MQTT + Node-RED + Heroku

廖冠雄

ghliaw@isu.edu.tw

大綱

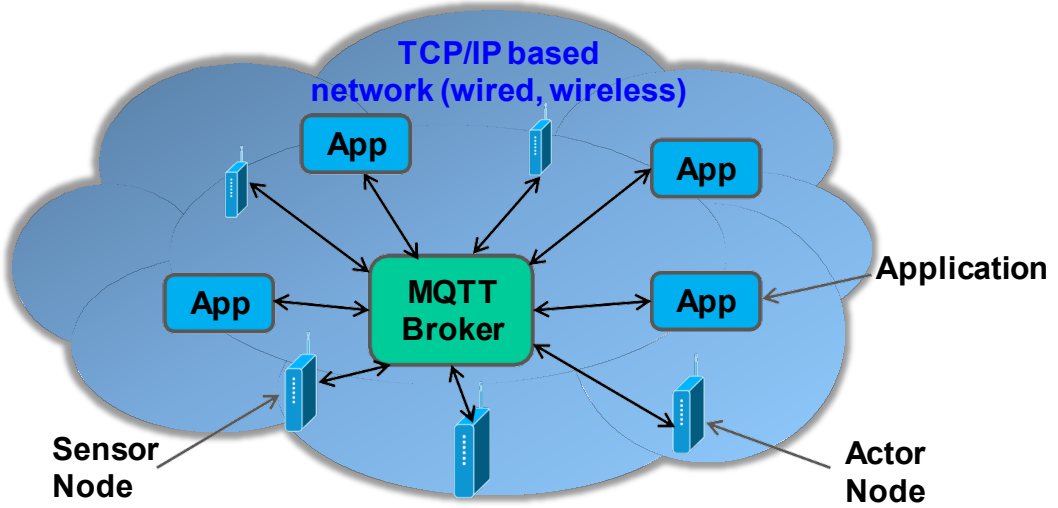
- **MQTT協定原理簡介**
  - 實驗#1：Arduino Basic MQTT Client
- **Node-RED開發平台簡介**
  - 實驗#2： Arduino UNO WiFi透過MQTT協定連結Node-RED物聯網後台
- **佈署Node-RED程式到Heroku平台**

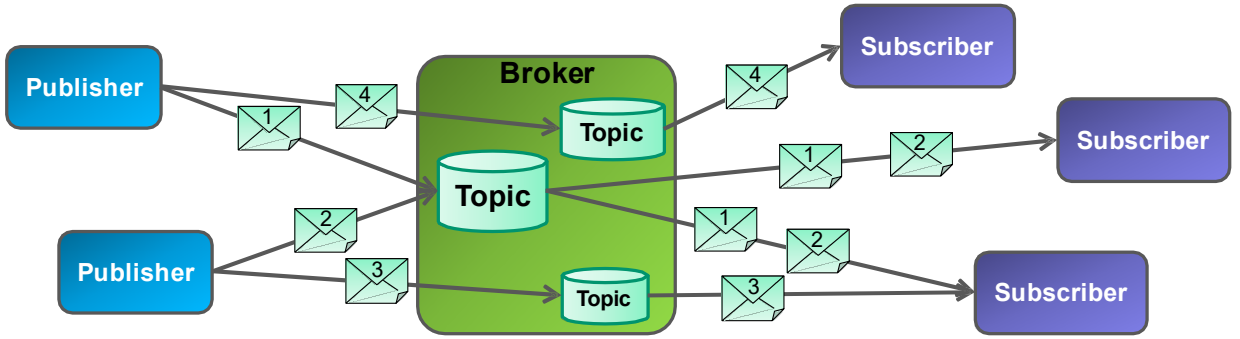
MQTT協定原理

輕量級的IoT資料交換協定

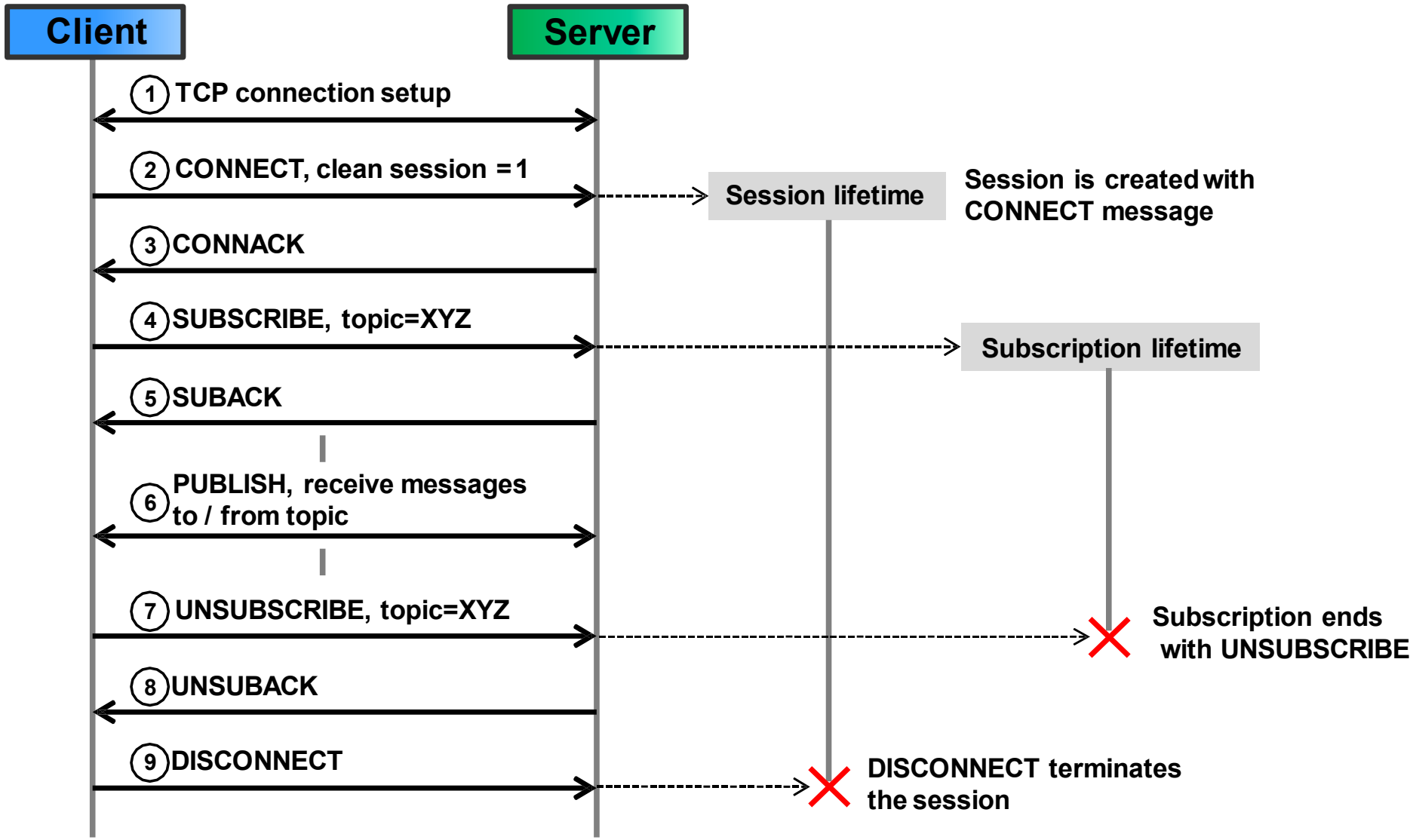
MQTT Publish/Subscribe Model

- **傳輸層協定採用TCP:**
  - 無加密：預設port 1883
  - TLS: 預設port 8883
- **基本上是Client/Server架構**
  - 以Broker為中心，所有資訊都透過Broker交換
- **角色**
  - **Publisher**： 發佈(*publish*)標示為某主題(topic)的資料到Broker
  - **Subscriber**：向Broker訂閱(*subscribe*)某主題的資料
  - **Broker**：接收來自Publisher的資料，並轉發佈此資料給有訂閱該主題的Subscriber



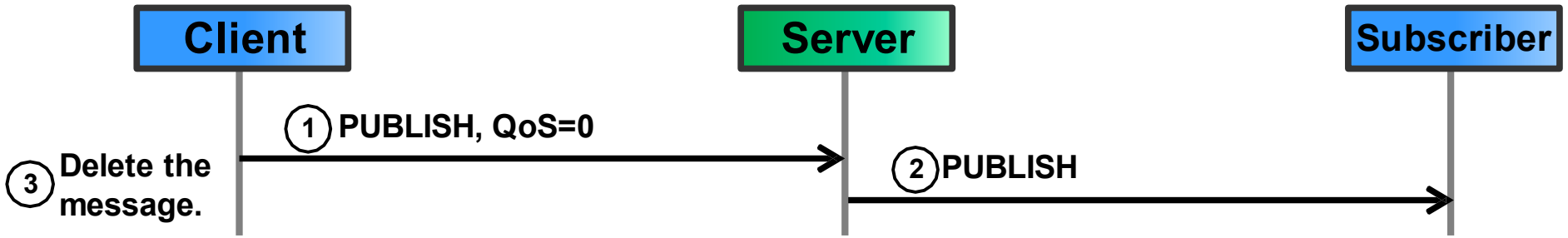


• MQTT訊息交換流程：

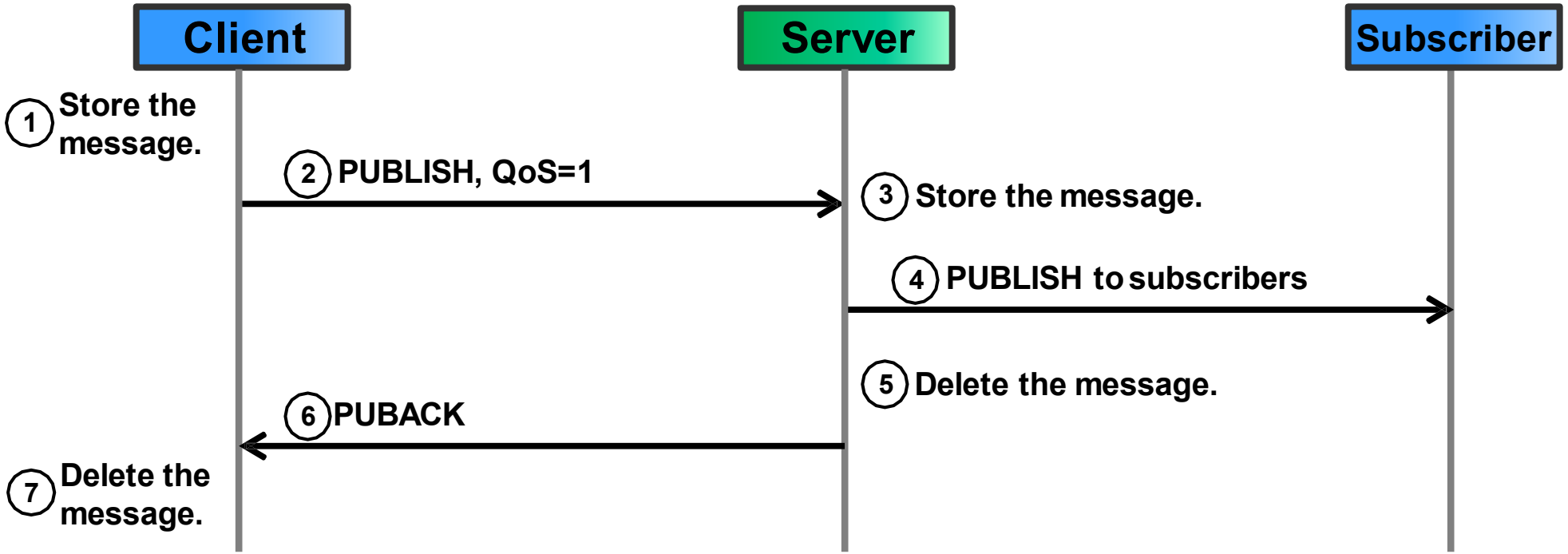

Client
Server
1 TCP connection setup
2 CONNECT, clean session =1
Session lifetime
Session is created with CONNECT message
3 CONNACK
4 SUBSCRIBE, topic=XYZ
Subscription lifetime
5 SUBACK
6 PUBLISH, receive messages to / from topic
7 UNSUBSCRIBE, topic=XYZ
Subscription ends with UNSUBSCRIBE
8 UNSUBACK
9 DISCONNECT
DISCONNECT terminates the session

QoS Levels for Publishing Messages

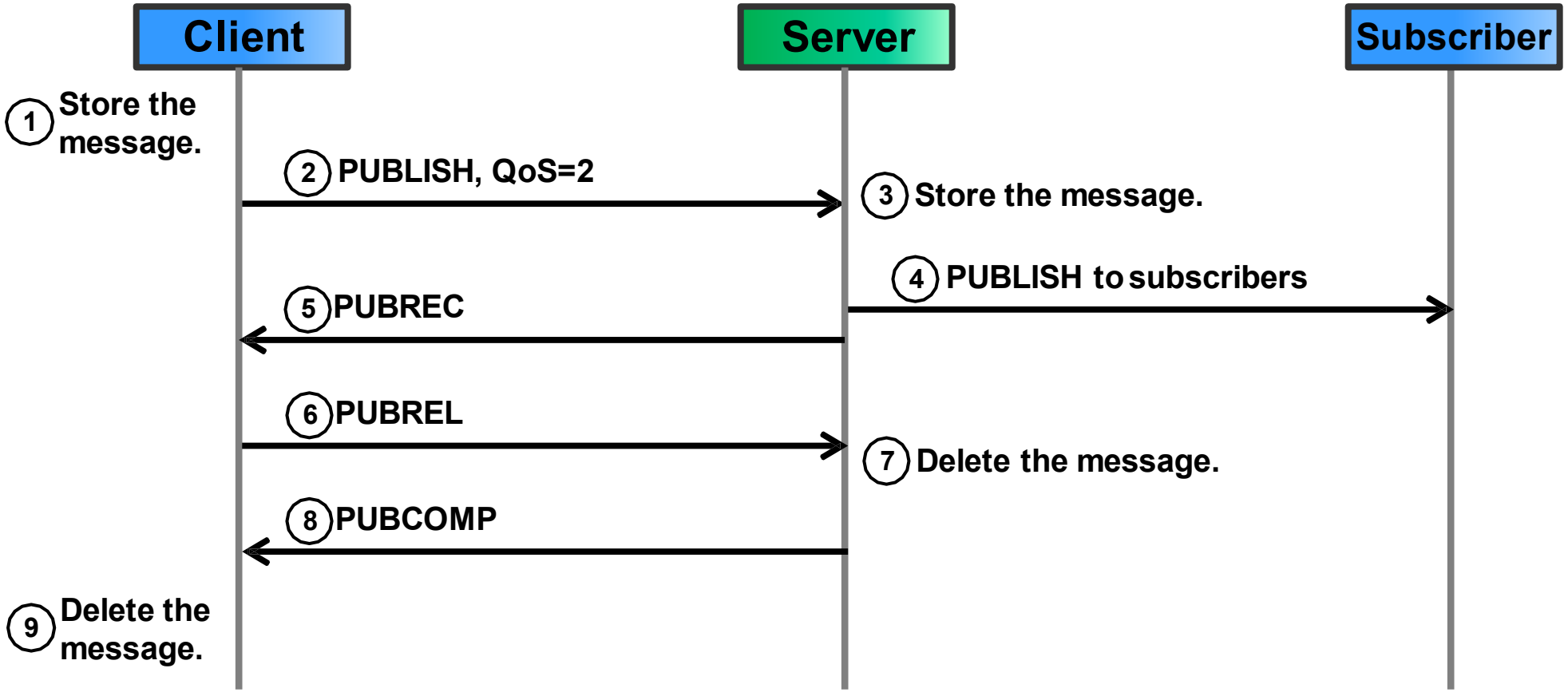
- **QoS Level 0 (*at-most-once*)**



- **QoS Levels 1 (*at-least-once*)**
  - If the client does not receive the PUBACK in time, it re-sends the message



• QoS Level 2 (*exactly-once*)



MATT Topic Format

- **topic是有階層的字串，以「/」作為階層的分隔符號**
  - 類似檔案路徑
    - 例：building/floor-1/sensors/temperature
    - 例：building/floor-1/sensors/humidity
- **Subscribe時可以用萬用字元**
  - 萬用字元「+」：
    - 代表單層的萬用字元
    - 例：building/+/sensors/temperature
      - 符合者：building/***floor-1***/sensors/temperature、 building/***floor-2***/sensors/temperature 、 building/***lobby***/sensors/temperature ...
  - 萬用字元「#」：
    - 代表多層的萬用字元
    - 例：building/floor-1/sensors/#
      - 符合者： building/floor-1/sensors/***number/temperature***、 building/floor-1/sensors/***number/humidity***、 building/floor-1/sensors/***state/light-switch***

MQTT Practices

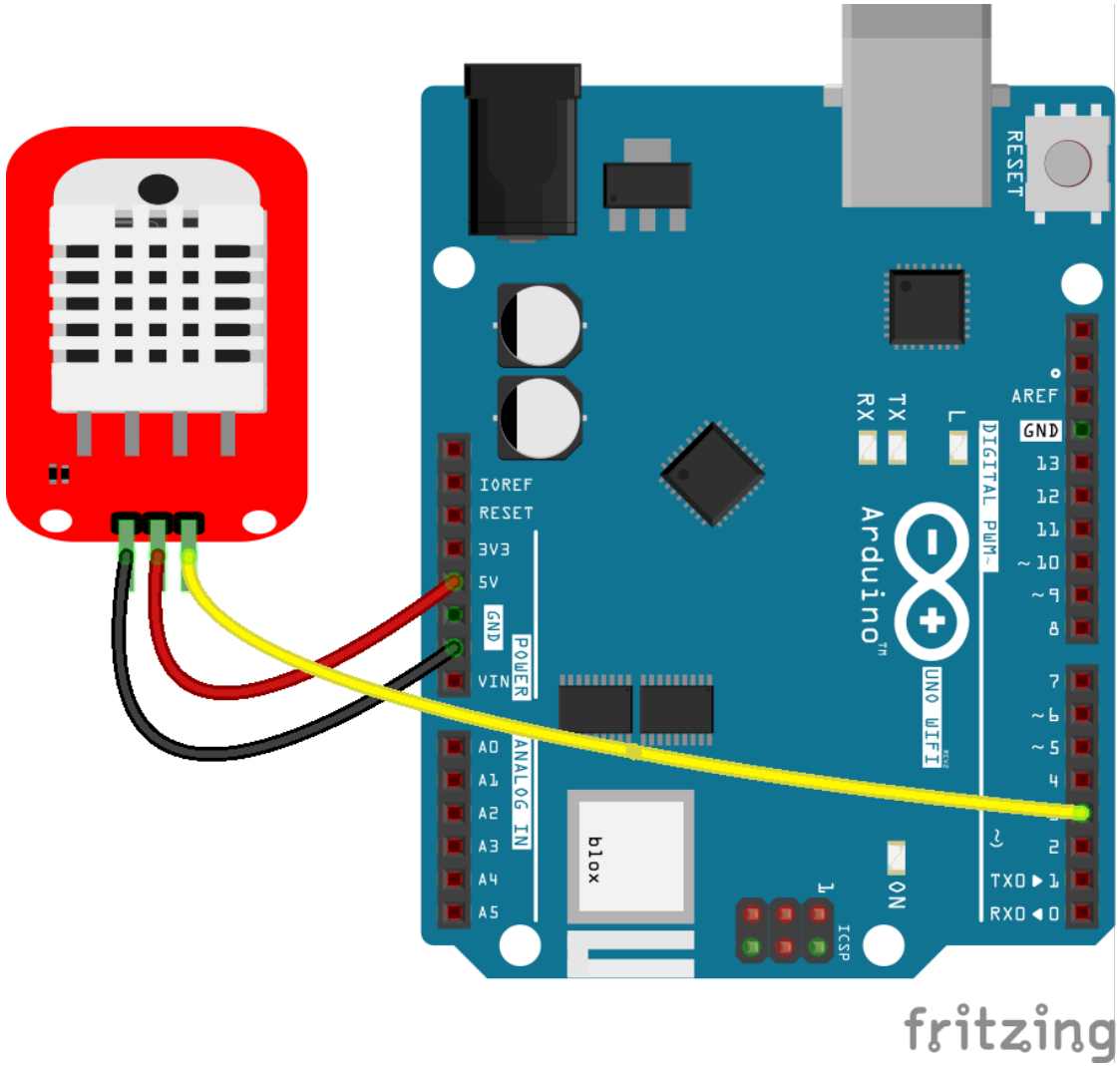
- **許多物聯網雲端平台都內建MQTT Broker功能**
  - Amazon AWS IoT, 中華電信IoT大平台等等
- **最有名的Open-source MQTT Broker (含client): Mosquitto**
  - https://mosquitto.org/
  - 已納入Eclipse基金會之下
- **Open-source MQTT Client函式庫**
  - Eclipse Paho
    - https://www.eclipse.org/paho/
    - 支援多種語言(C, C++, Java, Python, C#, Javascript, GoLang, Rust, ...)
  - Arduino最常用的MQTT Client函式庫: pubsub client
    - https://github.com/knolleary/pubsubclient
  - Aedes.js：以Node.js (javascript) 實現的MQTT Broker
    - https://github.com/moscajs/aedes
    - Node-RED也有人實現aedes的Node可供使用： https://flows.nodered.org/node/node-red-contrib-aedes

