

PROJECT REPORT

Intelligent Customer Help Desk with Smart Document Understanding

Category: Artificial Intelligence Developer

Application ID: SPS_APL_20200000638

Project ID: SPS_PRO_99

Internship at SmartInternz

Rahil Desai

rahildesai.rd99@gmail.com

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

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.

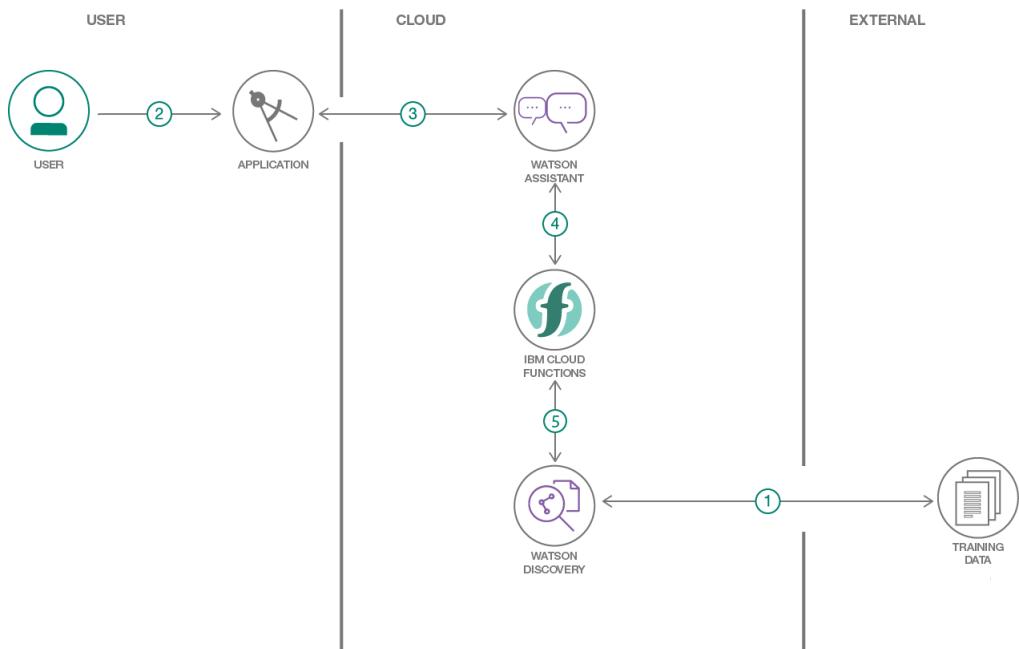
2.2 Proposed solution

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



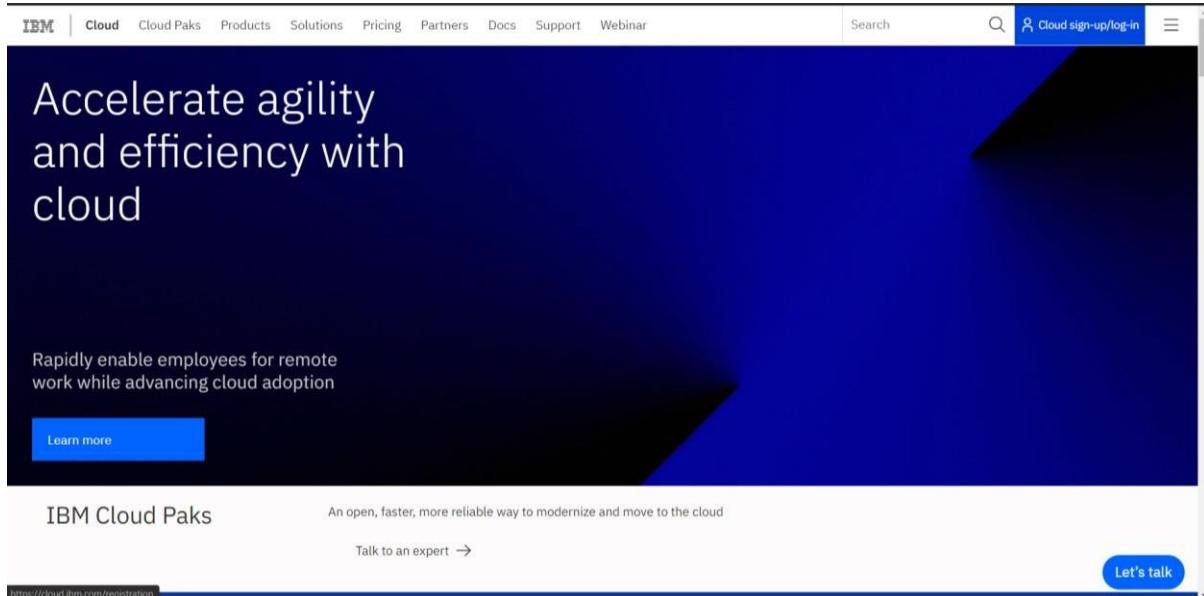
3.2 Hardware / Software designing

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

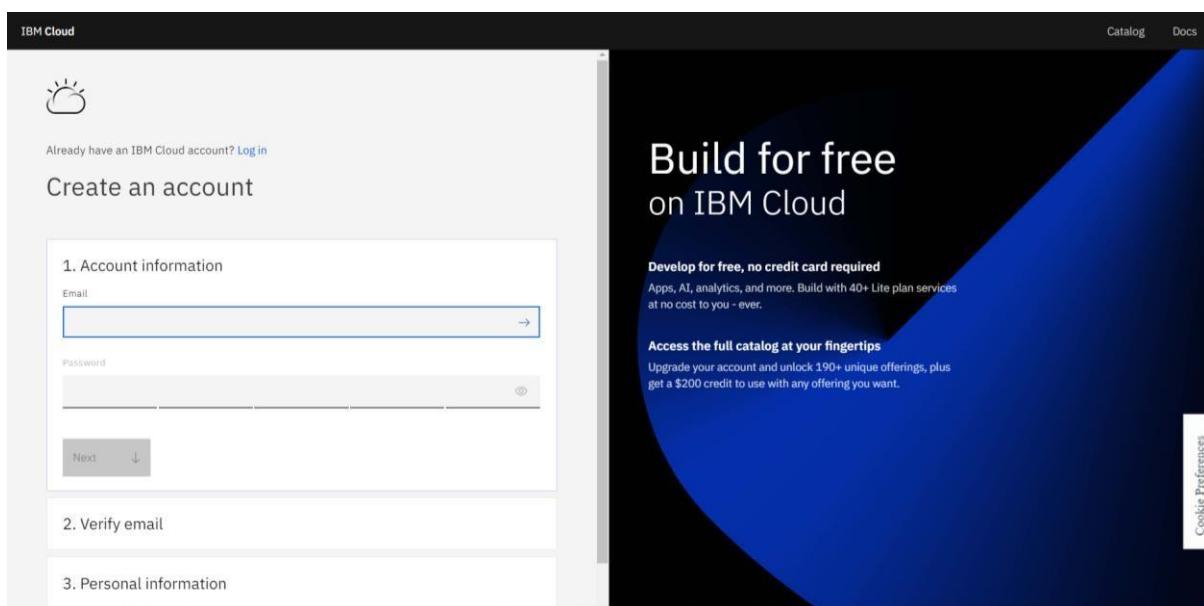
1. Create IBM Cloud Services

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



The screenshot shows the top navigation bar of the IBM Cloud website. It includes links for IBM, Cloud, Cloud Paks, Products, Solutions, Pricing, Partners, Docs, Support, and Webinar. A search bar and a "Cloud sign-up/log-in" button are also present. The main banner features the text "Accelerate agility and efficiency with cloud" and a subtext about enabling remote work. A blue "Learn more" button is visible. Below the banner, there's a section for "IBM Cloud Paks" with a subtext about modernizing and moving to the cloud, a "Talk to an expert" link, and a "Let's talk" button.

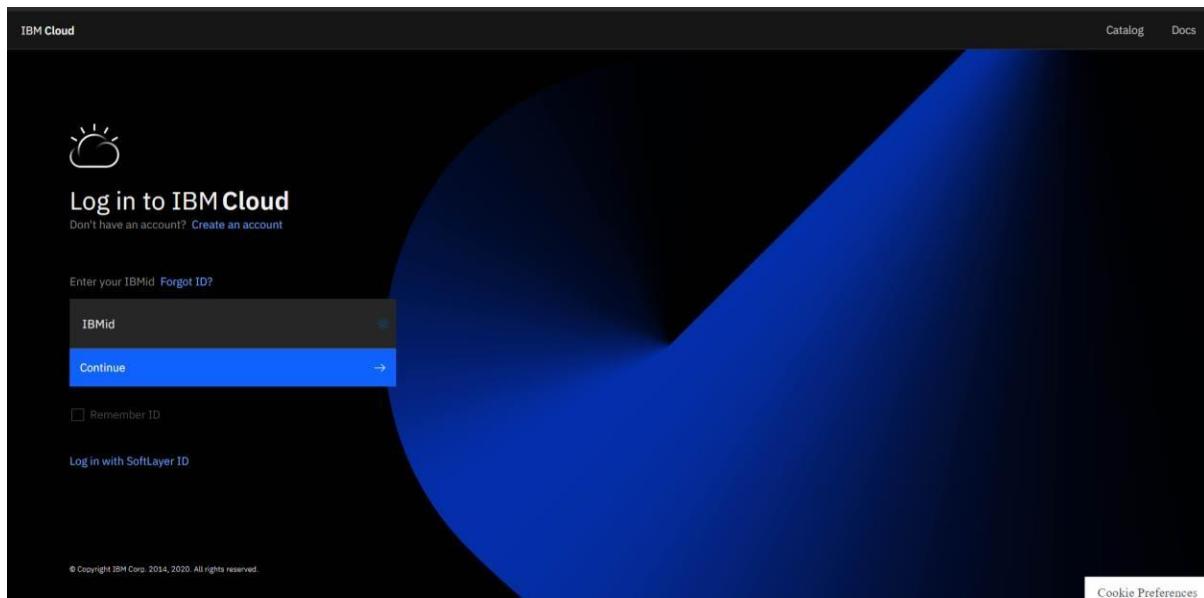
Click on [Cloud sign-up/log-in](#) to sign up or login to IBM Cloud.



The screenshot shows the "Create an account" page. It has three steps: 1. Account information (Email and Password fields), 2. Verify email (Email field), and 3. Personal information (Name and Phone number fields). To the right, there's a promotional section for "Build for free on IBM Cloud" with text about no credit card required and a \$200 credit. There are also links for "Catalog" and "Docs". A "Cookie Preferences" link is at the bottom right.

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. The dashboard features several cards: 'Resource summary' (13 resources, including Cloud Foundry apps, Cloud Foundry services, Services, Storage, Functions namespaces, and Apps), 'Planned maintenance' (clear skies), 'For you' (lightboard video on Infrastructure as Code), 'Leverage code patterns, practises, and architectures to modernize workloads in stages or through a complete...', 'User access' (manage users), 'IBM Cloud status' (world map showing global reach), and news items like 'Airtel Selects IBM and Red Hat to build Open Hybrid Cloud Network' and 'Vodafone Idea Limited Achieves Major Production Milestone with IBM and Red Hat for its Open Universal Hybrid Cloud for Network a...'. A 'Create resource' button is located at the top right of the dashboard area.

To Create any Resource (Services/Apps/etc), click on [Create resource](#) +

The screenshot shows the IBM Cloud Catalog homepage. On the left, there's a sidebar with categories like Catalog, IBM Cloud catalog, Featured, Services, and Software. The main area features a large banner for 'IBM Cloud products' with a search bar below it. Below the banner, there's a section titled 'Recommended for you' with cards for various services: App ID, Object Storage, Visual Recognition, Streaming Analytics, Continuous Delivery, and Speech to Text. A feedback button is visible on the right.

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 'discovery'. Below the search bar, it says 'Search results for 'discovery' 2 results'. There are two service cards: 'Discovery' (selected) and 'HashiCorp Consul'. The 'Discovery' card is highlighted with a blue border and shows its details: 'Add a cognitive search and content analytics engine to applications.', 'Lite • Free • IAM-enabled', and a link to 'https://cloud.ibm.com/catalog/services/discovery?location=eu-uk'. The 'HashiCorp Consul' card shows its details: 'Highly available and distributed service discovery and key-value store designed with support for the modern data cente...', 'Helm charts • IBM Kubernetes Service • Free', and a link to 'https://cloud.ibm.com/catalog/services/consul?location=eu-uk'. A feedback button is visible on the right.

Click on



A screenshot of the IBM Cloud service creation interface for 'Discovery'. The top navigation bar shows 'IBM Cloud', a search bar, and various menu items like Catalog, Docs, Support, Manage, and a user profile. The main area shows the 'Discovery' service card with a 'Summary' section on the right detailing the service name, region (London), plan (Lite), and resource group (Default). Below this, there are tabs for 'Create' and 'About', and a section to 'Select a region' with 'London' selected. A pricing table shows the 'Lite' plan details: 0 - 1,000 documents per month, 200 news queries per month, and 1 custom model. It also notes that the service is free. On the right side, there are buttons for 'Create', 'Add to estimate', and 'View terms', along with a 'FEEDBACK' link.

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

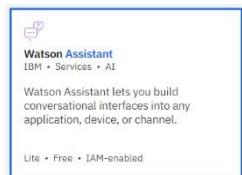
Your Watson Discovery service is created successfully.

(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 text "assistant". Below the search bar, the results are displayed under the heading "Search results for 'assistant' 1 result". The single result is a card for "Watson Assistant" from IBM, categorized under Services > AI. The card includes a brief description: "Watson Assistant lets you build conversational interfaces into any application, device, or channel." It also indicates the plan as "Lite • Free • IAM-enabled". The entire service card is highlighted with a blue border.

Click on



The screenshot shows the "Watson Assistant" creation page. At the top, there is a breadcrumb navigation: Catalog / Services / Watson Assistant. Below the title, it says "Author: IBM • Date of last update: 05/14/2020 • Docs • API docs". There are two tabs: "Create" (which is selected) and "About". The main area has two sections: "Select a region" and "Select a pricing plan". Under "Select a region", a dropdown menu is open, showing "London" as the selected option. Under "Select a pricing plan", it says "Displayed prices do not include tax. Monthly prices shown are for country or region: United States". A table compares the "Lite" plan against the "Features" and "Pricing". The "Lite" plan includes 10,000 Messages/Month, AI-Based Intent and Entity Recognition, Entity Synonym Recommendations, Visual Dialog Edit with Simple Response Types (Text, Options, Images, etc...), Prebuilt Content Available, Analytics Dashboard with 7 Days of Storage, 5 Dialog Skills, Each with 100 Dialog Nodes, and Shared Public Cloud. The "Pricing" column shows "Free". On the right side, there is a "Summary" panel with the following details: Watson Assistant (Service name), Region: London, Plan: Lite, Service name: Watson Assistant-ro, Resource group: Default. Below the summary, there are buttons for "Create", "Add to estimate", and "View terms". A feedback button is located at the bottom right.

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.

Resource summary

13 Resources

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

Add resources +

View all

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

IBM Cloud

Search resources and offerings...

Catalog Docs Support Manage Hashim Irfan Al... Create resource

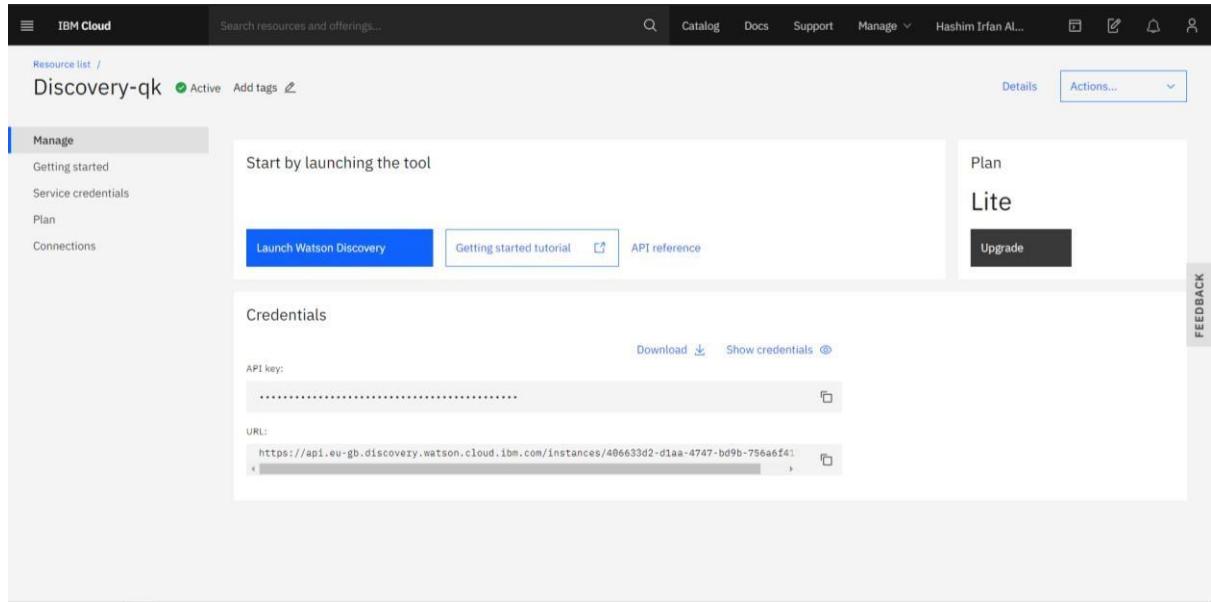
Resource list

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.

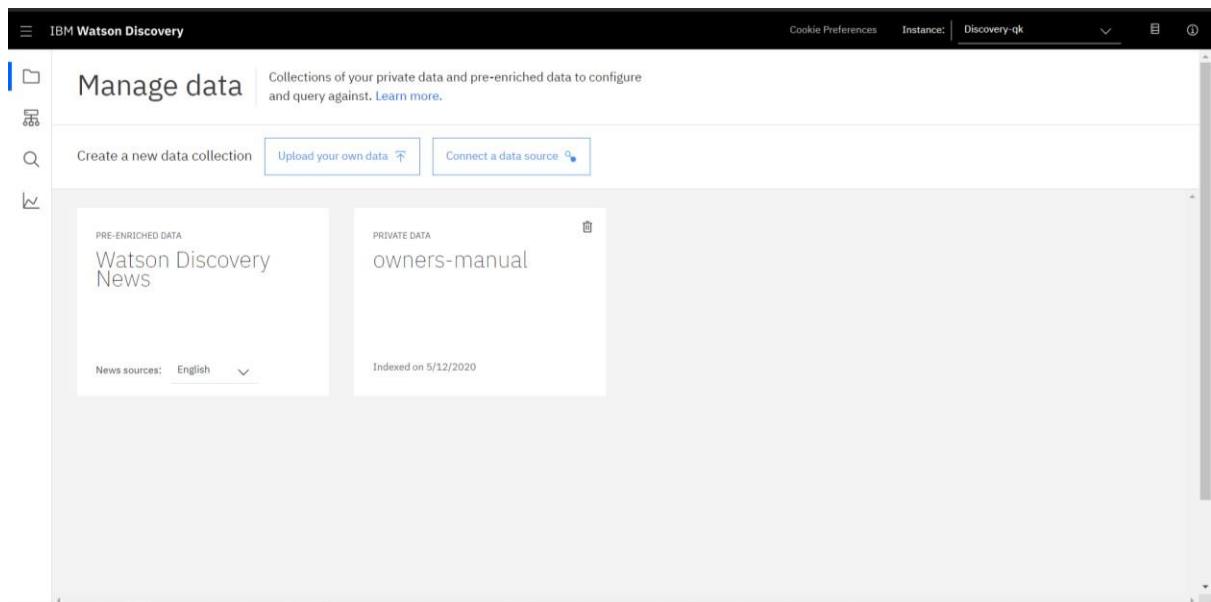
2. Configure Watson Discovery

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



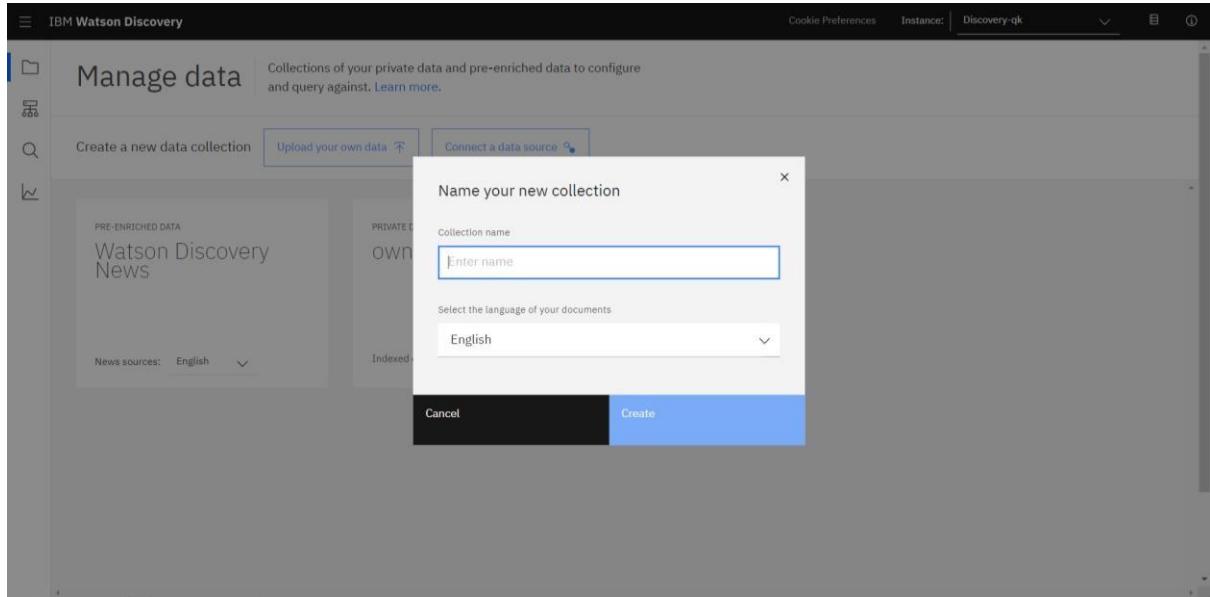
The screenshot shows the IBM Cloud Resource List interface. In the center, there is a card for the "Discovery-qk" service. On the left side of the card, under the "Manage" section, there is a blue button labeled "Launch Watson Discovery". To the right of this button are links for "Getting started tutorial" and "API reference". On the far right of the card, there is a "Plan" section showing "Lite" and a "Upgrade" button. At the top of the page, there is a navigation bar with links for Catalog, Docs, Support, Manage, and a user profile.

Click on  to launch Watson Discovery Service.

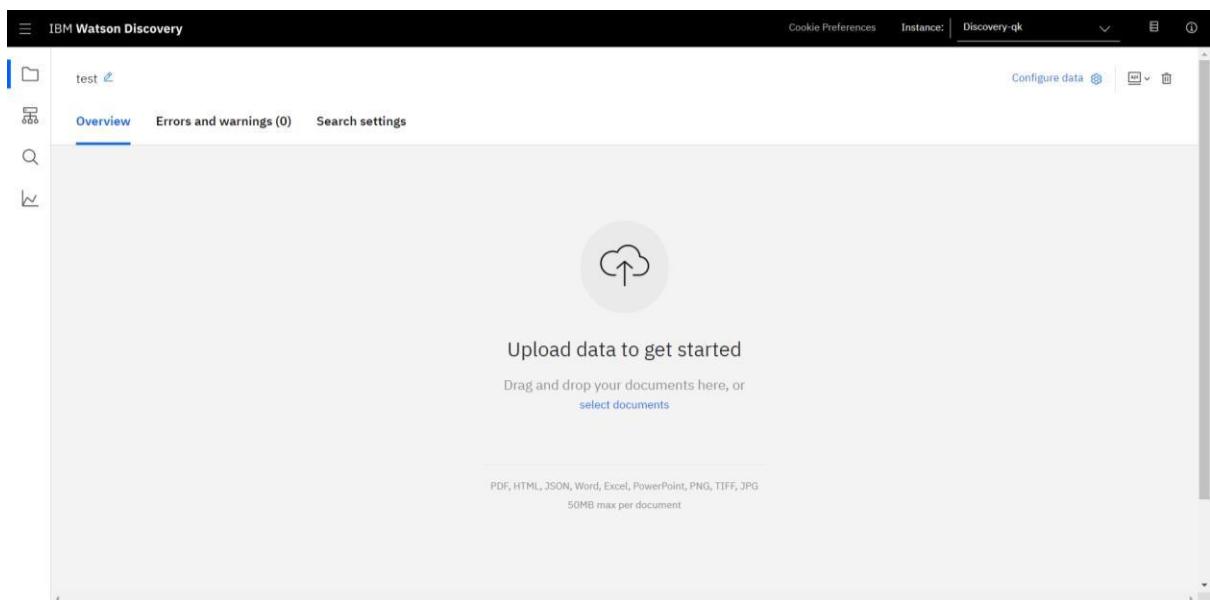


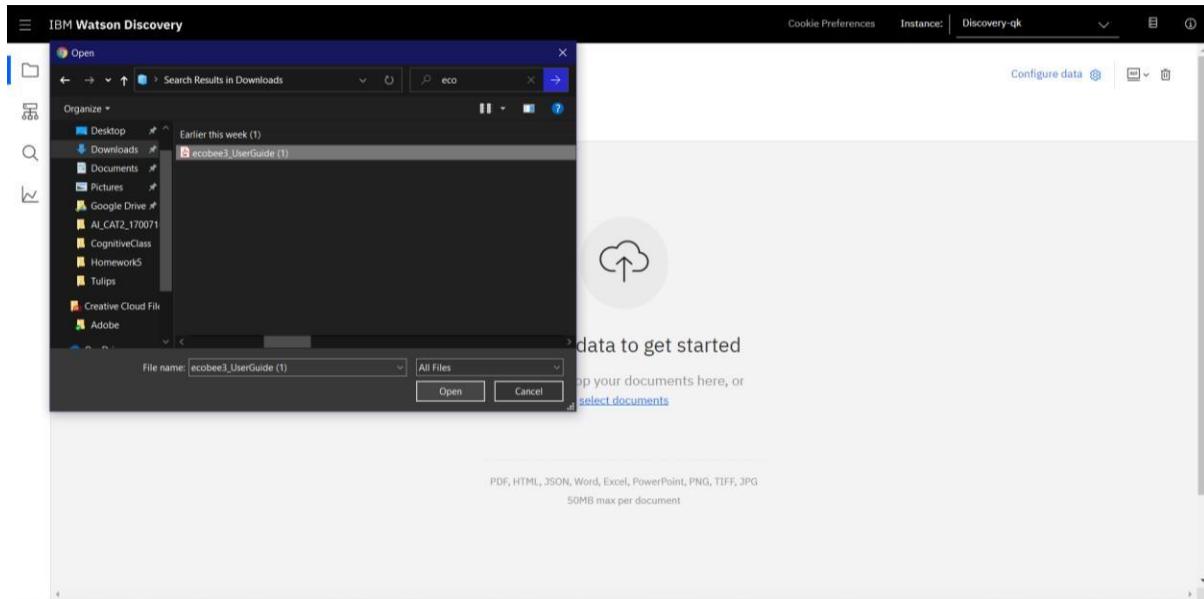
The screenshot shows the IBM Watson Discovery interface. At the top, there is a header with the title "IBM Watson Discovery" and a dropdown for "Discovery-qk". Below the header, there is a section titled "Manage data" which says "Collections of your private data and pre-enriched data to configure and query against. Learn more.". Under this section, there are three buttons: "Create a new data collection", "Upload your own data", and "Connect a data source". Below these buttons, there are two data preview cards. The first card, titled "PRE-ENRICHED DATA", shows "Watson Discovery News" and "News sources: English". The second card, titled "PRIVATE DATA", shows "owners-manual". At the bottom of the page, there is a footer with links for "Cookie Preferences", "Instance", and "Discovery-qk".

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.

The screenshot shows the Watson Discovery interface after processing the uploaded document. Key details include:

- Overview:** Shows 1 document.
- Identified fields:** 1 field identified, labeled "text".
- Added enrichments:** 4 enrichments added, including Entity Extraction (104 °F, 20 min, etc.), Sentiment Analysis (100%, 0%, 0%), Concept Tagging (Air conditioner, Energy recovery, Geothermal heat pump), and Category Classification.
- Analysis metrics:** Entity Extraction (104 °F, 20 min, etc.), Sentiment Analysis (100%, 0%, 0%), Concept Tagging (Air conditioner, Energy recovery, Geothermal heat pump), and Category Classification.
- Ready to query:** A sidebar suggests queries like "Most common entity types and their top entities" and "Documents that contain Air conditioner, but not Energy recovery".

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 the tabs, 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 button to 'Upload documents'. 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). On the right, it says 'Now you're ready to query!' with three cards: '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 at the bottom right.

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 'test / Build queries' and a main area for building a query using components like Entity Extraction, Sentiment Analysis, and Category Classification. A sample query 'how do i turn on the heater' is entered. Below it, there are options to 'Include analysis of your results' and 'Filter which documents you query'. On the right, the 'Summary' tab is selected, showing a summary URL: <https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/4>. The 'Passages' section contains a snippet from a document about the ecobee3 smart Wi-Fi thermostat. The 'Results' section shows one matching document: 'ecobee3_UserGuide (1).pdf' with sentiment 'positive'.

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)

The screenshot shows the IBM Watson Discovery interface. At the top, there's a navigation bar with 'IBM Watson Discovery' and 'Discovery-qk'. Below it, a sidebar has 'test' and 'Configure data' (which is highlighted with a red box). The main area has tabs for 'Overview', 'Errors and warnings (1)', and 'Search settings'. Under 'Overview', it says '1 document' with a 'View details' link. It also shows '0 documents failed'. On the right, there are sections for 'Created on' (5/16/2020 11:29:11 am EDT) and 'Last updated' (5/16/2020 11:29:11 am EDT). A 'Upload documents' button is present. Below this, there's a summary of identified fields and enrichments. 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, there's a section titled 'Now you're ready to query!' with three examples: 'Documents about 104 °F as a Quantity with a very negative sentiment', 'Top people related to /business and industrial/energy', and 'Most common entity types and their top entities'. A note at the bottom says '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

The screenshot shows the 'Identify fields' tab in the SDU annotation panel. At the top, there's a navigation bar with 'IBM Watson Discovery' and 'Discovery-qk'. Below it, a sidebar has 'Identify fields' (which is highlighted with a red box), 'Manage fields', and 'Enrich fields'. The main area shows a list of pages: 'ecobee3_UserGuide (1).pdf' with '1/41'. A large red number '1' is over the sidebar. A red number '2' is over the current page 'User Guide ecobee3'. A red number '3' is over a yellow selection bar. A red number '4' is over the 'Field labels' section on the right, which lists labels like 'answer', 'author', 'footer', etc. A red number '5' is over the 'Submit page' button at the bottom right.

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.

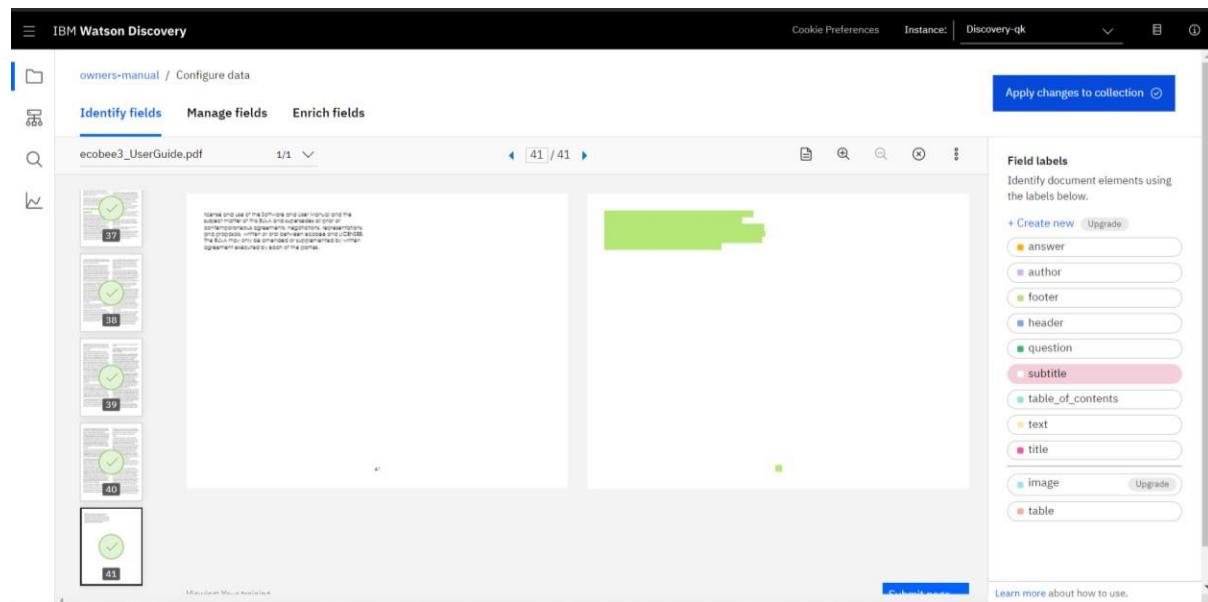
As you go through the annotations one page at a time, Discovery is learning and should start automatically updating the upcoming pages. Once you get to a page that is already correctly annotated, you can stop, or simply click Submit [5] to acknowledge it is correct. The more pages you annotate, the better 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 sidebar with a folder icon, a search icon, and a refresh icon. The main area displays a PDF document titled "ecobee3_UserGuide.pdf". The document has five pages numbered 37, 38, 39, 40, and 41. Each page has a green checkmark icon in the top-left corner. To the right of the document, 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. Each category is represented by a colored circle and a small icon. At the top right of the interface, there's a blue button labeled "Apply changes to collection".

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

IBM Watson Discovery

owners-manual / Configure data

Identify fields Manage fields Enrich fields

Identify fields to index

All fields are indexed by default. Switch off any fields you do not want to be indexed. [Learn more.](#)

answer	Off
author	Off
footer	Off
header	Off
image	Off
question	Off
subtitle	On
table	Off
table_of_contents	Off
text	On
title	Off

Improve query results by splitting your documents

You can split your documents into segments based on fields. Once split, each segment is a separate document that will be enriched, indexed, and returned as a query separately. [Learn more.](#)

Split document on each occurrence of

subtitle

Apply changes to collection

[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

IBM Watson Discovery

owners-manual

Configure data

Overview Errors and warnings (133) Search settings

133 documents

0 documents failed View details

Created on Last updated

5/12/2020 4:09:01 am EDT 5/12/2020 4:09:01 am EDT

Upload documents

Identified **6 fields** from your data

author
footer
subtitle
table_of_contents
text
title

Need to identify more fields? [Add fields](#)

Added **4 enrichments** to your data

Entity Extraction
0.3°C (4) | 0.5°F (4) | 10°F (4) |
20 min (3) | 4-digit (3)

Sentiment Analysis
54% positive
35% neutral
11% negative

Concept Tagging
HVAC (13) | Heat (13) | Internet (12) |
Temperature (11) | Netscape (10)

Category Classification
technology and com... operating systems

5 enrichments available. [Add enrichments](#)

Now you're ready to **query**!

Top people related to /technology and computing/operating systems
Run

Entities of type **Quantity** which have negative sentiment
Run

Entities of type **Quantity** which have positive sentiment
Run

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" and a search icon. The main area has a search bar with the placeholder "Build a query using one or more of these components. Learn more." Below it, there are two tabs: "Use natural language" (which is selected) and "Use the Discovery Query Language". A text input field contains the query "how do i turn on the heater?". To the right, there are several sections: "Summary" (which is selected), "JSON", "Passages", and a detailed description of the HVAC system settings. At the bottom, there are buttons for "Run query" and "Close".

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 "Overview" tab of the IBM Watson Discovery interface. It displays basic statistics: 132 documents, 0 failed documents, and creation and update dates. On the right, there are three boxes: "Collection ID" (3be...), "Configuration ID" (aa5...), and "Environment ID" (d5f...). Below these are four cards: "Entity Extraction" (0.3°C | 0.5°F | 10 °F | 20 min | 4-digit), "Sentiment Analysis" (54% positive, 35% neutral, 11% negative), "Concept Tagging" (HVAC (13) | Heat (13) | Internet (12) | Temperature (11) | Netscape (10)), and "Category Classification" (technology and com... operating systems). At the bottom, there are buttons for "Run" and "Add enrichments".

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 'Service credentials' section for a service named 'Discovery-qk'. On the left, there's a sidebar with 'Manage', 'Getting started', 'Service credentials' (which is selected and highlighted in blue), 'Plan', and 'Connections'. The main area has a search bar 'Search credentials...' and a table header with 'Key name' and 'Date created'. A single row is listed under 'Auto-generated service credentials':

Key name	Date created
apikey	MAY 12, 2020 - 01:06:25 PM

The 'apikey' row contains the following JSON data:

```
{ "apikey": "c[REDACTED]88", "iam_apikey_description": "Auto-generated for key 94fa8a96-9c49-404d-8072-5dee56e0d754", "iam_apikey_name": "Auto-generated service credentials", "iam_crn": "cnr:v1:bluemix:public:iam::::serviceRole:Manager", "iam_serviceid_crn": "cnr:v1:bluemix:public:iam:identity:a/5a31b9c581d4b939514c678b699110c::serviceid:ServiceId-d18ad745-ed3f-44a6-a175-38db2b17307f", "url": "https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/406633d2-d1aa-4747-bd9b-756a6f417348" }
```

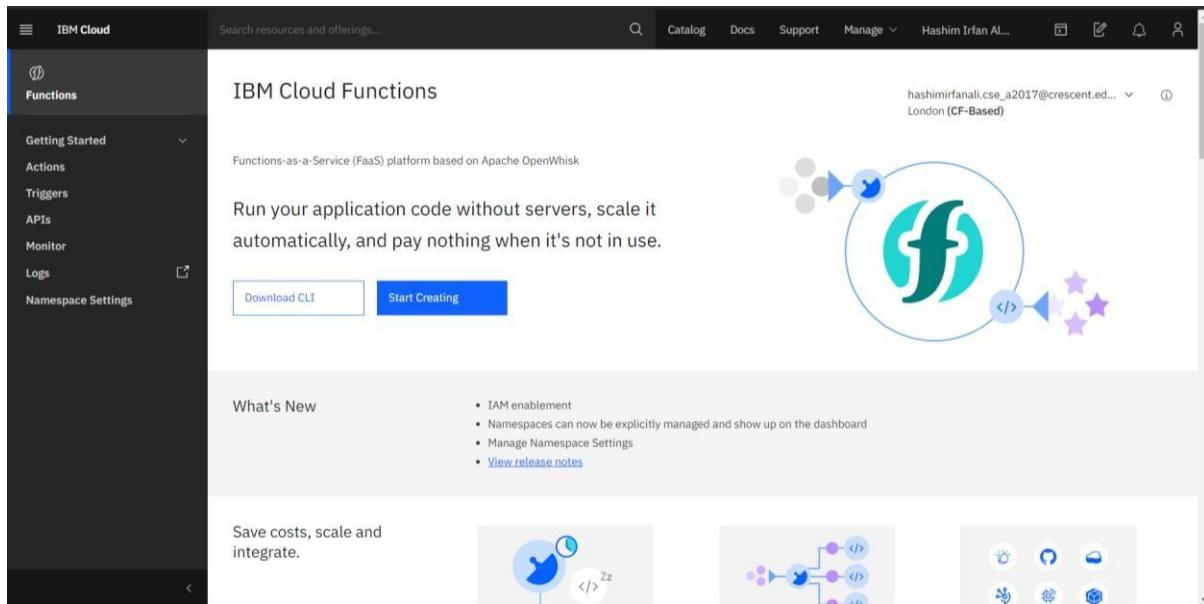
3.Create IBM Cloud Functions action

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

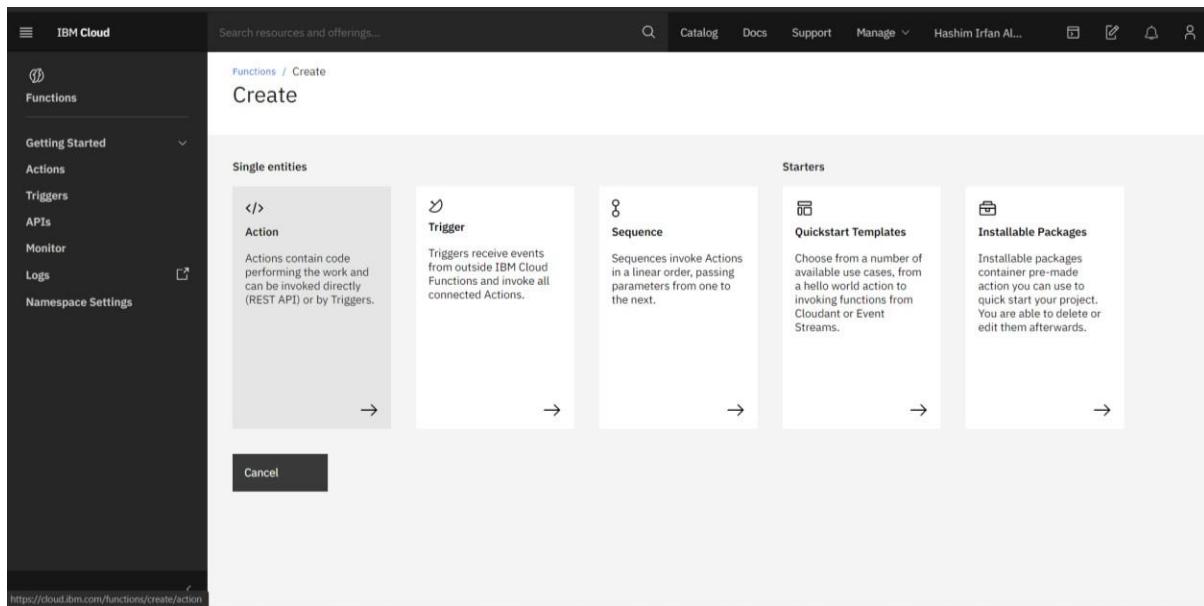
The screenshot shows the IBM Cloud Catalog search results for 'functions'. The search bar at the top has 'functions' typed into it. Below the search bar, the results are displayed with a title 'Search results for \'functions\' 1 result'. One result is shown in a box: 'Functions' (Services > Compute). It is described as a Function-as-a-Service (FaaS) platform that executes functions in response to incoming events. A note below says 'IAM-enabled'. The sidebar on the left includes sections for Catalog, IBM Cloud catalog, Featured, Services, Software, Category (Compute), Product (Services), Provider (IBM), and Compliance (IAM-enabled).

Select the Functions Card.



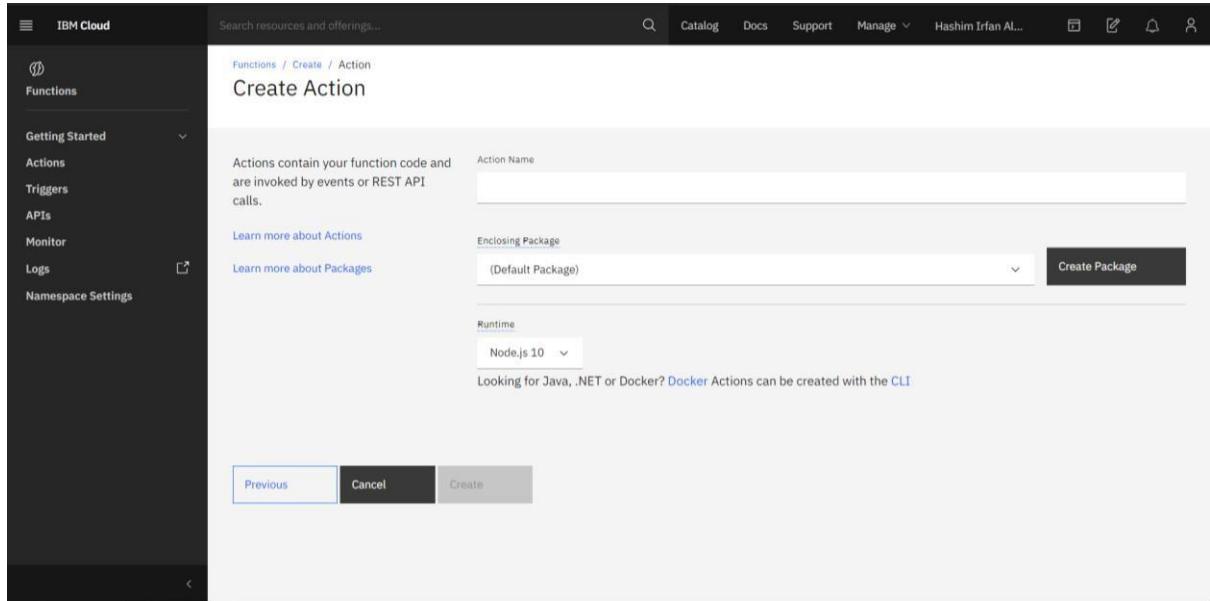
The screenshot shows the IBM Cloud Functions main panel. On the left, there's a sidebar with a 'Functions' card selected. The main content area is titled 'IBM Cloud Functions' and describes it as a 'Functions-as-a-Service (FaaS) platform based on Apache OpenWhisk'. It encourages users to 'Run your application code without servers, scale it automatically, and pay nothing when it's not in use.' Below this are two buttons: 'Download CLI' and 'Start Creating'. To the right is a large circular icon featuring a stylized 'ff' logo with arrows pointing towards it. The top navigation bar includes links for Catalog, Docs, Support, Manage, and user information.

From the Functions main panel, click on Start Creating.



The screenshot shows the 'Create' panel within the IBM Cloud Functions interface. The left sidebar remains the same. The main area is titled 'Create' and contains several cards under the heading 'Single entities': 'Action' (described as containing code performing work), 'Trigger' (described as receiving events from outside), 'Sequence' (described as invoking Actions in a linear order), 'Quickstart Templates' (described as choosing from available use cases), and 'Installable Packages' (described as container pre-made actions). Each card has a small icon and a right-pointing arrow indicating they can be selected. A 'Cancel' button is at the bottom left.

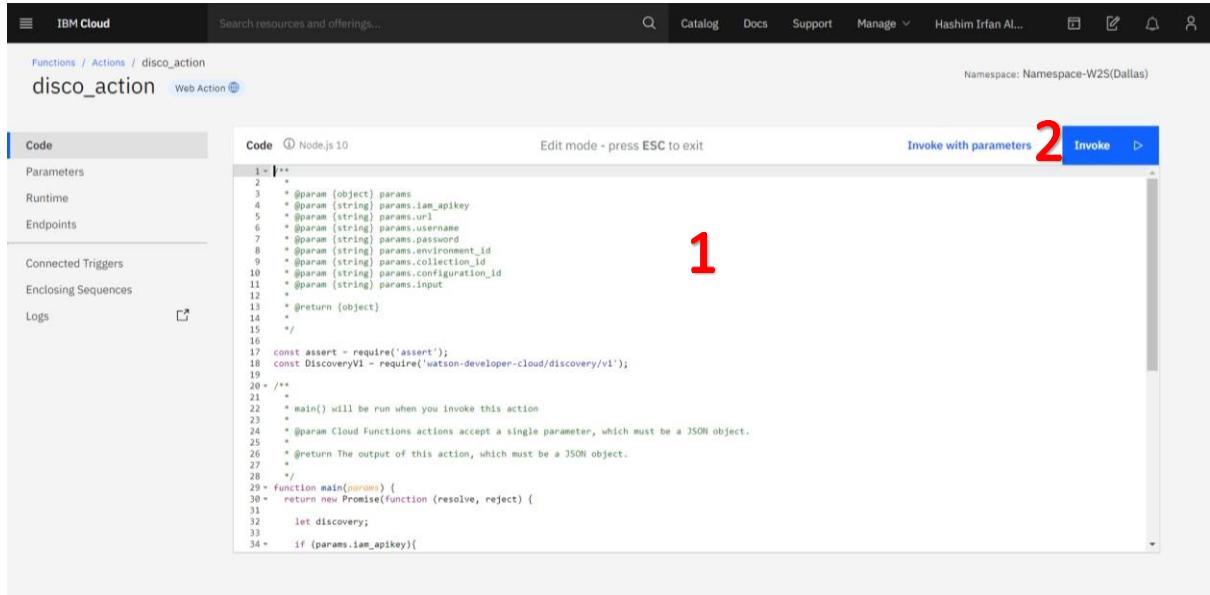
Here, select Actions.



The screenshot shows the 'Create Action' page in the IBM Cloud Functions interface. On the left, there's a sidebar with 'Functions' selected. The main area has a title 'Create Action'. It includes fields for 'Action Name' (empty), 'Enclosing Package' (set to '(Default Package)'), and 'Runtime' (set to 'Node.js 10'). A note at the bottom says 'Looking for Java, .NET or Docker? Docker Actions can be created with the CLI'. At the bottom are buttons for 'Previous', 'Cancel', and '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.



