation Service

Step1:

Creating a new Conversation workspace

Complete the following steps:
1. Log in to Bluemix and open the Dashboard.
http://bluemix.net
(If you have not signed up please sign up and login using credentials)

2. Creating a Watson Conversation service instance

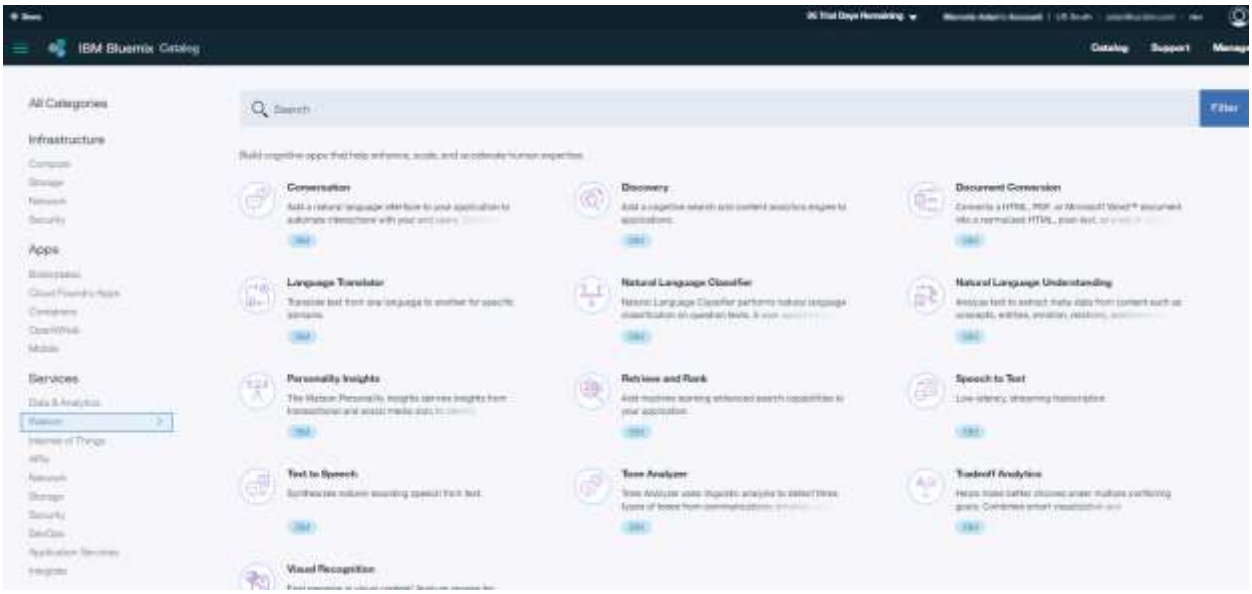
Bluemix provides resources to your applications through a service instance. Before you can use the Watson APIs you must create an instance of the corresponding service. You will need to create a Watson Conversation service instance for use.

To create an instance of the Conversation service, follow these steps:

1. Log in to IBM Bluemix.

2. Click **Watson** (under Services).
The Watson services that are available in Bluemix are listed.

3. Click **Conversation**.



Access the Conversation service instance

Do these steps on the next web page:
a. Enter Conversation as the service instance name.
b. Notice the credential name, Credentials-1.
c. Select the pricing plan you want to use.
d. Click **Create** and wait for Bluemix to create an instance of your Conversation service.

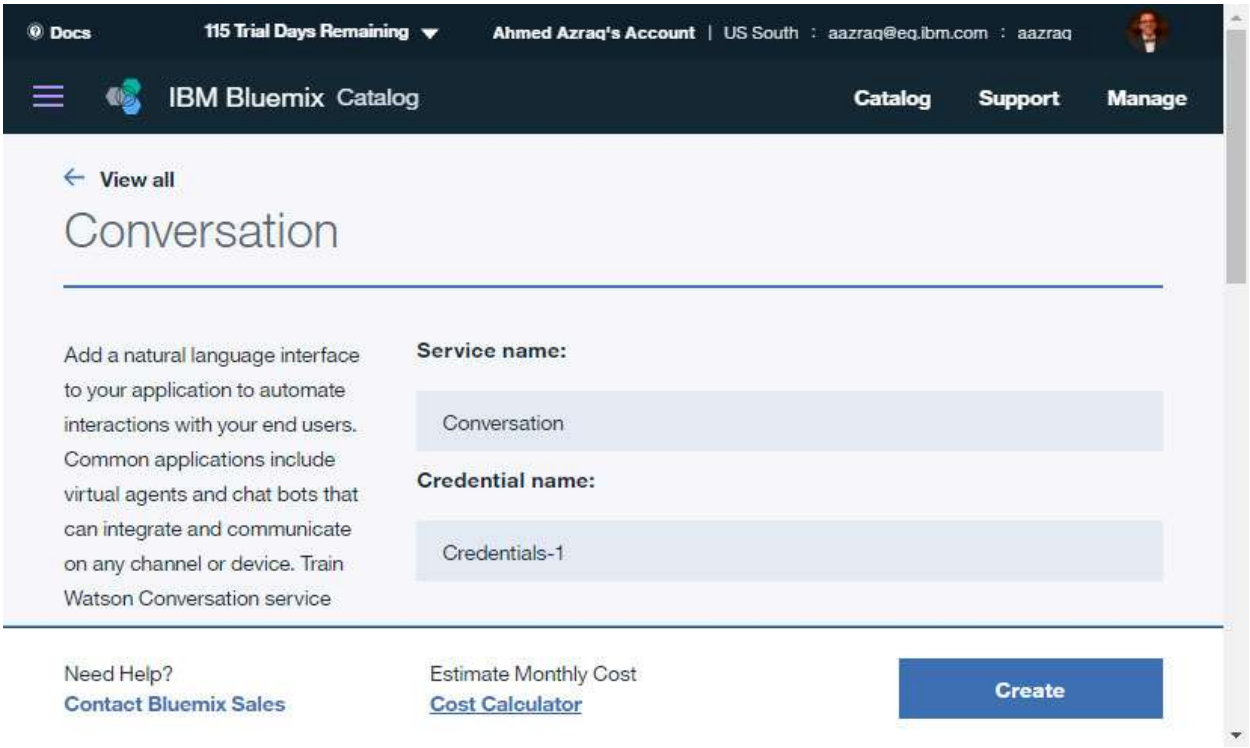


Figure: Conversation service instance name

Launching the Conversation tool

The Conversation tool is a visual dialog builder to help you create natural conversations between your apps and users, without any coding experience required. Complete these steps to launch the tooling:

1. After creating the Conversation service instance, click **Launch tool**

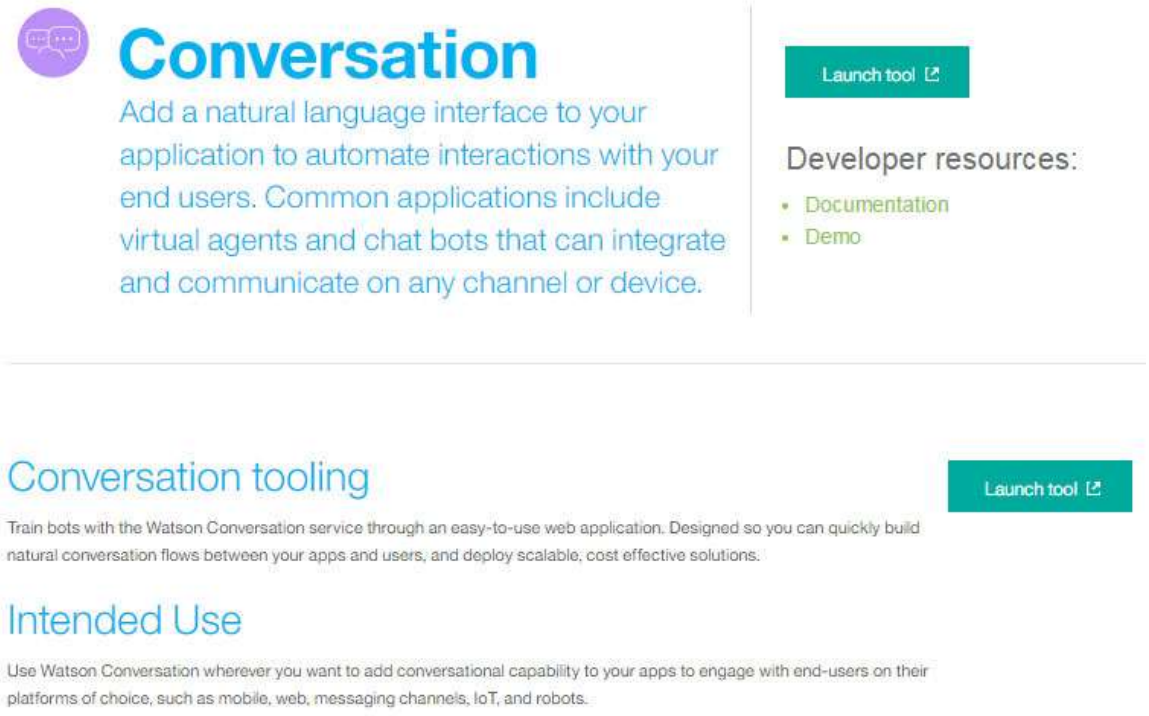
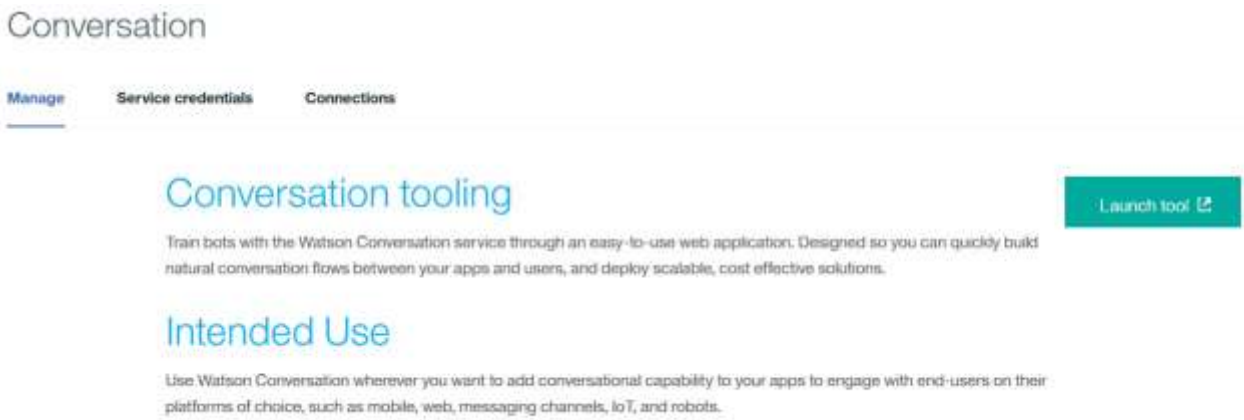


Figure Launching the conversation tool immediately after creating the service instance

2. Alternatively, you can launch the tool at a later time:
a. Go to the Bluemix dashboard.
b. Click your Conversation service instance.
c. On the service details page, click the **Manage** tab (Figure), scroll to Conversation tooling, and click **Launch tool**.



