Smart Agriculture system based on IoT

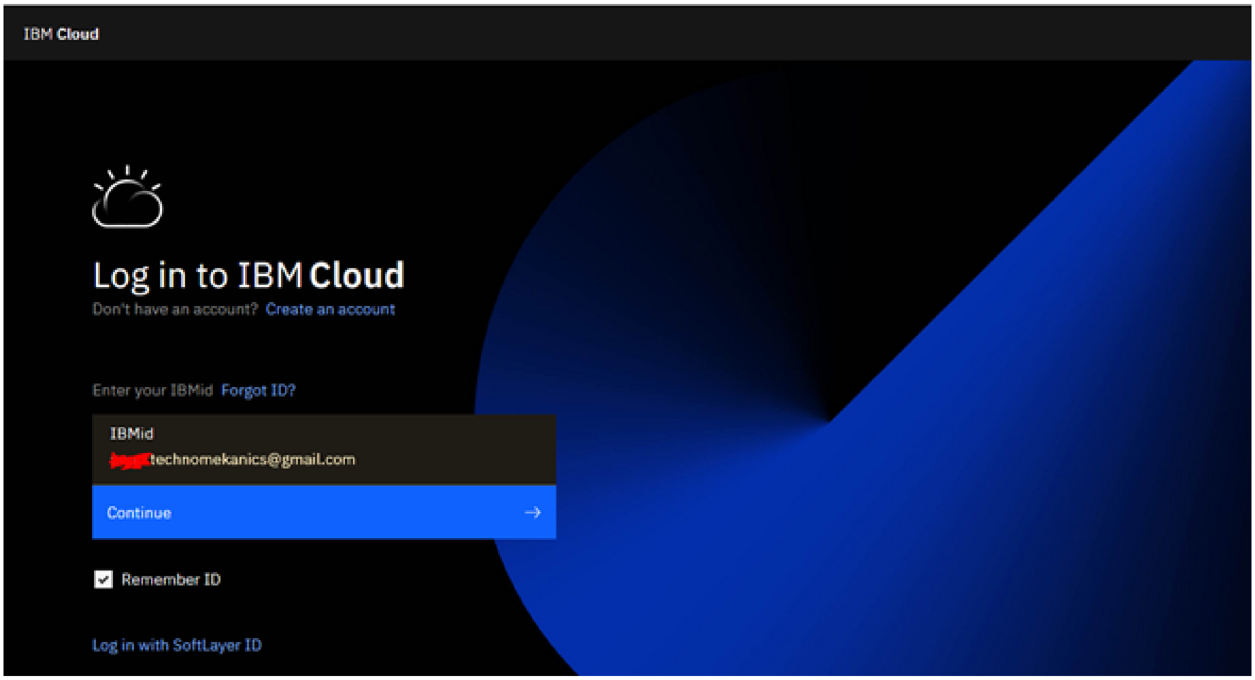
Brief: This tutorial will guide you in developing the Smart Agriculture system based on IoT project using node-red and IBM IOT Platform.

Activities:

1. Check whether IOT Device is receiving the data from IOT simulator in IBM Watson IoT platform
2. Configure Node-red to get Data by installing Required nodes
3. Create the Web UI to visualize the indoor weather parameters and control the lights

TASK 1: Connecting device to cloud to see the data in the cards section

Step 1: Login to your IBM cloud account and click on services.



IBM Cloud

Log in to IBM Cloud

Don't have an account? Create an account

Enter your IBMid Forgot ID?

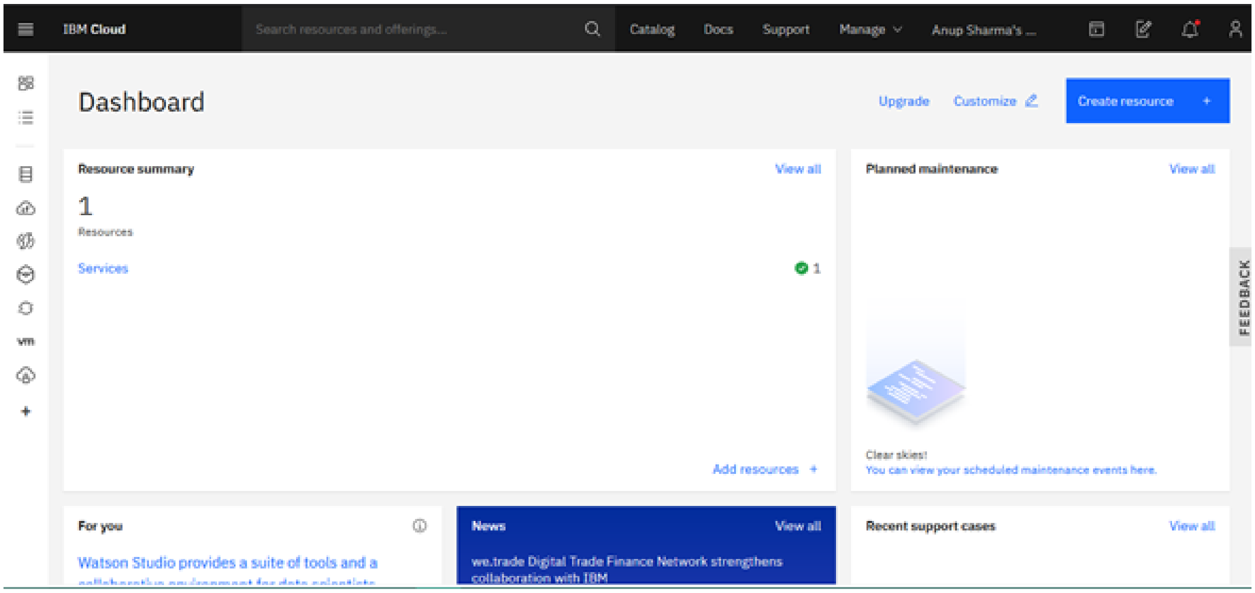
IBMid
[redacted]technomekanics@gmail.com

Continue →

☑ Remember ID

Log in with SoftLayer ID

Then we will get into **dashboard** as shown below

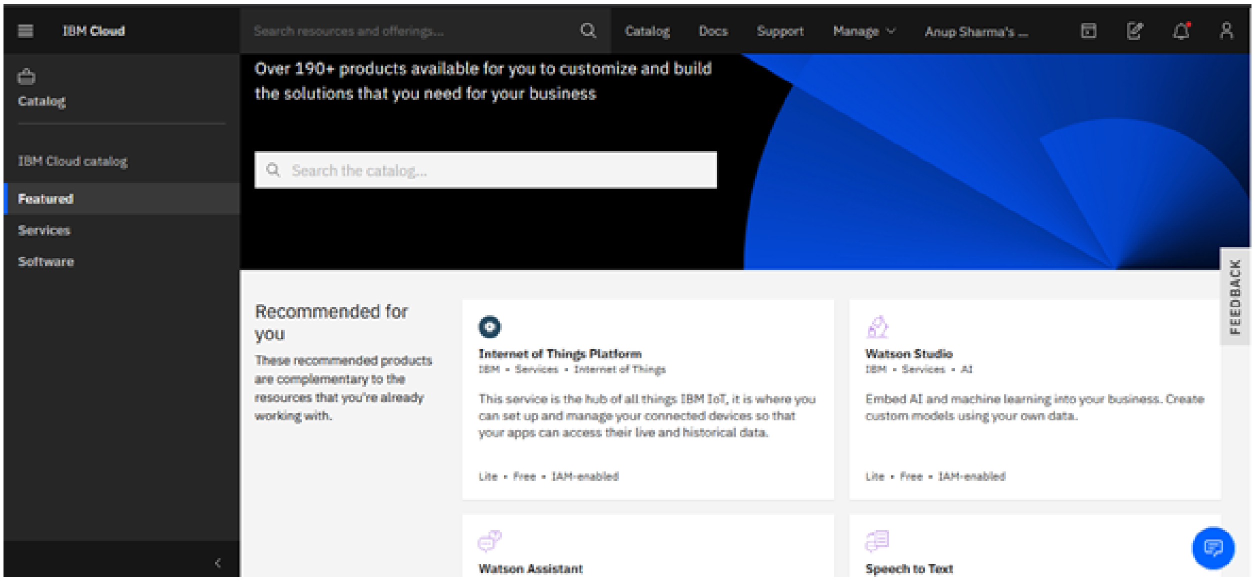


Clear skies!
You can view your scheduled maintenance events here.

Watson Studio provides a suite of tools and a collaborative environment for data scientists

we.trade Digital Trade Finance Network strengthens collaboration with IBM

Then we will click on **Catalog** then search for Internet of things after that click on create



Over 190+ products available for you to customize and build the solutions that you need for your business

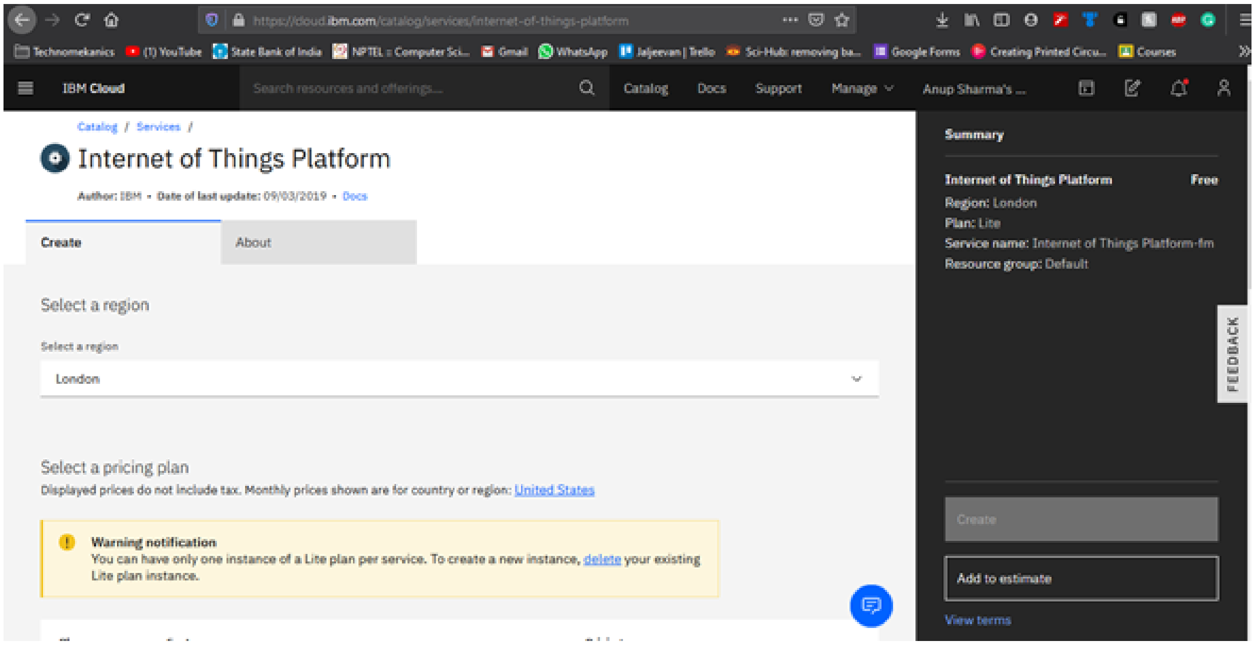
Recommended for you

These recommended products are complementary to the resources that you're already working with.

Internet of Things Platform
IBM • Services • Internet of Things
This service is the hub of all things IBM IoT, it is where you can set up and manage your connected devices so that your apps can access their live and historical data.
Lite • Free • IAM-enabled

Watson Studio
IBM • Services • AI
Embed AI and machine learning into your business. Create custom models using your own data.
Lite • Free • IAM-enabled

Then we will click on **Catalog** then search for **Internet of things** after that click on create



Internet of Things Platform

Author: IBM • Date of last update: 09/03/2019 • Docs

Select a region

Select a region: London

Select a pricing plan

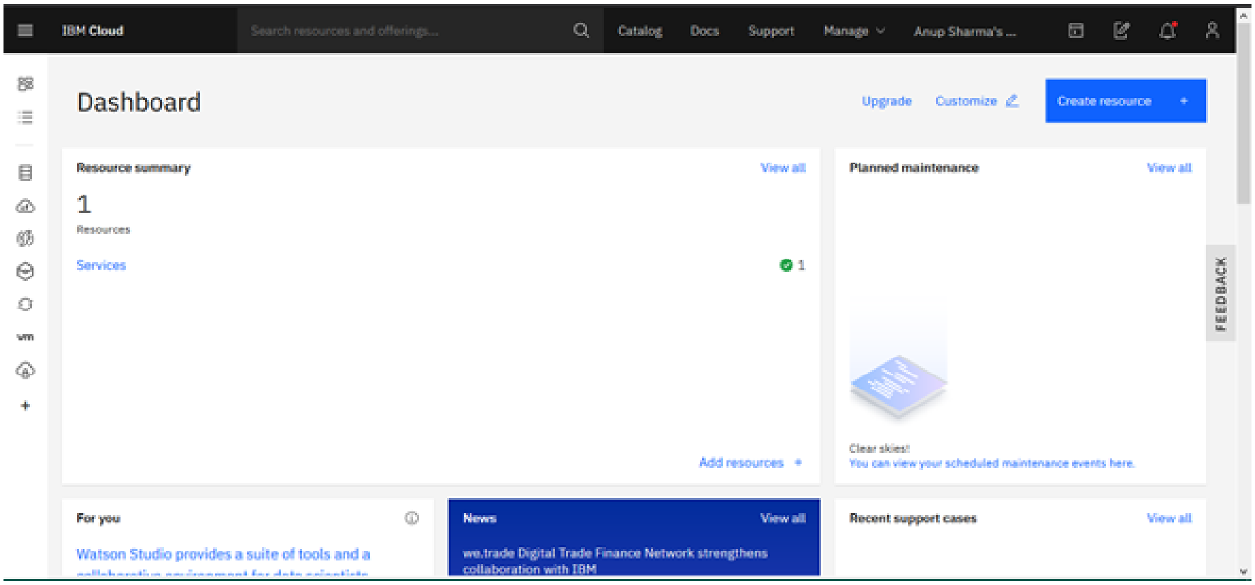
Displayed prices do not include tax. Monthly prices shown are for country or region: United States

Warning notification
You can have only one instance of a Lite plan per service. To create a new instance, delete your existing Lite plan instance.

Summary

Internet of Things Platform Free
Region: London
Plan: Lite
Service name: Internet of Things Platform-fm
Resource group: Default

After that you will be again land into **dashboard** page. Then click on **Service**



Dashboard

Resource summary
1
Resources
Services 1

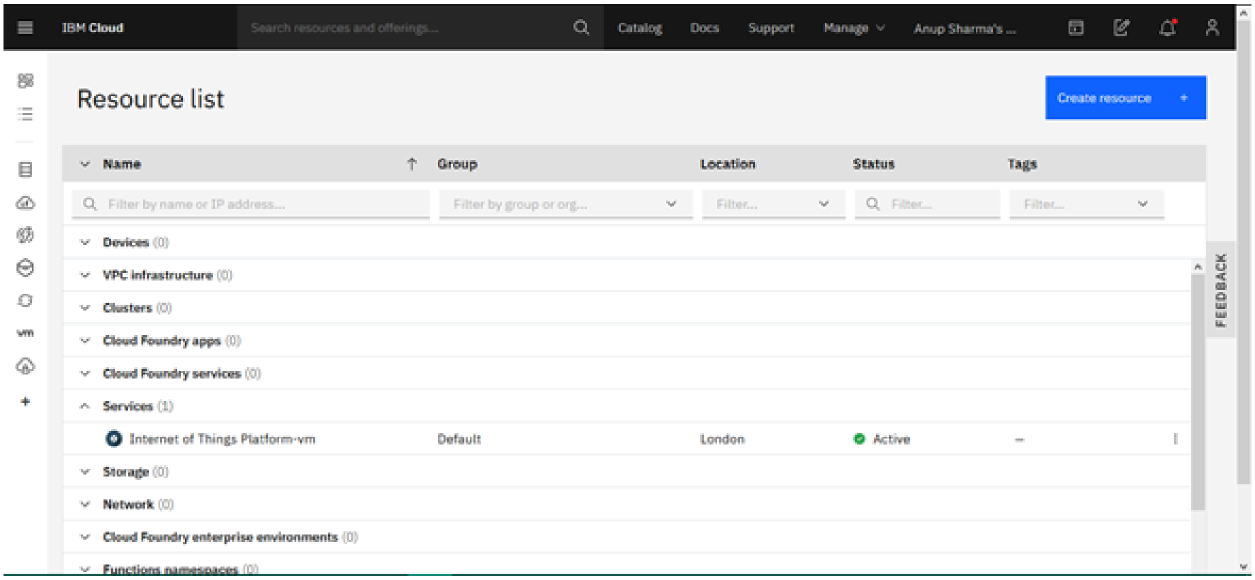
Planned maintenance
Clear skies!
You can view your scheduled maintenance events here.

