

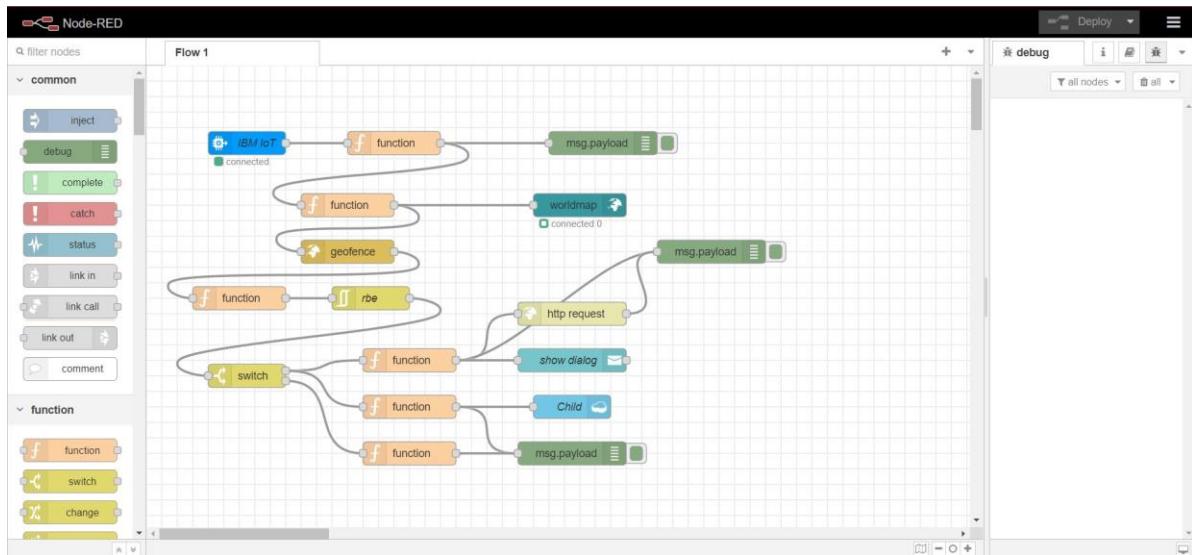
Sprint-3

IOT Based Safety Gadget for Child Safety Monitoring & Notification

