OST
Ostschweizer
Fachhochschule

Digital Manufacturing

Ex04 – Cloud Computing

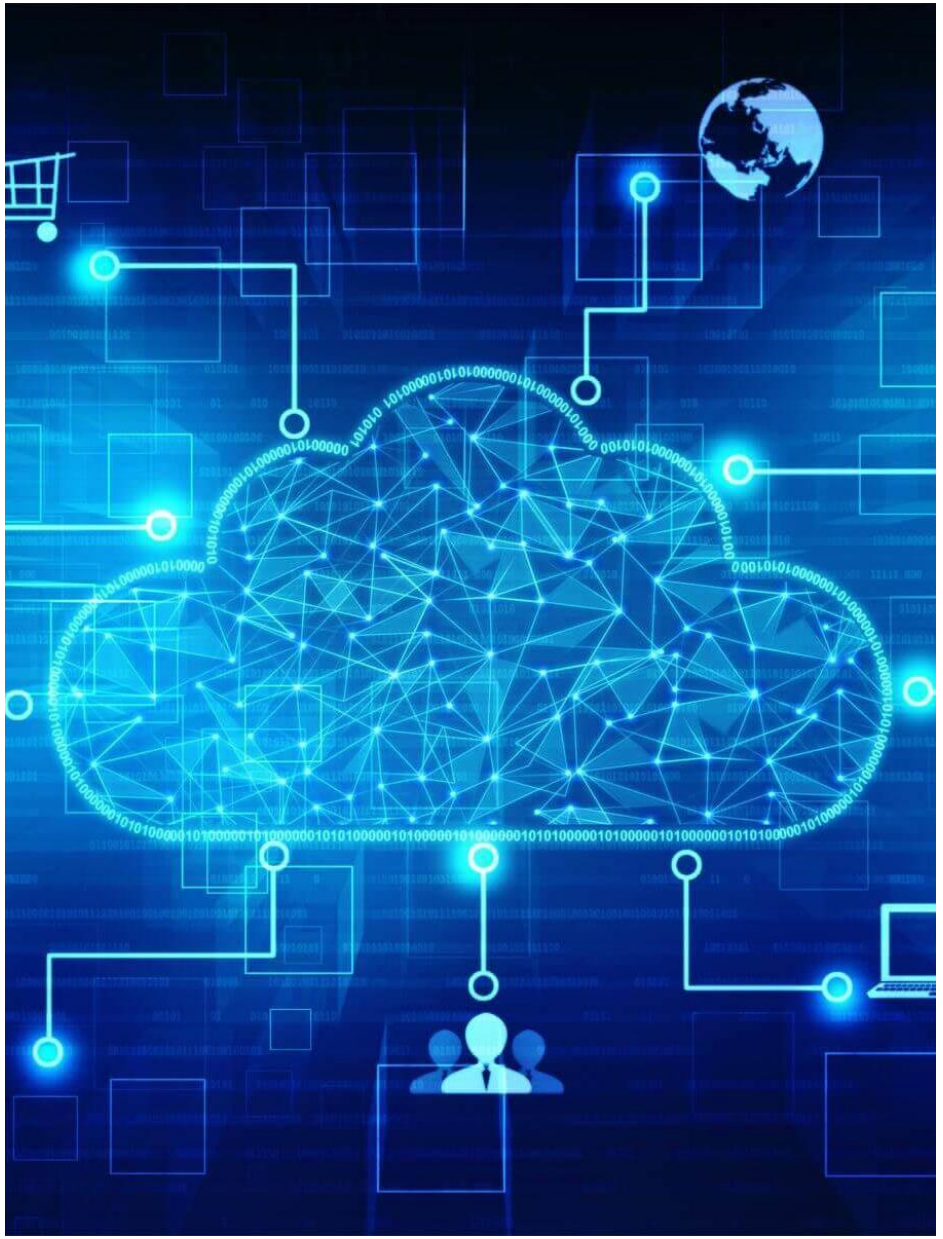
Lukas Kretschmar, MSc FHO in Engineering (ICT & BEP)

15. November 2021

Industrial Engineering (OST-RJ)

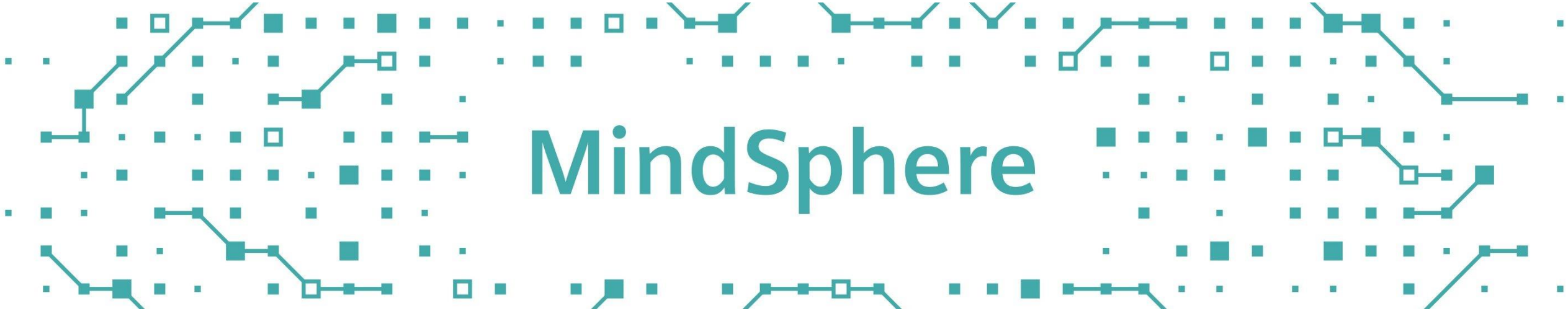
Today's Objectives

- ❑ Create physical and virtual Assets
- ❑ Send Sensor Data to the Cloud
- ❑ Receive Machine Data on your Asset
- ❑ Create a Dashboard for your Mock Machine (Raspberry Pi)
  - ❑ Visualize the Temperatures
  - ❑ Layout your Tiles



Create your Account for Siemens MindSphere

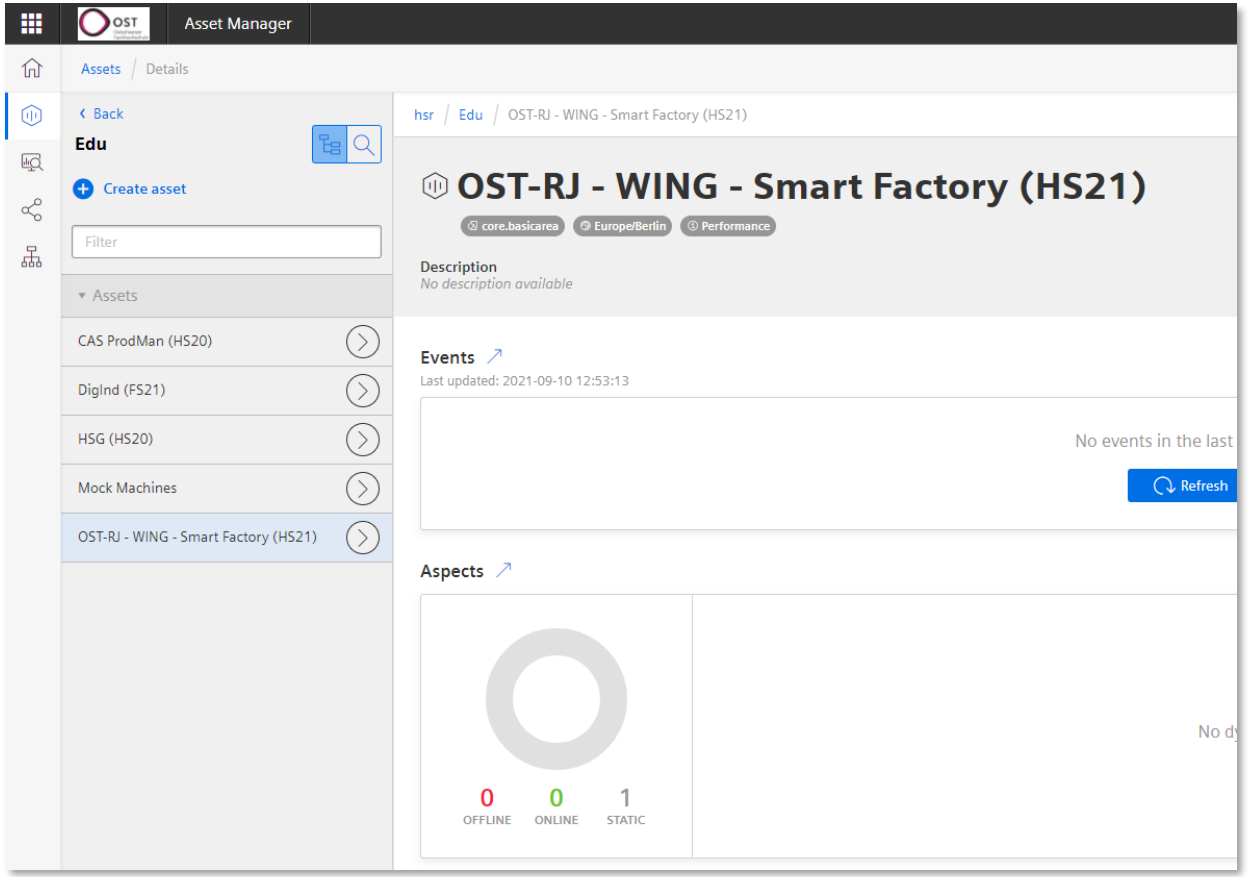
Preparation



OST

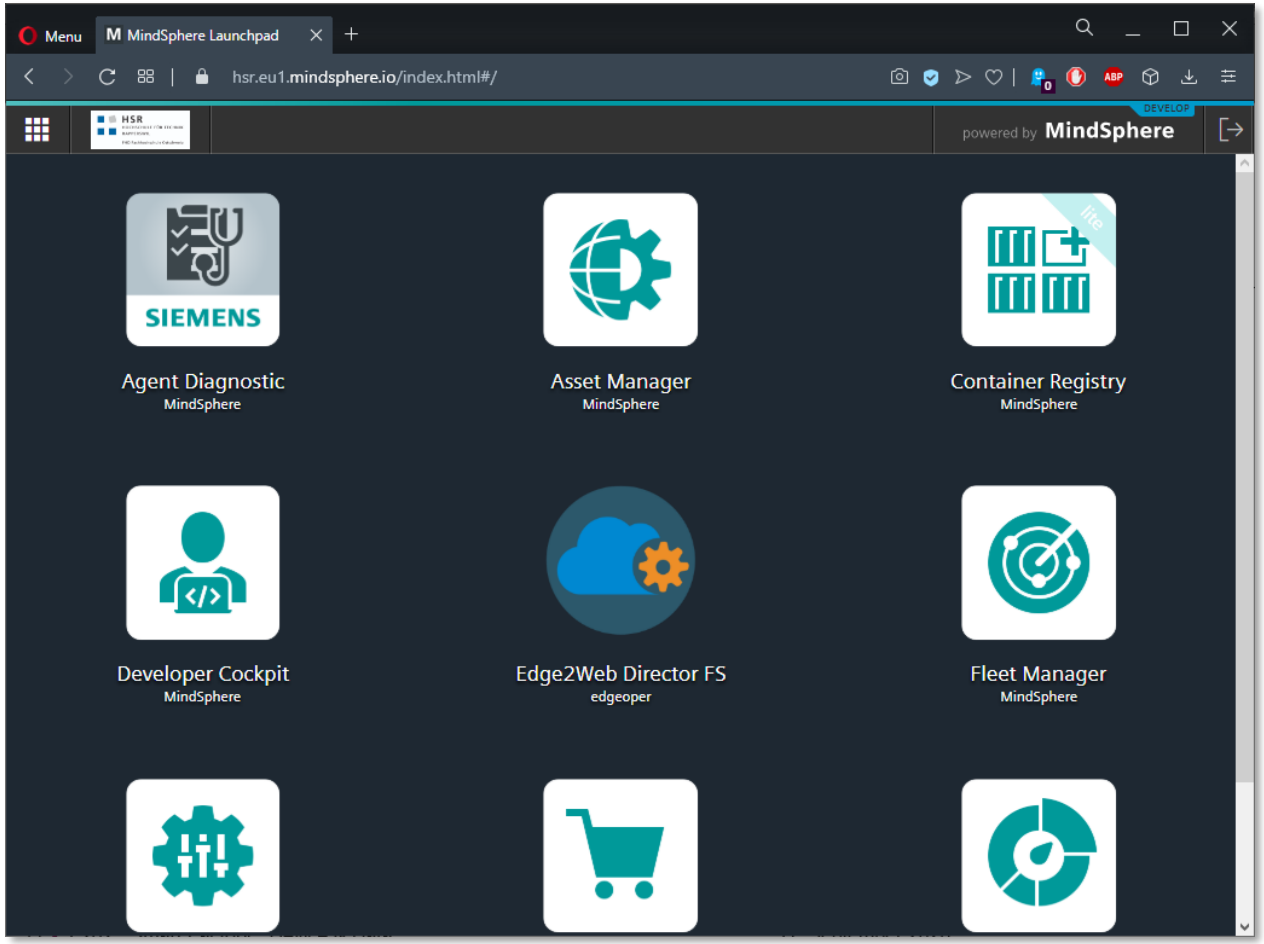
Create your Siemens MindSphere Account

- In this exercise, you will send your sensor data to the cloud
- You'll use Siemens MindSphere as cloud platform
- And to do so, you'll need to create an account to work with the platform
- You have received an e-mail from Siemens with a link to create an account
- Just follow the steps to create your login



Test your Login

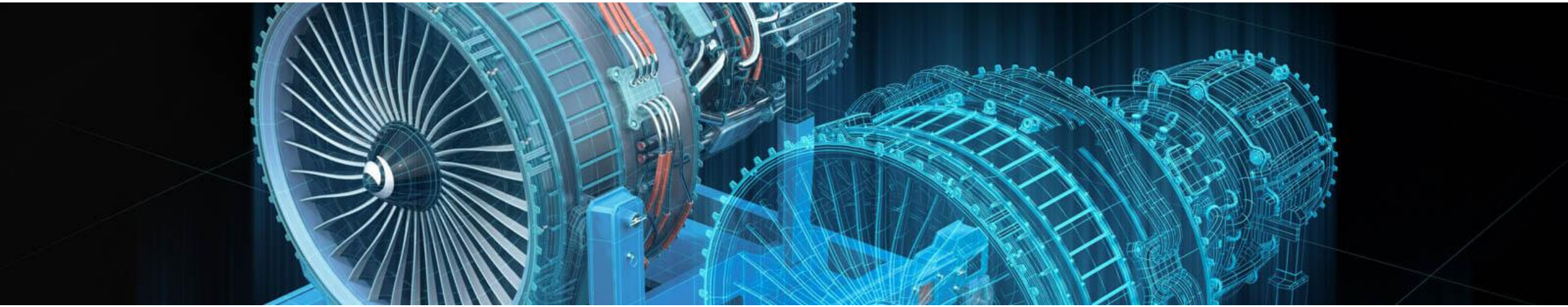
- After creating the account, test your login to be ready for the exercise
- Just use the following link, and if you see the page below, it worked
  - https://hsr.eu1.mindsphere.io
  - Your page may look different (fewer icons)



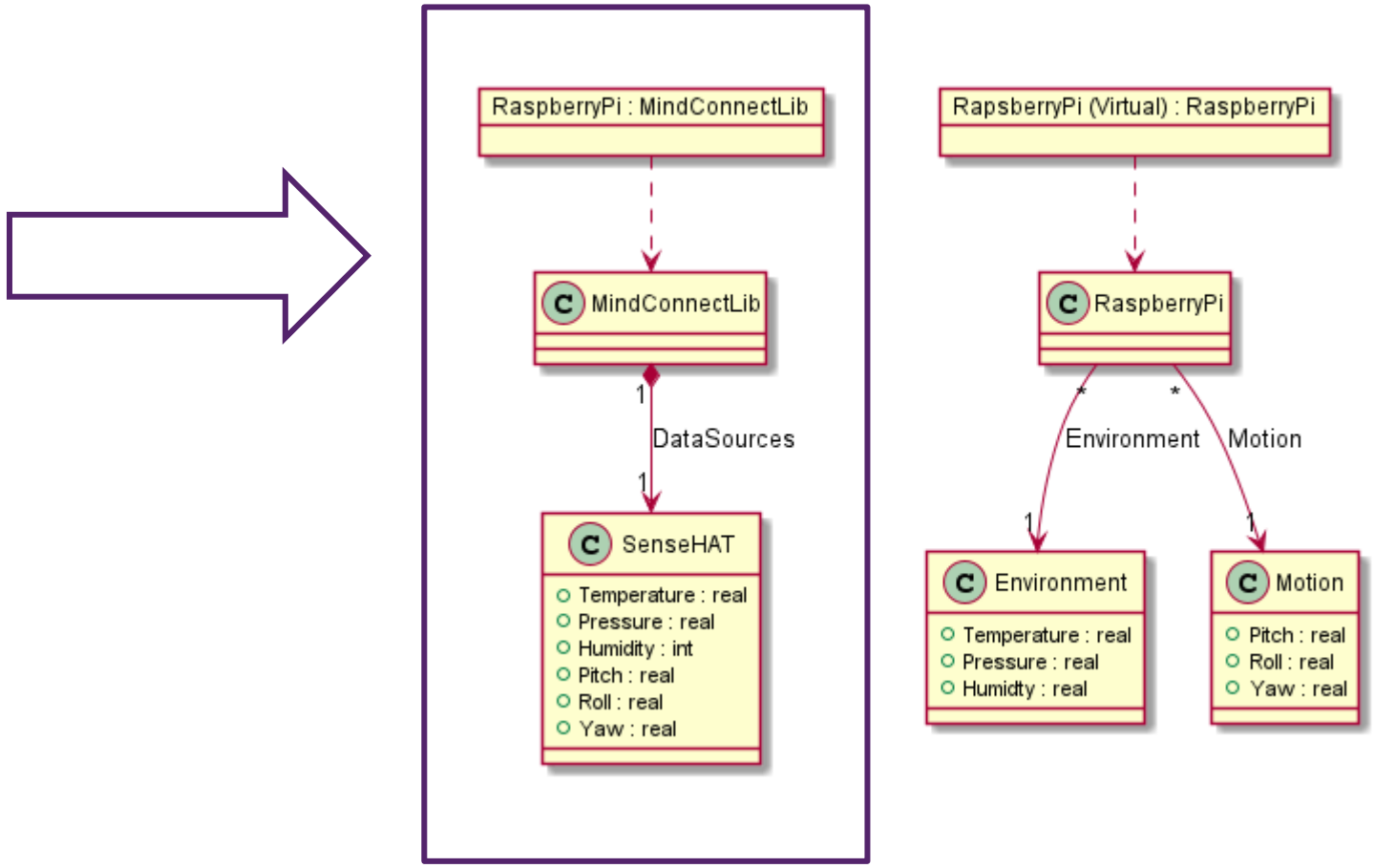
OST

Physical Asset

Get your Digital Twin up and running (Part 1)

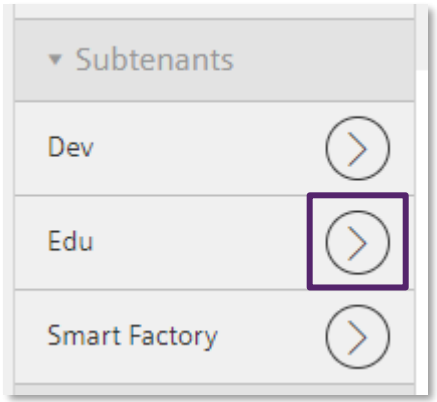


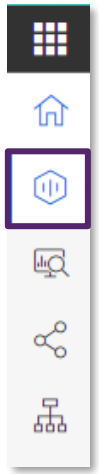
Physical Asset



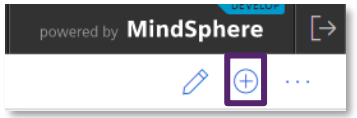
Create an Asset for your Assets

1. Login to MindSphere https://hsr.eu1.mindsphere.io
2. Open the Asset Manager
3. Open the Asset list
4. Navigate to the course asset *Edu > DIMA (HS21)*

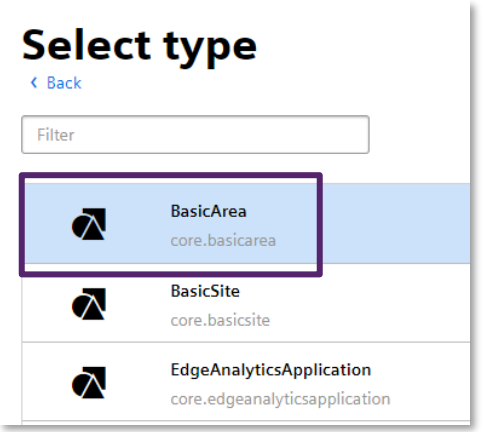




5. Create a new child asset (top right corner)



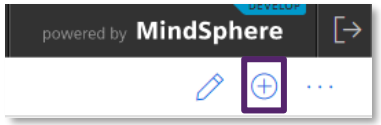
6. Select *BasicArea* as type and click *Create (bottom right corner)*
7. Enter your name as the assets name
8. You should now see your asset in the asset list on the left



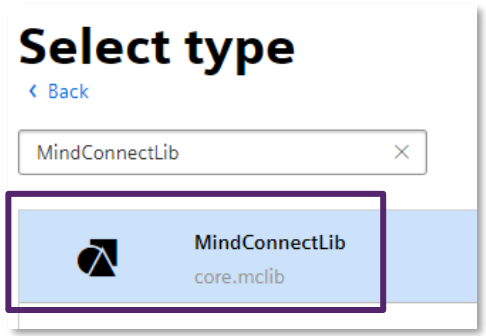
OST

Create your physical Asset

1. Select your created asset



2. Again, create a child asset but this this time it must be of type *MindConnectLib*

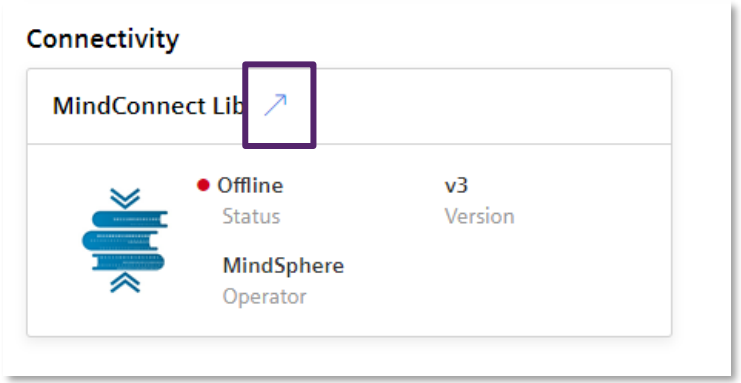


3. Give it the name *Raspberry Pi [Nr of RPi]* e.g. Raspberry Pi 42

4. You'll see your asset in the list on the left

5. Select your created *MindConnectLib* asset

6. On the bottom of the left side, you'll find a *Connectivity* group



7. Click the arrow in the box's heading

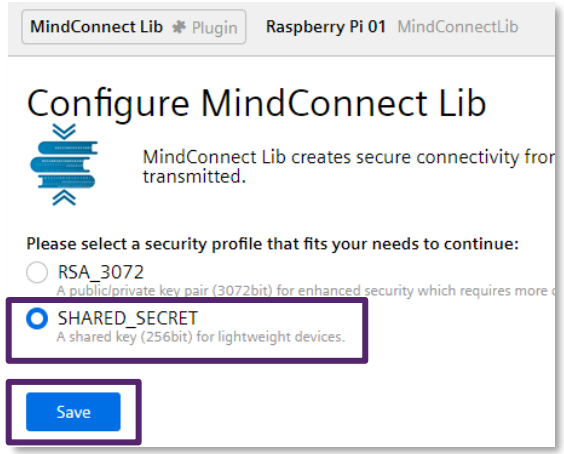
Configure your MindConnectLib Asset

1. The first time you open a *MindConnectLib* asset, you'll get redirected to the configuration screen

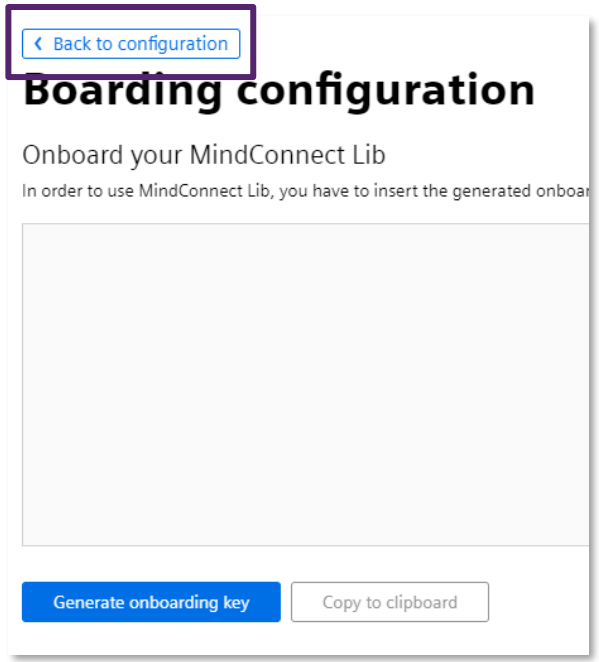
2. Select *SHARED_SECRET*

   - A shared secret is something your asset and your Raspberry Pi will know, so it can identify itself within the MindSphere platform when sending data

3. Click save to get to the next view

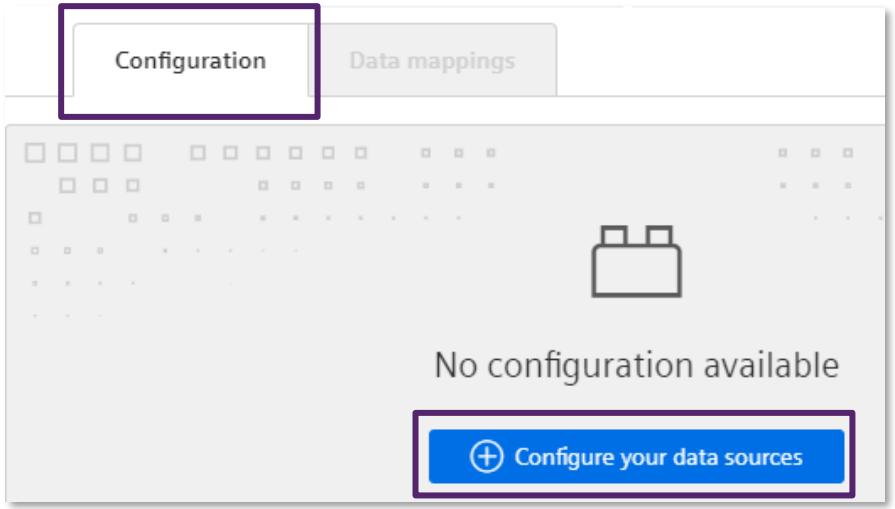


4. On the next screen, click *Back to configuration*

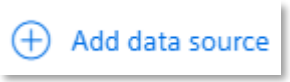


Create your Data Source

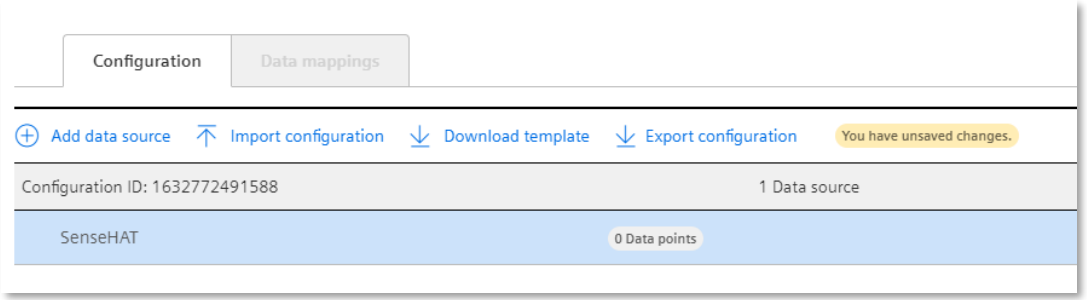
1. Create a new configuration



2. Add a new data source, give it a meaningful name (e.g. SenseHAT) and click *Accept*

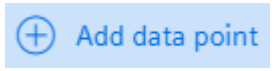


- A data source just groups your data points and has no further meaning



Create your Data Points

- For each variable you used in the last exercise (temperature, pressure, humidity, etc) do the following steps
  1. Click *Add data point*

  2. Enter a meaningful name
  3. Choose a *type* according to the data that will be sent
  4. Define a *unit*
     - Units are used later in data point mappings
- When completed, save your changes

- When completed, your data source should look like below

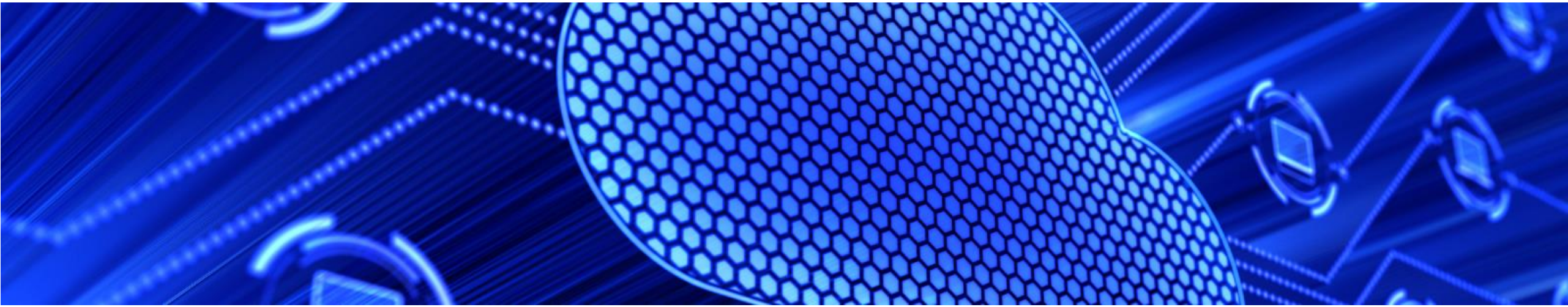
SenseHAT — 6 Data points

Data point	Data point ID	Linked	Data type	Unit
Temperature	1600081578627		DOUBLE	°C
Pressure	1600099715090		DOUBLE	hPa
Humidity	1600099735463		INT	%
Pitch	1600099755767		DOUBLE	°
Roll	1600099771715		DOUBLE	°
Yaw	1600099783602		DOUBLE	°

- You have successfully prepared your physical asset in MindSphere
- Next, you'll complete the configuration on your Raspberry Pi

Send Data to the Cloud

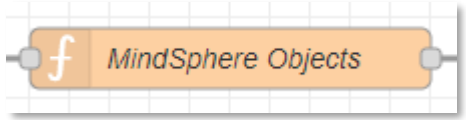
Fire and forget



OST

Update the existing Content

1. Open Node-RED
2. Switch to the *MindSphere* flow
3. Open the *MindSphere Objects* node



4. You'll find the following code

```
const values = [
    {
        "dataPointId": "",
        "qualityCode": "1",
        "value": global.get("temp")
    }
];

msg._time = new Date();
msg.payload = values;
return msg;
```

5. Go to your data source on MindSphere, to get the data point id

SenseHAT — 6 Data points

Data point	Data point ID	Linked	Data type	Unit
Temperature	1600081578627		DOUBLE	°C
Pressure	1600099715090		DOUBLE	hPa
Humidity	1600099735463		INT	%
Pitch	1600099755767		DOUBLE	°
Roll	1600099771715		DOUBLE	°
Yaw	1600099783602		DOUBLE	°

6. Copy the value and paste it to your property in the flow

```
const values = [
    {
        "dataPointId": "1600081578627",
        "qualityCode": "1",
        "value": global.get("temp")
    }
];
```

Value must be string

- To avoid any troubles later in the exercise, please note the following
  - Values sent to MindSphere MUST be of type string (text)

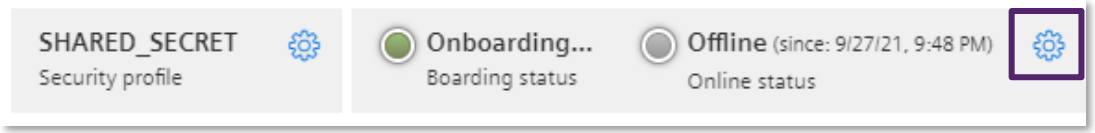
```
1  const values = [
2      {
3          "dataPointId": "",
4          "qualityCode": "1",
5          "value": global.get("temp")
6      }
7  ];
8
9  msg._time = new Date();
10 msg.payload = values;
11 return msg;
```

  - It doesn't matter what data type you'll use on the cloud, when sending, it must be string

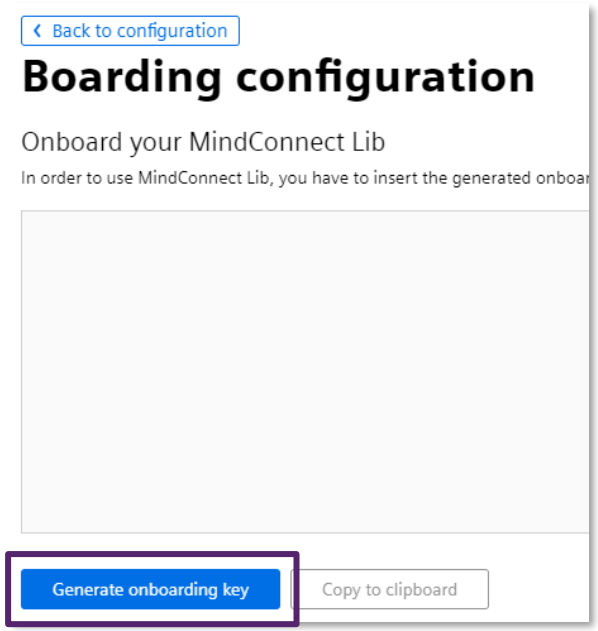
- In the example, the global variable *temp* contains a string
  - The *toFixed()* method stores the value as string
- For your code, this might be relevant
  - The easiest way for you is to use the *toFixed()* method every time you want to send a value

Generate the Secret used for Communication

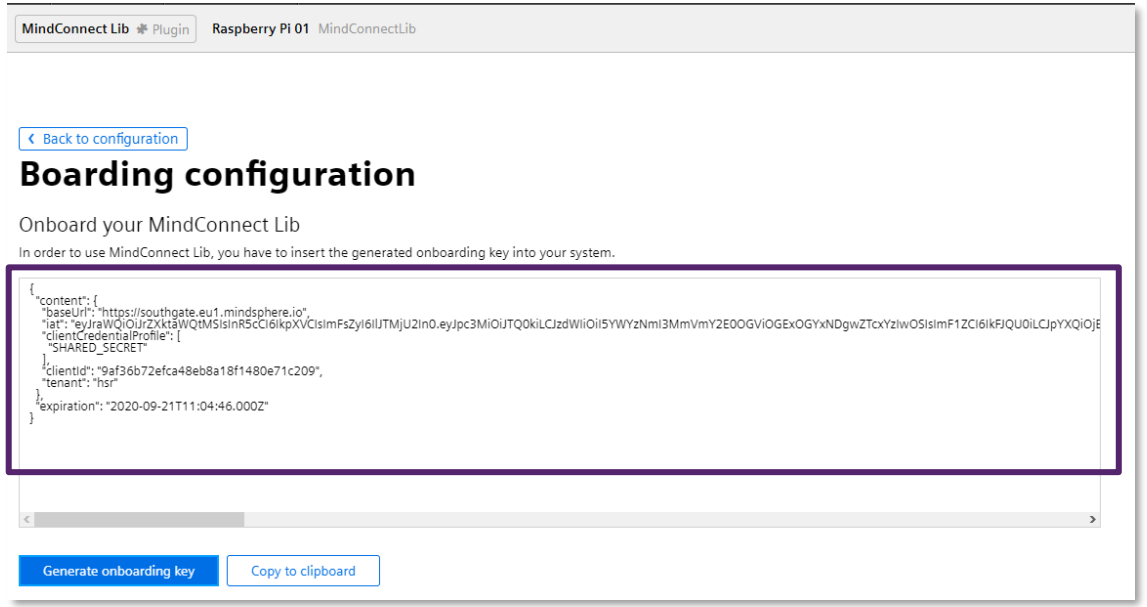
1. On the top right corner, open the *onboarding* menu



2. Click *Generate onboarding key*

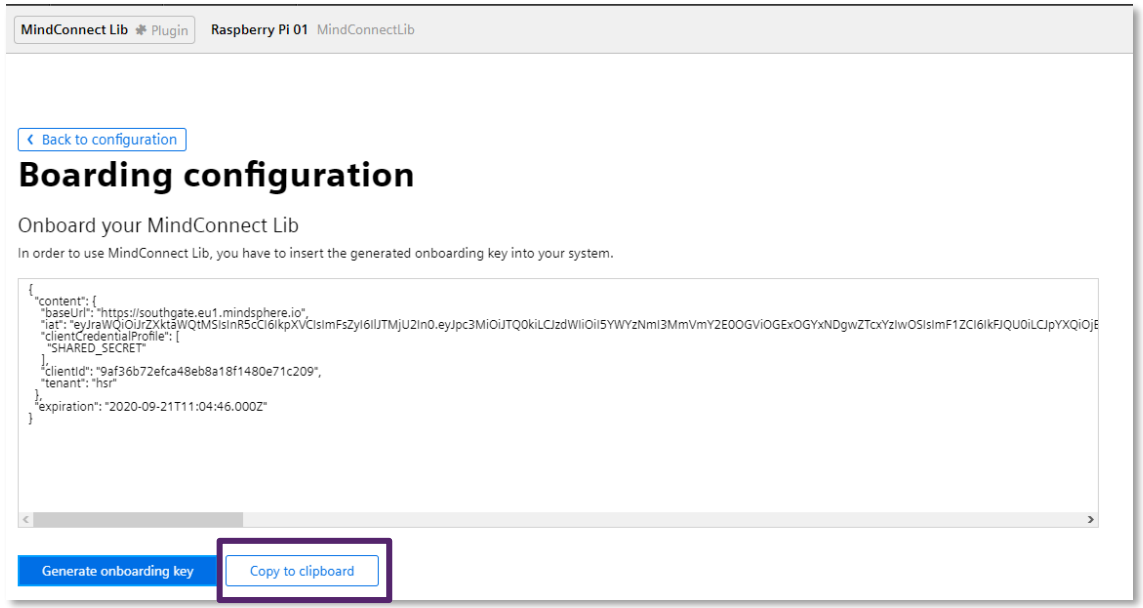


3. In the field above the button, the secret – in the form of a JSON object – will be shown

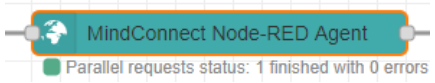


Configure your MindConnect Node

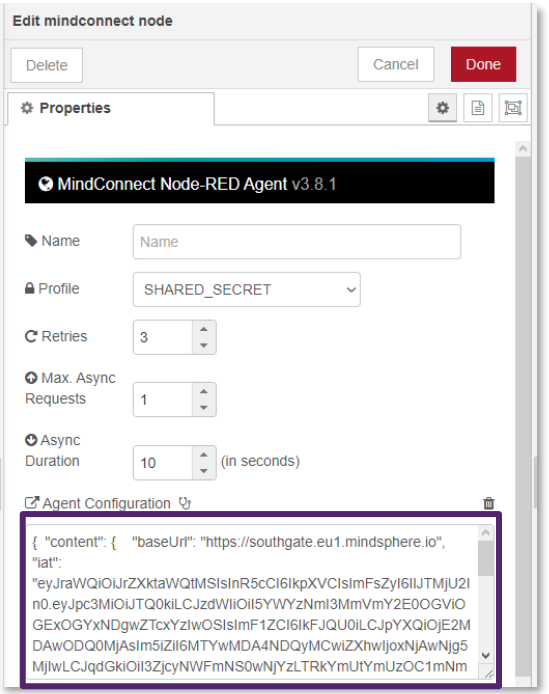
1. Click *Copy to clipboard* (or select all the content within the field and copy it)



2. Open your Node-RED webpage
3. Switch to the *MindSphere* flow
4. Select the *MindConnect* node



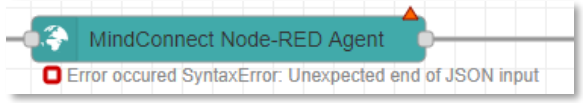
5. Paste the key into the field
6. You have successfully configured your node



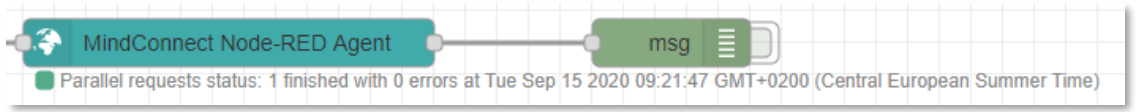
OST

Deploy the Update

- Let's check if your changes were successful
- Deploy the changes you made in Node-RED
- After a while, you're MindConnect node should change
  - From

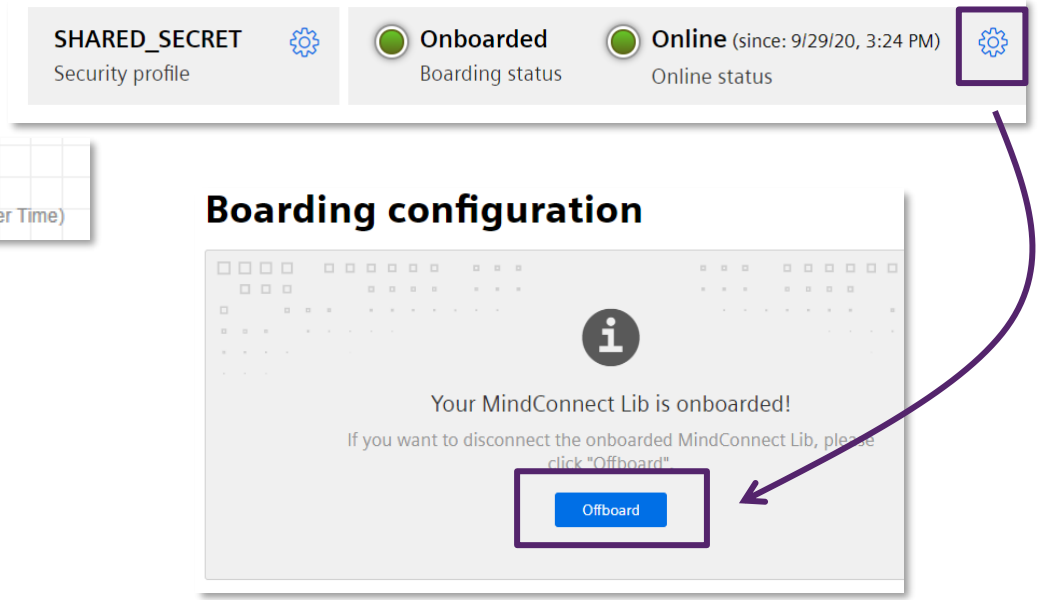


  - To



- If you still have errors, please check
  - Is the data point id correct
  - Is the shared secrete still set

- Important
  - Sometimes, the node will still show errors
  - In this case, create a new shared secret on MindSphere and replace it in the *MindConnect* node

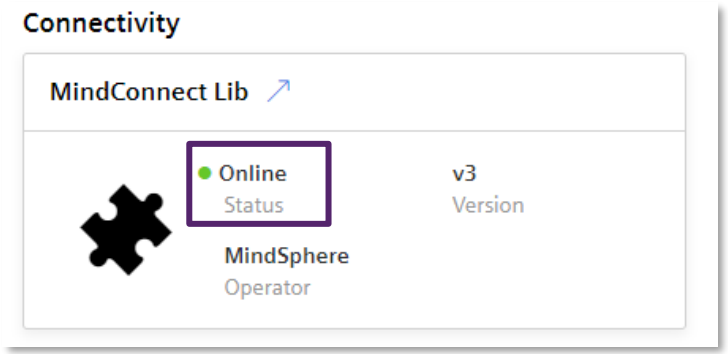


OST

Check if Data is received

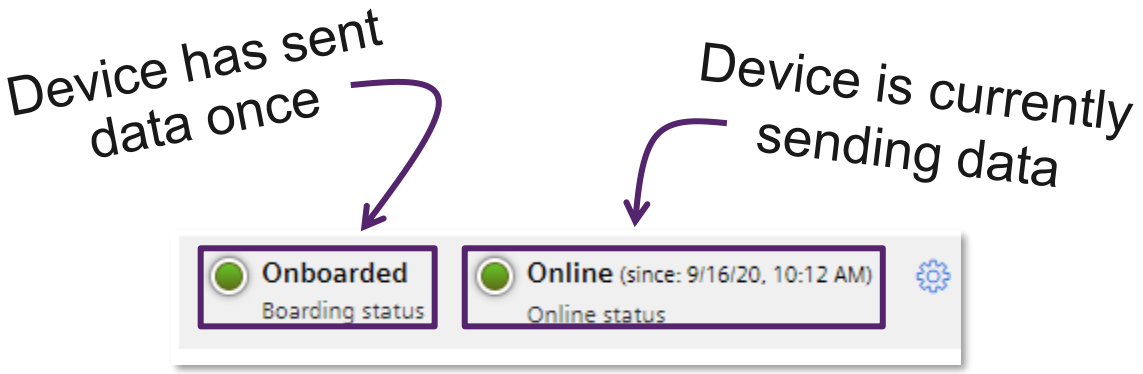
Physical Asset

1. Navigate to your physical asset in the Asset Manager
2. Select your physical asset
3. You should see that the state is *Online*



MindConnectLib

1. You can also open the *MindConnectLib* type
2. On the top right corner, you should see the following information



Extend your Data

- Being able to send the temperature to the cloud, let's extend the object to send the other data as well
  - Add the pressure to the data object

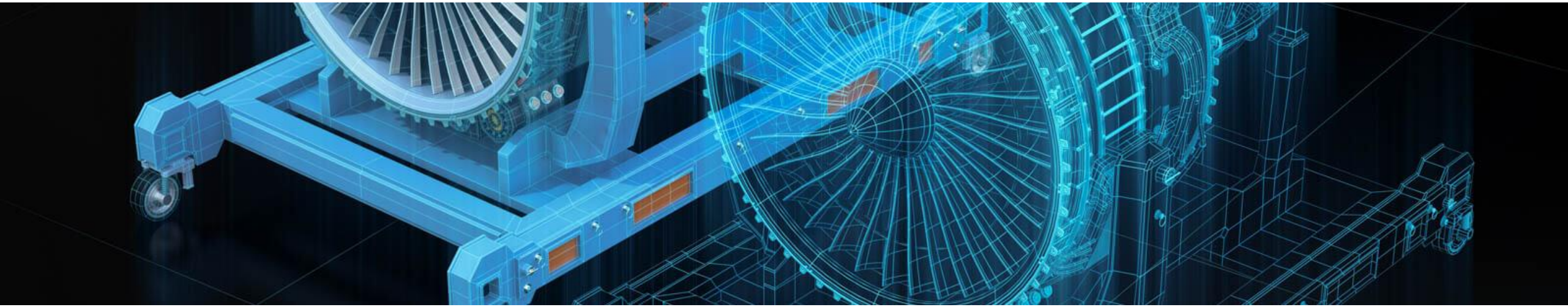
```
const values = [
    {
        "dataPointId": "1600081578627",
        "qualityCode": "1",
        "value": global.get("temp")
    },
    {
        "dataPointId": "1600099715090",
        "qualityCode": "1",
        "value": global.get("press")
    }
];
```

  - Deploy the flow again to check if it still works
- Repeat it for humidity, pitch, roll and yaw

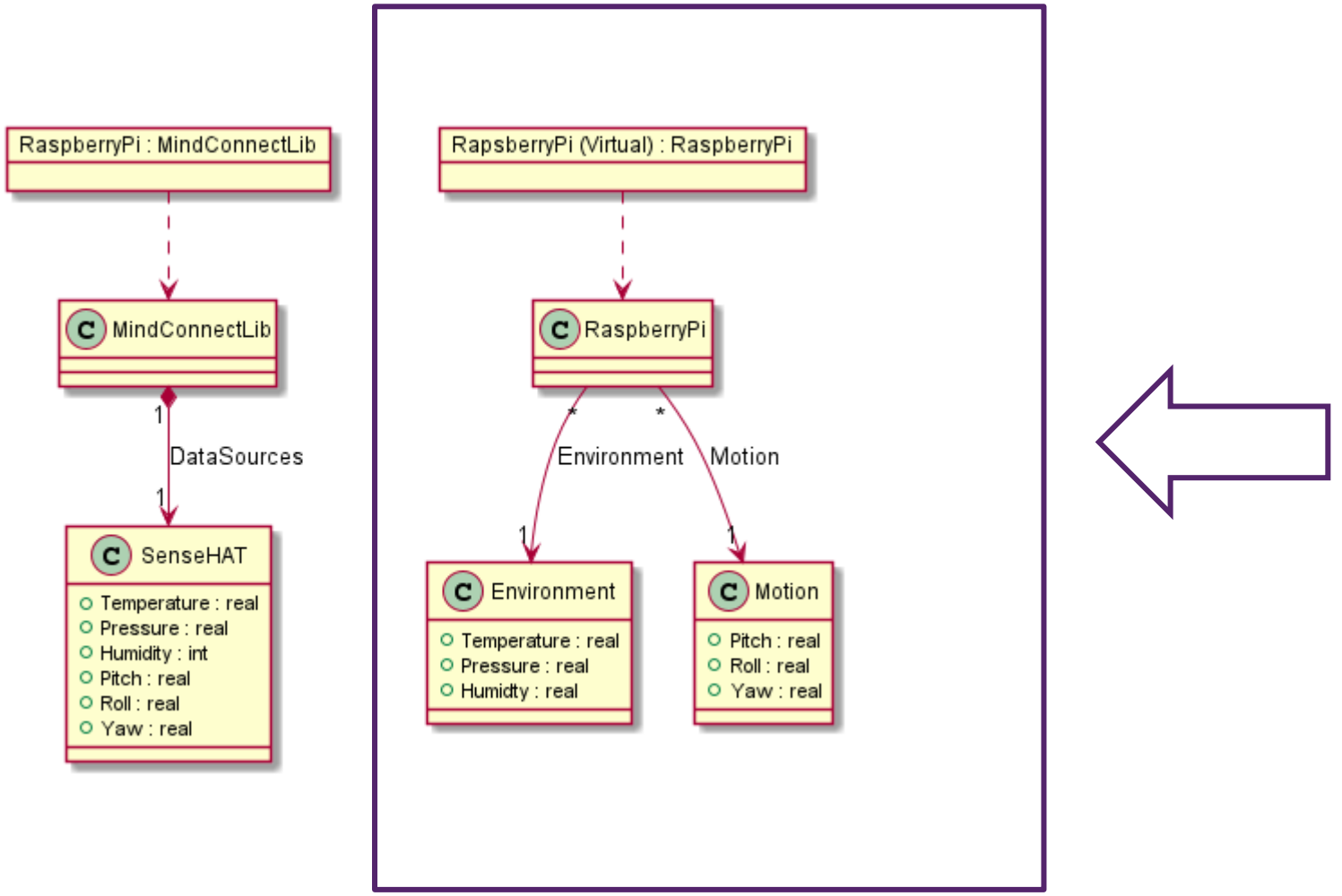
- Please note
  - Always deploy your flow after each change so you are sure your code works as expected
    - Debugging an error isn't easy
  - If you run into an issue
    - Double check your data point id
      - Using the same id twice (copy/paste error) will result in an error
    - Check if you have the objects in the array separated with commas
    - Check if the last entry has no comma
      - Entries in an array or properties in an object MUST NOT end with a comma

Virtual Asset

Get your Digital Twin up and running (Part 2)



Virtual Asset

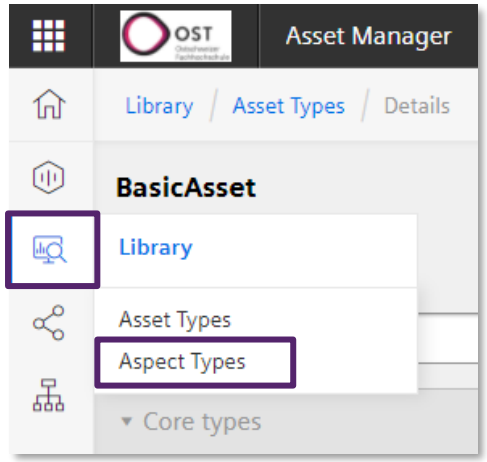


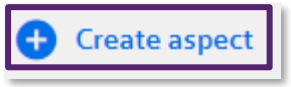
Steps to create a virtual Asset

- Compared to the physical asset, a virtual asset is a bit more complex to create
  - We have to define the type ourselves first
  - And then we can create the asset
- Creating a virtual asset requires the following steps
  1. Creating aspects
  2. Creating a type
  3. Creating the asset
- Please note
  - The terms and distinction between aspects and types is MindSphere specific
  - Other platforms may use other names, but the concepts used will be the same

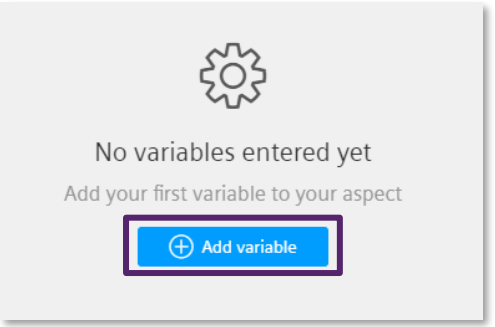
Creating an Aspect

- An **aspect** is the building block of **types**
  - It contains variables which will be used to map on the data points of physical assets
- Aspects are here for reusability
  - You can understand aspects as groups of variables
  - Used wisely, you don't have to redefine variables multiple times



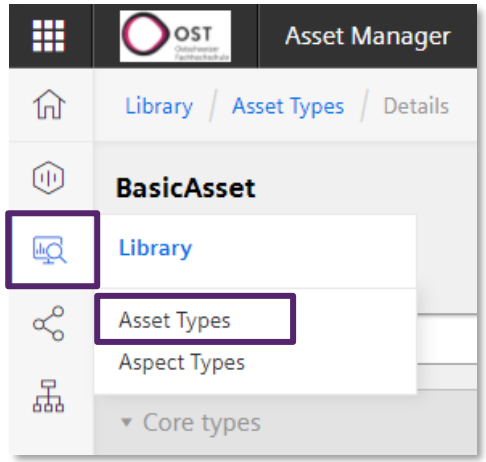
1. Open the *Aspects* view in the *Asset Manager*
2. Click *Create aspect* 

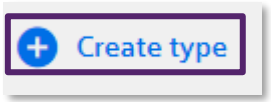
3. Enter a name according the following pattern:
   Edu_[YourFullName]_Environment
   e.g.: Edu_DoeJohnny_Environment
4. Click *Add variable*
5. Add the variables for … at the bottom
   - Temperature
   - Pressure
   - Humidity
6. Save your aspect



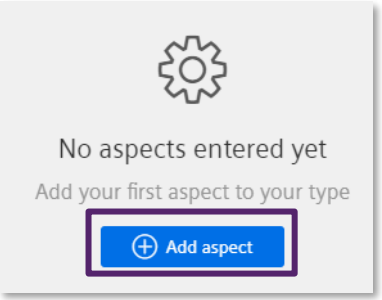
Creating a Type

- A **type** is the base for your **assets**
  - The type is an abstract definition what variables (within aspects) will appear together
  - Multiple assets can use the same type, imagine it as a blueprint
- Like aspects, types are here for reusability



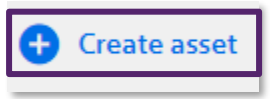
1. Open the *Types* view in the Asset Manager
2. Create a new type 

3. Enter the name with the following pattern:
   Edu_[YourFullName]_RaspberryPi
   e.g.: Edu_DoeJohnny_RaspberryPi
4. Open the *Aspects* group
5. Click *Add aspect*
6. Give the meaningful name and select your aspect
7. Save your type

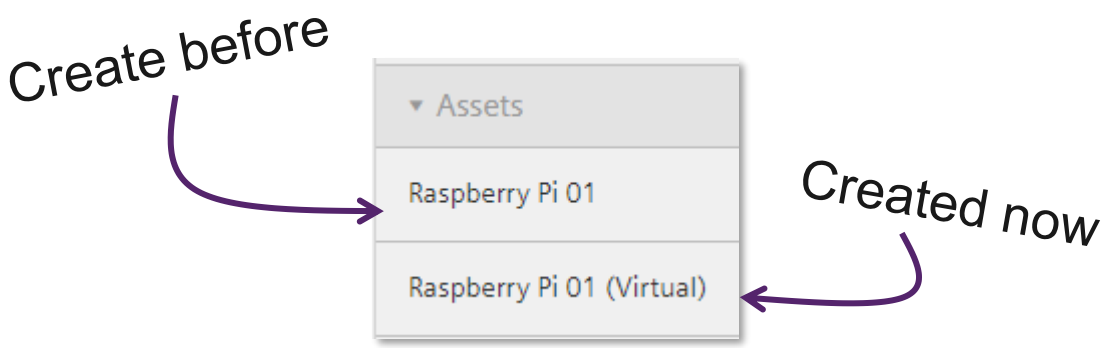


Create your virtual Asset

- Now it's time to create your **virtual asset**

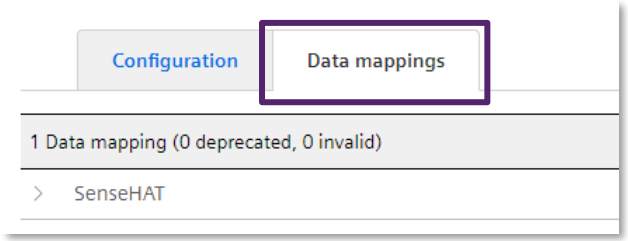
1. Open the *Assets* view in the Asset Manager
2. Navigate to your assets
3. Click *Create asset* 

4. Filter for your type *Edu_[YourFullName]_RaspberryPi*
5. Select the type and click *Create*
6. Give your asset a name
   We suggest *Raspberry Pi [Nr] (Virtual)*
7. Click *Save* to create your asset
8. Your virtual asset is now visible in your list on the left



Mapping your virtual Asset on your physical Asset

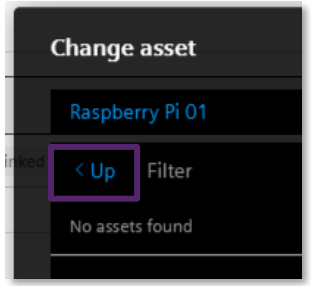
1. Select your physical asset in the Asset Manager
2. Open the *MindConnectLib*
3. Switch to the *Data mappings* tab



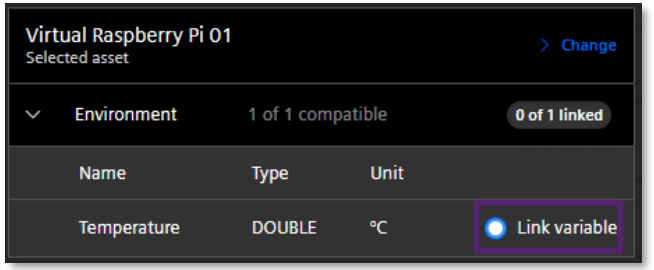
4. Open your *data source* (e.g., SenseHAT)
5. Select *Link variable* (on the right side of a data point)
6. A view will open pointing to the current data point

6. Click > *Change* to navigate to your virtual asset

7. In the *Change asset* dialog, click < *Up*



8. Select your virtual asset and click *Accept*
9. Select *Link variable* for your data point
   - You'll only see variables that are compatible in type and unit

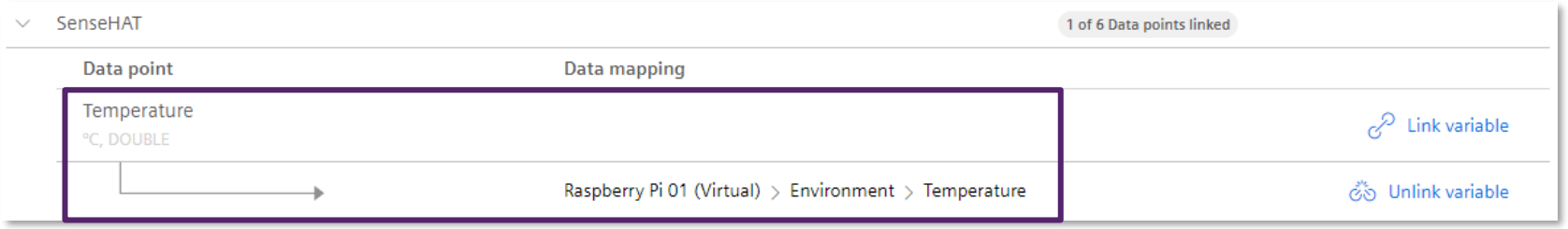


10. Click *Accept*

OST

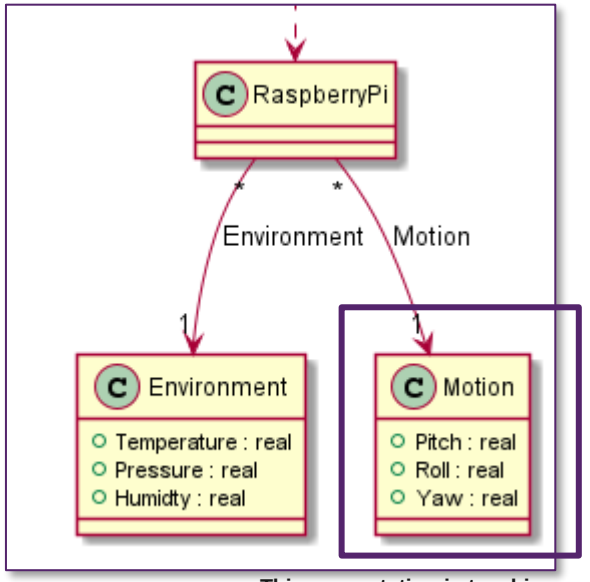
Complete your virtual Asset and the Links

- You should now see that your data point is link with your variable



  - With *Unlink variable* you can remove the link between a data point and a variable

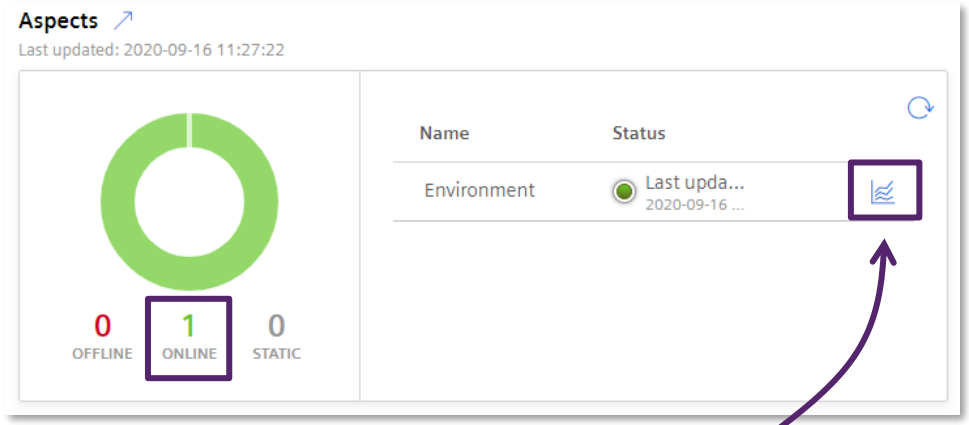
- Now, do the mapping for all the other data points
  - Add also the missing aspect
  - And extend your type of the virtual asset



Check if Data is received

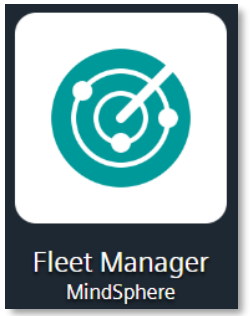
On your virtual Asset

1. Select your virtual asset in the Asset Manager
2. You should see the following in the details

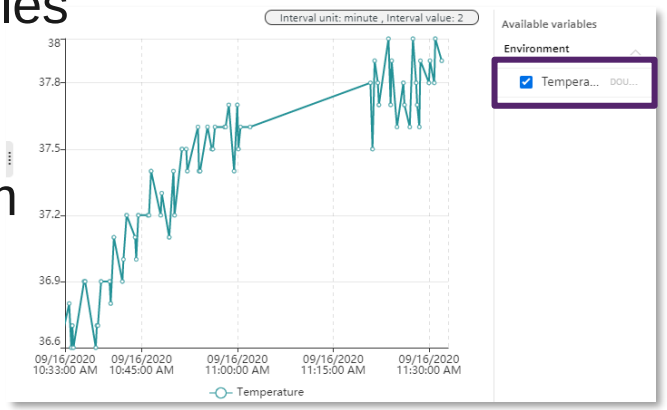


- You can also open a simple time series visualization of the received data

In the Fleet Manager



1. Open the *Fleet Manager*
2. Search for your virtual asset
3. Select your asset
4. Choose *Time Series*
5. Select the variables on the right side that you want to see the data from

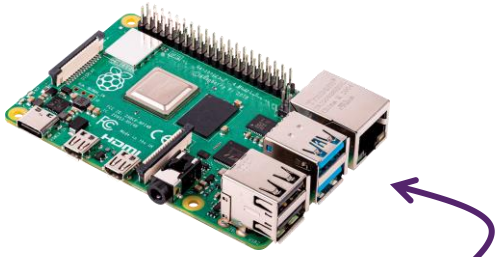


Machine Data in the Cloud

Let's have some Fun with real Data

The Scenario

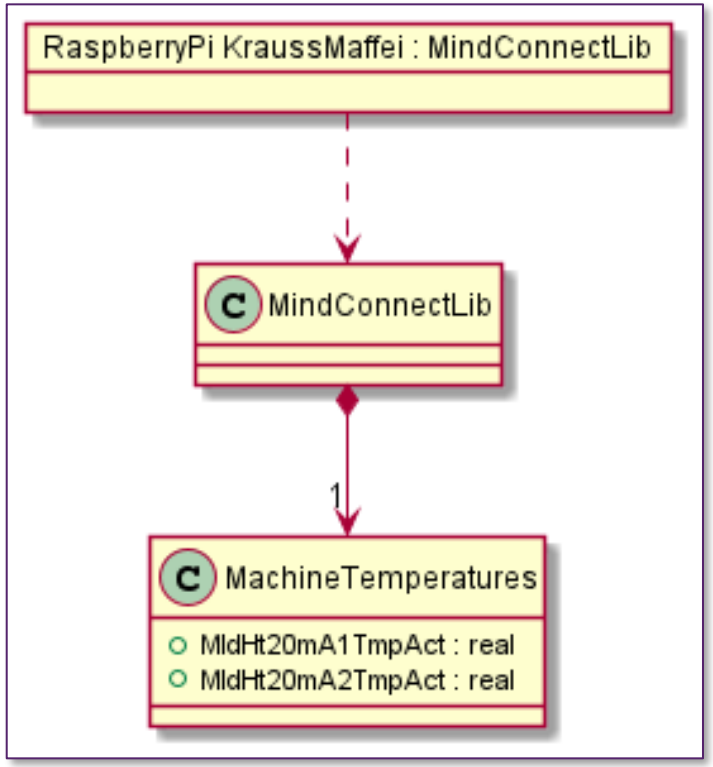
- Your Raspberry Pi is actually a mock machine
  - It contains data of real production cycles collected on a Krauss Maffei PX 120-038
  - When it has power and configured, it will send data between 07:30 and 20:30 every day
- With the next steps, you'll prepare your Raspberry Pi and MindSphere to send and receive this data
  - This is a preparation for the next exercise
- In the next exercise, there will be another Raspberry Pi that will send more data of the same production cycle to get the full scenario



- Your Raspberry Pi
  - Sending 2 Data Points
- Other Raspberry Pi
  - Sending 20+ Data Points

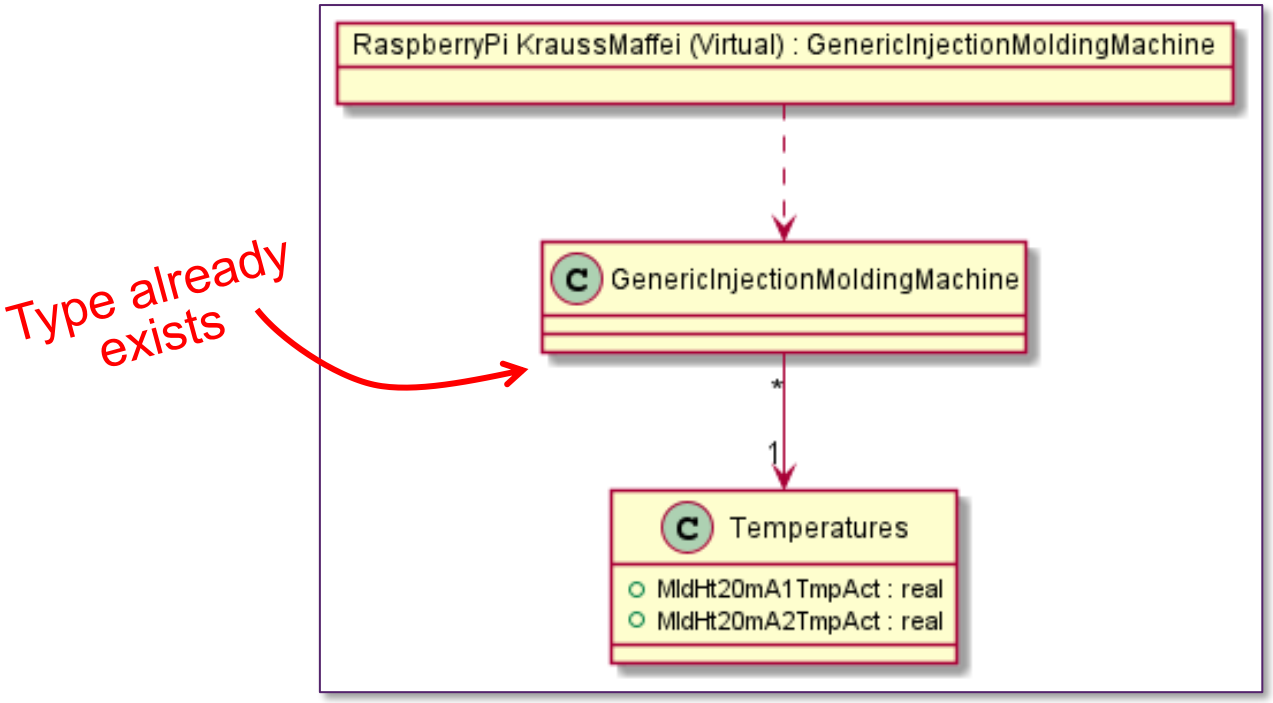
Create your physical Asset

- Your Raspberry Pi is prepared to send real machine data to the cloud
- In this part of the exercise, you'll complete what's needed so the data from your Raspberry Pi reaches MindSphere
- First, define a physical asset according to the definition on the right side on MindSphere
- Data points are
  - MldHt20mA1TmpAct (°C)
  - MldHt20mA2TmpAct (°C)



Create your virtual Asset

- We have already prepared the aspects and types for your virtual asset



1. Create a new virtual asset, using the type *GenericInjectionMoldingMachine*
   - We've already prepared the type containing the aspect with the two variables
   - You just have to use this type
2. Link the data points of your physical asset with the variables of your newly create virtual asset

Prepare your Raspberry Pi

- Your Raspberry Pi comes prepared with real machine data
- Switch to the *Machine* flow
- Here, you'll find a quite complex logic
  - It basically reads machine data from a list of files
  - And sends it to MindSphere
  - Every file represents a full production cycle
- Feel free to check the logic
  - But you don't have to change anything
  - Just the two points on the right side

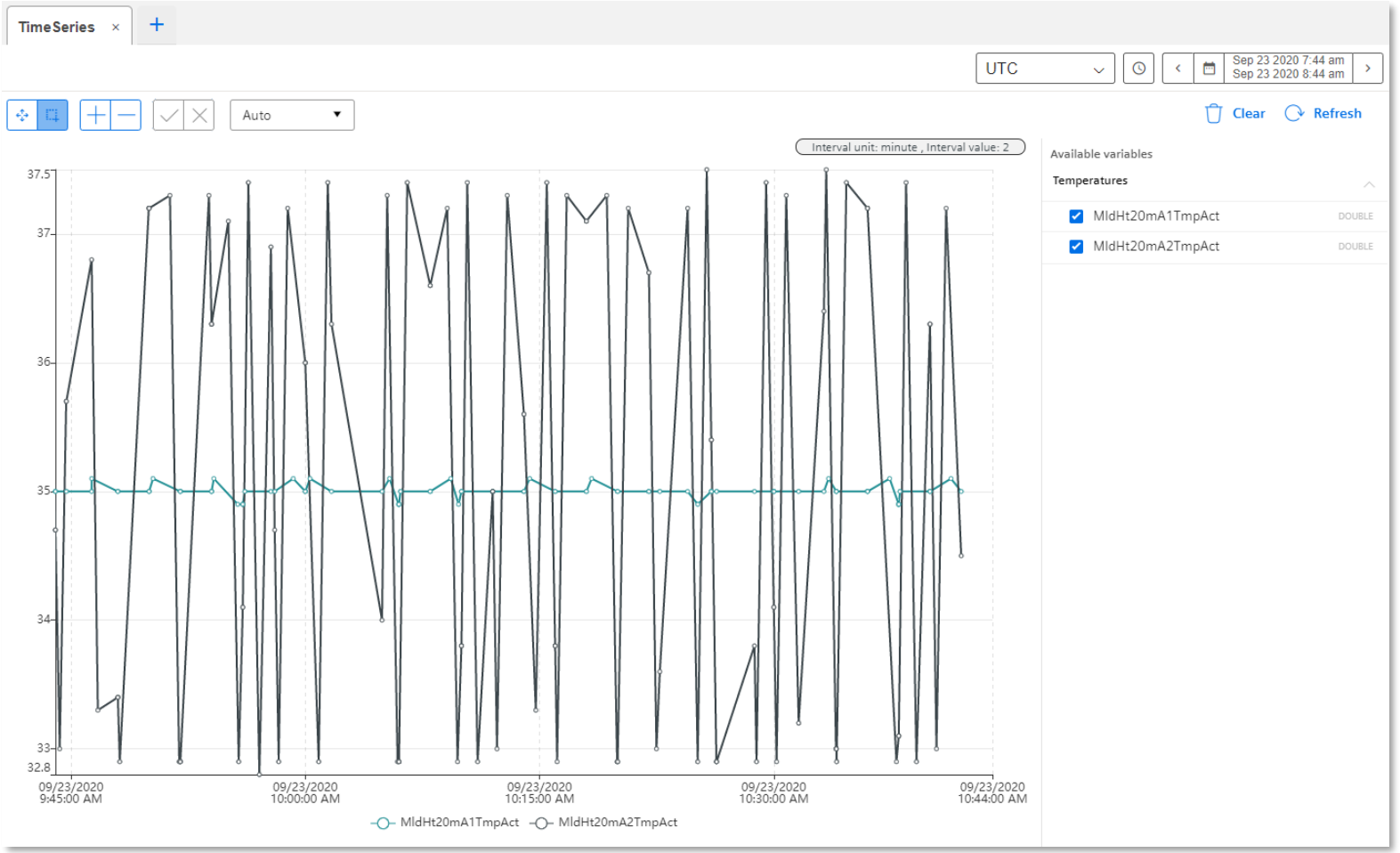
1. Define your data points in the *Configure MindSphere* node
   - Don't change the names, just assign the data point ids

```
const dataPoints = {
    MldHt20mA1TmpAct: "",
    MldHt20mA2TmpAct: ""
};

flow.set("dataPoints", dataPoints)

return msg;
```

2. Get the MindConnect node to work
   - Enter your shared secret

Test your Work with the Fleet Manager

1. Open the Fleet Manager
2. Check if you receive data
   - It may take a while until you see data points



Edge2Web

Data Viz made easy, well, kinda…

What is Edge2Web

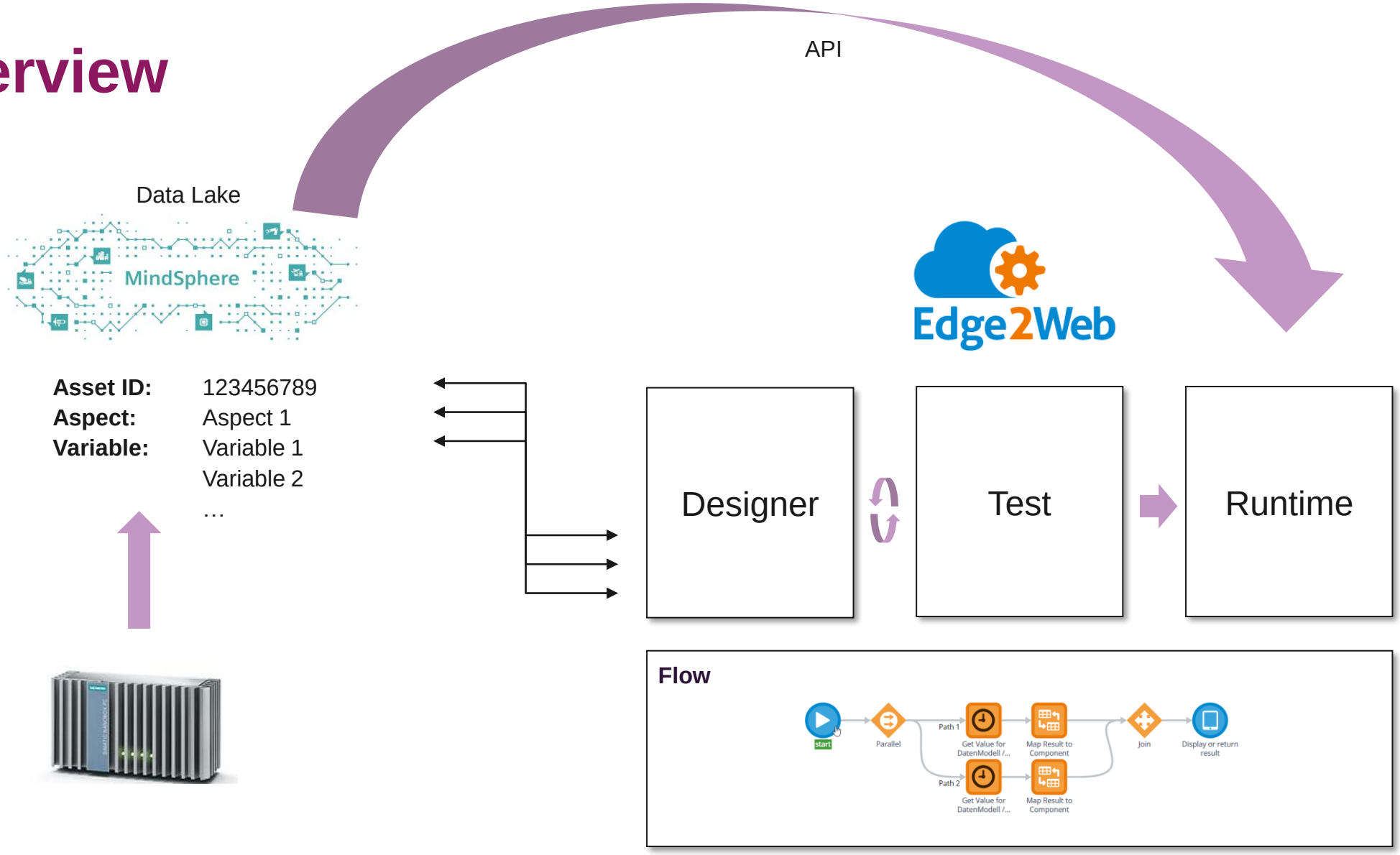
- https://hsr.eu1.mindsphere.io

- A simple way to
  - Visualize data
  - Build your own dashboard

- It offers building blocks to define your logic
  - But is JavaScript-based and can run custom code

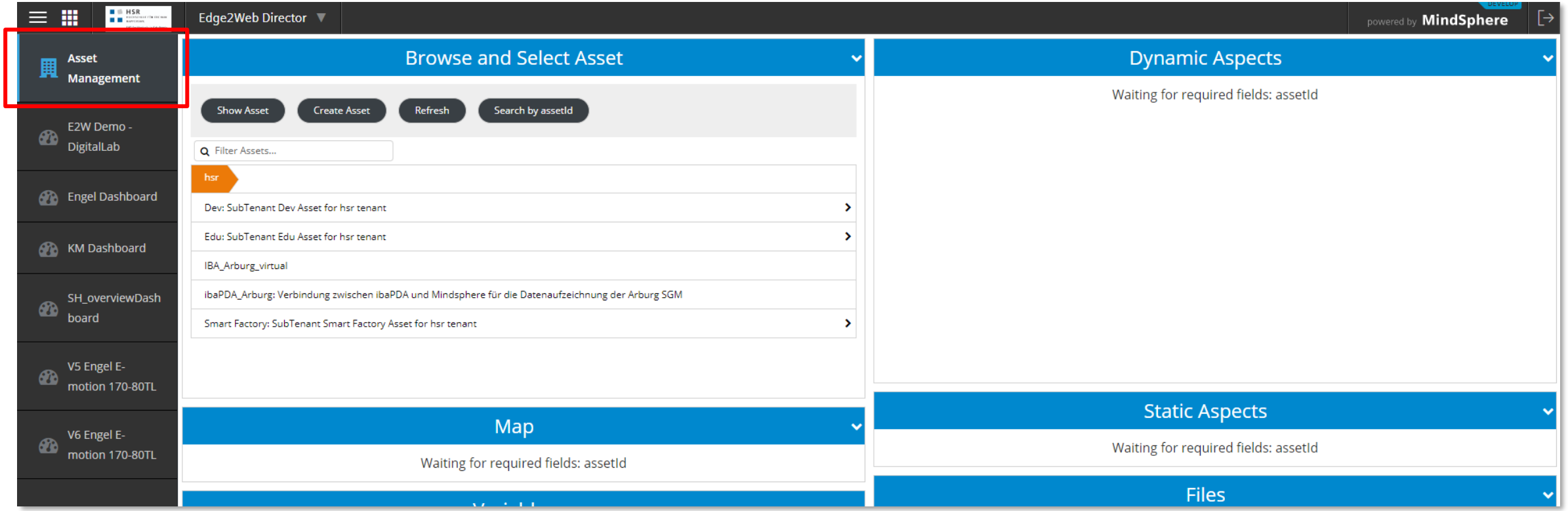
- It's integrated into the Siemens MindSphere

Overview



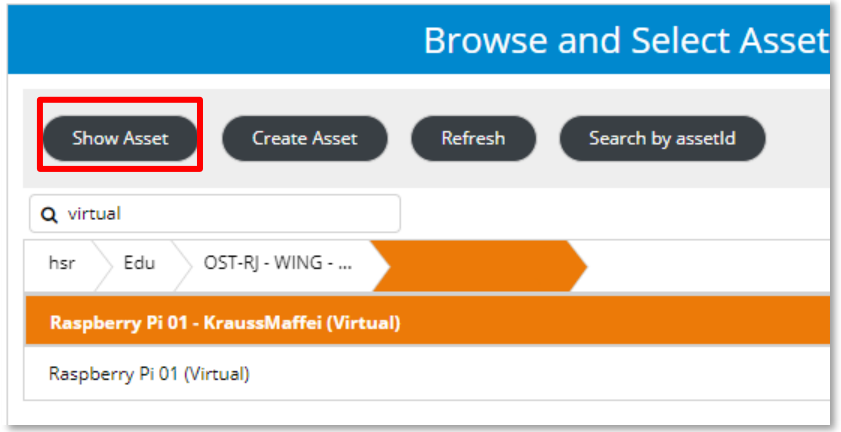
The Asset Management

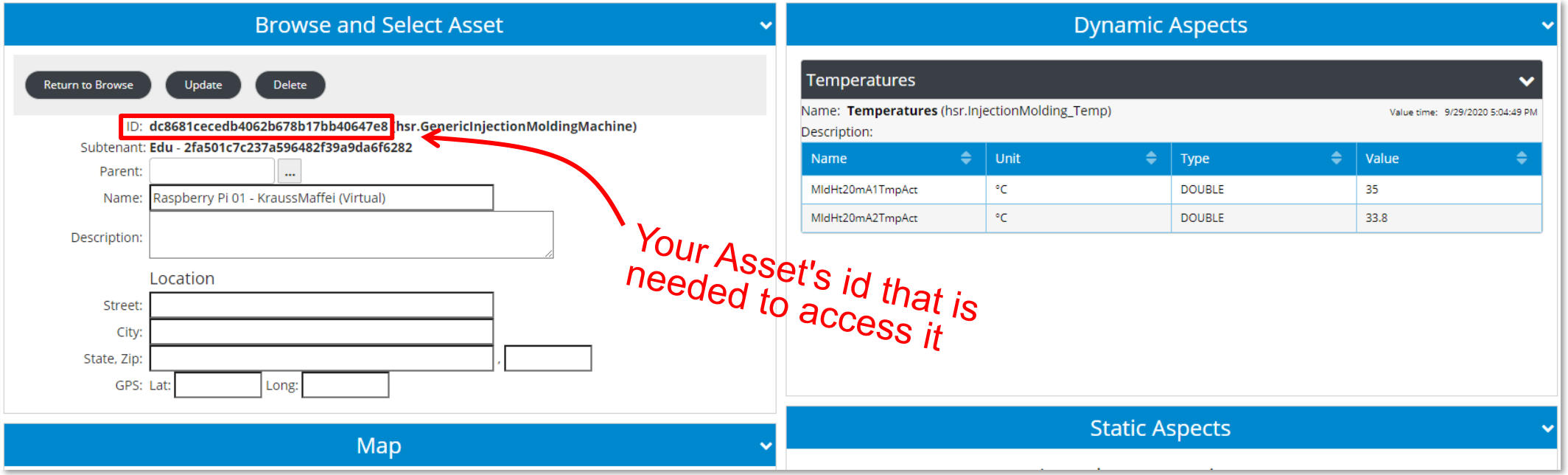
- A simple visualization of your asset



Using the Asset Management

1. Navigate to your Asset
2. Click *Show Asset* to see details
   - You can navigate back with *Return to Browse*

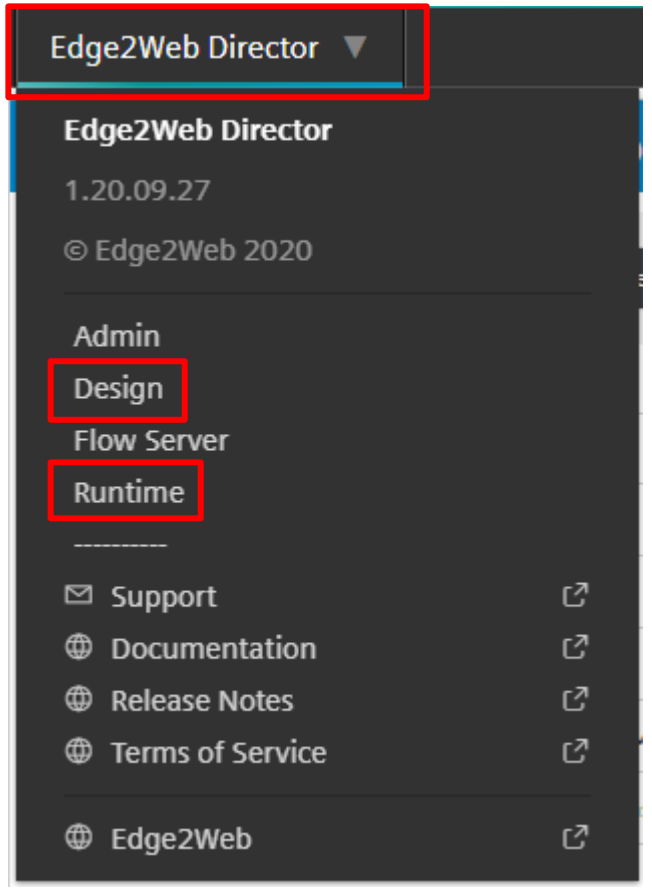




OST

How to switch between Views

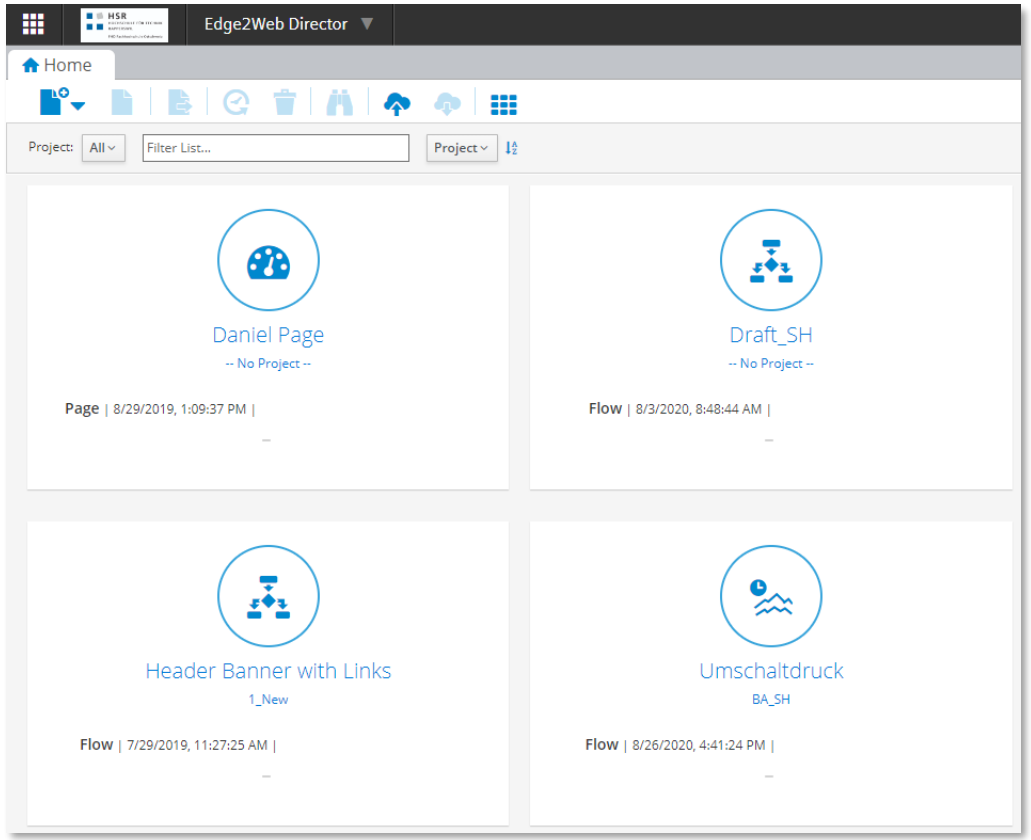
1. Click on *Edge2Web Director*
   - A detail view will open
2. Click on the view you want to open
   - The default is *Runtime*



The two relevant Views for you

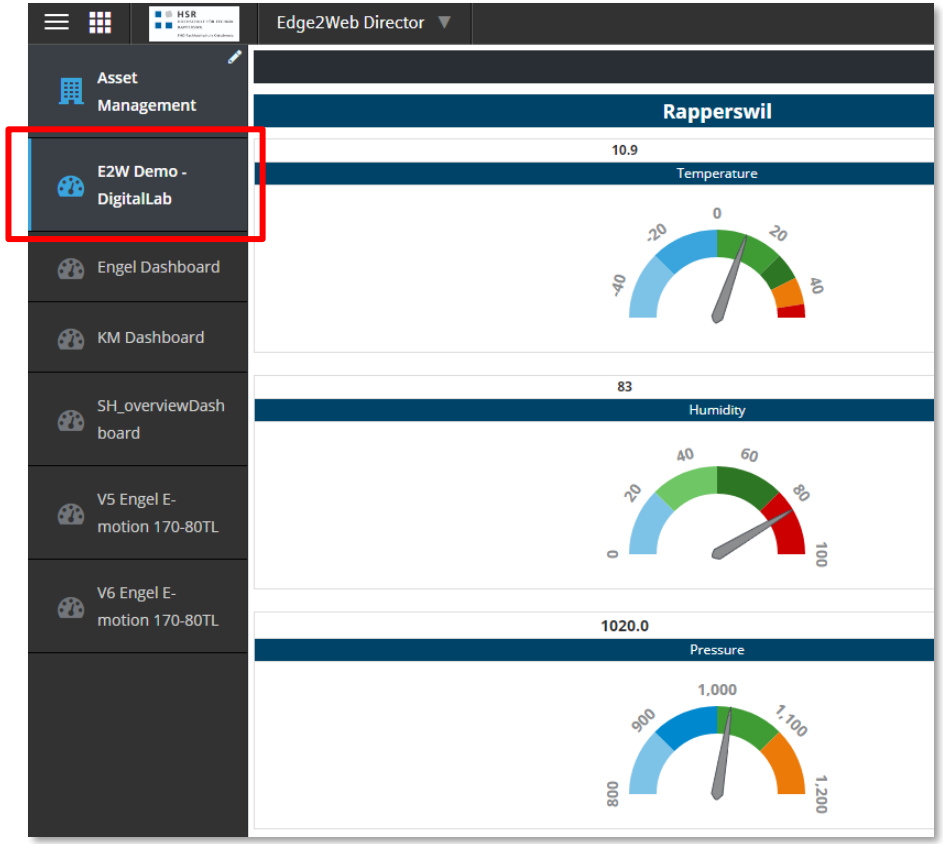
Design

- Used to create all your components



Runtime

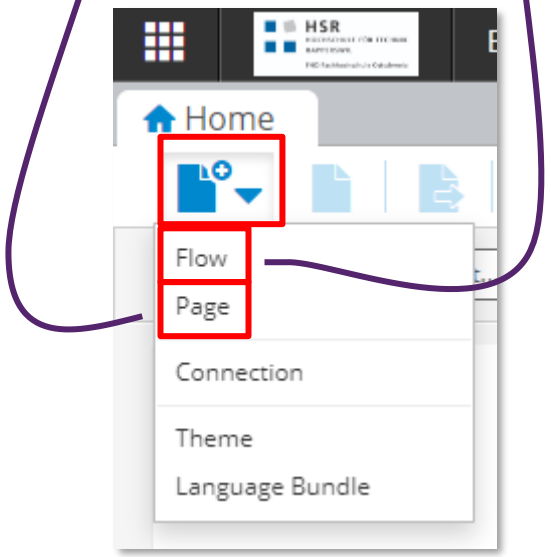
- Used to view your dashboards



The two relevant Types for you

Page

- Defines the layout
- Is your dashboard



Flow

- Get data from an asset
- Can manipulate data
- Defines how data is visualized
- Can also be as simple as a heading
  - Contains no actual logic

