FlowFuse

eBook

The Ultimate Beginner Guide to a Professional Node-RED

Published March 2024

Introduction

Node-RED is the open source, low-code programming environment that makes it easy to connect hardware, sensors, actuators and services using a visual development environment. This makes it possible for even non-developers to build applications, which has led to its wide popularity. A large library of nodes and flows makes it trivial to connect different types of analog and digital services, including everything from industrial protocols such as OPC-UA, Modbus, and Siemens S7, and databases to weather services. The Node-RED dashboard makes it easy to visualize data in dashboards, and unlike other dashboards, Node-RED dashboards can include actions and events that respond to the data.

Node-RED allows developers to connect a wide variety of endpoints, such as industrial PLCs, APIs, databases, enterprise applications and online services. Users of Node-RED visually drag 'nodes' from a palette that represents the endpoints and then connect them into flows to accomplish the desired task.

Node-RED applications are accessible via a web browser and can be used for a wide variety of use cases, including:

- IoT and IIOT use cases, such as collecting data from different pieces of factory equipment or geographically dispersed devices.
- Creating dashboards that visualize data and allow for event triggers to occur based on the data.
- Automation and integration of digital platforms from chatbots to data integration pipelines.
- Extract, transform and integrate data from many different sources in the enterprise.
- Home automation.
- Integrating data with machine learning models.

Node-RED is an open source project hosted at the OpenJS Foundation. It is made available under the Apache Software License so individuals can use Node-RED free of charge. Participation in the open source project is encouraged to help support the Node-RED community.

This ebook will take you through the history of Node-RED, an indepth look at its key features, and how you can get started. It is a good starting point for anyone interested in learning more about this popular integration software and making citizen development a reality in your enterprise.

Table of Content

Chapter 1

History of Node-RED

In January 2013, FlowFuse's co-founder and CTO, Nick O'Leary, could have never foreseen that his fun little proof-of-concept project would become Node-RED, an open source low-code environment with millions of deployments in IoT and automation.

He was working in IBM's Emerging Technology Group playing around with capturing data from devices and doing interesting things with it, long before IoT became the ubiquitous term it is today. The team focused on very fast-paced, short, proof-of-concept projects, and were afforded time to learn new skills, innovate and work on side projects.

Nick's background working in the MQTT protocol space before it was known outside of IBM led him to a side project: He wanted some way to visualize mapping messages on an MQTT infrastructure to see how they come in on one topic and get sent out in another.

Starting with web visualization technology and Node.js runtime, relatively new at the time, he spent a day or two putting together a little demo of an application that would connect to an MQTT broker that had an API. Asking what mappings it had, the application drew a visualization in the browser which became a very early version of the topic explorer.

Nick wanted to make it more interactive. Not having the terms nailed down yet, he wanted to drag a blob onto the screen, draw a line, configure it, and then hit a button to apply the transformation. Just 24 hours later, he had a simple browser-based application that could define and apply mappings between MQTT topics.

It very quickly became useful.

Dave Conway Jones, Nick's colleague, took the code to the next step, adapting it to add a serial node to work with the project data he was collecting. Over the next few weeks, as part of the team's work on proof of concepts for customer engagements, they hard-coded different nodes into the palette and the utility of this new application was clear.

With the blessing of IBM management to spend more time on the project, Nick spent a few days redesigning the code to make it easier to write in new nodes, unlocking the ability to quickly add in the function node, change node, and switch node, which became the basic building blocks of the tool.

It became Node-RED when the application was submitted as an idea to map web services visually as part of IBM's new public cloud offering, and it started gathering more traction within the company.
To get it into the hands of a wider audience, IBM supported the decision to make Node-RED open source, and it was published on npm and GitHub. Internally, Node-RED had a one-click deployment to IBM Cloud and was used by its developers to demo the IBM services available on the cloud.

In late 2013, Nick O'Leary demoed Node-RED at a London IoT meetup and word spread among his peers in that community. A week later, at an open source hardware conference, Nick entered a workshop and was shocked to see Node-RED on everyone's screens. The facilitator had seen Node-RED and reworked his workshop so that people didn't have to worry about writing lines of code and were able to do useful things much more quickly.

Word was also spreading among the companies using IBM Cloud. One voiced concerns that Node-RED might have been an IBM-specific technology and suggested that we consider moving it to an open source foundation. This led to Node-RED becoming one of the founding projects in the relaunch of the Node.js Foundation. Having independent governance allowed companies like Hitachi to contribute to the project and have an equal voice in its development.

For software developers, time spent writing boilerplate code is not time adding value to the application they're building. With low-code, Node-RED abstracts all that boilerplate so they can focus on the business problem.

Device manufacturers paid attention when Node-RED was installed on the Raspberry Pi image, with its low-code accessibility attracting a broad range of people from systems engineers building automation to IoT hobbyists.

Now with millions of deployments, Node-RED continues to collect, transform, and integrate data through visualized dashboards. And, as this open source community grows it remains rooted in the two pillars of extensibility and a low-code, with an ever-expanding flow library to empower users to collaborate and build their projects.

Chapter 2

Getting Started with Node-RED

Node-RED can run on most modern computer systems including your local computer running Windows, MAC or Linux, devices or in the cloud. The Node-RED website has instructions on how to install it to run:

- Locally
- On Raspberry Pi
- Using Docker
- From the source on GIT
- BeagleBone Boards
- Android
- AWS
- Microsoft Azure

However, the easiest way to get started is to get Node-RED running on FlowFuse Cloud. When you sign up as a new user you are enrolled in a free trial and a Node-RED instance will be started for you within a minute. Once that instance has booted up you access Node-RED by pressing "Open Editor".

Creating your First Flow

A Flow is represented as a tab within the editor workspace and is the main way to organise nodes. The term “flow” is also used to describe a single set of connected nodes informally. So a flow (tab) can contain multiple flows (sets of connected nodes). A Node is the basic building block of a flow. Nodes are triggered by either receiving a message from the previous node in a flow or by waiting for some external event, such as an incoming HTTP request, a timer or a GPIO hardware change. They process that message, or event, and then may send a message to the next nodes in the flow. A node can have at most one input port and as many output ports as it requires.

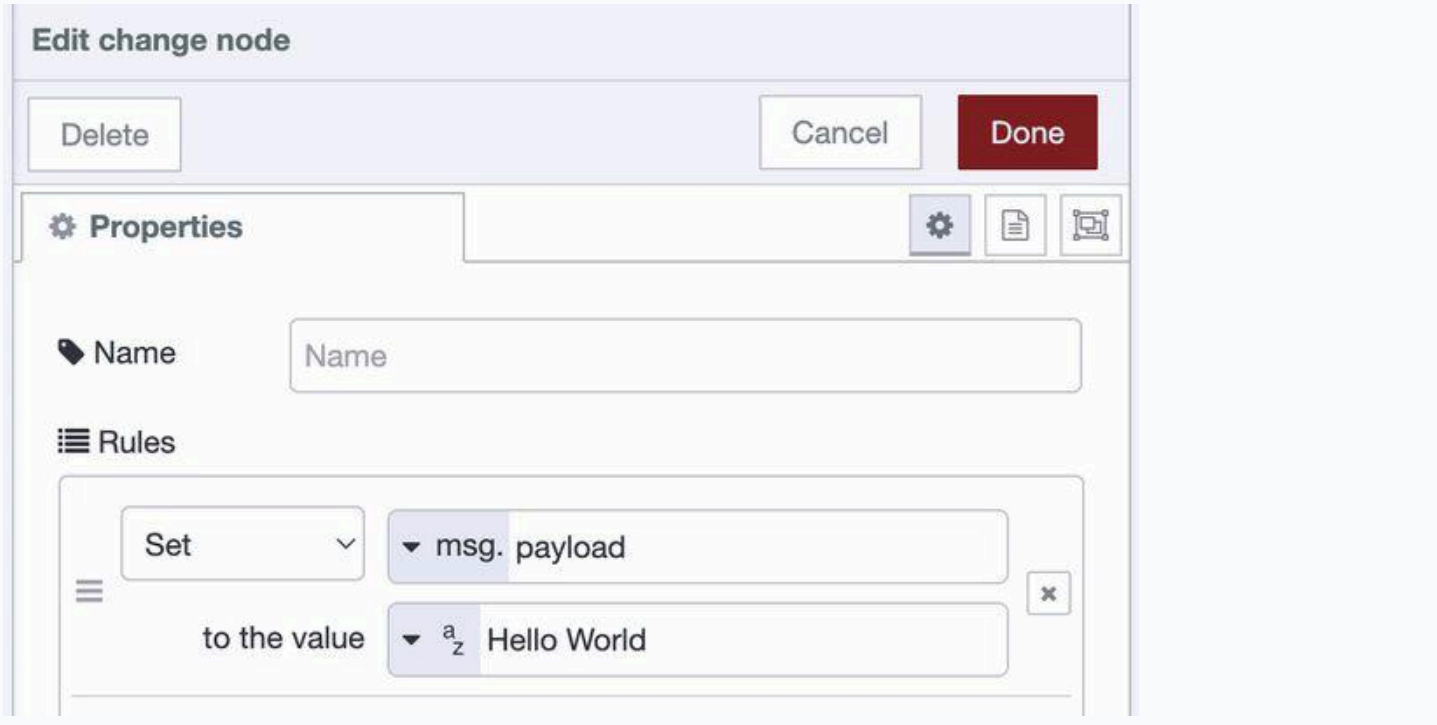
Let's take a look at how to create a simple flow.

In this example, we'll create a simple "Hello World" endpoint. To do this, we'll use the http in, http response, and the change nodes, which can be found in the common nodes menu on the left of Node-RED.

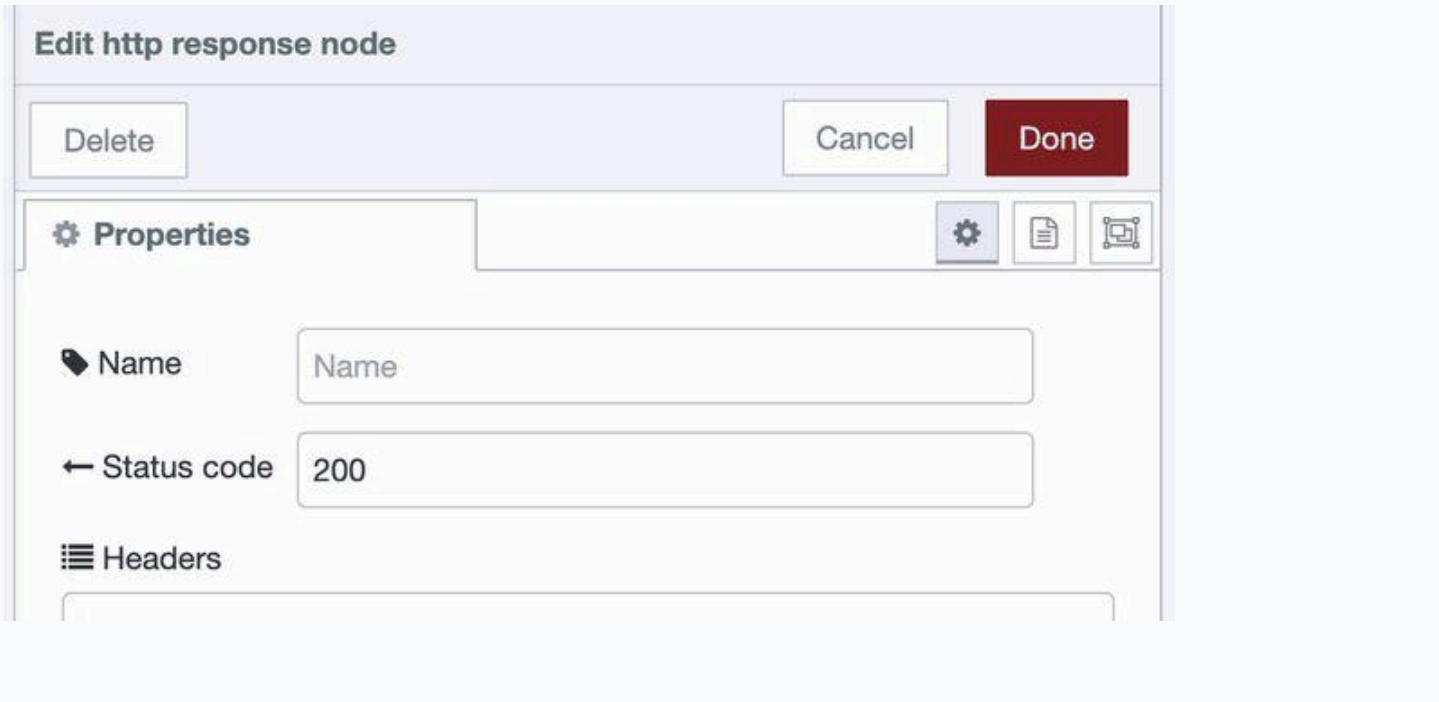
First, drag an http in node into the editor. This node will listen for incoming HTTP requests. Next drag in the "change" and the http response node into the editor. Connect the http in node to the change node and connect the change node to the http response node. Hopefully, your flow looks similar to this:

To configure the http in node, double-click on it to open its properties. Here, you can set the URL that the node will listen to, as well as the method (GET, POST, etc.). In this example, we'll set the URL to /hello and the method to GET.

Now we need to set what the endpoint will respond with, we will do that in the change node. Double-click the change node then add "Hello World" to the field which says "to the value". It should look like this:



To configure the http response node, double-click on it to open its properties. Here, you should set the "Status Code" to be 200. This is not vital for the demo to work but it's good practice to return the correct codes when something connects to an API. Status code 200 means the API responded OK. This is how your http response node should look:



You can read more about HTTP response codes in this article.

Testing Your Flow

Now that we have our flow set up, we can deploy it by clicking the "Deploy" button in the top right corner of the editor. Once the flow is deployed, you can test it by opening up a web browser, if you installed Node-RED using npm navigate to "http://localhost:1880/hello". If you are working on FlowFuse take the URL of your project and add "/hello" to the end, it should looks something like this "https://your-instance-name.flowfuse.cloud/hello". You should see "Hello World!" displayed in the browser.

Debug Output

One of the most powerful features in Node-RED is the ability to debug your flow, this can be done by adding a debug node to your flow and connecting it to the nodes you want to debug. When a message is sent through the connected node, the debug node will print the message in the debug sidebar in the right side of the editor. This can be very helpful when trying to understand how a flow is working or troubleshoot any issues.

The Palette Manager

In addition to the built-in nodes, Node-RED also has a palette manager feature which allows users to easily install and manage additional nodes from the community. To access the palette manager, go to the menu in the top right corner and select "Manage Palette". Here, you can search for and install new nodes, as well as update or remove existing ones. This is a great way to extend the functionality of Node-RED and add new capabilities to your flows.

Import the flow

If you want to view this flow you can import it using the code below. Copy the code then select Import from the top right menu in Node-RED. Paste the code into the field then press Import.

Unset

```
[
 {
  "id": "a742e7a95697bb40",
  "type": "http in",
  "z": "9e9af3caa4dc14d3",
  "name": "",
  "url": "/hello",
  "method": "get",
  "upload": false,
  "swaggerDoc": "",
  "x": 180,
  "y": 200,
  "wires": [
   [
    "883e7d597f7c7c4b"
   ]
  ]
 },
 {
  "id": "aca024dcb79bdb92",
  "type": "http response",
  "z": "9e9af3caa4dc14d3",
  "name": "",
  "statusCode": "200",
  "headers": {},
  "x": 500,
  "y": 200,
  "wires": []
```

```
 },
 {
   "id": "883e7d597f7c7c4b",
   "type": "change",
   "z": "9e9af3caa4dc14d3",
   "name": "",
   "rules": [
     {
       "t": "set",
       "p": "payload",
       "pt": "msg",
       "to": "Hello World",
       "tot": "str"
     }
   ],
   "action": "",
   "property": "",
   "from": "",
   "to": "",
   "reg": false,
   "x": 340,
   "y": 200,
   "wires": [
     [
       "aca024dcb79bdb92"
     ]
   ]
 }
]
```

Now that we've covered the basics, you should be able to create flows for your own use cases. You can check out the Node-RED Library to find new nodes, share your flows and see what others have done with Node-RED.

In addition, FlowFuse offers a curated collection of certified nodes, ensuring top-notch quality, security, and support. Explore FlowFuse Certified Nodes and benefit from a robust ecosystem while maintaining operational reliability.

Chapter 3

Node-RED Core Nodes

In Node-RED, the core nodes are the set of nodes that are included with the Node-RED runtime by default without the node install procedure. These nodes are maintained and supported by the Node-RED development team and are intended to provide the basic building blocks for creating Node-RED flows.

Core nodes include nodes for basic functionality like input/output, processing, and control flow. They are the foundation upon which more complex workflows can be built, and they are essential to the operation of Node-RED.

Following are some of the more widely used core nodes.

Inject

The Inject node is the beginning of many flows that are triggered manually. The box to the left of the node sends a message to connected nodes. For that reason it's often used for debugging too, to inject values at a point of choosing. The message item can be empty too. The message to send defaults to the timestamp as payload and an empty topic.

Message properties can be set to flow, or global variable values, and many other types, including JSONata expressions.

Flows can also be triggered once right after Node-RED starts the flows or with a delay. This is useful to set an initial state from a flow on boot.
An inject node can also start a flow based on a schedule. The schedules have capabilities mimicking cron. In the bottom section of the properties pane the repeat section one can select "at a specific time".

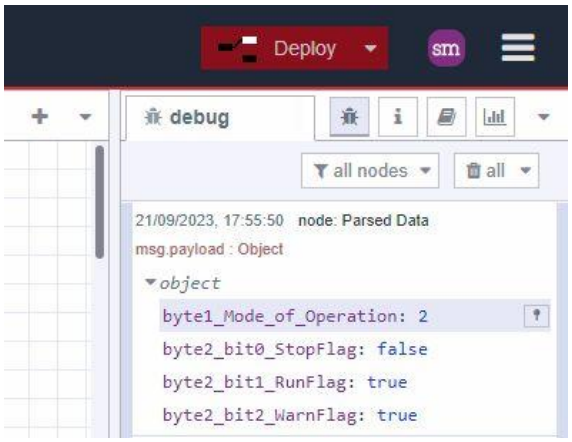
Repeating the measure on an interval is done by selecting "interval" in the "repeat" section. On a schedule requires an input higher than 1 and below 2^31. When the repeat value is 0 or below Node-RED will not display an error.

Examples of the Inject Node

Debug

The Debug node is used to understand the message and data traveling through your flows. During development, it is highly recommended to add Debug nodes at key points in your flow so that you have visibility and understanding of what is being passed around.

In the Node-RED editor, debug messages can be viewed in the right-hand sidebar panel, under the debug icon.



Examples of the Debug Node

Change

The Change node in Node-RED is used for modifying the content of messages within a flow. It allows you to add, remove, modify, or set message properties and payload values, making it a fundamental node for data transformation and manipulation. The Change node is essential for preparing data for further processing, formatting messages for specific outputs, and adapting data to suit the requirements of downstream nodes in a flow.

Examples of the Change Node

Spilt

Granular data processing is important in IoT use cases as multiple tags, for example, might be sent in one request to a server. Or, when a SQL query results in many results that need individual processing. In Node-RED, flows are message-based, but a message can be split in multiple messages if needed using the Split node. The split node is one of the core nodes in Node-RED, thus installed by default. It's a fundamental building block for powerful automation.

The Split node is used to divide a single message into multiple messages based on defined rules.
The Split node in Node-RED is used to split an incoming message object into several different message objects. The incoming object can be a simple string, an array, or an object. The Split node will split the object based on the following criteria:

- **String:** The Split node will split the string on a delimiter character. The delimiter character can be specified in the node's configuration.
- **Array:** The Split node will split the array into a series of messages, each containing one element of the array.
- **Object:** The Split node will split the object on the keys of the object. The keys of the object can be specified in the node's configuration.

The Split node can be used to process data in a variety of ways. For example, it can be used to split a string of text into a series of messages, each containing one word of the text. It can also be used to split an array of data into a series of messages, each containing one element of the array.

The following are some examples of how the Split node can be used:

1. Splitting a string of text into a series of messages, each containing one word of the text.
2. Splitting an array of data into a series of messages, each containing one element of the array.
3. Splitting an object on the keys of the object, to create a series of messages, each containing one key-value pair of the object.
4. Splitting a message on a delimiter character, to create a series of messages, each containing one part of the message.

The Split node is a powerful tool that can be used to process data in a variety of ways. It is a valuable addition to any Node-RED flow.

Examples of the Split Node

Delay

The Delay node allows you to introduce a delay in the flow of messages between nodes. It can be useful in various scenarios where you need to control the timing of message processing. For example, the delay node can limit the rate at which messages are processed downstream or throttle the flow of messaging. Both can be useful for interacting with external systems that may have limitations in place.

Here are some other use cases for using the Delay node:

Batch Processing: If you're dealing with a stream of incoming data that you want to process in batches, you can use the Delay node to introduce a delay between groups of messages. This can be helpful when you need to aggregate or analyze data in chunks.

Sequential Processing: Sometimes you need to ensure that messages are processed in a specific order. The Delay node can be used to enforce a sequence of message processing, especially when dealing with asynchronous systems that might not guarantee order.

Simulation and Testing: In testing and simulation scenarios, you might want to mimic real-world timing conditions. The Delay node can help you introduce delays that simulate actual conditions, allowing you to test how your system behaves over time.

Time-based Triggers: You can use the Delay node to trigger actions at specific time intervals. For instance, you might want to send a status update every hour or perform a cleanup task at the end of the day.

Circuit Breaker: The Delay node can be employed as a simple form of circuit breaker. If a downstream system is failing or experiencing issues, you can introduce a delay before retrying, giving the system some time to recover.

Examples of the Delay Node

Exec

Node-RED is written in Javascript, as are the custom nodes in the Flows Library. If you prefer to use programs written in other languages, Node-RED by default comes with the Exec node. This node allows you to run a command as if you're on the command line. The exec node has one input, and three outputs.

Examples of the Exec Node

Link

The Link In and Link Out nodes are used to help Node-RED developers to organize their flows to make them easier to understand. The Link nodes can be used to connect two sets of Nodes but the connection is only shown when one of the nodes is selected. This allows developers to group nodes that complete a specific function. The Link node will connect two groups but visually it isn't apparent until one of the nodes is selected.

Examples of the Link Node

Switch

The Switch node allows you to route messages based on certain conditions. It acts as a decision-making tool within your flow, allowing you to define rules for directing messages to different output branches.

Here are some common use cases for using the Switch node in Node-RED:

Message Filtering: You can use the Switch node to filter messages based on specific criteria. For example, you might want to filter out messages that don't meet a certain threshold or that don't contain certain keywords.

Conditional Routing: The Switch node enables you to route messages down different paths in your flow based on conditions. You can set up rules that determine which output branch a message should be sent to, depending on its content or properties.

Event Processing: If you're working with events or data streams, the Switch node can help you process different types of events differently. For instance, you might have events related to temperature and humidity readings, and you want to process them separately.

Value Conversion: In cases where you need to convert values from one format to another, the Switch node can route messages to different converters based on the incoming value's properties.

Error Handling: When working with data or APIs, you might receive error messages that need to be handled differently from regular data. The Switch node can direct error messages to a separate branch for appropriate handling.

Language or Region-Based Processing: In applications involving localization or multilingual support, the Switch node can route messages based on language or region information in the message.

Examples of the Switch Node

Function

Function nodes are an essential part of Node-RED. They allow you to write custom JavaScript functions that can be used in your Node-RED flows. The messages are passed in as a JavaScript object called msg. By convention, it will have a msg.payload property containing the body of the message.

The function is expected to return a message object (or multiple message objects) but can choose to return nothing to halt a flow. The On Start tab contains code that will be run whenever the node is started. The On Stop tab contains code that will be run when the node is stopped.

If the On Start code returns a Promise object, the node will not start handling messages until the promise is resolved.

Read more about the benefits and drawbacks of using Function nodes.

Comment

Maintaining flows over longer periods can save you time and interpretation errors if you add the comment node to your flows.

The 4 core benefits of adding the Comment node to your Node-RED flows are:

1. Improved readability: Comments can help to make flows easier to understand, especially when adding easy to miss context and explanation like implicit requirements of the flow. This can be especially helpful for complex flows with link nodes, or when you're editing in many tabs.
2. Enhanced maintainability: Comments can help to make flows easier to maintain by providing a record of the flow's purpose and functionality. This can be helpful when making changes to the flow or when troubleshooting problems, particularly if you collaborate with multiple team members.
3. Improved debugging: Comments can help to make debugging easier by providing information about the expected behavior. This can be helpful when tracking down errors and identifying the source of problems, or reasoning errors when previously developing the flow.
4. Increased documentation: Comments can be used to document the code, providing additional information about the code's purpose, functionality, and usage. Beyond what the flow or nodes do in the sequence, business logic and requirements documented next to the flow increased developer efficiency.

