NexAIoT Co., Ltd.

IoT Studio

User Manual

NexAIoT Co., Ltd.
Document Version: v2.1
Published September 2019

www.nexaiot.com

Contents

Preface

NexAIoT IoT Studio is a web-based configuration tool designed to develop IoT applications with or without coding. Through simple click-through or drag-and-drop actions, NexAIoT IoT Studio turns your ideas into reality.

Disclaimer

The information in this document is subject to change without prior notice and does not represent commitment from NexAIoT Co., Ltd. However, users may update their knowledge of any product in use by constantly checking its manual posted on our website: https://www.nexaiot.com. NexAIoT shall not be liable for direct, indirect, special, incidental, or consequential damages arising out of the use of any product, nor for any infringements upon the rights of third parties, which may result from such use. Any implied warranties of merchantability or fitness for any particular purpose is also disclaimed.

Acknowledgements

The IoT Studio is a trademark of NexAIoT Co., Ltd. All other product names mentioned herein are registered trademarks of their respective owners.

Revision History

Version	Date	Description
v2.0	March 2019	Initial release
v2.1	September 2019	1. Released IoT Studio version 2.20.035. 2. Added mysql node, one-click deploy to edge server & one-click deploy to cloud. 3. Added SOP for creating a virtual machine for Google / Azure / AWS Cloud.

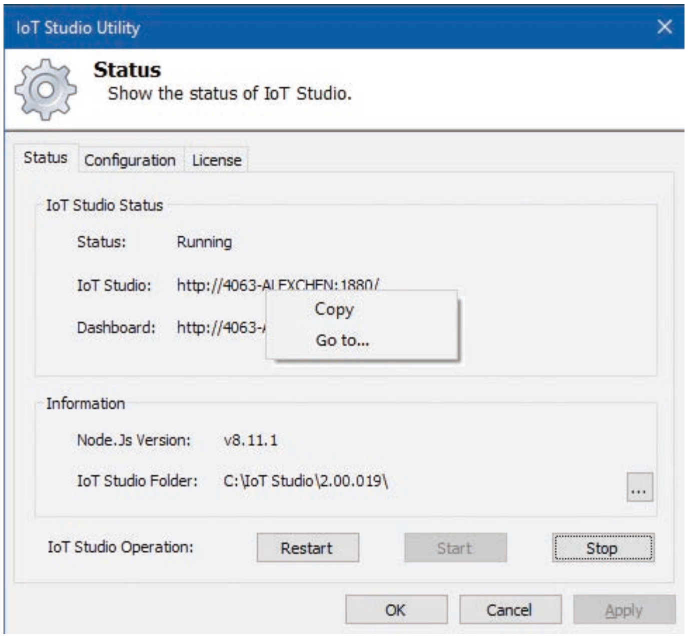
Chapter 1: Using IoT Studio

This chapter will guide you through on how to launch IoT Studio and IoT Studio Dashboard.

1.1 Launching IoT Studio

For Windows:

1. Launch IoT Studio, input the password in the respective field, and click **OK**.
2. Click the **Start** button on the right of **IoT Studio Operation:** if the status shown in **Status:** is not running.
3. Right click on the URL after **IoT Studio:** and select **Go to...** as shown.
4. The login page of IoT Studio will launch in the browser. The default username and password are both "*admin*".



Status
Show the status of IoT Studio.

IoT Studio Status

Status: Running

IoT Studio: http://4063-ALEXCHEN:1880/

Dashboard: http://4063-A

Information

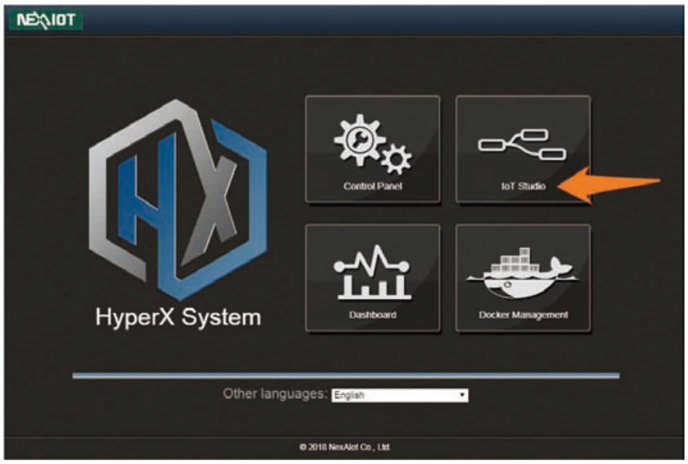
Node.Js Version: v8.11.1

IoT Studio Folder: C:\IoT Studio\2.00.019\

IoT Studio Operation:

For HyperX:

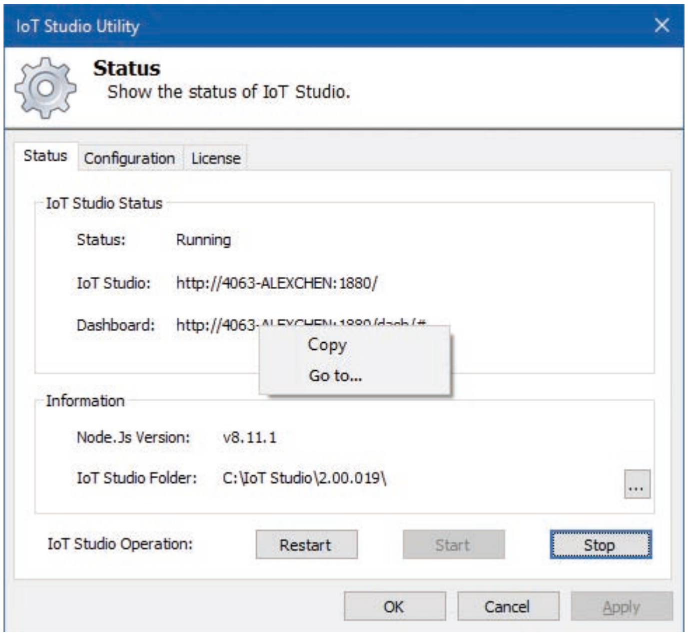
1. Log onto the main page of NexAIoT HyperX system.
2. Click on the **IoT Studio** icon as shown.
3. The login page of IoT Studio will launch in the browser. The default username and password are “*admin*” and “*12345678*” respectively.



1.2 Launching IoT Studio Dashboard

For Windows:

1. Launch IoT Studio, input the password in the respective field, and click **OK**.
2. Click the **Start** button on the right of **IoT Studio Operation:** if the status shown in **Status:** is not running.
3. Right click on the URL after **Dashboard:** and select **Go to...** as shown.
4. The login page of Dashboard will launch in the browser. The default username and password are both *"admin"*.



IoT Studio Utility

Status
Show the status of IoT Studio.

IoT Studio Status

Status: Running

IoT Studio: http://4063-ALEXCHEN:1880/

Dashboard: http://4063-ALEXCHEN:1880/dash/#

Information

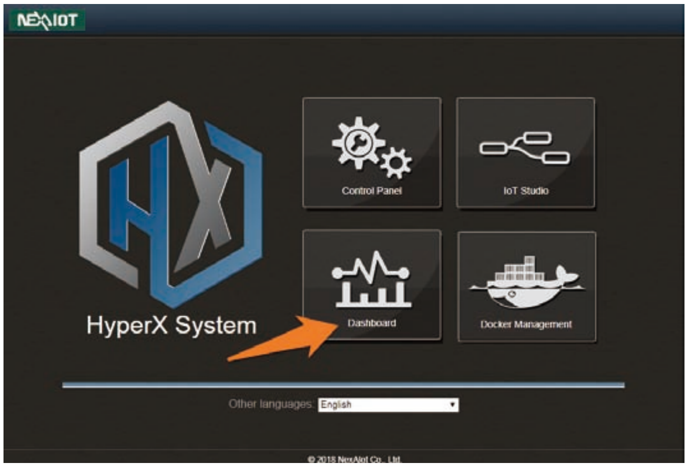
Node.Js Version: v8.11.1

IoT Studio Folder: C:\IoT Studio\2.00.019\

IoT Studio Operation:

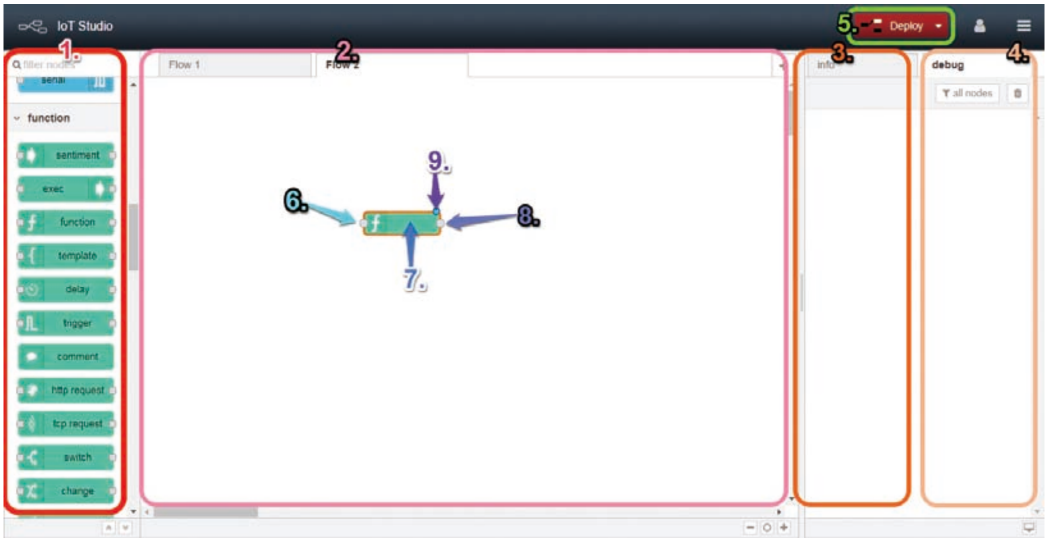
For HyperX:

1. Log onto the main page of NexAIoT HyperX system.
2. Click on the **Dashboard** icon as shown.
3. The login page of Dashboard will launch in the browser. The default username and password are “*admin*” and “*12345678*” respectively.



Chapter 2: IoT Studio Basics

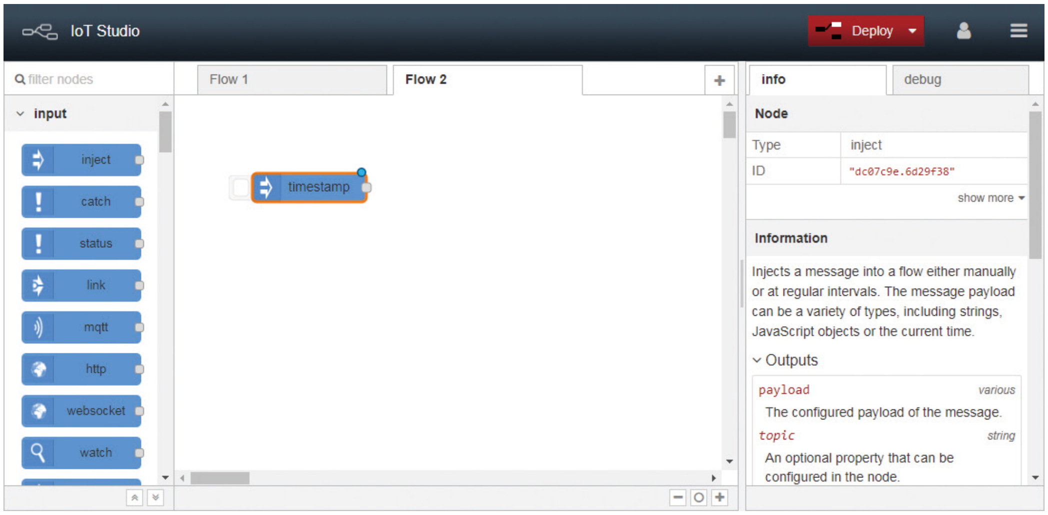
This chapter introduces the user interface and the basic operation of NexAIoT IoT Studio. Once you log onto NexAIoT IoT Studio with your browser, you will see the page as shown below.



Item	Description	Item	Description
1	Nodes	6	Input port of the node, received from a connected node.
2	Workspace	7	Node title
3	Information	8	Output port of the node, delivered to other node.
4	Debug message when debugging a flow.	9	Status of the node.
5	Press to deploy your flow to the server.		

2.1 Basic Operations

The icons on the left side of the page are nodes corresponding to different needs. You can drag and drop any of them to the workspace. Click on the **Info** tab on the upper right to view the information of the node selected.



Node

Type	inject
ID	"dc07c9e.6d29f38"

Information

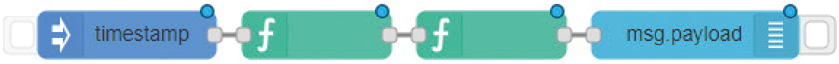
Injects a message into a flow either manually or at regular intervals. The message payload can be a variety of types, including strings, JavaScript objects or the current time.

Outputs

payload various
The configured payload of the message.
topic string
An optional property that can be configured in the node.

2.1.1 Drag and Drop

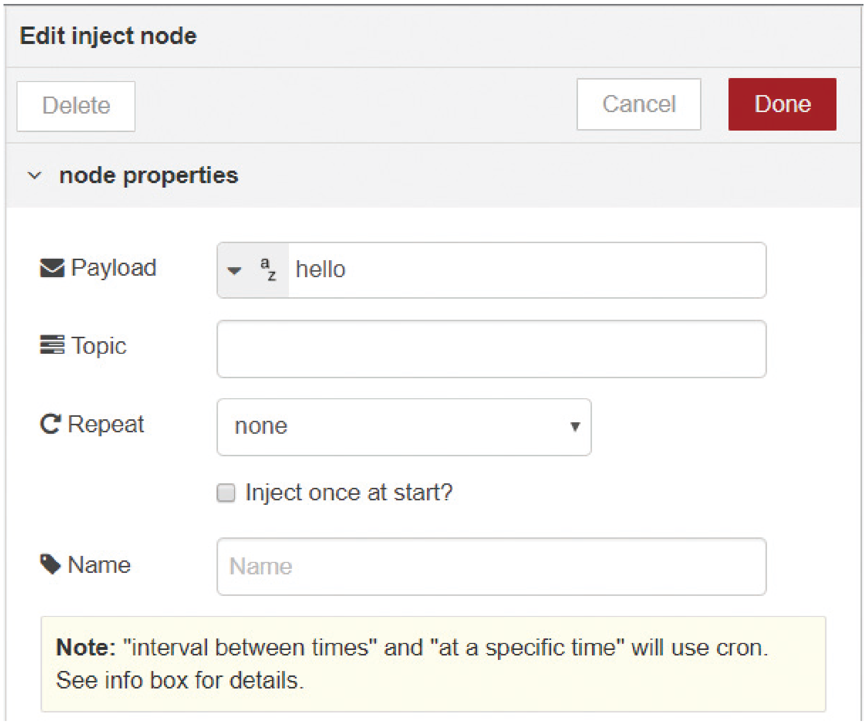
You can drag multiple nodes onto a sheet and make them a flow based on the functions and connections. Please note that due to the nature of the different nodes, some of them have both the input and the output ports while the others have either one of the ports.



You can make as many connections as you like between your start and end nodes. Furthermore, you can connect one output port to different input nodes. Once the connection is set, you can code up the flow.

2.1.2 Code Up Your Flow

Double click on the inject node, and the edit dialogue should pop up. Select **string** in the drop-down menu next to Payload, fill the next field with hello, and click **Ok**.



Payload: ᵃz hello

Topic:

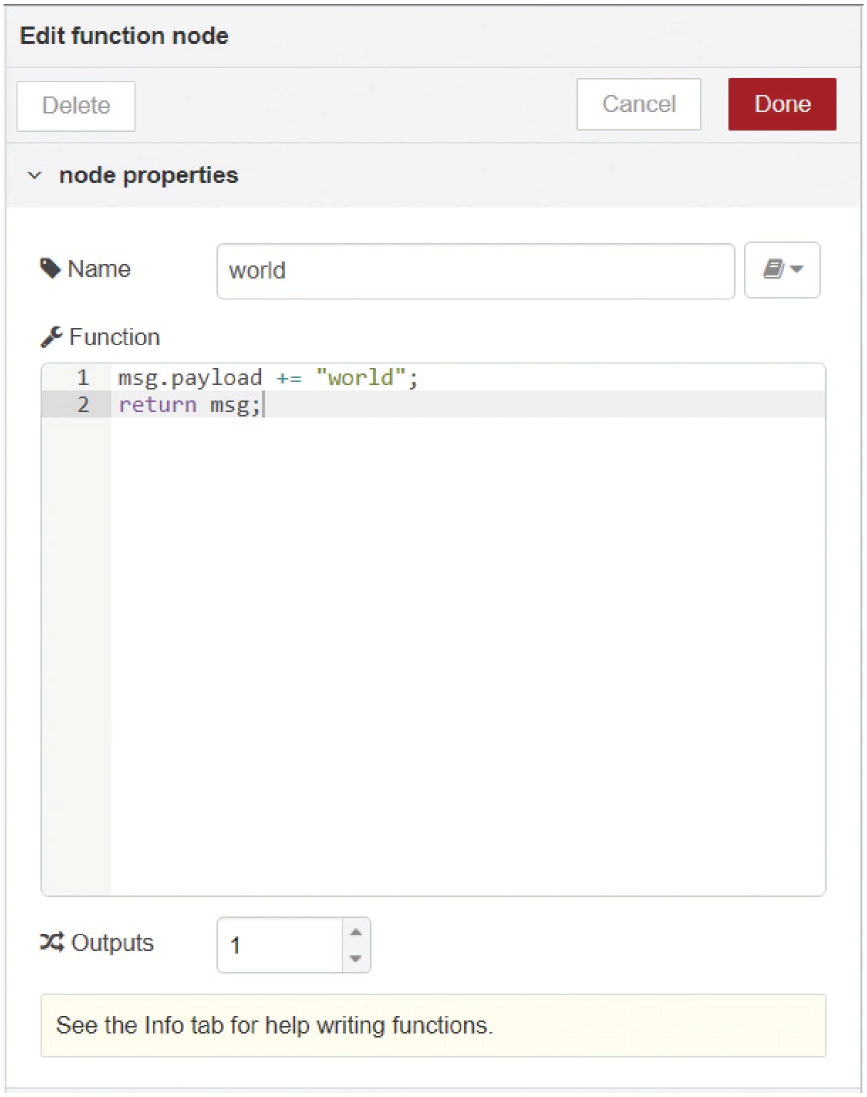
Repeat: none

□ Inject once at start?

Name: Name

Note: "interval between times" and "at a specific time" will use cron. See info box for details.

Double click on the second node, and name it world. Write some codes to the node in the function field and click **Ok**.



Name: world

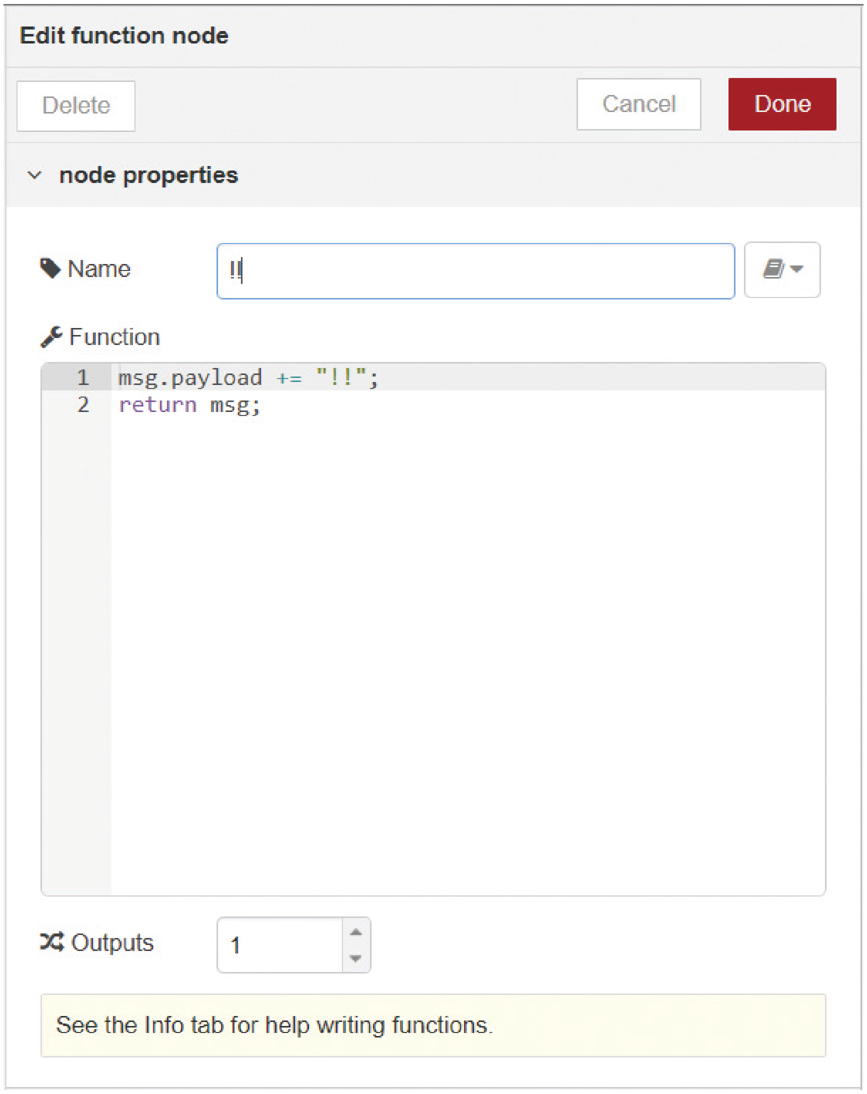
Function

```
msg.payload += "world";
return msg;
```

Outputs: 1

See the Info tab for help writing functions.

Double click on the third node, and name it !!. Write some codes to the node in the function field as shown below and click **Ok**.



Name: !!

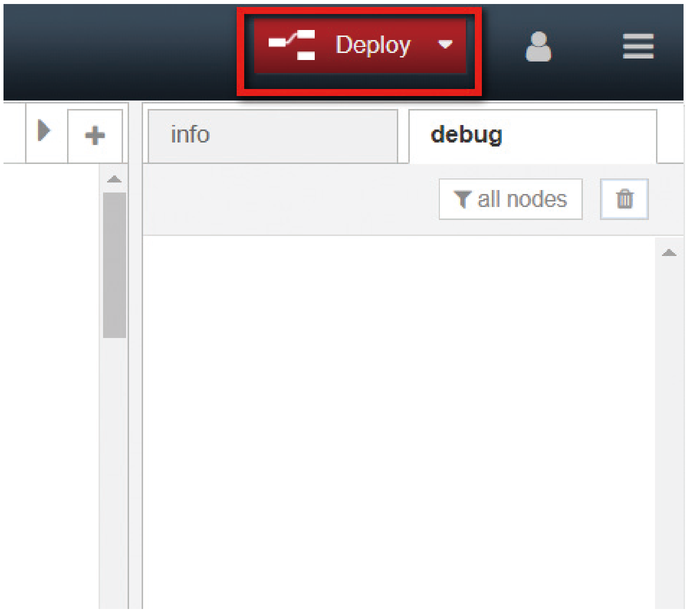
Function

```
msg.payload += "!!";
return msg;
```

Outputs: 1

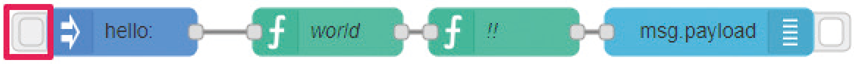
See the Info tab for help writing functions.

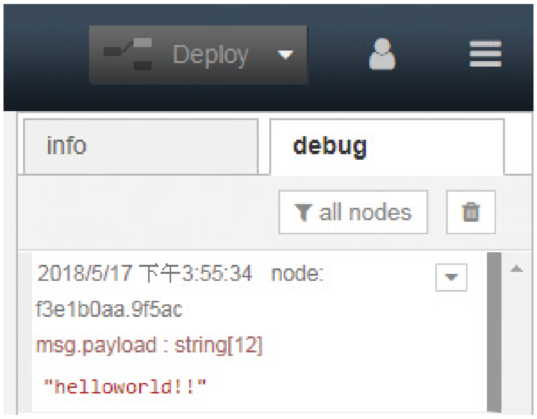
Then, click the **Deploy** button on the upper right side to deploy your flow.



When you can read the message **Successfully deployed** on the top, the deployment is complete.

Click the button on the left side of the inject node to see the result in the **debug** tab.





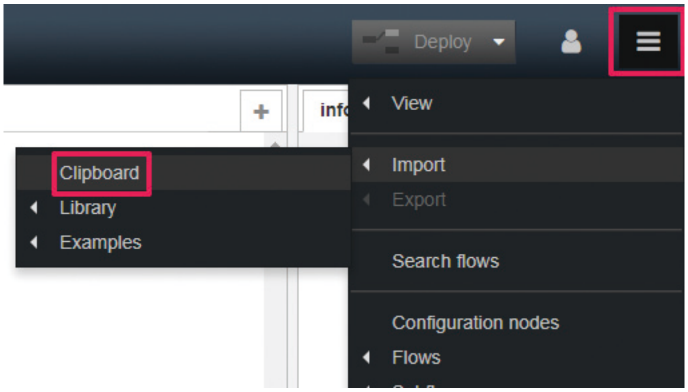
```
2018/5/17 下午3:55:34  node:
f3e1b0aa.9f5ac
msg.payload : string[12]
"helloworld!!"
```

2.2 IoT Studio Administrations

You can share the codes by selecting your node, and then choose **Export** from the menu on the upper right corner either by copying the codes or exporting them to a file directly.

2.2.1 Import

To import the codes, you can choose **Import** from the menu on the upper right corner and paste the codes to the clipboard or select a file to import.

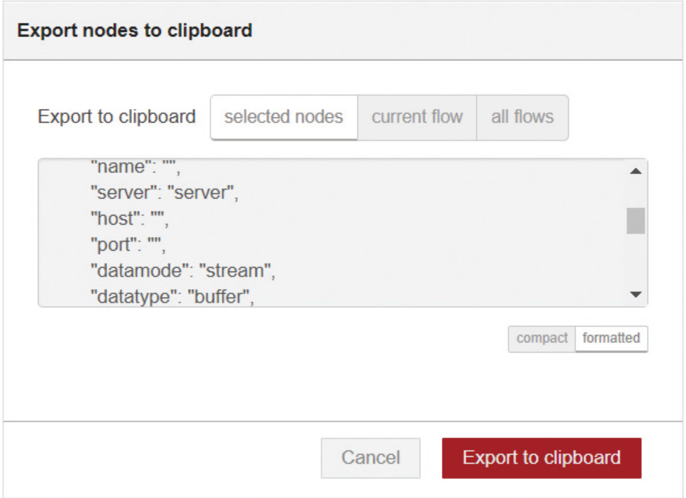


Click **Deploy** to execute the code in the flow.

2.2.2 Export

To export your codes, select the node and choose **Export** from the menu on the upper right corner.

You can copy the code or export the code to a file directly.

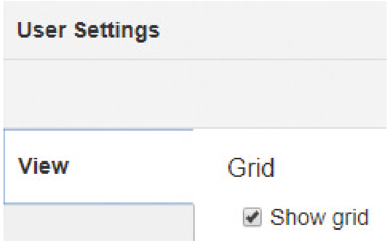


```
        "name": "",
        "server": "server",
        "host": "",
        "port": "",
        "datamode": "stream",
        "datatype": "buffer",
```

2.2.3 Grid

You can turn on grid to better organize nodes in the workspace.

To turn on grid, choose **Settings** from the menu on the upper right corner, and check **Show grid**.



User Settings

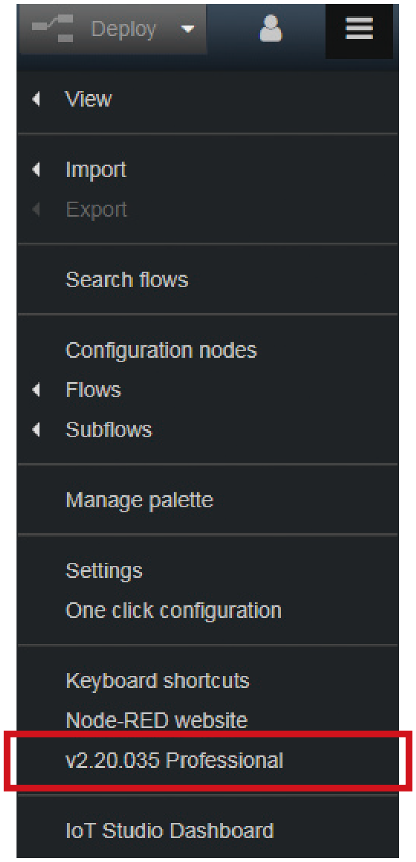
View

Grid

☑ Show grid

2.2.4 Version

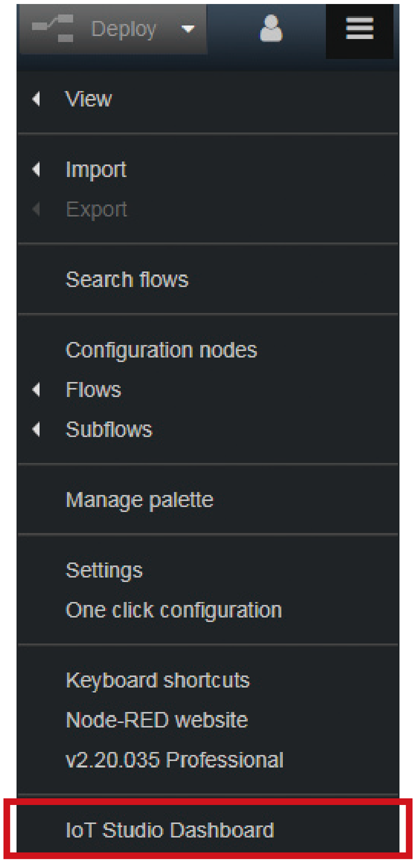
To check the release note of your current NexAIoT IoT Studio, choose the version number from the menu on the upper right corner as shown below, and you can find details of the release note in the **Info** tab.



Note: Version number varies from the license.

2.2.5 Dashboard

You can launch IoT Studio Dashboard with a simple click. To do so, choose **IoT Studio Dashboard** from the menu on the upper right corner, and the Dashboard page will launch in the browser.



Chapter 3: IoT Studio Notes

3.1 Input Nodes

Input nodes load data from various interfaces and transfer to their next stops.

3.1.1 inject

Pressing the button on the left of the node allows a message on a topic to be injected into the flow.

Item	Option	Description
Payload	**flow**	The flow variable.
	global	The global variable.
	string	The string (character type).
	number	The number.
	boolean	false/true
	JSON	The json format.
	timestamp	The current time in milliseconds since 1970.
Topic		The string used to filter messages.
Repeat	**none**	The repeat function allows the payload to be sent on the required schedule.
	Interval	
	Interval between times	
	at a specific time	
	Inject once at start?	This option actually waits a short interval before firing to give other nodes a chance to instantiate properly.
Name		The name of the node.

Note:

- "Interval between times" and "at a specific time" use cron job (crontab). This means that 20 minutes will be at the next hour: at the 20 minute-mark, another 40 minutes is needed to reach 1 hour, not in 20 minutes time. If you want every 20 minutes from now, use the "interval" option.
- All string input is escaped. To add a carriage return to a string you should use a following function.

3.1.2 catch

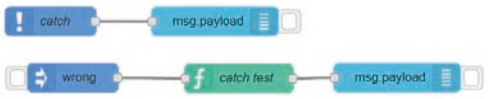
The catch node catches errors thrown by nodes on the same tab. If a node throws an error whilst handling a message, the flow will typically halt. This node can be used to catch those errors and handle them with a dedicated flow. The node will catch errors thrown by nodes on the same tab. If there are multiple catch nodes on a tab, the nodes will all get triggered.

If an error is thrown within a subflow, the error will get handled by any catch nodes within the subflow. If none exists, the error is propagated up to the tab the subflow instance is on.

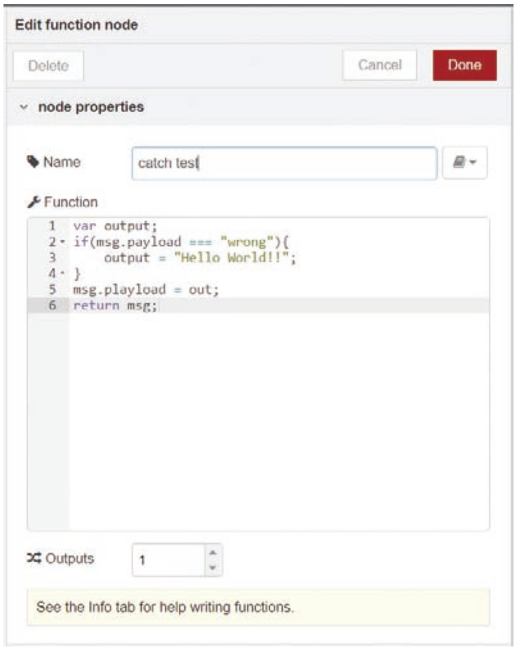
The message sent by this node will be the original message if the node that threw the error provided it. The message will have an error property with the following attributes:.

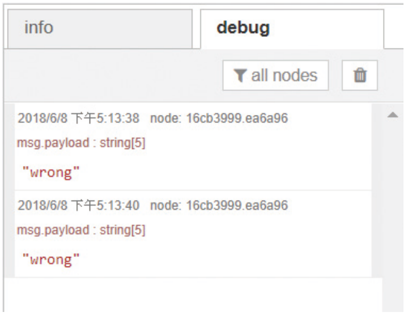
message: The error message.
source.id: The id of the node that threw the error.
source.type: The type of the node that threw the error.
source.name: The name, if set, of the node that threw the error.

If the message already had an error property, it is copied to error.

Example: Use catch node to problematic nodes.		
Step	**Description**	**Screenshot**
1	Add and connect 1 **inject** node, 1 **function** node, 1 **catch** node, and 2 **debug** nodes to the workspace as shown.	
2	Edit the **inject** node by setting msg.payload to be a string **wrong** and click **Done**.	

Example: Use catch node to problematic nodes.		
Step	**Description**	**Screenshot**

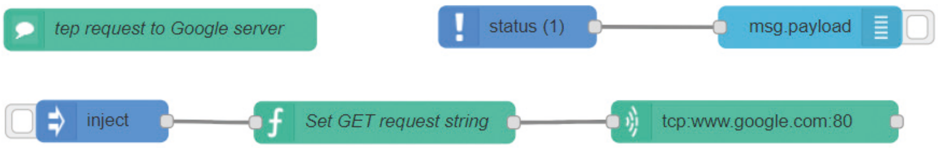
3	Edit the **function** node as shown and click **Ok**. Notice that there is an error in the code on purpose.	
		Edit function node
		Delete Cancel Done
		node properties
		Name: catch test
		Function
		`var output;`
		`if(msg.payload === "wrong"){`
		`    output = "Hello World!!";`
		`}`
		`msg.playload = out;`
		`return msg;`
		Outputs: 1
		See the Info tab for help writing functions.

4	Deploy your flow and click the button on the left of the inject node. The user shall see the debug information as shown. The message in red is the message that the catch node throws to indicate there is a problem with "wrong" node.	 info debug all nodes 2018/6/8 下午5:13:38 node: 16cb3999.ea6a96 msg.payload : string[5] "wrong" 2018/6/8 下午5:13:40 node: 16cb3999.ea6a96 msg.payload : string[5] "wrong"

3.1.3 status

The status node sends status message from other nodes on the same tab.

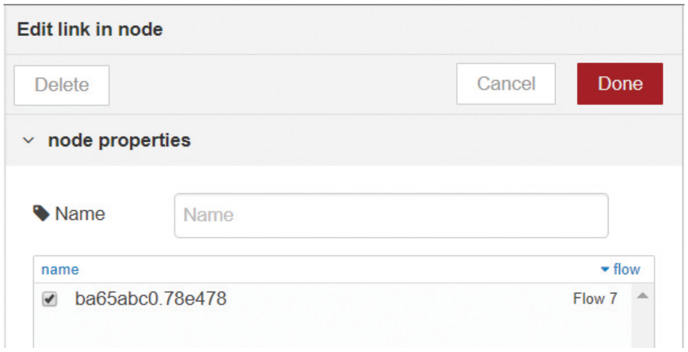
Example: Get the status of the URL www.google.com.



3.1.4 link

The link input node can be connected to any link out node that exists on any tab. Once connected, they behave as if they were wired together. The wires between link nodes are only displayed when a link node is selected. If there are any wires to other tabs, a virtual node will be shown and can be clicked on to jump to the appropriate tab. Links cannot be created going into, or out of, a subflow.

Click on the node to select which link out node it will connect to. Check the checkbox of the node to connect to, and click **Done** when finished.



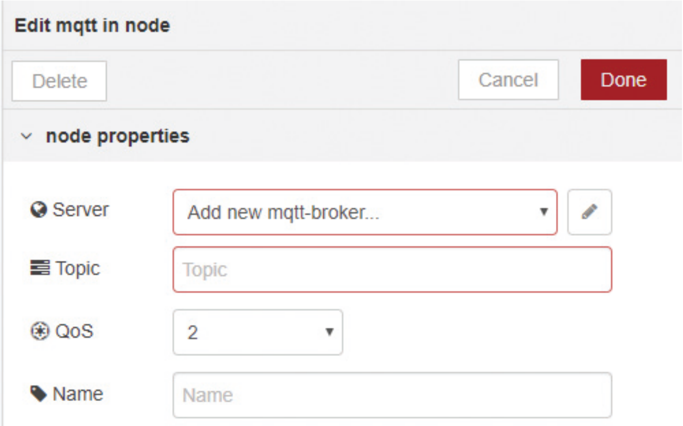
name	flow
☑ ba65abc0.78e478	Flow 7

3.1.5 mqtt

The mqtt input node connects to a broker and subscribes to the specified topic. The topic may contain MQTT wildcards. Outputs and object called msg contain the following:

msg.topic
msg.payload
msg.qos
msg.retain

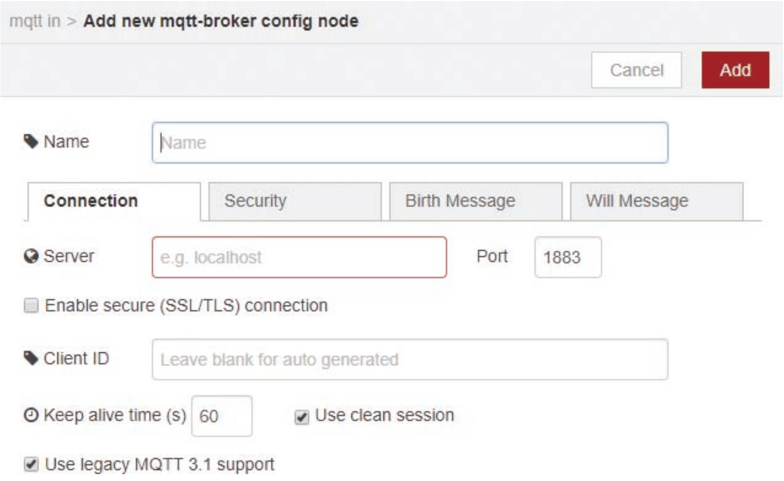
msg.payload is usually a string, but can be a binary buffer.



Server: Add new mqtt-broker...

QoS: 2

Item	Option	Description
Server		The MQTT broker that is currently used.
Topic		The string used by the broker to filter messages. A topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator).
QoS	The Quality of Service (QoS) level is an agreement between sender and receiver of a message regarding the guarantees of delivering a message.	
	0 (default)	The message is delivered at most once, or it is not delivered at all. Its delivery across the network is not acknowledged. The message is not stored. The message might be lost if the client is disconnected, or if the MQTT broker fails.
	1	The message is always delivered at least once. If the sender does not receive an acknowledgement, the message is sent again with the DUP flag set until an acknowledgement is received. As a result, the receiver can be sent the same message multiple times, and might process it multiple times.
	2	The message is always delivered exactly once. The message must be stored locally at the sender and receiver until it is processed.
Name		The name of the node.



mqtt in > Add new mqtt-broker config node

Name: Name

Server: e.g. localhost Port: 1883

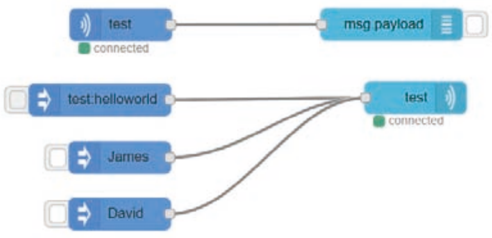
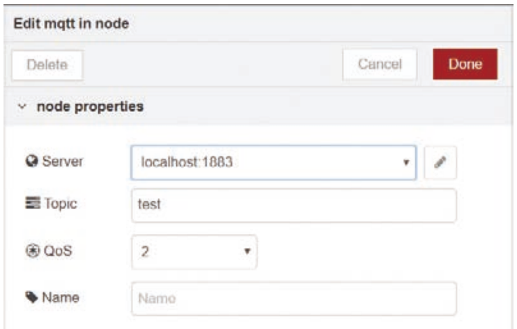
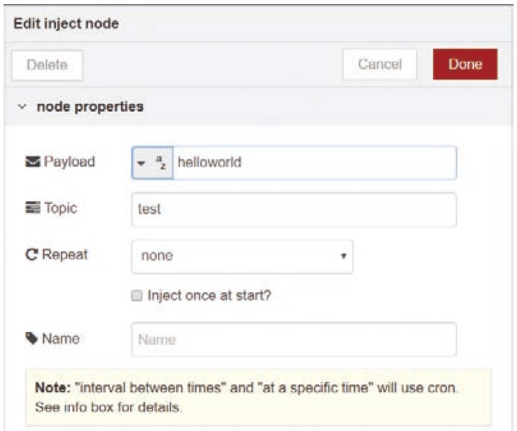
□ Enable secure (SSL/TLS) connection

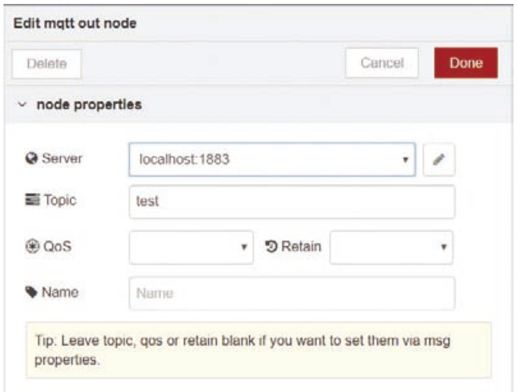
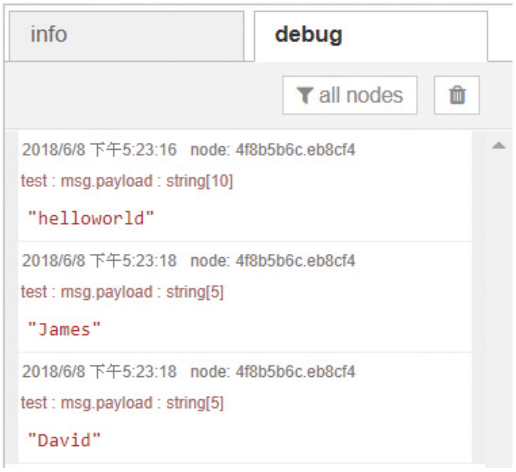
Client ID: Leave blank for auto generated

Keep alive time (s): 60 ☑ Use clean session

☑ Use legacy MQTT 3.1 support

Category	Item	Description
Connection	Server	The URL or the IP address of MQTT broker.
	Port	The network port listening to publish/ subscribe, requests with the default value 1883.
	Enable secure session	Enable or disable SSL/TTS security.
	Client ID	With the unique Client ID, the broker can recognize when a client reconnects and close an old potentially half-open TCP connection for the client.
	Keep alive time	After a period of inactivity, client will send a request and expect the broker to respond. The setting sets how often should clients check connection.
Security	Username	The broker refuses the anonymous connection by setting "allow_anonymous false".
	Password	
Birth Message	Whenever a client is connected, the client just connected will send a birth message to the topic to notify a new connected client.	
Will Message	Whenever a client is ungracefully disconnected, it sends a last message (aka will message) to the topic to notify an ungracefully disconnected client.	

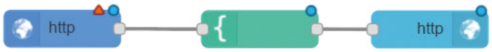
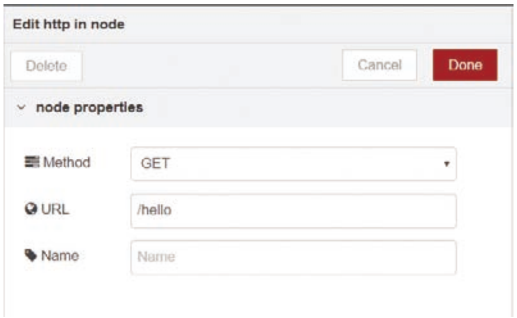
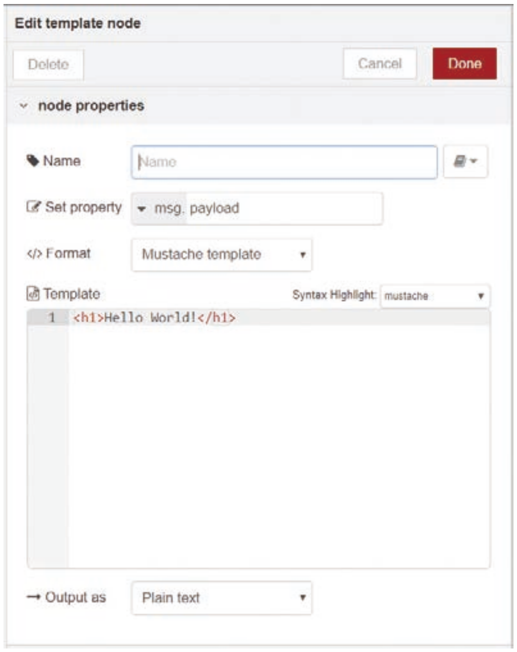
Example: Build up an mqtt connection scenario.		
Step	**Description**	**Screenshot**
1	Add and connect 3 **inject** nodes, 1 **mqtt input** node, 1 **mqtt output** node, and 1 **debug** node to the workspace as shown.	 test connected msg.payload test:helloworld test connected James David
2	Set the **mqtt input** node with your mqtt broker details and click **Ok**.	Edit mqtt in node Server localhost:1883 Topic test QoS 2 Name Name
3	Edit an **inject** node. Select **string** in **Payload**. Enter helloworld in the **Payload** field, and enter test in the **Topic** field.	Edit inject node Payload helloworld Topic test Repeat none □ Inject once at start? Name Name **Note:** "interval between times" and "at a specific time" will use cron. See info box for details.
4	Edit the other nodes. Select **string** in **Payload**. Enter James and David respectively in the **Payload** field of each node.	

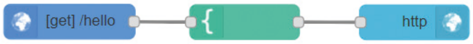
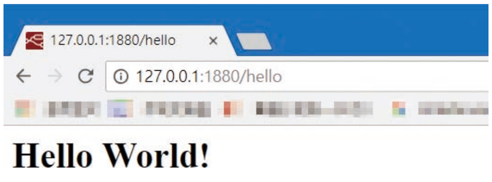
Example: Build up an mqtt connection scenario.		
Step	**Description**	**Screenshot**
5	Set the **mqtt output** node with your mqtt broker details click **Ok**.	 Edit mqtt out node Delete Cancel Done node properties Server localhost:1883 Topic test QoS Retain Name Name Tip: Leave topic, qos or retain blank if you want to set them via msg properties.
6	Deploy the flow and click the button on the left of the three inject nodes, and three messages will show up in the debug tab as shown.	 info debug all nodes 2018/6/8 下午5:23:16 node: 4f8b5b6c.eb8cf4 test : msg.payload : string[10] "helloworld" 2018/6/8 下午5:23:18 node: 4f8b5b6c.eb8cf4 test : msg.payload : string[5] "James" 2018/6/8 下午5:23:18 node: 4f8b5b6c.eb8cf4 test : msg.payload : string[5] "David"

3.1.6 http

The http input node allows the creation of simple web services. This node does not send any response to the http request. This should be done with a subsequent HTTP Response node.

Method	**GET/POST/PUT/DELETE/PATCH**
URL	

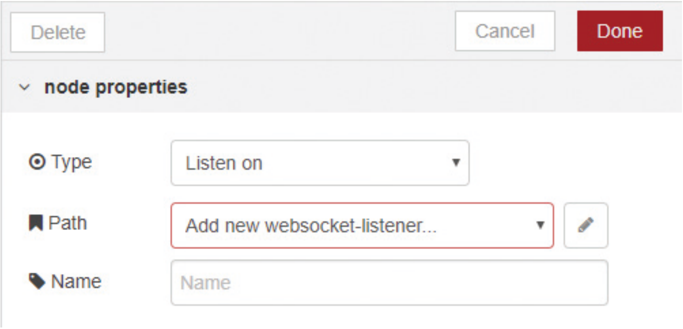
Example: Create a HTTP request and return a page with "Hello World!".		
Step	**Description**	**Screenshot**
1	Add and connect 1 **http** node, 1 **template** node, and 1 **http response** node to the workspace as shown.	
2	Edit **http** node, set **URL** to **/hello** and click **Ok**.	Edit http in node Delete Cancel Done node properties Method: GET URL: /hello Name: Name
3	Edit **template** node, add **<h1>Hello World!</h1>** in **template** node and click **Ok**.	Edit template node Delete Cancel Done node properties Name: Name Set property: msg. payload Format: Mustache template Template Syntax Highlight: mustache `<h1>Hello World!</h1>` Output as: Plain text

Example: Create a HTTP request and return a page with "Hello World!".		
Step	**Description**	**Screenshot**
4	Confirm the changes on the nodes shown on the right and deploy the flow.	
5	Open a new tab on local web browser with the address **http://device IP:1880/hello**, and the result will be as shown.	 Hello World!

3.1.7 websocket

The websocket input node provides a duplex TCP connection designed to allow web browsers and servers to maintain a "backchannel" that could be used to augment traditional HTTP interactions, allowing servers to update web pages without the client making a new pull request.

It features input and output function that allow users to listen for incoming data or to send output data on a websocket. The output version is designed to check to see if the output payload is originated at a websocket in a node, in which case it responds to the original sender. Otherwise, it will broadcast the payload to all connected websockets. Furthermore, both input and output websocket nodes can be configured as either a server listening on a URL or a client connecting to a specified IP address.

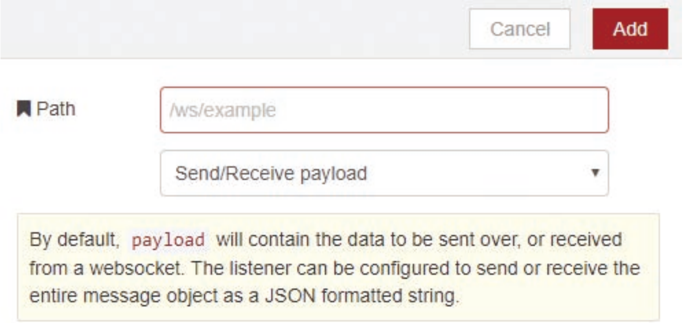


Type: Listen on

Path: Add new websocket-listener...

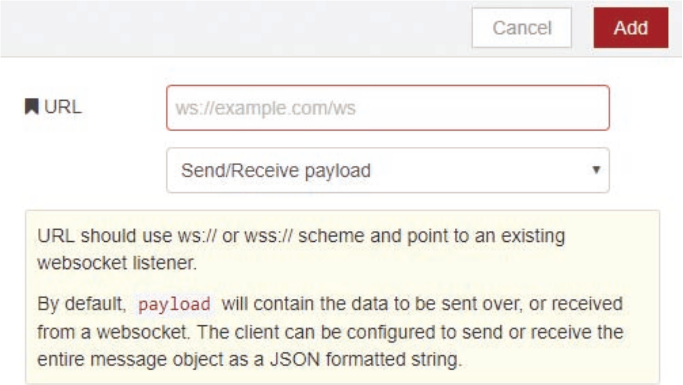
Name: Name

Item	Option	Description
Type	**Listen on**	Creates a path and awaits a remote device to connect.
	Connect to	Connect to the target websocket.
Path/URL	**Path**	Only available on type "Listen on".
	URL	Only available on type "Connect to".
Name	The name of the node.	



By default, payload will contain the data to be sent over, or received from a websocket. The listener can be configured to send or receive the entire message object as a JSON formatted string.

Add new websocket-listener config (Listen on)		
Item	**Option**	**Description**
Path		The path of websocket.
Send/ Receive	**Payload**	Will only send/receive msg.payload.
	Entire message	Will send the entire msg labeled message.



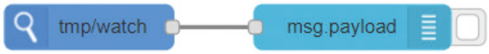
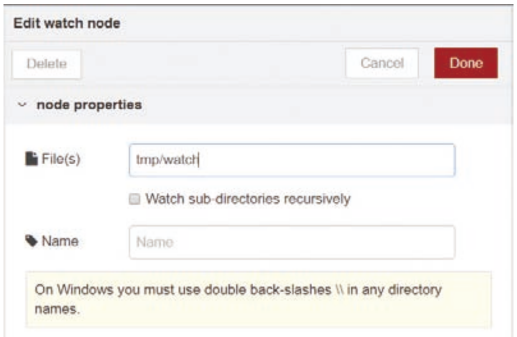
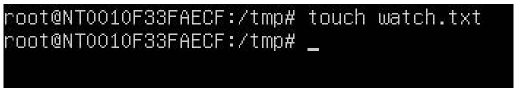
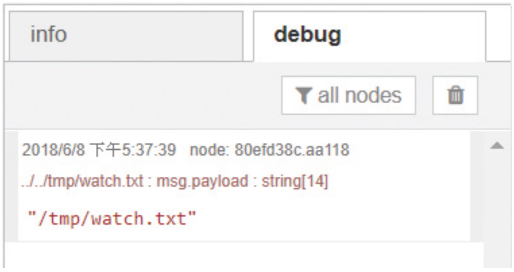
URL should use ws:// or wss:// scheme and point to an existing websocket listener.

By default, payload will contain the data to be sent over, or received from a websocket. The client can be configured to send or receive the entire message object as a JSON formatted string.

Add new websocket-listener config (Connect to)		
Item	**Option**	**Description**
URL		The target websocket URL, "**ws://[IP.addr]/path**" for unsecured connections and "**wss://[IP.addr]/path**" for secured connections.
Send/ Receive	**Payload**	Will only send/receive msg.payload.
	Entire message	Will send the entire msg labeled message.

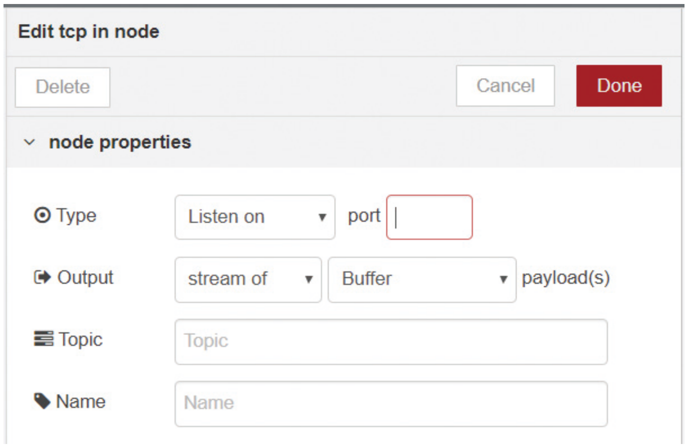
3.1.8 Watch

The watch node monitors the changes of a directory or a file. Multiple targets are allowed by using a list of comma separated directories and/or files. Putting quotes "…" around any that have spaces is required.

Example: Continue to monitor the changes of a file called /tmp/watch.txt.		
Step	**Description**	**Screenshot**
1	Add and connect a **watch** node and a **debug** node to the workspace as shown.	
2	Input the target file name with full path for monitoring.	 Edit watch node Delete Cancel Done node properties File(s) tmp/watch ☐ Watch sub-directories recursively Name Name On Windows you must use double back-slashes \\ in any directory names.
3	Deploy the flow and switch to the CLI mode. Execute the command **touch** on the target file, then the result will be shown in the debug tab.	`root@NT0010F33FAECF:/tmp# touch watch.txt` `root@NT0010F33FAECF:/tmp# _`  2018/6/8 下午5:37:39 node: 80efd38c.aa118 ../../tmp/watch.txt : msg.payload : string[14] "/tmp/watch.txt"

3.1.9 tcp

The tcp input node is used to accept incoming TCP requests on a specified port or to connect to a remote TCP port.



Edit tcp in node

node properties

Type: Listen on port

Output: stream of Buffer payload(s)

Topic: Topic

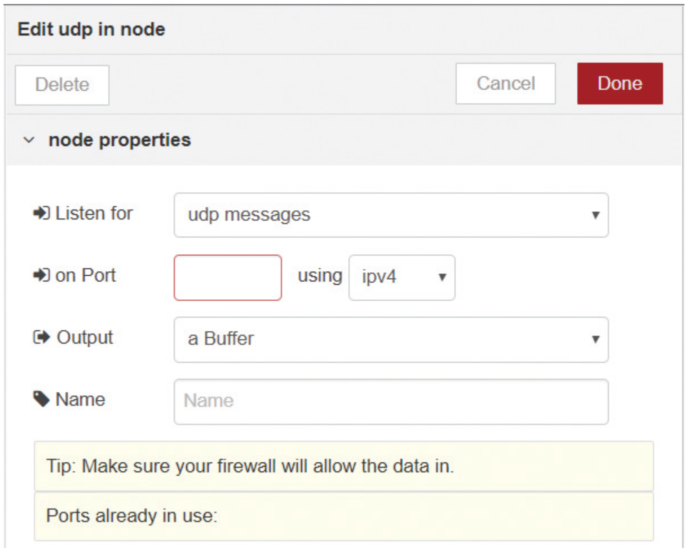
Name: Name

Item	Option	Description
Type	Listen on	The heart of publish/subscribe protocol.
	Connect to	
	Port	The service port number.
Output	Stream of	The consecutive data structure/single output.
	Single	
	Buffer	The custom data structure/the string/compressed data string.
	String	
	Based64 String	

The example below shows you how to send TCP requests using the tcp node. In this case you will need to make an HTTP request that follows the specifications in (http://tools.ietf.org/html/rfc2616#section-5.1.2).

