

Name : Aanchal Malhotra

Internship Title : Machine Learning Engineer

**Project : Intelligent Customer Help Desk with
Title Smart Document Understanding**

Internship at smartinternz.com @ 2020

1	INTRODUCTION
	1.1 Overview
	1.2 Purpose
2	LITERATURE SURVEY
	2.1 Existing problem
	2.2 Proposed solution
3	THEORETICAL ANALYSIS
	3.1 Block diagram
	3.2 Hardware / Software designing
4	EXPERIMENTAL INVESTIGATIONS
5	FLOWCHART
6	RESULT
7	ADVANTAGES & DISADVANTAGES
8	APPLICATIONS
9	CONCLUSION
10	FUTURE SCOPE
11	BIBLIOGRAPHY
	APPENDIX
	A. Source code

1. INTRODUCTION

1.1 OVERVIEW

Create a Customer Care - Dialog Skill in Watson Assistant using Smart Document Understanding in order to build an enhanced Watson Discovery Collection.

Ultimately integrate the services with a web application and deploy it on IBM Cloud Platform.

- Project Requirements :
A manual document to upload in discovery, Python, IBM Cloud, IBM Watson
- Functional Requirements:
IBM Cloud
- Technical Requirements:
Access to IBM platform , NODE - RED Flow
- Software Requirements:
Watson Assistant, Watson Discovery, Cloud Functions
- Project Deliverable:
The customer- help desk chatbot should be able to provide solutions to the customer queries efficiently.
- Project Team:
1) Aanchal Malhotra
- Project Schedule:
Start Date - 02/05/20
End Date - 17/05/20

1.2 PURPOSE

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 predetermined 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 owner's manual is important and what is not. This will improve the answers returned from the queries.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

Generally Chatbots means getting input from users and getting only response questions and for some questions the output from bot will be like "try again", "I don't understand", "will you repeat again", and so on... and directs customer to customer agent but a good customer Chatbot should minimize involvement of customer agent to chat with customer to clarify his/her doubts. So to achieve this we should include a virtual agent in chatbot so that it will take care of real involvement of the customer agent and customer can clarify his doubts with fast chatbots.

2.2 PROPOSED SOLUTION

In order to minimize the involvement of customer care agents we use Smart Document Understanding (SDU). We feed the manual which is normally used by the agents to answer queries of the customers to Watson discovery, from where the SDU is brought into use.

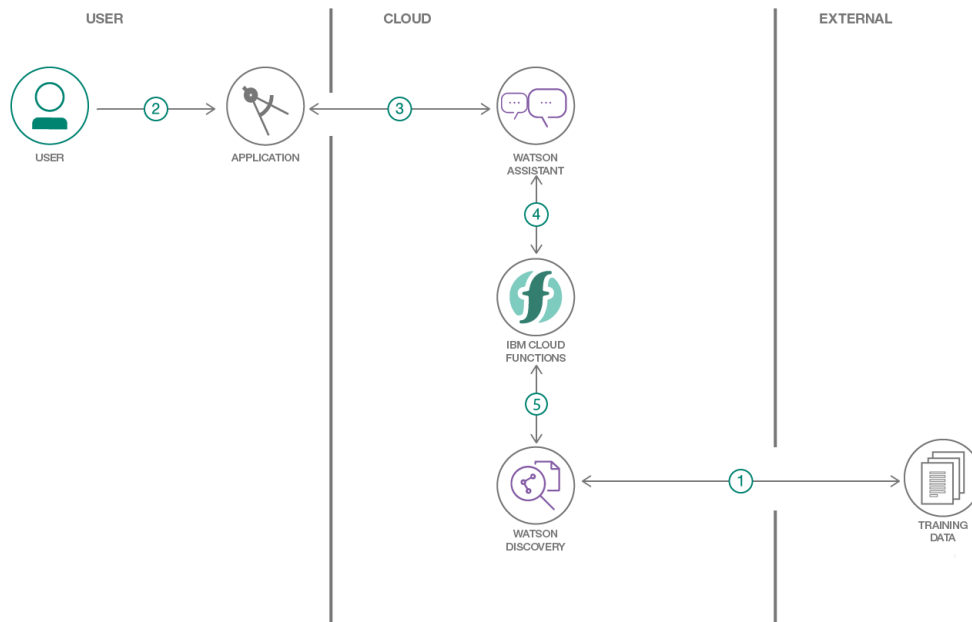
SDU trains Watson Discovery to extract custom fields in your documents. Customizing how your documents are indexed into Discovery will improve the answers returned from queries.

With SDU, you annotate fields within your documents to train custom conversion models. As you annotate, Watson is learning and will start predicting annotations. SDU models can also be exported and used on other collections.

Watson Assistant which acts as the chatbot provided by the IBM platform is connected to the cloud functions which in turn is connected to the Watson Discovery service . All of these instances are integrated in a flow using NODE - RED, where the User interface is created for the chatbot to work.

3. THEORETICAL ANALYSIS

3.1 BLOCK 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

3.2 HARDWARE / SOFTWARE DESIGNING

1. Create IBM account
2. Configure Watson Discovery
3. Configure Watson Assistant
4. Create Cloud Functions action (code)
5. Integrate all instances in NODE - RED flow
6. Create User Interface in NODE - RED flow
7. Deploy the final flow

4. EXPERIMENTAL INVESTIGATIONS

- Create IBM Cloud services

Create the following services:

1. Watson Discovery
2. Watson Assistant
3. Cloud Functions
4. Node Red

1. Watson Discovery

Create a discovery service in dashboard, then launch watson discovery:

The screenshot shows the IBM Cloud Resource list dashboard. The top navigation bar includes the IBM Cloud logo, a search bar, and links to Catalog, Docs, Support, and Manage. The main content area is titled 'Resource list' and features a table with columns for Name, Group, Location, Status, and Tags. The table lists several resources, including VPC infrastructure, Clusters, Cloud Foundry apps, and Services. A 'Create resource' button is visible in the top right corner.

Name	Group	Location	Status	Tags
VPC infrastructure (0)				
Clusters (0)				
Cloud Foundry apps (1)				
Node RED GVLGG	aanchal23m.am@gmail.com / dev	London	Started	—
Cloud Foundry services (1)				
node-red-gvlgg-cloudant-1588876303...	aanchal23m.am@gmail.com / dev	London	Provisioned	—
Services (4)				
Continuous Delivery	Default	London	Active	—
Discovery-s6	Default	London	Active	—
Watson Assistant-6x	Default	London	Active	—
node-red-gvlgg-cloudant-1588876303...	Default	Chennai 01	Active	—

The screenshot shows the IBM Cloud Discovery-s6 service details page. The top navigation bar is the same as the previous screenshot. The main content area is titled 'Discovery-s6' and shows the service is 'Active'. The 'Manage' section on the left includes links for 'Getting started', 'Service credentials', 'Plan', and 'Connections'. The 'Start by launching the tool' section contains a 'Launch Watson Discovery' button, a 'Getting started tutorial' link, and an 'API reference' link. The 'Credentials' section shows the 'API key' and 'URL' fields, with a 'Download' button and a 'Show credentials' link. The 'Plan' section on the right shows the 'Lite' plan and an 'Upgrade' button.

