PROJECT REPORT

Intelligent Customer Help Desk with Smart Document Understanding

Project By :-
Ishika Garg
gargishika1998@gmail.com

INDEX

1 INTRODUCTION

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:

To create a chat-bot which can answer simple questions as well as return more and more accurate answers.

2. LITERATURE SURVEY

2.1. EXISTING PROBLEM:

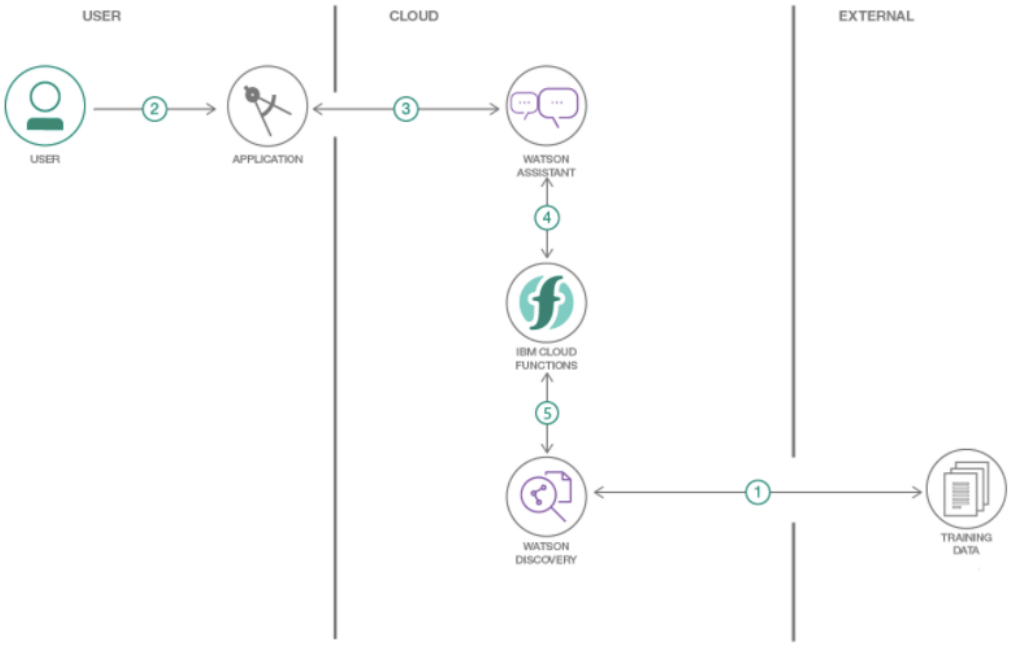
The typical customer care chat-bot can answer simple questions, such as store locations and hours, directions, and maybe even making appointments. When a question falls outside of the scope of the pre-determined question set, the option is typically to tell the customer the question isn't valid or offer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

2.2. PROPOSED SOLUTION:

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:



1. The document is annotated using Watson Discovery SDU

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

3.2. HARDWARE/SOFTWARE DESIGNING:

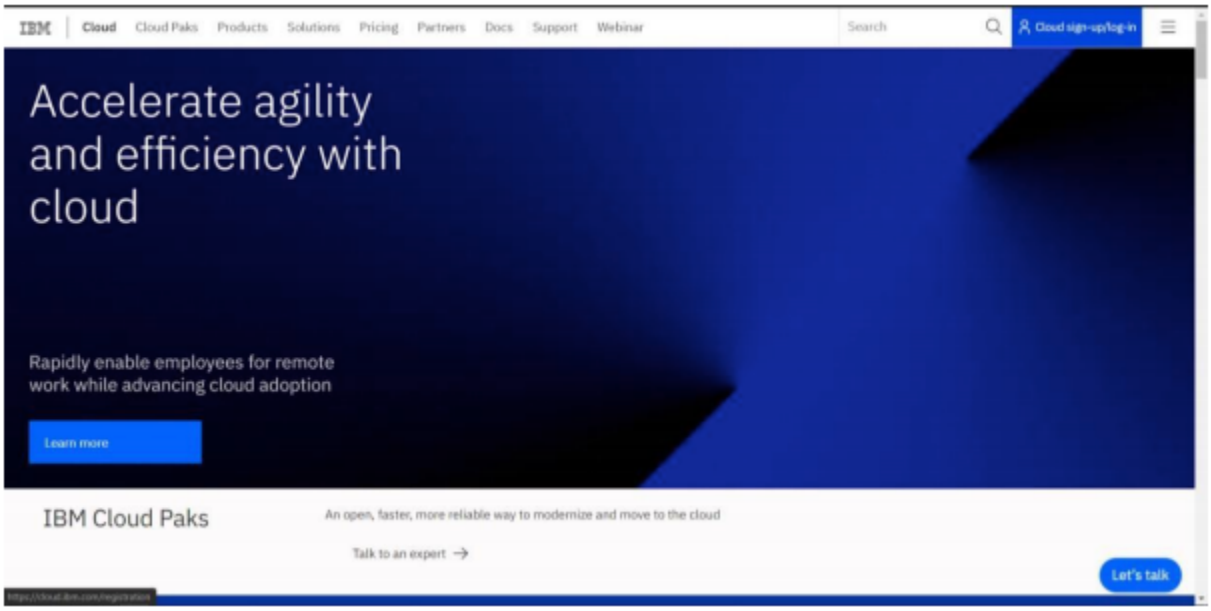
1. Create IBM Cloud services
2. Configure Watson Discovery
3. Create IBM Cloud Functions action
4. Configure Watson Assistant
5. Create flow and configure node

6. Deploy and run Node Red app.

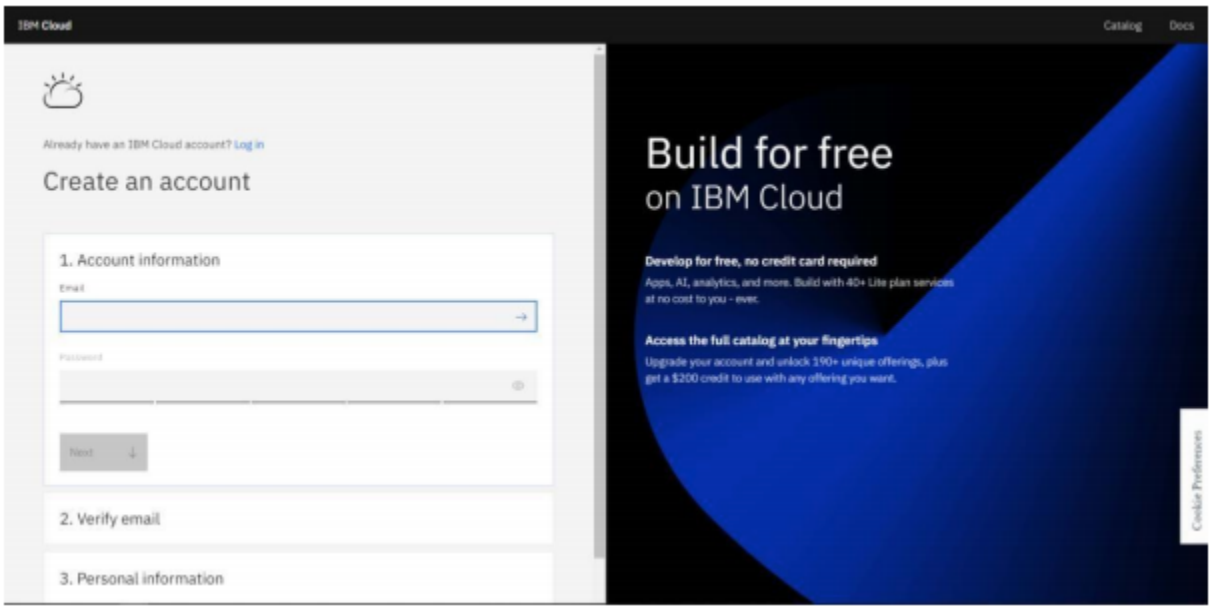
4. EXPERIMENTAL INVESTIGATIONS

1.Create IBM Cloud Services -

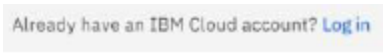
To Create IBM Cloud, go to https://www.ibm.com/cloud



Click on sign up or login. 

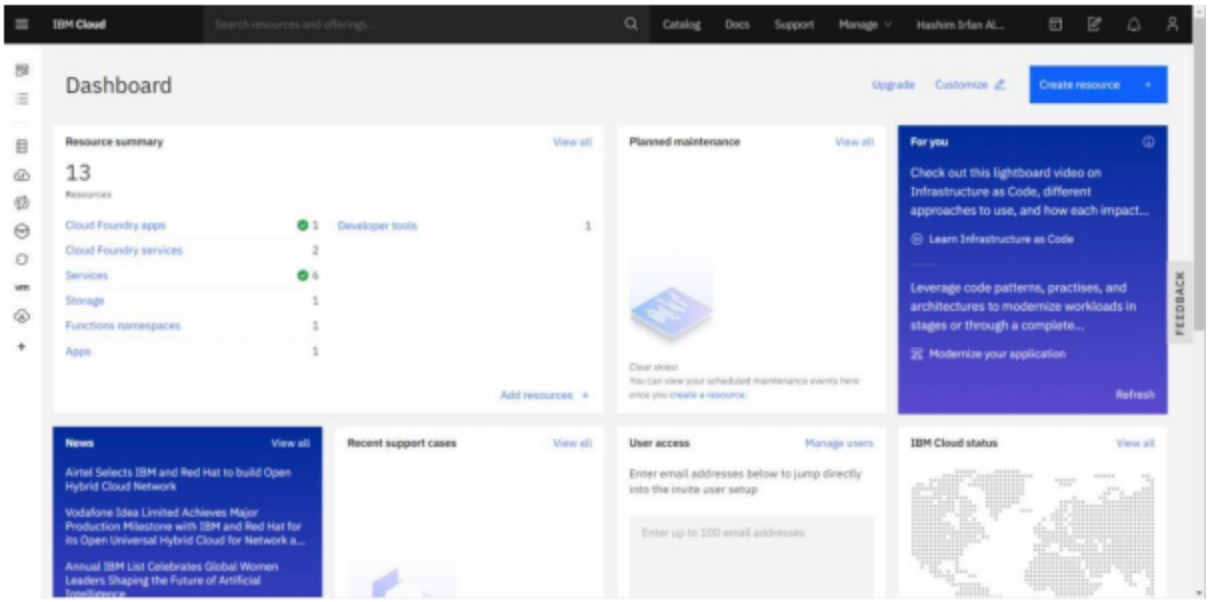


For Cloud Sign-Up: Follow the steps on the screen and fill in all the required details to create a new cloud account.

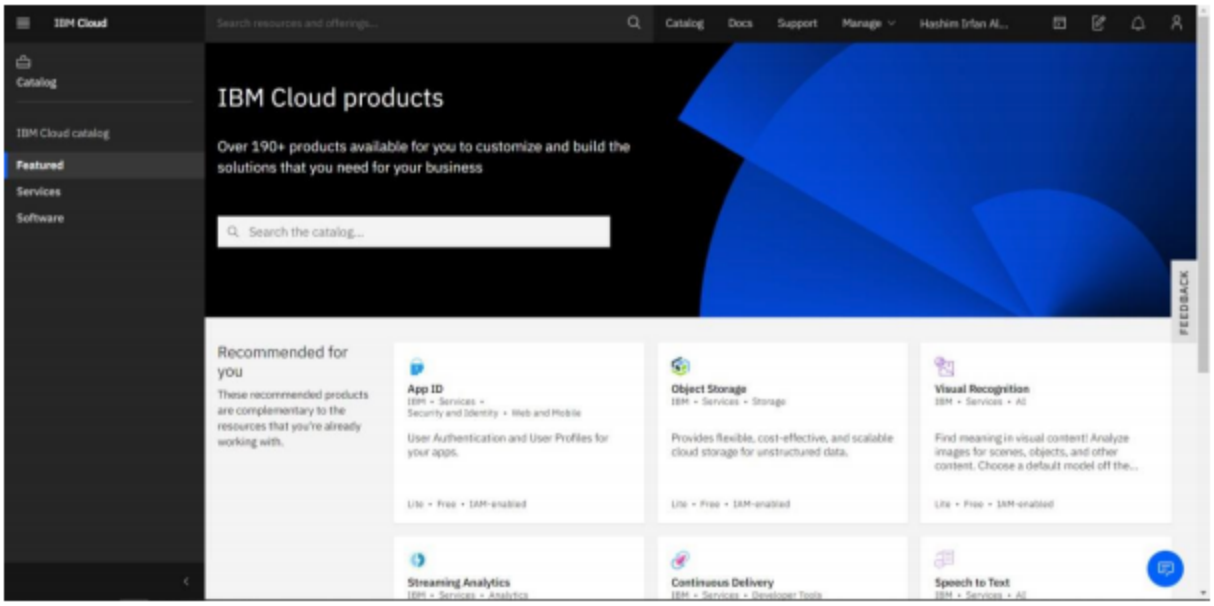
For Cloud Log In: Click on 

and fill your credentials to Log in to your cloud account.

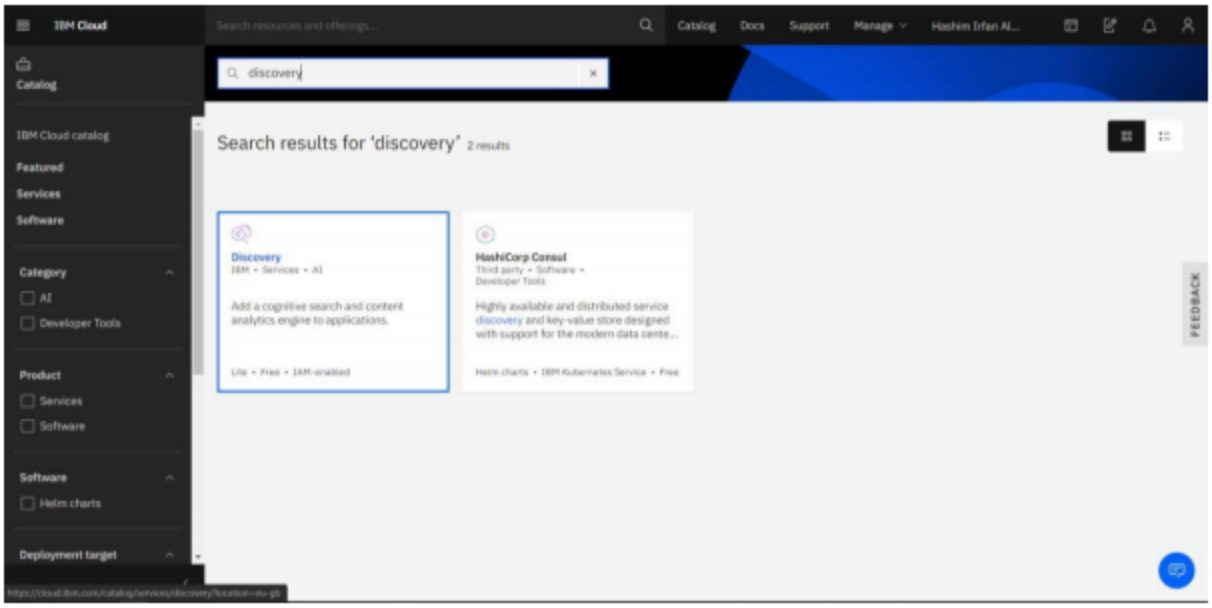
After Logging in, you can see the IBM Cloud Dashboard.

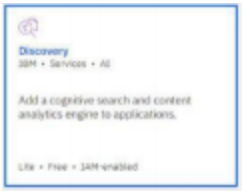


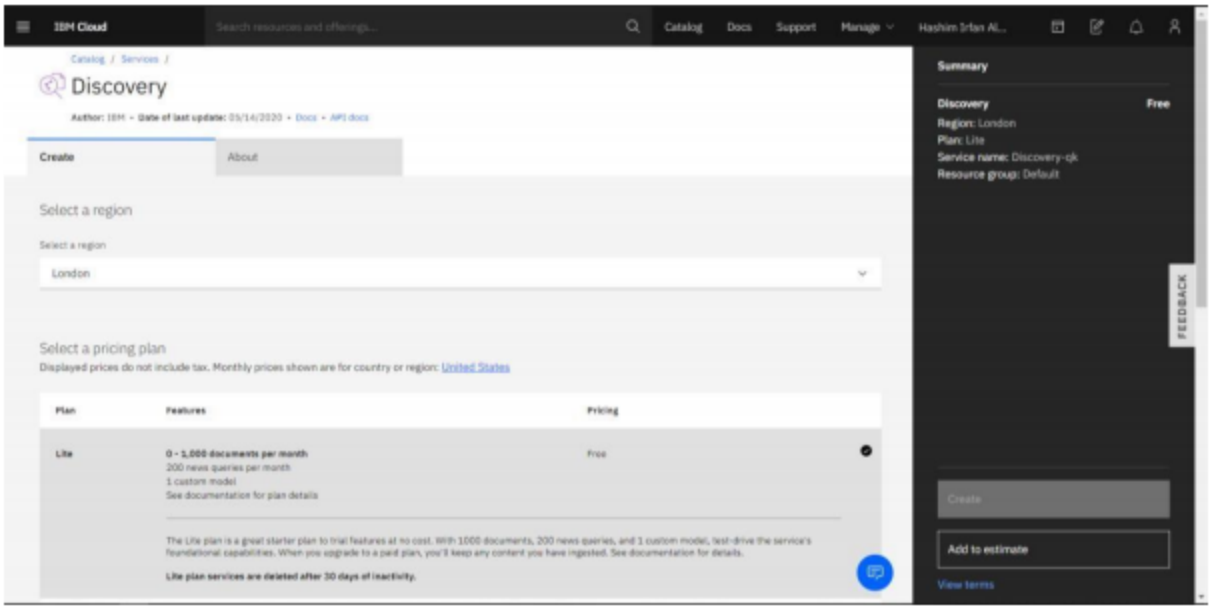
To Create any Resource (Services/Apps/etc), click on 



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.

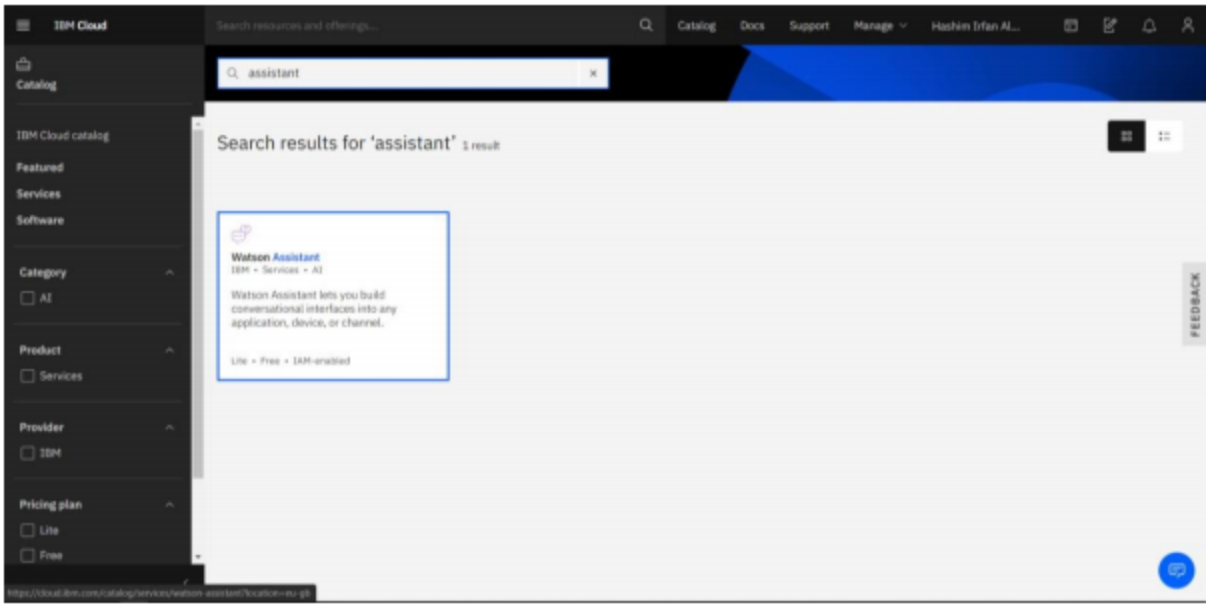


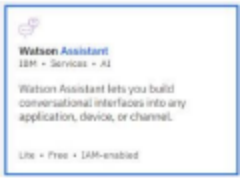
Click on 

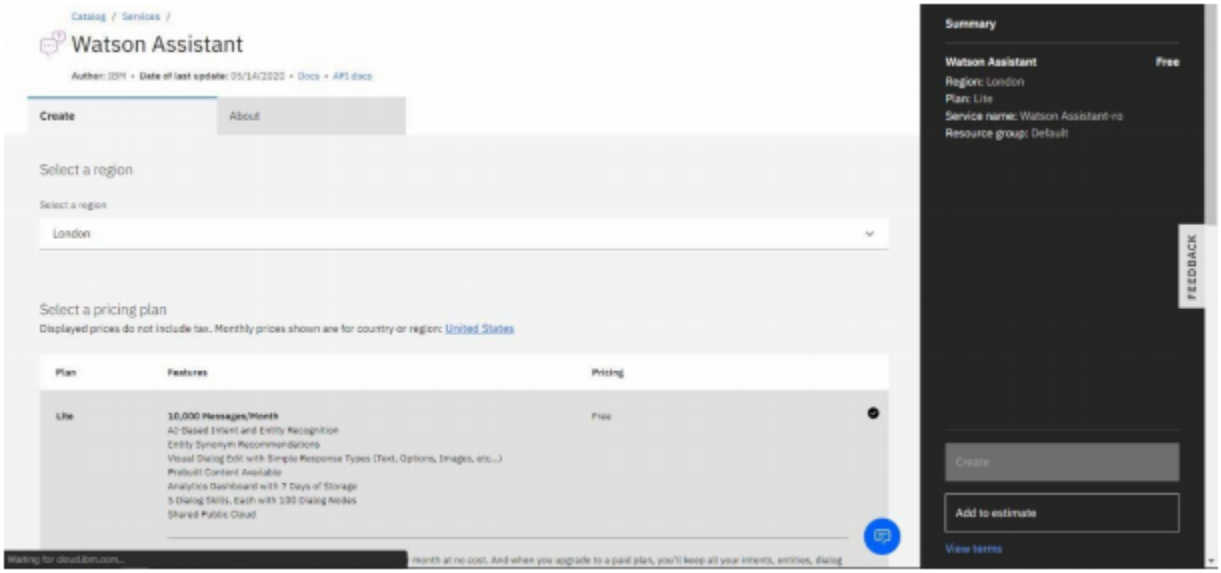


Select a region, select a plan, configure your service (Service name, etc) and click Create. Your Watson Discovery service is created successfully. (If you are on Lite Plan, you can have only one instance per service).