Launch Conversation service tool

If this is the first workspace, the Watson Conversation login page opens (Figure). If you have an IBMid, click **Log in with IBM ID**; otherwise, click **Sign up for IBM ID**

Figure Log in Watson Conversation tooling

Working with a workspace

This section describes how to create, delete, import, and rename a workspace.

Previously created workspaces are listed. However, for this app you need a new workspace, so click **Create**.

Create a new workspace

Complete the following steps:
1. Launch Conversation tooling.
2. Click **Create** to create a workspace

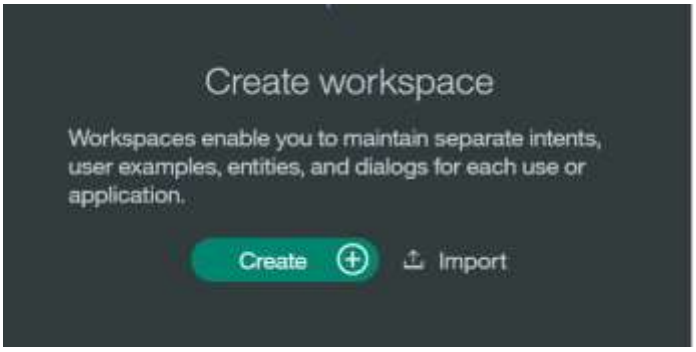
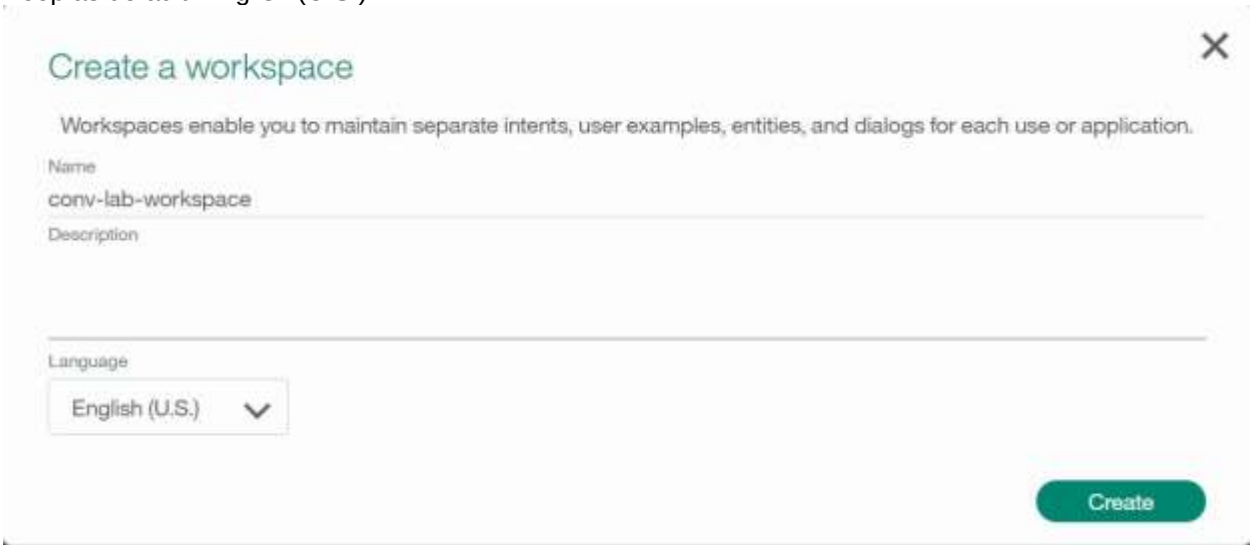


Figure Create new workspace

3. As shown in Figure , specify the details of the new workspace:

– Name: conv-lab-workspace (or any name which you wish)
– Description: Any description not more than 128 characters.
– Language: Language of user input that the workspace will be trained to understand;
Keep as default: English (U.S.).



4. Click **Create**.

The new Conversation workspace is created

Now you will start building workspace.

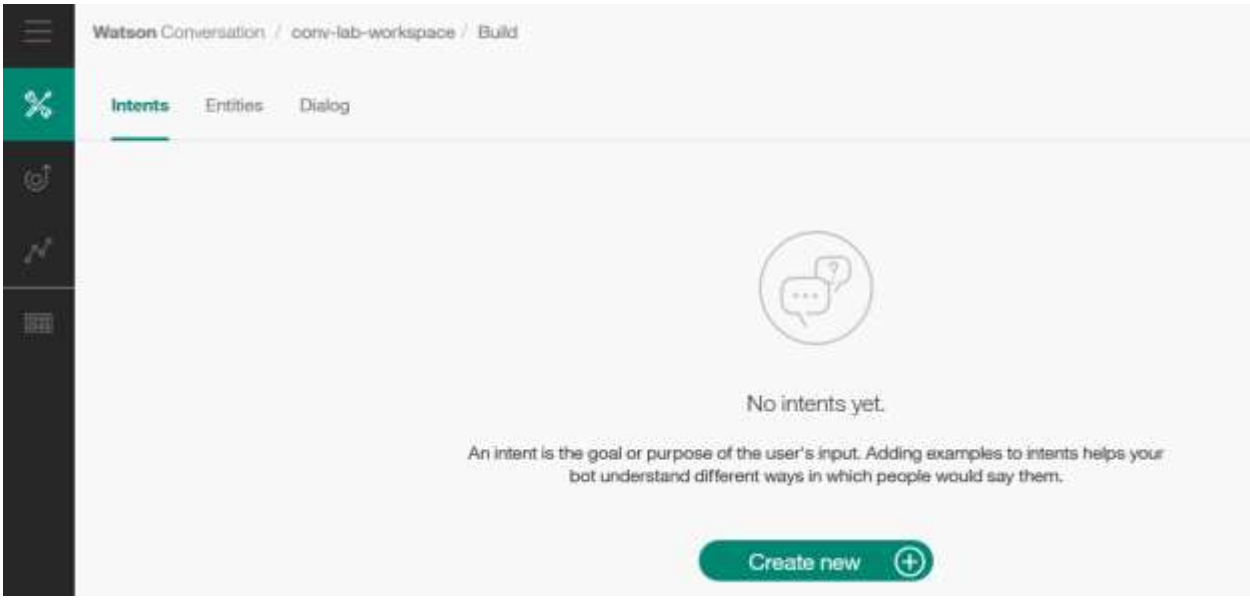


Figure Watson Conversation workspace - Build

Step2:

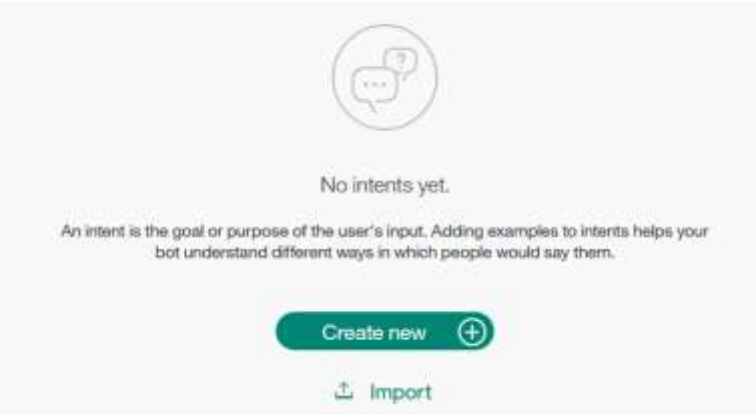
Adding intents

In this section, you add intents to the Chatbot workspace. The intents should be appropriate for the Help Desk Assistant chatbot.

In this section, you add the following intents to the workspace.
a) Software-Issues
b) Hardware-Issues
c) Hello
d) Affirmative

a)
Create a Software-Issues intent

Use the Conversation tool to create a new intent:
1. Click the **Chatbot workspace**. The Intents tab opens automatically.
2. Click **Create new** (Figure).



Name the intent # Software-Issues

In the User example section, add these examples to the
#Software-issues intent; click the plus sign (**+**) or press Enter to add each user example:
– Application issue
– Issues with Office
– Problem with automatic updates
– My email not working
– Application not running

Add as many software issues examples as you can, so that the application can be more accurate
(Five examples is the minimum).

When you finish adding user examples, click **Create** to save the intent.
After you create the intent, the system starts to train itself with the new data.

Note: The hashtag symbol (#) is added by default to the name; do not add it yourself.

