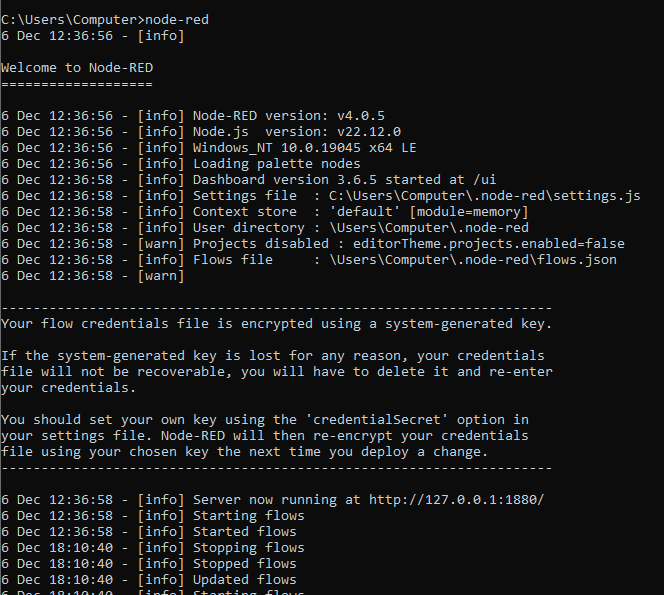
**1. Installation of Node-Red in PC**  
Node-red is the Low-code programming for event-driven applications. It is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

**1.1 Steps for creating a local host node red in our own PC:**

1. Go to [nodered.org](https://nodered.org/) website and have some knowledge about node red platform.
2. To run locally the node-red go to getting started option on the website  
   <https://nodered.org/docs/getting-started/>
3. To install Node-RED locally you will need a [supported version of Node.js](https://nodered.org/docs/faq/node-versions).
4. If you are using Windows, detailed instructions for installing Node-RED can be found [here](https://nodered.org/docs/getting-started/windows).
5. Download the latest LTS version of Node.js from the official [Node.js home page](https://nodejs.org/en/).
6. Once installed, open a command prompt and run the following command to ensure Node.js and npm are installed correctly.

Using cmd:  node --version && npm –version

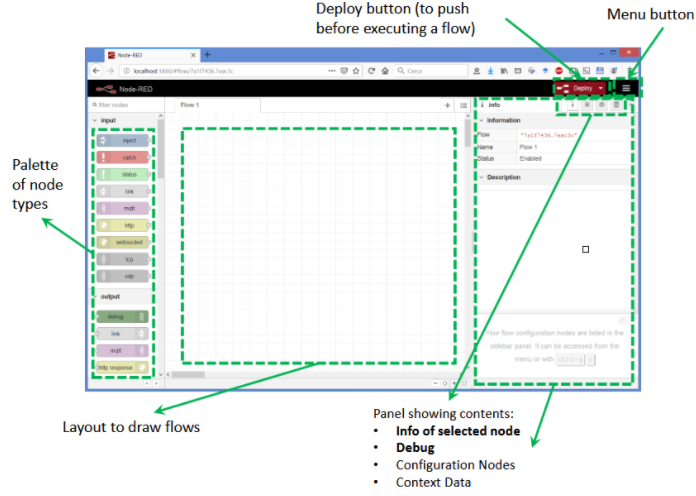
1. Now you are ready to run node-red command  
   Using cmd:  node-red

  
***Congratulations on successfully installing and running Node-RED! Now that Node-RED is up and running***

**1.2 Access the Node-RED Editor**

* Open a web browser and go to the URL mentioned in the logs: <http://127.0.0.1:1880/>.
* This will open the Node-RED flow editor where you can start designing your flows.

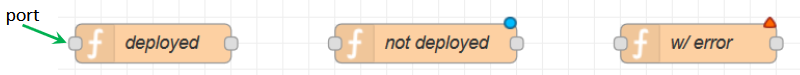
**1.3 Explore the Interface**



**1.3.1 Nodes:**

Nodes consist of code that runs in the Node-RED service (JavaScript .js file), and an HTML file consisting of a description of the node, so that it appears in the node pane with a category, colour, name and an icon, code to configure the node, and help text Nodes are added to a flow by simple **drag drop.**

A node can be linked to (multiple) input and (multiple) output via its ports which enable messages to be passed between nodes.

