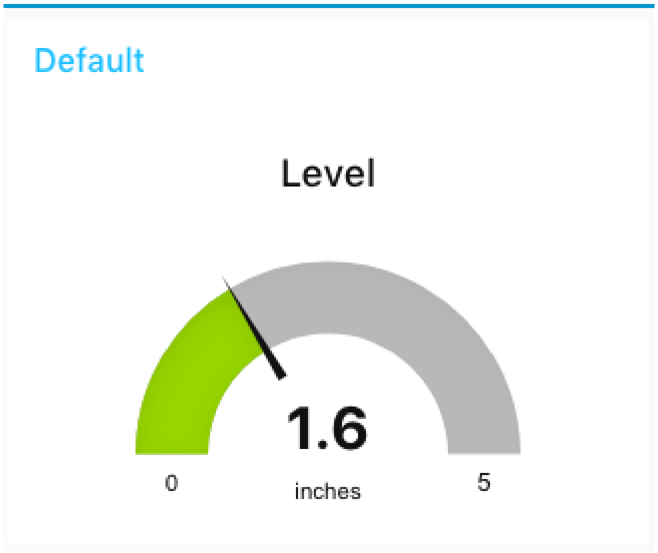
VISUALIZATION AND ANALYTICS WITH RULES ENGINE

With all the data coming from IoT devices, it's really difficult to consume without some aspect of visualization. Data visualization can help your team or your customers glean insight from connected devices in the field.

In this tutorial, we'll be creating dashboards, a handy way to visualize information using the Rules Engine.

First Dashboard

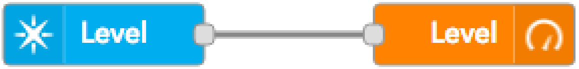
We'll make a simple gauge to show the current level of our level sensor from the first tutorial as our first dashboard.



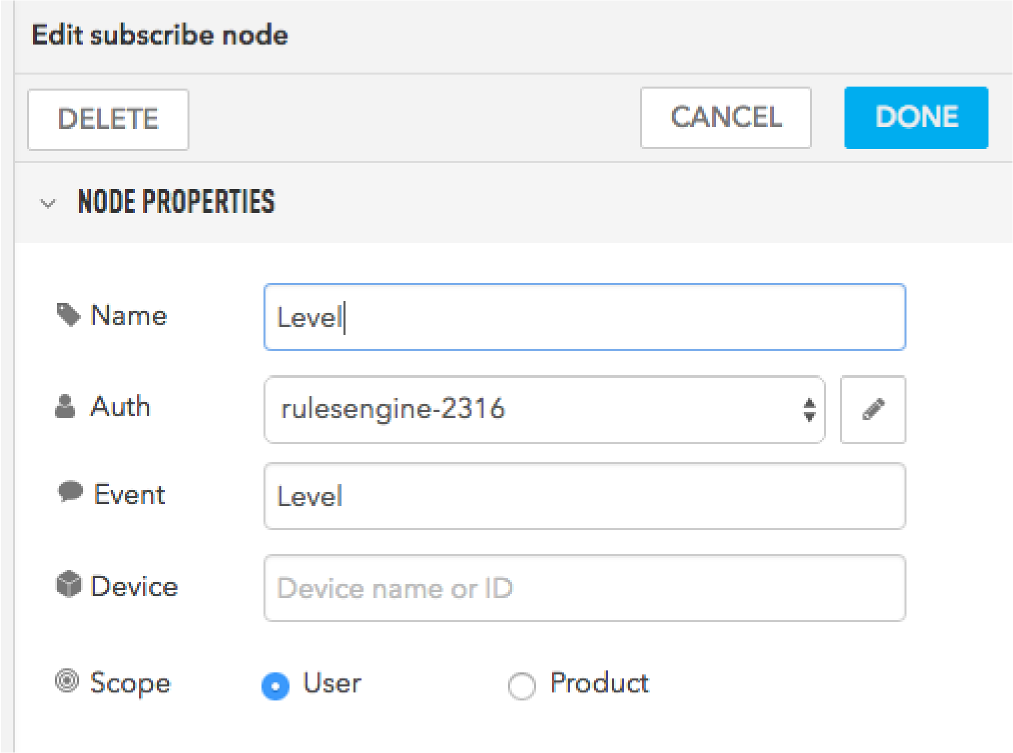
Drag the Copy Rules button into the Rules Engine window to create the flow automatically, or you can create the flow from scratch with the steps below.

Copy Rules ▶

This is the flow we'll be building:



- From the **Particle** group, drag **subscribe** node to your flow.
- Double click to edit it.
- Set **Name** to **Level** (can be anything).
- Set **Auth** to the authentication we created in the real-time alerting tutorial.
- Set **Event** to **Level**.
- Leave the **Device** field empty
- Leave the **Scope** at **User**.
- Click **Done**.



