
Node-RED Hands-on with Open Technologies

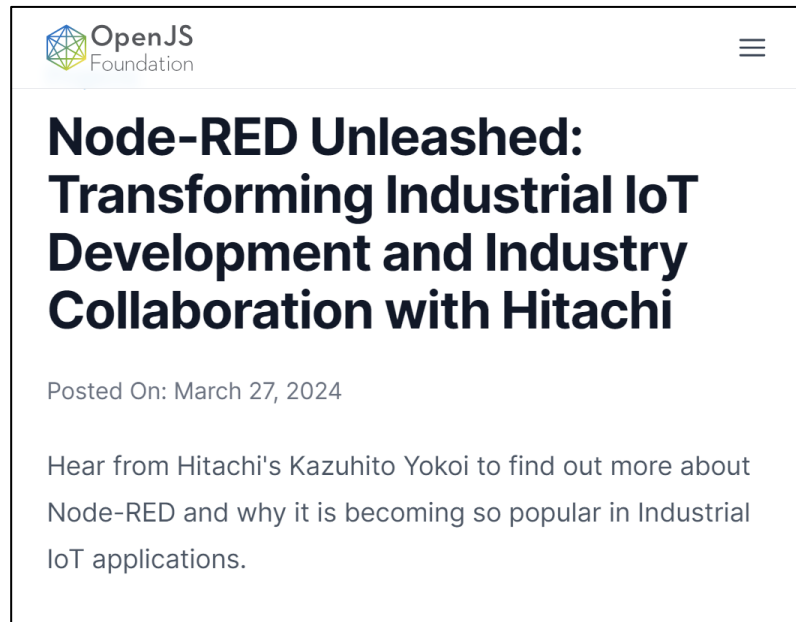
October 22, 2024

Kazuhito Yokoi
Hitachi Academy

Kazuhito Yokoi (横井 一仁)



- Software Engineer, Hitachi Academy
- No.3 contributor in Node-RED project
- Organizer of Node-RED User Group



My interview on
OpenJS Foundation blog

<https://openjsf.org/blog/node-red-unleashed>

Contents

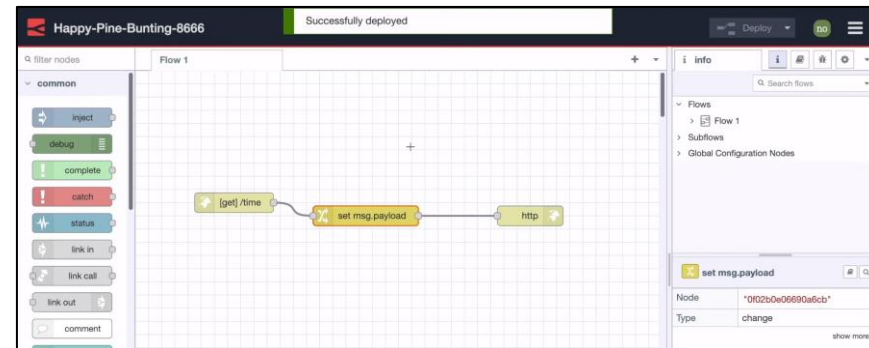
1. What is Node-RED?
2. Preparing Node-RED Environment
3. Basic Operations
4. Data Visualization
5. E-mail Notification
6. ChatGPT with Node-RED
7. Granite Code with Node-RED
8. Conclusion

What is Node-RED?

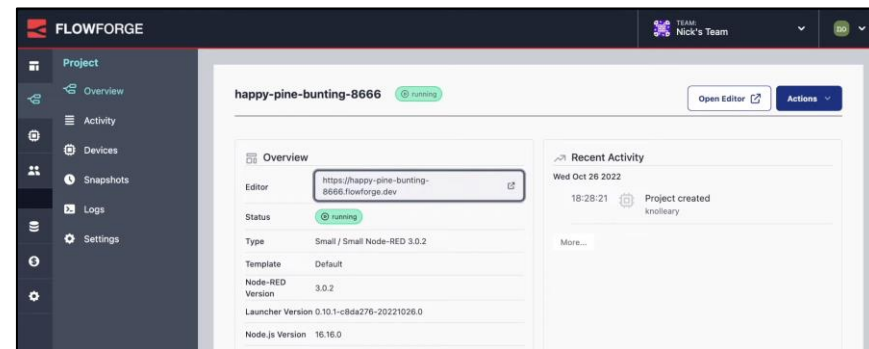
What is Node-RED?

Visual programming tool for IoT and web applications

- Developed by IBM in 2013
- OSS hosted in OpenJS Foundation
- Browser based development environment
- Management OSS available



Node-RED flow editor



Management software

Visual programming environment by connecting functional blocks in the processing order

- No code development environment for beginners
- Extensibility through partial code writing

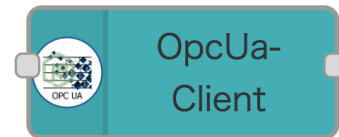
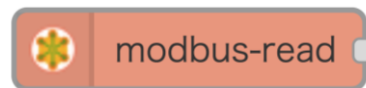
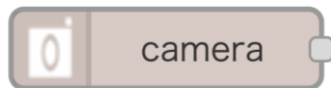
The screenshot displays the Node-RED Flow Editor interface. On the left, a sidebar shows a search bar 'filter nodes' and a 'common' category with nodes like 'inject', 'debug', and 'complete'. The main workspace shows a flow with four nodes: 'mqtt' (purple), 'set msg.payload' (yellow), 'function' (orange), and 'sendgrid' (blue). A callout box labeled 'Development by connecting blocks' points to the connections between the 'mqtt' and 'set msg.payload' nodes, and between the 'function' and 'sendgrid' nodes. Another callout box labeled 'Partial coding to extend flow' points to the 'function' node, which contains a JavaScript code snippet for processing MQTT messages.

```
1 const values = context.get("values") || []; // 直近の値を保存する配列を取得
2 const threshold = 0.5; // 50%の閾値
3
4 values.push(msg.payload); // 受け取った値を配列に追加
5
6 if (values.length > 7) {
7   values.shift(); // 配列の先頭から要素を削除して7つに保つ
8 }
9
10 const average = values.reduce((total, value) => total + value, 0) / values.length;
11
12 if (msg.payload > average * (1 + threshold)) {
```

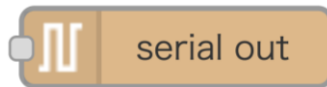
Partial coding to extend flow

Nodes are functional blocks that connect to services and process data

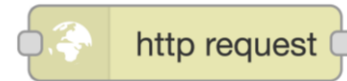
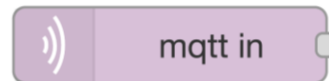
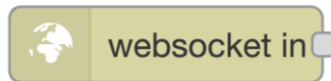
(1) Data collection



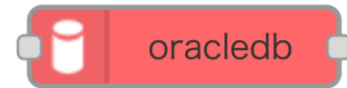
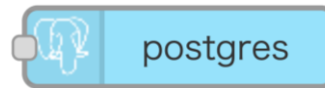
(2) Device control



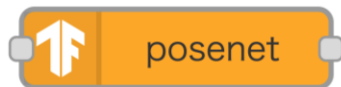
(3) Connection with external systems



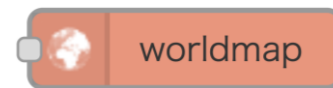
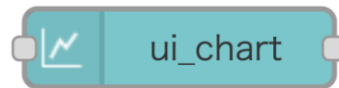
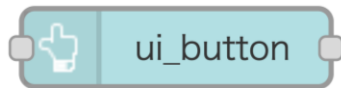
(4) Databases



(5) Data analysis

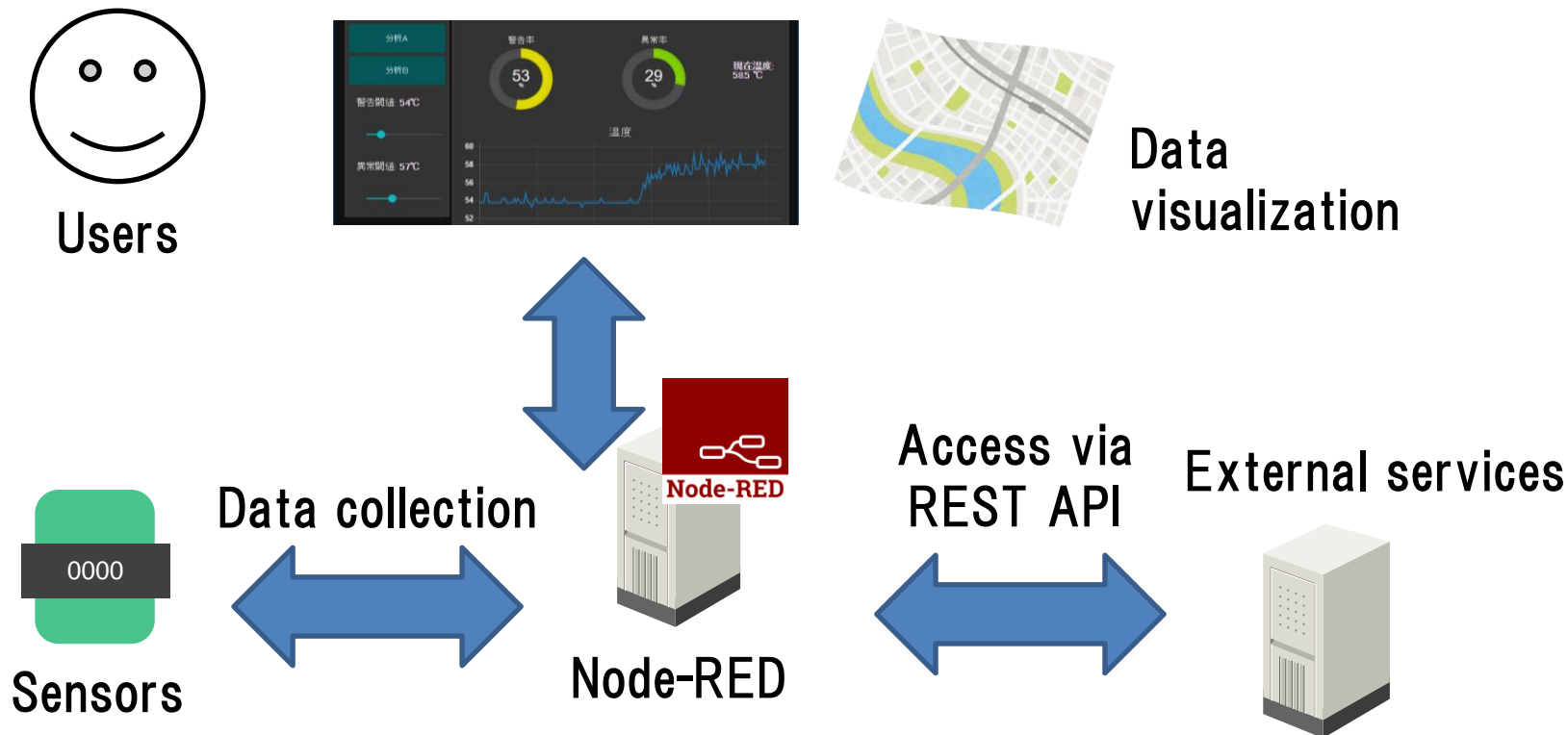


(6) Data visualization



Node-RED as the Hub Tool to Integrate Services

Node-RED is suitable for building applications by integrating many components.



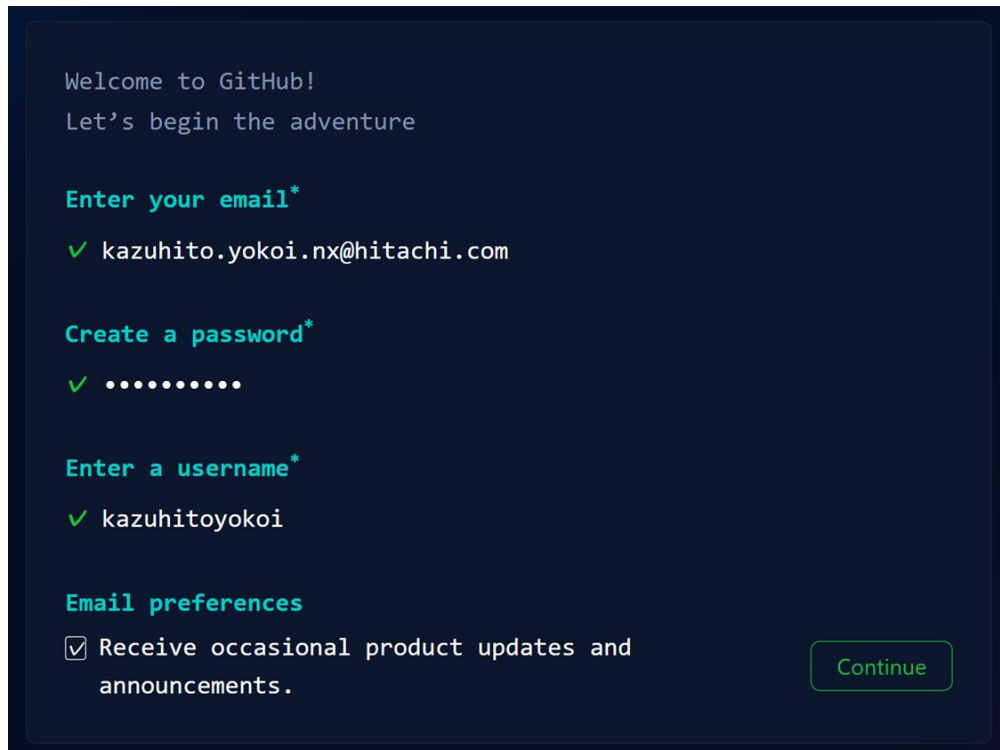
Preparing Node-RED Environment

If you have no GitHub account, create your GitHub account from the following URL.

<https://github.com/signup>

To register for the account, you need to input your information.

- e-mail address
- Password
- Username



Welcome to GitHub!
Let's begin the adventure

Enter your email*

✓ kazuhito.yokoi.nx@hitachi.com

Create a password*

✓

Enter a username*

✓ kazuhitoyokoi

Email preferences

☒ Receive occasional product updates and announcements.

Continue

Registration for creating GitHub account

Because the default timeout of 30 minutes is too short, increase the timeout to the maximum on the settings page.

(1) Access the Codespaces settings from the following URL.

<https://github.com/settings/codespaces>

(2) Set 240 minutes (4 hours)

(3) Click the "Save" button

Default idle timeout

A codespace will suspend after a period of inactivity. You can change the default idle timeout for all codespaces created after the default is changed. You can also change the timeout for existing codespaces, even if it is idle. The maximum value is **240 minutes** (4 hours).

