PROJECT REPORT

Intelligent Customer Help Desk with Smart Document Understanding

Category: Artificial Intelligence Developer

Application ID: SPS_APL_20200001112
Project ID: SPS_PRO_99
Internship at SmartInternz

Saundarya Kumar
sksaundarya10@gmail.com

INTRODUCTION

1

1.1 Overview

1.2 Purpose

2 LITERATURE SURVEY

2.1 Existing problem

2.2 Proposed solution

3 THEORITICAL 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 BIBILOGRAPHY

APPENDIX

A. Source code

B. Reference

1. INTRODUCTION

1.1 Overview

We use the typical customer care chatbot experience but instead of relying on predefined responses, our dialog will provide a hook that can call out to other IBM Watson services for additional sources of information. In our case, it will be an owner's manual that has been uploaded into Watson Discovery.

1.2 Purpose

The purpose is to Enhance the customer helpdesks with Smart Document Understanding using webhooks in Watson Assistant.

2. LITERATURE SURVEY

2.1 Existing Problem

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.

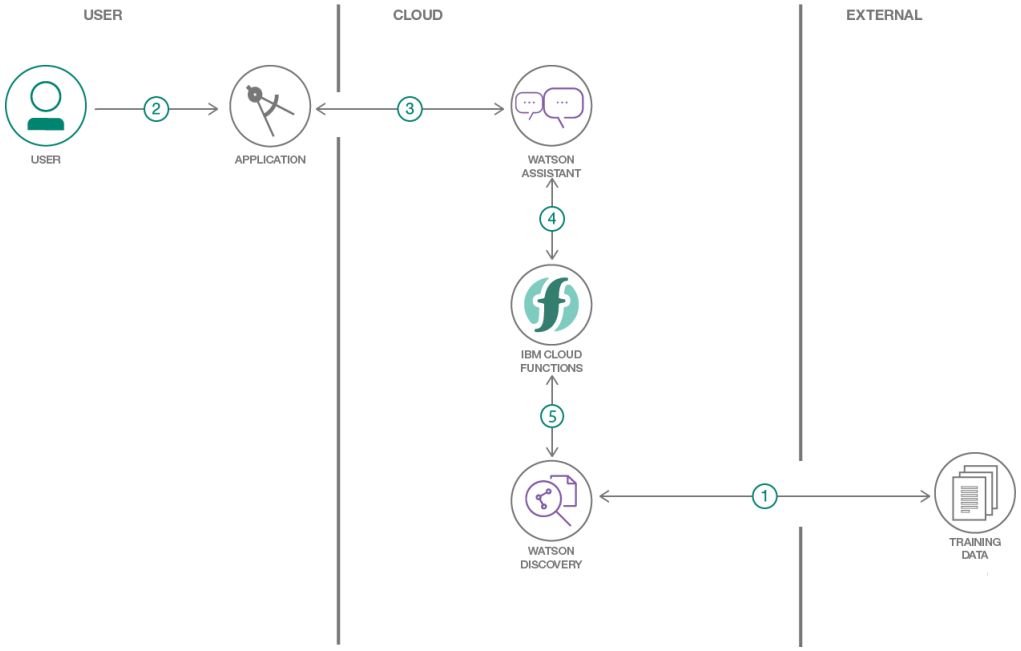
2.2 Proposed solution

In this project, 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 now, 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.

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.

3. THEORITICAL ANALYSIS

3.1 Block Diagram



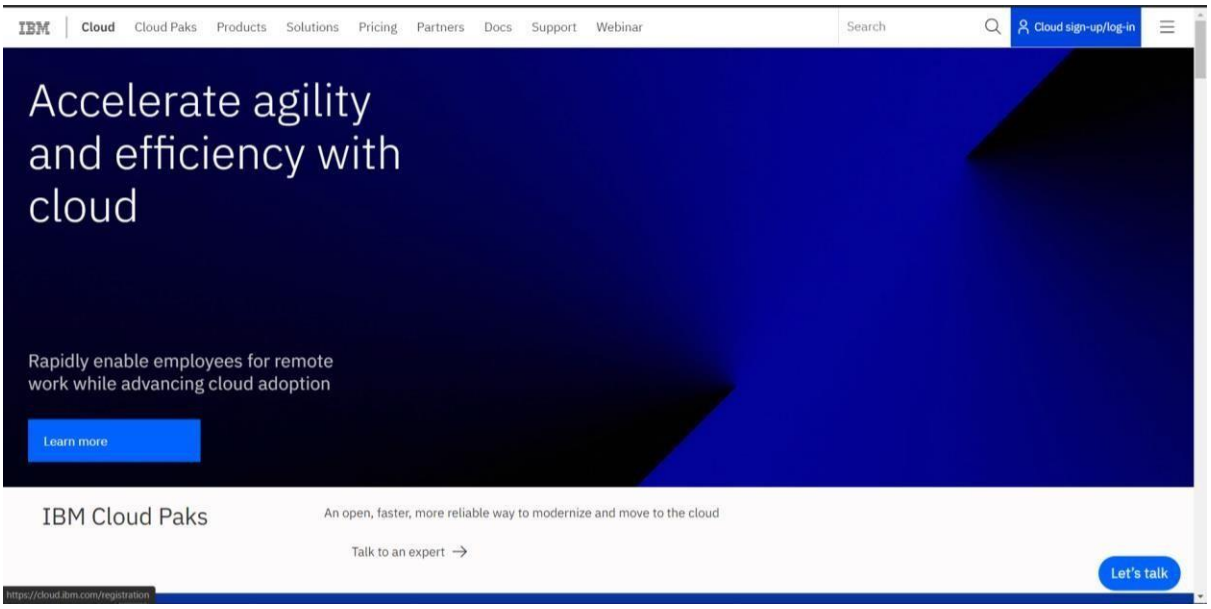
3.2 Hardware / Software designing

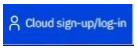
1. Create IBM Cloud Services
2. Configure Watson Discovery
3. Create IBM Cloud Functions action
4. Configure Watson Assistant
5. Build Node-RED Flow to Integrate All Services
6. Configure the nodes and Build A Web Dashboard in Node-RED
7. Deploy and Run the application

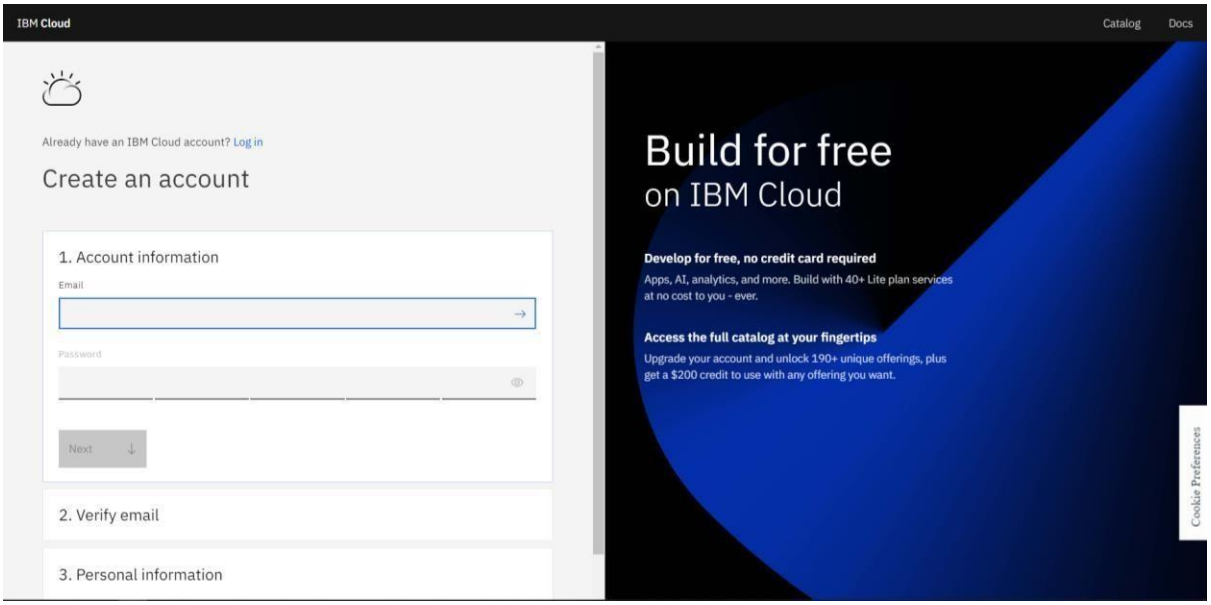
4. EXPERIMENTAL INVESTIGATIONS

1. Create IBM Cloud Services

To Create IBM Cloud, go to https://www.ibm.com/cloud



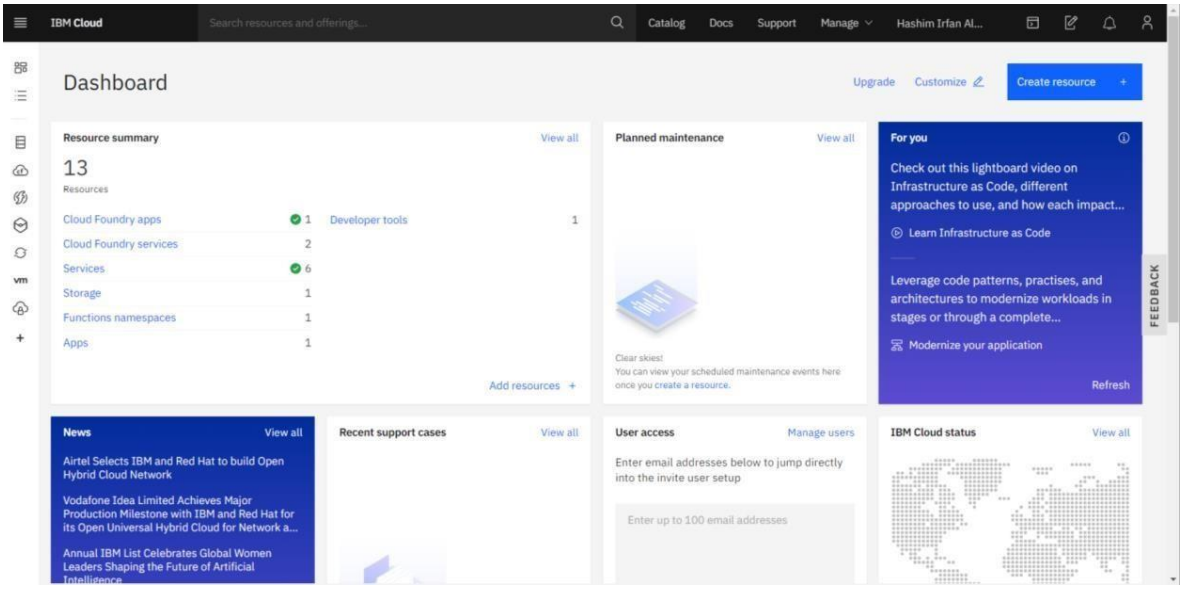
Click on  to sign up or login to IBM Cloud.

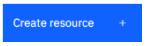


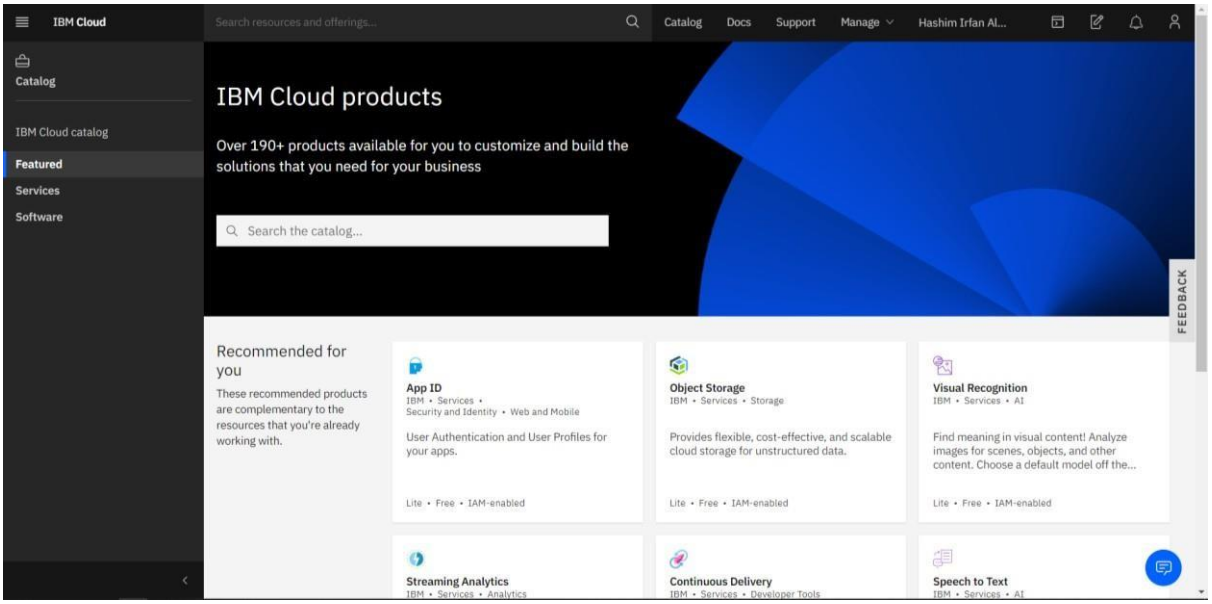
For Cloud Sign-Up: Follow the steps on the screen and fill in all the required details to create a new cloud account

For Cloud Log In, click on Already have an IBM Cloud account? Log in and Log in to your cloud account.

After Logging in, you can see the IBM Cloud Dashboard.



To Create any Resource (Services/Apps/etc), click on 

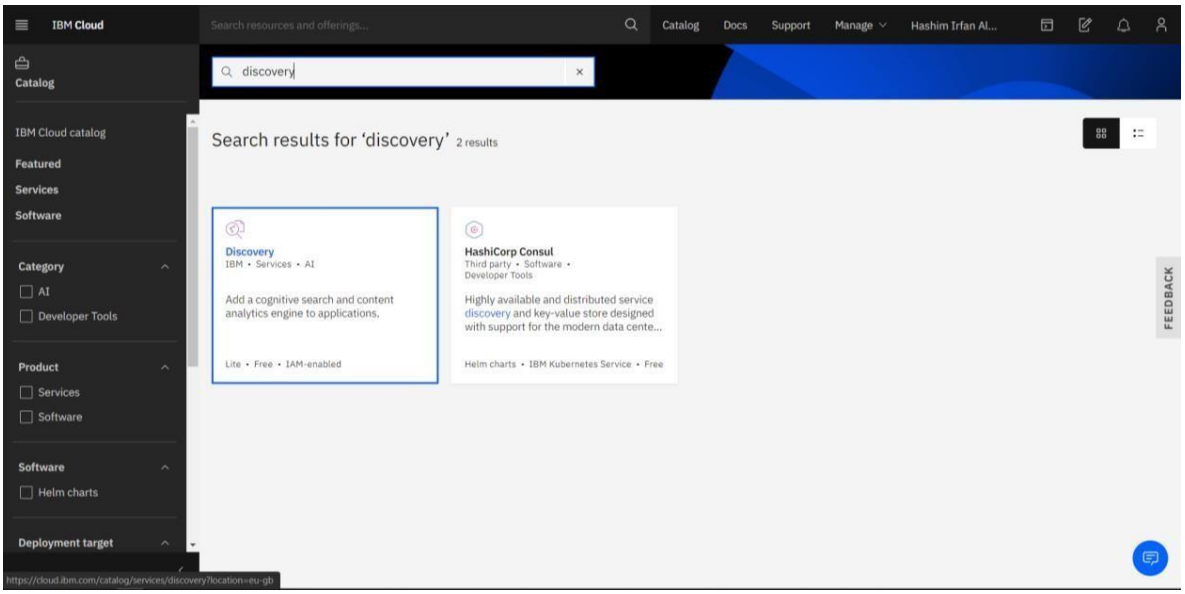


Using the search box, we can find the service we want.

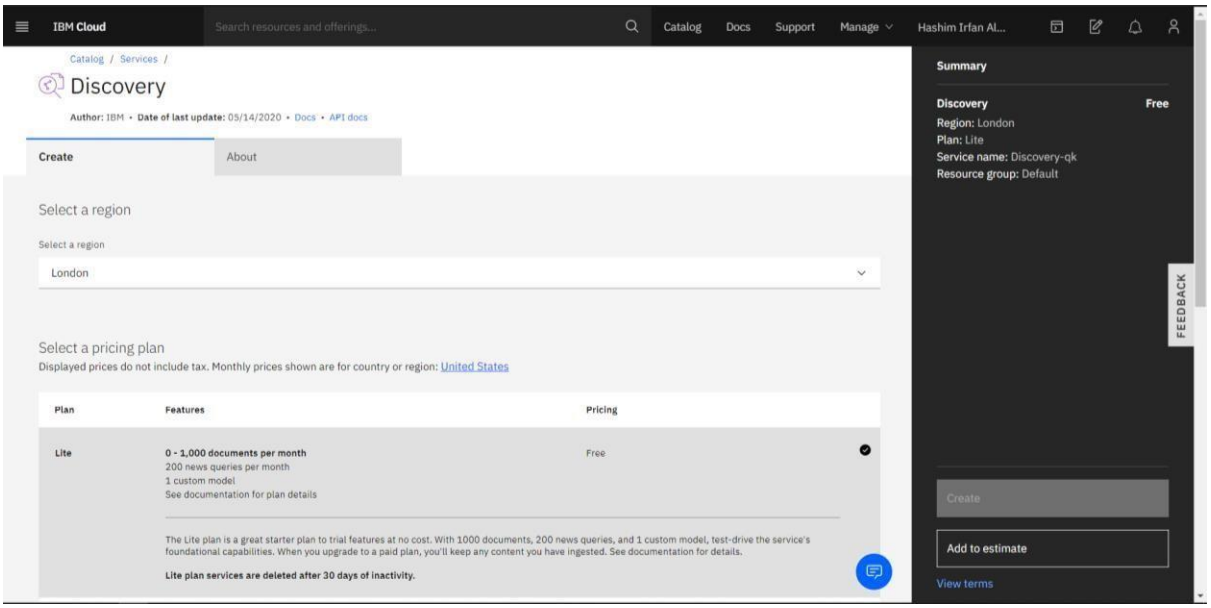
For this project, we need to Create the following services:

1. Watson Discovery
2. Watson Assistant

1. To create a Watson Discovery Service, search for Discovery in the search box



Click on 

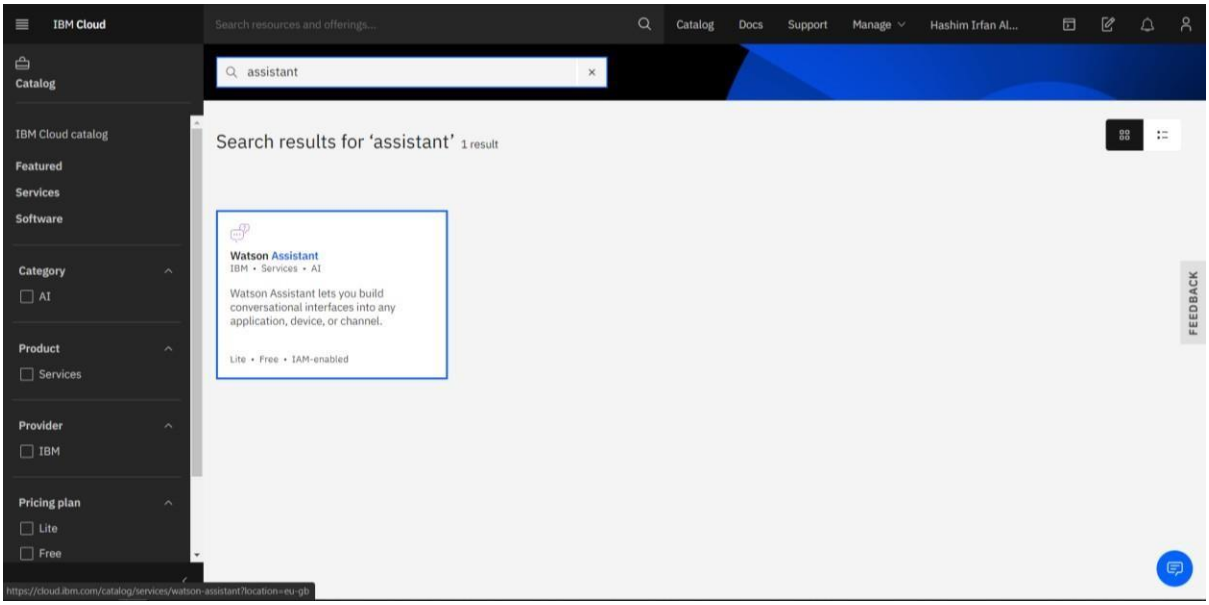


Select a region, select a plan, configure your service (Service name, etc) and click Create.

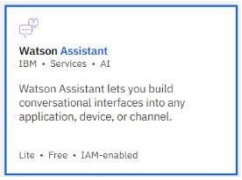
Your Watson Discovery service is created successfully.

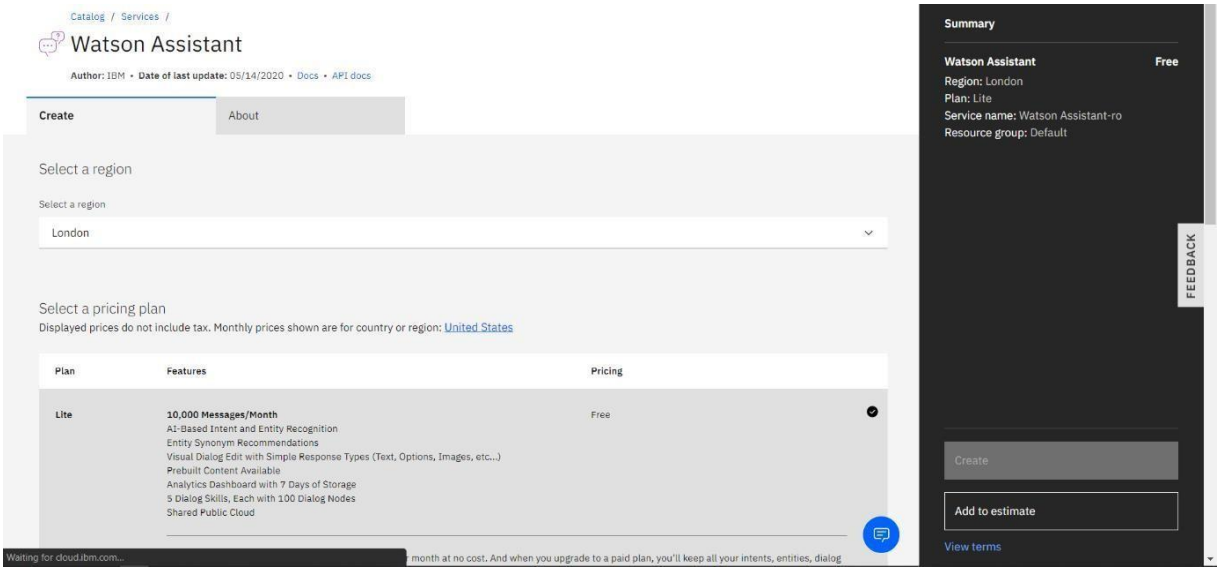
(If you are on Lite Plan, you can have only one instance per service)

2. To create a Watson Assistant Service, search for Assistant in the search box



Click on



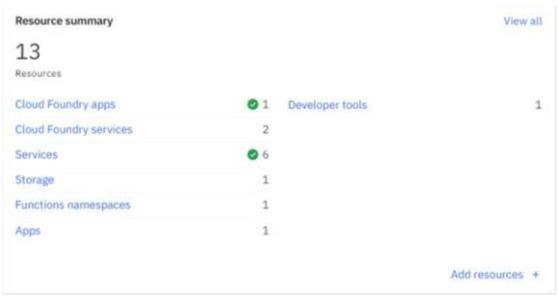


Select a region, select a plan, configure your service (Service name, etc) and click Create.

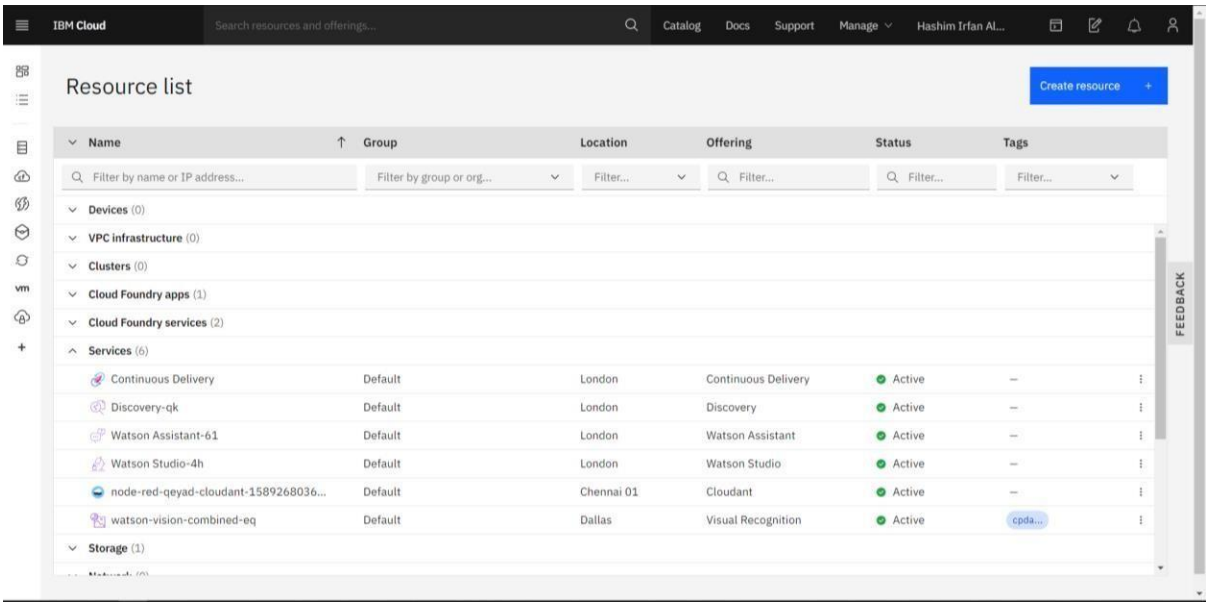
Your Watson Assistant service is created successfully.