Edit subscribe node

Name: Level

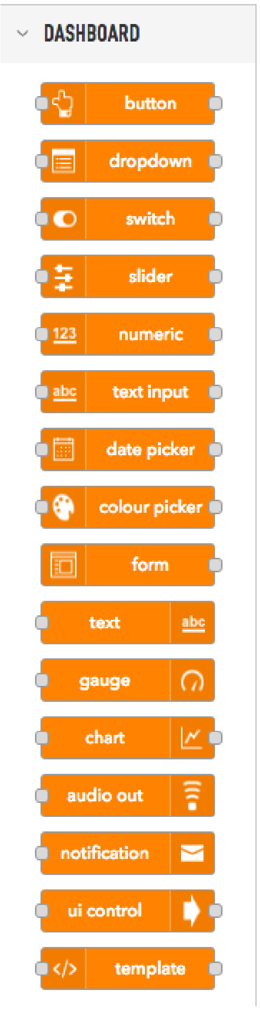
Auth: rulesengine-2316

Event: Level

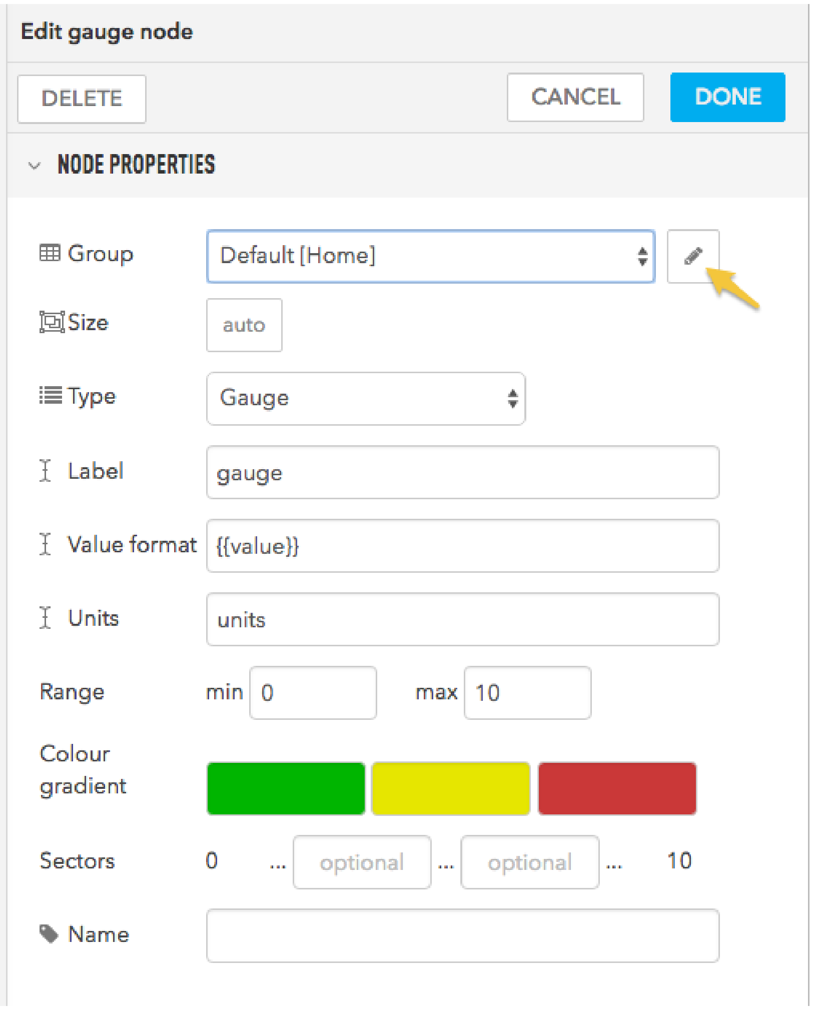
Device: Device name or ID

Scope: ● User ○ Product

- Find the **Dashboard** section of the palette.
- Drag a **gauge** output to a new flow.



- Double click the **gauge** node to edit it.
- Click the pencil icon to edit the dashboard group node.



Edit gauge node

Group: Default [Home]

Size: auto

Type: Gauge

Label: gauge

Value format: {{value}}

Units: units

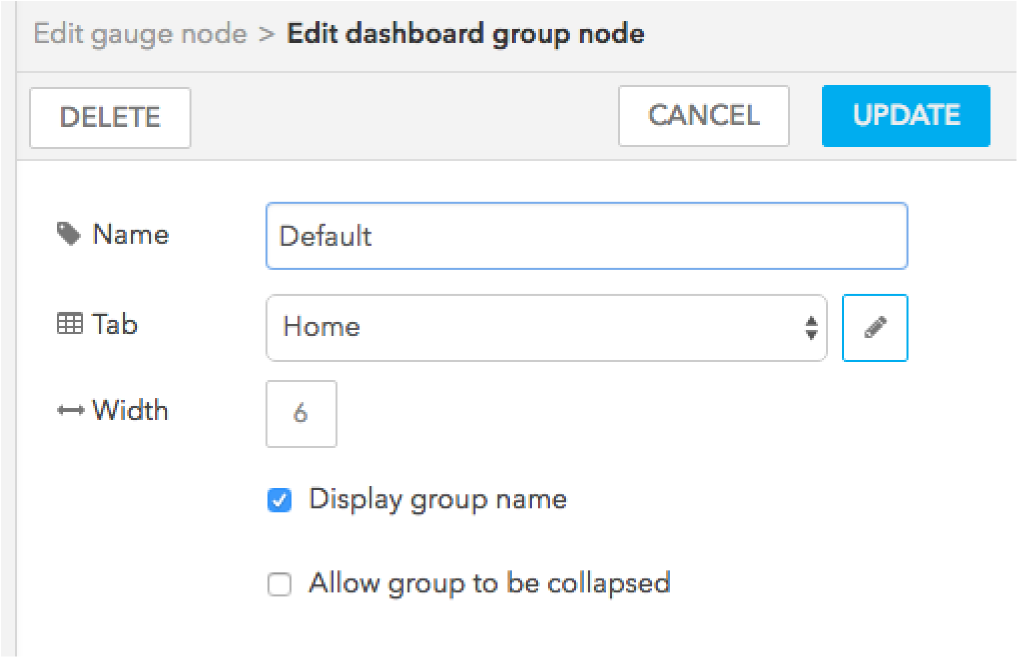
Range: min 0 max 10

Colour gradient: 🟩 🟨 🟥

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

Name:

- You don't have anything you really need to change in the dashboard group node, but this is what it looks like.



