Project Report

Name : Sanskar Gupta
(Sanskar.gupta2017@vitstudent.ac.in)

Title : Intelligent Customer Help Desk With Smart Document Understanding

Category: Artificial Intelligence/ Machine Learning

Internship at smartinternz.com@2020

INTRODUCTION

Overview

We will be designing an application that leverages multiple Watson AIServices (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 be used to 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: Sanskar Gupta(SG)
- ➢ Project Duration: 19 days

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. The purpose of this project is to build a customer helping chatbot such that 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 owner's manual. So, instead of "Would you like to speak to a customer representative?" we can return relevant sections of the owner's manual to help solve our customers' problems.

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.

LITERATURE SURVEY

Existing problem

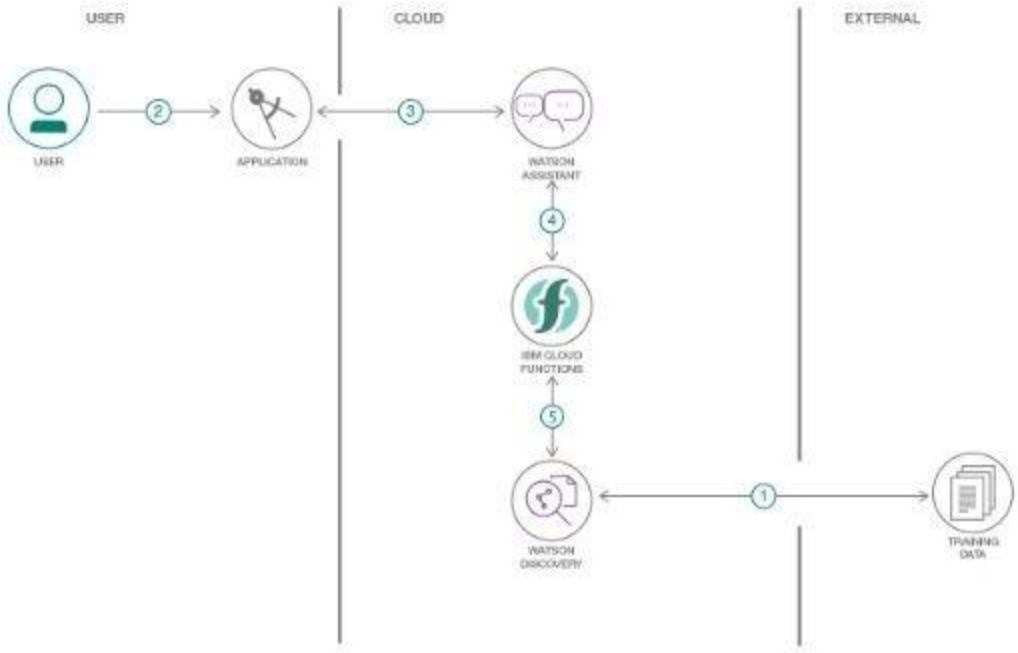
Generally chatbots means getting input from users and getting only response questions and for some questions the output from bot will be like " try again", "I don't understand", "will you repeat again", and so on... and directs customer to customer agent but a good customer Chatbot should minimize involvement 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 the customer can clarify his doubts with fast chatbots.

Proposed solution

For the above problem we are able to put a virtual agent in chatbot so it can understand the queries that are posted by customers. The virtual agent should train from some insight records based company background so it can answer queries supported by the merchandise or associated with the company. In other words, some styles of manual will be accustomed to train the bot using AI. Here I'm using Watson Discovery as a tool for implementing AI and getting trained by the owner's manual.

THEORETICAL ANALYSIS

Block/Flow Diagram



1. The document is annotated using Watson Discovery Smart Document Understanding
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.

Hardware / Software designing

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

EXPERIMENTAL INVESTIGATIONS

Create IBM Cloud services

Create the following services:

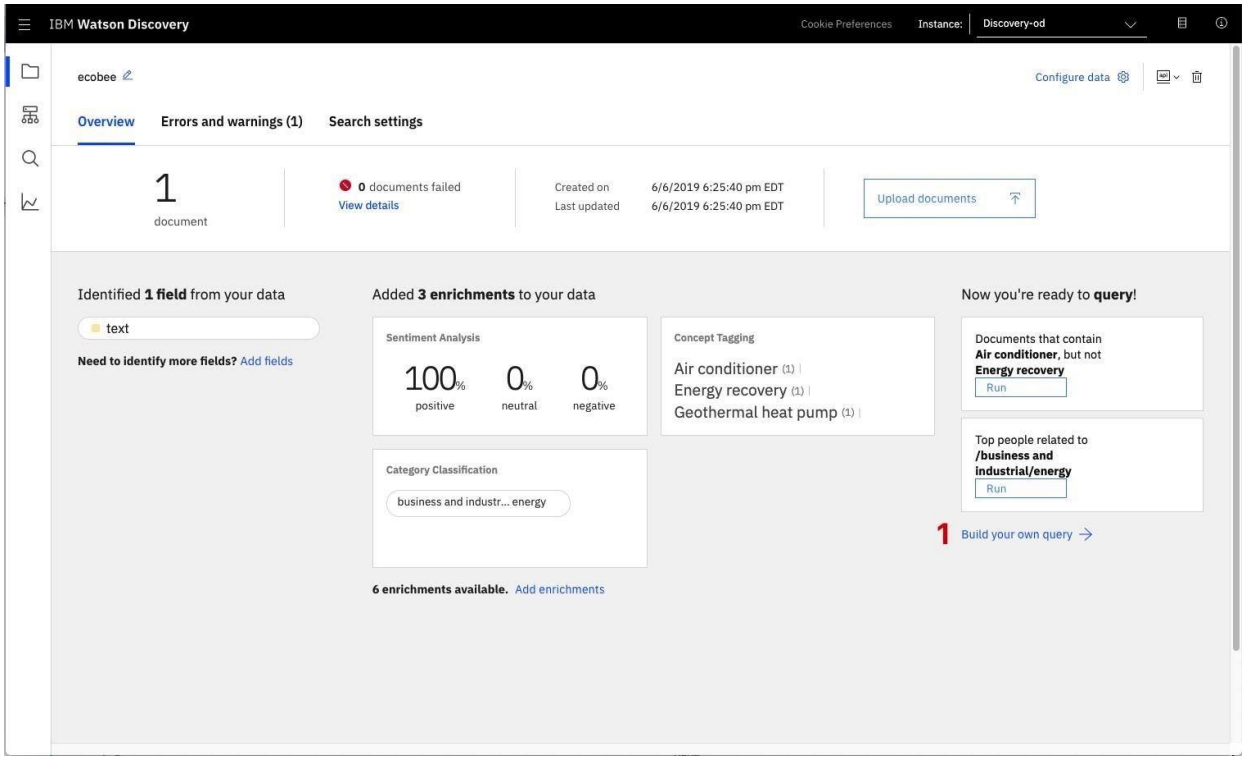
- ➢ Watson Discovery
- ➢ Watson Assistant
- ➢ IBM cloud function
- ➢ Node Red

Configure Watson Discovery

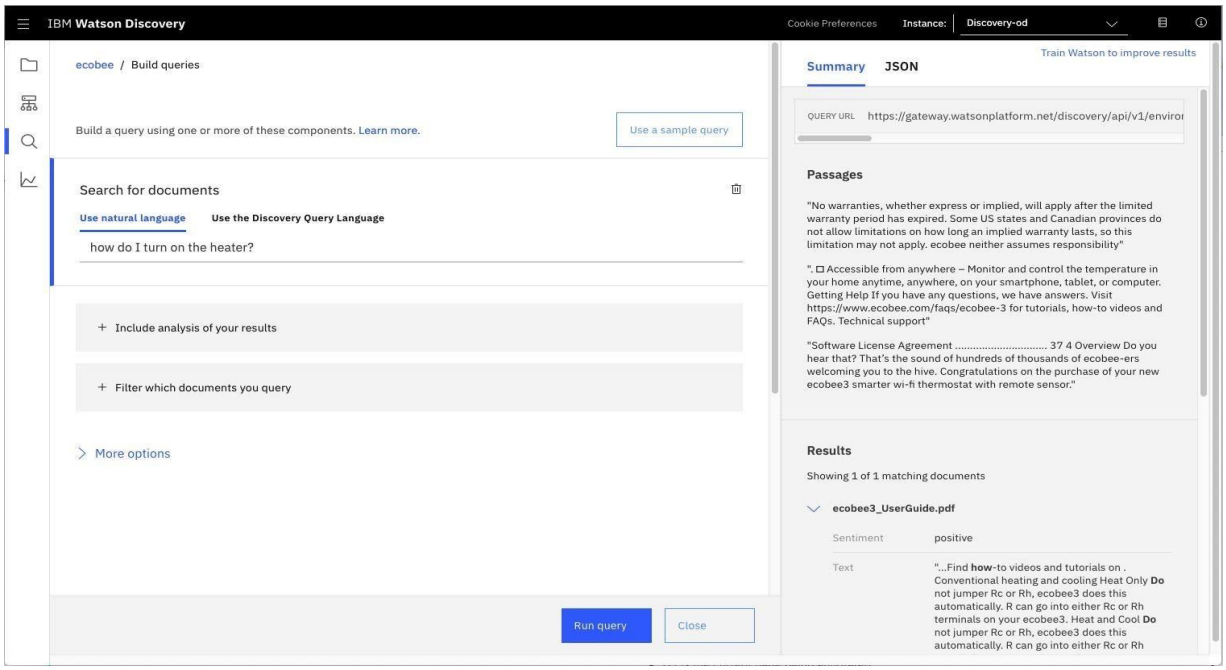
Import the document

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, let's do some simple queries on the data so that we can compare it to results found after applying SDU.



Click the Build your own query [1] button.

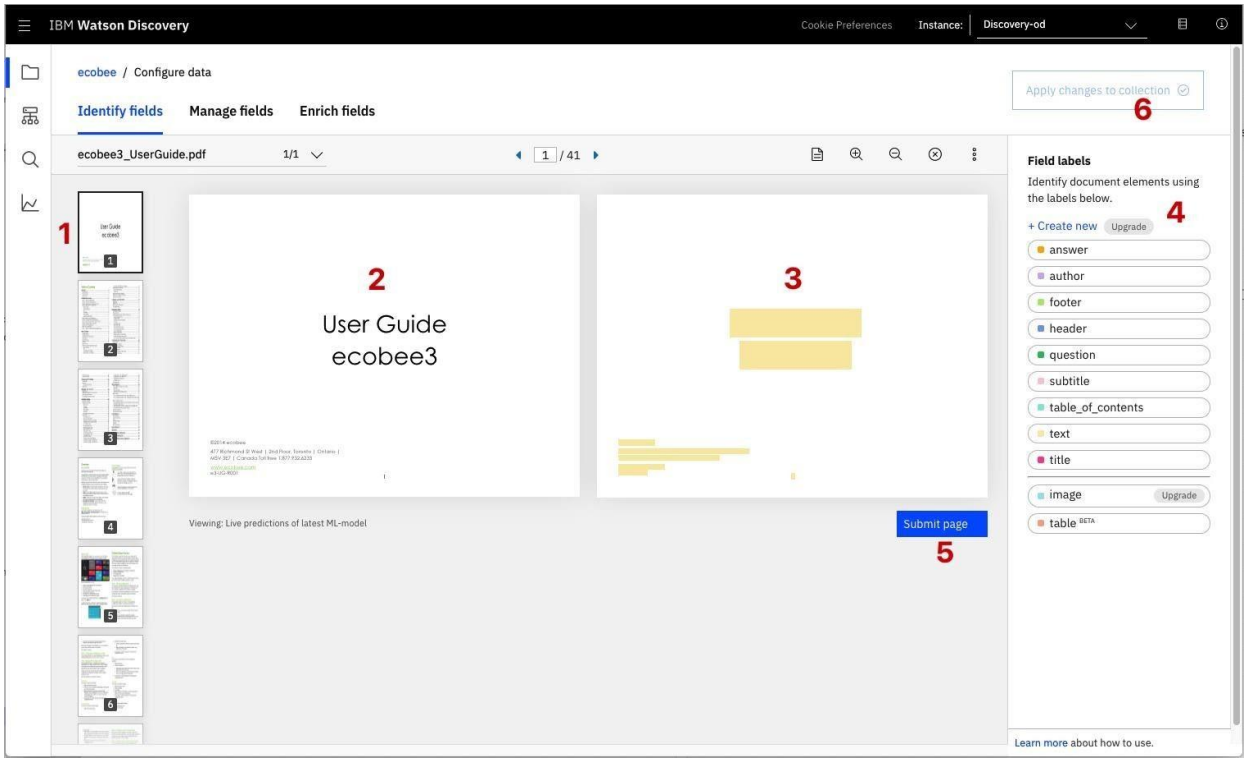


Enter queries related to the operation of the thermostat and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question.