For you
Watson Studio provides a suite of tools and a collaborative environment for data scientists

News
we.trade Digital Trade Finance Network strengthens collaboration with IBM

Recent support cases

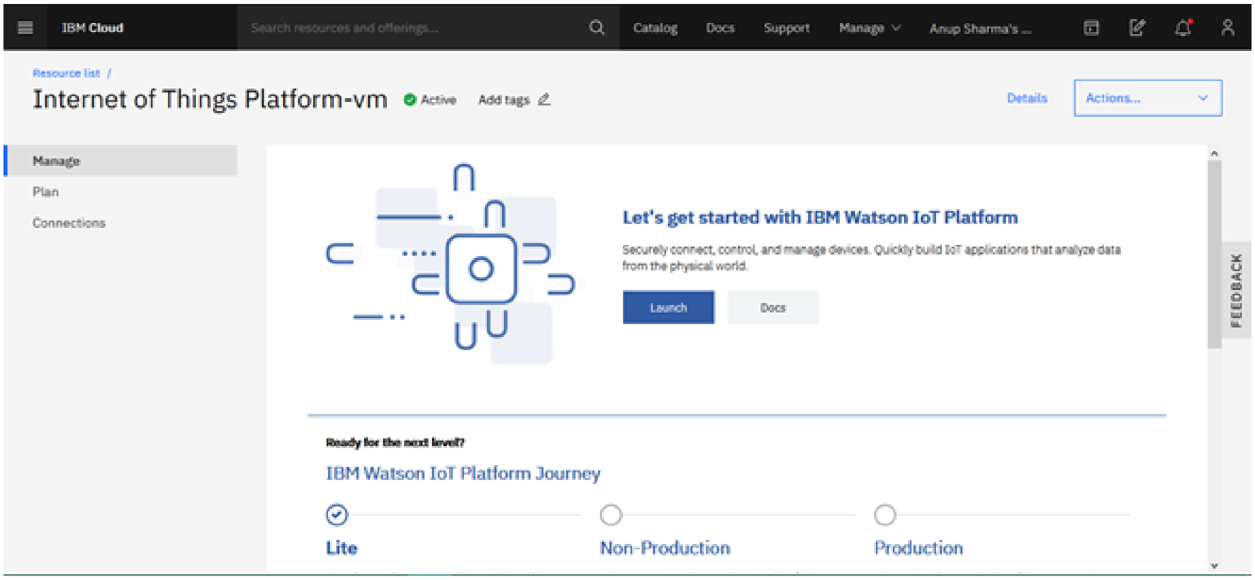
Then click on **Internet of things Platform-vm**



Resource list

Name	Group	Location	Status	Tags
Filter by name or IP address...	Filter by group or org...	Filter...	Filter...	Filter...
Devices (0)				
VPC infrastructure (0)				
Clusters (0)				
Cloud Foundry apps (0)				
Cloud Foundry services (0)				
Services (1)				
Internet of Things Platform-vm	Default	London	Active	–
Storage (0)				
Network (0)				
Cloud Foundry enterprise environments (0)				
Functions namespaces (0)				

Then click on **Launch**



Internet of Things Platform-vm

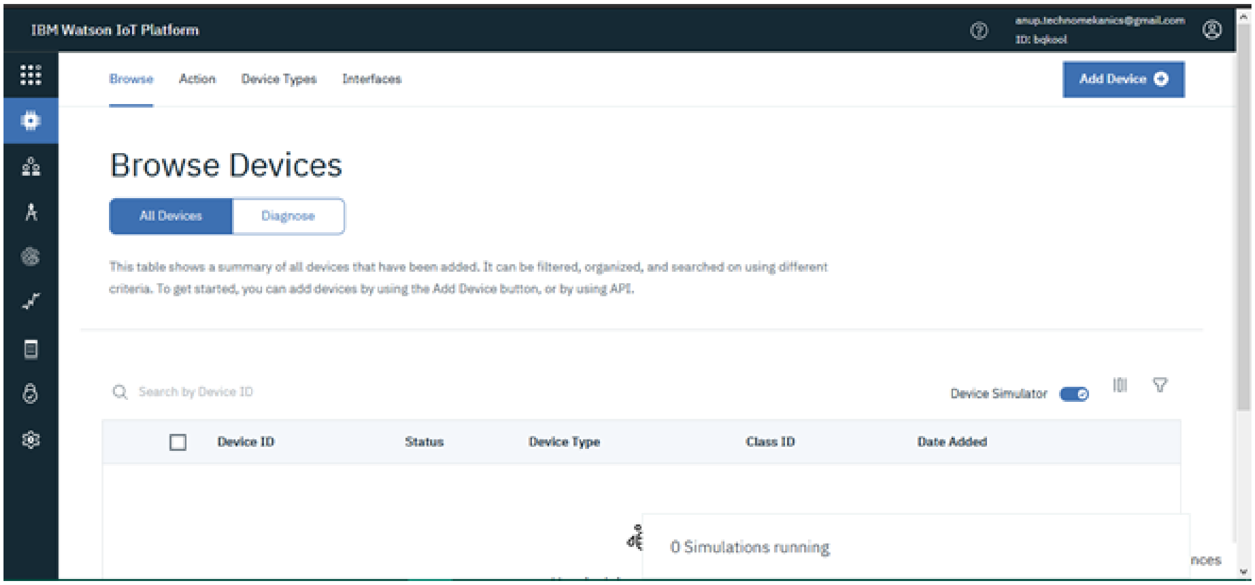
Let's get started with IBM Watson IoT Platform

Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.

Ready for the next level?

IBM Watson IoT Platform Journey

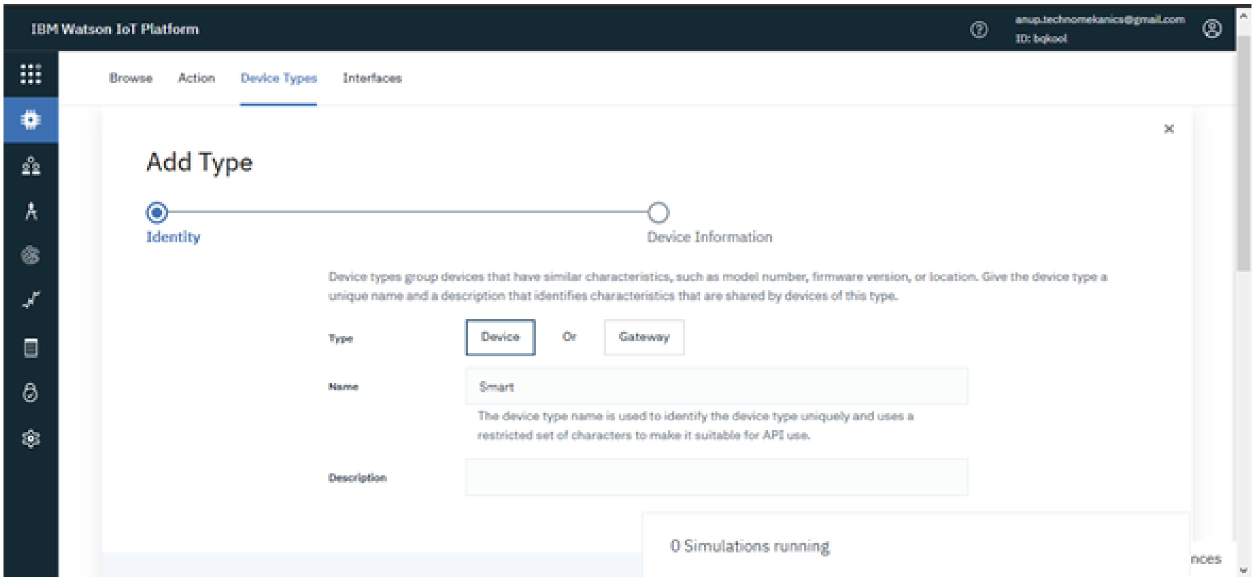
Then we will see the page as shown below



This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

☐	Device ID	Status	Device Type	Class ID	Date Added

Click on **Device Type** then select **Type** as **Device** then **name** <smart>.



Device types group devices that have similar characteristics, such as model number, firmware version, or location. Give the device type a unique name and a description that identifies characteristics that are shared by devices of this type.

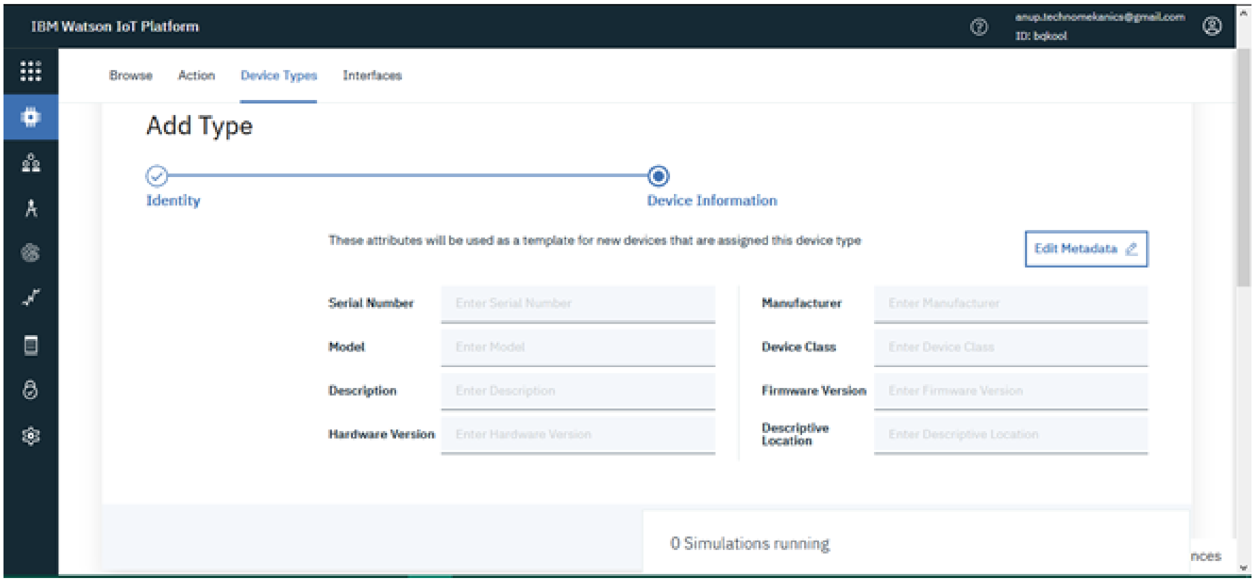
Type: Device Or Gateway

Name: Smart

The device type name is used to identify the device type uniquely and uses a restricted set of characters to make it suitable for API use.

Description:

Then leave this space blank then click on **Next**



Add Type

These attributes will be used as a template for new devices that are assigned this device type

Serial Number: Enter Serial Number

Model: Enter Model

Description: Enter Description

Hardware Version: Enter Hardware Version

Manufacturer: Enter Manufacturer

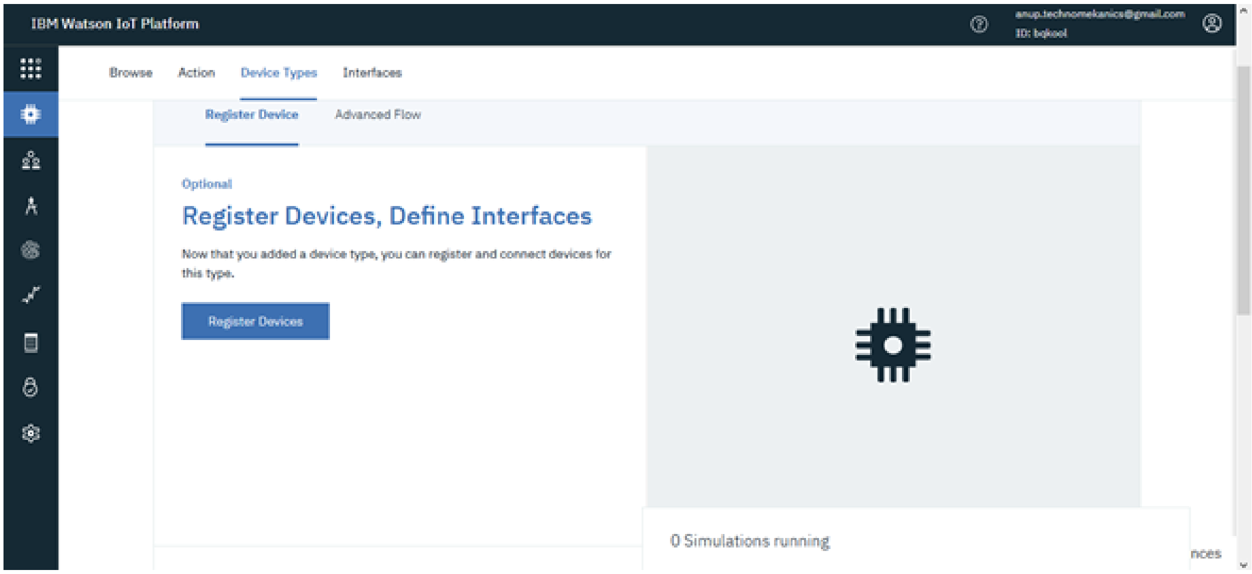
Device Class: Enter Device Class

Firmware Version: Enter Firmware Version

Descriptive Location: Enter Descriptive Location

0 Simulations running

Then click on **Register Devices**



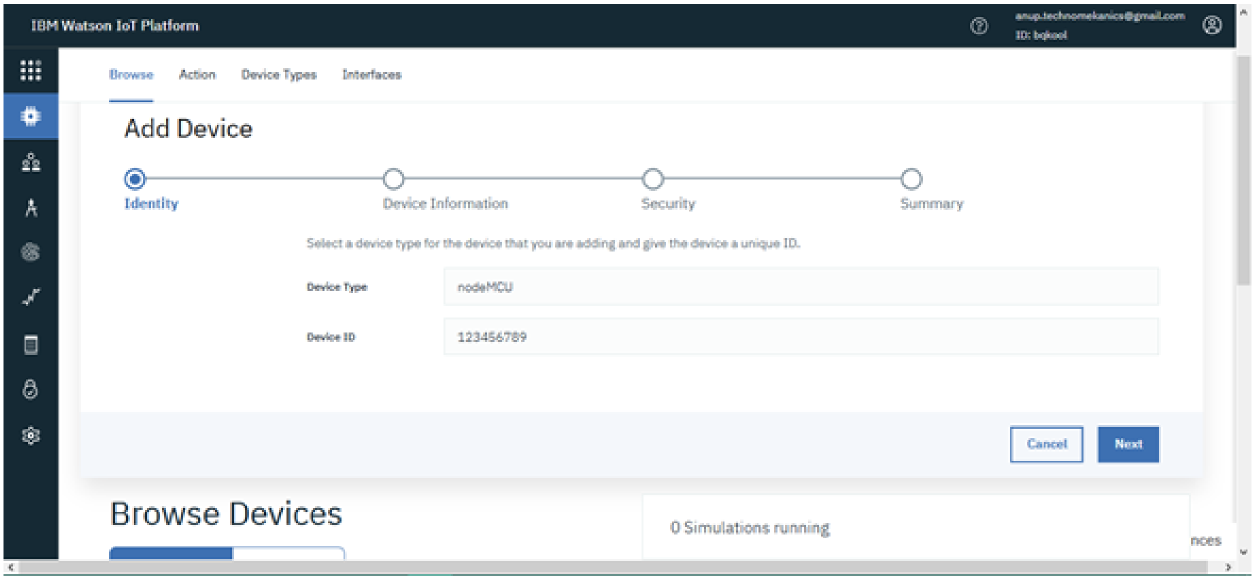
Optional

Register Devices, Define Interfaces

Now that you added a device type, you can register and connect devices for this type.

0 Simulations running

Then set **Device Type** and **Device ID**.



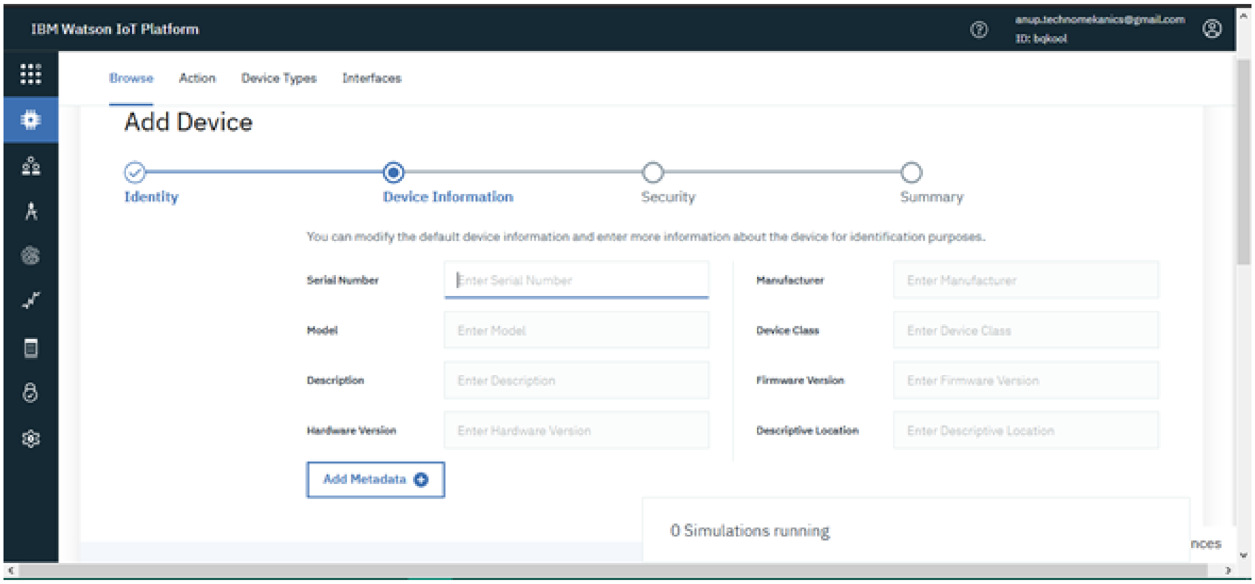
Add Device

Select a device type for the device that you are adding and give the device a unique ID.

