

Project Report

Name : *UTTAM SHARMA*(uttamsharma231@gmail.com)

Title : *Intelligent Customer Help Desk With Smart
Document Understanding*

Category: *Artificial Intelligence*

Internship at smartinternz.com@2020

Project Overview: We will be able to write an application that leverages multiple Watson AI

Services (Discovery , Assistant, Cloud function and Node Red). By the end of the project, we'll learn best practices of combining Watson services, and how they can build interactive information retrieval systems with Discovery + Assistant.

- ❖ **Project Requirements:** Python, IBM Cloud, IBM Watson
- ❖ **Functional Requirements:** IBM cloud
- ❖ **Technical Requirements:** AI,ML,WATSON AI,PYTHON
- ❖ **Software Requirements:** Watson assistant, Watson discovery.
- ❖ **Project Deliverables:** Smartinternz Internship
- ❖ **Project Team:** Uttam Sharma
- ❖ **Project Duration:**19 days

Project Description:

The typical customer care chatbot can answer simple questions, such as store locations and hours, directions, and maybe even making appointments. When a question falls outside of the scope of the pre-determined question set, the option is typically to tell the customer the question isn't valid or offer to speak to a real person.

In this project, there will be another option. If the customer question is about the operation of a device, the application shall pass the question onto Watson Discovery Service, which has been pre-loaded with the device's owners manual. So now, instead of "Would you like to speak to a customer representative?" we can return relevant sections of the owners manual to help solve our customers' problems.

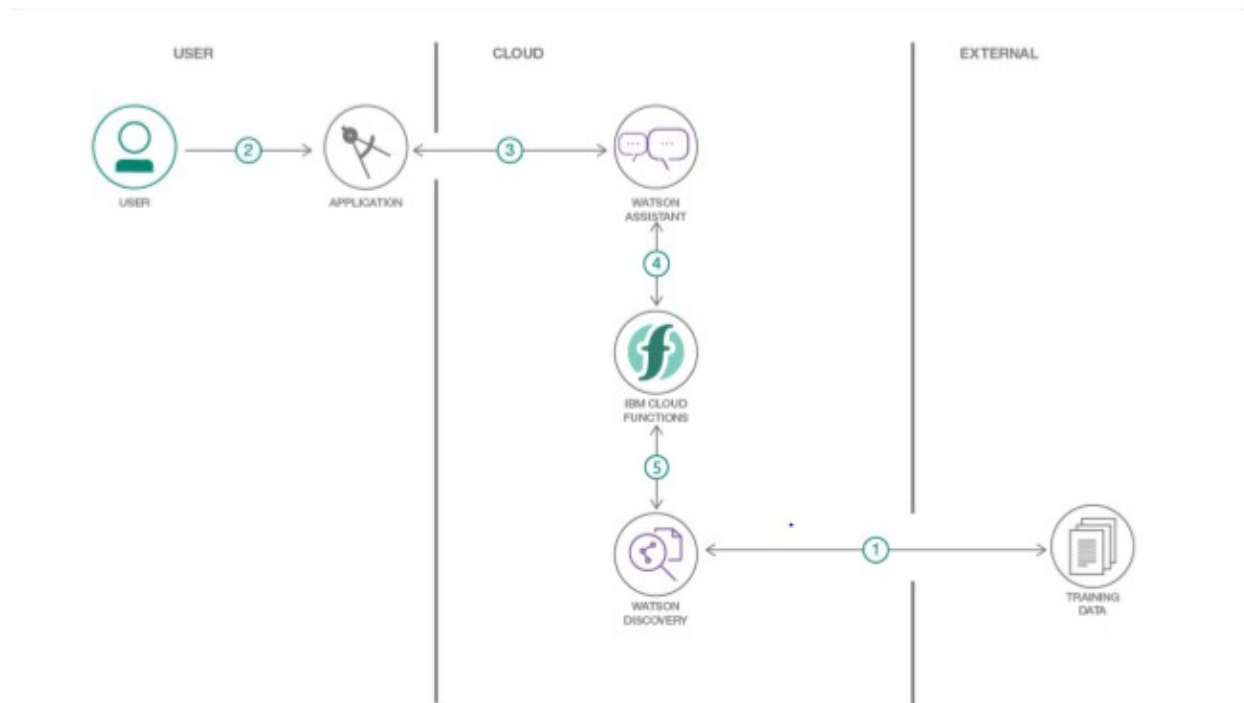
To take it a step further, the project shall use the Smart Document Understanding feature of Watson Discovery to train it on what text in the owners manual is important and what is not. This will improve the answers returned from the queries.

Scope of Work

- ✓ Create a customer care dialog skill in Watson Assistant
- ✓ Use Smart Document Understanding to build an enhanced Watson Discovery collection
- ✓ Create an IBM Cloud Functions web action that allows Watson Assistant to post queries to Watson Discovery

- ✓ Build a web application with integration to all these services & deploy the same on IBM Cloud Platform

Flow Diagram



1. The document is annotated using Watson Discovery SDU
2. The user interacts with the backend server via the app UI. The frontend app UI is a chatbot that engages the user in a conversation.
3. Dialog between the user and backend server is coordinated using a Watson Assistant dialog skill.
4. If the user asks a product operation question, a search query is passed to a predefined IBM Cloud Functions action.
5. The Cloud Functions action will query the Watson Discovery service and return the results.

Steps:

1. Create IBM Cloud services
2. Configure Watson Discovery
3. Create IBM Cloud Functions action

4. Configure Watson Assistant
5. Create flow and configure node
6. Deploy and run Node Red app.

1.Create IBM Cloud services

Create the following services:

- Watson Discovery
- Watson Assistant
- Node Red

2. Configure Watson Discovery

Import the document

Launch the Watson Discovery tool and create a new data collection by selecting the Upload your own data option. Give the data collection a unique name. When prompted, select and upload the ecobee3_UserGuide.pdf file located in the data directory of your local repo.

The Ecobee is a popular residential thermostat that has a wifi interface and multiple configuration options.

Before applying SDU to our document, lets do some simple queries on the data so that we can compare it to results found after applying SDU.

The screenshot shows the IBM Watson Discovery Overview page for the 'ecobee' dataset. The top navigation bar includes 'Cookie Preferences', 'Instance: Discovery-od', and a 'Configure data' link. The main content area has tabs for 'Overview', 'Errors and warnings (1)', and 'Search settings'. The Overview tab is active, showing 1 document, 0 failed documents, and creation/last updated dates of 6/6/2019 6:25:40 pm EDT. An 'Upload documents' button is present. Below this, the 'Identified 1 field from your data' section shows 'text'. The 'Added 3 enrichments to your data' section includes Sentiment Analysis (100% positive, 0% neutral, 0% negative) and Concept Tagging (Air conditioner (1), Energy recovery (1), Geothermal heat pump (1)). A 'Category Classification' section shows 'business and industr... energy'. On the right, a 'Now you're ready to query!' section has two 'Run' buttons for queries like 'Documents that contain Air conditioner, but not Energy recovery' and 'Top people related to /business and industrial/energy'. A red '1 Build your own query' button is at the bottom right.

Click the Build your own query [1] button.

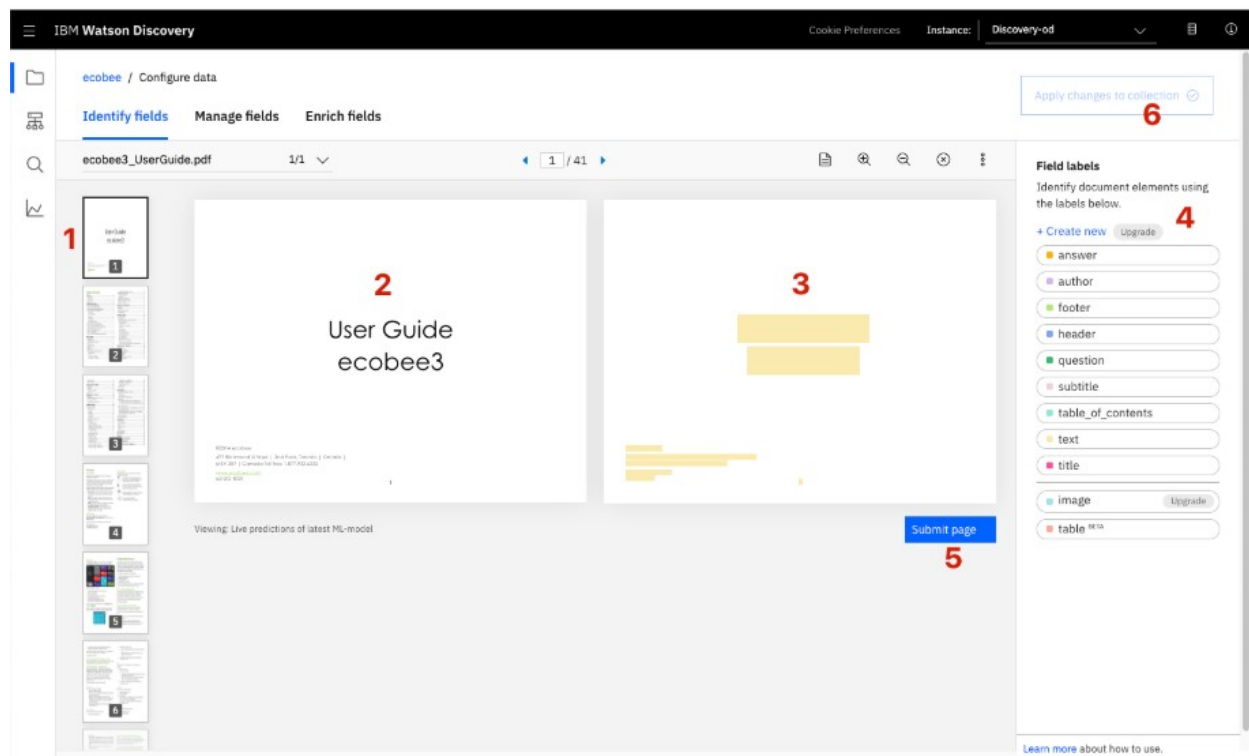
The screenshot shows the IBM Watson Discovery 'Build queries' page. The left sidebar has a search bar with the query 'how do I turn on the heater?'. Below the search bar are options to 'Include analysis of your results' and 'Filter which documents you query'. The main content area shows the 'Summary' tab for the query, with a 'JSON' tab also available. The 'Summary' tab displays the 'QUERY URL' as 'https://gateway.watsonplatform.net/discovery/api/v1/enviro'. The 'Passages' section shows a snippet from the document: 'No warranties, whether express or implied, will apply after the limited warranty period has expired. Some US states and Canadian provinces do not allow limitations on how long an implied warranty lasts, so this limitation may not apply. ecobee neither assumes responsibility'. The 'Results' section shows 'Showing 1 of 1 matching documents' and lists the document 'ecobee3_UserGuide.pdf' with a sentiment of 'positive'. The 'Text' section shows a snippet: '...Find how-to videos and tutorials on... Conventional heating and cooling Heat Only Do not jumper Rc or Rh, ecobee3 does this automatically. R can go into either Rc or Rh terminals on your ecobee3. Heat and Cool Do not jumper Rc or Rh, ecobee3 does this automatically. R can go into either Rc or Rh'. At the bottom, there are 'Run query' and 'Close' buttons.

Enter queries related to the operation of the thermostat and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question.

Annotate with SDU

Now let's apply SDU to our document to see if we can generate some better query responses. From the Discovery collection panel, click the Configure data button (located in the top right corner) to start the SDU process.

Here is the layout of the Identify fields tab of the SDU annotation panel:



The goal is to annotate all of the pages in the document so Discovery can learn what text is important, and what text can be ignored.

[1] is the list of pages in the manual. As each is processed, a green check mark will appear on the page.

[2] is the current page being annotated.

[3] is where you select text and assign it a label.

[4] is the list of labels you can assign to the page text.

Click [5] to submit the page to Discovery.

Click [6] when you have completed the annotation process.

As you go through the annotations one page at a time, Discovery is learning and should start automatically updating the upcoming pages. Once you get to a page that is already correctly annotated, you can stop, or simply click Submit [5] to acknowledge it is correct. The more pages you annotate, the better the model will be trained.

For this specific owner's manual, at a minimum, it is suggested to mark the following:

The main title page as title

The table of contents (shown in the first few pages) as table_of_contents

All headers and sub-headers (typed in light green text) as a subtitle

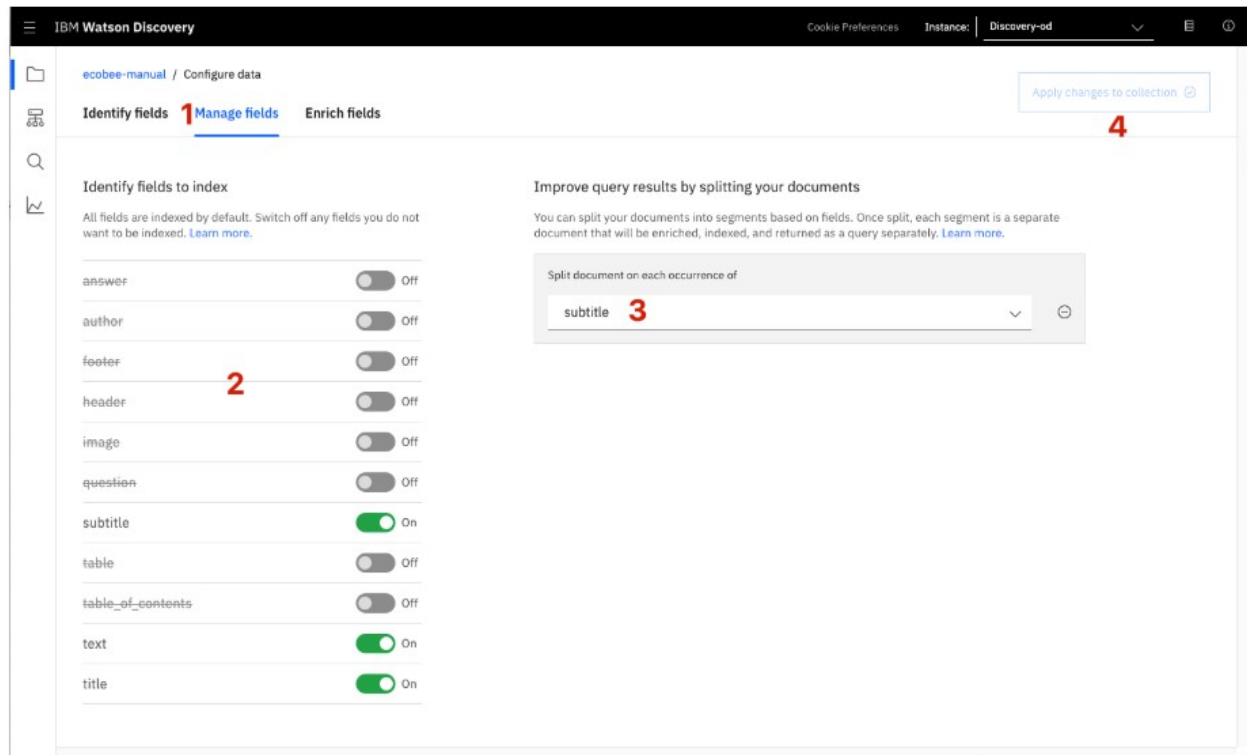
All page numbers as footers

All warranty and licensing information (located in the last few pages) as a footer

All other text should be marked as text.

Once you click the Apply changes to collection button [6], you will be asked to reload the document. Choose the same owner's manual .pdf document as before.

Next, click on the Manage fields [1] tab.



[2] Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except subtitles and text.

[3] is telling Discovery to split the document apart, based on subtitle.

Click [4] to submit your changes.

Once again, you will be asked to reload the document.

Now, as a result of splitting the document apart, your collection will look very different:

The screenshot shows the IBM Watson Discovery Overview page for a dataset named 'ecobee-manual'. The page displays 130 documents, 0 failed documents, and search settings. It highlights identified fields (footer, subtitle, table_of_contents, text, title) and added enrichments (Entity Extraction, Sentiment Analysis, Concept Tagging, Category Classification). The Sentiment Analysis section shows 37% positive, 26% neutral, and 36% negative sentiment. The Category Classification section shows 'technology and com...' and 'operating systems'. The page also includes a 'Build your own query' link.

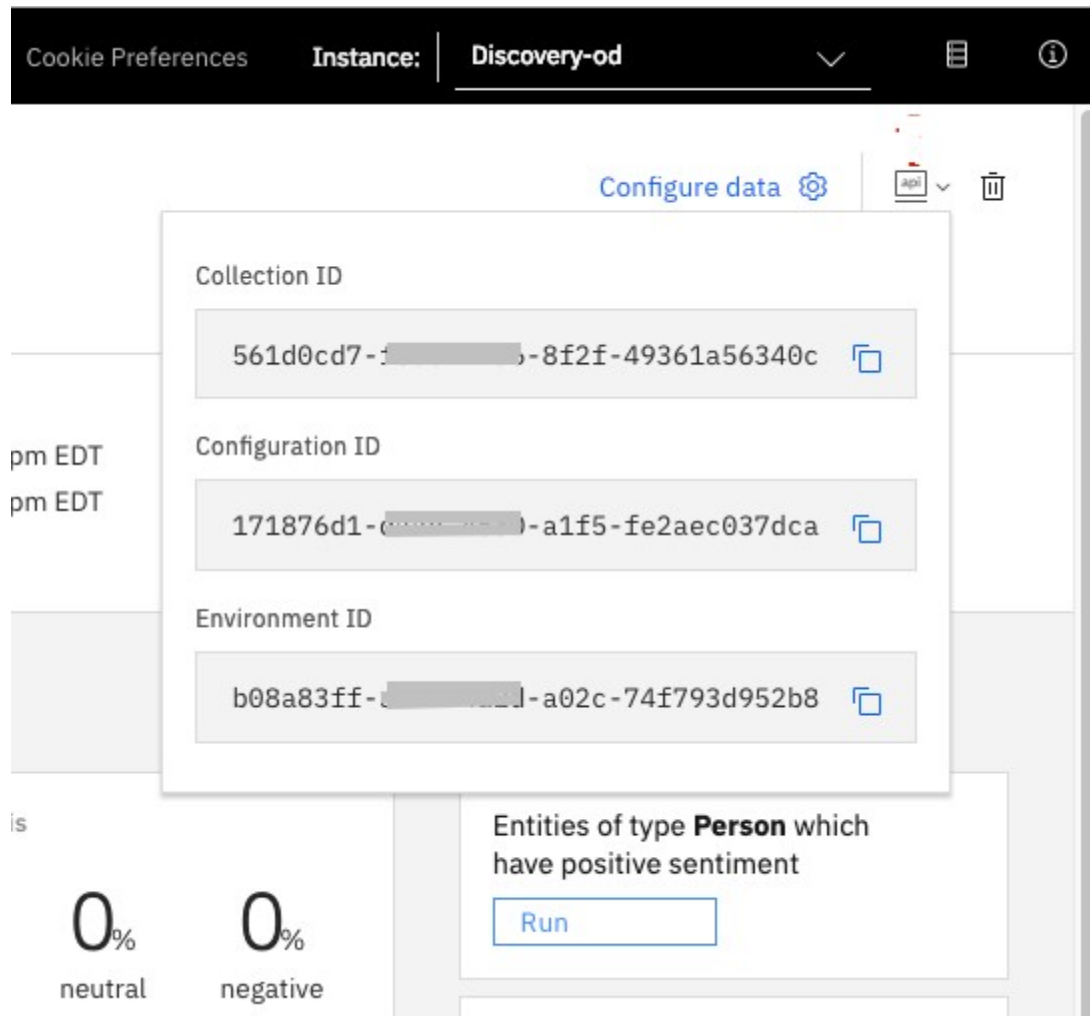
Return to the query panel (click Build your own query) and see how much better the results are.

The screenshot shows the IBM Watson Discovery Query panel. The search query is 'how do I turn on the heater?'. The results are displayed in a list format, showing 10 of 38 matching documents. The first result is 'ecobee3_UserGuide.pdf'. The panel also includes a 'Run query' button and a 'Close' button.

Store credentials for future use

In upcoming steps, you will need to provide the credentials to access your Discovery collection. The values can be found in the following locations.

The Collection ID and Environment ID values can be found by clicking the dropdown button [1] located at the top right side of your collection panel:



For credentials, return to the main panel of your Discovery service, and click the Service credentials [1] tab:

IBM Cloud

Search resources and offerings...

Resource list /

Discovery-od

Resource group: default Location: Dallas Add Tags

Service credentials

Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service. [Learn more](#)

Service credentials

New credential

Items per page 10 | 1-1 of 1 items 1 of 1 pages

KEY NAME	DATE CREATED	ACTIONS
Service credentials-1	FEB 5, 2019 - 09:26:31 AM	View credentials 2

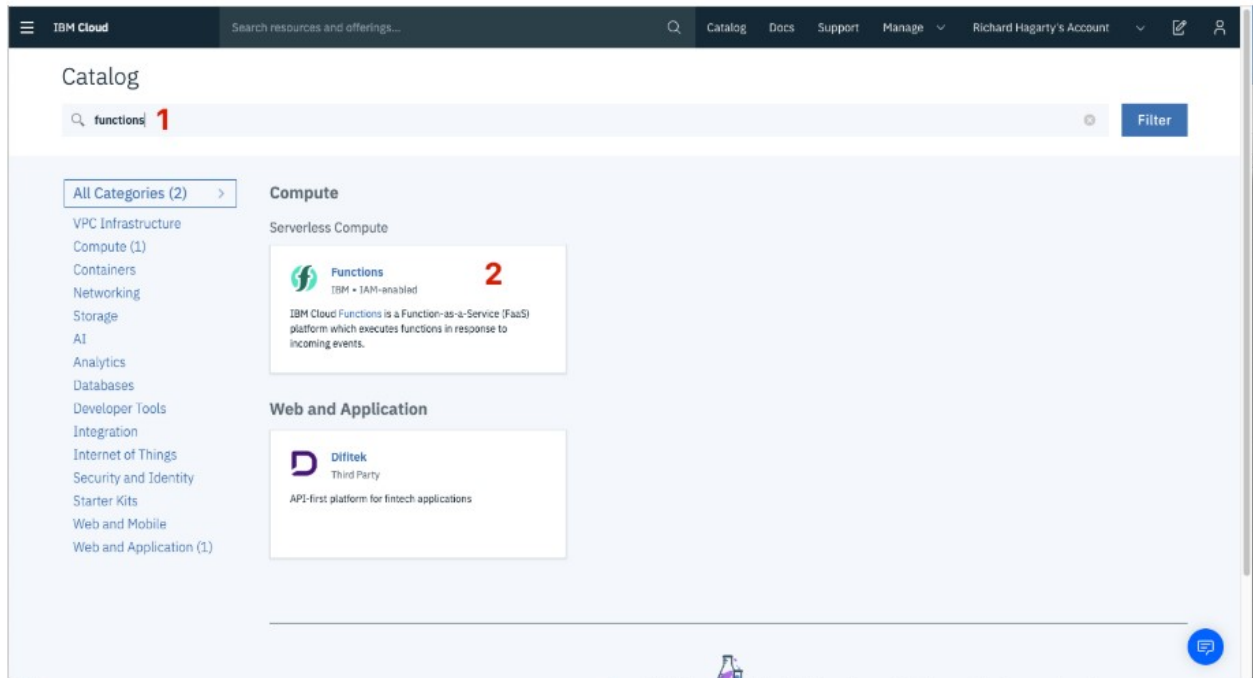
```
{
  "apikey": "dry8f3a1Tnsy; [REDACTED] #h1au8bko4fu10",
  "iam_apikey_description": "Auto generated apikey during resource-key operation for Instance - crn:v1:bluemix:public:discovery:us-south:a/bc1bd51c396536dc7d5f81d5a4e19533:acf2871-3b0d-4e04-a0f9-0da59779852::",
  "iam_apikey_name": "auto-generated-apikey-f5i36cdd-did2-4a17-b41d-8ca5d1f1c7a6",
  "iam_role_crn": "crn:v1:bluemix:public:iam:::serviceRole:Manager",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity:a/bc1bd51c396536dc7d5f81d5a4e19533::serviceid:ServiceId-016b8efa-a050-4708-a191-0b71f43c0ddb",
  "url": "https://gateway.watsonplatform.net/discovery/api"
}
```

Click the View credentials [2] drop-down menu to view the IAM apikey [3] and URL endpoint [4] for your service.

3.Create IBM Cloud Functions action

Now let's create the web action that will make queries against our Discovery collection.

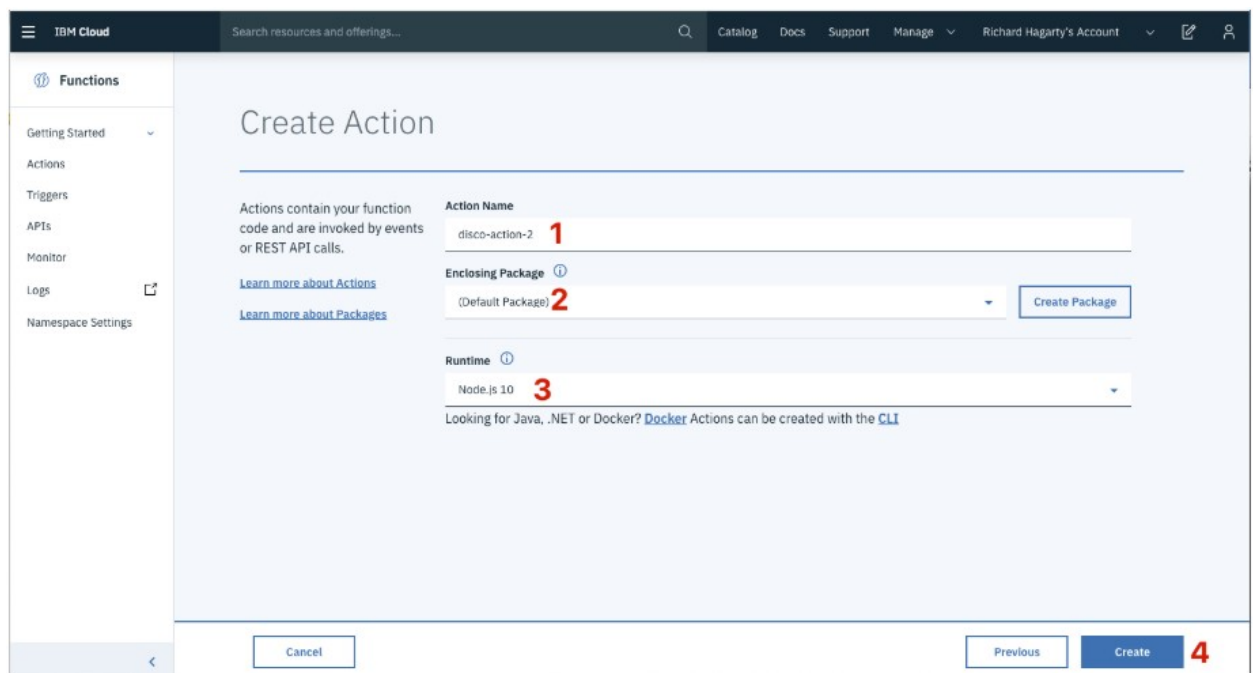
Start the IBM Cloud Functions service by selecting Create Resource from the IBM Cloud dashboard. Enter functions as the filter [1], then select the Functions card [2]:



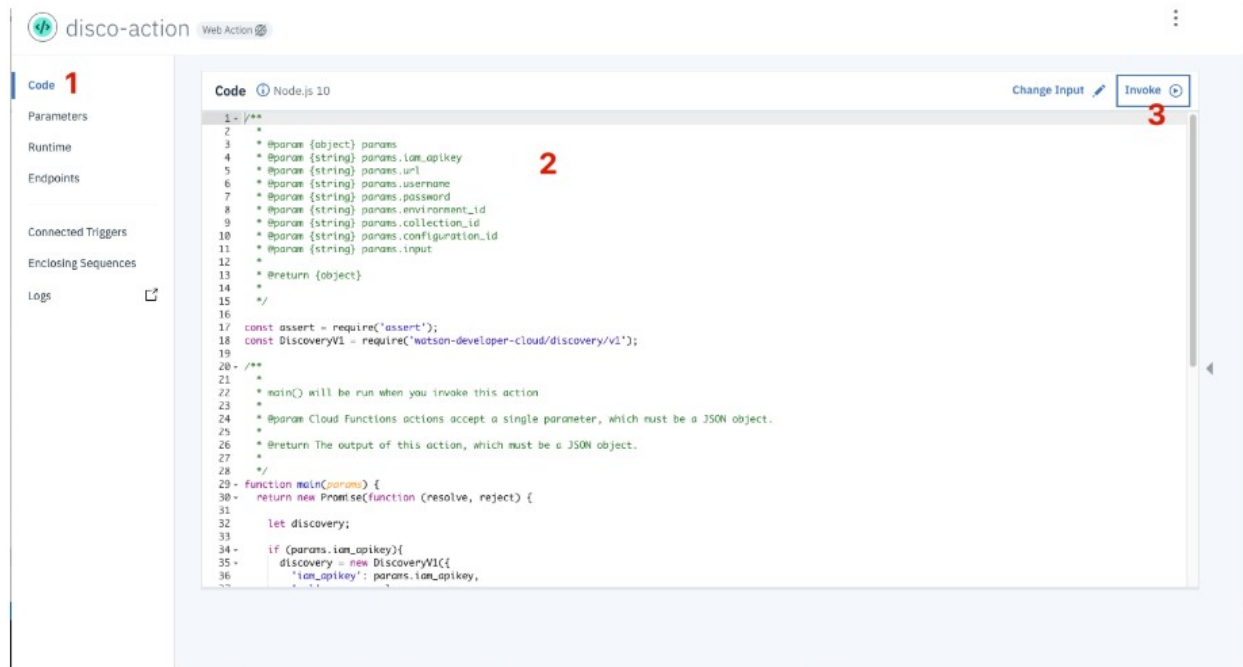
From the Functions main panel, click on the Actions tab. Then click on Create.

From the Create panel, select the Create Action option.

On the Create Action panel, provide a unique Action Name [1], keep the default package [2], and select the Node.js 10 [3] runtime. Click the Create button [4] to create the action.



Once your action is created, click on the Code tab [1]:



In the code editor window [2], cut and paste in the code from the disco-action.js file found in the actions directory of your local repo. The code is pretty straight-forward - it simply connects to the Discovery service, makes a query against the collection, then returns the response.

If you press the Invoke button [3], it will fail due to credentials not being defined yet. We'll do this next.

Select the Parameters tab [1]:



Add the following keys:

- ❖ url
- ❖ environment_id
- ❖ collection_id
- ❖ iam_apikey

For values, please use the values associated with the Discovery service you created in the previous step.

Now that the credentials are set, return to the Code panel and press the Invoke button again. Now you should see actual results returned from the Discovery service:

The screenshot shows the IBM Cloud Functions console for the 'disco-action' function. The 'Code' panel displays the following Node.js code:

```
1- /**
2-  * @param {object} params
3-  * @param {string} params.iam_apikey
4-  * @param {string} params.url
5-  * @param {string} params.username
6-  * @param {string} params.password
7-  * @param {string} params.environment_id
8-  * @param {string} params.collection_id
9-  * @param {string} params.configuration_id
10-  * @param {string} params.input
11-  * @param {string} params.input
12-  * @return {object}
13-  */
14-
15-
16-
17- const assert = require('assert');
18- const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');
19-
20- /**
21-  *
22-  * main() will be run when you invoke this action
23-  *
24-  * @param Cloud Functions actions accept a single parameter, which must be a JSON object.
25-  *
26-  * @return The output of this action, which must be a JSON object.
27-  *
28-  */
29- function main(params) {
30-   return new Promise(function (resolve, reject) {
31-
32-     let discovery;
33-
34-     if (params.iam_apikey){
35-       discovery = new DiscoveryV1({
36-         'iam_apikey': params.iam_apikey,
37-         'url': params.url,
38-         'version': '2019-03-25'
39-       });
40-     }
41-
42-     // ... (rest of the code) ...
43-   });
44- }
```

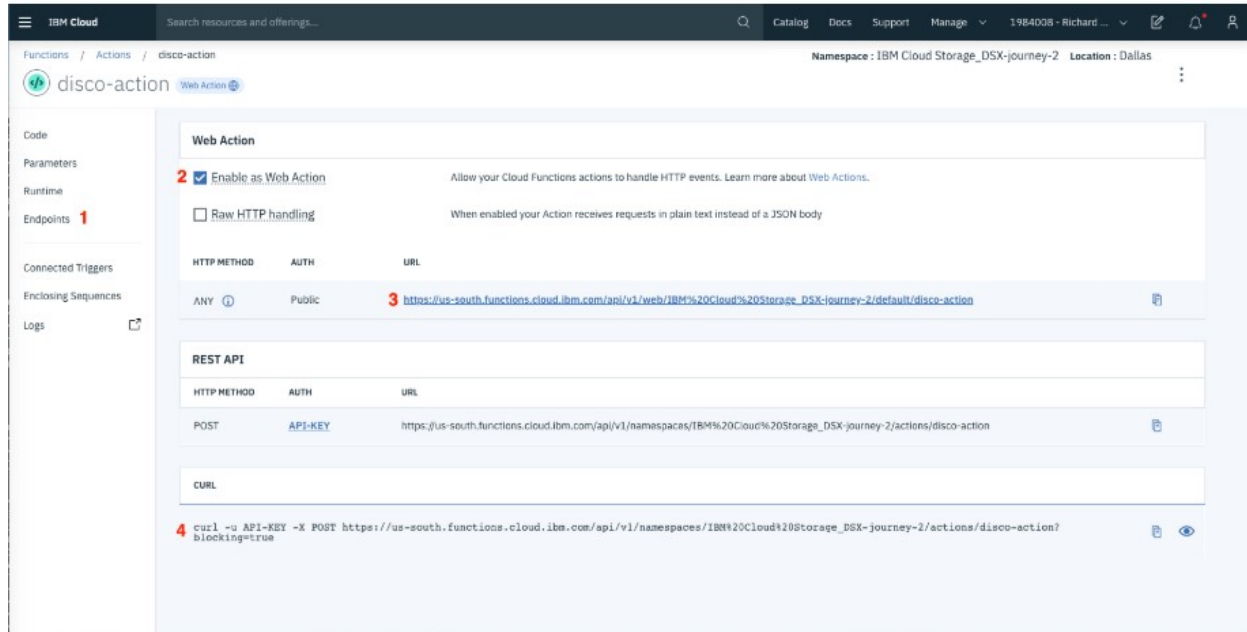
The 'Activations' panel shows the results of a successful invocation:

```

Activation ID:
e1bfc0ff21544c85bdc6ff21549e85a1

Results:
{
  "matching_results": 14,
  "passages": [],
  "results": [
    {
      "enriched_text": {
        "categories": [
          {
            "label": "/technology and computing/operating systems",
            "score": 0.842265
          },
          {
            "label": "/technology and computing/hardware/computer:",
            "score": 0.835879
          },
          {
            "label": "/technology and computing/hardware/computer peripherals/computer monitors",
            "score": 0.832254
          }
        ],
        "concepts": [
          {
            "dbpedia_resource": "http://dbpedia.org/resource/IPhone",
            "relevance": 0.917306,
            "text": "IPhone"
          },
          {
            "dbpedia_resource": "http://dbpedia.org/resource/Personal_digital_assistant",
            "relevance": 0.887088,
            "text": "Personal digital assistant"
          }
        ]
      }
    }
  ]
}
```

Next, go to the Endpoints panel [1]:



Click the checkbox for Enable as Web Action [2]. This will generate a public endpoint URL [3].

Take note of the URL value [3], as this will be needed by Watson Assistant in a future step.

To verify you have entered the correct Discovery parameters, execute the provided curl command [4]. If it fails, re-check your parameter values.

4. Configure Watson Assistant

Launch the Watson Assistant tool and create a new dialog skill. Select the Use sample skill option as your starting point. This dialog skill contains all of the nodes needed to have a typical call center conversation with a user.

Add new intent

The default customer care dialog does not have a way to deal with any questions involving outside resources, so we will need to add this.

Create a new intent that can detect when the user is asking about operating the Ecobee thermostat.

From the Customer Care Sample Skill panel, select the Intents tab.

Click the Create intent button.

Name the intent #Product_Information, and at a minimum, enter the following example questions to be associated with it.

The screenshot shows the Google Assistant 'Create intent' interface. At the top, the intent is named '#Product_Information' and was last modified 2 hours ago. Below this, there are fields for 'Intent name' (with a hint to match a customer's question or goal), 'Description (optional)' (with the example 'User wants help using the thermostat'), and 'Add user example' (with a placeholder 'Type a user example here'). There are buttons for 'Add example' and 'Show recommendations'. At the bottom, a table lists three user examples, all added 2 hours ago, with no conflicts.

<input type="checkbox"/> User examples (3) ▼	Added	0 conflicts	Show only conflicts ⓘ
<input type="checkbox"/> How do I access the settings ✎	2 hours ago		
<input type="checkbox"/> How do I set the time ✎	2 hours ago		
<input type="checkbox"/> How do I turn on the heater ✎	2 hours ago		

Create new dialog node

Now we need to add a node to handle our intent. Click on the Dialog [1] tab, then click on the drop down menu for the Small Talk node [2], and select the Add node below [3] option.

IBM Watson Assistant

[Skills](#) /

Customer Care Sample Skill copy

Sample simple customer service skill to get you started.

Intents Entities **1 Dialog** Analytics Options Versions Content Catalog

>

Directions and location
#Customer_Care_Store_Location
3 Responses / 0 Context Set / Skip user input / Returns

Make an appointment
#Customer_Care_Appointments
3 Responses / 7 Context Set / 5 Slots / Does not return

>

Transfer to agent
#General_Connect_to_Agent
1 Responses / 0 Context Set / Does not return

Small Talk
3 Dialog nodes / No digressions

anything_else
1 Responses / 0 Context Set / Returns

2

Add node to folder

Add node above

Add node below 3

Add folder

Move

Duplicate

Jump to

Delete

Name the node "Ask about product" [1] and assign it our new intent [2].

IBM Watson Assistant

Skills /

Customer Care Sample Skill copy
Sample simple customer service skill to get you started.

Intents Entities **Dialog** Analytics Options Versions Content Catalog

1 Ask about product

If assistant recognizes:

2 #Product_Information

Then respond with

Text

Enter response text

Response variations are set to **sequential**. Set to **random**

Add response type

And finally:

This means that if Watson Assistant recognizes a user input such as "how do I set the time?", it will direct the conversation to this node.

Enable webhook from Assistant

Set up access to our WebHook for the IBM Cloud Functions action you created in Step #4.

Select the Options tab [1]:

IBM Watson Assistant

Skills /

Customer Care Sample Skill for Disco

Sample simple customer service skill to get you started.

Options

Webhooks

Autocorrection

System Entities

Webhooks

A webhook is a mechanism that allows your dialog skill to call an external API when specific dialog nodes are triggered. Specify the request URL for the external API you want to be able to invoke. You will then be able to access this URL from within the dialog editor.

[Learn more](#)

URL

2 `https://us-south.functions.cloud.ibm.com/api/v1/web/IBM%20Cloud%20Stor`

Headers

Add HTTP headers for authorization or any other parameters required for invoking the specified request URL.

HEADER NAME	HEADER VALUE
-------------	--------------

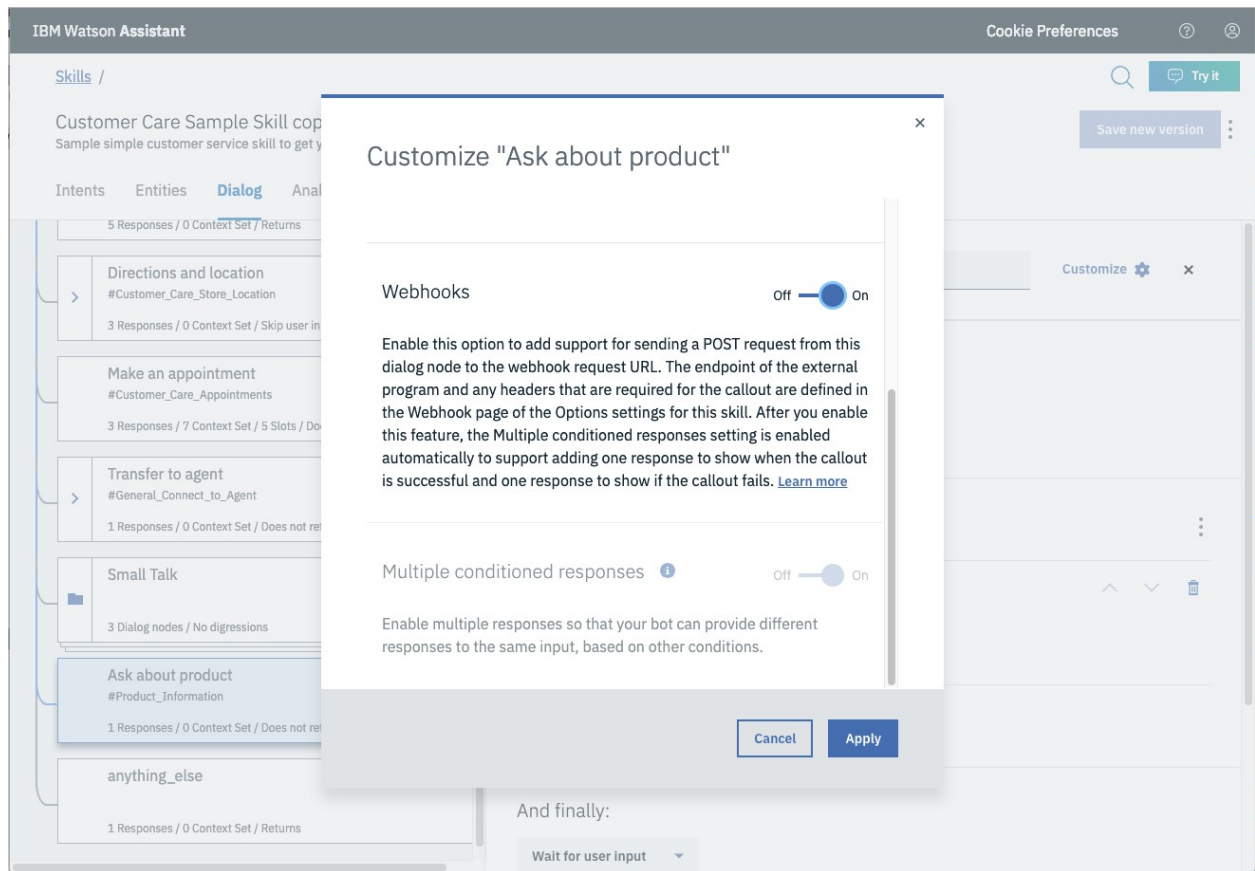
[Add header](#) [Add authorization](#)

Next step

To trigger this webhook from an individual dialog node, enable the webhook from the Customize page in node details. [Go to dialog](#).

Enter the public URL endpoint for your action [2].

Return to the Dialog tab, and click on the Ask about product node. From the details panel for the node, click on Customize, and enable Webhooks for this node:



Click Apply.

The dialog node should have a Return variable [1] set automatically to \$webhook_result_1. This is the variable name you can use to access the result from the Discovery service query.

IBM Watson Assistant

Customer Care Sample Skill for Disco
Sample simple customer service skill to get you started.

Intents Entities **Dialog** Analytics Options Versions Content Catalog

#Customer_Care_Store_Hours
5 Responses / 0 Context Set / Returns

Directions and location
#Customer_Care_Store_Location
3 Responses / 0 Context Set / Skip user input / Returns

Make an appointment
#Customer_Care_Appointments
3 Responses / 7 Context Set / 5 Slots / Does not return

Transfer to agent
#General_Connect_to_Agent
1 Responses / 0 Context Set / Does not return

Small Talk
3 Dialog nodes / No digressions

Ask about product
#Product_Information
2 Responses / 0 Context Set / Does not return

anything_else
1 Responses / 0 Context Set / Returns

Ask about product

If assistant recognizes:

#Product_Information

Then callout to my webhook:

Parameters

KEY	VALUE
2 input	"<?input.text?>"

Add parameter +

Return variable

1 \$webhook_result_1

Then respond with

You will also need to pass in the users question via the parameter input [2]. The key needs to be set to the value: "<?input.text?>"

If you fail to do this, Discovery will return results based on a blank query.

Optionally, you can add these responses to aid in debugging:

Return variable

\$webhook_result_1

Then respond with

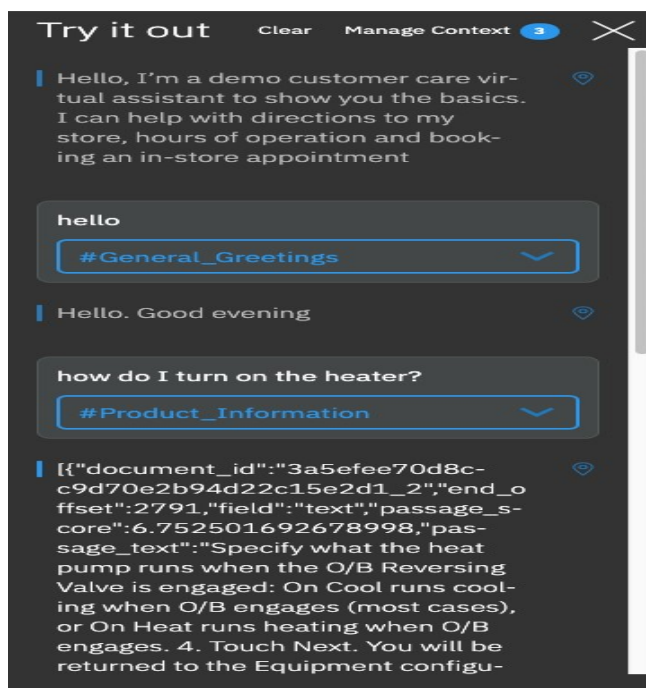
	IF ASSISTANT RECOGNIZES	RESPOND WITH		
1	\$webhook_result_1	\$webhook_result_1	⚙️	🗑️
2	anything_else	Try again later	⚙️	🗑️

[Add response](#) (+)

Test in Assistant Tooling

From the Dialog panel, click the Try it button located at the top right side of the panel.

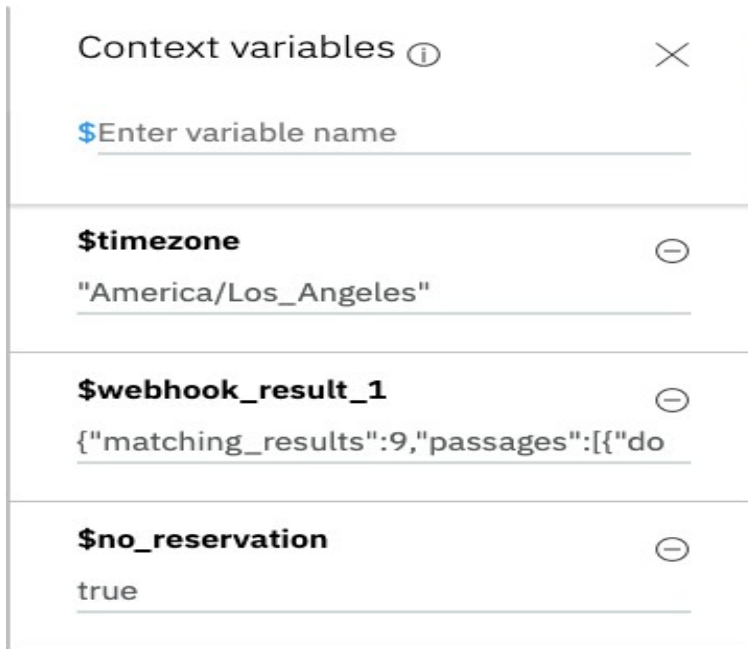
Enter some user input:



Note that the input "how do I turn on the heater?" has triggered our Ask about product dialog node, which is indicated by the #Product_Information response.

And because we specified that \$webhook_result_1.passages be the response, that value is displayed also.

You can also verify that the call was successfully completed by clicking on the Manage Context button at the top right. The response from the Discovery query will be stored in the \$webhook_result_1 variable:



The screenshot shows a 'Context variables' dialog box with a close button (X) in the top right corner. Below the title bar, there is a search input field labeled '\$Enter variable name'. The dialog lists three variables, each with a minus sign icon to its right:

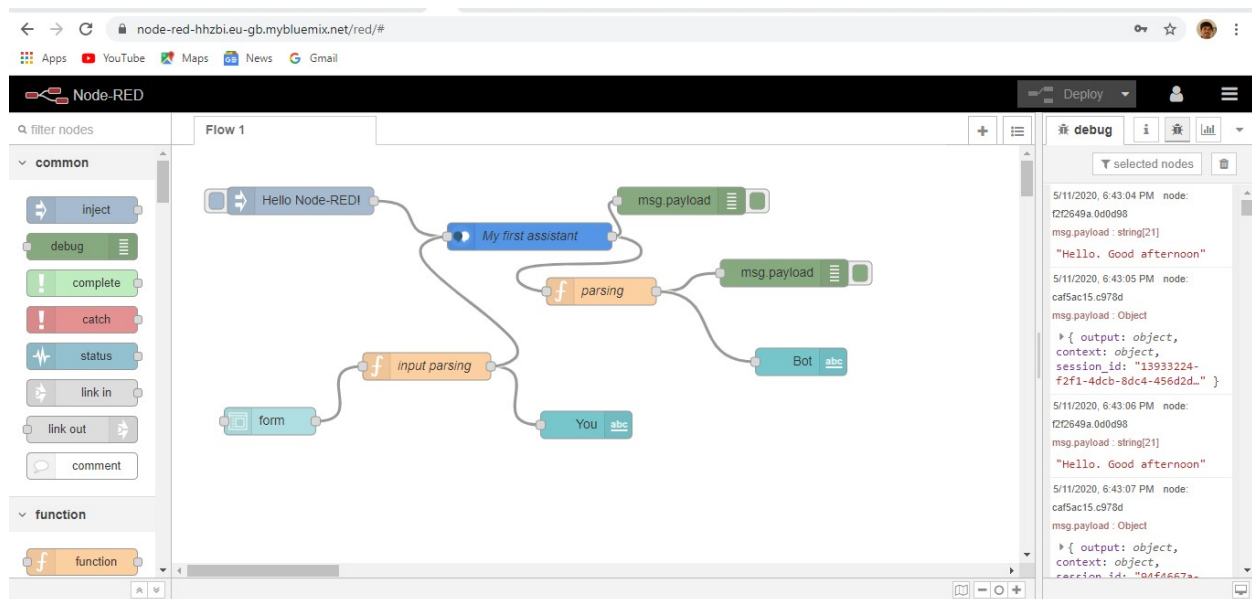
- \$timezone**: "America/Los_Angeles"
- \$webhook_result_1**: {"matching_results":9,"passages":[{"do
- \$no_reservation**: true

5.Create flow and configure node:

At first go to manage pallete and install dashboard.

Now,Create the flow with the help of following node:

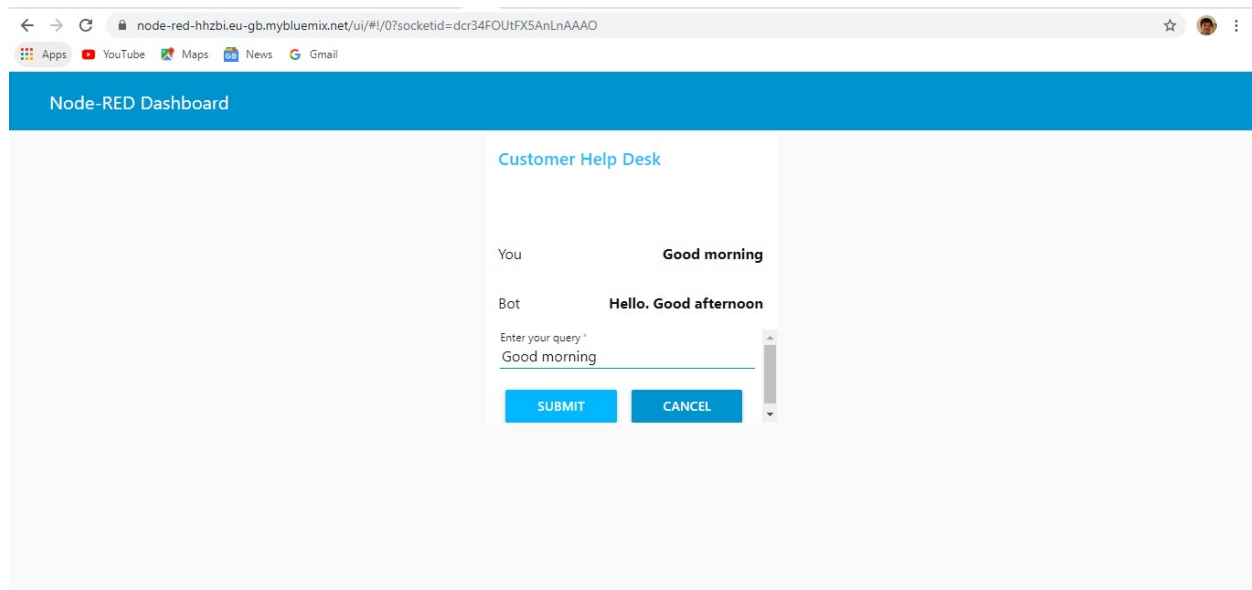
- ✓ Inject
- ✓ Assistant
- ✓ Debug
- ✓ Function
- ✓ Ui_Form
- ✓ Ui_Text



6. Deploy and run Node Red app.

Deploy the Node Red flow.

Then copy the link url upto .net/ and paste at anew tab by ui at the end of the url, like this,
<https://node-red-hhzbi.eu-gb.mybluemix.net/ui>



Reference:

1. https://www.ibm.com/cloud/architecture/tutorials/cognitive_discovery

2. <https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started>
3. <https://developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chatbot-on-nodered/>
4. <http://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red>
5. <https://github.com/IBM/watson-discovery-sdu-with-assistant>
6. <https://www.youtube.com/watch?v=Jpr3wVH3FVA>