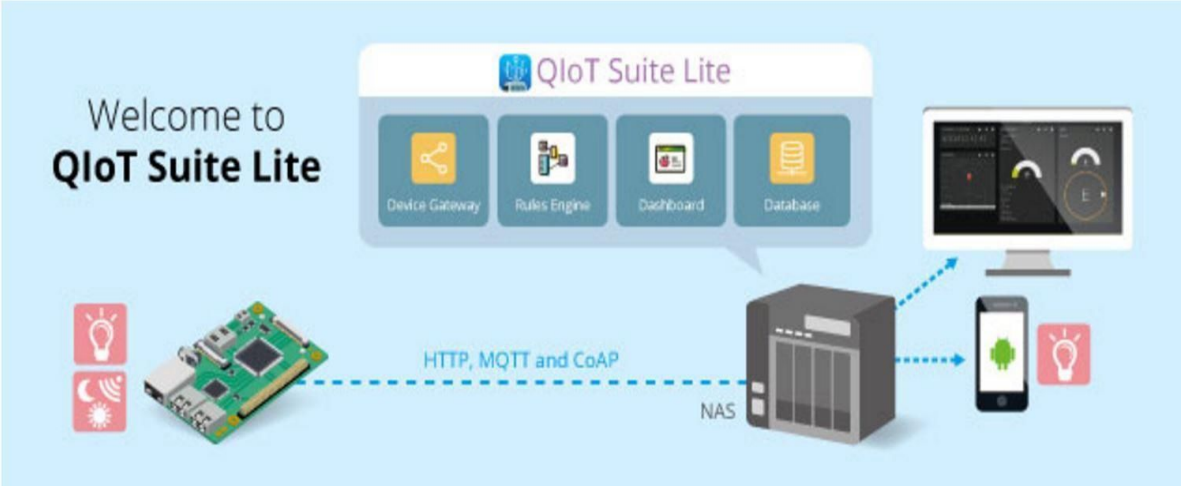
Get started with Intel Edison (Nodejs)

In this tutorial, you begin by learning the basics of working with Intel Edison that's running Linux OS based on Yocto. You then learn how to seamlessly connect your devices to QNAP NAS by using QIoT Suite Lite.
Please ensure your Intel Edison and NAS is under the same LAN.



Lesson 1: Configure your device

In this lesson, you configure your Intel Edison device with an operating system, set up your development environment, and deploy an application to Intel Edison.

1.1 Download Intel Edison drivers

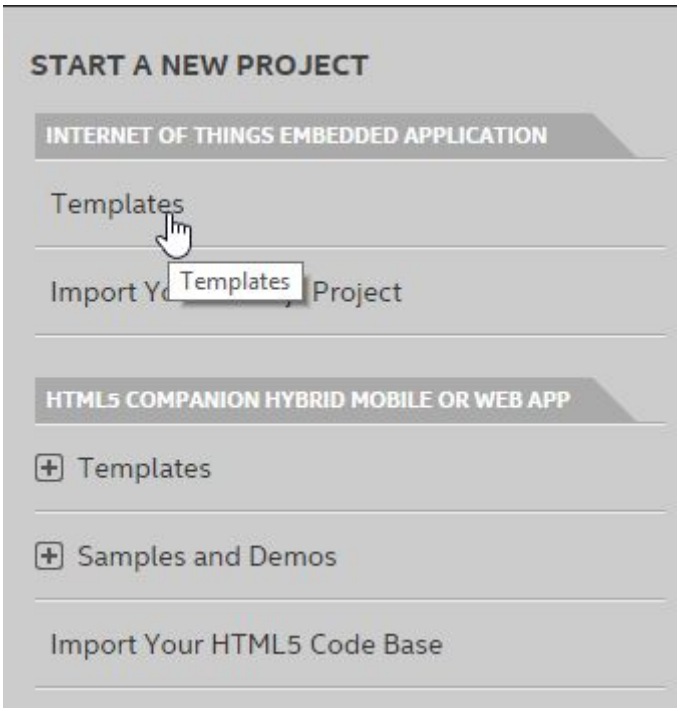
- Download Intel Edison drivers based on your host PC, available here:
  - Windows users
  - Mac users
  - Linux users

1.2 Configure and test your device

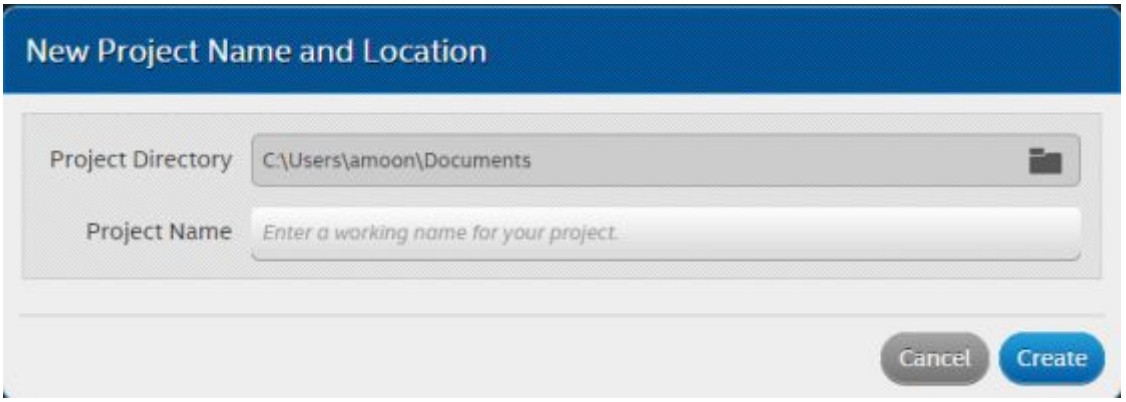
- Get started with Intel Edison
  If this is the first time you use your Intel Edison board, you will have to follow some steps to assemble it.
  Please follow steps 1-3 of the instructions.
  - Step 1: Assemble boards and sensors.
  - Step 2: Run setup tools.
    Use the setup tool to flash the latest firmware on the Intel® Edison development board via a convenient wizard. The setup tool also lets you enable SSH and Wi-Fi* connectivity to your board, as described in the steps to follow.
  - Step 3: Download development environment.
    Download Intel® **XDK** cross-platform integrated development environment

- Verify and upload your onBoardBlink project to Intel Edison

- Creating your project
  1. Launch the Intel XDK.
  2. Follow the on-screen instructions to log in to your Intel® XDK account or sign up for a new Intel XDK account.
  3. From the Projects tab, click **Start a New Project** in the bottom left. The Start a New Project page opens.
  4. In the Internet of Things Embedded Application list on the left, click **Templates**. A list of templates are displayed on the right.

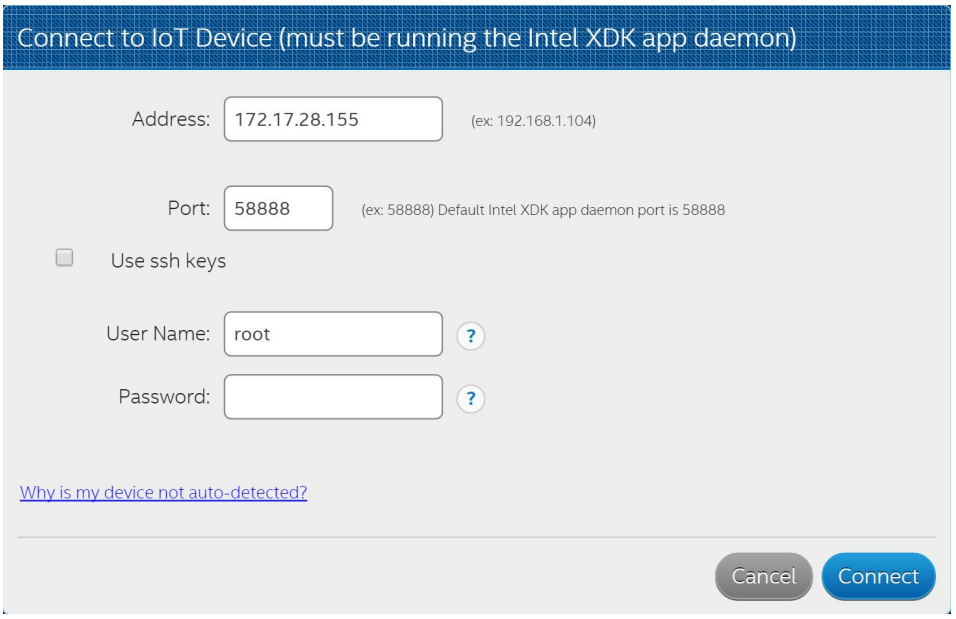


5. Select the **Onboard LED Blink** template, then click **Continue**.
6. If desired, type or browse for the folder you want to store your projects in the **Project Directory** field.
7. In the **Project Name** field, type a name for your project.



8. Click **Create**. Your project is created.

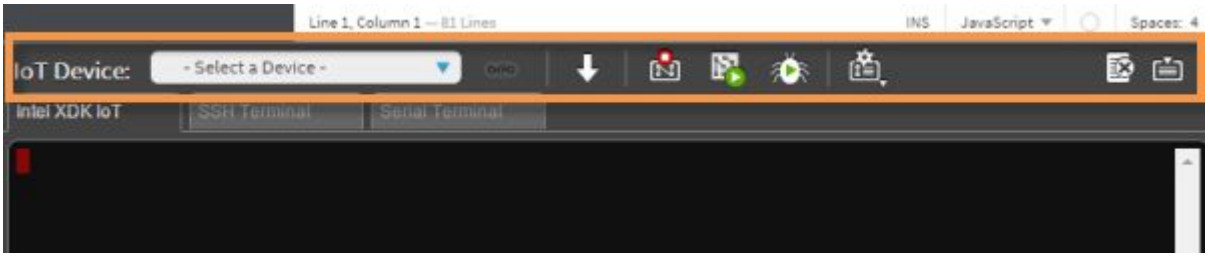
- Connecting to the board
  1. From the **IoT Device** drop-down list in the bottom left, select your development board. The Connect to your IoT Device dialog box opens.



If your development board is not included in the drop-down list, you must connect to your board manually. Select **Add Manual Connection** and supply your board's IP address in the **Address** field.

2. If you have created a user name and password to log in to your board, type them in the **User Name** and **Password** fields. Otherwise, leave the default values.
3. Click **Connect**.
4. If you see a message saying that your daemon is out of date, update the daemon. This will disconnect from the board, as the daemon has to be restarted, so follow the instructions to reconnect when prompted.
5. If you see a message about the clock on your board and the Intel XDK IoT Edition being out of sync, click **Sync**.
6. A confirmation message appears, displaying the connection status and IP address of your development board. Click **Dismiss**. Your board is now connected.

- Upload and run the project
  1. From the top of the application window, click the **Develop** tab. Look for the IoT toolbar at the bottom of the window.



  2. Click the **Upload** icon to upload your project to the board.
  3. Now click the **Run** icon to run your code. You should see an LED on your board flashing on and off. Click the **Stop** icon to stop the LED.

  Congratulations, you have run your first IoT app using the Intel XDK IoT Edition!

- More Intel Edison setup guide, please refer to

https://software.intel.com/en-us/get-started-edison-windows

1.3 Install Nodejs and required libraries

- Open a terminal application on your computer and connect to Intel Edison.
- Install the latest nodejs

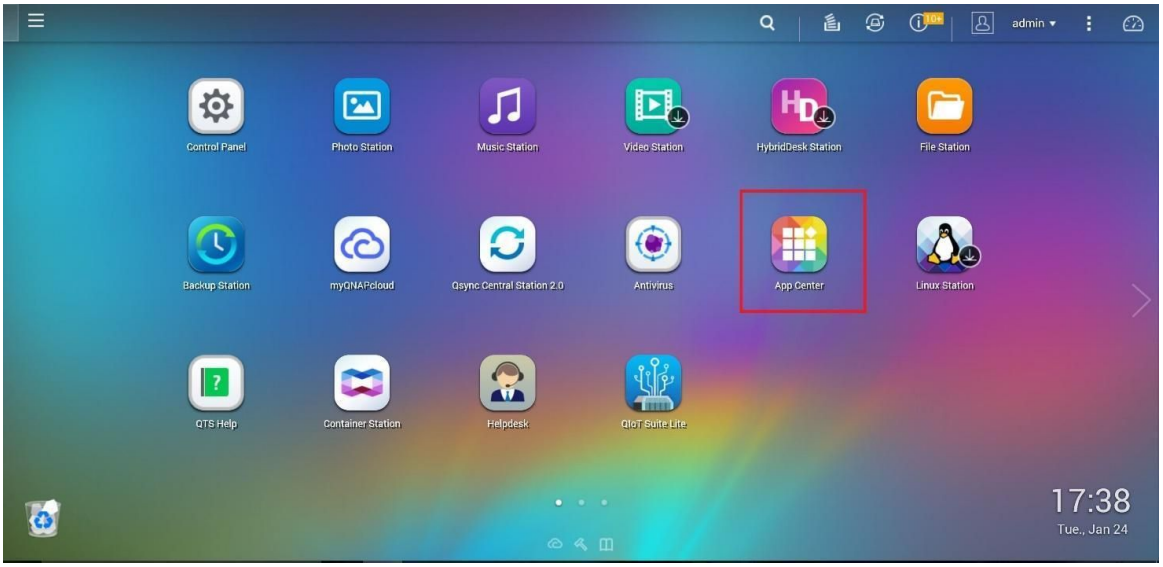
```
root@Edison:~# opkg update
root@Edison:~# opkg install nodejs
```

Lesson 2: Create your device in QIoT Suite Lite

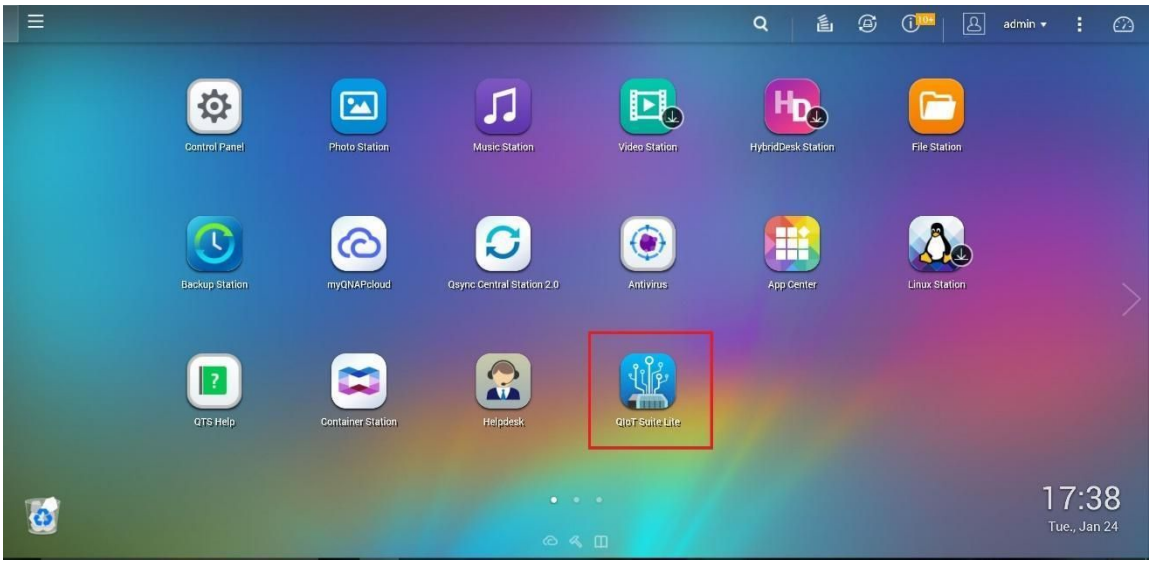
In this lesson, you provision your QNAP QIoT Suite Lite software, and create your first device in QIoT Suite Lite.

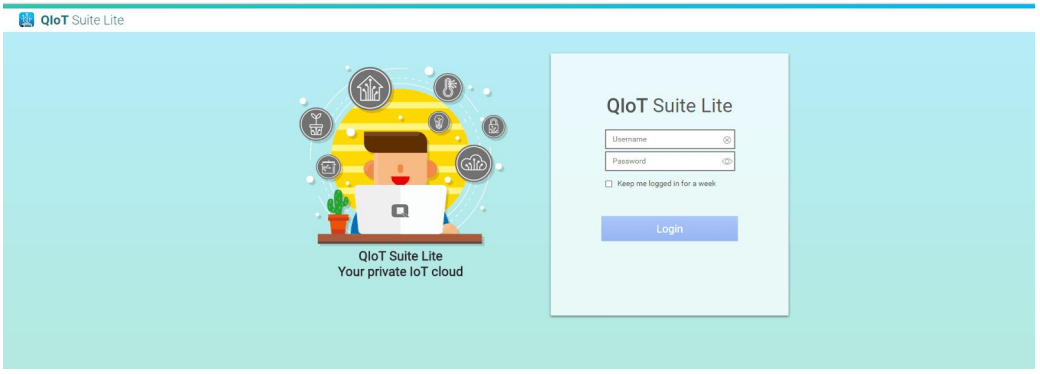
2.1 Install QIoT Suite Lite

- Go to QNAP App Center and download QIoT Suite Lite application.



- Launch and log in QIoT Suite Lite software. Use Nas admin and password to login.





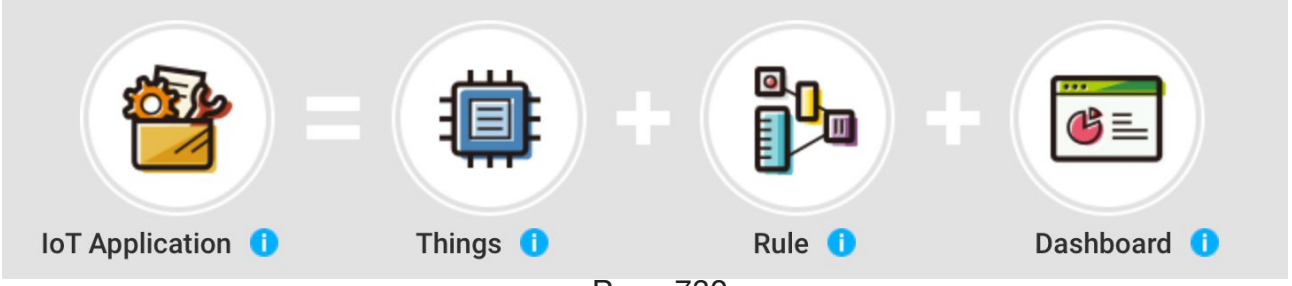
2.2 Create a new IoT application

IoT Application is a combination of multiple Things, Rule, and Dashboard. We recommend that you first create a “Things” in IoT Application. This IoT Application allows you to keep a record of all of the devices that are connected to your NAS.

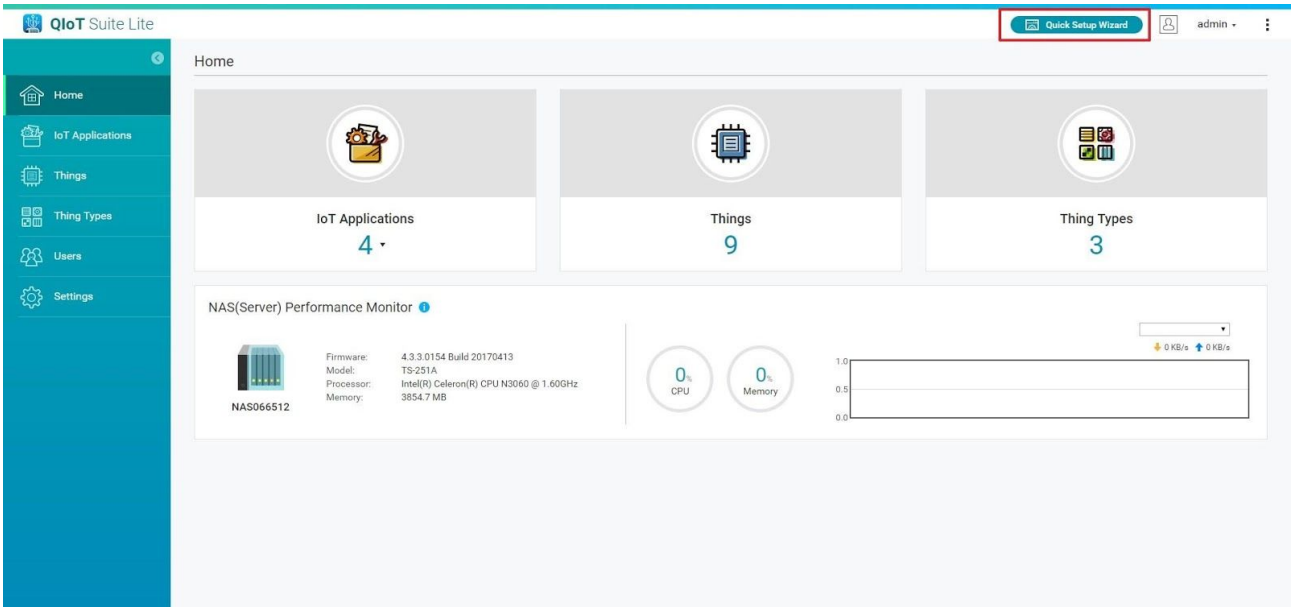
The Rule makes it possible to build IoT applications that gather, process, analyze and act on data generated by connected devices based on business rules you define. A rule can apply to data from one or many devices, and it can take one or many actions in parallel.

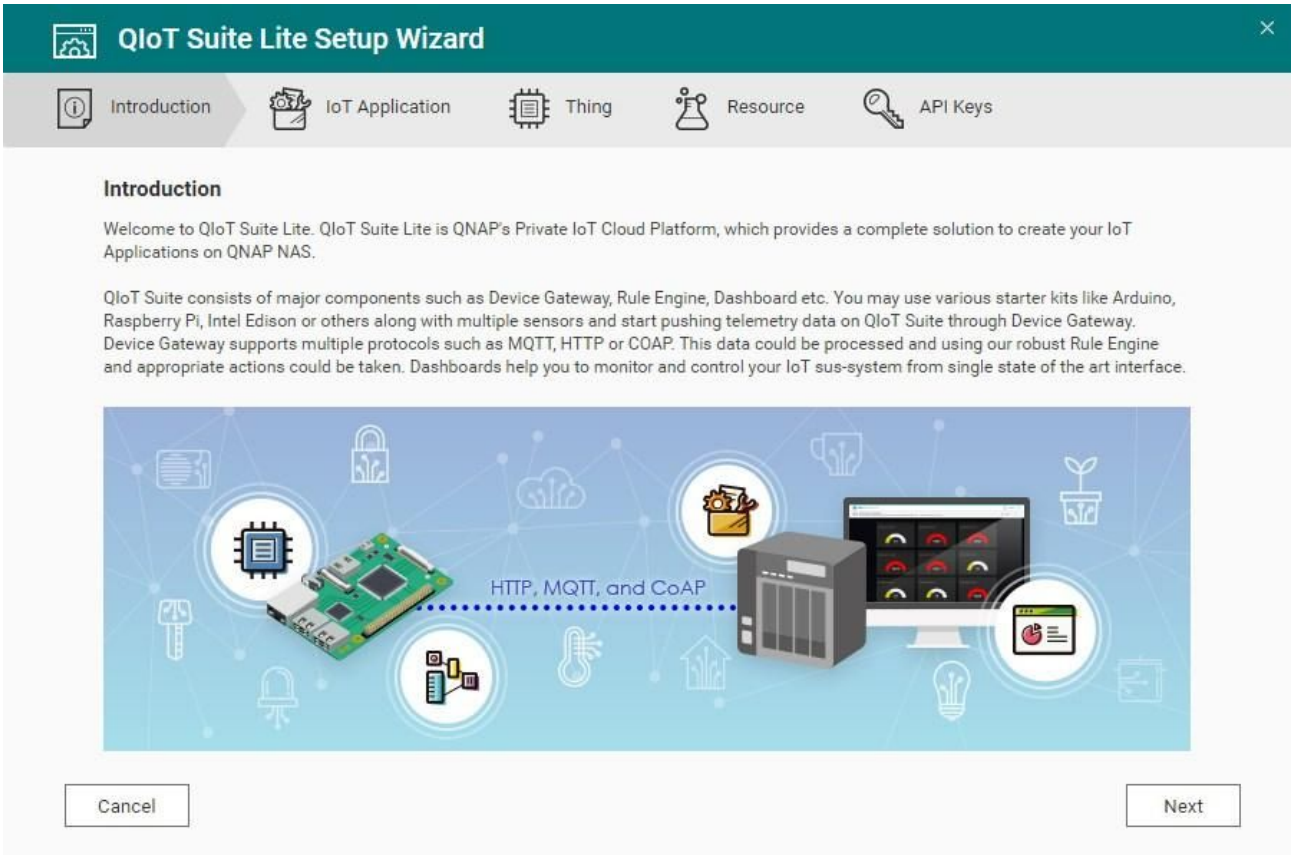
With Dashboard, you can turn your data processing efforts into analytics and reports that provide real-time insights into your business.

All these elements provide user a complete IoT Application environment.

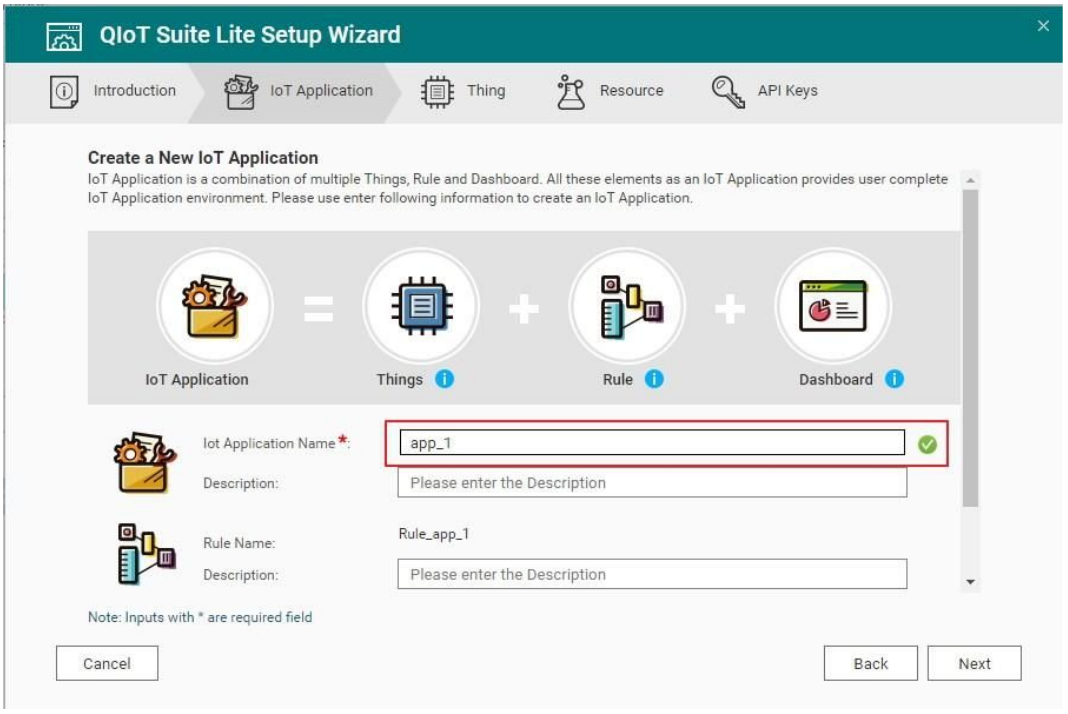


- If this is your first time to use QIoT Suite Lite, QIoT Suite Lite provides a wizard to help you quick setup a IoT application.
- Click “Quick Setup Wizard” to start quick setup wizard . After you read the QIoT Suite Lite introduction in pop window, click Next button.

