Practical Node-RED Programming

Learn powerful visual programming techniques and best practices for the web and IoT

Taiji Hagino

Foreword by Nick O'Leary, Co-creator of Node-RED

Practical Node-RED Programming

Learn powerful visual programming techniques and best practices for the web and IoT

Taiji Hagino

Packt

BIRMINGHAM—MUMBAI

Practical Node-RED Programming

Associate Group Product Manager: Pavan Ramchandani

Publishing Product Manager: Kaustubh Manglurkar

Senior Editor: Sofi Rogers

Content Development Editor: Rakhi Patel

Technical Editor: Saurabh Kadave

Copy Editor: Safis Editing

Language Support Editor: Safis Editing

Project Coordinator: Divij Kotian

Proofreader: Safis Editing

Indexer: Manju Arasan

Production Designer: Alishon Mendonca

First published: March 2021

Production reference: 1190321

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-80020-159-0

`www.packt.com`

To my colleague, Nick O'Leary, and Node-RED Community co-organizers, Atsushi Kojo, Seigo Tanaka, and Kazuhito Yokoi, I would like to thank you for taking time from your busy schedules to help me with the book. I would also like to thank my wife, Akiko, for being my loving partner and supporting me throughout writing this book and always.

– Taiji Hagino

Foreword

Taiji has been deeply involved with the Node-RED User Group Japan since its creation. In his developer advocate role, he has worked with many users to help them build meaningful applications with Node-RED. This book reflects Taiji's skills and experience with the project and will be a great resource for many readers.

This book will provide you with a good introduction to Node-RED and give you a sense of how quickly you can get started with creating applications. The examples in each chapter will give you a taste of how much can be achieved with very little coding.

I hope it inspires you to continue building with Node-RED and to explore everything that is possible.

Nick O'Leary

Co-creator of Node-RED

Taiji has extensive development knowledge in the web/cloud, mobile, IoT, blockchain, and so on. We have known each other since the inception of the Node-RED User Group Japan 5 years ago.

Taiji has been an active contributor to the Node-RED community since the early days of Node-RED, running Node-RED meetups with us. He was a co-author of the book *First Node-RED* published by the Node-RED User Group Japan 3 years ago.

For more than 5 years, Node-RED has been evolving to meet the needs of developers around the world. During this time, Taiji has been a key member of IBM and has been active in Developer Advocates and Developer Relations.

In addition, Taiji has been able to gain a deep understanding of other languages and cultures through his global activities as a developer advocate and in developer relations.

Taiji has used his knowledge and experience from these global activities to organize the Node-RED Conference Tokyo, a global Node-RED event that has run for two consecutive years, where he has used his global skills to communicate with speakers from overseas and to facilitate the day of the event.

I believe Taiji will continue to serve as a global career model for Japanese developers and will be a key player in the development of the Node-RED community around the world.

Atsushi Kojo

Chief research officer at Uhuru Corporation

Taiji and I have been working together at the Node-RED User Group Japan for 5 years. He is one of the user group organizers. Taiji is especially looking globally with the aim of sharing technological possibilities, such as setting up a meeting between the organizer of a Japanese user group and the Node-RED development team at IBM Hursley. Recently, we held Node-RED Con Tokyo 2019 and 2020 together. Taiji has also carried out an important role as an online moderator and manager.

Taiji has written various blogs where he has shared his immense knowledge of Node-RED smartly. The source of his knowledge comes from his great experience as an excellent developer and developer advocate at IBM. He has gained experience with business use cases and development knowledge such as IoT, mobile applications, cloud technologies, databases, and blockchain in his developer relations activity.

He has a strong understanding of the synergies and difficulties of combining each technology. Many developers find Node-RED attractive because of him. This book represents how knowledgeable he is as a developer.

Read this book and discover how wonderful it is to combine various technologies such as IoT and the cloud using Node-RED, and expand your possibilities as a developer.

Seigo Tanaka

President, 1ft seabass

Contributors

About the author

After becoming a software engineer, **Taiji Hagino** started Accurate System Ltd. with his amazing software development experience. After working as a system integrator of a subsidiary of a general trading company, he now works as a developer advocate in the IBM Global team, developing DevRel (developer relations), a marketing approach to engineers. He also works as a lecturer at the Faculty of Informatics, University of Tsukuba. Works he has authored include *Developer Marketing DevRel Q&A* (Impress R&D), *First Node-RED*, *Practical Node-RED Application Manual* (Kogakusha), and so on. He has been awarded Microsoft MVP and was previously a musician and a hairdresser.

I want to thank all the people who have been close to me and supported me throughout writing this book, especially my wife, Akiko, and my family.

About the reviewers

Nick O'Leary is an open source developer and leads the OpenJS Node-RED project. He spends his time playing with IoT technologies, having worked on projects ranging from smart meter energy monitoring to retrofitting sensors to industrial manufacturing lines with Raspberry Pis and Arduinos. With a background in pervasive messaging, he is a contributor to the Eclipse Paho project and sits on the OASIS MQTT Technical Committee and the OpenJS Cross Project Council.

Kazuhito Yokoi works for OSS Solution Center in Hitachi, Ltd. as a software engineer. On GitHub, he is a member of the Node-RED project. Hitachi has used Node-RED in their IoT platform, Lumada. To improve the code quality and add new features, his team joined the Node-RED project as contributors. For 4 years, 19 contributors in his team have added over 700 commits and 80,000 lines to the project. Currently, they are contributing to not only Node-RED but also sub-projects such as a node generator to generate nodes from various sources without coding, and a Node-RED installer to set up Node-RED without CLI operations. He held sessions about Node-RED at the Open Source Summit Japan 2020, Node+JS Interactive 2018, and other global conferences.

Table of Contents

Section 3: Practical Matters

12
Developing a Chatbot Application Using Slack and IBM Watson

13
Creating and Publishing Your Own Node on the Node-RED Library

Appendix
Node-RED User Community

Other Books You May Enjoy
Index

Preface

Node-RED is a flow-based programming tool that was made by Node.js. This tool is mainly used for connecting IoT devices and software applications. However, it can cover not only IoT but also standard web applications.

Node-RED is expanding as a no-code/low-code programming tool. This book covers the basics of how to use it, including new features that have been released from version 1.2, as well as advanced tutorials.

Who this book is for

This book is best for those who are learning about software programming for the first time with no-code/low-code programming tools. Node-RED is a flow-based programming tool, and this tool can build web applications for any software applications easily, such as IoT data handling, standard web applications, web APIs, and so on. So, this book will help web application developers and IoT engineers.

What this book covers

Chapter 1, *Introducing Node-RED and Flow-Based Programming*, teaches us what Node-RED is. The content also touches on flow-based programming, explaining why Node-RED was developed and what it is used for. Understanding this new tool, Node-RED, is helpful to improve our programming experience.

Chapter 2, *Setting Up the Development Environment*, covers setting up the development environment by installing Node-RED. Node-RED can be installed for any OS Node.js can run, such as Windows, macOS, Rasberry Pi OS, and so on. We install Node-RED on each environment with the command line or using the installer. This chapter covers important notes for specific OSes.

Chapter 3, *Understanding Node-RED Characteristics by Creating Basic Flows*, teaches us about the basic usage of Node-RED. In Node-RED, various functions are used with parts called nodes. In Node-RED, we create an application with a concept called a flow, like a workflow. We will create a sample flow by combining basic nodes.

Chapter 4, *Learning the Major Nodes*, teaches us how to utilize more nodes. We will not only learn about the nodes provided by Node-RED by default but also how to acquire various nodes published on the internet by the community and how to use them.

Chapter 5, *Implementing Node-RED Locally*, teaches us best practices for leveraging Node-RED in our local environment, our desktop environment. Since Node-RED is a tool based on Node.js, it is good at building server-side applications. However, servers aren't just on beyond the network. It is possible to use it more conveniently by using Node-RED in a virtual runtime on the local environment of an edge device such as Raspberry Pi.

Chapter 6, *Implementing Node-RED in the Cloud*, teaches us best practices for leveraging Node-RED on a cloud platform. Since Node-RED is a tool based on Node.js, it is good at building server-side applications. It is possible to use it more conveniently by using Node-RED on any cloud platform, so we will make flows with Node-RED on IBM Cloud as one of the use cases with cloud platforms.

Chapter 7, *Calling a Web API from Node-RED*, teaches us how to utilize the web API from Node-RED. In order to maximize the appeal of web applications, it is essential to link with various web APIs. Its application development architecture is no exception in Node-RED. Understanding the difference between calling a web API from a regular Node.js application and calling it from Node-RED can help us get the most out of Node-RED.

Chapter 8, *Using the Project Feature with Git*, teaches us how to use source code version control tools in Node-RED. With Node-RED, the project function is available in version 1.x and later. The project function can be linked with each source code version control tool based on Git. By versioning the flows with the repository, our development will be accelerated.

Chapter 9, *Creating a ToDo Application with Node-RED*, teaches us how to develop standard web applications with Node-RED. The web application here is a simple ToDo application. The architecture of the entire application is very simple and will help us understand how to develop a web application, including the user interface, using Node-RED.

Chapter 10, *Handling Sensor Data on the Raspberry Pi*, teaches us application development methods for IoT data processing using Node-RED. Node-RED was originally developed to handle IoT data. Therefore, many of the use cases where Node-RED is still used today are IoT data processing. Node-RED passes the data acquired from sensors for each process we want to do and publishes it.

Chapter 11, *Visualize Data by Creating a Server-Side Application in the IBM Cloud*, teaches us about application development methods for IoT data processing using Node-RED on the cloud platform side. We usually use the data from edge devices on any cloud platform for analyzing, visualization, and so on. Node-RED handles the data subscribed from the MQTT broker and visualizes it for any purpose.

Chapter 12, *Developing a Chatbot Application Using Slack and IBM Watson*, teaches us how to create a chatbot application. At first glance, Node-RED and chatbots don't seem to be related, but many chatbot applications use Node-RED behind the scenes. The reason is that Node-RED can perform server-side processing on a data-by-data basis like a workflow. Here, we create a chatbot that runs on Slack, which is used worldwide.

Chapter 13, *Creating and Publishing Your Own Node on the Node-RED Library*, teaches us how to develop nodes ourselves. For many use cases, we can find the node for the processing we need from the Node-RED Library. This is because many nodes are exposed on the internet thanks to the contributions of many developers. Let's aid a large number of other Node-RED users by developing our own node and publishing it to the Node-RED Library.

To get the most out of this book

You will need Node-RED version 1.2 or later, Node.js version 12 or later, npm version 6 or later, and preferably the latest minor version installed on your computer. But this is the case when running Node-RED in a local environment. In the case of running on IBM Cloud, which is one of the tutorials in this book, it depends on the environment of the cloud platform. All code examples have been tested on macOS, Windows, and Raspberry Pi OS, but some chapters have command-line instructions based on macOS.

Software/hardware covered in the book	OS requirements
Node-RED 1.2	Windows, macOS, or Raspberry Pi OS
Node.js 12	Windows, macOS, or Raspberry Pi OS
npm 6	Windows, macOS, or Raspberry Pi OS

If you are using the digital version of this book, we advise you to type the code yourself or access the code via the GitHub repository (link available in the next section). Doing so will help you avoid any potential errors related to the copying and pasting of code.

Download the example code files

You can download the example code files for this book from GitHub at `https://github.com/PacktPublishing/-Practical-Node-RED-Programming`. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `https://static.packt-cdn.com/downloads/9781800201590_ColorImages.pdf`.

Conventions used

There are a number of text conventions used throughout this book.

`Code in text`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "Let's attach a page heading to the body with the `<h1>` tag."

A block of code is set as follows:

```
// generate random number
var min = 1 ;
var max = 10 ;
var a = Math.floor( Math.random() * (max + 1 - min) ) + min ;

// set random number to message
msg.payload = a;

// return message
return msg;

```

Any command-line input or output is written as follows:

```
$ node --version
v12.18.1
```

```

$ npm -version
6.14.5
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "After selecting the name and payment plan, click the **Select Region** button."

Tips or important notes
Appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at `customercare@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packtpub.com/support/errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packt.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit `packt.com`.

Section 1: Node-RED Basics

In this section, readers will understand what a **flow-based programming** (**FBP**) tool is, including Node-RED, along with how to undertake IoT/web programming with it, and will learn how to use the Node-RED flow editor at a basic level.

In this section, we will cover the following chapters:

1
Introducing Node-RED and Flow-Based Programming

This chapter will help you grow from being a reader to being a Node-RED user. First, you'll learn about the history of **Flow-based programming (FBP)** tools, not just Node-RED. You will then gain a broad understanding of the entirety of Node-RED as a useful tool for building web applications and the **Internet of Things** (**IoT**) data handling, before learning what IoT and Node.js are in terms of Node-RED.

Providing technical content will help accelerate your software application development, but if you take a look at the history of the Node-RED tool itself, it will help you better understand why you need a FBP tool such as Node-RED. That is what we will be doing in this chapter.

More specifically, we'll be covering the following topics:

- What is FBP?
- What is Node-RED?
- Node-RED benefits
- Node-RED and IoT

Let's get started!

What is FBP?

So, what is FBP in the first place? It's the workflows you use in your work that you can easily imagine. Let's recall those workflows.

Workflows

In a normal workflow, boxes and wires indicate the process flow. It may be just one business design. Boxes represent processes. Box processing is defined by who, when, where, what, and how much. Sometimes, it's like explicitly writing out the flow of processing, such as by using swim lanes or placing writing definitions inside boxes. In any case, looking at the box should reveal what will be done.

On the other hand, let's try to summarize this business process as a document. Don't you think it will be complicated? Who will do what as they read it, even if they use some paragraphs well to put it together? When will you do it? It could be confusing:

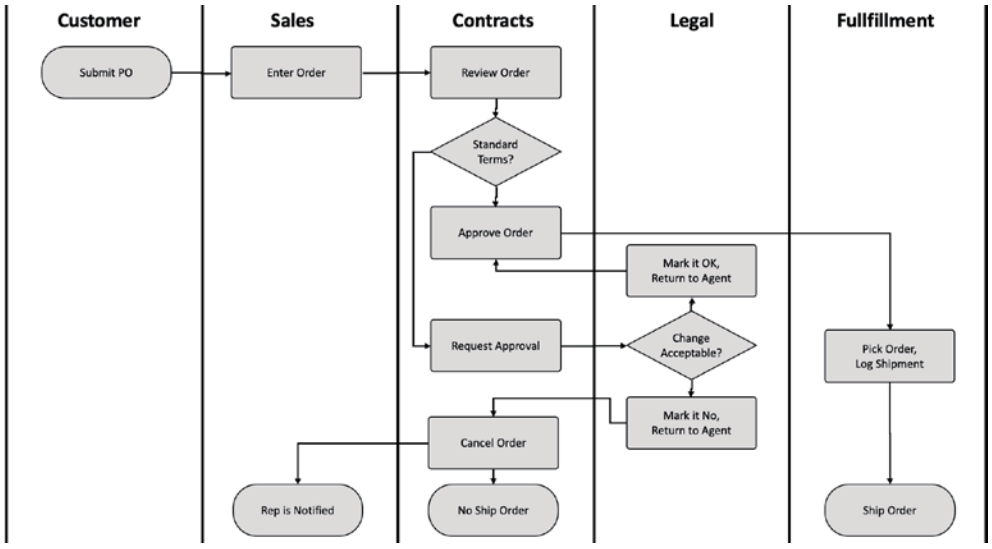


Figure 1.1 – Workflow example

Now, let's get back to software programming. FBP is a kind of concept for software programming that defines an application with a data flow. Each part of the process is there as a black box. They communicate data between connected black boxes that have been predefined. FBP is said to be component-oriented because these black-box processes can be connected repeatedly to form several applications without needing to be modified internally. Let's explore FBP in more detail.

Flow-based programming (FBP)

I think FBP is a good blend of workflow and dataflow. FBP uses a **data factory** metaphor to define an application. It sees an application as a network of asynchronous processes that start at some point and do a single sequential process that does one operation at a time until it ends, rather than communicating by using a stream of structured chunks of data. This is called an **information packet** (**IP**). This view focuses on the data and its transformation process to produce the output that is needed. Networks are usually defined outside a process as a list of connections that is interpreted by a piece of software called a **scheduler**.

Processes communicate via fixed capacity connections. Connections are connected to processes using ports. The port has a specific name that is agreed on by the network definition and the process code. At this point, it is possible to execute the same code by using multiple processes. A particular IP is usually only owned by a single process or transferred between two processes. The port can be either a normal type or an array type.

FBP applications typically run faster than traditional programs, since FBP processes can continue to run as long as there is room to put in data and output to process. It does not require any special programming and makes optimal use of all the processors on the machine.

FBP has a high-level, functional style so that the behavior of the system can be easily defined; for example, in a distributed multi-party protocol such as a distributed data flow model, for accurately analyzing the criteria for determining whether a variable or statement behaves correctly:

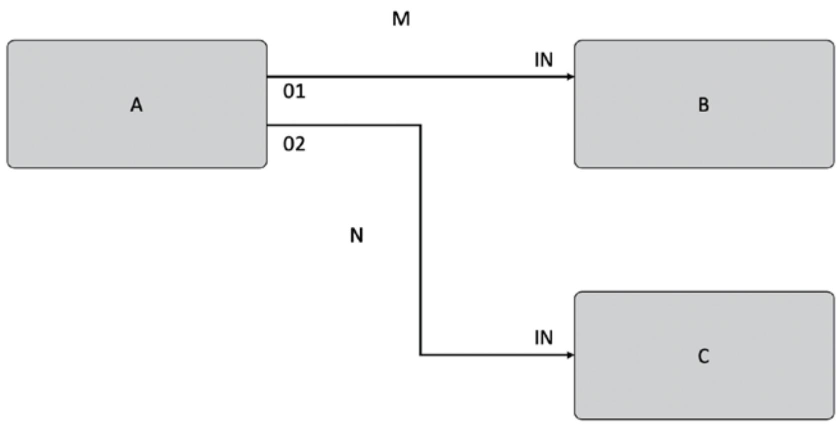


Figure 1.2 – Simple FBP design example

Now that you have a solid understanding of FBP, let's learn how Node-RED can be implemented in this way.

What is Node-RED?

Node-RED is one of the FBP tools that we have described so far. Developed by IBM's Emerging Technology Services team, Node-RED is now under the OpenJS Foundation.

Overview

FBP was invented by J. Paul Morrison in the 1970s. As we mentioned earlier, FBP describes the behavior of the application as a black box network, which in Node-RED is described as a "node." Processing is defined in each node; data is given to it, processing is performed using that data, and that data is passed to the next node. The network plays the role of allowing data to flow between the nodes.

This kind of programming method is very easy to use to make a model visually and makes it easy to access for several layer users. Anybody can understand what the flow is doing if a problem is broken down into each step. That's why you don't need to the code inside the nodes:

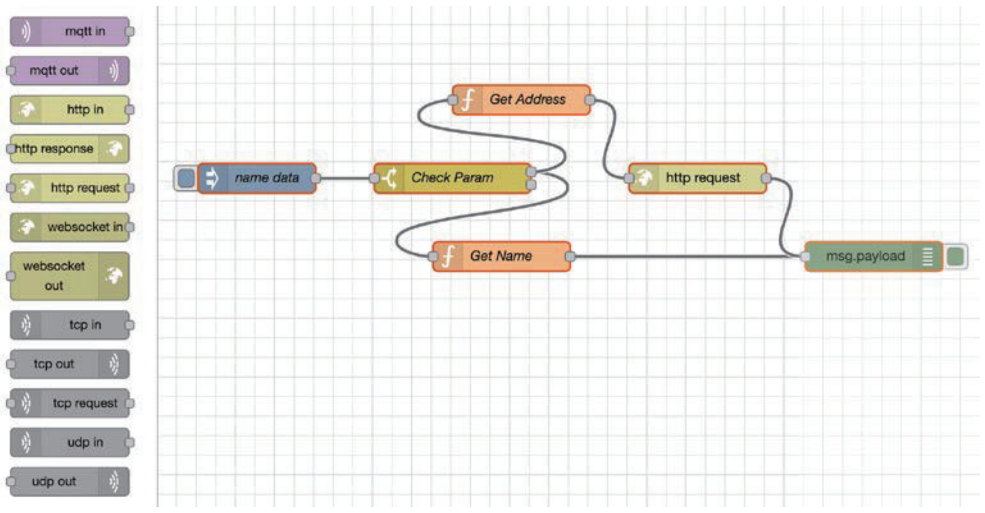


Figure 1.3 – Node-RED Flow Editor as an FBP tool

Flow editor and runtime

Node-RED is not only a programming tool but also an execution platform that wraps up the Node.js runtime for applications that are built using Node-RED.

We need to use the **flow editor** to make Node-RED applications for IoT, web services, and more. The flow editor is also a Node.js web application. We will tell you how to use flow editor clearly in *Chapter 3, Understanding Node-RED Characteristics by Creating Basic Flows*.

The flow editor, which is the core function of Node-RED, is actually a web application made with Node.js. It works with the Node.js runtime. This flow editor operates within the browser. You must select the node you want to use from the various nodes in the palette and drag it to the workspace. Wiring is the process of connecting the nodes to each other, which creates an application. The user (developer) can deploy the application to the target runtime with just one click.

The palette that contains various nodes can easily be expanded as you can install new nodes created by developers, meaning you can easily share the flow you created as a JSON file to the world. Before we explore the benefits of Node-RED, let's look at the brief history behind its creation.

History and origin of Node-RED

In early 2013, Nick-O'Leary and Dave Conway-Jones from IBM UK's Emerging Technology Services Team created Node-RED.

Originally, it was a just **proof of concept** (**PoC**) to help visualize and understand the mapping between **Message Queue Telemetry Transport** (**MQTT**) topics, but soon, it became a very popular tool that could be easily extended to various uses.

Node-RED became open source in September 2013 and remains to be developed as open source now. It became one of the founding projects of the JS Foundation in October 2016, which has since merged with the Node.js Foundation to create the OpenJS Foundation, doing so in March 2019.

The OpenJS Foundation supports the growth of JavaScript and web technologies as a neutral organization to lead and keep any projects and fund activities jointly, which is beneficial to the whole of the ecosystem. The OpenJS Foundation currently hosts over 30 open source JavaScript projects, including Appium, Dojo, jQuery, Node.js, and webpack.

Node-RED has been made available under the Apache 2 license, which makes it favorable to use in a wide range of settings, both personal and commercial:

Figure 1.4 – Dave Conway-Jones and Nick O'Leary

Why is it Called Node-RED?

The official documentation (`https://nodered.org/about/` states that the name was an easy play on words that sounded like "Code Red." This was a dead end, and Node-RED was a big improvement on what it was called in its first few days of conception. The "Node" part reflects both the flow/node programming model, as well as the underlying Node.js runtime.

Nick and Dave never did come to a conclusion on what the "RED" part stands for. "Rapid Event Developer" was one suggestion, but it's never been compelled to formalize anything. And so, the name "Node-RED" came to life.

Node-RED benefits

Let's think a little here. Why do you use cars? I think the answer is very simple and clear. First of all, we can come up with the answer that they are used as a means of transportation in a broad sense. There are other options for transportation, such as walking, bicycle, train, and bus. Then, we have the reasons for choosing a car from among these other options, as follows:

- You do not get exhausted.
- You can reach your destination quickly.
- You can move at your own pace.
- You can keep your personal space.

Of course, there are some disadvantages, but I think these are the main reasons for using a car. Although other means of transportation can also serve the same purpose, the important thing is to consider the advantages and disadvantages of each, and use the car as a transportation tool for the reason that you feel is the most suitable to you.

We can see the same situation in software. As an example, why do you use Word, Excel, and PowerPoint? You'll probably use Word because it's the most efficient way to write a document. However, you could use a word processor separately or handwrite anything. Similarly, instead of Excel, you can use any other means to make spreadsheets. There are also other means if you want to make presentation materials and make them look effective, besides PowerPoint. However, you are likely to choose the optimum tool for your situation.

Let's recall what Node-RED is for. It is a FBP tool, suitable for making data control applications for web applications and IoT. Its development environment and execution environment are browser-based applications made with Node.js, which makes their development as easy as possible.

So, what is the reason for using Node-RED, which provides these kinds of features? Do you want to avoid heavy coding? Do you not have coding skills? Yes, of course, these are also reasons to use the program.

Let's recall the example of a car. In a broad sense, our dilemma (transportation) is replaced here by developing (creating) a Node.js application for describing software tools. The transport options, such as cars, bicycles, trains, buses, ships, planes, and so on, are options, and with software development, we also have numerous options, such as using Node.js scratch, or using various frameworks of Node.js and using Node-RED. As for reasons to choose Node-RED, let's take a look at some essential points.

Simplification

When programming with Node-RED, you'll notice its simplicity. As the name no-code/low-code indicates, coding is eliminated and programming is intuitively completed with a minimal number of operations needing to be used.

Efficiency

The FBP typified by Node-RED can be completed with almost only GUI operations. Node-RED flow editor takes care of building the application execution environment, library synchronization, the **integrated development environment** (**IDE**), and editor preparation so that you can concentrate on development.

Common

As represented by object-oriented development, making the source code a common component is one of the most important ideas in development. In normal coding-based development, each common component exists in functions and classes, but in Node-RED, they exist as an easy-to-understand node (just a box). If you don't have a node as a common component you want to use, anyone can create one immediately and publish it to the world.

High quality

High quality is the true value of flow-based and visual programming. Each node provided as a component is a complete module that has been unit tested. As a result, app authors can focus on checking the operation at the join level without worrying about the contents of node. This is a big factor that eliminates human error at the single level and ensures high quality.

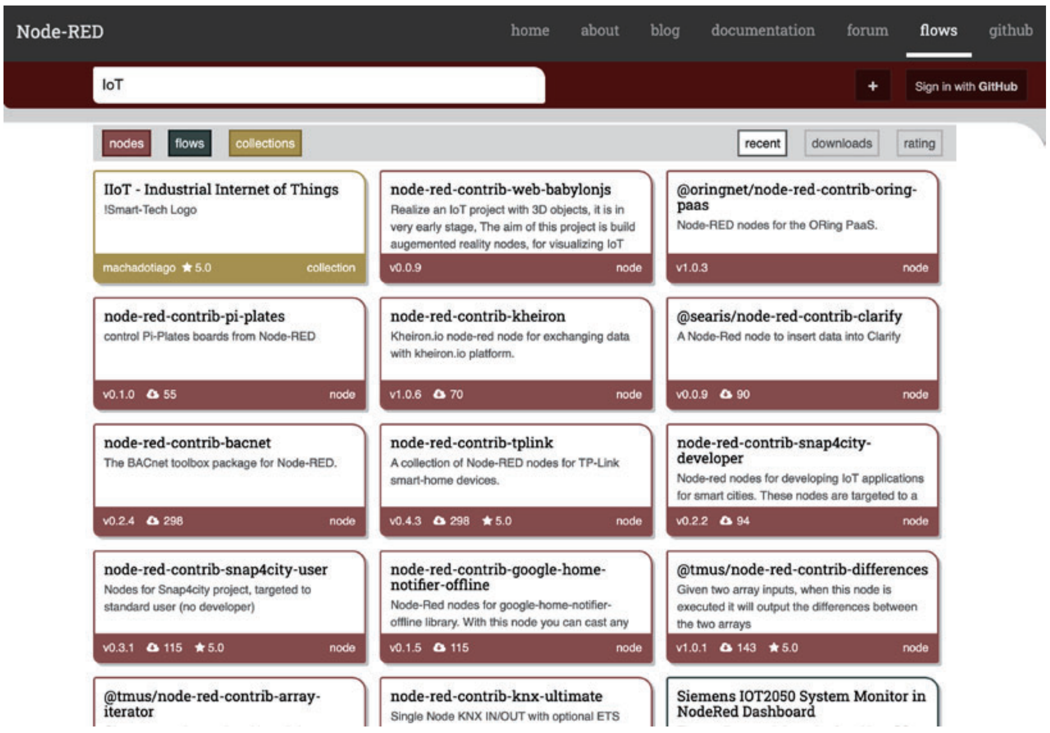
Open source

Node-RED is an open source piece of software. Therefore, it can be used flexibly under the Apache2 license. Some are developing their own services based on Node-RED, while others are changing to their own UI and deploying it as built-in. As we mentioned previously, we have also established a platform where we can publish our own developed node so that anyone can use it.

Node-RED library

The library indexes all Node-RED modules published to the public npm repository (`https://www.npmjs.com/`), assuming they follow the proper packaging guidelines.

This is the area in which we've seen the most community contribution, with well over 2,000 nodes available – which means there's something for everyone:



IIoT - Industrial Internet of Things
!Smart-Tech Logo
machadotiago ★ 5.0 collection

node-red-contrib-web-babylonjs
Realize an IoT project with 3D objects, it is in very early stage, The aim of this project is build augemented reality nodes, for visualizing IoT
v0.0.9 node

@oringnet/node-red-contrib-oring-paas
Node-RED nodes for the ORing PaaS.
v1.0.3 node

node-red-contrib-pi-plates
control Pi-Plates boards from Node-RED
v0.1.0 55 node

node-red-contrib-kheiron
Kheiron.io node-red node for exchanging data with kheiron.io platform.
v1.0.6 70 node

@searis/node-red-contrib-clarify
A Node-Red node to insert data into Clarify
v0.0.9 90 node

node-red-contrib-bacnet
The BACnet toolbox package for Node-RED.
v0.2.4 298 node

node-red-contrib-tplink
A collection of Node-RED nodes for TP-Link smart-home devices.
v0.4.3 298 ★ 5.0 node

node-red-contrib-snap4city-developer
Node-red nodes for developing IoT applications for smart cities. These nodes are targeted to a
v0.2.2 94 node

node-red-contrib-snap4city-user
Nodes for Snap4city project, targeted to standard user (no developer)
v0.3.1 115 ★ 5.0 node

node-red-contrib-google-home-notifier-offline
Node-Red nodes for google-home-notifier-offline library. With this node you can cast any
v0.1.5 115 node

@tmus/node-red-contrib-differences
Given two array inputs, when this node is executed it will output the differences between the two arrays
v1.0.1 143 ★ 5.0 node

@tmus/node-red-contrib-array-iterator

node-red-contrib-knx-ultimate
Single Node KNX IN/OUT with optional ETS

Siemens IOT2050 System Monitor in NodeRed Dashboard

Figure 1.5 – Node-RED library

Various platforms

Node-RED can be used on various platforms. That's because Node-RED itself is a Node.js application, as we mentioned previously. If you have a runtime environment for Node.js, you can run it. It is mostly used on Edge devices, cloud services, and in embedded formats.

You can get a sense of this by understanding the relationship between Node-RED and IoT and the architecture of IoT, which will be explained in the next section.

Node-RED and IoT

Again, Node-RED is a **virtual environment** that combines hardware devices, APIs, and online services in a revolutionary way on a browser. It provides the following features:

- Browser-based UI.
- Works with Node.js and is lightweight.
- Encapsulates function and can be used as a node (meaning functions are locked in an abstract capsule) .
- You can create and add your own nodes.
- Easy access to IBM Cloud services.

In other words, it can be said that this tool is suitable for building IoT-related services, such as data control on devices, and linking edge devices and cloud services. Originally, the development concept of Node-RED was for IoT, so this makes sense.

Now, let's look at the basic structure of IoT so that those who are only vaguely aware of IoT can understand it. It can be said that IoT is basically composed of six layers, as shown in the following diagram:

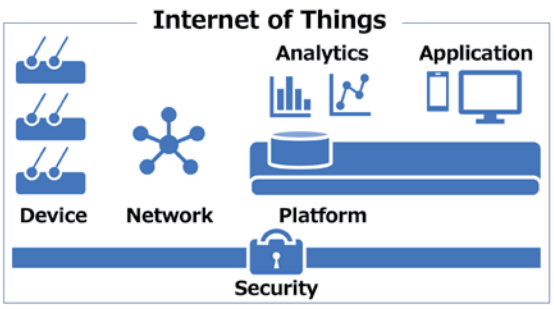


Figure 1.6 – IoT six layers

Let's take a look at these in more detail.

Device

The device is a so-called edge device. IoT has various sensors and handles the data that's acquired from them. Since it doesn't make sense to have the data only on the edge device, we need to send that data through the gateway to the network.

Network

This is the network that's required to send the data that's been obtained from the device to a server on the internet. It usually refers to the internet. In addition to the internet, there is also a P2P connection via Bluetooth or serial.

Platform

The party that receives and uses the data is the platform. We may also have a database for activating and authenticating things, managing communications, and persisting received data.

Analytics

This is a layer that analyzes the received data. Broadly speaking, it may be classified as an application. This is the part that prepares the data so that it can be processed into a meaningful form.

Application

An application provides a specific service based on data analysis results. It can be a web or mobile application, or it can be a hardware-specific embedded application. It can be said to be the layer that's used by the end user of the IoT solution.

Now that we have an understanding of IoT, we will examine why Node-RED should be used for it.

Node-RED and IoT

While explaining IoT so far, we've made it clear why Node-RED is suitable for IoT. For example, you can understand why FBP tools that have been developed for IoT survive when used with Node-RED. In particular, the following three points should be taken into account:

- Since it can be run on edge devices (pre-installed on specific versions of Raspberry Pi OS), it is ideal for data handling at the device layer.
- Since it can be run on the cloud (provided as a default service in IBM Cloud), it is easy to link with storage and analysis middleware.

- Since MQTT and HTTP protocols can be covered, it is very easy to exchange data between the edge device and the server processing cloud.

In this way, Node-RED, which largely covers the elements required for IoT, is now used for a wide range of applications, such as web services and chart display, as well as programming for IoT. Also, as of June 2020, if you look at Google Trends for Node-RED, you can see that the number of users is gradually increasing. As such, Node-RED is a very attractive FBP tool:

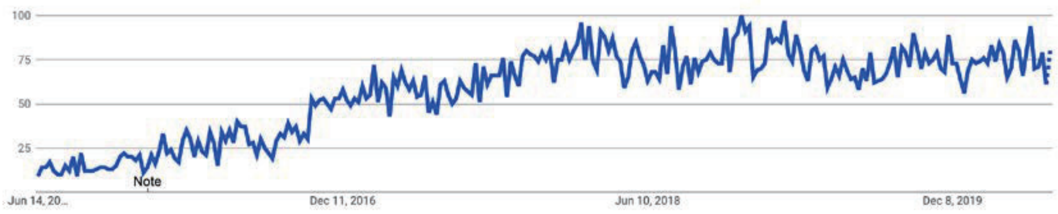


Figure 1.7 – Google Trends for "Node-RED"

A typical edge device that can use Node-RED is Raspberry Pi. Of course, it is possible to use Node-RED on other platforms, but it goes well with Raspberry Pi, which also has a pre-installed version of the OS.

> **Raspberry Pi OS Supports Node-RED**
>
> Node-RED has also been packaged for the Raspberry Pi OS repositories and appears in their list of recommended software. This allows it to be installed using `apt-get install Node-RED` and includes the Raspberry Pi OS-packaged version of Node.js, but does not include npm. More information can be found at `https://nodered.org/docs/getting-started/raspberrypi`.

IBM Cloud is a typical cloud platform that can use Node-RED. Of course, you can use Node-RED on other clouds, but IBM Cloud provides a service that anyone can easily start.

> **Important Note**
>
> Node-RED is available on the IBM Cloud platform as one of its Starter Kits applications in their catalog. It is very easy to start using the flow editor as a web application on IBM Cloud (`https://nodered.org/docs/getting-started/ibmcloud`).

Summary

In this chapter, you learned what FBP and Node-RED are. Due to this, you now understand why Node-RED is currently loved and used by lots of people as an FBP tool. At this point, you may want to build an application using Node-RED. In the next chapter, we'll install Node-RED in our environment and take a look at it in more depth.

2
Setting Up the Development Environment

In this chapter, you will install the tools that you'll need to use Node-RED. This extends not only to Node-RED itself, but to its runtime, Node.js, and how to update both Node-RED and Node.js.

Node-RED released its 1.0 milestone version in September 2019. This reflects the maturity of the project, as it is already being widely used in production environments. It continues to be developed and keeps up to date by making changes to the underlying Node.js runtime. You can check the latest status of Node-RED's installation at `https://nodered.org/docs/getting-started/`.

There are a number of installation guides on the Node-RED official website, such as local, Raspberry Pi, Docker, and major cloud platforms.

In this chapter, you will learn how to install Node-RED on your local computer, whether you are running it on Windows, Mac, or on a Raspberry Pi. We will cover the following topics:

- Installing `npm` and Node.js for Windows
- Installing `npm` and Node.js for Mac
- Installing `npm` and Node.js for Raspberry Pi
- Installing Node-RED for Windows
- Installing Node-RED for Mac
- Installing Node-RED for Raspberry Pi

By the end of this chapter, we'll have all the necessary tools installed and be ready to move on to building some basic flows with Node-RED.

For reference, the author's test operation environment is Windows 10 2004 18363.476, macOS Mojave 10.14.6 (18G5033), and Raspberry Pi OS 9.4 stretch.

Technical requirements

You will need to install the following for this chapter:

- Node.js (v12.18.1)*
- npm (v6.14.5)*

*LTS version at the time of writing for both.

Installing npm and Node.js for Windows

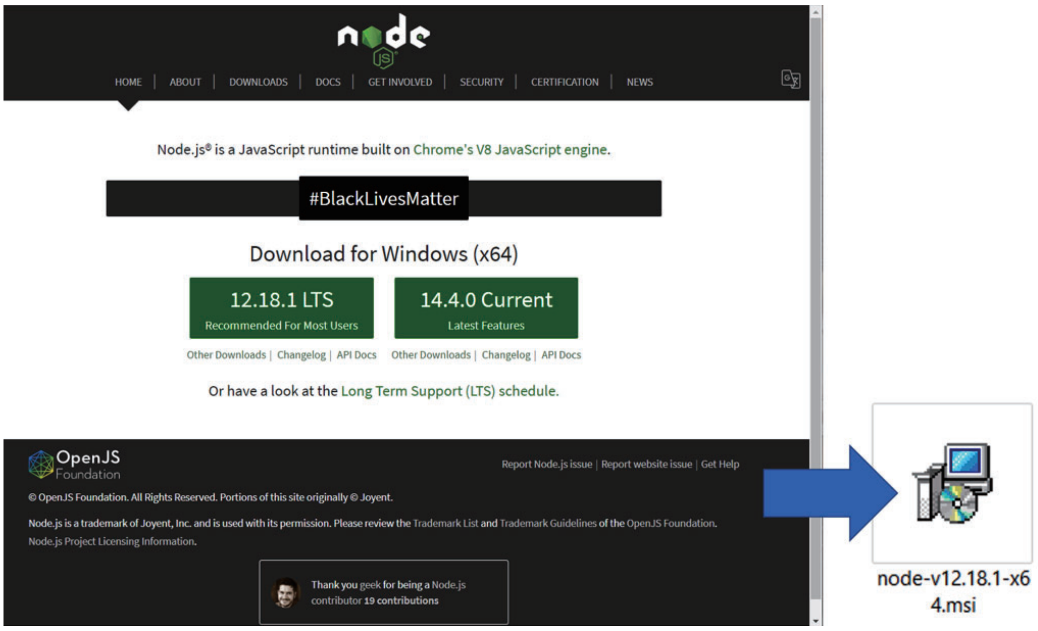
If you want to use Node-RED on Windows, you must install npm and Node.js via the following website:

`https://nodejs.org/en/#home-downloadhead`.

You can get the Windows Installer of Node.js directly there. After that, follow these steps:

1. Access the original Node.js website and download the installer.

   You can choose both versions – **Recommended** or **Latest Features** – but in this book, you should use the **Recommended** version:



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Windows (x64)

Or have a look at the Long Term Support (LTS) schedule.

© OpenJS Foundation. All Rights Reserved. Portions of this site originally © Joyent.

Node.js is a trademark of Joyent, Inc. and is used with its permission. Please review the Trademark List and Trademark Guidelines of the OpenJS Foundation.

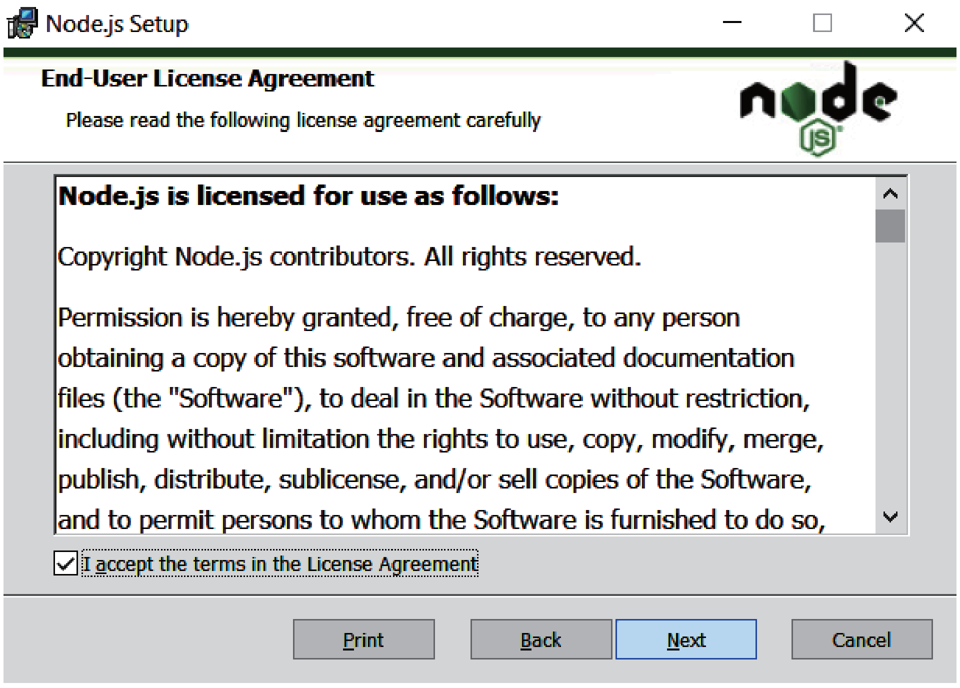
Node.js Project Licensing Information.

Thank you geek for being a Node.js contributor 19 contributions

Figure 2.1 – Choosing a Recommended version installer

2. Click the `msi` file you downloaded to start installing Node.js. It includes the current version of npm. Node-RED is running on the Node.js runtime, so it is needed.
3. Simply click the buttons of the dialog windows according to the installation wizard, though there are some points to bear in mind during the install.

4. Next, you need to accept the End-User License Agreement:


Node.js Setup
End-User License Agreement
Please read the following license agreement carefully

Node.js is licensed for use as follows:

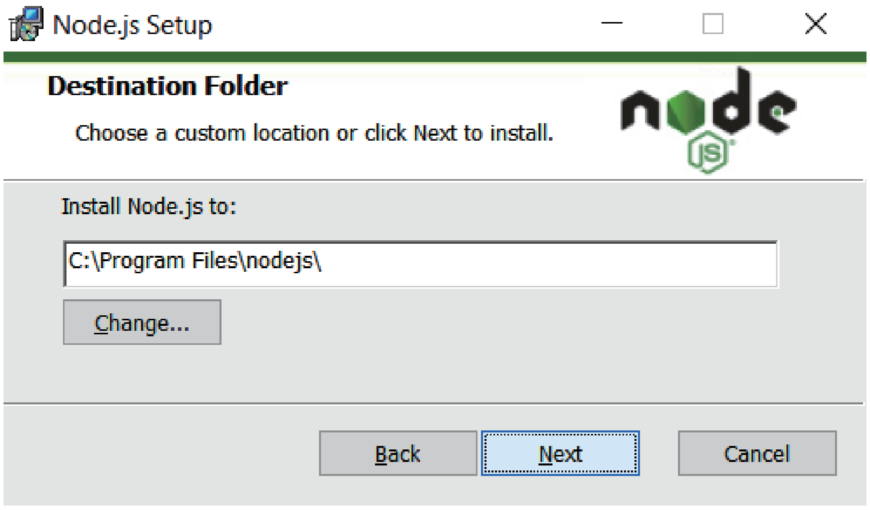
Copyright Node.js contributors. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so,

☑ I accept the terms in the License Agreement

Figure 2.2 – End-User License Agreement window

You can also change the install destination folder. In this book, the default folder (`C:/Program Files/nodejs/`) will be used:

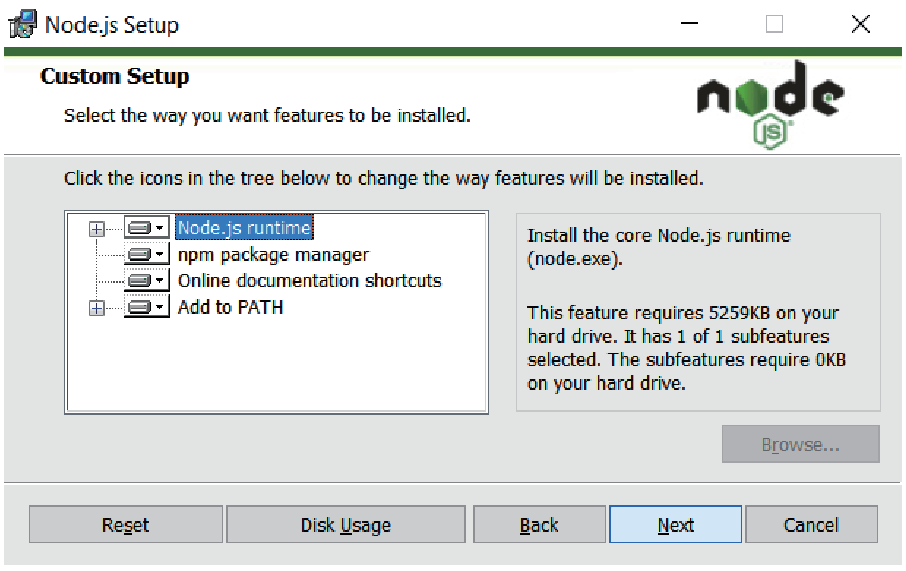


Install Node.js to:

C:\Program Files\nodejs\

Figure 2.3 – Installing the destination folder

5. No custom setup is needed on the next screen. You can select **Next** with only the default features selected:

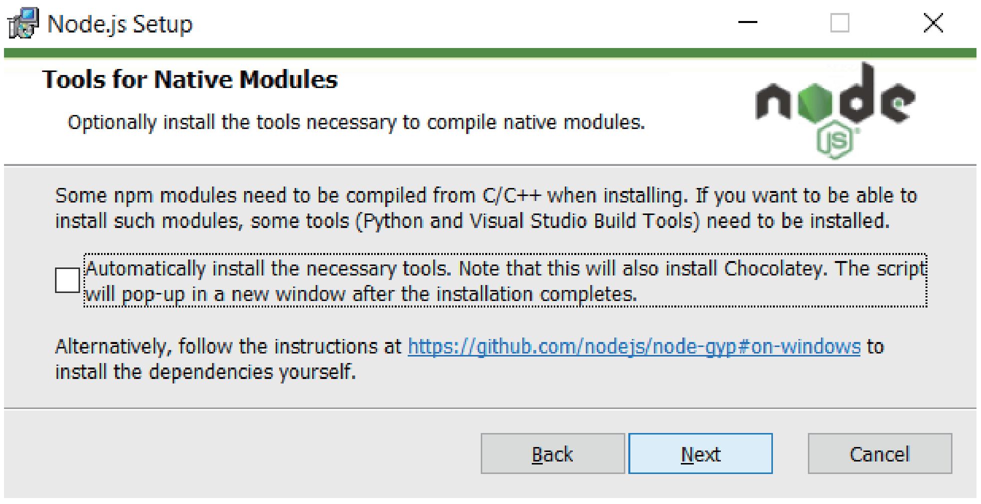

Node.js Setup
Custom Setup
Select the way you want features to be installed.
Click the icons in the tree below to change the way features will be installed.
Node.js runtime
npm package manager
Online documentation shortcuts
Add to PATH
Browse...
Reset
Disk Usage
Back
Next
Cancel

Install the core Node.js runtime (node.exe).

This feature requires 5259KB on your hard drive. It has 1 of 1 subfeatures selected. The subfeatures require 0KB on your hard drive.

Figure 2.4 – No custom setup is needed

6. On the following screen, you can click **Next** without checking anything. However, it's OK to install the tools that can be selected here. This includes the installations and settings the path of these environments (Visual C++, windows-build-tools, and Python):


Node.js Setup
Tools for Native Modules
Optionally install the tools necessary to compile native modules.
Back
Next
Cancel

Some npm modules need to be compiled from C/C++ when installing. If you want to be able to install such modules, some tools (Python and Visual Studio Build Tools) need to be installed.

☐ Automatically install the necessary tools. Note that this will also install Chocolatey. The script will pop-up in a new window after the installation completes.

Alternatively, follow the instructions at https://github.com/nodejs/node-gyp#on-windows to install the dependencies yourself.

Figure 2.5 – Tools for Native Modules window

7. Check the versions of your tools with the following commands when the installation for Node.js has finished:

```
$ node --version
v12.18.1

$ npm –version
6.14.5
```

When the installations of Node.js and npm are complete, you can check their version numbers. With this, you are prepared to install Node-RED.

> **Important note**
> Depending on the project, it is common for the operation to be stable with the old Node.js version but for it not to work if you use a different version of Node.js. However, uninstalling your current version of Node.js and installing the desired version of Node.js every time you switch projects takes time. So, if you're using Windows, I recommend using a Node.js version management tool such as nodist (`https://github.com/nullivex/nodist`). There are other kinds of version control tools for Node.js, so please try to find one that is easy for you to use.

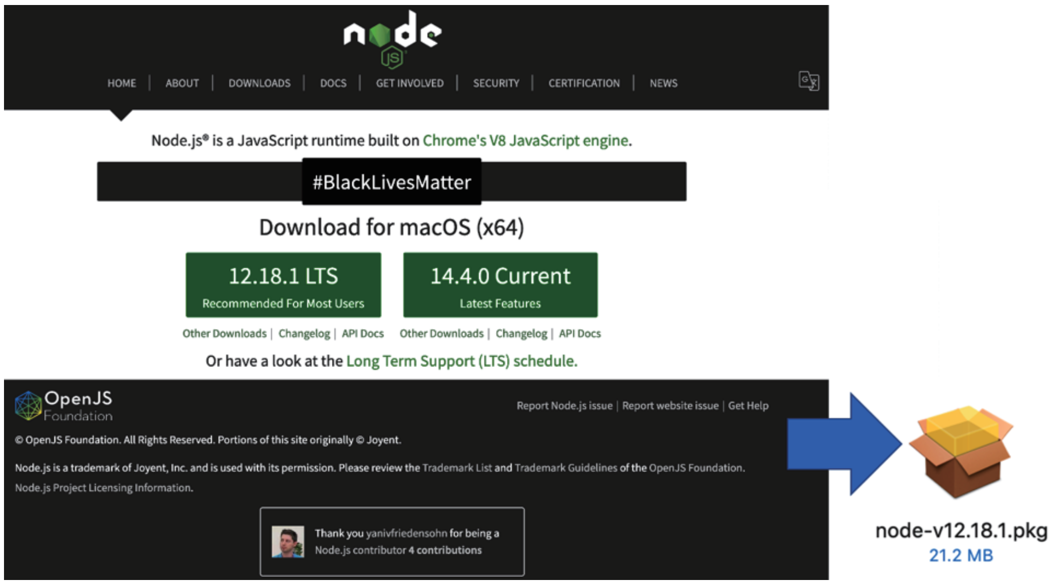
Installing npm and Node.js for Mac

If you want to use Node-RED on macOS, you must install `npm` and Node.js via the following website:

`https://nodejs.org/en/#home-downloadhead`

You can get the Mac Installer for Node.js directly there.

Access the original Node.js website and download the installer. You can choose either the recommended or latest features version, but for this book, you should use the recommended version:


HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS
#BlackLivesMatter
12.18.1 LTS
Recommended For Most Users
14.4.0 Current
Latest Features
Other Downloads | Changelog | API Docs
node-v12.18.1.pkg
21.2 MB

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

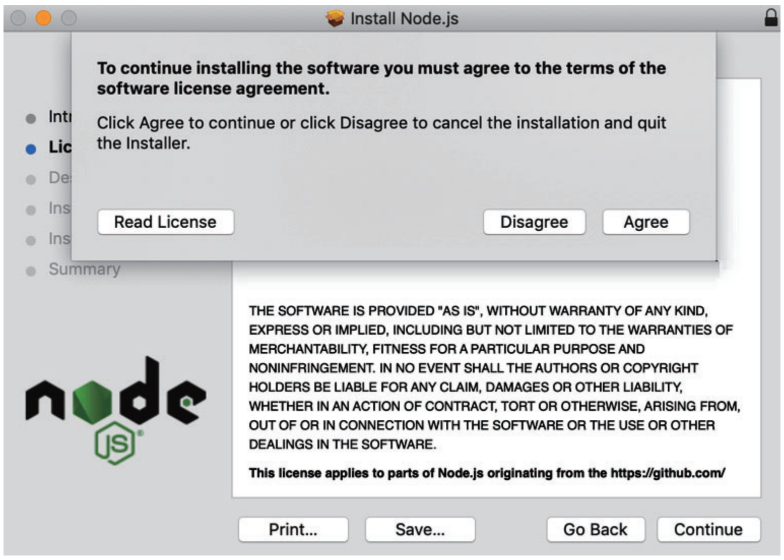
Download for macOS (x64)

Or have a look at the Long Term Support (LTS) schedule.

Figure 2.6 – Choosing a recommended version installer

Click the `.pkg` file you downloaded to start installing Node.js. It includes the current version of `npm`. Node-RED is running on the Node.js runtime, so it is needed. Simply click according to the installation wizard, though there are some points in the installation to pay attention to.

You need to accept the End-User License Agreement:



To continue installing the software you must agree to the terms of the software license agreement.

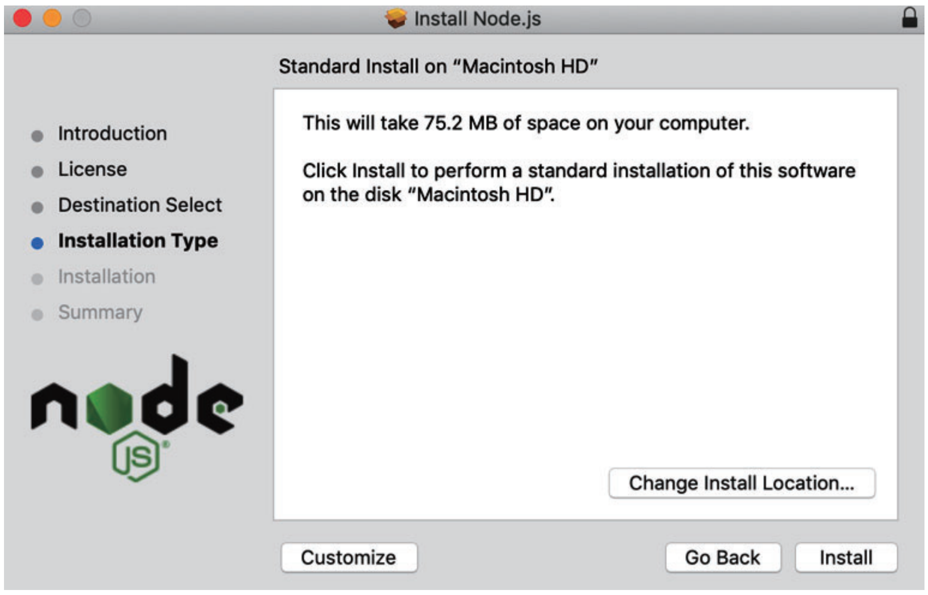
Click Agree to continue or click Disagree to cancel the installation and quit the Installer.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This license applies to parts of Node.js originating from the https://github.com/

Figure 2.7 – End-User License Agreement window

You can change the installation location. In this book, the default location (Macintosh HD) will be used:



Standard Install on "Macintosh HD"

This will take 75.2 MB of space on your computer.

Click Install to perform a standard installation of this software on the disk "Macintosh HD".

Figure 2.8 – Install location

You can check the versions of your tools with the following commands when the installation for Node.js has finished. Once you have finished installing Node.js and `npm`, you can check their version numbers. You have already prepared to install Node-RED:

```
$ node --version
v12.18.1

$ npm -version
6.14.5
```

Note

Depending on the project, it is common for the operation to be stable with the old Node.js version, and that it will not work if you use a different version of Node.js. However, uninstalling the current Node.js version and installing the desired version of Node.js every time you switch projects takes time. So, if you're using macOS, I recommend using a Node.js version management tool such as Nodebrew (`https://github.com/hokaccha/nodebrew`). There are other kinds of version control tools for Node.js, so please try to find one that is easy for you to use.

Now that we have covered the installation processes for both Windows and Mac, let's learn how to install `npm` and Node.js for Raspberry Pi.

Installing npm and Node.js for Raspberry Pi

If you want to use Node-RED on Raspberry Pi, congratulations – you are already prepared to install Node-RED. This is because Node.js and npm are installed by default. You can use the existing installation script to install Node-RED, including Node.js and npm. This script will be described later in this chapter, in the *Installing Node-RED for Raspberry Pi* section, so you can skip this operation for now.

However, you should check your Node.js and npm versions on your Raspberry Pi. Please type in the following commands:

```
$ node --version
v12.18.1

$ npm –version
6.14.5
```

If it is not the LTS version or stable version, you can update it via the CLI. Please type in and run the following commands to do this. In this command, on the last line, `lts` has been used, but you can also put `stable` instead of `lts` if you want to install the stable version:

```
$ sudo apt-get update
$ sudo apt-get install -y nodejs npm
$ sudo npm install npm n -g
$ sudo n lts
```

Now that we have successfully checked the versions of Node.js and npm on our Raspberry Pi and updated them (if applicable), we will move on to installing Node-RED for Windows.

Important note

The script the Node-RED project provides takes care of installing Node.js and `npm`. It is not generally recommended to use the versions that are provided by Raspberry Pi OS due to the strange ways they package them.

Installing Node-RED for Windows

In this section, we will explain how to set up Node-RED in a Windows environment. This procedure is for Windows 10, but it will work for Windows 7 and Windows Server 2008 R2 and above as well. Windows 7 or earlier versions of Windows Server 2008 R2 are not currently supported and are not recommended.

For Windows, installing Node-RED as a global module adds the `node-red` command to your system path. Run the following command in Command Prompt:

```
$ npm install -g --unsafe-perm node-red
```

Once you have finished installing Node-RED, you can use Node-RED straight away. Please run the following command. After running this command, you will recognize the URL being used to access the Node-RED flow editor. Usually, localhost (127.0.0.1) with the default port 1880 will be allocated:

```
$ node-red
Welcome to Node-RED
===================
…
[info] Starting flows
[info] Started flows
[info] Server now running at http://127.0.0.1:1880/

```

Let's access Node-RED on a browser. For this, type in the URL you received from Command Prompt. I strongly recommend using Chrome or Firefox for running Node-RED:

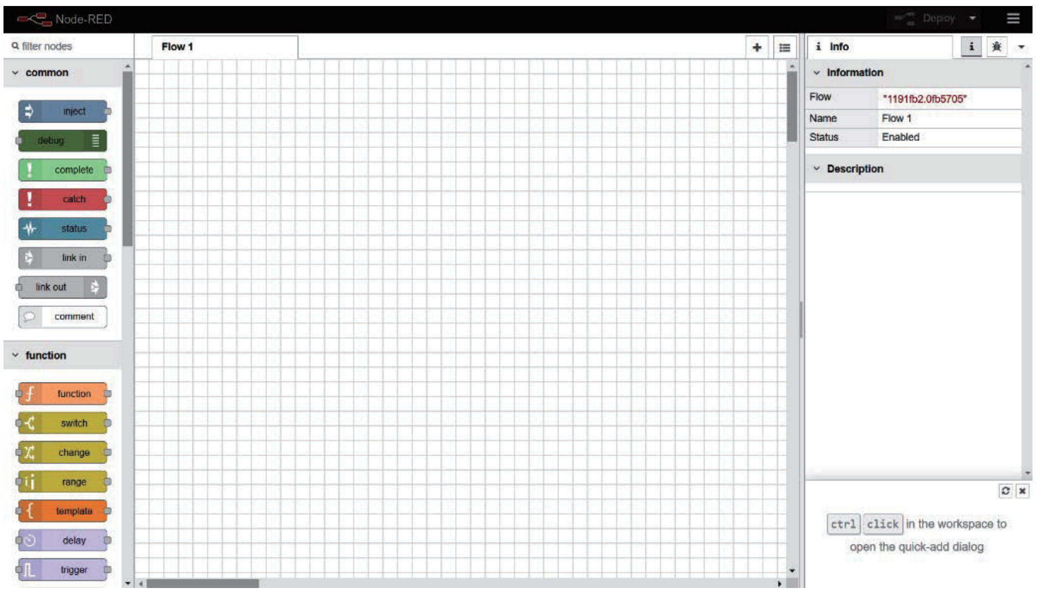


Figure 2.9 – Node-RED flow editor

Now, you are ready to program in Node-RED. From *Chapter 3*, *Understanding Node-RED Characteristics by Creating Basic Flows*, onward, we will learn how to actually build an application using Node-RED.

For now, let's move on to installing Node-RED in macOS.

Installing Node-RED for Mac

In this section, we will explain how to set up Node-RED in a macOS environment. This procedure is for macOS Mojave. It will likely work for all versions of Mac OS X, but I strongly recommend that you use the current version of macOS.

For macOS, installing Node-RED as a global module adds the `node-red` command to your system path. Run the following command in the Terminal. You may need to add `sudo` at the front of the command, depending on your local settings:

```
$ sudo npm install -g --unsafe-perm node-red
```

You can also install Node-RED with other tools. This is mainly for Mac/Linux or the kinds of OS that support the following tools:

1. Docker (`https://www.docker.com/`), if you have the environment for running Docker.

   The current Node-RED 1.x repository on Docker Hub has been renamed "`nodered/node-red`".

   Versions up to 0.20.x are available from `https://hub.docker.com/r/nodered/node-red-docker`.

   > **Important note**
   > When running Node-RED with Docker, you need to ensure that the added nodes and flows will not be lost if the container breaks. This user data can be persisted by mounting the data directory to a volume outside the container. You can also do this by using a bound mount or a named data volume.

   Run the following command to install Node-RED with Docker:

   ```
   $ docker run -it -p 1880:1880 --name mynodered nodered/
   node-red
   ```

2. Snap (`https://snapcraft.io/docs/installing-snapd`) if your OS supports it.

   If you install it as a Snap package, you can run it in a secure container that doesn't have access to the external features you have to use, such as the following:

- Access main system storage (only read/write to local home directory is allowed).
- Gcc: Required to compile the binary components for the node you want to install.
- Git: Required if you want to take advantage of project features.
- Direct access to GPIO hardware.
- Access to external commands, such as flows executed in Exec nodes.

  There's less security for containers, but you can also run them in **classic** mode, which gives you more access.

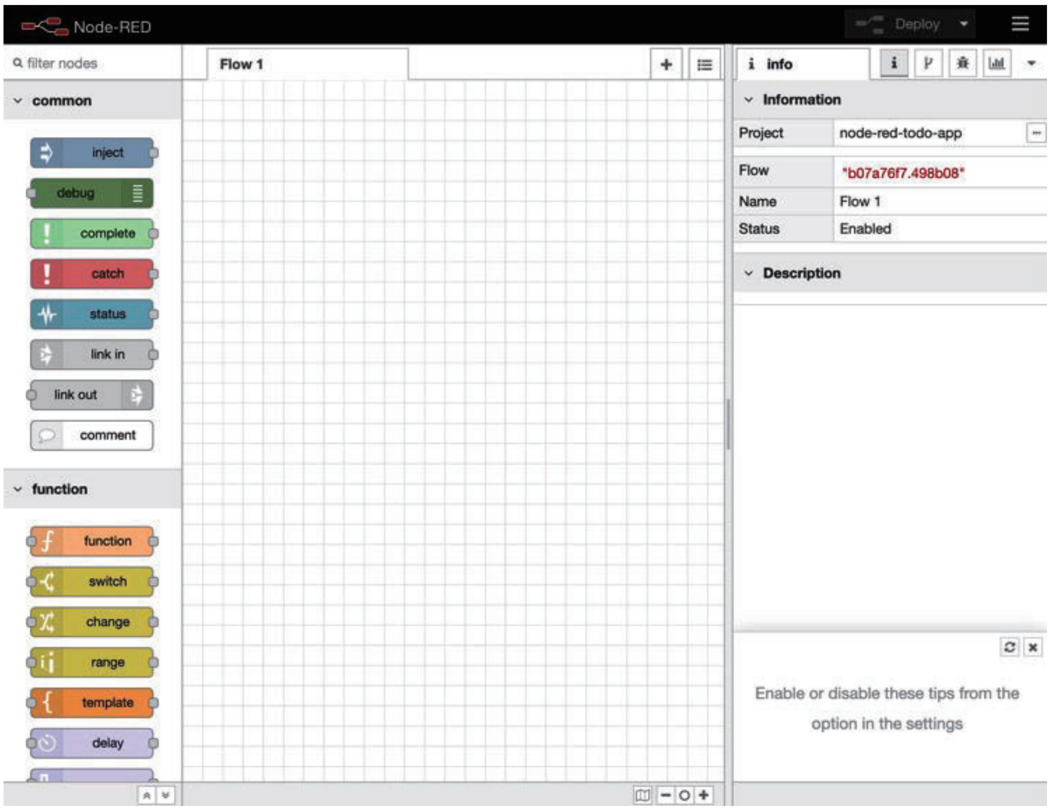
  Run the following command to install Node-RED with Snap:

```
$ sudo snap install node-red
```

Once you have finished installing Node-RED, you can use Node-RED immediately. Please run the following command. After running this command, you can find the URL for accessing the Node-RED flow editor. Usually, localhost (`127.0.0.1`) with the default port `1880` will be allocated:

```
$ node-red
Welcome to Node-RED
===================
…
 [info] Server now running at http://127.0.0.1:1880/
 [info] Starting flows
 [info] Started flows

```

Let's access Node-RED on a browser. Type in the URL you received from Command Prompt. I strongly recommend using Chrome or Firefox for running Node-RED:



Project node-red-todo-app
Flow "b07a76f7.498b08"
Name Flow 1
Status Enabled

Figure 2.10 – Node-RED flow editor

Now, you are ready to program in Node-RED. In *Chapter 3*, *Understanding Node-RED Characteristics by Creating Basic Flows,* we will learn how to actually build an application using Node-RED.

Our final installation will be for Node-RED on Raspberry Pi.

Installing Node-RED for Raspberry Pi

In this section, we will explain how to set up Node-RED in a Raspberry environment. This procedure is for Raspberry Pi OS Buster (Debian 10.x), but it will work for Raspberry Pi OS Jessie (Debian 8.x) and above.

You can check your version of Raspberry Pi OS easily. Just run the following command on your Terminal:

```
$ lsb_release -a
```

If you want to also check the version of Debian you have, please run the following command:

```
$ cat /etc/debian_version
```

You have now prepared to install Node-RED. The following script installs Node-RED, including Node.js and `npm`. This script can also be used for upgrading your application, which you have already installed.

> **Note**
>
> This instruction is subject to change, so it is recommended that you refer to the official documentation as needed.

This script works on Debian-based operating systems, including Ubuntu and Diet-Pi:

```
$ bash <(curl -sL https://raw.githubusercontent.com/node-red/
linux-installers/master/deb/update-nodejs-and-nodered)
```

You may need to run `sudo apt install build-essential git` to ensure that npm can build the binary components that need to be installed.

Node-RED is already packaged as a Raspberry Pi OS repository and is included in the *Recommended Software* list. It can be installed with the `apt-get install Node-RED` command, and it also contains a Raspberry Pi OS packaged version of Node.js, but npm is not included.

While using these packages may seem convenient at first glance, it is highly recommended to use the installation script instead.

After the installation, you can start Node-RED and access the Node-RED flow editor. We have two ways to start it, as follows:

1. Run with the CLI: If you want to run Node-RED locally, you can start Node-RED by using the `node-red` command in your Terminal. Then, you can stop it by pressing *Ctrl* + *C* or closing the Terminal window:

   ```
   $ node-red
   ```

2. Run via Programming menu: Once Node-RED has been installed, you can start it from the Raspberry Pi menu. Click **Menu | Programming | Node-RED** to open the Terminal and launch Node-RED. Once Node-RED has been launched, you can access the Node-RED flow editor from your browser, just as you would in the CLI:

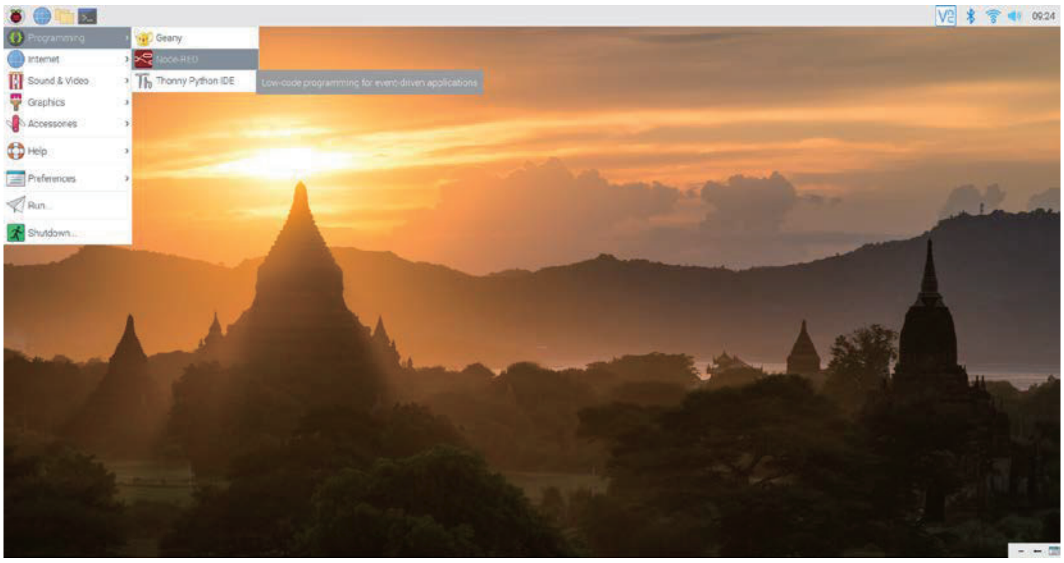
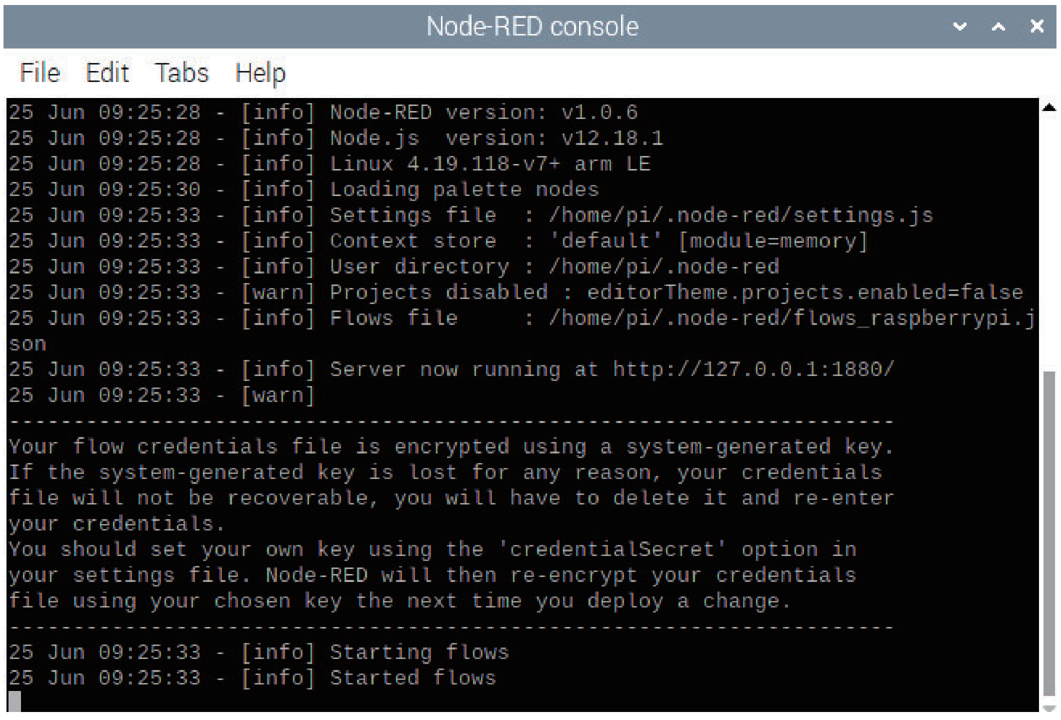


Figure 2.11 – Accessing Node-RED via the Raspberry Pi menu

After launching Node-RED from the menu, you should check the Node-RED running process on your Terminal and find the URL of the Node-RED flow editor. It is usually the same URL as the one that can be launched via the CLI directly:



```
25 Jun 09:25:28 - [info] Node-RED version: v1.0.6
25 Jun 09:25:28 - [info] Node.js  version: v12.18.1
25 Jun 09:25:28 - [info] Linux 4.19.118-v7+ arm LE
25 Jun 09:25:30 - [info] Loading palette nodes
25 Jun 09:25:33 - [info] Settings file  : /home/pi/.node-red/settings.js
25 Jun 09:25:33 - [info] Context store  : 'default' [module=memory]
25 Jun 09:25:33 - [info] User directory : /home/pi/.node-red
25 Jun 09:25:33 - [warn] Projects disabled : editorTheme.projects.enabled=false
25 Jun 09:25:33 - [info] Flows file     : /home/pi/.node-red/flows_raspberrypi.j
son
25 Jun 09:25:33 - [info] Server now running at http://127.0.0.1:1880/
25 Jun 09:25:33 - [warn]
---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------
25 Jun 09:25:33 - [info] Starting flows
25 Jun 09:25:33 - [info] Started flows
```

Figure 2.12 – Checking the URL to access the Node-RED flow editor

Let's access Node-RED on a browser. You can type in the URL you received from the Command Prompt to do this. If your Raspberry Pi default web browser is Chromium, then there should be no problems with using Node-RED. However, if you wish to use another browser, I strongly recommend installing Chromium for running Node-RED:

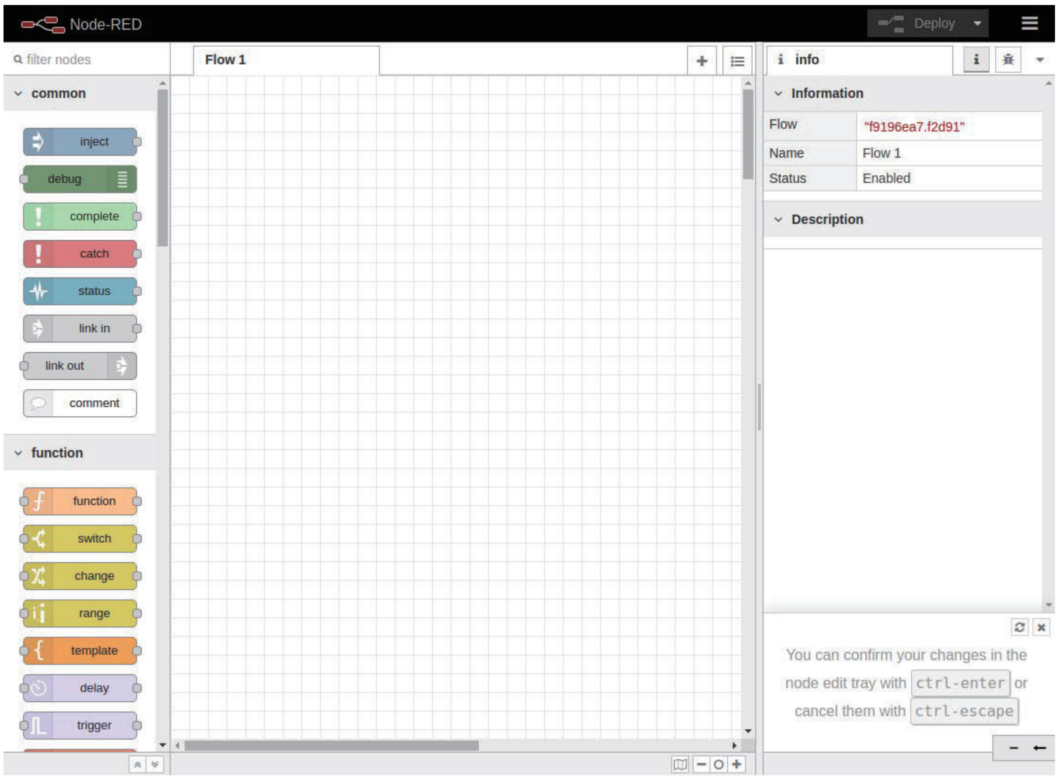


Figure 2.13 – Node-RED flow editor

And that's it! We have now covered all the installation options for each tool we'll need in order to start using Node-RED.

Summary

In this chapter, you've gotten your environment ready so that you can use the Node-RED flow editor. At this point, I believe that you can already access the Node-RED flow editor, so you'll want to learn how to use it. In the next chapter, we'll make a sample flow on it and learn about the major features of the Node-RED flow editor.

3 Understanding Node-RED Characteristics by Creating Basic Flows

In this chapter, we'll actually create a flow using Node-RED Flow Editor. By creating a simple flow, you will understand how to use the tool and its characteristics. For a better understanding, we will create some sample flows.

From now on, you will create applications called flows using Node-RED. In this chapter, you will learn how to use Node-RED and how to create an application as a flow. To do this, we will cover the following topics:

- Node-RED Flow Editor mechanisms
- Using the Flow Editor
- Making a flow for a data handling application

- Making a flow for a web application
- Importing and exporting a flow definition

By the end of this chapter, you will have mastered how to use Node-RED Flow Editor and know how to build a simple application with it.

Technical requirements

To complete this chapter, you will need the following:

- Node-RED (v1.1.0 or above).
- The code for this chapter can be found in `Chapter03` folder at `https://github.com/PacktPublishing/-Practical-Node-RED-Programming`.

Node-RED Flow Editor mechanisms

As you learned in the previous chapters, Node-RED has two logical parts: a development environment called the Flow Editor and an execution environment for executing the application that's been created there. These are called the runtime and the editor, respectively. Let's take a look at them in more detail:

- **Runtime**: This includes a Node.js application runtime. It is responsible for running the deployed flows.
- **Editor**: This is a web application where the user can edit their flows.

The main installable package includes both components, with a web server to provide Flow Editor as well as a REST Admin API for administering the runtime. Internally, these components can be installed separately and embedded into existing Node.js applications, as shown in the following diagram:

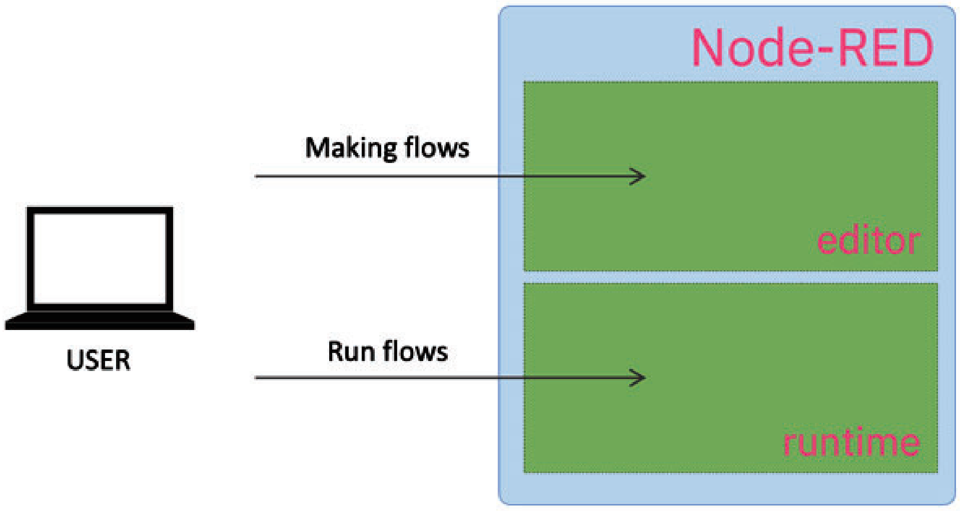


Figure 3.1 – Node-RED overview

Now that you understand the mechanisms of Node-RED, let's immediately learn how to use the Flow Editor.

Using the Flow Editor

Let's take a look at the main functions of the Flow Editor.

The main features of the Flow Editor are as follows:

- **Node**: The main building block of Node-RED applications, they represent well-defined pieces of functionality.
- **Flow**: A series of nodes wired together that represent the series of steps messages pass through within an application.
- **The panel on the left is the palette**: A collection of nodes that are available within the editor that you can use to build your application.
- **Deploy button**: Press this button to deploy your apps once you've edited them.
- **Sidebar**: A panel for displaying various functions, such as processing parameter settings, specifications, and debugger display.
- **Sidebar tabs**: Settings for each node, standard output, change management, and so on.
- **Main menu**: Flow deletion, definition import/export, project management, and so on.

These functions are arranged on the screen of the Flow Editor like so:

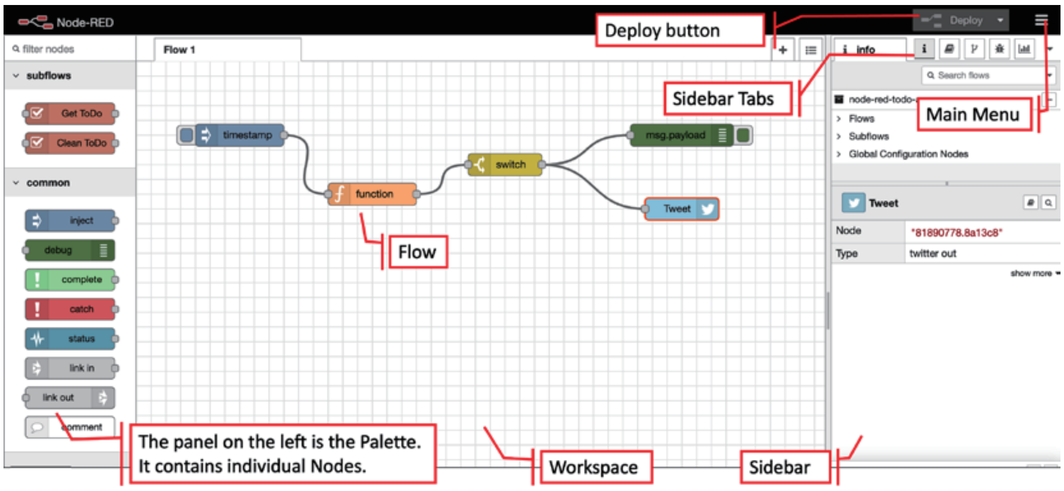


Figure 3.2 – Node-RED Flow Editor

You need to understand what is contained in the Flow menu before you start using Node-RED. Its contents may differ, depending on the version of Node-RED you're using, but it has some setting items such as **Project management of flow**, **Arrange view**, **Import / export of flow**, **Installation of node published in library**, and so on that are universal. For more information on how to use Node-RED, it's a good idea to refer to the official documentation as needed.

Important note

Node-RED User Guide: `https://nodered.org/docs/user-guide/`.

The following diagram shows all these Flow Editor menu options inside Node-RED:

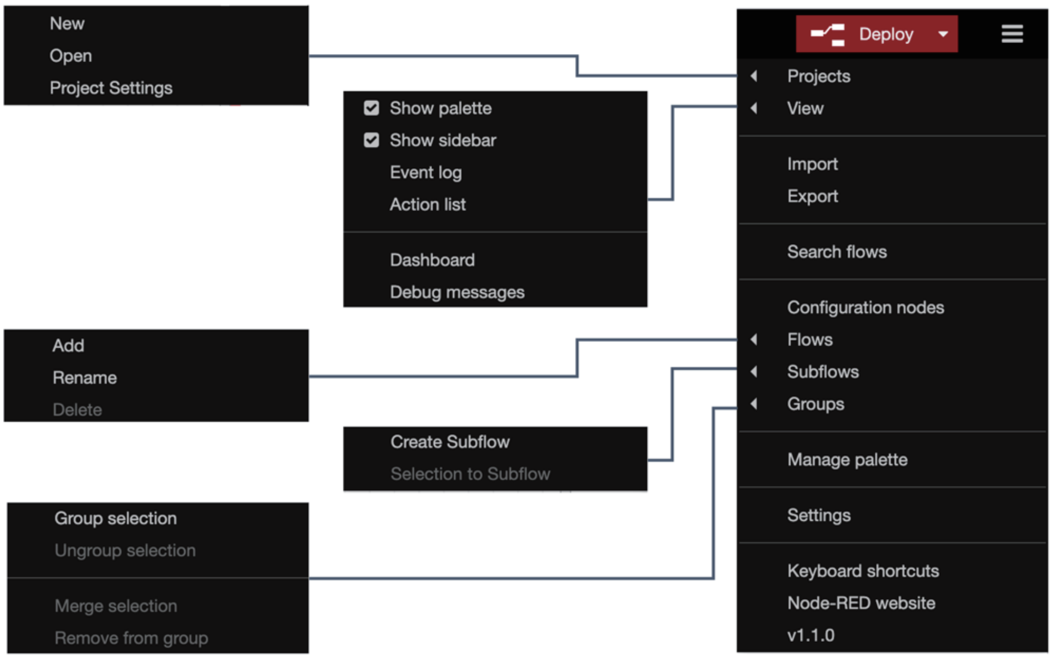


Figure 3.3 – Node-RED Flow Editor menu

With that, you are ready to use Node-RED to build an application. So, let's get started!

First of all, you need to run Node-RED in your environment. Please refer to *Chapter 2, Setting Up the Development Environment*, to learn how to set it up with your environment, such as Windows, Mac, or Raspberry Pi, if you haven't done so already.

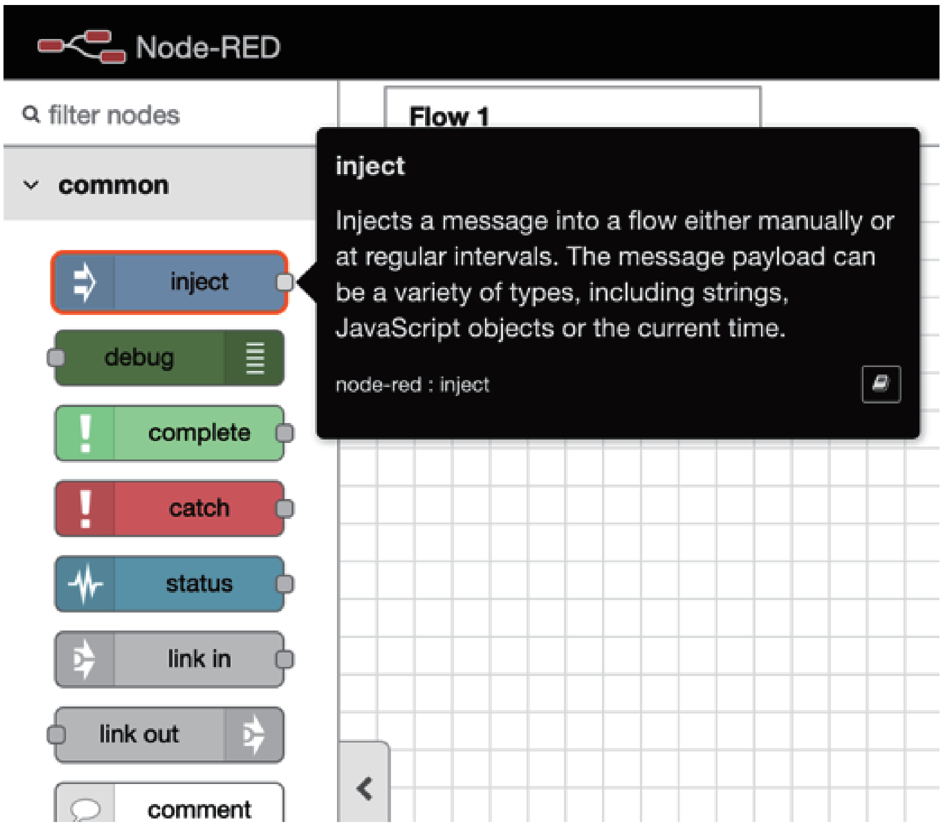
With Node-RED running, let's move on to the next section, where we'll be making our first flow.

Making a flow for a data handling application

In this section, you will create a working application (called a flow in Node-RED). Whether it is the **internet of things** (**IoT**) or server processing as a web application, the basic operation that Node-RED performs is sequentially transferring data.

Here, we'll create a flow where JSON data is generated in a pseudo manner, and the data is finally output to standard output via some nodes on Node-RED.

There are many nodes on the left-hand side of the palette. Please pay attention to the **common** categories here. You should be able to easily find the **inject** node, as shown in the following screenshot:



inject

Injects a message into a flow either manually or at regular intervals. The message payload can be a variety of types, including strings, JavaScript objects or the current time.

node-red : inject

Figure 3.4 – Inject node

This node can inject a message into the next node. Let's get started:

1. Drag and drop it onto the palette of Flow 1 (the default flow tab).

   You will see that the node is labeled with the word **timestamp**. This is because its default message payload is a timestamp value. We can change the data type, so let's change it to a JSON type.

2. Double-click the node and change its settings when the **Properties** panel of the node is opened:

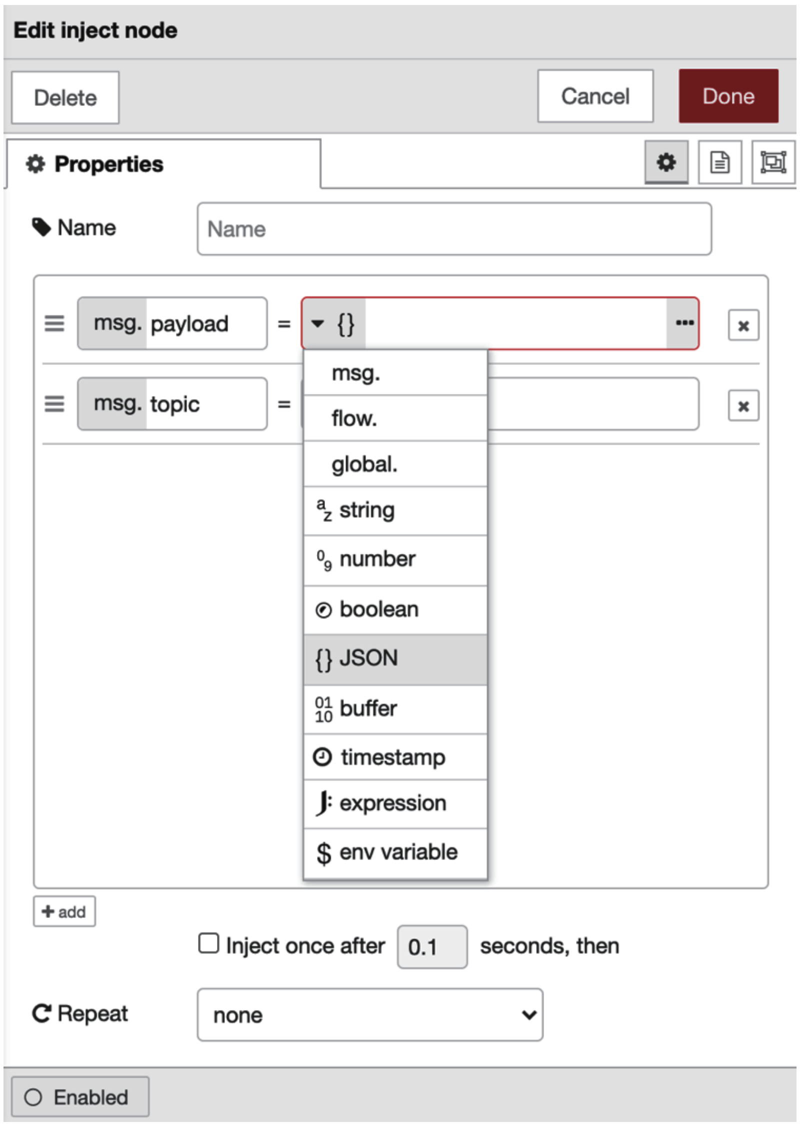
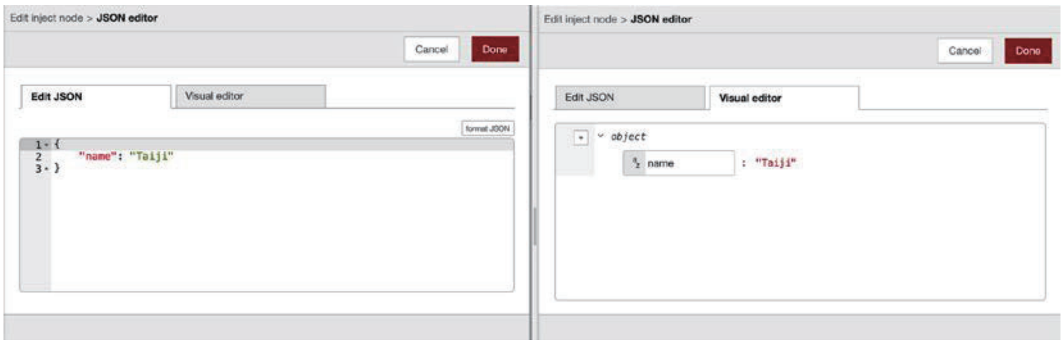


Figure 3.5 – Edit inject node Properties panel

3. Click the drop-down menu of the first parameter and select **{}JSON**. You can edit the JSON data by clicking the **[...]** button on the right-hand side.
4. Click the **[...]** button, and the JSON editor will open. You can make JSON data with a text-based editor or a visual editor.

5. This time, let's make JSON data with an item called `{"name" : "Taiji"}`. You should replace my name with your name:



```
{
    "name": "Taiji"
}
```

Figure 3.6 – JSON editor

   Great – you have successfully made some sample JSON data!

6. Click the **Done** button and close this panel.
7. Similarly, place a **Debug** node on the palette.
8. After placing it, wire the **Inject** and **Debug** nodes to it.

   Once you execute this flow, the JSON data that was passed from the **Inject** node will be output to the debug console (standard output) by the **Debug** node. You don't need to configure anything on the **Debug** node:

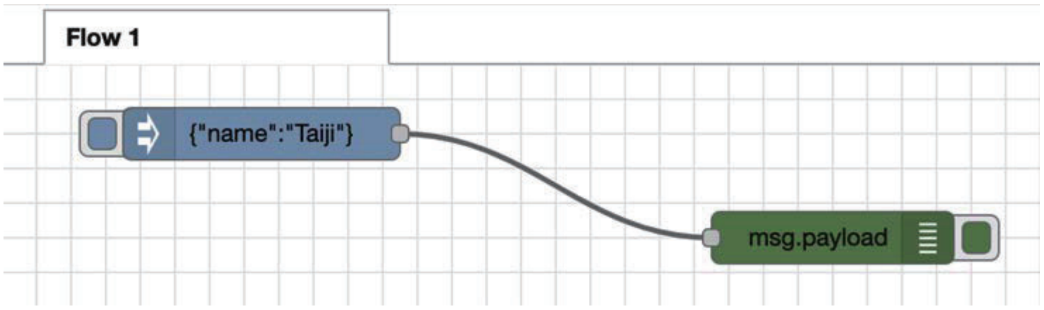


Figure 3.7 – Placing the Debug node and wiring it

9. Finally, you need to deploy the flow you created. In Node-RED Flow Editor, we can deploy all our flows on the workspace to the Node-RED runtime by clicking the **Deploy** button in the top-right corner.

10. Before running the flow, you should select the **Debug** tab from the node menu's side panel to enable the debug console, as shown in the following screenshot:

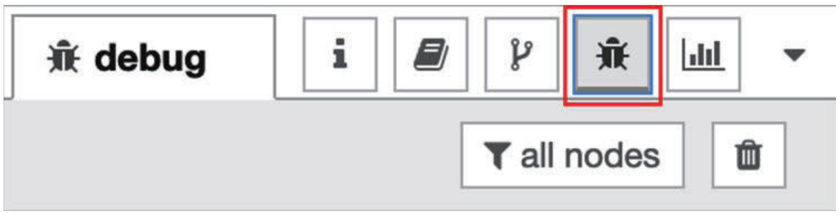
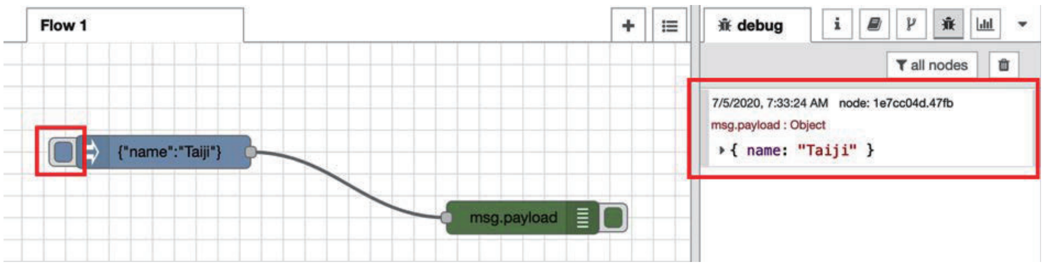


Figure 3.8 – Enabling the debug console

11. Let's run this flow. Click the switch of the **Inject** node to see the result of executing the flow on the debug console:



```
7/5/2020, 7:33:24 AM   node: 1e7cc04d.47fb
msg.payload : Object
▸{ name: "Taiji" }
```

Figure 3.9 – Executing the flow and checking the result

This is a very simple and easy data handling flow sample. In the latter half of this book, we will also experiment with data handling by actually connecting IoT devices and passing data obtained from a web API. In this section, it is enough that you understand how to handle data in Node-RED. Next, we're going to experiment with making a flow for a web application.

Making a flow for a web application

In this section, you will create a new flow for a web application. We'll create this flow in the same way we created the previous data handling flow.

You can create it in the workspace of the same flow (Flow 1), but to make things clear and simple, let's create a new workspace for the flow by following these steps:

1. Select **Flows | Add** from the **Flow** menu. Flow 2 will be added to the right-hand side of Flow 1. These flow names, such as "Flow 1" and "Flow 2," are default names that are provided upon creation. You can rename the flow so that it has a more specific name if you want to:

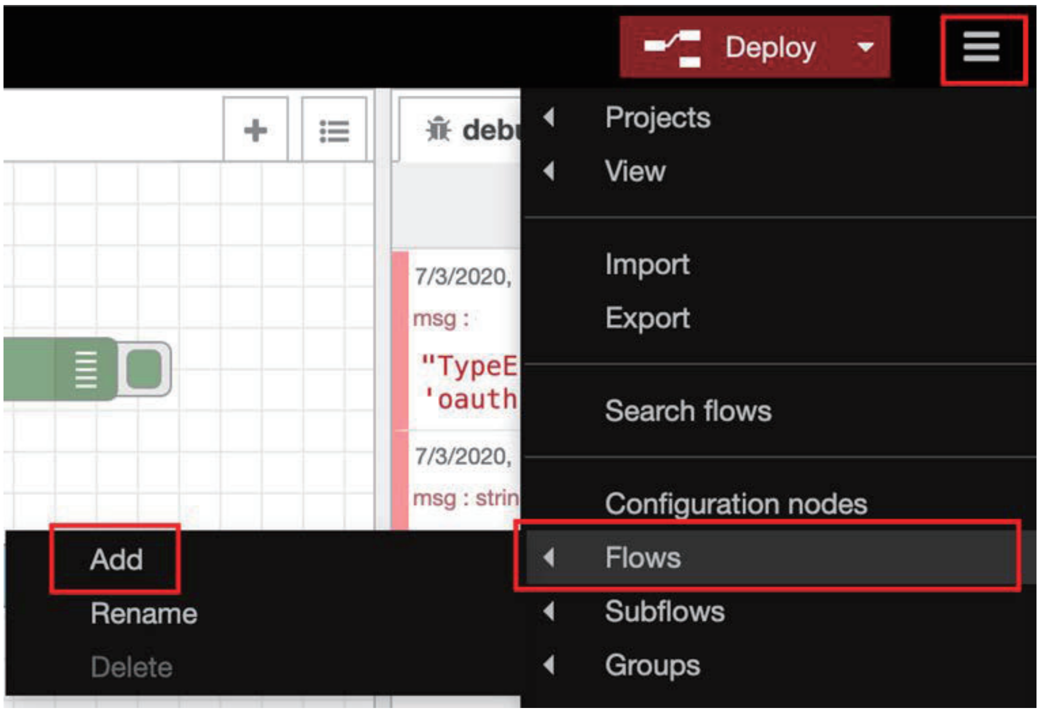
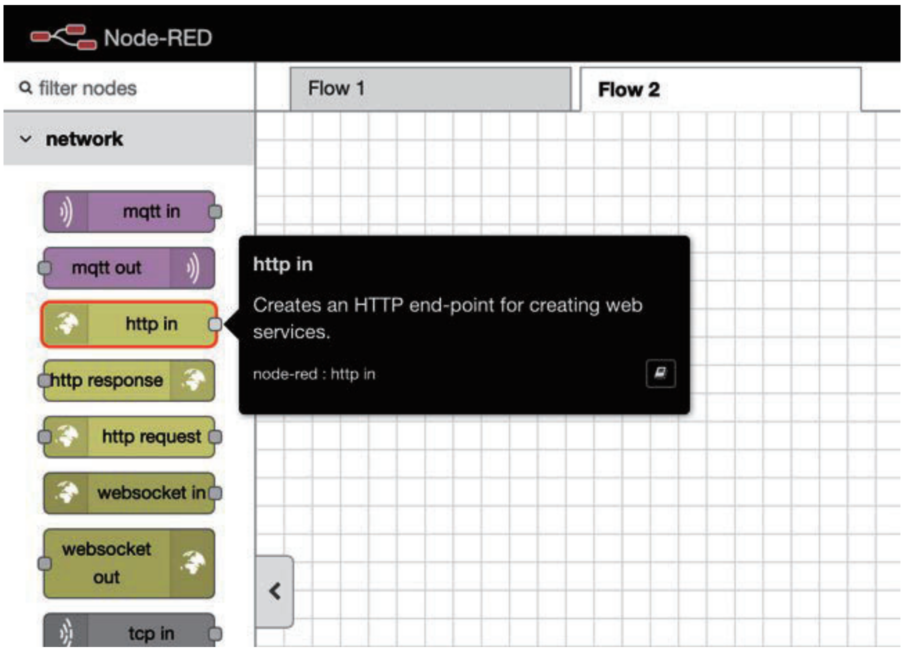


Figure 3.10 – Adding a new flow

2. Select the **http in** node from the **network** category on the palette, and then drag and drop it onto the palette of Flow 2 (the new flow tab you just added):



http in

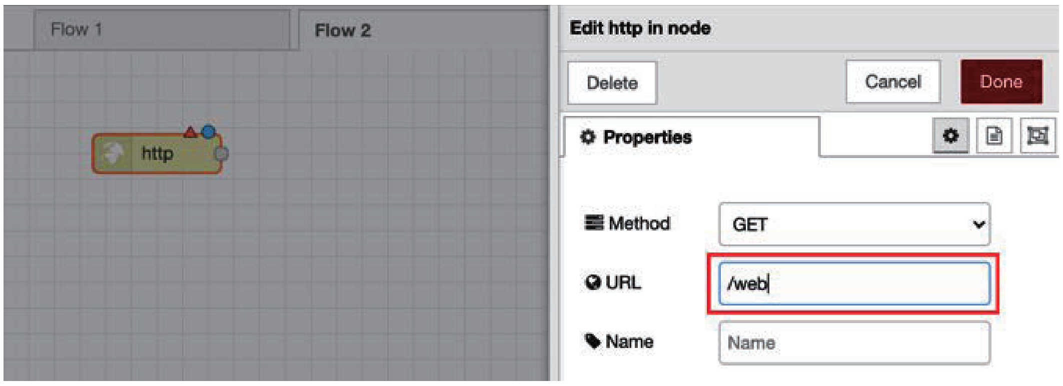
Creates an HTTP end-point for creating web services.

node-red : http in

Figure 3.11 – An http in node

3. Double-click the node to open its **Edit** dialog.
4. Enter the URL (path) of the web application you will create.

   This path will be used as part of the URL for the web application you will be creating, under the Node-RED URL. In this case, if your Node-RED URL is `http://localhost:1880/`, your web application URL will be `http://localhost:1880/web`. An example of this can be seen in the following screenshot:



Method: GET

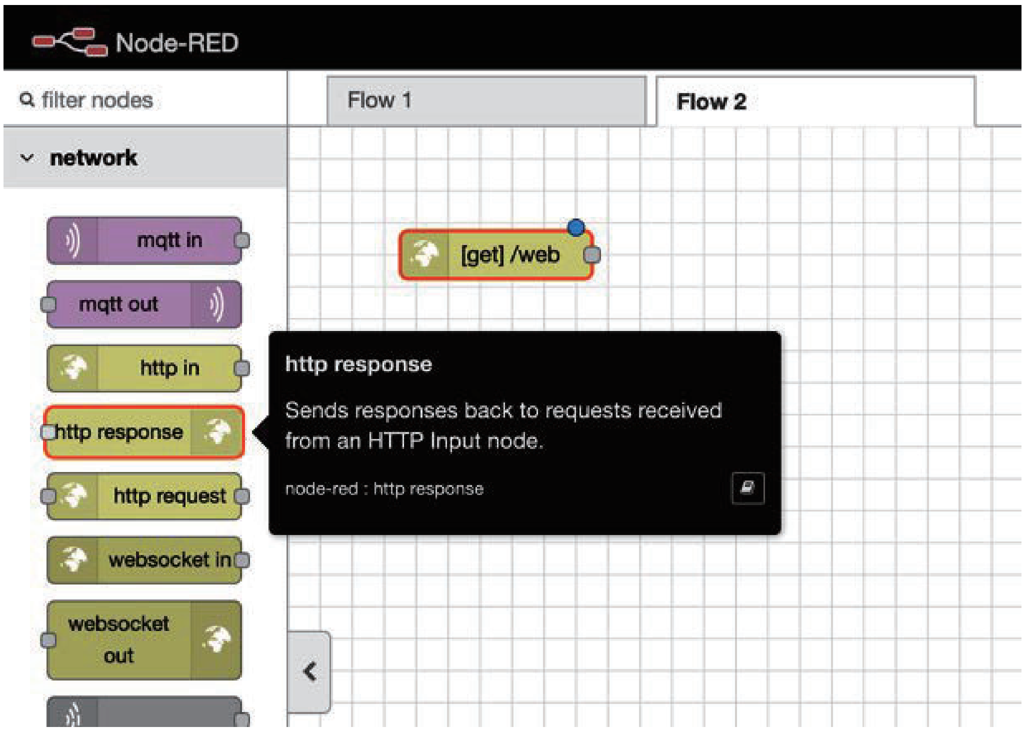
URL: /web

Name: Name

Figure 3.12 – Setting the path of the URL

5. To send a request via HTTP, an HTTP response is required. So, place an **http response** node on the workspace of your Node-RED.

   You can find this node in the **network** category of the palette, next to the **http in** node. Here, the **http response** node simply returns the response, so you don't need to open the configuration panel. You can leave it as-is. If you want to include a status code in the response message, you can do so from the **settings** panel, as shown in the following screenshot:



http response

Sends responses back to requests received from an HTTP Input node.

node-red : http response

Figure 3.13 – An http response node

6. After placing an **http response** node on the palette, add a wire from the **http in** node to the **http response** node.

This completes the flow for the web application, since we've allowed an HTTP request and response. You will see a light blue dot in the top-right corner of each node, which indicates that they haven't been deployed yet – so please make sure you click the **Deploy** button:

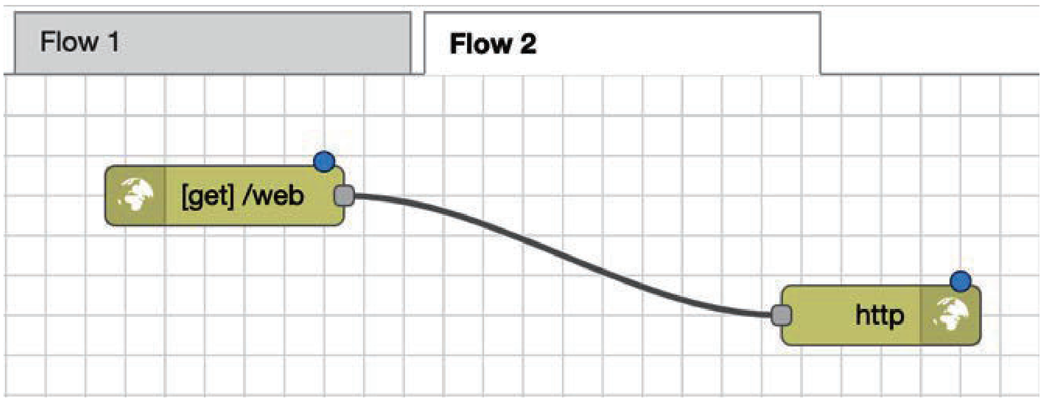
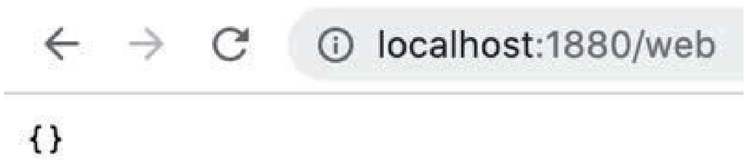


Figure 3.14 – Wired nodes

7. Once it's been successfully deployed, open a new tab in your browser.
8. Then, access the URL of the web application shown in the **http in** node section by entering `http://localhost:1880/web`.

You should find that only {} is displayed on your screen. This is not a mistake. It is a result of sending an HTTP request and returning a response to it. Right now, since we have not set the content to be passed to the response, an empty JSON is passed as message data. This looks as follows:

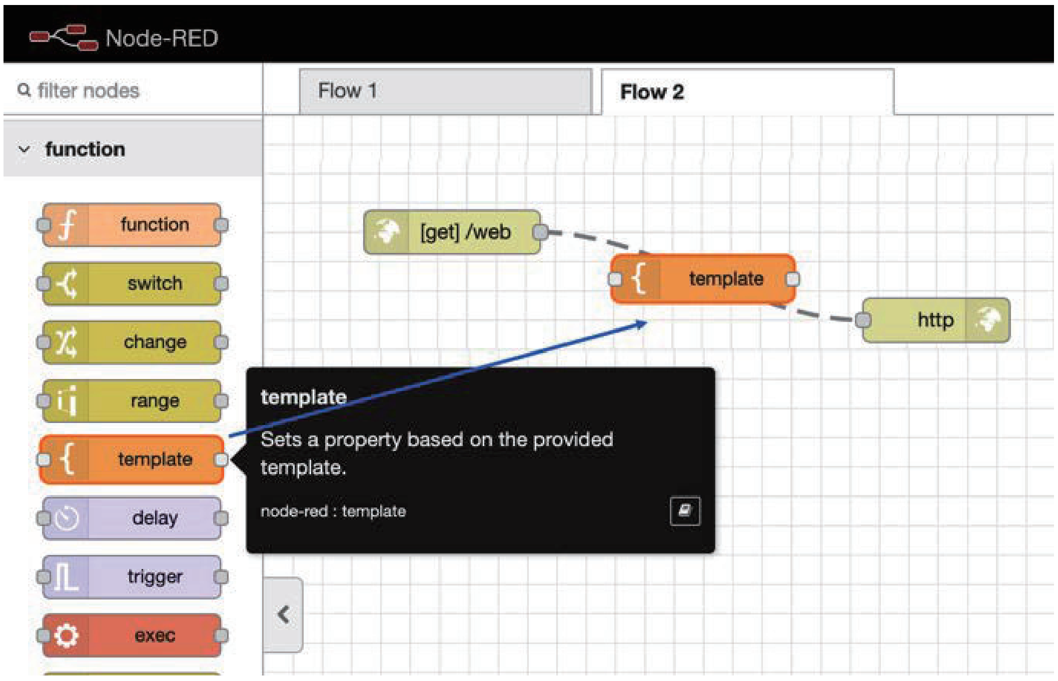


```
{}
```

Figure 3.15 – Web application result

This isn't great, so let's create some content. Let's do something very simple and implement some simple HTML code. So, where should I code this? The answer is simple. Node-RED has a template node that allows you to specify the HTML code as-is as output. Let's use this:

1. Drag and drop a **template** node between the **http in** node and the **http response** node on the wire, so that the **template** node will be connected on it:



template

Sets a property based on the provided template.

node-red : template

Figure 3.16 – Placing a "template" node on the wire between our two existing nodes

2. Next, double-click the **template** node to open the settings panel. You can code on the **Template** area of the **settings** panel. This time, use the following sample HTML. The title is specified for the head. Let's attach a page heading to the body with the `<h1>` tag. Arrange the contents resembling the menu with the `<h2>` tag. The code will look like this:

```
<html>
  <head>
    <title>Node-RED Web sample</title>
  </head>
  <body>
    <h1>Hello Node-RED!!</h1>
```

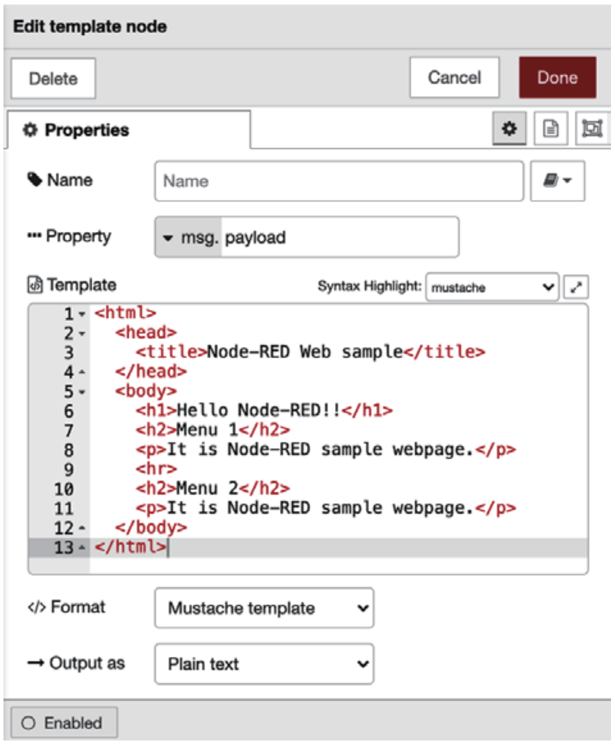
```
      <h2>Menu 1</h2>
      <p>It is Node-RED sample webpage.</p>
      <hr>
      <h2>Menu 2</h2>
      <p>It is Node-RED sample webpage.</p>
    </body>
  </html>
```

Note

You can also get this code from this book's GitHub repository at `https://github.com/PacktPublishing/-Practical-Node-RED-Programming/tree/master/Chapter03`.

3. Once you have finished editing the **template** node, click the **Done** button to close it.

   The following screenshot shows what your template node will look like as you edit it:



Name: Name

Property: msg. payload

Template — Syntax Highlight: mustache

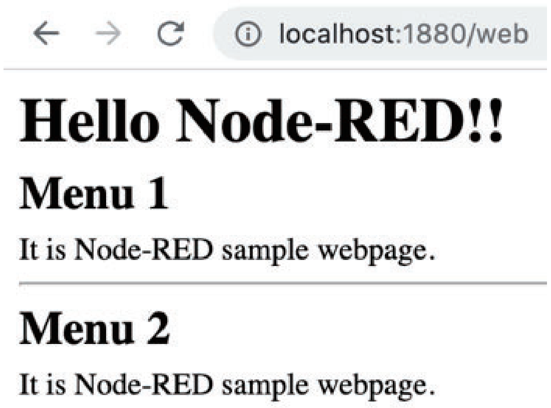
```
<html>
  <head>
    <title>Node-RED Web sample</title>
  </head>
  <body>
    <h1>Hello Node-RED!!</h1>
    <h2>Menu 1</h2>
    <p>It is Node-RED sample webpage.</p>
    <hr>
    <h2>Menu 2</h2>
    <p>It is Node-RED sample webpage.</p>
  </body>
</html>
```

Format: Mustache template

Output as: Plain text

Figure 3.17 – Code in the Template area

With that, we have finished preparing the HTML to be shown on our page. Please make sure you click the **Deploy** button. Access the web page by going to `http://localhost:1880/web` once more. You should now see the following output:



Hello Node-RED!!

Menu 1

It is Node-RED sample webpage.

Menu 2

It is Node-RED sample webpage.

Figure 3.18 – Web application result

At this point, you should understand how to make a web application on Node-RED. I imagine it has been nice and easy so far. Now that we have built up some momentum, let's continue learning. In the next section, we will import and export the flow definition that we have created.

Importing and exporting a flow definition

In this section, you will import and export the flow definition you have created. Usually, when developing, it is necessary to back up the source code and version control. You may also import source code created by others, or export your own source code and pass it on to others. Node-RED has a similar concept. In Node-RED, it is a normal practice to import and export the flow itself instead of importing or exporting the source code (for example, the template node described previously).

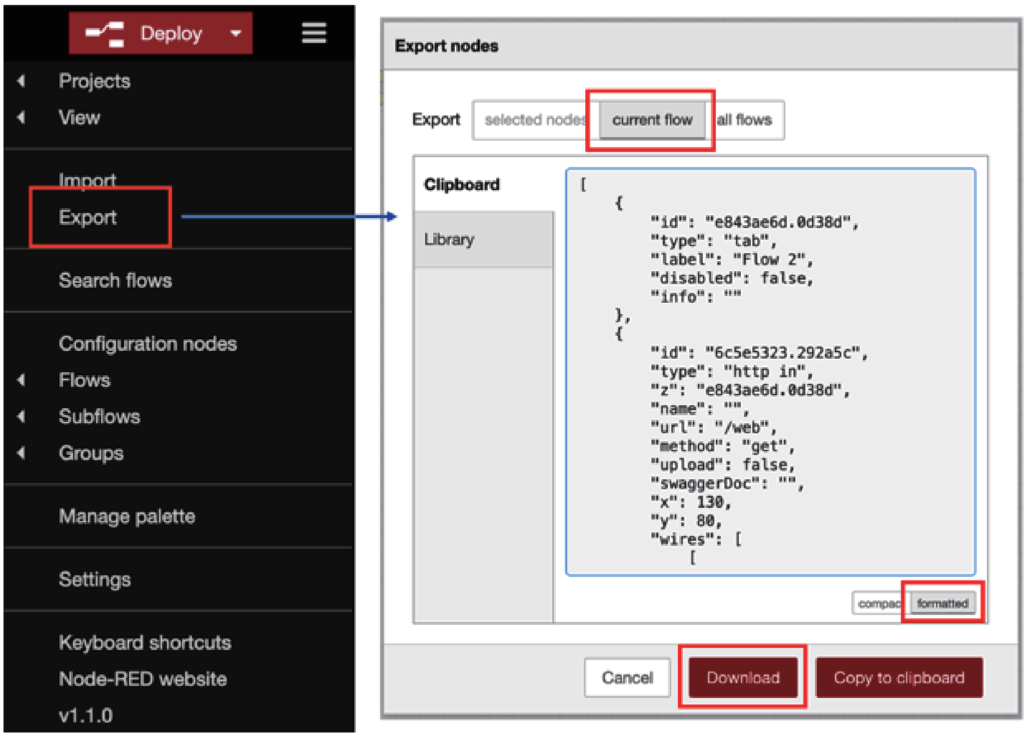
So, first, let's export the flow we have created so far. This is easy to do:

1. Simply select **Export** from the **Edit** dialog under the **Main** menu of the Node-RED Flow Editor.

   When the **Export** menu is displayed, you can only select the current flow or all your flows. You can also select raw JSON, without indentation, or formatted JSON, with indentation.

2. Here, select the current flow and select **Formatted**.

3. Now, you can select how to save the exported JSON data – **Copy to clipboard** or **Download**. Here, we'd want to download the JSON data, so click the **Download** button:



```
[
    {
        "id": "e843ae6d.0d38d",
        "type": "tab",
        "label": "Flow 2",
        "disabled": false,
        "info": ""
    },
    {
        "id": "6c5e5323.292a5c",
        "type": "http in",
        "z": "e843ae6d.0d38d",
        "name": "",
        "url": "/web",
        "method": "get",
        "upload": false,
        "swaggerDoc": "",
        "x": 130,
        "y": 80,
        "wires": [
            [
```

Figure 3.19 – Export operation

You will see a file called `flows.json` in the downloads location of your machine.

4. Open this file in a text editor so that you can check the contents of the JSON file.

With that, we have learned how to export.

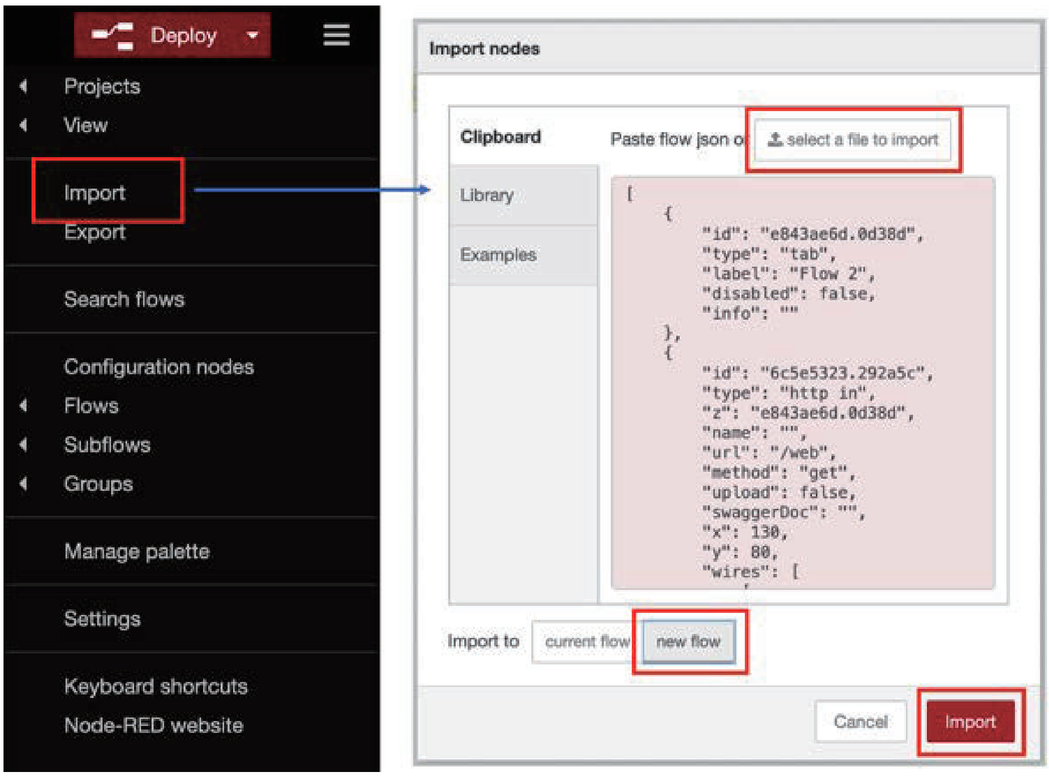
Next, we need to import this definition (`flows.json`) into our Node-RED Flow Editor. Do this by following these steps:

1. Simply select **Import** from the **Flow** menu in the Node-RED Flow Editor.

   When the **Import** menu is displayed, you can select **Paste flow json** or **Select a file based import**. You can also select a **current flow** or a **new flow** from the flow tab. If you select **new flow**, a new flow tab will be added automatically.

2. Here, please choose **Select a file based import** and import to **new flow**. Then, pick the JSON file called `flows.json` you exported to your local machine.

3. Once the file has loaded, click the **Import** button:



```
[
    {
        "id": "e843ae6d.0d38d",
        "type": "tab",
        "label": "Flow 2",
        "disabled": false,
        "info": ""
    },
    {
        "id": "6c5e5323.292a5c",
        "type": "http in",
        "z": "e843ae6d.0d38d",
        "name": "",
        "url": "/web",
        "method": "get",
        "upload": false,
        "swaggerDoc": "",
        "x": 130,
        "y": 80,
        "wires": [
```

Figure 3.20 – Import operation

4. You now have the new tab, named Flow 2, next to the same flow on the old Flow 2 tab. It has been imported completely, but it hasn't been deployed yet, so click the **Deploy** button, as follows:

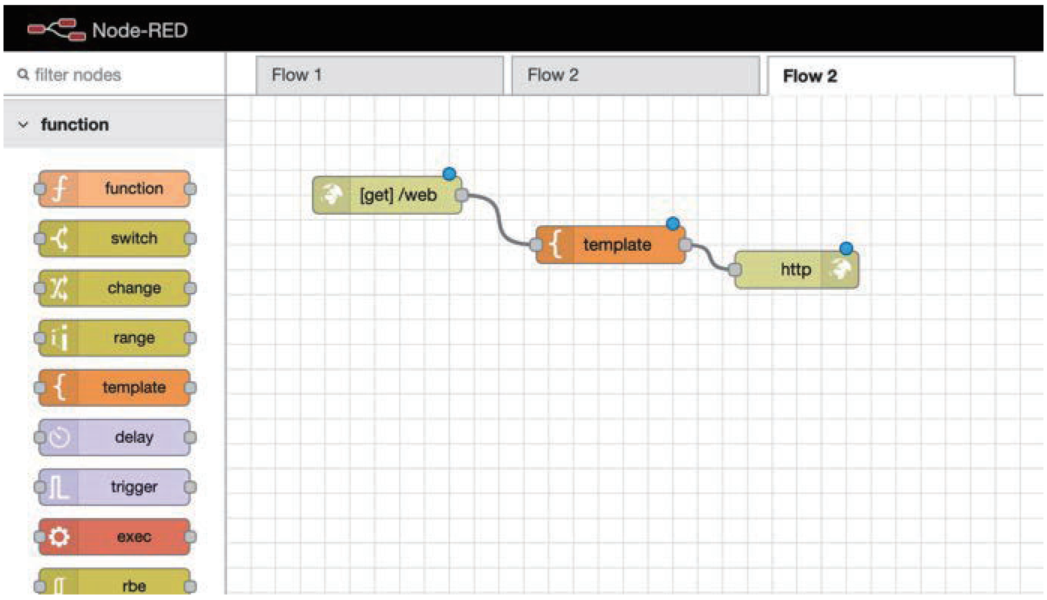
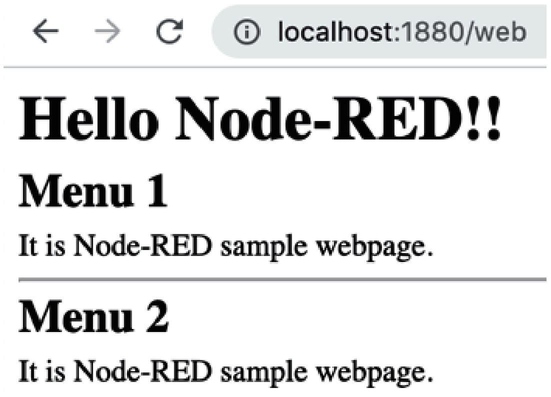


Figure 3.21 – Adding the new flow

With that, we've successfully prepared what will be shown on our web page using the flow we imported. Please make sure you click **Deploy** button.

5. Access the web page again by going to `http://localhost:1880/web`.

Here, you will see that this web page has the same design as the web page you exported. Great work!


localhost:1880/web

Hello Node-RED!!

Menu 1

It is Node-RED sample webpage.

Menu 2

It is Node-RED sample webpage.

Figure 3.22 – Result of the web application

Now, let's wrap this chapter up.

Summary

In this chapter, you learned how to use Node-RED Flow Editor to make basic flows and import/export flows. Now that you know how to use Node-RED Flow Editor, you'll want to learn about more of its features. Of course, Node-RED doesn't only have basic nodes such as **Inject**, **http**, and **template**, but also more attractive nodes such as **switch**, **change**, **mqtt**, and **dashboard**. In the next chapter, we'll try to use several major nodes so that we can code JavaScript, catch errors, perform data switching, delay functions, use the CSV parser, and more.

4
Learning the Major Nodes

In this chapter, you will learn about the major nodes used in Node-RED. Node-RED, which is an open source project, provides some major nodes by default, but it is possible to import and use nodes from the public library as required.

Node-RED has a lot of nodes. Therefore, this book is not sufficient to explain all of them. So, in this chapter, let's pick up the main nodes and most commonly used basic nodes and learn how to use them, exploring these topics in this chapter:

- What is a node?
- How to use nodes
- Getting various nodes from the library

By the end of this chapter, you will have mastered how to use major nodes in the Node-RED flow editor.

Technical requirements

To progress in this chapter, you will need the following technical requirements:

- Node-RED (v1.1.0 or above).
- The code used in this chapter can be found in `Chapter04` folder at `https://github.com/PacktPublishing/-Practical-Node-RED-Programming`.

What is a node?

Let's first understand what exactly a node is in Node-RED.

Node-RED is a tool for programming Node.js applications with **Graphical User Interface** (**GUI**) tools. Node-RED also serves as an environment for executing software (Node-RED Flow) programmed on Node-RED.

Normally, when programming with Node.js, the source code is written with a code editor or **Integrated Development Environment** (**IDE**). An executable file is generated by building the written source code (compiling, associating with dependency files, and so on).

Visual programming on Node-RED basically follows the same process. The difference is that the coding part is the act of placing the node on Node-RED instead of the editor.

In Node-RED, the basic processing used when programming with Node.js is provided by implemented parts called nodes. In normal object-oriented programming, these parts may often be provided as library files in the form of common parts.

Since Node-RED is a GUI-based visual programming tool, these common parts are more than just library files. These common parts are shaped like boxes and are called nodes in Node-RED. Also, except for some nodes, generally nodes can set the things that can be variables (arguments, parameters, and so on) as node properties when programming.

In other words, since there are already programmed parts (nodes), programming is completed simply by placing them in the GUI. The following figure compares pure Node.js programming with flow creation in Node-RED:

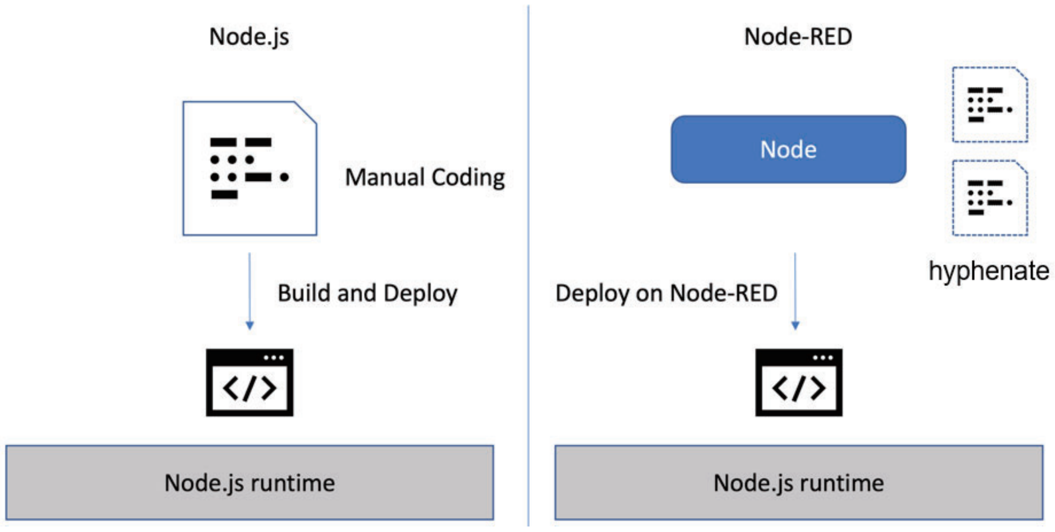


Figure 4.1 – Node-RED versus Node.js programming

Now that you understand the concepts of Node-RED and nodes, let's take a closer look at nodes.

As you can see when you start Node-RED, the basic processing nodes are provided in the Node-RED flow editor by default. This is called a **pre-installed node**.

The following are typical categories of pre-installed nodes:

- **Common**: This includes nodes that inject specific data into the flow, nodes that judge the processing status, and nodes that output logs for debugging.
- **Function**: This includes nodes that can write directly in JavaScript and HTML, nodes that convert parameter variables, and nodes that make conditional branches depending on the contents of those parameters.
- **Network**: This includes nodes that handle the protocol processing required for communication, such as MQTT, HTTP, and WebSockets.

Of course, the examples given here are just a few. There are actually many more categories and nodes.

Important note

The pre-installed nodes also depend on the Node-RED version. It's a good idea to check the official documentation for information on your Node-RED version: `https://nodered.org/docs/`.

Nodes are arranged like parts on the Node-RED flow editor and can be used simply by connecting up the wiring. As mentioned earlier, you don't have to code it yourself, except for some nodes.

Basically, the flow editor has the appearance of a box and has a settings window inside it. In the settings window, you can set the required parameters and configurations for each node:

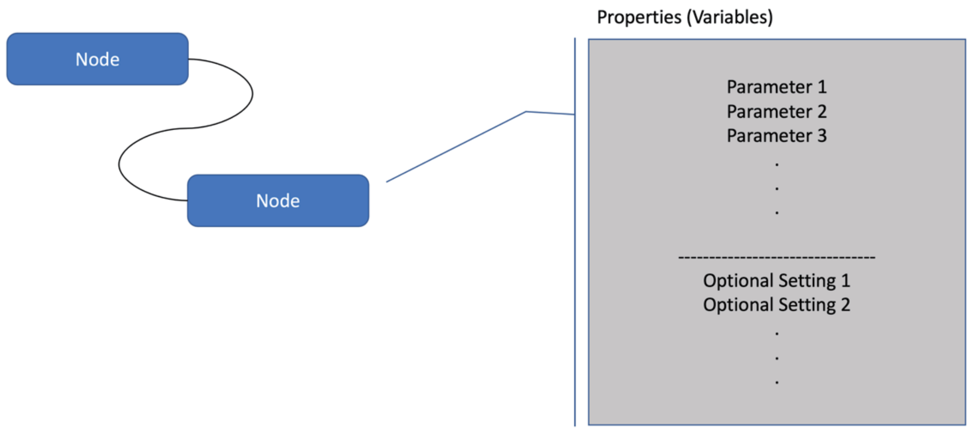


Figure 4.2 – Nodes

That's all the concepts you need to know about nodes. In the next section, you will learn how to actually use nodes.

How to use nodes

In this section, we will learn how to use nodes.

Visual programming in Node-RED is a little different from other visual programming tools because it uses flow-based programming. But rest assured, it's not difficult at all. If you actually create a few simple flows, you should be able to master how to use nodes in Node-RED.

So, let's now create a sample flow using some typical preinstalled nodes. The environment is the same for Raspberry Pi, Windows, and macOS systems. Please use your favorite environment.

Common category

Let's introduce the nodes that we'll use to make our flow. You can pick all of the nodes up and place them on the palette from the common category.

Create a sample flow with nodes in the common category. The following four nodes are used:

- The **inject** node
- The **complete** node
- The **catch** node
- The **debug** node

Place and wire up the nodes as shown in the following figure:

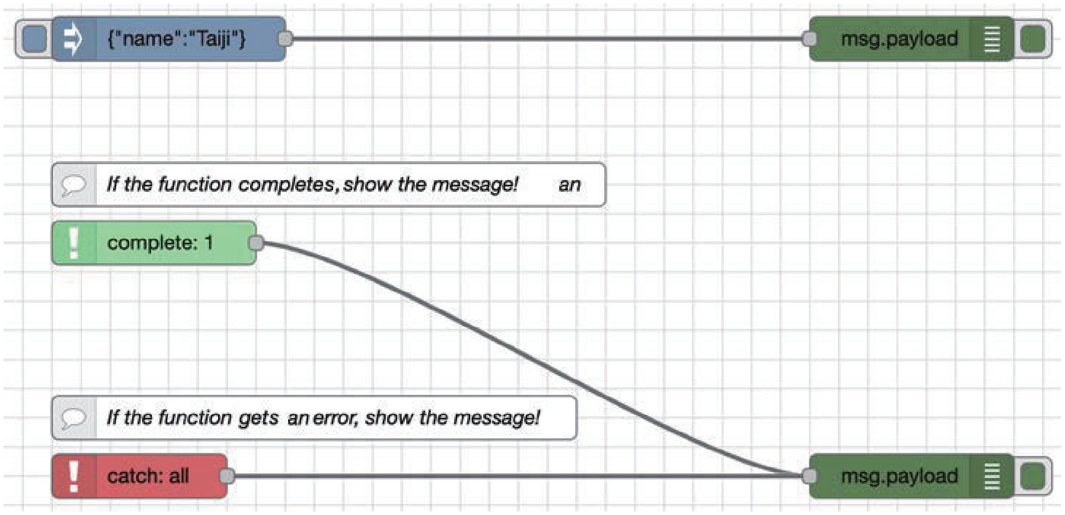

{"name":"Taiji"}
msg.payload
If the function completes, show the message! an
complete: 1
If the function gets an error, show the message!
catch: all
msg.payload

Figure 4.3 – The flow with our common category nodes

The data in the **inject** node is simple JSON data here. Double-click the placed **inject** node to open the settings panel and set the JSON data. Please refer to the following:

```
{"name":"Taiji"}
```

You can change the JSON data in the **inject** node for what you want to send. Also, you should set the properties for the **complete** node. Open the settings panel and set a node to watch the status.

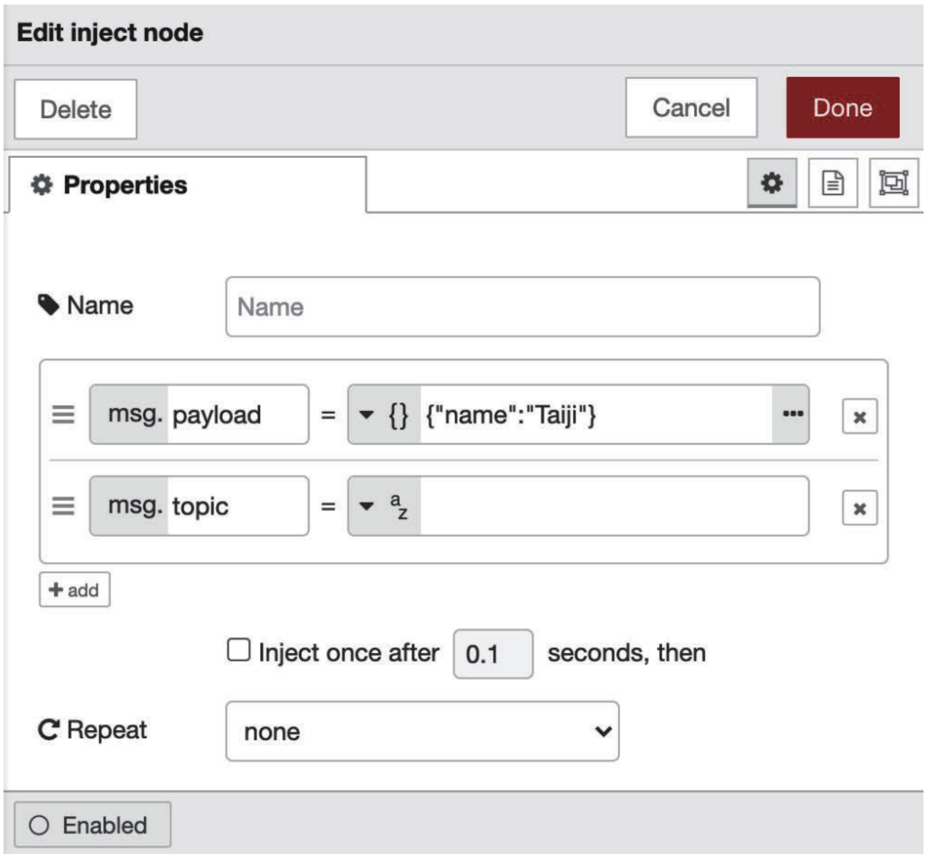
Set each node's parameters as follows:

- The **inject** node:

  Please set the first parameter as `msg.payload` with the following JSON:

  ```
  {"name": "Taiji"}
  ```

  You can set any value here:



Edit inject node

Name: Name

msg.payload = {} {"name":"Taiji"}

msg.topic = az

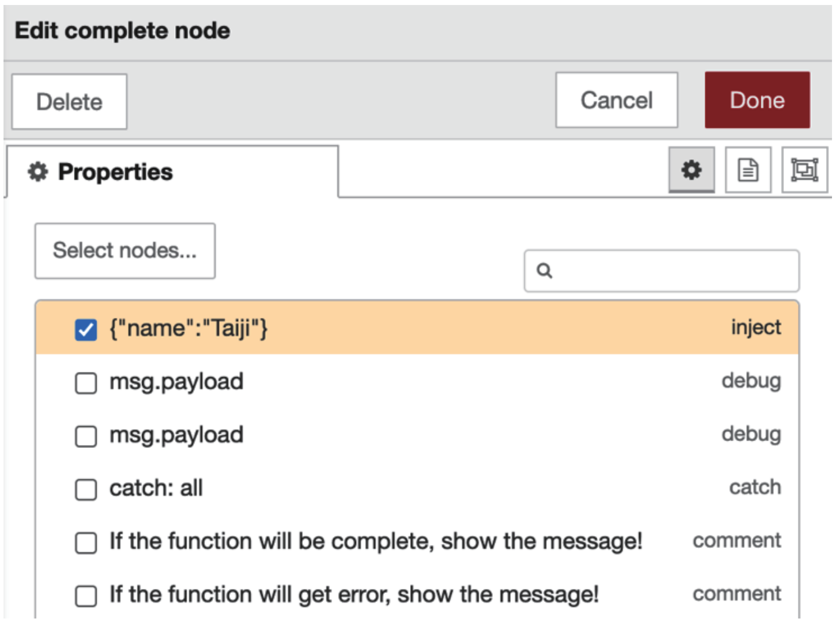
Inject once after 0.1 seconds, then

Repeat: none

Figure 4.4 – An inject node for inserting data

- The **complete** node:

  Check the first option of the **Properties** tab to watch the status of the **inject** node:



✓ {"name":"Taiji"}	inject
☐ msg.payload	debug
☐ msg.payload	debug
☐ catch: all	catch
☐ If the function will be complete, show the message!	comment
☐ If the function will get error, show the message!	comment

Figure 4.5 – A complete node for watching the status

No properties of other nodes need to be changed.

After the setting changes, you need to deploy and click the button of the **inject** node. After that, you can see the JSON data in the right-hand panel of the **debug** tab.

You can get the flow definition from the book's GitHub repo at `https://github.com/PacktPublishing/-Practical-Node-RED-Programming/blob/master/Chapter04/common-flows.json`.

Function category

In this section, we will learn how to use some major nodes from the function category, and will make a flow with these nodes.

Create a sample flow using the nodes in the function category. Here, we will use the following six nodes:

- The **inject** node
- The **function** node
- The **switch** node
- The **change** node
- The **template** node

- The **debug** node

Place and wire the nodes as shown in the following figure:

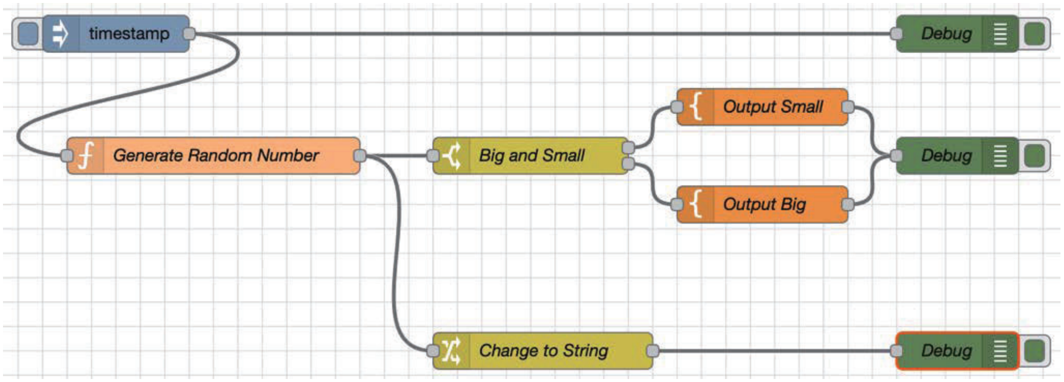


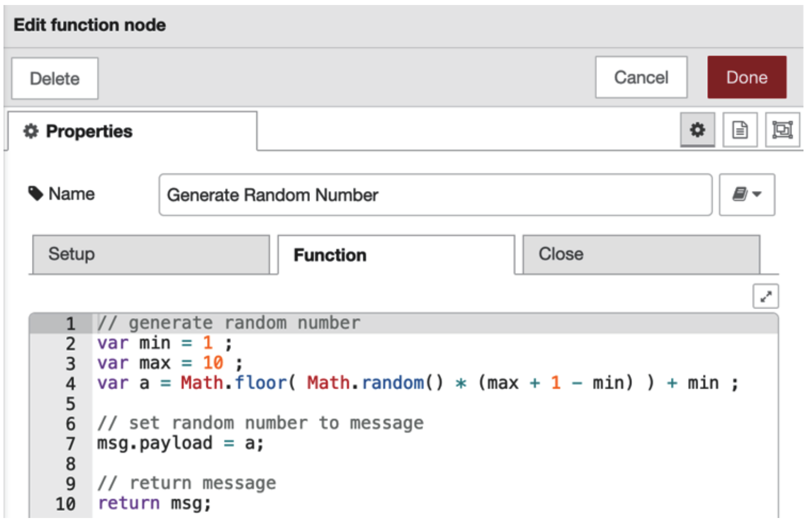
Figure 4.6 – The flow with function category nodes

Please follow these steps to make the flow:

1. Place the **inject** node and **debug** node on the palette. These two nodes can be used with their default parameters. No change of settings is required here.
2. Place a **function** node on the palette.
3. Open the settings panel of the **function** node and enter the following code:

```
// generate random number
var min = 1 ;
var max = 10 ;
var a = Math.floor( Math.random() * (max + 1 - min) ) +
  min ;

// set random number to message
msg.payload = a;

// return message
return msg;
```

4. After coding, click on **Done** to save the settings:



Name: Generate Random Number

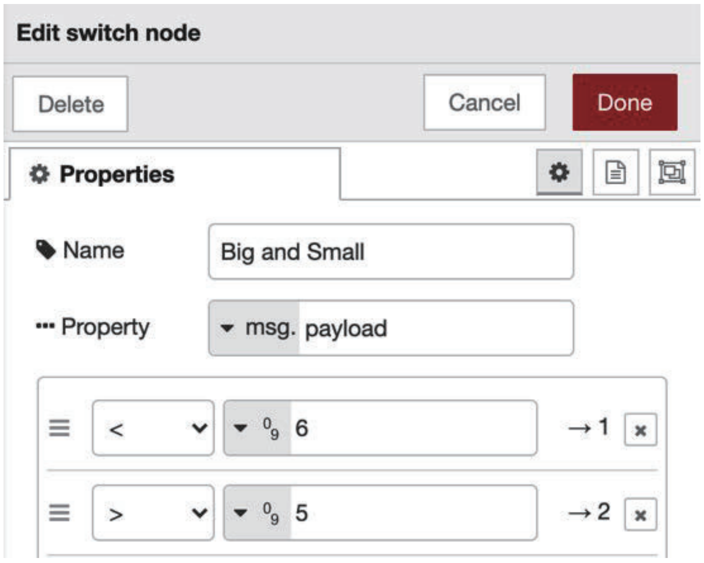
```
// generate random number
var min = 1 ;
var max = 10 ;
var a = Math.floor( Math.random() * (max + 1 - min) ) + min ;

// set random number to message
msg.payload = a;

// return message
return msg;
```

Figure 4.7 – Function node settings

5. Place the **switch** node on the palette, then open the settings panel of the **switch** node and set the value rules as follows:

- The < field: 6
- The > field: 5

This should look as follows:



Name: Big and Small

Property: msg. payload

```
<   6   →1
>   5   →2
```

Figure 4.8 – The switch node settings

If the input parameter is `5` or less, the output route is `1`, and if the input parameter is `6` or more, the output route is `2`. This means that the next node depends on the number of input parameters.

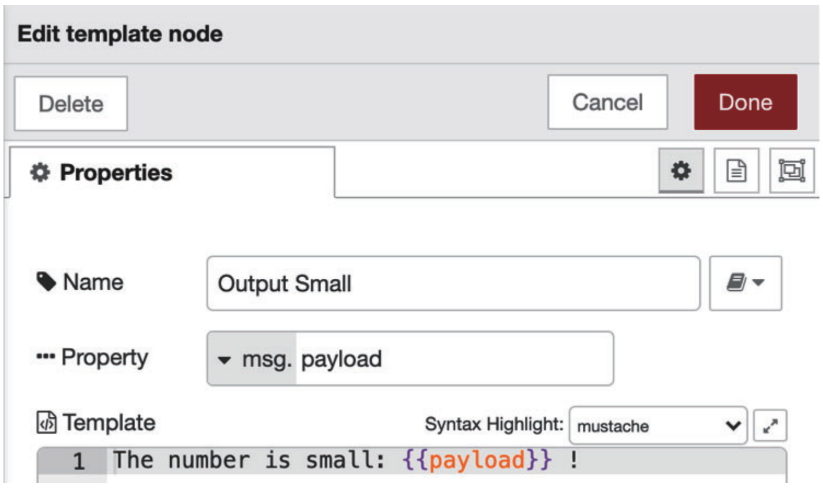
6. Place two **template** nodes on the palette.

   The previous function was the **switch** node, so the data splits depending on the result of the output.

7. Open the settings panel of each **template** node and enter the following code for the first **template** node connected to output route `1` of the **switch** node:

```
The number is small: {{payload}} !
```

   The **template** node will look something like the following screenshot once we add the preceding code:



Name: Output Small
Property: msg. payload
Template
Syntax Highlight: mustache

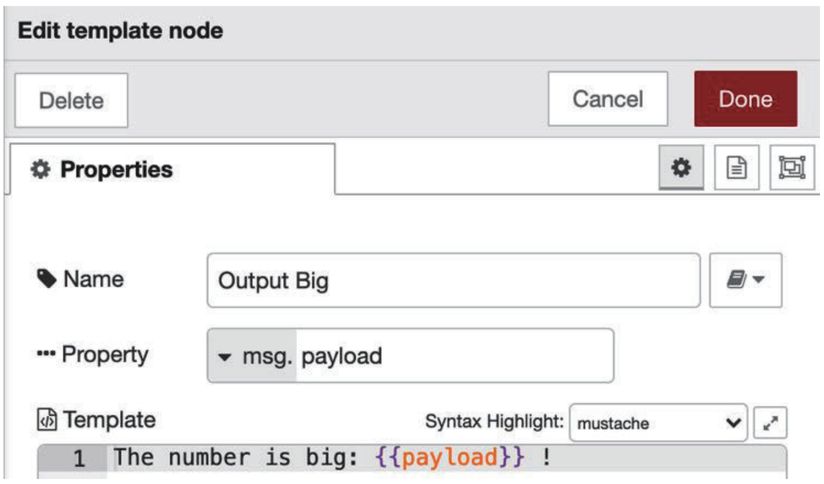
```
The number is small: {{payload}} !
```

Figure 4.9 – The first template node settings

8. Enter the following code for the second **template** node, which is connected to output route `2` of the **switch** node:

```
The number is big: {{payload}} !
```

   It will look something like the following screenshot:



Name: Output Big

Property: msg. payload

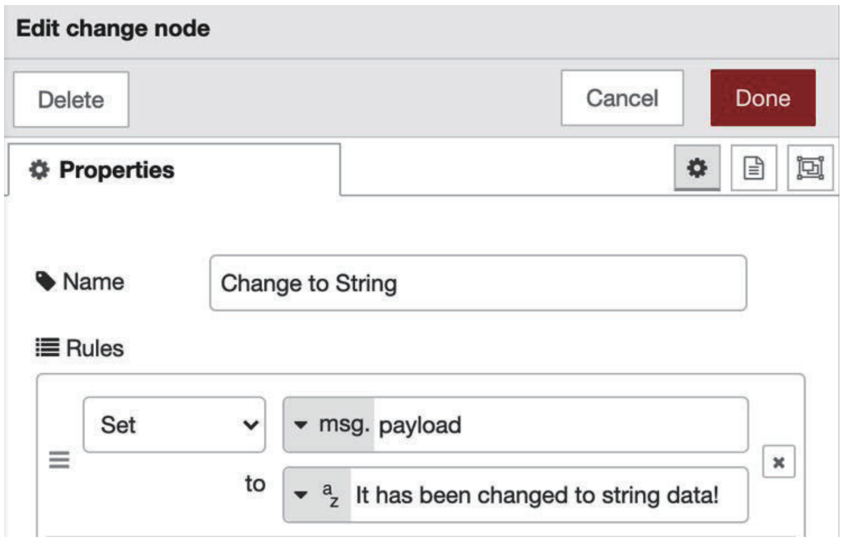
Template

Syntax Highlight: mustache

```
The number is big: {{payload}} !
```

Figure 4.10 – The second template node settings

9. Place the **change** node on the palette, open the settings panel of the **change** node, and look at the settings box below **Rules**.
10. Select **string** from the drop-down menu in the box next to **to** and enter the desired character string in the text box next to this. Here, it says **It has been changed to string data!**. Please refer to the following screenshot:



Name: Change to String

Rules

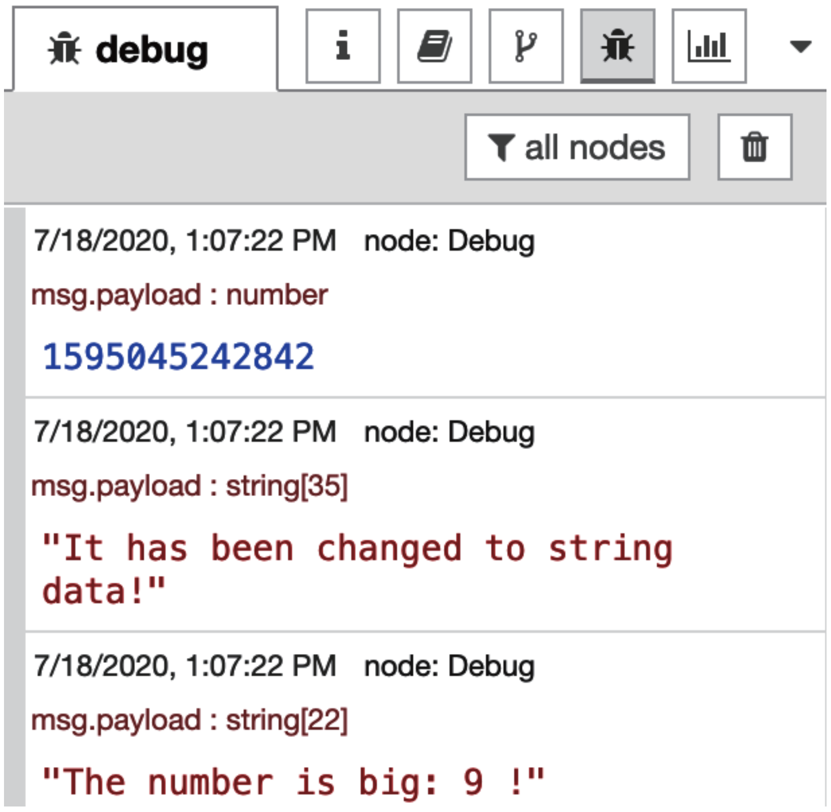
Set msg. payload

to It has been changed to string data!

Figure 4.11 – The change node settings

11. After changing the settings, you need to deploy and click the button of the **inject** node.

Once you do this, you can see the data in the debug tab in the right-hand panel, as follows:



```
7/18/2020, 1:07:22 PM   node: Debug
msg.payload : number
1595045242842

7/18/2020, 1:07:22 PM   node: Debug
msg.payload : string[35]
"It has been changed to string
data!"