- **Public MQTT Brokers**
  - broker.hivemq.com (port: 1883)
  - test.mosquitto.org (port: 1883)
  - broker.emqx.io(port: 1883)
- **Free MQTT Client tools**
  - MQTT.fx (free only before v1.7.1)
    - https://mqttfx.jensd.de/index.php/download
  - MQTTLens (Chrome application)
  - MQTT Explorer
    - http://mqtt-explorer.com/

實驗#1：Arduino Basic MQTT Client

- **目標：**
  - Arduino WiFi連接溫溼度感測模組(DHT22)，每10秒透過MQTT發佈溫溼度資料，由電腦MQTT Client軟體(MQTT.fx)訂閱並接收
    - 使用public MQTT Broker: *test.modquitto.org*
    - 溫度topic: *xxxxxx/sensor/temperature*
    - 濕度topic: *xxxxxx/sensor/humidity*
    - 提示資訊topic: xxxxxx/info
  - 透過電腦MQTT Client軟體(MQTT.fx)，發佈訊息給Arduino WiFi，控制LED燈的開、關、與閃爍
    - 控制LDE的topic: xxxxxx/control/led
    - LED控制訊息：
      - "0"：關、"1"：開、"2"：閃爍

• 硬體材料與連接：



硬體清單：

- Arduino UNO WiFi Rev2
- DHT22溫溼度感測模組

DHT	Arduino WiFi
+ (VCC)	5V
OUT	D3
- (GND)	GND

- **軟體環境準備：**
  - Arduino IDE開發環境
    - 須將Arduino IDE更新到最新版(目前是1.8.15)
    - 到開發版管理員安裝/更新「Arduino megaAVR Boards」開發版套件到最新版
    - 到程式庫管理員安裝/更新「WiFiNINA」函式庫套件到最新版
      - **裝完之後要檢查板子的韌體是否夠新，若不夠新就要更新韌體**。
      - 作法網址： https://www.arduino.cc/en/Reference/WiFiNINA，看Firmware Update那一節。
    - 另外再安裝/更新以下函式庫套件到最新版
      - PubSubClient (MQTT Client函式庫)
      - DHT Sensor Library
  - MQTT.fx：電腦MQTT Client軟體
    - 以Java實作
    - 載點： http://www.jensd.de/apps/mqttfx/1.7.1/
    - 支援：windows 32/64-bit、MacOS、Linux (.rpm, .deb)

- **本實驗程式皆放在GitHub，網址如下：**
  - https://github.com/ghliaw/Heroku_Nodered_MQTT
- **Arduino程式內容重點概述(01_MQTT_DHT22)：**
  - 使用PubsubClient函式庫來實現MQTT協定
    - 必須先準備一個實體線路(WiFi或Ethernet)的Client物件 【行號34】
    - 宣告一個PubSubClient的物件(以實體線路的Client物件為參數)來操作MQTT的動作 【行號35】
    - 要準備一個callback()函數【行號63~88】，
      連線前先帶入PubSubClient的物件中 【行號117】，
      當有訂閱的訊息進來時會自動呼叫callback()，傳入進來的訊息
      - 本程式callback()函數主要是根據收到的payload內容("0", "1", or "2")變更LED的狀態變數，真正對LED的I/O控制則寫在loop()函數中。

- PubSubClient常用函數：
  - 連線到MQTT Broker：client.connect(client_id) 【行號95】
    - » MQTT Broker只會允許同一個client_id建立一個連線，因此若多人同時進行此實驗，每人的client_id必須設為不同 【行號25】
  - 訂閱某主題的訊息：client.subscribe(topic) 【行號100】
  - 發布某主題的訊息：client.publish(topic, payload) 【行號168, 170】
- PubSubClient API Reference網址： https://pubsubclient.knolleary.net/api

– 本實驗採用public MQTT Broker： test.mosquitto.org

- **實驗步驟：**

1. 根據硬體連線圖將DHT22模組連接在Arduino UNO WiFi板子上
2. 使用Arduino ID開啟程式「01_MQTT_DHT11.ino」，並做以下修改：

a. (行號12) `char ssid[] = "3715";`
➔ 改成現場上網用的WiFi網路名稱

b. (行號13) `char pass[] = "12345678";`
➔ 改成現場上網用的WiFi網路密碼

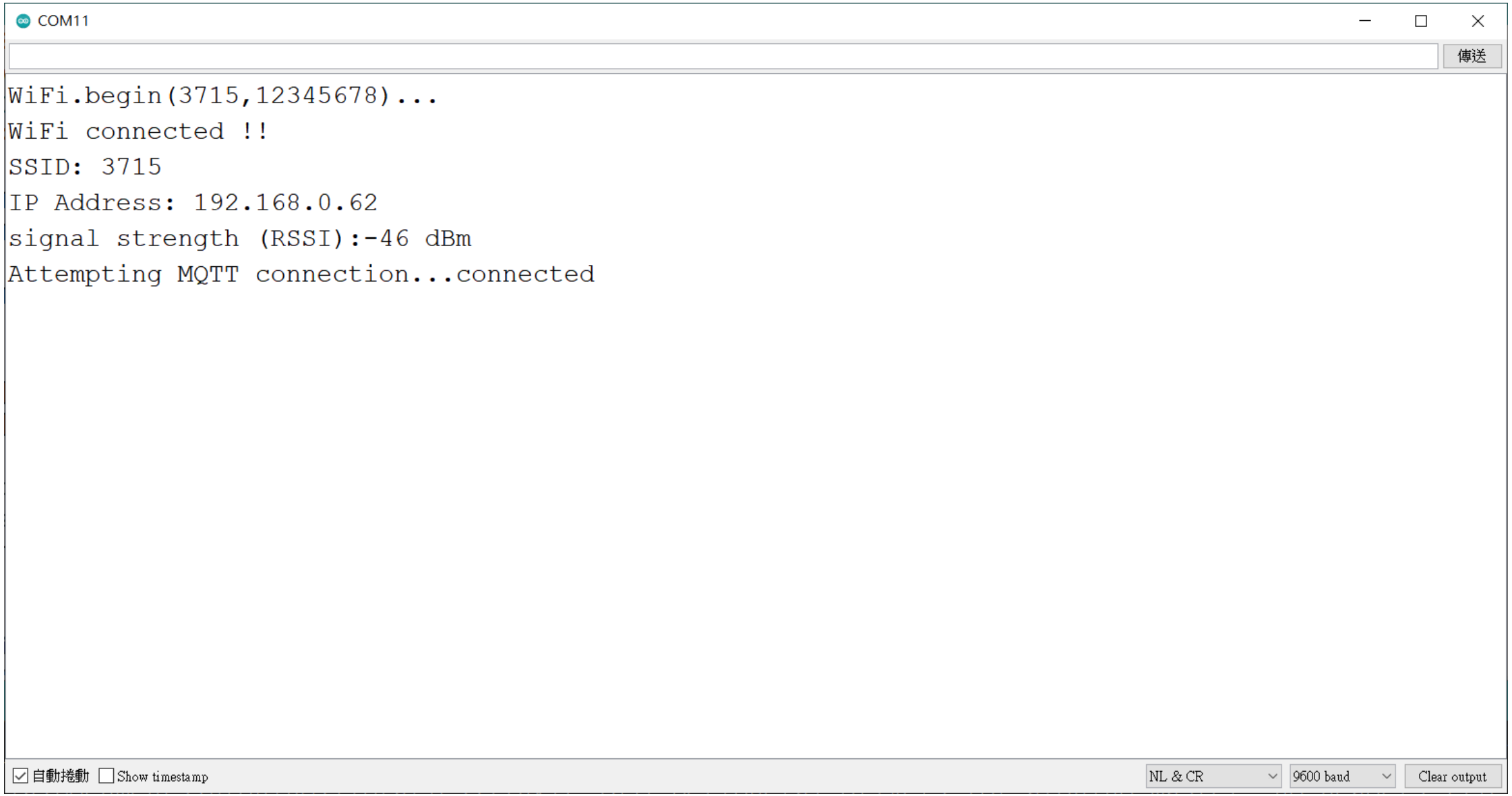
c. (行號25) `char client_id[] = "SensorNode_001";`
➔ 這是MQTT Client ID，必須每組都改成不相同

d. (行號28~31) 這幾行是發佈與訂閱用的topics，請將字串中的"ghliaw"改成代表你自己的字串，要跟其他組不同，因為這樣大家不會都用相同的topic發布資訊。

e. (行號47) `#define TEMP_PERIOD 10000`
➔ 這是溫溼度發布週期，單位為ms (預設為10秒)，請自行調整

f. (行號48) `#define LED_FLASH_PERIOD 200`
➔ 這是LED閃爍狀態下的明暗週期，單位為ms (預設為200 ms)，請自行調整

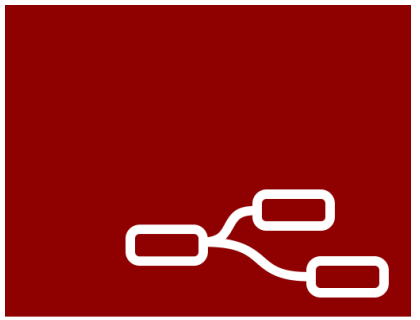
3. 電腦端執行MQTT.fx，連線到test.mosquitto.org，並subscribe以下topics： (xxxxxx是你自己修改的部分)
   a. 溫度topic: xxxxxx/sensor/temp
   b. 濕度topic: xxxxxx/sensor/hum
   c. 提示資訊topic: xxxxxx/info
4. 將Arduino程式上傳到板子上，成功上傳後開啟序列埠監控視窗(Baud rate: 9600)。板子的程式會先連線到WiFi網路並取得IP，然後會與MQTT Broker建立連線，若成功會印出如下頁圖的訊息。
5. 板子在成功建立MQTT Broker的連線後，會先用"xxxxxx/info"的topic發布"sensor node ready ..."的訊息。用MQTT.fx觀察是否有收到此訊息。
6. 接下來，觀察是否有週期性收到溫溼度的訊息，並與序列埠監控視窗印出來的值做比對。
7. 在MQTT.fx用"xxxxxx/control/led"為topic，分別發布"0"、"1"、"2"等訊息，觀察板子上的LED是否有被關閉、開啟、或閃爍。


COM11
傳送
WiFi.begin(3715,12345678)...
WiFi connected !!
SSID: 3715
IP Address: 192.168.0.62
signal strength (RSSI):-46 dBm
Attempting MQTT connection...connected
自動捲動
Show timestamp
NL & CR
9600 baud
Clear output

Node-RED開發平台簡介

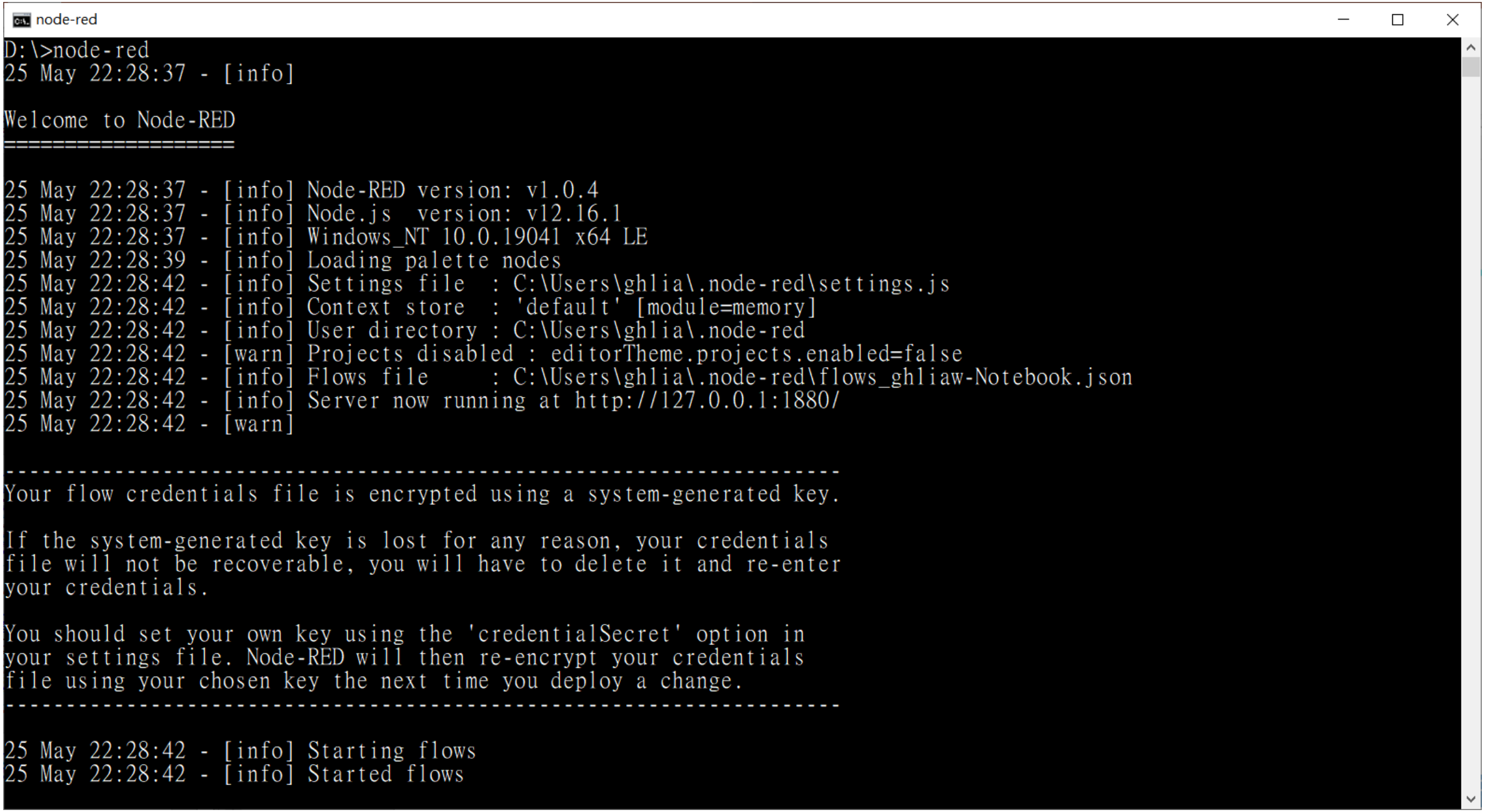
What is Node-RED

- **Node-RED 是 IBM 以 Node.js為基礎，開發出來的視覺化 IOT 開發工具**
  - –Node.js可以被視為「 Server版的javascript 」
- **純粹透過流程圖(flow diagram)的方式工作，所以不需要會 Node.js 也可以透過 Node-RED 完成許多後端才能做的事情。**
  - –開發環境不須安裝任何特殊軟體，只要有瀏覽器即可。
- **官方網站：https://nodered.org/**
- **可安裝在個人電腦或單板電腦 (Raspberry Pi、BeagleBone)**



在PC安裝Node-RED

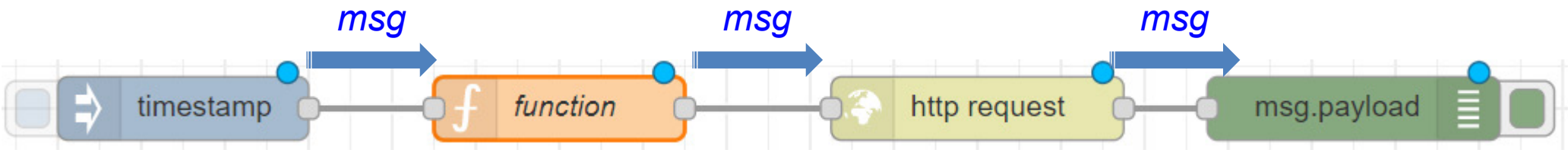
- **https://nodered.org/docs/getting-started/local**
- **在Windows下安裝：**
  - 參考網址：https://nodered.org/docs/getting-started/windows
  - 先安裝Node.js
  - 再用node.js的套件安裝工具npm來安裝node-red
    - 指令：npm install -g --unsafe-perm node-red
  - 執行Node-RED：
    - 指令：node-red
      - 會在當前使用者的家目錄中建立「.node-red」資料夾，所有的設定與程式都會存放在此
      - 也可開機就把Node-RED跑起來，參考：https://nodered.org/docs/getting-started/windows#running-on-windows
    - 打開瀏覽器，連線到http://127.0.0.1:1880/


node-red
D:\>node-red
25 May 22:28:37 - [info]
Welcome to Node-RED
==========
25 May 22:28:37 - [info] Node-RED version: v1.0.4
25 May 22:28:37 - [info] Node.js version: v12.16.1
25 May 22:28:37 - [info] Windows_NT 10.0.19041 x64 LE
25 May 22:28:39 - [info] Loading palette nodes
25 May 22:28:42 - [info] Settings file : C:\Users\ghlia\.node-red\settings.js
25 May 22:28:42 - [info] Context store : 'default' [module=memory]
25 May 22:28:42 - [info] User directory : C:\Users\ghlia\.node-red
25 May 22:28:42 - [warn] Projects disabled : editorTheme.projects.enabled=false
25 May 22:28:42 - [info] Flows file : C:\Users\ghlia\.node-red\flows_ghliaw-Notebook.json
25 May 22:28:42 - [info] Server now running at http://127.0.0.1:1880/
25 May 22:28:42 - [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 May 22:28:42 - [info] Starting flows
25 May 22:28:42 - [info] Started flows

Node-RED用法簡介

- 左邊的組件介面中有一些節點可以進行拖移的動作，每個節點都有各自的功能，可以拖移到中間的區域讓我們組成流程圖(Flow)。

• Flow的概念



– 一件工作可由串連的許多node來完成

– 每個node從前一個node取得資料，處理完之後再傳資料到下一個node

  • 所以一個node有輸入也有輸出

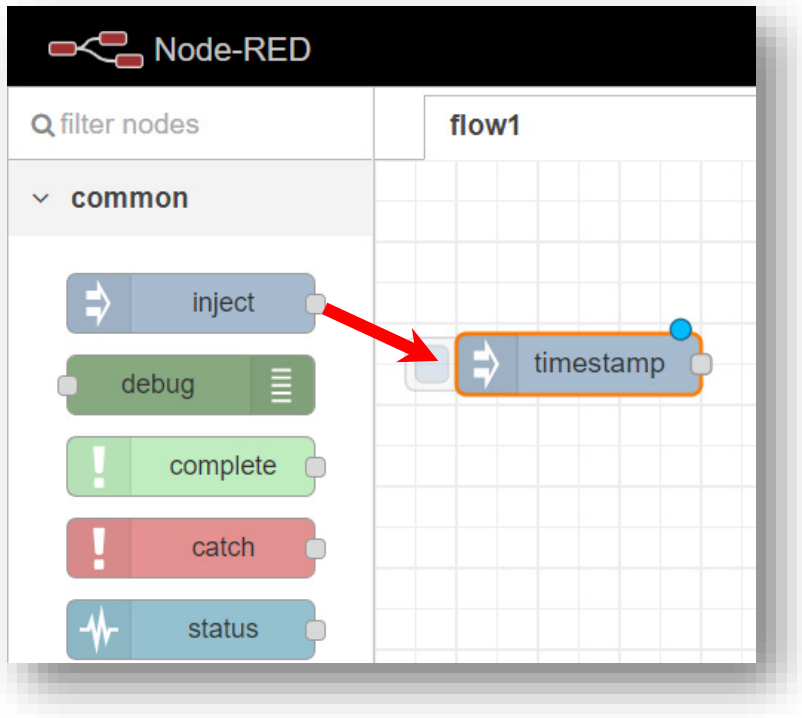
– node之間傳的是名稱為「msg」(訊息)的javascript物件

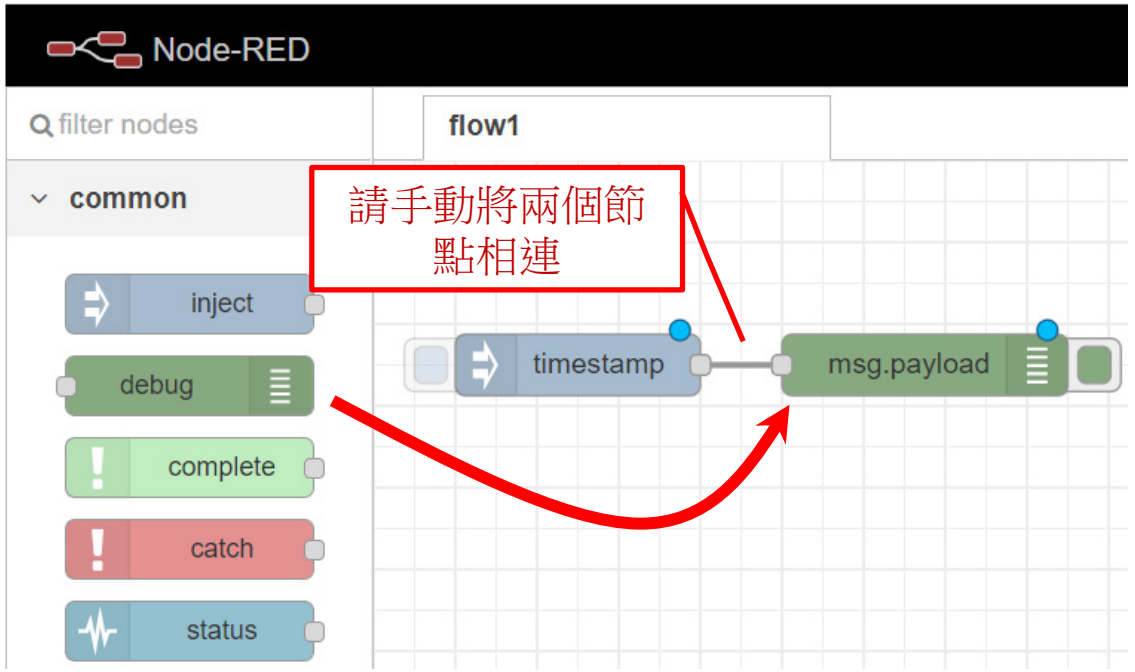
  • 預設內含兩個屬性：topic、payload
  • 通常要傳給下一個node的資料會放在payload屬性
  • 可以自由的在裡面加入任意欄位
  • 相當適用於在交換資訊時採用JSON格式的網路應用服務

