

Creating a Node-Red Web Application to view data in Separate Numerical form

Date	07 November 2022
Team ID	PNT2022TMID26635
Project Name	Project – IoT based Smart Crop Protection for Agriculture

➤ In IBM cloud dashboard, click on Cloud Foundry apps

The screenshot displays the IBM Cloud dashboard interface. The top navigation bar includes the IBM Cloud logo, a search bar, and user account information (JAGADESH S's Account). The main content area is titled "Resource list" and features a "Create resource" button. A sidebar on the left lists various resource categories: Compute (1), Containers (1+), Networking (0), Storage (0), AI / Machine Learning (0), Analytics (0), Blockchain (0), Databases (2+), Developer tools (3+), and Logging and monitoring (0). The main table lists resources with columns for Name, Group, Location, Product, Status, and Tags. A single resource is listed: "Node RED LPHBJ 2022-11-14" under the group "PNT2022TMID26635 / SMART CROP ...", located in "London", with the product "Node.js" and status "Started". The bottom of the image shows a Windows taskbar with the date and time as 20:57 on 15-11-2022.

Name	Group	Location	Product	Status	Tags
Node RED LPHBJ 2022-11-14	PNT2022TMID26635 / SMART CROP ...	London	Node.js	Started	—

- A new window appears where we need to NODE-RED SELDZ app created before

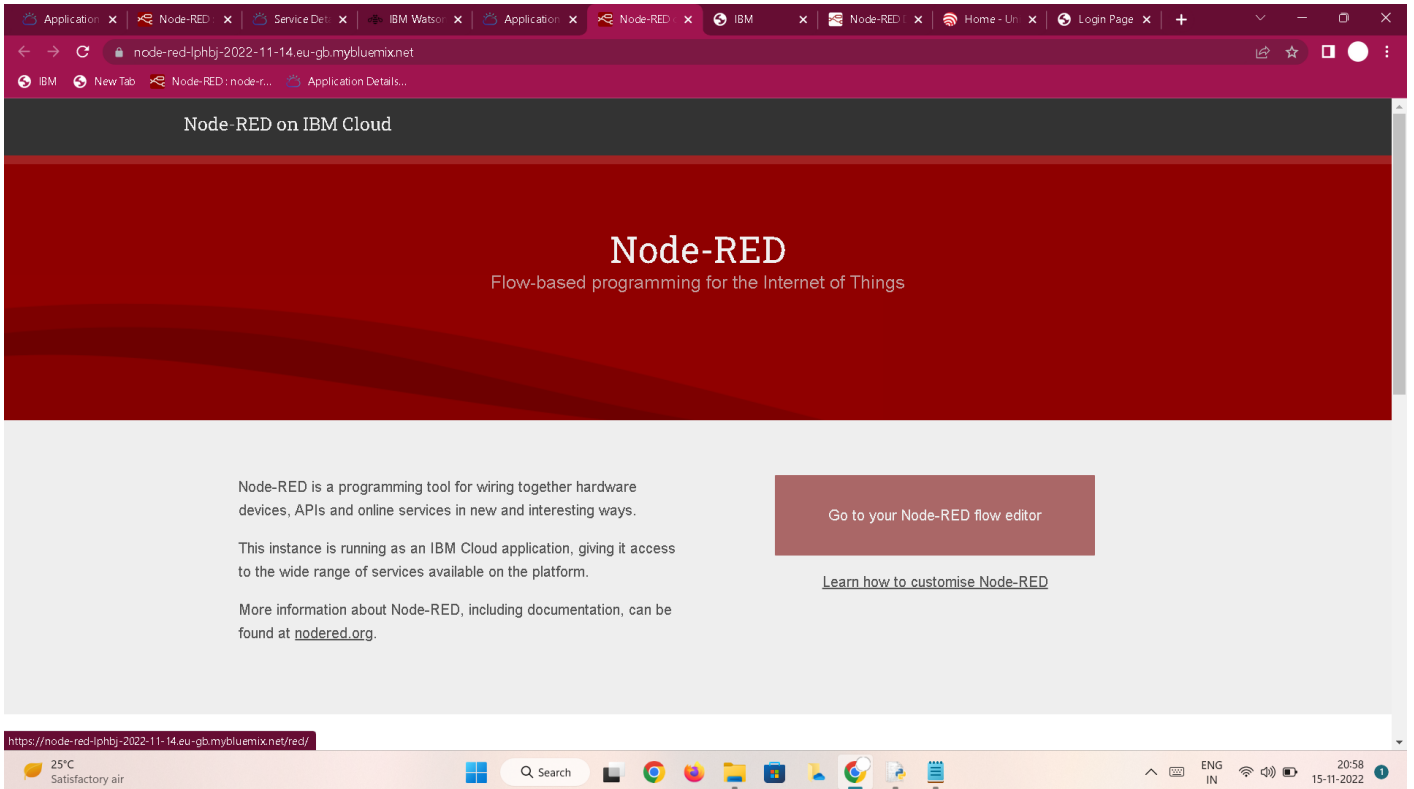
The screenshot displays the IBM Cloud Resource List page. The top navigation bar includes the IBM Cloud logo, a search bar, and links to Catalog, Manage, and JAGADESH S's Account. The main heading is 'Resource list', with a 'Create resource' button on the right. Below the heading is a table with columns: Name, Group, Location, Product, Status, and Tags. A search bar and filters are provided above the table. The table lists resources under various categories like Compute, Networking, Storage, AI / Machine Learning, Analytics, Blockchain, Databases, Developer tools, and Logging and monitoring. A resource named 'Node RED LPHBJ 2022-11-14' is highlighted, and a tooltip shows its details: PNT2022TMD26635 / SMART CROP ... London Node.js Started. The bottom of the page shows a system tray with weather information and a taskbar with various application icons.