The screenshot shows the 'disco_action' function details page. The 'Code' tab is selected, showing the Node.js 10 code. The code is a simple function that uses the Watson Developer Cloud Discovery service to query a collection. The code editor window is labeled [1]. To the right, there's an 'Invoke with parameters' section with an 'Invoke' button, which is highlighted with a red box and labeled [2].

```
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 */
12 *
13 * @return {object}
14 *
15 */
16
17 const assert = require('assert');
18 const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');
19
20 /**
21 *
22 * main() will be run when you invoke this action
23 *
24 * @param Cloud Functions actions accept a single parameter, which must be a JSON object.
25 * @param {Object} params
26 * @return {Object} The output of this action, which must be a JSON object.
27 */
28
29 function main(params) {
30 return new Promise(function (resolve, reject) {
31
32 let discovery;
33
34 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 left sidebar has tabs for Code, Parameters (which is selected and highlighted with a red '1'), Runtime, Endpoints, Connected Triggers, Enclosing Sequences, and Logs. The main area is titled 'Parameters' and contains four entries:

Parameter Name	Parameter Value
url	"https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/406633d2-d1aa-474
environment_id	"d1aa-4745"
collection_id	"315f"
iam_apikey	"cA8B8"

At the top right of the parameters table is a blue 'Add Parameter' button.

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 left sidebar has tabs for Code (which is selected), Parameters, Runtime, Endpoints, Connected Triggers, Enclosing Sequences, and Logs. The main area is titled 'Code' and shows a Node.js file named 'Node.js 10'. The code content is as follows:

```
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  */
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   * @param Cloud Functions actions accept a single parameter, which must be a JSON object.
23   * @return The output of this action, which must be a JSON object.
24   */
25
26  function main(params) {
27    return new Promise(function (resolve, reject) {
28      let discovery;
29      if (params.iam_apikey){
```

On the right side, there is an 'Activations' panel showing one entry:

Activation ID	disco_action	1450 ms	5/16/2020, 22:24:23
Results:	{ "matching_results": 132, "passages": 1, "results": [{ "enriched_text": { "categories": [{ "label": "/business and industrial/energy", "score": 0.69535 }], "label": "/business and industrial/green solutions", "score": 0.643533 }, { "label": "/business and industrial/business operations", "score": 0.633224 }], "concepts": [{ "label": "/business and industrial/green solutions", "score": 0.643533 }] }		

Now click on endpoints[1] tab

The screenshot shows the IBM Cloud Functions Actions interface. The left sidebar has tabs for Code, Parameters, Runtime, Endpoints (which is highlighted with a blue box and has a red number '1' over it), Connected Triggers, Enclosing Sequences, and Logs. The main area is titled 'Web Action'. It contains two sections: 'Web Action' and 'REST API'. Under 'Web Action', there is a checked checkbox for 'Enable as Web Action' with a note about returning a dict object. There is also an unchecked checkbox for 'Raw HTTP handling'. Below these are fields for 'HTTP Method' (set to ANY), 'Auth' (set to Public), and 'URL' (set to https://us-south.functions.cloud.ibm.com/api/v1/web/728c6dcb-3bf4-4108-b946-4e5cc6fc84d6/default/disco_action). Under 'REST API', there is a field for 'HTTP Method' (set to POST), 'Auth' (set to IAM TOKEN), and 'URL' (set to https://us-south.functions.cloud.ibm.com/api/v1/namespaces/728c6dcb-3bf4-4108-b946-4e5cc6fc84d6/actions/disco_action). At the bottom, there is a 'CURL' section with a pre-generated command.

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

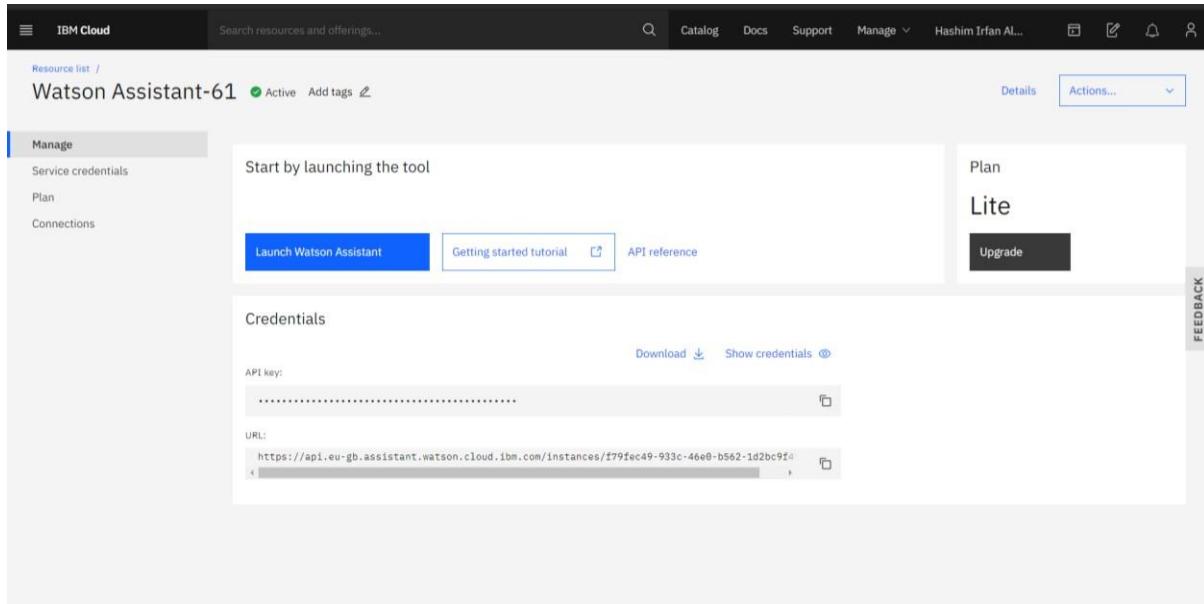
Take note of the URL value [3], as this will be needed by Watson Assistant in a future step.

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).]

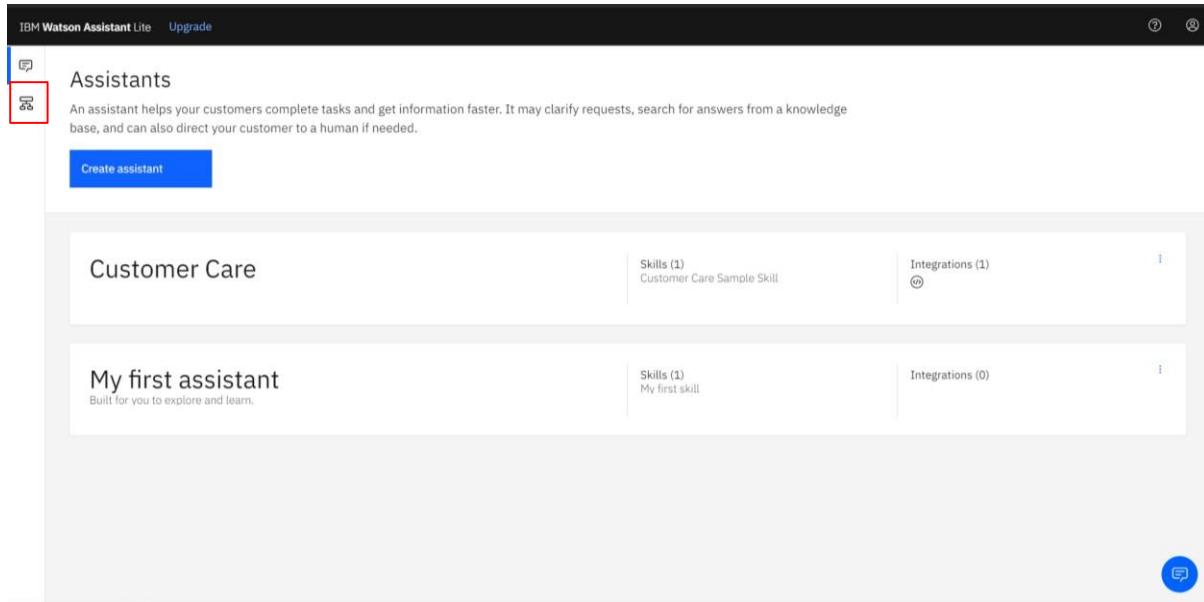
4. Configure Watson Assistant

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



The screenshot shows the IBM Cloud Resource list interface. A service named "Watson Assistant-61" is listed as active. The service card includes sections for "Manage" (Service credentials, Plan, Connections), "Start by launching the tool" (Launch Watson Assistant button, Getting started tutorial, API reference), and "Plan" (Lite, Upgrade). The "Lite" plan is selected. A "Feedback" button is visible on the right side of the card.

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



The screenshot shows the "IBM Watson Assistant Lite" service details page. It features a sidebar with "Assistants" (Create assistant button) and two main cards: "Customer Care" (Skills: Customer Care Sample Skill, Integrations: 1) and "My first assistant" (Skills: My first skill, Integrations: 0). A blue speech bubble icon is located at the bottom right of the page.

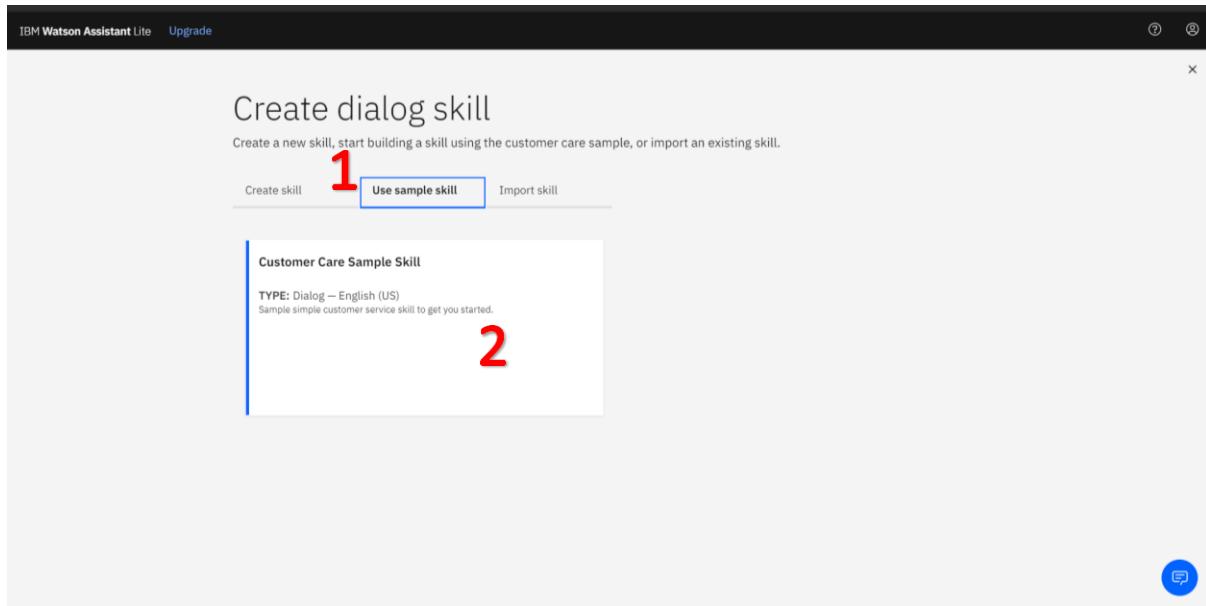
Click on the skills tab

The screenshot shows the 'Skills' section of the IBM Watson Assistant Lite interface. At the top, there's a header with 'IBM Watson Assistant Lite' and 'Upgrade' buttons. Below the header, a sidebar has a 'Skills' icon. The main area is titled 'Skills' and contains a sub-header: '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 highlighted with a red box. Below this, two skill cards are listed: 'Customer Care Sample Skill' and 'My first skill'. Each card displays 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 is 'Create a skill' and it says 'Skills can be combined to improve your assistant's capabilities. [Learn more](#)'. It asks to 'Select skill type'. Two options are shown: 'Dialog skill' (selected) and 'Search skill'. The 'Dialog skill' card is highlighted with a blue box. It describes dialog flows for addressing customer issues. The 'Search skill' card is described as for customer questions or requests that require lengthy or complex responses, mentioning self-service content access. A 'Next' button is at the bottom.

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, there is a sidebar with options like Entities, Dialog, Options, Analytics, Versions, and Content Catalog. The main area displays a table of intents:

	Intents (10) ↑	Description	Modified ↑↑	Examples ↑↑
<input type="checkbox"/>	#Cancel	Cancel the current request	15 days ago	7
<input type="checkbox"/>	#Customer_Care_Appointments	Schedule or manage an in-store appointment.	15 days ago	20
<input type="checkbox"/>	#Customer_Care_Store_Hours	Find business hours.	15 days ago	48
<input type="checkbox"/>	#Customer_Care_Store_Location	Locate a physical store location or an address.	15 days ago	25
<input type="checkbox"/>	#General_Connect_to_Agent	Request a human agent.	15 days ago	47
<input type="checkbox"/>	#General_Greetings	Greetings	15 days ago	30
<input type="checkbox"/>	#Goodbye	Good byes	15 days ago	6
<input type="checkbox"/>	#Help	Ask for help	15 days ago	8
<input type="checkbox"/>	#Product_Information		4 days ago	3

At the bottom, it says 'Showing 1–10 of 10 intents'. There are buttons for 'Create intent' (highlighted with a red box), 'Save new version', 'Try it', and other navigation controls.

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 bar says '#Product_Information'. On the right side of the title bar, there are buttons for 'Last updated: 4 days ago', a download icon, a search icon, and a 'Try it' button. The main area has sections for 'Intent name' (containing '#Product_Information'), 'Description (optional)' (with a placeholder 'Add a description to this intent'), and 'User example' (with a placeholder 'Type a user example here, e.g. I want to pay my credit card bill'). Below these sections is a button bar with 'Add example' and 'Show recommendations'. A list of user examples is shown, each with a checkbox, a title, and a timestamp ('4 days ago'). The list starts with 'How do i access the settings' and 'How do i set the time'. At the bottom of the list, it says 'Showing 1-3 of 3 examples'.

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 left sidebar has tabs for 'Intents', 'Entities', 'Dialog' (which is selected and highlighted in blue), 'Options', 'Analytics', 'Versions', and 'Content Catalog'. In the main area, there's a 'Dialog' section with a tree view. The root node is 'What can I do?'. Below it are several child nodes: 'Opening welcome', 'What are your hours? #Customer_Care_Store_Hours', 'Where are you located? #Customer_Care_Store_Location', 'I want to make an appointment #Customer_Care_Appointments', '#General_Greetings', and '#Goodbyes'. Each node has a small icon and three dots to its right. Above the tree, there are buttons for 'Add node' (highlighted in blue), 'Add child node', and 'Add folder'. On the right side, there are buttons for 'Save new version' and 'Try it'.

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 dialog flow for a 'Customer Care Sample Skill'. The flow starts with an 'Intents' section containing nodes for '#General_Greetings', '#Goodbye', '#Thanks', and 'Please transfer me to an agent'. Below this is an 'Entities' section with a single node for 'What can I do'. At the bottom of the flow is a 'Dialog' section with a node for 'Ask product information'. To the right of the flow, there's a configuration panel for the 'Ask product information' node. It has a title 'Ask product information' with a note: 'Node name will be shown to customers for disambiguation so use something descriptive.' There are 'Customize' and 'Settings' buttons. Below this is a section titled 'If assistant recognizes' with a dropdown menu showing '#Product_Information'. Further down is an 'Assistant responds' section with a 'Text' input field containing 'Enter response text' and a note about response variations being set to 'sequential'. There are also 'Learn more' and 'Add response type' buttons.

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

Intents Entities Dialog Options Webhooks

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/728c6db-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

Intents Entities Dialog Options Analytics Versions Content Catalog

#Good

#Thank

Please

What c

Ask pr

anythin

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 'Dialog' tab is selected, showing 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', and 'anything_else'. The 'Ask product information' node is currently selected. On the right, the configuration panel for this node is open, titled 'Ask product information'. It includes fields for 'Node name will be shown to customers for disambiguation so use something descriptive.' and 'Customize' and 'Settings' buttons. Below this is a section titled 'Then callout to my webhook' with a 'Learn more' link. Under 'Parameters', there is a table with a single row: 'Key' (input) and 'Value' ('<?input.text?>'). Under 'Return variable', the value '\$webhook_result_1' is listed. A green success message at the bottom states 'Webhook URL Your webhook URL is configured.' with a 'Options' link. Red numbers '1' and '2' are overlaid on the screen: '1' points to the 'Return variable' field, and '2' points to the 'Value' field of the parameter table.