- **Node元件的種類**
  - 預設就有多種node，包含流程控制、網路協定(HTTP, MQTT, TCP, UDP)、資料處理、檔案儲存等
  - 有線上函式庫提供開源的node元件擴充，在Node-RED開發環境中使用manage palette功能即可即時搜尋與安裝想要的node元件。

Node簡易範例：Hello world

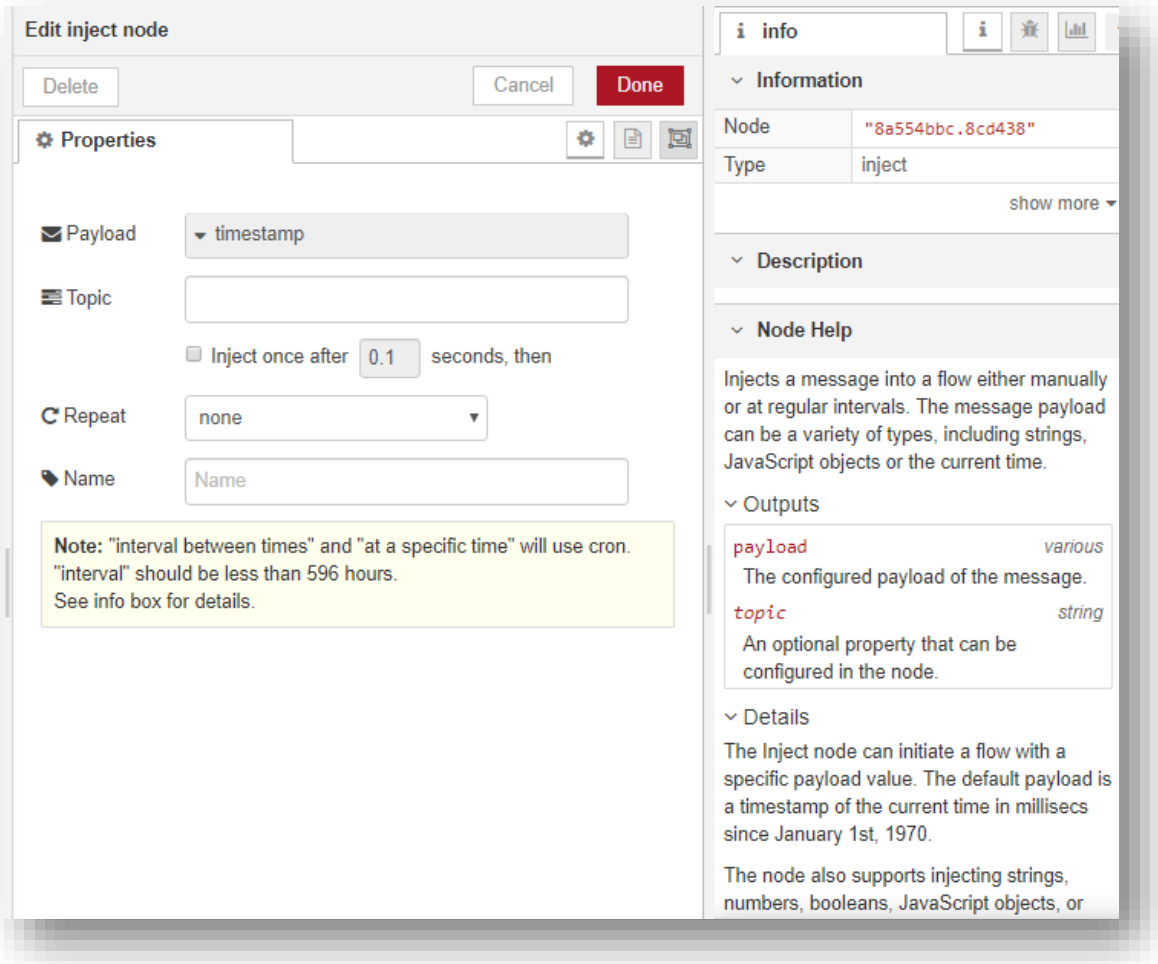
1. 將common種類的inject節點拖移到工作區
2. 將common種類的debug節點拖移到工作區，且把兩個節點相連

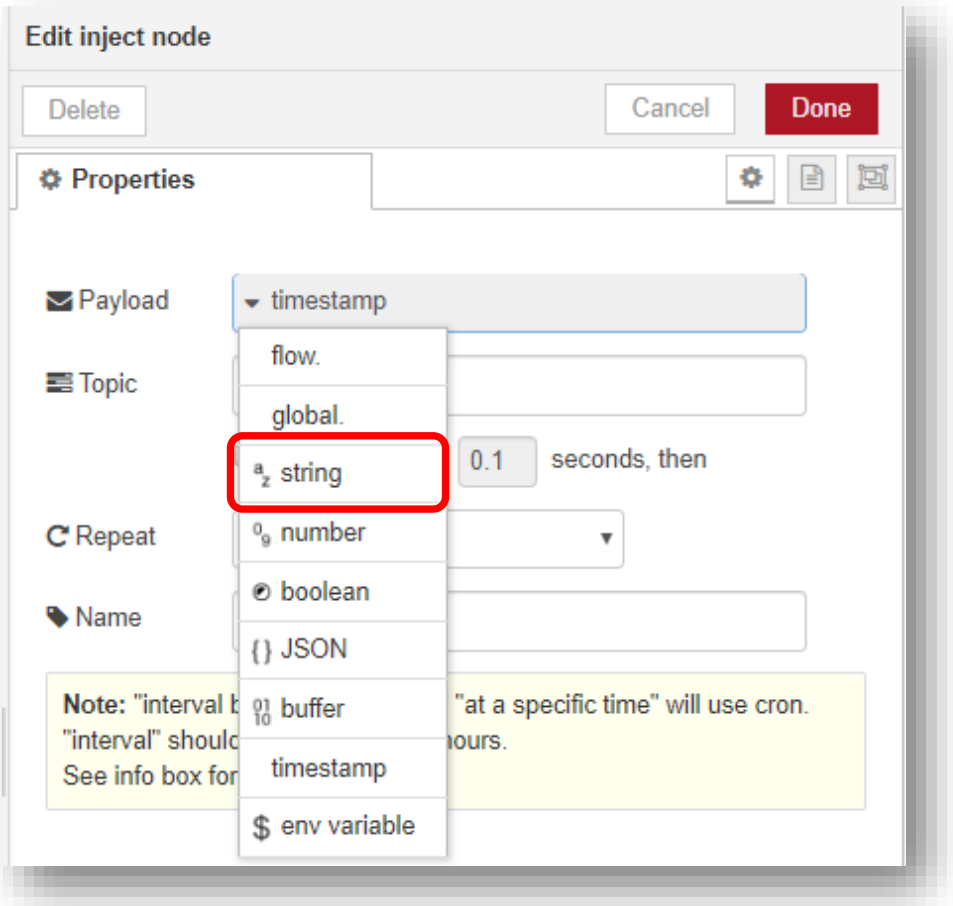




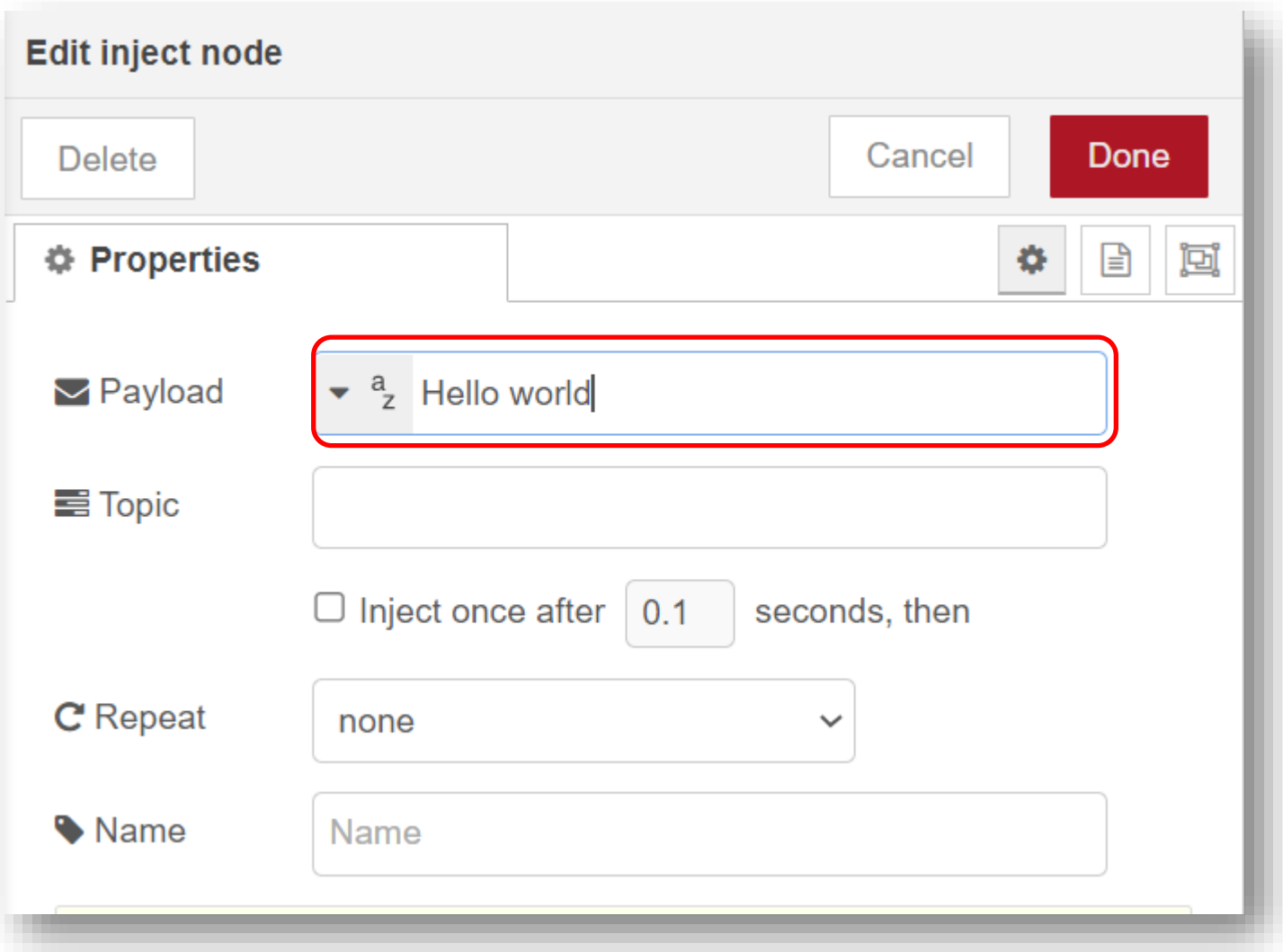
3. 在inject節點上點兩下左鍵就可以編輯，同時也會看到這個節點的詳細說明

4. 選擇Payload的型態， 這裡我們挑選string

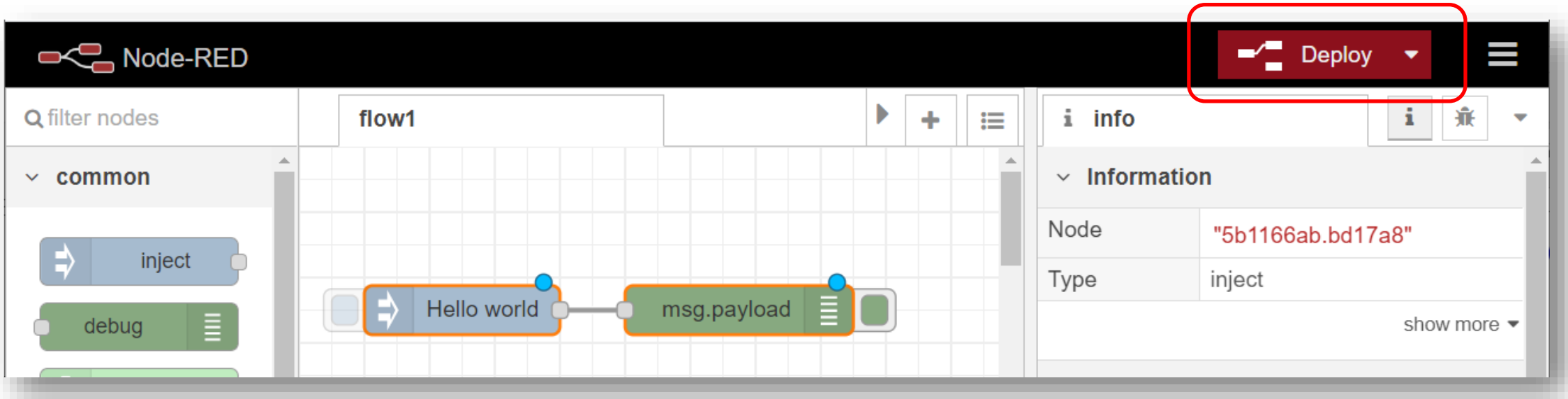




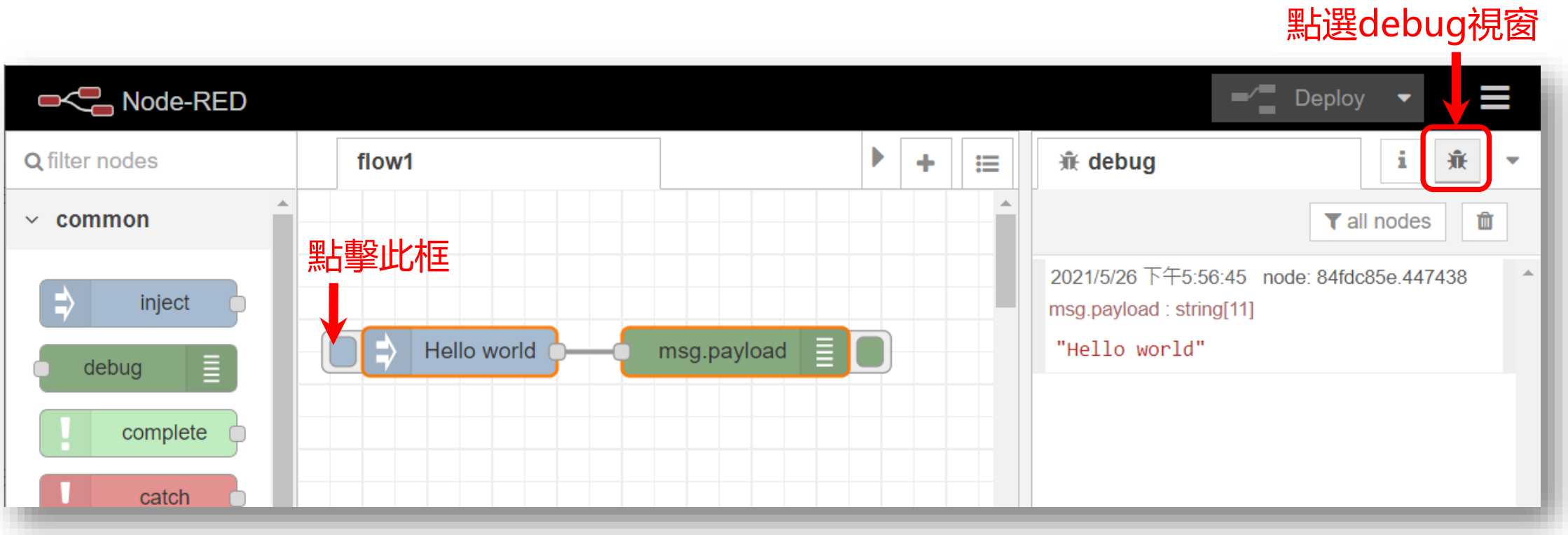
5. 在Payload後方的欄位填入"Hello world" ，然後按下Done按鈕結束編輯



6. 最後點選Node-RED視窗右上角的 deploy按鈕，就能佈署並開始運行。



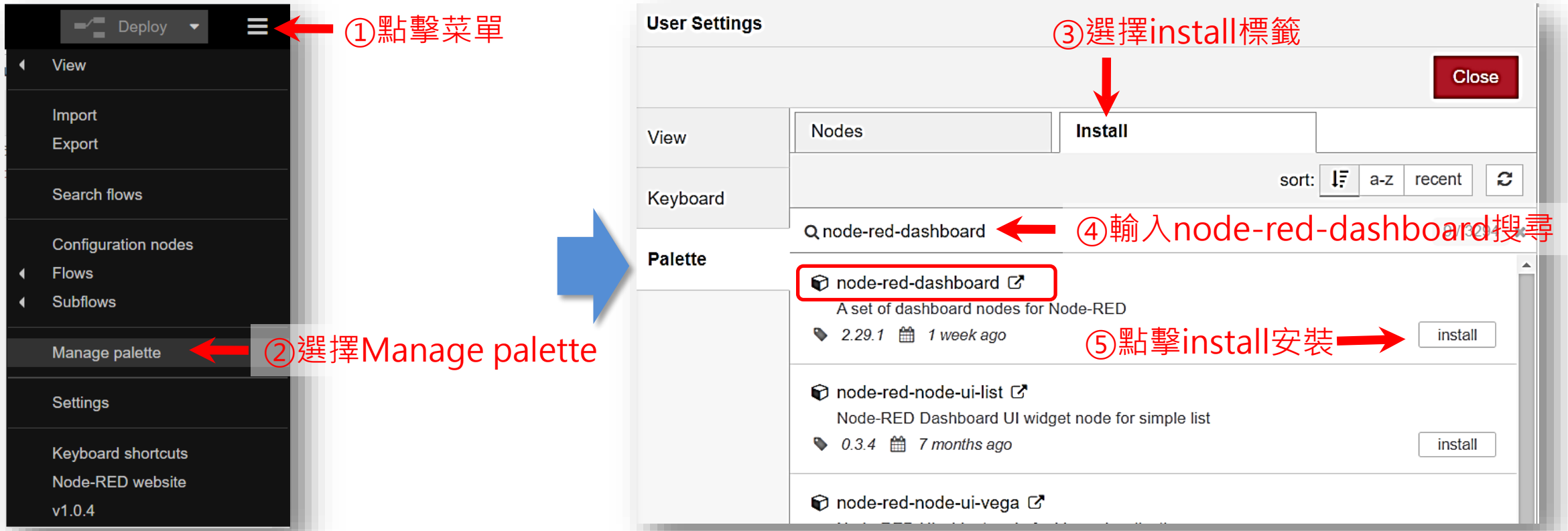
7. 點擊inject node左端的方框，就會對debug node送出訊息，其payload內容即為"Hello world"。若右方debug視窗有出現Hello world代表成功。



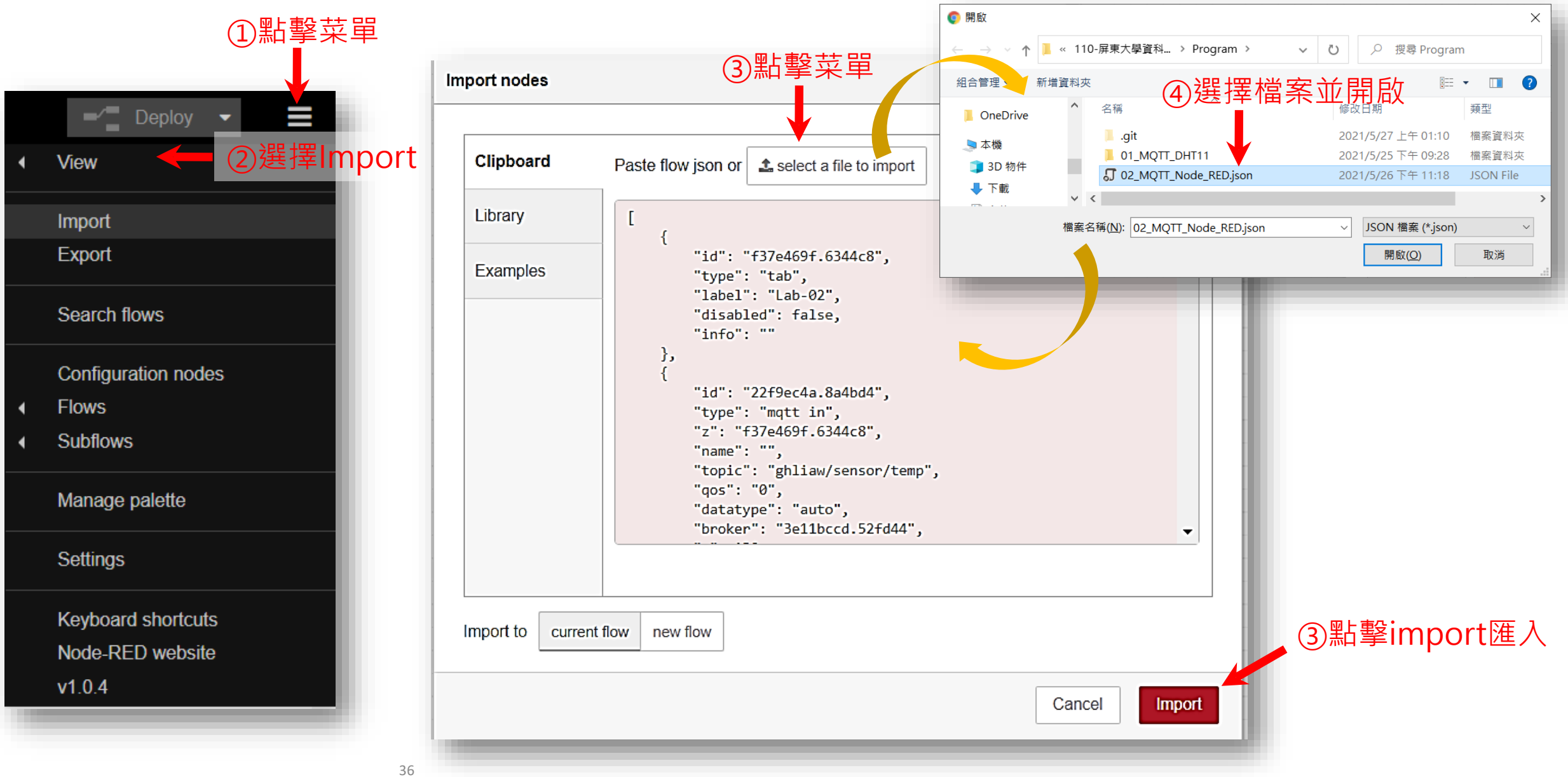
實驗#2：Node-RED MQTT後台

- **目標：**
  - 撰寫Node-RED程式以及儀表板(dashboard)頁面，使其可透過MQTT收到實驗#1的Arduino WiFi所發布的溫溼度資料，並將結果即時視覺化顯示在儀表板頁面上。
  - 同時，使用者也可透過儀表板頁面上的開關圖示，遠端控制板子內建LED的明滅。

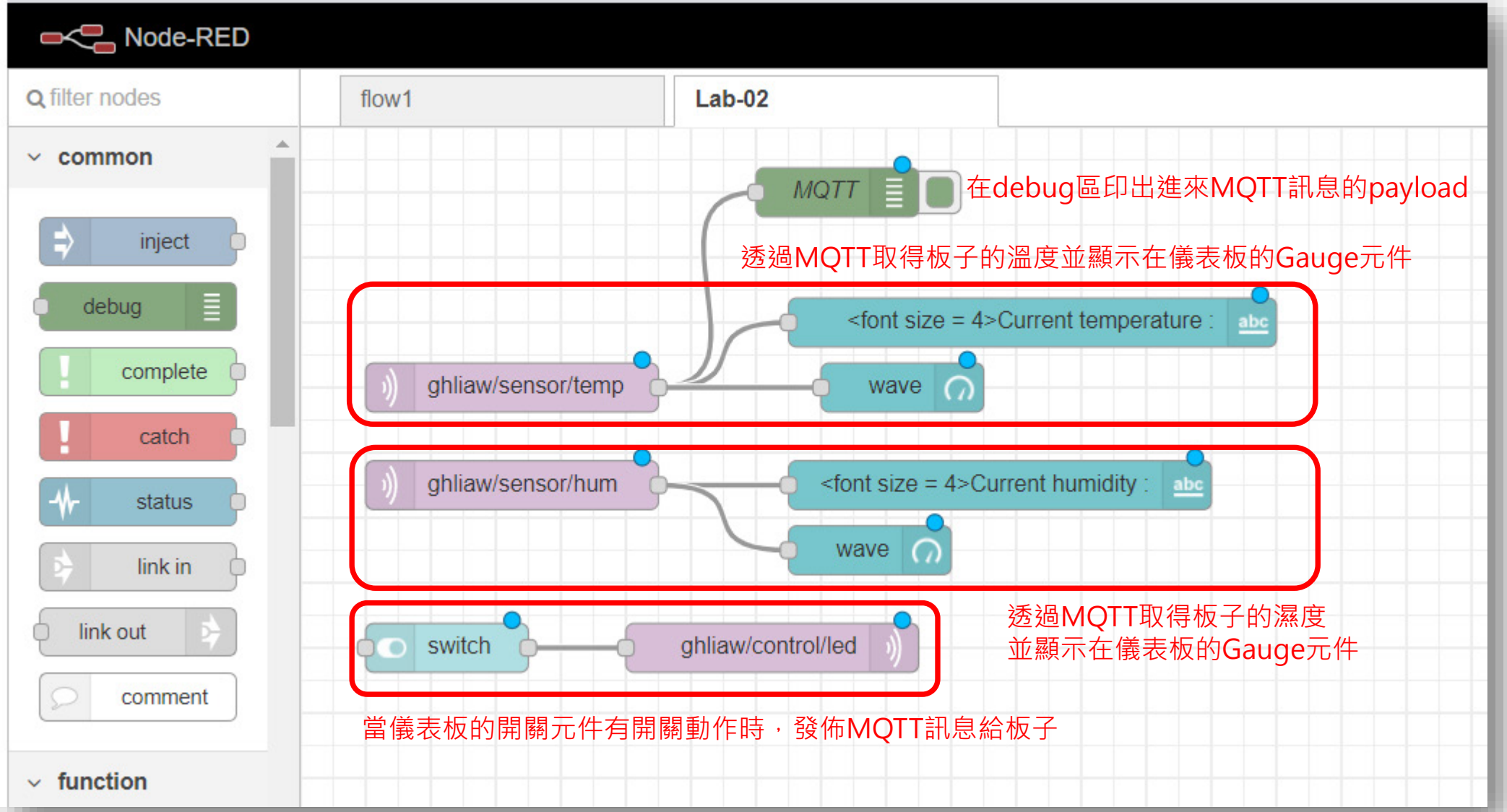
- **實驗步驟：**
  1. 安裝並啟動Node-RED開發環境
  2. 開啟web瀏覽器連線到http://127.0.0.1:1880，進入Node-RED開發環境
  3. 安裝node-red-dashboard套件



