

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.

# 第十七讲 Node-RED (I) Lecture 17 Node-RED (I)

明玉瑞 Yurui Ming  
yrming@gmail.com

# 声明

## Disclaimer

- 本讲义在准备过程中由于时间所限，所用材料来源并未规范标示引用来源。所引材料仅用于教学所用，作者无意侵犯原著者之知识产权，所引材料之知识产权均归原著者所有；若原著者介意之，请联系作者更正及删除。

The time limit for the preparation of these slides incurs the situation that not all the sources of the used materials (texts or images) are properly referenced or clearly manifested. However, all materials in these slides are solely for teaching and the author is with no intention to infringe the copyright bestowed on the original authors or manufacturers. All credits go to corresponding IP holders. Please address the author for remedy including deletion have you had any concern.

# 背景

## Background

- ▶ Node-RED最初是由IBM新兴技术服务小组的Nick O'Leary和Dave Conway-Jones两位研究人员，于2013年初用于可视化和操作MQTT主题之间映射的概念验证而启动的一个辅助项目。由于可以轻松地进行扩展，很快就成为一种更通用的工具。在此情况下，其于2013年9月开源，然后在2016年10月成为JS基金会的创始项目之一。

Node-RED started in early 2013 as a side-project by Nick O'Leary and Dave Conway-Jones of IBM's Emerging Technology Services group. It began as a proof-of-concept for visualising and manipulating mappings between MQTT topics, quickly became a much more general tool that could be easily extended in any direction. It was open-sourced in September at the same year and later culminated in being one of the founding projects of the JS Foundation in October 2016.

- ▶ Node-RED名称中的“Node”部分反映了流/节点编程模型以及底层Node.JS运行时，但RED实质上并没有正式的名称，一种说法是英文Rapid Event Developer的首字母缩写。

The 'Node' part of the name “Node-RED” reflects both the flow/node programming model as well as the underlying Node.JS runtime. However, it never comes to a conclusion on what the 'RED' part stands for. One suggestion was the initialism of "Rapid Event Developer".

# 背景

## Background

- ▶ 基于流程的编程由J. Paul Morrison在20世纪70年代发明，其将应用程序的行为描述为黑盒网络。每个节点都有明确的功能；基于输入数据，节点做出相应的逻辑处理，然后输出数据。网络负责节点之间的数据流。

Flow-based Programming was invented by J. Paul Morrison in the 1970s, which is a way of describing an application's behavior as a network of black-boxes. Each node has a well-defined purpose: it does something with the given data and then passes that data on. The network is responsible for the flow of data between the nodes.

- ▶ 该模型非常适合视觉呈现，容易被广泛的用户使用。通过将问题分解为离散步骤，用户可以查看当前业务流程，但无需理解每个节点内的每一行代码。

Such a model suits well to visual representation and is accessible to a wider range of users. By breaking down a problem into discrete steps the user can check a flow and get a sense of what it is doing, without having to understand the individual lines of code within each node.

- ▶ Node-RED是基于流程的编程工具，由基于Node.js的运行时组成。用户可以将Web浏览器指向该运行时来访问流编辑器。

Node-RED is a flow-based programming tool, consisting of a Node.js based runtime. User can direct a web browser at the runtime library to access the flow editor.

# 安装

## Installation

- ▶ Node-RED基于浏览器来编程。在浏览器中，用户通过将节点从调色板拖到工作区中来创建应用程序，然后将节点连接在一起。通过单击，应用程序就会部署回运行它的运行时。同时，通过安装社区创建的新节点，可以轻松扩展节点选项板，并且一个用户创建的流可以作为JSON文件轻松共享。

Node-red is web-based programming. Within the browser users create applications by dragging nodes from the palette into a workspace and start to wire them together. With a single click, the application is deployed back to the runtime where it is run. The palette of nodes can be easily extended by installing new nodes created by the community and the flows users create can be easily shared as JSON files.

- ▶ 下面，我们说明如何在Windows环境中安装并设置Node-RED。注意：以下一些说明提到了“命令提示符”，它是指Windows cmd或PowerShell终端shell。首先，从Node.js官方主页下载Node.js的最新LTS版本，然后以本地管理员权限运行下载的MSI文件。

Now, we instruct on installing and setting up Node-RED in Windows environment. Note that some of the following instructions mention the "command prompt". Where this is used, it refers to either the Windows cmd or PowerShell terminal shells. First, Download the latest LTS version of Node.js from the official Node.js home page. Then run the downloaded MSI file with local administrator rights.

# 安装

## Installation

- ▶ 安装完成后，关闭所有打开的命令提示符并重新打开，以确保新的环境变量生效。然后打开命令提示符并运行以下命令以确保 Node.js 和 npm 正确安装：

- ▶ 使用 Powershell: `node --version; npm --version`
- ▶ 使用命令: `node --version && npm --version`

如果安装无误，用户应该收到类似于以下内容的返回输出：

- ▶ v18.15.0
- ▶ 9.5.0。

- ▶ After installation completes, close any open command prompts and re-open to ensure new environment variables are picked up. Then open a command prompt and run the following command to ensure Node.js and npm are installed correctly:

- ▶ Using Powershell: `node --version; npm --version`
- ▶ Using cmd: `node --version && npm --version`

One should receive back output that looks similar to the following is everything is correct:

- ▶ v18.15.0
- ▶ 9.5.0

# 运行

## Running

- 为方便起见，可选择将Node-RED安装为全局模块，并将命令node-red到系统路径中。上述操作可在命令提示符下执行以下命令完成：

For convenience, the user can choose to install Node-RED as a global module and adds the command node-red to the system path. Execute the following at the command prompt:

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

- Node-RED安装为全局npm软件包后，运行Node-RED的简单方法是在命令提示符中执行node-red命令：C:>node-red。但由于此种方式会将Node-RED日志输出到终端，因此用户必须保持终端打开才能保持Node-RED运行。注意，运行Node-RED将在当前用户的%HOMEPATH%文件夹中创建一个名为.node-red的新文件夹，可将其视为当前用户的Node-RED配置的主文件夹。

Once installed, the simple way to run Node-RED is to use the node-red command in a command prompt: C:>node-red. However, this will output the Node-RED log to the terminal, you must keep the terminal open in order to keep Node-RED running. Note that running Node-RED will create a new folder in your %HOMEPATH% folder called .node-red. This is your userDir folder, think of it as the home folder for Node-RED configuration for the current user.



# 使用PM2

## Using PM2

- 如果在Windows平台开发Node-RED流或节点，可能会发现使用PM2运行Node-RED很方便。可以将其配置为在文件更改时自动重新启动，且始终保持 Node-RED 运行并管理日志输出。要使用PM2，推荐使用下面命令将PM2安装为全局模块：

If to develop Node-RED flows or nodes on the Windows platform, the user may find it helpful to use PM2 to run Node-RED. This can be configured to automatically restart when files change, always keep Node-RED running and manage log output. To use PM2, it is recommended to install PM2 as a global module using the following command:

```
npm install pm2 -g
```

- 下面表格总结了使用PM2管理Node-RED的常见命令：

The following table summarizes the commonly-used command of PM2 to manage Node-RED:

功能（Usage）	命令（Command）
启动 Node-RED（Start Node-RED）	pm2 start node-red -v
停止（Stop Node-RED）	pm2 stop node-red
查看 Node-RED 日志（View Node-RED logs）	pm2 logs node-red
查看 Node-RED 状态（View Node-RED status）	pm2 show node-red



# 使用PM2

## Using PM2

- ▶ PM2通过特定的语法来区分传递给自身的选项和传递给所管理模块的选项。一般而言，传递给PM2的选项会放在--（双破折号）之前，而在--之后的选项则会直接传递给被管理的模块。例如，用户可以通过如下命令在特定端口如3000运行Node-RED服务：

PM2 uses a special syntax to distinguish between options passed to PM2 itself and options passed to the managed module. In general, options intended for PM2 are placed before the -- (double dash), while options after the -- are passed directly to the managed module. For example, users can run Node-RED service on a specific port for instance 3000 by running:

```
pm2 start node-red -- -v --port 3000
```

- ▶ 通常情况下，PM2会使用默认的node解释器，但对于某些服务模块而言，作者可能会通过批处理脚本包装，在设置相应环境变量或配置选项之后，再调用node执行相关JS脚本。但用PM2管理时，默认的解释器可能会产生问题，因此，要更改成合适的解释器，例如：

Generally, PM2 uses the default node interpreter to execute the scripts. However, for some service module, the author might choose to wrap it in batch script in order to make the environment variable effective or configure some parameters first, then launch the JS scripts by pointing to node executable. If such service modules are managed by PM2, the default node interpreter might cause some problem. Therefore, users need to adjust the interpreter accordingly, for example:

```
pm2 start node-red --interpreter "C:\Windows\System32\cmd.exe" -- -v
```