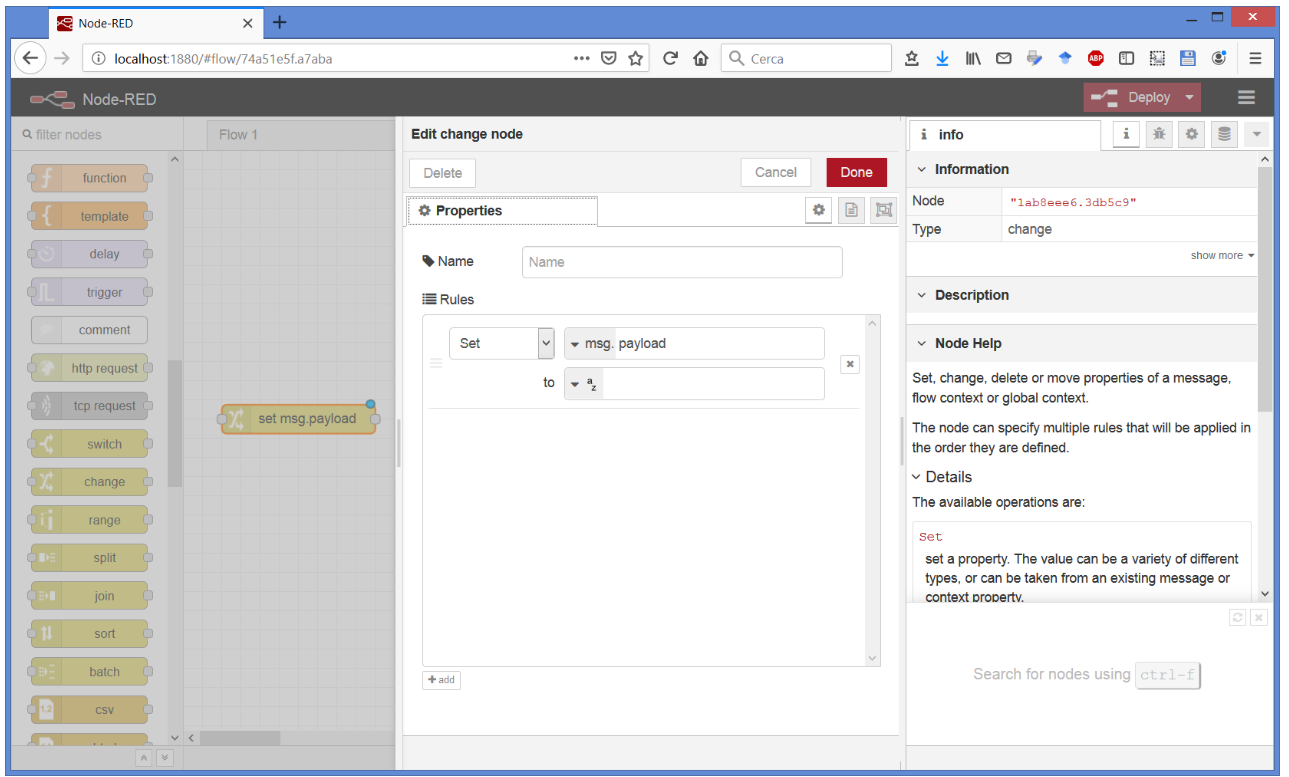
If a node has been changing after the latest deployment, then it displays a blue circle above it. If there are errors with its configuration, it displays a red triangle

Some nodes include a button on either its left or right edge. These allow some interaction with the node from within the editor. The Inject and Debug nodes are the only core nodes that have buttons.



A node configuration can be edited by double clicking on the node,or pressing Enter when the workspace has focus. If multiple nodes are selected, the first node in the selection will be edited.

The node edit dialog has typically three sections: Properties, Description, Appearance. The Properties section is used to set what the node does



Wires define the connections between node input and output endpoints in a flow.

They (typically) connect the output endpoints of nodes to inputs of downstream nodes indicating that messages generated by one node should be processed by the connected node next.

It is possible to connect more than one node to an endpoint using wires. It is also possible to connect downstream nodes to upstream nodes to form loops.

When multiple nodes are connected to an output endpoint, messages are sent to each connected node in turn in the order they were wired to the output.

When more than one node output is connected to an input endpoint, messages from any of those nodes will be processed by the connected node when they arrive

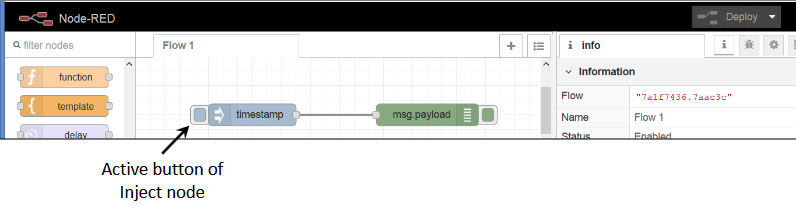
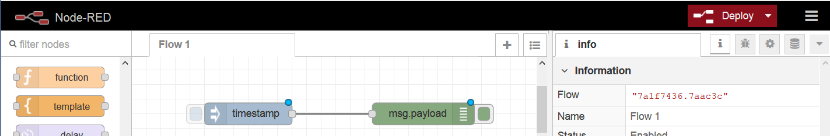
 **1.3.2 How to create wires?**

Left click on an output node port and, while holding down the mouse button, move to the destination input node port. Release the mouse button.

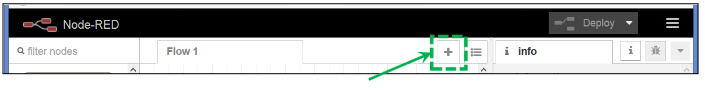
Flow Deployment is needed before executing the flow itself.

The Deploy button is on the top right corner and changes colour from grey to red when any change has been made to a flow to indicate that in it needs to be deployed.

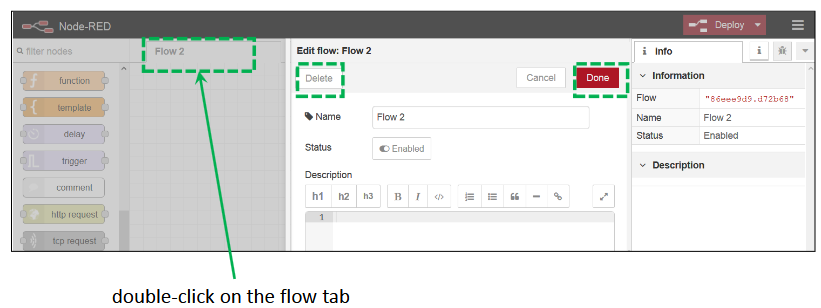
After deployment the flow can be run, e.g. clicking on the button of an Inject node.