4. 匯入本實驗事先寫好的流程「02_MQTT_Node_RED.json」

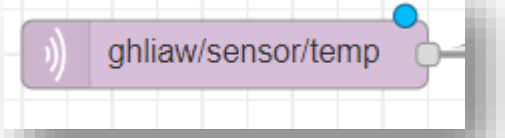


5. 匯入後會新增「lab-02」的流程，如下圖：

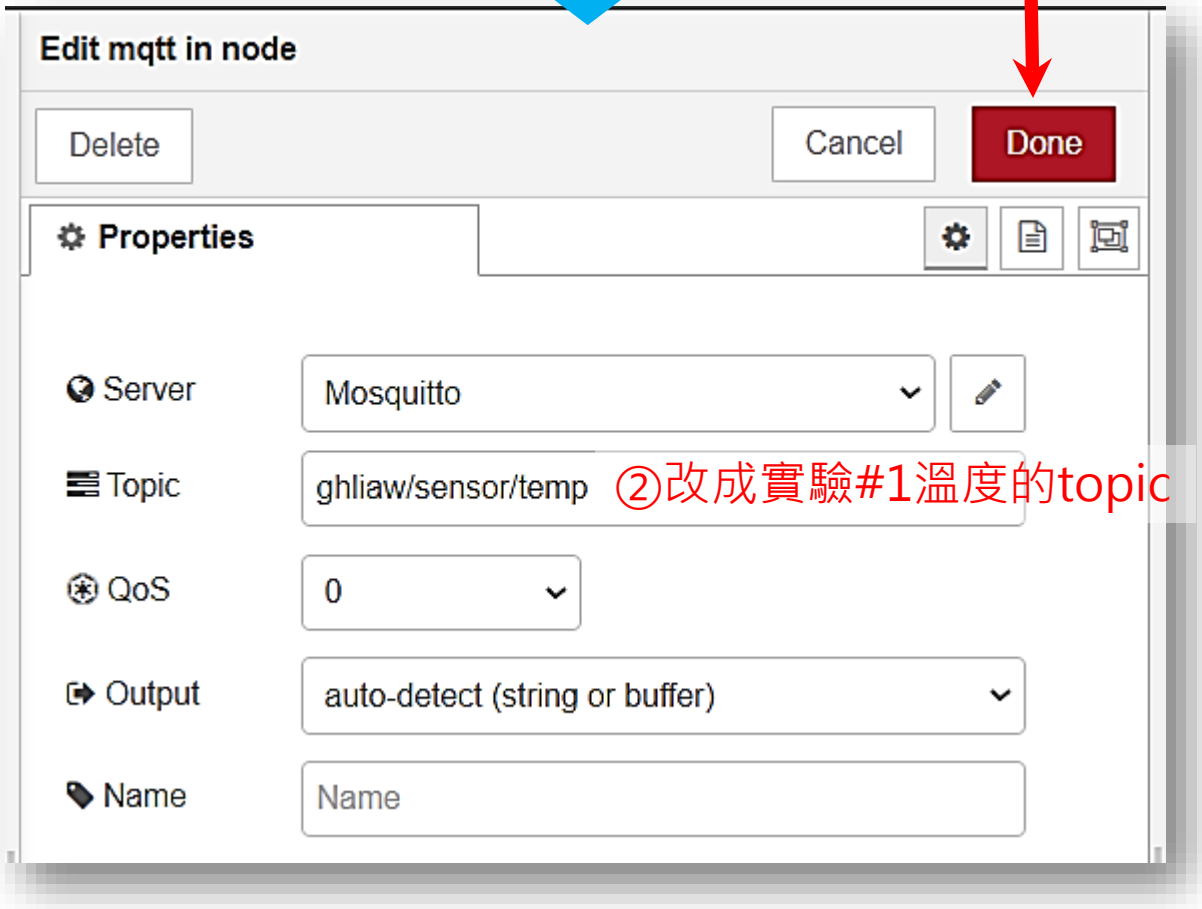


6. 修改MQTT nodes，使其訂閱或發佈的topics與實驗#1的設定一致：

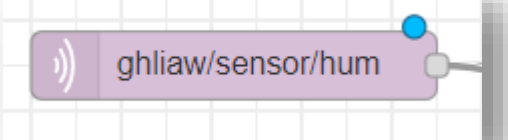
①雙擊取得溫度的MQTT node



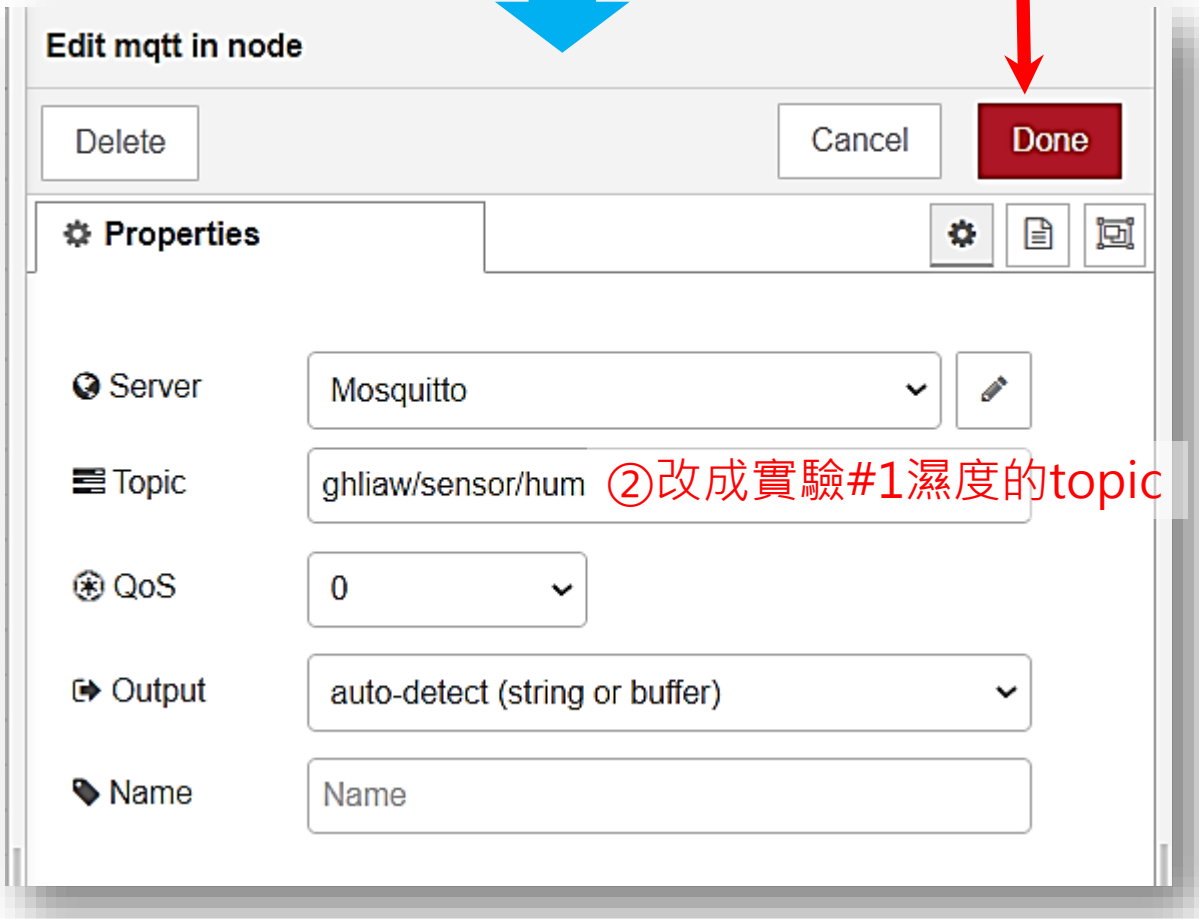
③按下Done完成



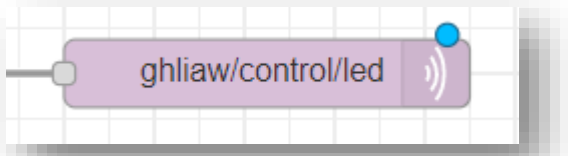
①雙擊取得濕度的MQTT node



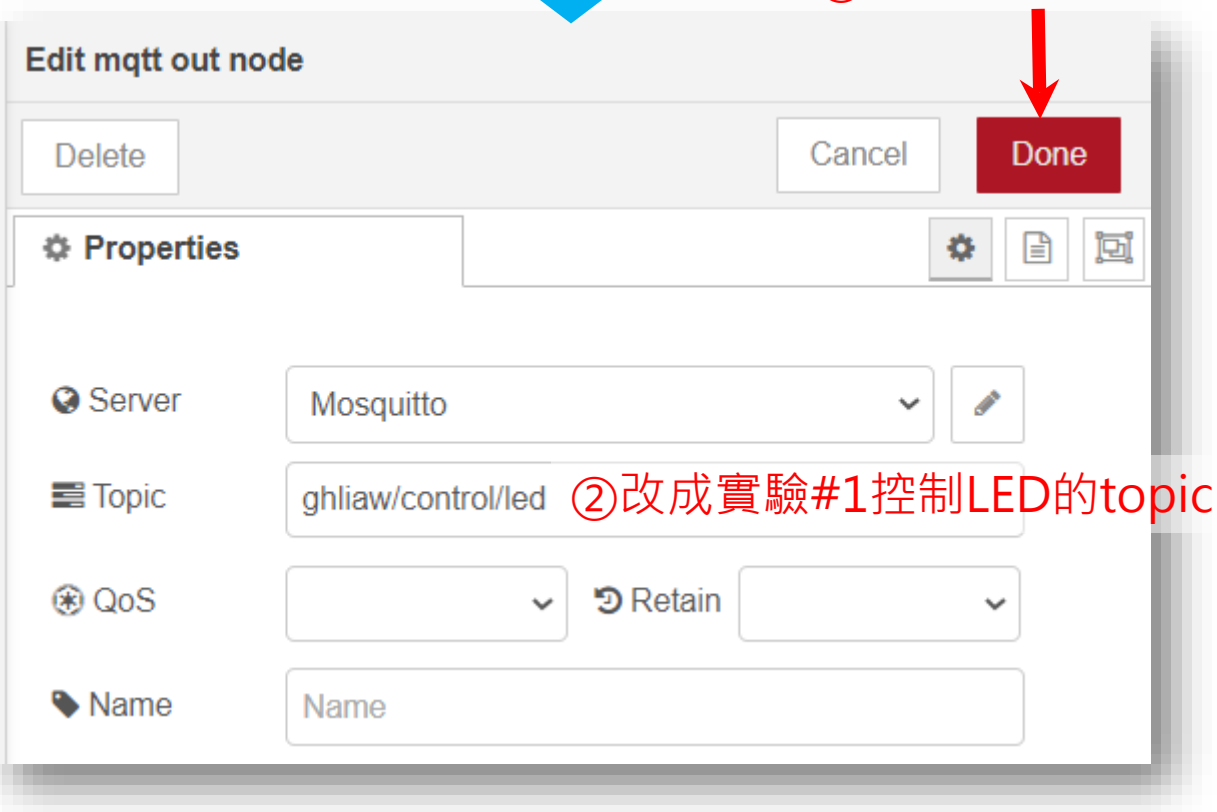
③按下Done完成



①雙擊控制LED的MQTT node



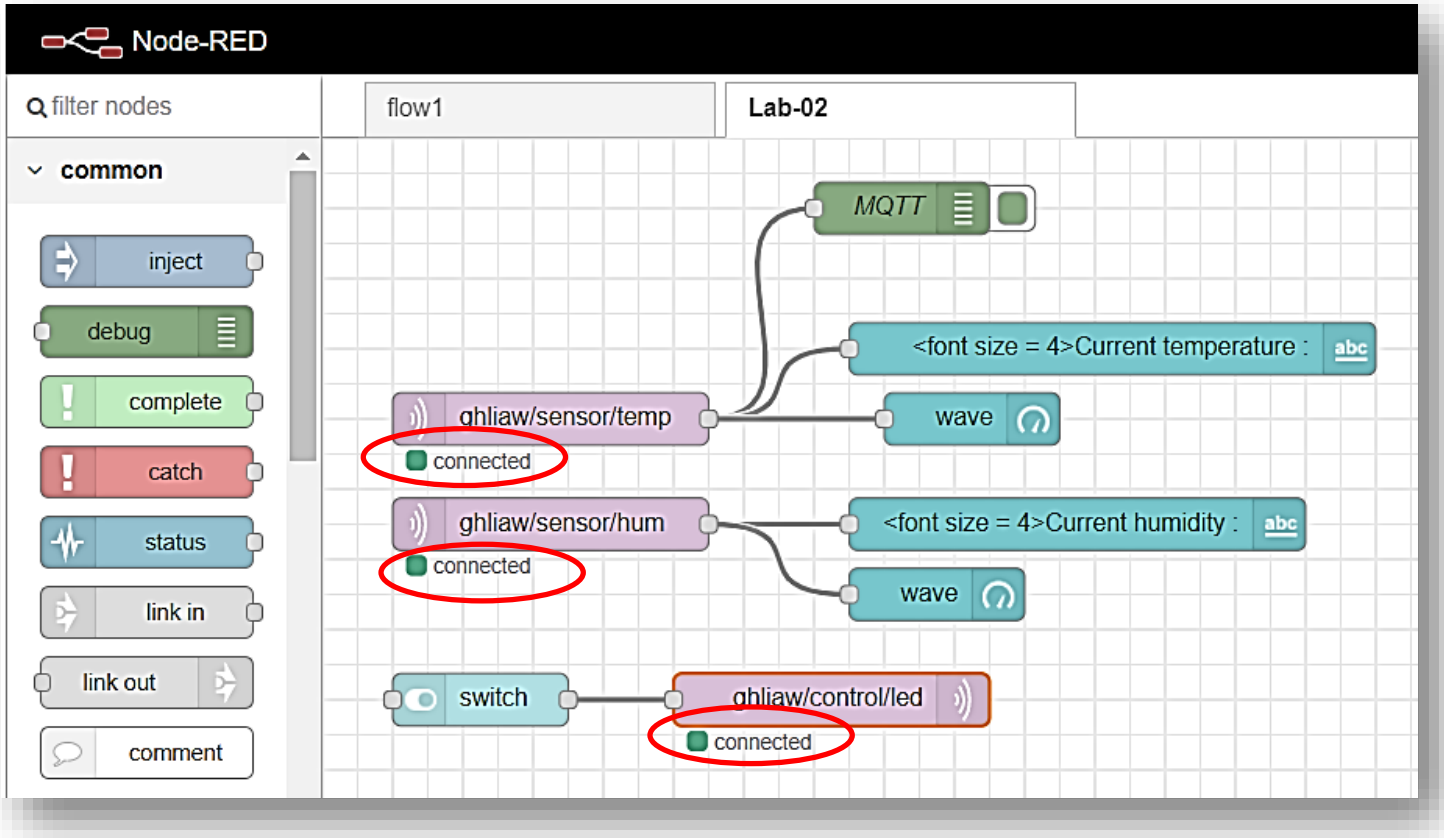
③按下Done完成



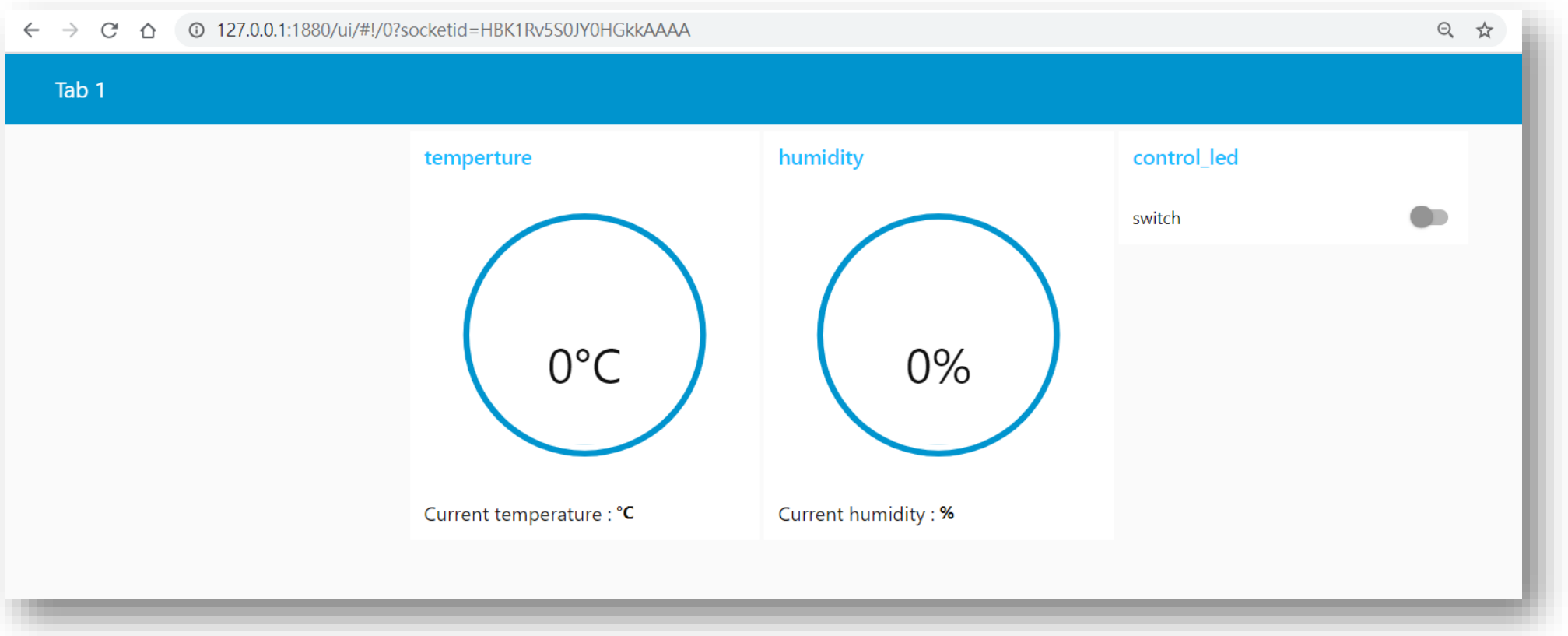
②改成實驗#1控制LED的topic

7. 按下右上角的Deploy按鈕，若顯示"Successfully Deployed"就表示成功佈署。

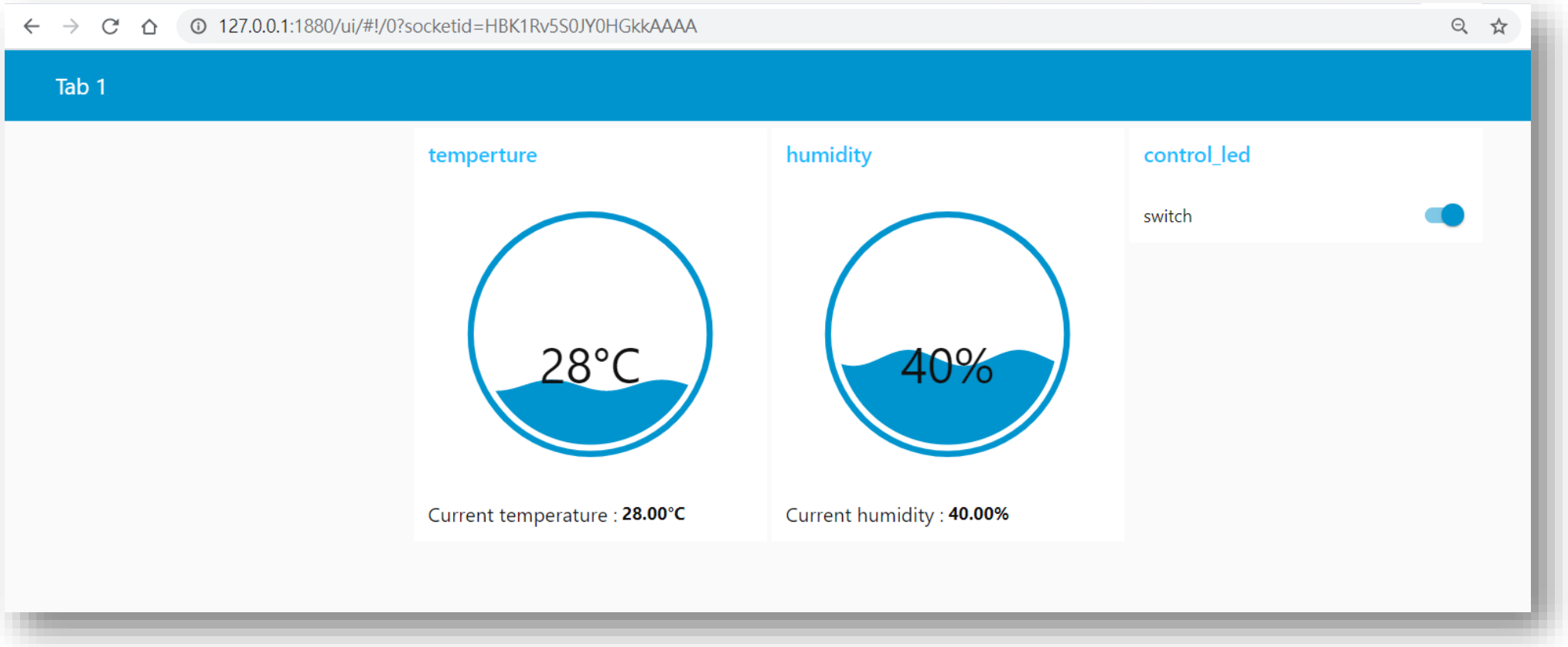
- 此時MQTT nodes下方會顯示與Broker的連線狀態，若為綠色connected，則表示與Broker已經成功建立連線。



8. 打開另一瀏覽器視窗，連線到網址http://127.0.0.1:1880/ui，此即為儀表板頁面。



9. 將實驗#1的板子重新接上電源，順便開啟Arduino IDE的序列埠監控視窗，可看到板子在序列的輸出。當板子發布溫溼度訊息時，觀察在瀏覽器的儀表板頁面是否開始顯示溫溼度數據。同時，也可操作儀表板頁面的開關元件，觀察板子的LED燈是否會跟著明滅。

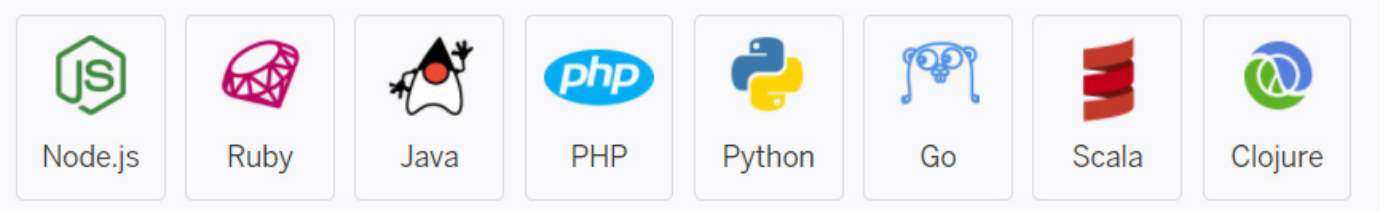


佈署Node-RED程式到Heroku平台

(以Windows 10為工作環境)

Heroku：PaaS雲端平台

- 網址：www.heroku.com
- 屬於Platform as a Service (PaaS)
- 可讓使用者自行佈署各式程式語言編寫的網站後台(稱之為app)



  - 會提供每個app一個對應的URL網址，形式為 <*app name*>.herokuapp.com，讓用戶可以透過HTTP連線到該服務
- 提供有限制的免費帳號，不須提供信用卡號碼便可申請&使用
  - 可參考其定價網頁： https://www.heroku.com/pricing
  - 實際上，Heroku是使用一部虛擬機來執行使用者所部署的應用服務
  - 但應用服務若30分鐘沒有被使用，該虛擬機就會進入休眠狀態，下次被連線時會先啟動再做回應，因此回應時間會比較慢

請先安裝本實驗用到的軟體

- **解壓縮工具：7-Zip**
  - 下載網址：https://www.7-zip.org/a/7z1900-x64.exe
- **Git版本控制工具**
  - 下載網址： https://github.com/git-for-windows/git/releases/download/v2.33.0.windows.2/Git-2.33.0.2-64-bit.exe
- **Heroku命令列工具軟體**
  - 下載網址： https://cli-assets.heroku.com/heroku-x64.exe

佈署步驟說明

- **本佈署步驟是參考以下網址而得：**
  - https://www.wissel.net/blog/2018/02/running-nodered-on-heroku-with-salesforce.html
- **Heroku佈署程序：**
  1. 在Heroku平台建立一個App
  2. 在自己的電腦中，用git將要佈署的網站內容(含程式)納管
  3. 使用heroku命令列工具，指定git的遠端倉儲為之前建立的Heroku App
  4. 使用git將網站內容推送(push)到Heroku App
  5. 推送完畢之後，Heroku會自動做內部佈署的流程，並直接啟用網站
  6. 使用瀏覽器連線到App對應的網址，便可看到網站內容

- **本實驗詳細步驟：**
  1. 若沒有Heroku帳號，請先申請一個帳號並登入
  2. 在Heroku平台上建立一個新的App
  3. 將實驗#2所寫的Node-RED程式之資料夾所有內容(位置在 C:\Users\<username>\.node-red)，全部複製到一個新的專案資料夾(例如：D:\heroku-test)
  4. 進入專案資料夾(即D:\heroku-test)， 依序做以下處理：
     - 修改setting.js與package.json
     - 新增一個文字檔.gitignore (其內容為不想被git納管的檔案/資料夾清單)

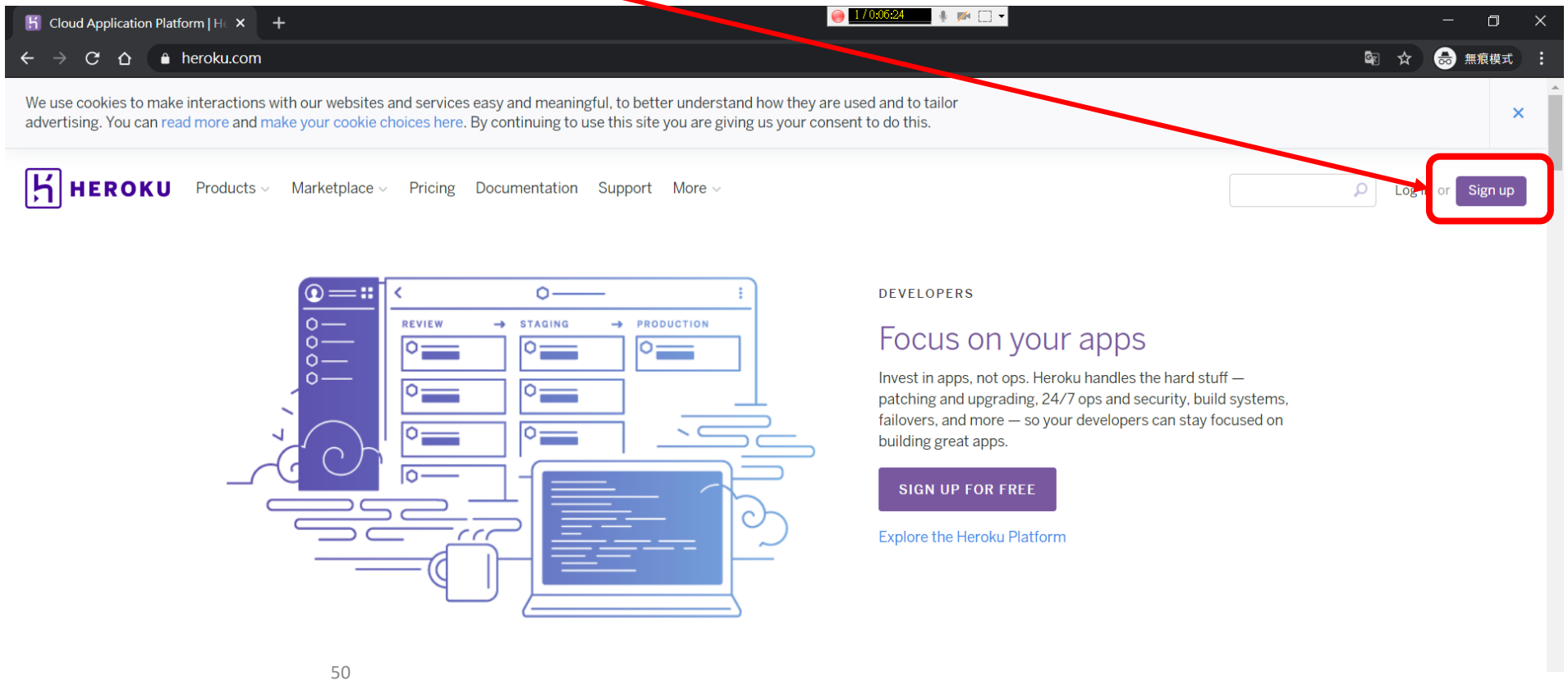
5. 打開命令提示字元視窗，變更其工作資料夾到專案資料夾(即 D:\heroku-test)，然後依序執行以下指令：
   - 因為有改過package.json,必須讓npm重新產生對應的package-lock.json，指令：**npm install**
   - 執行此資料夾被git納管的初始化，指令：**git init**
6. 接續在上一步驟的命令提示字元視窗中，依序執行以下heroku命令列指令：
   - 登入heroku帳號：**heroku login -i**
   - 設定要佈署到哪個APP，指令：**heroku git:remote –a *你的Heroku App名稱***

7. 在命令提示字元視窗中，依序執行以下git指令，進行內容確認與推送：
   - 將所有內容納管，指令：**git add --all**
   - 確認更動過的內容，指令：**git commit -m "initial creation"**
   - 推送到heroku平台，指令：**git push heroku master**

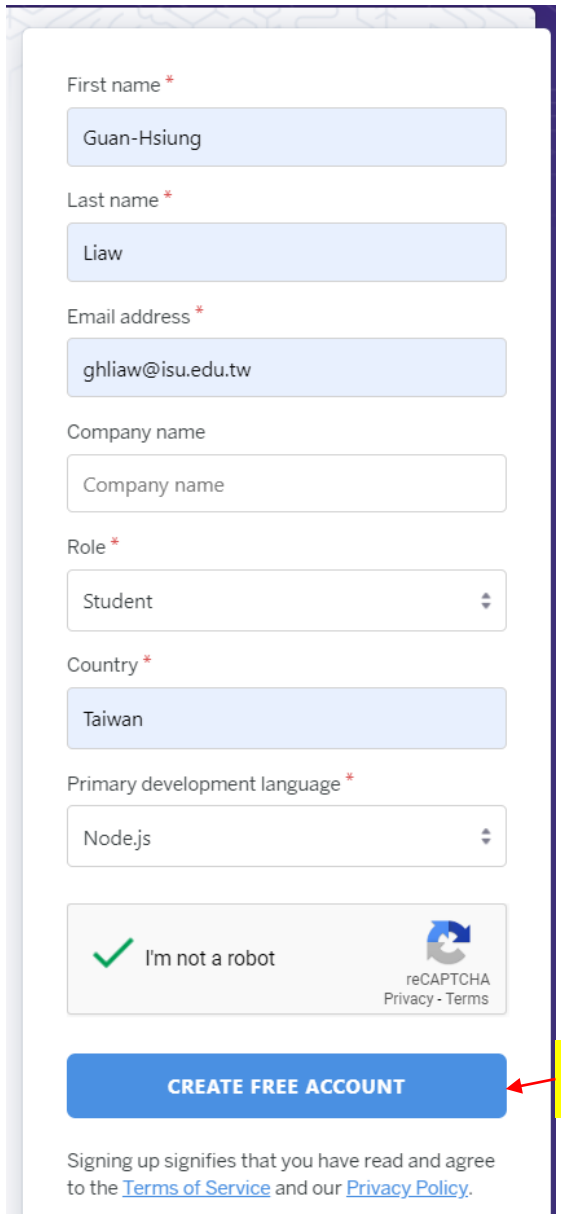
   (在這個命令提示字元視窗尚未關閉時，若資料夾內的任何檔案有所更動，則重新執行這三個命令便可將更動後的內容重新推送到Heroku做更新)
8. 打開瀏覽器，連結到此App對應的網址，驗證是否佈署成功

1. 註冊Heroku平台帳號(www.heroku.com)

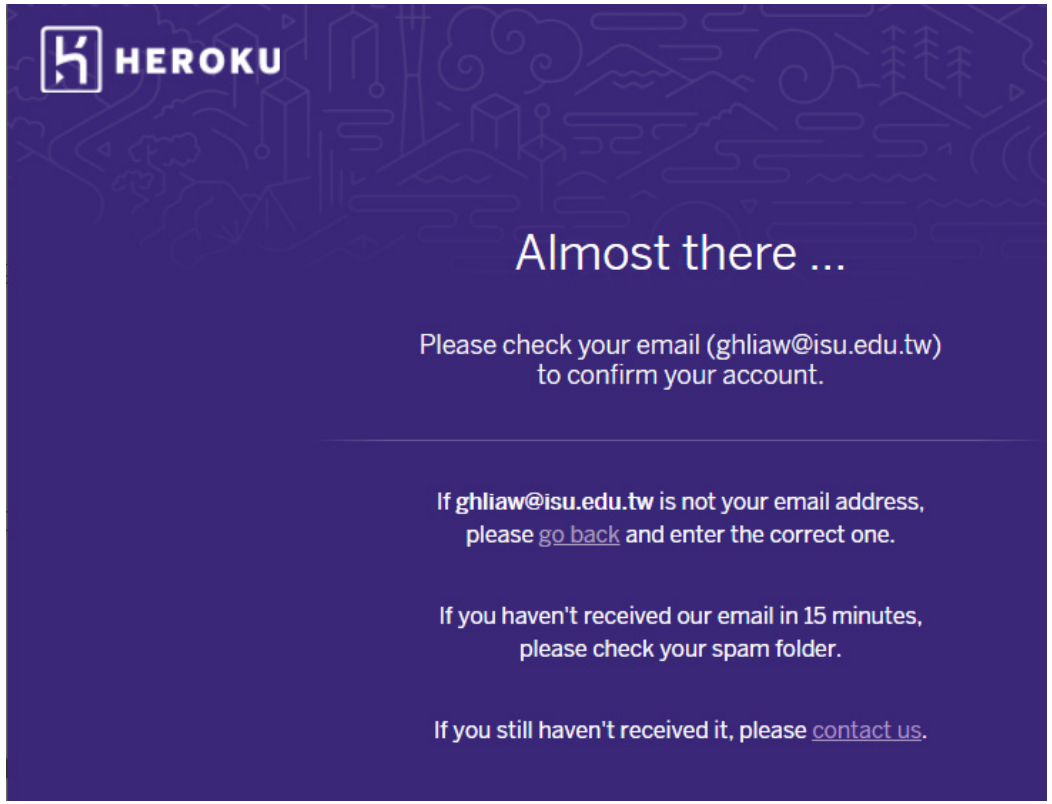
- 開啟web瀏覽器(例如Chrome)，連線到www.heroku.com
- 點擊畫面中的 Signup 按鈕，進入建立帳號頁面



- 填入帳號資料(打*號的必填)