240 minutes

Save

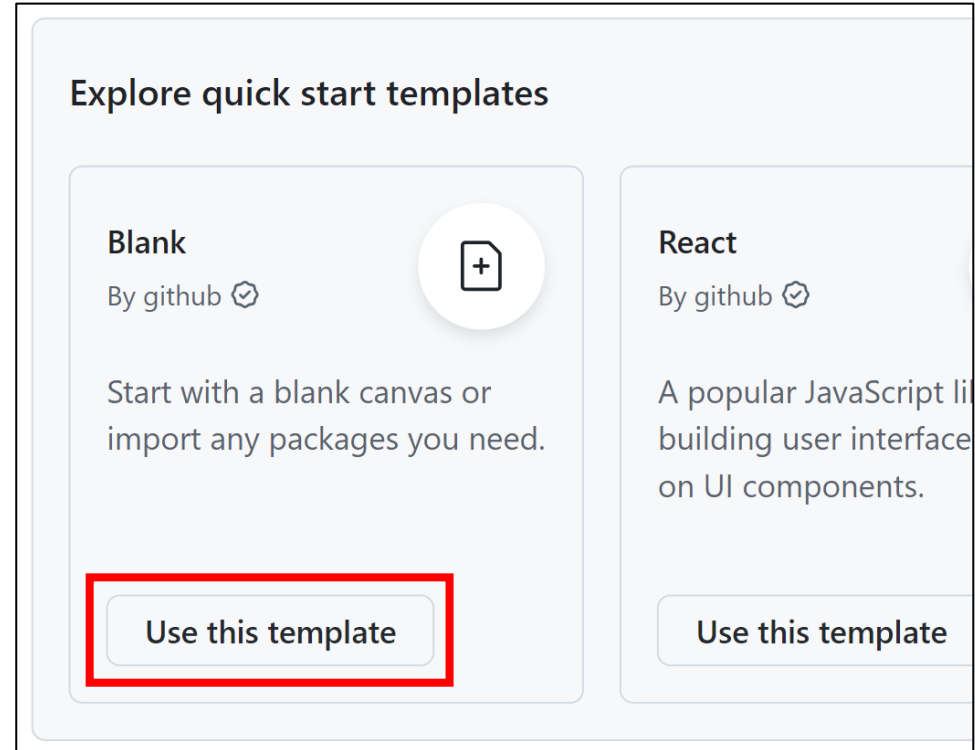
Creating the empty environment on the GitHub Codespaces

(1) Go to the Codespaces page

<https://github.com/codespaces>

(2) Click the "Use this template" button in the Blank item


-> Visual Studio Code-like editor will be opened.



After opening the code editor, you can use Terminal in the bottom right corner.

(1) Install Node-RED with the npm command

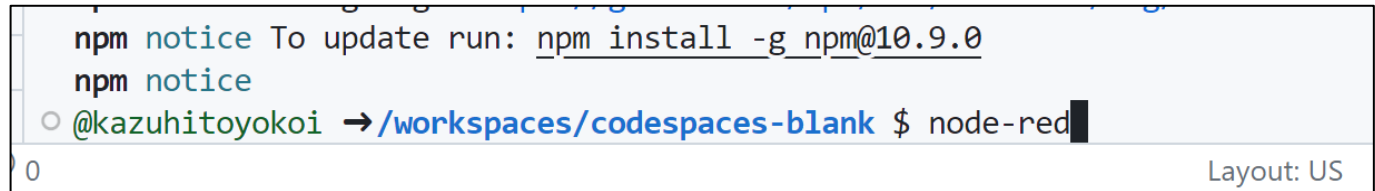
```
npm install -g node-red
```



A screenshot of a terminal window with tabs for PROBLEMS, OUTPUT, and TERMINAL. The TERMINAL tab is active, showing a bash prompt. The command `npm install -g node-red` has been entered and is being executed. The prompt shows the user is `@kazuhiyokoi` and the current directory is `/workspaces/codespaces-blank`.

(2) Start Node-RED

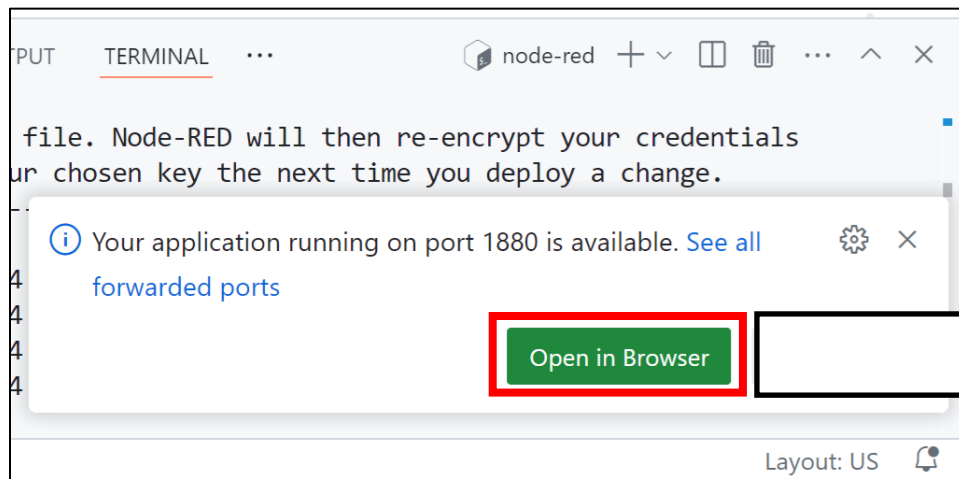
```
node-red
```



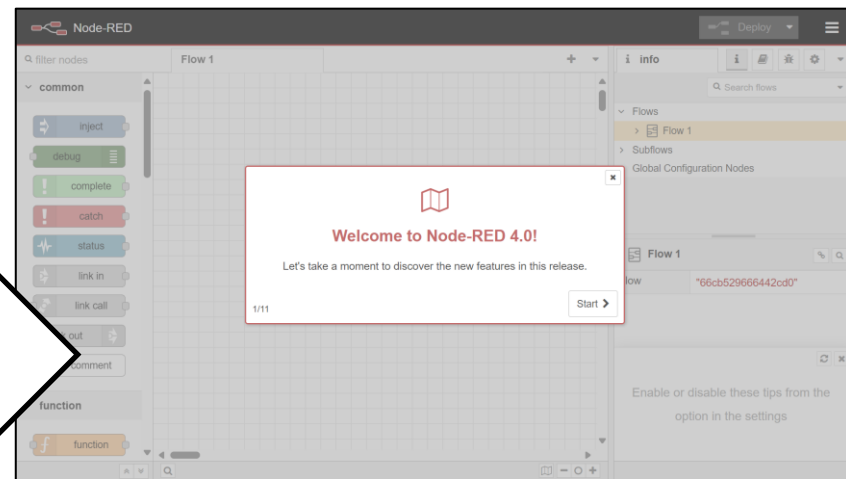
A screenshot of a terminal window showing the output of the `npm install -g node-red` command. It displays two lines of npm notices: `npm notice To update run: npm install -g npm@10.9.0` and `npm notice`. Below these, the `node-red` command has been entered and is being executed. The prompt shows the user is `@kazuhiyokoi` and the current directory is `/workspaces/codespaces-blank`. The bottom right corner of the terminal window shows "Layout: US".

[Hands-on] Opening Node-RED Flow Editor

- After starting Node-RED, the pop-up dialog will appear to open flow editor.
- Click the green "Open in Browser" button
-> You can see the Node-RED flow editor in new tab.



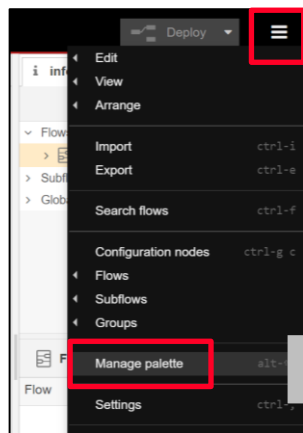
Notification dialog



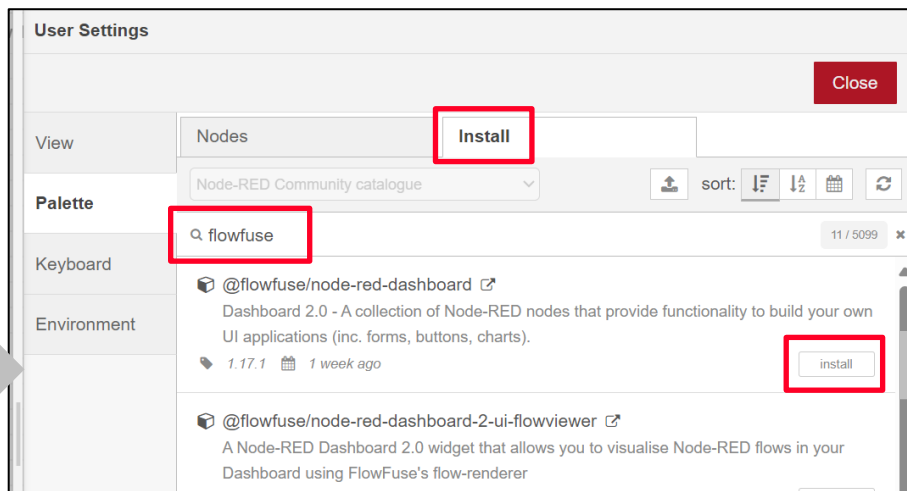
Node-RED flow editor

[Hands-on] Required Nodes Installation

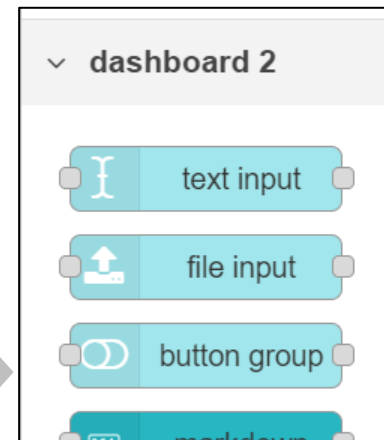
- (1) Click the Hemberger icon on the top right to open the menu
- (2) Select "Manage palette" in the menu to open the user settings
- (3) Select the "Install" tab in the Palette tab
- (4) Type "flowfuse" in the search box
- (5) Click the "install" button in the "@flowfuse/node-red-dashboard" to install nodes



Select
"Manage palette"



Search and add node

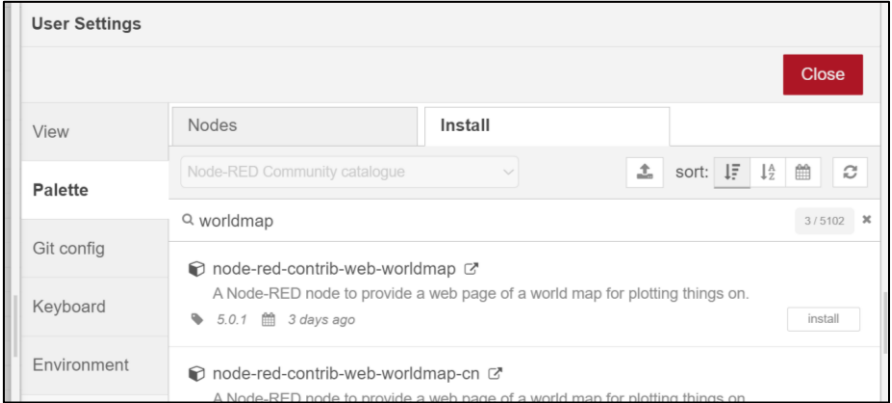
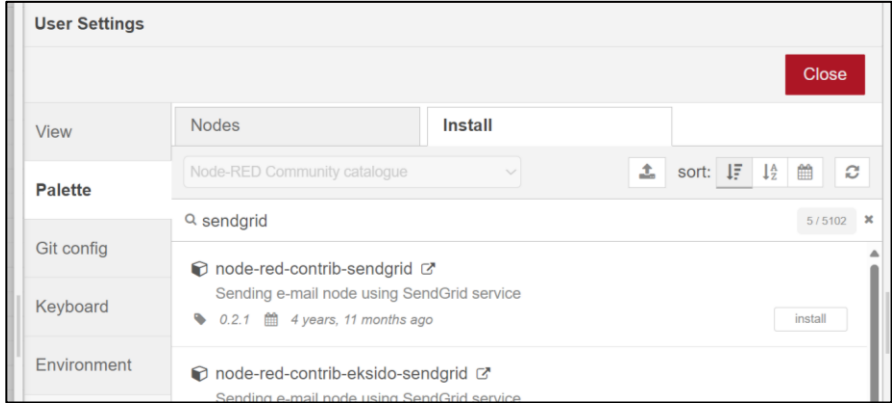


Palette after
installation

[Hands-on] Other Required Nodes Installation

Install two more red nodes in the following table

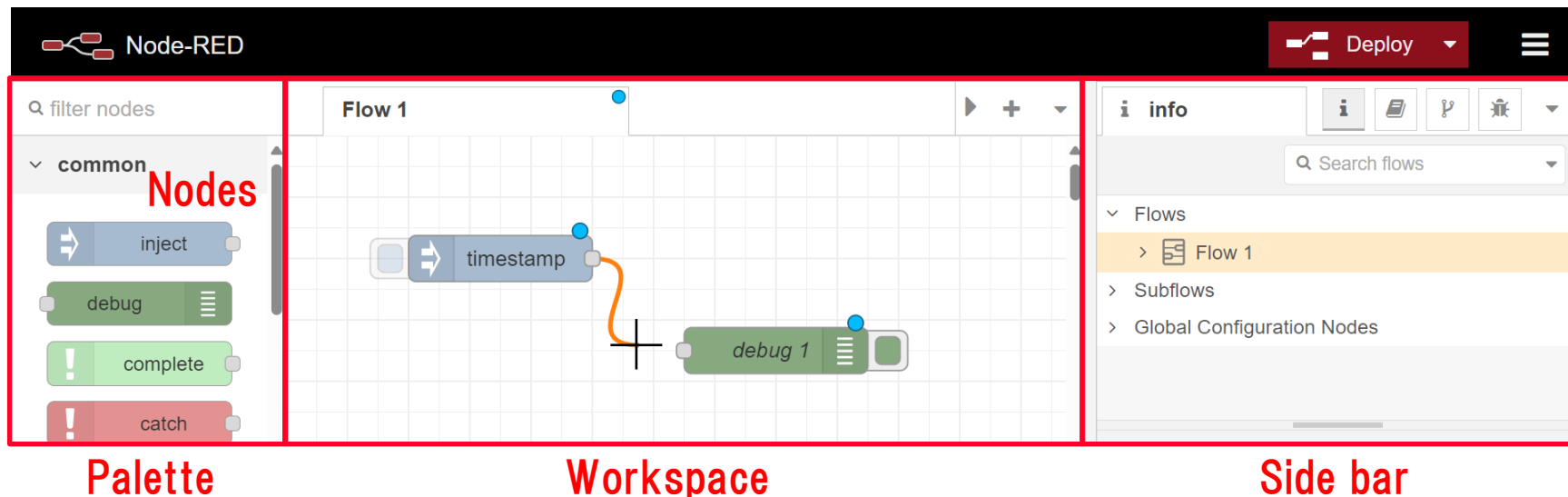
#	Search keyword	Module name
1	flowfuse	@flowfuse/node-red-dashboard
2	sendgrid	node-red-contrib-sendgrid
3	worldmap	node-red-contrib-web-worldmap