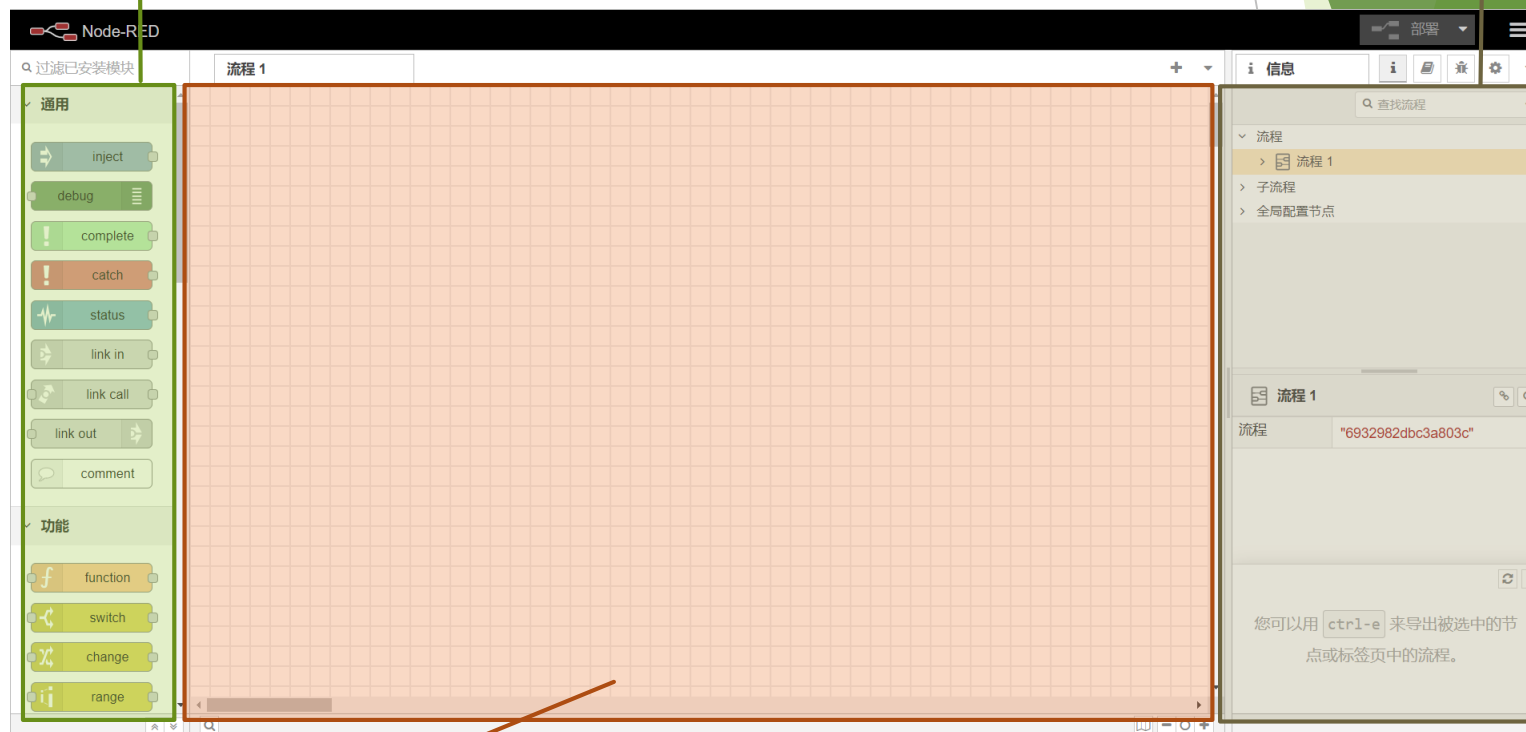
# 布局 Layout

- ▶ 右侧展示了Node-RED编辑器的布局, 各区域的功能如图注所示:

The right figure demonstrates the layout of the Node-RED editor, which is parted into several functional areas as stated:

调色板区, 包含了流式编程中的常用节点, 并按照功能进行分类。  
The palette area contains all the nodes for flow-based programming. And they have been sorted into different categories according to their functionalities.

边栏区, 汇聚了包括节点信息、调试信息、配置节点、上下文数据在内的若干功能面板。  
The sidebar area which aggregates functional tabs such as node information, debug messages, configuration nodes, context data.



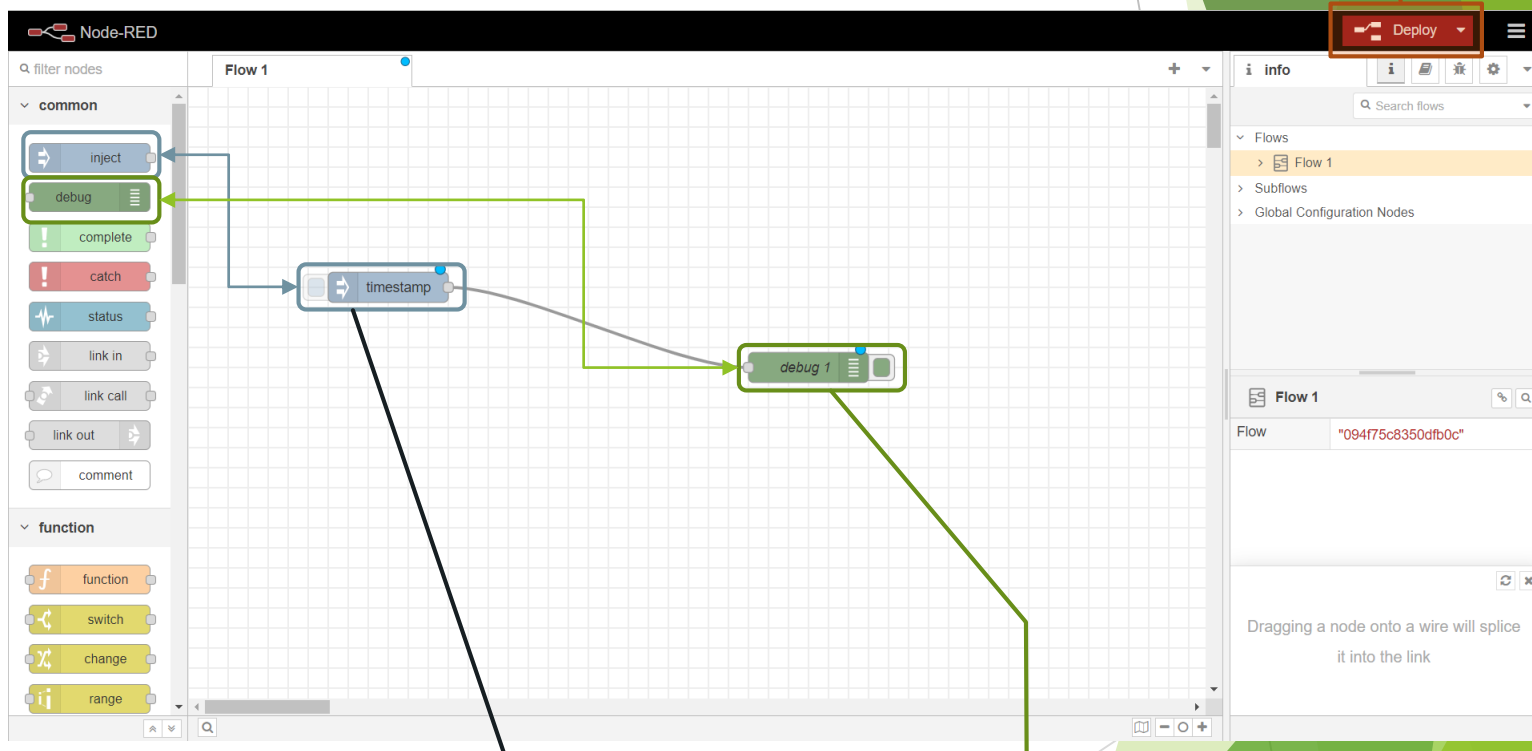
工作区, 用户可将节点在工作区组成流进行编程。  
The workspace where nodes are arranged into flows for performing various tasks

# 创建流

## Creating a Flow

- Node-RED基于流的编程可通过将节点直接拖入工作区的方式进行，右侧展示了由“注入”节点与“调试”节点组成的最简单的流的情况：

The flow-based programming in Node-RED is done by dragging nodes from the palette into the workspace. The right figure demonstrates a simple flow consists of inject node and debug node.



部署按钮，当流创建好之后，需要部署才能使之生效。  
Deployment button. To enable the flow, you must deploy it to server to activate the functionality.