The comment node can be added to any open space in the editor, and it's advised to add a comment to all flow groups.

As the comment node will not take up more space if you write a larger comment, Node-RED allows you to be more explicit and elaborate further in your comments. ASCII diagrams or so do not distract from your flow, and as such you're encouraged to add these.

Examples of the Comment Node

Chapter 4

Formatting your Node-RED Flows

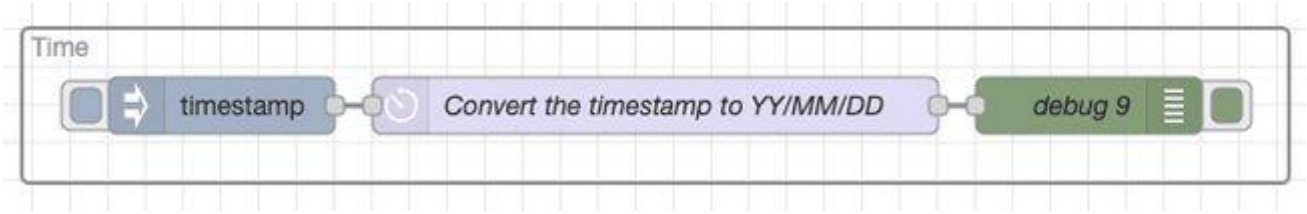
When it comes to working on Node-RED flows as part of a team, there are a few best practices that can make things go more smoothly.

From formatting your flows for readability to providing clear comments on nodes and groups, a little bit of effort upfront can save you and your team a lot of headaches down the road.

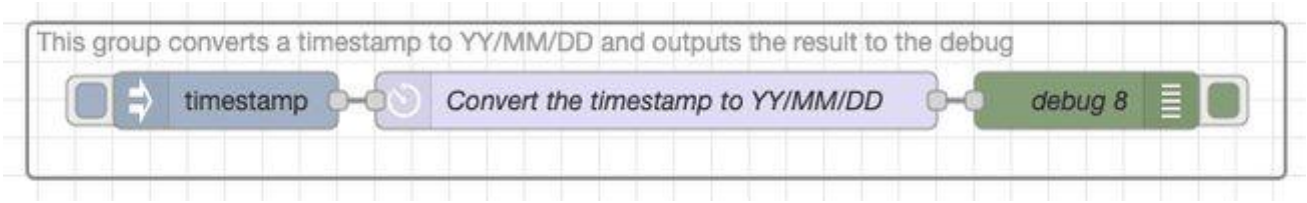
Give your groups descriptive names

Let's start with grouping your flows and giving each group a clear explanation of what it does. Compare the first to the second example below and consider how much more quickly you can understand what the flow is doing.

This is not helpful, 'Time' doesn't tell you enough to understand the flow's purpose.



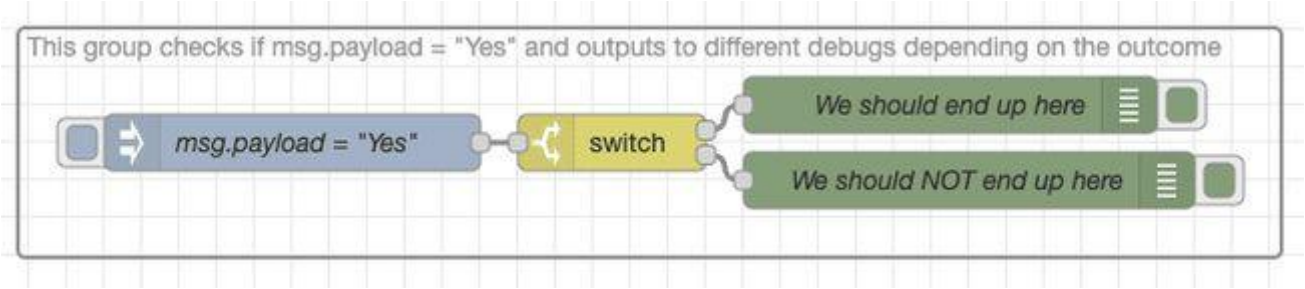
This is much better, we know what the flow is doing without inspecting the nodes.



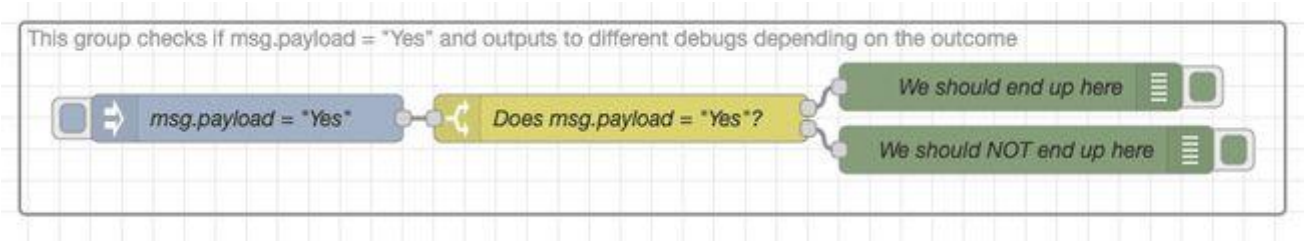
Explain what your switches do

Try to make it obvious what each switch does without having to open the node editor. Ask a question in the switch's name and make a positive answer.

This is not easy to understand, what does the switch do?



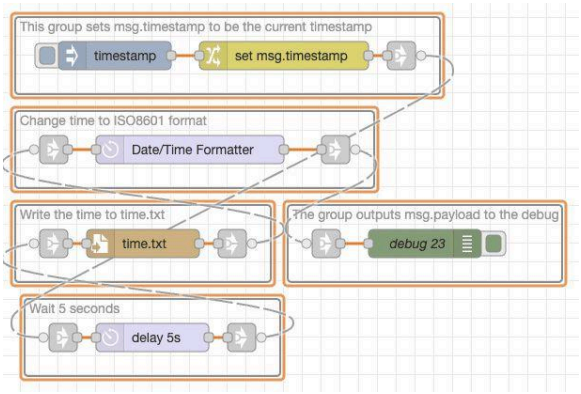
This is a lot better, we can see that the top debug should be triggered.



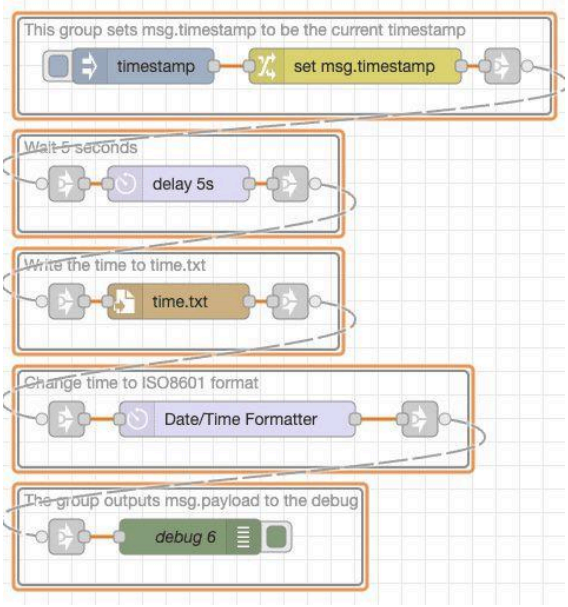
Where possible, your flows should work down the canvas

It makes it so much easier to understand what happens and in which order if your flows start at the top of the canvas and work down to the bottom.

This is almost unreadable, it's very hard to work out the order of the groups.



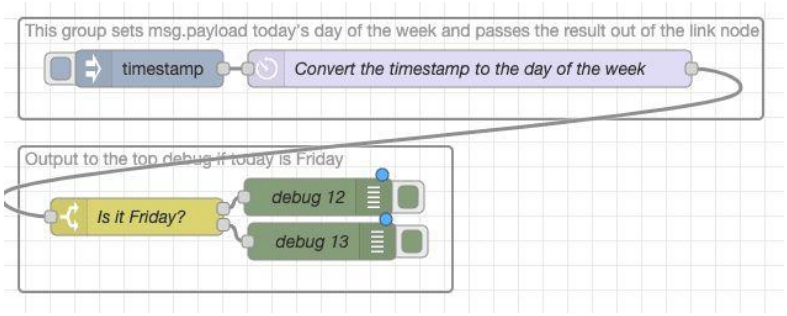
Whereas this is so much easier to understand.



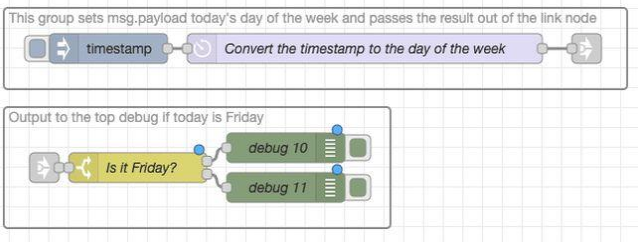
Use link nodes rather than wires to join groups

Groups should not be joined using wires, it just looks untidy and quickly reduces the readability of your flows.

The wire is blocking the title, it only gets worse as you add more wires.



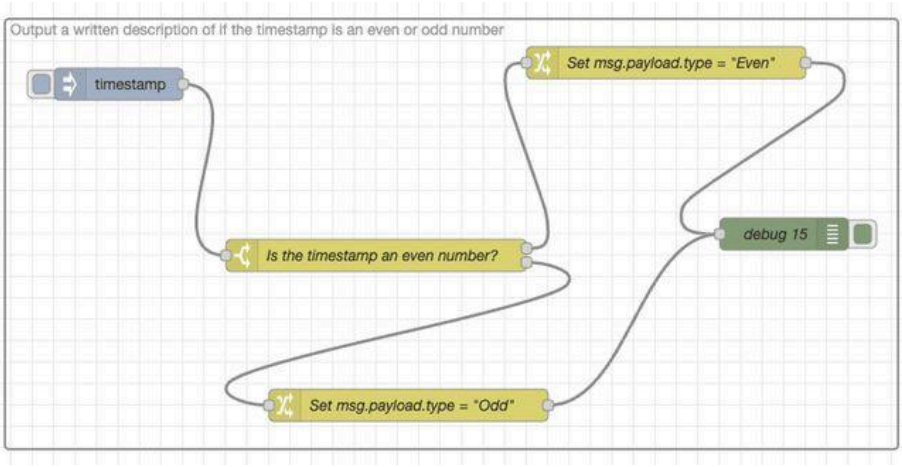
You can see the group titles easily now.



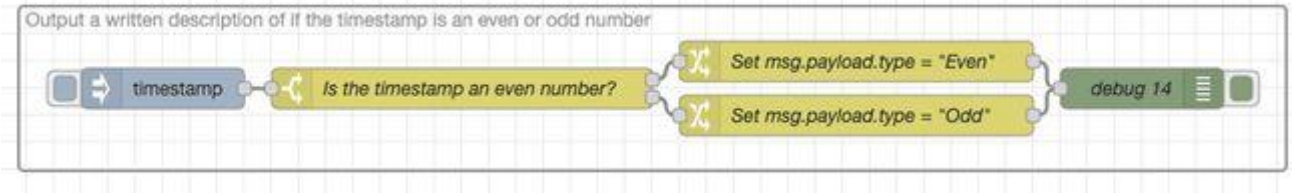
Keep your groups compact

Keeping your groups compact will save time when reading the flow. This is especially helpful when viewed on a smaller screen.

Consider how hard a flow made of groups spaced out like this would be to read on a smaller laptop screen.



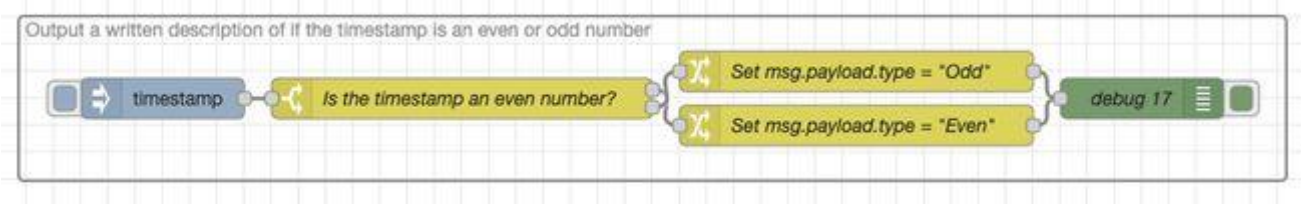
This now takes up less space and is arguably easier to read on any screen size.



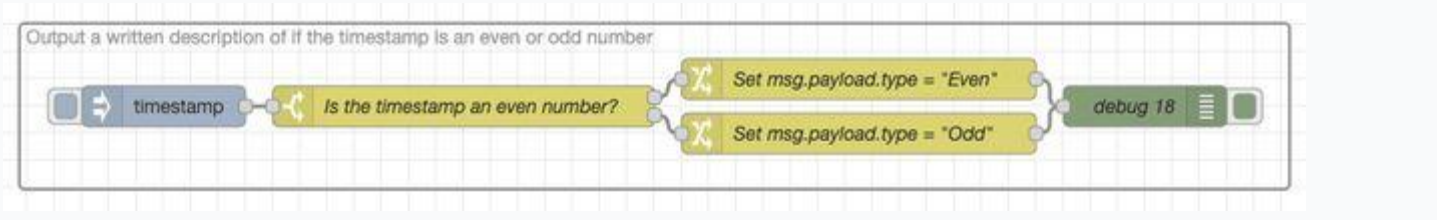
Don't cross ~~beams~~ wires

Crossed wires are not only hard to read, they can lead to misinterpretation of what a flow actually does. Where possible don't cross your wires, where you can't avoid it try to make sure it's easy for the reader to understand where wires cross as rather than join.

This is confusing, which change node does the top switch output link to?



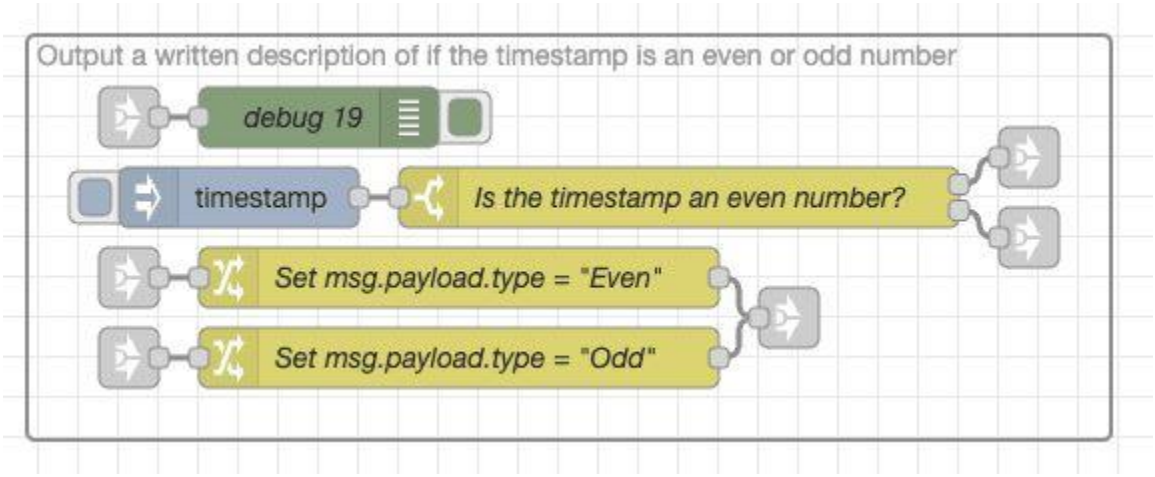
This is better, much less chance of confusing the change nodes.



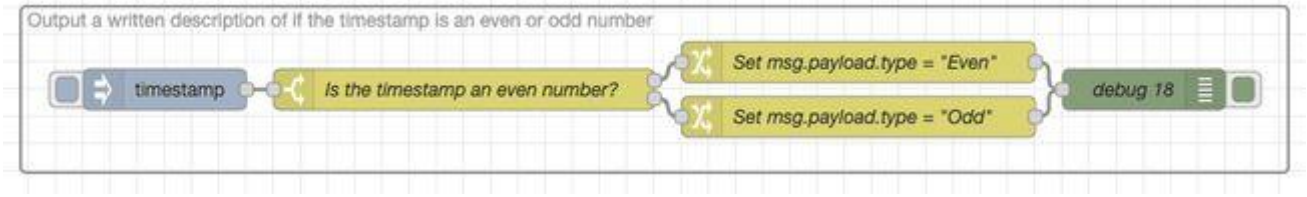
Don't use link nodes in groups where avoidable

Excessive link nodes within groups can make a flow much harder to understand, where possible use wires to join nodes within a group.

This is hard to read and you will end up checking the link nodes again and again.



Functionally identical to the example above, it should only take a few seconds to understand this flow now.



Chapter 5

Capturing Data from Edge Devices

While cloud computing has revolutionized data access and analysis, not all data can be accessed from the cloud. In many scenarios, data collection from the edge – the location where data is generated – is essential for real-time decision-making or process observability.
FlowFuse enables data to be collected through Node-RED. Data can be processed locally on the edge or sent on to other services. FlowFuse doesn't rely on continuous connections to the cloud, making it a good choice for locations with unreliable internet connectivity. Use cases like real-time monitoring of critical systems, proactive maintenance, and improved operational efficiency are now possible to implement.

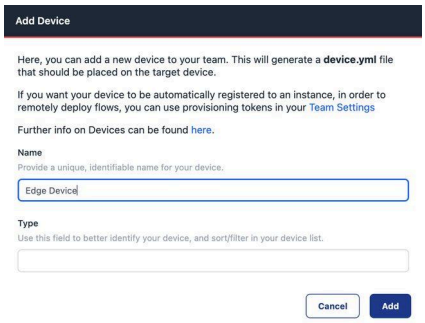
Installing the FlowFuse agent

To manage the capturing of data on the edge you need to first install the FlowFuse agent. It's installed on your device to manage the communication between the edge device and the FlowFuse server, manage the installation of Node-RED, and its execution environment, and facilitate communication between devices and the cloud.

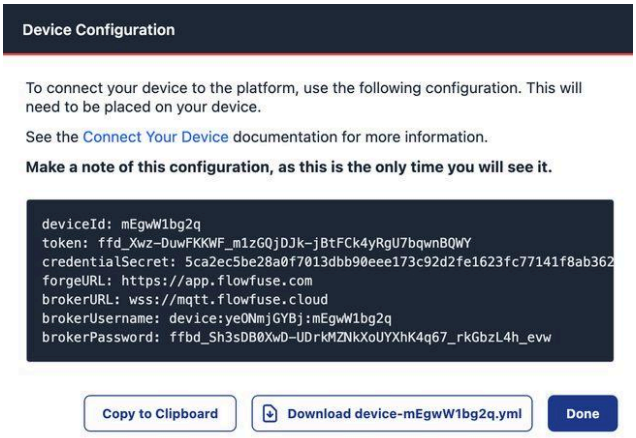
The device agent can run anywhere you can run a Docker container or Node.JS runtime (version 16.0+) can be installed.

Registering a device on FlowFuse

For the edge device to know what it's supposed to do, it needs to listen to the FlowFuse commands. The agent's configuration is provided by a device.yml file from FlowFuse. Go to the team you'd like to add an edge device to, and select "Devices" on the left-hand menu, followed by the "Add Device" button.



FlowFuse will prompt you to add a name (required), and a type (not required). When you've clicked Add you'll get a new dialog to download the required file.



Install the FlowFuse agent through Docker

If your device supports it, the fastest way to run the FlowFuse agent is with containers. Assuming you've already got Docker installed, there are two steps to follow: first, move the device YAML file downloaded from FlowFuse to the edge device and save it in /opt/flowfuse/device.yml. Start the agent by running:

```
Unset
docker run --mount type=bind,src=/path/to/device.yml,target=/opt/flowfuse-device/device.yml -p 1880:1880 flowfuse/device-agent:latest
```

Note that for production cases, ensure the container is restarted on reboot. Docker can do this for you, please follow their guide.

Install the FlowFuse agent with npm

To install the agent through NPM, you'll need a Node.JS version of 18.0 or later. Open a command prompt and run: npm install -g @flowfuse/device-agent.

This will install the FlowFuse Device Agent as a global npm module, making the flowfuse-device-agent command available in any directory on your system.

Once the installation is complete, you must configure the Device Agent to connect to your FlowFuse instance. In this guide, you've previously downloaded the device.yml file that's needed now.

On Linux or Mac, move the file to /opt/flowfuse-device/device.yml, and for Windows-based systems, move the file to c:\opt\flowfuse-device\device.yml.

Afterward, start the agent with: flowfuse-device-agent.

This will launch the Device Agent and connect it to your FlowFuse instance. The Device Agent will wait for instructions on which flows to run.

Programming flows for the edge

Now the agent is running, the FlowFuse platform will show it has contacted back to the platform and is ready to do some work. First, add it to the application and start the developer mode. That enables the device editor and provides you secure access to the editor anywhere in the world for everyone in the FlowFuse team with the right access role.

When the development is done, be sure to create a snapshot of the developed flows to create a point-in-time backup, or to roll the snapshot out to many other devices later.

Chapter 6

Data Visualization with Dashboards

The Node-RED Dashboard is a vital tool for creating live dashboards and user interfaces for Node-RED flows. The original version however was built on the no-longer-maintained Angular v1. This outdated foundation presented potential security issues that cannot be addressed with patches.
To counter this, FlowFuse launched a new version of the Dashboard. Dashboard 2.0 will maintain the core principles of open-source, community-driven development under the Apache 2.0 license, and is designed to safely usher the Node-RED community into the future of data visualization.

Dashboard 2.0 represents a comprehensive reconstruction of the original framework, now based on VueJS. The revamped version incorporates complete responsiveness, extending from desktop to mobile devices. Quality of life improvements have been implemented across the existing widget collection, with several new additions to enhance user experience.

Notable features include Dynamic Markdown, Tables & Notebooks, UI Chart improvements, and a custom video player. In addition, it introduces a groundbreaking feature – Personalized Multi-User Dashboards, exclusively available on Node-RED Dashboard 2.0 when running on FlowFuse Cloud. This new feature allows users to build dashboards that provide unique data to each user, build admin-only views, and track user activity.

Installation

FlowFuse Node-RED Dashboard 2.0 is available in the Node-RED Palette Manager. To install it:

- Open the menu in the top-right of Node-RED
- Click "Manage Palette"
- Switch to the "Install" tab
- Search node-red-dashboard
- Install the @flowfuse/node-red-dashboard package (not node-red/node-red-dashboard)

The nodes will then be available in your editor for you to get started.
If you want to use npm to install your nodes, you can instead follow these instructions.