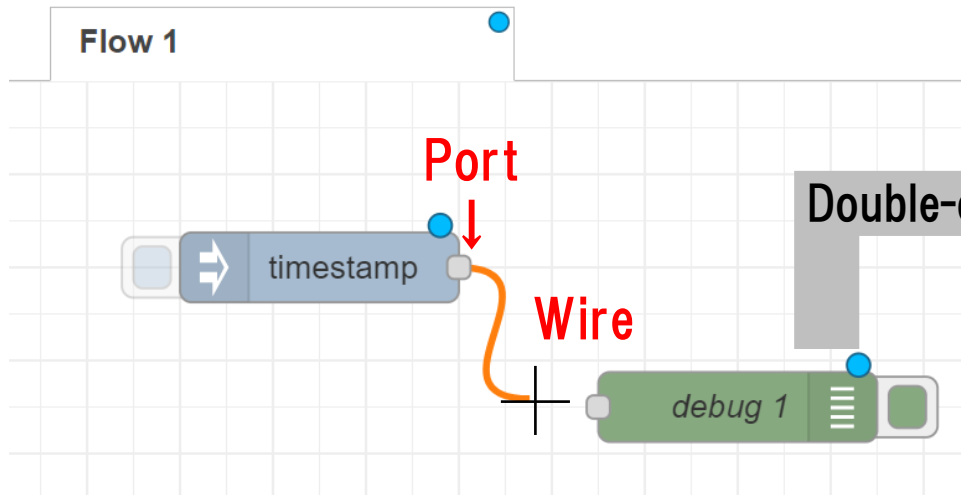
Basic Operations

Typical Development on Node-RED Flow Editor

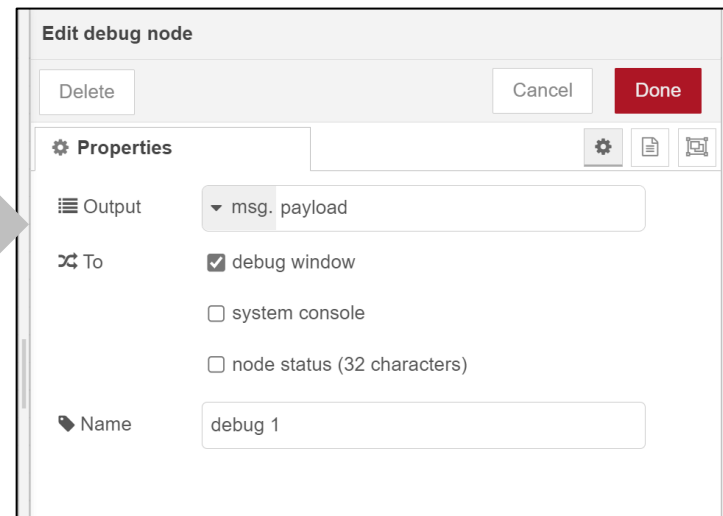
- (1) Select the node which you want to use from the left palette
- (2) Drag and drop the node to the workspace
- (3) Connect nodes with wire in the processing order
- (4) Double-click the node to configure setting
- (5) Hit the "Deploy" button to execute the flow



- To connect nodes, wire the output port to the input port of another node.
- To configure node properties, double-click the node to open the settings UI (after entering the properties, click the “Done” button to apply them).



Conncting nodes by wire



Node property settings UI

The area that displays the information about the flow, nodes, and messages

- Information tab: Tab that provides an overview of nodes in the workspace
- Help tab: Tab for node details about how to use the selected node
- Debug tab: Tab for viewing the data output from the debug nodes

Switch the tabs by clicking the button

Information tab

info info help debug

Search flows

Flows

Flow 1

debug 1

timestamp

Subflows

Help tab

help info help debug

Search help

inject

inject

Injects a message into a flow either manually

Debug tab

debug info help debug

all nodes all

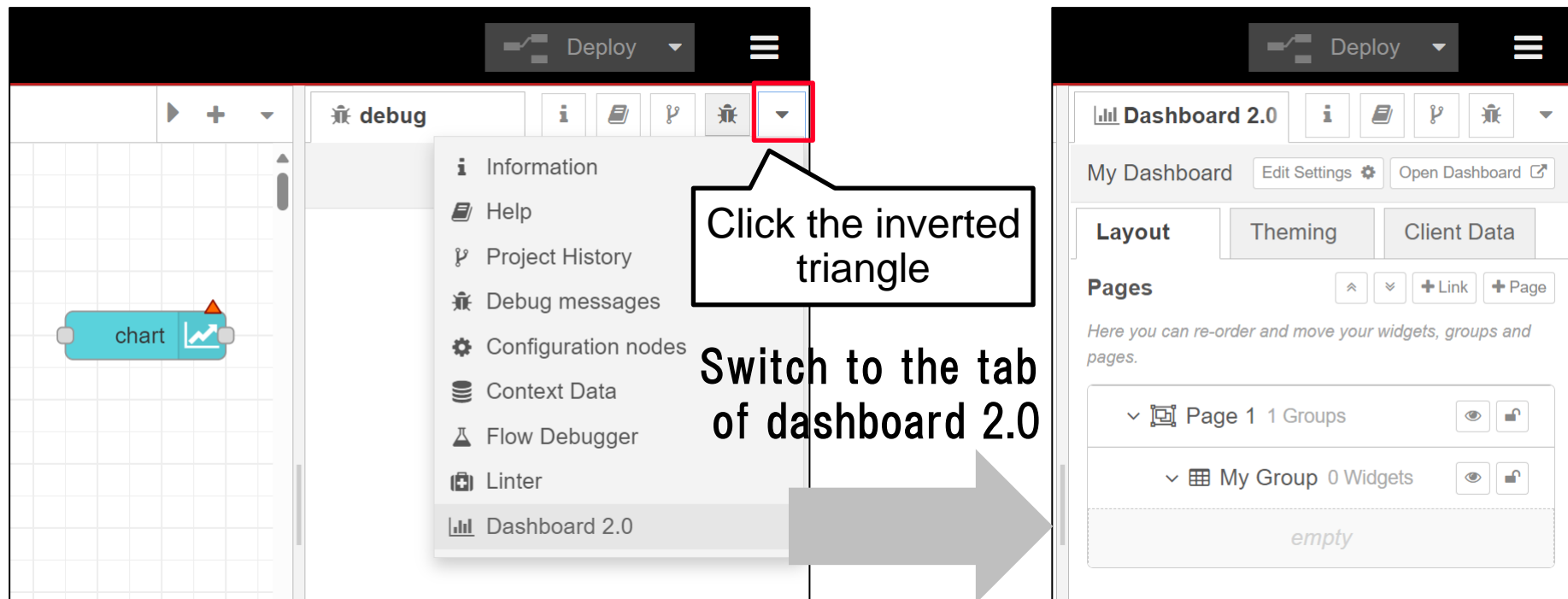
2024/10/8 12:30:05 node: debug 1

msg.payload : number

1728358205676

Sidebar (2/2)

When there is no button on the sidebar, you can display the menu by clicking the inverted triangle button in the right corner.



Create the flow to output the current timestamp by the following steps

- (1) Drag and drop the inject node in the common category to the workspace
- (2) Place the debug node to the right side of the inject node
- (3) Connect the inject node and the debug node by wiring between the ports
- (4) Hit the "Deploy" button to execute the flow

The screenshot shows the Node-RED web interface. On the left, the 'common' category is expanded in the node palette, showing 'inject', 'debug', 'complete', and 'catch' nodes. In the center workspace, a flow named 'Flow 1' is being built. It consists of an 'inject' node, a 'timestamp' node, and a 'debug 1' node. Red arrows and text annotations indicate the steps: (1) Drag & drop points to the 'inject' node being moved from the palette; (2) Drag & drop points to the 'debug 1' node being moved to the workspace; (3) Wire the ports points to the connection line between the 'timestamp' and 'debug 1' nodes. On the right, the 'info' sidebar shows the flow structure, and a red callout box points to the 'Deploy' button in the top right corner with the text '(4) Hit the Deploy button'.

[Hands-on] Creating the First Flow (2/3)

Execute the developed flow by the following steps

- (1) Click the bug button in the sidebar to open the debug tab
- (2) Click the button of the inject node to execute the flow
- (3) The number representing the current time will be output to the debug tab

The screenshot shows the Node-RED web interface. On the left sidebar, the 'common' tab is active, showing nodes like 'inject', 'debug', 'complete', and 'catch'. The main workspace contains a flow titled 'Flow 1' with an 'inject' node connected to a 'timestamp' node, which is then connected to a 'debug 1' node. A diagram above the flow shows a message object with a 'time stamp' and a 'payload'. A callout box (1) points to the 'bug' icon in the top right of the sidebar. Another callout box (2) points to the 'inject' node's button. A third callout box (3) points to the output in the debug console, which shows the message: '2024/10/8 12:47:50 node: debug 1' and 'msg.payload : number' followed by the value '1728359270625'.

Node-RED

Flow 1

inject

debug

complete

catch

timestamp

debug 1

msg

time stamp

payload

Passing data

(1) Click the bug button

(2) Click the button

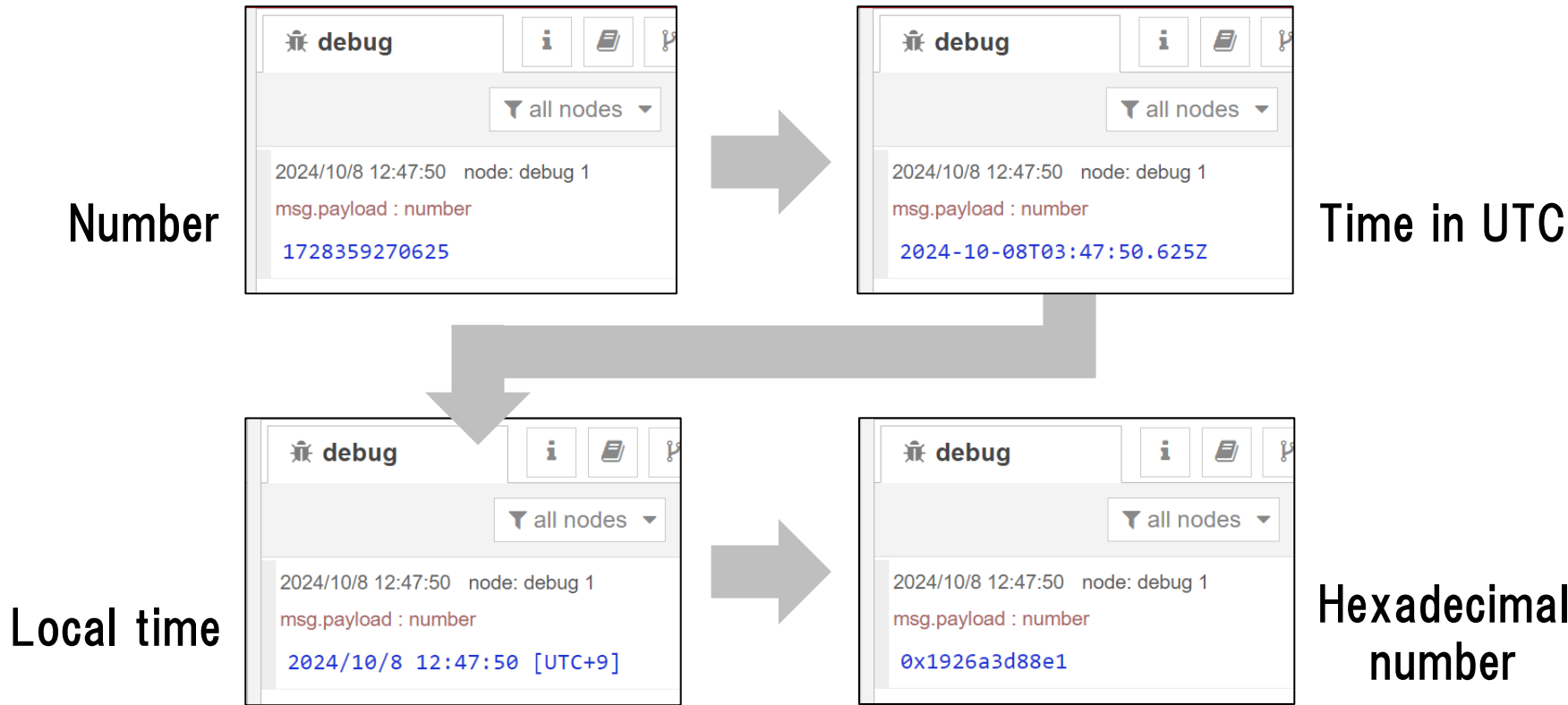
(3) Outputted number

2024/10/8 12:47:50 node: debug 1

msg.payload : number

1728359270625

The outputted data in the debug tab can be switch to another format by clicking



[Hands-on] Creating Flow to Output Hello World (1/2)

To develop the flow to output "Hello world", change the output from the current time to the string, "Hello world", in the node property setting of the inject node.

- (1) Double-click the inject node to open the property settings UI
- (2) Change the data type from "milliseconds since epoch" to "string"
- (3) Type "Hello world" in the text box next to the "az" icon
- (4) Click the "Done" button to apply the settings

The diagram illustrates the steps to configure an inject node in a flow. On the left, a flow is shown with a 'timestamp' node (blue) and a 'debug 1' node (green). A large grey arrow points from the 'timestamp' node to the 'Edit inject node' dialog on the right. The dialog has a 'Delete' button, 'Cancel', and 'Done' buttons. The 'Properties' section shows 'Name' and 'msg. payload' fields. The 'msg. payload' field is set to 'az Hello world'. The 'az' icon is highlighted, and a callout points to it with the text '(2) Select "az string"'. Another callout points to the 'Hello world' text with the text '(3) Type "Hello world"'. A final callout points to the 'Done' button with the text '(4) Click "Done"'. A callout points to the 'timestamp' node with the text '(1) Double-click'.