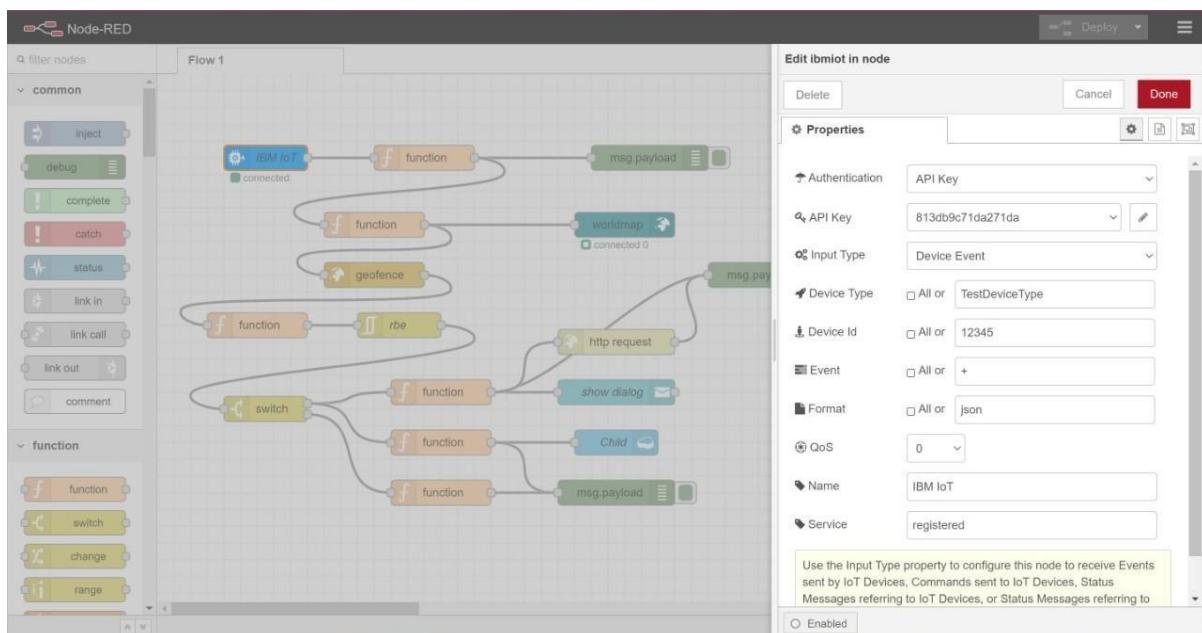
TEAM ID: PNT2022TMID06213

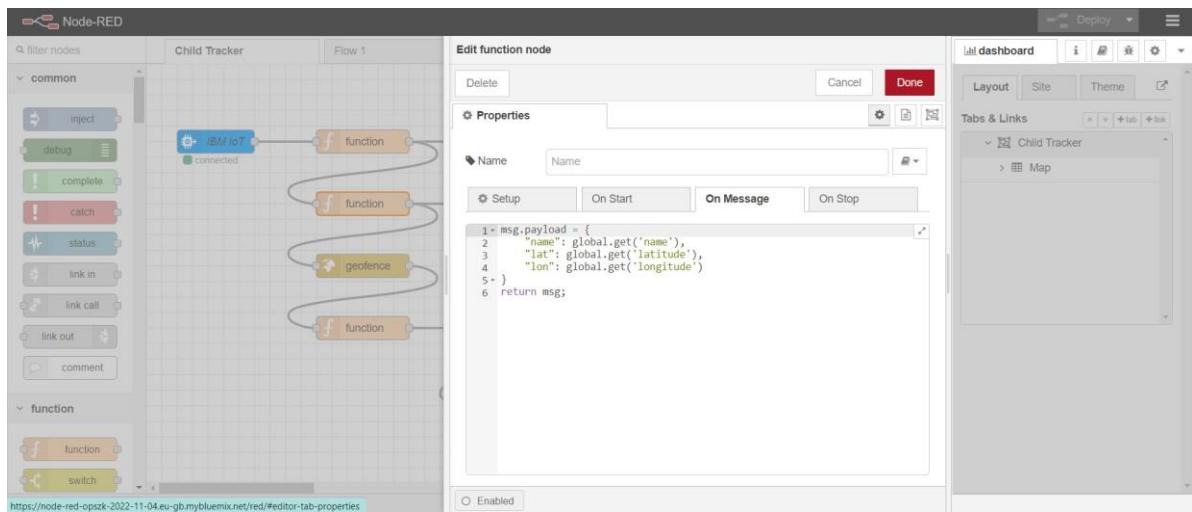