Name	Group	Location	Product	Status	Tags
Node RED LPHBJ 2022-11-14	PNT2022TMD26635 / SMART CROP ...	London	Node.js	Started	-

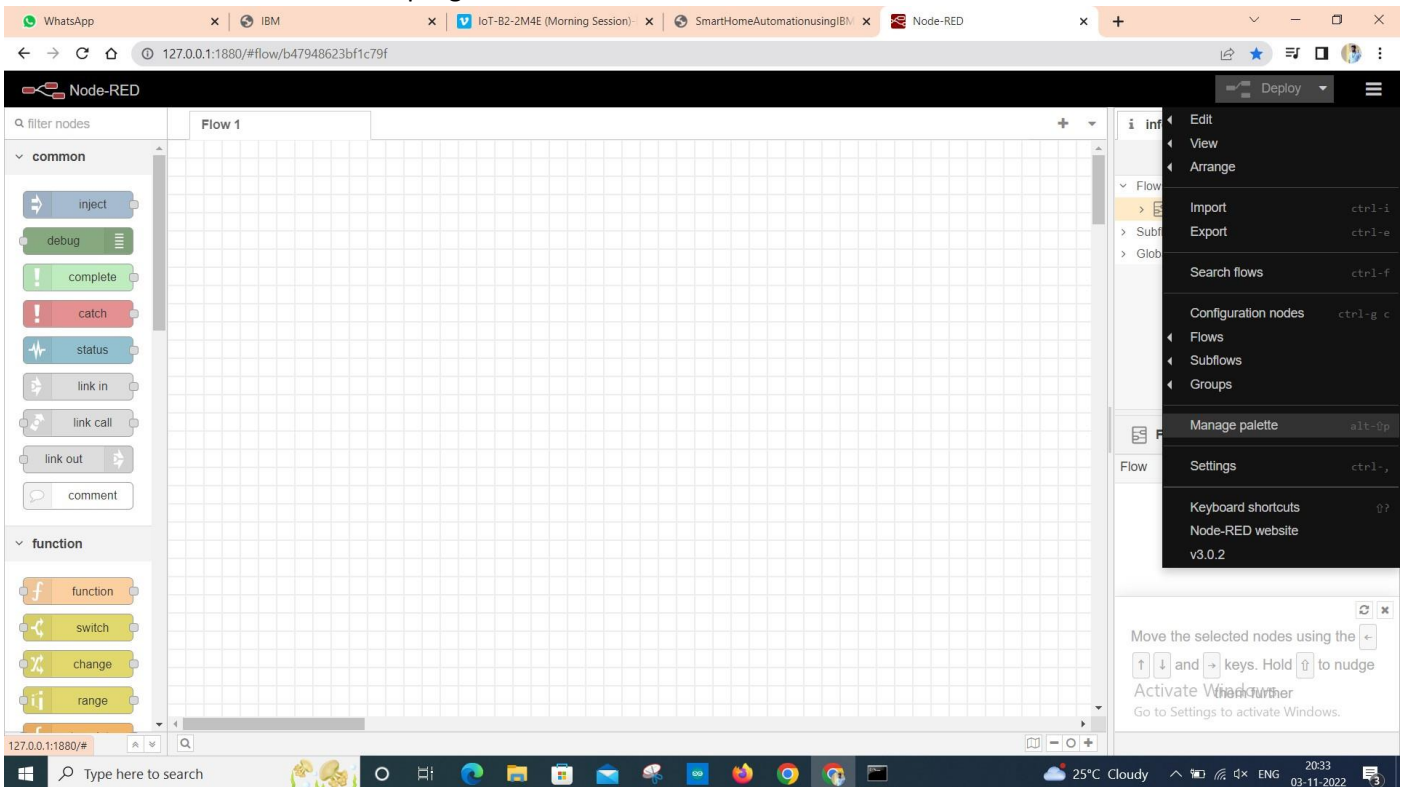
- Click on Visit App URL in Node RED SELDZ service dashboard.

[illegible]

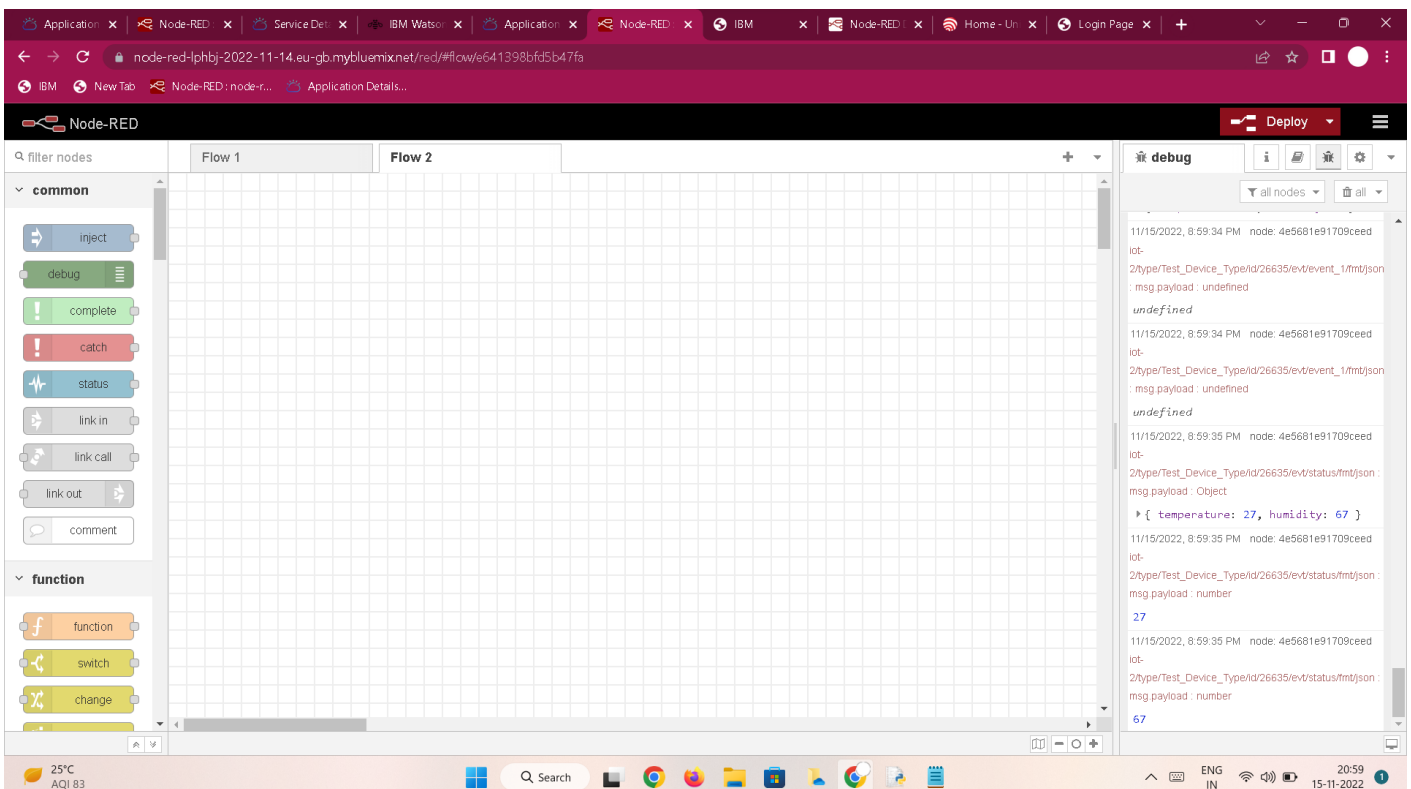
- Click on your Node-RED flow editor where you will be redirected to the Node-RED flow editor.