Name: Default

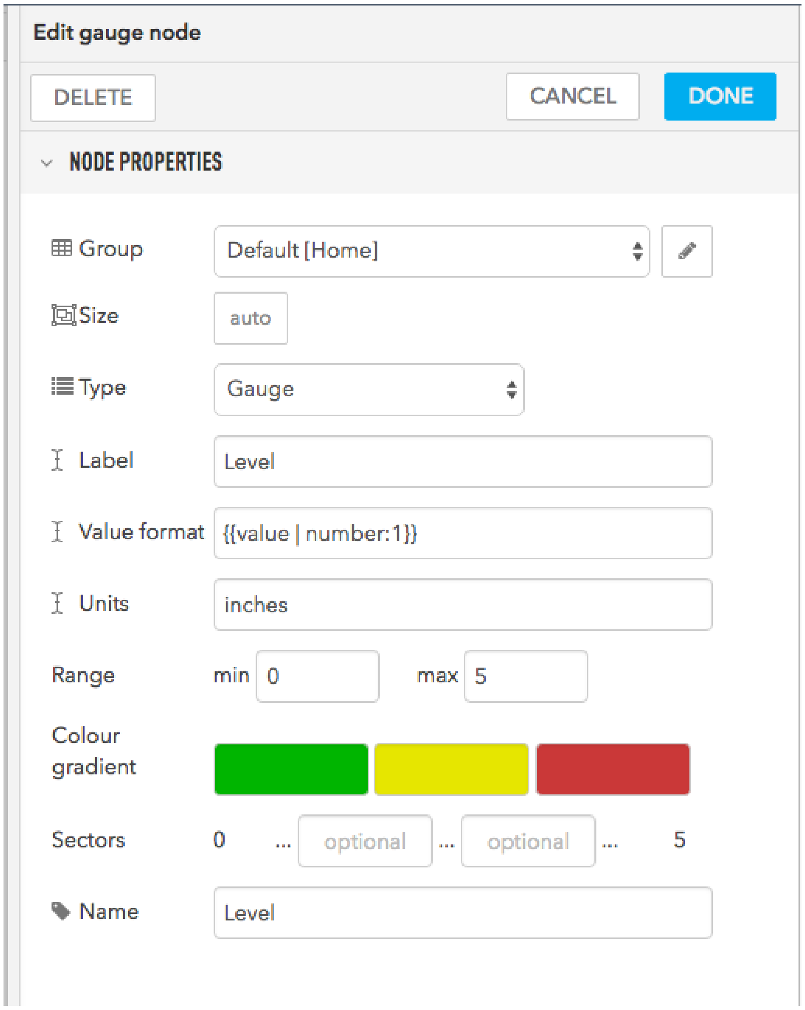
Tab: Home

Width: 6

☑ Display group name

□ Allow group to be collapsed

- Back in the **edit gauge node** window:
- Set the **Label** to **Level** (can be anything).
- Set the **Value format** to **{{value | number:1}}**. This displays the value, as a number, with 1 decimal point.
- Set the **Units** to **inches**.
- Set the **Range** to 0 - 5.
- Set the **Name** to **Level** (can be anything).



Group: Default [Home]