Resource list /
Discovery-s6 Active [Add tags](#)

Manage

- Getting started
- Service credentials
- Plan
- Connections

Start by launching the tool

[Launch Watson Discovery](#) [Getting started tutorial](#) [API reference](#)

Credentials

API key: [Download](#) [Show credentials](#)

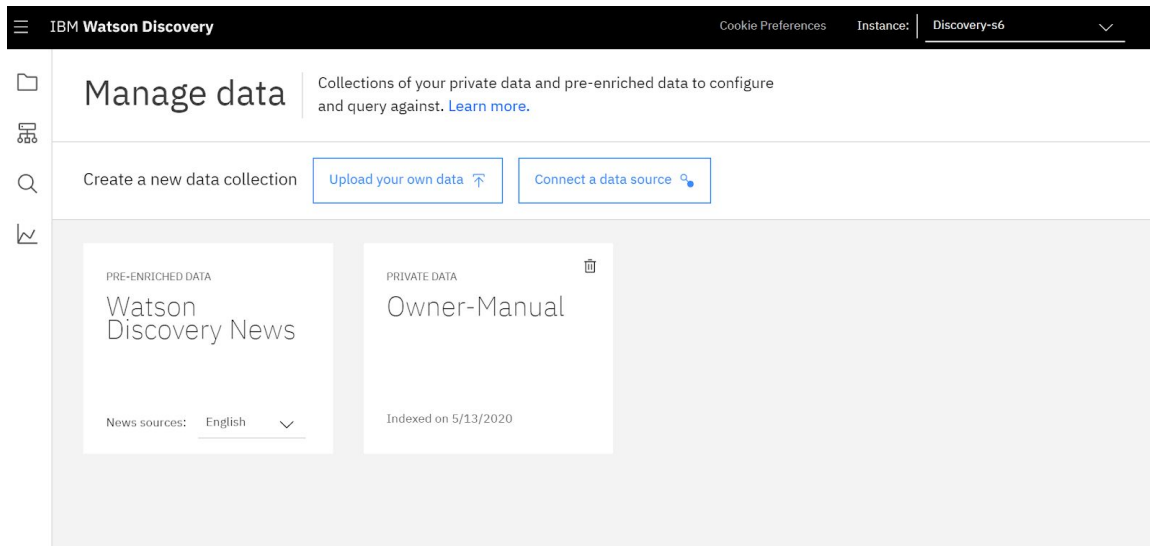
URL: <https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/c1aae7a1-56d4-47f0-acc9-bb198980>

Plan

Lite

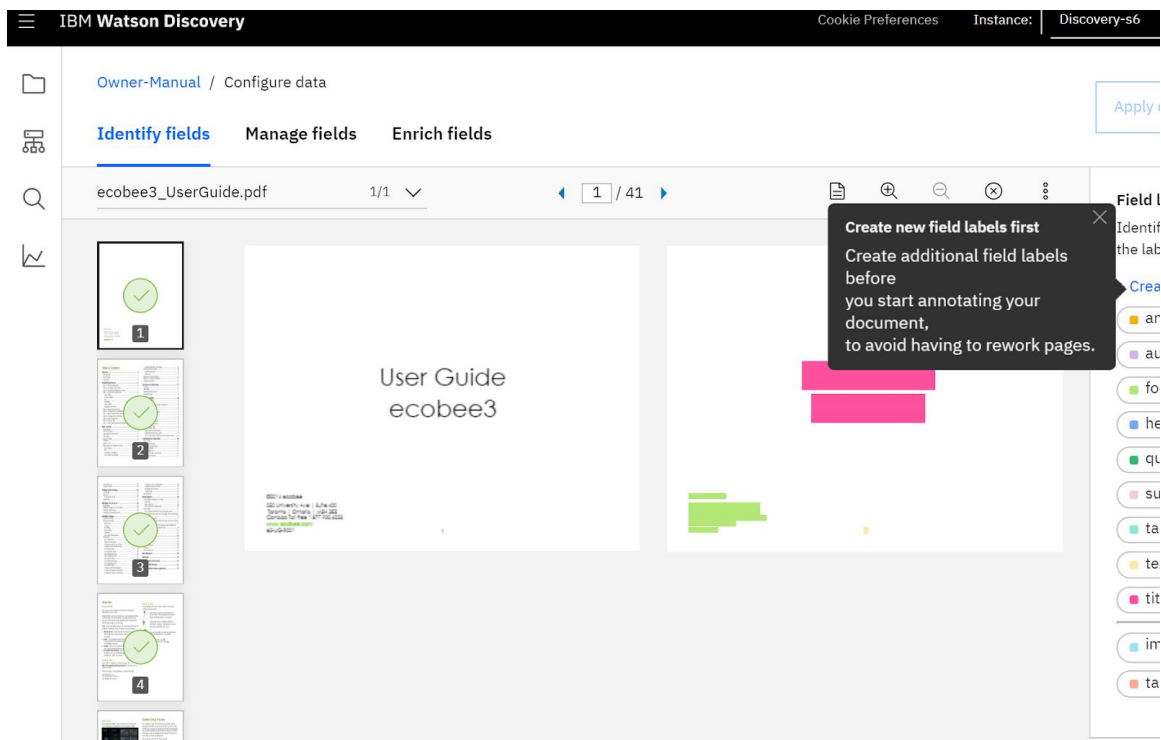
[Upgrade](#)

Upload your own data



Configure data by identifying and managing fields

SDU: Highlight the Title, footer, subtitle, and text and submit the initial pages till eventually watson discovery identifies and highlights the different parts on its own



IBM Watson Discovery

Cookie PreferencesInstance: Discovery-s6

Identify fieldsManage fieldsEnrich fields

Identify fields to index

All fields are indexed by default. Switch off any fields you do not want to be indexed. [Learn more.](#)

answer	Off
author	Off
footer	Off
header	Off
image	Off
question	Off
subtitle	On
table	Off
table_of_contents	Off
text	On
title	On

Improve query results by splitting your documents

You can split your documents into segments based on fields. Once split, each segment is a separate document that will be enriched, indexed, and returned as a query separately. [Learn more.](#)

Split document on each occurrence of

subtitle

IBM Watson Discovery

Cookie PreferencesInstance: Discovery-s6

Owner-Manual

Configure data

OverviewErrors and warnings (127)Search settings

127

documents

0 documents failed

[View details](#)

Created on

5/13/2020 4:18:43 pm EDT

Last updated

5/13/2020 4:18:43 pm EDT

Upload documents

Identified 5 fields from your data

footer

subtitle

table_of_contents

text

title

Need to identify more fields? [Add fields](#)

Added 4 enrichments to your data

Entity Extraction

0.3°C (4) | 0.5°F (4) | 10 °F (4) | 20 min (3) | 4-digit (3)

Sentiment Analysis

54%

33%

13%

positive

neutral

negative

Now you're ready to query!

