



Internet of Thing (IoT)

โดย

นายณัฐวัฒน์ พัลวัล

มหาวิทยาลัยเทคโนโลยีราชมงคล ล้านนา

Internet of Thing (IoT) คืออะไร

Internet of Things (IoT)

IoT ย่อมาจาก Internet of Things ซึ่งเป็นแนวคิดที่เกี่ยวข้องกับการเชื่อมต่ออุปกรณ์ต่าง ๆ กันทางอินเทอร์เน็ต เพื่อให้สามารถรับส่งข้อมูลและทำงานร่วมกันได้อย่างอัตโนมัติและมีประสิทธิภาพมากขึ้น

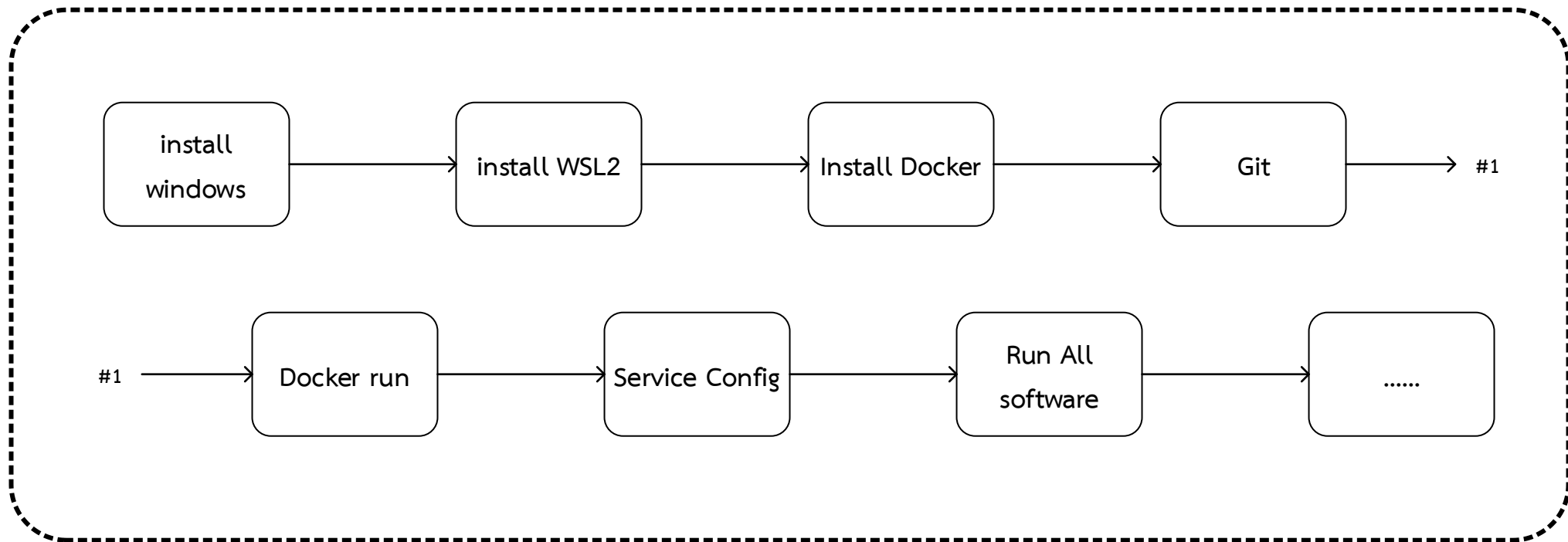
ในระบบ IoT อุปกรณ์ต่าง ๆ เช่น เซ็นเซอร์ เครื่องมือวัด หรืออุปกรณ์ไฟฟ้า เชื่อมต่อกับเครือข่ายอินเทอร์เน็ตและสื่อสารกันได้ ซึ่งทำให้สามารถเก็บรวบรวมข้อมูลจากอุปกรณ์เหล่านั้นและใช้ข้อมูลเหล่านั้นในการวิเคราะห์ ประมวลผล หรือใช้ในการควบคุมอุปกรณ์ต่าง ๆ ได้อย่างสะดวกและรวดเร็ว

ระบบ IoT มีการนำเอาเทคโนโลยีเครือข่าย การสื่อสารไร้สาย การรวมระบบคอมพิวเตอร์ การวิเคราะห์ข้อมูลและปัญญาประดิษฐ์ เพื่อให้เกิดความสามารถในการติดต่อสื่อสารและการทำงานอัตโนมัติระหว่างอุปกรณ์ต่าง ๆ



จะเรียนอะไรบ้าง ?

ในหน่วยเรียนนี้เราจะศึกษาและเรียนรู้เกี่ยวกับการใช้งาน Internet of Things (IoT) ซึ่งเป็นเทคโนโลยีที่เชื่อมต่ออุปกรณ์ต่าง ๆ กับโลกดิจิทัล เพื่อให้เกิดการสื่อสารและการแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์ที่เชื่อมต่อกัน



Windows Subsystem for Linux 2 (WSL2)

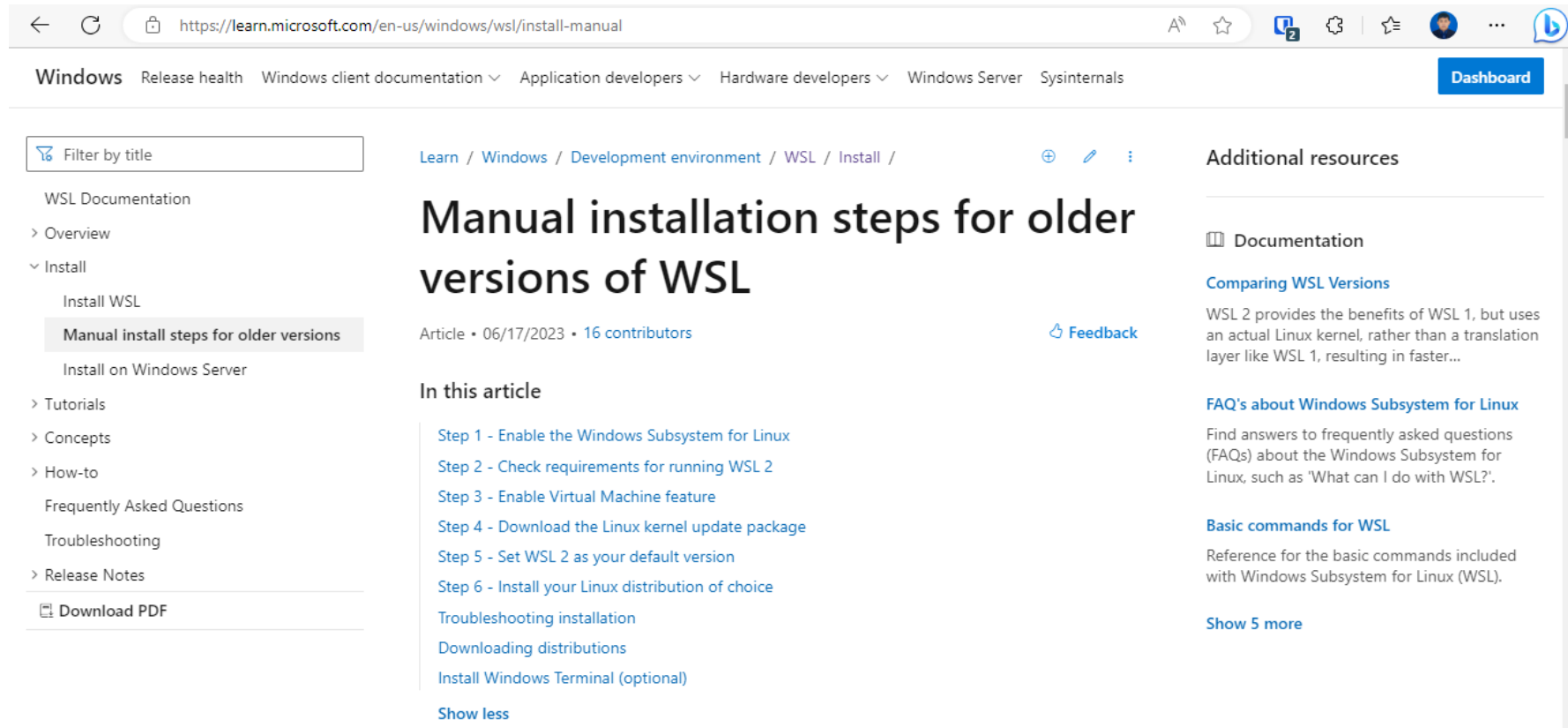
WSL2 เป็นเทคโนโลยีที่ออกแบบมาเพื่อเป็นช่วงส่วนกลางที่ทำให้สามารถรันระบบปฏิบัติการ Linux บนระบบปฏิบัติการ Windows ได้

WSL2 เป็นการพัฒนาของ Microsoft ที่มีเป้าหมายในการเพิ่มประสิทธิภาพและความเข้ากันได้ระดับสูงของ WSL ซึ่งเป็นเวอร์ชันก่อนหน้านี้ โดย WSL2 ใช้เทคโนโลยีการจำลองสถาปัตยกรรมหนึ่งเพื่อสร้างเคอร์เนล Linux แยกออกมาตัวเองที่ทำงานบน Windows ในลักษณะเป็นระบบปฏิบัติการเสมือน (virtualized) นั้นหมายความว่า WSL2 ทำงานบนระบบปฏิบัติการ Windows แต่ให้บริการและรันโปรแกรม Linux ในพื้นที่เมีนูล์ของตัวเอง

การใช้งาน WSL2 ช่วยให้ผู้ใช้งานสามารถเรียกใช้คำสั่ง Linux และรันแอปพลิเคชัน Linux ได้โดยตรงในเครื่อง Windows โดยไม่ต้องติดตั้งเครื่องมือสำหรับจำลองหรือตั้งค่าเซิร์ฟเวอร์ Linux แยกต่างหาก ผู้ใช้งานสามารถเรียกใช้คำสั่งที่รู้จักของ Linux, ติดตั้งและใช้งานซอฟต์แวร์ Linux, และทำงานกับไฟล์และไดเรกทอรีของ Linux ได้ในระบบไฟล์ของ Windows

การติดตั้ง Windows Subsystem for Linux 2 (WSL2)

การเตรียมความพร้อมสำหรับการติดตั้ง WSL2 สามารถเข้าทำการติดตั้งได้จาก
<https://learn.microsoft.com/en-us/windows/wsl/install-manual>



The screenshot shows the Microsoft Learn website page for "Manual installation steps for older versions of WSL". The page is viewed in a web browser with the URL <https://learn.microsoft.com/en-us/windows/wsl/install-manual>. The page layout includes a top navigation bar with links like "Windows", "Release health", and "Windows client documentation". A left sidebar contains a "Filter by title" search box and a list of navigation items: "WSL Documentation", "Overview", "Install" (expanded), "Install WSL" (selected), "Manual install steps for older versions" (highlighted), "Install on Windows Server", "Tutorials", "Concepts", "How-to", "Frequently Asked Questions", "Troubleshooting", "Release Notes", and "Download PDF". The main content area has the title "Manual installation steps for older versions of WSL" and a sub-header "Article • 06/17/2023 • 16 contributors". Below the title is a section "In this article" with a list of steps: "Step 1 - Enable the Windows Subsystem for Linux", "Step 2 - Check requirements for running WSL 2", "Step 3 - Enable Virtual Machine feature", "Step 4 - Download the Linux kernel update package", "Step 5 - Set WSL 2 as your default version", "Step 6 - Install your Linux distribution of choice", "Troubleshooting installation", "Downloading distributions", and "Install Windows Terminal (optional)". A "Show less" link is at the bottom of this list. On the right side, there is an "Additional resources" section with links to "Documentation", "Comparing WSL Versions", "FAQ's about Windows Subsystem for Linux", and "Basic commands for WSL".

วิธีการติดตั้ง WSL 2 ประกอบด้วยขั้นตอนทั้งหมด 6 ขั้นตอนดังนี้

ขั้นตอนที่ 1: เปิดใช้งาน Windows Subsystem for Linux (WSL) ใน Windows 10

เปิดหน้าต่าง PowerShell หรือ Command Prompt ด้วยสิทธิ์ Administrator และใช้คำสั่งต่อไปนี้

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

ขั้นตอนที่ 2: ตรวจสอบความต้องการสำหรับการใช้งาน WSL 2

ต้องใช้ Windows 10 เวอร์ชัน 1903 (build 18362) หรือใหม่กว่าและต้องใช้ระบบปฏิบัติการที่เปิดใช้งาน Hyper-V ดังนั้นให้ตรวจสอบให้แน่ใจว่ามีระบบปฏิบัติการและคอมพิวเตอร์ที่สามารถรัน WSL 2 ได้

ขั้นตอนที่ 3: เปิดใช้งานคุณสมบัติ Virtual Machine เปิดหน้าต่าง PowerShell หรือ Command Prompt ด้วยสิทธิ์ Administrator และใช้คำสั่งต่อไปนี้:

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

เมื่อคำสั่งดังกล่าวทำงานเสร็จ ให้ทำการรีสตาร์ทคอมพิวเตอร์อีกครั้ง

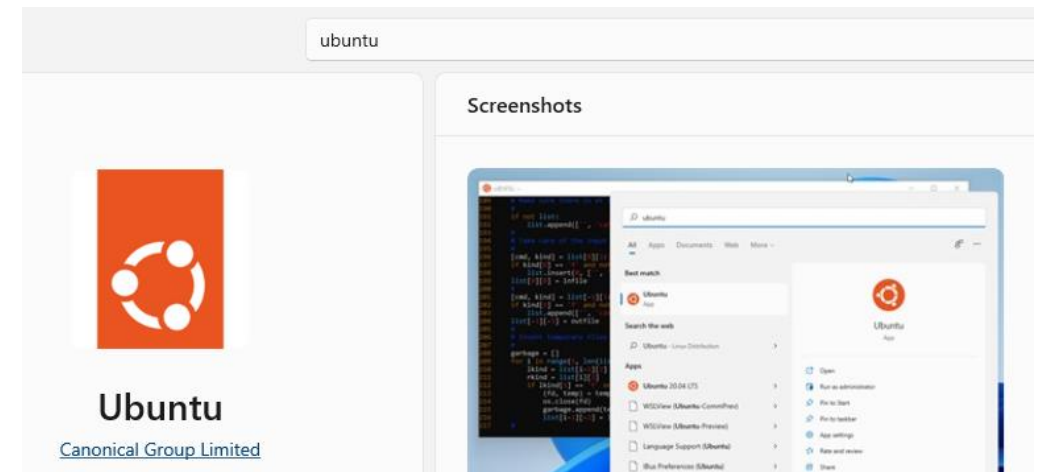
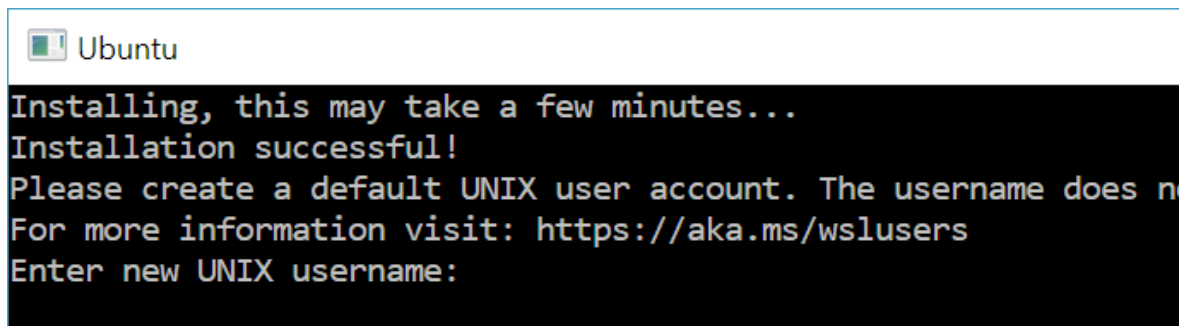
ขั้นตอนที่ 4: ดาวน์โหลดแพคเกจอัปเดต Linux kernel โดยไปที่เว็บไซต์การดาวน์โหลดของ Microsoft เพื่อรับแพคเกจอัปเดต Linux kernel สำหรับ WSL 2 และคลิกที่ลิงก์ดาวน์โหลด

https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi

ขั้นตอนที่ 5: ตั้งค่า WSL 2 เป็นเวอร์ชันเริ่มต้น เปิดหน้าต่าง PowerShell หรือ Command Prompt และใช้คำสั่งนี้เพื่อตั้งค่า WSL 2 เป็นเวอร์ชันเริ่มต้น:

```
wsl --set-default-version 2
```

ขั้นตอนที่ 6: ติดตั้งเวอร์ชันของ Linux ที่คุณต้องการ เปิด Microsoft Store และค้นหาเวอร์ชันของ Linux ที่ต้องการติดตั้ง เมื่อค้นพบแอปพลิเคชัน Linux ที่ต้องการให้คลิกที่นั้นและติดตั้งเพื่อเริ่มกระบวนการติดตั้ง เลือก Ubuntu



เสร็จขั้นตอนการติดตั้ง Ubuntu บน Windows

Linux



Linux เป็นระบบปฏิบัติการ (Operating System) ที่สร้างขึ้นโดย Linus Torvalds ในปี 1991 และได้รับการพัฒนาและส่งเสริมให้เติบโตต่อมาโดยชุมชนนักพัฒนาที่มีส่วนร่วมมากมายต่อเนื่อง (Open Source Community) ระบบปฏิบัติการนี้เป็นตัวกลางที่ทำงานร่วมกับฮาร์ดแวร์และซอฟต์แวร์ต่าง ๆ บนเครื่องคอมพิวเตอร์

Linux เน้นความเสถียรและความประสิทธิภาพ มีลักษณะการทำงานแบบหลายภารกิจ (Multitasking) และสามารถให้บริการในรูปแบบหลายผู้ใช้ (Multiuser) โดยมีความสามารถในการจัดการและประมวลผลคำสั่งในรูปแบบเส้นคำสั่ง (Command Line Interface) และเมนูกราฟิก (Graphical User Interface) ที่ให้ควบคุมและบริหารจัดการระบบได้อย่างสะดวกสบาย

การพัฒนา Linux ได้ใช้ภาษาโปรแกรมที่เปิดเผยแพร่รหัสต้นฉบับ (Open Source) ทำให้นักพัฒนาทั่วโลกสามารถศึกษาและพัฒนาเพิ่มเติมได้ ชุมชน Open Source ได้เปิดโอกาสให้นักพัฒนาเข้าร่วมพัฒนาต่อยอด และนำมาใช้กันหลากหลายสถานการณ์ ไม่ว่าจะเป็นบนคอมพิวเตอร์ส่วนบุคคล เครื่องเซิร์ฟเวอร์ อุปกรณ์ IoT (Internet of Things) และอื่น ๆ ที่ต้องการระบบปฏิบัติการที่เสถียรและประสิทธิภาพสูง

พื้นฐานของเชลล์ (Shell) บน Ubuntu

```
natta@DESKTOP-J8KEBA2:~$ ls  
docker  
natta@DESKTOP-J8KEBA2:~$ pwd  
/home/natta  
natta@DESKTOP-J8KEBA2:~$
```

Shell คืออะไร?

ในระบบปฏิบัติการ (Operating System) ที่ใช้ระบบ Linux เช่น Ubuntu, Shell คือโปรแกรมหรืออินเตอร์เฟซที่ทำหน้าที่เป็นตัวกลางในการเชื่อมต่อระหว่างผู้ใช้ (User) กับระบบปฏิบัติการ (Operating System) และแอปพลิเคชันภายในระบบปฏิบัติการนั้น ๆ โดยใช้คำสั่งหรือคำสั่งเรียกใช้งาน (Command) ที่ผู้ใช้พิมพ์เข้าไปใน Shell แล้วระบบปฏิบัติการจะทำการประมวลผลและดำเนินการตามคำสั่งนั้น ๆ

คำสั่งพื้นฐานใน Shell

- cd: เปลี่ยนไดเรกทอรี (Change Directory)
- ls: แสดงรายชื่อไฟล์และไดเรกทอรีในตำแหน่งปัจจุบัน (List)
- pwd: แสดงตำแหน่งปัจจุบัน (Print Working Directory)
- mkdir: สร้างไดเรกทอรีใหม่ (Make Directory)
- rm: ลบไฟล์หรือไดเรกทอรี (Remove)
- cp: คัดลอกไฟล์หรือไดเรกทอรี (Copy)
- mv: ย้าย (หรือเปลี่ยนชื่อ) ไฟล์หรือไดเรกทอรี (Move)

ตัวอย่างการใช้คำสั่งใน Shell

- การเปลี่ยนไดเรกทอรี: `cd /home/user/documents`
- การแสดงรายชื่อไฟล์และไดเรกทอรี: `ls`
- แสดงตำแหน่งปัจจุบัน: `pwd`
- การสร้างไดเรกทอรี: `mkdir project`
- การลบไฟล์: `rm file.txt`
- การคัดลอกไฟล์: `cp file.txt /backup`
- การย้ายไดเรกทอรี: `mv folder /home/user/documents/projects`

การแสดงรายละเอียดของไฟล์และไดเรกทอรี

ls -l: แสดงรายละเอียดของไฟล์และไดเรกทอรีทั้งหมด

ls -a: แสดงไฟล์และไดเรกทอรีทั้งหมดรวมถึงไฟล์ที่ซ่อน (Hidden)

การจัดการไฟล์ข้อความ

- touch: สร้างไฟล์เปล่า
- chown: เปลี่ยนเจ้าของไฟล์
- nano: แก้ไขไฟล์ข้อความด้วย Text Editor Nano
- chmod: เปลี่ยนสิทธิ์การเข้าถึงไฟล์
- cat: อ่านเนื้อหาของไฟล์ข้อความ

ตัวอย่างการใช้คำสั่งในการจัดการไฟล์และไดเรกทอรี

- แสดงรายละเอียดของไฟล์และไดเรกทอรีทั้งหมด: `ls -l`
- สร้างไฟล์เปล่า: `touch newfile.txt`
- แก้ไขไฟล์ด้วย Nano: `nano newfile.txt`
- อ่านเนื้อหาของไฟล์: `cat newfile.txt`
- เปลี่ยนสิทธิ์การเข้าถึงไฟล์: `chmod 644 newfile.txt`
- เปลี่ยนเจ้าของไฟล์: `chown user newfile.txt`

การใช้งาน Package Manager (apt)

Package Manager คืออะไร?

Package Manager เป็นเครื่องมือที่ใช้ในระบบปฏิบัติการเพื่อจัดการและติดตั้งแพ็คเกจซอฟต์แวร์ (Software Package) ซึ่งเป็นชุดของโปรแกรมหรือแอปพลิเคชันที่ถูกห่อหุ้มเข้าไปในแพ็คเกจเดียวกัน เพื่อให้ง่ายต่อการติดตั้ง ดังนั้นผู้ใช้สามารถใช้ Package Manager ในการติดตั้งและลบแพ็คเกจซอฟต์แวร์ได้โดยง่ายและสะดวกสบายมากยิ่งขึ้น

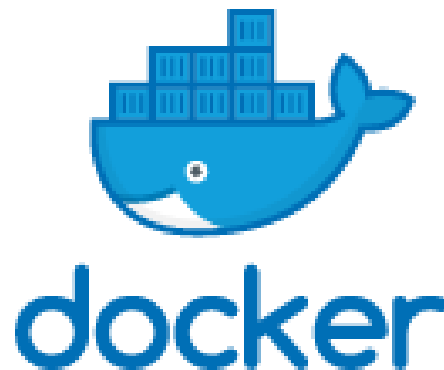
คำสั่ง apt: การติดตั้งและจัดการแพ็คเกจซอฟต์แวร์

- ติดตั้งแพ็คเกจ: `sudo apt-get install package_name`
- อัปเดตรายชื่อแพ็คเกจ: `sudo apt-get update`
- อัปเดตแพ็คเกจที่ติดตั้ง: `sudo apt-get upgrade`
- ค้นหาแพ็คเกจ: `apt-cache search keyword`

คำสั่ง apt-add-repository: เพิ่ม Repository เพื่อติดตั้งแพ็คเกจจากแหล่งที่มาที่ไม่ได้รวมมาในรายการประจำ

- เพิ่ม Repository: `sudo apt-add-repository repository_url`
- อัปเดตรายชื่อแพ็คเกจหลังจากเพิ่ม Repository: `sudo apt-get update`

Docker



Docker เป็นแพลตฟอร์มสำหรับการจัดการและปรับใช้แอปพลิเคชันที่อยู่ในสภาพแวดล้อมที่เรียกว่า "คอนเทนเนอร์" (Containers) ซึ่งเป็นเทคโนโลยีที่ช่วยให้สามารถแพ็คเกจแอปพลิเคชันและส่วนประกอบต่าง ๆ ที่เกี่ยวข้องไว้ในคอนเทนเนอร์เดียวกันได้ ทำให้ง่ายต่อการโอนย้ายและเปิดใช้งานแอปพลิเคชันในสภาพแวดล้อมที่ต่างกันได้อย่างยืดหยุ่นและมีประสิทธิภาพมากขึ้น นอกจากนี้ Docker ยังมีเครื่องมือที่ช่วยในการจัดการคอนเทนเนอร์ เช่น Docker Compose เพื่อการจัดการและกำหนดคอนเทนเนอร์และบริการที่เกี่ยวข้องอื่น ๆ

การติดตั้ง Docker Community Edition (Docker CE) ใน Ubuntu ต้องทำตามขั้นตอนดังนี้

1. อัปเดตรายการแพ็คเกจที่มีอยู่ของคุณโดยใช้คำสั่ง:

```
sudo apt update
```

2. ติดตั้งแพ็คเกจบางตัวที่จำเป็นให้ apt สามารถใช้แพ็คเกจผ่าน HTTPS ได้:

```
sudo apt install -y apt-transport-https ca-certificates curl software-properties-common
```

3. เพิ่มคีย์ GPG สำหรับที่เก็บ Docker อย่างเป็นทางการเข้าสู่ระบบของคุณ:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4. เพิ่มที่เก็บ Docker ลงใน APT sources:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

5. ติดตั้ง Docker ในรุ่นล่าสุดที่มีให้:

```
sudo apt install -y docker-ce
```


หากต้องการหลีกเลี่ยงการพิมพ์ sudo เมื่อรันคำสั่ง Docker ให้เพิ่มชื่อผู้ใช้ของคุณในกลุ่ม Docker:

```
sudo usermod -aG docker ${USER}
```

```
su - ${USER}
```

```
sudo usermod -aG docker username
```

จากนั้นเขียนคำสั่งให้ Docker เริ่มทำงาน (จะต้องพิมพ์ทุกครั้งเมื่อเริ่มการทำงานของคอมพิวเตอร์)

```
sudo service docker start
```

หากไม่สามารถเริ่มทำงานได้ให้ตั้งค่า iptables ก่อน โดยพิมพ์คำสั่ง

```
sudo update-alternative --config iptables
```

จากนั้นเลือก /usr/sbin/iptables-legacy 10 manual mode โดยการกดเลข 1 แล้ว Enter

```
Processing triggers for libc-bin (2.33-0ubuntu3.1) ...
natta@DESKTOP-J8KEBA2:~$ sudo update-alternatives --config iptables
There are 2 choices for the alternative iptables (providing /usr/sbin/iptables):

  Selection    Path                                Priority  Status
  -----
*  0            /usr/sbin/iptables-nft              20      auto mode
    1            /usr/sbin/iptables-legacy           10      manual mode
    2            /usr/sbin/iptables-nft              20      manual mode

Press <enter> to keep the current choice[*], or type selection number: 2
```

ในการใช้งาน Docker จะใช้เพื่อสร้างการทำงานของ Service ของโปรแกรมต่าง ๆ
เกี่ยวกับ IoT ซึ่งในหน่วยเรียนนี้ ประกอบไปด้วย

- Node-Red
- MQTT
- Influxdb
- Grafana

Clone โปรเจคจากผู้สอนมาดำเนินงานโดยพิมพ์คำสั่ง

```
git clone https://github.com/woeis-me/docker.git
```

The screenshot shows the GitHub interface for the repository 'docker' by user 'woeis-me'. The repository is public and has 1 branch and 0 tags. The 'Code' button is highlighted, and a dropdown menu is open showing the 'Clone' option. The 'HTTPS' URL is displayed as 'https://github.com/woeis-me/docker.git', which is highlighted with a red box. A red arrow points to the copy icon next to the URL. Below the repository list, a terminal window shows the command 'git clone https://github.com/woeis-me/docker.git' being executed, with the output showing the cloning progress and completion.

```
natta@DESKTOP-J8KEBA2:~$ git clone https://github.com/woeis-me/docker.git
Cloning into 'docker'...
remote: Enumerating objects: 2794, done.
remote: Counting objects: 100% (2794/2794), done.
remote: Compressing objects: 100% (1978/1978), done.
remote: Total 2794 (delta 609), reused 2748 (delta 567), pack-reused 0
Receiving objects: 100% (2794/2794), 17.06 MiB | 7.15 MiB/s, done.
Resolving deltas: 100% (609/609), done.
```

พิมพ์คำสั่งย้ายไปยัง Directory docker

```
cd docker
```

จากนั้นพิมพ์คำสั่งเริ่มทำงานของ docker แล้วรอจนเสร็จ

```
docker compose up -d --build
```

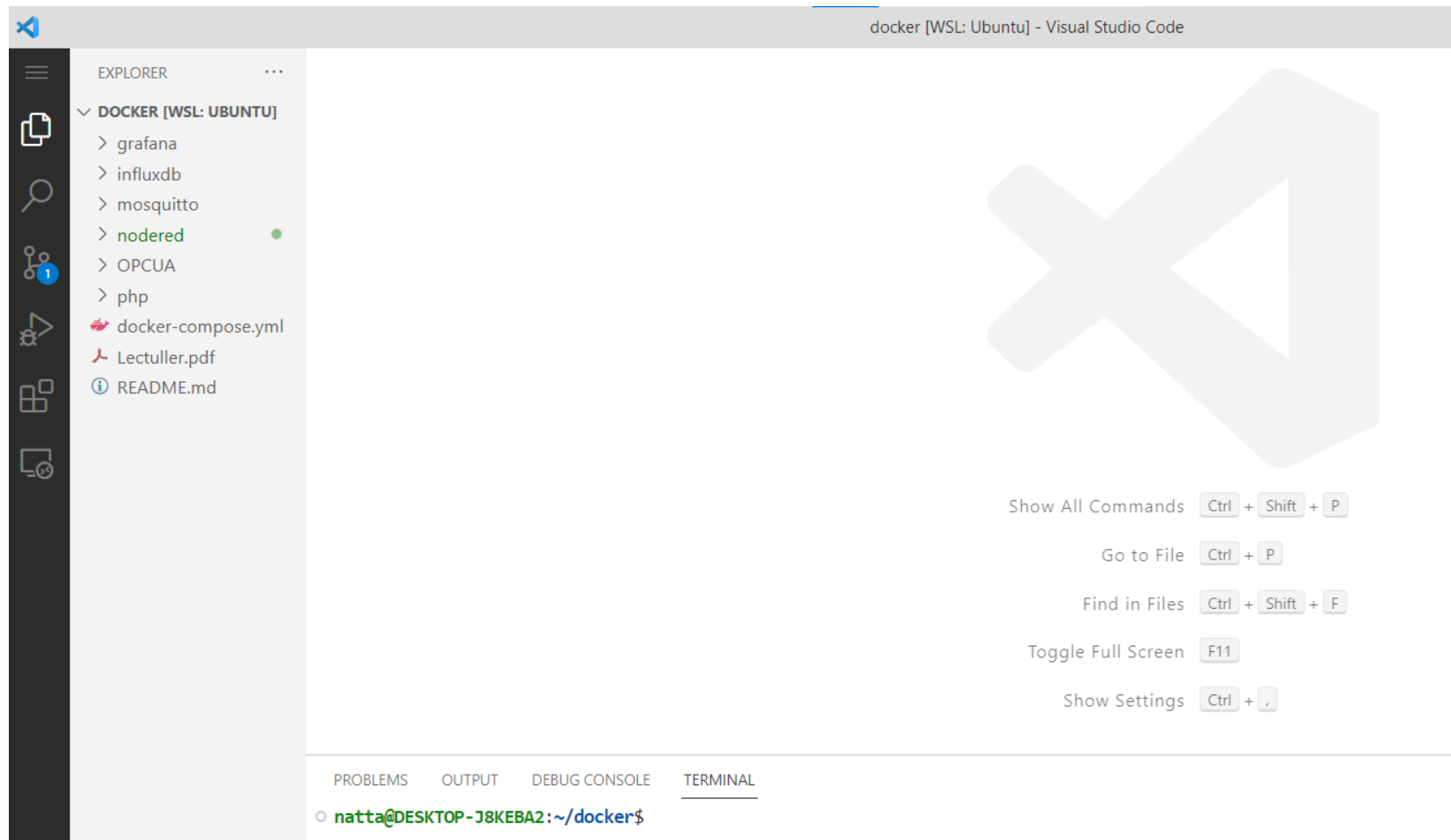
```
natta@DESKTOP-J8KEBA2:~$ cd docker
natta@DESKTOP-J8KEBA2:~/docker$ docker compose up -d --build
[+] Running 0/6
  ⬢ db Pulling
  ⬢ adminer Pulling
  ⬢ phpmyadmin Pulling
  ⬢ mosquitto Pulling
  ⬢ influxdb Pulling
  ⬢ nodered Pulling
```

```
⬢ Network docker_networkName Created
⬢ Volume "docker_grafana_data" Created
⬢ Container user-influxdb Started
⬢ Container user-db Started
⬢ Container user-adminer Started
⬢ Container user-mosquitto Started
⬢ Container user-php-apache Started
⬢ Container user-grafana Started
⬢ Container user-myadmin Started
⬢ Container user-nodered Started
natta@DESKTOP-J8KEBA2:~/docker$
```

เสร็จขั้นตอนการรันโปรแกรม docker

พิมพ์คำสั่งเปิดโปรแกรม visual studio code

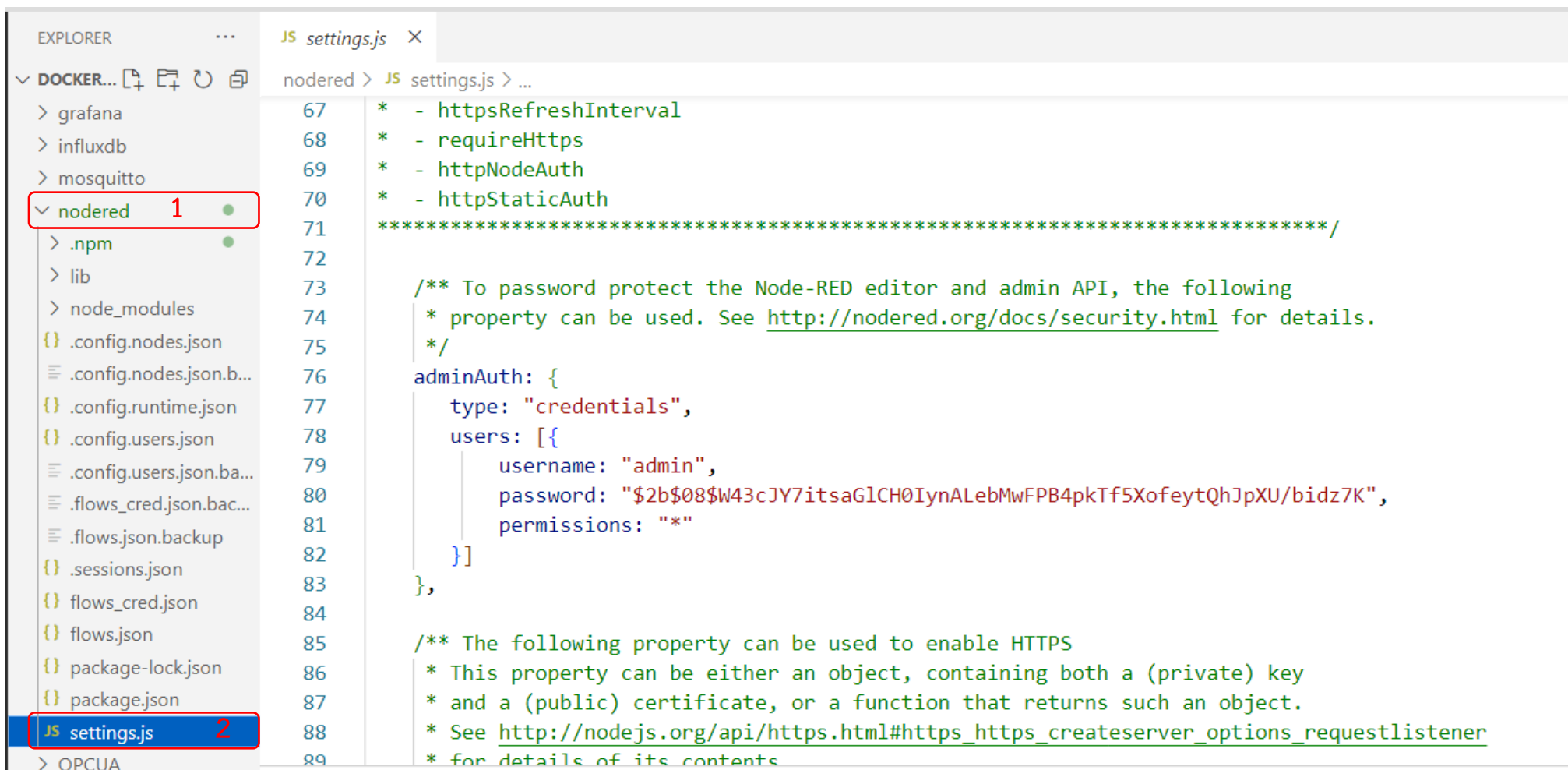
code .



เมื่อเปิดโปรแกรม visual studio code เสร็จ ขั้นตอนต่อไปคือการตั้งค่า Service แต่ละตัว เพื่อเข้ารหัส

Node-Red

1. เปิดไฟล์ settings.js ใน nodered/setting.js จากนั้นเลื่อนไฟล์ลงมาที่บรรทัด 76 หรือ ช่วงโปรแกรม adminAuth



```
EXPLORER
  > grafana
  > influxdb
  > mosquito
  ▼ nodered 1
    > .npm
    > lib
    > node_modules
    {} .config.nodes.json
    {} .config.nodes.json.b...
    {} .config.runtime.json
    {} .config.users.json
    {} .config.users.json.ba...
    {} .flows_cred.json.bac...
    {} .flows.json.backup
    {} .sessions.json
    {} flows_cred.json
    {} flows.json
    {} package-lock.json
    {} package.json
    JS settings.js 2
    > OPCUA

JS settings.js
nodered > JS settings.js > ...
67 * - httpsRefreshInterval
68 * - requireHttps
69 * - httpNodeAuth
70 * - httpStaticAuth
71 *****/
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: "$2b$08$W43cJY7itsaGlCH0IynALebMwFPB4pkTf5XofeytQhJpXU/bidz7K",
81     permissions: "*"
82   }]
83 },
84
85 /** The following property can be used to enable HTTPS
86  * This property can be either an object, containing both a (private) key
87  * and a (public) certificate, or a function that returns such an object.
88  * See http://nodejs.org/api/https.html#https\_https\_createserver\_options\_requestlistener
89  * for details of its contents
```

2. พิมพ์คำสั่งเพื่อสร้างรหัสผ่านในกับ Node-Red

```
docker exec -it user-nodered sh
```

```
npx node-red admin hash-pw
```

```
exit
```

จากนั้นกรอกรหัสผ่าน (ขณะกรอกจะไม่แสดงค่า) และกด Enter จะได้รหัสผ่านที่เข้ารหัส จากนั้นคัดลอกรหัสผ่านไป
กรอกในไฟล์ settings.js แล้วเซฟไฟล์

1

```
natta@DESKTOP-J8KEBA2:~/docker$ docker exec -it user-nodered sh
~ $ npx node-red admin hash-pw
Password:
$2b$08$/cPfLivtwLBTNnFUD.6m7.k1NmE7QGebUlwGMOT4j6f3kMnWxoFHu
```

```
76   adminAuth: {
77     type: "credentials",
78     users: [{
79       username: "admin",
80       password: "$2b$08$/cPfLivtwLBTNnFUD.6m7.k1NmE7QGebUlwGMOT4j6f3kMnWxoFHu",
81       permissions: "*"
82     }]
83   },
84 }
```

2

MQTT

สร้างไฟล์ password_file ใน mosquitto/config จากนั้นพิมพ์คำสั่ง

```
docker exec -it user-mosquitto sh
```

```
mosquitto_passwd -b /mosquitto/config/password_file user pass
```

```
exit
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

○ natta@DESKTOP-J8KEBA2:~/docker$ docker exec -it user-mosquitto sh
/ # mosquitto_passwd -b /mosquitto/config/password_file admin [REDACTED]
/ # █

EXPLORER  ...
▼ DOCKER [WSL: UBUNTU]
  > grafana
  > influxdb

password_file U X
mosquitto > config > password_file
1 admin:$6$6i1LWG9vvNRiFpGF$H4IheMZsDxRLlXKvNLEF5bPkHuh2jwwp9OC8yKMMXxx
2 █
```

Influxdb

1. พิมพ์คำสั่งดังต่อไปนี้

```
docker exec -it user-influxdb sh
```

```
influx
```

```
CREATE USER "admin" WITH PASSWORD 'admin_passwd' WITH ALL PRIVILEGES
```

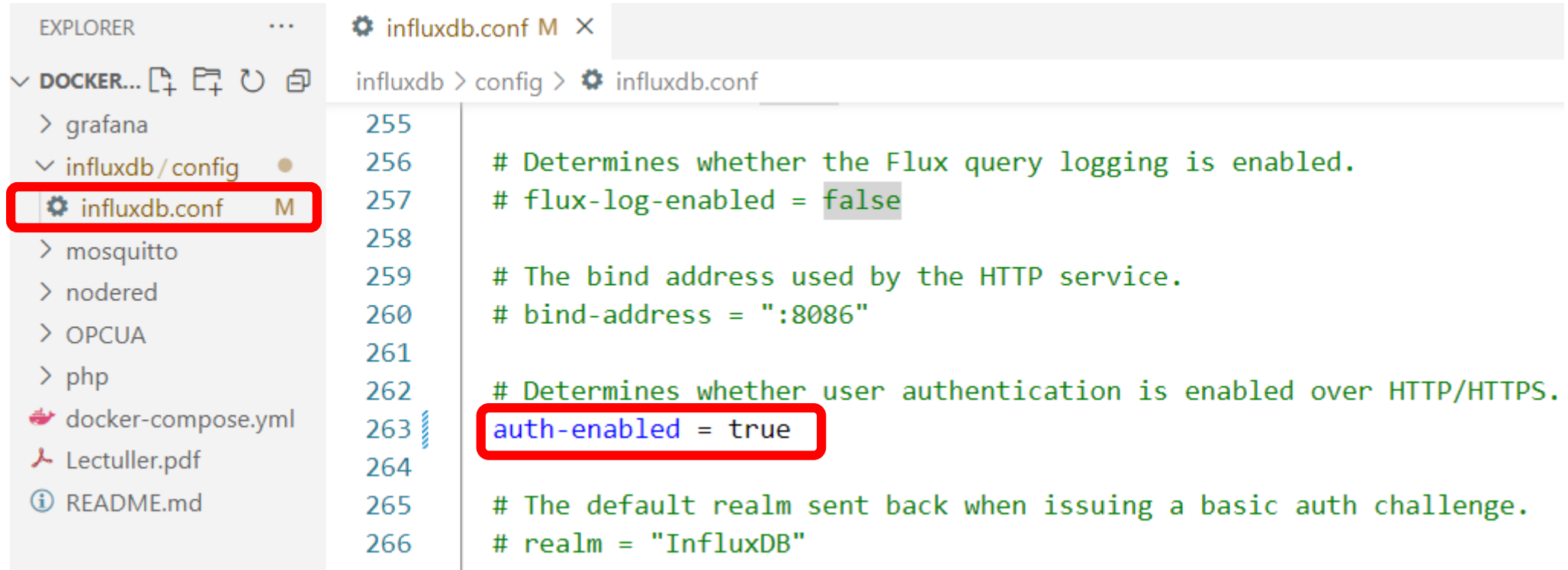
```
CREATE DATABASE db_name
```

```
exit
```

```
exit
```

```
natta@DESKTOP-J8KEBA2:~/docker$ docker exec -it user-influxdb sh
# influx
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
> CREATE USER "admin" WITH PASSWORD 'Nattawat034p' WITH ALL PRIVILEGES
> create database mydb
> exit
# exit
```

2. แก้ไขไฟล์ influxdb.conf ที่บรรทัด 263 หรือ #auth-enabled = true โดยลบ # แล้วเซฟไฟล์

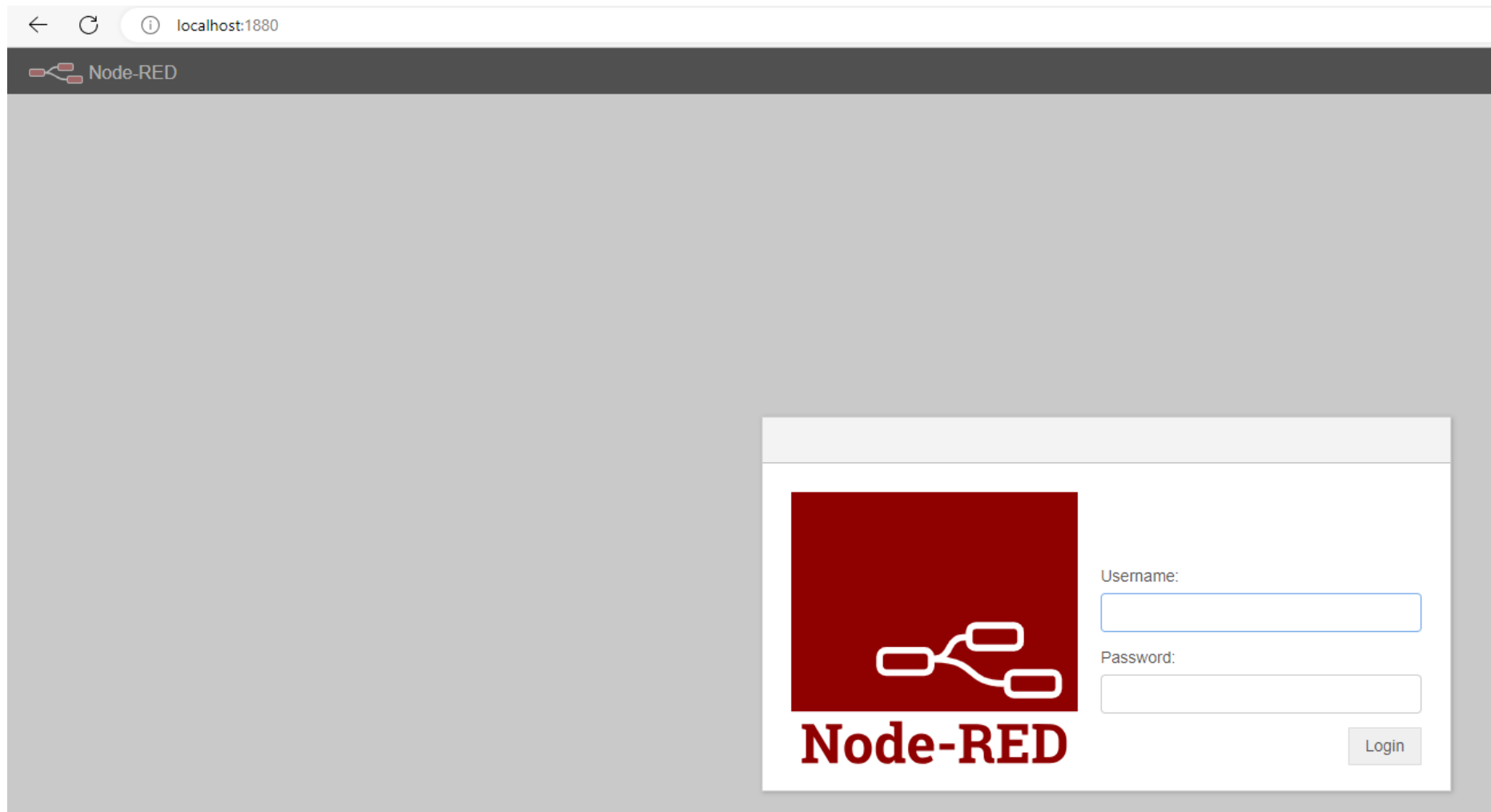


The screenshot shows a code editor with the file explorer on the left and the code editor on the right. The file explorer shows a directory structure with 'influxdb/config' expanded, and 'influxdb.conf' selected. The code editor shows the contents of 'influxdb.conf' with line numbers 255 to 266. The line '# auth-enabled = false' is highlighted with a red box, and the text 'auth-enabled = true' is written over it, also highlighted with a red box.

```
255  
256 # Determines whether the Flux query logging is enabled.  
257 # flux-log-enabled = false  
258  
259 # The bind address used by the HTTP service.  
260 # bind-address = ":8086"  
261  
262 # Determines whether user authentication is enabled over HTTP/HTTPS.  
263 auth-enabled = true  
264  
265 # The default realm sent back when issuing a basic auth challenge.  
266 # realm = "InfluxDB"  
---
```

เริ่มเข้าหน้าเว็บของ Node-Red โดยเข้า Link บนเว็บเบราว์เซอร์

Node-Red : localhost:1880

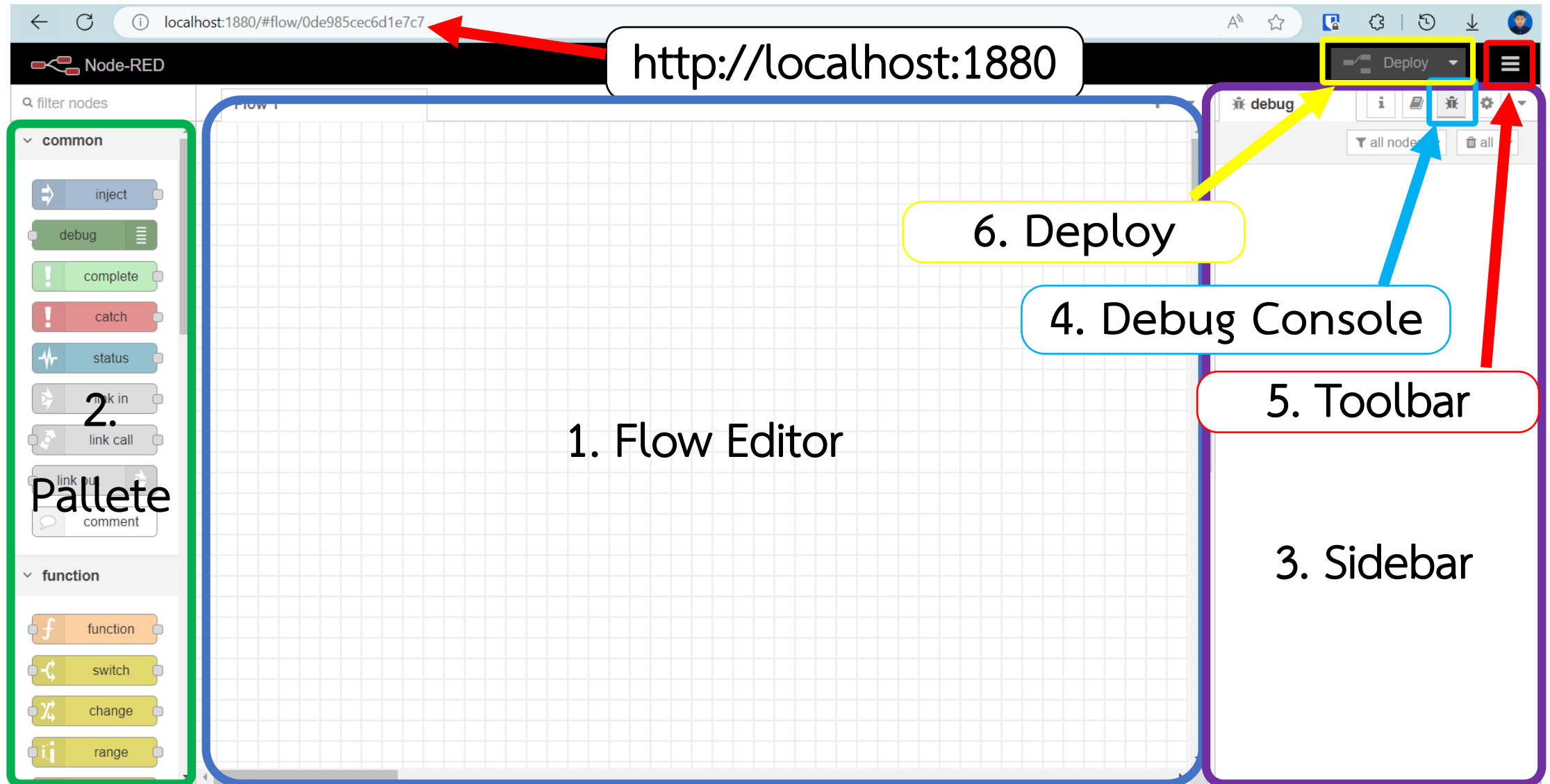


Node-Red

Node-Red เป็นแพลตฟอร์มแบบเบสออนไลน์เปิดต้นฉบับที่ใช้สำหรับสร้างและจัดการกระบวนการอัตโนมัติที่เชื่อมต่อกับอุปกรณ์และบริการต่างๆ ส่วนใหญ่ใช้งานในการสร้างและควบคุมอินเทอร์เน็ตของสร้างของอุปกรณ์อิเล็กทรอนิกส์ (IoT) โดยใช้กราฟสร้างเส้นทางที่เรียกว่า "flow" ซึ่งประกอบด้วยโหนด (nodes) และการเชื่อมต่อกันเพื่อให้ข้อมูลไหลผ่านกัน

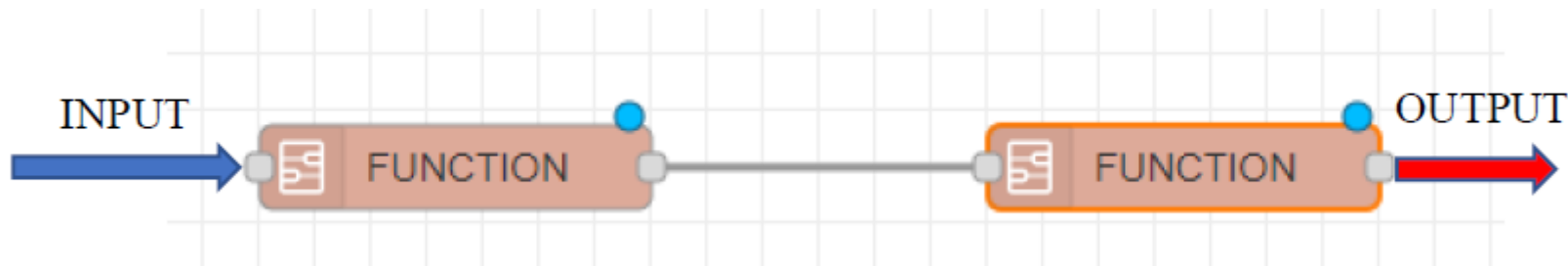
หน้าที่หลักของ Node-Red คือให้ผู้ใช้งานสร้างและจัดการกราฟแพลตฟอร์มการประมวลผลข้อมูลต่าง ๆ ในรูปแบบของโหนด โดยทำหน้าที่เชื่อมต่อและประมวลผลข้อมูลในรูปแบบของแพลตฟอร์มสตรีมข้อมูล (streaming platform) โดยรองรับการสร้างและปรับแต่งโหนดต่าง ๆ เพื่อให้เหมาะสมกับงานที่ต้องการทำในโครงการแต่ละรายการ

ส่วนประกอบของ Node-Red



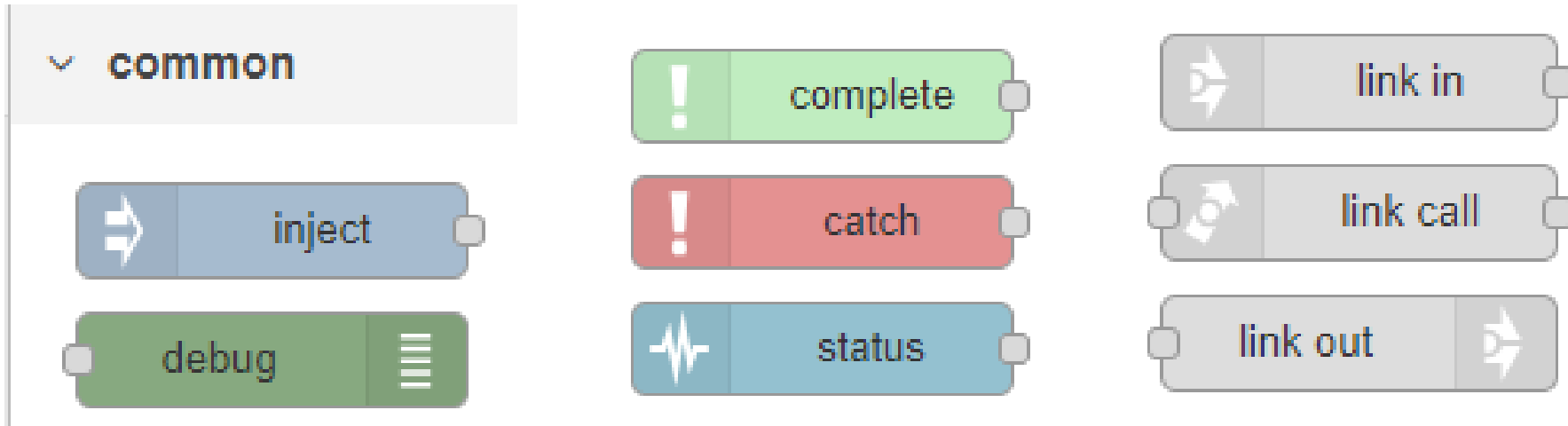
การใช้งาน Node-Red

Node ใน Node-RED มีการทำงานรับข้อมูลเข้า (Input), ประมวลผลข้อมูล (Function), และส่งข้อมูลออก (Output) โดยมีความหมายและภาระงานที่แตกต่างกัน เช่น HTTP Input Node ใช้รับคำขอ HTTP, Function Node ใช้เขียนโค้ด JavaScript, และ Debug Node ใช้แสดงข้อมูลผลลัพธ์ โหนดทำงานร่วมกันเพื่อสร้างกระบวนการและประมวลผลข้อมูลใน Node-RED ตามความต้องการของผู้ใช้



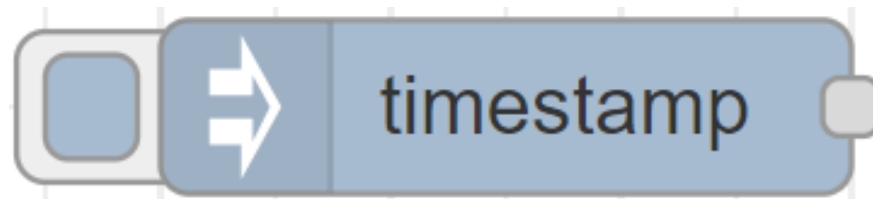
การใช้งานโหนดต่าง ๆ ใน Node-Red

Common



Common Node (โหนดทั่วไป) เป็นประเภทของโหนดที่มีการใช้งานทั่วไปมากที่สุด โดยเป็นโหนดที่มีความเกี่ยวข้องกับกระบวนการพื้นฐานของ Node-RED ทำหน้าที่ในการตั้งค่าและจัดการเส้นทางการไหลของข้อมูล ซึ่งประกอบด้วยโหนดต่าง ๆ ที่เชื่อมต่อกัน ในตัวอย่างนี้อาจมีโหนดเข้า (Input Nodes) เช่น MQTT, HTTP หรือ WebSocket, และโหนดออก (Output Nodes) เช่น Debug, HTTP Response ซึ่งจะช่วยกำหนดทิศทางการไหลของข้อมูลตามลำดับของโหนดนี้

1. Inject Node



Inject ใน Node-Red เป็นโหนดที่ใช้เป็นจุดเริ่มต้นของกระบวนการ โดยเป็นตัวแทนของเหตุการณ์ (event) หรือเหตุการณ์ที่เริ่มต้นกระบวนการทำงาน

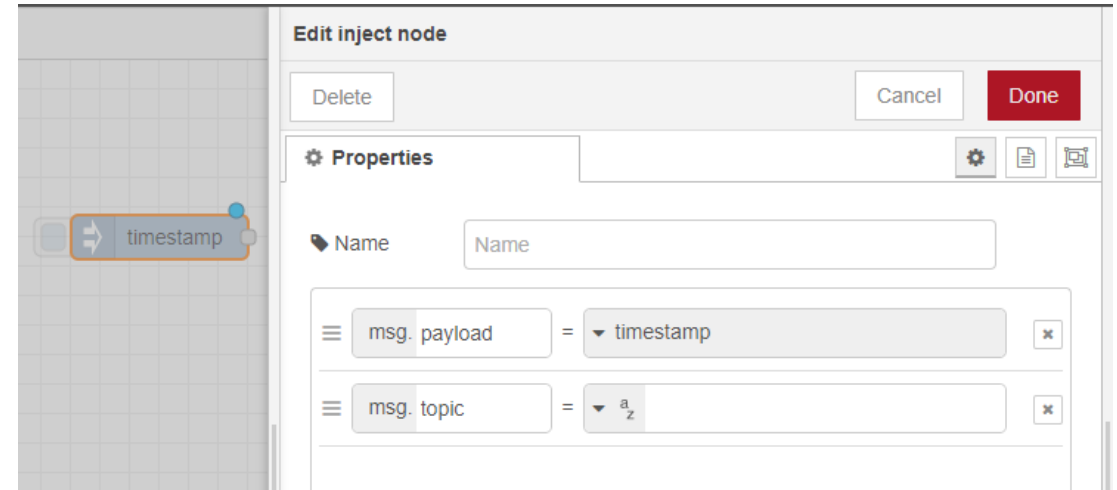
องค์ประกอบหลักของโหนด Inject ประกอบด้วย:

1. Timestamp (ประเภท: timestamp): เป็นข้อมูลเวลาที่เริ่มต้นกระบวนการ ค่านี้สามารถกำหนดได้ว่าเป็นเวลาปัจจุบันหรือเวลาที่กำหนดเอง

2. Payload (ประเภท: ข้อมูล): เป็นข้อมูลที่ส่งผ่านไปยังโหนดถัดไปในกราฟ ข้อมูลใน Payload สามารถเป็นประเภทต่างๆ เช่น ข้อความ (string), ตัวเลข (number), ออบเจกต์ (object) หรืออาร์เรย์ (array) ซึ่งขึ้นอยู่กับข้อกำหนดค่าในการใช้งาน

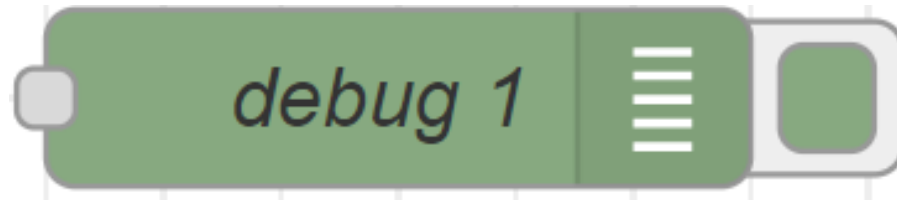
โหนด Inject สามารถใช้ในการเริ่มต้นกระบวนการในหลายวิธี

1. เริ่มต้นกระบวนการที่กำหนดเวลา: โหนด Inject สามารถกำหนดให้เริ่มต้นกระบวนการในเวลาที่กำหนด โดยการเลือกค่า Timestamp เป็นเวลาที่ต้องการให้เกิดเหตุการณ์
2. เริ่มต้นกระบวนการด้วยการกดปุ่ม: โหนด Inject ยังสามารถใช้เป็นปุ่มเริ่มต้นกระบวนการ สามารถกำหนดให้เหตุการณ์เกิดขึ้นเมื่อคลิกที่ปุ่ม Inject ในหน้าต่าง Node-Red
3. เริ่มต้นกระบวนการจากเหตุการณ์ภายนอก: สามารถใช้โหนด Inject เพื่อรอรับเหตุการณ์จากแหล่งข้อมูลภายนอก เช่น การรับข้อมูลจากเซ็นเซอร์หรือระบบอื่น ๆ และนำข้อมูลที่ได้รับมาเป็น Payload เพื่อส่งต่อไปยังโหนดอื่น ๆ ในกราฟ



โหนด Inject เป็นโหนดที่มีความสำคัญในการสร้างกราฟ Node-Red เนื่องจากมันเป็นจุดเริ่มต้นของกระบวนการและเหตุการณ์ที่เกิดขึ้นในโหนดอื่น ๆ สามารถตั้งค่าและใช้งานโหนด Inject ได้ตามความต้องการของโปรเจกต์

2. Debug Node



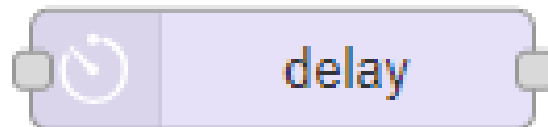
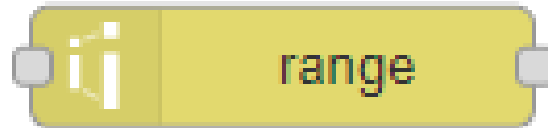
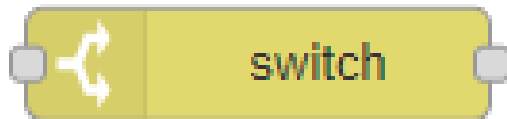
Debug เป็นกระบวนการการตรวจสอบและแก้ไขข้อผิดพลาดหรือปัญหาที่เกิดขึ้นในโปรแกรมหรือระบบที่กำลังทำงาน การ debug ช่วยให้นักพัฒนาซอฟต์แวร์สามารถวิเคราะห์และตรวจสอบข้อมูลที่เกี่ยวข้องกับข้อผิดพลาดเพื่อหาสาเหตุและแก้ไขได้

การ debug มักใช้เครื่องมือหรือโปรแกรมช่วย เช่น:

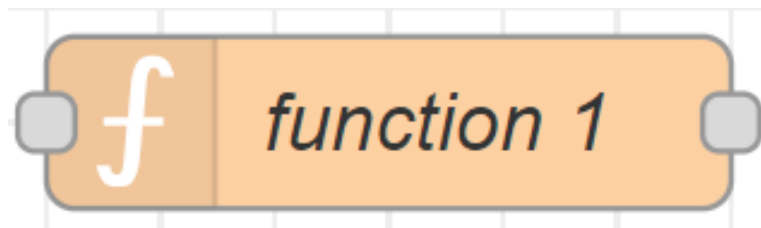
1. ตัวแปรการแสดงผล (print statement): การแสดงผลข้อมูลต่างๆ ในตำแหน่งที่สำคัญของโค้ด เช่น ค่าตัวแปร ข้อความสถานะ เพื่อตรวจสอบค่าและสถานะของโปรแกรมในขณะทำงาน
2. เครื่องมือ Debugging: เป็นโปรแกรมหรืออุปกรณ์ที่ให้ความสามารถในการตรวจสอบและแก้ไขข้อผิดพลาด มีฟังก์ชันต่างๆ เช่น การหยุดทำงานที่จุดหนึ่ง (breakpoint) เพื่อตรวจสอบค่าข้อมูล การดูค่าตัวแปร การติดตามการเข้าถึงฟังก์ชัน และอื่นๆ
3. การบันทึกข้อมูล (logging): การบันทึกข้อมูลเกี่ยวกับการทำงานของโปรแกรม รวมถึงข้อผิดพลาดที่เกิดขึ้น ไว้ในไฟล์หรือระบบบันทึกเพื่อวิเคราะห์หรือตรวจสอบภายหลัง
4. เครื่องมือตรวจสอบการทำงาน (debugger): เครื่องมือที่ช่วยให้นักพัฒนาสามารถติดตามกระบวนการทำงานของโปรแกรมได้ละเอียด รวมถึงเข้าถึงค่าตัวแปร การเรียกฟังก์ชัน และกระบวนการอื่นๆ ในระหว่างการทำงาน

Function

▼ function



1. Function Node



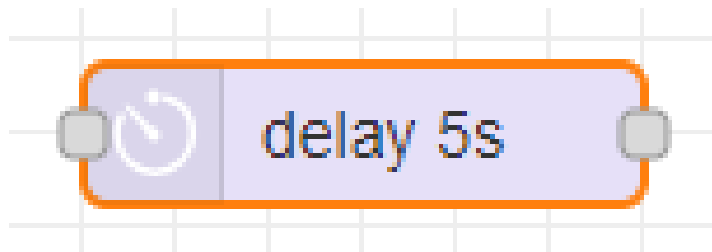
Function ใน Node-Red เป็นโหนดที่ใช้ในการเขียนโค้ดหรือสคริปต์ที่กำหนดเองเพื่อประมวลผลข้อมูลหรือกระบวนการที่ซับซ้อนกว่าที่โหนดอื่นๆ ใน Node-Red สามารถทำได้ โดยโหนดนี้ให้คุณเขียนโค้ด JavaScript ภายในตัวเองเพื่อประมวลผลข้อมูลที่เข้าสู่โหนดและส่งผลลัพธ์ออกไปยังโหนดถัดไปในกราฟ Node-Red

โหนด Function มีลักษณะเด่นต่อไปนี้:

1. ภาษาโปรแกรม: สามารถเขียนโค้ด JavaScript ภายในโหนด Function ซึ่งเป็นภาษาโปรแกรมที่กว้างขวางและมีความสามารถมากมาย เพื่อประมวลผลข้อมูลตามต้องการ
2. การประมวลผลเส้นทางที่ซับซ้อน: โหนด Function ช่วยให้สามารถดำเนินการประมวลผลที่ซับซ้อนและยืดหยุ่นขึ้นได้ โดยคุณสามารถเขียนโค้ดที่ต้องการในการประมวลผลข้อมูล การควบคุมการไหลของข้อมูล หรือการเรียกใช้งานบริการหรืออุปกรณ์ภายนอกได้
3. การเข้าถึงข้อมูล: โหนด Function สามารถเข้าถึงข้อมูลที่เข้าสู่โหนดได้ โดยข้อมูลนั้นจะถูกส่งผ่านเป็นพารามิเตอร์ในฟังก์ชัน JavaScript ภายในโหนด เช่น ข้อมูลจากโหนดที่เชื่อมต่อมา หรือข้อมูลที่ได้รับผ่านการสื่อสารกับอุปกรณ์หรือบริการอื่นๆ

โหนด Function เป็นอีกหนึ่งเครื่องมือที่ทรงพลังในการทำงาน โดยมีความยืดหยุ่นและความสามารถในการปรับแต่งการประมวลผลข้อมูลและการควบคุมไหลข้อมูลให้ตรงตามความต้องการของโปรเจกต์

2. Delay Node



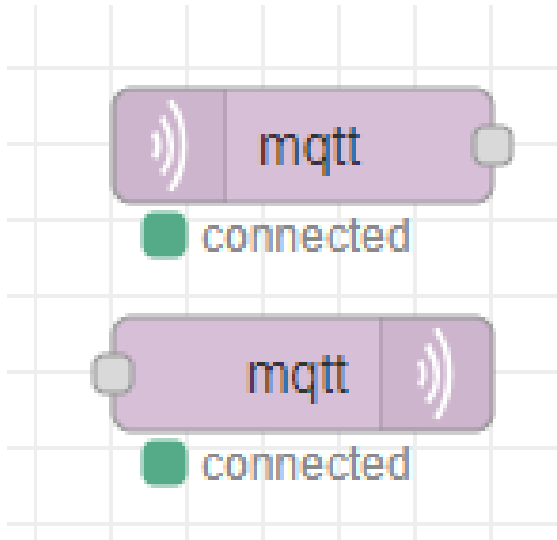
โหนด Delay ใน Node-RED เป็นอีกหนึ่งโหนดที่มีความสำคัญในการควบคุมและจัดการเวลาในกระบวนการของแพลตฟอร์มนี้ โหนดนี้ใช้สำหรับการทำให้กระบวนการหยุดชะงักหรือหน่วงเวลาก่อนที่จะส่งข้อมูลออกไปยังโหนดถัดไปในการไหลของข้อมูล ซึ่งจำเป็นต้องใช้เมื่อต้องการเสียเวลาหรือควบคุมการกระทำหลังจากเหตุการณ์เกิดขึ้น โดยค่าที่ตั้งค่าในโหนด Delay จะเป็นเวลาที่กำหนดให้หน่วงเวลา (เช่น 1 วินาที, 5 นาที) ก่อนที่จะส่งข้อมูลต่อไปยังโหนดถัดไปในกระบวนการ

Network



Network Node (โหนดเครือข่าย): เป็นประเภทของโหนดที่ใช้ในการเชื่อมต่อ Node-RED กับอุปกรณ์หรือระบบอื่นๆ ผ่านสถานะของเครือข่าย โดยสามารถทำการส่งข้อมูลออกไปยังเครือข่ายอื่นๆ หรือรับข้อมูลเข้ามาจากเครือข่ายนั้นๆ ตัวอย่างของโหนดเครือข่ายคือ MQTT, TCP, UDP ซึ่งช่วยให้ Node-RED สามารถสื่อสารและทำงานร่วมกับอุปกรณ์หรือระบบอื่นในเครือข่ายนั้นได้

1. MQTT Node

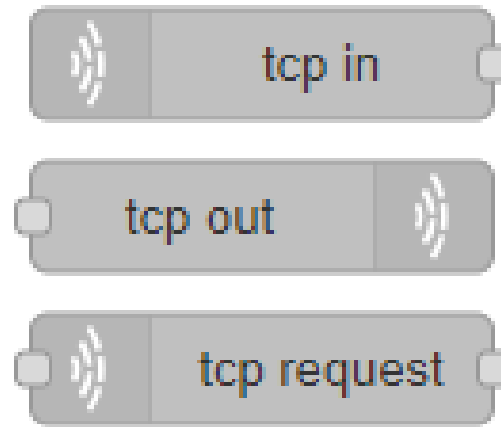


MQTT Node เป็น Node ที่ใช้สำหรับการเชื่อมต่อและการสื่อสารกับโครงสร้างระบบ MQTT (Message Queuing Telemetry Transport) ซึ่งเป็นโพรโทคอลการสื่อสารที่มีให้ใช้กันอย่างแพร่หลายในการเชื่อมต่อและแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์ในระบบอินเทอร์เน็ตของสรรพสิ่ง (Internet of Things - IoT) และระบบเครือข่ายอื่น ๆ

MQTT เป็นโพรโทคอลที่เบาที่สุดและง่ายต่อการใช้งานในการส่งข้อมูลระหว่างอุปกรณ์ และเหมาะสำหรับใช้งานในอุปกรณ์ที่มีข้อจำกัดทรัพยากรหรือแบตเตอรี่เล็กน้อย ในโครงสร้างระบบ MQTT จะมีสองส่วนหลักคือ Broker และ Client

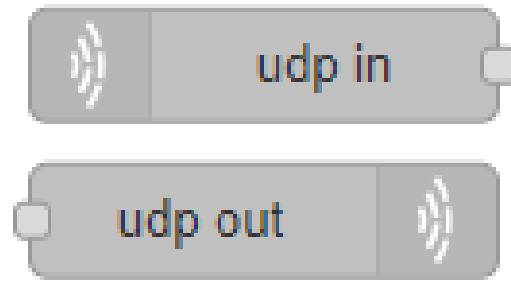
- Broker: เป็นตัวกลางที่ใช้ในการรับส่งข้อมูลระหว่าง Client ที่เชื่อมต่อเข้ามา ตัว Broker นั้นจะรับข้อมูลที่ส่งมาจาก Client และส่งต่อไปยัง Client อื่น ๆ ที่ต้องการรับข้อมูลนั้น ๆ
- Client: เป็นอุปกรณ์หรือแอปพลิเคชันที่เชื่อมต่อกับ Broker เพื่อส่งข้อมูลหรือรับข้อมูล โดย Client สามารถเป็นอุปกรณ์หรือแอปพลิเคชันที่ติดตั้งในอุปกรณ์เครือข่ายอื่น ๆ ที่เชื่อมต่อกับ Broker เดียวกัน

2. TCP Node



โหนด TCP (TCP Node) ใน Node-RED เป็นอีกหนึ่งชนิดของโหนดที่มีความสำคัญในการเชื่อมต่อและสื่อสารผ่านโปรโตคอล TCP (Transmission Control Protocol) ระหว่าง Node-RED และอุปกรณ์หรือระบบอื่นๆ ในเครือข่าย TCP เป็นโปรโตคอลที่ใช้ในการส่งข้อมูลตามแบบเชิงควบคุมและเชื่อมต่อในรูปแบบที่เสถียรและน่าเชื่อถือ

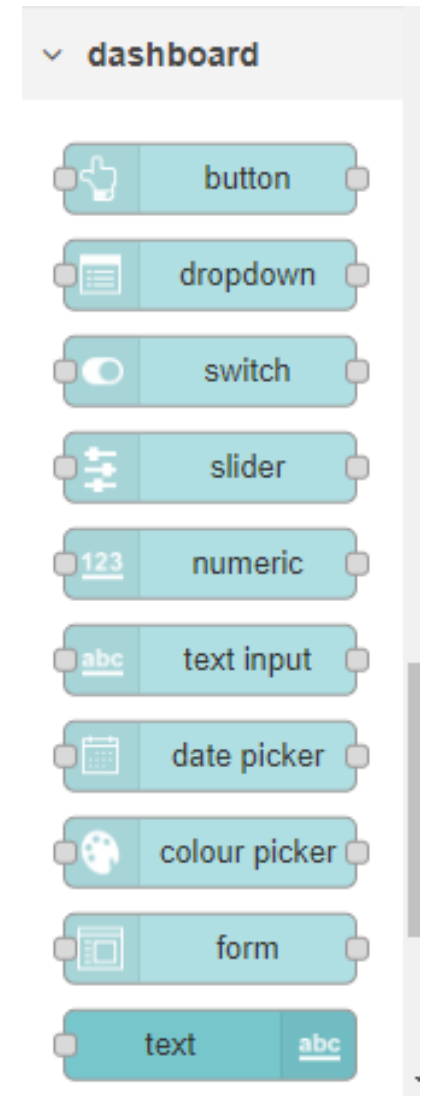
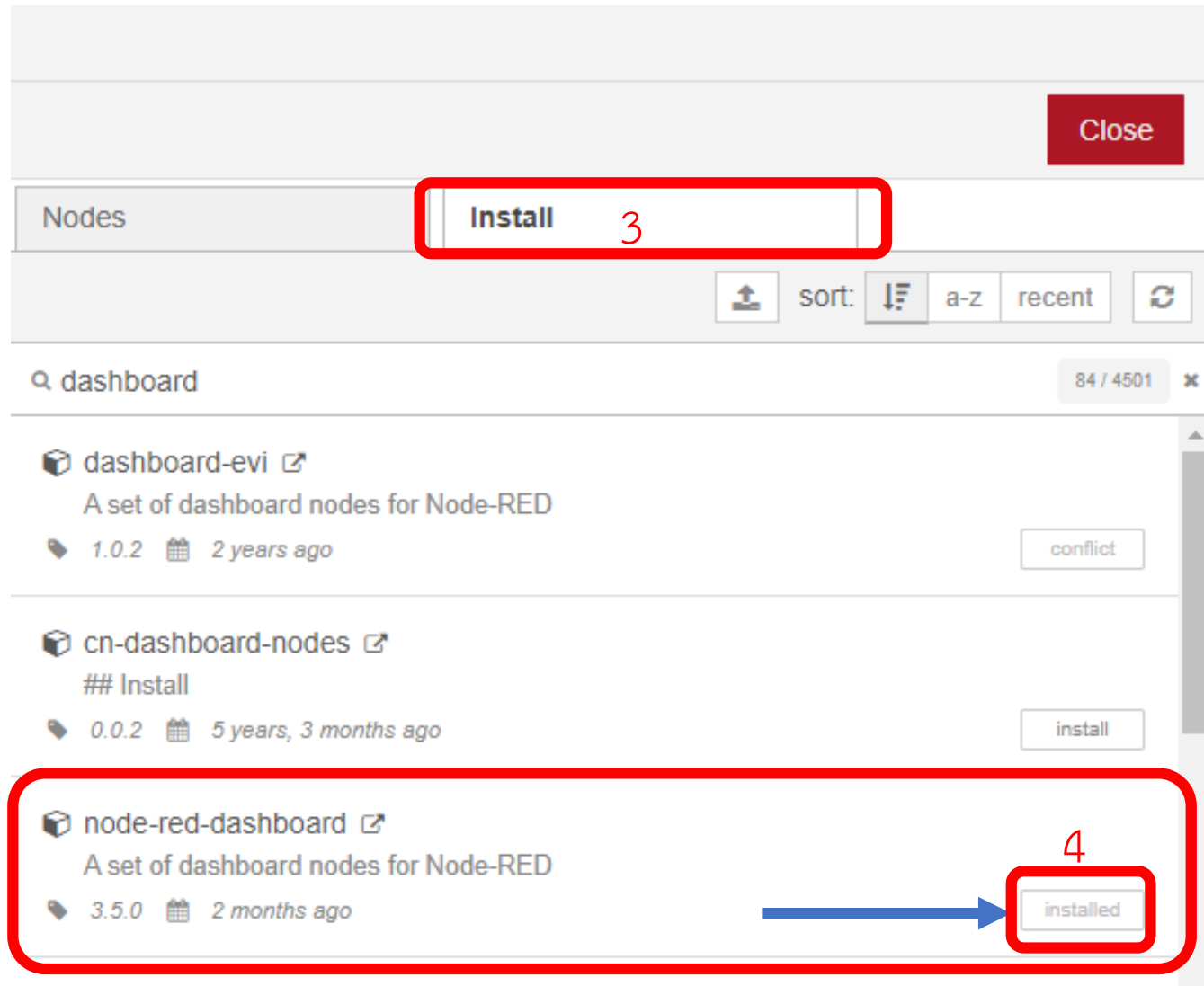
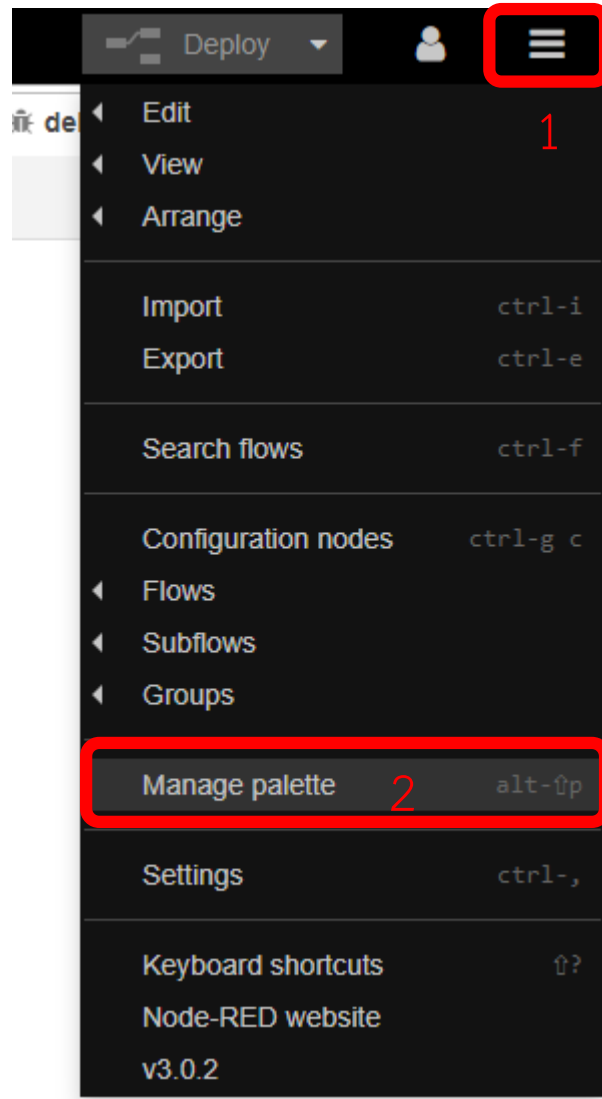
3. UDP Node



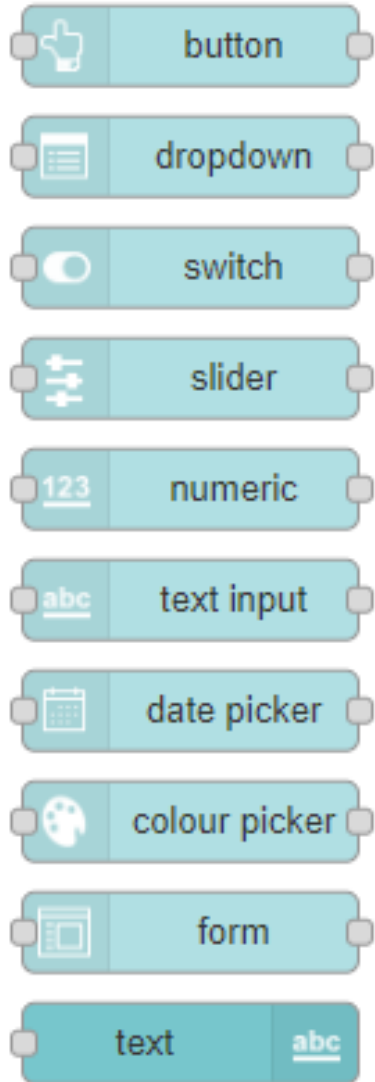
โหนด UDP (UDP Node) ใน Node-RED เป็นโหนดที่ใช้ในการเชื่อมต่อและสื่อสารผ่านโปรโตคอล UDP (User Datagram Protocol) ระหว่าง Node-RED และอุปกรณ์หรือระบบอื่นๆ ในเครือข่าย UDP เป็นโปรโตคอลที่ใช้ในการส่งข้อมูลแบบไม่มีการเชื่อมต่อ ซึ่งไม่มีการตรวจสอบความถูกต้องหรือการเสียหายของข้อมูล และมีประสิทธิภาพในการส่งข้อมูลอย่างรวดเร็ว

การติดตั้ง Node เพิ่ม

การติดตั้ง Dashboard Node เพิ่ม



▼ dashboard



Dashboard

Dashboard ใน Node-RED เป็นเครื่องมือที่ใช้ในการสร้างและแสดงผลข้อมูลในรูปแบบของหน้าแสดงผล (User Interface) ที่ใช้งานได้อย่างง่ายดาย โดยเป็นเครื่องมือที่ช่วยให้ผู้ใช้สามารถควบคุมและติดตามข้อมูลใน Node-RED ได้อย่างสะดวกสบาย ผู้ใช้สามารถสร้างตัวแสดงผลต่าง ๆ ได้เช่น กราฟ, แดชบอร์ด, ปุ่มควบคุม, กล่องข้อความ และอื่น ๆ ซึ่งสามารถปรับแต่งรูปแบบและระเบียบหน้าแสดงผลตามต้องการของโปรแกรมหรือโปรเจกต์ที่กำลังพัฒนาใน Node-RED

เริ่มใช้งานโปรแกรม Node-Red

1. ตัวอย่างการใช้งาน Function node รับค่าจาก INPUT โดยจะเป็นตัวอักษร String จากนั้นเพิ่มตัวอักษรด้วย Function node

1.1 ส่งคำว่า World ออกมาจาก inject node



Edit inject node

Delete Cancel Done

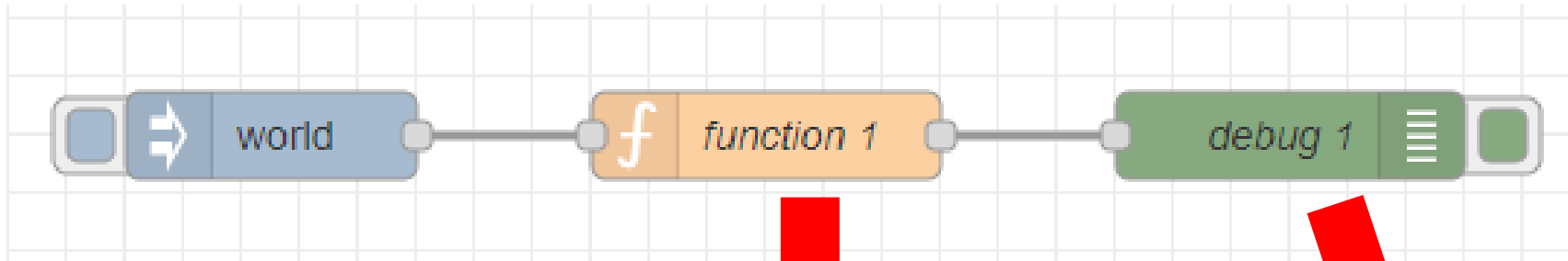
Properties

Name Name

msg. payload = a_z world

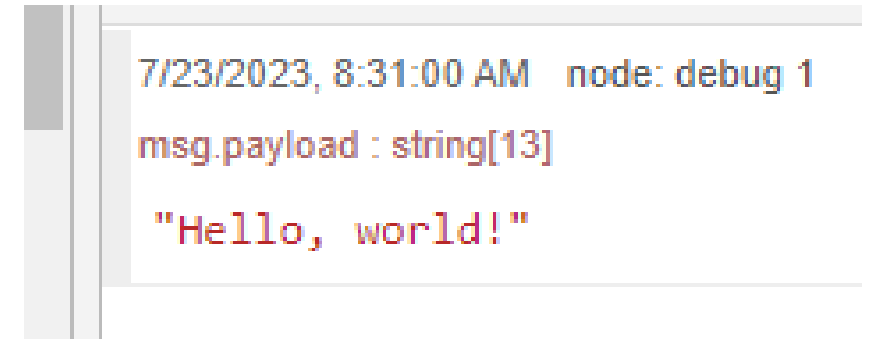
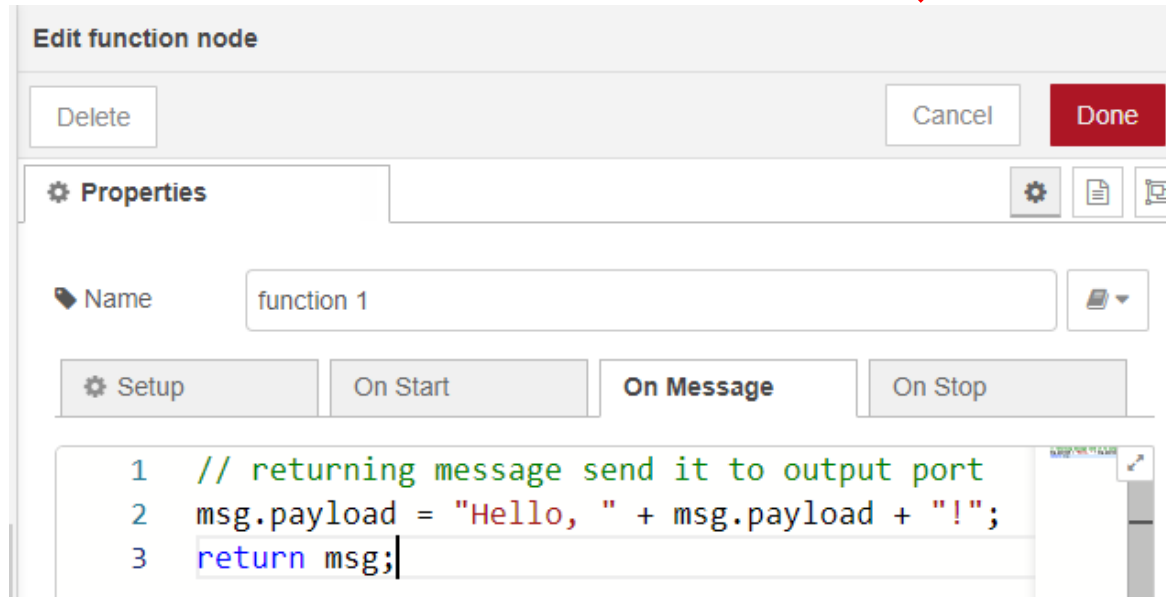
msg. topic = a_z

1.2 เพิ่ม Function node ในเขียนโปรแกรม โดยอ้างอิงค่าข้อมูลจาก inject node จากนั้นส่งออกไปยัง Output Function node และเชื่อมต่อกับ Debug node เพื่อดูข้อมูล



ตัวอย่างโปรแกรม

```
// returning message send it to output port  
msg.payload = "Hello, " + msg.payload + "!";  
return msg;
```



2. ตัวอย่างการใช้งาน Function node ทำ Counter นับจำนวนตั้งแต่ 1-9 โดยจะทำงานเมื่อมีสัญญาณส่งมาที่ input Function node

กดปุ่มนี้ ตัวเลขจะนับเพิ่ม

The screenshot shows a Node-RED workflow on a grid background. It consists of three nodes connected in sequence: a blue 'button' node, an orange 'function 1' node, and a green 'debug 1' node. A red arrow points to the button node with the text 'กดปุ่มนี้ ตัวเลขจะนับเพิ่ม'. Below the workflow, the 'function 1' node's configuration is shown, including its name and the 'On Message' tab. The code for the function node is as follows:

```
1 var count = context.get('count') || 0;
2 if (count == 9) {
3   count = 0
4 }
5 count++;
6 context.set('count', count);
7 msg.payload = count;
8 return msg;
```

On the right, the 'debug 1' node's output is shown, displaying the following log entries:

```
7/23/2023, 8:35:52 AM node: debug 1
msg.payload : number
1
7/23/2023
msg.payload
2
7/23/2023
msg.payload
3
7/23/2023
msg.payload
4
```

ตัวอย่างโปรแกรม

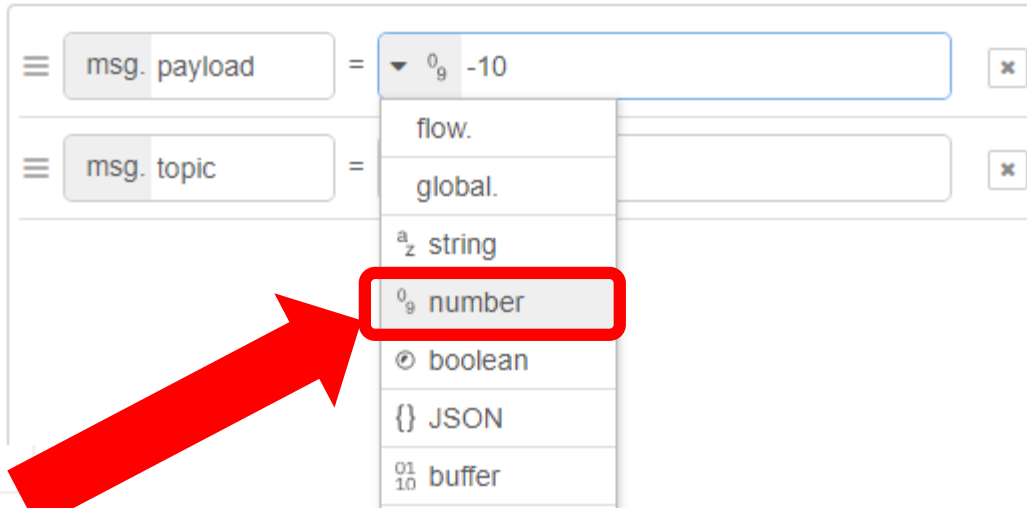
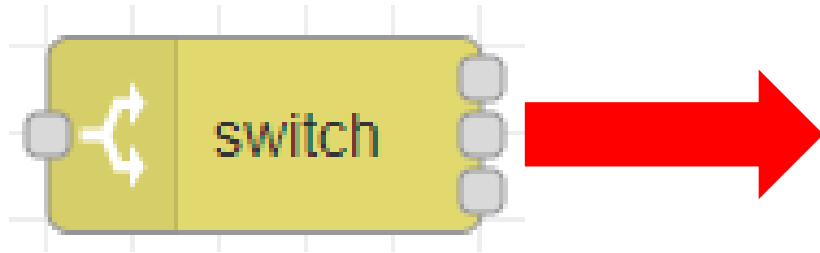
```
var count = context.get('count') || 0;
if (count == 9) {
  count = 0
}
count++;
context.set('count', count);
msg.payload = count;
return msg;
```

ณัฐวัฒน์ พัลลัด

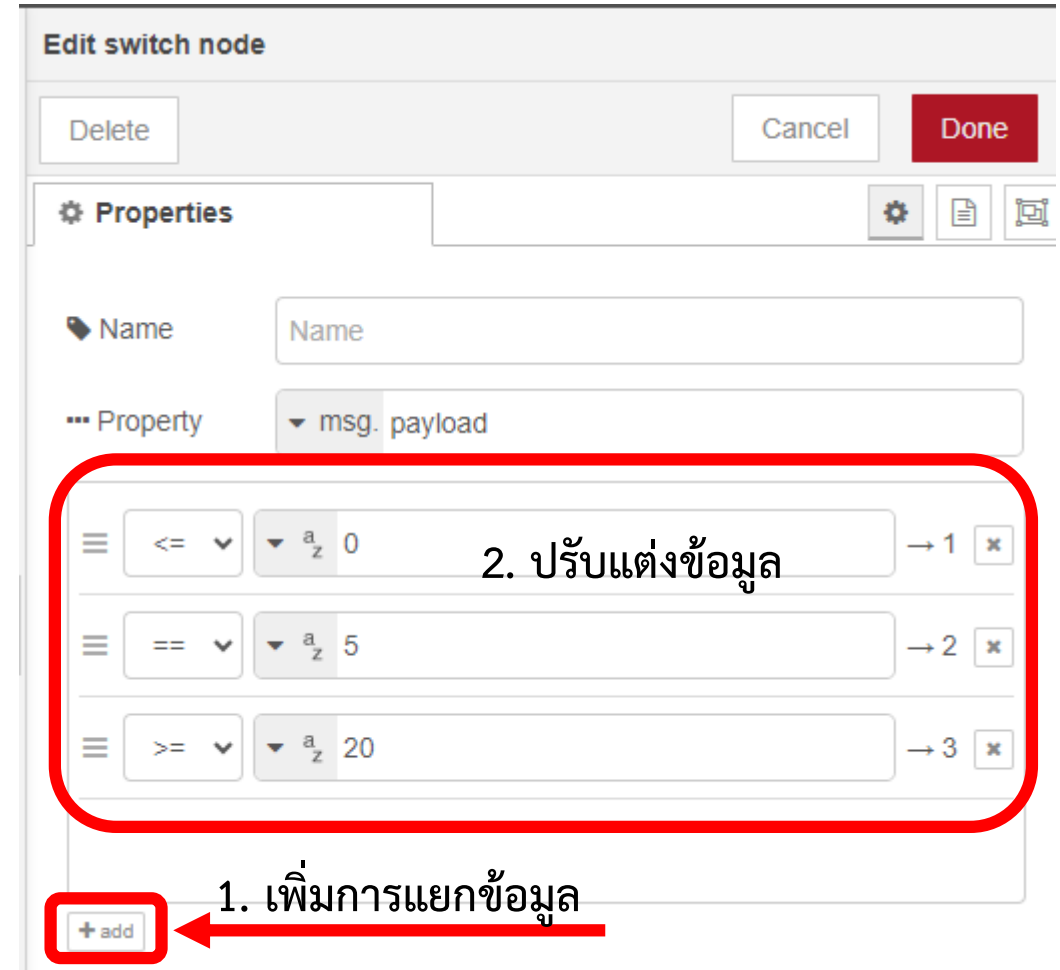
3. ตัวอย่างการใช้ Switch node ในการแยกข้อมูลออกเป็นหลายทาง

โจทย์ รับค่าตัวเลขเข้ามาให้กับ Switch node โดยแบ่งค่าออกไปสาม output ดังนี้

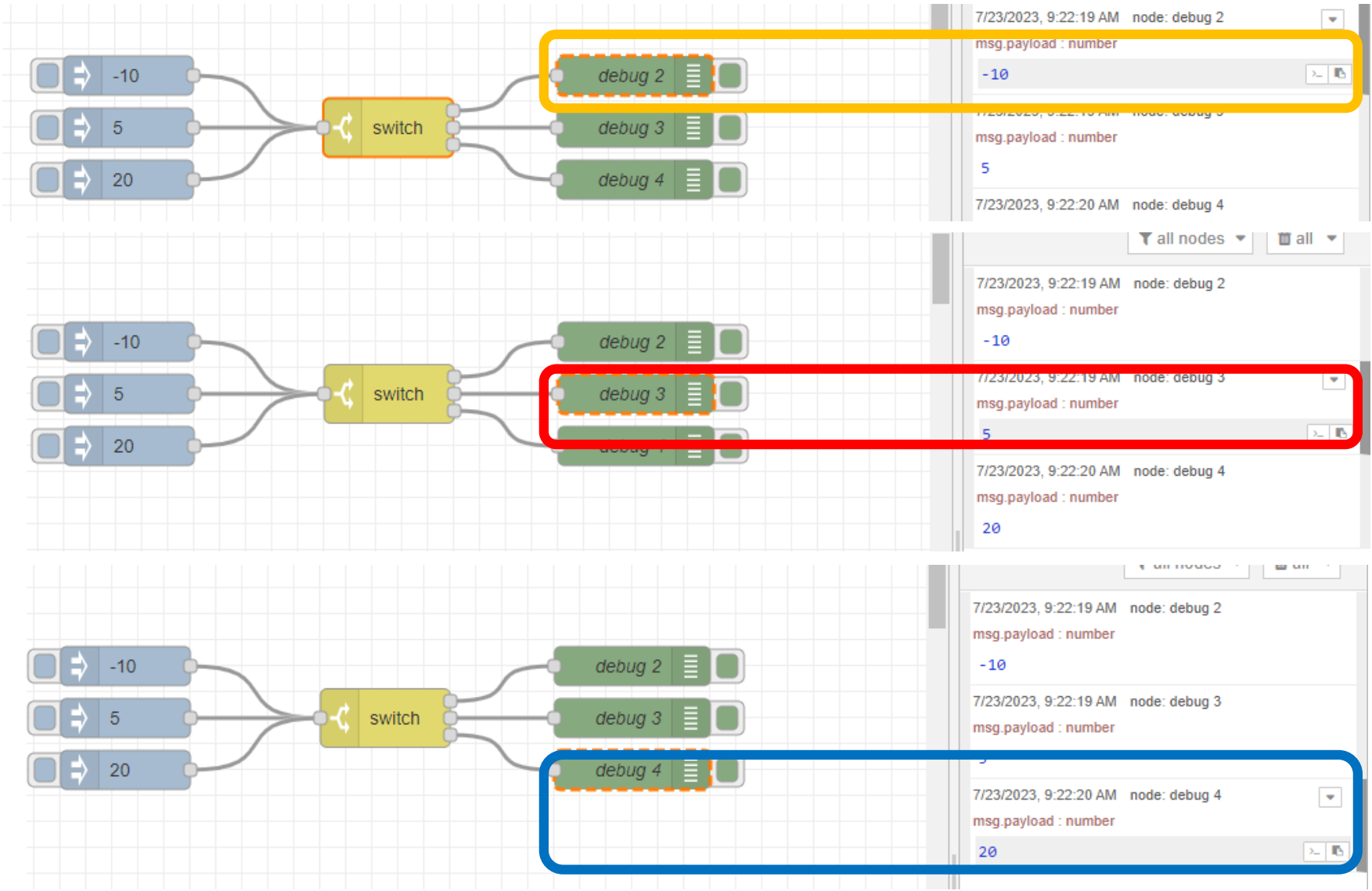
1. ค่าที่น้อยกว่า 0
2. ค่าที่เท่ากับ 0
3. ค่าที่มากกว่า 0



3. เปลี่ยนค่าใน inject node ให้เป็นตัวแปรชนิด number แล้วกรอกตัวเลข -10, 5 และ 20



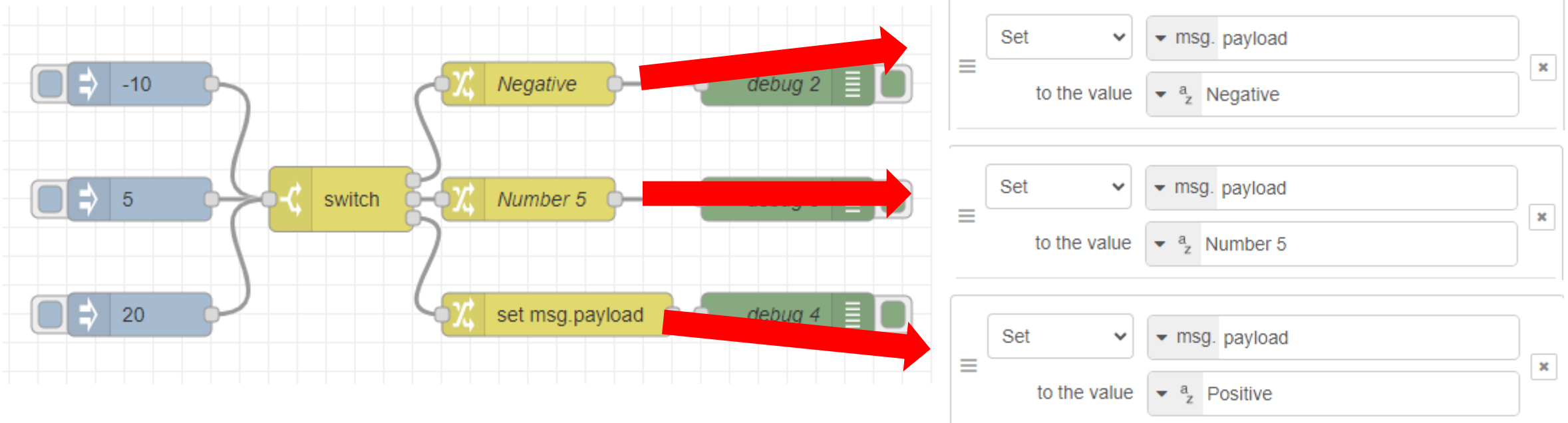
ผลการทำงาน



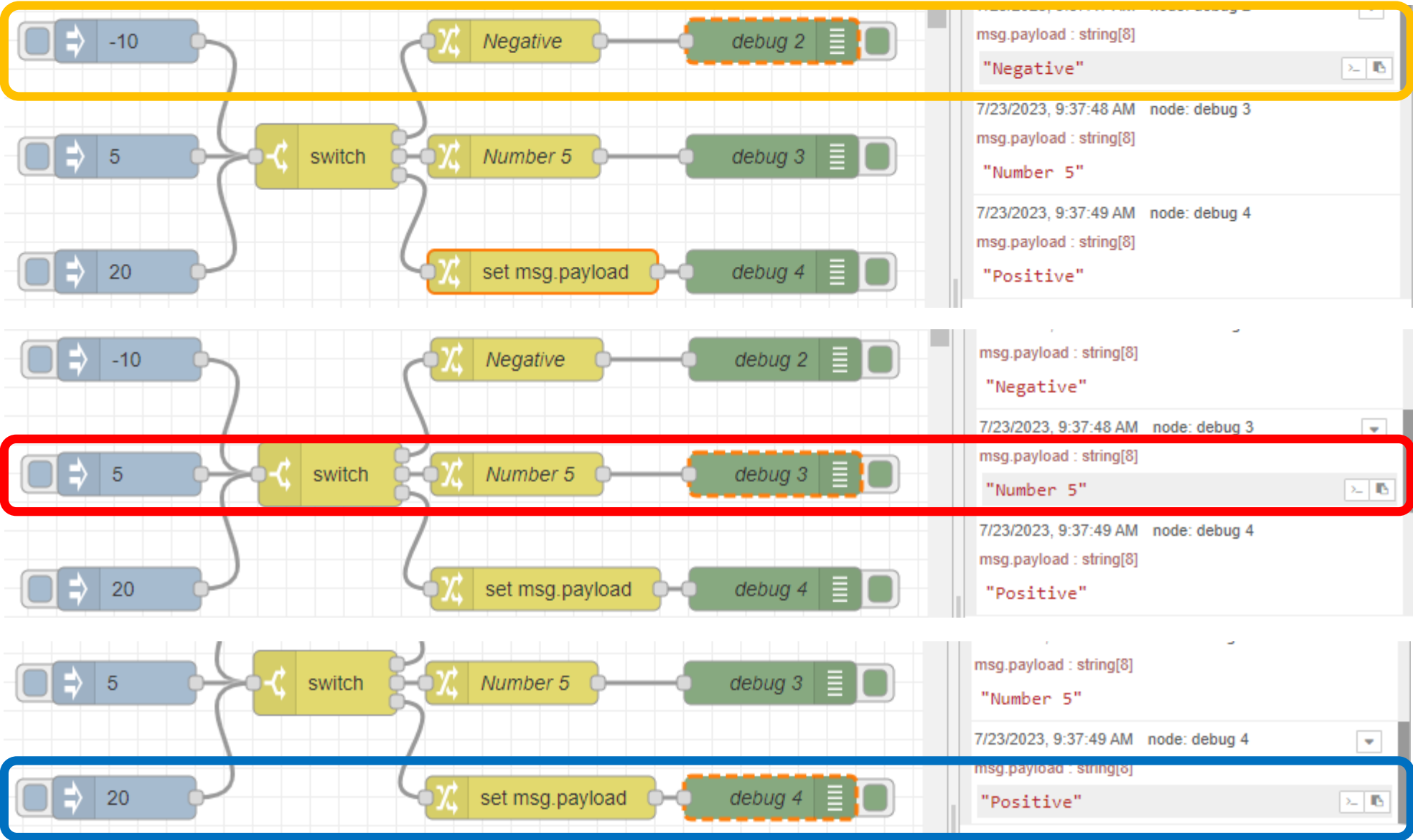
4. ตัวอย่างการใช้ Change node ในการเปลี่ยนแปลงข้อมูล

โจทย์ เปลี่ยนแปลงค่าตัวเลขจาก Switch node ให้เลขที่ออกมาเปลี่ยนแปลงเป็นค่าString ดังนี้

1. ค่าที่น้อยกว่า 0 เปลี่ยนแปลงเป็น Negative
2. ค่าที่เท่ากับ 0 เปลี่ยนแปลงเป็น Zero
3. ค่าที่มากกว่า 0 เปลี่ยนแปลงเป็น Positive



ผลการทำงาน



JSON

JavaScript Object Notation

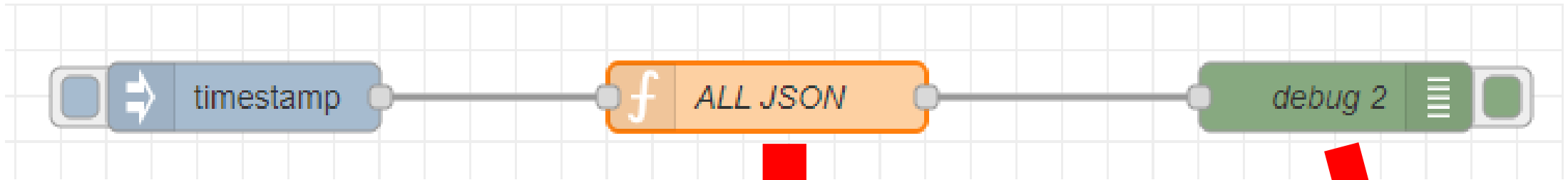
JSON (JavaScript Object Notation) คือรูปแบบของข้อมูลที่ใช้ในการแลกเปลี่ยนและเก็บข้อมูลในรูปแบบของข้อความ (text-based format) ซึ่งเป็นรูปแบบที่เข้าใจง่ายและนิยมใช้ในการสื่อสารข้อมูลระหว่างแอปพลิเคชันต่าง ๆ รูปแบบของ JSON เป็นรูปแบบที่นิยมใช้ในการส่งข้อมูลข้ามเว็บ (Web data exchange) และใช้ในการเก็บข้อมูล (data storage) อย่างกว้างขวาง

{“key” : “values”}

คุณสมบัติที่สำคัญของ JSON คือ:

- JSON จัดเก็บข้อมูลในรูปแบบของชุดของคู่ของ "key-value" ซึ่งแสดงถึงชื่อ (key) และค่า (value) ของข้อมูล
- รูปแบบของ JSON เป็นข้อความธรรมดาและอ่านเข้าใจง่าย ทำให้สามารถอ่านและแก้ไขข้อมูลได้ง่าย
- รองรับชนิดข้อมูลพื้นฐานเช่น ตัวเลข (numbers), สตริง (strings), บูลีน (booleans), อาร์เรย์ (arrays), และ วัตถุ (objects)
- JSON เป็นรูปแบบที่เปิดเผยและเป็นที่ยอมรับในการใช้งานในภาษาโปรแกรมต่าง ๆ เช่น JavaScript, Python, PHP, Java, C++, และอื่น ๆ

ตัวอย่างการทำงาน



ตัวอย่างโปรแกรม

```
var data = {  
  "name": "John Doe",  
  "age": 30,  
  "isMarried": false,  
  "hobbies": ["reading", "gaming", "cooking"],  
  "address": {  
    "street": "123 Main Street",  
    "city": "New York",  
    "country": "USA"  
  }  
}  
msg.payload = data;  
return msg;
```

Edit function node

Delete Cancel Done

Properties

Name function 2

Setup On Start On Message On Stop

```
1 var data = {  
2   "name": "John Doe",  
3   "age": 30,  
4   "isMarried": false,  
5   "hobbies": ["reading", "gaming", "cooking"],  
6   "address": {  
7     "street": "123 Main Street",  
8     "city": "New York",  
9     "country": "USA"  
10  }  
11 }  
12 msg.payload = data;  
13 return msg;
```

all nodes all

7/23/2023, 10:24:44 AM node: debug 2

msg.payload : Object

- object
 - name: "John Doe"
 - age: 30
 - isMarried: false
 - hobbies: array[3]
 - 0: "reading"
 - 1: "gaming"
 - 2: "cooking"
 - address: object
 - street: "123 Main Street"
 - city: "New York"
 - country: "USA"

ผลลัพธ์ที่ได้

ตัวอย่างการทำงาน ในการนำบางส่วนข้อมูลไปใช้งาน

ตัวอย่างเช่น การนำค่า address ไปใช้งาน

7/23/2023, 10:24:41 AM node: debug 2
msg.payload : Object

▼ object

- name: "John Doe"
- age: 30
- isMarried: false

1.เลือก Path copied ที่ address

1: "gaming"
2: "cooking"

▼ address: object Path copied

- street: "123 Main Street"
- city: "New York"
- country: "USA"

2. เขียนโปรแกรมใน function node เพิ่ม
โดยนำข้อมูลที่คัดลอกมาวางในโปรแกรม

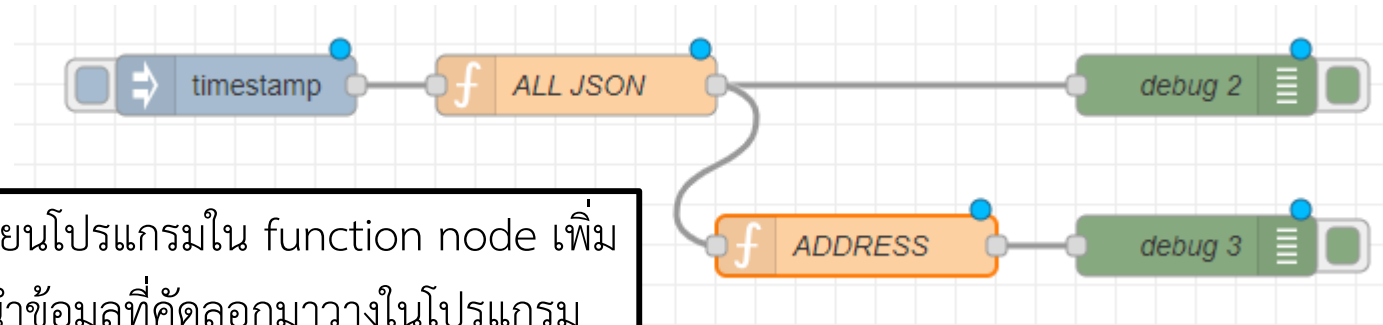
Delete

Properties

Name ADDRESS

Setup On Start On Message

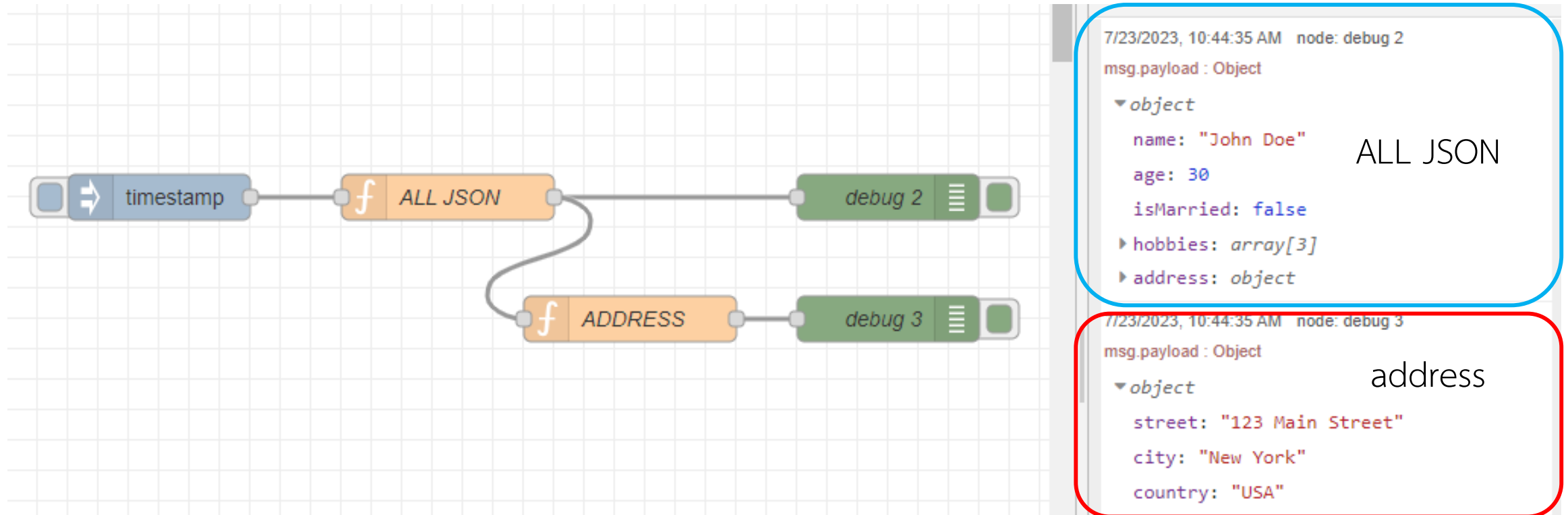
```
1 msg.payload = msg.payload.address;  
2 return msg;
```



msg.payload = msg.payload.address;
return msg;

ส่วนที่คัดลอก

ผลการทำงาน

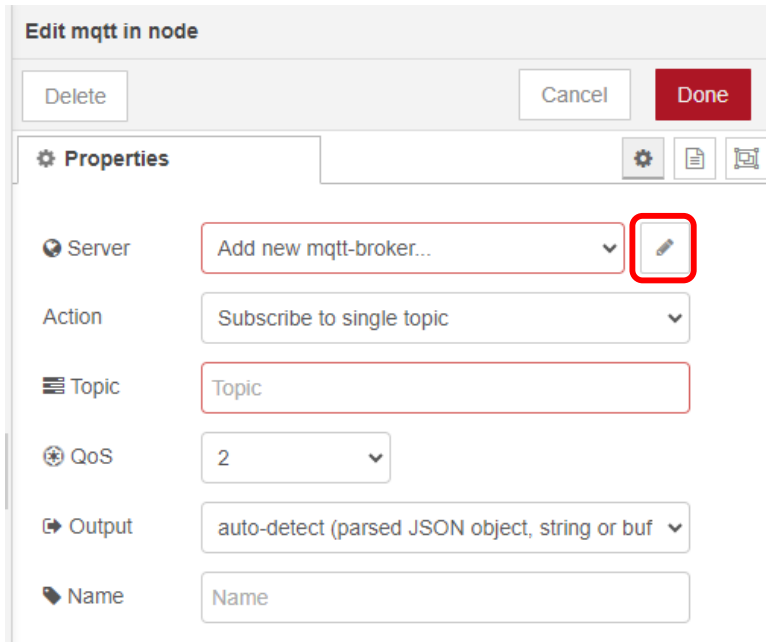


MQTT

Message Queuing Telemetry Transport

การตั้งค่า MQTT


1. สร้าง Server MQTT เพื่อให้ MQTT IN ค้นหาที่อยู่ MQTT



Edit mqtt in node

Delete Cancel Done

Properties

Server Add new mqtt-broker... 

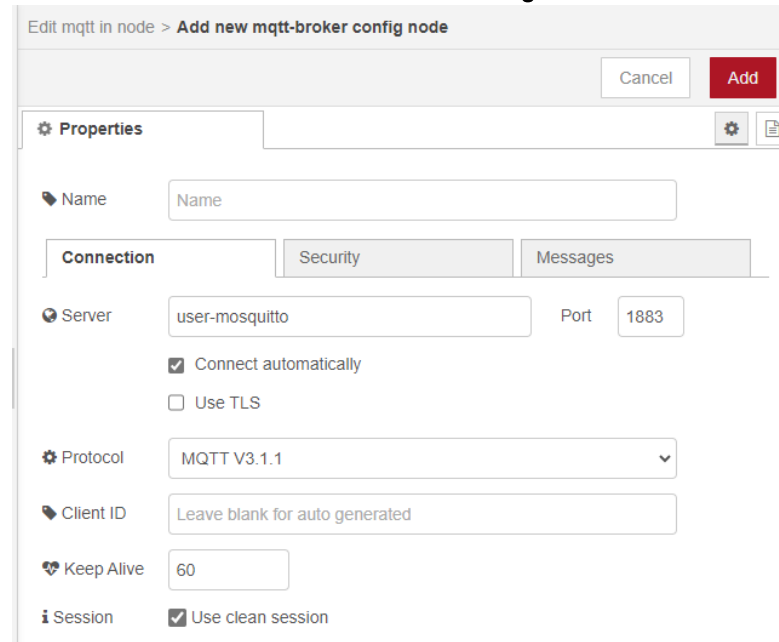
Action Subscribe to single topic

Topic Topic

QoS 2

Output auto-detect (parsed JSON object, string or buf)

Name Name



Edit mqtt in node > Add new mqtt-broker config node

Cancel Add

Properties

Name Name

Connection Security Messages

Server user-mosquitto Port 1883

☒ Connect automatically

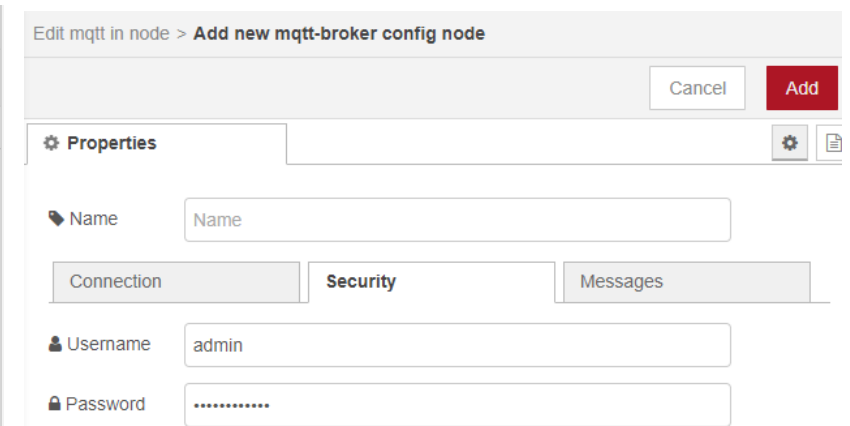
☐ Use TLS

Protocol MQTT V3.1.1

Client ID Leave blank for auto generated

Keep Alive 60

Session ☒ Use clean session



Edit mqtt in node > Add new mqtt-broker config node

Cancel Add

Properties

Name Name

Connection Security Messages

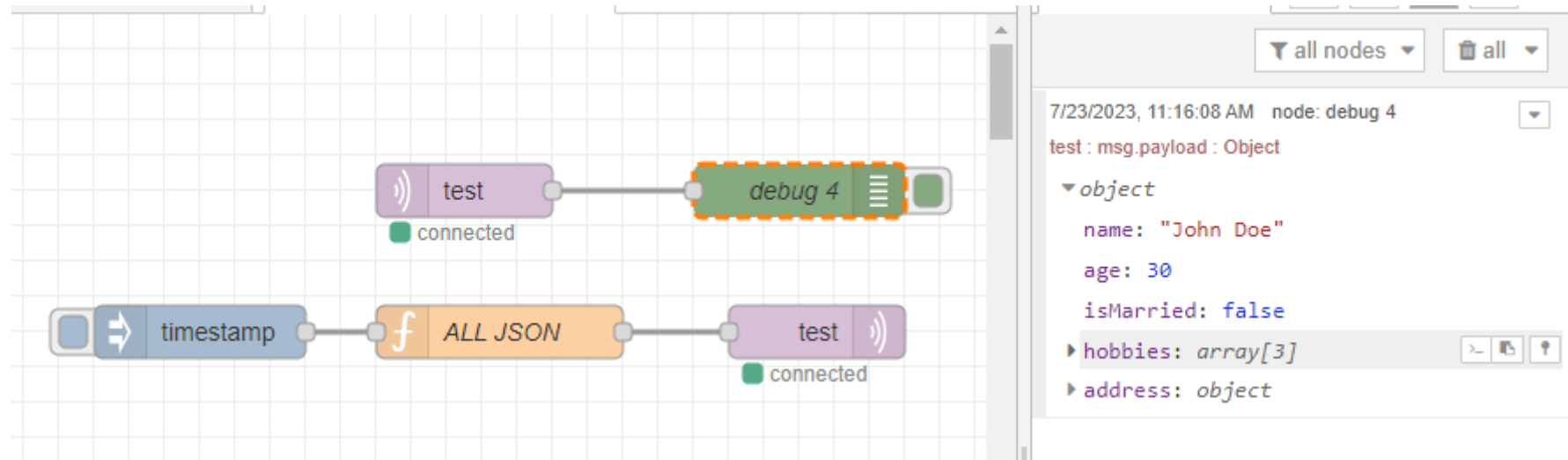
Username admin

Password

*Topic ต้องตั้งค่าให้ตรงกันระหว่าง Public และ Subscribe

Server	user-mosquitto
Port	1883
Protocol	MQTT V3.1.1
Client ID	ใส่หรือไม่ใส่ก็ได้
Username	admin
Password	Passord ตอนตั้งค่า Docker

ตัวอย่างการใช้งาน ส่งข้อมูลแบบ JSON โดยใช้ Topic = “test”

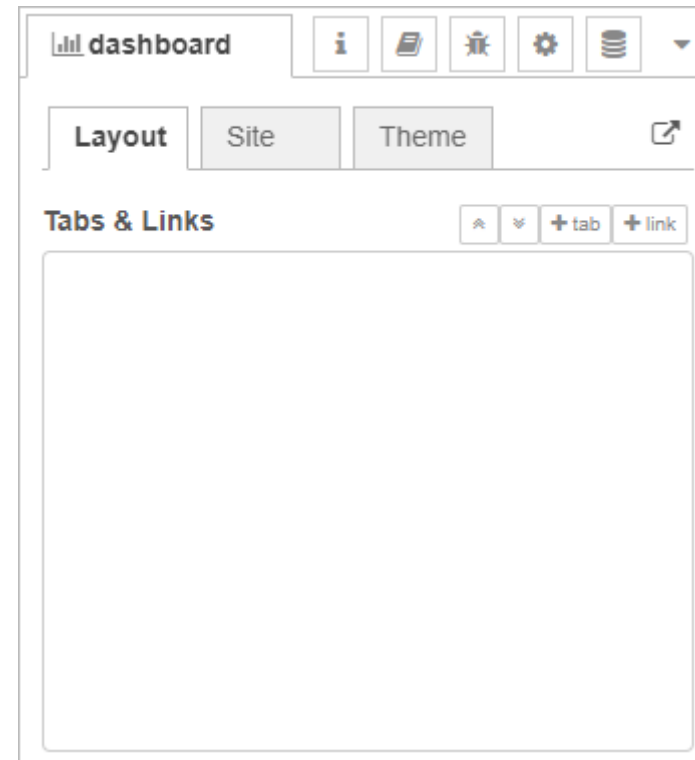
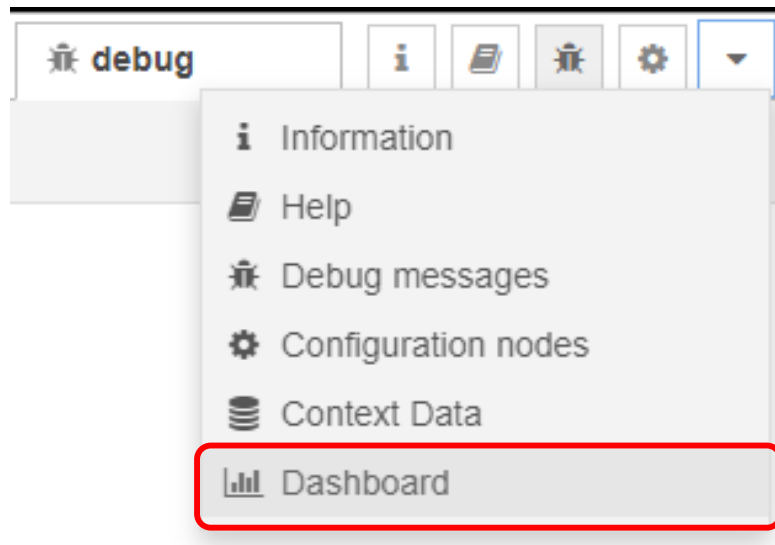


DASHBOARD

การใช้งาน Dashboard

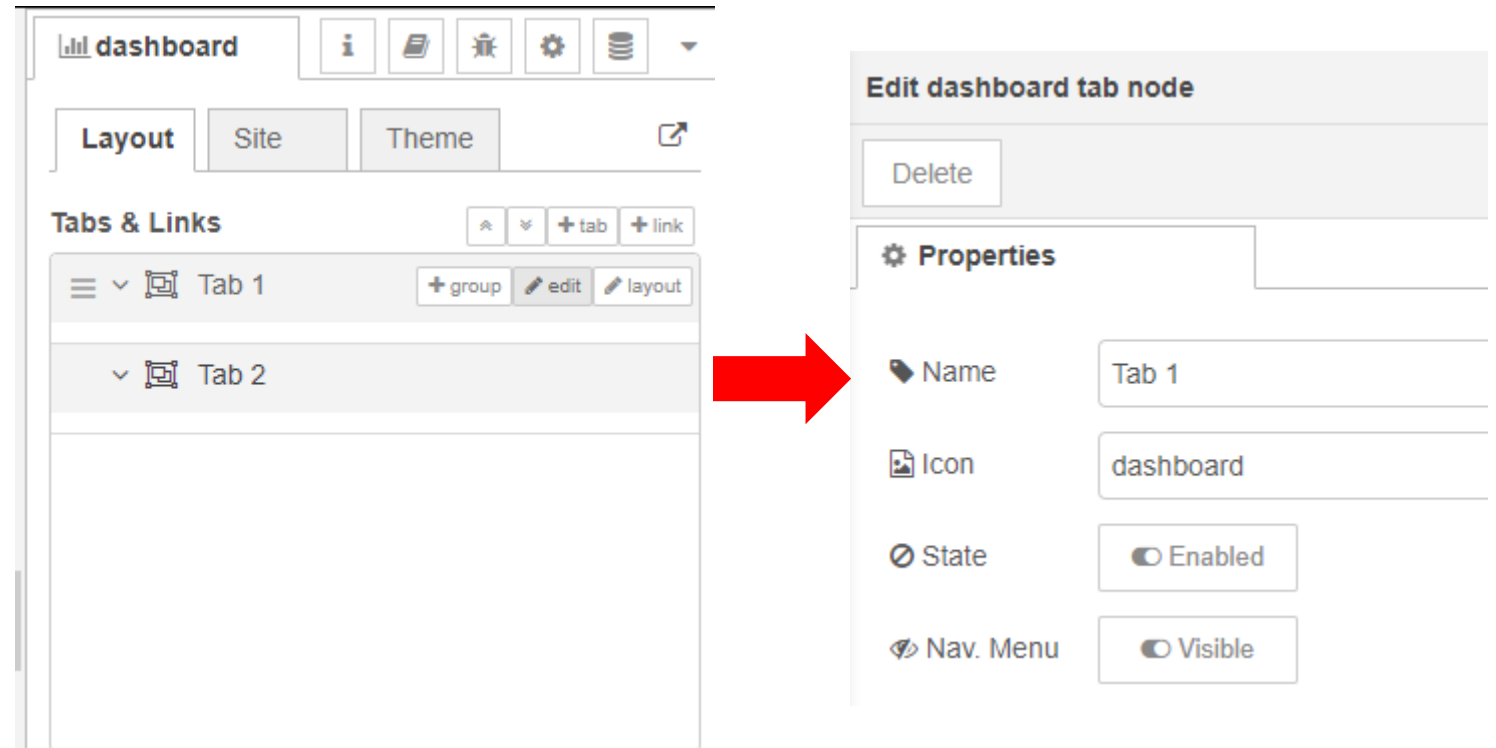
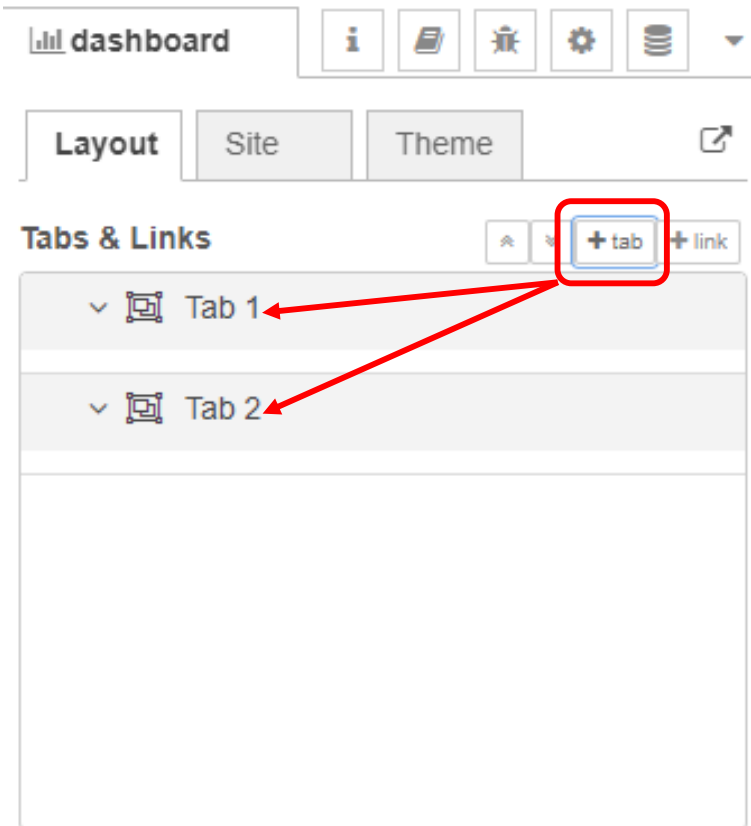
1. การเพิ่มหัวข้อหลักหน้า Dashboard ก่อนการใช้งาน

การจะใช้งาน Dashboard จะต้องทำการเพิ่มหน้า tap ให้กับ Dashboard ก่อน เพื่อเป็นการระบุที่อยู่ของกราฟฟิคที่ผู้ใช้งานสร้างขึ้นว่าต้องการให้ไปอยู่ตำแหน่งใดในหน้า Dashboard

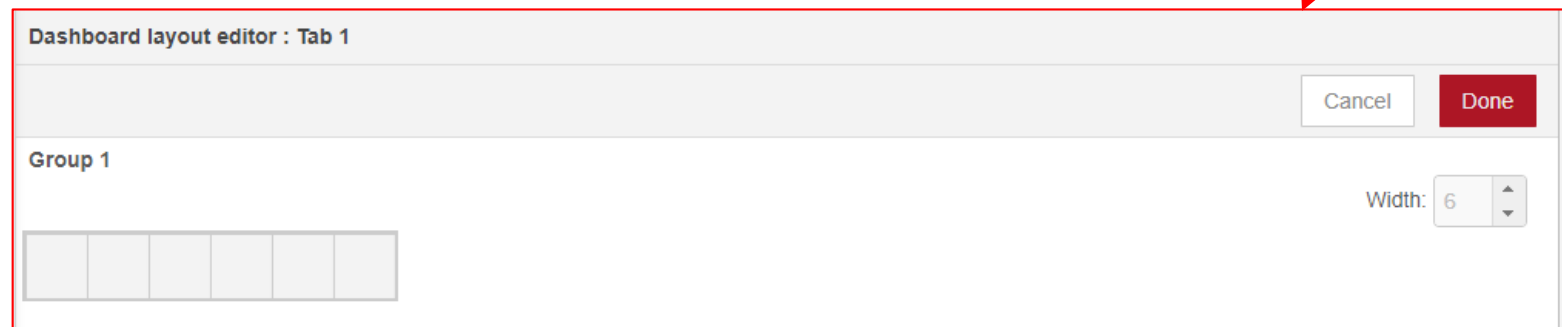
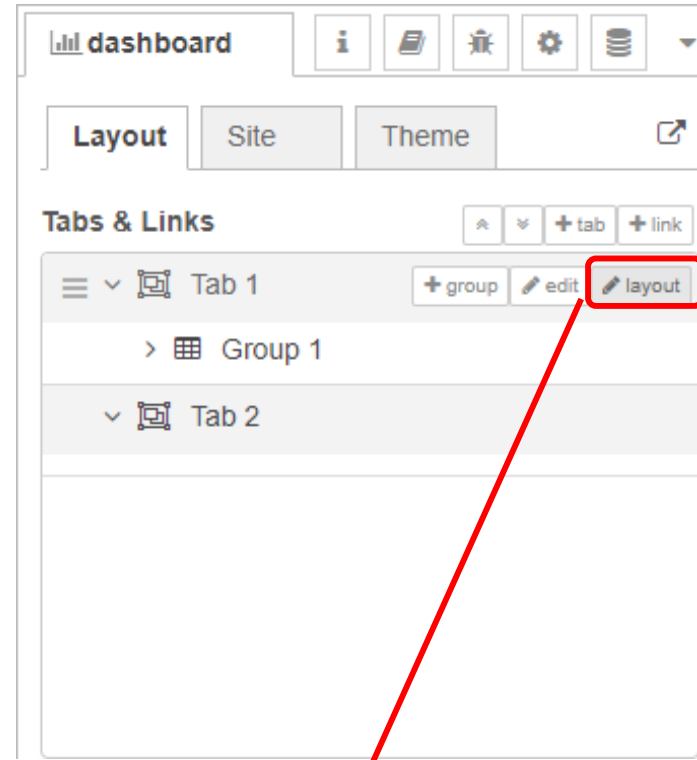
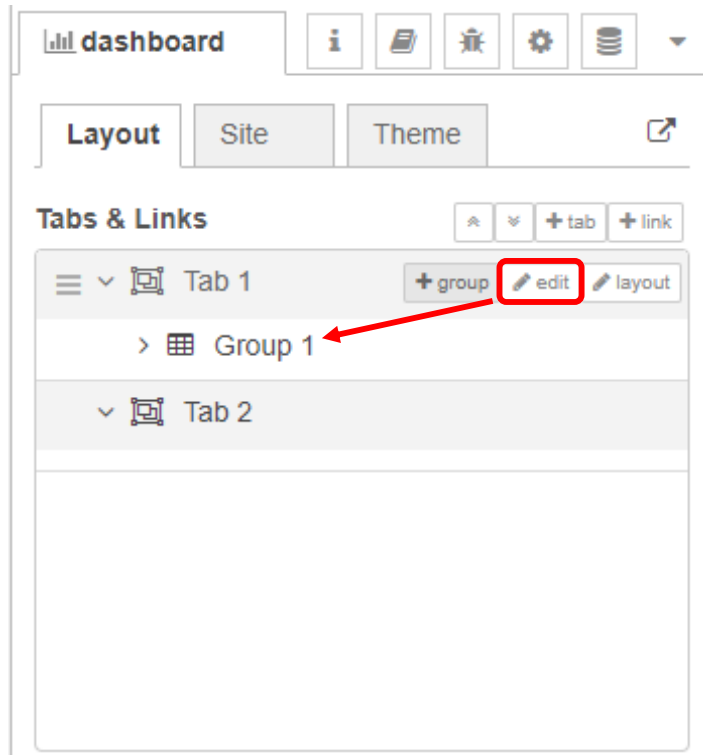


2. เพิ่มหัวข้อ tap ในแต่ละ tap จะเป็นชื่อหัวข้อหลักในหน้าจอ Dashboard

3. แก้ชื่อหัวข้อได้ใน edit

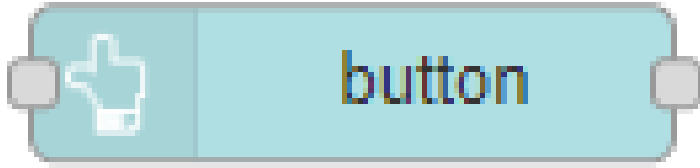


4. การเพิ่ม Group ภายในหัวข้อ tab



Dashboard Node

1. Button node



ปุ่มกดสั่งงานในหน้า Dashboard สามารถตั้งค่าข้อมูลที่เป็นไปได้ การจะสั่งงานจำพวก Dashboard Node จะต้องไปสั่งในหน้า เว็บ UI จึงจะสามารถสั่งงานได้

1.1 เพิ่ม Button node กดเข้าไปใน node เพื่อตั้งค่าการสั่งงาน

Edit button node

Delete

Cancel

Done

Properties

Group

[Tab 1] Group 1

ที่อยู่ Group

Size

auto

กำหนดขนาดของปุ่ม

Icon

optional icon

กำหนดรูป icon

Label

button

ตั้งชื่อ node

Tooltip

optional tooltip

Color

optional text/icon color

ตั้งค่าสีตัวอักษร (HTML Color Codes)

Background

optional background color

ตั้งค่าสีปุ่ม (HTML Color Codes)

When clicked, send:

Payload

a_z

ตั้งค่า output ที่จะส่งออกไปขณะปุ่มกดทำงาน

Topic

msg.topic

If msg arrives on input, emulate a button click:

Class

Optional CSS class name(s) for widget

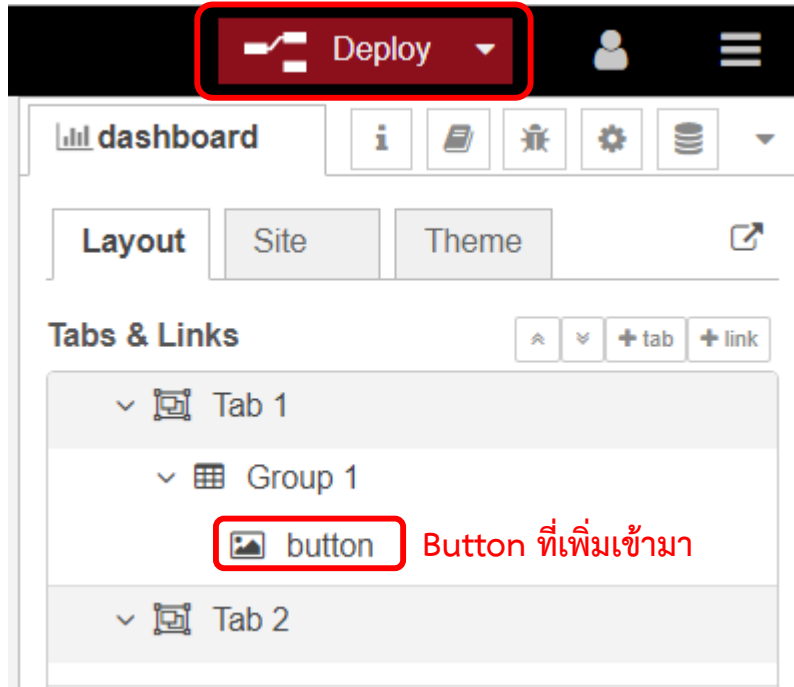
Name

Name

ณัฐวัฒน์ พัลลวล

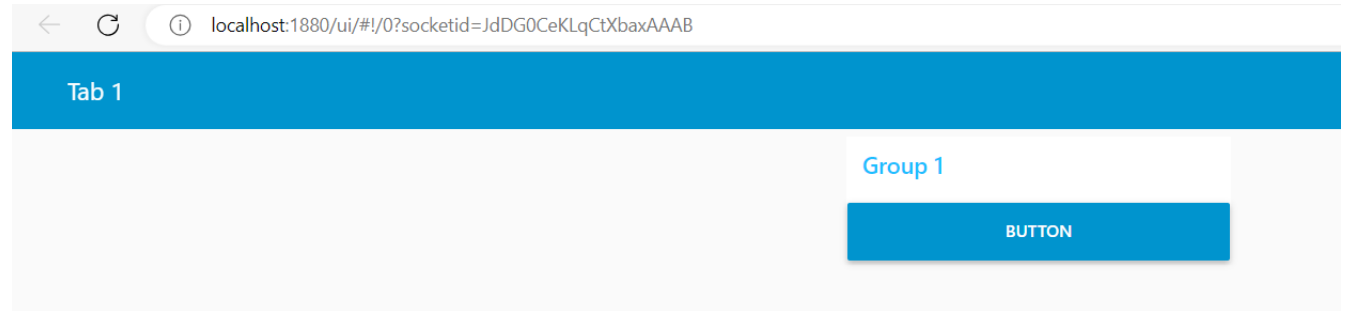
75

1.2 เมื่อมี button เพิ่มเข้ามา กด deploy เพื่อเป็นการเริ่มทำงานของ dashboard



1.3 การเปิดหน้า UI Dashboard

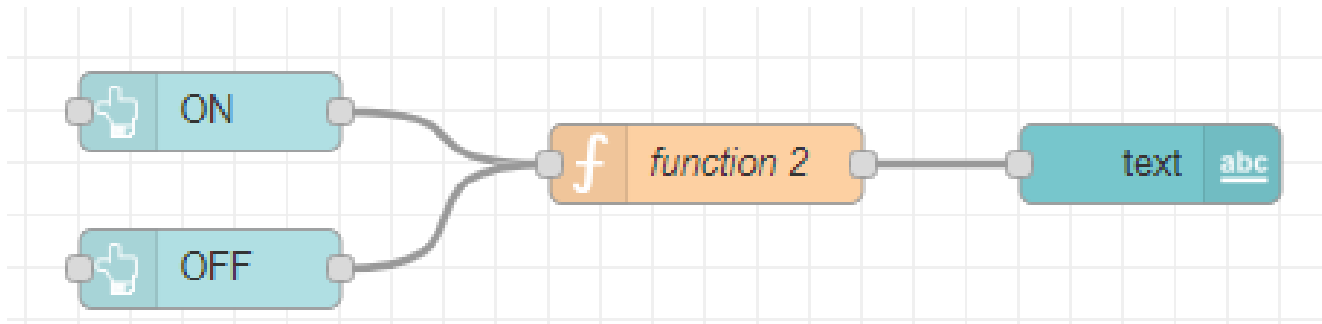
สามารถเข้าใช้งานหน้าเว็บ Dashboard ได้ โดยการเข้าไปที่เว็บไซต์ <http://127.0.0.1:1880/ui>



การตั้งค่าแรก สามารถนำไปใช้ต่อยอดใน Node อื่น ๆ ของ Dashboard ได้

ตัวอย่างการใช้งาน button และ text ร่วมกัน

โจทย์ เมื่อกดปุ่ม ON ให้แสดงชื่อตัวเอง หากกดปุ่ม OFF จะไม่มีการแสดงค่า



```
if (msg.payload === true){  
    msg.payload = "Nattawat Panlawan";  
}  
else {  
    msg.payload = "";  
}  
return msg;
```



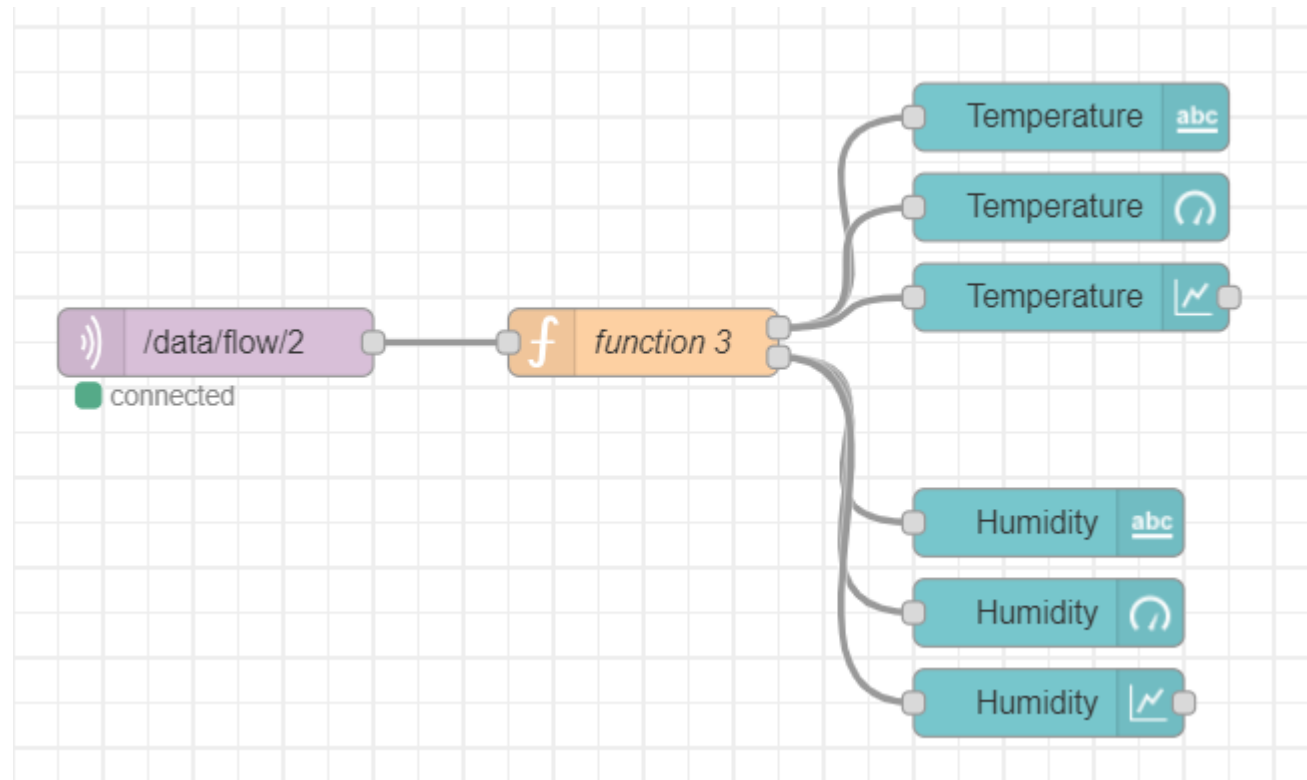
ผลการทำงาน

ตัวอย่างการใช้งาน MQTT กับ Dashboard

โจทย์ รับค่าสภาพแวดล้อมจาก MQTT แล้วนำค่าอุณหภูมิและความชื้นมาแสดงผลยังหน้า dashboard



realtime-dashboard.json



Node-RED interface showing a flow editor with a single node connected to a data source.

The interface includes a sidebar with node categories: **filter nodes**, **storage**, and **dashboard**.

- filter nodes**: json, xml, yaml
- storage**: write file, read file, watch, influxdb in, influxdb out, influx batch
- dashboard**: button, dropdown, switch, slider, numeric, text input

The main workspace displays a single node labeled `/data/flow/2` with a status indicator `connected`.

The right sidebar shows the **debug** console with options to **all nodes** and **all**.

The bottom status bar displays the Windows taskbar with the search bar and system tray icons.