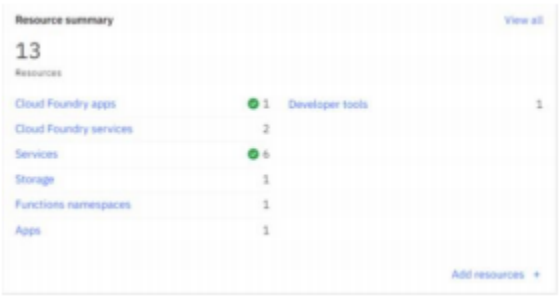
2.To create a Watson Assistant Service, search for Assistant in the search box



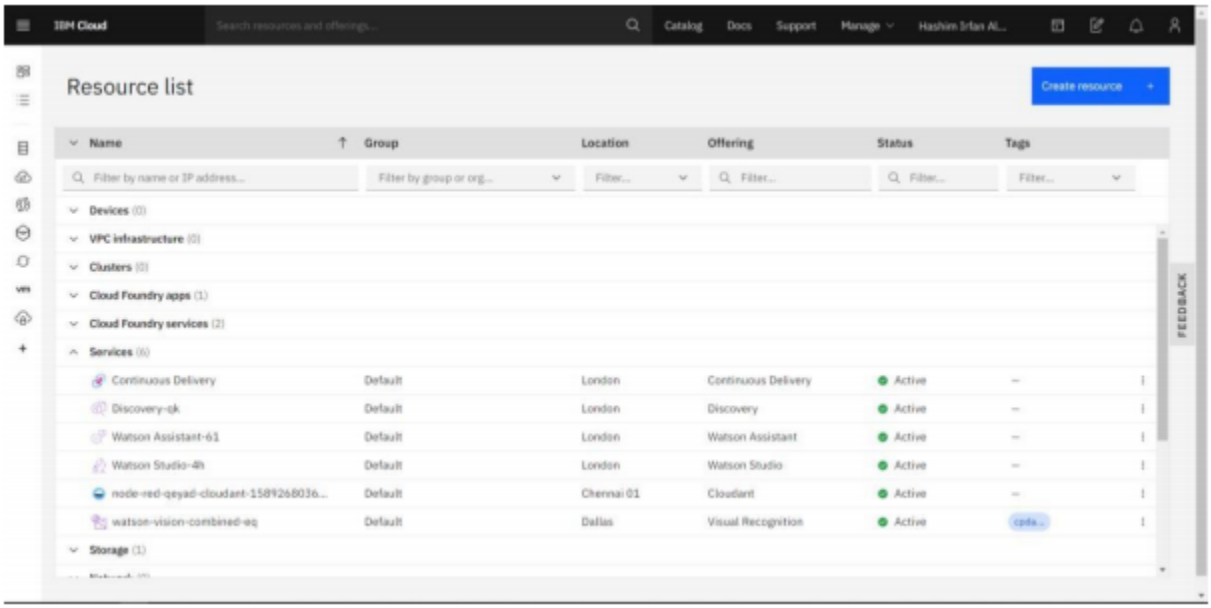
click on 



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.

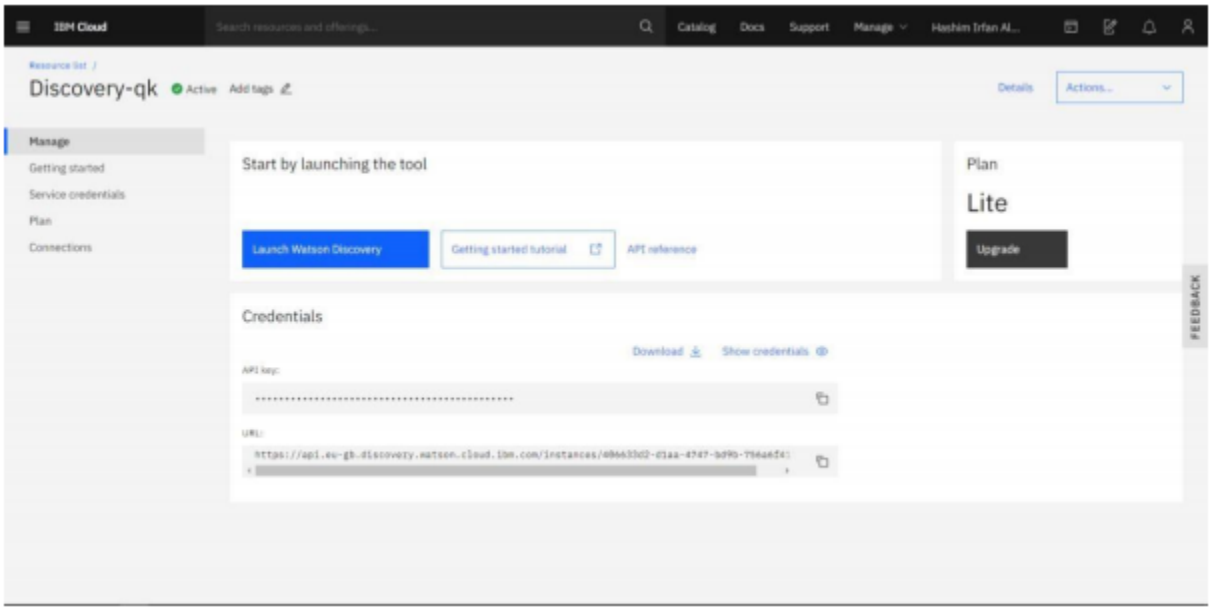


All of your existing Resource list will be shown here, click on Services to unveil the list of services you have.

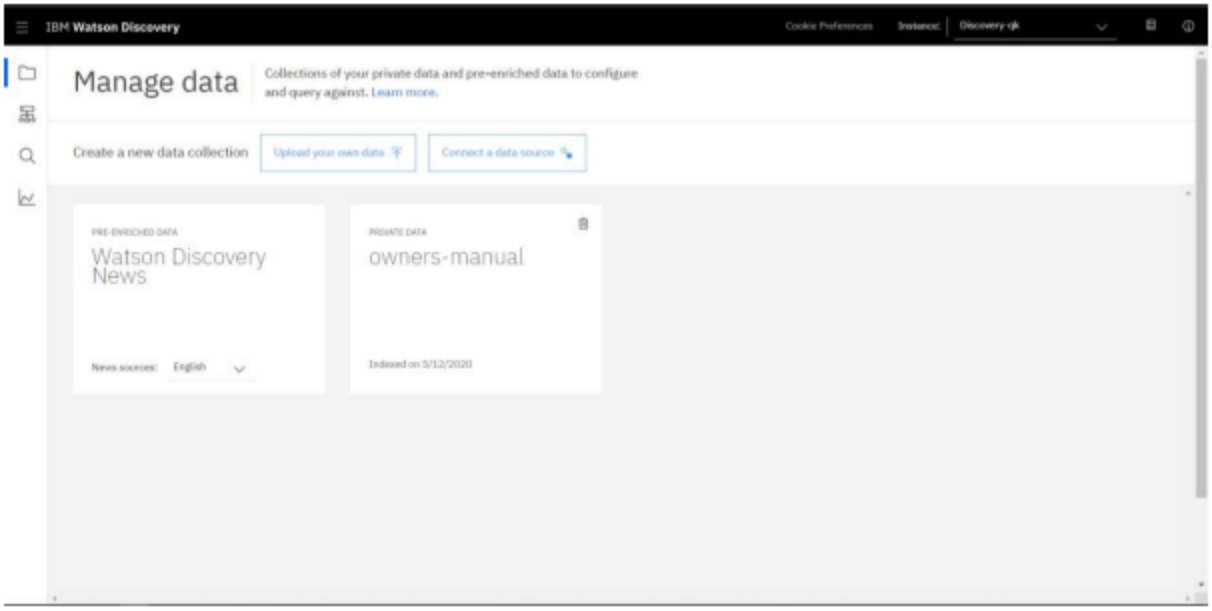


Here we can find that the status of Watson Discovery and Watson Assistant as Active which means we have configured the services correctly.

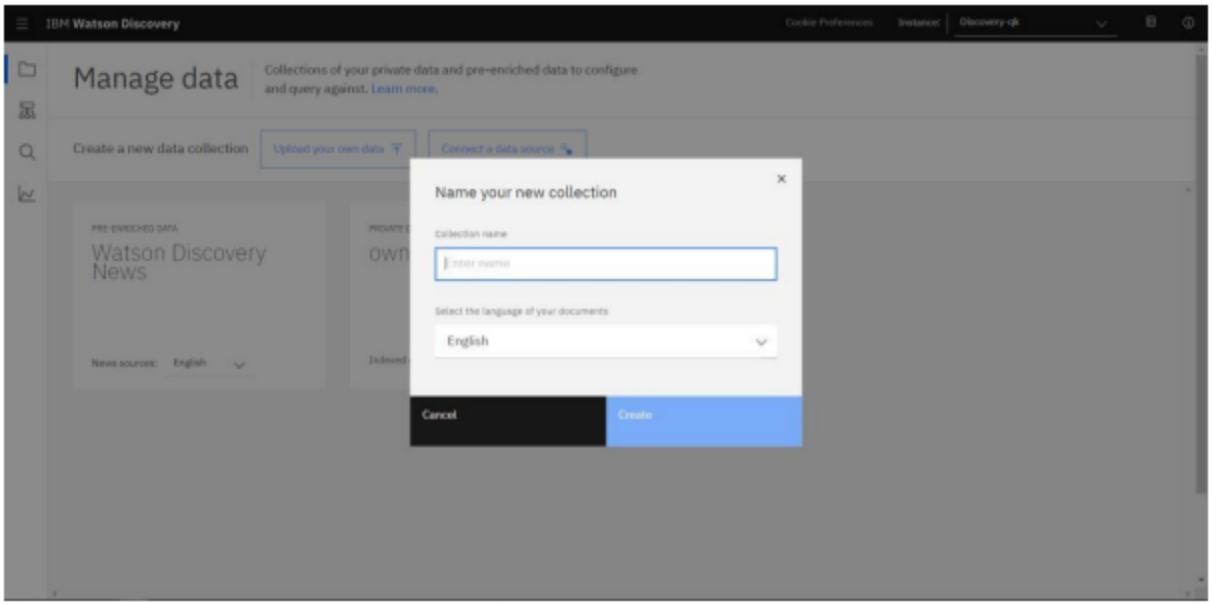
3. Configure Watson Discovery from the resource list screen, click to open Watson Discovery service



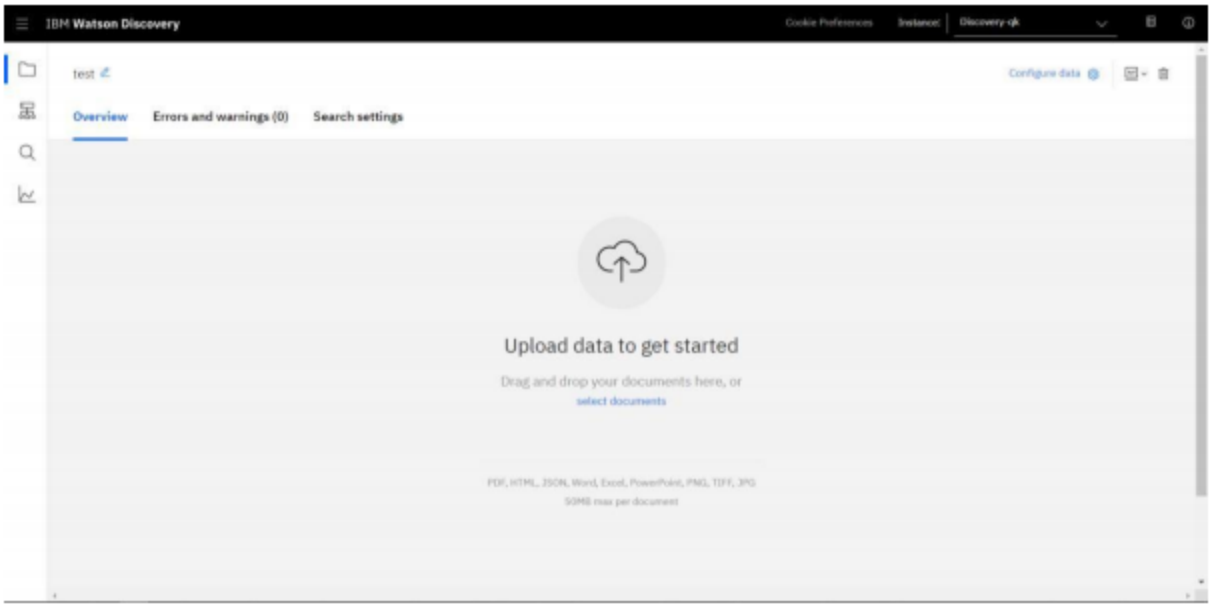
Click on Launch Watson Assistant to launch Watson Discovery Service.

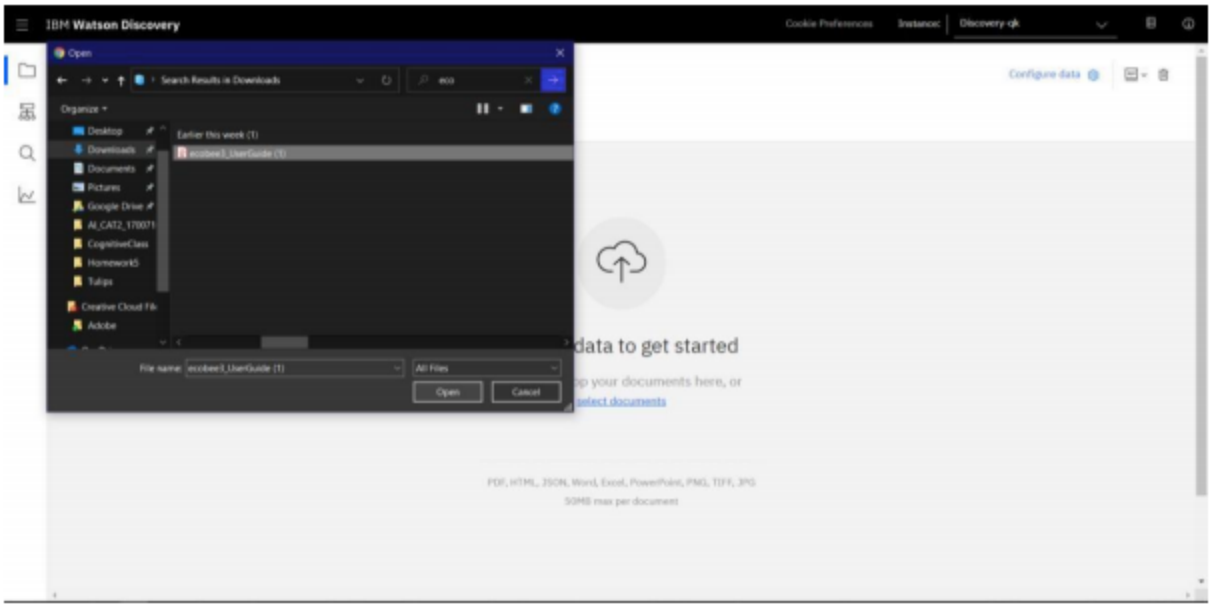


Click on  to create a new data collection.

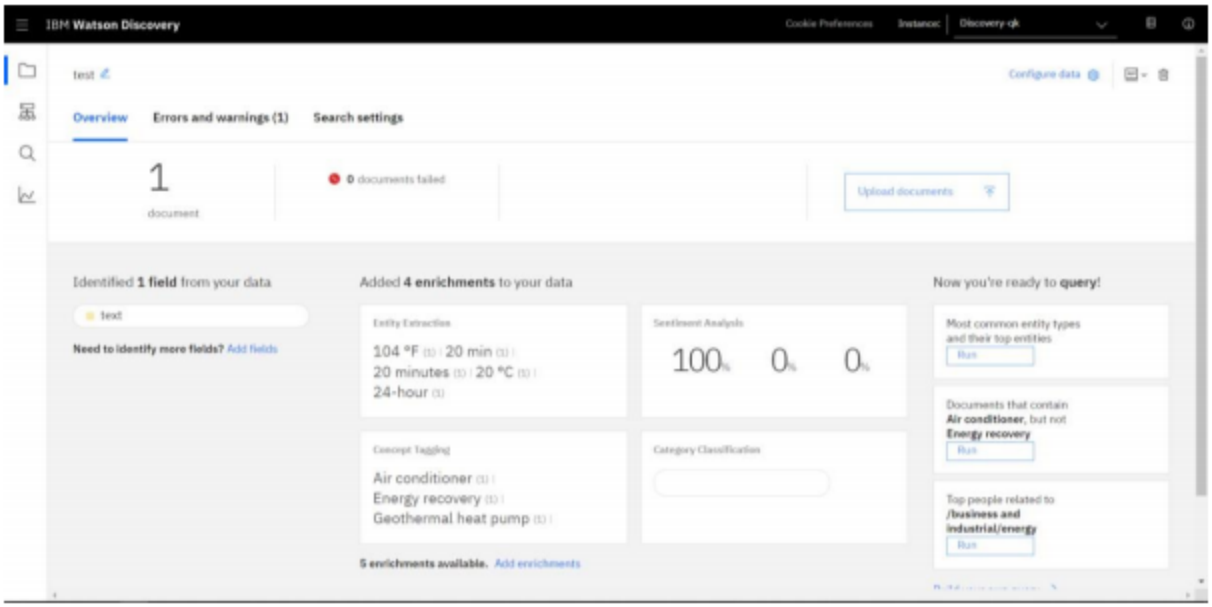


Give the data collection a unique name.





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

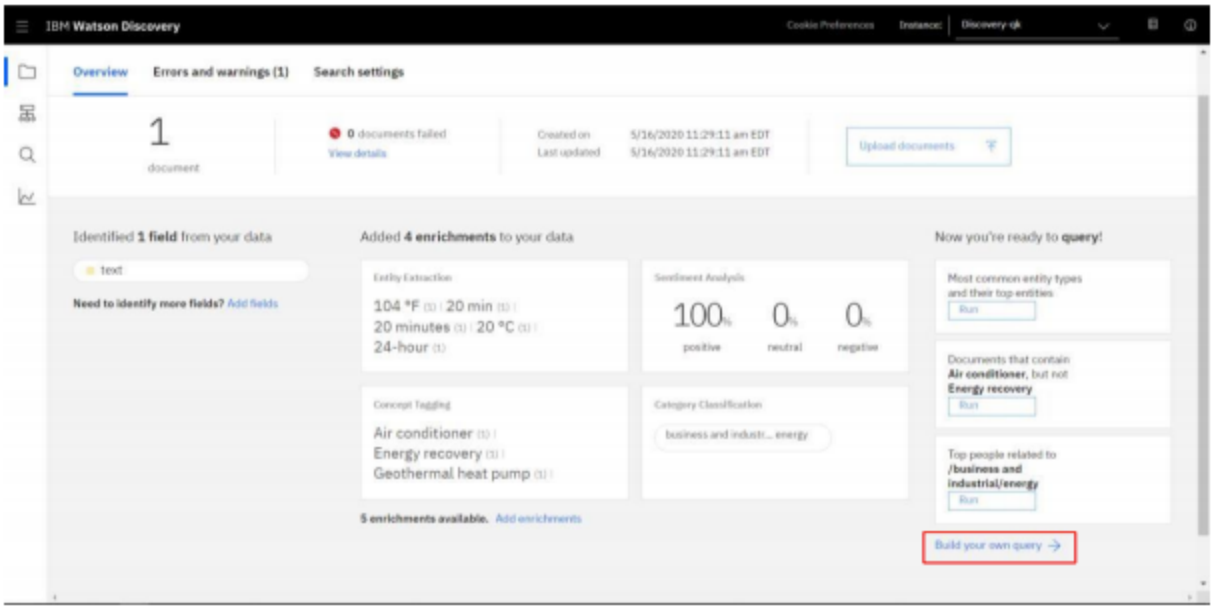


Before proceeding further, let's learn about Smart Document Understanding(SDU). SDU trains Watson Discovery to extract custom fields in your documents. Customizing how your documents are indexed into Discovery will improve the answers returned from queries.With SDU, you annotate fields within your documents 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:

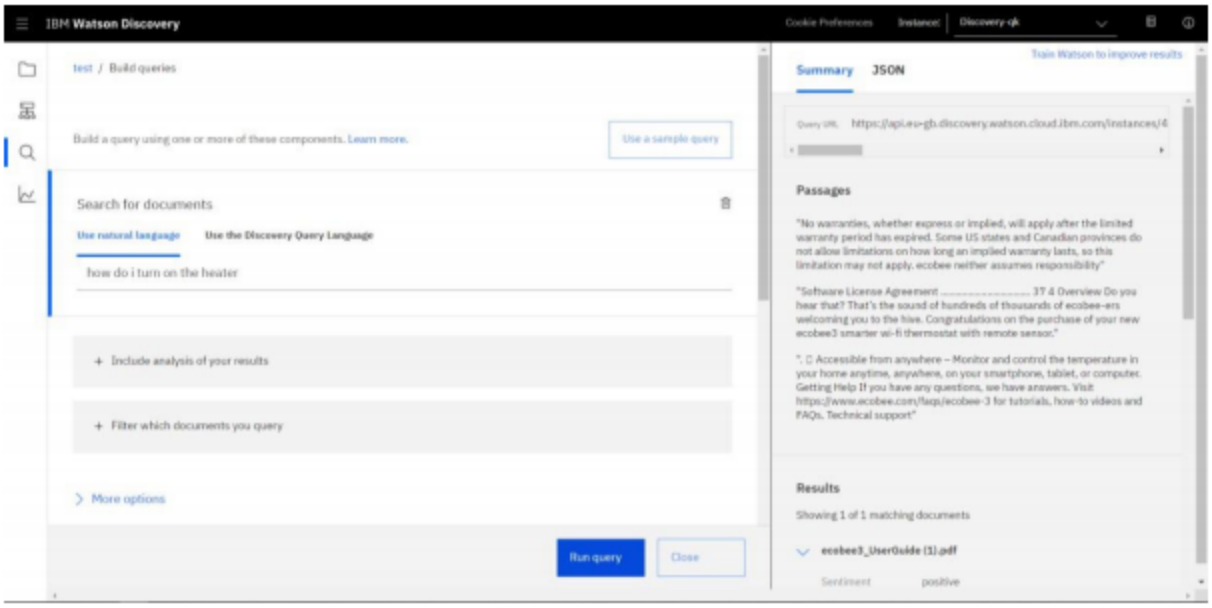
• Lite plans: PDF, Word, PowerPoint, Excel, JSON, HTML

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

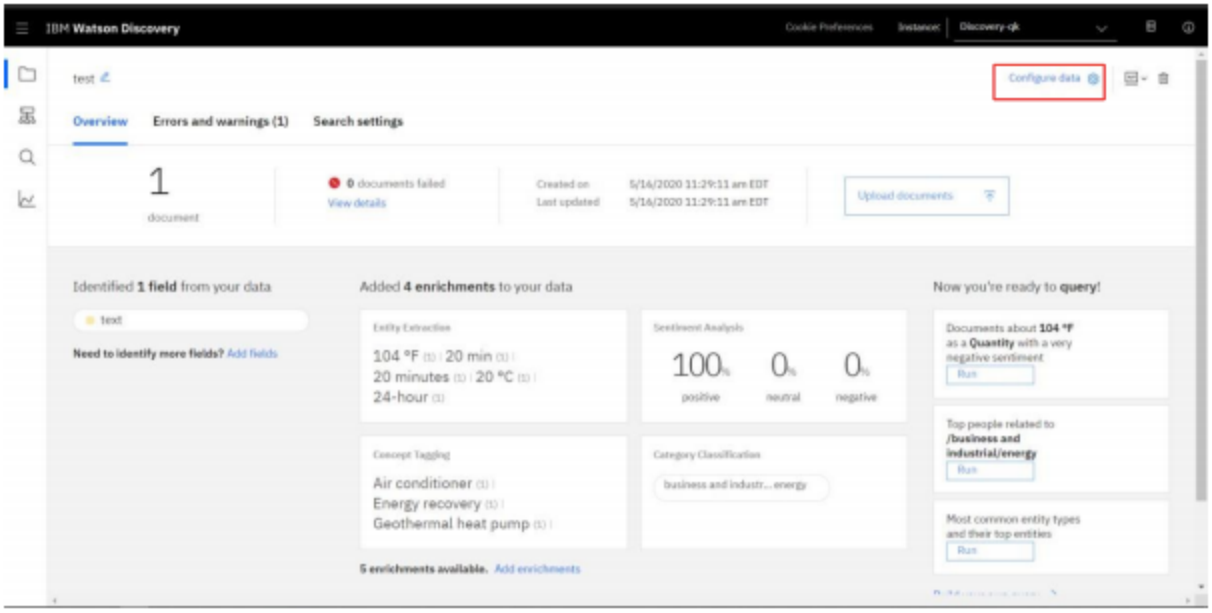
Before applying SDU to our document, let's do some simple queries on the data so that we can compare it to results found after applying SDU.