Size: auto

Type: Gauge

Label: Level

Value format: {{value | number:1}}

Units: inches

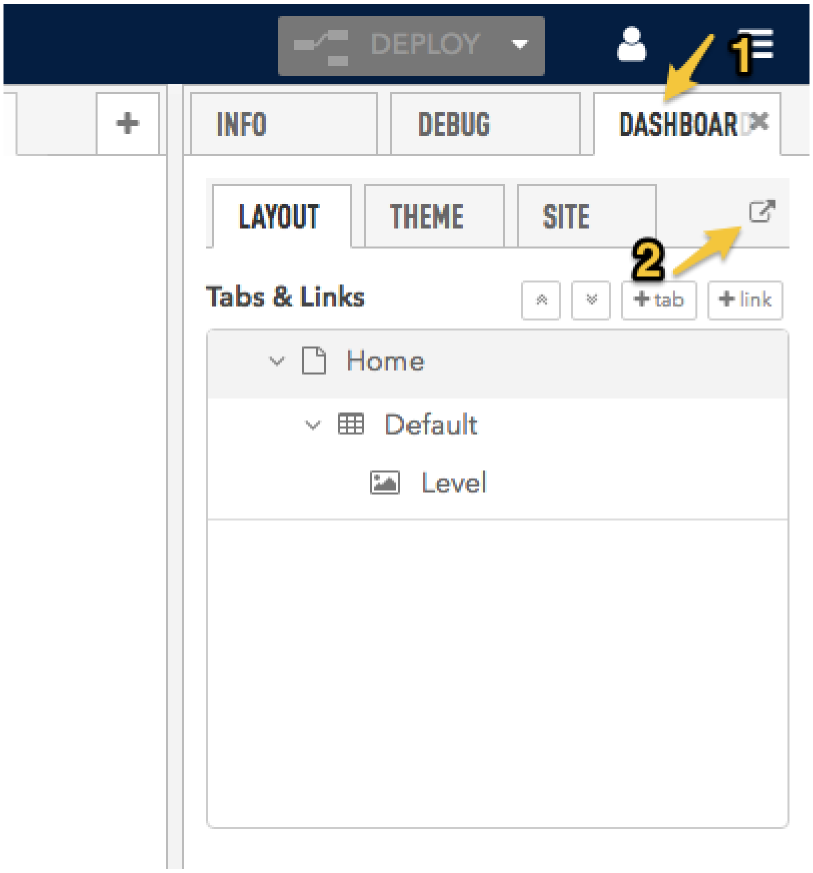
Range: min 0 max 5

Colour gradient: 🟩 🟨 🟥

Sectors: 0 ... optional ... optional ... 5

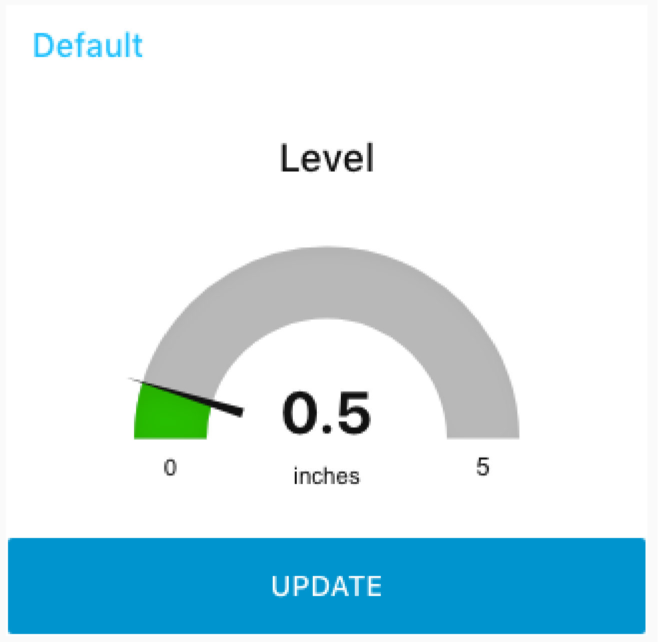
Name: Level

- Connect the nodes.
- Deploy your flow.
- In the upper right corner of the window, select the **Dashboard** tab (1).
- Then click the small **open dashboard** icon (2) under it. This will open a new browser tab or window with the dashboard.

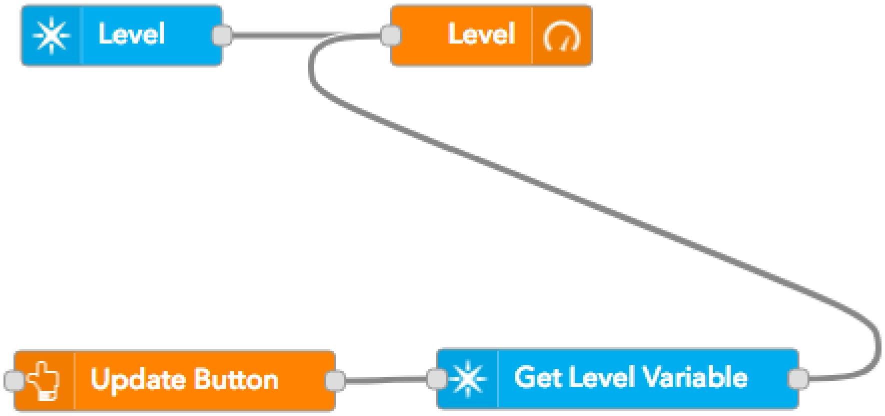


Querying the value

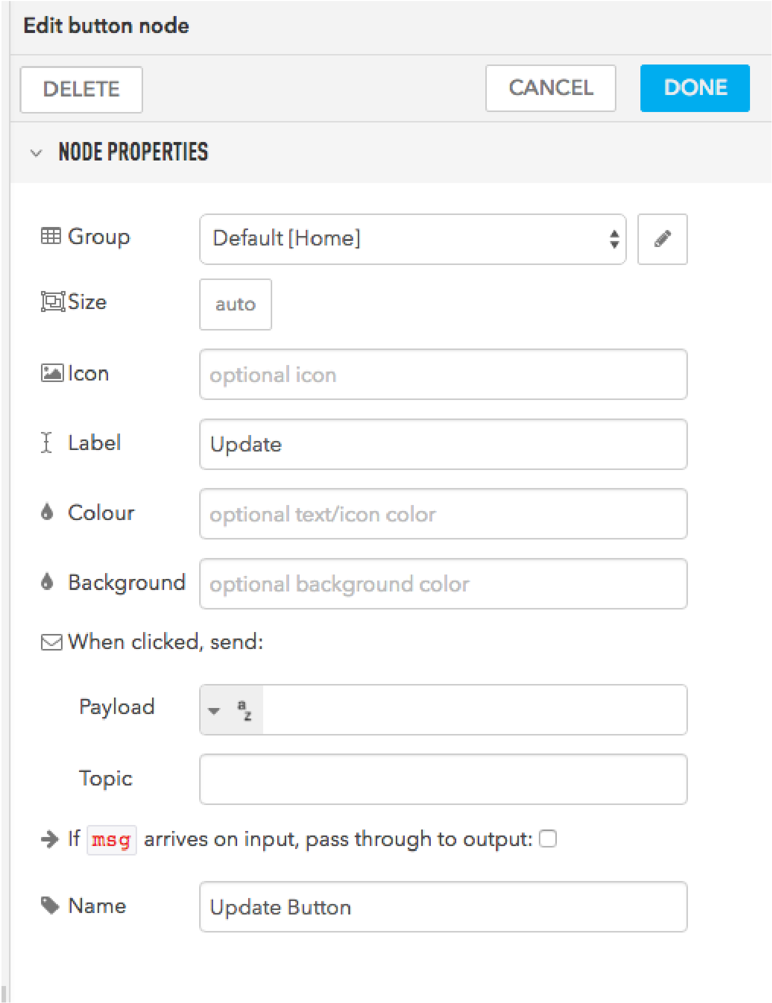
The way our firmware is designed, it only updates the level every minute, or when an alarm occurs or clears. What if you want to get the value right now?



Adding onto the previous flow:



- Find the **Dashboard** section of the palette.
- Drag a **button** output to a new flow.
- Double click the **button** node to edit it.
- Set the **Label** to **Update** (this is what the title of the button is when displayed in the dashboard).
- Set the **Name** to **Update Button** (can be anything).



Group: Default [Home]

Size: auto

Icon: optional icon

Label: Update

Colour: optional text/icon color

Background: optional background color

When clicked, send:

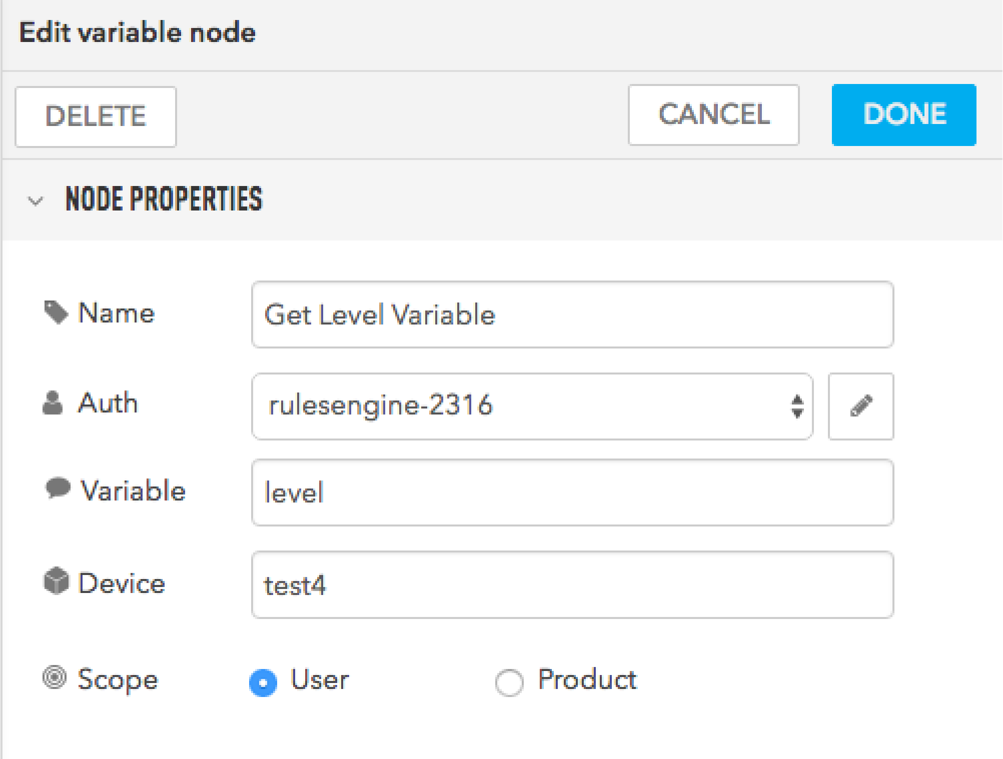
Payload:

Topic:

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

Name: Update Button

- From the **Particle** section of the palette, drag a **variable** to your flow.
- Double click the **variable** node to edit it.
- Set the **Name** to **Get Level Variable** (can be anything).
- Set the **Auth** to the authentication we set up earlier.
- Set the **Variable** to **level** (case-sensitive).
- Set the **Device** to the device name of your Photon running the level sensor.
- Leave the **Scope** at **User**.



Name: Get Level Variable

Auth: rulesengine-2316

Variable: level

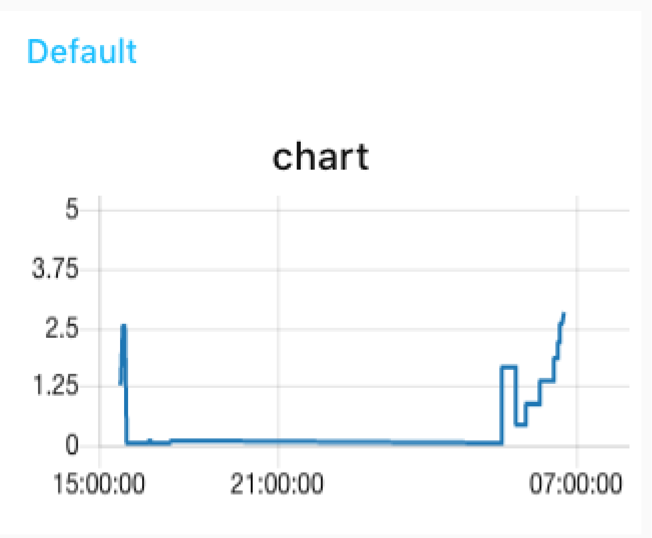
Device: test4

Scope: ● User ○ Product

- Connect up the nodes into a flow.
- Note that both the output from the **Get Level Variable** and subscribe node to into the same gauge node.
- Deploy your flow.
- View the dashboard page.
- Now, when you click the **Update** button, the gauge is updated right away.

Charting from a database

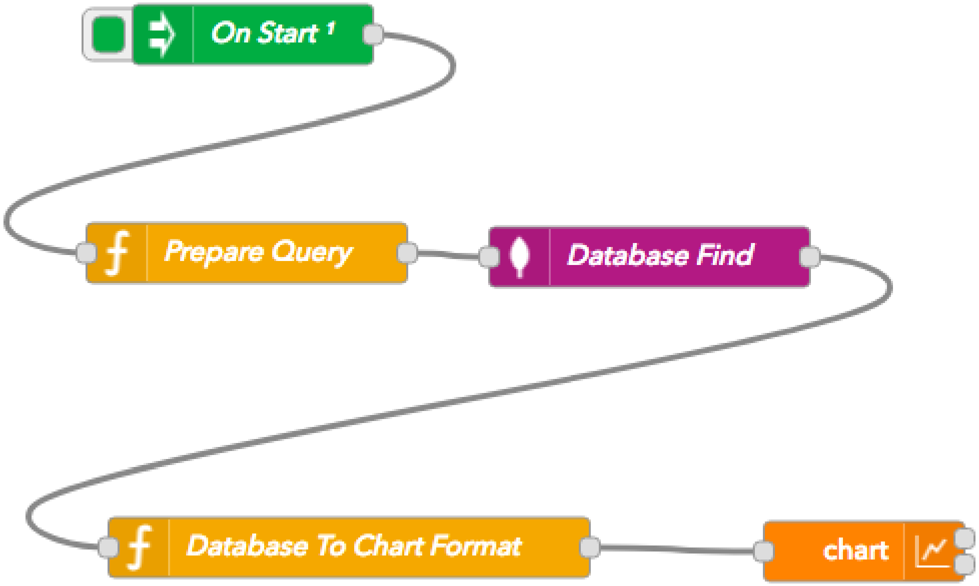
This is a much more complicated example that charts the level over time.