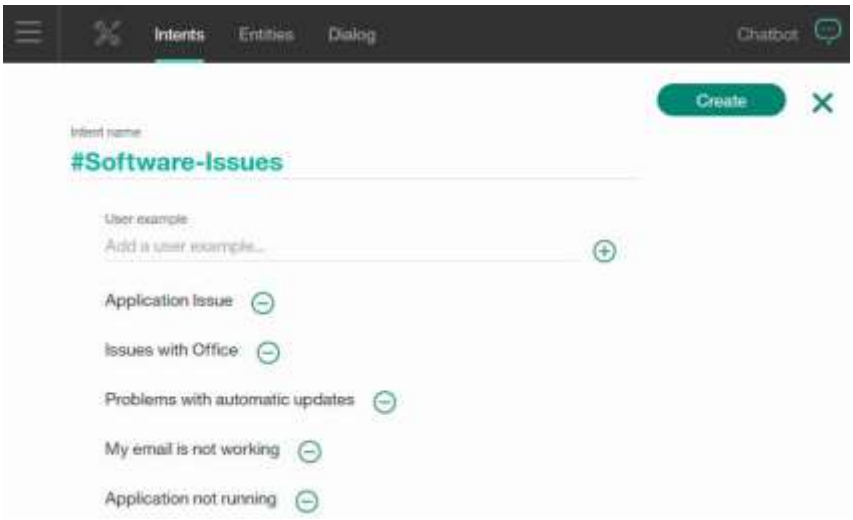


Figure Add #Software-Issues intent (part 1 of 4)

b), c), d) Add the other intents that are shown in Figures below

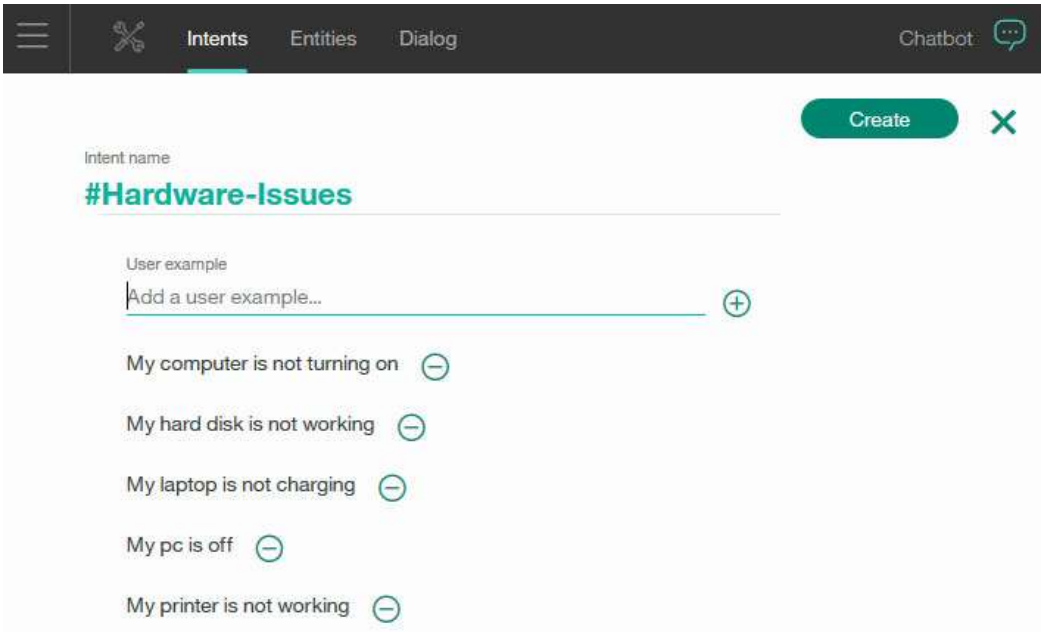


Figure Add #Hardware-Issues intent (part 2 of 4)

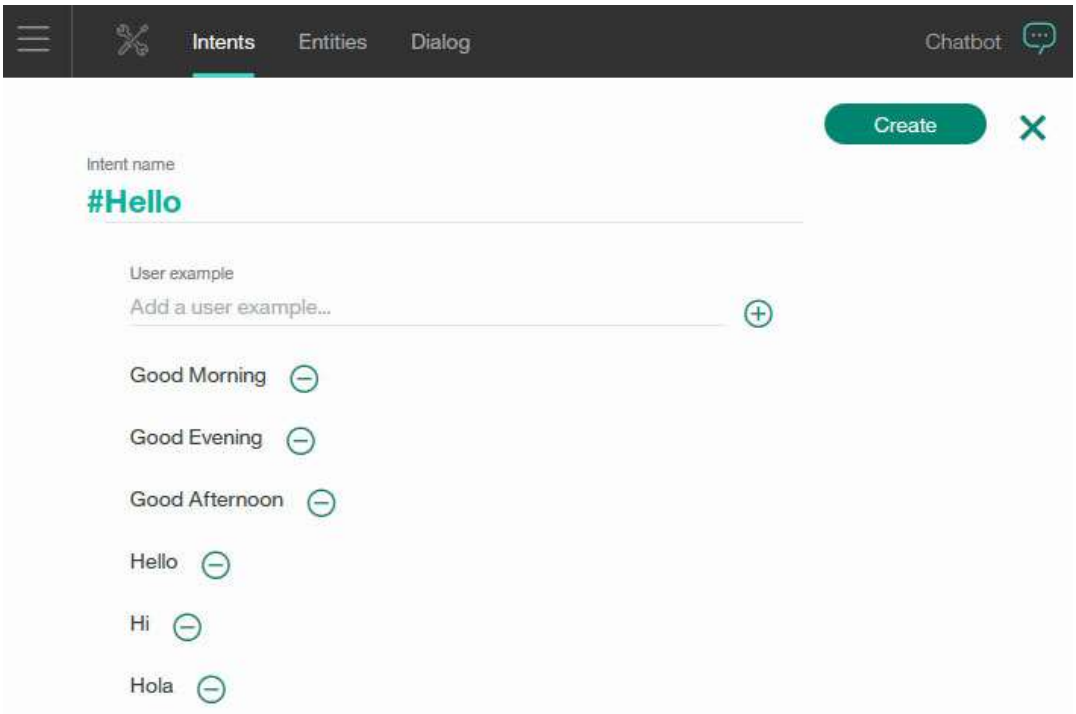


Figure Add #Hello intent (part 3 of 4)

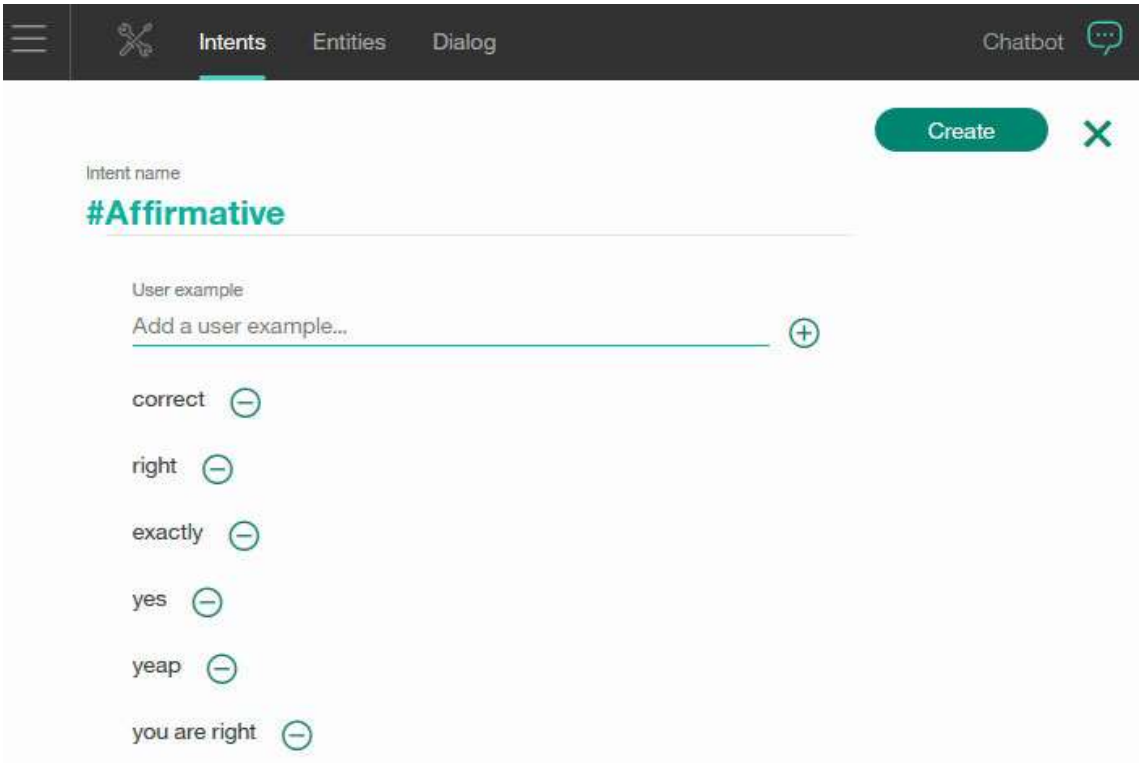


Figure Add #Affirmative intents (part 4 of 4)

These intents are enough for this example; however, you can create as many as you want. Some examples include OutOfScope (for incomprehensible user input), Bye (to close the conversation), and others.

Figure shows the final list of intents in the Chat-bot workspace.

