Project Report

Name:K.L.V.*SOWMYA SRI*(sowmyakattamuri61@gmail.com)

Title:*Intelligent Customer Help Desk With SmartDocument Understanding*

Category:*Artificial Intelligence*

1. INTRODUCTION

1.1 Overview: We will be able to write an application that leverages multiple WatsonAI

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, IBMWatson
- **Functional Requirements**: IBMcloud
- **Technical Requirements**: AI,ML,WATSONAI,PYTHON
- **Software Requirements**: Watson assistant, Watsondiscovery.
- **Project Deliverables**: SmartinternzIntership
- **Project Team**: SOWMYA SRI
- **Project Duration**:19days

1.2 Purpose:

Thetypical customercarechatbot can answersimple questions,suchas storelocations and hours, directions, and maybe even makingappointments.Whena question fallsoutsideof 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 arealperson.

In thisproject,there will be another option. If thecustomer questionis aboutthe operationof a device, theapplication shallpass the question ontoWatsonDiscovery Service, which has beenpre-loadedwiththedevice's ownersmanual.So now, instead of "Would you like to speak to a customer representative?" we canreturnrelevant sections of the ownersmanualto help solve ourcustomers'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.

1.2.1 Scope of Work

- ✓ Create acustomercare dialog skill inWatsonAssistant

- ✓ Use Smart DocumentUnderstandingto buildanenhancedWatsonDiscoverycollection
- ✓ Create anIBMCloud Functions webactionthat allowsWatsonAssistant to postqueriestoWatsonDiscovery
- ✓ Build a webapplicationwithintegrationto alltheseservices &deploy thesame onIBMCloudPlatform

2. LITERATURESURVEY

2.1 Existingproblem:

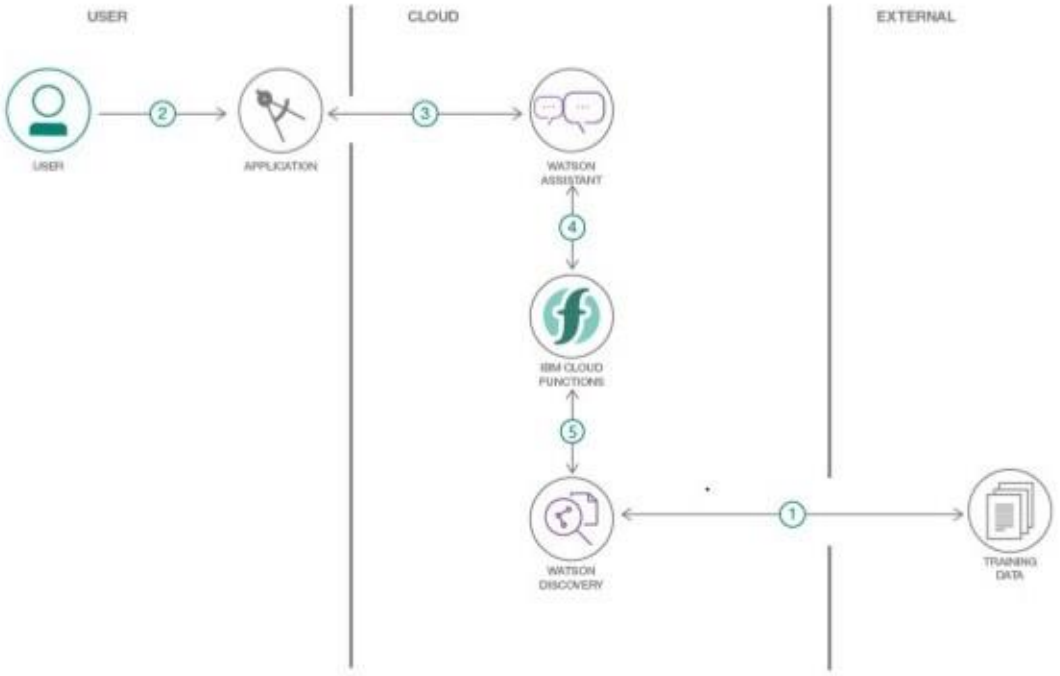
Generally Chatbots means getting input from users and getting only response questions and for some questions the output from bot will be like “try again”, “I don’t understand”, “will you repeat again”, and so on… and directs customer to customer agent but a good customer Chatbot should minimize involvement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

2.2 Proposedsolution:

For the above problem to get solved we have to put an virtual agent in chatbot so it can understand the queries that are posted by customers. The virtual agent should trained from some insight records based company background so it can answer queries based on the product or related to company. In this project I used Watson Discovery to achieve the above solution. And later including Assistant and Discovery on Node-RED

3. THEORITICALANALYSIS

3.1 Block/FlowDiagram



1. ThedocumentisannotatedusingWatsonDiscoveryS D U
2. The user interacts with the backend server viatheapp UI. Thefrontendapp UI is a chatbot thatengagesthe user in aconversation.
3. Dialog between theuserandbackendserver iscoordinatedusing aWatsonAssistant dialogskill.
4. If the userasksa product operation question, asearch queryis passed to apredefined IBMCloud Functionsaction.
5. The CloudFunctionsaction willquerytheWatson Discoveryservice and return the results.

3.2 Hardware / Softwaredesigning:

1. CreateIBMCloudservices
2. Configure WatsonDiscovery
3. CreateIBMCloudFunctionsaction
4. Configure WatsonAssistant
5. Create flowand configuren o d e
6. Deployand run Node Red a p p .

4.EXPERIMENTAL INVESTIGATIONS

1.Create IBM Cloudservices

Create the following services:

- WatsonDiscovery
- WatsonAssistant
- NodeRed

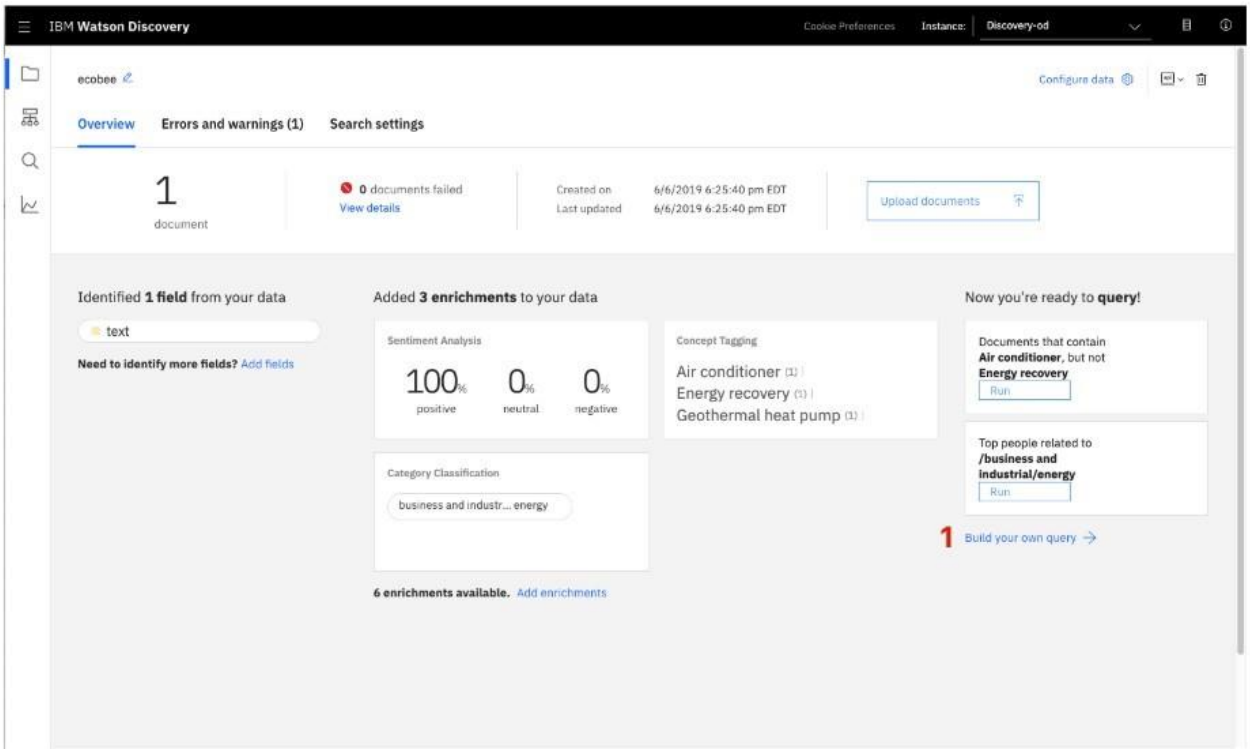
2. ConfigureWatsonDiscovery

Import the document

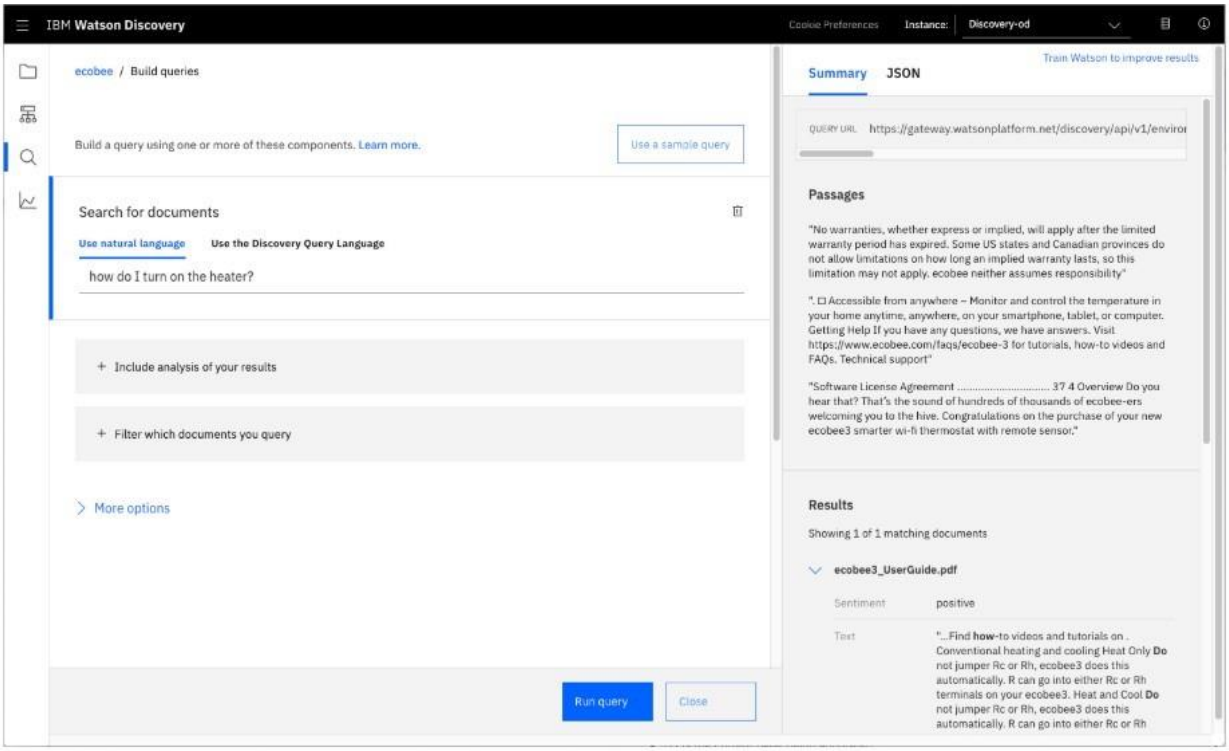
Launch the Watson Discovery tool and create a new data collection by selecting the Upload your own data option. Give the data collection a unique name. When prompted, select and upload the ecobee3_UserGuide.pdf file located in the data directory of your local repo.