It's based on the earlier MongoDB example so it can retrieve the previous values from the database. There are more nodes in the example than previous examples!

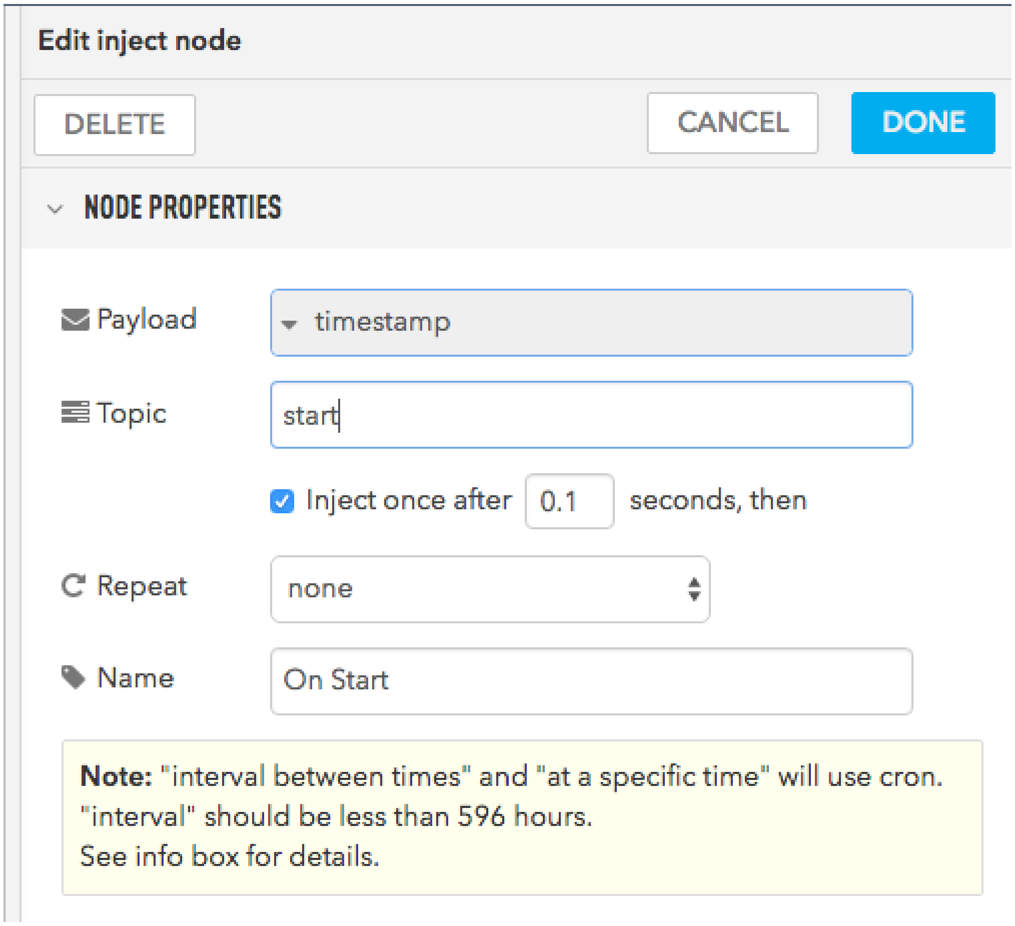
Drag the Copy Rules button into the Rules Engine window to create the flow automatically, or you can create the flow from scratch with the steps below.

Copy Rules ▶



- From the **Input** section of the palette, drag an **Inject** node into a new flow.
- The payload and topic are mostly ignored.

- Make sure **Inject once after 0.1 seconds, then** is checked.
- Make sure **Repeat** is **none**.
- Set **Name** to **On Start** (or anything else).
- This construct is used to run a flow once when deployed.



Edit inject node

Payload: timestamp

Topic: start

☑ Inject once after 0.1 seconds, then

Repeat: none

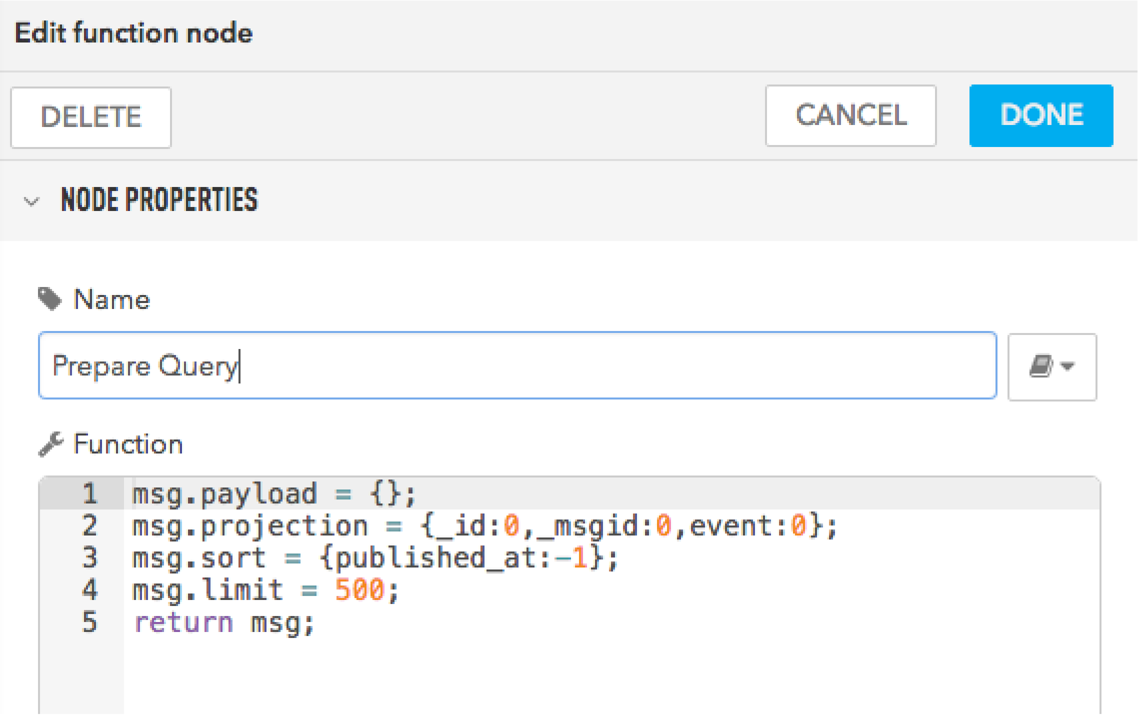
Name: On Start

Note: "interval between times" and "at a specific time" will use cron. "interval" should be less than 596 hours.
See info box for details.

- From the **function** group group drag a **function** to your flow. Note: Not a Particle function.
- Set the **Name** to **Prepare Query**.
- This node sets the parameters for querying MongoDB.
- Set the **Function** to:

```
msg.payload = {};
msg.projection = {_id:0,_msgid:0,event:0};
msg.sort = {published_at:-1};
msg.limit = 500;
return msg;
```

This searches all documents (payload), omitting several fields (projection), sorts descending, and limits to the most recent 500 documents.

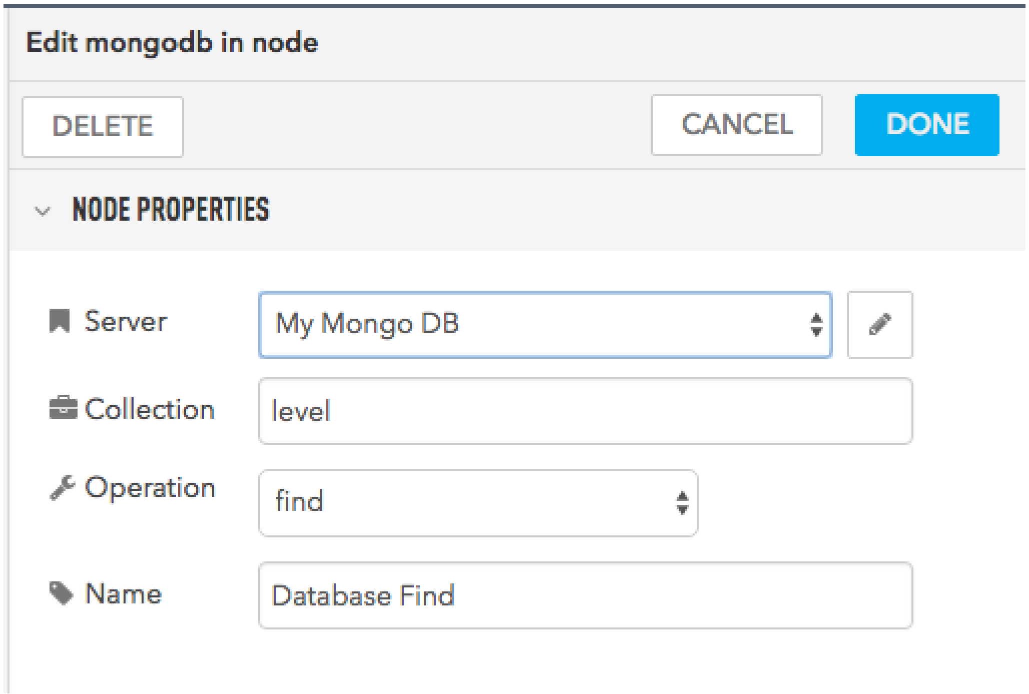


Name: Prepare Query

Function

```
msg.payload = {};
msg.projection = {_id:0,_msgid:0,event:0};
msg.sort = {published_at:-1};
msg.limit = 500;
return msg;
```

- From the **Storage** section of the palette, drag a **mongodb** (in) node to your flow.
- Use the previously configured **Server**.
- Set the **Collection** to **level**.
- Set the **Operation** to **find**.
- Set the **Name** to **Database Find** (or anything else).