Final intents list in workspace

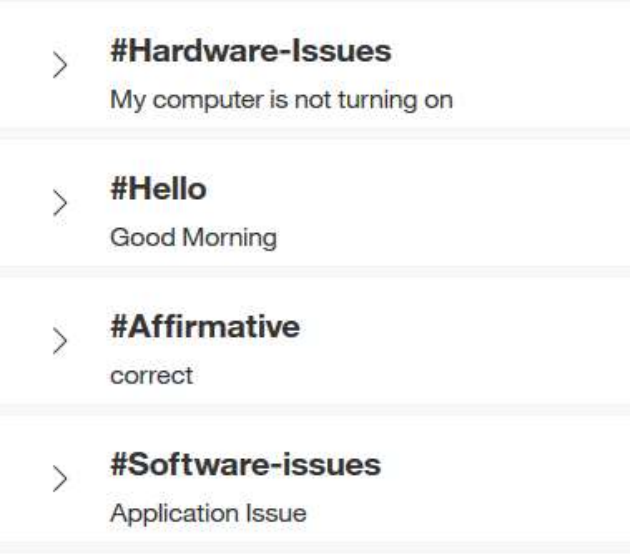


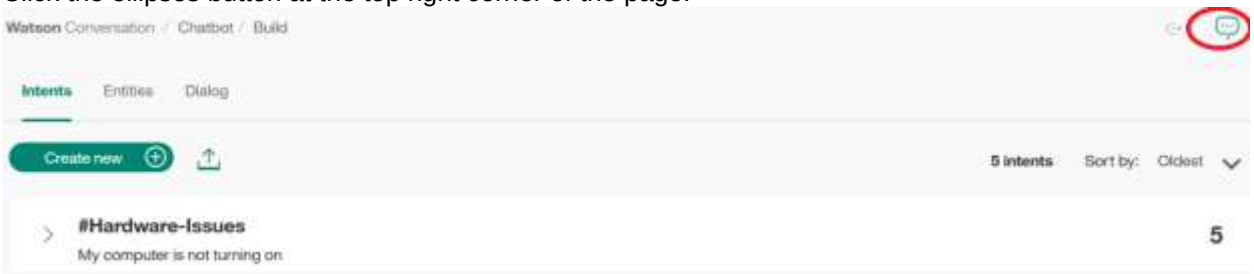
Figure Chatbot Intents

Test your intent

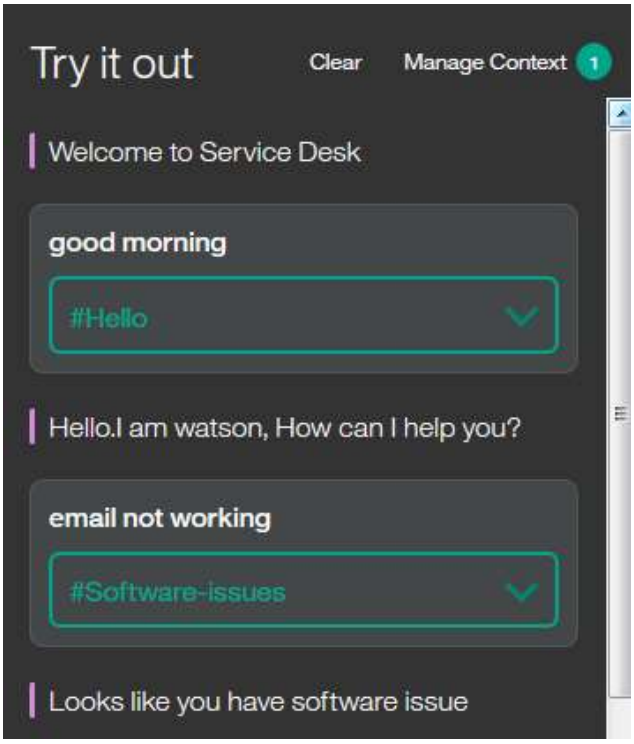
After defining the new intents and examples. You can test your system to be sure it accurately recognizes the intents. If not, then the intents must be refined.

Complete these steps to test your system:

1. Click the ellipses button at the top right corner of the page.



2. Enter a question or a phrase to test whether the system recognizes the correct intent


Try it out
Clear
Manage Context
1
Welcome to Service Desk
good morning
#Hello
Hello.I am watson, How can I help you?
email not working
#Software-issues
Looks like you have software issue

Step3:

Adding entities

In this section, you add entities to the Chatbot workspace. The entities should be appropriate for the Help Desk Assistant chatbot.

An entity represents a class of object or a data type that is relevant to a user's purpose. By recognizing the entities that are mentioned in the user's input, the Conversation service can choose the specific actions to take to fulfill an intent.

Select **Entities** and create the four entities that are shown below figures

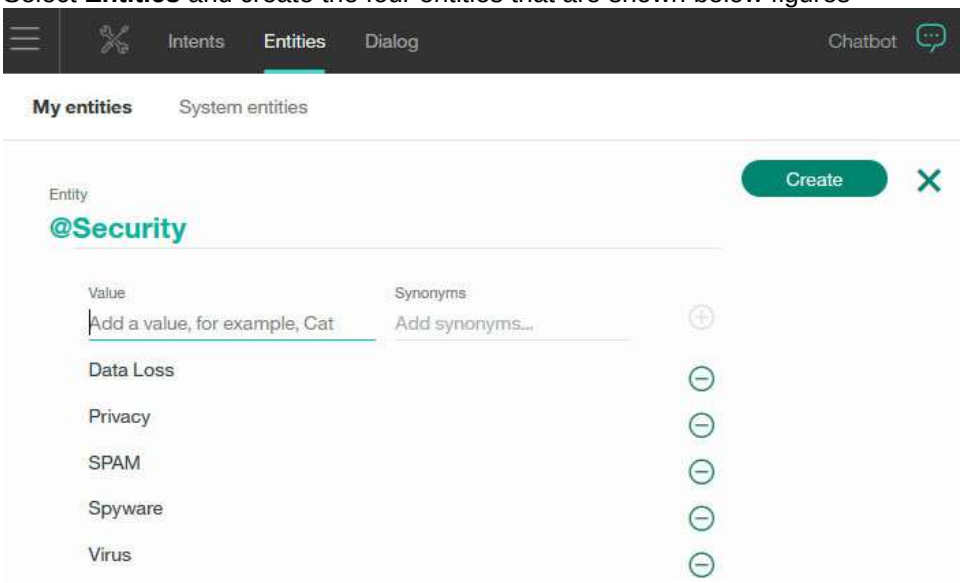


Figure Add @Security entity (part 1 of 4)

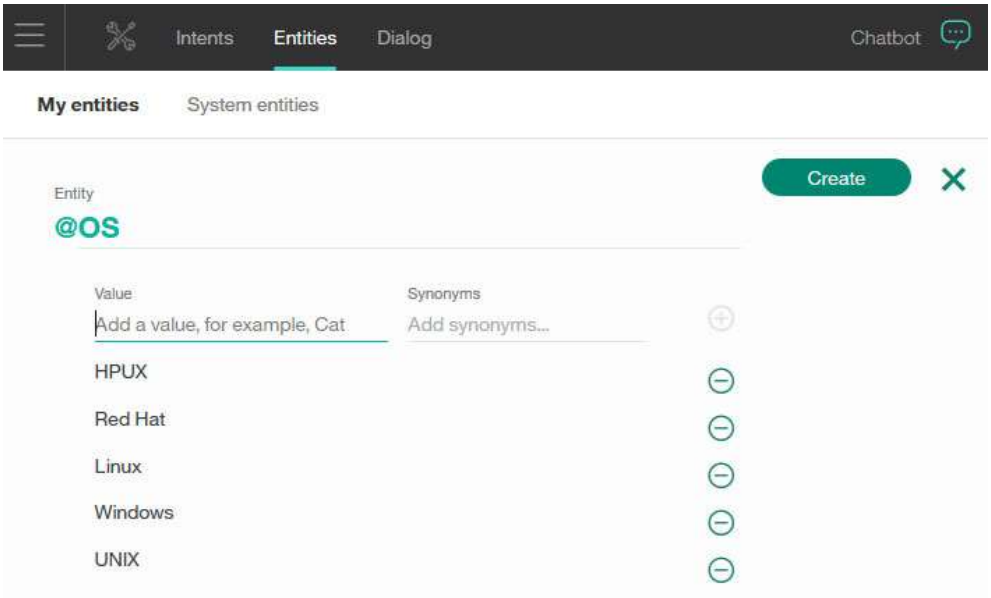


Figure Add @OS entity (part 2 of 4)

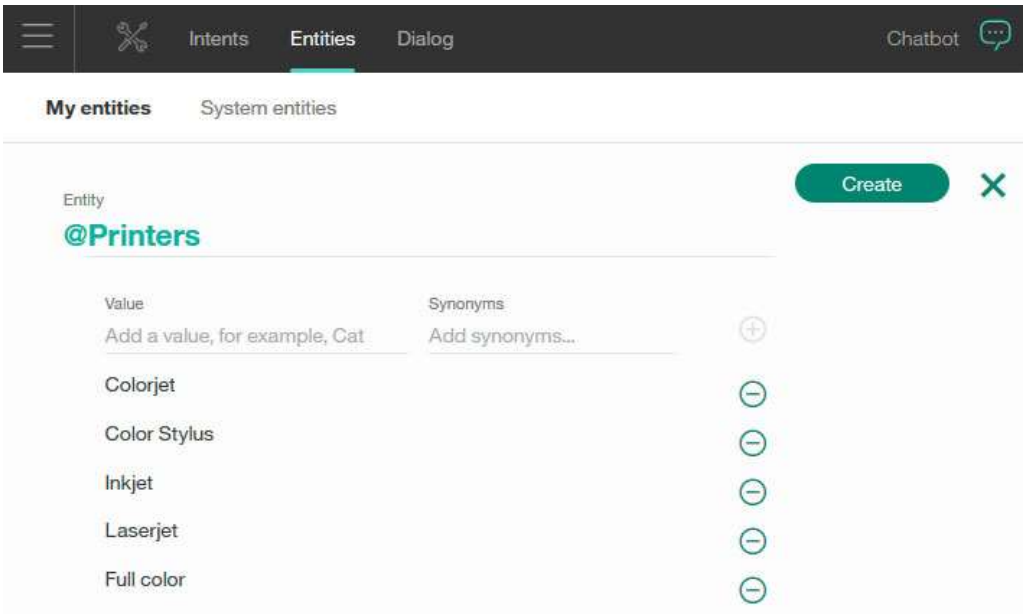


Figure Add @Printers entity (part 3 of 4)

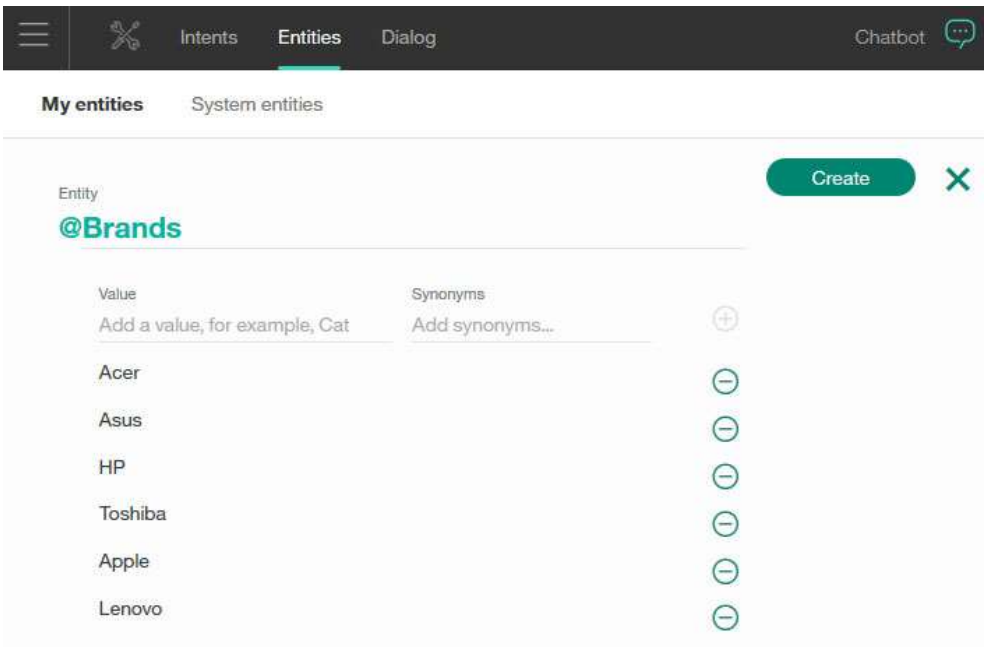


Figure Add @Brands entity (part 4 of 4)

Following is the list of entities created.