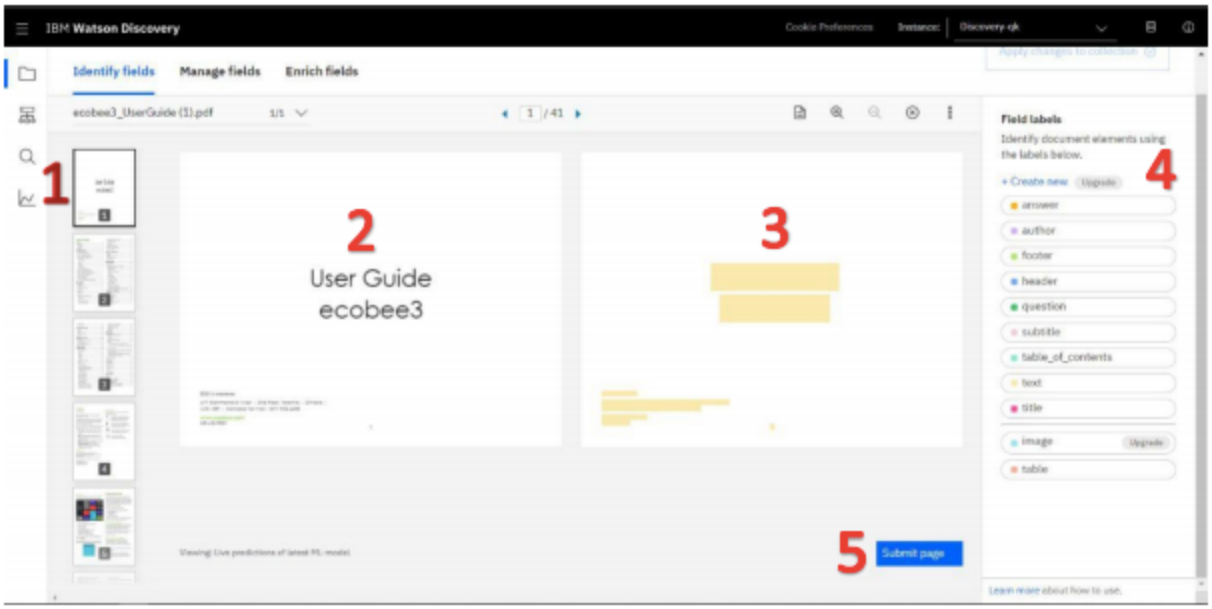
Click on Build your own query. Now, enter queries related to the operation of the thermostat and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question.



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).

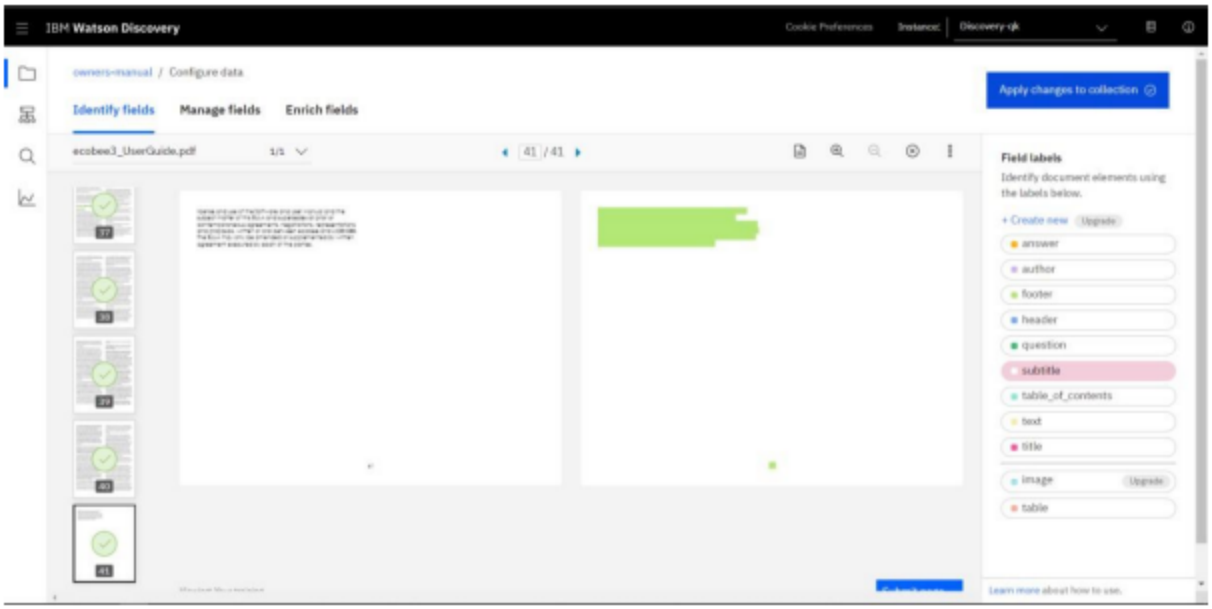


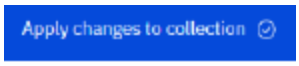
Click the Configure data button (located in the top right corner) to start the SDU process. Here is the layout of the Identify fields tab of the SDU annotation panel.



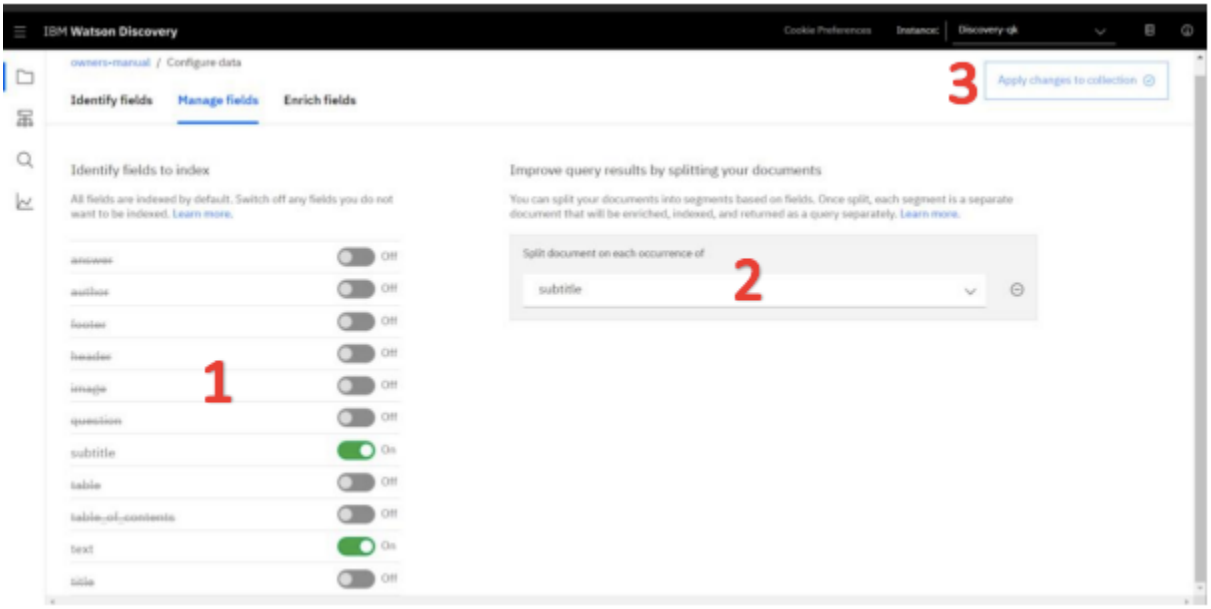
The goal is to annotate all of the pages in the document so Discovery can learn what text is important, and what text can be ignored. [1] is the list of pages in the manual. As each is processed, a green check mark will appear on the page. [2] is the current page being annotated. [3] is where you select text and assign it a label. [4] is the list of labels you can assign to the page textClick [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.

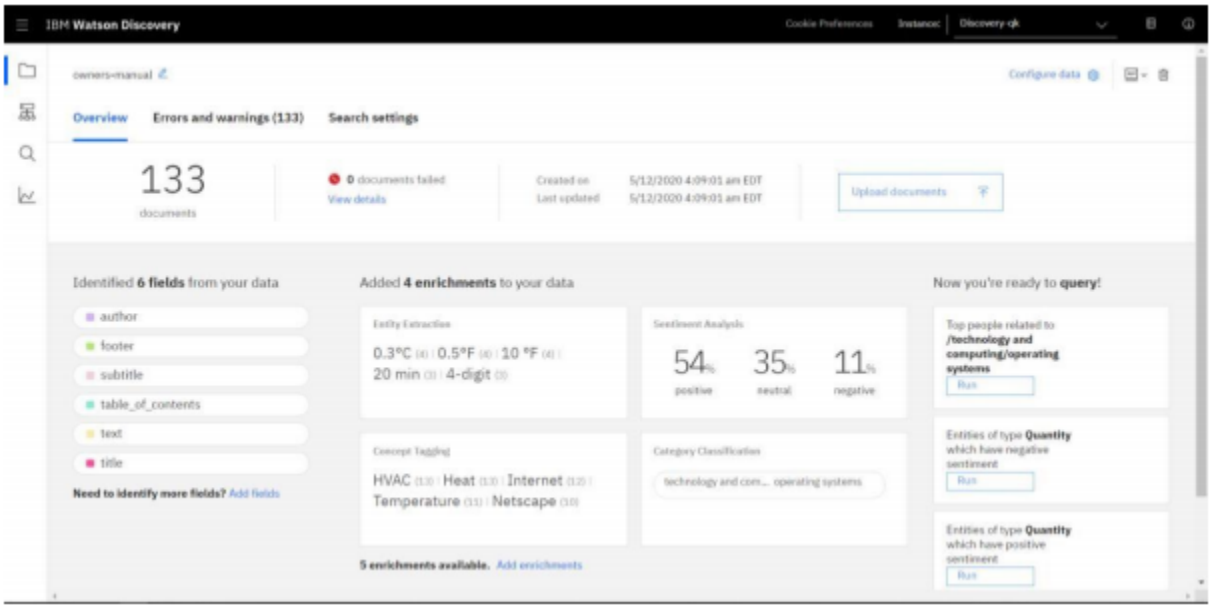


Click the 

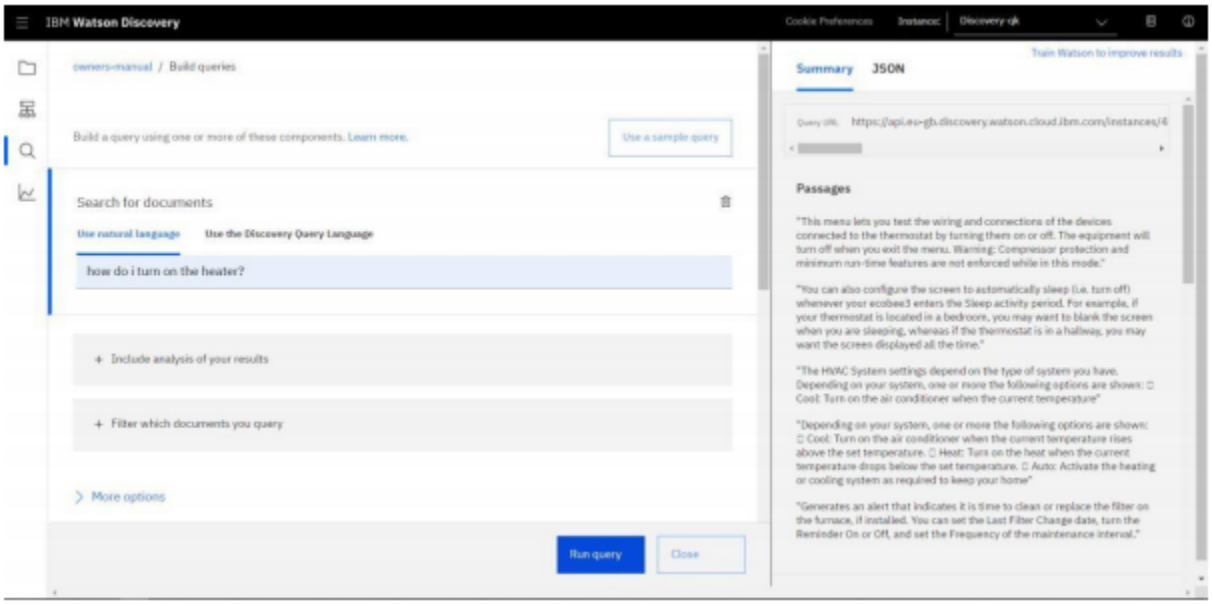
You will be asked to reload the document. Choose the same ecobee3_UserGuide .pdf document as before. Now, click on Manage fields tab on the Configure data panel.



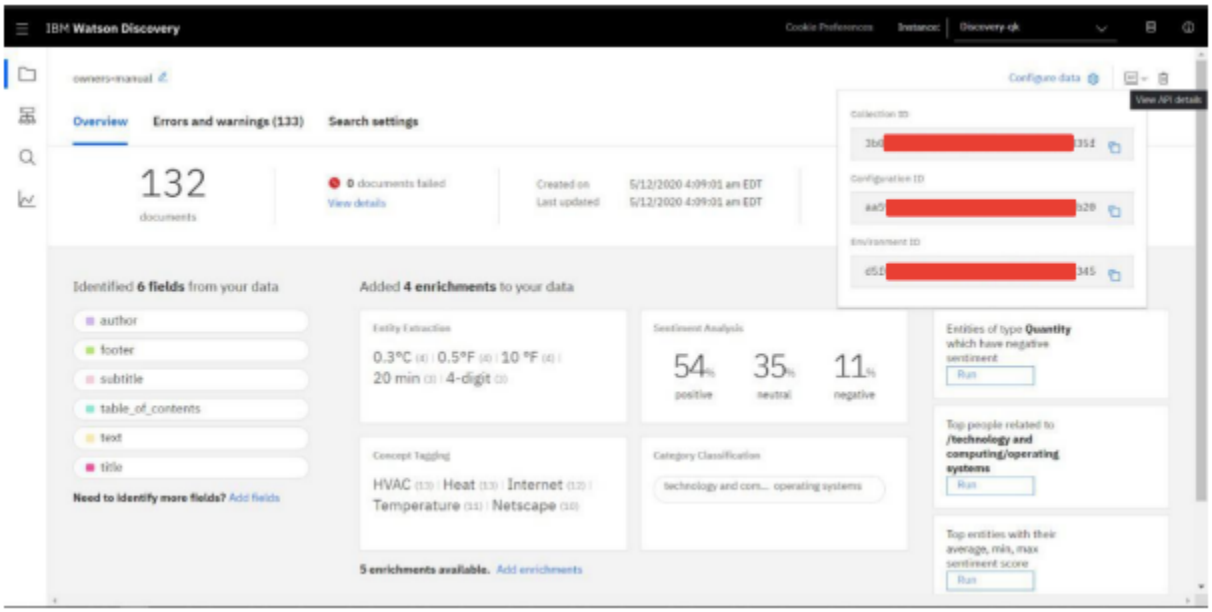
[1] Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except subtitles and text. [2] is telling Discovery to split the document apart, based on subtitle. Click [3] to submit your changes. Once again, you will be asked to reload the document. Choose the same ecobee3_UserGuide .pdf document as before. Now, as a result of splitting the document apart, your collection will look different.



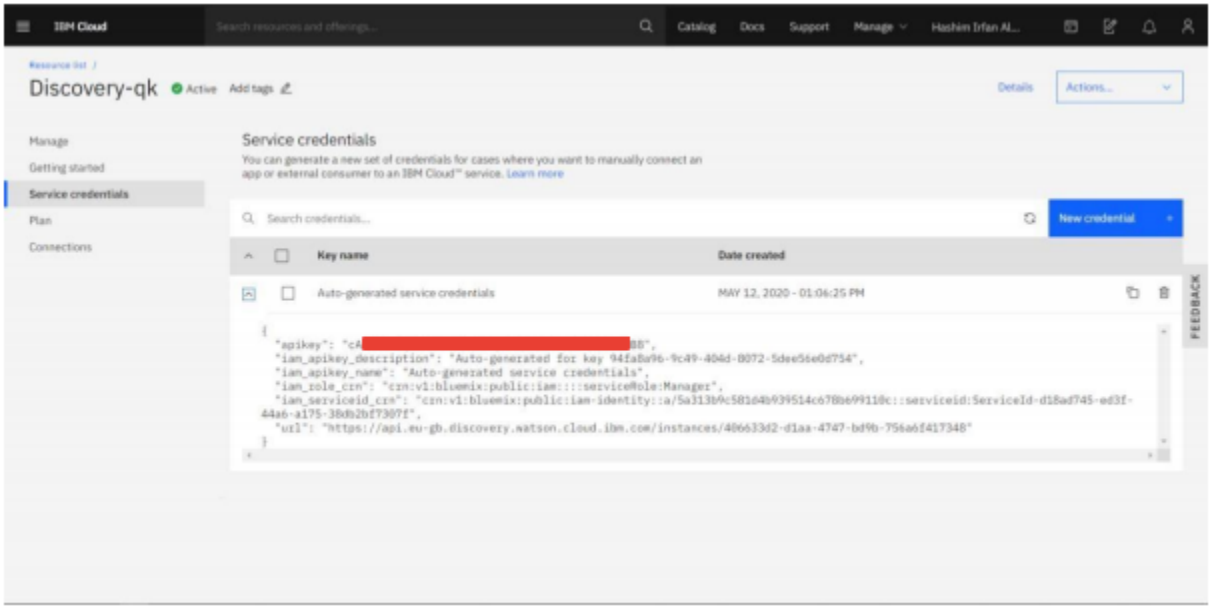
Now click the Build your own query and see how much better the results are.



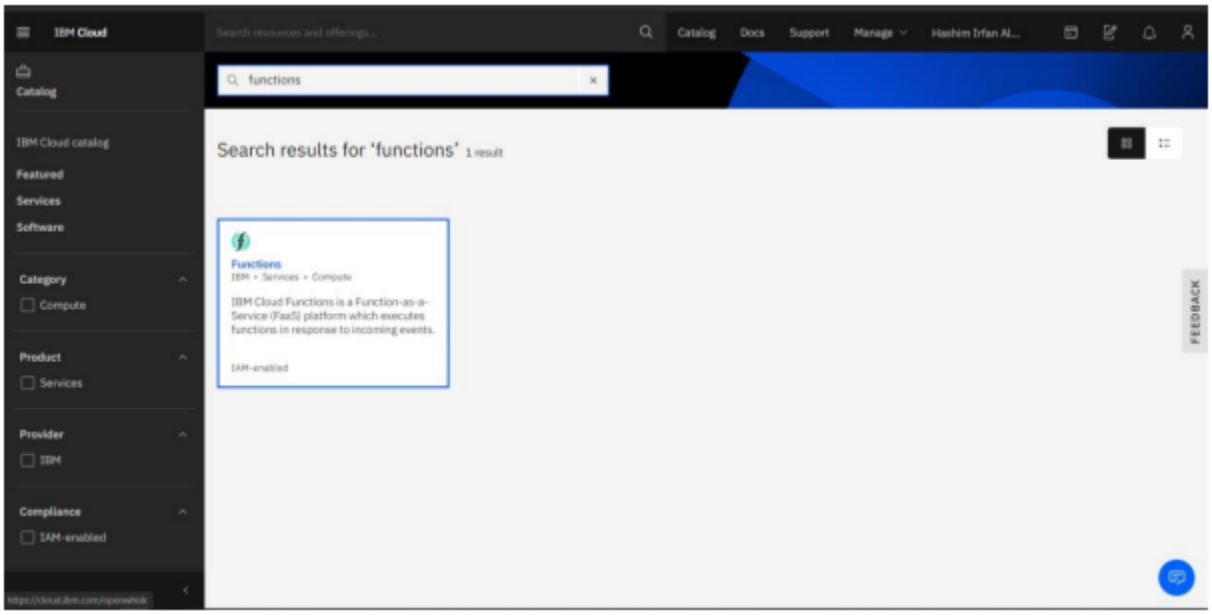
In upcoming steps, you will need to provide some credentials to access your Discovery collection so to Store credentials for future use follow the below steps. The Collection ID and Environment ID values can be found by clicking the located at the top right side of your collection panel.



