

# 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.

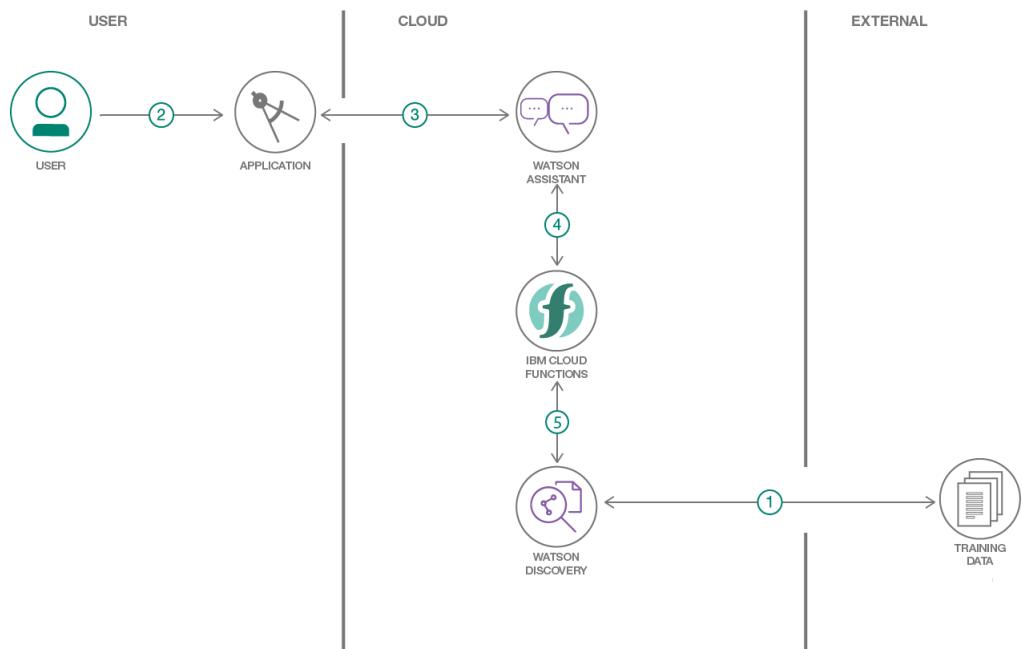
### 2.2 Proposed solution

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



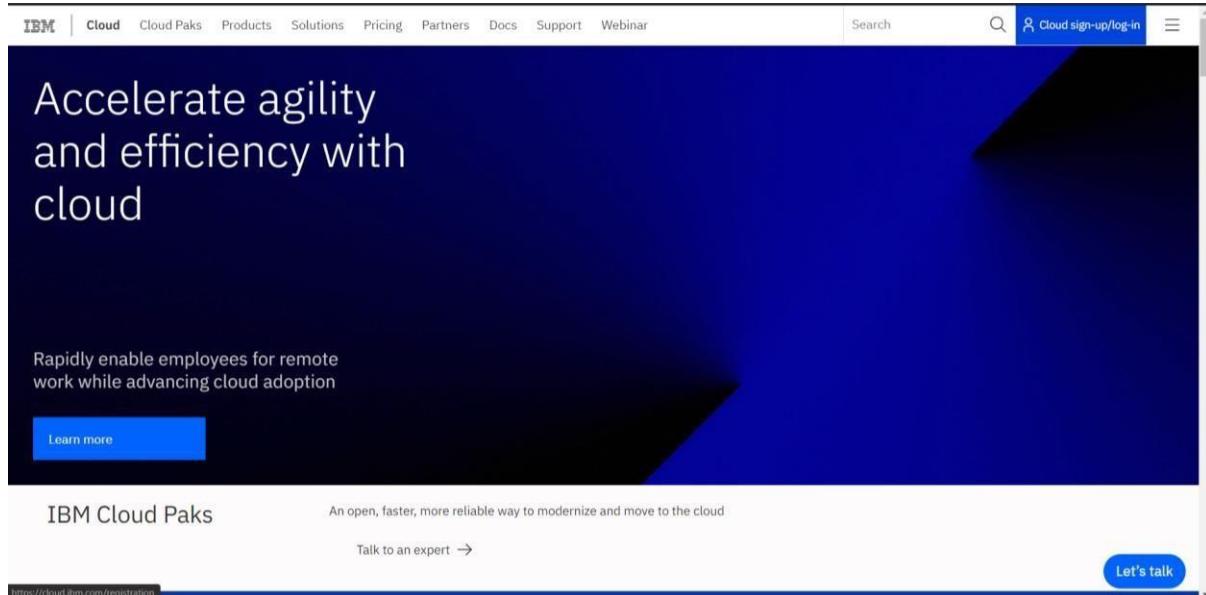
#### 3.2 Hardware / Software designing

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

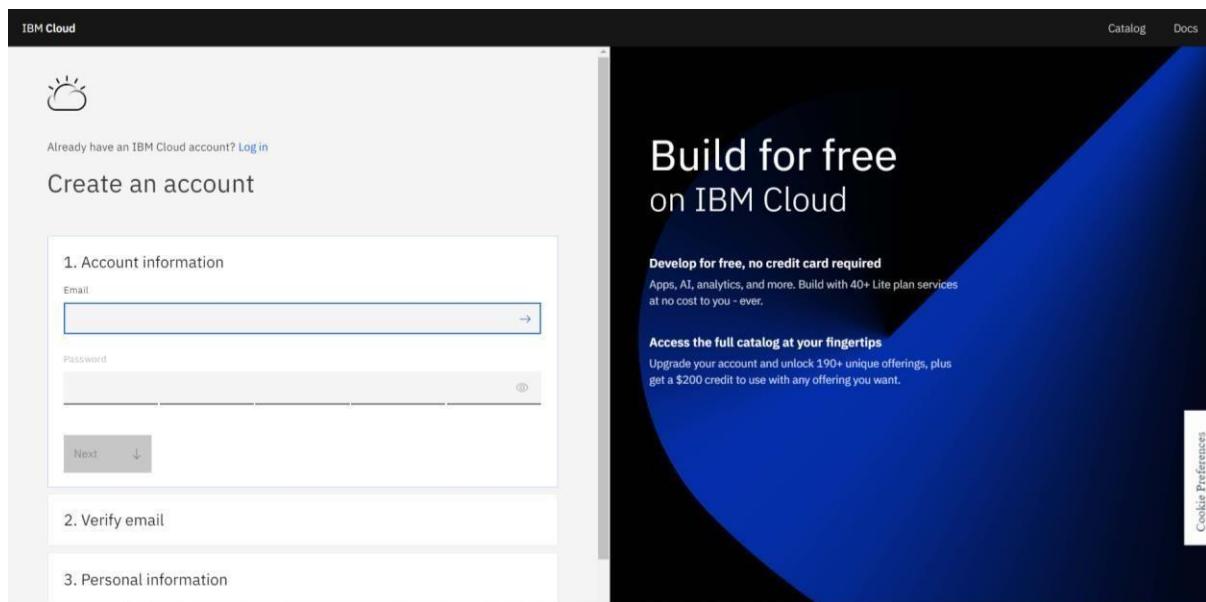
### 1. Create IBM Cloud Services

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



The screenshot shows the top navigation bar of the IBM Cloud website with links for Cloud, Cloud Paks, Products, Solutions, Pricing, Partners, Docs, Support, and Webinar. A search bar and a 'Cloud sign-up/log-in' button are also visible. The main banner features the text 'Accelerate agility and efficiency with cloud' and a subtext 'Rapidly enable employees for remote work while advancing cloud adoption'. A 'Learn more' button is present. Below the banner, there's a section for 'IBM Cloud Paks' with a subtext 'An open, faster, more reliable way to modernize and move to the cloud' and a 'Talk to an expert' link. A 'Let's talk' button is located on the right side of the page.

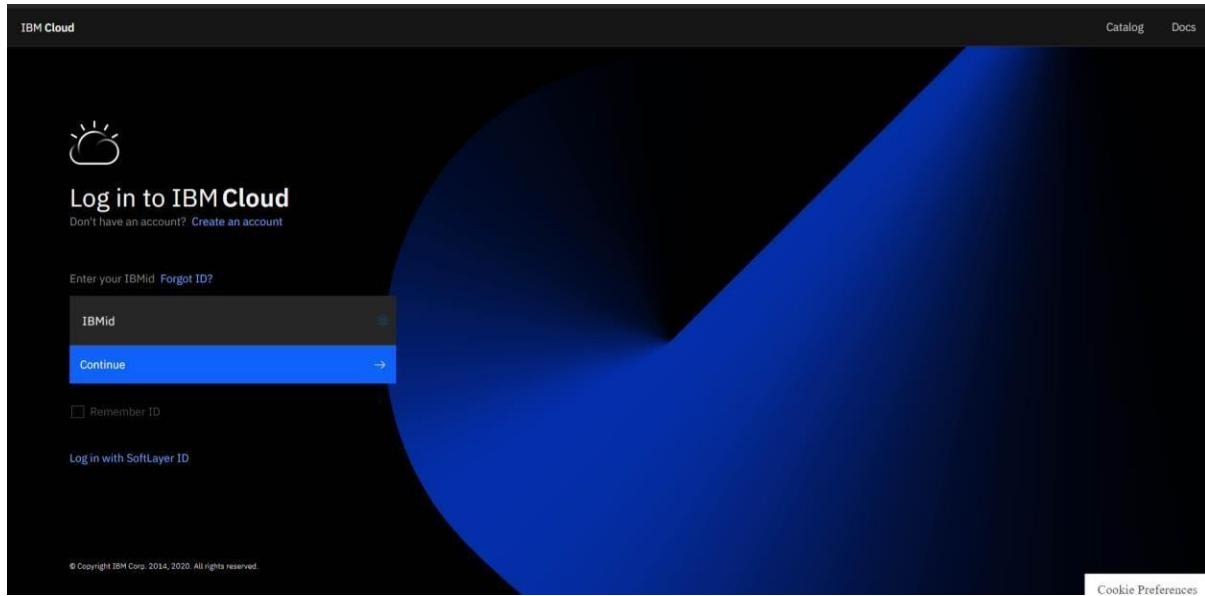
Click on  to sign up or login to IBM Cloud.



The screenshot shows the 'Create an account' step of the sign-up process. It includes fields for 'Email' and 'Password', and a 'Next' button. To the right, there's a promotional section for 'Build for free on IBM Cloud' with sub-sections for 'Develop for free, no credit card required' and 'Access the full catalog at your fingertips'. A 'Cookie Preferences' link is located in the bottom right corner of the sidebar.

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 [Log in](#) and Log in to your cloud account.



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

A screenshot of the IBM Cloud Dashboard. The top navigation bar includes 'IBM Cloud', a search bar, and links for 'Catalog', 'Docs', 'Support', 'Manage', and a user profile. On the left, a sidebar shows resource counts: 13 Resources, 1 Cloud Foundry app, 2 Cloud Foundry services, 6 Services, 1 Storage, 1 Functions namespaces, and 1 App. The main dashboard features several cards: 'Resource summary' (13 resources), 'Planned maintenance' (clear skies), 'For you' (lightboard video on Infrastructure as Code), 'News' (Airtel Selects IBM and Red Hat to build Open Hybrid Cloud Network, Vodafone Idea Limited Achieves Major Production Milestone with IBM and Red Hat for its Open Universal Hybrid Cloud for Network a...), 'Recent support cases' (empty), 'User access' (manage users), and 'IBM Cloud status' (world map).

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

The screenshot shows the IBM Cloud Catalog interface. On the left, there's a sidebar with categories like Catalog, Featured, Services, and Software. The main area displays "IBM Cloud products" with a message about over 190+ products. Below this is a search bar labeled "Search the catalog...". A section titled "Recommended for you" lists several services: App ID, Object Storage, Visual Recognition, Streaming Analytics, Continuous Delivery, and Speech to Text. Each service has a small icon, a name, a category, and a brief description. At the bottom right of the main area is a blue feedback button.