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

Beforeapplying SDU to our document, lets dosomesimplequeries onthe data so that we can compare it toresultsfoundafterapplyingS D U .



Click the Build your own query [1] button.

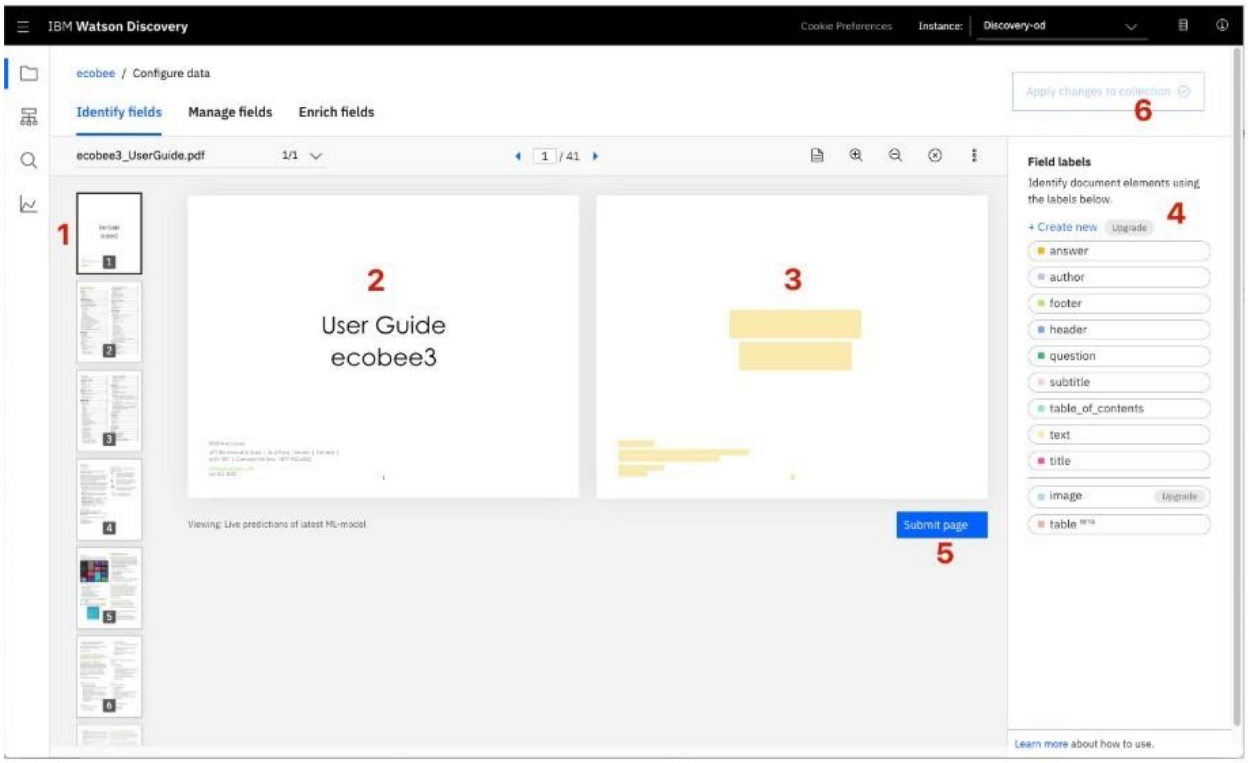


Enter queries related to the operation of the thermostat and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question.

Annotate with SDU

Nowlet'sapply SDU to our document to see if we cangeneratesome betterquery responses.FromtheDiscovery collection panel,click theConfiguredata button (located in the top right corner) to start the SDUp r o c e s s .

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 ofpagesin the manual. As each is processed, agreen checkmark will appear on thepage.

[2] is the currentpagebeinga n n o t a t e d .

[3] is where you select text andassignit al a b e l .

[4] is the list of labels you canassignto thepage text.

Click [5] to submit the page to Discovery.

Click [6] when you havecompletedtheannotationp r o c e s s .

As yougo thoughthe annotations one page at a time, Discovery islearningand should startautomatically updatingthe upcomingpages.Once you get to a page that is alreadycorrectlyannotated, you canstop,or simply click Submit [5] toacknowledgeit iscorrect. Themorepagesyouannotate,the better the model will bet r a i n e d .

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

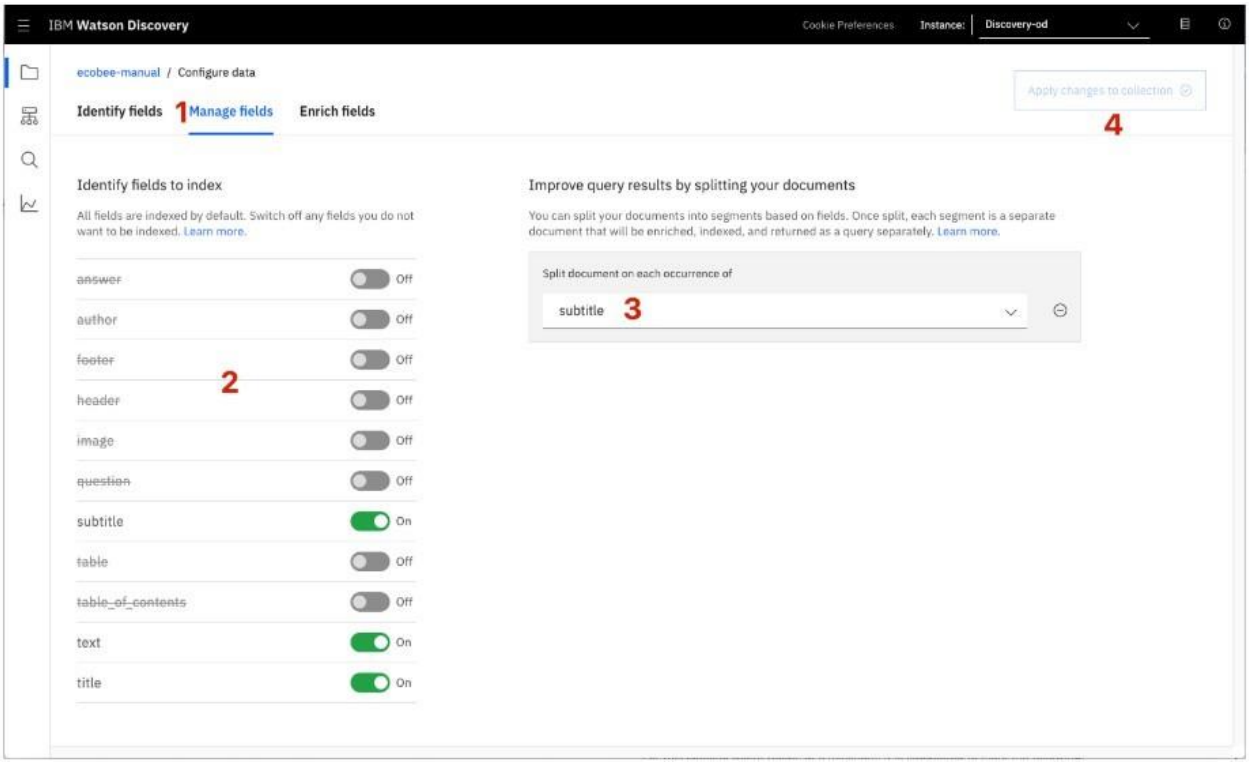
All page numbers as footers

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

All other text should be marked as text.

Once you click the Applychangesto collection button [6], you will beaskedto reload thedocument. Choosethe same owner's manual.pdfdocument asbefore.

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



[2] Here is where youtellDiscoverywhichfieldstoignore.Using the on/offbuttons,turn off all labels exceptsubtitlesandtext.

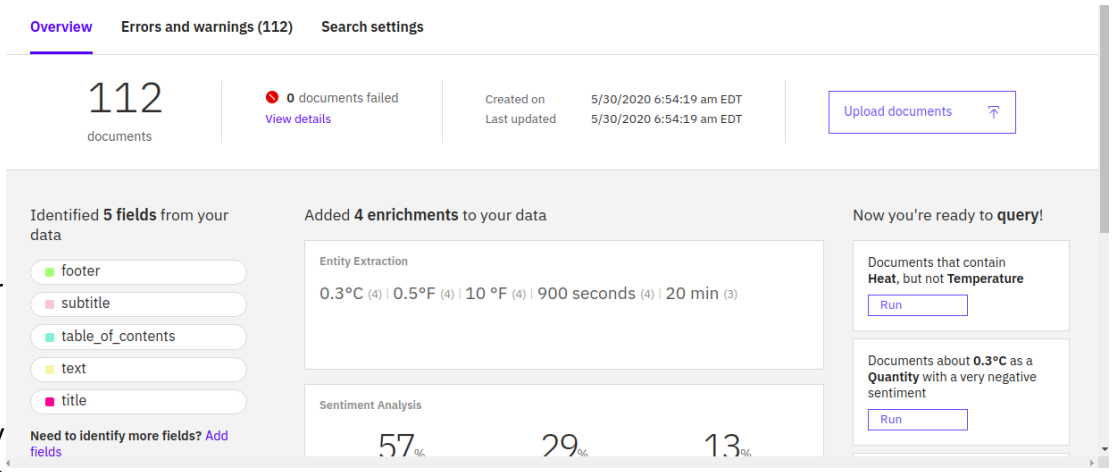
[3] is tellingDiscoveryto split the documentapart,based onsubtitle.

Click [4] to submit your changes.

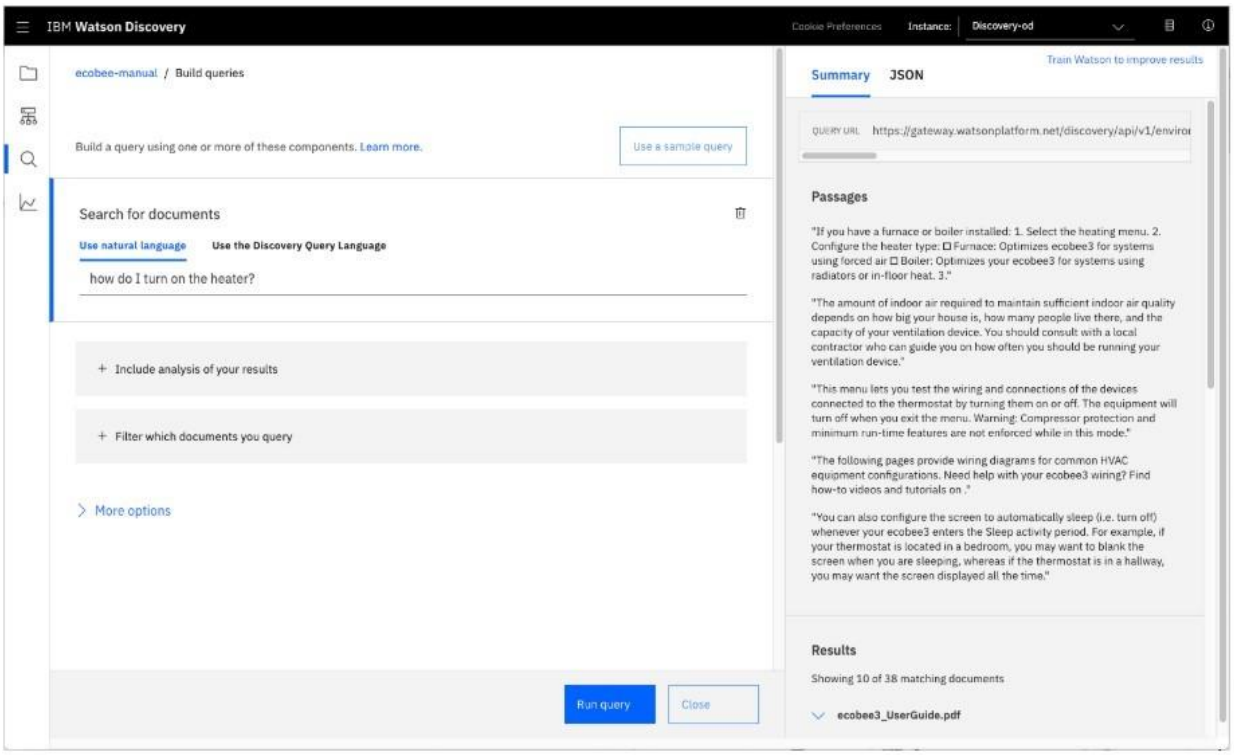
Once again, you willbe askedto reload thedocument.