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:

This screenshot shows the same interface as the previous one, but with additional configuration for 'Assistant responds'. In the 'If assistant recognizes' column, two entries are listed: '1 \$webhook_result_1' and '2 anything_else'. In the 'Respond with' column, the first entry is '\$webhook_result_1' and the second is 'Try again later'. The row for '2 anything_else' is highlighted with a red box. Below this, under 'Then assistant should', there is a 'Try it' button.

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, the sidebar includes options like Intents, Entities, Dialog (which is selected), Options, Analytics, Versions, and Content Catalog. The main workspace displays a dialog node named "Ask product information". The "If assistant recognizes" section contains two entries: "1 \$webhook_result_1" and "2 anything_else". The "Respond with" section shows responses: "\$webhook_result_1" for the first entry and "Try again later" for the second. A "Customize" button and a "Settings" link are also present. To the right, a "Try it out" window shows a conversation history: "Hello, I'm your virtual customer care assistant, Gary. I can help with directions to my store, hours of operation and booking an in-store appointment", followed by "Hi", "#General_Greetings", "Hello. Good evening", "How to turn on heater?", "#Product_Information", and a JSON response object. At the bottom right of the workspace is a blue message icon.

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 shows the same interface as the previous one, but with a "Context variables" panel open on the right side. The panel lists several variables with their current values: \$ Enter variable name (empty), \$timezone ("Asia/Calcutta"), \$no_reservation ("true"), and \$webhook_result_1 (a JSON object: {"matching_results":22,"passages":[{"document_id":"35a5eef70d8cc9d70e2b94d2c15e2d1_11","end_offset":240,"field_text","passage_score":0,"passage_text":"I"}]). A "Remove all context variables" button is at the bottom of the panel. The workspace itself is mostly identical to the previous screenshot, showing the dialog node configuration and the "Try it out" window.

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. On the left, there's a card for 'Customer Care Sample Skill' with details like TYPE: Dialog – English (US), CREATED: May 1, 2020 3:18 PM IST, and UPDATED: May 1, 2020 3:18 PM IST. It also lists LINKED ASSISTANTS (1): Customer Care. A context menu is open over this card, with the 'View API Details' option highlighted. A red '1' is placed above the menu, and a red '2' is placed on the 'View API Details' button. To the right, another skill card is visible: 'My first skill' with similar details.

The Skill ID and API Key is to be noted.

The screenshot shows the 'Skill details' page for the 'Customer Care Sample Skill'. Under the 'Skill name' section, it shows 'Skill name: Customer Care Sample Skill' and 'Skill ID: 617d5db3-c85f-47b5-8f90-35db631250aa' (which is highlighted with a red box). Below that, it shows 'Legacy v1 workspace URL: https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/f79fec49-933c-46e0-b562-1d2bc9f497b5/v1/workspaces/617d5db3-c85f-47b5-8f90-35db631250aa/message'. Under the 'Service credentials' section, it shows 'Service credentials name: Auto-generated service credentials' and 'API key: JNnaA2SthfUKDFm50ct6g6D0_cWAmw69V6yrbdk_UIpN' (which is highlighted with a red box).

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 dashboard with the Watson Assistant resource selected. A red number '1' highlights the 'Service credentials' tab in the left sidebar. The main area displays a table of service credentials, with one row selected. A red box highlights the 'url' field, which contains the URL: `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 number '1' is placed over the search bar containing 'node-red'. A red box highlights the first search result, the 'Node-RED App' tile, which is described as a 'Developer Tools' under the 'Software' category. A red number '2' is placed over this tile.

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 'IBM Cloud' and a search bar. Below the navigation is a breadcrumb trail: Catalog / Create app / Node-RED. The main content area has tabs for 'About' and 'Create'. A large red number '1' is overlaid on the 'Create' tab. The 'About' tab section includes 'Details' (Author: IBM, Updated: 2/11/2020, Type: Starter kit), 'Source code' (GitHub link), and 'Helpful links' (Tutorial link). The 'Create' tab section includes 'Overview' (describing the pre-configured Node-RED application), 'This starter kit will help you' (with three bullet points: generate an application with Node-RED, generate an application with files for deploying to Cloud Foundry or a DevOps Pipeline, and connect to provisioned services), and 'What's included?' (listing 'Cloudant' with a 'View docs' link). A blue 'Get started' button is at the bottom. On the right side, there are '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, indicated by a red number '1' over the tab. The form fields include: 'App name' (Node RED KVHBI), 'Resource group' (Default), 'Tags' (Examples: env:dev, version-1), and 'Platform' (Node.js). Below the form is a 'Service details' section. On the right side, there are 'ASK A QUESTION' and 'FEEDBACK' buttons.

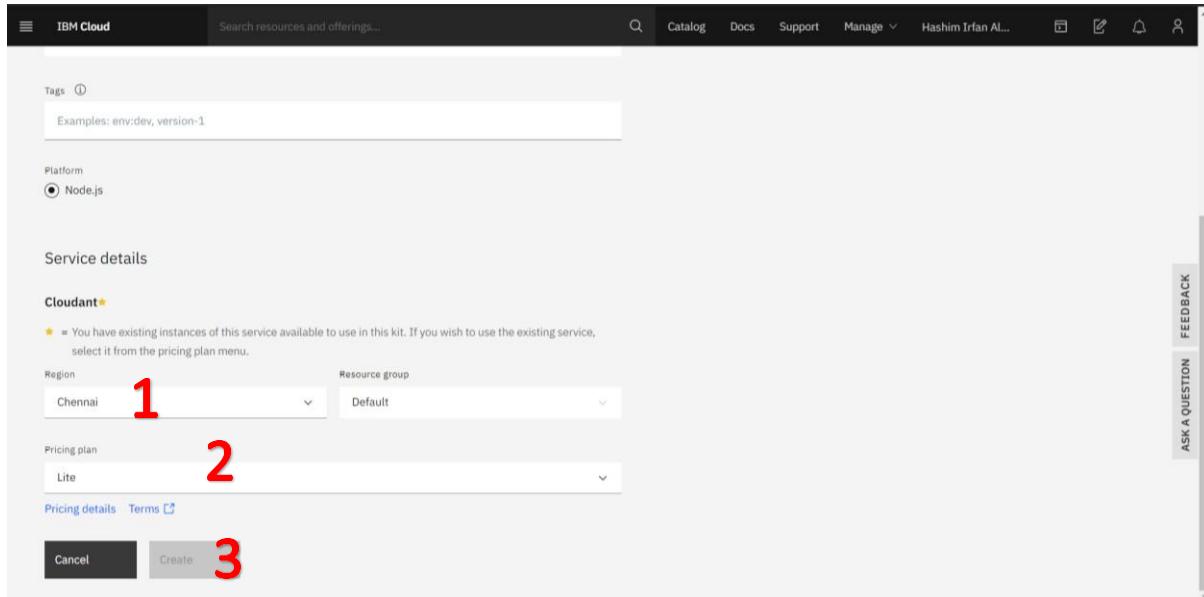
Now you need to configure the Node-RED Starter application.

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.

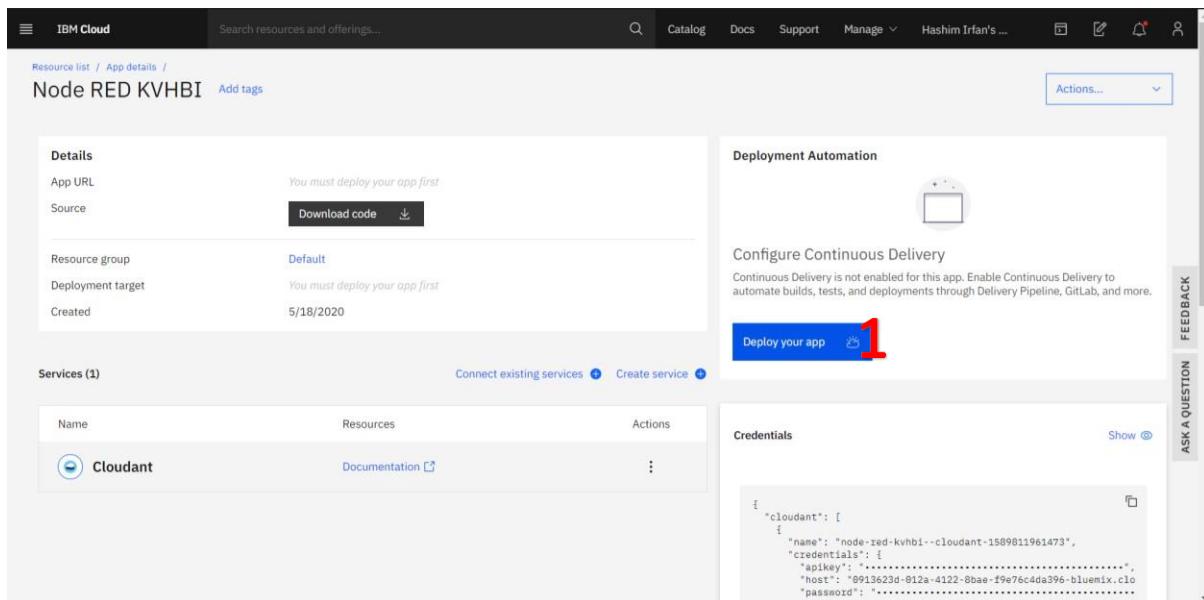
The Node-RED Starter application requires an instance of the Cloudant database service to store your application flow configuration. To do this,

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 IBM Cloud interface for creating an app. The top navigation bar includes 'IBM Cloud', 'Search resources and offerings...', 'Catalog', 'Docs', 'Support', 'Manage', and 'Hashim Irfan's ...'. Below the navigation is a 'Resource list / App details' section for 'Node RED KVHBI'. Two tabs are visible: 'Select the deployment target' (selected) and 'Configure the DevOps toolchain'. The 'Deployment Automation' section contains a 'Deployment target' card for 'Cloud Foundry IBM' and an 'IBM Cloud API key' input field. A red box highlights the 'New' button next to the input field, labeled '1'. To the right, a sidebar titled 'Getting started with apps' provides instructions for selecting a deployment target, mentioning Cloud Foundry as the premier PaaS. It also lists steps for account creation and region selection.

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 continuation of the app creation process. The top navigation bar and resource list are identical to the previous screenshot. The 'Configure the DevOps toolchain' tab is selected. The 'Deployment Automation' section shows the 'Number of instances' set to 1, the 'Memory allocation per instance' slider at 256 MB (labeled '1'), the 'Region' set to 'London' (labeled '2'), and the 'Domain' set to 'eu-gb.mybluemix.net'. A red box highlights the 'Next' button, labeled '3'. The right sidebar continues the deployment automation steps, including account creation and region selection.

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

The screenshot shows the 'Configure the DevOps toolchain' step. It includes fields for 'DevOps toolchain name' (NodeREDKVHBI) and 'Region' (Dallas). A numbered callout '1' points to the 'Dallas' option in the dropdown. A numbered callout '2' points to the blue 'Create' button at the bottom.

This will take you back to the application details page.

The screenshot shows the application details page. It lists basic information like App URL (disabled), Source (Download code), Resource group (Default), Deployment target (disabled), and Created date (5/18/2020). Under 'Services (1)', it shows a Cloudant service. The 'Deployment Automation' section has a 'Configure Continuous Delivery' button. A numbered callout '1' points to the 'Actions...' button in the top right. A numbered callout '2' points to the Cloudant service entry.

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.

Details

App URL: You must deploy your app first
Source: <https://us-south.git.cloud.ibm.com/hashimirfan99/NodeREDKVHBI>

Resource group: Default
Deployment target: You must deploy your app first
Created: 5/18/2020

Services (1)

Name	Resources	Actions
Cloudant	Documentation	⋮

Deployment Automation

Name: NodeREDKVHBI
Location: Dallas
Tool integrations:

Delivery Pipelines

Name: NodeREDKVHBI
Status: In progress
Last input: Last commit by IBM Cloud (1 minute ago)
Clone from zip

Credentials