**1.3.4 Create a Flow:**

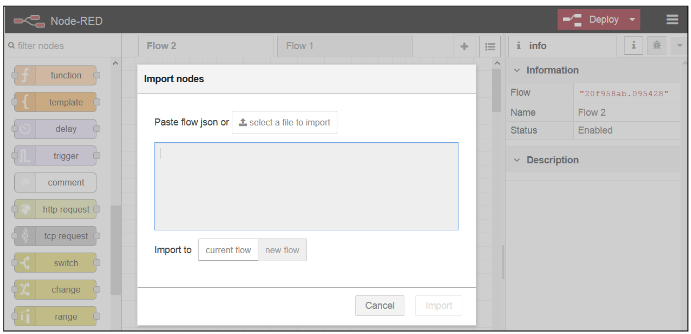
**1.3.5 Delete a Flow:** double-click on the flow tab and press the button «Delete»

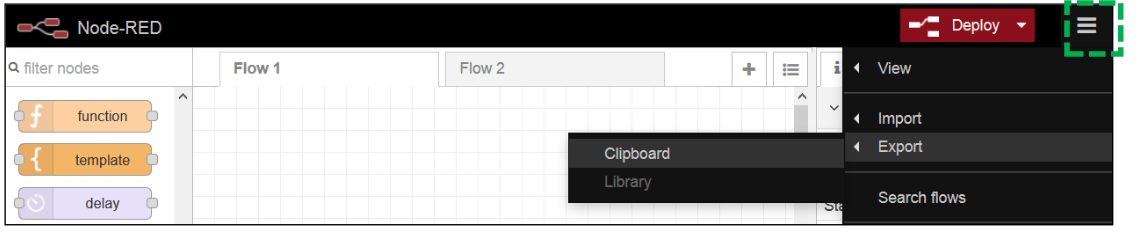
**1.3.6 Rename a Flow:** double-click on the flow tab, change the name and press the button «Done»

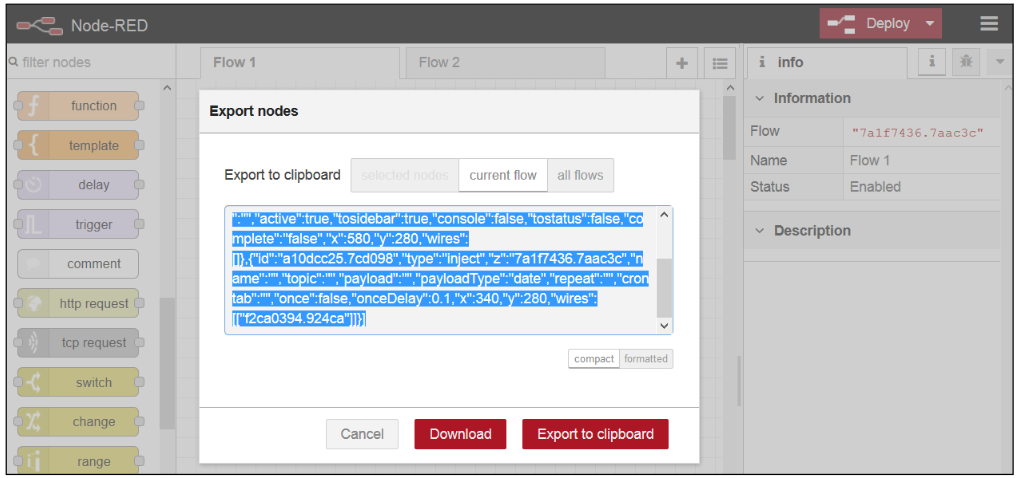


**1.3.7 Import a Flow**: click on the main menu and select «Import», «Clipboard»

Then copy & paste the flow or select an input Json file

**1.3.8 Export a Flow:** click on the main menu and select «Export», «Clipboard»

Then copy the flow or download it as a Json file.

A Node-RED flow works by passing messages between nodes.

The messages are simple JavaScript objects that can have any set of properties.

The message is passed in as an object called msg. Messages usually have a payload property (msg.payload), i.e. the default property containing the body of the message.

Node-RED also adds a property called \_msgid to each message, i.e. an identifier of the message.

* **1.4 Set Up the Credential Secret**
* To secure sensitive data in your flows, update the settings.js file to define a credential Secret.
  + Open C:\Users\Computer\.node-red\settings.js.
  + Find or add the line:  
    credentialSecret: "your-secure-secret",
  + Save the file and restart Node-RED.

**1.5 Secure node-red**

* To protect Node-RED when accessed remotely, consider adding a username and password:

1. Open the settings.js file using VS code or notepad
2. Find the admin Auth section and uncomment it. Update it like this:  
   adminAuth:   
   {

type: "credentials",

users: [{ username: "admin",

password: "$2b$08$...",

permissions: "\*" }]

},

1. After uncommenting use the bcrypt tool to generate a hashed password:
2. Install bcrypt globally:  
    **npm install -g bcrypt**
3. If you want to use bcrypt only within the context of your Node-RED project, navigate to the. node-red directory:  
   **cd C:\Users\Computer\.node-red**  
   **npm install bcrypt**

A screenshot of a computer

Description automatically generated

1. Run: node -e "console.log(require('bcrypt').hash Sync('your-password', 8))"
2. Replace your-password with your desired password.  
   
3. Copy the output and paste it in the password field (settings.js) above.
4. Save and restart Node-RED for the changes to update.