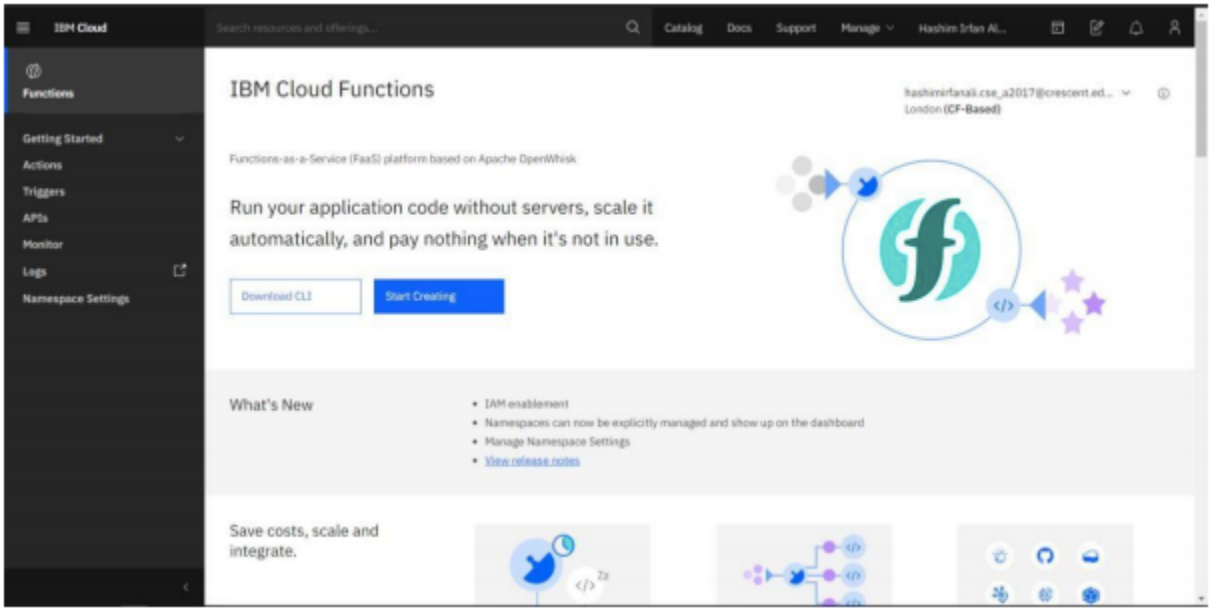
Now go to the Watson Discovery Resource List Screen. Here, select service credentials. The apikey and URL endpoint for your service can be found here.



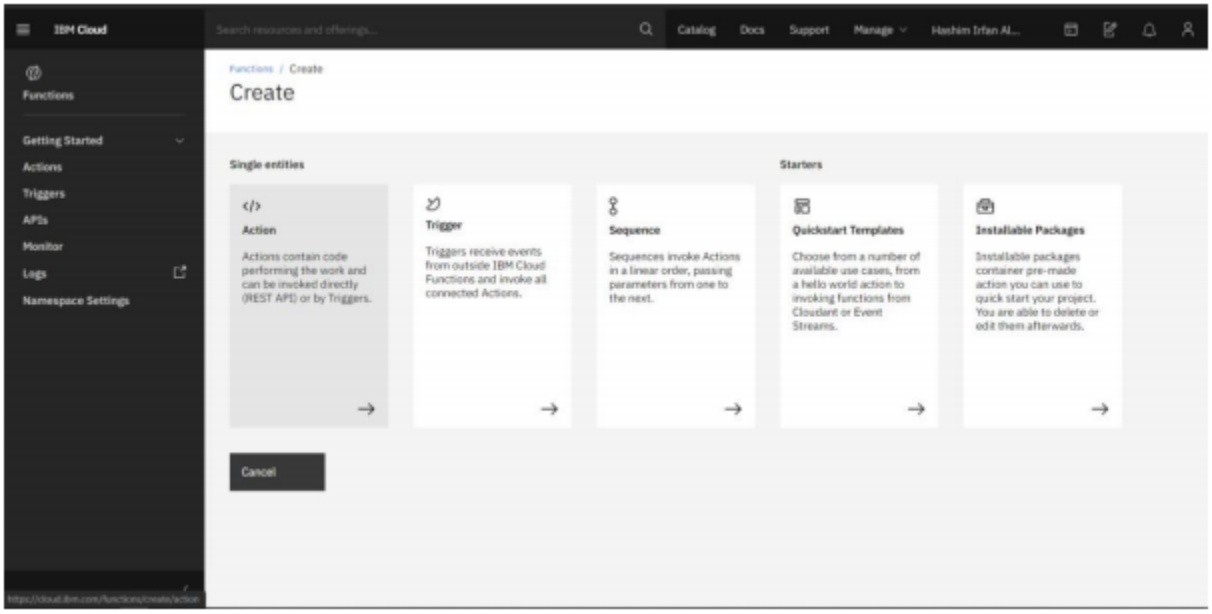
4. Create IBM Cloud Functions action - Now let's create the web action that will make queries against our Discovery collection. Go to IBM Cloud Dashboard, click on Create Resource and search for Functions



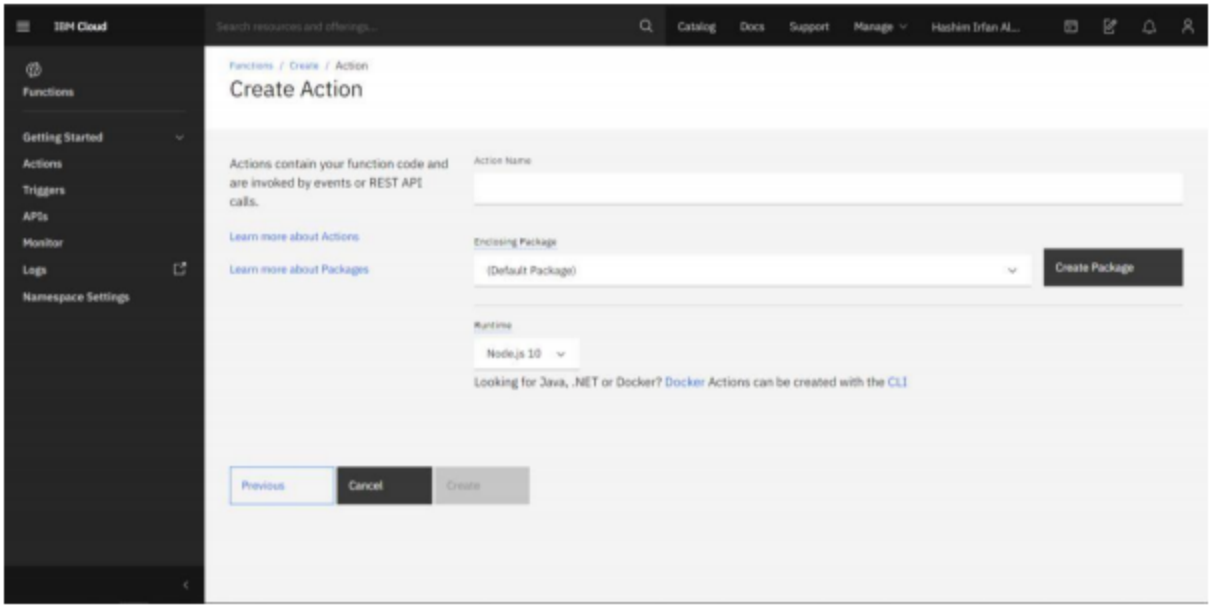
Select the Functions Card.



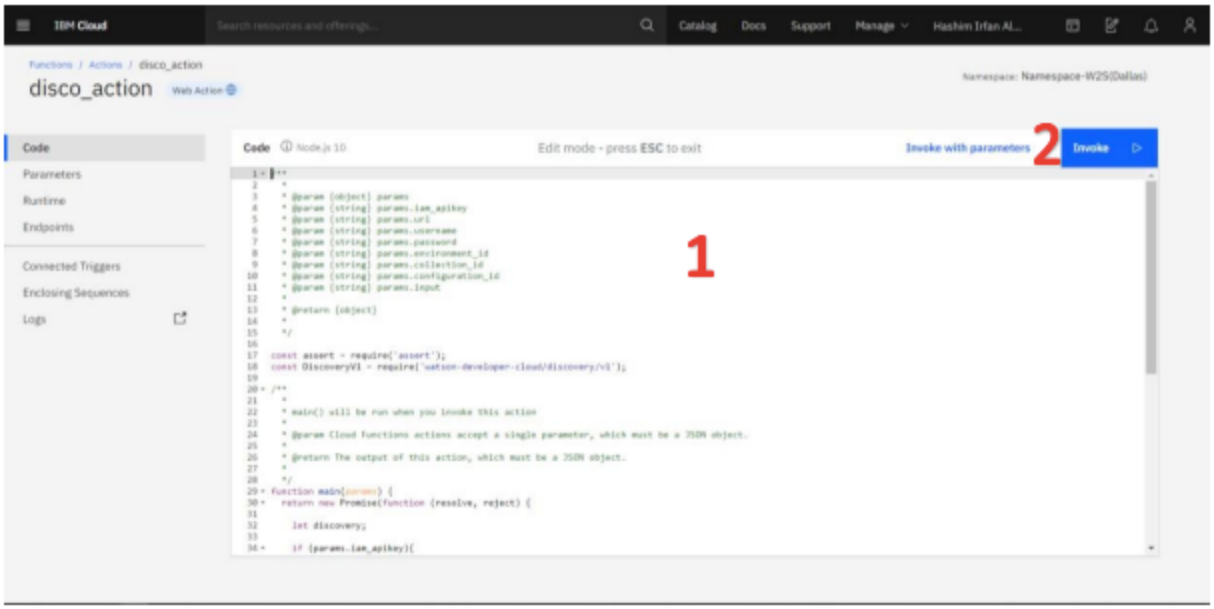
From the Functions main panel, click on Start Creating.



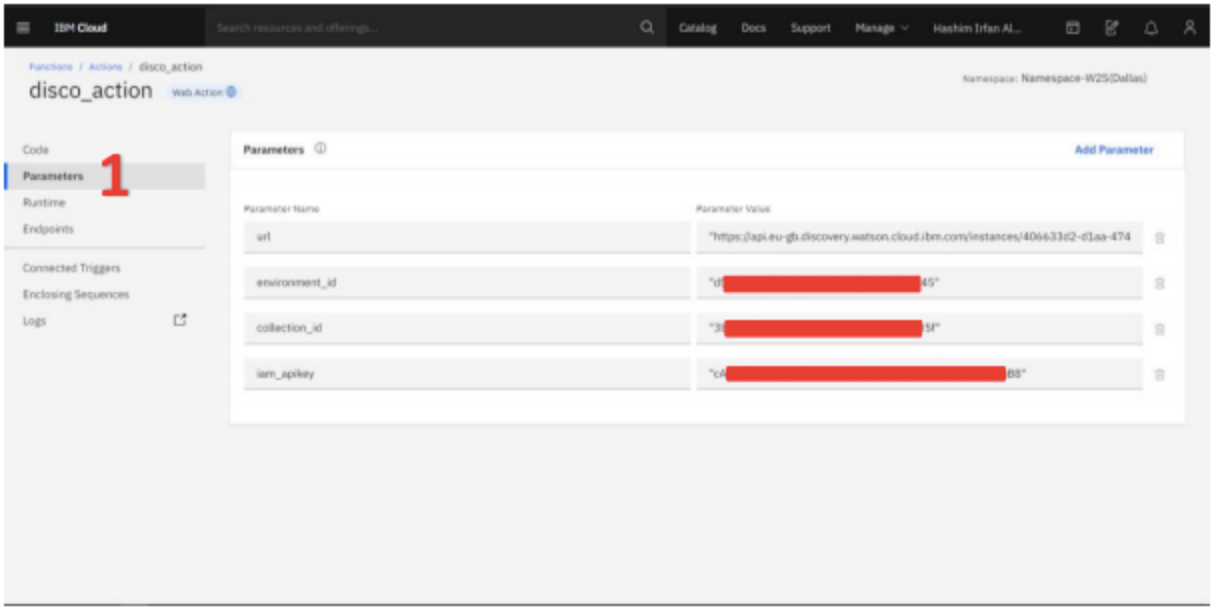
Here, select Actions.



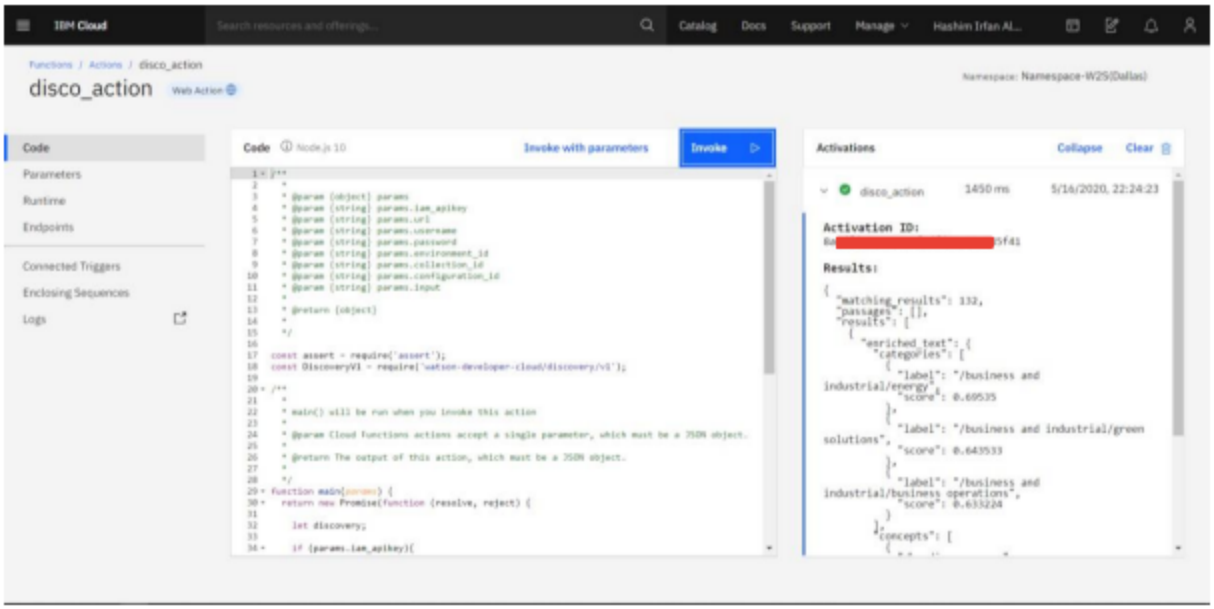
On the Create Action panel, provide a unique Action Name, keep the default package, and select the Node.js 10 runtime. Click the Create button to create the action. Once function is created, click on code tab.

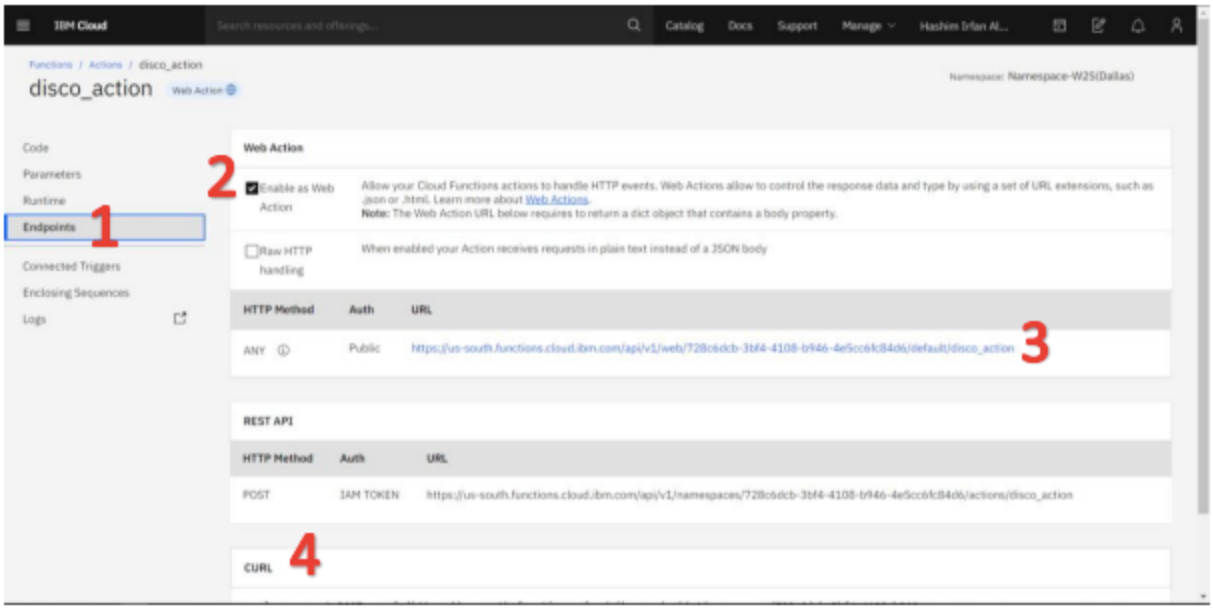


In the code editor window [1], cut and paste in the code from the disco-action.js file found in the actions directory of your local repository. The code is pretty straight-forward - it simply connects to the Discovery service, makes a query against the collection, then returns the response. Now if click the invoke [2] button, it will fail due to credentials not being defined yet.To solve this, select parameter[1] tab



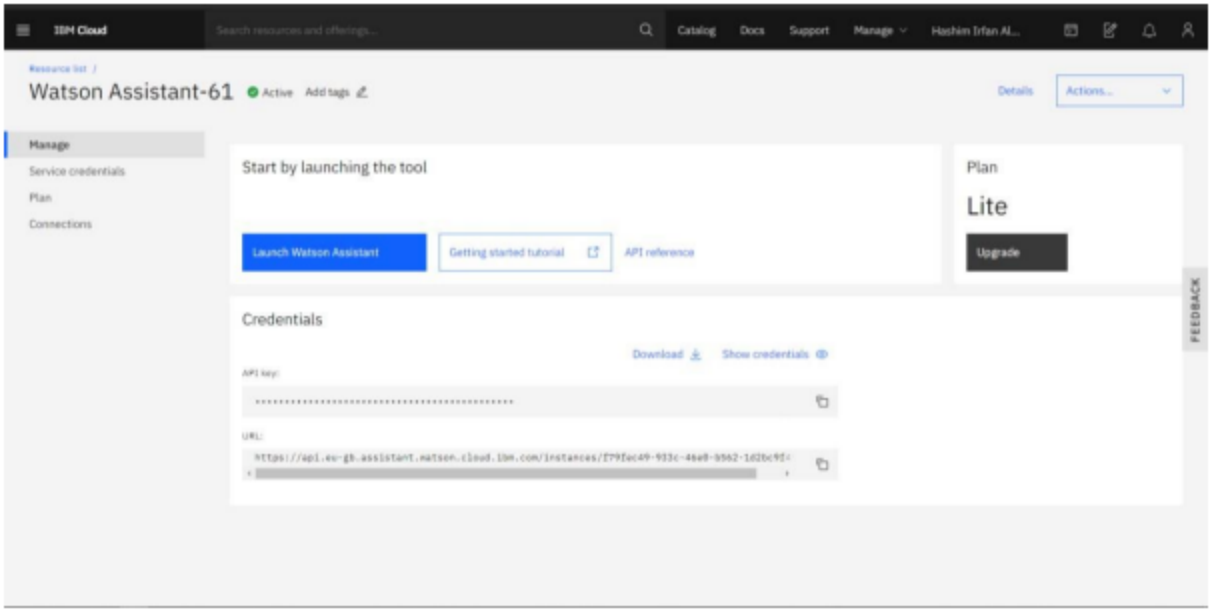
Add the following keys: • url • environment_id • collection_id • iam_apikey. For the above values, please use the values associated with the Discovery service you created in the previous step. Enclose your values in double quotes. Now that the credentials are set, return to the Code panel tab and press the Invoke button again. Now you should see actual results returned from the Discovery service.



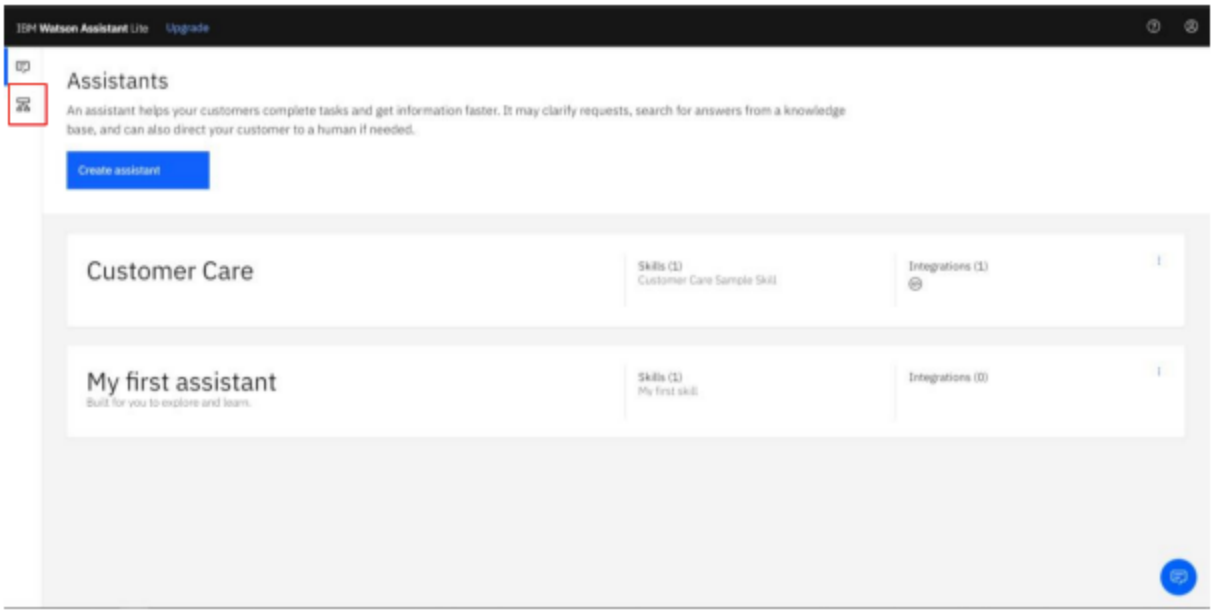


Now, click on endpoints[1] tab. Click the checkbox for Enable as Web Action [2]. This will generate a public endpoint URL [3]. Take note of the URL value [3], as this will be needed by Watson Assistant in a future step. To verify you have entered the correct Discovery parameters, execute the provided 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).]