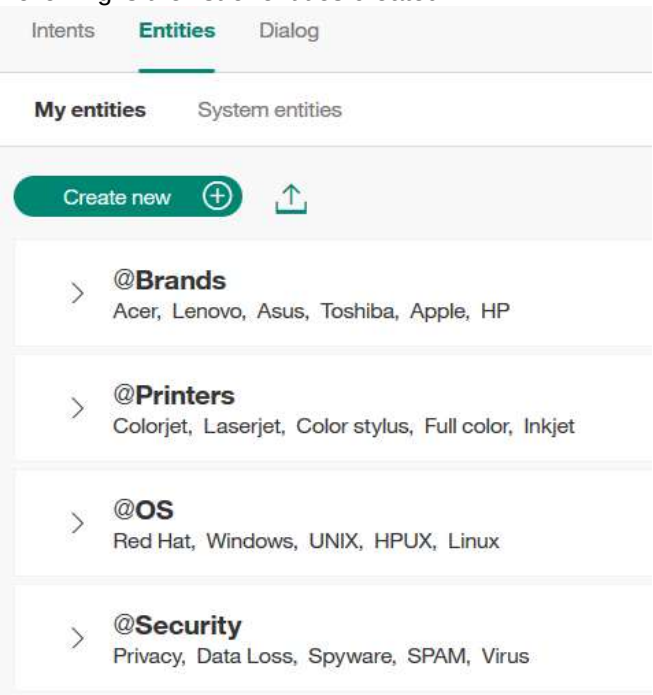


Figure : Entities Added

Step 4:

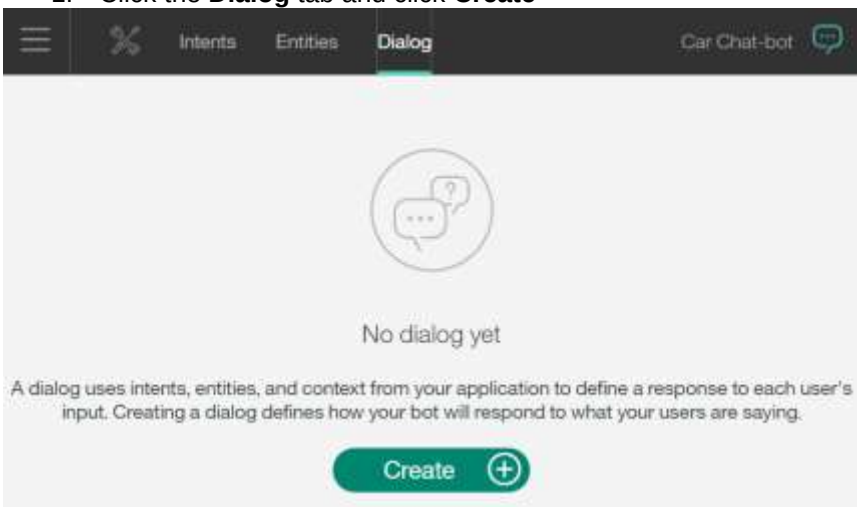
Creating the dialog

In this section, you build the Conversation dialog for the chatbot by using the created or imported intents and entities.

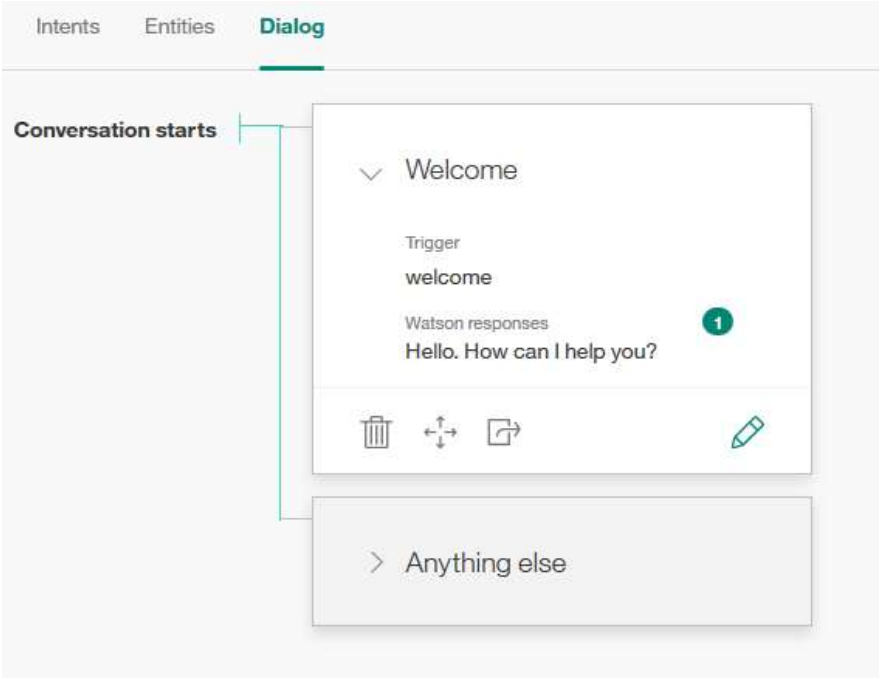
Start the dialog

Complete the following steps:

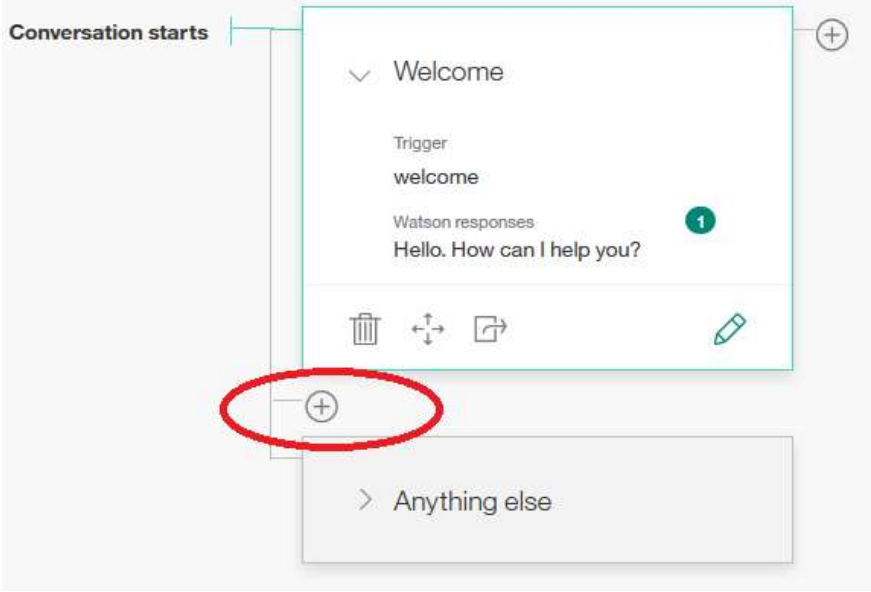
1. Click the **Dialog** tab and click **Create**



Once you create a dialog, by default you will see Welcome and Anything else dialog



Once you select Welcome, you will be shown below option  click on that



An untitled node is displayed in the dialog, when it is first created

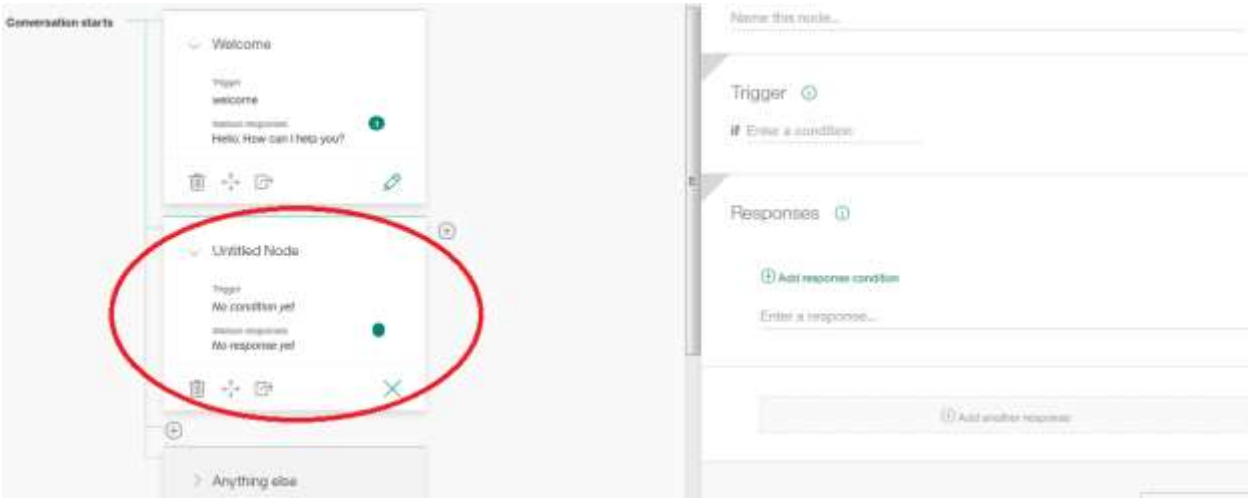


Figure Create the First node

On the right section, you can fill in the details like trigger condition, and Response as shown below figures

Create the dialog branch shown in Figure with the following nodes:
– Hardware Issues (parent)
– Affirmative HW (child of Hardware Issues)
– HW Brands (child of Affirmative HW)