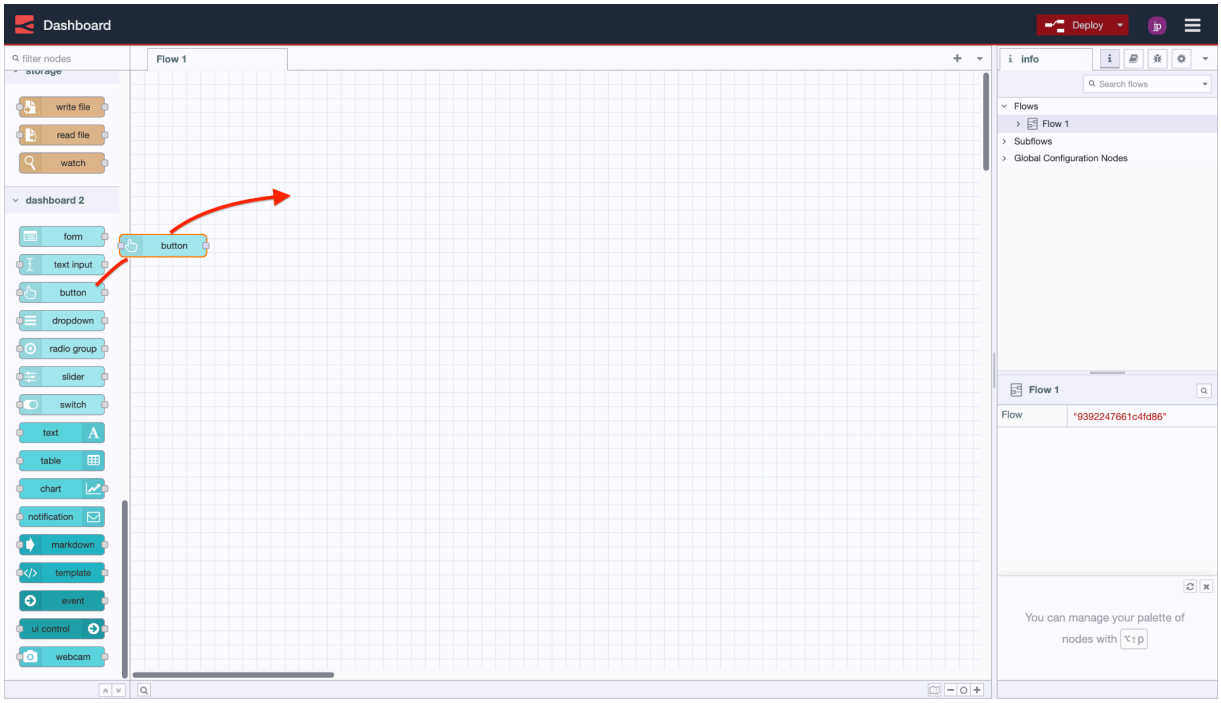
Dashboard Hierarchy

Each Dashboard is a collection of widgets (e.g. charts, buttons, forms) that can be configured and arranged in our own User Interface. The hierarchy of a Dashboard is as follows:

- **Base:** Defines the base URL (e.g. /dashboard) for your Dashboard.
- **Page:** A given page that a visitor can navigate to, URL will extend the base, e.g. /dashboard/page1. Each page can also have a defined, unique, Theme which controls the styling of all groups/widgets on the page.
- **Group:** - A collection of widgets. Rendered onto a page.
- **Widget:** - A single widget (e.g. chart, button, form) created in Dashboard.

Building a Dashboard

To get started, drop any widget from the flows on the left-side of Node-RED onto the editor.



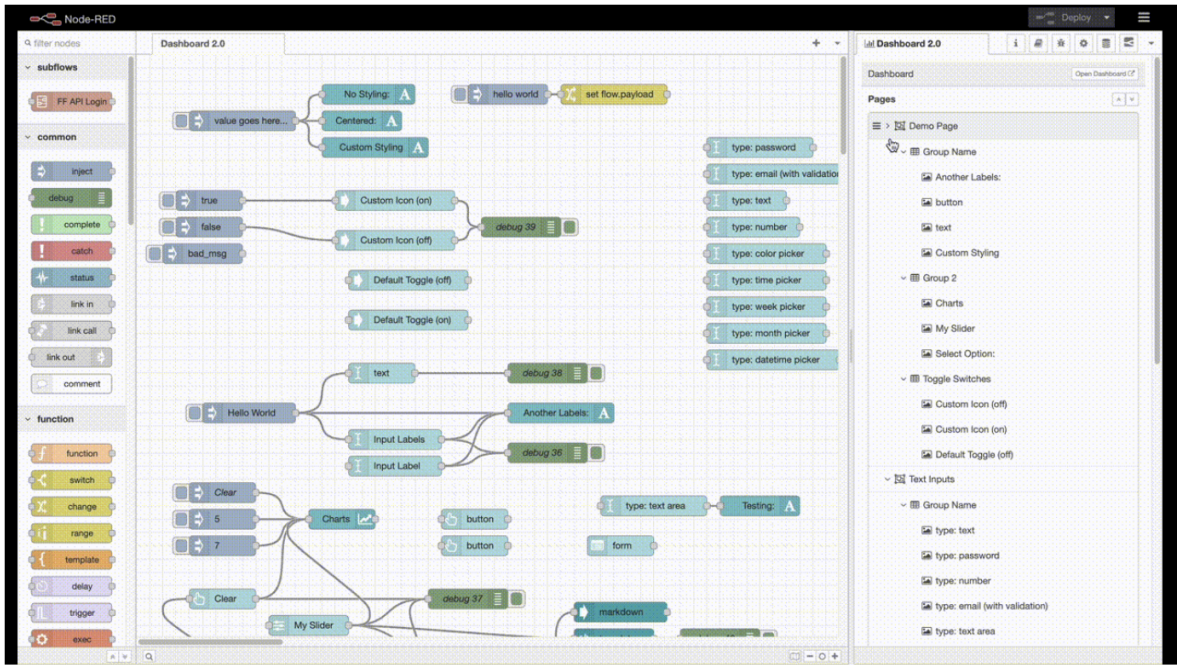
Initial Manual Step

For the first widget you add, you will need to double click your new widget, and create your first group, page, theme and base. Once you have created these, you can add more widgets to the same group freely, and add any more groups/pages as you need to.

Note: we do have a plan to automate this initial manual step of creating group/page/base.

Managing your Dashboard

Once you have created a few widgets, groups and/or pages, you may want to make use of the Dashboard 2.0 sidebar. Sidebars are available on the right-hand side of the Node-RED interface. The down chevron provides access to more if you don't see Dashboard 2.0 in the initial tabs.



The sidebar offers you a central place to manage your pages, groups and widgets. You can control the order that each of these appears in their respective parents, and even move widgets between groups, and groups between pages.

For more details on Dashboard 2.0, FlowFuse has published rich documentation that covers all available nodes, insights into the framework's architecture, and guidelines for contributing to the project. You can also watch the on-demand webinar.

Chapter 7

Securing Node-RED

One thing that can get overlooked in many projects is the security of Node-RED. Let's look at some easy ways to make your Node-RED project more secure, even when first learning about it in a hobby environment.

While we will focus on securely hosting Node-RED on a Pi on your own LAN, note that if you use FlowFuse Cloud to host Node-RED, solutions discussed below are either ready out of the box or are not needed.

By default, the editor for Node-RED is protected using your FlowFuse user credentials. You can also use SSO to further protect your user accounts and give access to Node-RED to your team members.
All traffic to FlowFuse and your Node-RED instances is protected by HTTPS. FlowFuse has set up the domain name and manages the certificates so you can spend time on your flows rather than configuring security.

FlowFuse has a free trial if you'd like to see how we've made secure hosting of Node-RED easy.

Protecting access from your LAN to Node-RED

Once you have an instance of Node-RED running it can usually be accessed from anywhere on your LAN (local area network) by pointing a web browser to the relevant IP address and port.
http://192.168.0.3:1880

With a URL similar to the one above, depending on your specific network and Node-RED configuration, anyone on your LAN can view but more importantly, edit your flows. This can be useful when first learning about Node-RED but it’s always a good idea to get into the habit of locking down access to the editor, even if you trust everyone who can access your LAN.

One of the easiest ways to protect your flows is to add a username and password to your Node-RED instance.

The first step is to find your Node-RED settings.js file. It's not always in the same place but on a default Debian Linux installation, it can be found in this directory.
cd ~/.node-red

If you list that directory you should now see something like this:

```
pi@raspberrypi:~ $ cd ~/.node-red
pi@raspberrypi:~/.node-red $ ls
lib  node_modules  package.json  package-lock.json  settings.js
pi@raspberrypi:~/.node-red $
```

We now need to edit settings.js. You can use any such as text editor, Nano to do that.
nano settings.js

We now need to find and edit the following section of the settings file:

```
    /** To password protect the Node-RED editor and admin API, the following
     * property can be used. See http://nodered.org/docs/security.html for details.
     */
    //adminAuth: {
    //    type: "credentials",
    //    users: [{
    //        username: "admin",
    //        password: "$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNbM6tl8sJogENOMcxWV9DN.",
    //        permissions: "*"
    //    }]
    //},
```

This example shows how to add a password and uncomment the relevant section of the settings file. You could also change the username for additional security.