**1. 6 Access Node-RED Remotely**

* In the settings.js file find the line that defines the uiHost. If it's not there, add it:  
  uiHost: "0.0.0.0",
* After editing the settings.js file, restart Node-RED for the changes to take effect:  
  node-red
* From another device on the same network:

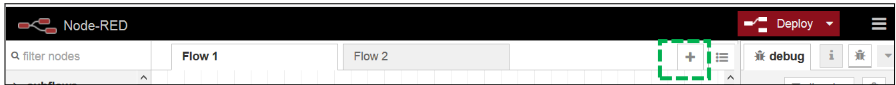
Open a web browser and navigate to:

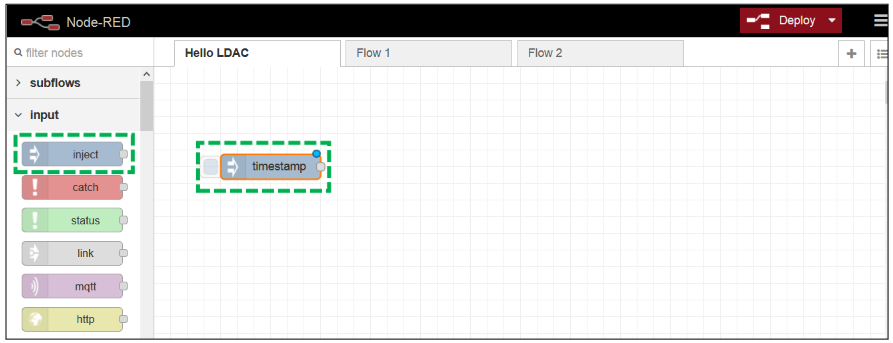
http://<Your-PC-IP>:1880/

Replace <Your-PC-IP> with your PC's local IP address, e.g., <http://192.168.1.100:1880/>.

### **2. Getting started with Node-red: Create Your First Flow**

**Create a new Flow and rename it**

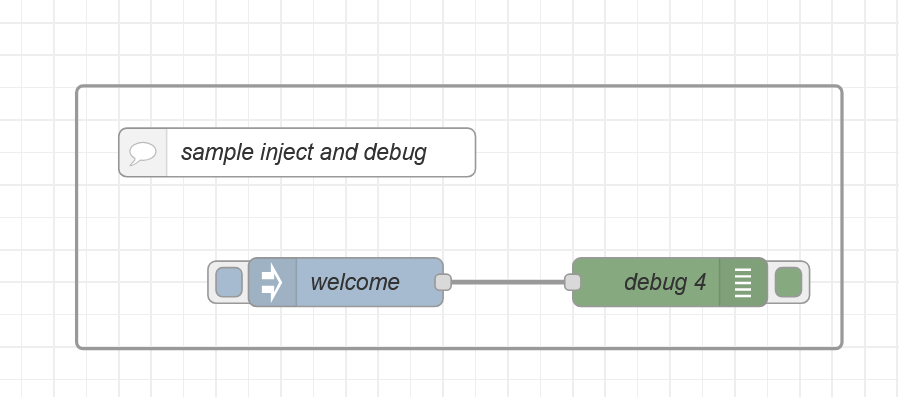
**Add Inject node (from input nodes)**



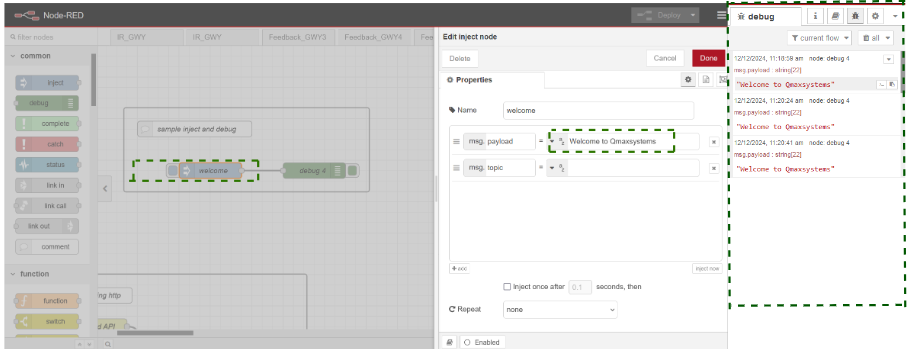
**Steps to create without using any MQTT and HTTP .**

The structure of a message can be better understood if it is passed to a Debug node. Thus, the contents of the message can be viewed in the Debug sidebar.

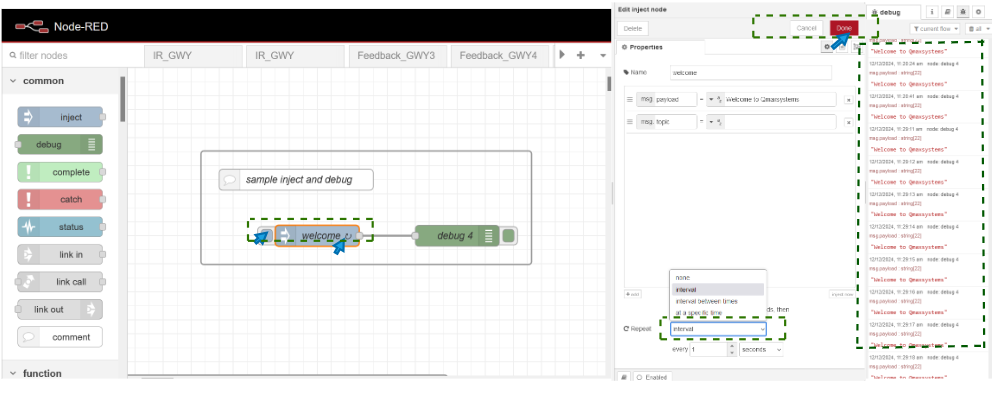
By default, the Debug node will display the **msg.payload** property, but it can be configured to display any property or the whole message.