Executing the developed flow and checking the outputted "Hello world" text

(1) Hit the Deploy button to apply the flow

(2) Click the button of the inject node to execute flow

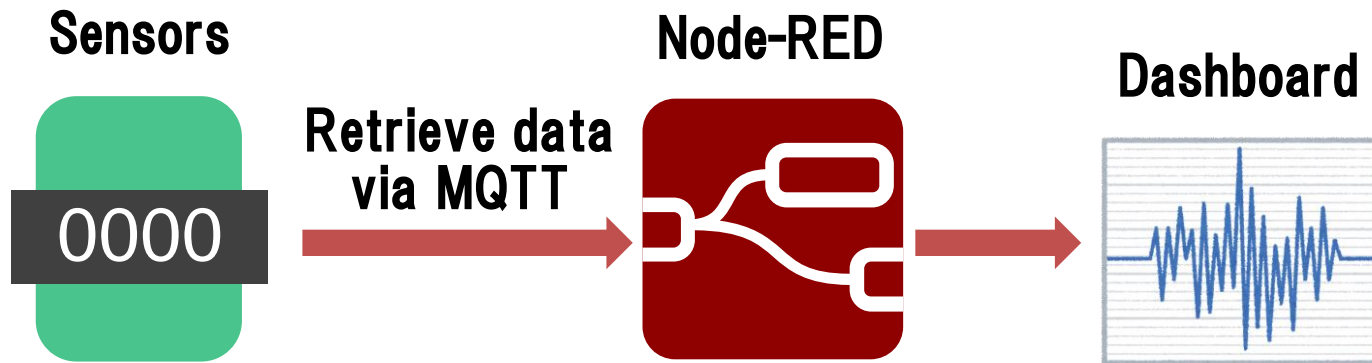
(3) The "Hello world" text will be shown in the debug tab

(1) Hit the
Deploy button

The screenshot shows the Node-RED web interface. On the left, the 'common' node palette contains an 'inject' node, a 'debug' node, and a 'complete' node. The main workspace shows a flow named 'Flow 1' with an 'inject' node connected to a 'Hello world' node, which is then connected to a 'debug 1' node. A callout box labeled '(2) Click the button' points to the 'inject' node. Above the flow, a diagram shows a box labeled 'msg' with a 'payload' field containing 'Hello world', with an arrow labeled 'Passing data' pointing to the 'Hello world' node. On the right, the 'debug' console shows the output: '2024/10/8 13:01:35 node: debug 1' and 'msg.payload : string[11]' followed by '"Hello world"'. A callout box labeled '(3) Outputted data' points to the '"Hello world"' text. At the top right, a 'Deploy' button is visible, with a callout box labeled '(1) Hit the Deploy button' pointing to it.

Data Visualization

Creating a flow to retrieve sensor data via MQTT protocol and then visualize the data in the time series graph on the dashboard



Node-RED has the advantage of natively supporting MQTT, the lightweight data transfer protocol invented by IBM.

Placing the mqtt-in node to retrieve sensor data from the MQTT broker

- (1) Drag and drop the mqtt-in node from the network category of palette to workspace
- (2) Double-click the mqtt-in to open the node property UI
- (3) Click the “+” button next to “Add new mqtt-broker...” pull-down menu

The image shows two parts of the Node-RED interface. On the left, the 'network' category in the node palette is expanded, showing 'mqtt in', 'mqtt out', and 'http in' nodes. A red arrow points from the 'mqtt in' node in the palette to a 'mqtt' node already placed in the workspace. A callout box labeled '(1) Drag & drop' points to this arrow. On the right, the 'Edit mqtt in node' dialog is open. It has tabs for 'Properties', 'Server', 'Action', and 'Topic'. The 'Properties' tab is active, showing a dropdown menu 'Add new mqtt-broker...' with a '+' button next to it. A callout box labeled '(2) Double-click' points to the 'mqtt' node in the workspace. Another callout box labeled '(3) クリック' (Click) points to the '+' button next to the 'Add new mqtt-broker...' dropdown.

[Hands-on] Configuring mqtt-in Node Property

- (1) Input the following URL of the public MQTT broker
"mqtt://public:public@public.cloud.shiftr.io"
- (2) Click the "Add" button to add the configuration
- (3) Type "nodered" in topic field to specify the data to retrieve from the broker
- (4) Click the "Done" button to apply for the settings

The image displays two sequential screenshots of the 'Edit mqtt in node' configuration window, illustrating the steps to add and configure an MQTT broker.

Left Screenshot: The window title is 'Edit mqtt in node > Add new mqtt-broker config node'. It features a 'Properties' section with a 'Name' field. Below this is a 'Connection' tab with a 'Server' field containing the URL 'mqtt://public:public@public.cloud.shiftr.io' and a 'Port' field with the value '1883'. A red 'Add' button is visible in the top right corner. A callout box labeled '(1) Input the URL of the MQTT broker' points to the 'Server' field. Another callout box labeled '(2) Click "Add"' points to the 'Add' button.

Right Screenshot: The window title is 'Edit mqtt in node'. It shows the 'Properties' section with a 'Topic' field containing the value 'nodered'. A red 'Done' button is in the top right corner. A callout box labeled '(3) Enter "nodered"' points to the 'Topic' field. Another callout box labeled '(4) Click "Done"' points to the 'Done' button.

A large grey arrow points from the 'Add' button in the left screenshot to the 'Topic' field in the right screenshot, indicating the flow of the configuration process.

[Hands-on] Placing debug Node to Check the Retrieved Data

- (1) Add the debug node next to mqtt-in node
- (2) Connect the mqtt-in node and debug node by wire
- (3) Hit the "Deploy" button to apply and execute the flow
- (4) Click the bug icon to show the debug tab where the sensor data is observed

The screenshot shows the Node-RED web interface. On the left, the 'common' node palette contains 'inject', 'debug', 'complete', 'catch', and 'status' nodes. A red curved arrow points from the 'debug' node in the palette to a 'debug 1' node on the canvas. The canvas also features a 'nodered' node (with a 'connected' status indicator) and a 'mqtt-in' node. A straight wire connects the output of the 'mqtt-in' node to the input of the 'debug 1' node. On the right, the 'Deploy' button is visible in the top bar. Below it, a 'bug' icon (a small icon with a bug) is highlighted by a callout box. The bottom right of the interface shows 'Flow 1'.

(1) Drag & drop

(2) Connect by wire

(3) Hit the button

(4) Click the bug button

[Hands-on] Checking the Retrieved Sensor Data

- After successfully connecting to the MQTT broker, the node status under the node will have "connected" message along with green rectangle.
- The retrieved data will be outputted on the debug tab every second.

The screenshot shows the Node-RED web interface. On the left, the 'common' node palette contains 'inject', 'debug', 'complete', and a red alert node. The main workspace shows a flow named 'Flow 1' with three nodes: 'inject', 'nodered', and 'debug 1'. The 'nodered' node has a green 'connected' status indicator. A callout box points to this status with the text: 'The status changes to "connected"'. The 'debug 1' node is connected to 'nodered'. On the right, the 'debug' console shows the output of the 'nodered' node, displaying a JSON object:

```
nodered : msg.payload : Object
{
  movement: object,
  acceleration: object
}
```

 A callout box points to this output with the text: 'Retrieved data'. The console also shows timestamps and the node name 'node: debug 1'.

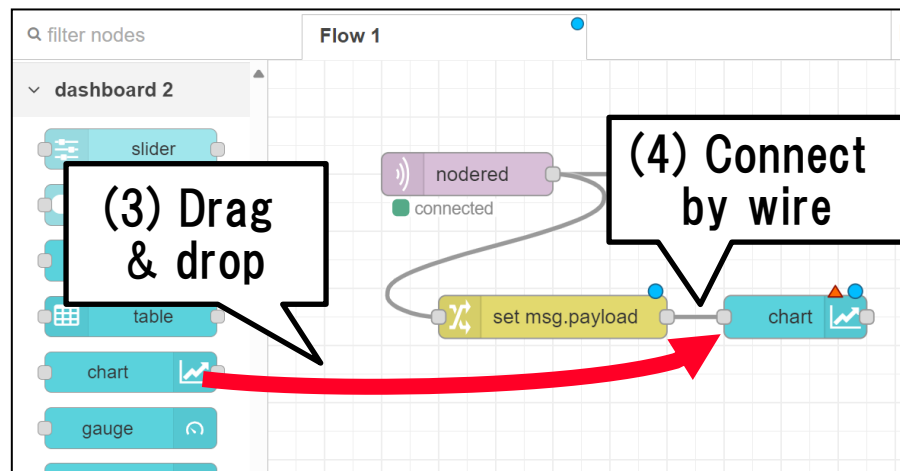
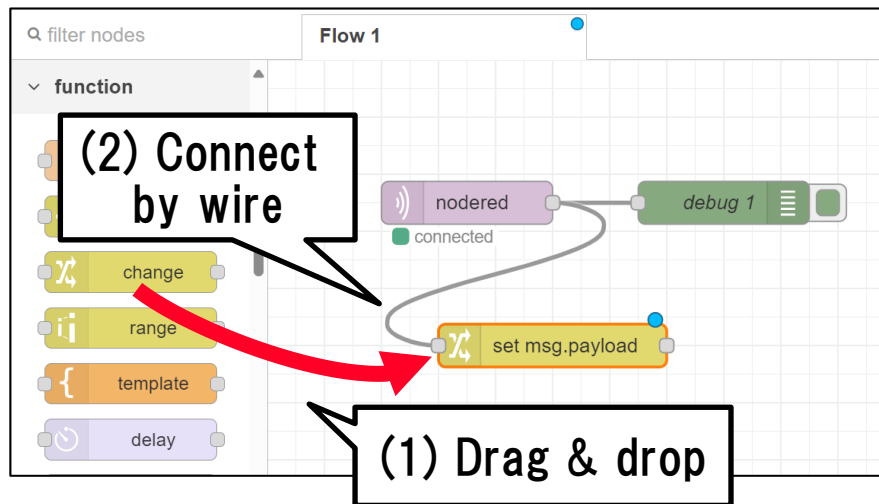
[Hands-on] How to Stop and Clear Output of debug Node

- If you want to stop outputting messages from the debug node, click the button of the debug node.
- To delete all of messages from the debug tab, click the "all" button with the trash can icon.

The screenshot shows the Node-RED web interface. On the left, the 'common' node palette contains 'inject', 'debug', 'complete', 'catch', and 'status' nodes. The central workspace shows a flow named 'Flow 1' with a 'nodered' node (labeled 'connected') connected to a 'debug 1' node. A callout box points to the 'debug 1' node with the text 'Click the button to stop outputting messages'. On the right, the 'debug' tab is active, showing a message log. A callout box points to the 'all' button (with a trash can icon) in the debug tab with the text 'Click "all" to clear messages'. The message log displays a JSON object: `{ movement: object, acceleration: object }` with a timestamp of 2024/10/8 13:48:56 and node: debug 1.

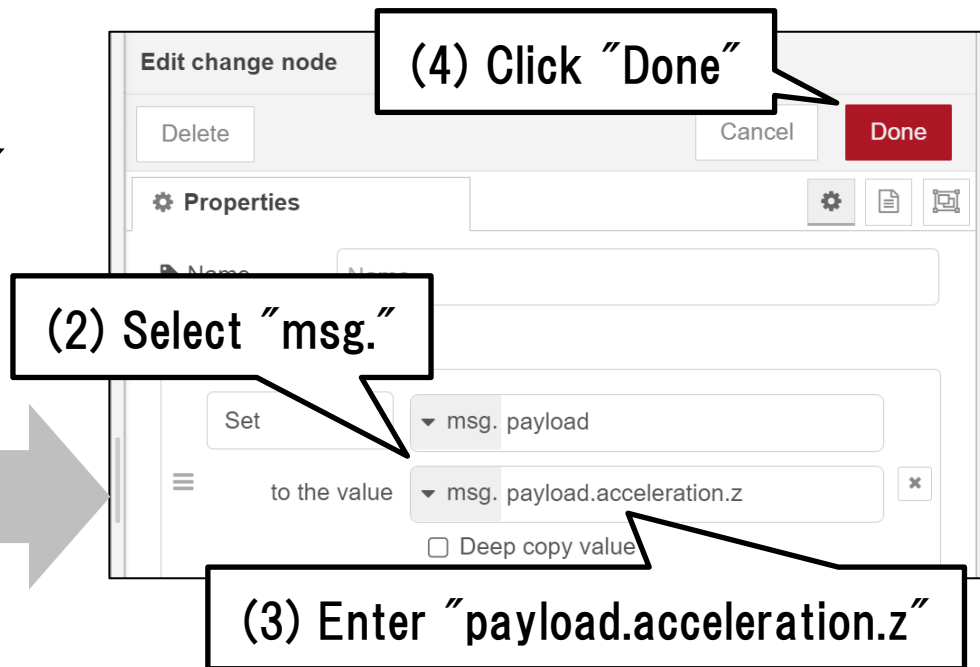
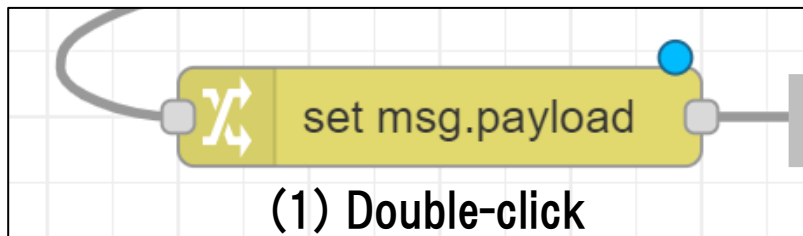
To show sensor data on a time series graph, use the change and chart nodes

- (1) Drag and drop the change node of the function category under the mqtt-in node (the name will be change to "set msg.payload")
- (2) Connect the mqtt-in node to the input port of change node by wire
- (3) Place the chart node of the "dashboard 2" category after the change node
- (4) Wire the output port of the change node to the chart node by wire



Edit the change node settings to extract specific value from the retrieved data

- (1) Double-click the change node to open the node property settings UI
- (2) Select "msg." from the data type menu of the "to the value" field
- (3) In the text box, enter the "payload.acceleration.z" to extract the value
- (4) Click the "Done" button

