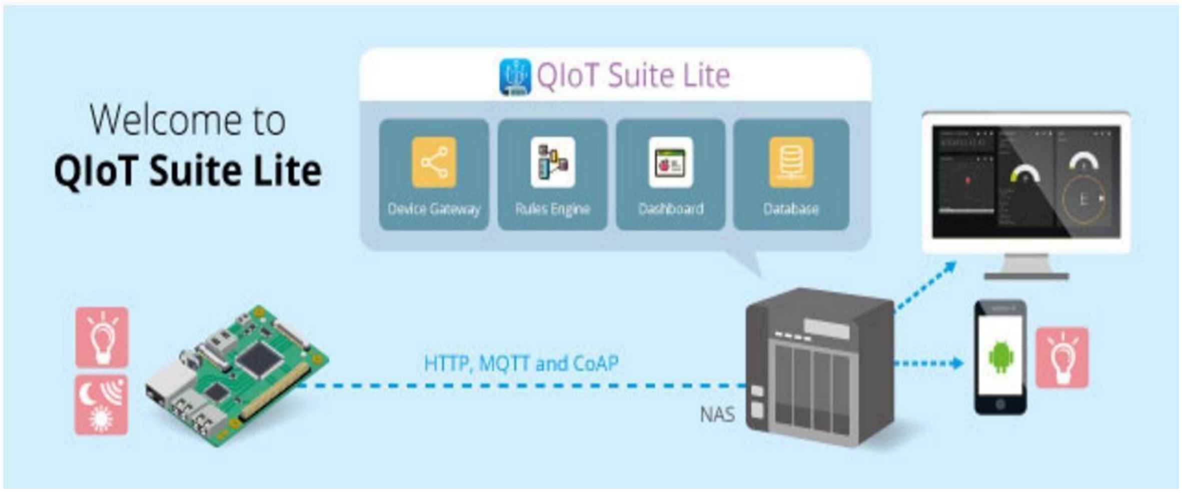
Get started with Intel Edison (Nodejs)

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

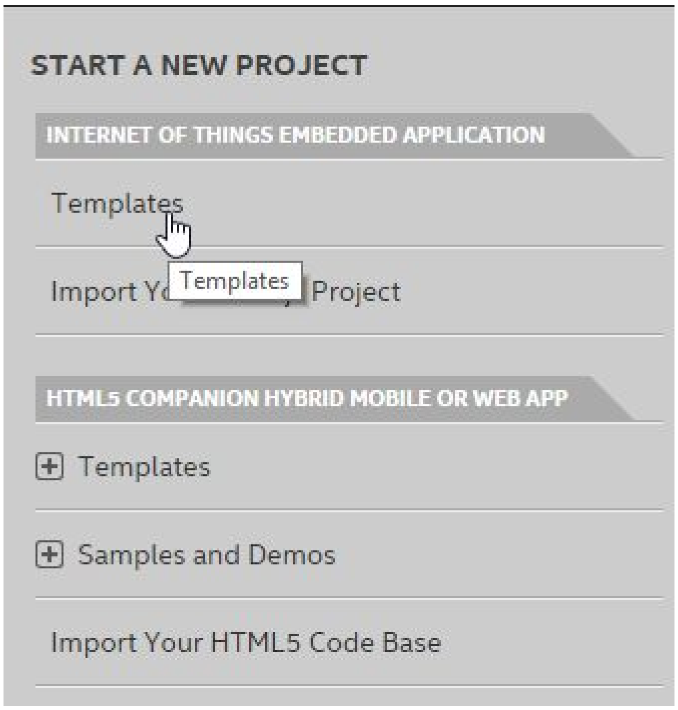
1.2 Configure and test your device

- 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

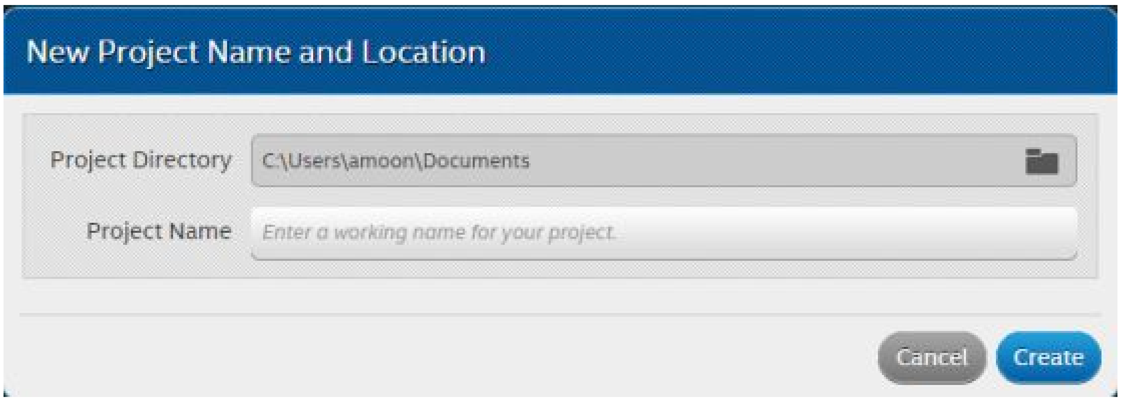
  1. Launch the Intel XDK.
  2. Follow the on-screen instructions to log in to your Intel® XDK account or sign up for a new Intel XDK account.
  3. From the Projects tab, click **Start a New Project** in the bottom left. The Start a New Project page opens.
  4. In the Internet of Things Embedded Application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


START A NEW PROJECT
INTERNET OF THINGS EMBEDDED APPLICATION
Templates
Import Your Project
Templates
HTML5 COMPANION HYBRID MOBILE OR WEB APP
Templates
Samples and Demos
Import Your HTML5 Code Base

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.



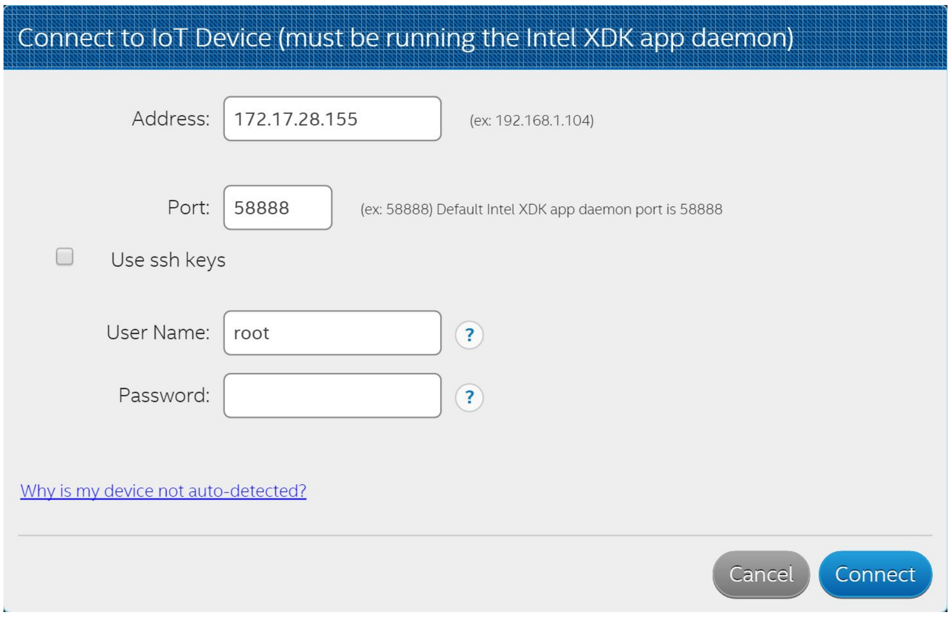
Project Directory: C:\Users\amoon\Documents

Project Name: Enter a working 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.


Connect to IoT Device (must be running the Intel XDK app daemon)

Address: 172.17.28.155 (ex: 192.168.1.104)

Port: 58888 (ex: 58888) Default Intel XDK app daemon port is 58888

☐ Use ssh keys

User Name: root

Password:

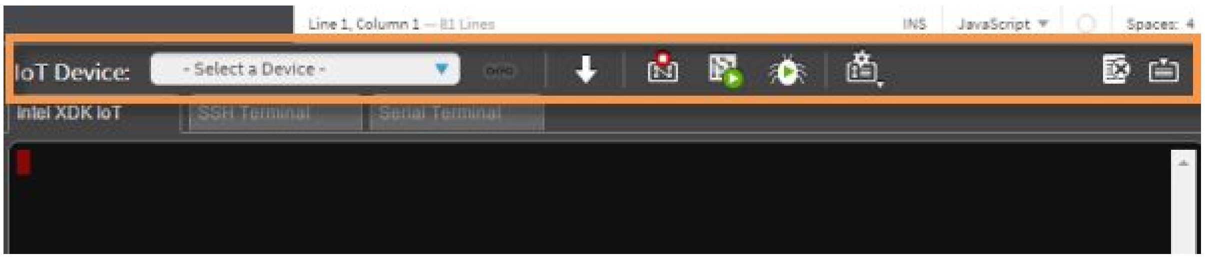
Why is my device not auto-detected?

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

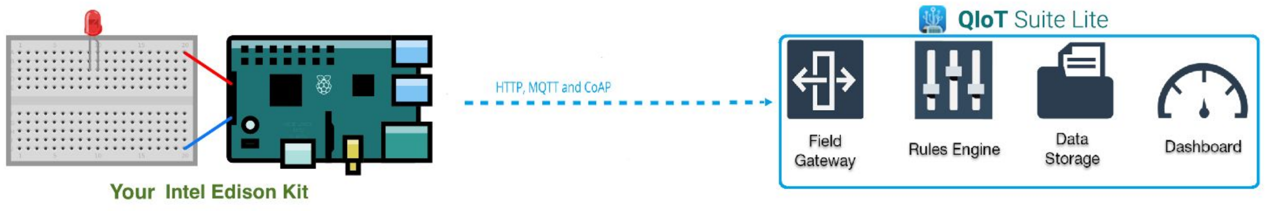
1.3 Install Nodejs and required libraries

- 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
```

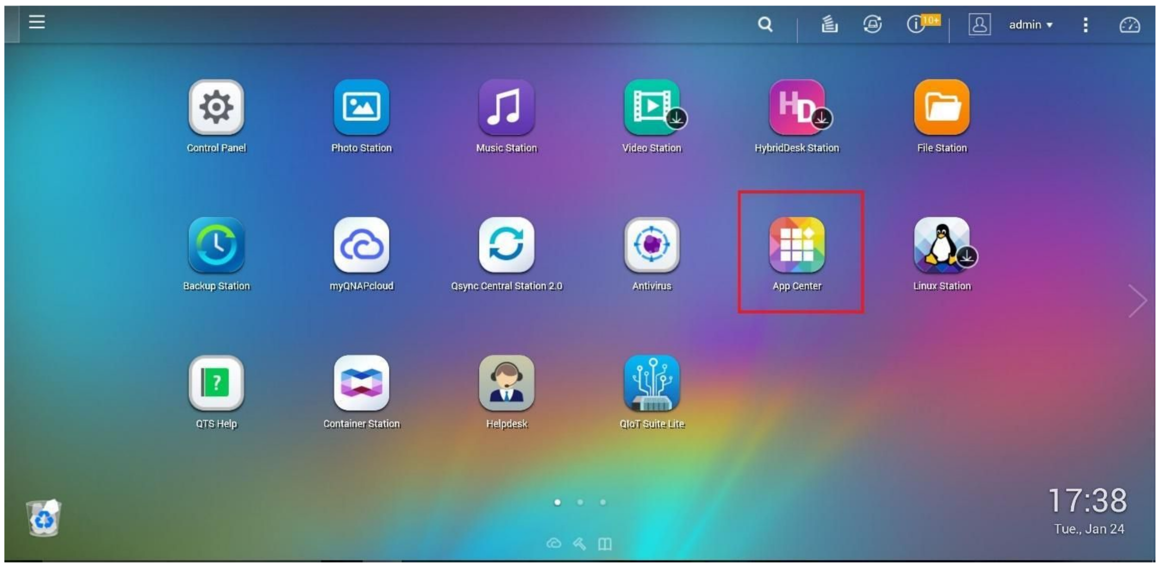
Lesson 2: Create your device in QIoT Suite Lite

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

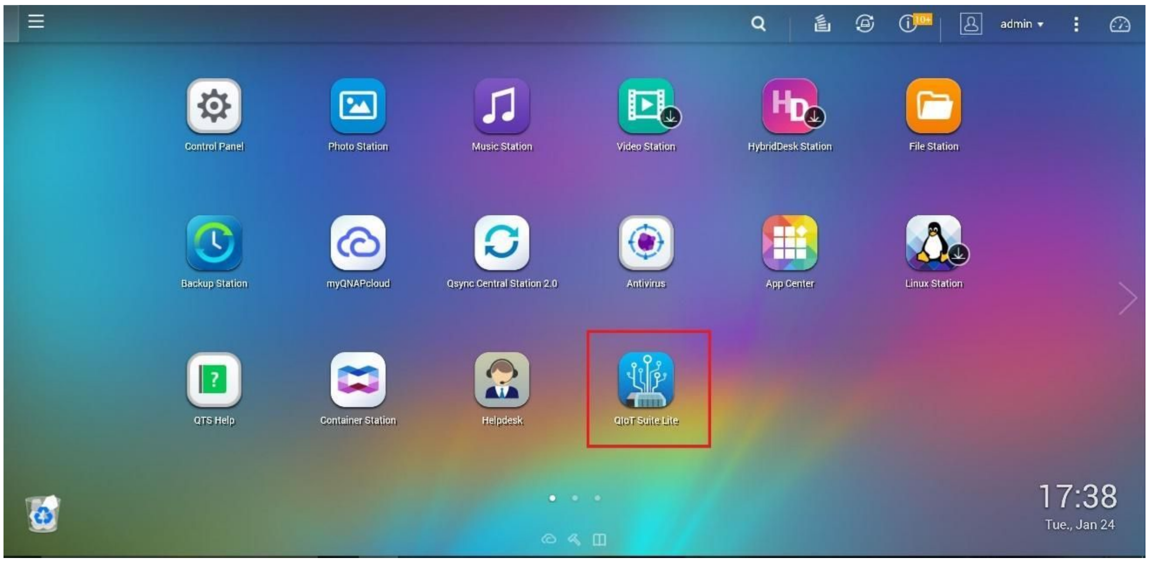


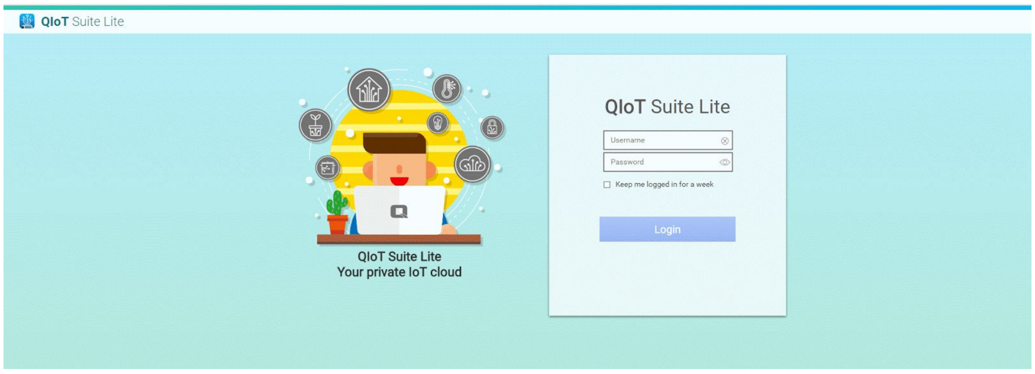
2.1 Install QIoT Suite Lite

- 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.





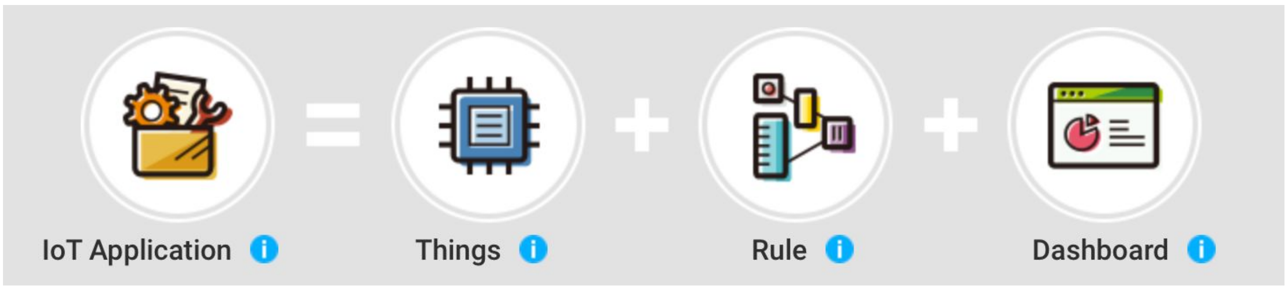
2.2 Create a new IoT application

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.

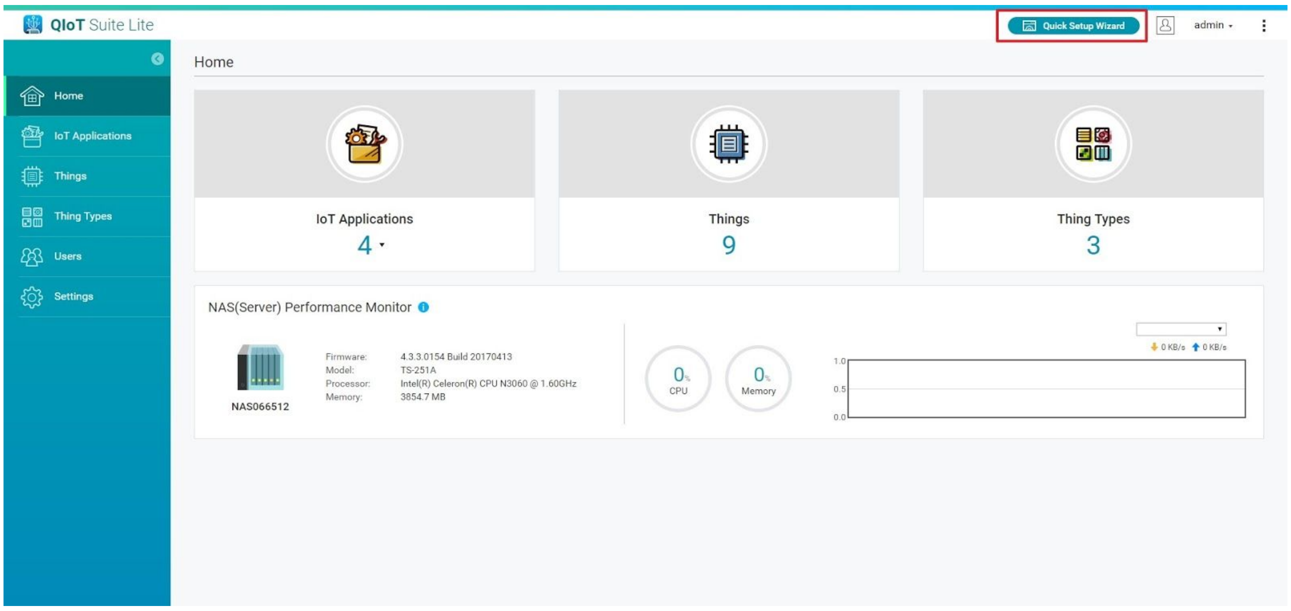
The Rule makes it possible to build IoT applications that gather, process, analyze and act on data generated by connected devices based on business rules you define. A rule can apply to data from one or many devices, and it can take one or many actions in parallel.

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

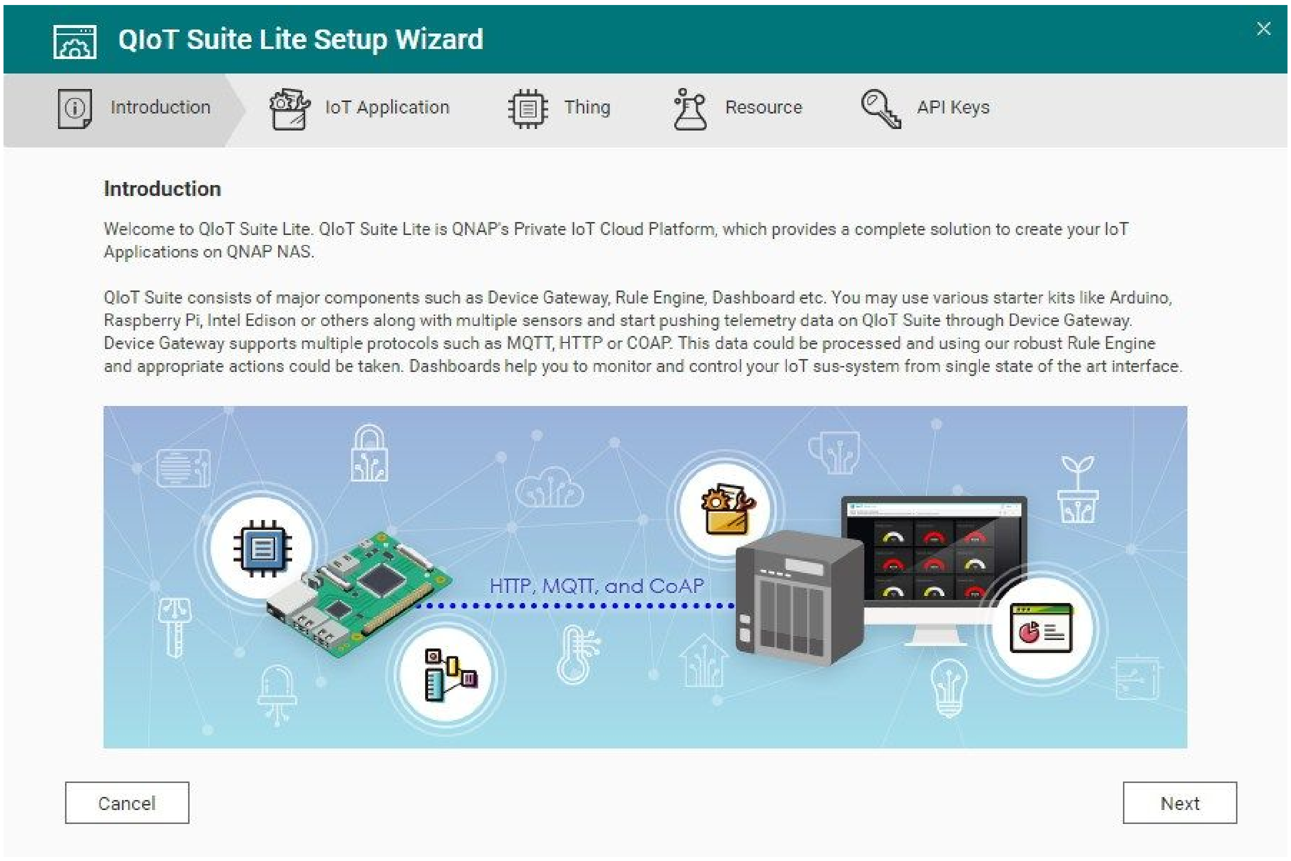
All these elements provide user a complete IoT Application environment.



- 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.



Firmware: 4.3.3.0154 Build 20170413
Model: TS-251A
Processor: Intel(R) Celeron(R) CPU N3060 @ 1.60GHz
Memory: 3854.7 MB

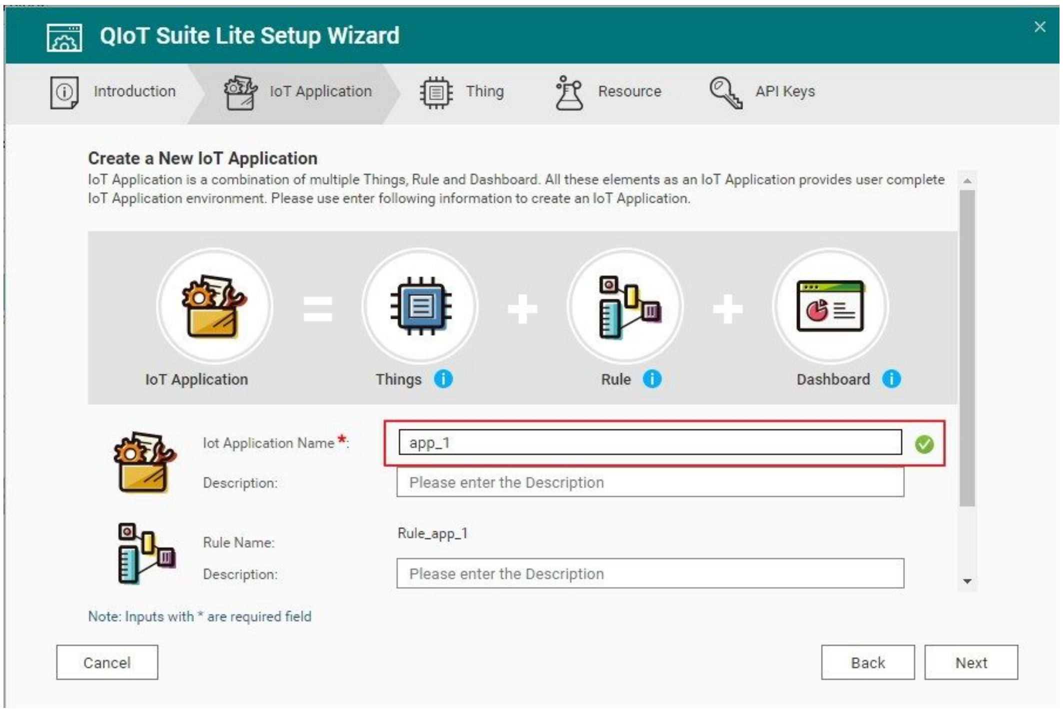


Introduction

Welcome to QIoT Suite Lite. QIoT Suite Lite is QNAP's Private IoT Cloud Platform, which provides a complete solution to create your IoT Applications on QNAP NAS.

QIoT Suite consists of major components such as Device Gateway, Rule Engine, Dashboard etc. You may use various starter kits like Arduino, Raspberry Pi, Intel Edison or others along with multiple sensors and start pushing telemetry data on QIoT Suite through Device Gateway. Device Gateway supports multiple protocols such as MQTT, HTTP or COAP. This data could be processed and using our robust Rule Engine and appropriate actions could be taken. Dashboards help you to monitor and control your IoT sus-system from single state of the art interface.

- 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


QIoT Suite Lite Setup Wizard

Introduction | IoT Application | Thing | Resource | API Keys

Create a New IoT Application

IoT Application is a combination of multiple Things, Rule and Dashboard. All these elements as an IoT Application provides user complete IoT Application environment. Please use enter following information to create an IoT Application.

Iot Application Name*: app_1

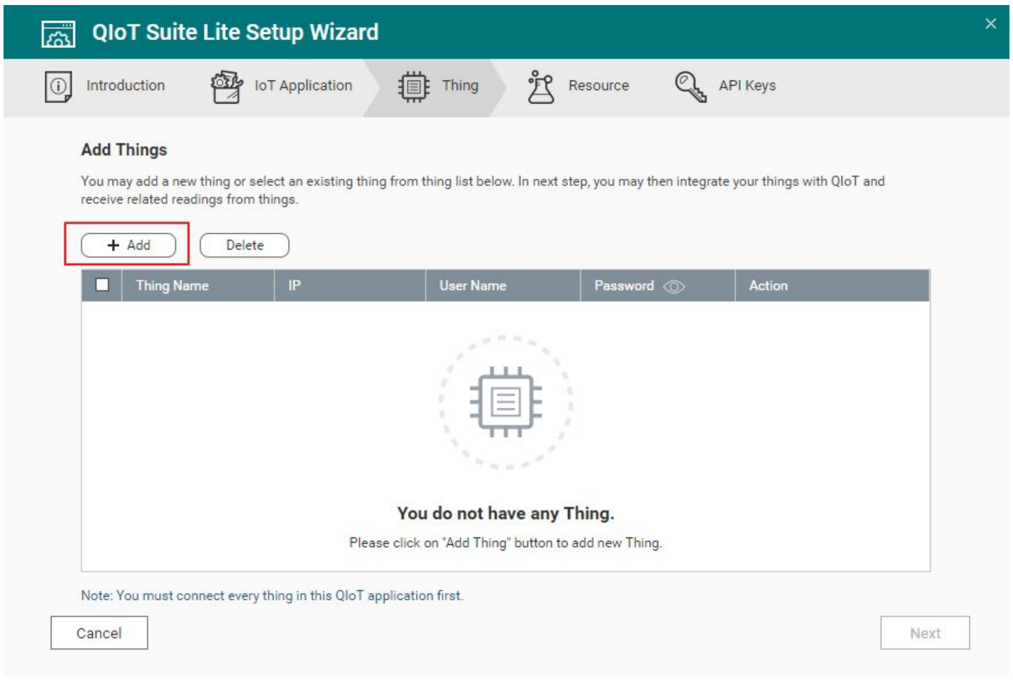
Description: Please enter the Description

Rule Name: Rule_app_1

Description: Please enter the Description

Note: Inputs with * are required field

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


QIoT Suite Lite Setup Wizard

Introduction | IoT Application | Thing | Resource | API Keys

Add Things

You may add a new thing or select an existing thing from thing list below. In next step, you may then integrate your things with QIoT and receive related readings from things.

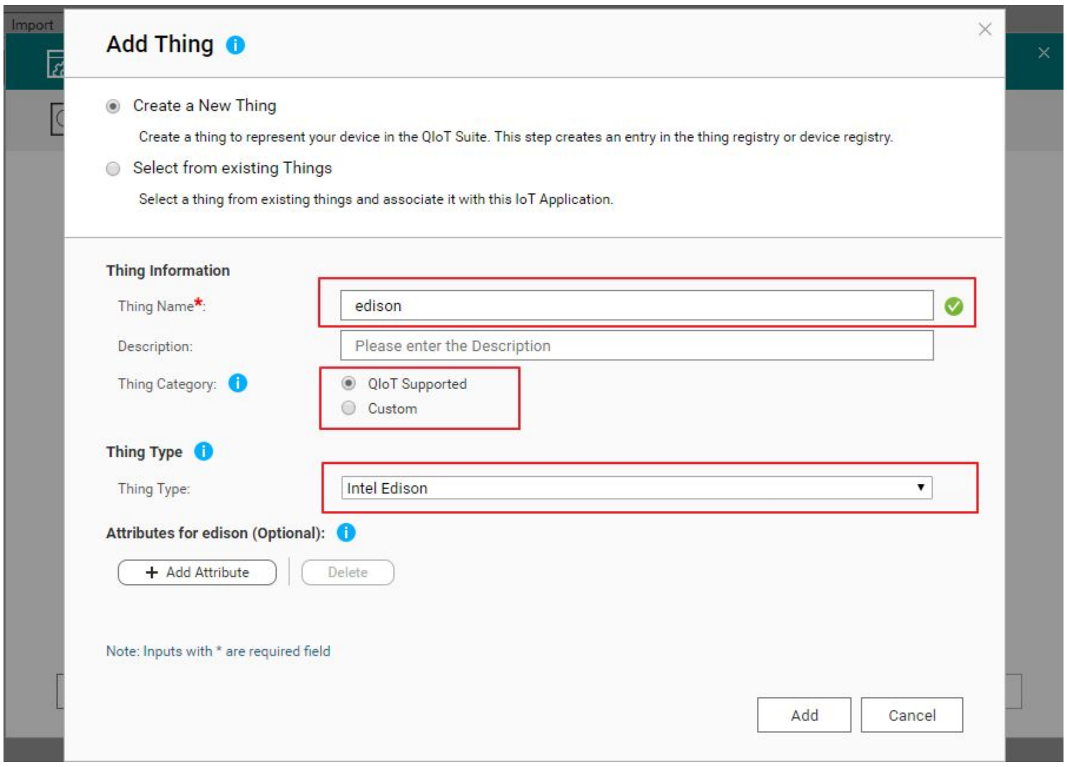
☐	Thing Name	IP	User Name	Password	Action

You do not have any Thing.

Please click on "Add Thing" button to add new Thing.

Note: You must connect every thing in this QIoT application first.

- 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.



Add Thing

(●) Create a New Thing

Create a thing to represent your device in the QIoT Suite. This step creates an entry in the thing registry or device registry.

( ) Select from existing Things

Select a thing from existing things and associate it with this IoT Application.

Thing Information

Thing Name*: edison

Description: Please enter the Description

Thing Category: [✓ QIoT Supported] Custom

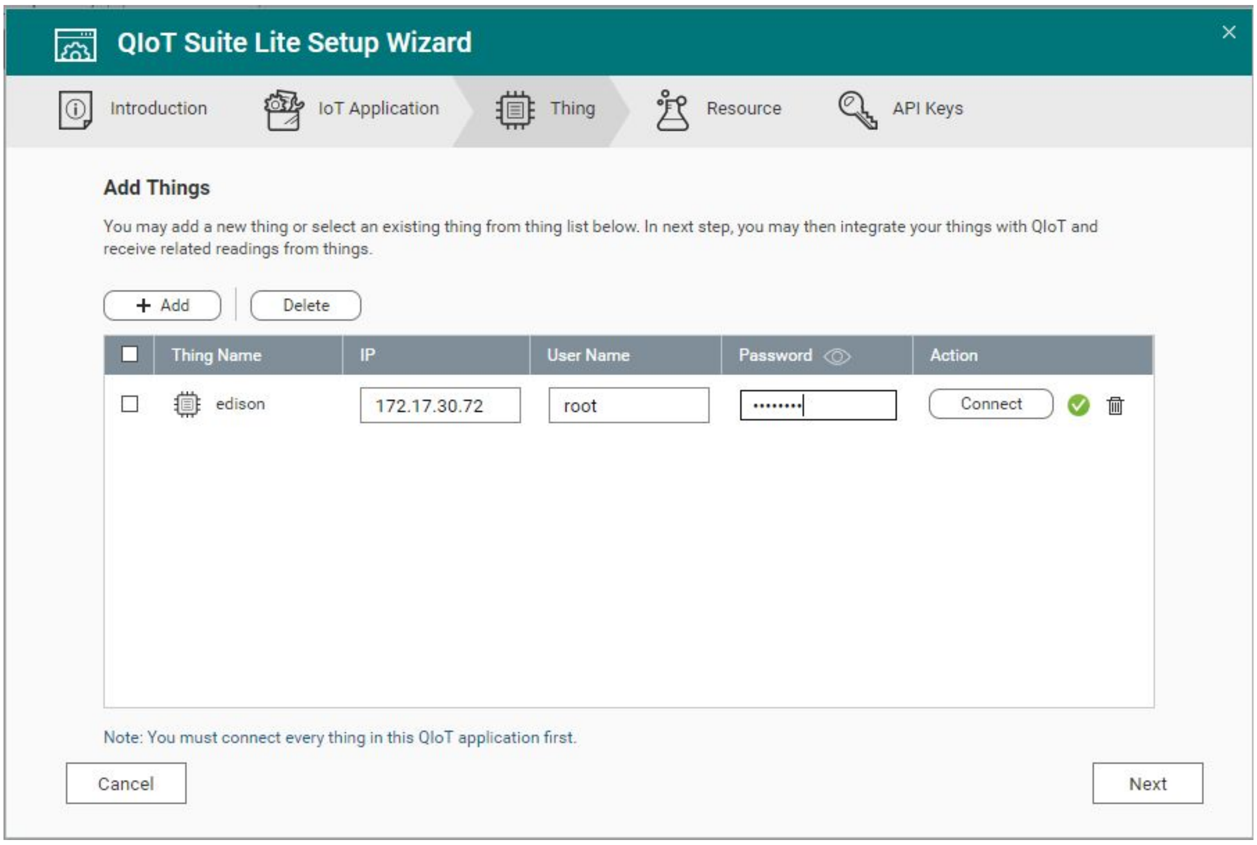
Thing Type

Thing Type: Intel Edison

Attributes for edison (Optional):

Note: Inputs with * are required field

- 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.



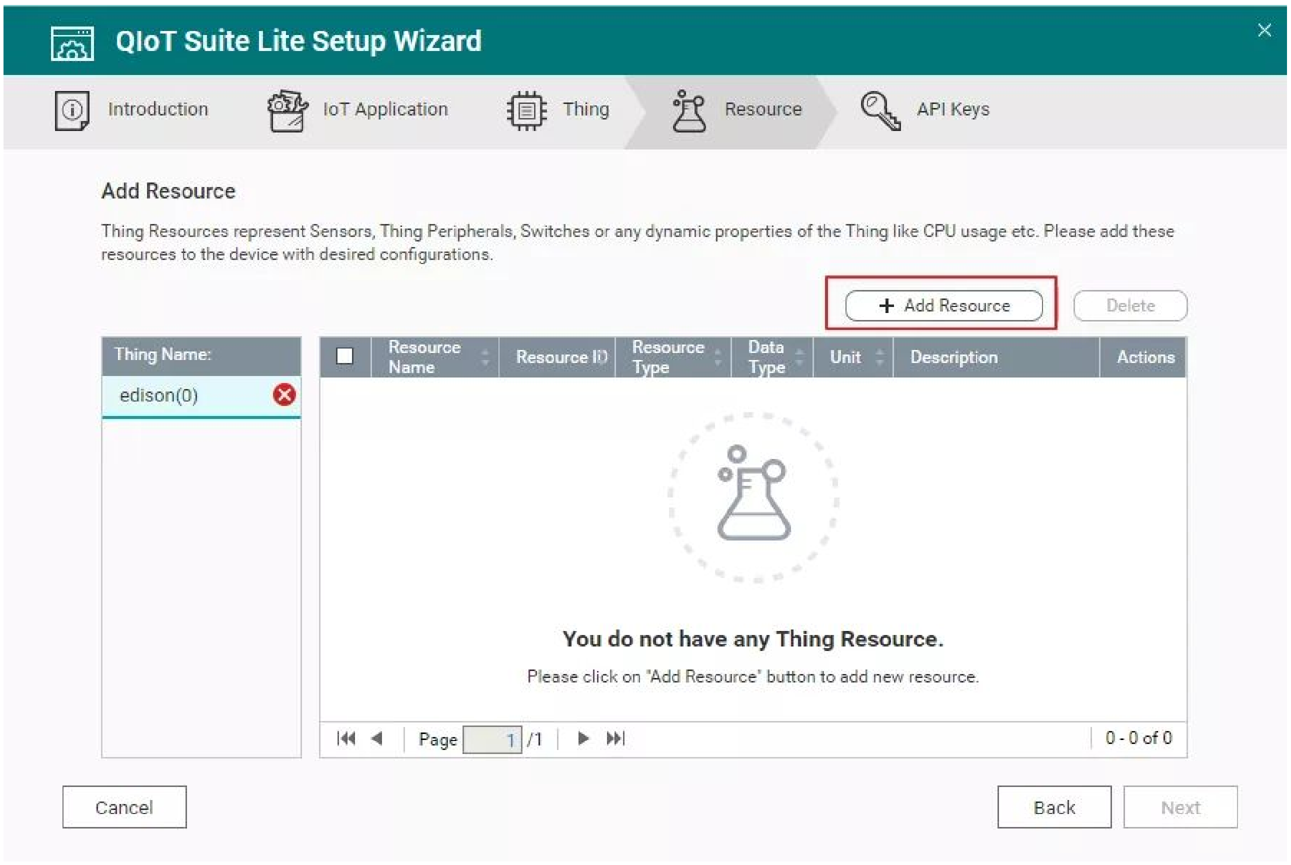
Add Things

You may add a new thing or select an existing thing from thing list below. In next step, you may then integrate your things with QIoT and receive related readings from things.

	Thing Name	IP	User Name	Password	Action
☐	edison	172.17.30.72	root	••••••••	Connect ✓

Note: You must connect every thing in this QIoT application first.

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



Add Resource

Thing Resources represent Sensors, Thing Peripherals, Switches or any dynamic properties of the Thing like CPU usage etc. Please add these resources to the device with desired configurations.

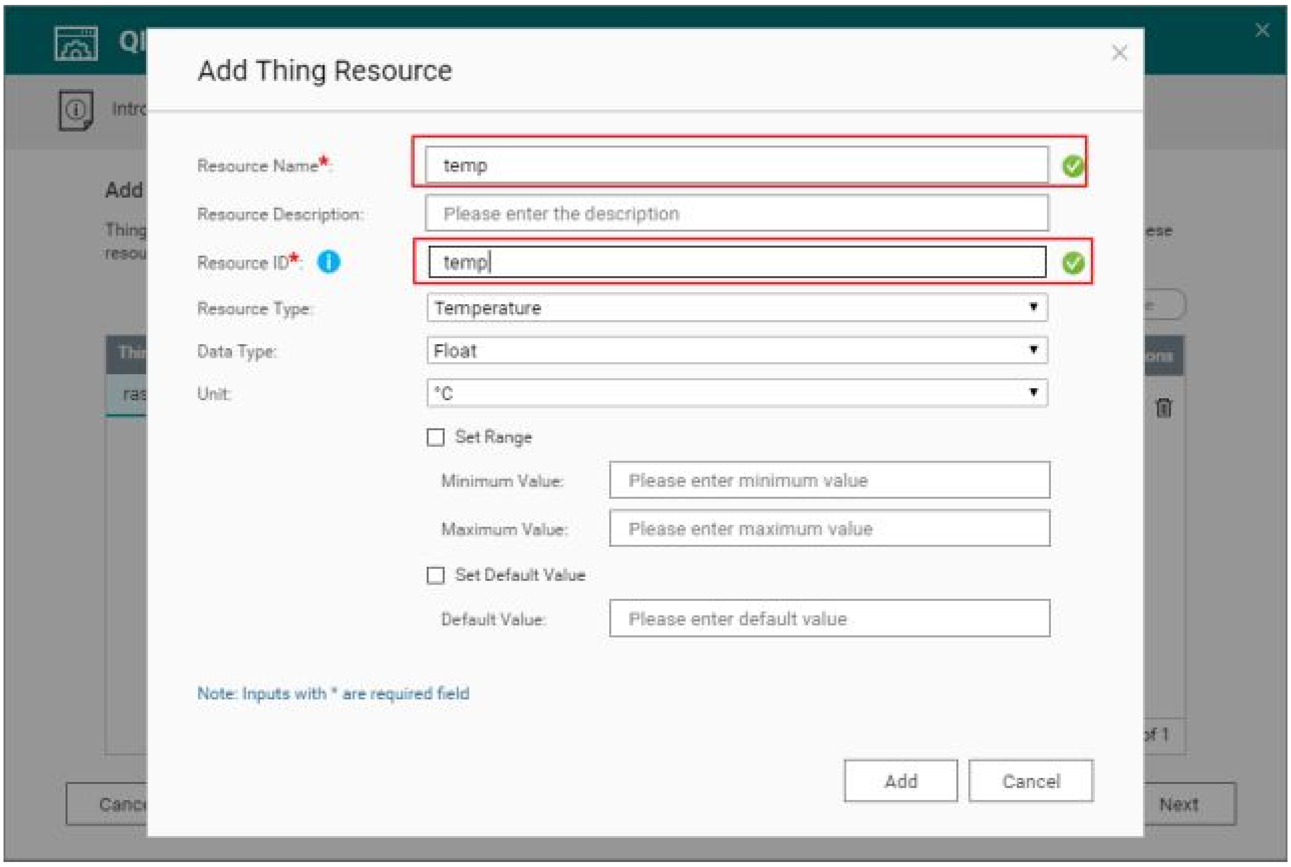
Thing Name:
edison(0)

☐	Resource Name	Resource ID	Resource Type	Data Type	Unit	Description	Actions

You do not have any Thing Resource.

Please click on "Add Resource" button to add new 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.

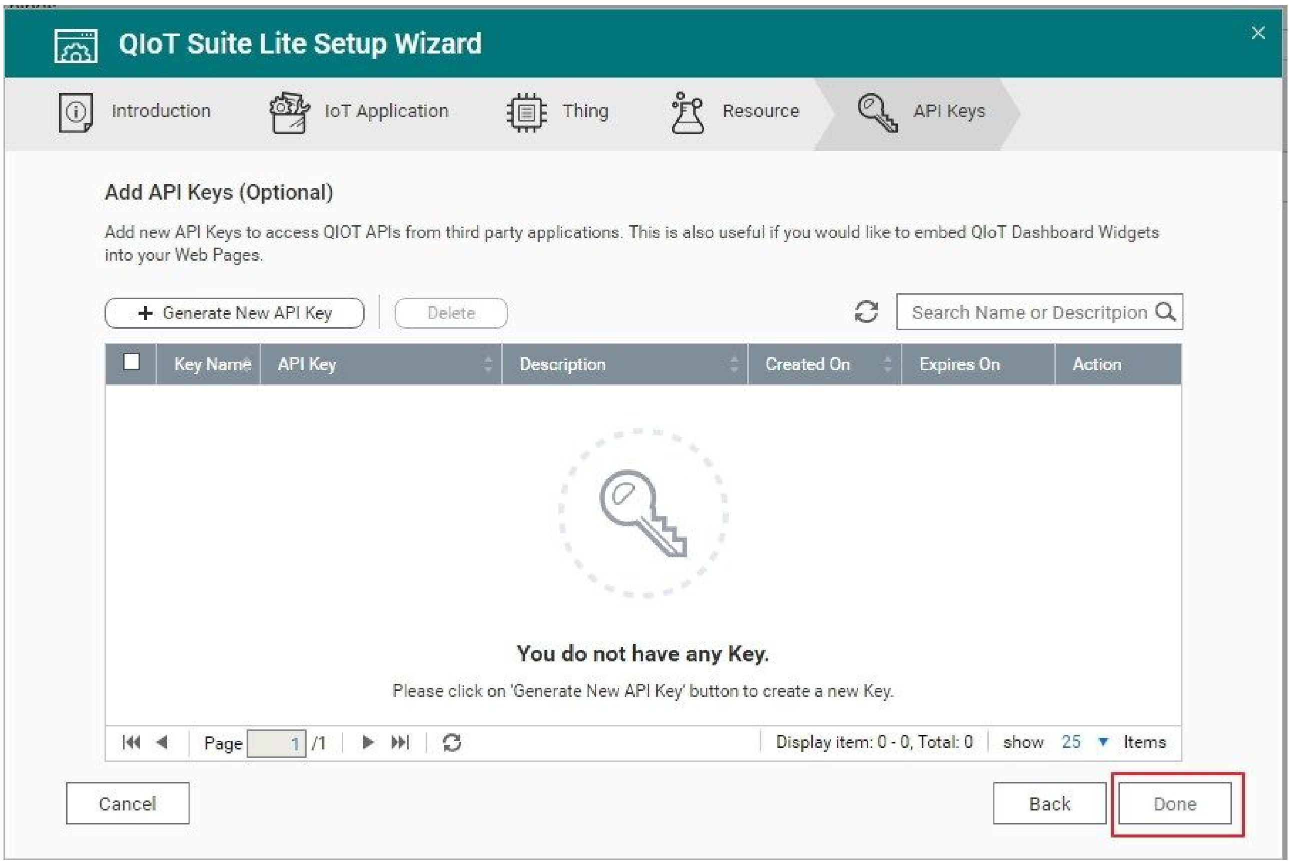

Add Thing Resource

Resource Name*: temp
Resource Description: Please enter the description
Resource ID*: temp
Resource Type: Temperature
Data Type: Float
Unit: °C
☐ Set Range
Minimum Value: Please enter minimum value
Maximum Value: Please enter maximum value
☐ Set Default Value
Default Value: Please enter default value

Note: Inputs with * are required field

- 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.



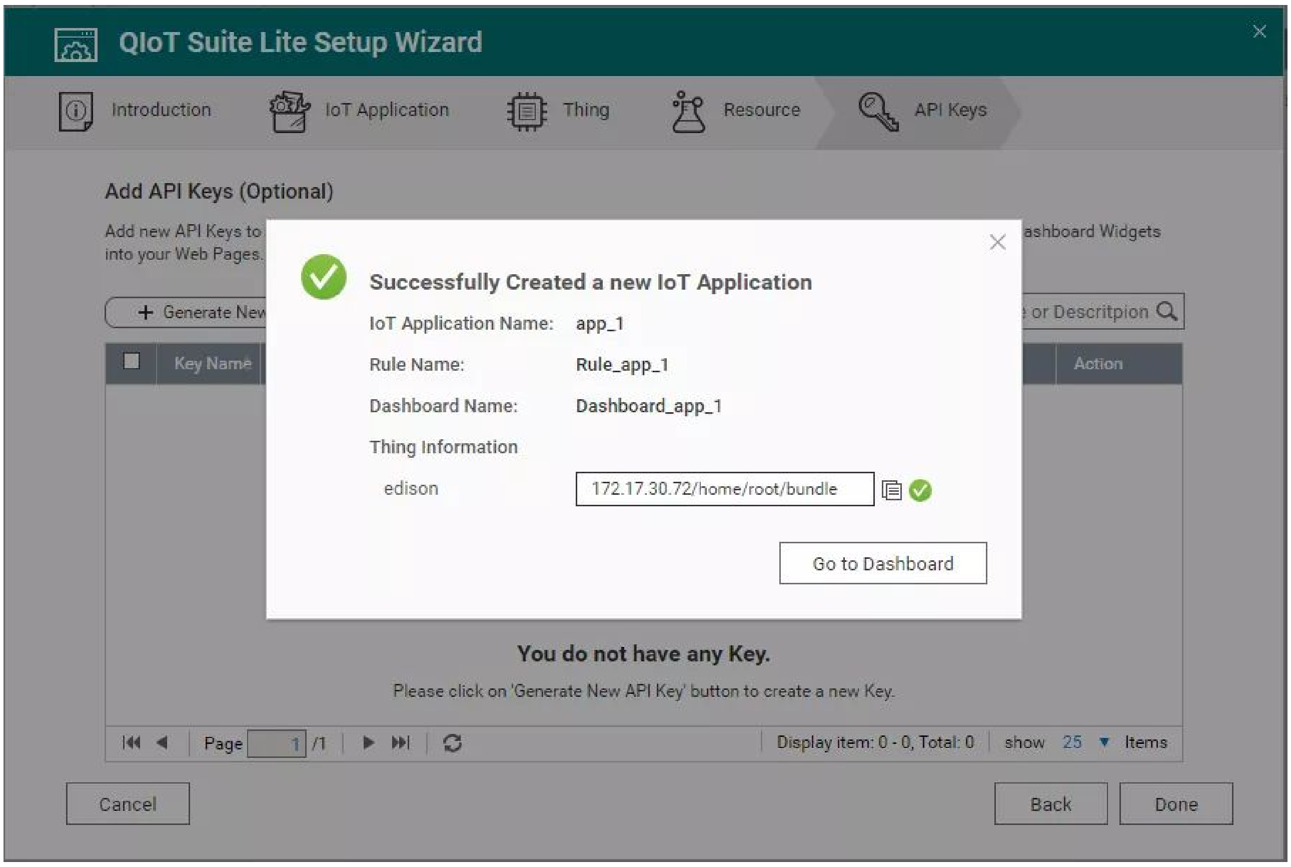
Add API Keys (Optional)

Add new API Keys to access QIOT APIs from third party applications. This is also useful if you would like to embed QIoT Dashboard Widgets into your Web Pages.

You do not have any Key.

Please click on 'Generate New API Key' button to create a new Key.

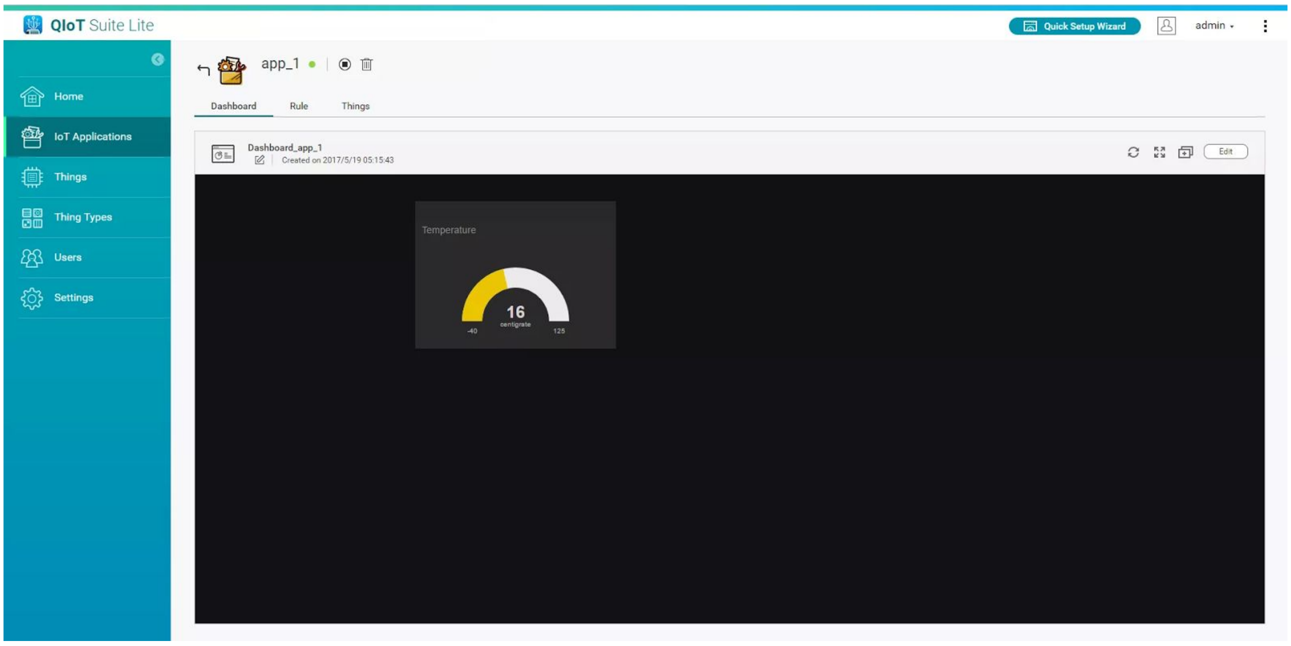
- For a while, sample codes and related files (certificate, resource information) have been upload to the specified path on your device.



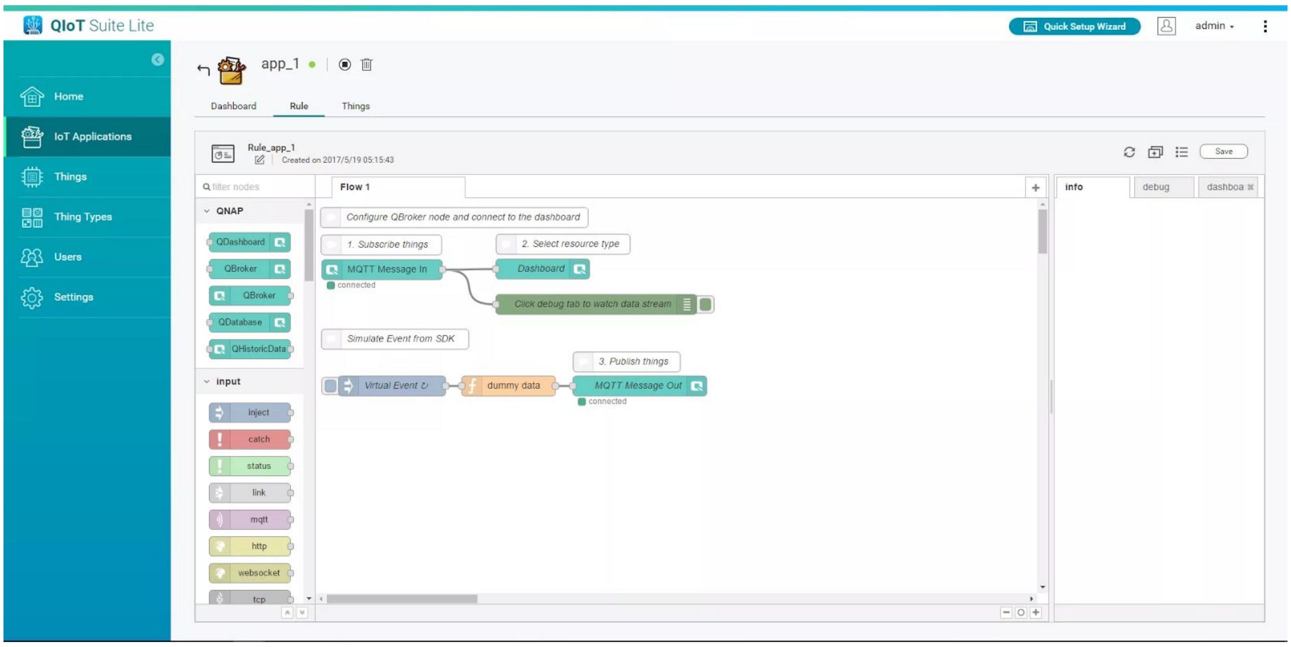
Successfully Created a new IoT Application

IoT Application Name:	app_1
Rule Name:	Rule_app_1
Dashboard Name:	Dashboard_app_1
Thing Information	
edison	172.17.30.72/home/root/bundle

- 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.

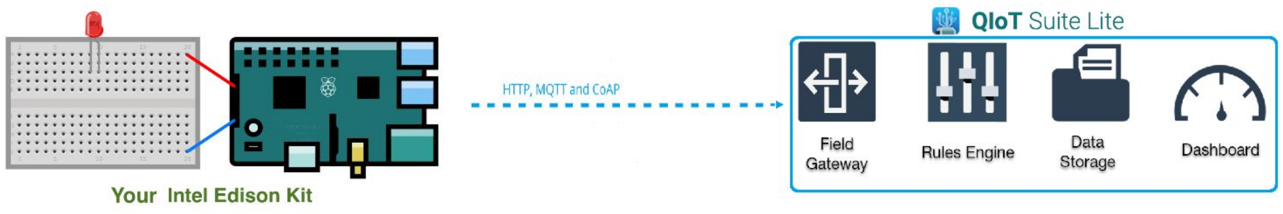


- Select "Rule" tab, you could to define the flow or rule about how to process the data sent from the device, and how to present in dashboard.



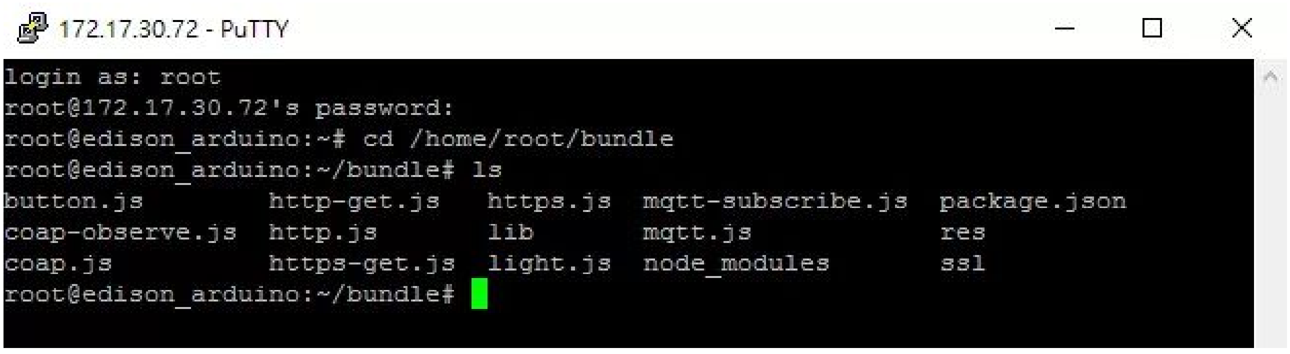
Lesson 3: Connect your device to QIoT Suite Lite using MQTTS

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

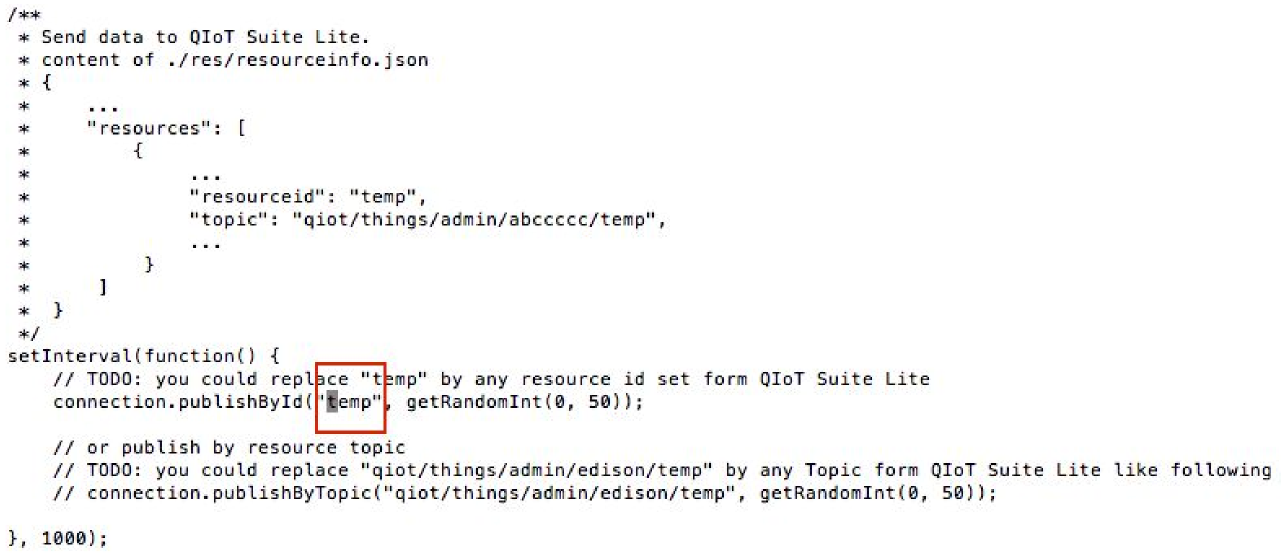
- Open Terminal application (e.g., PuTTY) on your PC.Connect to your device by SSH and enter the folder where put sample code (e.g., /home/root/bundle).



```
login as: root
root@172.17.30.72's password:
root@edison_arduino:~# cd /home/root/bundle
root@edison_arduino:~/bundle# ls
button.js        http-get.js   https.js  mqtt-subscribe.js  package.json
coap-observe.js  http.js       lib       mqtt.js            res
coap.js          https-get.js  light.js  node_modules       ssl
root@edison_arduino:~/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.

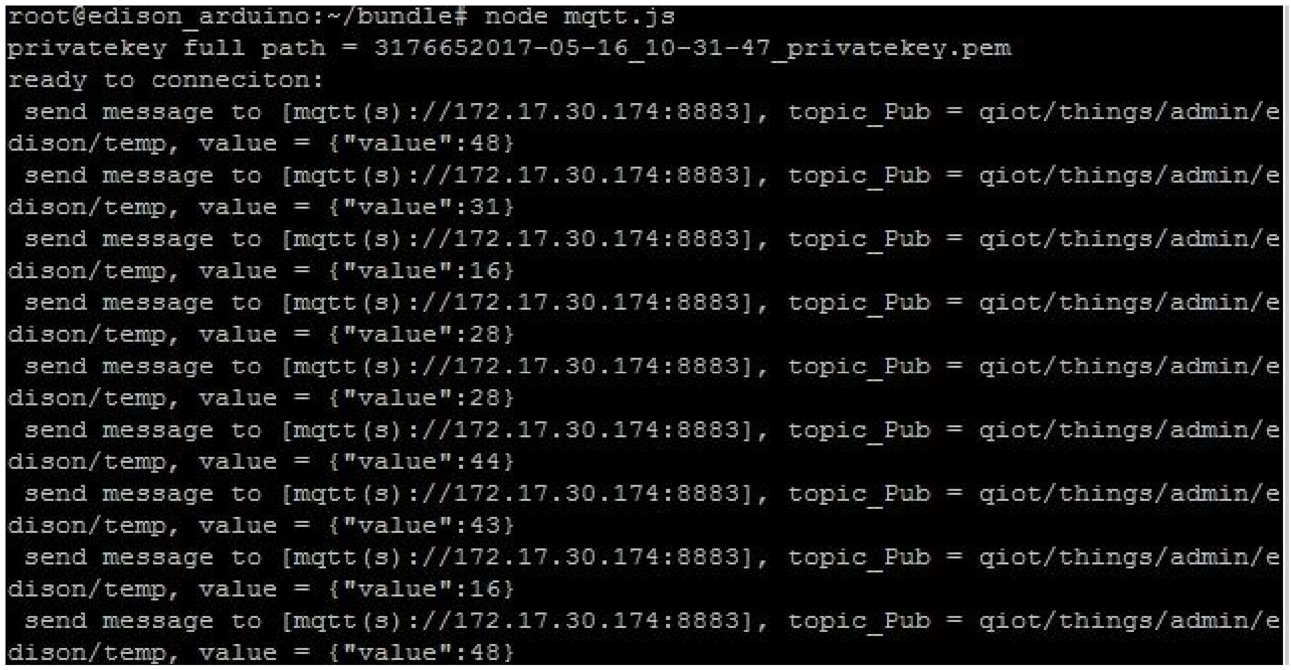


```
/**
 * Send data to QIoT Suite Lite.
 * content of ./res/resourceinfo.json
 * {
 *     ...
 *     "resources": [
 *         {
 *             ...
 *             "resourceid": "temp",
 *             "topic": "qiot/things/admin/abccccc/temp",
 *             ...
 *         }
 *     ]
 *  }
 */
setInterval(function() {
    // 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

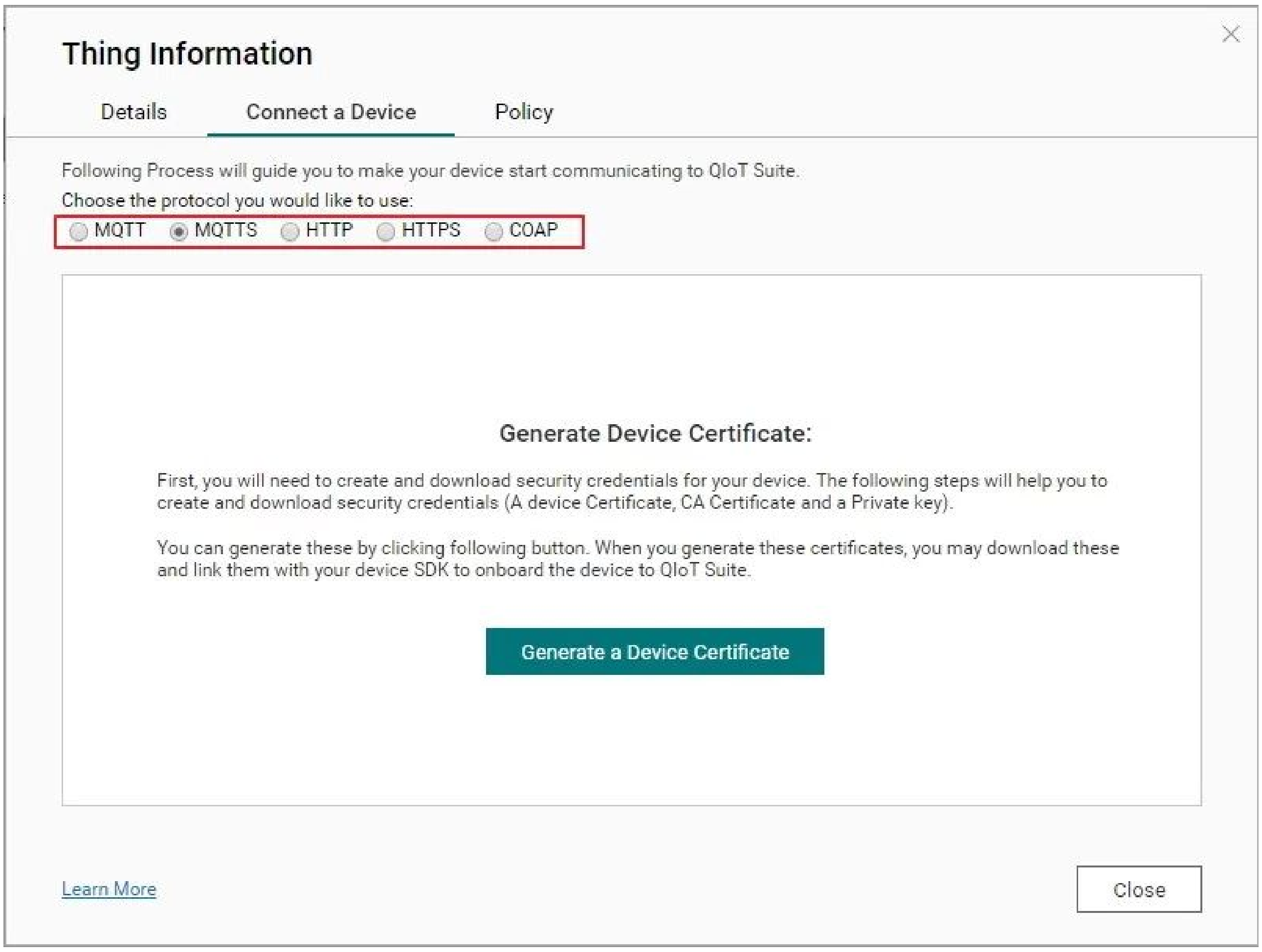
- **device will send message to topic “temp” or that you defined ,as below image.**



```
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


Thing Information

Details | Connect a Device | Policy

Following Process will guide you to make your device start communicating to QIoT Suite.

Choose the protocol you would like to use:

MQTT [✓ MQTTS] HTTP HTTPS COAP

Generate Device Certificate:

First, you will need to create and download security credentials for your device. The following steps will help you to create and download security credentials (A device Certificate, CA Certificate and a Private key).

You can generate these by clicking following button. When you generate these certificates, you may download these and link them with your device SDK to onboard the device to QIoT Suite.

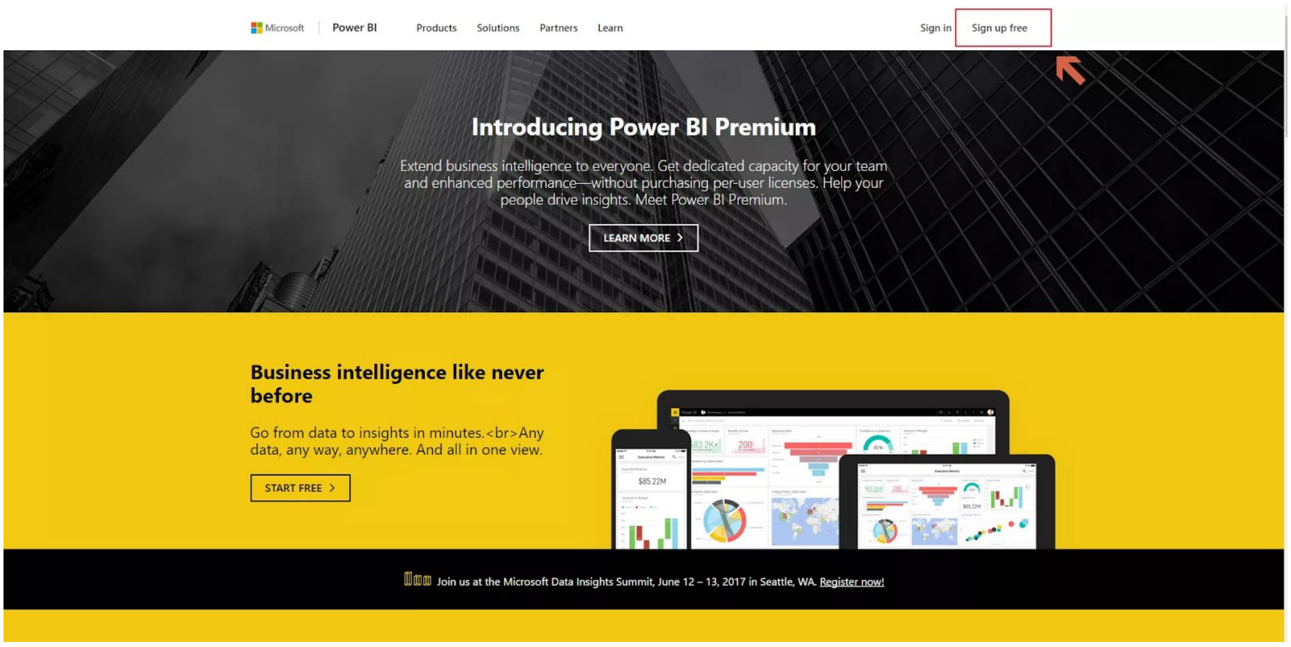
- SSH to your device , and input command as following.

```
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.



Introducing Power BI Premium

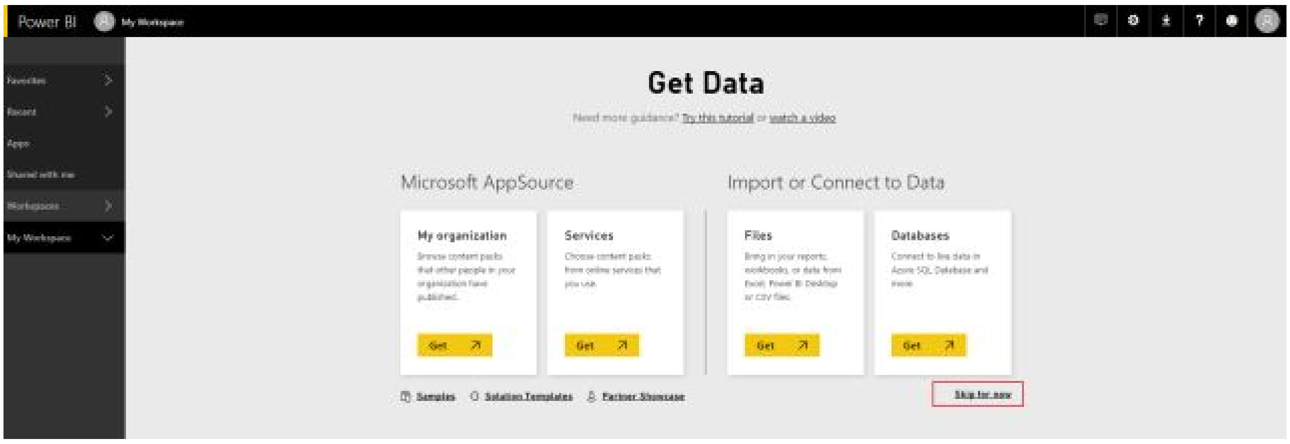
Extend business intelligence to everyone. Get dedicated capacity for your team and enhanced performance—without purchasing per-user licenses. Help your people drive insights. Meet Power BI Premium.

Business intelligence like never before

Go from data to insights in minutes.
Any data, any way, anywhere. And all in one view.

Join us at the Microsoft Data Insights Summit, June 12 – 13, 2017 in Seattle, WA. Register now!

- 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.



Browse content packs that other people in your organization have published.

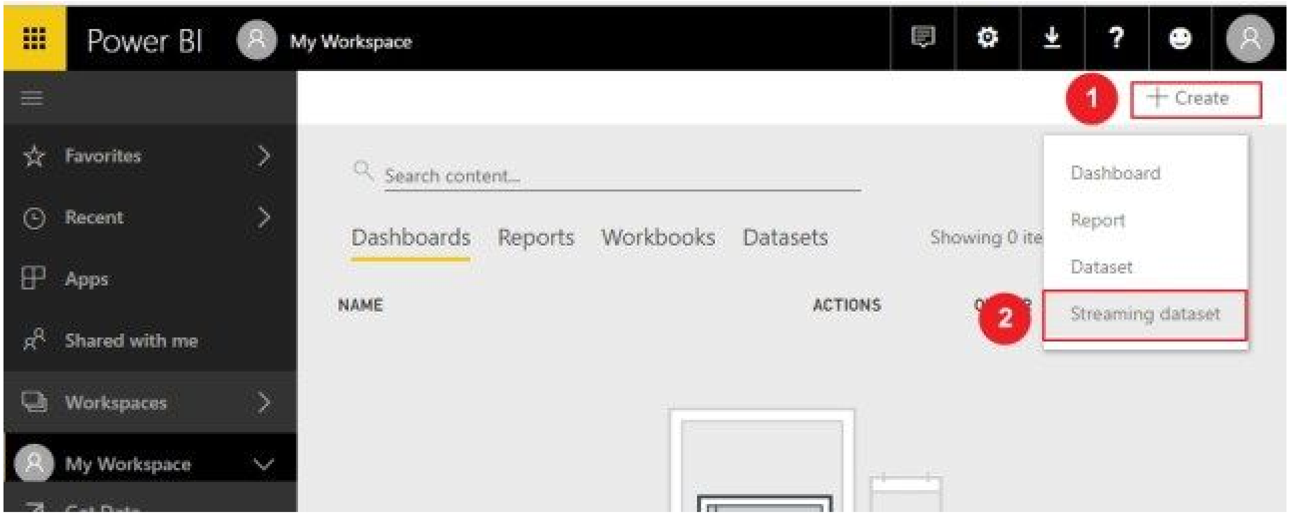
Choose content packs from online services that you use.

Bring in your reports, workbooks, or data from Excel, Power BI Desktop or CSV files.

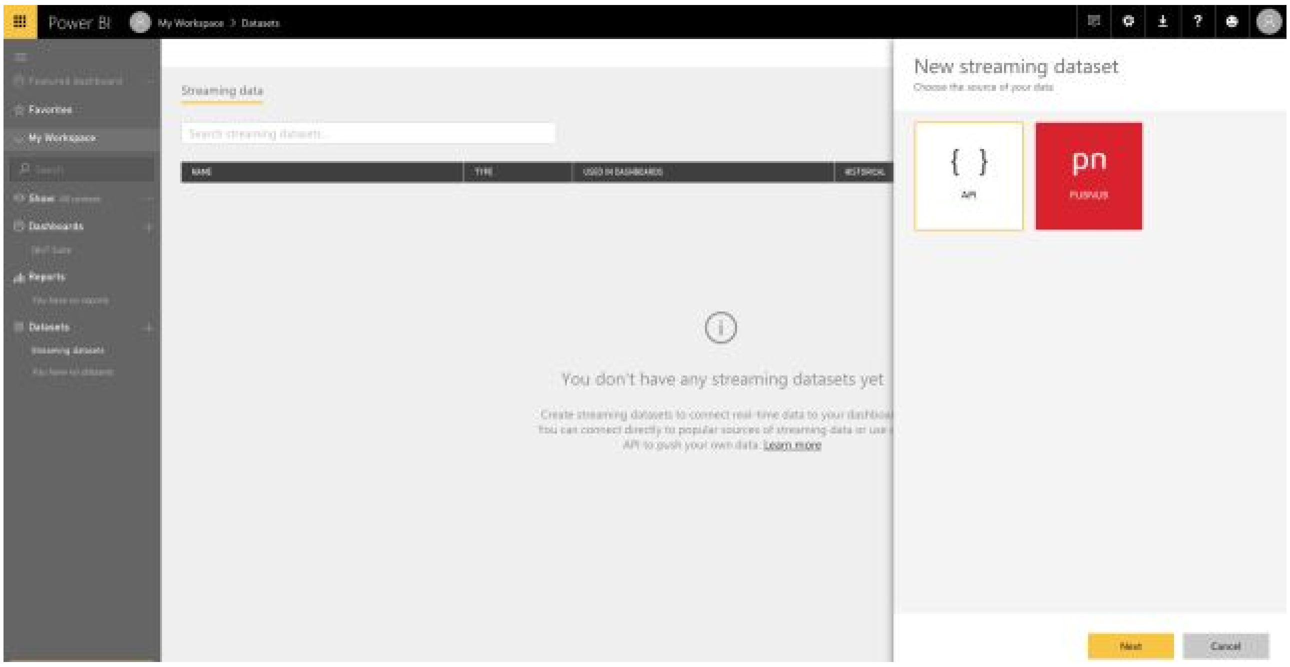
Connect to live data in Azure SQL Database and more.

4.2 Setup your streaming dataset API

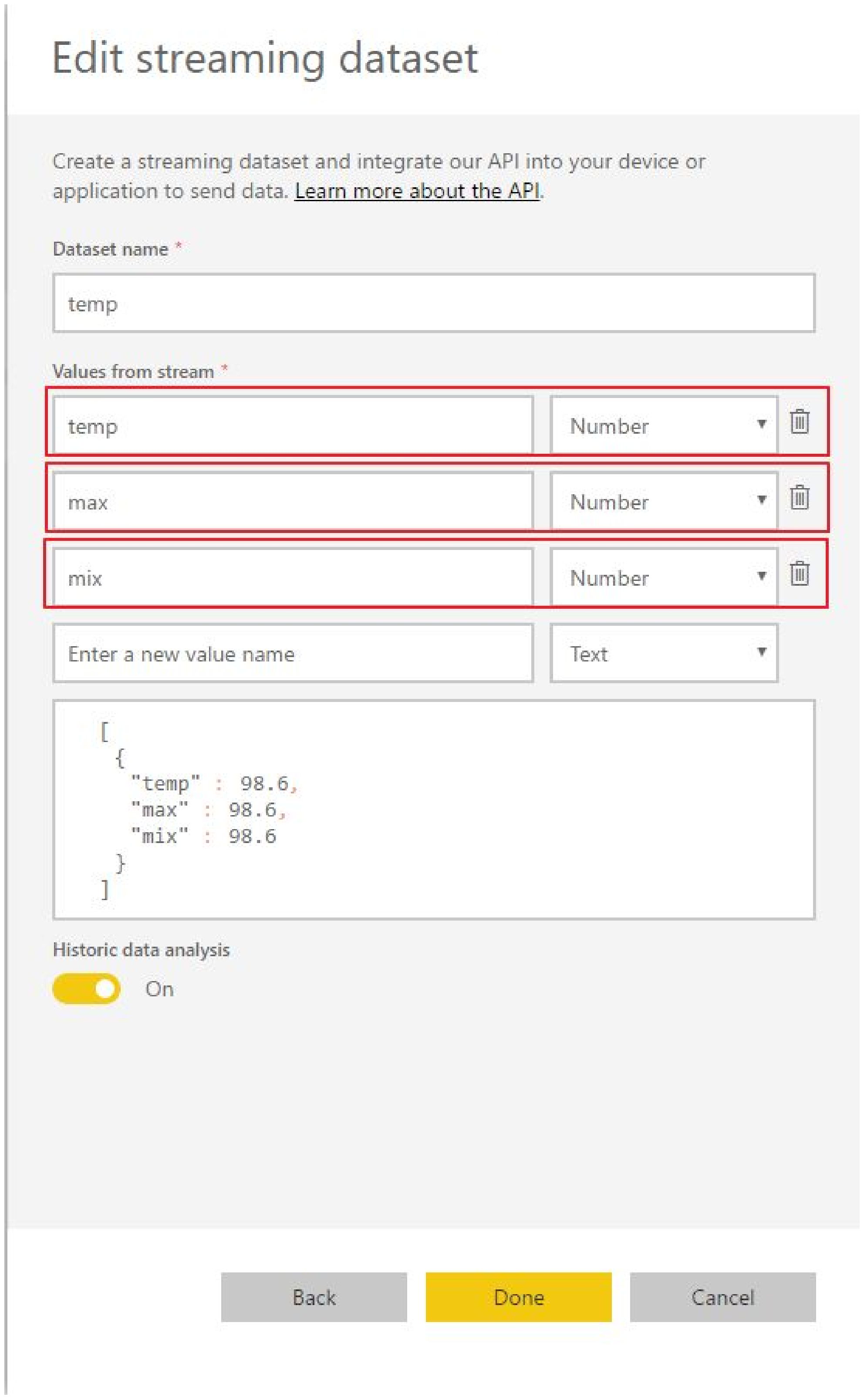
- Create “Datasets”
    - Click “Create” in scrren upper right corner
    - And then click “Streming dataset”



- 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.



Edit streaming dataset

Create a streaming dataset and integrate our API into your device or application to send data. Learn more about the API.

Dataset name *

temp

Values from stream *

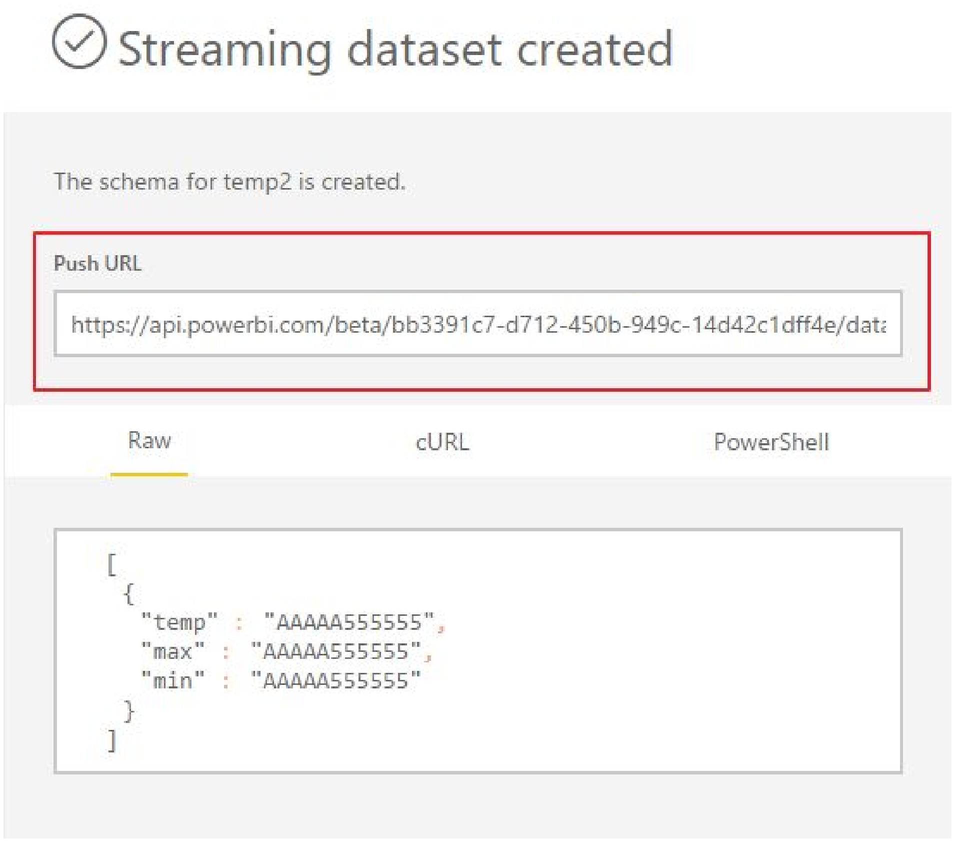
temp	Number
max	Number
mix	Number
Enter a new value name	Text

```
[
  {
    "temp" : 98.6,
    "max" : 98.6,
    "mix" : 98.6
  }
]
```

Historic data analysis

On

- 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.


Streaming dataset created

The schema for temp2 is created.

Push URL

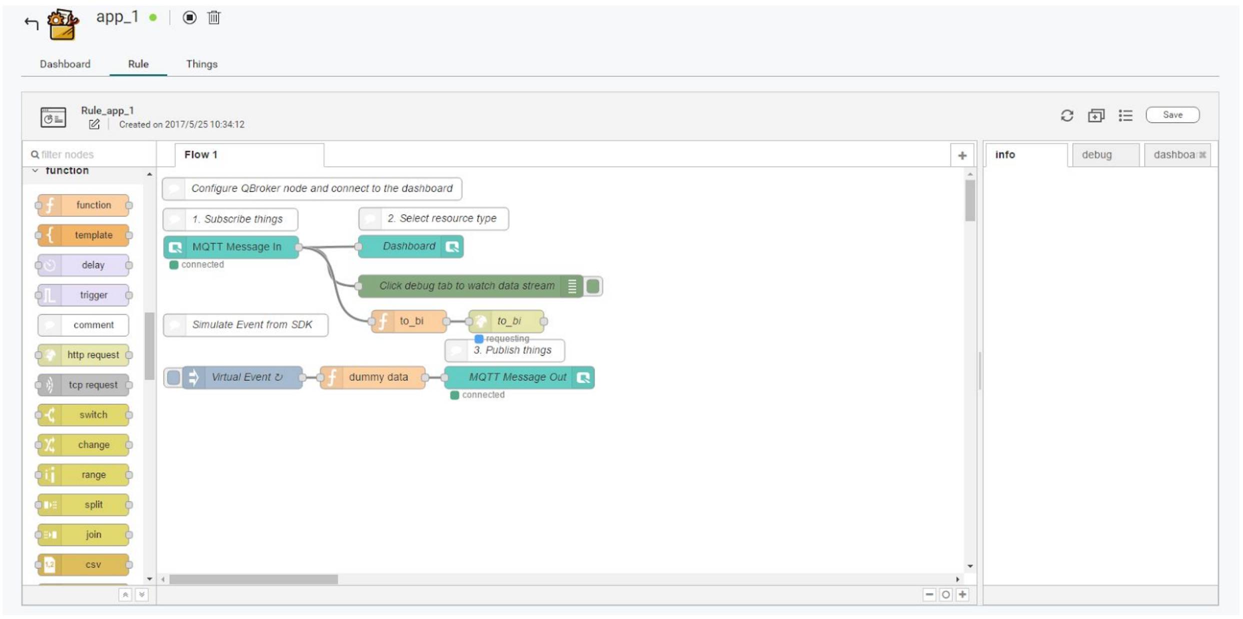
https://api.powerbi.com/beta/bb3391c7-d712-450b-949c-14d42c1dff4e/dat

Raw cURL PowerShell

```
[
 {
  "temp" : "AAAAA555555",
  "max" : "AAAAA555555",
  "min" : "AAAAA555555"
 }
]
```

4.3 Configure Node-RED's nodes in IoT application

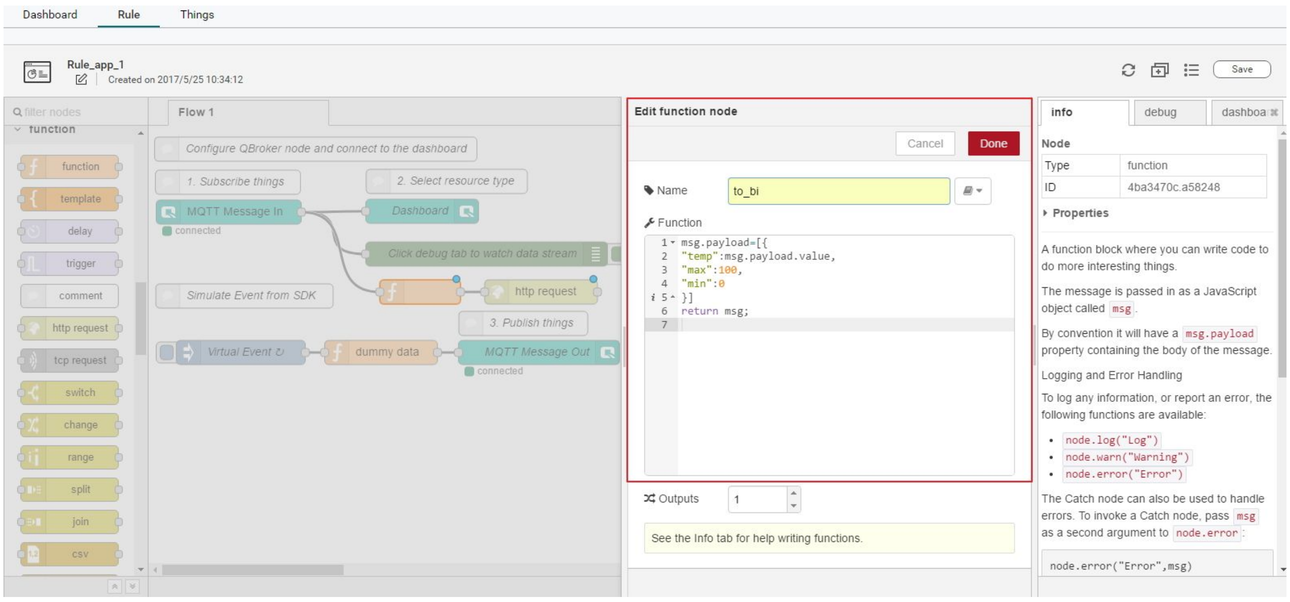
- Create IoT application in QIoT suite.
- The following one is your first node-red flow, and then you can start to create your own IoT flow. more node-red information can be found in "Node-Red".



- 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;
```



Edit function node

Cancel Done

Name: to_bi

Function

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

Outputs: 1

See the Info tab for help writing functions.

Node

Type	function
ID	4ba3470c.a58248

Properties

A function block where you can write code to do more interesting things.

The message is passed in as a JavaScript object called `msg`.

By convention it will have a `msg.payload` property containing the body of the message.

Logging and Error Handling

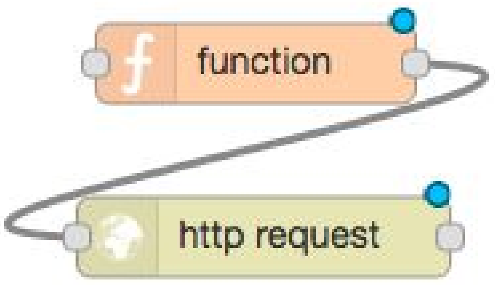
To log any information, or report an error, the following functions are available:

- `node.log("Log")`
- `node.warn("Warning")`
- `node.error("Error")`

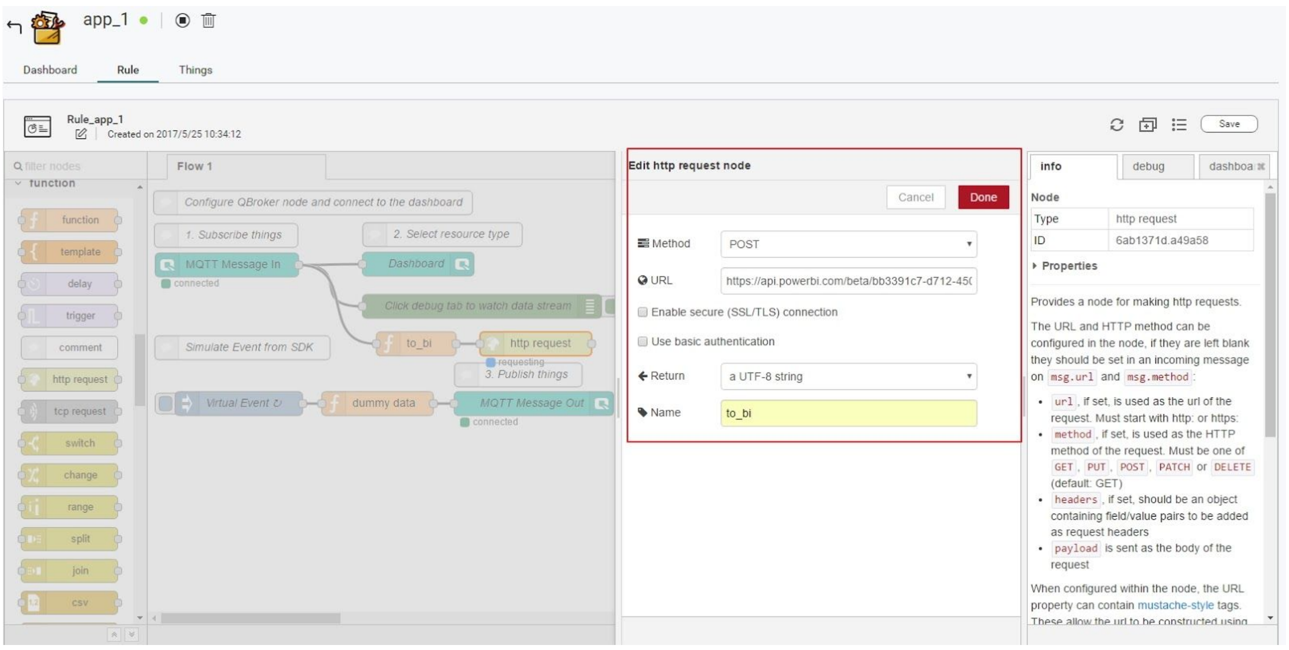
The Catch node can also be used to handle errors. To invoke a Catch node, pass `msg` as a second argument to `node.error`:

```
node.error("Error",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.



Edit http request node

Cancel Done

Method	POST
URL	https://api.powerbi.com/beta/bb3391c7-d712-45(
☐ Enable secure (SSL/TLS) connection	
☐ Use basic authentication	
Return	a UTF-8 string
Name	to_bi

Node

Type	http request
ID	6ab1371d.a49a58

Properties

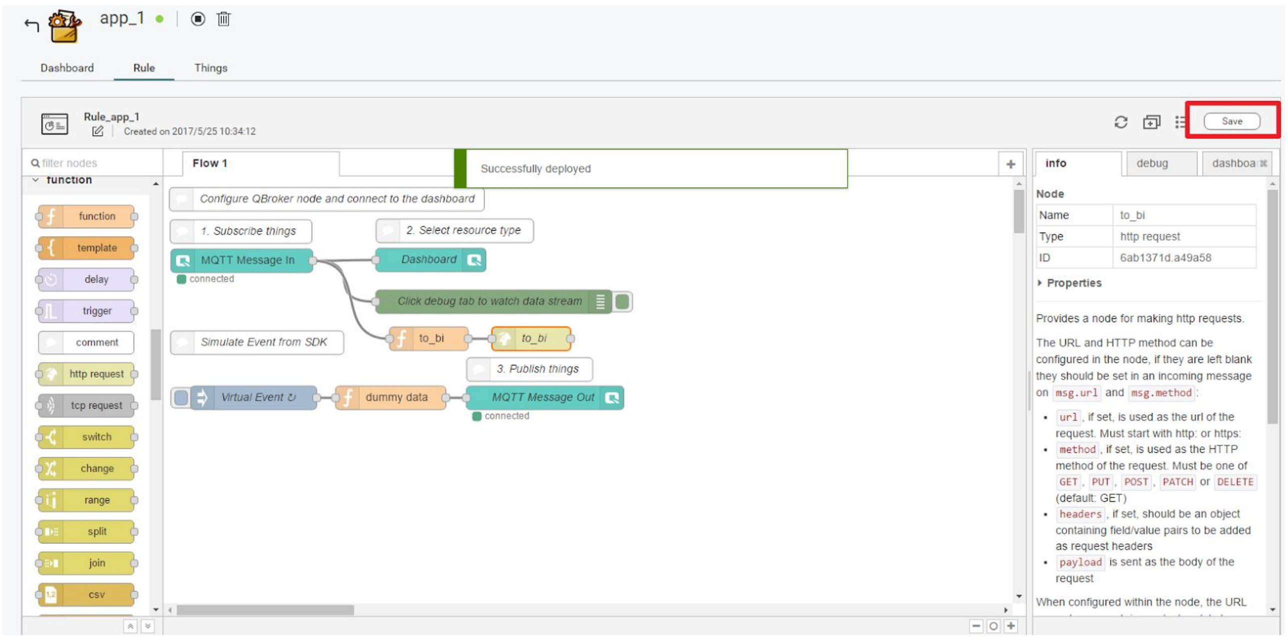
Provides a node for making http requests.

The URL and HTTP method can be configured in the node, if they are left blank they should be set in an incoming message on `msg.url` and `msg.method`:

- `url`, if set, is used as the url of the request. Must start with http: or https:
- `method`, if set, is used as the HTTP method of the request. Must be one of `GET`, `PUT`, `POST`, `PATCH` or `DELETE` (default: GET)
- `headers`, if set, should be an object containing field/value pairs to be added as request headers
- `payload` is sent as the body of the request

When configured within the node, the URL property can contain mustache-style tags. These allow the url to be constructed using

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



Successfully deployed

Node

Name	to_bi
Type	http request
ID	6ab1371d.a49a58

Properties

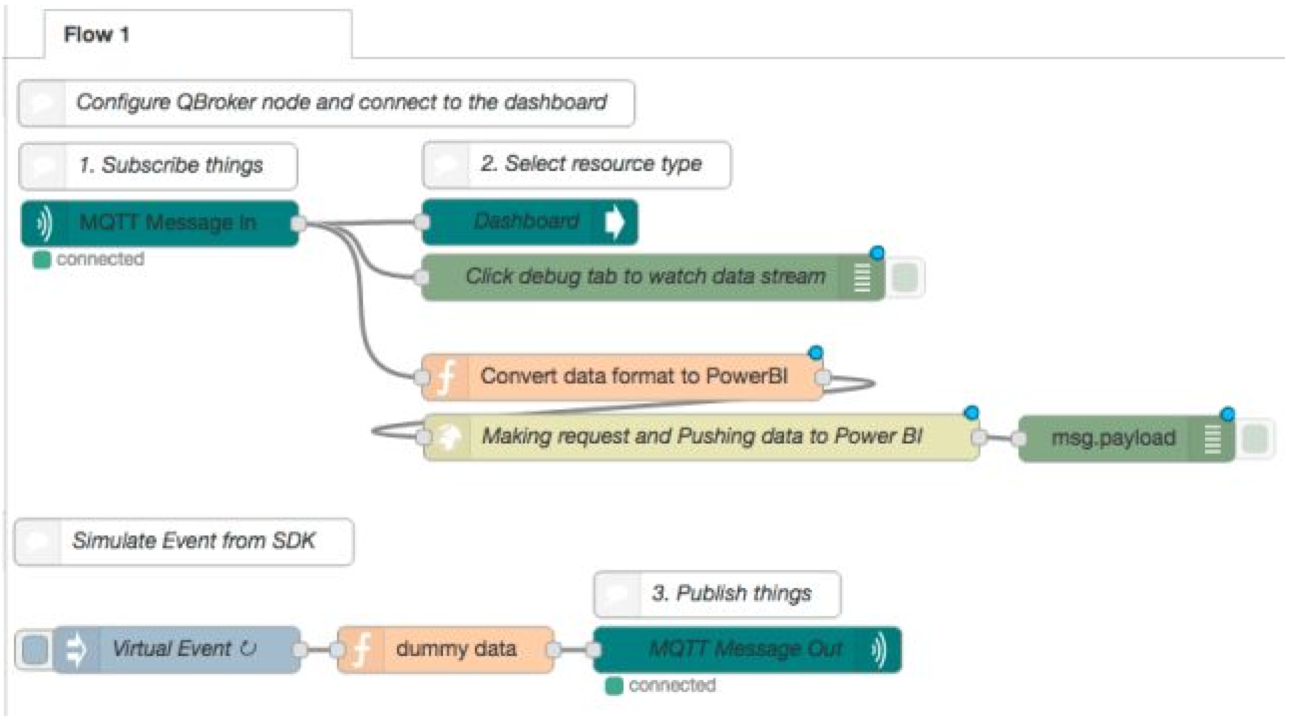
Provides a node for making http requests.

The URL and HTTP method can be configured in the node, if they are left blank they should be set in an incoming message on `msg.url` and `msg.method`:

- `url`, if set, is used as the url of the request. Must start with http: or https:
- `method`, if set, is used as the HTTP method of the request. Must be one of `GET`, `PUT`, `POST`, `PATCH` or `DELETE` (default: GET)
- `headers`, if set, should be an object containing field/value pairs to be added as request headers
- `payload` is sent as the body of the request

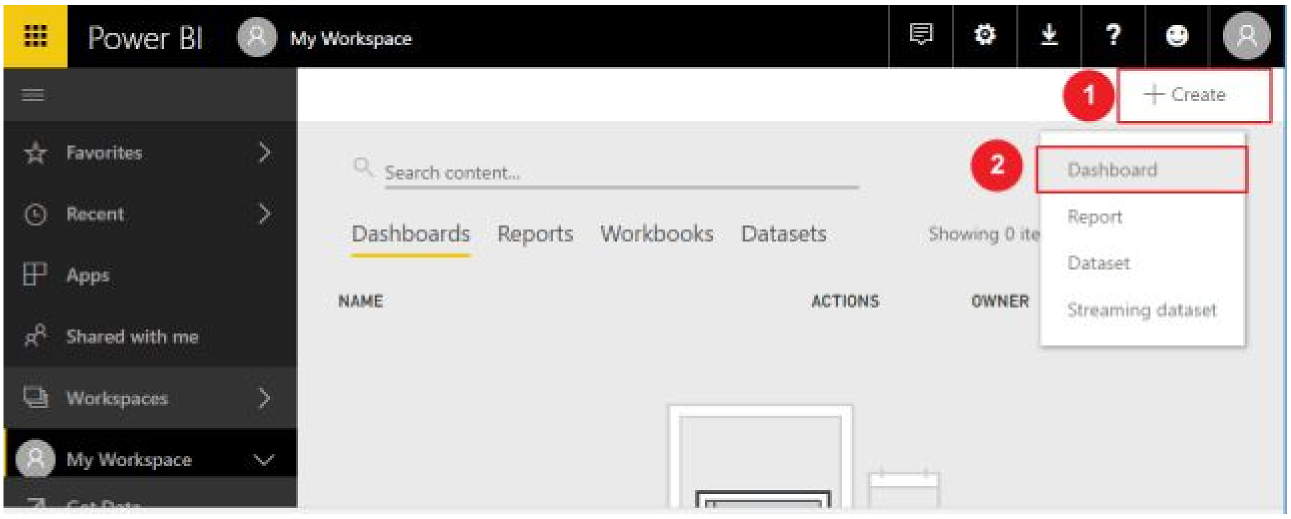
When configured within the node, the URL

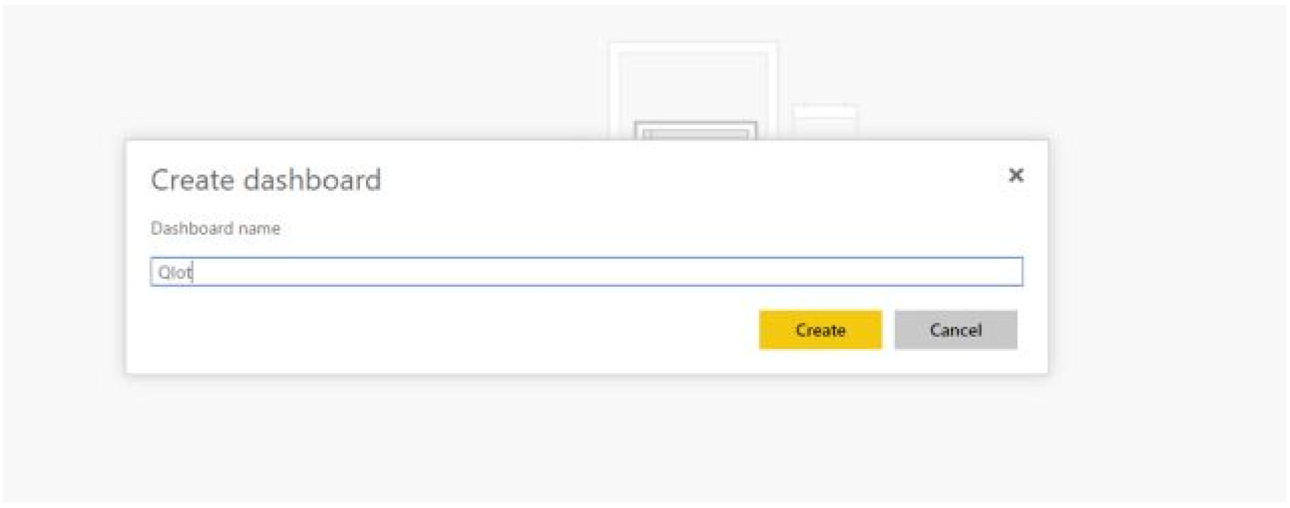
- Finally, your node-red flow will look like below one.



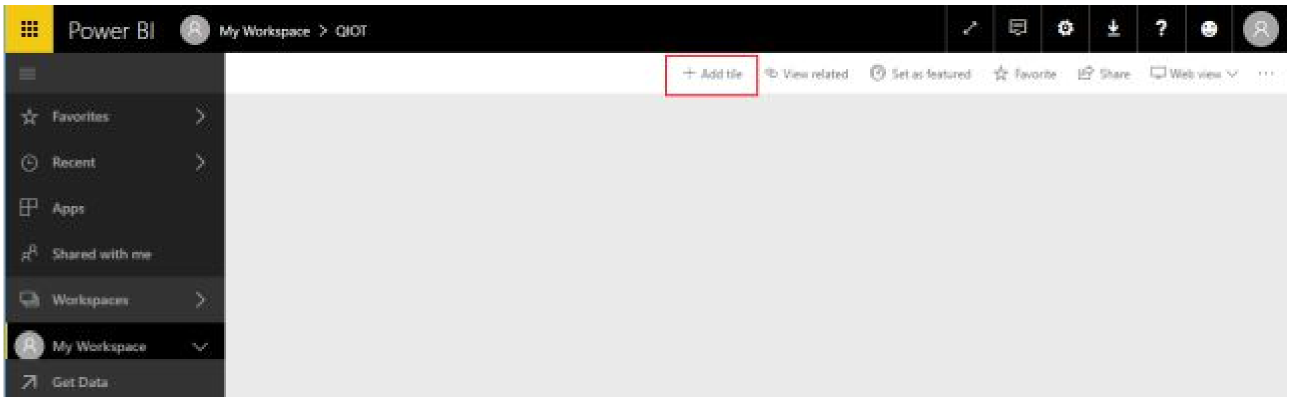
4.4 Add tile to display real-time data

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

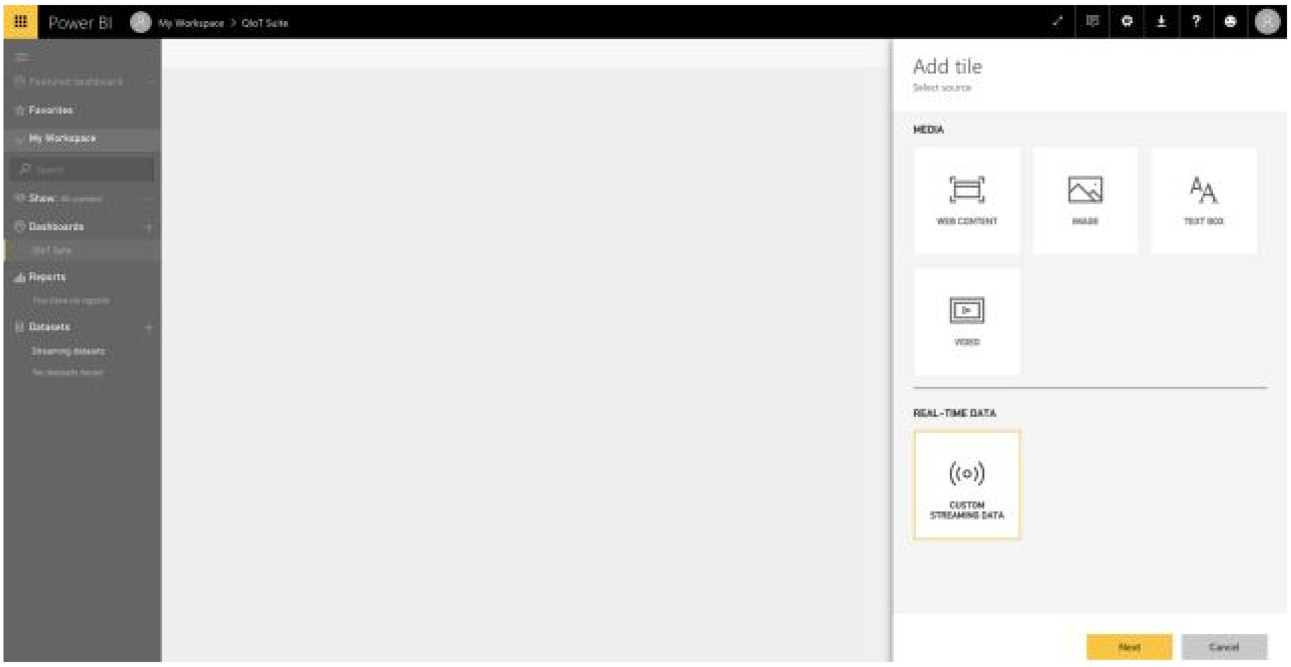




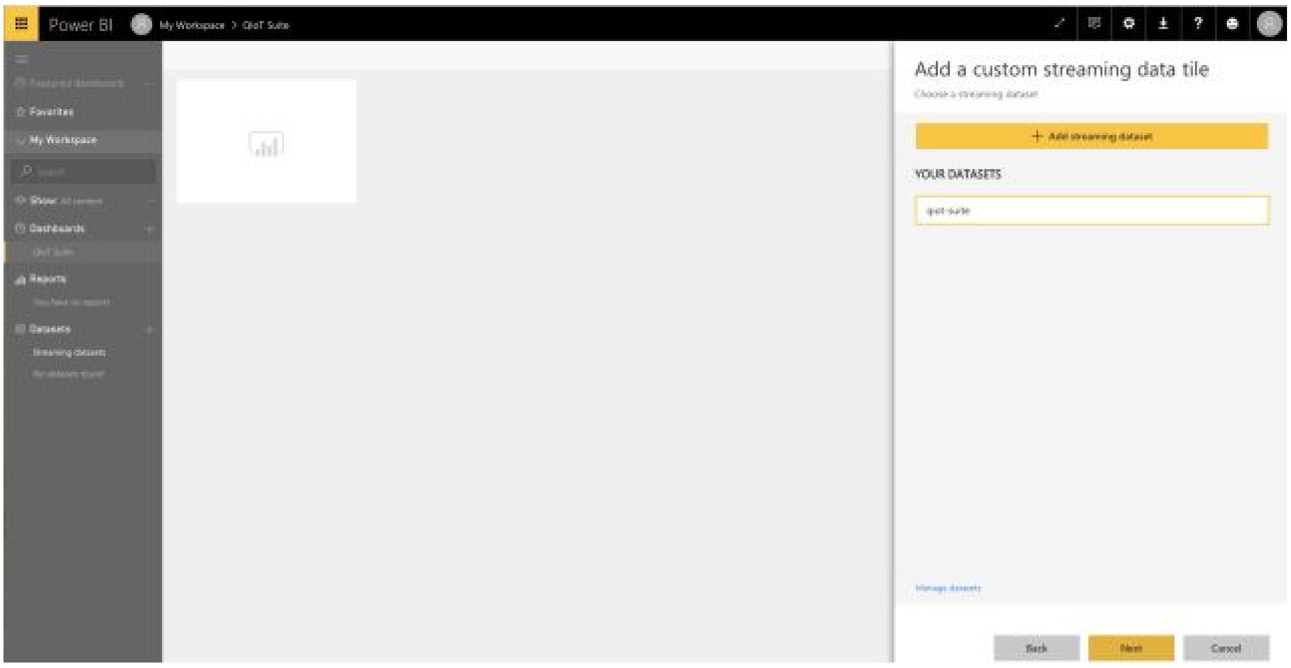
- Click “Add tile” in screen upper right corner



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



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



Add a custom streaming data tile

Choose a streaming dataset

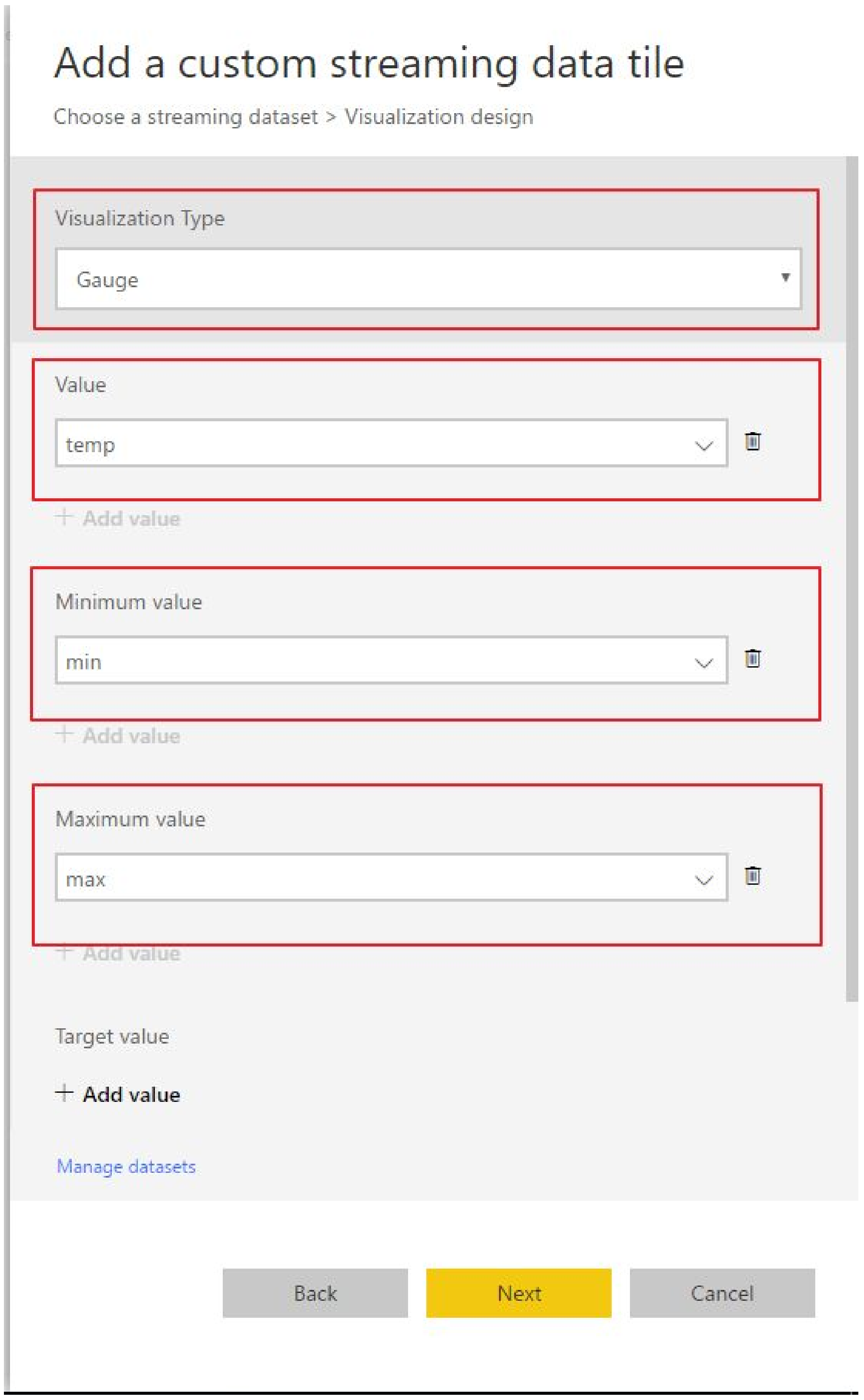
+ Add streaming dataset

YOUR DATASETS

qiot-suite

Manage datasets

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



Add a custom streaming data tile

Choose a streaming dataset > Visualization design

Visualization Type: Gauge

Value: temp

+ Add value

Minimum value: min

+ Add value

Maximum value: max

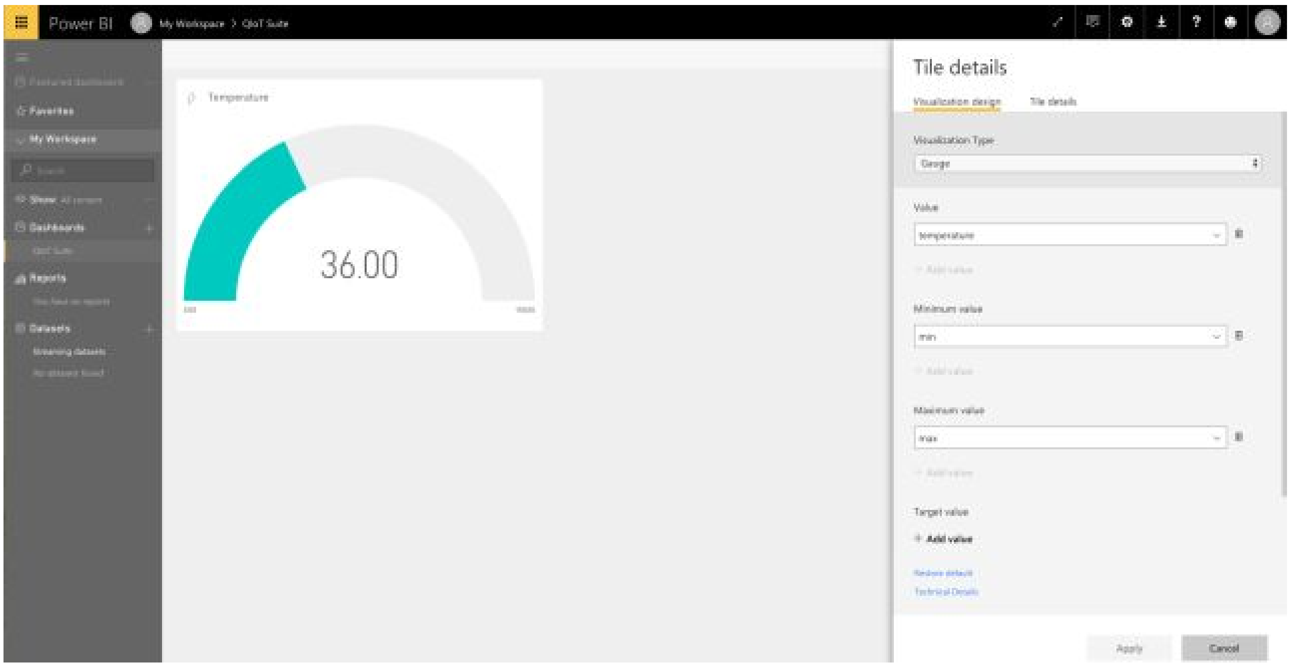
+ Add value

Target value

+ Add value

Manage datasets

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



Tile details

Visualization design | Tile details

Visualization Type: Gauge

Value: temperature

Minimum value: min

Maximum value: max

Target value: + Add value

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/
      intel-edison/
        examples/
          lib/
          res/
          ssl/
         mqtt.js
         http.js
          ...
```

- content of resourceinfo.json

protocol	resourceinfo.json content

mqtts	
	```
	{
	"host": [
	  "172.17.28.28"                         # nas ip
	],
	"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"
}
```

https

```
{
   "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
   ...
}
```

CoAP

```
{
   "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
   ...
}
```