Documents about 0.3°C as a Quantity with a very negative sentiment

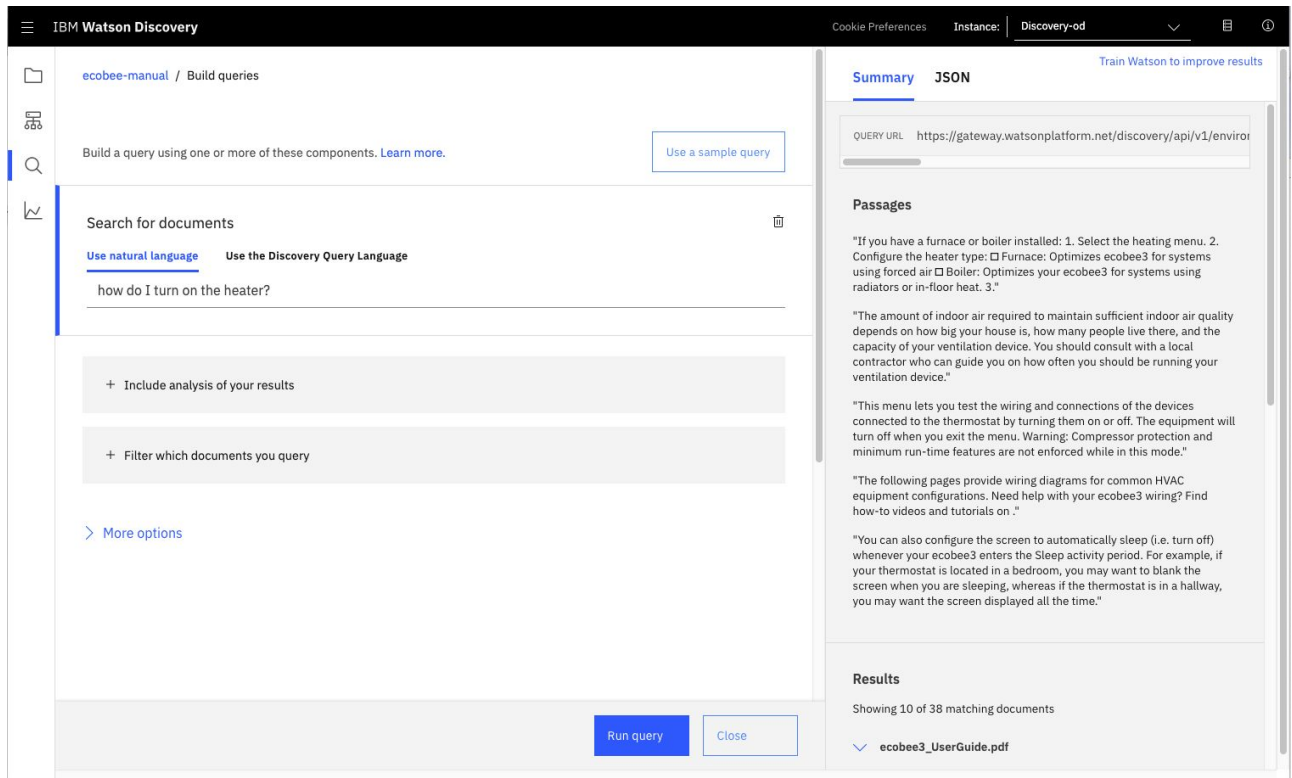
Run

Top entities with their average, min, max sentiment score

Run

Documents that contain Heat, but not HVAC

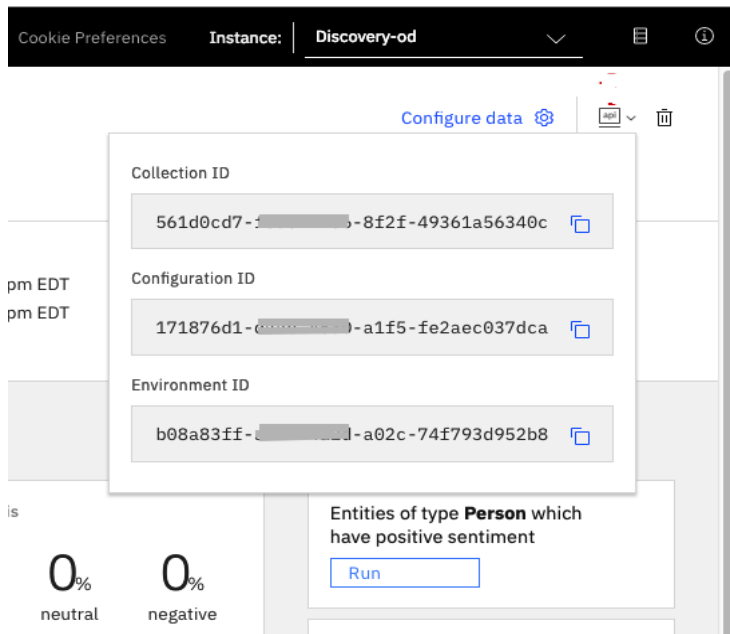
Return to the query panel (click Build your own query) and see how much better the results are.



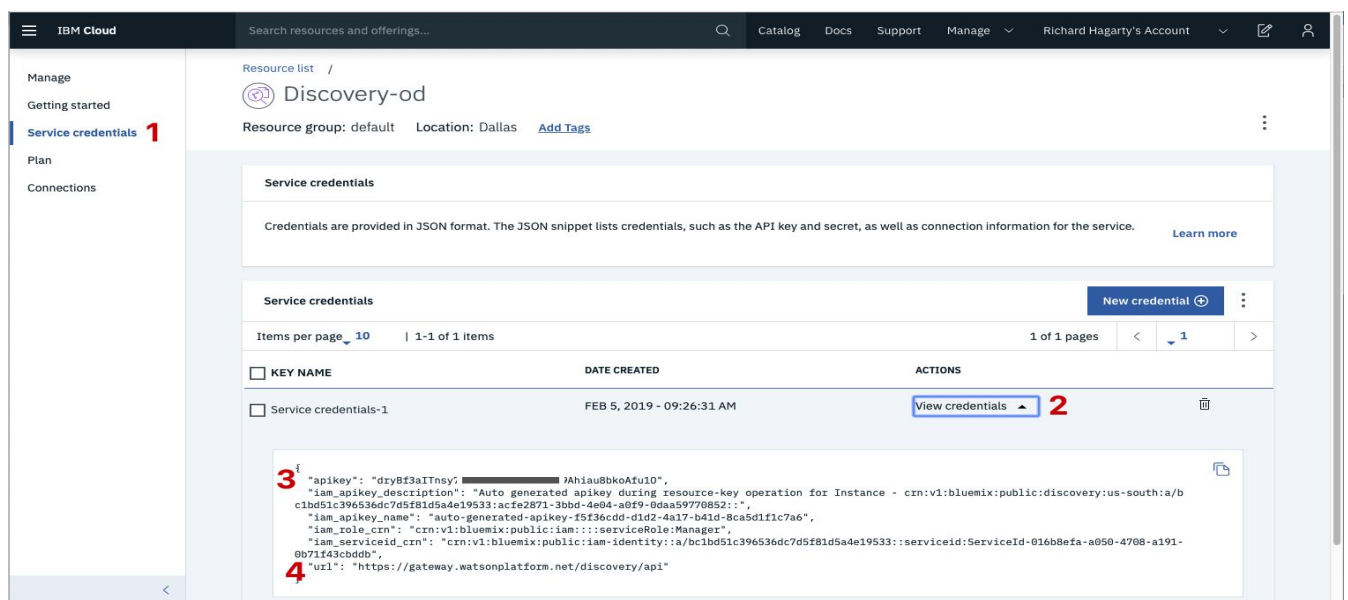
Store credentials for future use

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:



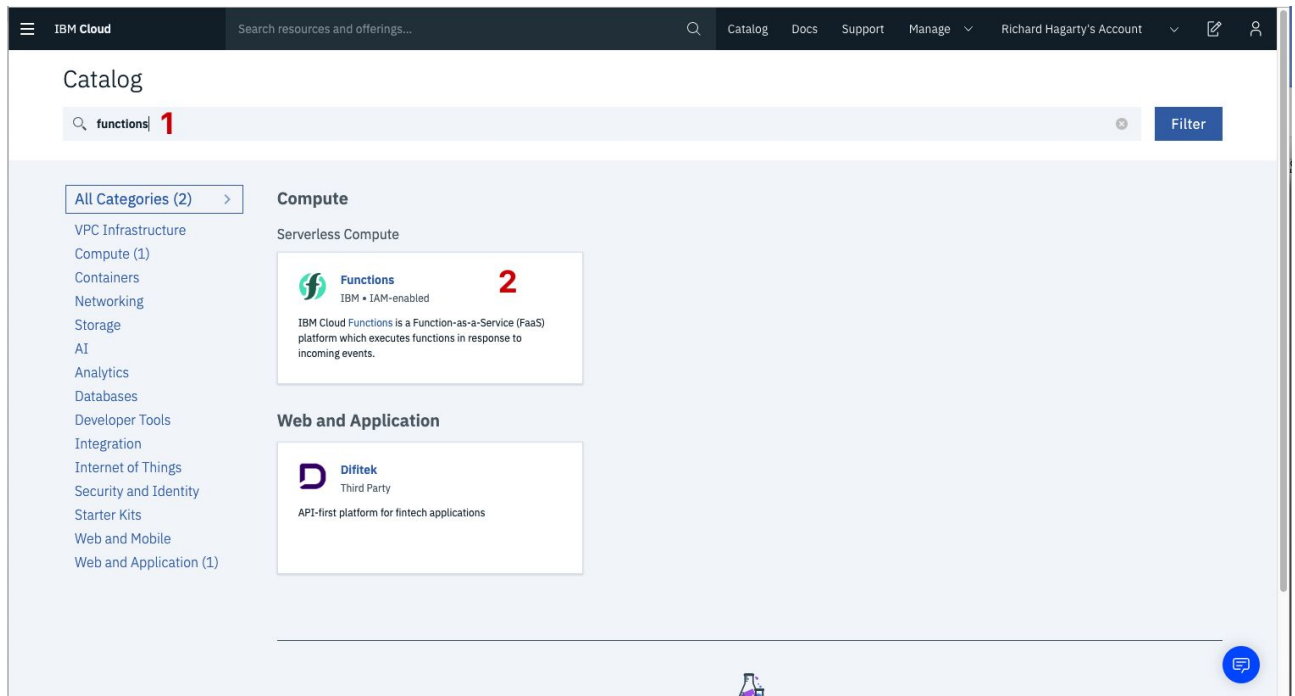
Click the View credentials [2] drop-down menu to view the IAM apikey [3] and URL endpoint [4] for your service.

