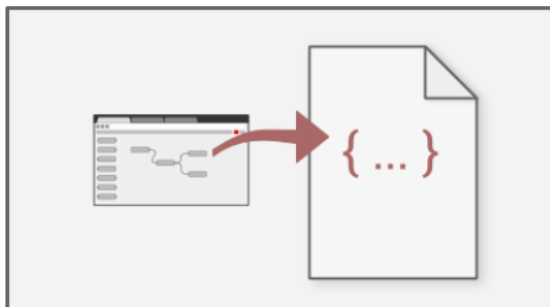


Built on Node.js

The light-weight runtime is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

With over 225,000 modules in Node's package repository, it is easy to extend the range of palette nodes to add new capabilities.

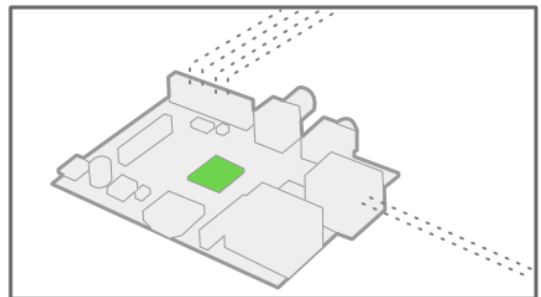


Browser-based flow editing

Node-RED provides a browser-based flow editor that makes it easy to wire together flows using the wide range of nodes in the palette. Flows can be then deployed to the runtime in a single-click.

JavaScript functions can be created within the editor using a rich text editor.

A built-in library allows you to save useful functions, templates or flows for re-use.



Social Development

The flows created in Node-RED are stored using JSON which can be easily imported and exported for sharing with others.

An online flow library allows you to share your best flows with the world.

1. JSON (JavaScript Object Notation)

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

1. JSON objects are surrounded by curly braces {}.
2. JSON objects are written in key/value pairs.
3. Keys must be strings, and values must be a valid JSON data type (string, number, object, array, Boolean or null).
4. Keys and values are separated by a colon i.e. ":".
5. Each key/value pair is separated by a comma i.e. ",".

Example of JSON object

```
1 {  
2   "key_string" : "XYZ",  
3   "key_num"    : 1,  
4   "Key_bool"   : true,  
5   "key_null"   : null,  
6   "key_arr"    : ["ABC",1,true,false,null]  
7 }
```

JSON in JavaScript

```
1 //Key can be written without double quotes  
2 var msg = {  
3   // Key      Value  
4   key_string : "XYZ",           //string  
5   key_num    : 1,               //number  
6   Key_bool   : true,           //Boolean  
7   key_null   : null,           //null  
8   key_arr    : ["ABC",1,true,false,null] //array  
9 };  
10  
11 //nested json object  
12 var a_msg = {  
13   name:"John",  
14   age:30,  
15   cars: {  
16     car1:"Ford",  
17     car2:"BMW",  
18     car3:"Fiat"  
19   }  
20 };
```

In JavaScript, object can be accessed through following methods: -

```
1 var x = object_name.key_name;  
2 //or  
3 var y = object_name["key_name"];  
4  
5 //for array  
6 var z = object_name.key_name[index_number]; //index starting from 0  
7
```

```

8 //for nested object
9 var a = object_name.sub_object_name.key_name;
10 var b = object_name.sub_object_name["key_name"];
11
12 //Example
13 var x = msg.key_string = "ABC";
14 var y = msg["key_string"] = "ABC";
15 var z = msg.key_arr[3]=false;
16 //for nested object
17 var a = a_msg.cars.car1="Honda";
18 var b = a_msg.cars["car3"]="Honda";

```

Modification of JSON object can be done through following methods: -

```

1 object_name.key_name = "XYZ";
2 object_name["key_name"] = "XYZ";
3 object_name.key_name[index_number] =2; //assign value at index
4
5 //for nested object
6 object_name.sub_object_name.key_name = 2;
7 object_name.sub_object_name["key_name"] =1;
8
9 //example
10
11 var carName = "Hyundai"
12 a_msg.cars.car1 = "the car is"+carName;           //return "the car is Hyundai"
13 a_msg.cars.car2 = carName+"is the car";           //return "Hyundai is the car"
14 a_msg.cars.car3 = "the"+carName+"car is good";    //return "the Hyundai car is good"

```

Print variables/objects in console: -

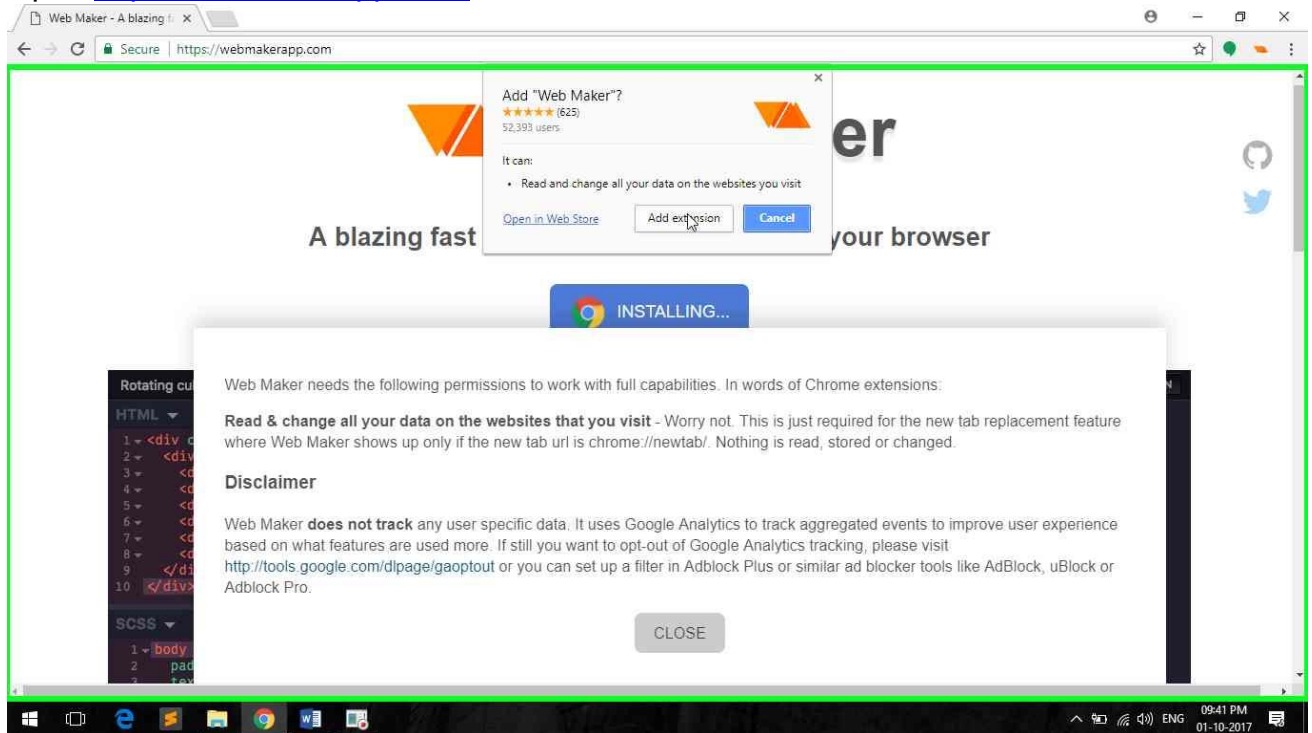
```

1 //print in console
2 console.log(var_name);           //replace var_name with variable name
3 console.log(JSON.stringify(object_name)); //replace object_name with object name
4
5 //Example
6 console.log(x);
7 console.log("hello"+name);
8 console.log(a_msg.name);
9 console.log(JSON.stringify(msg));

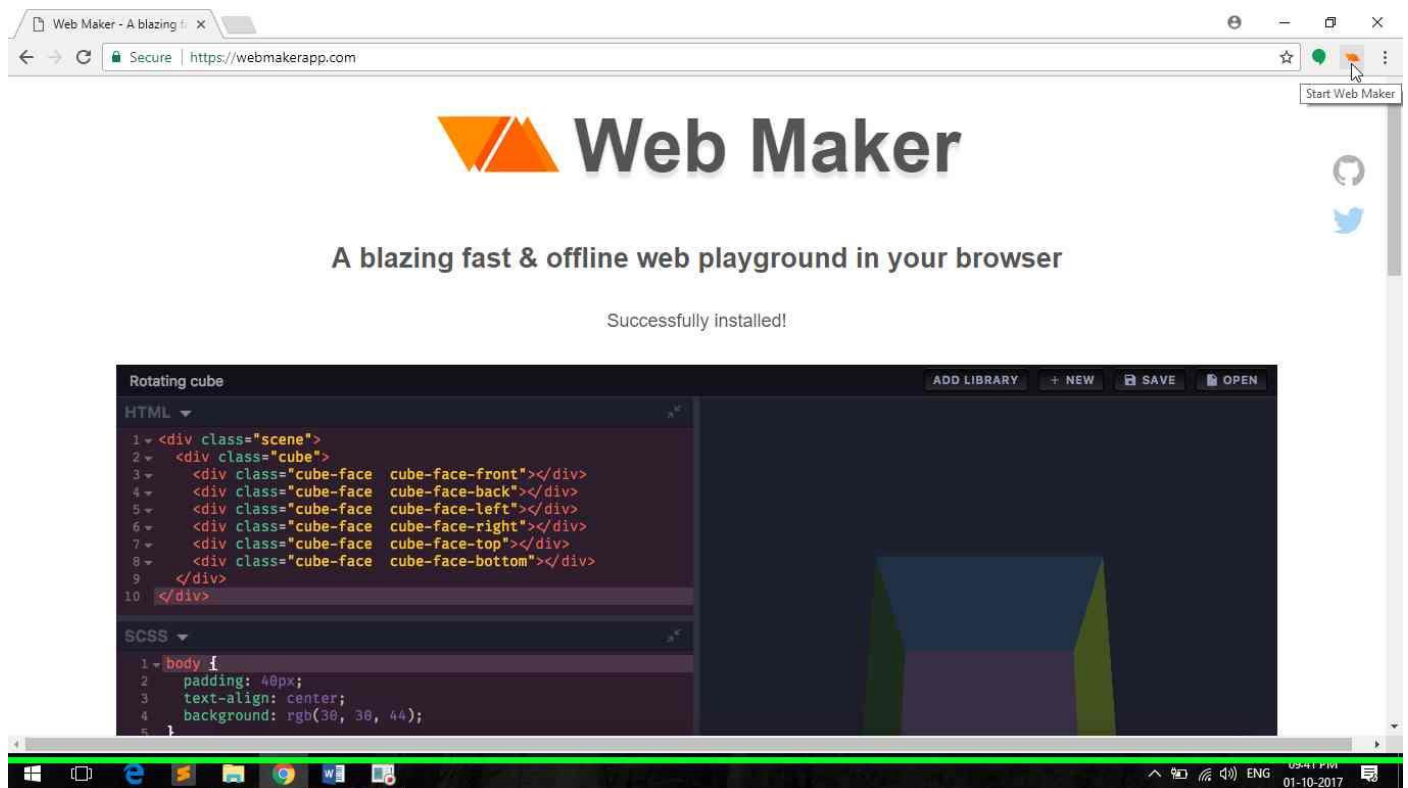
```

For testing above examples go through following steps: -

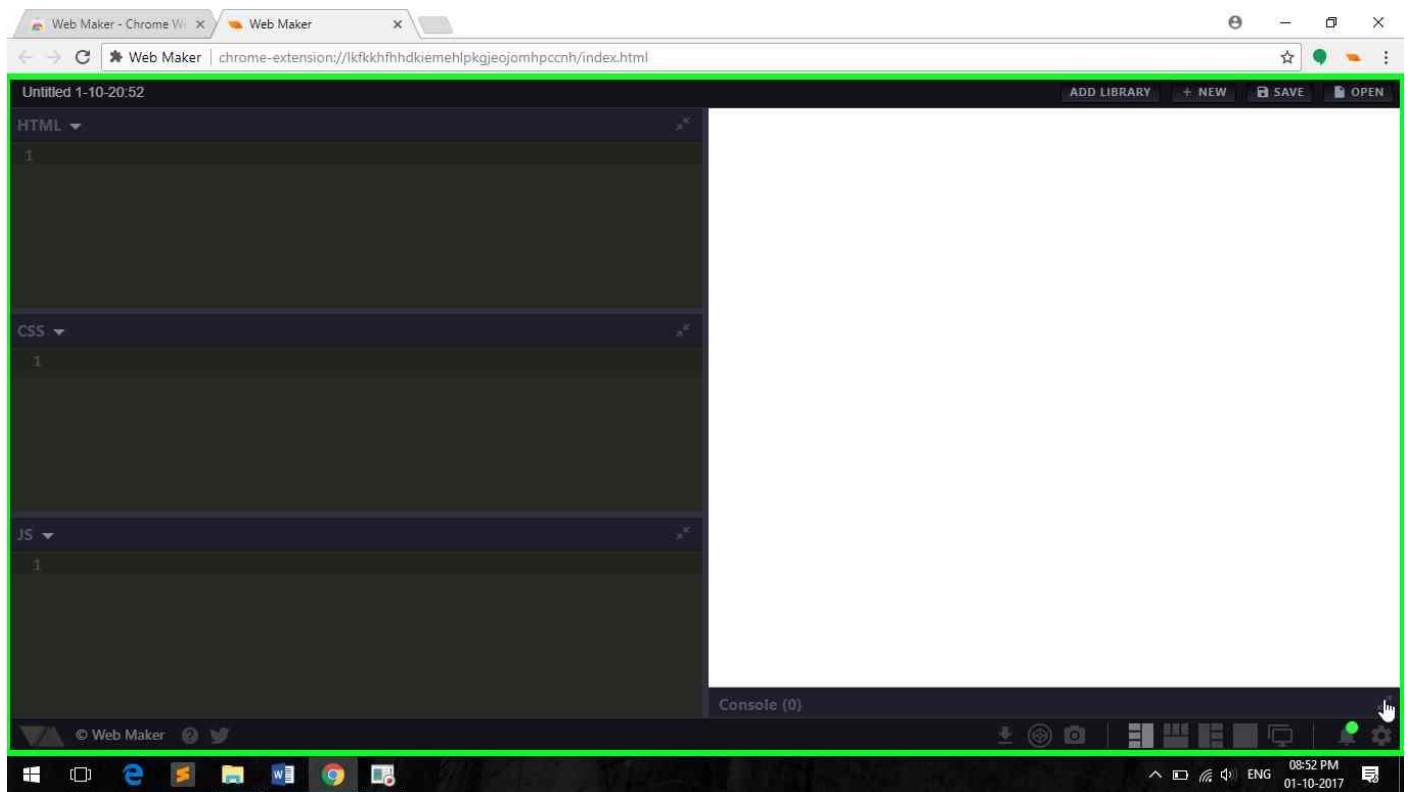
1. Open <https://webmakerapp.com/>



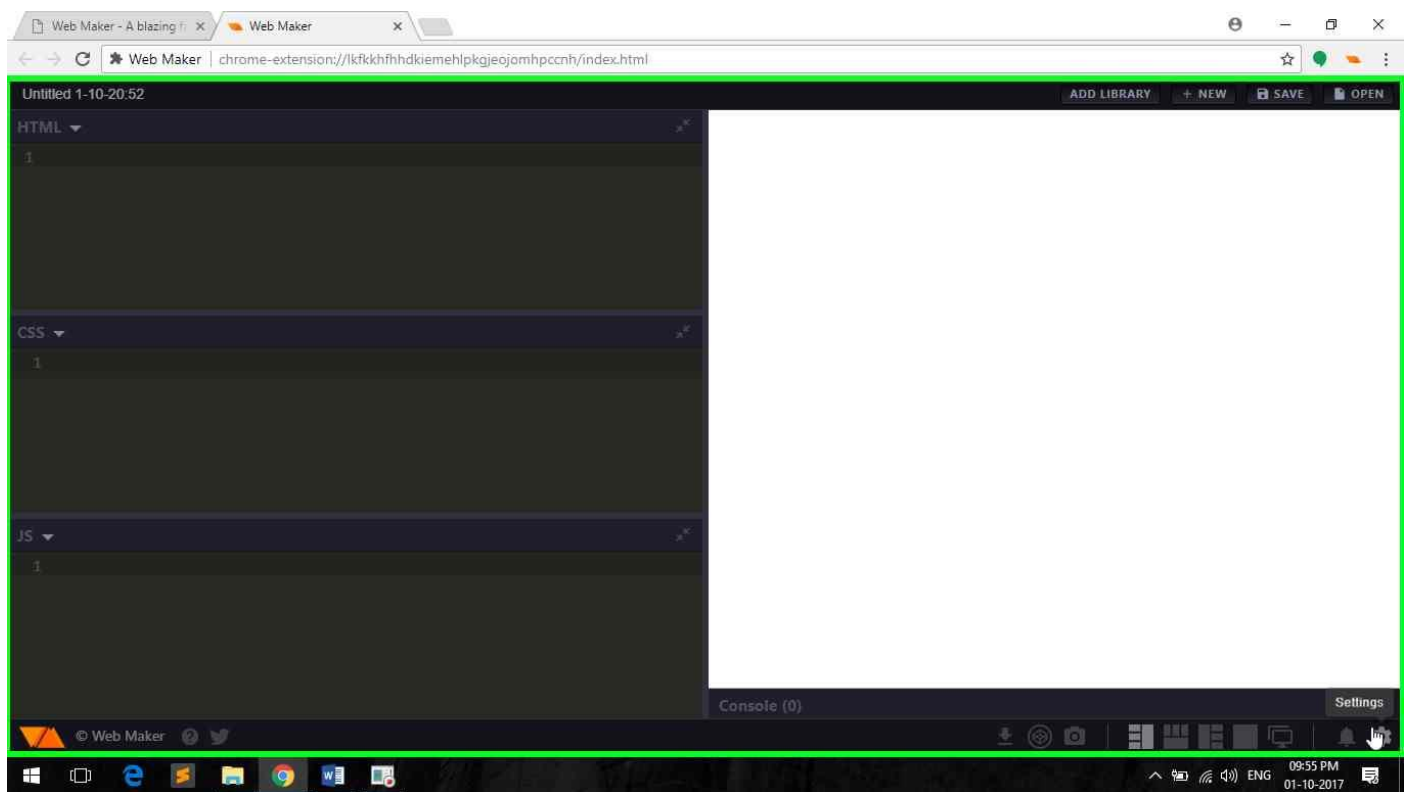
2. Click on **web maker** button navigator to start Web Maker.



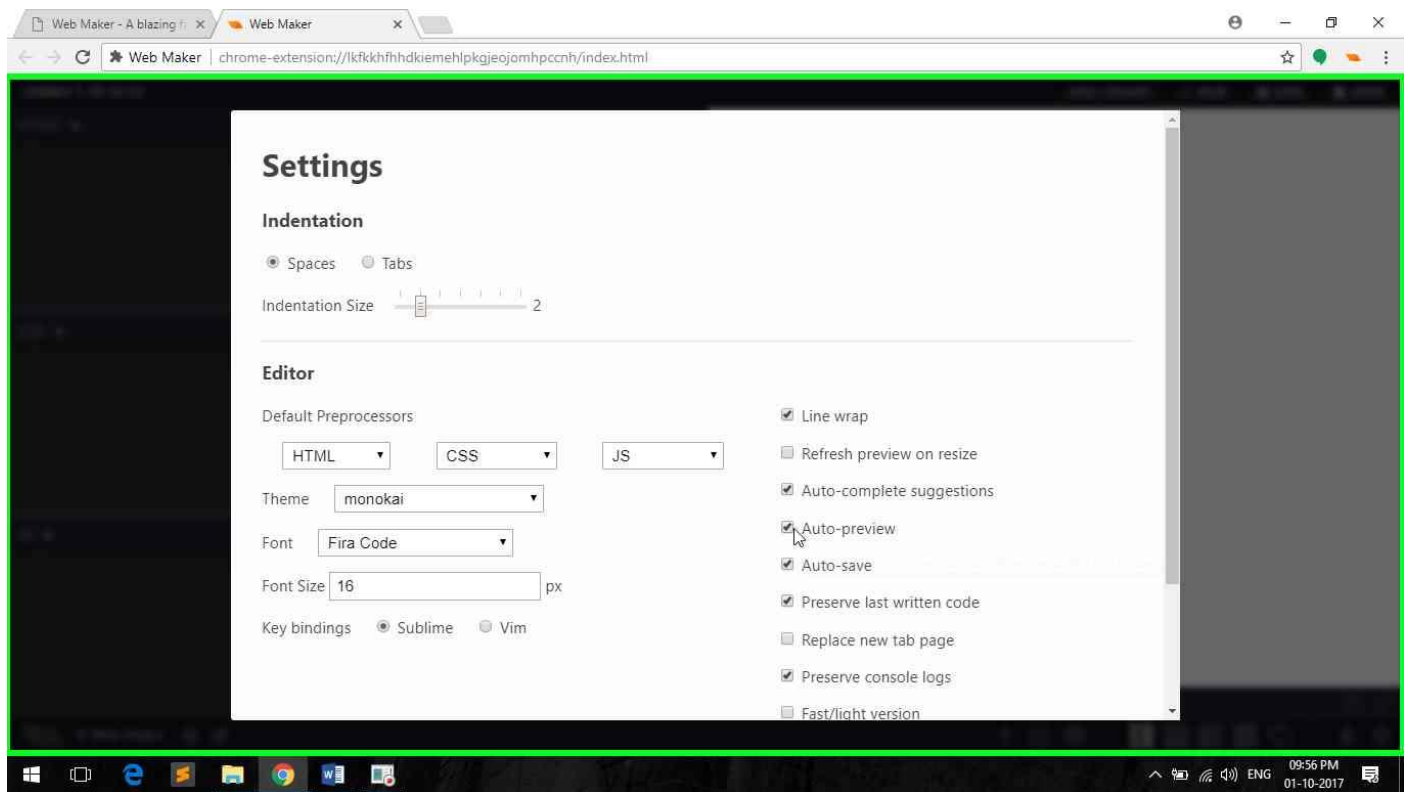
3. Double click on **console** to open console tab.



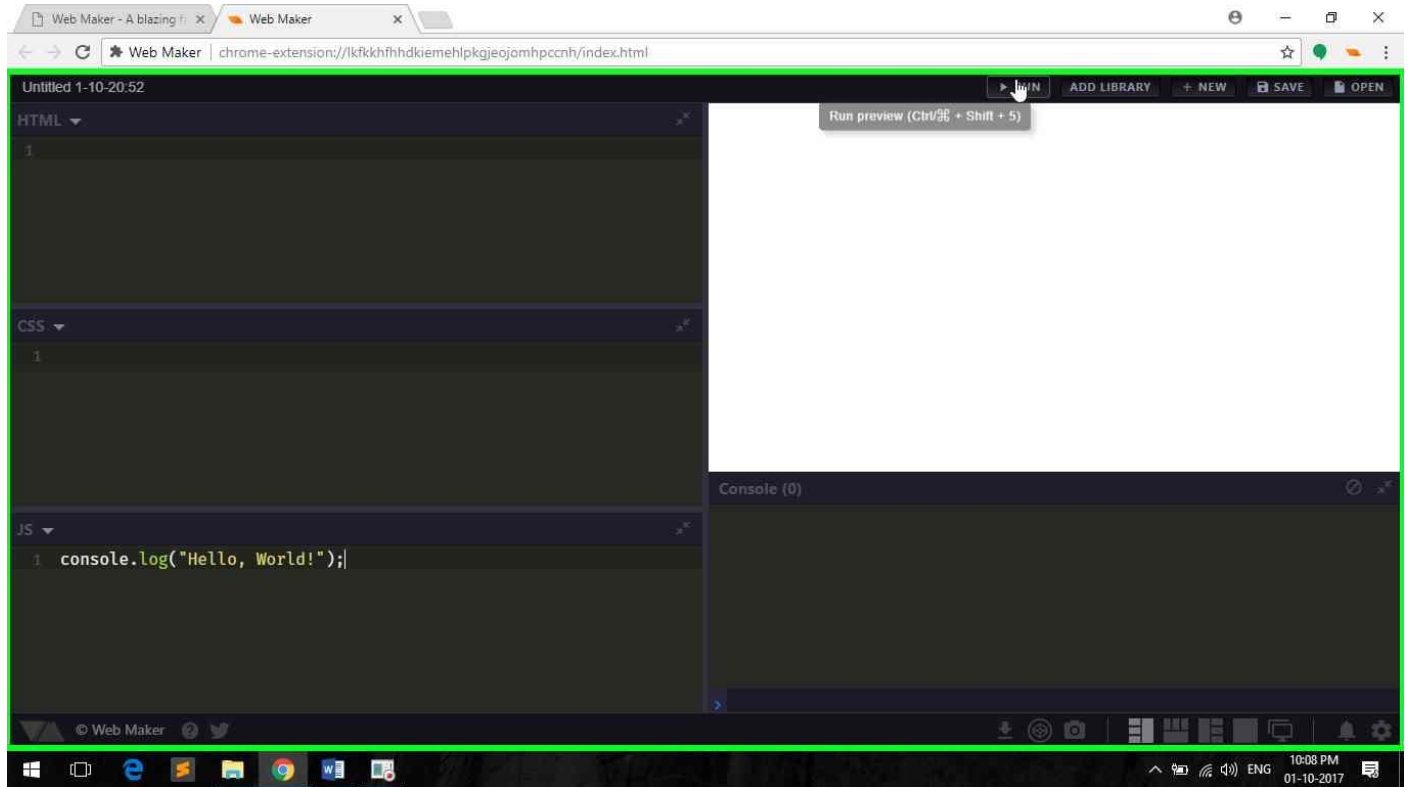
4. Click on **setting**.



5. Uncheck **Auto-preview**.



6. Write your code in **js** tab, click on **run** to test.

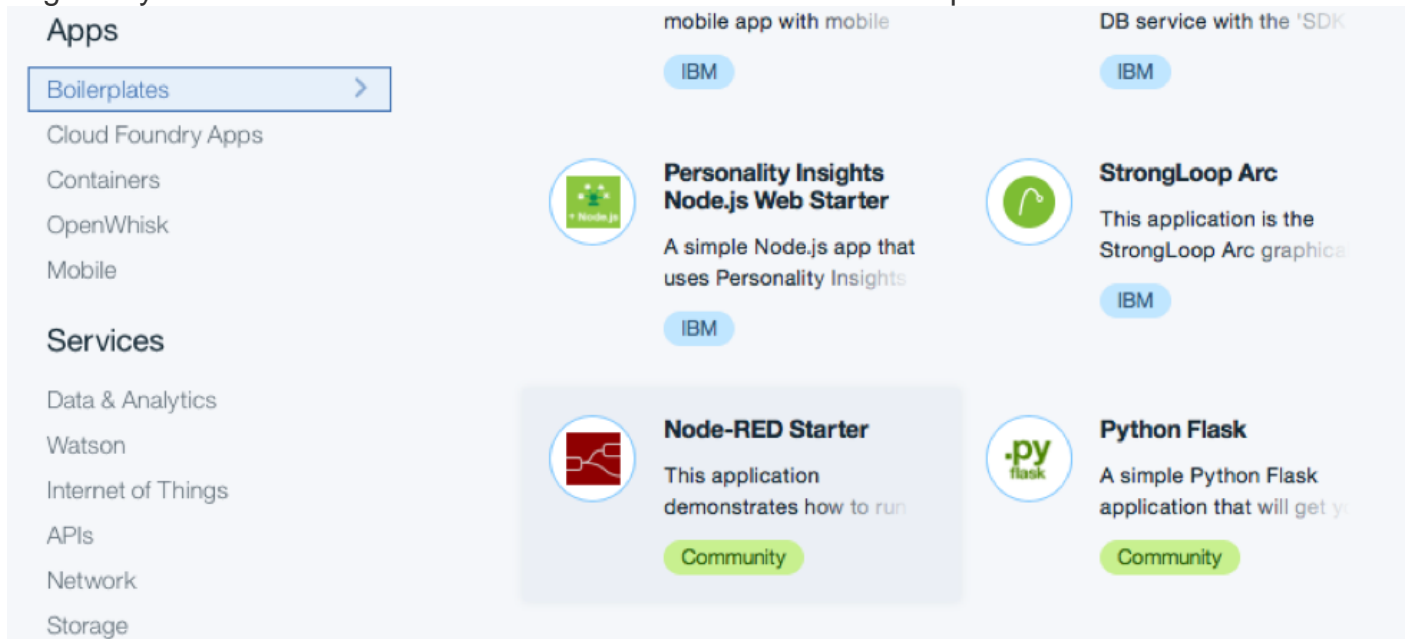


TRY NOW!!

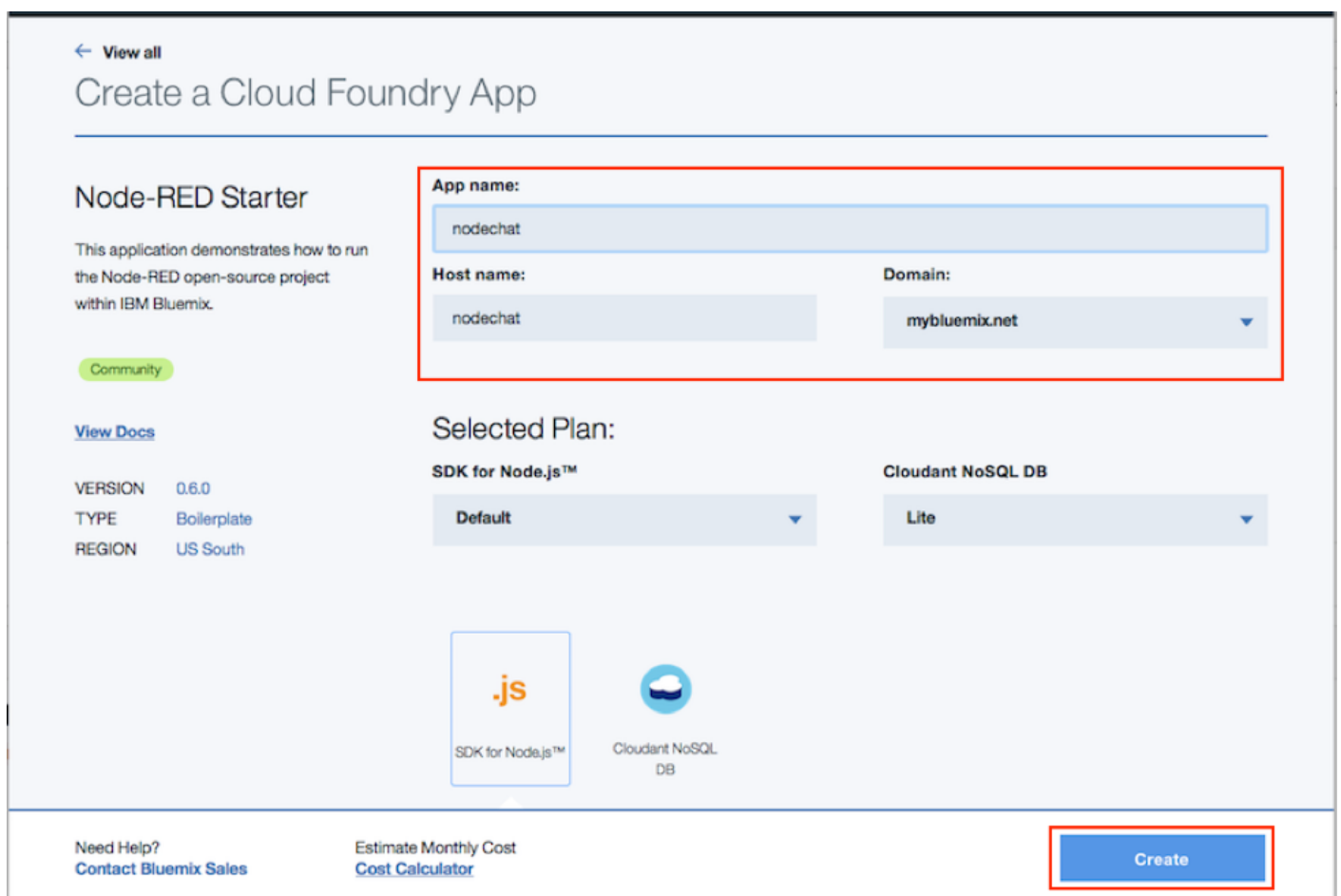
Play around.....

2. Introduction to Interface

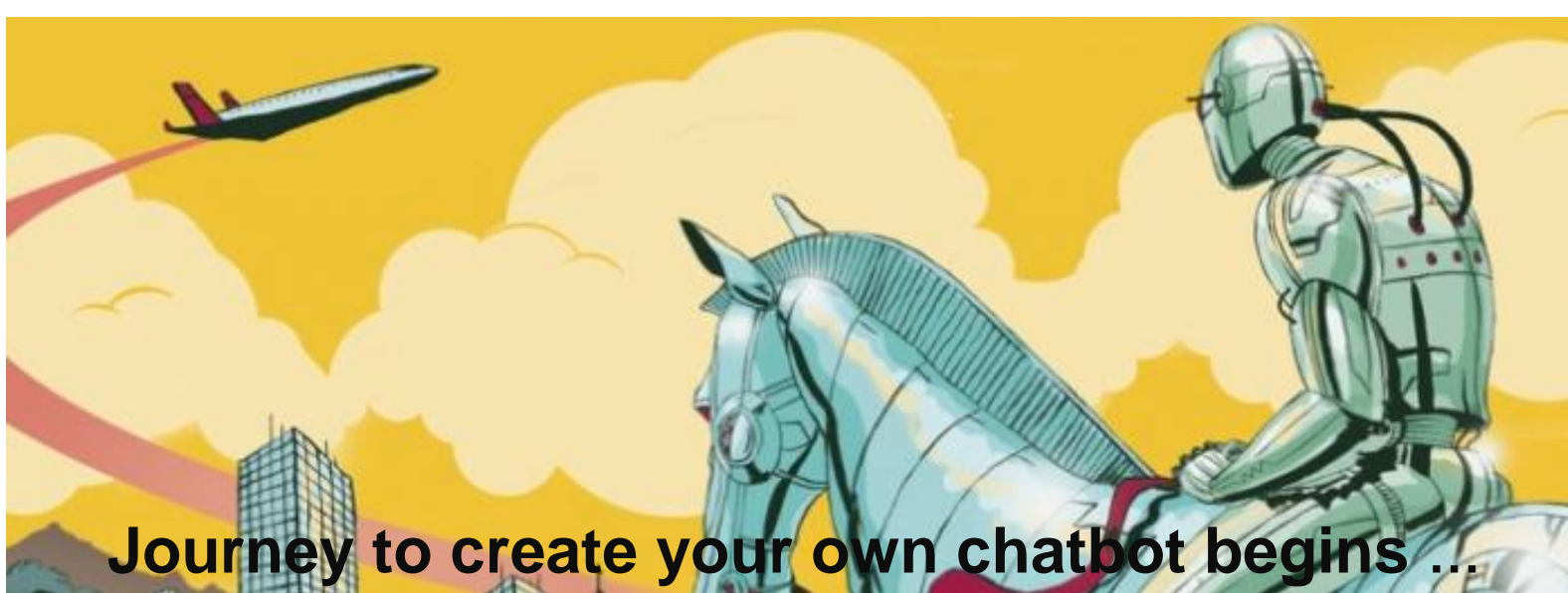
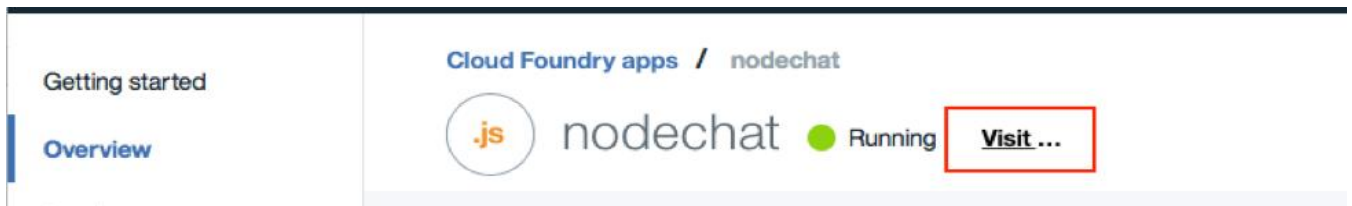
1. Log in to your Bluemix account and create a new Node-RED boilerplate.



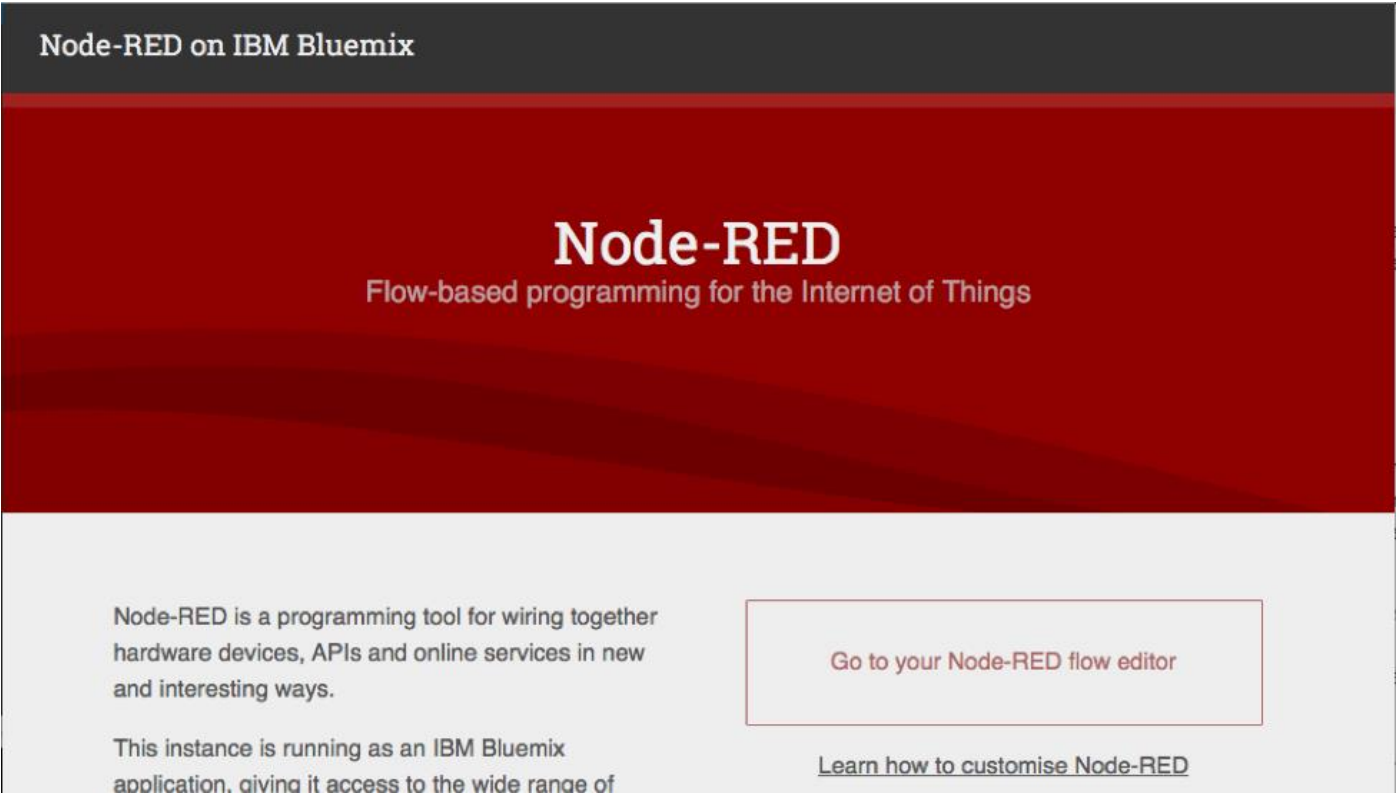
2. Give your application a name and click **CREATE**.



- From the Bluemix dashboard, navigate to the Overview page for your application, then click the **Visit...** link to launch Node-RED's main page. It can take a while for the application to start; the green circle and "Running" text will be visible when it's ready to try.



4. The first time you visit the site, you will be asked to do some. basic configuration. Once that's done, you can click **Go to your Node-RED flow editor** to open the flow editor.



5. Next, before we are able to use the app, there are some minor setup need to be done. It will ask for the Username and Password for the app. Enter the Username and Password accordingly.

Secure your Node-RED editor

☒ Secure your editor so only authorised users can access it

Username

my_username

Password

.....

strong

☐ Allow anyone to view the editor, but not make any changes

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

Previous

Next

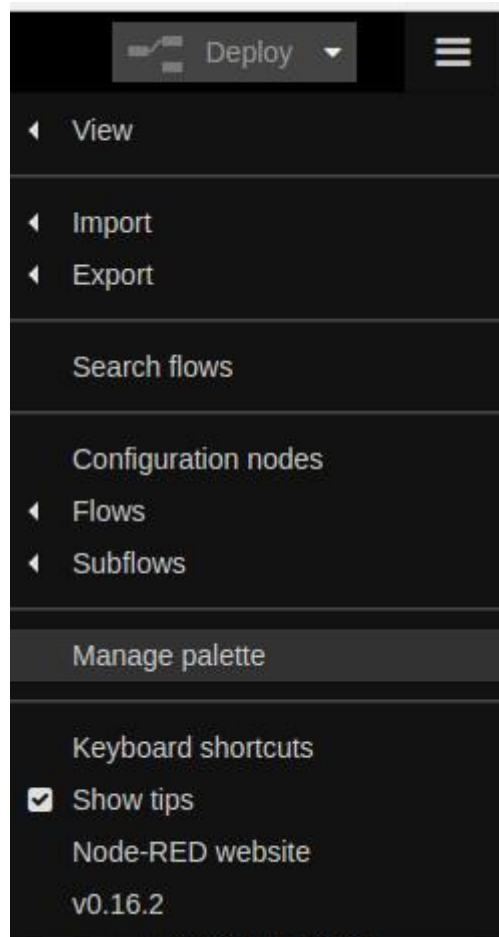
Click **Next** then **Finish**.

6. You should see a blank flow where you can start building your app. When using Node-RED we build our apps using this graphical editor interface to wire together the blocks we need. We can simply drag and drop the blocks from the left menu into the workspace in the center of the screen. and connect them to create a new flow.

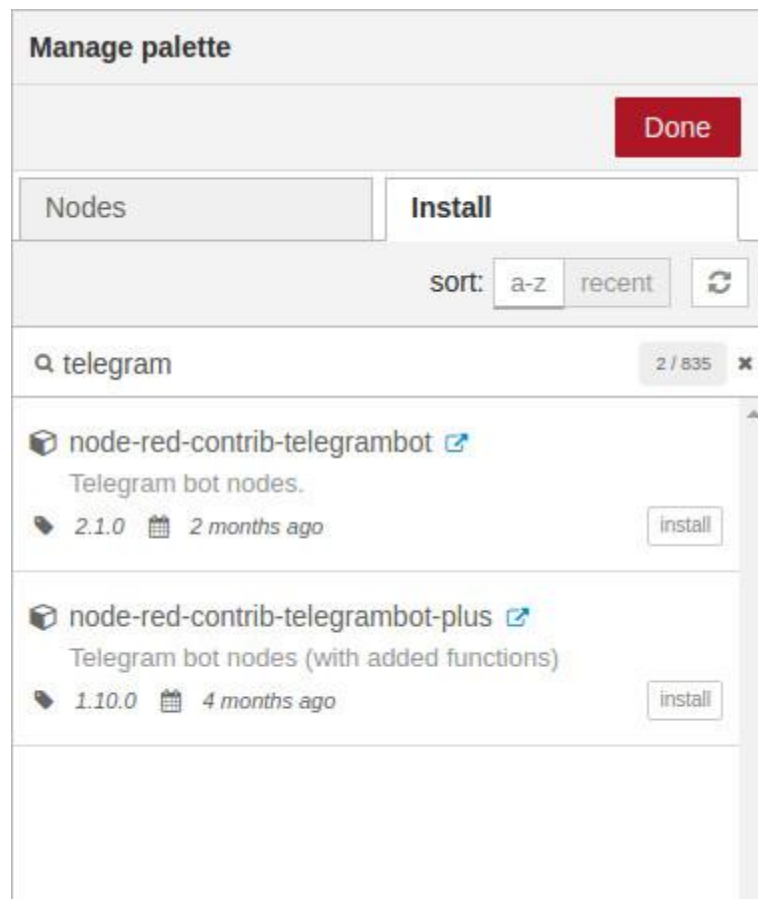


3. Install Telegram Nodes on Node-RED

1. Once the app has been started, click **View app** or go to the url of the app directly
2. Click on the **Go to your Node-RED flow editor** or add **/red** path to the url to view the Node-RED flow. For example:
 - i) `http://<your_subdomain>.mybluemix.net/red/`
3. Click on the top-right menu and select **Manage pallete**.



4. Click on the **Install** tab
5. Enter **telegram** on the **search module** textbox
6. Click **install** for **node-red-contrib-telegrambot**



7. Then click **Install** again to confirm
8. Once the node has been installed, click **Done**
 - i) Note: There will be 4 telegram nodes installed

One Step closer to create your own **Chatbot** ...

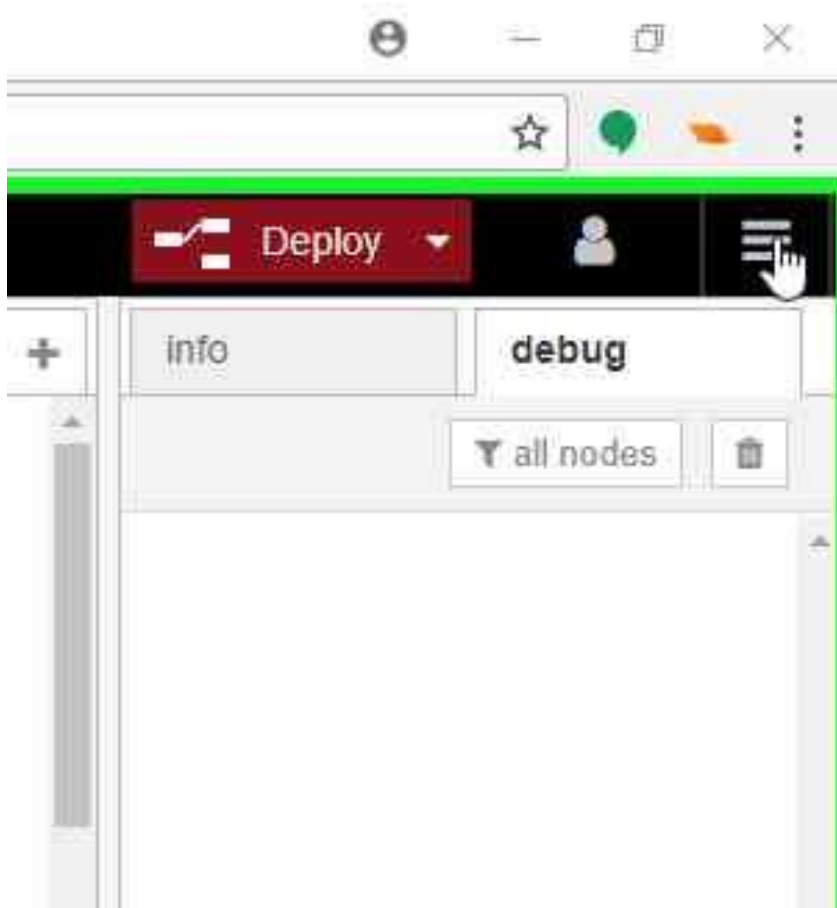


4. Importing/Exporting a flow

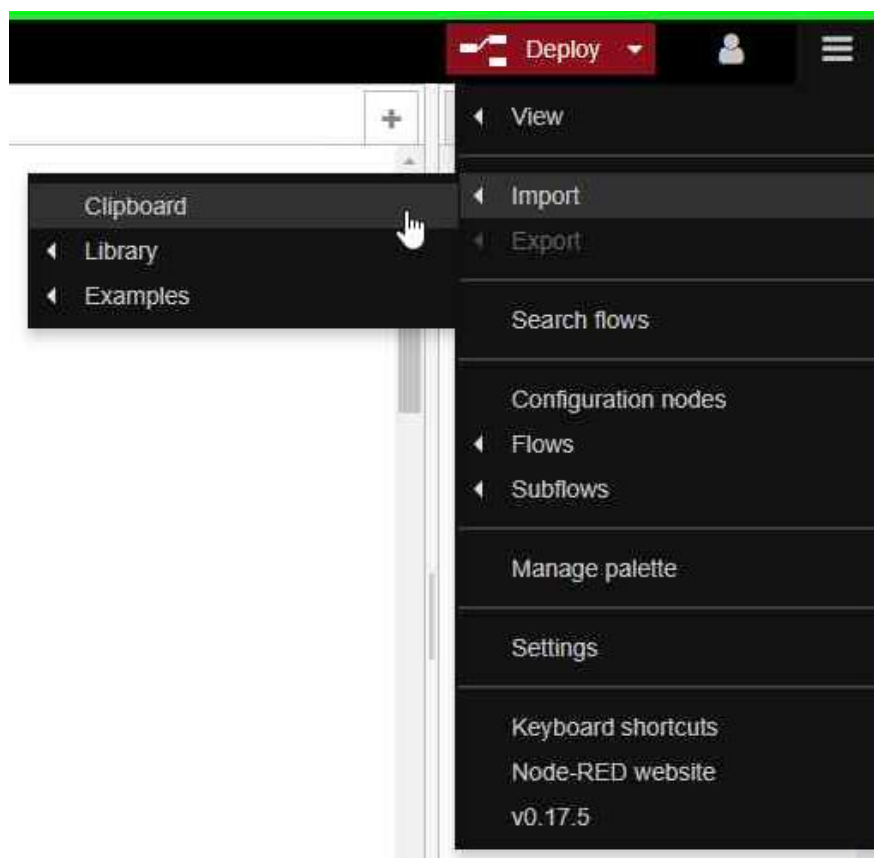
1. Copy below sample code .

```
[{"id": "392be441.ecc12c", "type": "http\nin", "z": "9ac42d9d.479cf", "name": "", "url": "/helloworld", "method": "get", "upload": false, "swaggerDoc": "", "x": 722.5000076293945, "y": 467.5000066757202, "wires": [{"id": "238921d3.cfccc"}]}, {"id": "3fba09ed.7d7d56", "type": "http\nresponse", "z": "9ac42d9d.479cf", "name": "", "statusCode": "", "headers": {}, "x": 1240.000015258789, "y": 465.0000066757202, "wires": []}, {"id": "238921d3.cfccc", "type": "template", "z": "9ac42d9d.479cf", "name": "html\npage", "field": "payload", "fieldType": "msg", "format": "handlebars", "syntax": "mustache", "template": "<html>\n<head>\n    <title>hello, world!</title>\n<style>\n    body{\nbackground: #272822;\n}\n\n\n.include{\n    color: #F92672;\n}\n\n\n.include2{\n    color:\n#F92672;\n}\n\n\n.text{\n    color: #fff;\n}\n\n\n.np{\n    display: inline;\n}\n\n\n.int{\n    color: #66D9EF;\n}\n\n\n\n.main{\n    color: #A6E22A;\n}\n\n\n\n.text2{\n    color:\n#E6DB74;\n}\n\n\n\n.div{\n    margin-top: 7%; \n    margin-right: auto;\n    margin-left:\n40%;\n}\n\n\n\n.div2{\n    margin-left: 30px;\n}\n\n\n\n.input-line{\n    width: 1px;\n    heigth:\n5px;\n    color: fff;\n}\n\n\n\n.nhr{\n    background-color: #fff;\n    color: #fff;\n    width:\n0.5px;\n    height: 15px;\n    margin-left: 8px;\n    margin-top: -17px;\n    -webkit-\nanimation: display 1s linear infinite;\n}\n\n\n\n@-webkit-keyframes display{\n    50%{opacity: 0;}\n}\n\n\n</style>\n</head>\n<body>\n<div class=\"div\">\n    <p\nclass=\"text\">#</p><p class=\"include\">include</p><p class=\"text2\">\n    &lt;iostream&gt;</p><br>\n        <p class=\"include2\">using</p><p class=\"text\">\nnamespace std;</p><br>\n        <p class=\"int\">int</p><p class=\"main\"> main</p><p\nclass=\"text\">( )</p><br>\n        <p class=\"text\"></p><br><div class=\"div2\"><p\nclass=\"text\"> cout <<</p><p class=\"text2\"> \"Hello, world!\"</p><p class=\"text\">\n    << endl;</p></div><p class=\"text\"></p><br>\n</div>\n</body>\n</html>\", \"output\": \"str\", \"x\": 960.0000114440918, \"y\": 467.5000066757202, \"wires\": [{"id": "3fba09ed.7d7d56"}]}]
```

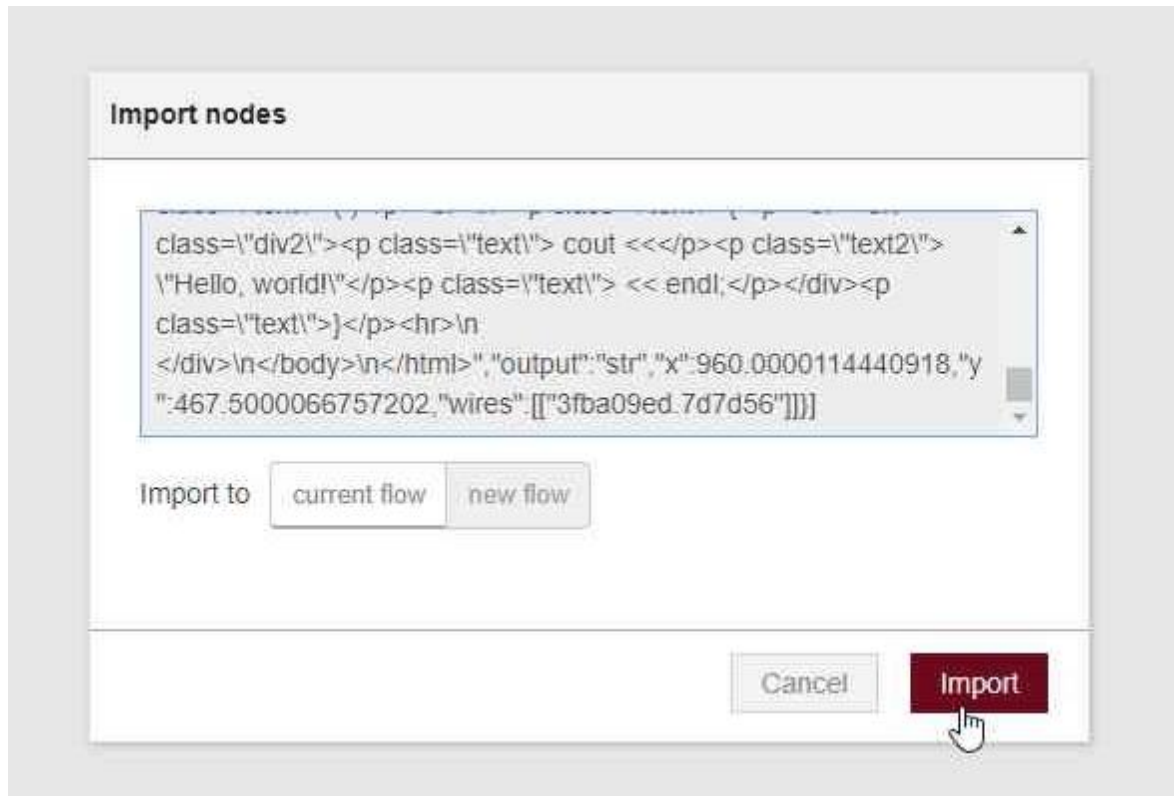
2. Click on **menu** on top-right side.



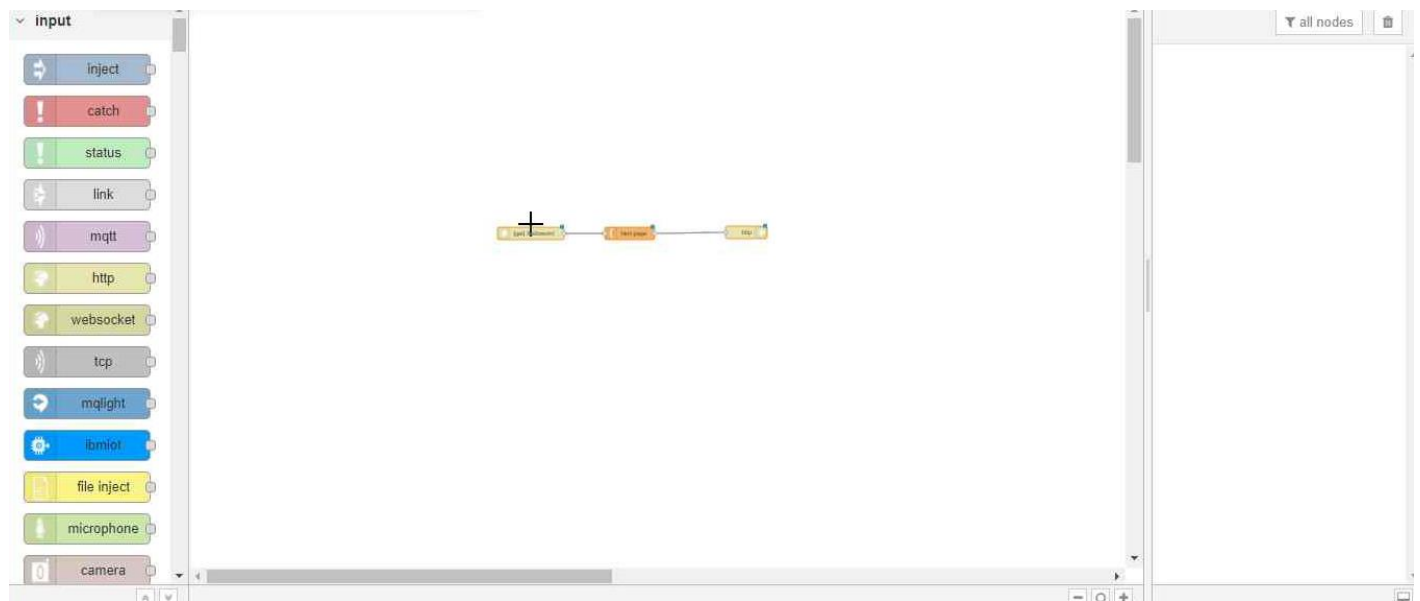
3. Select **import** and click on **clipboard**.



4. Paste (Ctrl + v) in text box and click on import.



5. **Left click** to drop nodes on flow.

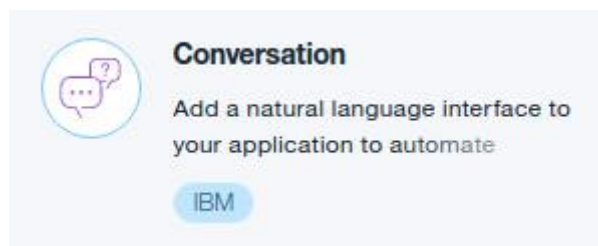




IBM Watson Conversation Service

1. Create and Configure Watson Conversation Service

1. Open Bluemix console
2. Click **Catalog**
3. Select **Conversation** under **Watson**



4. You can leave everything as default but select your node-red app from **Connect to:** and click **Create**

Launch tool [↗](#)

5. Click on the [Launch tool](#) button to launch the Watson Conversation portal
6. Login with the same IBM ID & password if it asks for it.
7. Create a new workspace by clicking **Create**.

Watson Conversation




Create workspace

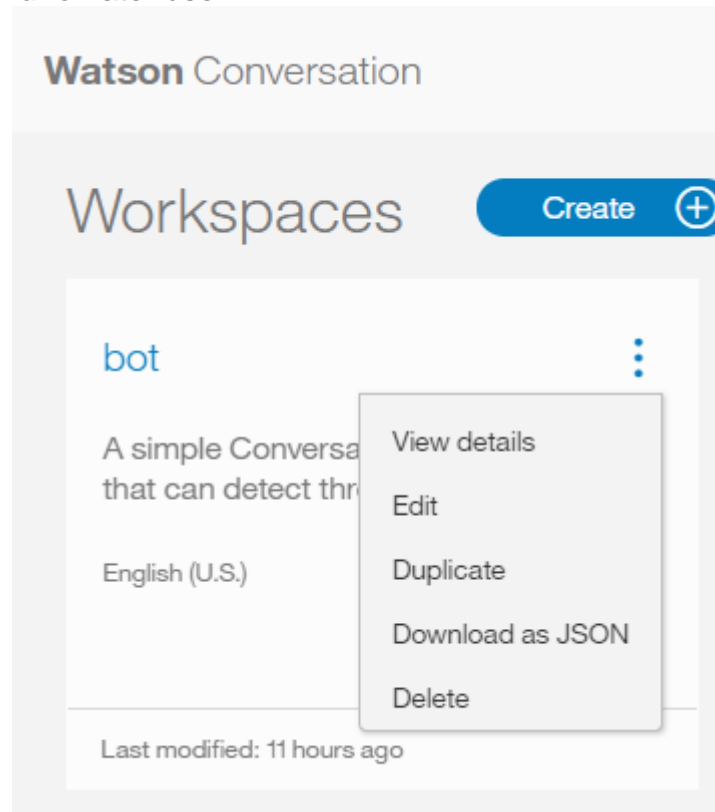
Workspaces enable you to maintain separate intents, user examples, entities, and dialogs for each use or application.

Create

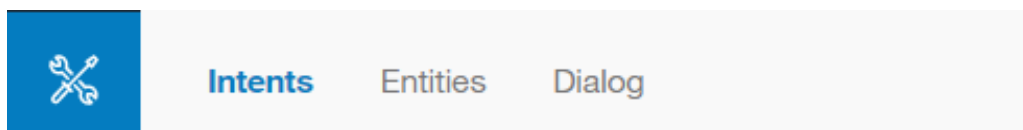


Import

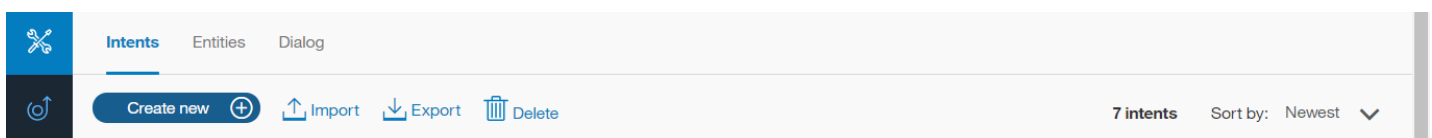
8. Enter **bot_yourname** as the **Name** and click **Create** and click on  and click on **view details**
Copy **workspace id** for later use.



9. Click on the top-left menu and select **Intent**



10. Click **Create new** to create a new intent



11. Enter the following values then click **Done**.

12. Intent name: **greetings**

User example:

- a. **hello**
- b. **hi**
- c. **howdy**

Intents Entities Dialog

Intent name
#greeting

User example
Add a user example... +

hello −

hi −

howdy −

Done

13. Create another one with the following values:

14. Intent name: **farewells**

15. User example:

- a. **goodbye**
- b. **bye**
- c. **see you later**

16. Click on **Entities**

Intents **Entities** Dialog

click create new.

Create new



Name: **food**

Value 1: noodles

Value 2: burger

Value 3: hot dog

17. Click on the **Dialog** tab to configure the dialog flow and click on **Add node**.

Intents Entities **Dialog**

Add node Add child node

test_bot

Welcome
#greet
1 Response / 0 Context set

No condition set
0 Responses / 0 Context set

Anything else
anything_else
1 Response / 0 Context set

Name this node... Customize ×

If bot recognizes:
Enter an intent, entity or context variable... −

Then respond with:

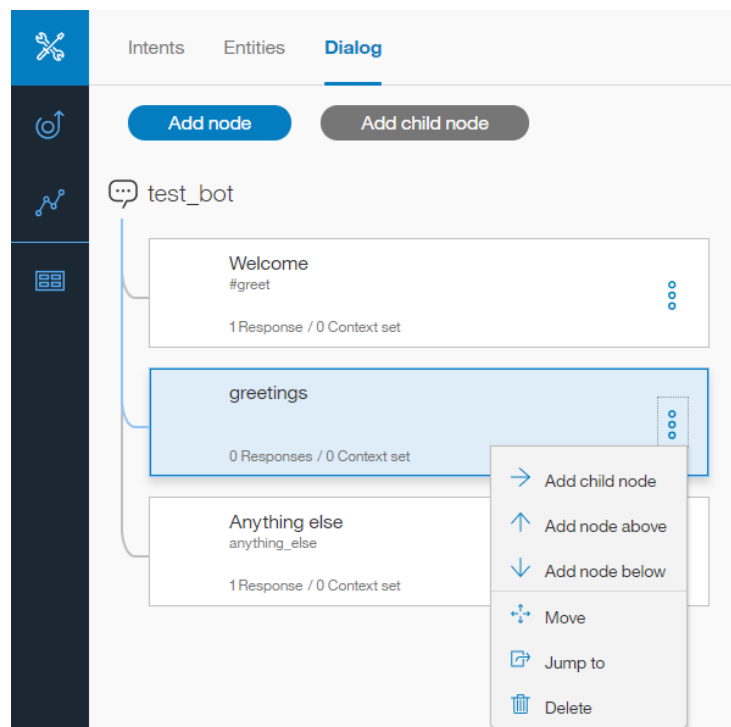
+ Add response condition


Enter a response...

+ Add response


18. Enter the following, then click the **X** button at the top-right:

- a. Name: **Greetings**
- b. Trigger: **#greetings**
- c. Responses: **hello**



19. Click on  of greeting node and then click on **Add node below**.

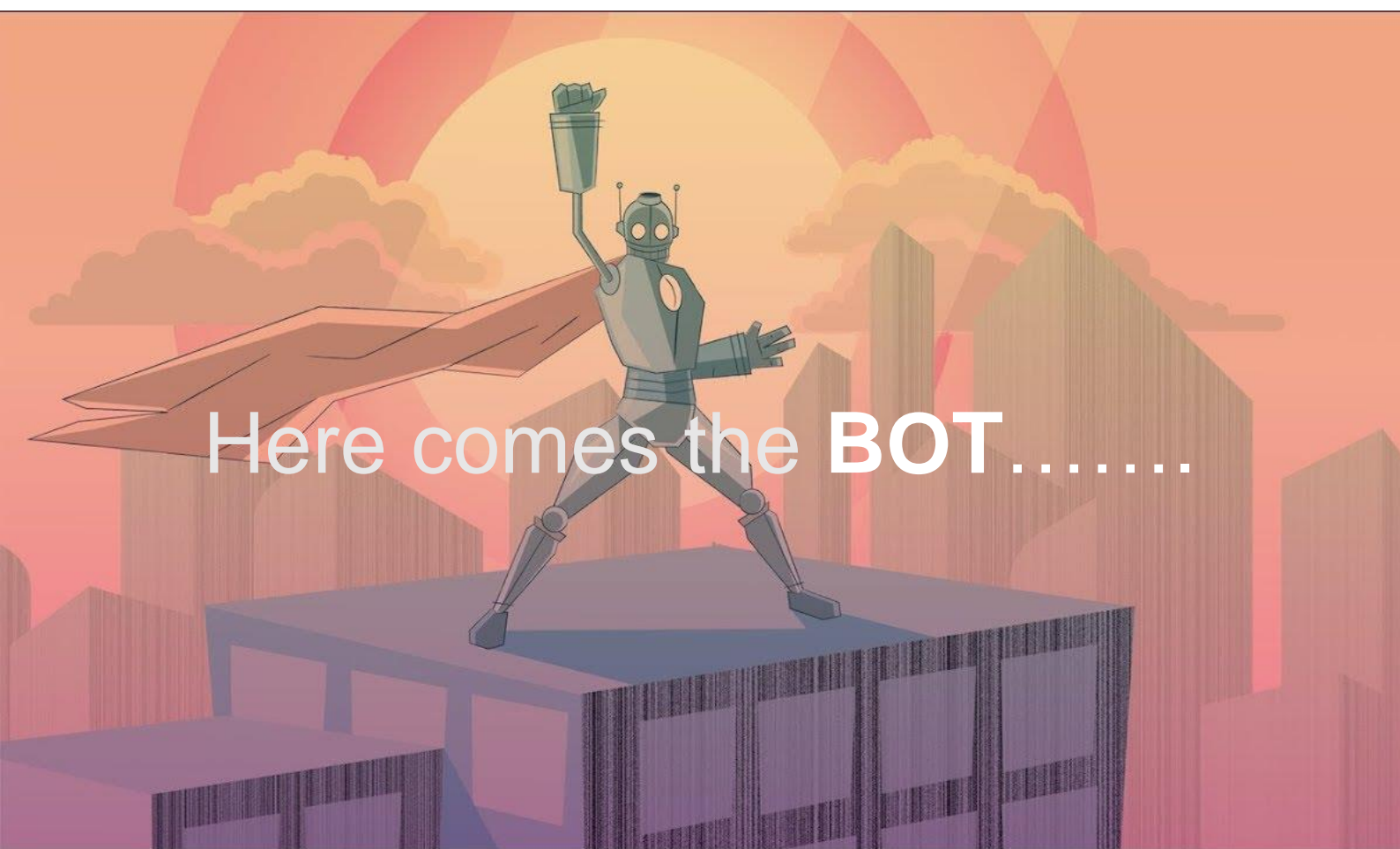
- a. Name: **Farewells**
- b. Trigger: **#farewells**
- c. Responses: **goodbye**

20. To test it, click on the  icon at the top-right, to open a chat box

21. Try saying the user examples that we created earlier, and it should respond accordingly

22. Even with some minor typos, or missing words, it may recognize your intent and respond accordingly.

NOTE:- To import workspace click on  in workspaces.



Here comes the **BOT**.....

Chat-Bot

A chatbot (also known as a talkbot, chatterbot, Bot, IM bot, interactive agent, or Artificial Conversational Entity) is a computer program which conducts a conversation via auditory or textual methods. Such programs are often designed to convincingly simulate how a human would behave as a conversational partner, thereby passing the Turing test. Chatbots are typically used in dialog systems for various practical purposes including customer service or information acquisition. Some chatterbots use sophisticated natural language processing systems.

- Source:Wikipedia

Essential Services used to create Chatbot



Text to speech

+

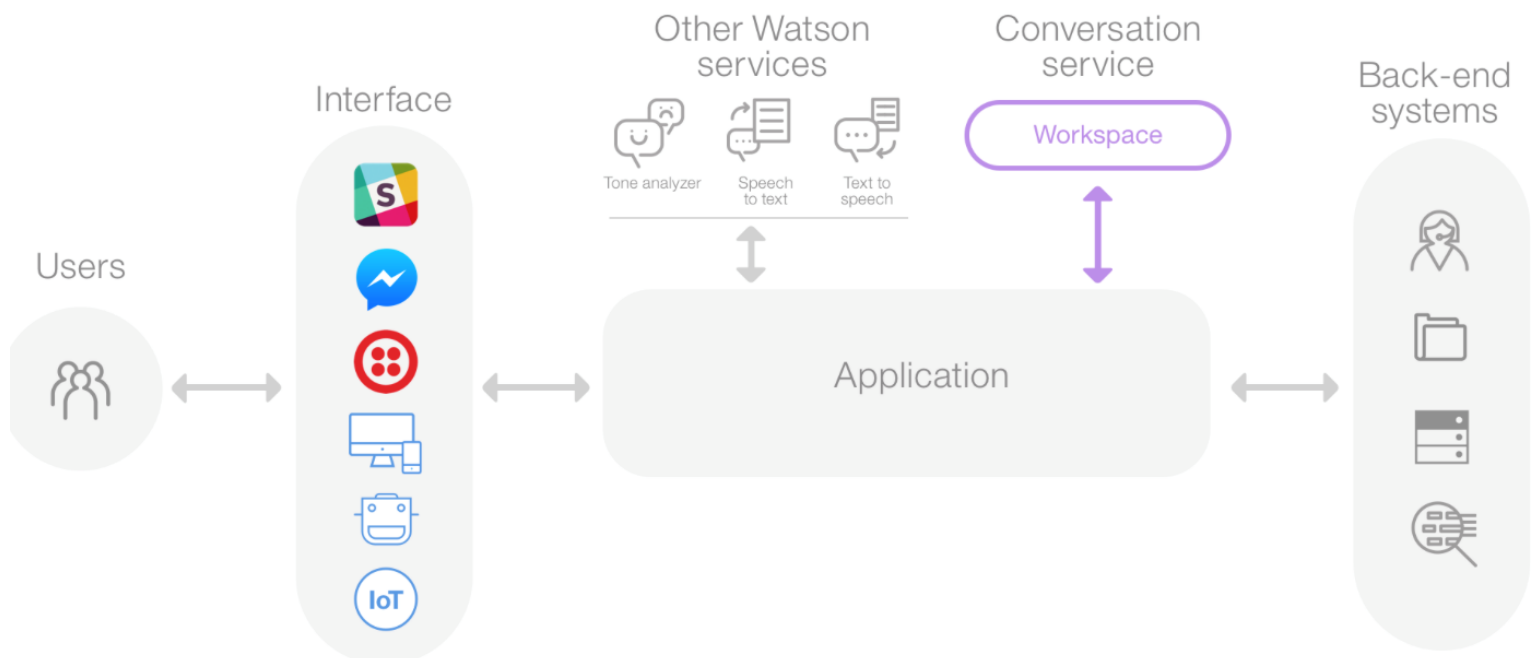


Conversation

+



Speech to text



The Five Main Areas of Development

1. Customer service

One of the most obvious uses for chatbots is customer service—and you might have encountered one of these bots already, without realizing it. Most websites, upon visiting, will have a small live chat tool present to help you find information that you need. Historically, these have been manned by human beings, but it's far more cost effective these days to have bots fielding simple requests. If a request grows too difficult, they can always forward it to a human representative. Customer service chatbots are even starting to be used over the phone, replacing the overly mechanical dial tone-based phone menus that drive us all crazy.

2. Education

Recently, Facebook developed an experimental AI program that mimicked Albert Einstein, engaging with users in natural conversation and doling out facts about his life as if you were having a conversation with the man himself. It wasn't entirely immersive, and probably didn't capture the complexities of his personality, but it did show off the capacity for chatbots' use as educational tools. If chatbots can be programmed to act like historical figures, or even provide users with basic information, they could make education more accessible and more engaging for most populations.

3. Mental health

A New Zealand startup known as Soul Machines is focusing on the emotional element of chatbots, creating programs like Nadia, which have the capacity to interpret human emotions based on conversational inputs like tone and word choice. Eventually, these findings could radically progress how we think about and treat mental health issues. Emotional sensitivity and compassionate responses are exactly what we need to recognize and address complex internal states.

4. Assistance

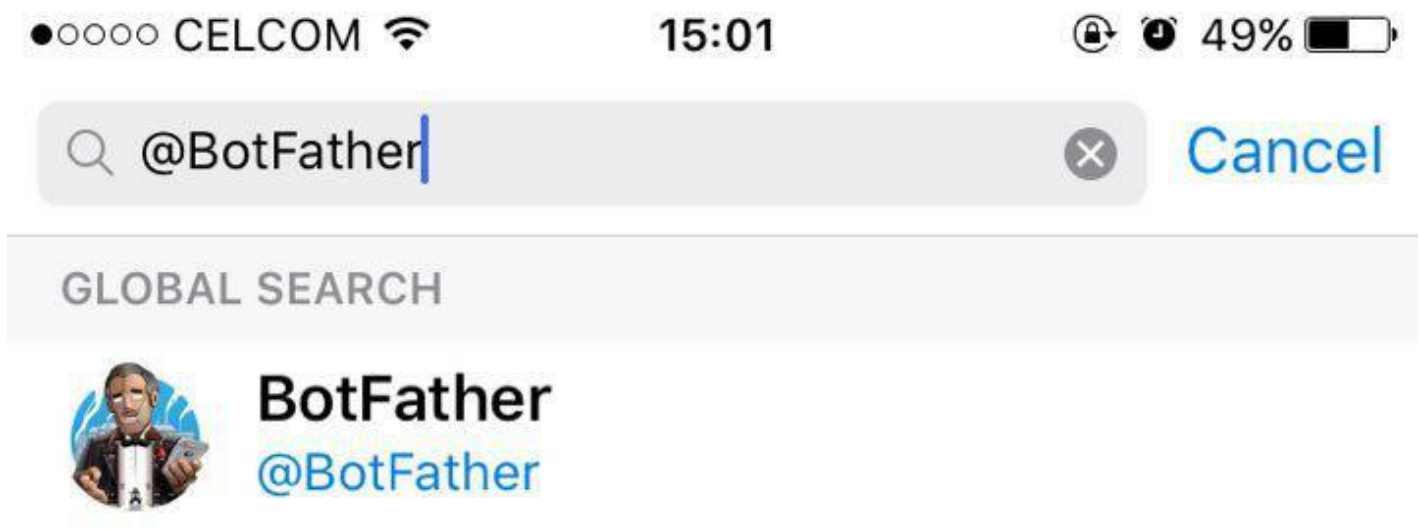
Finally, chatbots are being used as modes of personal assistance, and the best example here is still Siri (and other digital assistants like it). These chatbots are usually connected to an operating system, and are capable of thousands of functions, including playing music, performing online searches, and even buying products online. Smart speakers like Amazon Echo and Google Home are also becoming more popular, introducing more users to the reality of controlling their lives through voice commands. The more accepted it becomes, the higher demand will rise for these features.

If you're a consumer, be on the lookout for chatbots to take over these five main areas of your life. If you're a business owner or an investor, now's the time to jump on these opportunities. Voice recognition and natural language processing have seen massive breakthroughs over the past five years or so, and they're only going to grow more sophisticated from here. Soon, any conversation you have with a human being could be quickly replicable through the use of machine learning algorithms.

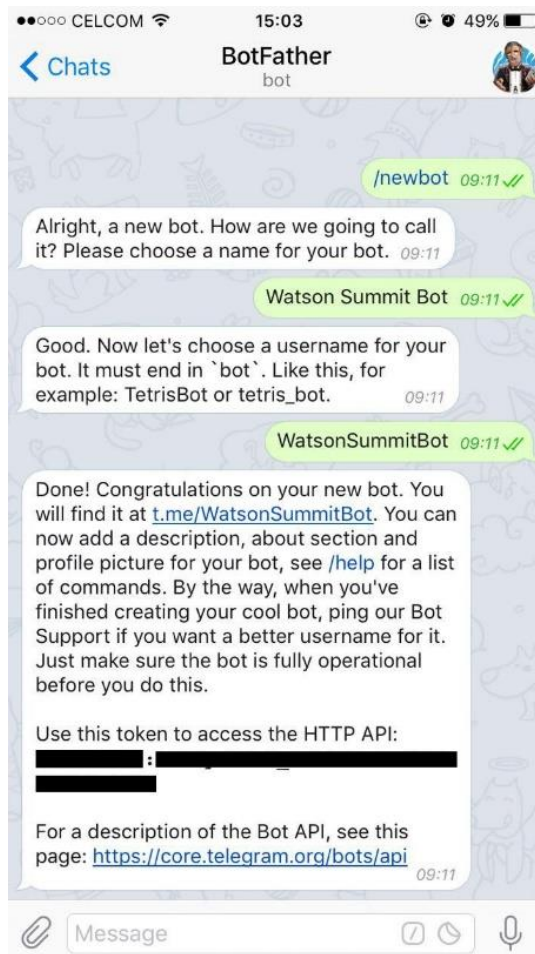
Let's get started.....

1. Create a new bot on Telegram's BotFather

1. Run the **Telegram** app, after installing it or login <https://web.telegram.org/> .
2. Search for **@BotFather** at the search bar on top and select it



3. Send **/newbot** command / message to **BotFather**.
4. Enter the name and username of your bot, for example:
 1. name:<your_bot_name>
 2. username: **Bot**<your_initials>
5. Once created, you'll be given a token string.



6. Save the token on your text editor.

2. Create chatbot service for telegram bot

1. Go to <https://console.bluemix.net/catalog/> search and create services mentioned below using similar method you created node-red cloud foundry app.
 - a. Text to speech
 - b. Speech to text
 - c. Conversation (already created)
 - d. Visual recognition
 - e. Language translator
 - f. Tone analyser
2. Get back to node-red flow editor.
3. Keep your copied credentials ready.

Demo time...