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

The screenshot shows the IBM Cloud Catalog search results for 'discovery'. The search bar at the top contains the query 'discovery'. Below the search bar, the results are displayed under the heading 'Search results for \'discovery\' 2 results'. There are two items listed: 'Discovery' and 'HashiCorp Consul'. The 'Discovery' item is highlighted with a blue border. It has a small icon, the name 'Discovery', its category 'IBM • Services • AI', a brief description 'Add a cognitive search and content analytics engine to applications.', and a status 'Lite • Free • IAM-enabled'. The 'HashiCorp Consul' item also has a small icon, the name 'HashiCorp Consul', its category 'Third party • Software • Developer Tools', a brief description 'Highly available and distributed service discovery and key-value store designed with support for the modern data cent...', and a status 'Helm charts • IBM Kubernetes Service • Free'.

Click on



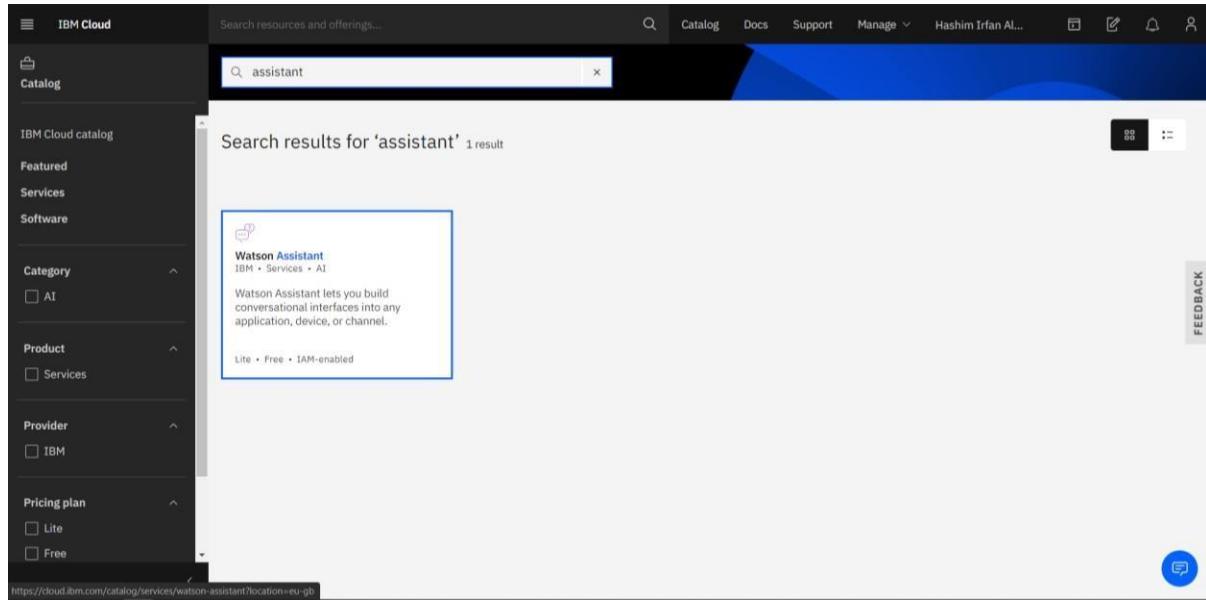
The screenshot shows the 'Discovery' service details page in the IBM Cloud interface. It includes a 'Summary' section with basic info like Region: London, Plan: Lite, and Service name: Discovery-qk. Below this, there's a 'Create' button and a 'About' tab. The main area shows configuration steps: 'Select a region' (set to London) and 'Select a pricing plan' (set to Lite). The Lite plan table lists features like 0 - 1,000 documents per month and 200 news queries per month. A note states that services are deleted after 30 days of inactivity. On the right side, there's a 'Feedback' button and a sidebar with 'Create', 'Add to estimate', and 'View terms' buttons.

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



The screenshot shows the IBM Cloud Catalog interface. In the top navigation bar, there is a search bar with the placeholder "Search resources and offerings...". Below the search bar, the word "assistant" is typed into the search field. The main content area displays the search results for "assistant", showing one result titled "Watson Assistant". The result card includes a small icon of a speech bubble with a brain, the service name "Watson Assistant", the provider "IBM", and the category "Services > AI". A brief description states: "Watson Assistant lets you build conversational interfaces into any application, device, or channel." Below the description, it says "Lite • Free • IAM-enabled". The URL of the page is visible at the bottom: <https://cloud.ibm.com/catalog/services/watson-assistant?location=eu-gb>.

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.

The screenshot shows the 'Resource summary' section of the IBM Cloud dashboard. It displays 13 resources across several categories:

Category	Count
Cloud Foundry apps	1
Cloud Foundry services	2
Services	6
Storage	1
Functions namespaces	1
Apps	1

At the bottom right, there is a button labeled 'Add resources +'

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

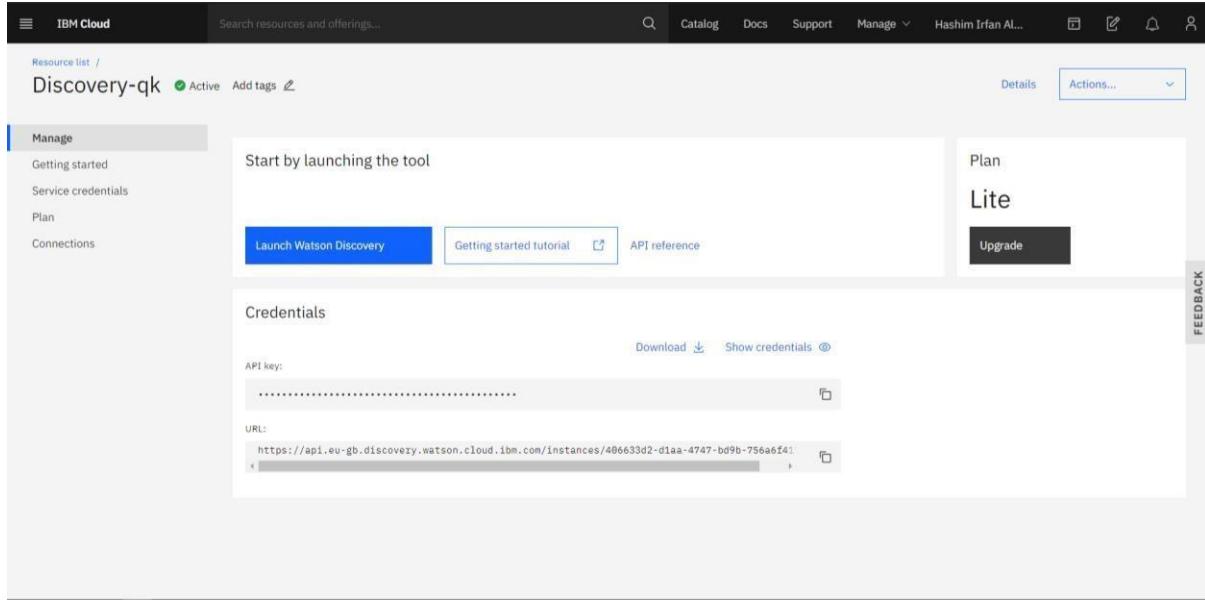
The screenshot shows the 'Resource list' page in the IBM Cloud dashboard, focusing on the 'Services' category. The table lists six services, all marked as 'Active':

Name	Group	Location	Offering	Status	Tags
Continuous Delivery	Default	London	Continuous Delivery	Active	
Discovery-qk	Default	London	Discovery	Active	
Watson Assistant-61	Default	London	Watson Assistant	Active	
Watson Studio-4h	Default	London	Watson Studio	Active	
node-red-qeyad-cloudant-1589268036...	Default	Chennai 01	Cloudant	Active	
watson-vision-combined-eq	Default	Dallas	Visual Recognition	Active	cpda...

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

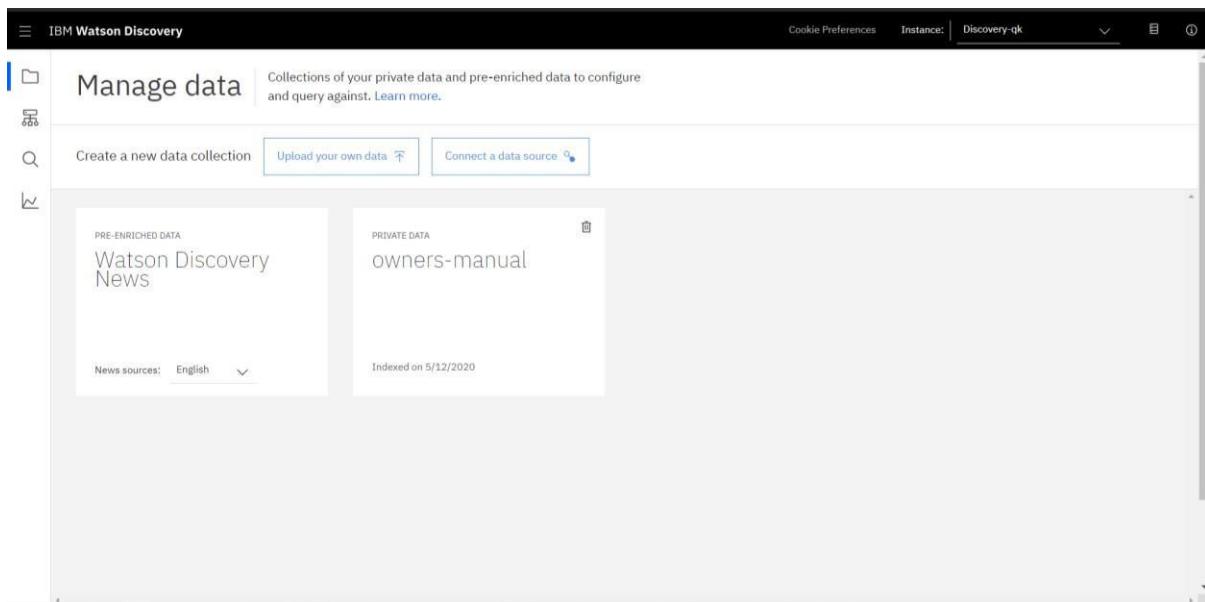
## 2. Configure Watson Discovery

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



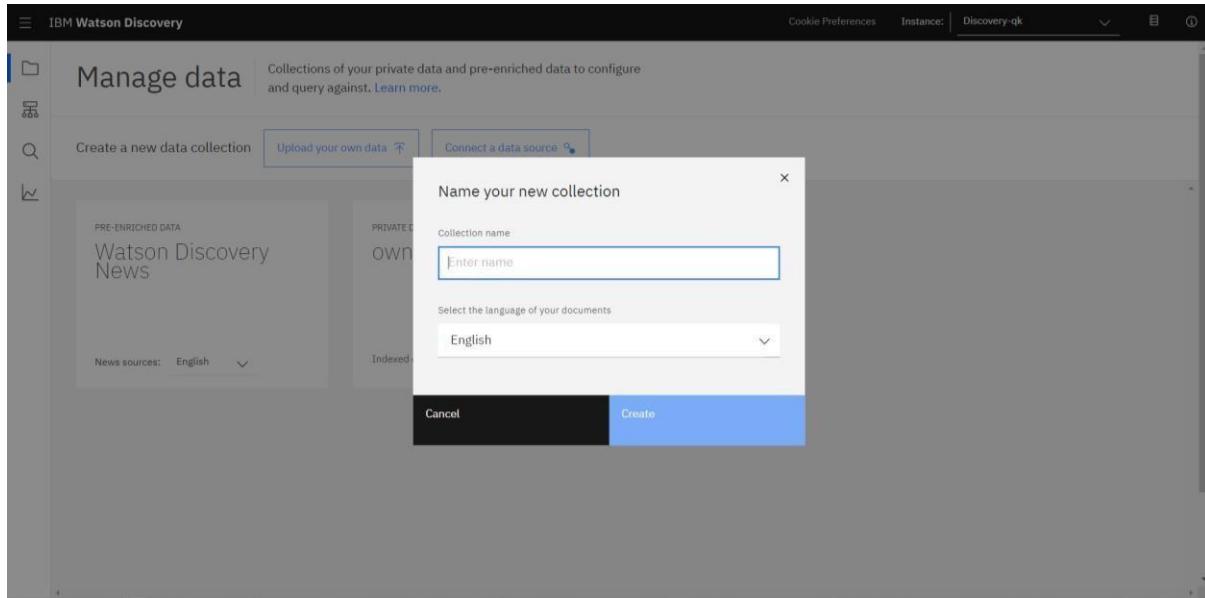
The screenshot shows the IBM Cloud Resource List interface. A service card for "Discovery-qk" is selected, indicated by a blue border. The card displays basic information: "Active", "Add tags", "Manage" (with sub-options: Getting started, Service credentials, Plan, Connections), "Start by launching the tool" (with "Launch Watson Discovery" button), "Getting started tutorial", "API reference", "Plan" (set to "Lite"), and "Upgrade". Below the card, there's a "Credentials" section with an API key and URL.