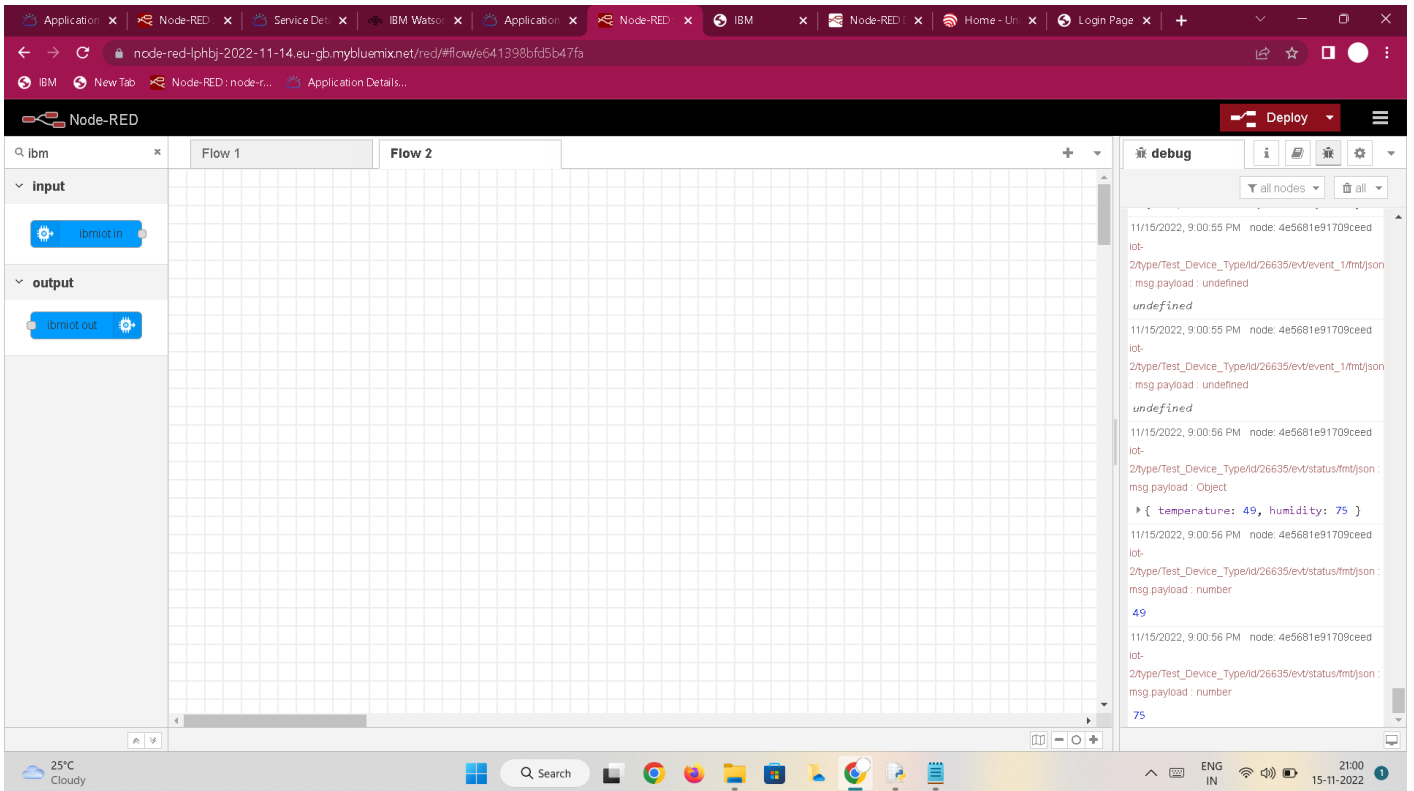
- To install IBM nodes in Node-red flow editor click on manage palette in the menu option which is on the top-right of the screen.