* Drag an **inject node** (Input node) onto the workspace. This node can trigger an event.
* Drag a **debug node** (Output node) and connect it to the inject node. The debug node will display messages in the debug panel.
* Deploy your flow using the **Deploy** button in the top-right corner.
* Click the button on the **with manual injection inject node** to see the output in the debug panel on the right.

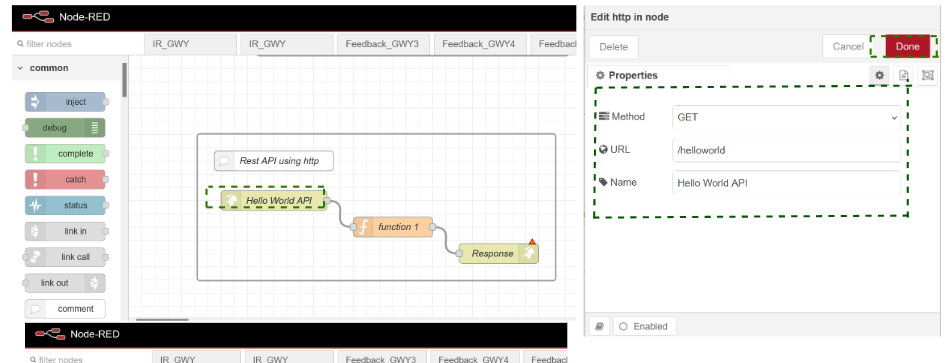


Click the button with manual injection and the interval inject node to see the output in the debug panel on the right.



**Steps to create a REST API endpoint using Node-RED1**

1. Open Node-red in the browser using http://<Your-PC-IP>:1880/
2. Use "http in" node to create an api over Node-RED url.
3. Give a URL. Example: "/helloworld" it should be available at  
   <http://localhost:1880/helloworld>

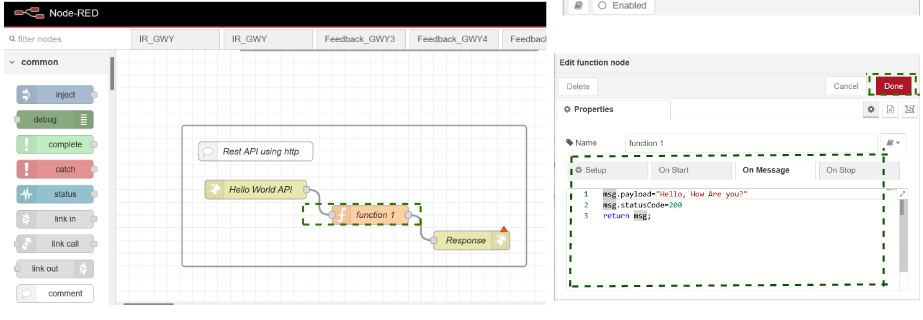


1. Now, drag the function node and connect with the api node,   
   also add the below config to function node:

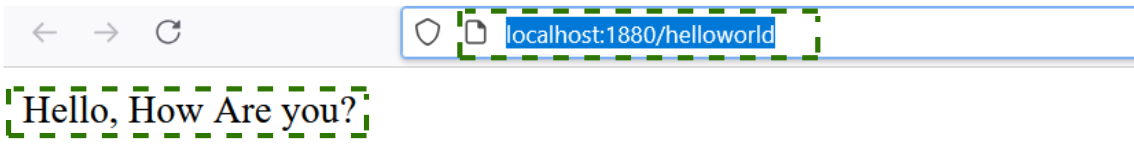
msg.payload="Hello, How Are you?"

msg.statusCode=200

return msg;



1. Finally, add "http response node" and connect with the function node.  
   Use the following config-  
   Name: Response   
   Status Code: msg.statusCode
2. Finally, Use URL: <http://localhost:1880/helloworld> to check the sample “helloworld” rest api



**1.3 Steps to Connect to an MQTT Broker in Node-RED:**

**1.3.1 Add and Configure the MQTT Nodes (MQTT In)**

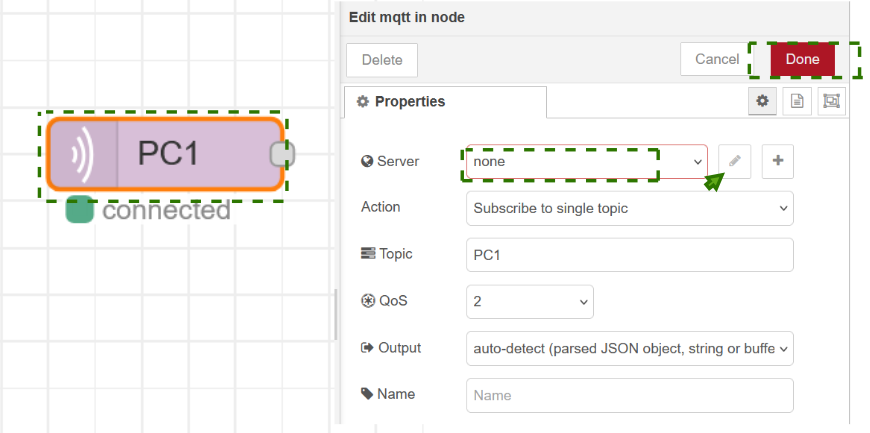
* Drag an **MQTT In** node onto the Node-RED workspace.
* Double-click the MQTT In node to configure it.
* In the properties, select the server address or select the pencil icon to add a new broker
* In the configuration window, enter:

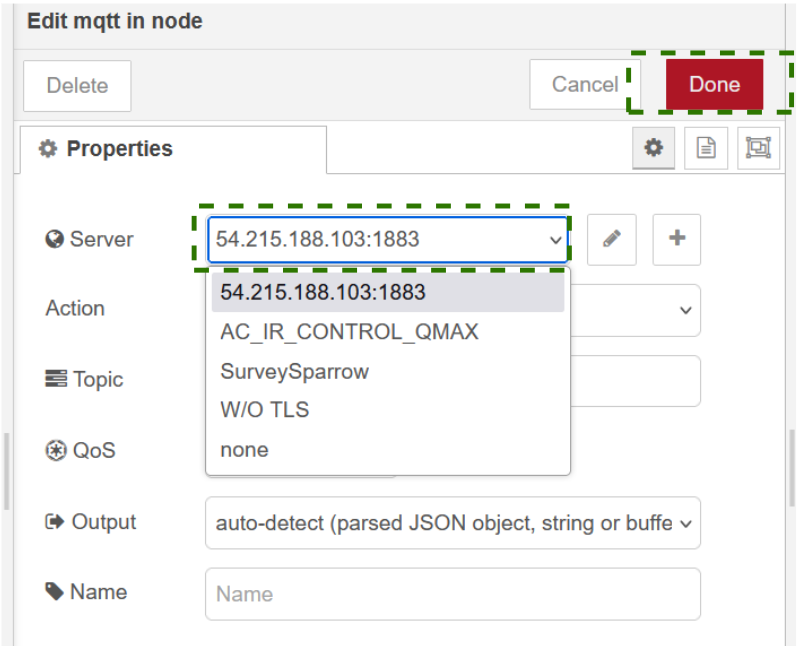
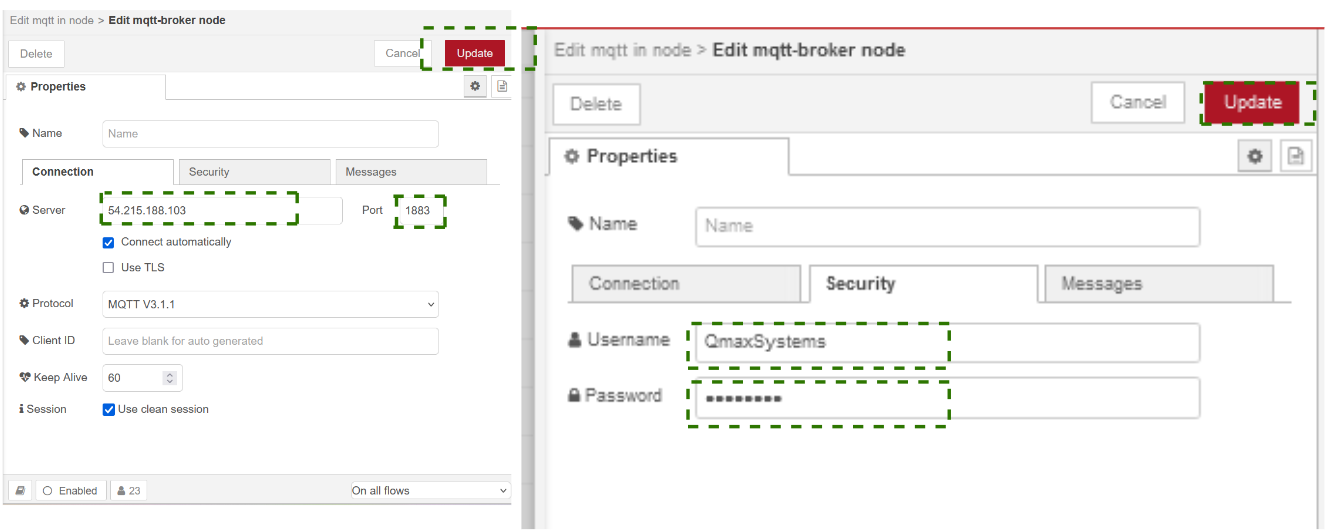
**Server**: The broker's IP address or hostname

**Port**: The default MQTT port is 1883

* Click **Add** to save the configuration.

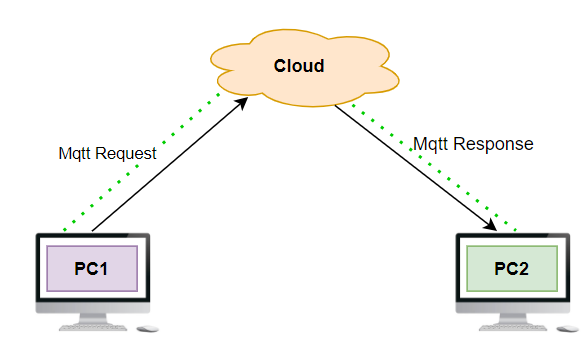
**1.3.5 Delete a Flow:** double-click on the flow tab and press the button «Delete»





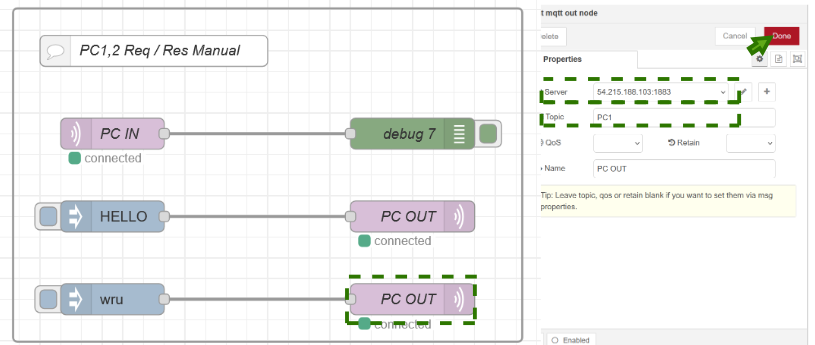
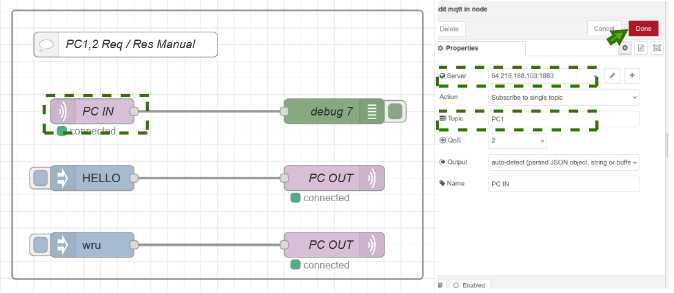
**1.3.2 Attach a Debug Node**