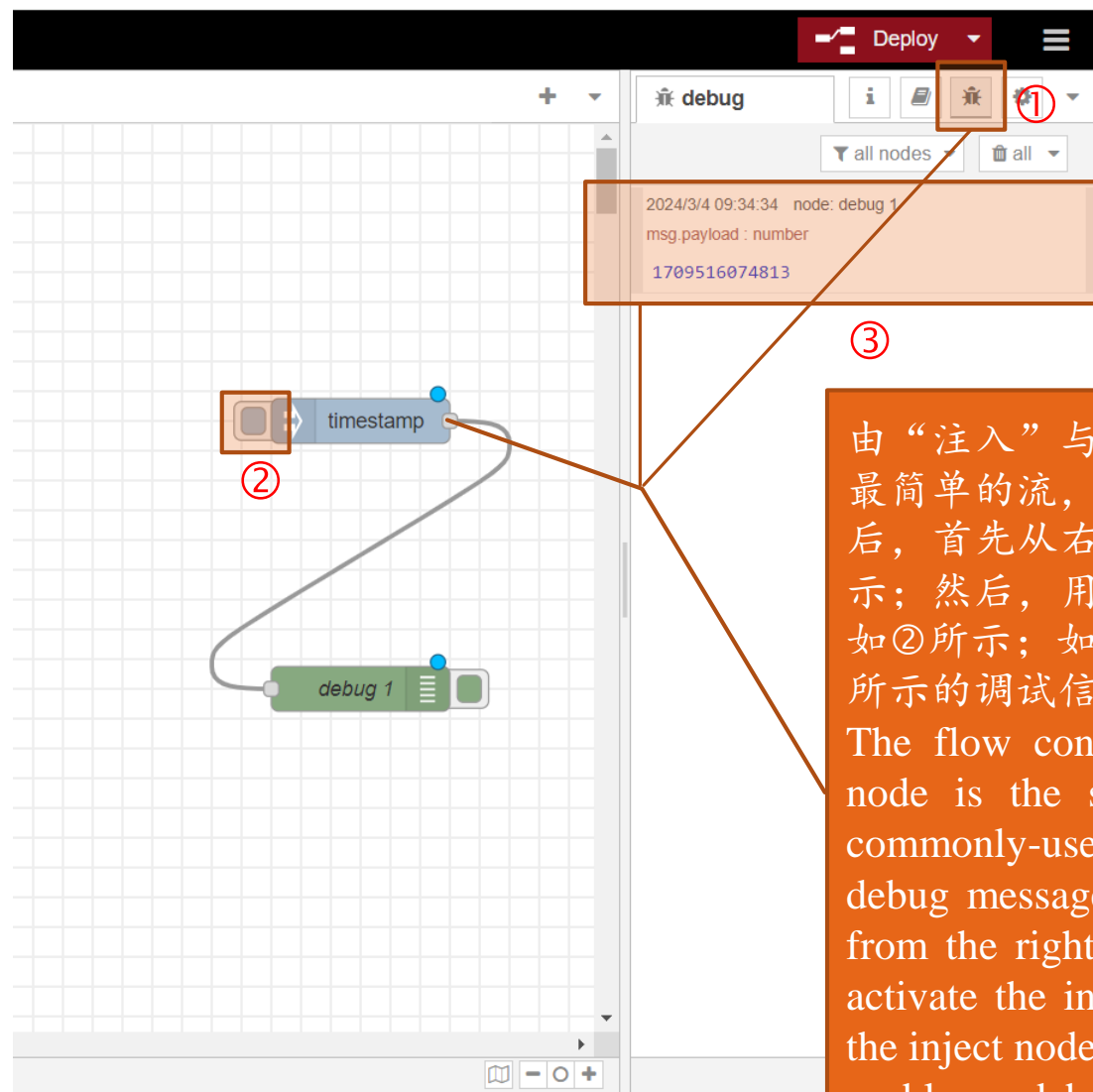
“注入”按钮默认情况下，注入的是时间戳。  
The default behavior for “injection” button is to insert the time stamp.

调试按钮，默认输出接收到的信息。  
Debug button. The default behavior is to output received message.

# 布局 Layout

- 右图展示了如何创建与使能Node-RED中最简单的流的情况：

The right figure demonstrates the creation and activation of a simple flow in Node-RED:



由“注入”与“调试”节点所组成的流虽然是最简单的流，但实际上是最常用的流。部署之后，首先从右侧栏选择调试信息面板，如①所示；然后，用户单击“注入”节点的头部按钮，如②所示；如果没有问题，则用户能看到如③所示的调试信息。

The flow consisting of inject node and debug node is the simplest flow however the most commonly-used one. After deployment, the debug message tab needs to be brought to front from the right side bar, as shown in 1. Second, activate the inject node by hitting the header of the inject node. At last, if everything runs without problems, debug message will pop up instantly as indicated by 3.

# 编辑节点

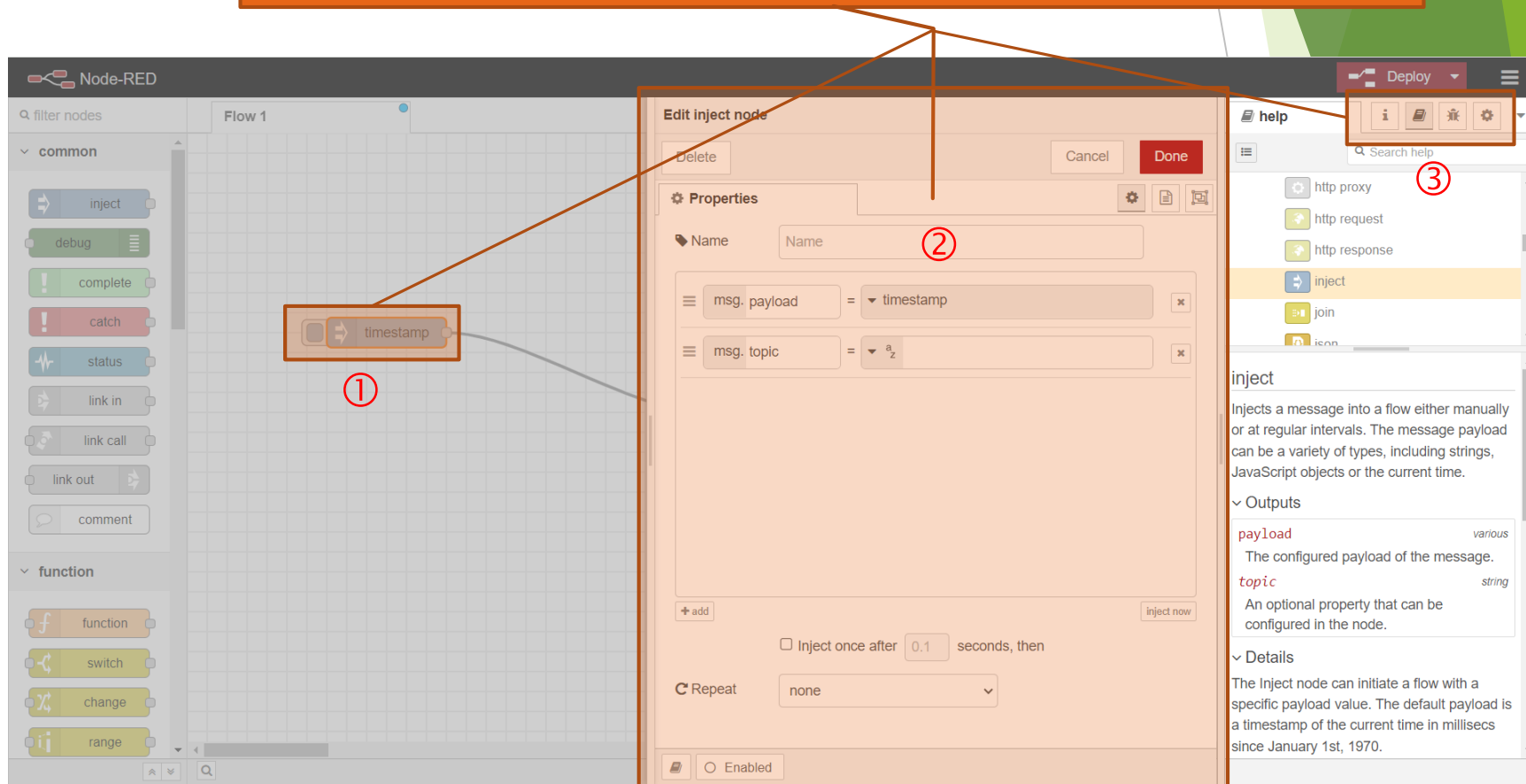
## Editing a Node

- 通常，节点默认的功能只是尝试涵盖最基本的情况，针对具体场景，读者可能需要编辑节点，这通过双击节点得到实现：

Generally, the default case for most nodes only covers the most general situation. For specific needs, the user might need to edit the node for customization. And this can be achieved by double-click the node:

以“嵌入”节点①为例，双击节点后，可打开如②所示的属性页，可以定制嵌入的内容，如时间戳，如固定的数字或字符串等。此时，右边侧栏③的功能面板会与当前被编辑的节点相关联。

Exemplified by the “inject” node, after double-click the node ①, a property dialog or page is popped up for configuring some properties of the node, as indicated by ②. Now the functional tabs aggregated in the side bar will automatically associated with the current editing node.