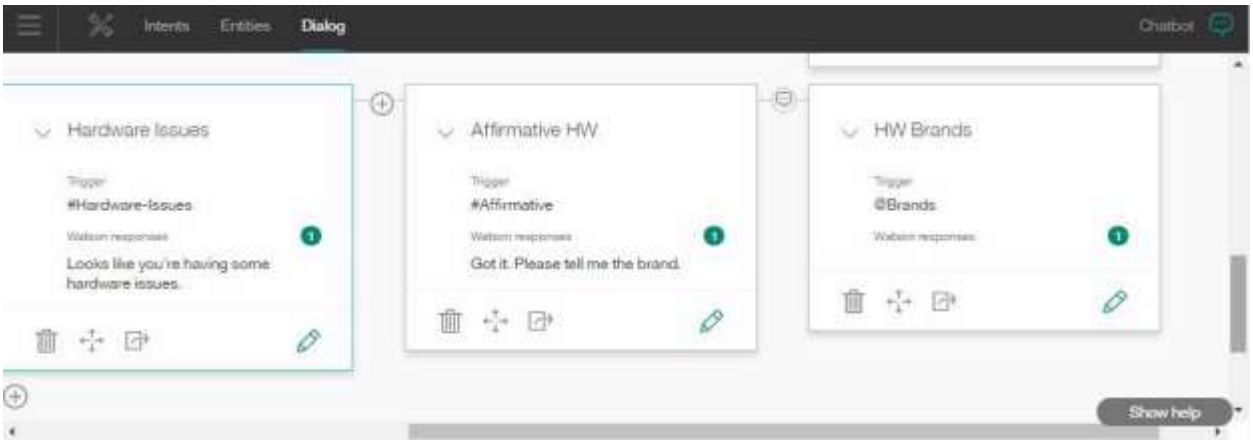


Figure Adding Hardware Issues, Affirmative HW, and HW Brands nodes

In the HW Brands node, create a response for each example in the @Brands entity (Acer, Asus, HP, Toshiba, Apple, Lenovo, and so on).

a. Click the **HW Brands** node and then click **Add response condition.**

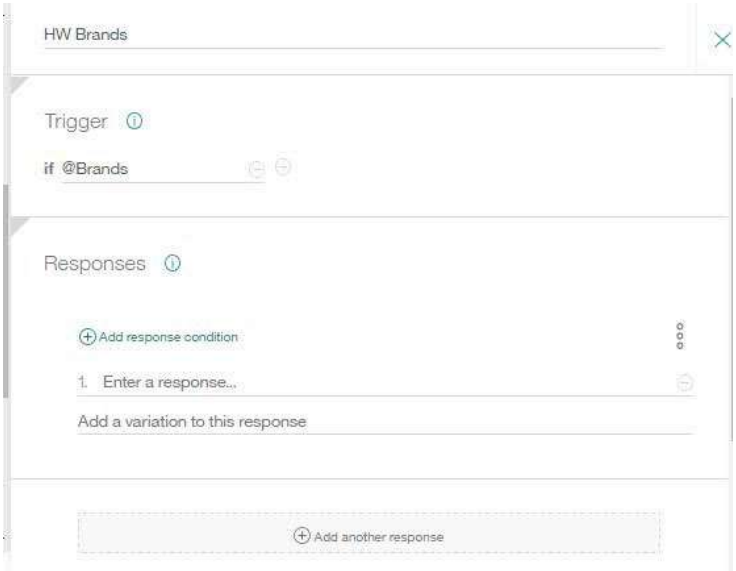


Figure Add response condition

b. Enter the appropriate response for each example in the @Brands entity

Figure shows the dialog branch built in this example for hardware issues.

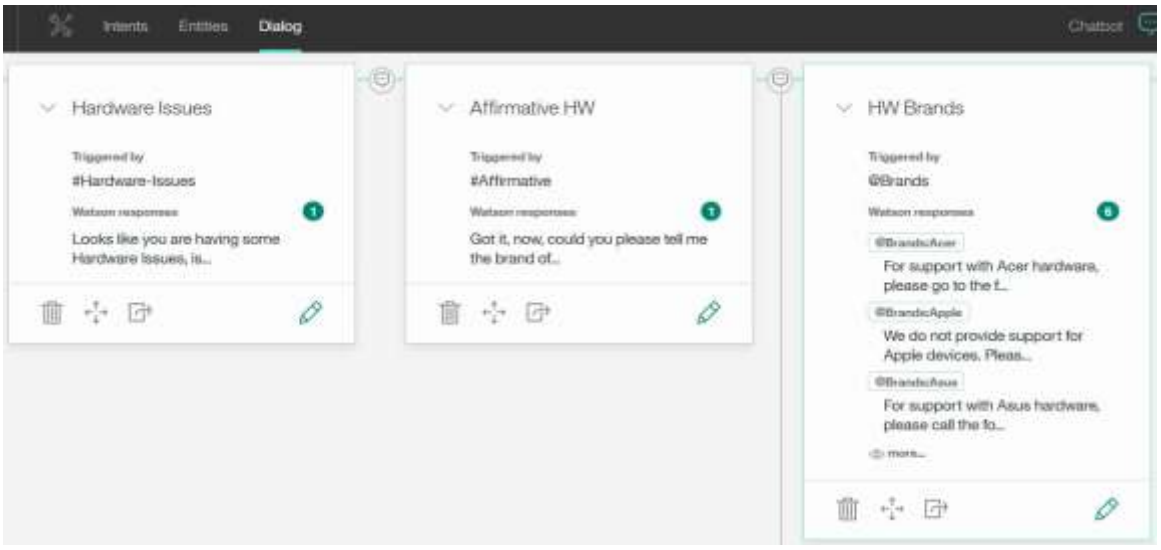


Figure Dialog branch for hardware issues

Repeat the process described earlier, for software - issues.

In the OS node, create a response for each example in the @OS entity (HPUX, Red Hat, Linux, Windows, UNIX, and so on) refer figure below.

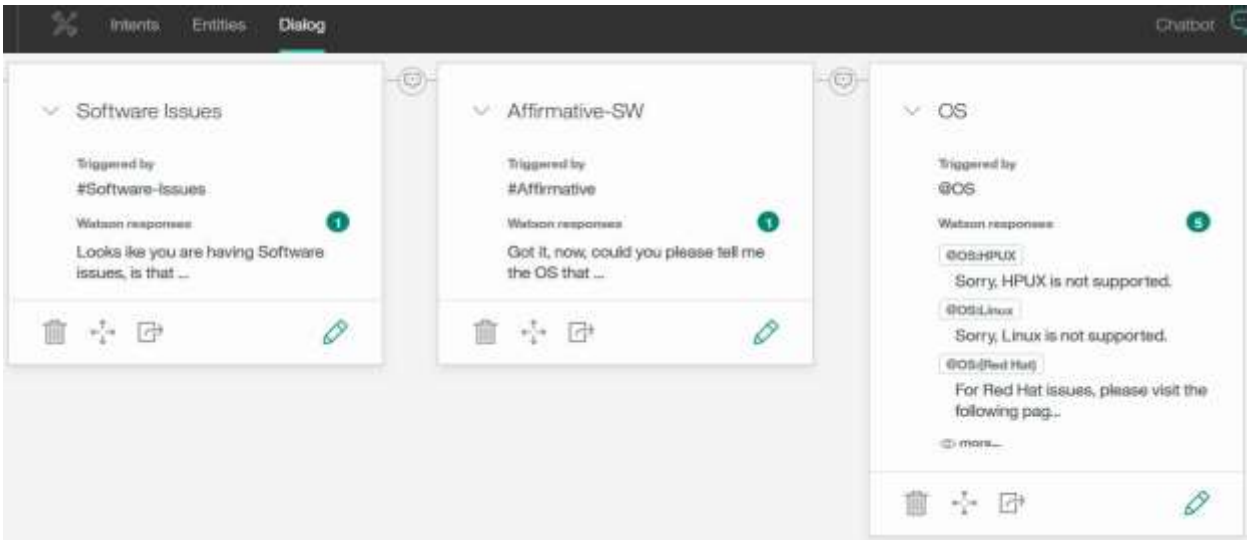
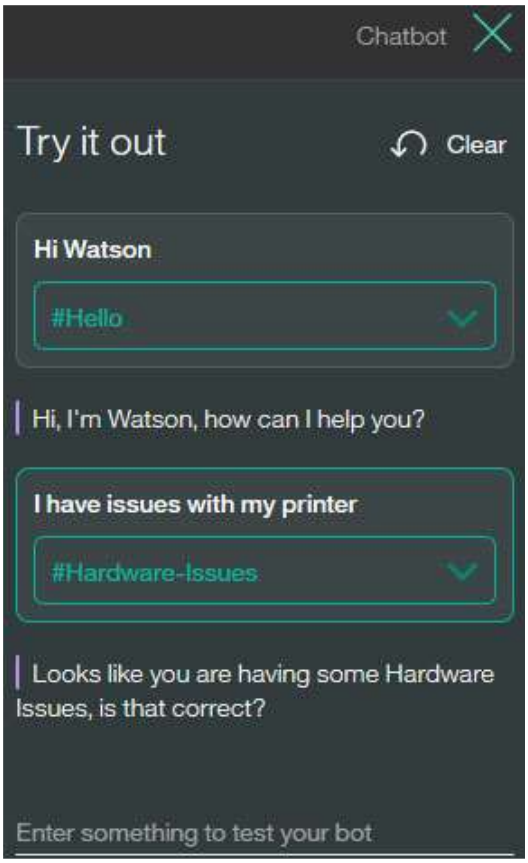


Figure Dialog branch for software issues

Step 5:

Testing the dialog

To test the dialog, first click the **Ask Watson** icon (upper right corner).
The Chatbot panel opens. Interact with the chatbot by asking questions to test the responses.



Step 6:

Creating the Help Desk Assistant - Chatbot application

Developing the Cognitive Chatbot application

This section describes how to use the sample application by creating a Node.js application that integrates with the Conversation service. You start by cloning a sample Node.js app, which is a simple chatbot, and deploy it to your Bluemix workspace.

The steps are summarized in the following list:
1. “Clone/Get Code the Conversation sample app”
2. “Integrate the application with the Conversation services”
3. “Push the application to Bluemix”

Step 1. “Clone/Get Code the Conversation sample app” on page 172
1. Create a new C:\chatbot directory.
2. Open a command prompt (cmd.exe).
3. Open that directory by using the **cd C:**chatbot command (Figure).

If you do not Git installed, please download for your OS from

https://git-scm.com/download

Once installed successfully.

Run the following Git command:
git clone https://github.com/watson-developer-cloud/conversation-simple

```
C:\Mangesh\Work\2017\3rdJuneChatbotHandsOn>git clone https://github.com/watson-developer-cloud/conversation-si
Cloning into 'conversation-simple'...
remote: Counting objects: 683, done.
remote: Total 683 (delta 0), reused 0 (delta 0), pack-reused 683
Receiving objects: 100% (683/683), 2.34 MiB | 329.00 KiB/s, done.
Resolving deltas: 100% (296/296), done.
```

You will have code download in C:\Chatbot directory.