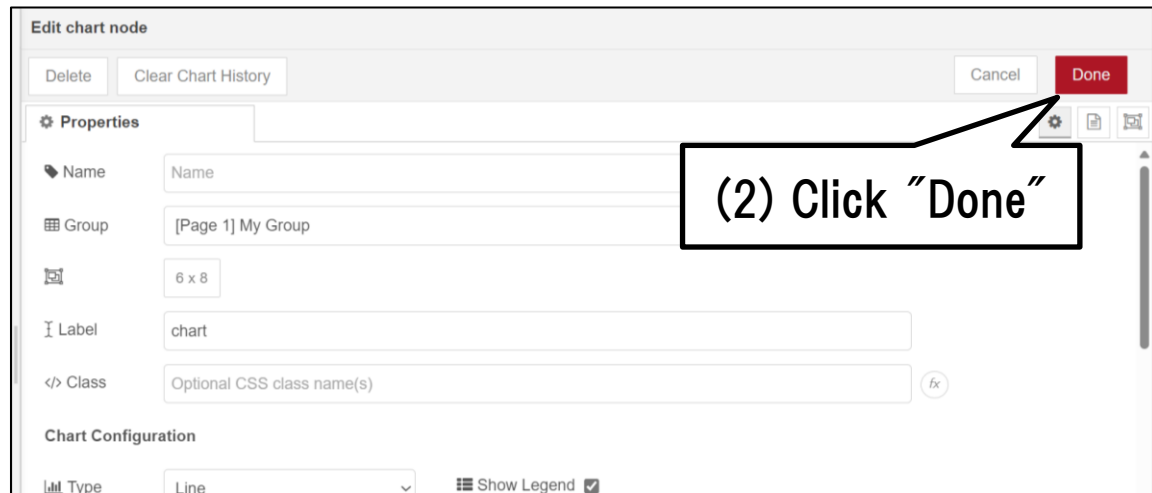
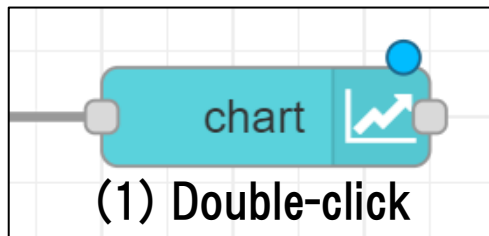


To apply the default dashboard setting, just open and close the node property UI

(1) Double-click the chart node to open the node property UI

(2) Click the "Done" button to close the node property UI

(The settings will be configured automatically without any changes)



(2) Click "Done"

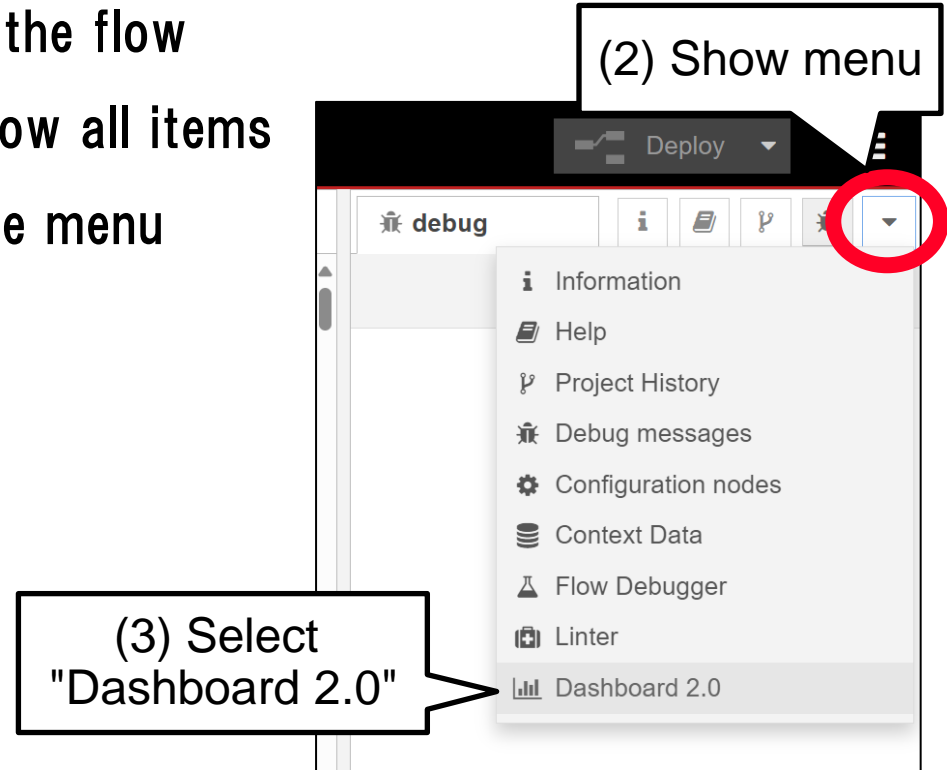
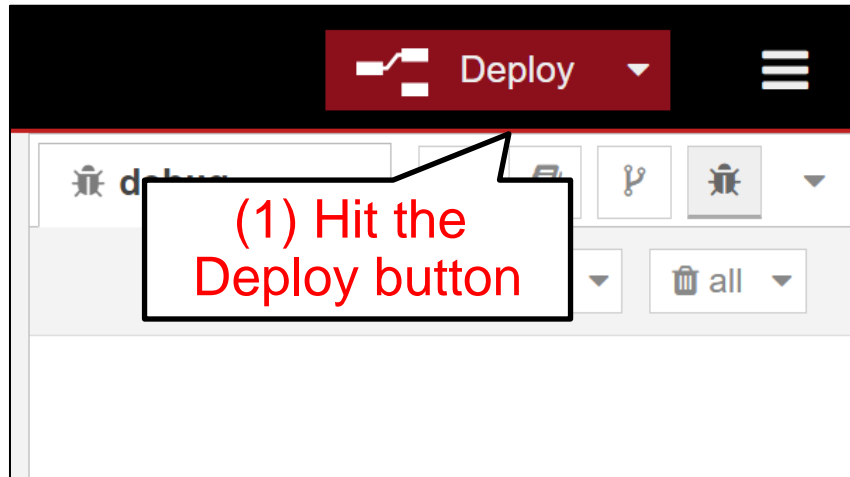
[Hands-on] Opening Dashboard UI (1/2)

After deploying the flow, move to the dashboard UI from the flow editor

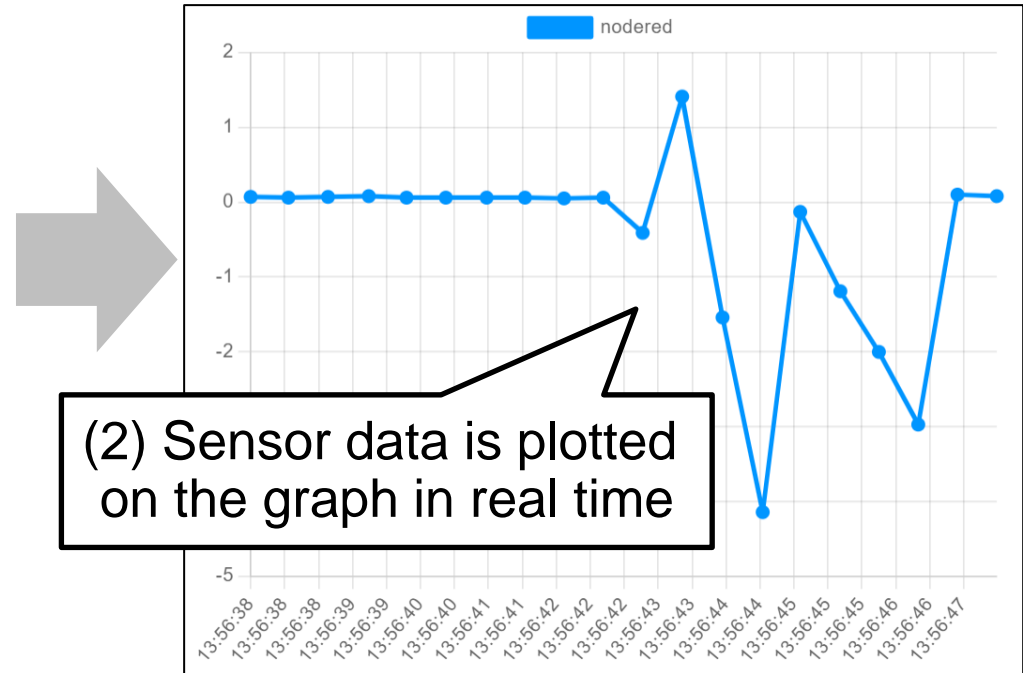
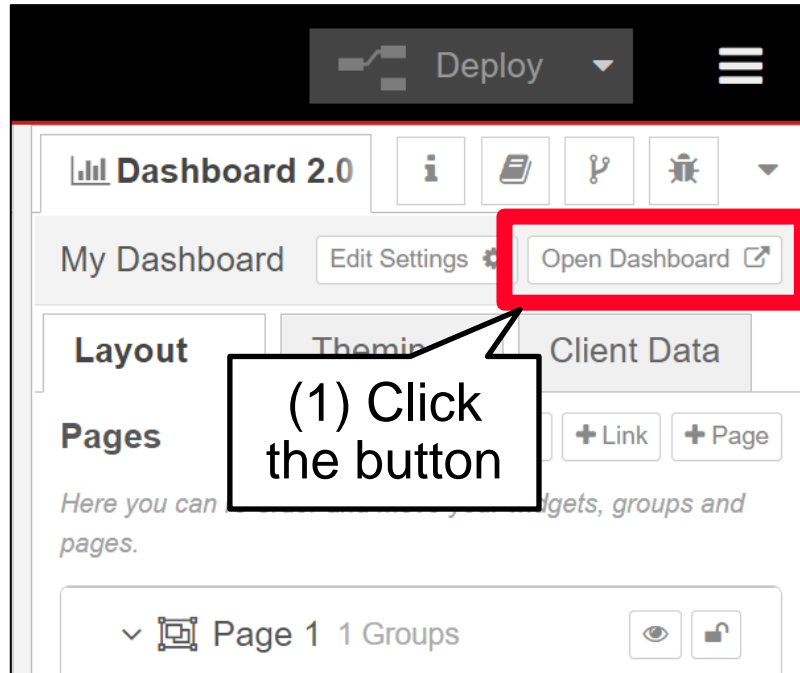
(1) Hit the "Deploy" button to execute the flow

(2) Open the menu in the sidebar to show all items

(3) Select the "Dashboard 2.0" from the menu

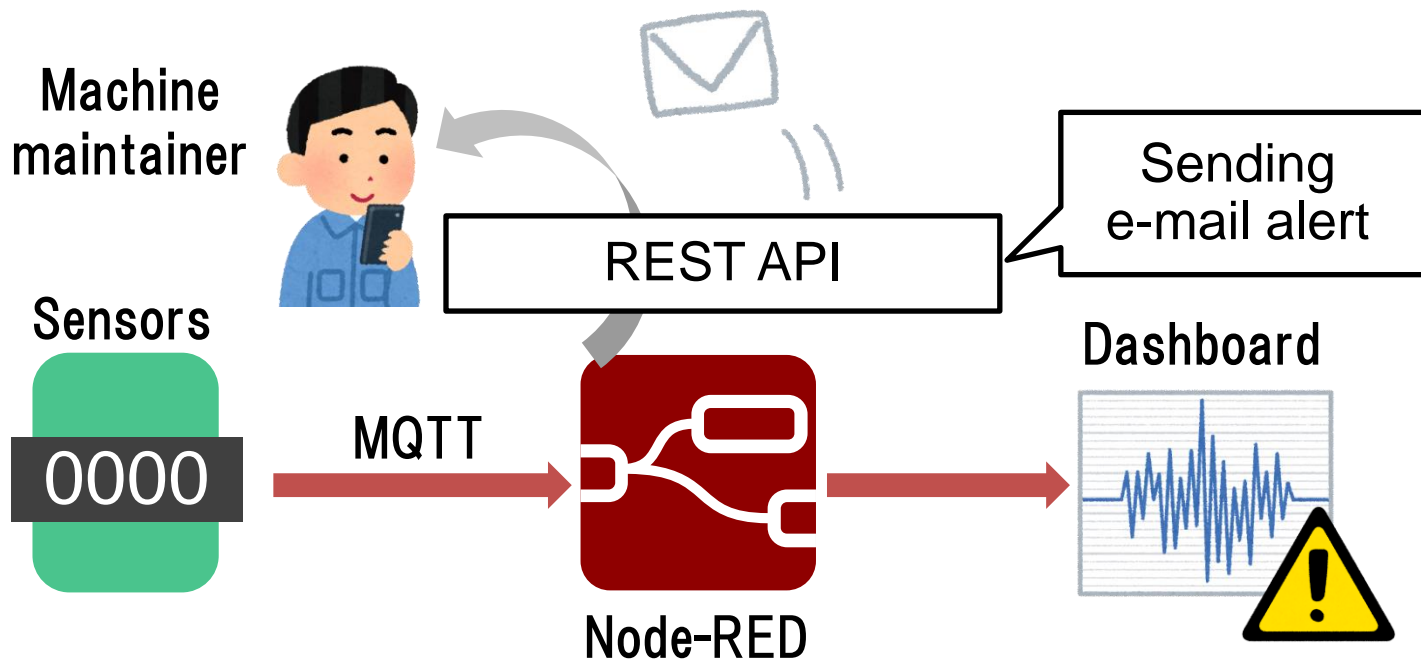


- (1) Click the "Open Dashboard" button in the Dashboard 2.0 tab
- (2) Dashboard UI in another browser tab shows the time series chart