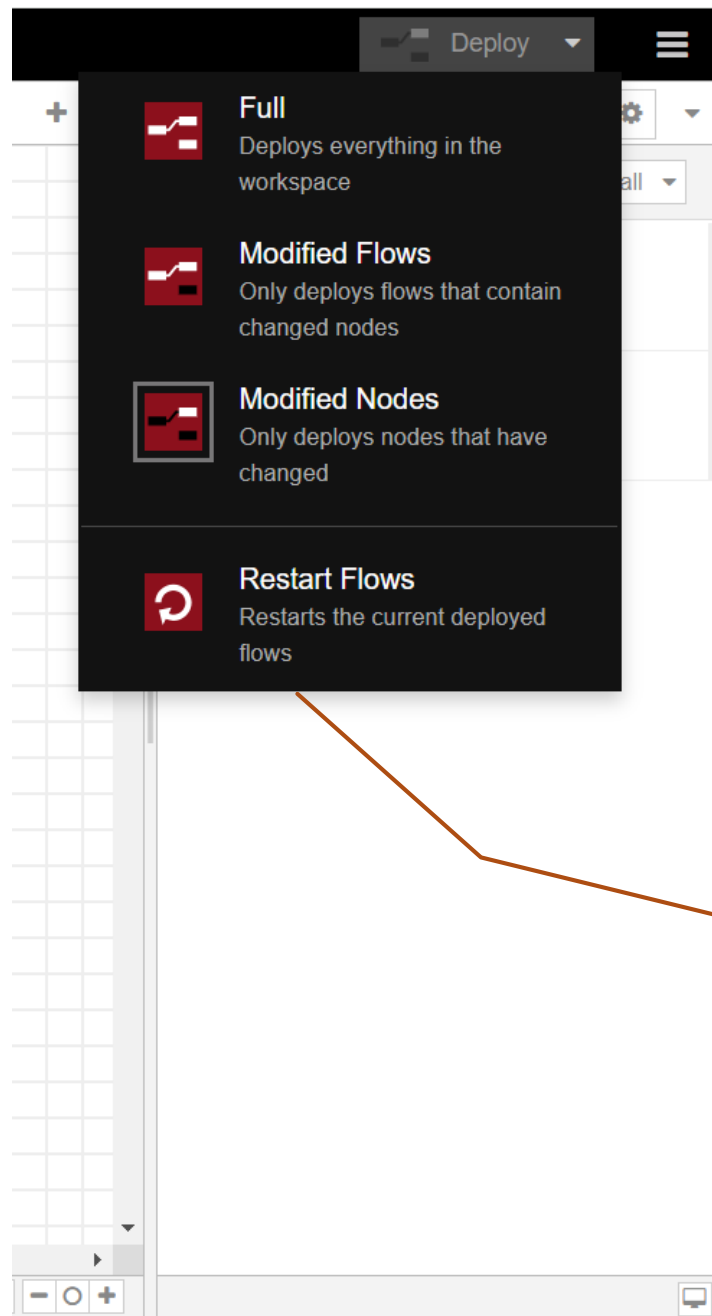


# 编辑节点

## Editing a Node

- ▶ 当我们修改节点之后，我们需要重新部署。部署有多种类型选项，特别在大型系统中，选择合适的部署类型会降低对系统的影响。右图显示了常见的部署类型：

After we modified the node, we need to redeploy the system. There are several options, which favors the minimum impact on large systems especially the most appropriate option is chosen. The right figure shows the available deployment options:



部署类型第一种是完全类型，即将项目或工程中涉及的流全部部署，这通常是在对项目全局性逻辑做重大修改后的选项。第二种是修改的流。当项目中有多条流时，该选项能加快部署速度，还能减少对未修改的流的影响。第三种是修改的节点。这是对项目有所修改时的最小部署。最后一项是仅重启，即当某种原因系统表现异常时，尝试重启以解决问题。

The first choice for deployment is the full option, which means you will do over all the system or project. This option is preferred if significant modification is made to the global logic of the system. The second option is just confine the modified node, especially when there are many flows in the project. This option can mitigate the impact on unmodified flows. The third option is the modified node. This is the minimum deployment. The last one is the restart one. Especially when the system behaves abnormally, to try to restart from scratch might be a better standpoint.

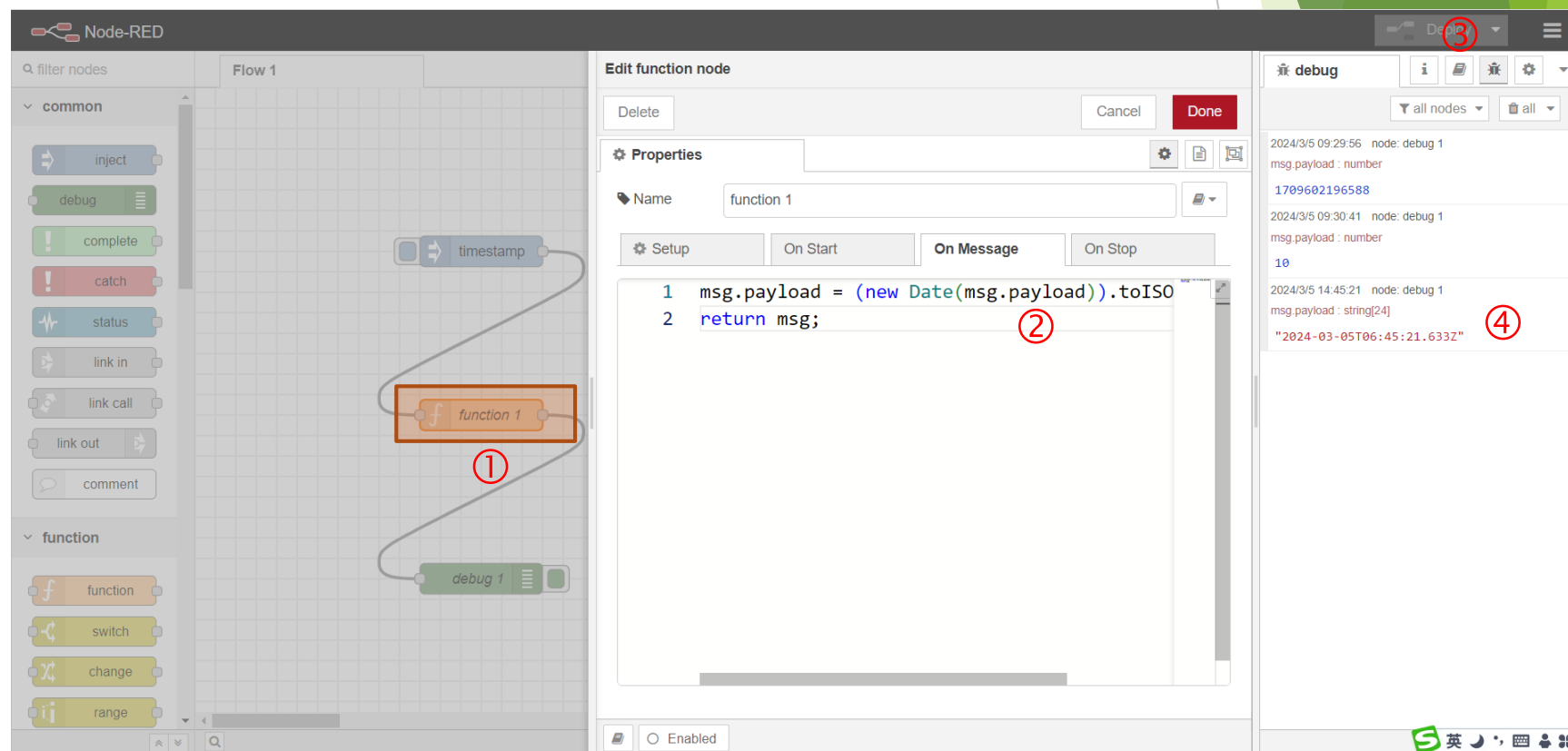


# 编辑节点

## Editing a Node

- ▶ Node-RED 提供了丰富的节点供工程师创建灵活的流以应对复杂的场景。以例如函数节点为例，实际上，用户可以根据业务需求编写任意的特定函数以满足要求。如将注入的时间戳由 Unix 格式改为 ISO 格式。右图展示了相应的例子。在右图中，通过编写代码，实现了期望的功能。

The Node-RED provides rich nodes to support constructing versatile flows for coping with complicated scenarios. Exemplified by function node, actually, users can customize it according to their business logic. For example, to modify the timestamp from the Unix format into ISO format, you can do it as follows. By programing the functionality of the node, it achieves desired output, verified by the debug message in ④