3.1.10 udp

The udp input node is used to accept incoming UDP packets (or multicast packets) on a specified port.



Edit udp in node

Listen for: udp messages

on Port: using ipv4

Output: a Buffer

Name: Name

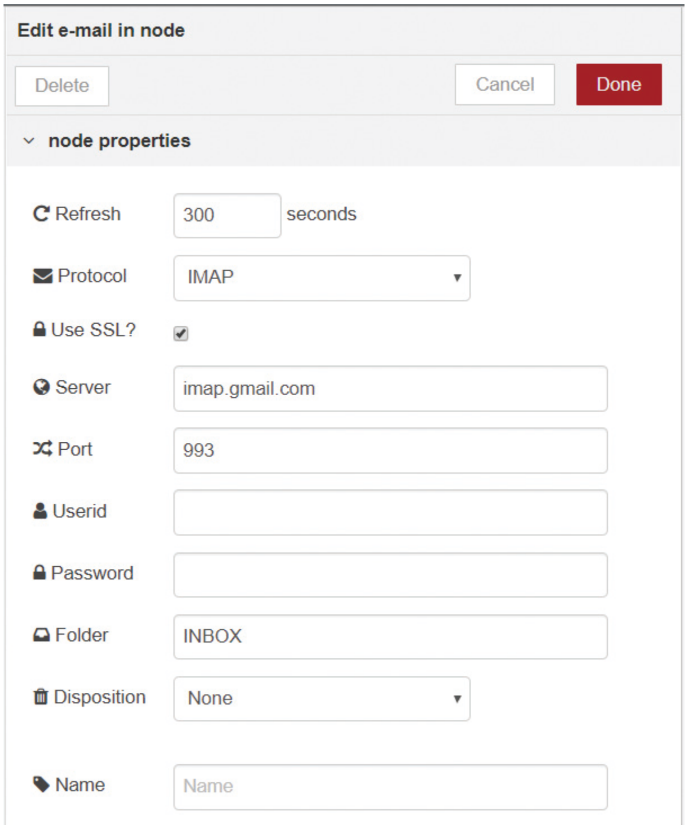
Tip: Make sure your firewall will allow the data in.

Ports already in use:

Item	Option	Description
Listen for	Udp messages	The heart of publish/subscribe protocol.
	multicast messages	
	Port	The service port number.
	Using ipv4/ipv6	The type of protocol this communication is using.
Output	Buffer	The custom data structure/the string/ compressed data string.
	String	
	Based64 String	
Name	The name of the node.	

3.1.11 email in

The email in node retrieves emails from an email server. For security concerns, SSL over IMAP on port 993 is enabled by default.



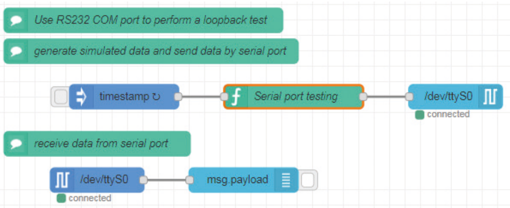
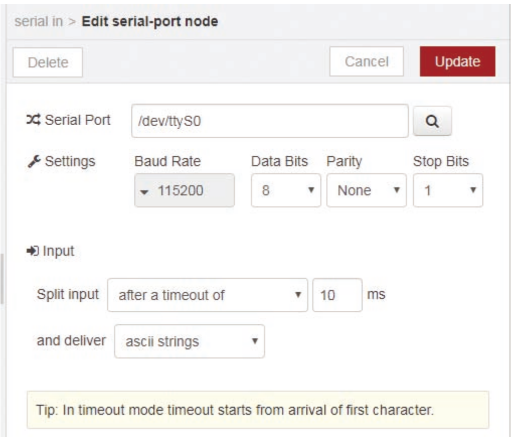
Edit e-mail in node

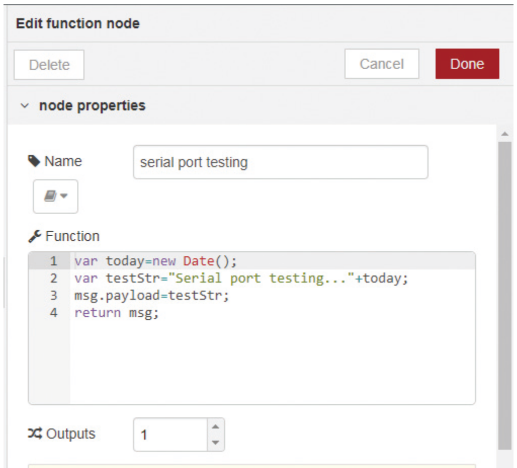
Refresh	300 seconds
Protocol	IMAP
Use SSL?	☑
Server	imap.gmail.com
Port	993
Userid	
Password	
Folder	INBOX
Disposition	None
Name	Name

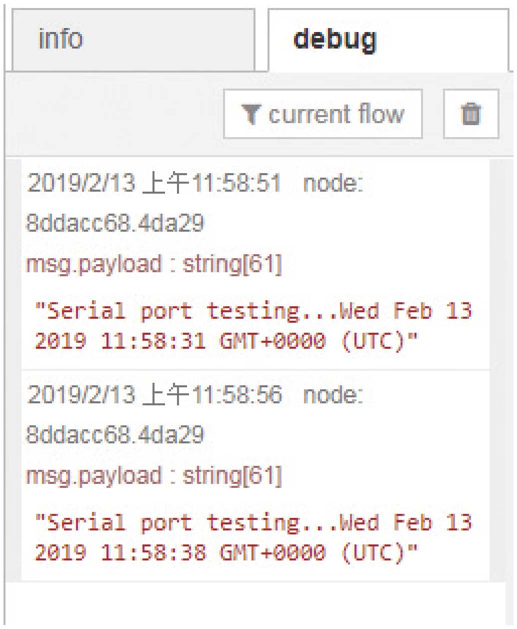
3.1.12 serial

The serial input node reads from a serial port on the local device. It can wait for a "split" character (default \n). It also accepts hex notation (0x0a), wait for a timeout in milliseconds for the first character received, or wait to fill a fixed sized buffer.

Next, it outputs **msg.payload** as either a UTF8 ASCII string or a binary Buffer object. If no split character is specified, or a timeout or buffer size of 0, then a stream of single characters is sent, again either as ASCII characters or size 1 binary buffers.

Example: Use RS232 COM port to perform a loopback test.		
Step	**Description**	**Screenshot**
1	Add and connect 1 **inject** node, 1 **serial** input node, 1 **serial** output node, 1 **function** node and 1 **debug** node to the workspace.	 Use RS232 COM port to perform a loopback test generate simulated data and send data by serial port timestamp ↻ Serial port testing /dev/ttyS0 connected receive data from serial port /dev/ttyS0 connected msg.payload
2	Configure the **serial input** and **output** node as shown.	 serial in > Edit serial-port node Delete Cancel Update Serial Port: /dev/ttyS0 Settings: Baud Rate 115200; Data Bits 8; Parity None; Stop Bits 1 Input Split input: after a timeout of 10 ms and deliver: ascii strings Tip: In timeout mode timeout starts from arrival of first character.

3	Edit the **function** node as shown.	 Edit function node
		Delete
		Cancel
		Done
		node properties
		Name: serial port testing
		Function
		`var today=new Date();`
		`var testStr="Serial port testing..."+today;`
		`msg.payload=testStr;`
		`return msg;`
		Outputs: 1

Example: Use RS232 COM port to perform a loopback test.		
Step	**Description**	**Screenshot**
4	Deploy your flow and any message received will show up in the debug tab.	 2019/2/13 上午11:58:51 node: 8ddacc68.4da29 msg.payload : string[61] "Serial port testing...Wed Feb 13 2019 11:58:31 GMT+0000 (UTC)" 2019/2/13 上午11:58:56 node: 8ddacc68.4da29 msg.payload : string[61] "Serial port testing...Wed Feb 13 2019 11:58:38 GMT+0000 (UTC)"

3.2 Output Nodes

Output nodes disclose information from services or debug messages.

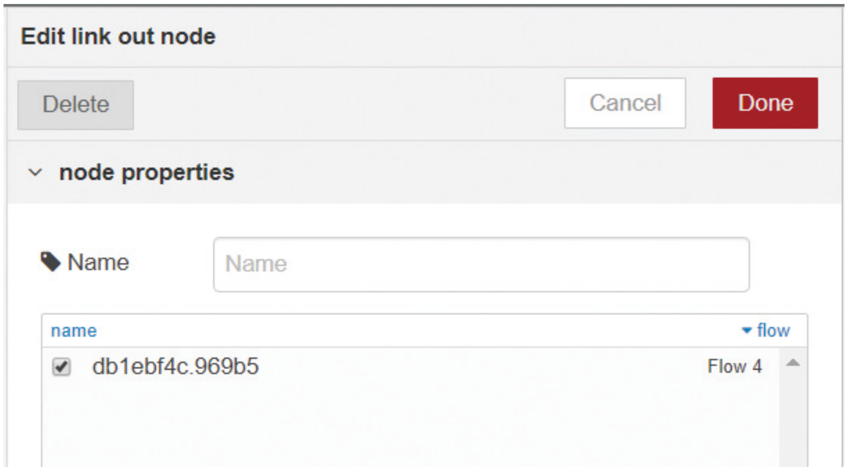
3.2.1 debug

The debug node can connect to the output of any node displaying any message property in the debug tab of the sidebar. The default is to display msg.payload. Each message will also display the timestamp, msg.topic, and the property chosen to output. Access the sidebar under the options drop-down menu on the top right corner. The button to the right of the node will toggle its output on and off so you can de-clutter the debug window. If the payload is an object or buffer, it will be stringified first for display and indicate that by saying "(Object)" or "(Buffer)". Selecting any particular message will highlight (in red) the debug node that reported it, which is useful if you wire up multiple debug nodes. Other than optionally showing the complete msg object, any calls to node.warn or node.error will appear here in the debug node.

3.2.2 link

The link output node can be connected to any link in node that exists on any tab. Once connected, they behave as if they were wired together. The wires between link nodes are only displayed when a link node is selected. If there are any wires to other tabs, a virtual node is shown, which can be clicked on to jump to the appropriate tab. Links cannot be created going into, or out of, a subflow.

Click on the node to select which link out node it will connect to. Check the checkbox of the node to connect to, and click **Done** when finished.



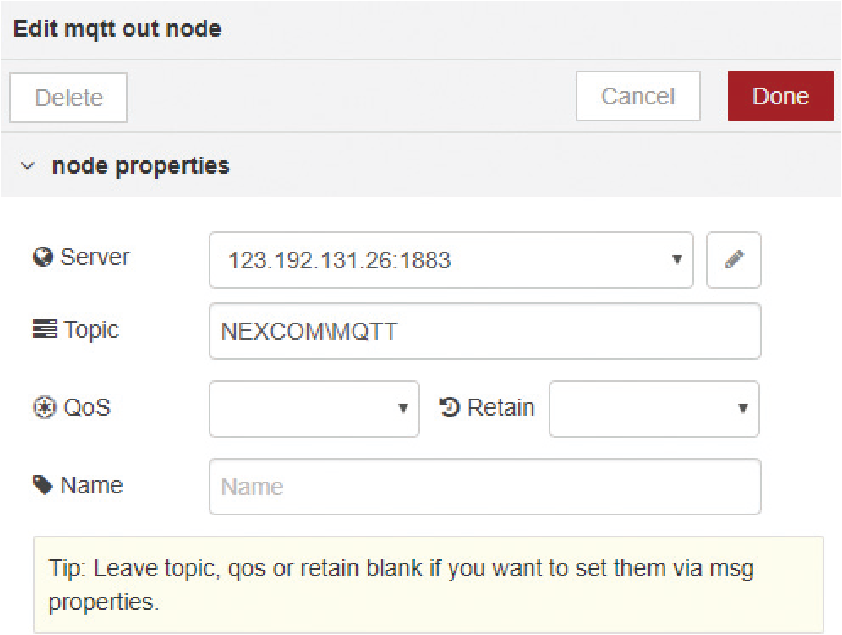
Edit link out node

Name: Name

name	flow
☑ db1ebf4c.969b5	Flow 4

3.2.3 mqtt

The mqtt output node connects to a MQTT broker and publishes msg.payload either to the msg.topic or to the topic specified in the edit window. The value in the edit window has precedence. Likewise QoS and/or retain values in the edit panel will overwrite msg.qos and msg.retain properties. If nothing is set, the default value is 0 and false respectively. If msg.payload contains an object, it will be converted into a string before being sent.



Server 123.192.131.26:1883

Topic NEXCOM\MQTT

QoS

Retain

Name Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

Item	Option	Description
Server	The MQTT broker that is currently used.	
Topic	The string used by the broker to filter messages. A topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator).	
QoS	The Quality of Service (QoS) level is an agreement between sender and receiver of a message regarding the guarantees of delivering a message.	
	0 (default)	The message is delivered at most once, or it is not delivered at all. Its delivery across the network is not acknowledged. The message is not stored. The message might be lost if the client is disconnected, or if the MQTT broker fails.

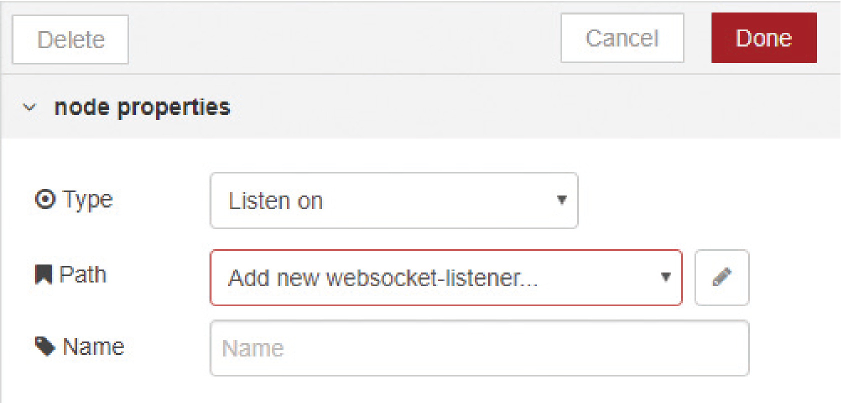
Item	Option	Description
QoS	**1**	The message is always delivered at least once. If the sender does not receive an acknowledgement, the message is sent again with the DUP flag set until an acknowledgement is received. As a result, the receiver can be sent the same message multiple times, and might process it multiple times.
	2	The message is always delivered exactly once. The message must be stored locally at the sender and receiver until it is processed.
Retain	A retained message is a normal MQTT message with the retained flag set to true. The broker will store the last retained message and the corresponding QoS for that topic. Each client that subscribes to a topic pattern, which matches the topic of the retained message, will receive the message immediately after subscribing. The broker will store one retained message only for each topic.	
Name	The name of the node.	

3.2.4 http response

The http response output node sends responses back to http requests received from an http input node.

3.2.5 websocket

By default, msg.payload will be sent over the websocket. The socket can be configured to encode the entire msg object as a JSON string and send that over the websocket. If the message arriving at this node started at a websocket input node, the message will be sent back to the client that triggered the flow. Otherwise, the message will be broadcasted to all connected clients. If you want to broadcast a message that started at a websocket input node, you should delete the msg._session property within the flow.

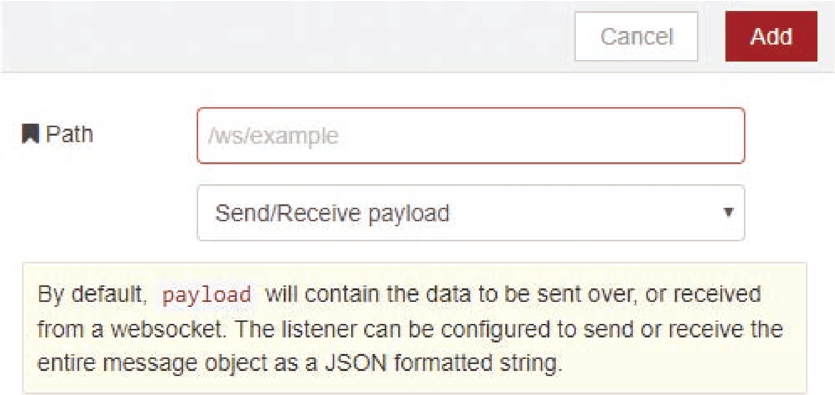


Type: Listen on

Path: Add new websocket-listener...

Name: Name

Item	Option	Description
Type	**Listen on**	Creates a path and awaits a remote device to connect.
	Connect to	Connect to target websocket.
Path/URL	**Path**	Only available on type "Listen on".
	URL	Only available on type "Connect to".
Name	The name of the node.	

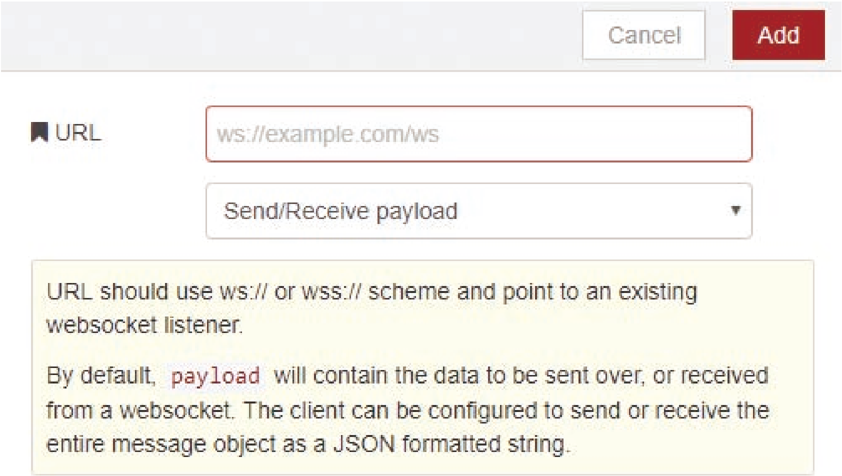


Path /ws/example

Send/Receive payload

By default, `payload` will contain the data to be sent over, or received from a websocket. The listener can be configured to send or receive the entire message object as a JSON formatted string.

Item	Option	Description
Path		The path of websocket.
Send/ Receive	**Payload**	Will only send/receive msg.payload.
	Entire message	Will send the entire msg labeled message.



URL ws://example.com/ws

Send/Receive payload

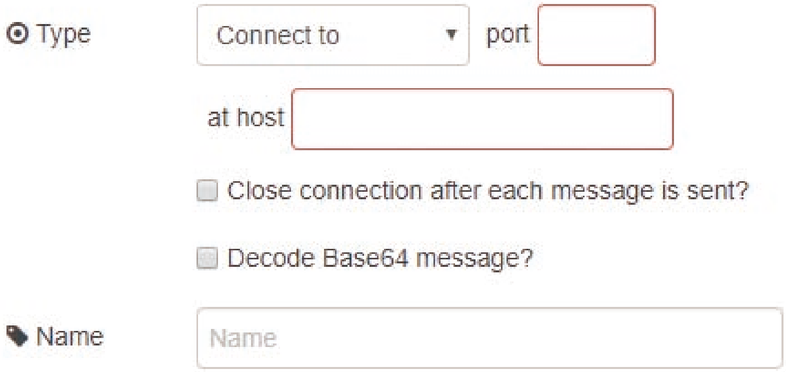
URL should use ws:// or wss:// scheme and point to an existing websocket listener.

By default, `payload` will contain the data to be sent over, or received from a websocket. The client can be configured to send or receive the entire message object as a JSON formatted string.

Item	Option	Description
URL		The target websocket URL, "**ws://[IP.addr]/path**" for unsecured connections and "**wss://[IP.addr]/path**" for secured connections.
Send/ Receive	**Payload**	Will only send/receive msg.payload.
	Entire message	Will send the entire msg labeled message.

3.2.6 tcp

The tcp output node provides a choice of TCP outputs to connect to a remote TCP port, accept incoming connections, or reply to message received from the tcp input node.



⊙ Type: Connect to ▾ port

at host

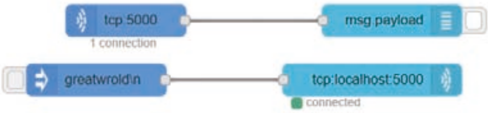
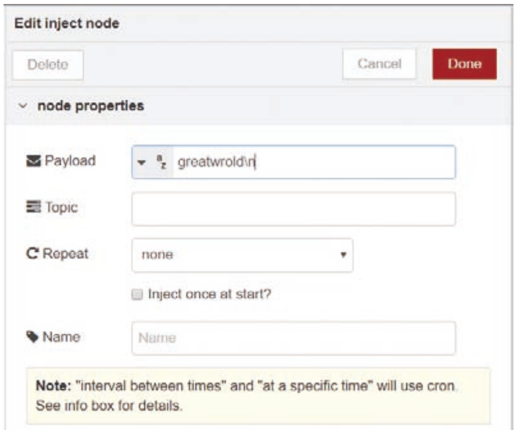
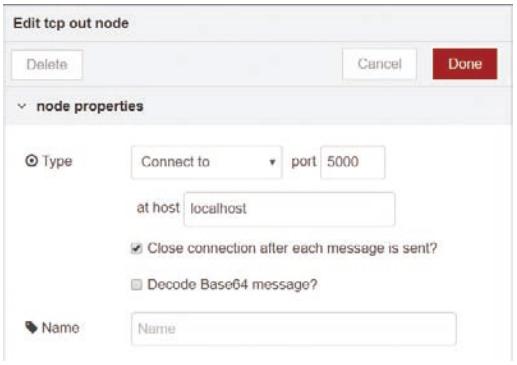
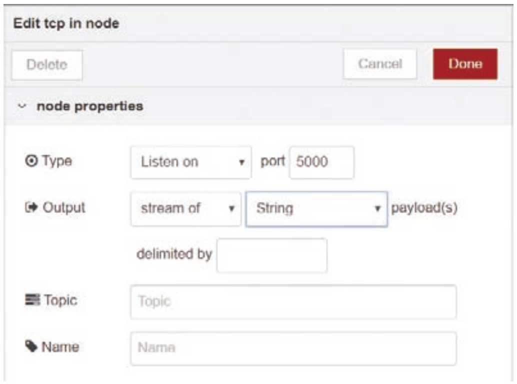
□ Close connection after each message is sent?

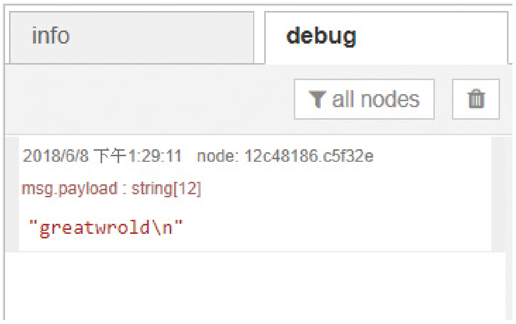
□ Decode Base64 message?

Name: Name

Item	Option	Description
Type	Listen on	Creates a port and awaits others' connection.
	Connect to	Send the message to target host and port.
	Reply to TCP	Reply message to the input TCP.
Port	Send message through target port, available on Type Listen on/Connect to.	
Host	Send message to target host, available on Type Connect to.	
Close connection after each message is sent	Close connection with the target after the message is sent. Message may be stacked if remain unchecked.	
Decode Base64 message	Decode Base64 message and sent decrypted message.	
Name	The name of the node.	

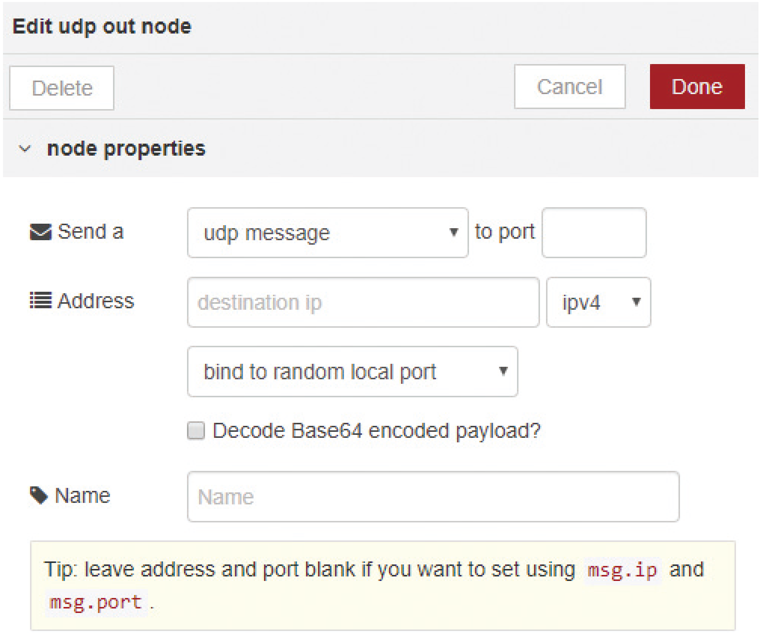
The example below shows you how to send TCP requests using the tcp node. In this case, users can establish a TCP connection to the target unit/port for sending out the data and the structure of the data as well as not expecting the response data is sent back by the target unit.

Example: Send TCP requests.		
Step	**Description**	**Screenshot**
1	Add and connect 1 **inject** node, 1 **tcp** output node, 1 **tcp** input node, and 1 **debug** node as shown.	 tcp:5000 1 connection msg.payload greatwrold\n tcp:localhost:5000 connected
2	Edit the **inject** node to add a string "**greatwrold\n**". Edit the **tcp** output node to set **Type** to **connect to**, **port** to **5000**, and **at host** to **localhost**. Check the checkbox before **Close connection after each message is sent?**	Edit inject node Delete Cancel Done node properties Payload: greatwrold\n Topic: Repeat: none □ Inject once at start? Name: Name **Note:** "interval between times" and "at a specific time" will use cron. See info box for details. Edit tcp out node Delete Cancel Done node properties Type: Connect to port 5000 at host localhost ☑ Close connection after each message is sent? □ Decode Base64 message? Name: Name
3	Edit the **tcp** input node and select **Listen on** and set the **port** to **5000**. Set **Output** to **stream of String** and click **Ok**.	Edit tcp in node Delete Cancel Done node properties Type: Listen on port 5000 Output: stream of String payload(s) delimited by Topic: Topic Name: Name

Example: Send TCP requests.		
Step	**Description**	**Screenshot**
4	Deploy your flow and any message received will show up in the debug tab.	 2018/6/8 下午1:29:11 node: 12c48186.c5f32e msg.payload : string[12] "greatwrold\n"

3.2.7 udp

The udp output node sends msg.payload to the designated UDP host and port and supports multicast. You may also use msg.ip and msg. port to set the destination values, but the statically configured values have precedence. Set the address to the local broadcast IP address for broadcast.



Send a: udp message, to port: (blank)

Address: destination ip, ipv4

bind to random local port

□ Decode Base64 encoded payload?

Name: Name

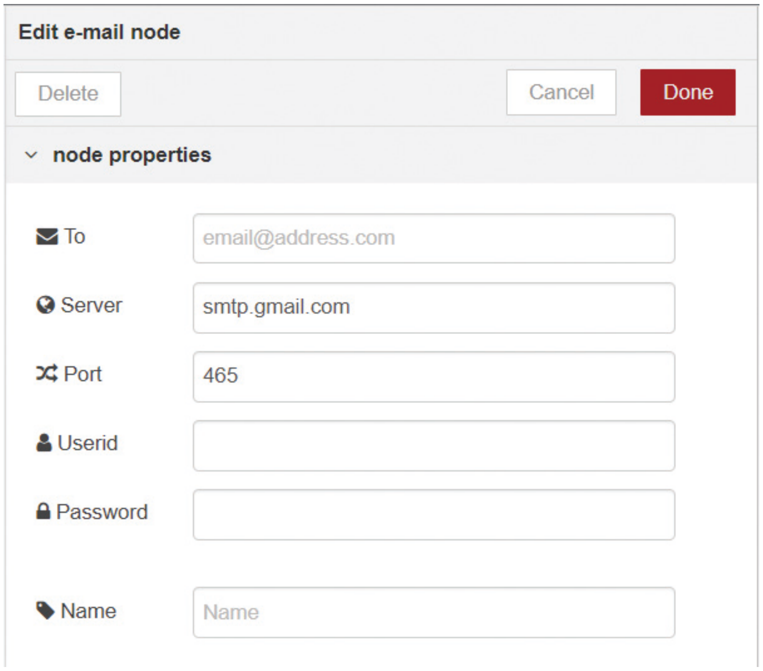
Tip: leave address and port blank if you want to set using `msg.ip` and `msg.port`.

Item	Option	Description
Message Type	**udp message**	Send payload to target IP.
	Broadcast message	Send payload to broadcast IP.
	Multicast message	Send payload message to a multicast IP.
Port	The service protocol.	
Address	The destination IP address.	
Ipv4/Ipv6	The protocol this communication is using.	
Outport	The port which is going to publish message. Only available when bind to local port is true.	
Decode Base64 payload?	If the payload is encoded to Base64, set this as true to send decoded message.	
Name	The name of the node.	

Note: On some systems, you may need to be root to use ports below 1024 and/or broadcast.

3.2.8 email out

The email out node sends emails through an email server. For security concerns, SSL over SMTP on port 465 is enabled by default.



Edit e-mail node

To: email@address.com

Server: smtp.gmail.com

Port: 465

Userid:

Password:

Name: Name

3.2.9 serial

The serial output port provides a connection to an outbound serial port. Only the **msg.payload** is sent. The new line character used to split the input can be appended to every message sent out to the serial port optionally.

3.3 Function Nodes

Function nodes manipulate data on demands.

3.3.1 exec

The exec node calls out to a system command and gets a callback on completion, returning the complete result in one message, along with any errors. The optional append gets added to the command after msg.payload, so you can do things like pipe the result to another command. Parameters with spaces should be enclosed in quotes.

3.3.2 function

The function node is a versatile utility that you can use when there is no existing node dedicated to the task. It is great for doing specialized data processing or formatting. As the name implies, a function node exposes a single JavaScript function. Using the function node, your JavaScript code can run against the messages passed in the returns zero or more messages to downstream nodes for further processing.

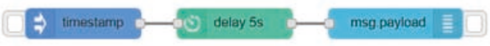
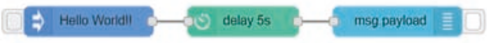
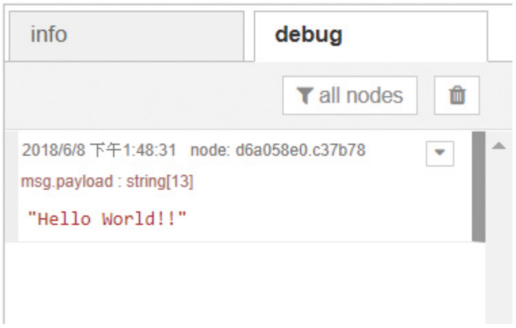
3.3.3 template

The template node is able to create a new message based on the provided template.

Format	**Mustache template**	Mustache is a simple web template system, described as a “logic-less” system because it lacks any explicit control flow statements, like “if and else” conditions or “for loops”; however, both looping and conditional evaluation can be achieved using section tags processing lists and lambdas. Furthermore, it is named “Mustache” because of heavy use of curly braces, { }, that resemble a sideways mustache.
	Plain text	The plain text shows the raw content in char.

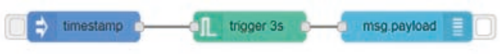
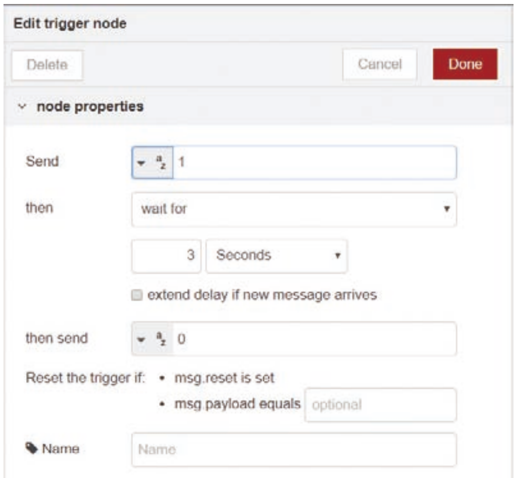
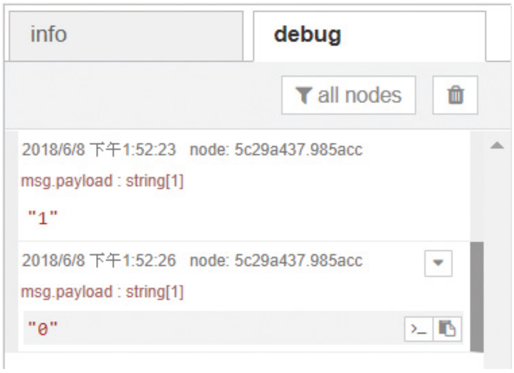
3.3.4 delay

The delay node introduces a delay into a flow or rate limits messages. Default delay time is 5 seconds and rate limit of 1 msg/second, but both can be configured.

<table>
<tr><th colspan="3">Example: Write a message "<i>Hello World!!</i>" and make it to delay for 5 seconds.</th></tr>
<tr><th>Step</th><th>Description</th><th>Screenshot</th></tr>
<tr><td>1</td><td>Add and connect 1 <b>inject</b> node, 1 <b>delay</b> node, and 1 <b>debug</b> nodes to the workspace as shown.</td><td> </td></tr>
<tr><td>2</td><td>Edit the <b>inject</b> node to set the payload to <b>string</b> "<b>Hello World!!</b>" and click <b>Ok</b>.</td><td> </td></tr>
<tr><td>3</td><td>Deploy the flow and click the button on the left of the inject node, and the message will be displayed after 5 seconds in the debug tab.</td><td>
2018/6/8 下午1:48:31 node: d6a058e0.c37b78
msg.payload : string[13]
"Hello World!!"</td></tr>
</table>

3.3.5 trigger

The trigger node creates two payloads on the output separated by a timeout whenever any message arrives on the input. The two output states can be specified as the duration of the timer, and either one can be set to a value or a template from the inbound message using the mustache syntax, "the payload is {{payload}}".

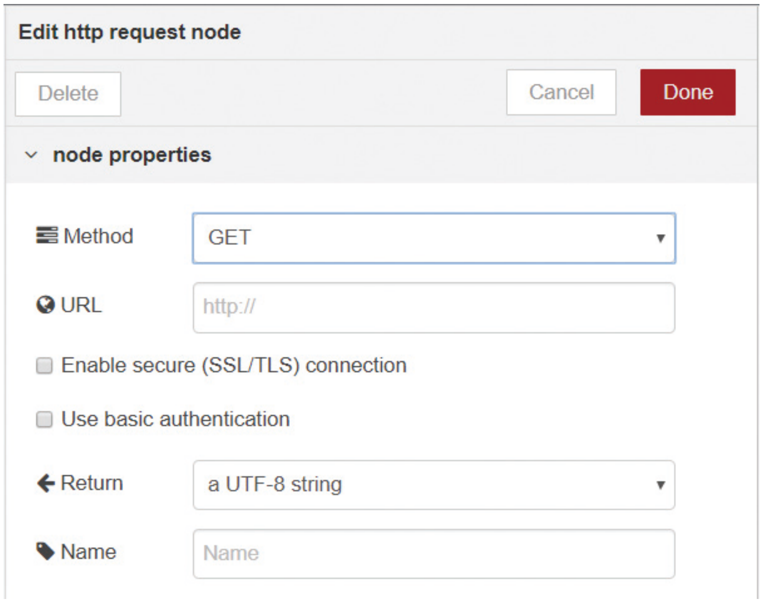
Example: Set the original value to 1 and change the value to 0 in 3 seconds.		
Step	**Description**	**Screenshot**
1	Connect 1 **inject** node, 1 **trigger** node, and 1 **debug** node to the workspace as shown.	 timestamp trigger 3s msg.payload
2	Double click the **trigger** node and set the time to 3 **Seconds**.	 Edit trigger node Delete Cancel Done node properties Send 1 then wait for 3 Seconds □ extend delay if new message arrives then send 0 Reset the trigger if: • msg.reset is set • msg.payload equals optional Name Name
3	Deploy the flow and click the button on the left of the inject node, and two messages will show up in the debug tab. Value 1 will show up followed by value 0 after 3 seconds.	 info debug all nodes 2018/6/8 下午1:52:23 node: 5c29a437.985acc msg.payload : string[1] "1" 2018/6/8 下午1:52:26 node: 5c29a437.985acc msg.payload : string[1] "0"

3.3.6 comment

The comment adds simple description or documentation about nodes or flow. Anything you write in the **Body** will be rendered in the info tab.

3.3.7 http request

The http request provides a node for making http request.



Edit http request node

node properties

Method: GET

URL: http://

☐ Enable secure (SSL/TLS) connection

☐ Use basic authentication

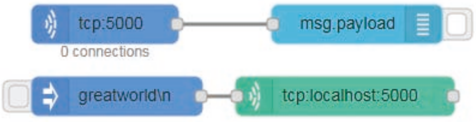
Return: a UTF-8 string

Name: Name

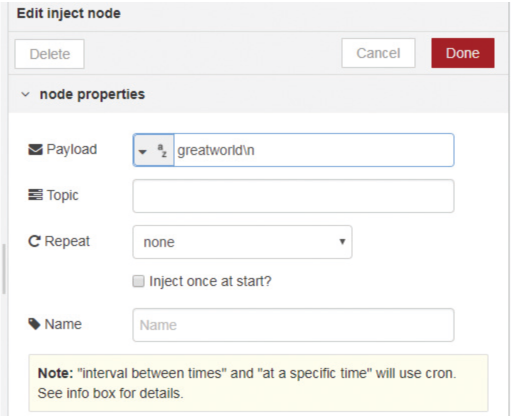
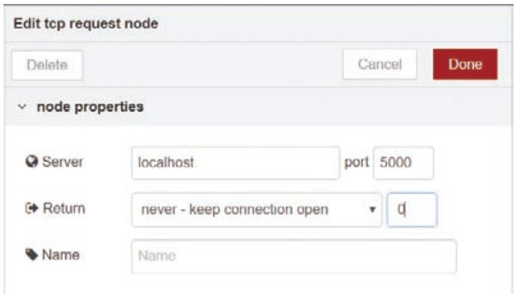
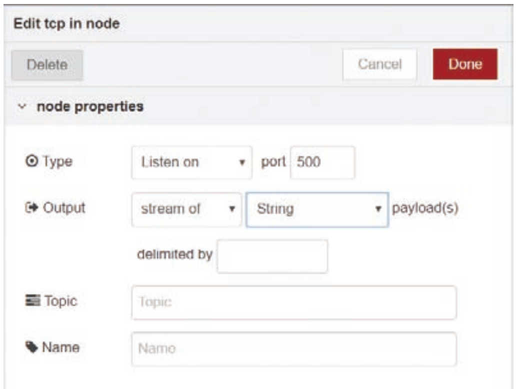
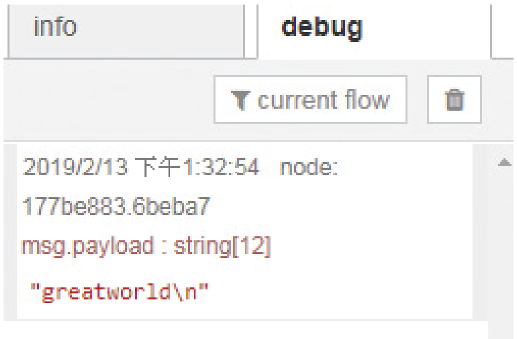
3.3.8 tcp request

The **tcp request** node sends the msg.payload to a server tcp port and expects a response. Connects, sends the "request", and reads the "response". It can either count a number of returned characters into a fixed buffer, match a specified character before returning, wait a fixed timeout from first reply and then return, or just sit and wait for data.

The example below shows you how to send TCP requests using the tcp node. In this case, users can establish a TCP connection to target unit/port for sending out the data and the structure of data as well as expecting the response data is sent back by the target unit.

Example: Send TCP requests and expects a response.		
Step	**Description**	**Screenshot**
1	Add and connect 1 **inject** node, 1 **tcp request** node, and 1 **tcp** input node as shown.	 tcp:5000 0 connections msg.payload greatworld\n tcp:localhost:5000

Example: Send TCP requests and expects a response.

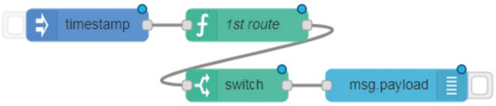
Step	Description	Screenshot
2	Edit the inject node to add the string "**greatworld\n**". Edit the **tcp request** node to connect to **localhost** at port **5000** and choose **never – keep connection open**.	 Edit inject node Delete Cancel Done node properties Payload greatworld\n Topic Repeat none ☐ Inject once at start? Name Name **Note:** "interval between times" and "at a specific time" will use cron. See info box for details.  Edit tcp request node Delete Cancel Done node properties Server localhost port 5000 Return never - keep connection open q Name Name
3	Edit the **tcp input** node to choose **Listen on** at port **5000** and **stream of** with **String**.	 Edit tcp in node Delete Cancel Done node properties Type Listen on port 500 Output stream of String payload(s) delimited by Topic Topic Name Name
4	Deploy your flow and any message received will show up in the debug tab.	 2019/2/13 下午1:32:54 node: 177be883.6beba7 msg.payload : string[12] "greatworld\n"

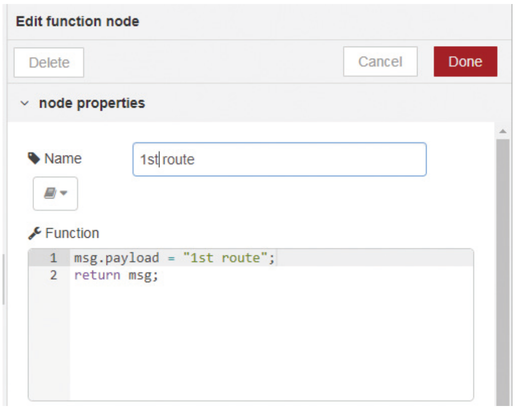
3.3.9 switch

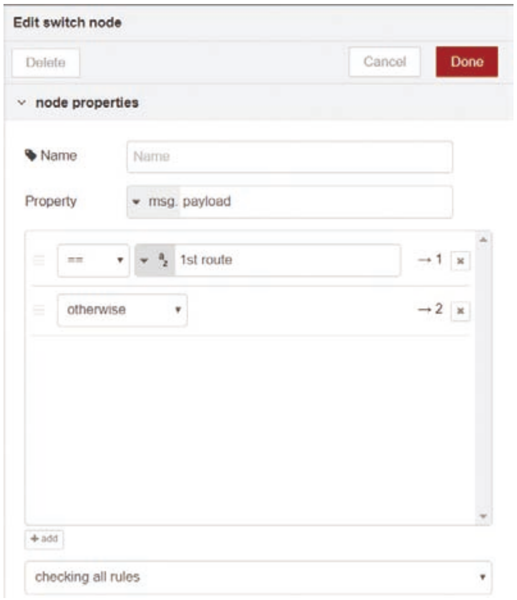
The switch routes messages based on their properties. When a message arrives, the selected property is evaluated against each of the pre-defined rules. The message is then sent to the output of all rules that pass.

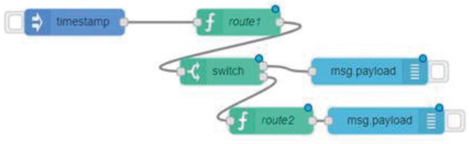
Note: The otherwise rule applies as a "not any of" the rules preceding it.

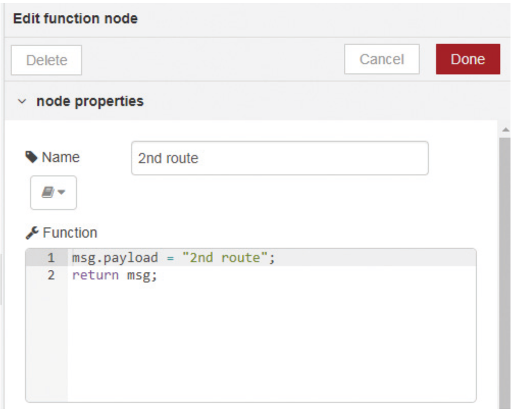
Example: Make a switch determine if the message received matches with the rule. If it does, go first route, otherwise go second route.

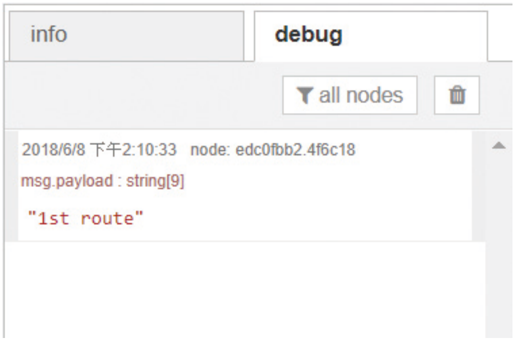
Step	Description	Screenshot
1	Add and connect 1 **inject** node, 1 **function** node, 1 **switch** node, and 1 **debug** node to the workspace as shown.	

2	Edit the **function** node as shown and click **Ok**.	Edit function node
		Delete Cancel Done
		node properties
		Name: 1st route
		Function
		`msg.payload = "1st route";`
		`return msg;`

3	Edit the **switch** node as shown, and click **Ok**.	Edit switch node Delete Cancel Done node properties Name: Name Property: msg. payload == 1st route →1 otherwise →2 +add checking all rules

Example: Make a switch determine if the message received matches with the rule. If it does, go first route, otherwise go second route.		
Step	**Description**	**Screenshot**
	Note: If the msg.payload is equal to "1st route", go first route; otherwise, go second route. Notice here, after you click **Ok**, the switch node now has two "out" ports for you to connect two routes.	
4	Add a **function** node and a **debug** node to the workspace and connect them as shown.	 timestamp route1 switch msg.payload route2 msg.payload

5	Edit the second **function** node as shown and click **Ok**.	 Edit function node
		Delete
		Cancel
		Done
		node properties
		Function
		Name: 2nd route
		`msg.payload = "2nd route";`
		`return msg;`

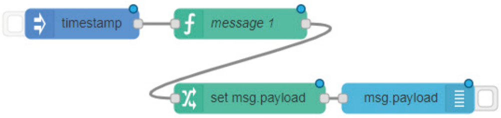
6	Deploy the flow and click the button on the left of the inject node, and the result will be shown in the debug tab.	 2018/6/8 下午2:10:33 node: edc0fbb2.4f6c18 msg.payload : string[9] "1st route"

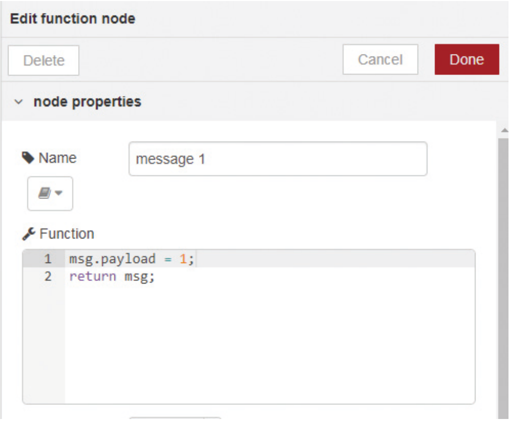
3.3.10 change

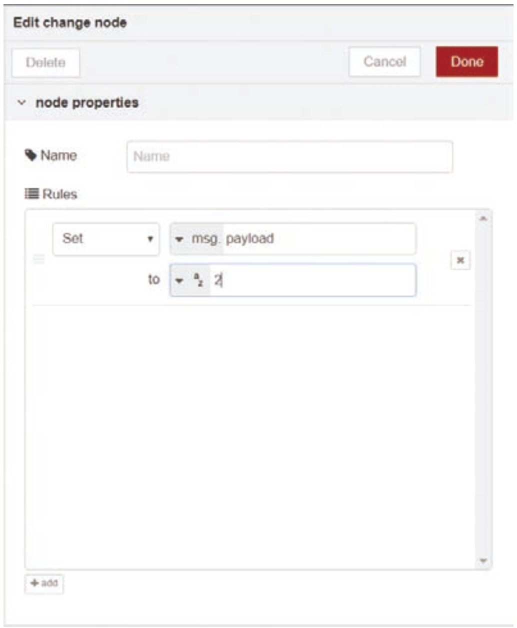
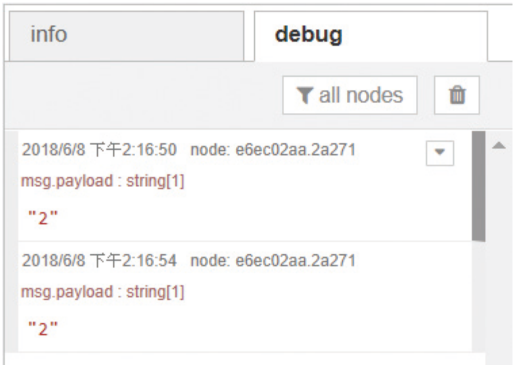
The change node sets, changes, or deletes properties of a message. It can specify multiple rules that apply to the message in turn.

Rule	**Set**	Set a property. The target property can either be a string value or reference another message property by name, for example: **msg.topic**.
	Change	Search and replace parts of the property. If regular expressions are enabled, the replace with property can include capture groups, for example $1.
	Delete	Delete a property.
	Move	Move or rename a property.

Example: Change a value message from 1 to 2.

Step	Description	Screenshot
1	Add and connect 1 **inject** node, 1 **function** node, 1 **change** node, and 1 **debug** node to the workspace as shown.	

2	Edit the **function** node as shown and click **Ok**.	 Edit function node
		Delete
		Cancel
		Done
		node properties
		Name: message 1
		Function
		`msg.payload = 1;`
		`return msg;`

Example: Change a value message from 1 to 2.		
Step	**Description**	**Screenshot**
3	Edit the **change** node as shown and click **Ok**.	 Edit change node Delete Cancel Done node properties Name Rules add Name: Name Set msg. payload to 2
4	Deploy the flow and click the button on the left of the inject node, and the result will be shown in the debug tab.	 2018/6/8 下午2:16:50 node: e6ec02aa.2a271 msg.payload : string[1] "2" 2018/6/8 下午2:16:54 node: e6ec02aa.2a271 msg.payload : string[1] "2"

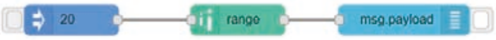
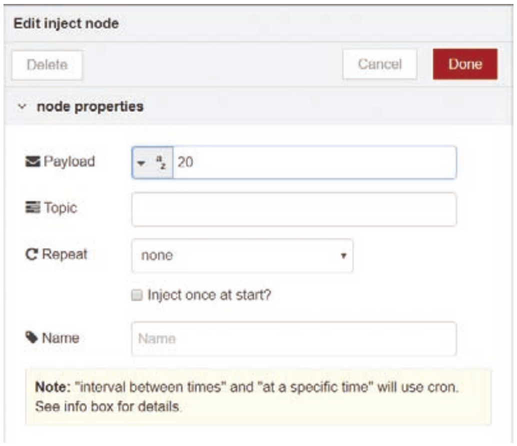
3.3.11 range

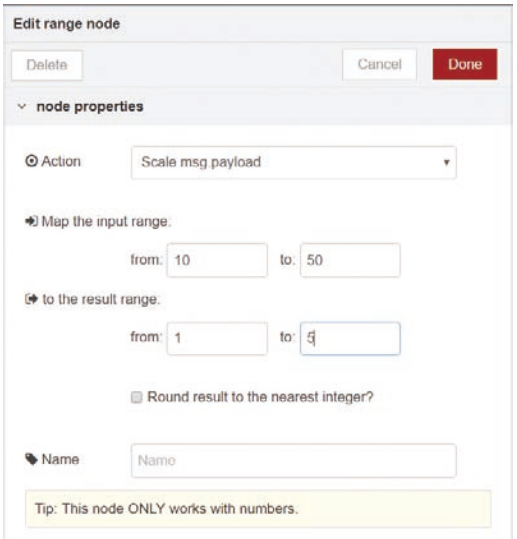
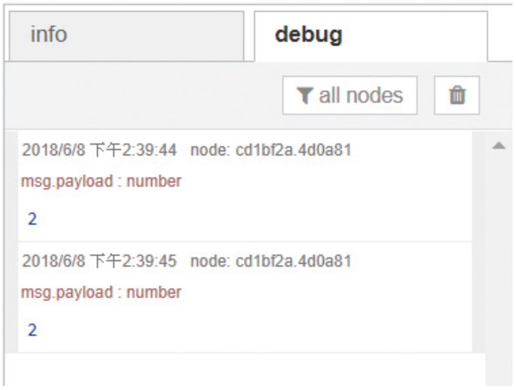
The range node maps numeric input values to another scale linearly.

Note: This only operates on numbers. Anything else will be converted into a number and rejected if fails.

The scale node has three options set by the action field:

Action	**1. "scale msg.payload"**	The result might be outside the given ranges scale according to the mapping given.
	2. Scale and limit to target range	The result will never be outside the range specified within the result range.
	3. Scale and wrap within the target range	The result will essentially be a "modulo-style" wrap-around within the result range.

Example: Map value 20 in the range of 10 to 50 to another value in the scale of 1 to 5.		
Step	**Description**	**Screenshot**
1	Add and connect 1 **inject** node, 1 **range** node, and 1 **debug** node to the workspace as shown.	
2	Edit the **inject** node as shown and click **Ok**.	Edit inject node node properties Payload: 20 Topic: Repeat: none □ Inject once at start? Name: Name Note: "interval between times" and "at a specific time" will use cron. See info box for details.

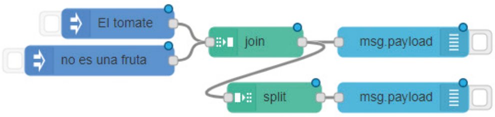
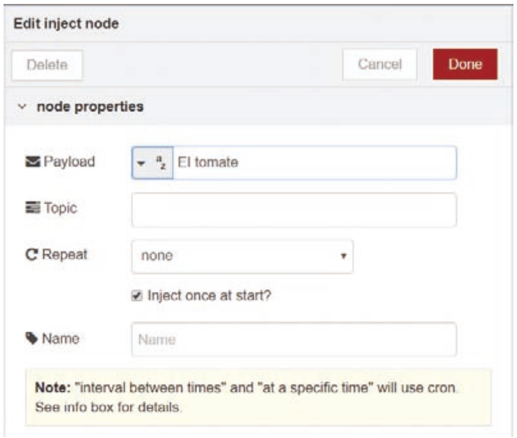
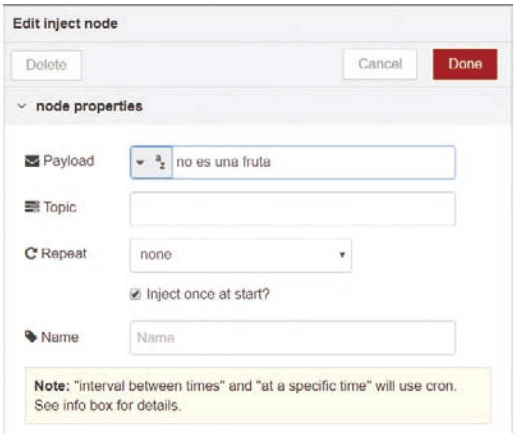
Example: Map value 20 in the range of 10 to 50 to another value in the scale of 1 to 5.		
Step	**Description**	**Screenshot**
3	Edit the **range** node as shown and click **Ok**.	 Edit range node Delete Cancel Done node properties Action: Scale msg.payload Map the input range: from: 10 to: 50 to the result range: from: 1 to: 5 □ Round result to the nearest integer? Name: Name Tip: This node ONLY works with numbers.
4	Deploy the flow and click the button on the left of the inject node, and the result will be shown in the debug tab.	 2018/6/8 下午2:39:44 node: cd1bf2a.4d0a81 msg.payload : number 2 2018/6/8 下午2:39:45 node: cd1bf2a.4d0a81 msg.payload : number 2

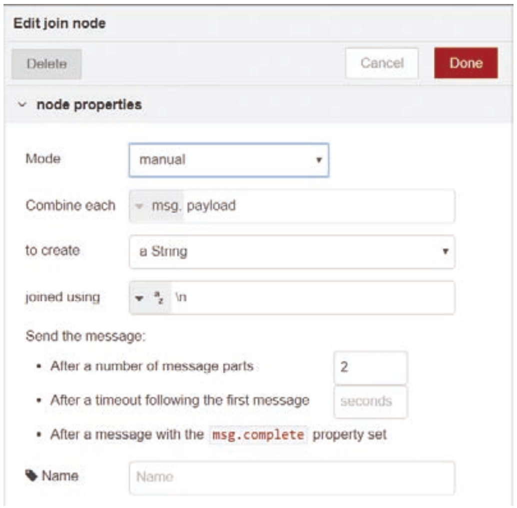
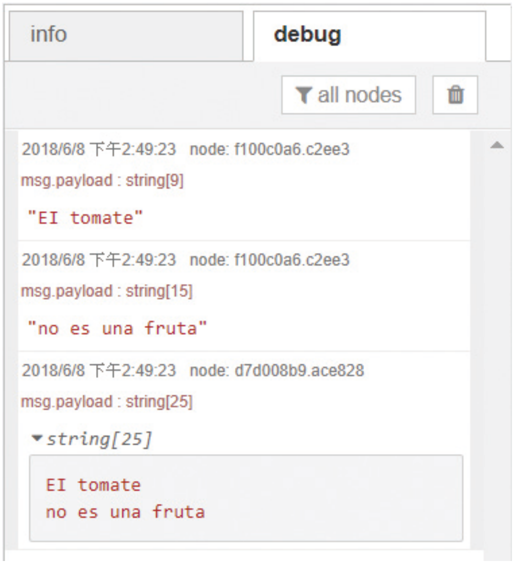
3.3.12 split

The split node splits an input into multiple outputs based on the provided configuration.

3.3.13 join

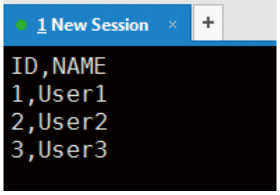
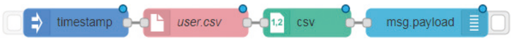
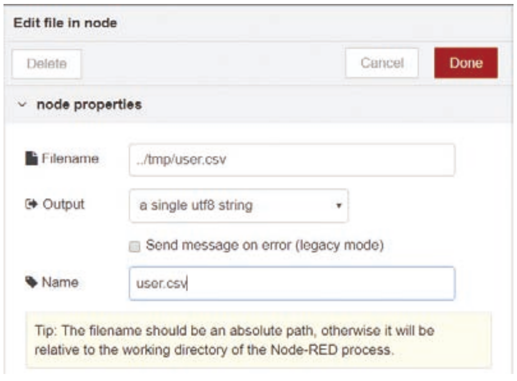
The join node joins a sequence of messages into a single message.

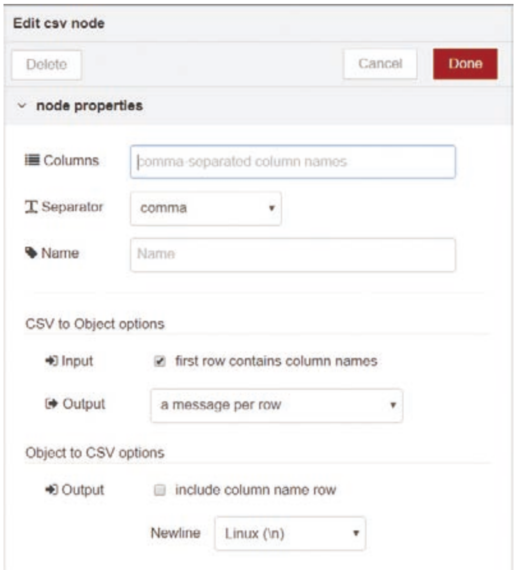
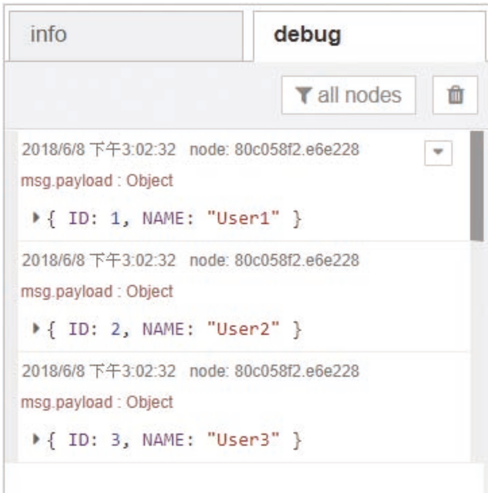
Example: Join a sequence of messages into a single message and split it into multiple outputs.		
Step	**Description**	**Screenshot**
1	Add and connect 2 **inject** nodes, 1 **join** node, 1 **split** node, and 2 **debug** nodes to the workspace as shown.	 El tomate no es una fruta join split msg.payload msg.payload
2	Edit the first **inject** node as shown and click **Done**.	Edit inject node Delete Cancel Done node properties Payload: El tomate Topic: Repeat: none ☑ Inject once at start? Name: Name **Note:** "interval between times" and "at a specific time" will use cron. See info box for details.
3	Edit the second **inject** node as shown and click **Done**.	Edit inject node Delete Cancel Done node properties Payload: no es una fruta Topic: Repeat: none ☑ Inject once at start? Name: Name **Note:** "interval between times" and "at a specific time" will use cron. See info box for details.

Example: Join a sequence of messages into a single message and split it into multiple outputs.		
Step	**Description**	**Screenshot**
4	Edit the **join** node and set **After a fixed number of messages:** to 2 as shown and click **Done**.	 Edit join node Delete Cancel Done node properties Mode: manual Combine each: msg. payload to create: a String joined using: \n Send the message: • After a number of message parts: 2 • After a timeout following the first message: seconds • After a message with the `msg.complete` property set Name: Name
5	Deploy the flow and click the button on the left of the first inject node and the following second node. The result will be shown in the debug tab.	 2018/6/8 下午2:49:23 node: f100c0a6.c2ee3 msg.payload : string[9] "EI tomate" 2018/6/8 下午2:49:23 node: f100c0a6.c2ee3 msg.payload : string[15] "no es una fruta" 2018/6/8 下午2:49:23 node: d7d008b9.ace828 msg.payload : string[25] ▾string[25] EI tomate no es una fruta

3.3.14 csv

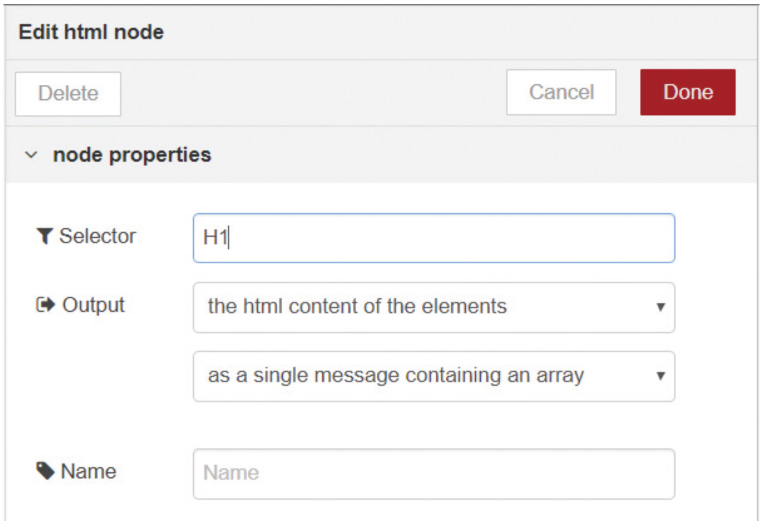
The csv node parses the msg.payload to convert csv to/from a JavaScript object and places the result in the payload. The source may be a string, a file, a buffer, or a readable stream. The columns template should contain an ordered list of column headers. For csv input, these become the property names. For csv output, these specify the properties to extract from the object and the order for the csv.

Example: Output the content of "user.csv" file as string messages.		
Step	**Description**	**Screenshot**
1	Create a **user.csv** file under the folder **../tmp/** on the device as shown.	 ID,NAME 1,User1 2,User2 3,User3
2	Connect 1 **inject** node, 1 **file in** node, 1 **csv** node, and 1 **debug** node to the workspace as shown.	
3	Edit the file in the node to assign the file named **user.csv** under **../tmp/**, and give the node the name **user.csv** then click **Ok**.	 Edit file in node Delete Cancel Done node properties Filename ../tmp/user.csv Output a single utf8 string □ Send message on error (legacy mode) Name user.csv Tip: The filename should be an absolute path, otherwise it will be relative to the working directory of the Node-RED process.

Example: Output the content of "user.csv" file as string messages.		
Step	**Description**	**Screenshot**
4	Edit the **csv** node as shown and click **Ok**.	 Edit csv node Delete Cancel Done node properties Columns: comma-separated column names Separator: comma Name: Name CSV to Object options Input ☑ first row contains column names Output: a message per row Object to CSV options Output □ include column name row Newline: Linux (\n)
5	Deploy the flow and click the button on the left of the inject node, and the result will be shown in the debug tab.	 2018/6/8 下午3:02:32 node: 80c058f2.e6e228 msg.payload : Object ▸{ ID: 1, NAME: "User1" } 2018/6/8 下午3:02:32 node: 80c058f2.e6e228 msg.payload : Object ▸{ ID: 2, NAME: "User2" } 2018/6/8 下午3:02:32 node: 80c058f2.e6e228 msg.payload : Object ▸{ ID: 3, NAME: "User3" }

3.3.15 html

The html node extracts elements from an html document held in msg.payload using a selector that uses Cheerio with the CSS selector syntax. The result can be either a single message with a payload containing an array of the matched elements, or multiple messages that each contains a matched element.



Selector: H1

Output: the html content of the elements

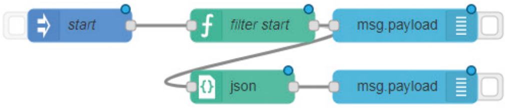
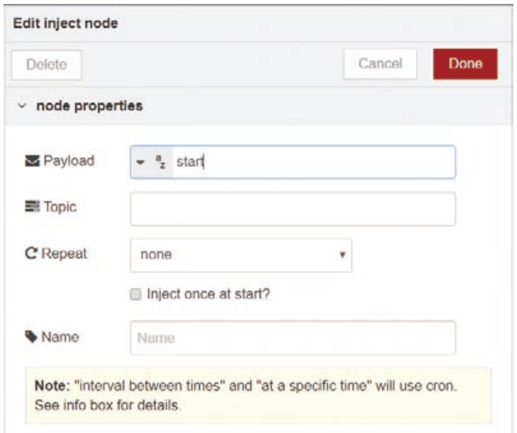
as a single message containing an array

Name: Name

Selector	The name of header of element.

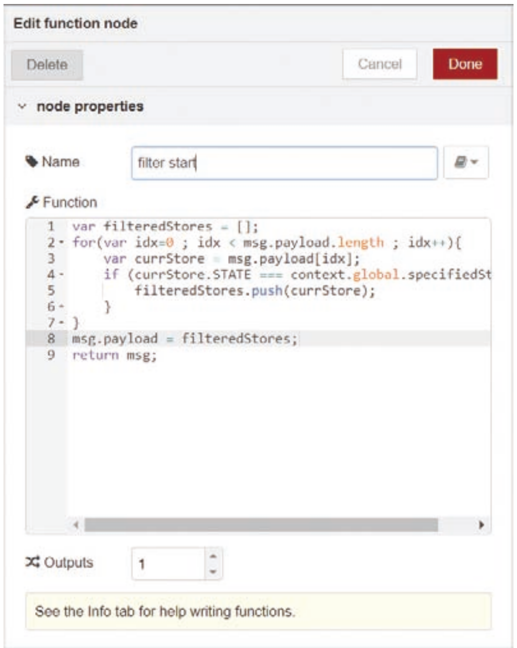
3.3.16 json

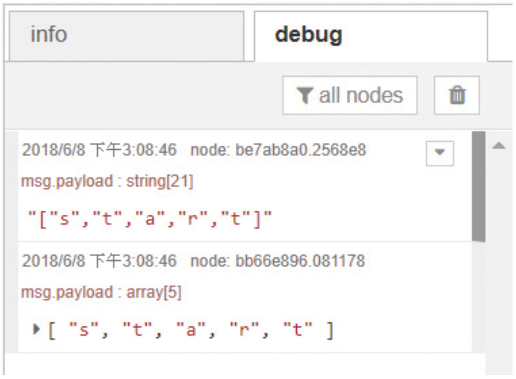
The json node parses the msg.payload to convert a json string to/from a JavaScript object and place the result back into the payload.

Example: Output a string “start” and the json string.		
Step	**Description**	**Screenshot**
1	Add and connect 1 **inject** node, 1 **function** node, 1 **json** node, and 2 **debug** nodes to the workspace as shown.	
2	Edit the **inject** node to insert a string **start**.	Edit inject node Delete Cancel Done node properties Payload: start Topic: Repeat: none □ Inject once at start? Name: Name **Note:** "interval between times" and "at a specific time" will use cron. See info box for details.

Example: Output a string “start” and the json string.

Step	Description	Screenshot

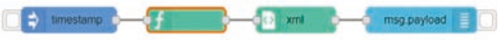
3	Edit the **function** node to add a set of JavaScript codes as shown.
	var filteredStores = [];
	for(var idx=0 ; idx < msg.payload.length ; idx++){
	    var currStore = msg.payload[idx];
	    if (currStore.STATE === context.global.specifiedState){
	        filteredStores.push(currStore);
	    }
	}
	msg.payload = filteredStores;
		
		Edit function node
		Delete Cancel Done
		node properties
		Name: filter start
		Function
		var filteredStores = [];
		for(var idx=0 ; idx < msg.payload.length ; idx++){
		    var currStore = msg.payload[idx];
		    if (currStore.STATE === context.global.specifiedSt
		        filteredStores.push(currStore);
		    }
		}
		msg.payload = filteredStores;
		return msg;
		Outputs: 1
		See the Info tab for help writing functions.

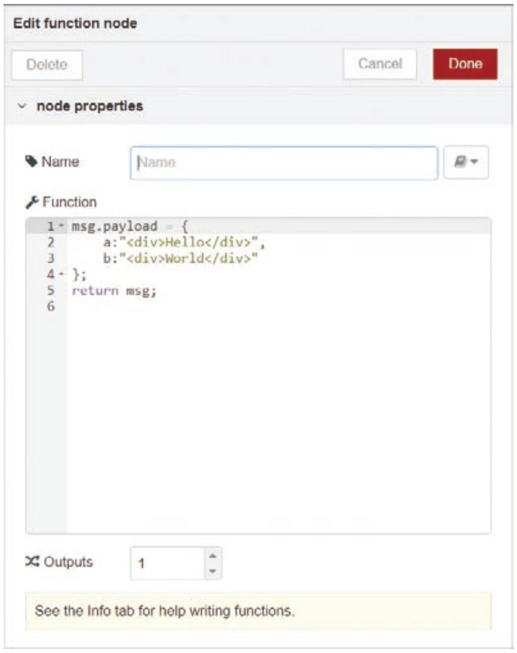
4	Deploy the flow and click the button on the left of the inject node and the result will be shown in the debug tab.	 info debug all nodes 2018/6/8 下午3:08:46 node: be7ab8a0.2568e8 msg.payload : string[21] "["s","t","a","r","t"]" 2018/6/8 下午3:08:46 node: bb66e896.081178 msg.payload : array[5] ▸["s", "t", "a", "r", "t"]

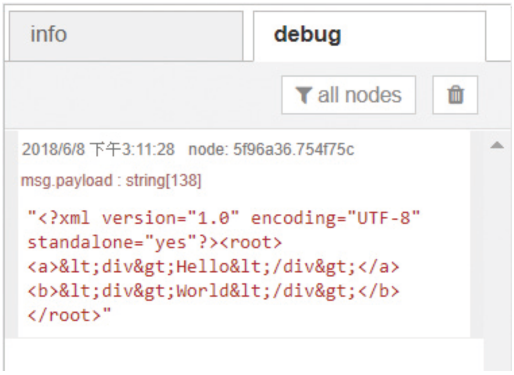
3.3.17 xml

The xml node parses the msg.payload to convert xml to/from a JavaScript object, and places the result in the payload.

Example: Generate an xml file for an object.

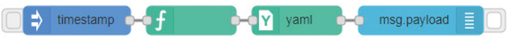
Step	Description	Screenshot
1	Add and connect 1 **inject** node, 1 **function** node, 1 **xml** node, and 1 **debug** node to the workspace as shown.	

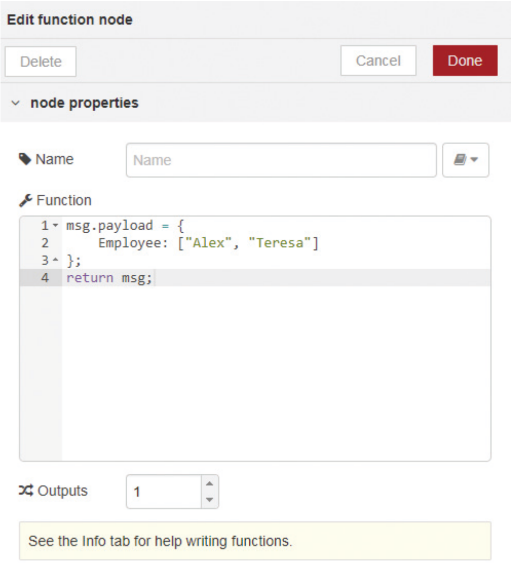
2	Edit the **function** node and add the codes as shown below.
	msg.payload = {
	    a:"<div>Hello</div>",
	    b:"<div>World</div>"
	};
	return msg;
		 Edit function node
		Delete
		Cancel
		Done
		node properties
		Name
		Function
		Outputs
		Name: Name
		msg.payload = {
		    a:"<div>Hello</div>",
		    b:"<div>World</div>"
		};
		return msg;
		Outputs: 1
		See the Info tab for help writing functions.

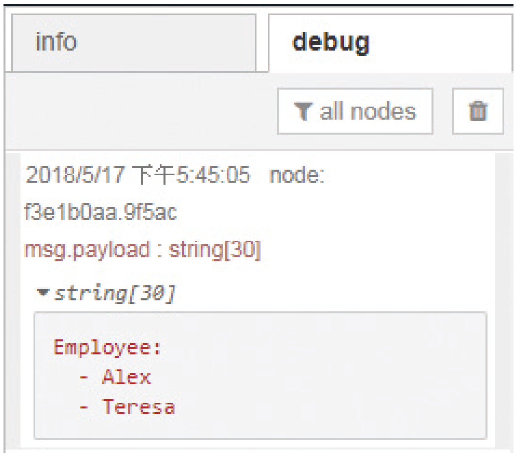
3	Deploy the flow and click the button on the left of the inject node and the result will be shown in the debug tab.	 2018/6/8 下午3:11:28 node: 5f96a36.754f75c msg.payload : string[138] "<?xml version="1.0" encoding="UTF-8" standalone="yes"?><root><a>&lt;div>Hello</div&gt;</a><b>&lt;div>World</div&gt;</b></root>"

3.3.18 yaml

The yaml node parses the msg.payload to convert yaml to/from a JavaScript object, and places the result in the payload.

Example: Generate an xml file for an object.		
Step	**Description**	**Screenshot**
1	Add and connect 1 **inject** node, 1 **function** node, 1 **yaml** node, and 1 **debug** node to the workspace as shown.	

2	Edit the **function** node and add the codes as shown below.
	msg.payload = {
	   Employee: ["Alex", "Teresa"]
	};
	return msg;
		 Edit function node
		Delete
		Cancel
		Done
		node properties
		Name: Name
		Function
		msg.payload = {
		    Employee: ["Alex", "Teresa"]
		};
		return msg;
		Outputs: 1
		See the Info tab for help writing functions.

3	Deploy the flow and click the button on the left of the inject node and the result will be shown in the debug tab.	 2018/5/17 下午5:45:05 node: f3e1b0aa.9f5ac msg.payload : string[30] ▾string[30] Employee: - Alex - Teresa

3.3.19 aws thing

A thing shadow (sometimes referred to as a device shadow) is a JSON document that is used to store and retrieve current state information for a thing (device, app, and so on). The Thing Shadows service maintains a thing shadow for each thing you connect to AWS IoT. You can use thing shadows to get and set the state of a thing over MQTT or HTTP, regardless of whether the thing is connected to the Internet. Each thing shadow is uniquely identified by its name.
The input pin accepts a msg.payload with JSON format following the structure of AWS ThingShadow document for UPDATE.

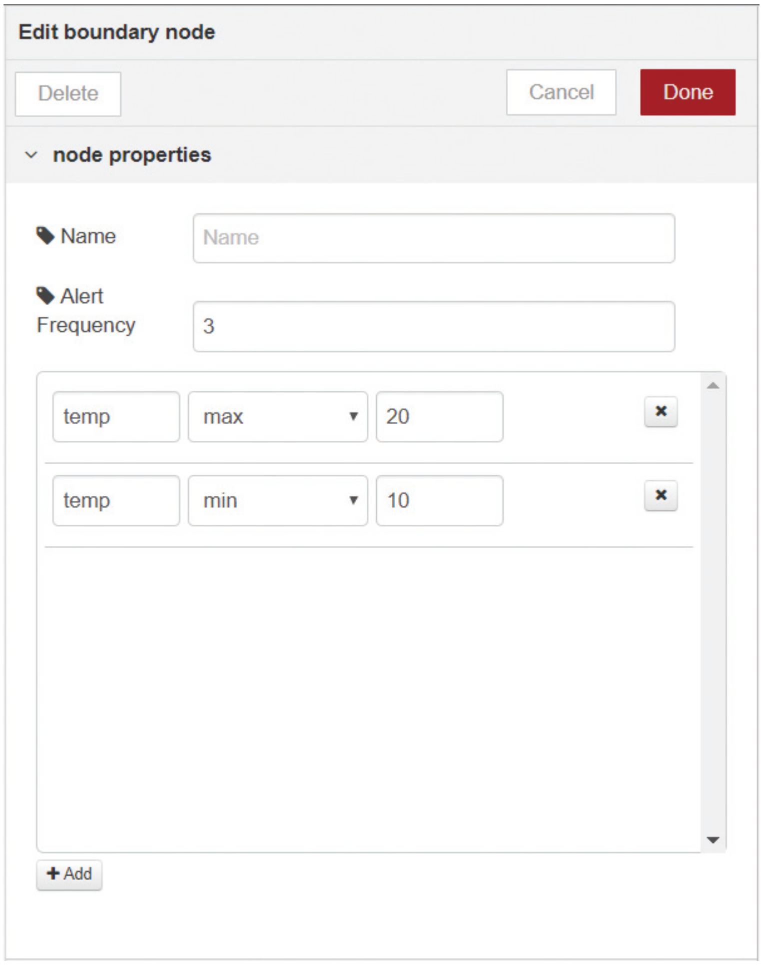
Device	**Add new aws-iot-device…**	
	Name	The name of the device to be connected.
	Type	MQTT Broker/Thing Shadow.
	Client ID	The name of the device.
	Endpoint	The Rest API Endpoint of the device.
	AWS Certs	The path where the certificate files of the device is located.
Method	**GET/UPDATE/DELETE**	

3.4 Data Process Nodes

Data process nodes provide functions for information manipulations.

3.4.1 boundary

The boundary node triggers the alarm since the value of object is out of the range of setting.

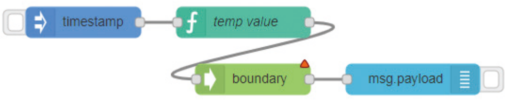


Edit boundary node

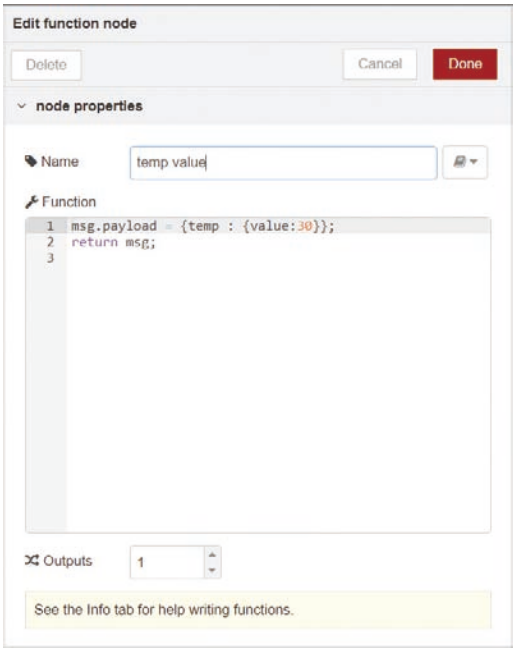
Name: Name

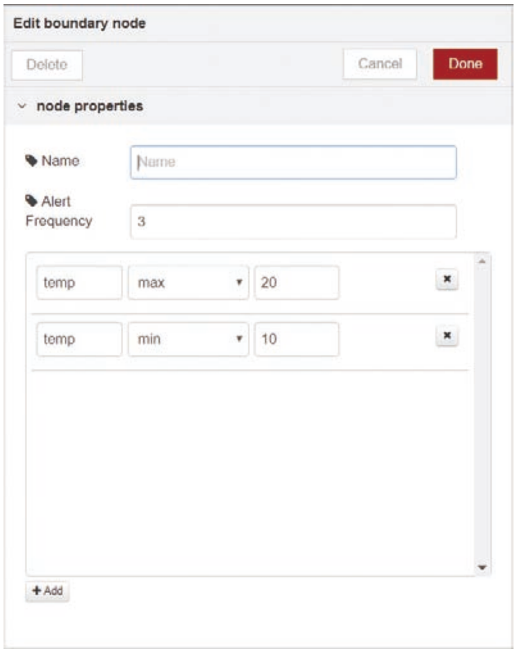
Alert Frequency: 3

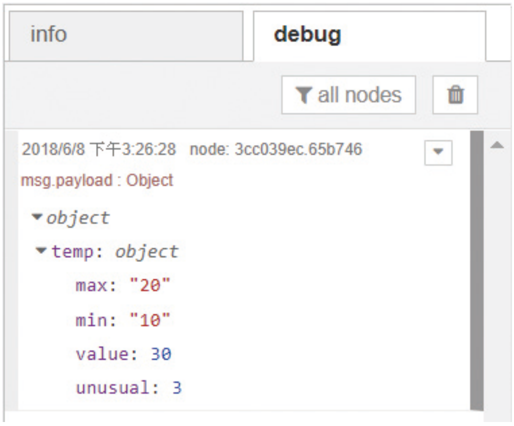
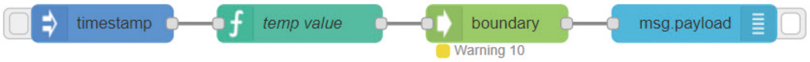
temp	max	20
temp	min	10

Example: Monitor a value in a preset range and trigger alarms if not in the range.		
Step	**Description**	**Screenshot**
1	Add and connect 1 **inject** node, 1 **function** node, 1 **boundary** node, and 1 **debug** node to the workspace as shown.	 timestamp temp value boundary msg.payload

Example: Monitor a value in a preset range and trigger alarms if not in the range.		
Step	**Description**	**Screenshot**

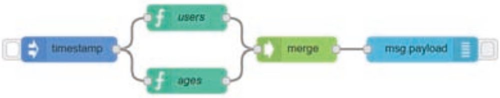
2	Edit the **function** node. Add **msg.payload = {temp : {value:30}}** in **Function** field as shown.	 Edit function node
		Delete
		Cancel
		Done
		node properties
		Name: temp value
		Function:
		`msg.payload = {temp : {value:30}};`
		`return msg;`
		Outputs: 1
		See the Info tab for help writing functions.

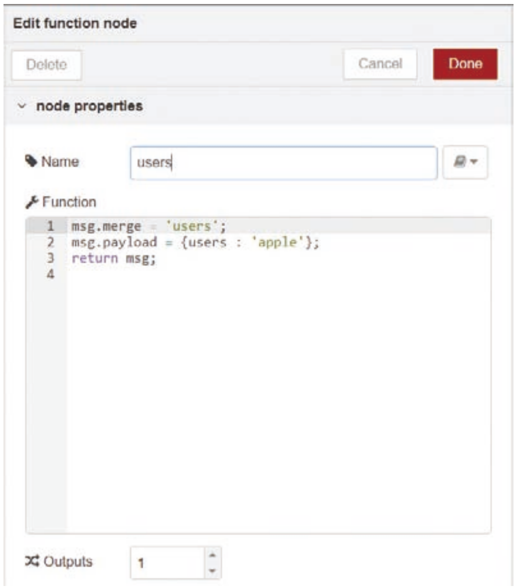
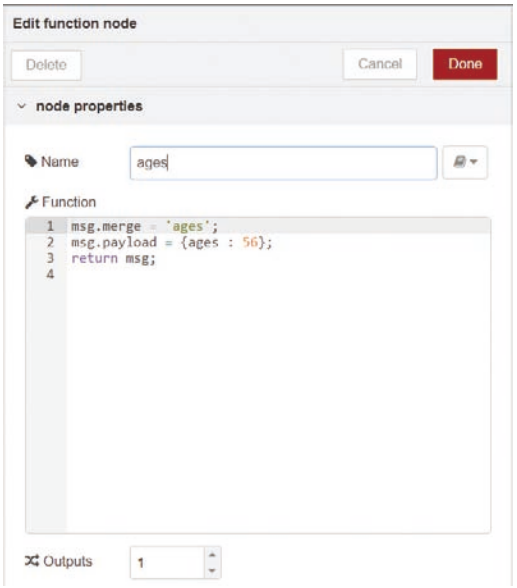
3	Edit the **boundary** node. Set the values as shown, so if the value is greater than 20 or less than 10, then the node triggers the alarm.	 Edit boundary node Delete Cancel Done node properties Name: Name Alert Frequency: 3 temp | max | 20 temp | min | 10

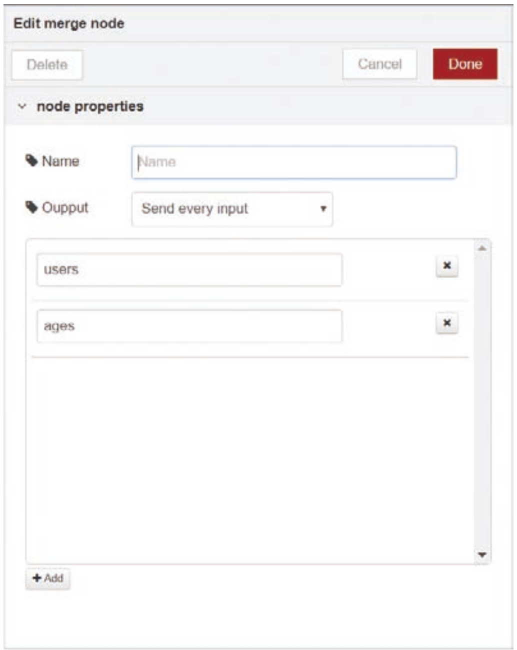
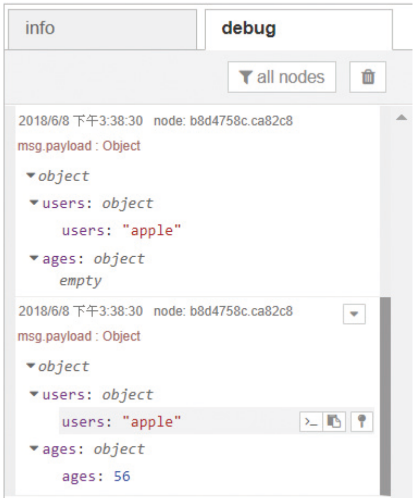
Example: Monitor a value in a preset range and trigger alarms if not in the range.		
Step	**Description**	**Screenshot**
4	Deploy the flow and click the button on the left of the inject node. The result will be shown in the debug tab. Notice the boundary node in the workspace prompt warnings.	 2018/6/8 下午3:26:28 node: 3cc039ec.65b746 msg.payload : Object ▾object ▾temp: object max: "20" min: "10" value: 30 unusual: 3
	 timestamp temp value boundary Warning 10 msg.payload	

3.4.2 merge

The merge node merges two objects into one object for specific data processing.

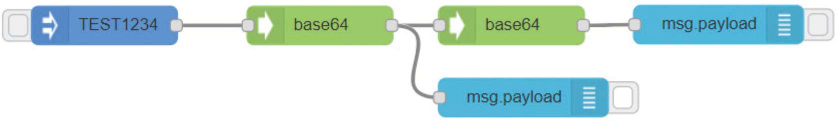
Example: Merge two columns, users and ages into one object for data processing.		
Step	**Description**	**Screenshot**
1	Connect 1 **inject** node, 2 **function** nodes, 1 **merge** node, and 1 **debug** node to the workspace as shown.	

2	Edit both the **function** nodes and create two objects named "**users**" and "**ages**" respectively as shown.	 Edit function node
		Delete
		Cancel
		Done
		node properties
		Name: users
		Function
		`msg.merge = 'users';`
		`msg.payload = {users : 'apple'};`
		`return msg;`
		Outputs: 1
		 Edit function node
		Delete
		Cancel
		Done
		node properties
		Name: ages
		Function
		`msg.merge = 'ages';`
		`msg.payload = {ages : 56};`
		`return msg;`
		Outputs: 1

Example: Merge two columns, users and ages into one object for data processing.		
Step	**Description**	**Screenshot**
3	Edit the **merge** node. Input the subject of each object with **user** and **ages**.	 Edit merge node Delete Cancel Done node properties Name: Name Oupput: Send every input users ages + Add
4	Deploy the flow and click the button on the left of the inject node, and the result will be shown in the debug tab.	 info debug all nodes 2018/6/8 下午3:38:30 node: b8d4758c.ca82c8 msg.payload : Object ▾object ▾users: object users: "apple" ▾ages: object empty 2018/6/8 下午3:38:30 node: b8d4758c.ca82c8 msg.payload : Object ▾object ▾users: object users: "apple" ▾ages: object ages: 56

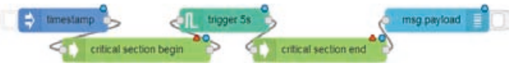
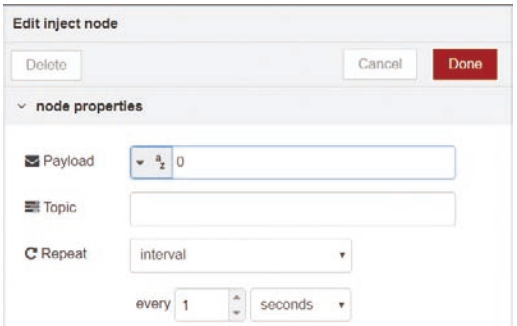
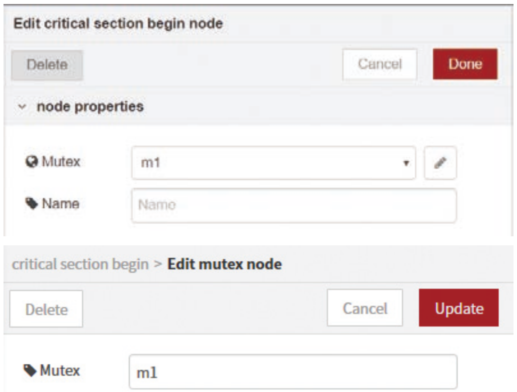
3.4.3 cypher

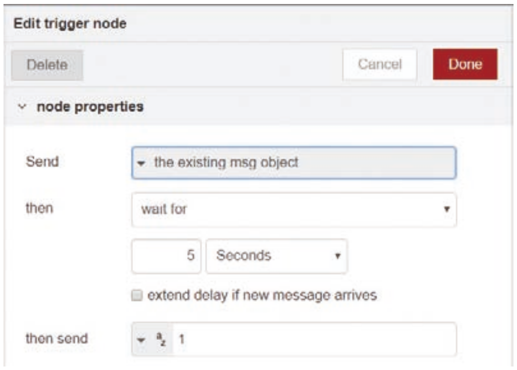
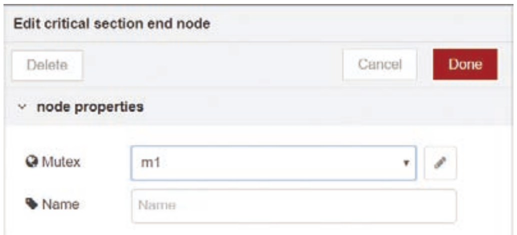
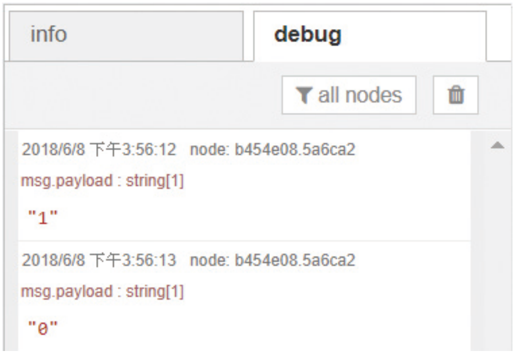
The cypher node encrypts or decrypts the data stream to and from input source based on base64 or 3DES algorithm.



3.4.4 critical section

The critical section nodes guarantee the first-in & first-out in a task pipeline. Applying both critical section begin and critical section end nodes concurrently to your flow will ensure the job gets done in sync without hassles. You can have more than one pair of critical section in your flow. Make sure the corresponding pair shares the same mutex.

Example: Set up a critical section to force the one second interval request to wait for 5 seconds for output.		
Step	**Description**	**Screenshot**
1	Connect 1 **inject** node, 1 **critical section begin** node, 1 **trigger** node, 1 **critical section end** node, and 1 **debug** node to the workspace as shown.	 timestamp critical section begin trigger 5s critical section end msg.payload
2	Edit the **inject** node as shown and click **Done**.	 Edit inject node Delete Cancel Done node properties Payload: 0 Topic: Repeat: interval every 1 seconds
3	Click on the **critical section begin** node and click ✎ . Input m1 in the **Mutex** field. Click **Update** when done. Click **Done** to exit editing node.	 Edit critical section begin node Delete Cancel Done node properties Mutex: m1 Name: Name critical section begin > Edit mutex node Delete Cancel Update Mutex: m1

Example: Set up a critical section to force the one second interval request to wait for 5 seconds for output.		
Step	**Description**	**Screenshot**
4	Edit the **trigger** node as shown and click **Done**.	 Edit trigger node Delete Cancel Done node properties Send: the existing msg object then: wait for 5 Seconds ☐ extend delay if new message arrives then send: 1
5	Click on the **critical section end** node and select m1 from the **Mutex** drop-down menu. Click **Done** when finished.	 Edit critical section end node Delete Cancel Done node properties Mutex: m1 Name: Name
6	Deploy the flow and click the button on the left of the inject node, and the result will be shown in the debug tab.	 2018/6/8 下午3:56:12 node: b454e08.5a6ca2 msg.payload : string[1] "1" 2018/6/8 下午3:56:13 node: b454e08.5a6ca2 msg.payload : string[1] "0"

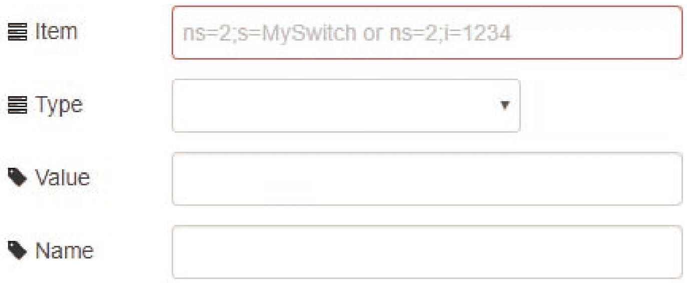
3.4.5 HWInfo

The HWInfo node reveals hardware information of the host installed with Xcare.

3.5 OPC UA Nodes

3.5.1 OpcUA Item

The OpcUA Item node defines the OPC UA item, type, and value to represent the connection to data sources within the server.



Item: ns=2;s=MySwitch or ns=2;i=1234

Type:

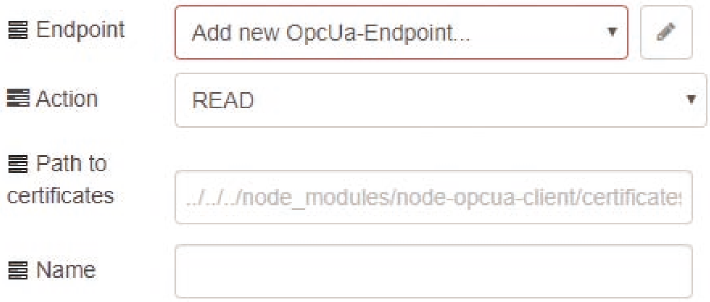
Value:

Name:

Item	Description
Item	The item block should contain OPC UA item address.
Type	The data type of the chosen item.
Value	The value would be written to the chosen item, if it isn't filled, the node sends payload.
Name	The name of the node.

3.5.2 OpcUA Client

Use this node to interact with an OpcUa Server. The value to write should be injected by an OpcUA Item. The value is written as json in msg.payload.



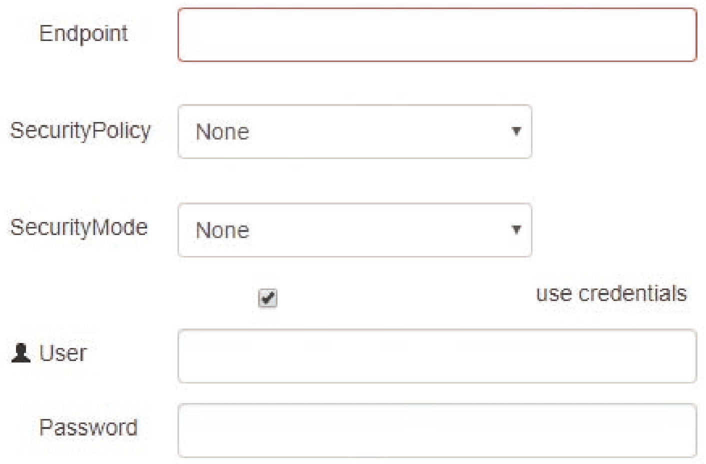
Endpoint: Add new OpcUa-Endpoint...

Action: READ

Path to certificates: ../../../node_modules/node-opcua-client/certificate:

Name:

Item	Option	Description
Endpoint	The OPC UA endpoint.	
Action	**Read**	Read data of the target item.
	Write	Write data to the target item.
	Browse	Browse and get information of the target item.
	Subscribe	Subscribe and check changes of the target item with fixed interval.
	Unsubscribe	Unsubscribe item.
	Event	Subscribe to the target item's event with fixed interval.
	Info	Get info of the target item.
Path to certificates	The data path of certificates in order to proceed.	
Name	The name of the node.	



Endpoint

SecurityPolicy None

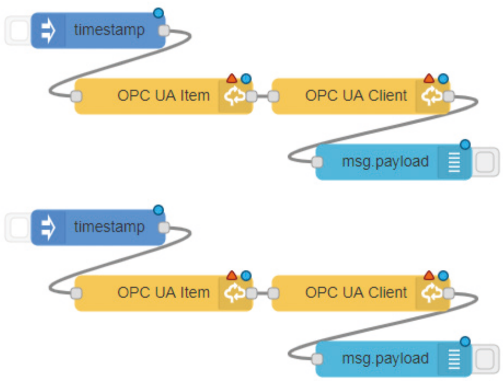
SecurityMode None

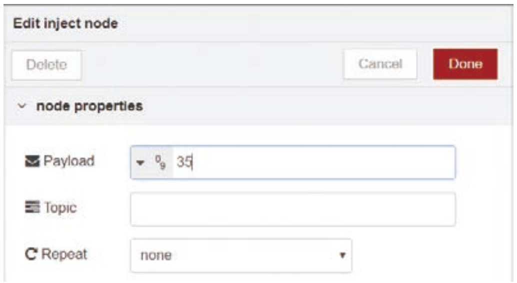
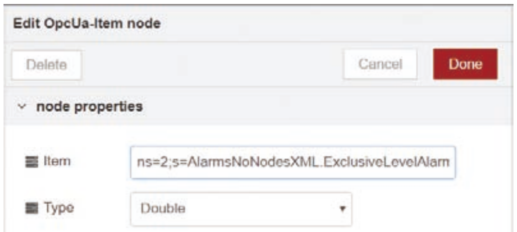
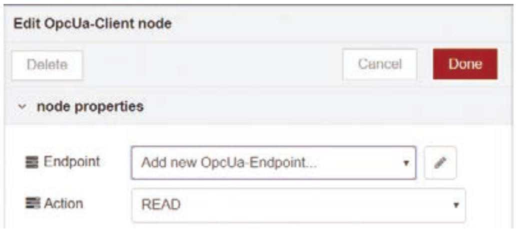
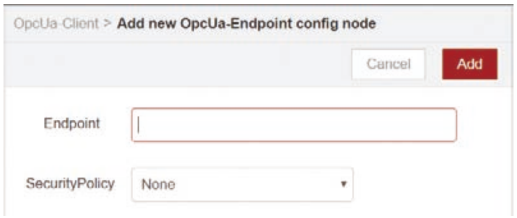
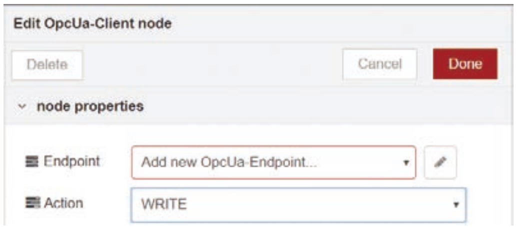
☑ use credentials

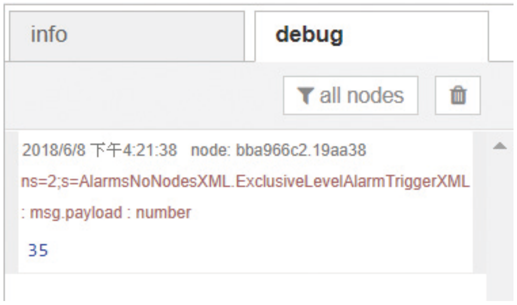
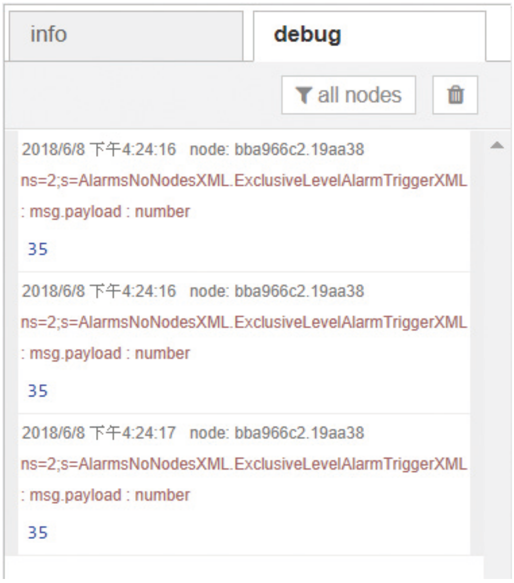
User

Password

Item	Option	Description
Endpoint	The path of the OPC UA endpoint.	
Security Policy	**None**	Enable/Disable security policies.
	Basic128	
	Basic128Rsa15	
	Basic256	
	Basic256Sha256	
Security Mode	**None**	Enable/Disable security mode.
	Sign	
	Sign & Encrypt	
use credentials	Enable log in to OPC UA.	
User/Password	Only available when credentials enabled.	

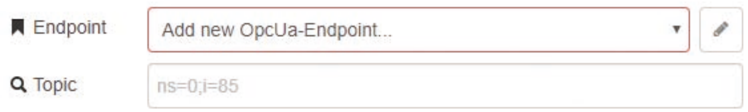
Example: Write a value to an OPC UA server & read a value from an OPC UA server.		
Step	**Description**	**Screenshot**
0	Prepare an OPC UA server.	
1	Connect 2 **inject** nodes, 2 **OPC UA Item** nodes, 2 **OPC UA Client** nodes, and 2 **debug** nodes to the workspace as shown.	 timestamp OPC UA Item OPC UA Client msg.payload timestamp OPC UA Item OPC UA Client msg.payload

Example: Write a value to an OPC UA server & read a value from an OPC UA server.		
Step	**Description**	**Screenshot**
2	Edit the first **inject** node as shown and click **Done**.	 Edit inject node Payload: 35 Topic: Repeat: none
3	Edit the first **OPC UA Item** node. Fill the **Item** value for your OPC UA server and choose the data type in the **Type** drop-down menu. Click **Done** when finished.	 Edit OpcUa-Item node Item: ns=2;s=AlarmsNoNodesXML.ExclusiveLevelAlarm Type: Double
	Repeat the above to edit the second OPC UA item node with the same **Item** value.	
4	Edit the first **OPC UA Client** node. Click ✎ to add a new OPC UA endpoint.	 Edit OpcUa-Client node Endpoint: Add new OpcUa-Endpoint... Action: READ
	Fill the **Endpoint** address from your OPC UA server. Click **Add** when finished.	 OpcUa-Client > Add new OpcUa-Endpoint config node Endpoint: SecurityPolicy: None
	Set **Action** to **Write** and click **Done**.	 Edit OpcUa-Client node Endpoint: Add new OpcUa-Endpoint... Action: WRITE
	Repeat the above to edit the second **OPC UA Client** node with the same **Endpoint** address but set **Action** to **Read**.	

Example: Write a value to an OPC UA server & read a value from an OPC UA server.		
Step	**Description**	**Screenshot**
5	Deploy the flow and click the button on the left of the first inject node, and the result will be shown in the debug tab. This means that the value had been injected to your OPC UA server. You can use the OPC UA client software to verify the injection.	 2018/6/8 下午4:21:38 node: bba966c2.19aa38 ns=2;s=AlarmsNoNodesXML.ExclusiveLevelAlarmTriggerXML : msg.payload : number 35
	Click the button on the left of the second inject node, and the result will be shown in the debug tab. This means that the value had been retrieved from your OPC UA server.	 2018/6/8 下午4:24:16 node: bba966c2.19aa38 ns=2;s=AlarmsNoNodesXML.ExclusiveLevelAlarmTriggerXML : msg.payload : number 35 2018/6/8 下午4:24:16 node: bba966c2.19aa38 ns=2;s=AlarmsNoNodesXML.ExclusiveLevelAlarmTriggerXML : msg.payload : number 35 2018/6/8 下午4:24:17 node: bba966c2.19aa38 ns=2;s=AlarmsNoNodesXML.ExclusiveLevelAlarmTriggerXML : msg.payload : number 35

3.5.3 OpcUA Browser

Use it with an inject node of a timestamp or fill topic of msg object to browse the OPC UA item.

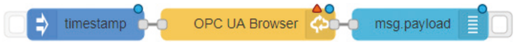
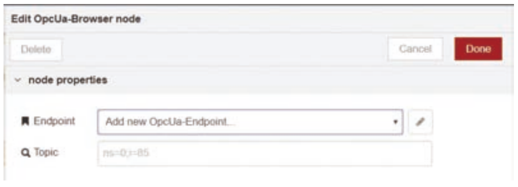
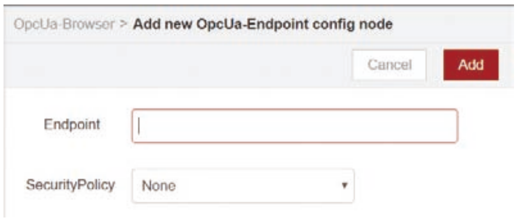


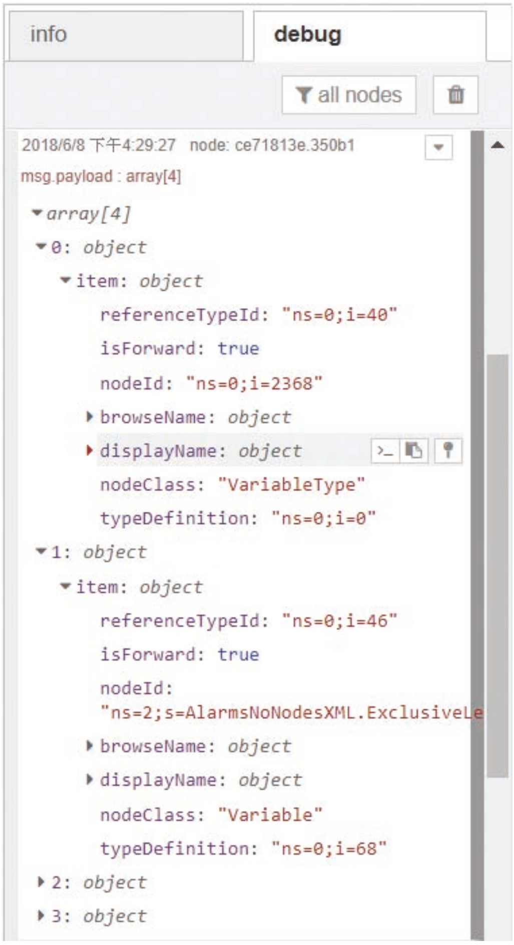
Endpoint: Add new OpcUa-Endpoint...

Topic: ns=0;i=85

Endpoint	The path of OPC UA endpoint.
Topic	Browse the target topic. msg.topic can also be used instead.

Example: Write a value to an OPC UA server & read a value from an OPC UA server.

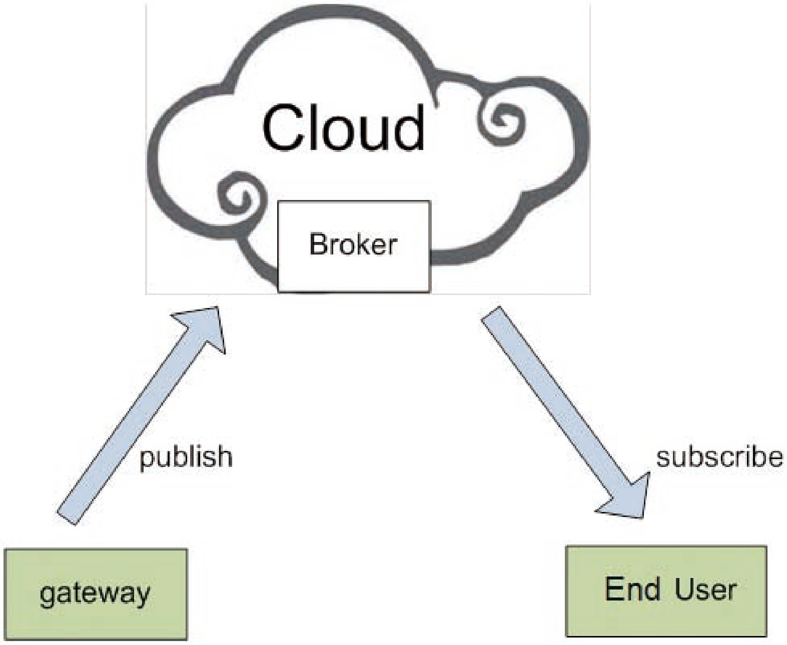
Step	Description	Screenshot
0	Prepare an OPC UA server.	
1	Connect 1 **inject** node, 1 **OPC UA Browser** node, and 1 **debug** node to the workspace as shown.	 timestamp → OPC UA Browser → msg.payload
2	Edit the **OPC UA Browser** node. Click ✎ to add a new OPC UA endpoint and the topic to browse.	 Edit OpcUa-Browser node Delete · Cancel · Done node properties Endpoint: Add new OpcUa-Endpoint... Topic: ns=0;i=85
	Fill the **Endpoint** address from your OPC UA server. Click **Add** when finished. Click **Done** to exit node editing.	 OpcUa-Browser > Add new OpcUa-Endpoint config node Cancel · Add Endpoint: SecurityPolicy: None

Example: Write a value to an OPC UA server & read a value from an OPC UA server.		
Step	**Description**	**Screenshot**
3	Deploy the flow and click the button on the left of the first inject node, and the result will be shown in the debug tab.	

```
2018/6/8 下午4:29:27   node: ce71813e.350b1
msg.payload : array[4]
▾array[4]
 ▾0: object
   ▾item: object
       referenceTypeId: "ns=0;i=40"
       isForward: true
       nodeId: "ns=0;i=2368"
      ▸browseName: object
      ▸displayName: object
       nodeClass: "VariableType"
       typeDefinition: "ns=0;i=0"
 ▾1: object
   ▾item: object
       referenceTypeId: "ns=0;i=46"
       isForward: true
       nodeId:
       "ns=2;s=AlarmsNoNodesXML.ExclusiveLe
      ▸browseName: object
      ▸displayName: object
       nodeClass: "Variable"
       typeDefinition: "ns=0;i=68"
 ▸2: object
 ▸3: object
```

3.6 Cloud Nodes

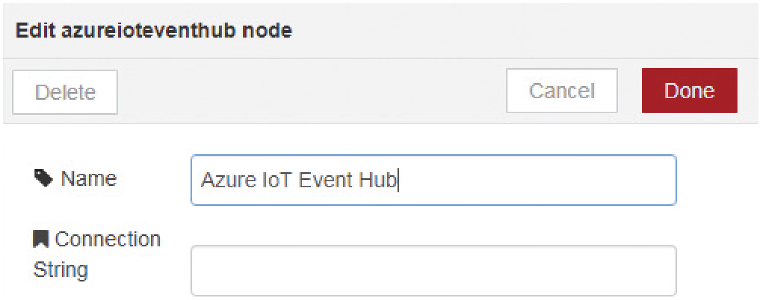
NexAIoT IoT Studio practices cloud applications via MQTT messaging protocol. The information on the gateway will be published to the brokers on the clouds such as IBM Bluemix, Microsoft Azure, or Amazon AWS, so end users can subscribe the information via MQTT from the brokers as shown.



Be sure to apply your own cloud account before using cloud nodes.

3.6.1 azureioteventhub

The azureioteventhub node receives the message payloads from a designated Azure IoT device.



Edit azureioteventhub node

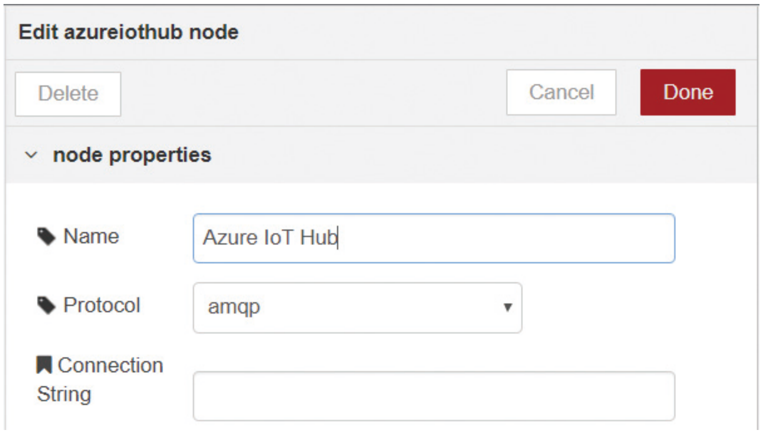
Name: Azure IoT Event Hub

Connection String:

Name	The name of the node.
Connection String	The given string as a key to link to the Azure^{TM} IoT Hub.

3.6.2 azureiothub

The azureiothub node sends field data to the Azure IoT Hub for live monitoring, field data extraction, device management, and so on. It supports multiple communication protocols including HTTP, MQTT, AMQP and AMQPWS.



Edit azureiothub node

Name: Azure IoT Hub

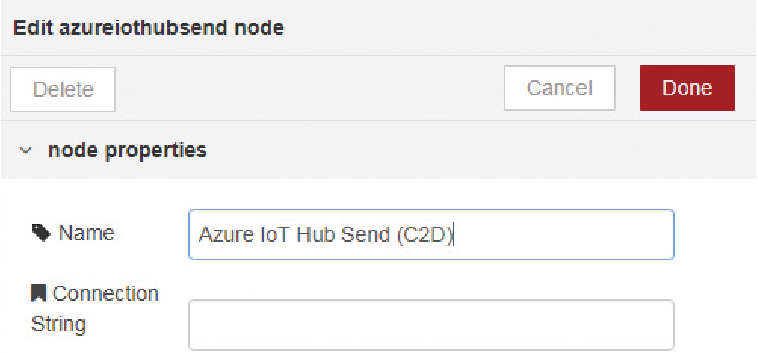
Protocol: amqp

Connection String:

Name	The name of the node.
Connection String	The given string as a key to link to the designated device of the Azure IoT Hub.

3.6.3 azureiothubsend

The azureiothubsend node sends the message payloads to a designated Azure IoT device.

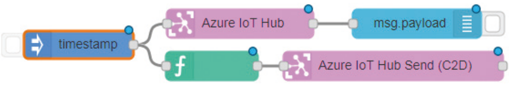
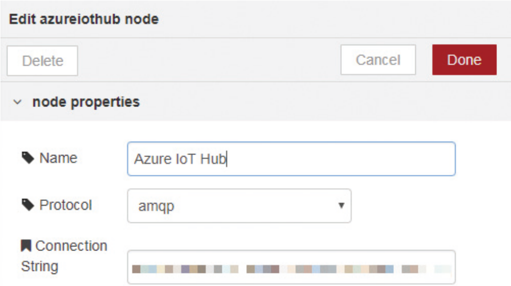


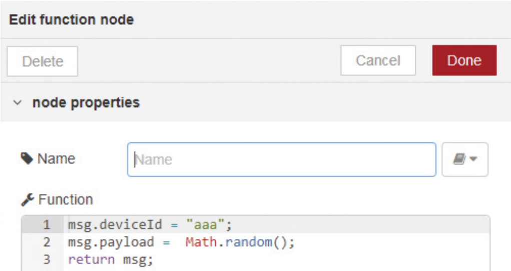
Edit azureiothubsend node

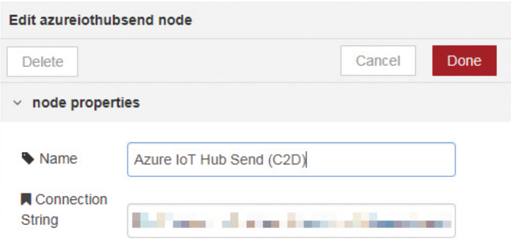
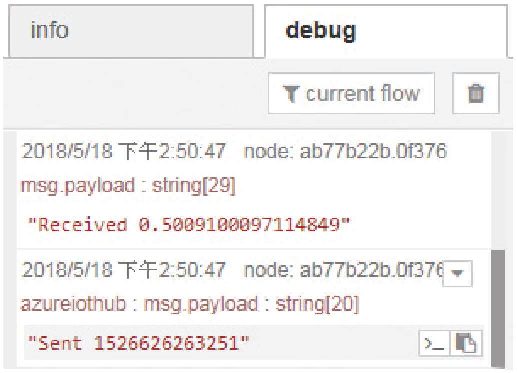
Name: Azure IoT Hub Send (C2D)

Connection String:

Name	The name of the node.
Connection String	The given string as a key to link to the Azure IoT Hub.

Example: Send a random value in the payload via the azureiothubsend node to the designated Azure IoT service.		
Step	**Description**	**Screenshot**
1	Add and connect 1 **inject** node, 1 **function** node, 1 **Azure IoT Hub Send (C2D)** node, 1 **Azure IoT hub** node, and 1 **debug** node to the workspace as shown.	 timestamp Azure IoT Hub msg.payload Azure IoT Hub Send (C2D)
2	Edit the **azureiothub** node. Input the connection string to the designated device of the Azure IoT Hub and click **Done**.	Edit azureiothub node Delete Cancel Done node properties Name: Azure IoT Hub Protocol: amqp Connection String: [redacted]

3	Edit the **function** node as shown and click **Done**.	Edit function node
		Delete Cancel Done
		node properties
		Name: Name
		Function
		`msg.deviceId = "aaa";`
		`msg.payload =  Math.random();`
		`return msg;`

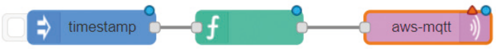
4	Edit the **azureiothubsend** node. Input the connection string to the Azure IoT Hub and click **Done**.	Edit azureiothubsend node Delete Cancel Done node properties Name: Azure IoT Hub Send (C2D) Connection String: [redacted]
5	Deploy the flow and click the button on the left of the inject node, and the result will be shown in the debug tab.	info debug current flow 2018/5/18 下午2:50:47 node: ab77b22b.0f376 msg.payload : string[29] "Received 0.5009100097114849" 2018/5/18 下午2:50:47 node: ab77b22b.0f376 azureiothub : msg.payload : string[20] "Sent 1526626263251"

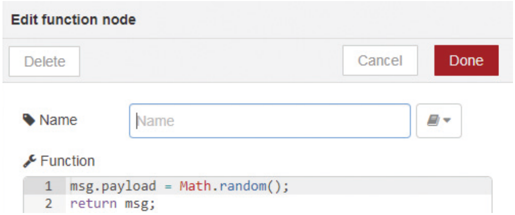
3.6.4 aws mqtt output

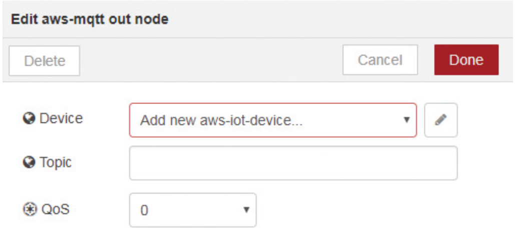
The aws matt output node writes to the Amazon Web Services AWS IoT as a publisher.

Item	Option	Description
Device	**Add new aws-iot-device...**	
	Name	The name of the device to be connected.
	Type	MQTT Broker/Thing Shadow.
	Client ID	The name of the device.
	Endpoint	The Rest API Endpoint of the device.
	AWS Certs	The path where the certificate files of the device is located.
Topic		The string used by the broker to filter messages. A topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator).
QoS	**0/1**	An agreement regarding the guarantees of delivering a message based on its network reliability and application logic.

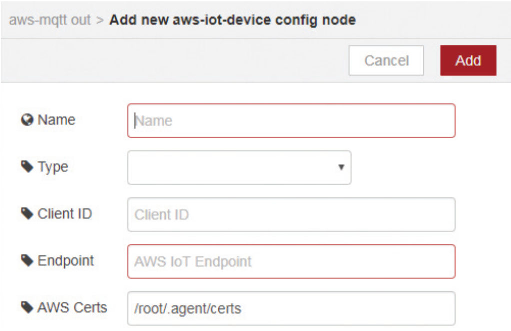
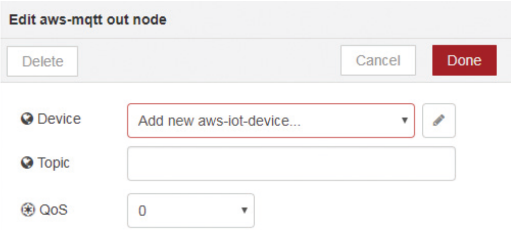
Example: Send a random value in the payload via the aws mqtt node to Amazon's AWS IoT service.

Step	Description	Screenshot
1	Add and connect 1 **inject** node, 1 **function** node, and 1 **aws mqtt** output node to the workspace as shown.	

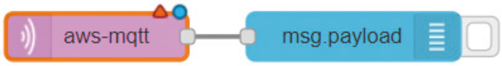
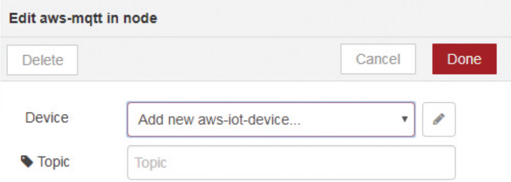
2	Edit the **function** node as shown and click **Done**.	
		Name: Name
		Function
		`msg.payload = Math.random();`
		`return msg;`

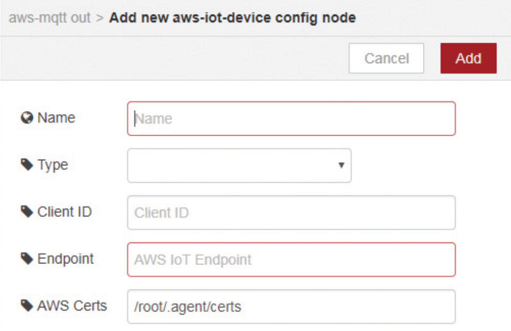
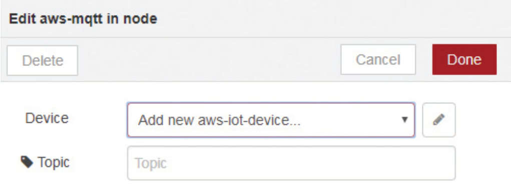
3	Edit the **aws mqtt** node. Select **Add new aws-iot-device...** and click the pencil icon.	 Device: Add new aws-iot-device... Topic: QoS: 0

Example: Send a random value in the payload via the aws mqtt node to Amazon's AWS IoT service.

Step	Description	Screenshot
4	Enter the name of your device in the **Name** and **Client** field. Select **MQTT Broker** in the **Type** drop-down menu. Enter the Rest API Endpoint of your device in the **Endpoint** field. Specify the path where the certificate files of your device will be placed in the **AWS Certs** field. Click **Add** to finish editing the node.	 aws-mqtt out > **Add new aws-iot-device config node** Cancel Add Name: Name Type: Client ID: Client ID Endpoint: AWS IoT Endpoint AWS Certs: /root/.agent/certs
5	Set a string for the subscribers to the broker in the **Topic** field. Set your desire QoS level in the **QoS** drop-down menu. Click **Done** when finished.	 **Edit aws-mqtt out node** Delete Cancel Done Device: Add new aws-iot-device... Topic: QoS: 0
6	Deploy the flow and log onto your AWS IoT service to view the result.	

Example: Subscribe to Amazon's AWS IoT service via the aws mqtt node.

Step	Description	Screenshot
1	Add and connect 1 **aws mqtt** output node and 1 **debug** node to the workspace as shown.	
2	Edit the **aws mqtt** node. Select **Add new aws-iot-device...** and click the pencil icon.	 **Edit aws-mqtt in node** Delete Cancel Done Device: Add new aws-iot-device... Topic: Topic

Example: Subscribe to Amazon's AWS IoT service via the aws mqtt node.		
Step	**Description**	**Screenshot**
3	Enter the name of your device in the **Name** and **Client** field. Select **MQTT Broker** in the **Type** drop-down menu. Enter the Rest API Endpoint of your device in the **Endpoint** field. Specify the path where the certificate files of your device will be placed in the **AWS Certs** field. Click **Add** to finish editing the node.	 aws-mqtt out > Add new aws-iot-device config node Cancel Add Name Name Type Client ID Client ID Endpoint AWS IoT Endpoint AWS Certs AWS Certs: /root/.agent/certs
4	Set a string for the subscribers to the broker in the **Topic** field. Set your desire QoS level in the **QoS** drop-down menu. Click **Done** when finished.	 Edit aws-mqtt in node Device: Add new aws-iot-device...
5	Deploy your flow and the subscribed messages from your AWS IoT service will show up in the **debug** tab.	

3.6.5 aws mqtt input

The aws matt input node reads from the Amazon Web Services AWS IoT as a publisher.

Item	Option	Description
Device	**Add new aws-iot-device…**	
	Name	The name of the device to be connected.
	Type	MQTT Broker/Thing Shadow.
	Client ID	The name of the device.
	Endpoint	The Rest API Endpoint of the device.
	AWS Certs	The path where the certificate files of the device is located.
Topic	The string used by the broker to filter messages. A topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator).	

3.6.6 Watson IoT input

The Watson IoT input node receives device commands from the IBM Watson Internet of Things Platform. The node can connect as either a Device or Gateway:

Item	Option	Description
Connect as	**Device**	Configured to either receive all commands or a specific command for the Device.
	Gateway	Configured to receive commands for all devices connected through the gateway, or to select a subset of them.
Credentials	**Add new wiotp-credentials**	
	Organization	The organization the device belongs to.
	Device Type	The type of device.
	Device ID	The ID of the device.
	Auth Token	The authorized token of the device.
Command	**specify command/all commands**	
QoS	**0/1/2**	An agreement regarding the guarantees of delivering a message based on its network reliability and application logic.
Name	The name of the node.	

3.6.7 Watson IoT output

The Watson IoT output node sends device events to the IBM Watson Internet of Things Platform. The node can connect as either a Device or Gateway in **Registered** mode or using **Quickstart** service.

Item	Option	Description
Connect as	**Device**	Configured to send events to the Device.
	Gateway	Configured to send events to all devices connected through the gateway, or to select a subset of them.
Credentials	**Add new wiotp-credentials**	
	Organization	The organization the device belongs to.
	Device Type	The type of device.
	Device ID	The ID of the device.
	Auth Token	The authorized token of the device.
Event type	The type of event.	
QoS	**0/1/2**	An agreement regarding the guarantees of delivering a message based on its network reliability and application logic.
Name	The name of the node.	

3.6.8 ibmiot output

The ibmiot output node can be used with Watson IoT Platform to send a command to a device or send an event on behalf of a device.

Item	Option	Description
Authentication	**Quickstart**	Use the Input Type property to configure this node to receive Events sent by IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to IoT Applications.
	API Key	
API Key	**Add new ibmiot...**	
	API Key	The API key to the device.
	API Token	The authentication token of the API key.
Output Type	**Device Event/Device Command**	
Device Type	The type of device.	
Device ID	The ID of the device.	
Event	The event type.	
Format	The format type.	
QoS	**0/1/2**	An agreement regarding the guarantees of delivering a message based on its network reliability and application logic.
Name	The name of the node.	

3.6.9 ibmiot input

The ibmiot input node can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications.

Item	Option	Description
Authentication	**Quickstart**	Use the Input Type property to configure this node to receive Events sent by IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to IoT Applications.
	API Key	
API Key	**Add new ibmiot...**	
	API Key	The API key to the device.
	API Token	The authentication token of the API key.
Input Type	**Device Event/Device Command/Device Status/ Application Status**	
Device Type	The type of device.	
Device ID	The ID of the device.	
Event	The event type.	
Format	The format type.	
QoS	**0/1/2**	An agreement regarding the guarantees of delivering a message based on its network reliability and application logic.
Name	The name of the node.	

3.7 Storage Nodes

Storage nodes initiate file transactions at the local host.

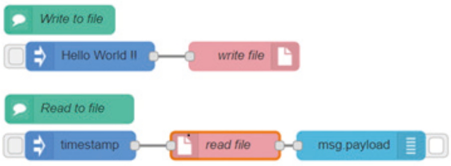
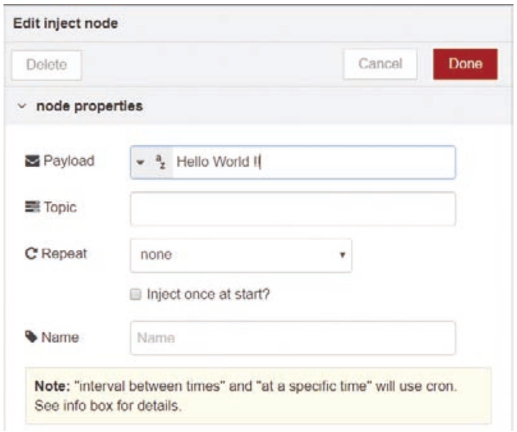
3.7.1 tail

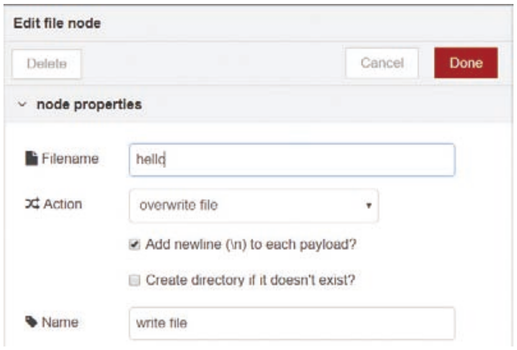
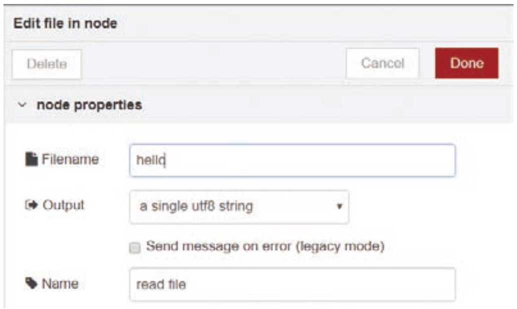
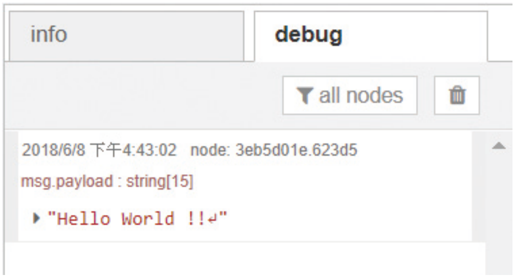
The tail node displays the last information appended into the file.

Note: This node is not available in Windows.

3.7.2 file in

The file in node reads the content of specific file in a UTF8 string/a buffer as **msg.payload** and the filename as **msg.filename** which can be configured in the node. If the filename is left blank, it should be set in an incoming message.

Example: Save a message to a file named "hello" on the gateway device and read the message from the file just saved.		
Step	**Description**	**Screenshot**
1	Add and connect 2 **inject** nodes, 1 **file in** node, 1 **file out** node and 1 **debug** node to the workspace as shown.	 Write to file Hello World !! write file Read to file timestamp read file msg.payload
2	Edit the **inject** node. Set the payload to string **Hello World!!**, the message saved to the file, and click **Ok**.	Edit inject node Delete Cancel Done node properties Payload: Hello World !! Topic: Repeat: none □ Inject once at start? Name: Name **Note:** "interval between times" and "at a specific time" will use cron. See info box for details.

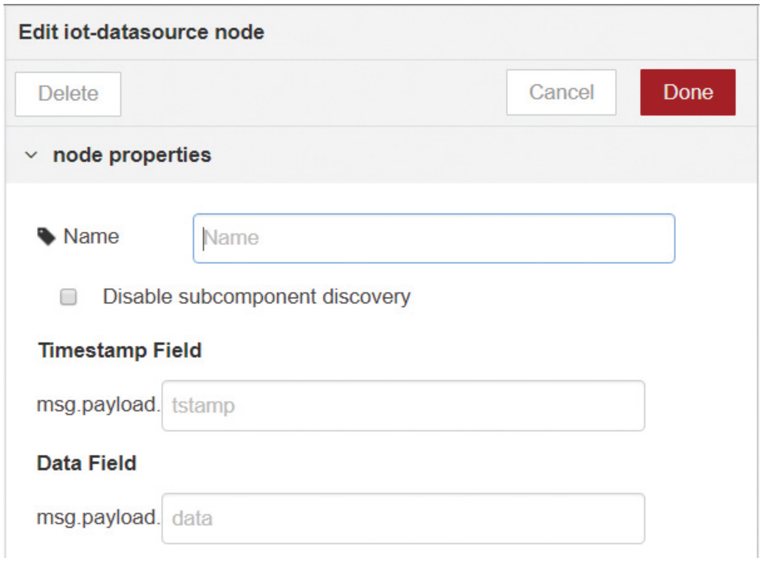
Example: Save a message to a file named “hello” on the gateway device and read the message from the file just saved.		
Step	**Description**	**Screenshot**
3	Edit the **file out** node. Put **hello** in the **Filename** field. In **Action** row, select **overwrite file** and check the checkbox before **Add newline (\n) to each payload?**. Put **write file** in the **Name** field and click **Ok**.	 Edit file node Filename: hello Action: overwrite file ☑ Add newline (\n) to each payload? □ Create directory if it doesn't exist? Name: write file
4	Edit the **file in** node. Put **hello** in the **Filename** field. Select **a utf8 string** in the field next to **Output**. Put **read file** in the **Name** field and click **Ok**.	 Edit file in node Filename: hello Output: a single utf8 string □ Send message on error (legacy mode) Name: read file
5	Deploy the flow and click the button on the left of the inject node and the result will be shown in the debug tab.	 2018/6/8 下午4:43:02 node: 3eb5d01e.623d5 msg.payload : string[15] ▸ "Hello World !!↵"

3.7.3 file out

The file out node writes msg.payload to the specified file, for example to create a log. You can configure the filename in the node. The filename shall be set in an incoming message on msg.filename if blank. A newline is added to every message, but this can be turned off if required, for example, to allow binary files to be written. The default behavior of the node is to append to the file, and this can be changed to overwrite the file each time, for example if you want to output a “static” web page or report, or to delete a file if required.

3.7.4 iot datasource

The iot datasource node provides the data as Dashboard data inputs.



Name: Name

□ Disable subcomponent discovery

Timestamp Field

msg.payload. tstamp

Data Field

msg.payload. data

<table>
<tr><th>Item</th><th>Description</th></tr>
<tr><td><b>Disable subcomponent discovery</b></td><td>If checked, the <b>iot datasource</b> node does not attempt to look inside the data field and split it into subfields. Example of discover enabled and data format is shown as below.<pre>
msg.payload = {
  tstamp: 1438637044000,
  data: {
    x: 3.14,
    y: 1.41,
    z: 6.02
  }
}
</pre><b>Iot datasource</b> node goes inside msg.payload.data and finds the fields x, y, and z, and presents them to the Dashboard as separate data points. Once disabled, the Dashboard will receive the entire JSON object msg.payload.data as one data point.</td></tr>
</table>

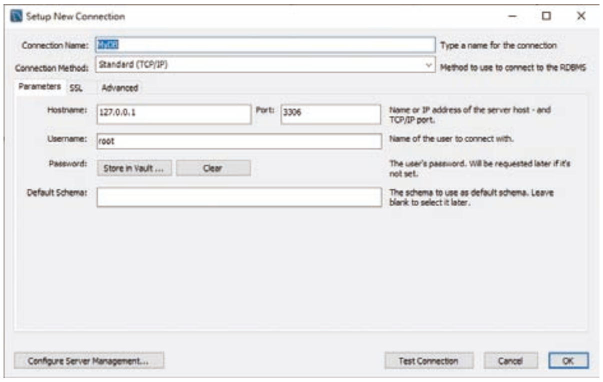
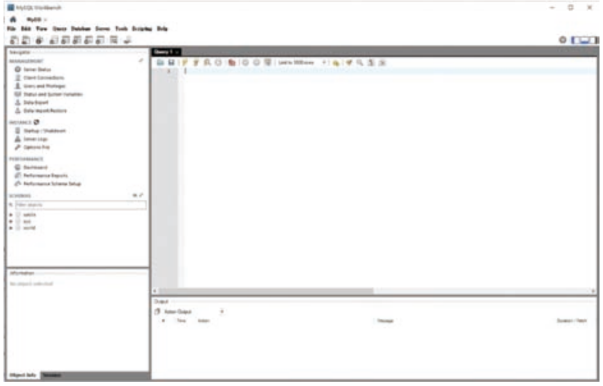
A line chart might need the data split up so it can chart the data points separately, but a 3D scattered plot would need the data intact since the entire object would represent just one data point on the plot.

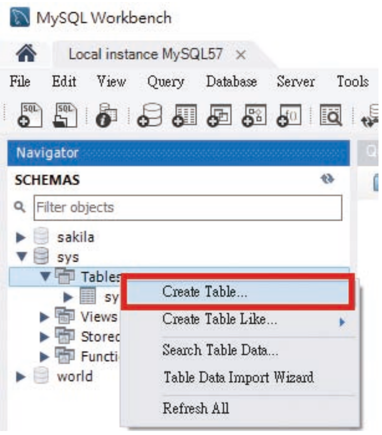
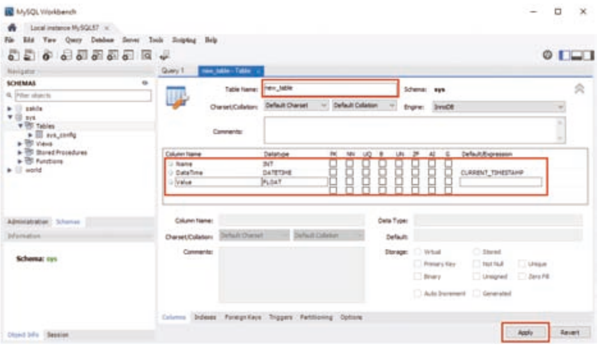
3.7.5 mysql

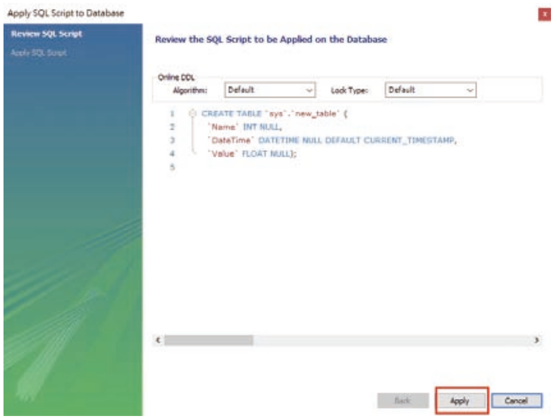
The mysql node allows basic access to a MySQL database. This node uses the query operation against the configured database and allows both INSERT and DELETE. Before using the node, please make sure you've downloaded MySQL to your device.

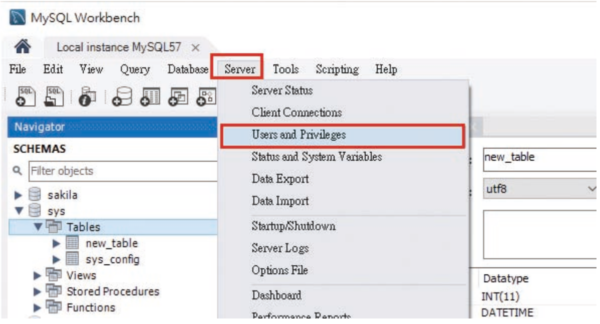
Note:

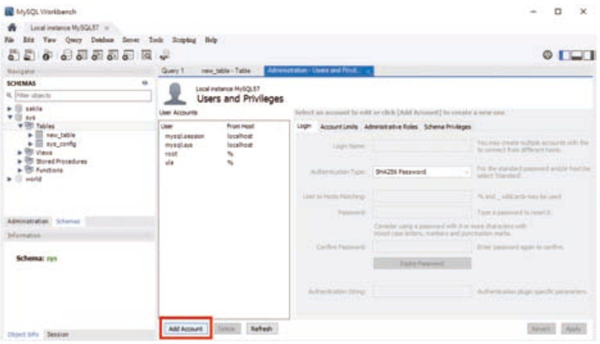
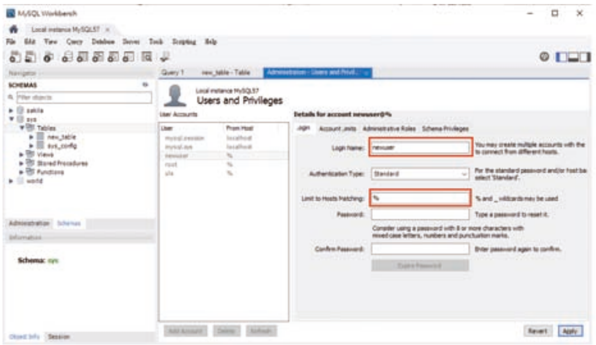
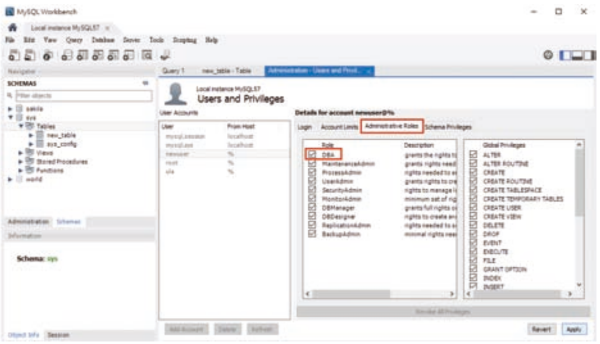
The MySQL Server version must be under 8.0.

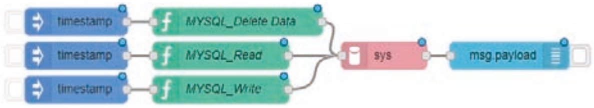
Example: Access MySQL database		
Step	**Description**	**Screenshot**
Step 1: Set Up the Database		
1-1	Launch **MySQL Workbench** and create a new connection. • **Connection Name**: any letters or numbers • **Hostname:** MySQL Server's IP address • **Port:** the default port is 3306 • **Username** • **Password** Click **OK**.	 Setup New Connection Connection Name: MyDB — Type a name for the connection Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS Parameters / SSL / Advanced Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host - and TCP/IP port. Username: root — Name of the user to connect with. Password: Store in Vault ... Clear — The user's password. Will be requested later if it's not set. Default Schema: — The schema to use as default schema. Leave blank to select it later. Configure Server Management... Test Connection Cancel OK
1-2	Connect to the database. (MyDB in this example.)	

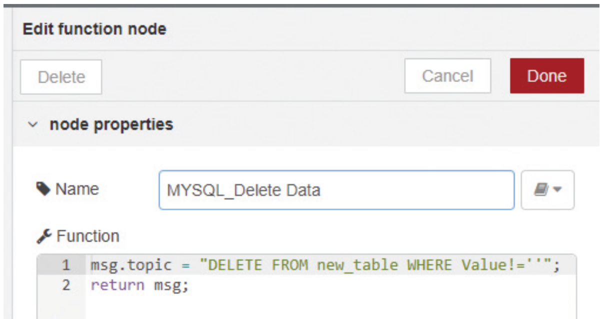
Example: Access MySQL database		
Step	**Description**	**Screenshot**
1-3	Right-click **sys** > **Tables** to the left of the page and click **Create Table**.	 MySQL Workbench Local instance MySQL57 File Edit View Query Database Server Tools Navigator SCHEMAS Filter objects sakila sys Tables Views Stored Functi world Create Table... Create Table Like... Search Table Data... Table Data Import Wizard Refresh All
1-4	Fill in the following information: • **Table Name:** the name of the table • **Column Name** • **Datatype** Click **Apply**.	Table Name: new_table Schema: sys Engine: InnoDB Name INT DateTime DATETIME CURRENT_TIMESTAMP Value FLOAT

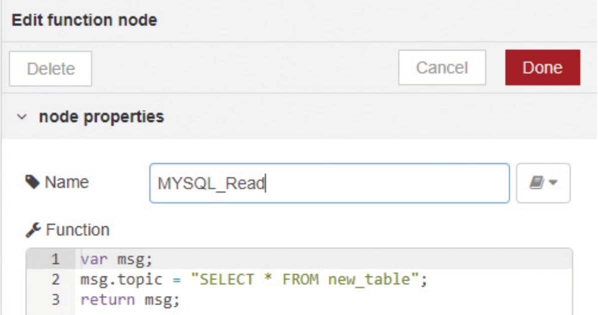
1-5	Review the SQL script and click **Apply** to create. Then click **Finish** when the application is complete.	Apply SQL Script to Database
		Review the SQL Script to be Applied on the Database
		Online DDL
		Algorithm: Default Lock Type: Default
		`CREATE TABLE `sys`.`new_table` (`
		`` `Name` INT NULL,``
		`` `DateTime` DATETIME NULL DEFAULT CURRENT_TIMESTAMP,``
		`` `Value` FLOAT NULL);``
		 Review SQL Script
		Apply SQL Script
		Back
		Apply
		Cancel

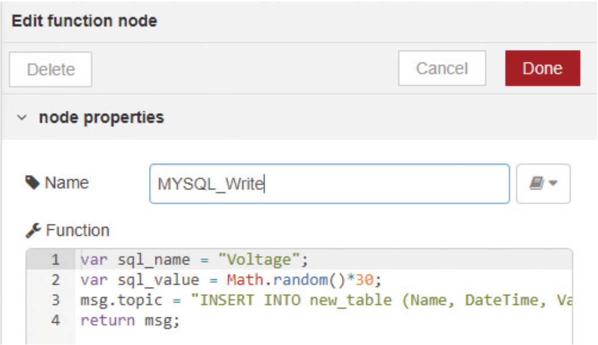
1-6	Navigate to **Server** > **Users and Privileges**.	 MySQL Workbench Local instance MySQL57 File Edit View Query Database Server Tools Scripting Help Navigator SCHEMAS Filter objects sakila sys Tables new_table sys_config Views Stored Procedures Functions Server Status Client Connections Users and Privileges Status and System Variables Data Export Data Import Startup/Shutdown Server Logs Options File Dashboard Performance Reports new_table utf8 Datatype INT(11) DATETIME

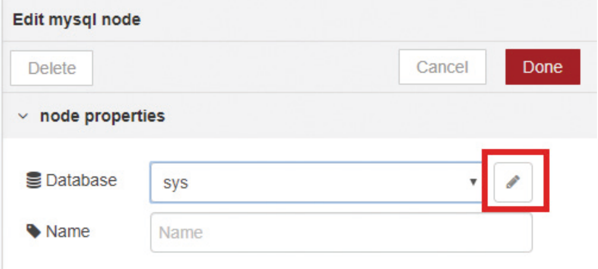
Example: Access MySQL database		
Step	**Description**	**Screenshot**
1-7	Click **Add Account** to create a new account.	
1-8	Fill in **Login Name**, **Limit to Hosts Matching** (enter % to let external IP connect), **Password**, and **Confirm Password**.	 Details for account newuser@% Login Name: newuser Authentication Type: Standard Limit to Hosts Matching: %
1-9	Select the **Administrative Roles** tab. Check the **DBA** box and click **Apply**.	 Details for account newuser@% ☑ DBA

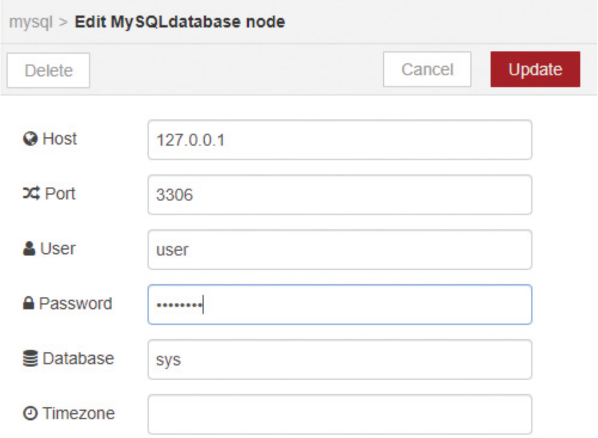
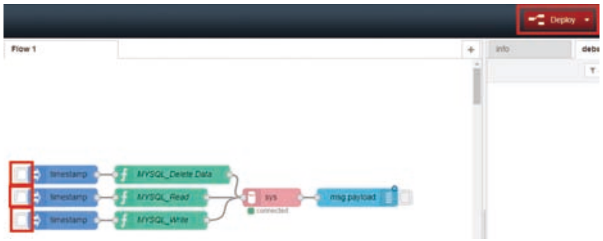
Example: Access MySQL database		
Step	**Description**	**Screenshot**
Step 2: IoT Studio Settings		
2-1	Add and connect 3 **inject** nodes, 3 **function** nodes, 1 **mysql** node, and 1 **debug** node to the workspace as shown.	 timestamp timestamp timestamp MYSQL_Delete Data MYSQL_Read MYSQL_Write sys msg.payload

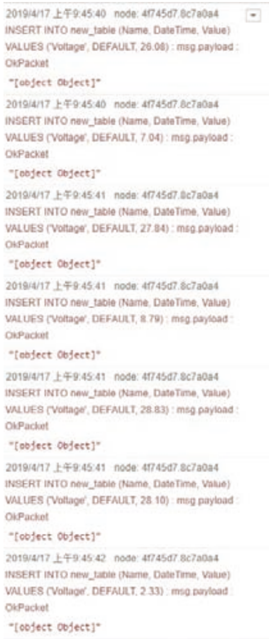
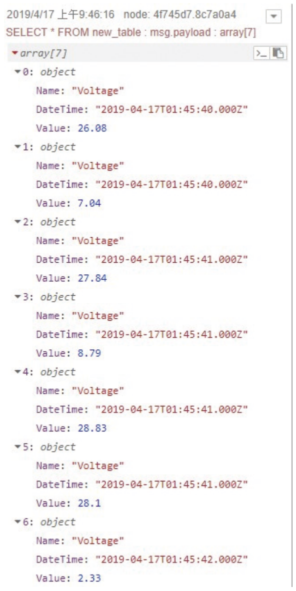
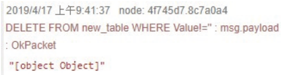
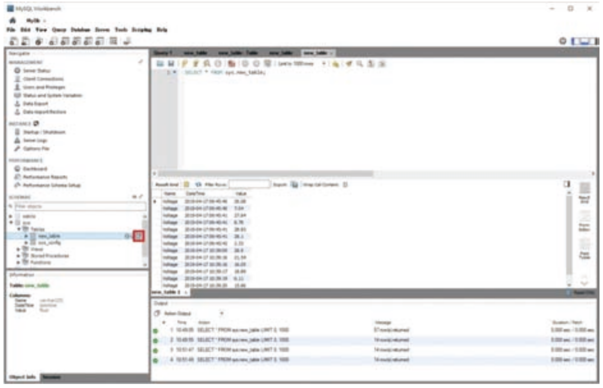
2-2	Edit one of the **function** nodes, input **MYSQL_ Delete Data** as the name, and edit the function code as shown.	 Edit function node
		Delete
		Cancel
		Done
		node properties
		Name: MYSQL_Delete Data
		Function:
		`msg.topic = "DELETE FROM new_table WHERE Value!=''";`
		`return msg;`

2-3	Edit another **function** node, input **MYSQL_Read** as the name, and edit the function code as shown.	 Edit function node
		Delete
		Cancel
		Done
		node properties
		Name: MYSQL_Read
		Function:
		`var msg;`
		`msg.topic = "SELECT * FROM new_table";`
		`return msg;`

2-4	Edit the final **function** node, input **MYSQL_Write** as the name, and edit the function code as shown.	 Edit function node
		Delete
		Cancel
		Done
		node properties
		Name: MYSQL_Write
		Function:
		`var sql_name = "Voltage";`
		`var sql_value = Math.random()*30;`
		`msg.topic = "INSERT INTO new_table (Name, DateTime, Va`
		`return msg;`

2-5	Edit the **mysql** node. Click ✎ to set server information.	 Edit mysql node Delete Cancel Done node properties Database: sys Name: Name

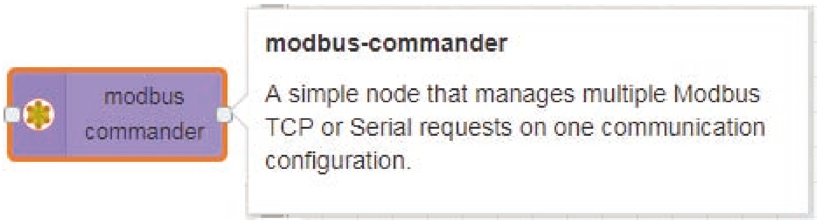
Example: Access MySQL database		
Step	**Description**	**Screenshot**
2-6	Fill in the **Host**, **Port**, **User**, **Password**, and **Database** fields as shown.	 mysql > **Edit MySQLdatabase node** Host: 127.0.0.1 Port: 3306 User: user Password: •••••••• Database: sys Timezone:
2-7	Click **Deploy** in the upper-right corner to deploy the flow and then click each inject button.	 Deploy Flow 1 info timestamp timestamp timestamp MYSQL_Delete Data MYSQL_Read MYSQL_Write sys connected msg.payload

Example: Access MySQL database		
Step	**Description**	**Screenshot**
2-8	You can see the data that has been written to, read from, and deleted in the database.	 2019/4/17 上午9:45:40 node: 4f745d7.8c7a0a4 INSERT INTO new_table (Name, DateTime, Value) VALUES ('Voltage', DEFAULT, 26.08) : msg.payload : OkPacket "[object Object]" 2019/4/17 上午9:45:40 node: 4f745d7.8c7a0a4 INSERT INTO new_table (Name, DateTime, Value) VALUES ('Voltage', DEFAULT, 7.04) : msg.payload : OkPacket "[object Object]" 2019/4/17 上午9:45:41 node: 4f745d7.8c7a0a4 INSERT INTO new_table (Name, DateTime, Value) VALUES ('Voltage', DEFAULT, 27.84) : msg.payload : OkPacket "[object Object]" 2019/4/17 上午9:45:41 node: 4f745d7.8c7a0a4 INSERT INTO new_table (Name, DateTime, Value) VALUES ('Voltage', DEFAULT, 8.79) : msg.payload : OkPacket "[object Object]" 2019/4/17 上午9:45:41 node: 4f745d7.8c7a0a4 INSERT INTO new_table (Name, DateTime, Value) VALUES ('Voltage', DEFAULT, 28.83) : msg.payload : OkPacket "[object Object]" 2019/4/17 上午9:45:41 node: 4f745d7.8c7a0a4 INSERT INTO new_table (Name, DateTime, Value) VALUES ('Voltage', DEFAULT, 28.10) : msg.payload : OkPacket "[object Object]" 2019/4/17 上午9:45:42 node: 4f745d7.8c7a0a4 INSERT INTO new_table (Name, DateTime, Value) VALUES ('Voltage', DEFAULT, 2.33) : msg.payload : OkPacket "[object Object]"  2019/4/17 上午9:46:16 node: 4f745d7.8c7a0a4 SELECT * FROM new_table : msg.payload : array[7] ▾array[7] ▾0: object Name: "Voltage" DateTime: "2019-04-17T01:45:40.000Z" Value: 26.08 ▾1: object Name: "Voltage" DateTime: "2019-04-17T01:45:40.000Z" Value: 7.04 ▾2: object Name: "Voltage" DateTime: "2019-04-17T01:45:41.000Z" Value: 27.84 ▾3: object Name: "Voltage" DateTime: "2019-04-17T01:45:41.000Z" Value: 8.79 ▾4: object Name: "Voltage" DateTime: "2019-04-17T01:45:41.000Z" Value: 28.83 ▾5: object Name: "Voltage" DateTime: "2019-04-17T01:45:41.000Z" Value: 28.1 ▾6: object Name: "Voltage" DateTime: "2019-04-17T01:45:42.000Z" Value: 2.33  2019/4/17 上午9:41:37 node: 4f745d7.8c7a0a4 DELETE FROM new_table WHERE Value!='' : msg.payload : OkPacket "[object Object]" 

3.8 Modbus TCP/RTU Commander Node

Modbus TCP/RTU Commander node is a simple node that manages multiple Modbus TCP or Serial requests on one communication configuration.

You can read coils/inputs/registers at the rate of the incoming message, and write coils/registers on each incoming message.



modbus-commander

A simple node that manages multiple Modbus TCP or Serial requests on one communication configuration.

The node is triggered manually by a specific payload message.

msg.enabled=true;.

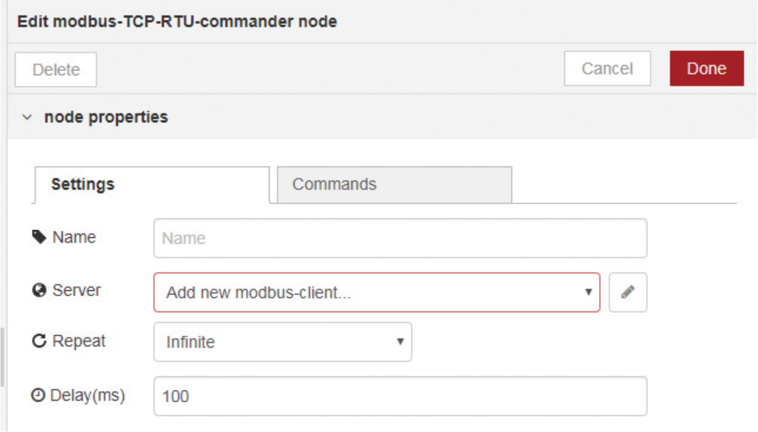
Send the following payload message to stop the flow.

msg.enabled=false;

The function codes currently supported include the following:

- FC 1: Read Coil Status
- FC 2: Read Input Status
- FC 3: Read Holding Registers
- FC 4: Read Input Registers
- FC 5: Force Single Coil
- FC 6: Preset Single Register
- FC 15: Force Multiple Coils
- FC 16: Preset Multiple Registers

Input parameter for connecting Modbus



Name: Name

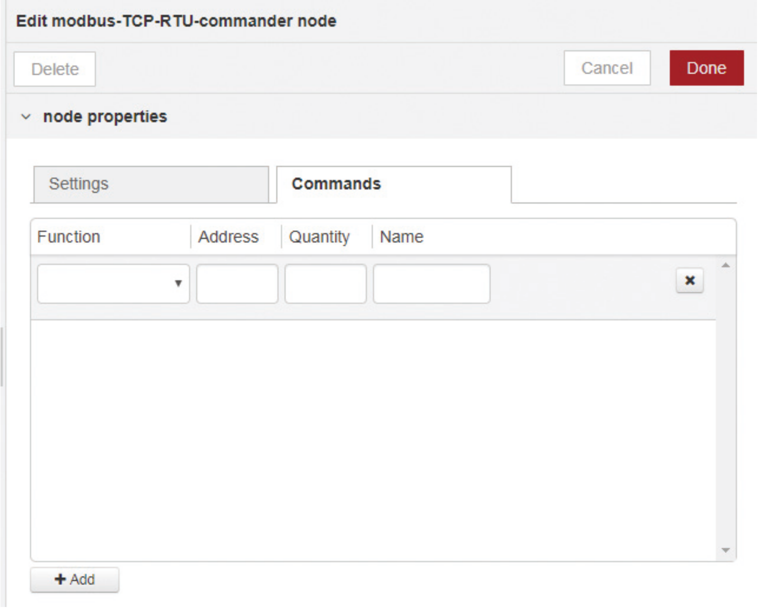
Server: Add new modbus-client...

Repeat: Infinite

Delay(ms): 100

- Server: Configure a **Modbus TCP** or **Serial** communication.
- Repeat: (Infinite | Once) — Select **Infinite** to set up a polling rate or select **Once** to just run once.
- Delay(ms): Time to wait before sending the next transmission.

Set up one or more Modbus request(s) and give a name for each request

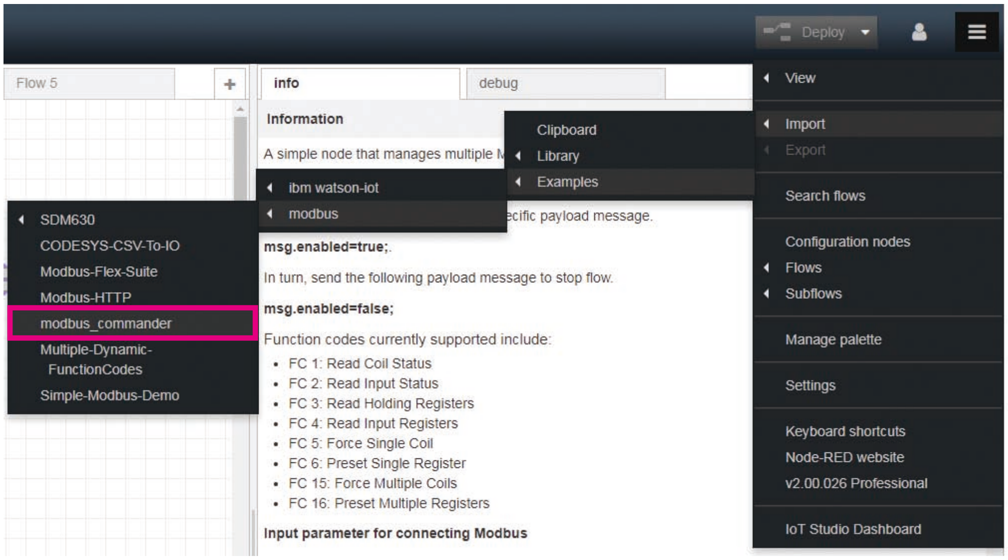


- Function: fc (1|2|3|4|5|6|15|16)
- Address: Start address (0:65535)
- Quantity: (1:65535) Quantity of coils/inputs/registers to be read or written from the start address.
- Name: **Must be defined**. The tag of the connection is callable by other nodes.

  Note: Leaving this field blank will keep other nodes from transporting values.

Sample flow

- To import the sample flow, you can choose **Import** from the menu on the upper right corner and **select modbus_commander** from the **Examples->node red-nexcom-modbus** option in the sub-menu of **Import**.



Information

A simple node that manages multiple M

ecific payload message.

msg.enabled=true;.

In turn, send the following payload message to stop flow.

msg.enabled=false;

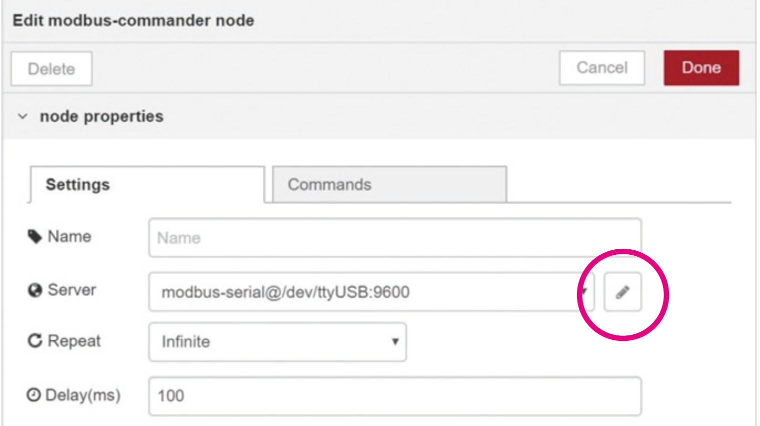
Function codes currently supported include:

- FC 1: Read Coil Status
- FC 2: Read Input Status
- FC 3: Read Holding Registers
- FC 4: Read Input Registers
- FC 5: Force Single Coil
- FC 6: Preset Single Register
- FC 15: Force Multiple Coils
- FC 16: Preset Multiple Registers

Input parameter for connecting Modbus

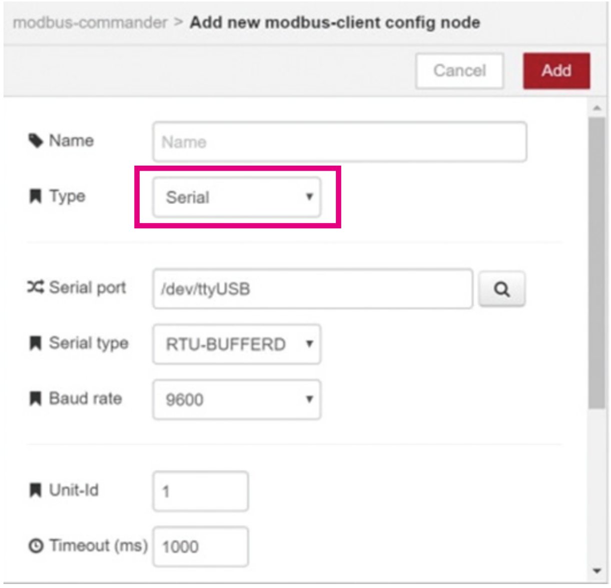
3.8.1 Modbus RTU

Use the modbus commander node to send the msg.payload to a serial interface and expect a response. The response will be output in the msg.payload as a buffer, so users may need to use ".toString()" for conversion.



Edit modbus-commander node

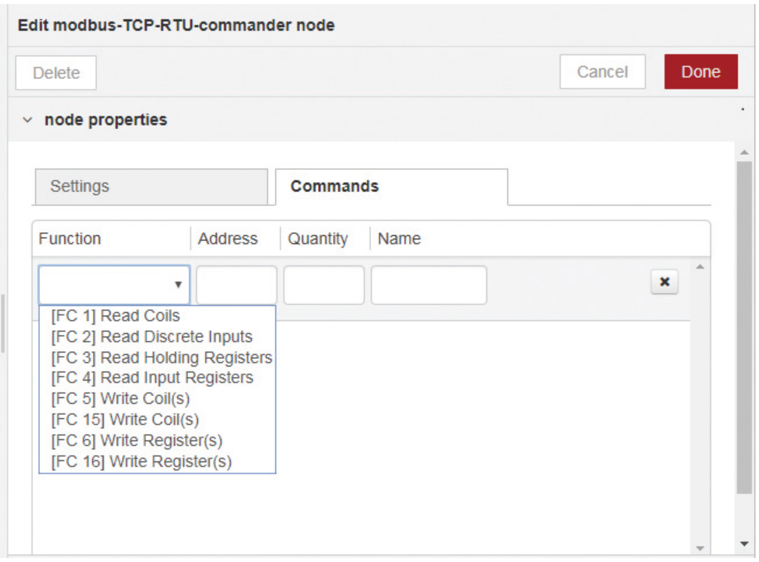
Name: Name
Server: modbus-serial@/dev/ttyUSB:9600
Repeat: Infinite
Delay(ms): 100



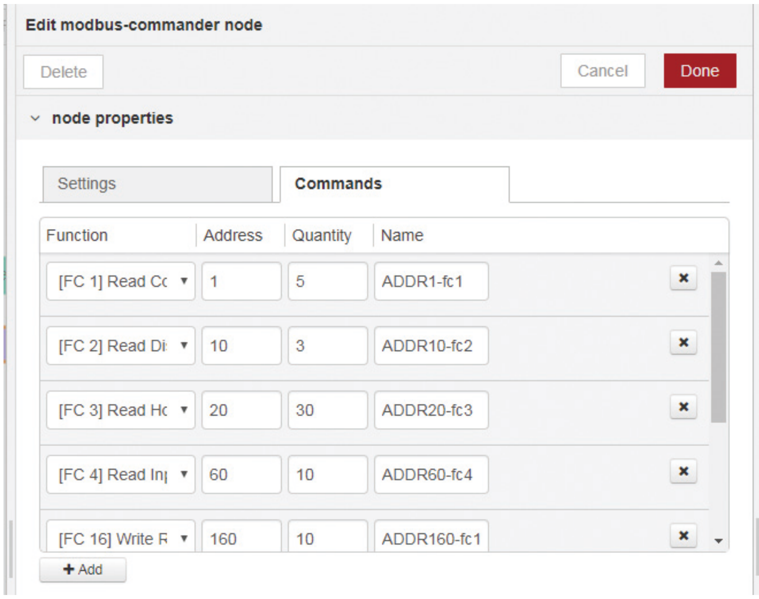
modbus-commander > Add new modbus-client config node

Name: Name
Type: Serial
Serial port: /dev/ttyUSB
Serial type: RTU-BUFFERD
Baud rate: 9600
Unit-Id: 1
Timeout (ms): 1000

Item	Description	
Serial Port	The local interface of serial input.	
	Settings	Baud Rate, Data Bits, Parity, Stop Bits: 57600, 8, None, 1



- [FC 1] Read Coils
- [FC 2] Read Discrete Inputs
- [FC 3] Read Holding Registers
- [FC 4] Read Input Registers
- [FC 5] Write Coil(s)
- [FC 15] Write Coil(s)
- [FC 6] Write Register(s)
- [FC 16] Write Register(s)



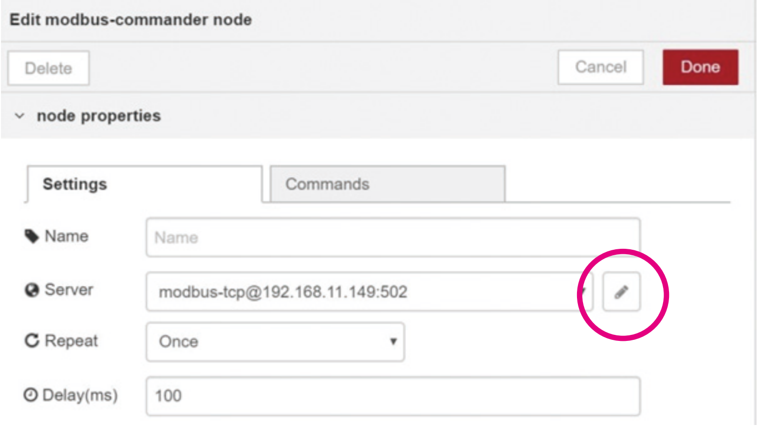
Function	Address	Quantity	Name
[FC 1] Read Cc	1	5	ADDR1-fc1
[FC 2] Read Di	10	3	ADDR10-fc2
[FC 3] Read Hc	20	30	ADDR20-fc3
[FC 4] Read Inj	60	10	ADDR60-fc4
[FC 16] Write F	160	10	ADDR160-fc1

Item	Option	Description
FC (Function Code)	**Read Coils**	Read single bit. This command reads the ON/OFF status of coils (0x reference address) in the slave/server.
	Read Discrete Inputs	Read single bit. This command reads the ON/OFF status of discrete inputs in the slave/server.
	Read Holding Registers	Read 16-bit word. This command reads the binary contents of holding registers (4x reference address) in the slave device.
	Read Input Registers	Read 16-bit word. This command reads the binary contents of input registers (3x reference address) in the slave device.

Item	Option	Description
FC (Function Code)	**Write Coil(s)**	Write a single bit. Simultaneously forces a series of coils (0x reference address) either ON/OFF.
	Write Register(s)	Write a 16-bit word. This command presets a single holding register (4x reference address) to a specific value. The Preset Multiple Registers normal response message returns the slave address, function code, starting register reference, and the number of registers preset, after the register contents have been preset.
Address	The value from 0 to 65535.	
Quantity	The value from 0 to 65535.	
Name	The tag of the connection callable by other nodes. **Note:** Leaving this field blank will keep other nodes from transporting values.	

3.8.2 Modbus TCP

Use the modbus commander node to send the payload to a Modbus TCP port and expect a response. The response will be output in msg.payload as a buffer, so you may need to use ".toString()" for conversion.

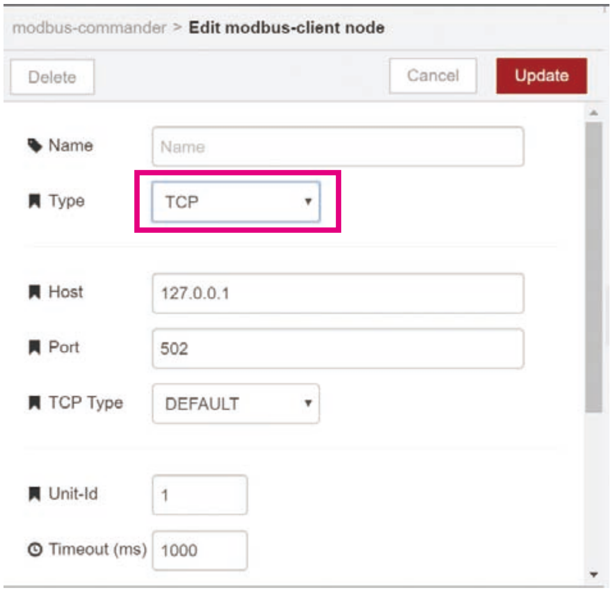


Name: Name

Server: modbus-tcp@192.168.11.149:502

Repeat: Once

Delay(ms): 100



Name: Name

Type: TCP

Host: 127.0.0.1

Port: 502

TCP Type: DEFAULT

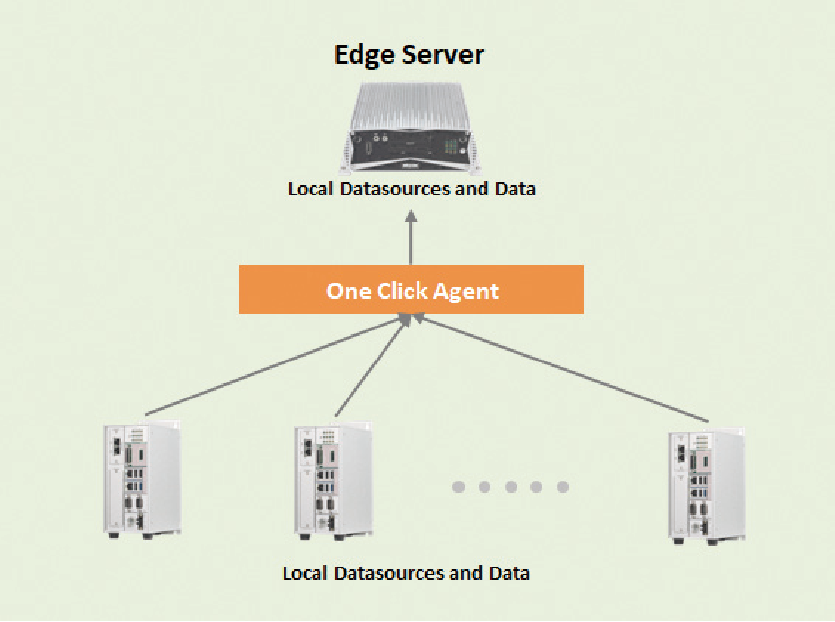
Unit-Id: 1

Timeout (ms): 1000

Item	Description
Host	The IP address to access.
Port	The port number of the IP address to access.

3.9 One Click Deploy

3.9.1 One-Click Deploy to Edge Server

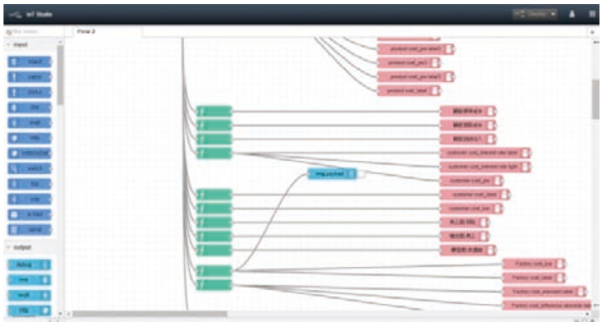
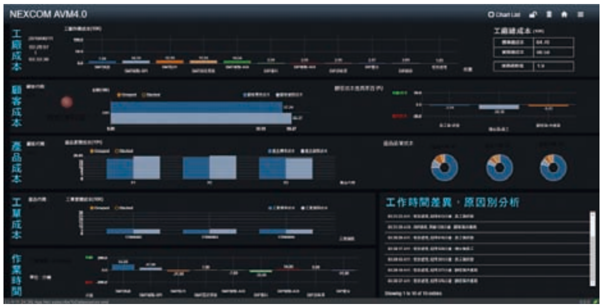
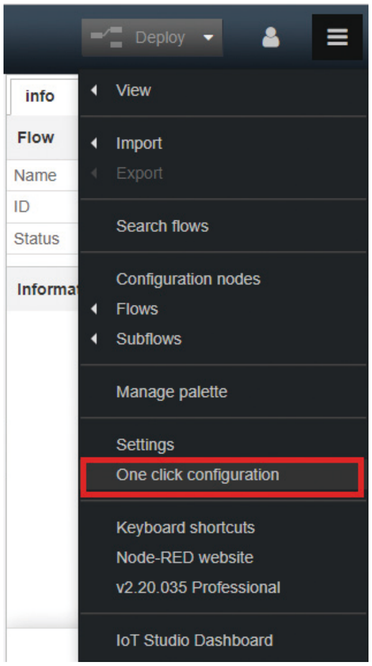


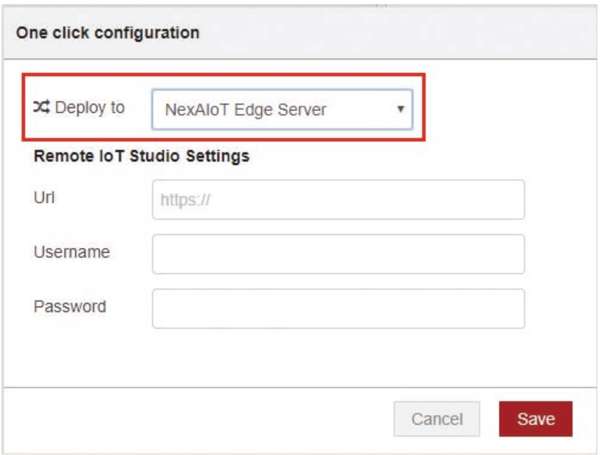
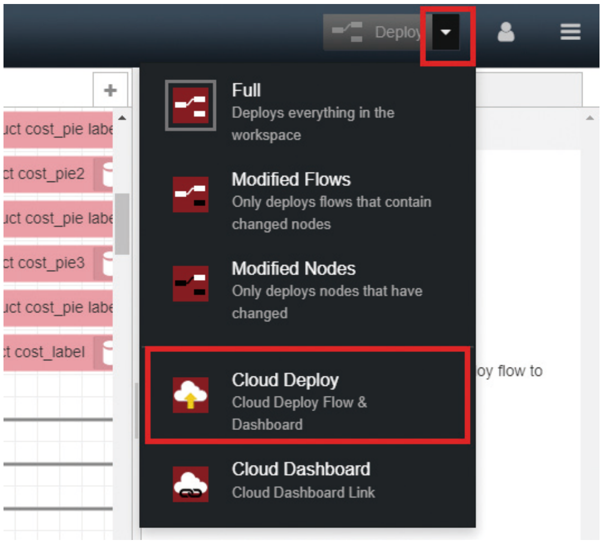
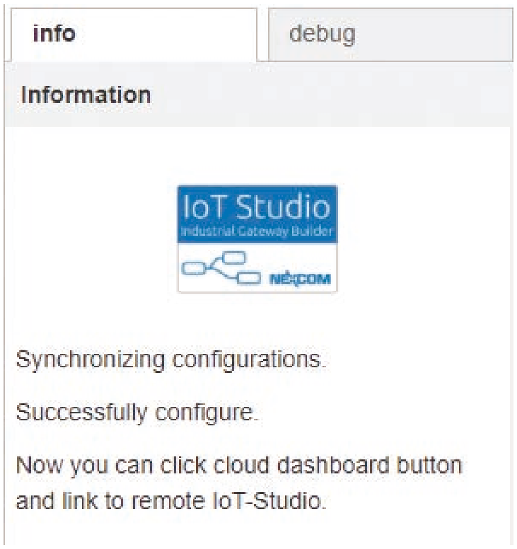
Prerequisites

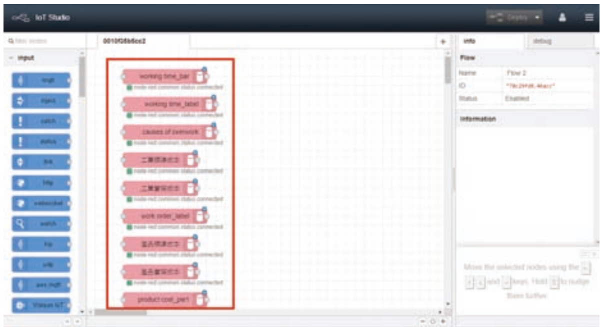
- Prepare a gateway and edge server with either the HyperX or IoT Studio application installed. A **Standard License** is required for IoT Studio.

Restrictions

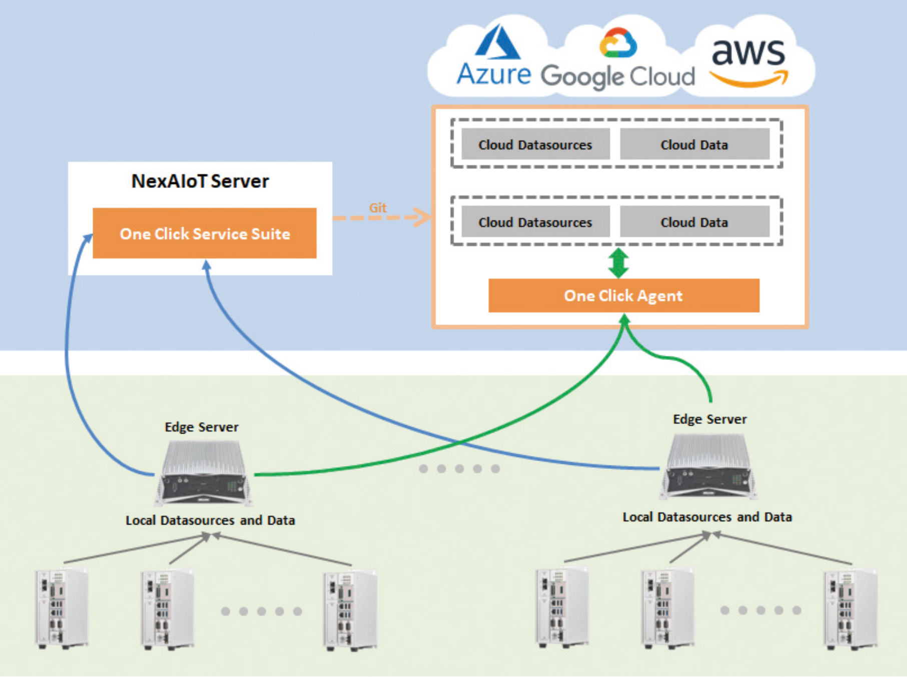
- A flow that has already been deployed cannot be deployed again to another device/cloud.
- The deployed node (iotdatasource) from the gateway can be reused by the edge server. However, the node which was additionally added by the edge server on the deployed flow will be overwritten when the gateway deploys again.

Step	Description	Screenshot
1	Prepare data flow and dashboard in your local gateway.	 
2	Click **One click configuration** from the triple bar menu ≡ in the upper-right corner.	 Deploy info Flow Name ID Status Informa View Import Export Search flows Configuration nodes Flows Subflows Manage palette Settings One click configuration Keyboard shortcuts Node-RED website v2.20.035 Professional IoT Studio Dashboard

Step	Description	Screenshot
3	Fill in the following information: • Choose **NexAIoT Edge Server** under the **Deploy to** drop-down list. • Enter the edge server URL for the IoT Studio **Url**. • Fill in **Username** and **Password** for the remote IoT Studio. • Click **Save**.	 One click configuration Deploy to: NexAIoT Edge Server Remote IoT Studio Settings Url: https:// Username: Password: Cancel Save
4	Expand the **Deploy menu** by clicking the inverted triangle and then click **Cloud Deploy** to push the data sources and dashboard to the edge server.	 Deploy Full Deploys everything in the workspace Modified Flows Only deploys flows that contain changed nodes Modified Nodes Only deploys nodes that have changed Cloud Deploy Cloud Deploy Flow & Dashboard Cloud Dashboard Cloud Dashboard Link ıct cost_pie labe ct cost_pie2 ıct cost_pie labe ct cost_pie3 ıct cost_pie labe t cost_label oy flow to
5	If deployment is successful, you will see this screen shown in the information field.	 info debug Information IoT Studio Industrial Gateway Builder NEXCOM Synchronizing configurations. Successfully configure. Now you can click cloud dashboard button and link to remote IoT-Studio.

Step	Description	Screenshot
6	Go to the IoT Studio dashboard of the edge server. You'll see that the data sources have been deployed. **Note:** The deployed flow is named after the local device MAC ID's last 6 digits.	 IoT Studio Deploy input 0010f26b6ce2 info debug Flow Name Flow 2 Status Enabled Information working time_bar working time_label causes of overwork work order_label product cost_part

3.9.2 One-Click Deploy to Cloud

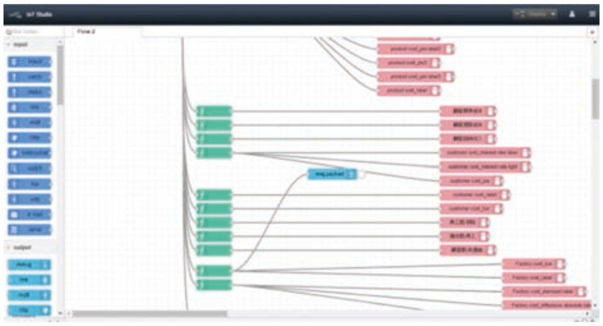
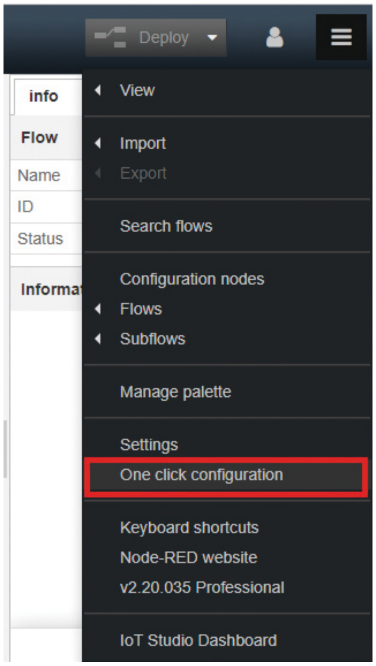


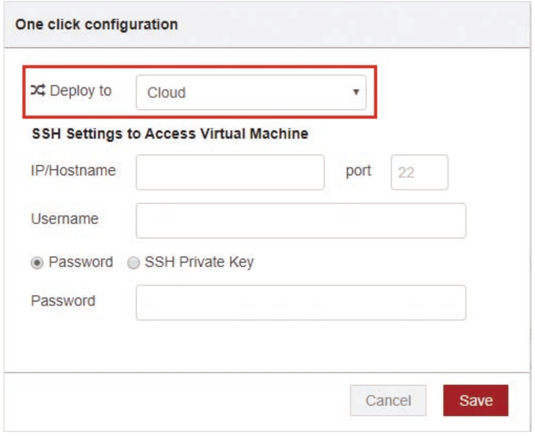
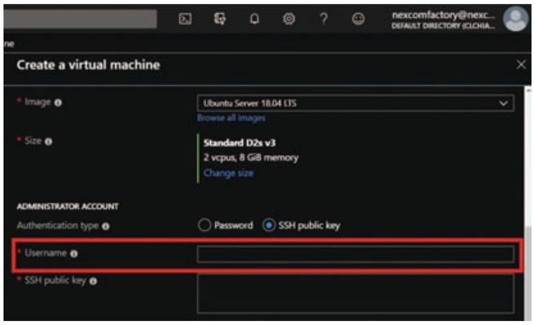
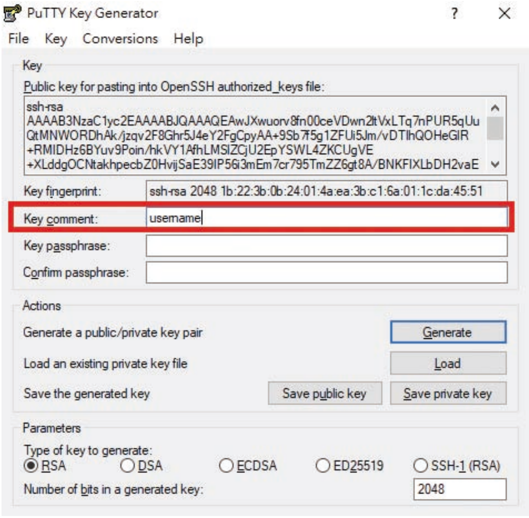
Prerequisites

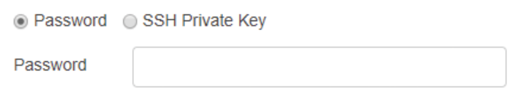
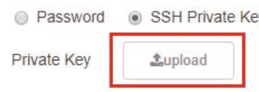
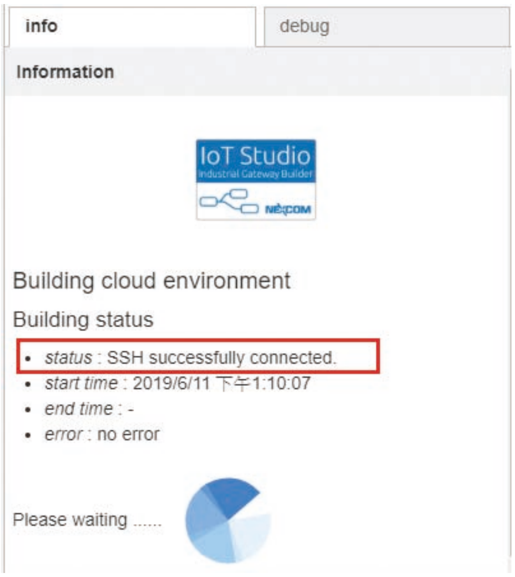
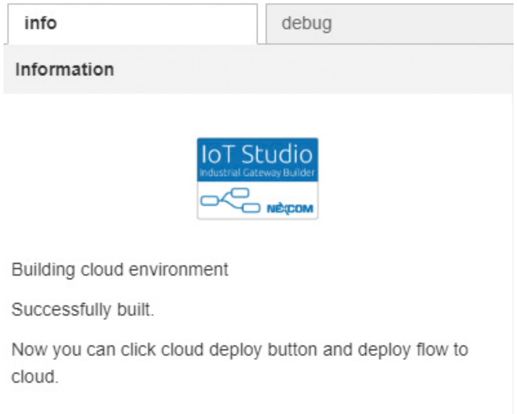
- Prepare a gateway or edge server with either the HyperX or IoT Studio application installed. A **Professional License** is required for IoT Studio.

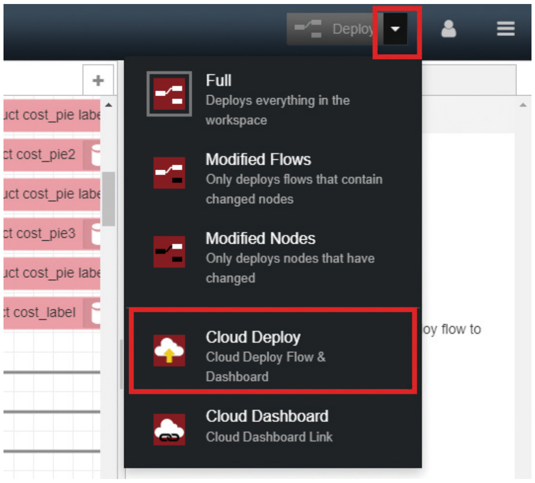
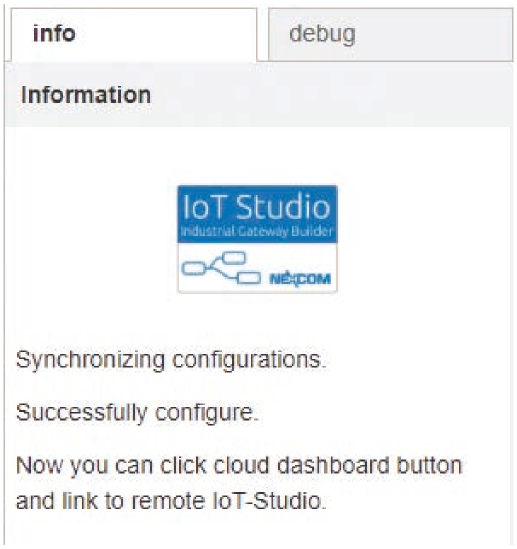
Restrictions

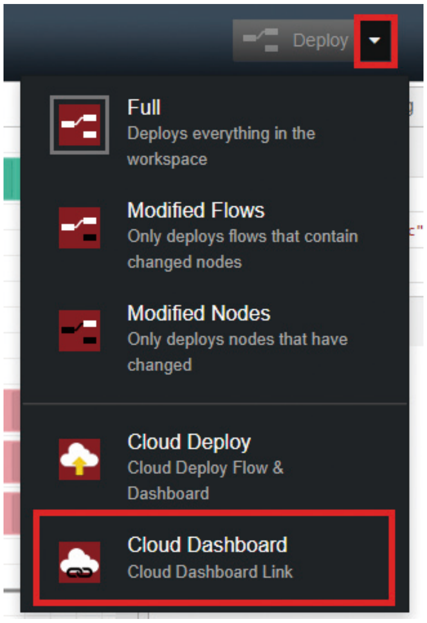
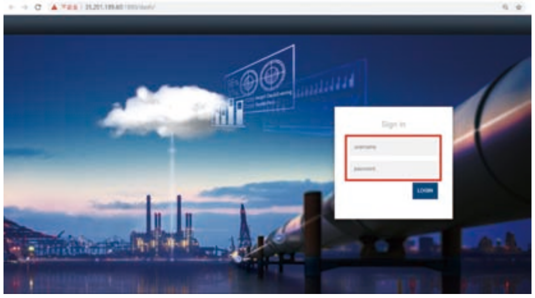
- A flow that has already been deployed cannot be deployed again to another device/cloud.
- The deployed node (iotdatasource) from the gateway can be reused by the edge server. However, the node which was additionally added by the edge server on the deployed flow will be overwritten when the gateway deploys again.

Step	Description	Screenshot
1	Create a VM for IoT Studio in Target Cloud. Depending on the cloud you use, please refer to the following appendix to create the virtual machine. • For Google – Appendix A "Create a Virtual Machine for IoT Studio in Google Cloud" • For Azure – Appendix B "Create a Virtual Machine for IoT Studio in Azure Cloud" • For AWS – Appendix C "Create a Virtual Machine for IoT Studio in AWS Cloud"	
2	Prepare data flow and dashboard in your local gateway.	
3	Click **One Click Configuration** from the triple bar menu ≡ in the upper-right corner.	 Deploy info Flow Name ID Status Informa View Import Export Search flows Configuration nodes Flows Subflows Manage palette Settings One click configuration Keyboard shortcuts Node-RED website v2.20.035 Professional IoT Studio Dashboard

Step	Description	Screenshot
4	Fill in the following information: • Choose **Cloud** under the **Deploy to** drop-down list. • Fill in the **IP/Hostname** of the VM you created in step 1. • Type in the **Username** of the VM. Here are the corresponding usernames for each cloud's VM: – **AWS**: always "**ubuntu**" – **Azure:** the one that the user chose when creating VM – **Google Cloud:** the key comment the user chose when generating SSH key • For authentication, you can choose from typing the **password** directly or choosing the **SSH private key** to log in. If you use the SSH private key, please **upload** the private key (.ppk) file. • Click **Save** to store the configuration and construct IoT Studio in the VM.	 One click configuration Deploy to: Cloud SSH Settings to Access Virtual Machine IP/Hostname: port: 22 Username: ● Password ○ SSH Private Key Password: Cancel Save • **Username** **Azure**  nexcomfactory@nexc... DEFAULT DIRECTORY (CLCHIA... Create a virtual machine * Image: Ubuntu Server 18.04 LTS Browse all images * Size: Standard D2s v3 2 vcpus, 8 GiB memory Change size ADMINISTRATOR ACCOUNT Authentication type: ○ Password ● SSH public key * Username: * SSH public key: **Google Cloud**  PuTTY Key Generator File Key Conversions Help Key Public key for pasting into OpenSSH authorized_keys file: ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAwJXwuorv8fn00ceVDwn2ltVxLTq7nPUR5qUu QtMNWORDhAk/jzqv2F8Ghr5J4eY2FgCpyAA+9Sb7f5g1ZFUi5Jm/vDTIhQOHeGIR +RMIDHz6BYuv9Poin/hkVY1AfhLMSIZCjU2EpYSWL4ZKCUgVE +XLddgOCNtakhpecbZ0HvijSaE39IP56i3mEm7cr795TmZZ6gt8A/BNKFIXLbDH2vaE Key fingerprint: ssh-rsa 2048 1b:22:3b:0b:24:01:4a:ea:3b:c1:6a:01:1c:da:45:51 Key comment: username Key passphrase: Confirm passphrase: Actions Generate a public/private key pair — Generate Load an existing private key file — Load Save the generated key — Save public key, Save private key Parameters Type of key to generate: ● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA) Number of bits in a generated key: 2048

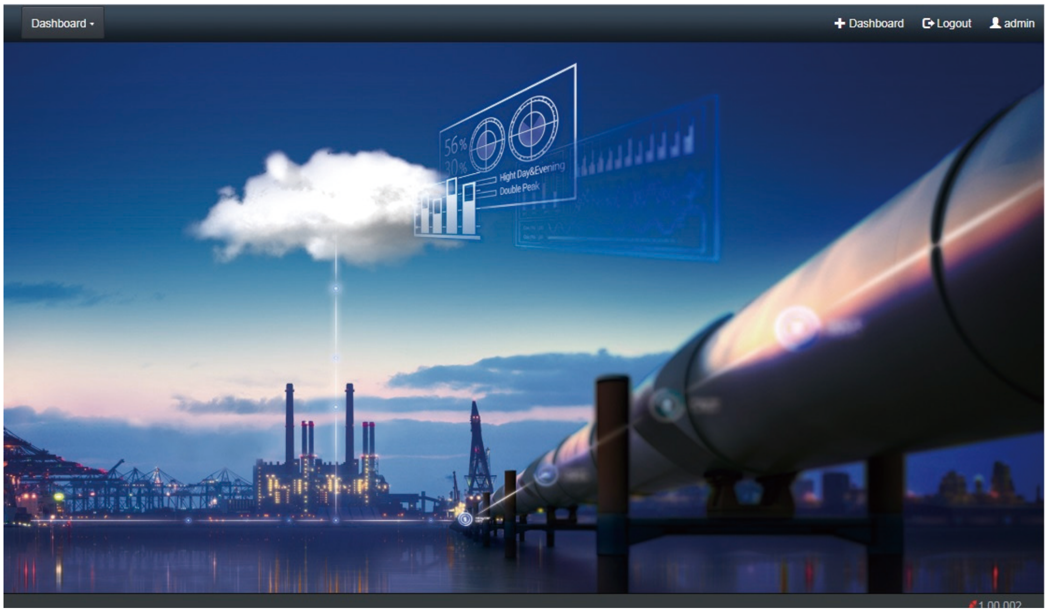
Step	Description	Screenshot
		• **Authentication** **Use password**  ◉ Password ○ SSH Private Key Password **Use SSH key**  ○ Password ◉ SSH Private Key Private Key
5	After completing the last step, the One-Click-Service-Suite will start constructing IoT Studio in the cloud's VM. Please wait about **5 to 10 minutes** (depending on VM specification and Internet speed) to complete. You can view the current progress from the information board on the right of the page.	 info debug IoT Studio Industrial Gateway Builder NEXCOM Information Building cloud environment Building status • *status* : SSH successfully connected. • *start time* : 2019/6/11 下午1:10:07 • *end time* : - • *error* : no error Please waiting ......
6	After you finish building the environment, you will see this information.	 info debug IoT Studio Industrial Gateway Builder NEXCOM Information Building cloud environment Successfully built. Now you can click cloud deploy button and deploy flow to cloud.

Step	Description	Screenshot
7	Expand the **Deploy** menu by clicking on the inverted triangle and then click **Cloud Deploy** to push the data sources and dashboard to the cloud.	 Deploy Full Deploys everything in the workspace Modified Flows Only deploys flows that contain changed nodes Modified Nodes Only deploys nodes that have changed Cloud Deploy Cloud Deploy Flow & Dashboard Cloud Dashboard Cloud Dashboard Link uct cost_pie label ct cost_pie2 uct cost_pie label ct cost_pie3 uct cost_pie label t cost_label oy flow to
8	If deployment is successful, you will see this screen shown in the information field.	 info debug Information IoT Studio Industrial Gateway Builder NEXCOM Synchronizing configurations. Successfully configure. Now you can click cloud dashboard button and link to remote IoT-Studio.

Step	Description	Screenshot
9	Go to the cloud's dashboard by clicking **Cloud Dashboard** in the **Deploy** menu.	 Deploy Full Deploys everything in the workspace Modified Flows Only deploys flows that contain changed nodes Modified Nodes Only deploys nodes that have changed Cloud Deploy Cloud Deploy Flow & Dashboard Cloud Dashboard Cloud Dashboard Link
10	The username and password should both be "**admin**".	

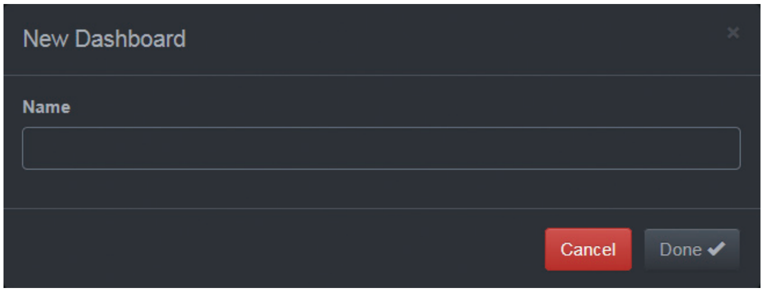
Chapter 4: Dashboard

This chapter introduces the user interface and the basic operation of NexAIoT IoT Studio Dashboard. Once you log onto NexAIoT IoT Studio Dashboard with your browser, you will see the page as shown below.



4.1 Create Your Dashboard

1. Click **+ Dashboard** on the top right of the page.
2. Fill a desired name in the **Name** field, and click **Done**.

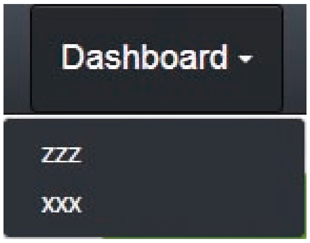

New Dashboard

Name

4.2 Select Your Dashboard

Once you have created a dashboard, you can select the dashboard anytime in the same web page.

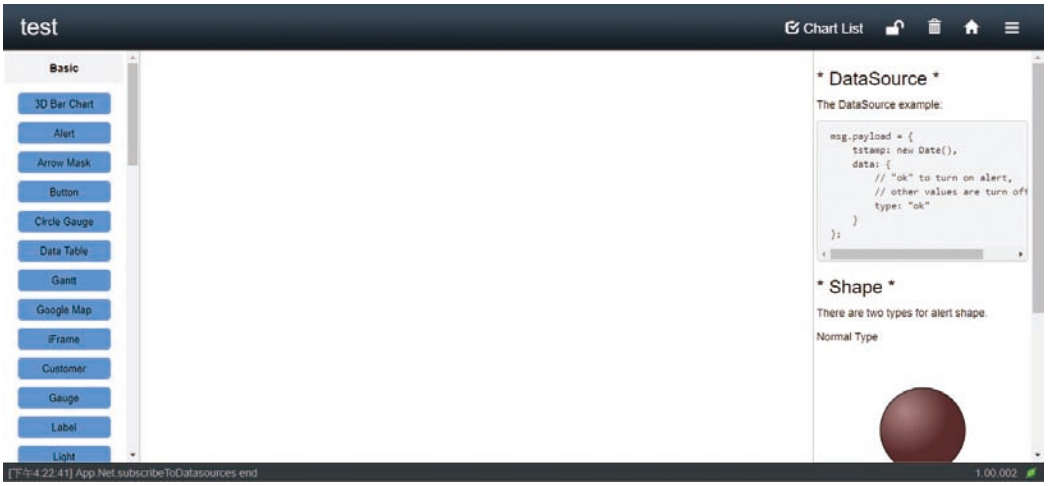
1. Click Dashboard ▾ on the top left of the page.
2. Click on the name in the drop-down list for your desired dashboard.



4.3 Edit Your Dashboard

In your dashboard, click ☐ Chart List on the top right of the page to switch on or off the sidebars of available charts and information of the charts. Refer to the next section for more information about the charts.

- Click 🔓 to keep your dashboard from being altered. Click 🔒 to unlock.
- Click 🗑 to abandon the current dashboard.
- Click 🏠 to return to the main page.



* DataSource *

The DataSource example:

```
msg.payload = {
    tstamp: new Date(),
    data: {
        // "ok" to turn on alert,
        // other values are turn off
        type: "ok"
    }
};
```

* Shape *

There are two types for alert shape.

Normal Type

```
[下午4:22:41] App.Net.subscribeToDatasources end
```

4.4 Available Charts

Simply drag and drop your desired chart into the workspace, and the chart will appear on the top left of the workspace.

Select and drag the black square at the bottom right of the chart to resize it. Select and drag the title at the top of the plug to move it to anywhere in the workspace.

- Click 🗑 to abandon the chart.
- Click ✎ to set the name of the chart and assign its data source.

As the window pops up, refer to the steps below:

1. Fill a desired name in the **Name** field.
2. Select a datasource from the **Datasources** drop-down menu. You should see the name of the IoT datasource nodes you created in your flow.
3. Click **Done** and you should see the chart that reflects with your datasource.

- Click ⧉ to duplicate the chart.

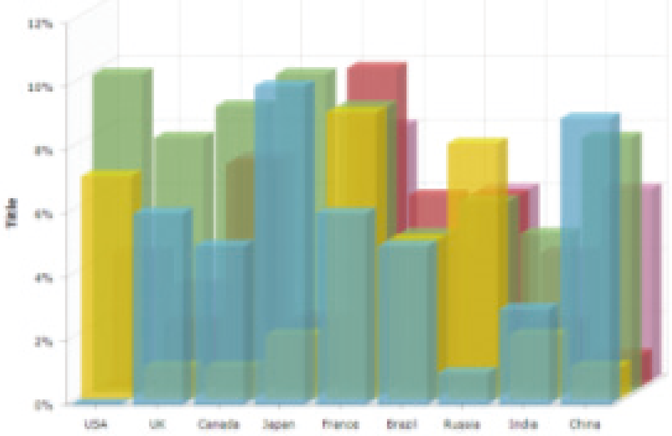
Click and drag a rectangle around the desired charts to select multiple charts at the same time. When multiple charts are selected, on the top of the dashboard, you can perform the following:

Click to align the charts to the left.	Click to align the charts to the bottom.
Click to align the charts to the vertical center.	Click ⋯ to set equal spacings among charts horizontally.
Click to align the charts to the right.	Click ⋮ to set equal spacings among charts vertically.
Click to align the charts to the top.	Click ⤢ to set charts equally large.
Click to align the charts to the horizontal center.	Click ⤡ to set charts equally small.

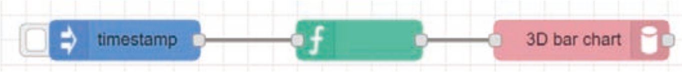
4.4.1 Basic

4.4.1.1 3D Bar Chart

Use a 3D bar chart to measure your data off in a stereoscopic front.



IoT Dataflow



To use a 3D bar chart in the dashboard, make sure the elements of the variable in the data object are set as shown before sending to the **iot datasource** node.

```
var data = [
  {
    "category": "USA",
    "zone1": Math.round(Math.random() * 10),
    "zone2": Math.round(Math.random() * 10),
    "zone3": Math.round(Math.random() * 10),
    "zone4": Math.round(Math.random() * 10),
    "zone5": Math.round(Math.random() * 10)
  }, {
    "category": "UK",
    "zone1": Math.round(Math.random() * 10),
    "zone2": Math.round(Math.random() * 10),
    "zone3": Math.round(Math.random() * 10),
    "zone4": Math.round(Math.random() * 10),
    "zone5": Math.round(Math.random() * 10)
  }, {
    "category": "Canada",
    "zone1": Math.round(Math.random() * 10),
    "zone2": Math.round(Math.random() * 10),
    "zone3": Math.round(Math.random() * 10),
    "zone4": Math.round(Math.random() * 10),
    "zone5": Math.round(Math.random() * 10)
  }];

msg.payload = {
  tstamp: new Date().getTime(),
  data: data
};
return msg;
```

Note:

- The value after “category” will apply to **Tool Tip** in **Edit Chart**.
- The value after “zone 1“, “zone2”, and so on will present in colors as set in **Value Colors** in **Edit Chart** separated by colons.

You can configure fields as shown below while selecting their data source.

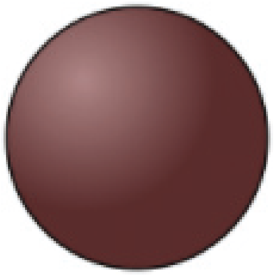
Name: The title of the chart.
Show: Check to show the title on the dashboard.
Title: The name to measure the Y axis.

Configuration

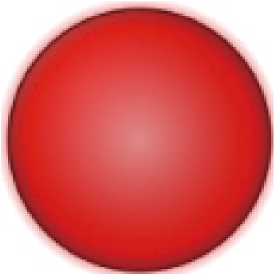
Position: The direction for measurement.	**Depth 3D:** The thickness of the bars.
Unit: The unit of measurement.	**Angle:** The inclination to the front face and the left face of the bars.
Tool Tip: The tags presented on the bars with the respective definitions and given values.	**Column Opacity:** The level of transparency of the bars.
Text Color: The html color codes for the display texts.	**Column Width:** The width of the bars.
Value Colors: The html color codes for the respective bars separated by colons.	**Legend:** Check to show the legend.
Axis Color: The html color code for the axis.	**Value Maximum:** The maximum value available on Y axis.
Grid Color: The html color code for the grid.	**Value Minimum:** The minimum value available on Y axis.
Rotate: Check to set the bars horizontal and uncheck to set the bars vertical.	

4.4.1.2 Alert

You can show an alert sign based on the data source.

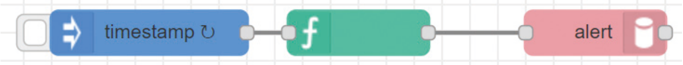


Disabled



Enabled

IoT Dataflow



To enable an alert in the dashboard, in **msg.payload**, make sure that the data type of a variable is set to character and stored with “**ok**” as shown before sending to the **iot datasource** node.

```
msg.payload = {
   tstamp: new Date(),
   data: {
     // “ok” to turn on alert,
     // other values are turn off alert
     type: “ok”
   }
 };
return msg;
```

To disable an alert in the dashboard, in **msg.payload**, make sure that the data type of a variable is set to character and stored with anything but “**ok**” as shown before sending to the **iot datasource** node.

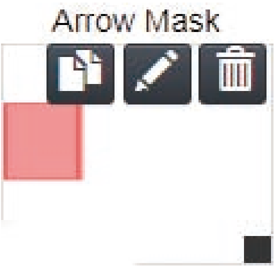
```
msg.payload = {
   tstamp: new Date(),
   data: {
     // “ok” to turn on alert,
     // other values are turn off alert
     type: “ko”
   }
 };
return msg;
```

You can configure fields as shown below while selecting their data source.

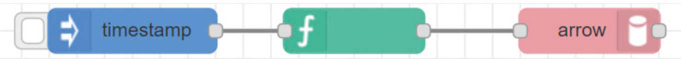
Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.1.3 Arrow Mask

Use an arrow mask to measure your data off in a bar from a preferred direction.



IoT Dataflow



```
msg.payload = {
    tstamp: new Date(),
    data: {
        switch   : “on”,
        color    : “#0000FF”,
        opacity  : “0.3”,
        duration : “2s”
    }
 };
return msg;
```

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.

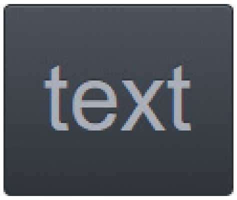
Show: Check to show the title on the dashboard.

Datasources: Assign one or multiple data sources as the target nodes for data reception. *No message payload is required to send to the* ***iot datasource*** *node for use with the arrow mask.*

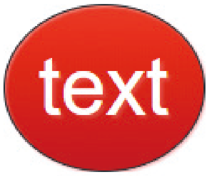
Type: The direction of the arrow mask moves.	**Mask Duration:** The time the arrow mask spends on the chart. The bigger the value is, the slower the arrow mask moves.
Mask Color: The html color code for the arrow mask.	**Mask Width:** The width of the arrow mask. The bigger the value is, the larger the arrow mask is.
Mask Opacity: The level of transparency of the arrow mask. For a value between 0 and 1 with decimal point, the bigger the value is, the more opaque the arrow mask is.	

4.4.1.4 Button

Use a button to trigger actions after the data sources.

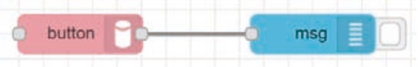


Rectangle



Round

IoT Dataflow



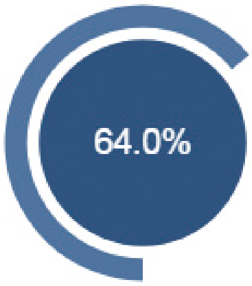
To use a button as a trigger in the dashboard, assign a data source in its **Edit Chart**, and connect the other nodes, which are the actions you will trigger, to the right end of the data source in IoT Studio workspace as shown above.

You can configure the fields as shown while assigning their data source.

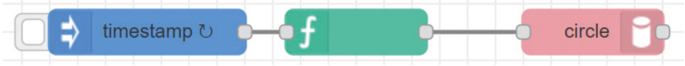
Name: The title of the chart.	**Round Button:** Check to select the round button or uncheck to select the rectangular button.
Show: Check to show the title on the dashboard.	**Extend Style:** Define your own applicable CSS styling here.
Text: The content to be shown on the button.	**Type:** Choose from **Button/Flat Area/ Hyperlink**. If set to **Flat Area**, no action will take place while the button is pressed.
Font Size: The size of the content.	**Hyperlink:** Input the URL to launch in the browser once the button is pressed.

4.4.1.5 Circle Gauge

Use a circle gauge to measure your data off in a circle.



IoT Dataflow



To use a circle gauge in the dashboard, make sure the data type of a variable is set to numeric as shown before sending to the **iot datasource** node.

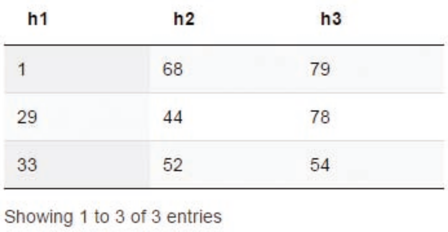
```
var value = Math.floor( Math.random() * 100 );
 msg.payload = {
   tstamp: new Date().getTime(),
   data: value
 };
return msg;
```

You can configure the fields as shown while assigning their data source.

Name: The title of the chart.	**Text:** The color code of the text on the circle area.
Show: Check to show the title on the dashboard.	**Value Unit:** The unit of the measured value.
Color **Inner:** The color code of progressive arc. **outer:** The color code of the circle area.	**Value Range** **Min:** The minimum value allowed. **Max:** The maximum value allowed.

4.4.1.6 Data Table

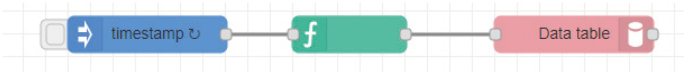
You can use a data table to present your data as if it was in a spreadsheet.



h1	h2	h3
1	68	79
29	44	78
33	52	54

Showing 1 to 3 of 3 entries

IoT Dataflow



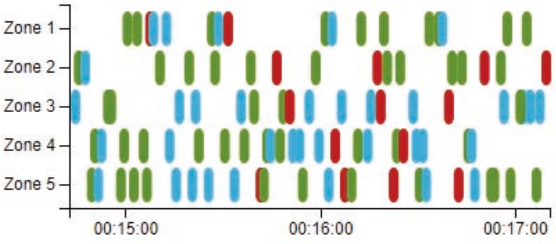
To use a data table in the dashboard, make sure the payload is set as shown on the right before sending to the **iot datasource** node.

```
msg.payload = {
    tstamp: new Date().getTime(),
    data: [
          [1,68,79],
          [29,44,78],
          [33,52,54]
       ]
 };
 return msg;
```

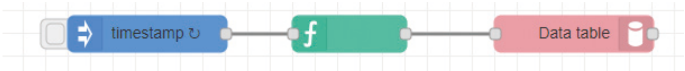
Name: The title of the chart.	**Info Text Color:** The html color codes for the information texts.
Show: Check to show the title on the dashboard.	**Data Area Transparent:** Check to set the table to the background.
Header: Texts in the header field apply to the headlines of the data table. The format is header1, header2, header3, etc.	**Ordering:** Check to enable sorting of first column data.
Chart Title Color: Set the chart title text color.	**First Column Data Sorting Order:** Use the drop-down menu to select ascending or descending order for sorting.
Hide Information: Check to keep information from showing at the bottom row of the table.	**Row Font Size:** The size of the content.
Header Text Color: The html color codes for the header texts.	**Row Condition:** Set the row css to depend on condition.
Body Text Color: The html color codes for the body texts.	**Row Count Max:** The available rows in the table for the last available pieces of data.

4.4.1.7 Gantt

Use a gantt to measure your data off in a Gantt chart.



IoT Dataflow



To use a gantt in the dashboard, make sure the name:value pairs in the data object are set as shown before sending to the **iot datasource** node.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.

Show: Check to show the title on the dashboard.

Task Names: Individual names of jobs to list in the Gantt chart separated by colons.

Task Status: Individual status descriptions paired with the respective html color code separated by colons.

Text Color: The html color code for texts around the chart.

Tick Format: Set the chart’s X axis value format to follow the d3 time format.

Time Domain Mode: The chart’s X axis show mode.

Fixed Start Time: For Fixed Mode only. The fixed start time setting.

Fixed End Time: For Fixed Mode only. The fixed end time setting.

Tick Count: Show the X axis tick count.

Max Data Count: Show the maximum data count setting.

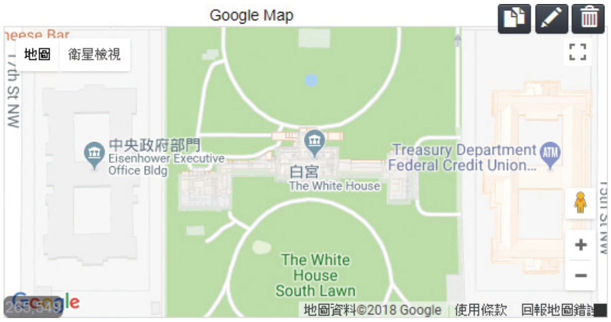
```
function getRandomInt(max) {
 return Math.floor(Math.random() * Math.
floor(max));
}
function getStatus() {
  switch (getRandomInt(3)) {
    case 0:
      return “SUCCEEDED”;
      break;
    case 1:
      return “FAILED”;
      break;
    default:
      return “RUNNING”
  }
  return “RUNNING”;
}

var now = new Date();
var eDate = new Date();
eDate.setSeconds(eDate.getSeconds() + 3);

var out = {
    “startDate” : now.toString(),
    “endDate” : eDate.toString(),
    “taskName” : “Zone “ + (getRandom-
Int(5)+1),
    “status”    : getStatus()
  }

msg.payload = {
  tstamp: now.getTime(),
  data: JSON.stringify(out)
};
return msg;
```

Note:

- Both "startDate" & "endDate" are in JavaScript default date format in full text string.
- "taskName" & "status" have to be filled with corresponding values set in configure fields of the edit chart.

4.4.1.8 Google Maps

Use Google Maps to disclose your desired location information to your dashboard.



To apply Google Maps in the dashboard, assign a set of geographic coordinate, a scale value, and a Google API Key in the respective fields to configure the subject matter.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.	**Longitude:** The angular distance of a place east or west of the meridian at Greenwich, England expressed in decimal degrees.
Show: Check to show the title on the dashboard.	**Zoom:** The scale of the map to display.
Latitude: The angular distance of a place north or south of the earth's equator expressed in decimal degrees.	**Google API Key:** The application programming interface key to grant your access to Google Maps.

4.4.1.9 iFrame

Use an iFrame to bring your desired web content to your dashboard.

To use an iFrame in the dashboard, assign any data source in its **Edit Chart**. In the **URL** field, enter the address of your web content.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

Note:
Viewing restrictions vary from hosts. Not all subjects will be presented when viewing from remote hosts.

4.4.1.10 Customer

Use customer figures to make up your dashboard.

To use a customer in the dashboard, assign any data source in its **Edit Chart**. In the **Image Files** field, enter the full file name of the desired image.

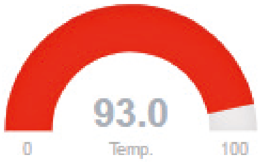
You can configure fields as shown below while selecting their data source.

Name: The title of the chart.	**Imagesources** **Select Image:** Click to select a desired image in the drop-down list.
Show: Check to show the title on the dashboard.	**Custom Upload:** Click to launch File Explorer and select an image file to upload to the dedicated folder.
	Background: Check to set the selected image as a maximized image in the background of the dashboard.

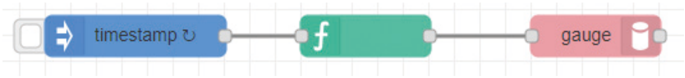
Note:
The desired image file should be stored in a dedicated folder such as {IoT Studio installed folder}\node_modules\node-red-contrib-graphs\static\images.

4.4.1.11 Gauge

Use a gauge to measure your data off in a semicircle.



IoT Dataflow



To use a gauge in the dashboard, make sure the data type of a variable is set to numeric as shown before sending to the **iot datasource** node.

```
var value = Math.floor( Math.random() * 100 );
msg.payload = {
            tstamp: new Date().getTime(),
            data: value
};
return msg;
```

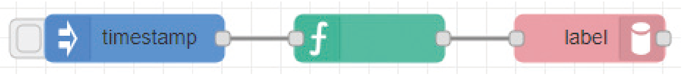
You can configure fields as shown below while selecting their data source.

Name: The title of the chart.	**Configure** **Decimals:** Put the number of allowed digits to the right of the decimal point here.
Show: Check to show the title on the dashboard.	**Symbol:** Put the unit of measure here.
Title: The title on top of the chart.	**Middle Mode:** Check to set the gauge progress from the middle of the arc.
Label: The name for the measurement of the gauge.	**Border Color:** The html color code for the border around the arc.
Value range **min:** The minimum value allowed. **max:** The maximum value allowed.	**Text Color:** The html color code for texts of the chart.

4.4.1.12 Label

Use a label to make captions for the dashboard.

IoT Dataflow



To use a label, the payload can be set as shown on the right before sending to the **iot datasource** node.

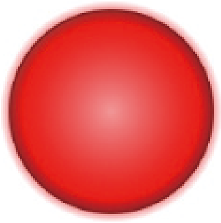
```
msg.payload = {
   tstamp: new Date(),
   data: {
     content: “change label”,
     color: “rgb(255,0,0)”,
     fontSize: 30
   }
 };
return msg;
```

You can configure fields as shown below while selecting their data source.

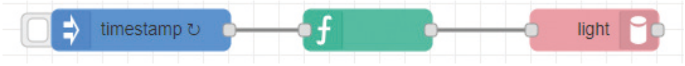
Name: The title of the chart.	**Font Size:** The size of the content.
Show: Check to show the title on the dashboard.	**Font Size Fixed:** Check to fix the size of the font so that it will not change along with the area of the label modifications.
Content: The content of the label.	**Customer Div:** Define your own parts of the HTML document here.
Content Fixed: Check to fix the ratio of the width and the length of the content.	**Customer CSS:** Define your own applicable CSS styling here.
Font Color: The color value of the content in HTML color code.	

4.4.1.13 Light

Use light as an alternative alert sian based on the data source.



IoT Dataflow



To use light in the dashboard, make sure the payload structure is set as shown on the right before sending to the **iot datasource** node.

```
msg.payload = {
   tstamp: new Date(),
   data: {
      color: 1,
      turnOn: 1,
      mode: 1
   }
};

return msg;
```

Reference code for each member in data.

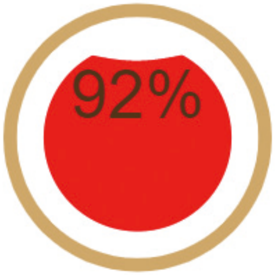
Color: 1 = green, 2 = yellow, 3 = red
turnOn: 0 = off, 1 = on
mode: 0 = lasting, 1 = flashing

You can configure fields as shown below while selecting their data source.

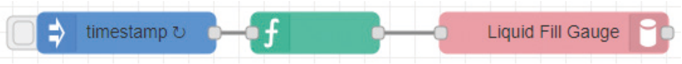
Name: The title of the chart.	**Light Type:** Three light types are available: red light, yellow light and green light.
Show: Check to show the title on the dashboard.	**Light Blinking:** Check to set the light to blink.
Chart Title Color: Set the chart title text color.	

4.4.1.14 Liquid Fill Gauge

Use a liquid fill gauge to measure your data off in a circle filled with liquescence animations.



IoT Dataflow



To use a liquid fill gauge in the dashboard, make sure the data type of a variable is set to numeric as shown before sending to the **iot datasource** node.

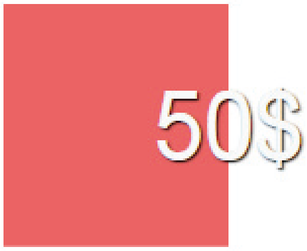
```
var value = Math.floor( Math.random() * 100 );
msg.payload = {
  tstamp: new Date().getTime(),
  data: value
};
return msg;
```

You can configure fields as shown below while selecting their data source.

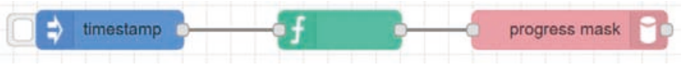
Name: The title of the chart.	**Upper Bound:** The value to split up statuses of warning and critical.
Show: Check to show the title on the dashboard.	**Color** **Normal/Warning/Critical:** The html color code for each status.
Value Range **Minimum:** The minimum value to display. **Maximum:** The maximum value to display.	**Text:** The html color code for the display number.
Lower Bound: The value to split up statuses of normal and warning.	**Circle:** The html color code for the ring.

4.4.1.15 Progress Mask

Use a progress mask to measure your data off in a bar from a preferred direction.



IoT Dataflow



To use a progress mask in the dashboard, make sure the data type of a variable is set to numeric as shown before sending to the **iot datasource** node.

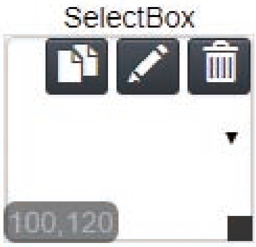
```
msg.payload = {
   tstamp: new Date(),
   data: {
     value: 50,     // the value range from 0 to
100
     unit: “$”,     // the value unit
     stroke:”#FF0000”  // the stroke color
   }
 };
return msg;
```

You can configure fields as shown below while selecting their data source.

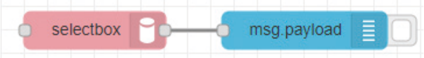
Name: The title of the chart.	**Opacity:** The value of the transparent level of the mask. From 0 to 1, the more the value is, the more opaque the mask is.
Show: Check to show the title on the dashboard.	**Text Size:** The size of the content.
Type: The direction the mask progresses toward.	**Value Unit:** Put the unit of measure here.
Stroke: The html color code for the mask.	

4.4.1.16 SelectBox

Use a select box to send the selection in the drop-down list back to the workspace of IoT Studio for further data processing.



IoT Dataflow



You can configure fields as shown below while selecting their data source.

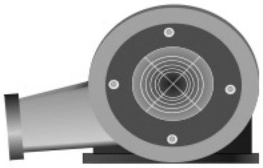
Name: The title of the chart.	**Title:** The title on top of the chart.
Show: Check to show the title on the dashboard.	**Options:** Put selectable items here. Separated by commons if more than one item.
Datasources: Assign one or multiple data sources as the target nodes for data reception. *No message payload is required to send to the* ***iot datasource*** *node for use with the select box.*	

4.4.2 Factory

4.4.2.1 Blower

Use blower figures to make up your dashboard.

Blower 1



Blower 2

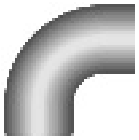
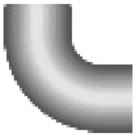
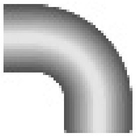
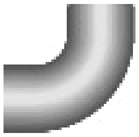
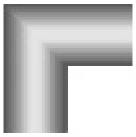
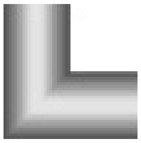
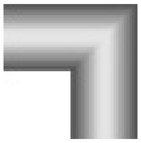
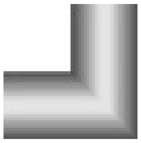
To use a blower in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired figure.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.2.2 Corner

Use corner figures to make up your dashboard.

 **Round Angle Left Top**	 **Round Angle Left Bottom**	 **Round Angle Right Top**	 **Round Angle Right Bottom**
 **Right Angle Left Top**	 **Right Angle Left Bottom**	 **Right Angle Right Top**	 **Right Angle Right Bottom**

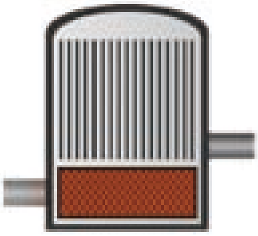
To use a corner in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired figure.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.2.3 Heater

Use heater figures to make up your dashboard.



Heater 1

Heater 2

Heater 3

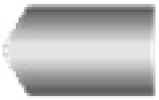
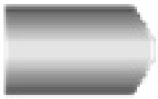
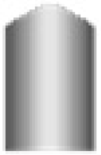
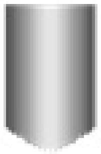
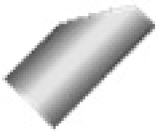
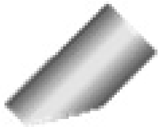
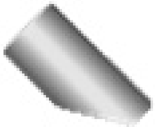
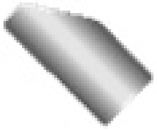
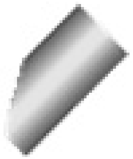
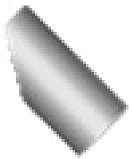
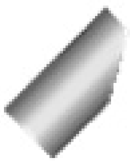
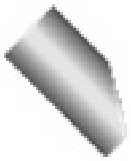
To use a heater in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired figure.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.2.4 Joint

Use joint figures to make up your dashboard.

 **Joint 1**	 **Joint 2**	 **Joint 3**	 **Joint 4**
 **Joint 5**	 **Joint 6**	 **Joint 7**	 **Joint 8**
 **Joint 9**	 **Joint 10**	 **Joint 11**	 **Joint 12**

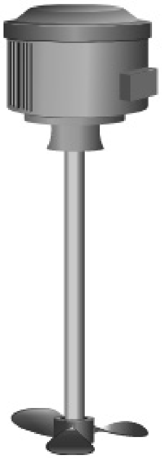
To use a joint in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired figure.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.2.5 Mixer

Use the mixer figure to make up your dashboard.



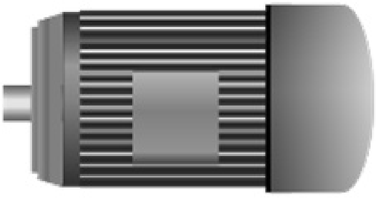
To use a mixer in the dashboard, assign any data source in its **Edit Chart**.

You can configure fields as shown below while selecting their data source.

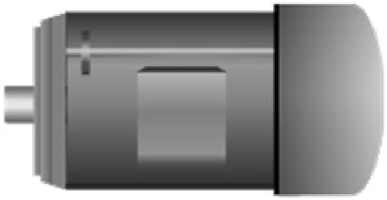
Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.2.6 Motor

Use motor figures to make up your dashboard.



Motor 1



Motor 2

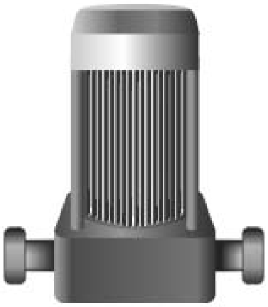
To use a motor in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired figure.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.2.7 Pump

Use the pump figure to make up your dashboard.



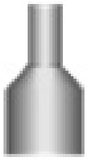
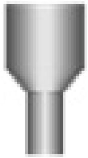
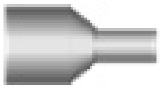
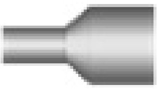
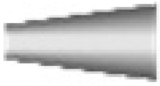
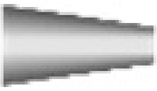
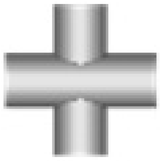
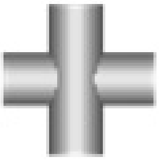
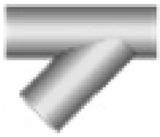
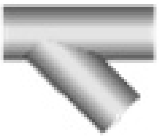
To use a pump in the dashboard, assign any data source in its **Edit Chart**.

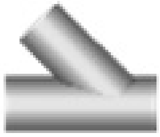
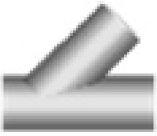
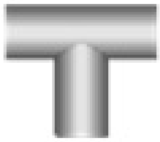
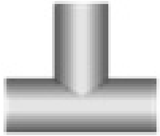
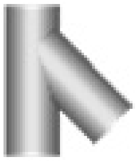
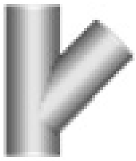
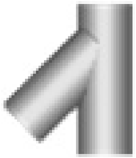
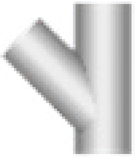
You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.2.8 Reduction

Use reduction figures to make up your dashboard.

 Reduction 1	 Reduction 2	 Reduction 3	 Reduction 4	 Reduction 5
 Reduction 6	 Reduction 7	 Reduction 8	 Reduction 9	 Reduction 10

 Reduction 11	 Reduction 12	 Reduction 13	 Reduction 14
 Reduction 15	 Reduction 16	 Reduction 17	 Reduction 18

To use a reduction in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired figure.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.2.9 Tank

Use the tank figure to make up your dashboard.

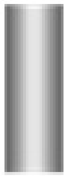
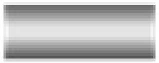
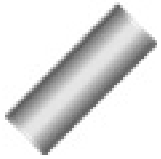
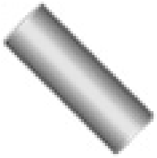
To use a tank in the dashboard, assign any data source in its **Edit Chart**.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.2.10 Tube

Use tube figures to make up your dashboard.

 **Tube 1**	 **Tube 2**	 **Tube 3**	 **Tube 4**

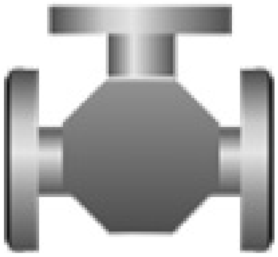
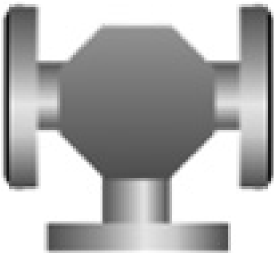
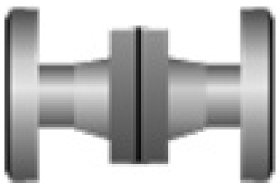
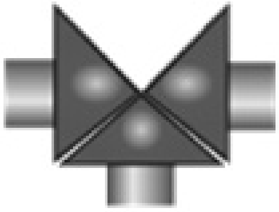
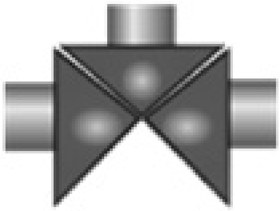
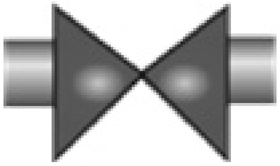
To use a tube in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired figure.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.2.11 Valve

Use valve figures to make up your dashboard.

 **Valve 1**	 **Valve 2**	 **Valve 3**
 **Valve 4**	 **Valve 5**	 **Valve 6**

To use a valve in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired figure.

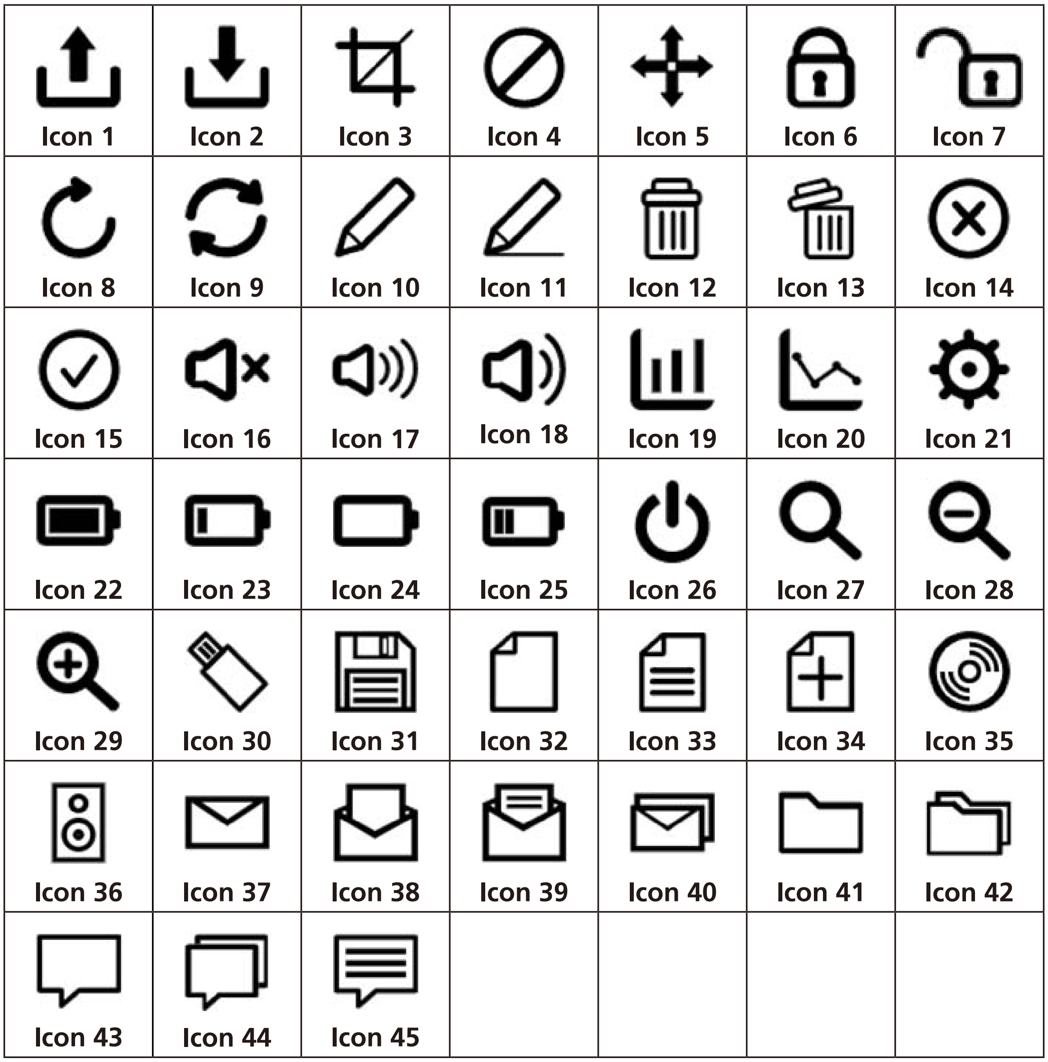
You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.3 Icon

4.4.3.1 Icon Class 1

Use icons to decorate your dashboard.



Icon 1	Icon 2	Icon 3	Icon 4	Icon 5	Icon 6	Icon 7
Icon 8	Icon 9	Icon 10	Icon 11	Icon 12	Icon 13	Icon 14
Icon 15	Icon 16	Icon 17	Icon 18	Icon 19	Icon 20	Icon 21
Icon 22	Icon 23	Icon 24	Icon 25	Icon 26	Icon 27	Icon 28
Icon 29	Icon 30	Icon 31	Icon 32	Icon 33	Icon 34	Icon 35
Icon 36	Icon 37	Icon 38	Icon 39	Icon 40	Icon 41	Icon 42
Icon 43	Icon 44	Icon 45				

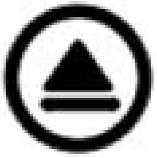
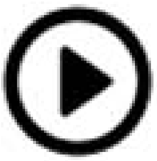
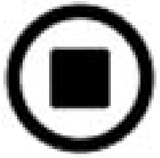
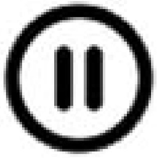
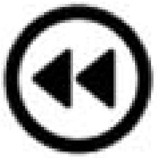
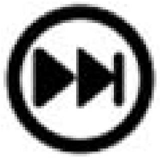
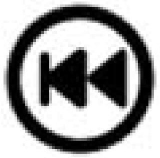
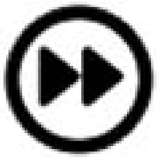
To use an icon in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired figure.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.3.2 Icon Class 2

Use icons to decorate your dashboard.

 **Icon 1**	 **Icon 2**	 **Icon 3**	 **Icon 4**
 **Icon 5**	 **Icon 6**	 **Icon 7**	 **Icon 8**

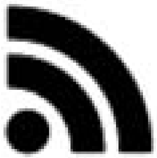
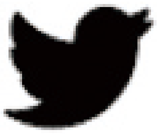
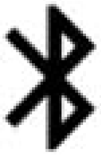
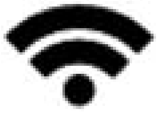
To use an icon in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired figure.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.3.3 Icon Class 3

Use icons to decorate your dashboard.

 **Icon 1**	 **Icon 2**	 **Icon 3**	 **Icon 4**

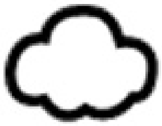
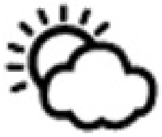
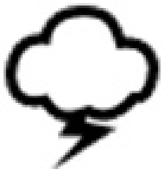
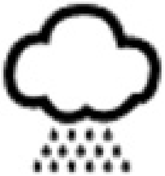
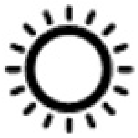
To use an icon in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired figure.

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.3.4 Icon Class 4

Use icons to decorate your dashboard.

				
Icon 1	**Icon 2**	**Icon 3**	**Icon 4**	**Icon 5**

To use an icon in the dashboard, assign any data source in its **Edit Chart**. In the **Type** drop-down menu, select your desired icon.

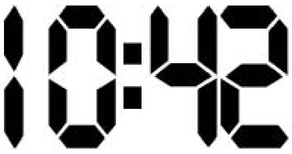
You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.4 Media

4.4.4.1 Digi-Clock

Use a digi-clock to present your data in time format.



To use a digi-clock in the dashboard, make sure the payload structure is set as shown on the right before sending to the **iot datasource** node.

```
var dt = new Date();
var h = dt.getHours();
var m = dt.getMinutes();
msg.payload = {
   tstamp: dt,
   data: {
      hH: parseInt(h/10),
      hL: h % 10,
      mH: parseInt(m/10),
      mL: m % 10,
   }
};

return msg;
```

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.
Show: Check to show the title on the dashboard.

4.4.5 Meter

4.4.5.1 Gauge

Present data in a meter gauge to measure your data off in the range of 0 to 100.

Gauge Round 1

Gauge Round 2

Gauge Round 3

Gauge Horizontal 1

To use a meter gauge in the dashboard, make sure the data type of a variable is set to numeric as shown before sending to the **iot datasource** node.

```
msg.payload = {
   tstamp: new Date().getTime(),
   data: {
      value: Math.floor(Math.random()*100)
   }
};

return msg;
```

You can configure fields as shown below while selecting their data source.

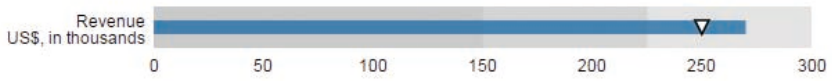
Name: The title of the chart.

Show: Check to show the title on the dashboard.

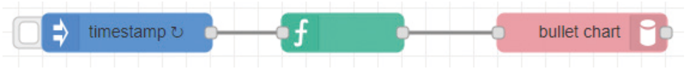
4.4.6 NVD3

4.4.6.1 Bullet Chart

You can use a bullet chart to display sufficient information and save space without useless and distracting decoration. The bullet chart features a primary measure, compares that measure to one or more other measures to enrich its meaning, and displays it in the context of qualitative ranges of performance.



IoT Dataflow



To use a bullet chart in the dashboard, in **msg.payload**, make sure that there are variables as shown before sending to the **iot datasource** node where title and subtitle stand for leads accompanying the chart; ranges and markers, for scales; measures, for amount; and color, for contrast.

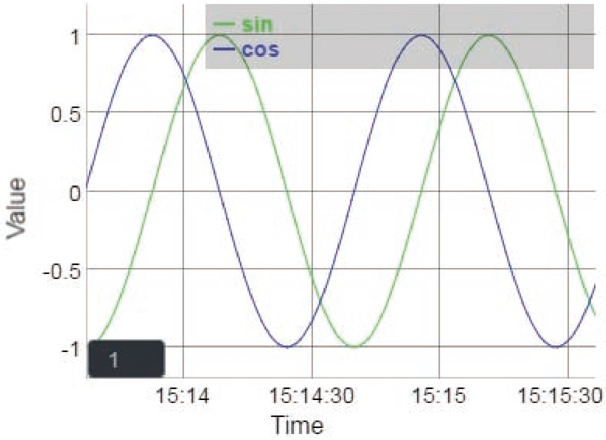
```
var value = Math.floor( Math.random()* 300);
msg.payload = {
   tstamp: new Date() .getTime(),
   data:{
         title:”Revenue”,
         subtitle:”US$, in thousands”,
         ranges:[150,225,300],
         measures:[value],
         markers:[250]
      }
   };
return msg;
```

You can configure fields as shown below while selecting their data source.

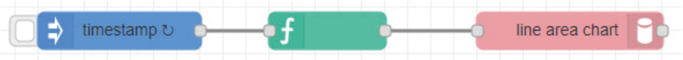
Name: The title of the chart.
Show: Check to show the title on the dashboard.
Text Color: The color code of the label strings.

4.4.6.2 Line/Area Chart

A Line/Area chart displays data as a series of points connected by line segments. You can link multiple values to a line/area chart with multiple lines. The presentation of each line is not coupling but independent from each other. While selecting the values to link to the chart, you can tick **Fill area under graph** to tint the area under the line.



IoT Dataflow



To use a line/area chart in the dashboard, make sure the types of variables inside **data** are set to numeric as shown before sending to the **iot datasource** node. The x-axis of the chart is fixed with timestamp.

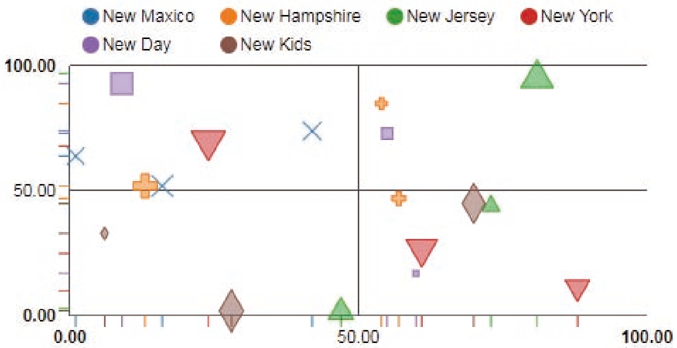
```
var data=10;
msg.payload = {
   tstamp: new Date().getTime(),
   data: {
      data1 : data,
      data2 : data
   }
};
return msg;
```

You can configure fields as shown below while selecting their data source.

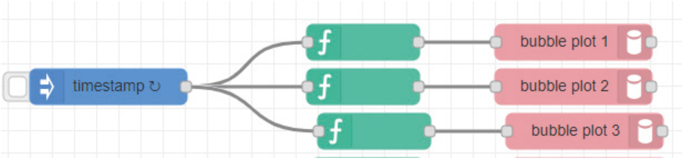
Name: The title of the chart.	**X Axis Label:** The X axis label string.
Show: Check to show the title on the dashboard.	**Y Axis Label:** The Y axis label string.
Request data between now and... : Select from 5 different periods: **second(s) ago**, **minute(s) ago**, **hour(s) ago**, **day(s) ago**, and **month(s) ago**.	**show Range Selector:** To show / hide the bottom range selector.
Maximum number of datapoints (leave blank for no limit): Set the max data number to keep in.	**Text Color:** The color code of the label strings.

4.4.6.3 Bubble Plot

You can present data as a collection of points with variables to determine the coordinates on the horizontal axis and the vertical axis, which is ideal for static data in a fixed time frame but not ideal for chronic records.



IoT Dataflow



To use a NVD3 bubble plot in the dashboard, make sure **data** in the **msg.payload** is associated with two numeric variables contributed to the coordinate and a value size between 0 and 1 as articles as shown before sending to the **iot datasource** node. You can put different html color codes for each labeled value for contrast.

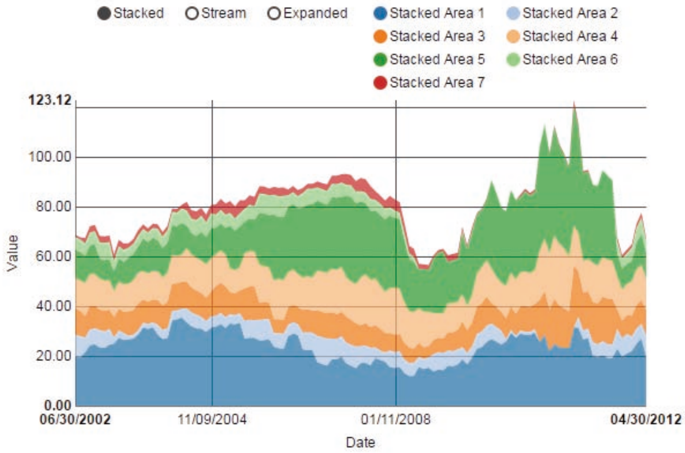
```
var shapes = [‘thin-x’, ‘cross’, ‘triangle-up’, ‘tri-
angle-down’, ‘diamond’, ‘square’];
var valueA = Math.floor( Math.random() * 100 );
var valueB = Math.floor( Math.random() * 100 );
msg.payload = {
   tstamp: new Date().getTime(),
   data: {
         x : valueA,
         y : valueB,
         size: Math.random(),
         shape: shapes[0]
   }
};
return msg;
```

You can configure fields as shown below while selecting their data source.

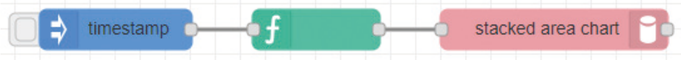
Name: The title of the chart.	**Y Axis Label:** The Y axis label string.
Show: Check to show the title on the dashboard.	**Y Axis Value Minimum:** The Y axis value range minimum.
X Axis Label: The X axis label string.	**Y Axis Value Maximum:** The Y axis value range maximum.
X Axis Value Minimum: The X axis value range minimum.	**Maximum Data Quantity:** The maximum data quantity in chart.
X Axis Value Maximum: The X axis value range maximum.	**Text Color:** The color code of the label strings.

4.4.6.4 Stack Area Chart

Use the stack area chart to present cumulated totals with numbers or percentages for showing trends among related attributes over time.



IoT Dataflow



To use a stack area chart in the dashboard, inside **data**, make sure the first variable acts as the x-axis and stored with time stamp, while the second variable, the y-axis, is set to a numeric value as shown before sending to the **iot datasource** node.

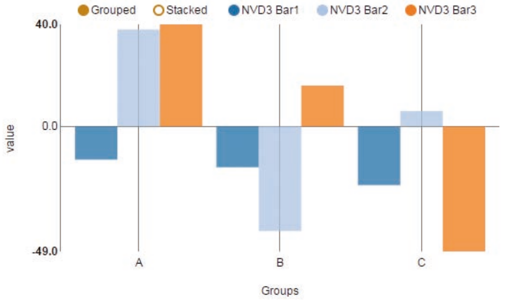
```
var data = [[ 1025409600000,
23.041422681023],
        [ 1028088000000, 19.854291255832],
        [ 1030766400000 , 21.02286281168]];

msg.payload = {
  tstamp: msg.payload,
  data: data
  };
return msg;
```

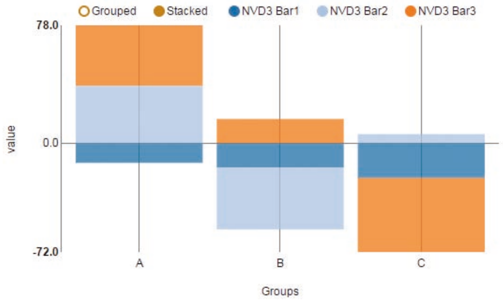
You can configure fields as shown below while selecting their data source.

Name: The title of the chart.	**Y Axis Label:** The Y axis label string.
Show: Check to show the title on the dashboard.	**Text Color:** The html color code for the text.
X Axis Label: The X axis label string.	

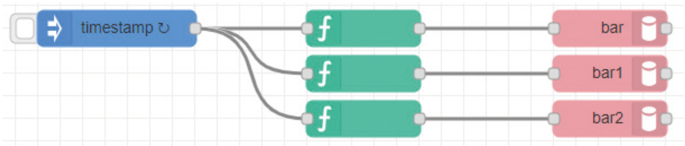
4.4.6.5 Bar

You can present data in rectangular bars vertically with the lengths analogous to the values and choices to line up with multiple data sources. Click on the circle before **Grouped** or **Stacked** for your desired presentation.





IoT Dataflow



To use a bar in the dashboard, make sure **data** includes at least a numeric variable or is labeled for each value as shown before sending to the **iot datasource** node. You can put different html color codes for each labeled value for contrast.

```
var valueA = Math.floor( Math.random() * 100
) - 50;
var valueB = Math.floor( Math.random() * 100
) - 50;
var valueC = Math.floor( Math.random() * 100
) - 50;
msg.payload = {
   tstamp: new Date().getTime(),
   data:[
      { label: "A", value : valueA },
      { label: "B", value : valueB },
      { label: "C", value : valueC }
   ]
};
return msg;
```

You can configure fields as shown below while selecting their data source.

Name: The title of the chart.	**Max:** The maximum value available on Y axis.
Show: Check to show the title on the dashboard.	**Ticks:** The density of the scale on Y axis.
X Axis Label: The X axis label string.	**Format:** The string format of Y axis.
Y Axis Label: The Y axis label string.	**Group Spacing:** The spaces between each column groups.
Min: The minimum value available on Y axis.	**Text Color:** The color code of the label strings.

Note:

This function is used to format the number showed on Y axis. It takes a number as the only argument, and returns a string representing the formatted result. The following is the general form of a specifier:

```
[[fill]align][sign][symbol][0][width][,][.precision][type]
```

`[[fill]align]`	`fill`	Any character

	`align`	Fill the character in the blank and align the result according to the following indicator:
		• ”>”- right alignment (Default behavior).
		• ”=”- center alignment
		• ”<”- left alignment
		Example:
		d3.format(”*>8”)(1); //”*******1”
		d3.format(”*^8”)(1); //”****1***”
		d3.format(”*<8”)(1); //”1*******”

`[sign]`	• ”+” — It is used for both positive and negative numbers. • ”-” — It is used only for negative numbers. (Default behavior). • ” ”(space) — A space for zero or positive and a minus sign for negative.	

`[symbol]`	• ”$” (currency) — A currency symbol should be prefixed (or suffixed) per your locale.
	Example:
	d3.format(”$,”)(1250); //”$1,250”
	d3.format(”$,.2f”)(1250); //”$1,250.00”
	• ”#” (base) — For binary, octal, or hexadecimal output, prefix by ”0b”, ”0o”, or ”0x”, respectively.
	Example:
	d3.format(”#0b”)(125); //”0b1111101”
	d3.format(”#0o”)(125); //”0o175”
	d3.format(”#0x”)(125); //”0x7d”	

`[0]`	If the width parameter is prepended with a 0, then zeros will be added in front of the string.
	Example:
	d3.format(”08”)(1234); //”00001234”
	d3.format(”08.2f”)(123.456); //”00123.46”	

`[width]`	To specify a minimum width that the output string of the formatter needs to have.
	Example:
	d3.format(”8”)(1); //” 1”
	d3.format(”8,.2f”)(1); //” 1.00”
	d3.format(”8g”)(1e6); //” 1000000”	

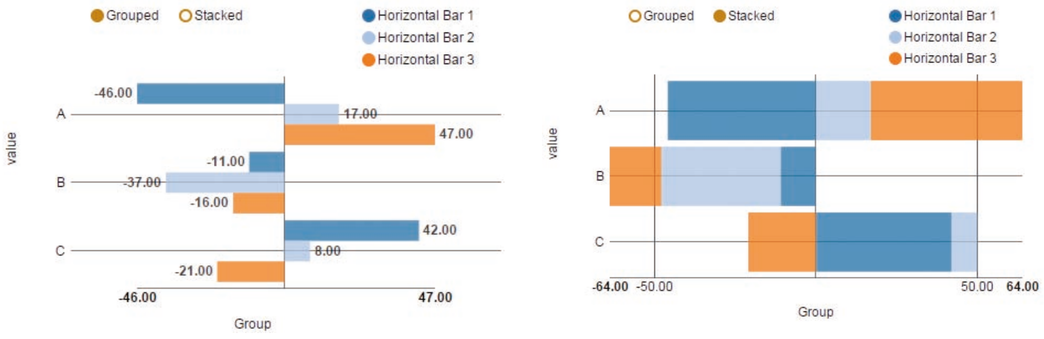
[,]	The comma (“,”) option enables the use of a comma for a thousand separator.
	Example:
	d3.format(“$,”)(1250); //”$1,250”
	d3.format(“$,.2f”)(1250); //”$1,250.00”

[.precision]	The precision indicates how many digits should be displayed after the decimal point for a value formatted with types “f” and “%”, or before and after the decimal point for a value formatted with types “g”, “r” and “p”.

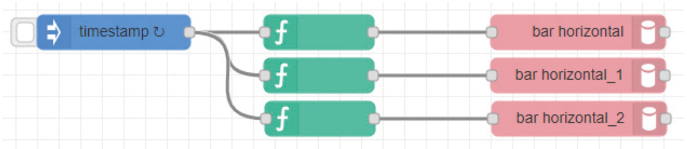
[type]	• “e” — Exponent notation. Print the number in scientific notation using letter ‘e’ to indicate the exponent.
	• “f” — Fixed point notation. Displays the number as a fixed-point number.
	• “g” — Round to the significant digits.
	Example:
	d3.format(“.4g”)(3.14159); //”3.142”
	d3.format(“.4f”)(3.14159); //”3.1416”
	• “s” — Decimal notation with an SI prefix, rounded to significant digits.
	Example:
	d3.format(“s”)(10000); //”10k”
	d3.format(“5s”)(10000); //” 10k”
	d3.format(“.5s”)(0.0001); //”100.00µ”
	• “%” — Multiplies the number by 100 and displays it in fixed (‘f’) format, followed by a percent sign.
	Example:
	d3.format(“%”)(0.1234); //”12%”
	d3.format(“.2%”)(0.1234); //”12.34%”
	• “p” — Multiplies the number by 100 and displays it in fixed (‘r’) format, followed by a percent sign.
	• “b” — Outputs the number in base 2.
	• “o” — Outputs the number in base 8.
	• “d” — Outputs the number in string format and ignore any non-integer values.
	• “x” — Outputs the number in base 16, using lower-case letters for digits above 9.
	• “X” — Outputs the number in base 16, using upper-case letters for digits above 9.
	• “c”— Converts the integer to the corresponding unicode character before printing.

4.4.6.6 Bar Horizontal

You can present data in rectangular bars horizontally with the lengths analogous to the values and choices to pile up multiple data sources. Click on the circle before **Grouped** or **Stacked** for your desired presentation.



IoT Dataflow



To use a horizontal bar in the dashboard, make sure **data** includes a **group** with at least a numeric variable or is labeled for each value as shown before sending to the **iot datasource** node. You can put different html color codes for each labeled value for contrast.

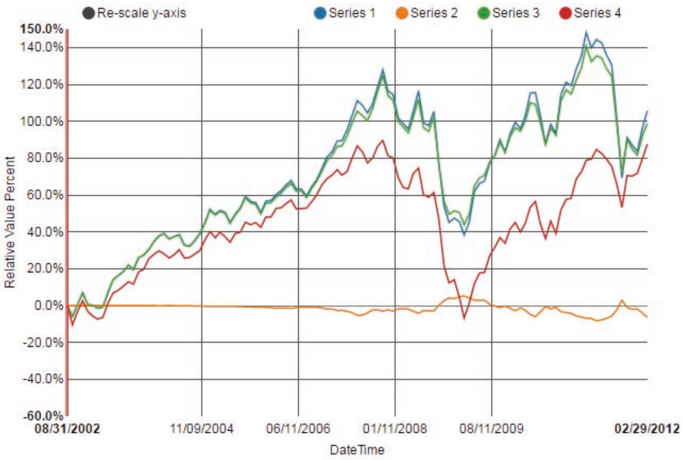
```
var valueA = Math.floor( (Math.random()-0.5) *
100 );
var valueB = Math.floor( (Math.random()-0.5) *
100 );
var valueC = Math.floor( (Math.random()-0.5) *
100 );
msg.payload = {
   tstamp: new Date().getTime(),
   data: {
      color: “rgb(31,119,180)”,
      group: [
         { label: “A”, value : valueA },
         { label: “B”, value : valueB },
         { label: “C”, value : valueC }
      ]
   }
};
return msg;
```

You can configure fields as shown below while selecting their data source.

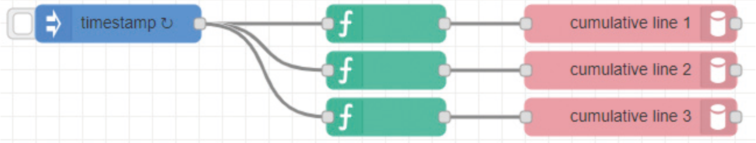
Name: The title of the chart.	**Y Axis Label:** The Y axis label string.
Show: Check to show the title on the dashboard.	**Text Color:** The color code of the label strings.
X Axis Label: The X axis label string.	

4.4.6.7 Cumulative Line

Use the cumulative line chart when you have one important grouping representing a chronic set of data and one value to show over time.



IoT Dataflow



To use a cumulative line in the dashboard, inside **data**, make sure the first variable acts as the x-axis, and stored with time stamp, while the second variable, the y-axis, is set to a numeric value as shown before sending to the **iot datasource** node. You can set every first variable to the same time stamp in order to compare values of the second variables from various data sources.

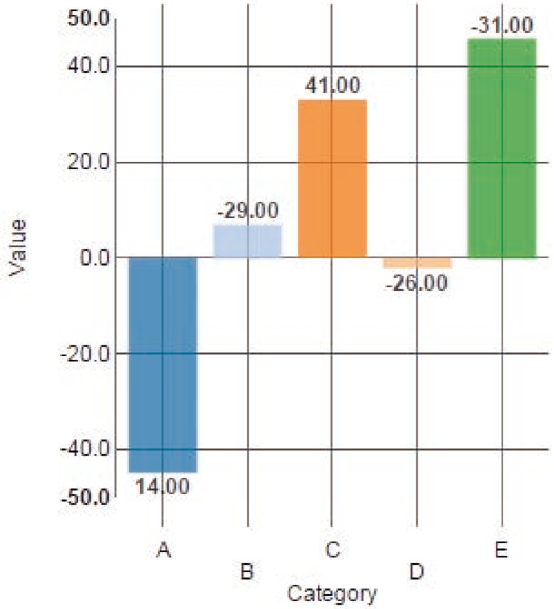
```
var value = Math.floor((Math.random()-0.5) *
100);
var now = new Date().getTime();
msg.payload = {
   tstamp: now,
   data:[[now, value]]
};
return msg;
```

You can configure fields as shown below while selecting their data sources.

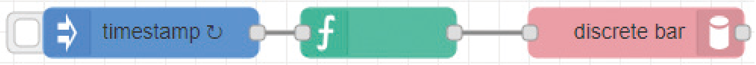
Name: The title of the chart.	**Y Axis Label:** The Y axis label string.
Show: Check to show the title on the dashboard.	**Y Axis Value Range:** The Y axis value range from the least to the most.
Text Color: The color code of the label strings.	**Maximum Data Quantity:** The maximum data quantity in chart.
X Axis Label: The X axis label string.	

4.4.6.8 Discrete Bar

Use the discrete bar to present categorical data visually and qualitatively.



IoT Dataflow



To use a discrete bar in the dashboard, make sure **data** includes at least a numeric value or is labeled for grouped values as shown before sending to the **iot datasource** node. You can put different html color codes for each labeled value for contrast.

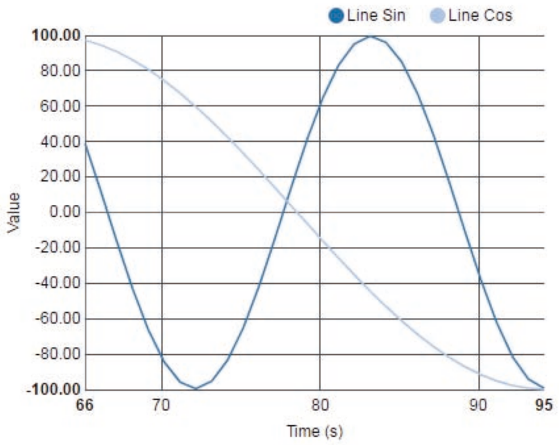
```
function getRandValue(offset, max) {
   return Math.floor((Math.random()-offset) *
max );
}
msg.payload = {
   tstamp: new Date().getTime(),
   data:
   [
      { label: “A”, value : getRandValue(0.5, 100) },
      { label: “B”, value : getRandValue(0.5, 100) },
      { label: “C”, value : getRandValue(0.5, 100) },
      { label: “D”, value : getRandValue(0.5, 100) },
      { label: “E”, value : getRandValue(0.5, 100) },
   ]
};
return msg;
```

You can configure fields as shown below while selecting their data sources.

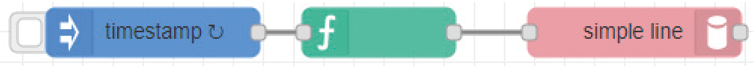
Name: The title of the chart.	**Y Axis Label:** The Y axis label string.
Show: Check to show the title on the dashboard.	**Text Color:** The html color code for the text.
X Axis Label: The X axis label string.	**Y Axis Value Range:** The Y axis value range from the least to the most.

4.4.6.9 Simple Line

A simple line displays data connected by non-linear segments. The presentation of each line is not coupling but independent from each other.



IoT Dataflow



To use a simple line in the dashboard, make sure **data** is set to a pair of numeric values as shown before sending to the **iot datasource** node. The first variable acts as the x-axis, while the second variable acts as the y-axis.

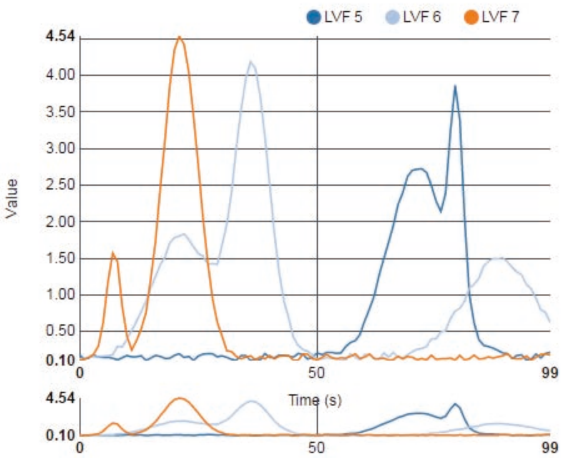
```
if (context.i === undefined) context.i=0;
msg.payload = {
  tstamp: new Date().getTime(),
  data: [
    { x: context.i, y: Math.sin(context.i*6)*100 }
  ]
};
context.i++;
return msg;
```

You can configure fields as shown below while selecting their data sources.

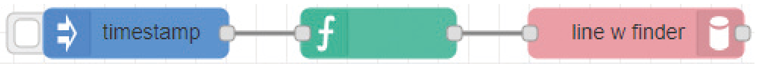
Name: The title of the chart.	**Y Axis Format:** The format of the Y axis label string.
Show: Check to show the title on the dashboard.	**Y Axis Value Range:** The Y axis value range from the least to the most.
X Axis Label: The X axis label string.	**Text Color:** The html color code for the text.
Y Axis Label: The Y axis label string.	**Maximum Data Quantity:** The maximum data quantity in chart.
X Axis Format: The format of the X axis label string.	

4.4.6.10 Line Chart with Finder

A line chart with view finder displays data as a chronic set of data points connected by line segments and lets you check a certain period of the presentation by dragging your mouse on the chart. The presentation of each line is not coupling but independent from each other.



IoT Dataflow



To use a line chart with view finder in the dashboard, make sure **data** is set to a numeric value in **msg.payload** as shown before sending to the **iot datasource** node. Inside **data**, the first variable acts as the x-axis, while the second variable acts as the y-axis.

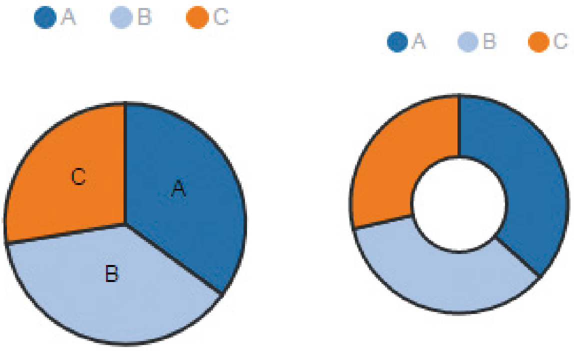
```
var data = [{“x”:0,”y”:0.17743331247475}
,{“x”:1,”y”:0.12145424904301763},{“x”:2,
”y”:0.13435301356948914}];
var now = new Date().getTime();
msg.payload = {
   tstamp: now,
   data: data
};
return msg;
```

You can configure fields as shown below while selecting their data sources.

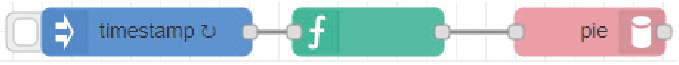
Name: The title of the chart.	**Y Axis Label:** The Y axis label string.
Show: Check to show the title on the dashboard.	**Text Color:** The html color code for the text.
X Axis Label: The X axis label string.	

4.4.6.11 Pie

Showing data in a circle divided into slices to illustrate numerical proportion. Typically, only one value will link to the chart.



IoT Dataflow



To use a pie in the dashboard, make sure **data** includes at least a numeric value or is labeled for grouped values in **msg.payload** as shown before sending to the **iot datasource** node. You can put different html color codes for each labeled value for contrast.

```
function getRandValue(offset, max) {
  return Math.floor((Math.random()-offset) *
max );
}
msg.payload = {
  tstamp: new Date().getTime(),
  data:
  [
    { label: “A”, value : getRandValue(0.5, 100) },
    { label: “B”, value : getRandValue(0.5, 100) },
    { label: “C”, value : getRandValue(0.5, 100) }
  ]
};
return msg;
```

You can configure fields as shown below while selecting their data sources.

Name: The title of the chart.	**Donut:** Check to set the chart to the donut chart.
Show: Check to show the title on the dashboard.	**Donut Ratio:** The width of the donut. The bigger the value is, the lesser the area of the donut is.
Text Color: The color code of the texts on each proportional area.	**Hide Upper Legend Bar:** Check to hide the legend.
Label Type: Select **Key**, **Value** or **Percentage** as the label for each proportional area.	**Colors:** The color code of each proportional area. Separated by commons.
Hide Pie Label: Check to hide the labels on each proportional area.	

4.5 Set Up IoT Studio with Modbus RTU Climate Sensors

Items to prepare:

1. A set of Modbus RTU temperature and humidity sensors.
2. A set of NexAIoT gateway device.
3. Required cables such as the power cable and the serial cable.

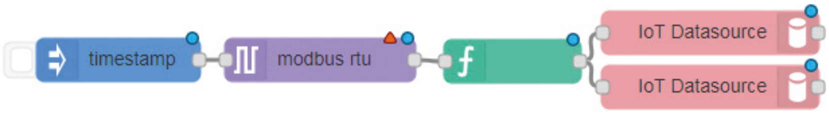
Prerequisite:

Every item above is well connected and turned on.

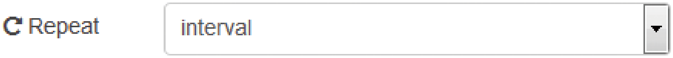
4.5.1 Plan your flow

Steps

1. Log onto your device's IP address for the IoT Studio page.
2. Add and connect 1 **inject** node, 1 **modbus rtu** node, 1 **function** node, and 2 **IoT Datasource** nodes to the workspace as shown.



3. Double click the inject node, select

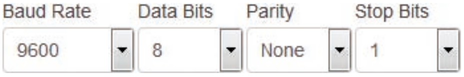

Repeat interval

, and then click **Ok**.

4. Double click the modbus rtu node, and click ✎.

Click 🔍 to select a port connected to your sensor.

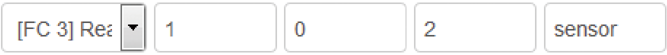
In **Settings**, adjust each value according to the specification of your sensor such as


Baud Rate 9600 Data Bits 8 Parity None Stop Bits 1

.

Click **Update**.

Make sure you have an FC input that looks like


[FC 3] Rea 1 0 2 sensor

, and then click **Ok**.

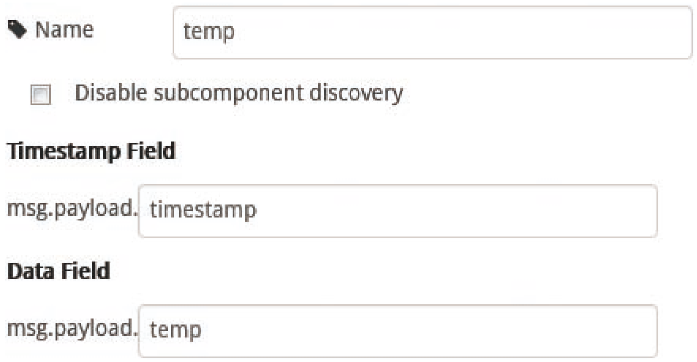
5. Double click the function node.

Copy and paste the codes below into the **Function** field.

```
msg.payload.temp=msg.payload.sensor.results[0]/100;
msg.payload.humi=msg.payload.sensor.results[1]/100;
msg.payload.timestamp=new Date().getTime();
return msg;
```

Click **Ok**.

6. Double click an IoT Datasource node.
   Fill temp in the **Name** field.
   Fill timestamp in **Timestamp Field**.
   Fill temp in **Data Field**.
   Click **Ok**.



Name: temp

□ Disable subcomponent discovery

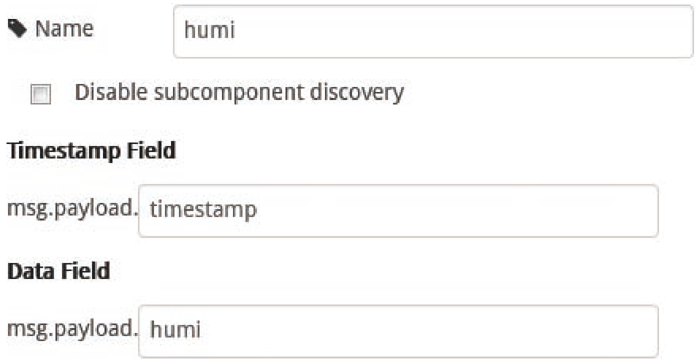
Timestamp Field

msg.payload. timestamp

Data Field

msg.payload. temp

Double click the other IoT Datasource node, and repeat the steps above but fill humi in both the **Name** field and **Data Field**.



Name: humi

□ Disable subcomponent discovery

Timestamp Field

msg.payload. timestamp

Data Field

msg.payload. humi

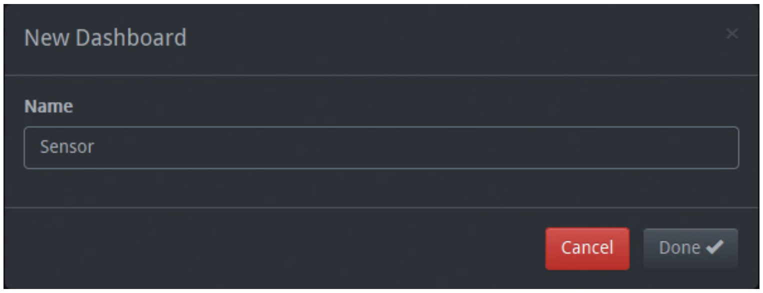
*Both of the names in the **Name** field will apply to dashboard configuration.

7. Click **Deploy** on the top right, and your flow should start running.

4.5.2 Configure Your Dashboard

Steps

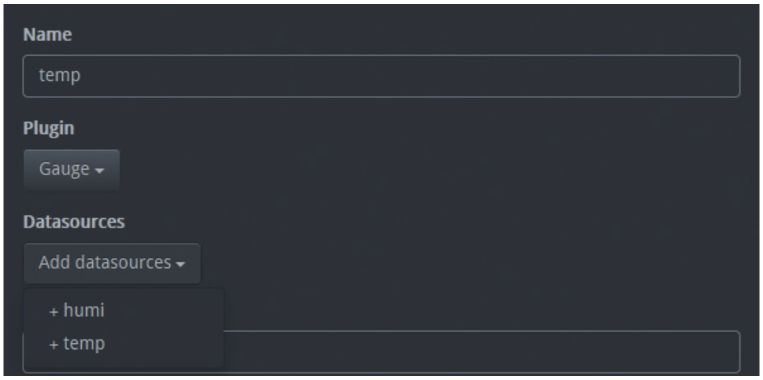
1. Log onto your device’s IP address/dash for the dashboard page.
2. Click + Create New Dashboard on the top right of the page.



New Dashboard

Name: Sensor

3. Fill Sensor in the **Name** field. Click **Done**.
4. Click + Create New Chart on the top right of the page. Fill temp in the **Name** field.

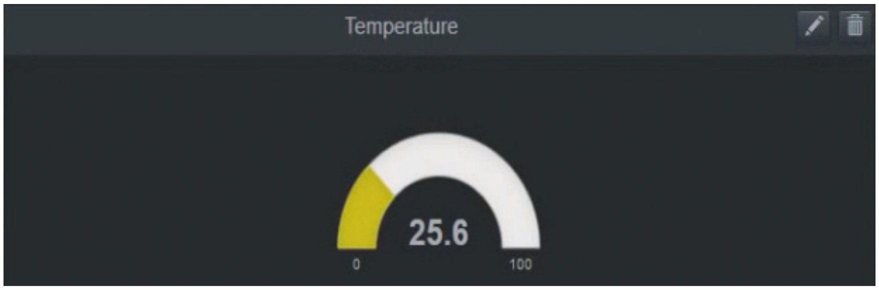


Name: temp

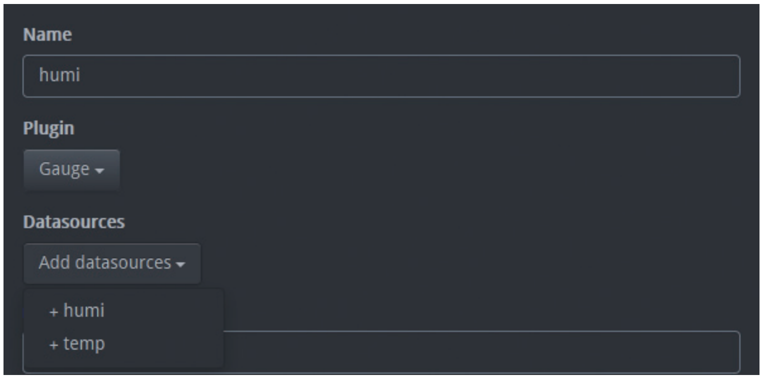
Plugin: Gauge

Datasources

Select **Gauge** from the **Plugin** drop-down menu. Select **+temp** in **Datasources**. Click **Done** and you should see the chart that reflects with your datasource.



5. Repeat step 4 but fill humi in the **Name** field. Select +humi in **Datasources**.



Name

humi

Plugin

Gauge ▾

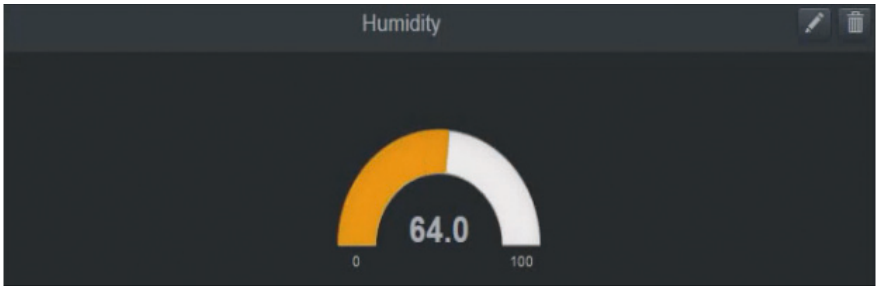
Datasources

Add datasources ▾

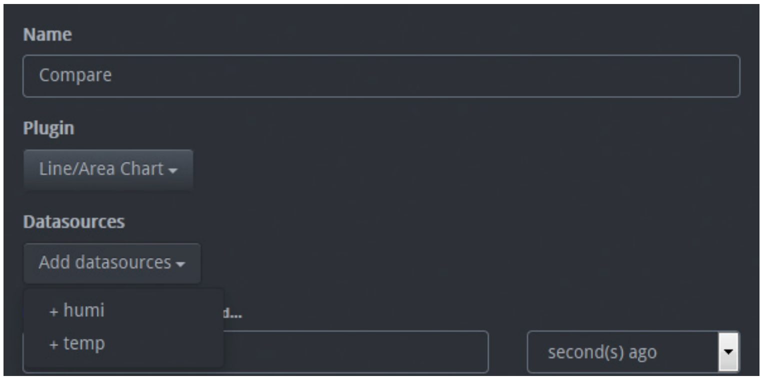
+ humi

+ temp

Click **Done** and you should see another chart that reflects with your datasource.



6. Click ✚ Create New Chart on the top right of the page. Fill Compare in the **Name** field. Select **Line/Area Chart** from the **Plugin** drop-down menu. Select both of the names in **Datasources**.



Name

Compare

Plugin

Line/Area Chart ▾

Datasources

Add datasources ▾

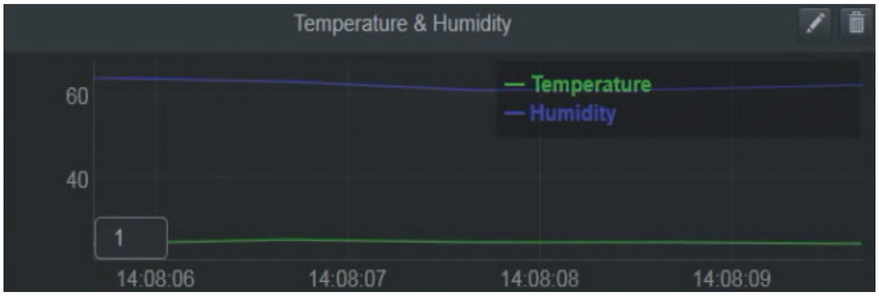
+ humi

+ temp

d...

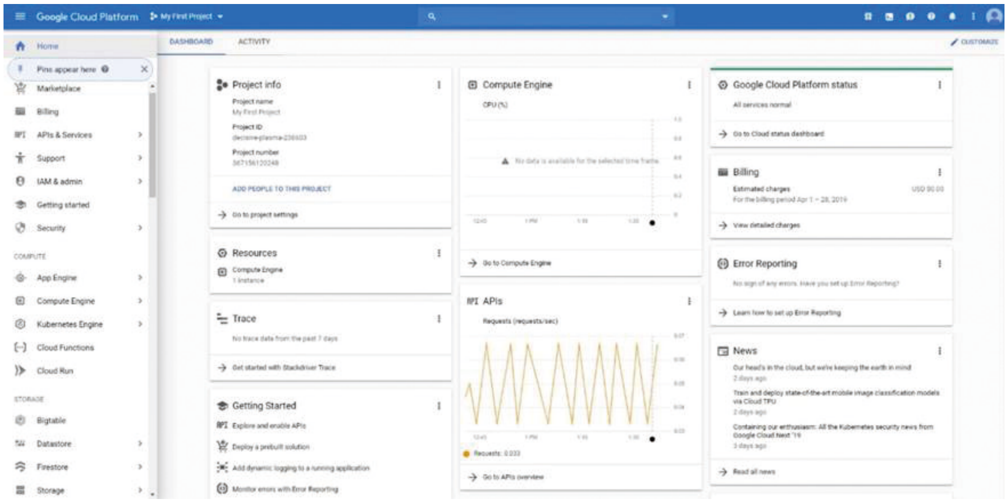
second(s) ago

Click **Done** and you should see the chart that reflects with both of your datasources.

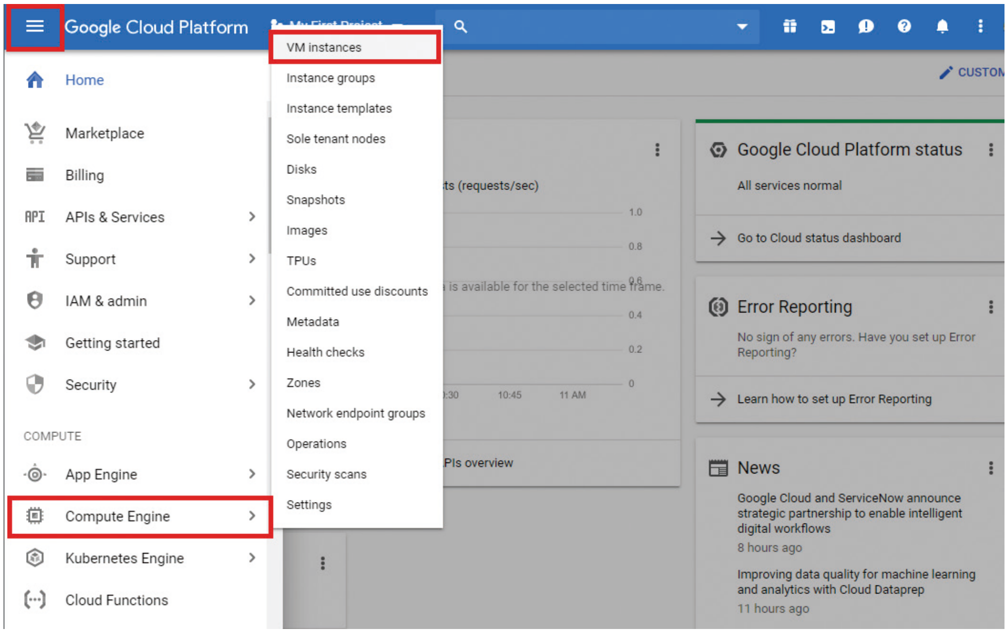


Appendix A: Create a Virtual Machine for IoT Studio in Google Cloud

1. Log in to Google Cloud.



2. Click the triple bar icon ≡ and select **Compute Engine** > **VM instances**.



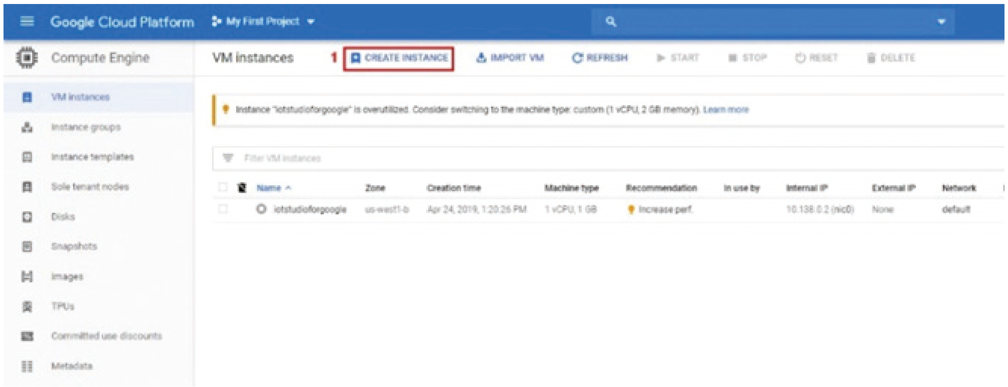
All services normal

No sign of any errors. Have you set up Error Reporting?

Google Cloud and ServiceNow announce strategic partnership to enable intelligent digital workflows
8 hours ago

Improving data quality for machine learning and analytics with Cloud Dataprep
11 hours ago

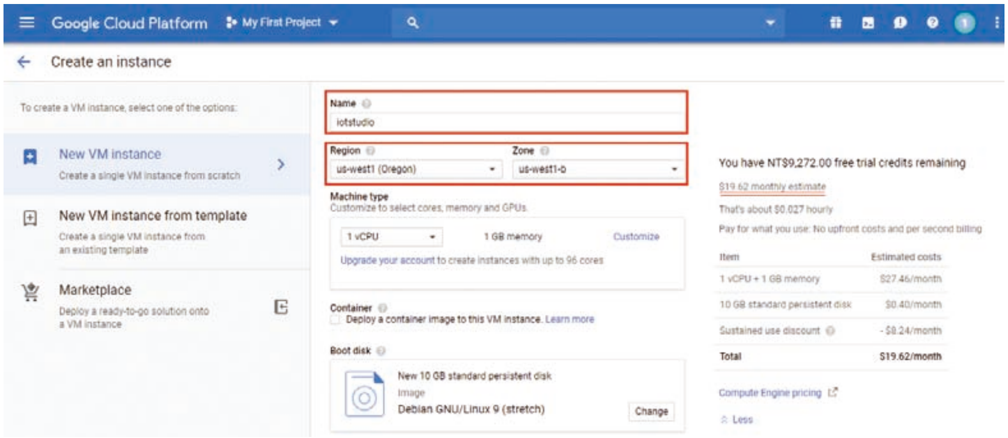
3. Click **CREATE INSTANCE** at the top of the page.



Instance "iotstudioforgoogle" is overutilized. Consider switching to the machine type: custom (1 vCPU, 2 GB memory). Learn more

Name	Zone	Creation time	Machine type	Recommendation	In use by	Internal IP	External IP	Network
iotstudioforgoogle	us-west1-b	Apr 24, 2019, 1:20:26 PM	1 vCPU, 1 GB	Increase perf.		10.138.0.2 (nic0)	None	default

4. Name the VM and select the region, which will affect the VM charge.


Google Cloud Platform
My First Project
Create an instance
To create a VM instance, select one of the options:
New VM instance
Create a single VM instance from scratch
New VM instance from template
Create a single VM instance from an existing template
Marketplace
Deploy a ready-to-go solution onto a VM instance
Customize
Change
Compute Engine pricing
Less

Name: iotstudio

Region: us-west1 (Oregon)

Zone: us-west1-b

Machine type
Customize to select cores, memory and GPUs.
1 vCPU 1 GB memory
Upgrade your account to create instances with up to 96 cores

Container
□ Deploy a container image to this VM instance. Learn more

Boot disk
New 10 GB standard persistent disk
Image
Debian GNU/Linux 9 (stretch)

You have NT$9,272.00 free trial credits remaining

$19.62 monthly estimate

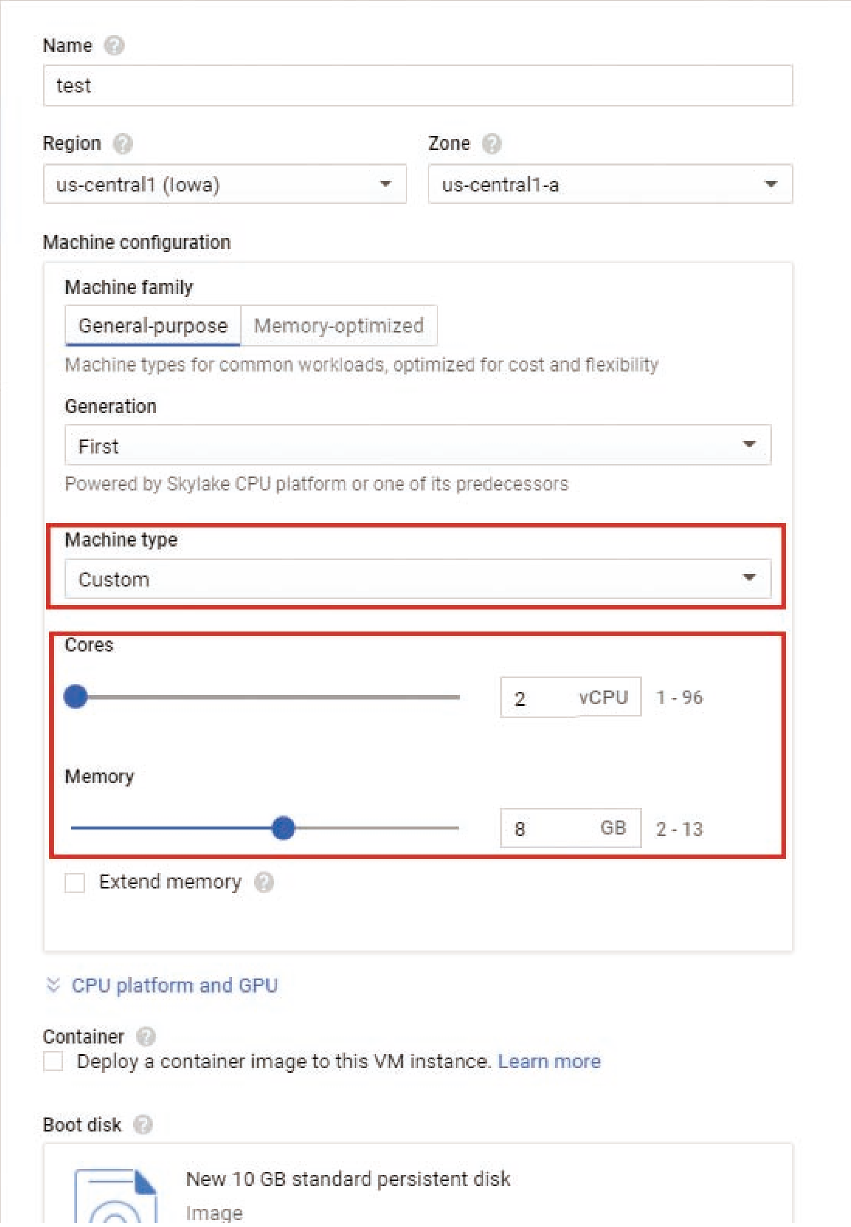
That's about $0.027 hourly

Pay for what you use: No upfront costs and per second billing

Item	Estimated costs
1 vCPU + 1 GB memory	$27.46/month
10 GB standard persistent disk	$0.40/month
Sustained use discount	- $8.24/month
Total	$19.62/month

5. Choose **Custom** under **Machine type** and set **Cores** to **2 vCPU** and **Memory** to **8GB**.

   Note: Please set **at least** 2 vCPU cores and 4GB memory. It's recommended to use 2 vCPU cores and 8GB memory. The speed of git clone and data deployment depends on the specifications of the VM and Internet environment.



Name

test

Region

us-central1 (Iowa)

Zone

us-central1-a

Machine configuration

Machine family

General-purpose | Memory-optimized

Machine types for common workloads, optimized for cost and flexibility

Generation

First

Powered by Skylake CPU platform or one of its predecessors

Machine type

Custom

Cores

2 vCPU 1 - 96

Memory

8 GB 2 - 13

□ Extend memory

CPU platform and GPU

Container

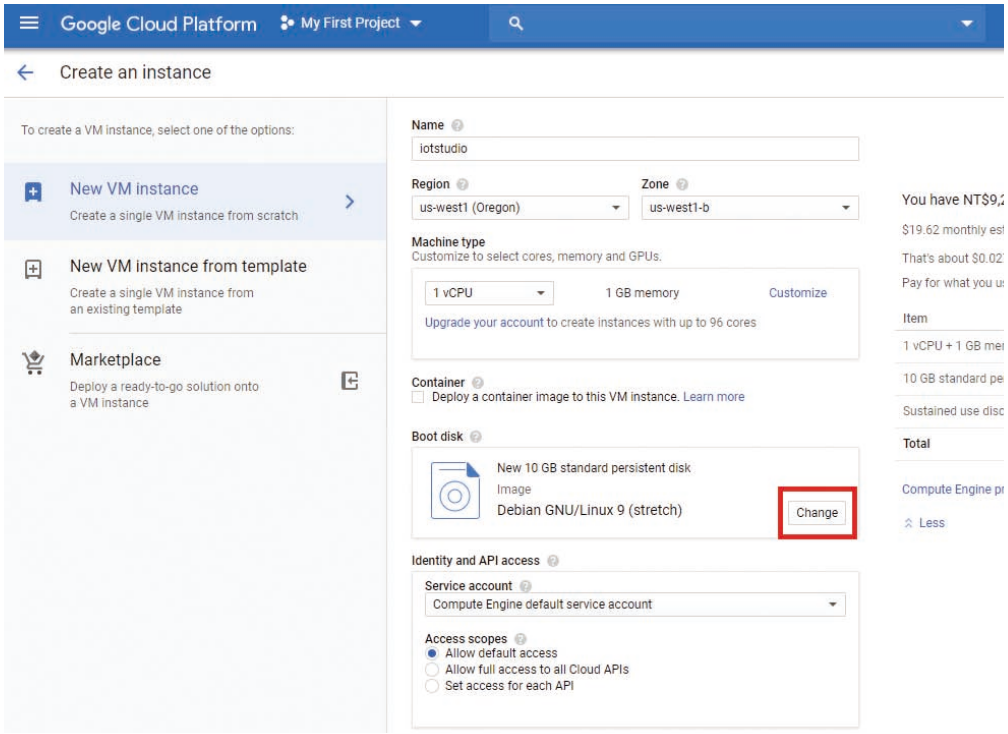
□ Deploy a container image to this VM instance. Learn more

Boot disk

New 10 GB standard persistent disk

Image

6. Click **Change** to set up OS and disk.



To create a VM instance, select one of the options:

New VM instance
Create a single VM instance from scratch

New VM instance from template
Create a single VM instance from an existing template

Marketplace
Deploy a ready-to-go solution onto a VM instance

Name: iotstudio

Region: us-west1 (Oregon)

Zone: us-west1-b

Machine type
Customize to select cores, memory and GPUs.

1 vCPU — 1 GB memory — Customize

Upgrade your account to create instances with up to 96 cores

Container
□ Deploy a container image to this VM instance. Learn more

Boot disk
New 10 GB standard persistent disk
Image
Debian GNU/Linux 9 (stretch)

Identity and API access
Service account: Compute Engine default service account

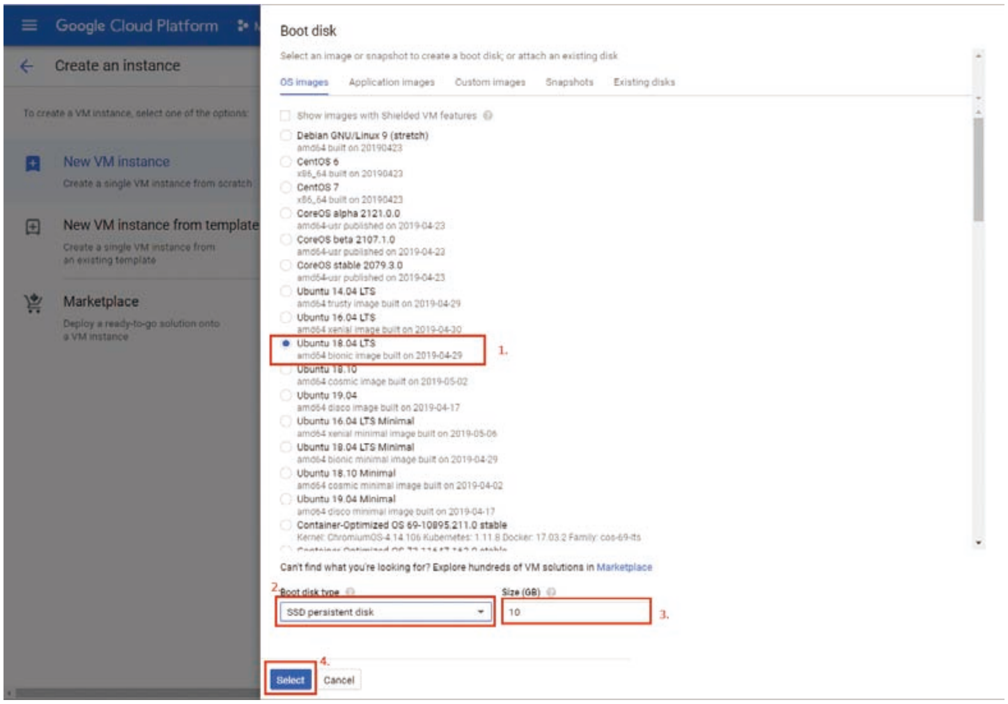
Access scopes
● Allow default access
○ Allow full access to all Cloud APIs
○ Set access for each API

You have NT$9,
$19.62 monthly est
That's about $0.02
Pay for what you u

Item
1 vCPU + 1 GB me
10 GB standard pe
Sustained use disc
Total

Compute Engine p
Less

7. Select **Ubuntu 18.04 TLS** as OS image. Choose **SSD persistent disk** under Boot disk type and set disk size as 10 GB. Then, click **Select**.



Boot disk

Select an image or snapshot to create a boot disk; or attach an existing disk

□ Show images with Shielded VM features

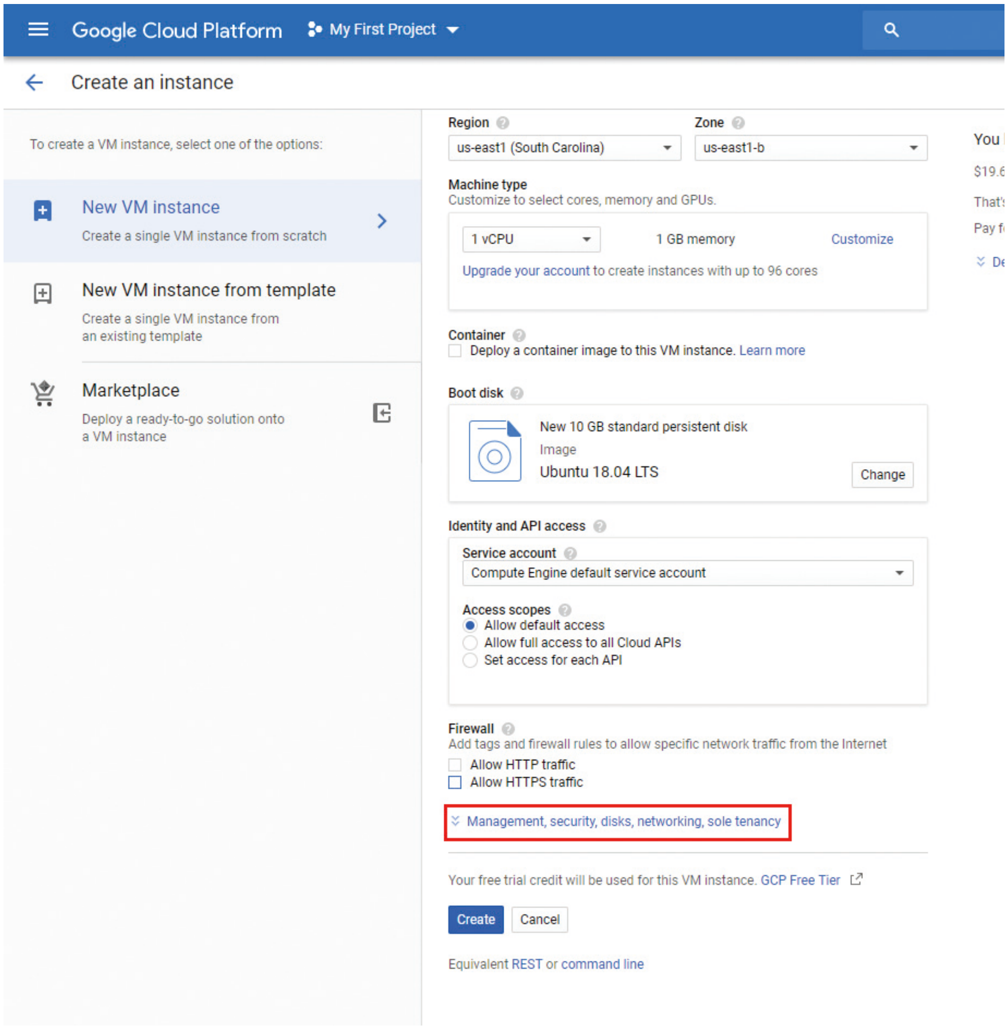
○ Debian GNU/Linux 9 (stretch)
amd64 built on 20190423
○ CentOS 6
x86_64 built on 20190423
○ CentOS 7
x86_64 built on 20190423
○ CoreOS alpha 2121.0.0
amd64-usr published on 2019-04-23
○ CoreOS beta 2107.1.0
amd64-usr published on 2019-04-22
○ CoreOS stable 2079.3.0
amd64-usr published on 2019-04-23
○ Ubuntu 14.04 LTS
amd64 trusty image built on 2019-04-29
○ Ubuntu 16.04 LTS
amd64 xenial image built on 2019-04-30
● Ubuntu 18.04 LTS
amd64 bionic image built on 2019-04-29
○ Ubuntu 18.10
amd64 cosmic image built on 2019-05-02
○ Ubuntu 19.04
amd64 disco image built on 2019-04-17
○ Ubuntu 16.04 LTS Minimal
amd64 xenial minimal image built on 2019-05-06
○ Ubuntu 18.04 LTS Minimal
amd64 bionic minimal image built on 2019-04-29
○ Ubuntu 18.10 Minimal
amd64 cosmic minimal image built on 2019-04-02
○ Ubuntu 19.04 Minimal
amd64 disco minimal image built on 2019-04-17
○ Container-Optimized OS 69-10895.211.0 stable
Kernel: ChromiumOS-4.14.106 Kubernetes: 1.11.8 Docker: 17.03.2 Family: cos-69-lts

Can't find what you're looking for? Explore hundreds of VM solutions in Marketplace

Boot disk type: SSD persistent disk

Size (GB): 10

8. Click **Management**, **security**, **disks**, **networking**, **sole tenancy**.



Google Cloud Platform · My First Project

Create an instance

To create a VM instance, select one of the options:

- New VM instance — Create a single VM instance from scratch
- New VM instance from template — Create a single VM instance from an existing template
- Marketplace — Deploy a ready-to-go solution onto a VM instance

Region: us-east1 (South Carolina)

Zone: us-east1-b

Machine type

Customize to select cores, memory and GPUs.

1 vCPU · 1 GB memory · Customize

Upgrade your account to create instances with up to 96 cores

Container

□ Deploy a container image to this VM instance. Learn more

Boot disk

New 10 GB standard persistent disk

Image

Ubuntu 18.04 LTS · Change

Identity and API access

Service account: Compute Engine default service account

Access scopes

● Allow default access

○ Allow full access to all Cloud APIs

○ Set access for each API

Firewall

Add tags and firewall rules to allow specific network traffic from the Internet

□ Allow HTTP traffic

□ Allow HTTPS traffic

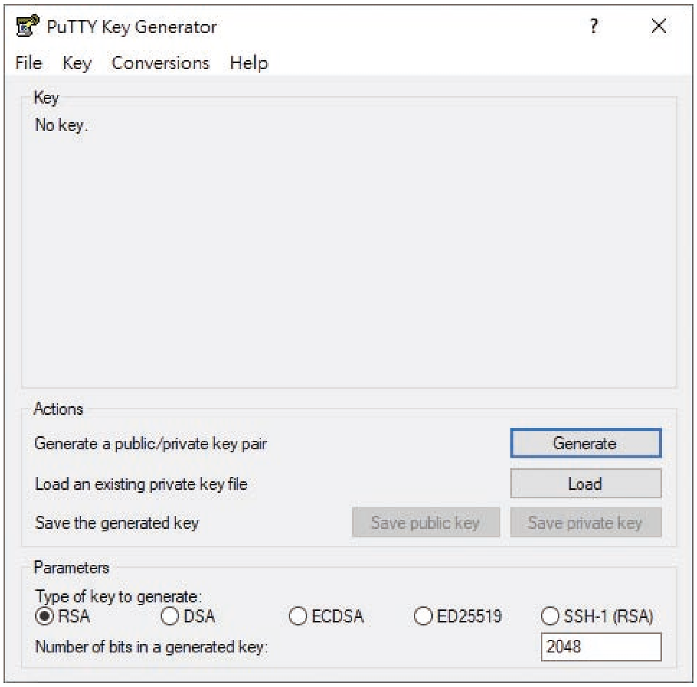
Management, security, disks, networking, sole tenancy

Your free trial credit will be used for this VM instance. GCP Free Tier

Create · Cancel

Equivalent REST or command line

9. In this step, we need to generate a key pair for the VM's SSH secure connection by using the puttygen application.
   i. Open PuTTYgen, which you can download at: https://puttygen.com/download.php?val=46



No key.

Generate a public/private key pair

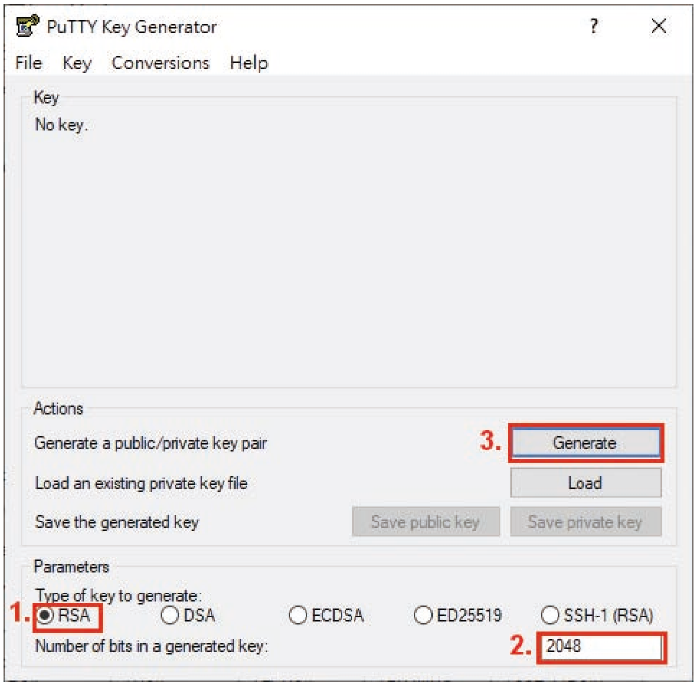
Load an existing private key file

Save the generated key

Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

   ii. Choose **RSA** under **Type of key to generate** and type **2048** in the **Number of bits in a generated key's** text box. Then, click **Generate**.



No key.

Generate a public/private key pair

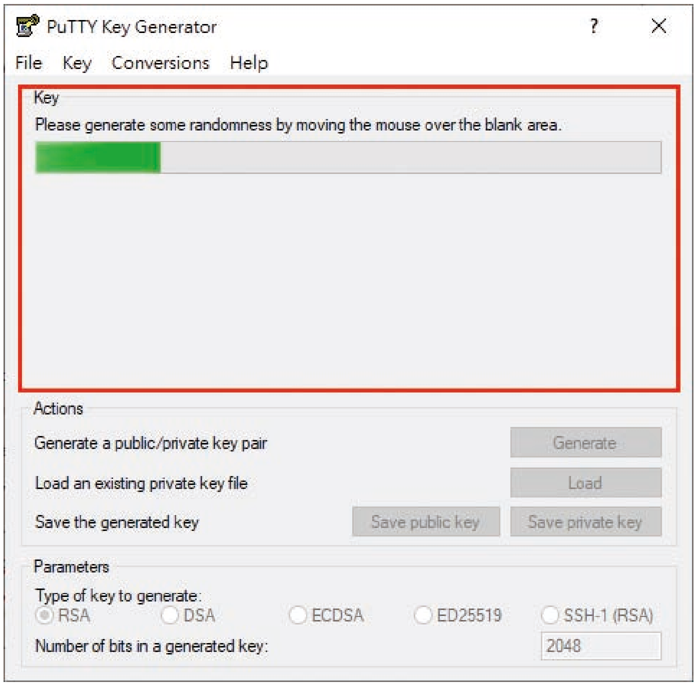
Load an existing private key file

Save the generated key

Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

iii. Move the mouse randomly in the red box until the progress is complete.



Please generate some randomness by moving the mouse over the blank area.

Generate a public/private key pair

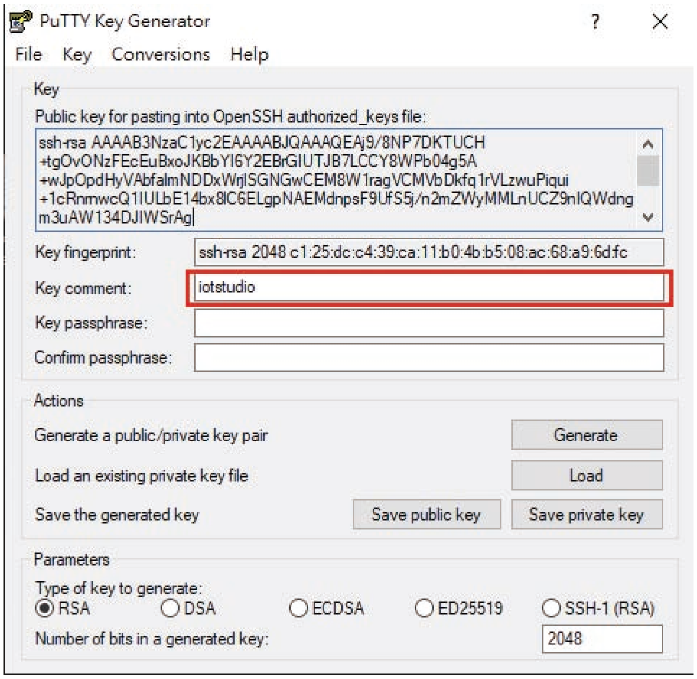
Load an existing private key file

Save the generated key

Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

iv. You need to modify the **Key comment** (do not use any special symbols) as a remark for your public key. It will affect the composition of the key and will also be used as the username to access the VM through SSH.



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAj9/8NP7DKTUCH
+tgOvONzFEcEuBxoJKBbYI6Y2EBrGIUTJB7LCCY8WPb04g5A
+wJpOpdHyVAbfalmNDDxWrjISGNGwCEM8W1ragVCMVbDkfq1rVLzwuPiqui
+1cRnmwcQ1IULbE14bx8IC6ELgpNAEMdnpsF9UfS5j/n2mZWyMMLnUCZ9nIQWdng
m3uAW134DJIWSrAg
```

Key fingerprint: ssh-rsa 2048 c1:25:dc:c4:39:ca:11:b0:4b:b5:08:ac:68:a9:6d:fc

Key comment: iotstudio

Key passphrase:

Confirm passphrase:

Generate a public/private key pair

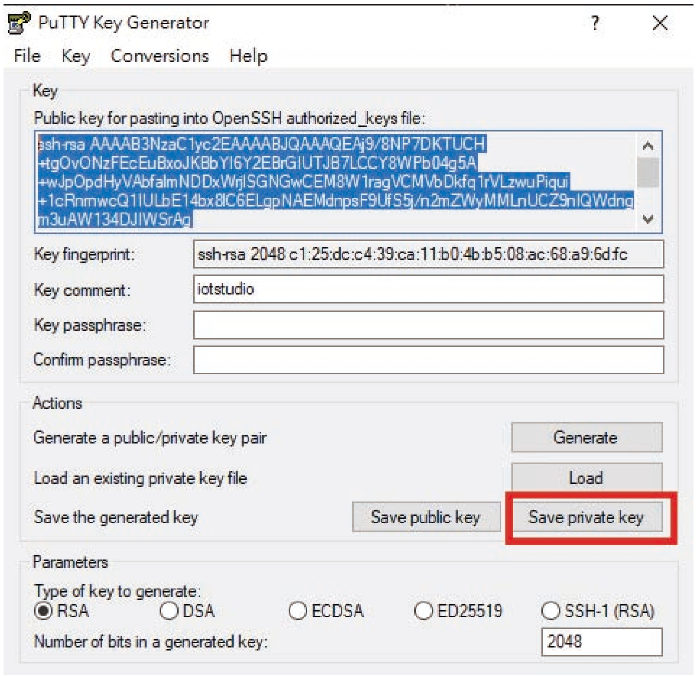
Load an existing private key file

Save the generated key

Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

v. Save the **private key**, which will be used in IoT Studio's One Click Configuration. Do not close PuTTYgen.



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAj9/8NP7DKTUCH
+tgOvONzFEcEuBxoJKBbYI6Y2EBrGIUTJB7LCCY8WPb04g5A
+wJpOpdHyVAbfalmNDDxWrjlSGNGwCEM8W1ragVCMVbDkfq1rVLzwuPiqui
+1cRnmwcQ1IULbE14bx8lC6ELgpNAEMdnpsF9UfS5j/n2mZWyMMLnUCZ9nIQWdng
m3uAW134DJIWSrAg
```

Key fingerprint: ssh-rsa 2048 c1:25:dc:c4:39:ca:11:b0:4b:b5:08:ac:68:a9:6d:fc

Key comment: iotstudio

Key passphrase:

Confirm passphrase:

Generate a public/private key pair

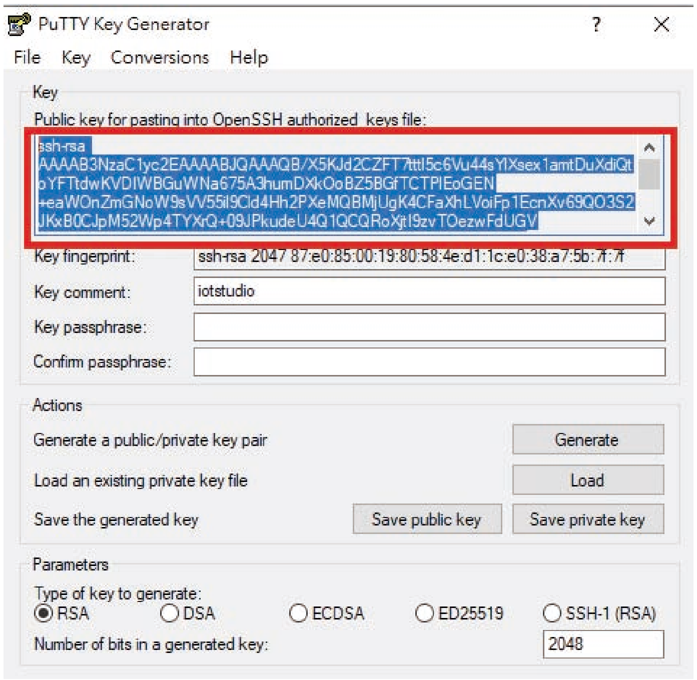
Load an existing private key file

Save the generated key

Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

vi. Copy the public key as shown in the red box below.



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQB/X5KJd2CZFT7ttI5c6Vu44sYIXsex1amtDuXdiQt
oYFTtdwKVDIWBGuWNa675A3humDXkOoBZ5BGfTCTPIEoGEN
+eaWOnZmGNoW9sVV55iI9Cld4Hh2PXeMQBMjUgK4CFaXhLVoiFp1EcnXv69QO3S2
JKxB0CJpM52Wp4TYXrQ+09JPkudeU4Q1QCQRoXjtI9zvTOezwFdUGV
```

Key fingerprint: ssh-rsa 2047 87:e0:85:00:19:80:58:4e:d1:1c:e0:38:a7:5b:7f:7f

Key comment: iotstudio

Key passphrase:

Confirm passphrase:

Generate a public/private key pair

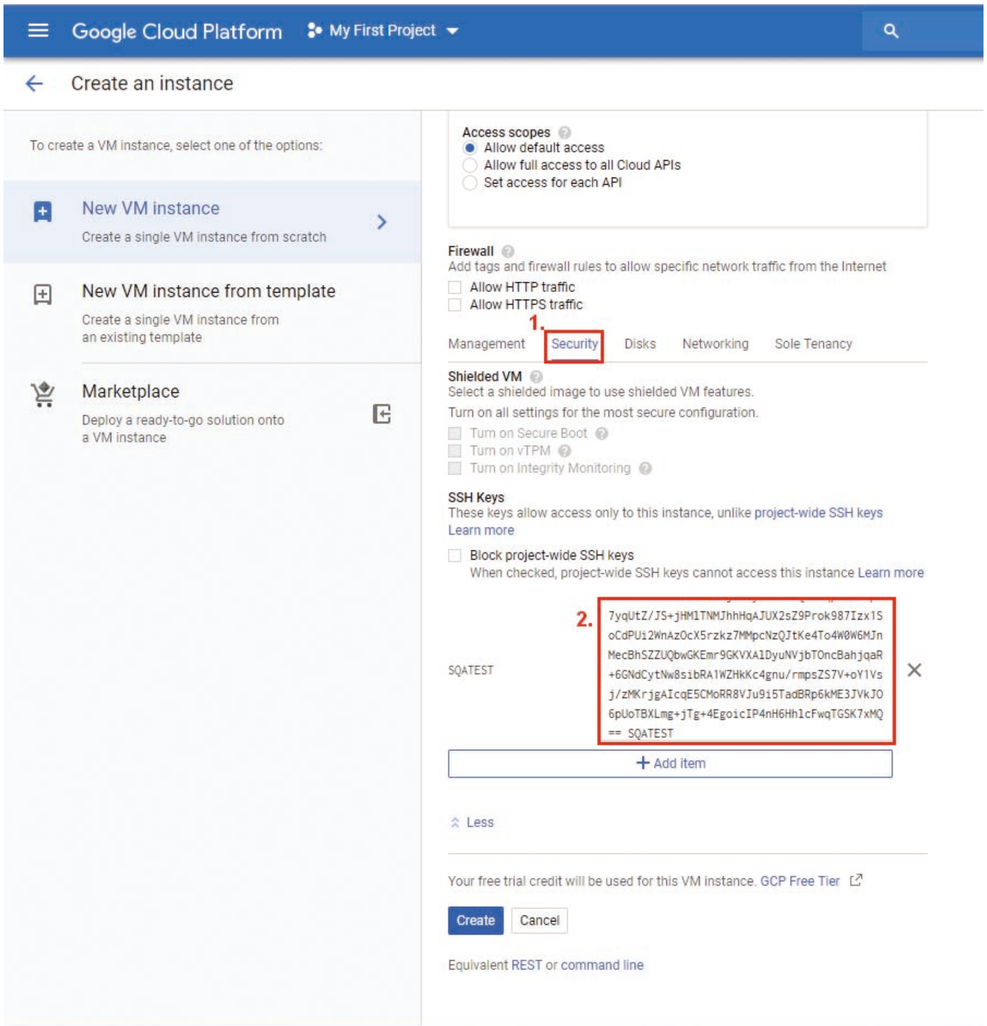
Load an existing private key file

Save the generated key

Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

10. Return to the browser. Choose **Security** and paste the public key into the marked box.



Google Cloud Platform
My First Project
Create an instance
To create a VM instance, select one of the options:
New VM instance
Create a single VM instance from scratch
New VM instance from template
Create a single VM instance from an existing template
Marketplace
Deploy a ready-to-go solution onto a VM instance
1.
Management
Security
Disks
Networking
Sole Tenancy
2.
Add item
Less
Create
Cancel

Access scopes
● Allow default access
○ Allow full access to all Cloud APIs
○ Set access for each API

Firewall
Add tags and firewall rules to allow specific network traffic from the Internet
□ Allow HTTP traffic
□ Allow HTTPS traffic

Shielded VM
Select a shielded image to use shielded VM features.
Turn on all settings for the most secure configuration.
□ Turn on Secure Boot
□ Turn on vTPM
□ Turn on Integrity Monitoring

SSH Keys
These keys allow access only to this instance, unlike project-wide SSH keys Learn more
□ Block project-wide SSH keys
When checked, project-wide SSH keys cannot access this instance Learn more

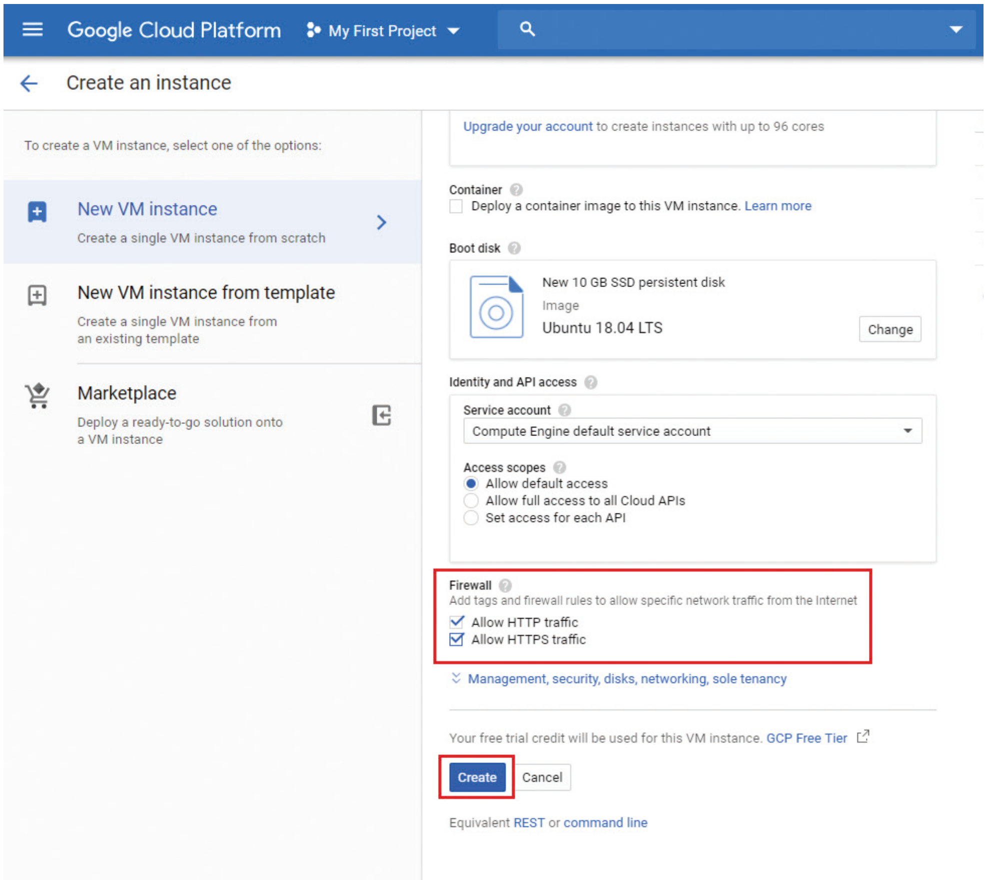
SQATEST

```
7yqUtZ/JS+jHM1TNMJhhHqAJUX2sZ9Prok987Izx1S
oCdPUi2WnAzOcX5rzkz7MMpcNzQJtKe4To4W0W6MJn
MecBhSZZUQbwGKEmr9GKVXAlDyuNVjbTOncBahjqaR
+6GNdCytNw8sibRA1WZHkKc4gnu/rmpsZS7V+oY1Vs
j/zMKrjgAIcqE5CMoRR8VJu9i5TadBRp6kME3JVkJO
6pUoTBXLmg+jTg+4EgoicIP4nH6HhlcFwqTGSK7xMQ
== SQATEST
```

Your free trial credit will be used for this VM instance. GCP Free Tier

Equivalent REST or command line

11. Check both **HTTP** and **HTTPS** boxes in the **Firewall** section and click **Create** below.



To create a VM instance, select one of the options:

- New VM instance — Create a single VM instance from scratch
- New VM instance from template — Create a single VM instance from an existing template
- Marketplace — Deploy a ready-to-go solution onto a VM instance

Upgrade your account to create instances with up to 96 cores

Container

□ Deploy a container image to this VM instance. Learn more

Boot disk

New 10 GB SSD persistent disk
Image
Ubuntu 18.04 LTS
Change

Identity and API access

Service account: Compute Engine default service account

Access scopes

● Allow default access
○ Allow full access to all Cloud APIs
○ Set access for each API

Firewall

Add tags and firewall rules to allow specific network traffic from the Internet

☑ Allow HTTP traffic
☑ Allow HTTPS traffic

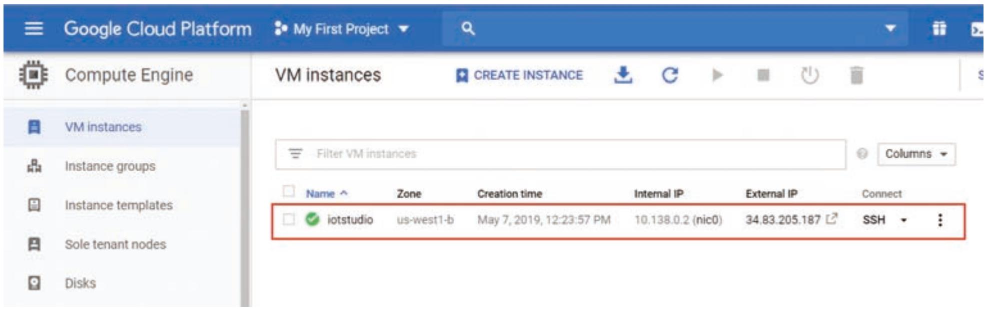
Management, security, disks, networking, sole tenancy

Your free trial credit will be used for this VM instance. GCP Free Tier

Create Cancel

Equivalent REST or command line

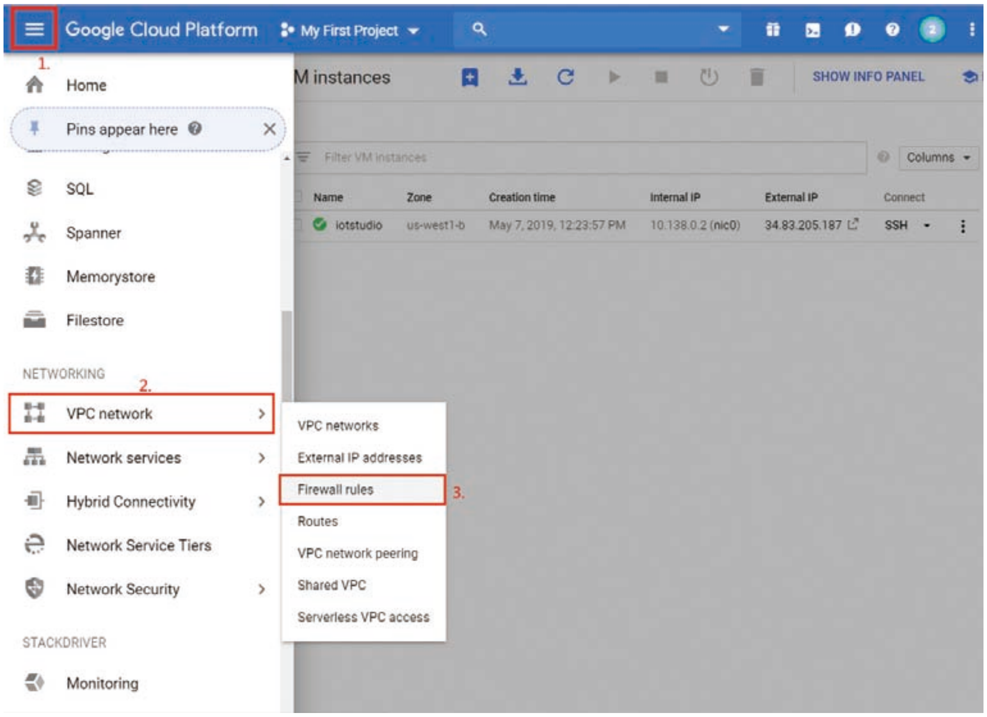
12. The VM has now been created.



VM instances

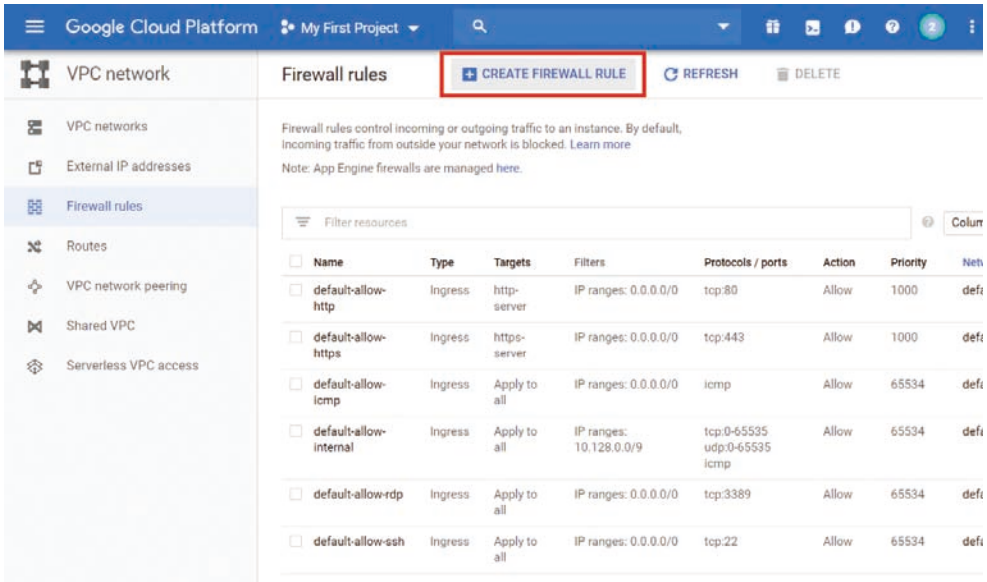
	Name	Zone	Creation time	Internal IP	External IP	Connect
□	iotstudio	us-west1-b	May 7, 2019, 12:23:57 PM	10.138.0.2 (nic0)	34.83.205.187	SSH

13. Click the triple bar icon ≡ and select **VPC network** -> **Firewall rules**.



Name	Zone	Creation time	Internal IP	External IP	Connect
iotstudio	us-west1-b	May 7, 2019, 12:23:57 PM	10.138.0.2 (nic0)	34.83.205.187	SSH

14. Click **CREATE FIREWALL RULE**.

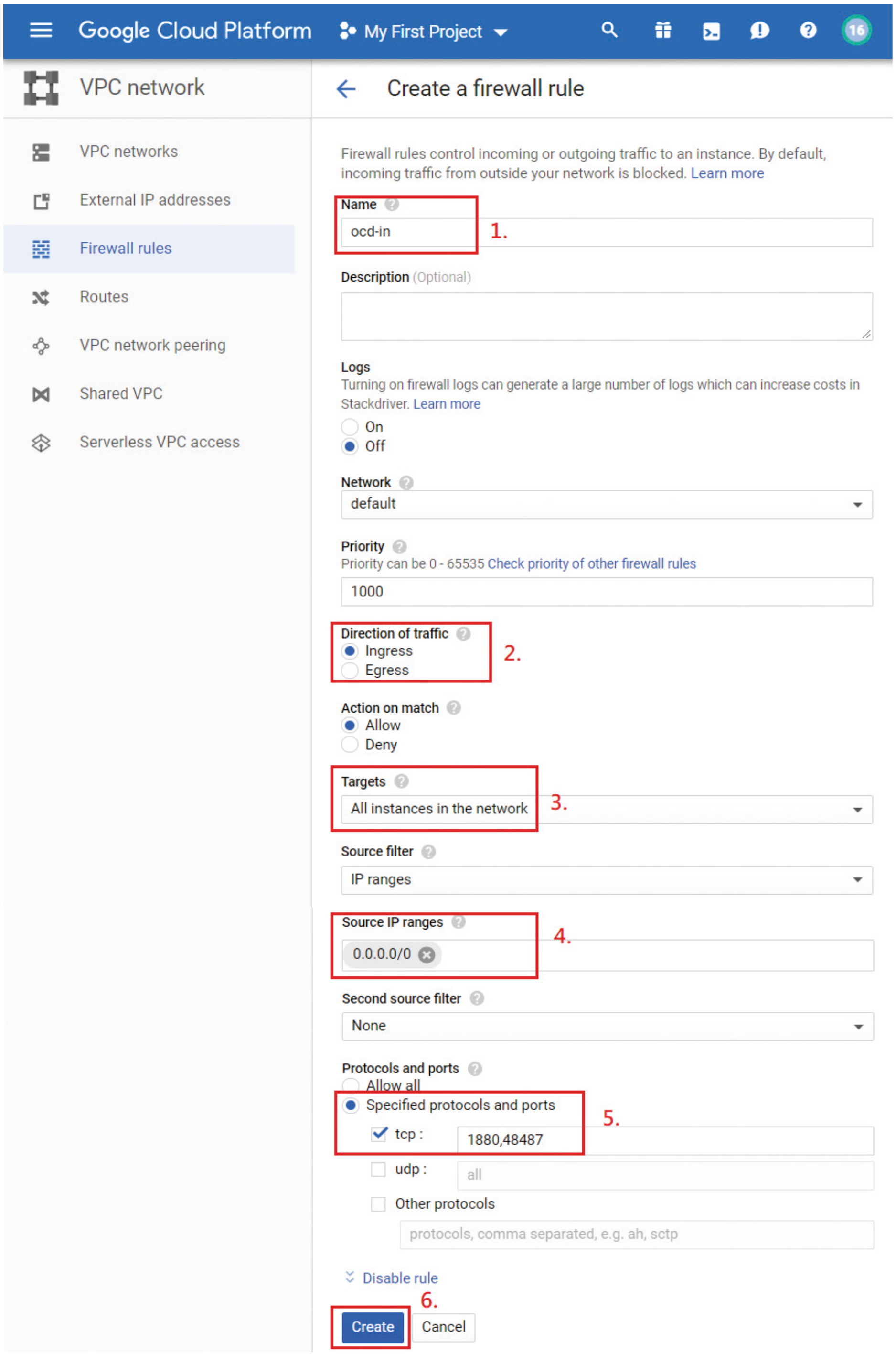


Firewall rules control incoming or outgoing traffic to an instance. By default, incoming traffic from outside your network is blocked. Learn more

Note: App Engine firewalls are managed here.

Name	Type	Targets	Filters	Protocols / ports	Action	Priority	Netw
default-allow-http	Ingress	http-server	IP ranges: 0.0.0.0/0	tcp:80	Allow	1000	defa
default-allow-https	Ingress	https-server	IP ranges: 0.0.0.0/0	tcp:443	Allow	1000	defa
default-allow-icmp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	icmp	Allow	65534	defa
default-allow-internal	Ingress	Apply to all	IP ranges: 10.128.0.0/9	tcp:0-65535 udp:0-65535 icmp	Allow	65534	defa
default-allow-rdp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:3389	Allow	65534	defa
default-allow-ssh	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:22	Allow	65534	defa

15. Name the firewall rule, choose **Ingress** under **Direction of traffic**, select **All instances in the network** under the **Targets** drop-down menu and type **0.0.0.0/0** under **Source IP ranges**. Under **Protocols and ports**, select the second option, check the **tcp** box and enter **1880** and **48487**. (IoT Studio uses the 1880 port, while One-Click Agent uses the 48487 port.)
Lastly, click **Create**.



Firewall rules control incoming or outgoing traffic to an instance. By default, incoming traffic from outside your network is blocked. Learn more

Name: ocd-in

Description (Optional)

Logs
Turning on firewall logs can generate a large number of logs which can increase costs in Stackdriver. Learn more
○ On
● Off

Network: default

Priority
Priority can be 0 - 65535 Check priority of other firewall rules
1000

Direction of traffic
● Ingress
○ Egress

Action on match
● Allow
○ Deny

Targets: All instances in the network

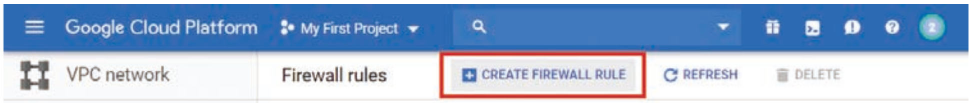
Source filter: IP ranges

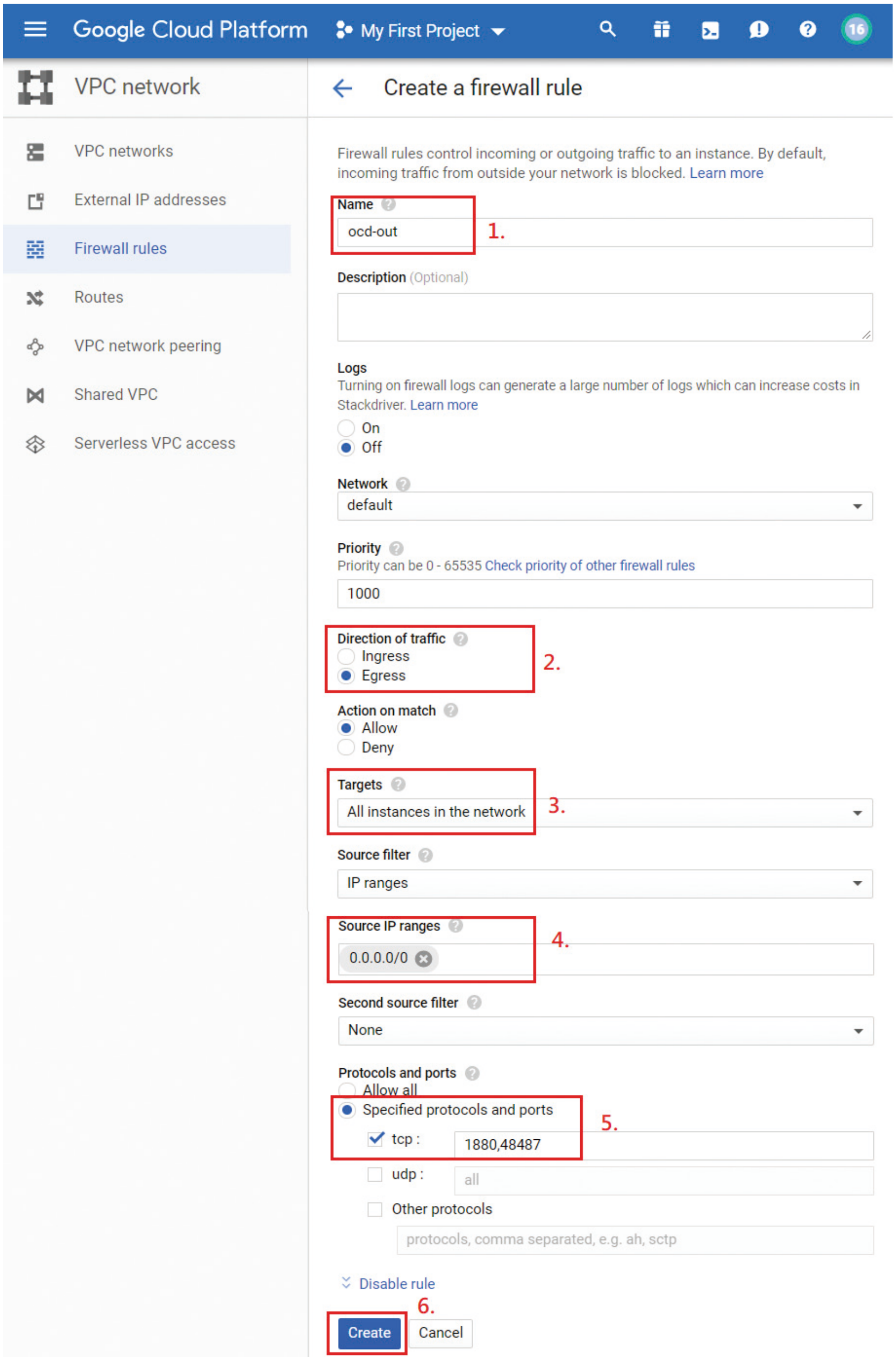
Source IP ranges: 0.0.0.0/0

Second source filter: None

Protocols and ports
○ Allow all
● Specified protocols and ports
☑ tcp : 1880,48487
□ udp : all
□ Other protocols
protocols, comma separated, e.g. ah, sctp

16. Repeat the last two steps, but choose **Egress** under **Direction of traffic**.





Create a firewall rule

Firewall rules control incoming or outgoing traffic to an instance. By default, incoming traffic from outside your network is blocked. Learn more

Name: ocd-out

Description (Optional)

Logs
Turning on firewall logs can generate a large number of logs which can increase costs in Stackdriver. Learn more
○ On
● Off

Network: default

Priority
Priority can be 0 - 65535 Check priority of other firewall rules
1000

Direction of traffic
○ Ingress
● Egress

Action on match
● Allow
○ Deny

Targets: All instances in the network

Source filter: IP ranges

Source IP ranges: 0.0.0.0/0

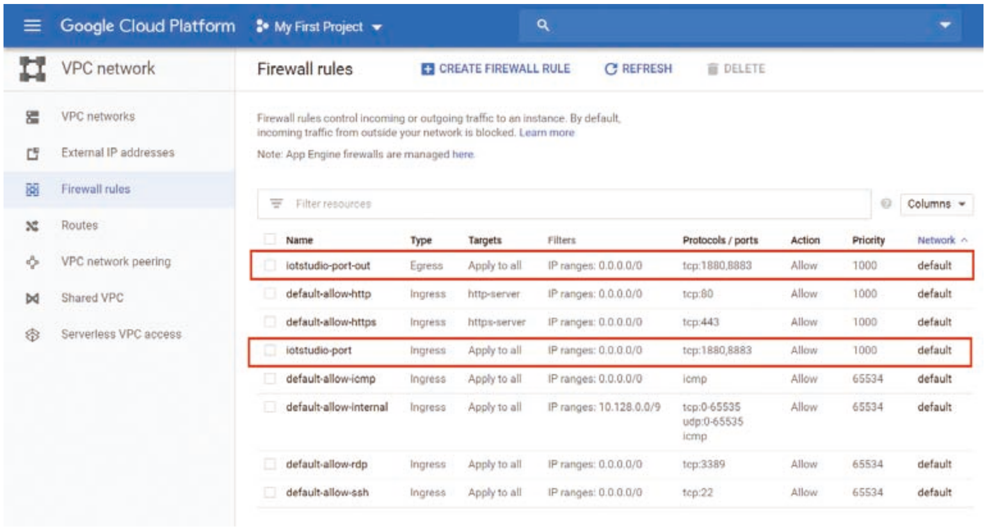
Second source filter: None

Protocols and ports
○ Allow all
● Specified protocols and ports
☑ tcp : 1880,48487
□ udp : all
□ Other protocols
protocols, comma separated, e.g. ah, sctp

Disable rule

Create Cancel

17. You'll see the results below.



Firewall rules

Firewall rules control incoming or outgoing traffic to an instance. By default, incoming traffic from outside your network is blocked. Learn more

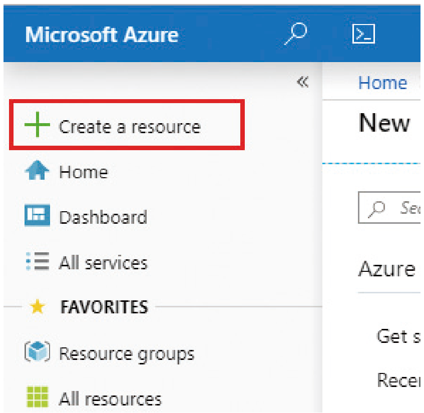
Note: App Engine firewalls are managed here.

Name	Type	Targets	Filters	Protocols / ports	Action	Priority	Network
iotstudio-port-out	Egress	Apply to all	IP ranges: 0.0.0.0/0	tcp:1880,8883	Allow	1000	default
default-allow-http	Ingress	http-server	IP ranges: 0.0.0.0/0	tcp:80	Allow	1000	default
default-allow-https	Ingress	https-server	IP ranges: 0.0.0.0/0	tcp:443	Allow	1000	default
iotstudio-port	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:1880,8883	Allow	1000	default
default-allow-icmp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	icmp	Allow	65534	default
default-allow-internal	Ingress	Apply to all	IP ranges: 10.128.0.0/9	tcp:0-65535 udp:0-65535 icmp	Allow	65534	default
default-allow-rdp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:3389	Allow	65534	default
default-allow-ssh	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:22	Allow	65534	default

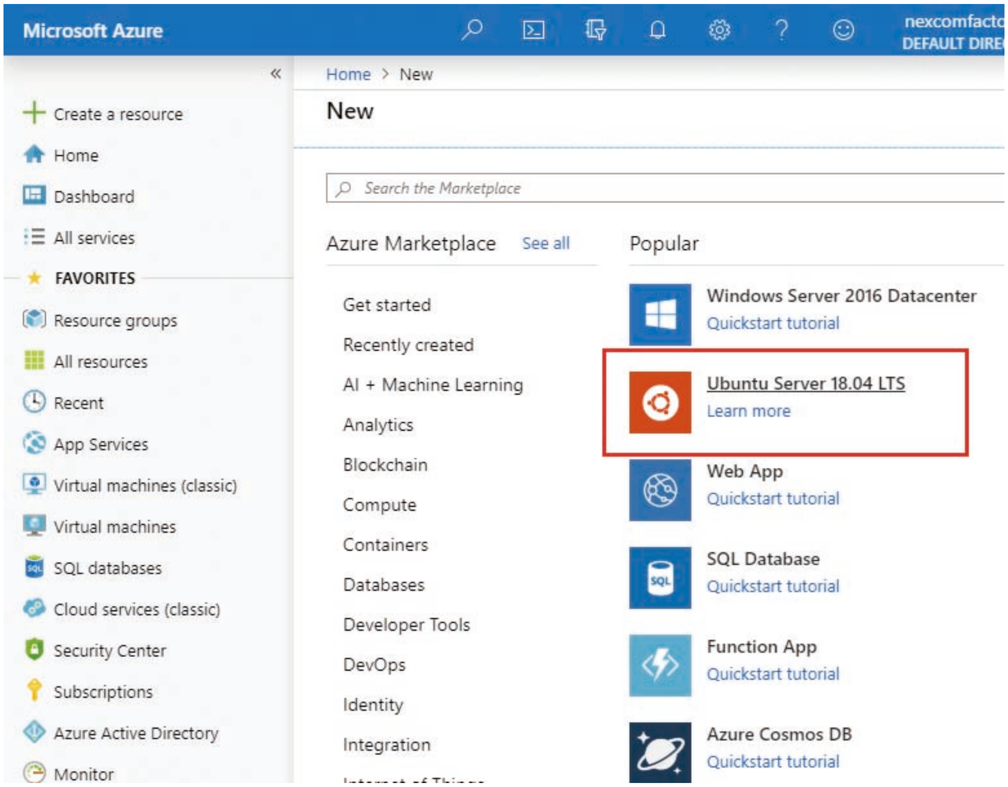
18. Return to the NexAIoT One Click Deploy user menu to continue the settings.

Appendix B: Create a Virtual Machine for IoT Studio in Azure Cloud

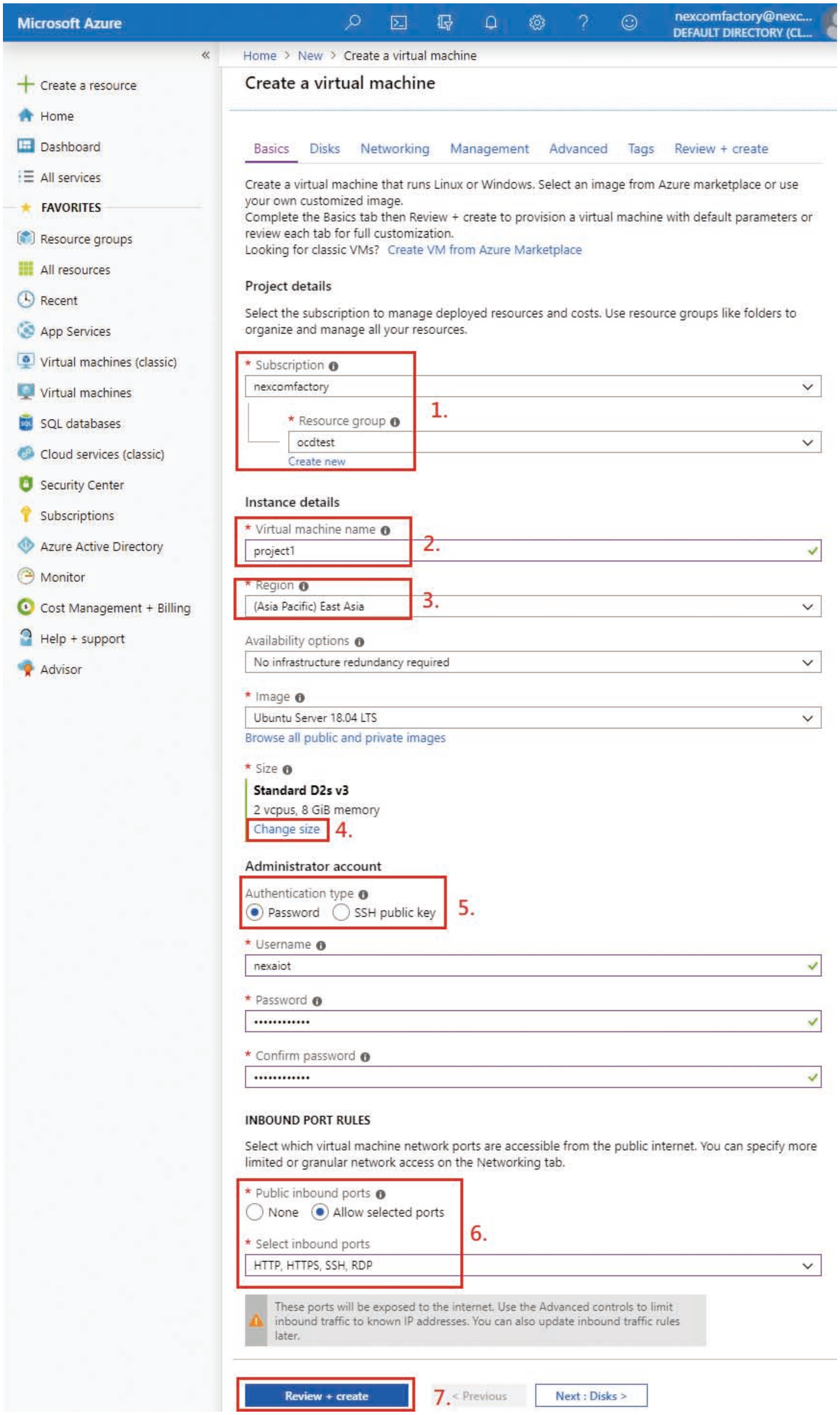
1. Log in to the Azure portal and click **Create a resource**.



2. Search for "Ubuntu Server" and click **Ubuntu Server 18.04 LTS**.



3. Fill in the fields as shown below:



Create a virtual machine

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image.
Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization.
Looking for classic VMs? Create VM from Azure Marketplace

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription: nexcomfactory

* Resource group: ocdtest

Create new

Instance details

* Virtual machine name: project1

* Region: (Asia Pacific) East Asia

Availability options: No infrastructure redundancy required

* Image: Ubuntu Server 18.04 LTS

Browse all public and private images

* Size: **Standard D2s v3**
2 vcpus, 8 GiB memory
Change size

Administrator account

Authentication type: ● Password ○ SSH public key

* Username: nexaiot

* Password: ••••••••••••

* Confirm password: ••••••••••••

INBOUND PORT RULES

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

* Public inbound ports: ○ None ● Allow selected ports

* Select inbound ports: HTTP, HTTPS, SSH, RDP

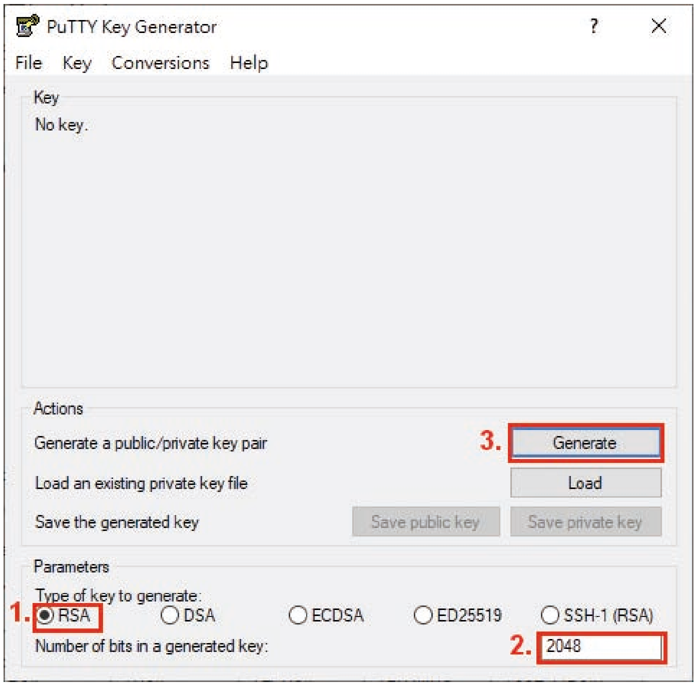
These ports will be exposed to the internet. Use the Advanced controls to limit inbound traffic to known IP addresses. You can also update inbound traffic rules later.

- Select your Subscription and Resource groups under their respective drop-down menus. You can click **Create new** if you'd like to create one.
- Under **Instance details**, input a name for **Virtual machine name**.
- Click **Change size** and then select the VM size according to your requirement.

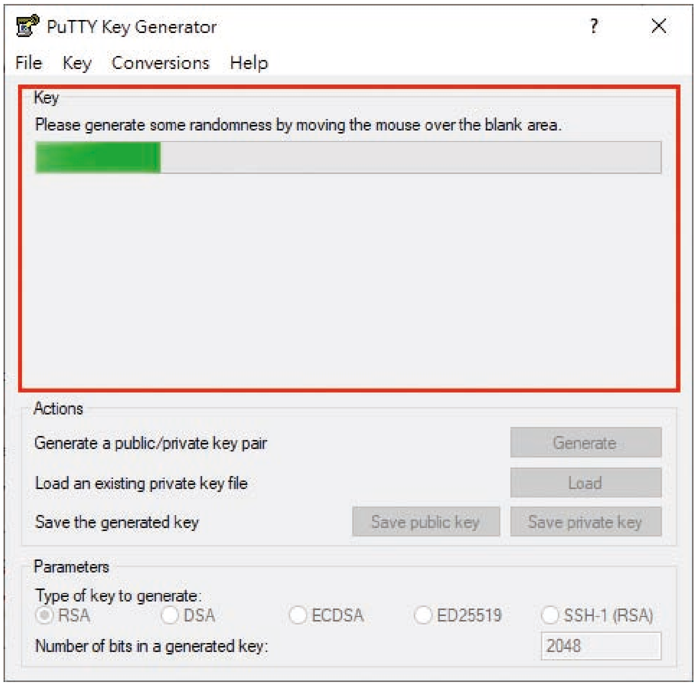
  Note: Please set **at least** 2 vCPU cores and 4GB memory. It's recommended to use 2 vCPU cores and 8GB memory. The speed of git clone and data deployment depends on the specifications of the VM and Internet environment.

- Under **Administrator account**, provide a username and password. Please remember it!
  For the **Authentication type**, besides **Password**, you can choose **SSH public key**, which accesses the VM more securely.
  Generate SSH key pair through the **puttygen** application, which you can download at: https://puttygen.com/download.php?val=46

  i. Open **puttygen**. Choose **RSA** under **Type of key to generate** and type **2048** in the **Number of bits in a generated key** text box. Then, click **Generate**.



ii. Move the mouse randomly in the red box until the progress is complete.



Please generate some randomness by moving the mouse over the blank area.

Generate a public/private key pair

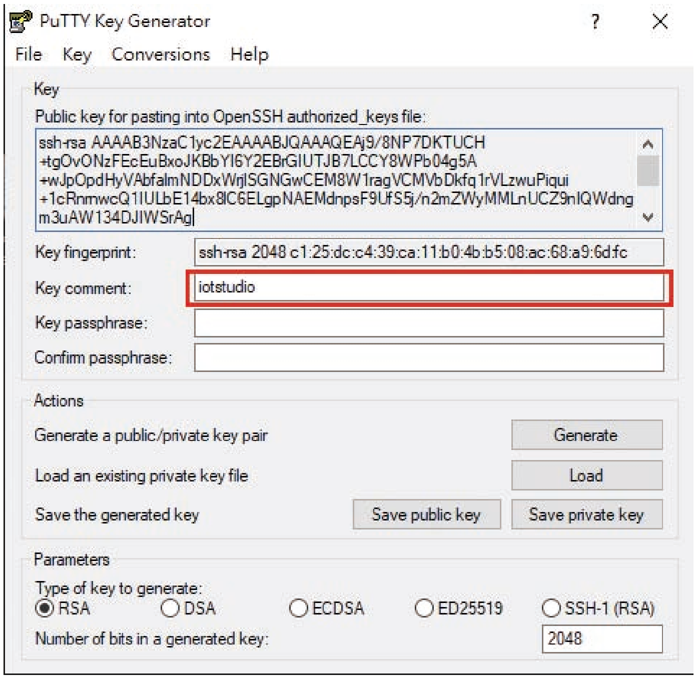
Load an existing private key file

Save the generated key

Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

iii. You need to modify the **Key comment** (do not use any special symbols) as a remark for your public key. It will affect the composition of the key. For example, change the default text from "rsa-key-190520" to "iotstudio."



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAj9/8NP7DKTUCH
+tgOvONzFEcEuBxoJKBbYI6Y2EBrGIUTJB7LCCY8WPb04g5A
+wJpOpdHyVAbfalmNDDxWrjISGNGwCEM8W1ragVCMVbDkfq1rVLzwuPiqui
+1cRnmwcQ1IULbE14bx8lC6ELgpNAEMdnpsF9UfS5j/n2mZWyMMLnUCZ9nIQWdng
m3uAW134DJIWSrAg
```

Key fingerprint: ssh-rsa 2048 c1:25:dc:c4:39:ca:11:b0:4b:b5:08:ac:68:a9:6d:fc

Key comment: iotstudio

Key passphrase:

Confirm passphrase:

Generate a public/private key pair

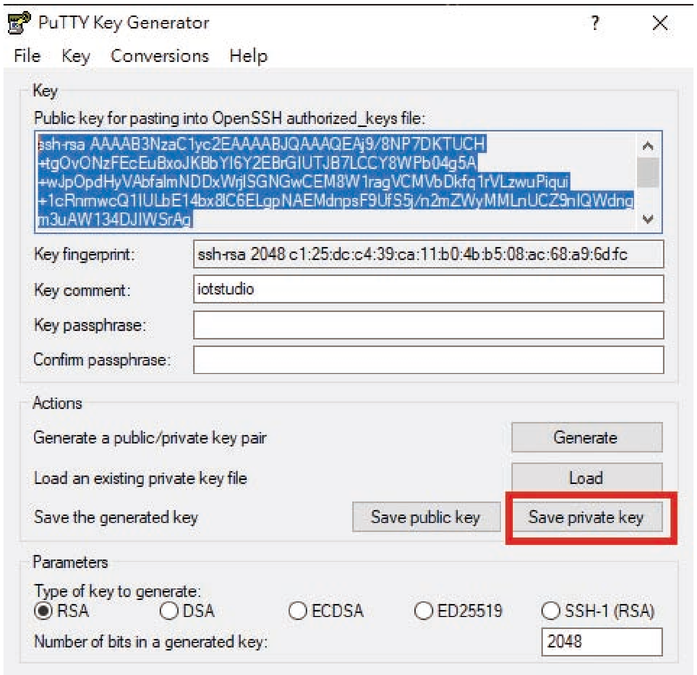
Load an existing private key file

Save the generated key

Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

iv. Save the private key, which will be used in IoT Studio's One Click Configuration.



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAj9/8NP7DKTUCH
+tgOvONzFEcEuBxoJKBbYI6Y2EBrGIUTJB7LCCY8WPb04g5A
+wJpOpdHyVAbfalmNDDxWrjlSGNGwCEM8W1ragVCMVbDkfq1rVLzwuPiqui
+1cRnmwcQ1IULbE14bx8lC6ELgpNAEMdnpsF9UfS5j/n2mZWyMMLnUCZ9nIQWdng
m3uAW134DJIWSrAg
```

Key fingerprint: ssh-rsa 2048 c1:25:dc:c4:39:ca:11:b0:4b:b5:08:ac:68:a9:6d:fc

Key comment: iotstudio

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

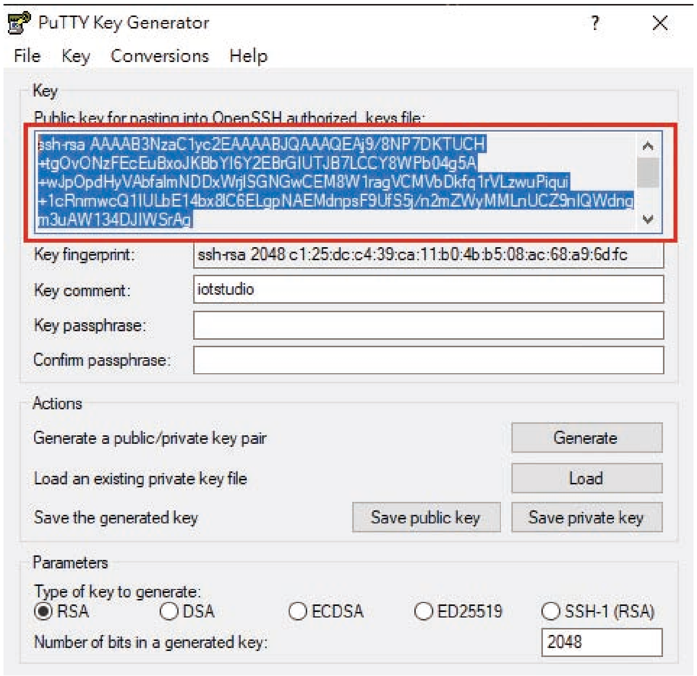
Parameters

Type of key to generate:

● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

v. Copy the **public key** as shown in the red box below.



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAj9/8NP7DKTUCH
+tgOvONzFEcEuBxoJKBbYI6Y2EBrGIUTJB7LCCY8WPb04g5A
+wJpOpdHyVAbfalmNDDxWrjlSGNGwCEM8W1ragVCMVbDkfq1rVLzwuPiqui
+1cRnmwcQ1IULbE14bx8lC6ELgpNAEMdnpsF9UfS5j/n2mZWyMMLnUCZ9nIQWdng
m3uAW134DJIWSrAg
```

Key fingerprint: ssh-rsa 2048 c1:25:dc:c4:39:ca:11:b0:4b:b5:08:ac:68:a9:6d:fc

Key comment: iotstudio

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

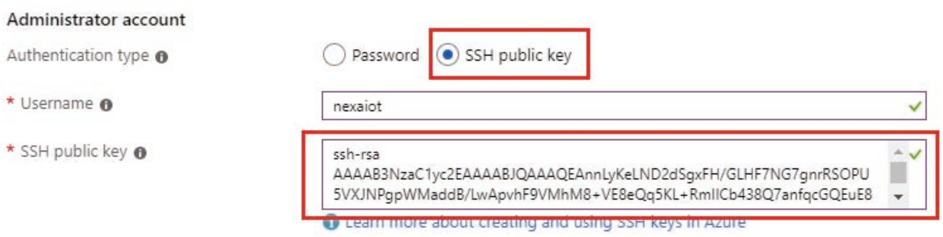
Parameters

Type of key to generate:

● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

vi. Return to the browser and paste the public key to **SSH public key**.



Administrator account

Authentication type ○ Password ● SSH public key

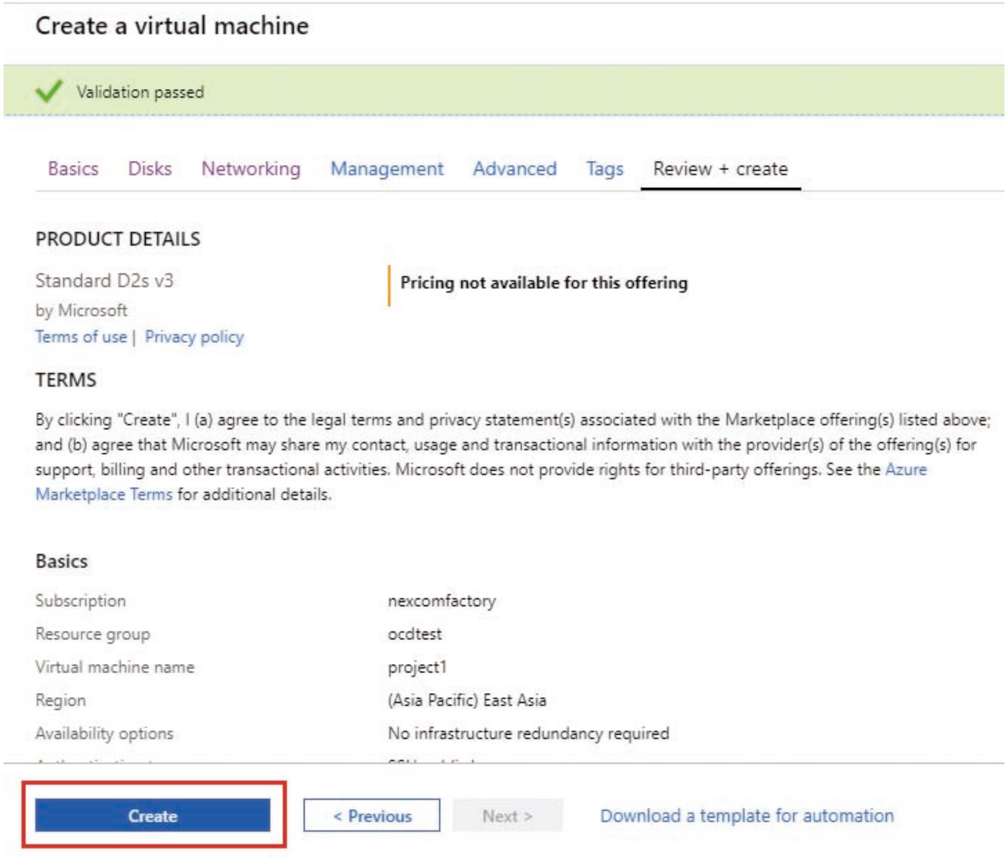
* Username: nexaiot ✓

* SSH public key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAnnLyKeLND2dSgxFH/GLHF7NG7gnrRSOPU
5VXJNPgpWMaddB/LwApvhF9VMhM8+VE8eQq5KL+RmlICb438Q7anfqcGQEuE8
```

- For the **INBOUND PORT RULES**, select **Allow selected ports** at **Public inbound ports** and check all **Select inbound ports**.
- Keep the rest of the columns as default.

4. Go to **Review + create** tab and double check all the configurations. Then click **Create** to complete setup.



Create a virtual machine

✓ Validation passed

PRODUCT DETAILS

Standard D2s v3
by Microsoft
Terms of use | Privacy policy

Pricing not available for this offering

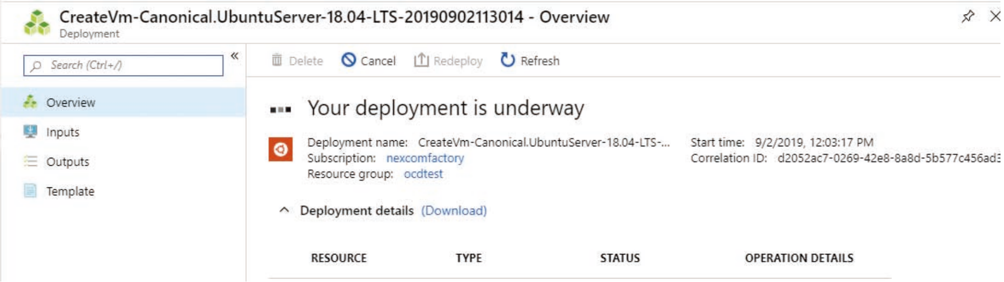
TERMS

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; and (b) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the Azure Marketplace Terms for additional details.

Basics

Subscription	nexcomfactory
Resource group	ocdtest
Virtual machine name	project1
Region	(Asia Pacific) East Asia
Availability options	No infrastructure redundancy required

Wait until the Azure portal finishes building.



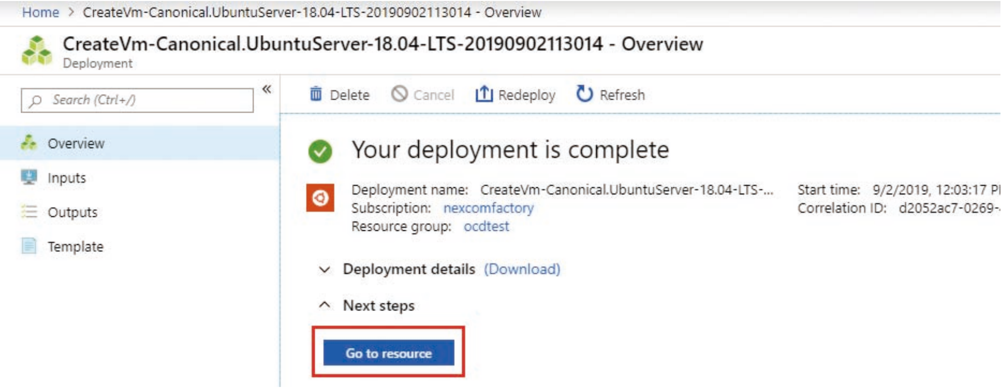
Your deployment is underway

Deployment name: CreateVm-Canonical.UbuntuServer-18.04-LTS-...
Subscription: nexcomfactory
Resource group: ocdtest
Start time: 9/2/2019, 12:03:17 PM
Correlation ID: d2052ac7-0269-42e8-8a8d-5b577c456ad3

Deployment details (Download)

RESOURCE	TYPE	STATUS	OPERATION DETAILS

Click **Go to resource** when complete.


Home > CreateVm-Canonical.UbuntuServer-18.04-LTS-20190902113014 - Overview
CreateVm-Canonical.UbuntuServer-18.04-LTS-20190902113014 - Overview
Deployment
Search (Ctrl+/)
Overview
Inputs
Outputs
Template
Delete
Cancel
Redeploy
Refresh

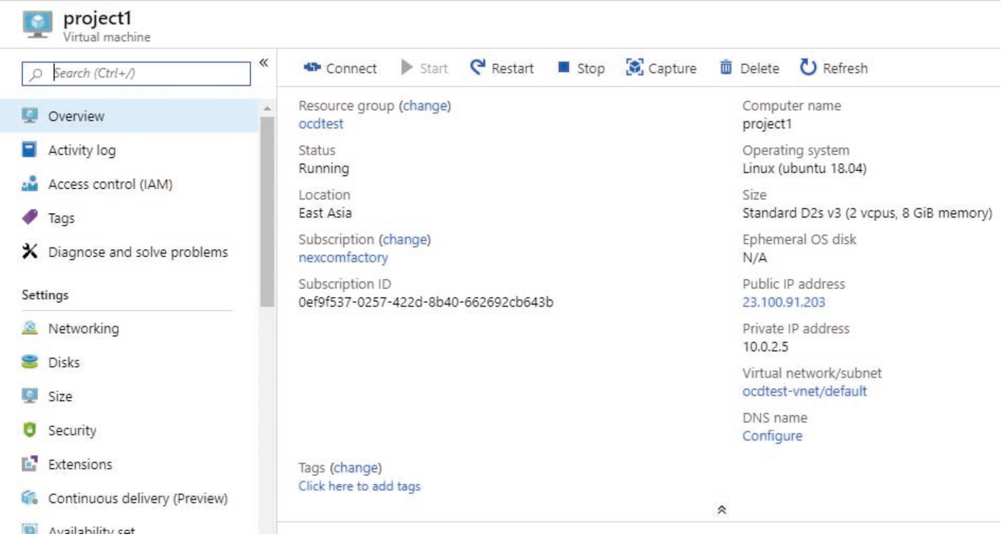
Your deployment is complete

Deployment name: CreateVm-Canonical.UbuntuServer-18.04-LTS-...
Subscription: nexcomfactory
Resource group: ocdtest
Start time: 9/2/2019, 12:03:17 PI
Correlation ID: d2052ac7-0269-

Deployment details (Download)

Next steps

You'll see the VM's settings page.



Resource group (change)
ocdtest

Status
Running

Location
East Asia

Subscription (change)
nexcomfactory

Subscription ID
0ef9f537-0257-422d-8b40-662692cb643b

Computer name
project1

Operating system
Linux (ubuntu 18.04)

Size
Standard D2s v3 (2 vcpus, 8 GiB memory)

Ephemeral OS disk
N/A

Public IP address
23.100.91.203

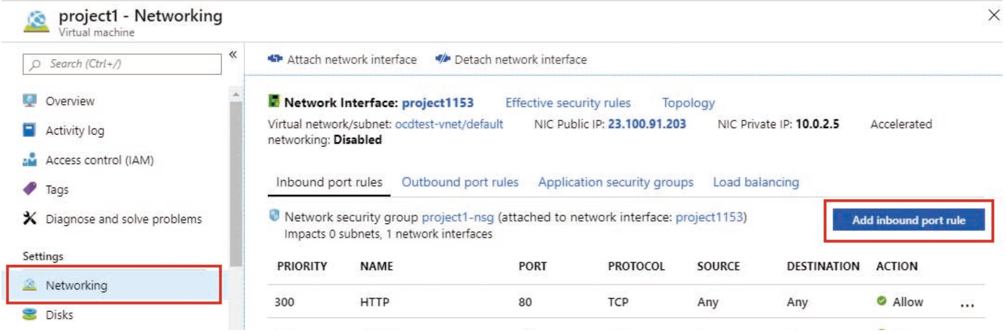
Private IP address
10.0.2.5

Virtual network/subnet
ocdtest-vnet/default

DNS name
Configure

Tags (change)
Click here to add tags

5. Next, open the ports for IoT Studio and One-Click Agent.
   Click **Networking** in the left-hand menu and click the **Add inbound port rule** button on the right side of the page.



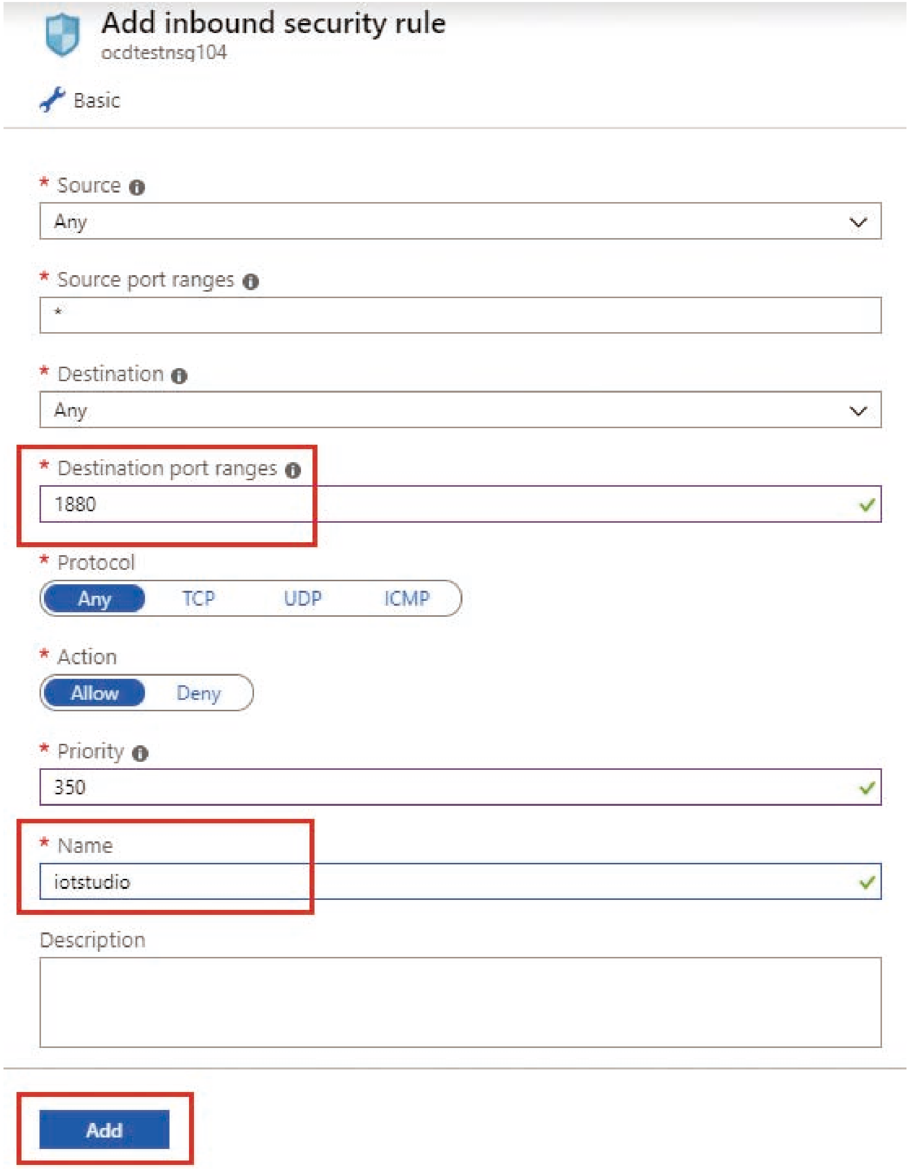
project1 - Networking
Virtual machine

Network Interface: project1153   Effective security rules   Topology
Virtual network/subnet: ocdtest-vnet/default   NIC Public IP: **23.100.91.203**   NIC Private IP: **10.0.2.5**   Accelerated networking: **Disabled**

Network security group project1-nsg (attached to network interface: project1153)
Impacts 0 subnets, 1 network interfaces

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION
300	HTTP	80	TCP	Any	Any	Allow

6. Add the port number **1880** for IoT Studio and **48487** for One-Click Agent.



Add inbound security rule
ocdtestnsg104

* Source: Any

* Source port ranges: *

* Destination: Any

* Destination port ranges: 1880

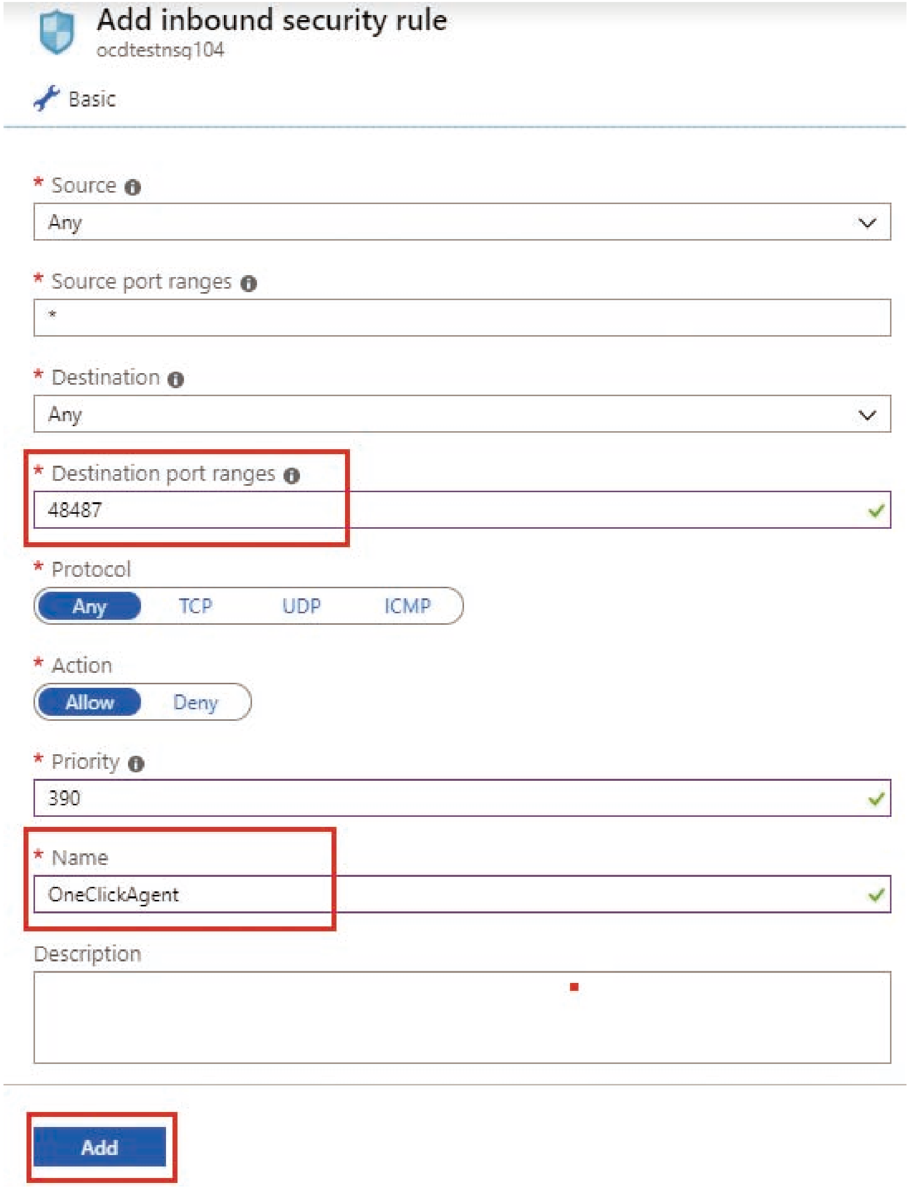
* Protocol: Any TCP UDP ICMP

* Action: Allow Deny

* Priority: 350

* Name: iotstudio

Description:



Add inbound security rule
ocdtestnsg104

* Source: Any

* Source port ranges: *

* Destination: Any

* Destination port ranges: 48487

* Protocol: [✓ Any] TCP UDP ICMP

* Action: [✓ Allow] Deny

* Priority: 390

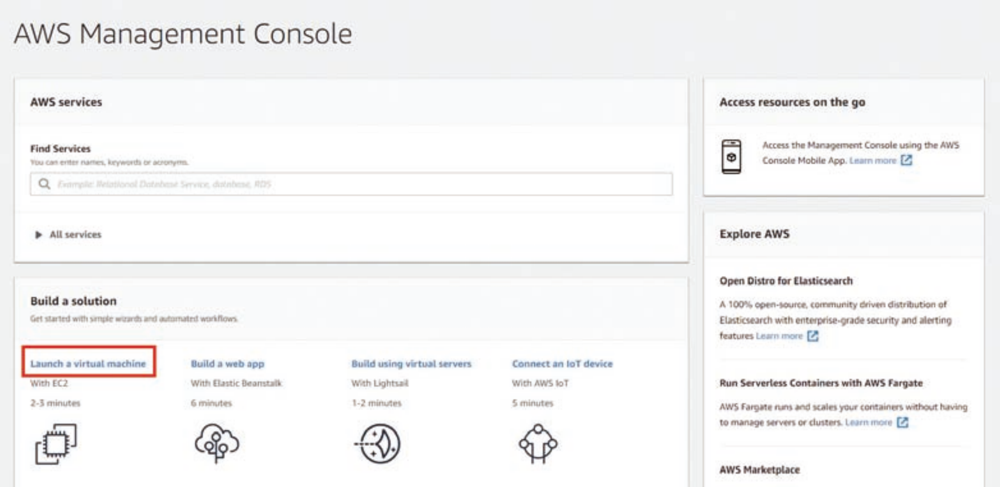
* Name: OneClickAgent

Description:

7. Return to the NexAIoT One Click Deploy user menu to continue the settings.

Appendix C: Create a Virtual Machine for IoT Studio in AWS Cloud

1. Log in to AWS Management Console and click **Launch a virtual machine**.
   https://console.aws.amazon.com/console/home



AWS Management Console

AWS services

Find Services
You can enter names, keywords or acronyms.

Example: Relational Database Service, database, RDS

▶ All services

Build a solution
Get started with simple wizards and automated workflows.

Launch a virtual machine
With EC2
2-3 minutes

Build a web app
With Elastic Beanstalk
6 minutes

Build using virtual servers
With Lightsail
1-2 minutes

Connect an IoT device
With AWS IoT
5 minutes

Access resources on the go

Access the Management Console using the AWS Console Mobile App. Learn more

Explore AWS

Open Distro for Elasticsearch

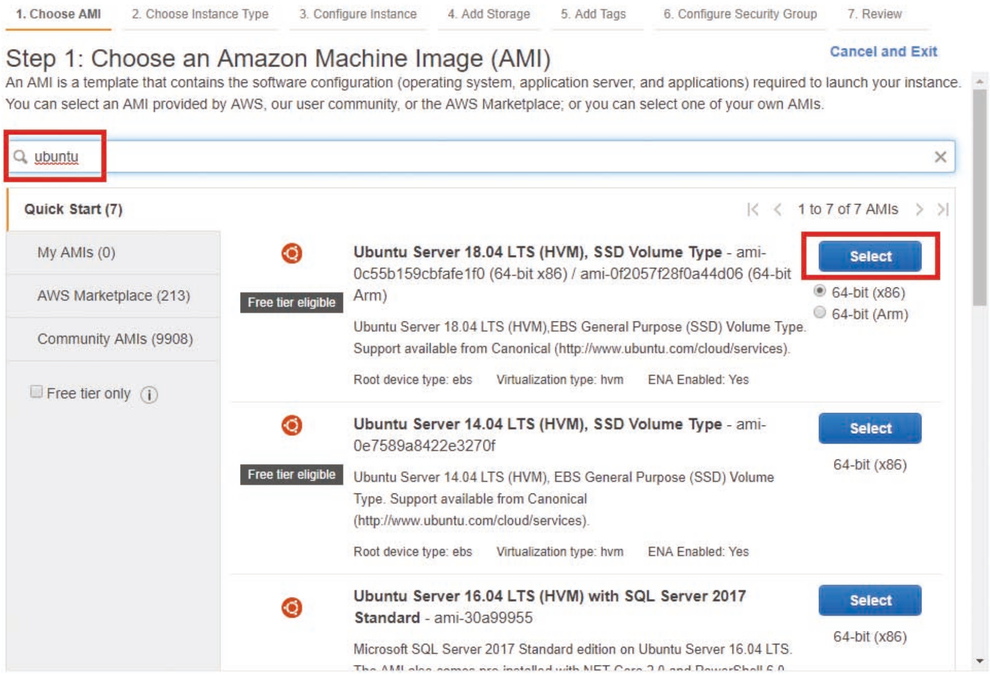
A 100% open-source, community driven distribution of Elasticsearch with enterprise-grade security and alerting features Learn more

Run Serverless Containers with AWS Fargate

AWS Fargate runs and scales your containers without having to manage servers or clusters. Learn more

AWS Marketplace

2. Search for “Ubuntu” and click **Select** to the right of the result **Ubuntu Server 18.04 LTS**.



Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

ubuntu

Quick Start (7)

My AMIs (0)

AWS Marketplace (213)

Community AMIs (9908)

□ Free tier only

1 to 7 of 7 AMIs

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0c55b159cbfafe1f0 (64-bit x86) / ami-0f2057f28f0a44d06 (64-bit Arm)

Free tier eligible

Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

● 64-bit (x86)
○ 64-bit (Arm)

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-0e7589a8422e3270f

Free tier eligible

Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)

Ubuntu Server 16.04 LTS (HVM) with SQL Server 2017 Standard - ami-30a99955

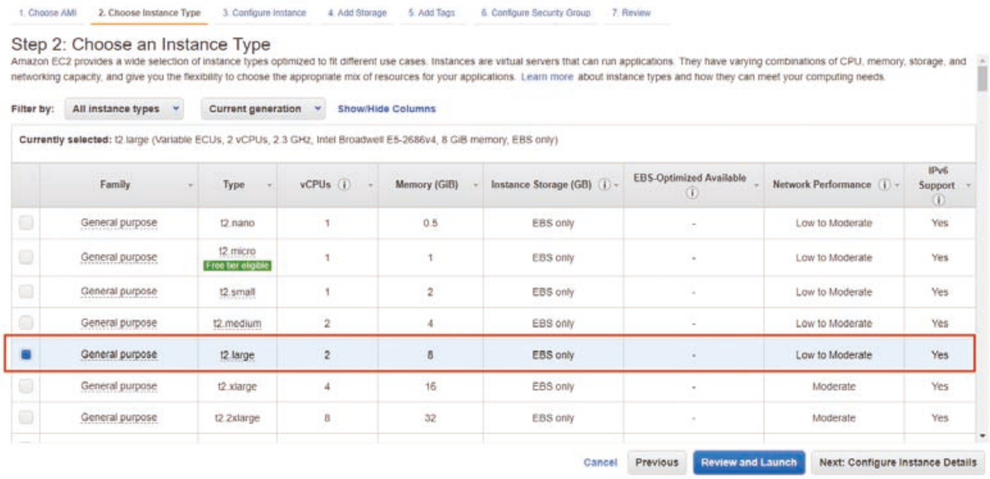
Microsoft SQL Server 2017 Standard edition on Ubuntu Server 16.04 LTS.

Select

64-bit (x86)

3. Choose an instance type, which comprises varying combinations of CPU, memory, storage, and networking capacity. Please choose the appropriate mix for your applications.

   Note: Please set **at least** 2 vCPU cores and 4GB memory. It's recommended to use 2 vCPU cores and 8GB memory. The speed of git clone and data deployment depends on the specifications of the VM and Internet environment.



Step 2: Choose an Instance Type

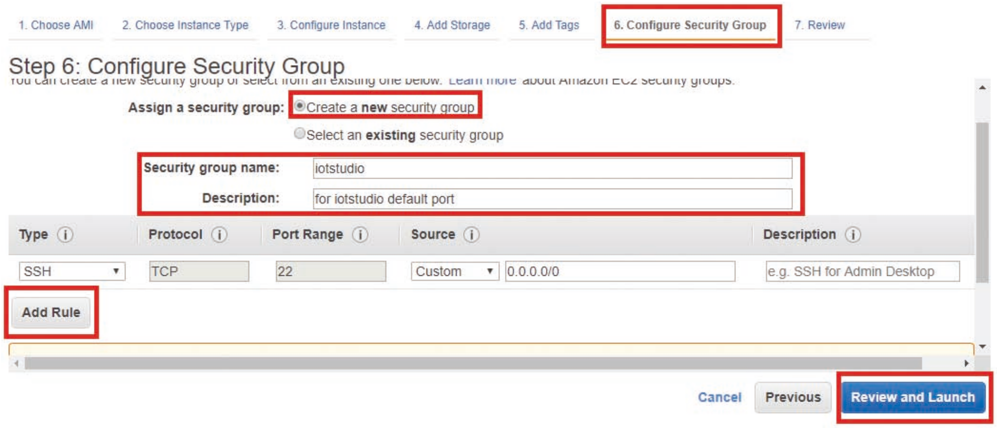
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.large (Variable ECUs, 2 vCPUs, 2.3 GHz, Intel Broadwell E5-2686v4, 8 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
■	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
☐	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

4. Change the tab to **Configure Security Group**. Choose **Create a new security group** and name it. Click **Add Rule** to set the required protocols for IoT Studio.



Step 6: Configure Security Group

You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

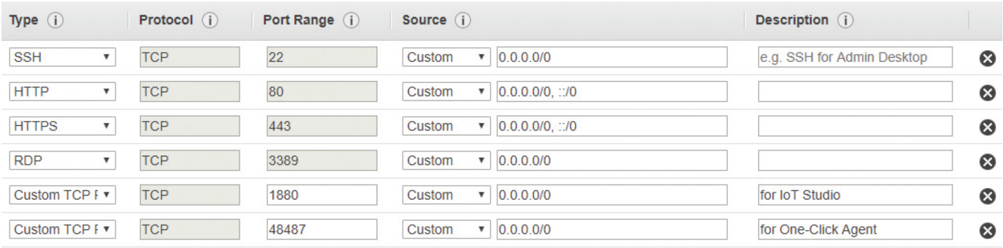
Assign a security group: ● Create a **new** security group
○ Select an **existing** security group

Security group name: iotstudio

Description: for iotstudio default port

Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop

Besides the default value SSH, add the following protocols:

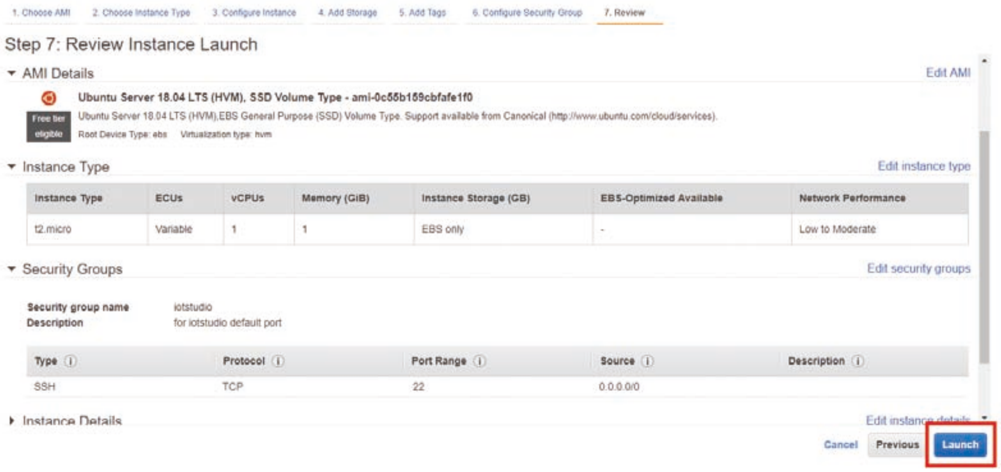


Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom	0.0.0.0/0, ::/0	
HTTPS	TCP	443	Custom	0.0.0.0/0, ::/0	
RDP	TCP	3389	Custom	0.0.0.0/0	
Custom TCP F	TCP	1880	Custom	0.0.0.0/0	for IoT Studio
Custom TCP F	TCP	48487	Custom	0.0.0.0/0	for One-Click Agent

(1880 for IoT Studio, 48487 for One-Click Agent)

Lastly, click **Review and Launch**.

5. Review your instance launch details and click **Launch** to assign a key pair to your instance.



Step 7: Review Instance Launch

AMI Details

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0c55b159cbfafe1f0

Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Root Device Type: ebs Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

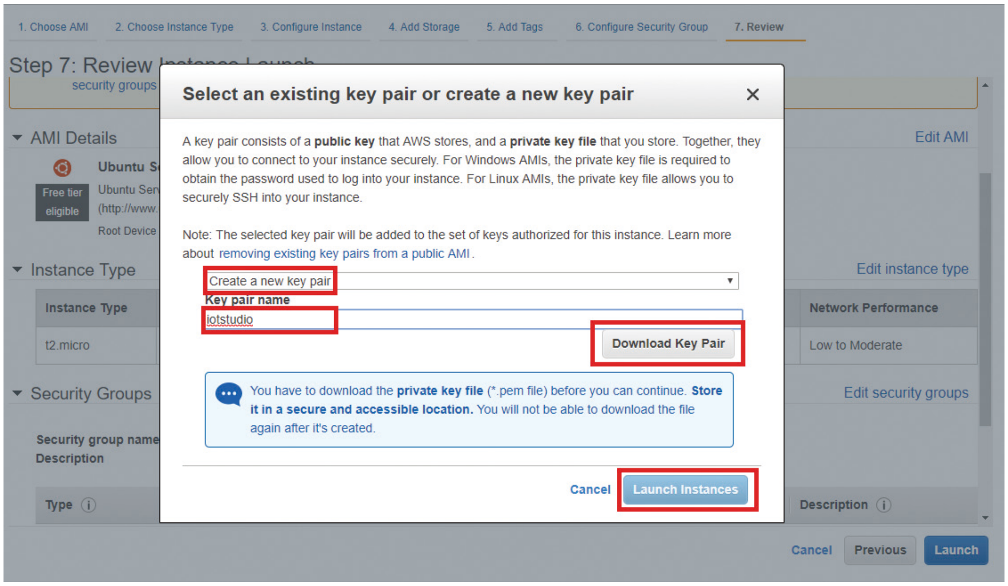
Security Groups

Security group name iotstudio
Description for iotstudio default port

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	

Instance Details

6. Select **Create a new key pair** in the drop-down menu and name it. Click **Download Key Pair**. Store the key file in a secure and accessible location. You will not be able to download the file again after it's created. Click **Launch Instances** to complete the launch process.



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

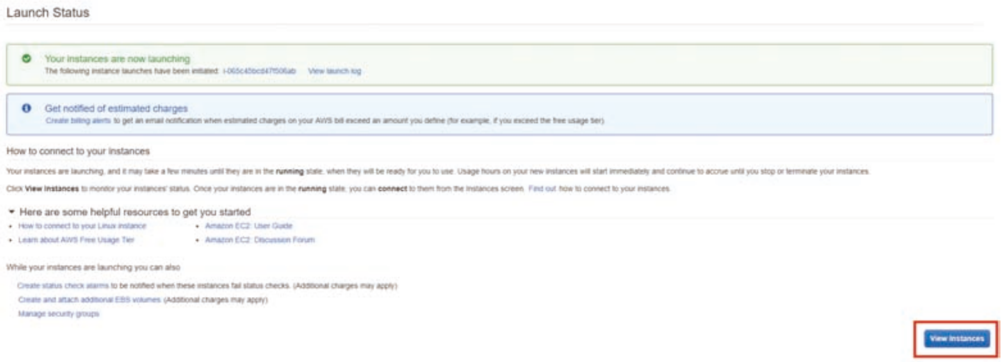
Key pair name

iotstudio

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

The key pair consists of a public key that AWS stores and a private key that the user stores. With this cryptography, you can access the VM securely.

7. Click **View Instances** to manage your instance.


Launch Status

Your instances are now launching
The following instance launches have been initiated: i-065c45bcd47f506ab View launch log

Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View Instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances screen. Find out how to connect to your instances.

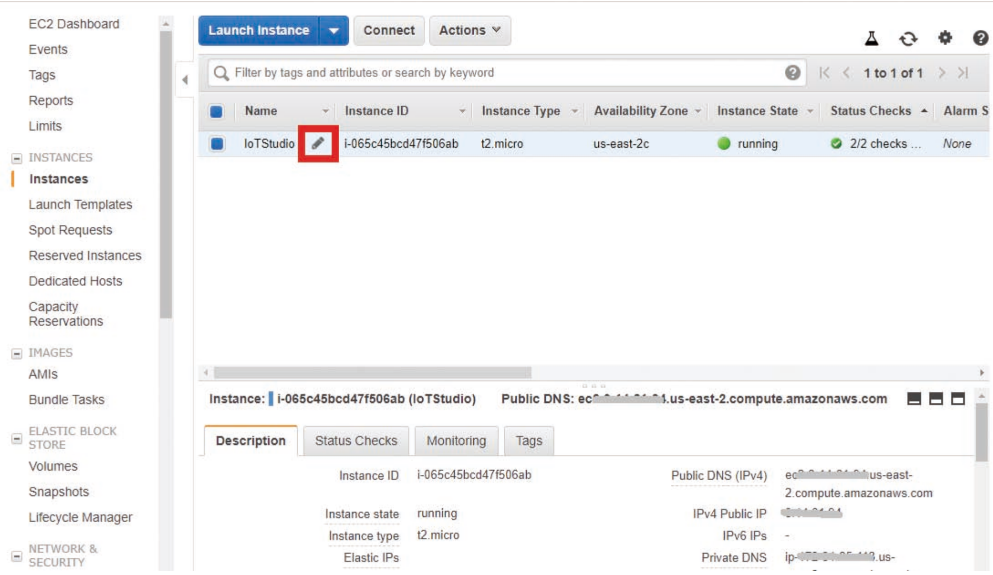
Here are some helpful resources to get you started

- How to connect to your Linux instance
- Learn about AWS Free Usage Tier
- Amazon EC2: User Guide
- Amazon EC2: Discussion Forum

While your instances are launching you can also

Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)
Create and attach additional EBS volumes (Additional charges may apply)
Manage security groups

8. You can monitor your instances' statuses in this page. Once your instance is in the **running** state and the **Status checks** have passed, you can connect to it. Click the edit button to rename your instance.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm S
IoTStudio	i-065c45bcd47f506ab	t2.micro	us-east-2c	● running	✓ 2/2 checks ...	None

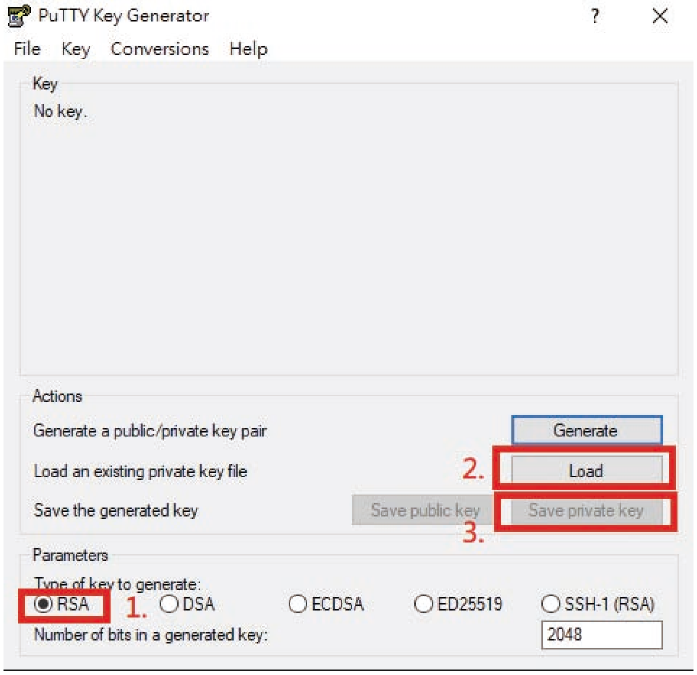
Instance: i-065c45bcd47f506ab (IoTStudio) Public DNS: ec[redacted].us-east-2.compute.amazonaws.com

Instance ID i-065c45bcd47f506ab
Instance state running
Instance type t2.micro
Elastic IPs

Public DNS (IPv4) ec[redacted]us-east-2.compute.amazonaws.com
IPv4 Public IP [redacted]
IPv6 IPs -
Private DNS ip-[redacted].us-

9. Prepare the puttygen application, which you can download at: https://puttygen.com/download.php?val=46

10. After installation is complete, open **puttygen** to convert the key type from **.pem** to **.ppk**. Under **Type of key to generate**, choose **RSA**. Click **Load** to upload the key pair you downloaded in step 6. Click **Save private key** and save the converted key file.



Key

No key.

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

Parameters

Type of key to generate:

● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

Note that the private key will be used in IoT Studio's One Click Configuration.

11. Return to the NexAIoT One Click Deploy user menu to continue the settings.