7/18/2020, 1:07:22 PM   node: Debug
msg.payload : string[22]
"The number is big: 9 !"
```

Figure 4.12 – Showing the results in the debug tab

The first debug message is the default **inject** node value as a timestamp. The second one is the debug message of the **debug** node placed after the **change** node. The last one depends on the random number and is formatted by the **template** node.

You can get the flow definition from the book's GitHub repo at `https://github.com/PacktPublishing/-Practical-Node-RED-Programming/blob/master/Chapter04/function-flows.json`.

Next, let's learn about nodes that are not provided by default.

Getting several nodes from the library

You can get several more attractive nodes that have been developed by Node-RED contributors and install them in your Node-RED flow editor. You can find new nodes, share your flows, and see what other people have done with Node-RED. In this section, we will learn how to get several other nodes from the Node-RED library. Let's first access the Node-RED library site: `https://flows.nodered.org/`. In the following screenshot, you can see how the Node-RED library looks:

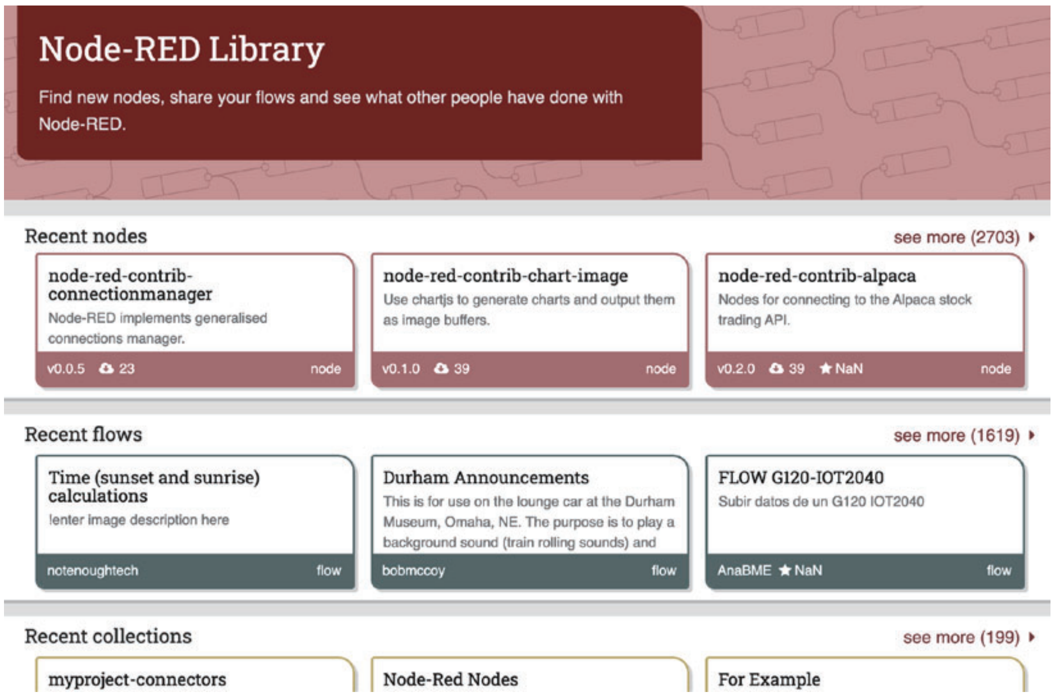
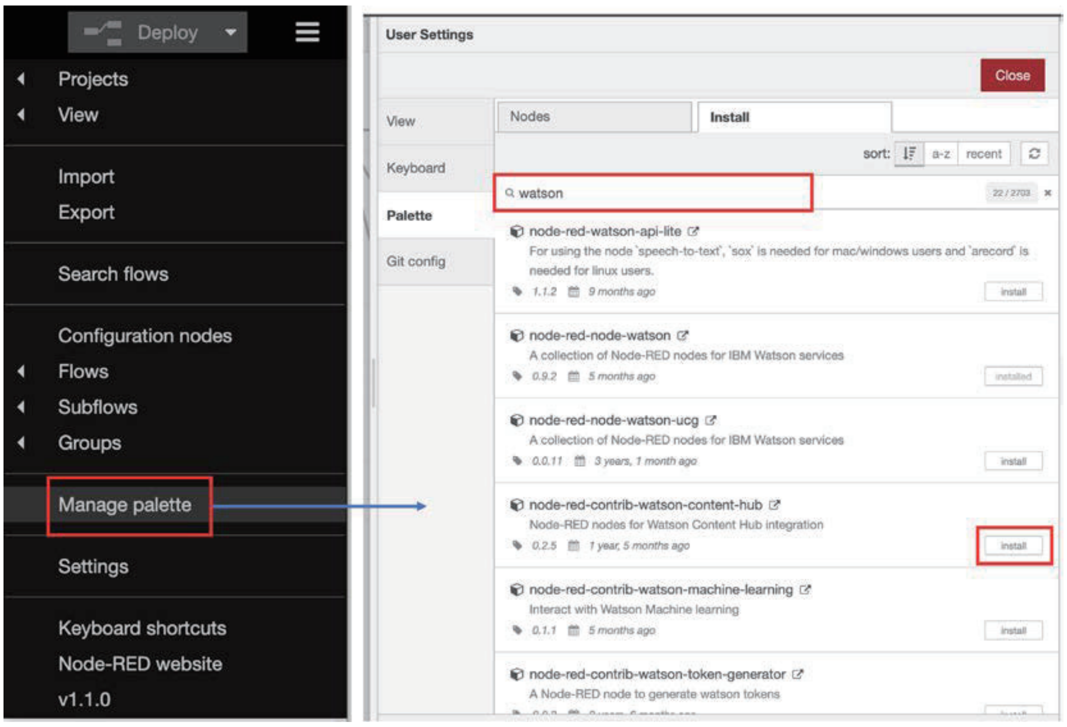

Node-RED Library
Find new nodes, share your flows and see what other people have done with Node-RED.
Recent nodes
see more (2703)
node-red-contrib-connectionmanager
Node-RED implements generalised connections manager.
v0.0.5 23 node
node-red-contrib-chart-image
Use chartjs to generate charts and output them as image buffers.
v0.1.0 39 node
node-red-contrib-alpaca
Nodes for connecting to the Alpaca stock trading API.
v0.2.0 39 NaN node
Recent flows
see more (1619)
Time (sunset and sunrise) calculations
!enter image description here
notenoughtech flow
Durham Announcements
This is for use on the lounge car at the Durham Museum, Omaha, NE. The purpose is to play a background sound (train rolling sounds) and
bobmccoy flow
FLOW G120-IOT2040
Subir datos de un G120 IOT2040
AnaBME NaN flow
Recent collections
see more (199)
myproject-connectors
Node-Red Nodes
For Example

Figure 4.13 – Node-RED Library

It's easy to use this library in your own Node-RED environment's flow editor. Let's see how to install a node from the library:

1. Select **Manage palette** from the sidebar menu. You will see the **User Settings** panel open with the **Palette** tab selected.
2. Type `watson` in the search field, or the name of any other node you want to use. If you find the node you want, click the **Install** button:



node-red-watson-api-lite
For using the node `speech-to-text`, `sox` is needed for mac/windows users and `arecord` is needed for linux users.
1.1.2 9 months ago

node-red-node-watson
A collection of Node-RED nodes for IBM Watson services
0.9.2 5 months ago

node-red-node-watson-ucg
A collection of Node-RED nodes for IBM Watson services
0.0.11 3 years, 1 month ago

node-red-contrib-watson-content-hub
Node-RED nodes for Watson Content Hub integration
0.2.5 1 year, 5 months ago

node-red-contrib-watson-machine-learning
Interact with Watson Machine learning
0.1.1 5 months ago

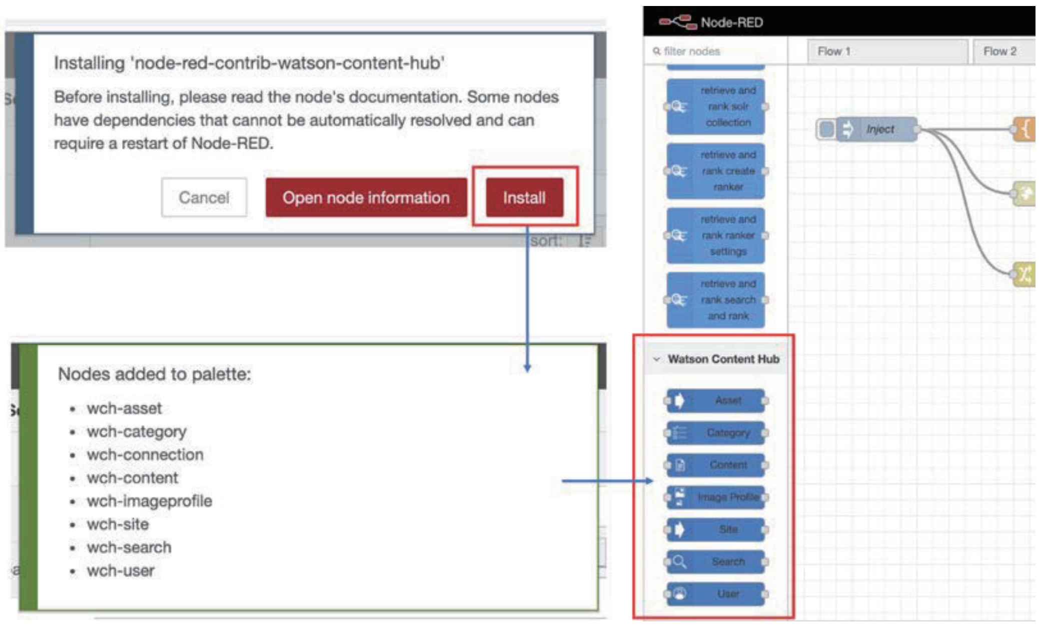
node-red-contrib-watson-token-generator
A Node-RED node to generate watson tokens

Figure 4.14 – Opening the User Settings panel and finding the node you want to use

3. After clicking on the **Install** button, a pop-up window will appear, on which you will need to click on **Install** once again.

   Once you do this and the installation has completed, you will get a pop-up message saying **Nodes added to palette**.

That's all! You can see all the nodes you have installed in your palette as shown in the following figure:


Installing 'node-red-contrib-watson-content-hub'

Before installing, please read the node's documentation. Some nodes have dependencies that cannot be automatically resolved and can require a restart of Node-RED.

Nodes added to palette:

- wch-asset
- wch-category
- wch-connection
- wch-content
- wch-imageprofile
- wch-site
- wch-search
- wch-user

Figure 4.15 – Nodes you have installed are added to your palette

> **Tip**
> You can search for useful nodes on the Node-RED Library website. It's possible to search by keywords, and sort the results in terms of most recently added, number of downloads, and ratings. I recommend sorting by number of downloads first because nodes that have been downloaded by lots of developers are likely to be very useful: `https://flows.nodered.org/search?type=node&sort=downloads`.

Now you have become a great Node-RED user and have mastered how to use the Node-RED flow editor to make some flows (applications).

Summary

In this chapter, you've learned how to use each major node in the Node-RED flow editor. You have successfully made your Node-RED flows! The flow steps you've created here are most of the steps you will need to do to create various flows in the future.

The important point learned in this chapter is that each node has its own unique features. By combining these like a puzzle, we can create an application similar to one made through regular programming just by creating a flow.

In the next chapter, let's create a more practical sample flow (application) for IoT edge devices.

Section 2: Mastering Node-RED

In this section, readers will actually create an application using the Node-RED flow editor. Instead of trying to build advanced applications from the beginning, first they will learn how to create a sample flow for each major environment (that is, stand-alone environments such as the Raspberry Pi, desktop, and cloud).

In this section, we will cover the following chapters:

5
Implementing Node-RED Locally

In this chapter, let's use the standalone version of Node-RED. Node-RED consists of a development environment, an execution environment, and the application itself. You can understand the mechanism by using the standalone version that runs in the local environment.

Specifically, the most common reason for starting the standalone version of Node-RED is when using it on an IoT edge device. IoT edge devices have sensors that are usually applied to the "Things" part of the "Internet of Things." In this chapter, we will look at the sensing data within the edge device and create a sample flow.

Let's get started with the following four topics:

- Running Node-RED on a local machine
- Using the standalone version of Node-RED
- Using IoT on edge devices
- Making a sample flow

By the end of this chapter, you will have learned how to build a flow for handling sensor data on IoT devices.

Technical requirements

To progress through this chapter, you will need the following:

- Node-RED (v1.1.0 or above): `https://nodered.org/`
- Raspberry Pi: `https://www.raspberrypi.org/`

The code used in this chapter can be found in `Chapter05` folder at `https://github.com/PacktPublishing/-Practical-Node-RED-Programming`.

Running Node-RED on a local machine

We can now create the flow for sensing data on an IoT edge device, and in this scenario, the local machine uses Raspberry Pi. The reason for this will be described in the *Using the standalone verison of Node-RED* section, but in summary, this tutorial is for IoT edge device.

I have already explained how to start Node-RED on Raspberry Pi, so you should now know how to run it, but if you need a refresher, please refer to the *Install Node-RED for Raspberry Pi* section in *Chapter 2*, *Setting Up the Development Environment*.

Now, follow these steps to start Node-RED on your Raspberry Pi:

1. Let's start by executing Node-RED from the Raspberry Pi menu:

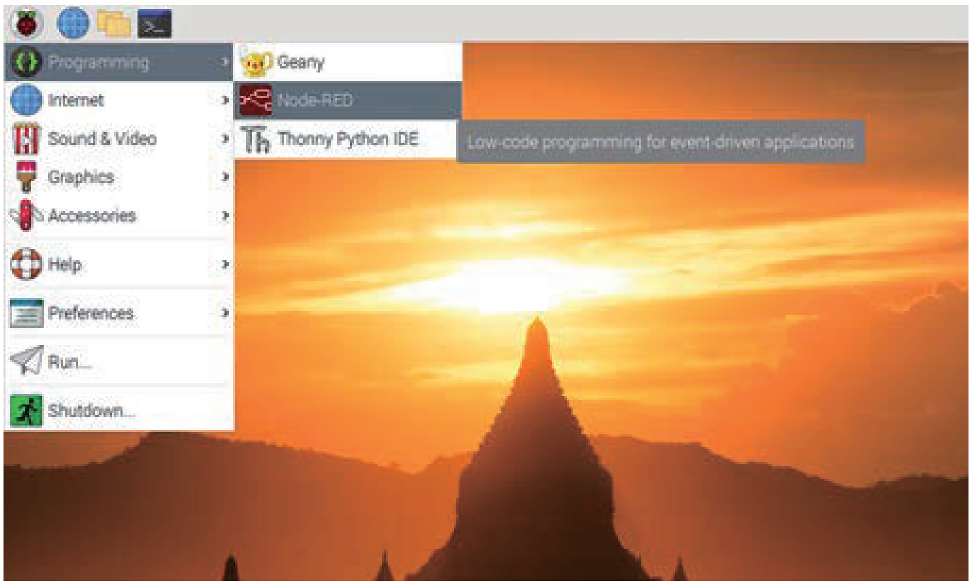
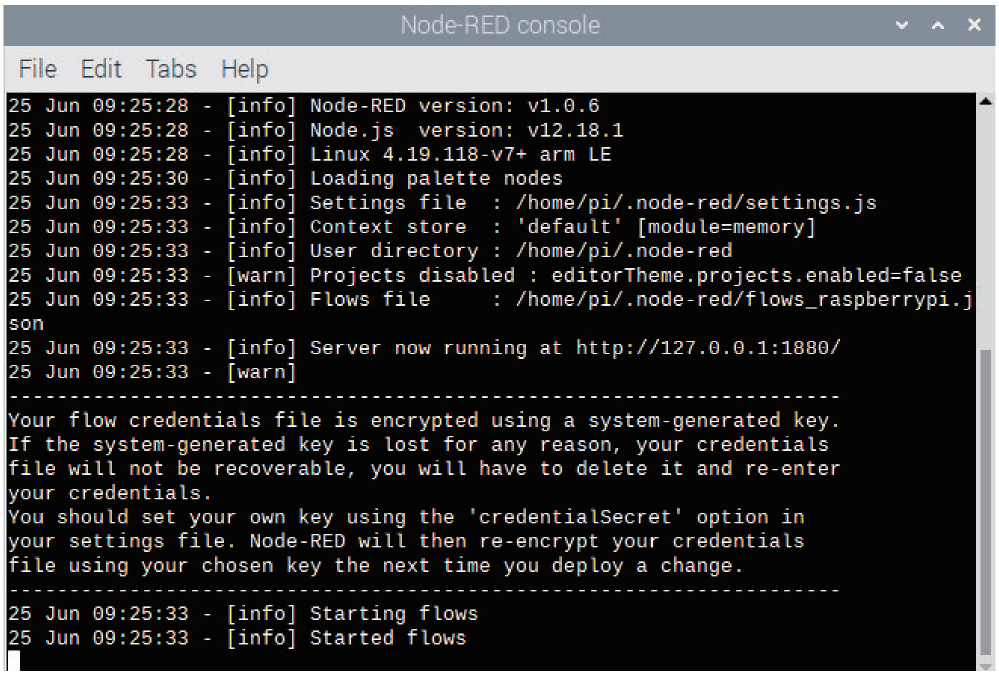


Figure 5.1 – Running Node-RED from the Raspberry Pi menu

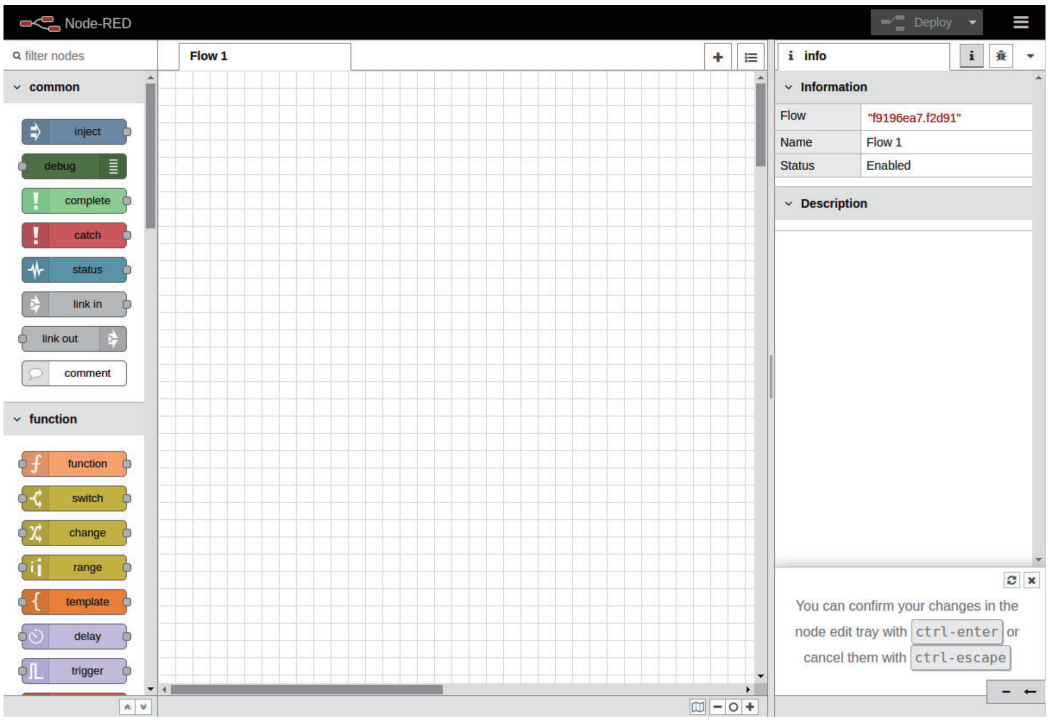
2. You can check the status of Node-RED on your terminal. If **Started flows** is shown, Node-RED is ready to use:



```
25 Jun 09:25:28 - [info] Node-RED version: v1.0.6
25 Jun 09:25:28 - [info] Node.js  version: v12.18.1
25 Jun 09:25:28 - [info] Linux 4.19.118-v7+ arm LE
25 Jun 09:25:30 - [info] Loading palette nodes
25 Jun 09:25:33 - [info] Settings file  : /home/pi/.node-red/settings.js
25 Jun 09:25:33 - [info] Context store  : 'default' [module=memory]
25 Jun 09:25:33 - [info] User directory : /home/pi/.node-red
25 Jun 09:25:33 - [warn] Projects disabled : editorTheme.projects.enabled=false
25 Jun 09:25:33 - [info] Flows file     : /home/pi/.node-red/flows_raspberrypi.j
son
25 Jun 09:25:33 - [info] Server now running at http://127.0.0.1:1880/
25 Jun 09:25:33 - [warn]
---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------
25 Jun 09:25:33 - [info] Starting flows
25 Jun 09:25:33 - [info] Started flows
```

Figure 5.2 – Terminal of Raspberry Pi

3. You can access the Node-RED flow editor with the `localhost:1880` URL:



Flow	"f9196ea7.f2d91"
Name	Flow 1
Status	Enabled

You can confirm your changes in the node edit tray with `ctrl-enter` or cancel them with `ctrl-escape`

Figure 5.3 – Node-RED flow editor

Let's learn a few concepts before making use of the flow editor.

Using the standalone version of Node-RED

Now we will learn what the standalone version of Node-RED is and how it differs from other versions. We usually use the Node-RED flow editor as a standalone editor; however, we can also use the Node-RED flow editor on any cloud with container technologies such as Docker, Kubernetes, or Cloud Foundry. We will explicitly demonstrate the use of the standalone version with relatively common use cases to learn how to use it.

Let's think about situations where Node-RED is used.

Node-RED is a tool for creating applications made with Node.js. It is also the execution environment. If you can write an application in Node.js, that's fine.

So, why build an application with Node-RED?

One answer is to black-box each individual unit of data processing. This makes the role of each process very clear and easy to build and maintain.

Another answer is to avoid human error. Since each process is modularized as a node, you only need to understand the input/output specifications when using that process. This means you can avoid human errors such as coding mistakes and missing test specifications. This can be the advantage of no-code/low-code as well as Node-RED.

Next, imagine a concrete situation that uses Node-RED with the characteristics just described.

Think of a business logic that controls data and connects it to the next process. This is a common situation in IoT solutions.

The standard architecture for IoT solutions is built with edge devices and cloud platforms. It sends the sensor data acquired by the edge device to the cloud and then, on the cloud work to process the data, such as visualizing, analyzing, and persistent.

In this chapter, I would like to focus on that edge device part.

It is common for edge devices to want to prepare the acquired sensor data to some extent before sending it to the cloud. The reason for this that if you send all the acquired data, there is a risk that the network will be overloaded.

So, the standalone Node-RED exercise uses Raspberry Pi, which is a famous IoT infrastructure for consumers.

In this chapter, we will use the **Grove Base HAT** for Raspberry Pi and Grove Base modules. This is one of the standards for the IoT edge device platform and so we need to install the Grove Base driver to Raspberry Pi.

Important Note

This chapter gives an example using Grove Base HAT, which is relatively inexpensive and can be purchased (the link to this is mentioned in the next section), but any sensor device that can be connected to a Raspberry Pi can handle data on Node-RED.

When using a module other than the Grove Base HAT sensor device, use the corresponding node and read this chapter. (Implementation is required if there is no corresponding node.)

You can check the Node-RED library for the existence of a node that corresponds to each device:

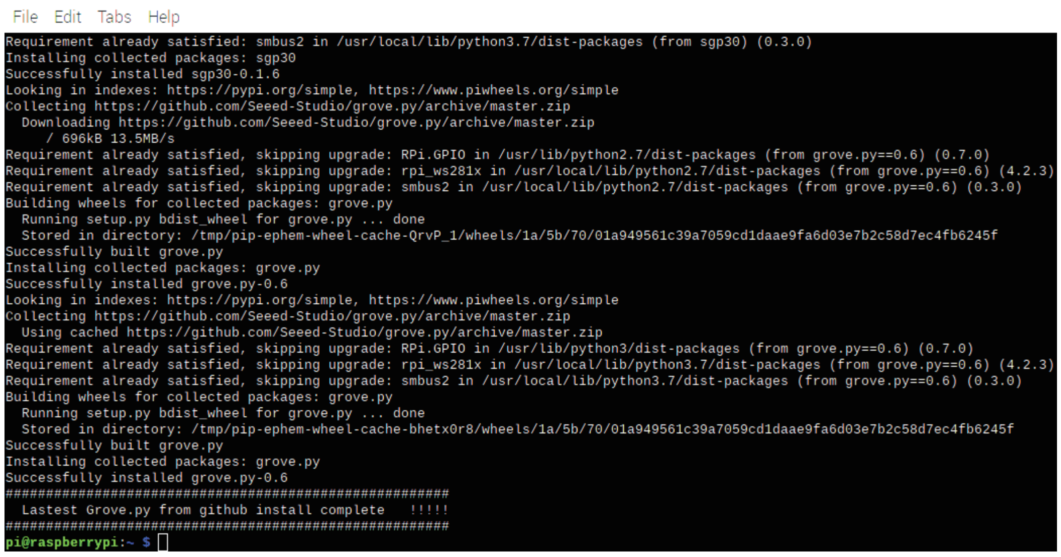
`https://flows.nodered.org/`

Let's prepare to use Grove Base HAT on our Raspberry Pi by following these steps:

1. Let's start by executing the following command on our Raspberry Pi:

   ```
     $ curl -sL https://github.com/Seeed-Studio/grove.py/raw/
   master/install.sh | sudo bash -s -
   ```

2. If everything goes well, you will see the following notice:



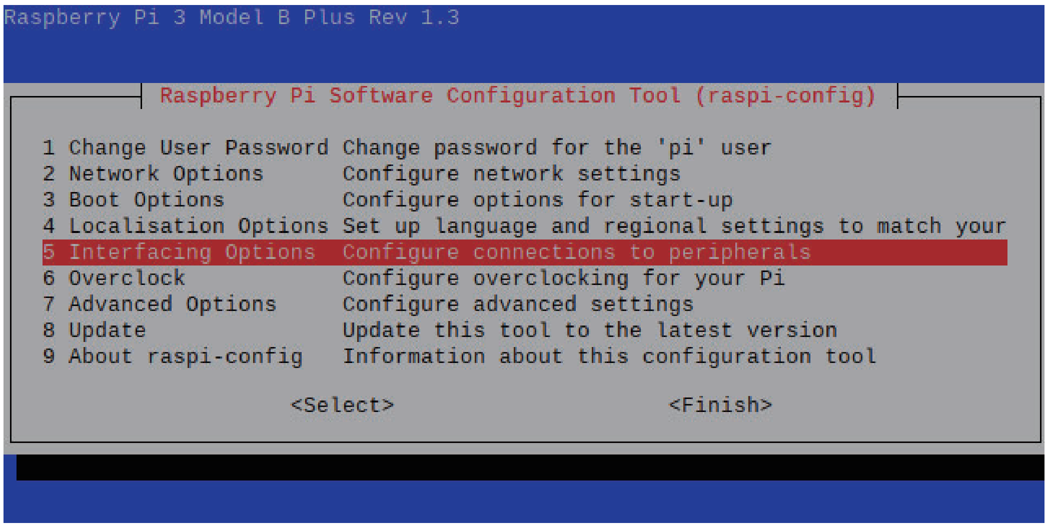
```
Requirement already satisfied: smbus2 in /usr/local/lib/python3.7/dist-packages (from sgp30) (0.3.0)
Installing collected packages: sgp30
Successfully installed sgp30-0.1.6
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting https://github.com/Seeed-Studio/grove.py/archive/master.zip
  Downloading https://github.com/Seeed-Studio/grove.py/archive/master.zip
     / 696kB 13.5MB/s
Requirement already satisfied, skipping upgrade: RPi.GPIO in /usr/lib/python2.7/dist-packages (from grove.py==0.6) (0.7.0)
Requirement already satisfied, skipping upgrade: rpi_ws281x in /usr/local/lib/python2.7/dist-packages (from grove.py==0.6) (4.2.3)
Requirement already satisfied, skipping upgrade: smbus2 in /usr/local/lib/python2.7/dist-packages (from grove.py==0.6) (0.3.0)
Building wheels for collected packages: grove.py
  Running setup.py bdist_wheel for grove.py ... done
  Stored in directory: /tmp/pip-ephem-wheel-cache-QrvP_1/wheels/1a/5b/70/01a949561c39a7059cd1daae9fa6d03e7b2c58d7ec4fb6245f
Successfully built grove.py
Installing collected packages: grove.py
Successfully installed grove.py-0.6
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting https://github.com/Seeed-Studio/grove.py/archive/master.zip
  Using cached https://github.com/Seeed-Studio/grove.py/archive/master.zip
Requirement already satisfied, skipping upgrade: RPi.GPIO in /usr/lib/python3/dist-packages (from grove.py==0.6) (0.7.0)
Requirement already satisfied, skipping upgrade: rpi_ws281x in /usr/local/lib/python3.7/dist-packages (from grove.py==0.6) (4.2.3)
Requirement already satisfied, skipping upgrade: smbus2 in /usr/local/lib/python3.7/dist-packages (from grove.py==0.6) (0.3.0)
Building wheels for collected packages: grove.py
  Running setup.py bdist_wheel for grove.py ... done
  Stored in directory: /tmp/pip-ephem-wheel-cache-bhetx0r8/wheels/1a/5b/70/01a949561c39a7059cd1daae9fa6d03e7b2c58d7ec4fb6245f
Successfully built grove.py
Installing collected packages: grove.py
Successfully installed grove.py-0.6
#######################################################
  Lastest Grove.py from github install complete   !!!!!
#######################################################
pi@raspberrypi:~ $ 
```

Figure 5.4 – Successful grove.py installation

3. The next step is to enable ARM I2C. We can do this by executing the following command:

   ```
     $ sudo raspi-config
   ```

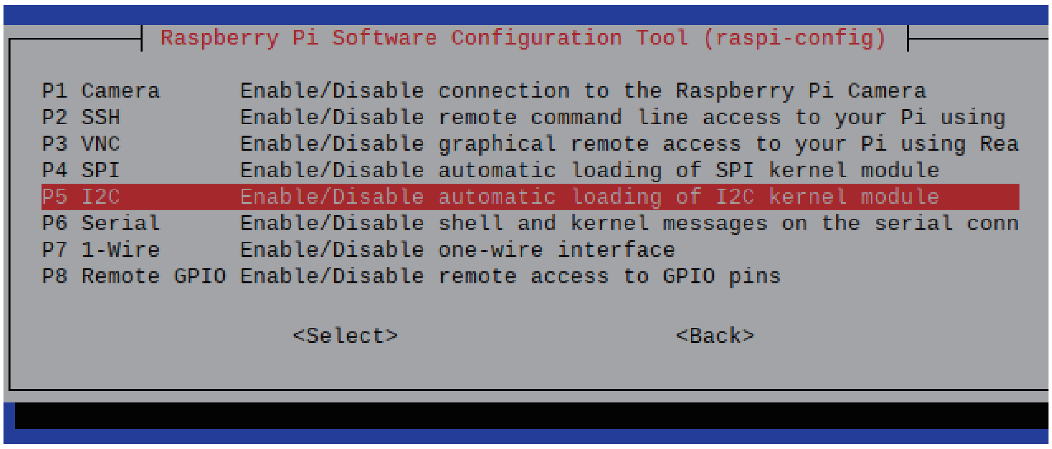
4. After executing the command, you will see the following configuration window. Please select **Interfacing Options**:



```
Raspberry Pi 3 Model B Plus Rev 1.3

┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

 1 Change User Password Change password for the 'pi' user
 2 Network Options      Configure network settings
 3 Boot Options         Configure options for start-up
 4 Localisation Options Set up language and regional settings to match your
 5 Interfacing Options  Configure connections to peripherals
 6 Overclock            Configure overclocking for your Pi
 7 Advanced Options     Configure advanced settings
 8 Update               Update this tool to the latest version
 9 About raspi-config   Information about this configuration tool

                <Select>                     <Finish>
```

Figure 5.5 – Software configuration tool

5. Select **I2C**:



```
┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

 P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
 P2 SSH         Enable/Disable remote command line access to your Pi using
 P3 VNC         Enable/Disable graphical remote access to your Pi using Rea
 P4 SPI         Enable/Disable automatic loading of SPI kernel module
 P5 I2C         Enable/Disable automatic loading of I2C kernel module
 P6 Serial      Enable/Disable shell and kernel messages on the serial conn
 P7 1-Wire      Enable/Disable one-wire interface
 P8 Remote GPIO Enable/Disable remote access to GPIO pins