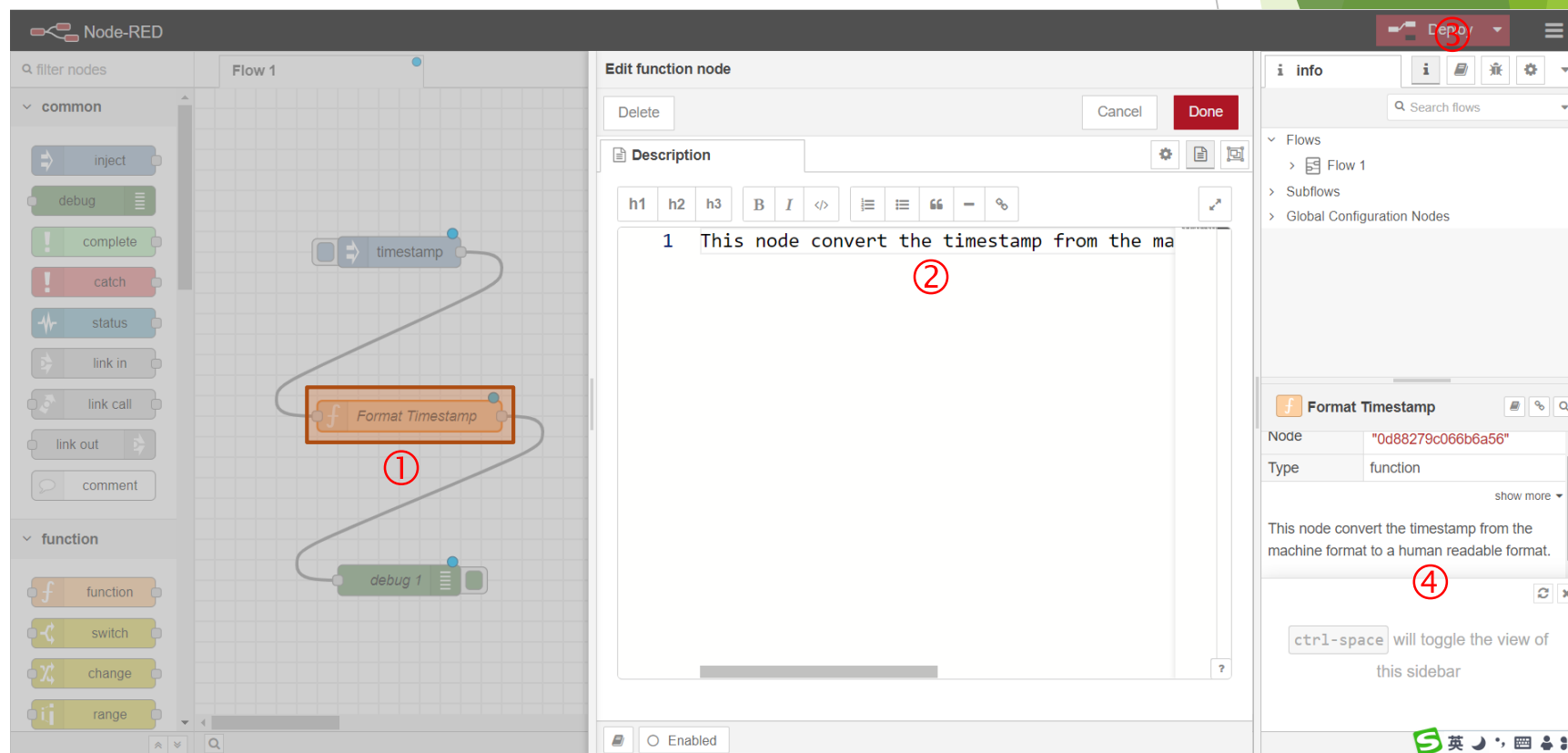


# 编辑节点

## Editing a Node

- ▶ 尽管在简单的应用中，读者可以保留节点的默认名称。但是在大型项目里，由于涉及的节点数目众多，因此最好给节点起有意义的名称以便在后续运维中知道节点的意义。同时，尽可能地对节点的功能进行注释，以减少项目后续运维或升级的难度。右图是一例子，展示了如何命名节点与进行注释。

User can retain the default name of nodes, however, it is highly recommended in large project, adopting some nomenclature is vitally important, especially from maintaining perspective. Meanwhile, to comment on the node is also helpful in the future maintaining or updating work. The right figure demonstrates how to name and comment a node.

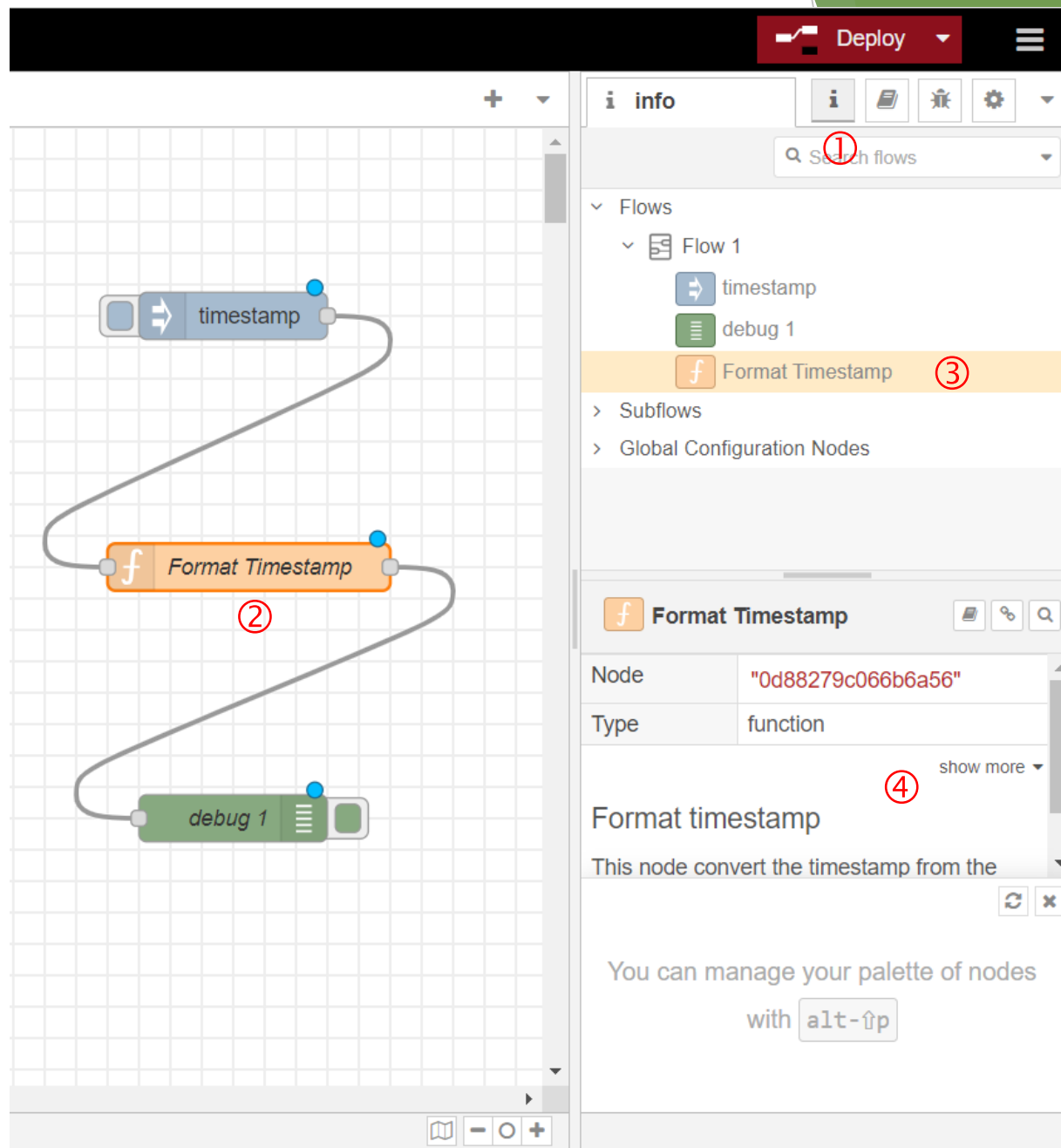


# 信息栏

## Info Tab/Sidebar

- ▶ 尽管有信息侧栏这种提法，但实际上右侧栏是一些功能页的集合。大家经常用的是信息栏，可以方便地显示节点的信息。

Although people might refer to information sidebar, but actually the right bar is an aggregation of many functional tags. Information sidebar or tag is quite used to inspect the information of the node(s) and most the attributes are just read-only.