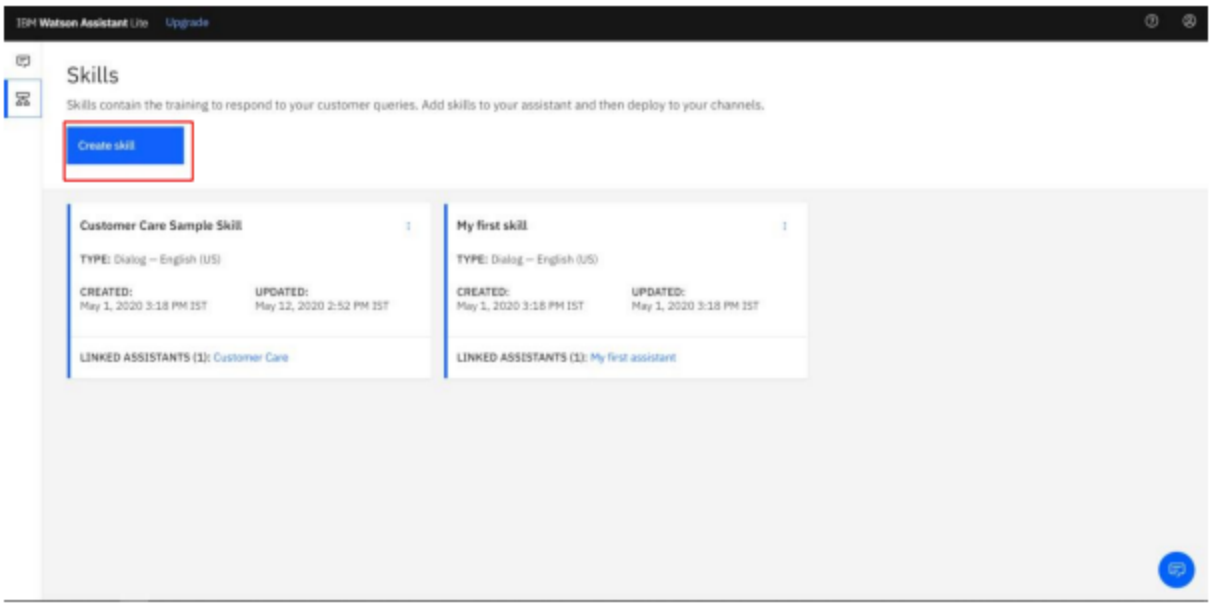
5. Configure Watson Assistant - Go back to the IBM Dashboard from the resource list screen. click to open Watson Assistant service.



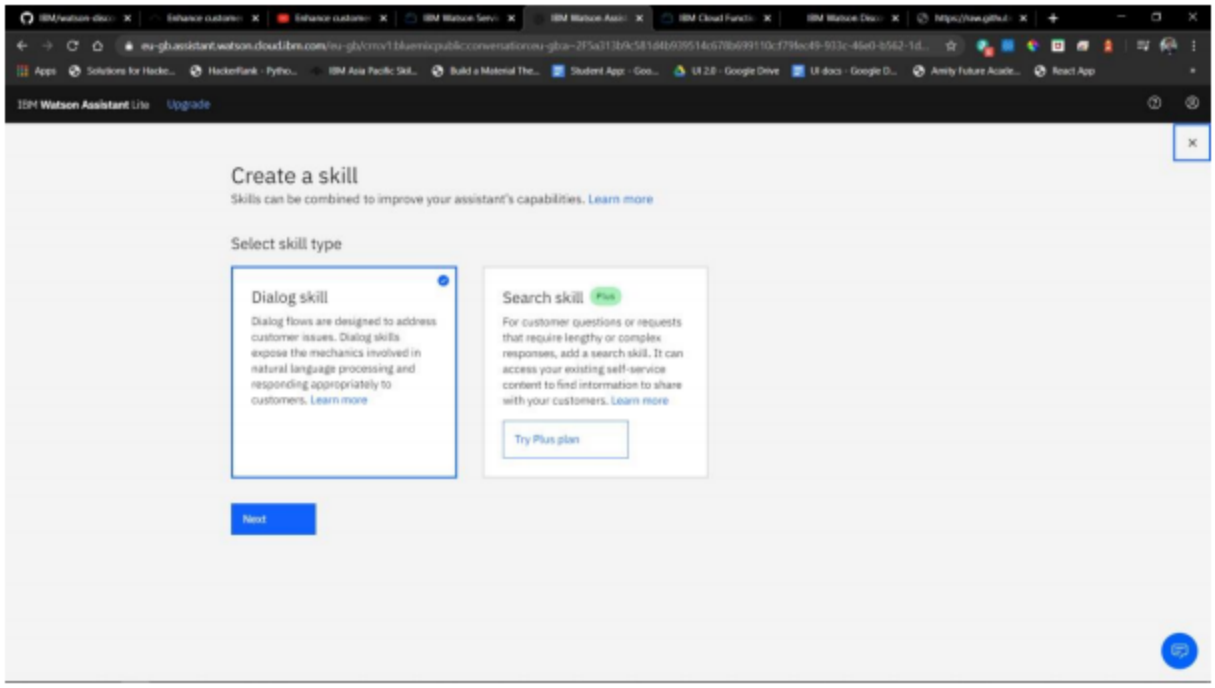
Click on Launch Watson Assistant



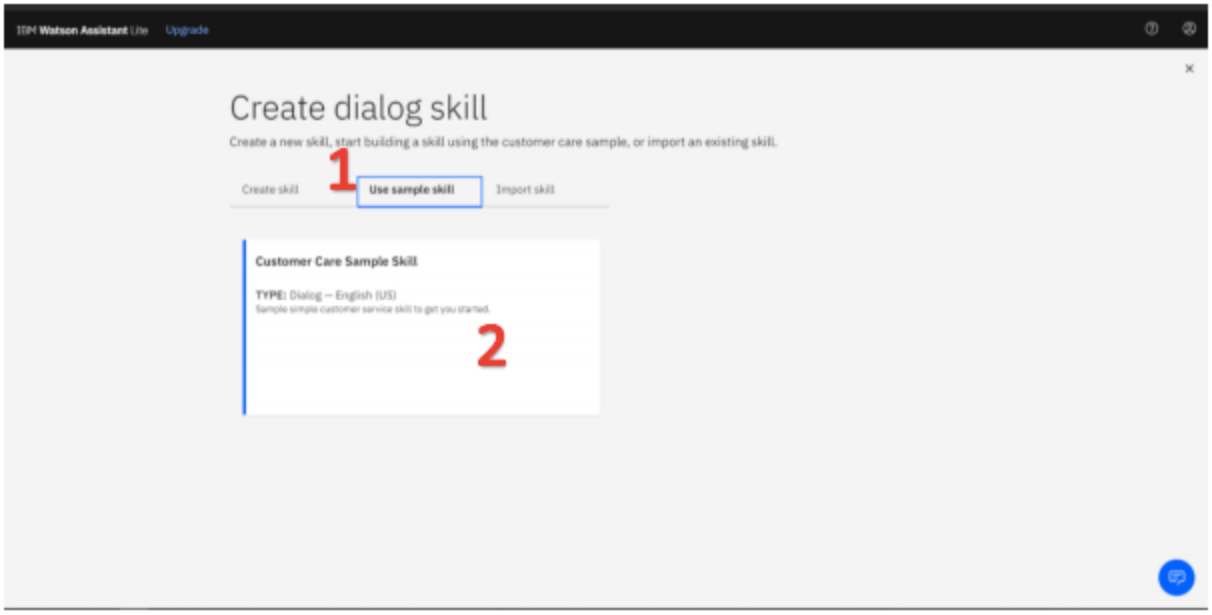
Click on the skills tab.



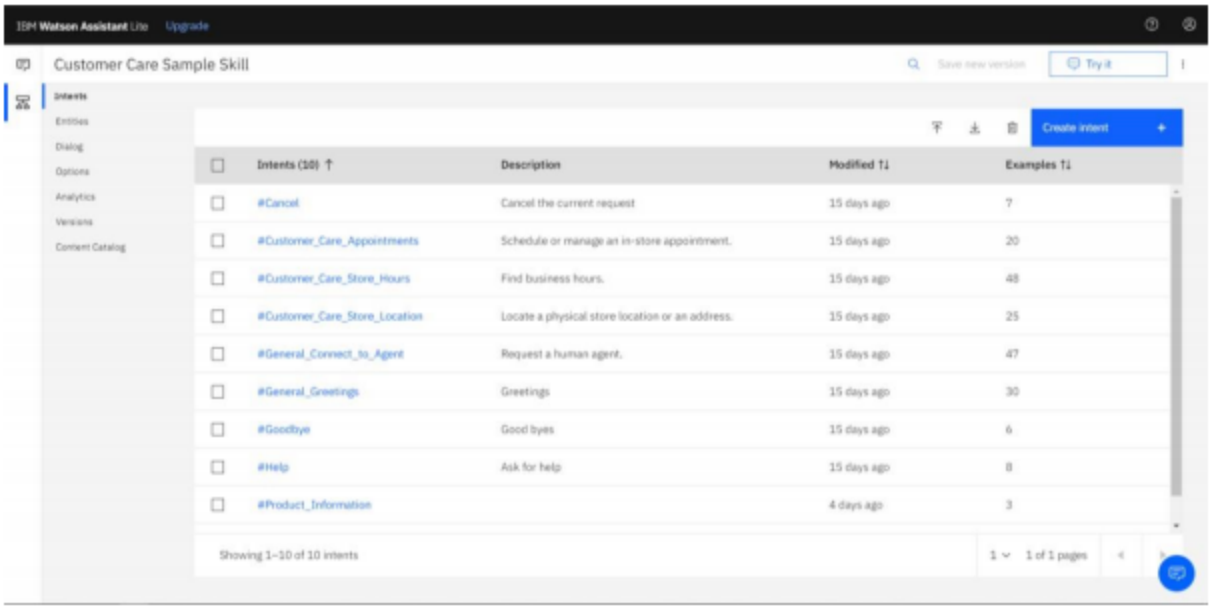
Click Create Skill



Select Dialog Skill Card and Click next.

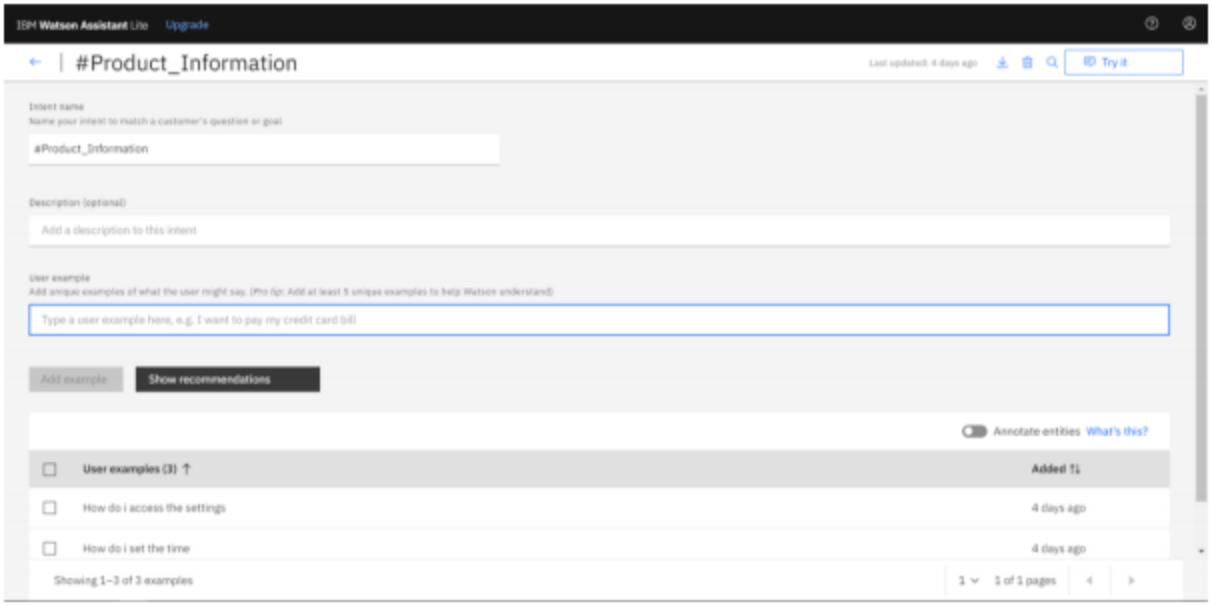


Select Use Sample Skill [1] and select Customer Care Sample Skill [2]. This dialog skill contains all of the nodes needed to have a typical call centre conversation with a user.

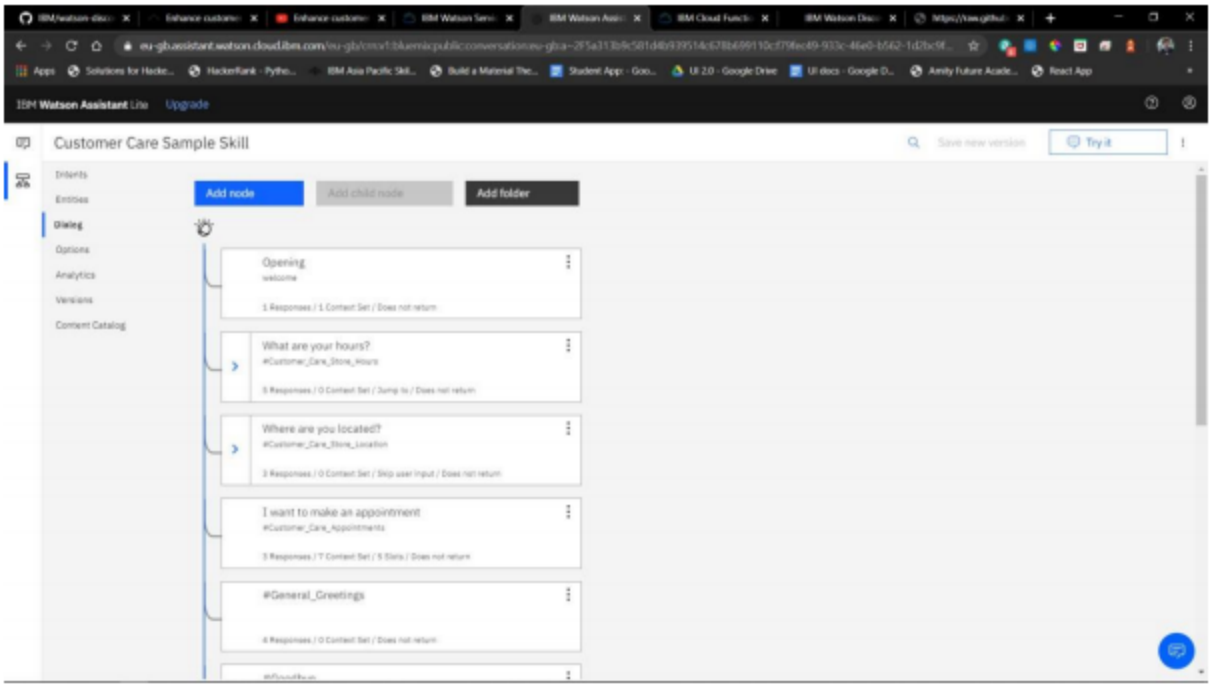


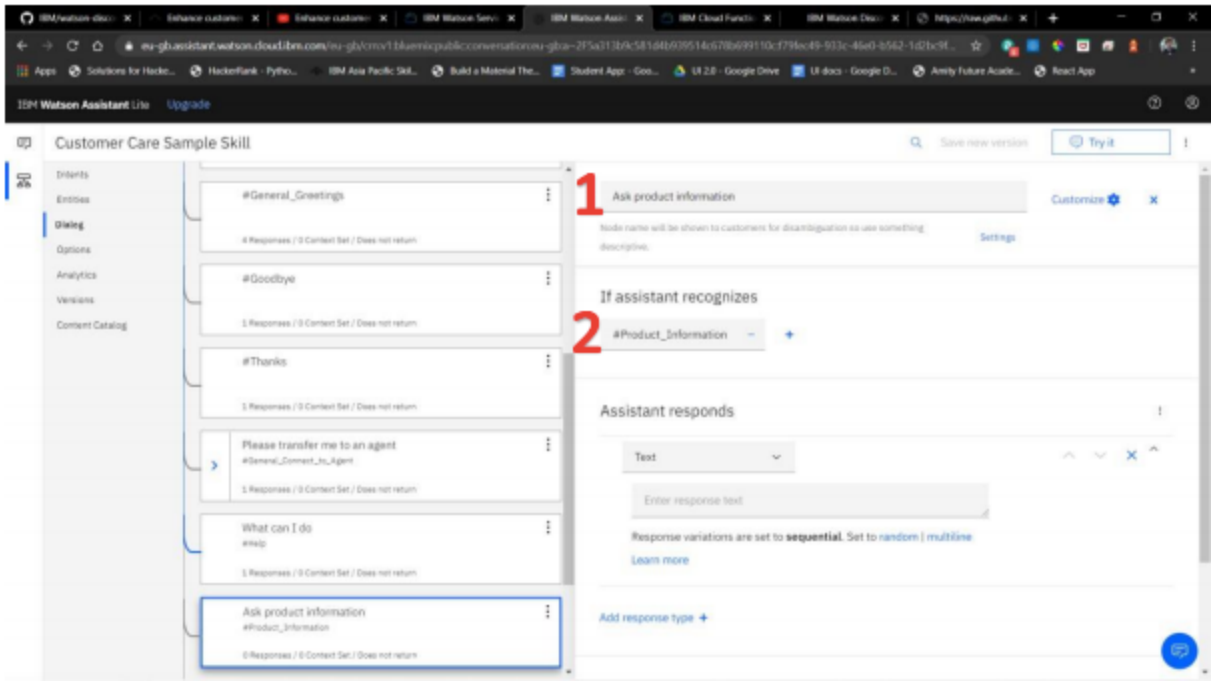
As the default customer care dialog does not have a way to deal with any questions involving outside resources, so we will need to add new intent. Create a new intent that can detect when the user is asking about operating the Ecobee thermostat. From the Customer Care Sample Skill panel, select the Intents 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.

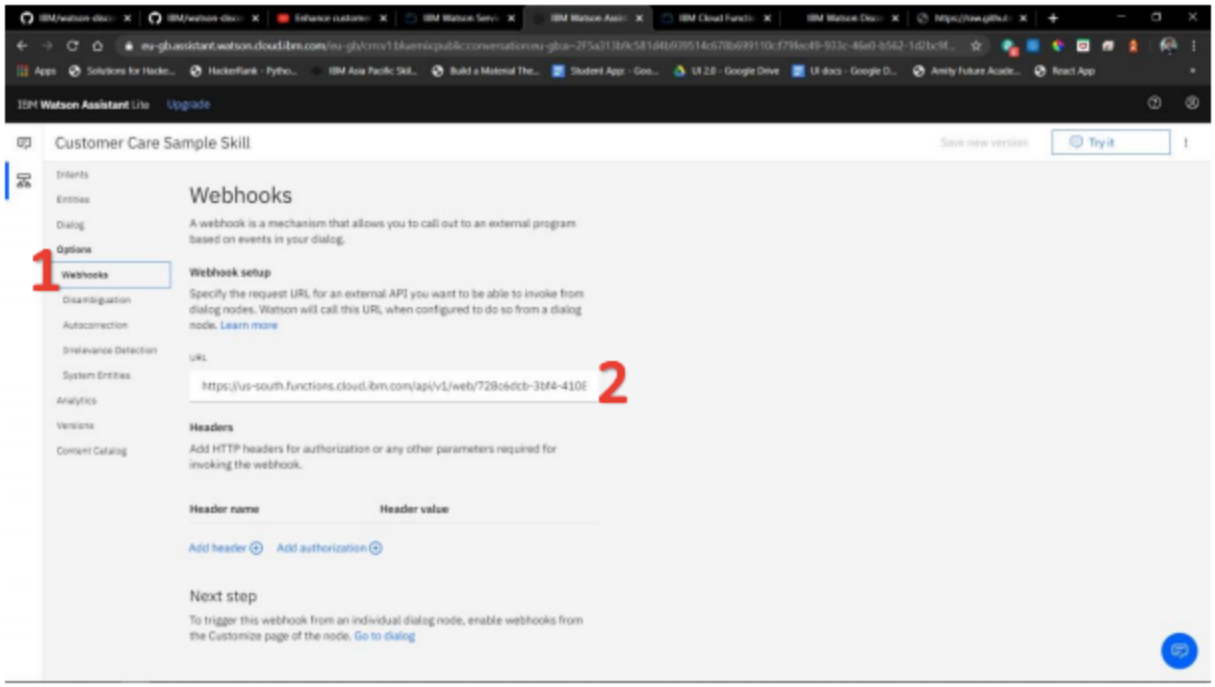


Go back to the previous page after doing this, then click on Dialog Tab and add a node below "What can I do node".

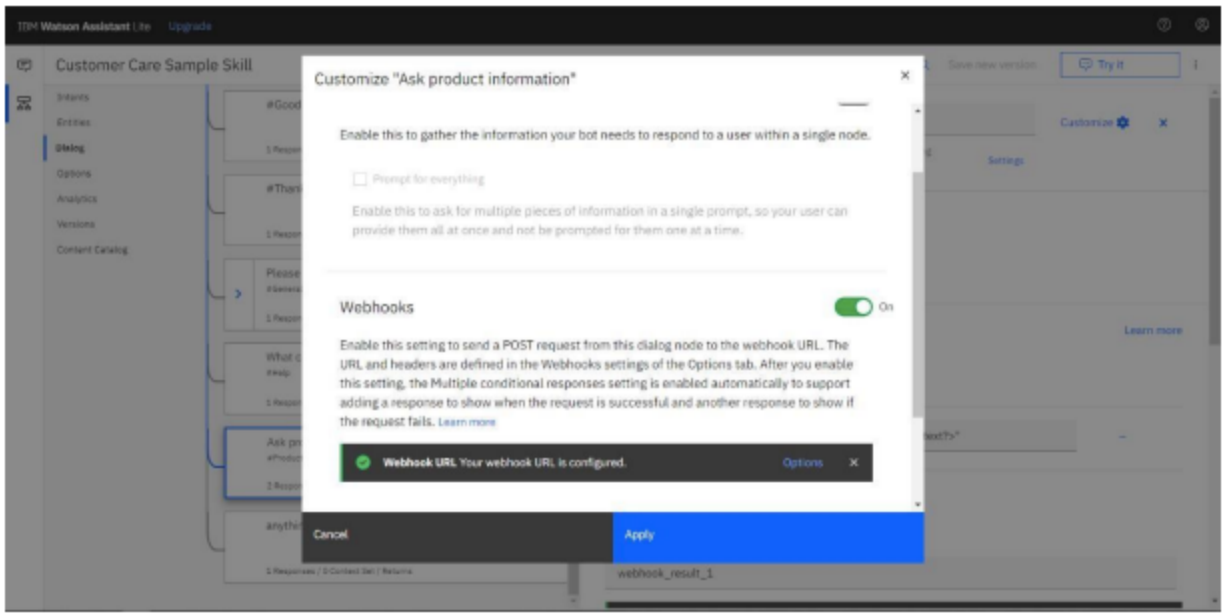




Name the node "Ask product information" [1] and assign it our new intent #Product_Information [2]. This means that if Watson Assistant recognizes a user input such as "how do I set the time?", it will direct the conversation to this node. Before proceeding further, let's learn about webhook. A webhook 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.