Device Type	nodeMCU
Device ID	123456789

Browse Devices

Then leave this space **blank**

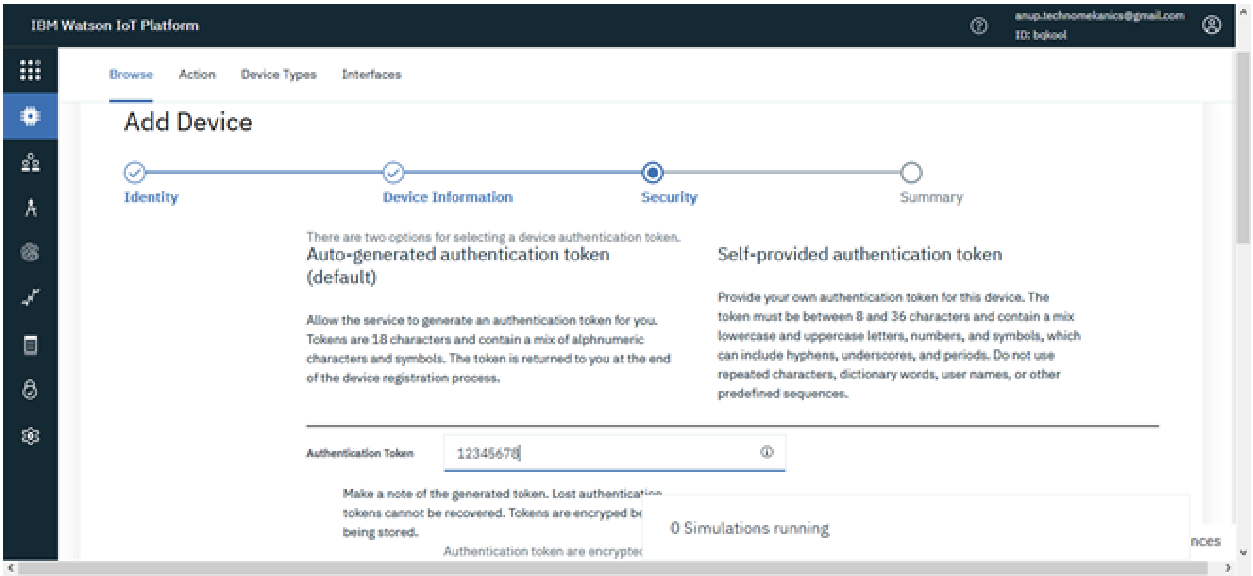


Add Device

You can modify the default device information and enter more information about the device for identification purposes.

Serial Number	Enter Serial Number	Manufacturer	Enter Manufacturer
Model	Enter Model	Device Class	Enter Device Class
Description	Enter Description	Firmware Version	Enter Firmware Version
Hardware Version	Enter Hardware Version	Descriptive Location	Enter Descriptive Location

Then set the **authentication key** according to your convince.



Add Device

There are two options for selecting a device authentication token.

Auto-generated authentication token (default)

Allow the service to generate an authentication token for you. Tokens are 18 characters and contain a mix of alphnumeric characters and symbols. The token is returned to you at the end of the device registration process.

Self-provided authentication token

Provide your own authentication token for this device. The token must be between 8 and 36 characters and contain a mix lowercase and uppercase letters, numbers, and symbols, which can include hyphens, underscores, and periods. Do not use repeated characters, dictionary words, user names, or other predefined sequences.

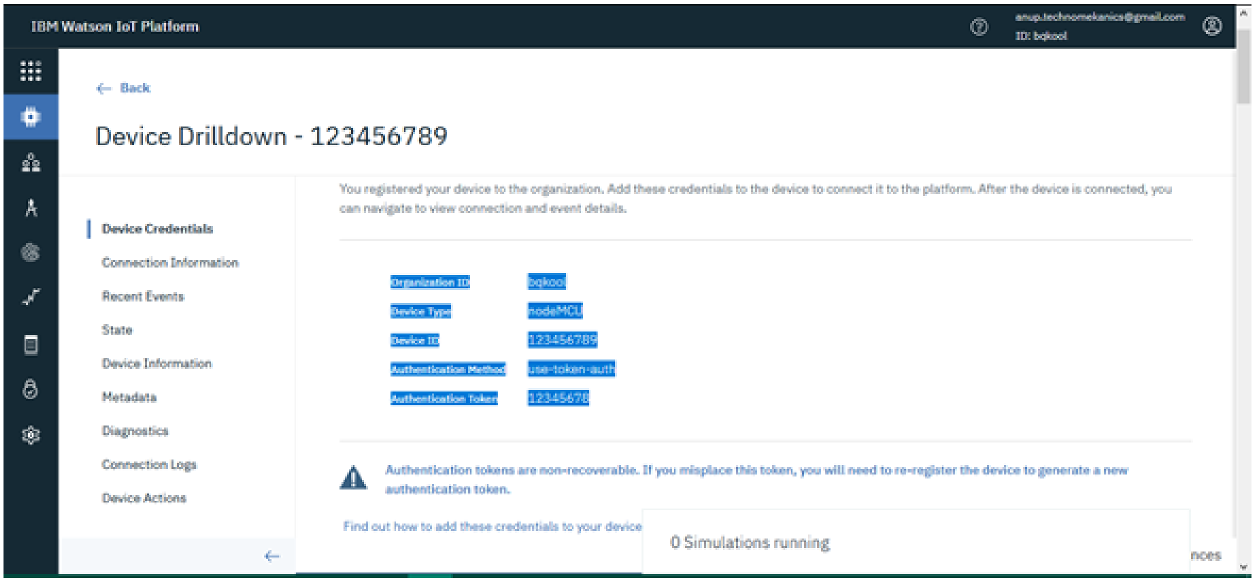
Authentication Token 12345678

Make a note of the generated token. Lost authenticat
tokens cannot be recovered. Tokens are encryped be
being stored.

Authentication token are encryptec

0 Simulations running

Then copy this details into **Notepad**



Device Drilldown - 123456789

You registered your device to the organization. Add these credentials to the device to connect it to the platform. After the device is connected, you can navigate to view connection and event details.

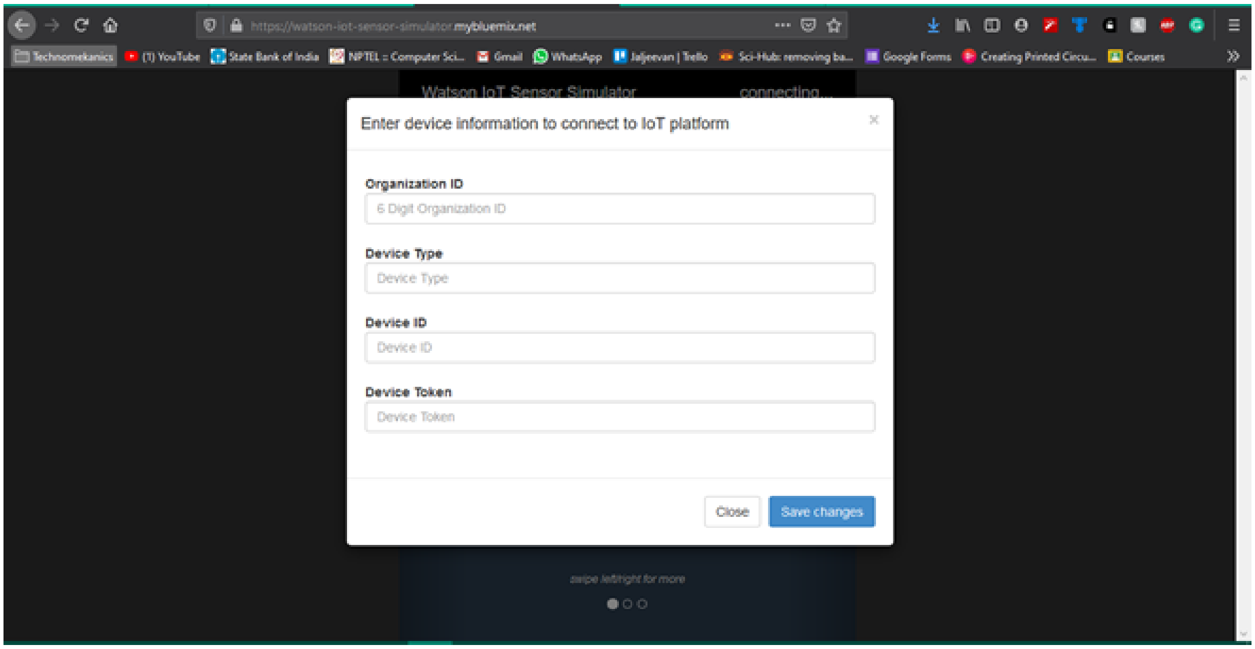
Organization ID	bqkool
Device Type	nodeMCU
Device ID	123456789
Authentication Method	use-token-auth
Authentication Token	12345678

Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the device to generate a new authentication token.

Find out how to add these credentials to your device

0 Simulations running

After that go to https://watson-iot-sensor-simulator.mybluemix.net/ then give input all the details that you have copied in Notepad.



Enter device information to connect to IoT platform

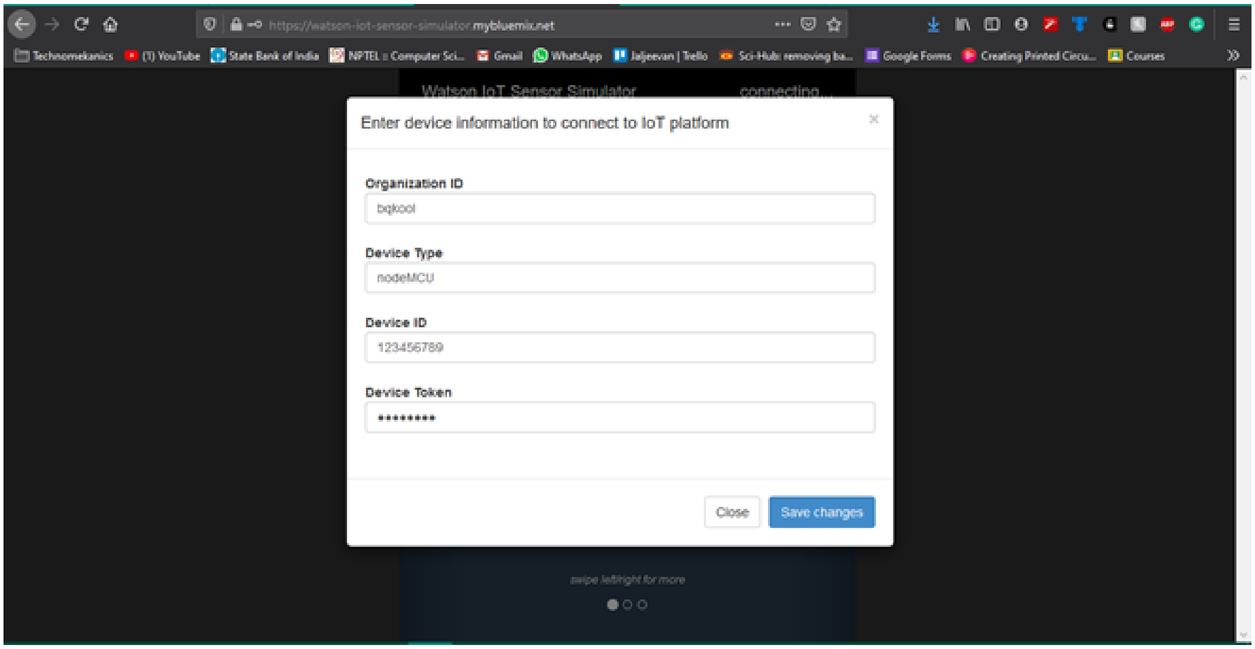
Organization ID
6 Digit Organization ID

Device Type
Device Type

Device ID
Device ID

Device Token
Device Token

After entering all the details:



Enter device information to connect to IoT platform

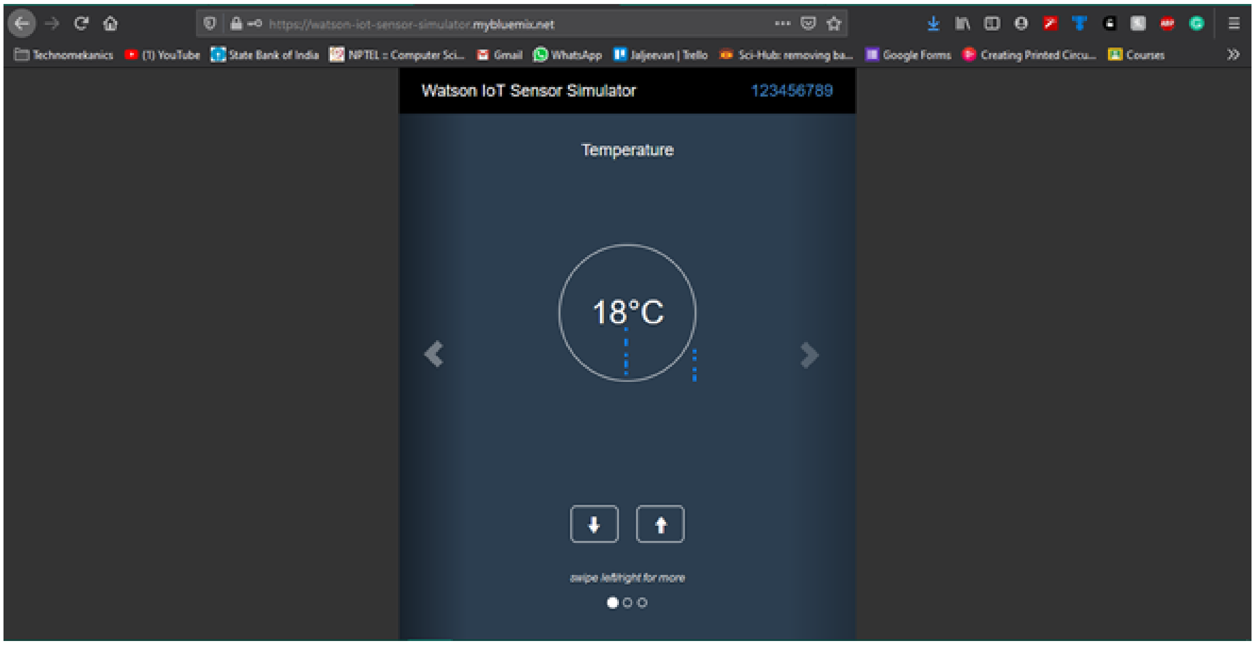
Organization ID
bqkool

Device Type
nodeMCU

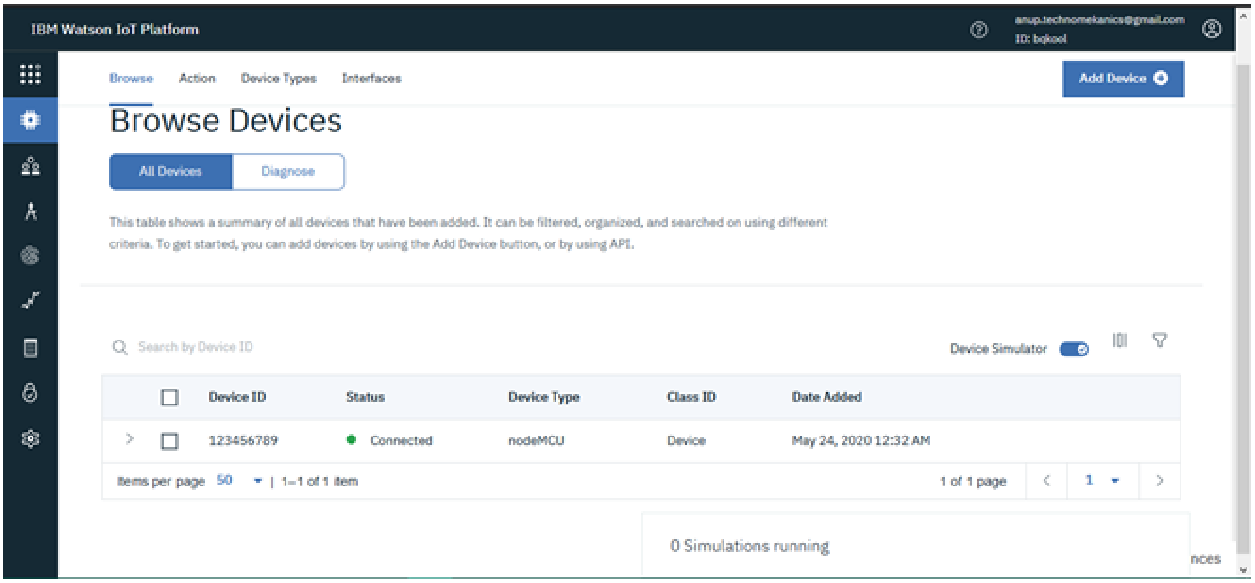
Device ID
123456789

Device Token
•••••••••

After that we can see this screen



Then we can see in cloud.ibm.com device has been **connected**



Browse Devices

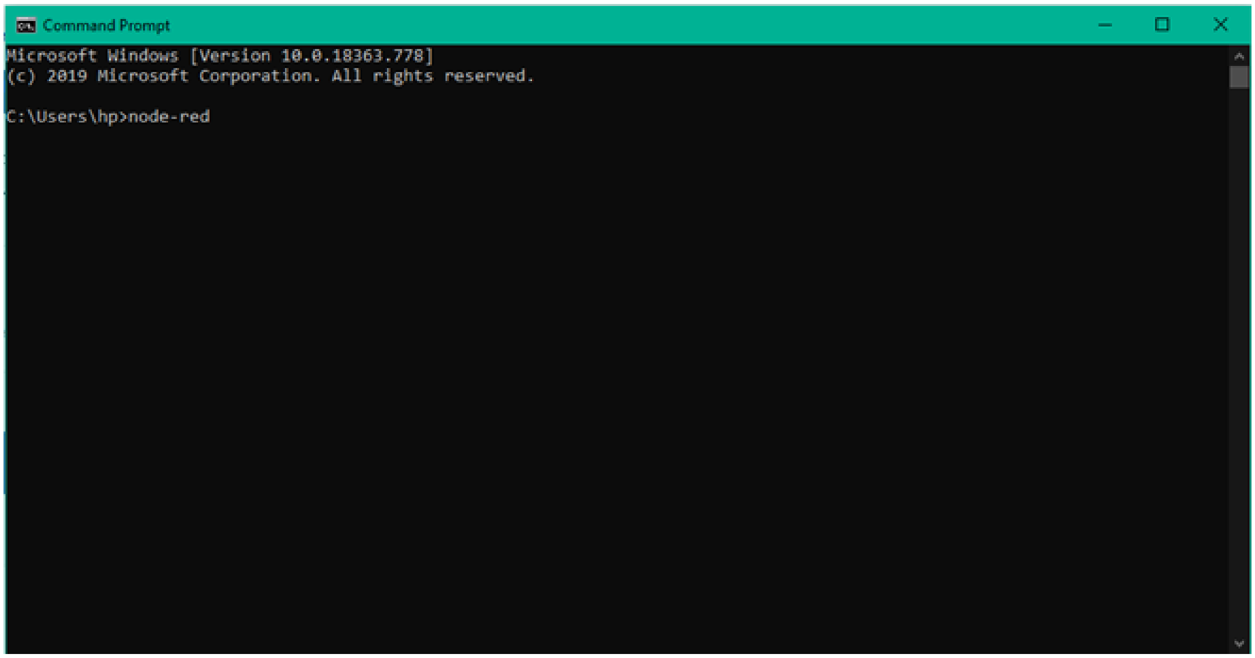
This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