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

Return to the query panel



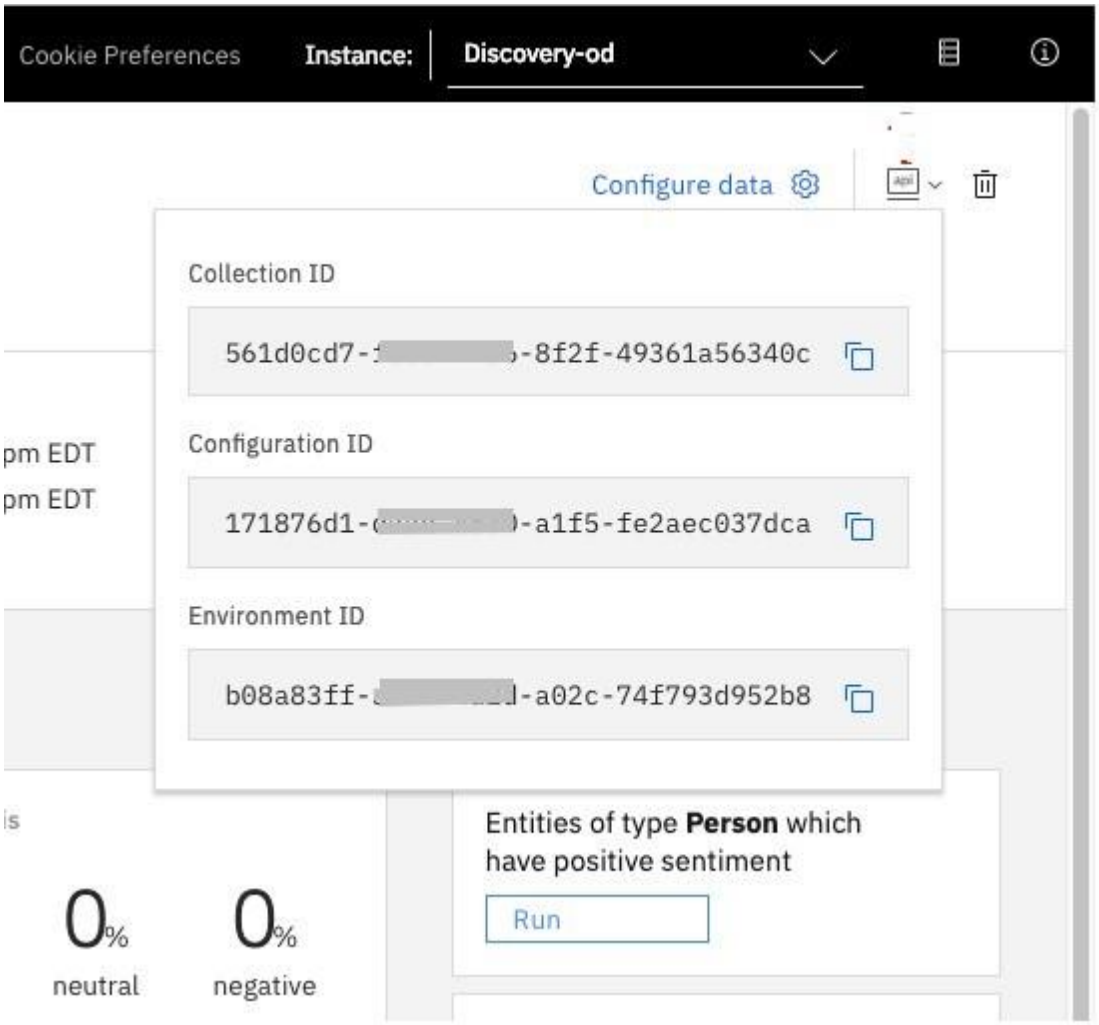
(click Build your own query) and see how much better the results are.



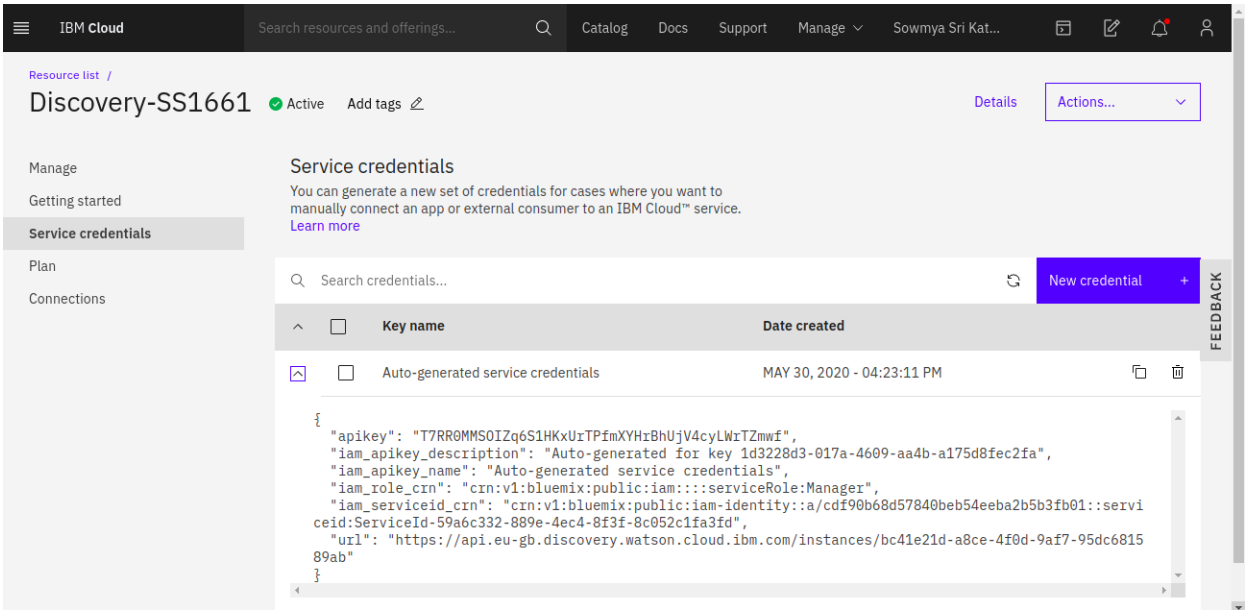
Store credentials for future use

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:

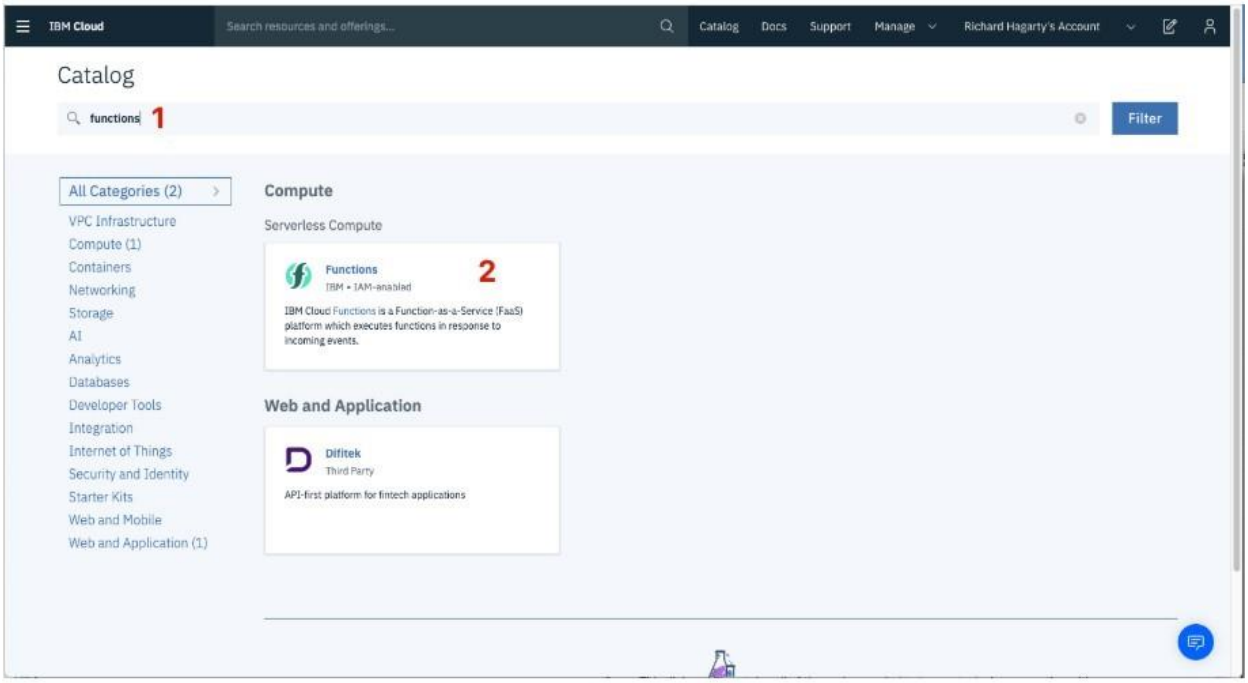


Click the View credentials [2] drop-down menu to view the IAM apikey [3] and URL endpoint [4] for your service.