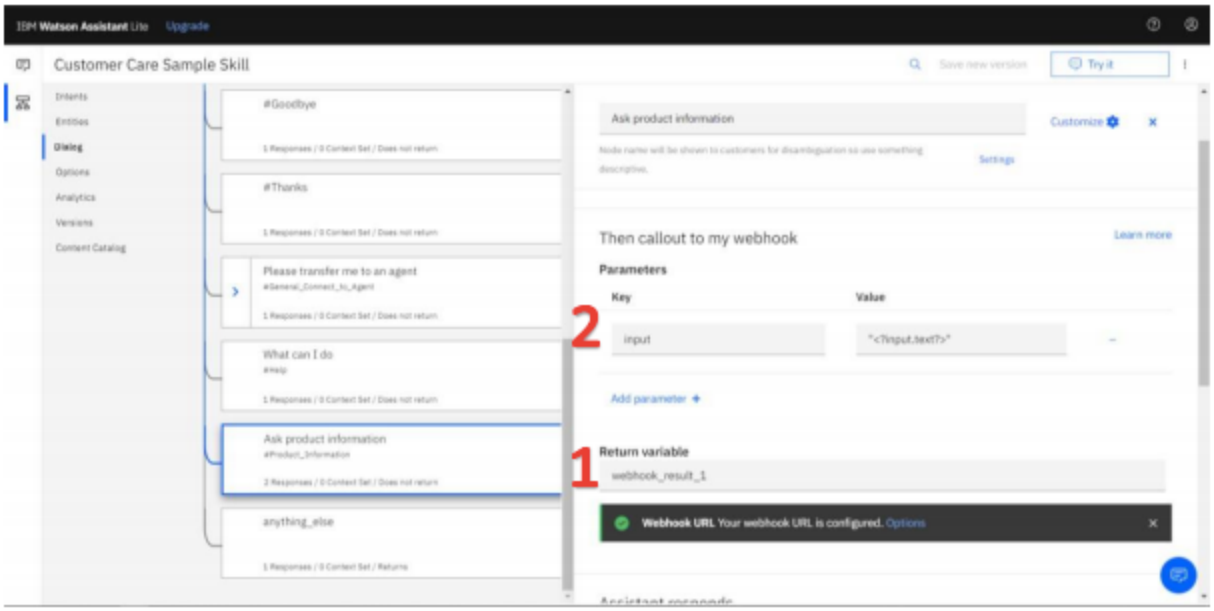


Click Webhooks[1] tab and enter the URL[2] to enable web hook for the IBM Cloud Functions action you created in Step 4.Add .json to the end of the URL to specify the result should be in JSON format. Return to the Dialog tab, and click on the Ask product information node. From the details panel for the node, click on Customize, and enable Webhooks for this node.

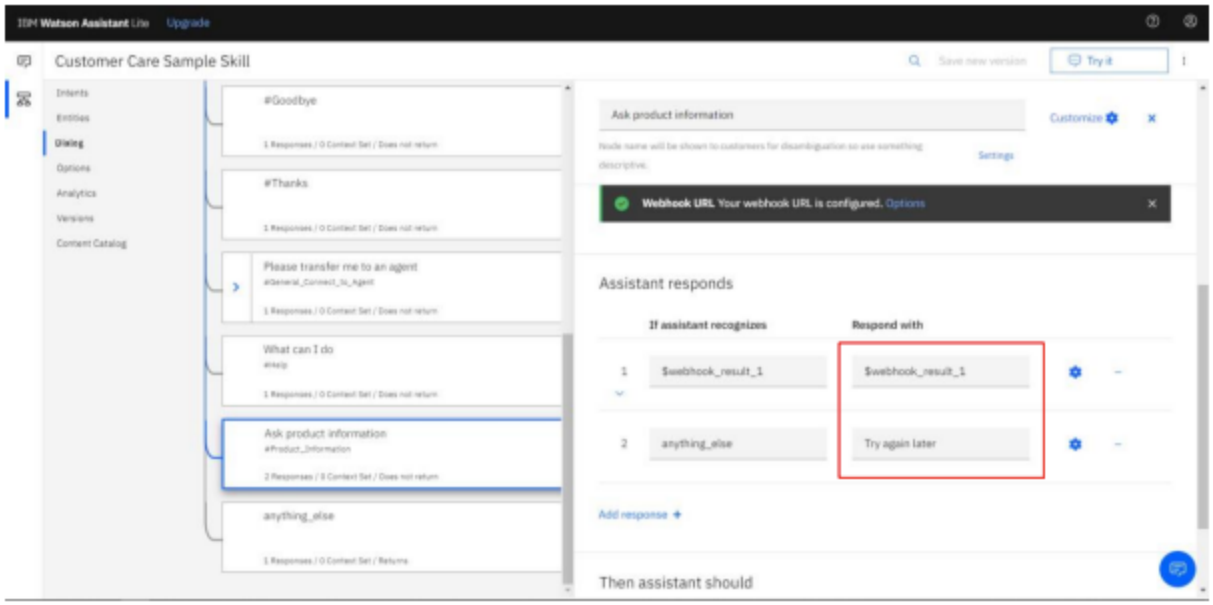


Click Apply. The dialog node should have a Return variable [1] set automatically to

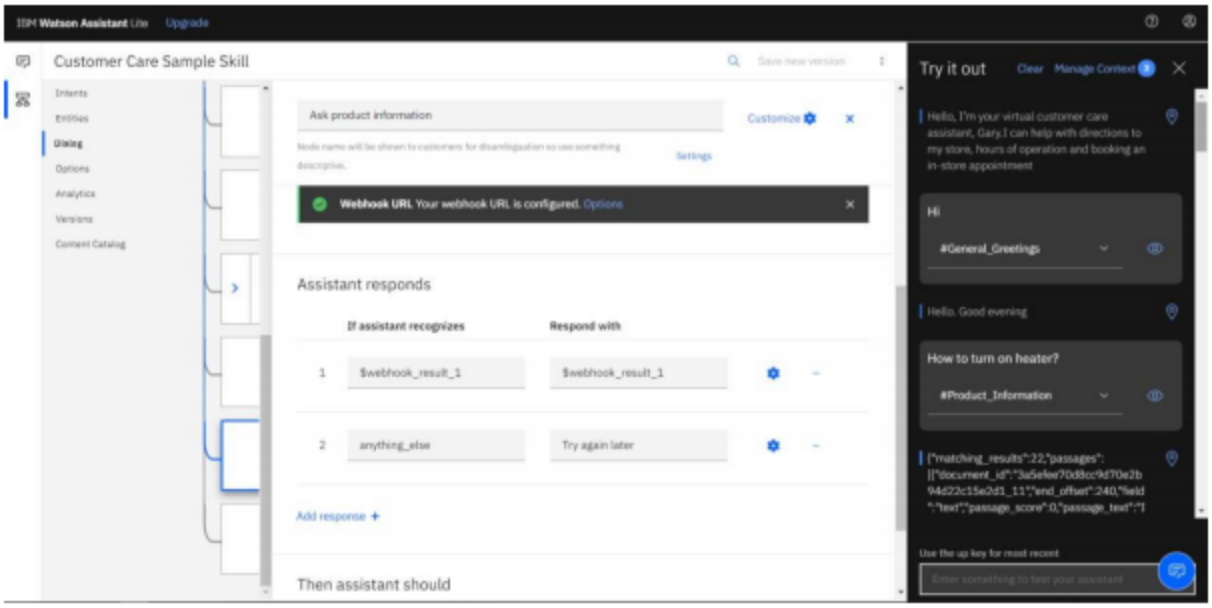
$webhook_result_1. This is the variable name you can use to access the result from the Discovery service query.



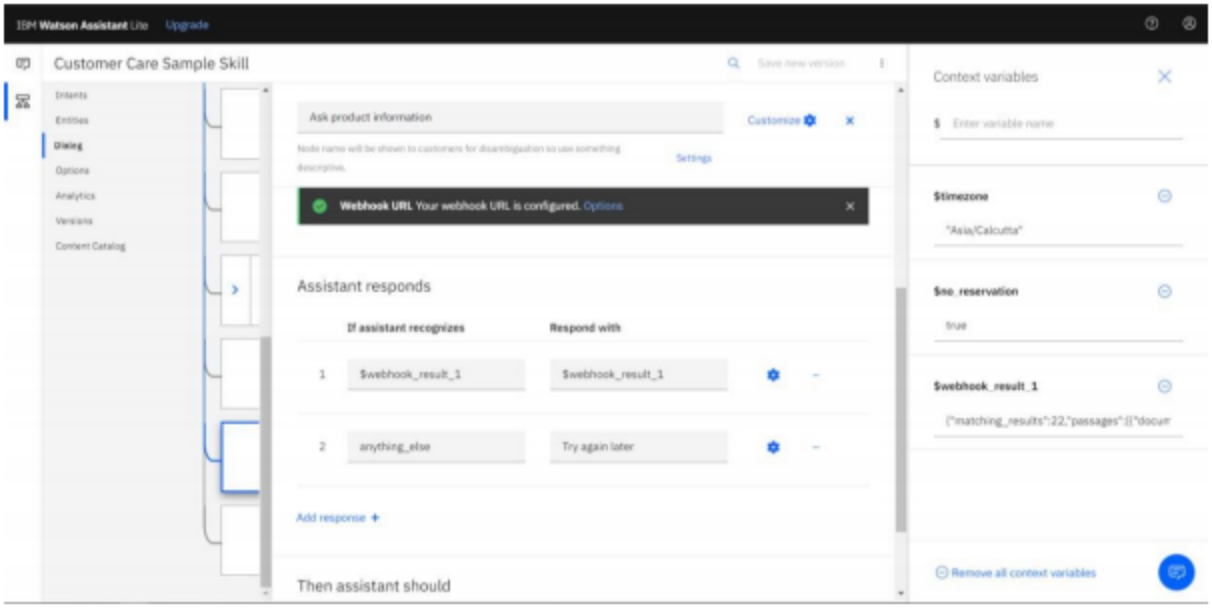
You will also need to pass in the users question via the parameter input [2]. The key needs to be set to the value: "" If you fail to do this, Discovery will return results based on a blank query. Optionally, you can add these responses to aid in debugging:



Now we should test the assistant by clicking on Try it button located at the top right side of the panel.

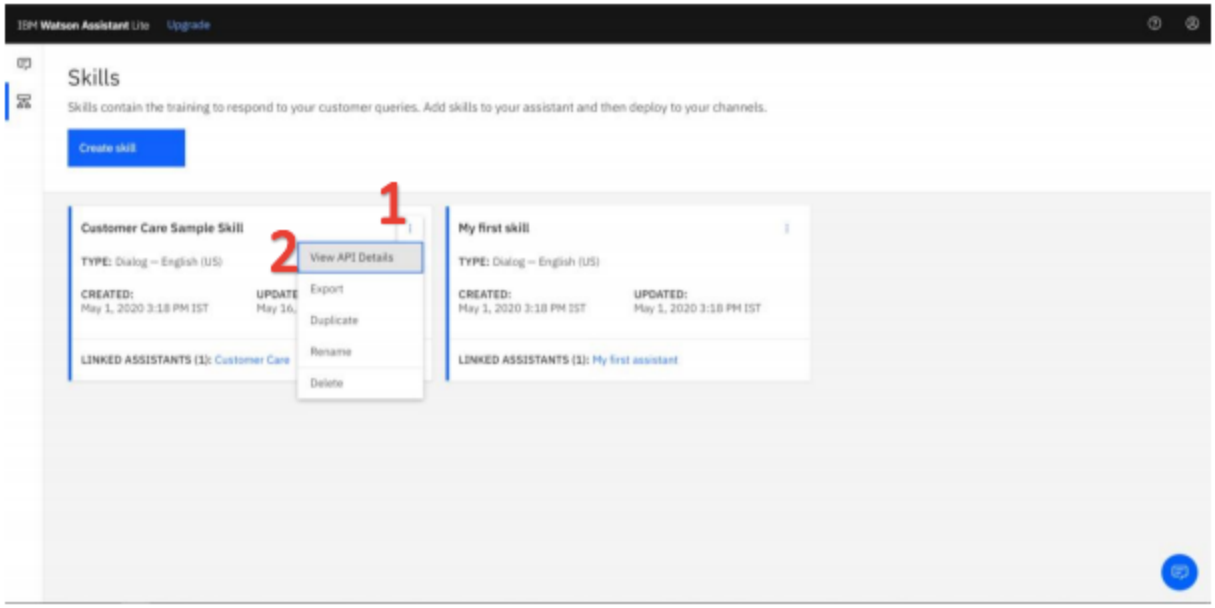


Note that the input "how do I turn on the heater?" has triggered our Ask about product dialog node, which is indicated by the #Product_Information response. And because we specified that $webhook_result_1.passages be the response, that value is displayed also. You can also verify that the call was successfully 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.

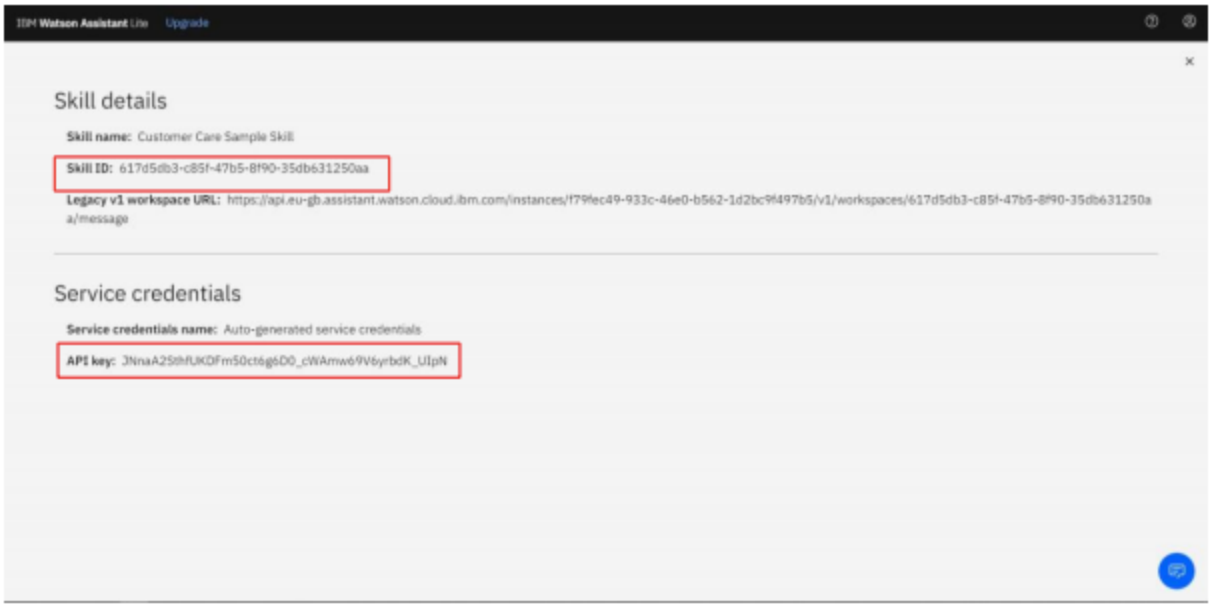


For upcoming steps, you will need to provide some credentials to access your assistant so to store credentials for future use follow these steps below. Go back to the skills tab,

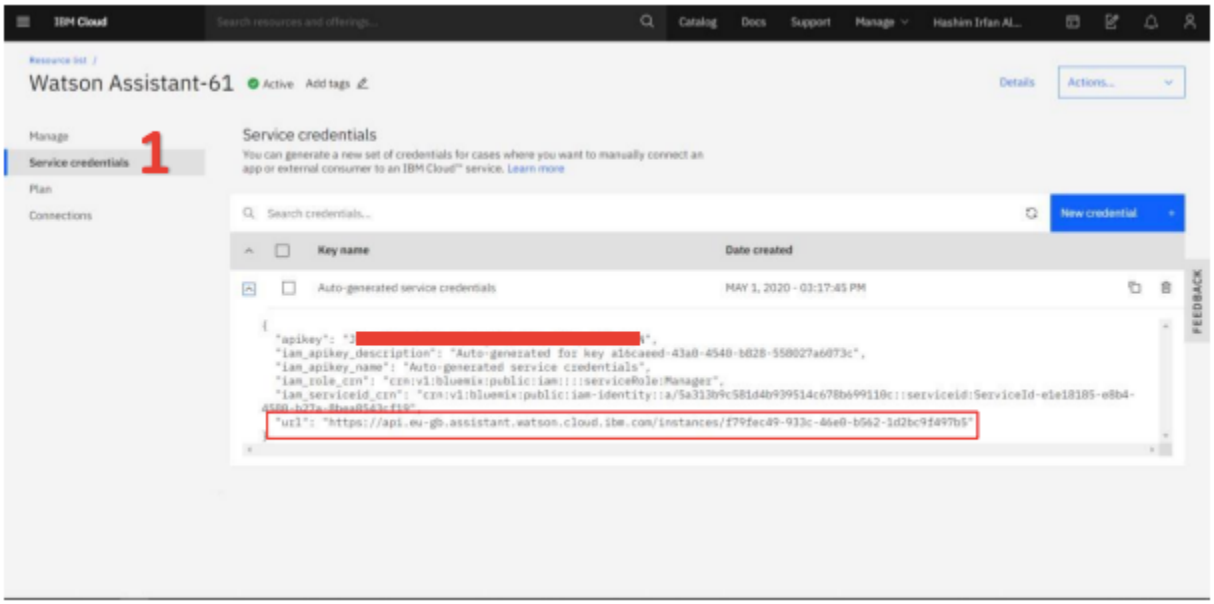
click [1] and then [2]



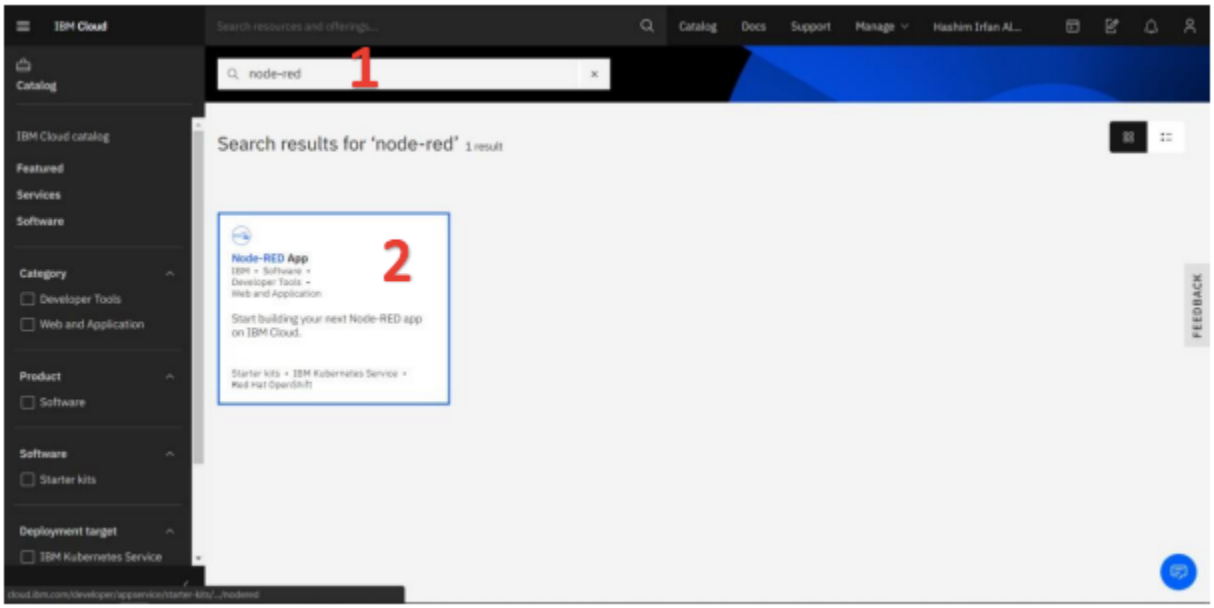
The Skill ID and API Key is to be noted.



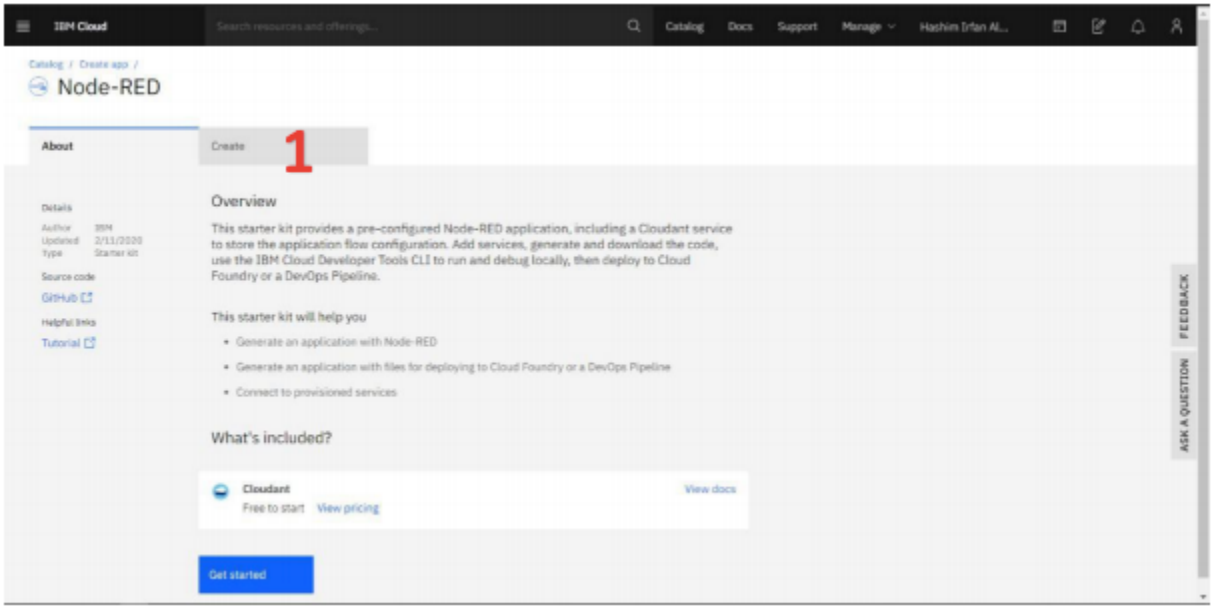
Go Back to the Watson Assistant Resource List, Select Service Credentials [1] and make note of the URL.APIKEY can be found here too.