* Drag a **Debug** node onto the workspace.
* Connect the output of the MQTT In node to the Debug node.
* Deploy the flow.



**1.3.3 Steps to Connect PC1 to MQTT Broker on PC2 with manual Request and Response**

**Add and Configure the MQTT Nodes**

* Drag an **MQTT Out** node onto the Node-RED workspace.
* Double-click the MQTT out node to configure it.
* Select the same broker address as given for the MQTT In node
* Ensure that the topic is also same as given for the MQTT In node

**1.4 Steps to Connect PC1 to MQTT Broker on PC2 with Automatic Request and Response**

* Drag an **MQTT Out** node onto the Node-RED workspace.
* Double-click the MQTT out node to configure it.
* Select the same broker address as given for the MQTT In node
* Ensure that the topic is also same as given for the MQTT In node
* Add a function node as mentioned in the picture and enter the code.  
     
  // Log the original message

const originalMessage = msg.payload;

// Generate a response message

const responseMessage = {

topic: "response\_topic", // Response topic

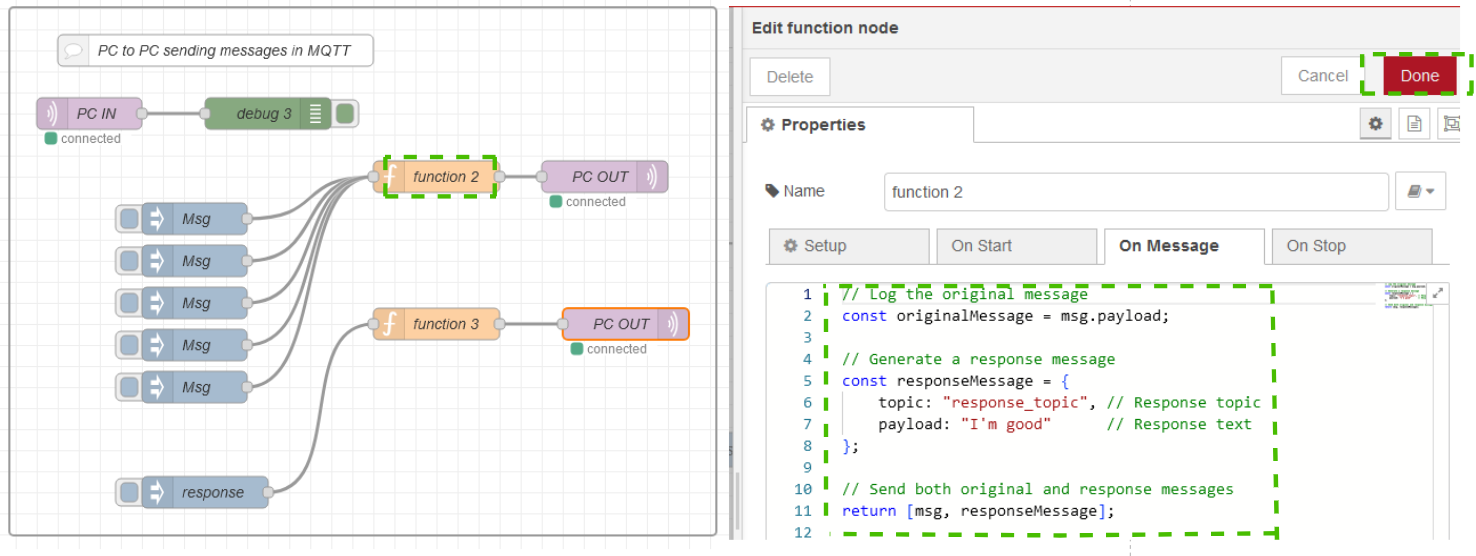
payload: "I'm good" // Response text

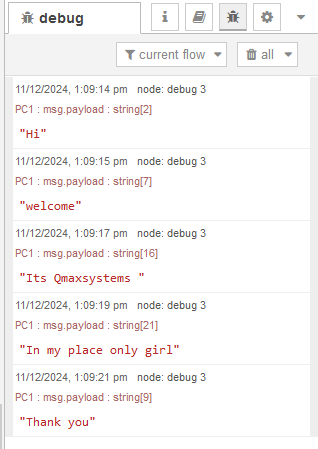
};

// Send both original and response messages

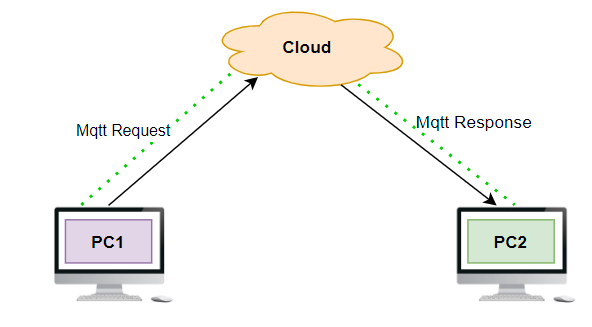
return [msg, responseMessage];

* Likewise do the same procedure in the other PC node red with another host
* Give the same topic and broker address for the MQTT node In and Out
* Deploy the flow.
* Both the PC will get the message deployed

**Debug output log:**



**1.5 Steps to Connect to an MQTT Broker from one PC1 to another PC2 automatically**



**PC1**

* Drag an **MQTT Out** node onto the Node-RED workspace.
* Double-click the MQTT out node to configure it.
* Select the same broker address as given for the MQTT In node
* Ensure that the topic is also same as given for the MQTT In node
* Add a function node as mentioned in the picture and enter the code.