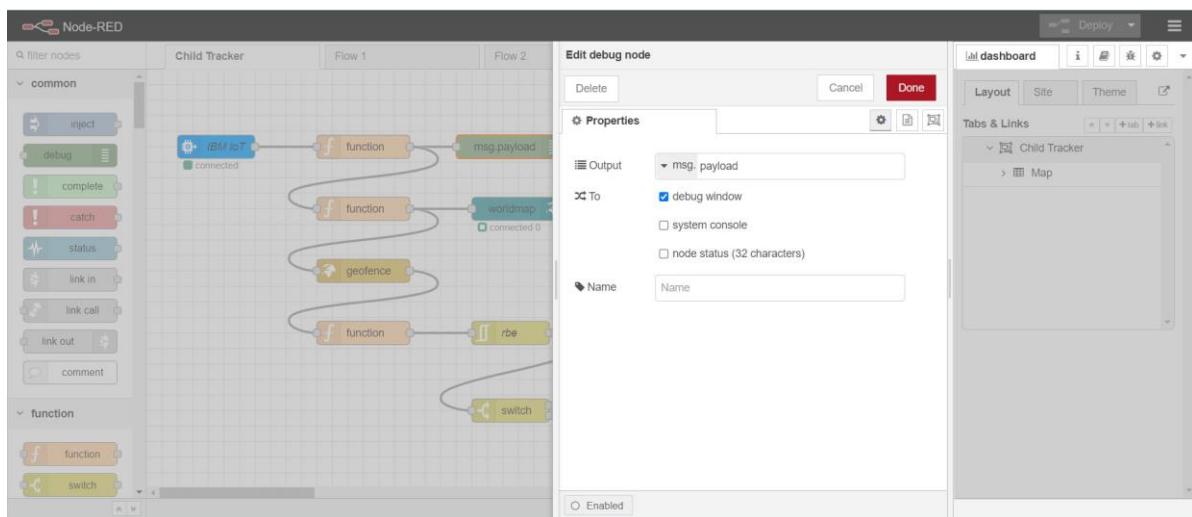
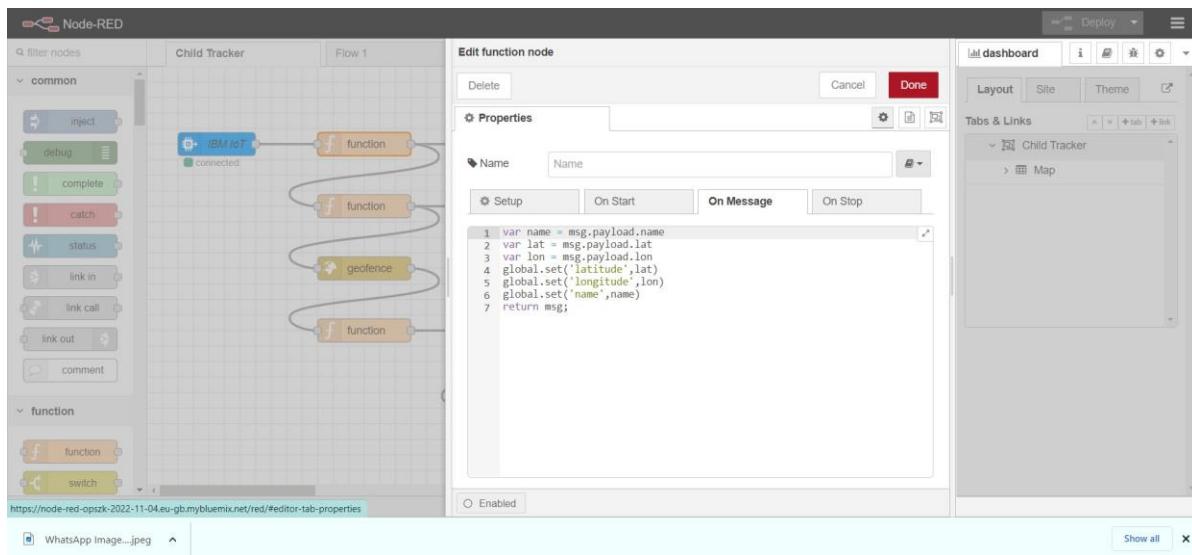
Creating Node-Red service and connecting with IBM cloud

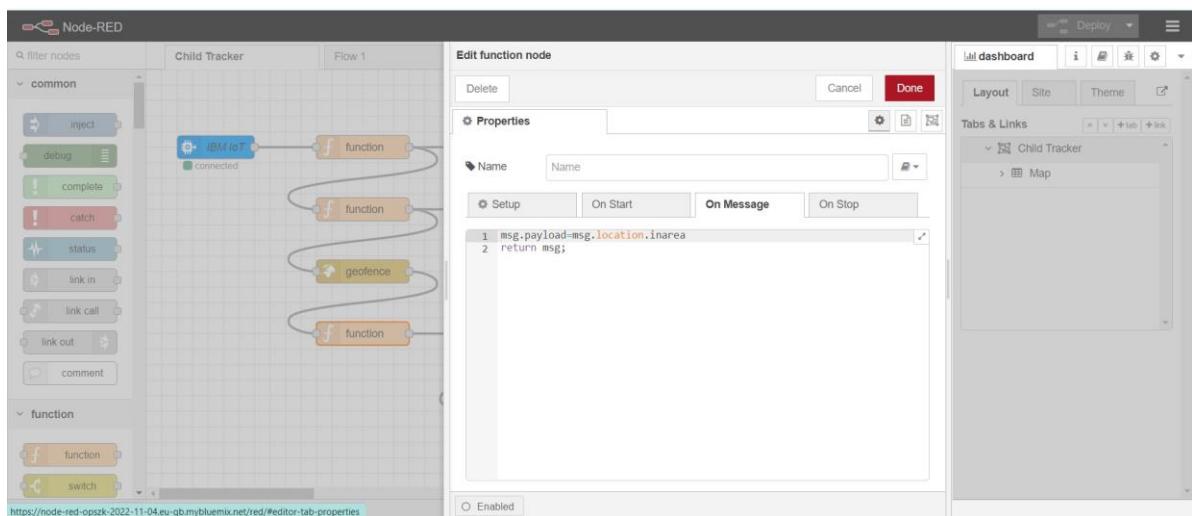
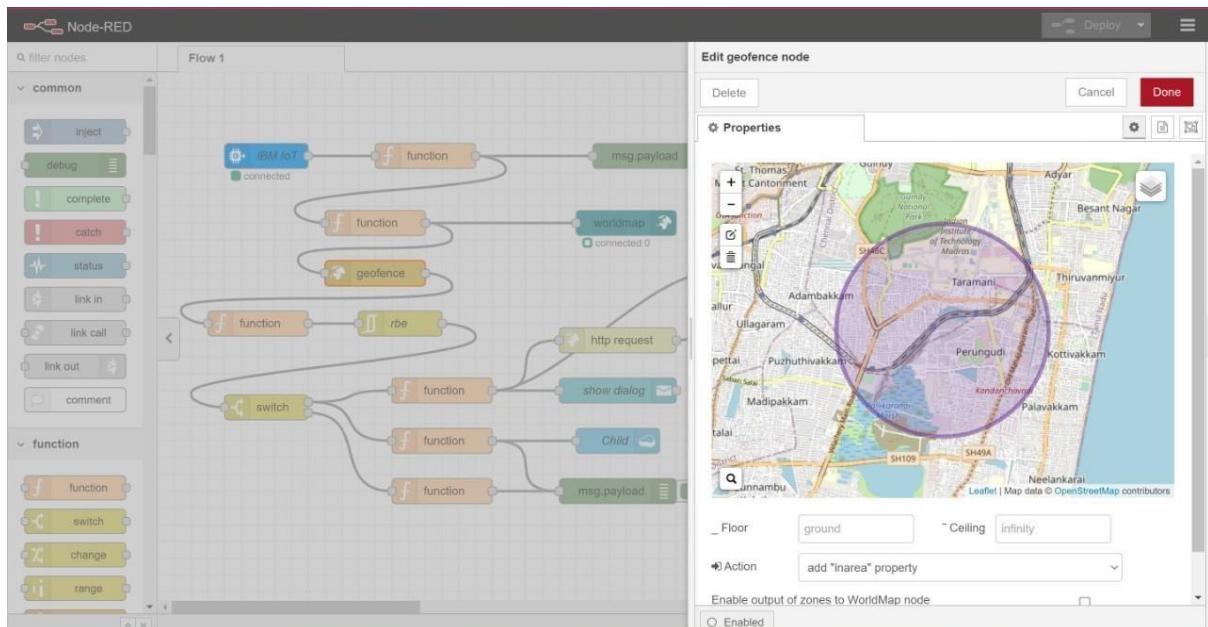
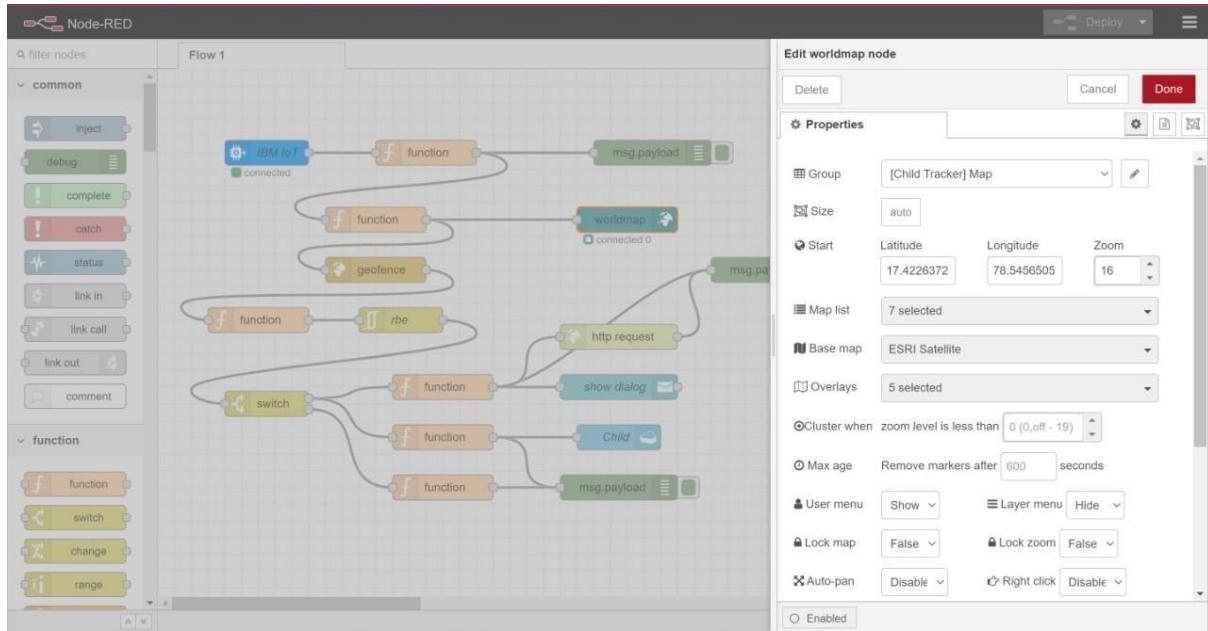
Creating Node-Red service:



Codes in each Node:







The screenshot shows the Node-RED interface with a flow titled "Child Tracker". The flow consists of the following nodes and connections:

- An "inject" node is connected to a "GPI/GPI-T" node.
- The "GPI/GPI-T" node has two outputs: one to a "function" node labeled "f" and another to a "function" node labeled "worldmap".
- The "f" function node has two outputs: one to a "geofence" node and another to an "rbe" node.
- The "worldmap" function node has one output to an "rbe" node.
- The "geofence" node has one output to an "rbe" node.
- The "rbe" node has one output to a "switch" node.
- The "switch" node has two outputs: one to a "comment" node and another to a "complete" node.

On the left sidebar, under the "common" category, the following nodes are listed: inject, debug, complete, catch, status, link in, link call, link out, and comment. Under the "function" category, there are two nodes: function and switch.

A modal window titled "Edit filter node" is open, showing the properties for the "function" node. The properties are as follows:

- Mode:** block unless value changes
- Property:** msg.payload
- Apply mode separately for each:** msg.topic (checkbox checked)
- Name:** rbe

The top right corner of the interface includes a "Deploy" button and a "dashboard" tab.

The screenshot shows a Node-RED flow titled "Child Tracker". The flow consists of the following nodes and connections:

- An "inject" node connects to an "IBM IoT" node.
- The "IBM IoT" node has a "connected" output.
- There are two parallel "function" nodes. The top one has an output "msg.payload" to a "worldmap" node. The bottom one has an output to a "geofence" node.
- The "worldmap" and "geofence" nodes both connect to an "rbe" (Run Block Expression) node.
- The "rbe" node connects to a "switch" node.
- A "comment" node is positioned between the "geofence" and "rbe" nodes.

The "Edit switch node" dialog is open, showing the following settings:

- Properties**:
 - Name: Name
 - Property: msg.payload
 - is false: → 1
 - is true: → 2
- comment**: checking all rules
- recreate message sequences
- Enabled

The dashboard on the right shows tabs for "dashboard", "Layout", "Site", "Theme", and "Settings". The "Child Tracker" tab is selected.

The screenshot shows a Node-RED flow titled 'Child Tracker' with the following structure:

```
graph LR; inject[inject] --> fn1(function); fn1 --> msgPayload[msg.payload]; fn1 --> fn2(function); fn2 --> worldmap[worldmap]; fn2 --> rbe[rbe]; worldmap -- Connected to --> rbe; rbe --> switch[switch]; switch --> fn3(function); fn3 --> dashboard[dashboard];
```

The 'Properties' panel for the second function node is open, showing the 'On Message' tab with the following JavaScript code:

```
var d = new Date();
var utc = d.getTime() + (d.getTimezoneOffset() * 60000);
var offset = 5.5; // This is the offset for UTC+3, in your case (UTC+1)
newDate = new Date(utc + (3600000 * offset));
msg.payload = {
  "date": newDate,
  "time": newDate.toLocaleString(),
  "name": global.get('name'),
  "lat": global.get('latitude'),
  "lon": global.get('longitude')
};
return msg;
```

Node-RED

Child Tracker

Flow 1

The screenshot shows a Node-RED flow titled "Child Tracker" with "Flow 1". It starts with an "inject" node, followed by a "function" node with the following code:

```

1 var d = new Date();
2 var utc = d.getTime() + (d.getTimezoneOffset() * 60000);
3 var offset = 5.5; // This is the offset for UTC+3, in your case (UTC+1)
4 newDate = new Date(utc + (3600000* offset));
5 msg.payload = {
6   "message": "Entry",
7   "Time": newDate.toLocaleString(),
8   "name":global.get('name'),
9   "lat":global.get('latitude'),
10  "lon":global.get('longitude')
11 };
12 return msg;

```

After the function node, the flow continues through a "worldmap" node, a "geofence" node, another "function" node, an "rbe" node, and a "switch" node.

Edit function node

Properties

Name: Name

Setup: On Start

On Message

On Stop

Code:

```

1 var d = new Date();
2 var utc = d.getTime() + (d.getTimezoneOffset() * 60000);
3 var offset = 5.5; // This is the offset for UTC+3, in your case (UTC+1)
4 newDate = new Date(utc + (3600000* offset));
5 msg.payload = {
6   "message": "Entry",
7   "Time": newDate.toLocaleString(),
8   "name":global.get('name'),
9   "lat":global.get('latitude'),
10  "lon":global.get('longitude')
11 };
12 return msg;

```

Enabled:

dashboard

Layout, **Site**, **Theme**

Tabs & Links

Child Tracker

Map

Node-RED

Child Tracker

Flow 1

The screenshot shows a Node-RED flow titled "Child Tracker" with "Flow 1". It starts with an "inject" node, followed by a "function" node with the same code as in the first flow, then a "worldmap" node, a "geofence" node, an "rbe" node, a "switch" node, and a "function" node. The output of the "function" node is connected to an "http request" node, which has the following properties:

Edit http request node

Properties

Method: GET

URL: <https://www.fast2sms.com/dev/bulkV2?authorization>

Payload: Ignore

Enable secure (SSL/TLS) connection:

Use authentication:

Enable connection keep-alive:

Use proxy:

Only send non-2xx responses to Catch node:

Return: a UTF-8 string

Name: Name

Enabled:

dashboard

Layout, **Site**, **Theme**

Tabs & Links

Child Tracker

Map

Node-RED

Child Tracker

Flow 1

Flow 2

The screenshot shows a Node-RED flow titled "Child Tracker" with "Flow 1" and "Flow 2". Both flows follow a similar structure: "inject" node, "function" node (with the same code), "worldmap" node, "geofence" node, "rbe" node, "switch" node, and a final "function" node. The output of the final "function" node in Flow 1 is connected to an "http request" node, while the output of the final "function" node in Flow 2 is connected to a "notification" node.

Edit notification node

Properties

Layout: OK / Cancel Dialog

Send to all browser sessions:

Default action label: OK

Secondary action label: (optional label for Cancel button)

Note: accepting raw HTML/JavaScript input in msg.payload to format popup.

Class: [msg.className]

Topic: [msg.topic]

Name: Show Dialoge

Enabled:

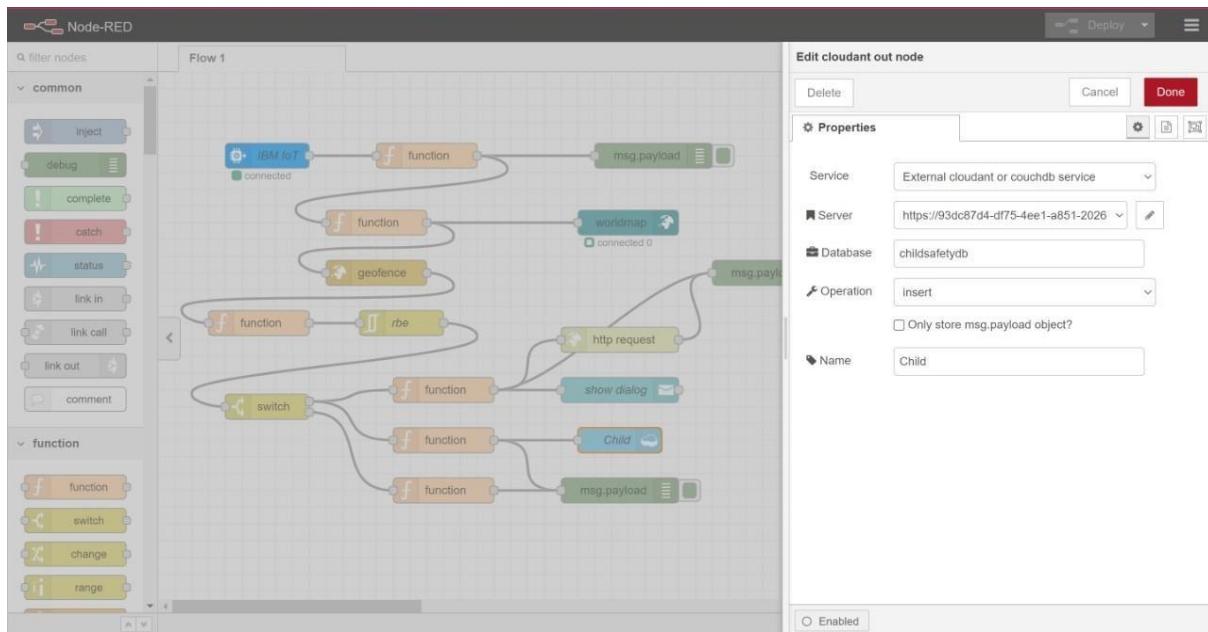
dashboard

Layout, **Site**, **Theme**

Tabs & Links

Child Tracker

Map

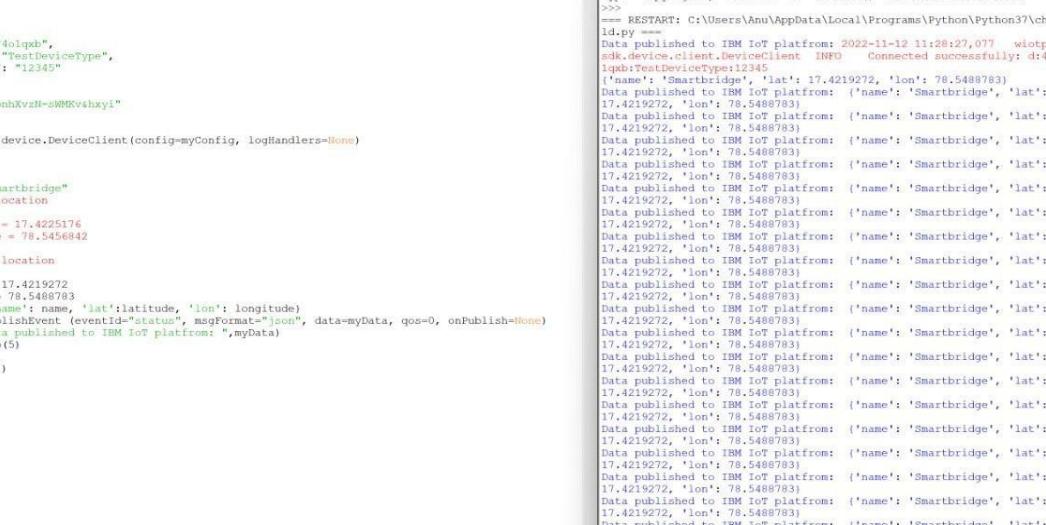


Connecting with IBM Cloud: Using IBM IOT node through the API key

Key	Description	Role	Expires
a-4o1qxb-d5wguvebrf	-	Standard Application	-
a-4o1qxb-ecmygwzdcde	API Key for the device simulator	Standard Application	-

Key	Last Edited By
a-4o1qxb-d5wguvebrf	310819106007@smartinternz.com
Description	-
Date Added	Nov 10, 2022 2:20 PM
Last Update	Nov 10, 2022 2:20 PM

Transferring values from Python Code:



The screenshot shows a terminal window titled "Python 3.7.0 Shell" running on a Windows 10 desktop. The terminal displays a Python script for publishing data to an IBM IoT platform. The script uses the `iot.device` module from the `iotivity` library. It defines a configuration object `myConfig` with fields like `identity`, `auth`, and `deviceID`. It then creates a `DeviceClient` and connects to the IBM IoT platform. A loop publishes event data every 5 seconds. The published data includes a timestamp, location coordinates (lat: 17.4219272, lon: 78.5488783), and a status message. The terminal also shows the script's copyright notice and the Python version information.

```
child.py - C:\Users\Anu\AppData\Local\Programs\Python\Python37\child.py (3.7.0)
File Edit Format Run Options Window Help
import json
import wiotp.sdk.device
import time
myConfig = {
    "identity": {
        "orgId": "401gpb",
        "typeId": "TestDeviceType",
        "deviceId": "12345"
    },
    "auth": {
        "token": "pnhXvzN-sWMKv4hxyl"
    }
}
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
    name = "Smartbridge"
    #in area location
    #latitude = 17.4225176
    #longitude = 78.5456842
    #out area location
    latitudes= 17.4219272
    longitudes= 78.5488783
    myData = {name : {name : 'lat':latitudes, 'lon': longitudes}}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Data published to IBM IoT platform:",myData)
    time.sleep(5)

client.disconnect()
```

Node-Red:

