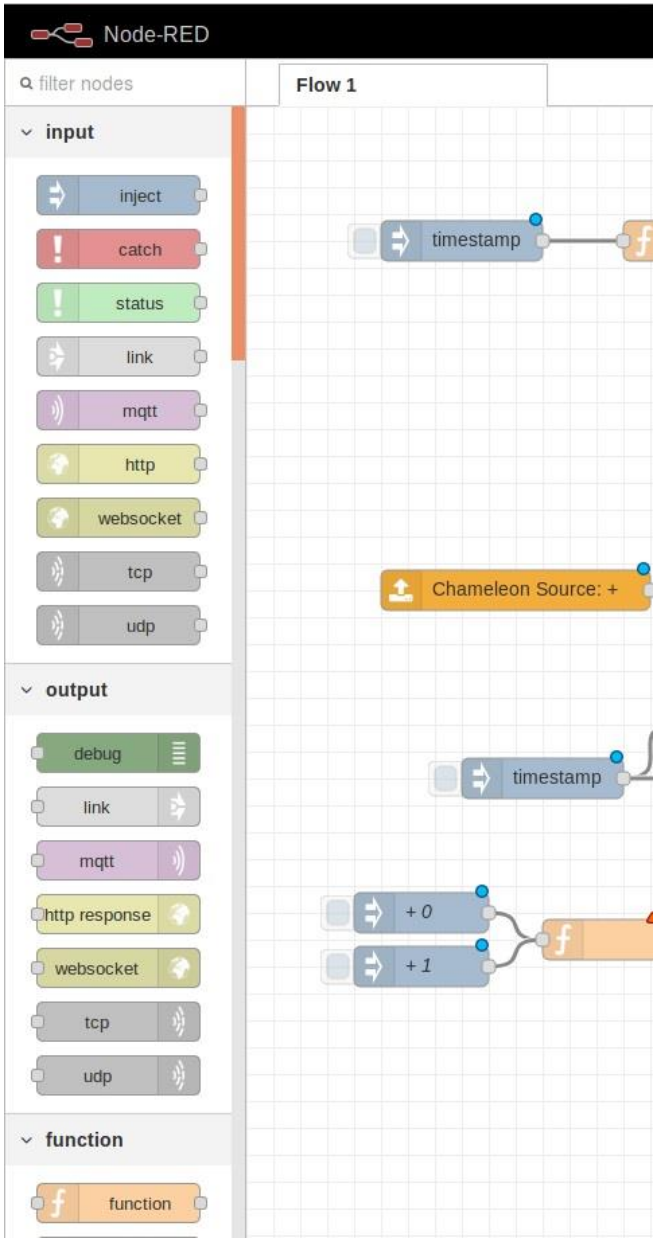
Chameleon
Node-RED 教育訓練

講師: Felix

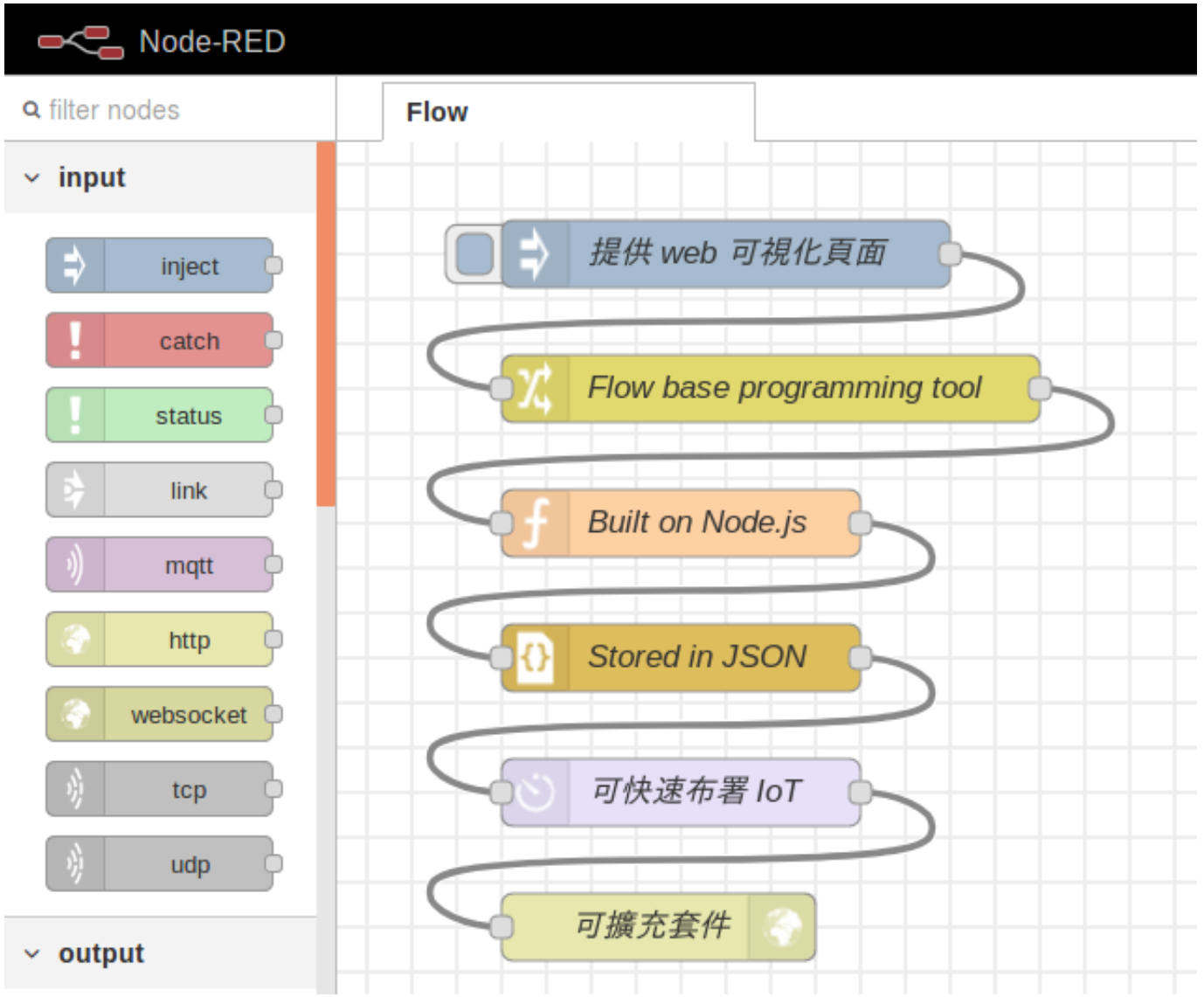
felixchiu@iii.org.tw
www.iii.org.tw



Agenda

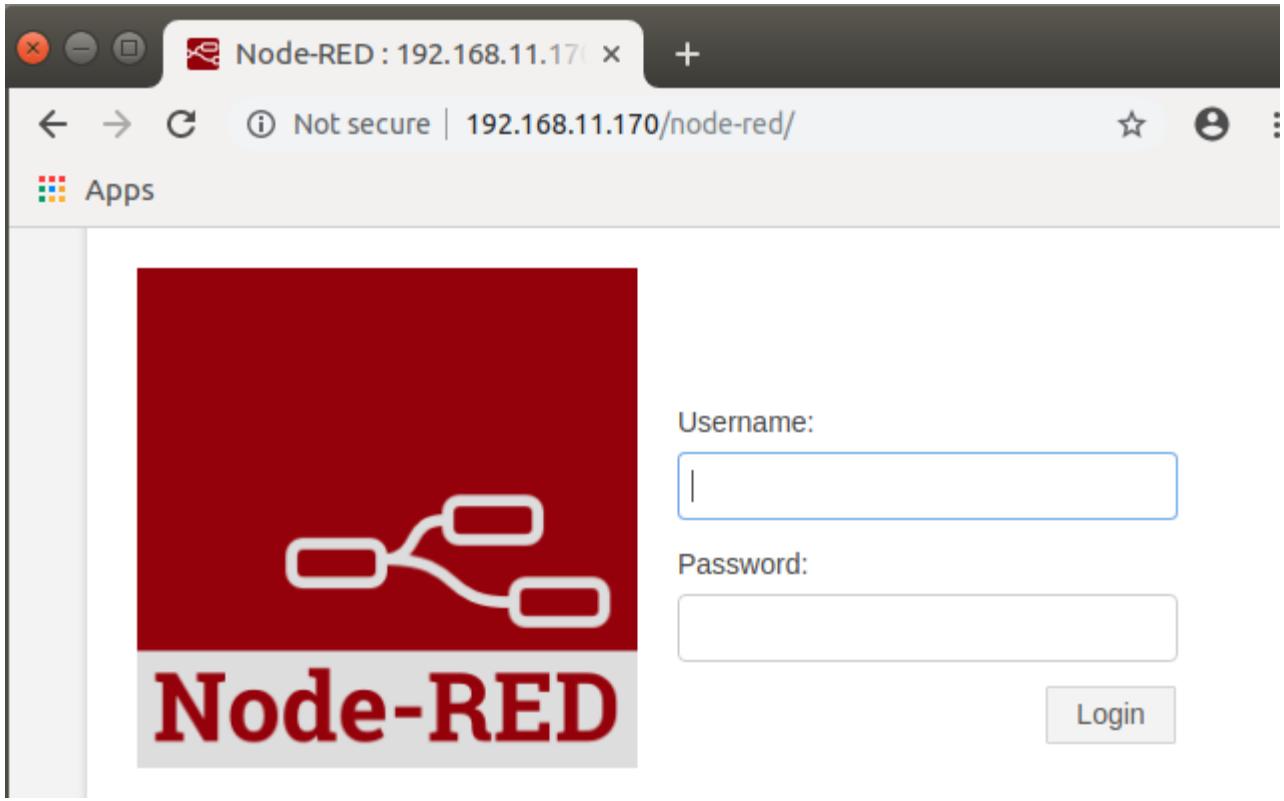
1. Node-RED 是什麼?
2. 基本操作
3. 常用 node 介紹
4. Chameleon x Node-RED
5. 練習 & QnA

What is Node-RED

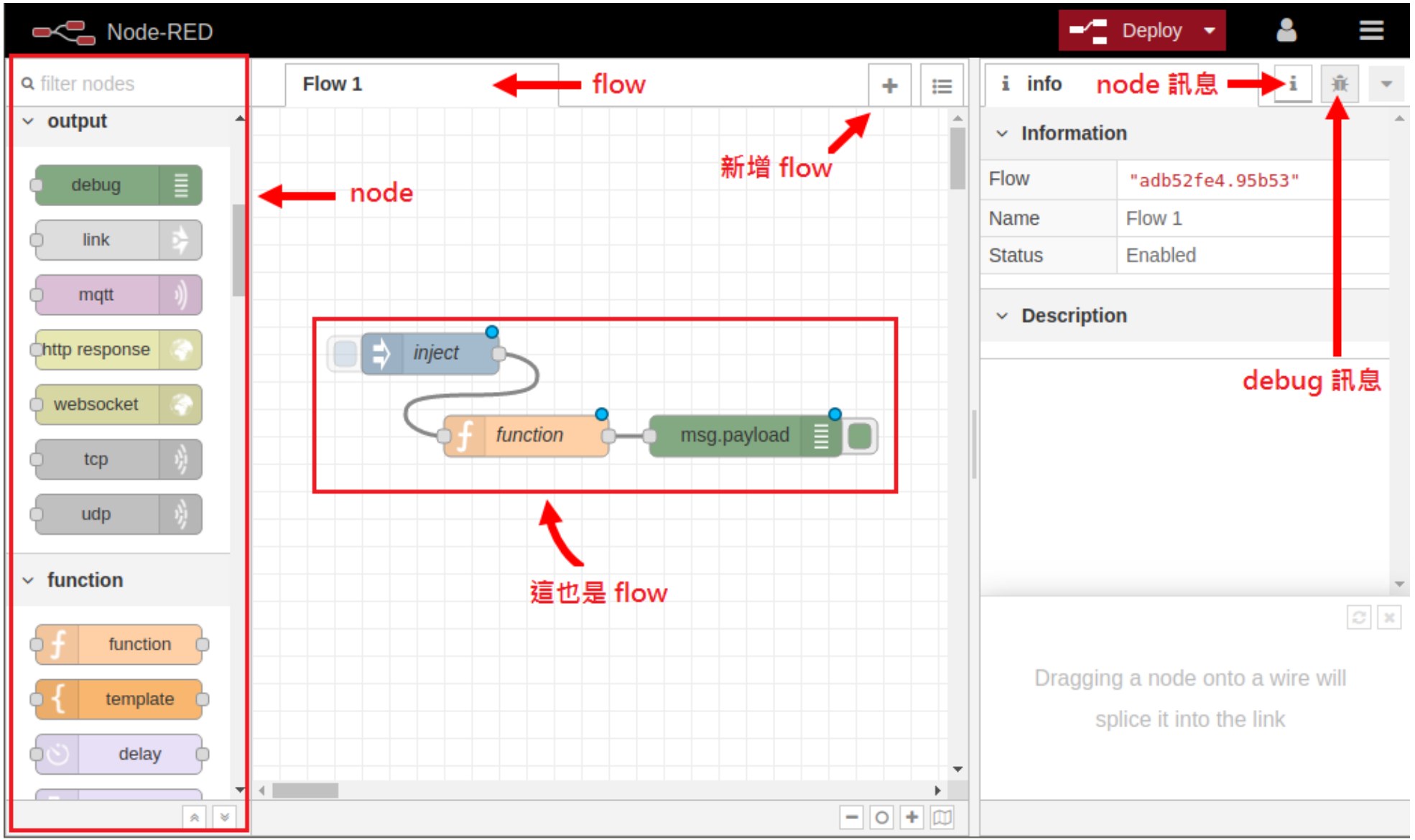


如何開啟 Node-RED

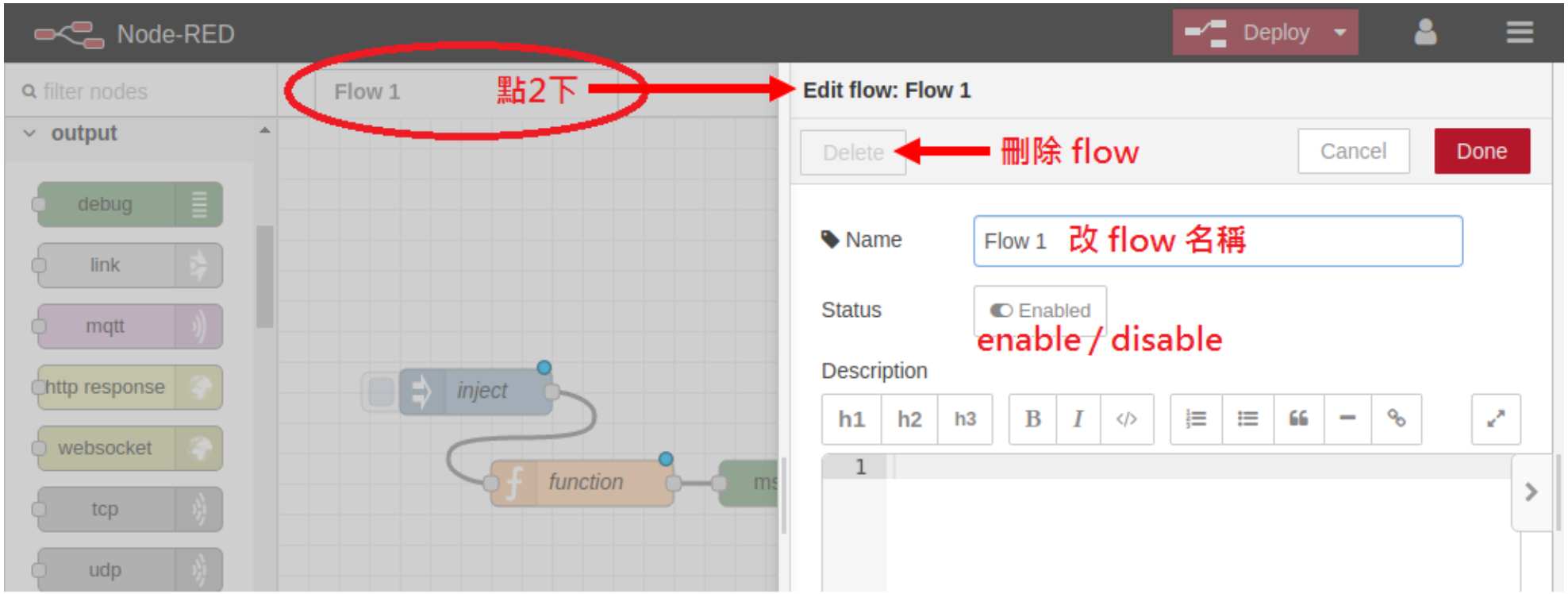
- 變色龍的 URL + /node-red
- 帳號密碼和變色龍的相同



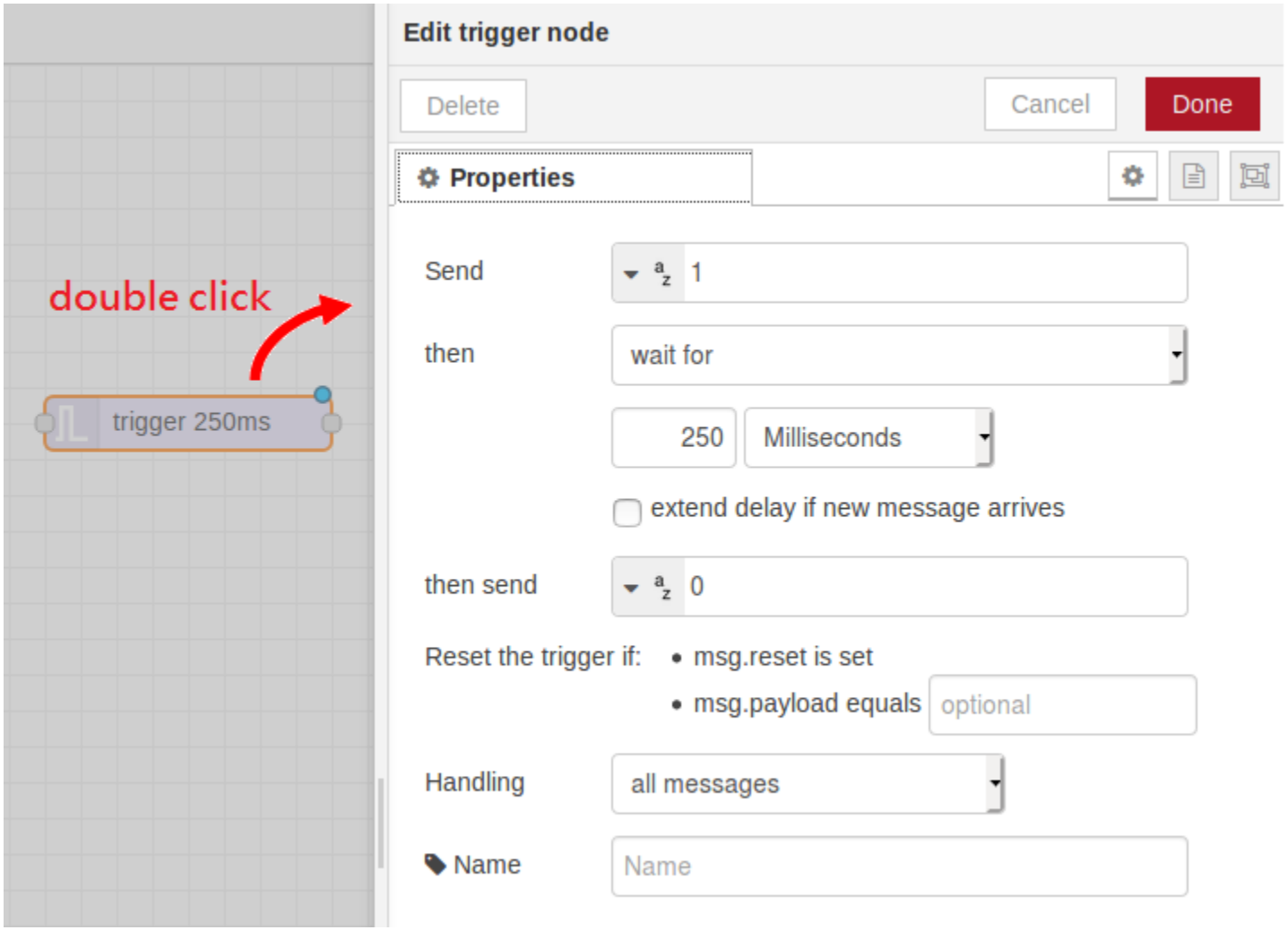
基本介紹



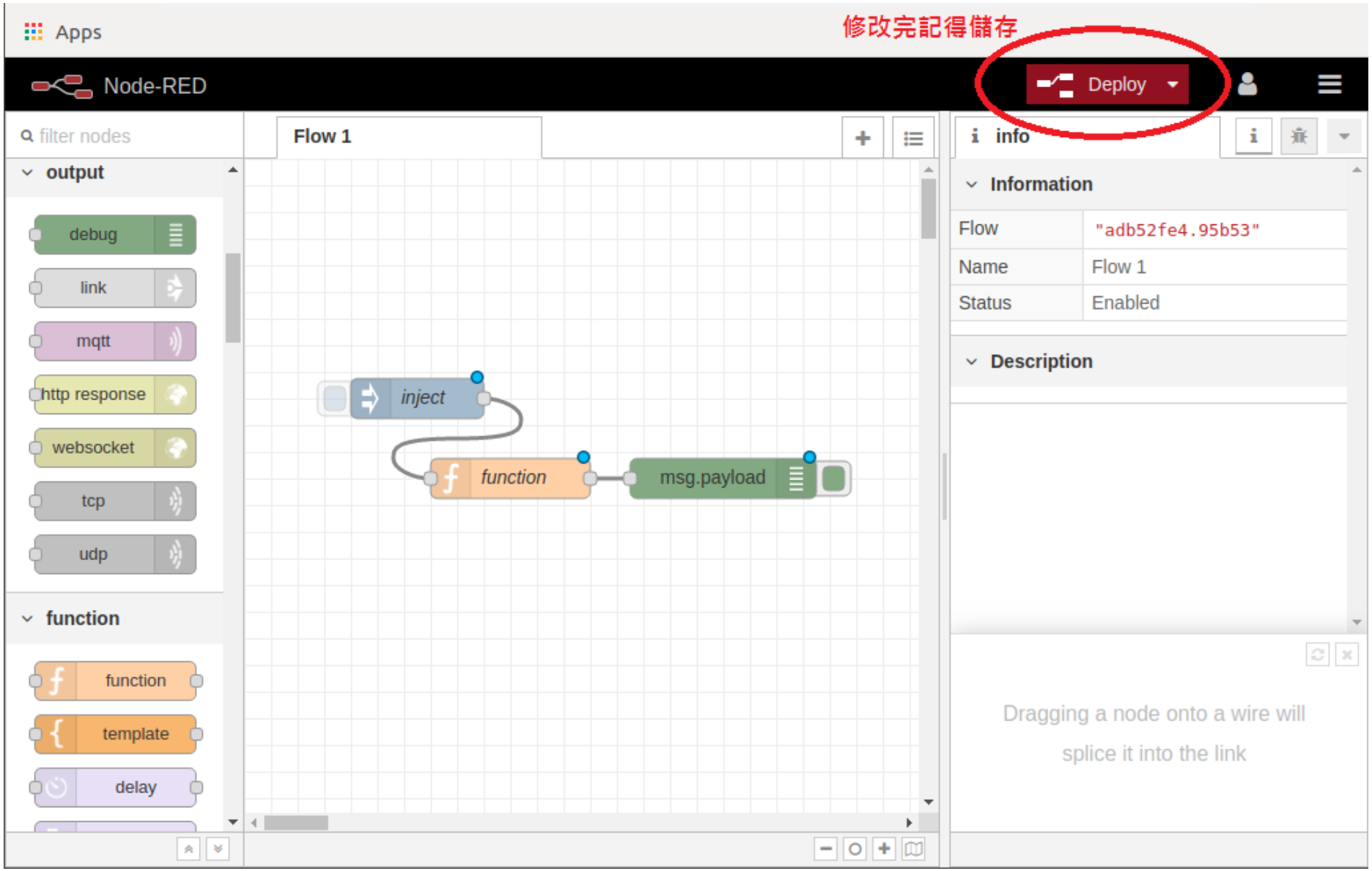
編輯 flow



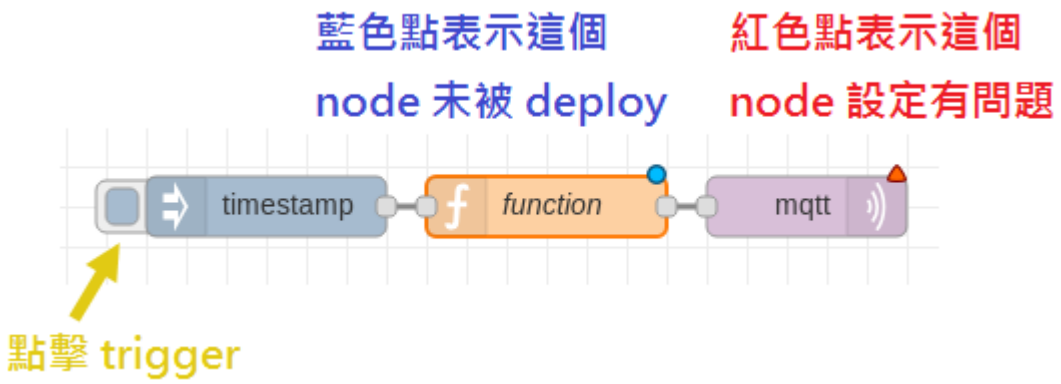
編輯 node

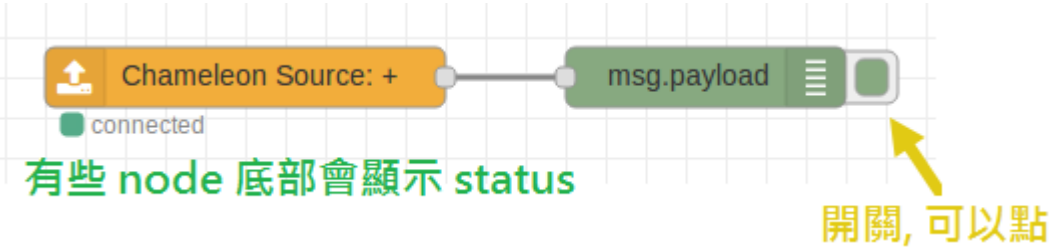


儲存

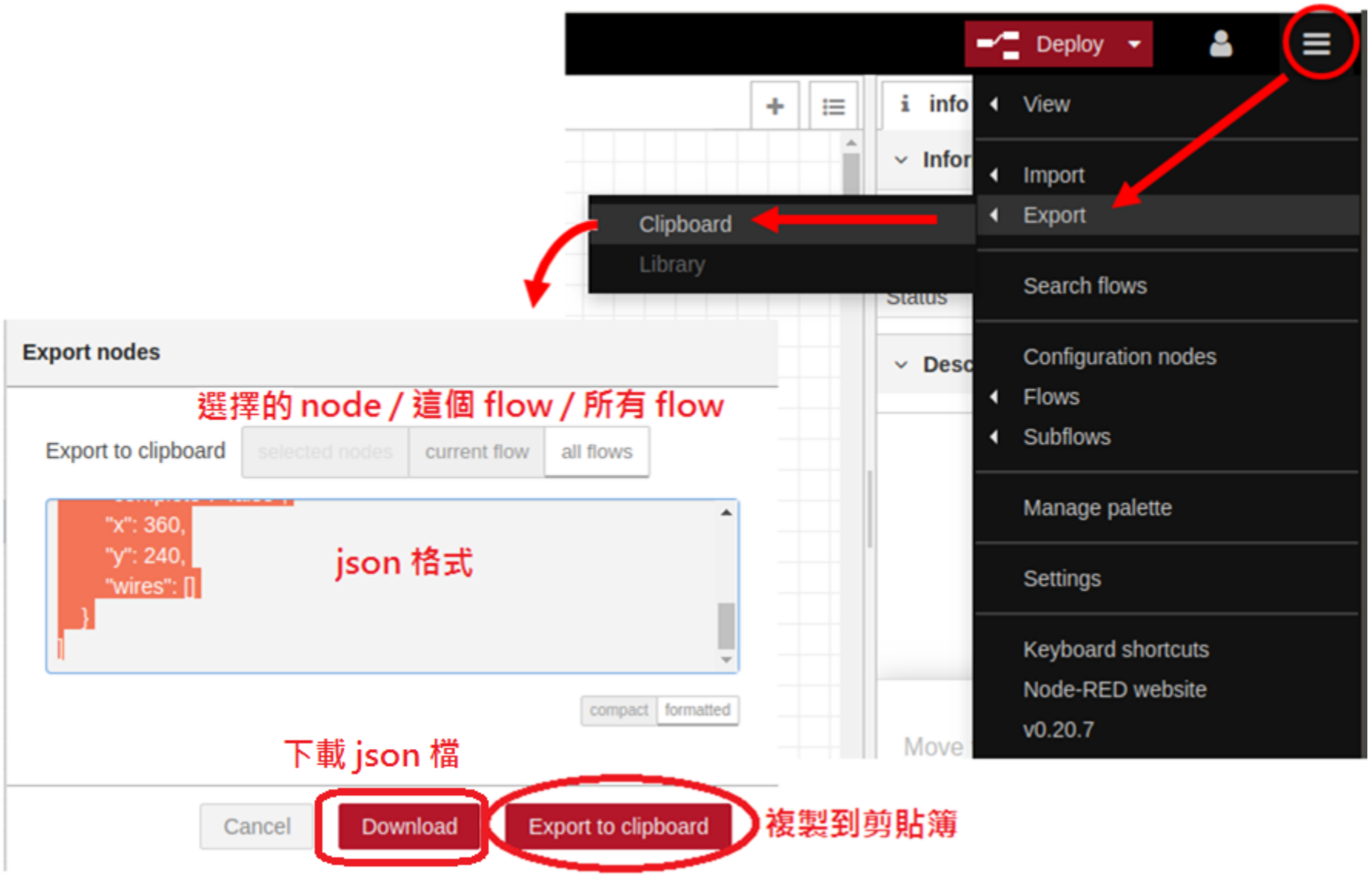


node status

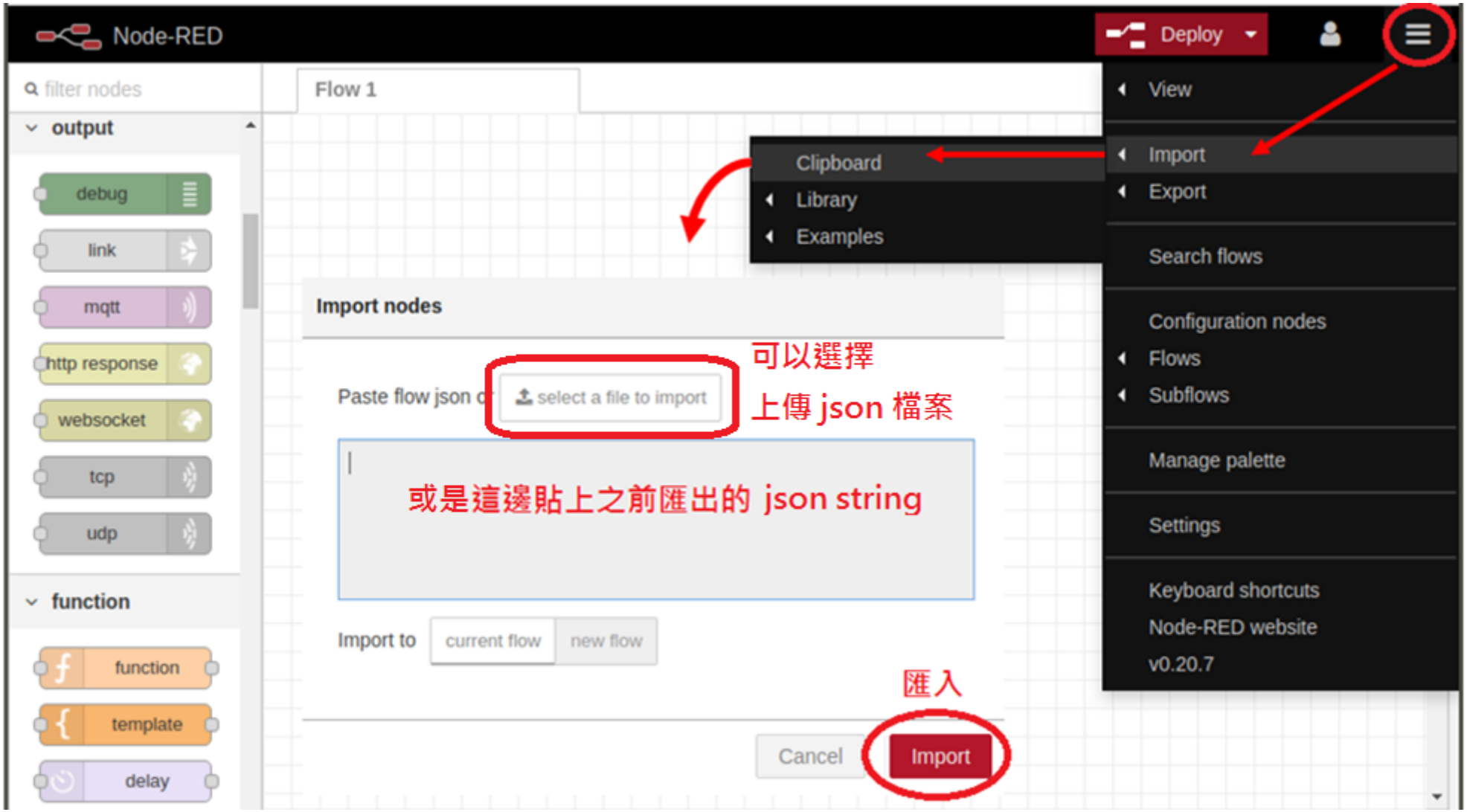




匯出 flow or node



匯入 flow or node



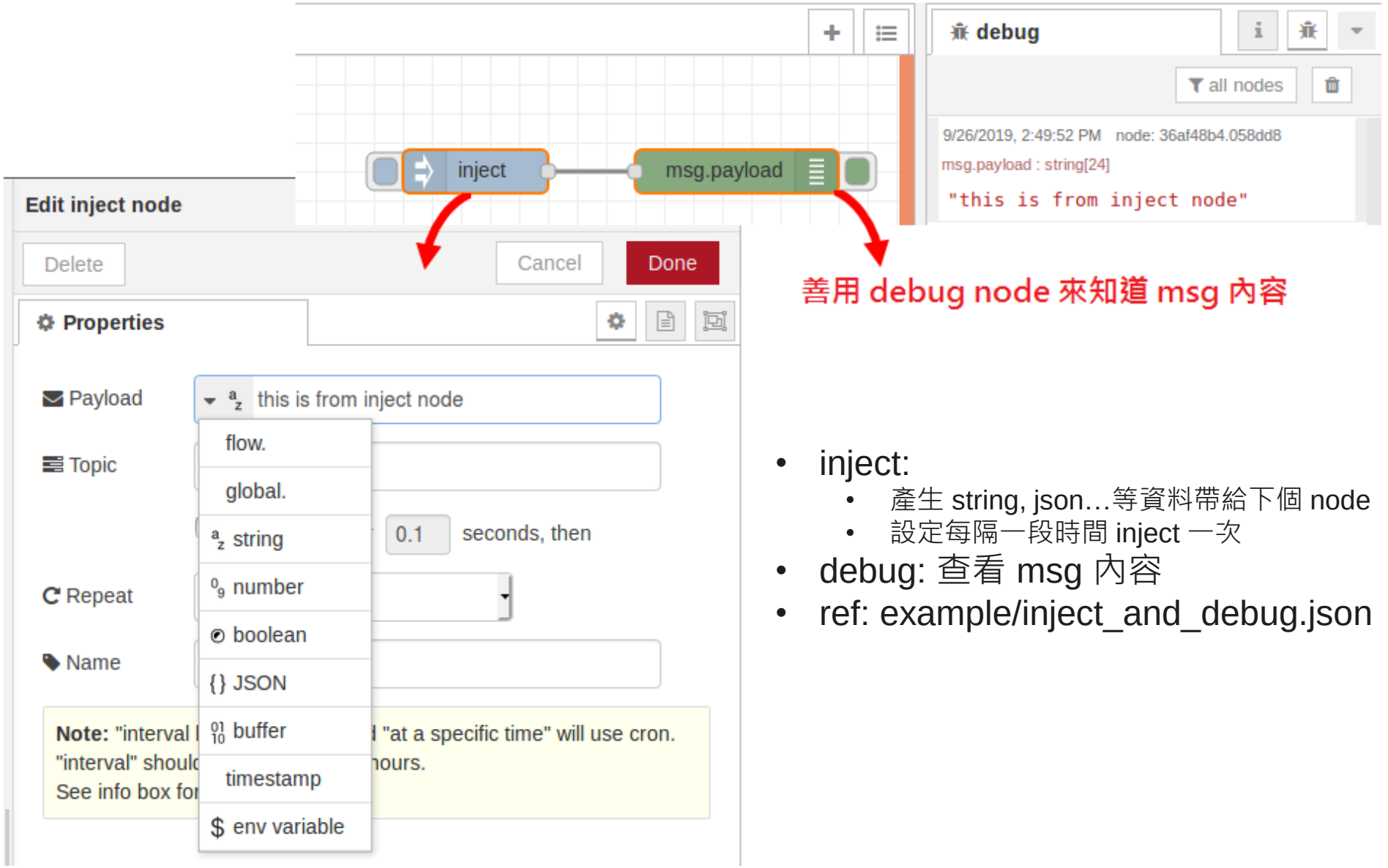
node & message

- node 是最基本的元件
- node 之間使用線連接來組成 flow
- node 與 node間傳遞使用 message, 程式中使用 msg variable 來取得訊息
- msg 的 type 是 JavaScript objects, 其中 payload property 是最常用來放要給下個 node 的資料
  - msg.payload
  - payload 內容可以是任何的 JavaScript type

常用 node 介紹

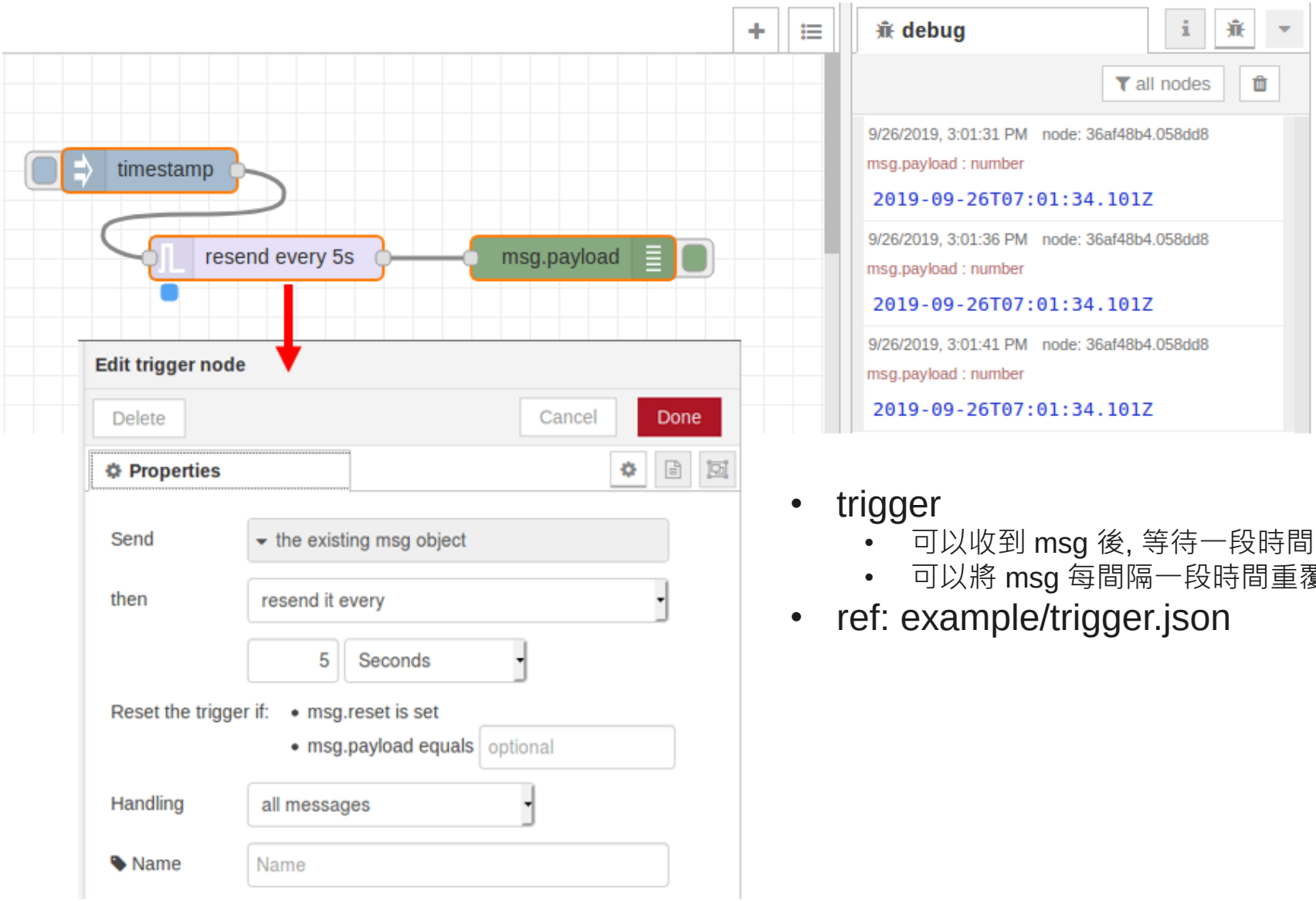
- inject
- debug
- trigger
- delay
- function
- switch
- json
- http request
- mqtt

inject & debug node



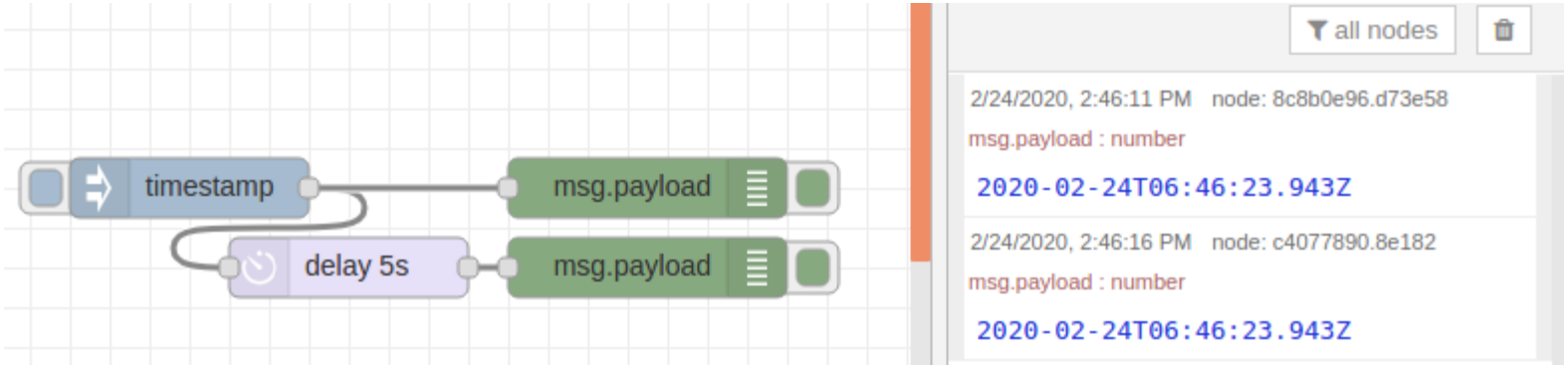
- inject:
  - 產生 string, json…等資料帶給下個 node
  - 設定每隔一段時間 inject 一次
- debug: 查看 msg 內容
- ref: example/inject_and_debug.json

trigger node



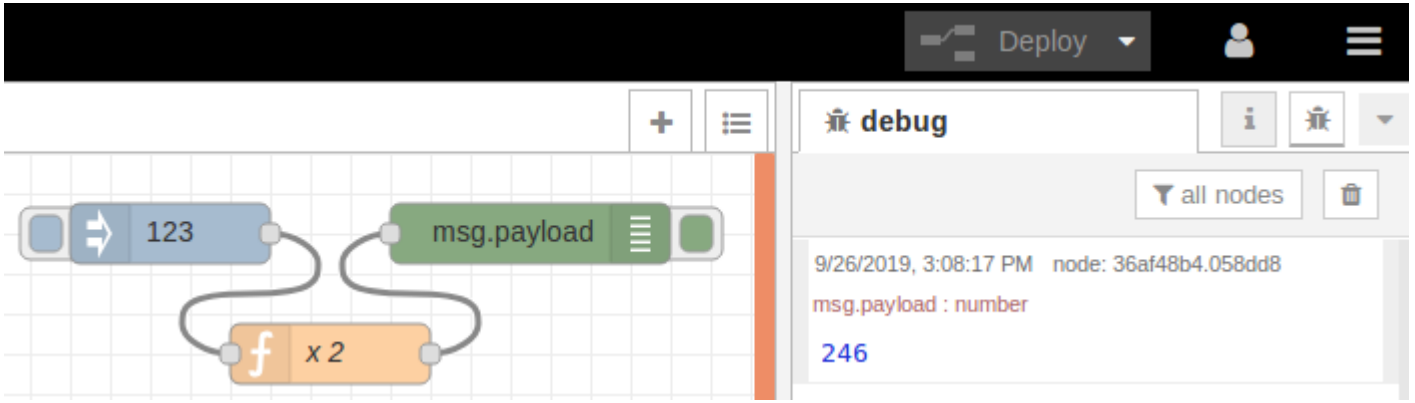
- trigger
  - 可以收到 msg 後, 等待一段時間後再送給下個 node
  - 可以將 msg 每間隔一段時間重覆送給下個 node
- ref: example/trigger.json

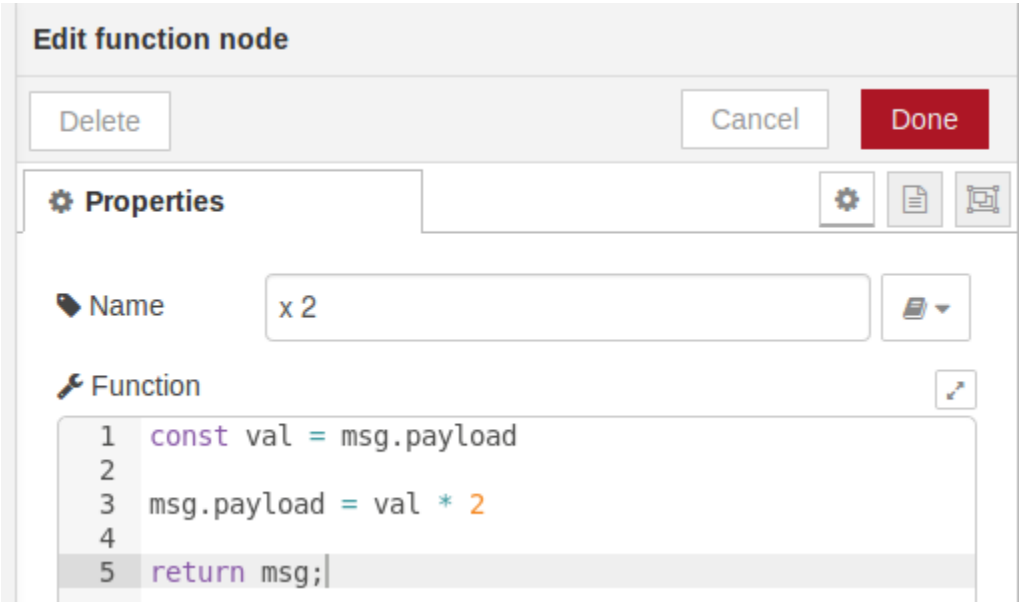
delay node



- delay: 將收到的 msg 留住, 等待一段時間後再送給下個 node
- ref: example/delay.json

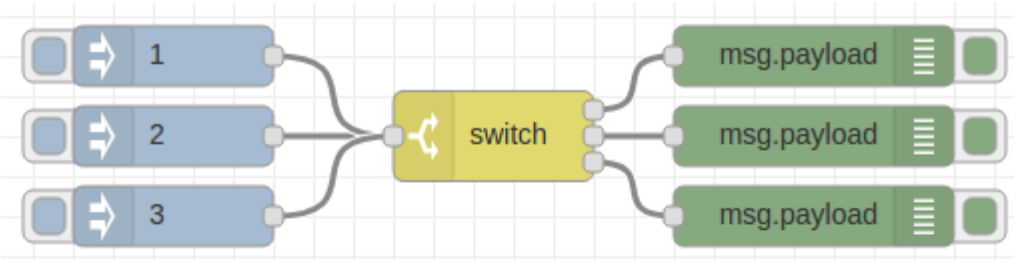
function node

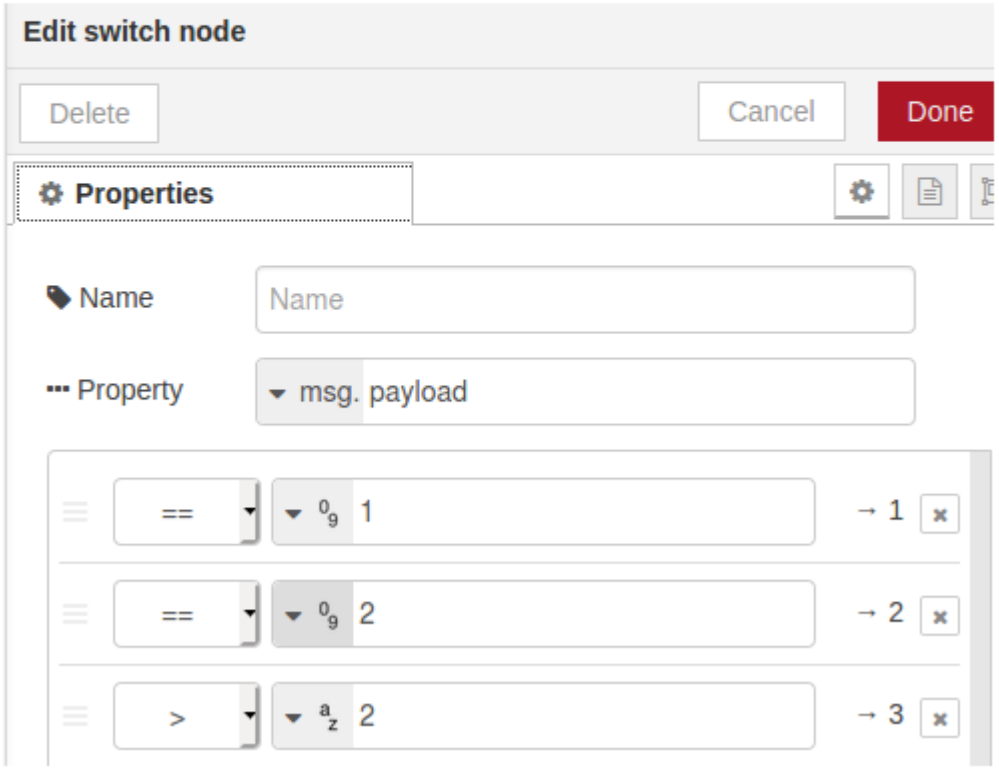




- function: 可寫自己的邏輯在裡面, 開發需要的功能
- ref: example/function.json

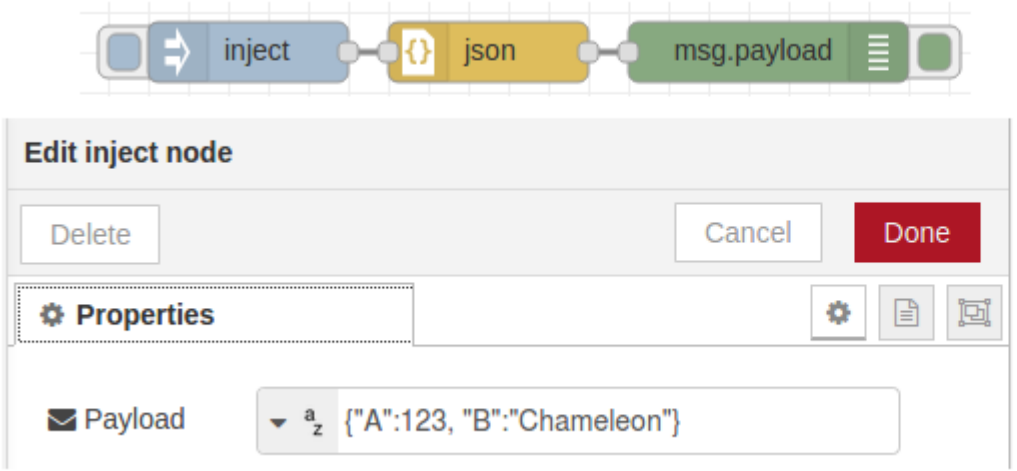
switch node

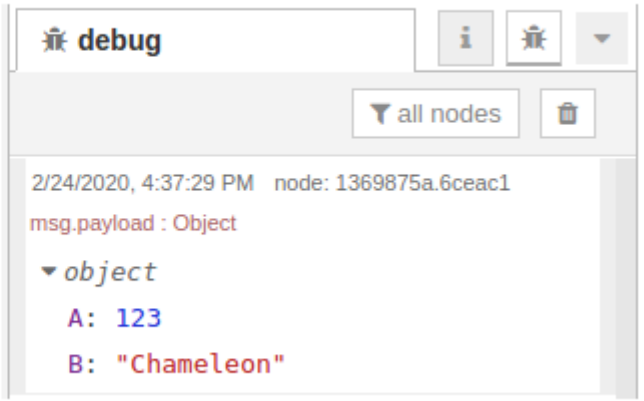




- switch: 將 msg 分送到不同 flow
- ref: example/switch.json

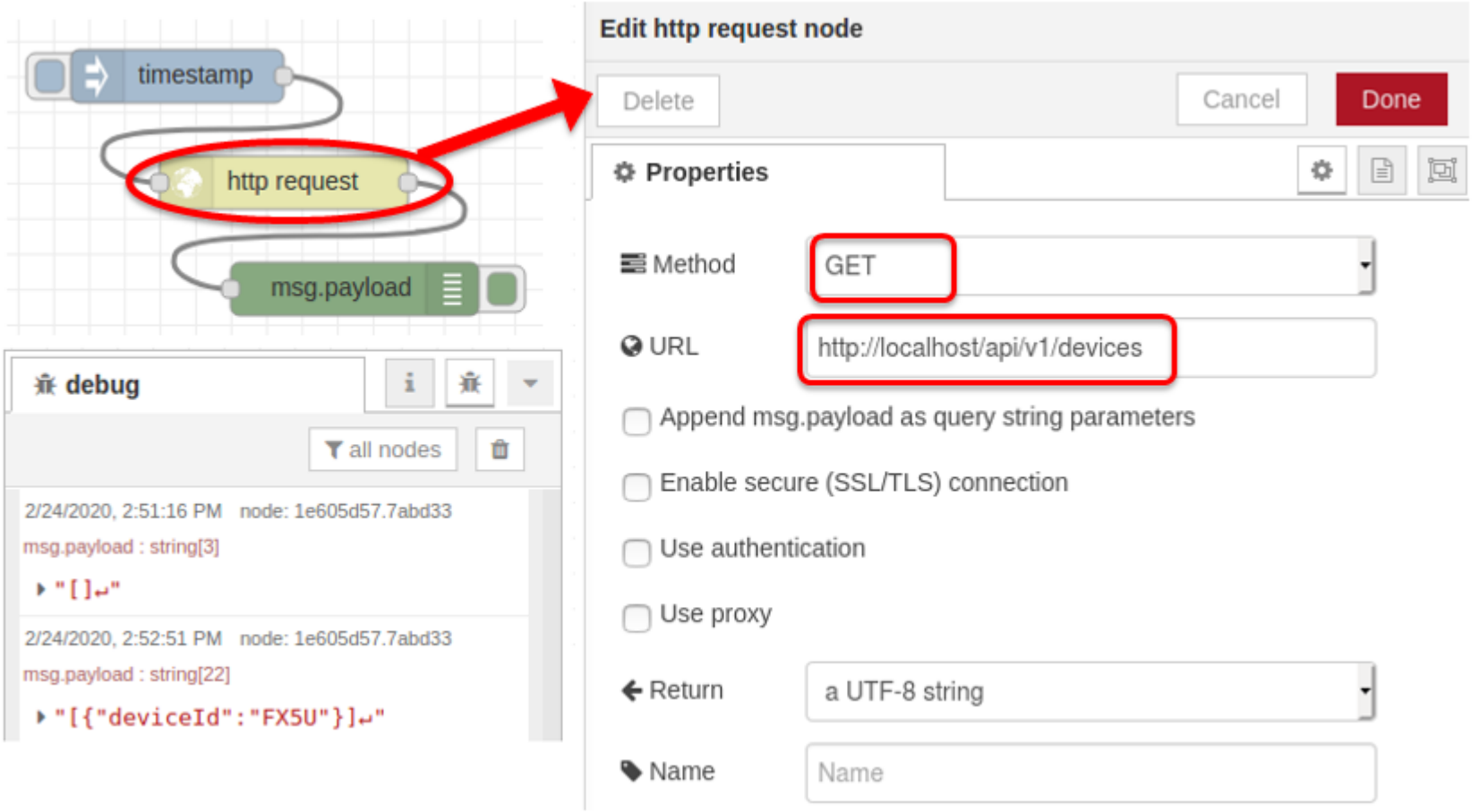
json node





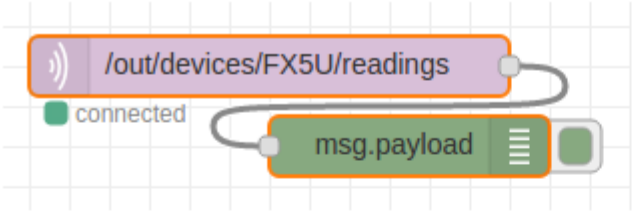
- json: 做 json 轉換
- ref: example/json.json

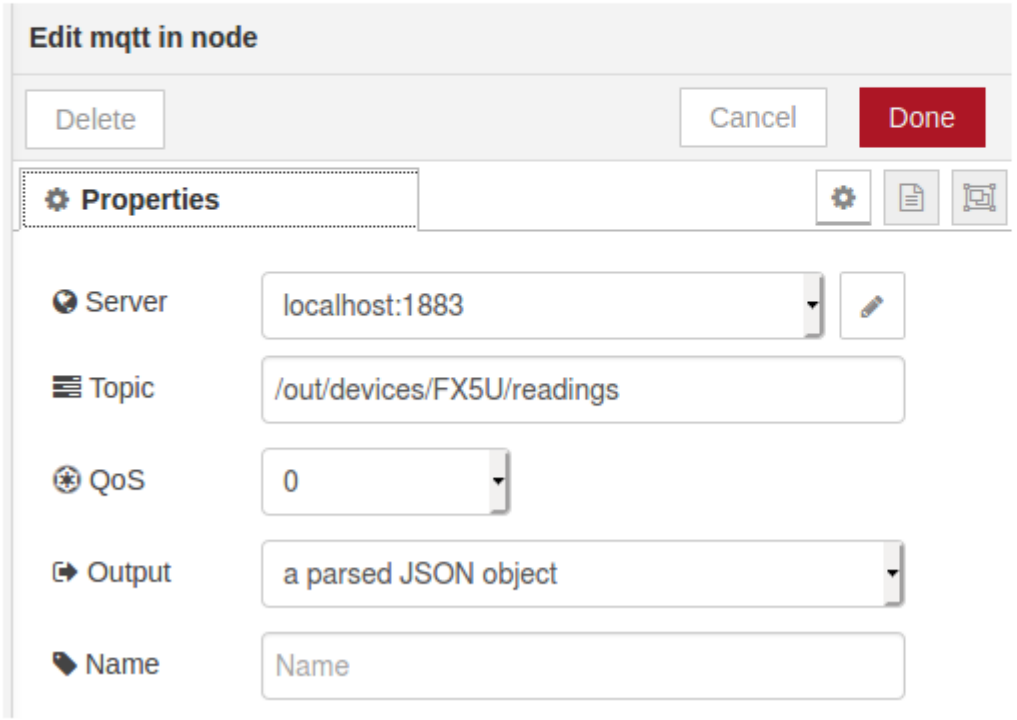
http request node

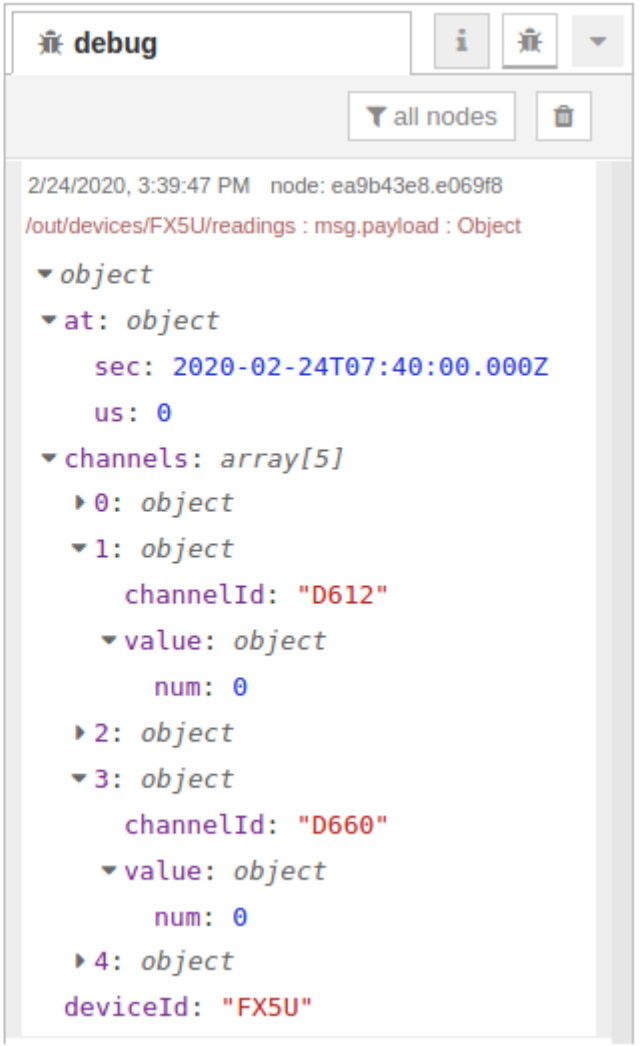


- http request: 可做GET,POST,PUT, DELETE的 http request
- ref: example/http_request.json

mqtt node







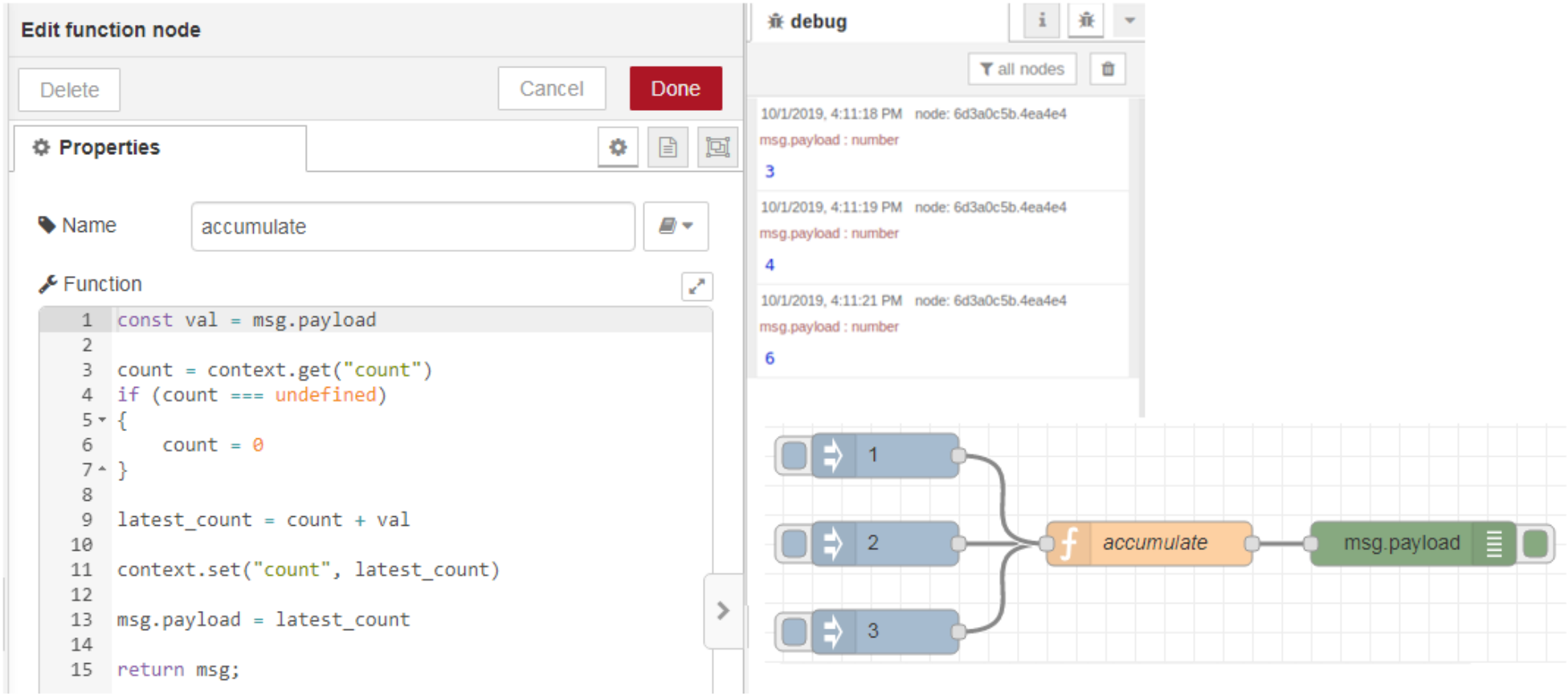
- mqtt in: 接收 mqtt 資料
- ref: example/mqtt_in.json

- 上機練習
- 休息 10 分鐘

context

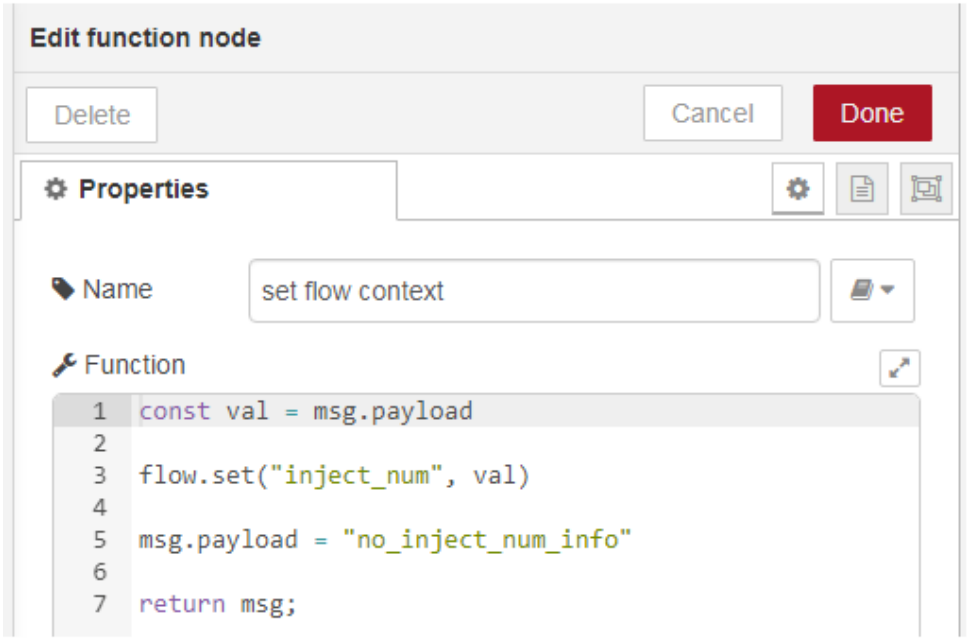
- 用來儲存資料的一種方法
- Context 分成
  - Node: 只有這個 node 可存取
  - Flow: 只有這個 flow (tab) 內的 node 可存取
  - Global: 所有 flow (tab)內的 node 都可以存取
- (預設) context 內容會在系統重啟情況下被清空, 若要避免被清空需要其他設定或是 plug-in
- 若 context get 不到, 值會是 undefined

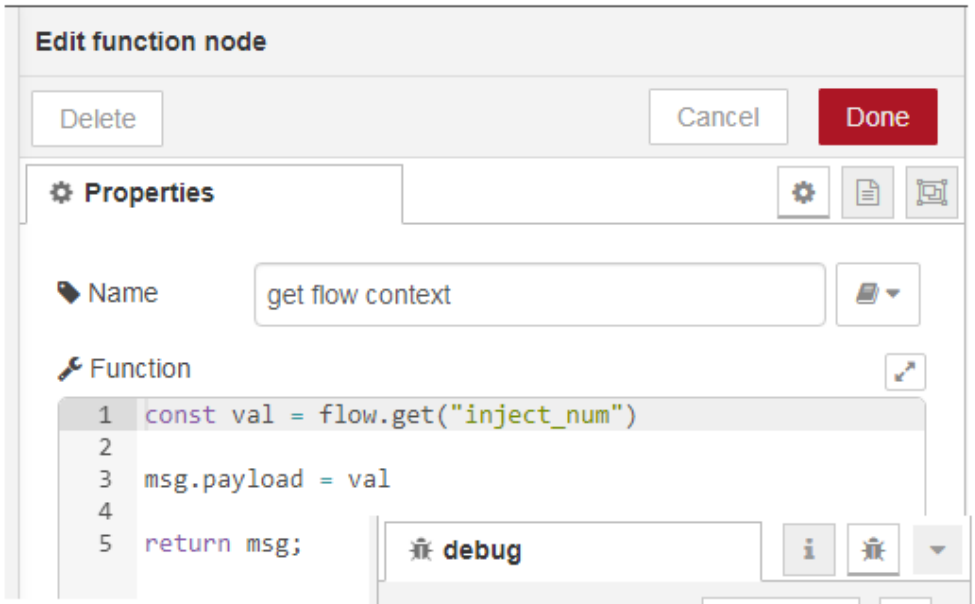
Node scope context

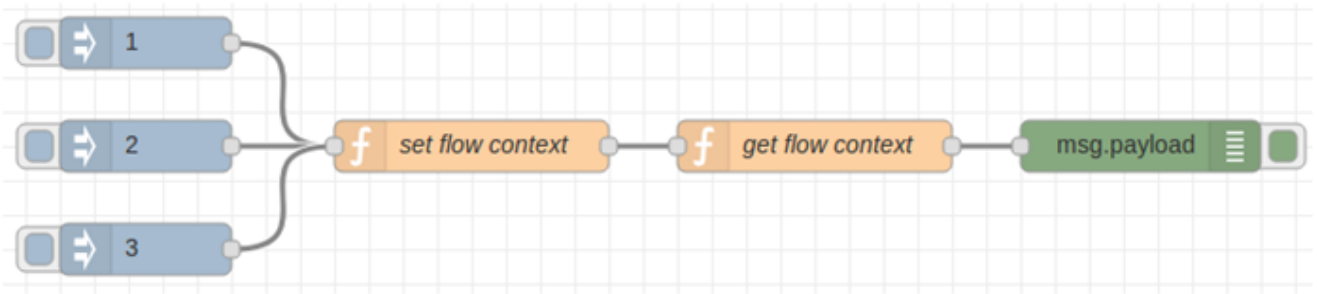


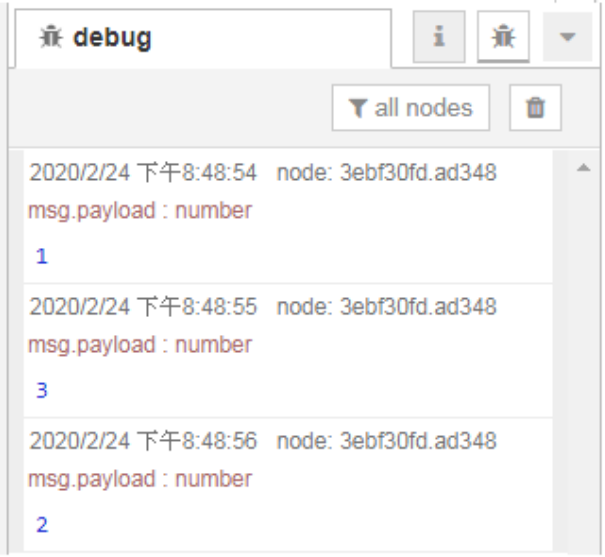
- 使用 predefined variables context
- 重新 deploy 會清空 node context
- ref: example/node_context.json

Flow scope context





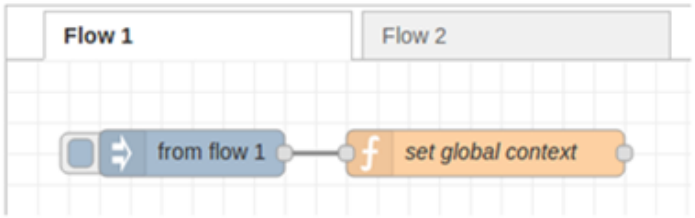




- 使用 predefined variables flow
- ref: example/flow_context.json

Global scope context

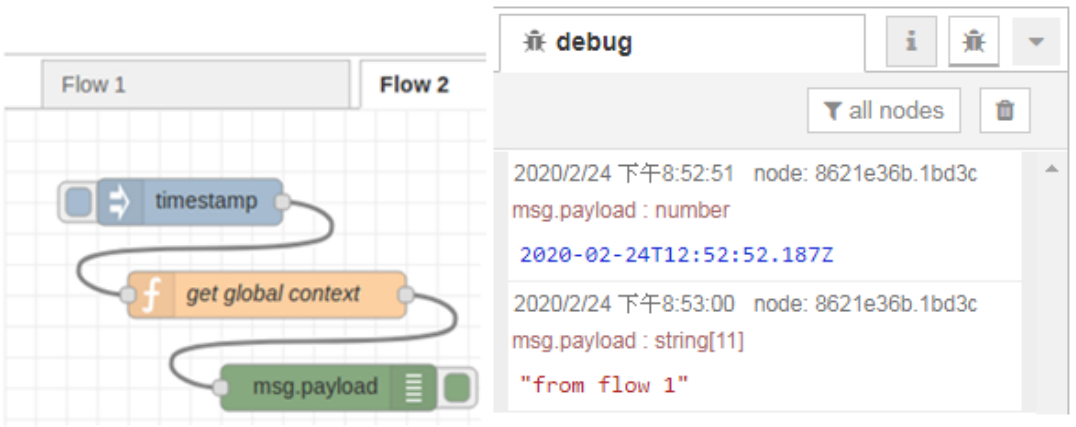
- 使用 predefined variables global
- ref: example/global_context.json



Name set global context

Function

```
1  const message = msg.payload
2
3  global.set("flow1_info", message)
4
5  return msg;
```



Name get global context

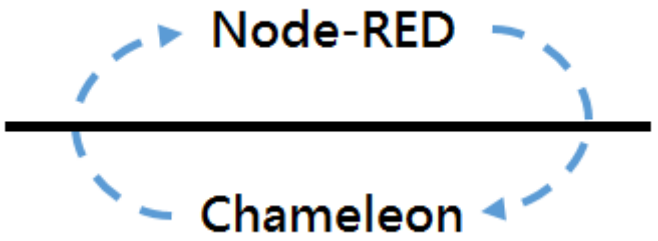
Function

```
1  const message = global.get("flow1_info")
2
3  if (message !== undefined)
4  {
5      msg.payload = message
6  }
7
8  return msg;
```

- 課堂練習
- QnA
- 中場休息

Chameleon x Node-RED

- Node-RED 上可收到變色龍所有資料
- 在 Node-RED 處理完的資料可拋回變色龍
- 常用的功能已包成 node, 減少額外開發的時間



前置作業

- Chameleon 頁面新增 Mitsubishi FX5U 機台
  - ip: 貼在 plc 上
  - port: 9600
  - 3E:ASCII
- 點位表: (fx5u.xlsx)

點位	意義	長度	備註
X0	Switch alarm	1	Alarm 點
D602	溫度	1	乘 0.1
D612	濕度	1	乘 0.1
D650	Count	1	每秒加 1
D660	Double count	1	每0.1秒加 1

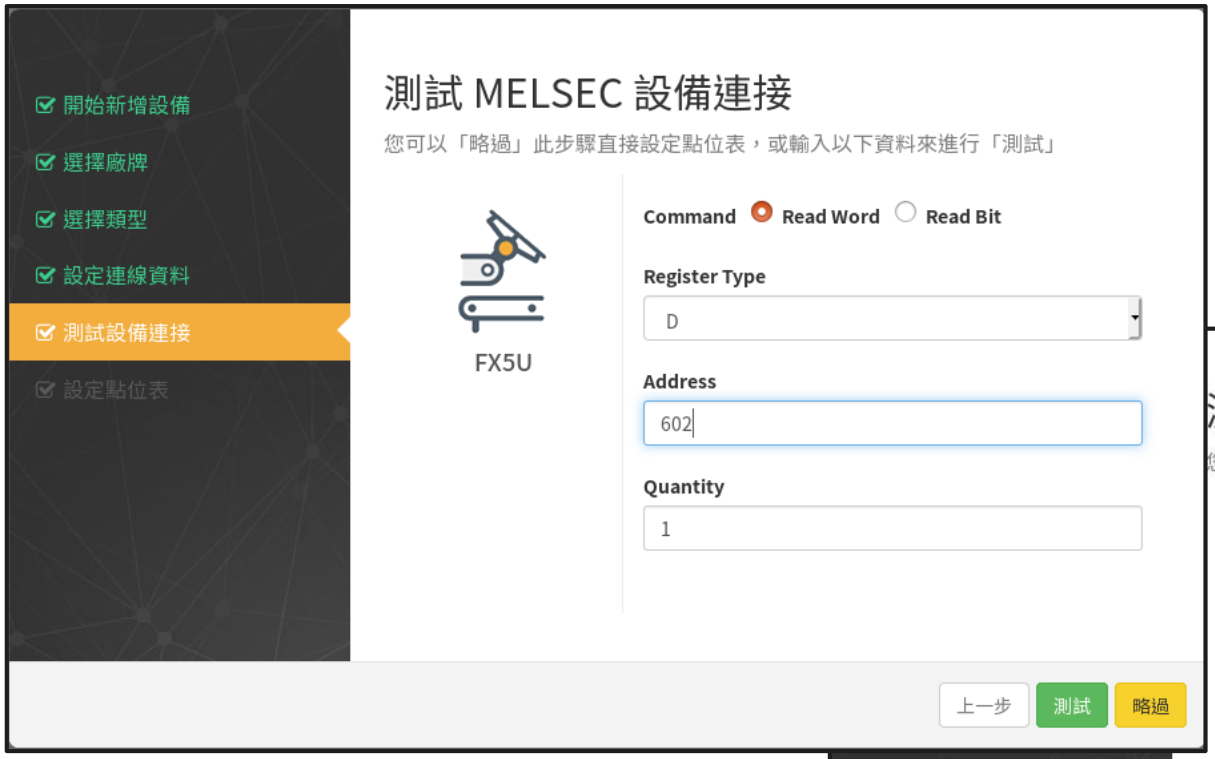
新增產線

新增機台

新增機台-選擇PLC廠牌

新增機台-設定連線資料

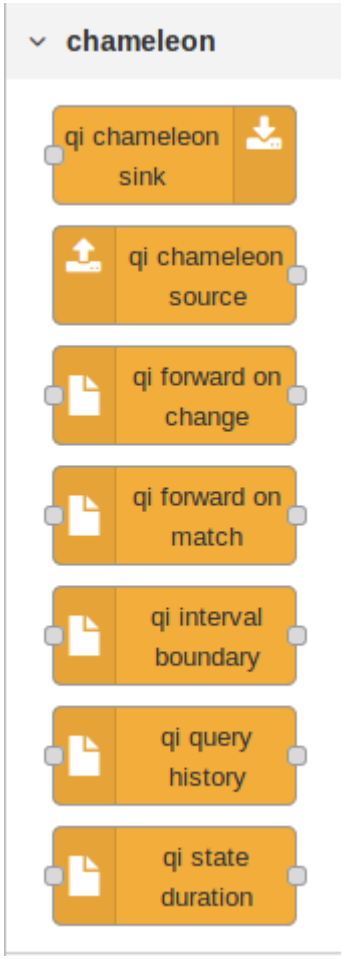
新增機台-連線測試

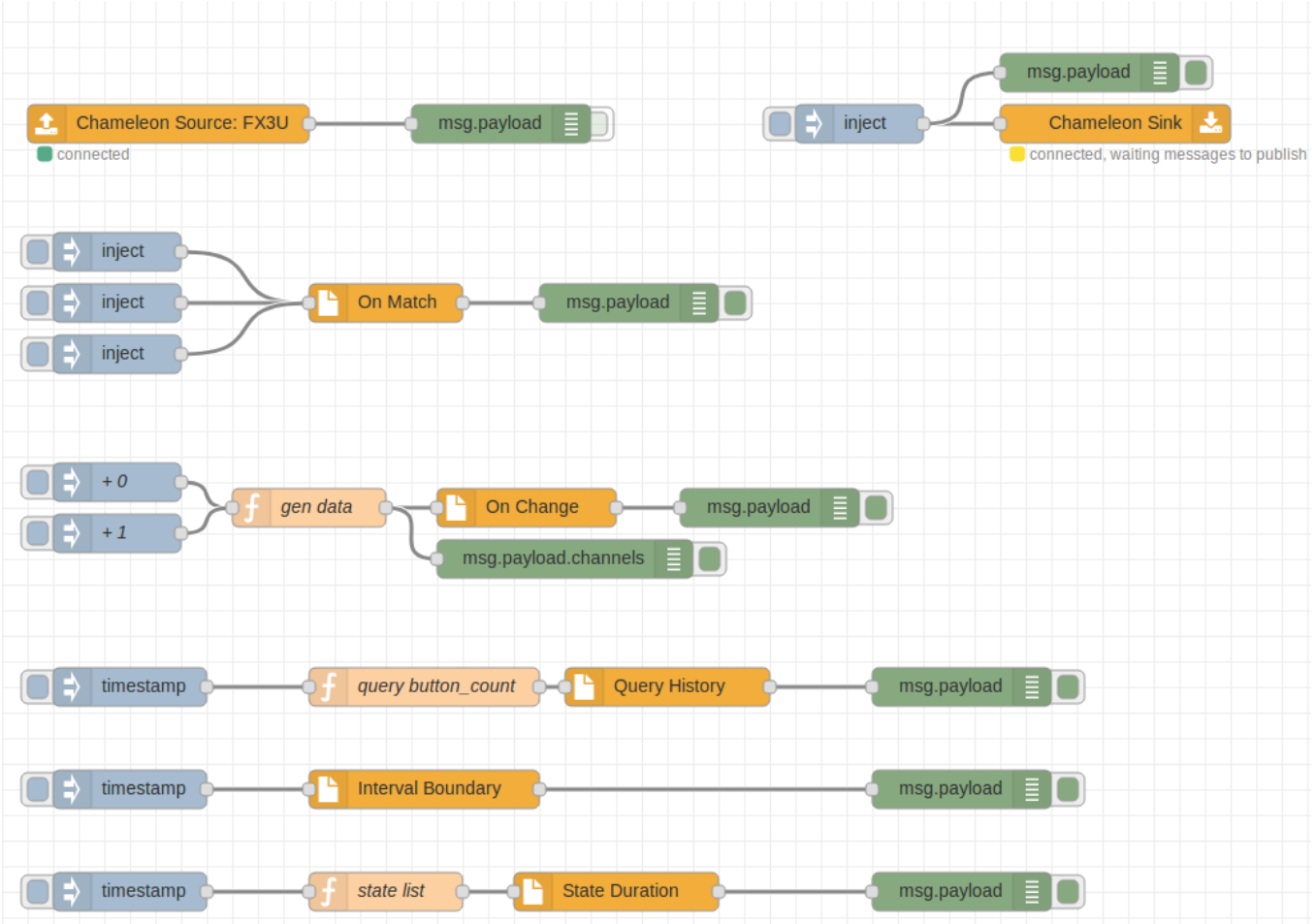


新增機台-上傳點位表

新增機台-完成

Node with Chameleon





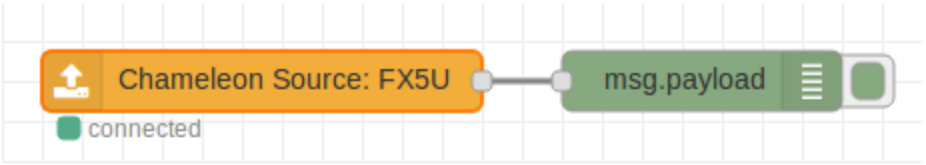
qi chameleon source

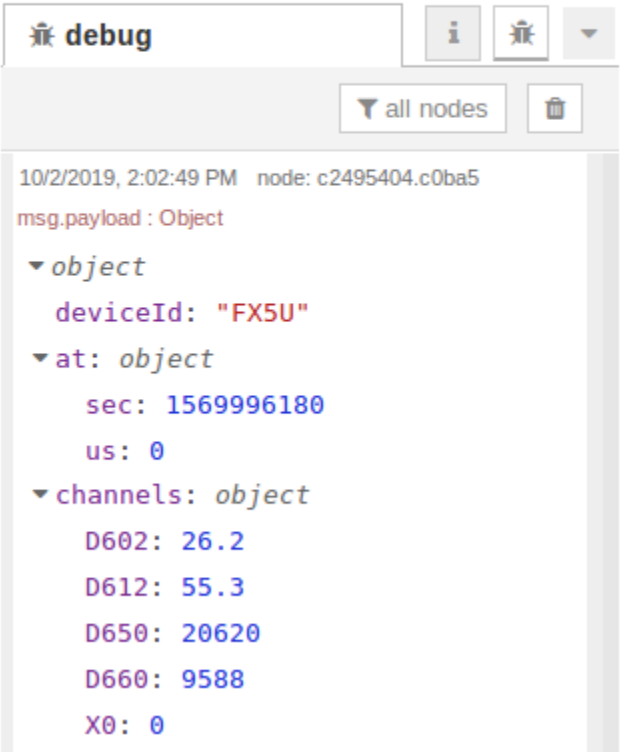
- 從 chameleon 接收資料
- 可在 node 編輯頁面設定要接收的 device id 或 “+” 接收所有 device 的資料
- 資料輸出格式

```
{
  "deviceId": "string",
  "at": {
    "sec": integer,
    "us": integer
  },
  "channels": {
    "channel": value
  }
}
```

```
{
    "deviceId":"FX5U",
    "at":{
        "sec":1569911173,
        "us":0
    },
    "channels":{
        "D602":25.7,
        "D612":57.5,
        "D650":101,
        "D660":1006,
        "X0":0
    }
}
```

qi chameleon source





ref: example/chameleon-source.json

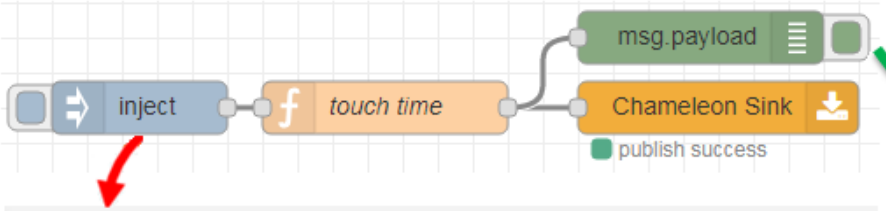
qi chameleon sink

- 將資料拋回 chameleon
- Chameleon 需先增加虛擬設備
- 資料輸入格式

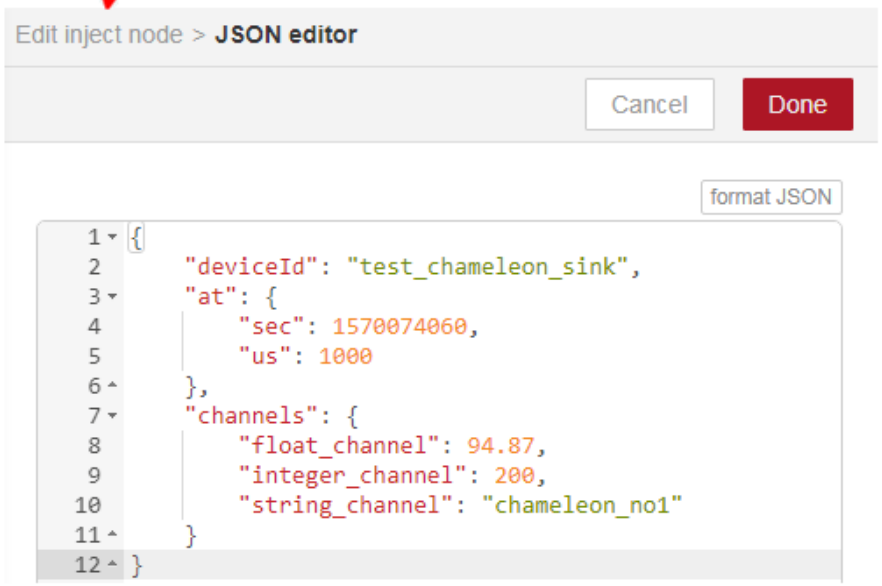
```
{
  "deviceId": "string",
  "at": {
    "sec": integer,
    "us": integer
  },
  "channels": {
    "channel": value
  }
}
```

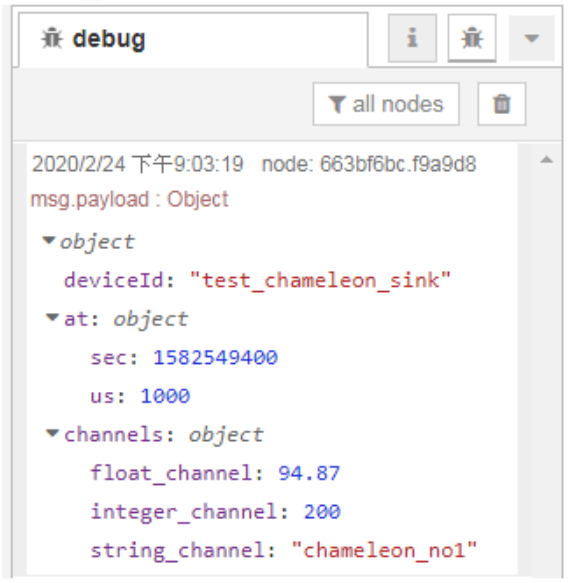
```
{
    "deviceId": "test_chameleon_sink",
    "at": {
        "sec": 1564042800,
        "us": 0
    },
    "channels": {
        "float_channel": 1.23,
        "integer_channel": 10,
        "string_channel": "chameleon_no1"
    }
}
```

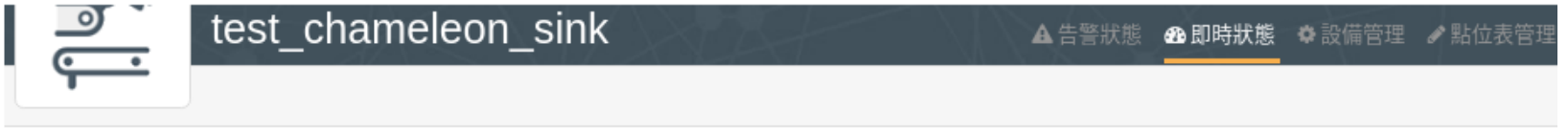
qi chameleon sink



- ref: example/chameleon-sink.json
- test_chameleon_sink_virtual.xlsx







#	點位 ID	點位名稱	數值類型	數值內容
👁	float_channel		number	94.87
👁	integer_channel		number	200
👁	string_channel		string	chameleon_no1

新增虛擬設備-選擇 VIRTUAL

新增虛擬設備-設定設備 ID

新增虛擬設備-上傳點位表

新增虛擬設備-完成

qi on match

- 資料 filter
- 需在 node 編輯頁面設定 Schema
- Schema 格式 (example: qi-on-match.schema)
  - $schema
  - type
  - required
  - properties

qi on match

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "type": "object",
    "required": [
        "deviceId",
        "at",
        "channels"
    ],
    "properties": {…
    }
}
```

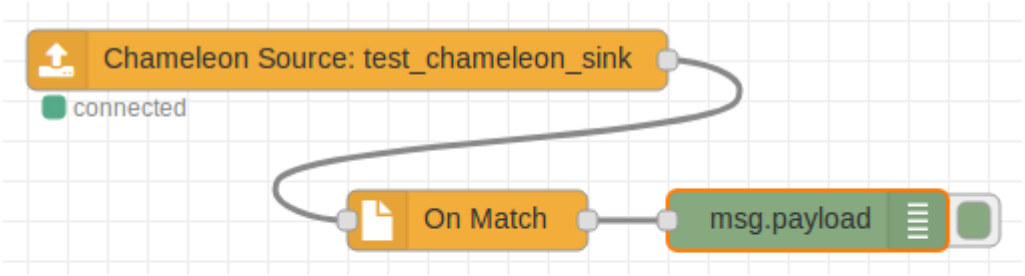
qi on match

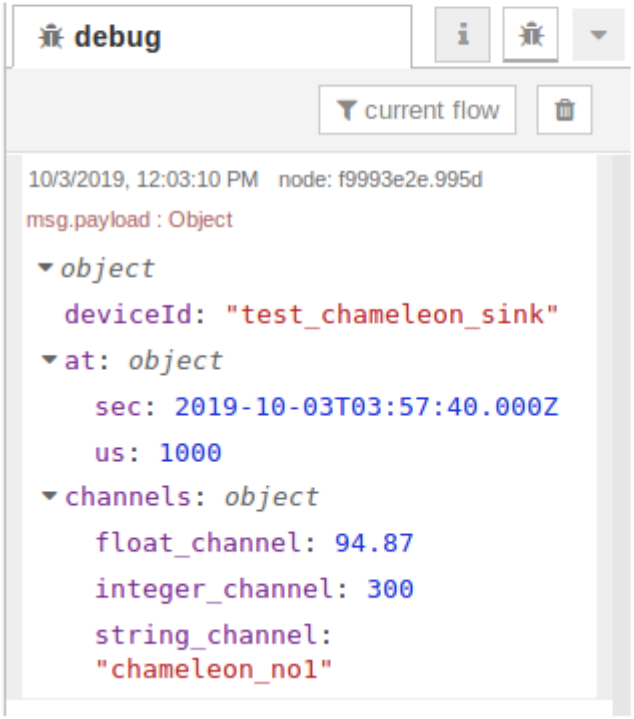
```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   "type": "object",
   "required": [ ⋯
   ],
   "properties": {
      "deviceId": {
         "type": "string"
      },
      "at": {
         "type": "object",
         "required": [
            "sec",
            "us"
         ],
         "properties": {
            "sec": {
               "type": "integer",
               "minimum": 0
            },
            "us": {
               "type": "integer",
               "minimum": 0,
               "maximum": 999999
            }
         }
      },
      "channels": { ⋯
      }
   }
}
```

qi on match

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   "type": "object",
   "required": [
   ],
   "properties": {
      "deviceId": {
         "type": "string"
      },
      "at": {
      },
      "channels": {
         "type": "object",
         "required": [
            "float_channel",
            "integer_channel",
            "string_channel"
         ],
         "properties": {
            "float_channel": {
               "type": "number"
            },
            "integer_channel": {
               "type": "number"
            },
            "string_channel": {
               "type": "string"
            }
         }
      }
   }
}
```

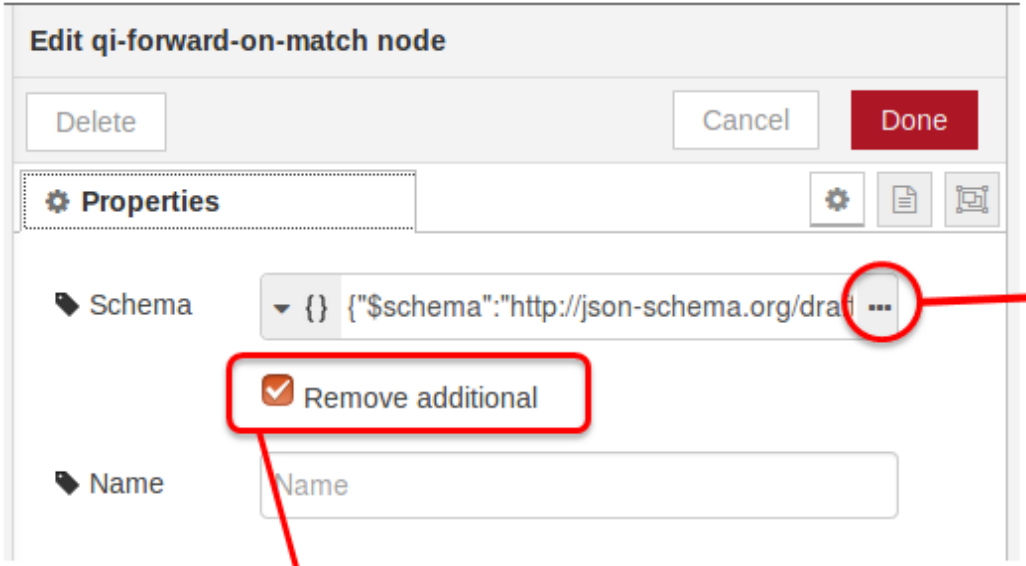
qi on match





- ref: example/on-match.json
- qi-on-match.schema

qi on match

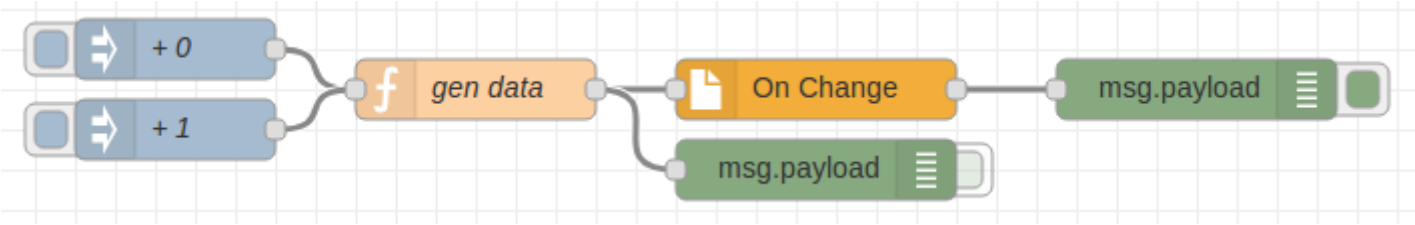


經過 on-match node 後

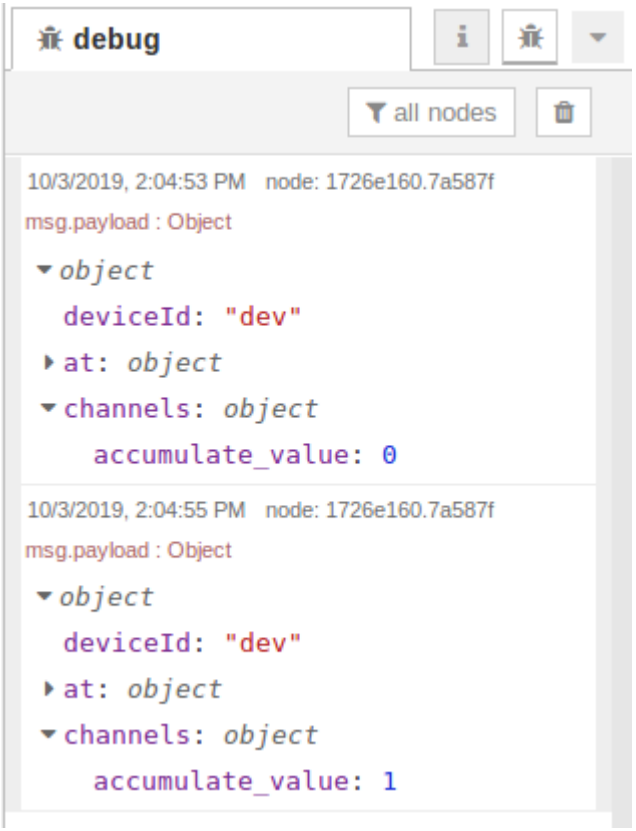
不在 properties::channels::required 的 channel 會被過濾掉

qi on change

- Channel 的 value值改變才會往下送



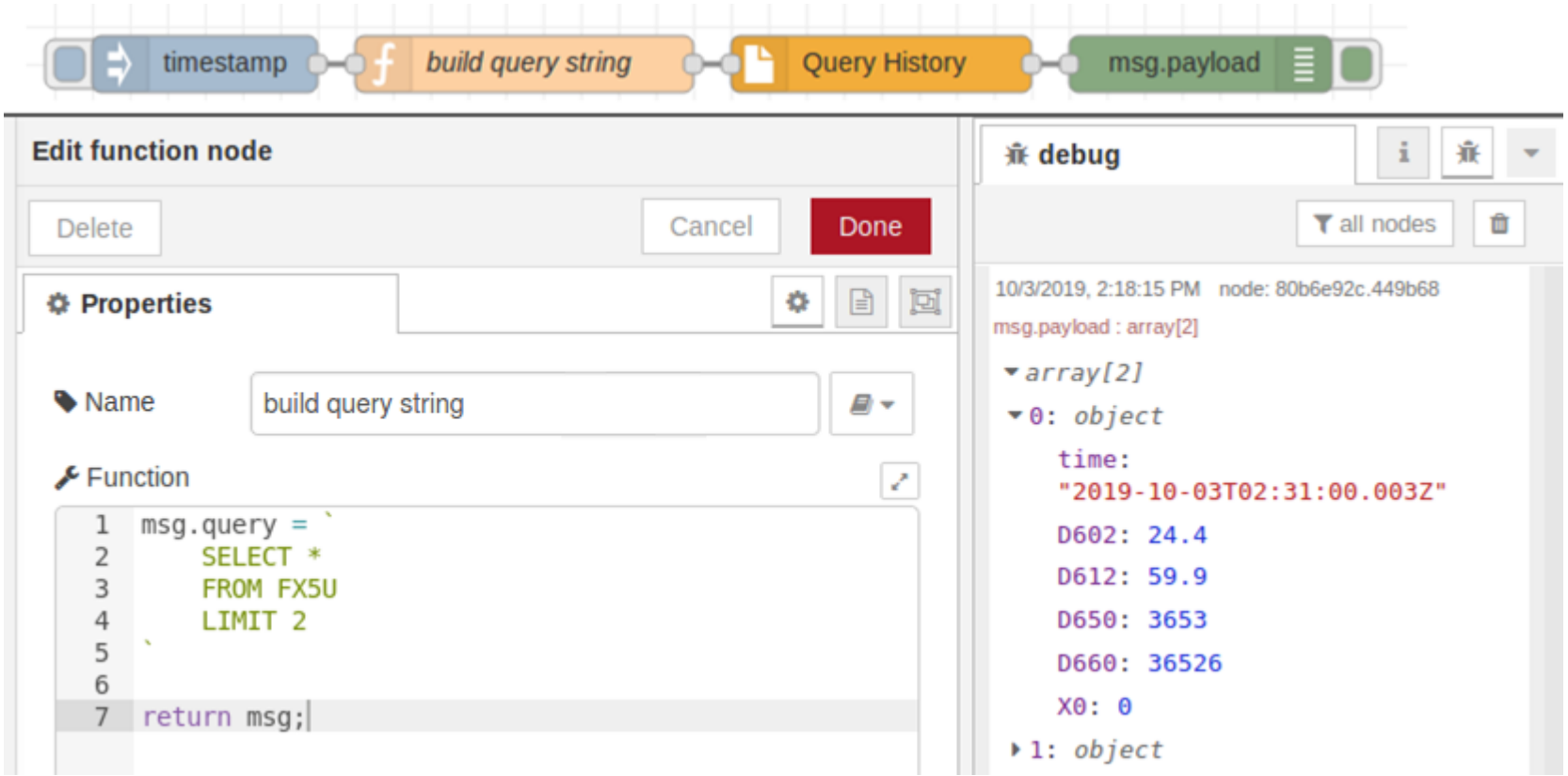
ref: example/on-change.json



qi query history

- 從變色龍的 database query 資料
- Query string (以 msg.query 為主)
  1. 編輯 node 內的 Query 欄位
  2. 或是在之前的 node 準備好 msg.query
- Query string 格式: InfluxQL
  - the InfluxDB SQL-like query language
  - https://docs.influxdata.com/influxdb/v1.7/query_language/