Edit mongodb in node

Server: My Mongo DB

Collection: level

Operation: find

Name: Database Find

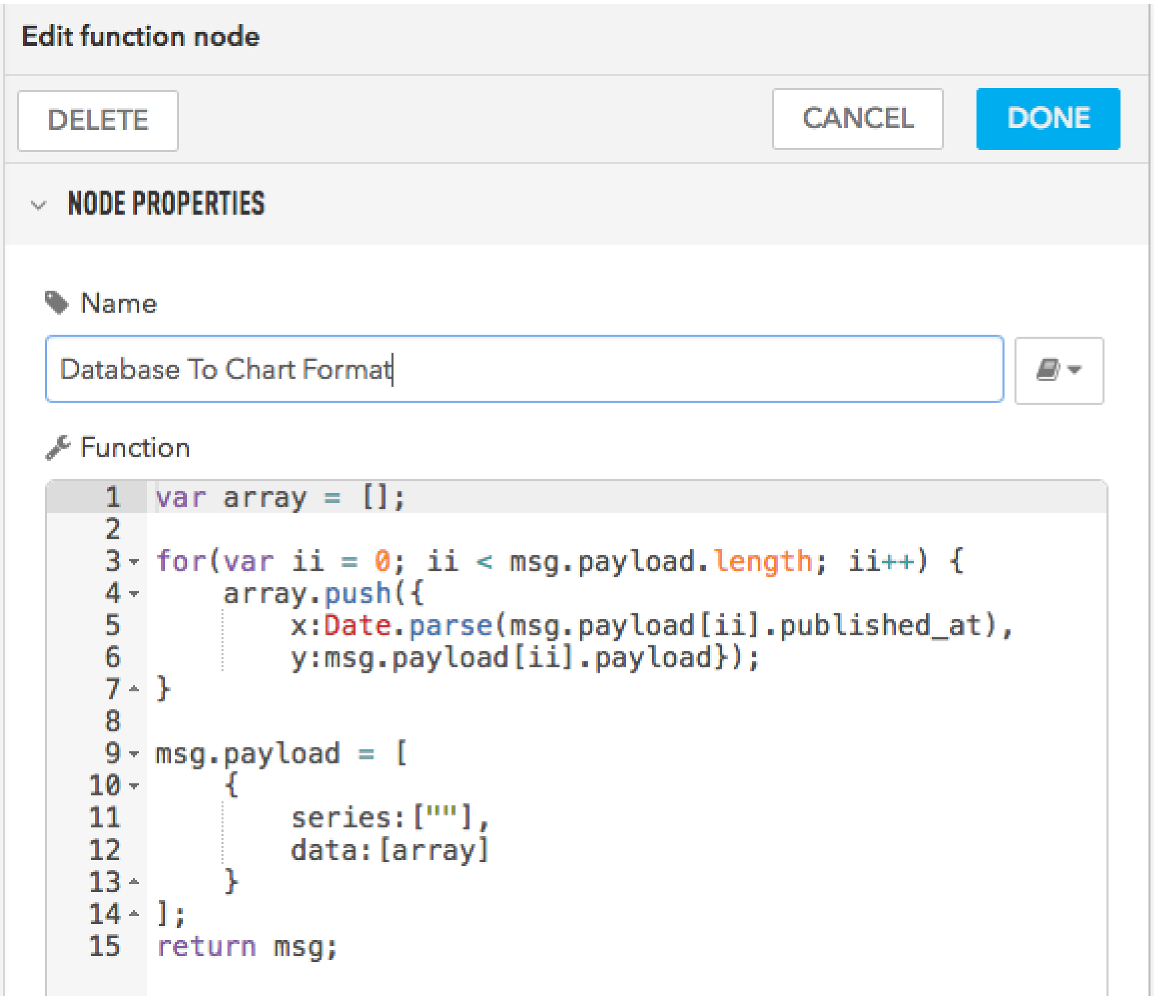
One problem is that the output from the database node is in database format, and we need to convert it into something the chart node can consume. This will require a function node.

- From the **function** group group drag a **function** to your flow. Note: Not a Particle function.
- Set the **Name** to **Database to Chart Format**.
- Set the **Function** to:

```
var array = [];

for(var ii = 0; ii < msg.payload.length; ii++) {
    array.push({
        x:Date.parse(msg.payload[ii].published_at),
        y:msg.payload[ii].payload});
}

msg.payload = [
    {
        series:[""],
        data:[array]
    }
];
return msg;
```



Name: Database To Chart Format

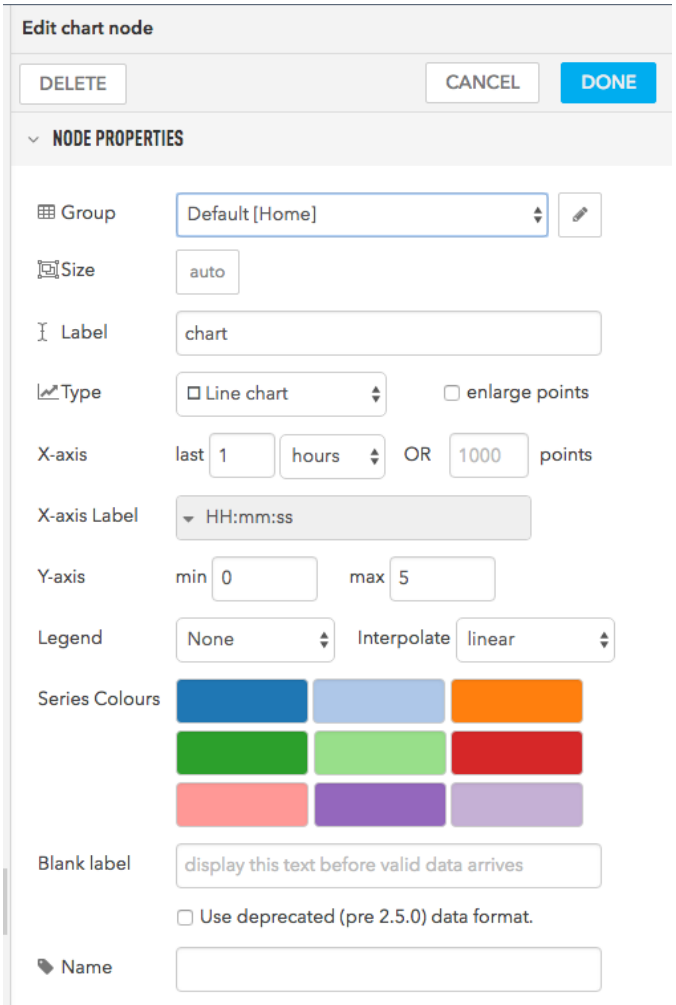
Function

```
var array = [];

for(var ii = 0; ii < msg.payload.length; ii++) {
    array.push({
        x:Date.parse(msg.payload[ii].published_at),
        y:msg.payload[ii].payload});
}

msg.payload = [
    {
        series:[""],
        data:[array]
    }
];
return msg;
```

- From the **dashboard** section of the palette, drag a chart into your flow.
- Change the **Y-axis** to 0 - 5.
- You can leave the rest of the settings as the defaults.



Group: Default [Home]

Size: auto

Label: chart

Type: Line chart ☐ enlarge points

X-axis last 1 hours OR 1000 points

X-axis Label HH:mm:ss

Y-axis min 0 max 5

Legend: None Interpolate: linear

Series Colours

Blank label: display this text before valid data arrives

☐ Use deprecated (pre 2.5.0) data format.

Name:

- Connect up the nodes in the flow as pictured above.
- Deploy.
- View the dashboard!