	Device ID	Status	Device Type	Class ID	Date Added
□	123456789	● Connected	nodeMCU	Device	May 24, 2020 12:32 AM

Items per page 50 | 1–1 of 1 item   1 of 1 page   1

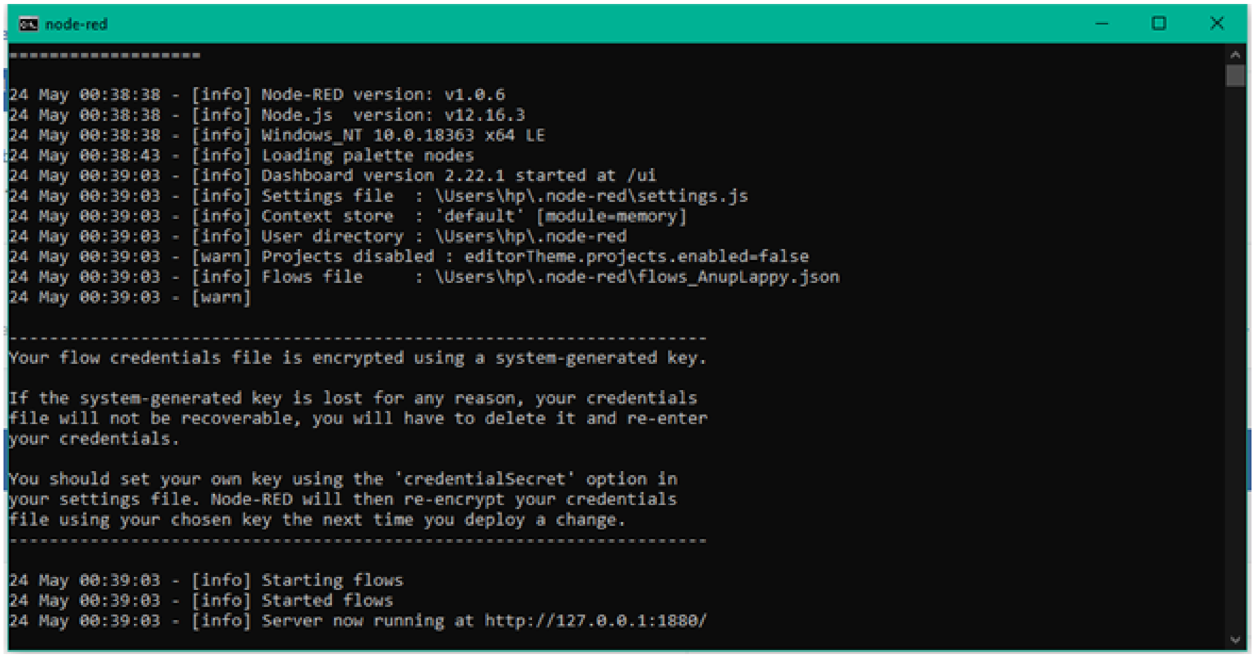
0 Simulations running

Then do to **CMD** then type **Node-red**



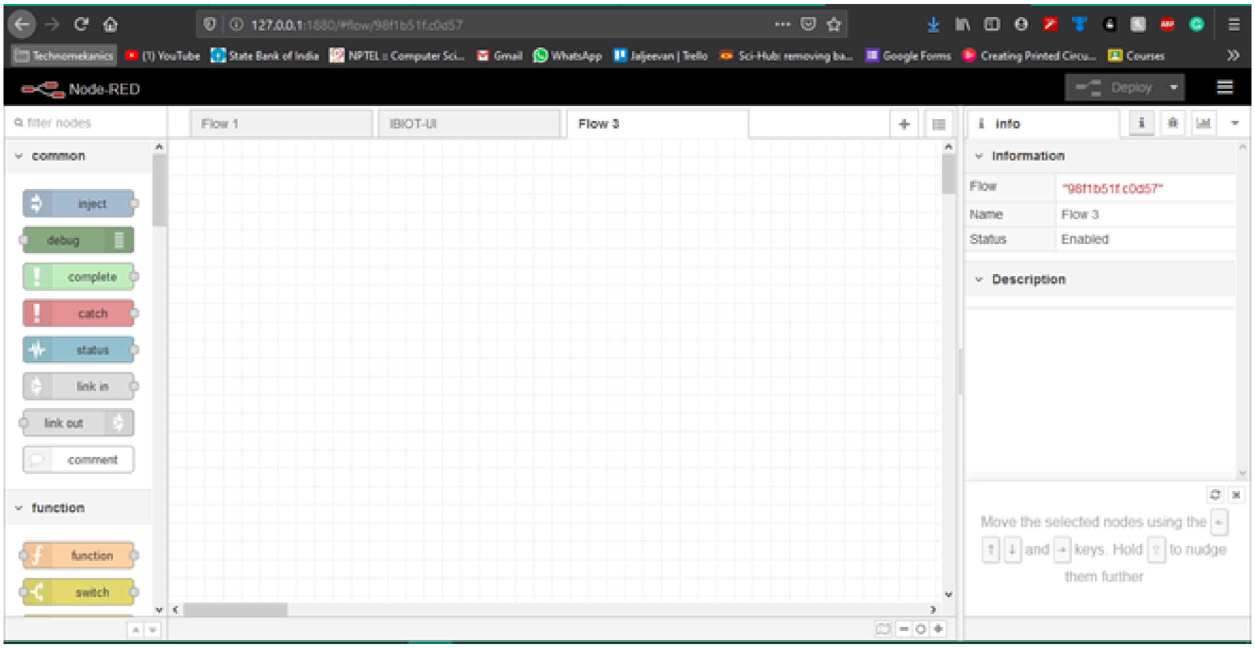
```
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\hp>node-red
```

After that copy the **URL**



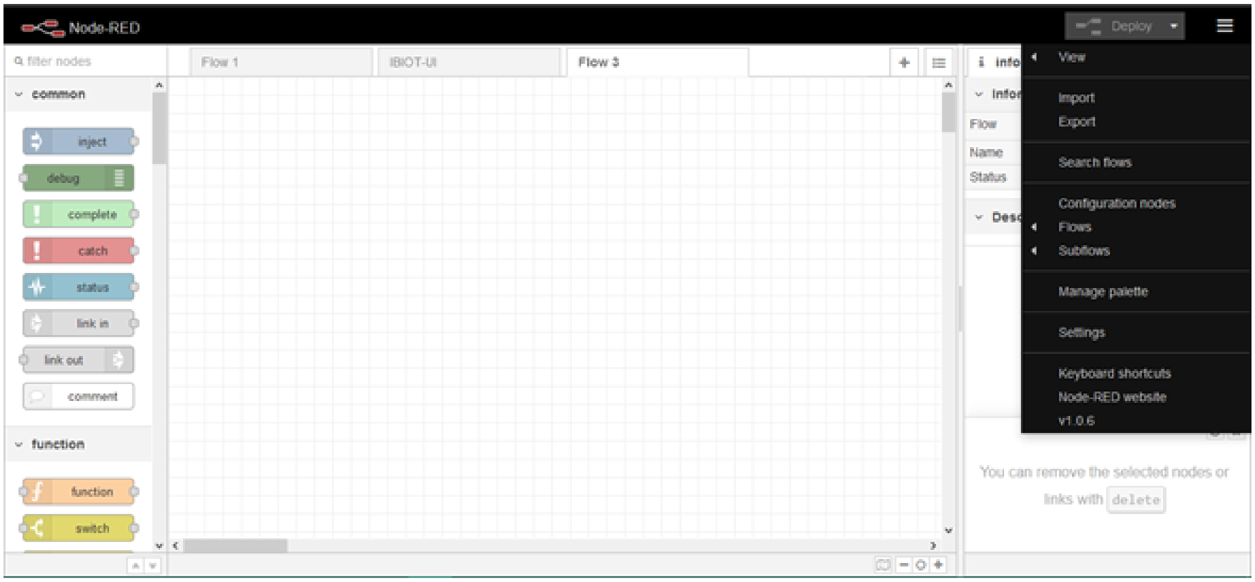
```
==================

24 May 00:38:38 - [info] Node-RED version: v1.0.6
24 May 00:38:38 - [info] Node.js  version: v12.16.3
24 May 00:38:38 - [info] Windows_NT 10.0.18363 x64 LE
24 May 00:38:43 - [info] Loading palette nodes
24 May 00:39:03 - [info] Dashboard version 2.22.1 started at /ui
24 May 00:39:03 - [info] Settings file  : \Users\hp\.node-red\settings.js
24 May 00:39:03 - [info] Context store  : 'default' [module=memory]
24 May 00:39:03 - [info] User directory : \Users\hp\.node-red
24 May 00:39:03 - [warn] Projects disabled : editorTheme.projects.enabled=false
24 May 00:39:03 - [info] Flows file     : \Users\hp\.node-red\flows_AnupLappy.json
24 May 00:39:03 - [warn]

---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------

24 May 00:39:03 - [info] Starting flows
24 May 00:39:03 - [info] Started flows
24 May 00:39:03 - [info] Server now running at http://127.0.0.1:1880/
```

Then enter the URL in browser than we can see such screen



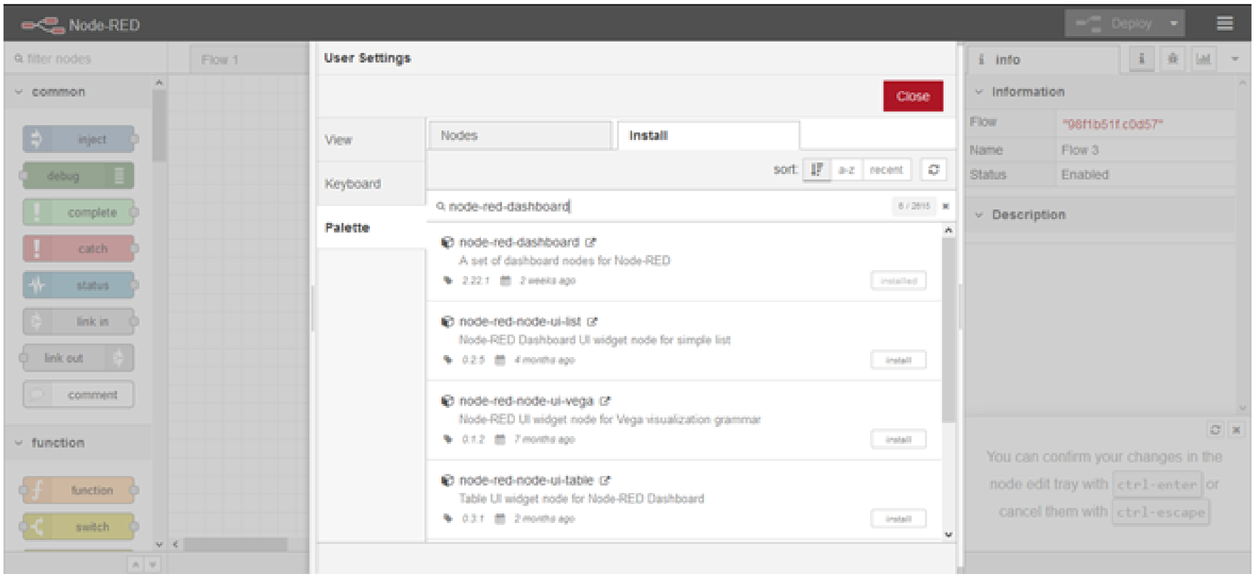
Move the selected nodes using the ← ↑ ↓ and → keys. Hold ⇧ to nudge them further

After that click on 3 dash then click on **manage palette.**



You can remove the selected nodes or links with delete

Then search for **node-red-dashboard** then click **install**



node-red-dashboard

6 / 2615

node-red-dashboard
A set of dashboard nodes for Node-RED
2.22.1 2 weeks ago installed

node-red-node-ui-list
Node-RED Dashboard UI widget node for simple list
0.2.5 4 months ago install

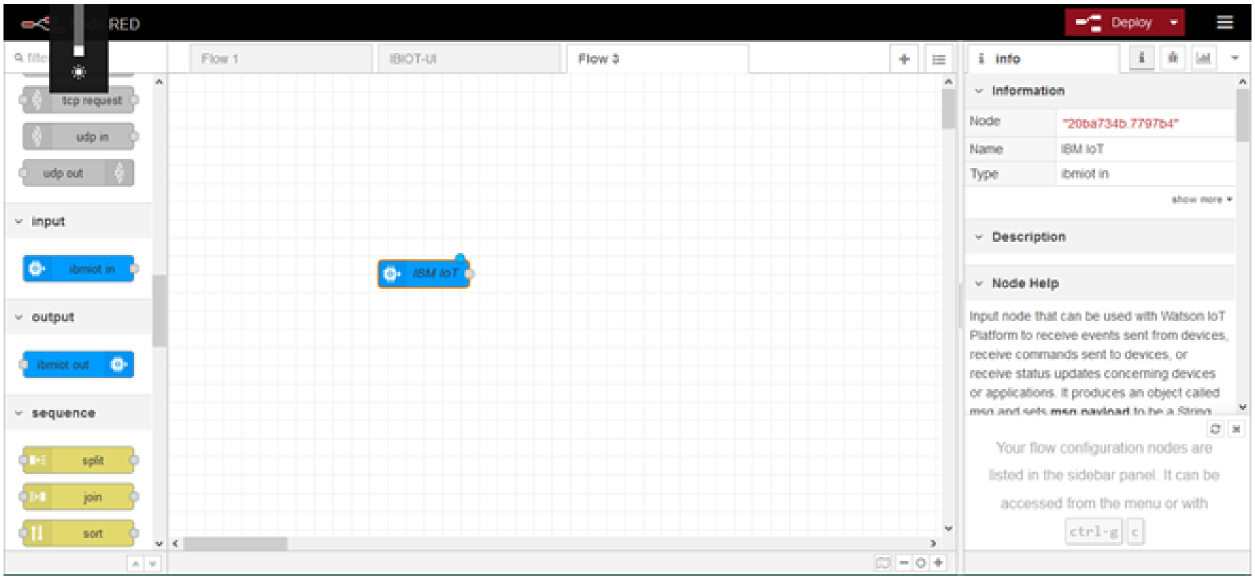
node-red-node-ui-vega
Node-RED UI widget node for Vega visualization grammar
0.1.2 7 months ago install

node-red-node-ui-table
Table UI widget node for Node-RED Dashboard
0.3.1 2 months ago install

Flow	"98f1b51f.c0d57"
Name	Flow 3
Status	Enabled

You can confirm your changes in the node edit tray with ctrl-enter or cancel them with ctrl-escape

After that drag the **IBM IoT input** node

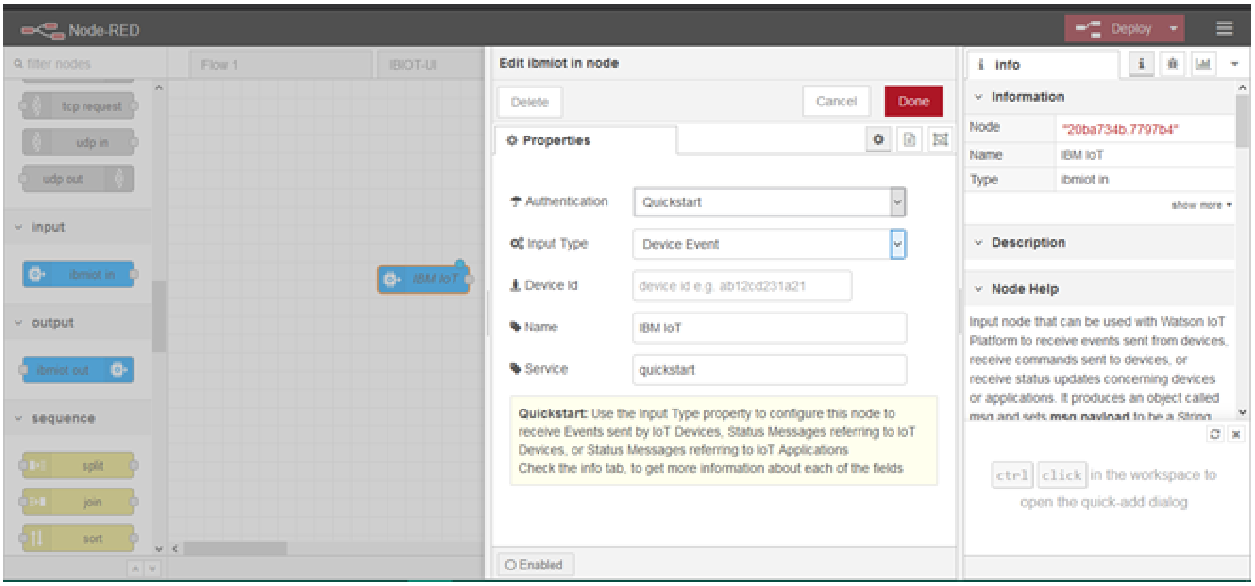


Node	"20ba734b.7797b4"
Name	IBM IoT
Type	ibmiot in

Input node that can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications. It produces an object called msg and sets msg.payload to be a String

Your flow configuration nodes are listed in the sidebar panel. It can be accessed from the menu or with ctrl-g c

Here you need to change it authentication from Quickstart to API key



Edit ibmiot in node

Properties

Authentication: Quickstart

Input Type: Device Event

Device Id: device id e.g. ab12cd231a21

Name: IBM IoT

Service: quickstart

Quickstart: Use the Input Type property to configure this node to receive Events sent by IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to IoT Applications
Check the info tab, to get more information about each of the fields

○ Enabled

Information

Node	"20ba734b.7797b4"
Name	IBM IoT
Type	ibmiot in

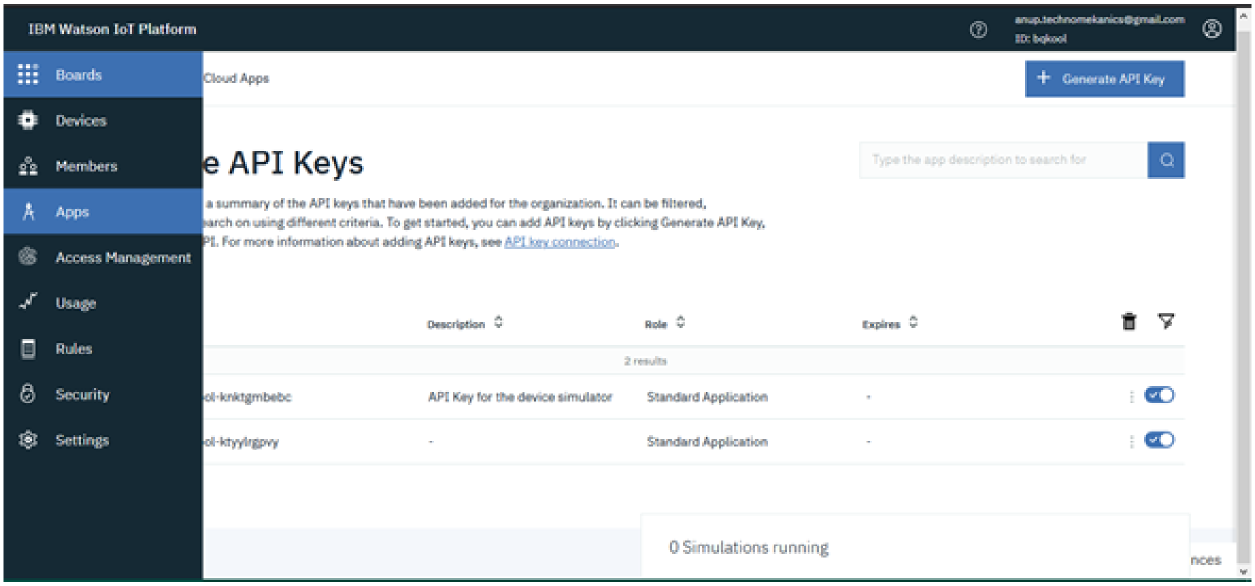
Description

Node Help

Input node that can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications. It produces an object called msg and sets **msg.payload** to be a String

Now again go to CLOUD IBM IOT website

Then go to **App** under navigation menu



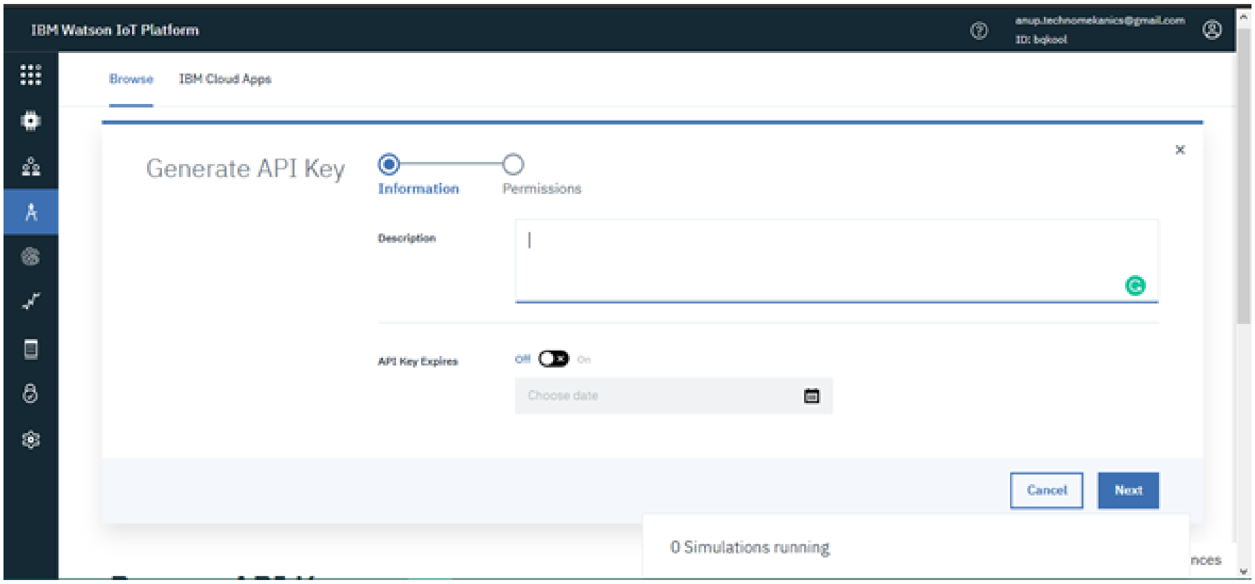
e API Keys

a summary of the API keys that have been added for the organization. It can be filtered, earch on using different criteria. To get started, you can add API keys by clicking Generate API Key, PI. For more information about adding API keys, see API key connection.

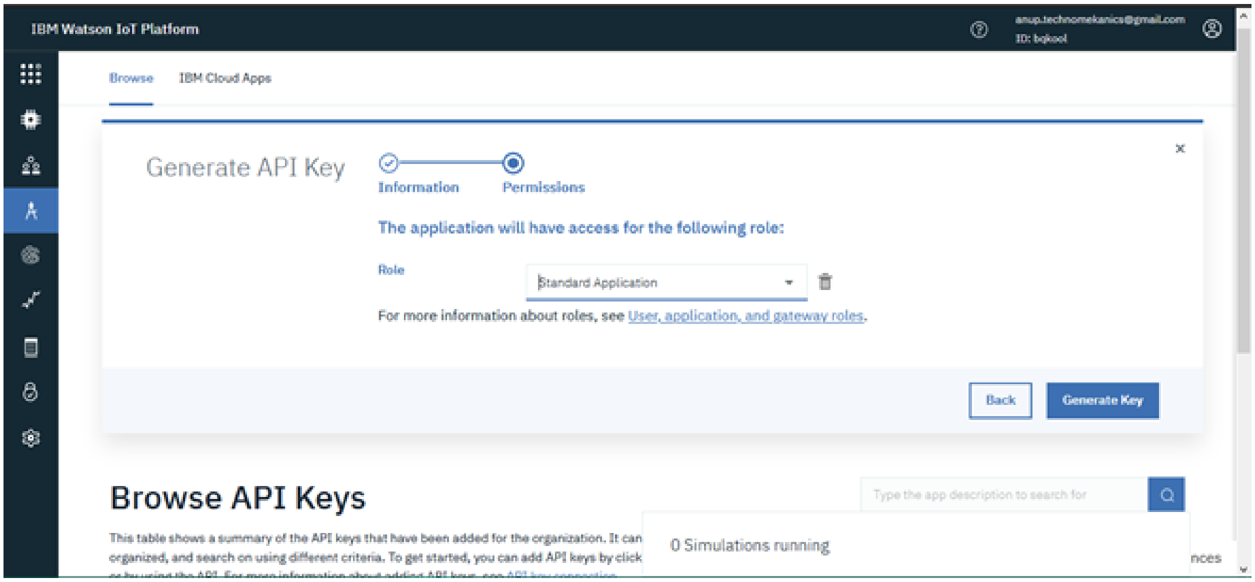
	Description	Role	Expires
2 results			
ol-knktgmbebc	API Key for the device simulator	Standard Application	-
ol-ktyylrgpvy	-	Standard Application	-

0 Simulations running

Then click on **generate key** then click on **Next**



Change the **role** to **Standard Application** then click on **Generate Key**



The application will have access for the following role:

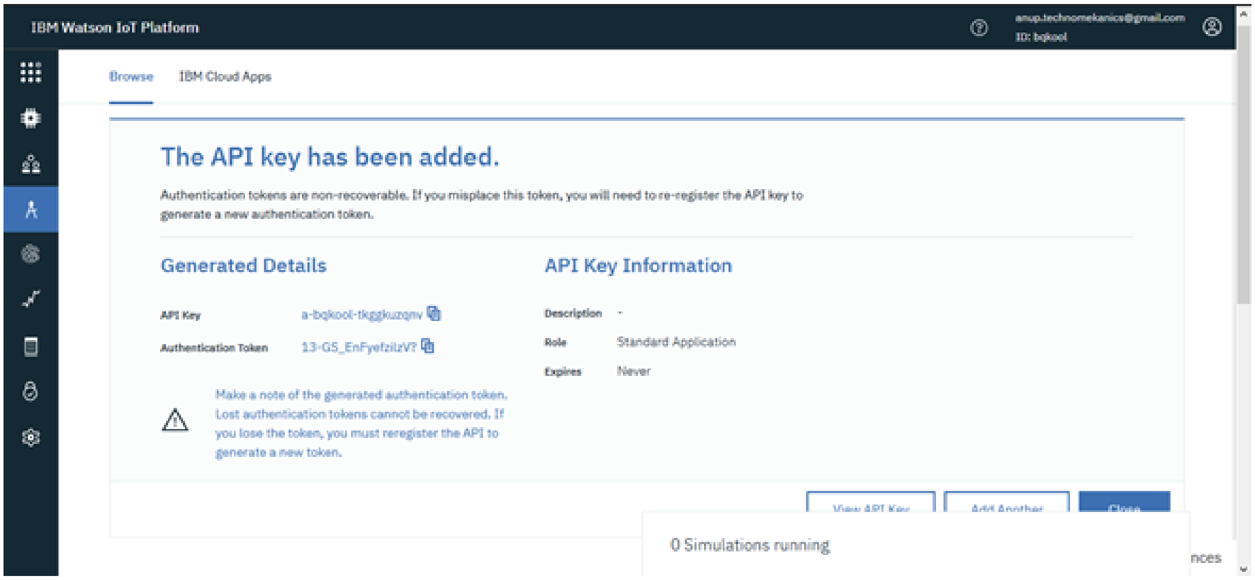
Role: Standard Application

For more information about roles, see User, application, and gateway roles.

Browse API Keys

This table shows a summary of the API keys that have been added for the organization. It can organized, and search on using different criteria. To get started, you can add API keys by click

Now copy the **API key** and **Authentication Token**



The API key has been added.

Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the API key to generate a new authentication token.

Generated Details

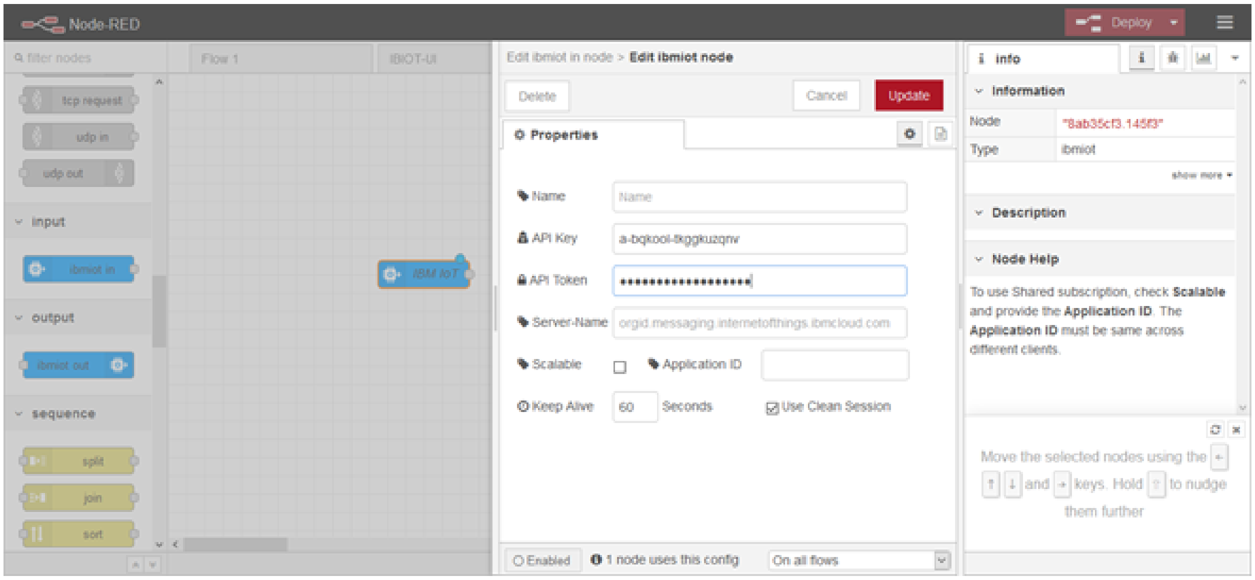
API Key	a-bqkool-tkggkuzqnv
Authentication Token	13-GS_EnFyefzilzV?

⚠ Make a note of the generated authentication token. Lost authentication tokens cannot be recovered. If you lose the token, you must reregister the API to generate a new token.

API Key Information

Description	-
Role	Standard Application
Expires	Never

Now go to **NodeRed** and change the **API key** and **auth token** after that click on **Update**



Name	Name
API Key	a-bqkool-tkggkuzqnv
API Token	••••••••••••••••••
Server-Name	orgid.messaging.internetofthings.ibmcloud.com
Scalable	□
Application ID	
Keep Alive	60 Seconds
☑ Use Clean Session	

○ Enabled 1 node uses this config On all flows

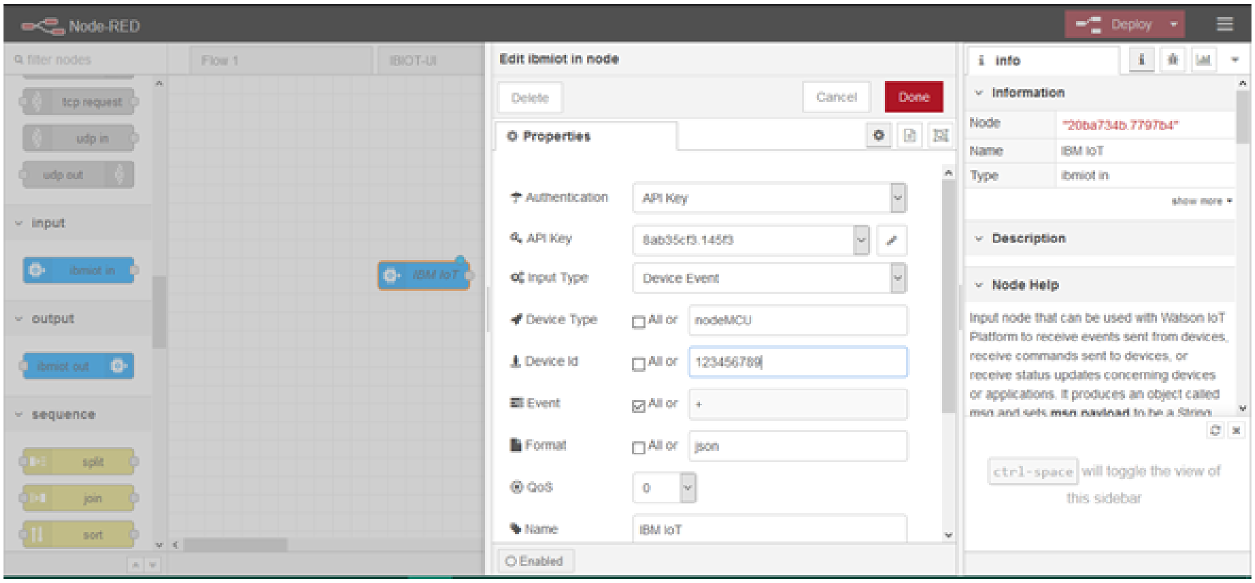
Information

Node	"8ab35cf3.145f3"
Type	ibmiot

To use Shared subscription, check **Scalable** and provide the **Application ID**. The **Application ID** must be same across different clients.