2. Cloud Functions

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.

IBM Cloud

Search resources and offerings...

Functions

Getting Started

Actions

Triggers

APIs

Monitor

Logs

Namespace Settings

Create Action

Actions contain your function code and are invoked by events or REST API calls.

[Learn more about Actions](#)

[Learn more about Packages](#)

Action Name

disco-action-2

Enclosing Package

(Default Package)

Create Package

Runtime

Node.js 10

Looking for Java, .NET or Docker? [Docker](#) Actions can be created with the [CLI](#)

Cancel

Previous

Create

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

disco-action

Web Action

Code

Parameters

Runtime

Endpoints

Connected Triggers

Enclosing Sequences

Logs

Change Input

Invoke

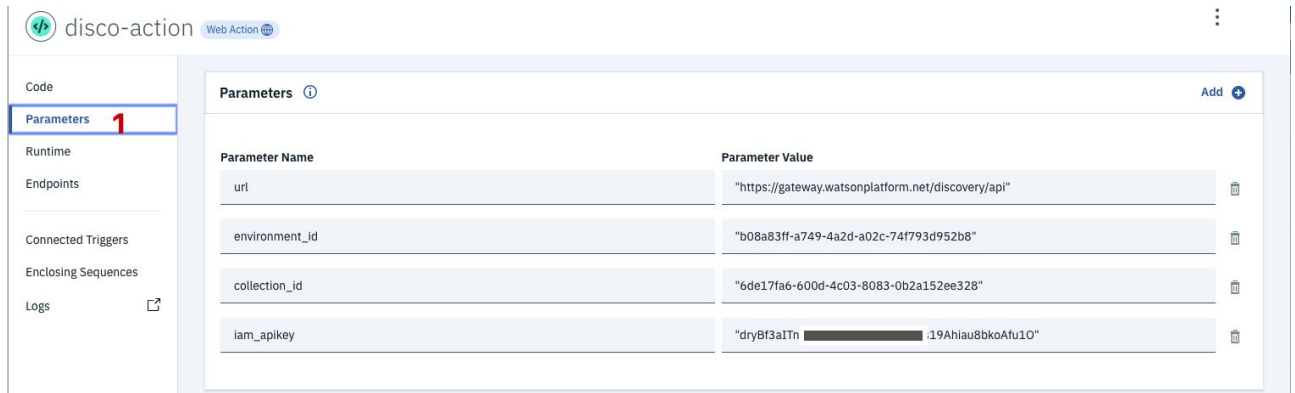
```

1  /**
2   *
3   * @param {object} params
4   * @param {string} params.iam_apikey
5   * @param {string} params.url
6   * @param {string} params.username
7   * @param {string} params.password
8   * @param {string} params.environment_id
9   * @param {string} params.collection_id
10  * @param {string} params.configuration_id
11  * @param {string} params.input
12  *
13  * @return {object}
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,

```

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]:

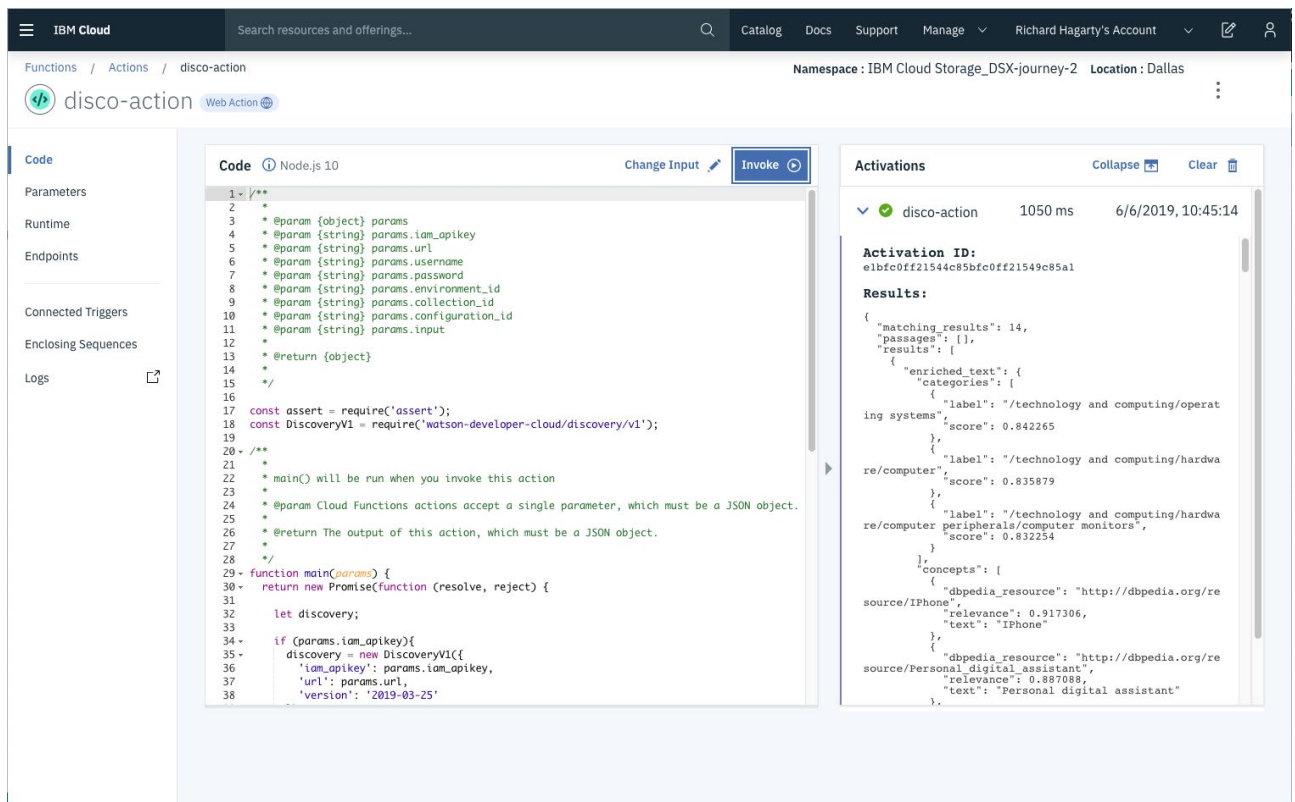


Add the following keys:

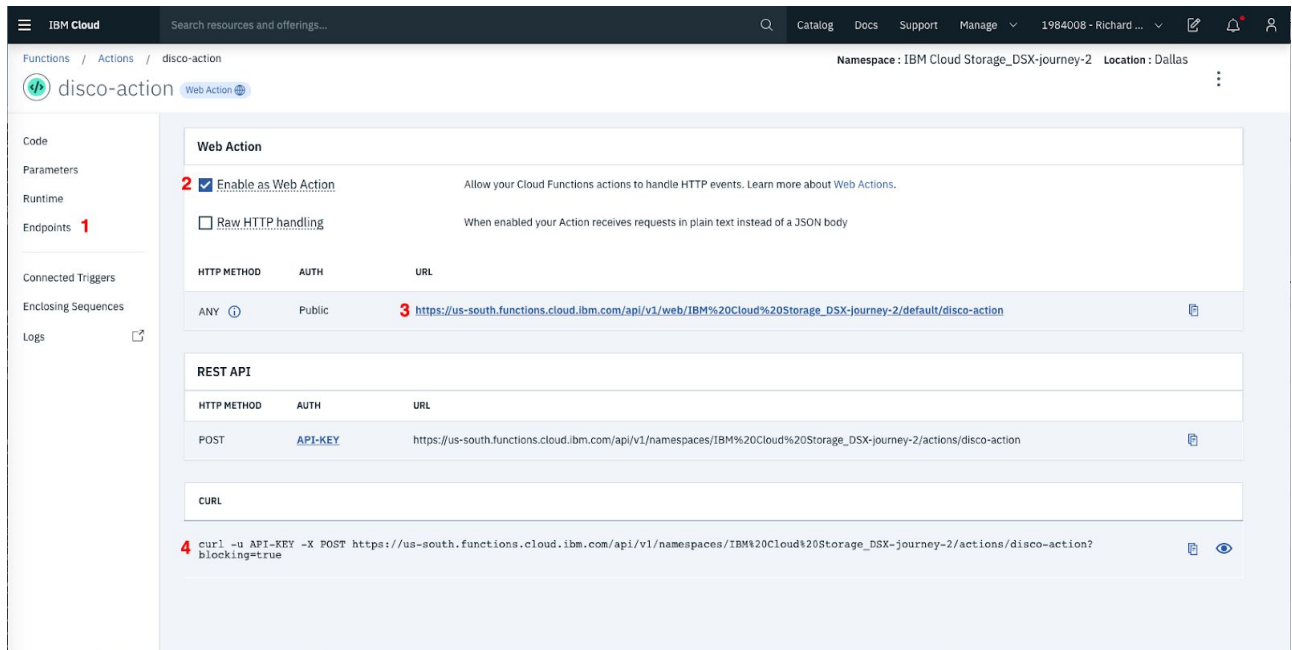
url environment_id collection_id iam_apikey

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:



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.

3. 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.

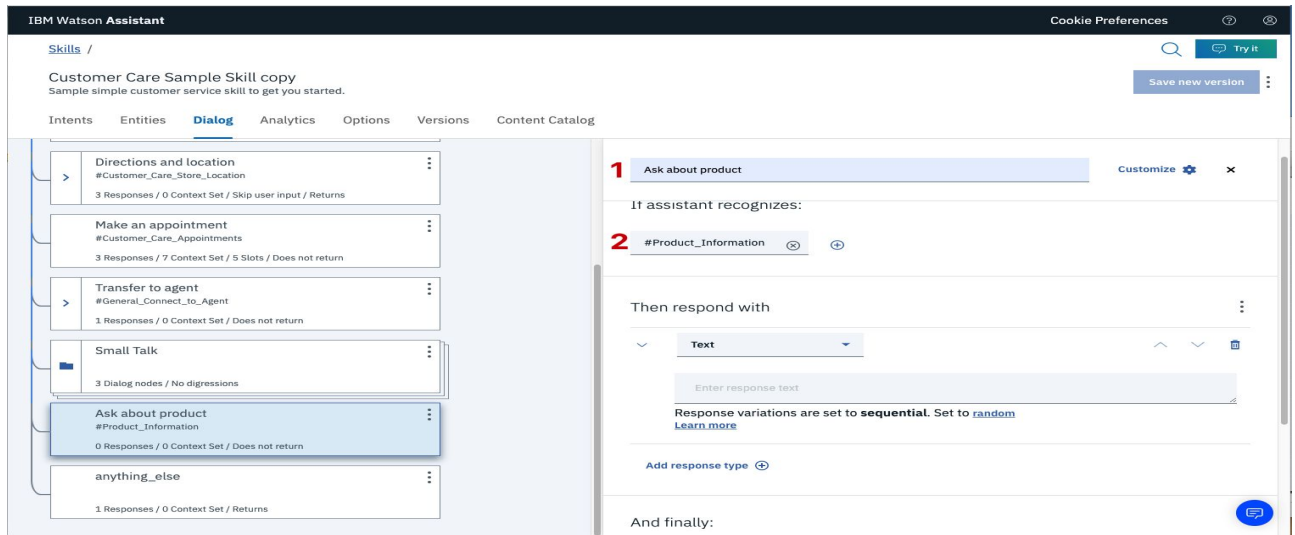
Create a new intent that can detect when the user is asking about operating the Ecobee thermostat.

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.

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

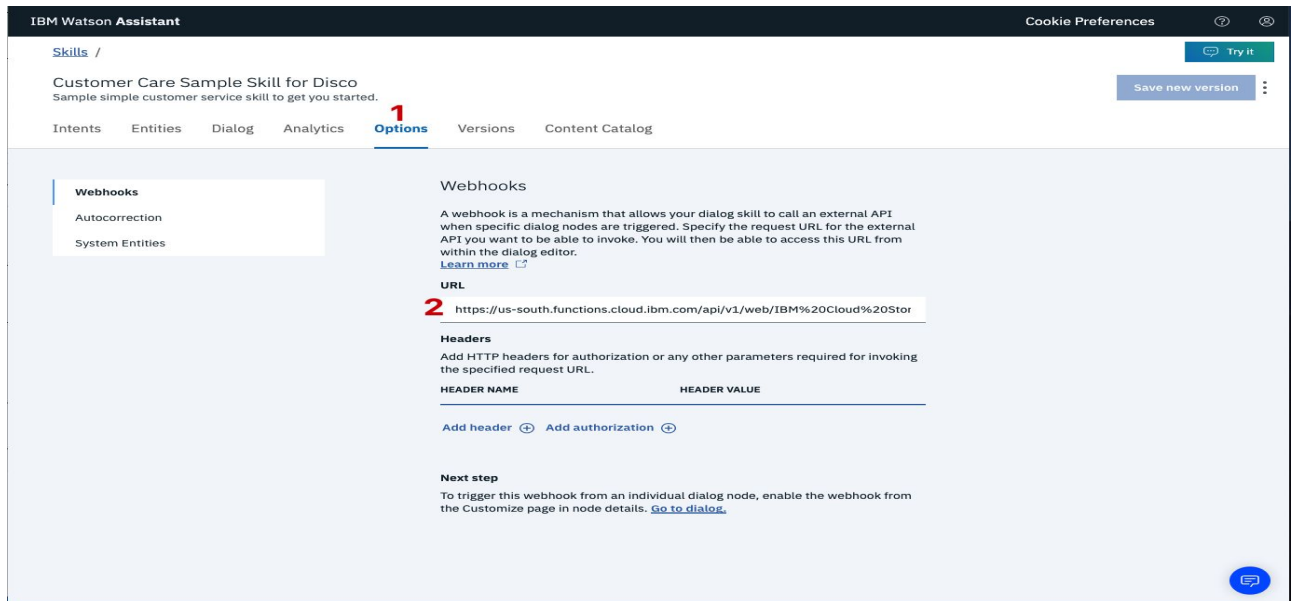


This means that if Watson Assistant recognizes a user input such as "how do I set the time?", it will direct the conversation to this node.