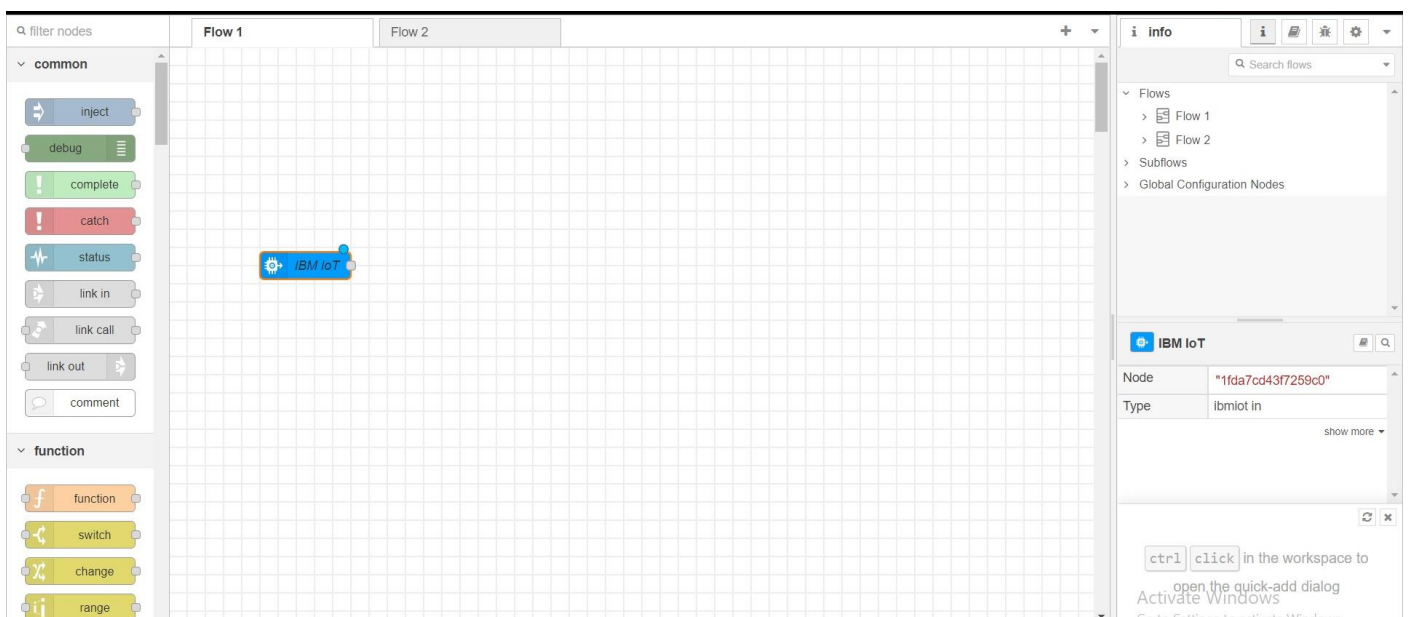
- In install section search for ibmiot and install the ibm nodes to flow editor.



- Search for IBM nodes in the filter nodes section



- To Retrieve the data from the IBM IoT platform by using Node-RED IBM IoT Input node and double click on the IBM IoT input node



- Select API Key from Authentication in properties.
 - In API Key paste API Key, API Token and server name and update it

The screenshot displays the Node-RED web interface in a browser. The main workspace shows a flow with an 'ibmiot in' node connected to an 'IBM IoT' node, which is then connected to 'Temp' and 'Humidity' nodes. The 'Edit ibmiot in node' panel is open, showing the following configuration:

- Authentication:** API Key
- API Key:** 20d32ce25d8a591c
- Input Type:** Device Event
- Device Type:** ☐ All or Test_Device_Type
- Device Id:** ☐ All or 26635
- Event:** ☒ All or +
- Format:** ☐ All or json
- QoS:** 0
- Name:** IBM IoT
- Service:** registered

A yellow tooltip at the bottom of the properties panel reads: "Use the Input Type property to configure this node to receive". The right sidebar shows a debug console with several log messages, including a JSON object: `{ temperature: -9, humidity: 1 }`. The bottom status bar shows the system clock as 21:01 on 15-11-2022.

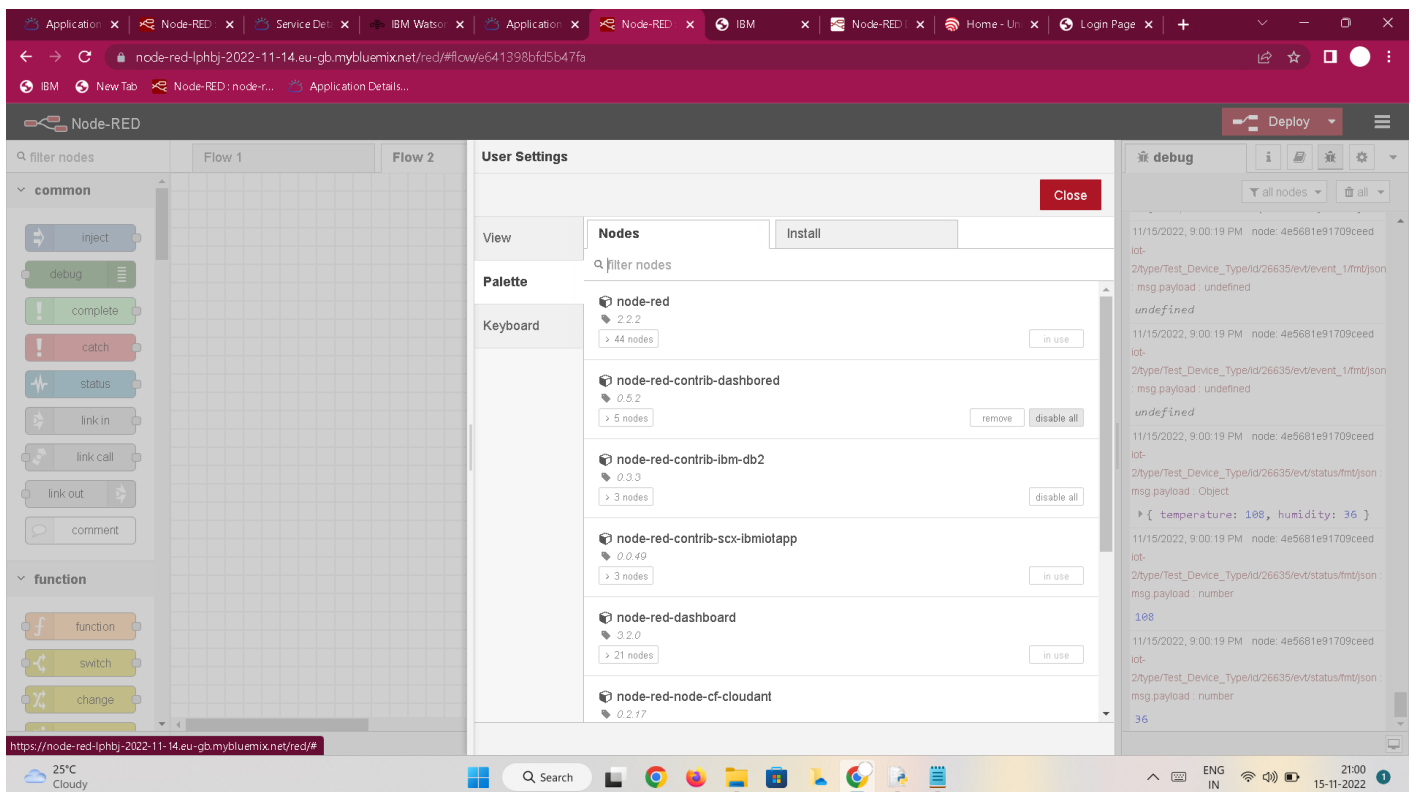
- Also update your input type as event, Device type, Device ID, command and format in the properties section and click on Done

To generate API Key go to IBM IoT platform

- In Apps Section -> Click on Generate API Key
- Click on Deploy option to check the connection status. If the status is disconnected check for IBM IoT properties and try again.

Place the debug node in the flow editor and click on deploy to see the temperature and humidity value in the debug tab

- Install the dashboard node from the manage pallet to create a UI to display temperature and humidity values in the Dashboard



- Drag and place the function node and gauge node in the flow editor to separate the temperature, humidity and soil moisture value and also for motor on and off.

The screenshot displays the Node-RED web interface in a browser. The top navigation bar includes tabs for 'Service Details - IBM Cloud', 'IBM Watson IoT Platform', 'Application Details - IBM C...', 'Node-RED: node-red-lph...', 'IBM', and 'Node-RED Dashboard'. The address bar shows the URL: `node-red-lphbj-2022-11-14.eu-gb.mybluemix.net/red/#/cw789a662aa366dc8`. The main workspace is titled 'Flow 1' and contains a flow diagram. The flow starts with an 'IBM IoT' node (labeled 'connected') that feeds into three function nodes: 'Soil Moisture', 'Temp', and 'Humidity'. These function nodes are connected to corresponding gauge nodes: 'Soil Moisture', 'Temperature', and 'Humidity'. Below these, there are 'MOTOR ON' and 'MOTOR OFF' nodes, which are connected to a 'control function' node. The 'control function' node is also connected to an 'IBM IoT' node (labeled 'connected'). The right sidebar shows a 'debug' console with a list of messages. The messages are JSON objects containing sensor data: `{ Temperature: 87, Humidity: 46, Soil Moisture: 74 }`. The bottom status bar shows the system time as 21:38 on 15-11-2022, along with network and battery icons.

- Double click on function and update the details as follow, ➤ Type `msg.payload=msg.payload.Temperature` in one function.
- Type `msg.payload=msg.payload.Humidity` in another function
- Type `msg.payload=msg.payload.soil_moisture`
- Type `msg.payload=msg.payload.Pressure`
- To separate the humidity and temperature values from payload and click deploy

The screenshot displays the Node-RED web interface in a browser. The main workspace shows a flow named 'Flow 1' with the following components:

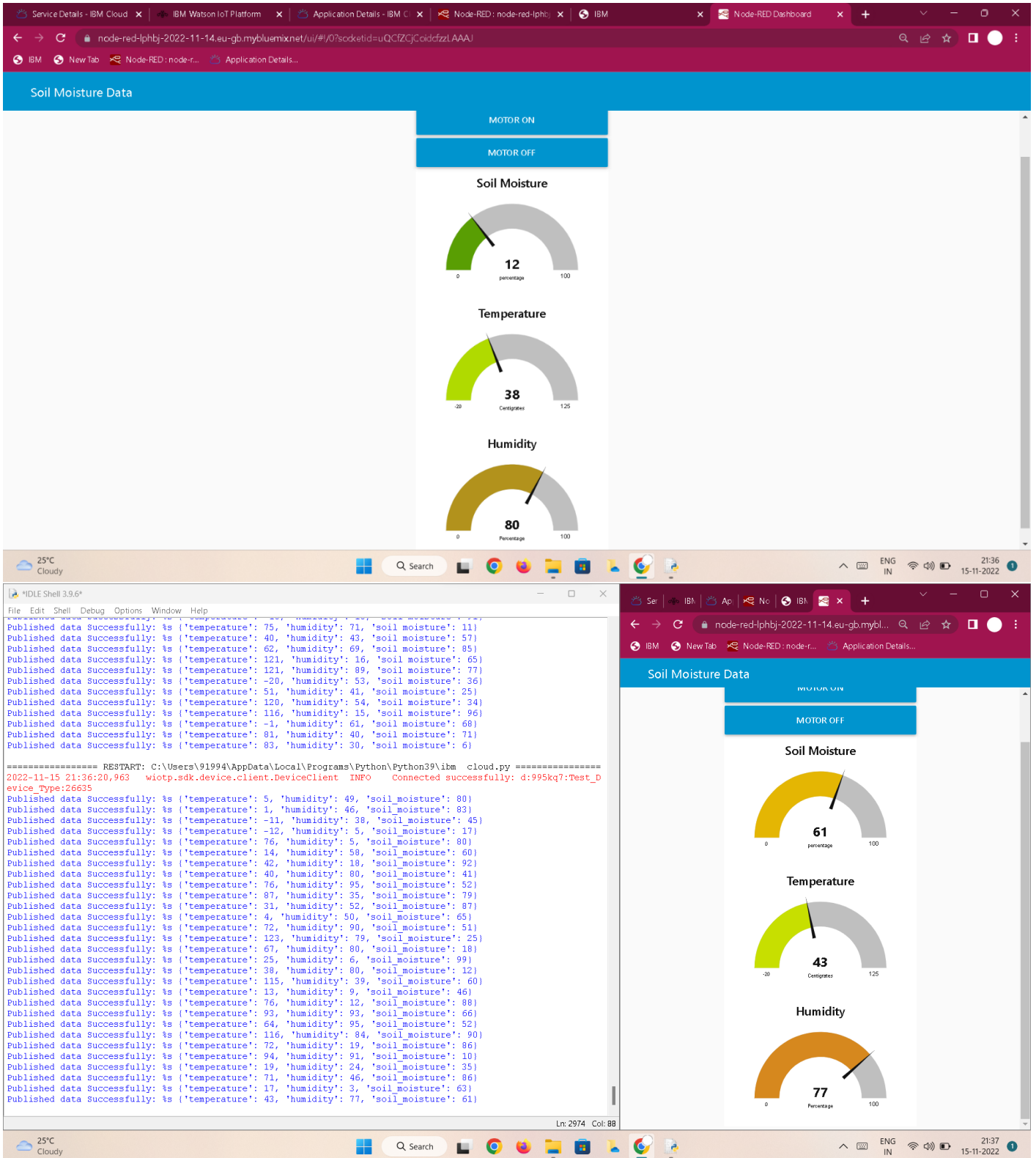
- Inputs:** A 'gauge' node on the left sidebar and an 'IBM IoT' node (connected) at the start of the flow.
- Processing Functions:** Three function nodes labeled 'Soil Moisture', 'Temp', and 'Humidity' are connected to the 'IBM IoT' node. Each function node has a configuration box where the code `msg.payload=msg.payload.[SensorName]` is entered.
- Outputs:** The outputs of the three function nodes are connected to 'msg.payload' nodes, which then connect to 'Temperature' and 'Humidity' gauge nodes.
- Control Logic:** Below the main flow, there is a 'control function' node connected to 'MOTOR ON' and 'MOTOR OFF' buttons, which in turn connect to another 'IBM IoT' node (connected).

The right-hand 'debug' console shows the following log entries:

```

iot-
2/type/Test_Device_TypeId26635/event_1/fmt/json:
msg.payload: Object
{ Temperature: 40, Humidity: 1,
  Soil Moisture: 41 }
11/15/2022, 9:37:39 PM node: 4e5681e91709ceed
iot-
2/type/Test_Device_TypeId26635/event_1/fmt/json:
msg.payload: undefined
undefined
11/15/2022, 9:37:39 PM node: 4e5681e91709ceed
iot-
2/type/Test_Device_TypeId26635/event_1/fmt/json:
msg.payload: undefined
undefined
11/15/2022, 9:37:39 PM node: 4e5681e91709ceed
iot-2/type/Test_Device_TypeId26635/event_status/fmt/json:
msg.payload: Object
{ temperature: 45, humidity: 93,
  soil_moisture: 50 }
11/15/2022, 9:37:40 PM node: 4e5681e91709ceed
iot-2/type/Test_Device_TypeId26635/event_status/fmt/json:
msg.payload: number
45
  
```

- Select gauge function and these nodes to temperature, pressure, soil moisture gas and humidity
- Edit temperature, soil moisture and humidity nodes and deploy it.
- After editing the nodes, deploy it



RESULT:

Thus, the Node-Red Web Application is created successfully.

.