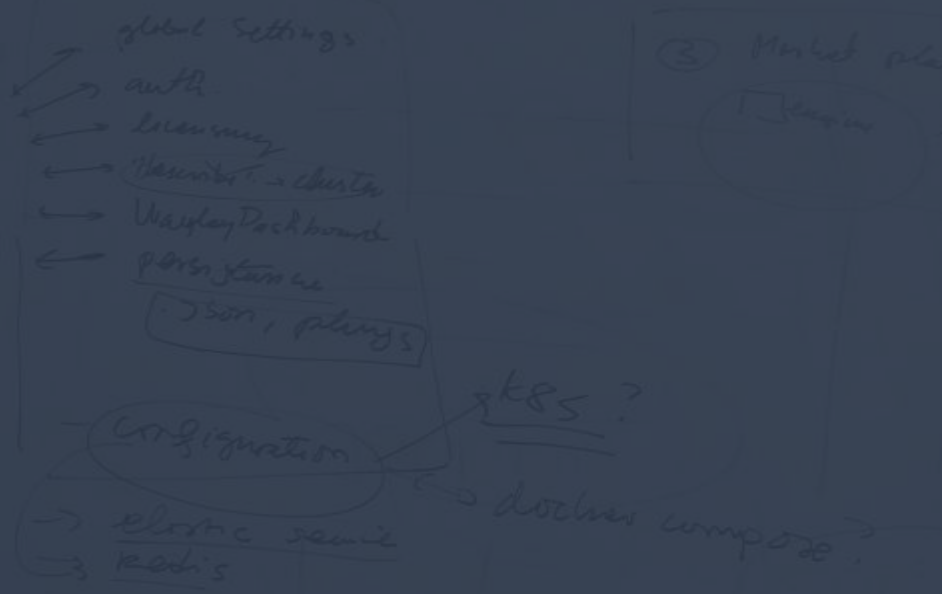


Waylay Newbie Exercises

2020-05



Exercise 1

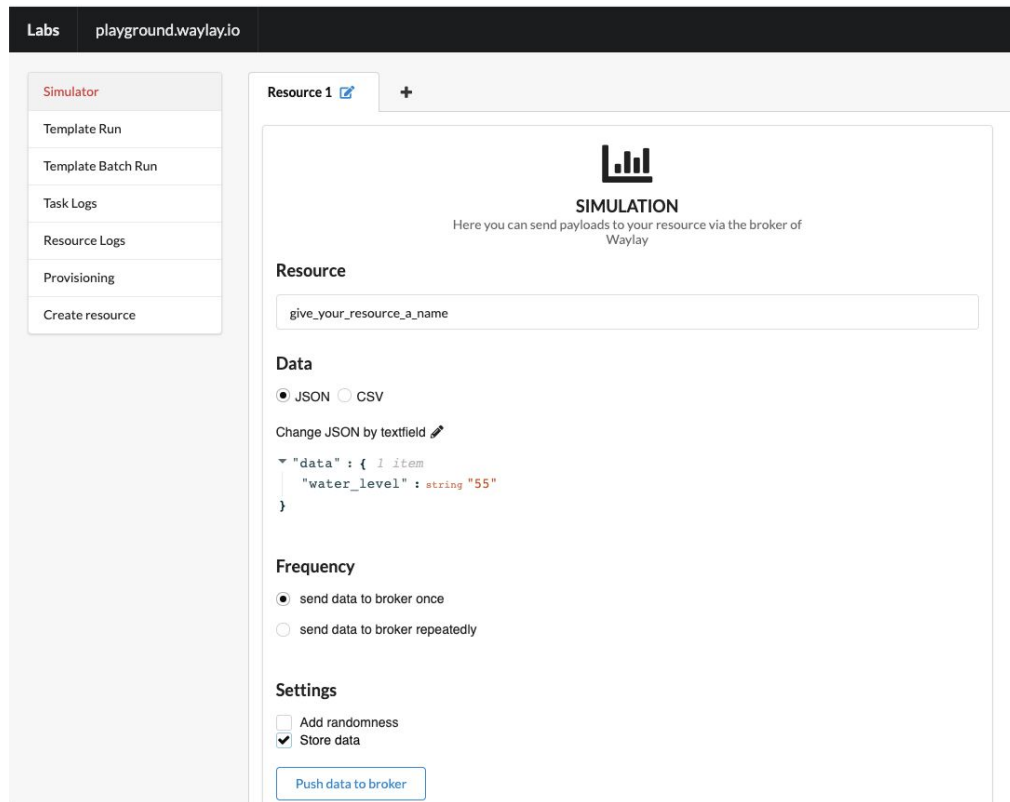


Exercise 1 - Simple streaming data rule

- We are going to build a simple streaming analytics rule step by step
- We will use 'Labs' to simulate a physical device
- Use case:
 - Create an alarm in Waylay when the water level is above a configurable threshold

Exercise 1 - Step 1

- Go to 'Labs' and shoot some data in Waylay
- This will automatically create a 'resource' in Waylay



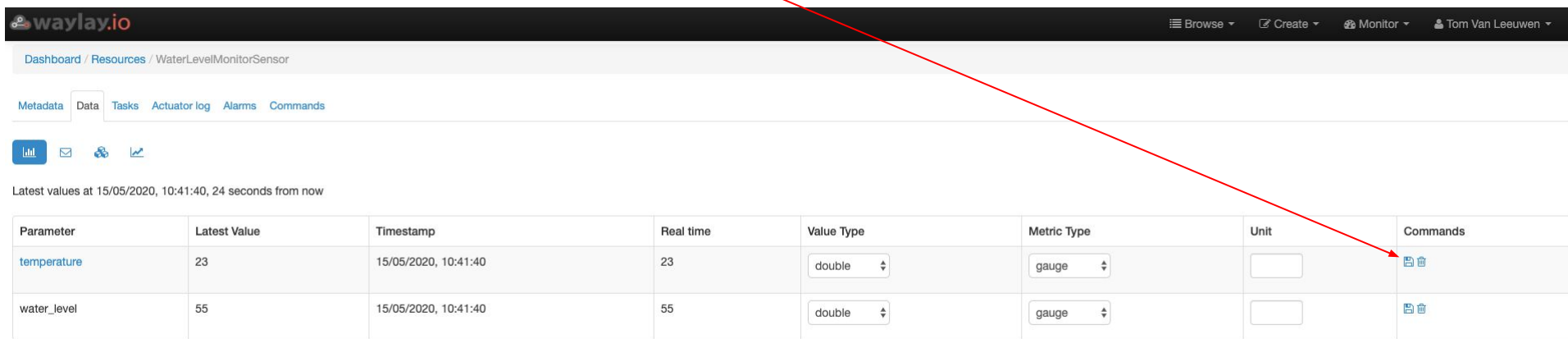
The screenshot shows the Waylay Playground interface. At the top, there is a navigation bar with 'Labs' and 'playground.waylay.io'. On the left, a 'Simulator' menu is visible with options: 'Template Run', 'Template Batch Run', 'Task Logs', 'Resource Logs', 'Provisioning', and 'Create resource'. The main area displays the 'Resource 1' configuration page. It features a 'SIMULATION' header with a bar chart icon and the text 'Here you can send payloads to your resource via the broker of Waylay'. Below this, there is a 'Resource' section with a text input field containing 'give_your_resource_a_name'. The 'Data' section has radio buttons for 'JSON' (selected) and 'CSV', and a 'Change JSON by textfield' link. A JSON payload is shown:

```
{ "data": { "item": { "water_level": "string \"55\""} } }
```

. The 'Frequency' section has radio buttons for 'send data to broker once' (selected) and 'send data to broker repeatedly'. The 'Settings' section has checkboxes for 'Add randomness' (unchecked) and 'Store data' (checked). A 'Push data to broker' button is at the bottom.