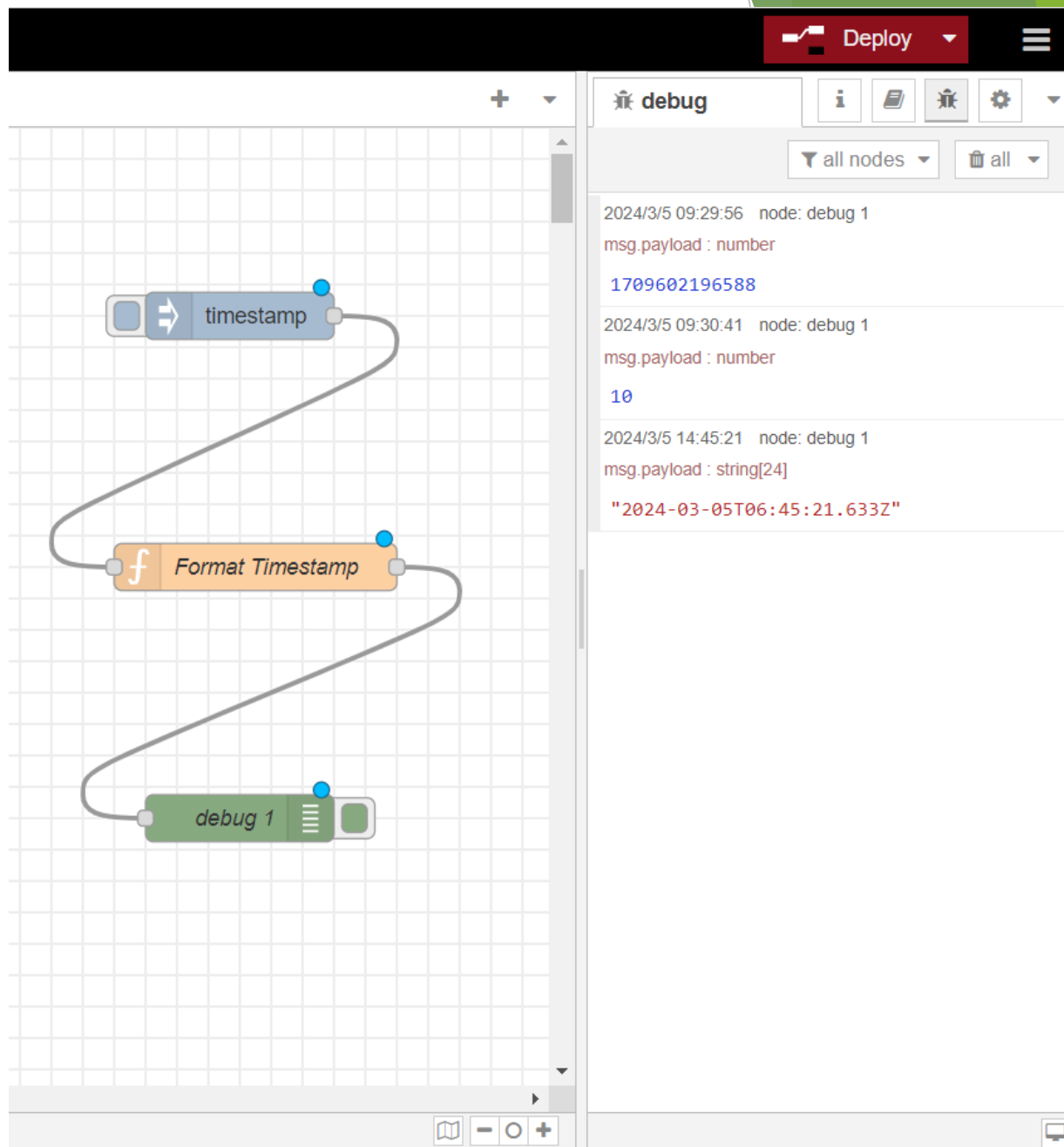


# 调试栏

## Debug Tab/Sidebar

- ▶ 实际上，调试在项目开发过程中是必不可少的。尽管通过拖拽基于流的编程简化了许多工作，但调试也是非常重要的。通过添加调试节点并观察调试页的调试输出，为定位问题提供了有力的支持。

Debug is an indispensable work during project. Although flow-based programming via dragging and dropping nodes simplify the process, however, without debugging, the system is buggy. By inserting debugging node and observe the output from the debug tab, it gives quite incredible support for locating the problem.

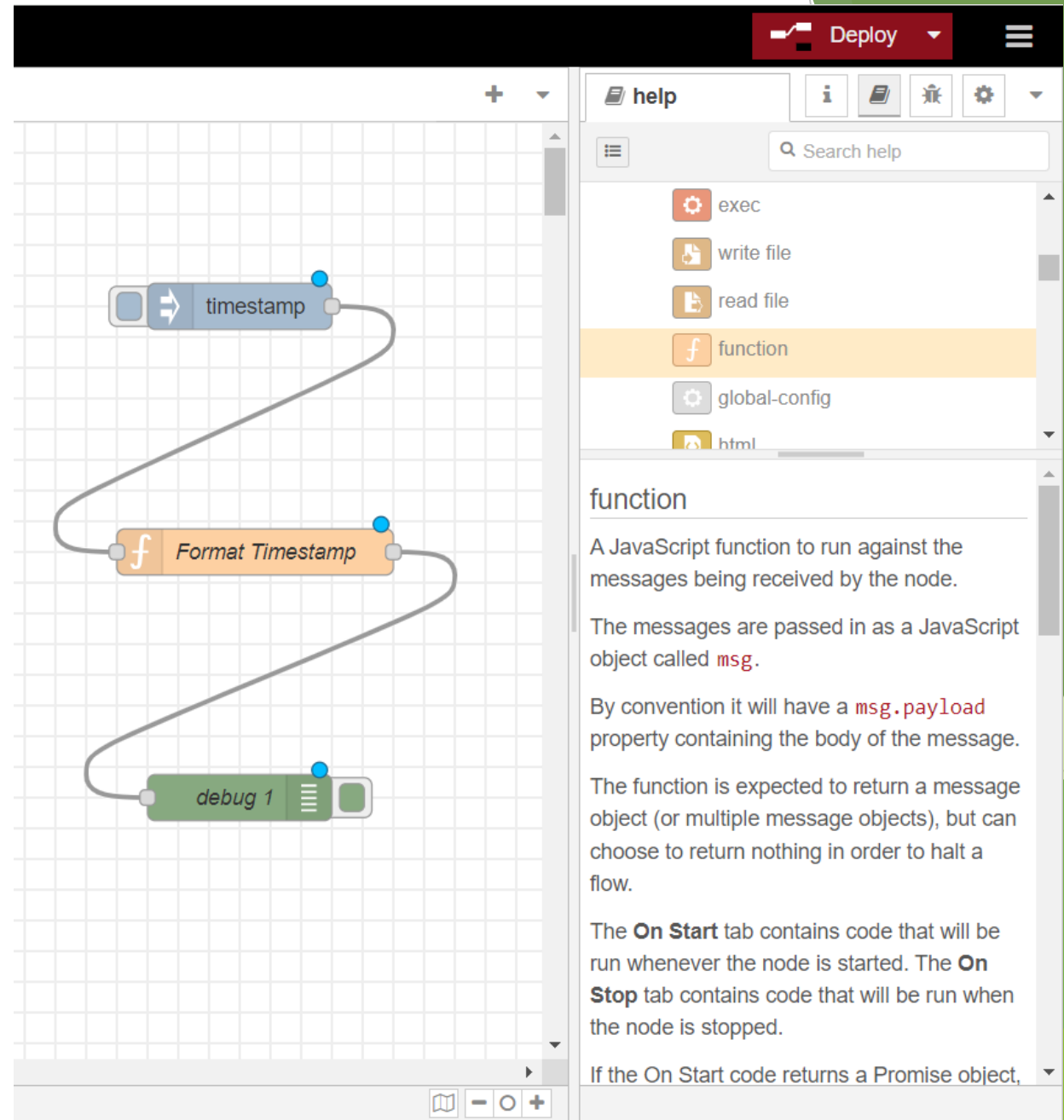


# 帮助栏

## Help Tab/Sidebar

- 通常，节点的名称基本可以反映节点的功能，特别是对于有经验的用户。但一些情况下，用户可能需要知道关于节点功能的细节。此时，通过查看帮助，用户通常会对节点的功能有足够好的了解。

Generally, an experienced user can guess the functionality of the node by just inspecting its default name. However, in some scenarios, users might want know about the nodes. In this case, the help tap/side is where the users should turn to. By going through the help text, users usually get a fairly good understanding of the nodes.



# 配置栏

## Config. Tab/Sidebar

- 对于有些节点，其功能是比较固定的，但其正常工作需要配置一些参数才可以。因此Node-RED提供了配置栏或配置页来配置或检查可配置的节点。我们以MQTT相关节点来说明。

Some nodes are of fixed functionalities, however, to functioning properly some configurations need to be performed. To cater to this, Node-RED provides the configuration sidebar or tab to fulfill it. We exemplify this by using MQTT related nodes.

- 由于Node-RED本身并不是设计成独立的MQTT服务器/代理使用，因此我们首先下载Mosquitto，作为外部的MQTT代理。通常，下载后我们首先需要安装，然后启动Mosquitto服务，最后将Mosquitto安装文件夹加入Path环境变量，方便进行相关测试。

Node-RED itself is not designed to function as a standalone MQTT server or broker, we need to download for example Mosquitto as an external MQTT proxy to leverage Node-RED's flow-based automation features. Usually, after download and install Mosquitto, we need to start the Mosquitto service. At last, we need to add the installation folder as part of the values for the path environment variable.

# 配置栏

## Config. Tab/Sidebar

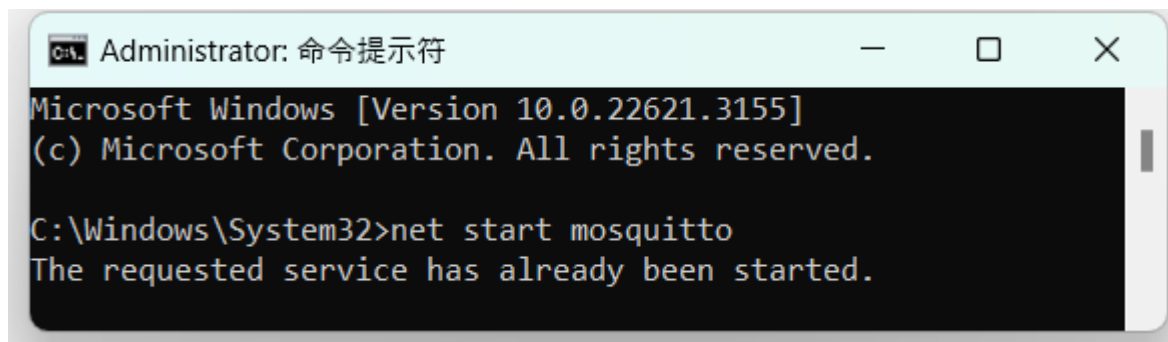
- Mosquitto的官网地址如下，用户直接下载安装就可以：

The user can download and install Mosquitto from the following website:

<https://mosquitto.org/download/>

- 注意安装后启动服务需要管理员权限：

Notice administrator privilege is require to start the service:

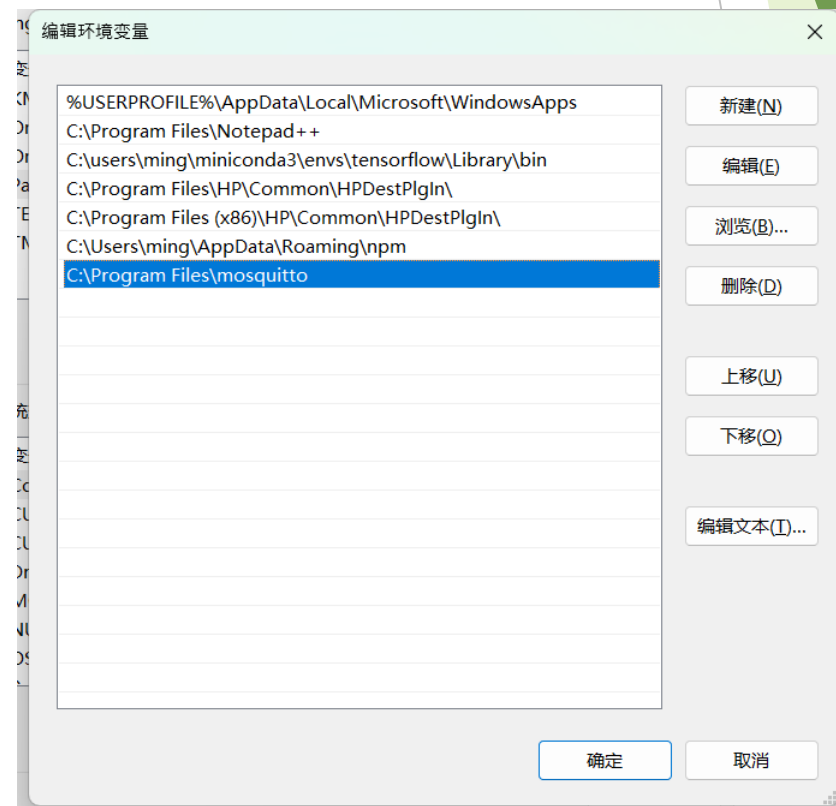


```
Administrator: 命令提示符
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>net start mosquitto
The requested service has already been started.
```

- 最后将Mosquitto安装路径加入环境变量：

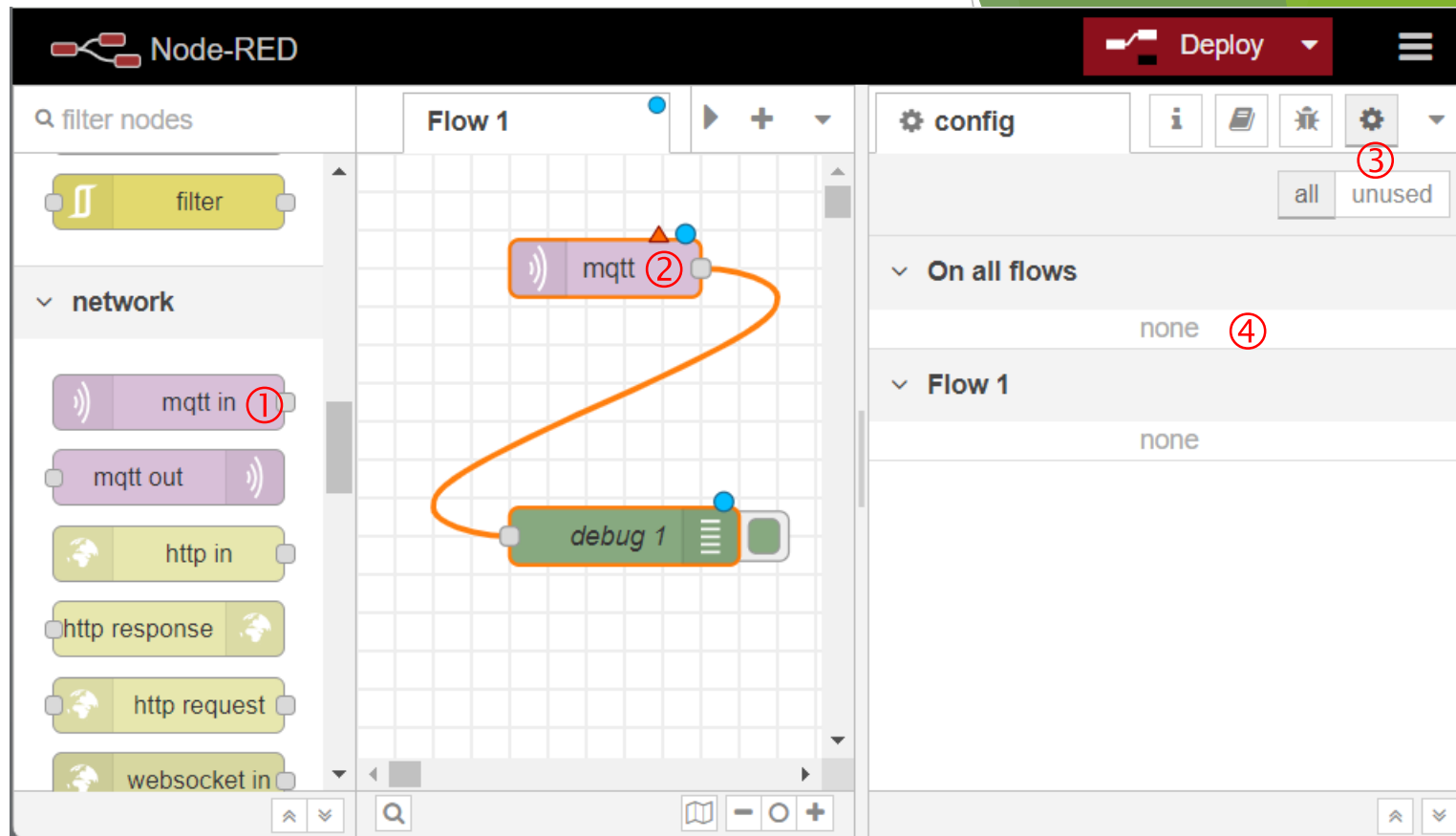
Add the installation directory of Mosquitto to the environment variable:



## 配置栏

## Config. Tab/Sidebar

- 下面我们来建立最简单的MQTT一条流。将mqtt in节点①拖入工作区，同时加入调试节点，然后将两个节点连接起来。注意，MQTT节点上的小红三角表示节点有错误，或者是编辑错误，或者是配置错误。我们选择③配置页，可以看到没有任何配置信息，如④所示。



- Now we create the simplest flow involving MQTT. First, drag the mqtt in node into the workspace, as well as the debug node, wire them together to form the simplest flow. Note the small red triangle on the top of the node means there is an error, either an error during editing the node or configuring the node. We choose the configuration tab, however, since nothing done, it shows no information about node configured.

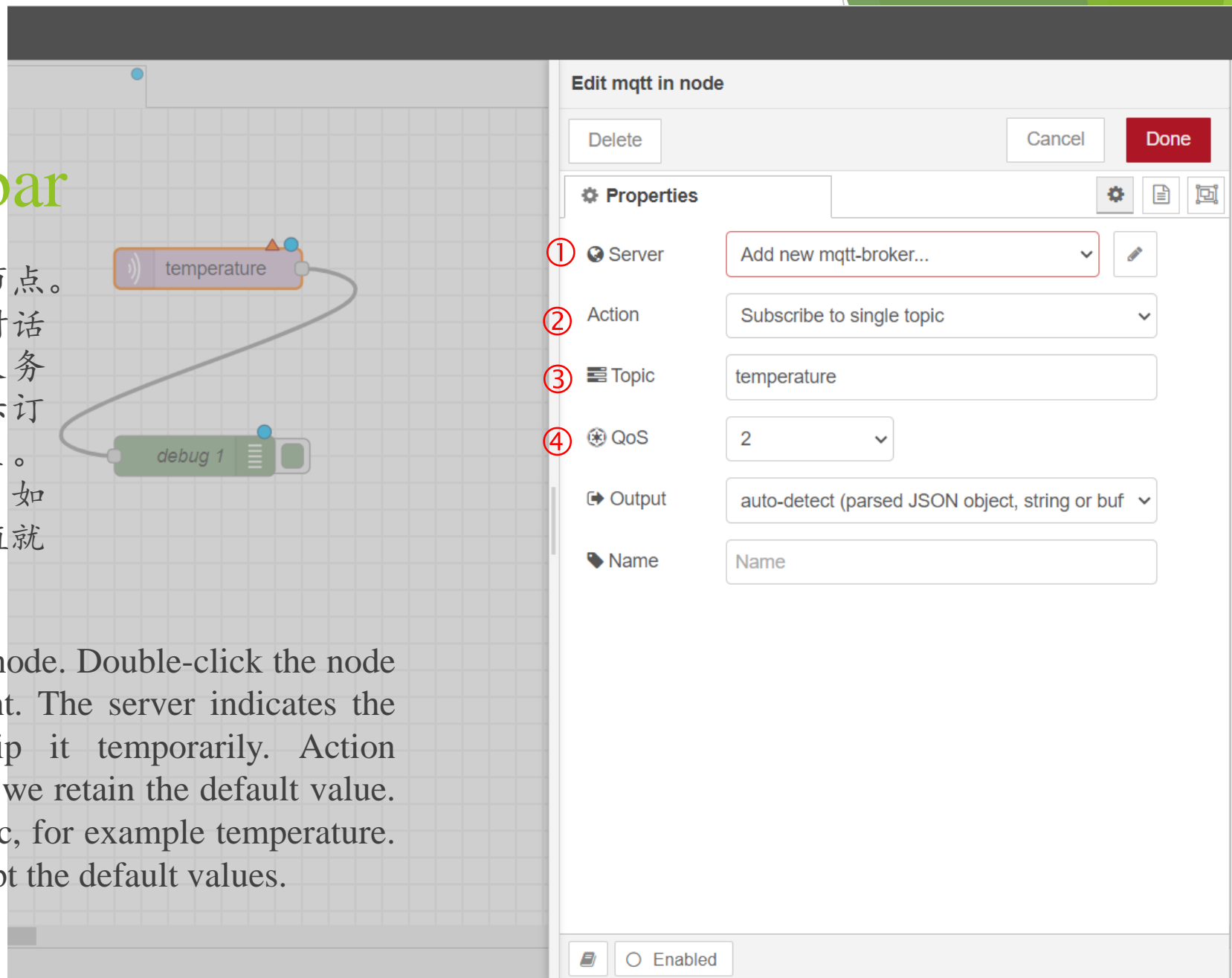


# 配置栏

## Config. Tab/Sidebar

- 接下来，我们来配置mqtt in节点。双击节点，打开如右所示的对话框。其中server表示MQTT服务器，先暂时略过。Action表示订阅的方式，可采用默认值。Topic根据业务决定，例如temperature。其他采用默认值就好。

- Now we configure the mqtt in node. Double-click the node to open the edit dialog as right. The server indicates the MQTT server/proxy. We skip it temporarily. Action indicates the subscription type, we retain the default value. Topic is up to the business logic, for example temperature. For all other fields, we just adopt the default values.



# 配置栏

## Config. Tab/Sidebar

- ▶ 我们通过单击上张幻灯片中Server后的图标，打开新建MQTT代理对话框。具体操作，我们在注释框里作了说明。

- ▶ By clicking the icon to the end of the server edit box, we open the add new mqtt-broker config node dialog. Detailed configurations are highlighted in the comment boxes.

The screenshot shows the 'mqtt-broker config node' dialog box. It has three tabs: 'Connection', 'Security', and 'Messages'. The 'Connection' tab is active. The dialog includes fields for 'Name', 'Server' (with a globe icon), 'Port', and 'Keep Alive' (with a heart icon). There are checkboxes for 'Connect automatically', 'Use TLS', and 'Use clean sess'. The 'Security' tab is highlighted with a red circle 4. The 'Server' field is highlighted with a red circle 2. The 'Port' field is highlighted with a red circle 3. The 'Keep Alive' field is highlighted with a red circle 1. The 'Messages' tab is also visible. The dialog has 'Cancel' and 'Add' buttons at the top right. The background shows a Node-RED flow with a 'temperature' node and a 'debug 1' node.

代理名称，例如Mosquitto  
Name of the server, for example, Mosquitto

① Name

② Server

③ Port 1883

④ Security

代理地址，IP或域名  
Address of the proxy, IP or domain name

如果需要安全特性，如用户名与密码，可在此配置  
Security features such as username and password can be configured here

代理端口，可采用知名端口  
Port of the proxy, For example well-known port defined by RFC

0 nodes use this config

On all flows

# 配置栏

## Config. Tab/Sidebar

► 右图展示了配置好之后的情况。

Right figure shows the configuration details.

The screenshot displays the Node-RED web interface. On the left, the 'network' category in the node palette is expanded, showing various MQTT-related nodes like 'mqtt in', 'mqtt out', 'http in', 'http response', 'http request', 'websocket in', 'websocket out', 'tcp in', 'tcp out', 'tcp request', 'udp in', and 'udp out'. The main workspace, labeled 'Flow 1', contains a 'temperature' node connected to a 'debug 1' node. On the right, the 'Edit mqtt-broker node' configuration panel is open. It features tabs for 'Properties', 'Connection', 'Security', and 'Messages'. The 'Properties' tab is active, showing fields for 'Name' (Mosquitto), 'Server' (localhost), 'Port' (1883), 'Protocol' (MQTT V3.1.1), 'Client ID' (Leave blank for auto generated), 'Keep Alive' (60), and 'Session' (Use clean session). The 'Connection' tab is also visible. On the far right, the 'config' sidebar shows a list of configurations. The 'Mosquitto' configuration is highlighted with a red box, and the '1' next to it is also highlighted. The 'config' sidebar also shows a 'Flow 1' section.

Node-RED interface showing the configuration of an MQTT broker node.

The left sidebar displays the 'network' category, listing various MQTT-related nodes (mqtt in, mqtt out, http in, http response, http request, websocket in, websocket out, tcp in, tcp out, tcp request, udp in, udp out).

The main workspace shows a flow named 'Flow 1' containing a 'temperature' node connected to a 'debug 1' node.

The right sidebar shows the configuration details for the 'mqtt-broker' node, including the 'Properties' tab (Name: Mosquitto) and the 'Connection' tab (Server: localhost, Port: 1883).

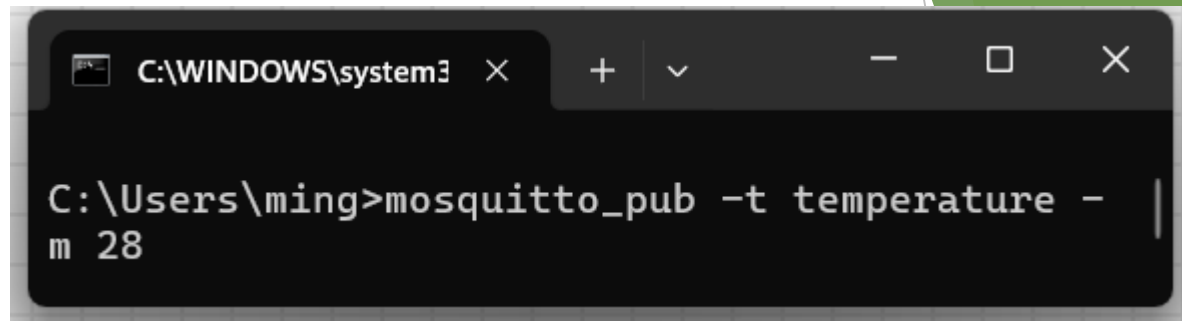
The bottom right corner shows the 'config' sidebar, indicating the configuration is applied to 'On all flows'.

# 配置栏

## Config. Tab/Sidebar

- 为测试配置的正确性，打开命令行，运行 mosquitto\_pub 程序并发布适当的消息，然后观察调试输出。

To test the configuration, open the command terminal and run the mosquitto\_pub executable to publish some message in accordance with the topic, then observe the output of debug node.



```
C:\WINDOWS\system32
C:\Users\ming>mosquitto_pub -t temperature -m 28
```