填完後按下CREATE FREE ACCOUNT

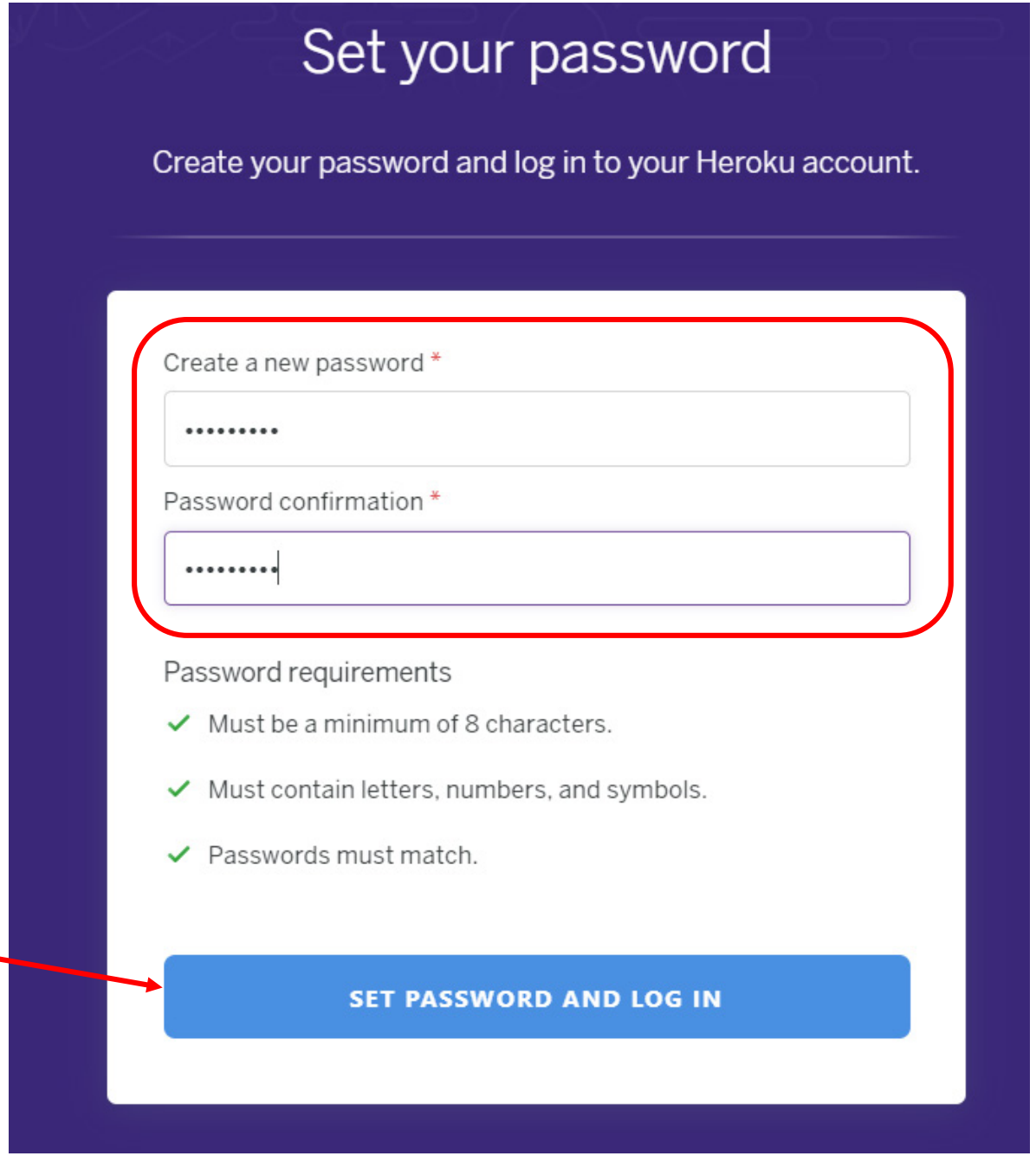


到電郵信箱檢查註冊確認信

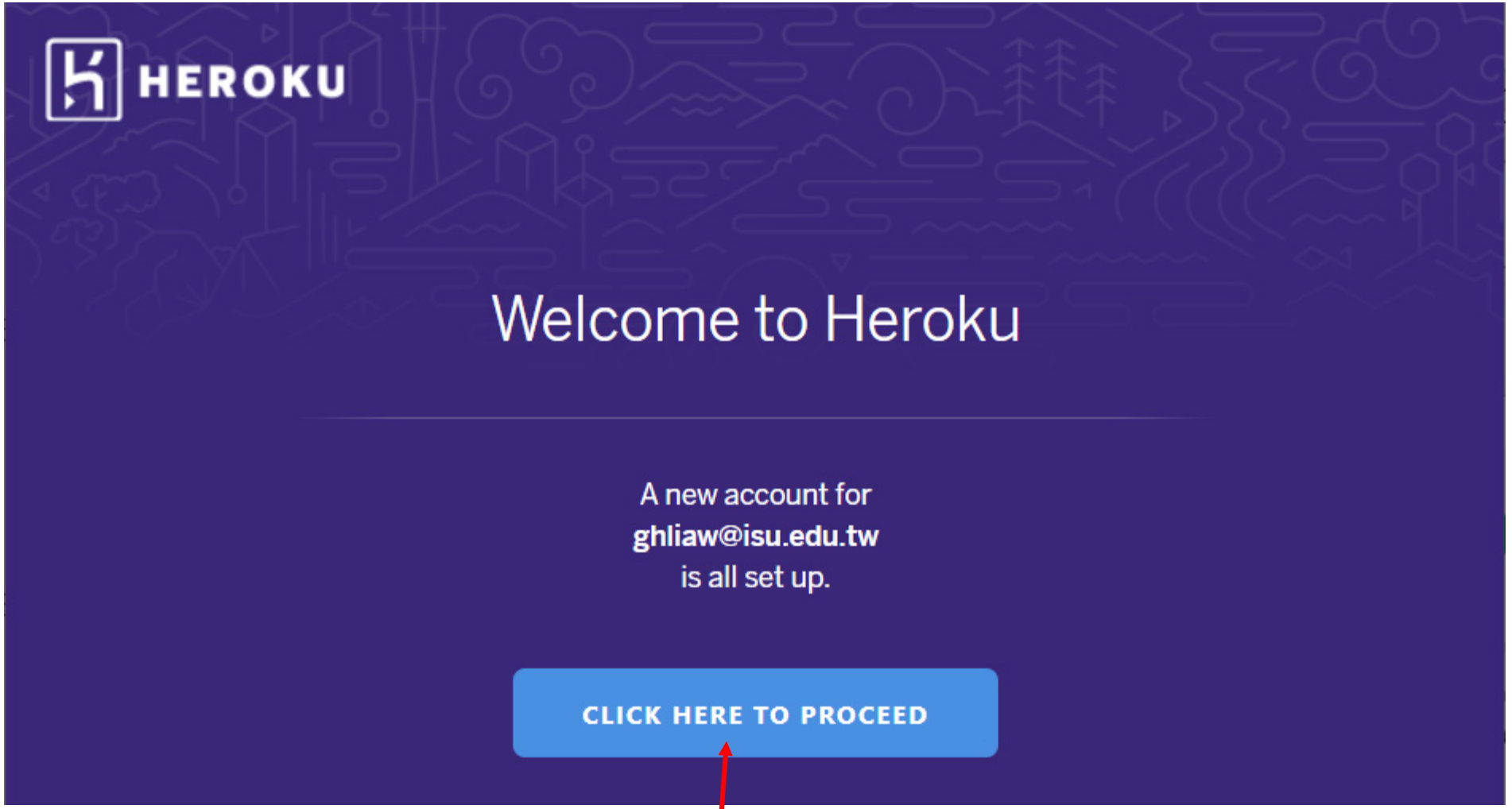
- **進入電郵信箱，檢視註冊確認信函，點擊啟動連結：**

  －會進入設定密碼頁面

- **設定密碼：**
  - 要含有字母、數字、符號

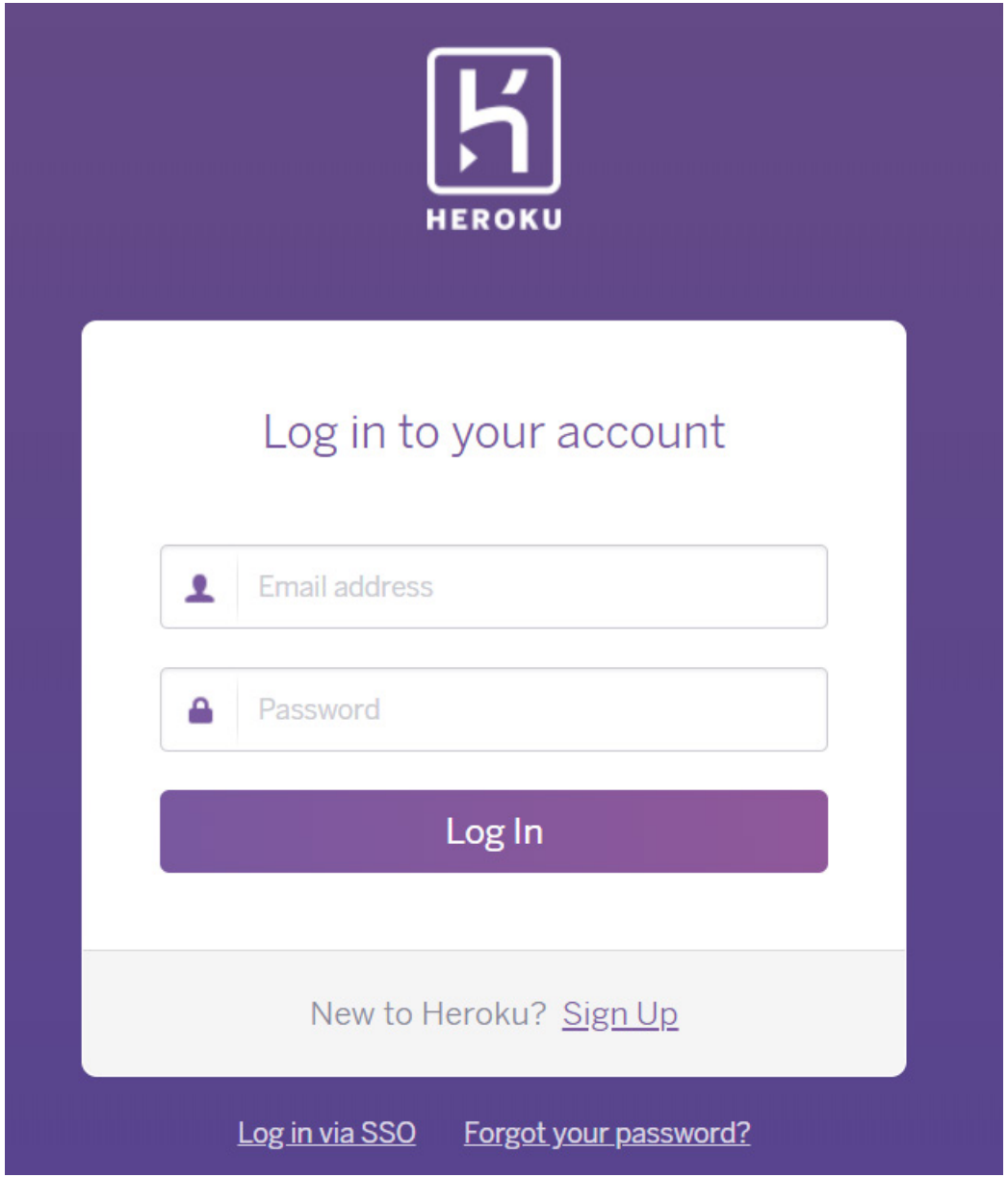


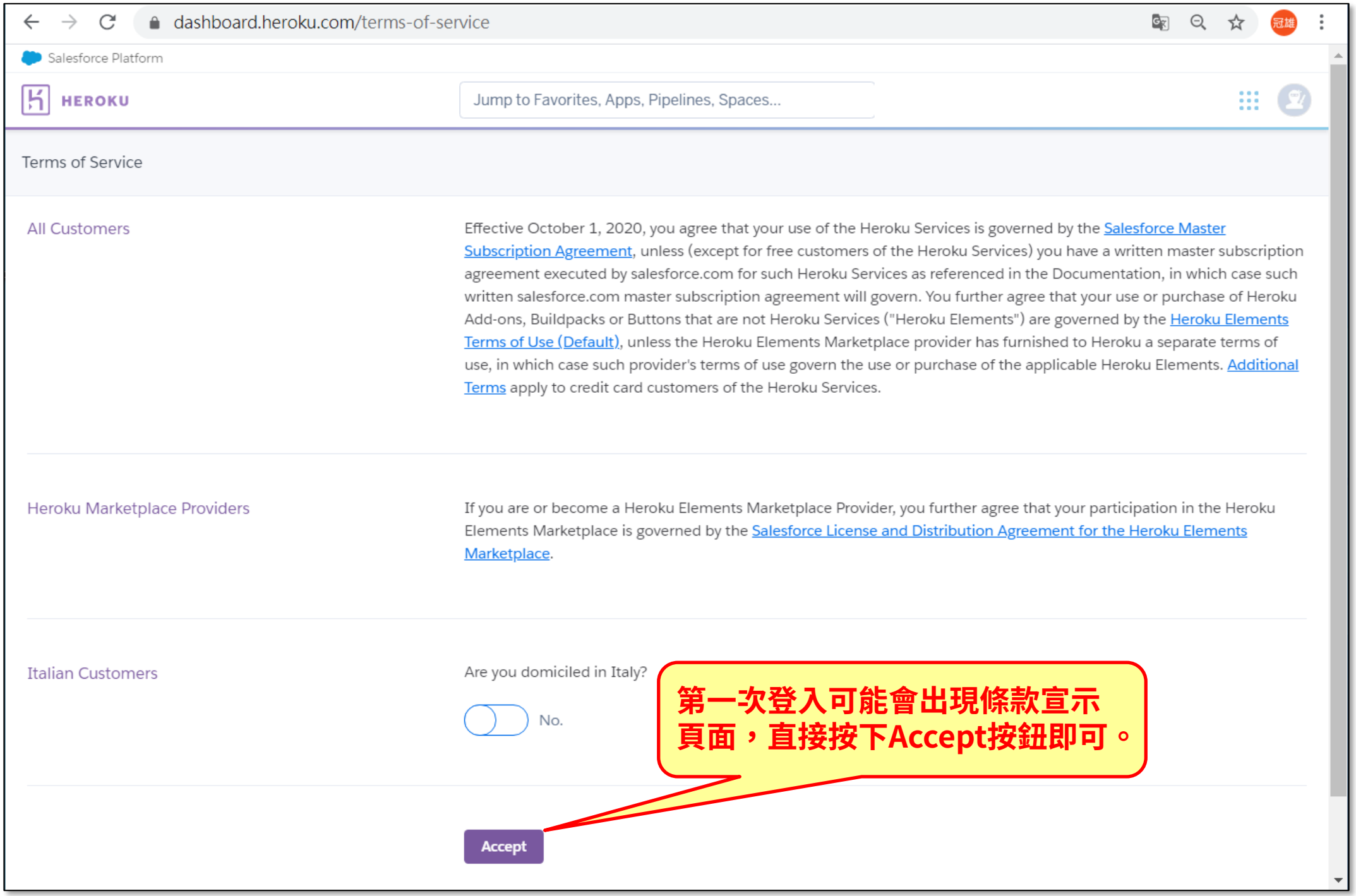
完成後，點擊此處登入


HEROKU
Welcome to Heroku
A new account for
ghliaw@isu.edu.tw
is all set up.
CLICK HERE TO PROCEED

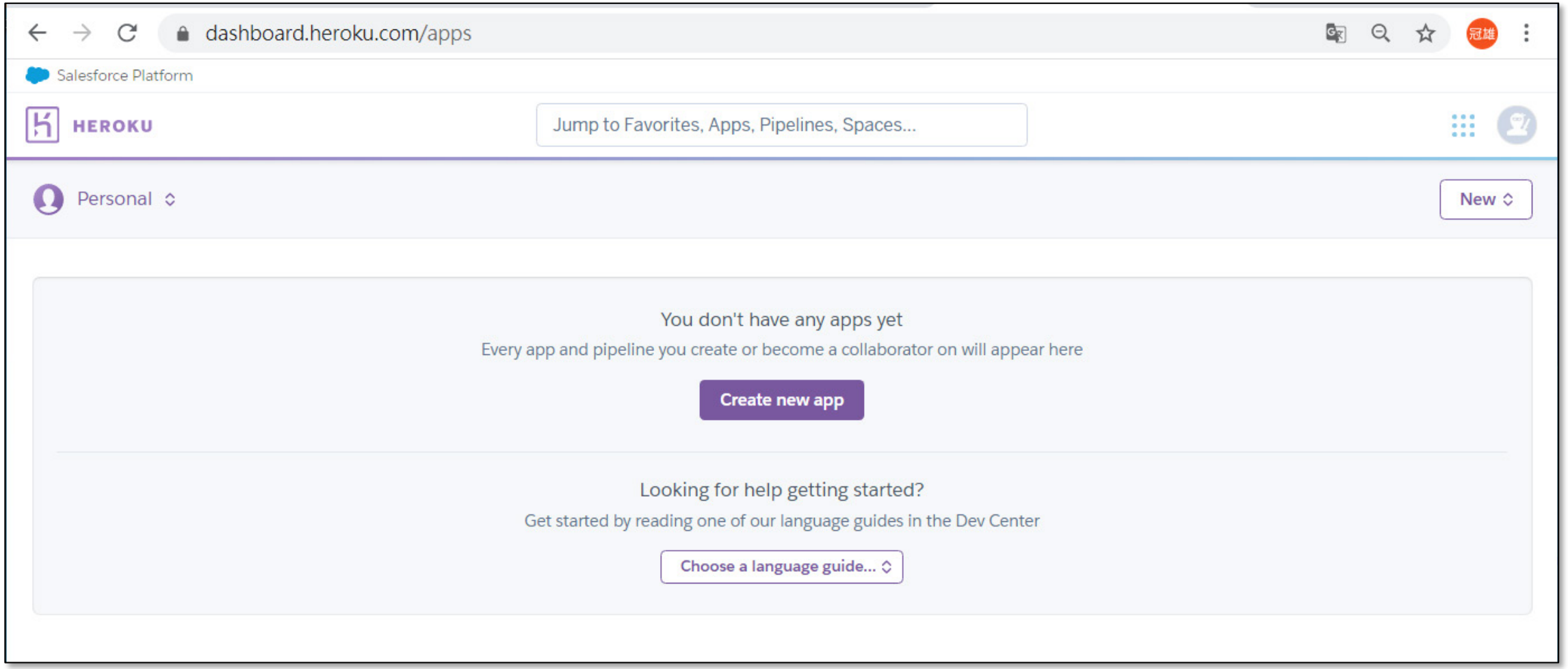
點擊此處繼續

- **進入登入畫面：**
  - 輸入電郵帳號與密碼，按下Login登入

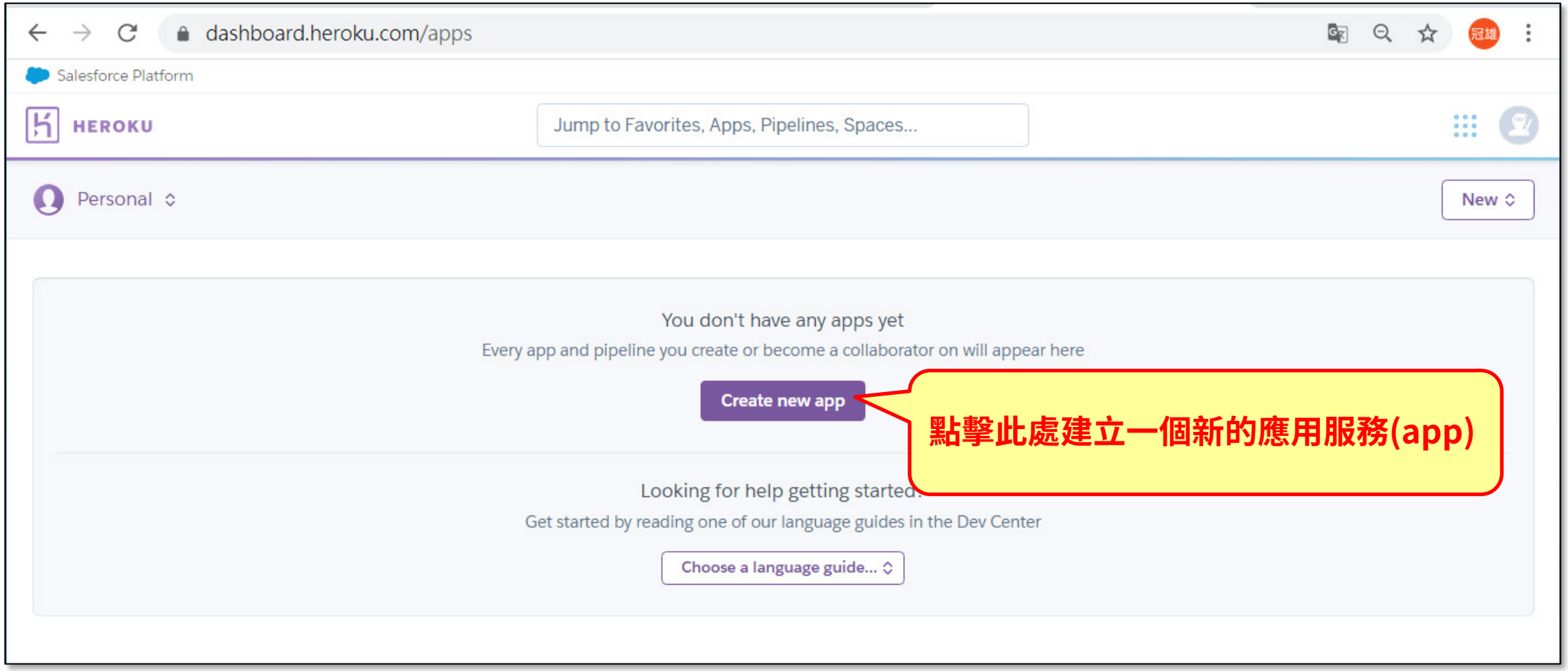



dashboard.heroku.com/terms-of-service
Salesforce Platform
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Terms of Service
All Customers
Effective October 1, 2020, you agree that your use of the Heroku Services is governed by the Salesforce Master Subscription Agreement, unless (except for free customers of the Heroku Services) you have a written master subscription agreement executed by salesforce.com for such Heroku Services as referenced in the Documentation, in which case such written salesforce.com master subscription agreement will govern. You further agree that your use or purchase of Heroku Add-ons, Buildpacks or Buttons that are not Heroku Services ("Heroku Elements") are governed by the Heroku Elements Terms of Use (Default), unless the Heroku Elements Marketplace provider has furnished to Heroku a separate terms of use, in which case such provider's terms of use govern the use or purchase of the applicable Heroku Elements. Additional Terms apply to credit card customers of the Heroku Services.
Heroku Marketplace Providers
If you are or become a Heroku Elements Marketplace Provider, you further agree that your participation in the Heroku Elements Marketplace is governed by the Salesforce License and Distribution Agreement for the Heroku Elements Marketplace.
Italian Customers
Are you domiciled in Italy?
No.
第一次登入可能會出現條款宣示
頁面，直接按下Accept按鈕即可。
Accept

- 登入後會進入儀表板(dashboard)頁面：



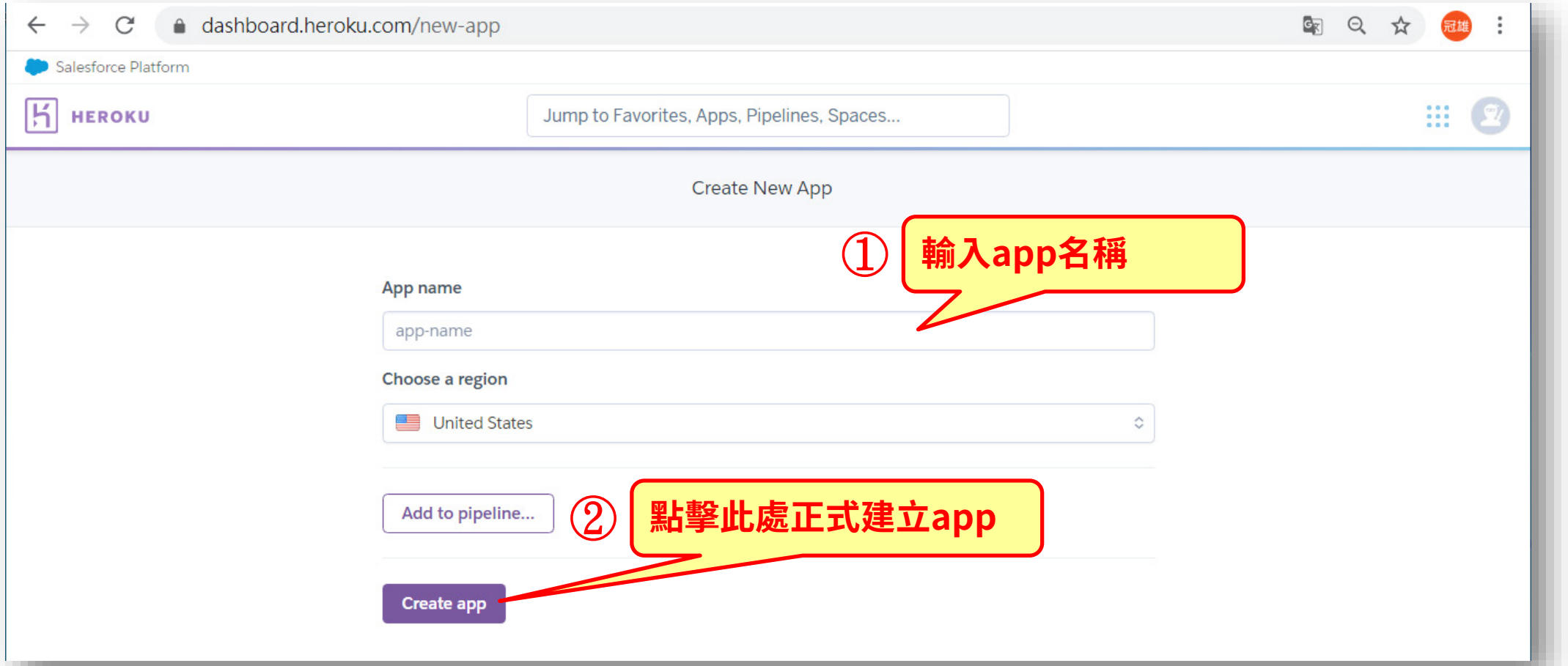
2. 建立一個應用服務(app)

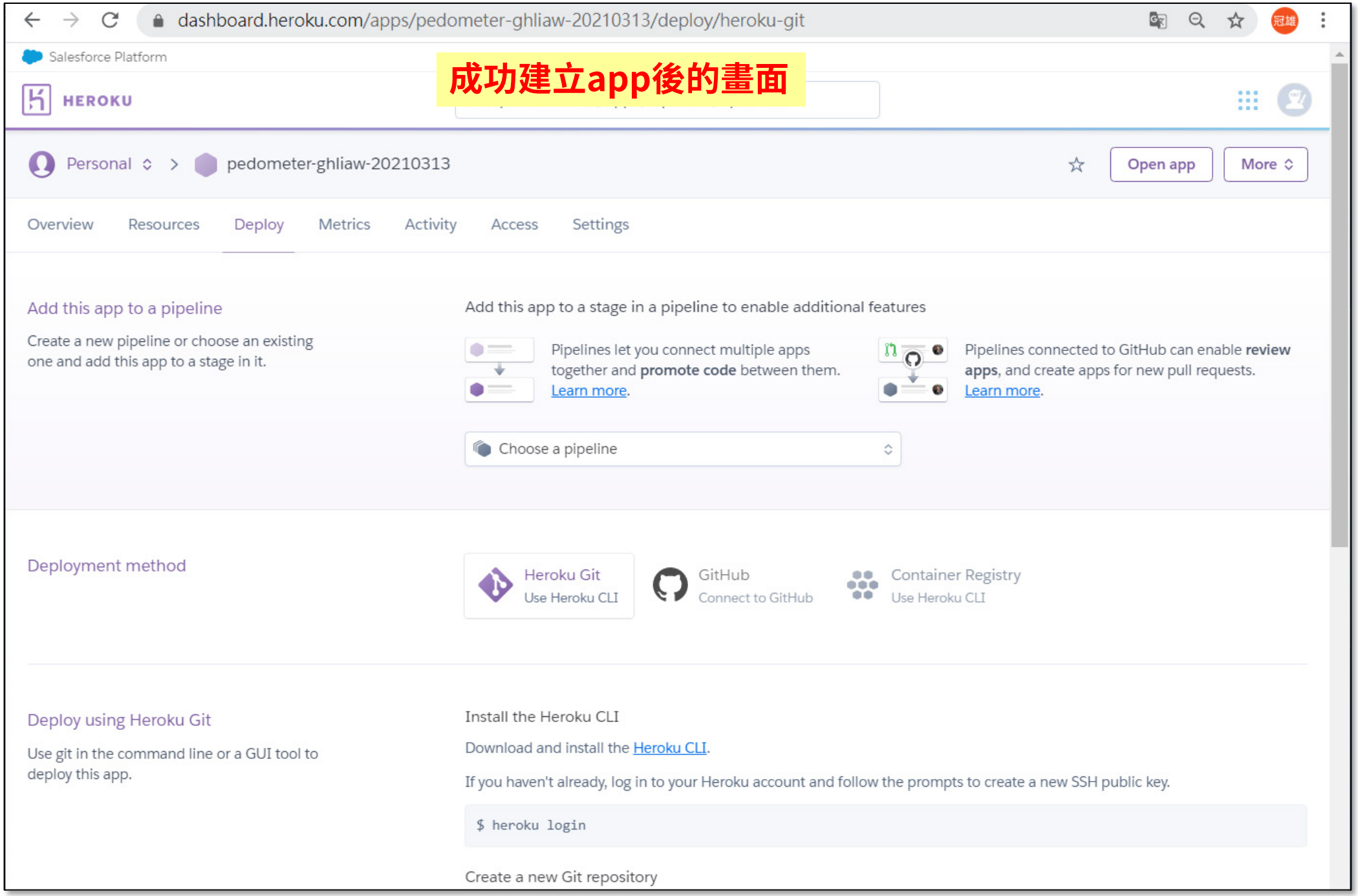


- **輸入應用服務名稱**

  –先想好這個應用服務的名稱，例如「mqtt-test-xxxxxxx」

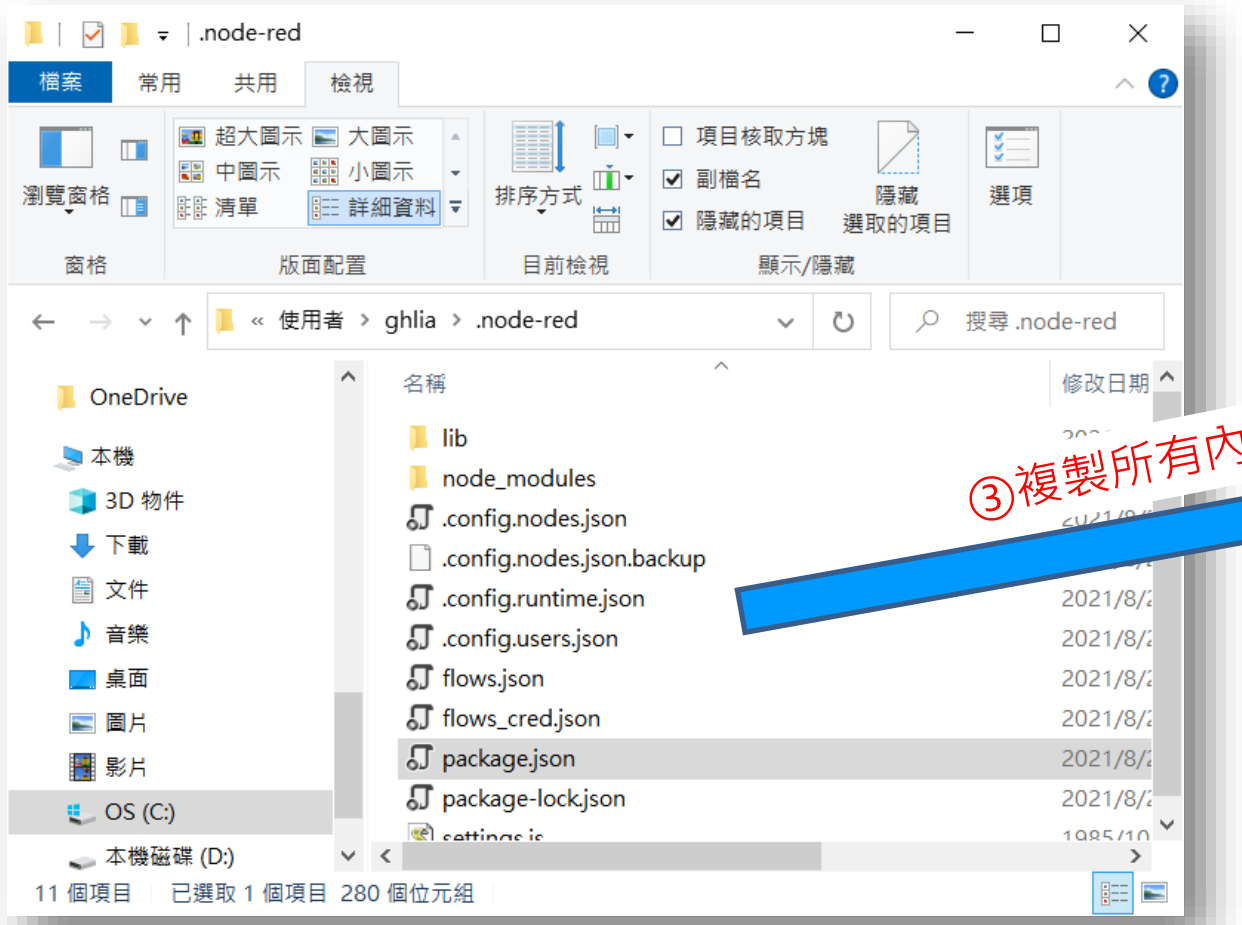
  - 其中，**xxxxxxx**請自行定義，必須要和別人不一樣，因為應用服務的名稱將會是網址的一部分，必須是全球唯一。



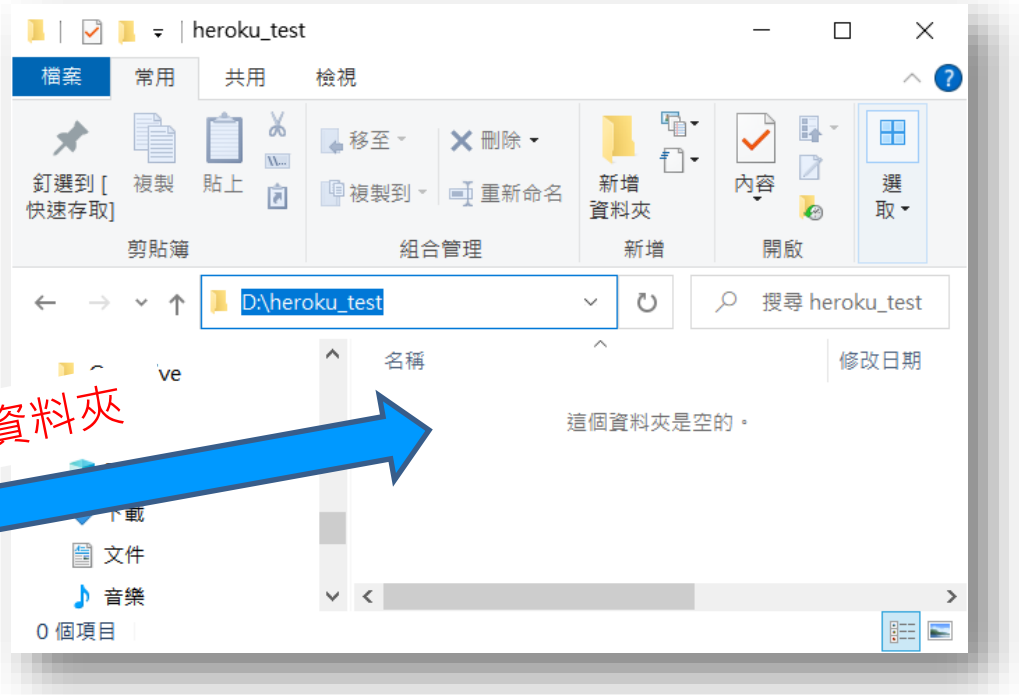

dashboard.heroku.com/apps/pedometer-ghliaw-20210313/deploy/heroku-git
Salesforce Platform
成功建立app後的畫面
HEROKU
Personal
pedometer-ghliaw-20210313
Open app
More
Overview
Resources
Deploy
Metrics
Activity
Access
Settings
Add this app to a pipeline
Create a new pipeline or choose an existing one and add this app to a stage in it.
Add this app to a stage in a pipeline to enable additional features
Pipelines let you connect multiple apps together and promote code between them. Learn more.
Pipelines connected to GitHub can enable review apps, and create apps for new pull requests. Learn more.
Choose a pipeline
Deployment method
Heroku Git
Use Heroku CLI
GitHub
Connect to GitHub
Container Registry
Use Heroku CLI
Deploy using Heroku Git
Use git in the command line or a GUI tool to deploy this app.
Install the Heroku CLI
Download and install the Heroku CLI.
If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.
$ heroku login
Create a new Git repository

3. 複製實驗#2的Node-RED程式到新資料夾

①找到Node-RED程式位置(C:\Users\<*username*>\.node-red)



②建立新資料夾(例如D:\heroku_test)



③複製所有內容到新資料夾

4-1. 修改settings.js

- **因為佈署到公有雲，等於是網站直接暴露在公眾Internet，而Node-RED編輯器頁面一連線就會開啟，所以要把管理介面的身分認證功能(帳號/密碼)打開**
  - settings.js為Node-RED系統設定檔，許多系統設定都在這裡，裡面註解寫得很完整
  - 有關管理介面的帳號密碼設定，可參考官網網頁：http://nodered.org/docs/security.html

- **修改步驟如下：**
  - 使用文字編輯器(例如notepad++)打開settings.js
  - 找到「adminAuth」的設定 (如下頁圖)
  - 把「adminAuth」這一節每行前面的「//」去掉(即啟用這一節的設定，如下頁圖)
  - 把username欄位的值改成你想要的帳號名稱(本例為"iot2021")
  - 注意：密碼必須使用node-red工具將密碼的明文轉成雜湊(hash)後的暗文，步驟如下(如下下頁圖)：
    - 在命令提示字元中，輸入指令：node-red admin hash-pw
    - 當出現「Passowrd:」提示語時，輸入密碼的明文(注意：此時游標不會隨著輸入文字而移動，盡儘管輸入便是)
    - 輸入完成後按下Enter鍵，就會出現雜湊後的結果
  - 複製雜湊後的密碼，填到password欄位中(覆蓋掉原有的)
  - 完成修改後記得存檔

修改
settings.js

```
72
73      /** To password protect the Node-RED editor and admin API, the following
74       * property can be used. See http://nodered.org/docs/security.html for
         details.
75       */
76      //adminAuth: {
77      //    type: "credentials",
78      //    users: [{
79      //        username: "admin",
80      //        password:
        "$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNbM6tl8sJogENOMcxWV9DN.",
81      //        permissions: "*"
82      //    }]
83      //},
```

```
72
73      /** To password protect the Node-RED editor and admin API, the following
74       * property can be used. See http://nodered.org/docs/security.html for
         details.
75       */
76      adminAuth: {
77          type: "credentials",
78          users: [{
79              username: "iot2021",
80              password:
                "$2b$08$58nUuc7RVohaoPIQLgP7Pe82gs9uWL1MX2TGi.i9yE.K8YwdN9CWW",
81              permissions: "*"
82          }]
83      },
```

產生密碼的雜湊暗文：

注意：同一密碼明文，重複執行上述指令所產生的雜湊暗文，每次都會不一樣!

4-2. 修改package.json

- 要在package.json加上以下兩個部分，才可以讓Heroku平台自動去安裝node-red以及所需的的套件，以及自動啟動node-red (如下頁圖)
  - 啟動專案時要跑的指令：

```
"scripts": {
        "start": "node-red --settings ./settings.js --userDir ./"
},
```

  - 在相依套件(dependencies)中加上"node-red"，並指名版本：

```
"node-red": "~2.0.5",
```

  - 完成修改後記得存檔

package.json修改：

```
{
    "name": "node-red-project",
    "description": "A Node-RED Project",
    "version": "0.0.1",
    "private": true,
    "dependencies": {
        "node-red-dashboard": "~2.30.0"
    }
}
```

```
{
    "name": "node-red-project",
    "description": "A Node-RED Project",
    "version": "0.0.1",
    "scripts": {
        "start": "node-red --settings ./settings.js --userDir ./"
    },
    "private": true,
    "dependencies": {
        "node-red": "~2.0.5",
        "node-red-dashboard": "~2.30.0"
    }
}
```

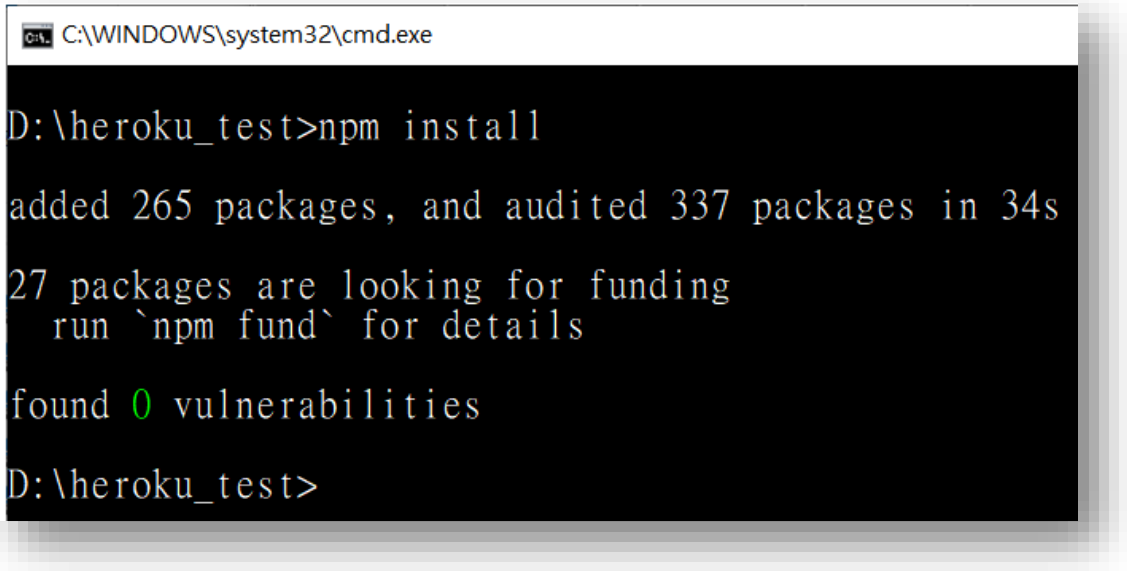
4-3. 新增文字檔「.gitignore」

- 「.gitignore」檔案是用來列出不想被git納管(即被git忽略)的檔案或資料夾清單
- 在專案資料夾下，新增「.gitignore」文字檔，並填入以下內容後存檔：

```
lib/
node_modules/
scripts/
.npm
#NodeRed runtime stuff
.node-red/.config.json
.config.nodes.json
.config.runtime.json
.config.users.json
.node-red/.sessions.json
.node-red/*.backup
```

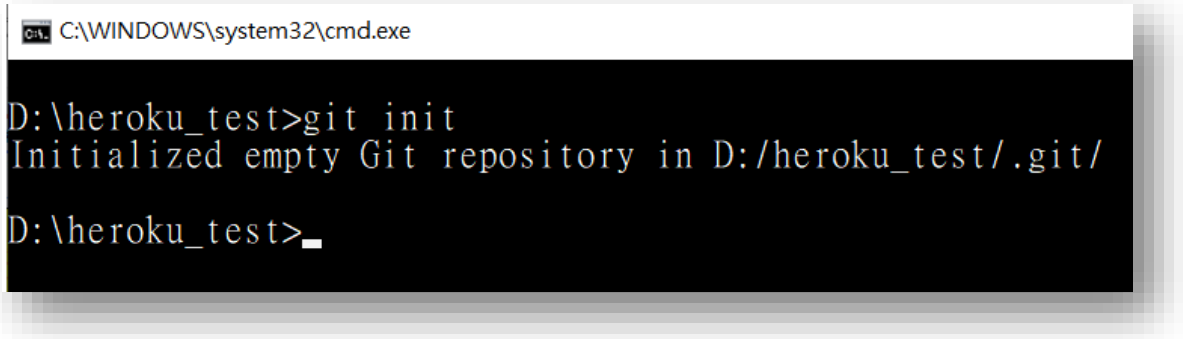
5-1. 執行指令「npm install」

- 因為package.json有被改過，因此要執行指令「必須讓npm重新產生對應的package-lock.json，
- 步驟如下：
  - 開啟一個命令提示字元視窗，變更工作資料夾到專案資料夾(本例為D:\heroku_test)
  - 執行指令「npm install」(如右圖)



5-2. 執行git初始化「git init」

- **專案被git納管要做初始化，指令：git init**
  - 注意：初始化指令只能做一次!
    - 若在佈署到heroku之後重新執行git init，整個專案的納管資訊會被重置，將導致此專案的後續佈署會不成功(因為版次內容有衝突)
- **步驟如下：**
  - 繼續在先前的命令提示字元視窗中，輸入指令：**git init**（如下圖）



6-1. 在命令提示字元中登入Heroku帳號

- 因為後續要在這個命令提示字元中推送專案到自己的Heroku App，因此必須先登入Heroku
  - 執行指令：**heroku login -i**
  - 按下Enter鍵後，依指示輸入帳號與密碼 (如下圖)

6-2. 設定要佈署到哪個App

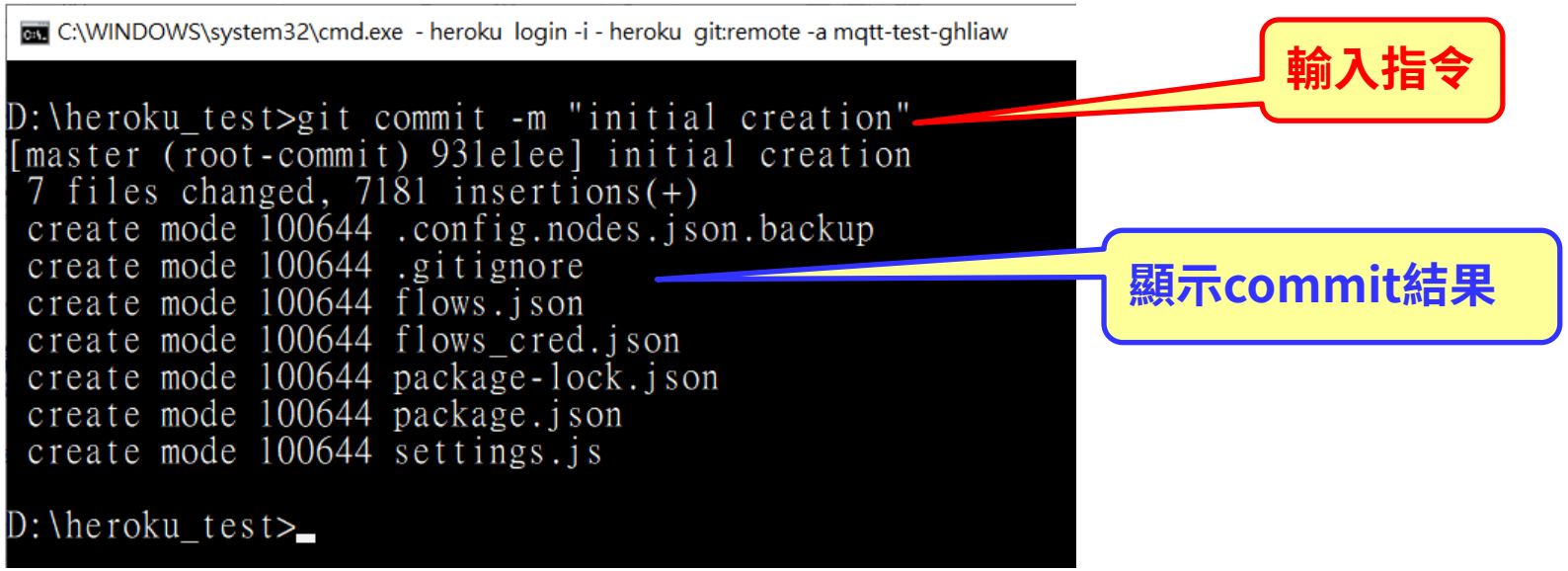
- 指定本專案要佈署到我們在heroku所建立某個App (本例為 mqtt-test-xxxxxxx)
  - 執行指令：**heroku git:remote -a mqtt-test-xxxxxxx**

7-1. 將專案所有內容納入git管理

- 將所有此專案的所有內容都納入git管理
  - 執行指令：**git add --all**
    - .gitignore檔案裡面有列出的不會被加入

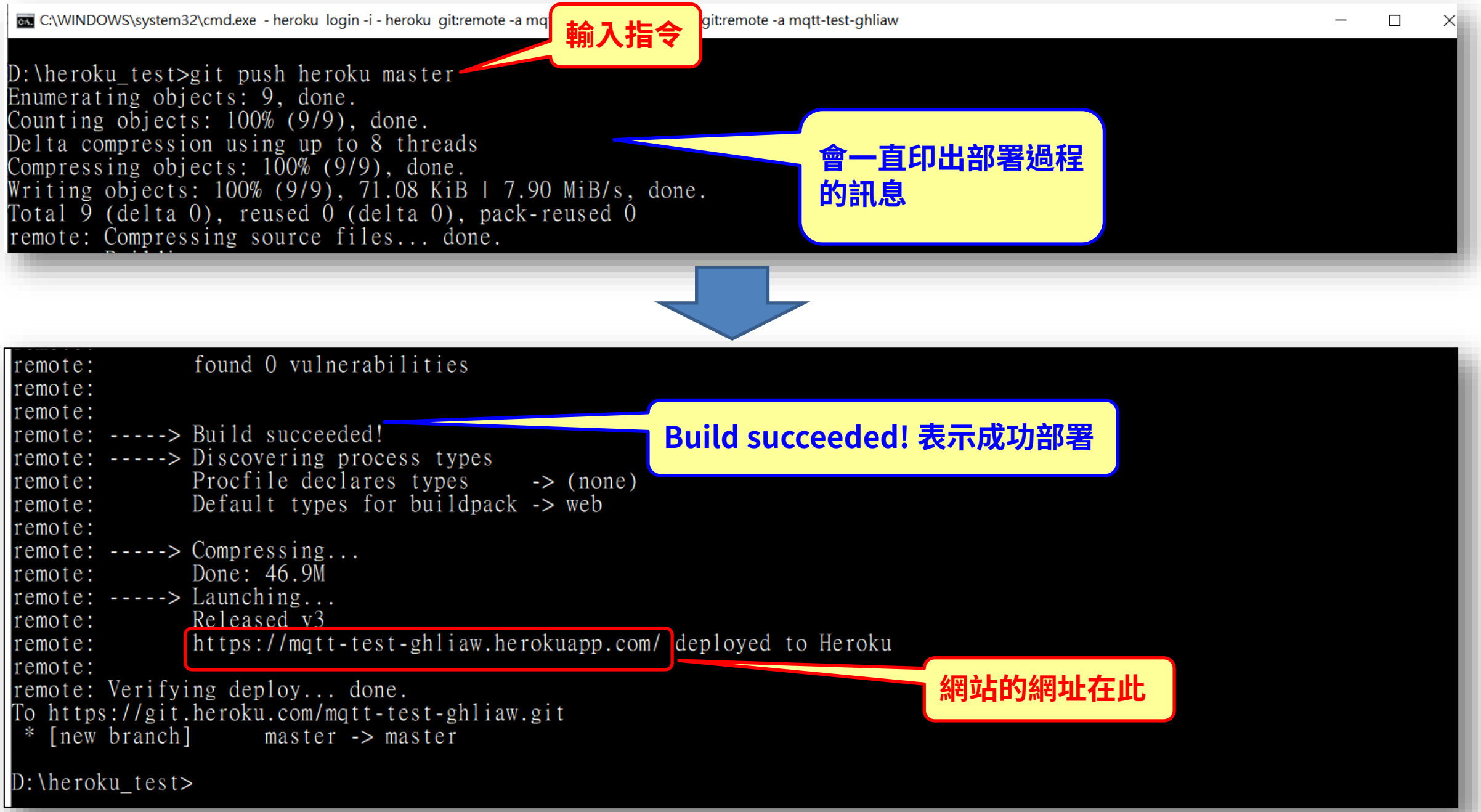
7-2. 確認專案內已更動過的內容

- **確認(commit)此專案內容的所有更動**
  - –Commit的意思是確認之前所有的更動 (即真正變更倉儲中的內容)
  - –執行指令：**`git commit -m "initial creation"`**
    - 「-m」之後的字串是這次commit的說明，用來註記本次變更的內容
    - 之後的每次內容變更，都要執行commit，才會真正更新倉儲中的內容



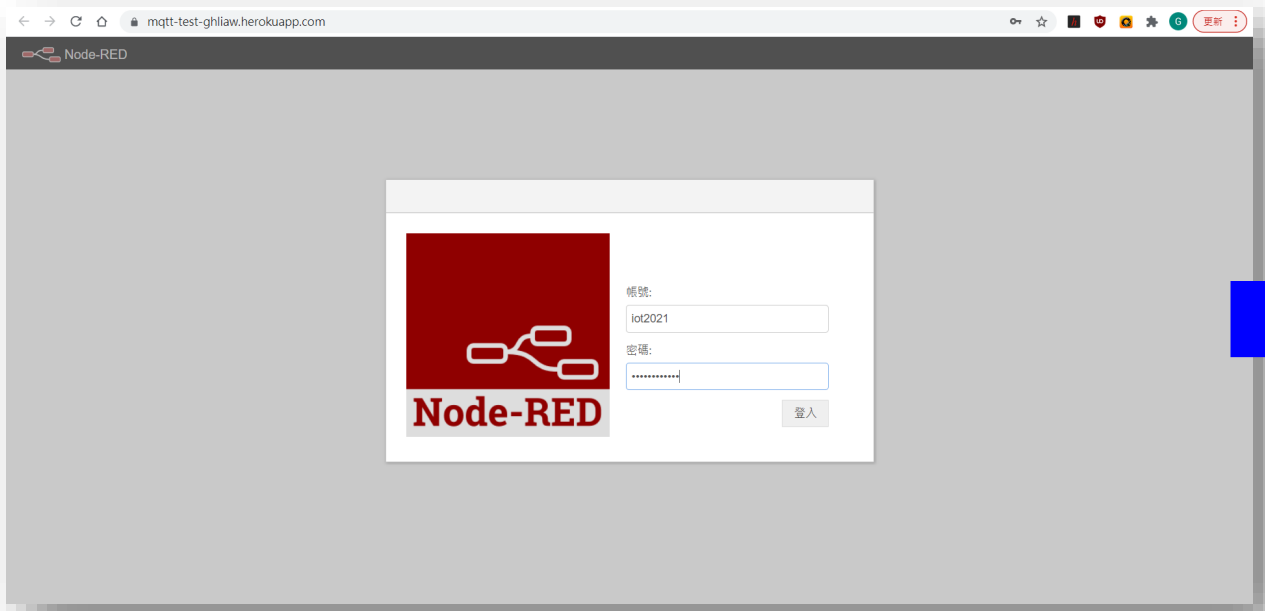
7-3. 推送到Heroku平台(final)

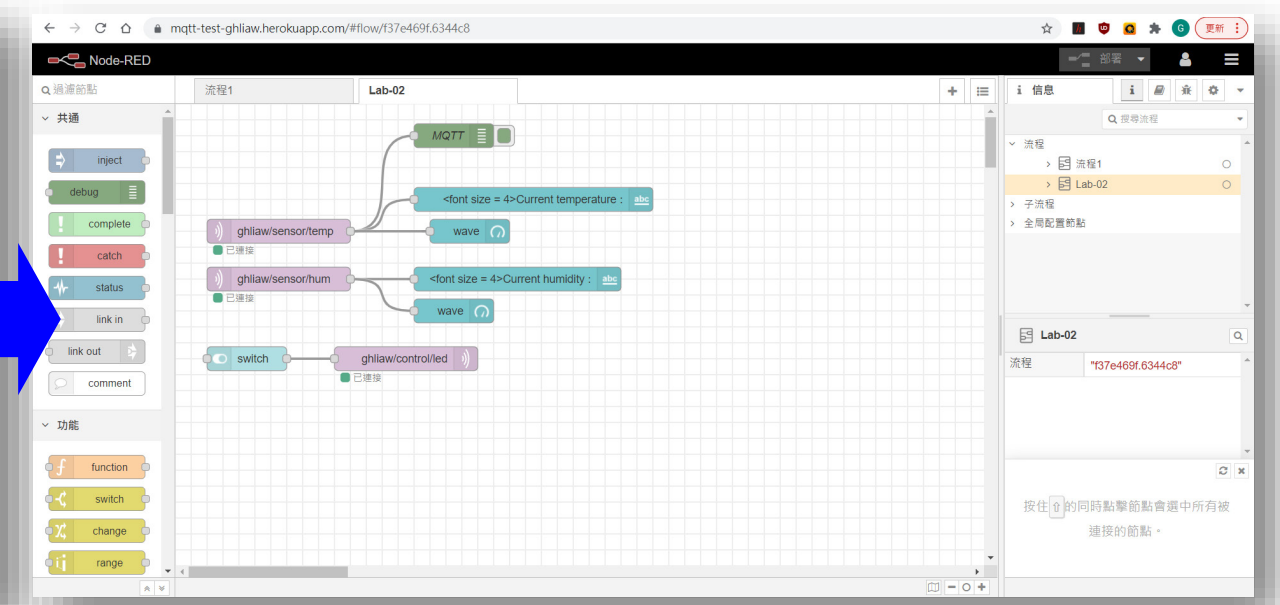
- 將專案內容推送到Heroku平台，使其自動佈署
  - 執行指令：**`git push heroku master`**
  - (過程見下頁)
- 步驟完成後，網站的佈署就大功告成!!


C:\WINDOWS\system32\cmd.exe - heroku login -i - heroku git:remote -a mq
git:remote -a mqtt-test-ghliaw
輸入指令
D:\heroku_test>git push heroku master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 71.08 KiB | 7.90 MiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
會一直印出部署過程的訊息
remote: found 0 vulnerabilities
remote:
remote:
remote: -----> Build succeeded!
Build succeeded! 表示成功部署
remote: -----> Discovering process types
remote: Procfile declares types -> (none)
remote: Default types for buildpack -> web
remote:
remote: -----> Compressing...
remote: Done: 46.9M
remote: -----> Launching...
remote: Released v3
remote: https://mqtt-test-ghliaw.herokuapp.com/ deployed to Heroku
remote:
網站的網址在此
remote: Verifying deploy... done.
To https://git.heroku.com/mqtt-test-ghliaw.git
* [new branch] master -> master
D:\heroku_test>

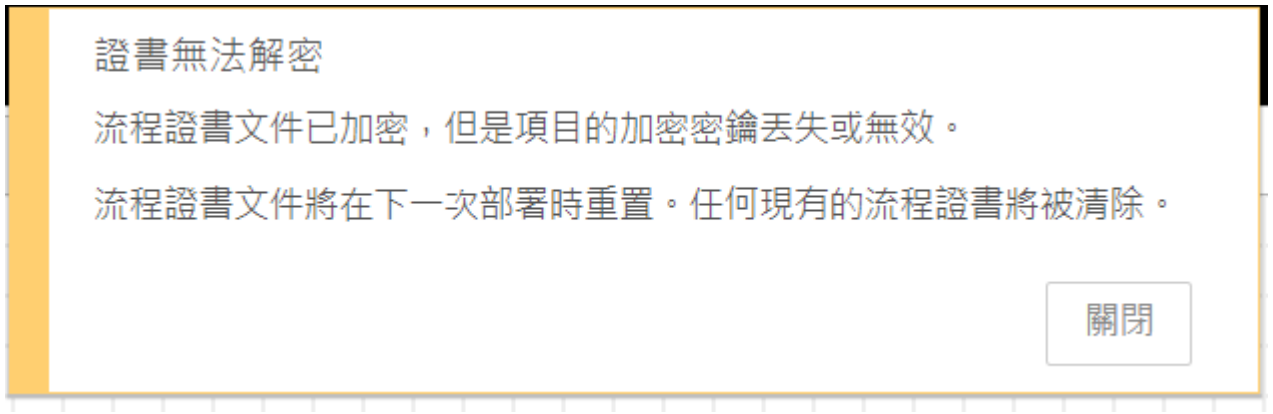
8. 使用瀏覽器驗證是否成功佈署

- 打開瀏覽器，在網址列輸入：
  - –https://mqtt-test-xxxxxxxx.herokuapp.com
- 若成功佈署，會出現登入畫面
  - –輸入先前設定的管理介面帳號與密碼，便可登入



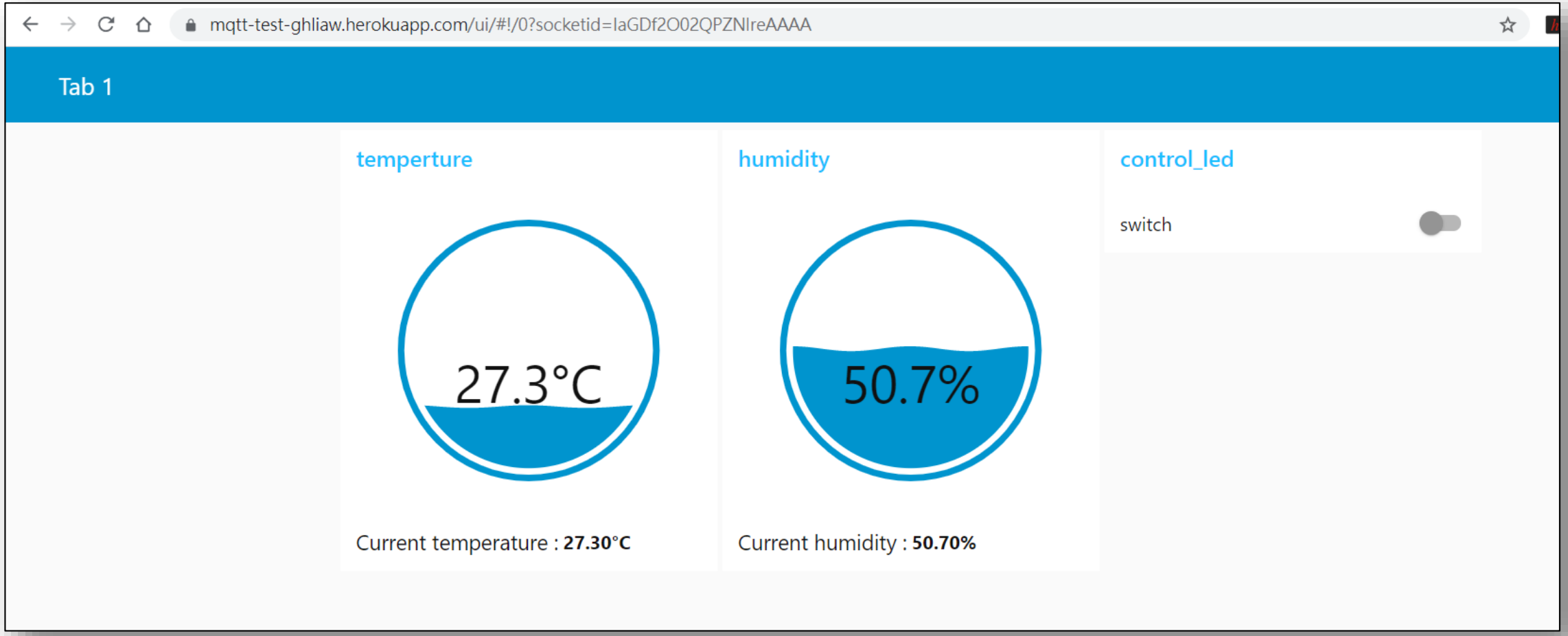


- **會顯示"流程證書無法解密"的訊息，如下：**



–沒有關係，稍微拉動流程中的任一個節點，然後重新佈署即可。之後就不會出現這個訊息了。

- **將App網址後面加上「/ui」變可開啟儀表板頁面**
  - 例如： https://mqtt-test-ghliaw.herokuapp.com/ui



直接使用範例Heroku_Demo佈署

- 可以直接使用資料夾「 Heroku_Demo」中的內容來佈署
  - 裡面的settings.js與package.json都已經改好，且package-lock.json已經與package.json同步
    - 登入帳號為"iot2021"，密碼為"isuCSIE2021#"
  - 只要先到heroku建立一個app，然後直接從步驟5.2 (git init)開始佈署即可。