Annotate with SDU

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.

[!] 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 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

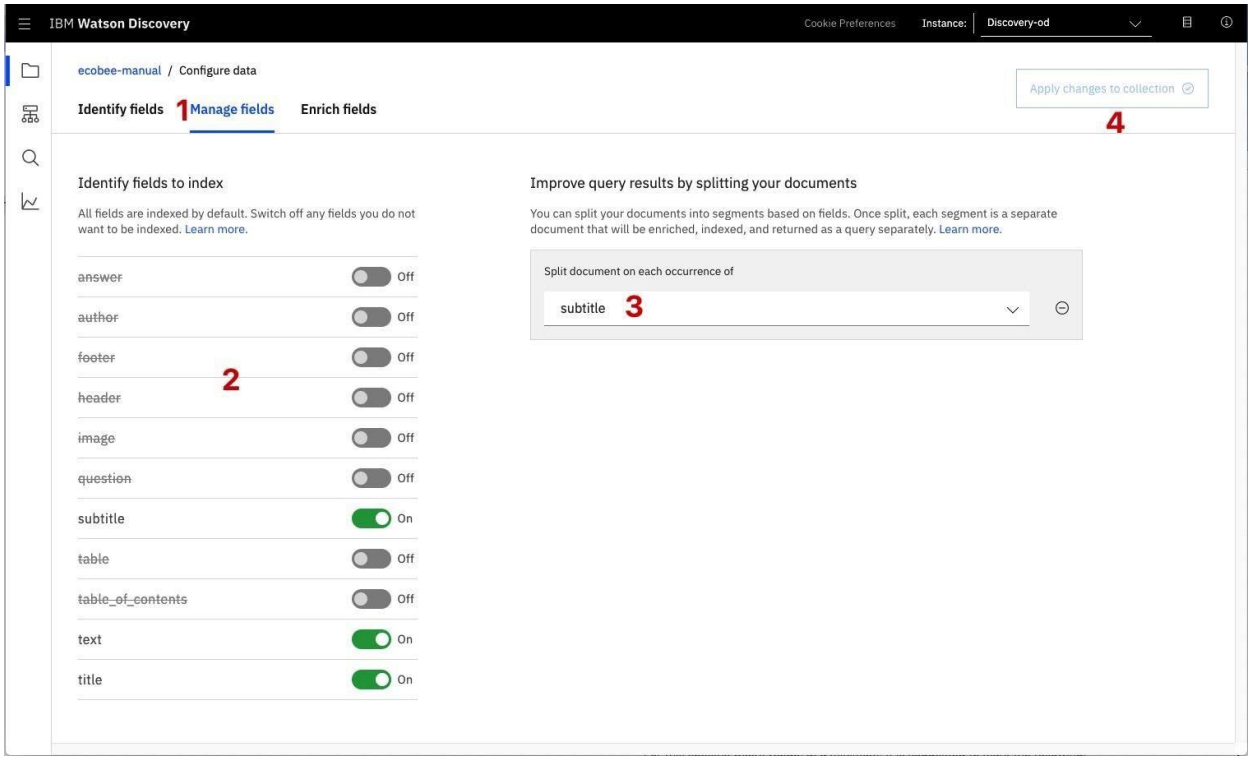
All page numbers as footers

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.



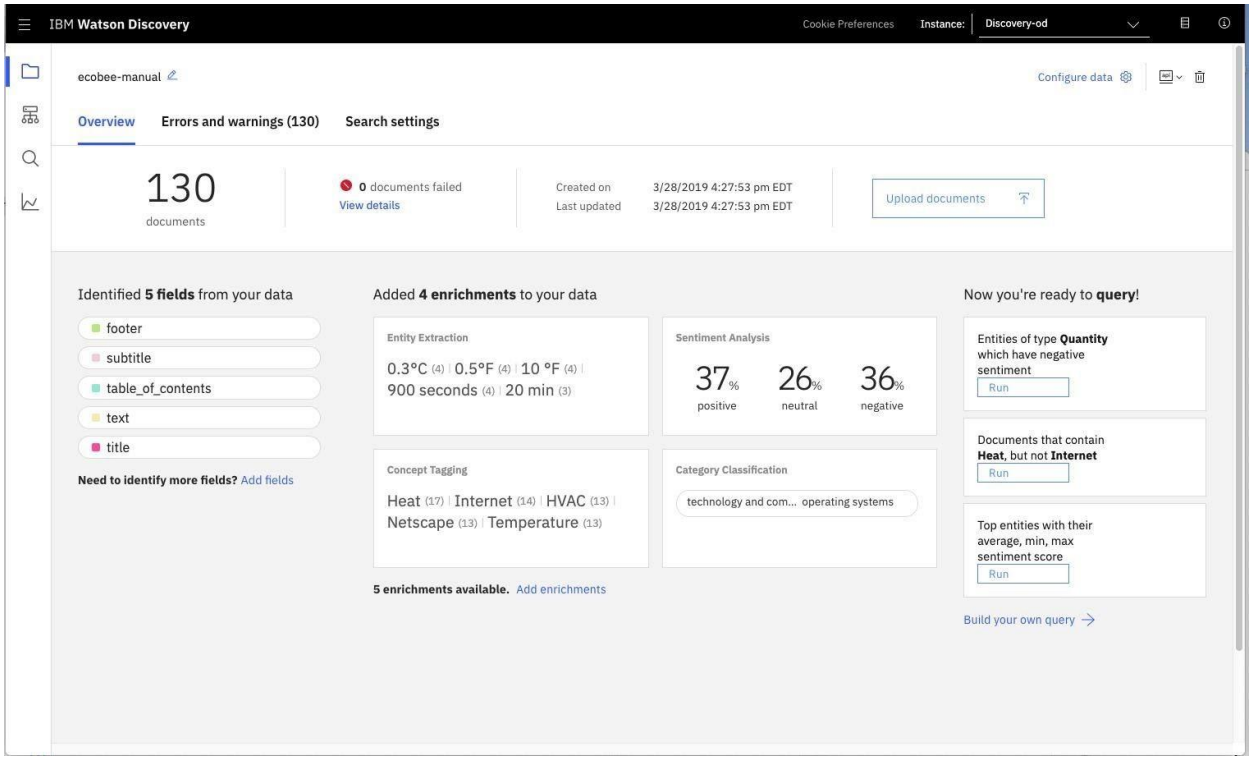
[2] Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except subtitles and text.

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

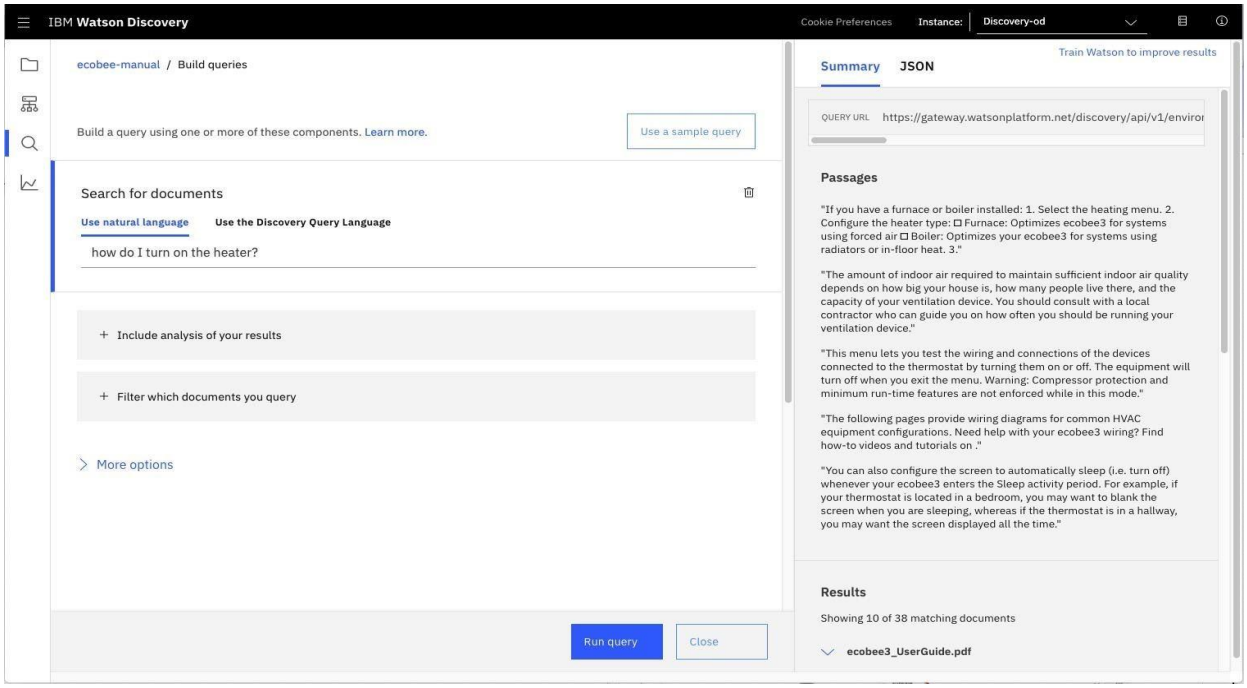
Click [4] to submit your changes.

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:



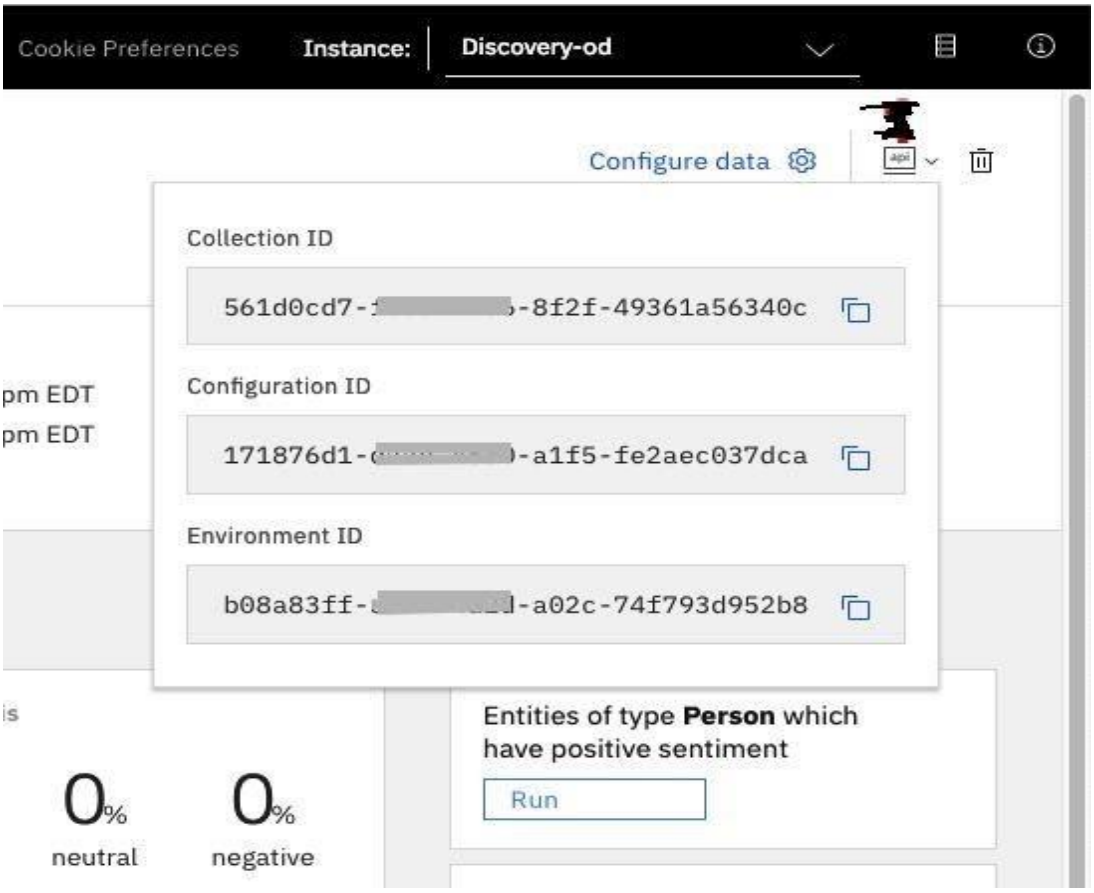
Return to the query panel (click Build your own query) and see how much better the results are.



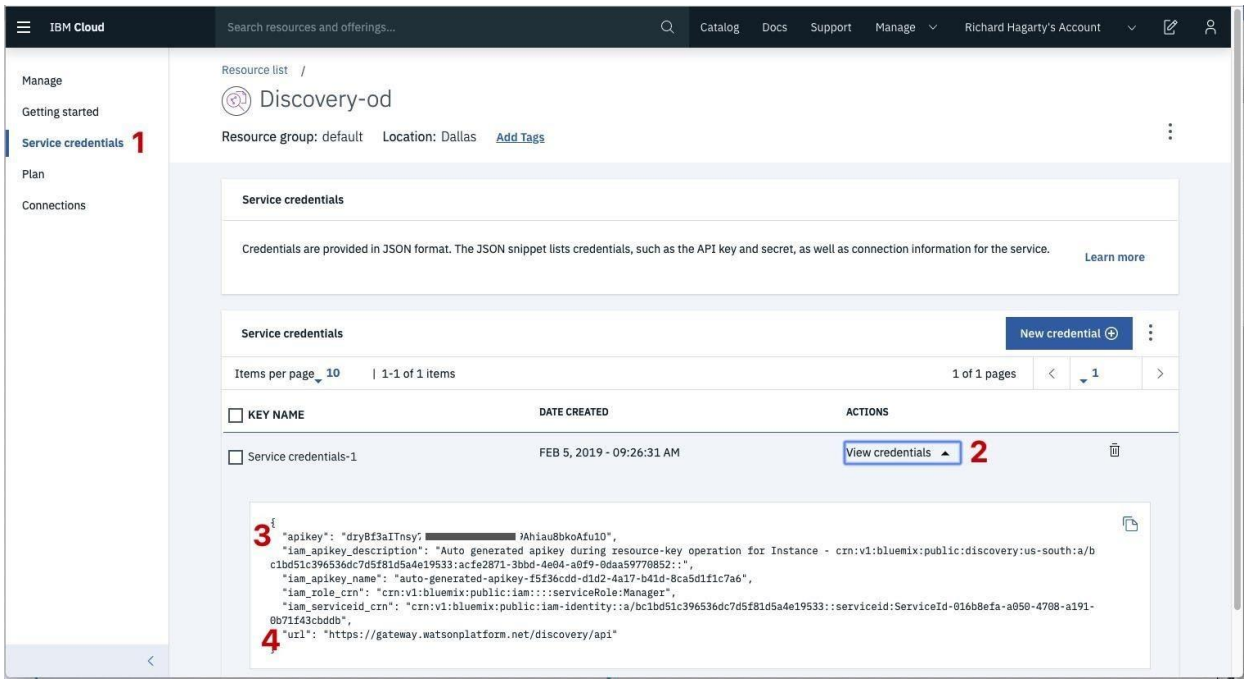
Store credentials for future use

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:

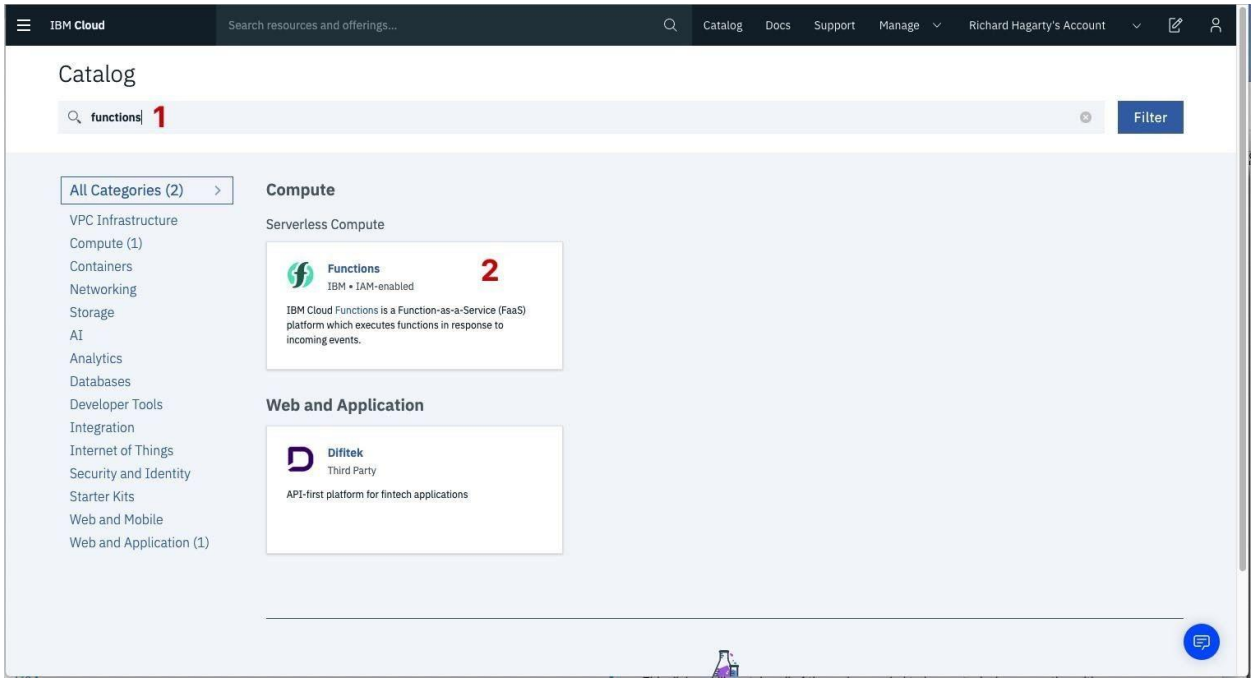


Click the View credentials [2] drop-down menu to view the IAM api key [3] and URL endpoint [4] for your service.

Create IBM Cloud Functions action

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

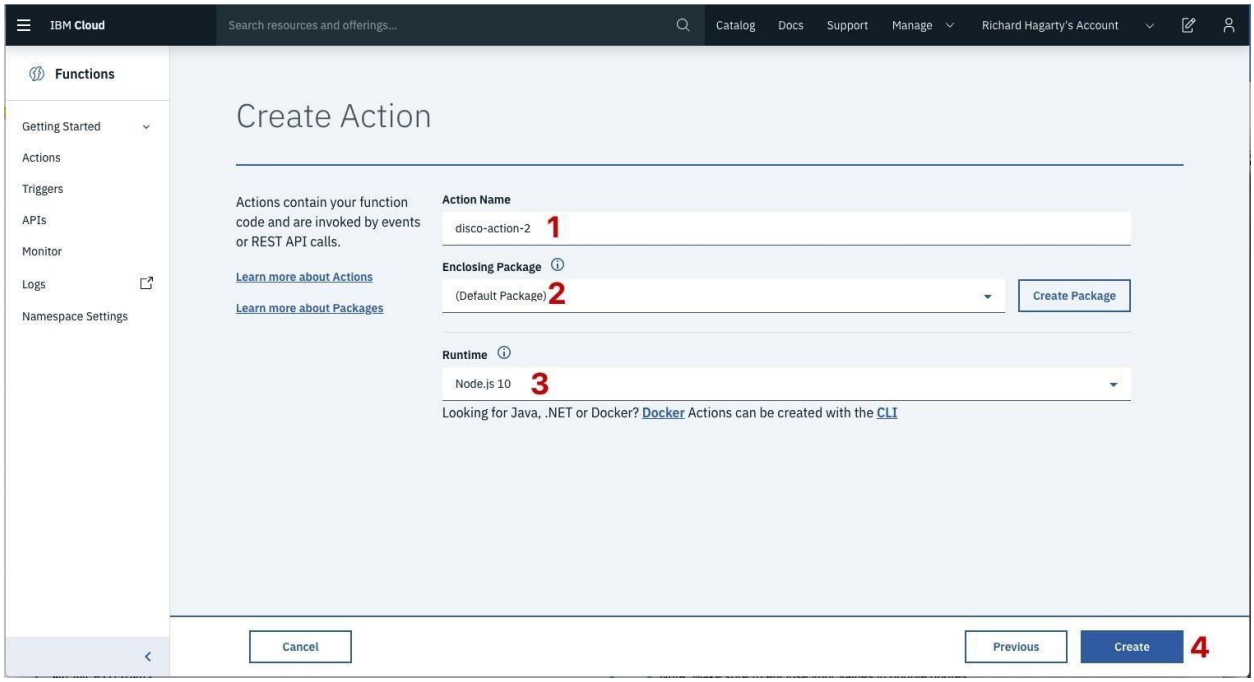
Start the IBM Cloud Functions service by selecting Create Resource from the IBM Cloud dashboard. Enter functions as the filter [1], then select the Functions card [2]:



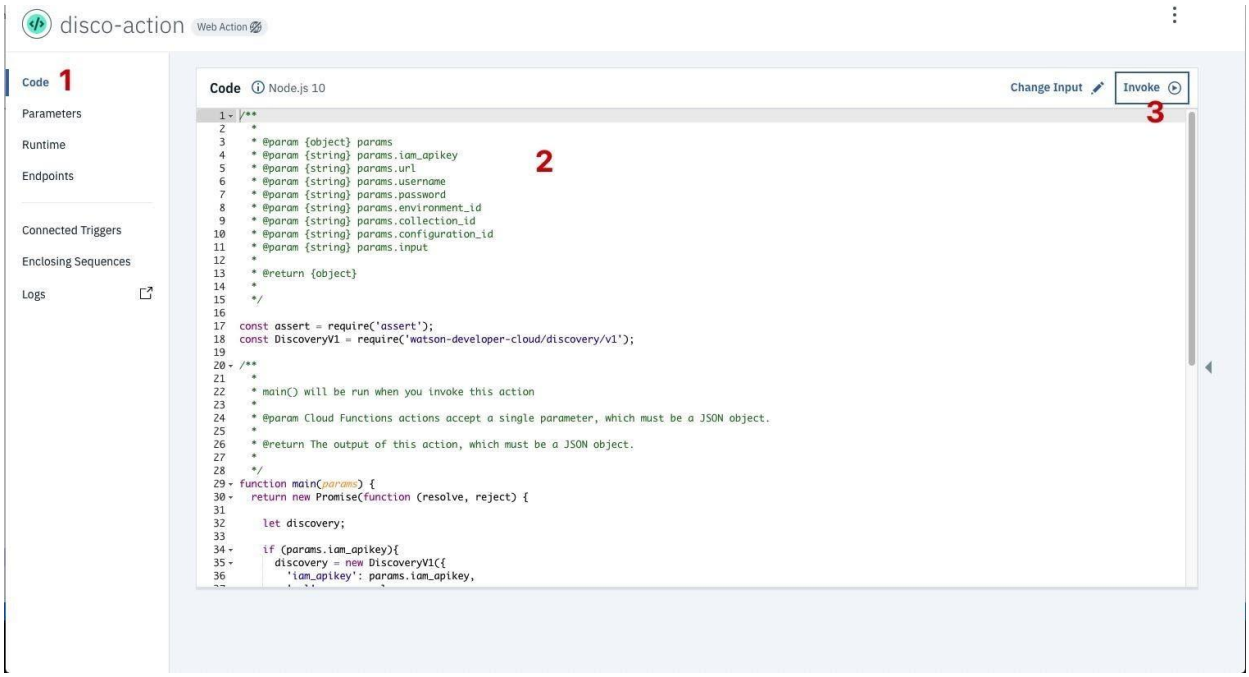
From the Functions main panel, click on the Actions tab. Then click on Create.

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]:

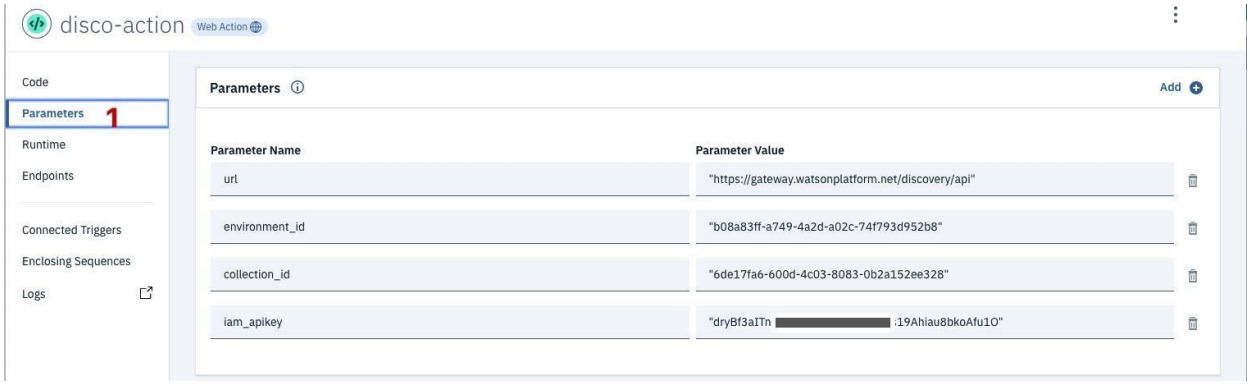


In the code editor window [2], cut and paste in the code from the disco-action.js file found in the actions directory of your local repo. The

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]:



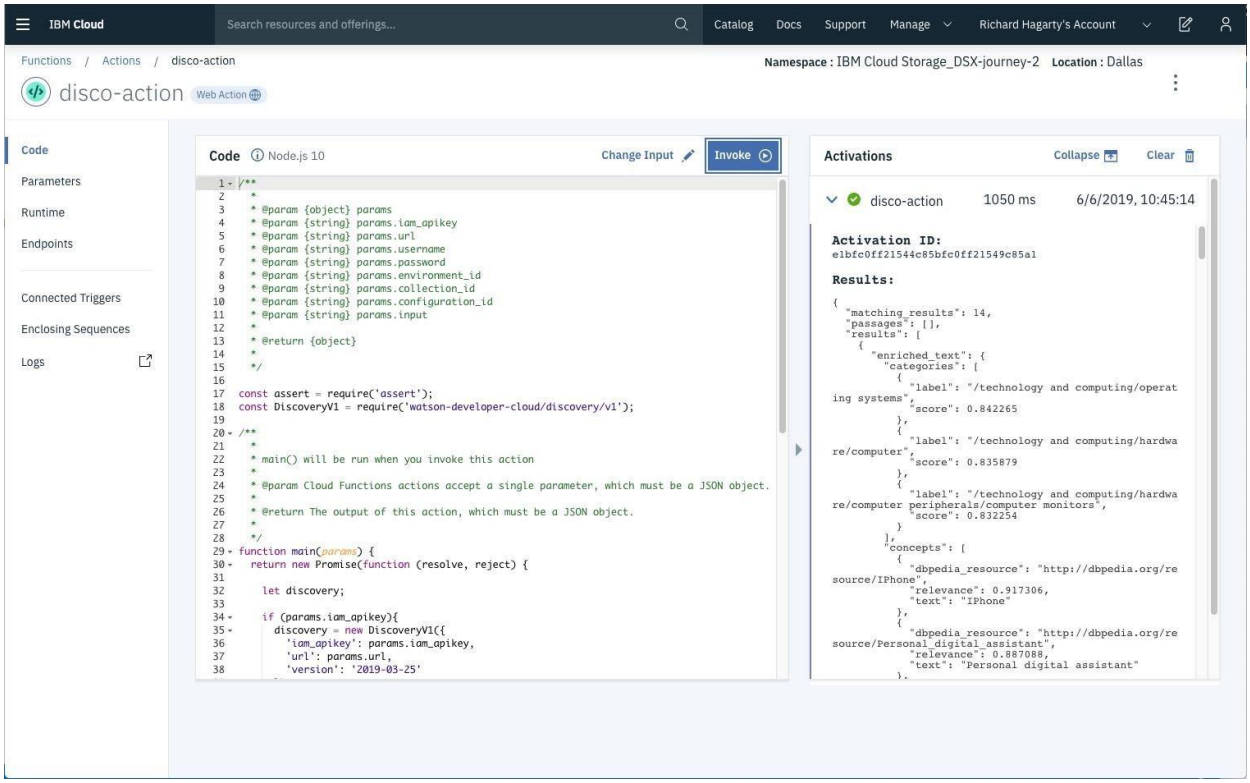
Add the following keys:

- url
- environment_id
- collection_id
- iam_apikey

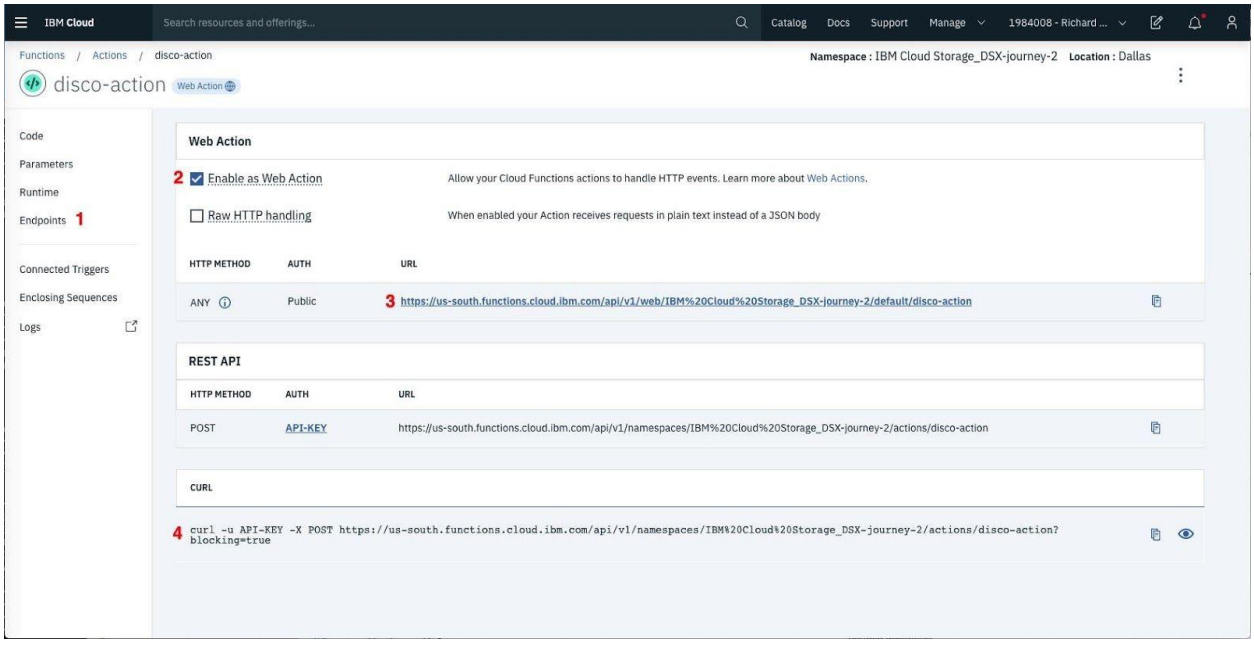
For values, please use the values associated with the Discovery service you created in the previous step.

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:



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.

Configure Watson Assistant

As shown below, 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

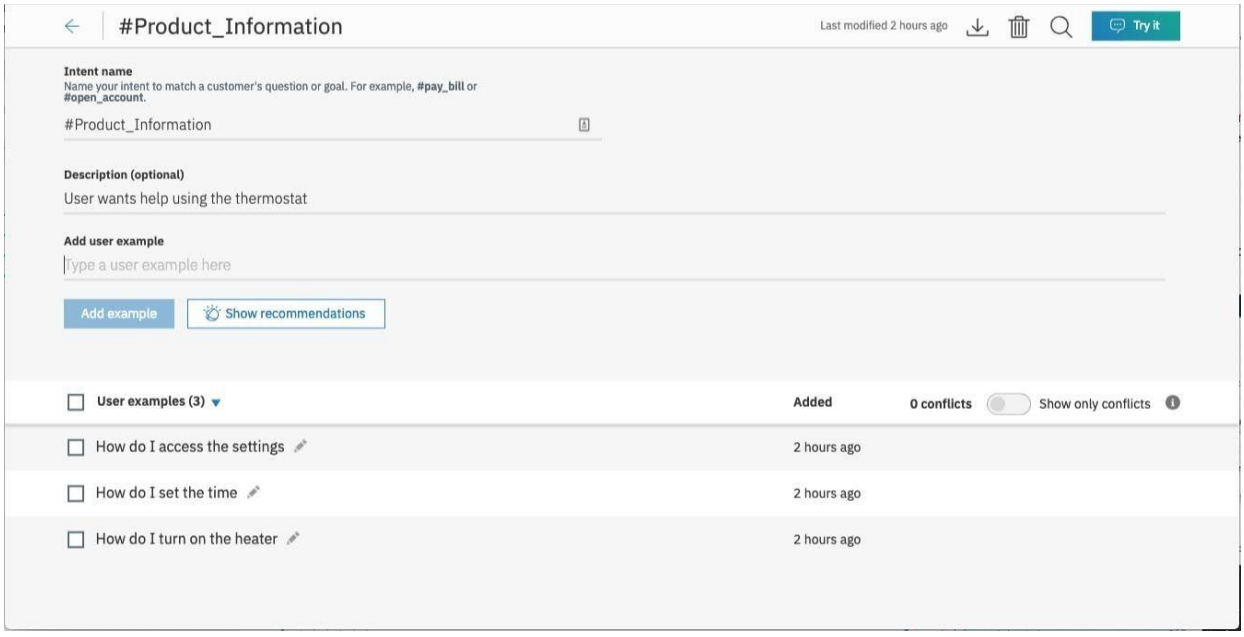
involving outside resources, so we will need to add this.

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.

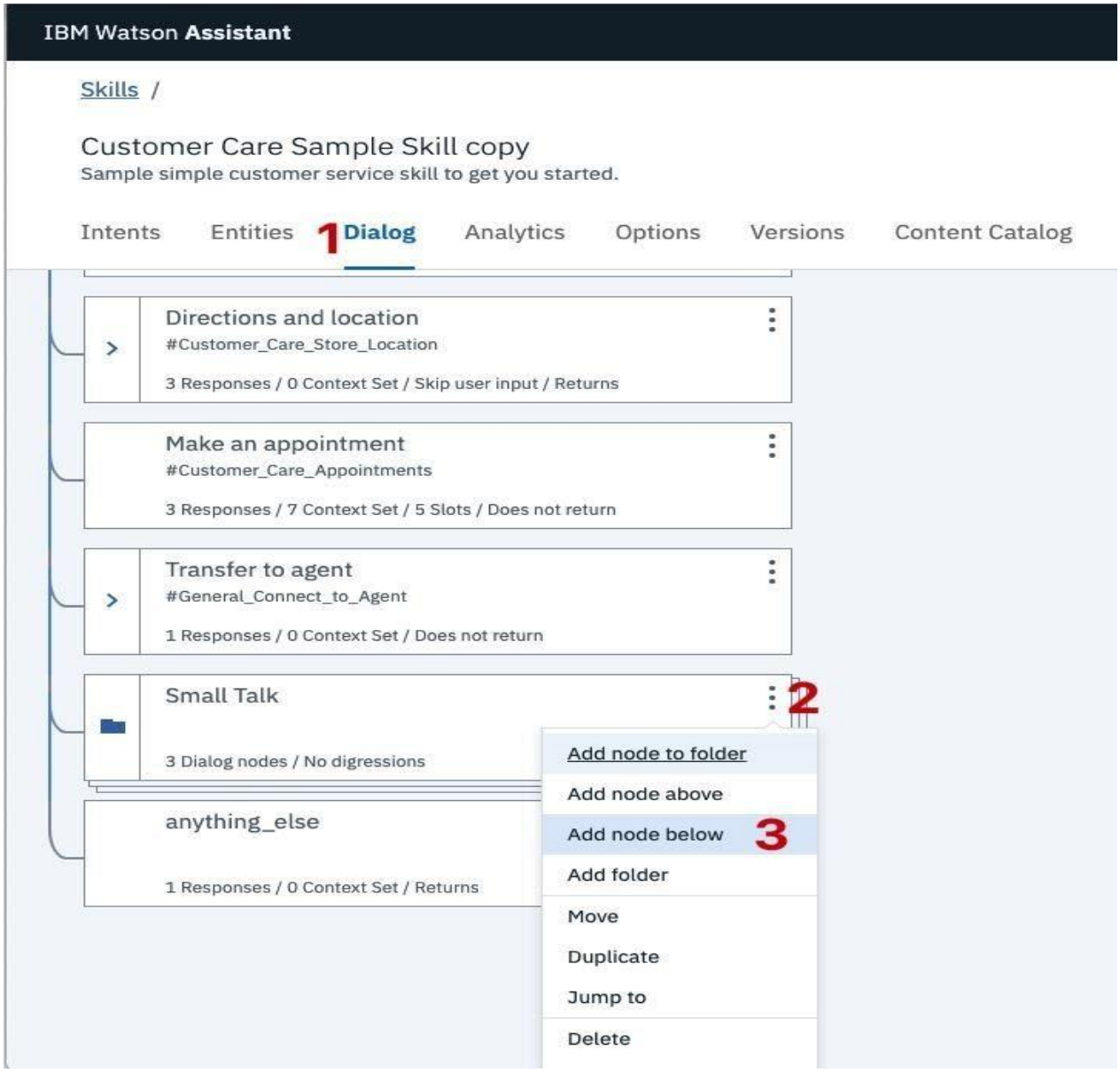
Click the Create intent button.

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

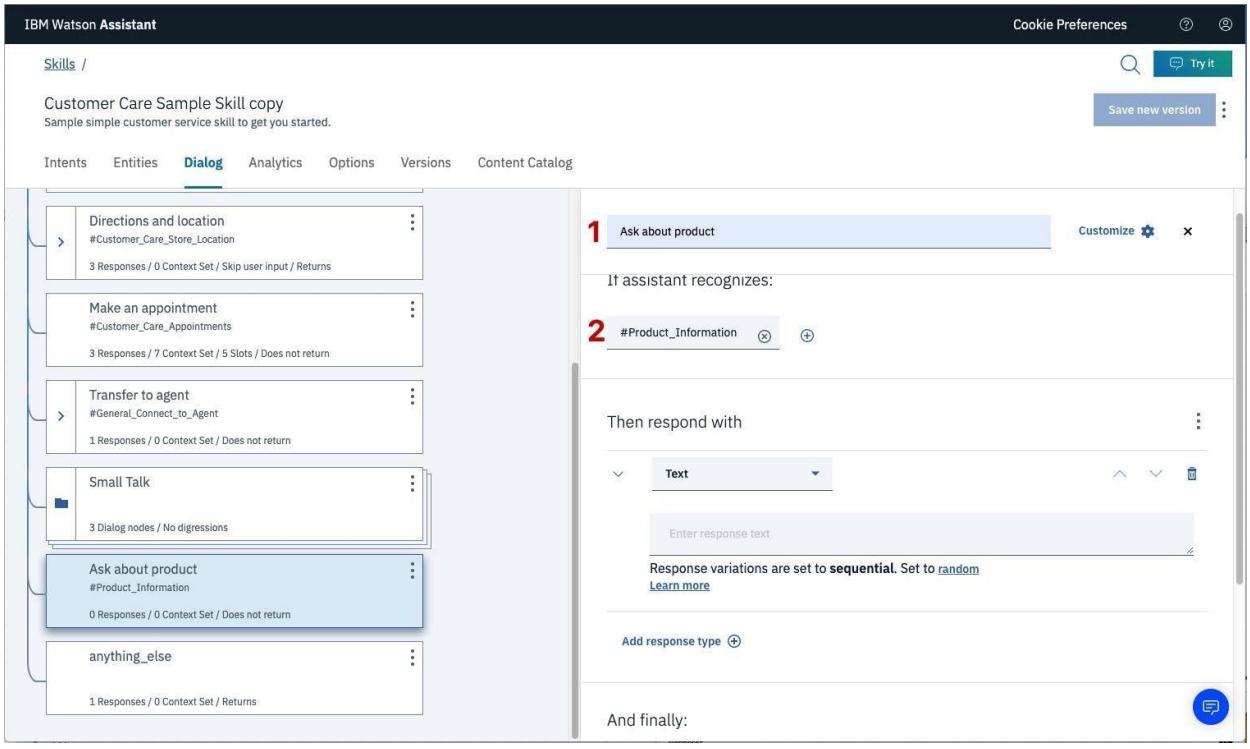


Create new dialog node

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.



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

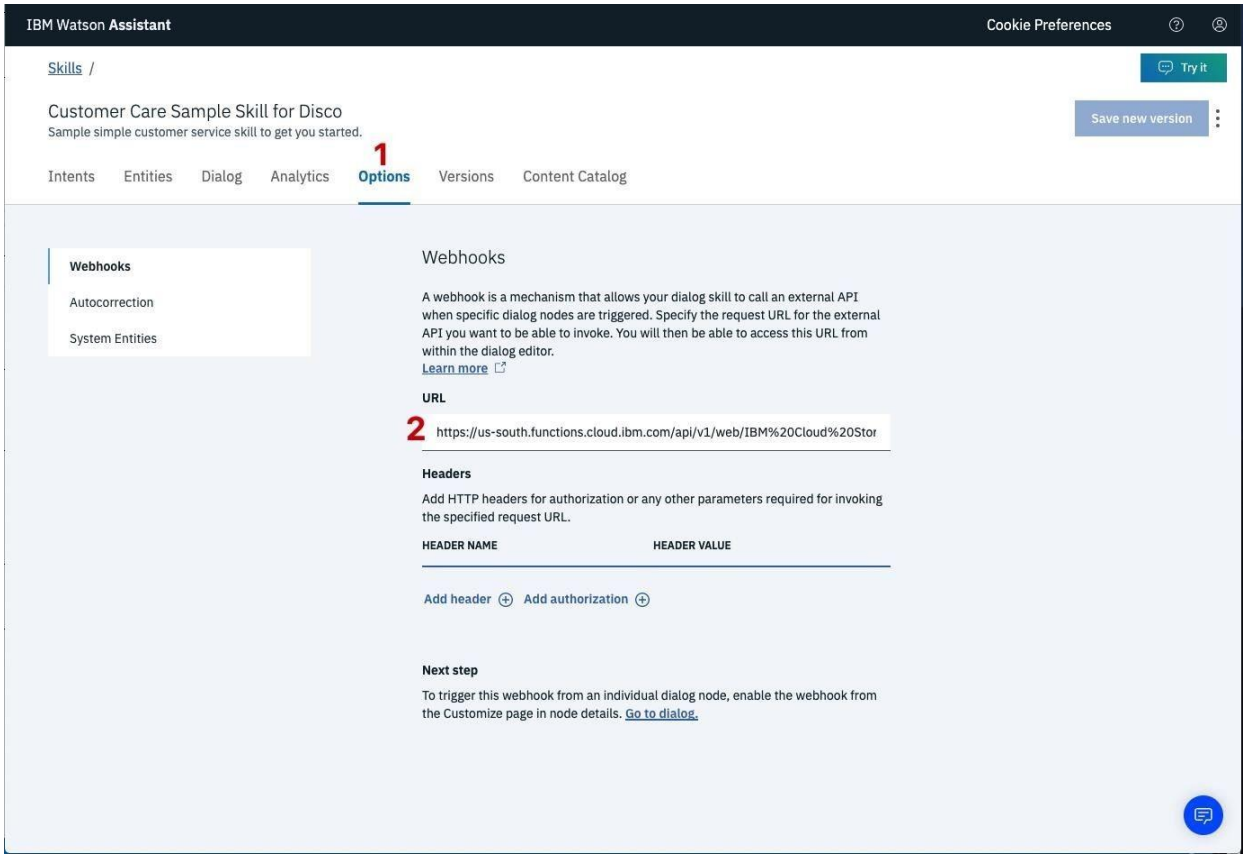


This means that if Watson Assistant recognizes a user input such as "how do I set the time?", it will direct the conversation to this node.

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]:

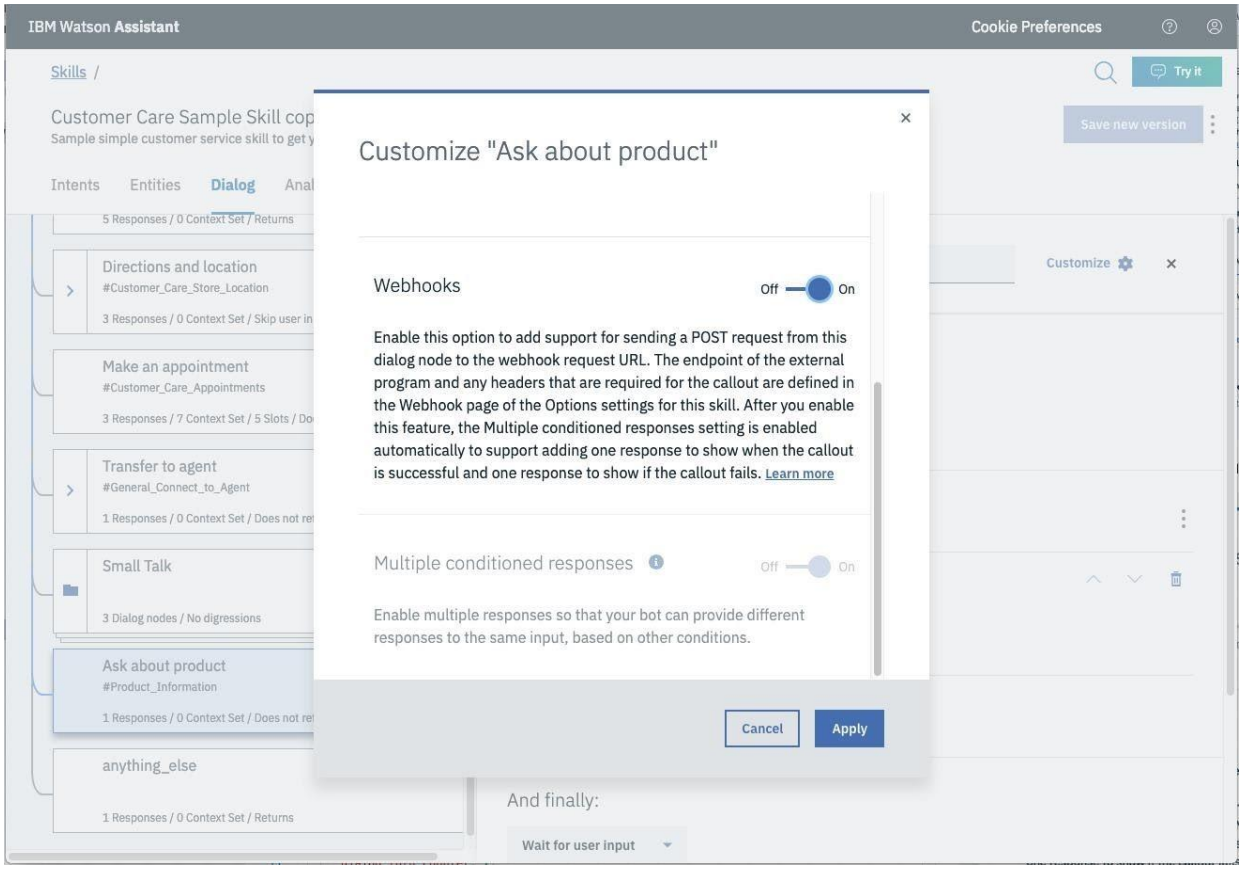


Enter the public URL endpoint for your action [2].

Important: Add .json to the end of the URL to specify the result should be in JSON format.

Return to the Dialog tab, and click on the Ask about product node. From the details

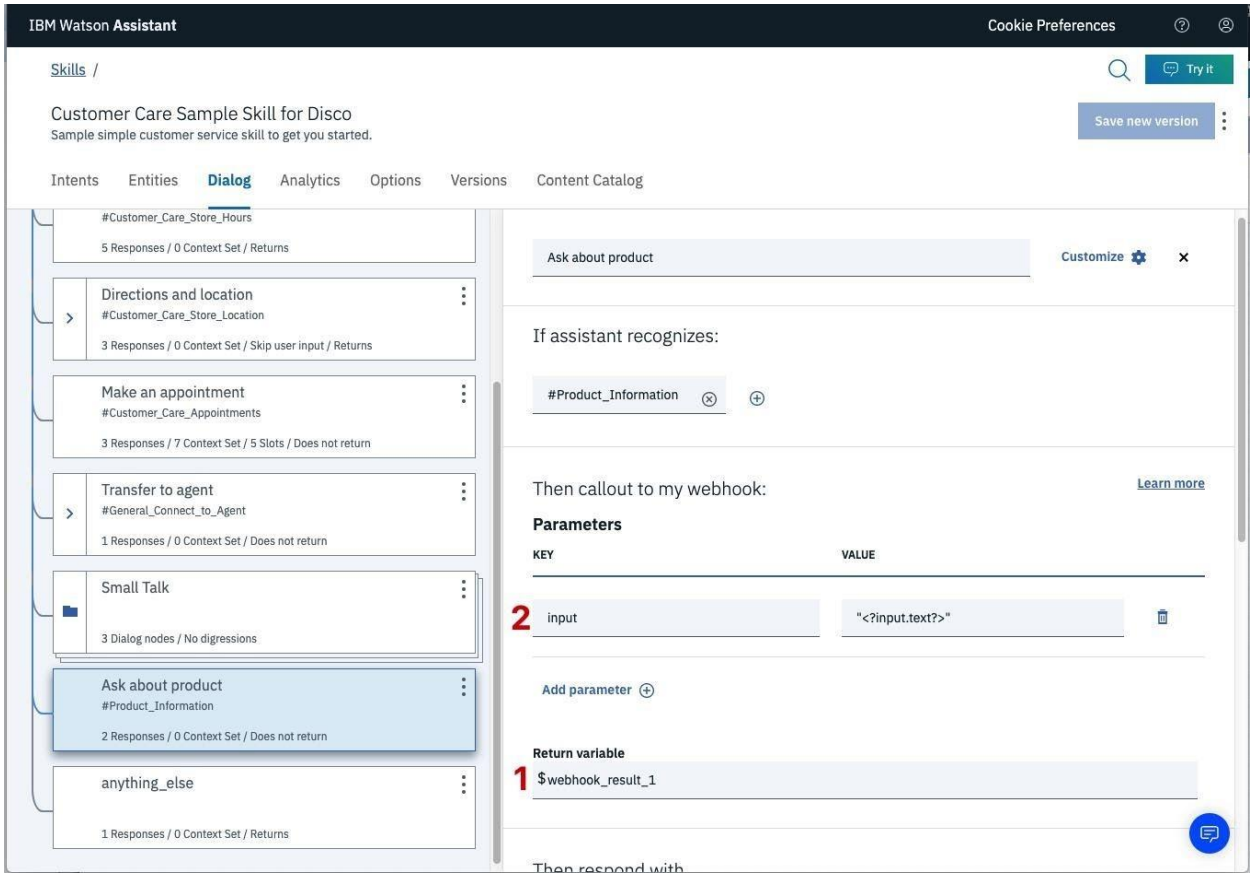
panel for the node, click on Customize, and enable Webhooks for this node:



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.



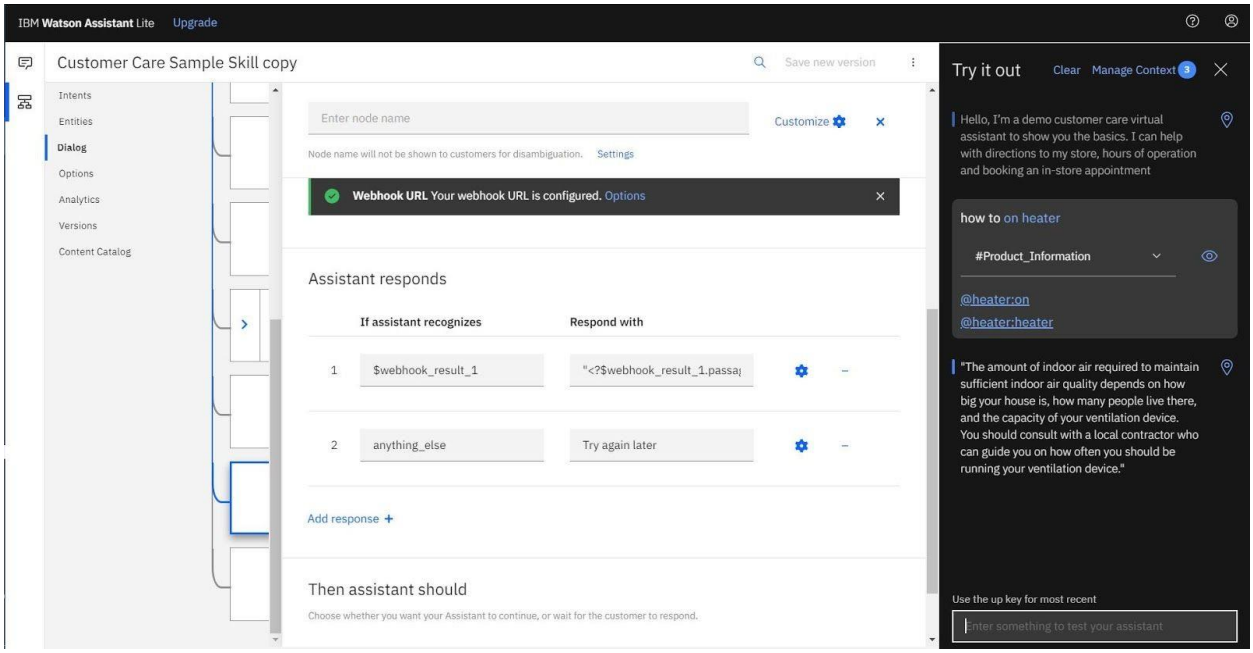
You will also need to pass in the users question via the parameter input [2]. The key needs to be set to the value:

"<?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:

Add Add “<?webhook_result_1.passages[0].passage_text?>” in response within the Assistant responds block as shown below.

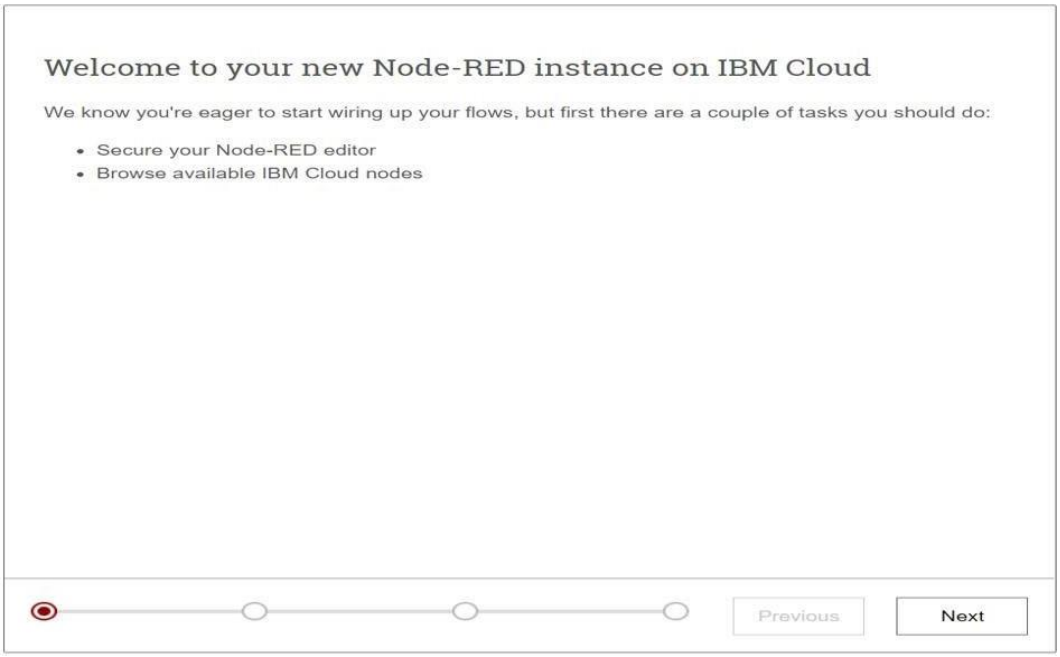


Creation of Node-RED in IBM cloud

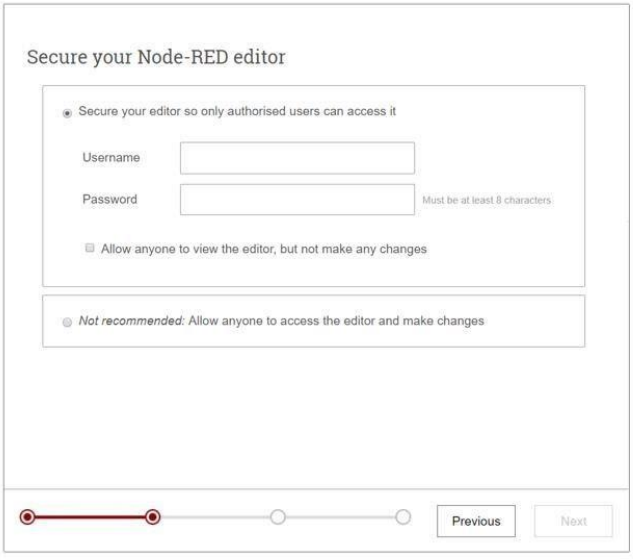
- Step-1: Login to IBM and go to the catalog
- Step-2: Search for node-red and select "Node-RED Starter " Service
- Step-3: Enter the Unique name and click on create a button

  Note: Your Node-red service is starting

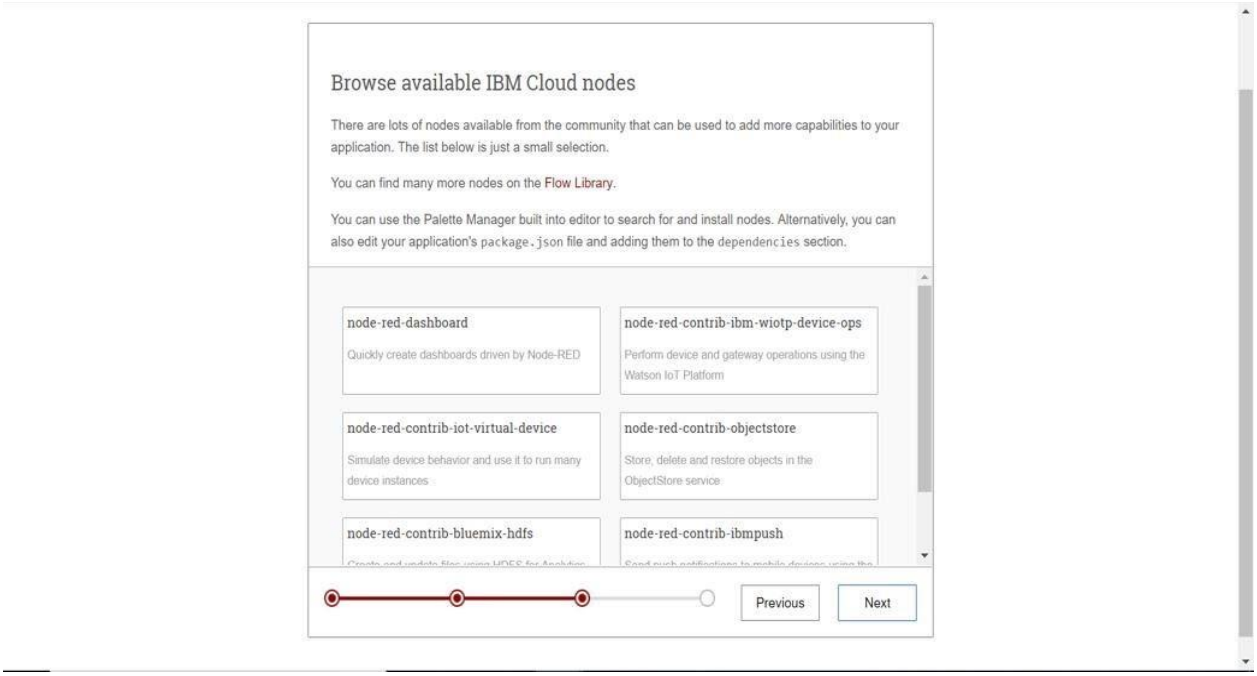
- Step – 5: We have to configure Node red for the first time. Click on next to continue



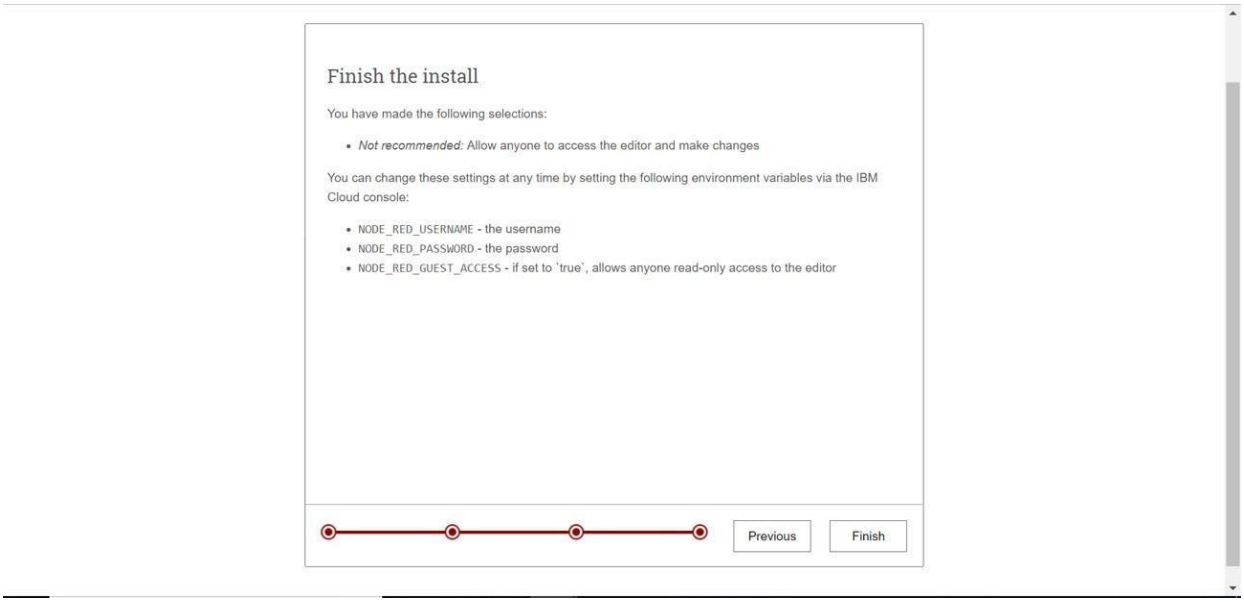
➢ Step – 6: Secure your node red editor by giving a username and password and click on Next



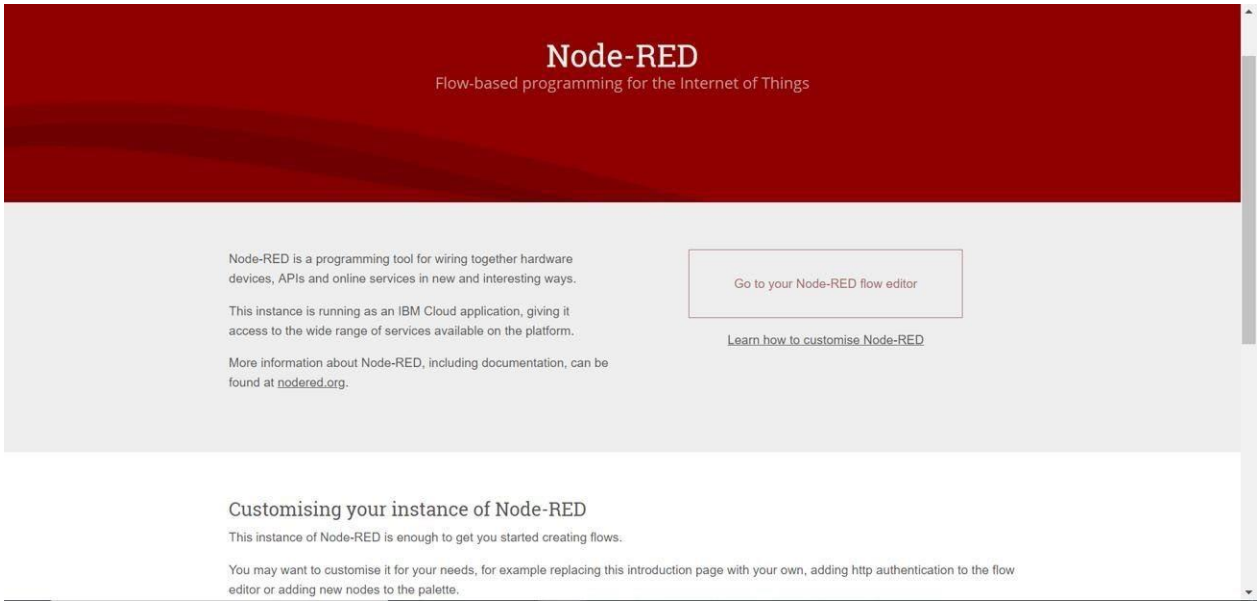
➢ Step – 7: Click Next to continue



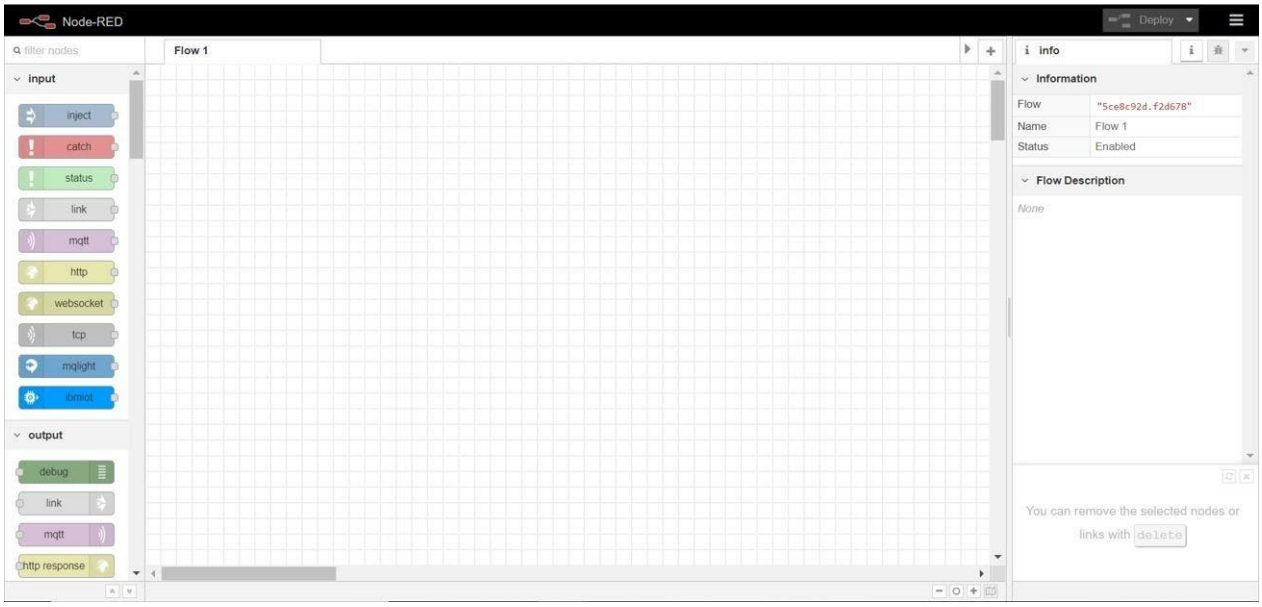
➢ Step – 8: Click Finish



➢ Step – 9: Click on Go to Node-Red flow editor to launch the flow editor



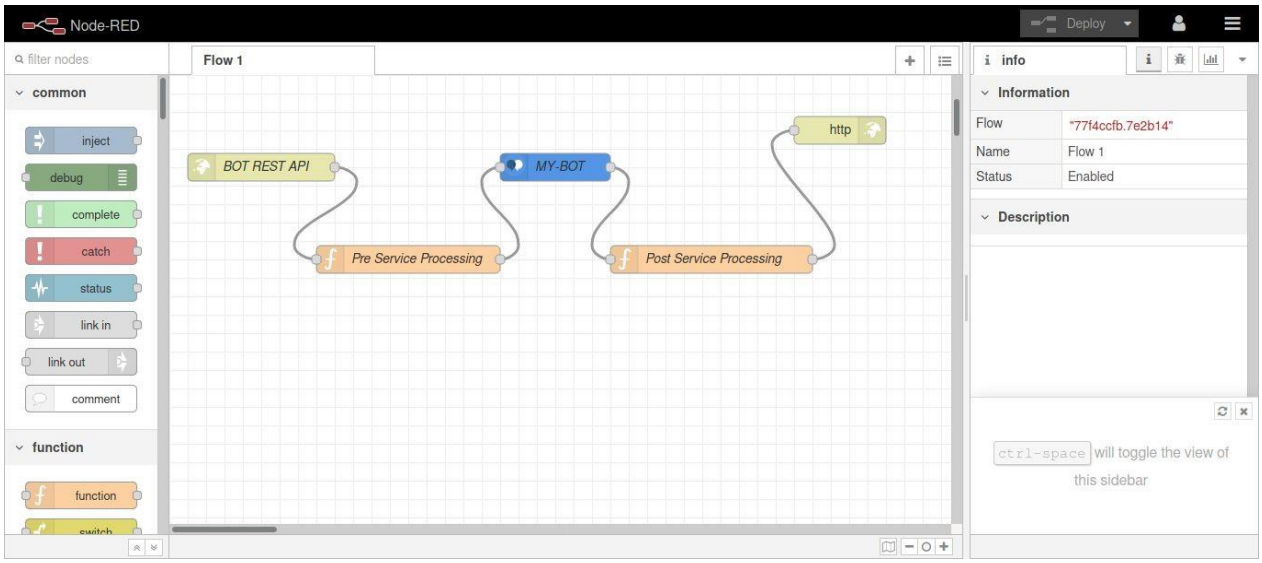
➢ Node red editor has various nodes with the respective functionality



Integration of watson assistant in Node-RED

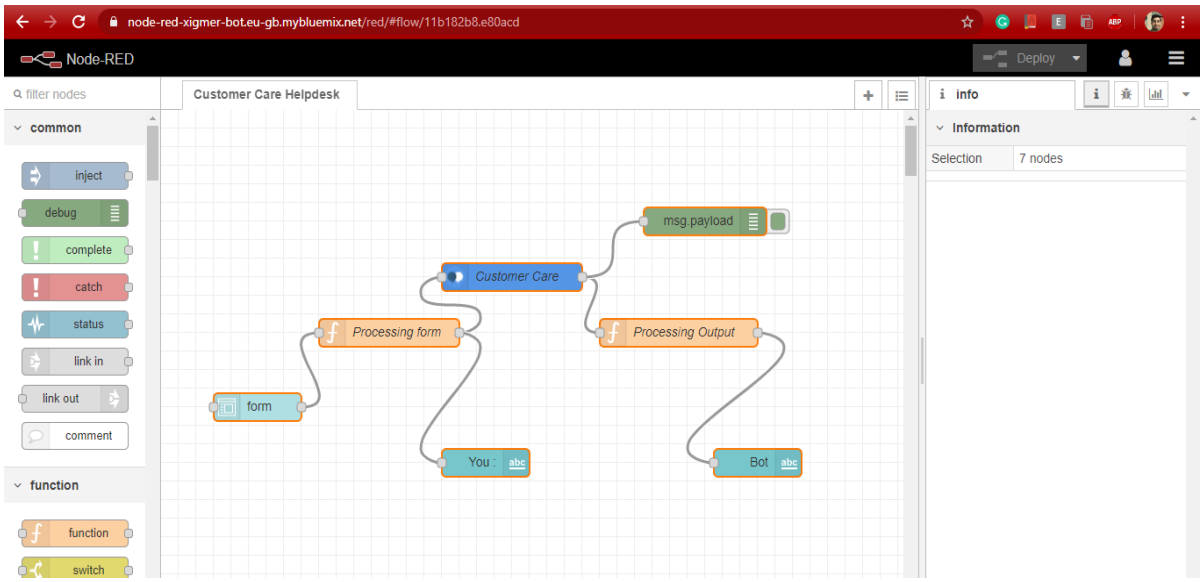
➢ Double-click on the Watson assistant node

- ➢ 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 on Done
- ➢ Drag a http-in node, http-response node into the workspace
- ➢ Drag two functions node, configure those to perform the pre and post processing of the input and output of the watson assistant node
- ➢ Connect the http-in, preprocessing, watson assistant, post processing and http-response nodes to form an API
- ➢ Connect the nodes as shown below and click on Deploy
- ➢ Drag the function node to parse the JSON data and get the bot response
- ➢ Connect the nodes as shown below and click on Deploy



We are done integrating Watson assistant service to Node-red. In the next lab, we will create a web application using Node-red for the chatbot. For creating a web application UI we need "dashboard" nodes which should be installed manually.

- ➢ Go to navigation pane and click on manage palette
- ➢ Click on install
- ➢ Search for "node-red-dashboard" and click on install and again click on install on the prompt
- ➢ The following message indicates dashboard nodes are installed, close the manage palette
- ➢ Drag a http-in, template and http-response node
- ➢ Make a request to the API and display the response in the web dashboard in the template node
- ➢ Connect those nodes



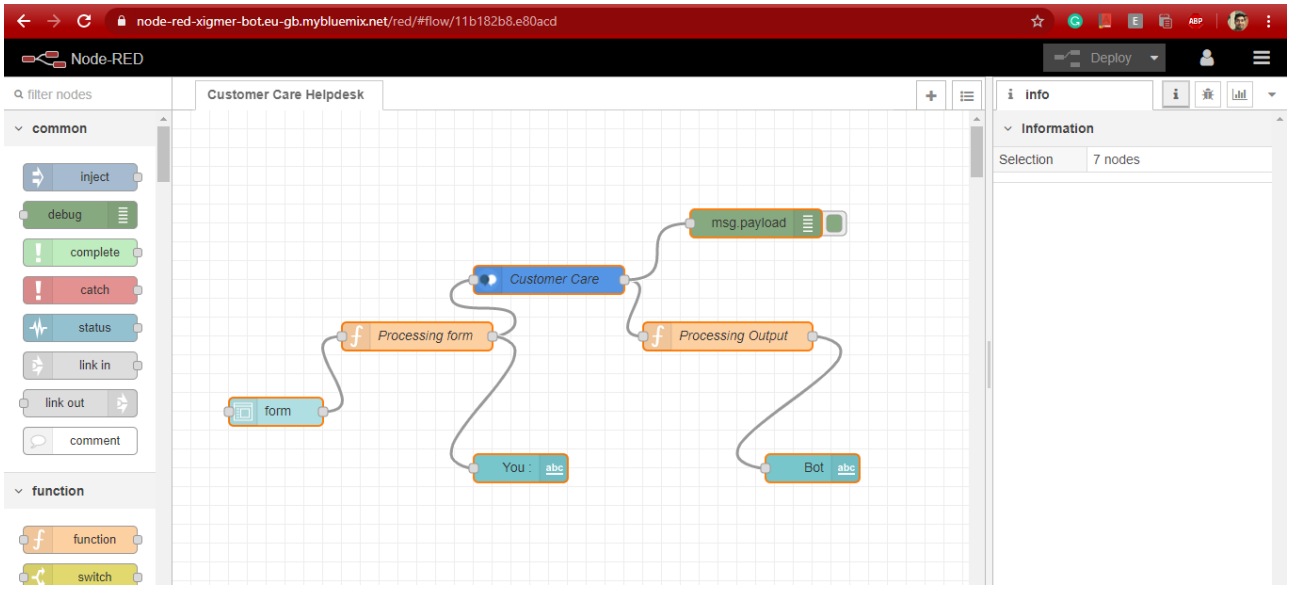
- ➢ Click on Deploy

FLOWCHART

1. Create flow and configure node:

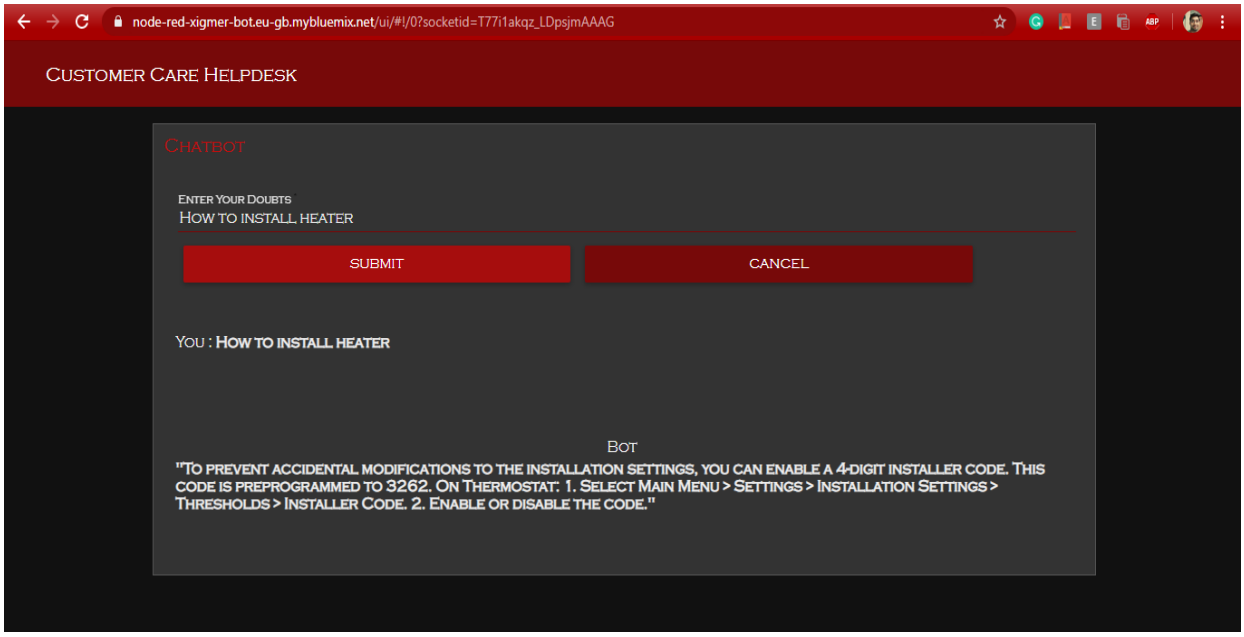
First go to manage the palette and install the dashboard. Now, create the flow with the help of following node:

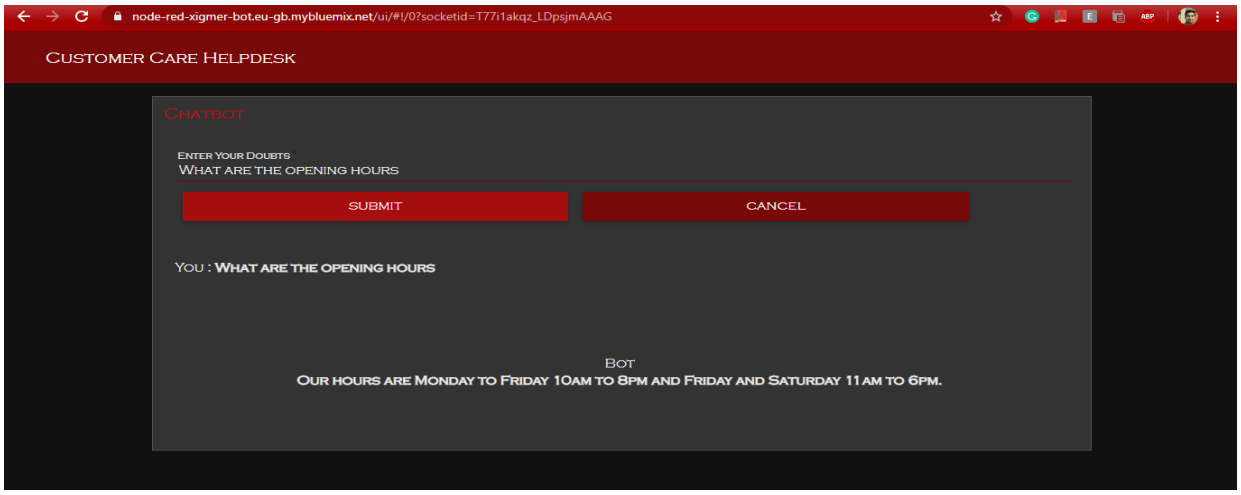
- ➢ Assistant
- ➢ Function
- ➢ Debug
- ➢ Text
- ➢ Function
- ➢ Template



RESULTS

Finally our Node-RED dashboard integrates all the components and displayed in the Dashboard UI -





ADVANTAGES & DISADVANTAGES

Advantages:

- Companies can deploy chatbots to rectify simple and general human queries .

- Reduces man power
- Cost efficient
- No need to divert calls to customer agents and customer agents can look at other works.

Disadvantages:

- Some times chatbot can mislead customers
- Giving the same answer for different sentiments.
- Sometimes cannot connect to customer sentiments and intentions

APPLICATIONS

- It can deploy in popular social media applications like facebook,slack,telegram.
- Chatbot can deploy any website to clarify basic doubts of viewer

CONCLUSION

By doing the above procedure and all we successfully created an Intelligent help desk smart chatbot using Watson assistant, Watson discovery, Node-RED and cloud-functions.

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 scope of this project.

BIBLIOGRAPHY

APPENDIX

NODE-RED FLOW


```
[{"id":"11b182b8.e80acd","type":"tab","label":"Customer Care Hel
pdesk","disabled":false,"info":""},{"id":"ef833f2a.47fbc","type"
:"ui_form","z":"11b182b8.e80acd","name":"","label":"","group":"4
3177b96.68f8d4","order":1,"width":0,"height":0,"options":[{"labe
l":"Enter Your Doubts","value":"text","type":"text","required":t
rue,"rows":null}],"formValue":{"text":""},"payload":"","submit":
"submit","cancel":"cancel","topic":"","x":110,"y":260,"wires":[[
"99302f4.0c955d"]]},{"id":"99302f4.0c955d","type":"function","z"
:"11b182b8.e80acd","name":"","func":"msg.payload = msg.payload.t
ext;\nreturn msg;","outputs":1,"noerr":0,"x":230,"y":180,"wires"
:[["2a47b2b2.839b2e","2c9b9387.3d1edc"]]},{"id":"7b027860.8833a8
","type":"function","z":"11b182b8.e80acd","name":"","func":"msg.
payload = msg.payload.output.text[0];\nreturn msg;","outputs":1,
"noerr":0,"x":550,"y":180,"wires":[["b604915c.053e1"]]},{"id":"2
a47b2b2.839b2e","type":"watson-conversation-
v1","z":"11b182b8.e80acd","name":"Customer Care","workspaceid":"
d60d94fc-591c-422e-b5e1-
6090098a76e7","multiuser":false,"context":true,"empty-
payload":false,"service-endpoint":"","timeout":"","optout-
learning":false,"x":400,"y":120,"wires":[["7b027860.8833a8","bdc
1c22c.4db29"]]},{"id":"2c9b9387.3d1edc","type":"ui_text","z":"11
b182b8.e80acd","group":"43177b96.68f8d4","order":2,"width":0,"he
ight":0,"name":"","label":"You :","format":"{{msg.payload}}","la
yout":"row-
left","x":370,"y":320,"wires":[]},{"id":"bdc1c22c.4db29","type":
"debug","z":"11b182b8.e80acd","name":"","active":true,"tosidebar
":true,"console":false,"tostatus":false,"complete":"false","x":6
20,"y":60,"wires":[]},{"id":"b604915c.053e1","type":"ui_text","z
":"11b182b8.e80acd","group":"43177b96.68f8d4","order":3,"width":
0,"height":0,"name":"","label":"Bot","format":"{{msg.payload}}",
```

```
"layout":"col-
center","x":680,"y":320,"wires":[]},{"id":"43177b96.68f8d4","typ
e":"ui_group","z":"","name":"Chatbot","tab":"e4e6162b.432e28","o
rder":1,"disp":true,"width":16,"collapse":false},{"id":"e4e6162b
.432e28","type":"ui_tab","z":"","name":"Customer Care Helpdesk",
"icon":"dashboard","disabled":false,"hidden":false}]
```


Cloud function Node.js 10 code for discovery integration webhook generation:

```
/**
 *
 * @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);
    });
  });
}
```