Click on  to launch Watson Discovery Service.

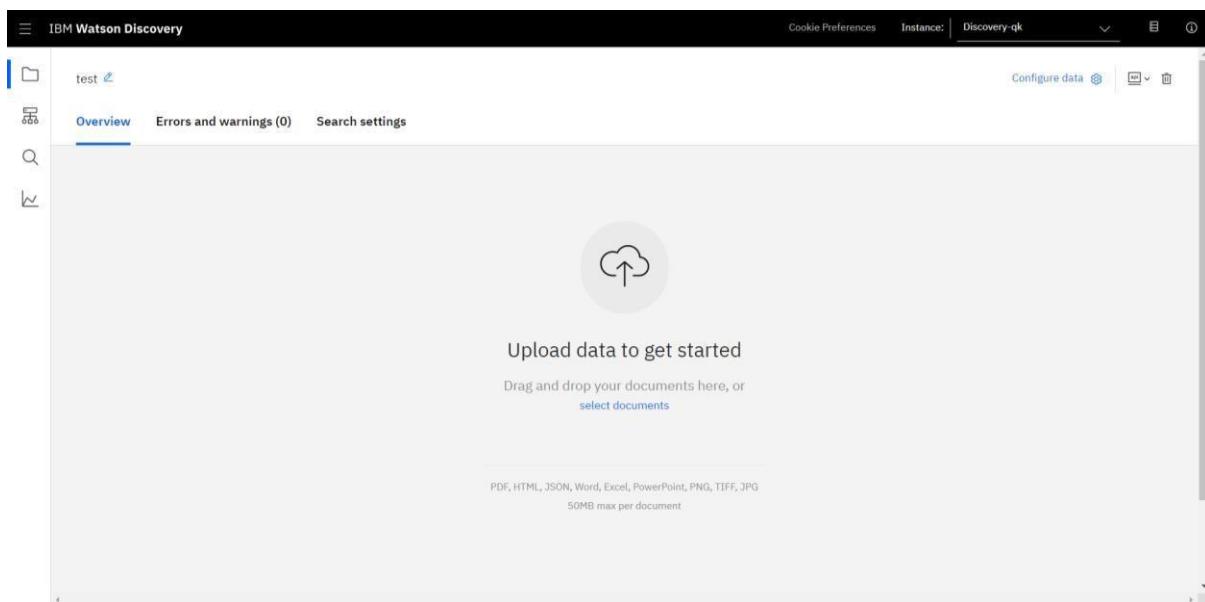


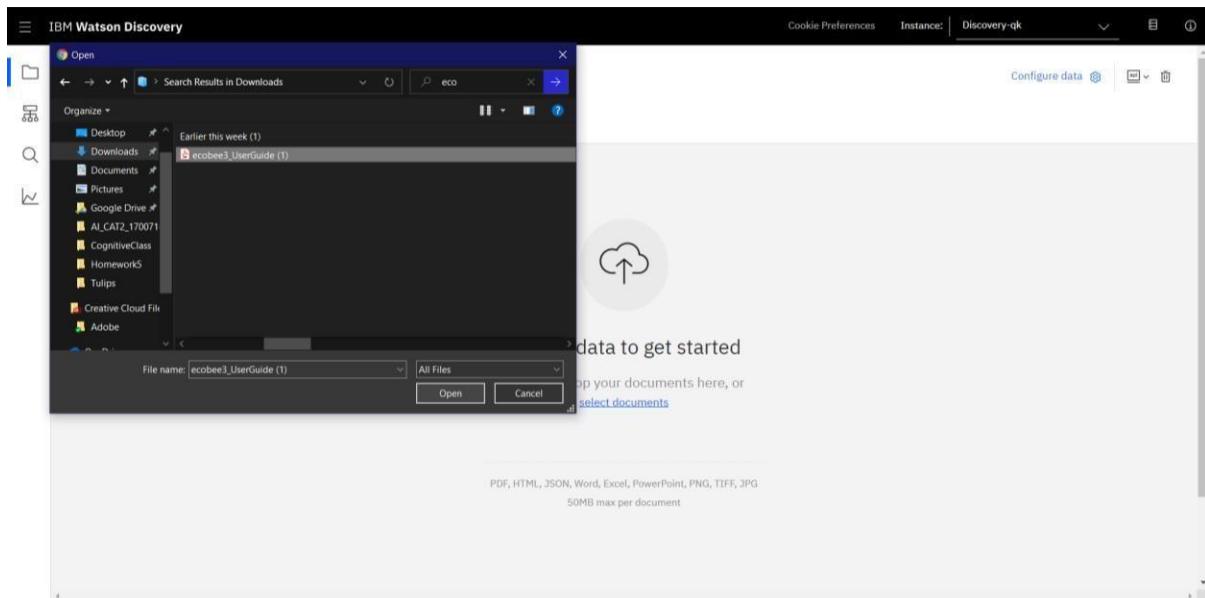
The screenshot shows the "IBM Watson Discovery" interface under the "Manage data" tab. It features a sidebar with icons for "Manage data", "Create new data collection", "Upload your own data", and "Connect a data source". The main area displays two data collections: "Watson Discovery News" (PRE-ENRICHED DATA) and "owners-manual" (PRIVATE DATA). Both collections show their indexing status: "Indexed on 5/12/2020".

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.

The screenshot shows the IBM Watson Discovery interface. At the top, there are tabs for 'Overview', 'Errors and warnings (1)', and 'Search settings'. Below this, it displays '1 document' with '0 documents failed'. It shows the creation date as 5/16/2020 11:29:11 am EDT and the last update as 5/16/2020 11:29:11 am EDT. There is a 'Upload documents' button. On the left, there are icons for file, search, and refresh. In the center, it says 'Identified 1 field from your data' (text) and 'Added 4 enrichments to your data' (Entity Extraction, Sentiment Analysis, Concept Tagging, Category Classification). To the right, it says 'Now you're ready to query!' with three examples: 'Most common entity types and their top entities' (Run), 'Documents that contain Air conditioner, but not Energy recovery' (Run), and 'Top people related to /business and industrial/energy' (Run). A red box highlights the 'Build your own query →' button.

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.

The screenshot shows the 'Build queries' page. It has a sidebar with a folder icon and a search icon. The main area has a search bar with the query 'how do i turn on the heater'. Below the search bar are buttons for 'Use natural language' and 'Use the Discovery Query Language'. There are also buttons for '+ Include analysis of your results' and '+ Filter which documents you query'. At the bottom, there are 'More options', 'Run query', and 'Close' buttons. To the right, there is a 'Summary' tab and a 'JSON' tab. The 'Summary' tab shows a URL: 'Query URL https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/4'. The 'JSON' tab shows a single result: 'ecobee3\_UserGuide (1).pdf' with a sentiment of 'positive'. A red box highlights the 'Run query' button.

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)

IBM Watson Discovery

Overview Errors and warnings (1) Search settings

1 document

0 documents failed View details

Created on 5/16/2020 11:29:11 am EDT Last updated 5/16/2020 11:29:11 am EDT

Upload documents

Identified 1 field from your data  
text

Need to identify more fields? Add fields

Added 4 enrichments to your data

- Entity Extraction: 104 °F (1), 20 min (1), 20 minutes (1), 20 °C (1), 24-hour (1)
- Sentiment Analysis: 100% positive, 0% neutral, 0% negative
- Concept Tagging: Air conditioner (1), Energy recovery (1), Geothermal heat pump (1)
- Category Classification: business and industr... energy

Now you're ready to query!

- Documents about 104 °F as a Quantity with a very negative sentiment
- Top people related to /business and industrial/energy
- Most common entity types and their top entities

5 enrichments available. Add enrichments

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

IBM Watson Discovery

Identify fields Manage fields Enrich fields

ecobee3\_UserGuide (1).pdf 1/41

Field labels

Identify document elements using the labels below.

+ Create new Upgrade

- answer
- author
- footer
- header
- question
- subtitle
- table\_of\_contents
- text
- title
- image Upgrade
- table

Viewing: Live predictions of latest ML-model.

Submit page

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 text.

Click [5] to submit the page to Discovery.

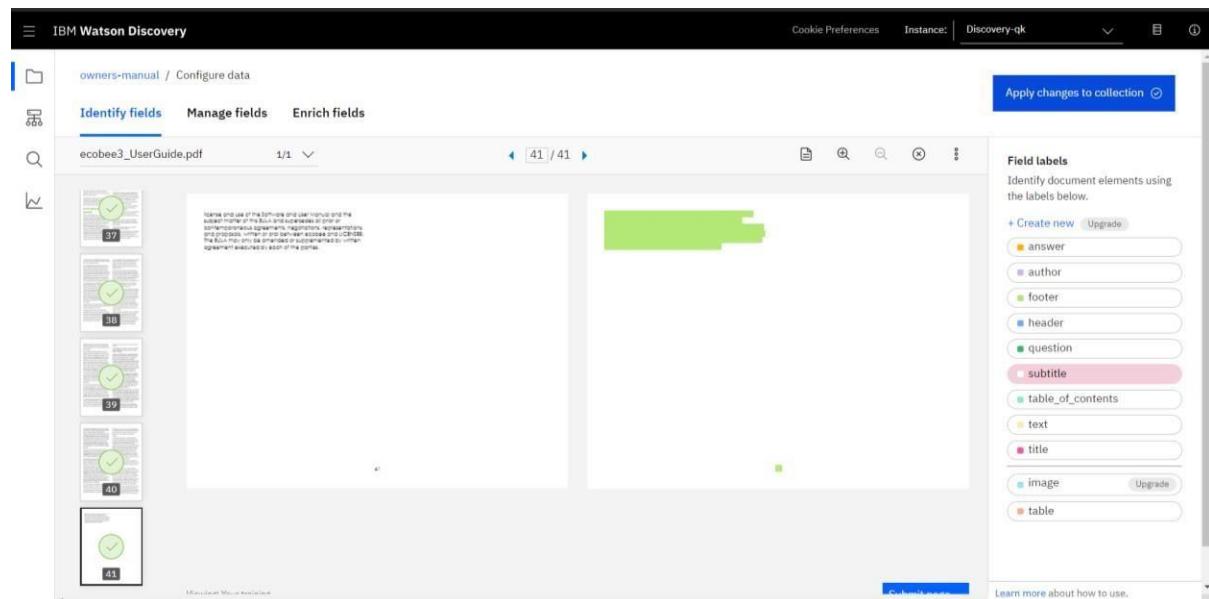
As you go through the annotations one page at a time, Discovery is learning and should start automatically updating the upcoming pages. Once you get to a page that is already correctly annotated, you can stop, or simply click Submit [5] to acknowledge it is correct. The more pages you annotate, the better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



The screenshot shows the IBM Watson Discovery interface. On the left, there's a navigation bar with a folder icon, the text 'owners-manual / Configure data', and three tabs: 'Identify fields' (which is selected), 'Manage fields', and 'Enrich fields'. Below this is a search bar with the text 'ecobee3\_UserGuide.pdf' and a dropdown showing '1/1'. The main area displays a list of 41 pages from the document, each with a small thumbnail and a green checkmark. To the right, there's a 'Field labels' panel with a list of categories: answer, author, footer, header, question, subtitle, table\_of\_contents, text, title, image, and table. The 'subtitle' category is highlighted with a pink background. At the bottom right of the interface, there's a blue button labeled 'Apply changes to collection' with a circular arrow icon. The entire interface has a clean, modern design with a white background and light gray borders.