(If you are on Lite Plan, you can have only one instance per service)

To check whether you have correctly configured the services, go back to the IBM Dashboard and click on View All from the Resource Summary Tab.



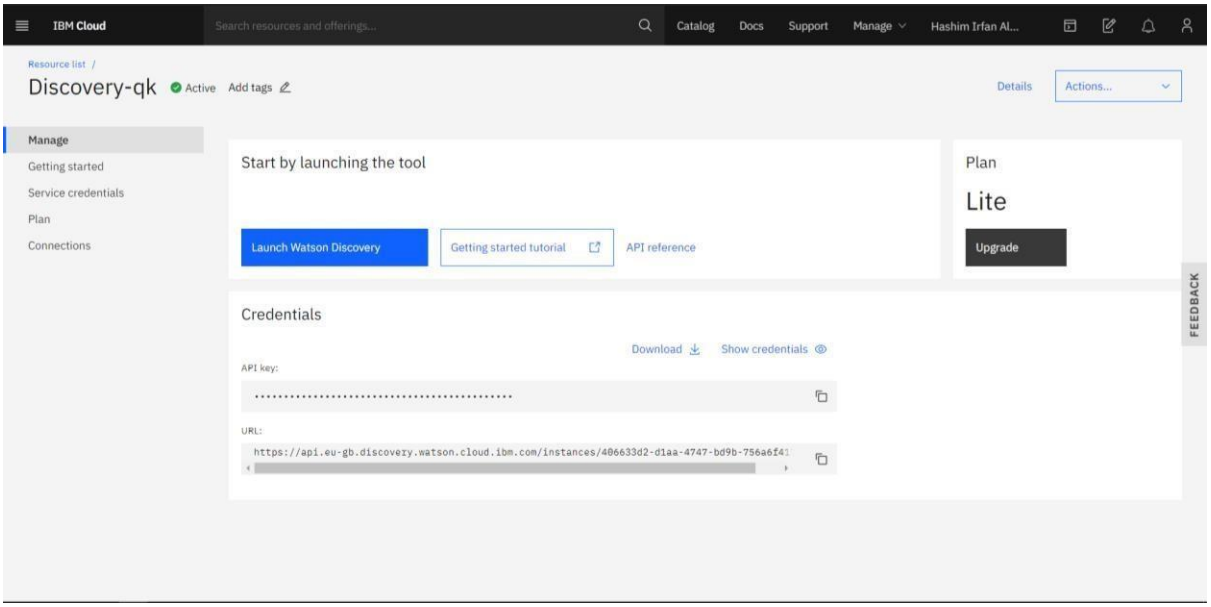
All of your existing Resource list will be shown here, click on Services to unveil the list of services you have.



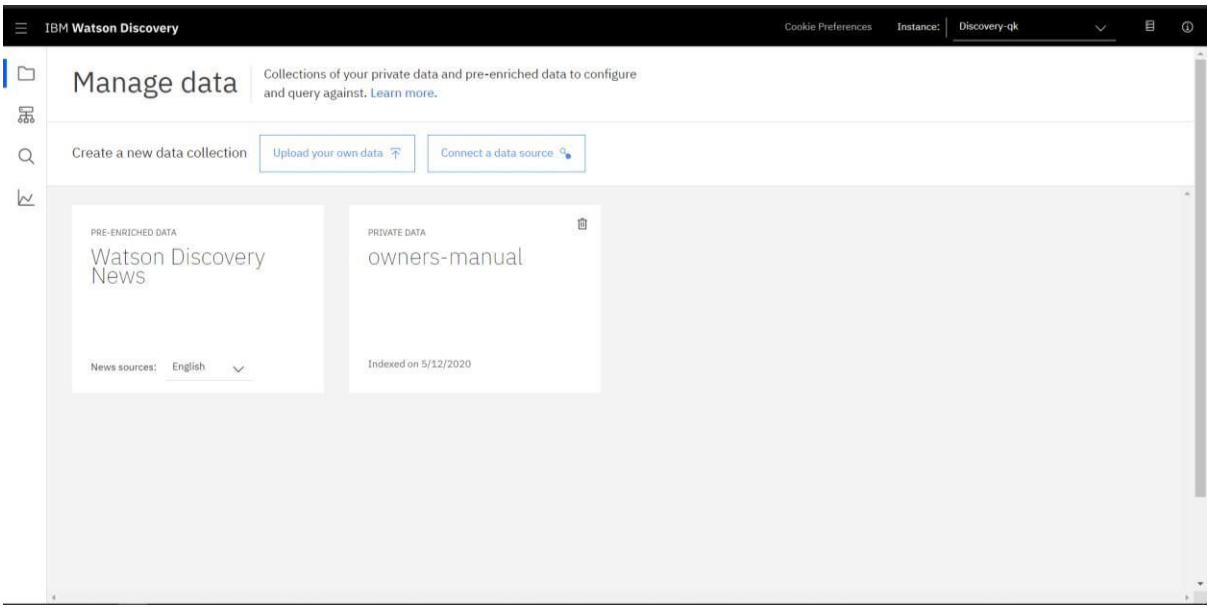
Here we can find that the status of Watson Discovery and Watson Assistant as Active which means we have configured the services correctly.

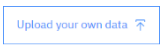
2. Configure Watson Discovery

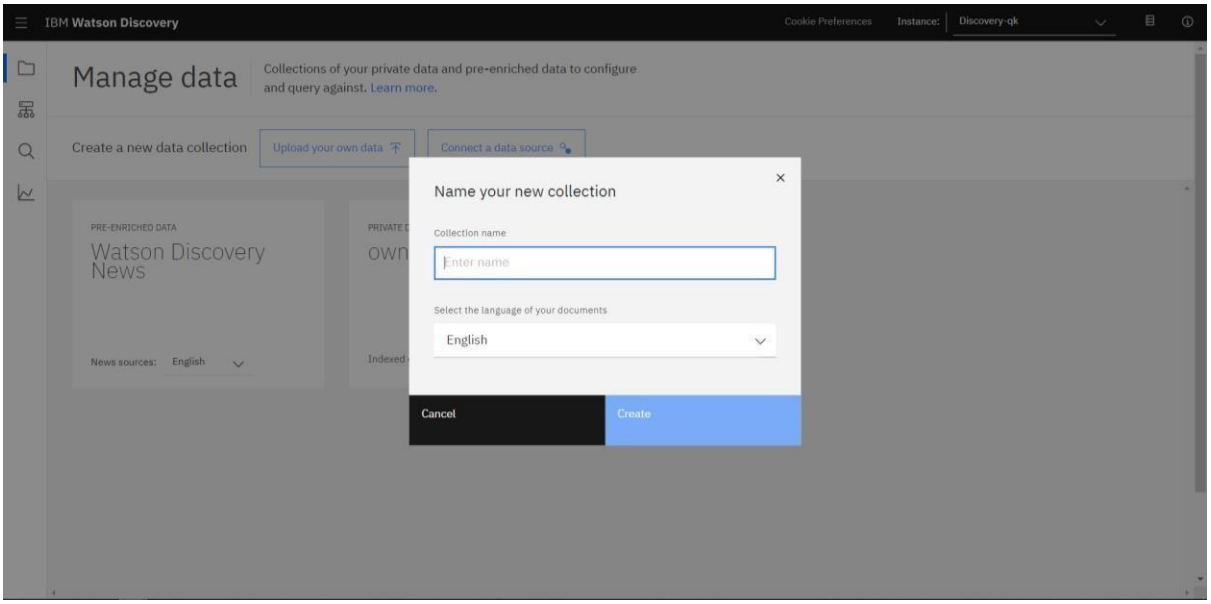
From the resource list screen, click Discovery-qk to open Watson Discovery service.



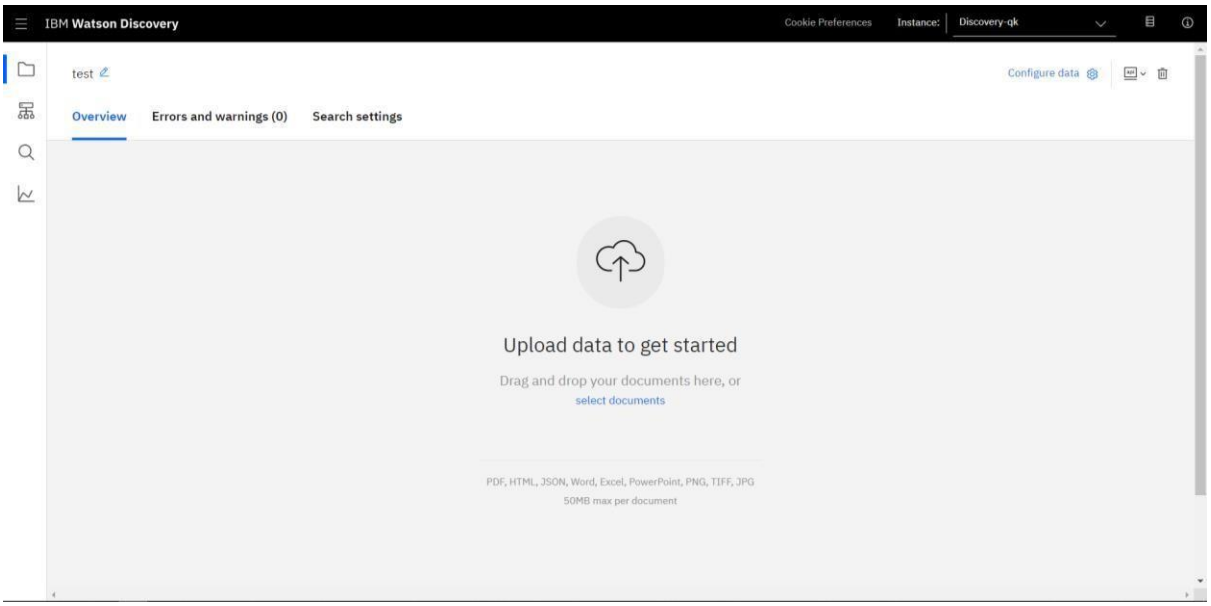
Click on Launch Watson Discovery to launch Watson Discovery Service.

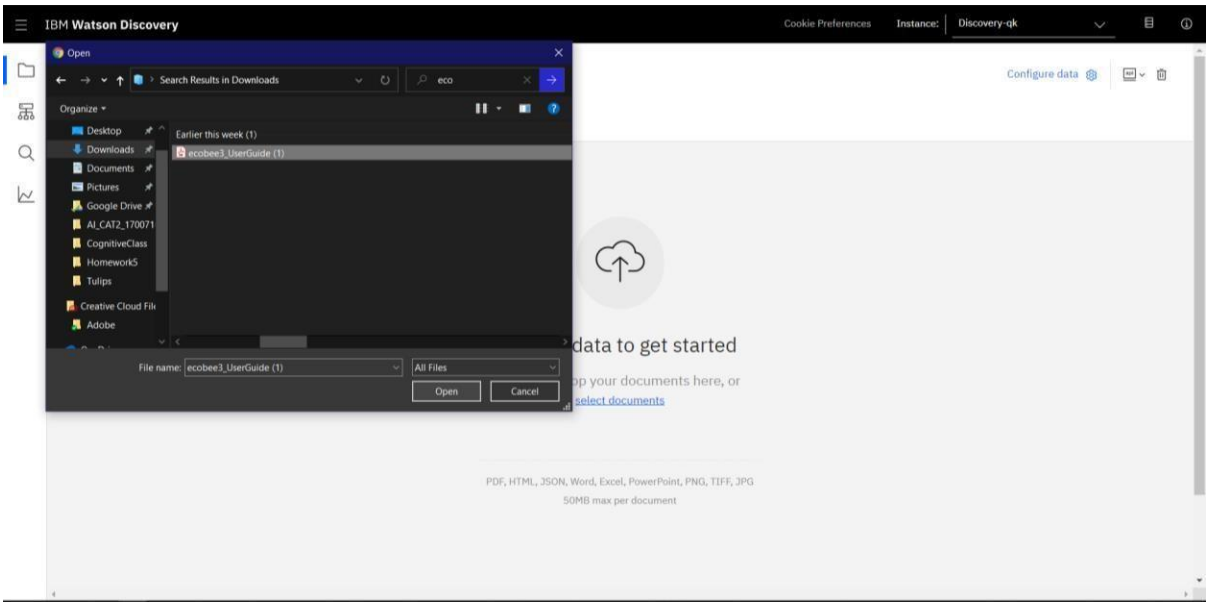


Click on  to create a new data collection.

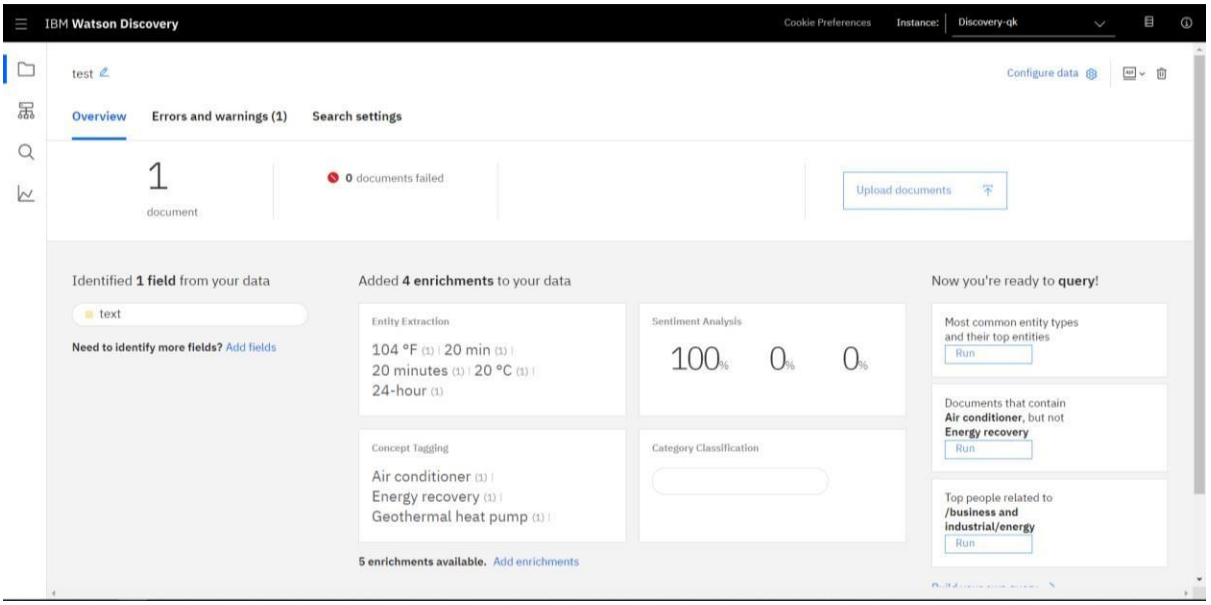


Give the data collection a unique name.





When prompted, select and upload the ecobee3_UserGuide.pdf file located in the data directory of your local repo.



Before proceeding further, let’s learn about Smart Document Understanding(SDU)

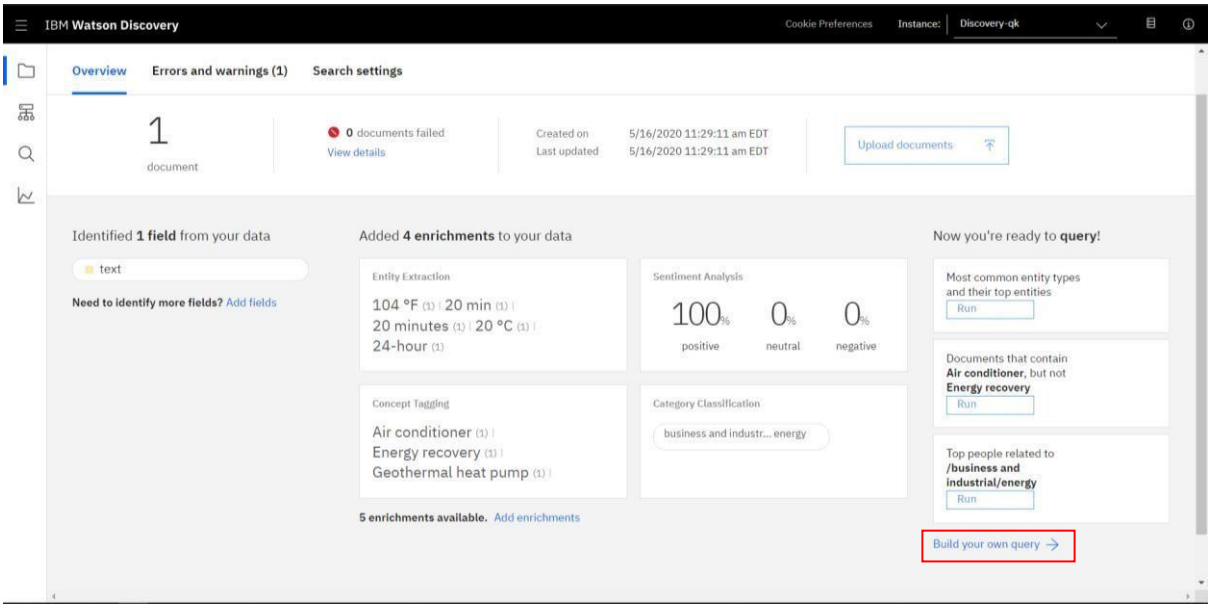
SDU trains Watson Discovery to extract custom fields in your documents. Customizing how your documents are indexed into Discovery will improve the answers returned from queries.With SDU, you annotate fields within your documents to train custom conversion models. As you annotate, Watson is learning and will start predicting annotations. SDU models can also be exported and used on other collections.

Current document type support for SDU is based on your plan:

- Lite plans: PDF, Word, PowerPoint, Excel, JSON, HTML

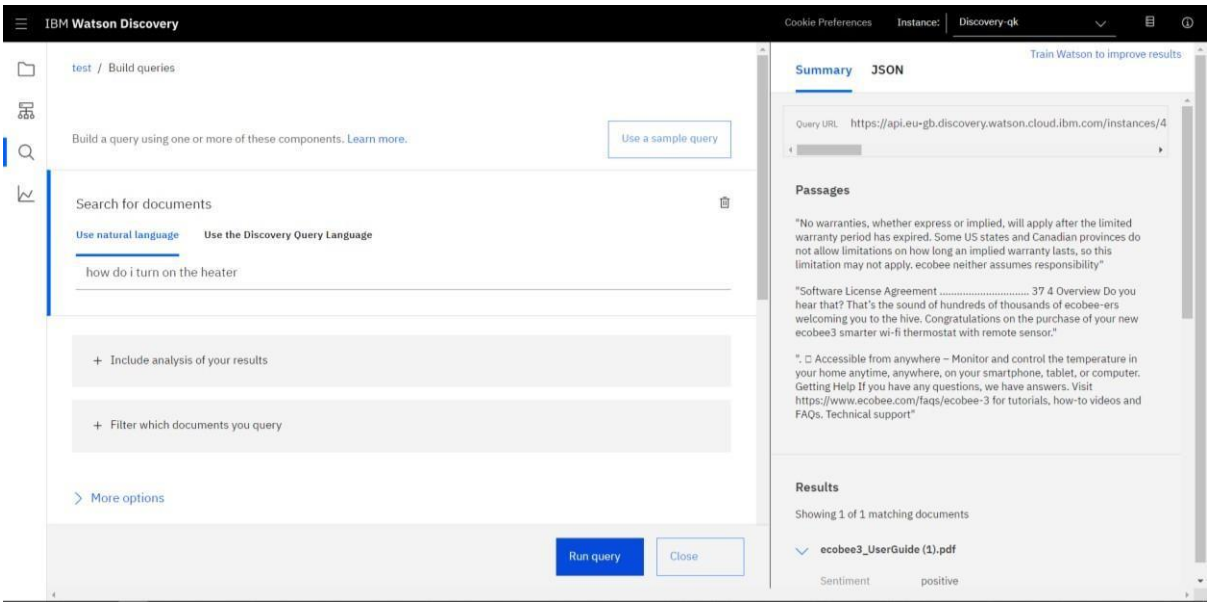
- Advanced plans: PDF, Word, PowerPoint, Excel, PNG, TIFF, JPG,JSON, HTML

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.



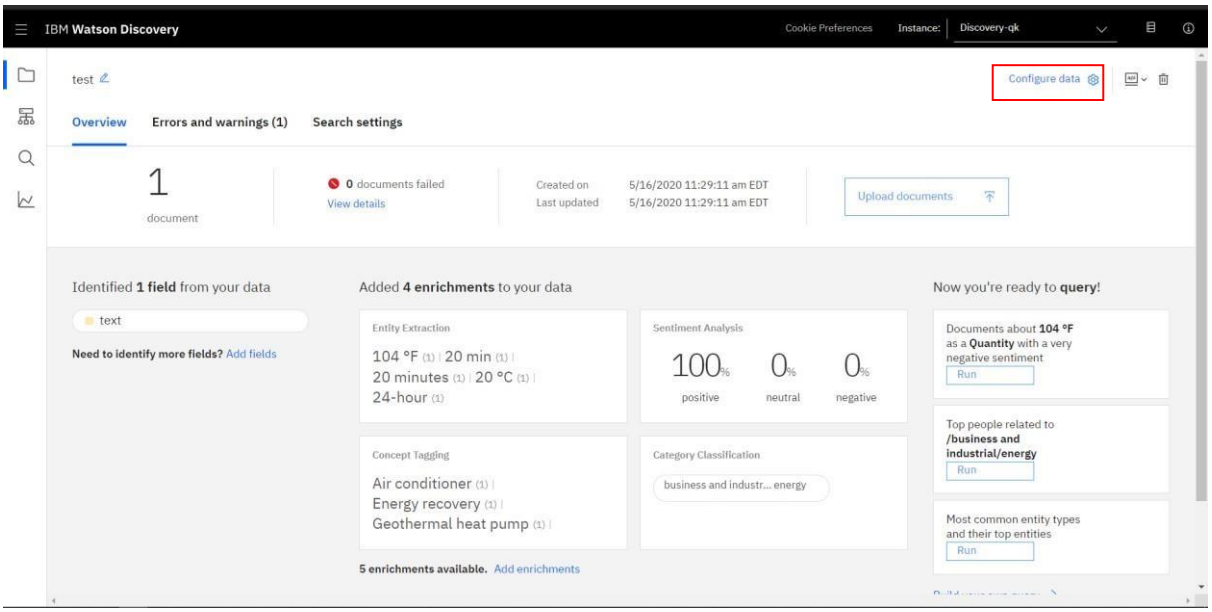
Click on Build your own query.

Now, 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.



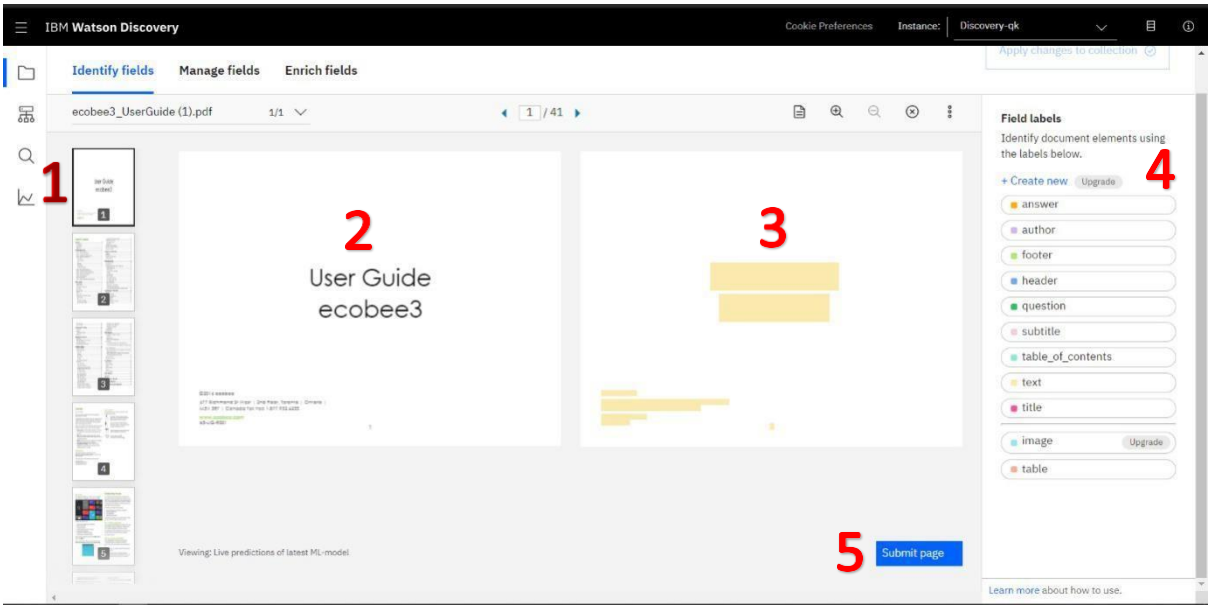
Now let's apply SDU to our document to see if we can generate some better query responses.

Go back to the Discovery collection panel (previous screen)



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 markwill 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.

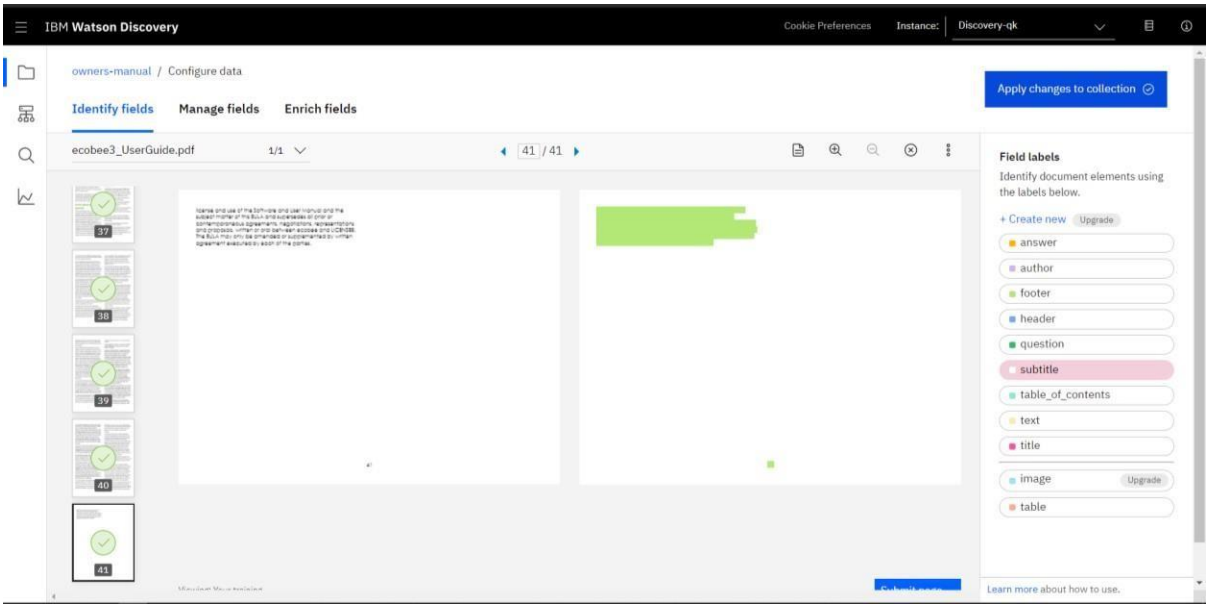
As you go through the annotations one page at a time, Discovery is learning and should start automatically updating the upcoming pages. Once you get to a page that is already correctly annotated, you can stop, or simply click Submit [5] to acknowledge it is correct. The more pages you annotate, the better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

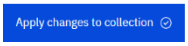
For this specific owner's manual, at a minimum, it is suggested to mark the following:

- 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. After completing the process for all

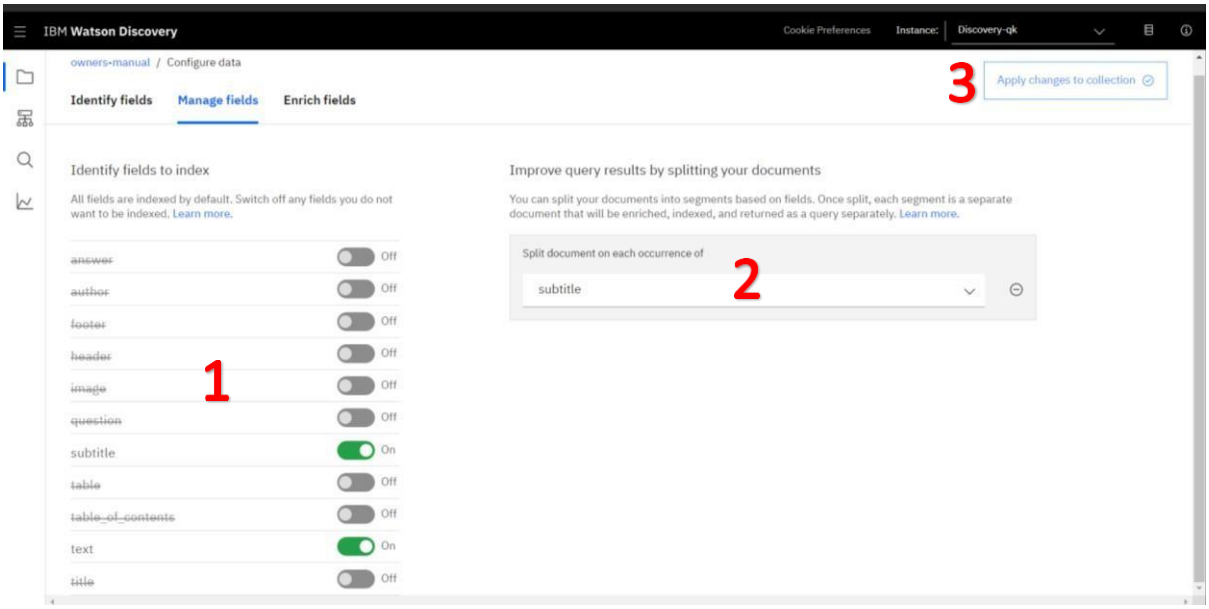
pages,



Click the 

You will be asked to reload the document. Choose the same ecobee3_UserGuide.pdf document as before.

Now, click on Manage fields tab on the Configure data panel



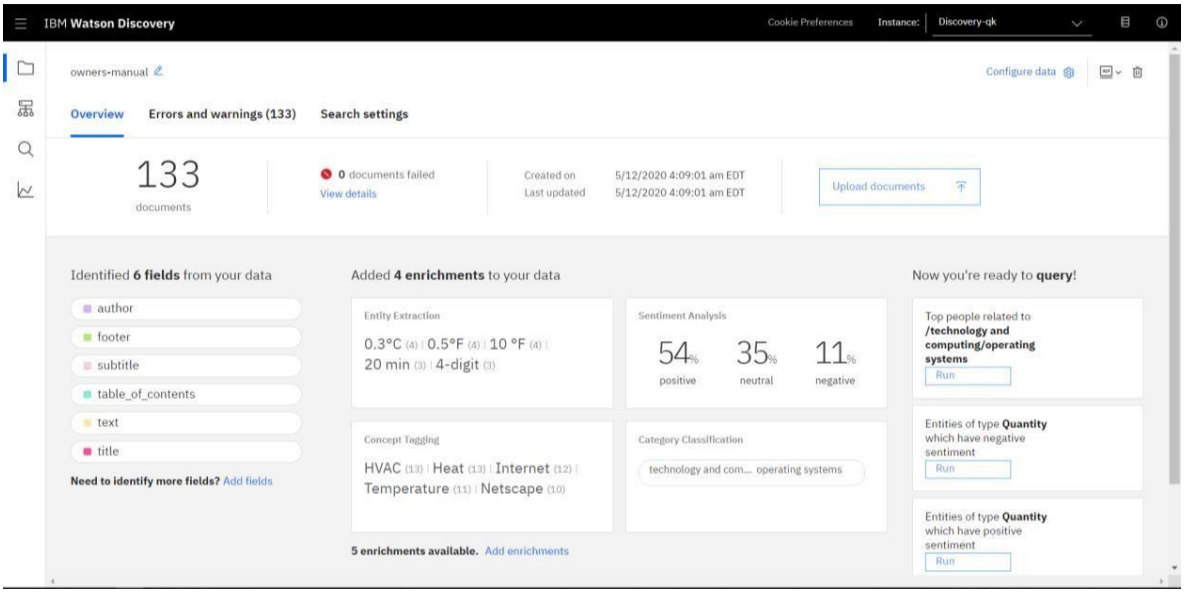
[1] Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except subtitles and text.

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

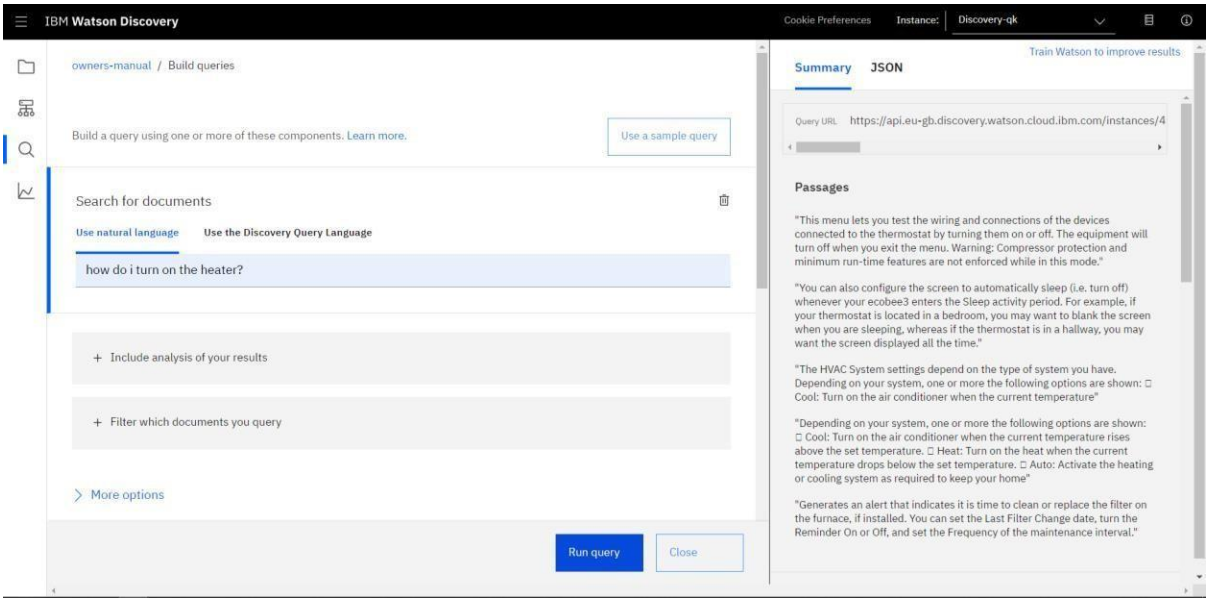
Click [3] to submit your changes.

Once again, you will be asked to reload the document. Choose the same ecobee3_UserGuide .pdf document as before.

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

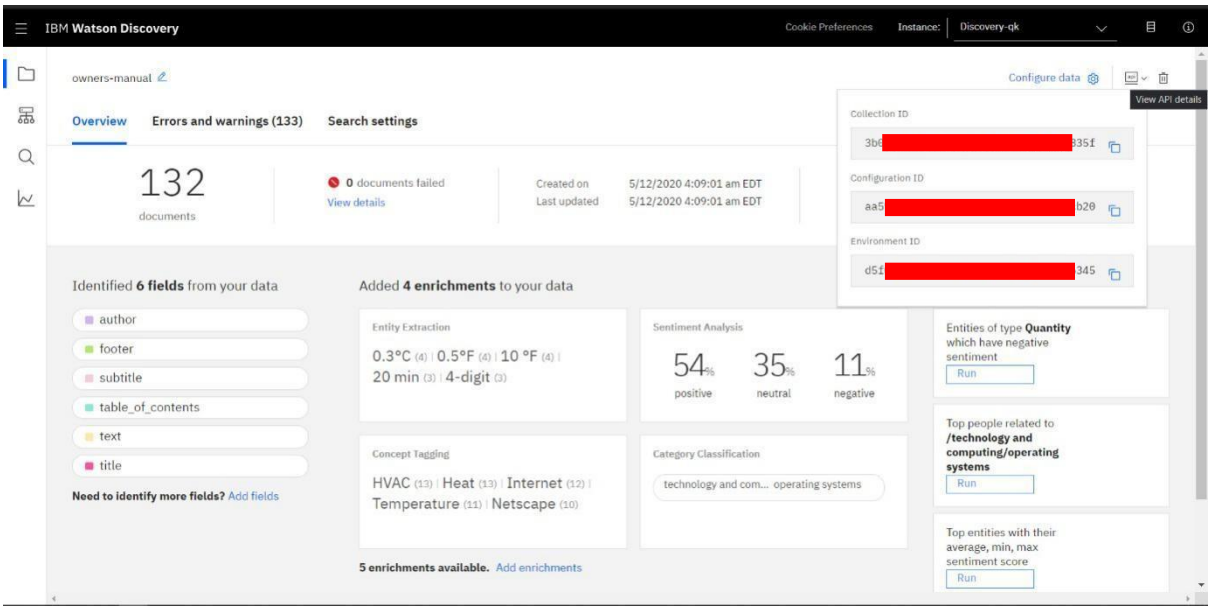


Now click the Build your own query and see how much better the results are.



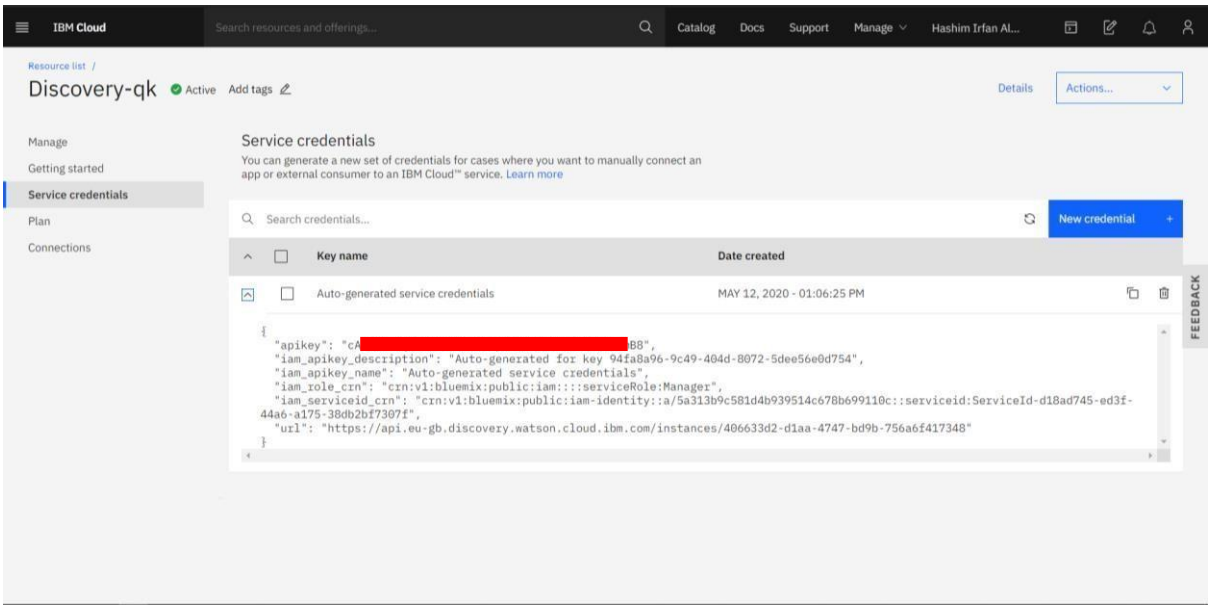
In upcoming steps, you will need to provide some credentials to access your Discovery collection so to Store credentials for future use follow the below steps.

The Collection ID and Environment ID values can be found by clicking the located at the top right side of your collection panel



Now go to the Watson Discovery Resource List Screen. Here, select service credentials.

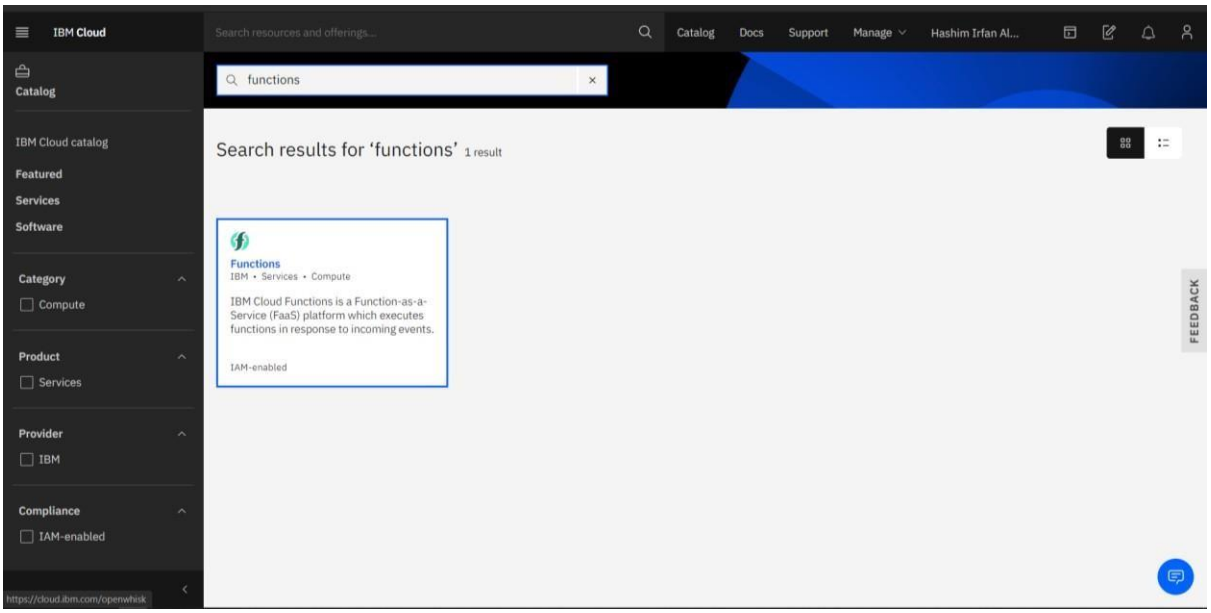
The apikey and URL endpoint for your service can be found here.



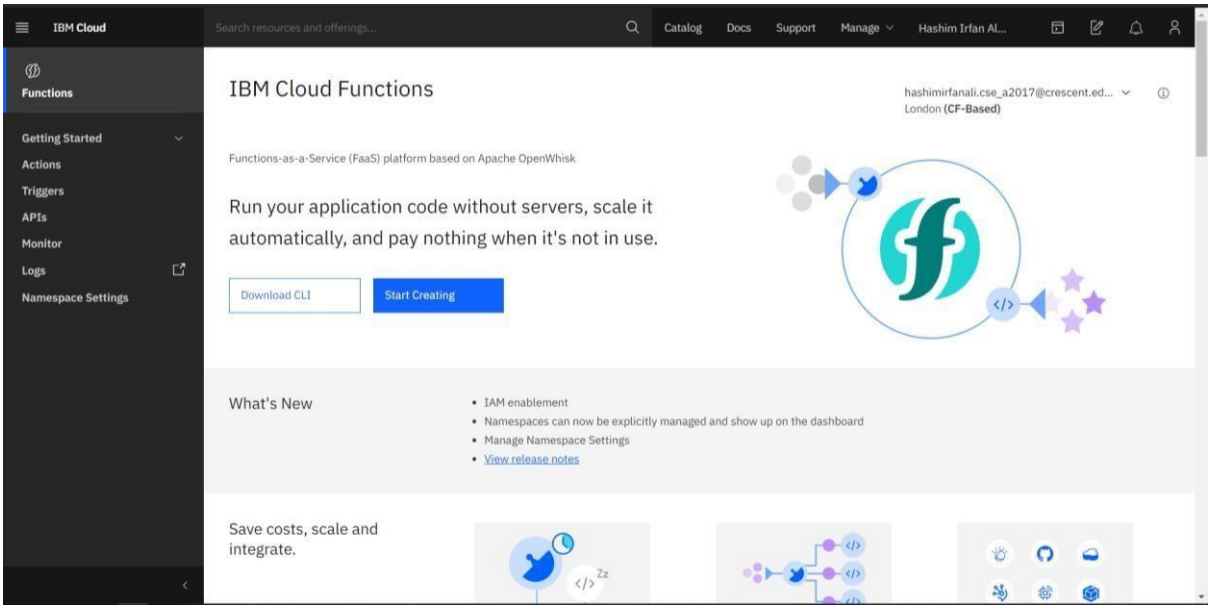
3. Create IBM Cloud Functions action

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

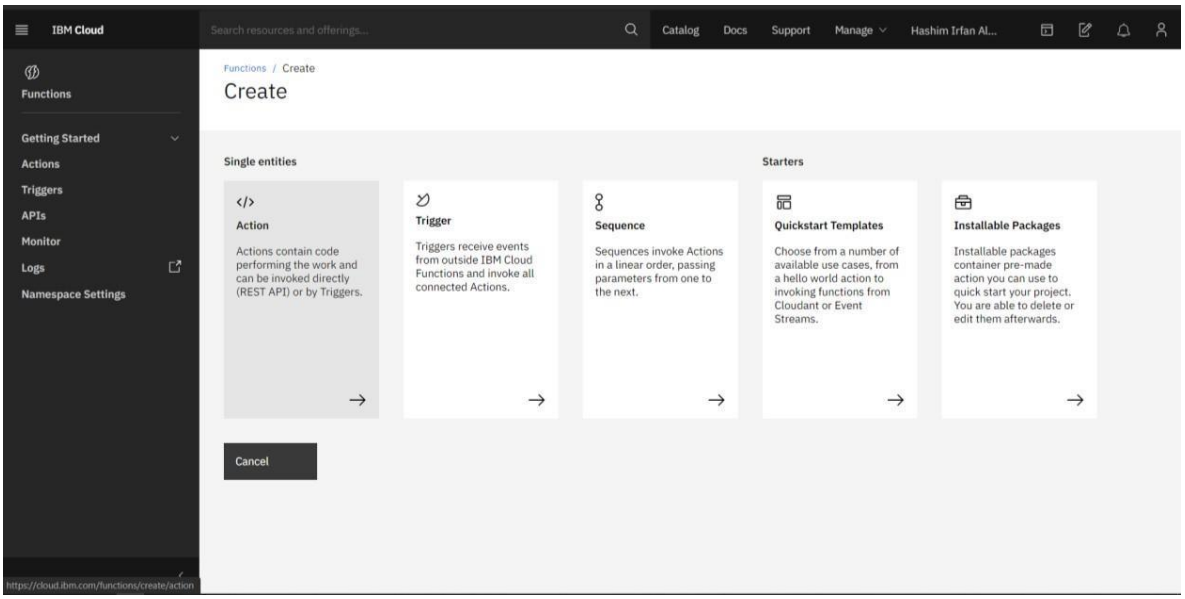
Go to IBM Cloud Dashboard, click on Create Resource and search for Functions



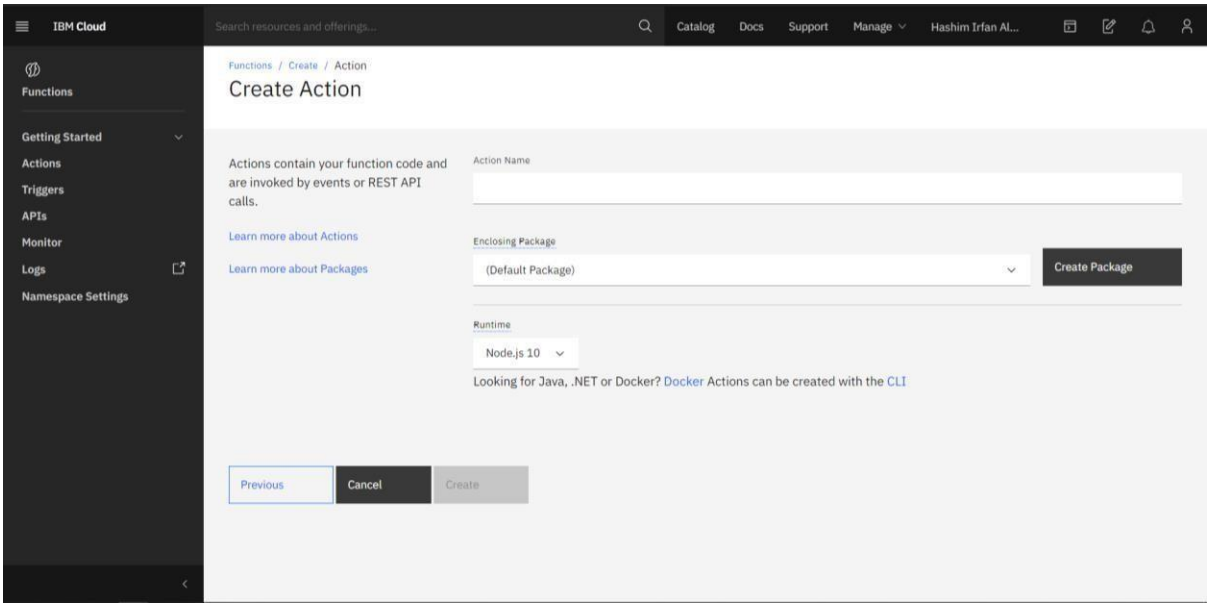
Select the Functions Card.



From the Functions main panel, click on Start Creating.

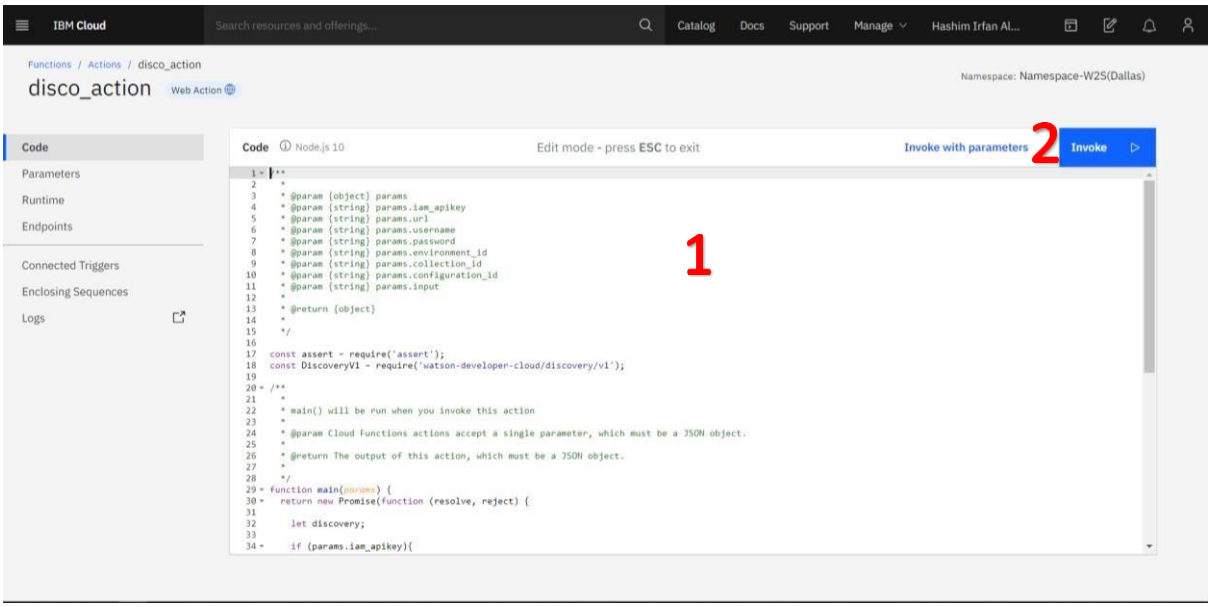


Here, select Actions.



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

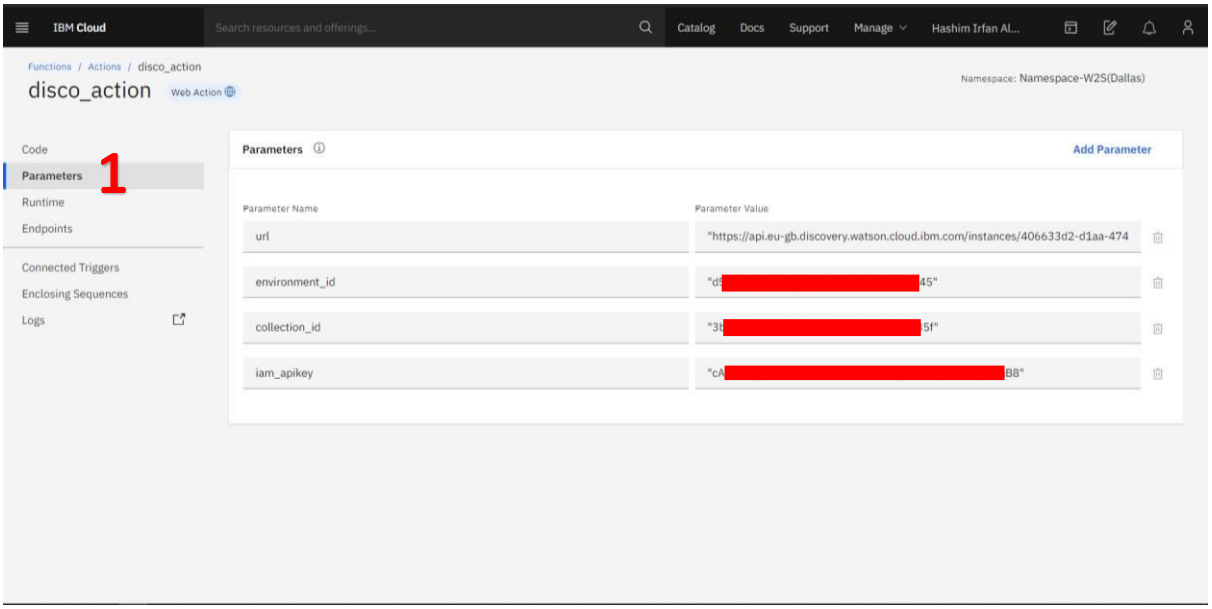
Once function is created, click on code tab.



In the code editor window [1], 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.

Now if click the invoke [2] button, it will fail due to credentials not being defined yet.

To solve this, select parameter[1] tab

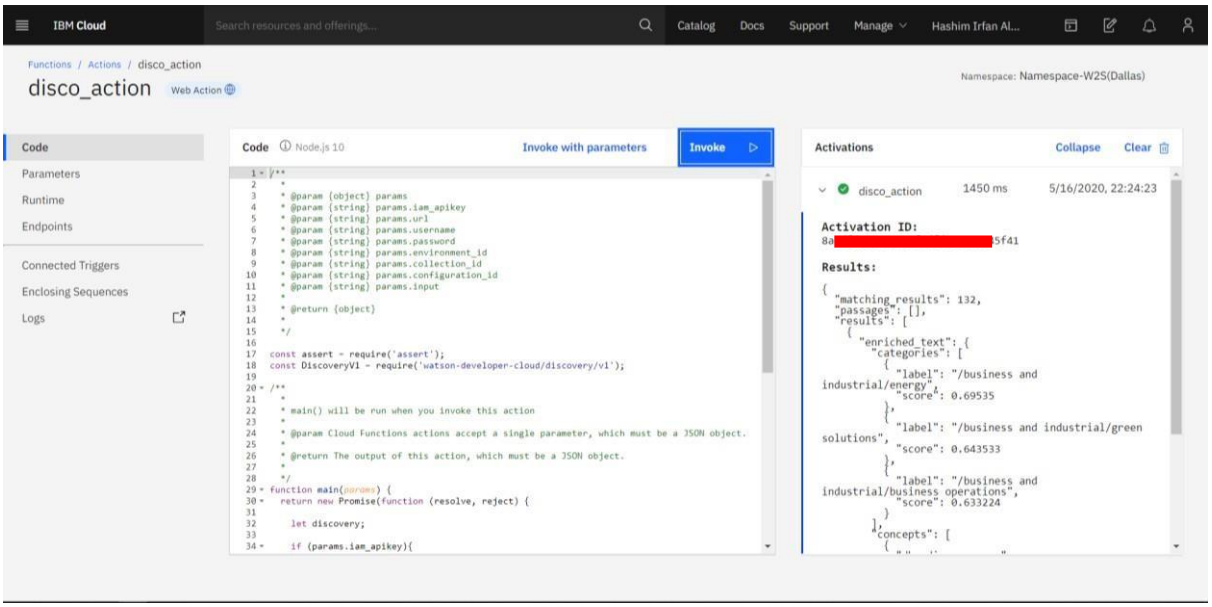


Add the following keys:

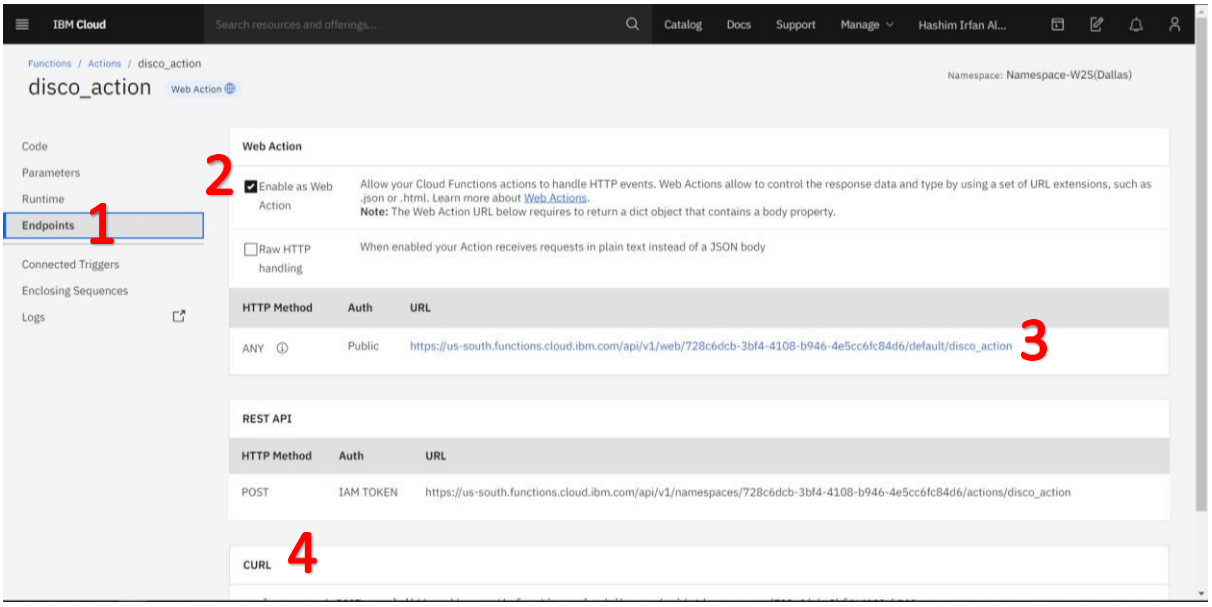
- url
- environment_id
- collection_id
- iam_apikey

For the above values, please use the values associated with the Discovery service you created in the previous step.Enclose your values in double quotes.

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



Now click on endpoints[1] tab



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

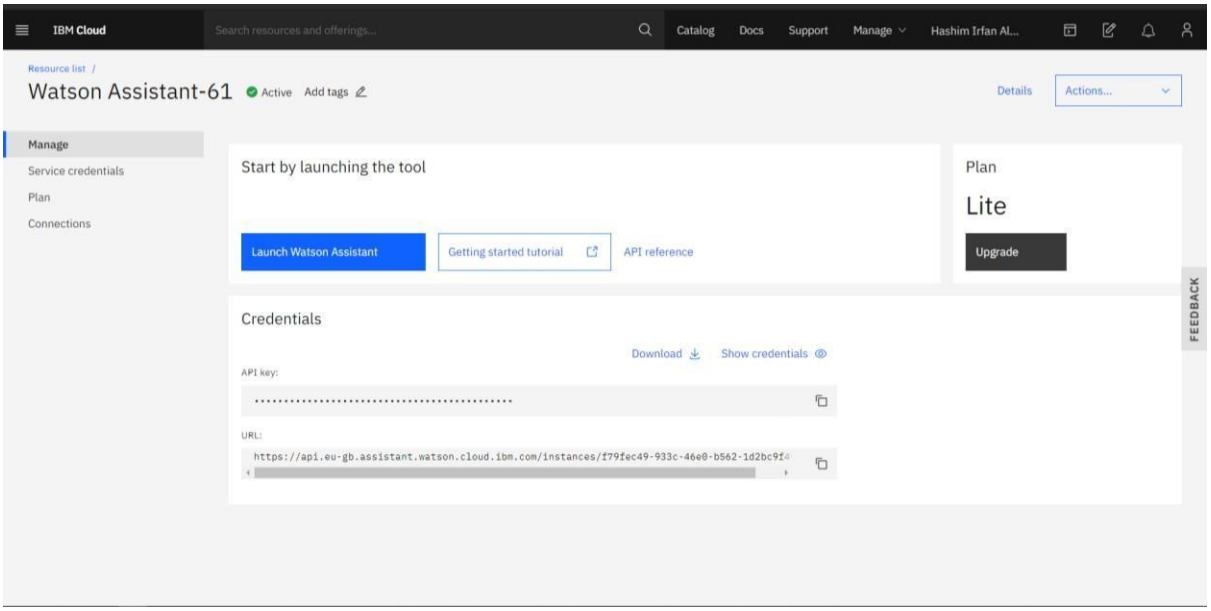
Take note of the URL value [3], as this will be needed by Watson Assistant in a future step.

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

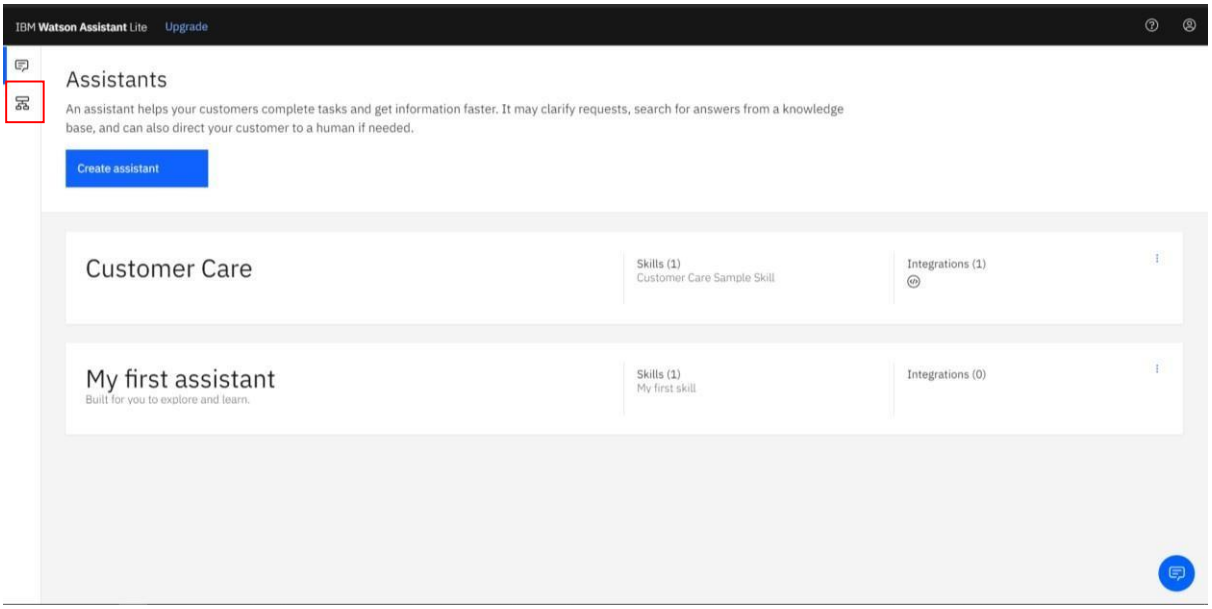
[An IBM Cloud Functions service will not show up in your dashboard resource list. To return to your defined Action, you will need to access Cloud Functions by selecting Create Resource from the main dashboard panel (as shown at the beginning of this step).]

4. Configure Watson Assistant

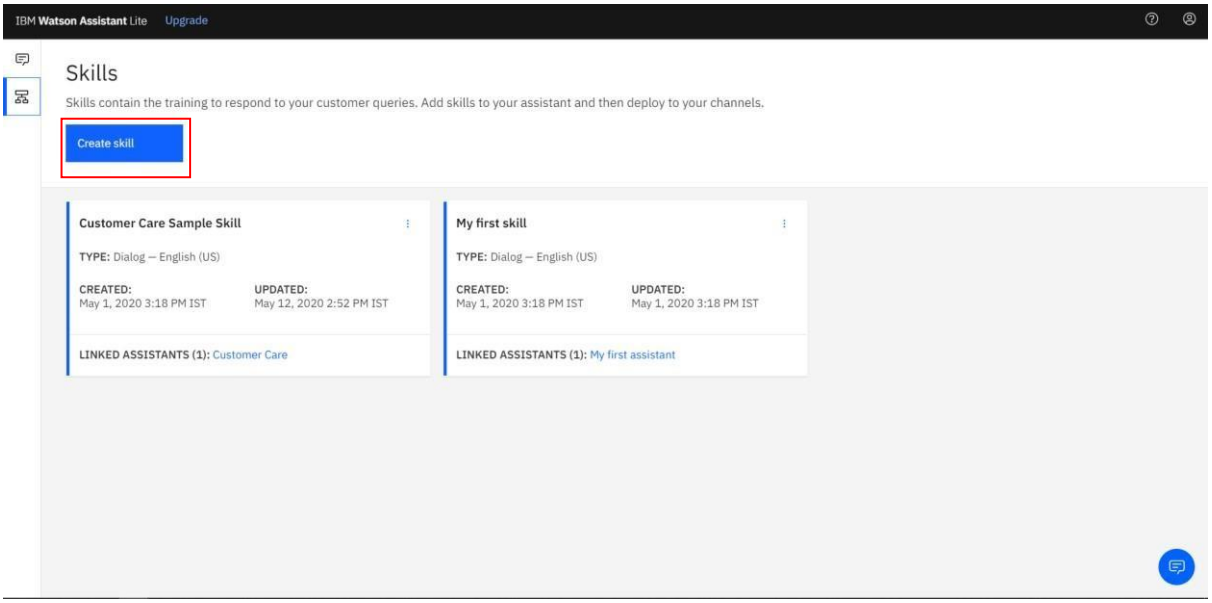
Go back to the IBM Dashboard From the resource list screen, Watson Assistant-61 click to open Watson Assistant service.



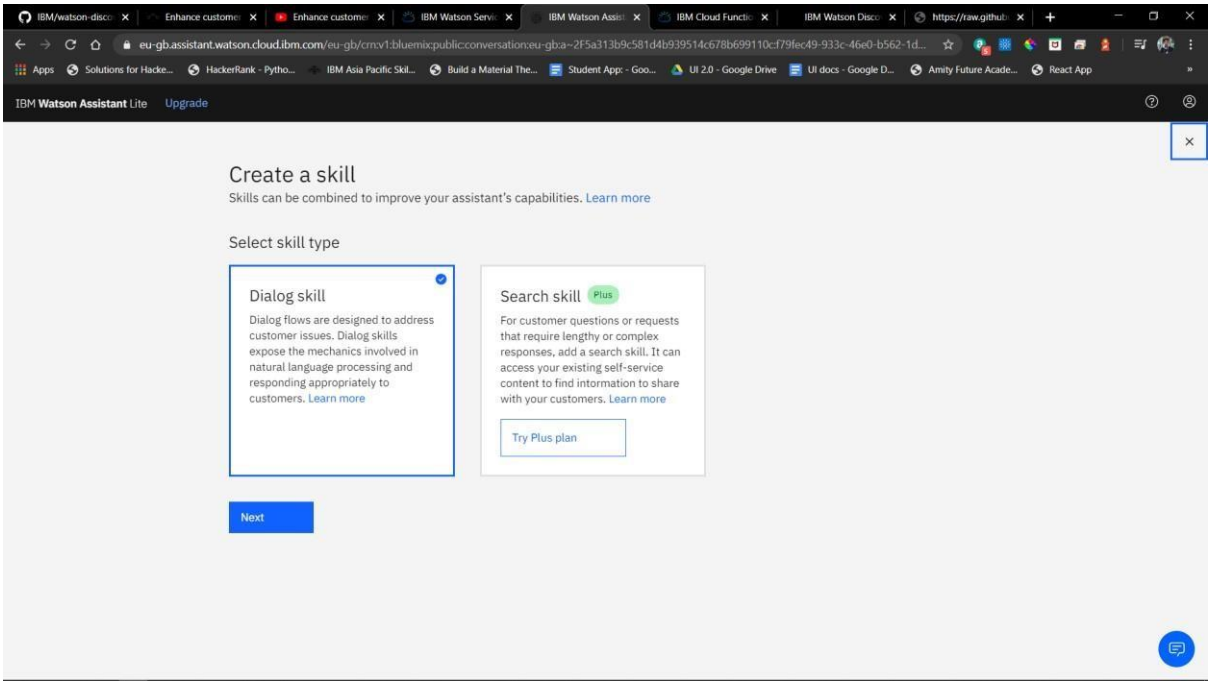
Click on Launch Watson Assistant to launch Watson Assistant.



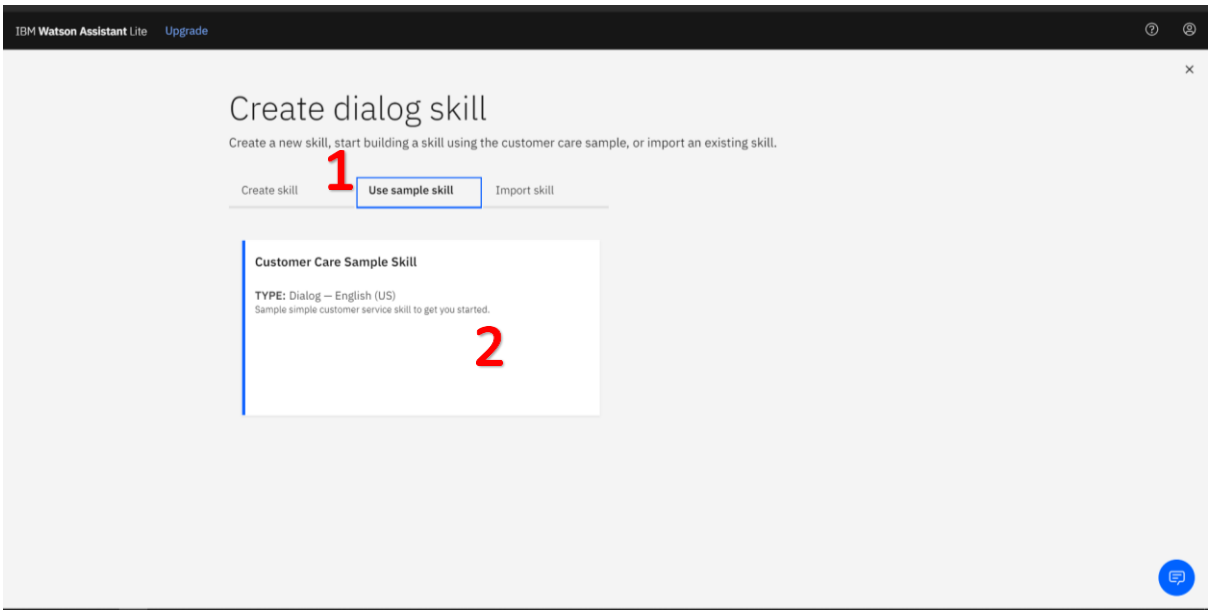
Click on the skills tab



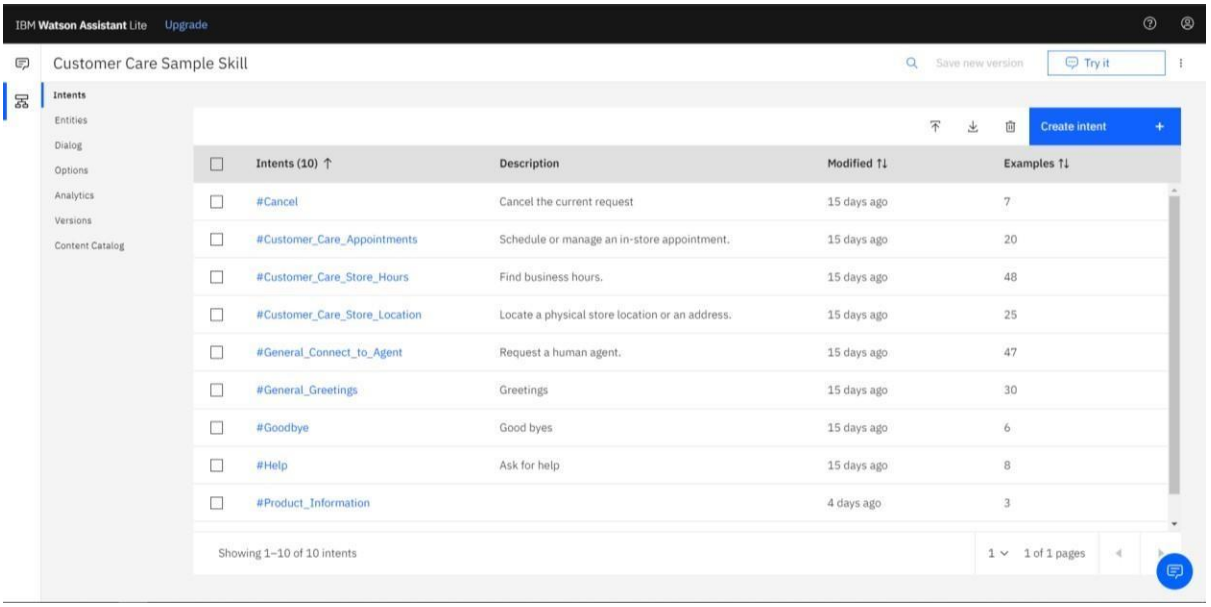
Click Create Skill



Select Dialog Skill Card and Click next



Select Use Sample Skill [1] and Select Customer Care Sample Skill [2]. This dialog skill contains all of the nodes needed to have a typical call centre conversation with a user.



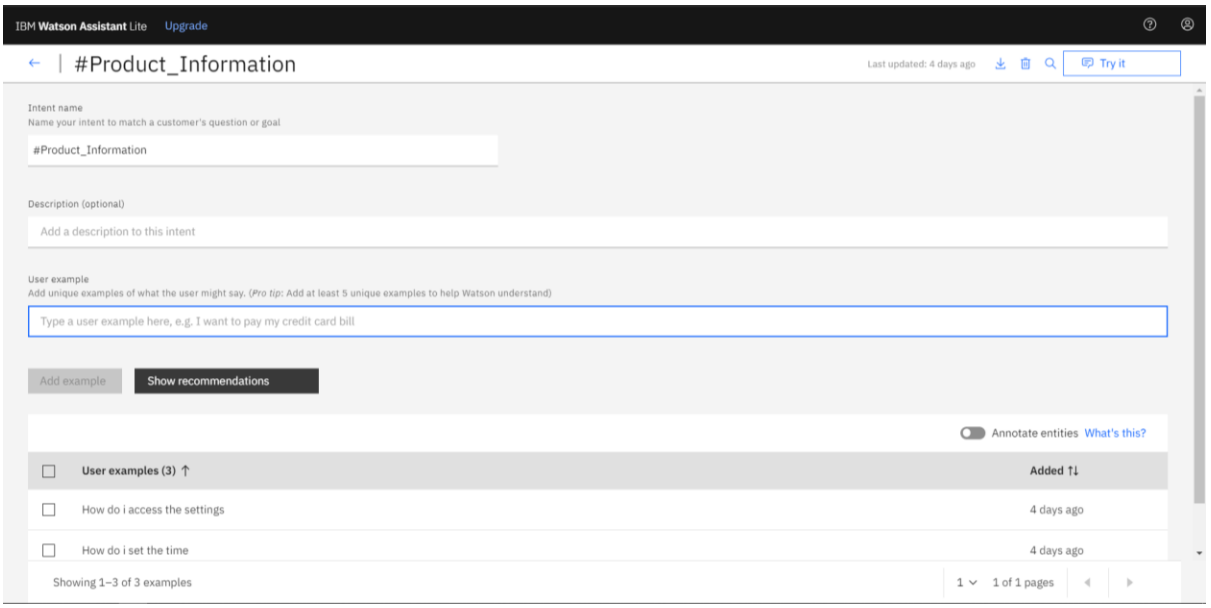
As the default customer care dialog does not have a way to deal with any questions involving outside resources, so we will need to add new intent.

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

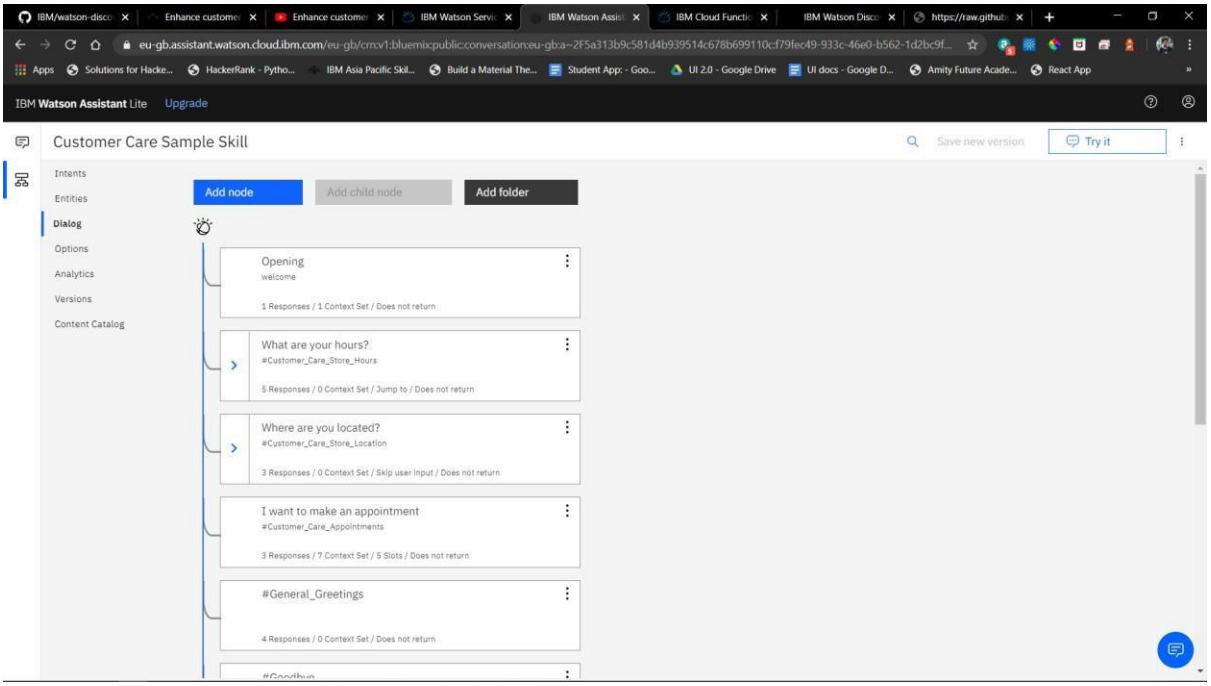
From the Customer Care Sample Skill panel, select the Intents tab.

Click the Create intent button.

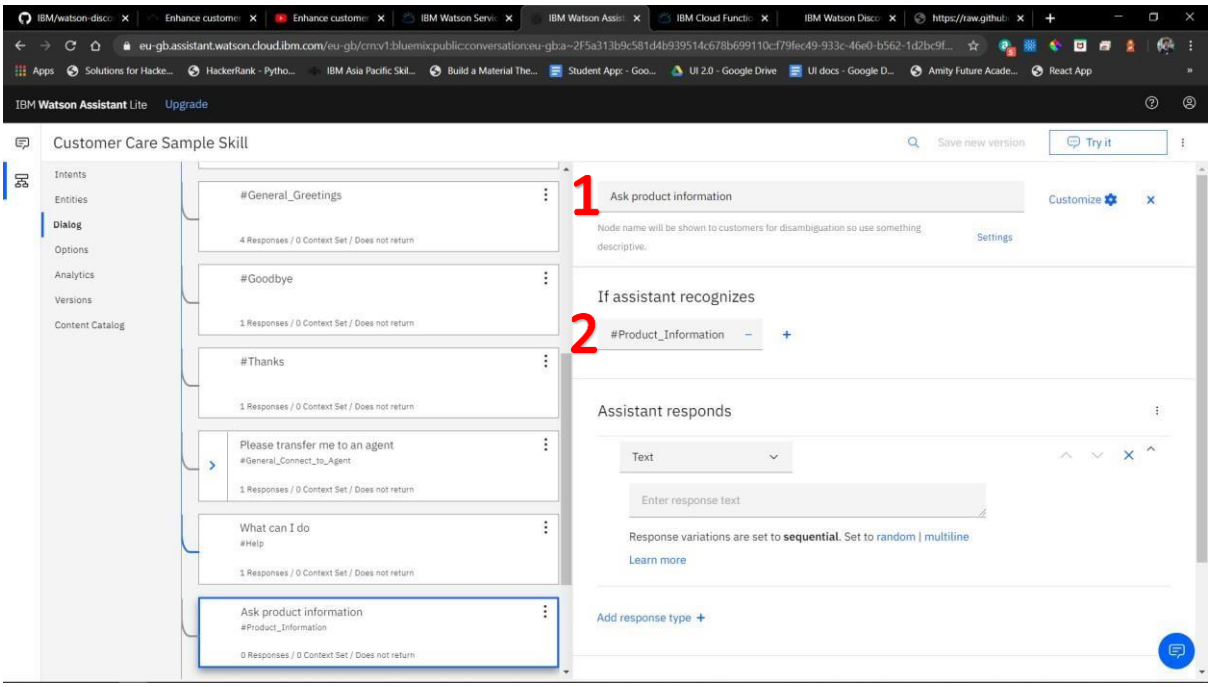
Name the intent #Product_Information, and at a minimum, enter the following example questions to be associated with it.



Go back to the previous page after doing this, then click on Dialog Tab and add a node below What can I do node.



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

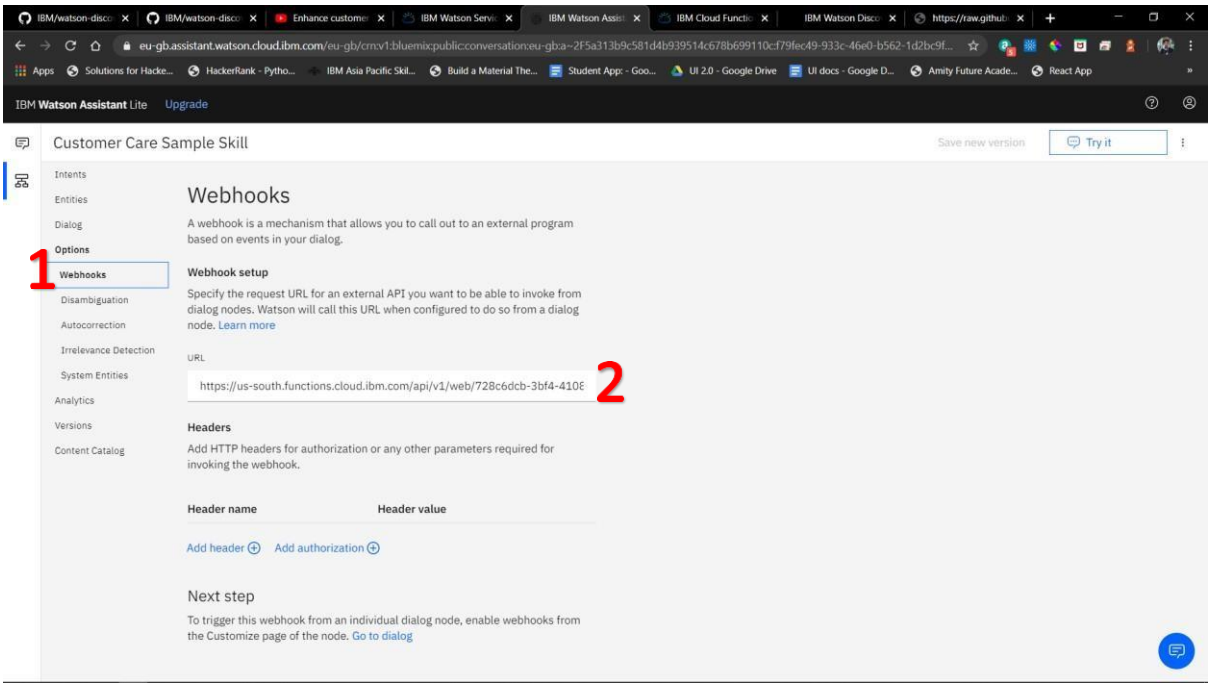


This means that if Watson Assistant recognizes a user input such as "how do I set the time?", it will direct the conversation to this node.

Before proceeding further, let’s learn about webhook.

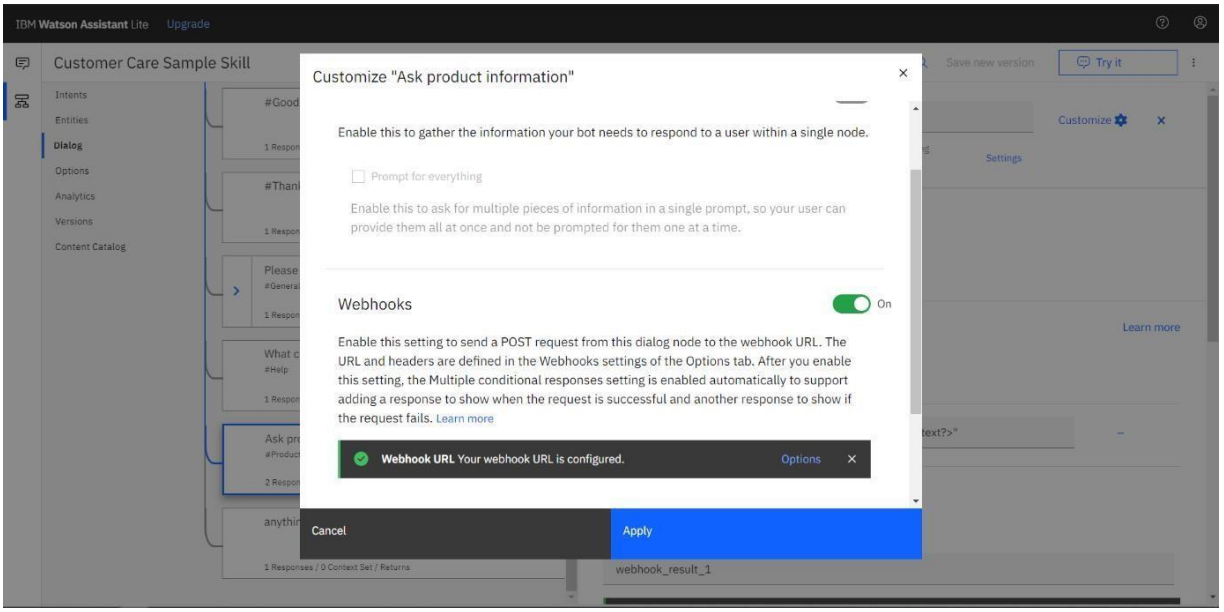
A webhook is a mechanism that allows you to call out to an external program based on something happening in your program. When used in a Watson Assistant dialog skill, a webhook is triggered when the Assistant processes a node that has a webhook enabled. The webhook collects data that you specify or that you collect from the user during the conversation and save in context variables, and sends the data to the Webhook request URL as an HTTP POST request. The URL that receives the webhook is the listener. It performs a predefined action using the information that is provided by the webhook as specified in the webhook definition, and can optionally return a response.

In our example, the webhook will communicate with an IBM Cloud Functions web action, which is connected to the Watson Discovery service.



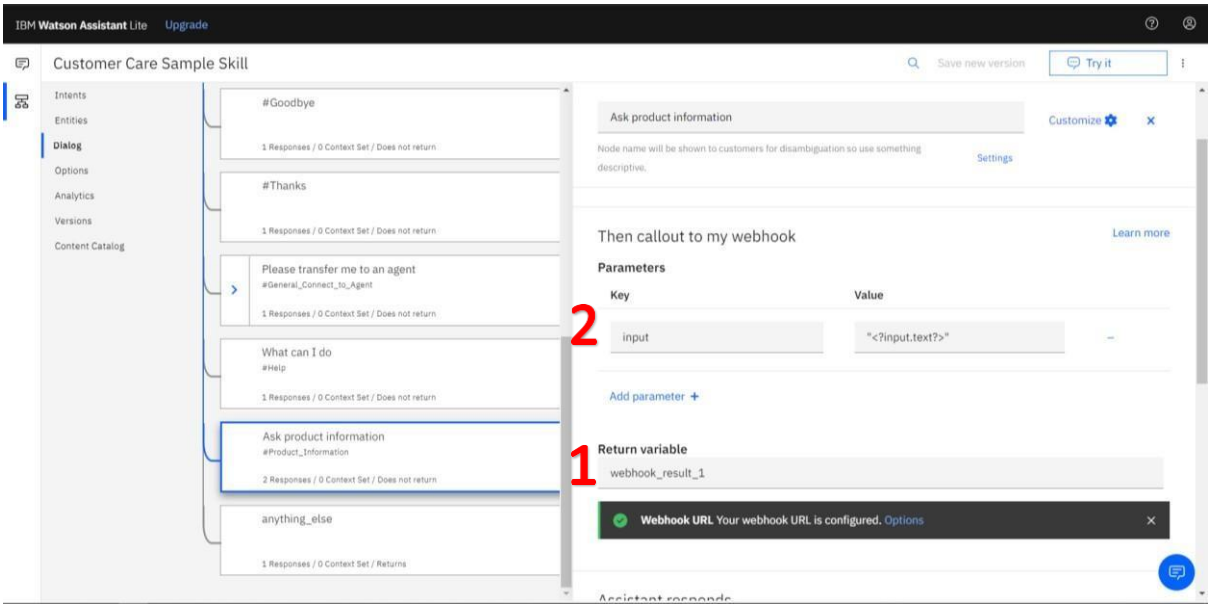
Click Webhooks[1] tab and enter the URL[2] to enable web hook for the IBM Cloud Functions action you created in Step 3.Add .json to the end of the URL to specify the result should be in JSON format.

Return to the Dialog tab, and click on the Ask product information 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.

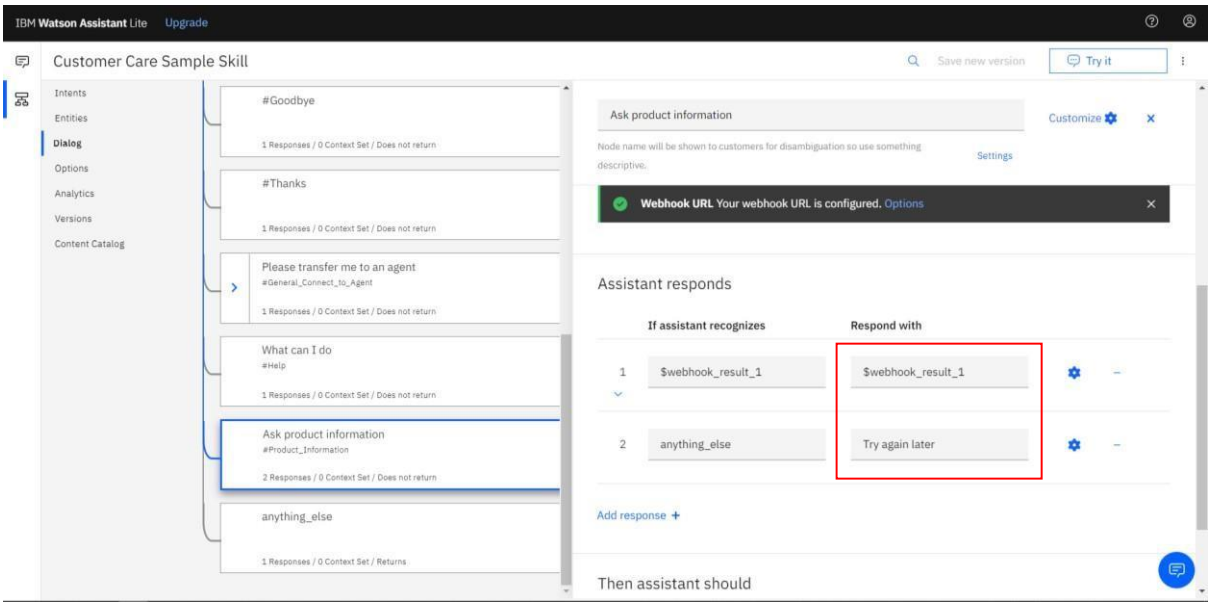


You will also need to pass in the users question via the parameter input [2]. The key needs to be set to the value:

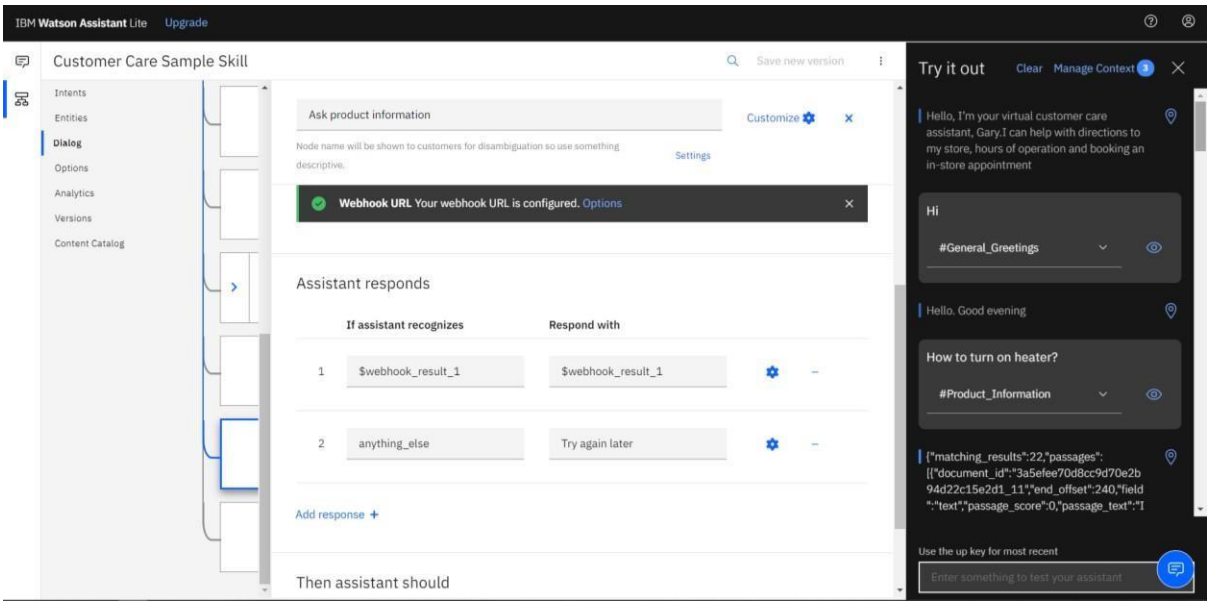
"<?input.text?>"

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:



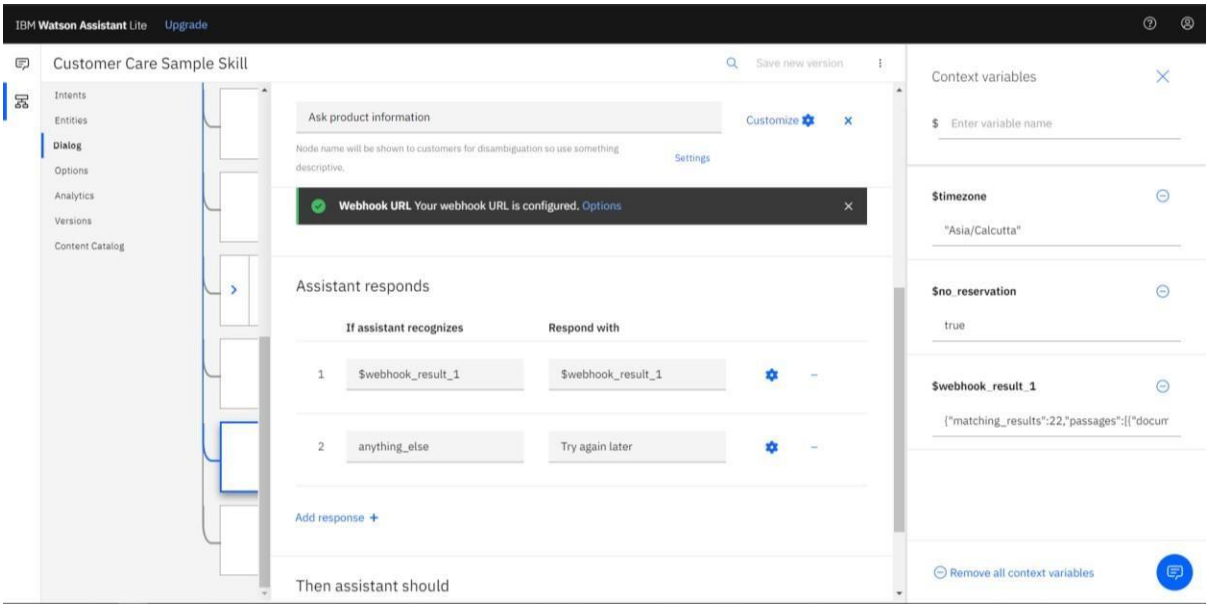
Now we should test the assistant, from click the Try it button located at the top right side of the panel.



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.

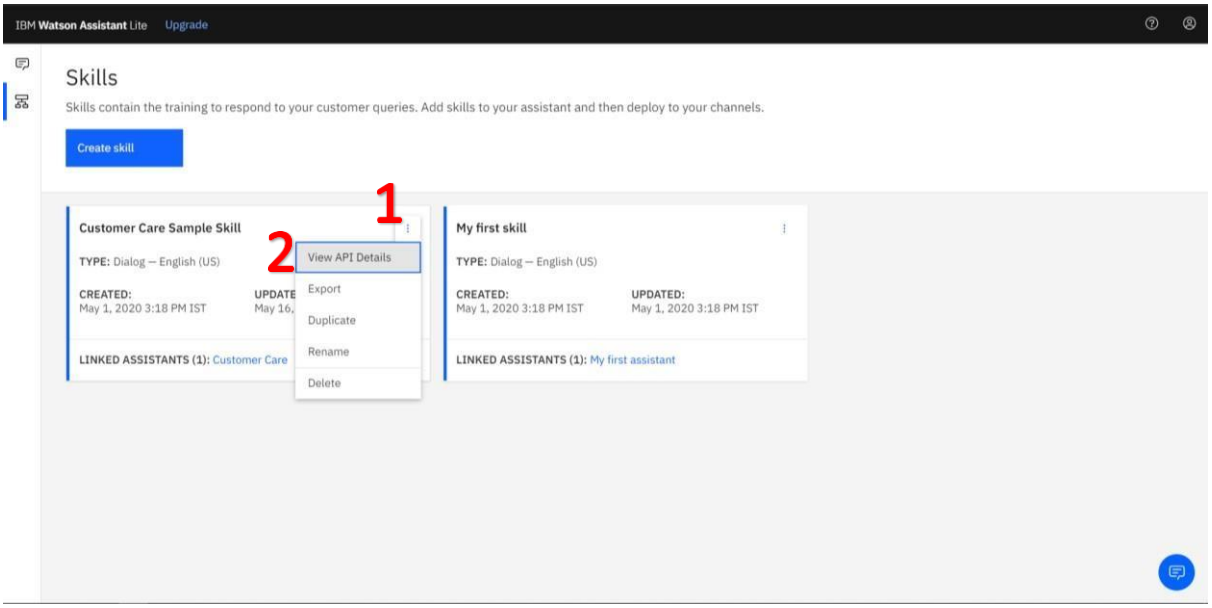
And because we specified that $webhook_result_1.passages be the response, that value is displayed also.

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

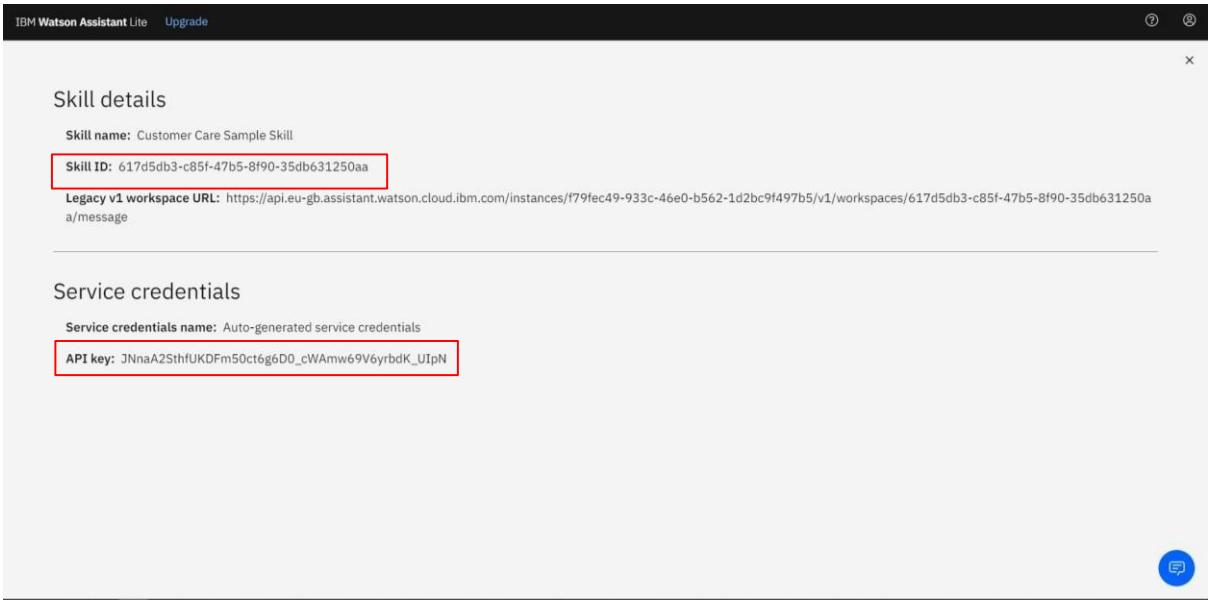


For upcoming steps, you will need to provide some credentials to access your assistant so to store credentials for future use follow these steps below.

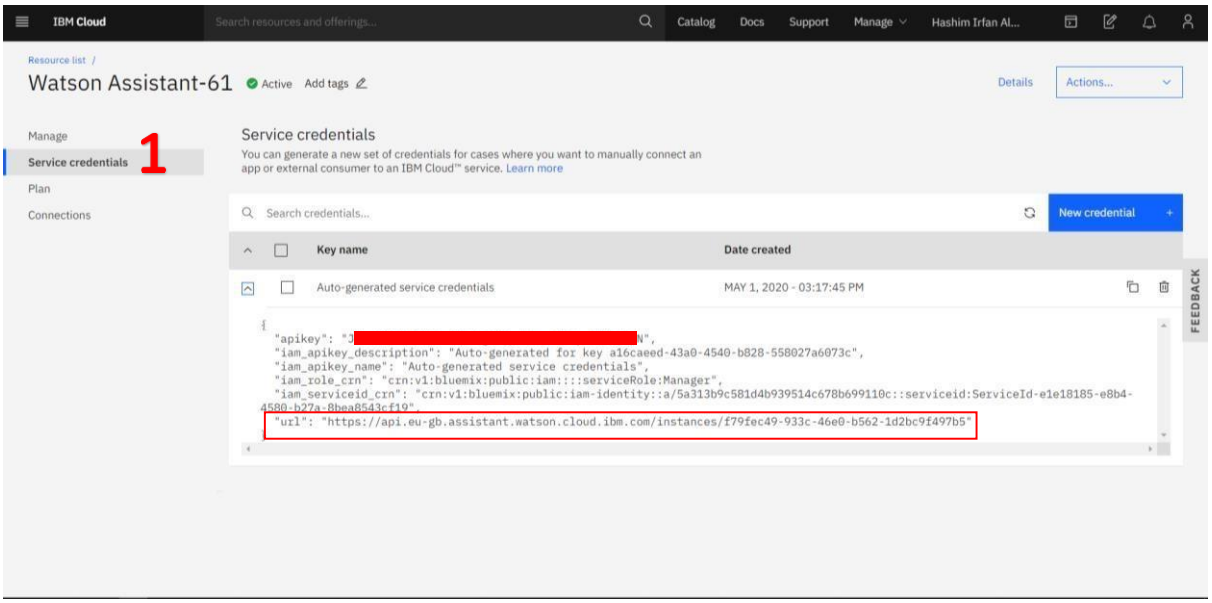
Go back to the skills tab, click [1] and then [2]



The Skill ID and API Key is to be noted.