3.Create IBM CloudFunctionsaction

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

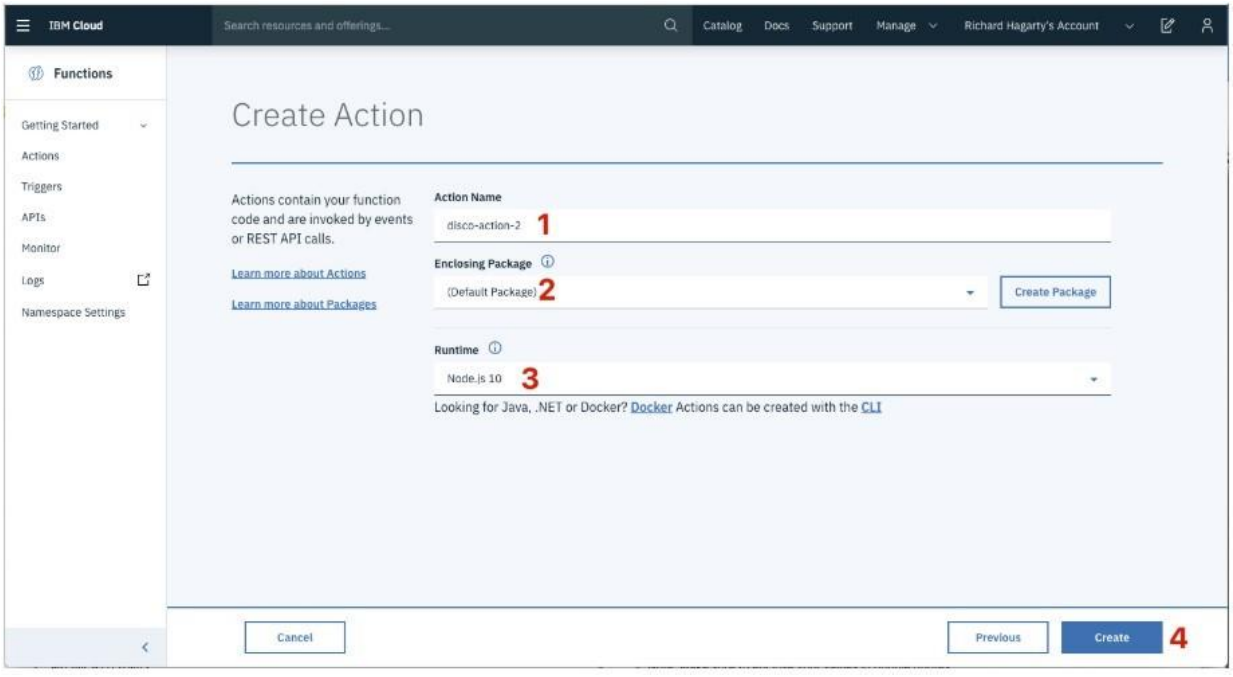
Start the IBM Cloud Functions service by selecting Create Resource from the IBM Cloud dashboard. Enter functions as the filter [1], then select the Functions card [2]:



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.

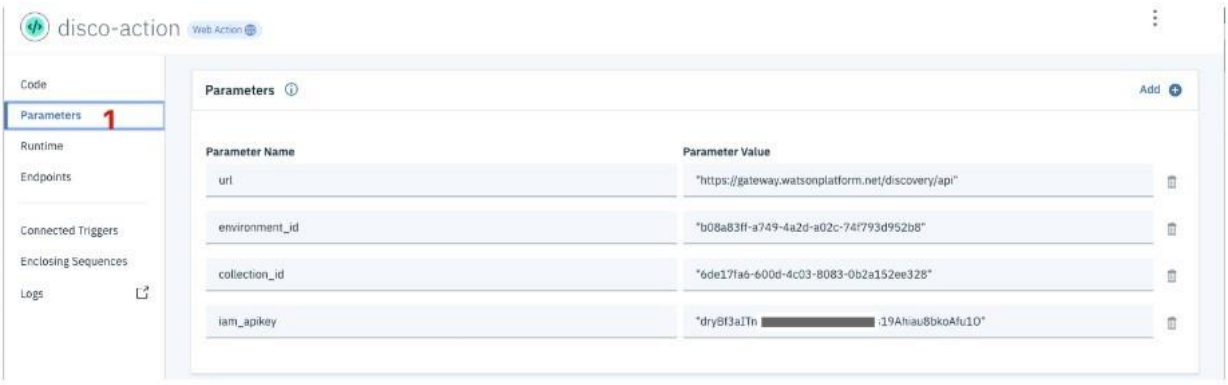


Once your action is created, click on the Code tab [1]:

In the code editor window [2], cut and paste in the codefromthedisco-action.js filefound in the actionsdirectoryof yourlocalrepo. The code is prettystraight-forward- itsimply connectsto theDiscoveryservice,makesaqueryagainst the collection, then returnstheresponse.

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]:

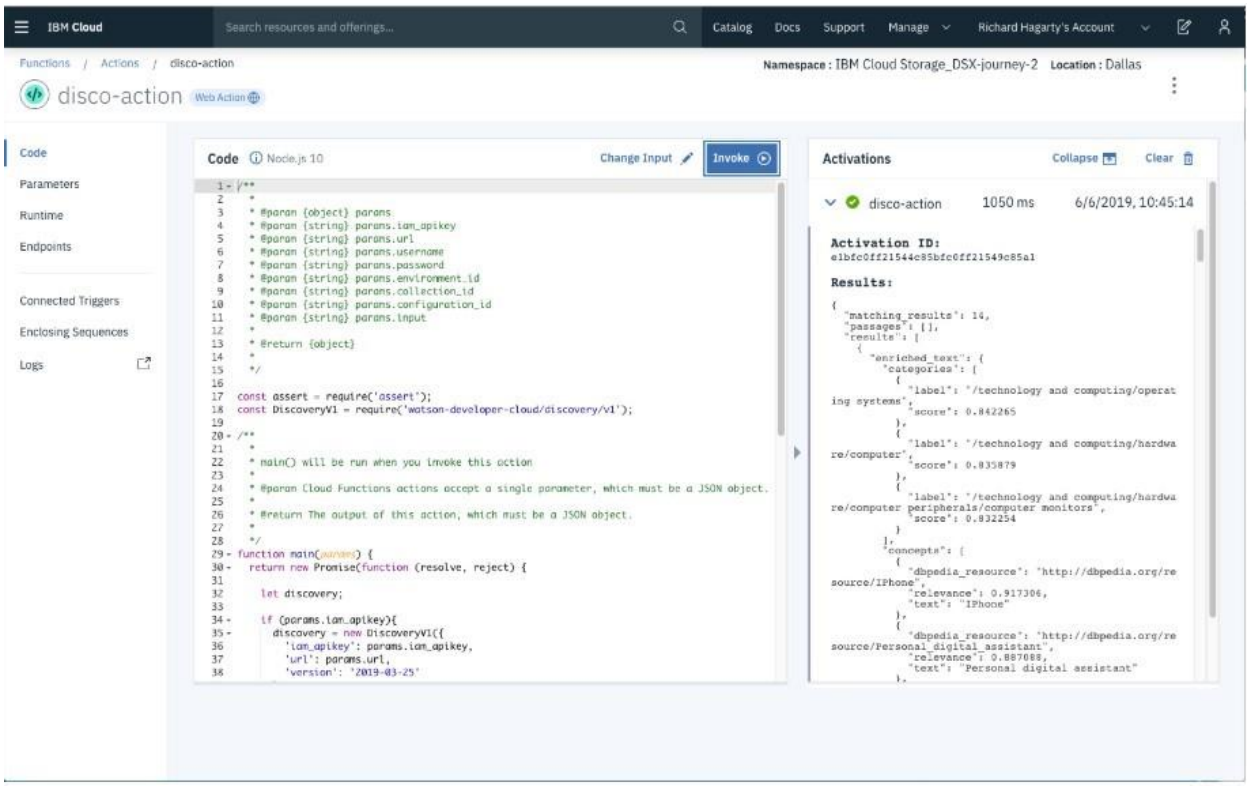


Add the following keys:

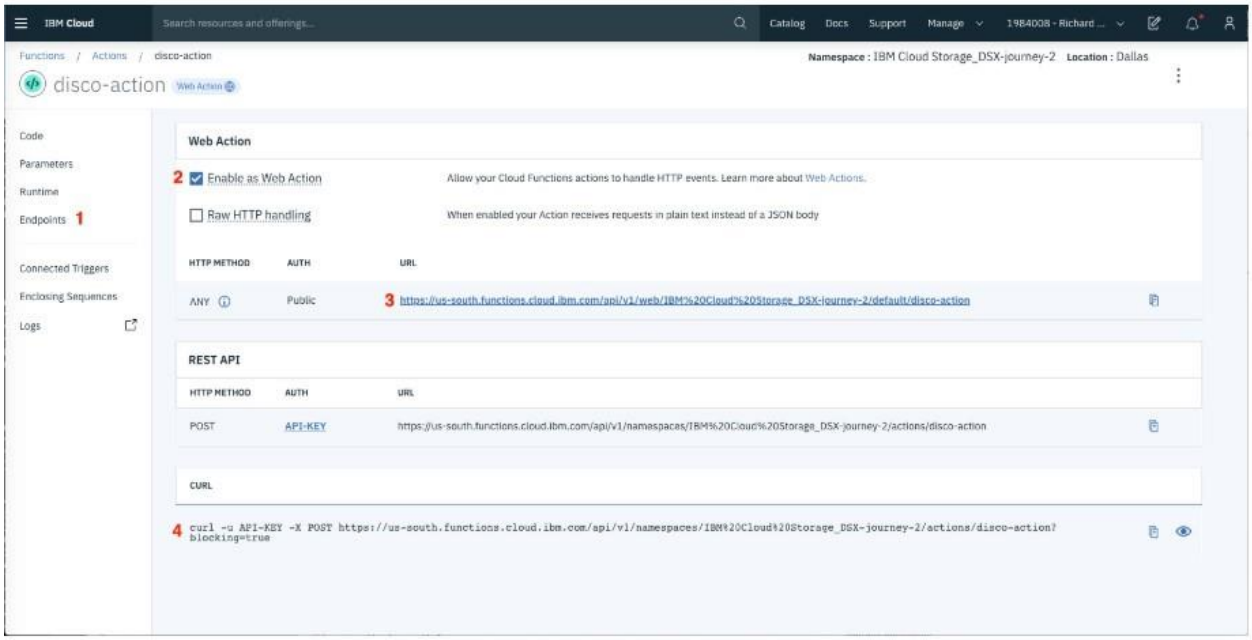
- url
- environment_id
- collection_id
- iam_apikey

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 haveenteredthe correctDiscovery parameters,executetheproviedcurlcommand [4]. If it fails, re-check yourparameterv a l u e s .

4.Configure WatsonAssistant

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

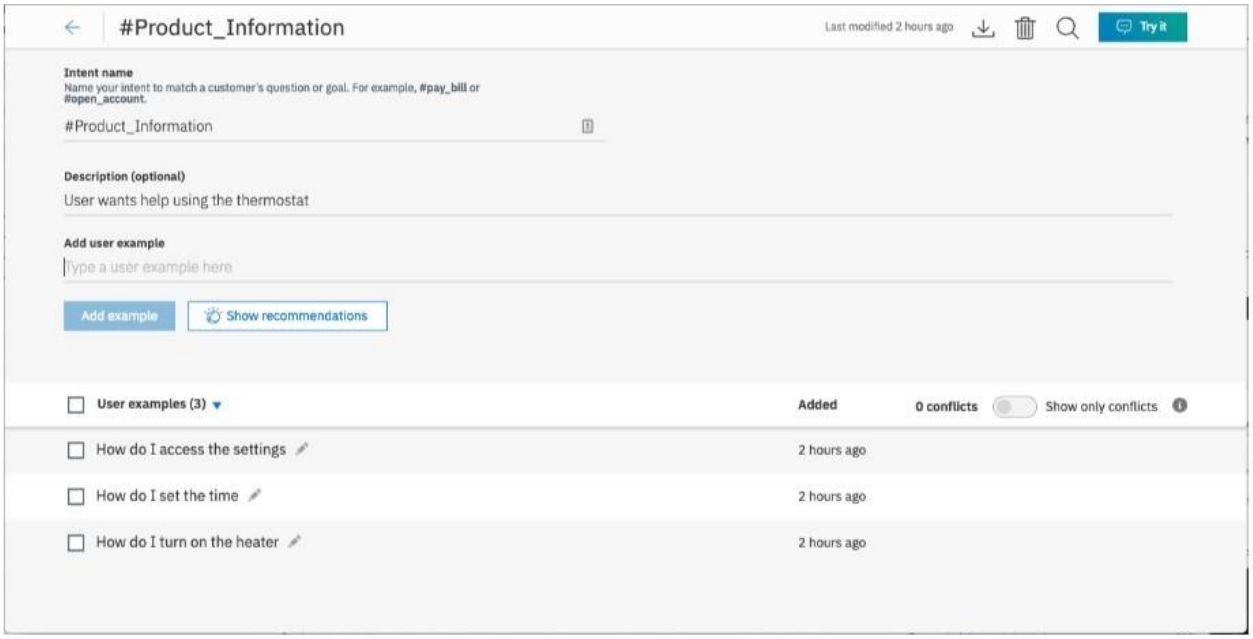
The default customer care dialog does not have a way to deal with any questions involving outside resources, so we will need to add this.

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

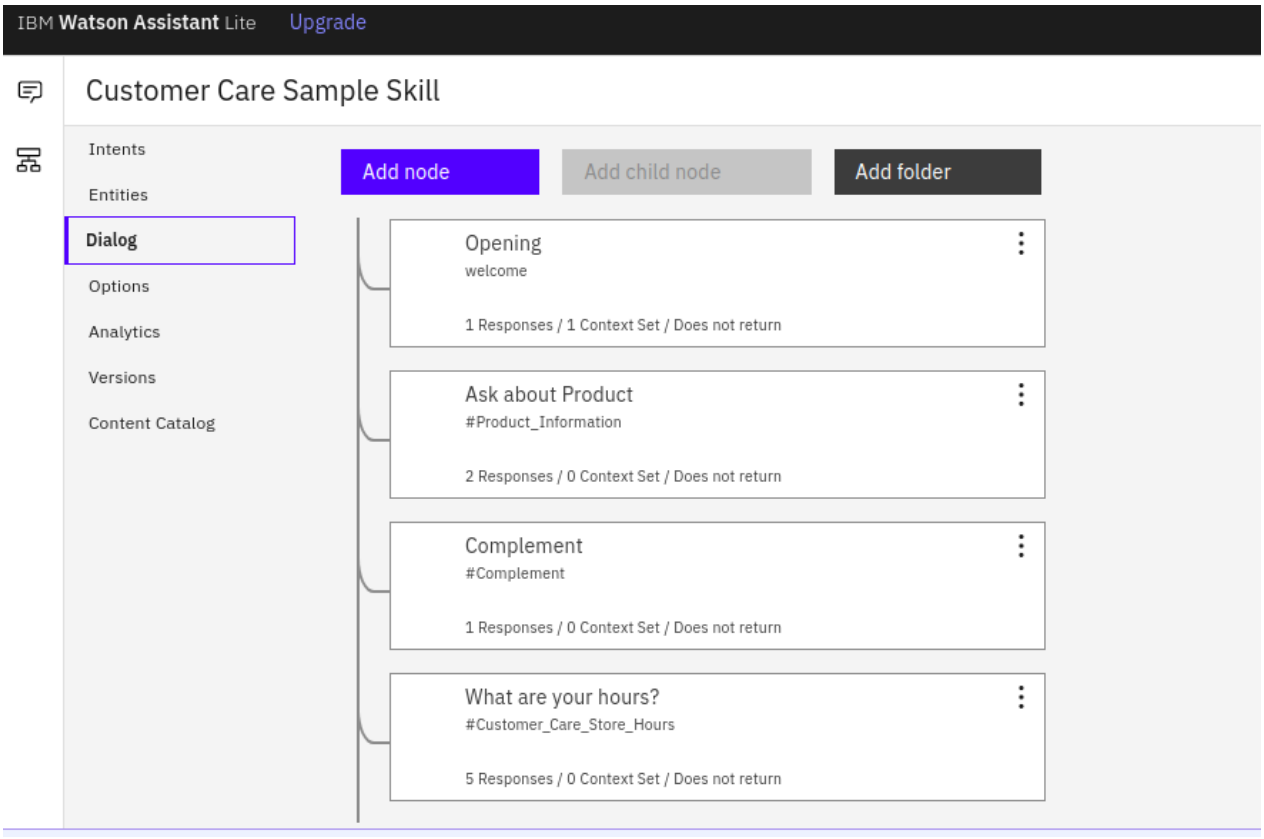
From the Customer Care Sample Skill panel, select the Intents tab.

Click the Create intent button.

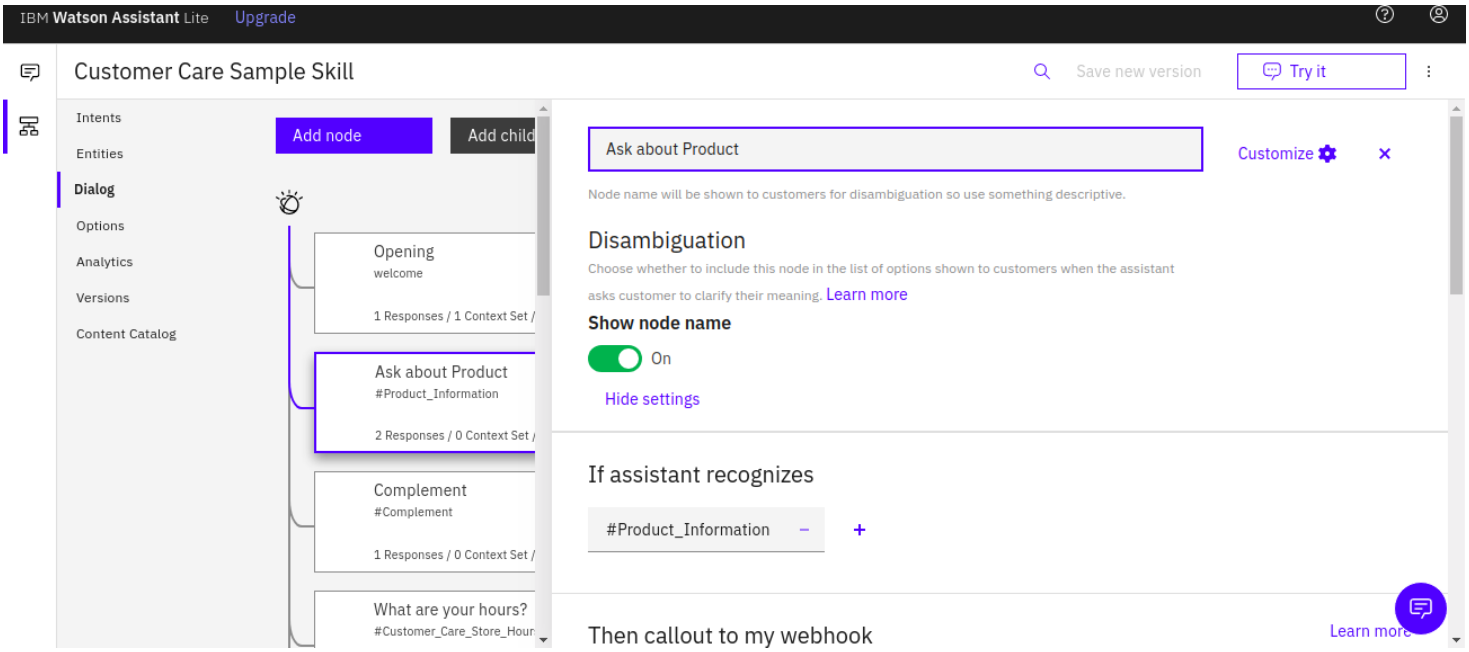
Name the intent #Product_Information, and at a minimum, enter the following example questions to be associated with it.



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.



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

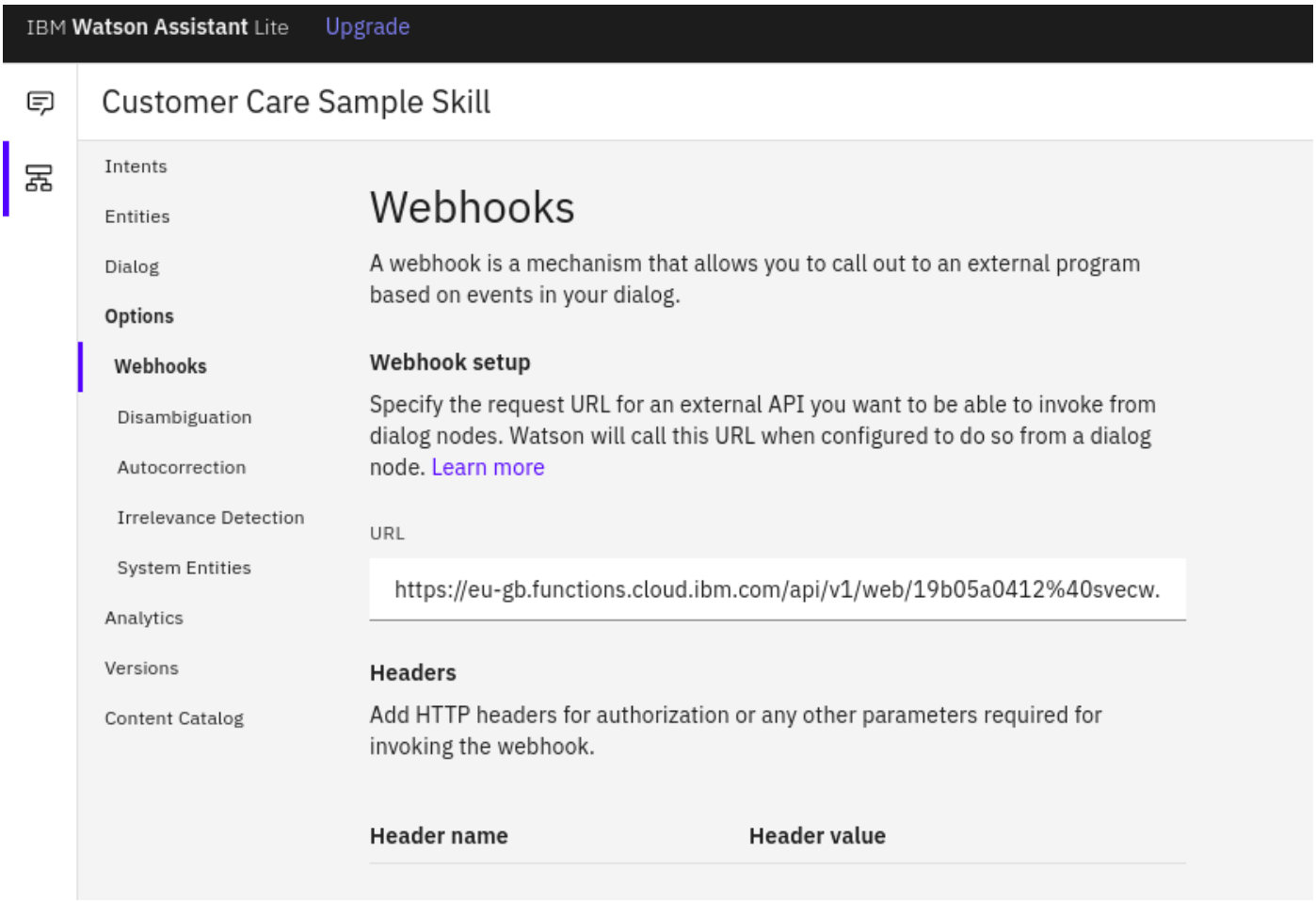


This means that if Watson Assistant recognizes a user input such as "how do I set the time?", it will direct the conversation to this node.

Edit the node for Assistant

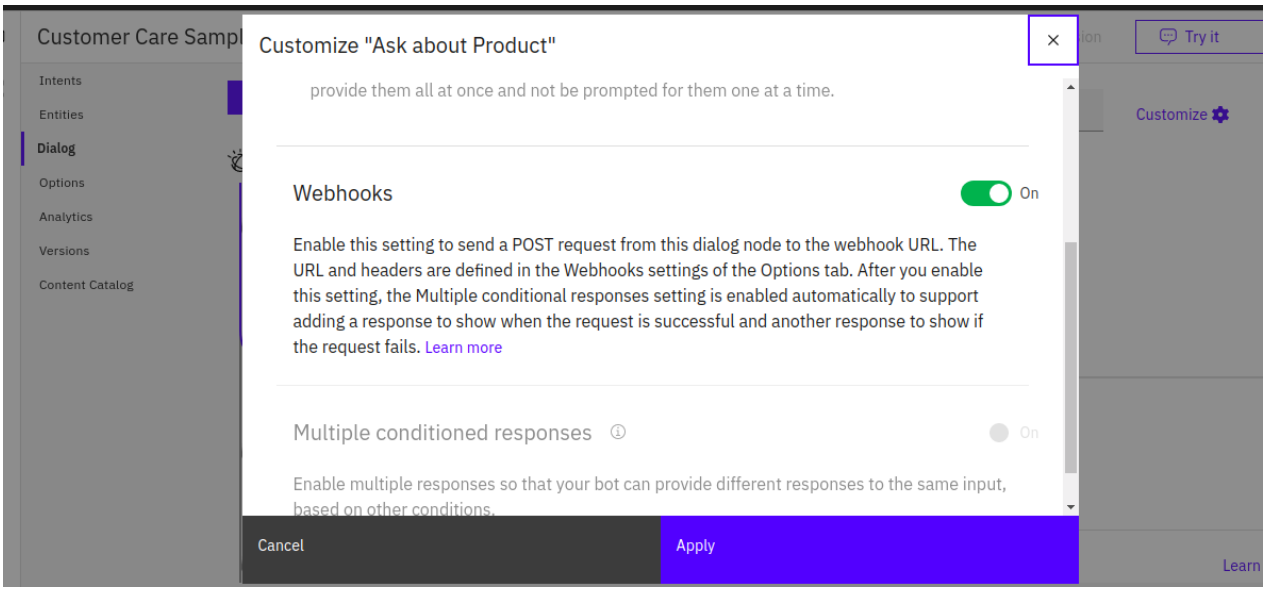
Set up access to our WebHook for the IBM Cloud Functions action you created in Step #4.

Select the Options tab [1]:



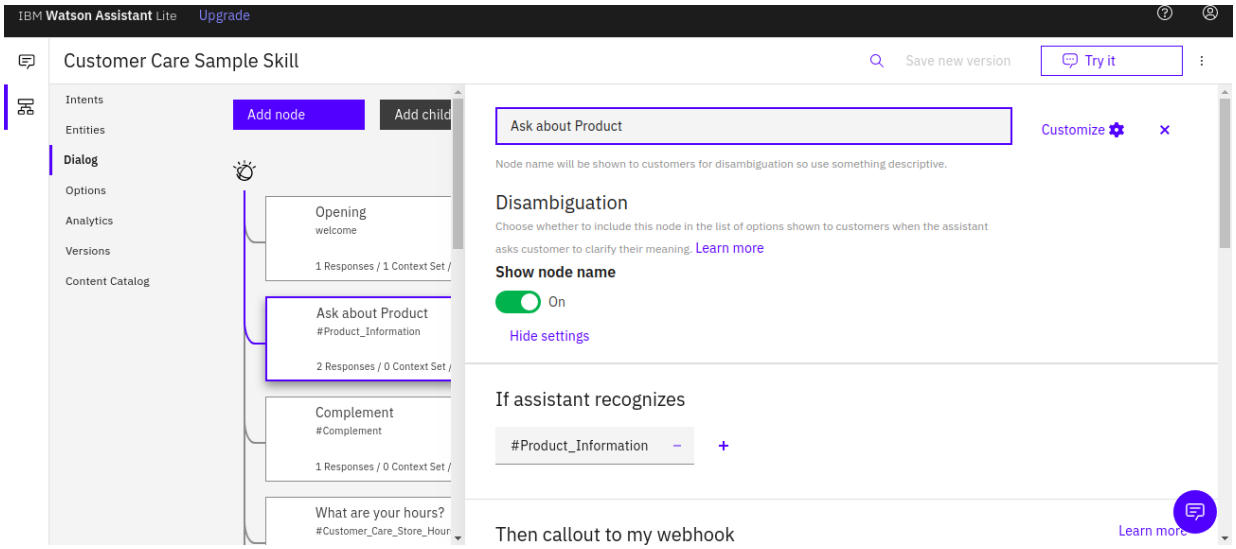
Enter the public URL endpoint for your action [2].

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.

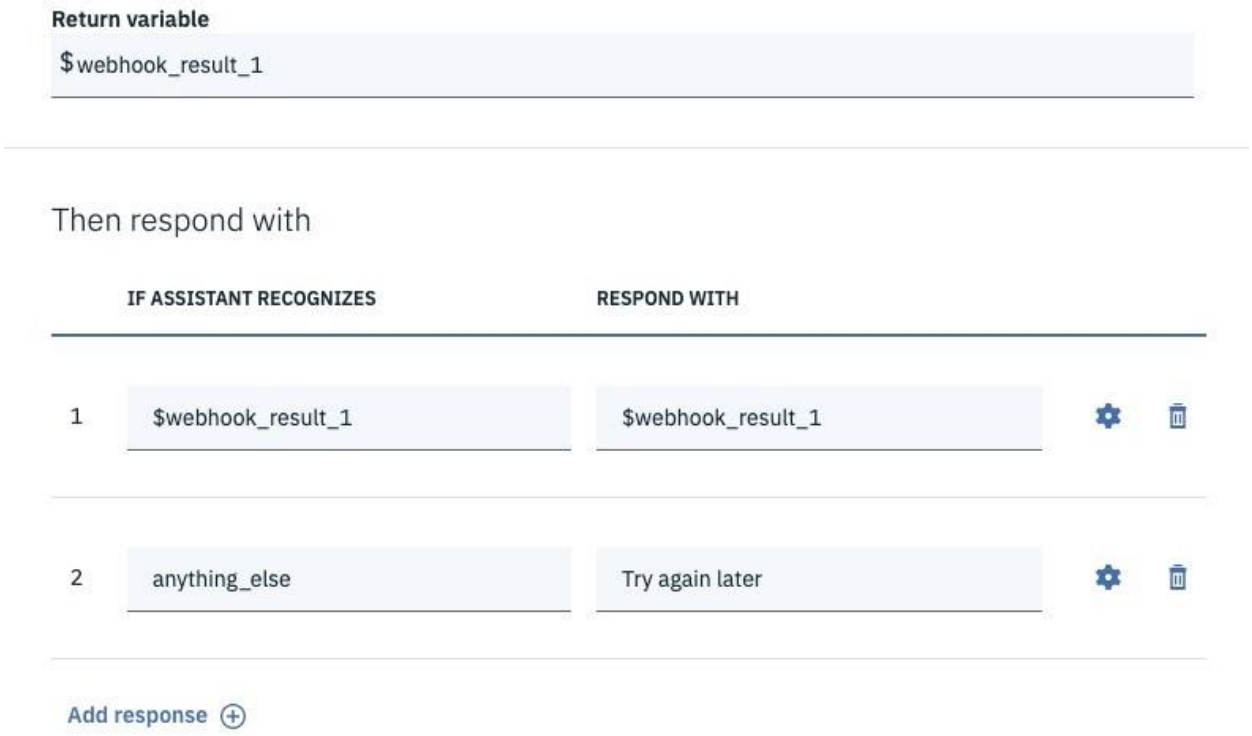
The dialog node should have a Return variable [1] set automatically to $webhook_result_1. This is the variable name you can use to access the result from the Discovery service query.



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.

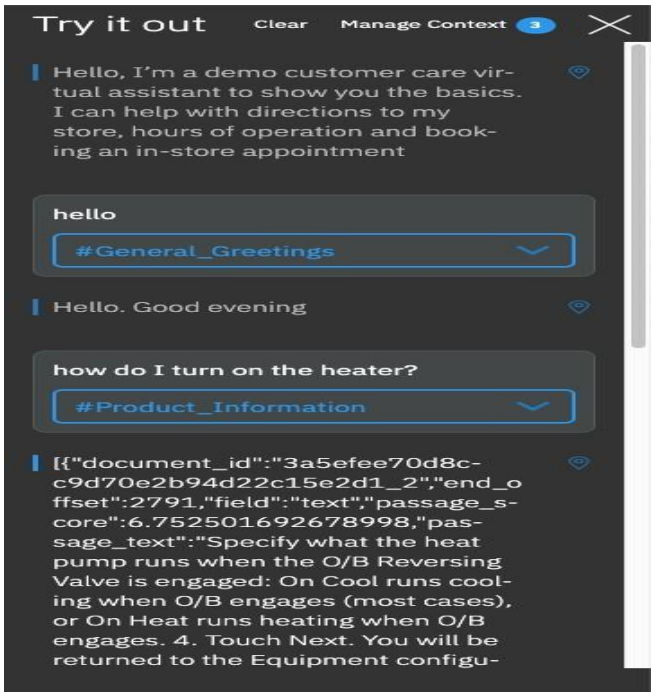
Optionally, you can add these responses to aid in debugging:



Testing Assistant Tooling

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

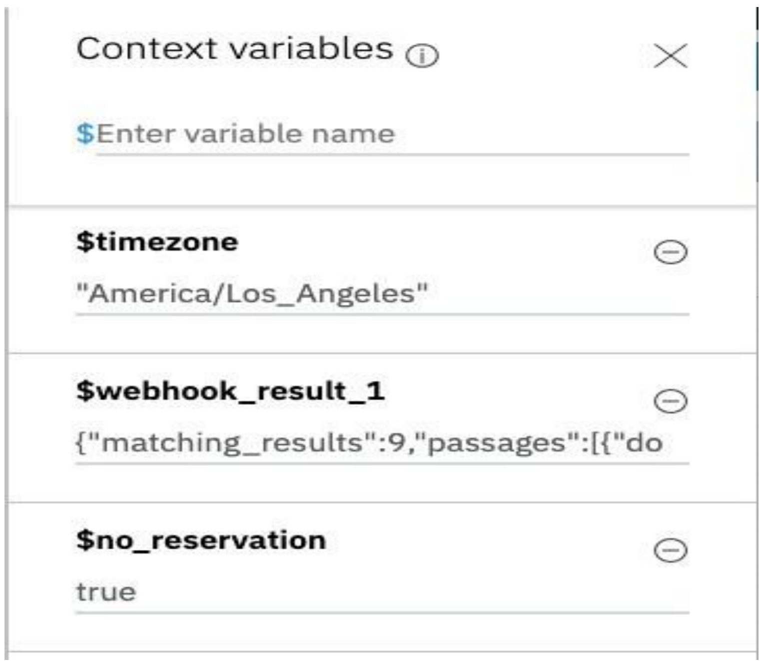
Enter some user input:



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.

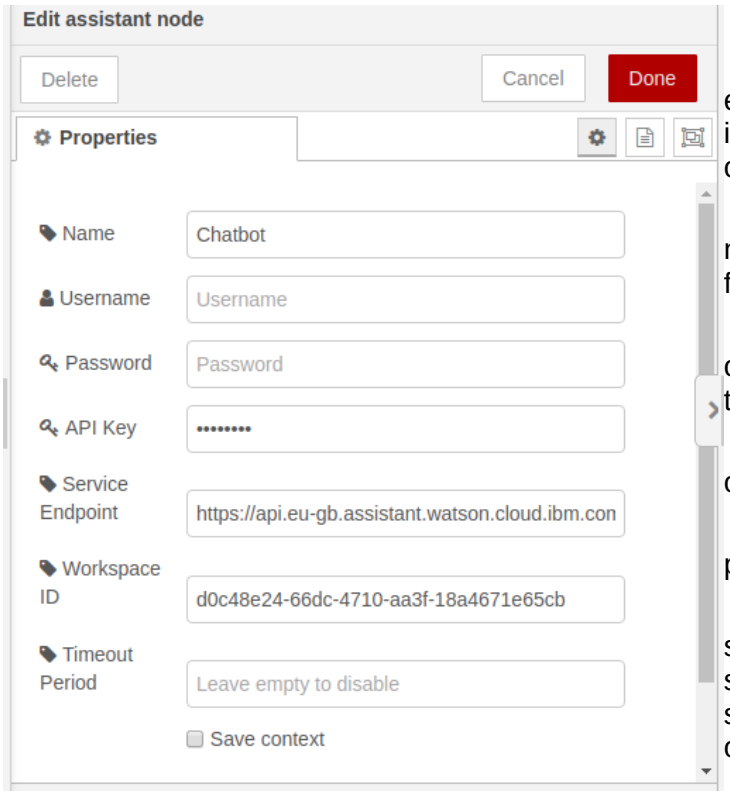
You can also verify that the call was successfully completed by clicking on the Manage Context button at the top right. The response from the Discovery query will be stored in the $webhook_result_1 variable:



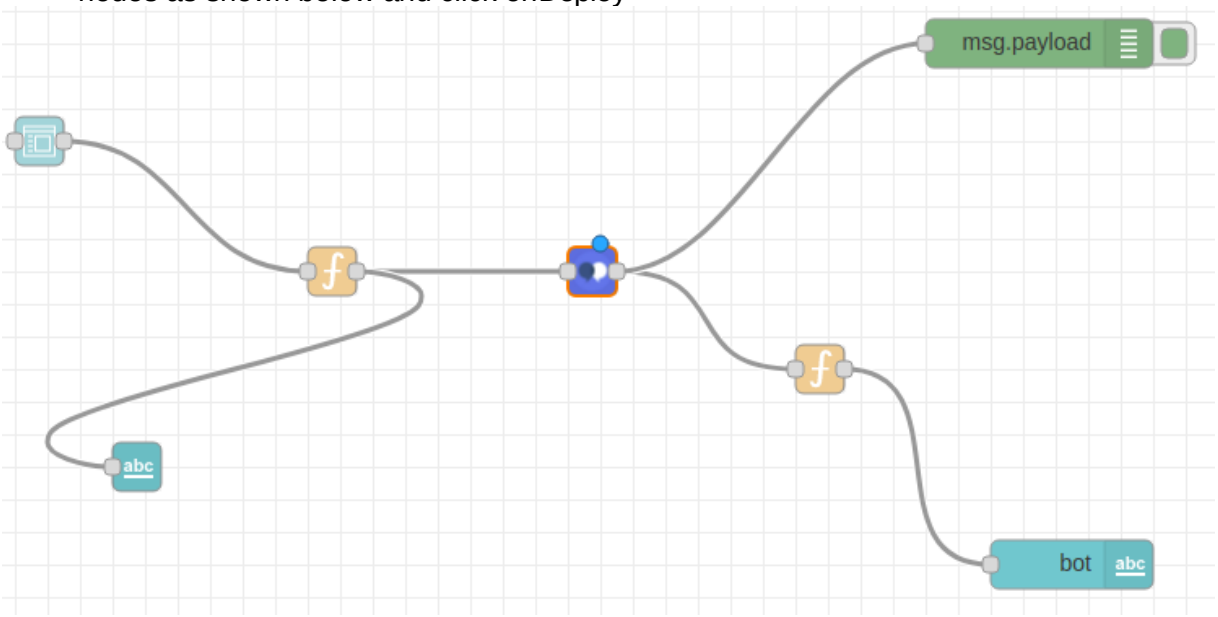
5.Create flow and configure node:

Integration of watson assistant in Node-RED

- ➢ Double-click on the Watson assistant node
- ➢ Give a name to your node and enter the username, password and workspace id of your Watson assistant service



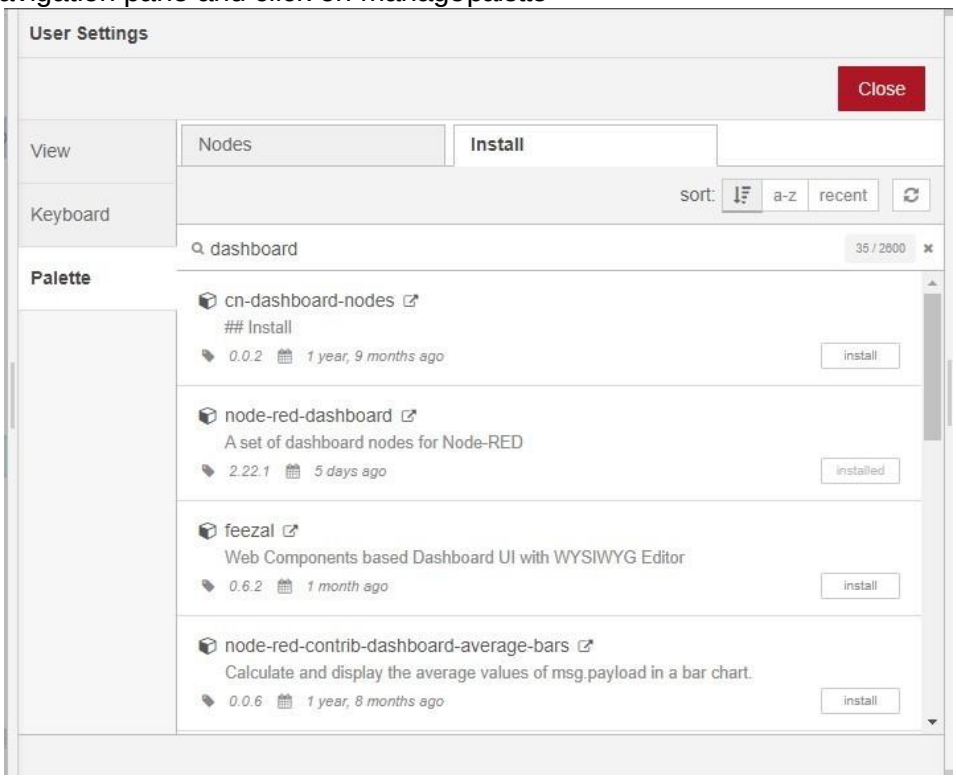
- After entering all the information click onDone
- Drag inject node on to the flow from the Inputsection
- Drag Debug on to the flow from the outputsection
- Double-click on the injectnode
- Select the payload as astring
- Enter a sample input to be sent to the assistant service and click ondone
- Connect the nodes as shown below and click onDeploy



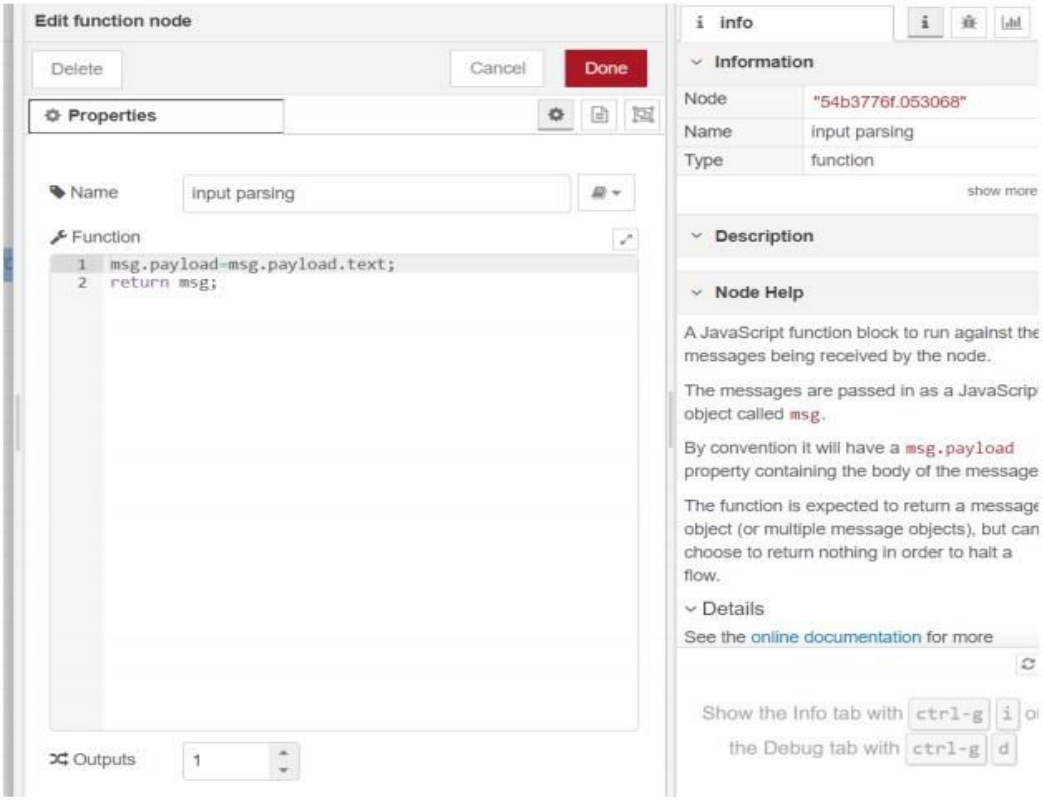
- Open Debug window as shownbelow
- Click on the button to send input text to the assistantnode
- Observe the output from the assistant servicenode
- The Bot output is located inside “output.text"
- Drag the function node to parse the JSON data and get the botresponse
- Double click on the function node and enter the JSON parsing code as shown below and click on done ● Connect the nodes as shown below and click onDeploy

- Re-inject the flow and observe the parsedoutput

- For creating a webapplication
  UI we need “dashboard“ nodes which should be installed manually.
- Go to navigation pane and click on managepalette



- Click oninstall
- Search for “node-red-dashboard” and click on install and again click on installon theprompt
- The following message indicates dashboard nodes are installed, closethe managepalette
- Search for “Form” node and drag on to theflow
- Doube click on the “form” node toconfigure
- Click on the edit button to add the “Group” name and “Tab”name
- Click on the edit button to add tab name to webapplication
- Give sample tab name and click on add do the same thing for thegroup
- Give the label as “Enter your input”, Name as “text” and click onDone
- Drag a function node, double-click on it and enter the input parsing codeas shownbelow



- Click ondone
- Connect the form output to the input of the function node and output ofthe function to input of assistantnode
- Search for “text” node from the “dashboard”section
- Drag two “text” nodes on to theflow
- Double click on the first text node, change the label as “You” and click onDone
- Double click on the second text node, change the label as “Bot” and click onDone
- Connect the output of “input parsing” function node to “ You” text node and output of “Parsing” function node to the input of “Bot” textnode
- Click onDeploy

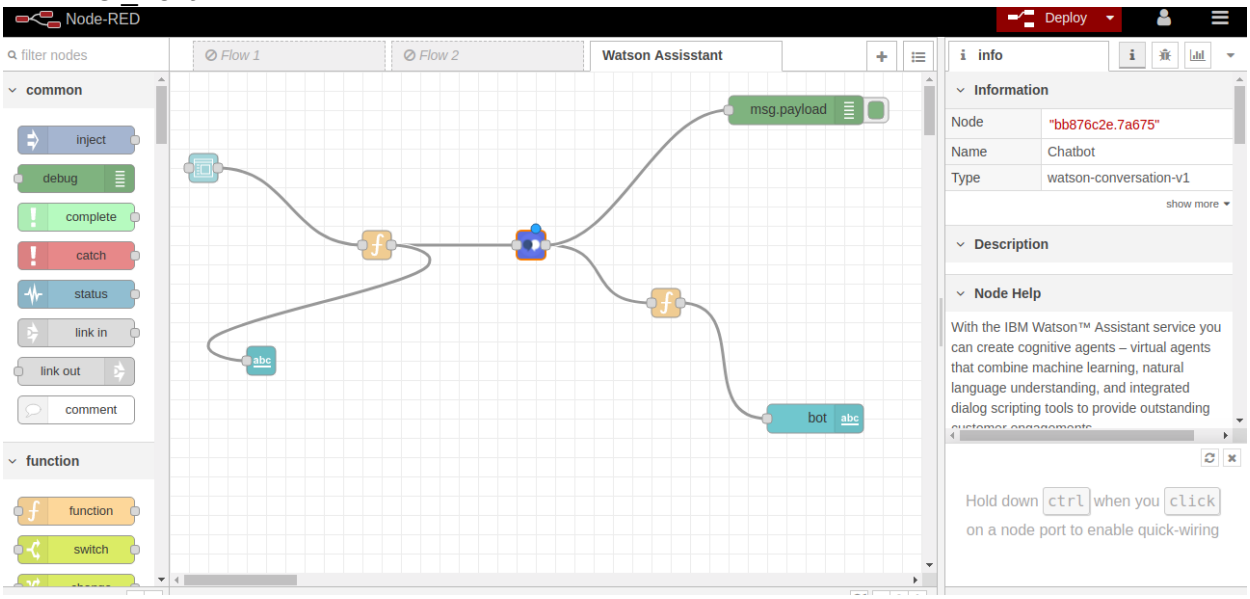
5. FLOWCHART

At first go to manage pallete and install dashboard.

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

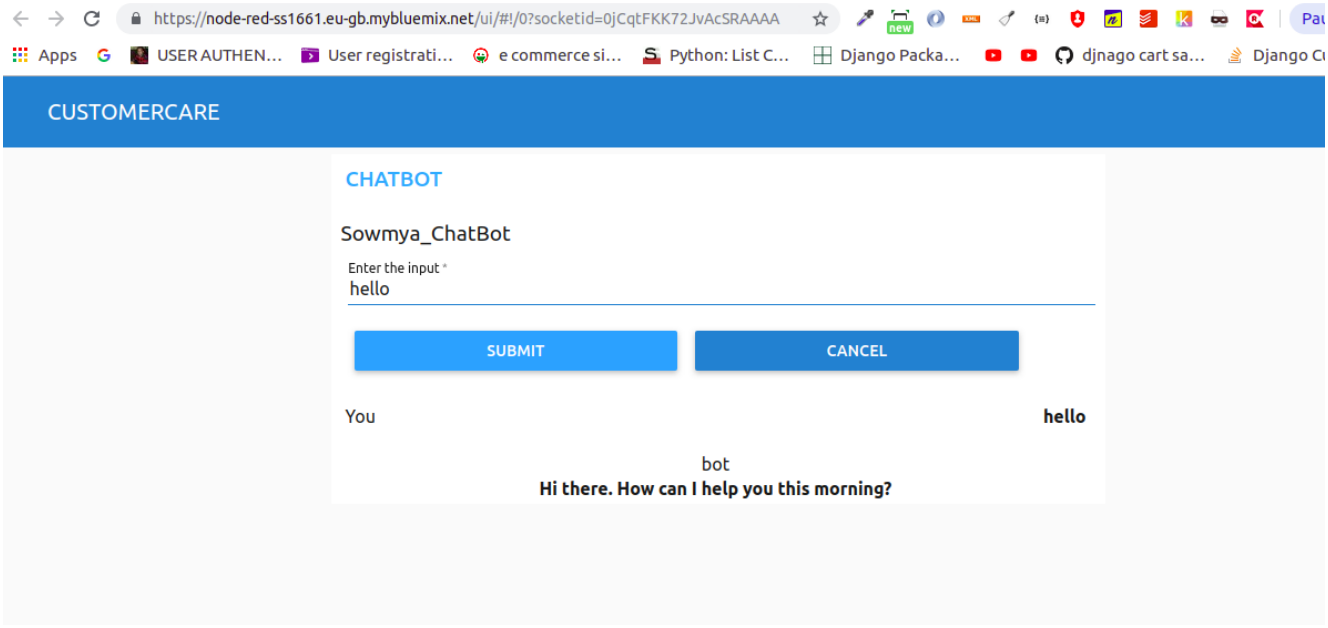
- ✓ Inject
- ✓ Assistant
- ✓ Debug

- ✓ Function
- ✓ Ui_Form
- ✓ Ui_Text



6. RESULTS

Finally our Node-RED dash board integrates all the components and displayed in the Dashboard UI by typing URL -https://node-red-ss1661.eu-gb.mybluemix.net/ui/#!/0?socketid=0jCqtFKK72JvAcSRAAAA



7. ADVANTAGES &DISADVANTAGES

Advantages:

- ✓ Campanies can deploy chatbots to rectifiy simple and general human queries.
- ✓ Reduces manpower
- ✓ Costefficient
- ✓ No need to divert calls to customer agent and customer agent can look on otherworks.

Disadvantages:

- ✓ Some times chatbot can misleadcustomers
- ✓ Giving same answer for differentsentiments.
- ✓ Some times cannot connect to customer sentiments andintentions.

8. APPLICATIONS

- ❖ It can deploy in popular social media applications likefacebook,slack,telegram.
- ❖ Chatbot can deploy any website to clarify basic doubts ofviewers.

9. CONCLUSION

By doing the above procedure and all we successfully created Intelligent helpdesk smart chartbot using Watson assistant, Watson discovery, Node-RED and cloud-functions.

10. FUTURESCOPE

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

11. BIBILOGRAPHY

APPENDIX

Source Code

1. CloudFunction(Node.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 whenyou invokethisaction
 *
 * @paramCloudFunctionsactionsacceptasingleparameter,whichmustbeaJSONobject.
 *
 * @returnTheoutputof thisaction,which mustbea JSONobject.
 *
 */
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': '2020-05-09'
  });
}
else {
  discovery = new DiscoveryV1({
    'username': params.username,
    'password': params.password,
    'url': params.url,
    'version': '2020-05-11'
  });
}

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);
    });
  });
}
```

2. Node Red(flow.json)

```
[
    {
        "id": "7253a121.16642",
        "type": "tab",
        "label": "Flow 1",
        "disabled": false,
        "info": ""
```

```
},
{
    "id": "b1b11140.4e4ef",
    "type": "inject",
    "z": "7253a121.16642",
    "name": "",
    "topic": "",
    "payload": "Hello Node-RED!",
    "payloadType": "str",
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": "",
    "x": 141,
    "y": 61,
    "wires": [
        [
            "2371449b.4bf2cc"
        ]
    ]
```

```
},
{
    "id": "f2f2649a.0d0d98",
    "type": "debug",
    "z": "7253a121.16642",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType":
    "msg","x":670,
    "y":140,
    "wires": []
},
{
    "id": "e150cc25.d7aa",
    "type": "function",
    "z": "7253a121.16642",
```

```
    "name": "input parsing",
    "func": "msg.payload=msg.payload.text;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x":270,
    "y":240,
    "wires": [
        [
            "2371449b.4bf2cc",
            "49234419.64d55c"
        ]
    ]
},
{
    "id": "d3215cca.6b835",
    "type": "ui_form",
    "z": "7253a121.16642",
    "name": "",
    "label": "",
    "group": "528b62f3.5c783c",
```

```
"order": 6,
"width": "0",
"height": "0",
"options": [
    {
        "label": "Enter your query",
        "value": "text",
        "type": "text",
        "required": true,
        "rows": null
    }
],
"formValue": {
    "text": ""
},
"payload": "",
"submit": "submit",
"cancel": "cancel",
"topic":"",
"x":107.5,
```

```
    "y": 302,
    "wires": [
        [
            "e150cc25.d7aa"
        ]
    ]
},
{
    "id": "49234419.64d55c",
    "type": "ui_text",
    "z": "7253a121.16642",
    "group": "528b62f3.5c783c",
    "order": 1,
    "width": "6",
    "height": "1",
    "name": "",
    "label": "You",
    "format": "{{msg.payload}}",
    "layout": "col-center",
    "x": 453.5,
```

```
    "y": 306,
    "wires": []
},
{
    "id": "d2a61bef.00f978",
    "type": "ui_text",
    "z": "7253a121.16642",
    "group": "528b62f3.5c783c",
    "order": 2,
    "width": "6",
    "height": "1",
    "name": "",
    "label": "Bot",
    "format": "{{msg.payload}}",
    "layout": "col-center",
    "x": 688.5,
    "y": 237,
    "wires": []
},
{
```

```
    "id": "29aac8ee.b98c68",
    "type": "function",
    "z": "7253a121.16642",
    "name": "parsing",
    "func": "msg.payload = msg.payload.output.generic[0].text;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x":470,
    "y":160,
    "wires": [
      [
        "f2f2649a.0d0d98",
        "d2a61bef.00f978"
      ]
    ]
},
{
    "id": "2371449b.4bf2cc",
    "type": "watson-assistant-v2",
    "z": "7253a121.16642",
```

```
        "name": "Chatbot",
        "service-endpoint": "https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/ae44476e-77ab-4578-9d11-23464ea66634",
        "assistant_id": "d3fa58b9-7946-4784-b120-5e025c58c9ba",
        "debug": false,
        "restart": false,
        "return_context": true,
        "alternate_intents": false,
        "multisession": true,
        "timeout": "",
        "optout-learning": false,
        "x": 391.5,
        "y": 100,
        "wires": [
            [
                "29aac8ee.b98c68",
                "caf5ac15.c978d"
            ]
        ]
    },
```

```
{
    "id": "caf5ac15.c978d",
    "type": "debug",
    "z": "7253a121.16642",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "x": 557.5,
    "y": 61,
    "wires": []
},
{
    "id": "528b62f3.5c783c",
    "type": "ui_group",
    "z": "",
    "name": "",
```

```
        "tab": "27e055be.28bb7a",
        "order": 1,
        "disp": true,
        "width": "6",
        "collapse": false
    },
    {
        "id": "27e055be.28bb7a",
        "type": "ui_tab",
        "z": "",
        "name": "Product",
        "icon": "dashboard",
        "order": 2,
        "disabled": false,
        "hidden": false
    }
]
```


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