Step 2. “Integrate the application with the Conversation services”

Modify the code to integrate the application with the Conversation service:

1. Update the manifest.yml file with the host name and the details of the Conversation service :

a. Open C:\~\conversation-simple\manifest.yml with your favorite text editor.
Default File

```
---
declared-services:
  my-conversation-service:
    label: conversation
    plan: free
applications:
- name: conversation-simple
  command: npm start
  path: .
  memory: 256M
  instances: 1
  services:
  - my-conversation-service
  env:
    NPM_CONFIG_PRODUCTION: false
```

What you will have to change

```
---
declared-services:
  my-conversation-service:
    label: conversation
    plan: free
applications:
- name: conversation-simple
  command: npm start
  path: .
  memory: 256M
  instances: 1
  services:
  - my-conversation-service
  env:
    NPM_CONFIG_PRODUCTION: false
```

Give Unique application name as you will be deploying on Bluemix Cloud.

Service Name: is the Conversation service, you have created previously while configuring the service in Bluemix

Save the file. It should look like below example shown

```
---
declared-services:
  my-conversation-service:
    label: conversation
    plan: free
applications:
- name: conversation-simple-mdp
  command: npm start
  path: .
  memory: 256M
  instances: 1
  services:
  - ConversationCustomerService
  env:
    NPM_CONFIG_PRODUCTION: false
```

1. Replace .env.example with right set of values and rename it to .env file

   Current .env.example will look like below

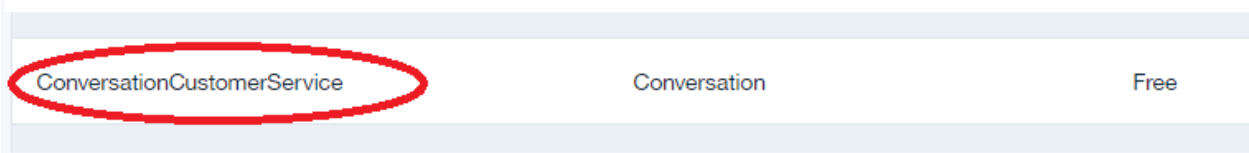
```
# Environment variables
WORKSPACE_ID=<workspace-id>
CONVERSATION_USERNAME=<conversation-username>
CONVERSATION_PASSWORD=<conversation-password>
```

Update Workspace id and Access Credentials of Watson Conversation Service. How to get that? Follow below steps

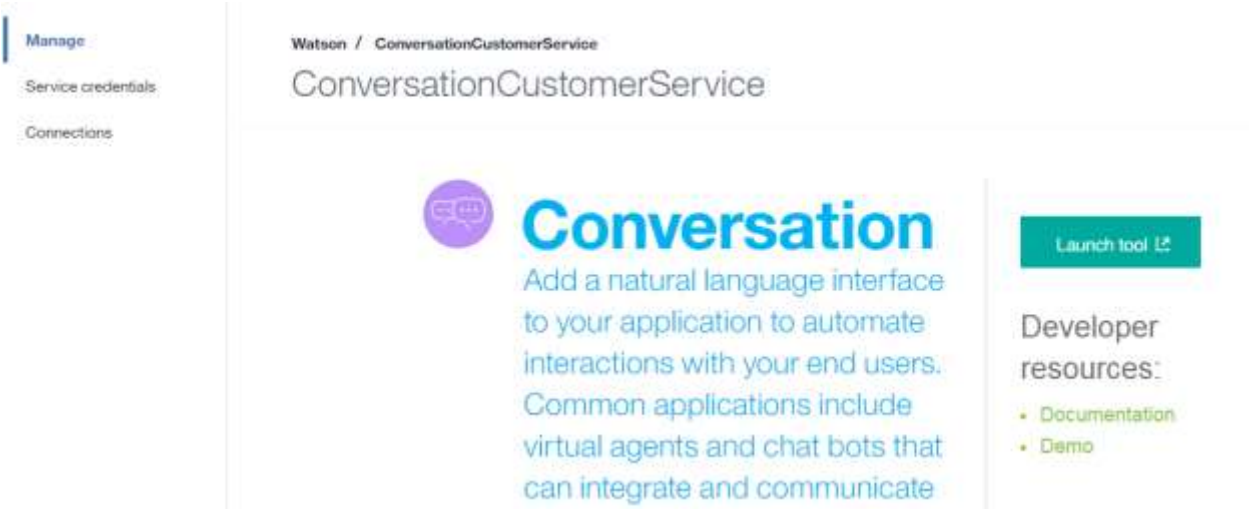
a) To get Workspace Id

   Go to Bluemix Services Dashboard:

   Select the conversation service, you created

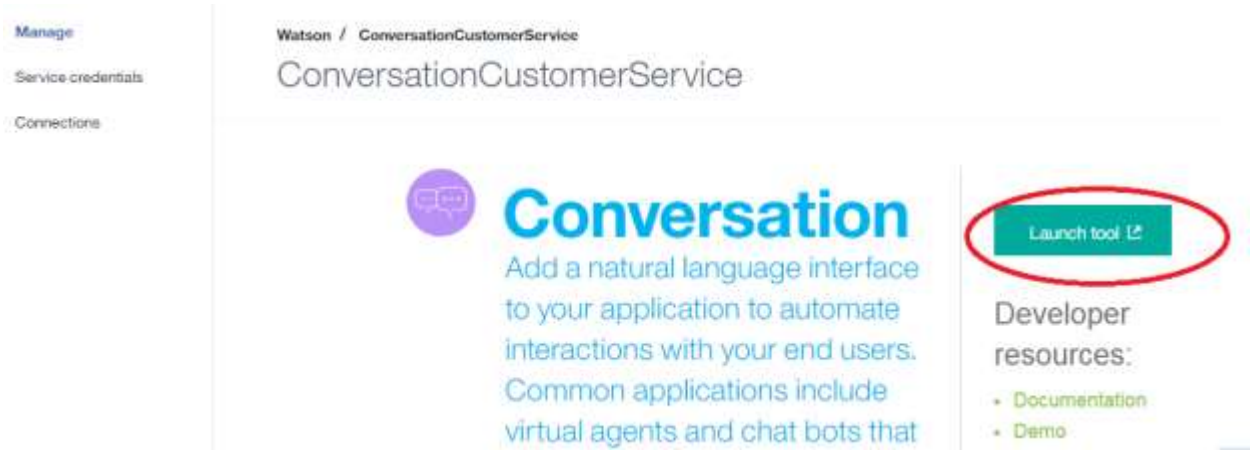


Once you click on Service, following screen comes up

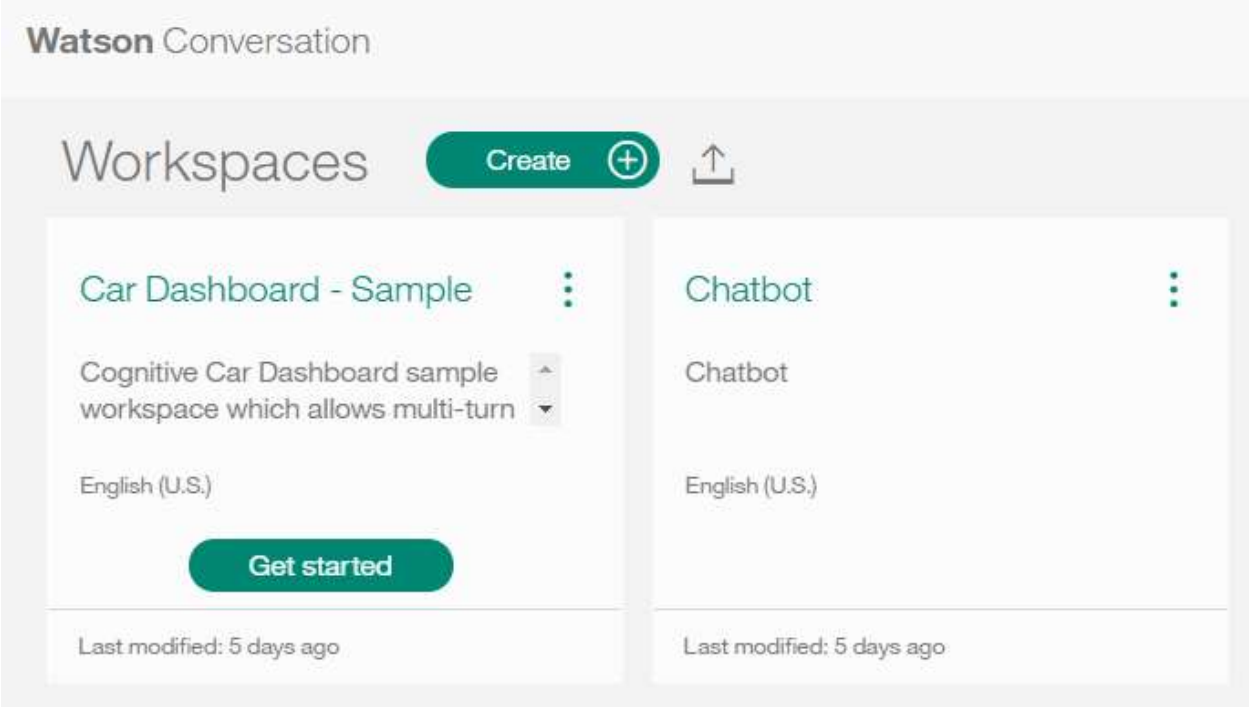


To Get access Credential → Click on Service Credentials and note username and password.

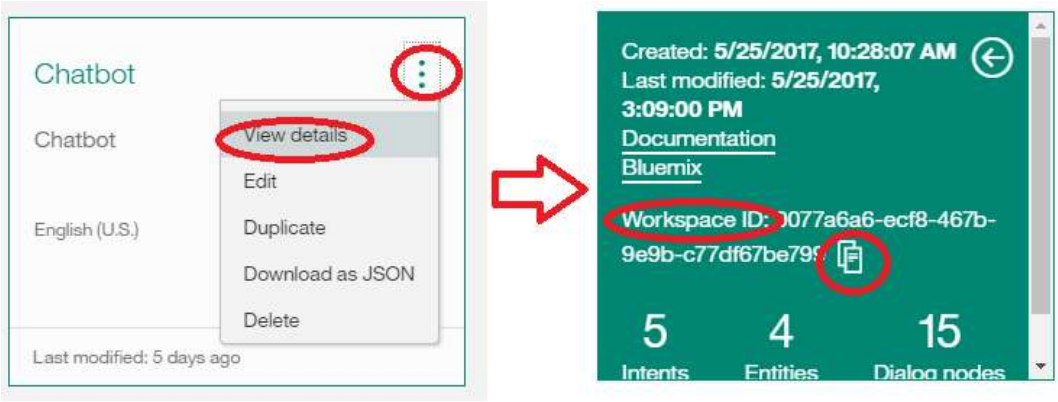
And you have to get Workspace_id as well. To get that Click on previous menu



You will see workspaces created by you



Select Chatbot and copy the workspace id as shown below



Update these parameters and rename this file as ".env" file.