Move the selected nodes using the ← ↑ ↓ and → keys. Hold ⇧ to nudge them further

Then Update the **Device type** and **Device ID** then click on **Done** then click on **Deploy**



Edit ibmiot in node

Authentication: API Key

API Key: 8ab35cf3.145f3

Input Type: Device Event

Device Type: ☐ All or nodeMCU

Device Id: ☐ All or 123456789

Event: ☑ All or +

Format: ☐ All or json

QoS: 0

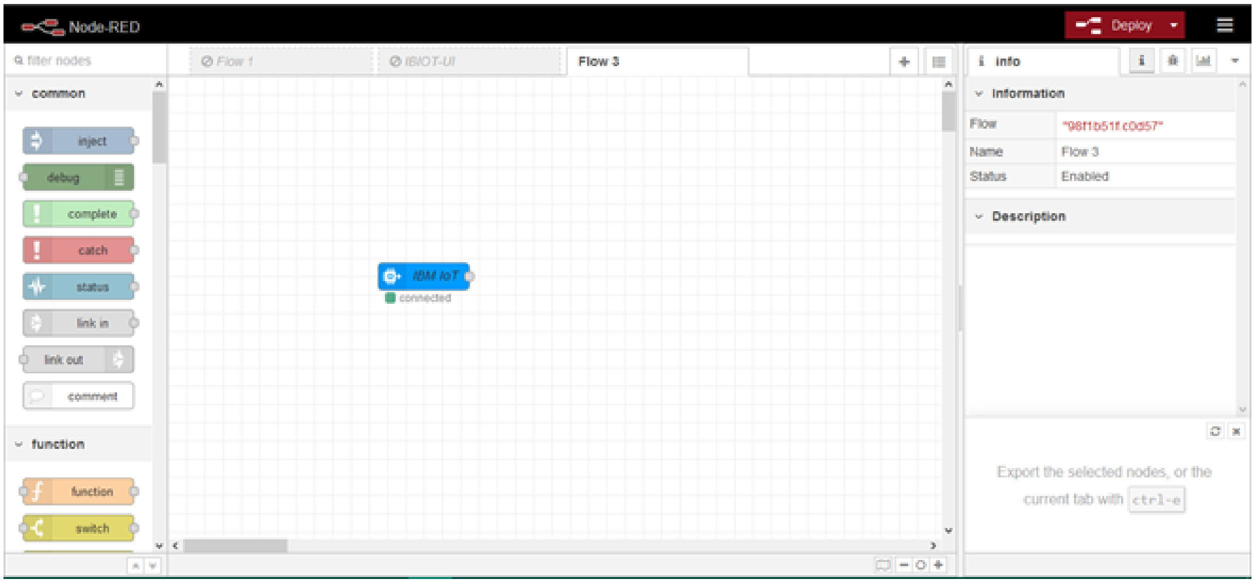
Name: IBM IoT

Node	"20ba734b.7797b4"
Name	IBM IoT
Type	ibmiot in

Input node that can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications. It produces an object called msg and sets msg.payload to be a String

ctrl-space will toggle the view of this sidebar

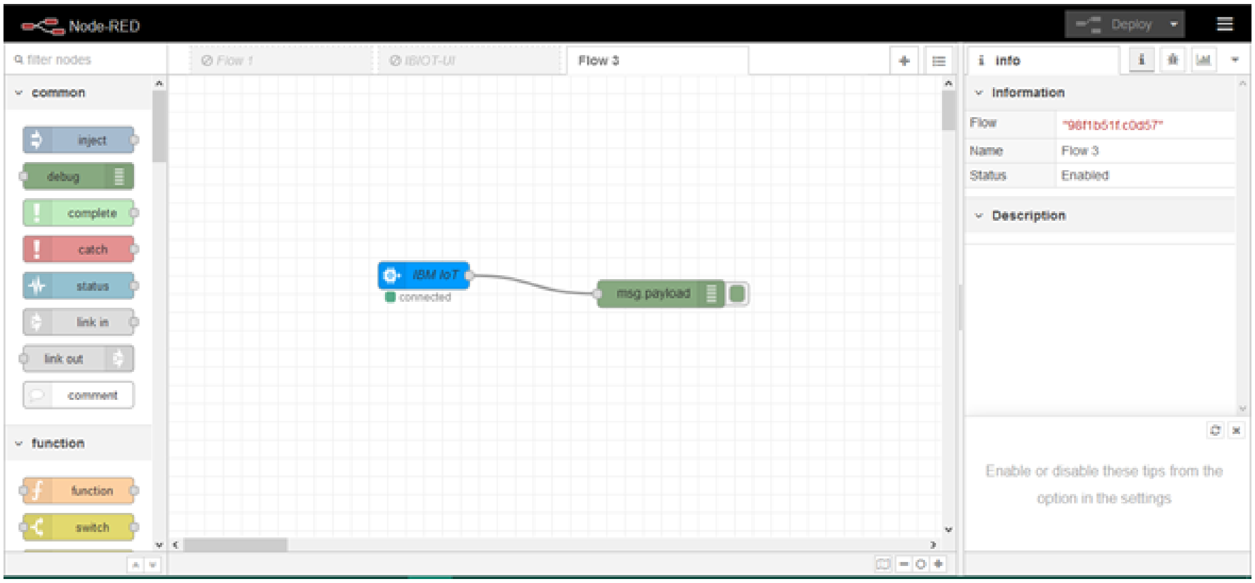
Then you can see it will show as **connected**.



Flow	"98f1b51f.c0d57"
Name	Flow 3
Status	Enabled

Export the selected nodes, or the current tab with ctrl-e

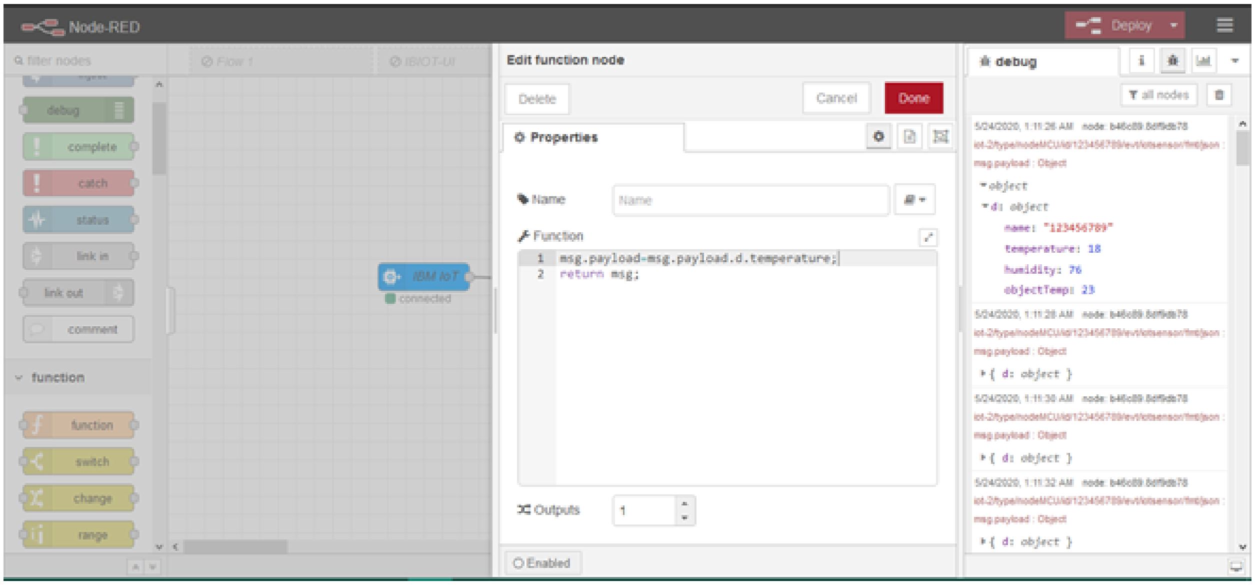
Then connect the **Debug** to IBM IoT then deploy



Flow	"98f1b51f.c0d57"
Name	Flow 3
Status	Enabled

Enable or disable these tips from the option in the settings

Then bring **one function node** and write code as shown below to **separate** the data



Name: Name

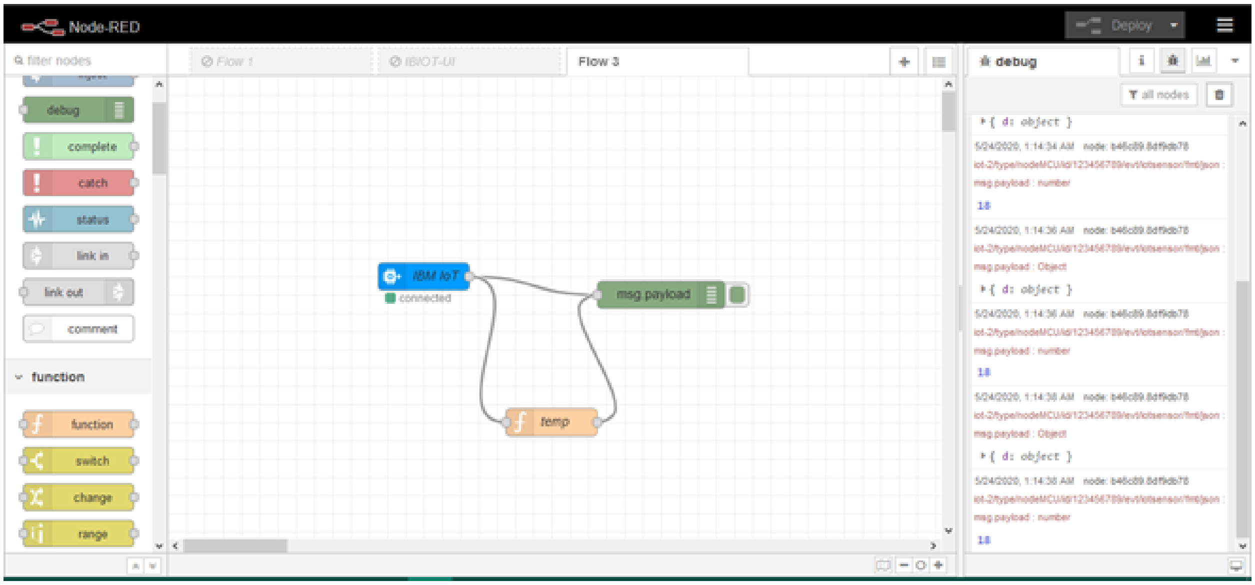
Function

```
msg.payload=msg.payload.d.temperature;
return msg;
```

Outputs: 1

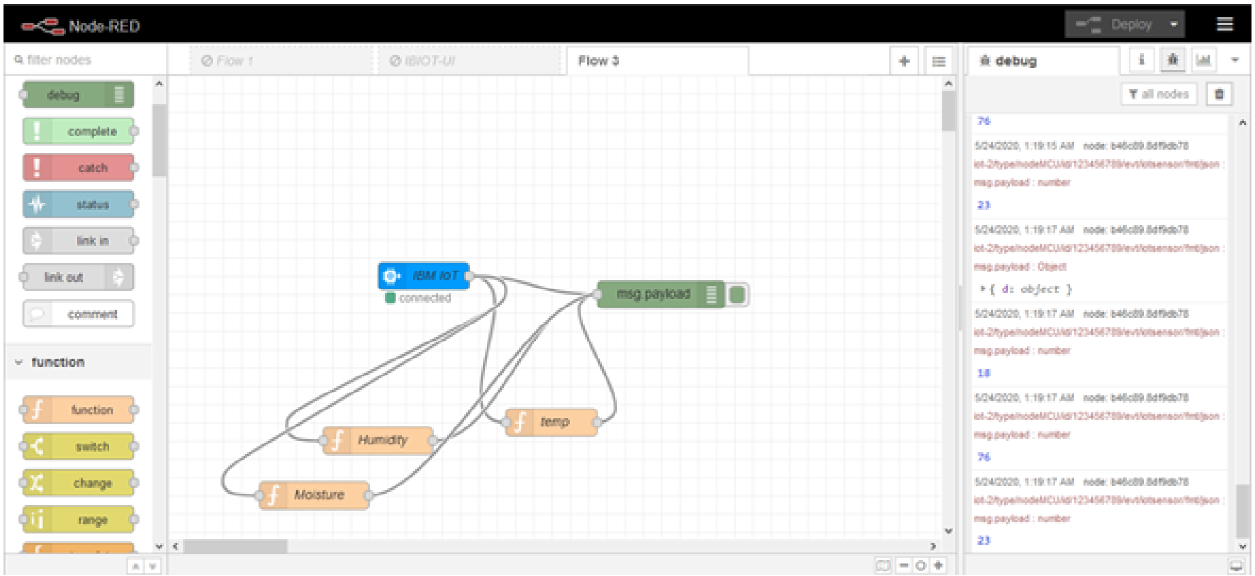
```
5/24/2020, 1:11:26 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▾object
  ▾d: object
     name: "123456789"
     temperature: 18
     humidity: 76
     objectTemp: 23
5/24/2020, 1:11:28 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:11:30 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:11:32 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
```

Then connect the temp function to the IBM IoT and Msg.payload.



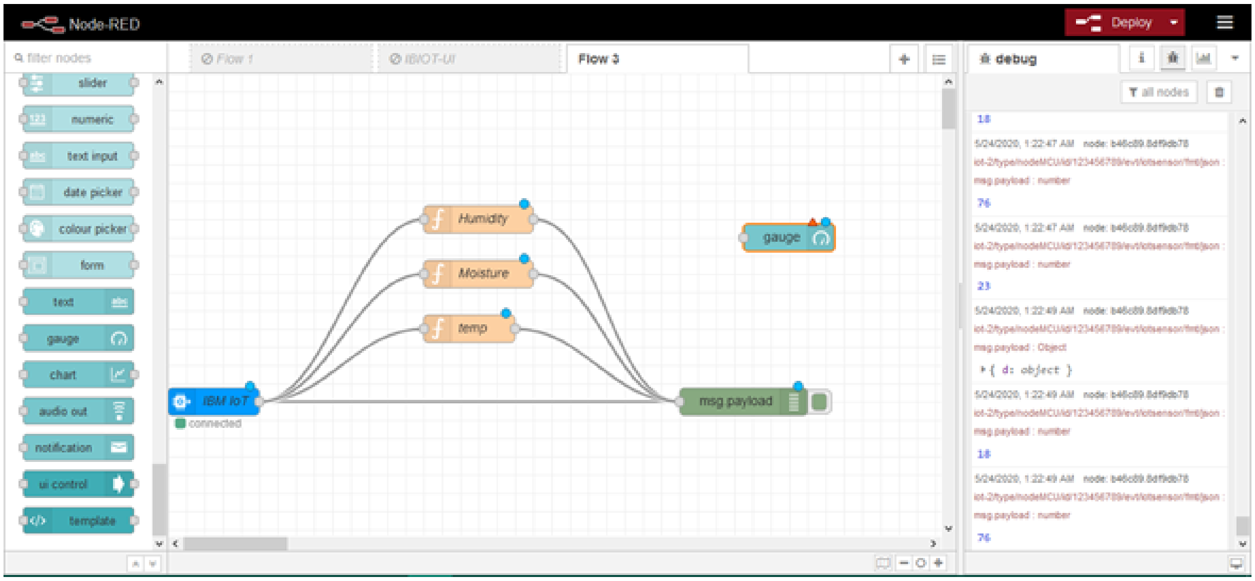
```
▸{ d: object }
5/24/2020, 1:14:34 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18
5/24/2020, 1:14:36 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:14:36 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18
5/24/2020, 1:14:38 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:14:38 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18
```

Then we will do the same step for **Moisture** and **Humidity.**



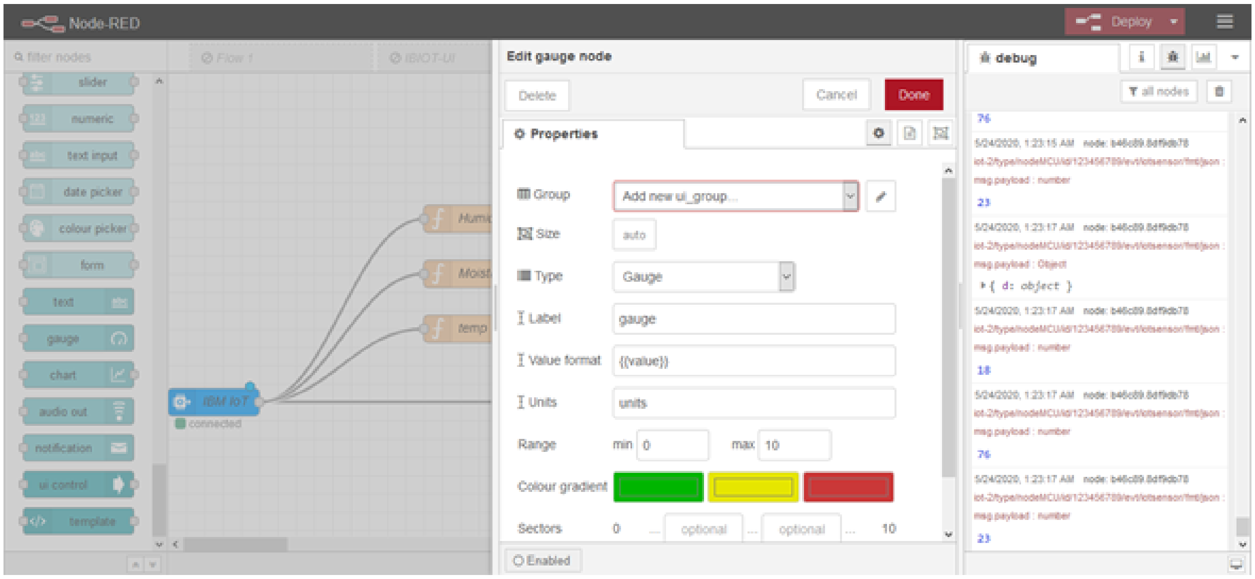
```
76
5/24/2020, 1:19:15 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
5/24/2020, 1:19:17 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:19:17 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18
5/24/2020, 1:19:17 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
76
5/24/2020, 1:19:17 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
```

Then we will add one new **gauge**



```
18
5/24/2020, 1:22:47 AM  node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
76
5/24/2020, 1:22:47 AM  node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
5/24/2020, 1:22:49 AM  node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:22:49 AM  node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18
5/24/2020, 1:22:49 AM  node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
76
```

Click on the group then select Add **new UI**

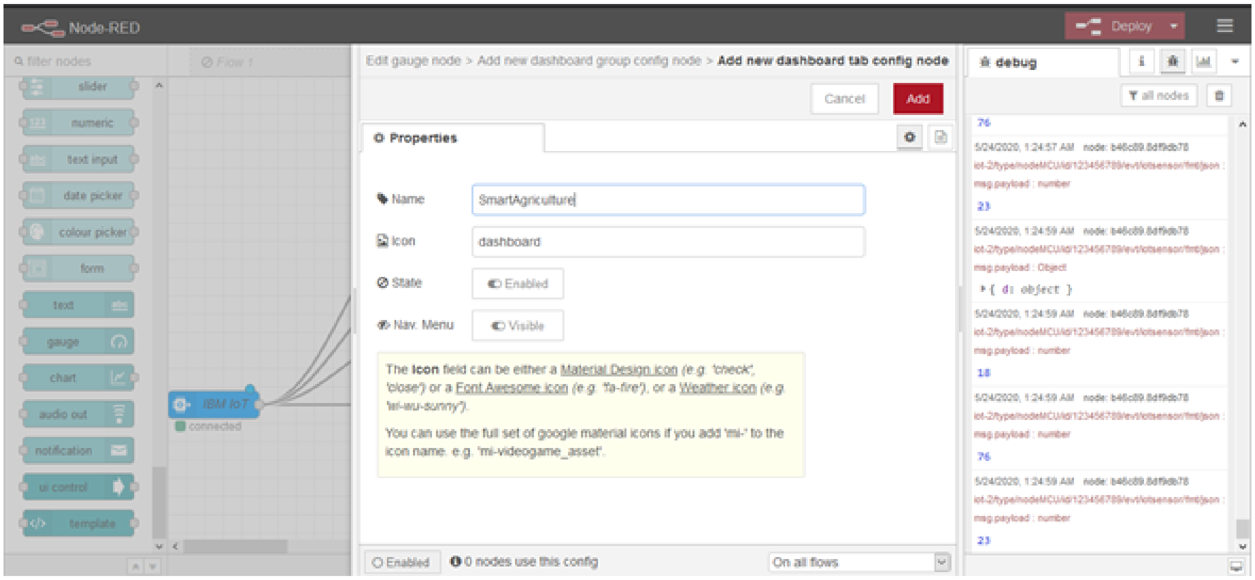


Edit gauge node

Group	Add new ui_group...
Size	auto
Type	Gauge
Label	gauge
Value format	{{value}}
Units	units
Range	min 0 max 10
Colour gradient	🟩 🟨 🟥
Sectors	0 ... optional ... optional ... 10

```
76
5/24/2020, 1:23:15 AM  node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
5/24/2020, 1:23:17 AM  node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:23:17 AM  node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18
5/24/2020, 1:23:17 AM  node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
76
5/24/2020, 1:23:17 AM  node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
```