Click the **Apply changes to collection** [5]

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

The screenshot shows the 'Manage fields' tab in the IBM Watson Discovery interface. On the left, there's a list of fields with toggle switches: 'answer' (off), 'author' (off), 'footer' (off), 'header' (off), 'image' (off), 'question' (off), 'subtitle' (on, highlighted with a red '1'), 'table' (off), 'table\_of\_contents' (off), 'text' (on, highlighted with a red '1'), and 'title' (off). In the center, there's a section about splitting documents by field, with a dropdown menu set to 'subtitle' (highlighted with a red '2'). At the top right, there's a large red '3' over the 'Apply changes to collection' button.

[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

The screenshot shows the 'Overview' tab in the IBM Watson Discovery interface. It displays basic stats: 133 documents, 0 failed documents, and creation and update times. Below, it shows identified fields (author, footer, subtitle, table\_of\_contents, text, title) and added enrichments (Entity Extraction, Sentiment Analysis, Concept Tagging, Category Classification). A red box highlights the 'Run' button next to the 'Top people related to technology and computing/operating systems' section. The bottom right shows three more query cards with 'Run' buttons.

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

The screenshot shows the IBM Watson Discovery interface. On the left, there's a sidebar with a folder icon labeled "owners-manual / Build queries". Below it is a search bar with placeholder text "Build a query using one or more of these components. Learn more." and a "Use a sample query" button. The main area has a search bar with the query "how do i turn on the heater?". Below the search bar are several options: "+ Include analysis of your results", "+ Filter which documents you query", and a "More options" dropdown. At the bottom are "Run query" and "Close" buttons. To the right, there's a "Summary" tab selected, showing a "Query URL" field with the value "https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/4". Below the URL is a "Passages" section with some descriptive text and configuration options. A "Train Watson to improve results" button is at the top right of the summary panel.

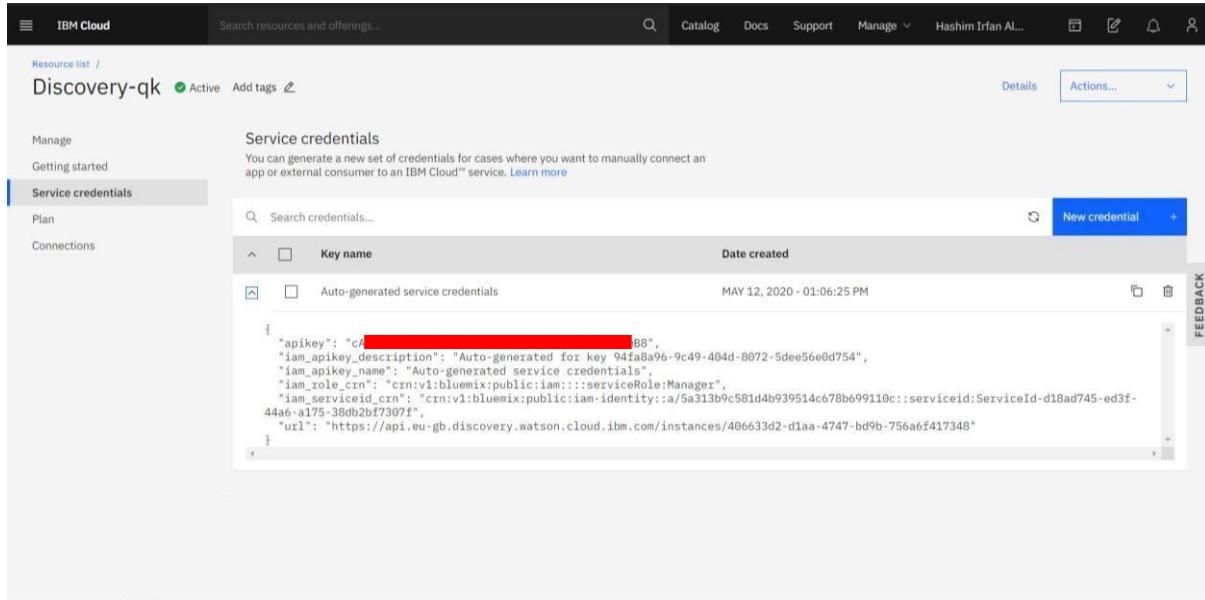
In upcoming steps, you will need to provide some credentials to access your Discovery collection so to Store credentials for future use follow the below steps.

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

The screenshot shows the IBM Watson Discovery interface with the "Collection ID" and "Environment ID" fields highlighted. The "Collection ID" field contains "3bc [REDACTED] 35f" and the "Environment ID" field contains "aa5 [REDACTED] b20". The interface also displays other collection details like "Created on" and "Last updated". Below these details, there are sections for "Identified 6 fields from your data" (author, footer, subtitle, table\_of\_contents, text, title) and "Added 4 enrichments to your data" (Entity Extraction, Sentiment Analysis, Concept Tagging, Category Classification). A "Configure data" button is visible at the top right of the collection details 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.

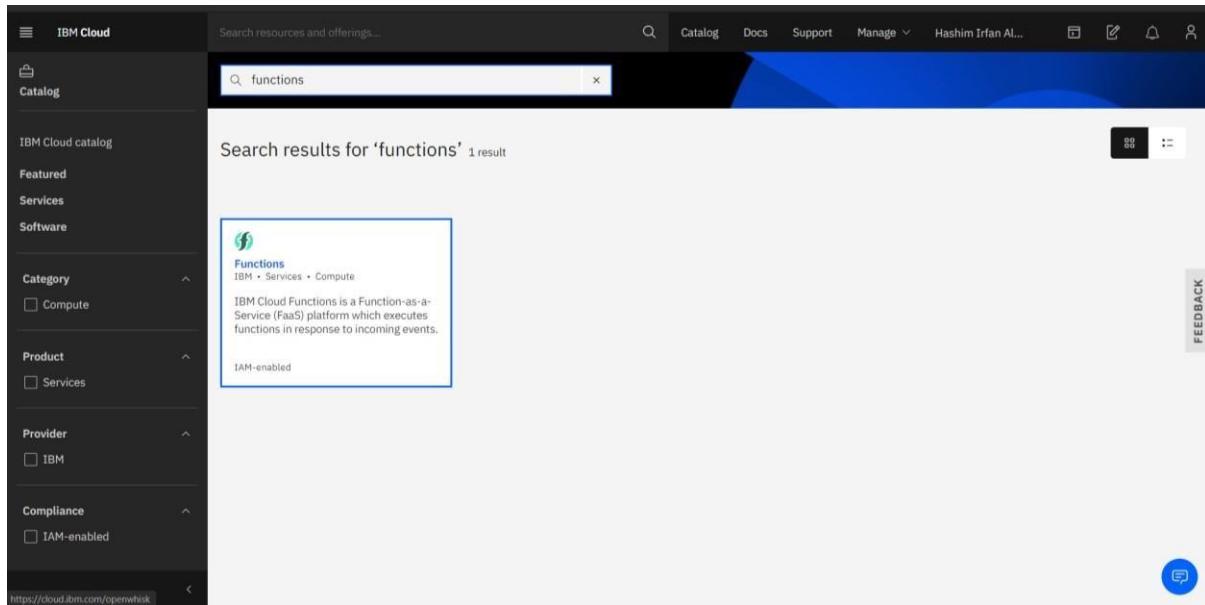


The screenshot shows the IBM Cloud Watson Discovery Resource List screen. The left sidebar has 'Discovery-qk' selected under 'Service credentials'. The main area displays 'Service credentials' with a note about generating new credentials for manual connection. A table lists one entry: 'Auto-generated service credentials' created on MAY 12, 2020 - 01:06:25 PM. The table includes columns for 'Key name' and 'Date created'. The details row shows a JSON object with fields like 'apikey', 'iam\_apikey\_crn', 'iam\_role\_crn', 'iam\_serviceid\_crn', and 'url'.

### 3. Create IBM Cloud Functions action

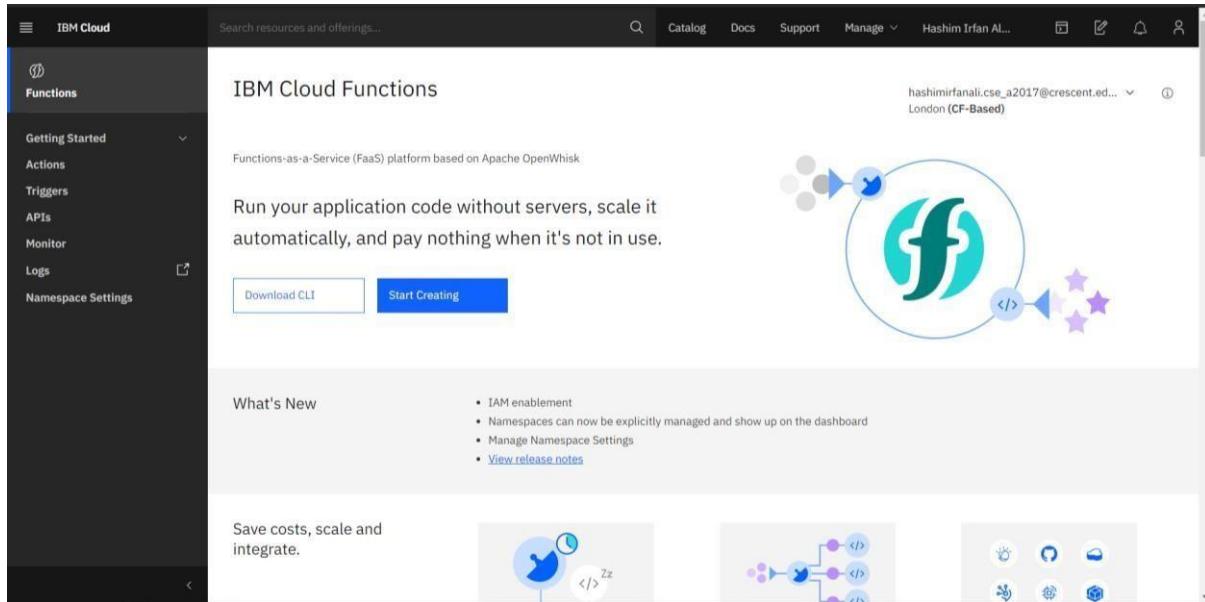
Now let's create the web action that will make queries against our Discovery collection.