// Check if the message payload is equal to "hi" (case-insensitive)

if(msg.payload === "green"){

// Change the message payload to "hello" if the input is "hi"

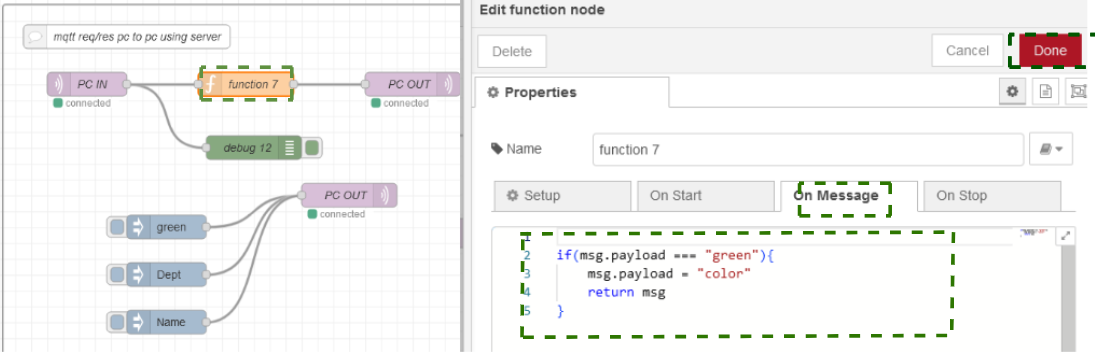
msg.payload = "color"

// return the modified message

return msg

}

* Likewise do the same procedure in the other PC node red with another host
* Give the same topic and broker address for the MQTT node In and Out
* Deploy the flow.
* Both the PC will get the message deployed and generate the response automatically



**Debug log Output:**



**1.6 Steps to Connect to an MQTT Broker from one PC sender to another PC receiver automatically**

* Drag an **MQTT Out** node onto the Node-RED workspace.
* Double-click the MQTT out node to configure it.
* Select the same broker address as given for the MQTT In node
* For the PC sender give the topic for MQTT in and then to the PC receiver give the same topic in MQTT out
* Add a function node as mentioned in the picture below and enter the code.

// Check if the message payload is equal to "hi" (case-insensitive)

if (msg.payload === "Name") {

// Change the message payload to "hello" if the input is "hi"

msg.payload = "Divi"

// return the modified message

return msg }

if (msg.payload === "Dept") {

msg.payload = "ECE"

return msg

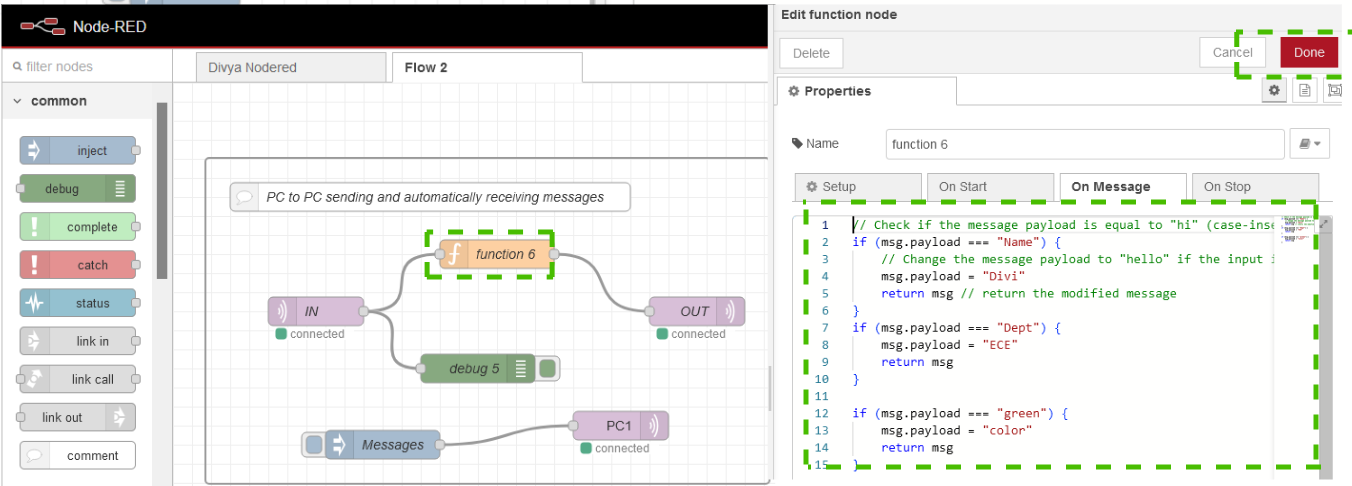
}

if (msg.payload === "green") {

msg.payload = "color"

return msg }

* Deploy the flow.
* Both the PC will get the message deployed and generate the response automatically when the PC sender injects the message
* Also, the PC receiver receives the message response as in the function



**Debug log Output:**