Save ".env" file. File should look like

```
# Environment variables
WORKSPACE_ID=9077a6a6-ecf8-467b-9e9b-c77df67be799
CONVERSATION_USERNAME=deaca612-5c60-4337-9363-a5afe881476d
CONVERSATION_PASSWORD=jwUgNHPhqvTH
```

Step 3. Push the application to Bluemix

Push the modified code to Bluemix:

1. At the command prompt, change to the C:\~\conversation-simple-<app-name> directory
2. Log in to Cloud Foundry by using the **cf login** command. When prompted enter the email and password that you use to log in to your Bluemix account.

```
C:\Mangesh\Work\2017\3rdJuneChatbotHandsOn>cf login
API endpoint: https://api.ng.bluemix.net

Email> mapatank@in.ibm.com

Password>
Authenticating...
OK

Select an org (or press enter to skip):
1. mapatank@in.ibm.com
2. gurudutt.kamath@icicibank.com

Org> 1
Targeted org mapatank@in.ibm.com

Select a space (or press enter to skip):
1. MySpace
2. ICICIDev
3. Dev
4. ohaidemo
5. demo

Space> 1
Targeted space MySpace


API endpoint:   https://api.ng.bluemix.net (API version: 2.54.0)
User:           mapatank@in.ibm.com
Org:            mapatank@in.ibm.com
Space:          MySpace

C:\Mangesh\Work\2017\3rdJuneChatbotHandsOn>_
```

Log in to Cloud Foundry (cf login)

```
C:\windows\system32\cmd.exe - cf push

C:\redbook\conversation-simple>cf push
Using manifest file C:\redbook\conversation-simple\manifest.yml

Creating app conv-201-xxx-weather in org aazraq@eg.ibm.com / space Conversation
as aazraq@eg.ibm.com...
OK

Creating route conv-201-xxx-weather.mybluemix.net...
OK

Binding conv-201-xxx-weather.mybluemix.net to conv-201-xxx-weather...
OK

Uploading conv-201-xxx-weather...
Uploading app files from: C:\redbook\conversation-simple
Uploading 1.1M, 73 files
Done uploading
OK
Binding service Conversation to app conv-201-xxx-weather in org aazraq@eg.ibm.co
m / space Conversation as aazraq@eg.ibm.com...
OK
Binding service weather-company-data to app conv-201-xxx-weather in org aazraq@e
g.ibm.com / space Conversation as aazraq@eg.ibm.com...
OK

Starting app conv-201-xxx-weather in org aazraq@eg.ibm.com / space Conversation
as aazraq@eg.ibm.com...
Downloading swift_buildpack_v2_0_3-20161217-1748...
Downloading python_buildpack...
Downloading java_buildpack...
Downloading ruby_buildpack...
Downloading nodejs_buildpack...
Downloading go_buildpack...
Downloaded swift_buildpack_v2_0_3-20161217-1748
Downloaded nodejs_buildpack
Downloading liberty-for-java...
Downloaded ruby_buildpack
Downloading xpages_buildpack...
Downloaded go_buildpack
Downloading php_buildpack...
Downloaded sdk-for-nodejs
Downloading staticfile_buildpack...
Downloaded liberty-for-java
```

Wait until the build and deployment are completed (Figure).

```
C:\windows\system32\cmd.exe

0 of 1 instances running, 1 starting
1 of 1 instances running

App started


OK

App conv-201-xxx-weather was started using this command `npm start`

Showing health and status for app conv-201-xxx-weather in org aazraq@eg.ibm.com
/ space Conversation as aazraq@eg.ibm.com...
OK

requested state: started
instances: 1/1
usage: 512M x 1 instances
urls: conv-201-xxx-weather.mybluemix.net
last uploaded: Mon Feb 20 00:36:41 UTC 2017
stack: cflinuxfs2
buildpack: SDK for Node.js(TM) (ibm-node.js-4.7.2, buildpack-v3.10-20170119-1146
)

     state     since                    cpu    memory       disk      details
#0   running   2017-02-20 03:39:03 AM   0.0%   0 of 512M    0 of 1G

C:\redbook\conversation-simple>_
```

Figure Pushing application completed

Wait until the build and deployment are completed (Figure).

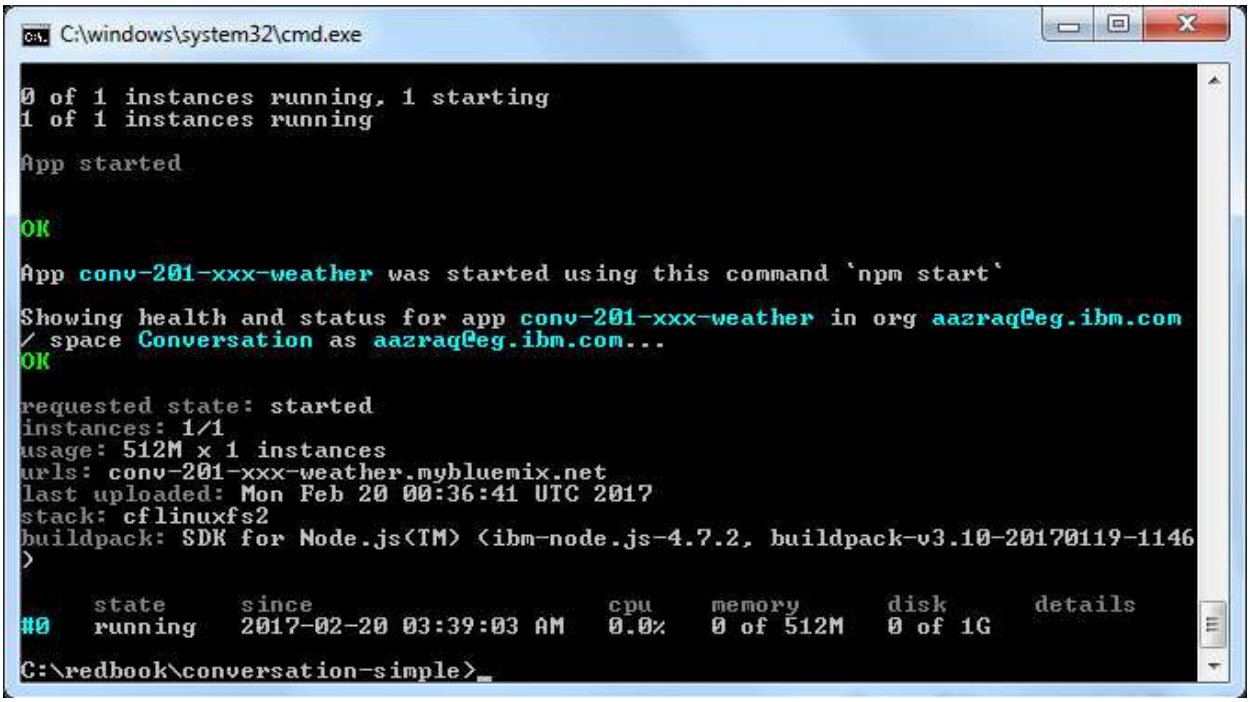


Figure Pushing application completed

Wait until the application is running

Testing the application

To test the application, follow these steps:
1. Open your application route (URL to access your application) in a web browser; xxx is the number you use to make your application name unique:
http://conv-201-xxx-weather.mybluemix.net/
Your application opens in the browser (Figure).

3. Check support queries for Software and Hardware issues

4. Try different scenarios. If the chatbot fails, more training is necessary. To provide more training, add more user examples to the intents in the Chat-bot Workspace, or edit the entities.

Congratulations !! Completed Lab yourself !!!

If you are not able to create Conversation Workspace follow quick steps below and integrate with simple application mentioned in Step 6 of above.

Quick deployment of service and application

Step 1: If you have not developed a Chatbot Workspace containing intents, entities and dialog. You can import the following location /file to get conversation workspace creation.

The workspace that was created for this chapter is in the following GitHub location:

https://github.com/mdpatankar/ChatbotHandsOn

Step 2: follow Step 6 from above to integrate with Chatbat Workspace

Further Reading :

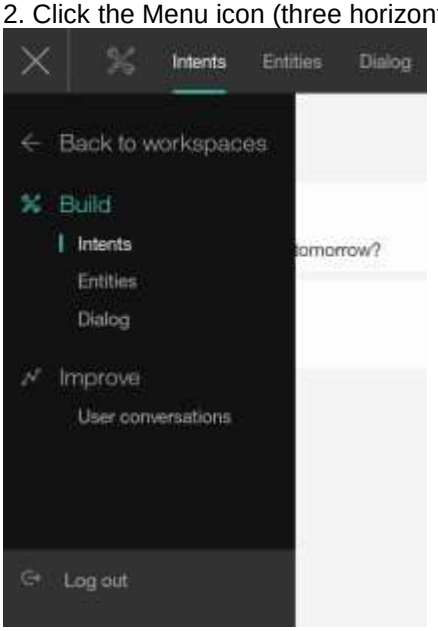
Using the Improve component to train the Conversation workspace

The *Improve* component of the Conversation service provides a history of conversations with users. You can use this history to improve your chatbot's understanding of user inputs. While you develop your workspace, you use the *Try it out* panel to verify that it recognizes the correct intents and entities in test inputs, and make corrections as needed. In the Improve panel, you can view actual conversations with your users and make similar corrections to improve the accuracy with which intents and entities are recognized.

In this example, you use the sample Car Chat-bot workspace to conduct a simple dialog with the user, and try to get information by communicating your intents and entities in unexpected ways.

Access the Improve component and open the chat logs

To access the Improve component and open the chat logs for the Car Chat-bot workspace:
1. Open the Car Chat-bot workspace
2. Click the Menu icon (three horizontal lines). Then, select **Improve → User conversations**



The chat logs saved represent the user interactions through the API (*not* the interactions through the *Try it out* panel in the workspace). The Improve feature shows you the most recent user interactions. The top intent and any entities used in the message, the message text, and the chatbot's reply are available.