Now do this change under **property** then click on **Add**



Edit gauge node > Add new dashboard group config node > **Add new dashboard tab config node**

Cancel Add

Properties

Name: SmartAgriculture

Icon: dashboard

State: Enabled

Nav. Menu: Visible

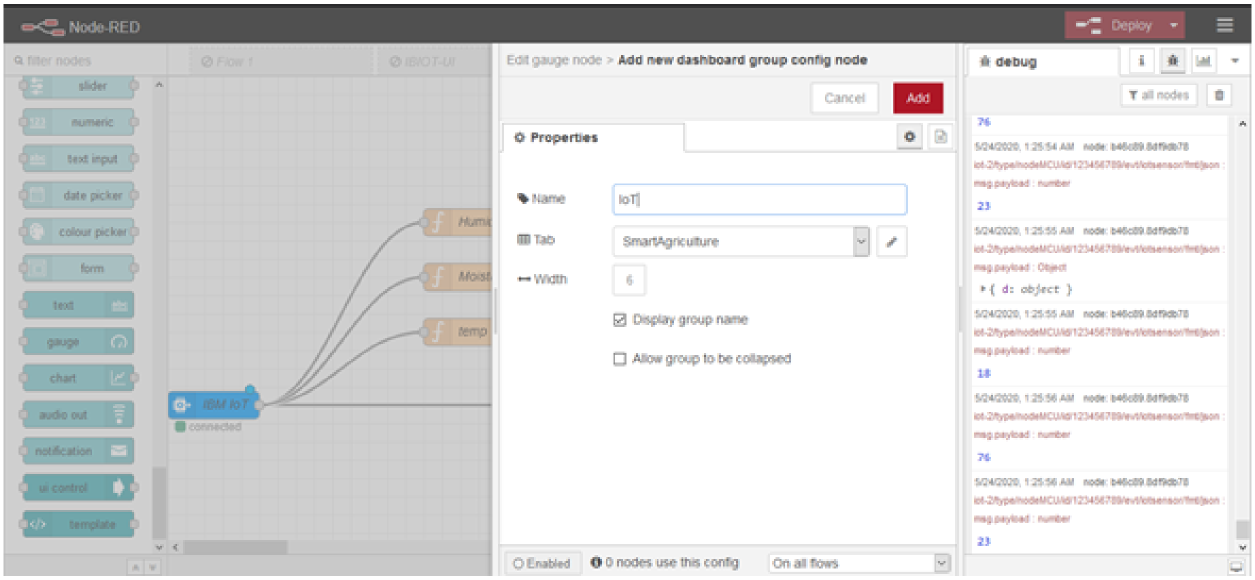
The **Icon** field can be either a Material Design icon *(e.g. 'check', 'close')* or a Font Awesome icon *(e.g. 'fa-fire')*, or a Weather icon *(e.g. 'wi-wu-sunny')*.

You can use the full set of google material icons if you add 'mi-' to the icon name. e.g. 'mi-videogame_asset'.

○ Enabled 0 nodes use this config On all flows

```
76
5/24/2020, 1:24:57 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
5/24/2020, 1:24:59 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:24:59 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18
5/24/2020, 1:24:59 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
76
5/24/2020, 1:24:59 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
```

Now change the name to IoT then click on **Add**



Edit gauge node > **Add new dashboard group config node**

Cancel Add

Properties

Name: IoT

Tab: SmartAgriculture

Width: 6

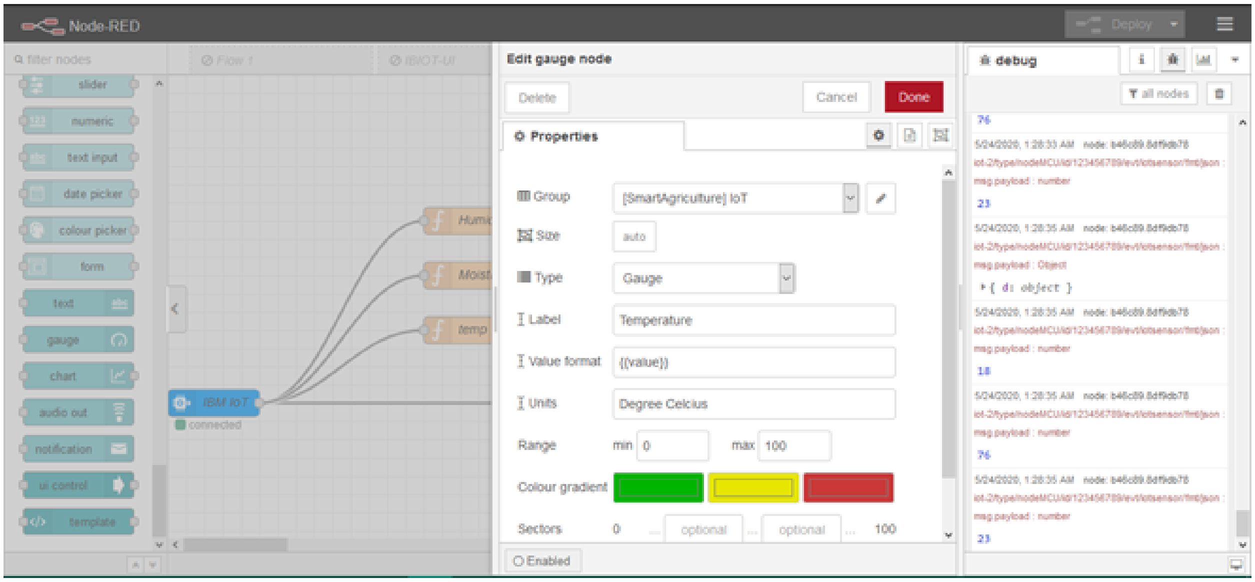
☑ Display group name

□ Allow group to be collapsed

○ Enabled 0 nodes use this config On all flows

```
76
5/24/2020, 1:25:54 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
5/24/2020, 1:25:55 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:25:55 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18
5/24/2020, 1:25:56 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
76
5/24/2020, 1:25:56 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
```

Afterward do this changes as mention below then **deploy**



Edit gauge node

Delete Cancel Done

Properties

Group: [SmartAgriculture] IoT

Size: auto

Type: Gauge

Label: Temperature

Value format: {{value}}

Units: Degree Celcius

Range: min 0 max 100

Colour gradient: 🟩 🟨 🟥

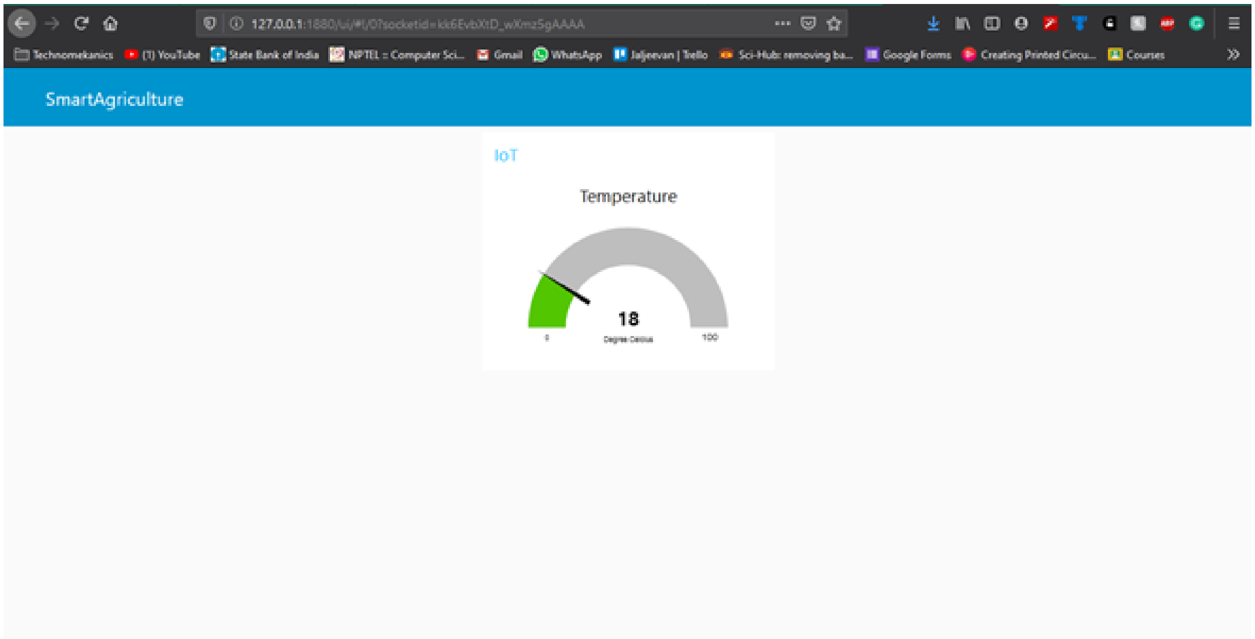
Sectors: 0 ... optional ... optional ... 100

○ Enabled

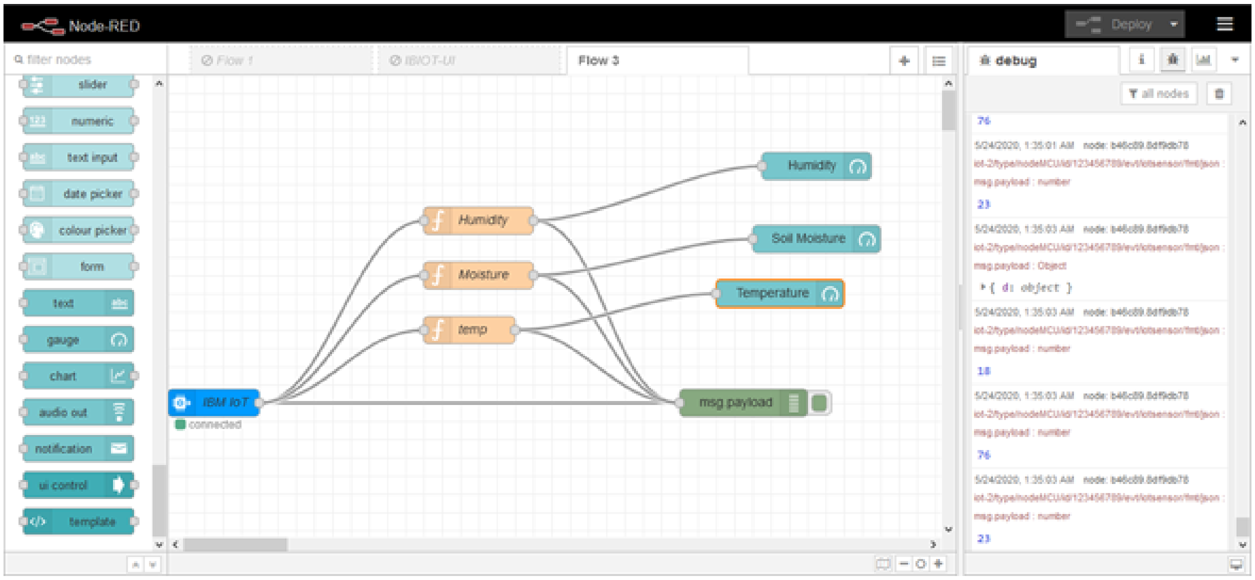
debug — all nodes

```
76
5/24/2020, 1:28:33 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json : msg.payload : number
23
5/24/2020, 1:28:35 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json : msg.payload : Object
▸{ d: object }
5/24/2020, 1:28:35 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json : msg.payload : number
18
5/24/2020, 1:28:35 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json : msg.payload : number
76
5/24/2020, 1:28:35 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json : msg.payload : number
23
```

After that go to http://127.0.0.1:1880/ui/

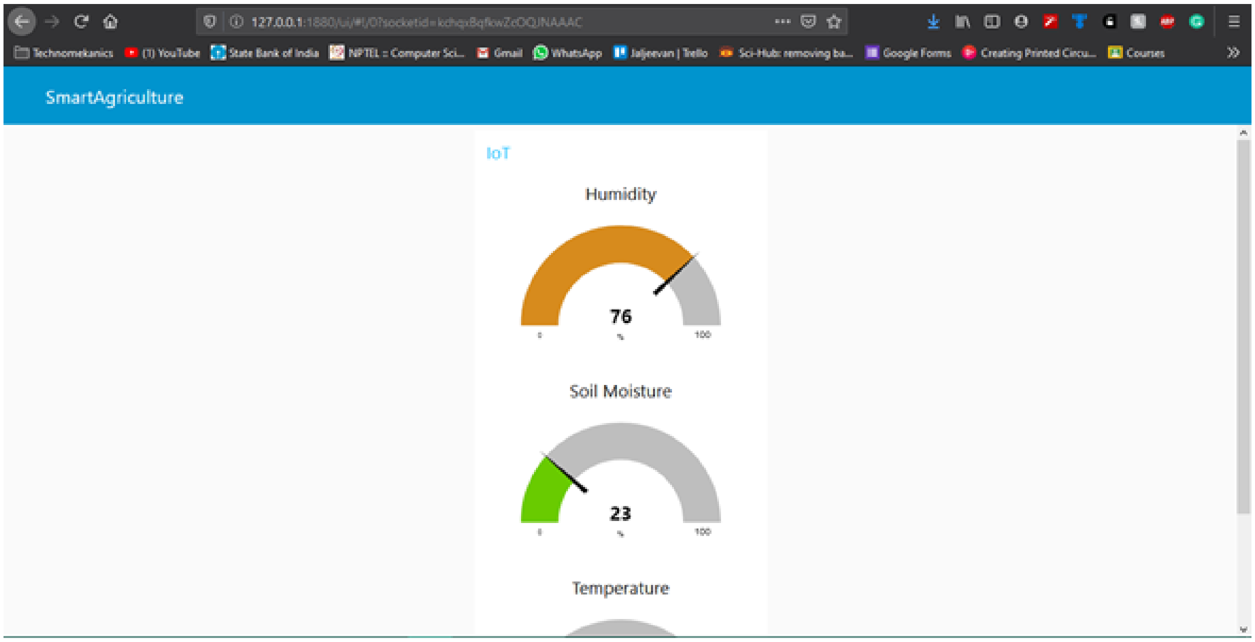


We will do the same for **Soil moisture and Humidity**

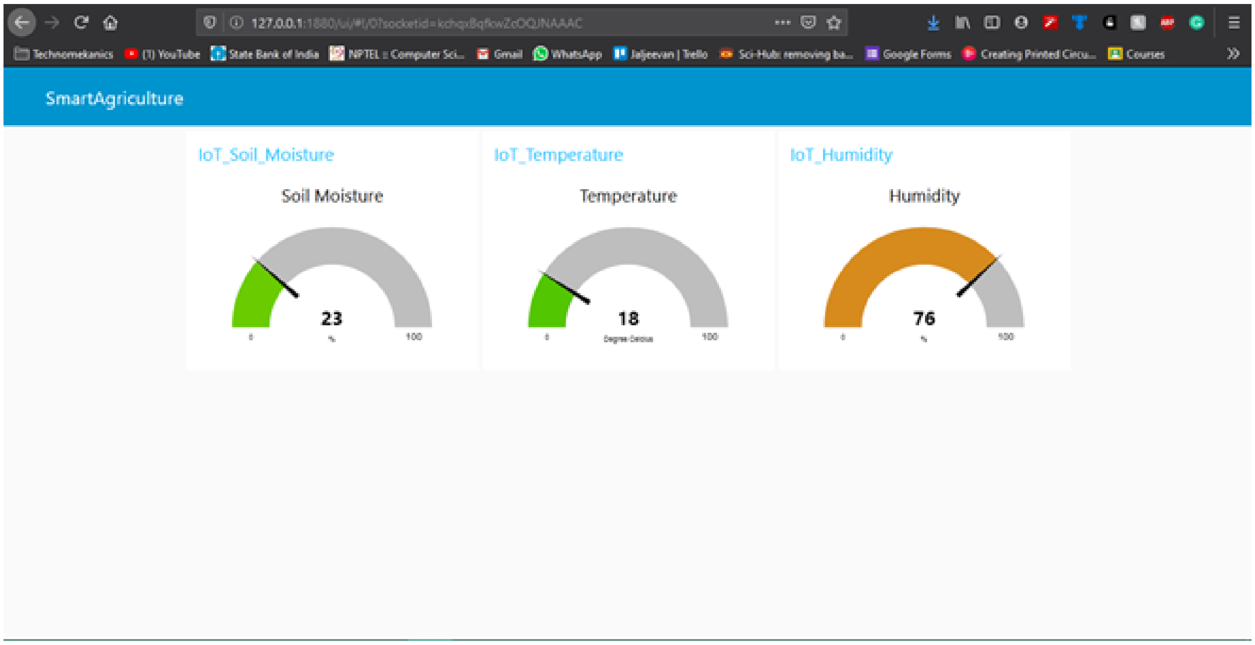


```
76
5/24/2020, 1:35:01 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
5/24/2020, 1:35:03 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:35:03 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18
5/24/2020, 1:35:03 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
76
5/24/2020, 1:35:03 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
```

