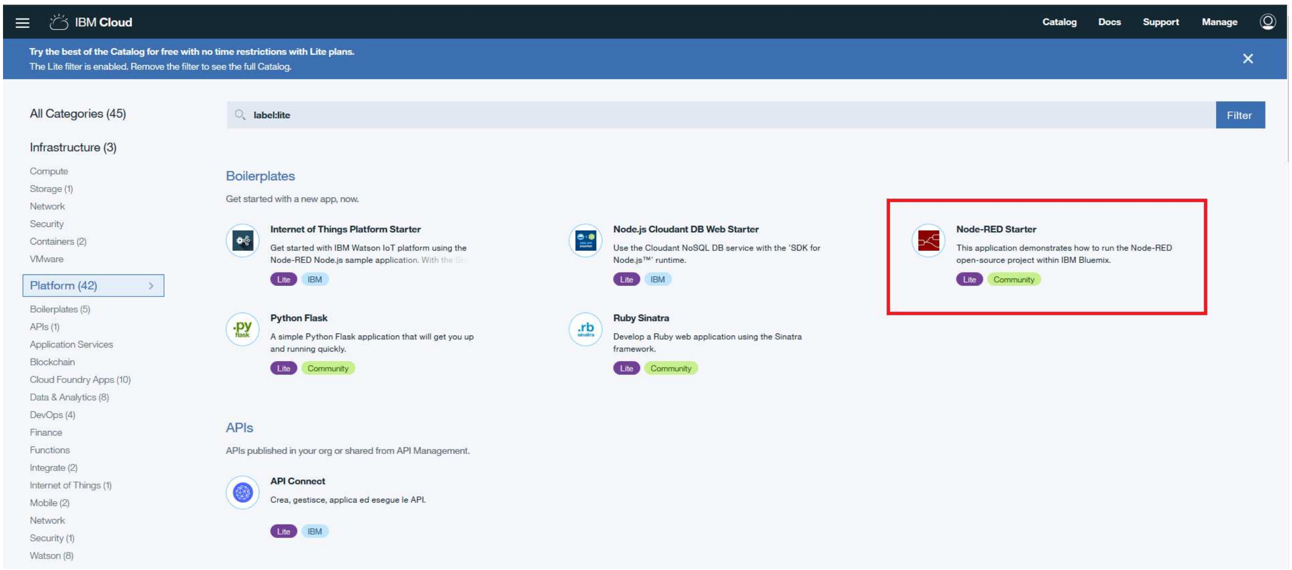
USER INTERFACE CREATION WITH Node-RED

Prerequisites

1) Have an account on Bluemix
2) Have created an Internet of Things Platform on Bluemix
3) Have created a device on Internet of Things Platform, and have its Device ID and its Authentication Token
4) Have generated an API KEY on Internet of Things Platform, and have its API Authentication Token
5) Have configured SS10680 gateway correctly

Procedure

1) Login on Bluemix (http://www.bluemix.net/)with your credentials
2) Go to 'Catalog' following this link: https://console.ng.bluemix.net/catalog/
3) From left menu, select 'Platform' and then click on 'Node-RED Starter'



Try the best of the Catalog for free with no time restrictions with Lite plans.
The Lite filter is enabled. Remove the filter to see the full Catalog.

Boilerplates

Get started with a new app, now.

Internet of Things Platform Starter
Get started with IBM Watson IoT platform using the Node-RED Node.js sample application. With the St

Node.js Cloudant DB Web Starter
Use the Cloudant NoSQL DB service with the 'SDK for Node.js™' runtime.

Node-RED Starter
This application demonstrates how to run the Node-RED open-source project within IBM Bluemix.

Python Flask
A simple Python Flask application that will get you up and running quickly.

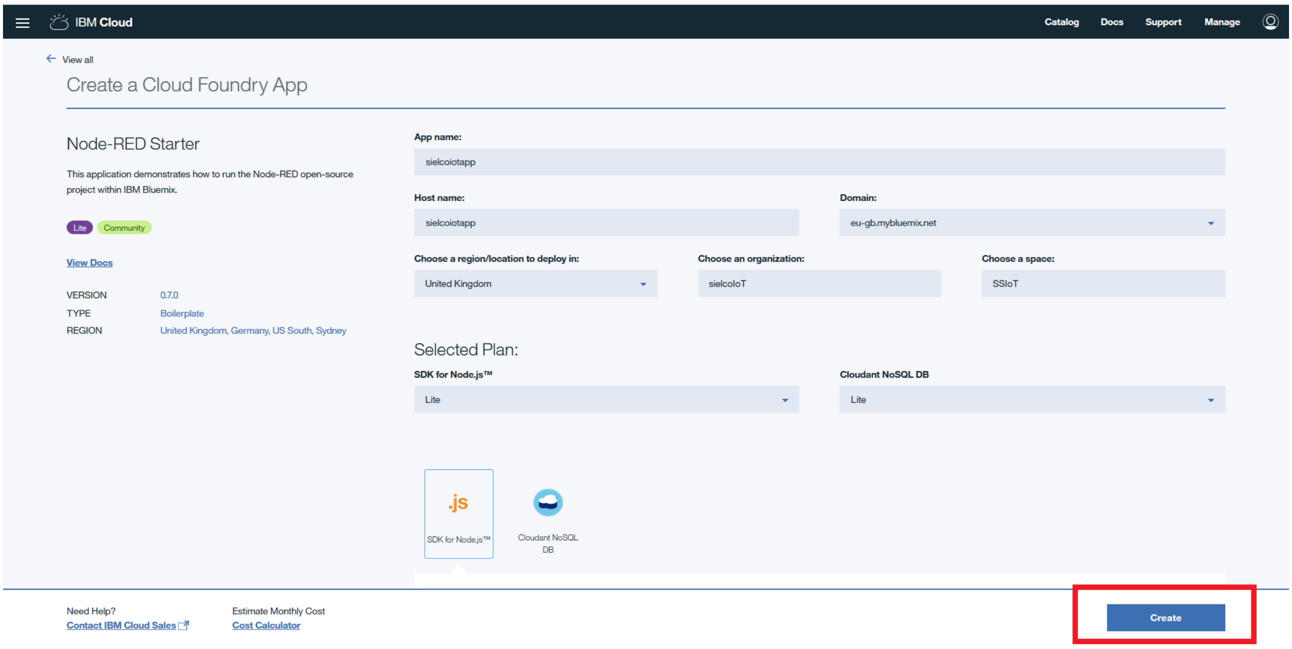
Ruby Sinatra
Develop a Ruby web application using the Sinatra framework.

APIs

APIs published in your org or shared from API Management.

API Connect
Crea, gestisce, applica ed esegue le API.

4) Fill the requested fields and then click on ‘Create’ button for create a Node-RED instance



Create a Cloud Foundry App

Node-RED Starter

This application demonstrates how to run the Node-RED open-source project within IBM Bluemix.

VERSION 0.7.0
TYPE Boilerplate
REGION United Kingdom, Germany, US South, Sydney

App name: sielcoiotapp

Host name: sielcoiotapp

Domain: eu-gb.mybluemix.net

Choose a region/location to deploy in: United Kingdom

Choose an organization: sielcoIoT

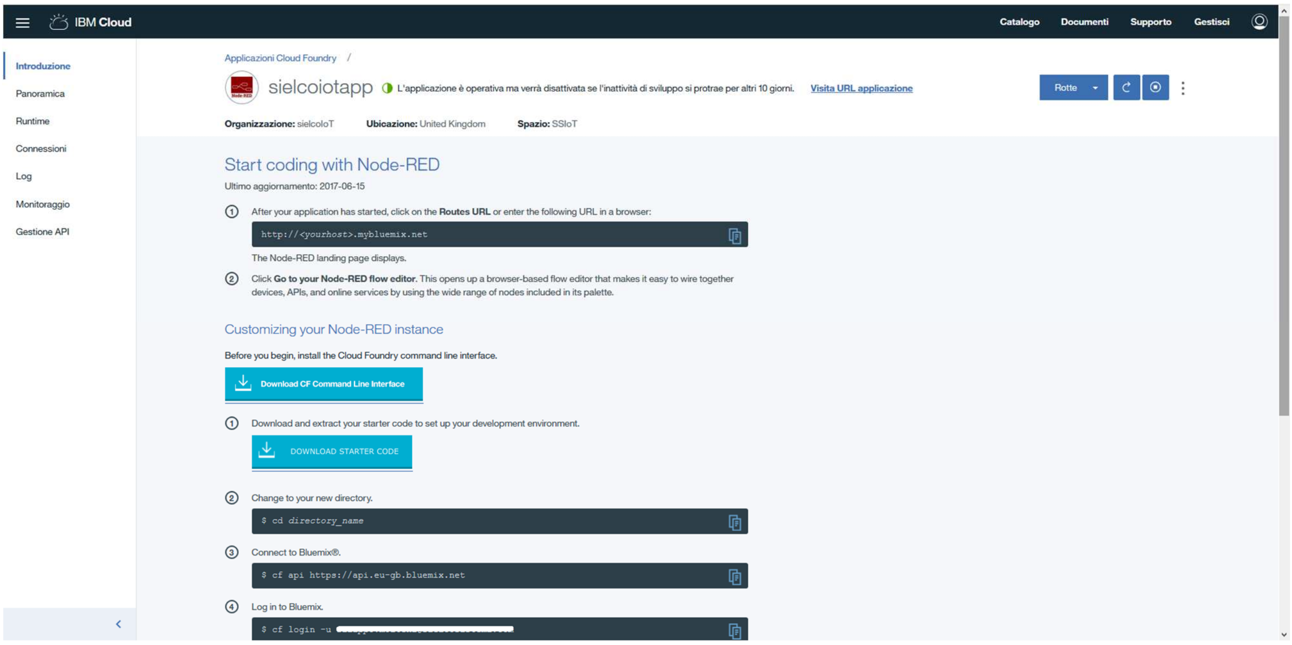
Choose a space: SSIoT

Selected Plan:

SDK for Node.js™: Lite

Cloudant NoSQL DB: Lite

So, now, Node-RED instance creation starts.
At the end of the procedure, the following page is shown to you



sielcoiotapp L'applicazione è operativa ma verrà disattivata se l'inattività di sviluppo si protrae per altri 10 giorni.

Organizzazione: sielcoIoT Ubicazione: United Kingdom Spazio: SSIoT

Start coding with Node-RED

Ultimo aggiornamento: 2017-06-15

1. After your application has started, click on the **Routes URL** or enter the following URL in a browser:

```
http://<yourhost>.mybluemix.net
```

The Node-RED landing page displays.

2. Click **Go to your Node-RED flow editor**. This opens up a browser-based flow editor that makes it easy to wire together devices, APIs, and online services by using the wide range of nodes included in its palette.

Customizing your Node-RED instance

Before you begin, install the Cloud Foundry command line interface.

1. Download and extract your starter code to set up your development environment.

2. Change to your new directory.

```
$ cd directory_name
```

3. Connect to Bluemix®.

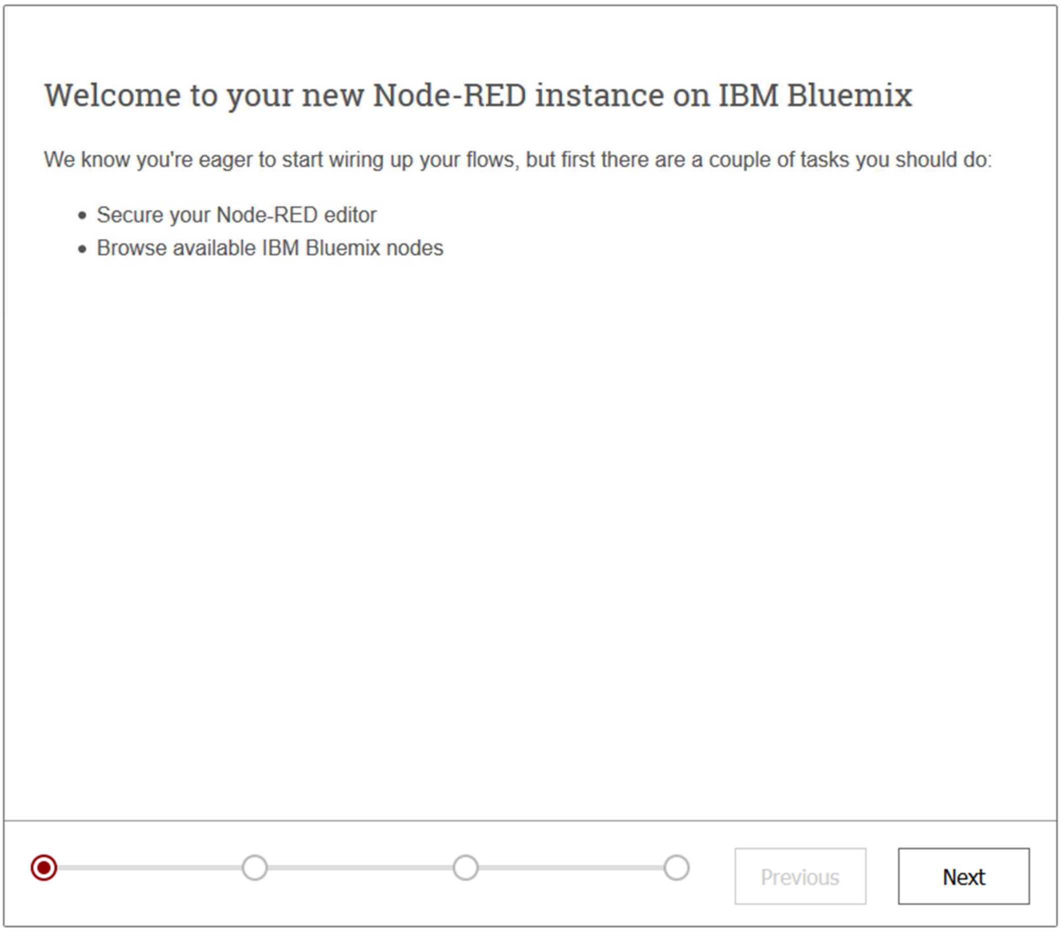
```
$ cf api https://api.eu-gb.bluemix.net
```

4. Log in to Bluemix.

```
$ cf login -u [redacted]
```

5) Click in ‘Visit application URL’

6) The first time that you execute the Node-RED instance, you must define some properties for it. On first page, click on ‘Next’ button

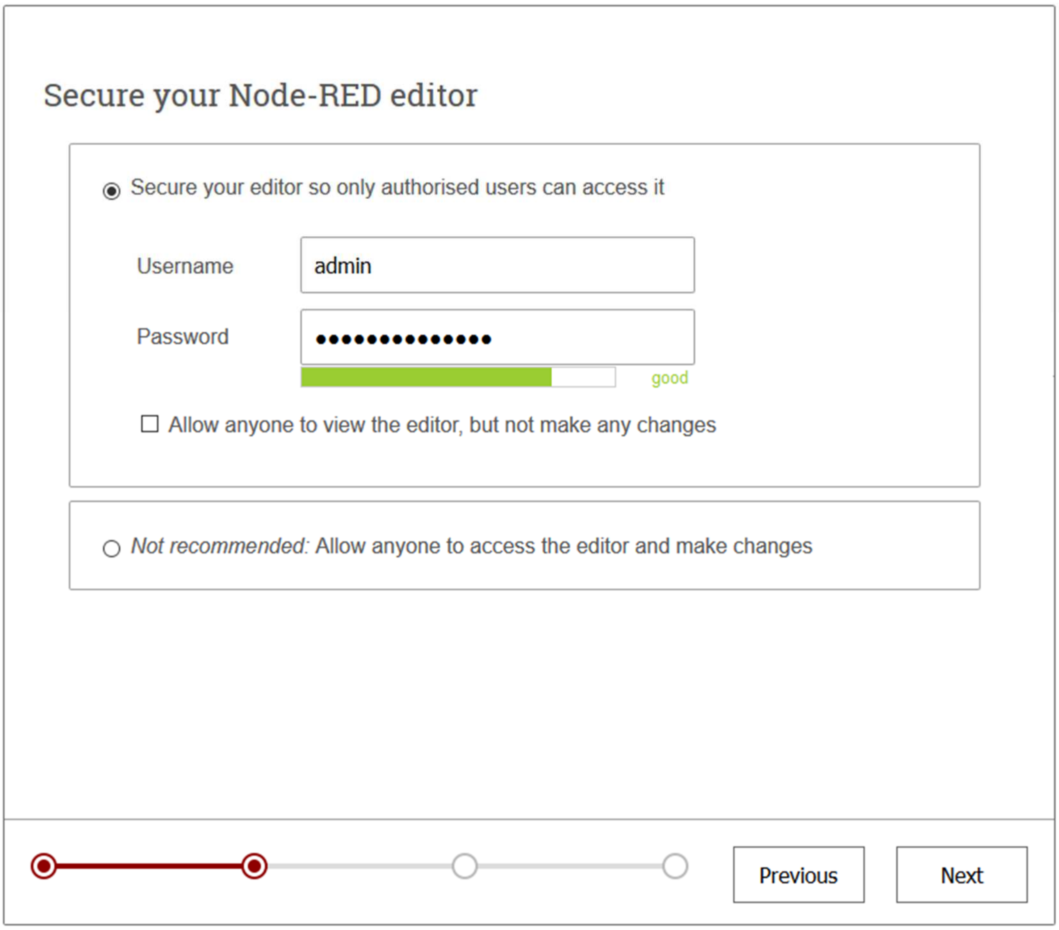


Welcome to your new Node-RED instance on IBM Bluemix

We know you're eager to start wiring up your flows, but first there are a couple of tasks you should do:

- Secure your Node-RED editor
- Browse available IBM Bluemix nodes

7) In case you want to allow access only authorized person to Node-RED instance select *'Secure your editor so only authorised users can access it'* and enter a username and a password for it.
   In case you want to allow access to any user but do not allow them to do any changes select *'Allow anyone to view the editor, but not make any changes'*.
   In case you want to allow access to any user and allow them to do changes select *'Allow anyone to access the editor and make changes'*.
   N.B. Last choice is not recommended.
   At the end, click on 'Next' button



Secure your Node-RED editor

● Secure your editor so only authorised users can access it

Username: admin

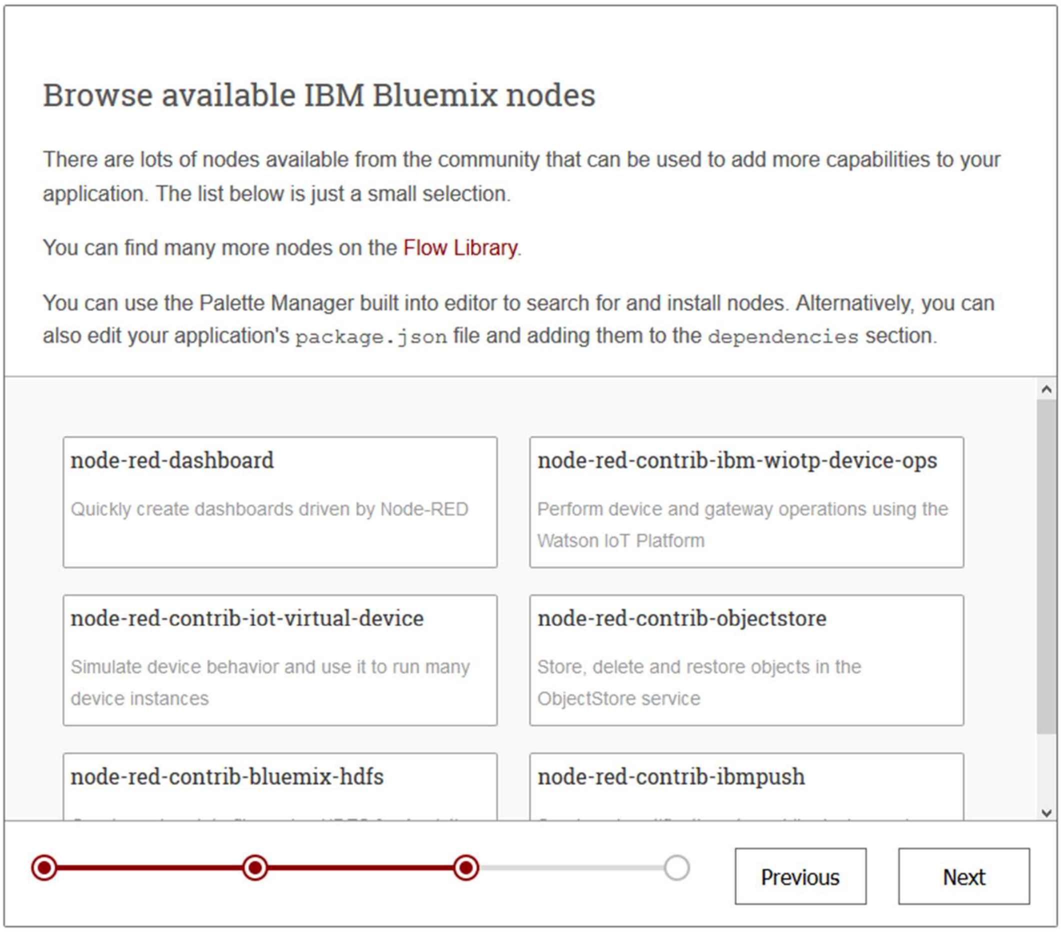
Password: ••••••••••••••

good

□ Allow anyone to view the editor, but not make any changes

○ *Not recommended:* Allow anyone to access the editor and make changes

8) Read general information and, then, click on ‘Next’ button



Browse available IBM Bluemix nodes

There are lots of nodes available from the community that can be used to add more capabilities to your application. The list below is just a small selection.

You can find many more nodes on the Flow Library.

You can use the Palette Manager built into editor to search for and install nodes. Alternatively, you can also edit your application's `package.json` file and adding them to the `dependencies` section.

node-red-dashboard

Quickly create dashboards driven by Node-RED

node-red-contrib-ibm-wiotp-device-ops

Perform device and gateway operations using the Watson IoT Platform

node-red-contrib-iot-virtual-device

Simulate device behavior and use it to run many device instances

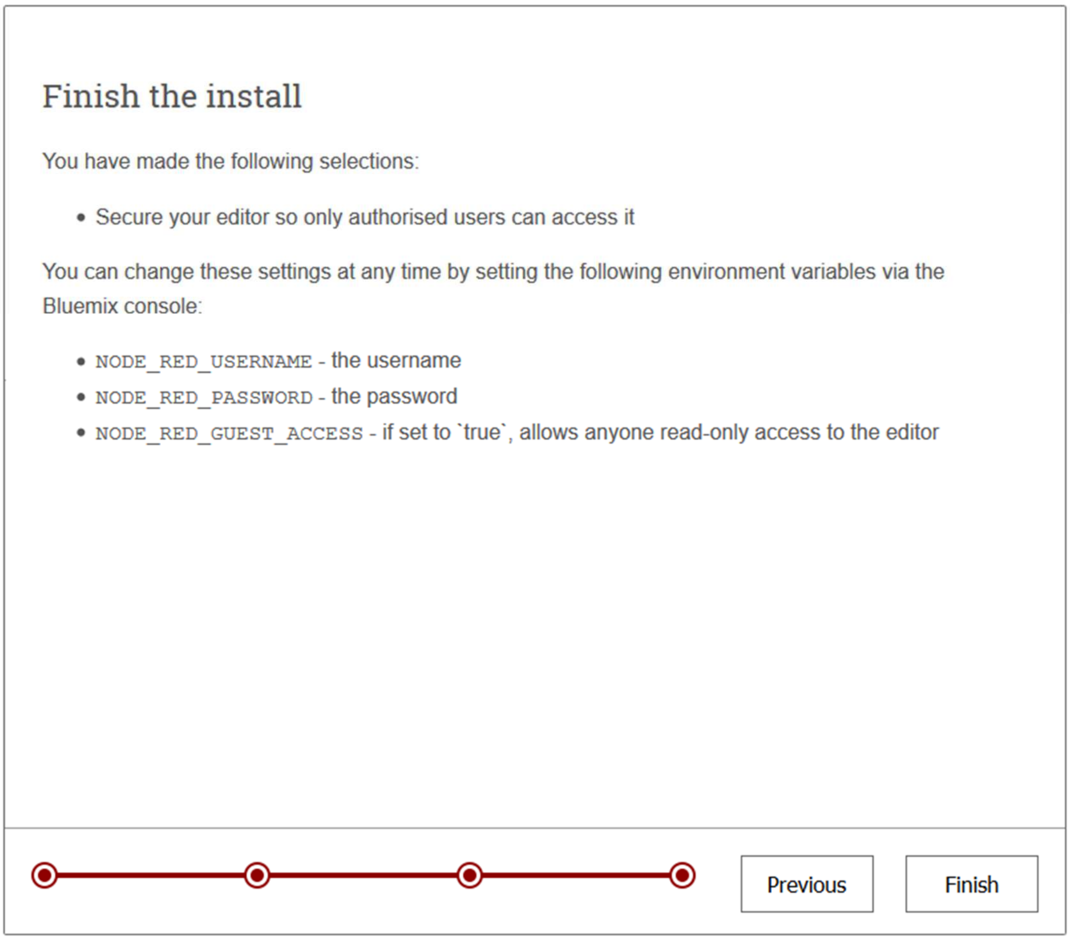
node-red-contrib-objectstore

Store, delete and restore objects in the ObjectStore service

node-red-contrib-bluemix-hdfs

node-red-contrib-ibmpush

9) Click on ‘Finish’ button to finish the install



Finish the install

You have made the following selections:

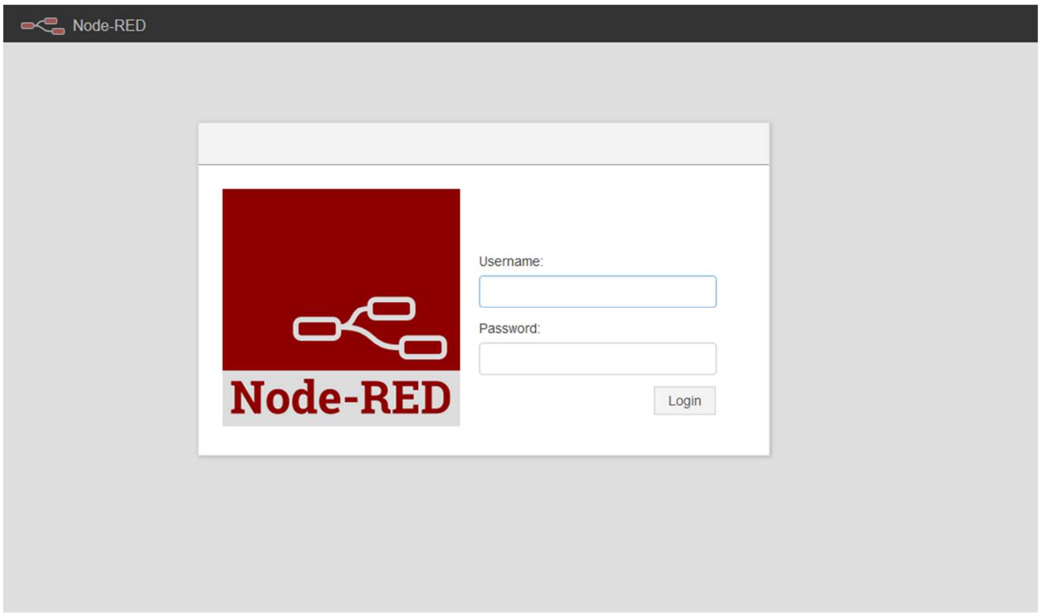
- Secure your editor so only authorised users can access it

You can change these settings at any time by setting the following environment variables via the Bluemix console:

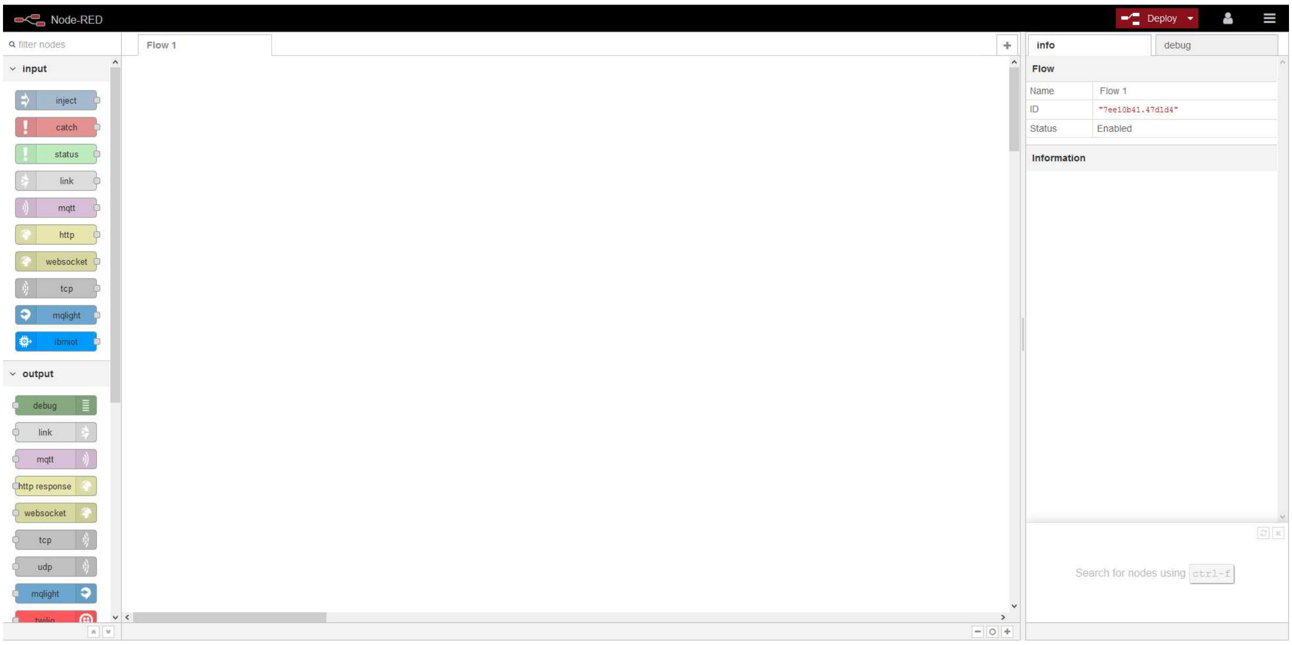
- `NODE_RED_USERNAME` - the username
- `NODE_RED_PASSWORD` - the password
- `NODE_RED_GUEST_ACCESS` - if set to `true`, allows anyone read-only access to the editor

10) When the install is finished, click on *‘Go to Node-RED flow editor’* to access the Workflow Editor

11) Enter Username and Password (if you made the first choice at point 7) and click on ‘Login’

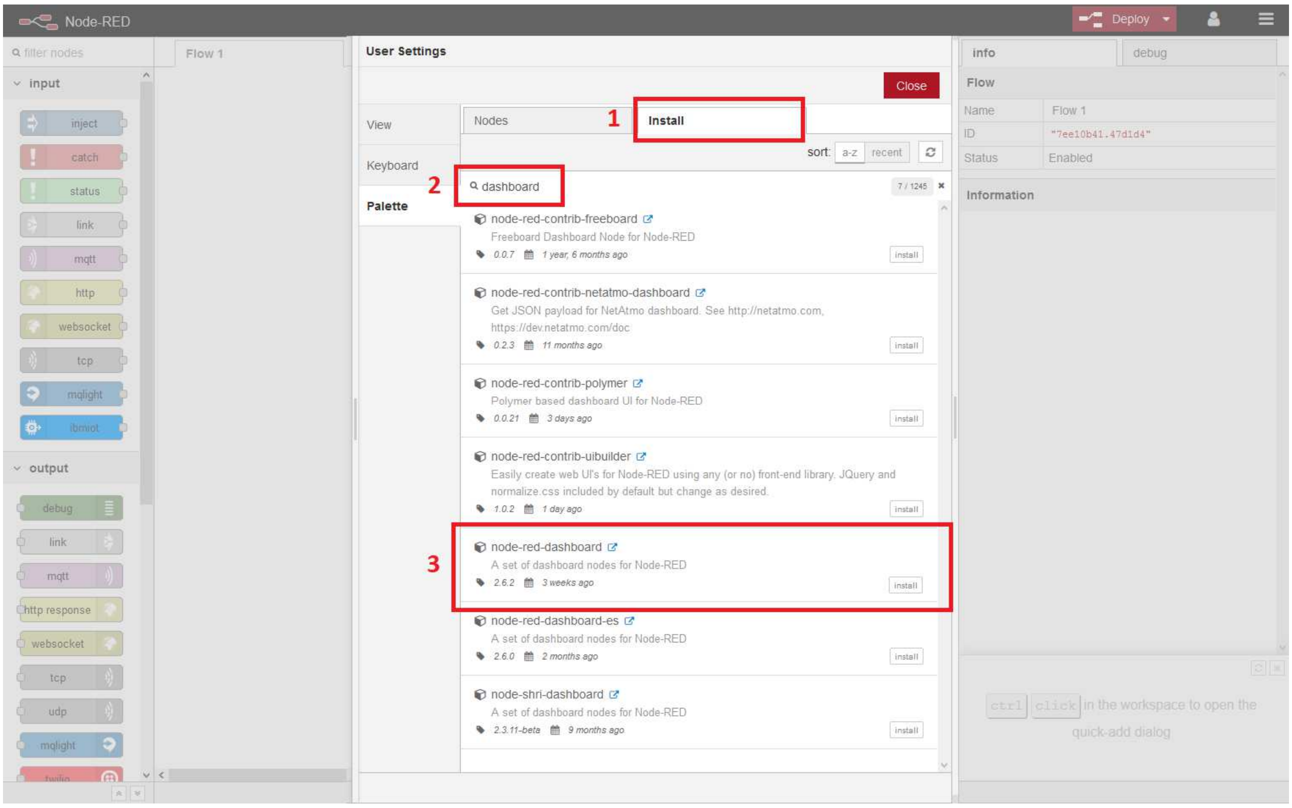


If login went well, you can see the following page



12) Install ‘**node-red-dashboard**’ packet, that we will use to build the user interface

Click on ≡ icon, and then click on ‘Manage Palette’ label. On the panel that opens click on ‘Install’ tab (1), on the search bar enter the keyword ‘dashboard’ (2), find ‘node-red-dashboard’ packet in the list and click on ‘install’ to install it.



node-red-contrib-freeboard
Freeboard Dashboard Node for Node-RED
0.0.7 1 year, 6 months ago install

node-red-contrib-netatmo-dashboard
Get JSON payload for NetAtmo dashboard. See http://netatmo.com, https://dev.netatmo.com/doc
0.2.3 11 months ago install

node-red-contrib-polymer
Polymer based dashboard UI for Node-RED
0.0.21 3 days ago install

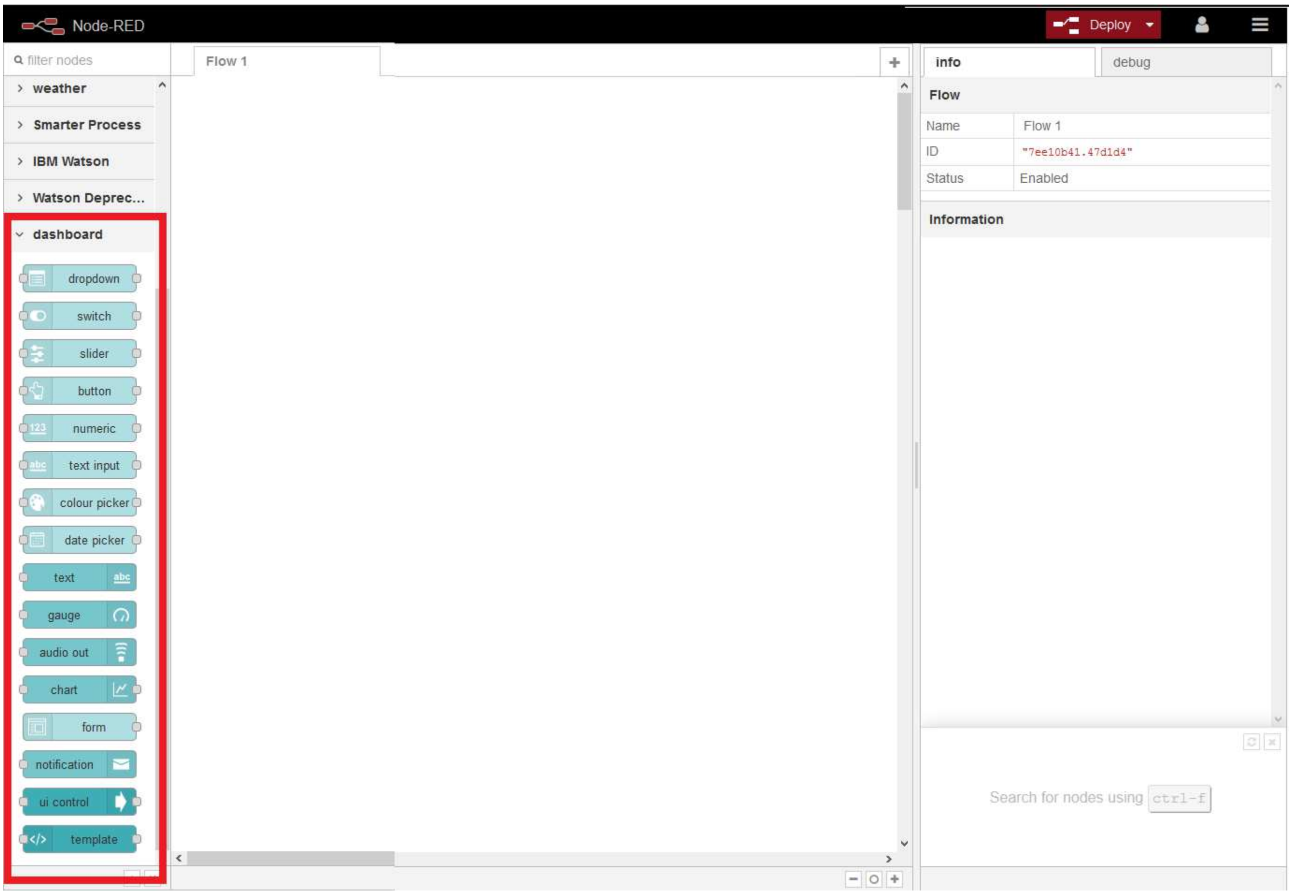
node-red-contrib-uibuilder
Easily create web UI's for Node-RED using any (or no) front-end library. JQuery and normalize.css included by default but change as desired.
1.0.2 1 day ago install

node-red-dashboard
A set of dashboard nodes for Node-RED
2.6.2 3 weeks ago install

node-red-dashboard-es
A set of dashboard nodes for Node-RED
2.6.0 2 months ago install

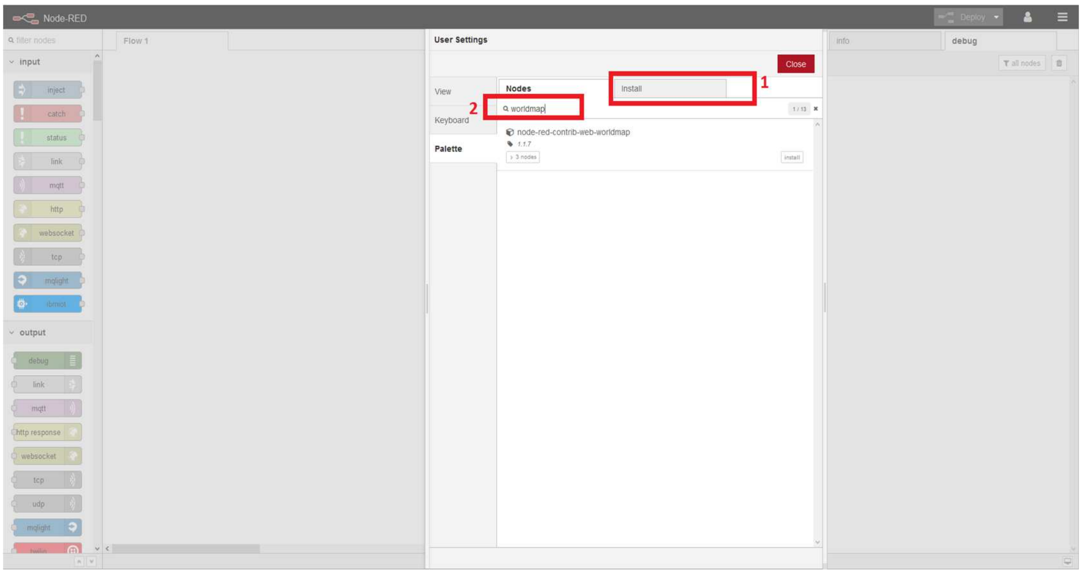
node-shri-dashboard
A set of dashboard nodes for Node-RED
2.3.11-beta 9 months ago install

If the installation of the package went well, on the left bar in the main window, you can find the ‘**dashboard**’ menu with all its nodes.

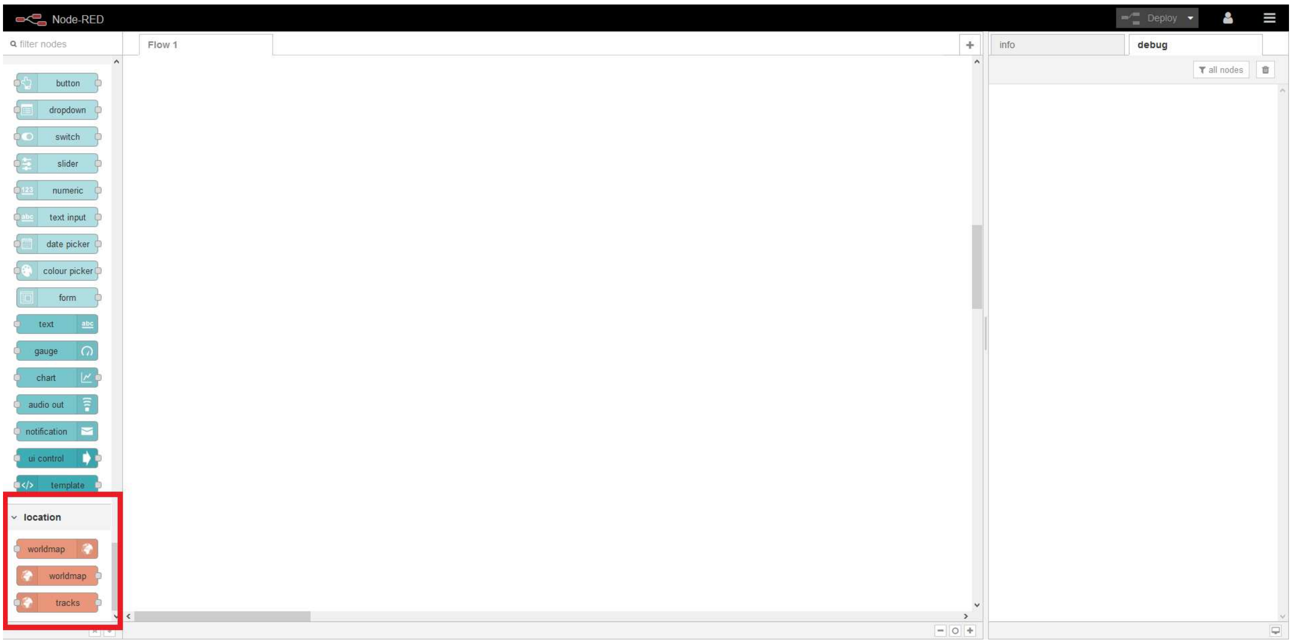


13) Install 'node-red-contrib-web-worldmap' packet, that will use for create a map in the user interface

Click on ☰ icon, and then click on 'Manage Palette' label. On the panel that opens click on 'Install' tab (1), on the search bar enter the keyword 'worldmap' (2), find 'node-red-contrib-web-worldmap' packet in the list and click on 'install' to install it.

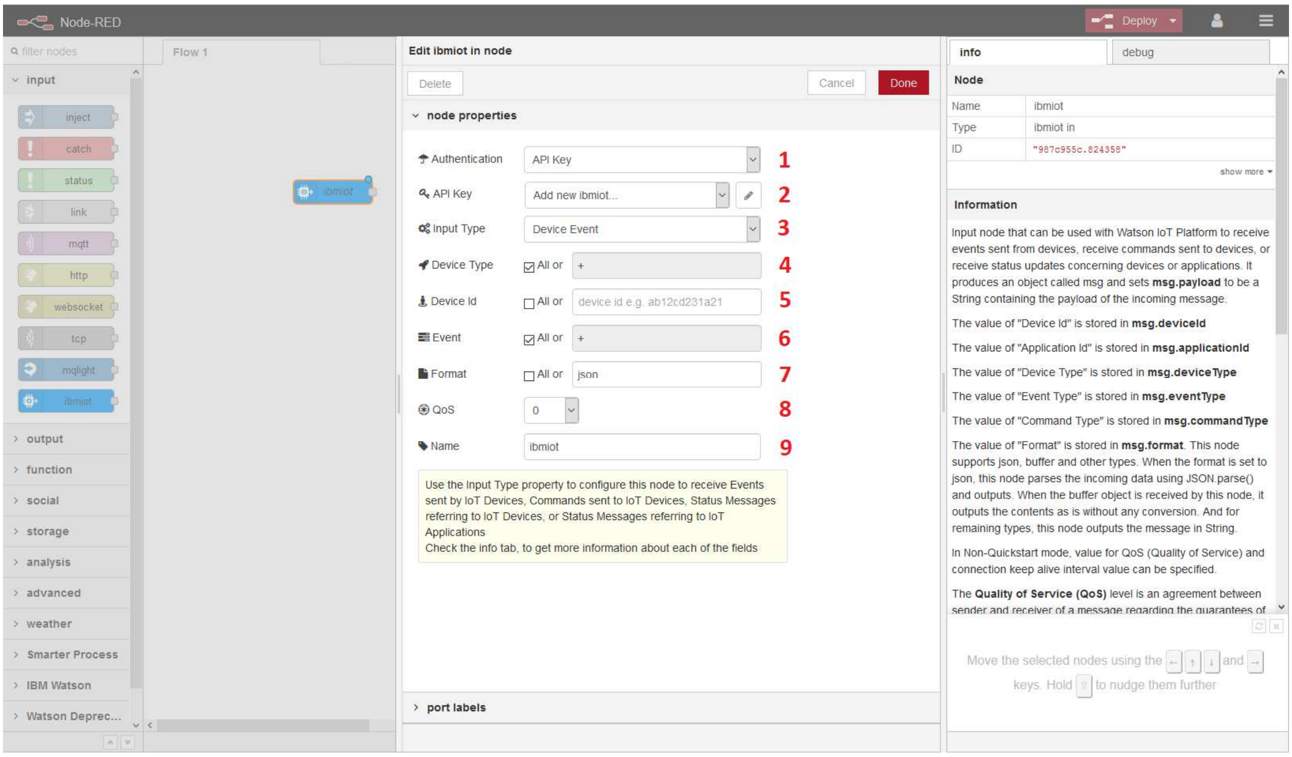


If the installation of the package went well, on the left bar in the main window, you can find the '**location**' menu with all its nodes.



Now, let’s start with the user interface building.

14) From ‘*input*’ menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into ‘Flow 1’ area. Once the node is in the working area, double click on it and the following panel will be shown



Edit ibmiot in node

node properties

Authentication	API Key	1
API Key	Add new ibmiot...	2
Input Type	Device Event	3
Device Type	☑ All or +	4
Device Id	□ All or device id e.g. ab12cd231a21	5
Event	☑ All or +	6
Format	□ All or json	7
QoS	0	8
Name	ibmiot	9

Use the Input Type property to configure this node to receive Events sent by IoT Devices, Commands sent to IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to IoT Applications
Check the info tab, to get more information about each of the fields

Node

Name	ibmiot
Type	ibmiot in
ID	"987c955c.824358"

Information

Input node that 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. It produces an object called msg and sets **msg.payload** to be a String containing the payload of the incoming message.

The value of "Device Id" is stored in **msg.deviceId**

The value of "Application Id" is stored in **msg.applicationId**

The value of "Device Type" is stored in **msg.deviceType**

The value of "Event Type" is stored in **msg.eventType**

The value of "Command Type" is stored in **msg.commandType**

The value of "Format" is stored in **msg.format**. This node supports json, buffer and other types. When the format is set to json, this node parses the incoming data using JSON.parse() and outputs. When the buffer object is received by this node, it outputs the contents as is without any conversion. And for remaining types, this node outputs the message in String.

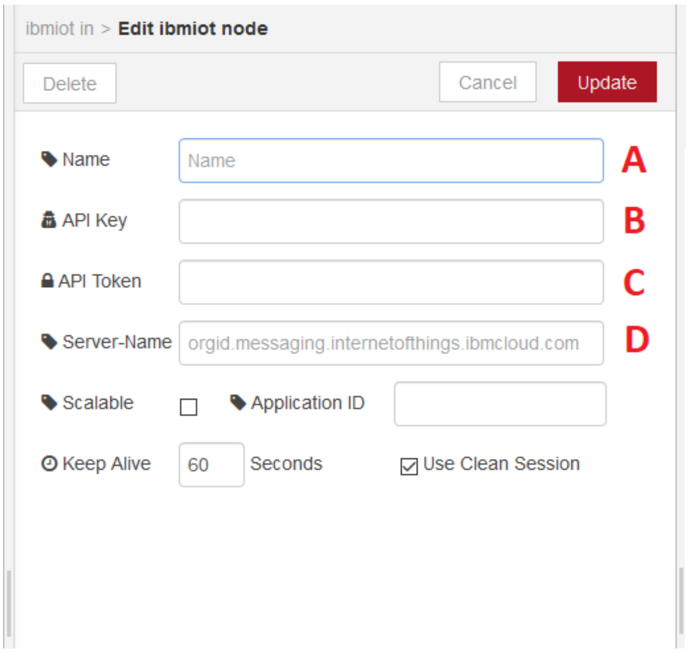
In Non-Quickstart mode, value for QoS (Quality of Service) and connection keep alive interval value can be specified.

The **Quality of Service (QoS)** level is an agreement between sender and receiver of a message regarding the guarantees of

Move the selected nodes using the ← ↑ ↓ and → keys. Hold ⇧ to nudge them further

Enter the following parameter:

1. Authentication: select ‘API KEY’
2. API KEY: click on ✎. In the panel that opens, you can configure how the Node-RED application communicates with Watson IoT Platform



ibmiot in > **Edit ibmiot node**

Delete | Cancel | Update

Name	Name
API Key	
API Token	
Server-Name	orgid.messaging.internetofthings.ibmcloud.com
Scalable	□ Application ID
Keep Alive	60 Seconds ☑ Use Clean Session

A. Name: enter a name for the connection or leave it blank
B. API Key: enter in this field the API Key of the APP created on Watson IoT
C. API Token: enter in this field the Authentication Token on the APP create on Watson IoT
D. Server-Name: enter in this field the address of the server used to communicate with Watson IoT. The address has the following syntax

   ogId.messaging.internetofthings.ibmcloud.com

   where *orgId* is the unique organization ID that was generated when you registered the service instance

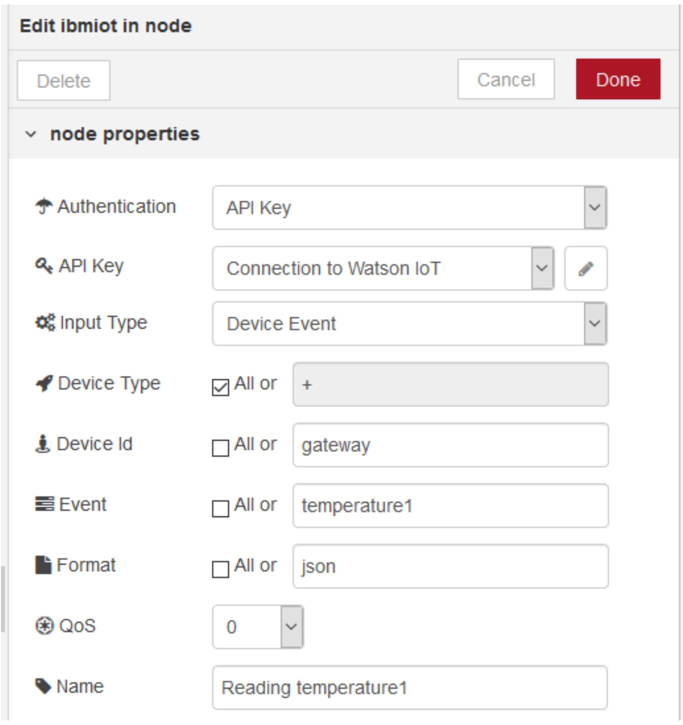
Click on Update button

3. Input Type: select ‘Device Event’
4. Device Type: if you check ‘*All*’, all the device types declared on Watson IoT are allowed. If you want allow only a device type, don’t check ‘*All*’ and enter into the text field the device type that you want to connect
5. Device Id: if you check ‘*All*’, all the devices that belong to the device types chosen are allowed. If you want allow only a device, don’t check ‘All’ and enter into the text field the device id that you want to connect
6. Event: if you check ‘*All*’, all the events of the allowed devices are considered. If you want to handle only an event, don’t check ‘*All*’ and enter into the text field the event that you want.
7. Format: if you check ‘*All*’, all the formats allowed by Watson IoT Platform (json, xml, text) can be used. If you want to use a specific format, don’t select ‘*All*’ and enter it into the text field.
8. QoS (Quality of Service):
   - 0: the message is delivered at most once, or it is not delivered at all. Its delivery across the network is not acknowledged.

- 1: the message is always delivered at least once. If the sender does not receive an acknowledgment, the message is sent again with the DUP flag set until an acknowledgment is received. As a result, receiver can be sent the same message multiple times, and might process it multiple times.
- 2: the message is always delivered exactly once.

9. Name: enter the node's name

At the end of node's configuration, the result is similar to the following



Authentication: API Key

API Key: Connection to Watson IoT

Input Type: Device Event

Device Type: ☑ All or +

Device Id: □ All or gateway

Event: □ All or temperature1

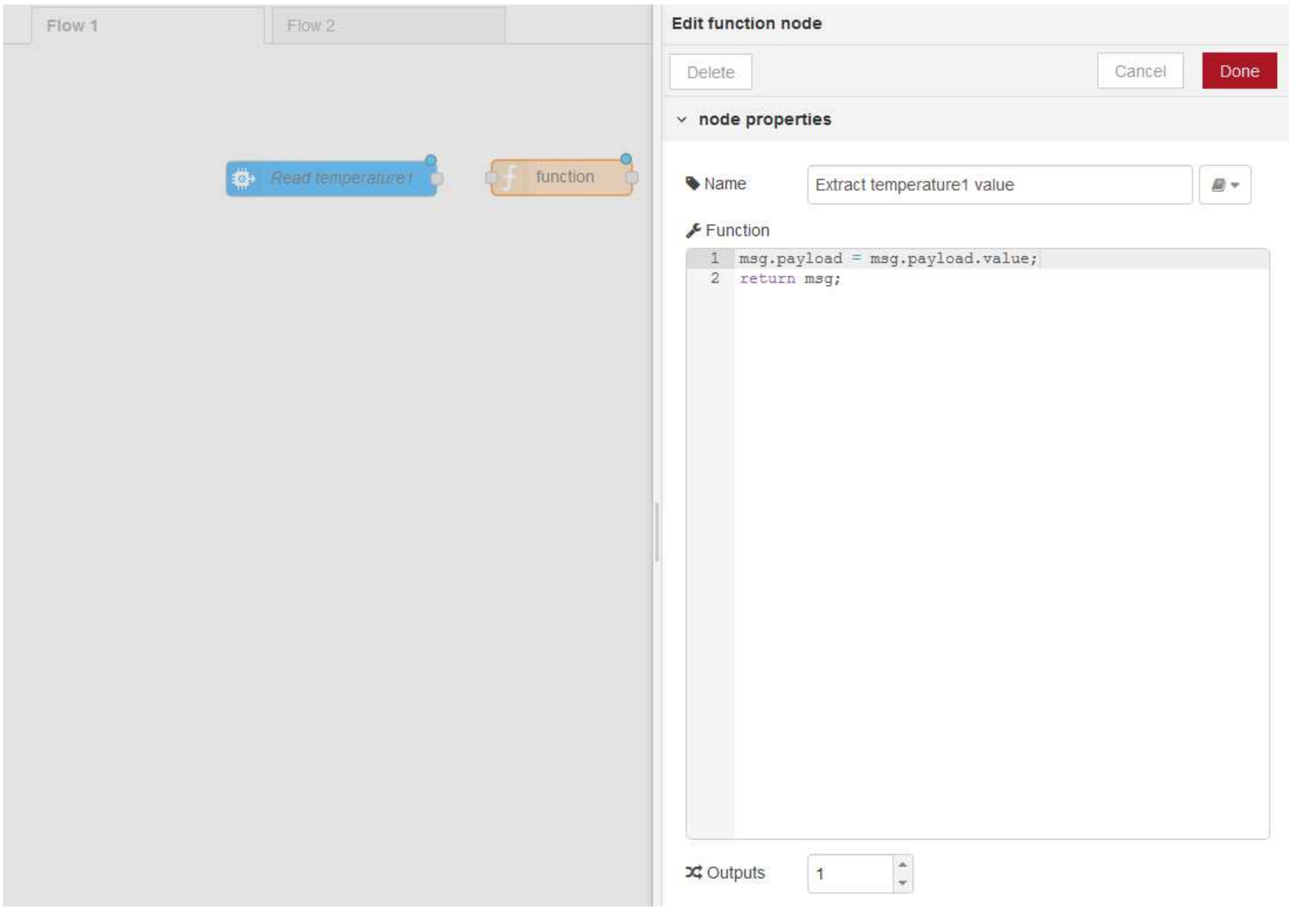
Format: □ All or json

QoS: 0

Name: Reading temperature1

To continue, click on Done button

15) From '*function*' menu in the left bar, select ***function*** node function and drag & drop it in the working area. Double click on it and configure it as below



Name: Extract temperature1 value

Function

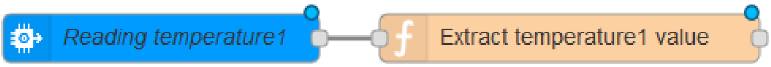
```
msg.payload = msg.payload.value;
return msg;
```

Outputs: 1

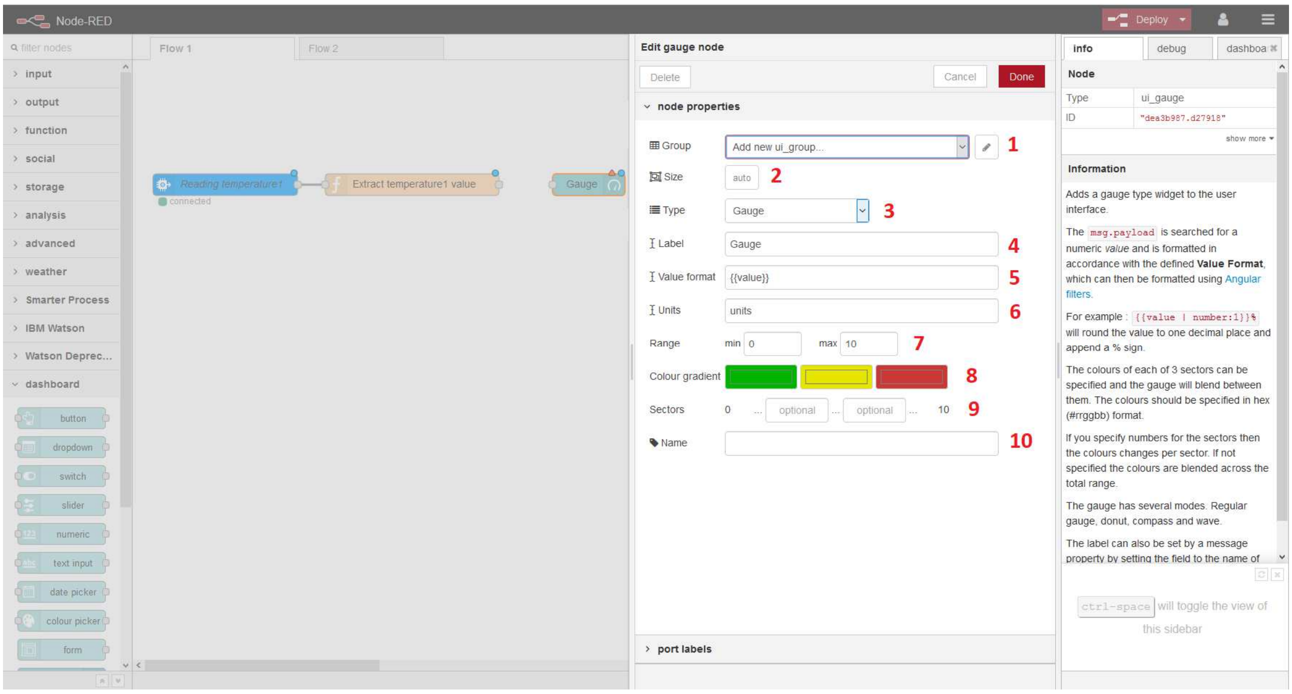
To continue, click on Done button.

16) Link the two nodes

Put the mouse over the grey square of the 'Read temperature1' node (the square becomes orange) and move the mouse over the grey square of 'Extract temperature 1 value' node. The result of the operation is shown below



17) From '*dashboard*' menu in the left bar, select ***gauge*** node gauge and drag & drop it into the working area. Double click on it



Edit gauge node

Delete Cancel Done

node properties

Group: Add new ui_group...

Size: auto

Type: Gauge

Label: Gauge

Value format: {{value}}

Units: units

Range: min 0 max 10

Colour gradient: 🟩 🟨 🟥

Sectors: 0 ... optional ... optional ... 10

Name:

port labels

Node

Type	ui_gauge
ID	"dea3b987.d27918"

show more

Information

Adds a gauge type widget to the user interface.

The `msg.payload` is searched for a numeric *value* and is formatted in accordance with the defined **Value Format**, which can then be formatted using Angular filters.

For example : `{{value | number:1}}%` will round the value to one decimal place and append a % sign.

The colours of each of 3 sectors can be specified and the gauge will blend between them. The colours should be specified in hex (#rrggbb) format.

If you specify numbers for the sectors then the colours changes per sector. If not specified the colours are blended across the total range.

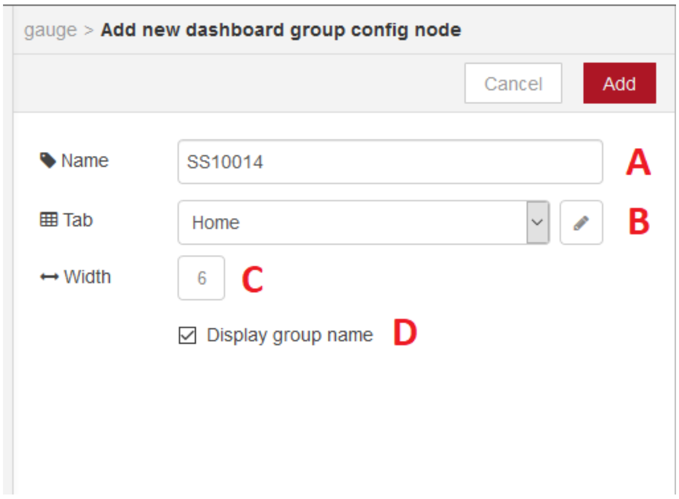
The gauge has several modes. Regular gauge, donut, compass and wave.

The label can also be set by a message property by setting the field to the name of

`ctrl-space` will toggle the view of this sidebar

Edit the node as below:

1. Group: select 'Add new ui_group' and click on ✎. In the panel that opens we can define the 'group' and the 'page' to insert the gauge



gauge > Add new dashboard group config node

Cancel Add

Name: SS10014

Tab: Home

Width: 6

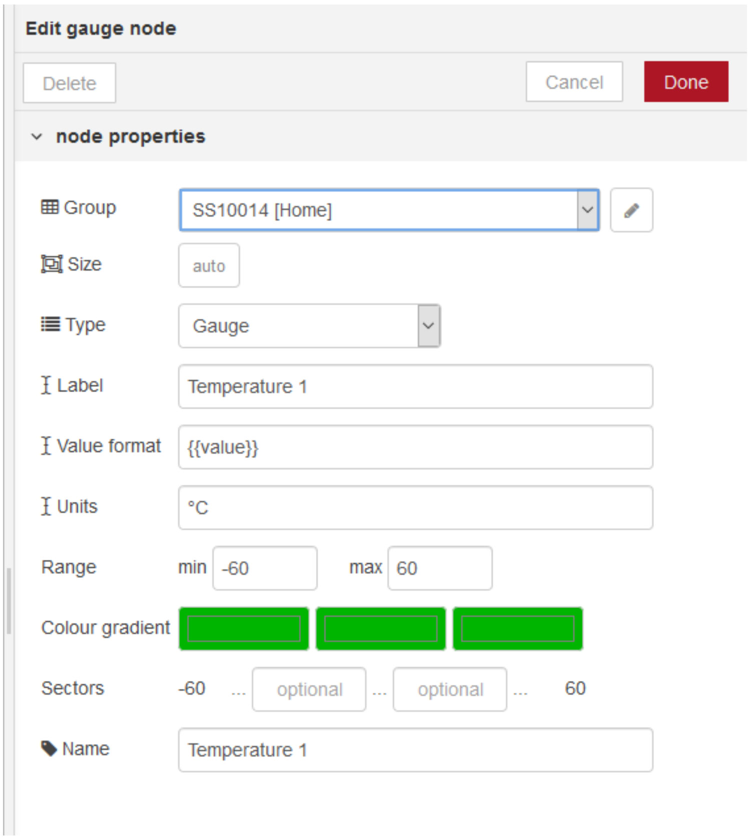
☑ Display group name

   A. Name: enter the name of the group. For this example, insert the name of the device '*SS10014*'
   B. Tab: select '*Home*'. If you want to add a new 'page', select 'Add new ui_tab' and then click on ✎.
   C. Width: enter the group's width
   D. Display group name: check the checkbox if you want to show the name of the group in the user interface

   At the end, click on Add button.
2. Size: enter the dimensions of the object in the UI. For this example, leave '*auto*'
3. Type: select '*Gauge*'
4. Label: enter '*Temperature 1*'
5. Value format: indicates the format of the value. Leave *{{value}}*.
6. Units: indicates the unit of measure. Enter *°C.*

7. Range: enter the min. possible value and the max. possible value. Enter *-60* as min. value and *60* as max. value
8. Colour gradient: for this example, create only one green sector
9. Sectors: leave the intermediate values blank
10. Name: enter the name of the node. In this case ‘*Temperature1*’

At the end, the result is as below



Group: SS10014 [Home]

Size: auto

Type: Gauge

Label: Temperature 1

Value format: {{value}}

Units: °C

Range: min -60 max 60

Colour gradient: 🟩 🟩 🟩

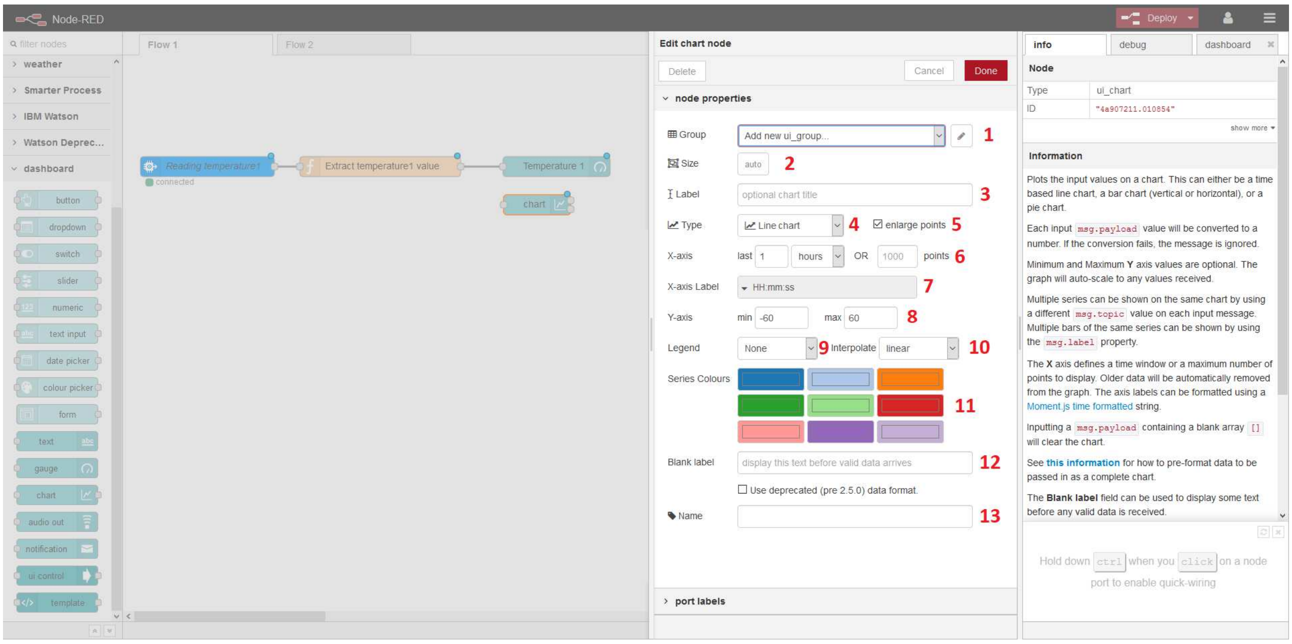
Sectors: -60 ... optional ... optional ... 60

Name: Temperature 1

To continue, click on Done button.

18) Link the ‘Extract temperature1 value’ node and the ‘Temperature1’, as shown at point 16.

19) From ‘*dashboard*’ menu in the left bar, select ***char***t node and drag & drop it in the working area. Double click on it



Edit chart node

Delete Cancel Done

node properties

Group Add new ui_group...

Size auto

Label optional chart title

Type Line chart ☑ enlarge points

X-axis last 1 hours OR 1000 points

X-axis Label HH:mm:ss

Y-axis min -60 max 60

Legend None Interpolate linear

Series Colours 🟦 🟦 🟧 🟩 🟩 🟥 🟥 🟪 🟪

Blank label display this text before valid data arrives

☐ Use deprecated (pre 2.5.0) data format.

Name

port labels

info debug dashboard

Node

Type	ui_chart
ID	"4a907211.010854"

show more

Information

Plots the input values on a chart. This can either be a time based line chart, a bar chart (vertical or horizontal), or a pie chart.

Each input `msg.payload` value will be converted to a number. If the conversion fails, the message is ignored.

Minimum and Maximum **Y** axis values are optional. The graph will auto-scale to any values received.

Multiple series can be shown on the same chart by using a different `msg.topic` value on each input message. Multiple bars of the same series can be shown by using the `msg.label` property.

The **X** axis defines a time window or a maximum number of points to display. Older data will be automatically removed from the graph. The axis labels can be formatted using a Moment.js time formatted string.

Inputting a `msg.payload` containing a blank array `[]` will clear the chart.

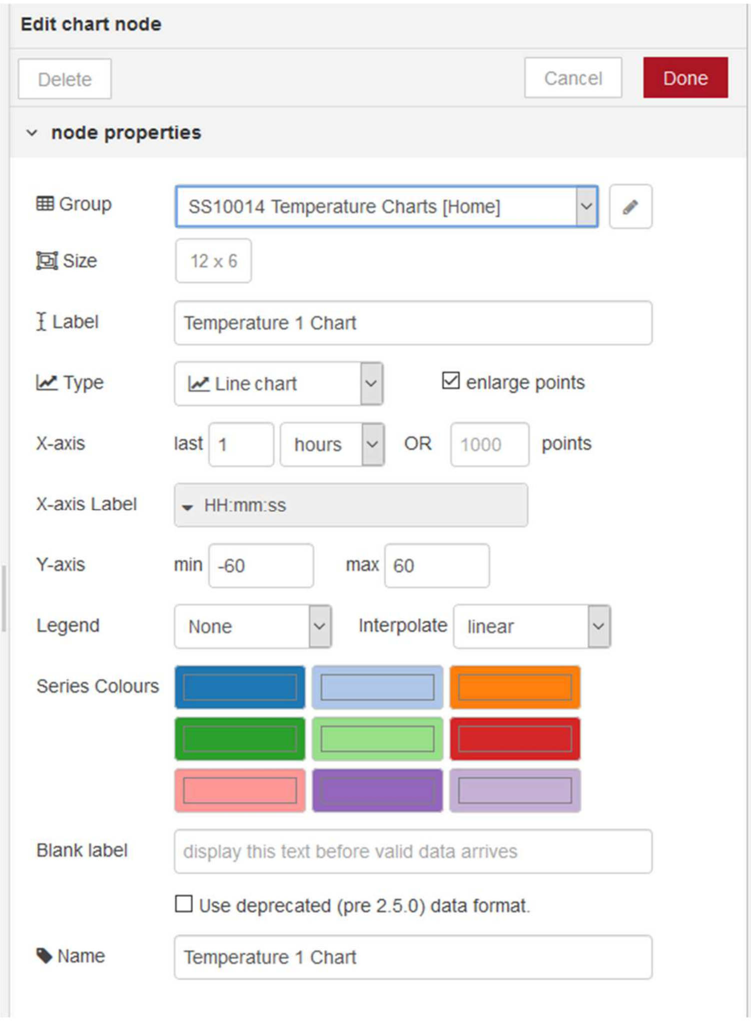
See **this information** for how to pre-format data to be passed in as a complete chart.

The **Blank label** field can be used to display some text before any valid data is received.

Hold down `ctrl` when you `click` on a node port to enable quick-wiring

1. Group: select ‘Add new ui_group’ and click on . On the panel that opens, follow the instruction explained at point 17.1, and insert a new group named ‘*SS10014 Temperature Charts*’. Leave the ‘Tab’ field on ‘*Home*’, set ‘*Width*’ field to *12*, and check the checkbox ‘*Display group name*’
2. Size: set ‘*12x6*’
3. Label: enter ‘*Temperature 1 Chart*’
4. Type: select ‘*Line Chart*’
5. Enlarge points: check the checkbox, for let value points visible
6. X-axis: select ‘*last 1 hour*’
7. X-axis Label: select ‘*HH:mm:ss*’
8. Y-axis: insert *-60* as min. value and *60* as max. value
9. Legend: select ‘*None*’
10. Interpolate: select ‘*linear*’
11. Series Colours: select the color for chart lines
12. Blank label: leave the field blank
13. Name: enter the name of the node. In this case ‘*Temperature Chart 1*’

At the end, the result is as below



Edit chart node

Delete | Cancel | Done

node properties

Group: SS10014 Temperature Charts [Home]

Size: 12 x 6

Label: Temperature 1 Chart

Type: Line chart ☑ enlarge points

X-axis: last 1 hours OR 1000 points

X-axis Label: HH:mm:ss

Y-axis: min -60 max 60

Legend: None Interpolate: linear

Series Colours

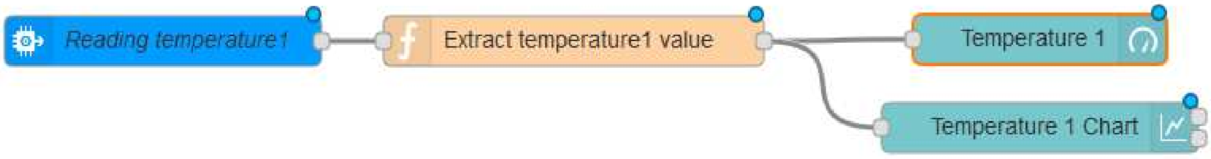
Blank label: display this text before valid data arrives

☐ Use deprecated (pre 2.5.0) data format.

Name: Temperature 1 Chart

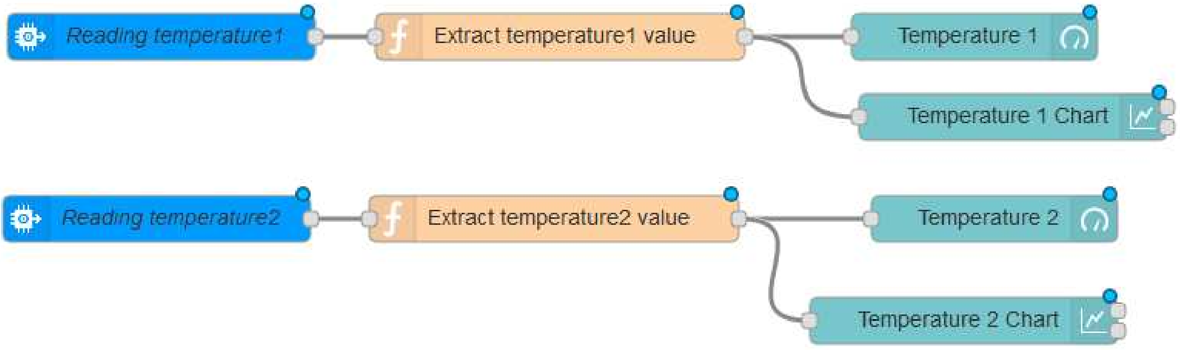
To continue, click on Done button.

20) Link the ‘*Extract temperature1 value*’ node and the ‘*Temperature 1 Chart*’ as shown at point 16
The result is the following



21) Repeat the points from 14 to 20 for the *temperature2* of the SS10014 device.
The procedure is the same followed for the *temperature1*, pay attention at point 14.2, where you have to select ‘*Connection to Watson IoT*’, and at point 14.6, where you have to enter ‘*temperature2*’ as event.

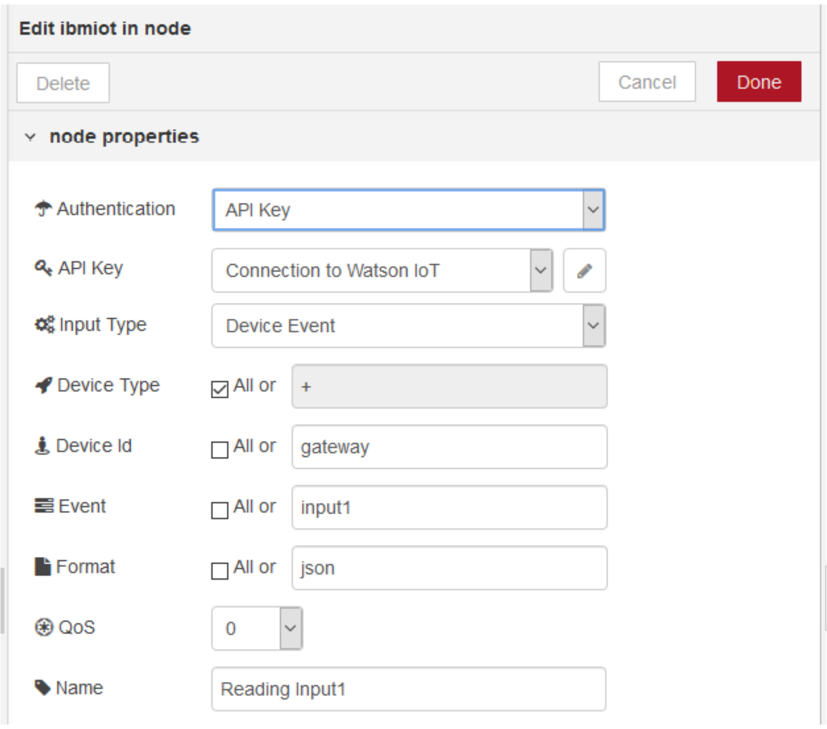
At the end of the procedure, the result must be similar at the one below



22) Save the flow by clicking on Deploy button in the upper right corner of the window

Now we can move on to define the user interface relative to the SS10130 device, defined in the SS10680 gateway interface.

23) From '*input*' menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.
Once the node is in the working area, double click on it and a panel as the one described at point 14 is shown. Select for the field 'API Key' the option '*Connect to Watson IoT*'. For the field 'Event', insert the string '*input1*'. At the end of node's configuration, the result is as the following one



Edit ibmiot in node

Delete | Cancel | Done

node properties

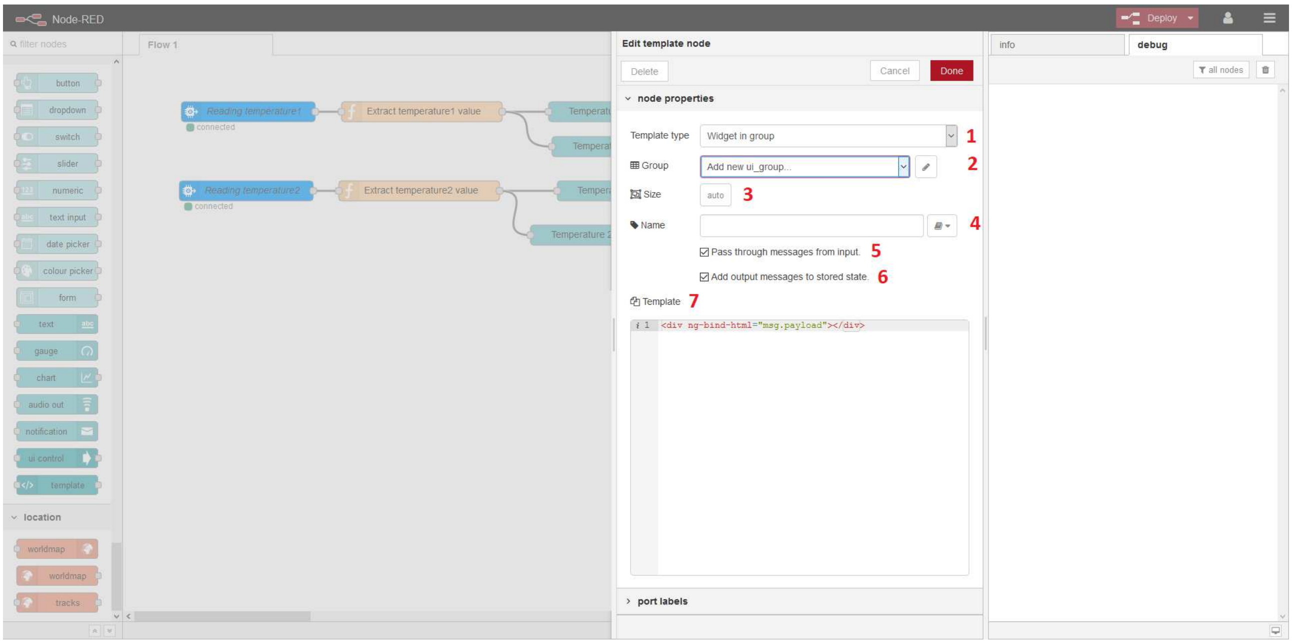
Authentication	API Key
API Key	Connection to Watson IoT
Input Type	Device Event
Device Type	☑ All or +
Device Id	☐ All or gateway
Event	☐ All or input1
Format	☐ All or json
QoS	0
Name	Reading Input1

To continue, click on Done button.

24) From '*function*' menu in the left bar, select ***function*** node function, drag & drop it into the working area.
Double click on it and edit it as shown at point 15.

25) Link the previous two nodes inserted, as shown at point 16

26) From '*dashboard*' menu in the left bar, select ***template*** node template e drag & drop it into the working area. Double click on it and edit it as shown below



Edit template node

node properties

Template type: Widget in group

Group: Add new ui_group...

Size: auto

Name

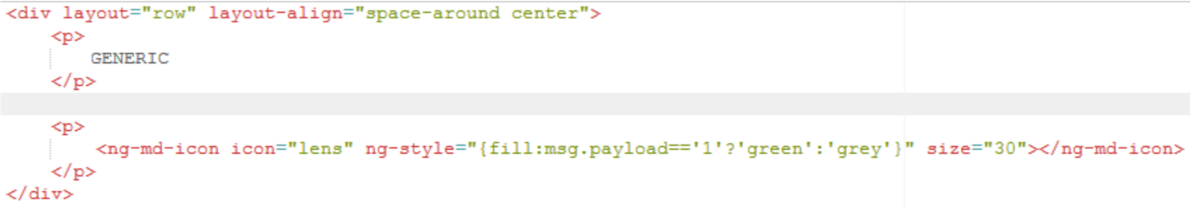
☑ Pass through messages from input.

☑ Add output messages to stored state.

Template

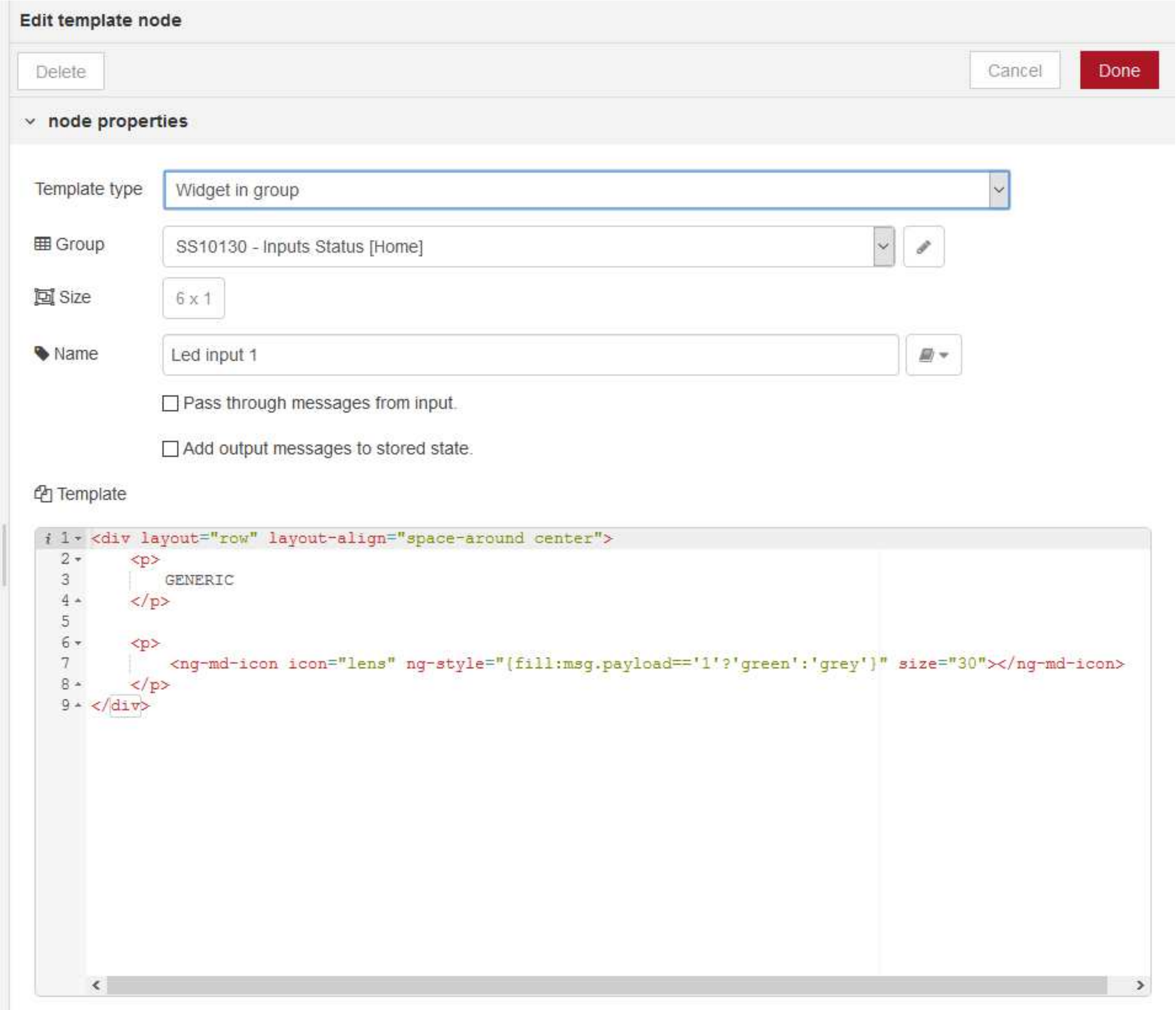
```
<div ng-bind-html="msg.payload"></div>
```

1. Template type: select ‘*Widget in group’*
2. Group: click on ✎ and insert a new group ‘*SS10130 – Input Status*’ as shown at point 17.1
3. Size: set ‘*6x1*’
4. Name: enter ‘*Led input 1*’
5. Don’t check ‘*Pass through messages from input*’ checkbox
6. Don’t check ‘*Add output messages to stored state.*’ checkbox
7. Template: enter the following code



```
<div layout="row" layout-align="space-around center">
    <p>
        GENERIC
    </p>

    <p>
        <ng-md-icon icon="lens" ng-style="{fill:msg.payload=='1'?'green':'grey'}" size="30"></ng-md-icon>
    </p>
</div>
```

At the end, the result is similar to the one shown below



Edit template node

Delete | Cancel | Done

⌄ **node properties**

Template type: Widget in group

Group: SS10130 - Inputs Status [Home]

Size: 6 x 1

Name: Led input 1

☐ Pass through messages from input.

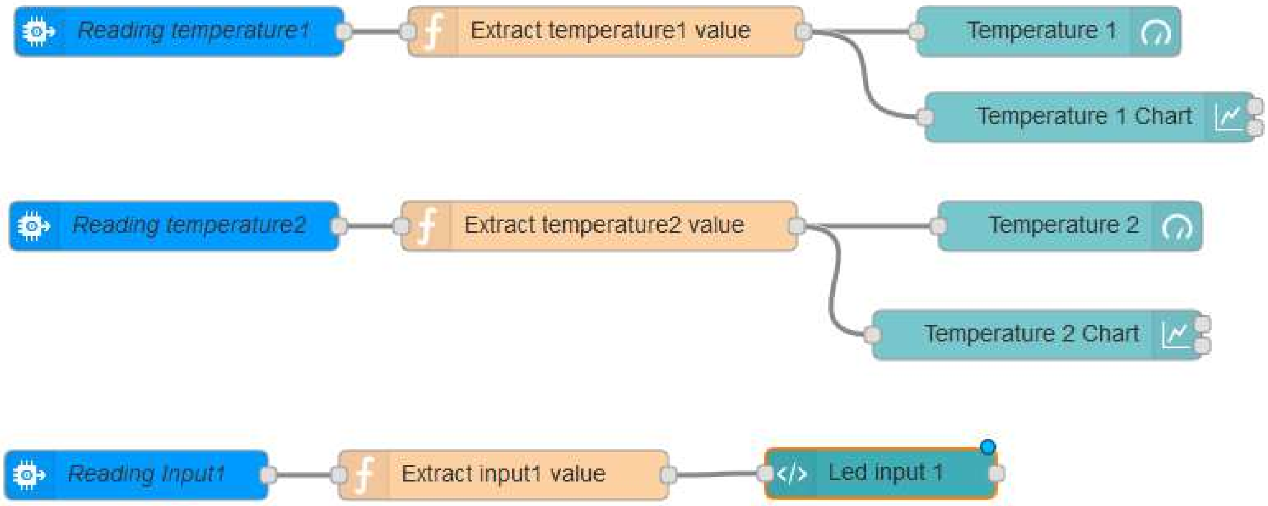
☐ Add output messages to stored state.

Template

```
<div layout="row" layout-align="space-around center">
    <p>
        GENERIC
    </p>

    <p>
        <ng-md-icon icon="lens" ng-style="{fill:msg.payload=='1'?'green':'grey'}" size="30"></ng-md-icon>
    </p>
</div>
```

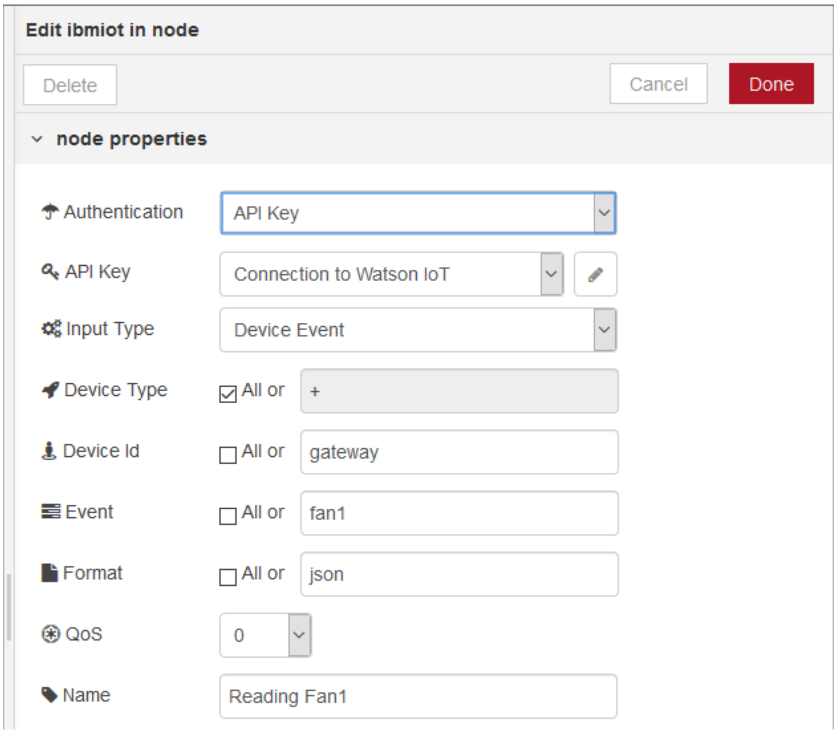
To continue, click on Done button.

27) Link the node added at the previous point at '*Extract input1 value*' node.



28) From '*input*' menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.

Once the node is in the working area, double click on it and a panel as the one described at point 14 is shown. Select for the field ‘API Key’ the option ‘*Connect to Watson IoT*’. For the field ‘Event’, insert the string ‘*fan1*’. At the end of node’s configuration, the result is as the following one



Edit ibmiot in node

Delete | Cancel | Done

node properties

Authentication: API Key

API Key: Connection to Watson IoT

Input Type: Device Event

Device Type: ☑ All or +

Device Id: □ All or gateway

Event: □ All or fan1

Format: □ All or json

QoS: 0

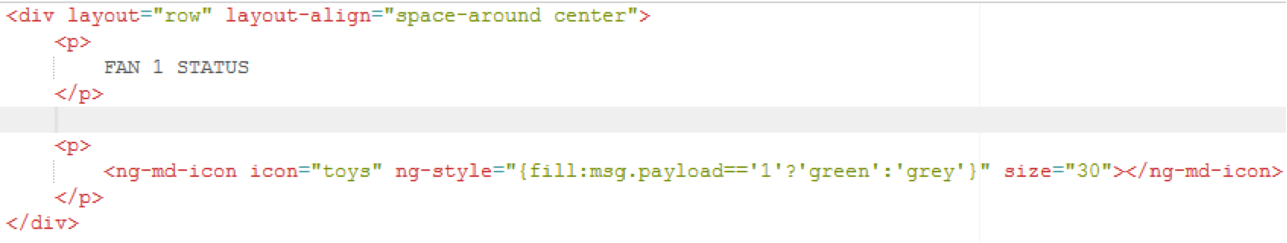
Name: Reading Fan1

To continue, click on Done button.

29) From ‘*function*’ menu in the left bar, select ***function*** node function, drag & drop it into the working area.
Double click on it and edit it as shown at point 15.

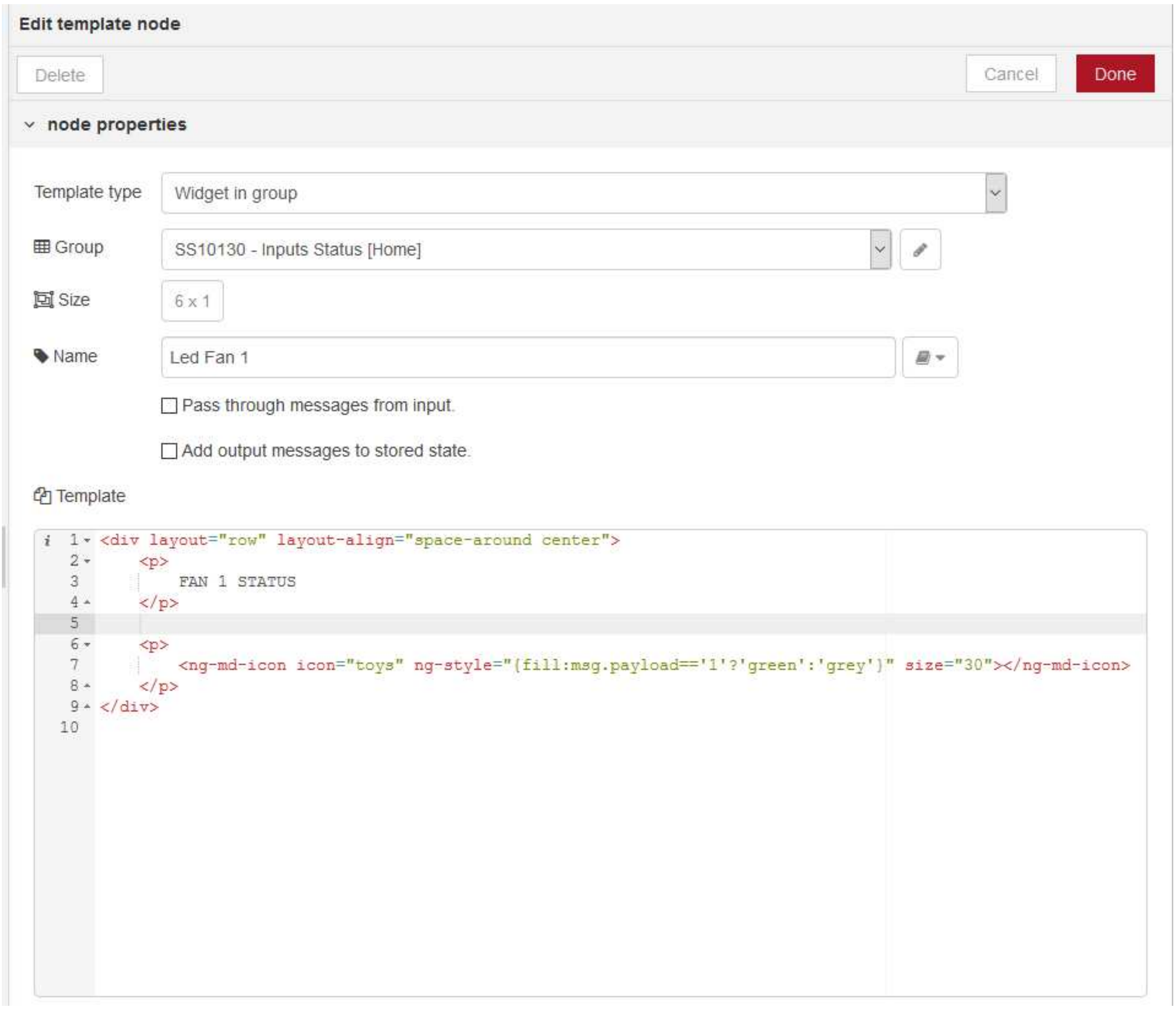
30) From ‘*dashboard*’ menu in the left bar, select ***template*** node template e drag & drop it into the working area. Double click on it and edit it as shown below

- Template type: select ‘*Widget in group*’
- Group: select ‘*SS10130 – Inputs Status*’
- Size: set ‘*6x1*’
- Name: enter ‘*Led fan 1*’
- Don’t check ‘*Pass through messages from input*’ checkbox
- Don’t check ‘*Add output messages to stored state.*’ checkbox
- Template: enter the following code



```
<div layout="row" layout-align="space-around center">
    <p>
        FAN 1 STATUS
    </p>

    <p>
        <ng-md-icon icon="toys" ng-style="{fill:msg.payload=='1'?'green':'grey'}" size="30"></ng-md-icon>
    </p>
</div>
```

At the end of node’s configuration, the result is the following one



Edit template node

Delete | Cancel | Done

node properties

Template type: Widget in group

Group: SS10130 - Inputs Status [Home]

Size: 6 x 1

Name: Led Fan 1

□ Pass through messages from input.

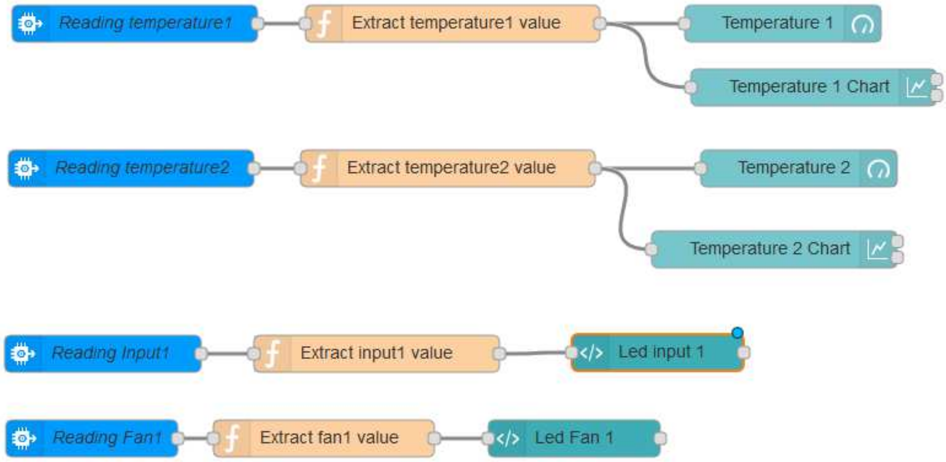
□ Add output messages to stored state.

Template

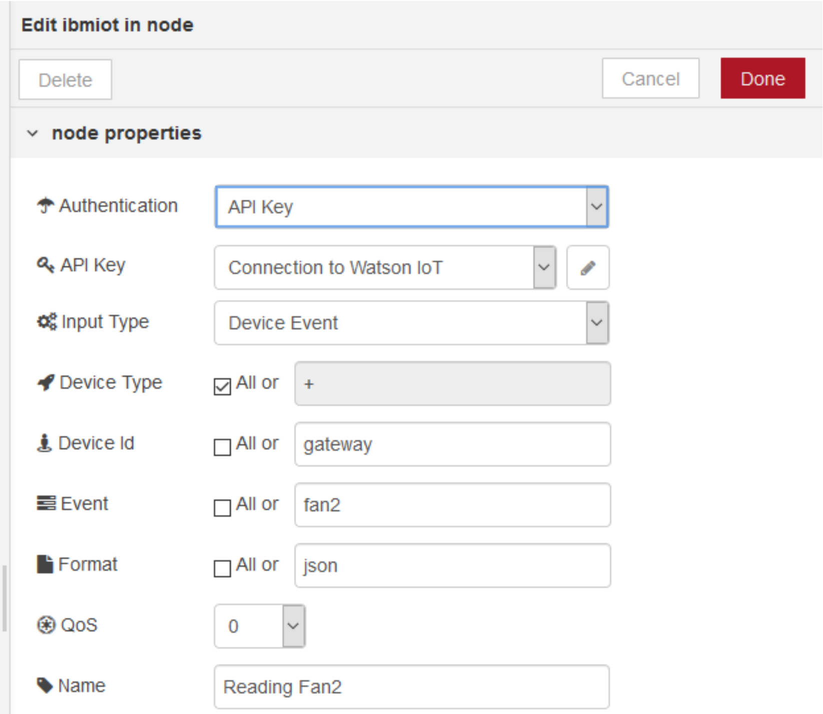
```
<div layout="row" layout-align="space-around center">
    <p>
        FAN 1 STATUS
    </p>

    <p>
        <ng-md-icon icon="toys" ng-style="{fill:msg.payload=='1'?'green':'grey'}" size="30"></ng-md-icon>
    </p>
</div>

```

To continue, click on Done button.

31) Link the node added at the previous point to ‘*Extract fan1 value node*’
At the end, the flow in the working area must be similar to the following one



32) From ‘*input*’ menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.
Once the node is in the working area, double click on it and a panel as the one described at point 14 is shown. Select for the field ‘API Key’ the option ‘*Connect to Watson IoT*’. For the field ‘Event’, insert the string ‘*fan2*’. At the end of node’s configuration, the result is as the following one



Edit ibmiot in node

Delete | Cancel | Done

node properties

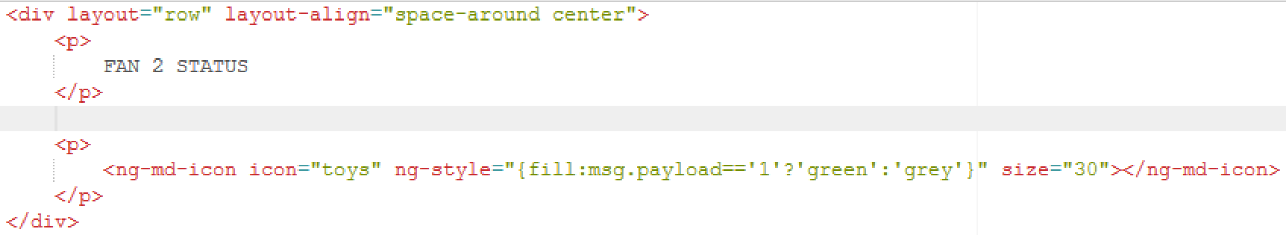
- Authentication: API Key
- API Key: Connection to Watson IoT
- Input Type: Device Event
- Device Type: ☑ All or +
- Device Id: □ All or gateway
- Event: □ All or fan2
- Format: □ All or json
- QoS: 0
- Name: Reading Fan2

To continue, click on Done button.

33) From ‘*function*’ menu in the left bar, select ***function*** node function, drag & drop it into the working area.
Double click on it and edit it as shown at point 15.

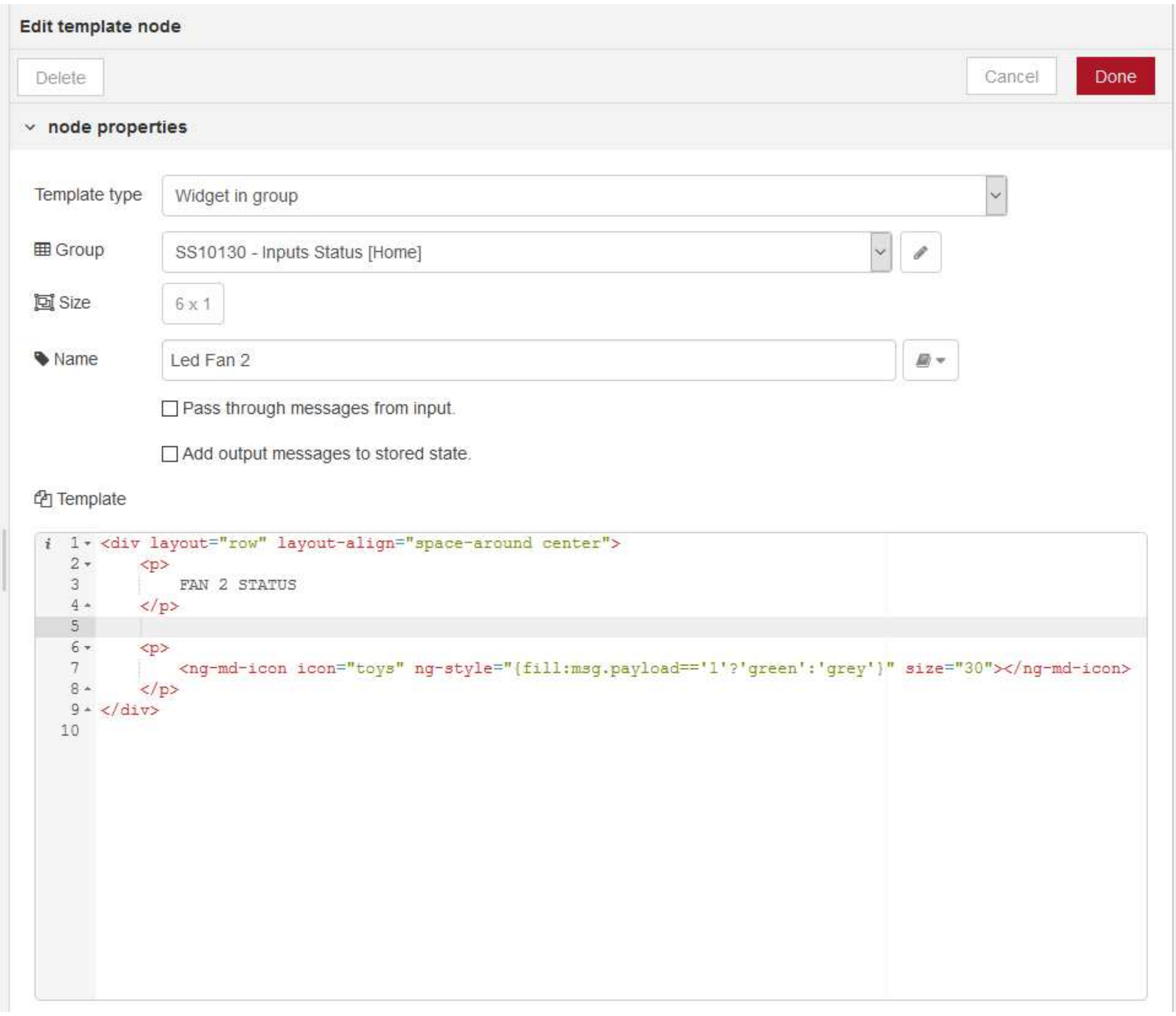
34) From ‘*dashboard*’ menu in the left bar, select ***template*** node template e drag & drop it into the working area. Double click on it and edit it as shown below

- Template type: select ‘*Widget in group*’
- Group: select ‘*SS10130 – Inputs Status*’
- Size: set ‘*6x1*’
- Name: enter ‘*Led fan 2*’
- Don’t check ‘*Pass through messages from input*’ checkbox
- Don’t check ‘*Add output messages to stored state.*’ checkbox
- Template: enter the following code



```
<div layout="row" layout-align="space-around center">
    <p>
        FAN 2 STATUS
    </p>

    <p>
        <ng-md-icon icon="toys" ng-style="{fill:msg.payload=='1'?'green':'grey'}" size="30"></ng-md-icon>
    </p>
</div>
```

At the end of node's configuration, the result is the following one



Edit template node

Delete   Cancel   Done

⌄ **node properties**

Template type: Widget in group

Group: SS10130 - Inputs Status [Home]

Size: 6 x 1

Name: Led Fan 2

☐ Pass through messages from input.

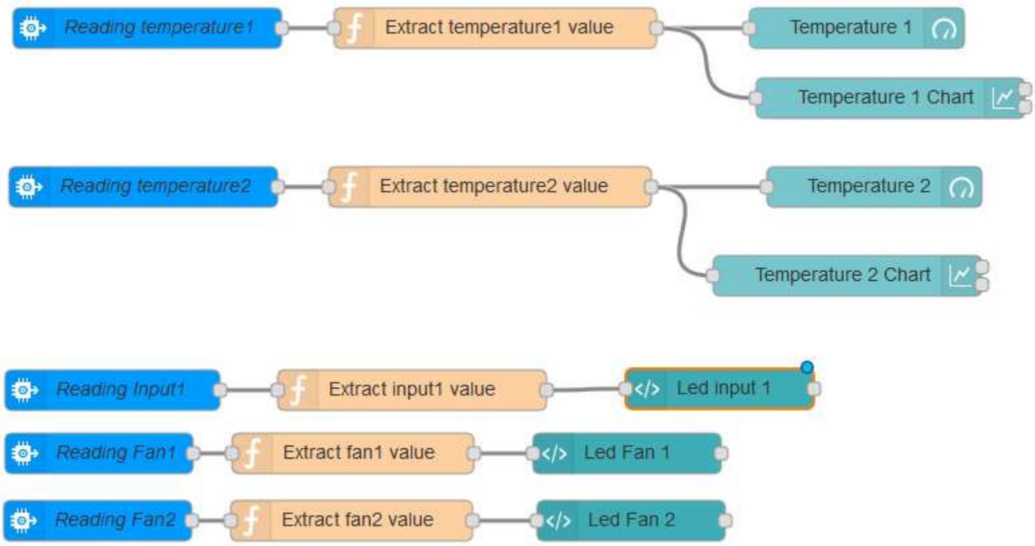
☐ Add output messages to stored state.

Template

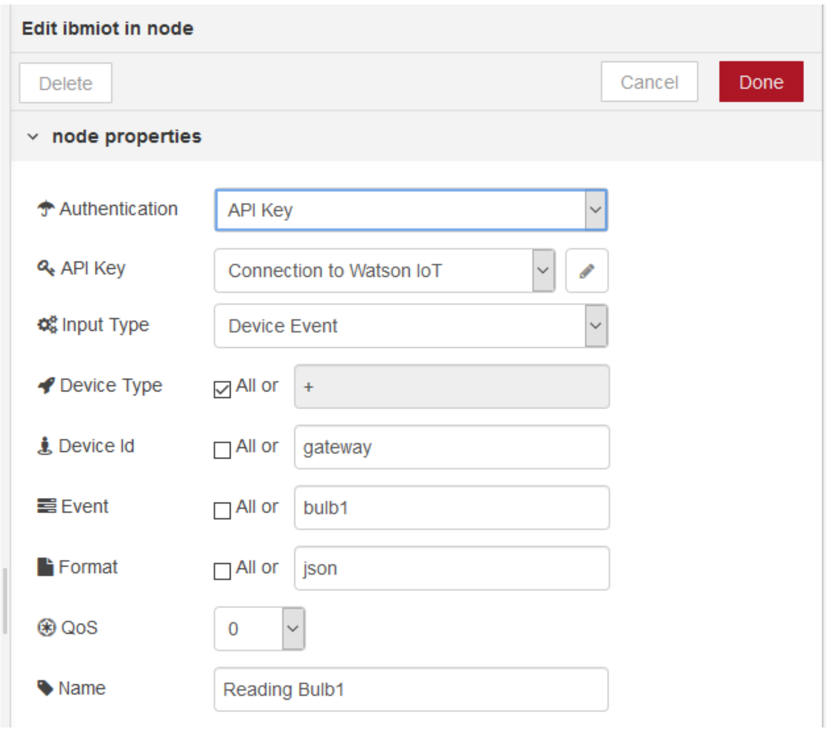
```
<div layout="row" layout-align="space-around center">
    <p>
        FAN 2 STATUS
    </p>

    <p>
        <ng-md-icon icon="toys" ng-style="{fill:msg.payload=='1'?'green':'grey'}" size="30"></ng-md-icon>
    </p>
</div>
```

To continue, click on Done button.

35) Link the node added at the previous point to '*Extract fan2 value*' node.
At the end, the flow in the working area must be similar to the following one



36) From ‘*input*’ menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.
Once the node is in the working area, double click on it and a panel as the one described at point 14 is shown. Select for the field ‘API Key’ the option ‘*Connect to Watson IoT*’. For the field ‘Event’, insert the string ‘*bulb1*’. At the end of node’s configuration, the result is as the following one



Edit ibmiot in node

Delete | Cancel | Done

node properties

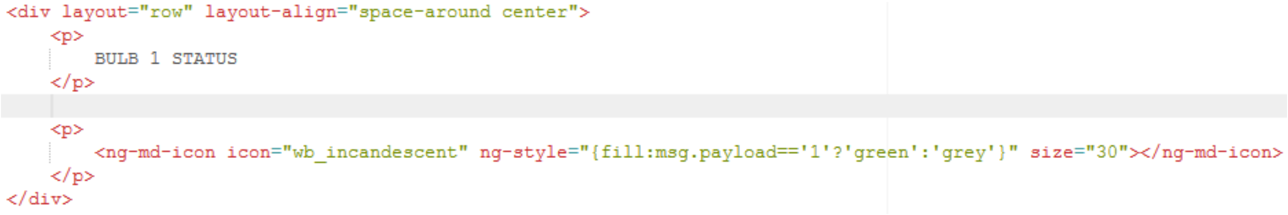
Authentication	API Key
API Key	Connection to Watson IoT
Input Type	Device Event
Device Type	☑ All or +
Device Id	□ All or gateway
Event	□ All or bulb1
Format	□ All or json
QoS	0
Name	Reading Bulb1

To continue, click on Done button.

37) From ‘*function*’ menu in the left bar, select ***function*** node function, drag & drop it into the working area.
Double click on it and edit it as shown at point 15.

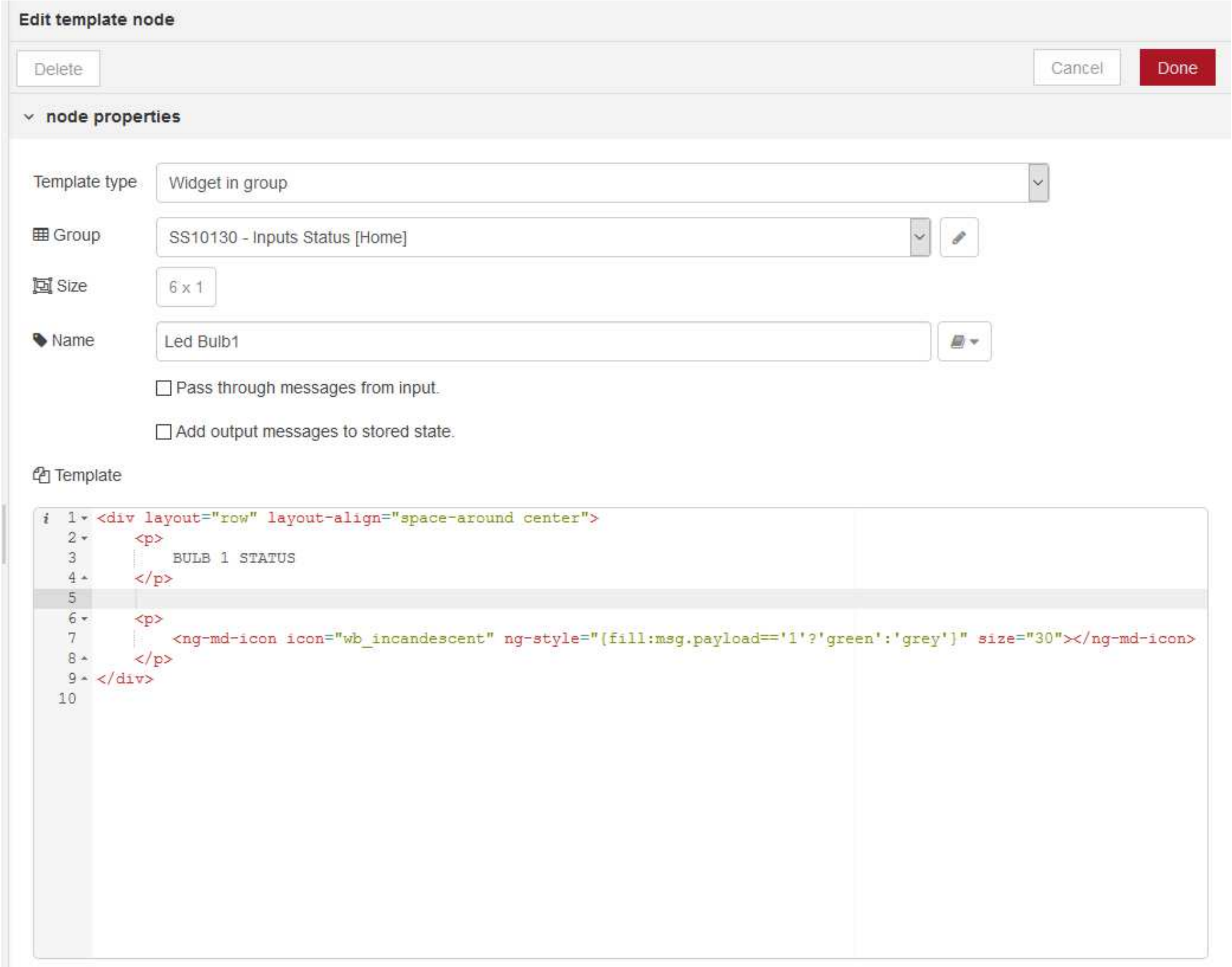
38) From ‘*dashboard*’ menu in the left bar, select ***template*** node template e drag & drop it into the working area. Double click on it and edit it as shown below

- Template type: select ‘*Widget in group*’
- Group: select ‘*SS10130 – Inputs Status*’
- Size: set ‘*6x1*’
- Name: enter ‘*Led bulb 1*’
- Don’t check ‘*Pass through messages from input*’ checkbox
- Don’t check ‘*Add output messages to stored state.*’ checkbox
- Template: enter the following code



```
<div layout="row" layout-align="space-around center">
    <p>
        BULB 1 STATUS
    </p>

    <p>
        <ng-md-icon icon="wb_incandescent" ng-style="{fill:msg.payload=='1'?'green':'grey'}" size="30"></ng-md-icon>
    </p>
</div>
```

At the end of node's configuration, the result is the following one



Template type: Widget in group

Group: SS10130 - Inputs Status [Home]

Size: 6 x 1

Name: Led Bulb1

□ Pass through messages from input.

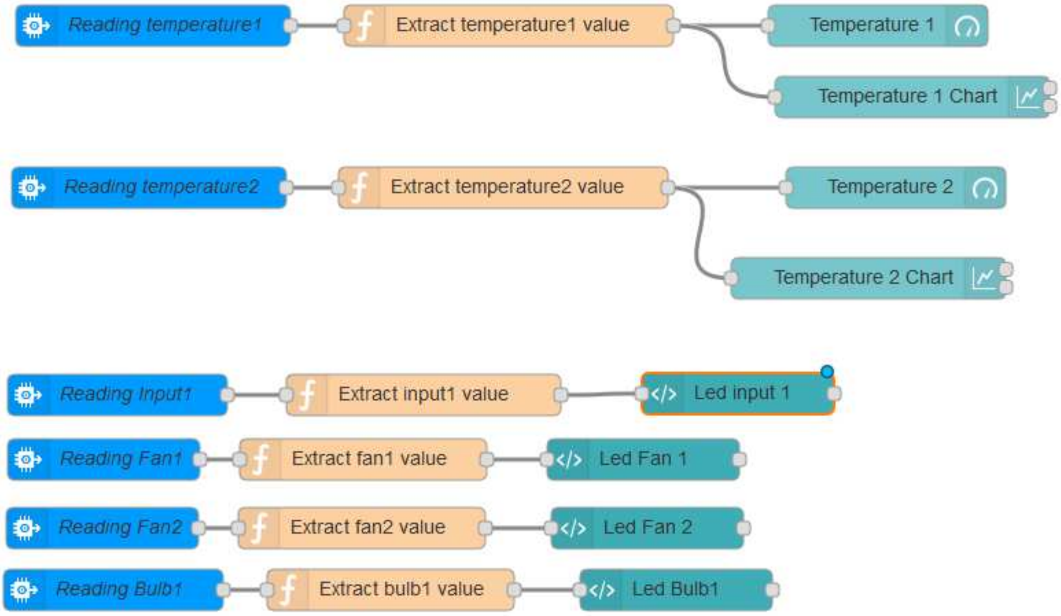
□ Add output messages to stored state.

Template

```
<div layout="row" layout-align="space-around center">
    <p>
        BULB 1 STATUS
    </p>

    <p>
        <ng-md-icon icon="wb_incandescent" ng-style="{fill:msg.payload=='1'?'green':'grey'}" size="30"></ng-md-icon>
    </p>
</div>

```

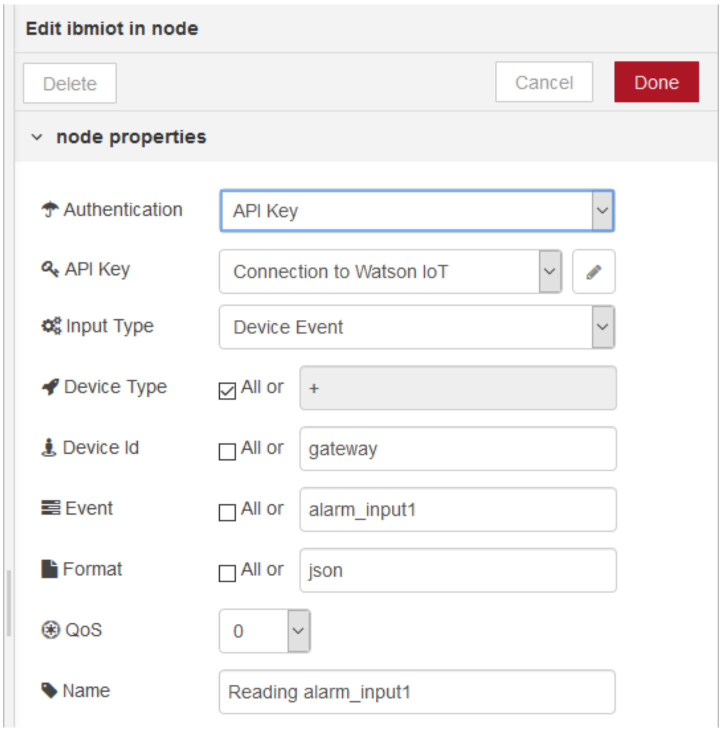
To continue, click on Done button.

39) Link the node added at the previous point to '*Extract bulb1 value*' node.
At the end, the flow in the working area must be similar to the following one



40) Save the flow by clicking on Deploy button in the upper right corner of the window

41) From ‘*input*’ menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.
Once the node is in the working area, double click on it and a panel as the one described at point 14 is shown. Select for the field ‘API Key’ the option ‘*Connect to Watson IoT*’. For the field ‘Event’, insert the string ‘*alarm_input1*’. At the end of node’s configuration, the result is as the following one



Edit ibmiot in node

Authentication: API Key

API Key: Connection to Watson IoT

Input Type: Device Event

Device Type: ☑ All or +

Device Id: □ All or gateway

Event: □ All or alarm_input1

Format: □ All or json

QoS: 0

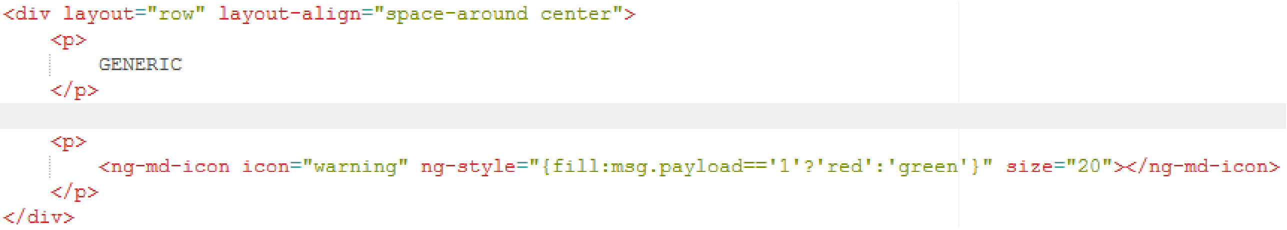
Name: Reading alarm_input1

To continue, click on Done button.

42) From ‘*function*’ menu in the left bar, select ***function*** node function, drag & drop it into the working area.
Double click on it and edit it as shown at point 15.

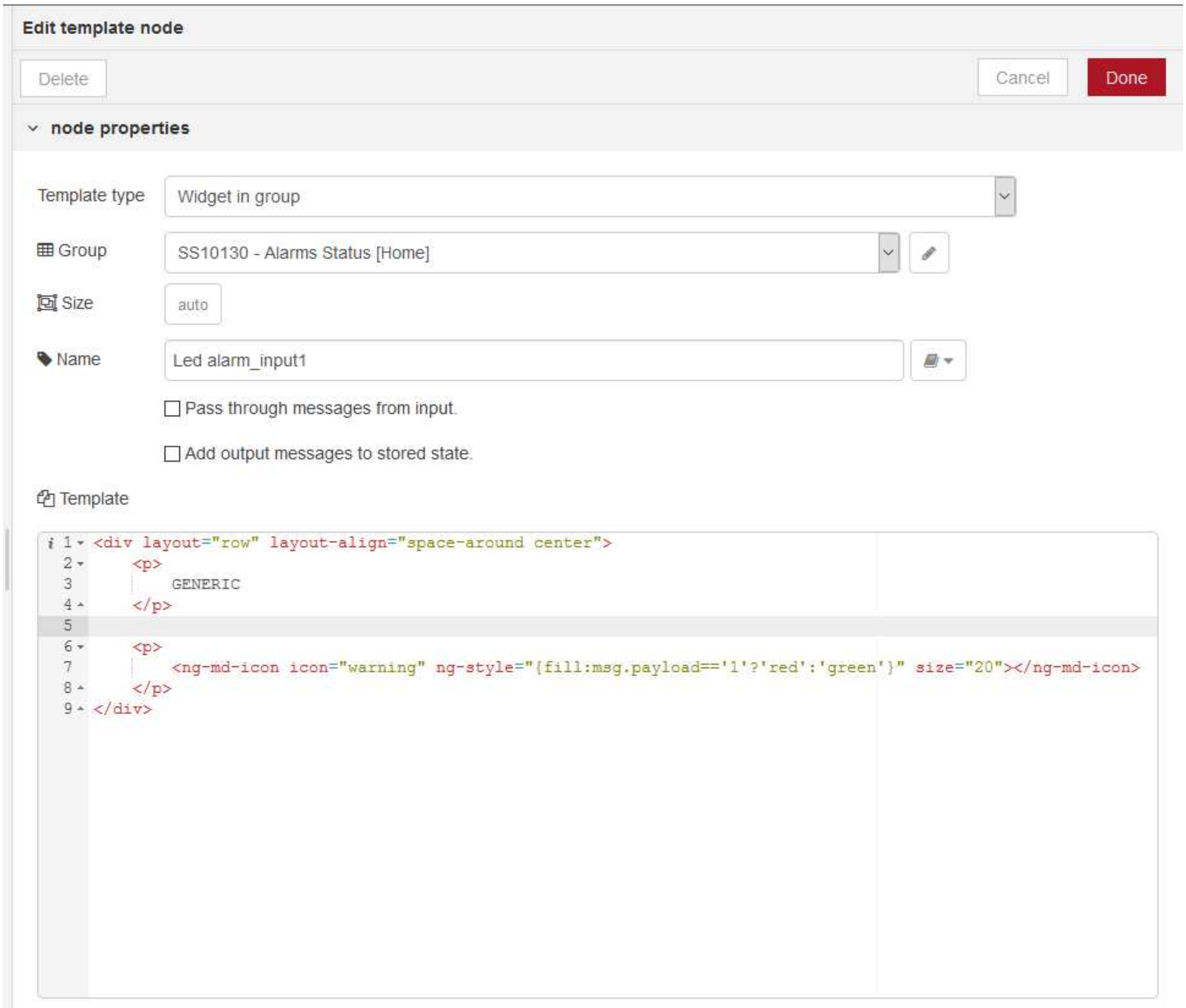
43) From ‘*dashboard*’ menu in the left bar, select ***template*** node template e drag & drop it into the working area. Double click on it and edit it as shown below

- Template type: select ‘*Widget in group*’
- Group: click on ✎ and add a new group ‘*SS10130 – Alarm Status*’
- Size: set ‘*auto*’
- Name: enter ‘*Led alarm input 1*’
- Don’t check ‘*Pass through messages from input*’ checkbox
- Don’t check ‘*Add output messages to stored state.*’ checkbox
- Template: enter the following code



```
<div layout="row" layout-align="space-around center">
    <p>
        GENERIC
    </p>

    <p>
        <ng-md-icon icon="warning" ng-style="{fill:msg.payload=='1'?'red':'green'}" size="20"></ng-md-icon>
    </p>
</div>
```

At the end of node's configuration, the result is the following one



Template type: Widget in group

Group: SS10130 - Alarms Status [Home]

Size: auto

Name: Led alarm_input1

□ Pass through messages from input.

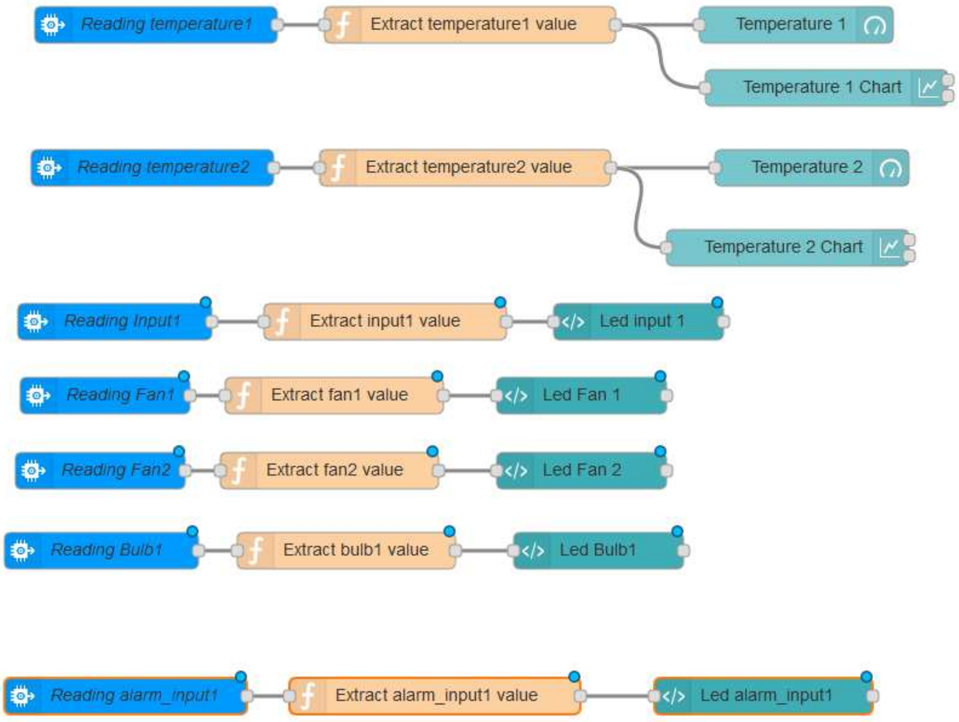
□ Add output messages to stored state.

Template

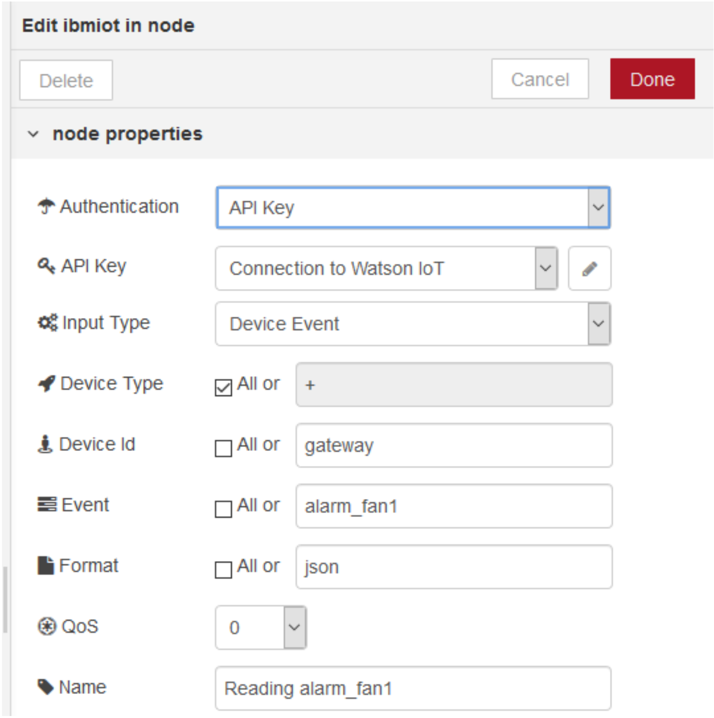
```
<div layout="row" layout-align="space-around center">
    <p>
        GENERIC
    </p>

    <p>
        <ng-md-icon icon="warning" ng-style="{fill:msg.payload=='1'?'red':'green'}" size="20"></ng-md-icon>
    </p>
</div>
```

To continue, click on Done button.

44) Link the node added at the previous point to ‘*Extract alarm_input1 value*’ node.
At the end, the flow in the working area must be similar to the following one



45) From ‘*input*’ menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.
Once the node is in the working area, double click on it and a panel as the one described at point 14 is shown. Select for the field ‘API Key’ the option ‘*Connect to Watson IoT*’. For the field ‘Event’, insert the string ‘*alarm_fan1*’. At the end of node’s configuration, the result is as the following one



Edit ibmiot in node

Delete Cancel Done

node properties

Authentication: API Key

API Key: Connection to Watson IoT

Input Type: Device Event

Device Type: ☑ All or +

Device Id: □ All or gateway

Event: □ All or alarm_fan1

Format: □ All or json

QoS: 0

Name: Reading alarm_fan1

To continue, click on Done button.

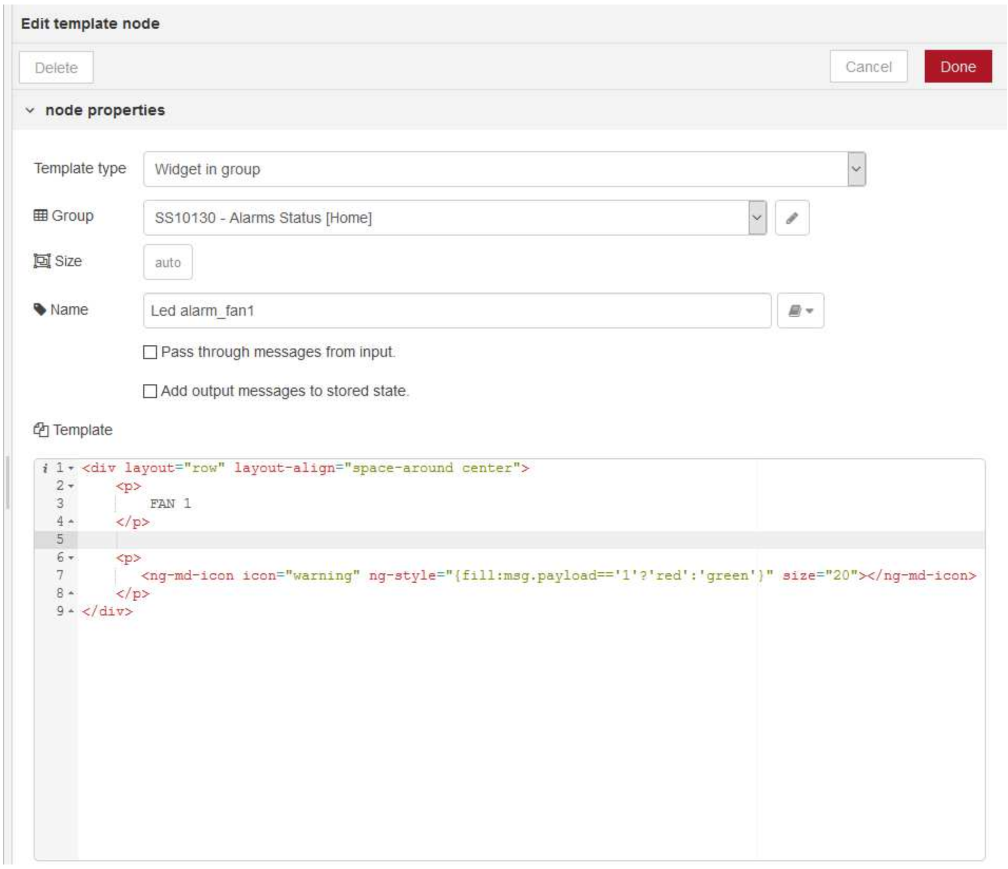
46) From '*function*' menu in the left bar, select ***function*** node function, drag & drop it into the working area.
Double click on it and edit it as shown at point 15.

47) From '*dashboard*' menu in the left bar, select ***template*** node </> template e drag & drop it into the working area. Double click on it and edit it as shown below

- Template type: select '*Widget in group*'
- Group: select '*SS10130 – Alarm Status*'
- Size: set '*auto*'
- Name: enter '*Led alarm fan 1*'
- Don't check '*Pass through messages from input*' checkbox
- Don't check '*Add output messages to stored state.*' checkbox
- Template: enter the following code

```
<div layout="row" layout-align="space-around center">
    <p>
        FAN 1
    </p>

    <p>
       <ng-md-icon icon="warning" ng-style="{fill:msg.payload=='1'?'red':'green'}" size="20"></ng-md-icon>
    </p>
</div>
```

At the end of node's configuration, the result is the following one



Edit template node

Delete | Cancel | Done

node properties

Template type: Widget in group

Group: SS10130 - Alarms Status [Home]

Size: auto

Name: Led alarm_fan1

□ Pass through messages from input.

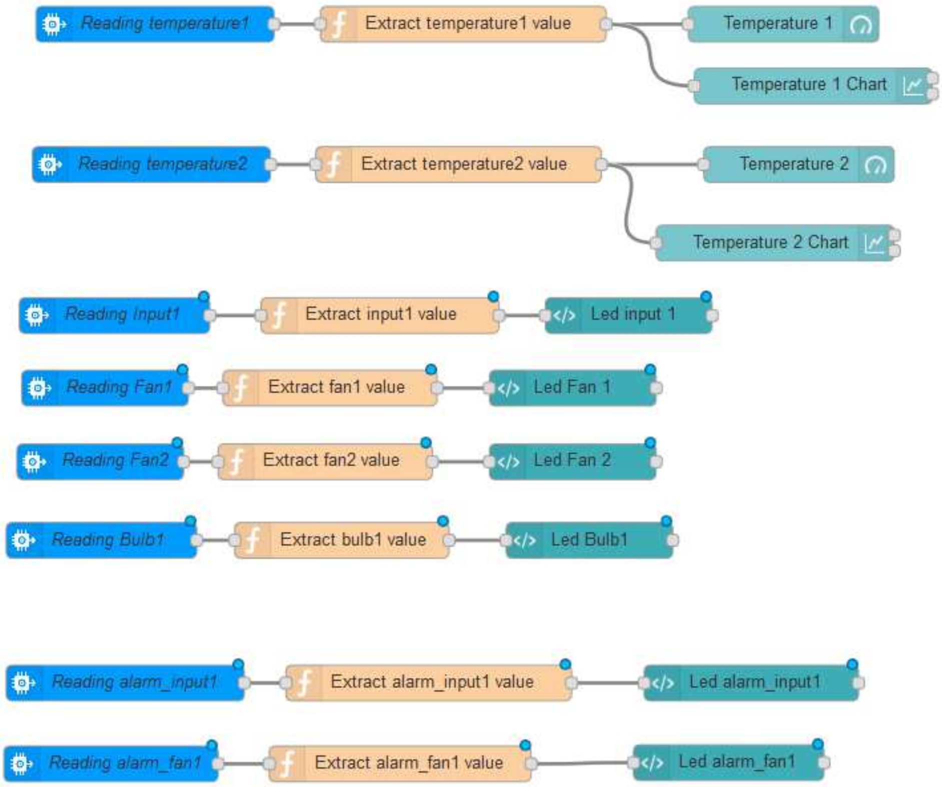
□ Add output messages to stored state.

Template

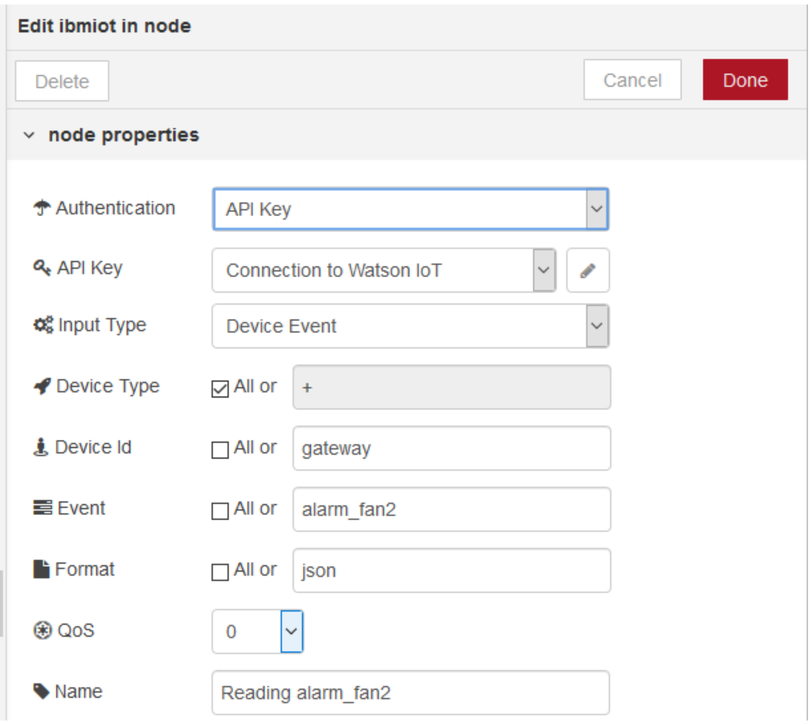
```
<div layout="row" layout-align="space-around center">
    <p>
        FAN 1
    </p>

    <p>
       <ng-md-icon icon="warning" ng-style="{fill:msg.payload=='1'?'red':'green'}" size="20"></ng-md-icon>
    </p>
</div>
```

To continue, click on Done button.

48) Link the node added at the previous point to '*Extract alarm_fan1 value*' node.
At the end, the flow in the working area must be similar to the following one



49) From ‘*input*’ menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.
Once the node is in the working area, double click on it and a panel as the one described at point 14 is shown. Select for the field ‘API Key’ the option ‘*Connect to Watson IoT*’. For the field ‘Event’, insert the string ‘*alarm_fan2*’. At the end of node’s configuration, the result is as the following one



Edit ibmiot in node

Delete | Cancel | Done

node properties

Authentication: API Key
API Key: Connection to Watson IoT
Input Type: Device Event
Device Type: ☑ All or +
Device Id: □ All or gateway
Event: □ All or alarm_fan2
Format: □ All or json
QoS: 0
Name: Reading alarm_fan2

To continue, click on Done button.

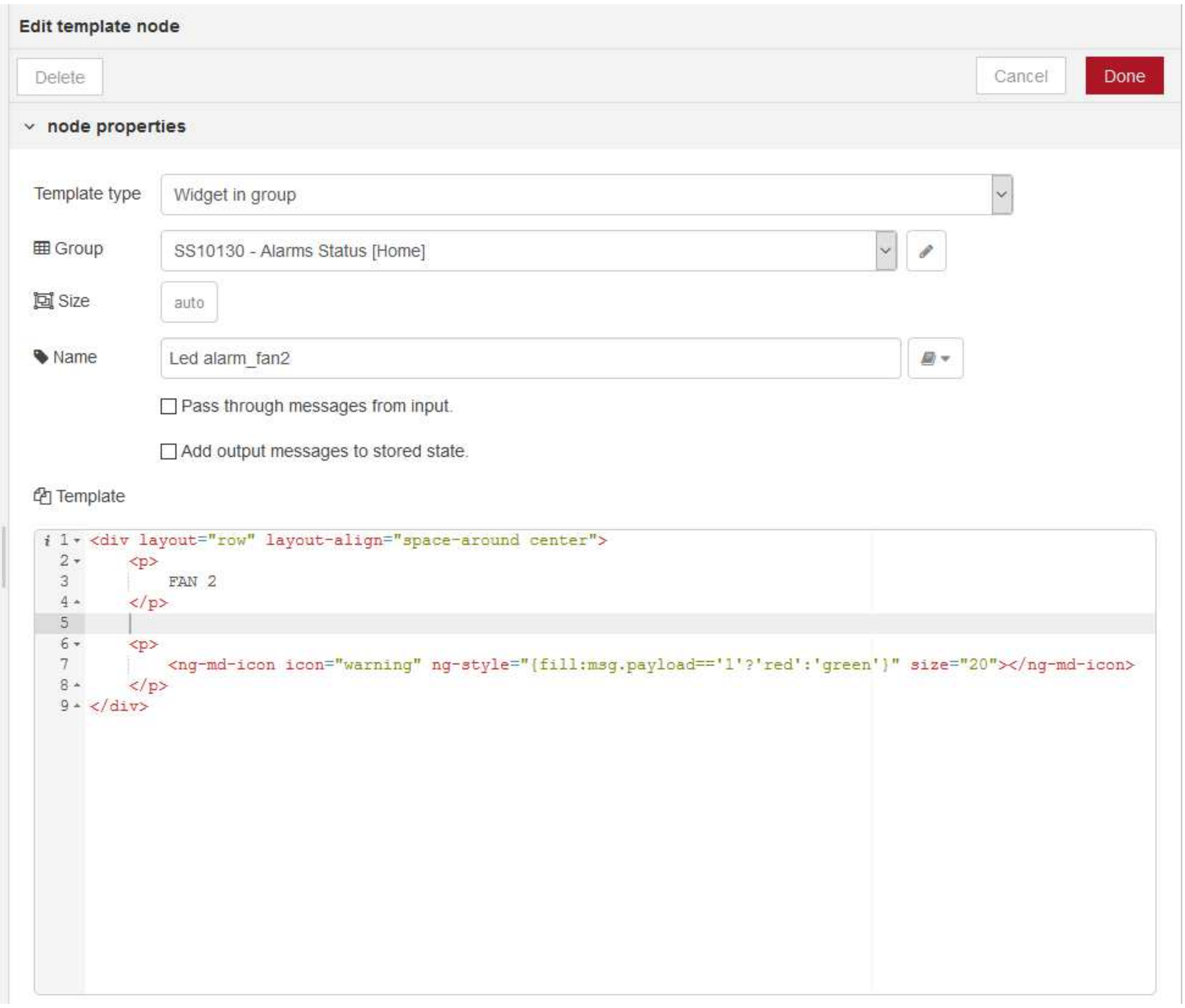
50) From ‘*function*’ menu in the left bar, select ***function*** node function, drag & drop it into the working area.
Double click on it and edit it as shown at point 15.

51) From ‘*dashboard*’ menu in the left bar, select ***template*** node template e drag & drop it into the working area. Double click on it and edit it as shown below

- Template type: select ‘*Widget in group*’
- Group: select ‘*SS10130 – Alarm Status*’
- Size: set ‘*auto*’
- Name: enter ‘*Led alarm fan 2*’
- Don’t check ‘*Pass through messages from input*’ checkbox
- Don’t check ‘*Add output messages to stored state.*’ checkbox
- Template: enter the following code

```
<div layout="row" layout-align="space-around center">
    <p>
        FAN 2
    </p>

    <p>
        <ng-md-icon icon="warning" ng-style="{fill:msg.payload=='1'?'red':'green'}" size="20"></ng-md-icon>
    </p>
</div>
```

At the end of node’s configuration, the result is the following one



Template type: Widget in group

Group: SS10130 - Alarms Status [Home]

Size: auto

Name: Led alarm_fan2

☐ Pass through messages from input.

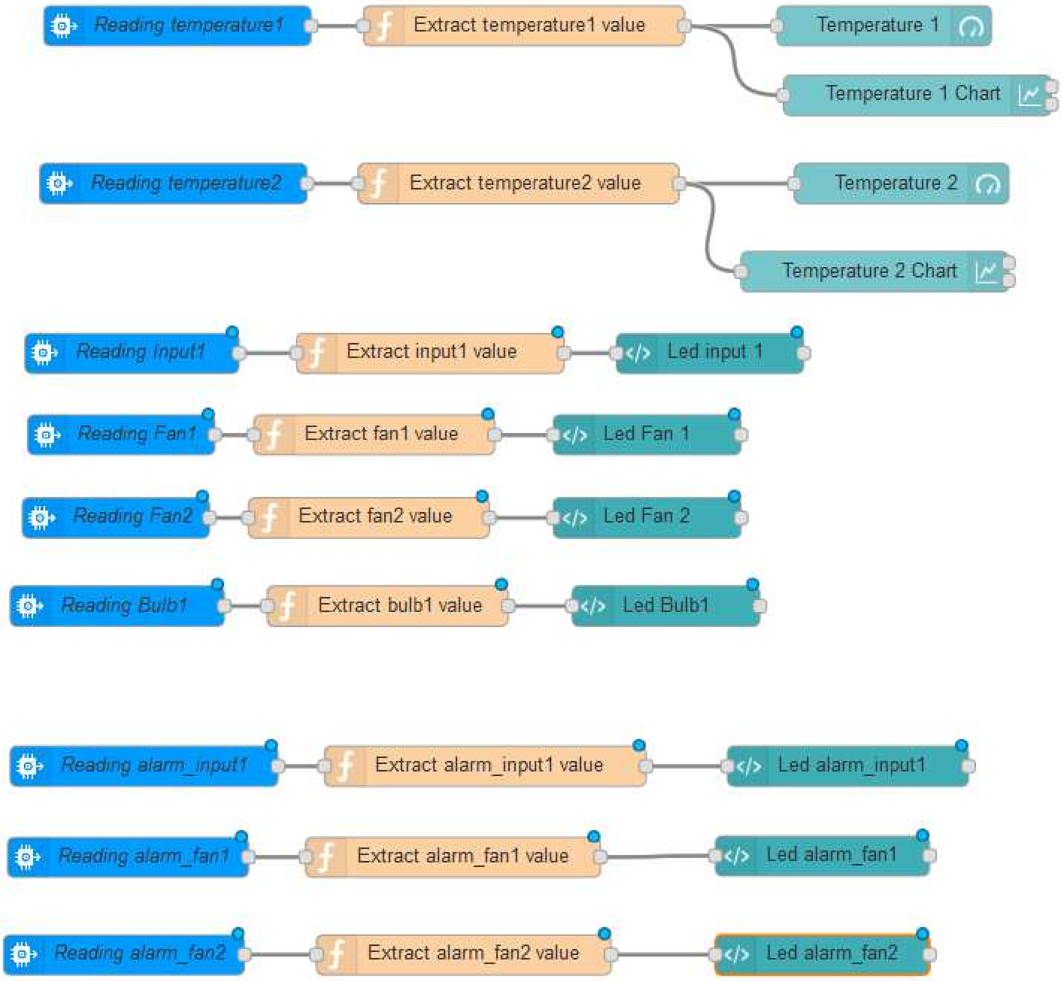
☐ Add output messages to stored state.

Template

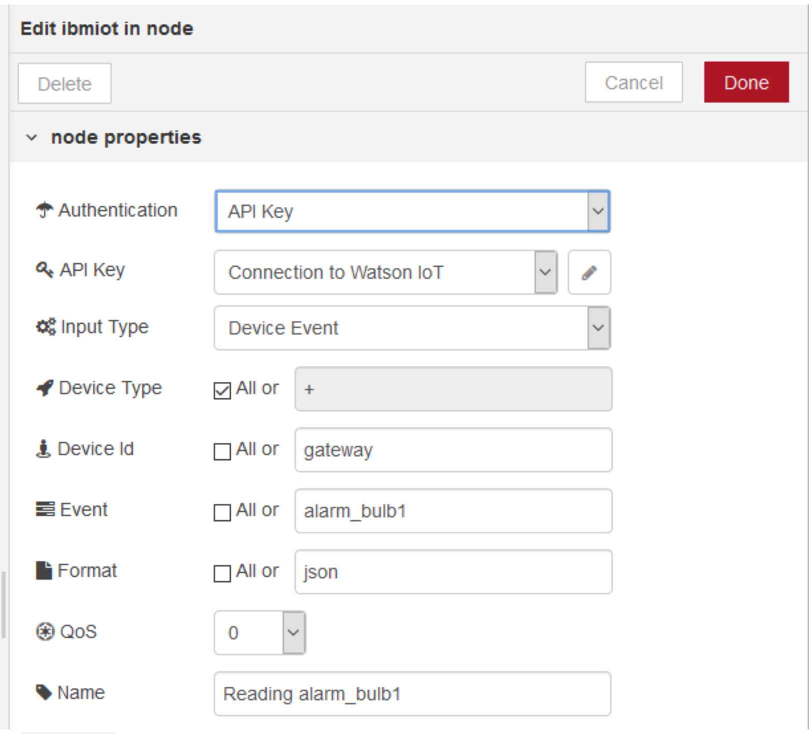
```
<div layout="row" layout-align="space-around center">
    <p>
        FAN 2
    </p>
    
    <p>
        <ng-md-icon icon="warning" ng-style="{fill:msg.payload=='1'?'red':'green'}" size="20"></ng-md-icon>
    </p>
</div>
```

To continue, click on Done button.

52) Link the node added at the previous point to '*Extract alarm_fan2 value*' node.
At the end, the flow in the working area must be similar to the following one



53) From ‘*input*’ menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.
Once the node is in the working area, double click on it and a panel as the one described at point 14 is shown. Select for the field ‘API Key’ the option ‘*Connect to Watson IoT*’. For the field ‘Event’, insert the string ‘*alarm_bulb1*’. At the end of node’s configuration, the result is as the following one



Edit ibmiot in node

Delete | Cancel | Done

node properties

- Authentication: API Key
- API Key: Connection to Watson IoT
- Input Type: Device Event
- Device Type: ☑ All or +
- Device Id: □ All or gateway
- Event: □ All or alarm_bulb1
- Format: □ All or json
- QoS: 0
- Name: Reading alarm_bulb1

To continue, click on Done button.

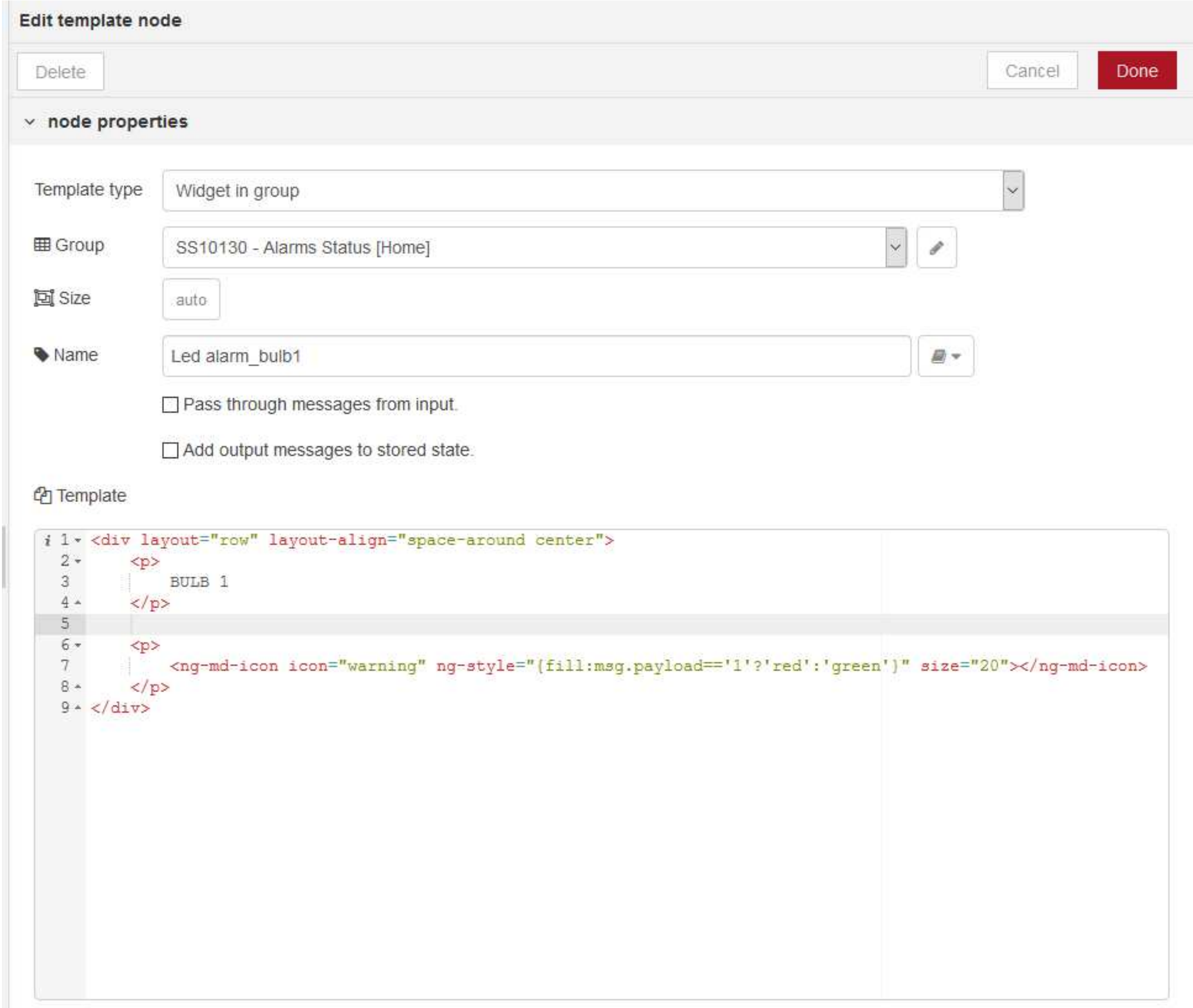
54) From ‘*function*’ menu in the left bar, select ***function*** node function, drag & drop it into the working area.
Double click on it and edit it as shown at point 15.

55) From ‘*dashboard*’ menu in the left bar, select ***template*** node template e drag & drop it into the working area. Double click on it and edit it as shown below

- Template type: select ‘*Widget in group*’
- Group: select ‘*SS10130 – Alarm Status*’
- Size: set ‘*auto*’
- Name: enter ‘*Led alarm bulb 1*’
- Don’t check ‘*Pass through messages from input*’ checkbox
- Don’t check ‘*Add output messages to stored state.*’ checkbox
- Template: enter the following code

```
<div layout="row" layout-align="space-around center">
    <p>
        BULB 1
    </p>

    <p>
        <ng-md-icon icon="warning" ng-style="{fill:msg.payload=='1'?'red':'green'}" size="20"></ng-md-icon>
    </p>
</div>
```

At the end of node's configuration, the result is the following one



Template type: Widget in group

Group: SS10130 - Alarms Status [Home]

Size: auto

Name: Led alarm_bulb1

□ Pass through messages from input.

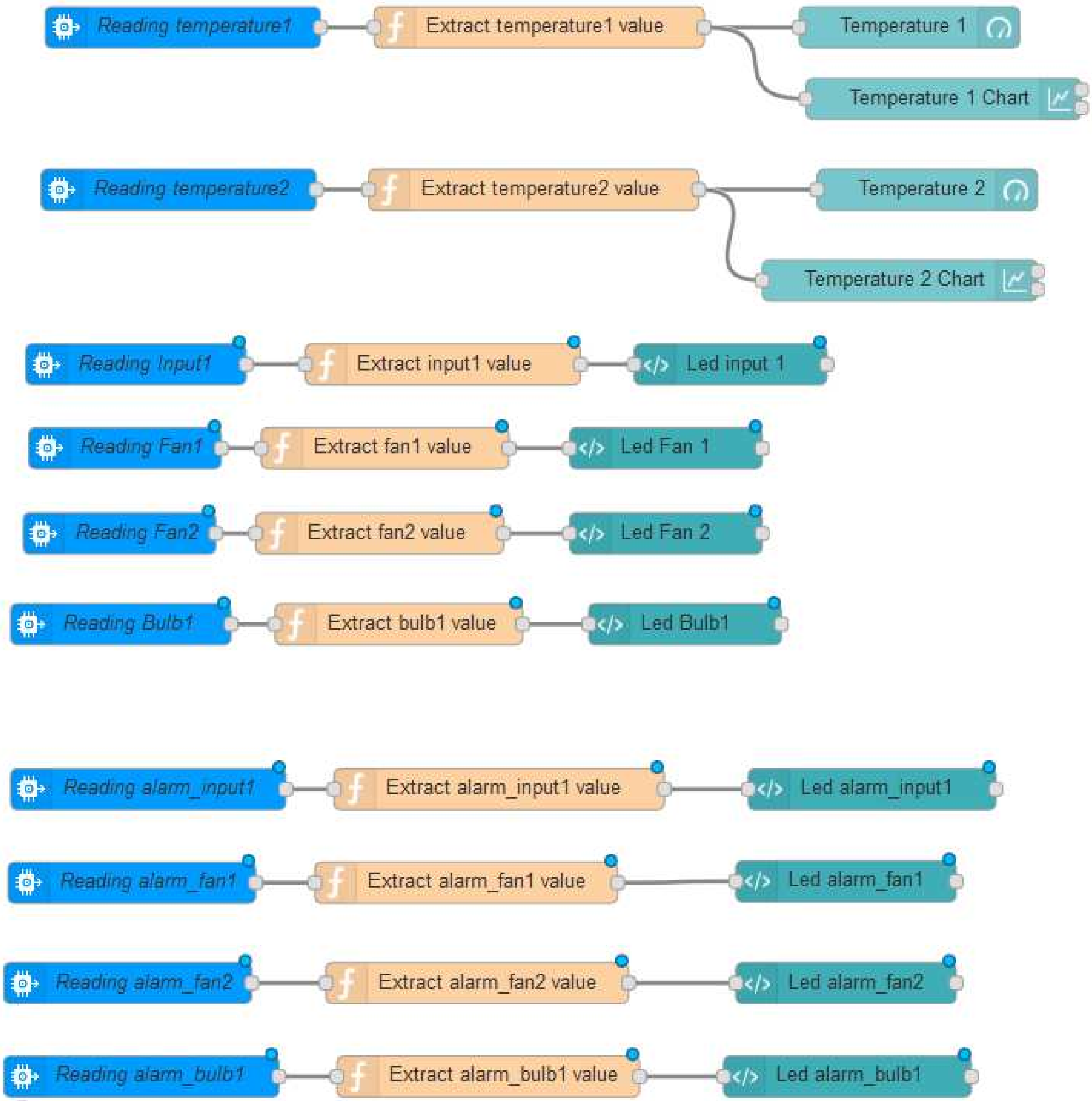
□ Add output messages to stored state.

Template

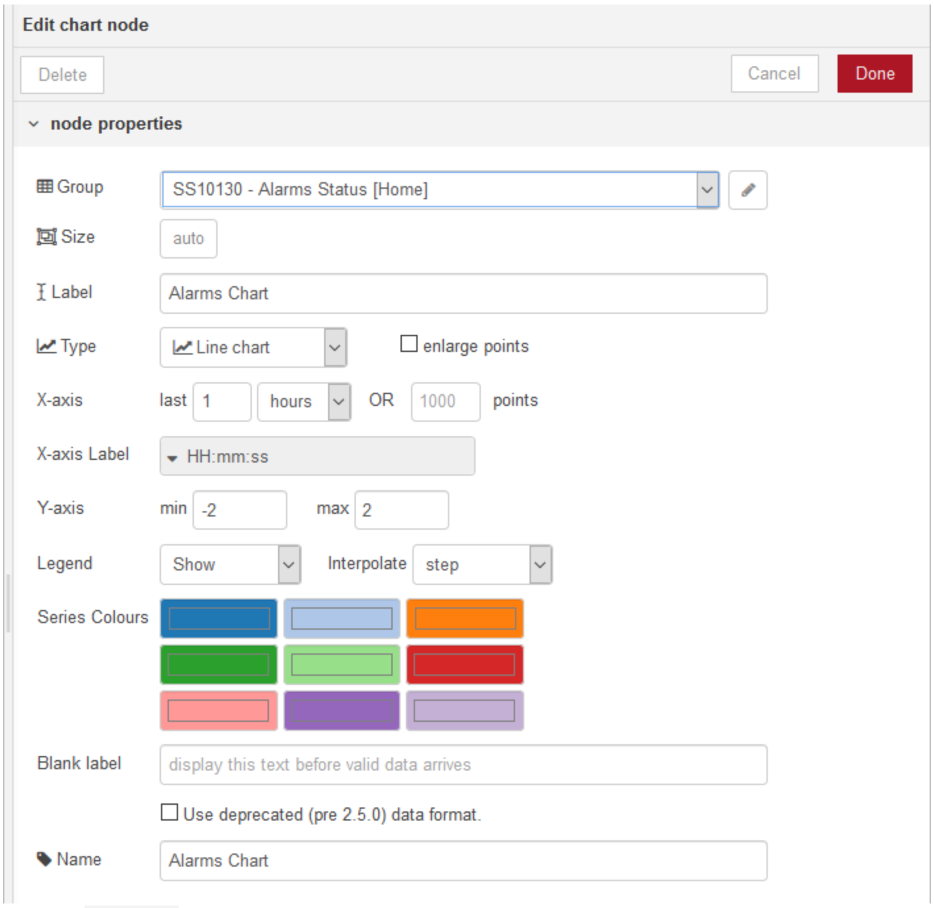
```
<div layout="row" layout-align="space-around center">
    <p>
        BULB 1
    </p>

    <p>
        <ng-md-icon icon="warning" ng-style="{fill:msg.payload=='1'?'red':'green'}" size="20"></ng-md-icon>
    </p>
</div>
```

To continue, click on Done button.

56) Link the node added at the previous point and '*Extract alarm_bulb1 value*' node.
At the end, the flow in the working area must be similar to the following one



57) From '*dashboard*' menu in the left bar, select ***chart*** node chart and drag & drop it into the working area. Double click on it and edit it as shown below



Edit chart node

Delete | Cancel | Done

node properties

Group: SS10130 - Alarms Status [Home]

Size: auto

Label: Alarms Chart

Type: Line chart ☐ enlarge points

X-axis: last 1 hours OR 1000 points

X-axis Label: HH:mm:ss

Y-axis: min -2 max 2

Legend: Show Interpolate: step

Series Colours: 🟦 🟦 🟧 🟩 🟩 🟥 🟥 🟪 🟪

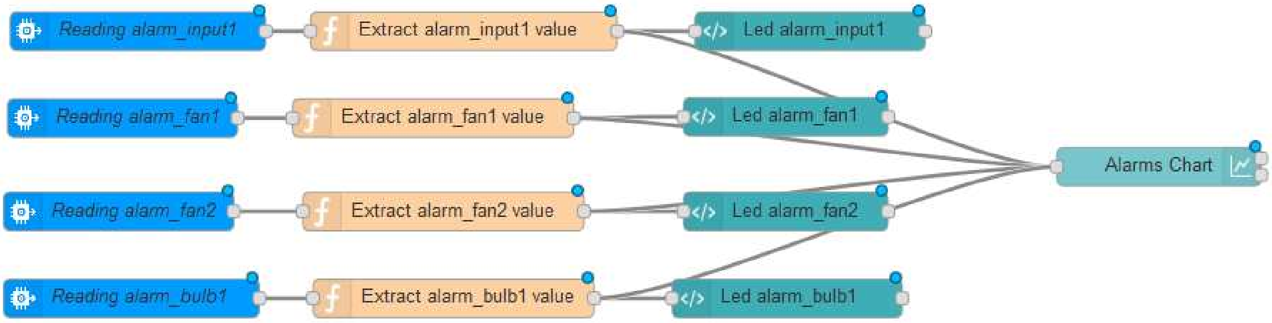
Blank label: display this text before valid data arrives

☐ Use deprecated (pre 2.5.0) data format.

Name: Alarms Chart

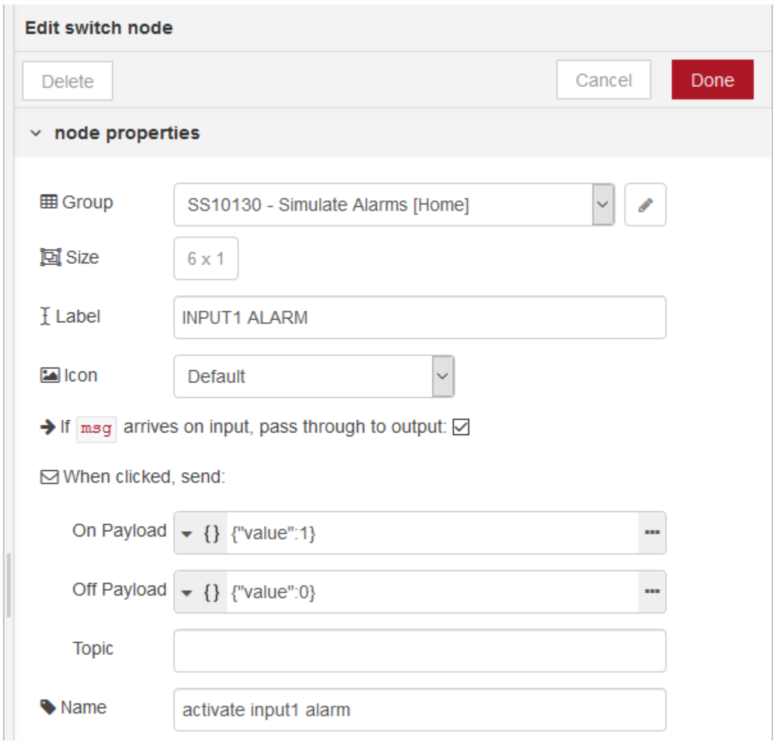
To continue, click on Done button.

58) Link '*Extract alarm_input1 value*', '*Extract alarm_fan1 value*', '*Extract alarm_fan2*', '*Extract alarm_bulb1*' nodes to the node added at the previous point. The result must be similar to the one shown below



59) From '*dashboard*' menu in the left bar, select ***switch*** node switch and drag & drop it into the working area. Double click on it and edit it as described below

- Group: click on ✎ and add a new group '*SS10130 – Simulate Alarms*'
- Size: set '*6x1*'
- Name: enter '*INPUT1 ALARM*'
- Icon: select '*Default*'
- On payload: from menu select JSON and enter '*{"value":1}*'
- Off payload: from menu select JSON and enter '*{"value":0}*'
- Topic: leave blank
- Name: enter '*activate input1 alarm*'



Edit switch node

Delete | Cancel | Done

⌄ **node properties**

Group: SS10130 - Simulate Alarms [Home] ✎

Size: 6 x 1

Label: INPUT1 ALARM

Icon: Default

If msg arrives on input, pass through to output: ☑

When clicked, send:

On Payload: {} {"value":1}

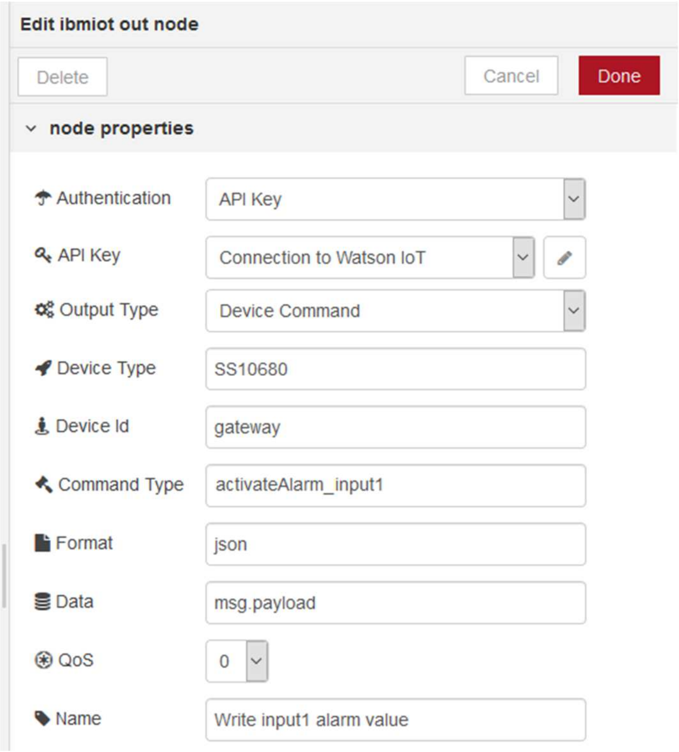
Off Payload: {} {"value":0}

Topic:

Name: activate input1 alarm

To continue, click on Done button.

60) From ‘*output*’ menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.
Double click on it and edit it as shown below



Edit ibmiot out node

Delete Cancel Done

node properties

Authentication: API Key

API Key: Connection to Watson IoT

Output Type: Device Command

Device Type: SS10680

Device Id: gateway

Command Type: activateAlarm_input1

Format: json

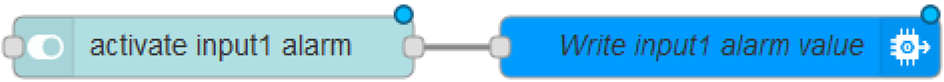
Data: msg.payload

QoS: 0

Name: Write input1 alarm value

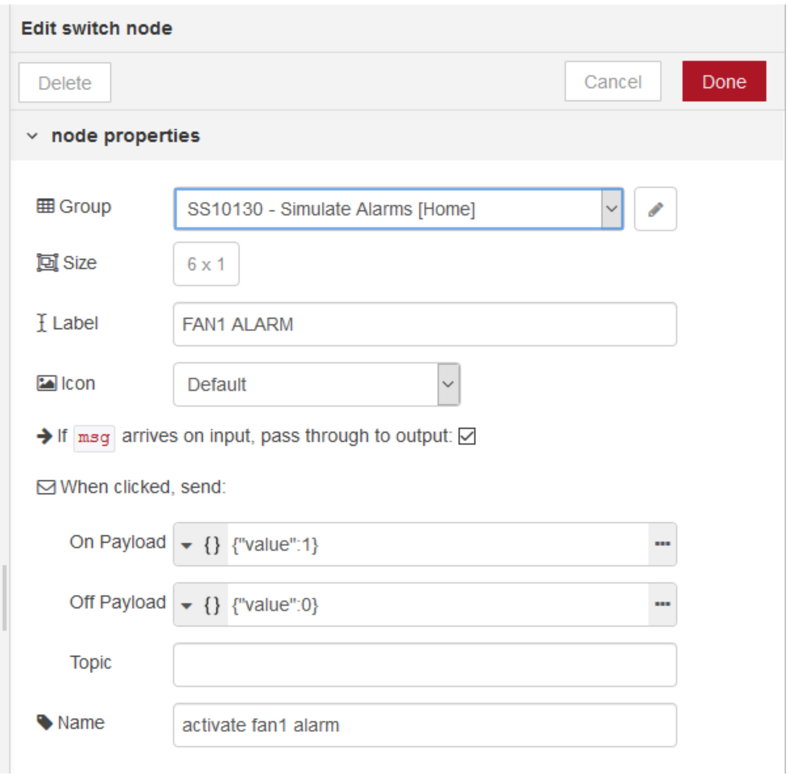
To continue, click on Done button.

61) Link the nodes added at the previous two points, as shown below



62) From ‘*dashboard*’ menu in the left bar, select ***switch*** node switch and drag & drop it into the working area. Double click on it and edit it as described below

- Group: select ‘*SS10130 – Simulate Alarms*’
- Size: set ‘*6x1*’
- Name: enter ‘*FAN 1 ALARM*’
- Icon: select ‘*Default*’
- On payload: from menu select JSON and enter ‘*{“value”:1}*’
- Off payload: from menu select JSON and enter ‘*{“value”:0}*’
- Topic: leave blank
- Name: enter ‘*activate fan1 alarm*’



Edit switch node

Group: SS10130 - Simulate Alarms [Home]

Size: 6 x 1

Label: FAN1 ALARM

Icon: Default

If msg arrives on input, pass through to output: ☑

When clicked, send:

On Payload: {} {"value":1}

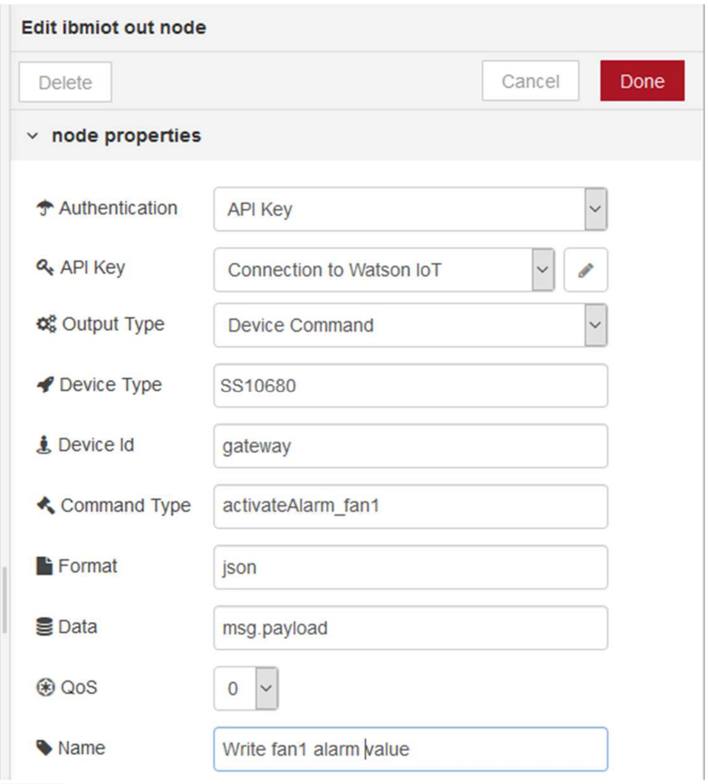
Off Payload: {} {"value":0}

Topic:

Name: activate fan1 alarm

To continue, click on Done button.

63) From '*output*' menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.
Double click on it and edit it as shown below



Edit ibmiot out node

Authentication: API Key

API Key: Connection to Watson IoT

Output Type: Device Command

Device Type: SS10680

Device Id: gateway

Command Type: activateAlarm_fan1

Format: json

Data: msg.payload

QoS: 0

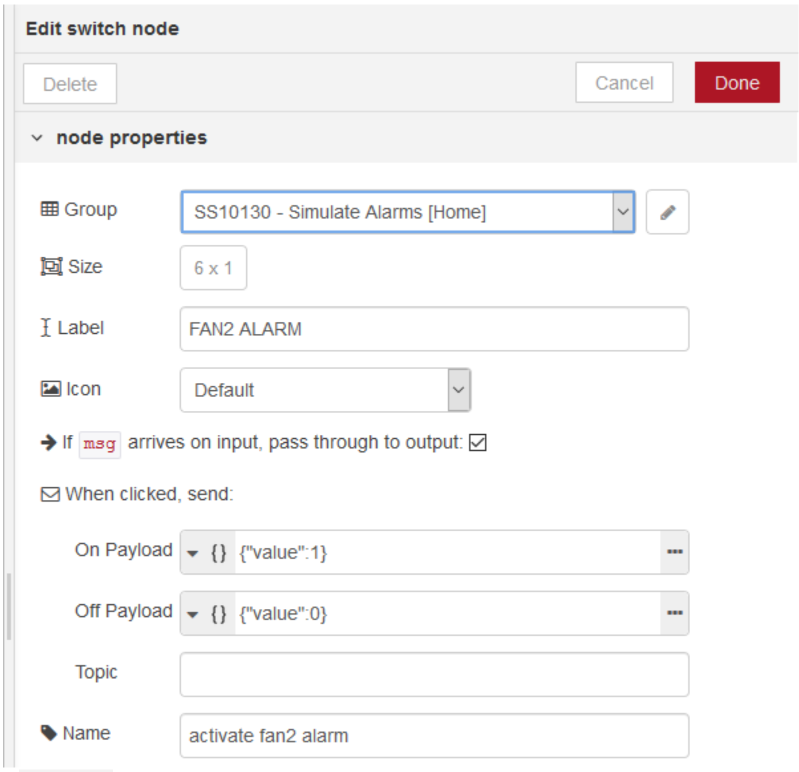
Name: Write fan1 alarm value

To continue, click on Done button.

64) Link the nodes added at the previous two points, as shown at point 61

65) From ‘*dashboard*’ menu in the left bar, select ***switch*** node switch and drag & drop it into the working area. Double click on it and edit it as described below

- Group: select ‘*SS10130 – Simulate Alarms*’
- Size: set ‘*6x1*’
- Name: enter ‘*FAN 2 ALARM*’
- Icon: select ‘*Default*’
- On payload: from menu select JSON and enter ‘*{“value”:1}*’
- Off payload: from menu select JSON and enter ‘*{“value”:0}*’
- Topic: leave blank
- Name: enter ‘*activate fan2 alarm*’



Edit switch node

Delete | Cancel | Done

node properties

Group: SS10130 - Simulate Alarms [Home]

Size: 6 x 1

Label: FAN2 ALARM

Icon: Default

If msg arrives on input, pass through to output: ☑

When clicked, send:

On Payload: {} {"value":1}

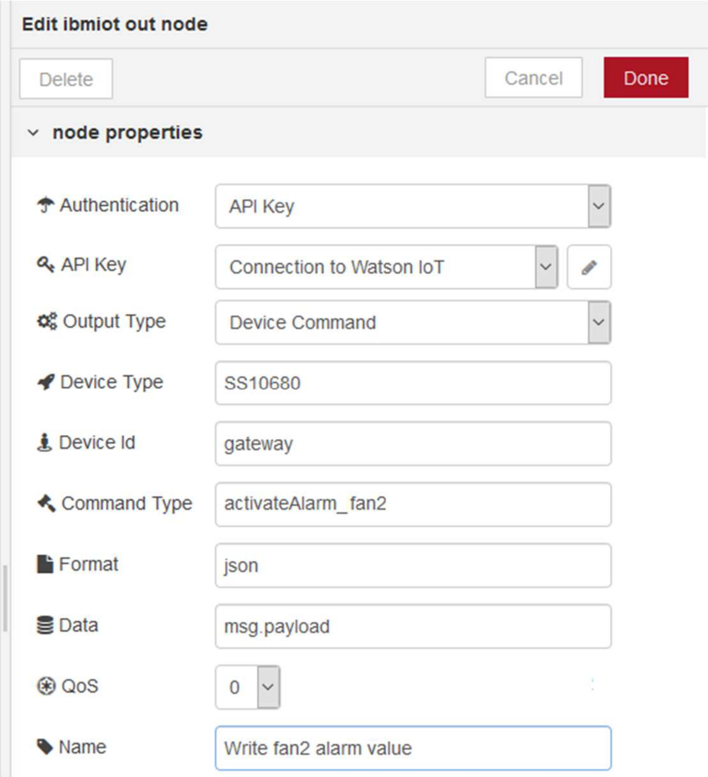
Off Payload: {} {"value":0}

Topic:

Name: activate fan2 alarm

To continue, click on Done button.

66) From ‘*output*’ menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.
Double click on it and edit it as shown below



Edit ibmiot out node

Delete | Cancel | Done

node properties

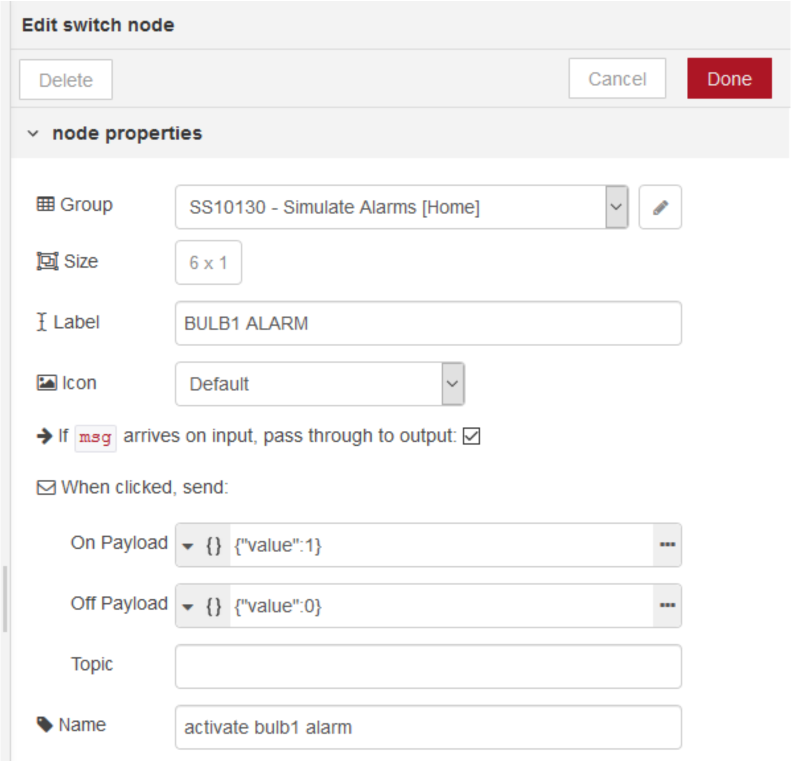
- Authentication: API Key
- API Key: Connection to Watson IoT
- Output Type: Device Command
- Device Type: SS10680
- Device Id: gateway
- Command Type: activateAlarm_fan2
- Format: json
- Data: msg.payload
- QoS: 0
- Name: Write fan2 alarm value

To continue, click on Done button.

67) Link the nodes added at the previous two points, as shown at point 61

68) From ‘*dashboard*’ menu in the left bar, select ***switch*** node switch and drag & drop it into the working area. Double click on it and edit it as described below

- Group: select ‘*SS10130 – Simulate Alarms*’
- Size: set ‘*6x1*’
- Name: enter ‘*BULB 1 ALARM*’
- Icon: select ‘*Default’*
- On payload: from menu select JSON and enter ‘*{“value”:1}*’
- Off payload: from menu select JSON and enter ‘*{“value”:0}*’
- Topic: leave blank
- Name: enter ‘*activate bulb1 alarm*’



Edit switch node

Delete | Cancel | Done

node properties

Group: SS10130 - Simulate Alarms [Home]

Size: 6 x 1

Label: BULB1 ALARM

Icon: Default

If msg arrives on input, pass through to output: ☑

When clicked, send:

On Payload: {} {"value":1}

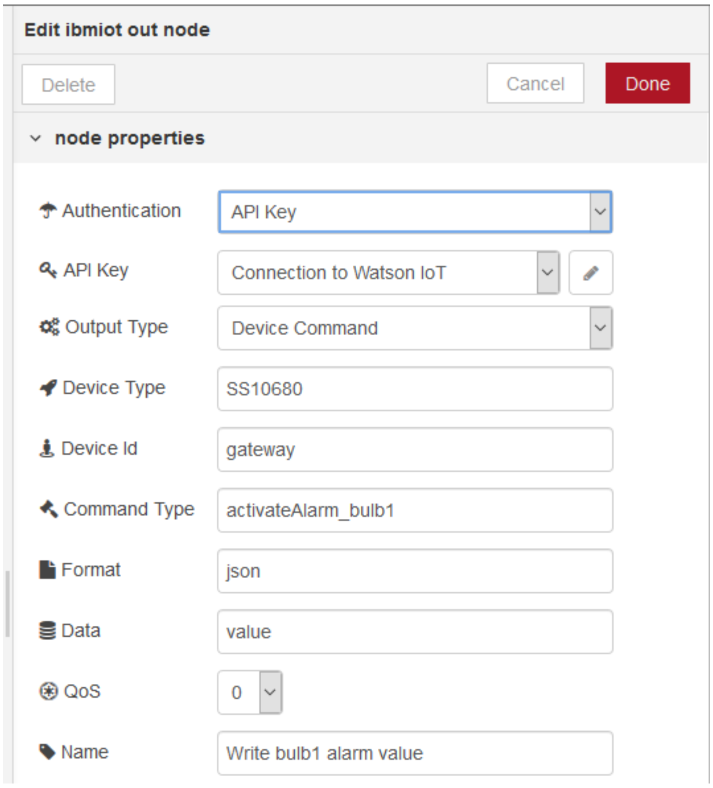
Off Payload: {} {"value":0}

Topic:

Name: activate bulb1 alarm

To continue, click on Done button.

69) From '*output*' menu in the left bar, select ***ibmiot*** node ibmiot and drag & drop it into the working area.
Double click on it and edit it as shown below



Edit ibmiot out node

Delete | Cancel | Done

node properties

Authentication: API Key

API Key: Connection to Watson IoT

Output Type: Device Command

Device Type: SS10680

Device Id: gateway

Command Type: activateAlarm_bulb1

Format: json

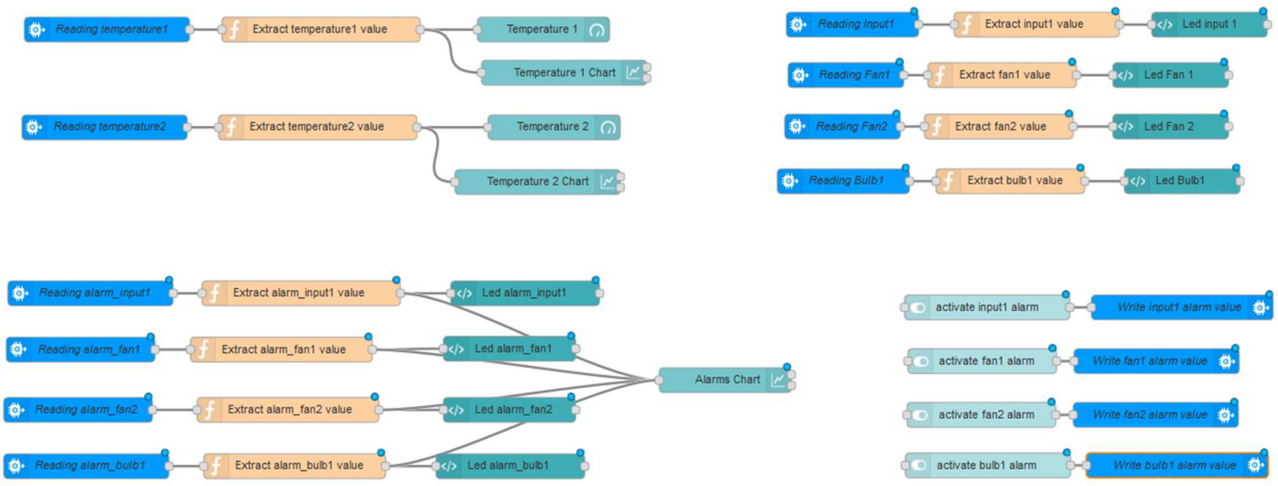
Data: value

QoS: 0

Name: Write bulb1 alarm value

To continue, click on Done button.

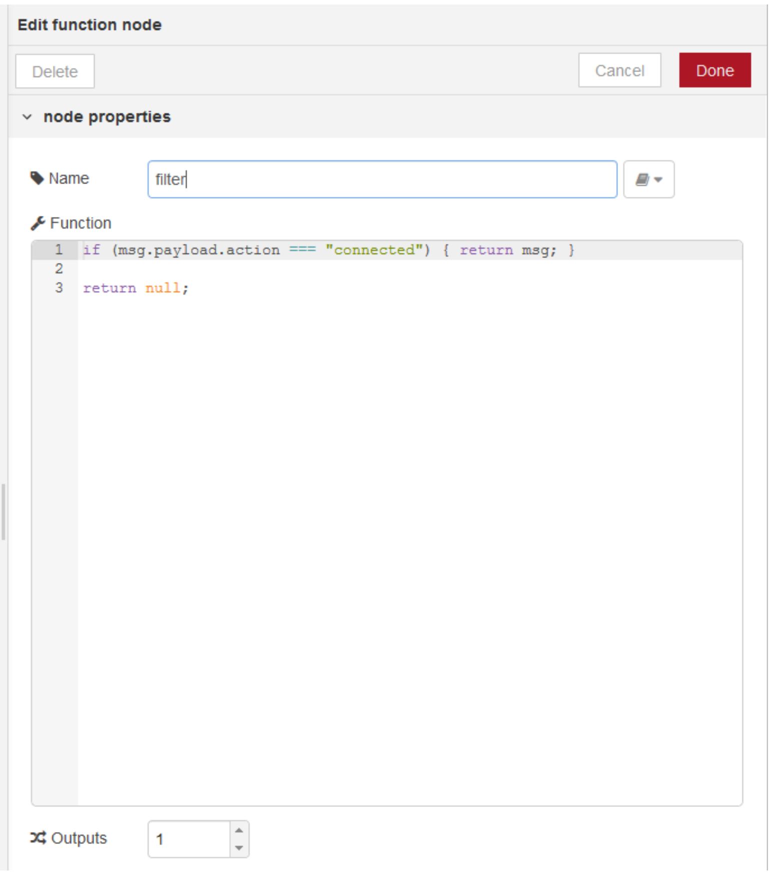
70) Link the nodes added at the previous two points, as shown at point 61
The result must be similar at the following one



71) Save the flow by clicking on Deploy button in the upper right corner of the window

72) From '*location*' menu in the left bar, select ***worlmap in*** node worldmap and drag & drop it into the working area.

73) From '*function*' menu in the left bar, select ***function*** node function and drag & drop it into the working area. Double click on it and edit it as shown below



Edit function node

Delete Cancel Done

node properties

Name: filter

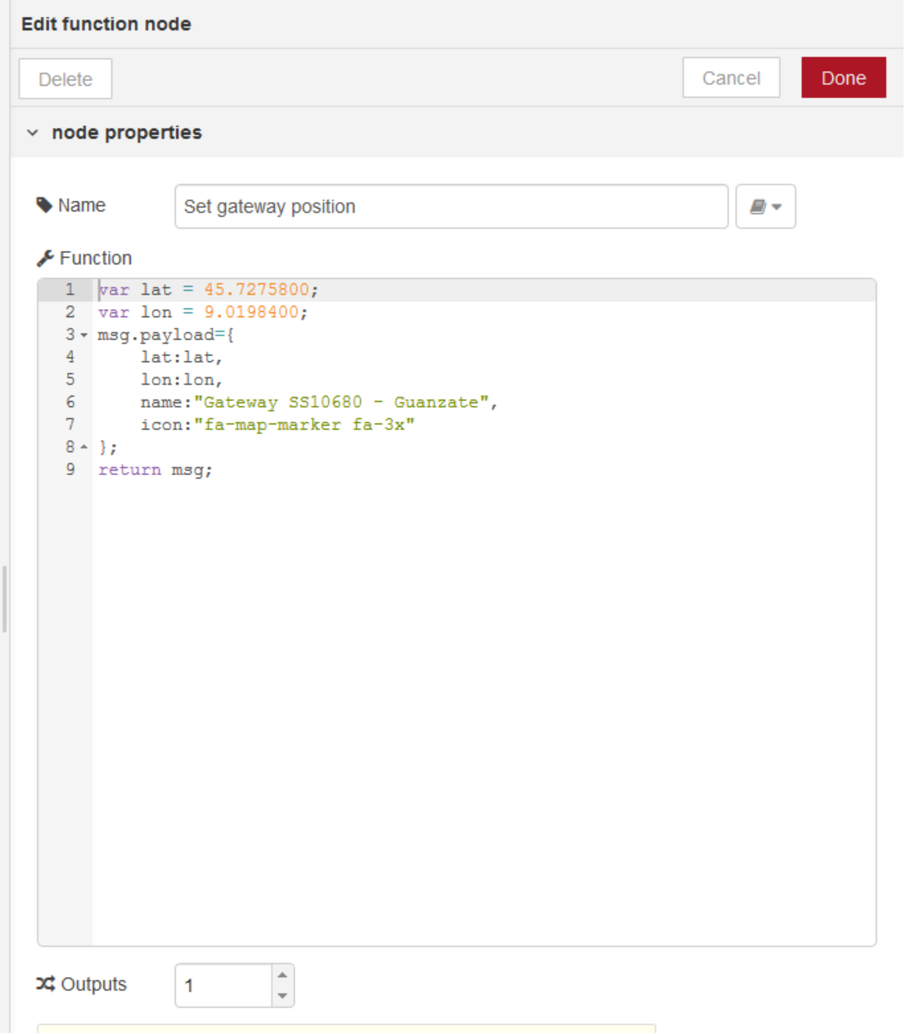
Function

```
if (msg.payload.action === "connected") { return msg; }

return null;
```

Outputs: 1

To continue, click on Done button.

74) From '*function*' menu in the left bar, select ***function*** node function and drag & drop it into the working area. Double click on it and edit it as shown below



Edit function node

Delete | Cancel | Done

node properties

Name: Set gateway position

Function

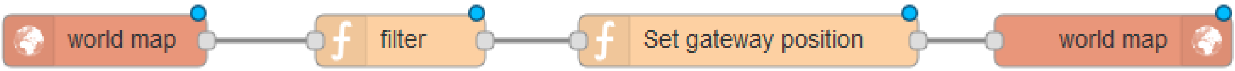
```
var lat = 45.7275800;
var lon = 9.0198400;
msg.payload={
    lat:lat,
    lon:lon,
    name:"Gateway SS10680 - Guanzate",
    icon:"fa-map-marker fa-3x"
};
return msg;
```

Outputs: 1

To continue, click on Done button.

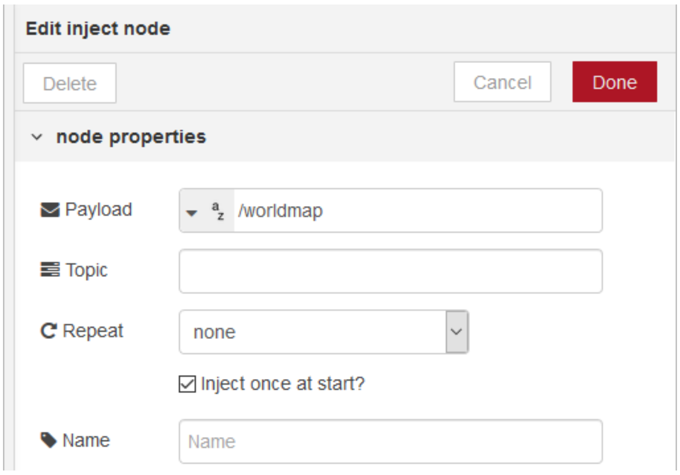
75) From '*location*' menu in the left bar, select ***worlmap out*** node worldmap and drag & drop it into the working area.

76) Link the previous four nodes added as shown in the following picture



77) From '*input*' menu in the left bar, select ***inject*** node inject and drag & drop it into the working area. Double click on it and edit it as described below:

- Payload: from menu select '*string*' and in the text field enter '*/worldmap*'
- Topic: leave blank
- Repeat: select '*None*'
- Check '*Inject once a start?*' checkbox


Edit inject node

Payload: /worldmap

Topic:

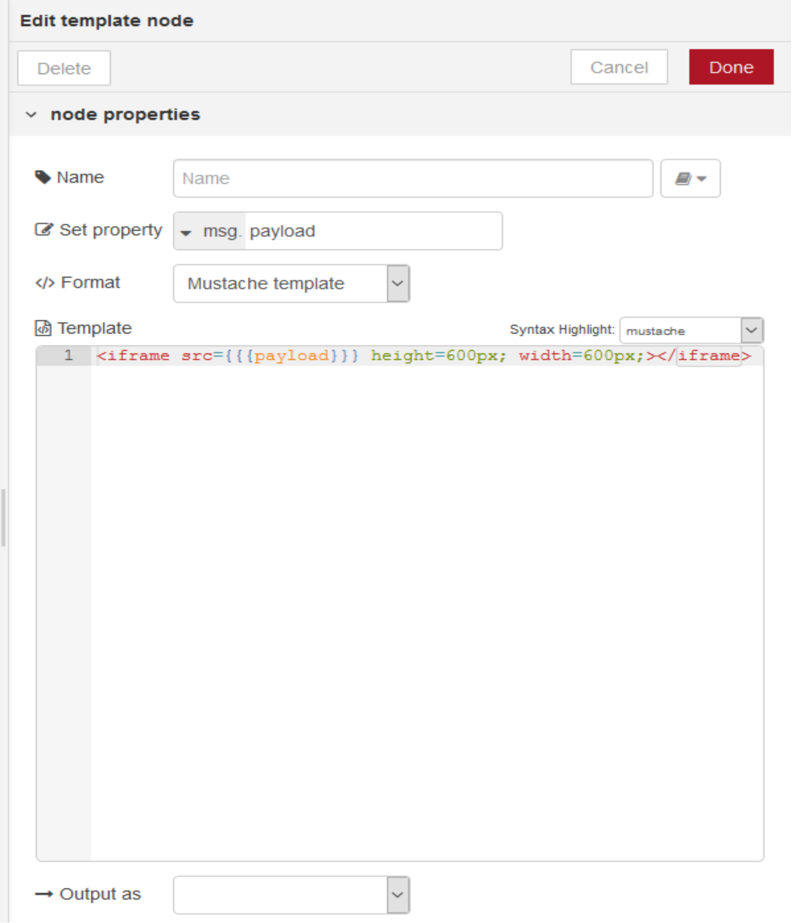
Repeat: none

☑ Inject once at start?

Name: Name

To continue, click on Done button.

78) From ‘*function*’ menu in the left bar, select ***template*** node { template and drag & drop it into the working area. Double click on it and edit it as show below


Edit template node

Name: Name

Set property: msg. payload

Format: Mustache template

Template

Syntax Highlight: mustache

```
<iframe src={{{payload}}} height=600px; width=600px;></iframe>
```

Output as:

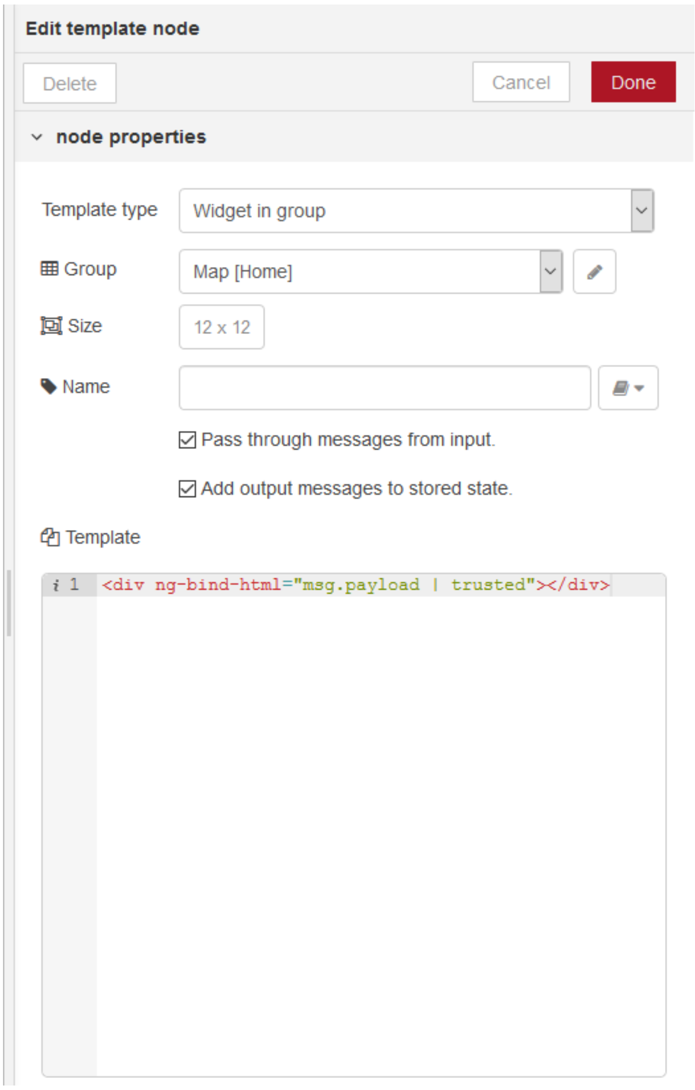
To continue, click on Done button.

79) From ‘*dashboard*’ menu in the left bar, select ***template*** node </> template and drag & drop it into the working area. Double click on it and edit it as described below

- Template type: select ‘*Widget in group*’
- Group: click on ✎ and add a new group ‘*Map*’
- Size: select ‘*12x12*’
- Check ‘*Pass through messages from input*’ checkbox
- Check ‘*Add output messages to stored state.*’ checkbox
- Template: enter the following code

```
<div ng-bind-html="msg.payload | trusted"></div>
```

At the end of the configuration, the result must be similar at the following one


Edit template node

Delete Cancel Done

node properties

Template type: Widget in group

Group: Map [Home]

Size: 12 x 12

Name:

☑ Pass through messages from input.

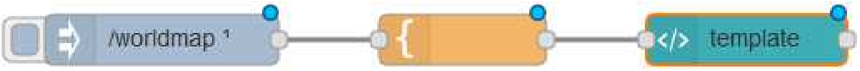
☑ Add output messages to stored state.

Template

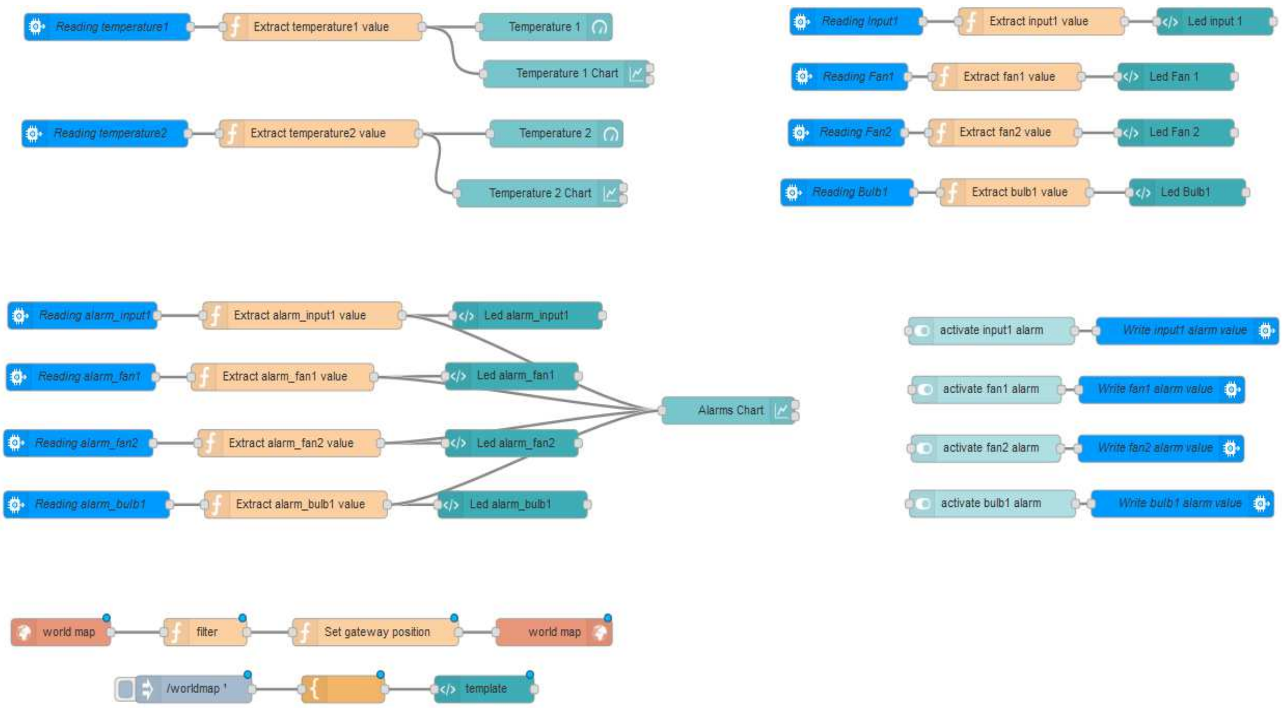
```
<div ng-bind-html="msg.payload | trusted"></div>
```

To continue, click on Done button.

80) Link the last three nodes as shown below

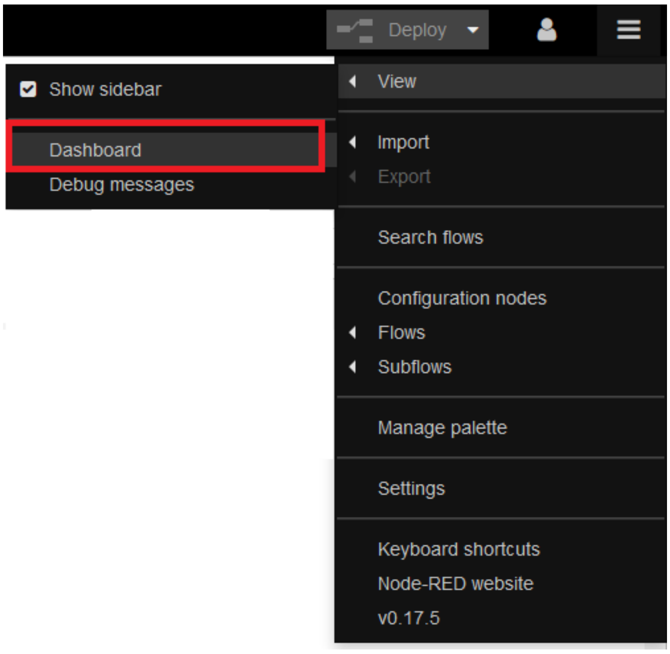


81) At the end, in the working area, there is a flow similar at the following one



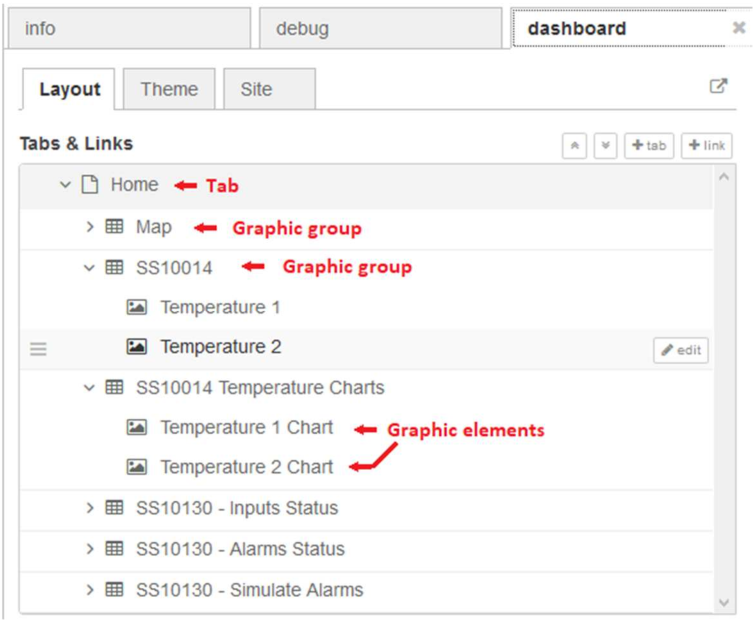
82) Save the flow by clicking on Deploy ▾ button in the upper right corner of the window

83) To have an idea on how the user interface is structured, click on ≡ in the upper right corner of the window, select '*View*' and then click on '*Dashboard*'



In the left bar, a new tab 'dashboard' opens and it has three sub-tabs:

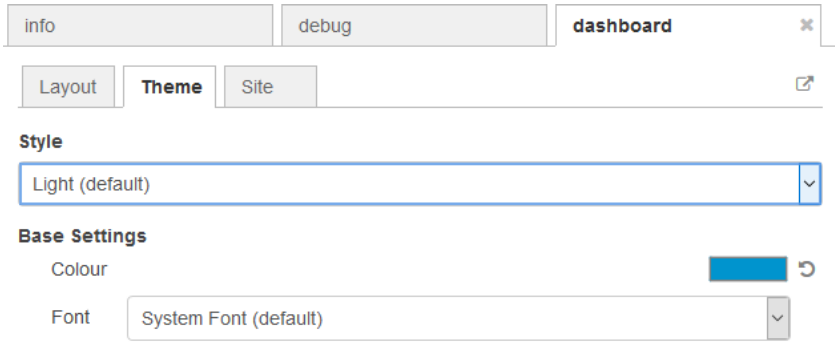
- Layout Tab: from this tab is it possible to see the structure of the user interface



Tabs & Links

- Home ← Tab
  - Map ← Graphic group
  - SS10014 ← Graphic group
    - Temperature 1
    - Temperature 2
  - SS10014 Temperature Charts
    - Temperature 1 Chart ← Graphic elements
    - Temperature 2 Chart
  - SS10130 - Inputs Status
  - SS10130 - Alarms Status
  - SS10130 - Simulate Alarms

- Themes Tab: from this tab is it possible to change the appearance of the user interface window



Style

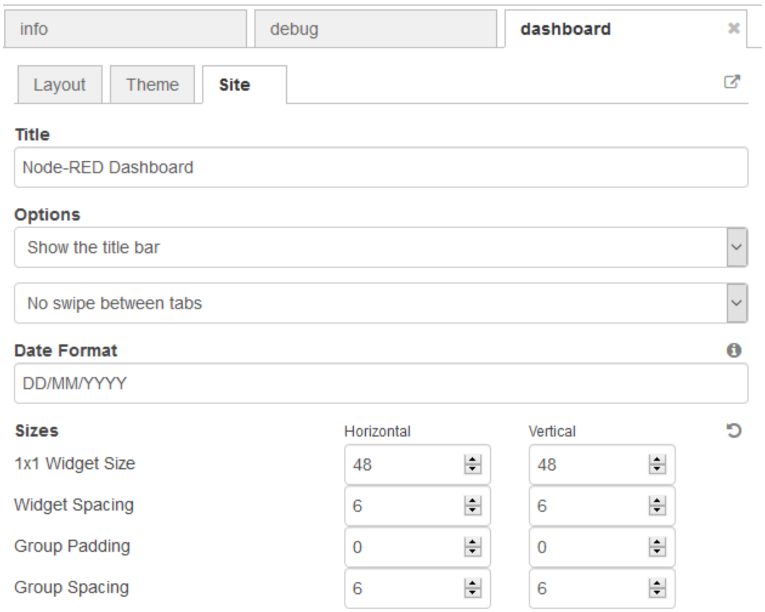
Light (default)

Base Settings

Colour 🟦

Font System Font (default)

- Site Tab: from this tab is it possible to change some properties of the site



Title

Node-RED Dashboard

Options

Show the title bar

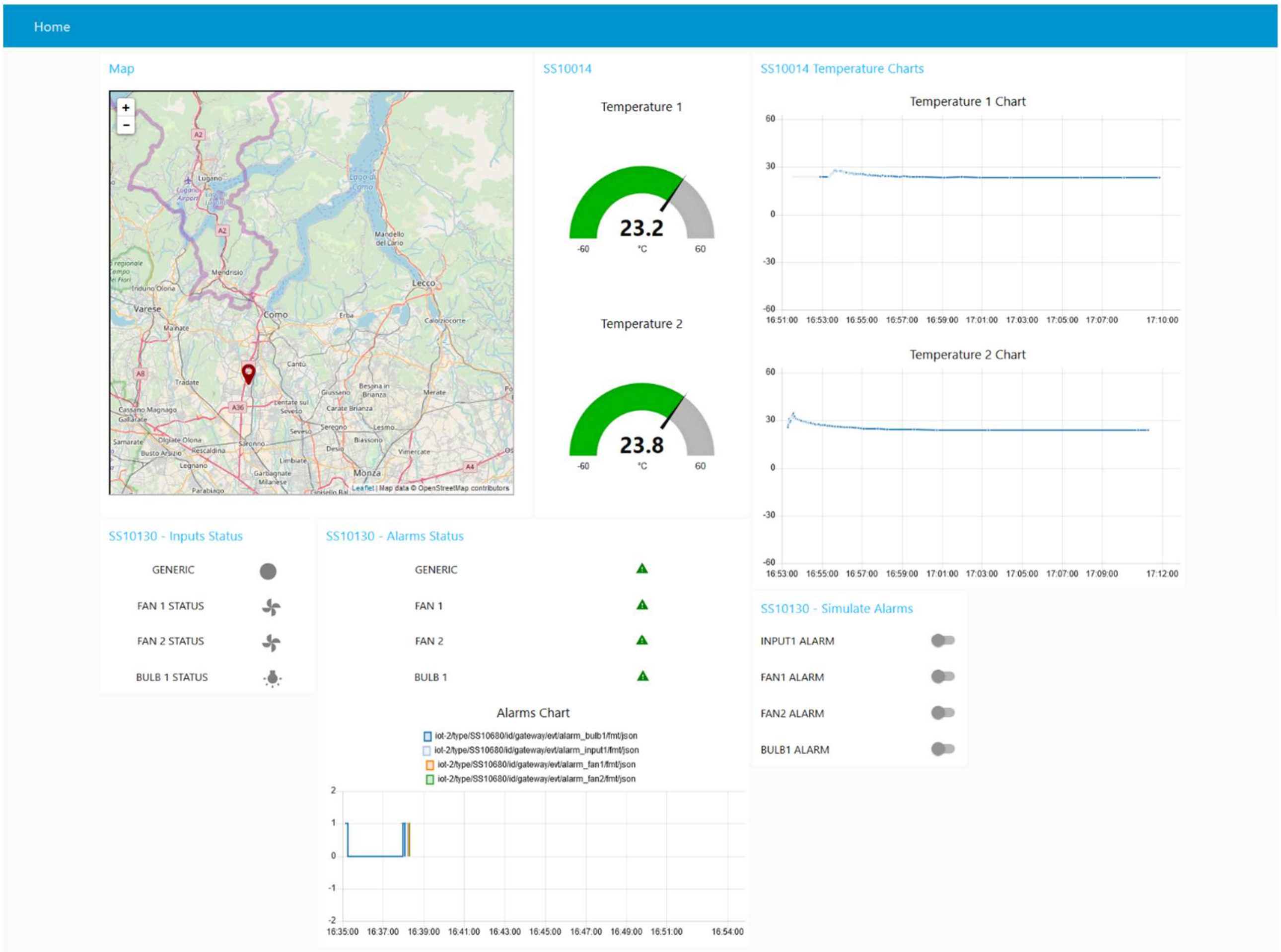
No swipe between tabs

Date Format

DD/MM/YYYY

Sizes	Horizontal	Vertical
1x1 Widget Size	48	48
Widget Spacing	6	6
Group Padding	0	0
Group Spacing	6	6

84) To see the user interface, you can click on ↗ button in dashboard tab, or enter in the address bar of the browser the route of your application assigned by Watson IoT followed by '*/ui*'.
The user interface realized for this example appears like the following one


Home
Map
+
-
Lugano
Como
Mendrisio
Varese
Lecco
Saronno
Monza
Leaflet | Map data © OpenStreetMap contributors
SS10014
Temperature 1
23.2
-60
°C
60
Temperature 2
23.8
-60
°C
60
SS10014 Temperature Charts
Temperature 1 Chart
60
30
0
-30
-60
16:51:00
16:53:00
16:55:00
16:57:00
16:59:00
17:01:00
17:03:00
17:05:00
17:07:00
17:10:00
Temperature 2 Chart
60
30
0
-30
-60
16:53:00
16:55:00
16:57:00
16:59:00
17:01:00
17:03:00
17:05:00
17:07:00
17:09:00
17:12:00
SS10130 - Inputs Status
GENERIC
FAN 1 STATUS
FAN 2 STATUS
BULB 1 STATUS
SS10130 - Alarms Status
GENERIC
FAN 1
FAN 2
BULB 1
Alarms Chart
iot-2/type/SS10680/id/gateway/evt/alarm_bulb1/fmt/json
iot-2/type/SS10680/id/gateway/evt/alarm_input1/fmt/json
iot-2/type/SS10680/id/gateway/evt/alarm_fan1/fmt/json
iot-2/type/SS10680/id/gateway/evt/alarm_fan2/fmt/json
2
1
0
-1
-2
16:35:00
16:37:00
16:39:00
16:41:00
16:43:00
16:45:00
16:47:00
16:49:00
16:51:00
16:54:00
SS10130 - Simulate Alarms
INPUT1 ALARM
FAN1 ALARM
FAN2 ALARM
BULB1 ALARM