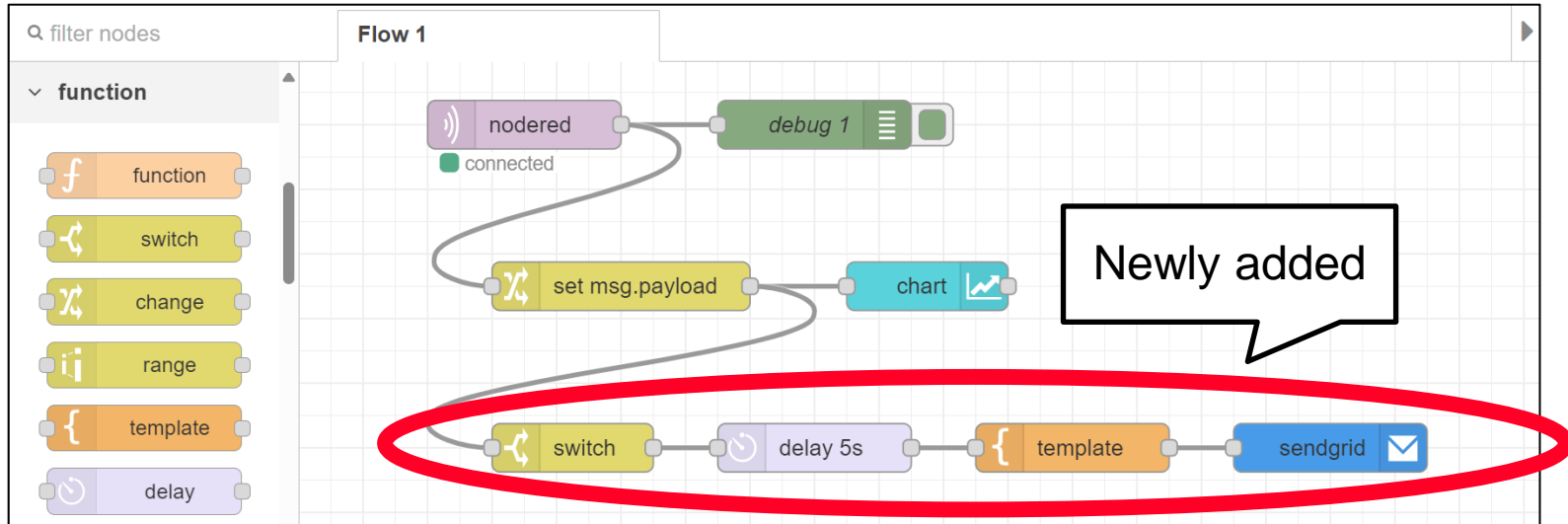
E-mail Notification

- Developing flow to send alert e-mail when the anomaly value is detected
- Using the SendGrid, e-mail delivery service, the flow sends e-mail to recipient



Add and connect the following nodes after the change node

- switch node: node to pass if the received values match the defined condition
- delay node: node to limit the number of messages in the specified period
- template node: node to write the template text for the e-mail message
- sendgrid node: node to send e-mail using the SendGrid service



Setting the threshold value to classify as an anomaly value
(In this hands-on, the received number greater than 20 is an anomaly value)

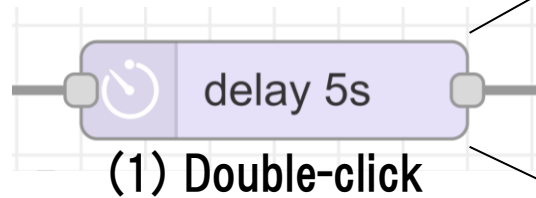
- (1) Double-click the switch node to open the node property UI
- (2) Change the comparison operator from "=" to ">", which means greater than
- (3) Change the data type from "az string" to "09 number" from the menu
- (4) Input the "20" in the text box
- (5) Click the "Done" button

The diagram illustrates the steps to configure a switch node for conditional filtering. On the left, a yellow 'switch' node is shown on a grid. A callout (1) 'Double-click' points to it. On the right, the 'Edit switch node' dialog is open. Callout (2) 'Select ">"' points to the comparison operator dropdown. Callout (3) 'Select "09 number"' points to the data type dropdown. Callout (4) 'Input "20"' points to the text input field. Callout (5) 'Click' points to the red 'Done' button.

[Hands-on] Configuring delay Node to Limit Messages

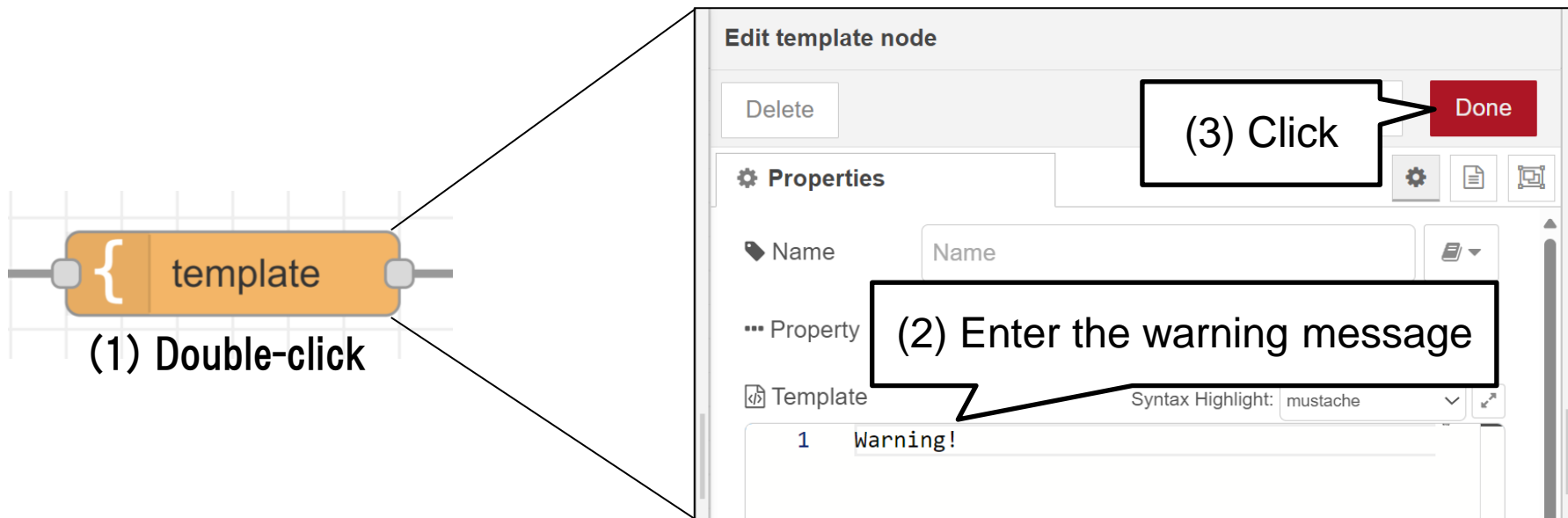
To prevent sending many e-mail messages in a short period of time, limit the maximum number of messages in a day using the delay node

- (1) Double-click the delay node
- (2) Select "Rate Limit" as action
- (3) Select "Day" to set 1 message per 1 day
- (4) Select "Drop intermediate messages" to remove overflow messages
- (5) Click the "Done" button



Defining the template message for sending e-mail

- (1) Double-click the template node to open the node property UI
- (2) In the template editor area, type the "Warning!" as the e-mail body
- (3) Click the "Done" button



[Hands-on] Configuring SendGrid Node

- (1) Double-click the sendgrid node to open the node property UI
- (2) Copy and paste the following API key into the "API key" field

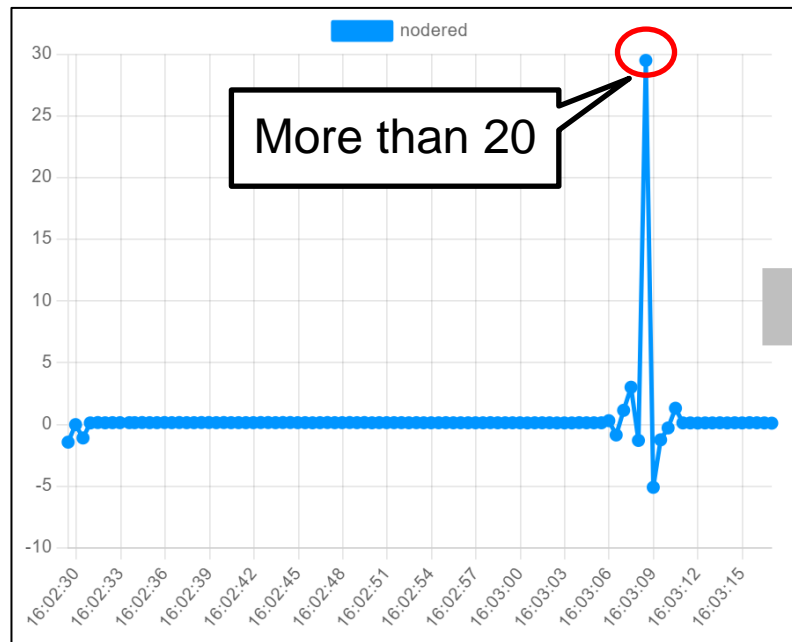
```
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
```

- (3) Input the sender's e-mail address, "hiac.it-kensyu74.hw@hitachi.com" into the "From" field
- (4) Input your e-mail address in the "To" field
- (5) Click the "Done" button

The diagram illustrates the configuration of a SendGrid node in a workflow editor. On the left, a blue node labeled 'sendgrid' with an envelope icon is shown on a grid. A line connects it to a larger 'Edit sendgrid node' dialog box on the right. The dialog box has a 'Properties' section with three fields: 'API key', 'From', and 'To'. The 'API key' field is empty, with a callout box (2) 'Paste an API key' pointing to it. The 'From' field contains the email address 'hiac.it-kensyu74.hw@hitachi.com', with a callout box (3) 'Input the sender's e-mail' pointing to it. The 'To' field contains the placeholder '<Your e-mail address>', with a callout box (4) 'Input the recipient e-mail' pointing to it. At the top right of the dialog are 'Cancel' and 'Done' buttons, with a callout box (5) 'Click' pointing to the 'Done' button. Below the 'sendgrid' node, a callout box (1) 'Double-click' points to the node itself.

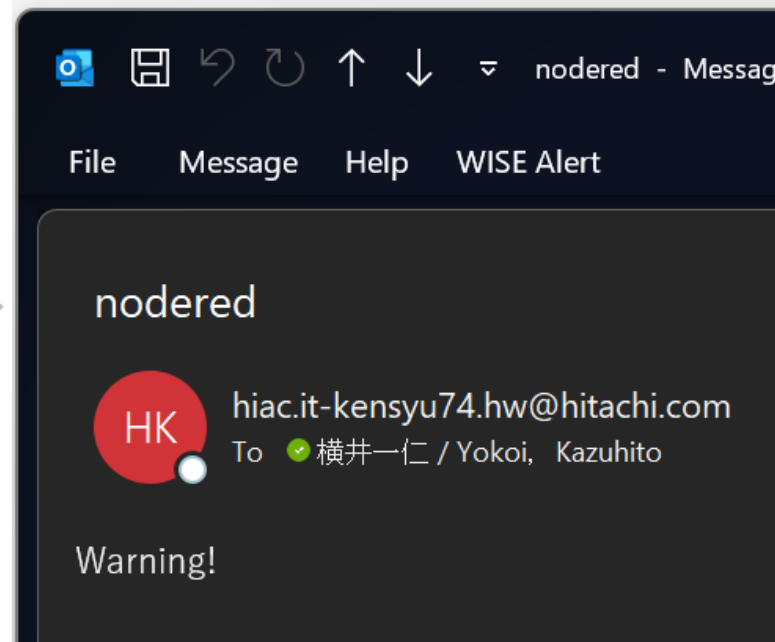
[Hands-on] Checking Behaviors of Sending E-mail

After the flow is deployed, Node-RED will send the alert e-mail to the recipient when the value is greater than 20.



Time-series graph on application

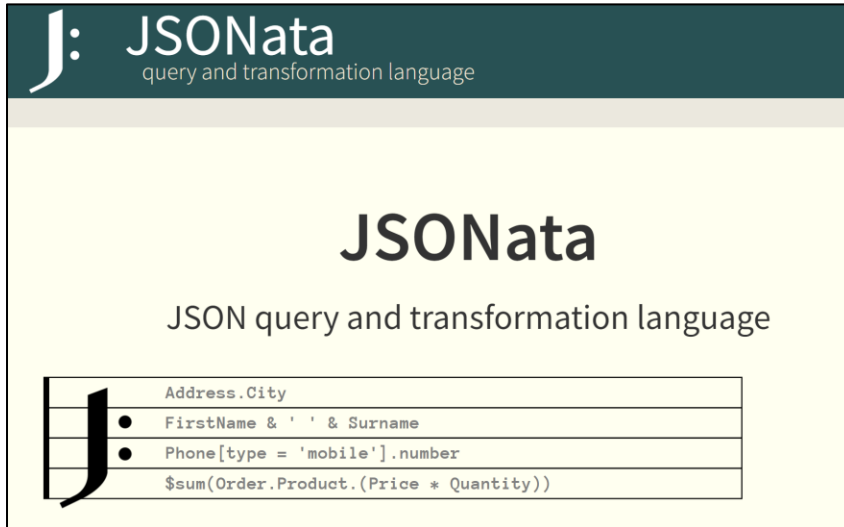
Send
e-mail



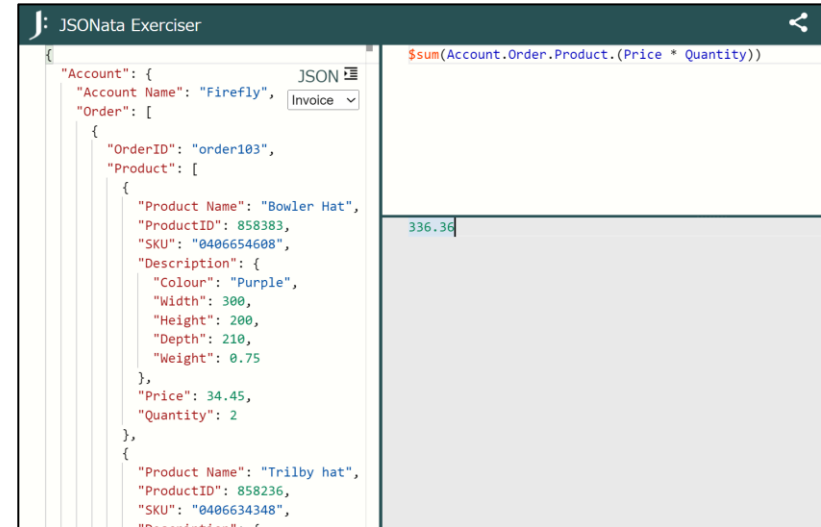
Received e-mail

A lightweight query and transformation language for JSON data

- Node-RED natively supports JSONata expressions
- It allows users to implement simple data handling without complex code



JSONata website



JSONata Exerciser

Using only the path to the key, "books.title", JSONata can extract the values as arrays from arrays containing JSON structures.

```
{  
  "books": [  
    {  
      "title": "Practical Node-RED Programming",  
      "author": "Taiji Hagino"  
    },  
    {  
      "title": "First time with Node-RED",  
      "author": "Node-RED User Group Japan"  
    },  
    {  
      "title": "Building your first app with Node-RED",  
      "author": "Hitachi Node-RED Evangelists"  
    }  
  ]  
}
```

Source JSON data

Transform JSON data
using JSONata

`books.title`

```
[  
  "Practical Node-RED Programming",  
  "First time with Node-RED",  
  "Building your first app with Node-RED"  
]
```

Output JSON data

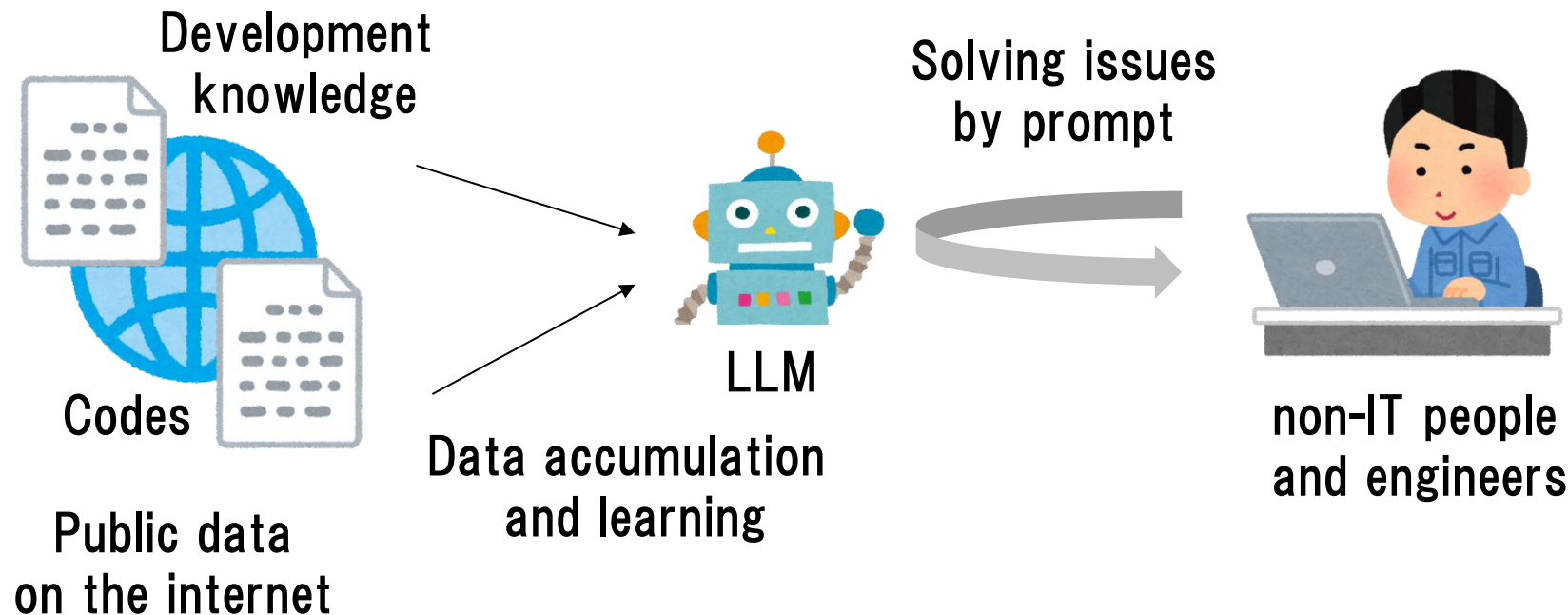
Setting the threshold value to classify as an anomaly value
(In this slide, the received value between -20 and 20 is an anomaly value)

- (1) Double-click the switch node to open the node property UI
- (2) Change the comparison operator from ">" to "JSONata exp"
- (3) Input the "msg.payload<-20 or 20<msg.payload" in the text box
- (4) Click the "Done" button

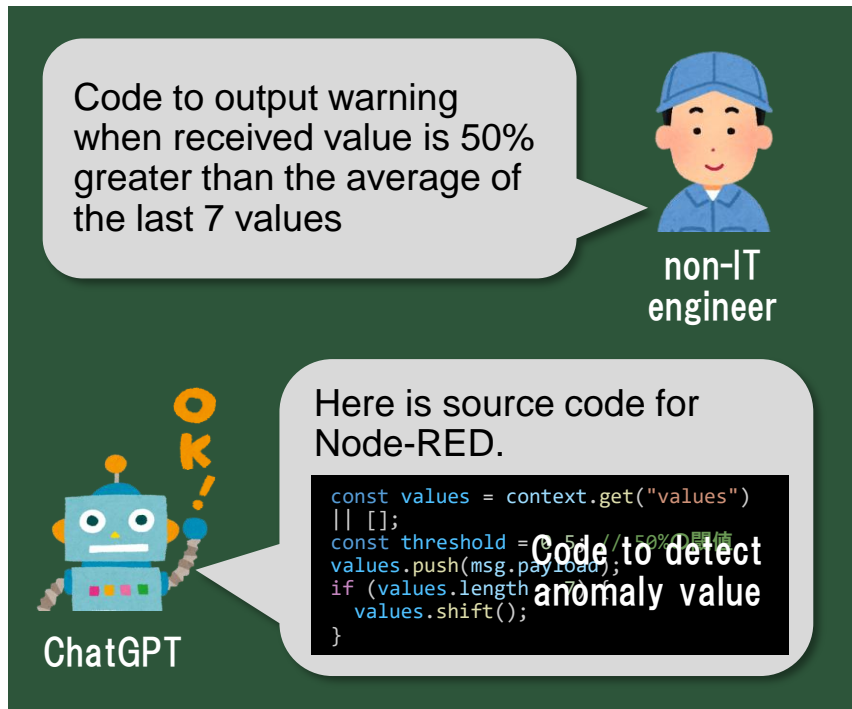


ChatGPT with Node-RED

Both non-IT people and engineers have used LLM like ChatGPT to solve the programming and system problems by themselves.



Generating code to detect sudden changes in the sensor data



Code to output warning when received value is 50% greater than the average of the last 7 values

non-IT engineer

Here is source code for Node-RED.

```
const values = context.get("values") || [];  
const threshold = 1.5; // 150%閾値  
values.push(msg.payload);  
if (values.length > 7) {  
  values.shift();  
}
```

Code to detect anomaly value

ChatGPT

[Algorithm of anomaly detection]

(1) Add value

Queue to store 7 values

(3) Calculate average

(4) Warn when latest value is 50% greater than average

(2) Remove the oldest value

ChatGPT can generate the complex code in a few seconds.

(1) Access the following URL from your browser

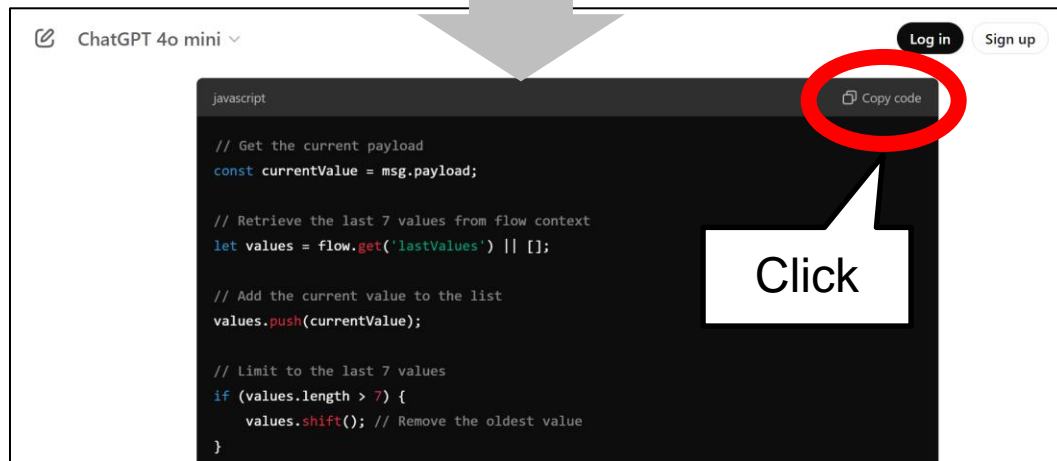
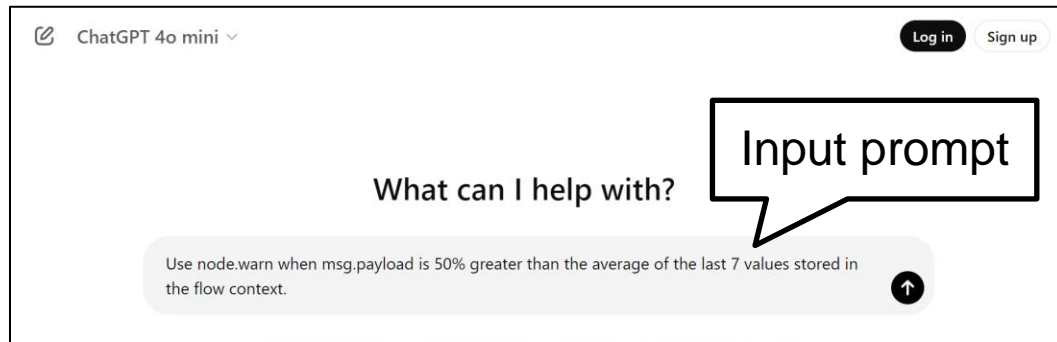
<https://chatgpt.com/>

(2) Input the following prompt in the text box

Use node.warn when msg.payload is 50% greater than the average of the last 7 values stored in the flow context.

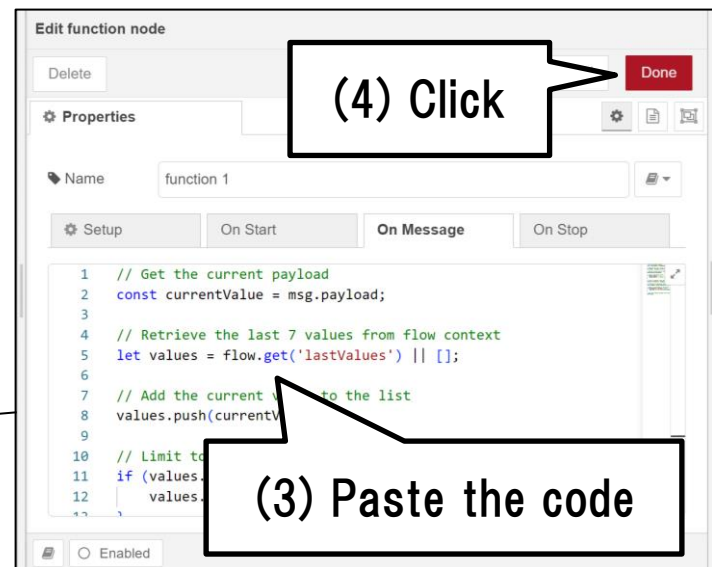
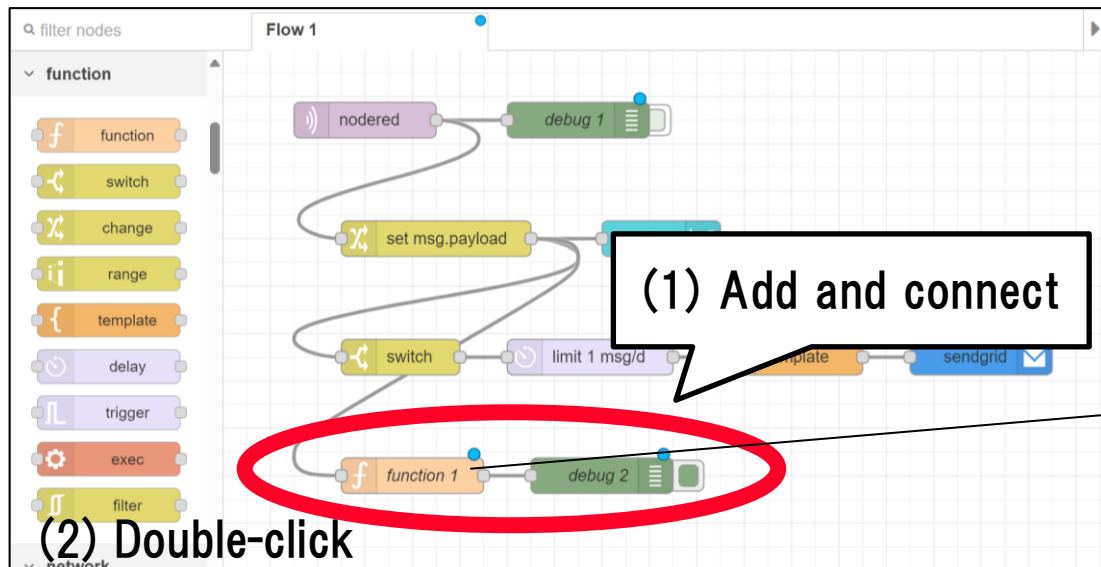
(3) Click the “copy code” button

(If there is no code in the result, try typing the same prompt again)



[Hands-on] Adding Code to the function Node

- (1) Add function node and debug node after the change node and wire them
- (2) Double-click the function node to open the node property UI
- (3) Paste the code from the clipboard into the editor of the "On Message" tab
- (4) Click the "Done" button to apply the code

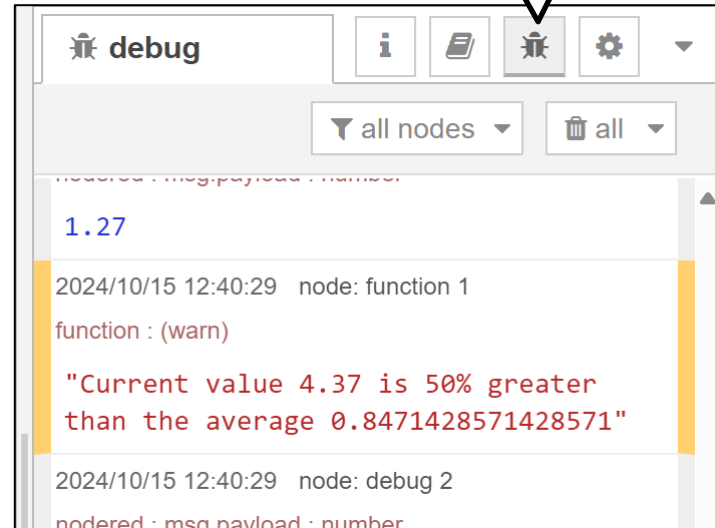


- (1) Hit the "Deploy" button to execute the flow
 - (2) Open the debug tab and check the warning message
- > Warning message will be displayed when the received value is changed suddenly.

(1) Hit the button



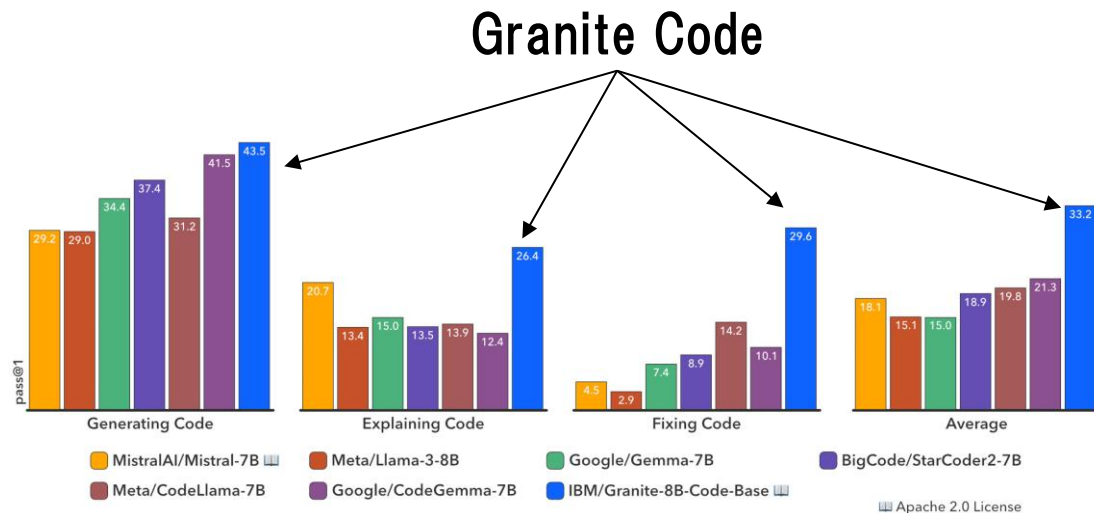
(2) Click



Granite Code with Node-RED

Local LLM for programming tasks released by IBM Research in May 2024

- High quality compared to other models
- Apache-2.0 license
- 116 programming languages supported
- Open data source
- Model size:
 - 3B parameters: 2.0 GB
 - 8B parameters: 4.6 GB
 - 20B parameters: 12 GB
 - 34B parameters: 19 GB



Quality Comparison with Local LLMs

<https://github.com/ibm-granite/granite-code-models>

Calculating Distance between Two Locations

Generating code to calculate the distance between two locations consist of latitude and longitude values

[Haversine Formula]

$$d = 2r \arcsin \left(\sqrt{\sin^2 \left(\frac{\phi_2 - \phi_1}{2} \right) + \cos(\phi_1) \cos(\phi_2) \sin^2 \left(\frac{\lambda_2 - \lambda_1}{2} \right)} \right)$$

It takes a time
to write the code :(



(1) Access the following URL from your browser

<https://www.ibm.com/granite/playground/code/>

(2) Input the following prompt in the text box

Haversine function in JavaScript

(3) Copy the generated code

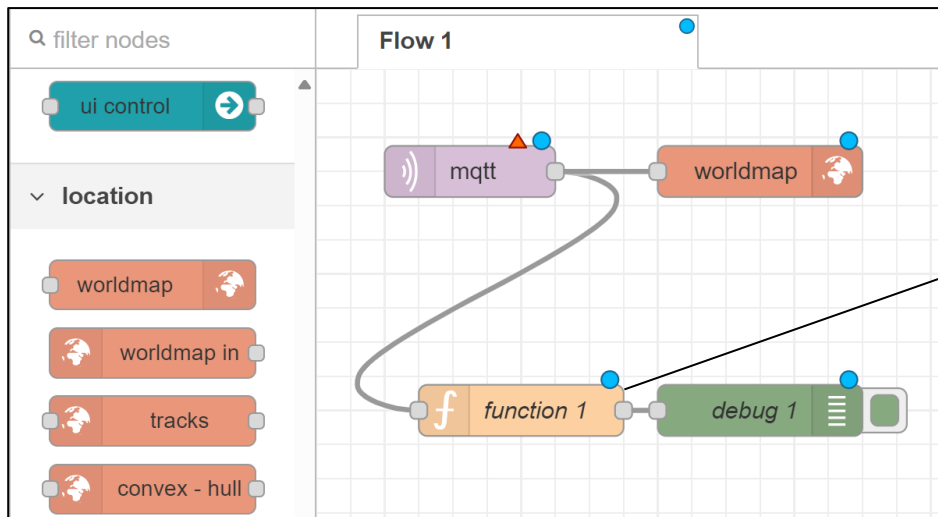
(If there is no code in the result, try typing the same prompt again)

The screenshot displays the IBM Granite Playground interface. At the top, it says "IBM Granite Playground" and "Code" is selected. A status bar indicates "24 API calls remaining". The main area shows the "Granite Code" header and a description: "The Granite Code models family are decoder-only models designed for code generative tasks, trained on code written in 116 programming languages". Below this is a text input box containing the prompt "Haversine function in JavaScript". A callout bubble labeled "Input prompt" points to this box. A large grey arrow points from the input box to the output area. The output area shows the generated JavaScript code for the Haversine function. A callout bubble labeled "Copy the generated code" points to this code block.

```
function haversine(lat1, lon1, lat2, lon2) {  
  const dLat = toRadians(lat2 - lat1);  
  const dLon = toRadians(lon2 - lon1);  
  const a =  
    Math.sin(dLat / 2) * Math.sin(dLat / 2) +  
    Math.cos(toRadians(lat1)) *  
    Math.cos(toRadians(lat2)) *  
    Math.sin(dLon / 2) *  
    Math.sin(dLon / 2);  
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));  
  const R = 6371; // Radius of the Earth in kilometers  
  return R * c;  
}
```

[Hands-on] Creating Flow to Show Distance and Location on Map

- (1) Add mqtt-in node and red worldmap node and then connect them
- (2) Configure mqtt-in node to use shiftr.io MQTT broker with "nodered" topic
- (3) Add function node and debug node after mqtt-in node and connect them
- (4) Double-click the function node to open the node property UI
- (5) Paste the code from the clipboard into the editor of the "On Message" tab



The screenshot shows the 'Edit function node' dialog box. A speech bubble points to the 'On Message' tab with the text 'Paste the code'. The code editor contains the following JavaScript code:

```
1 function toRadians(degrees) {
2   return degrees * Math.PI / 180;
3 }
4
5 function haversine(lat1, lon1, lat2, lon2) {
6   const dlat = toRadians(lat2 - lat1);
7   const dlon = toRadians(lon2 - lon1);
8   const a =
```

(1) Add the following code that specifies two locations at the end of the lines

```
msg.payload = {  
  distance: haversine(msg.payload.lat, msg.payload.lat, 36.09, -115.15)  
};  
return msg;
```

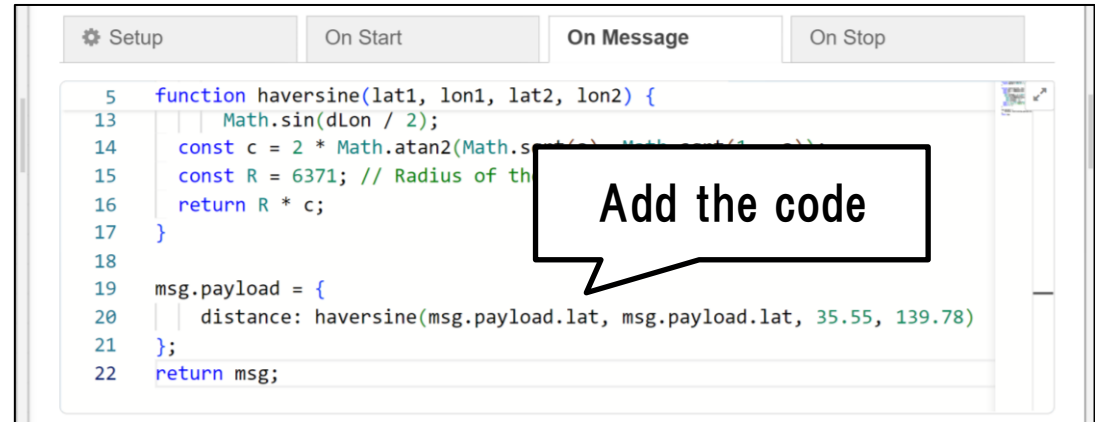
Latitude and longitude of
the received data

Latitude
and longitude
of LAS Airport

(2) Click the "Done" button
of the function node
to apply the code

(3) Hit the "Deploy" button
to execute the flow

* Screenshot is the case of Haneda Airport



[Hands-on] Executing the Flow

Access the following URL to open map

<https://<GitHub Codespaces URL>/worldmap>

The screenshot displays the Node-RED web interface. On the left, a flow named 'Flow 1' is shown with three nodes: 'nodered' (purple), 'worldmap' (orange), and 'function 1' (orange). The 'worldmap' node is connected to 'function 1'. Below the flow, a speech bubble points to the 'function 1' node with the text 'Distance from airport'. On the right, the 'debug' console shows five log entries, each containing a 'distance' property with a numerical value. A speech bubble points to these log entries with the text 'Distance from airport'. On the far right, a map titled 'Node-RED - map all the things' is displayed. A red pin is placed on the map, and a speech bubble points to it with the text 'Current location'. The map shows a street view of an area with various landmarks and stations.

Node-RED

Flow 1

nodered

worldmap

function 1

debug

all nodes

2024/10/7 11:17:44 node: debug 1
nodered : msg.payload : Object
> { distance: 6.056573178515396 }

2024/10/7 11:17:45 node: debug 1
nodered : msg.payload : Object
> { distance: 6.057187495802668 }

2024/10/7 11:17:45 node: debug 1
nodered : msg.payload : Object
> { distance: 6.057187495802668 }

2024/10/7 11:17:46 node: debug 1
nodered : msg.payload : Object
> { distance: 6.057187495802668 }

2024/10/7 11:17:46 node: debug 1
nodered : msg.payload : Object
> { distance: 6.057187495802668 }

Distance from airport

Current location

Node-RED - map all the things

Current location



Conclusion

Node-RED is a visual programming tool ideal for rapid development.

- It natively supports IBM open technologies such as MQTT and JSONata.
- Recently, leveraging the latest LLMs like Granite Code and ChatGPT is the hot topic in Node-RED.

Try to develop your own systems by yourself.

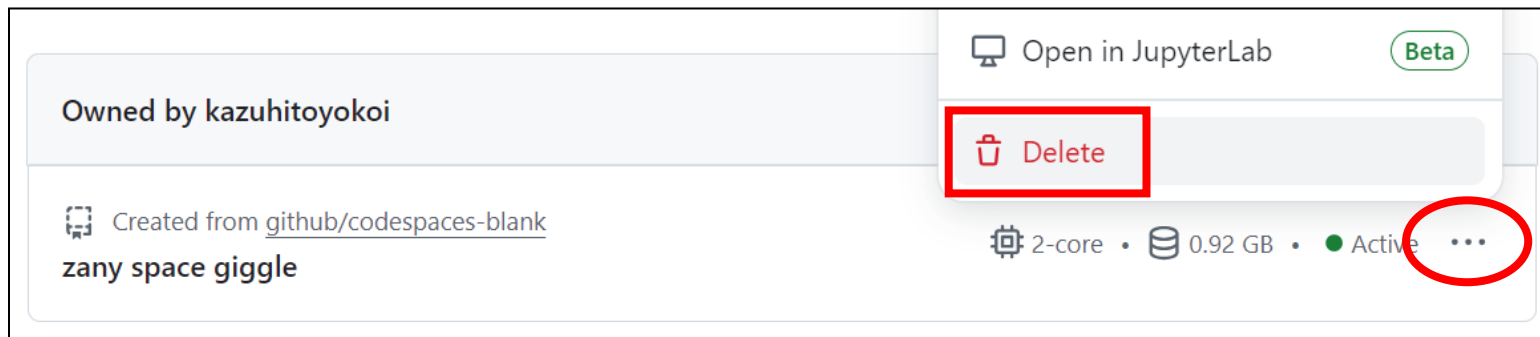
When you have finished hands-on,
you can delete the Codespaces environment by following steps.

(1) Go to the Codespaces page

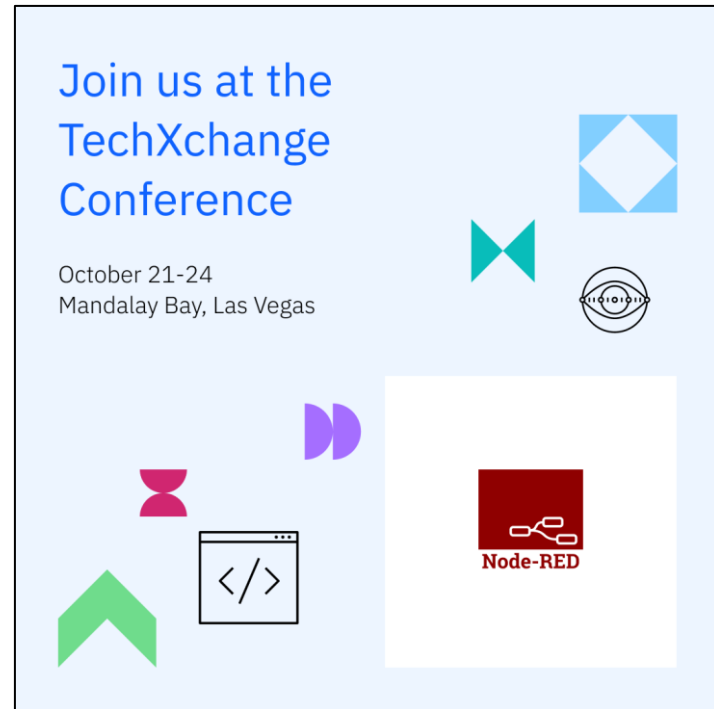
<https://github.com/codespaces>

(2) Click the "..."

(3) Select "Delete" item on the menu



- **Boosting Productivity of Node-RED with Large Language Model**
 - October 21 12:00PM - 12:30PM
 - South, Reef D, Level 2
- **Node-RED hands-on with Open Technologies**
 - October 22 3:00PM - 4:00PM
 - South, Banyan F, Level 3
- **Tips for controlling HVAC and lighting with a voice from xR using Code Engine and watsonx.ai!**
 - Thursday, October 24 11:30AM - 12:30PM
 - South, Surf C, Level 2



<https://reg.tools.ibm.com/flow/ibm/techxchange24/sessioncatalog/page/sessioncatalog?search=Node-RED>

END

**Node-RED Hands-on
with Open Technologies**

October 22, 2024
Kazuhito Yokoi
Hitachi Academy

- Node-RED is a trademarks of OpenJS Foundation.
- GitHub Copilot is a trademark of GitHub, Inc.
- SendGrid is a trademark of Twilio SendGrid, Inc.
- Granite Code, Code Engine and watsonx.ai are trademarks of International Business Machines Corporation.
- ChatGPT is a trademark of OpenAI, Inc.
- Other company, product or service names may be trademark or registered trademark of others.