qi query history

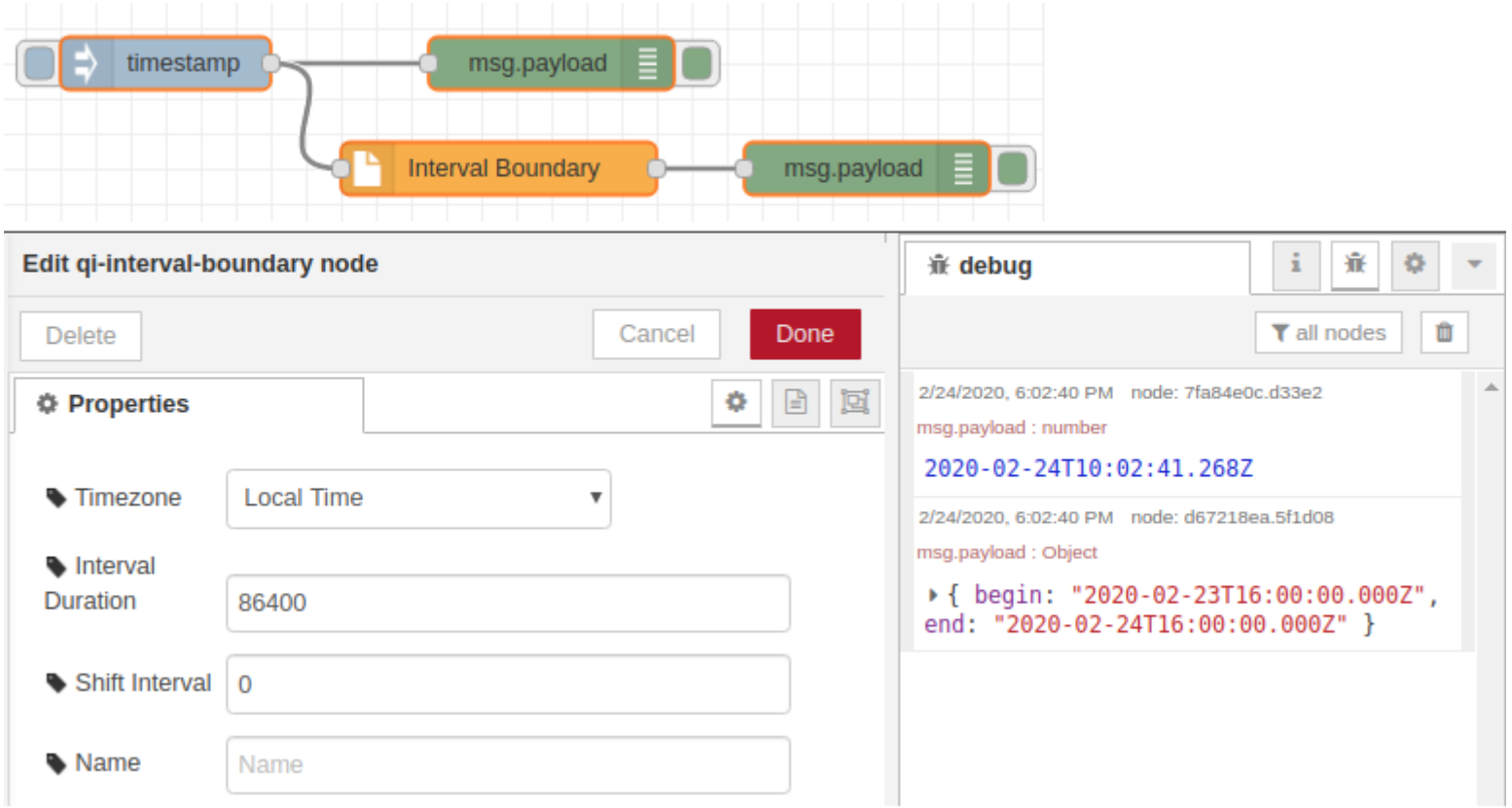


- ref: example/query-history.json

qi interval boundary

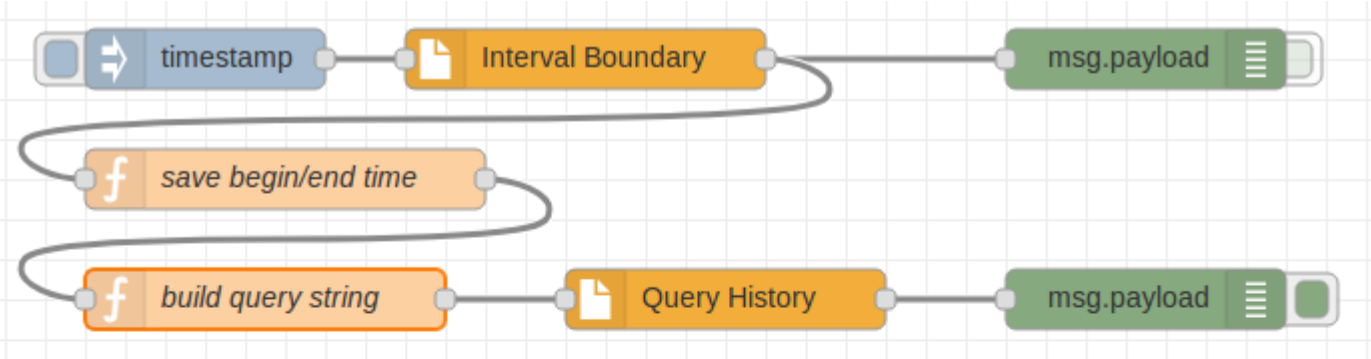
- 取得某段的開始及結束時間
- node 編輯頁面設定
  - Timezone: UTC/Local Time
  - Interval Duration(秒): default is 86400 (1 天)
  - Shift interval: 輸入整數, 以當下時間為基礎做 interval 的 shift, 若 interval duration 是 86400, 輸入 “-2” 會取得前天的開始/結束時間

qi interval boundary

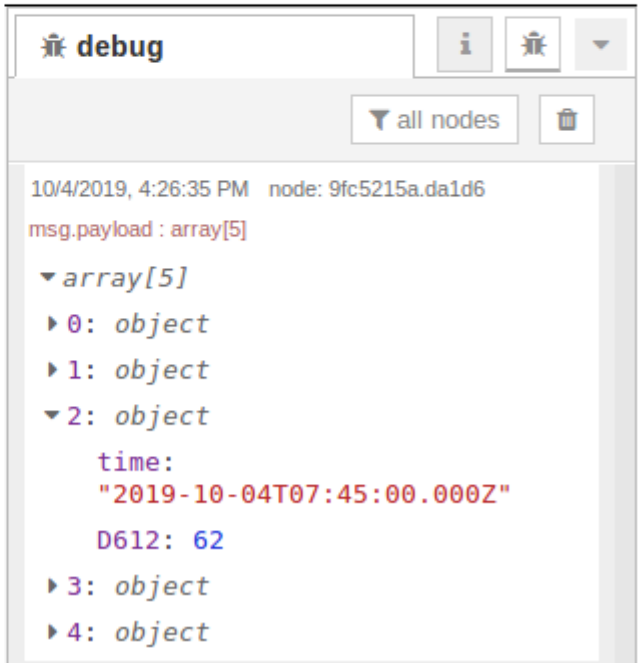


- ref: example/interval-boundary.json

interval boundary + query history example

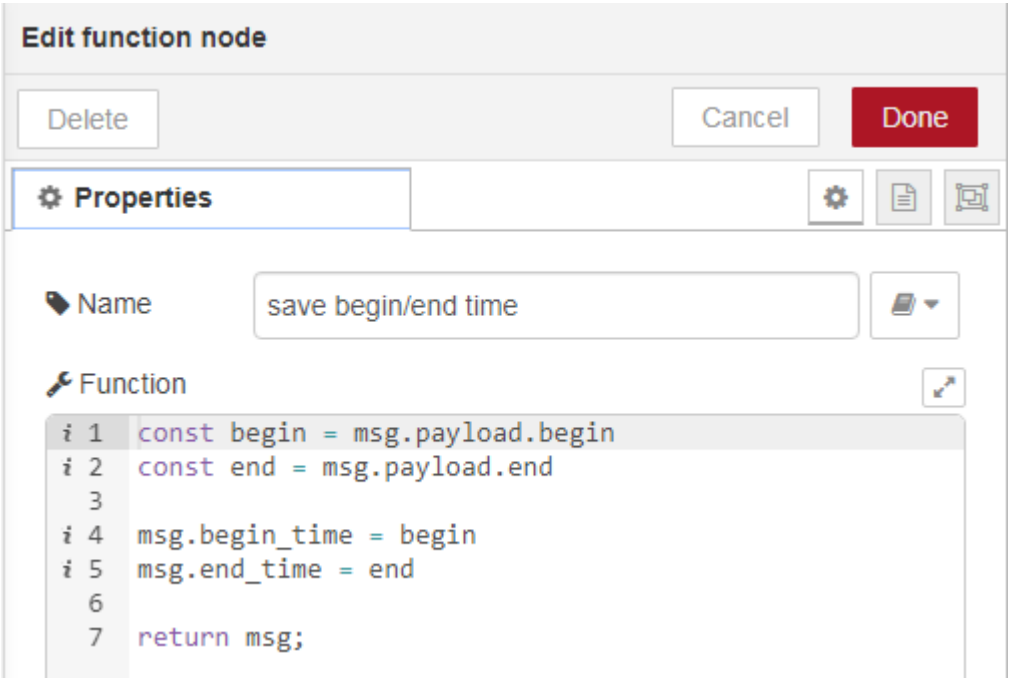


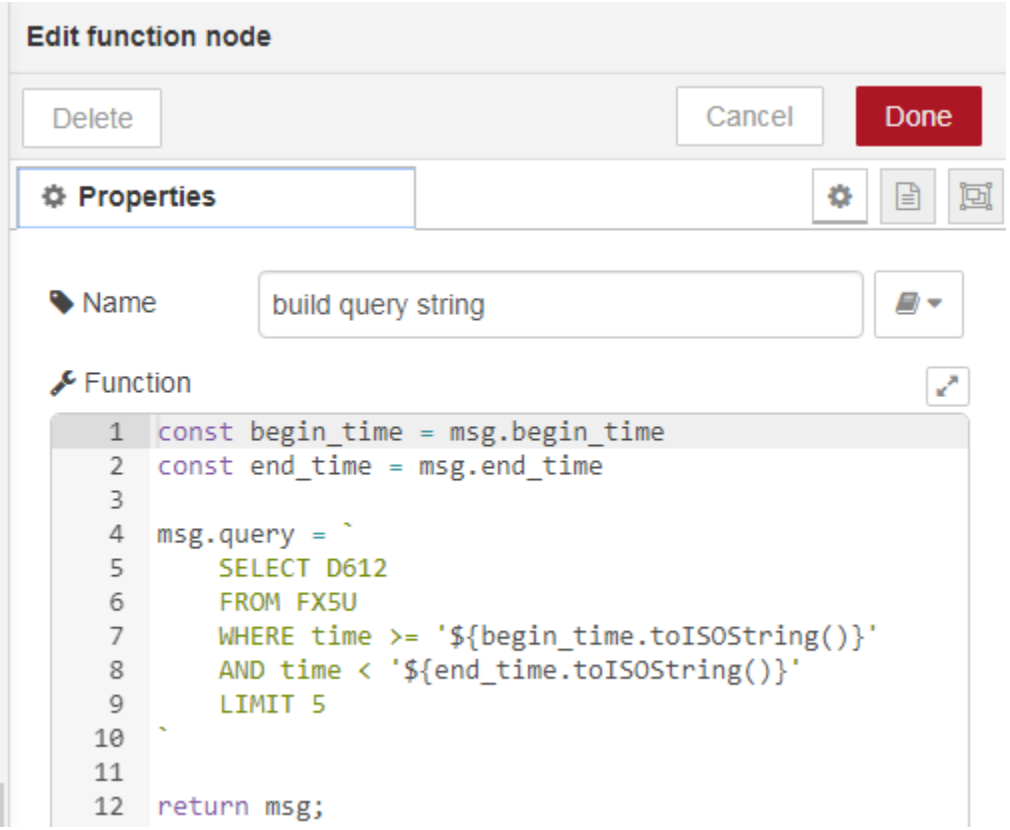
result of Interval Boundary



- ref: example/example_query-history.json

interval boundary + query history example





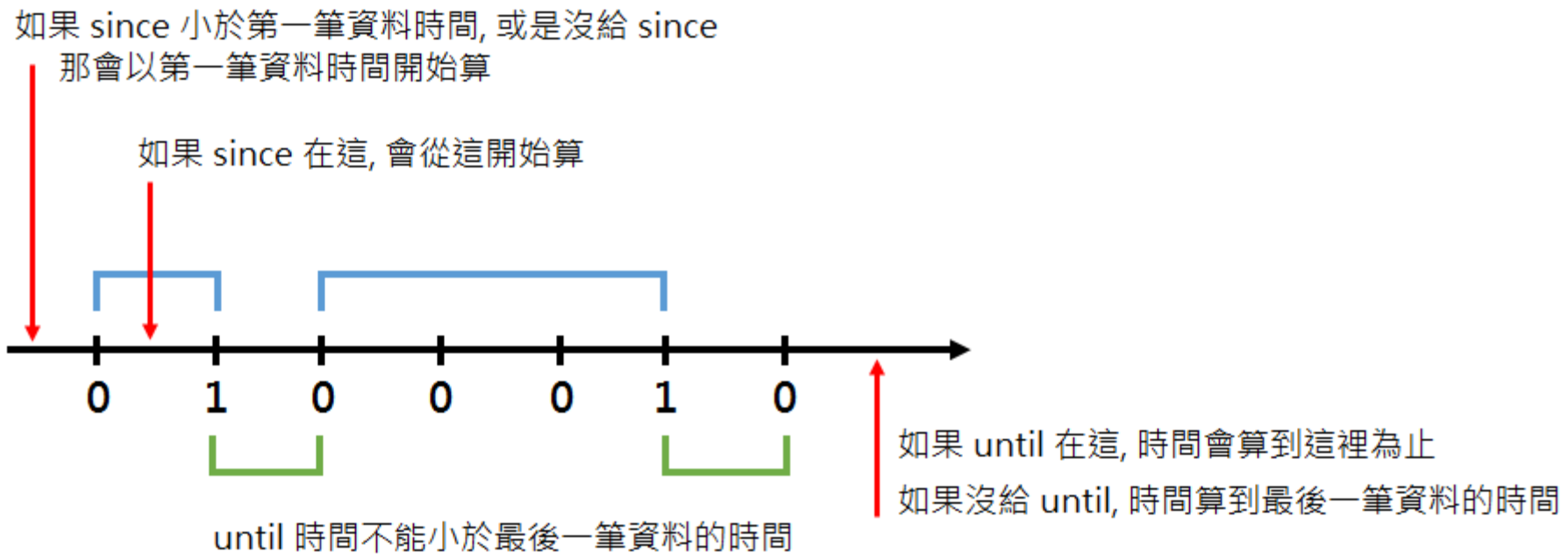
qi state duration

- 取得每個 state 的累積時間(單位: 毫秒 millisecond)
- 設定要算的 state name
- 資料輸入格式:
  - msg.payload: influxDB query result
  - msg.since: (optional) Date
  - msg.until: (optional) Date

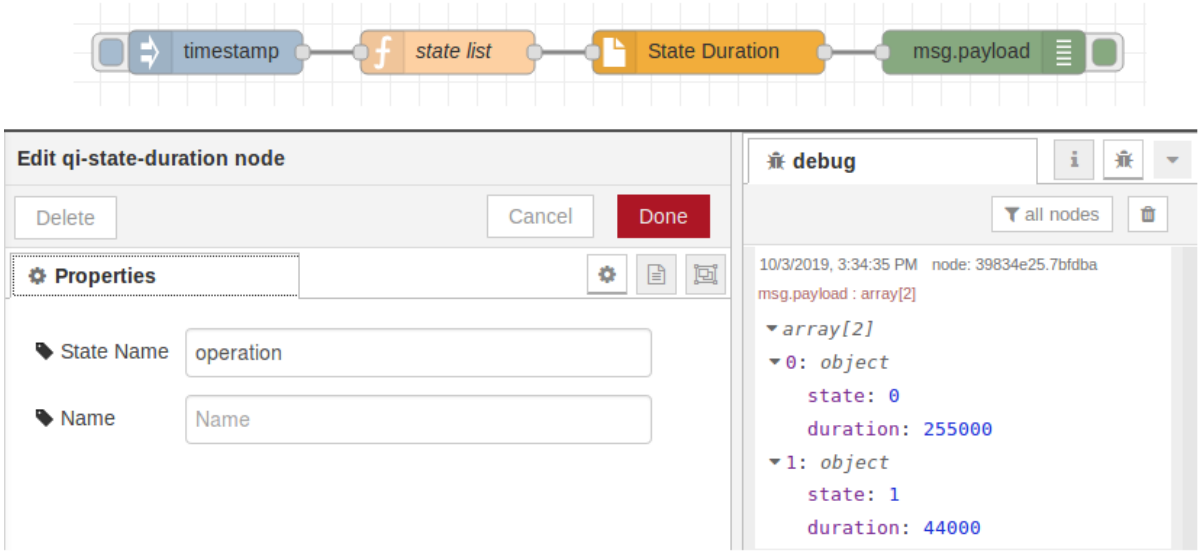
```
[
  {
    "time": <date>,
    "<channel>": <value>
  },
]
```

qi state duration

- msg.payload: 允許重覆連續的 state
- msg.since: 若沒給, 會以 payload 第一筆資料的時間開始算
- msg.until: 若沒給, 會以 payload 最後一筆資料的時間為止



qi state duration



Function

```
1  msg.payload = [
2      {
3          "time":new Date("2019-10-02T02:50:35.000Z"),
4          "operation":0
5      },  // 0:46 -> since: 2019-10-02T02:50:54.000Z -> 0:27
6      {
7          "time":new Date("2019-10-02T02:51:21.000Z"),
8          "operation":1
9      },  // 0:26
10     {
11         "time":new Date("2019-10-02T02:51:47.000Z"),
12         "operation":0
13     },  // 0:42
14     {
15         "time":new Date("2019-10-02T02:52:29.000Z"),
16         "operation":1
17     },  // 0:9
18     {
19         "time":new Date("2019-10-02T02:52:38.000Z"),
20         "operation":0
21     },  // 0:6
22     {
23         "time":new Date("2019-10-02T02:52:44.000Z"),
24         "operation":1
25     },  // 0:9
26     {
27         "time":new Date("2019-10-02T02:52:53.000Z"),
28         "operation":0
29     }  // 3:0
30 ]
31
32 msg.since = new Date("2019-10-02T02:50:54.000Z")
33 msg.until = new Date("2019-10-02T02:55:53.000Z")
34
35 return msg;
```

- ref: example/state-duration.json

iii PRIDE
資策會40週年

Thank you