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



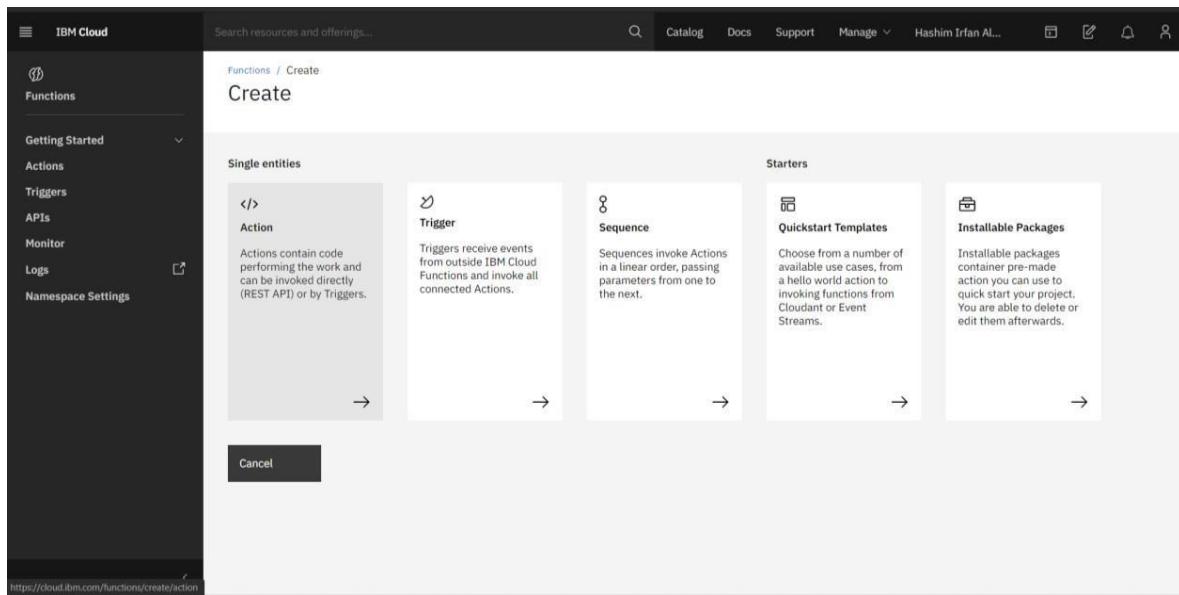
The screenshot shows the IBM Cloud Catalog search results for 'functions'. The search bar contains 'functions'. One result is shown: 'Functions' by IBM, categorized under Services > Compute. It is described as a Function-as-a-Service (FaaS) platform. The result is highlighted with a blue border.

## Select the Functions Card.



The screenshot shows the IBM Cloud Functions main panel. On the left, there's a sidebar with a 'Functions' card highlighted. The main content area has a heading 'IBM Cloud Functions' and a sub-heading 'Functions-as-a-Service (FaaS) platform based on Apache OpenWhisk'. It features a large 'Start Creating' button. To the right is a circular icon with a stylized 'ff' logo and some network nodes. Below the main area, there's a 'What's New' section with a bulleted list and a 'View release notes' link. At the bottom, there are three small icons representing different function types: Action, Trigger, and Sequence.

From the Functions main panel, click on Start Creating.



The screenshot shows the 'Create' screen for IBM Cloud Functions. The left sidebar still has the 'Functions' card selected. The main area has a heading 'Create' and a sub-heading 'Functions / Create'. It displays four categories under 'Single entities': 'Action', 'Trigger', 'Sequence', and 'Starters'. Each category has a brief description and a right-pointing arrow indicating it can be selected. At the bottom left is a 'Cancel' button.

Here, select Actions.

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

Action Name

Enclosing Package

(Default Package)

Create Package

Runtime

Node.js 10

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

Previous Cancel Create

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.

Code

Parameters

Runtime

Endpoints

Connected Triggers

Enclosing Sequences

Logs

Namespace: Namespace-W2S(Dallas)

Invoke with parameters

2

1

```
1 /**
2 * @param {object} params
3 * @param {string} params.iam_apikey
4 * @param {string} params.url
5 * @param {string} params.username
6 * @param {string} params.password
7 * @param {string} params.environment_id
8 * @param {string} params.collection_id
9 * @param {string} params.configuration_id
10 * @param {string} params.input
11 * @param {string} params.output
12 *
13 * @return {object}
14 *
15 */
16
17 const assert = require('assert');
18 const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');
19
20 /**
21 * main() will be run when you invoke this action
22 *
23 * @param Cloud Functions actions accept a single parameter, which must be a JSON object.
24 * @param {Object} params
25 * @return The output of this action, which must be a JSON object.
26 */
27
28 /**
29 * function main(params) {
30 *   return new Promise(function (resolve, reject) {
31 *     let discovery;
32 *     let configuration;
33 *     if (params.iam_apikey){
```

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

The screenshot shows the IBM Cloud Functions Actions interface. The top navigation bar includes 'IBM Cloud', 'Catalog', 'Docs', 'Support', 'Manage', and a user profile. Below the navigation is a search bar 'Search resources and offerings...'. The main area shows a 'disco\_action' entry under 'Functions / Actions / disco\_action'. The 'Web Action' button is highlighted. On the left, a sidebar lists 'Code', 'Parameters' (which is bolded and has a red '1' over it), 'Runtime', 'Endpoints', 'Connected Triggers', 'Enclosing Sequences', and 'Logs'. The 'Parameters' tab is active, displaying four parameters: 'url' (value: "https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/406633d2-d1aa-474"), 'environment\_id' (value: "d1aa-4745"), 'collection\_id' (value: "31-5f"), and 'iam\_apikey' (value: "CA-88"). An 'Add Parameter' button is at the top right of the parameters table.

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

The screenshot shows the IBM Cloud Functions Actions interface. The top navigation bar and search bar are identical to the previous screenshot. The main area shows the 'disco\_action' entry. The 'Code' tab is selected, displaying a Node.js 10 file named 'Node.js 10' with the following code:

```
1 /**
2  * @param {object} params
3  * @param {string} params.iam_apikey
4  * @param {string} params.environment_id
5  * @param {string} params.username
6  * @param {string} params.password
7  * @param {string} params.collection_id
8  * @param {string} params.configuration_id
9  * @param {string} params.input
10 */
11
12 /**
13  * @return {object}
14 */
15
16
17 const assert = require('assert');
18 const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');
19
20 /**
21  * main() will be run when you invoke this action
22  *
23  * @param Cloud Functions actions accept a single parameter, which must be a JSON object.
24  * @param The output of this action, which must be a JSON object.
25  */
26
27 /**
28  * Function main(params) {
29  *   return new Promise(function (resolve, reject) {
30  *     let discovery;
31  *     if (params.iam_apikey){
```

Below the code editor is an 'Invoke with parameters' button and an 'Invoke' button with a play icon. To the right, the 'Activations' panel shows a single activation entry: 'disco\_action' with an activation ID of '8a-5f41', completed in 1450 ms on 5/16/2020, 22:24:23. The 'Results' section displays the JSON response from the Discovery service, including matching results, categories, and concepts.

Now click on endpoints[1] tab

The screenshot shows the IBM Cloud Functions dashboard for an action named 'disco\_action'. The 'Endpoints' tab is selected, indicated by a blue highlight and the number '1'. Under the 'Web Action' section, there is a checked checkbox labeled 'Enable as Web Action' (indicated by '2'). This checkbox allows your Cloud Functions actions to handle HTTP events. Below it is an unchecked checkbox for 'Raw HTTP handling'. The 'HTTP Method' dropdown is set to 'ANY' and the 'Auth' dropdown is set to 'Public'. The resulting URL is [https://us-south.functions.cloud.ibm.com/api/v1/web/728c6dcb-3bf4-4108-b946-4e5cc6fc84d6/default/disco\\_action](https://us-south.functions.cloud.ibm.com/api/v1/web/728c6dcb-3bf4-4108-b946-4e5cc6fc84d6/default/disco_action), which is highlighted with a red box labeled '3'. Below this is the 'REST API' section, which shows a POST method with IAM TOKEN authentication and the same URL. At the bottom, there is a 'CURL' section with a provided command, also highlighted with a red box labeled '4'.

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, open Watson Assistant service.

Watson Assistant-61

click to open Watson Assistant service.

The screenshot shows the IBM Cloud Resource list interface. At the top, there's a navigation bar with 'IBM Cloud' and a search bar. Below it, the 'Resource list' section displays a single service entry: 'Watson Assistant-61' (Active), with options to 'Add tags' and 'Actions...'. On the left, a sidebar titled 'Manage' offers links to 'Service credentials', 'Plan', and 'Connections'. The main content area has three main sections: 'Start by launching the tool' (with a 'Launch Watson Assistant' button), 'Credentials' (showing an API key and URL), and 'Plan' (set to 'Lite', with an 'Upgrade' button). A 'FEEDBACK' button is located on the right side of the page.

Click on **Launch Watson Assistant** to launch Watson Assistant.

The screenshot shows the IBM Watson Assistant service interface. At the top, there are buttons for 'IBM Watson Assistant Lite' and 'Upgrade'. The main area is titled 'Assistants' and contains two cards: 'Customer Care' (Skills: 1, Customer Care Sample Skill; Integrations: 1) and 'My first assistant' (Skills: 1, My first skill; Integrations: 0). A blue message icon is at the bottom right. A red box highlights the 'Create assistant' button on the left.

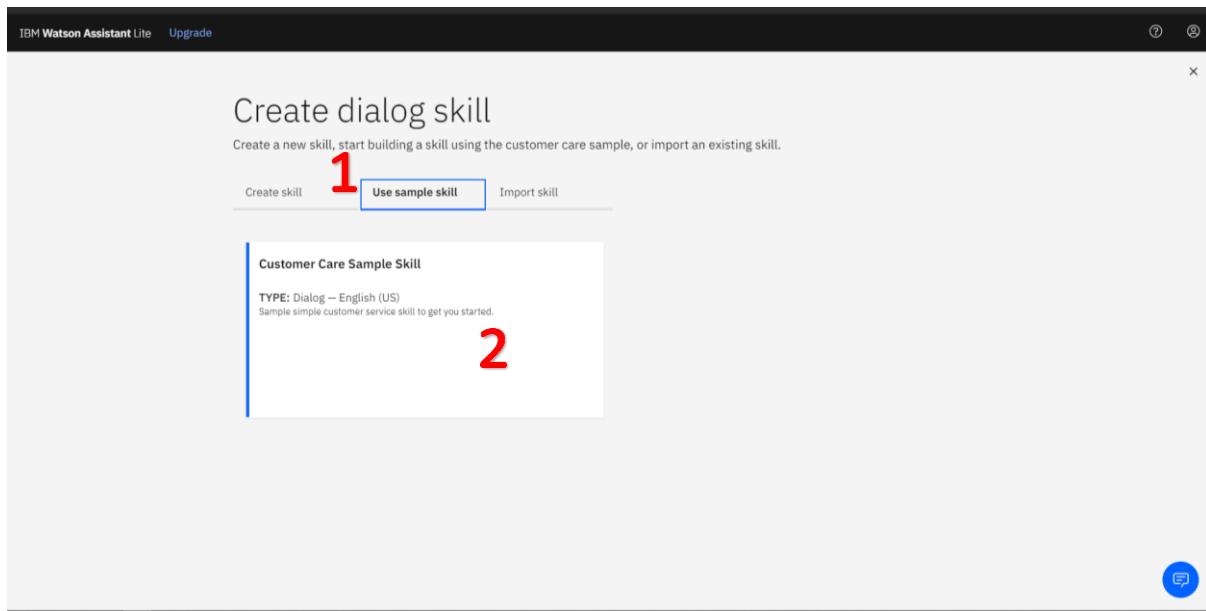
Click on the skills tab

The screenshot shows the 'Skills' section of the IBM Watson Assistant Lite interface. At the top, there's a header bar with the IBM Watson Assistant Lite logo and an 'Upgrade' link. Below the header, the word 'Skills' is displayed next to a small icon. A descriptive text states: 'Skills contain the training to respond to your customer queries. Add skills to your assistant and then deploy to your channels.' A prominent blue 'Create skill' button is centered, with a red box drawn around it to indicate where the user should click. Below the button, two skill cards are listed: 'Customer Care Sample Skill' and 'My first skill'. Each card provides details like type (Dialog - English (US)), creation and update dates, and linked assistants.