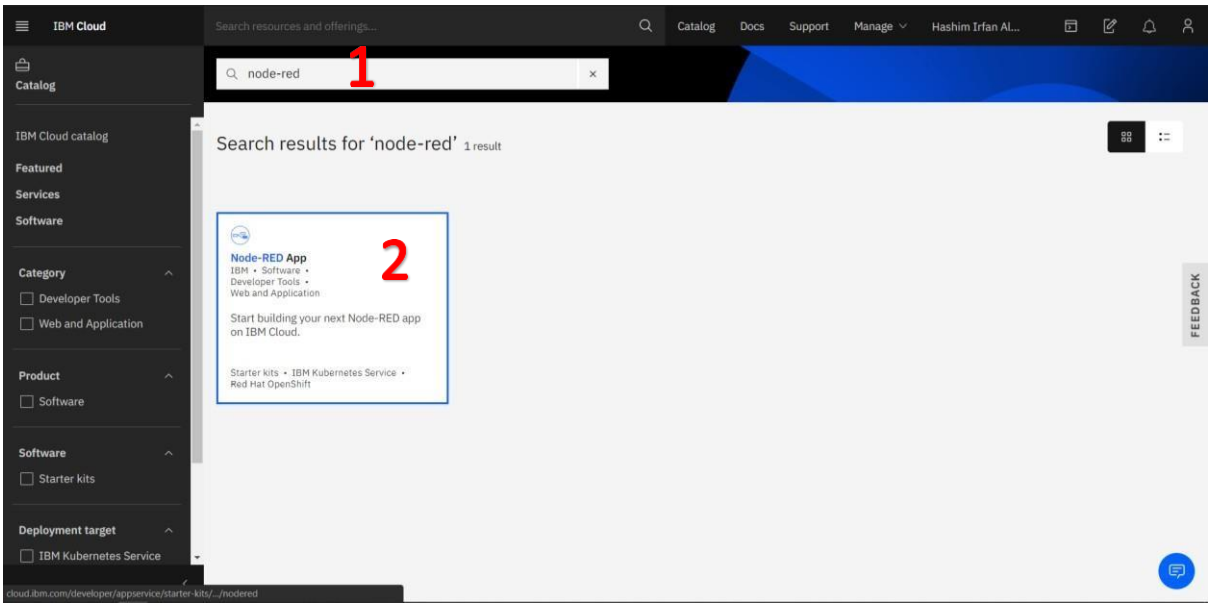


Go Back to the Watson Assistant Resource List, Select Service Credentials [1] and make note of the URL.APIKEY can be found here too.



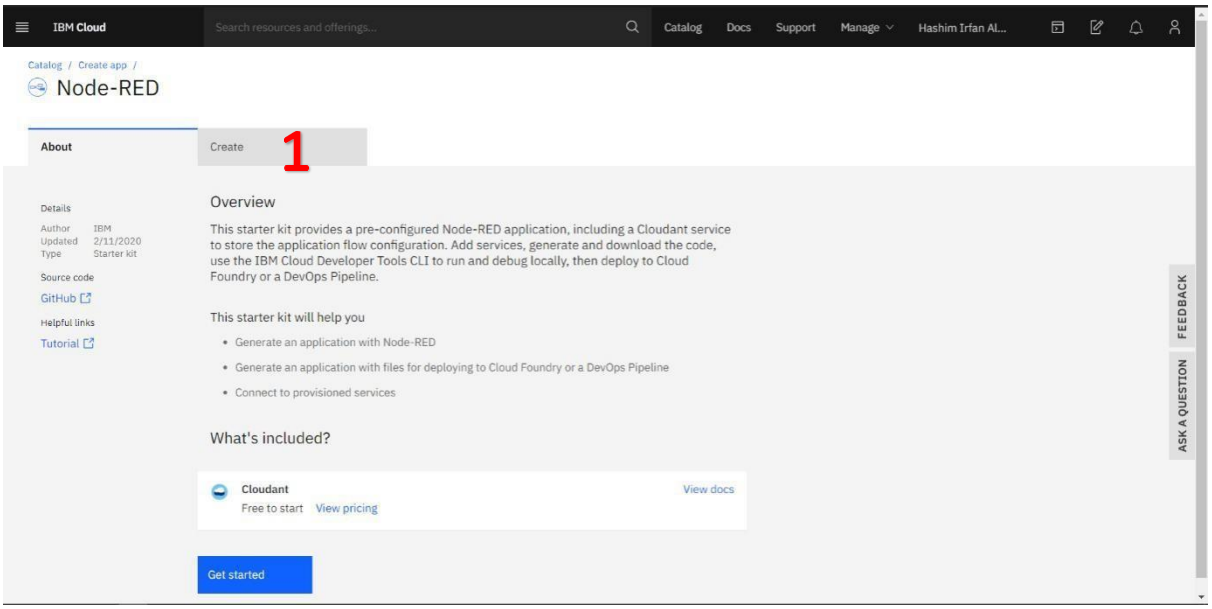
5. Build Node-RED Flow to Integrate All Services

Now it's time to create Node-Red, go to IBM Cloud Dashboard, click on Create Resource and search for node-red[1].

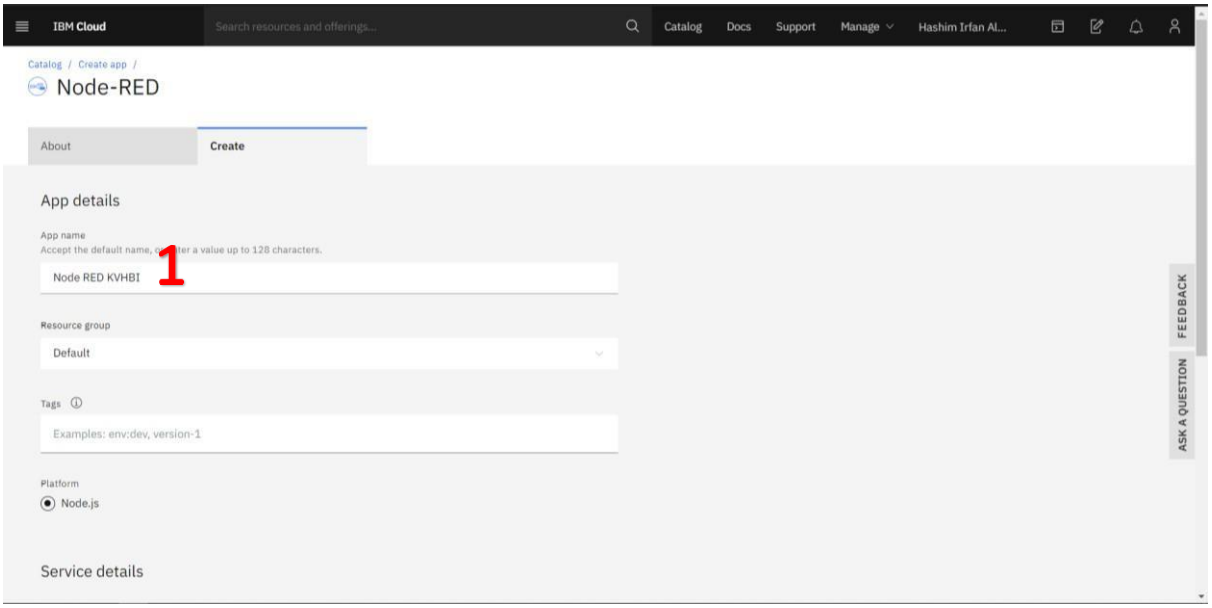


Click on the Node-RED App tile [2].

This will show you an overview of the Starter Kit and what it provides.



Click on Create [1].



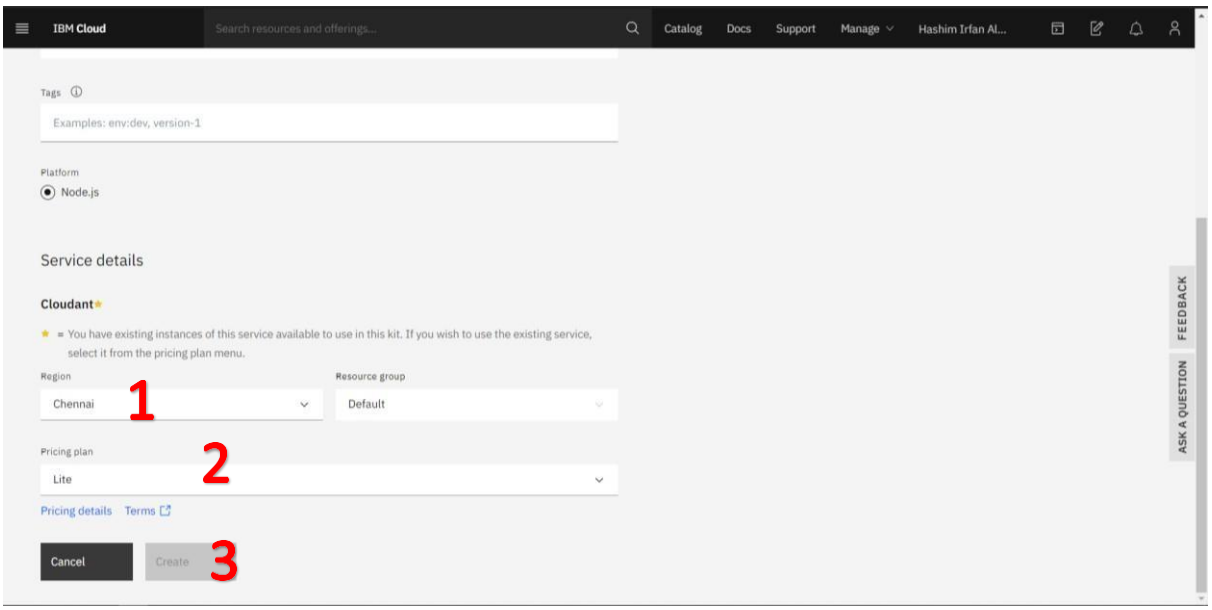
Now you need to configure the Node-RED Starter application.

On the App details page, a randomly generated name will be suggested – Node RED KVHBI in the screenshot above. Either accept that default name or provide a unique name for your application [1]. This will become part of the application URL. Note: If the name is not unique, you will see an error message and you must enter a different name before you can continue.

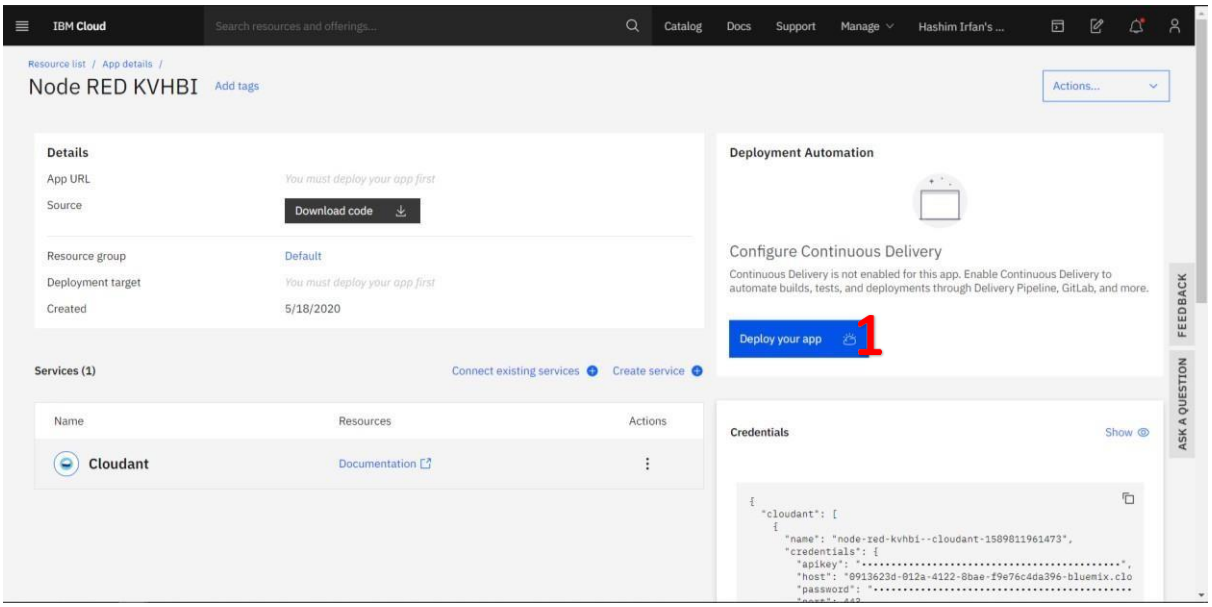
The Node-RED Starter application requires an instance of the Cloudant database service to store your application flow configuration. To do this,

Select the region [1] the service should be created in and what pricing plan it should use. You can only have one Cloudant instance using the Lite plan and You can have more than one Node-RED Starter application using the same Cloudant service instance.

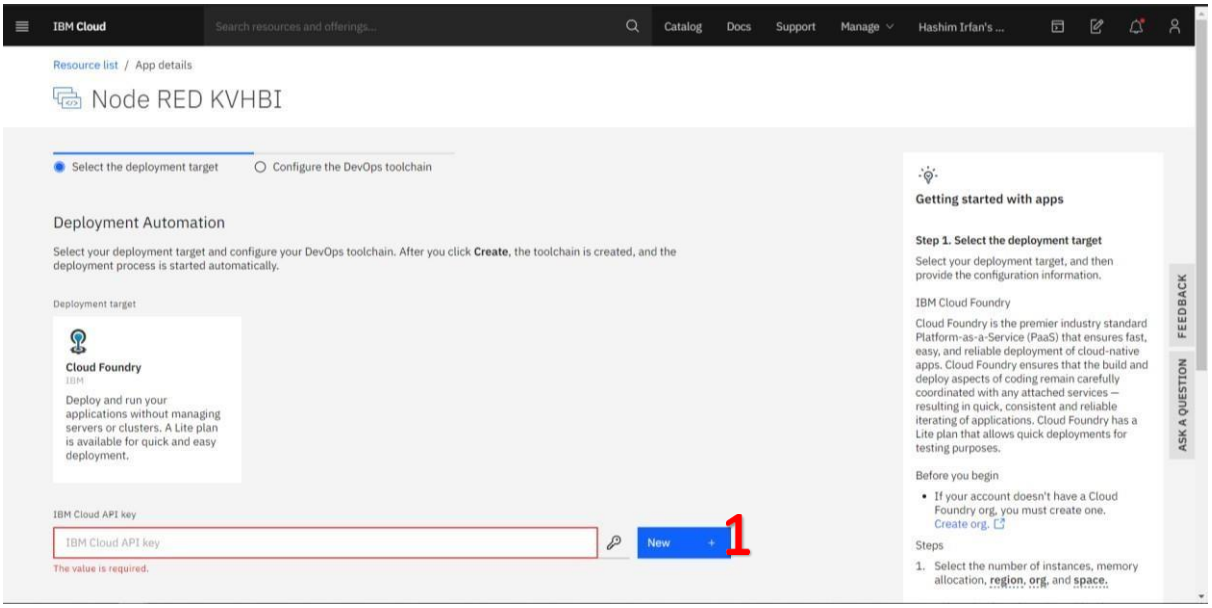
If you have already got an instance, you will be able to select it from the Pricing plan select box [2]. Click the Create button [3] to continue. This will create your application, but it is not yet deployed to IBM Cloud.



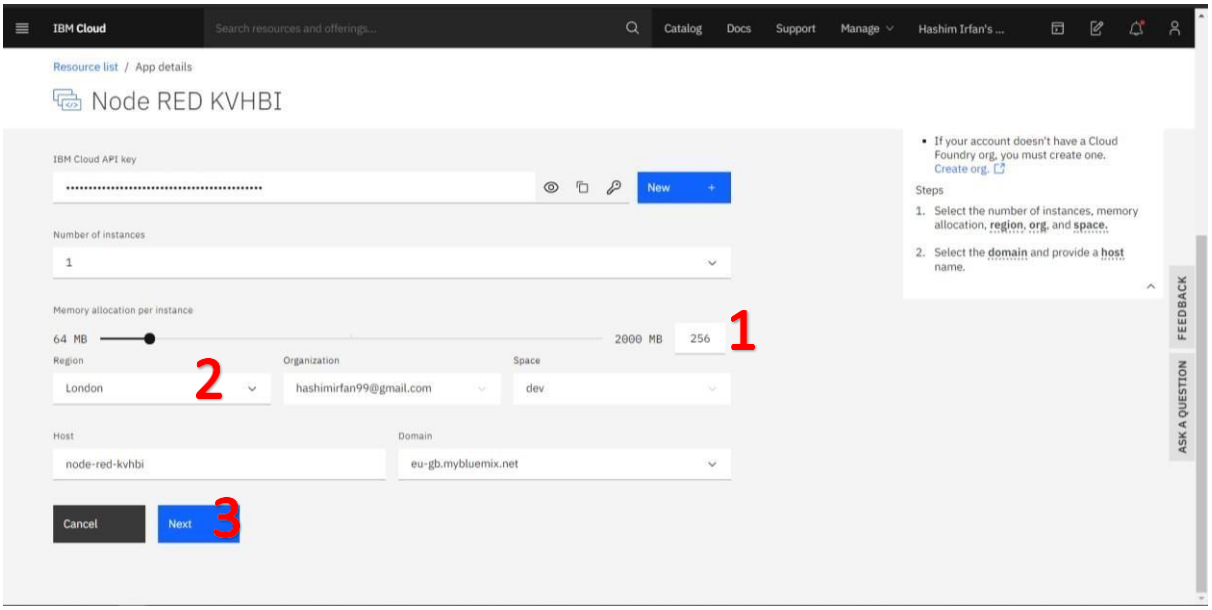
At this point, you have created the application and the resources it requires, but you have not deployed it anywhere to run, so this step shows how to setup the Continuous Delivery feature that will deploy your application into the Cloud Foundry space of IBM Cloud. Click on Deploy your App[1]



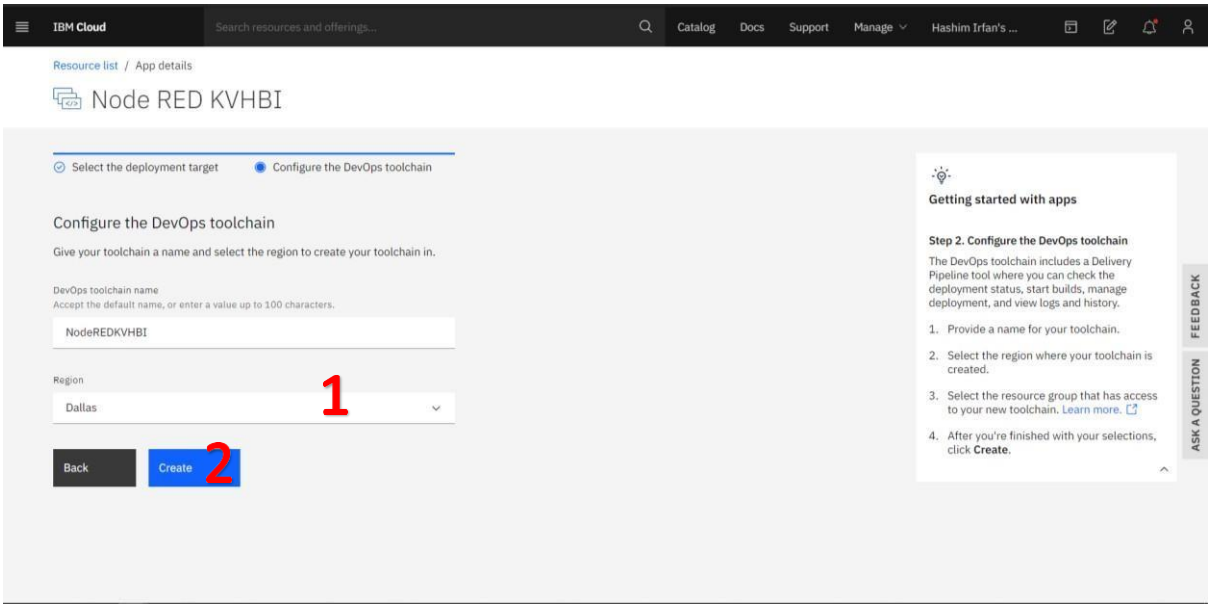
You will need to create an IBM Cloud API key to allow the deployment process to access your resources. Click the New button (1) to create the key. A message dialog will appear. Read what it says and then confirm and close the dialog.



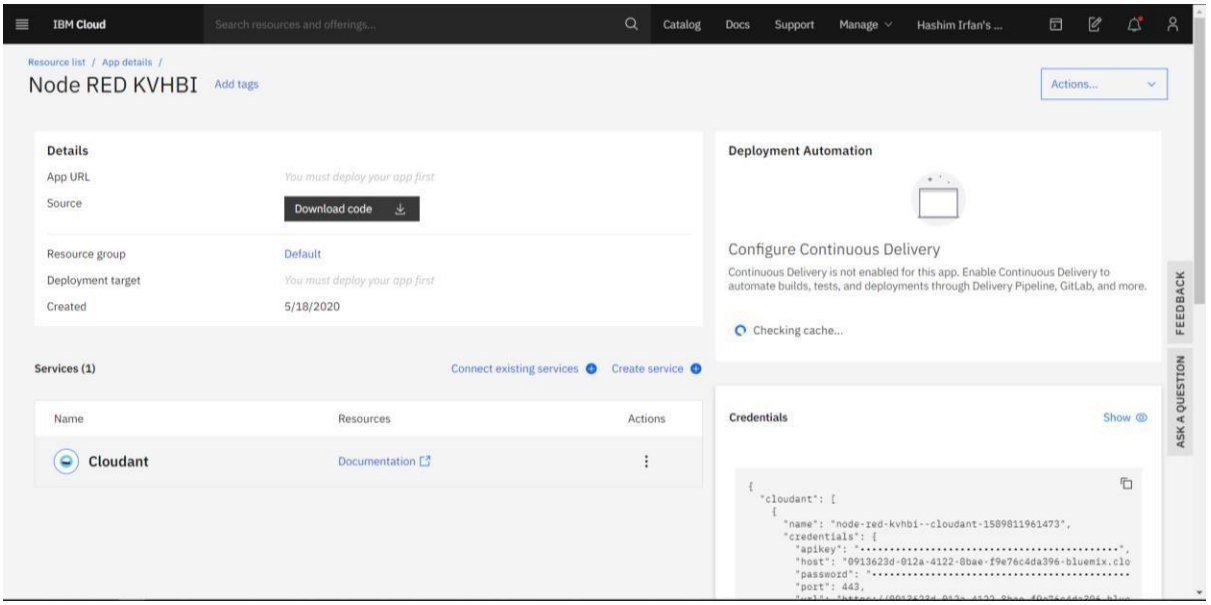
After creating the API Key, Increase the Memory allocation per instance slider [1] to 256MB. If you do not increase the memory allocation, your Node-RED application might not have sufficient memory to run successfully. The Node-RED Starter kit only supports deployment to the Cloud Foundry space of IBM Cloud. Select the region [2] to deploy your application to. This should match the region you created your Cloudant instance in.Click Next [3].



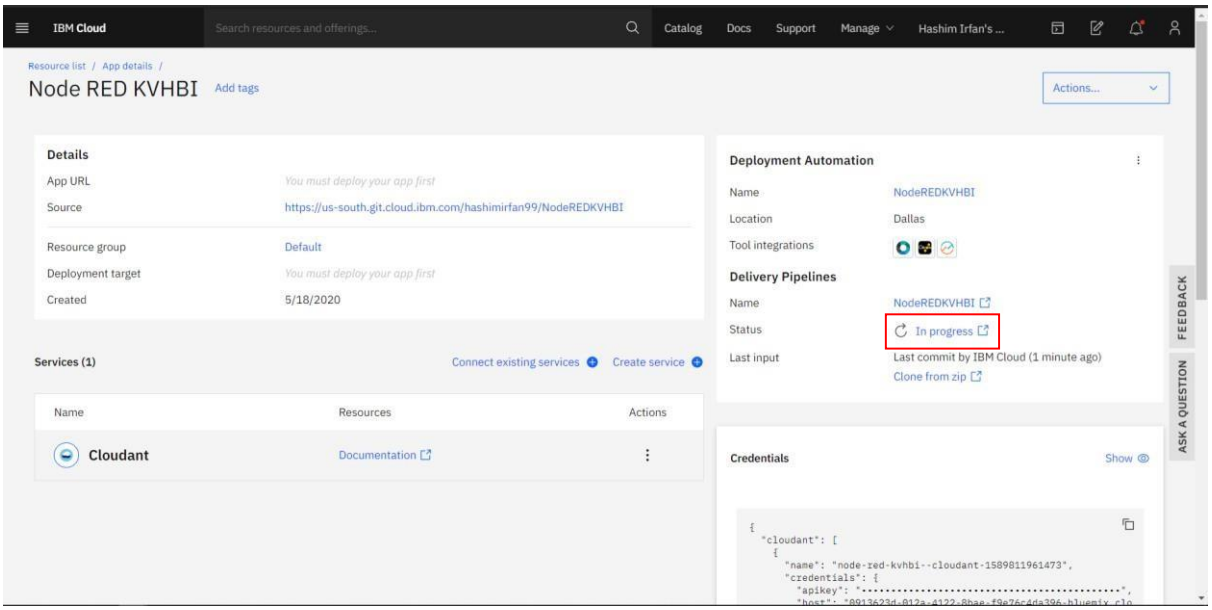
Now, Select the region [1] to create the DevOps toolchain and then Click Create [2].



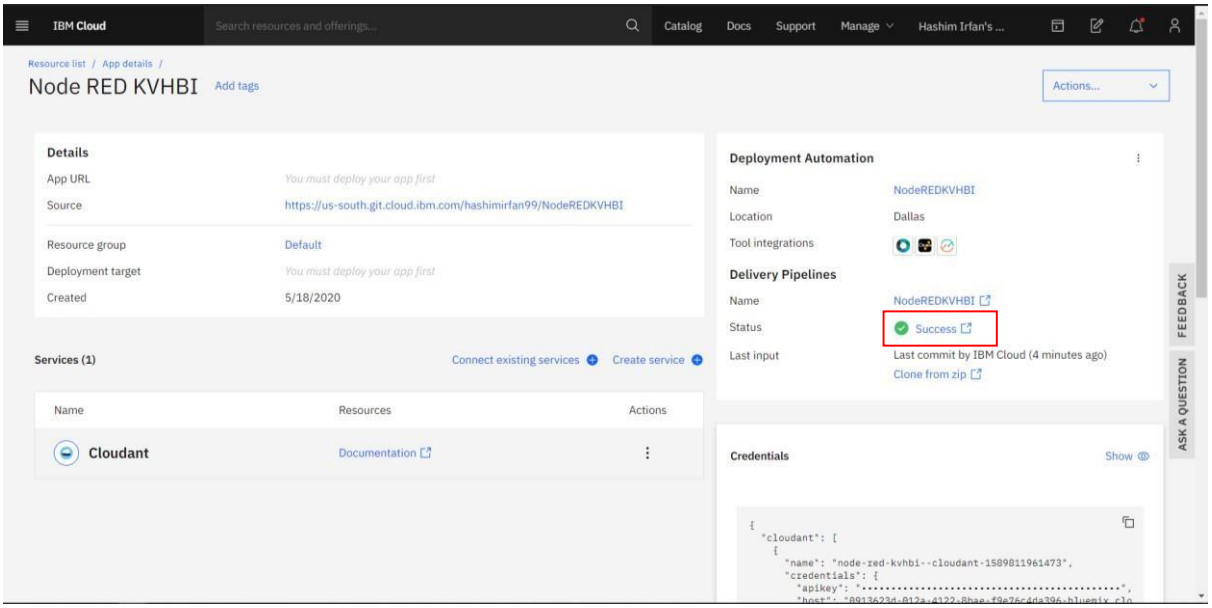
This will take you back to the application details page.



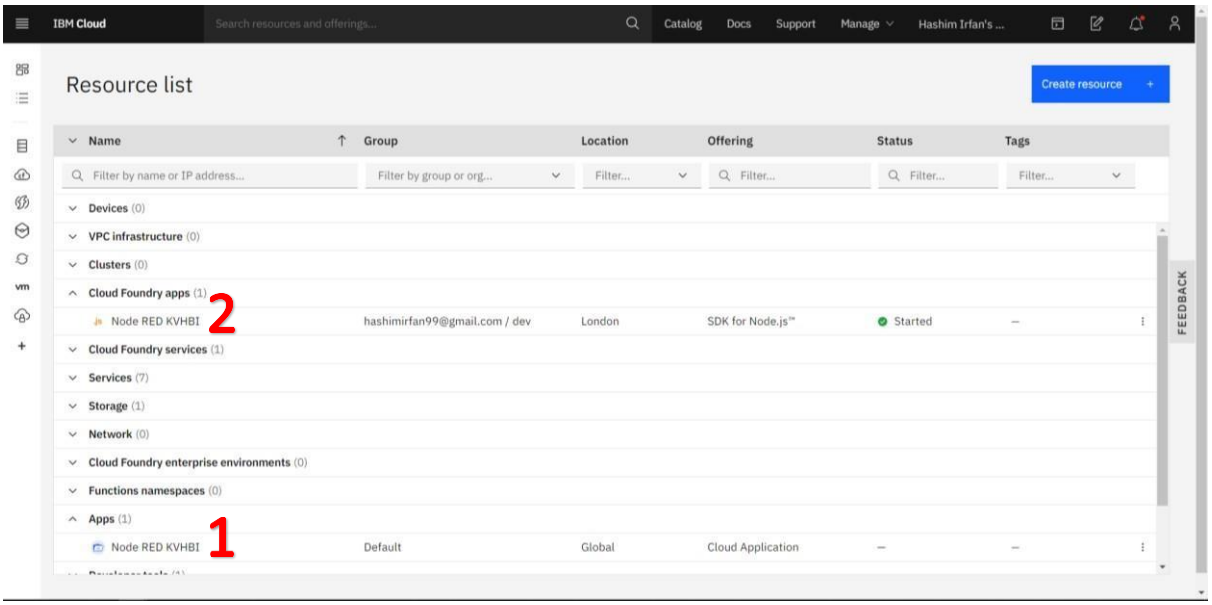
The Continuous Delivery section will refresh with the details of your newly created Toolchain. The Status field of the Delivery Pipeline will show In progress. That means your application is still being built and deployed.



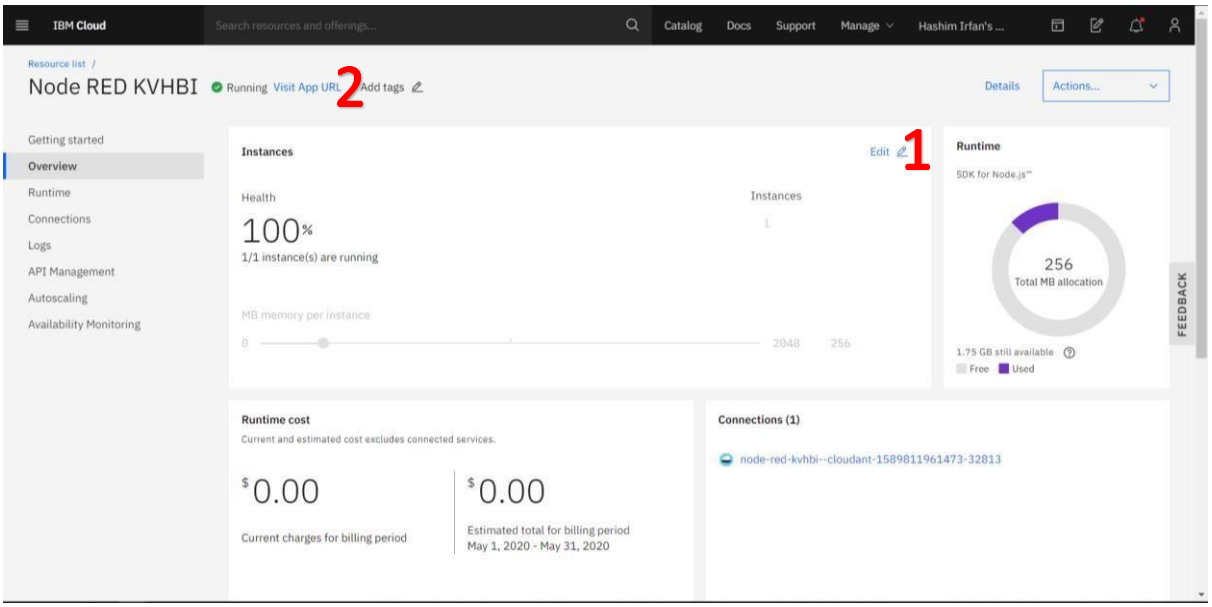
The Deploy stage will take a few minutes to complete. Eventually the Deploy stage will go green to show it has passed. This means your Node-RED Starter application is now running.



Now that you've deployed your Node-RED application, let's open it up! Open your IBM Cloud Resource list. You will see your newly created Node-RED Application listed under the Apps section [1]. You will also see a corresponding entry under the Cloud Foundry apps section [2].

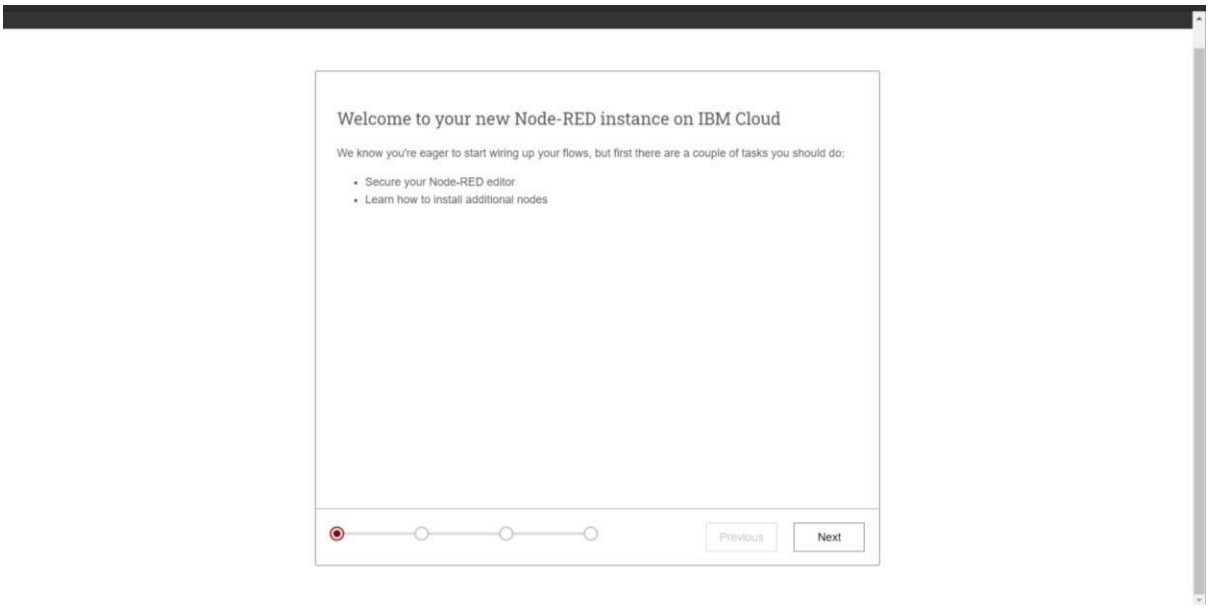


Click on this Cloud Foundry app entry to go to your deployed application's details page.

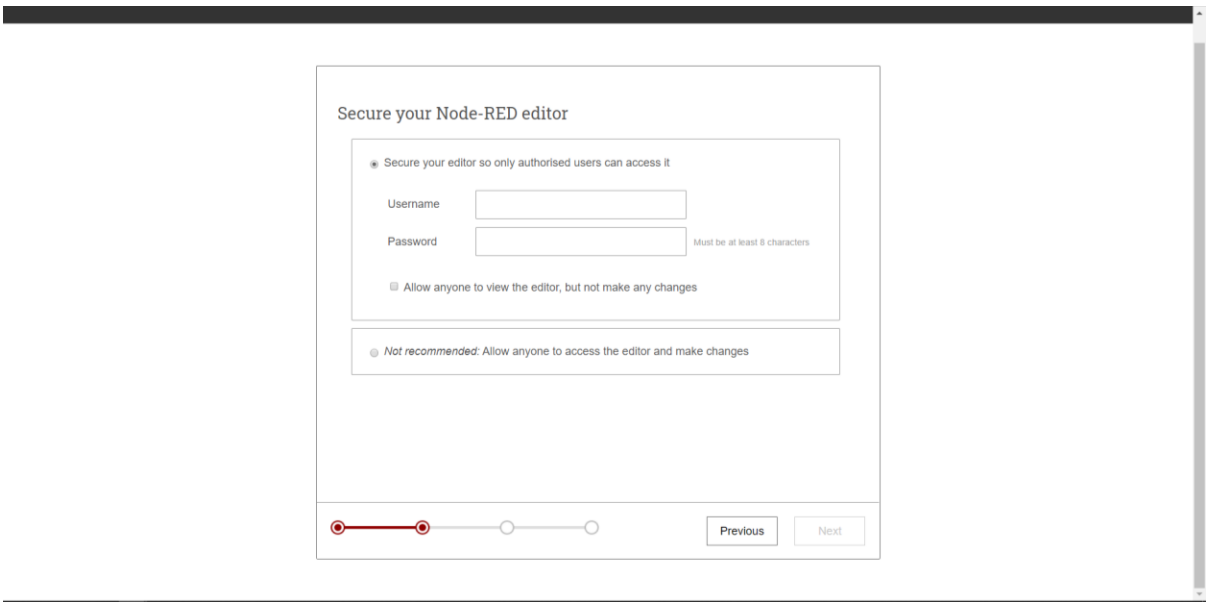


Special Cases: If your Runtime Instance is running full (0MB Free), Click on Edit[1] and reduce memory per instance to 128mb.

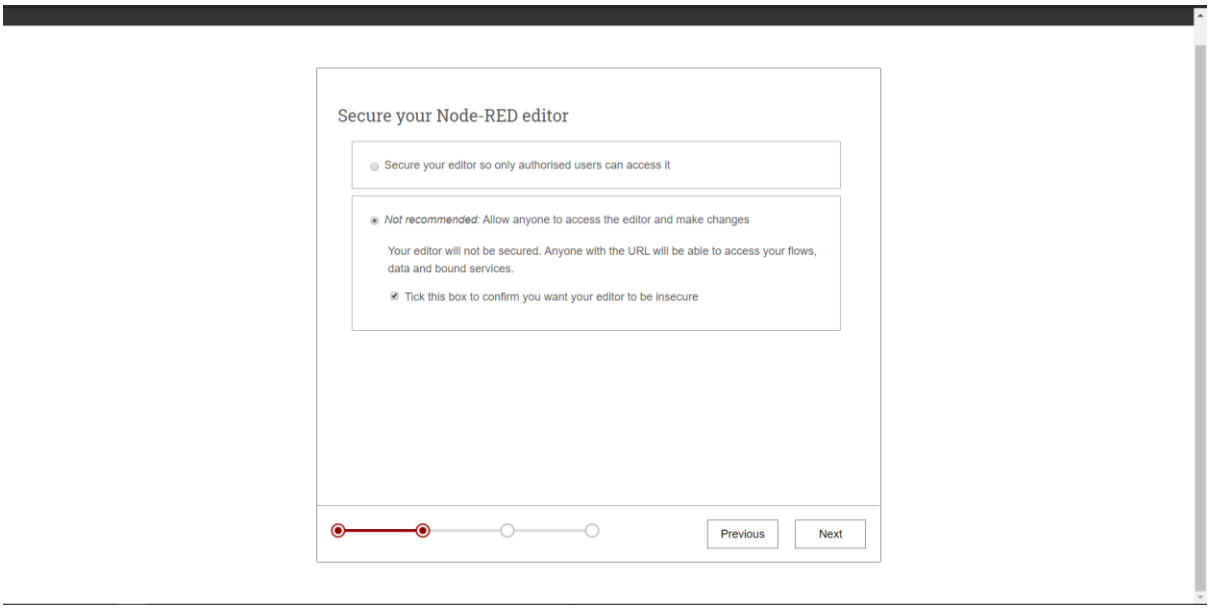
If you have Free space on your runtime skip the previous step[1] and Click on Visit App URL[2]



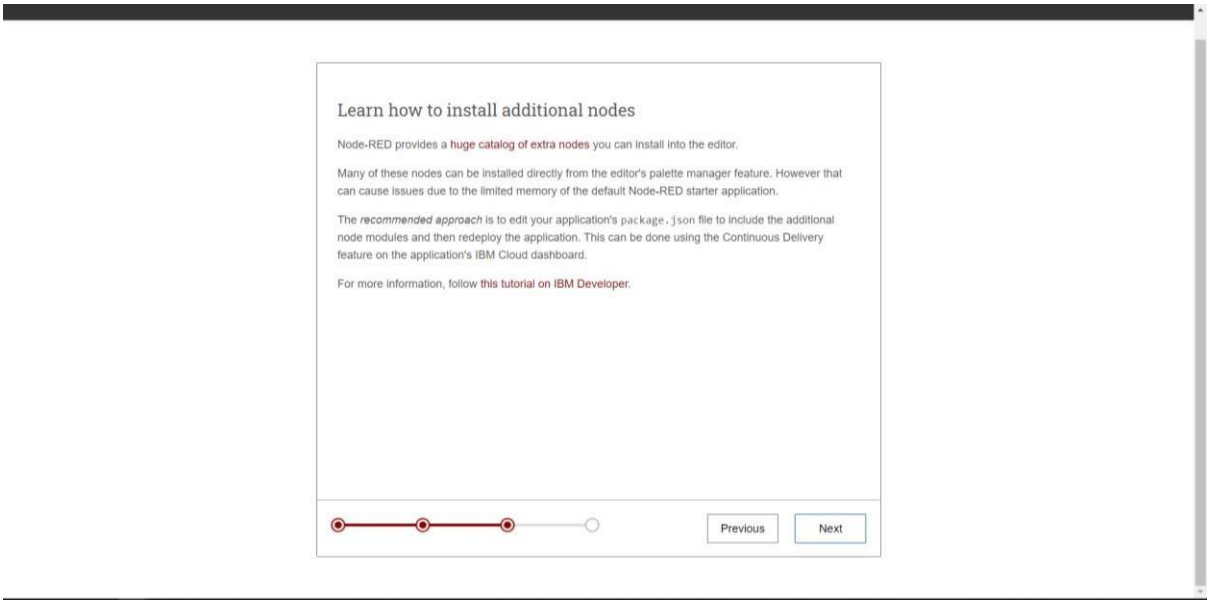
Click Next.



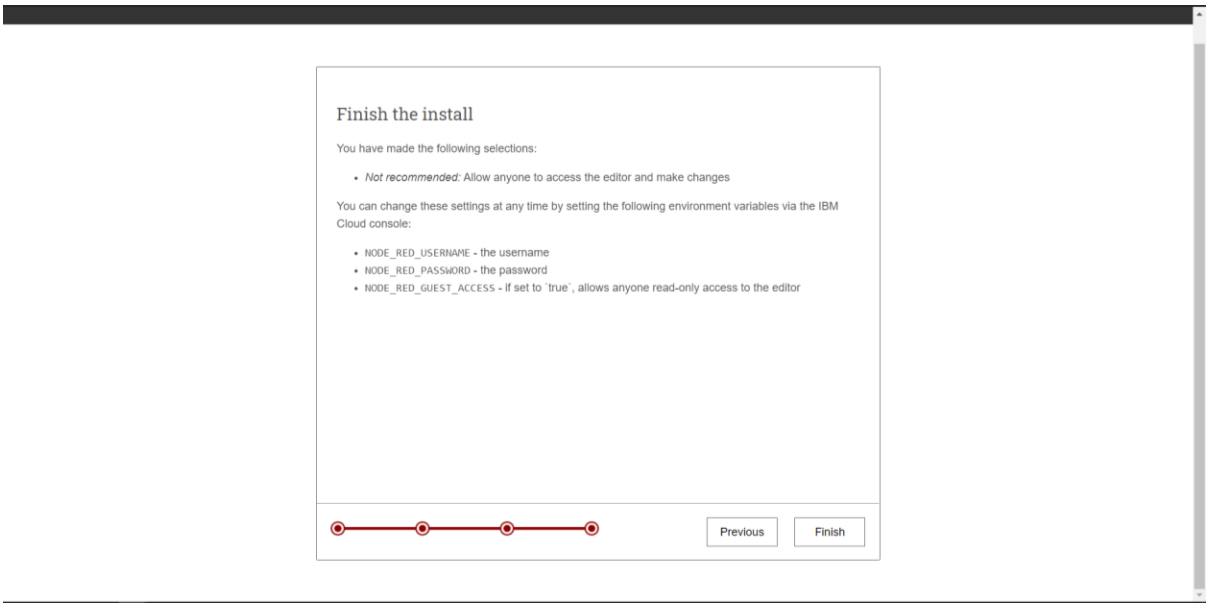
You can choose to Secure your Node-RED editor by providing a username and password. I am selecting the other option which is Allow anyone to access the editor and make changes.



Tick the box, and click Next.

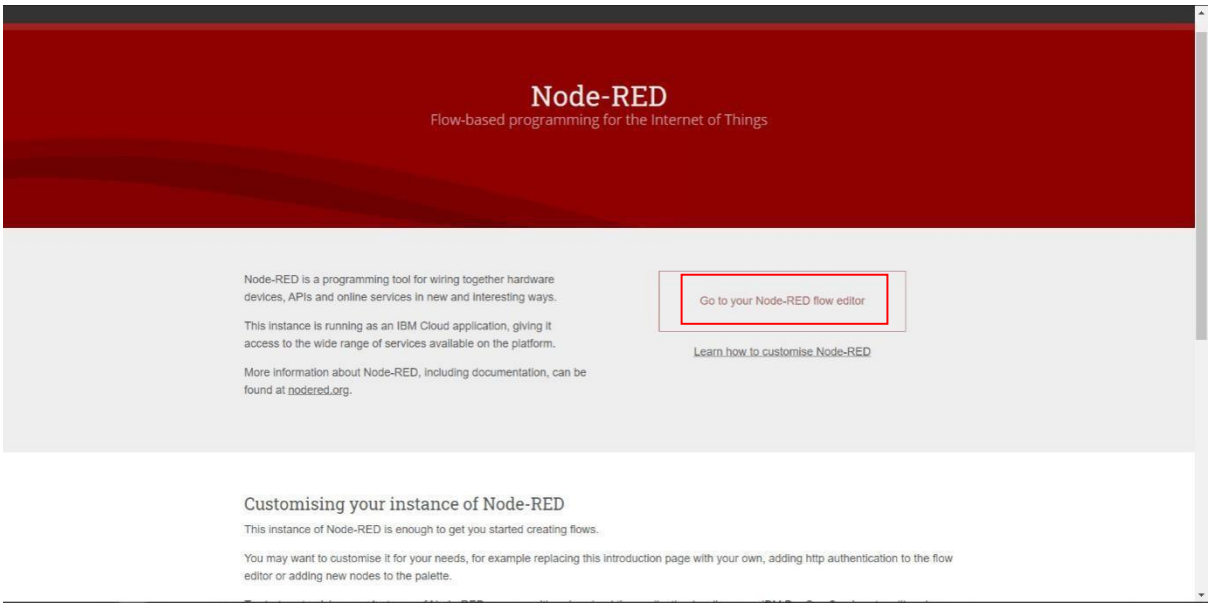


Click Next.



The final screen summarizes the options you've made and highlights the environment variables you can use to change the options in the future. Click Finish to proceed. Node-RED will save your changes and then load the main application.

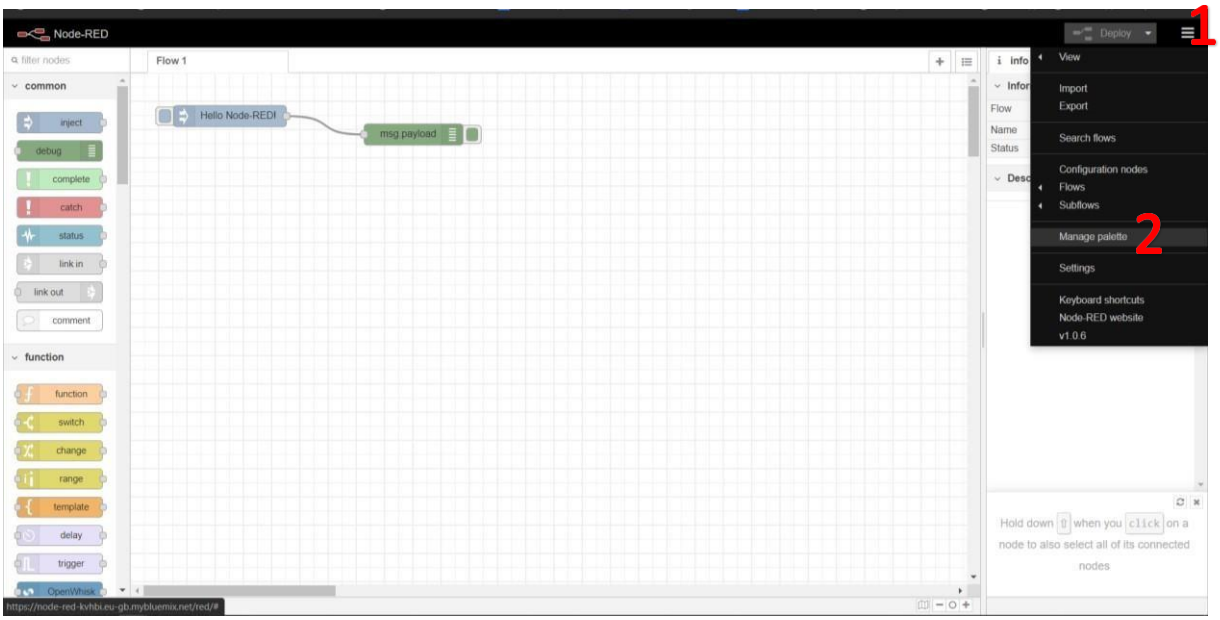
From here you can click the Go to your Node-RED flow editor button to open the editor.



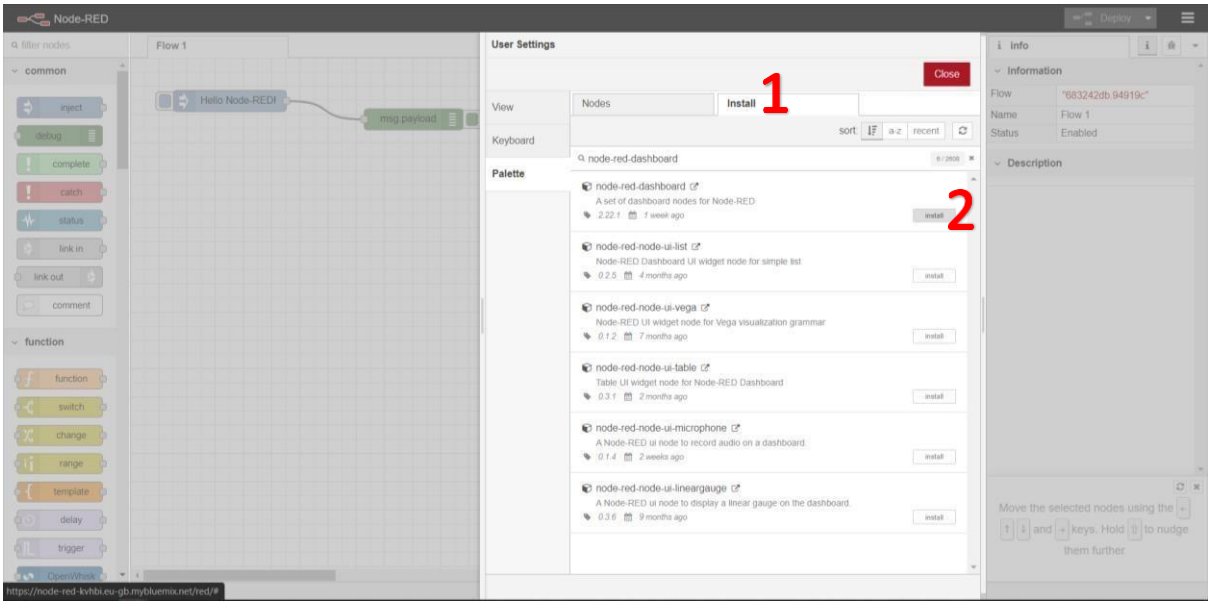
The Node-RED editor opens showing the default flow.

6. Configure the nodes and Build A Web Dashboard in Node-RED

To add Nodes to integrate Assistant, click [1] and then select Manage Palette [2]

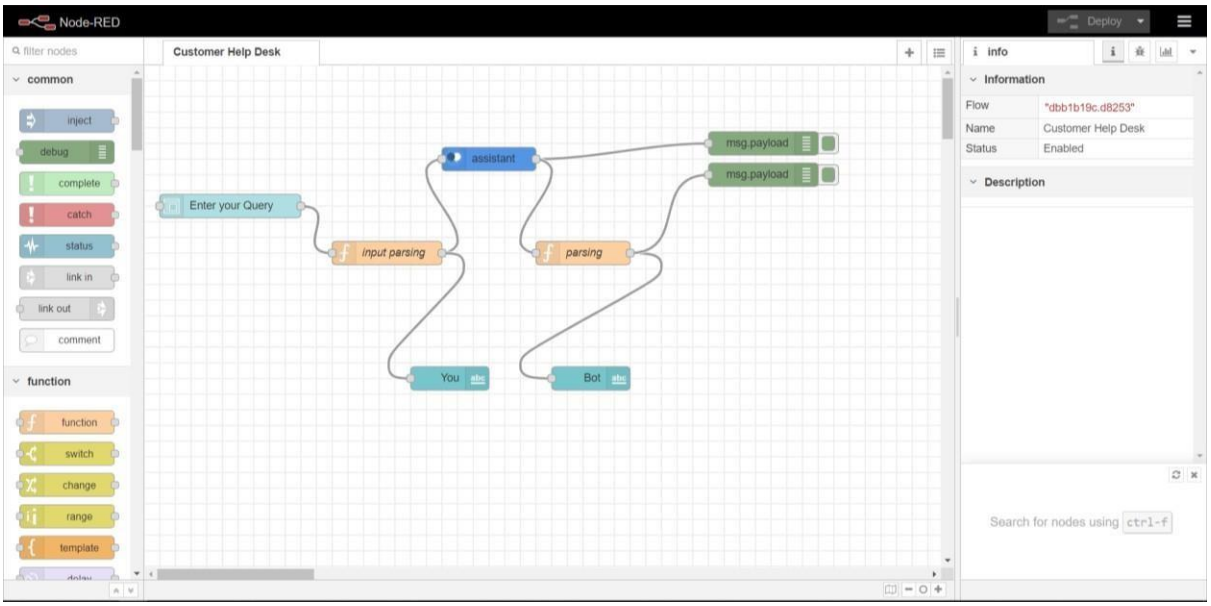


Go to Install Tab [1] and search for node-red-dashboard and Install [2] it.

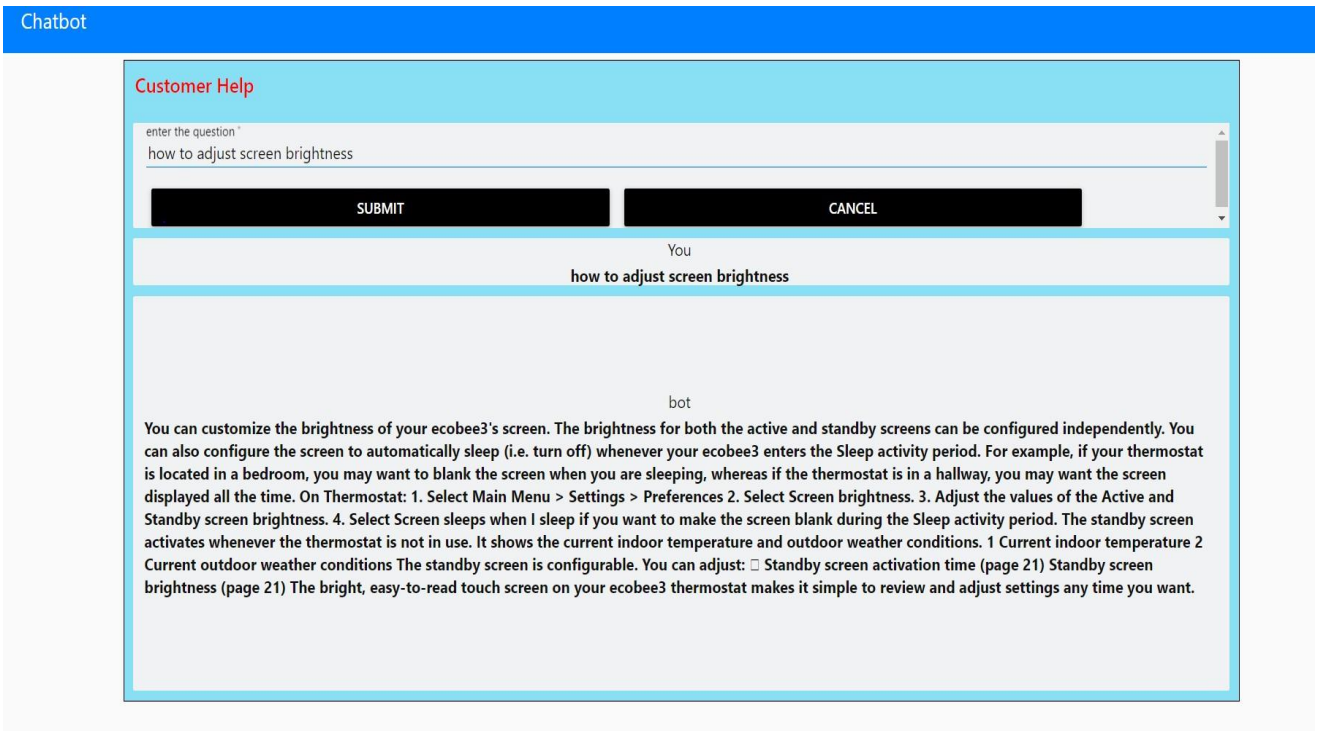


Using the nodes in the palette, Configure the required nodes and build web dashboard in Node-RED.

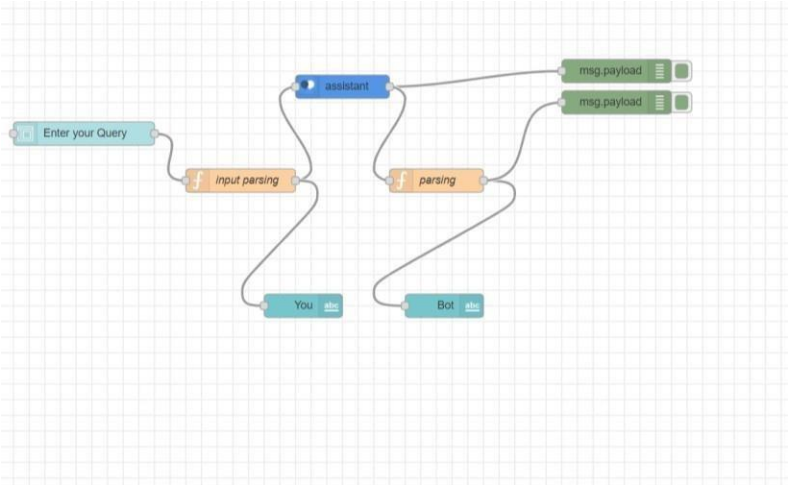
7. Deploy and Run the application



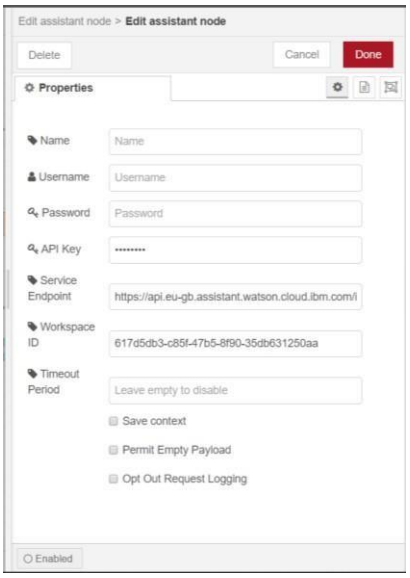
After Deploying the App, Run it


Chatbot
Customer Help
enter the question *
how to adjust screen brightness
SUBMIT
CANCEL
You
how to adjust screen brightness
bot
You can customize the brightness of your ecobee3's screen. The brightness for both the active and standby screens can be configured independently. You can also configure the screen to automatically sleep (i.e. turn off) whenever your ecobee3 enters the Sleep activity period. For example, if your thermostat is located in a bedroom, you may want to blank the screen when you are sleeping, whereas if the thermostat is in a hallway, you may want the screen displayed all the time. On Thermostat: 1. Select Main Menu > Settings > Preferences 2. Select Screen brightness. 3. Adjust the values of the Active and Standby screen brightness. 4. Select Screen sleeps when I sleep if you want to make the screen blank during the Sleep activity period. The standby screen activates whenever the thermostat is not in use. It shows the current indoor temperature and outdoor weather conditions. 1 Current indoor temperature 2 Current outdoor weather conditions The standby screen is configurable. You can adjust: □ Standby screen activation time (page 21) Standby screen brightness (page 21) The bright, easy-to-read touch screen on your ecobee3 thermostat makes it simple to review and adjust settings any time you want.

5. FLOWCHART



First, Add a Form Node. Connect it with a function node and name it input parsing. To that add a Text Node and name it You. To the input parsing node, add assistant node.



Enter the API Key, Service Endpoint (URL) and Workspace ID (Skill ID) from Step 4 and click Done. To the assistant node add debug node. Add another function node to assistant node and name it parsing. Add text node to parsing node and name it Bot. Add another debug node to parsing node.

For Function Node named input parsing, use the code below:

```
msg.payload=msg.payload.text;
return msg;
```

For Function Node named parsing, use the code below:

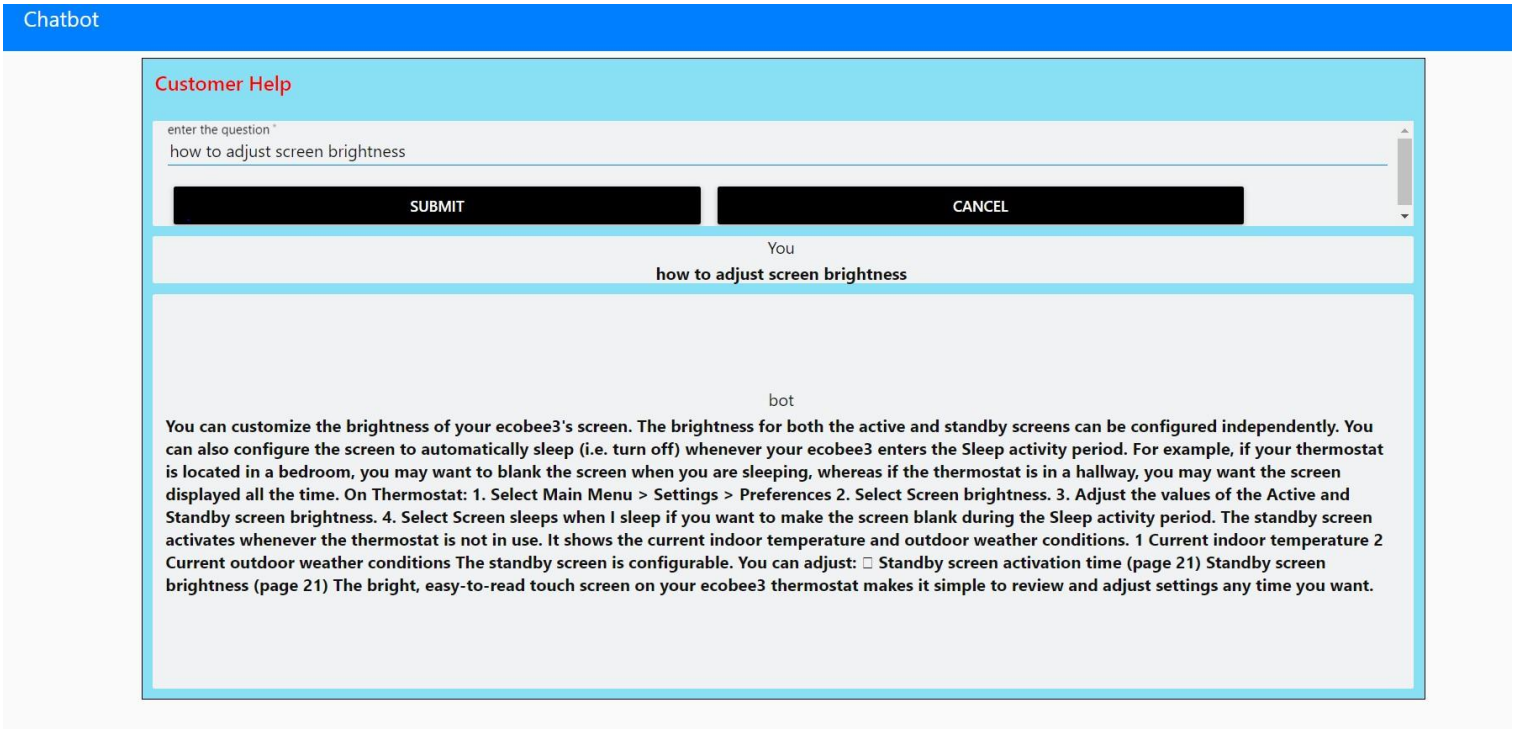
```
msg.payload.text="";
if(msg.payload.context.webhook_result_1){
    for(var i in msg.payload.context.webhook_result_1.results){

msg.payload.text=msg.payload.text+"\n"+msg.payload.context.webhook_result_1.results[i].text;
}
    msg.payload=msg.payload.text;
}
```

else

msg.payload = msg.payload.output.text[0];

return msg;

6. RESULT



Chatbot Link:
https://node-red-ifzfu.eu-gb.mybluemix.net/ui/#!/0?socketid=xz-UgfIhk-HZGtpIAABa

Explanation Video Link:
https://youtu.be/sRKkOALM_do

7. ADVANTAGES & DISADVANTAGES

Advantages:

- Faster Customer Service
- Increased Customer Satisfaction
- Lower Labour Costs
- Variety of Uses
- Data collection
- 24-7 availability
- Multiple Customer Handling

Disadvantages:

- Limited Responses for Customers
- Customers Could Become Frustrated
- Maintenance
- They aren’t human
- Time-Consuming

8. APPLICATIONS

A Product or Software Company Customer Help Desk

9. CONCLUSION

An Intelligent Customer Helpdesk with Smart Document Understanding is made using various IBM Services like IBM Watson Discovery, IBM Watson and IBM Cloud Function.

10. FUTURE SCOPE

A More Human Friendly Chatbot, or a personalized Chatbot is to be expected.

11. BIBILOGRAPHY

APPENDIX

A. Source Code

disco_action.js(Cloud Function Code)

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

skill-Customer-Care-Sample-Skill.json (Watson Assistant Skill Code)

```
{

  "intents": [

    {

      "intent": "General_Security_Assurance",

      "examples": [

        {

          "text": "How do I know this chat is not a scam?"

        },

        {

          "text": "How do I know if my card info is kept safe?"

        },

        {

          "text": "Do you have protection against hacks?"

        },

        {
```

```
  "text": "Customer info secured?"
},
{
  "text": "Conversation secure?"
},
{
  "text": "Can I be sure that my stored card details are held securely?"
},
{
  "text": "Are you safe?"
},
{
  "text": "Are you safe on hackers?"
},
{
  "text": "Why would I trust this chat?"
},
{
  "text": "What safety measures are used to ensure that chat is secure?"
},
{
  "text": "What makes this chat trustworthy?"
},
```

```
{
  "text": "Secure conversation?"
},
{
  "text": "What guards do you have in place to protect your customer's data?"
},
{
  "text": "Will my conversation remain secured?"
},
{
  "text": "Protection hacks?"
},
{
  "text": "Precautions hackers?"
},
{
  "text": "Is this conversation secure?"
},
{
  "text": "Is this an official chat?"
},
{
  "text": "Is this a trusted chat site?"
```

```
  },
  {
    "text": "Is there a reason to distrust this chat?"
  },
  {
    "text": "Is my payment information secure?"
  },
  {
    "text": "Is my card info secure?"
  },
  {
    "text": "In what way can I tell if my plastic is safe?"
  },
  {
    "text": "I'd like proof of a secure chat"
  },
  {
    "text": "How safe is my data?"
  },
  {
    "text": "How is my credit card info secured?"
  }
],
```

```
  "description": "Express concerns about the security of the bot."
},
{
  "intent": "General_Positive_Feedback",
  "examples": [
    {
      "text": "Can't believe you are that good"
    },
    {
      "text": "You've been so helpful :)"
    },
    {
      "text": "You're a genius!"
    },
    {
      "text": "You the man"
    },
    {
      "text": "You gave me exactly what I need!"
    },
    {
      "text": "You are wonderful"
    },
```

```
{
  "text": "You are the best"
},
{
  "text": "You are great"
},
{
  "text": "You are awesome"
},
{
  "text": "This is so cool"
},
{
  "text": "This is great"
},
{
  "text": "This is good"
},
{
  "text": "Thank you"
},
{
  "text": "Ok thank you"
```

```
    },
    {
      "text": "Love your work"
    },
    {
      "text": "I'm looking forward to working with you again! :)"
    },
    {
      "text": "I like what you did there! :)"
    },
    {
      "text": "How cool is this?"
    },
    {
      "text": "Brilliant!"
    }
  ],
  "description": "Express positive sentiment or gratitude."
},
{
  "intent": "General_Greetings",
  "examples": [
    {
```

```
  "text": "What's up?"
},
{
  "text": "Good day"
},
{
  "text": "Good evening"
},
{
  "text": "Good morning"
},
{
  "text": "Good to see you"
},
{
  "text": "Greetings"
},
{
  "text": "Have you been well?"
},
{
  "text": "Hello Agent"
},
```

```
{
  "text": "Hello I am looking for some help here"
},
{
  "text": "Hello"
},
{
  "text": "Hey how are you doing"
},
{
  "text": "Hey there all"
},
{
  "text": "Hey there"
},
{
  "text": "Hey twin"
},
{
  "text": "Hey you"
},
{
  "text": "Hi advisor"
```

```
},
{
  "text": "Hi there"
},
{
  "text": "How are things going?"
},
{
  "text": "How are you today?"
},
{
  "text": "How have you been?"
},
{
  "text": "How is it going?"
},
{
  "text": "How r u?"
},
{
  "text": "Looking good eve"
},
{
```

```
      "text": "Ok take me back"
    },
    {
      "text": "What's new?"
    },
    {
      "text": "Who is this?"
    },
    {
      "text": "You there"
    }
  ],
  "description": "Greet the bot."
},
{
  "intent": "Thanks",
  "examples": [
    {
      "text": "thank you"
    },
    {
      "text": "thanks"
    },
```

```
    {
      "text": "thanks for the help"
    }
  ],
  "description": ""
},
{
  "intent": "Greetings",
  "examples": [
    {
      "text": "Hello"
    },
    {
      "text": "Good morning"
    },
    {
      "text": "Hii"
    }
  ],
  "description": ""
},
{
  "intent": "General_Negative_Feedback",
```

```
"examples": [
  {
    "text": "You are having delusions"
  },
  {
    "text": "Why are you stupid?"
  },
  {
    "text": "Why are you so annoying?"
  },
  {
    "text": "Stupid"
  },
  {
    "text": "Robots are stupid"
  },
  {
    "text": "Robots are boring"
  },
  {
    "text": "Quit annoying me"
  },
  {
```

```
  "text": "It is annoying"
},
{
  "text": "I hate you"
},
{
  "text": "I hate this!"
},
{
  "text": "I do not like you"
},
{
  "text": "Hate you"
},
{
  "text": "Everyone hates you"
},
{
  "text": "Do not like you?"
},
{
  "text": "You're too stupid"
},
```

```
    {
      "text": "You're really frustrating"
    },
    {
      "text": "You're really irritating"
    },
    {
      "text": "You do not seem smart"
    },
    {
      "text": "You are very frustrating"
    },
    {
      "text": "You are on my nerves"
    }
  ],
  "description": "Express unfavorable feedback."
},
{
  "intent": "usermanual",
  "examples": [
    {
      "text": "customizing thermostat"
```

```
    },
    {
      "text": "how to adjust date and time?"
    },
    {
      "text": "how to configure thermostat?"
    },
    {
      "text": "how to turn on heater?"
    },
    {
      "text": "basic functions"
    },
    {
      "text": "hvac"
    }
  ],
  "description": "User wants to ask information regarding the heater"
},
{
  "intent": "General_Jokes",
  "examples": [
    {
```

```
  "text": "Do you have a joke?"
},
{
  "text": "Can you tell me a joke?"
},
{
  "text": "Can you tell a joke?"
},
{
  "text": "Are there jokes?"
},
{
  "text": "Another joke"
},
{
  "text": "I'm bored"
},
{
  "text": "One more joke"
},
{
  "text": "Surprise me with something hilarious"
},
```

```
{
  "text": "Tell me a joke"
},
{
  "text": "Tell me something funny"
},
{
  "text": "What do you do for fun?"
},
{
  "text": "What is your favorite joke?"
},
{
  "text": "I want a joke"
},
{
  "text": "I am getting bored"
},
{
  "text": "Do you like humor?"
},
{
  "text": "Do you like fun?"
```

```
        },
        {
          "text": "Do you have humor?"
        }
      ],
      "description": "Request a joke."
    }
  ],
  "entities": [],
  "metadata": {
    "api_version": {
      "major_version": "v2",
      "minor_version": "2018-11-08"
    },
    "from-sample": true
  },
  "webhooks": [
    {
      "url": "https://eu-
gb.functions.cloud.ibm.com/api/v1/web/rahildesai.rd99%40gmail.com_dev/default/usermanual.json",
      "name": "main_webhook",
      "headers": []
    }
  ],
```

```
"dialog_nodes": [
  {
    "type": "response_condition",
    "output": {
      "text": {
        "values": [
          "pls try again"
        ],
        "selection_policy": "sequential"
      }
    },
    "parent": "node_1_1589645573304",
    "conditions": "anything_else",
    "dialog_node": "response_5_1589645777811",
    "previous_sibling": "response_6_1589645774502"
  },
  {
    "type": "response_condition",
    "output": {
      "generic": [
        {
          "values": [
            {
```

```
              "text": "$webhook_result_1"
            }
          ],
          "response_type": "text",
          "selection_policy": "sequential"
        }
      ]
    },
    "parent": "node_1_1589645573304",
    "conditions": "$webhook_result_1",
    "dialog_node": "response_6_1589645774502"
  },
  {
    "type": "standard",
    "title": "user_manual",
    "actions": [
      {
        "name": "main_webhook",
        "type": "webhook",
        "parameters":
{
          "input": "<?input.text?>"
        },
        "result_variable": "webhook_result_1"
```

```
    }
  ],
  "metadata": {
    "_customization": {
      "mcr": true
    }
  },
  "conditions": "#usermanual",
  "dialog_node": "node_1_1589645573304",
  "previous_sibling": "node_9_1589206004629"
},
{
  "type": "standard",
  "title": "Anything else",
  "output": {
    "generic": [
      {
        "values": [
          {
            "text": "I didn't understand. You can try rephrasing."
          },
          {
            "text": "Can you reword your statement? I'm not understanding."
```

```
          },
          {
            "text": "I didn't get your meaning."
          }
        ],
        "response_type": "text",
        "selection_policy": "random"
      }
    ]
  },
  "conditions": "anything_else",
  "dialog_node": "Anything else",
  "previous_sibling": "node_6_1589206224204",
  "disambiguation_opt_out": true
},
{
  "type": "standard",
  "title": "Greeting",
  "output": {
    "generic": [
      {
        "values": [
          {
```

```
            "text": "Hello, hope you are having a good day, how can I help you?"
          }
        ],
        "response_type": "text",
        "selection_policy": "sequential"
      }
    ]
  },
  "conditions": "#Greetings || #General_Greetings",
  "dialog_node": "node_9_1589206004629",
  "previous_sibling": "Welcome"
},
{
  "type": "standard",
  "title": "Thanks",
  "output": {
    "generic": [
      {
        "values": [
          {
            "text": "Happy to help always"
          },
          {
```

```
            "text": "Anything more could I help?"
          },
          {
            "text": "Have a good day!"
          },
          {
            "text": "Anytime"
          }
        ],
        "response_type": "text",
        "selection_policy": "random"
      }
    ]
  },
  "conditions": "#Thanks || #General_Positive_Feedback",
  "dialog_node": "node_6_1589206224204",
  "previous_sibling": "node_1_1589645573304"
},
{
  "type": "standard",
  "title": "Welcome",
  "output": {
    "generic": [
```

```
        {
          "values": [
            {
              "text": "Hello. How can I help you?"
            },
            {
              "text": "Hi, my name is SmartBot, happy to provide help regarding user manual"
            }
          ],
          "response_type": "text",
          "selection_policy": "random"
        }
      ]
    },
    "conditions": "welcome",
    "dialog_node": "Welcome"
  }
],
"counterexamples": [],
"system_settings": {
  "off_topic": {
    "enabled": true
  },
```

```
    "disambiguation": {
      "prompt": "Did you mean:",
      "enabled": true,
      "randomize": true,
      "max_suggestions": 5,
      "suggestion_text_policy": "title",
      "none_of_the_above_prompt": "None of the above"
    },
    "system_entities": {
      "enabled": true
    },
    "human_agent_assist": {
      "prompt": "Did you mean:"
    },
    "spelling_auto_correct": true
  },
  "learning_opt_out": false,
  "name": "assistant",
  "language": "en",
  "description": ""
}
```

B. Reference

https://developer.ibm.com/patterns/enhance-customer-help-desk-with-smart-document- understanding/

1. https://github.com/IBM/watson-discovery-sdu-with-assistant

2. https://www.youtube.com/watch?v=-yniuX-Poyw&feature=youtu.be

3. https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/

THE END