                <Select>                     <Back>
```

Figure 5.6 – Enabling I2C

6. Once you select it, a **Would you like the ARM I2C interface to be enabled?** message will be shown in the same window. Please select **Yes** to accept it.

You have now successfully enabled I2C. Restart the Raspberry Pi and restart the Node-RED flow editor. In doing this, your Raspberry Pi has been made available to use the I2C interface, and for the next step, we need to connect the sensor devices and Raspberry Pi via the I2C interface.

Using IoT on edge devices

Now let's consider a case study on edge devices in IoT.

IoT has recently been adopted in several industries, for example, in the fields of weather forecasting and agriculture; however, the basic composition is the same. Various data acquired by the edge device is sent to the server-side platform, such as the cloud, and the data is handled and visualized on the server side, which is full of resources. There are various ways to visualize, but in the simplest case, it will be to output the necessary data values to the log as a standard output.

In this chapter, I would like to consider the edge device part in the use case of IoT. This is about handling the sensor data, acquired using the sensor module, before it goes to the server side for formatting and narrowing down.

What are the different kinds of sensors?

The following sensors are often used at the experimental level of IoT:

- Temperature
- Humidity
- Gyroscope (acceleration, angular velocity)
- Light
- Sound
- Pressure-sensitive
- Magnetic

Here we will consider the use case of outputting the acquired value to the log using a light sensor and a temperature/humidity sensor.

In order to get sensor data, you'll need a device. In this sample flow (application), Raspberry Pi is used, but it does not have a sensing function because it is just a foundation. With the old-fashioned board, you had to solder the sensor device/module, but the convenient thing about the Raspberry Pi is that there are many sensor module kits that can be connected with one touch.

As already introduced, we'll use the Grove series provided by Seeed, which has a sensor module and connection board for Raspberry Pi: `https://wiki.seeedstudio.com/Grove_Base_Hat_for_Raspberry_Pi/`

Let's prepare the Grove Base HAT for Raspberry Pi modules.

Important Note

If you don't have the Grove Base HAT for Raspberry Pi and want to run this tutorial, please buy it via the official site (`https://www.seeedstudio.com/Grove-Base-Hat-for-Raspberry-Pi.html`).

This is what the Grove Base HAT for Raspberry Pi looks like:

Figure 5.7 – Grove Base HAT for Raspberry Pi

We need to connect the Grove Base HAT and the sensor modules to the Raspberry Pi. To do so, follow these steps:

1. Place the Grove Base HAT on your Raspberry Pi and screw it in:

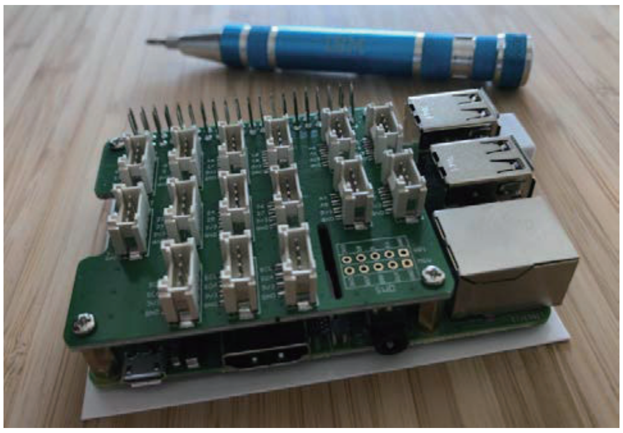


Figure 5.8 – Setting the Base HAT on your Raspberry Pi

This is what the Grove - Light Sensor v1.2 - LS06-S phototransistor looks like:

Figure 5.9 – Grove - Light Sensor v1.2

You can get it from `https://www.seeedstudio.com/Grove-Light-Sensor-v1-2-LS06-S-phototransistor.html`.

2. Connect the Grove light sensor to the analog port of your Base HAT:

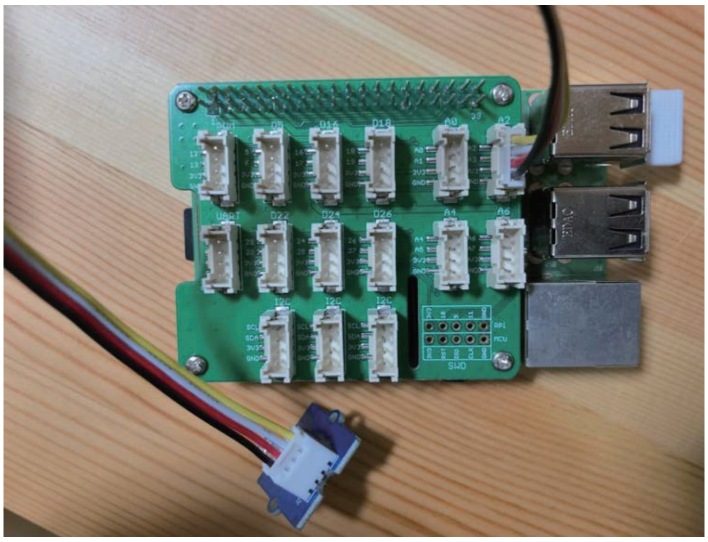


Figure 5.10 – Connecting the light sensor to your Base HAT

Important Note

Please be careful! This vendor, **Seeed,** has a similar module for temperature/humidity sensor **SHT35**, but it's not supported by the Grove Base HAT node. You need to use **SHT31**.

This is what the Grove - Temperature&Humidity Sensor (SHT31) looks like:

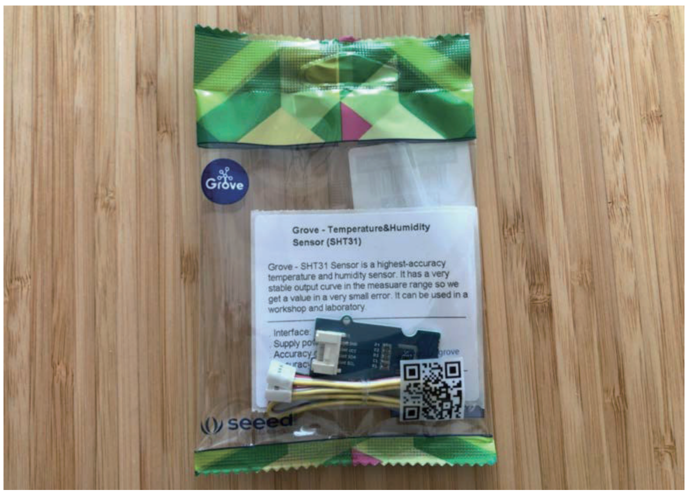

Grove
Grove - Temperature&Humidity Sensor (SHT31)
Grove - SHT31 Sensor is a highest-accuracy temperature and humidity sensor. It has a very stable output curve in the measure range so we get a value in a very small error. It can be used in a workshop and laboratory.
seeed

Figure 5.11 – Grove – Temperature&Humidity Sensor (SHT31)

You can get it from `https://www.seeedstudio.com/Grove-Temperature-Humidity-Sensor-SHT31.html`.

3. Connect the Grove temperature and humidity sensor to the I2C port of your Base HAT:

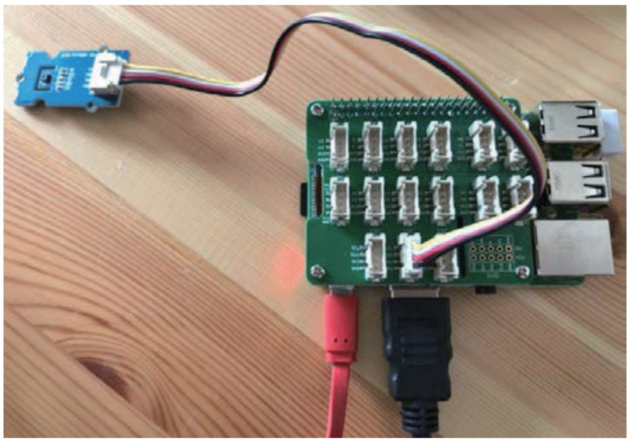


Figure 5.12 – Connecting the temperature/humidity sensor to your Base HAT

And that's it. Now your device is set up and we are ready to go on to the next step! In this part, we have learned about popular, simple use cases of IoT edge devices and next, we will make a flow for these use cases.

Making a sample flow

In this section, we will create these two sensor data output flows in the Node-RED flow editor.

You will use the sensor modules you have prepared to collect data and create a sample flow to visualize it on Node-RED. By using two different sensor modules, we can learn the basics of data handling in Node-RED.

Use case 1 – light sensor

The first is a light sensor. Let's create a flow (application) that detects light and outputs the value detected by a fixed-point observation to a log:

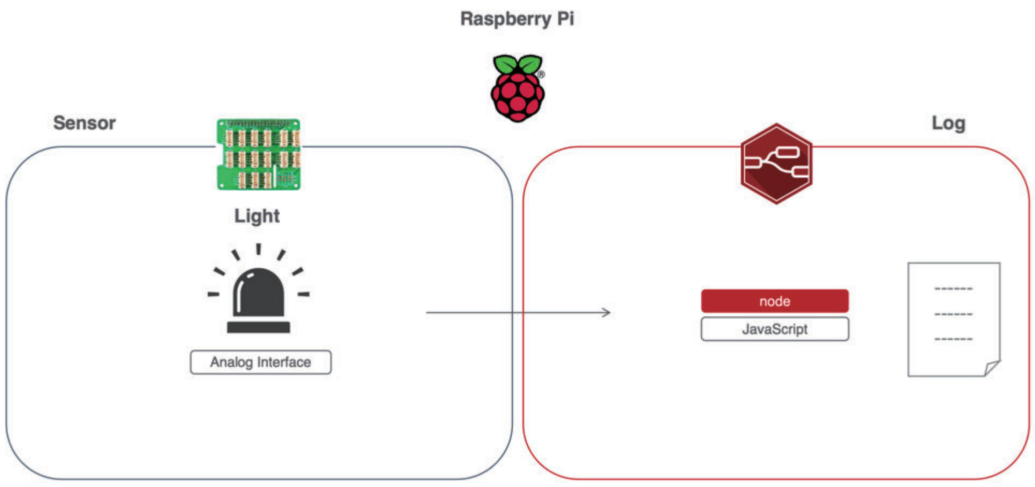


Figure 5.13 – Use case 1 – getting light sensor data

Connect the light sensor module to the Raspberry Pi and use the Node-RED flow editor on the Raspberry Pi to output the data obtained as a standard output.

Use case 2 – temperature/humidity sensor

The second one is a temperature/humidity sensor. Let's create an application (flow) that detects temperature and humidity and outputs the value detected by a fixed-point observation to a log:

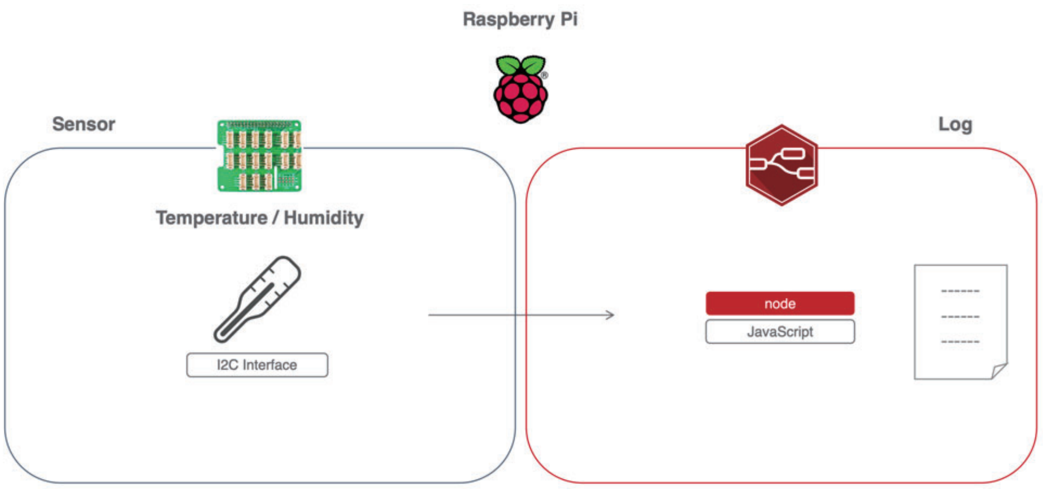


Figure 5.14 – Use case 2 – getting temperature/humidity data

Connect the temperature/humidity sensor module to the Raspberry Pi and use the Node-RED flow editor on the Raspberry Pi to output the data obtained as a standard output.

If you want to spot test these two use cases on your device, you need to connect a sensor that you can use to obtain sensor data.

You may have to prepare this before creating the flow.

This time, we will use Grove Base HAT, which is easy to use with Raspberry Pi, and as this setup was completed in the previous step, we are ready to access the data on Raspberry Pi. However, we have not yet prepared Node-RED. It is difficult to access this data with Node-RED as default. One way is to use a Function node and code the script from scratch, which is very difficult but not impossible.

For handling the sensing data recognized by Raspberry Pi on Node-RED, a "node" dedicated to Grove Base HAT is required.

The good news is that you can start using the node right away. This is because Seigo Tanaka, a Node-RED User Group Japan board member (`https://nodered.jp/`) and Node-RED contributor, has already created and released a node for Grove Base HAT. This is the node for the Grove Base HAT for Raspberry Pi:

```
node-red-contrib-grove-base-hat
```

You can read more about it here: `https://www.npmjs.com/package/node-red-contrib-grove-base-hat`.

If you need a refresher on how to install nodes that are published on the node library, please read the *Getting several nodes from the library* section in *Chapter 4, Learning the Major Nodes*.

The reason I refer you back to this is that the next step is to install the node for the Grove Base HAT from the library into your environment.

Let's enable the use of this Grove Base HAT node in our Node-RED flow editor:

1. Click the menu at the top right and select **Manage palette** to open the settings panel:

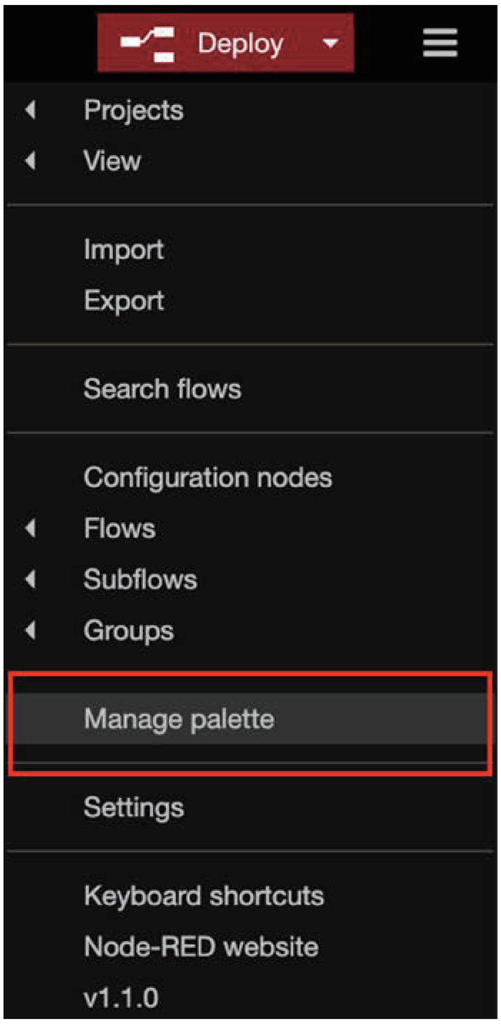


Figure 5.15 – Selecting Manage palette

2. When the settings panel is opened, type the name of the node you want to use in the search window. We want to use **node-red-contrib-grove-base-hat**, so please type the following:

```
grove base
```

3. After that, you can see the **node-red-contrib-grove-base-hat** node in the search window. Click the **Install** button:

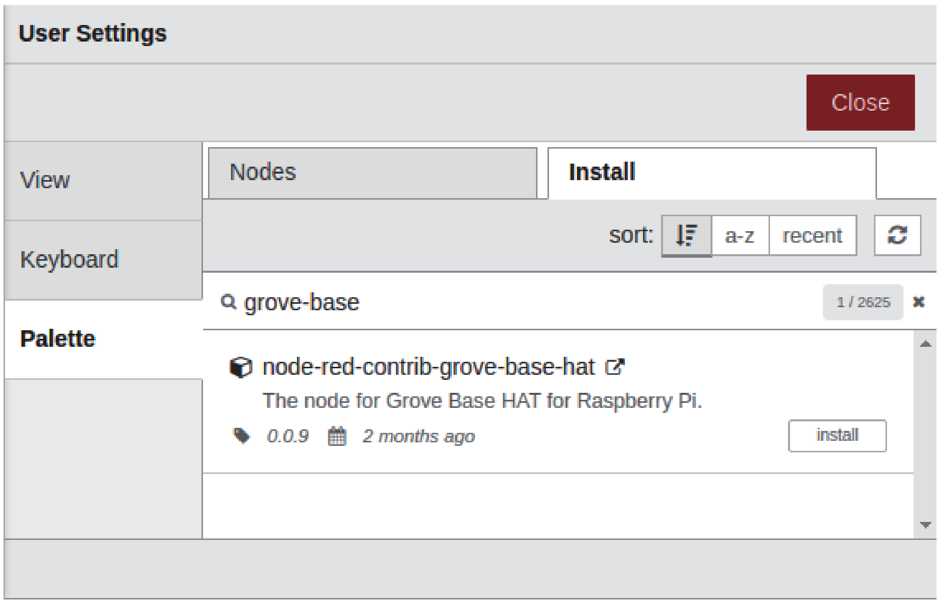
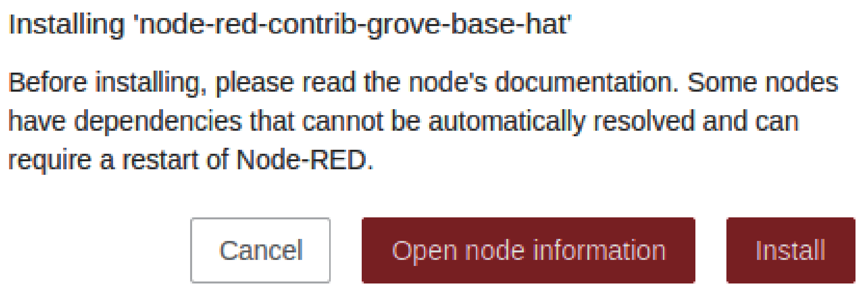


Figure 5.16 – Installing the node-red-contrib-grove-base-hat node

4. After clicking the **Install** button, you will see a message asking you to read the documentation to find out more information about this node. Read the document if necessary, and then click the **Install** button on the message box:


Installing 'node-red-contrib-grove-base-hat'

Before installing, please read the node's documentation. Some nodes have dependencies that cannot be automatically resolved and can require a restart of Node-RED.

Figure 5.17 – A message window to read the node documentation

Now you are ready to use the node for Grove Base HAT. Check the palette in the flow editor. At the bottom of the palette, you can see that the Grove Base HAT node has been added:

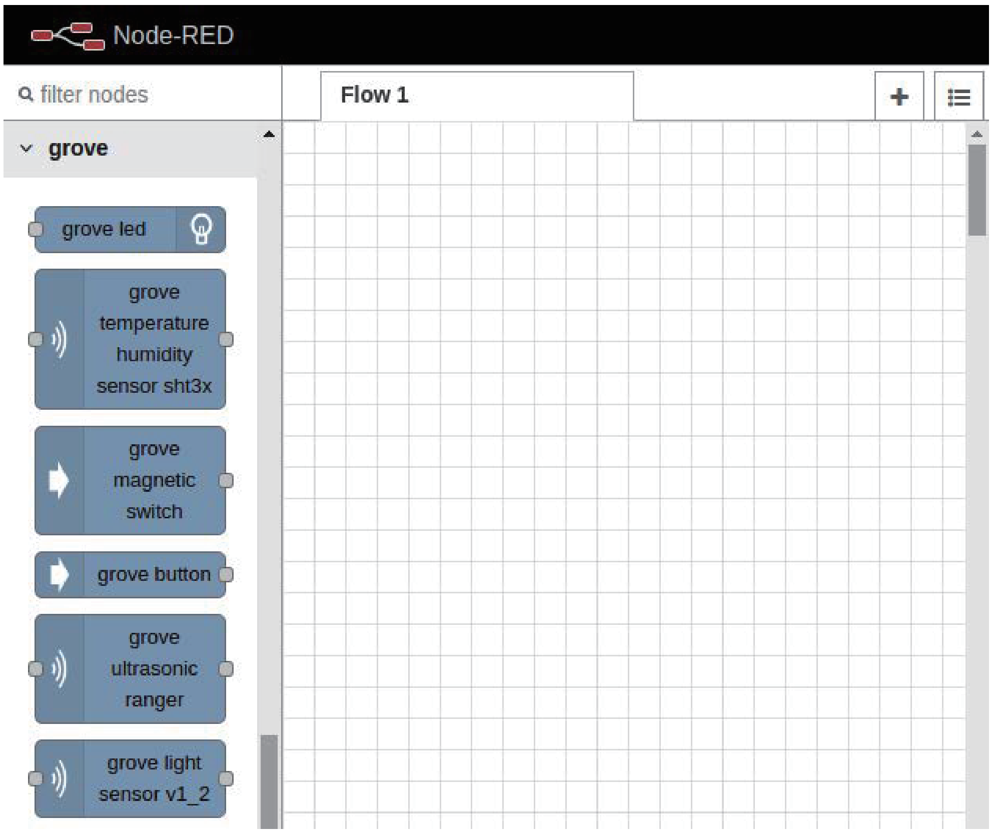


Figure 5.18 – Grove Base HAT nodes on your dashboard

There are many sensing modules that can be connected to Grove Base HAT. This time, only the light and temperature/humidity sensors are used, but there are other things that can be seen by looking at the types of nodes.

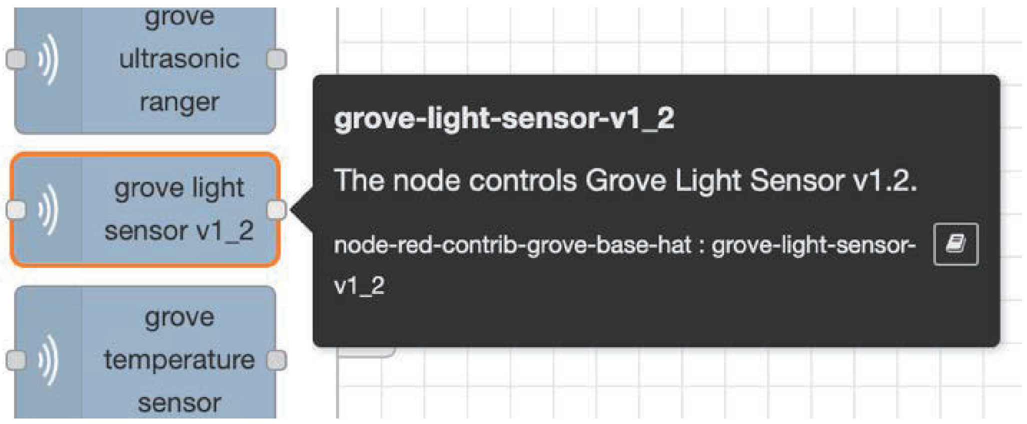
The procedure followed for the two use cases created here can also be applied when using other sensors. If you are interested, please try other sensors too. In the next section, we will make a flow for use case 1.

Making a flow for use case 1 – light sensor

In use case 1, Node-RED can be used to handle the illuminance obtained from the light sensor as JSON data. That data can be handled as JSON data, then be sent to the server side afterward, and various processes can be easily performed on the edge device.

The value obtained from the light sensor is received by Node-RED and the output is a debug log (standard output). We can set this using the following steps:

1. Select the **grove light sensor v1_2** node from the palette on the left side of the flow editor and drag and drop it into the workspace to place it:



grove-light-sensor-v1_2

The node controls Grove Light Sensor v1.2.

node-red-contrib-grove-base-hat : grove-light-sensor-v1_2

Figure 5.19 – grove light sensor v1_2

   This node allows the value of the sensor device, which is continuously acquired on the Raspberry Pi via the Grove Base HAT, to be handled as a JSON format message object on Node-RED.

2. After placing the **grove-light-sensor-v1_2** node, place the **inject** node and **debug** nodes and wire them so that the **grove-light-sensor-v1_2** node you placed is sandwiched between them:

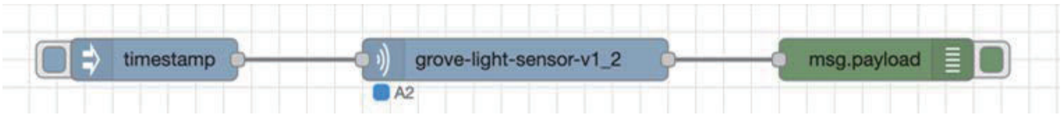
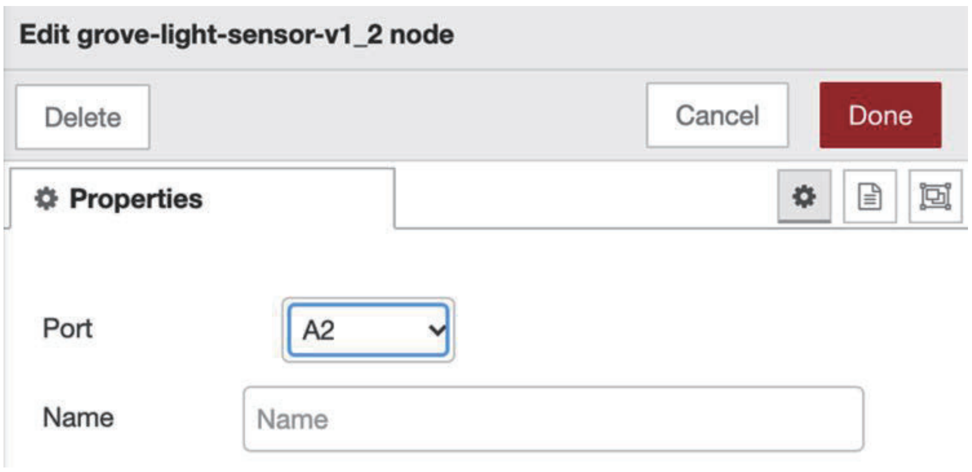


Figure 5.20 – Placing nodes and wiring them for the light sensor

3. Next, check the settings of the **grove-light-sensor-v1_2** node. Double-click the node to open the settings panel.
4. There is a selection item called **Port** in the settings panel. **A0** is selected by default.

   This **Port** setting is to specify which connector on the Grove Base HAT gets data from the connected module.

5. Earlier, we connected the Grove light sensor to the Grove Base HAT. If the connection is made according to the procedure in this tutorial, it should be connected to port A2, so select **A2** as the node setting value. If you are connecting to another port, select the port you are connecting to:



Port: A2
Name: Name

Figure 5.21 – Select A2 as the port if you connected the sensor to A2 of Base HAT

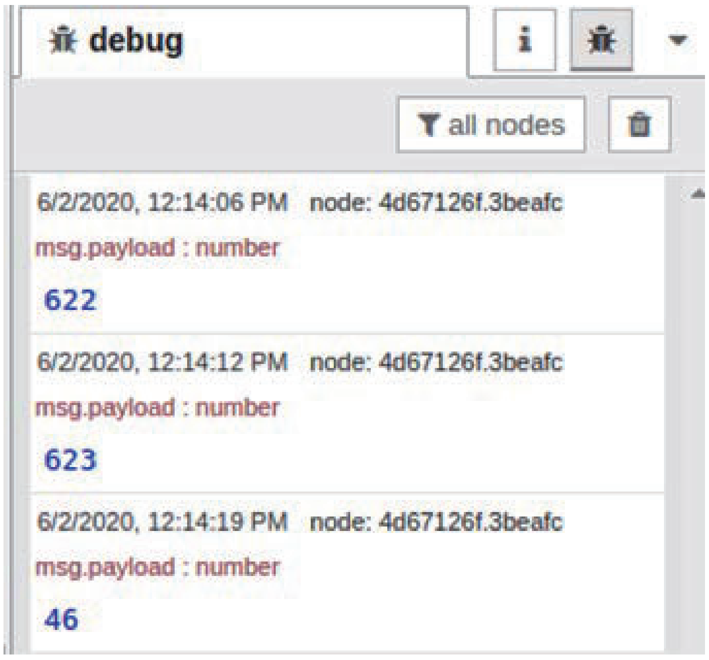
6. After checking and setting **Port** on the settings panel, click the **Done** button in the upper-right corner to close the settings panel.

   That's it! Don't forget to click the **deploy** button.

You should remember how to execute a flow from a inject node, because you learned about this in the previous chapter. Click the switch on the inject node to run the flow. The data for the timing when the switch is clicked is outputted as a log, so please try clicking it a couple of times.

> **Important Note**
> Do not forget to display the debug window to show that the value of the acquired data will be the output to the debug window. Node-RED does not automatically show the debug window even if the debug output is activated.

The resulting output in the **debug** window looks like the following:



```
6/2/2020, 12:14:06 PM   node: 4d67126f.3beafc
msg.payload : number
622
6/2/2020, 12:14:12 PM   node: 4d67126f.3beafc
msg.payload : number
623
6/2/2020, 12:14:19 PM   node: 4d67126f.3beafc
msg.payload : number
46
```

Figure 5.22 – Result of the light sensor flow

You can see that the result was output to the **debug** window.

Congratulations! With this, we have successfully created a basic flow (application) that handles the value of our first light sensor with Node-RED.

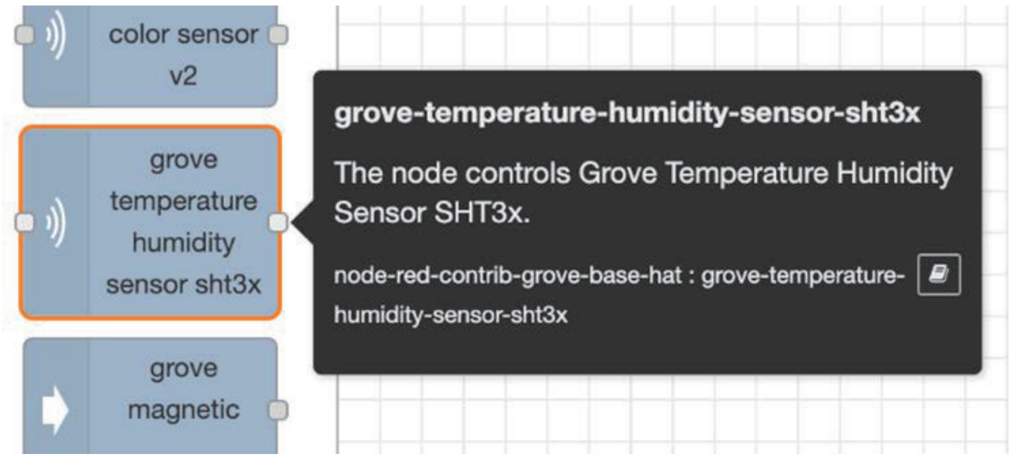
You can also download this flow definition file here: `https://github.com/PacktPublishing/-Practical-Node-RED-Programming/blob/master/Chapter05/light-sensor-flows.json`.

Making a flow for use case 2 – temperature/humidity sensor

In use case 2, Node-RED can be used to handle the temperature and the humidity obtained from the temperature/humidity sensor as JSON data. The data, which can be handled as JSON data, can be sent to the server side afterward, and various processes can be easily performed on the edge device.

The value obtained from the temperature/humidity sensor is received by Node-RED and is outputted as a debug log (standard output):

1. Select the **grove temperature humidity sensor sht3x** node from the palette on the left side of the flow editor and drag and drop it into the workspace to place it:


color sensor v2
grove temperature humidity sensor sht3x
grove magnetic
grove-temperature-humidity-sensor-sht3x

The node controls Grove Temperature Humidity Sensor SHT3x.

node-red-contrib-grove-base-hat : grove-temperature-humidity-sensor-sht3x

Figure 5.23 – grove temperature humidity sensor sht3x

This node allows the value of the sensor device, which is continuously acquired on the Raspberry Pi via Grove Base HAT, to be handled as a JSON format message object on Node-RED.

2. After placing the **grove-temperature-humidity-sensor-sht3x** node, place the **inject** and **debug** nodes, respectively, and wire them so that the **grove-temperature-humidity-sensor-sht3x** node you placed is sandwiched between them:

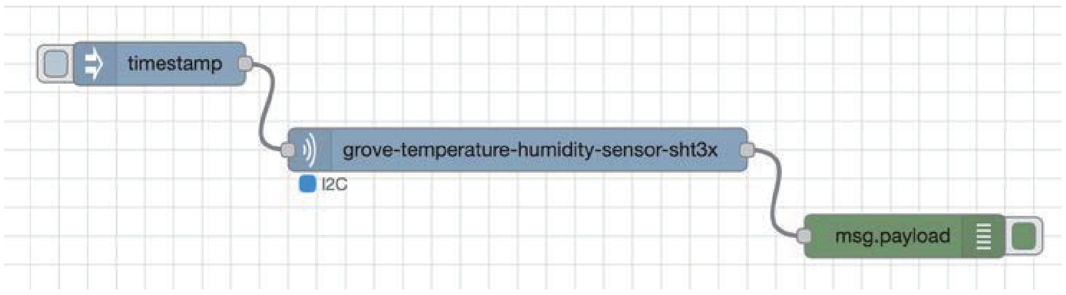
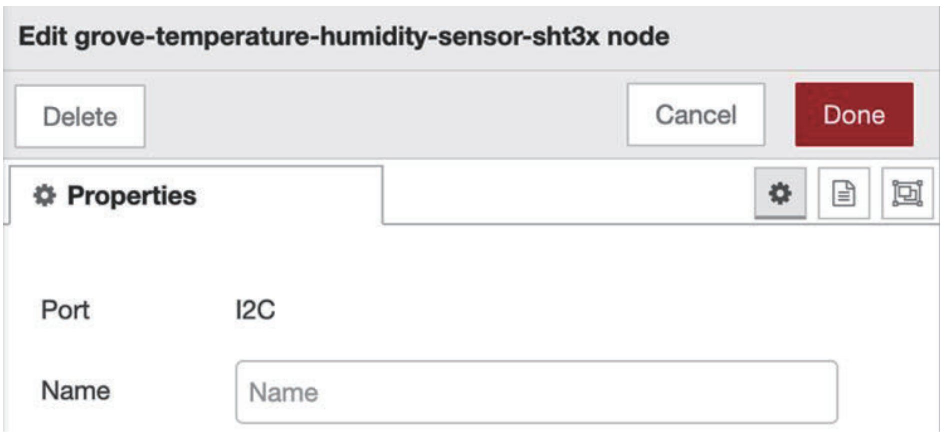


Figure 5.24 – Placing the nodes and wiring them for the temperature and humidity sensor

3. Next, check the settings of the **grove-temperature-humidity-sensor-sht3x** node and double-click the node to open the settings panel.

Actually, this node has no values to set (strictly speaking, the name can be set, but the presence or absence of this setting does not affect the operation):


Edit grove-temperature-humidity-sensor-sht3x node

Delete | Cancel | Done

Properties

Port I2C

Name Name

Figure 5.25 – Already set to the I2C port

You can see on the settings panel that the port is designated as **I2C** (not changeable). If you have connected the Grove temperature and humidity sensor to the Grove Base HAT according to the procedure in this document, the module should be correctly connected to the **I2C** port. If it is connected to a port other than I2C, reconnect it properly.

4. After checking **Port** on the settings panel, click the **Done** button in the upper-right corner to close the settings panel.

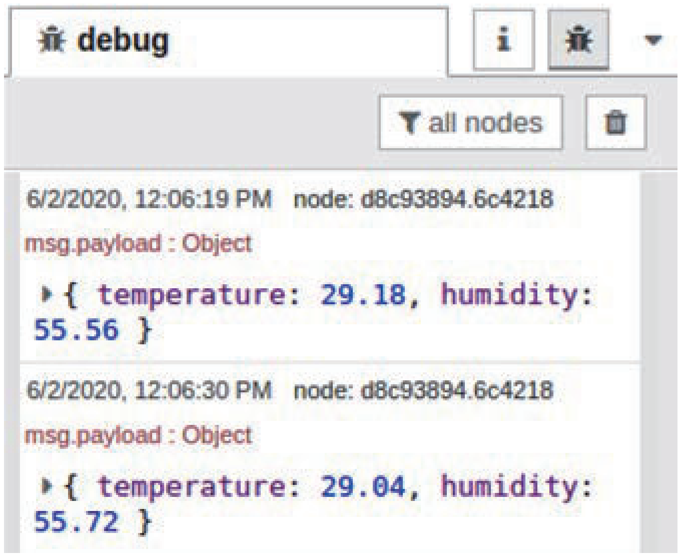
   That's it! Don't forget to click the **deploy** button.

5. Click the switch on the inject node to run the flow. The data for the timing when the switch is clicked is outputted as a log, so please try clicking it a couple of times.

Important Note

As noted in the previous section, do not forget to display the debug window to show that the value of the acquired data will be the output to the debug window. Node-RED does not automatically show the debug window even if the debug output is activated.

The resulting output in the **debug** window looks like the following:



```
6/2/2020, 12:06:19 PM   node: d8c93894.6c4218
msg.payload : Object
▸{ temperature: 29.18, humidity:
55.56 }
6/2/2020, 12:06:30 PM   node: d8c93894.6c4218
msg.payload : Object
▸{ temperature: 29.04, humidity:
55.72 }
```

Figure 5.26 – Result of the temperature/humidity sensor flow

You can see that the result was outputted to the **debug** window.

Congratulations! With this, we have successfully created a basic flow (application) that handles the value of the second sample, the temperature/humidity sensor, with Node-RED.

You can also download this flow definition file here: `https://github.com/PacktPublishing/-Practical-Node-RED-Programming/blob/master/Chapter05/light-sensor-flows.json`.

Well done! Now you have learned how to handle the data obtained from the illuminance sensor and temperature and humidity sensor in JSON format on Node-RED.

Summary

In this chapter, you learned how to create a sample flow (application) by comparing Node-RED to a real IoT use case. We experienced using the sensor module and Raspberry Pi to exchange data with Node-RED, so we had a feel for IoT.

The flow steps created here will help you create different flows with other sensor modules in the edge device in the future.

In the next chapter, we will use the IoT use case as we did this time, but we will create a practical sample flow (application) on the cloud side (server side).

6 Implementing Node-RED in the Cloud

In this chapter, we will learn how to utilize Node-RED, which can be used standalone on a cloud platform (mainly Platform as a Service). **Platform as a Service** (**PaaS**) provides an instance that acts as the execution environment for an application, and the application developers only focus on executing the application created by themselves without using their power to build the environment. Node-RED is actually a Node.js application, so you can run it wherever you have a runtime environment for Node.js.

There are various major mega clouds such as Azure, AWS, and GCP, but Node-RED is prepared as a Starter App (a web application that can be launched on IBM Cloud is called a Starter App) by default in IBM Cloud, so we will use it in this chapter.

In this chapter, we'll cover the following topics:

- Running Node-RED on the cloud
- What is the specific situation for using Node-RED in the cloud?
- IoT case study spot on the server side
- Making a sample flow

By the end of this chapter, you will have mastered how to build a flow for handling sensor data on the cloud.

Technical requirements

The code that will be used in this chapter can be found in the `Chapter06` folder at `https://github.com/PacktPublishing/-Practical-Node-RED-Programming`.

Running Node-RED on the cloud

This time, we will use IBM Cloud. The reason for this is that IBM Cloud has Node-RED Starter Kit on it. This is a kind of software boilerplate that includes services needed for Node-RED on the cloud, such as a database, CI/CD tools, and more.

If you have not used IBM Cloud yet, don't worry – IBM provides a free IBM Cloud account (Lite account) with no credit card registration needed. You can register for an IBM Cloud Lite account at `http://ibm.biz/packt-nodered`.

Before using Node-RED on IBM Cloud, you need to finish the registration process for your IBM Cloud Lite account.

> **Important Note**
>
> In this book, we strongly recommend that you select a Lite account when using IBM Cloud. You can upgrade from a Lite account to a standard account (PAYG/Pay as you go) at your own will. This means you can automatically upgrade to PAYG by registering your credit card.
>
> Please note that services that can be used free of charge with a Lite account may be charged for with PAYG.

Now, let's launch Node-RED on IBM Cloud by following these steps:

> **Important Note**
>
> The instructions/screenshots provided here are correct at the time of writing. The UI of IBM Cloud changes so often that it might be different from the current UI.

1. Log in to IBM Cloud (`https://cloud.ibm.com`) with the account you created previously:

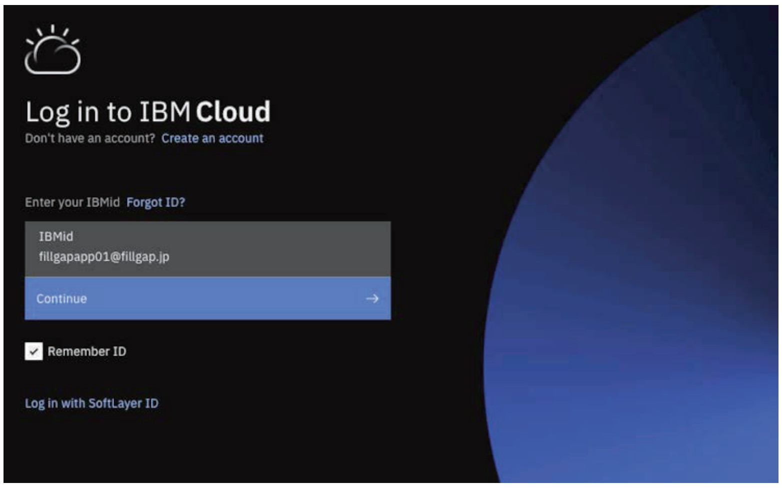


Figure 6.1 – Logging in via your Lite account

2. After logging into IBM Cloud, you will see your own dashboard on your screen. If this is your first time using IBM Cloud, no resources will be shown on the dashboard:

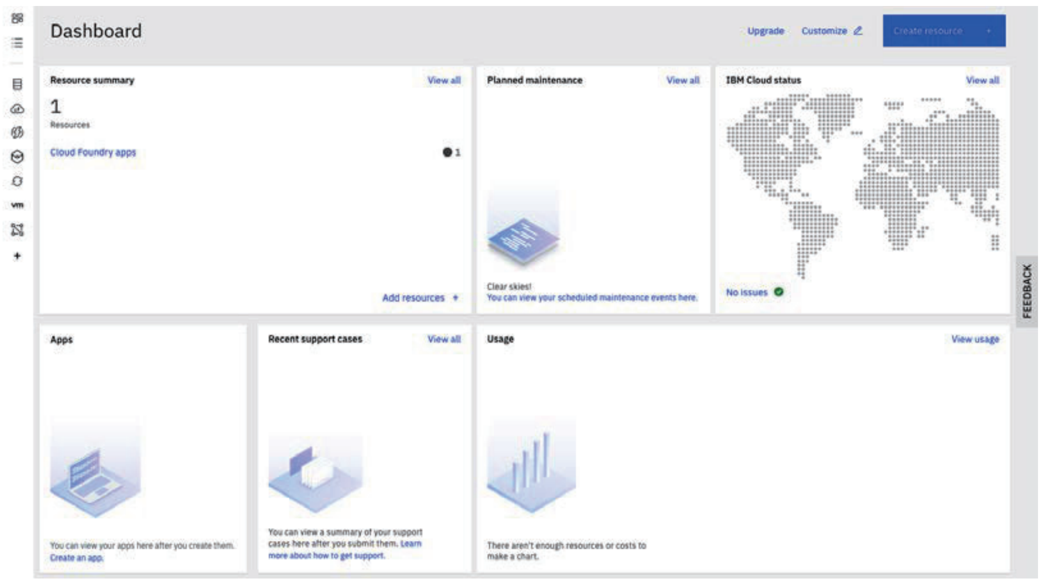
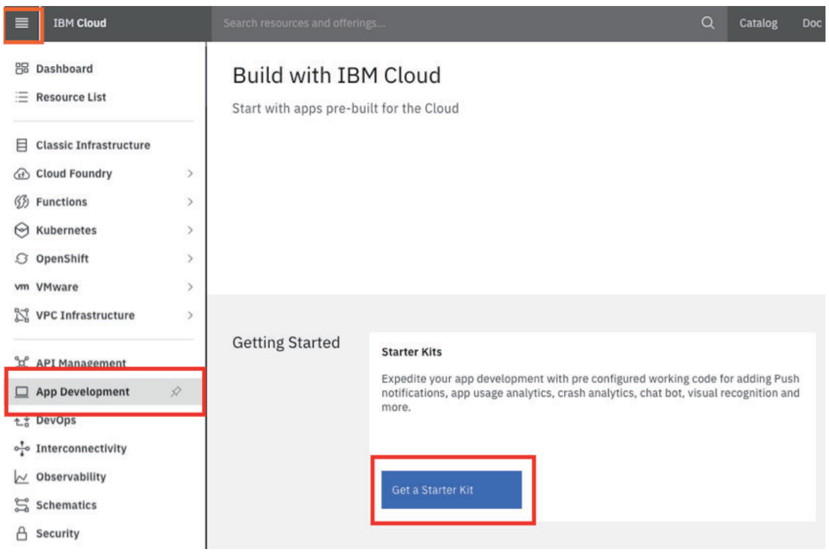

Dashboard
Upgrade
Customize
Create resource
Resource summary
View all
1
Resources
Cloud Foundry apps
Add resources
Planned maintenance
Clear skies!
You can view your scheduled maintenance events here.
IBM Cloud status
No issues
FEEDBACK
Apps
You can view your apps here after you create them.
Create an app.
Recent support cases
You can view a summary of your support cases here after you submit them. Learn more about how to get support.
Usage
View usage
There aren't enough resources or costs to make a chart.

Figure 6.2 – IBM Cloud dashboard

Next, we will create Node-RED on this cloud platform.

3. We will create Node-RED as a service on this cloud. Click **App Development** from the menu at the top left and click the **Get a Starter Kit** button. This lets you create a new application service:


Build with IBM Cloud

Start with apps pre-built for the Cloud

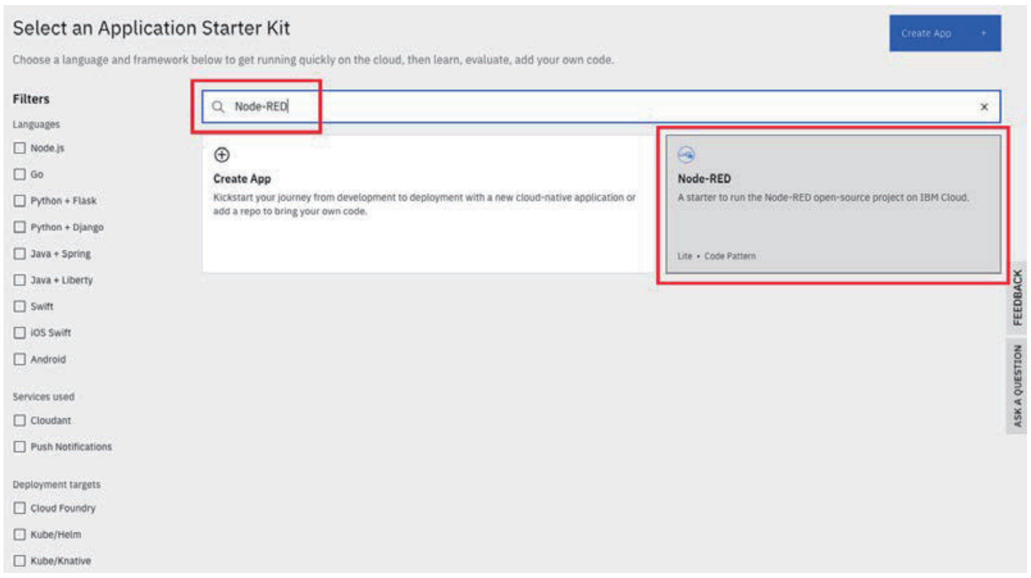
Getting Started

Starter Kits

Expedite your app development with pre configured working code for adding Push notifications, app usage analytics, crash analytics, chat bot, visual recognition and more.

Figure 6.3 – Get a Starter Kit button

4. You can find Node-RED if you type `Node-RED` into the search text box. Once you've found it, click on the **Node-RED** panel:


Select an Application Starter Kit

Choose a language and framework below to get running quickly on the cloud, then learn, evaluate, add your own code.

Node-RED

Create App

Kickstart your journey from development to deployment with a new cloud-native application or add a repo to bring your own code.

Node-RED

A starter to run the Node-RED open-source project on IBM Cloud.

Lite • Code Pattern

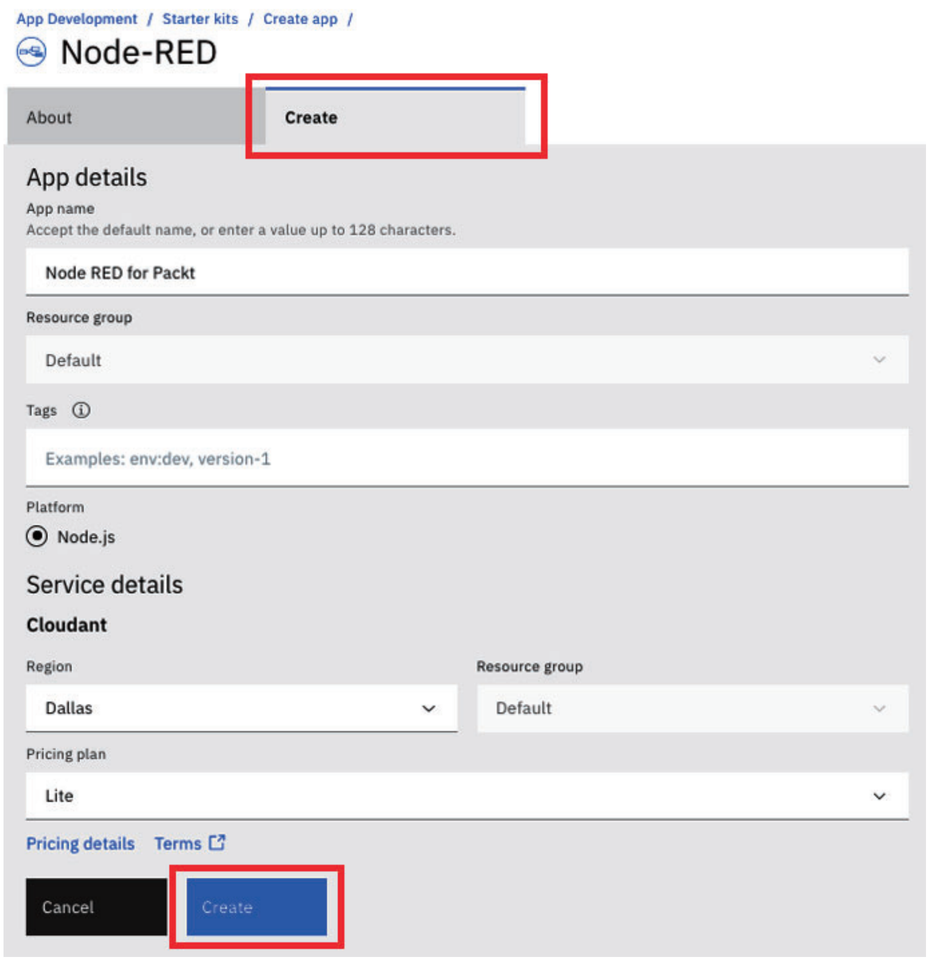
Figure 6.4 – Node-RED Starter Kit

5. After clicking on the **Node-RED** panel, we need to set some items.

   You can freely change each item by providing your own values, but in this chapter, the values that have been set here will be used for explanation purposes.

   See *Figure 6.5* for the settings and values to configure. Please note that once they are set, these items cannot be changed later.

6. After setting all the items, click the **Create** button:


App Development / Starter kits / Create app /

Node-RED

About | Create

App details

App name: Node RED for Packt
Accept the default name, or enter a value up to 128 characters.

Resource group: Default

Tags: Examples: env:dev, version-1

Platform: ● Node.js

Service details

Cloudant

Region: Dallas

Resource group: Default

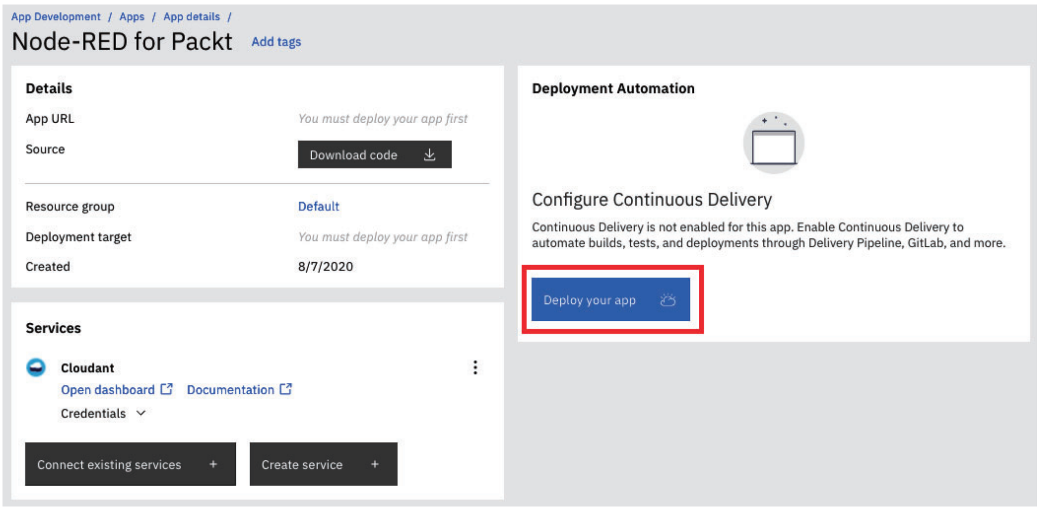
Pricing plan: Lite

Figure 6.5 – Create Node-RED as a Node.js application

You have now created the framework for the applications that make up Node-RED. After this, you will be redirected to the **App Details** screen automatically, where you will be able to see that the **Cloudant** instance of the linked service has also been provisioned.

However, only the application source code and the instance of the cooperation service are created, and they haven't been deployed to the Node.js execution environment on IBM Cloud yet. The actual deployment will be done when the CI/CD toolchain is enabled.

7. When everything is ready, click on the **Deploy your app** button in the center of the screen to enable it:



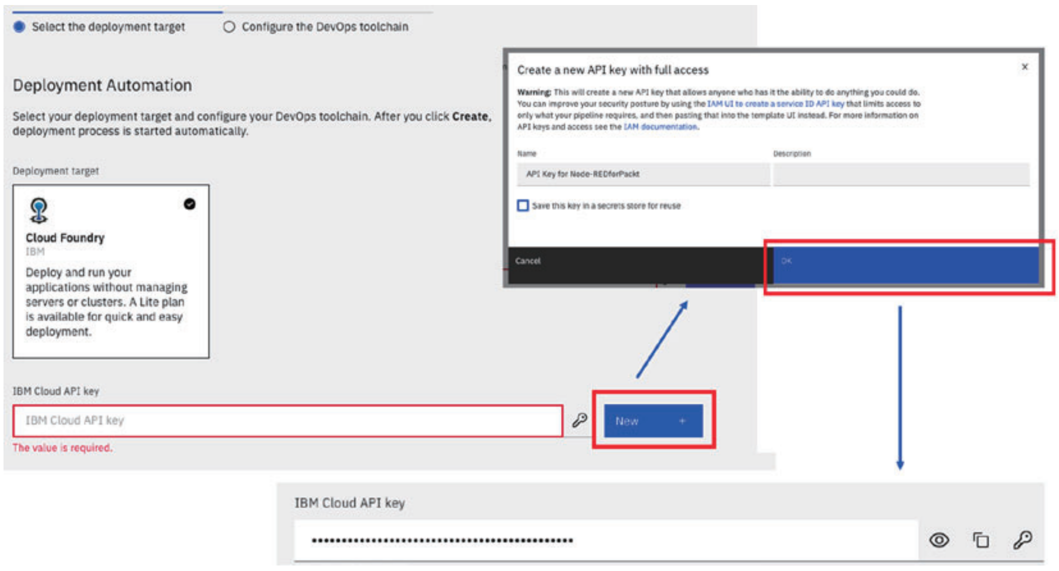
Continuous Delivery is not enabled for this app. Enable Continuous Delivery to automate builds, tests, and deployments through Delivery Pipeline, GitLab, and more.

Figure 6.6 – Deploying your Node-RED application

8. After clicking the **Deploy your app** button, move to the application settings window.

9. You will be asked to create an IBM Cloud API Key. Don't worry about this, as one will be generated automatically. Click the **New** button to open a new popup window, and then the **OK** button on the popup window. Once you do this, an IBM Cloud API Key will be generated:

> **IBM Cloud API Key**
> The IBM Cloud API Key is used to control your IBM Cloud account and various services (for example, it's Cloud Foundry in this tutorial). You can use this to issue a token for external access to services on IBM Cloud, for example. You can find out more about the IBM Cloud API Key here: `https://cloud.ibm.com/docs/account?topic=account-manapikey`.


Select the deployment target

Configure the DevOps toolchain

Deployment Automation

Select your deployment target and configure your DevOps toolchain. After you click **Create**, deployment process is started automatically.

Deployment target

Cloud Foundry
IBM
Deploy and run your applications without managing servers or clusters. A Lite plan is available for quick and easy deployment.

IBM Cloud API key: IBM Cloud API key

The value is required.

Create a new API key with full access

Warning: This will create a new API key that allows anyone who has it the ability to do anything you could do. You can improve your security posture by using the IAM UI to create a service ID API key that limits access to only what your pipeline requires, and then pasting that into the template UI instead. For more information on API keys and access see the IAM documentation.

Name: API Key for Node-REDforPackt

Description:

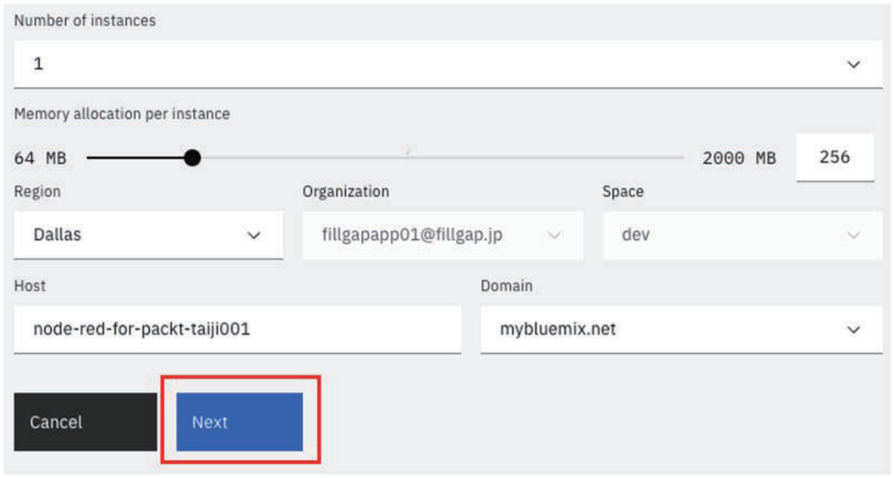
Save this key in a secrets store for reuse

IBM Cloud API key: ••

Figure 6.7 – Generating an IBM Cloud API Key

10. Select the resource spec on the window.

    This time, we are using IBM Cloud with a Lite account, so we have only 256 MB of memory available for all our services on IBM Cloud. So, if we use 256 MB for the Cloud Foundry Node.js service, we won't be able to use more memory for other services. But Node-RED needs 256 MB to run on IBM Cloud, so please use 256 MB here. It is already allocated 256 MB for the instance by default, so click the **Next** button, with no parameters changed:


Number of instances: 1

Memory allocation per instance: 64 MB – 2000 MB: 256

Region: Dallas

Organization: fillgapapp01@fillgap.jp

Space: dev

Host: node-red-for-packt-taiji001

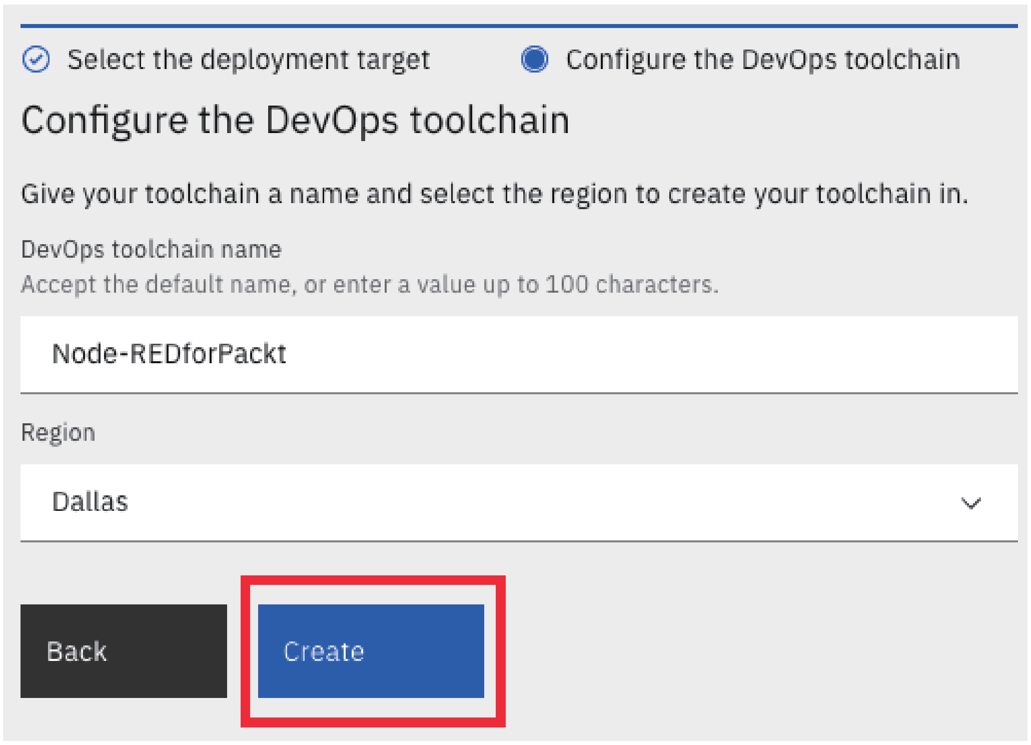
Domain: mybluemix.net

Figure 6.8 – Node.js runtime instance details

Once you've done this, a **DevOps toolchain** setting screen will be displayed.

11. Click the **Create** button, with the default values filled in.

    You can change the DevOps toolchain name to any name you like. This is the name that identifies the toolchain you've created in IBM Cloud:



Configure the DevOps toolchain

Give your toolchain a name and select the region to create your toolchain in.

DevOps toolchain name
Accept the default name, or enter a value up to 100 characters.

Node-REDforPackt

Region

Dallas

Figure 6.9 – Configure the DevOps toolchain window

Now, you are ready to use the environment (Node.js runtime and DevOps toolchain) to run the Node-RED application you created in the previous step. The Node-RED application you created is automatically deployed on the Node.js runtime through the toolchain.

12. Confirm that the **Status** that's displayed in the **Delivery Pipelines** (pipeline for executing each tool in the DevOps toolchain) area is **Success**, and click the toolchain's name (**Node-REDforPackt**, in this case) above it: