

Node-RED 使用手册

1.	功能节点库介绍	2
2.	选择合适节点	4
3.	配置节点属性	5
4.	连接和布置节点	6
5.	项目和案例导出	6
6.	项目和案例导入	9
7.	项目部署运行	10
8.	流程的添加和删除	10
9.	控制节点有效无效	11
10.	子流程的建立	11
11.	卸载安装节点	14
12.	进入 UI 界面	16
13.	Node-RED 使用示例	20
13.1.	CAN 总线收发	20
13.1.1.	前置配置	20
13.1.2.	使用 Exec 节点收发 CAN 数据	23
13.1.3.	使用 node-red-contrib-socketcan 收发 CAN 数据	29
13.2.	modbus 转 mqtt	35
13.2.1.	前置配置	35
13.2.2.	modbus 数据转发 mqtt	36
13.3.	modbus RTU 转 modbus TCP	41
13.3.1.	前置配置	41
13.3.2.	modbus RTU 采集数据并转发到 modbus TCP	42

版本	描述	日期
V1.0	创建文档	2026-9-8

1. 功能节点库介绍

这里是一些在工业自动化、智能家居、物联网等领域中常见和实用的节点库，方便快速实现 PLC、传感器、数据库与云端的高效互通。

(1)、工业物联与智能楼宇协议节点

节点	协议
node-red-contrib-modbus	Modbus RTU / TCP / ASCII
node-red-contrib-opcua	OPC UA
node-red-contrib-bacnet	BACnet
node-red-contrib-knx-easy	KNX
node-red-contrib-m-bus	M-Bus
node-red-contrib-canal / canopen-mcan / socketcan	CAN / CANopen
node-red-dlt645	DL/T645-2007
node-red-contrib-meter-works	DL/T 614-2007

(2)、主流 PLC 协议节点

节点	品牌 / 协议
node-red-contrib-s7	西门子 S7
node-red-contrib-cip-ethernet-ip	罗克韦尔 AB
node-red-contrib-ads-client	倍福 / 欧姆龙 / 施耐德

节点	品牌 / 协议
node-red-contrib-mcprotocol	三菱 PLC
node-red-contrib-omron-fins	欧姆龙
node-red-contrib-keyence-plc	基恩士
node-red-contrib-panasonic-plc	松下 PLC
@kjalr/node-red-plcnext-connector	Phoenix Contact PLCnext

(3)、数据库与存储节点

节点	类型
node-red-node-sqlite	SQLite
node-red-node-mysql	MySQL
node-red-contrib-influxdb	InfluxDB
node-red-contrib-oracledb-mod	Oracle
node-red-contrib-buffer	缓存

(4)、云平台节点

节点	平台
node-red-contrib-aliyun-living	阿里
node-red-contrib-azure-iot-device	微软 Azure

(5)、消息与告警节点

节点	类型
node-red-node-email	邮件
node-red-contrib-wechats	企业微信
node-red-dx-dingding	钉钉

(6)、可视化节点

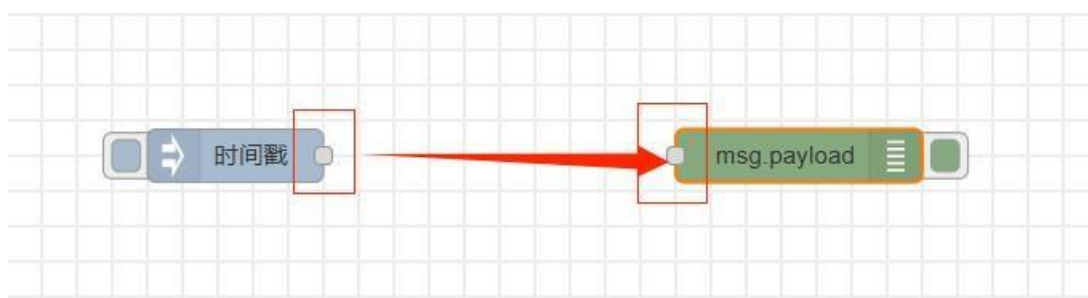
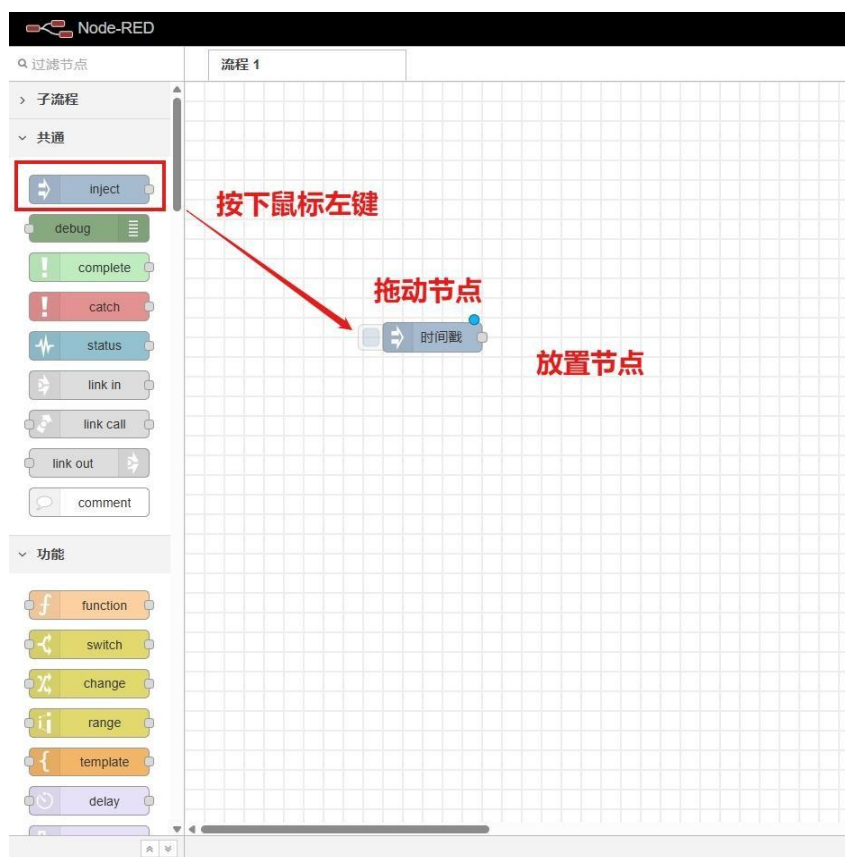
节点	类型
node-red-dashboard	经典可视化界面
@flowfuse/node-red-dashboard	新版仪表盘

2. 选择合适节点

在 Node-RED 中，有各种内置节点和我们提供的节点可供选择。选择节点时，需要考虑以下因素：

功能需求：确定需要的功能，然后在节点库中搜索相应的节点。例如，如果需要发送 HTTP 请求，可以选择 "HTTP Request" 节点。

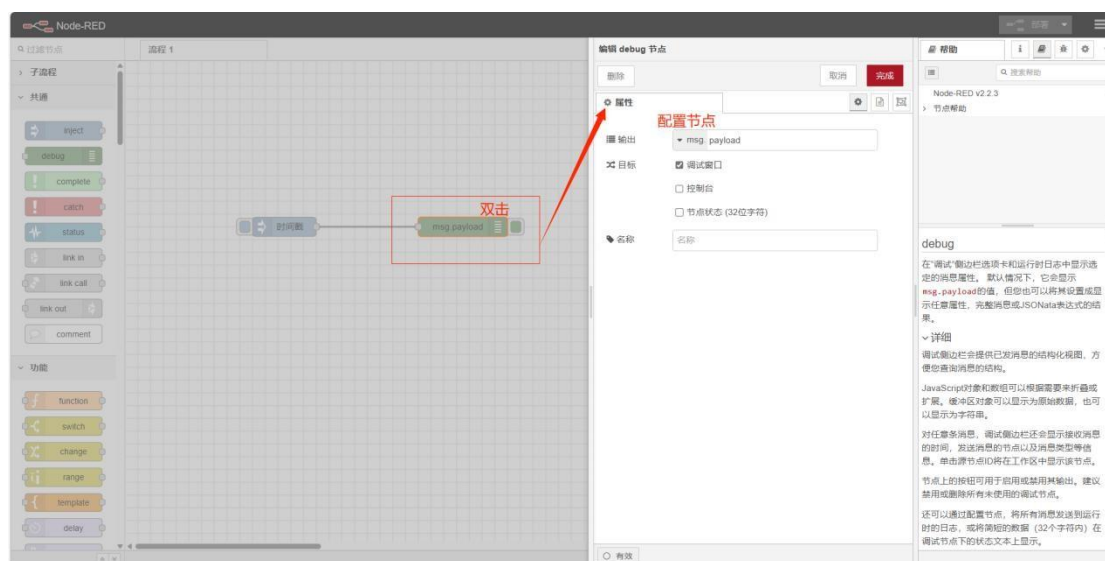
输入和输出：考虑节点的输入和输出要求，确保它们与你的流程相匹配。某些节点可能需要特定类型的消息作为输入，同时生成特定类型的消息作为输出。



3. 配置节点属性

每个节点都有一组配置属性，用于定义其行为和设置。配置节点属性的方法如下：

- (1)、双击节点：在编辑器中双击节点以打开其配置窗口。
- (2)、在节点上右键单击，然后选择 "编辑" 或 "属性"：这也会打开节点的配置窗口。
- (3)、在配置窗口中，你可以设置节点的不同属性，这取决于节点的类型和用途。常见的属性包括输入和输出的主题、消息的转换选项、连接到外部服务的设置等。
- (4)、配置完节点属性后，点击 "完成" 或 "OK" 保存配置



4. 连接和布置节点

连接和布置节点是构建流程的关键步骤，以下是如何连接和布置节点的方法：

连接节点：要连接两个节点，只需将它们之间的鼠标拖动一条导线即可。导线表示数据的流动路径。确保导线连接正确，以便消息可以从一个节点传递到另一个节点。

调整节点位置：你可以在编辑器中拖动节点，以调整它们的位置。布置节点以使流程更清晰和可读。

复制和粘贴节点：你可以选择一个节点，然后使用复制和粘贴操作来创建相似的节点。这对于重复使用相同配置的节点非常有用。

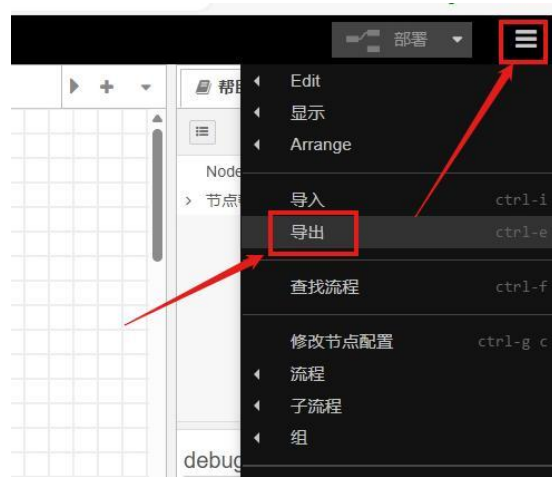
删除节点：如果不再需要节点，可以选择它并按下删除键或右键单击并选择删除选项来删除节点。

编辑导线：你可以双击导线以编辑它们的属性，如修改主题或设置条件。

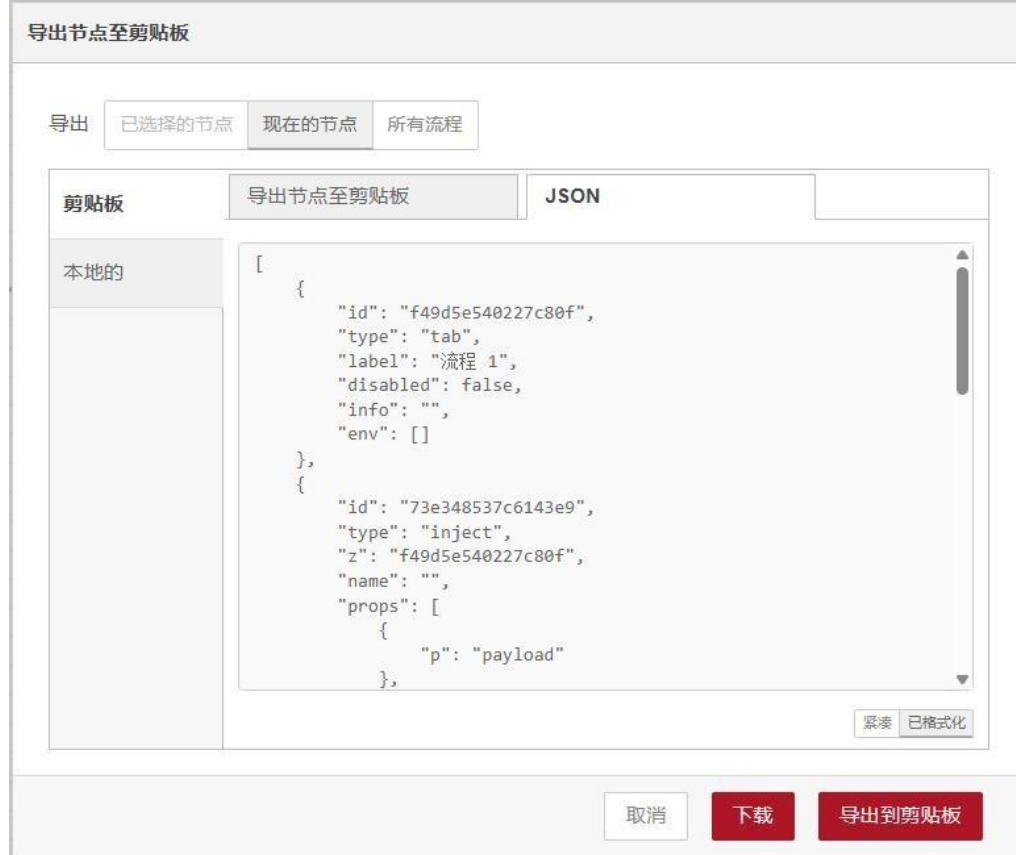
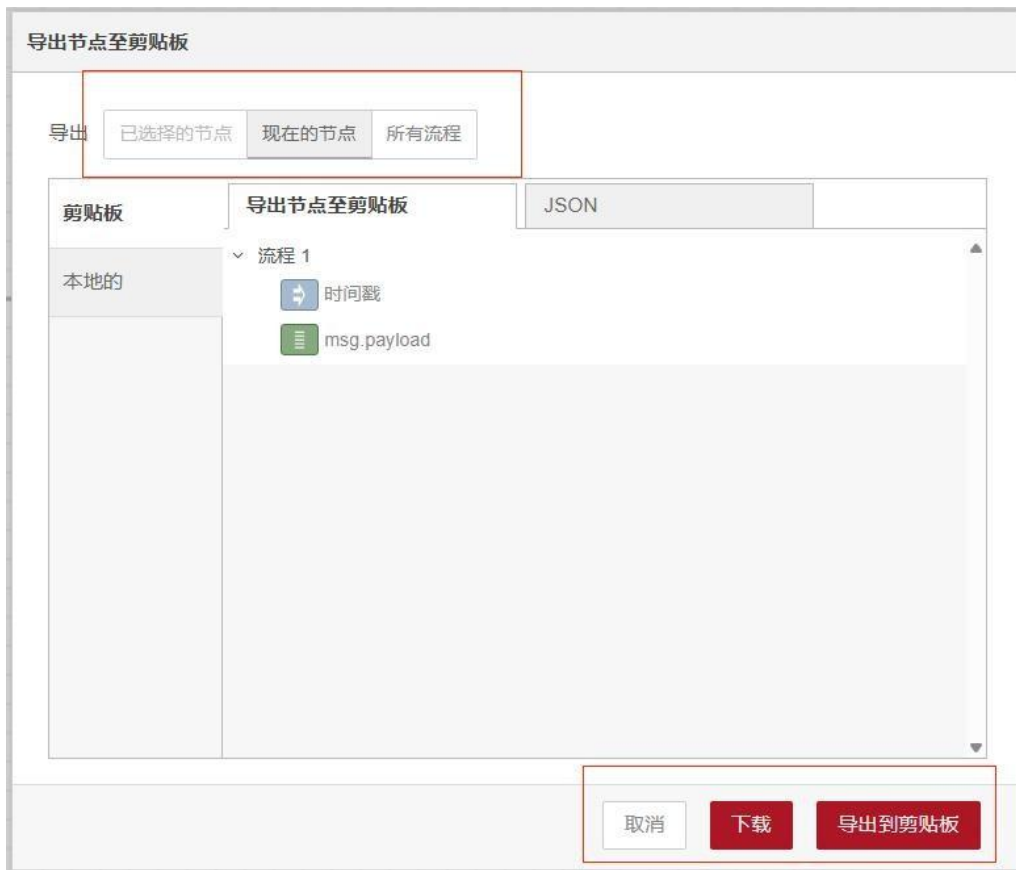
注释节点：Node-RED 允许你添加注释节点，以在流程中添加说明和注释。这有助于文档化你的流程。通过选择合适的节点，配置其属性，并连接它们，你可以创建一个功能强大的 Node-RED 流程，用于执行各种任务和操作。节点操作是使用 Node-RED 进行流程编程的核心部分。

5. 项目和案例导出

(1)、进入导出界面

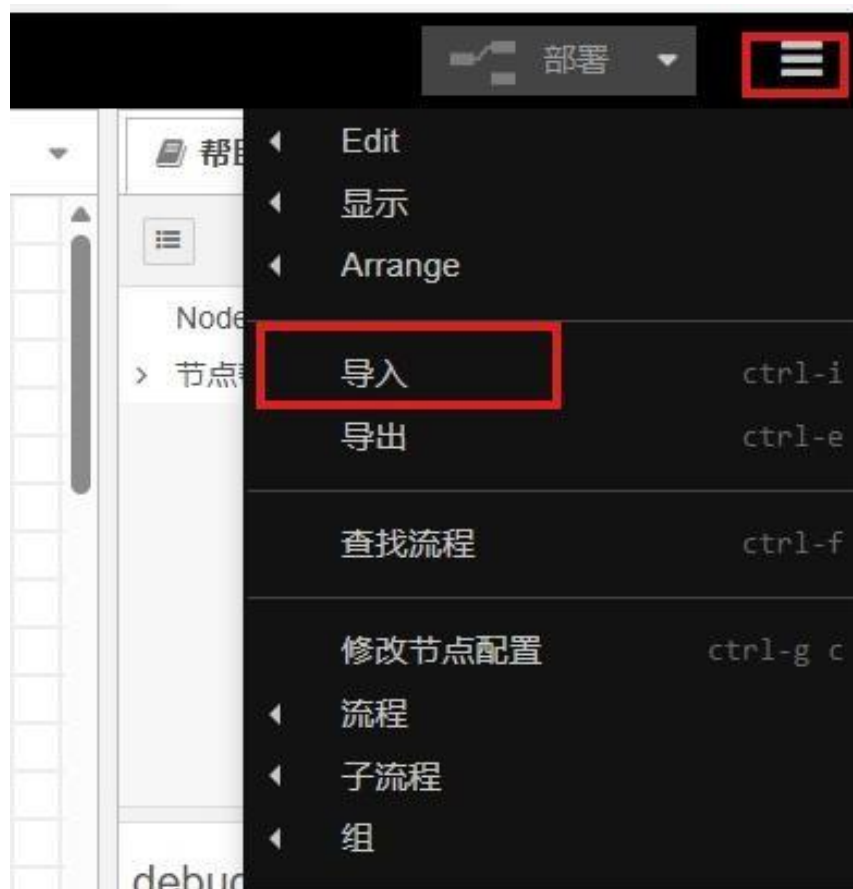


(2)、选择导出的类型，可导出以选择的节点、现在的节点、所有流程，导出的格式可选 下载（下载 json 文件）、导出到剪贴板（复制 json 格式文本到剪切板）。

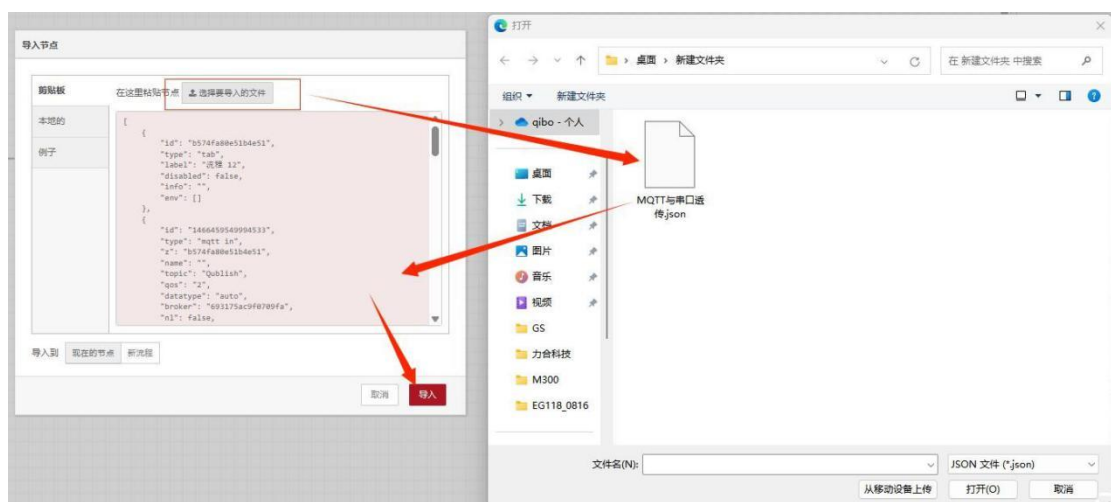


6. 项目和案例导入

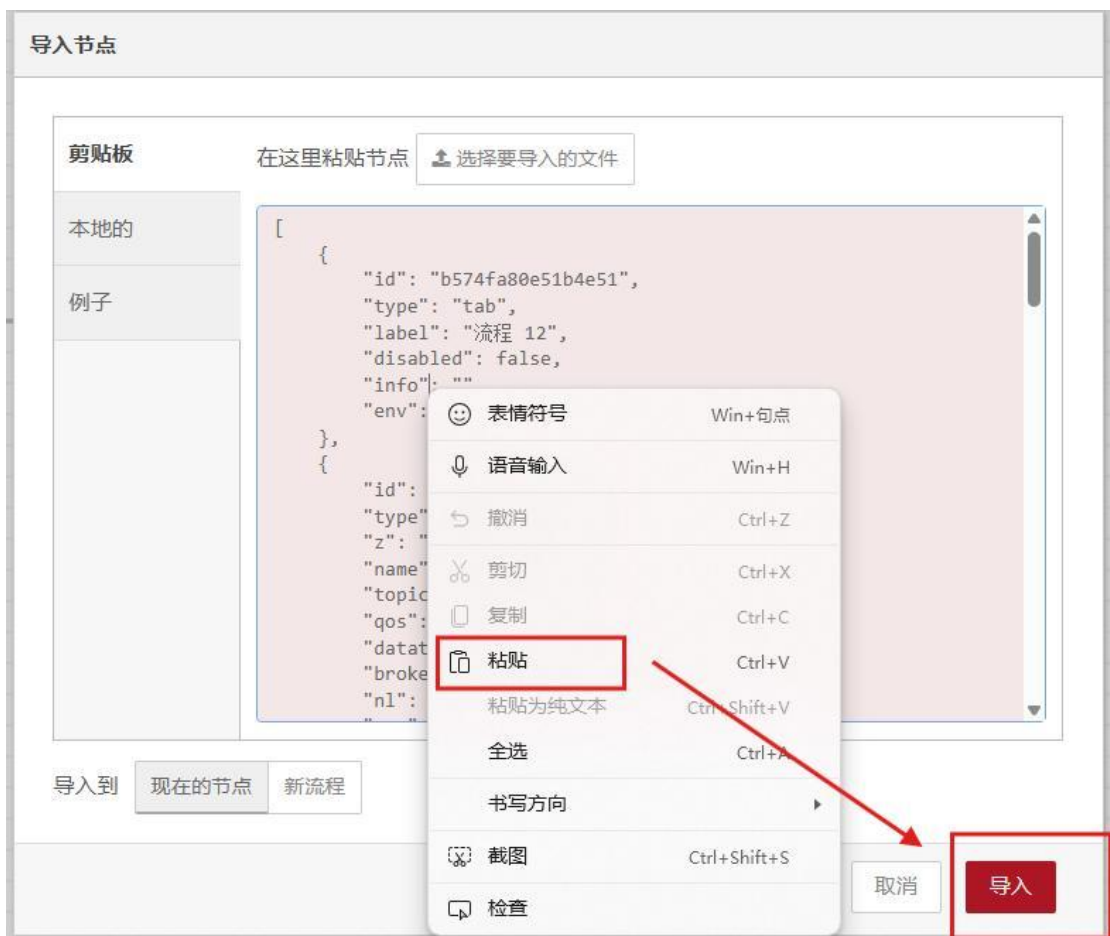
(1)、进入导入界面



(2)、方式一：导入文件



(3)、方式二：粘贴 json



7. 项目部署运行

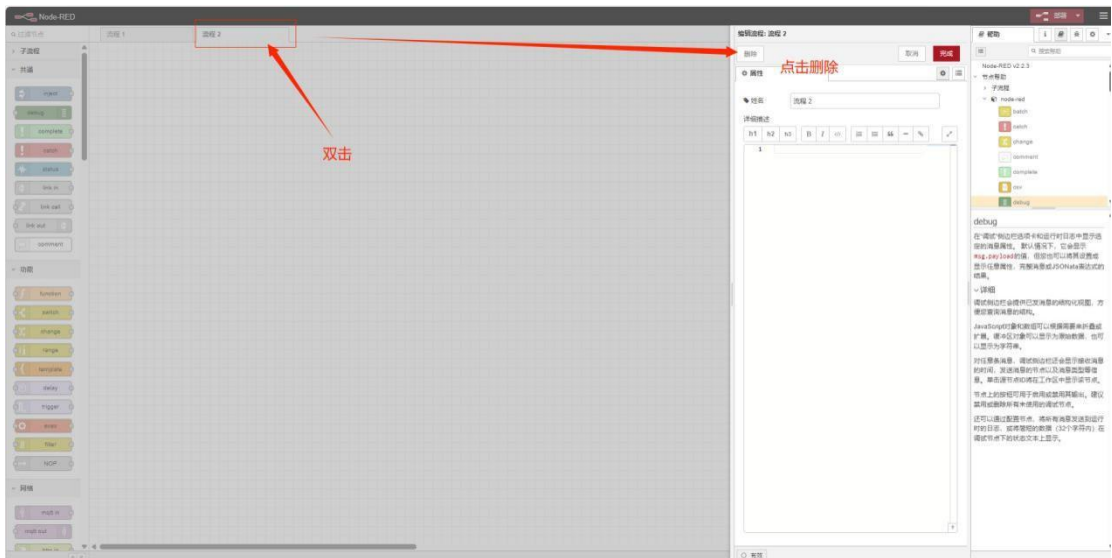


8. 流程的添加和删除

(1)、添加流程

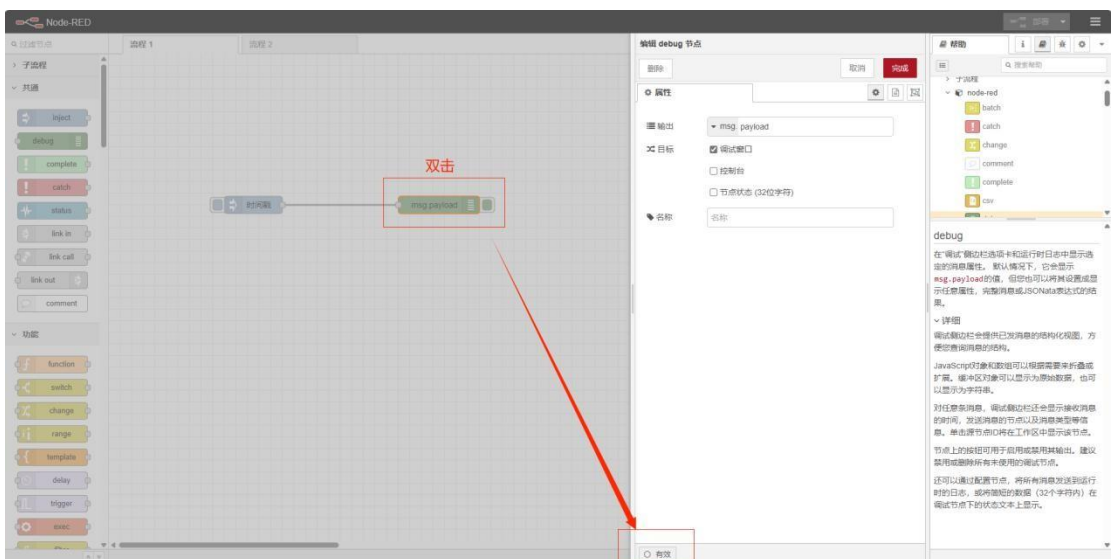


(2)、删除流程



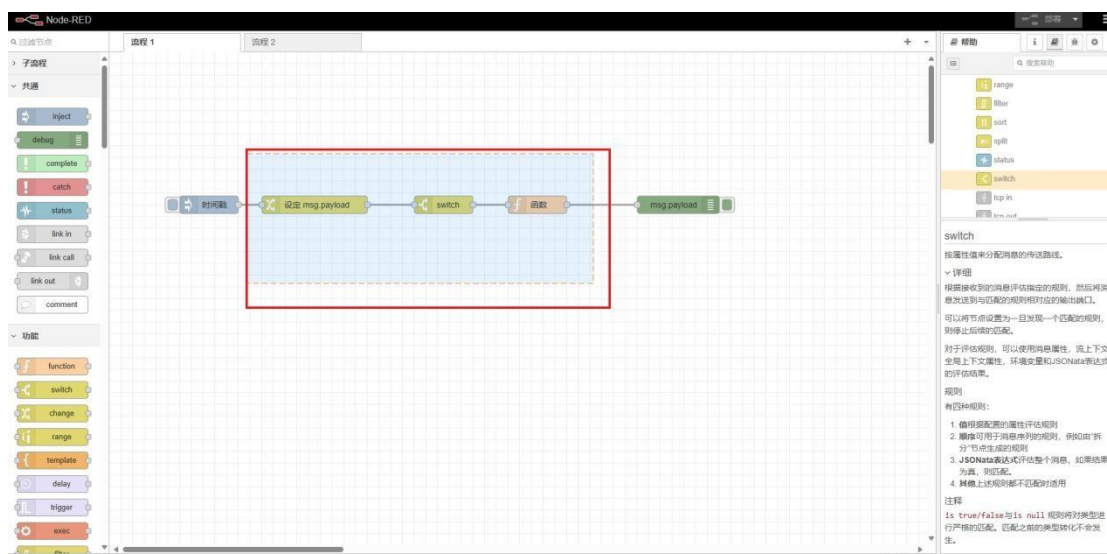
9. 控制节点有效无效

双击节点，点击左下角的有效或者无效。

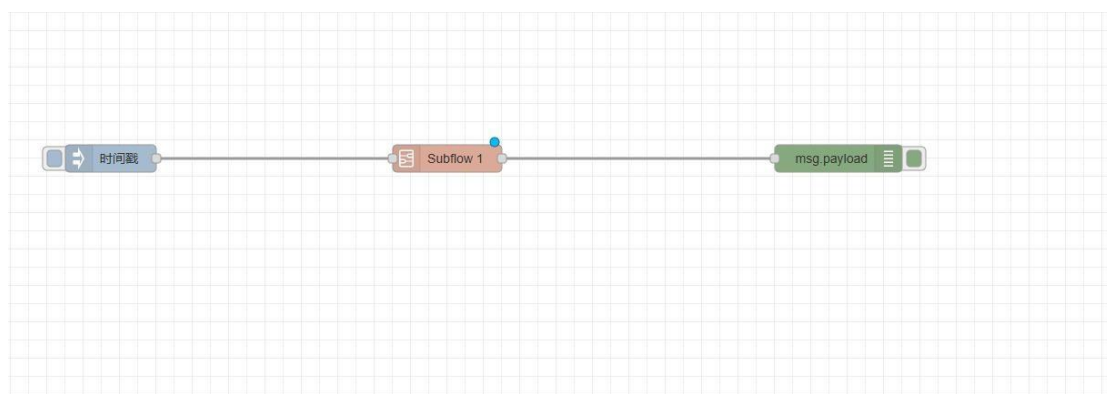
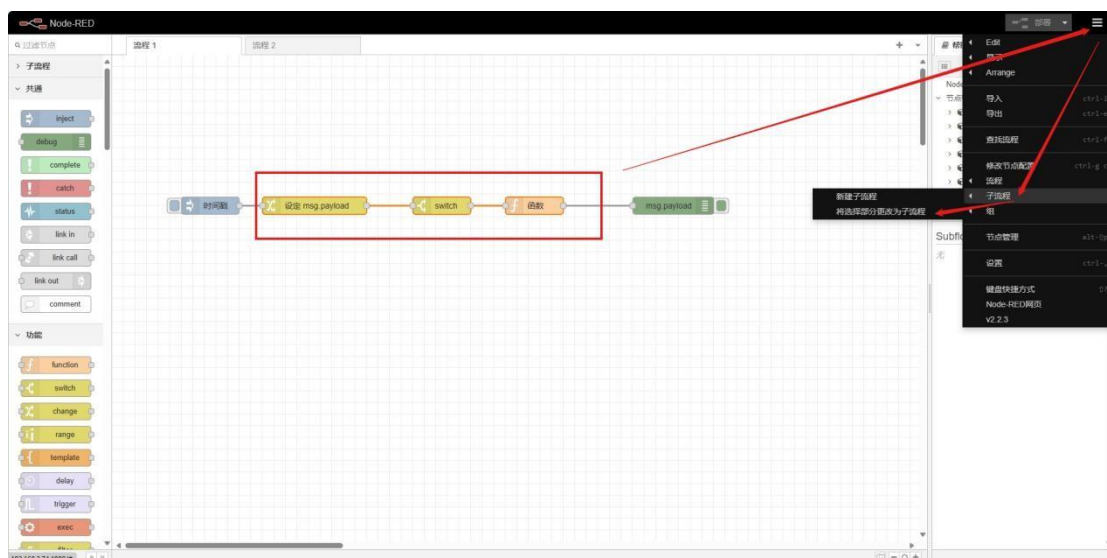


10. 子流程的建立

(1)、选择需要在一下子流程的节点，inject 节点是无法放在子流程里面的。

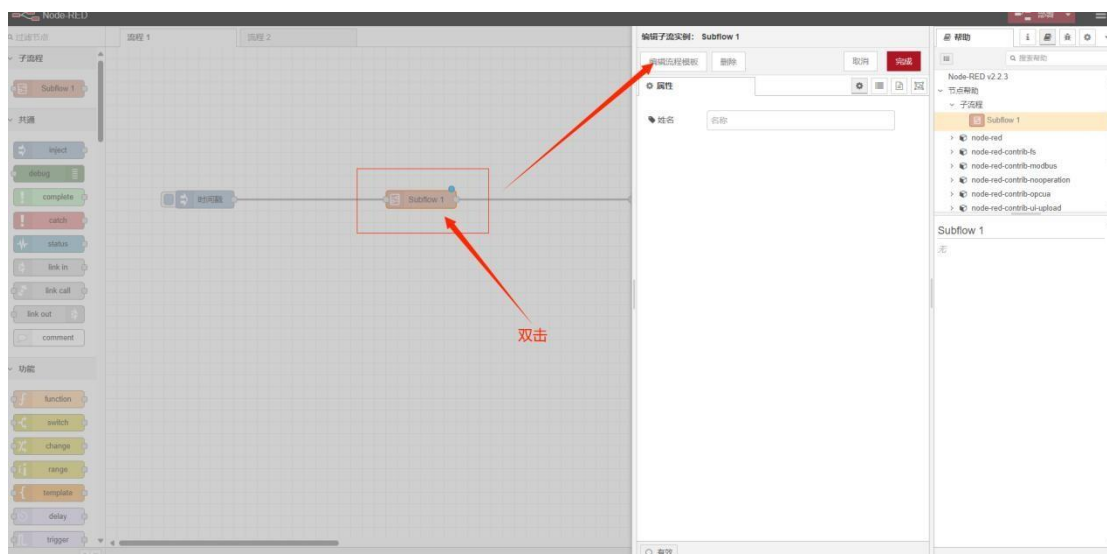


(2)、将选择的部分改为子流程。

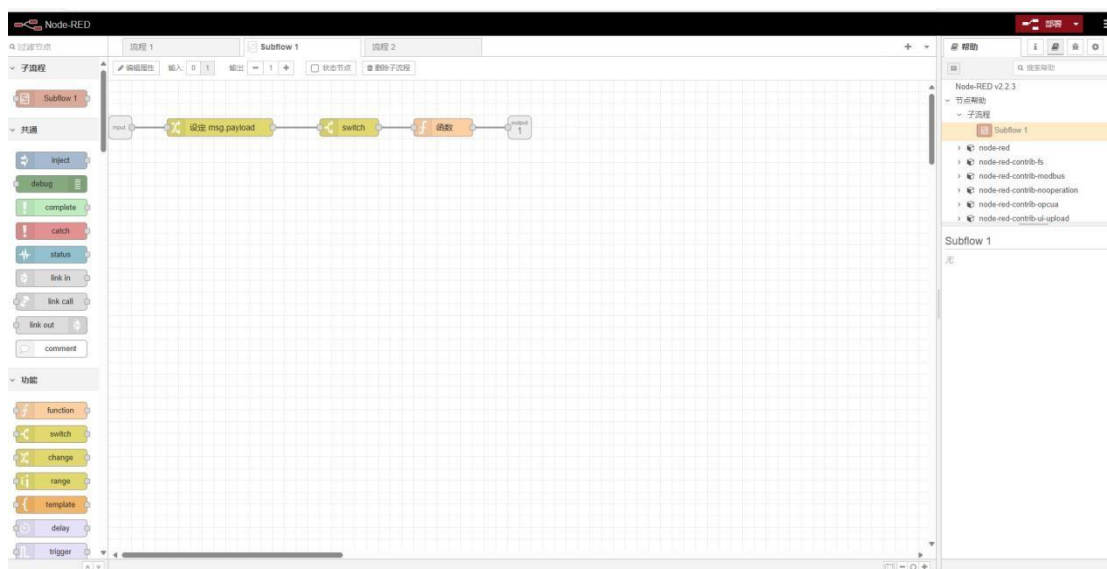


(3)、进入子流程。

方法一：



方法二：

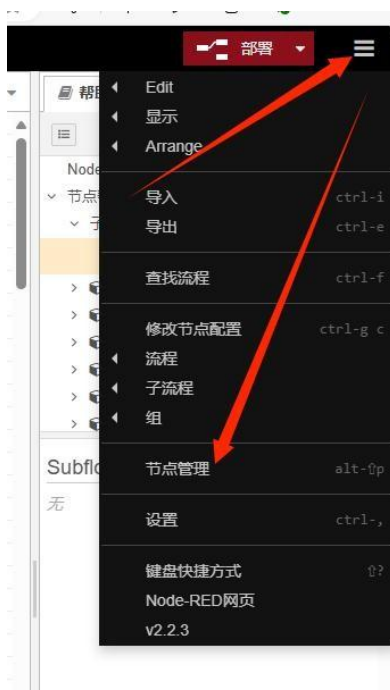


(4)、修改子流程的配置

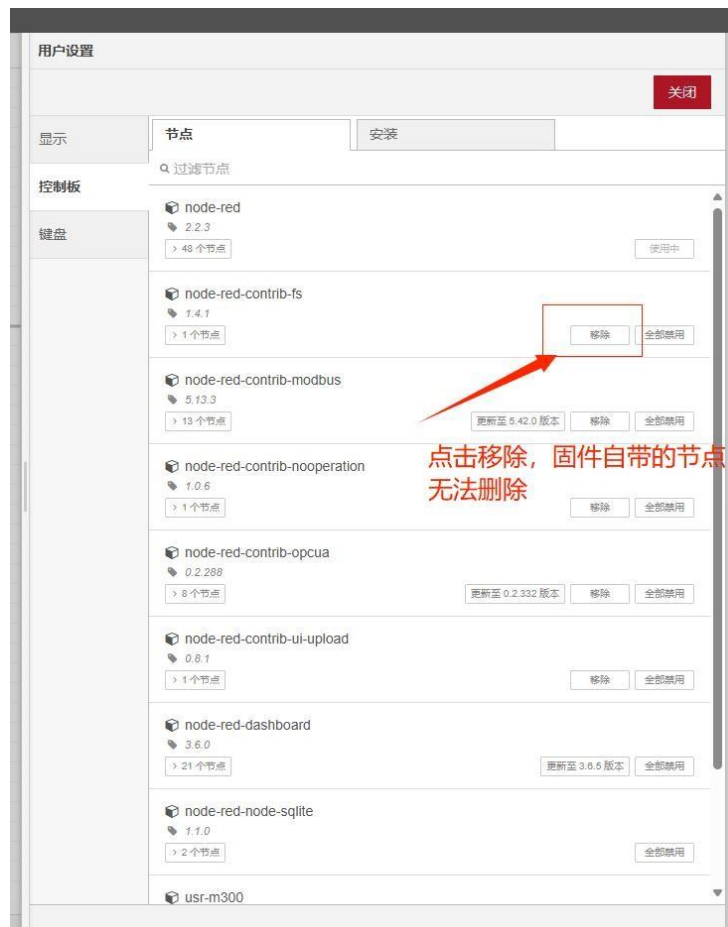


11. 卸载安装节点

(1)、打开界面

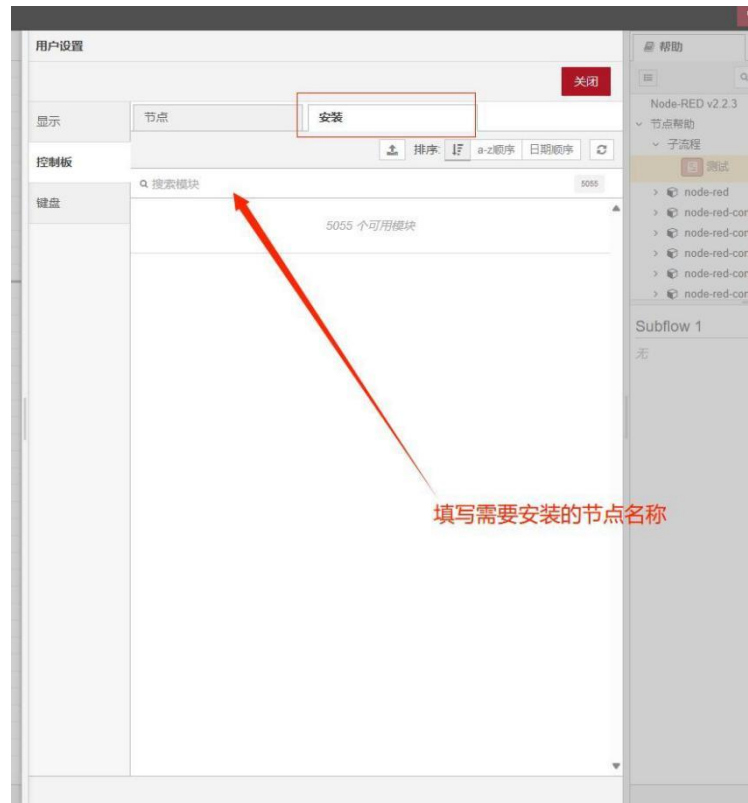


(2)、节点删除

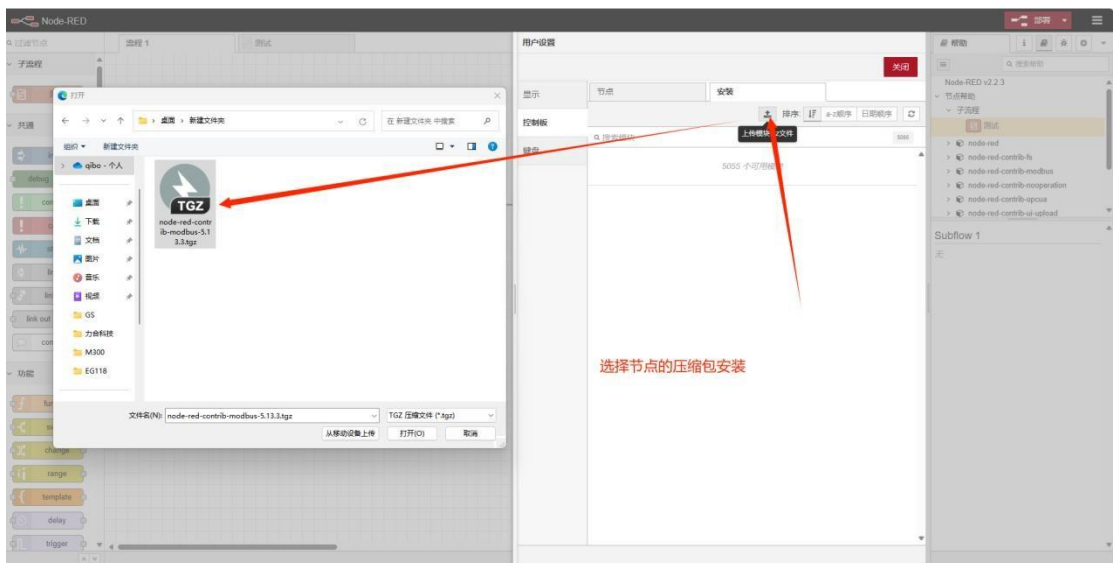


(3) 节点安装

方法一：在线安装

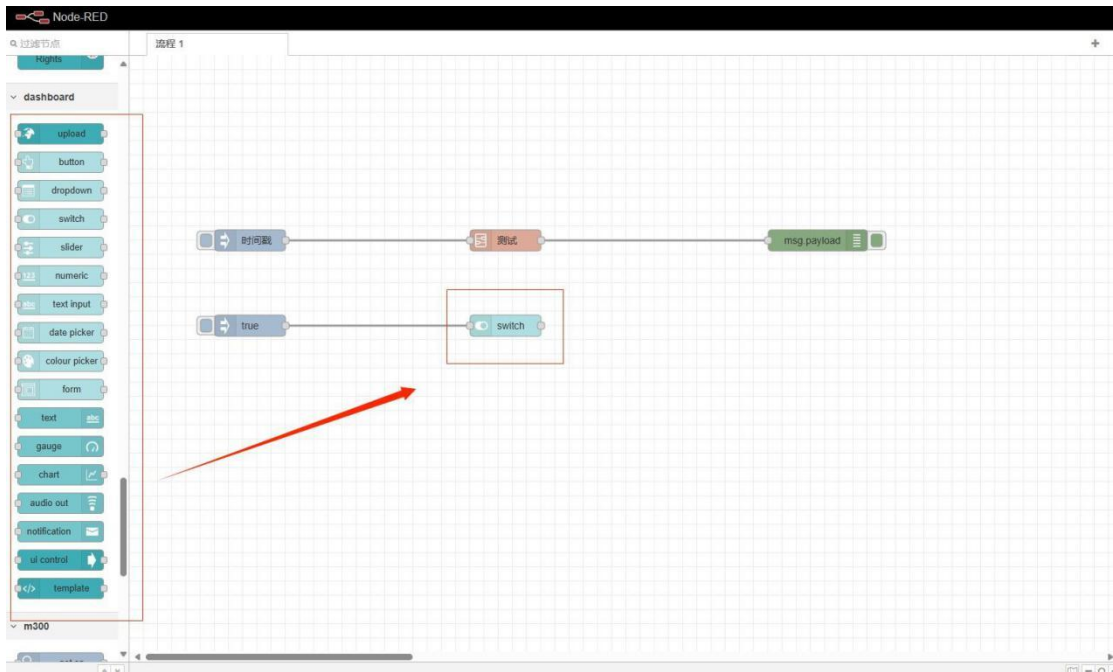


方法二：本地安装

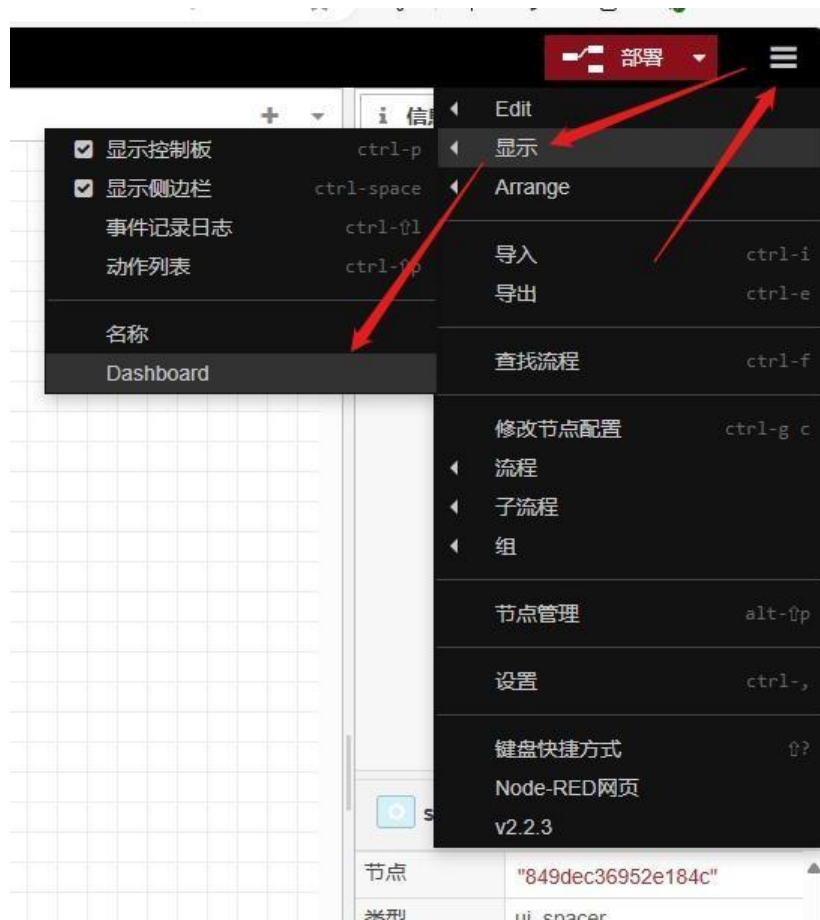


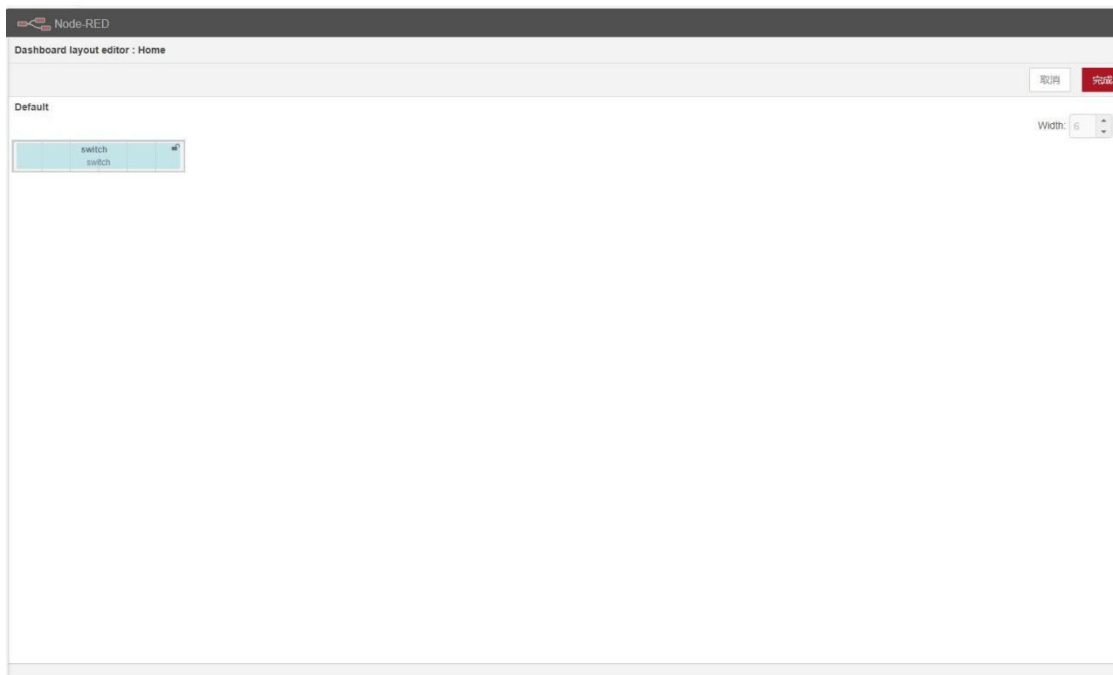
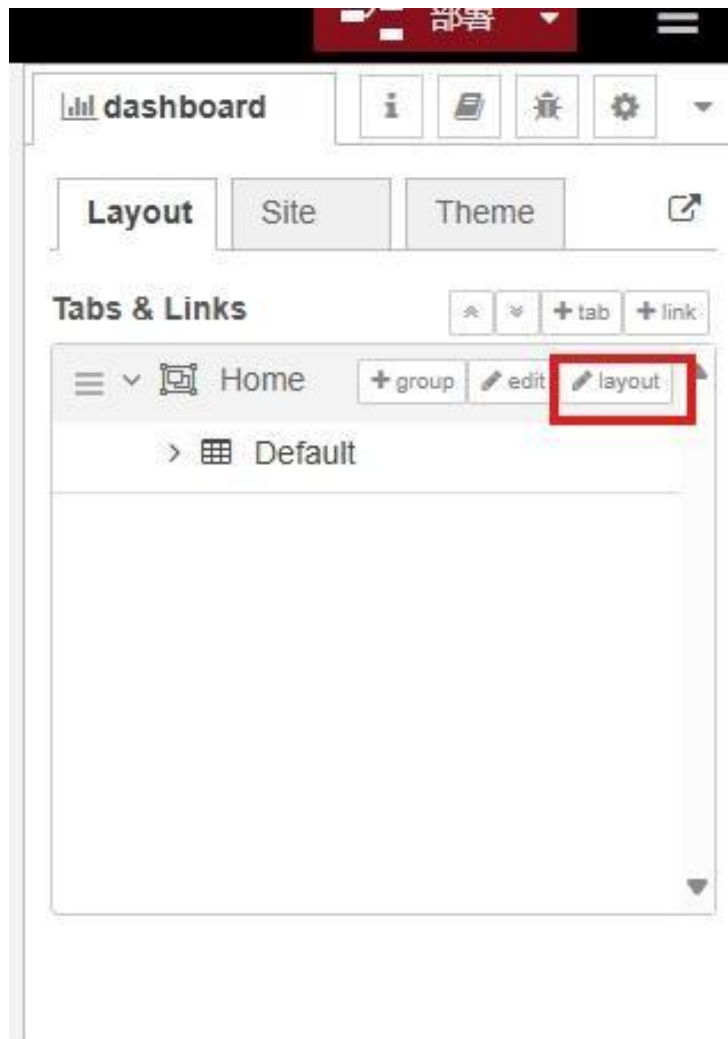
12. 进入 UI 界面

(1)、首先创建 UI 界面的节点

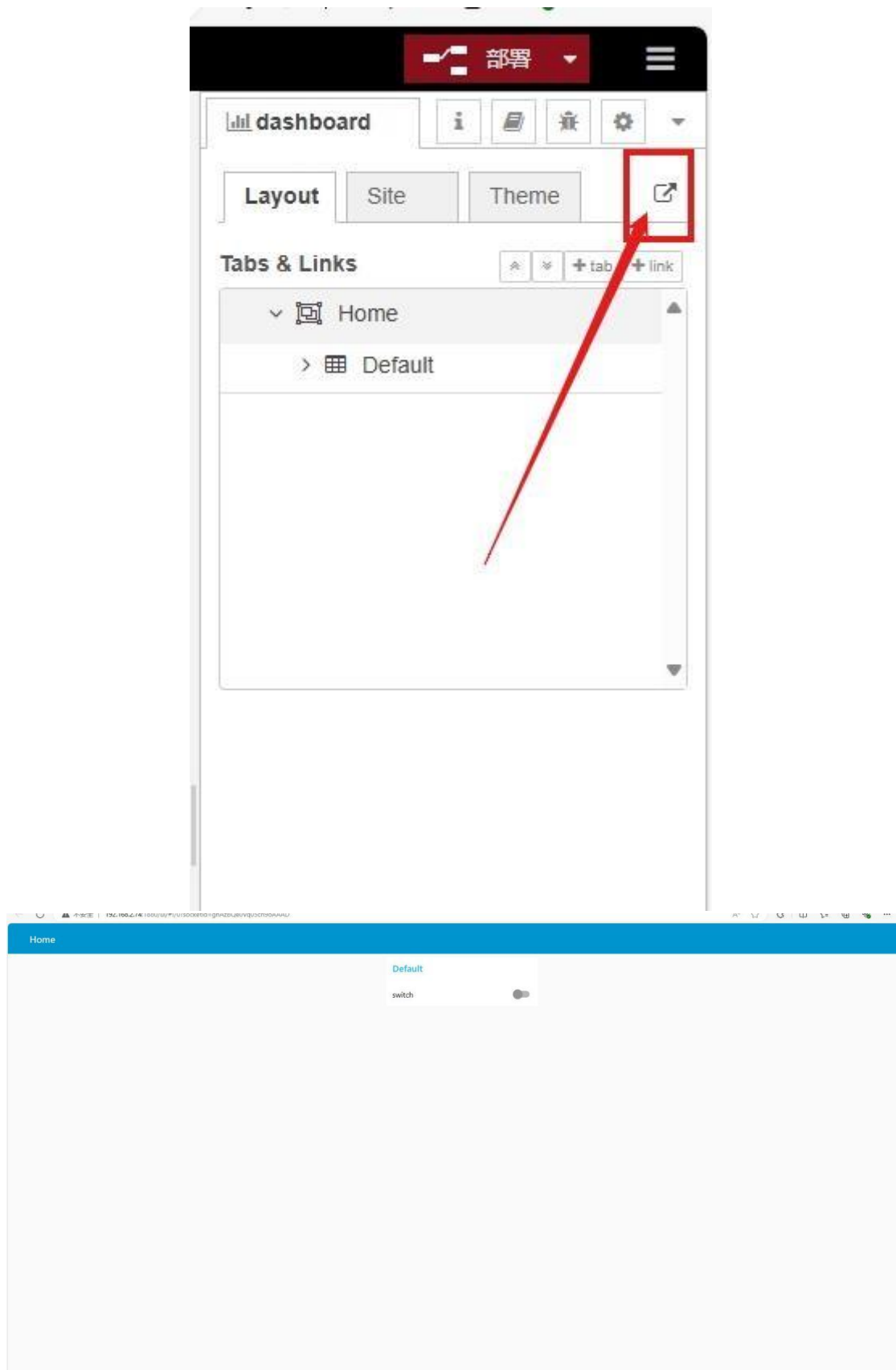


(2)、修改图标的大小





(3)、进入 UI 界面



13. Node-RED 使用示例

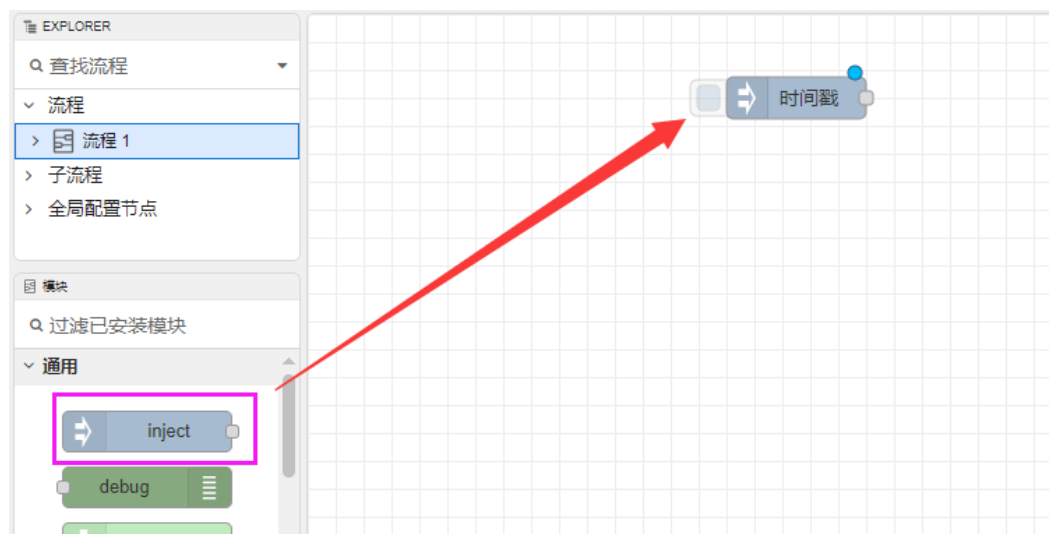
13.1.CAN 总线收发

13.1.1. 前置配置

配置并启动 CAN 接口。



➤ 添加 inject 节点用于触发 can0 参数配置；



双节时间戳节点，属性选择：

编辑 inject 节点

删除

取消

完成

属性

名称

名称

msg. payload

=

⌵ ⌚ milliseconds since epoch

×

msg. topic

=

⌵ a_z

×

+ 添加

立即注入

☒ 立刻执行于

1

秒后, 此后

⌲ 重复

无

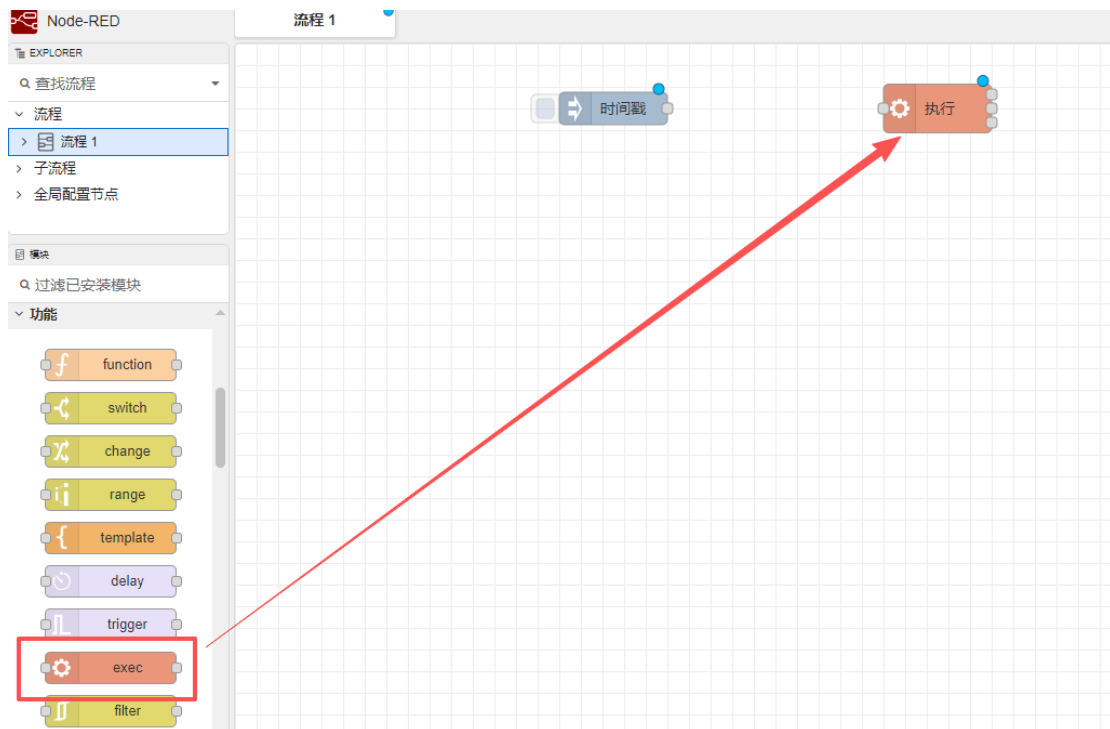
⌵



☐ 有效

注：此处勾选立即于 1 秒后，部署后重新启动设备会随着启动 node-red 自动执行，如需重复执行可再下面重复里选择具体间隔时间进行重复执行。

➤ 添加 exec 节点，设置 can0 设备波特率为 250000 并开启 can0。（配置如图）



双击 exec 节点，添加属性：

```
sudo ip link set can0 type can bitrate 250000 && sudo ifconfig can0 up
```

编辑 exec 节点

删除取消完成

⚙️ 属性

📄 命令

pe can bitrate 250000 && sudo ifconfig can0 up

➕ 追加

☐ msg. payload

💡 输出

额外的输入参数

⌚ 超时

当命令完成时 - exec模式

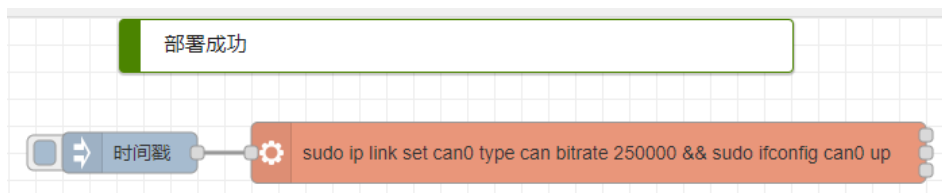
🖥️ 隐藏控制台

☐

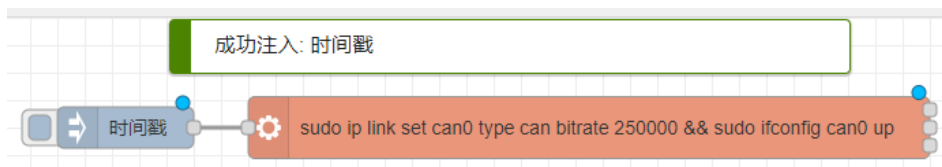
🏷️ 名称

名称

配置完成后连接 inject 与 exec 节点并部署运行：



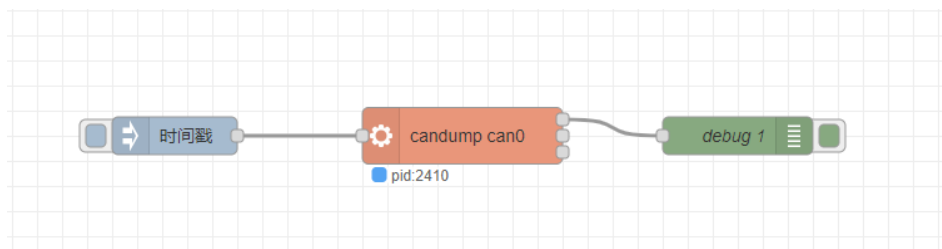
点击 inject 节点按钮（首次运行点击 inject）：



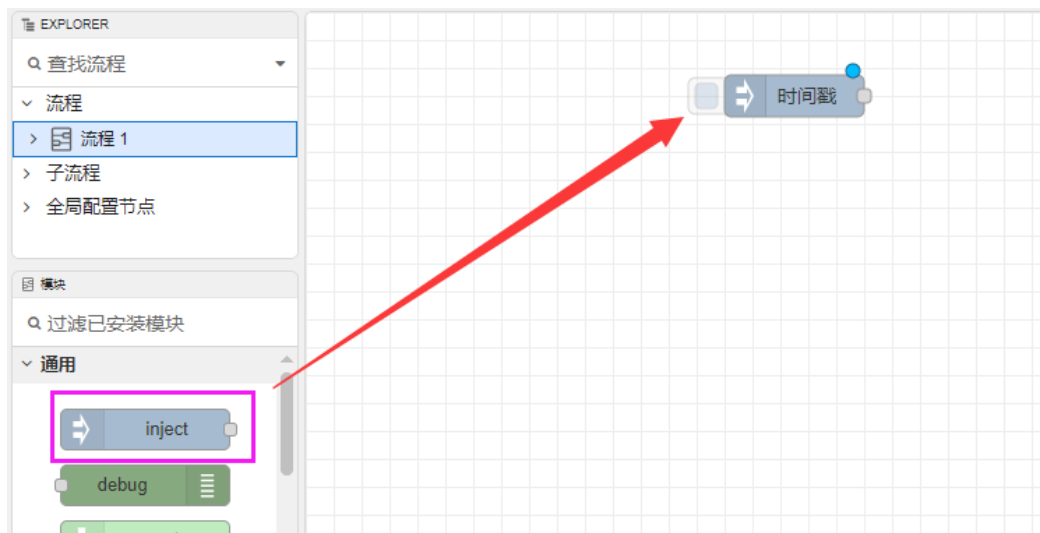
提示成功注入表示 can0 参数设置成功并已经开启。

13.1.2. 使用 Exec 节点收发 CAN 数据

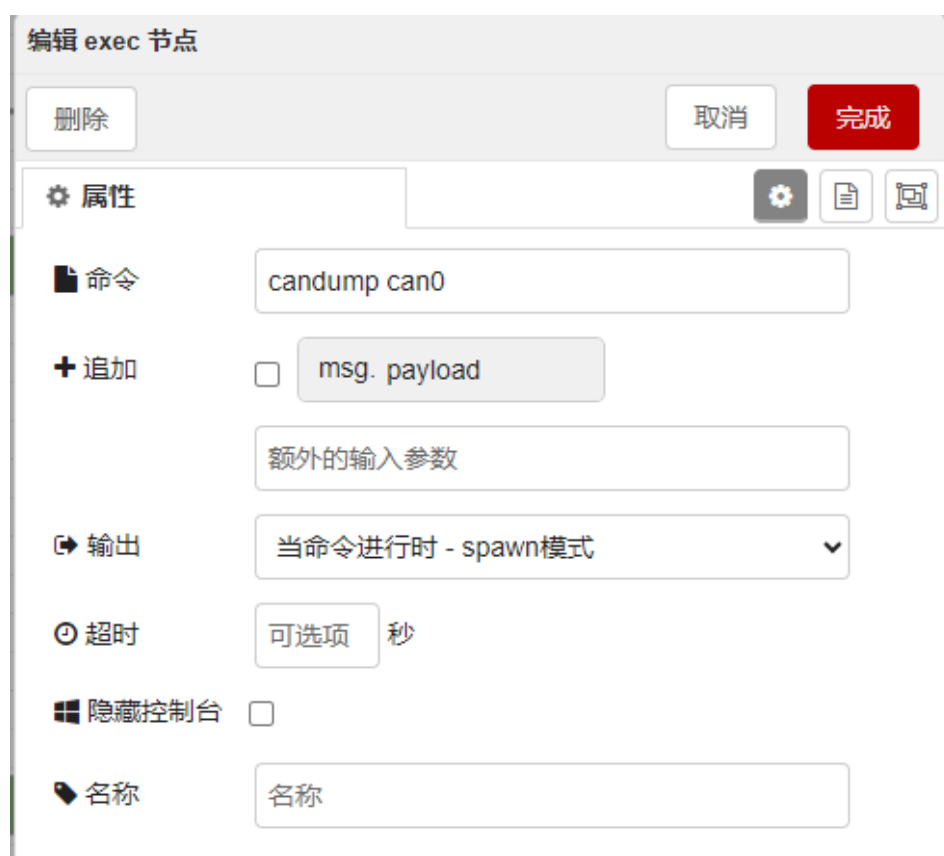
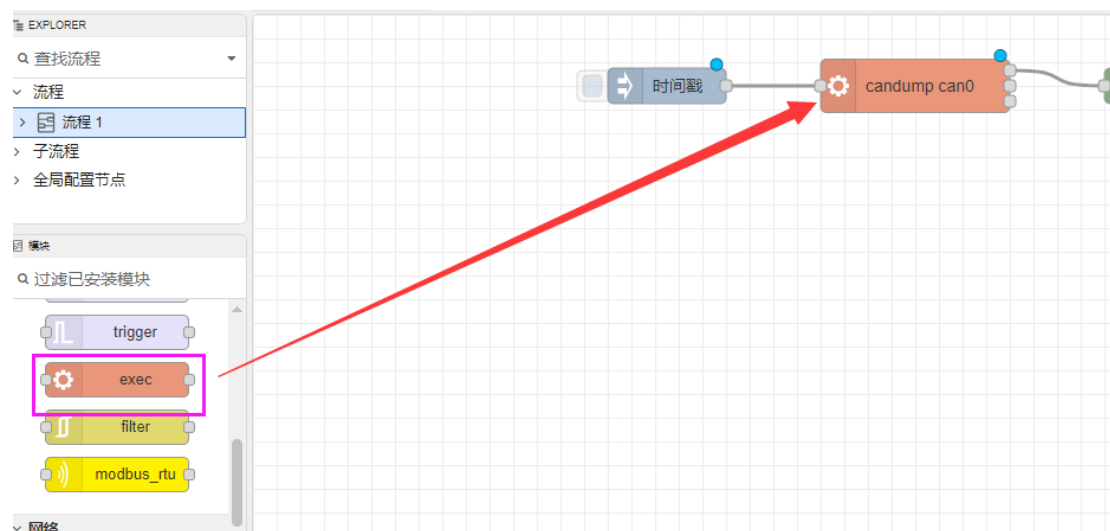
(1)、接收 CAN 数据 (candump)



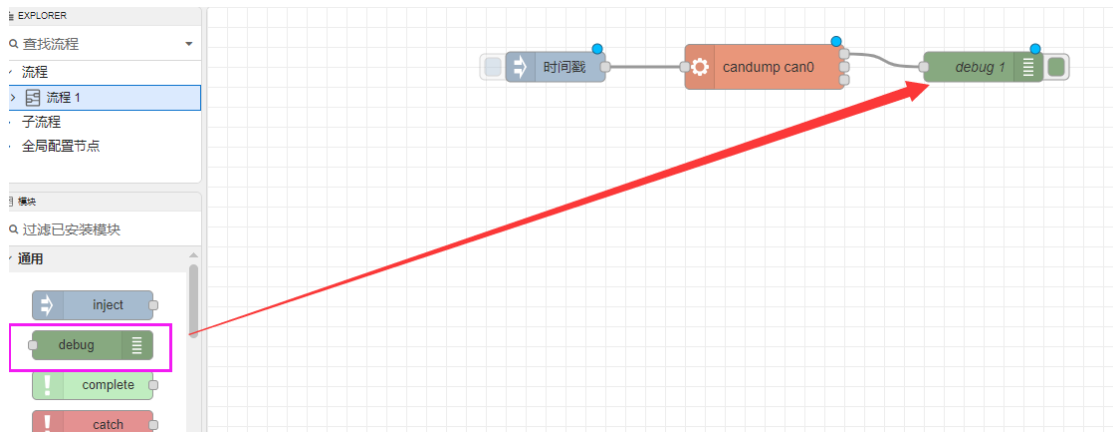
➤ 添加 inject 节点用于触发开启 can 接收；



➤ 添加 exec 节点并设置命令 `candump can0`，属性配置中输出改为“当命令行时 - spawn 模式”；(如图)



- 添加 debug 节点，输出接收到的 can 数据；

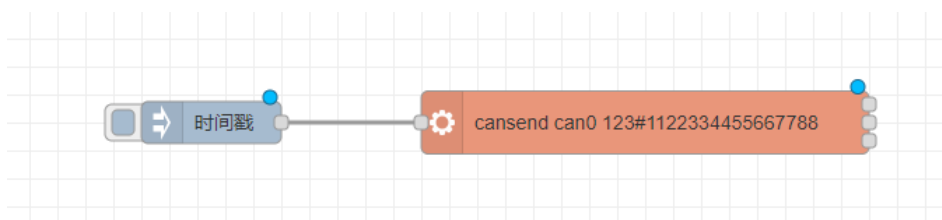


设置完成后点击 inject 按钮开启接收。

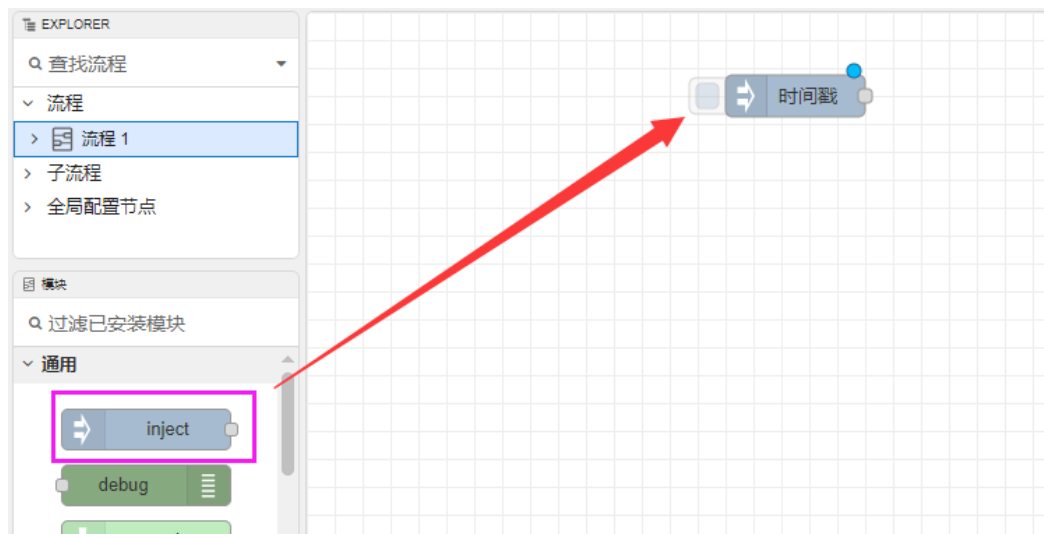
接收结果如下图：



(2)、发送 CAN 数据 (cansend)



- 添加 inject 节点用于点击触发 can 发送；



- 添加 exec 节点并设置命令 `cansend can0 123#1122334455667788`，用于发送固定 can 命令。（可设置指定间隔周期性执行，如图）

编辑 inject 节点

删除

取消

完成

属性

名称

名称

msg. payload

=

⌵

🕒

milliseconds since epoch

⌵

✕

msg. topic

=

⌵

a_z

⌵

✕

+ 添加

立即注入

☒ 立刻执行于

1

秒后, 此后

重复

周期性执行

⌵

每隔

10

⌵

秒

⌵

www.cosofteck.com

27

编辑 exec 节点

删除 取消 完成

属性

命令

+ 追加 ☐ msg. payload

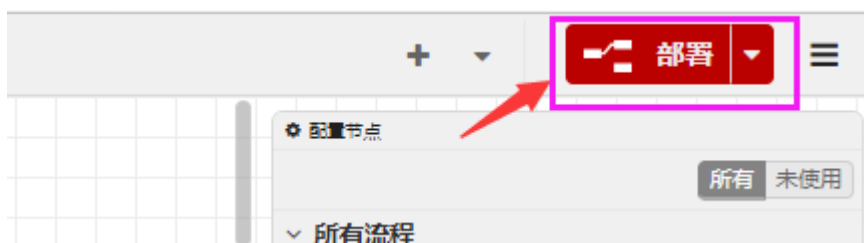
输出

超时 秒

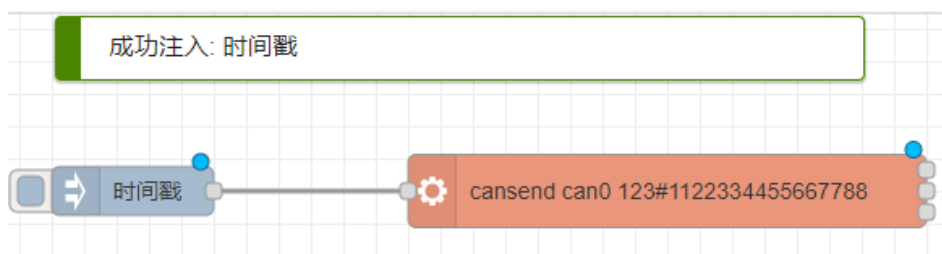
隐藏控制台 ☐

名称

- 配置完成后点击部署：



- 部署完成进行发送：



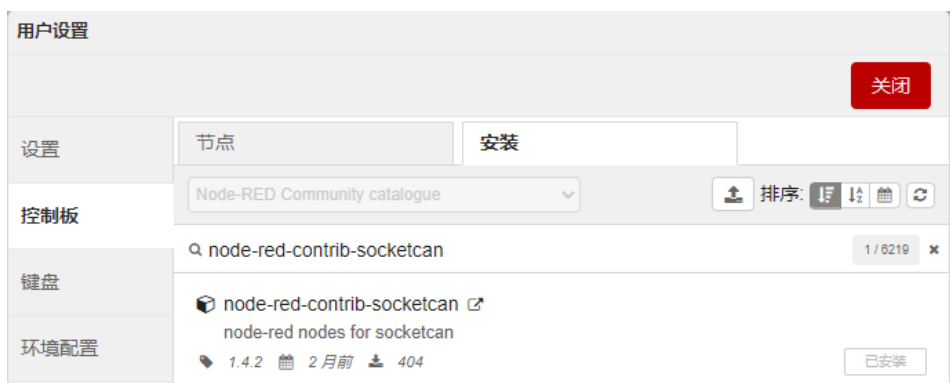
接收端：

帧ID	帧格式	帧类型	数据长度	数据(HEX)
0x00000123	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
0x00000123	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
0x00000123	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
0x00000123	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
0x00000123	数据帧	标准帧	0x08	11 22 33 44 55 66 77 88

13.1.3. 使用 node-red-contrib-socketcan 收发 CAN 数据

(1)、节点安装

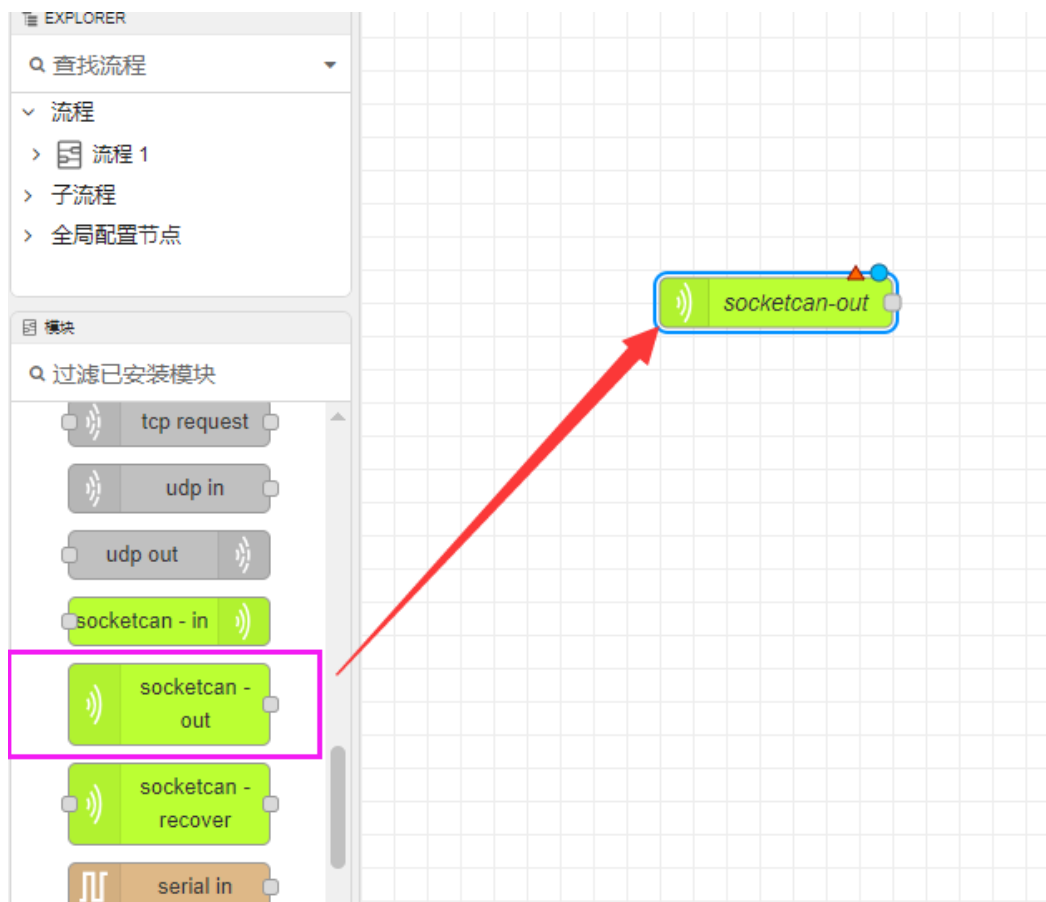
添加 node-red-contrib-socketcan 支持：找到 Node-RED 右上角“更多 -- 节点管理 -- 控制面板 -- 安装”的搜索栏，搜索 node-red-contrib-socketcan 点击安装。（如下图）



(2)、接收 CAN 数据 (socketcan-out)



- 添加 socketcan-out 节点，并设置好对应 can 设备号（注：此处名为 out 实际作用为获取设备接收到的 can 数据）；



- 双击 socketcan-out 节点配置属性（如无已配置好的属性点击+号添加，以 can0 为例）：

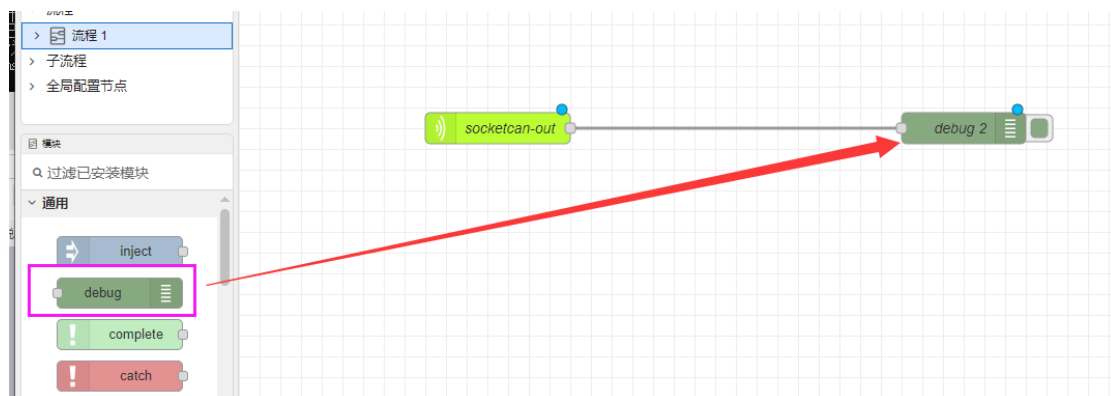




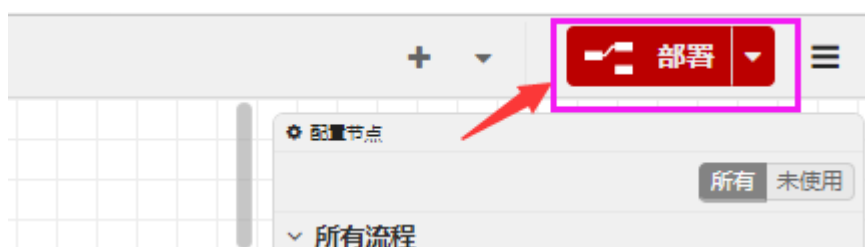
- 添加完成后返回上级页面选中添加后的 can 设备：



- 添加 debug 节点用于输出显示接收到的数据；



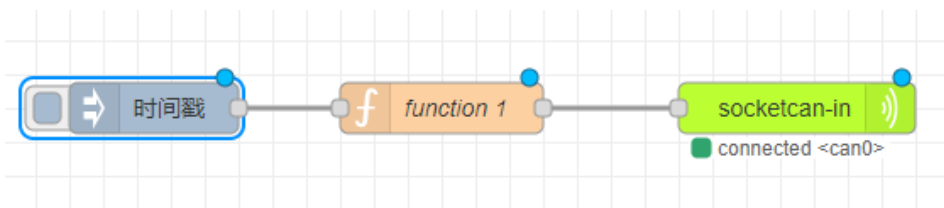
- 点击部署后生效；



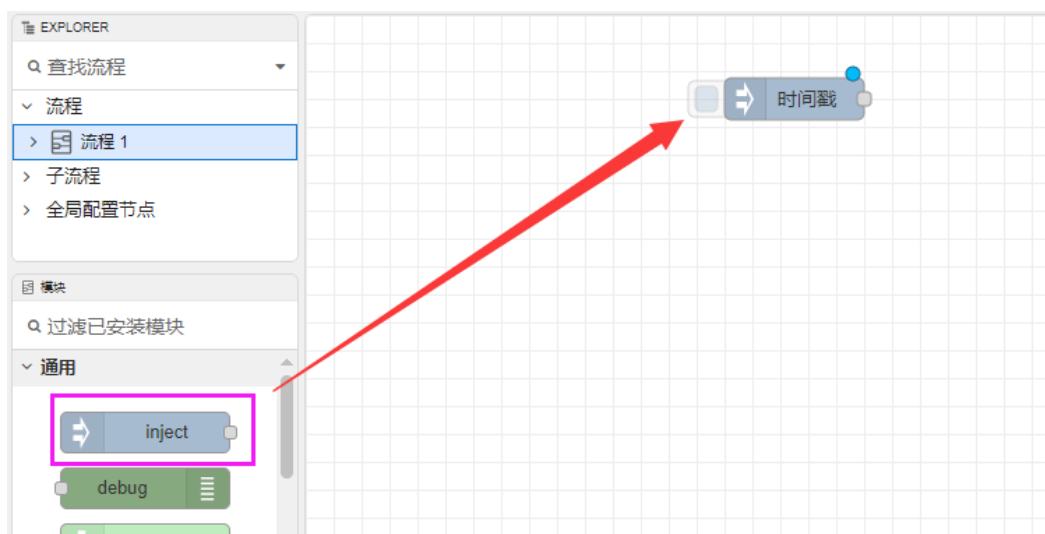
接收到的数据会在右侧调试窗口中显示，如下图：



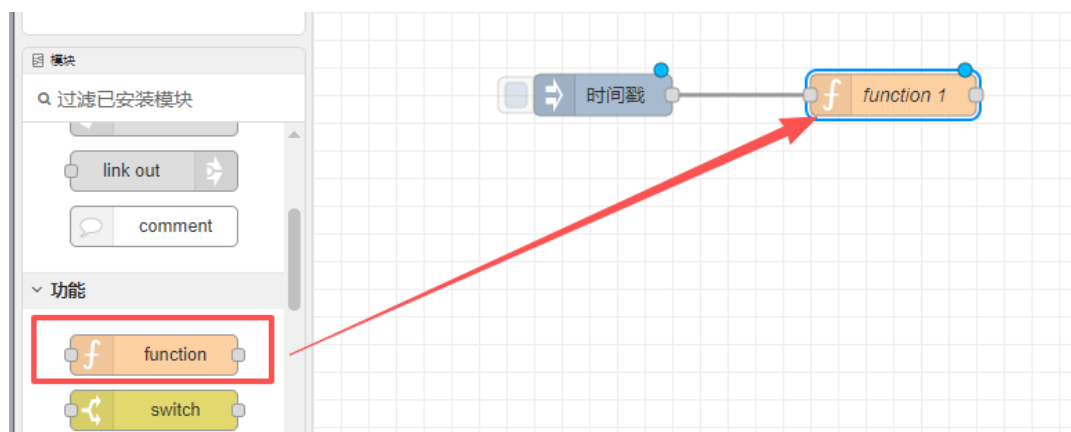
(3)、发送 CAN 数据 (socketcan-in)



- 添加 inject 节点用于点击触发 can 发送；



- 添加 function 节点，用于设置发送内容；(如图)



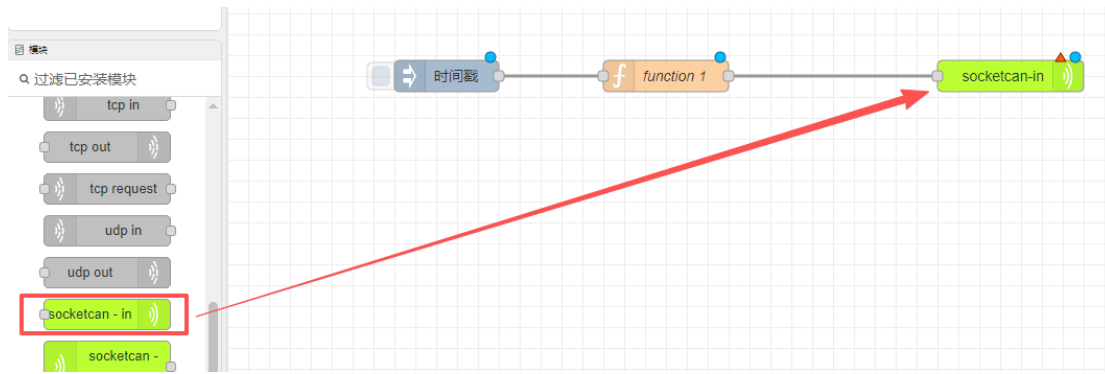
- 双击 function 节点添加需要发送的内容：



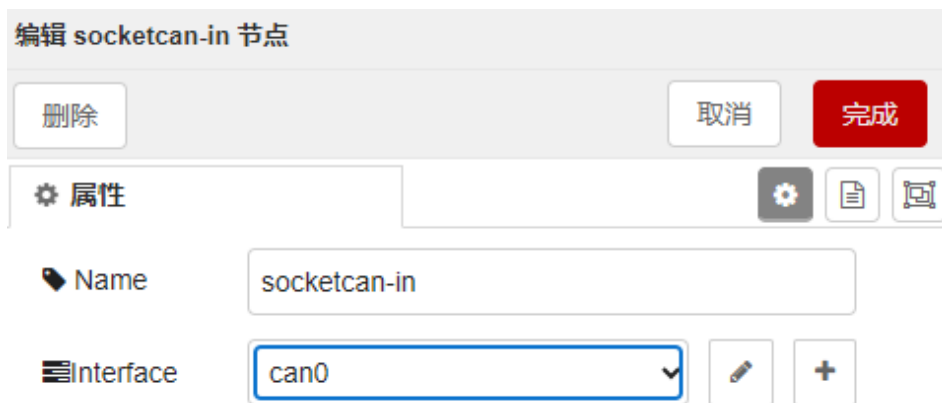
- 添加运行函数，内容如下：

```
msg.payload = {
  id: 0x123,
  ext: false,
  data: [
    0x11,0x22,0x33,0x44,
    0x55,0x66,0x77,0x88
  ]
};
return msg;
```

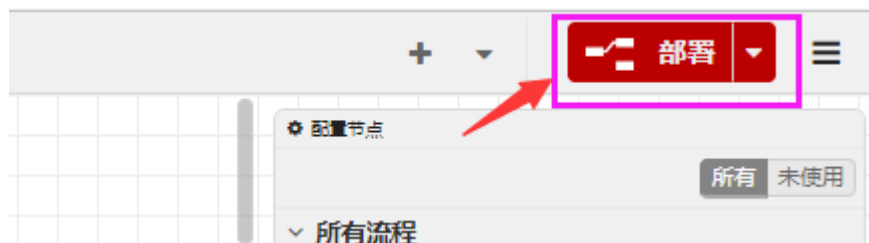
- 添加 socketcan-in 节点，并设置关联 can0 设备；



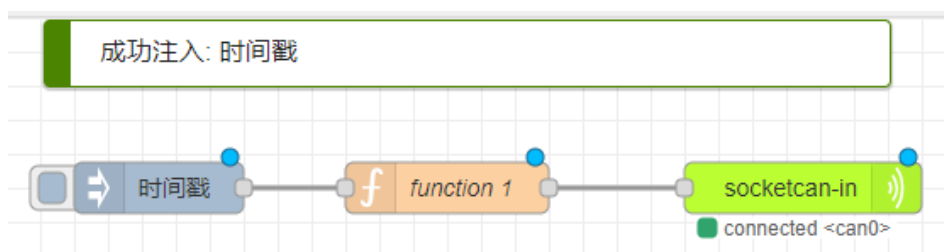
- 双击 socketcan-in 节点，添加属性：



- 点击部署后生效；



- 点击发送:



- 接收端：

帧格式	帧类型	数据长度	数据(HEX)
数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
数据帧	标准帧	0x08	11 22 33 44 55 66 77 88
数据帧	标准帧	0x08	11 22 33 44 55 66 77 88

13.2.modbus 转 mqtt

13.2.1. 前置配置

安装 modbus 节点。

- 安装对应节点：

打开“更多 --- 节点管理”

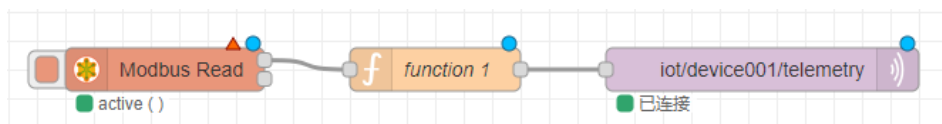


- 选择安装，输入 node-red-contrib-modbus 并安装；

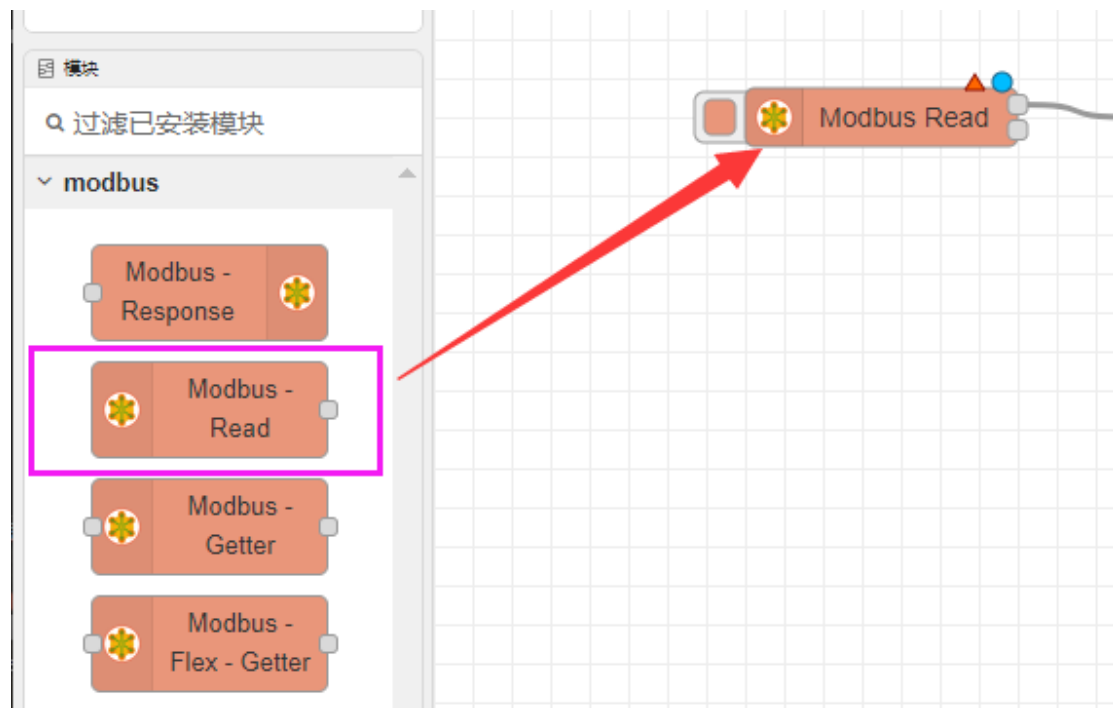


安装完成后检查 Node-RED 左侧工作台，能查到有 modbus 相关节点即表示已安装成功。

13.2.2. modbus 数据转发 mqtt



- 添加 modbus Read 节点用于读取 modbus 数据；(双击 modbus Read 节点进行配置)；



- 双击 modbus-read 节点进行配置：

编辑 Modbus-Read 节点

删除 取消 完成

属性

Settings Optional

名称 Name

主题 Topic

Unit-Id 1

FC FC 3: Read Holding Registers

Address 0

Quantity 10

Poll Rate 10 second(s)

Delay to activate input ☐

Allow oversized messages ☐

Server 串口1

1 从机地址

2 寄存器类型为保持寄存器

3 起始地址为0查询10个寄存器地址

4 10s轮询一次

5 指定串口，如无定义号的串口点击+号添加

➤ 串口 1 配置：

编辑 Modbus-Read 节点 > 编辑 modbus-client 节点

删除 取消 更新

属性

Settings Queues Optionals

名称 串口1

Type Serial 1 模式为串口，使用modbus tcp协议时也可以选为tcp

Serial port /dev/ttyS9

Serial type RTU 2 串口号及相关参数

Baud rate 9600

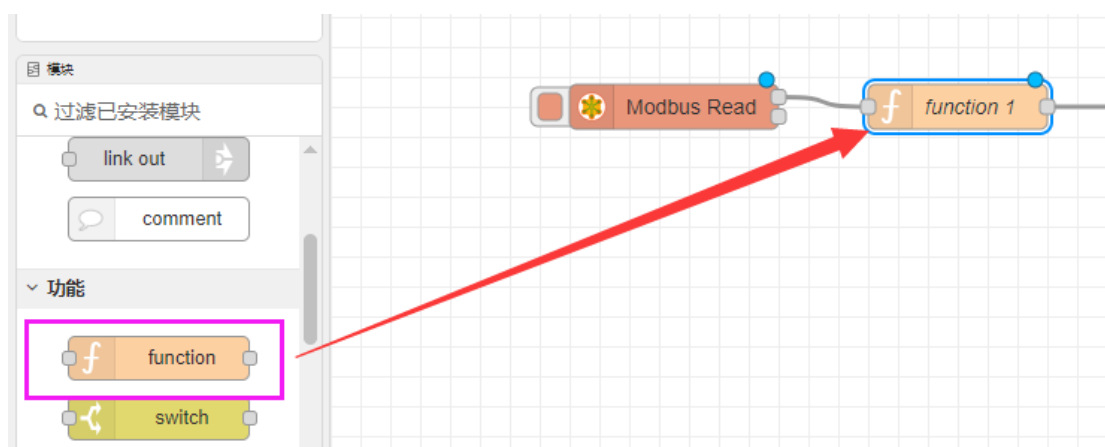
Unit-Id 1

Timeout (ms) 1000

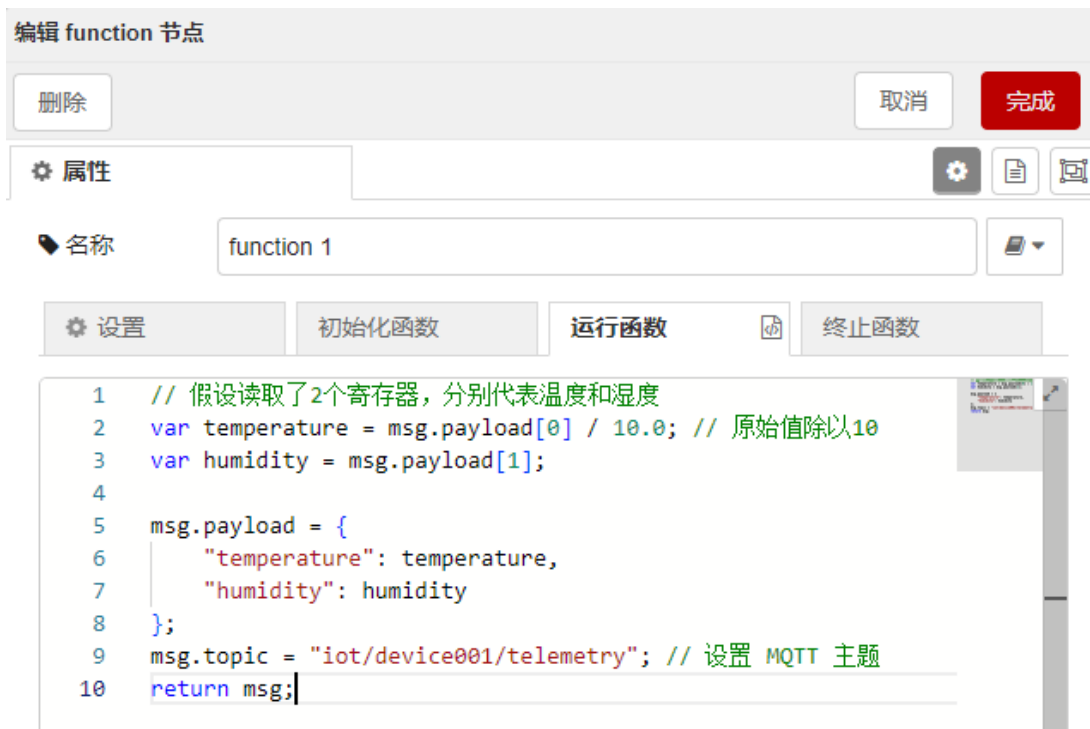
Reconnect on timeout ☒

Reconnect timeout (ms) 2000

- 添加 function 节点，将读取到的 modbus 数据转为 json 格式便于通过 mqtt 发送；（此处以采集到的前两个寄存器数值为例，如图：）



- 双击 function 节点添加内容：



- 添加 mqttout 节点，输出 modbus 采集到的数据；(节点配置如图：)



编辑 mqtt out 节点 > 编辑 mqtt-broker 节点

删除 取消 更新

属性

名称 modbusTest

连接参数 安全 消息

服务端 192.168.1.200 端口 1883

☒ 自动连接
☐ 启用 TLS

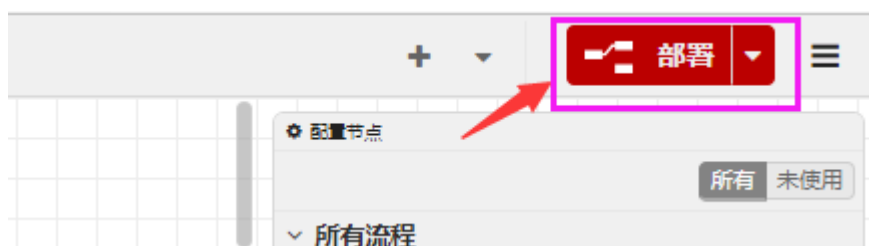
协议 MQTT V3.1.1

客户端ID modbus-rtu-gateway-01

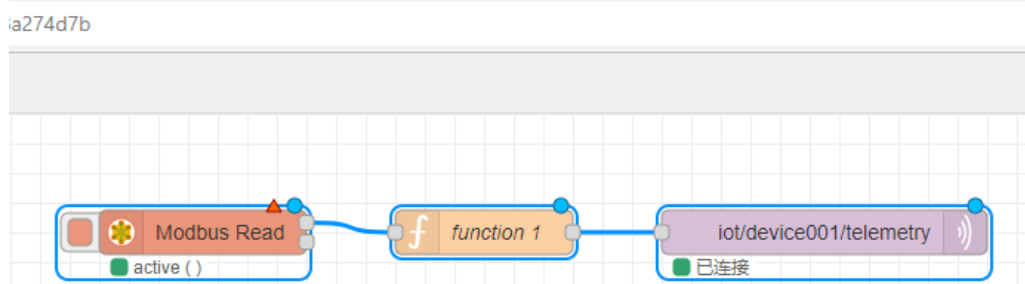
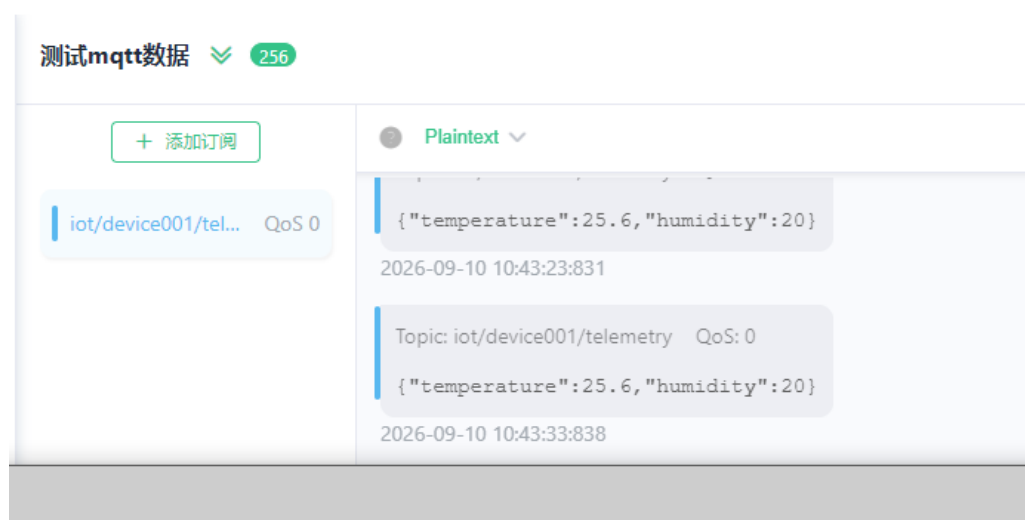
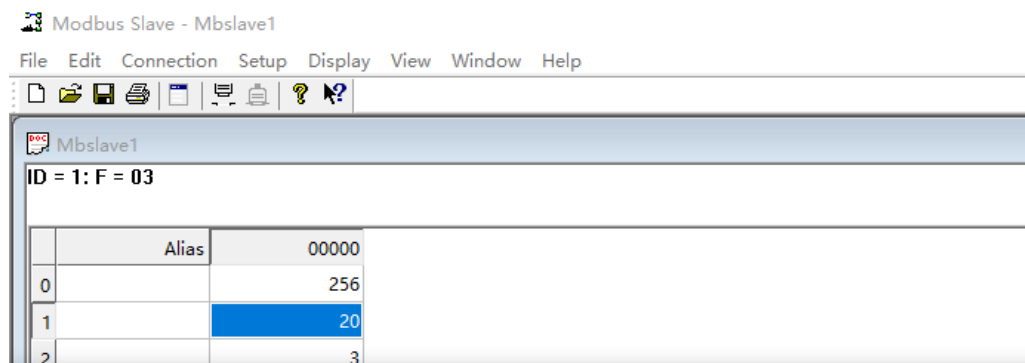
KeepAlive计时(秒) 60

会话 ☒ 使用新的会话

- 配置完成后点击部署生效：



部署完成后点击 modbus Read 左侧按钮，开始采集 modbus 数据并转发到 mqtt。效果如下图：



13.3.modbus RTU 转 modbus TCP

13.3.1. 前置配置

安装 modbus 节点。

➤ 安装对应节点：

打开“更多 --- 节点管理”

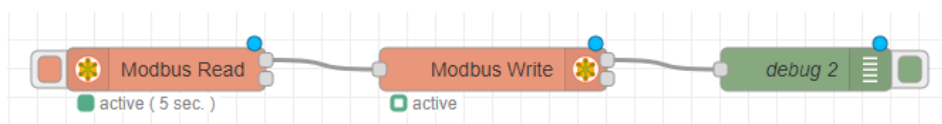


- 选择安装，输入 node-red-contrib-modbus 并安装；

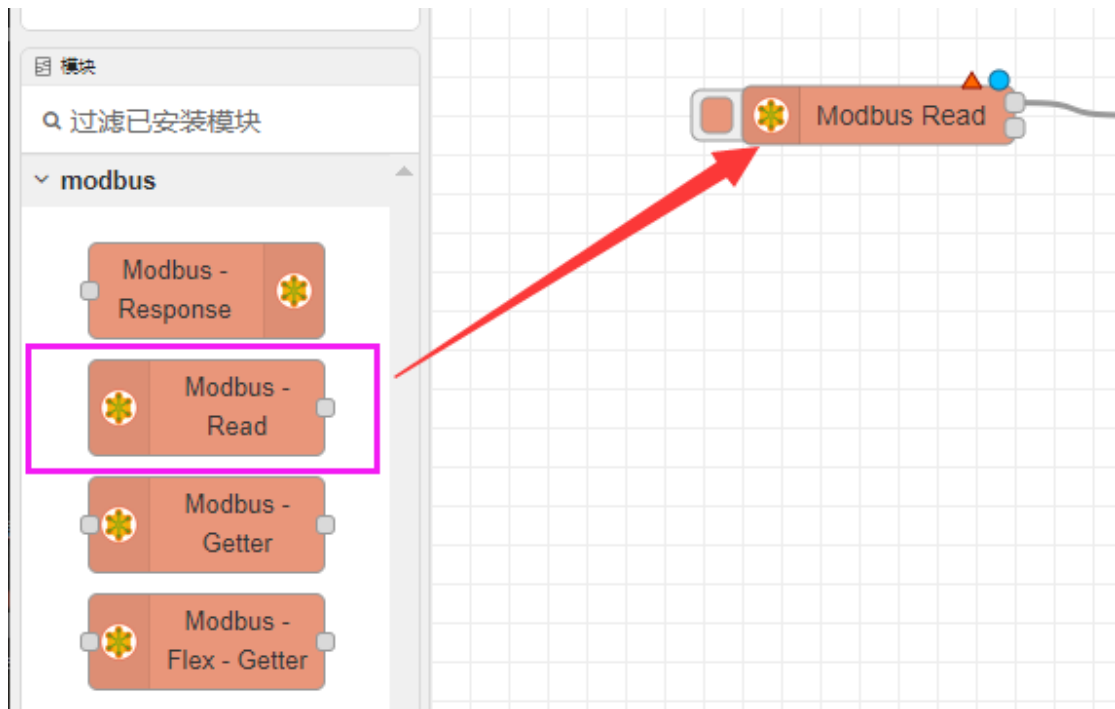


安装完成后检查 Node-RED 左侧工作台，能查到有 modbus 相关节点即表示已安装成功。

13.3.2. modbus RTU 采集数据并转发到 modbus TCP



- 添加 modbus Read 节点用于读取 modbus RTU 数据；



- 双击 modbus-read 节点进行配置：

编辑 Modbus-Read 节点

删除 取消 完成

属性

Settings Options

名称 Name

主题 Topic

Unit-Id 1

FC FC 3: Read Holding Registers

Address 0

Quantity 10

Poll Rate 10 second(s)

Delay to activate input ☐

Allow oversized messages ☐

Server 串口1

1 从机地址

2 寄存器类型为保持寄存器

3 起始地址为0查询10个寄存器地址

4 10s轮询一次

5 指定串口，如无定义号的串口点击+号添加

➤ 串口 1 配置：

编辑 Modbus-Read 节点 > 编辑 modbus-client 节点

删除 取消 更新

属性

Settings Queues Optionals

名称 串口1

Type Serial 1 模式选为串口

Serial port /dev/ttyS1

Serial type RTU-BUFFERD 2 指定串口号，并设置类型为RTU-BUFFERD 波特率为9600

Baud rate 9600

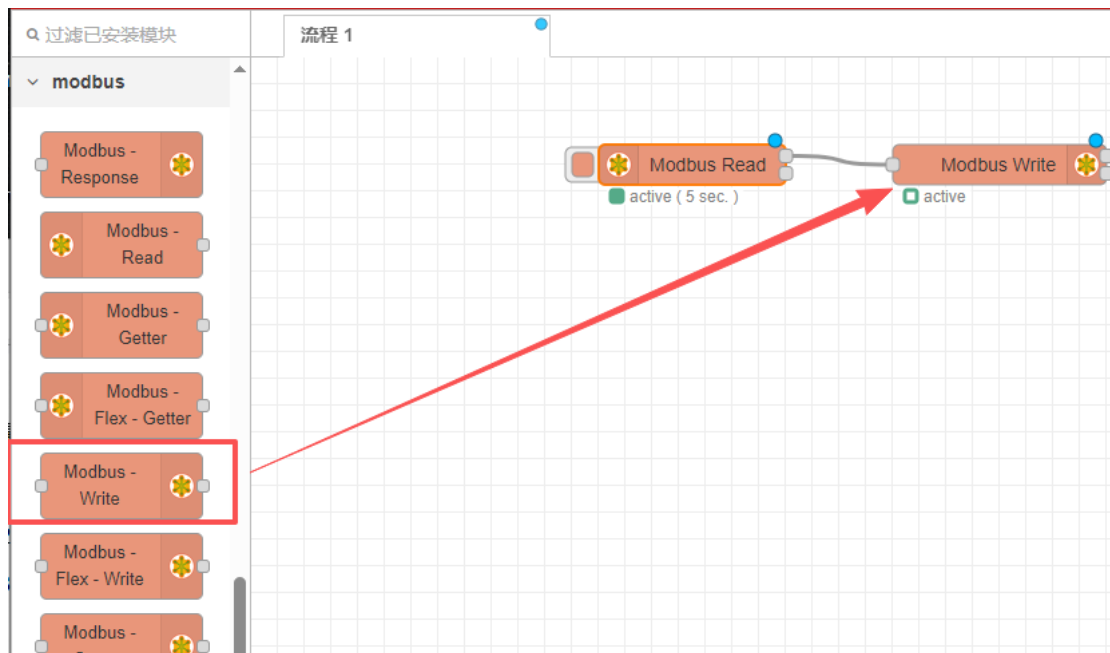
Unit-Id 1

Timeout (ms) 1000

Reconnect on timeout ☒

Reconnect timeout (ms) 2000

- 添加 Modbus Write 节点，发送 Modbus TCP 数据；



➤ 双击 Modbus Write 节点进行配置:

编辑 Modbus-Write 节点

删除 取消 完成

⚙️ 属性

Settings Options

Name: Name

Unit-Id: 1 1 从机地址

FC: FC 16: Preset Multiple Register 2 连续写保持寄存器

Address: 0

Quantity: 10 3 写地址0~10的保持寄存器

⏻ Delay to activate input ☐

Server: modbus-tcp@192.168.1.200:502 4 modbus TCP服务器

➤ 点击+号配置 Modbus TCP 服务器参数:

编辑 Modbus-Write 节点 > 编辑 modbus-client 节点

删除 取消 更新

属性

Settings Queues Optionals

名称 Name

Type TCP 1 服务器类型

Host 192.168.1.200 2 服务器地址和端口号

Port 502

TCP Type DEFAULT

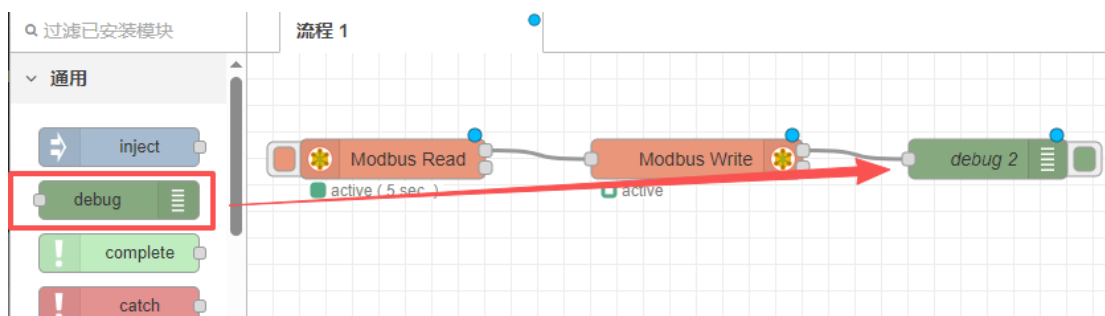
Unit-Id 1 3 服务端从机地址

Timeout (ms) 1000

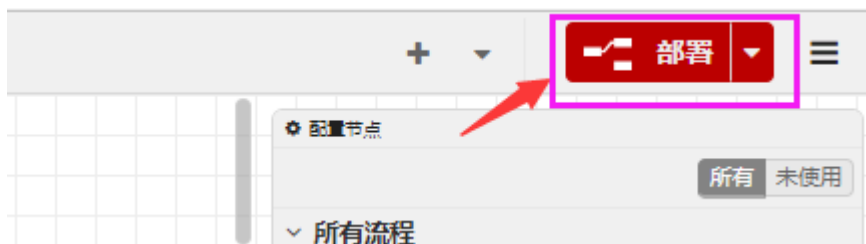
Reconnect on timeout ☒

Reconnect timeout (ms) 2000

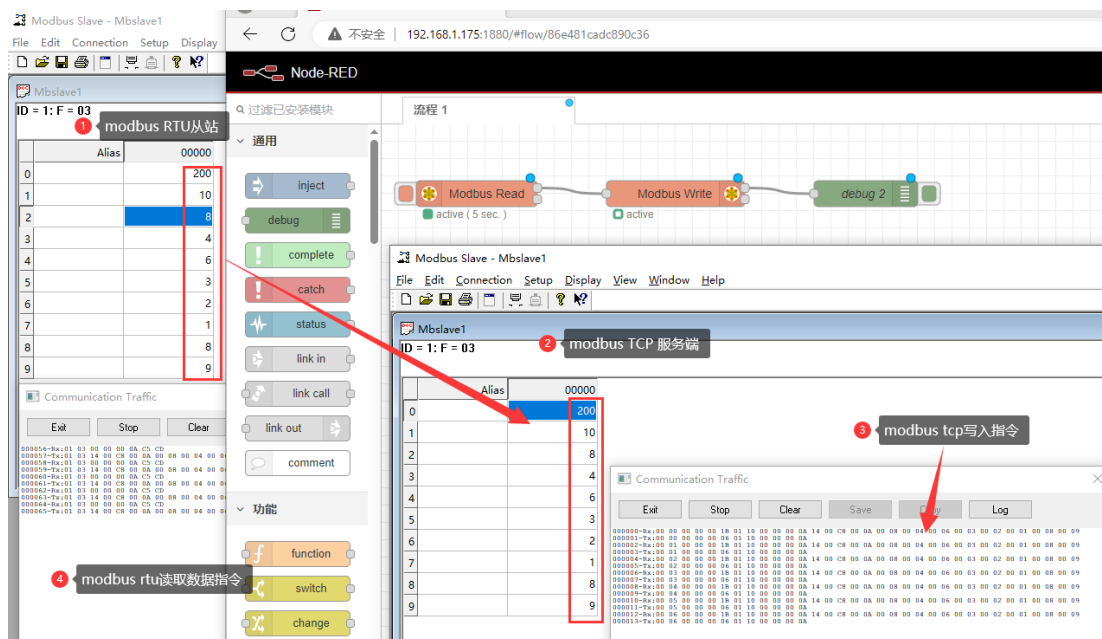
➤ 添加 debug 节点，打印发送的 Modbus TCP 内容：



➤ 配置完成后点击部署生效：



部署完成后 5s，开始自动采集 Modbus RTU 数据并转发到 Modbus TCP。效果如下图：



Node-Red 的 debug 窗口输出：

The screenshot shows the Node-RED interface with a 'debug 2' node on the left and a '调试窗口' (Debug Console) on the right. The debug console displays the following information:

- Time: 2026/9/23 14:22:05
- Node: 节点: debug 2
- Message: polling : msg.payload : Object
- Object structure:
 - value: array[10]
 - 0: 200
 - 1: 10
 - 2: 8
 - 3: 4
 - 4: 6
 - 5: 3
 - 6: 2
 - 7: 1
 - 8: 8
 - 9: 9
 - unitid: "1"
 - fc: 16
 - address: 0
 - quantity: "10"
 - messageId: "6ab3700c633af717630d6762"