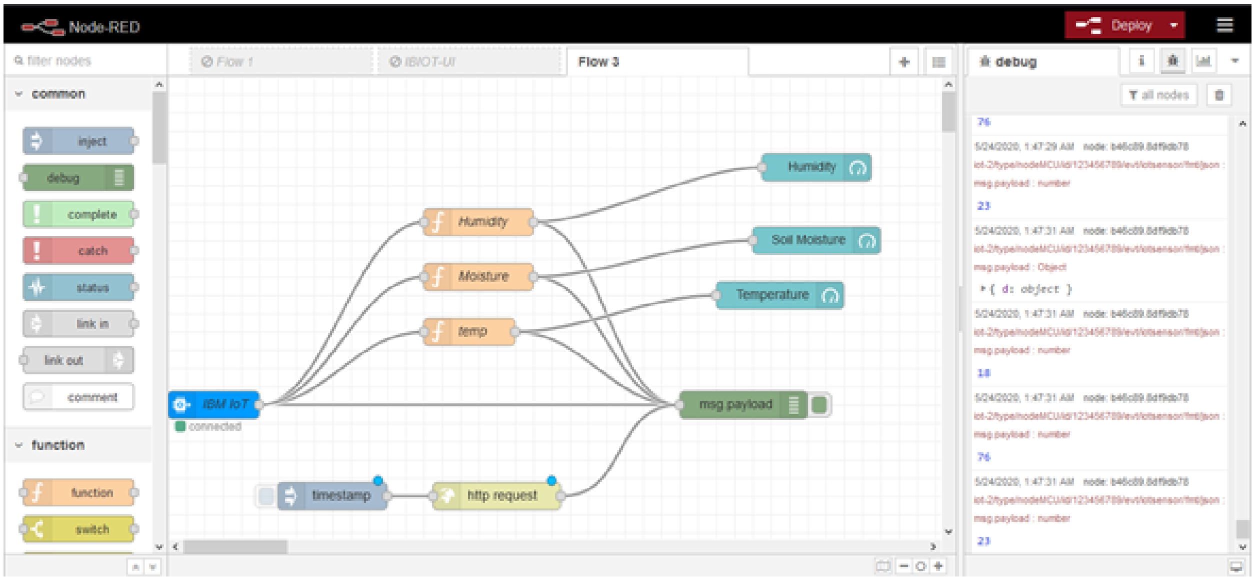
Then in the dashboard we can observe



Again we will do some changes in **UI setting** to see separately



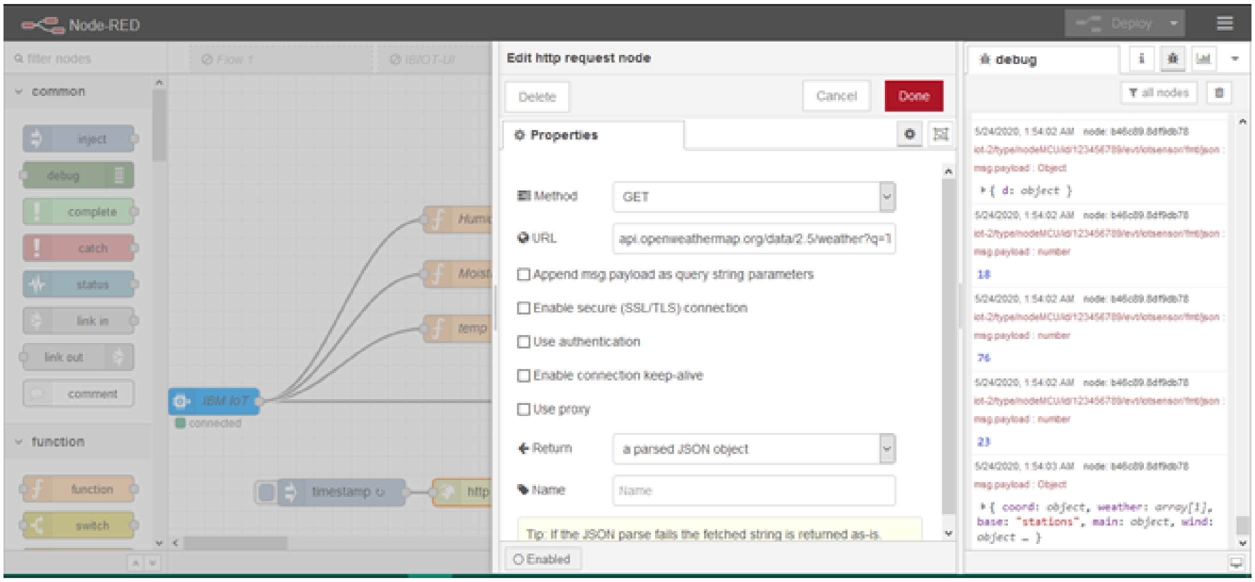
Next we will bring the HTTP request node and inject node and connect it with msg.payload



```
76
5/24/2020, 1:47:29 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
5/24/2020, 1:47:31 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:47:31 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18
5/24/2020, 1:47:31 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
76
5/24/2020, 1:47:31 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
```

After that we will configure the **HTTP** request by getting the API from https://openweathermap.org/ and my URL was api.openweathermap.org/data/2.5/weather?q=London,uk&APPID=df7f23761987ca590a660866c32ac38b

Then click on deploy



Edit http request node

Delete | Cancel | Done

Properties

Method: GET

URL: api.openweathermap.org/data/2.5/weather?q=1

□ Append msg.payload as query string parameters

□ Enable secure (SSL/TLS) connection

□ Use authentication

□ Enable connection keep-alive

□ Use proxy

Return: a parsed JSON object

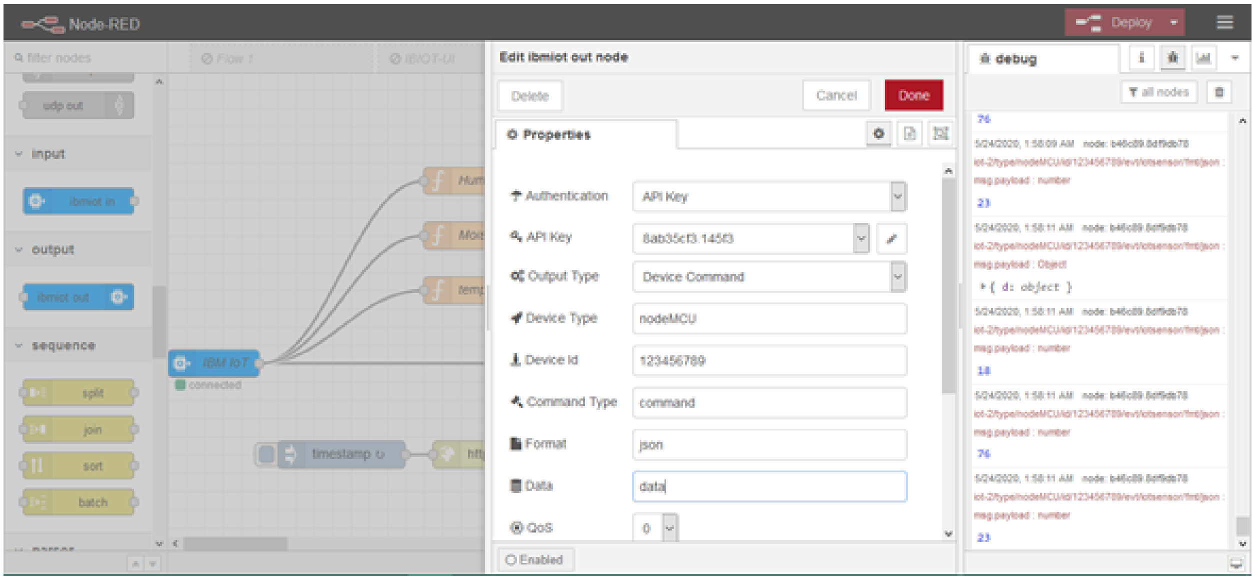
Name: Name

Tip: If the JSON parse fails the fetched string is returned as-is.

○ Enabled

```
5/24/2020, 1:54:02 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }

5/24/2020, 1:54:02 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18

5/24/2020, 1:54:02 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
76

5/24/2020, 1:54:02 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23

5/24/2020, 1:54:03 AM   node: b46c89.8df9db78
msg.payload : Object
▸{ coord: object, weather: array[1],
base: "stations", main: object, wind:
object … }
```

Now bring the IBM IoT output node then edit it as shown below then click on done then deploy

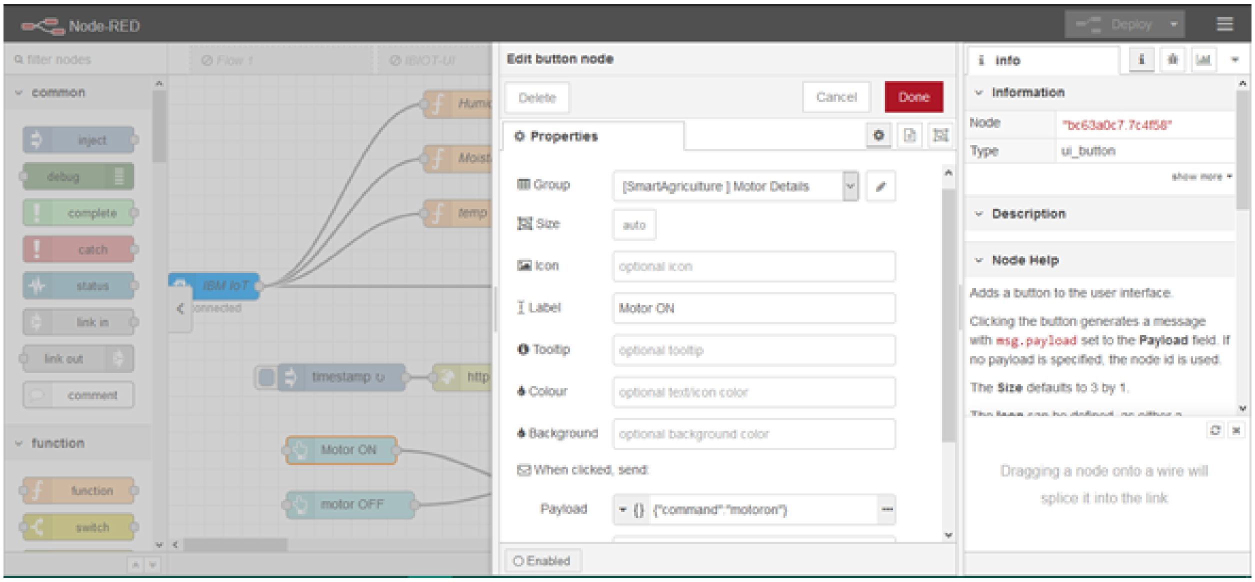


Edit ibmiot out node

- Authentication: API Key
- API Key: 8ab35cf3.145f3
- Output Type: Device Command
- Device Type: nodeMCU
- Device Id: 123456789
- Command Type: command
- Format: json
- Data: data
- QoS: 0

```
5/24/2020, 1:58:09 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
5/24/2020, 1:58:11 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
5/24/2020, 1:58:11 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
18
5/24/2020, 1:58:11 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
76
5/24/2020, 1:58:11 AM   node: b46c89.8df9db78
iot-2/type/nodeMCU/id/123456789/evt/iotsensor/fmt/json :
msg.payload : number
23
```

Then we will bring two button node and configure them as shown below:



Edit button node

- Group: [SmartAgriculture] Motor Details
- Size: auto
- Icon: optional icon
- Label: Motor ON
- Tooltip: optional tooltip
- Colour: optional text/icon color
- Background: optional background color
- When clicked, send:
- Payload: {} {"command":"motoron"}

Information

Node	"bc63a0c7.7c4f58"
Type	ui_button

Node Help

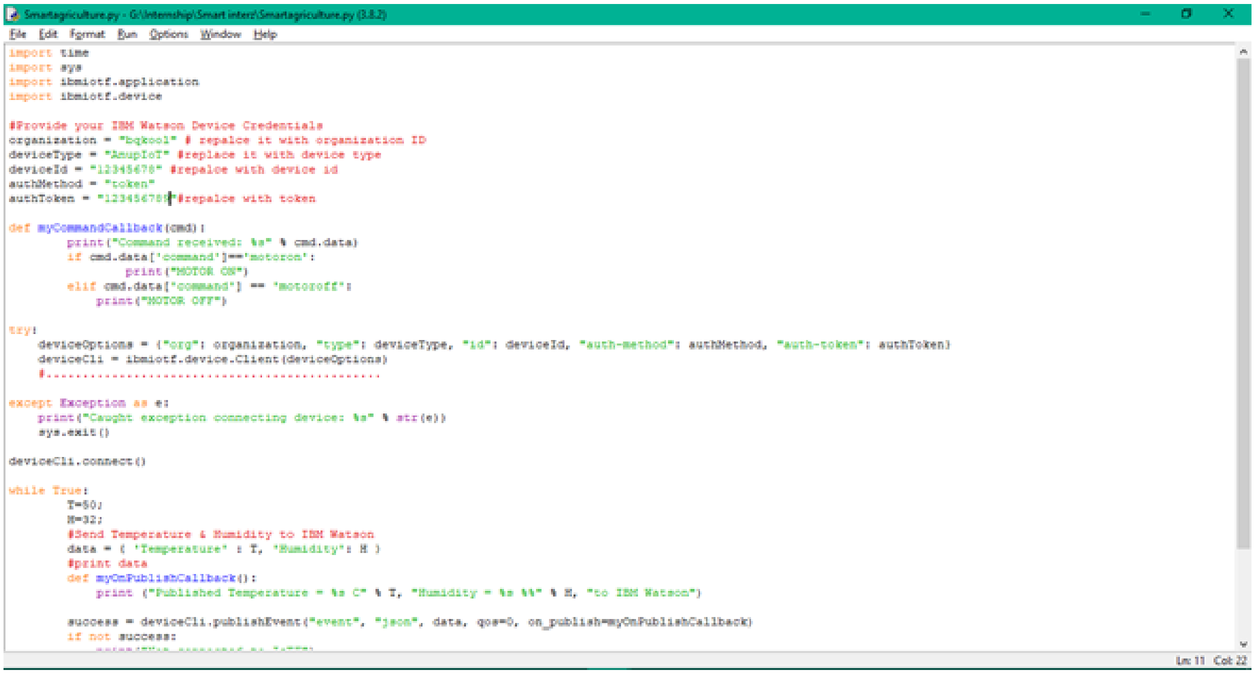
Adds a button to the user interface.

Clicking the button generates a message with msg.payload set to the **Payload** field. If no payload is specified, the node id is used.

The **Size** defaults to 3 by 1.

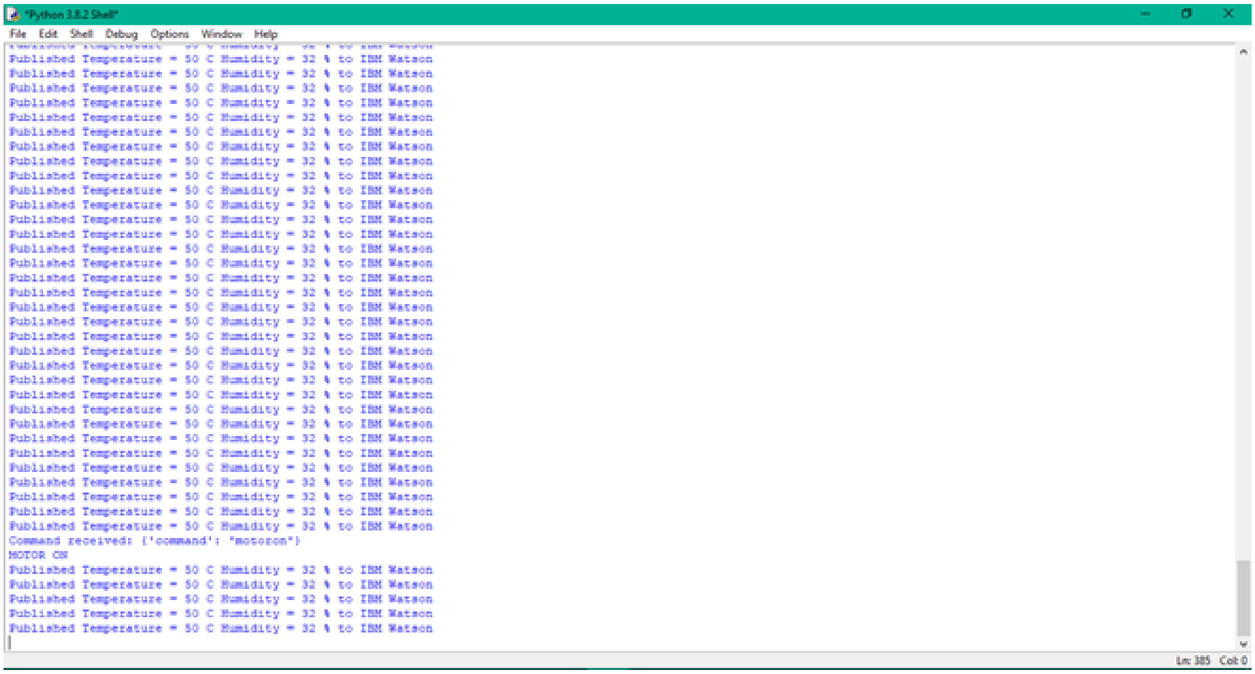
Dragging a node onto a wire will splice it into the link

After that go to IDLE python shell then write the code



```
import time
import sys
import ibmiotf.application
import ibmiotf.device

#Provide your IBM Watson Device Credentials
organization = "bqkool" # repalce it with organization ID
deviceType = "AnupIoT" #replace it with device type
deviceId = "12345678" #repalce with device id
authMethod = "token"
authToken = "123456789"#repalce with token

def myCommandCallback(cmd):
        print("Command received: %s" % cmd.data)
        if cmd.data['command']=='motoron':
                print("MOTOR ON")
        elif cmd.data['command'] == 'motoroff':
            print("MOTOR OFF")

try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
    #..............................................

except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()

deviceCli.connect()

while True:
        T=50;
        H=32;
        #Send Temperature & Humidity to IBM Watson
        data = { 'Temperature' : T, 'Humidity': H }
        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % T, "Humidity = %s %%" % H, "to IBM Watson")

        success = deviceCli.publishEvent("event", "json", data, qos=0, on_publish=myOnPublishCallback)
        if not success:
```

Now we will run the python code:



```
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Command received: {'command': 'motoron'}
MOTOR ON
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
Published Temperature = 50 C Humidity = 32 % to IBM Watson
```

Thank you