Enable webhook from Assistant

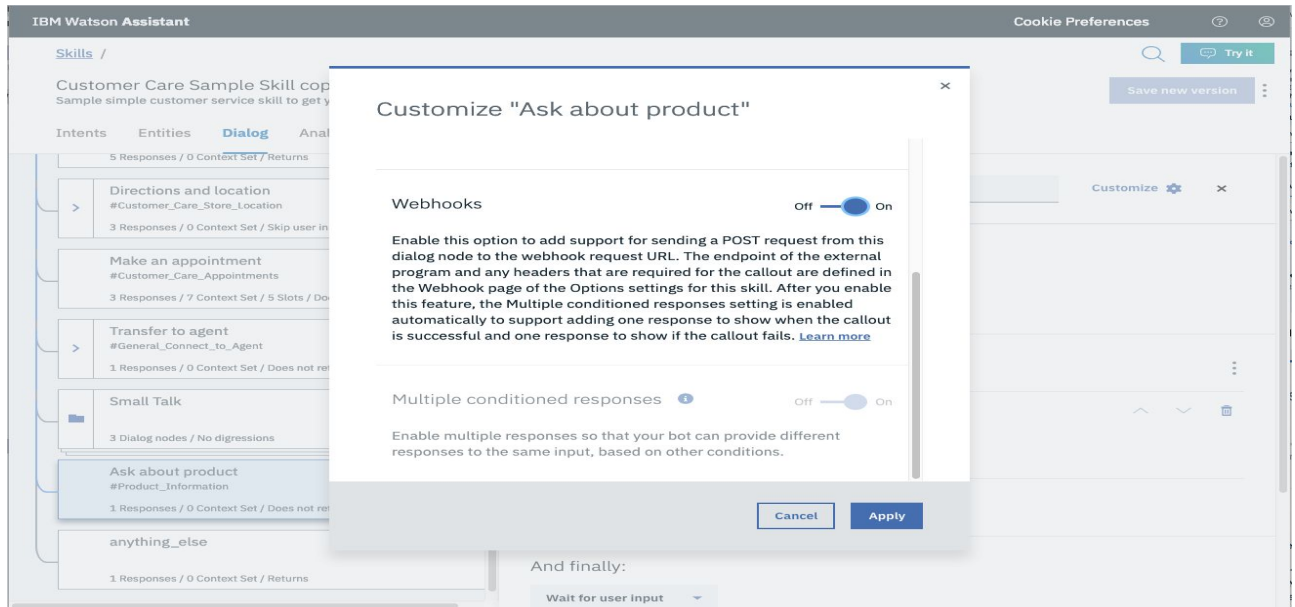
Set up access to our WebHook for the IBM Cloud Functions action you created in Step #4.

Select the Options tab [1]:



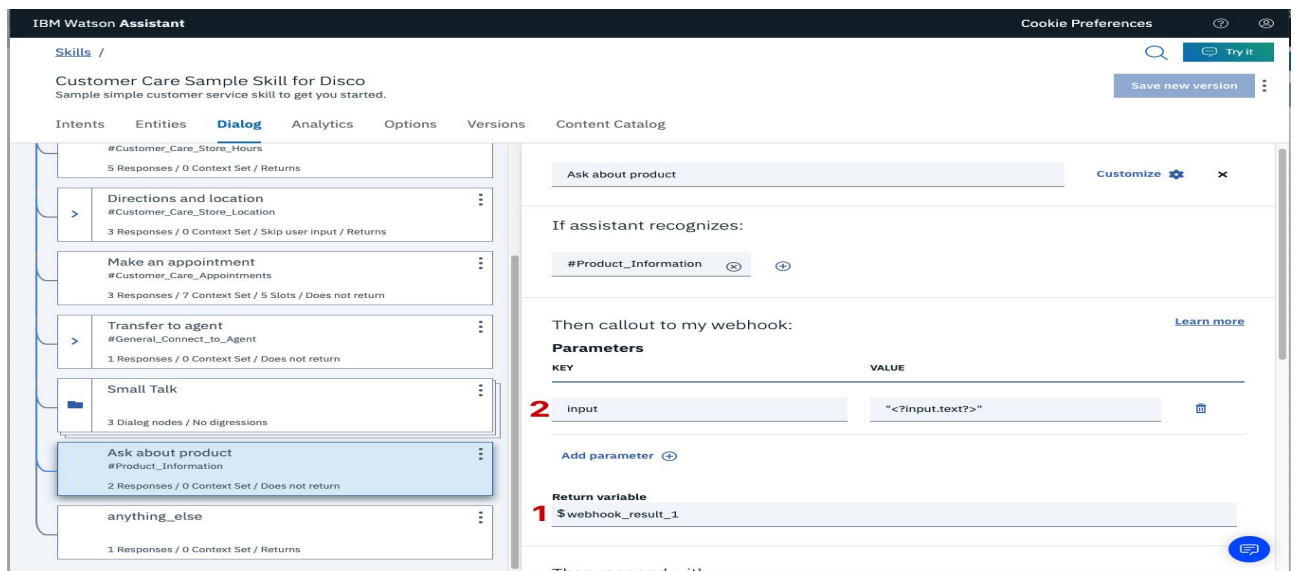
Enter the public URL endpoint for your action [2].

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.



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:

Context variables ⓘ		✕
\$Enter variable name		
\$timezone	"America/Los_Angeles"	⊖
\$webhook_result_1	{"matching_results":9,"passages":[{"do	⊖
\$no_reservation	true	⊖

4. NODE-RED

Enter the API Key, Skill Id and the URL in the watson node

Edit assistant node

Delete Cancel Done

Properties

Name: Name

Username: Username

Password: Password

API Key:

Service Endpoint: https://api.eu-gb.assistant.watson.cloud.ibm.com/i

Workspace ID: afeeb948-eaed-4f88-aed0-26935a34785d

Timeout Period: Leave empty to disable

☐ Save context

☐ Permit Empty Payload

☐ Opt Out Request Logging

☐ Enabled

Install the NODE-RED Dashboard form the manage pallet section

User Settings

Close

View Nodes Install

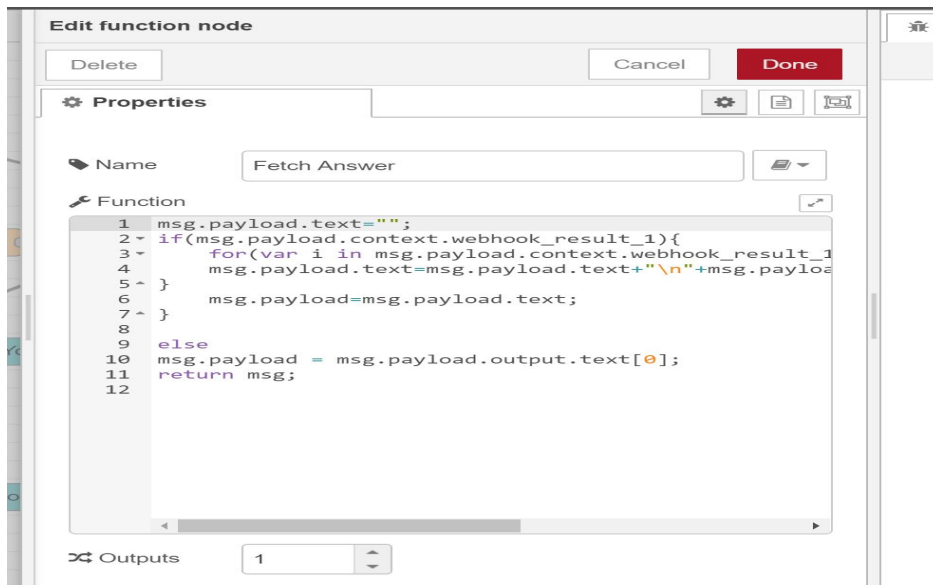
Keyboard

Palette

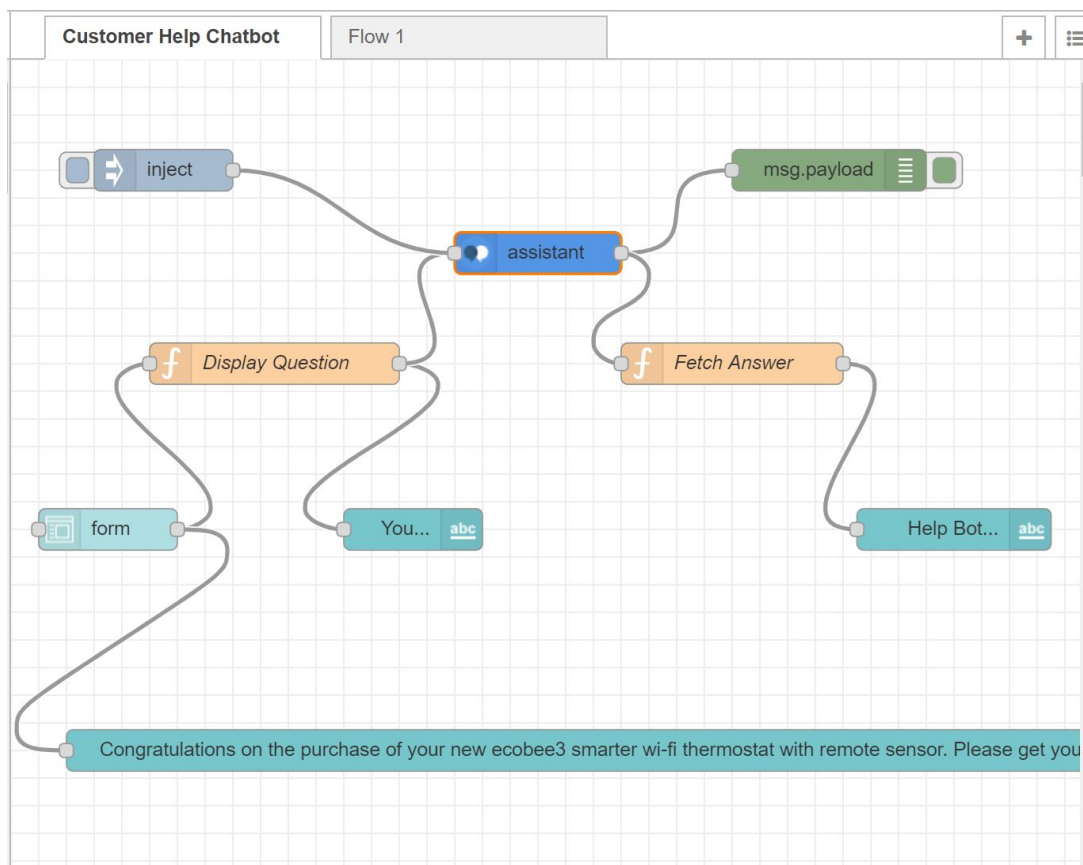
filter nodes

node-red 1.0.6 > 42 nodes	in use
node-red-dashboard 2.22.1 > 21 nodes	in use
node-red-node-cf-cloudant 0.2.17 > 3 nodes	disable all
node-red-node-openwhisk 0.3.4 > 3 nodes	disable all
node-red-node-rbe 0.2.8 > 1 node	disable all
node-red-node-tail 0.1.1	

Enter the code for the function node as follows



Finally configure the whole flow using different nodes like form, text, function and debug.



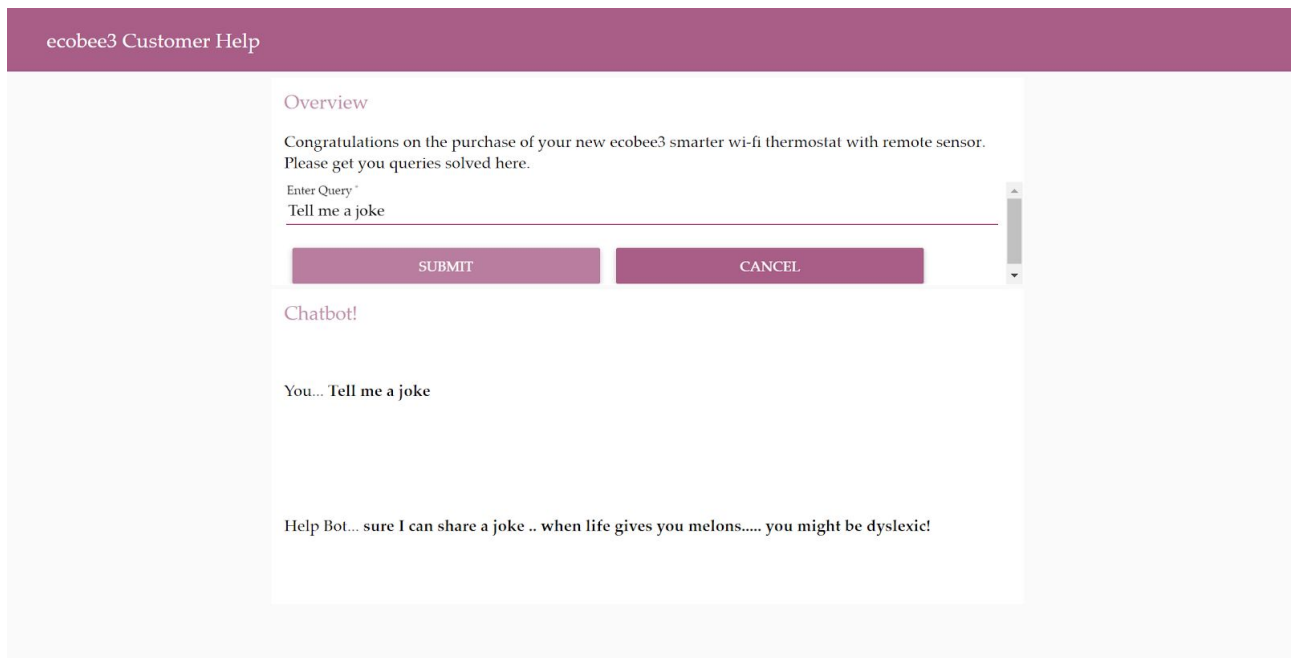
5. FLOW CHART

1. Configure the Watson Discovery Services
2. Configure Watson Assistant Services
3. Configure the Cloud Function
4. Create Actions in Functions
5. Add parameters and connect Assistant and Discovery to it
6. Create NODE-RED Flow
7. Deploy the flow
8. User Interface Chatbot is created successfully!

6. RESULT

NODE-RED Dashboard link after deploying : <https://node-red-gvlgg.eu-gb.mybluemix.net/ui>

The final output for the project:



The screenshot shows a web interface for 'ecobee3 Customer Help'. It features a purple header bar with the text 'ecobee3 Customer Help'. Below the header, there is a white chatbot window. The window has a title 'Overview' and a message: 'Congratulations on the purchase of your new ecobee3 smarter wi-fi thermostat with remote sensor. Please get you queries solved here.' Below this message is a text input field with the placeholder 'Enter Query *' and the text 'Tell me a joke'. There are two buttons, 'SUBMIT' and 'CANCEL', below the input field. Below the input field, there is a section titled 'Chatbot!'. The chatbot's response is: 'You... Tell me a joke' followed by 'Help Bot... sure I can share a joke .. when life gives you melons..... you might be dyslexic!'. The chatbot's response is displayed in a white box with a purple border.

Overview

Congratulations on the purchase of your new ecobee3 smarter wi-fi thermostat with remote sensor. Please get you queries solved here.

Enter Query *

How to turn on the heater

SUBMIT

CANCEL

Chatbot!

You... How to turn on the heater

Your ecobee3 will ask you to select Fahrenheit or Celsius as your preferred temperature units. Touch Next to continue. Sets the percentage of relative indoor humidity at which the ecobee3 will generate a High Humidity Alert. The options are:

- ☐ Off: No alert will be generated. A humidity range of 5% to 95%. You can configure

Help the ecobee3 to only display a specific heat and/or cool set point range. This prevents Bot... users from selecting values outside the displayed range. On Thermostat: 1. Select Main Menu > Settings > Preferences 2. Select Heating range or Cooling range. 3. Adjust the allowed upper and lower values. 4. Touch Save. On Web: 1. Select Settings tile. 2. Select Preferences. 3. Select Heat Set Point Range or Cool Set Point Range. 4. Set the allowed upper and lower values by sliding the values left or right.