Exercise 1 - Step 2

- Go to Resources and find your resource
- Select the 'data' tab and click the disk icon next to each variable (=metric). This will store the metric name in the resource meta data







waylay.io

Browse Create Monitor Tom Van Leeuwen

Dashboard / Resources / WaterLevelMonitorSensor

Metadata Data Tasks Actuator log Alarms Commands

Latest values at 15/05/2020, 10:41:40, 24 seconds from now

Parameter	Latest Value	Timestamp	Real time	Value Type	Metric Type	Unit	Commands
temperature	23	15/05/2020, 10:41:40	23	double	gauge		 
water_level	55	15/05/2020, 10:41:40	55	double	gauge		 

Exercise 1 - Step 3

- Open the rule designer, which gives you a blank canvas
- Add sensor “streamingDataSensor” and actuator “waylayAlarm” to the canvas and connect them
- Configure them both like this

Name	Value
parameter	water_level
threshold	10

Resource: \$

State	Select
Below	<input type="checkbox"/>
Equal	<input type="checkbox"/>
Above	<input checked="" type="checkbox"/>

Trigger policy: Every time

Rocket icon to launch the rule as a task

Notice that we are “templating” the resource using ‘\$’, which means that we will assign a real resource at runtime

Exercise 1 - Step 4

- Save your rule by clicking the 'save button'
- Launch the rule by clicking the rocket icon
- Start the task in reactive mode on an actual resource, cfr the resource that you just created in step 1
- Click the 'view task' button

Deploy task


Name

Resource ⓘ

Start task ⓘ

Task type

Task configuration

 Reactive tasks do not have a task tick. Mostly used in combination with streaming data.

[Advanced Settings](#)

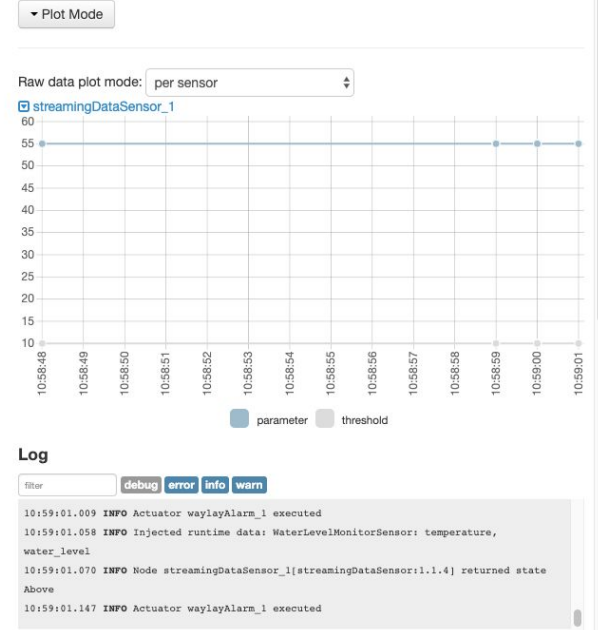
Exercise 1 – Step 5

Dashboard / Tasks / 00000000-0000-0000-0000-0000000000c5

Task 00000000-0000-0000-0000-0000000000c5 YourTaskName status: running show info stop task fork designer

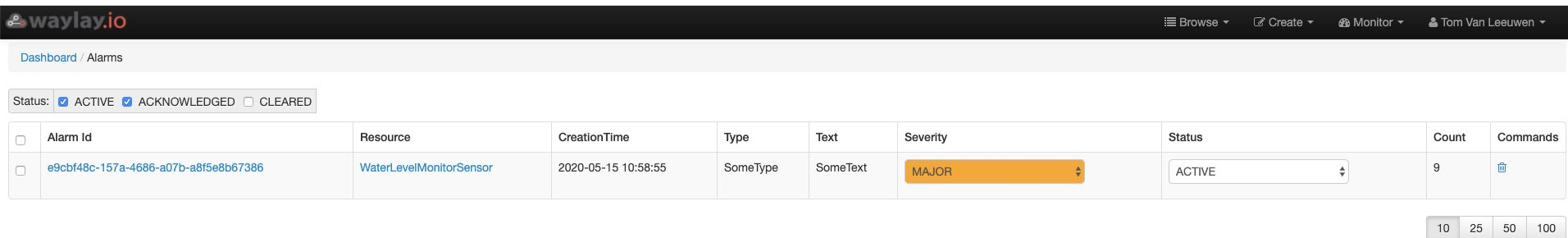
Show timeline

- Keep the task tab open because we want to monitor the behavior of your rule
- Go back to 'Labs' in another tab and shoot some data to Waylay like in step 1. You should see something like this




Exercise 1 – Step 6

- Go to the 'Alarms' tile and see the alarms that were generated by the rule



The screenshot shows the 'Alarms' dashboard in the waylay.io interface. The top navigation bar includes 'Browse', 'Create', 'Monitor', and the user 'Tom Van Leeuwen'. The breadcrumb path is 'Dashboard / Alarms'. A status filter is set to 'ACTIVE'. The table below displays one alarm with the following details:

<input type="checkbox"/>	Alarm Id	Resource	CreationTime	Type	Text	Severity	Status	Count	Commands
<input type="checkbox"/>	e9cbf48c-157a-4686-a07b-a8f5e8b67386	WaterLevelMonitorSensor	2020-05-15 10:58:55	SomeType	SomeText	MAJOR	ACTIVE	9	

At the bottom right of the table, there are pagination controls for 10, 25, 50, and 100 items.

Exercise 1 - Step 7

- Go to your resource and add a new meta data key and value for the threshold

The screenshot shows the waylay.io web interface. At the top, there is a navigation bar with the logo, a 'Browse' menu, a 'Create' button, a 'Monitor' button, and the user name 'Tom Van Leeuwen'. Below the navigation bar, the breadcrumb trail reads 'Dashboard / Resources / WaterLevelMonitorSensor'. A secondary navigation bar contains tabs for 'Metadata', 'Data', 'Tasks', 'Actuator log', 'Alarms', and 'Commands', with 'Metadata' being the active tab. Below the tabs, there are two dropdown menus: 'Select type:' with a value of '-- none --' and 'Select resource parent:' with a search box and a value of '-- resource parent --'. The main content area is titled 'Resource metadata' and contains a table with three columns: 'Key', 'Value', and 'Commands'. The table has three rows: one for 'MyThreshold' with value '60', one for 'id' with value '"WaterLevelMonitorSensor"', and one for an empty row with input fields for 'key' and 'value' and an 'Add' button. The 'Commands' column for the first two rows contains edit and delete icons, while the third row is empty.

waylay.io


Dashboard / Resources / WaterLevelMonitorSensor

Metadata Data Tasks Actuator log Alarms Commands

Select type:
-- none --

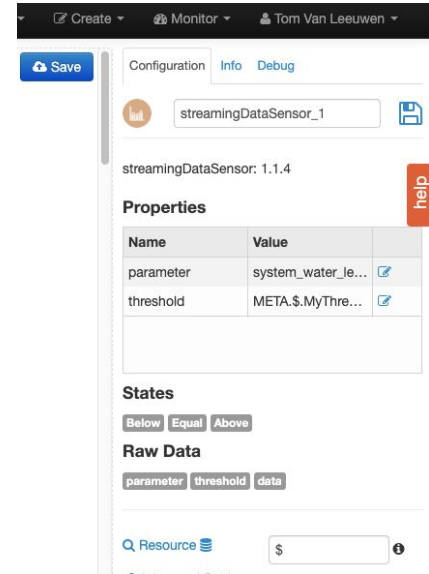
Select resource parent:
Search... -- resource parent --

Resource metadata

Key	Value	Commands
MyThreshold	60	 
id	"WaterLevelMonitorSensor"	
<input type="text" value="key"/>	<input type="text" value="value"/>	<input type="button" value="Add"/>

Exercise 1 - step 8

- Go back to your rule and edit the streamingDataSensor 'threshold' property and enter
 - 'META.\$.MyThreshold'
- Save your rule
- Launch your rule as a task and view it
- Use Labs to shoot some data into it
- Go to your resource meta data and change the threshold value to a large number (keep the task running)
- Use Labs to shoot new data into it (below the new threshold) => see that the alarm is not triggered
 - Note: your previous task may still be running, which will generate its own alarm



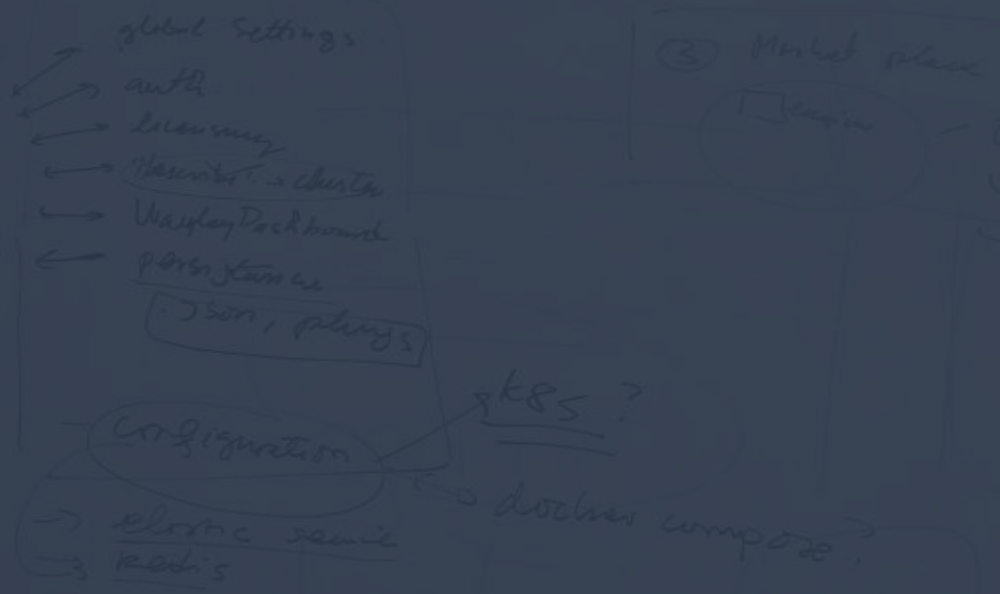
The screenshot shows the configuration page for a rule named 'streamingDataSensor_1'. The 'Configuration' tab is active, displaying the 'streamingDataSensor: 1.1.4' configuration. A 'Properties' table is visible, showing the 'threshold' property set to 'META.\$.MyThre...'. Below the table, there are 'States' (Below, Equal, Above) and 'Raw Data' (parameter, threshold, data) sections. A 'Q Resource' field is also present at the bottom.

Name	Value
parameter	system_water_le...
threshold	META.\$.MyThre...

Exercise 1 - What did you just learn

- You learned how to use Labs to simulate data
- You learned how to create a simple streaming analytics rule
- You learned how to create alarms
- You learned how to monitor a running rule
- You learned how to parameterize thresholds using resource meta data

Exercise 2



Exercise 2 - Creating virtual KPIs

- In this exercise we will study the rule 'virtual_kpi2.json' available on your environment
- This rule is supposed to run periodically

The screenshot shows the Waylay.io designer interface. The main workspace displays a workflow diagram with the following components and connections:

- Swater_sensor** (Sensor) connected to **Input Collector** (Function).
- Spump** (Sensor) connected to **Input Collector** (Function).
- Input Collector** (Function) outputs to a central node labeled **Spump 2**.
- Spump 2** is connected to **beescript** (Script).
- beescript** has two outgoing connections labeled "done" to **debugDialog_1** (Dialog) and **store_bep** (Storage).

The right-hand configuration panel is open for the **input2** function. It shows the following details:

- Configuration:** input2
- getLastValueFromTS:** 1.0.4
- Properties Table:**

Name	Value	
resource		🔗
metric	pump_power_c...	🔗

Below the table, there are sections for **States** (Collected, Not Collected) and **Raw Data** (lastValue, lasttimestamp). At the bottom, there is a search field for the resource, currently showing "\$pump", and a "View" button.

Exercise 2 - step 1

The screenshot shows the Waylay.io playground interface. On the left, there are panels for 'Sensors', 'Gates', and 'Actuators'. The main workspace contains a flow diagram with nodes: '\$water_sensor', 'Spump', 'debugDialog_1', and 'store_bep'. A red box highlights two nodes on the left: '\$water_sensor' and 'Spump'. The right sidebar shows the configuration for 'getLastValueFromTS: 1.0.4'. The 'Properties' section contains a table:

Name	Value
resource	
metric	pump_power_c...

The 'States' section shows 'Collected' and 'Not Collected' options. The 'Raw Data' section shows 'lastvalue' and 'lasttimestamp' options. The 'Resource' dropdown is set to '\$pump'.

These two nodes take the last value out of the time series data for two different metrics from two different resources
Study the configuration for both of these nodes

Exercise 2 - step 2

The screenshot shows the waylay.io playground interface. The main workspace contains a workflow diagram with several nodes: 'water_sensor', 'input1 Collected from sensor', 'Spump', 'bipscript', 'debugDialog_1', and 'store_bep'. A red box highlights the 'bipscript' node, which is a script node. The 'bipscript' node is connected to 'input1 Collected from sensor' and 'input2 Collected from sensor'. It has two outgoing connections labeled 'done' to 'debugDialog_1' and 'store_bep'. The right sidebar shows the configuration for the selected 'bipscript' node, including a 'Configuration' section with 'Input2', a 'Properties' table, and 'States' and 'Raw Data' sections.

Name	Value
resource	
metric	pump_power_c...

States
Collected
Not Collected

Raw Data
lastvalue
lasttimestamp

This is a script node that will run some javascript code
Have a look at the code. The formula is nonsense. Feel free to be creative and change it. Understand how information is passed between the time series nodes and the script node.

Notice the sequence number 2: this node will only be executed when the time series nodes have finished with retrieving their data

Exercise 2 - step 3

The screenshot shows the waylay.io playground interface. The main workspace contains a workflow diagram with the following nodes and connections:

- Water Sensor** (blue) connected to **Input1 Collected** (blue).
- Script** (orange) node labeled `bepscript` with `data: pump` and `script: pump` properties.
- Debug Dialog** (purple) node labeled `debugDialog_1`.
- Store** (black) node labeled `store_bep`, highlighted with a red box.

Connections are labeled `done`. The `store_bep` node is highlighted with a red box. The right sidebar shows the configuration for the `store_bep` node:

Configuration: Info Debug

Input: input2

getLastValueFromTS: 1.0.4

Properties

Name	Value	
resource		edit
metric	pump_power_c...	edit

States

Collected Not Collected

Raw Data

lastvalue lasttimestamp

This node will store the calculated variable 'bep' as a new metric in the time series database. It has an empty 'resource' property, which means that it will use the one from the previous node (the script node)

Study its configuration and understand how data is passed between the nodes in this rule

Exercise 2 - step 4

- You can now launch this rule. Make sure you launch it as a periodic task, set periodicity=1min
- The wizard will ask you to assign resources to
 - \$water_sensor
 - \$pump
- You can create those using Labs - shoot some data in Waylay for 2 different resources (make sure to set the flag 'store data' to ensure that data is stored in the time series database)

Exercise 2 - step 5

- If your task is executing properly and you are shooting some randomized waterlevel and power consumption data in Waylay, your task shall calculate every 1 min the derived/virtual metric 'bep' and store it as a new metric for the pump resource
- Go to the 'Resources' tile and select the pump resource. See in the data 'tab' the new metric bep appearing
- Click the 'bep' metric and the Time Series Analytics data explorer will open

Exercise 2 - What did you just learn

- You learned how to use periodic tasks
- You learned how to use historical time series data in rules
- You learned how to combine data from multiple metrics/resources and create a derived/virtual metric

End