Click Create Skill

The screenshot shows the 'Create a skill' wizard. The title 'Create a skill' is at the top, followed by a sub-instruction: 'Skills can be combined to improve your assistant's capabilities. Learn more'. Below this, a section titled 'Select skill type' contains two options: 'Dialog skill' and 'Search skill'. The 'Dialog skill' card is highlighted with a red box. It contains a brief description: 'Dialog flows are designed to address customer issues. Dialog skills expose the mechanics involved in natural language processing and responding appropriately to customers. Learn more'. To the right of the dialog skill card is a 'Try Plus plan' button. At the bottom of the screen, a 'Next' button is visible.

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.

The screenshot shows the 'Customer Care Sample Skill' panel with the 'Intents' tab selected. On the left, a sidebar lists 'Entities', 'Dialog', 'Options', 'Analytics', 'Versions', and 'Content Catalog'. The main area displays a table of intents. A red '1' highlights the 'Create intent' button at the top right of the table header. The table columns are: Intent ID (checkbox), Intent Name, Description, Modified Date, and Examples Count. The examples count column is sorted in descending order.

	Intent Name	Description	Modified	Examples
#Cancel	Cancel the current request	15 days ago	7	
#Customer_Care_Appointments	Schedule or manage an in-store appointment.	15 days ago	20	
#Customer_Care_Store_Hours	Find business hours.	15 days ago	48	
#Customer_Care_Store_Location	Locate a physical store location or an address.	15 days ago	25	
#General_Connect_to_Agent	Request a human agent.	15 days ago	47	
#General_Greetings	Greetings	15 days ago	30	
#Goodbye	Good byes	15 days ago	6	
#Help	Ask for help	15 days ago	8	
#Product_Information		4 days ago	3	

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.

The screenshot shows the IBM Watson Assistant Lite interface. At the top, there's a header with 'IBM Watson Assistant Lite' and 'Upgrade' buttons. Below the header, the title '#Product\_Information' is displayed. There are three input fields: 'Intent name' containing '#Product\_Information', 'Description (optional)' with the placeholder 'Add a description to this intent', and 'User example' with the placeholder 'Type a user example here, e.g. I want to pay my credit card bill'. Below these fields is a 'Show recommendations' button. A list of user examples is shown, each with a checkbox, a timestamp ('4 days ago'), and a 'Delete' icon. The examples are: 'How do i access the settings', 'How do i set the time', and 'How do i cancel my order'. At the bottom right, there are buttons for 'Annotate entities' and 'What's this?'. The footer shows 'Showing 1–3 of 3 examples' and navigation icons.

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

The screenshot shows the 'Customer Care Sample Skill' dialog in the IBM Watson Assistant Lite interface. The 'Dialog' tab is selected in the sidebar. A tree view of dialog nodes is displayed, starting with 'Opening' (with sub-node 'welcome') and 'What are your hours?' (with sub-node '#Customer\_Care\_Store\_Hours'). Other nodes include 'Where are you located?', 'I want to make an appointment', 'General\_Greetings', and '#GrowthHub'. Each node has a 'More options' menu icon. In the top right, there are buttons for 'Save new version', 'Try it', and a search bar. The 'Add node' button is highlighted in blue.

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

The screenshot shows the IBM Watson Assistant interface. On the left, there's a sidebar with options like Intents, Entities, Dialog (which is selected), Options, Analytics, Versions, and Content Catalog. The main area displays a 'Customer Care Sample Skill' dialog flow. A node labeled '#General\_Greetings' is at the top. Below it are nodes for '#Goodbye', '#Thanks', and 'Please transfer me to an agent'. A node labeled 'What can I do' follows. At the bottom of the list is a node labeled 'Ask product information' with the intent '#Product\_Information' assigned to it. This last node is highlighted with a blue box and has a red number '1' above it. To its right, under the heading 'If assistant recognizes', is another node labeled '#Product\_Information' with a red number '2' above it. Further down, there's a section for 'Assistant responds' with a text input field and a 'Text' dropdown.

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.

Customer Care Sample Skill

Save new version Try it

**Webhooks**

A webhook is a mechanism that allows you to call out to an external program based on events in your dialog.

**Webhook setup**

Specify the request URL for an external API you want to be able to invoke from dialog nodes. Watson will call this URL when configured to do so from a dialog node. [Learn more](#)

URL: <https://us-south.functions.cloud.ibm.com/api/v1/web/728c6dcb-3bf4-410e...>

**Headers**

Add HTTP headers for authorization or any other parameters required for invoking the webhook.

Header name	Header value
Add header	Add authorization

**Next step**

To trigger this webhook from an individual dialog node, enable webhooks from the Customize page of the node. [Go to dialog](#)

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

Customer Care Sample Skill

Save new version Try it

Customize "Ask product information"

Enable this to gather the information your bot needs to respond to a user within a single node.

Prompt for everything

Enable this to ask for multiple pieces of information in a single prompt, so your user can provide them all at once and not be prompted for them one at a time.

**Webhooks**  On

Enable this setting to send a POST request from this dialog node to the webhook URL. The URL and headers are defined in the Webhooks settings of the Options tab. After you enable this setting, the Multiple conditional responses setting is enabled automatically to support adding a response to show when the request is successful and another response to show if the request fails. [Learn more](#)

**Webhook URL** Your webhook URL is configured.

Cancel Apply

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.

The screenshot shows the IBM Watson Assistant Lite interface. On the left, the sidebar has 'Dialog' selected. The main area shows a list of nodes:

- #Goodbye
- #Thanks
- Please transfer me to an agent
  - #General\_Connect\_to\_Agent
- What can I do
  - #Help
- Ask product information
  - #Product\_Information
- anything\_else

The 'Ask product information' node is highlighted with a blue border. To its right, the configuration panel shows:

- Node name:** Ask product information
- Parameters:**

Key	Value
input	"<?input.text?>"
- Return variable:** webhook\_result\_1
- Webhook URL:** Your webhook URL is configured. Options

Red numbers 1 and 2 are overlaid on the 'Return variable' and 'input' fields respectively.

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:

The screenshot shows the same configuration as before, but now includes optional responses under 'Assistant responds'.

**If assistant recognizes:**

- 1 \$webhook\_result\_1
- 2 anything\_else

**Respond with:**

- \$webhook\_result\_1
- Try again later

A red box highlights the 'Try again later' response row.

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

The screenshot shows the IBM Watson Assistant Lite interface. On the left, there's a sidebar with options: Intents, Entities, Dialog (which is selected), Options, Analytics, Versions, and Content Catalog. The main area has a title "Customer Care Sample Skill". It contains a "Ask product information" node under "Assistant responds". This node has two response options: 1. \$webhook\_result\_1 (Respond with \$webhook\_result\_1) and 2. anything\_else (Respond with Try again later). A tooltip says "Webhook URL Your webhook URL is configured." Below this is a "Then assistant should" section. To the right, there's a "Try it out" panel showing a conversation: "Hi", "#General\_Greetings", "Hello. Good evening", "#Product\_Information", "How to turn on heater?", and a JSON response object. At the bottom right of the interface is a blue button labeled "Enter something to test your assistant".

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

This screenshot is similar to the previous one but includes a "Context variables" sidebar on the right. It lists variables: \$ (with a placeholder "Enter variable name"), \$timezone ("Asia/Calcutta"), \$no\_reservation ("true"), and \$webhook\_result\_1 (with a partial JSON value: {"matching\_results":22,"passages": [{"document\_id": "3a5feef70dccc9d70e2b94d2c15e2d1\_11","end\_offset":240,"field": "text","passage\_score":0,"passage\_text": "I"}]). There are buttons for "Remove all context variables" and a blue "Enter something to test your assistant" button.

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 screenshot shows the 'Skills' section of the IBM Watson Assistant Lite interface. There are two skills listed: 'Customer Care Sample Skill' and 'My first skill'. The 'Customer Care Sample Skill' card has a context menu open, with the second item, 'View API Details', highlighted. Red numbers 1 and 2 are overlaid on the menu items.

The Skill ID and API Key is to be noted.

The screenshot shows the 'Skill details' page for the 'Customer Care Sample Skill'. It displays the skill name, creation date, and a legacy workspace URL. Below this, under 'Service credentials', it shows the service credentials name and an API key. Both the 'Skill ID' and the 'API key' fields are highlighted with red boxes.

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

The screenshot shows the IBM Cloud Watson Assistant resource list. A red box labeled '1' highlights the 'Service credentials' tab under the 'Manage' section. The main area displays a table of service credentials, with one row selected. A red box highlights the 'url' field, which contains the API endpoint: `https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/f79fec49-933c-46e0-b562-1d2bc9f497b5`.

## 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].

The screenshot shows the IBM Cloud Catalog search results for 'node-red'. A red box labeled '1' is over the search bar. A red box labeled '2' highlights the 'Node-RED App' tile, which is described as a software solution for building Node-RED apps on IBM Cloud.

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

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

The screenshot shows the IBM Cloud interface for the Node-RED Starter Kit. At the top, there's a navigation bar with links for Catalog, Docs, Support, Manage, and a user profile. Below the navigation is a search bar labeled "Search resources and offerings...". The main content area has a title "Node-RED" with a "Create" button highlighted by a red number "1". On the left, there's a sidebar with "About" and "Create" tabs. The "About" tab is selected, showing details like Author (IBM), Updated (2/11/2020), and Type (Starter kit). It also links to Source code (GitHub) and Helpful links (Tutorial). The "Create" tab shows an "Overview" section with a brief description of the starter kit, a "What's included?" section listing Cloudant services, and a "Get started" button. A sidebar on the right includes "ASK A QUESTION" and "FEEDBACK" buttons.

Click on Create [1].

The screenshot shows the "App details" page for creating a new application. The "Create" tab is selected. The "App name" field contains "Node RED KVHBI", with a red number "1" overlaid. Other fields include "Resource group" (Default), "Tags" (empty), and "Platform" (Node.js). A "Service details" section is partially visible at the bottom.

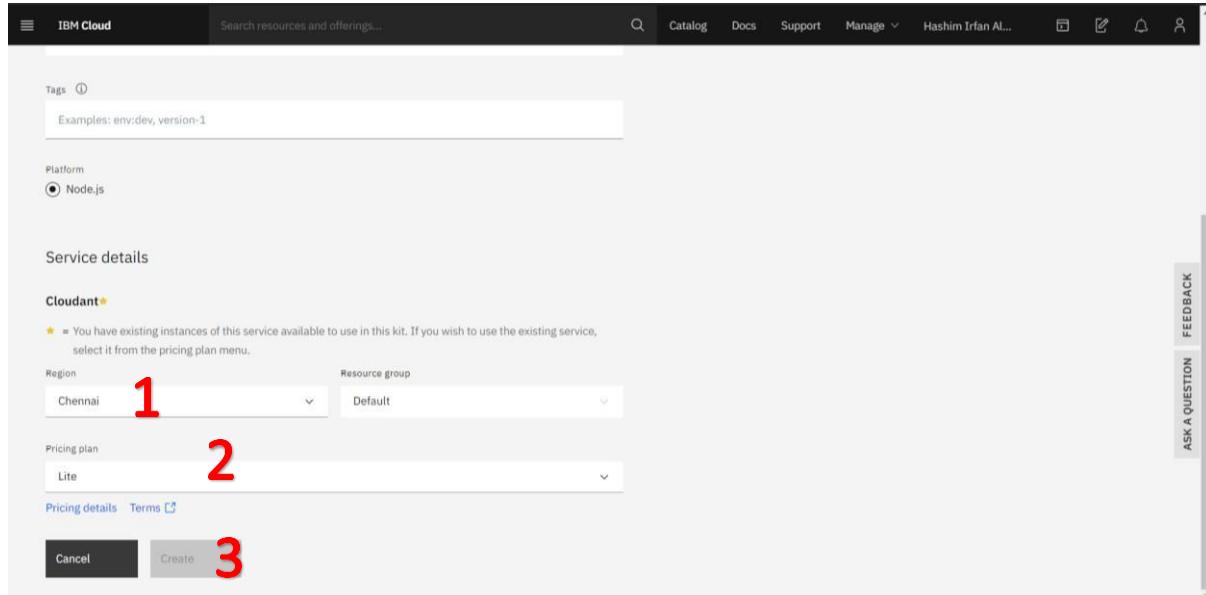
Now you need to configure the Node-RED Starter application.

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.

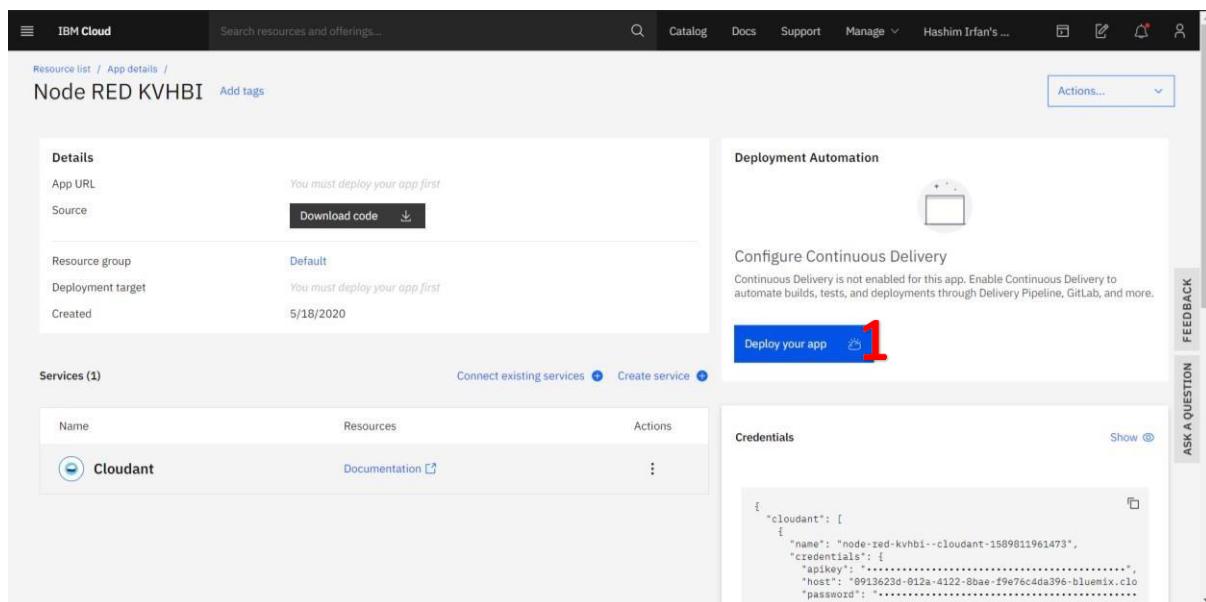
The Node-RED Starter application requires an instance of the Cloudant database service to store your application flow configuration. To do this,

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.

The screenshot shows the 'Deployment Automation' step of the deployment wizard. It includes a 'Deployment target' section with 'Cloud Foundry' selected and a detailed description of its benefits. Below it is an 'IBM Cloud API key' input field, which is currently empty and has a placeholder message: 'The value is required.' To the right of this field is a blue 'New' button, which is highlighted with a large red number '1'. To the right of the input field, there is a sidebar with steps and a 'Getting started with apps' section.

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].

The screenshot shows the deployment configuration step. It includes a 'Number of instances' dropdown set to '1', a 'Memory allocation per instance' slider set to '256 MB' (highlighted with a red number '1'), a 'Region' dropdown set to 'London' (highlighted with a red number '2'), and a 'Host' and 'Domain' dropdowns both set to their respective values. At the bottom are 'Cancel' and 'Next' buttons, with the 'Next' button highlighted with a red number '3'.

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

The screenshot shows the 'Configure the DevOps toolchain' step. A red '1' is placed over the 'Region' dropdown menu, which is set to 'Dallas'. A red '2' is placed over the blue 'Create' button.

This will take you back to the application details page.

The screenshot shows the application details page. The 'Deployment Automation' section has a note: 'Continuous Delivery is not enabled for this app. Enable Continuous Delivery to automate builds, tests, and deployments through Delivery Pipeline, GitLab, and more.' The 'Credentials' section shows a JSON object:

```
{ "cloudant": [ { "name": "node-red-kvhbi--cloudant-1509011961473", "credentials": { "apikey": "...", "host": "0913623d-012a-4122-8bae-f9e76c4da396-bluemix.clo", "password": "...", "port": 443, "url": "https://node-red-kvhbi--cloudant-1509011961473.ngrok.bluemix.net" } } ] }
```

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.

**Deployment Automation**

- Name: NodeREDKVHBI
- Location: Dallas
- Tool integrations:
- Delivery Pipelines: NodeREDKVHBI In progress
- Last commit by IBM Cloud (1 minute ago)
- Clone from zip

**Credentials**

```
{
  "cloudant": [
    {
      "name": "node-red-kvhbi--cloudant-1589811961473",
      "credentials": [
        "apikey": "...",
        "host": "...0013623d-012a-4122-8bae-f9a76c4da396.ng.bluemix.net"
      ]
    }
  ]
}
```

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.

**Deployment Automation**

- Name: NodeREDKVHBI
- Location: Dallas
- Tool integrations:
- Delivery Pipelines: NodeREDKVHBI Success
- Last commit by IBM Cloud (4 minutes ago)
- Clone from zip

**Credentials**

```
{
  "cloudant": [
    {
      "name": "node-red-kvhbi--cloudant-1589811961473",
      "credentials": [
        "apikey": "...",
        "host": "...0013623d-012a-4122-8bae-f9a76c4da396.ng.bluemix.net"
      ]
    }
  ]
}
```

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].

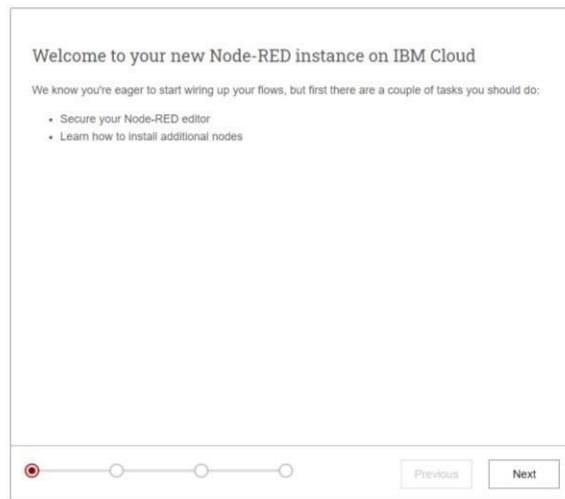
The screenshot shows the IBM Cloud Resource list interface. On the left, there's a sidebar with various service categories like Devices, VPC infrastructure, Clusters, and Cloud Foundry apps. Under Cloud Foundry apps, there is a section for 'Cloud Foundry services' which contains a single entry labeled 'Node RED KVHBI'. This entry is highlighted with a red number '2'. To the right of the sidebar, the main area displays a table with columns: Name, Group, Location, Offering, Status, and Tags. The 'Node RED KVHBI' entry has a status of 'Started' and is associated with the 'SDK for Node.js™' offering. A red number '1' is placed over the first row of the table.

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

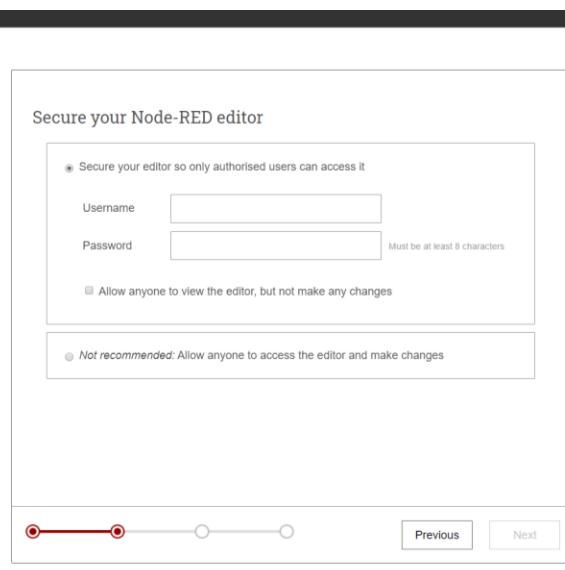
The screenshot shows the detailed view of the 'Node RED KVHBI' application. At the top, it says 'Running' with a green dot, 'Visit App URL' (labeled '2'), and 'Add tags'. Below this, the 'Overview' tab is selected. The 'Instances' section shows 100% health with 1/1 instance running. The 'Runtime' section shows a donut chart with 256 MB total allocation, where 1.75 GB is available (Free) and none is used (Used). The 'Runtime cost' section shows \$0.00 for current charges and an estimated total of \$0.00 for May 1, 2020 - May 31, 2020. The 'Connections' section lists one connection: 'node-red-kvhbi--cloudant-1589811961473-32813'. A red number '1' is placed over the 'Edit' link in the Runtime section.

**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.

### Secure your Node-RED editor

Secure your editor so only authorised users can access it

*Not recommended:* Allow anyone to access the editor and make changes

Your editor will not be secured. Anyone with the URL will be able to access your flows, data and bound services.

Tick this box to confirm you want your editor to be insecure

Previous Next

Tick the box, and click Next.

### Learn how to install additional nodes

Node-RED provides a [huge catalog](#) of extra nodes you can install into the editor.

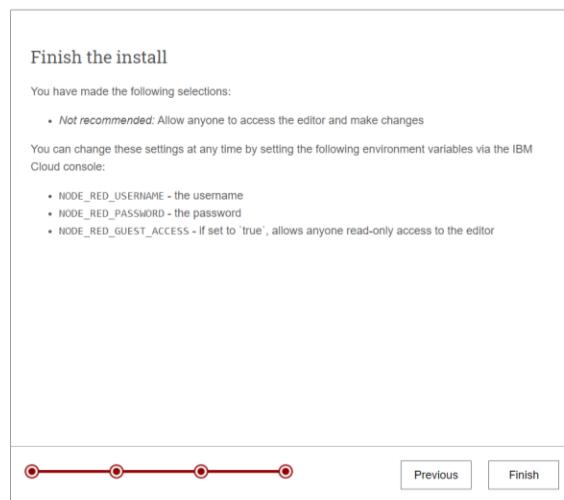
Many of these nodes can be installed directly from the editor's palette manager feature. However that can cause issues due to the limited memory of the default Node-RED starter application.

The [recommended approach](#) is to edit your application's package.json file to include the additional node modules and then redeploy the application. This can be done using the Continuous Delivery feature on the application's IBM Cloud dashboard.

For more information, follow this tutorial on [IBM Developer](#).

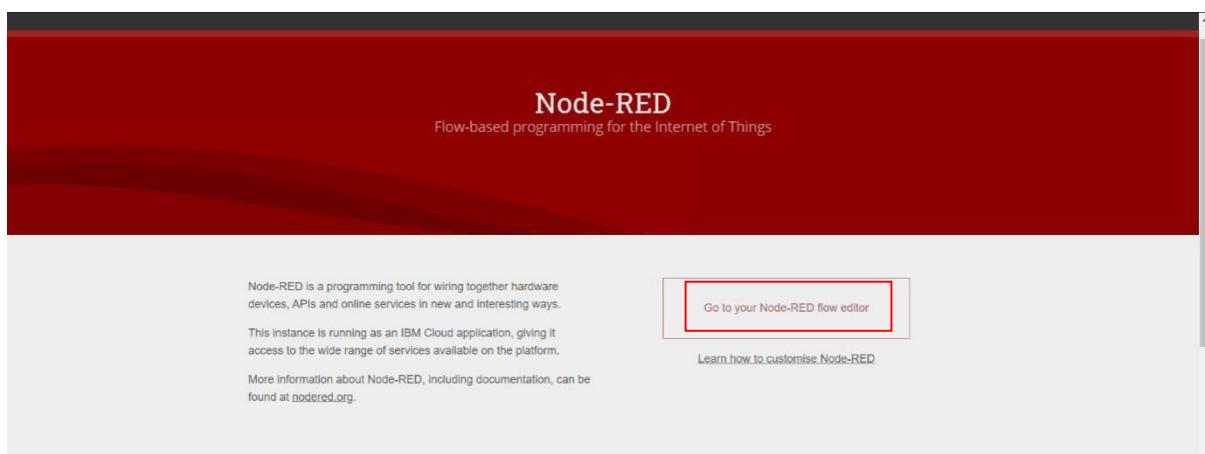
Previous Next

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.



#### Customising your instance of Node-RED

This instance of Node-RED is enough to get you started creating flows.

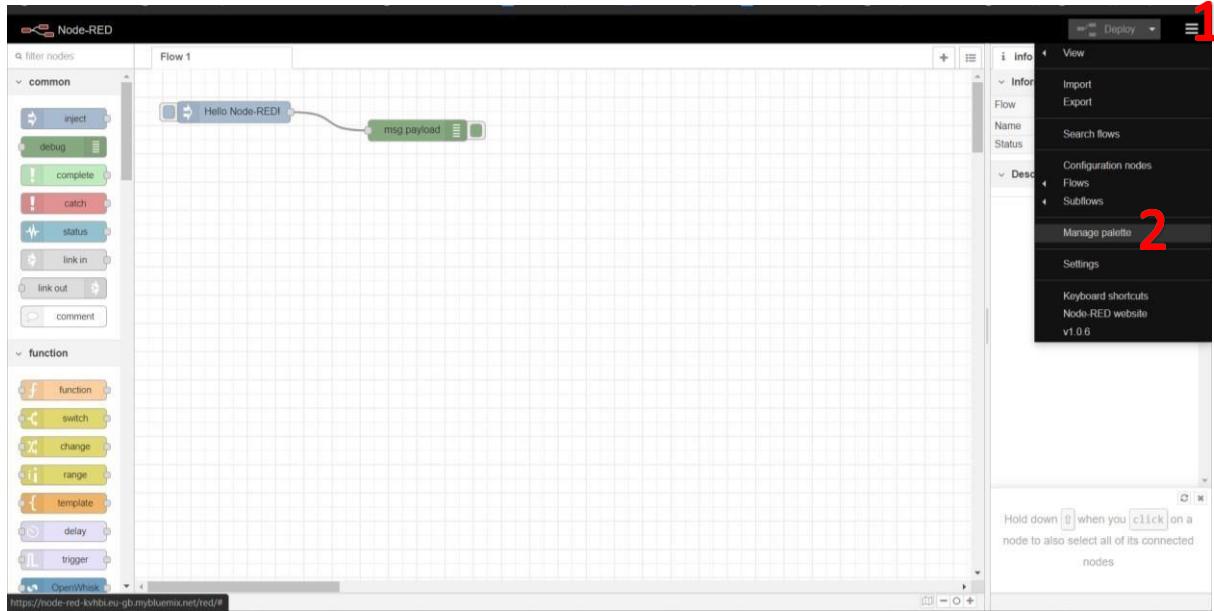
You may want to customise it for your needs, for example replacing this introduction page with your own, adding http authentication to the flow editor or adding new nodes to the palette.



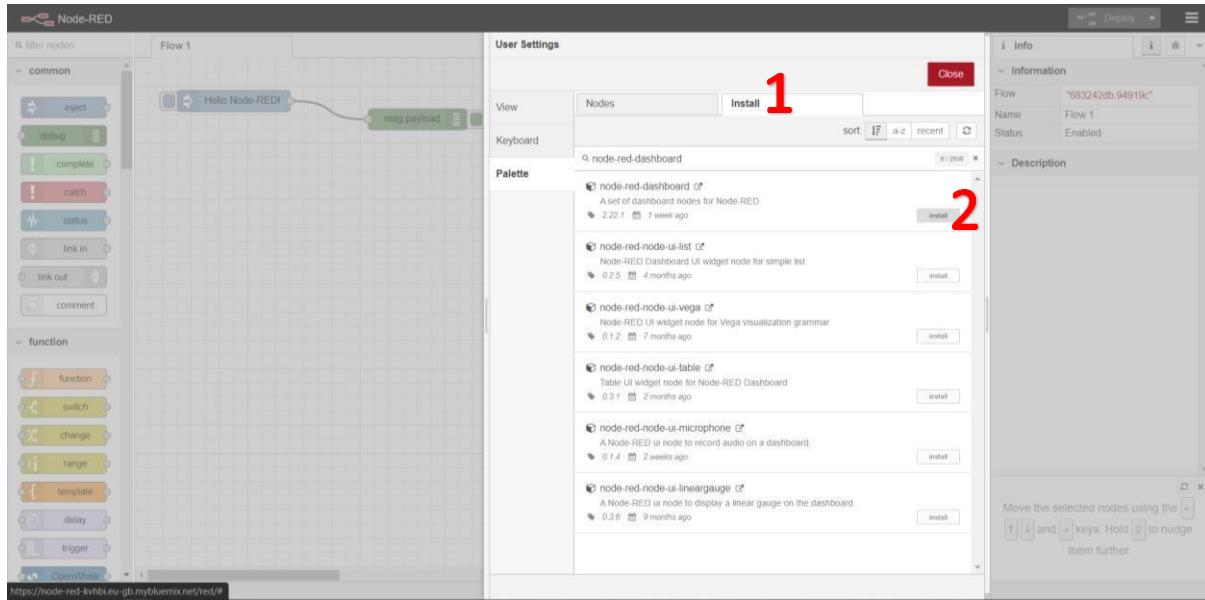
The Node-RED editor opens showing the default flow.

## 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]

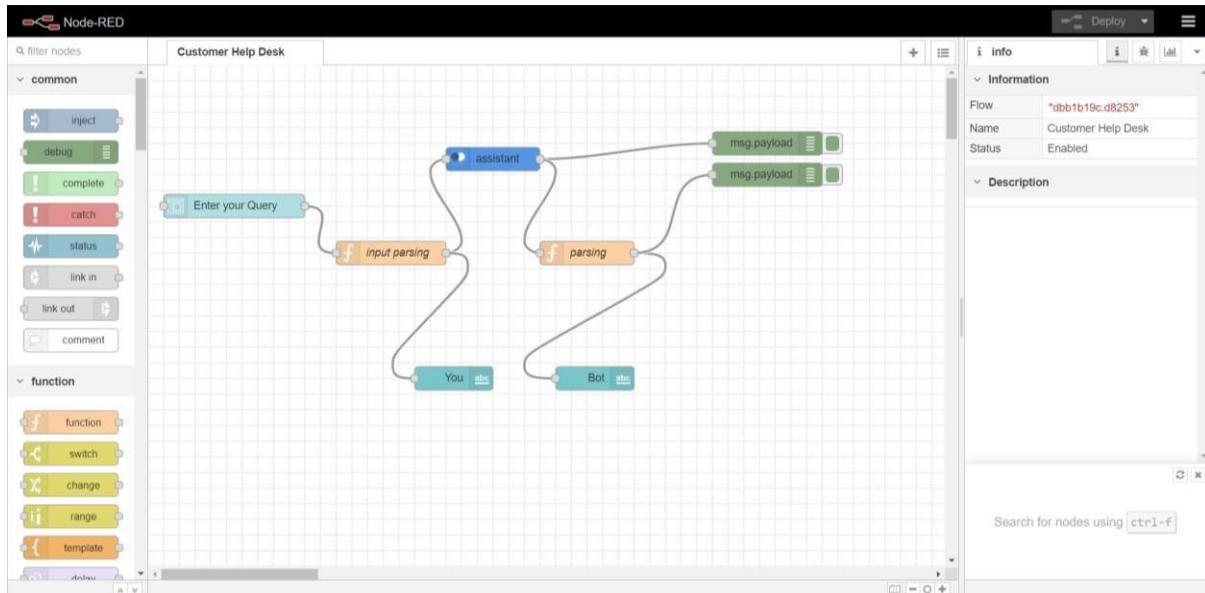


Go to Install Tab [1] and search for node-red-dashboard and Install [2] it.



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

**Customer Help**

enter the question \*  
how to adjust screen brightness

SUBMIT

CANCEL

You

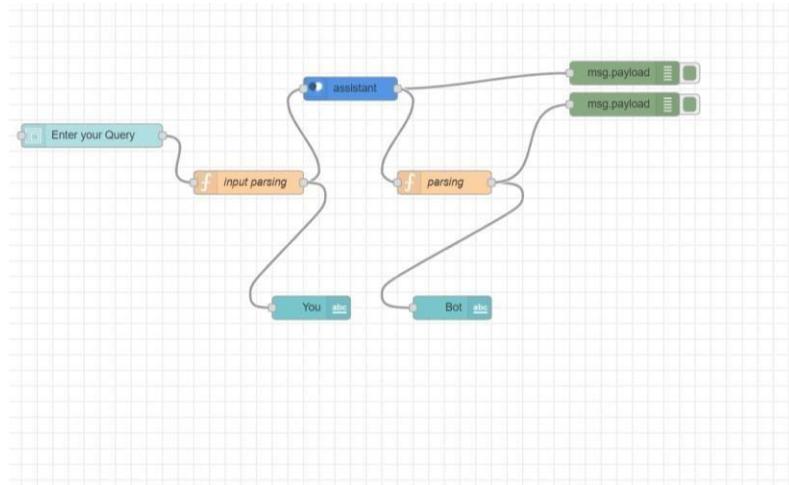
how to adjust screen brightness

bot

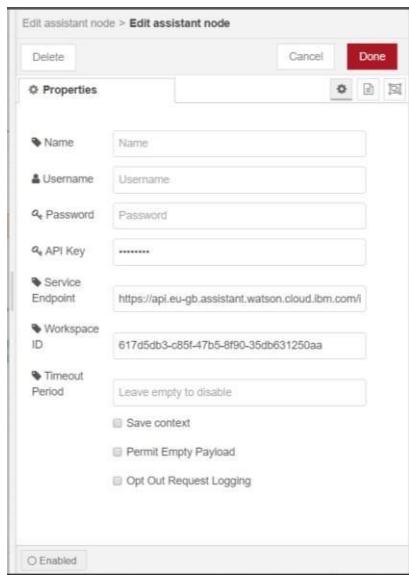
You can customize the brightness of your ecobee3's screen. The brightness for both the active and standby screens can be configured independently. You can also configure the screen to automatically sleep (i.e. turn off) whenever your ecobee3 enters the Sleep activity period. For example, if your thermostat 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.re  
sults[i].text;  
    }  
    msg.payload=msg.payload.text;  
}
```

```
else  
msg.payload = msg.payload.output.text[0];  
return msg;
```

## 6.

## RESULT

The screenshot shows a chatbot interface with a blue header bar labeled "Chatbot". Below it is a light blue "Customer Help" section. A user input field contains the question "how to adjust screen brightness". Below the input field are two buttons: "SUBMIT" on the left and "CANCEL" on the right. The bot's response is displayed in a white box below the input field. It starts with "You" followed by the question "how to adjust screen brightness". The bot then provides a detailed answer: "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."

Chatbot Link:

<https://node-red-ifzfu.eu-gb.mybluemix.net/ui/#!/0?socketid=xz-Ugflhk-HZGtpIAABa>

Explanation Video Link:

[https://youtu.be/sRKkOALM\\_do](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  
 *  
 * 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": "pls try again"  
                        }  
                    ]  
                }  
            ]  
        }  
    }  
]
```

```
        "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!"  
  
                    }  
  
                ]  
  
            }  
  
        ]  
  
    }  
  
}
```

```
        "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": "How can I assist you today?"  
  
            }  
        ]  
    ]  
}
```

```
        "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": [  
            "text": "Hello! How can I assist you today?"  
        ]  
    }  
}
```

```
{
  "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
  },
  "max_retries": 3
}
}
```

```
"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***