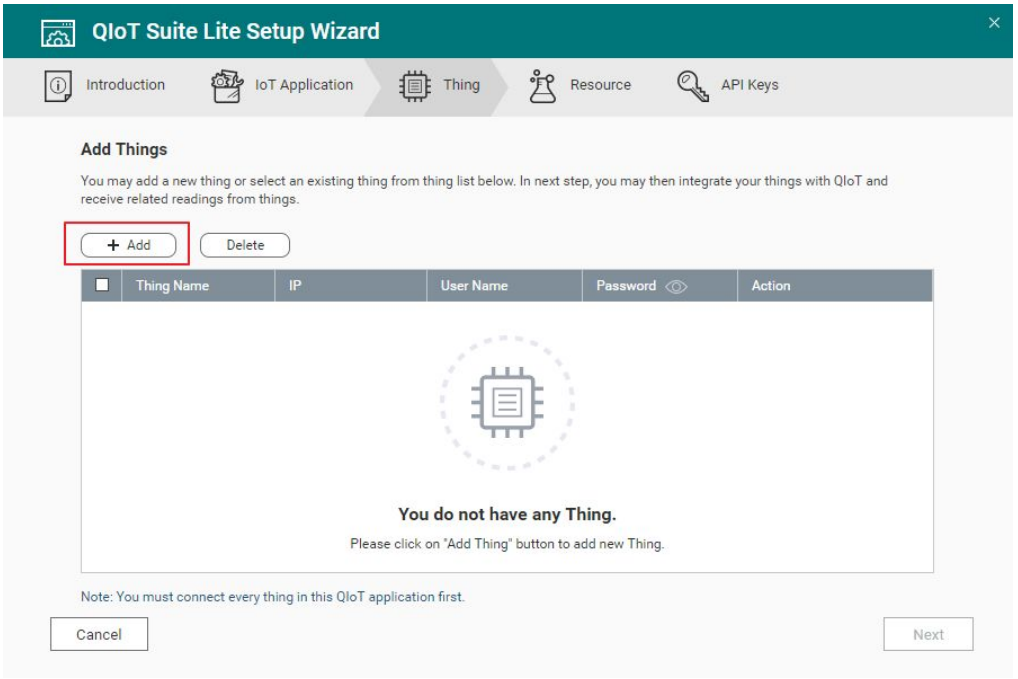




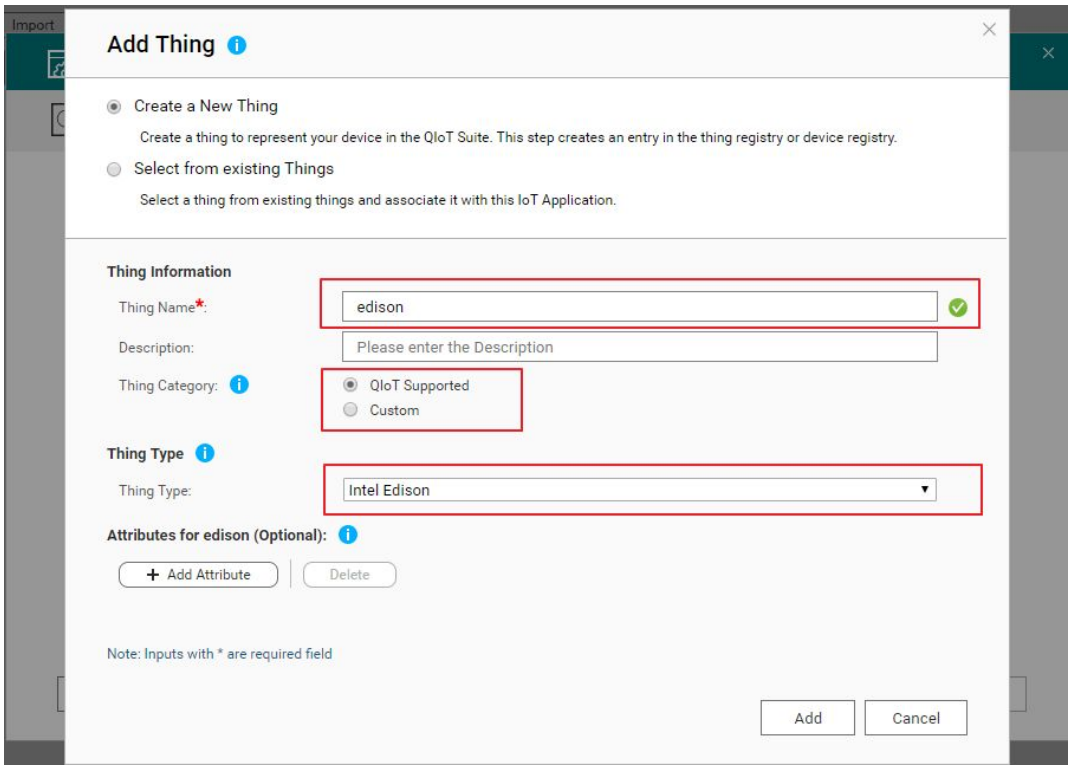
- Create a New IoT application
  - Enter IoT Application name, e.g., app_1.
  - Rule name and Dashboard name will be generated automatically based on the name of IoT Application you fill in.
  - Click “Next” to complete create a new IoT application



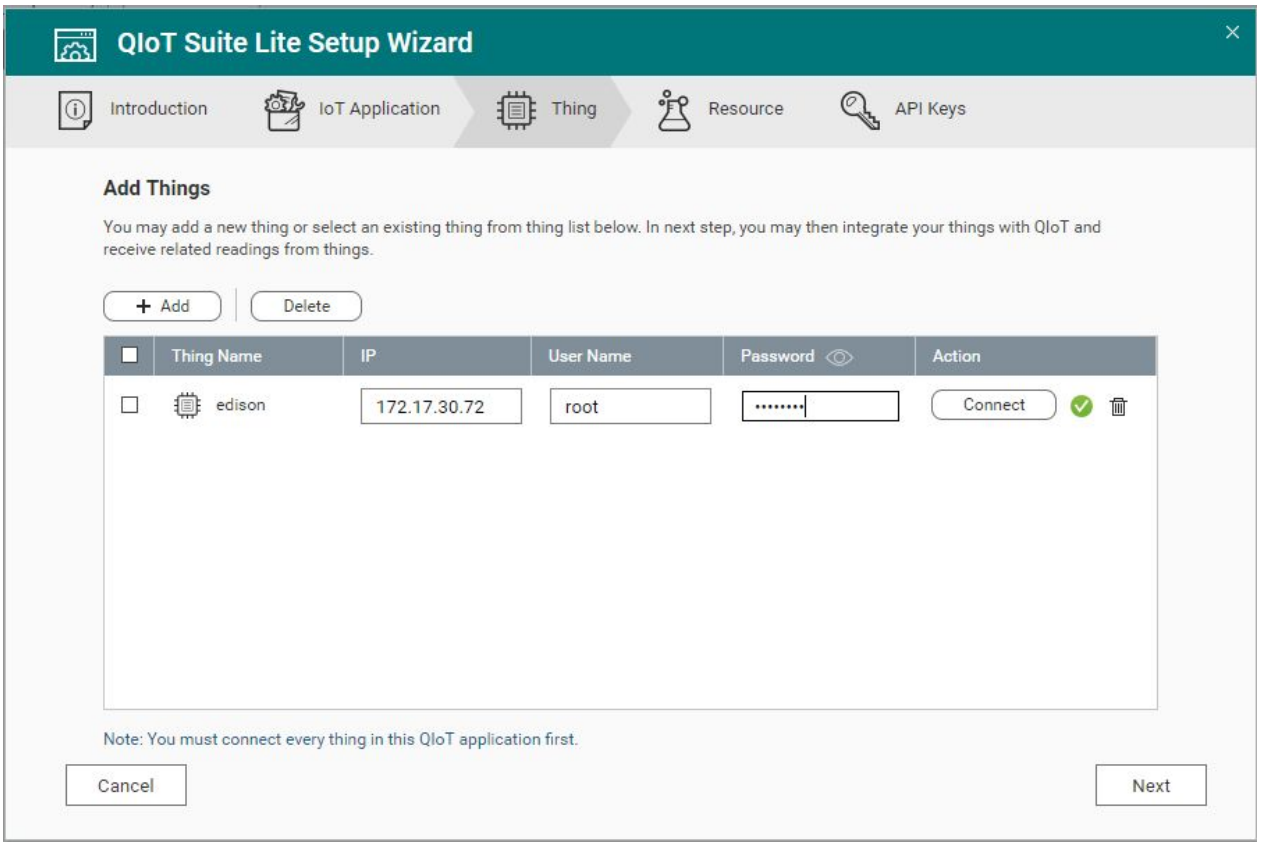
- Click “+Add” to add this application’s device.



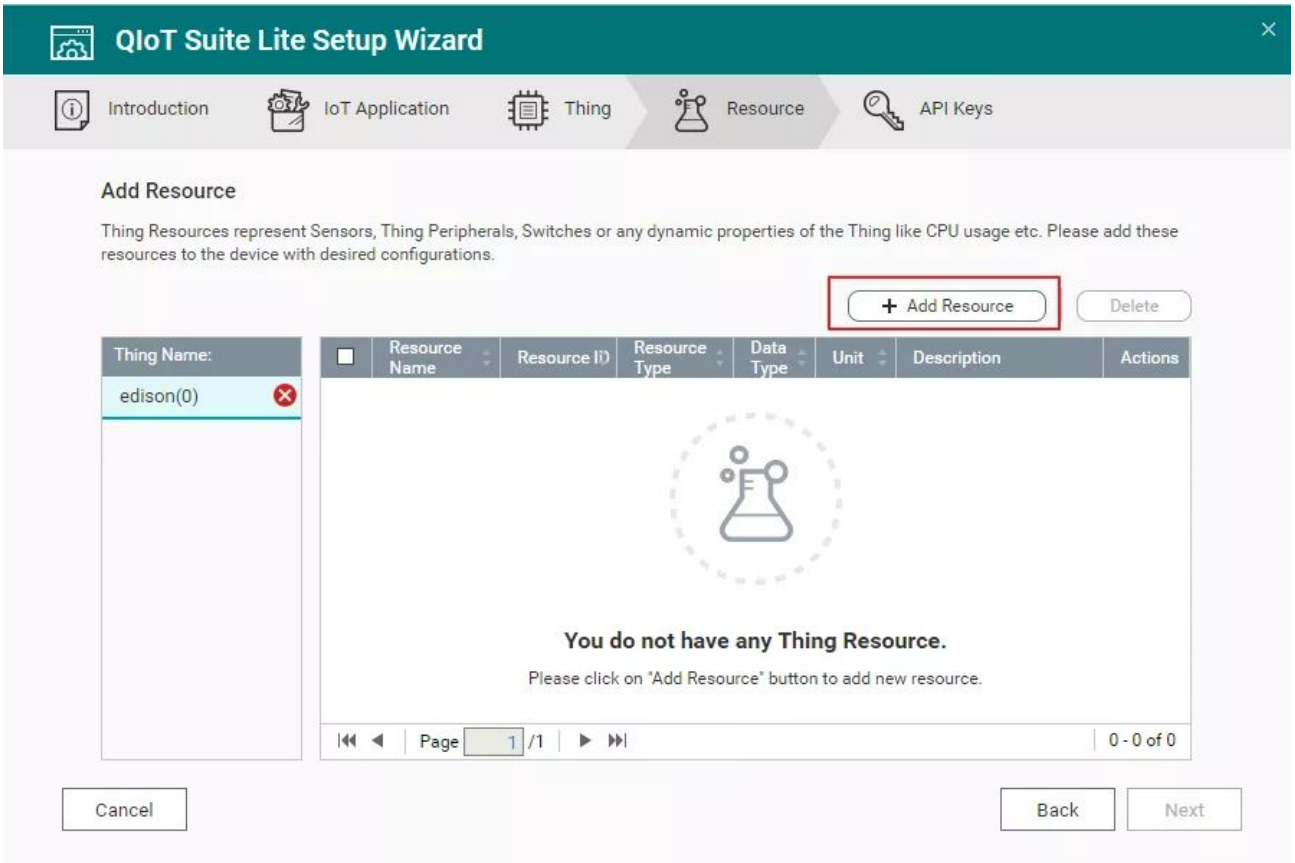
- After click “+Add ”,In “Add Thing” pop window:
  - Set device’s name (e.g., edison).
  - QIot Suite Lite now support Arduino Yun,Raspberry Pi,and Edison ,so you can select “QIoT Supported” thing category and select “Thing Type” is “Raspberry Pi”
  - (optional) add attribute to device’s detail information (e.g., its serial number, manufacturer, and more).
    Click “Add” to add the device to complete create a device.



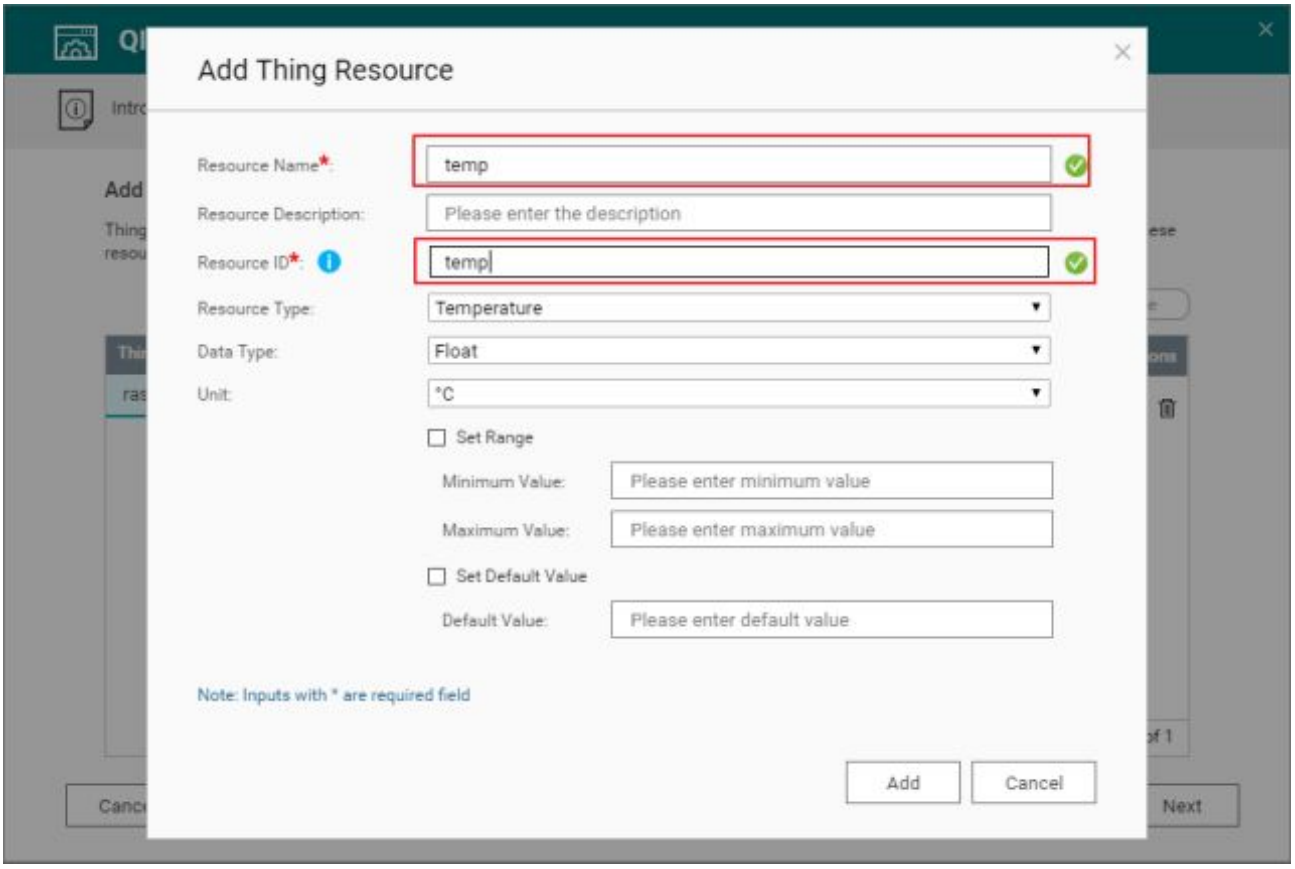
- Please provide IP address, user name, and password of your device, then click “Connect” .After waiting test connection between your device and QIot Suite Lite success, you could click “Next” to next step.



- In “Resources” step,Click “Add Resource”.

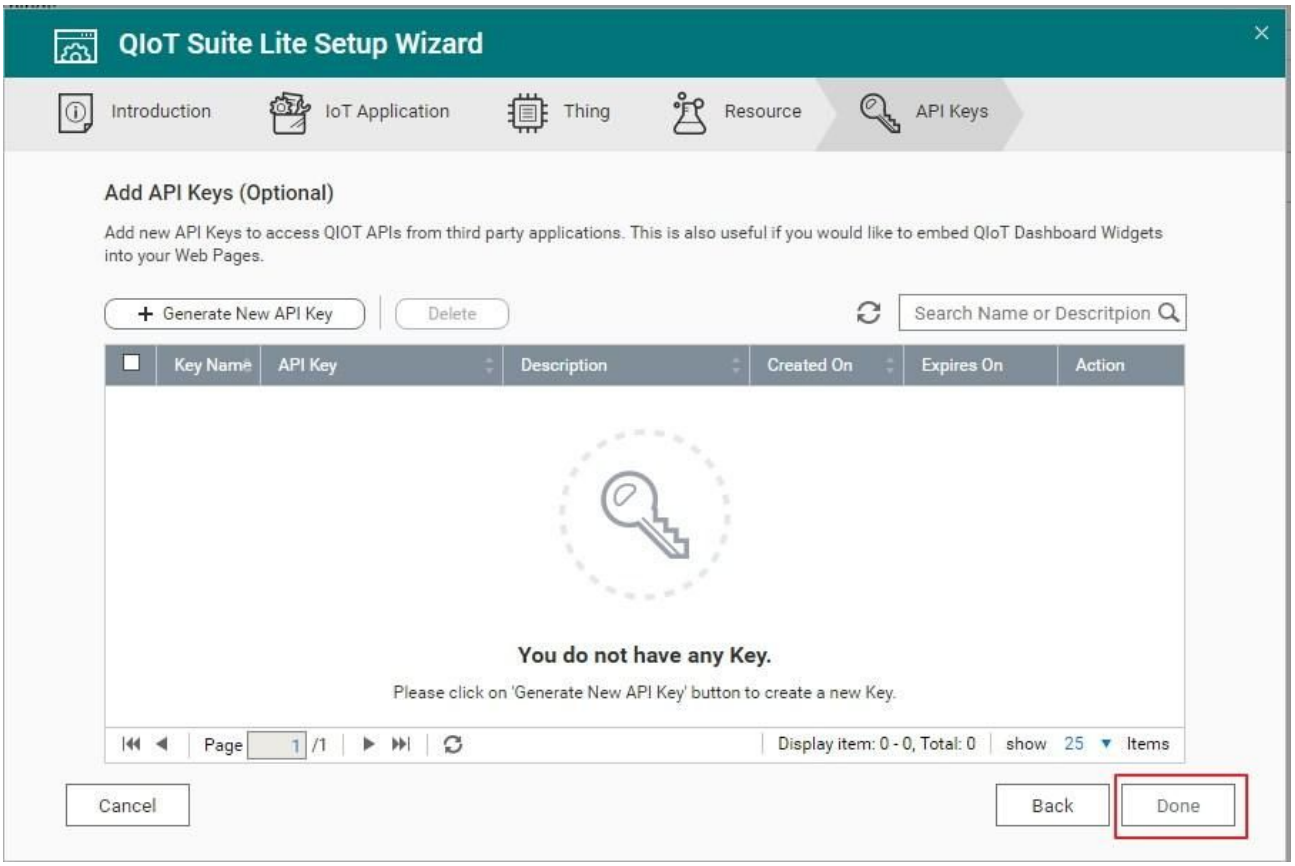


- After click “Add Resource”,In “Add Thing Resource” pop window:
    - Set resource’s name (e.g., temp).
    - Set resource’s id.Resource id will be used to create a topic in the QIoT broker.This ID should be unique for the device and no duplicates should be allowed for the same device
    - And set another optional attribute.

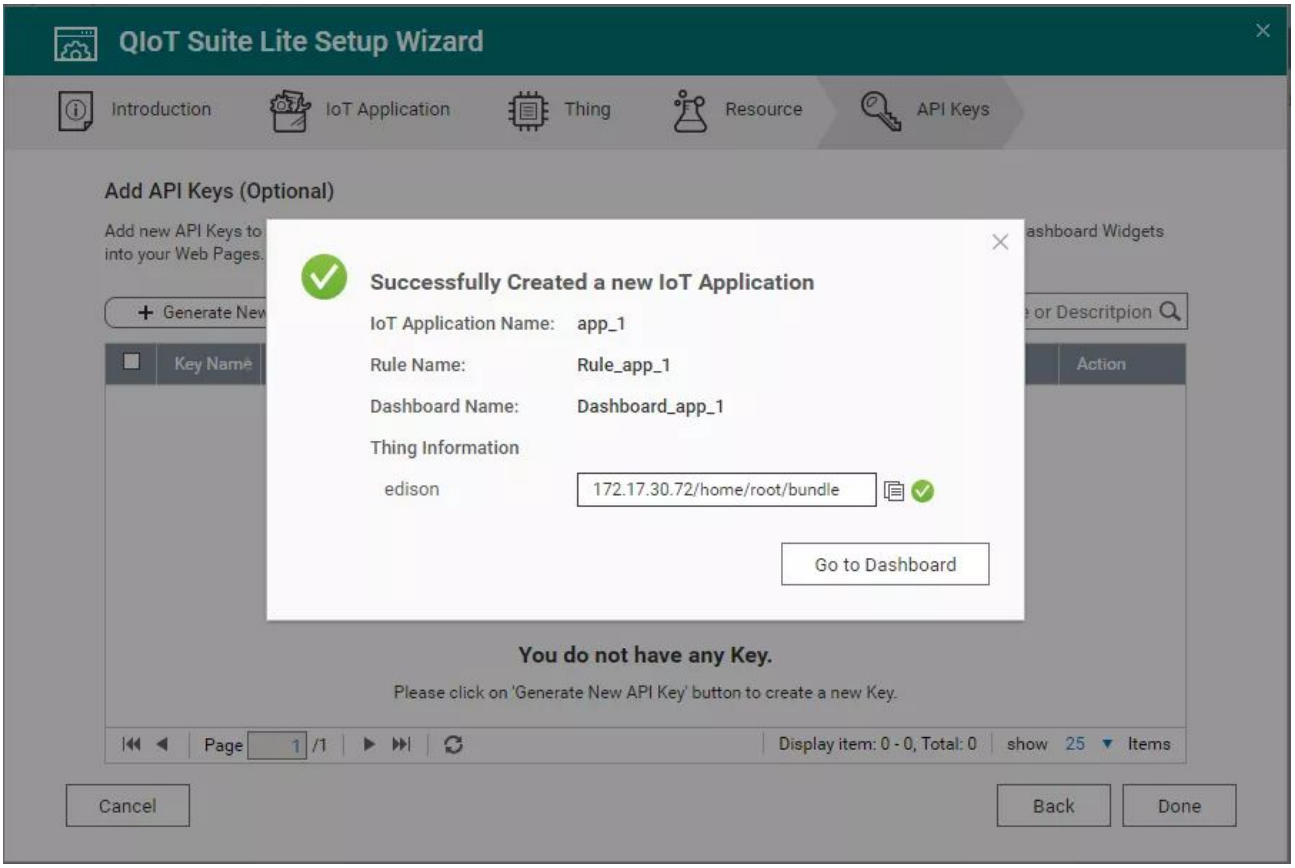


- Click Next after you add all the resources (sensors) on your device.

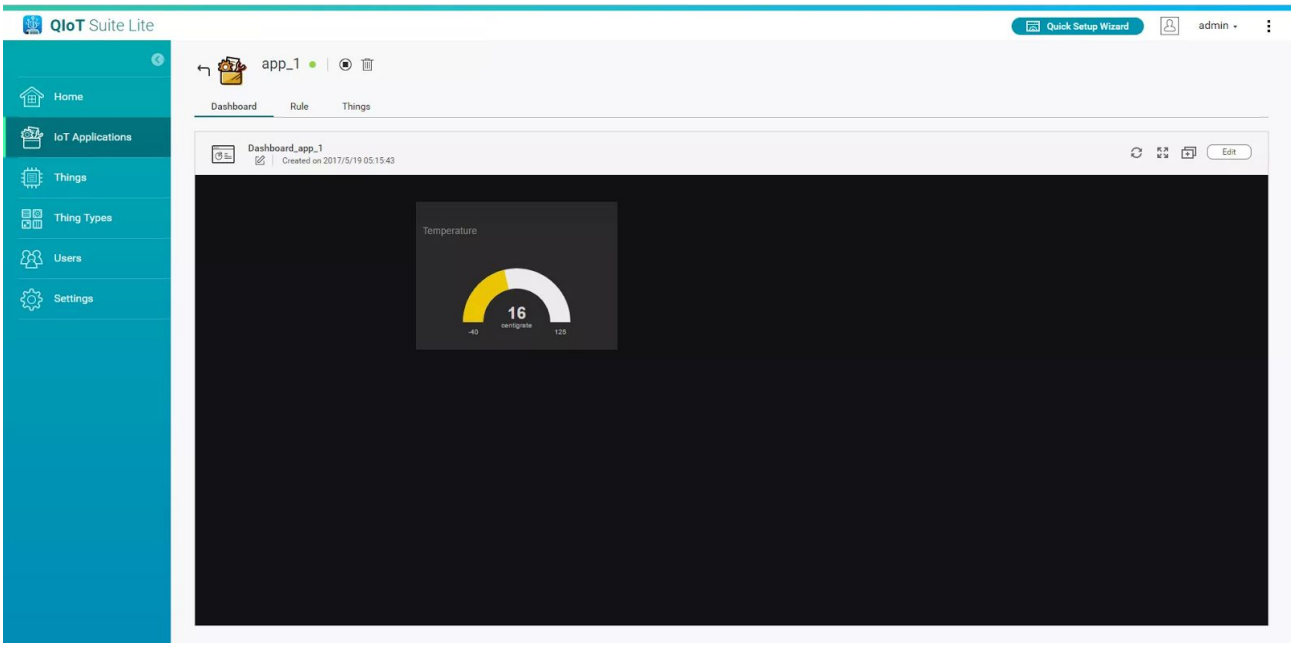
- In “API Keys” step :
    - If you would like to embed QIoT dashboard widgets into your web pages or access QIoT APIs from third party applications.You could click “generate new API Key” to create API key or click “done” start deploy sample code to your device.



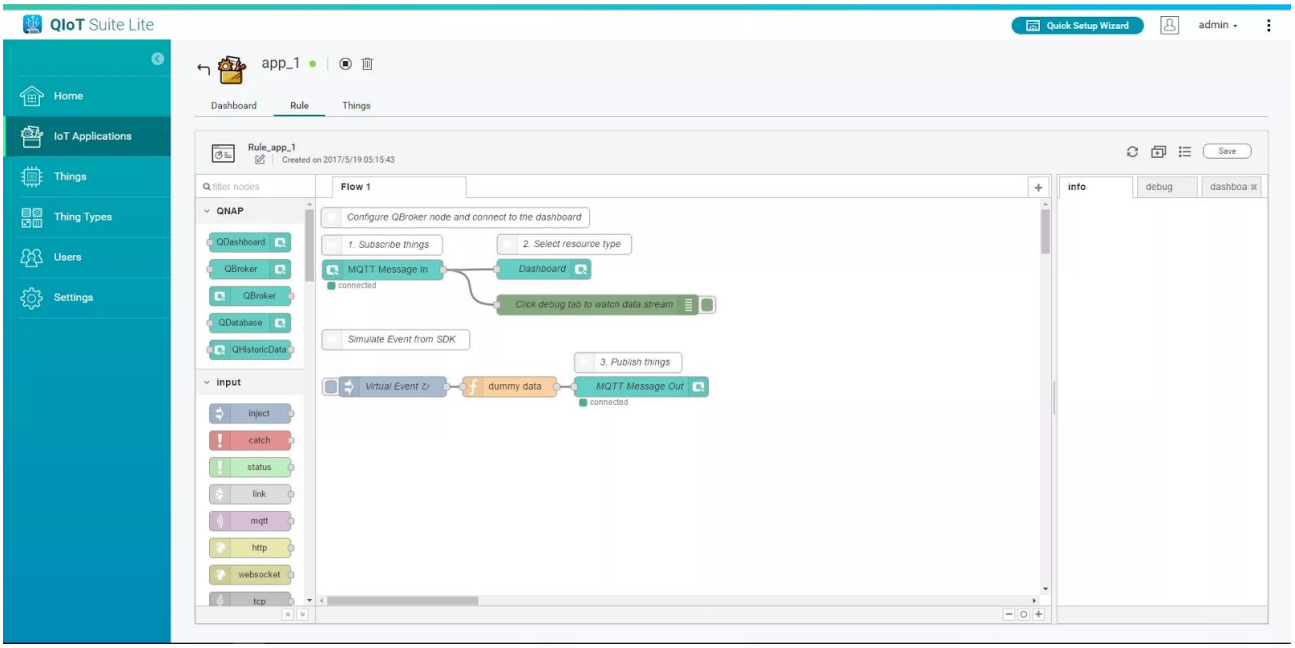
- For a while, sample codes and related files (certificate, resource information) have been upload to the specified path on your device.



- Your IoT application already created successfully .You could click”Go to Dashboard” to your application page.
- Select “Dashboard” tab , you could see a sample dashboard is created.

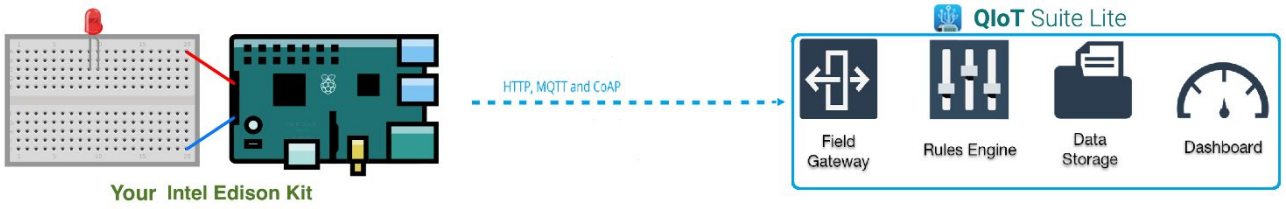


- Select “Rule” tab, you could to define the flow or rule about how to process the data sent from the device, and how to present in dashboard.



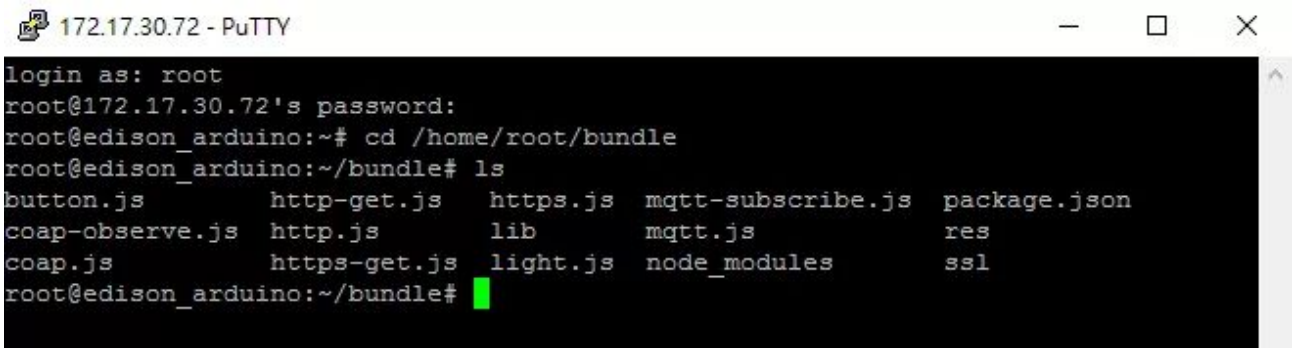
Lesson 3: Connect your device to QIoT Suite Lite using MQTTS

In this lesson, you generate certificate from QIoT Suite Lite, download SDK, and connect Intel Edison to QIoT Suite Lite.



3.1 Run Sample Code

- Open Terminal application (e.g., PuTTY) on your PC.Connect to your device by SSH and enter the folder where put sample code (e.g., /home/root/bundle).



- Install sample code dependency,enter command as following
  root@Edison:~ $ **cd /home/{{user}}/bundle**
  root@Edison:~ $ **npm install**

- Run sample code in device will publish message to topic “temp” by MQTTS as following picture. Topic is define from resource id that you setted.

```
/**
 * Send data to QIoT Suite Lite.
 * content of ./res/resourceinfo.json
 * {
 *     ...
 *     "resources": [
 *         {
 *              ...
 *              "resourceid": "temp",
 *              "topic": "qiot/things/admin/abccccc/temp",
 *              ...
 *          }
 *      ]
 *  }
 */
setInterval(function() {
    // TODO: you could replace "temp" by any resource id set form QIoT Suite Lite
    connection.publishById("temp", getRandomInt(0, 50));

    // or publish by resource topic
    // TODO: you could replace "qiot/things/admin/edison/temp" by any Topic form QIoT Suite Lite like following
    // connection.publishByTopic("qiot/things/admin/edison/temp", getRandomInt(0, 50));

}, 1000);
```

- Run the sample application.

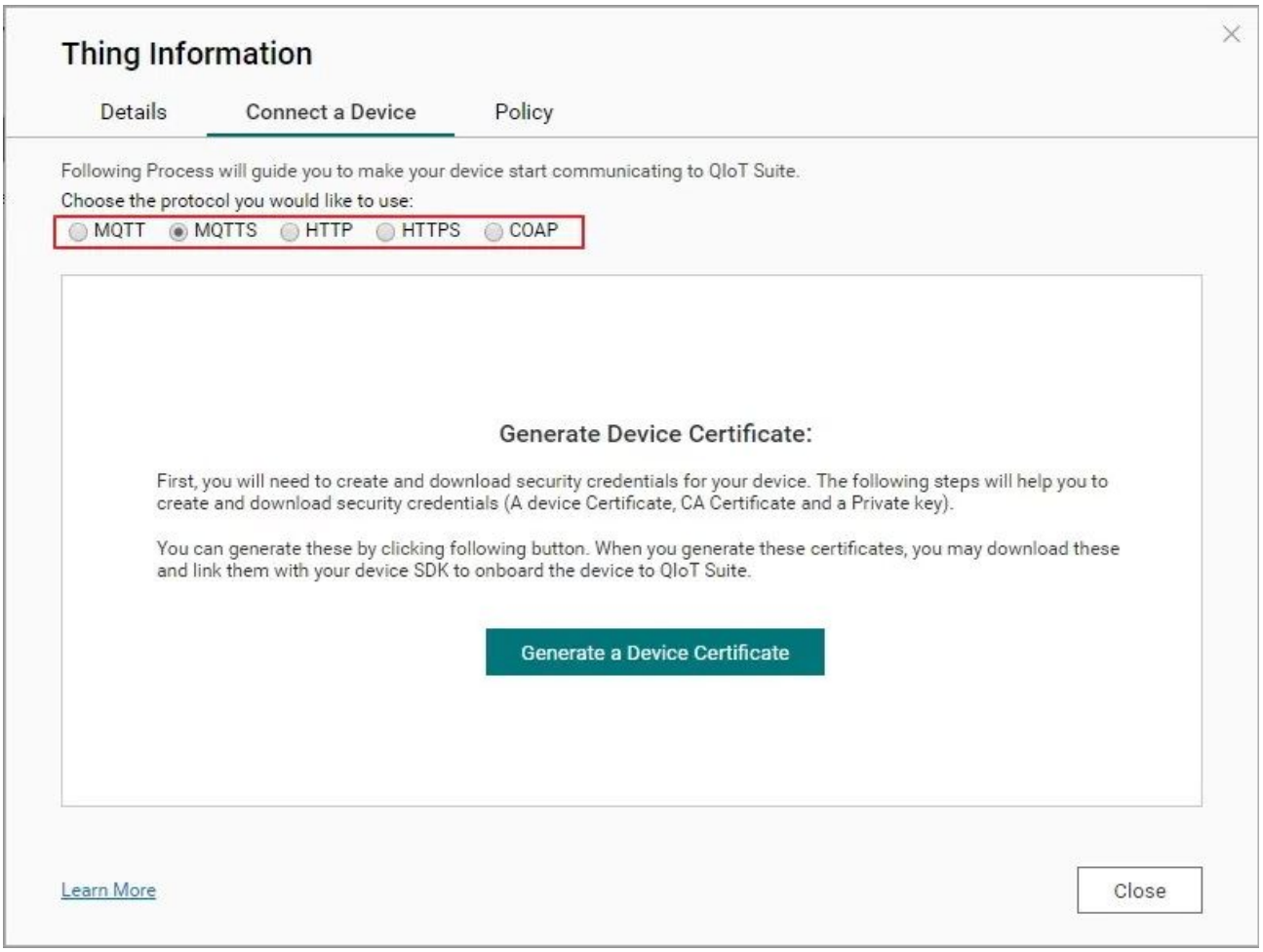
```
root@Edison:~ $ node mqtt.js
```

- **device will send message to topic “temp” or that you defined ,as below image.**

```
root@edison_arduino:~/bundle# node mqtt.js
privatekey full path = 3176652017-05-16_10-31-47_privatekey.pem
ready to conneciton:
 send message to [mqtt(s)://172.17.30.174:8883], topic_Pub = qiot/things/admin/e
dison/temp, value = {"value":48}
 send message to [mqtt(s)://172.17.30.174:8883], topic_Pub = qiot/things/admin/e
dison/temp, value = {"value":31}
 send message to [mqtt(s)://172.17.30.174:8883], topic_Pub = qiot/things/admin/e
dison/temp, value = {"value":16}
 send message to [mqtt(s)://172.17.30.174:8883], topic_Pub = qiot/things/admin/e
dison/temp, value = {"value":28}
 send message to [mqtt(s)://172.17.30.174:8883], topic_Pub = qiot/things/admin/e
dison/temp, value = {"value":28}
 send message to [mqtt(s)://172.17.30.174:8883], topic_Pub = qiot/things/admin/e
dison/temp, value = {"value":44}
 send message to [mqtt(s)://172.17.30.174:8883], topic_Pub = qiot/things/admin/e
dison/temp, value = {"value":43}
 send message to [mqtt(s)://172.17.30.174:8883], topic_Pub = qiot/things/admin/e
dison/temp, value = {"value":16}
 send message to [mqtt(s)://172.17.30.174:8883], topic_Pub = qiot/things/admin/e
dison/temp, value = {"value":48}
```

3.2 Another protocol

- Click “Connection a device” button
- You can choose another protocol you would like to use



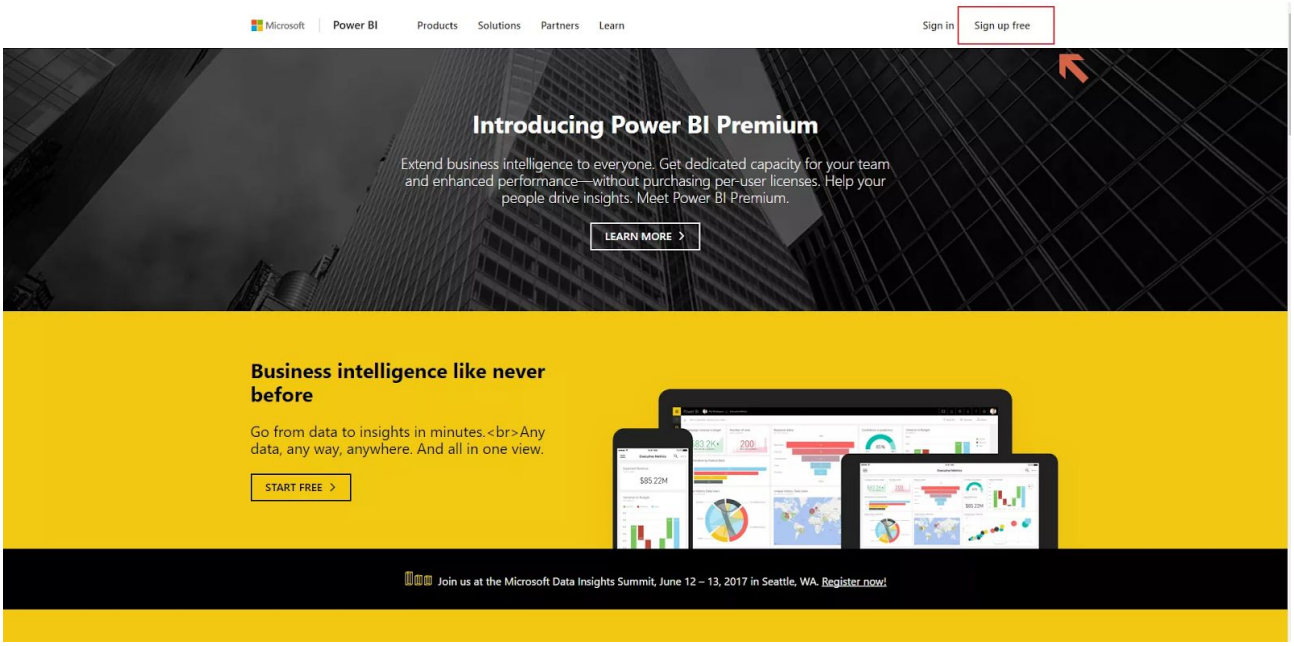
- SSH to your device , and input command as following.

```
root@Edison:~# cd /home/{{user}}/bundle
// mqtt(dont' need certificate,just put JSON file to "res" folder):
root@Edison:~/bundle# node mqtt.js
// http
root@Edison:~/bundle# node http.js
// https
root@Edison:~/bundle# node https.js
// coap
root@Edison:~/bundle# node coap.js
```

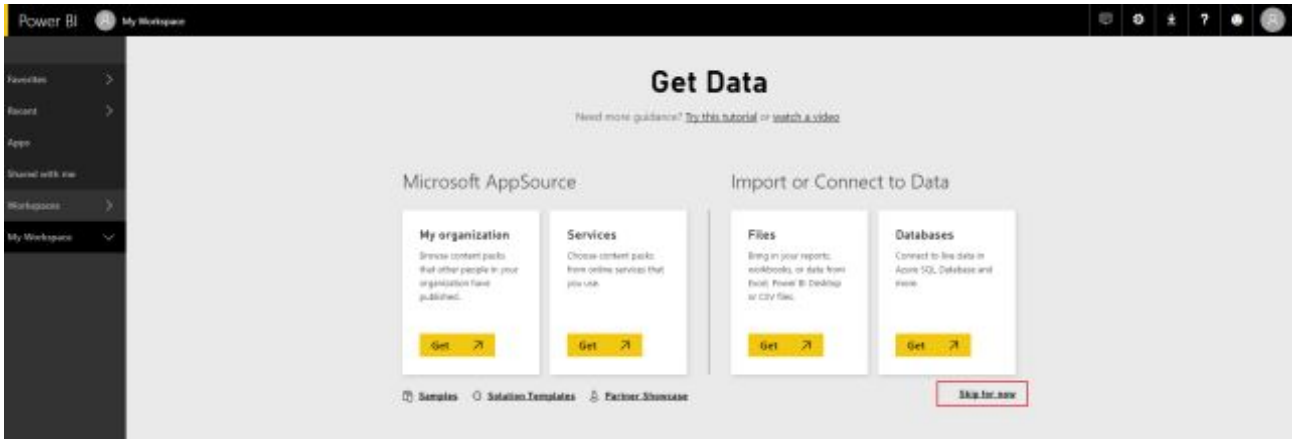
Lesson 4: Integrate Power BI

4.1 Get your first Power BI account

- Go to the offical website “https://powerbi.microsoft.com/en-us/” to sign up your free account.



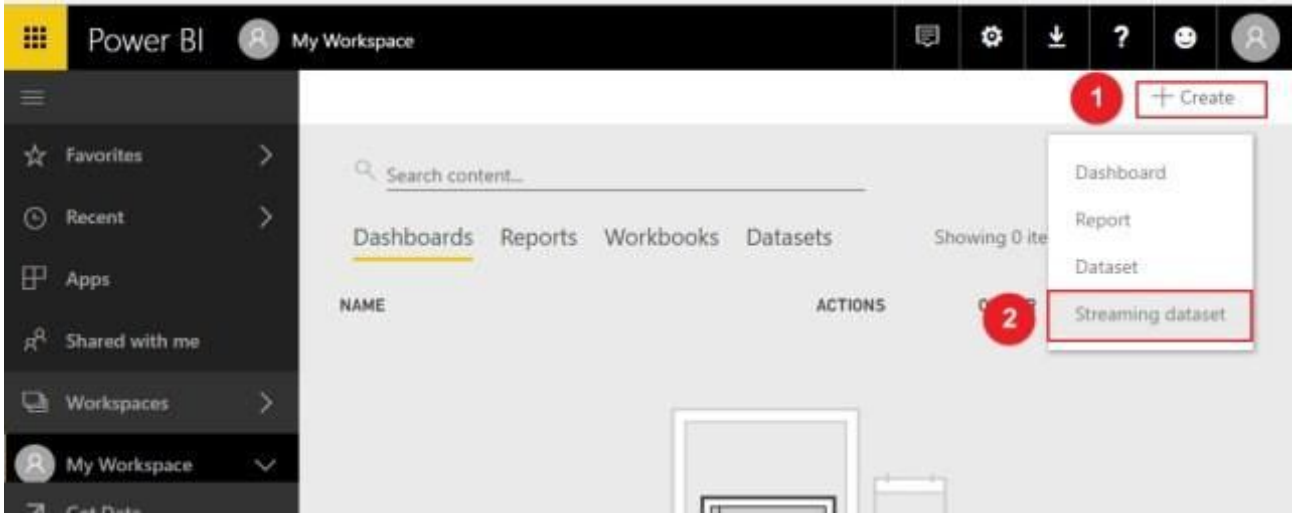
- After a sequence of registration, the page will lead you to below page,you can press “My workspace”, and “skip for now” button appear. You could click “skip for now “ to start create dataset.



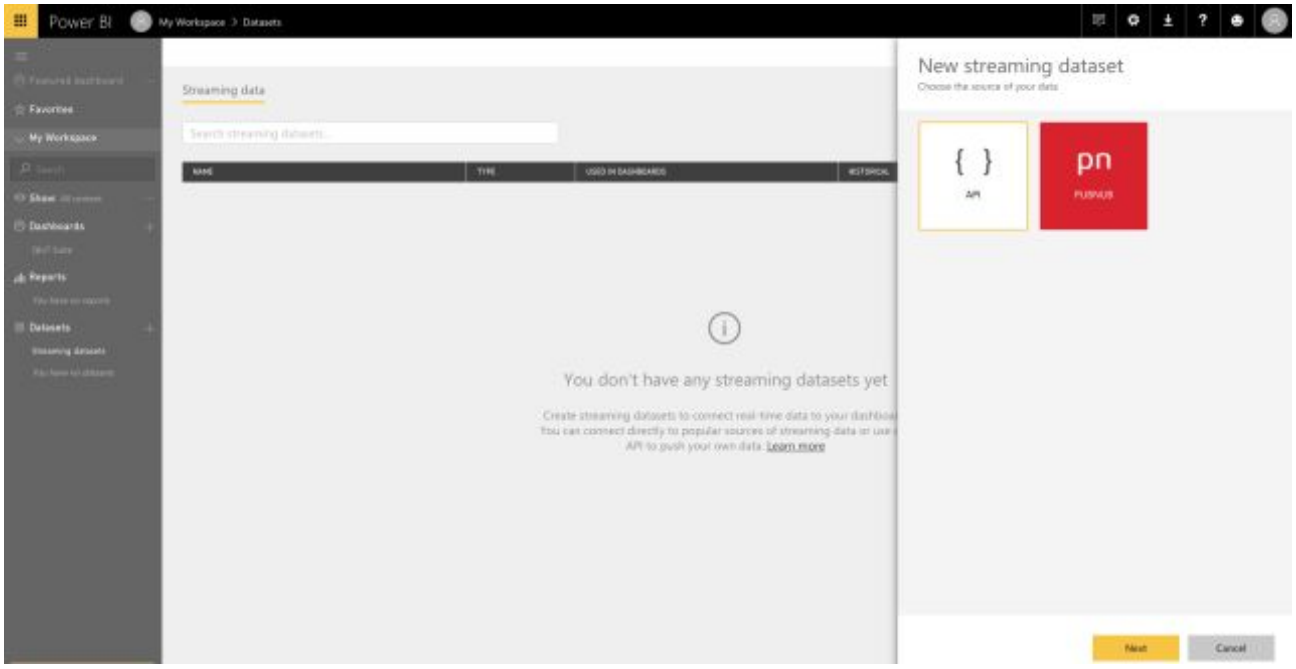
4.2 Setup your streaming dataset API

- Create “Datasets”
  - Click “Create” in scrren upper right corner

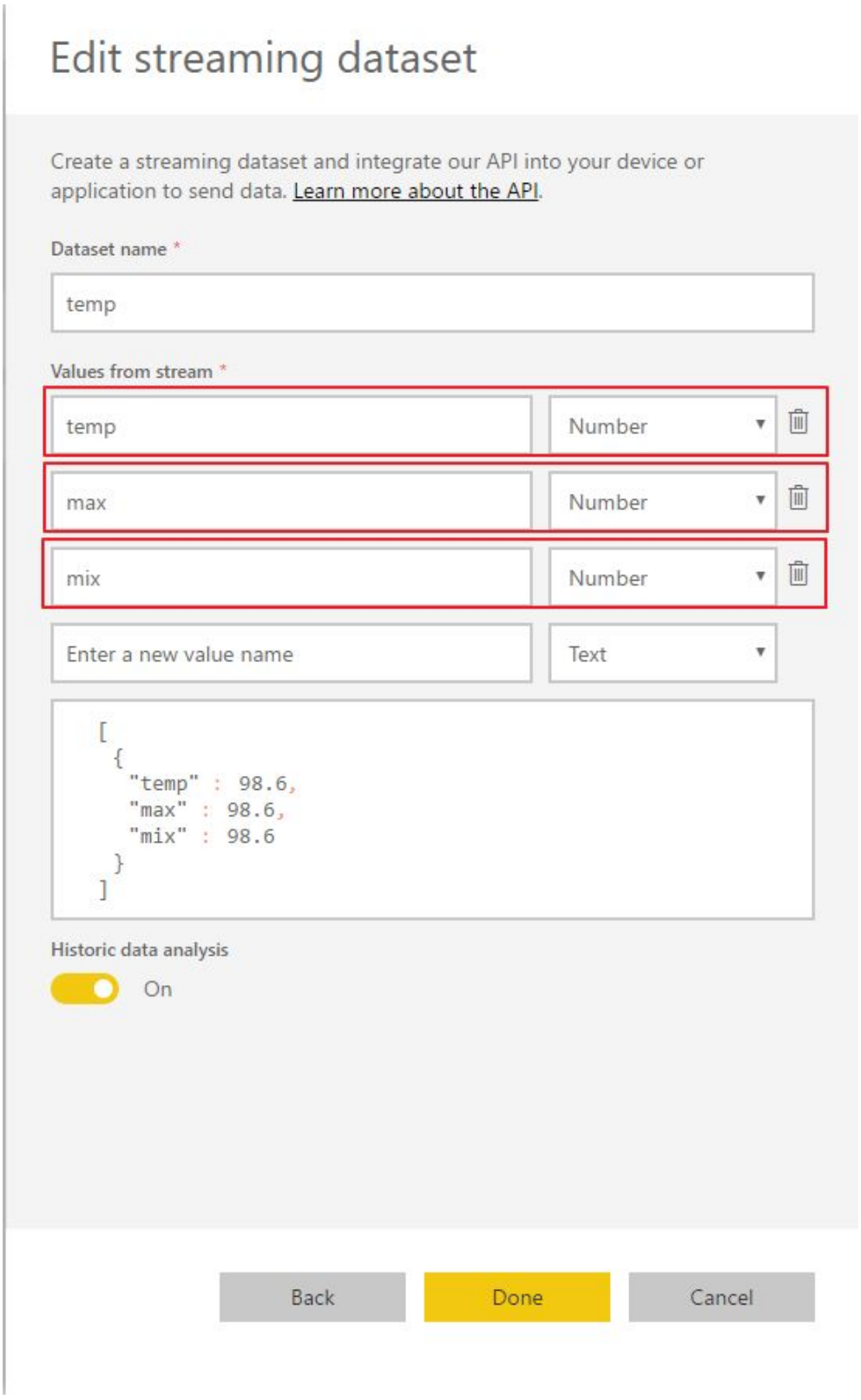
- And then click “Streming dataset”



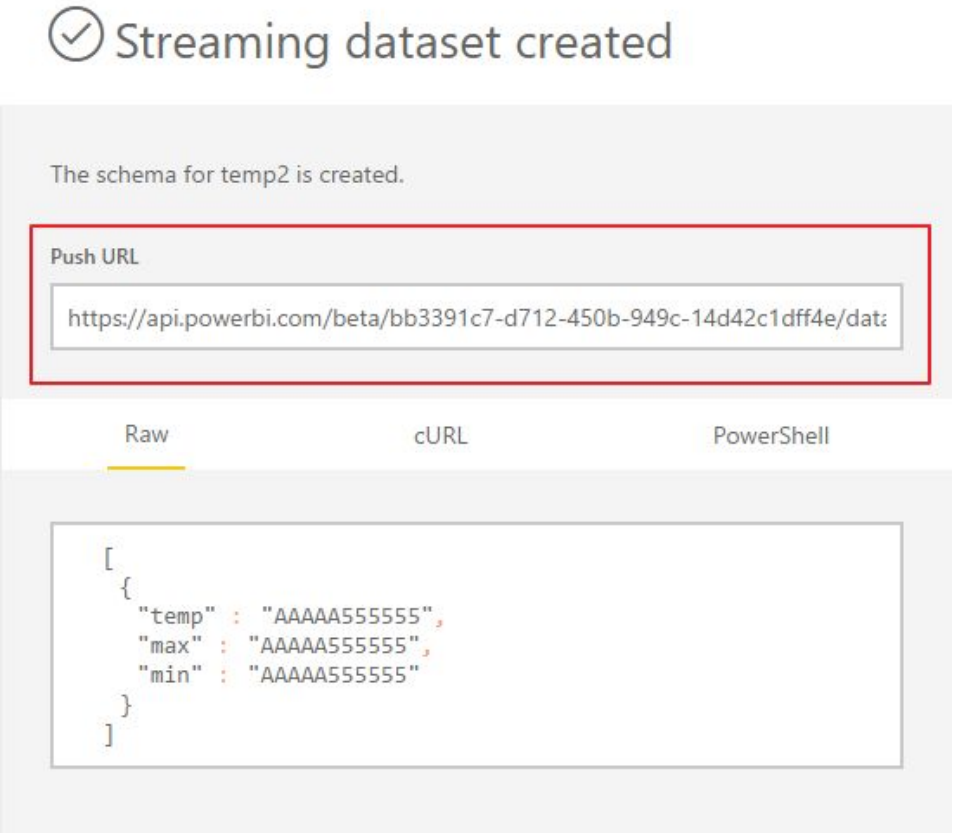
- Select “API”, and click “Next”.



- Define your values from stream(e.g., temp,max,min), and you will get a result of JSON in textbox. QIot Suite’s application will post this data format to Power BI. Click the “Create” button to finish create streaming dataset.

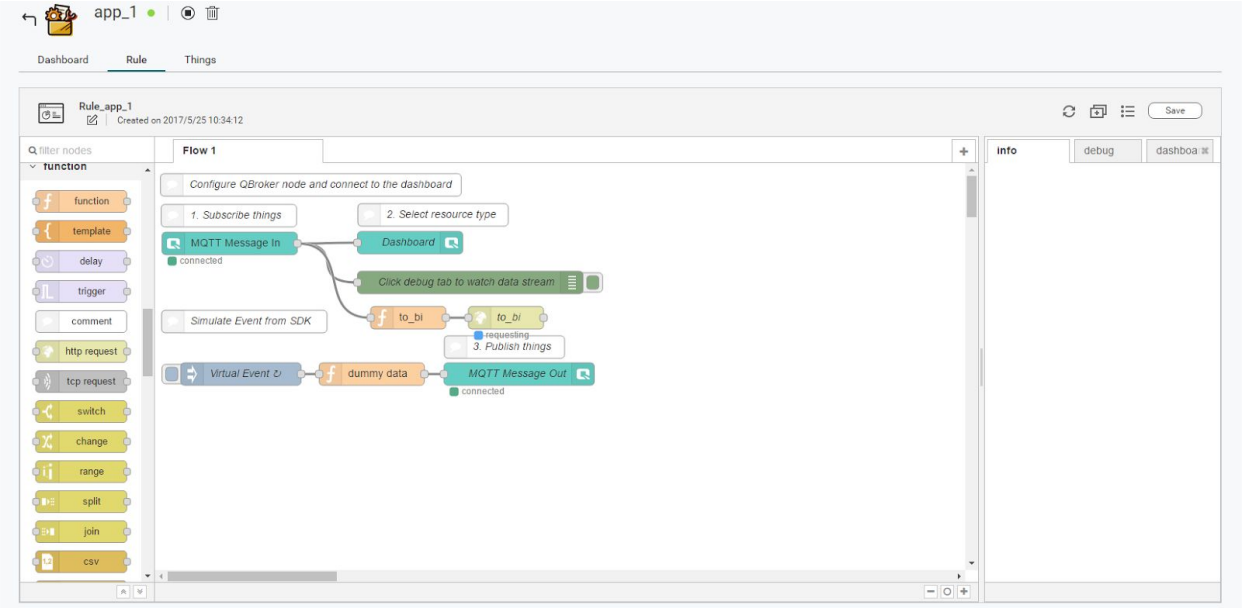


- Once you successfully create your data stream, you get REST API URL which QIoT suite application can call using POST request to push your live data to streaming data dataset you created.



4.3 Configure Node-RED's nodes in IoT application

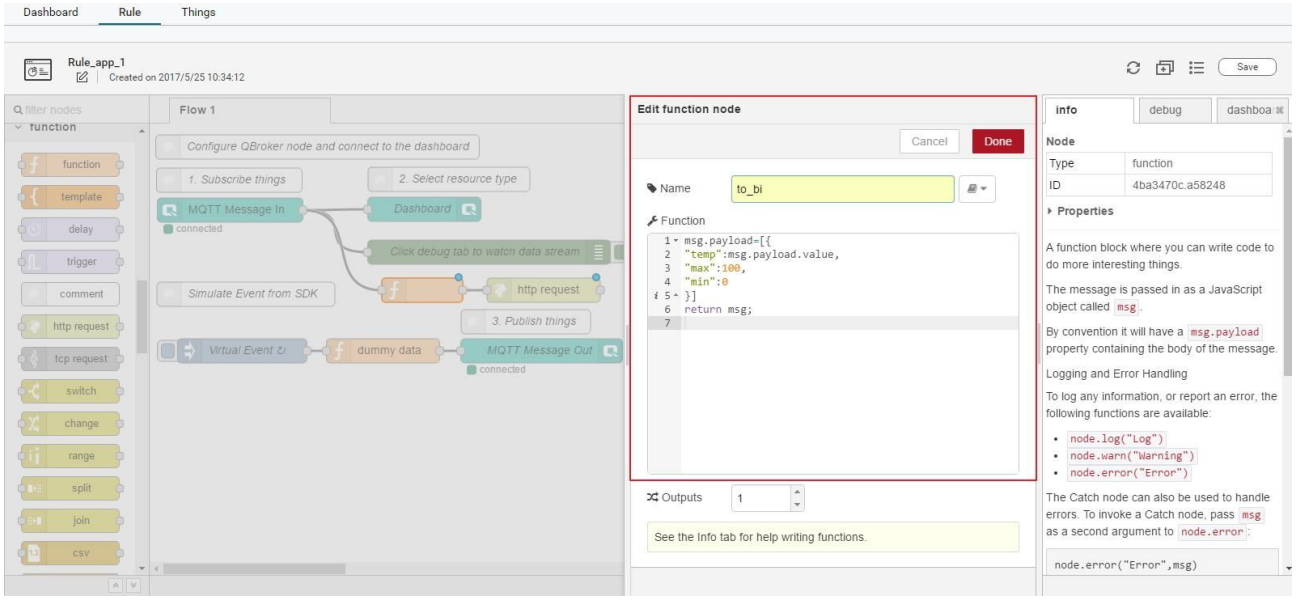
- Create IoT application in QIoT suite.
- The following one is your first node-red flow, and then you can start to create your own IoT flow. more node-red information can be found in "Node-Red".



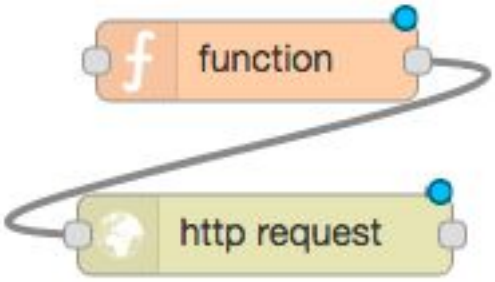
- Before you start to push live data to Power BI.
- We need a "function" node to convert IoT data to streaming data dataset. Here you can replace msg.payload to your JSON dataset.

- Double click function node, and type Function code as following:

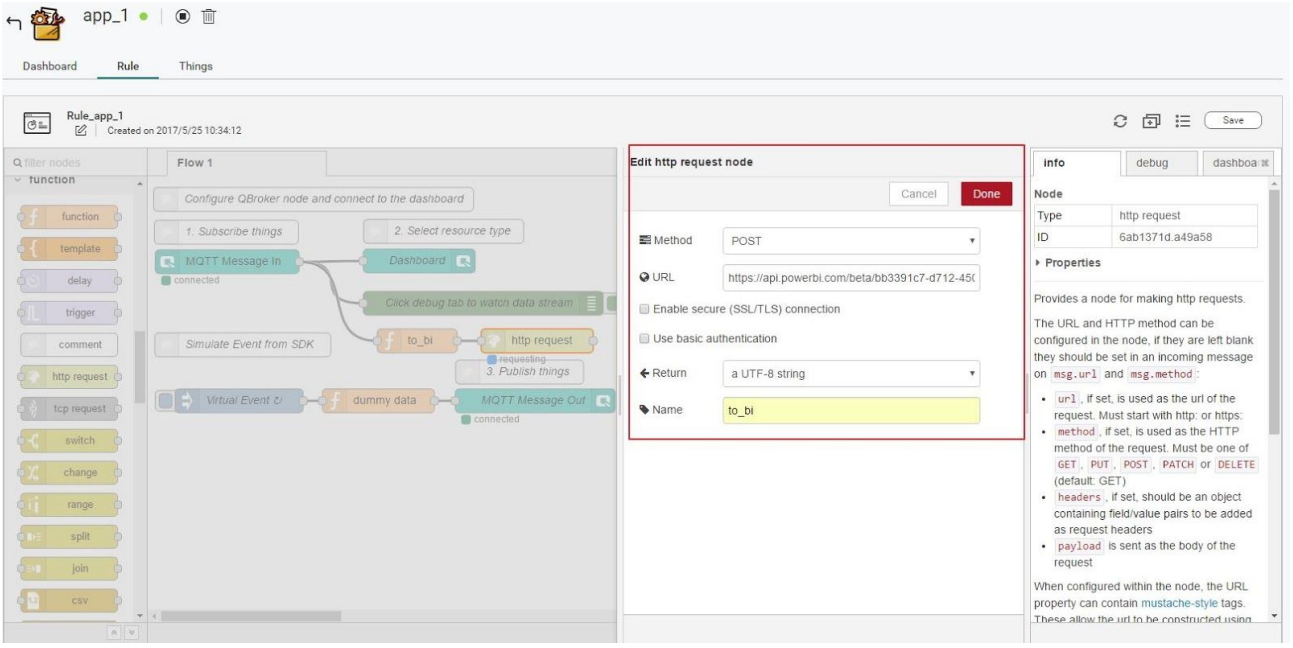
```
msg.payload=[{
        "temp":msg.payload.value,
        "max":100,
        "min":0
}]
return msg;
```



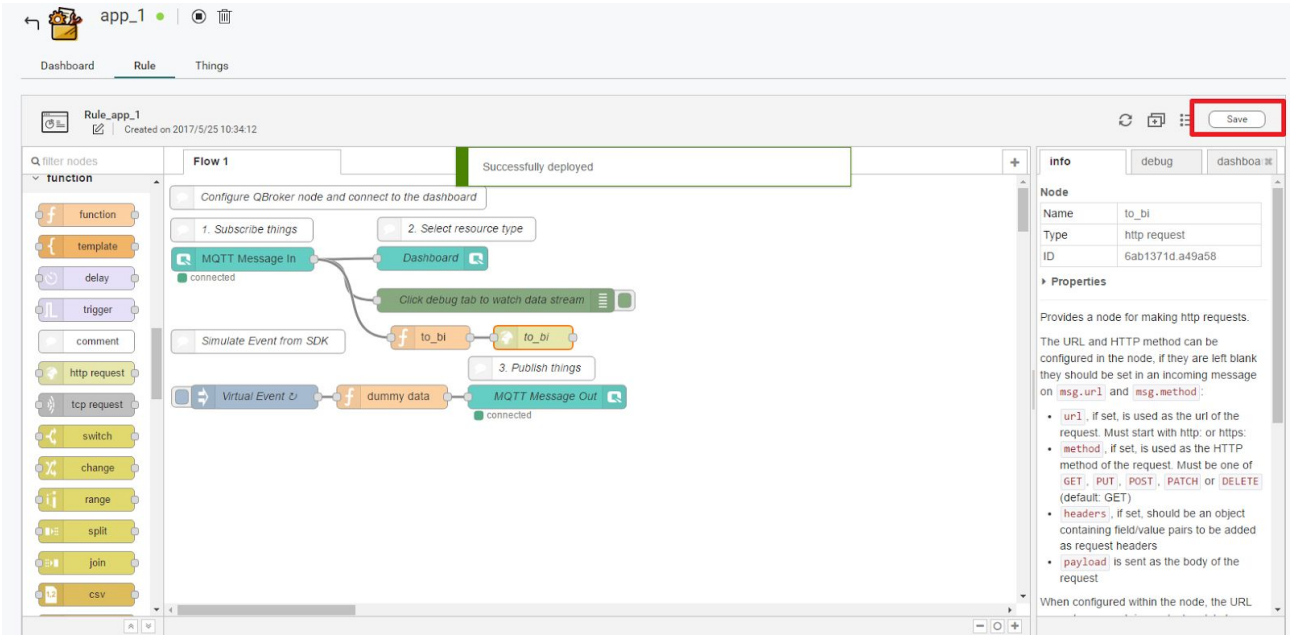
- We need a “http request” node to help us to push live data to Power BI. Just drop and drag “http request” node and connect to tail of “function“ node.



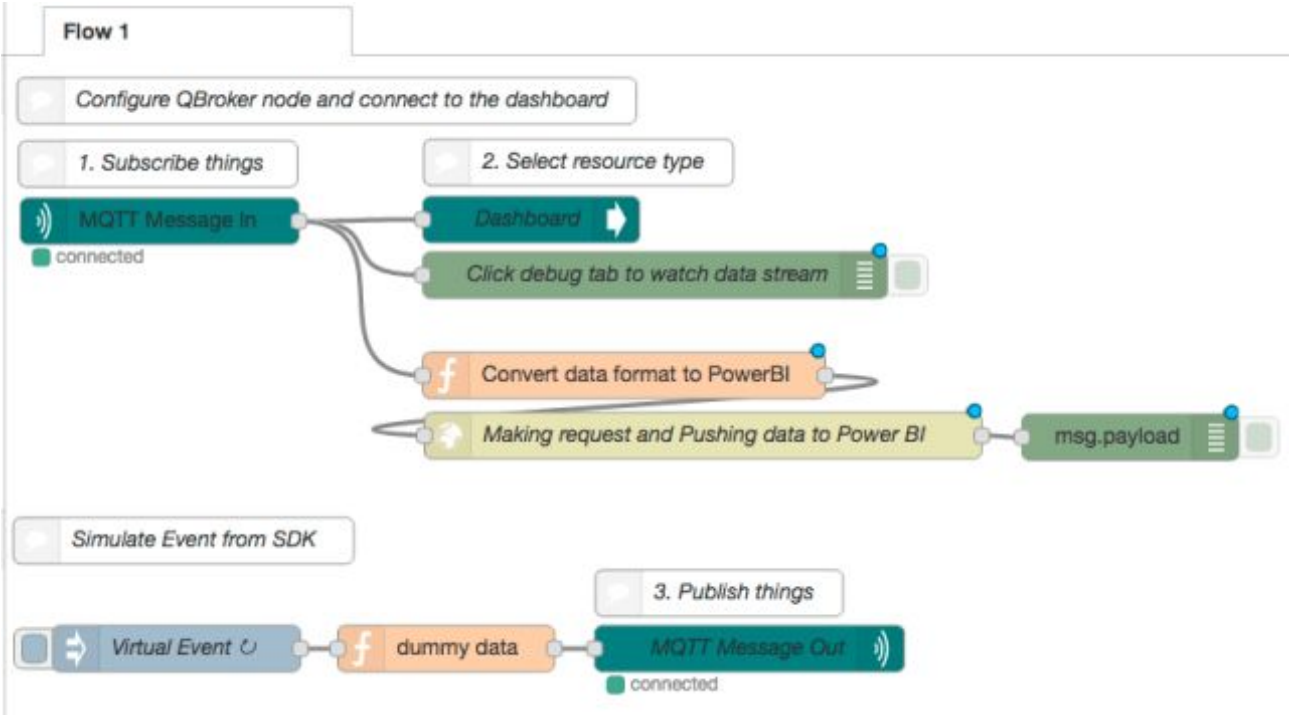
- Double click http request node, copy and paste REST API URL that you got from Power BI console, and set http method to POST.



- Finally, don’t forget to press “Save” button to save changes.

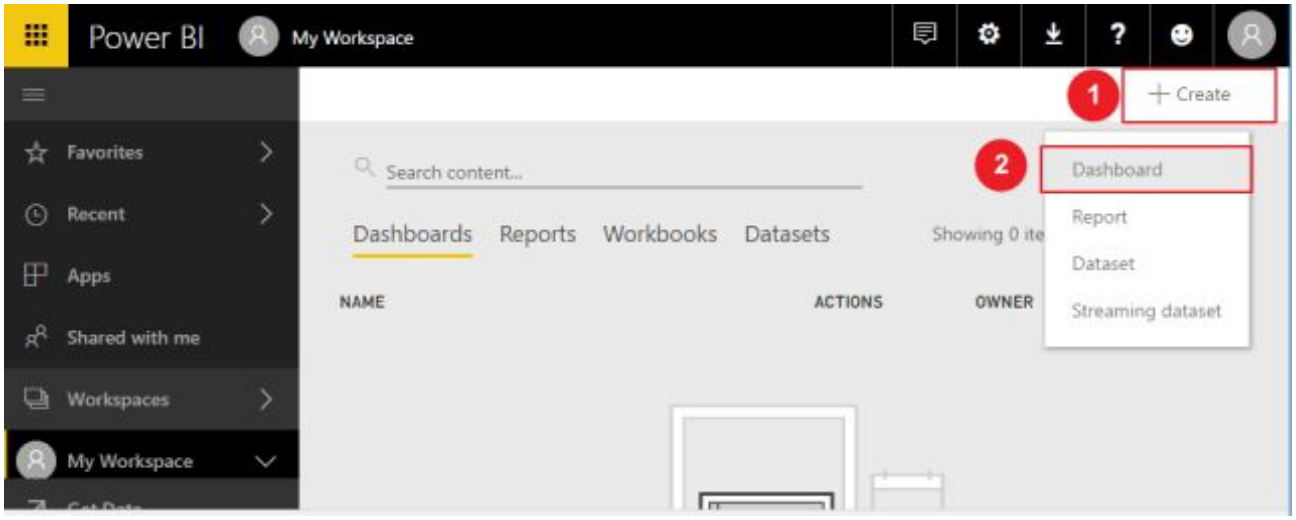


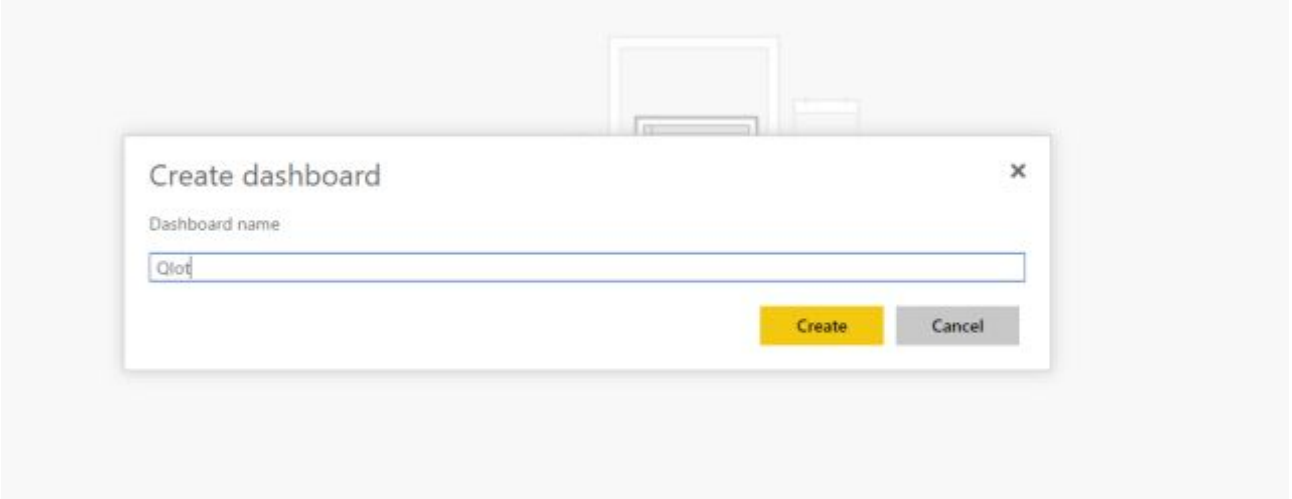
- Finally, your node-red flow will look like below one.



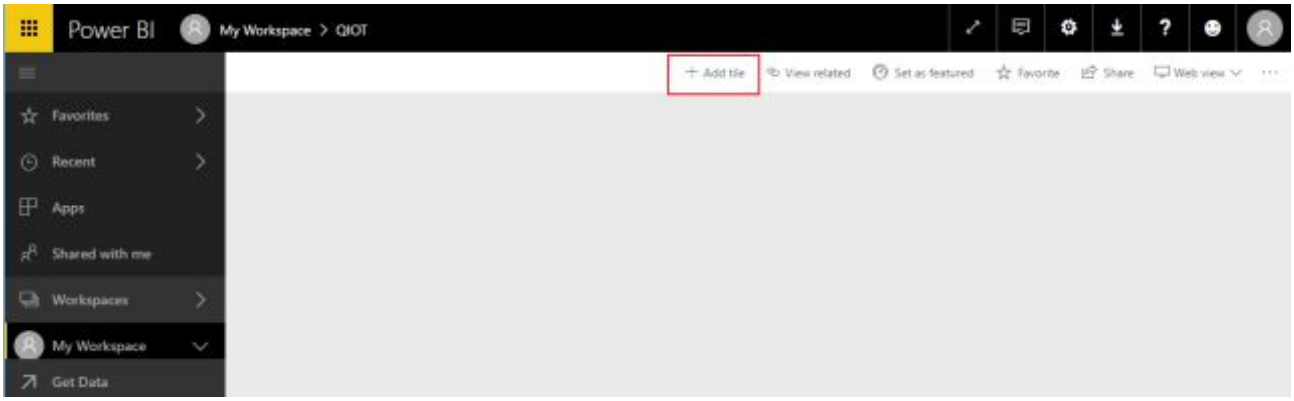
4.4 Add tile to display real-time data

- Create “Dashboard”
  - Click “Create” in scrren upper right corner
  - And then click “Dashboard”
  - Enter dashboard’s name,and click “Create” to complete create dashboard.

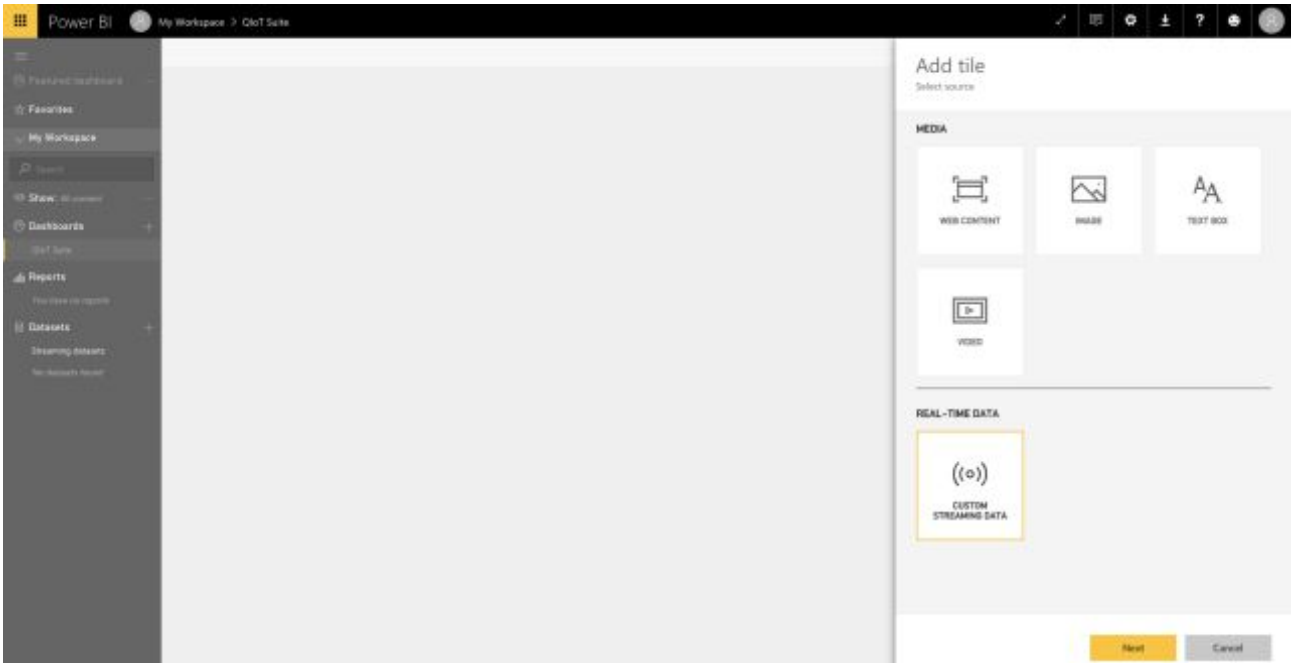




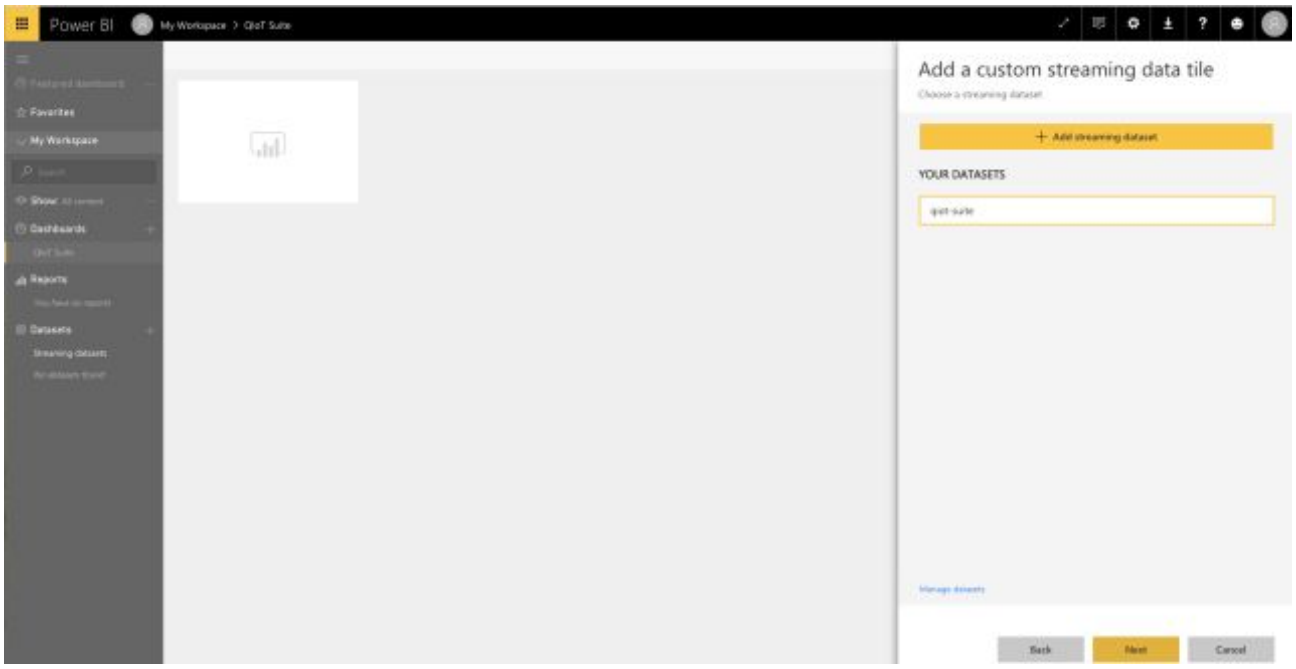
- Click “Add tile” in screen upper right corner



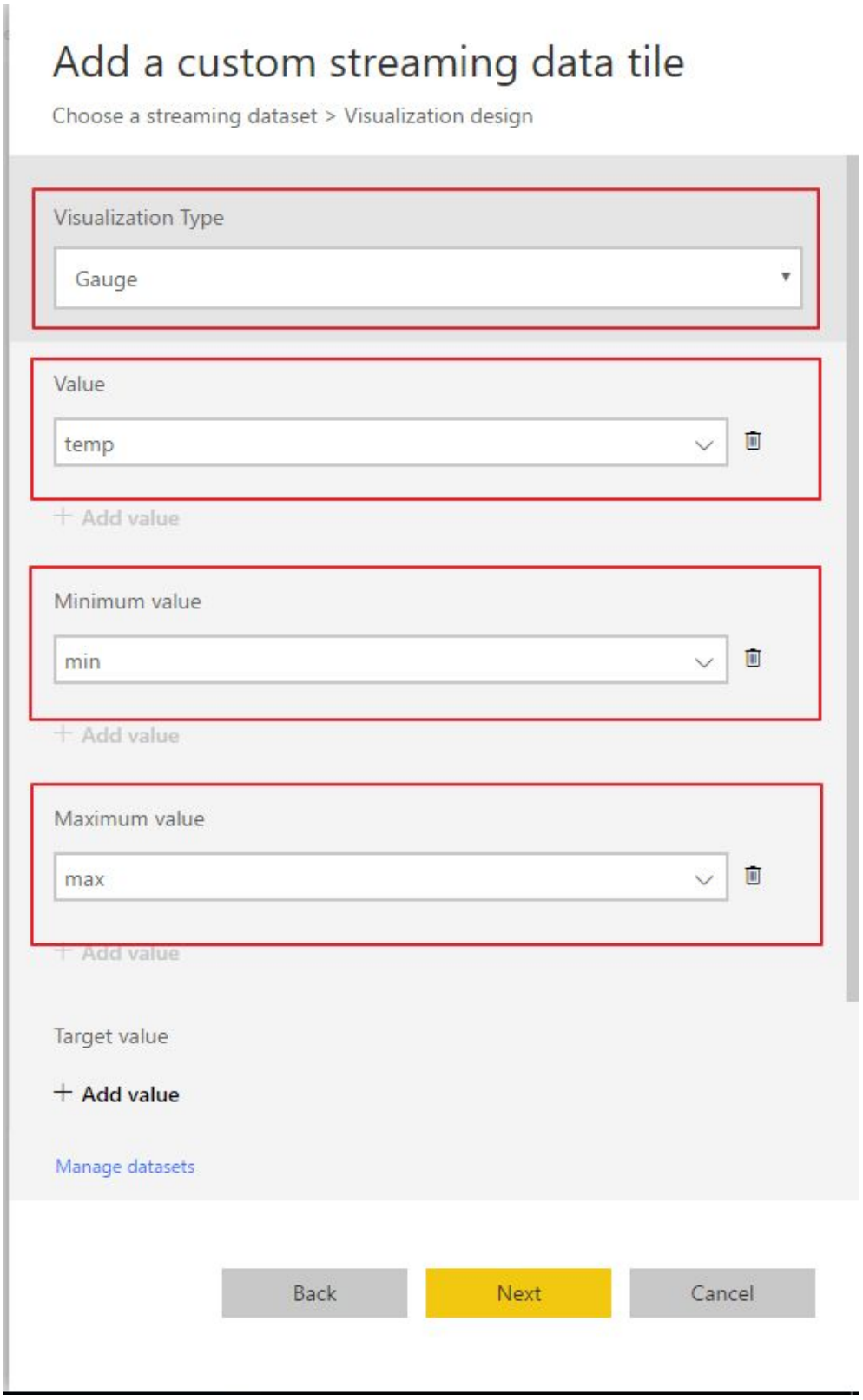
- Select “CUSTOM STREAMING DATA” and then select the “Next” button.



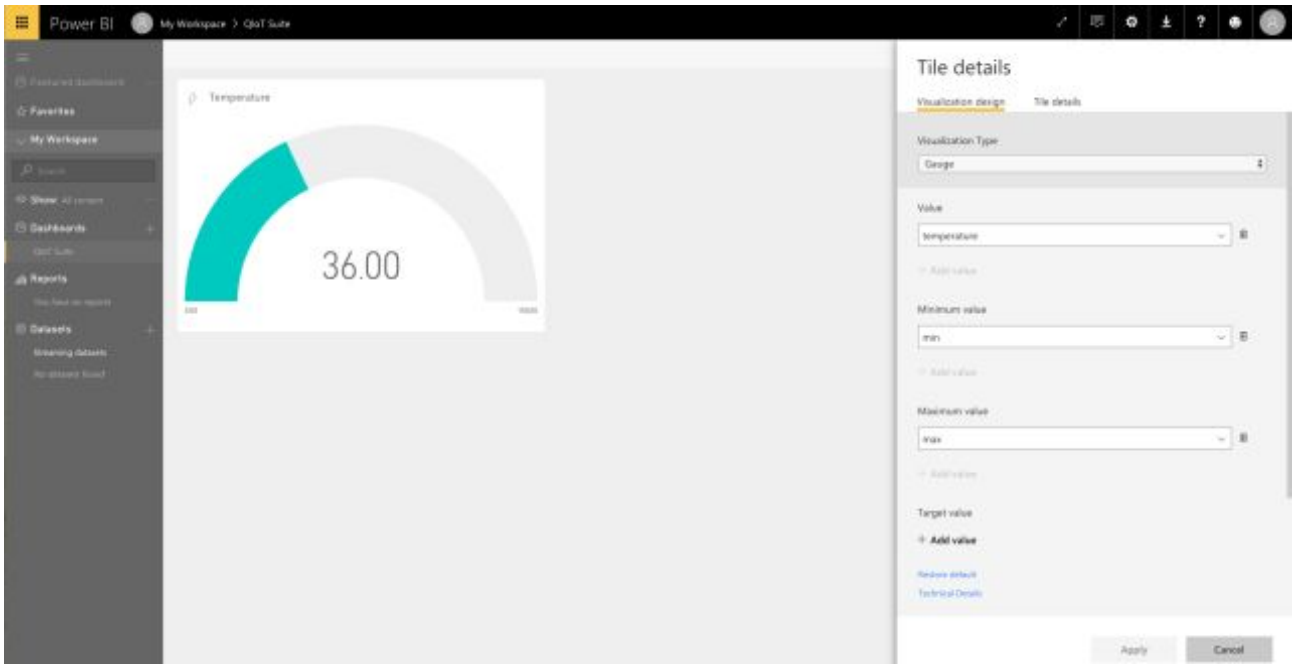
- Select datasets and then click the “Next” button.



- Select visualization type (e.g.,gauge),and set value,min,and max value.



- You have a streaming dataset to work with, you can get a real time gauge that looks like as following.


Power BI
Tile details
Visualization design
Tile details
Visualization Type
Gauge
Value
temperature
Minimum value
min
Maximum value
max
Target value
Add value
Apply
Cancel
Temperature
36.00

Appendix

QNAP QIoT Startkit Sample Code Introduction

- GitHub: Sample Code
- Sample Code Structure

```
qnap-qiot-sdks/
    nodejs/                         # nodejs program language
        device/
            intel-edison/           # intel-edison/raspberrypi...
                examples/
                    lib/            # QIoT command Lib
                    res/            # QIoT resourceinfo.json folder
                    ssl/            # QIoT certificate files folder.
                    mqtt.js         # sample code - mqtt/mqtts publish
                    http.js         # sample code - http post
                    https.js        # sample code - https post
                    coap.js         # sample code - coap postt
                    mqtt-subscribe.js   # sample code - mqtt/mqtts subscribe
                    http-get.js     # sample code - http get
                    https-get.js    # sample code - https get
                    coap-observe.js     # sample code - coap get
                    packagae.json       # npm packages document
    python/                         # python program language
        device/
            arduino-yun/
                examples/
                    lib/
                    res/
                    ssl/
                    mqtt.py
                    http.py
                    ...
```

- content of resourceinfo.json

protocol	resourceinfo.json content
mqtts	{ "host": ["172.17.28.28" # nas ip],

	<pre>"myqnapcloudHost": "Not Available", # myqnapcloudHost
	"port": 8883, # mqtts port
	"clientId": "adfa_1491561635", # thing Id
	"username": "00477f86-425b-49de-8590-xx", # username
	"password": "r:2825dedfb012969e1dfb6adb8", # password
	"resources": [# resource des
	{
	"resourcename": "adf", # resource name
	"resourceid": "dfadf", # resource id
	"resourcetypename": "Temperature", # resource type
	"datatype": "Float", # data type
	"unit": "°C", # data unit
	"description": "adfa", # resource des
	"topic": "qiot/things/admin/adfa/dfadf" # topic name
	}
	],
	"caCert": "/v1/media/ca-crt.pem", # certificate file
	"clientCert": "/v1/media/xx-04-07_10-40-35/xx_certificate.pem",
	"privateCert": "/v1/media/xx-04-07_10-40-35/xx_privatekey.pem"
	}</pre>

https	<pre>{
	"accesstoken": "r:2825dedfb012969e1dfb6adb8", # password
	"myqnapcloudHost": "Not Available",
	"clientId": "adfa_1491562164",
	"host": [
	"172.17.28.28"
	],
	"requesterid": "00477f86-425b-49de-8590-xx", # username
	"port": 3443, # https port
	...
	}</pre>

CoAP	<pre>{
	"myqnapcloudHost": "Not Available",
	"clientId": "adfa_1491562176",
	"host": [
	"172.17.28.28"
	],
	"r": "00477f86-425b-49de-8590-1282c65b4348", # username
	"t": "r:2825dedfb012969e1dfb6adb80a419df", # password
	"port": 5683, # coap port
	...
	}</pre>