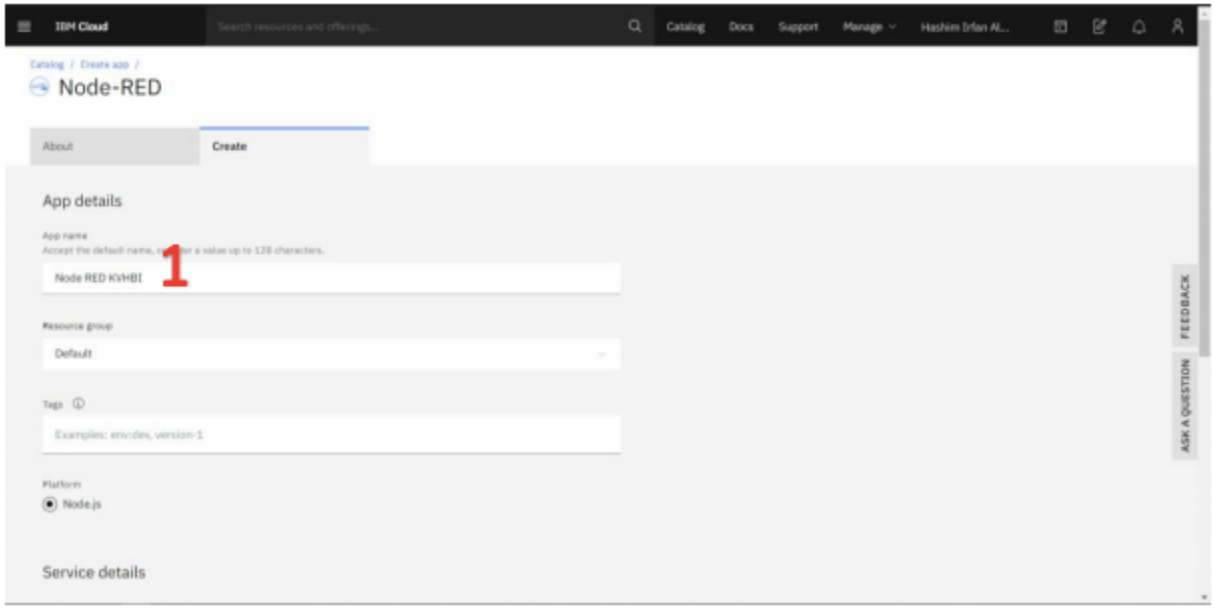
6. Build Node-RED Flow to Integrate All Services - Now it's time to create Node-Red, go to IBM Cloud Dashboard, click on Create Resource and search for node-red[1].



Click on the Node-RED App tile [2]. This will show you an overview of the Starter Kit and what it provides.

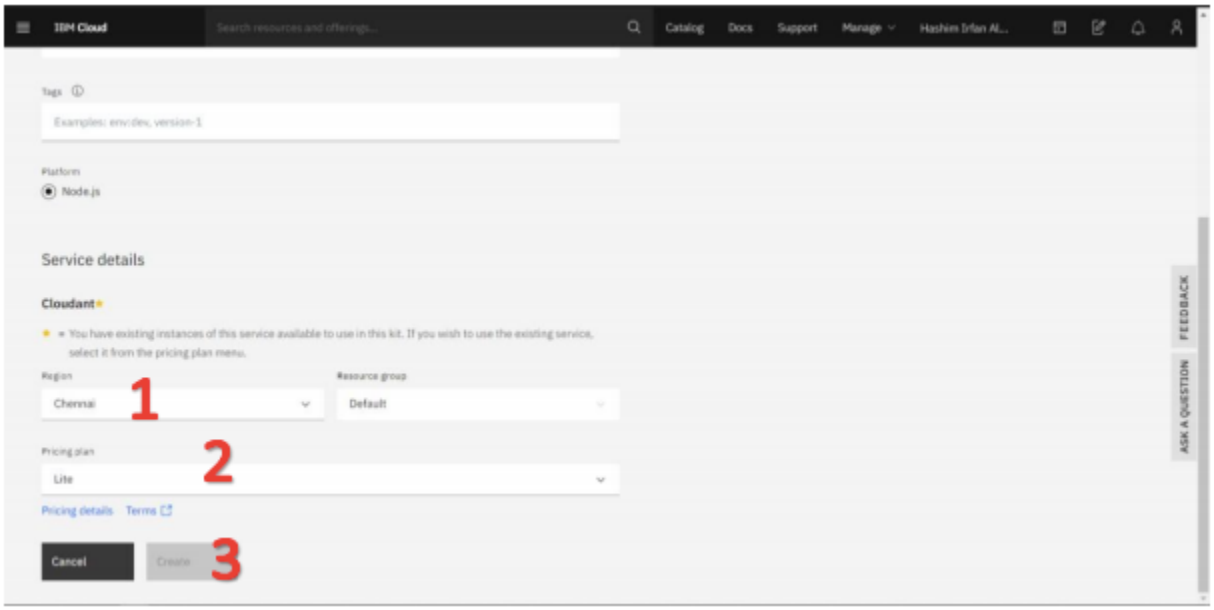


Click on Create [1].

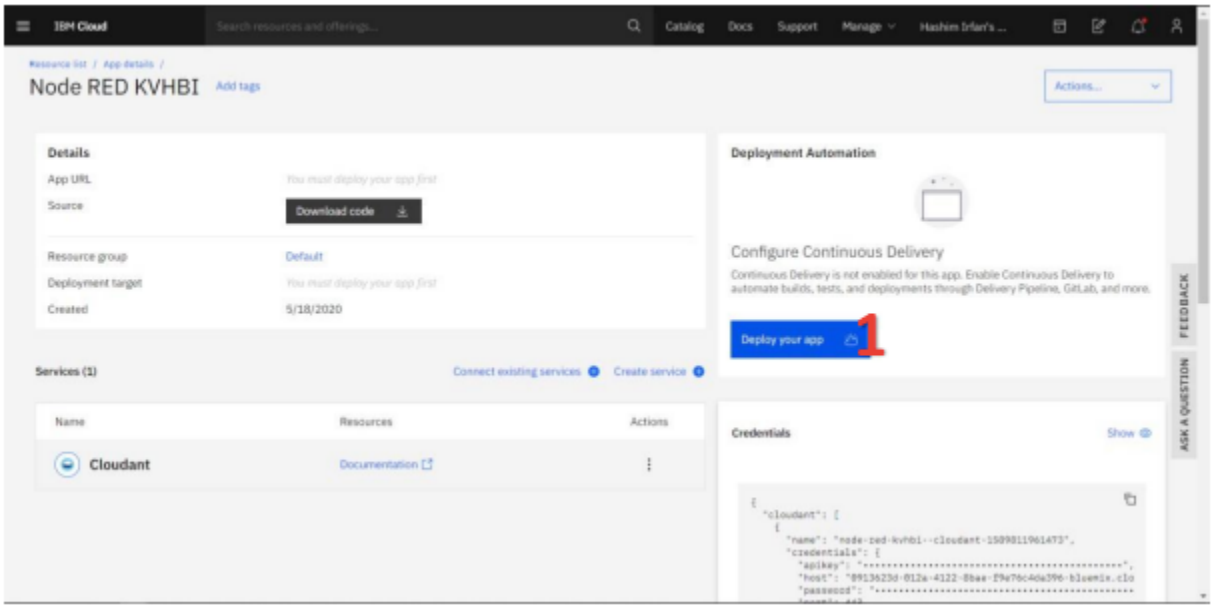


Now you need to configure the Node-RED Starter application. On the App details page, a randomly generated name will be suggested – Node RED KVHBI in the screenshot above. Either accept that default name or provide a unique name for your application [1]. This will become part of the application URL. Note: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

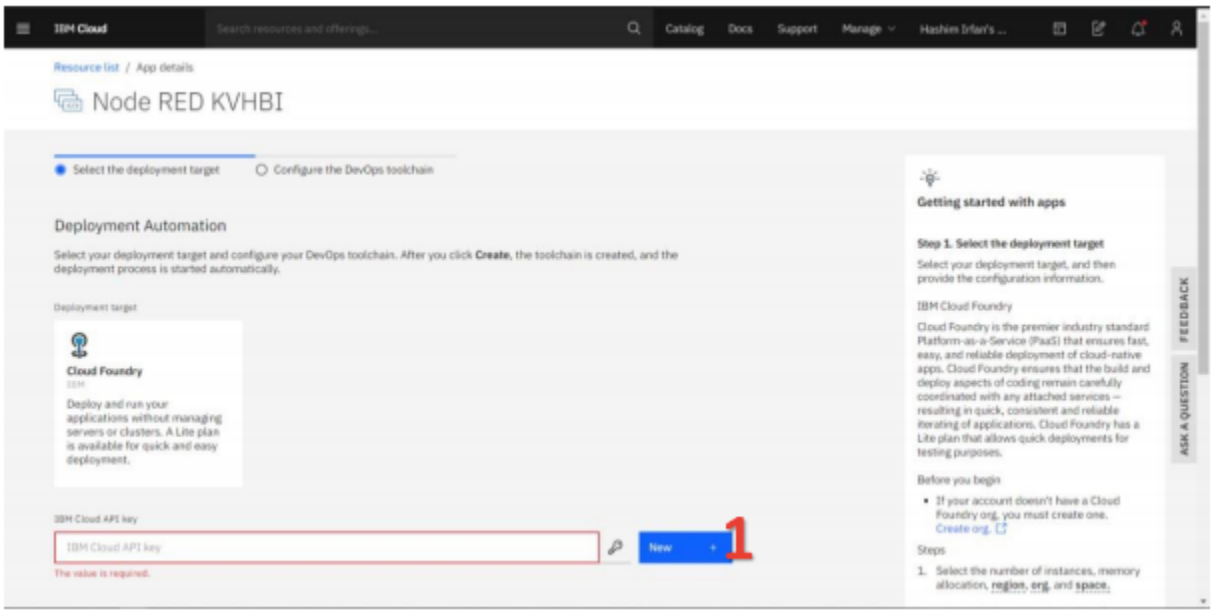
should be created in and what pricing plan it should use. You can only have one Cloudant instance using the Lite plan and you can have more than one Node-RED Starter application using the same Cloudant service instance. If you have already got an instance, you will be able to select it from the Pricing 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].

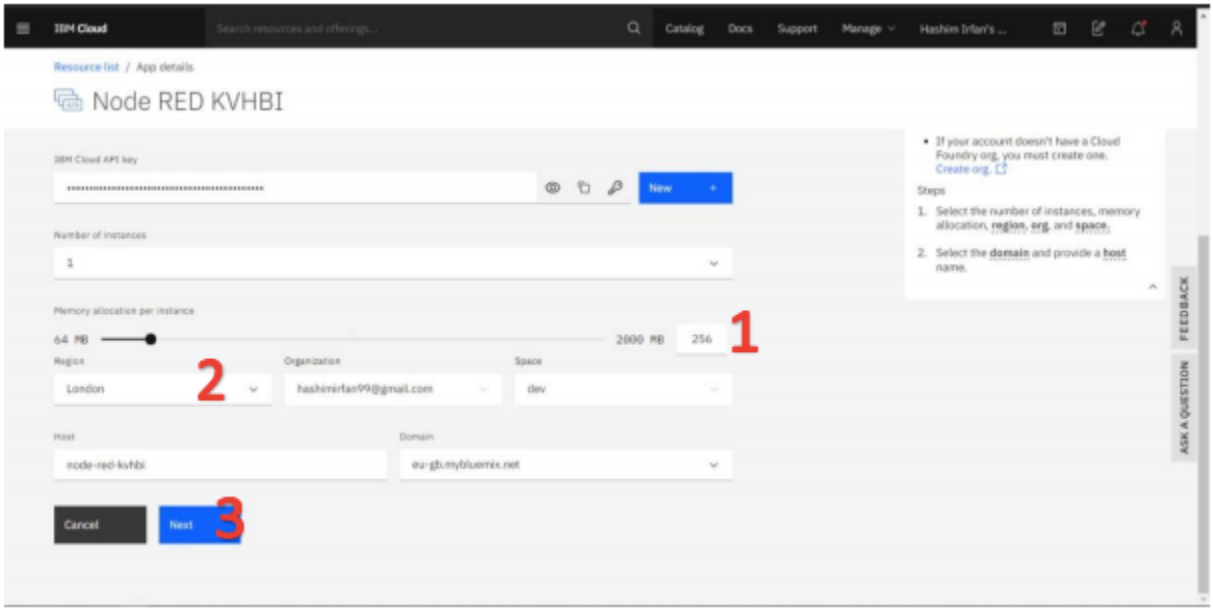


You will need to create an IBM Cloud API key to allow the deployment process to access your resources. Click the New button (1) to create the key. A message dialog will appear. Read what it says and then confirm and close the dialog.

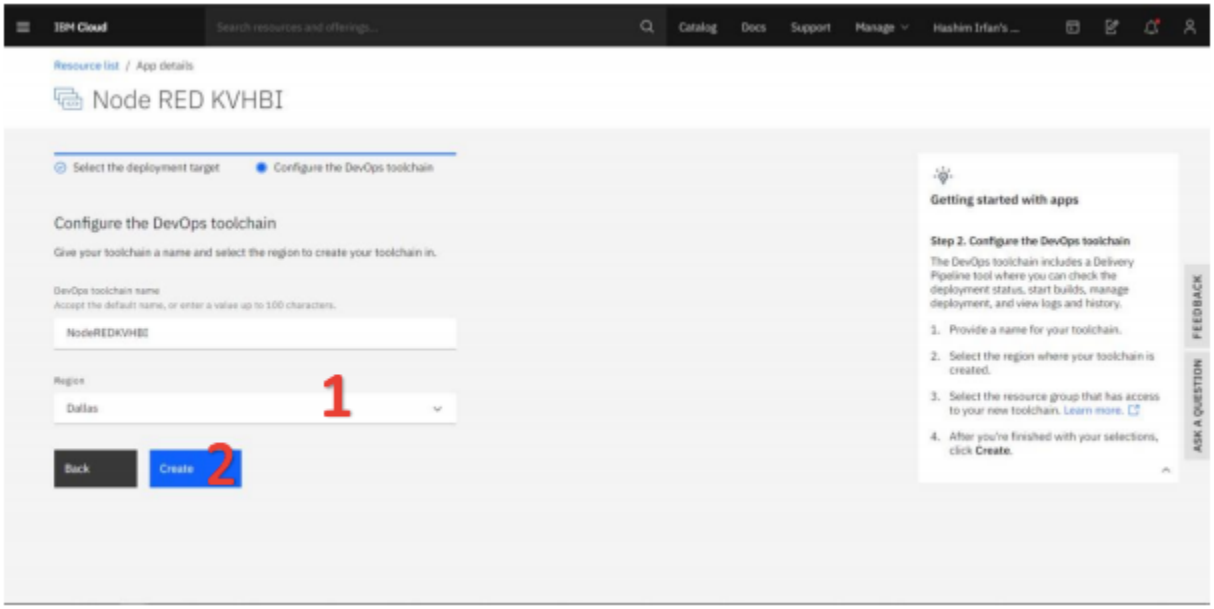


After creating the API Key, Increase the Memory allocation per instance slider [1] to 256MB. If you do not increase the memory allocation, your Node-RED application might not have sufficient memory to run successfully. The Node-RED Starter kit only supports deployment to the Cloud Foundry space of IBM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

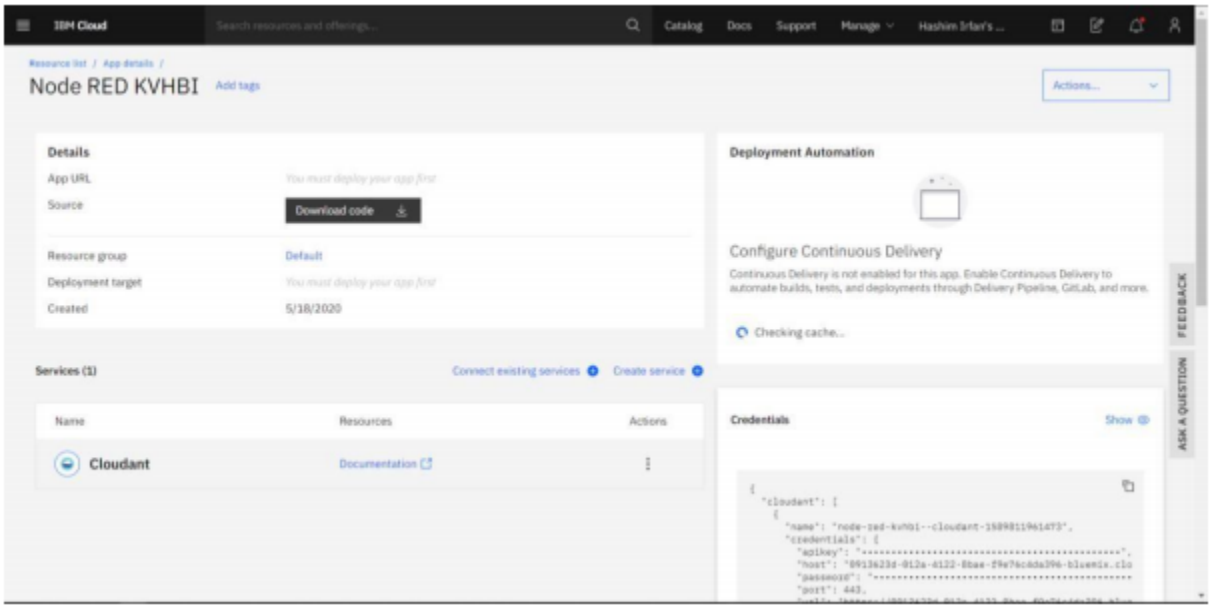
in.Click Next [3].



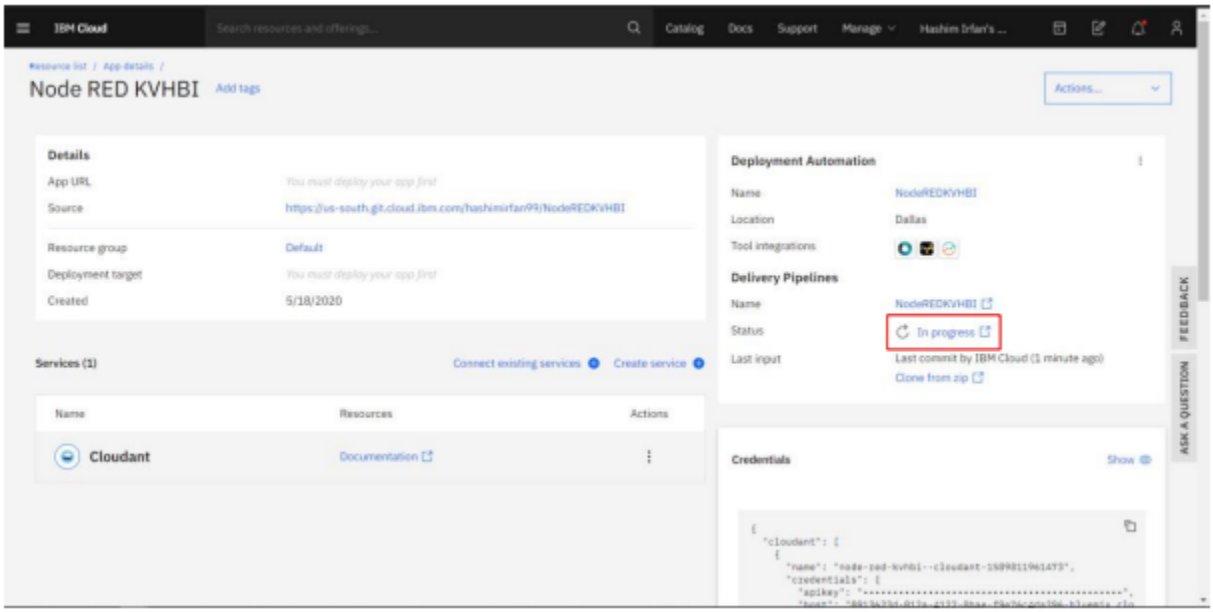
Now, select the region [1] to create the DevOps toolchain and then Click Create [2].



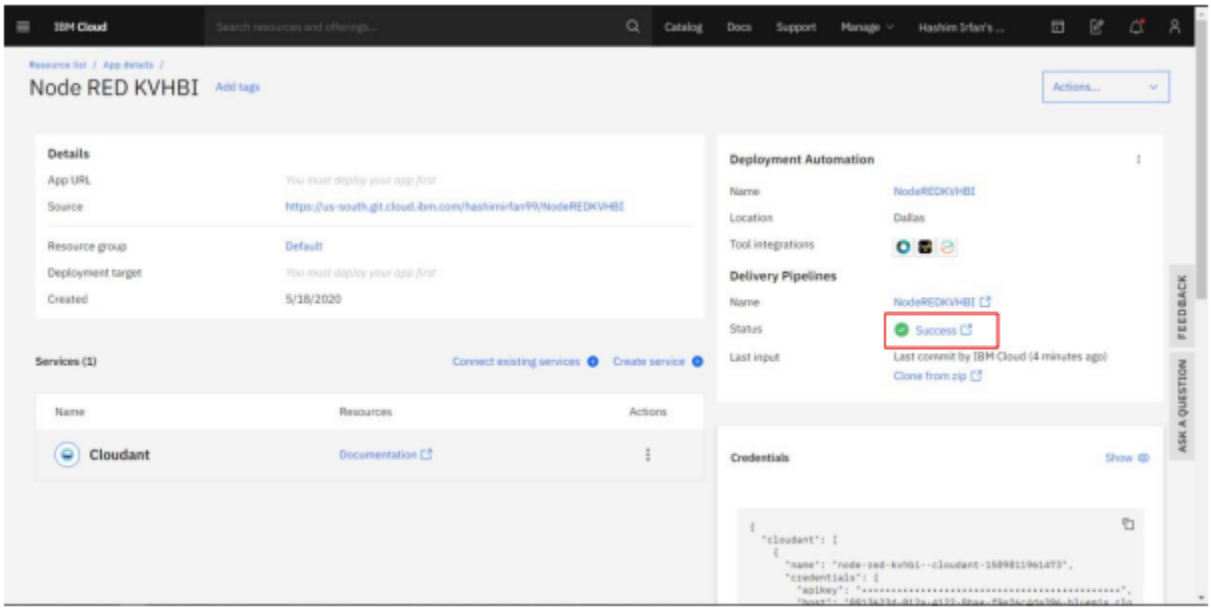
This will take you back to the application details page.