```
{
  "cloudant": [
    {
      "name": "node-red-kvhbi--cloudant-1589811961473",
      "credentials": {
        "apikey": "...",
        "host": "0013423d-012a-4122-9bae-f9e74c4de204 bluemix.cl"
      }
    }
  ]
}
```

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.

Details

App URL: You must deploy your app first
Source: <https://us-south.git.cloud.ibm.com/hashimirfan99/NodeREDKVHBI>

Resource group: Default
Deployment target: You must deploy your app first
Created: 5/18/2020

Services (1)

Name	Resources	Actions
Cloudant	Documentation	⋮

Deployment Automation

Name: NodeREDKVHBI
Location: Dallas
Tool integrations:

Delivery Pipelines

Name: NodeREDKVHBI
Status: Success
Last input: Last commit by IBM Cloud (4 minutes ago)
Clone from zip

Credentials

```
{
  "cloudant": [
    {
      "name": "node-red-kvhbi--cloudant-1589811961473",
      "credentials": {
        "apikey": "...",
        "host": "0013423d-012a-4122-9bae-f9e74c4de204 bluemix.cl"
      }
    }
  ]
}
```

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 navigation icons and a tree view of resources. The main area displays a table with columns: Name, Group, Location, Offering, Status, and Tags. A single row is selected, highlighted with a red box labeled '2'. The row details are: Name - Node RED KVHBI, Group - hashimirfan99@gmail.com / dev, Location - London, Offering - SDK for Node.js™, Status - Started, and Tags - none. A red box labeled '1' highlights the 'Cloud Foundry apps' section in the sidebar.

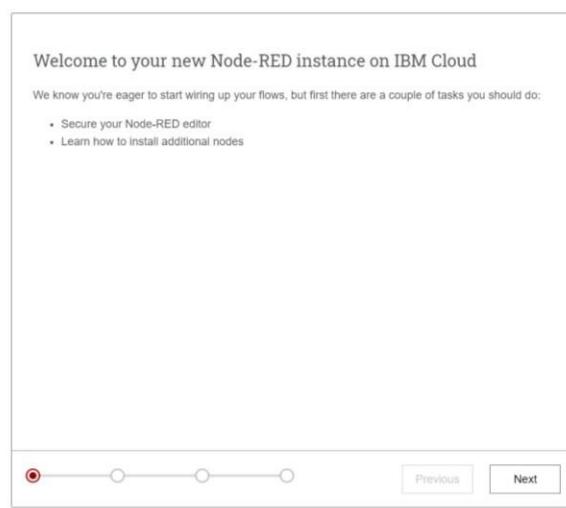
Click on this Cloud Foundry app entry to go to your deployed application's details page.

The screenshot shows the IBM Cloud App Details page for the 'Node RED KVHBI' application. The top bar includes the application name, status (Running), and links to Visit App URL and Add tags. The main content area has tabs for Overview, Runtime, Connections, and Logs. The Overview tab is selected, showing the following data:

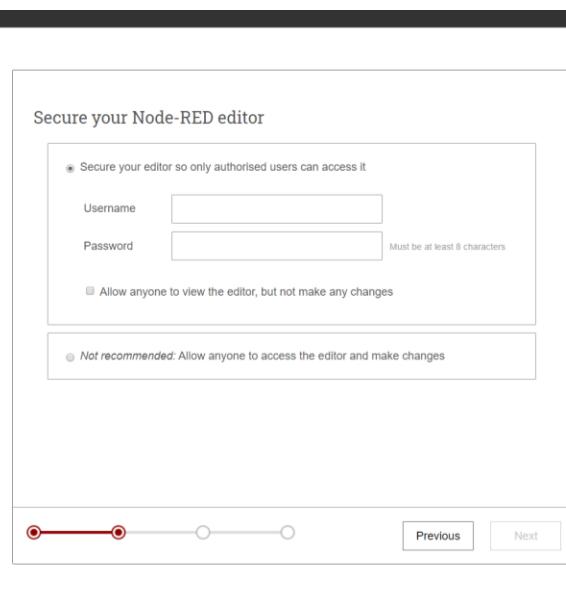
- Instances**: Health 100%, 1/1 instance(s) are running.
- Runtime**: SDK for Node.js™, Total MB allocation 256, 1.75 GB still available (128 Free, 128 Used).
- Runtime cost**: \$0.00 (Current charges for billing period May 1, 2020 - May 31, 2020) and Estimated total for billing period May 1, 2020 - May 31, 2020.
- Connections**: 1 connection listed.

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.

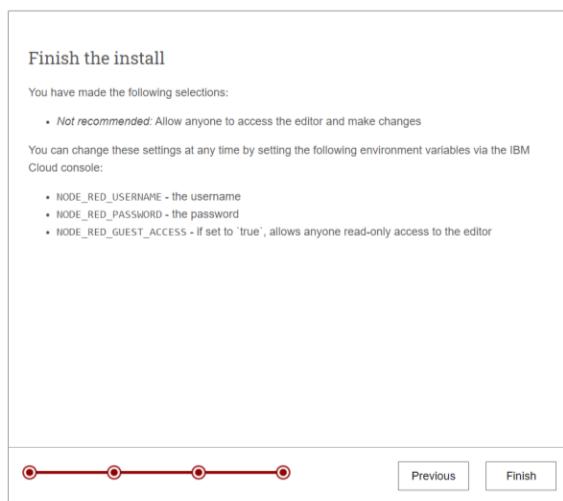
For more information, follow this tutorial on [IBM Developer](#).



Previous

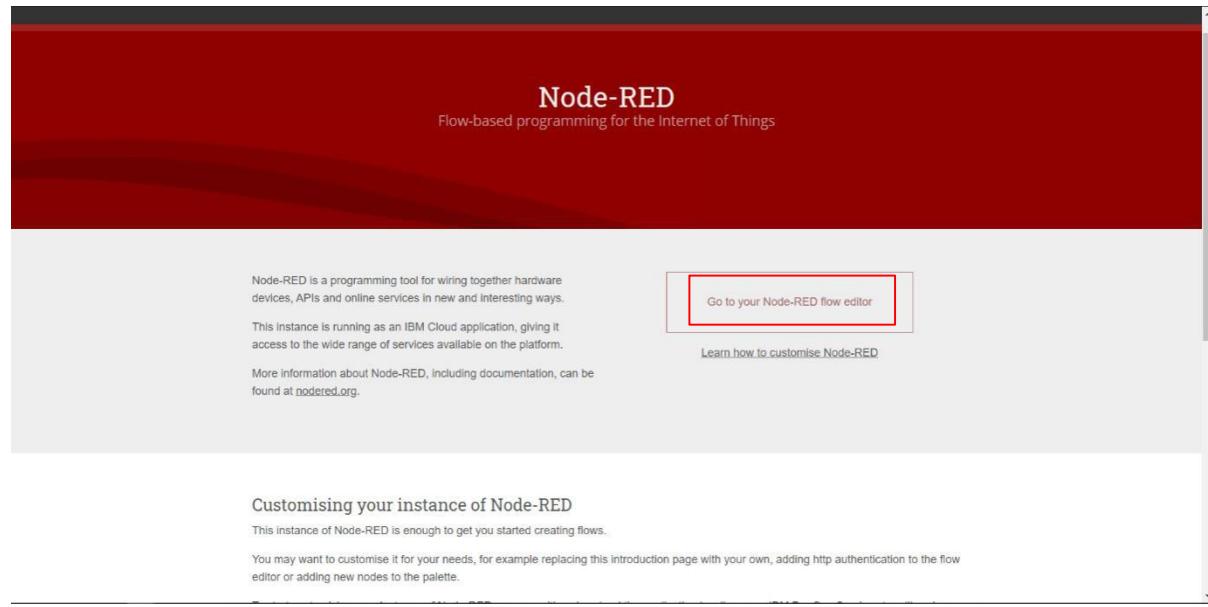
Next

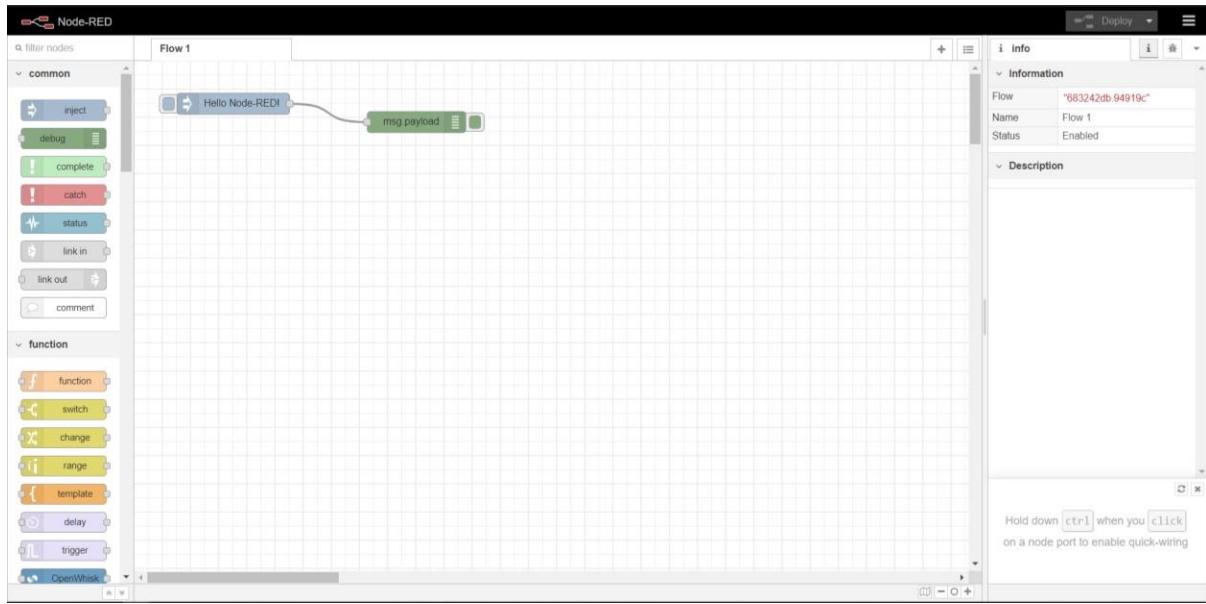
Click Next.



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.

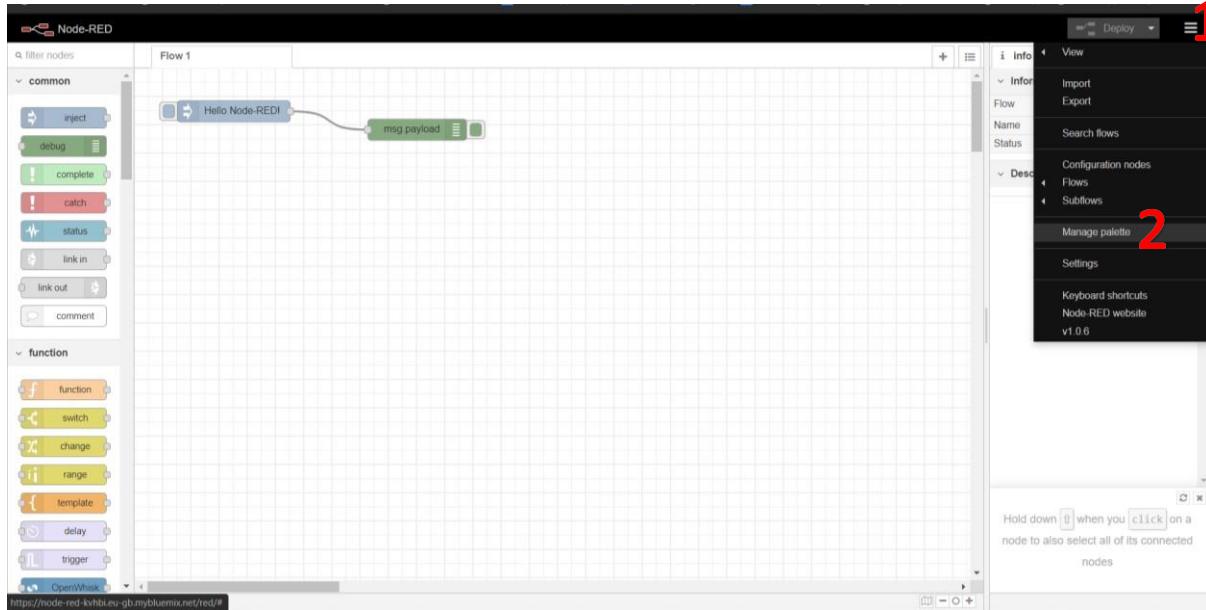




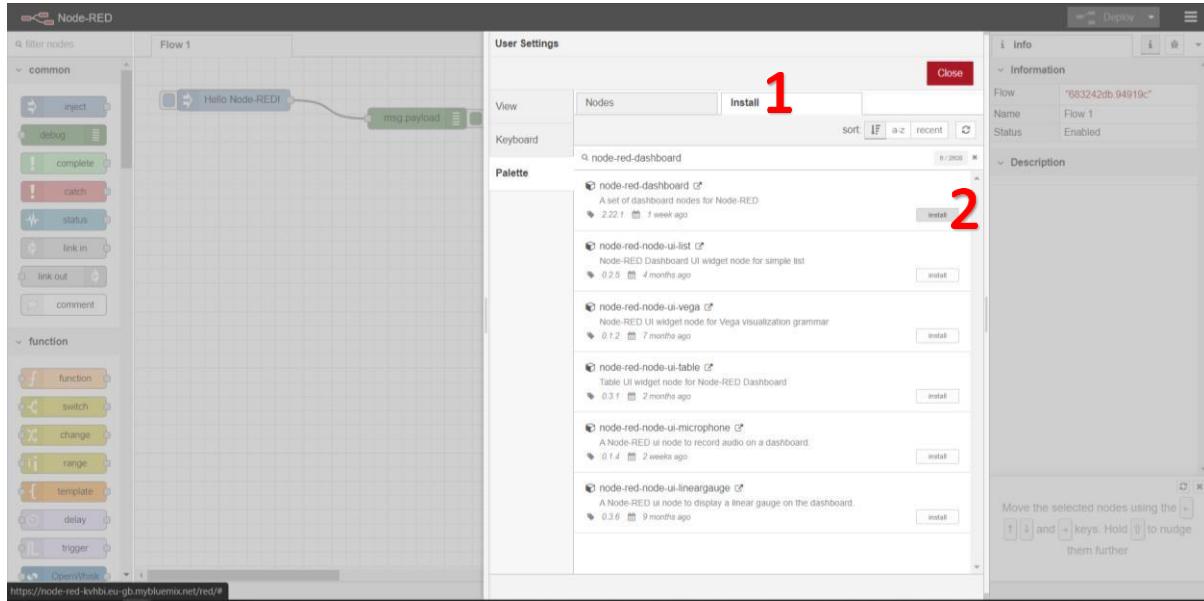
The Node-RED editor opens showing the default flow.

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]

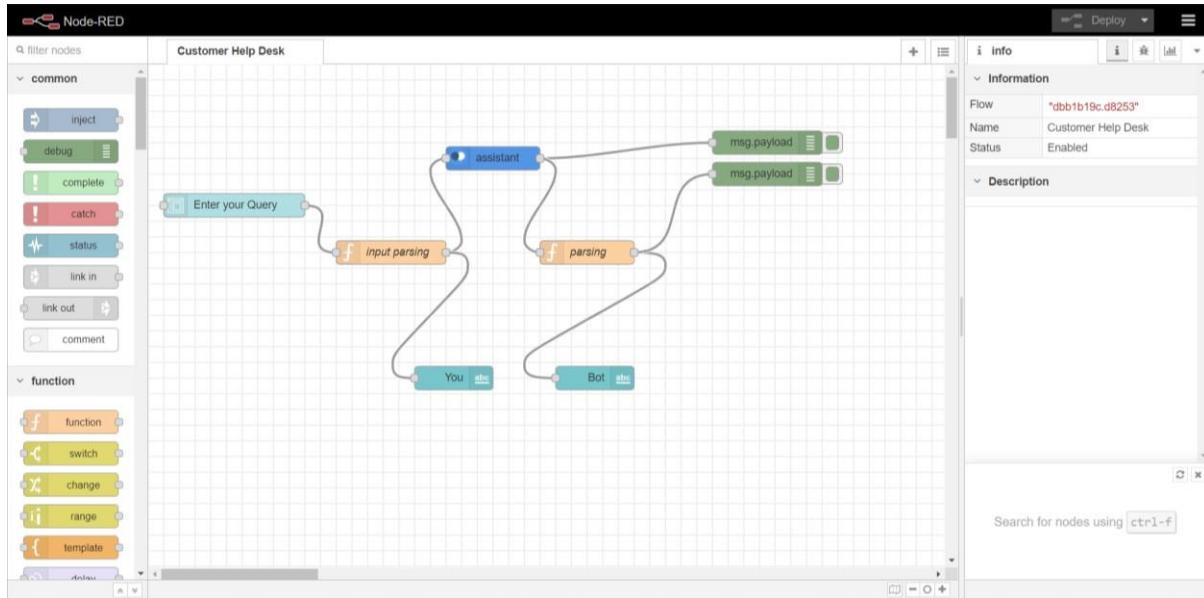


Go to Install Tab [1] and search for node-red-dashboard and Install [2] it.



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

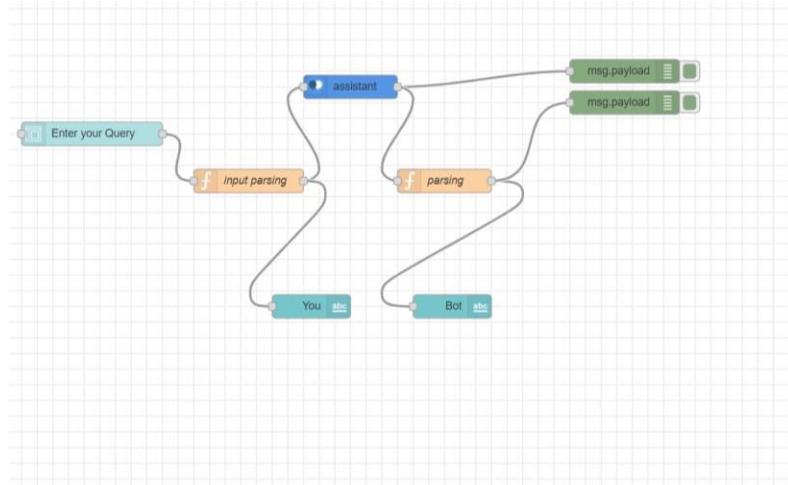
You
how to adjust screen brightness

bot

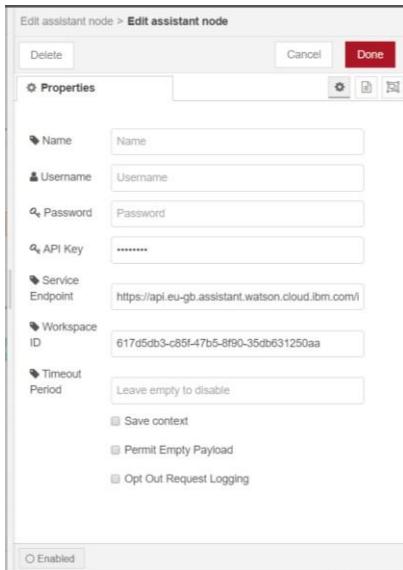
You can customize the brightness of your ecobee3's screen. The brightness for both the active and standby screens can be configured independently. You can also configure the screen to automatically sleep (i.e. turn off) whenever your ecobee3 enters the Sleep activity period. For example, if your thermostat 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:

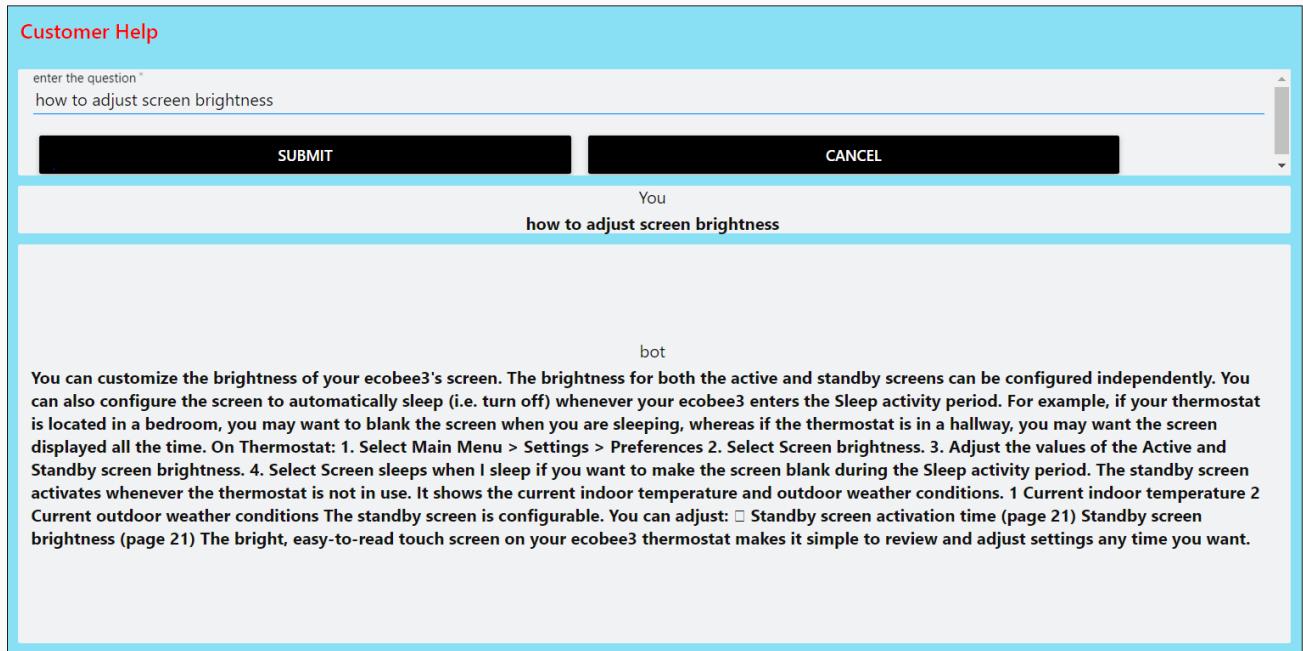
```
msg.payload.text="";  
if(msg.payload.context.webhook_result_1){  
    for(var i in msg.payload.context.webhook_result_1.results){  
  
        msg.payload.text=msg.payload.text+"\n"+msg.payload.context.webhook_result_1.re  
sults[i].text;  
    }  
    msg.payload=msg.payload.text;  
}
```

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

6.

RESULT

Chatbot



Chatbot Link:

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

Explanation Video Link:

<https://youtu.be/0woSMeowXq8>

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": [  
    }
```



```
"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

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/watch?v=-yniuX-Poyw&feature=youtu.be>

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

THE END