7 .ADVANTAGES & DISADVANTAGES

Advantages:

- Saves time for the customer
- Reduces man power
- Cost efficient
- No need to divert calls to customer agents

Disadvantages:

- Chatbot can be misleading sometimes
- May not be able to provide accurate solutions if not configured properly
- Sometimes chatbot can fail to understand customer sentiments and intentions

8.APPLICATIONS

- It can be integrated in popular social media applications like facebook , slack, telegram.
- Chatbot can act as a medium to have casual conversation
- Any website can make use of it to provide solutions to their clients without the involvement of a third party

9.CONCLUSION

This project is a great way to begin learning about the services provided by the IBM platform; using these services we get to understand and implement the concept of Smart Document Understanding which ultimately helps us build an Intelligent Customer Help-desk Chatbot.

10.FUTURE SCOPE

We can include watson studio text to speech and speech to text services to access the chatbot handsfree. This is one of the future scopes for this project.

11. BIBLIOGRAPHY

APPENDIX

A. Source Code

- Cloud Function (cloudfunction.js)

```
/**  
  
 *  
 * @param {object} params  
 * @param {string} params.iam_apikey  
 * @param {string} params.url  
 * @param {string} params.username  
 * @param {string} params.password  
 * @param {string} params.environment_id
```



```

* @param {string} params.collection_id
* @param {string} params.configuration_id
* @param {string} params.input
*
* @return {object}
*
*/

const assert = require('assert');
const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');

/**
 *
 * main() will be run when you invoke this action
 *
 * @param Cloud Functions actions accept a single parameter, which must be a
JSON object.
 *
 * @return The output of this action, which must be a JSON object.
 *
*/

function main(params) {
  return new Promise(function (resolve, reject) {

    let discovery;

    if (params.iam_apikey){
      discovery = new DiscoveryV1({
        'iam_apikey': params.iam_apikey,

```

```

        'url': params.url,
        'version': '2019-03-25'
    });
}
else {
    discovery = new DiscoveryV1({
        'username': params.username,
        'password': params.password,
        'url': params.url,
        'version': '2019-03-25'
    });

    discovery.query({
        'environment_id': params.environment_id,
        'collection_id': params.collection_id,
        'natural_language_query': params.input,
        'passages': true,
        'count': 3,
        'passages_count': 3
    }, function(err, data) {
        if (err) {
            return reject(err);
        }
        return resolve(data);
    });
});
}

```

- NODE-RED (flows.json)

```
[
  {
    "id": "428c869.9d44078",
    "type": "tab",
    "label": "Customer Help Chatbot",
    "disabled": false,
    "info": ""
  },
  {
    "id": "f2f2649a.0d0d98",
    "type": "debug",
    "z": "428c869.9d44078",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "x": 590,
    "y": 80,
    "wires": []
  },
  {
    "id": "33ee0287.4b505e",
    "type": "ui_form",
    "z": "428c869.9d44078",
    "name": "",
    "label": "",
    "group": "b25b0315.a262e",
    "order": 2,
    "width": 0,
    "height": 0,
    "options": [
      {
```

```

        "label": "Enter Query",
        "value": "question",
        "type": "text",
        "required": true,
        "rows": null
    }
],
"formValue": {
    "question": ""
},
"payload": "",
"submit": "submit",
"cancel": "cancel",
"topic": "",
"x": 70,
"y": 340,
"wires": [
    [
        "9af5382.c4eebc8",
        "8b3d873.c439178"
    ]
]
},
{
    "id": "8b3d873.c439178",
    "type": "ui_text",
    "z": "428c869.9d44078",
    "group": "b25b0315.a262e",
    "order": 1,
    "width": 0,
    "height": 0,
    "name": "",
    "label": "Congratulations on the purchase of your new ecobee3 smarter wi-fi thermostat with remote sensor. Please get you queries solved here.",
    "format": "",
    "layout": "row-spread",
    "x": 490,
    "y": 500,
    "wires": []
}

```

```
},
{
  "id": "9af5382.c4eebc8",
  "type": "function",
  "z": "428c869.9d44078",
  "name": "Display Question",
  "func": "msg.payload = msg.payload.question;\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "x": 190,
  "y": 220,
  "wires": [
    [
      "e72c71f3.cda0f",
      "af6fd9ae.7ba1a8"
    ]
  ]
},
{
  "id": "e72c71f3.cda0f",
  "type": "ui_text",
  "z": "428c869.9d44078",
  "group": "69652ad6.2f3994",
  "order": 1,
  "width": 8,
  "height": 2,
  "name": "",
  "label": "You...",
  "format": "{{msg.payload}}",
  "layout": "row-left",
  "x": 290,
  "y": 340,
  "wires": []
},
{
  "id": "dbbd0f70.3bd49",
  "type": "function",
  "z": "428c869.9d44078",
  "name": "Fetch Answer",
```

```

    "func": "msg.payload.text=\"\\\";\\nif(msg.payload.context.webhook_result_1){\\n  for(var i in
msg.payload.context.webhook_result_1.results){\\n
msg.payload.text=msg.payload.text+\"\\\"\\n\\\"+msg.payload.context.webhook_result_1.results[i].text;\\n}\\n
n  msg.payload=msg.payload.text;\\n}\\n\\nelse\\nmsg.payload = msg.payload.output.text[0];\\nreturn
msg;\\n ",
    "outputs": 1,
    "noerr": 0,
    "x": 520,
    "y": 220,
    "wires": [
      [
        "4c61e88b.977b58"
      ]
    ]
  },
  {
    "id": "af6fd9ae.7ba1a8",
    "type": "watson-conversation-v1",
    "z": "428c869.9d44078",
    "name": "",
    "workspaceid": "afeeb948-caed-4f88-aed0-26935a34785d",
    "multiuser": false,
    "context": false,
    "empty-payload": false,
    "service-endpoint":
"https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/6e2e47d9-54f8-4a84-8f1b-e9c533c51e34",
    "timeout": "",
    "optout-learning": false,
    "x": 380,
    "y": 140,
    "wires": [
      [
        "f2f2649a.0d0d98",
        "dbbd0f70.3bd49"
      ]
    ]
  },
  {
    {
      "id": "4c61e88b.977b58",

```

```
"type": "ui_text",
"z": "428c869.9d44078",
"group": "69652ad6.2f3994",
"order": 4,
"width": 13,
"height": 3,
"name": "",
"label": "Help Bot...",
"format": "{{msg.payload}}",
"layout": "row-left",
"x": 680,
"y": 340,
"wires": []
```

```
},
```

```
{
  "id": "75401475.72a5fc",
  "type": "inject",
  "z": "428c869.9d44078",
  "name": "",
  "topic": "",
  "payload": "",
  "payloadType": "str",
  "repeat": "",
  "crontab": "",
  "once": false,
  "onceDelay": 0.1,
  "x": 110,
  "y": 80,
  "wires": [
    [
      "af6fd9ae.7ba1a8"
    ]
  ]
}
```

```
},
```

```
{
  "id": "b25b0315.a262e",
  "type": "ui_group",
  "z": "",
  "name": "Overview",
```

```
    "tab": "cb026f97.c6b13",
    "order": 1,
    "disp": true,
    "width": "14",
    "collapse": false
  },
  {
    "id": "69652ad6.2f3994",
    "type": "ui_group",
    "z": "",
    "name": "Chatbot!",
    "tab": "cb026f97.c6b13",
    "order": 2,
    "disp": true,
    "width": "14",
    "collapse": false
  },
  {
    "id": "cb026f97.c6b13",
    "type": "ui_tab",
    "z": "",
    "name": "ecobee3 Customer Help",
    "icon": "dashboard",
    "disabled": false,
    "hidden": false
  }
]
```