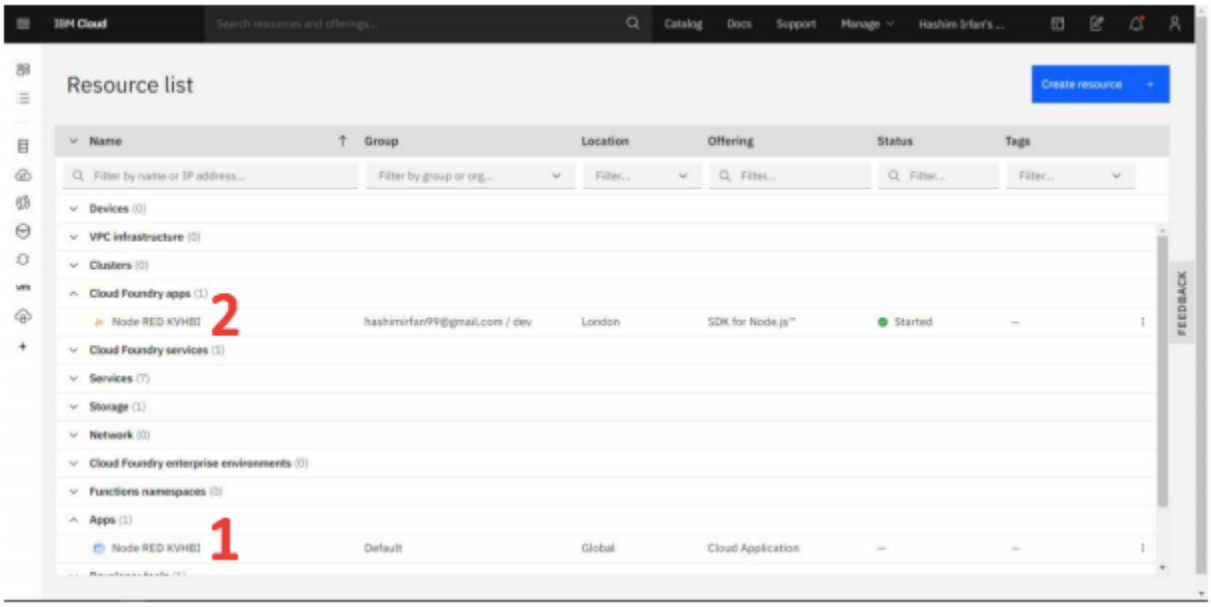
The Continuous Delivery section will refresh with the details of your newly created Toolchain. The Status field of the Delivery Pipeline will show "In progress". That means your application is still being built and deployed.



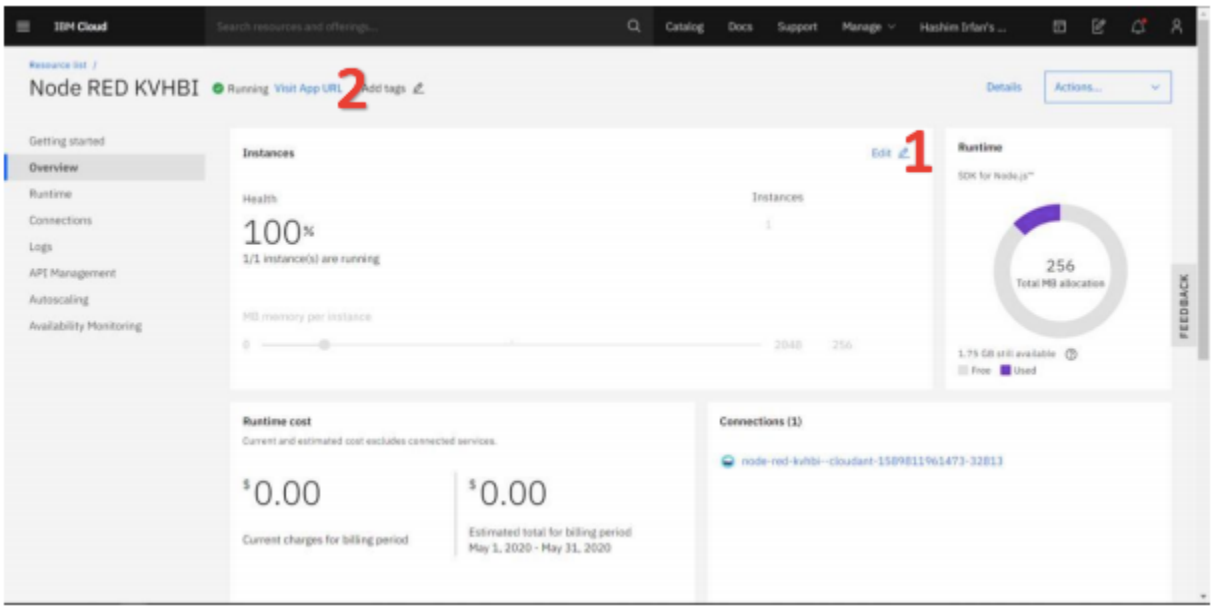
The Deploy stage will take a few minutes to complete. Eventually the Deploy stage will go green to show it has passed. This means your Node-RED Starter application is now running.



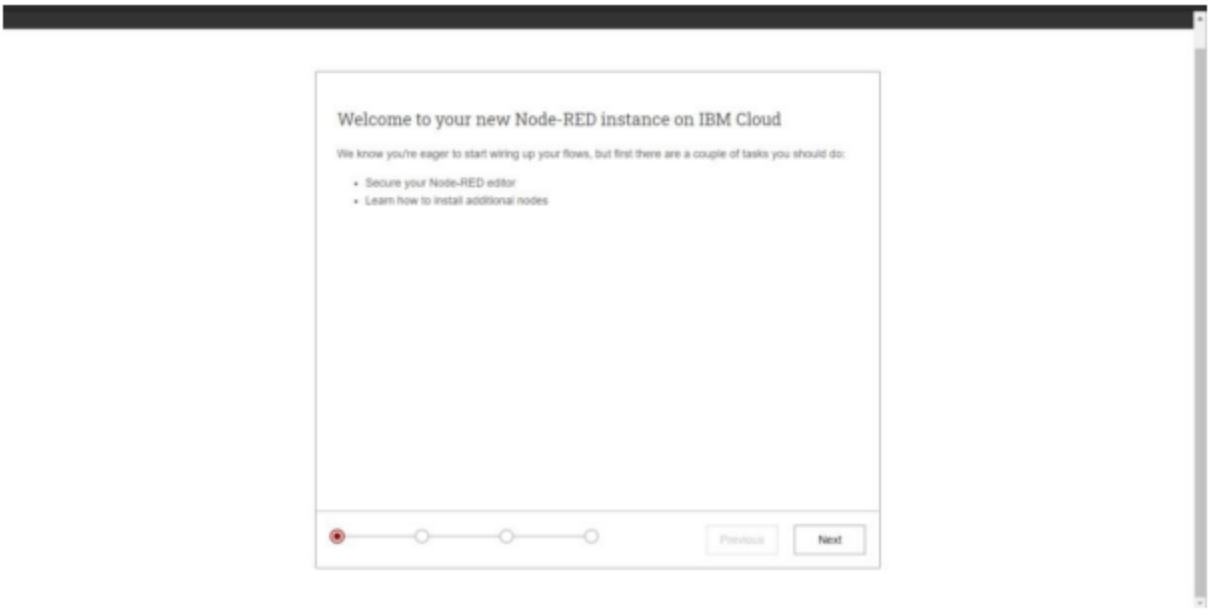
Now that you've deployed your Node-RED application, let's open it up! Open your IBM Cloud Resource list. You will see your newly created Node-RED Application listed under the Apps section [1]. You will also see a corresponding entry under the Cloud Foundry apps section [2].



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



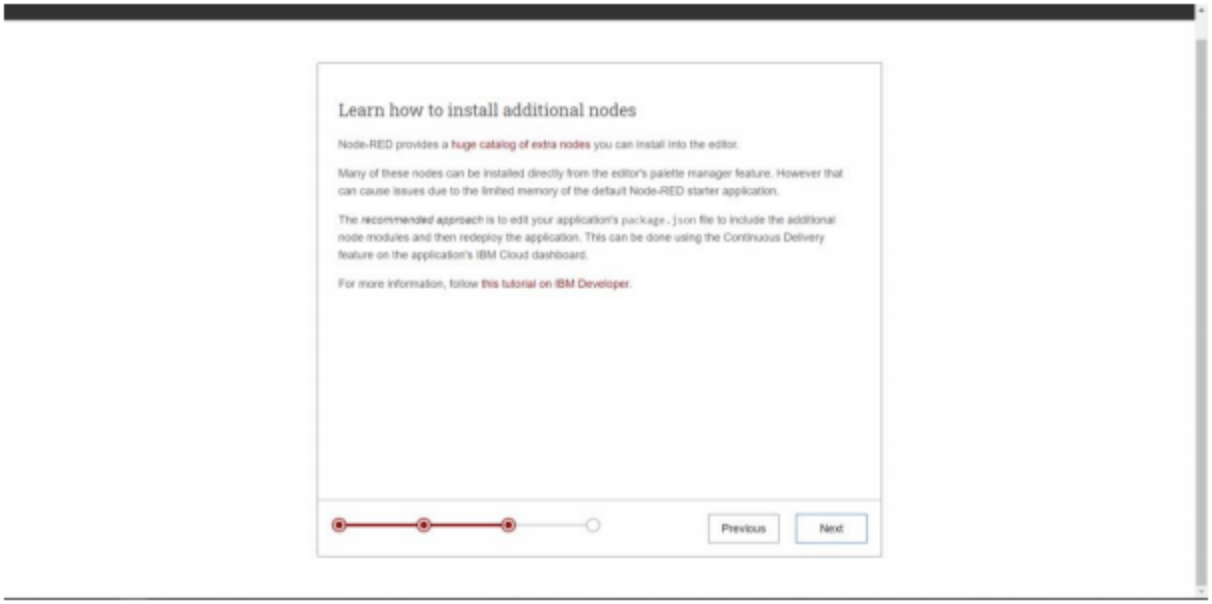
Special Cases: If your Runtime Instance is running full (0MB Free), Click on Edit[1] and reduce memory per instance to 128mb. If you have Free space on your runtime skip the previous step[1] and Click on Visit App URL[2].



Click on Next button.

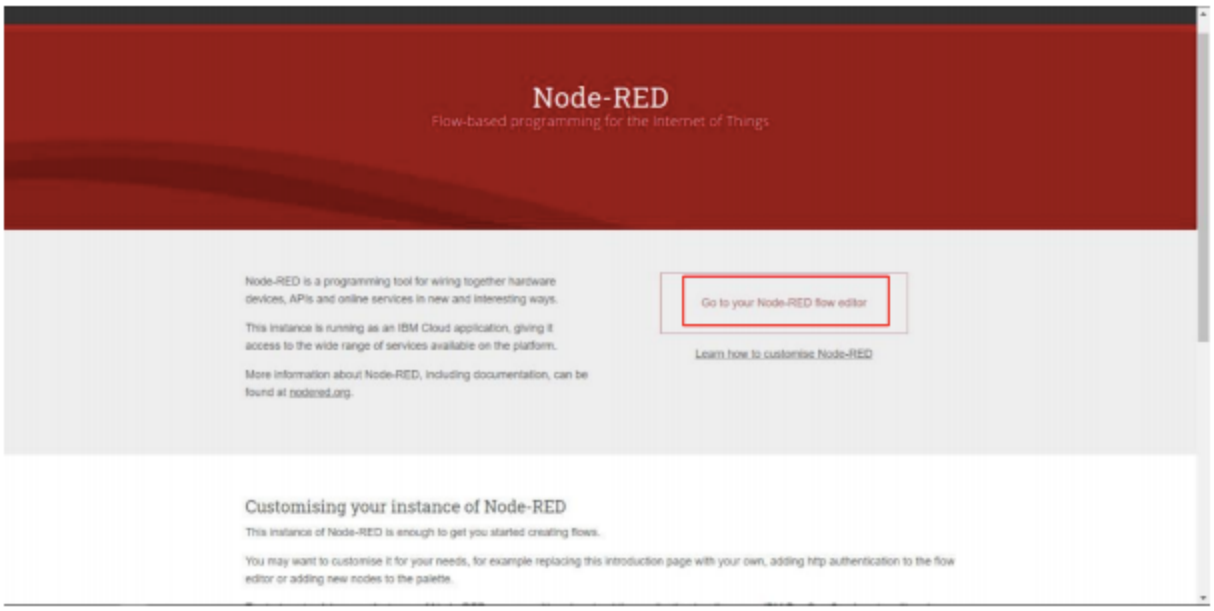
You can choose to secure your Node-RED editor by providing a username and password. I am selecting the other option which is Allow anyone to access the editor and make changes.

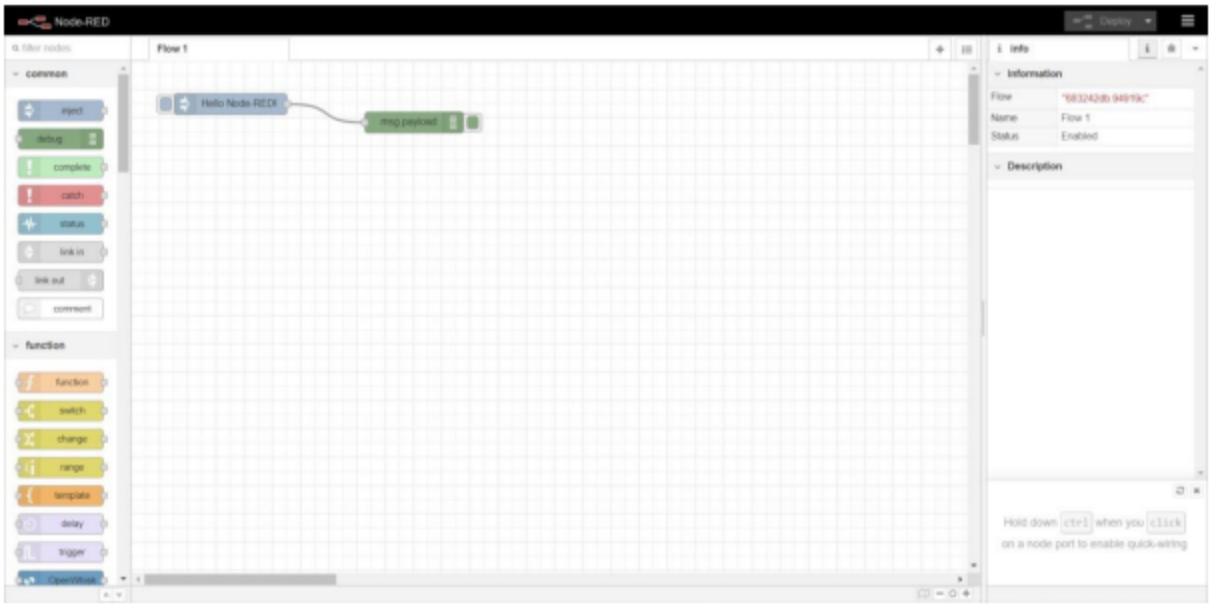
Tick the box, and click Next.



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.

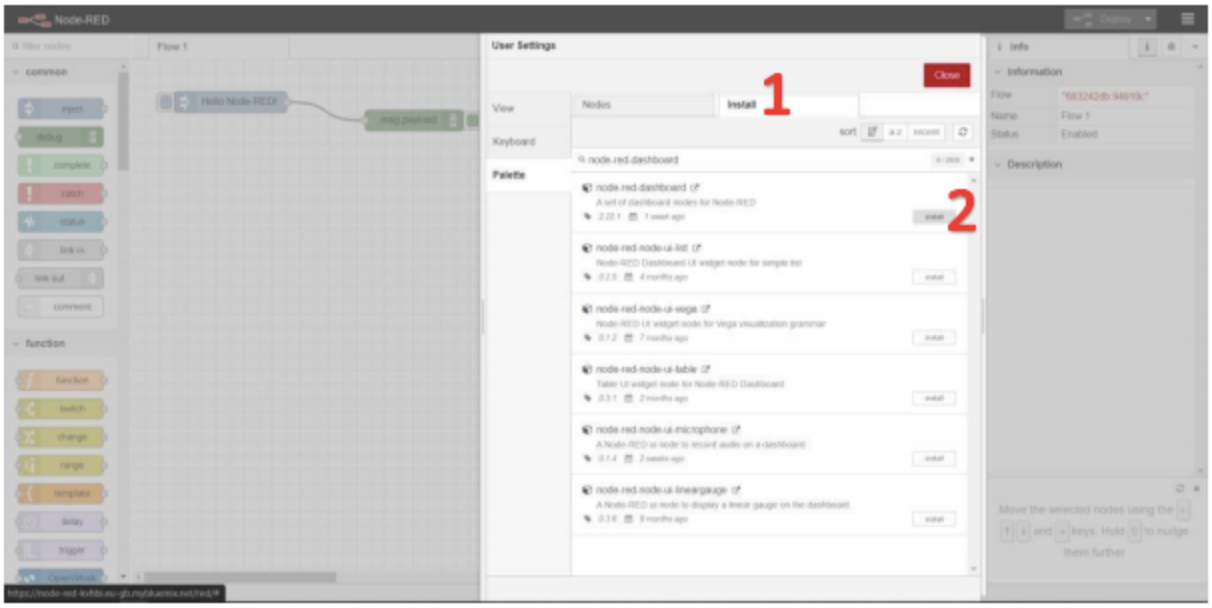




The Node-RED editor opens showing the default flow.

7.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].

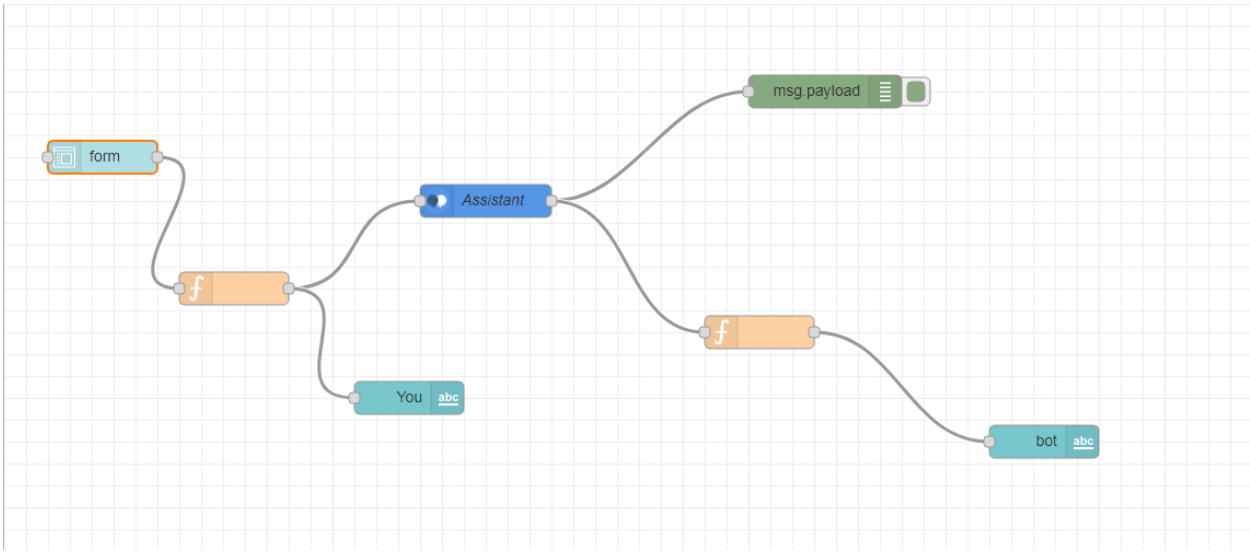
Go to Install Tab [1] and search for node-red-dashboard and Install [2] it.



5. FLOWCHART

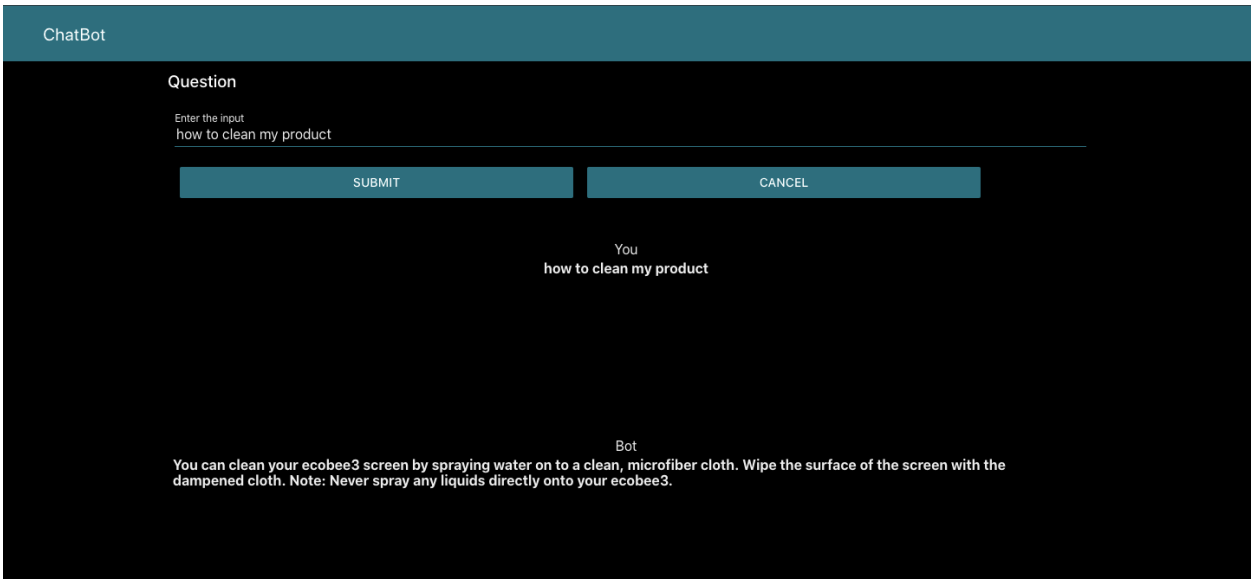
Create flow and configurenode:
At first go to manage pallete and install node-red-dashboard. Now,Create the flow with the help of following node:

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



6. RESULT

Finally our Node-RED dash board integrates all the components and displayed in the Dashboard UI by typing URL-
https://node-red-gargsul.eu-gb.mybluemix.net/ui/#!/0?socketid=ohllRTBGxWEnr1tWAABc



7. ADVANTAGES & DISADVANTAGES

Advantages:

- Campanies can deploy chatbots to rectify simple and general human queries.
- Reduces man power.
- Cost efficient.
- No need to divert calls to customer agent and customer agent can look on other works**.**
- Faster Customer Service.
- Increased Customer Satisfaction.
- 24-7 availability.
- Multiple Customer Handling.

Disadvantages:

- Some times chatbot can mislead customers.
- Giving same answer for different sentiments.
- Some times cannot connect to customer sentiments and intentions.
- Limited Responses for Customers.
- Maintenance.

8. APPLICATIONS

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

9. CONCLUSION

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

10. FUTURE SCOPE

We can include Watson studio text to speech and speech to text services to access the chat-bot hands-free. This is one of the future scope of this project.

11. BIBILOGRAPHY

APPENDIX

A.Source Code (cloud function) -

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

B. Reference

1.https://developer.ibm.com/patterns/enhance-customer-help-desk-with-smart-document- understanding/ 2.https://github.com/IBM/watson-discovery-sdu-with-assistant
3.https://www.youtube.com/watcht?v=-yniuX-Poyw&feature=youtu.be
4.https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/

THE END