To create the password you can use a command line tool which is included in Node-RED. Open a second terminal then run this command:
node-red admin hash-pw

Put in your new password, this example uses password 'flowforge'. The tool returns your password in a hashed format:

```
pi@raspberrypi:~ $ node-red admin hash-pw
Password:
$2b$08$pVd7SPN0Q5jI.VyFiVoZIOg10AWkmbLKFezpBxp.rYnwcN4FLtswe
```

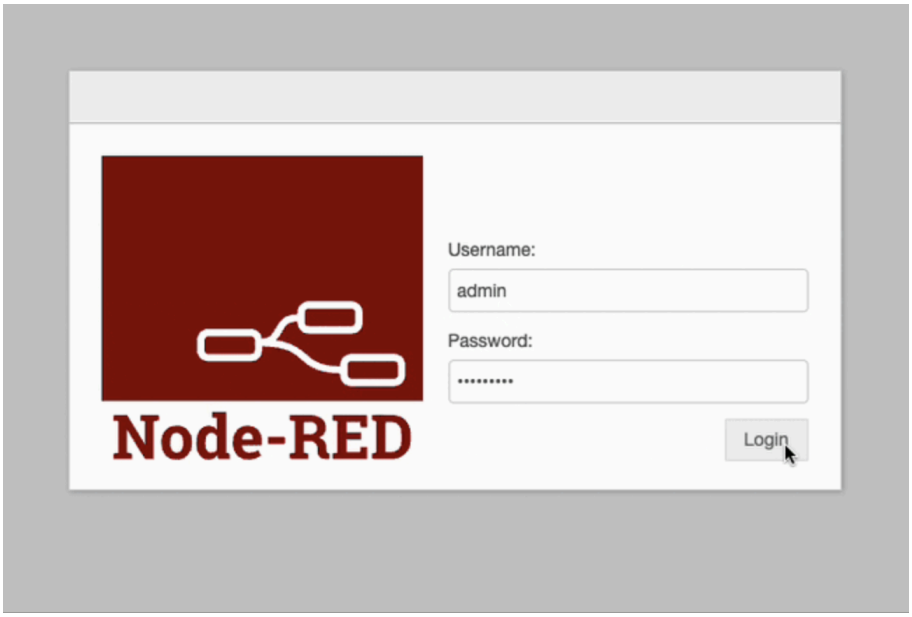
You can now return to the other terminal window, uncomment the section then paste in the new password. This is how it will look:

```
/** To password protect the Node-RED editor and admin API, the following
 * property can be used. See http://nodered.org/docs/security.html for details.
 */
adminAuth: {
    type: "credentials",
    users: [{
        username: "admin",
        password: "$2b$08$pVd7SPN0Q5jI.VyFiVoZIOg10AWkmbLKFezpBxp.rYnwcN4FLtswe",
        permissions: "*"
    }]
},
```

You can now save and exit out of the settings file.

The last step is to restart Node-RED. You can use Debian where the command is:
node-red-restart

Now, when you try to access Node-RED you will need to provide a username and password.



You might also want to consider turning off the editor interface once you are happy with your flows. This can make it a little harder to make changes to your project but it also gives you peace of mind that nobody has accidentally or deliberately changed your flows. You can turn off the editor interface as follows.

Edit your settings.js file as explained above, look for the following section:

```
    /** The following property can be used to disable the editor. The admin API
     * is not affected by this option. To disable both the editor and the admin
     * API, use either the httpRoot or httpAdminRoot properties
     */
    // disableEditor: true,
```

All you need to do is uncomment the bottom line then change the value from false to true, once done it should look something like this:

```
    /** The following property can be used to disable the editor. The admin API
     * is not affected by this option. To disable both the editor and the admin
     * API, use either the httpRoot or httpAdminRoot properties
     */
    disableEditor: true,
```

Now restart Node-RED as covered above, then try accessing your Node-RED instance again. You will no longer be able to edit or view your flows.

Using these two features, we now have much better control over who can access the design interface for Node-RED.

Traffic to your Node-RED instance is unencrypted

It is important to encrypt your connections between devices to stop people from intercepting your traffic. This isn't a huge concern when working on your home LAN but what if you want to access your Node-RED instance from a remote location?

There are two obvious options, HTTPS, and a VPN (Virtual Private Network).

1. You can set up your Node-RED traffic to run over HTTPS, this solution ensures that all traffic to and from your Node-RED is encrypted. The downside to this approach is, it's quite complex to set up. You will need to have a domain name, open up ports on your LAN's firewall, use a HTTPS certificate provider and then make sure you remember to renew the certificates as needed. It's doable if you are comfortable with these concepts.
2. A VPN provides a lot of security advantages depending on which one you are using and how it is configured. It is the easiest option of the two. To secure your traffic you can use a service called Tailscale which is free for personal projects.

Let's install Tailscale on a Raspberry Pi running Node-RED as well as any other device you want to access your project from. Once that's done you can access Node-RED from anywhere with internet access but more importantly, the traffic to and from my devices is also encrypted.

Before you start, it's important to remember that a VPN is only as secure as the company that runs it. You should always consider if you trust the VPN provider as they could potentially access your devices. While Tailscale is good, it is always best to also do your own research before using a VPN provider.

The first step is creating a Tailscale account, you can sign up for free here. Next, you need to add your devices to your VPN using their software. Let's install Tailscale on an Apple laptop, Google phone as well as a Raspberry Pi that runs Node-RED.

The install process is really easy, even on the Pi running Raspbian the steps you need to take are well explained in the Tailscale docs.

For the Pi, these are the commands you need to run.

1. Add Tailscale to the Apt package manager.
   ```
   curl -fsSL https://pkgs.tailscale.com/stable/debian/bullseye.noarmor.gpg | sudo tee /usr/share/keyrings/tailscale-archive-keyring.gpg >/dev/null

   curl -fsSL https://pkgs.tailscale.com/stable/debian/bullseye.tailscale-keyring.list | sudo tee /etc/apt/sources.list.d/tailscale.list

   sudo apt-get update
   ```

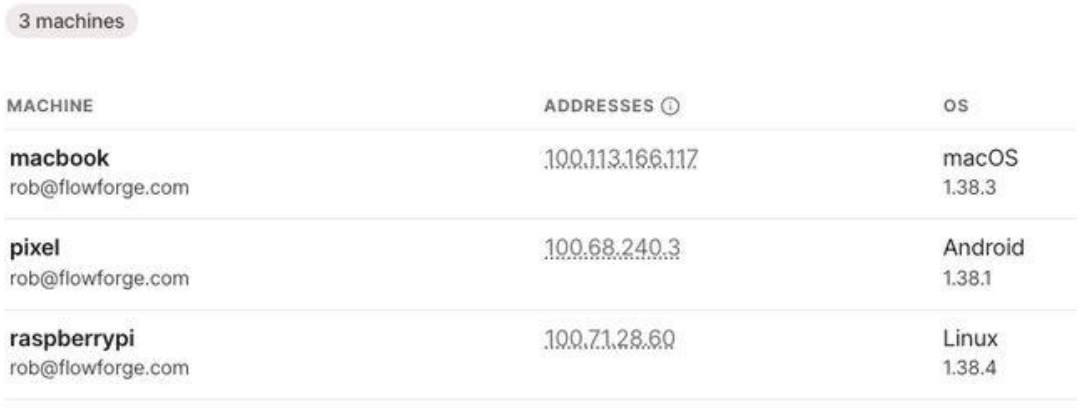
2. Install Tailscale
   ```
   sudo apt-get install tailscale
   ```

3. Start Tailscale and connect your device
   ```
   sudo tailscale up
   ```

After running the last command, you need to follow the on screen prompts to link your devices to your VPN.

One last thing which you might want to consider doing, every few months you will need to reconnect your devices to your VPN, if you are only going to be accessing your Node-RED device over the VPN you should consider disabling your Tailscale key expiry.

Now that we've got our devices all connected you should see something like this in the Tailscale dashboard.



You can now access Node-RED on your Pi from your laptop and phone by pointing a browser to the correct IP address (as shown in the image above) with the port for Node-RED:
`http://100.71.28.60:1880`

You've now secured all traffic between your devices and the Node-RED project, and you can access Node-RED from anywhere on the internet.

If you follow these steps you should be on the right path to running a more secure Node-RED instance. There is a lot more you can do, see relevant docs on the Node-RED website to gain some more ideas.

About FlowFuse

FlowFuse is a platform that empowers any engineer; mechanical, electrical, or software engineer, to build, deploy, manage, and secure enterprise-grade applications. We enable IT experts and operational engineers to collaborate on data acquisition, merging of data streams, create data visualization, and interactive applications.

Our platform enhances the capabilities of Node-RED by providing security and compliance guardrails, increasing developer productivity, and centralized operations for scalability and reliability.

At FlowFuse we are committed to providing the best tools and support to seamlessly integrate into your existing business, offering an adaptable foundation for complex application development.
Contact us to find out how we can help develop innovative applications that revolutionize your operations.