Naming Patterns

- Names must be unique through the whole application
  - And you cannot change the name after your page/flow was created
- Please use the following naming patterns

Project

- Edu_OST-HS21_*[Lastname][Firstname]*
  - E.g. Edu_OST-HS21_DoeJohnny

Page & Flow

- Edu_OST-HS21_*[Nr]_[Name]*
  - Nr = The number of your Raspberry Pi
  - Name = The name of your component
    - You are free in choosing the name

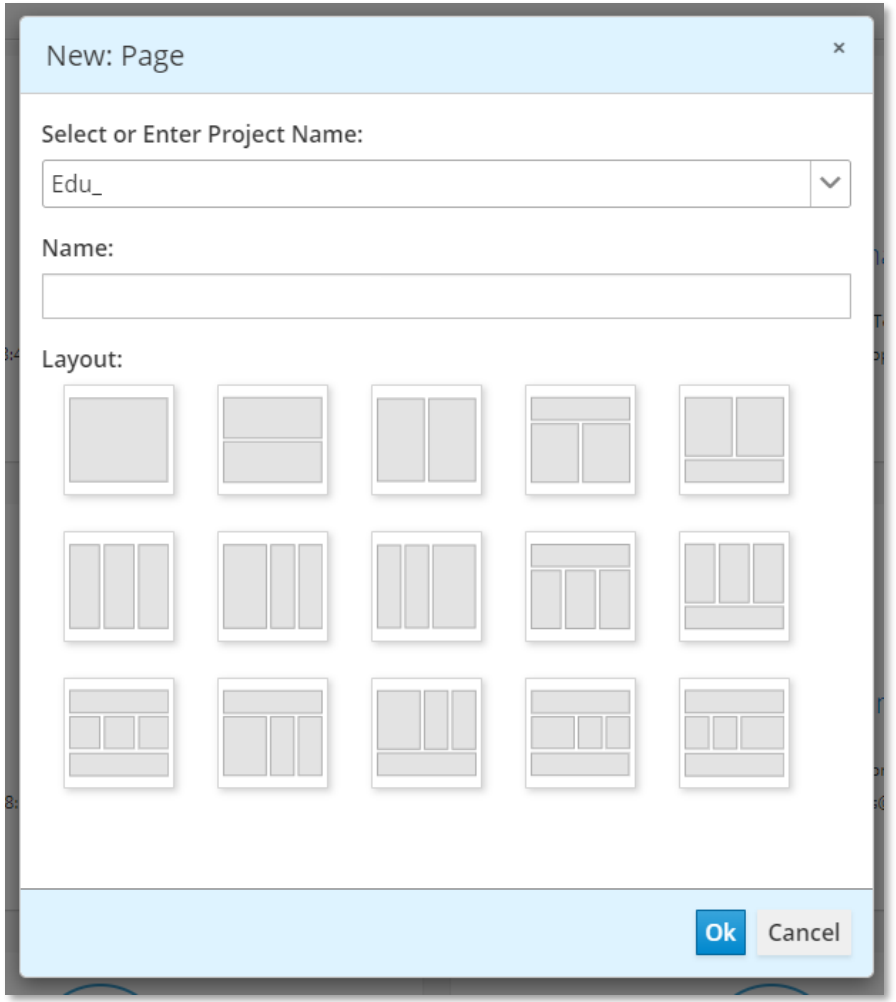
- Please follow these patterns, otherwise you might run into naming conflicts
- And you make our work easier when we have to clean up

Create a Dashboard

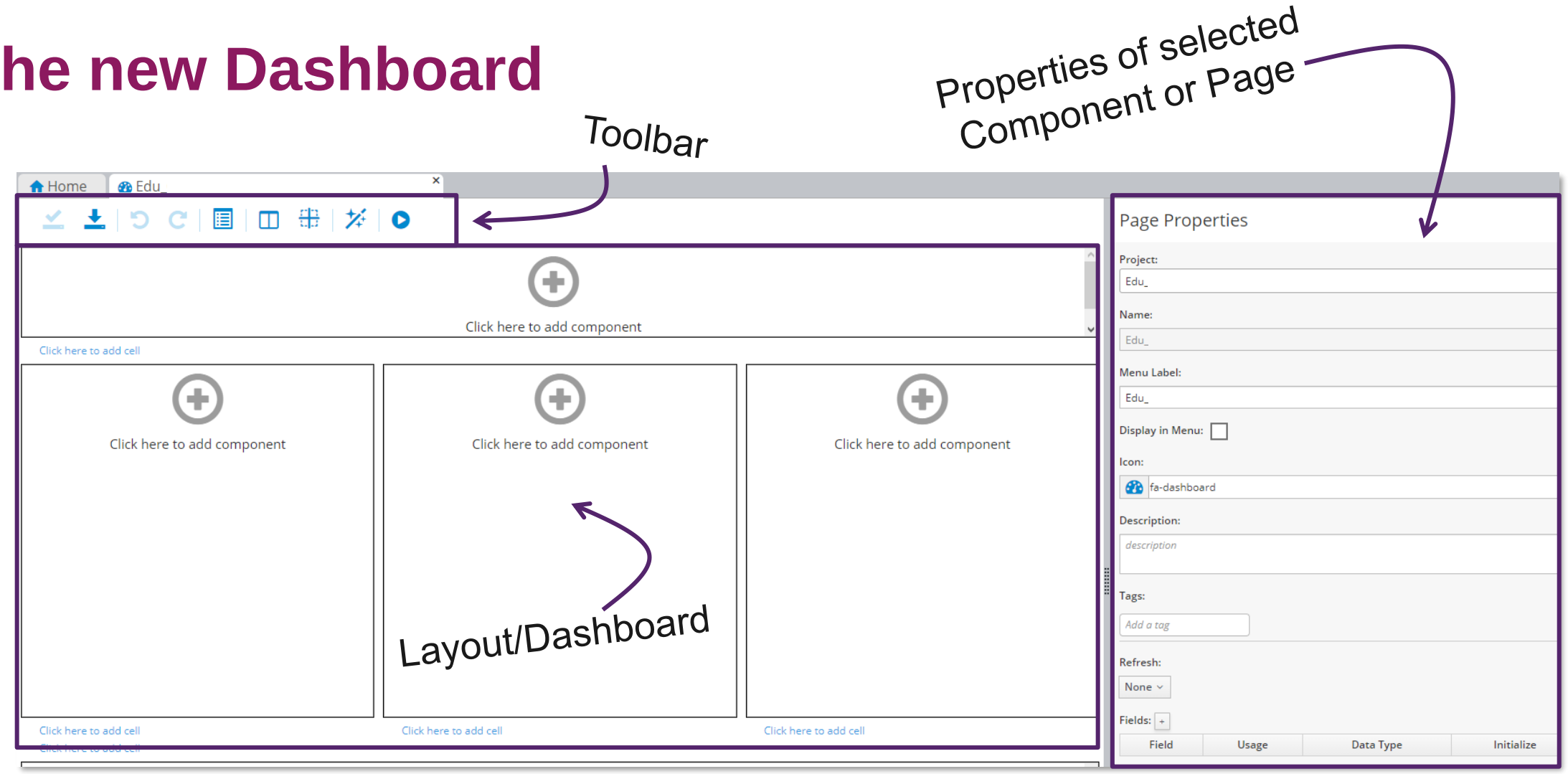
Getting started

A new Dashboard

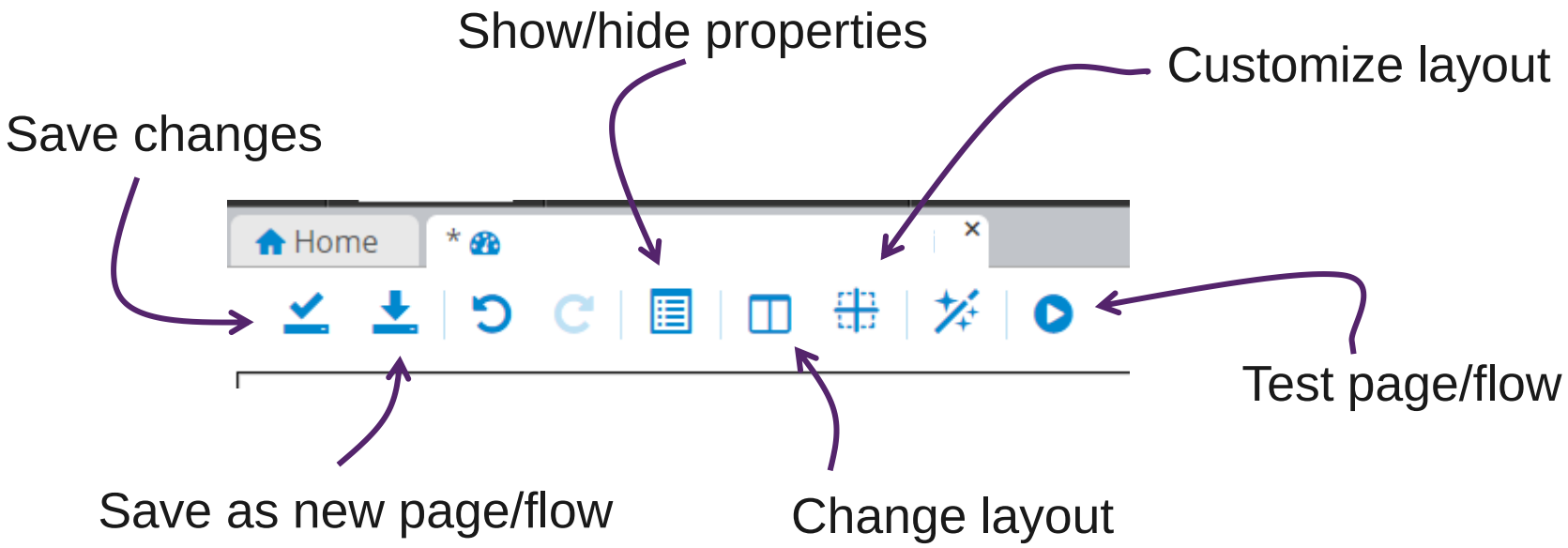
1. Create a new *Page*
2. Use a project name with the following pattern
   - *Edu_OST-HS21_[Lastname][Firstname]*
   - e.g. Edu_OST-HS21_DoeJohnny
   - Please always use this for all your components in E2W
3. Define a name for your dashboard
4. Choose a layout at your discretion
   - At the end, you need to show 6 tiles



The new Dashboard

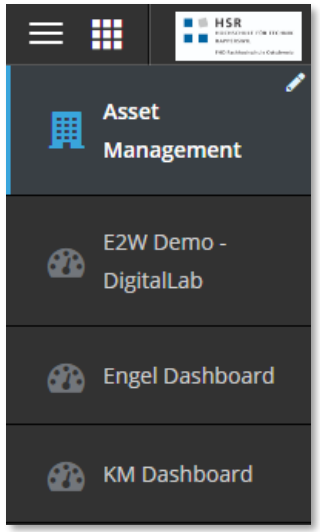


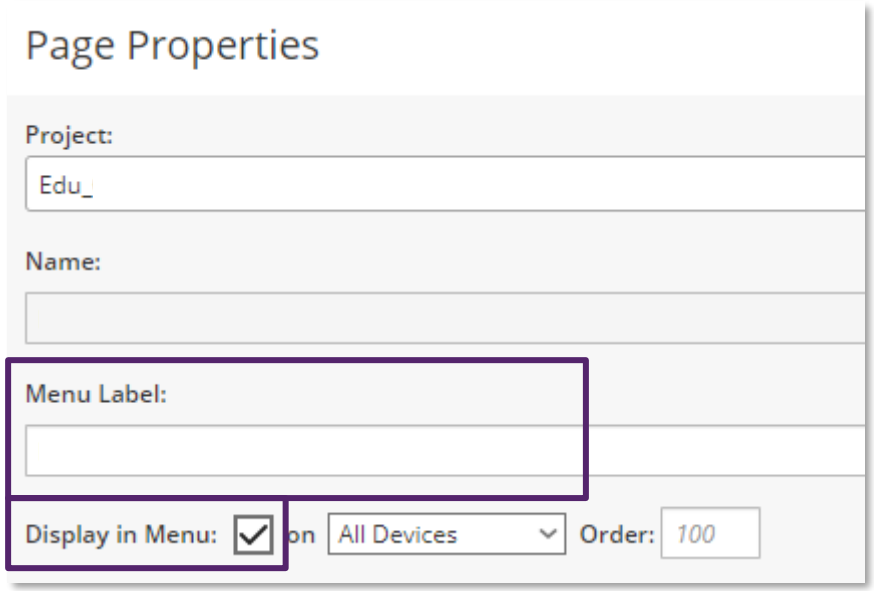
The Toolbar



Make it visible in the Runtime View

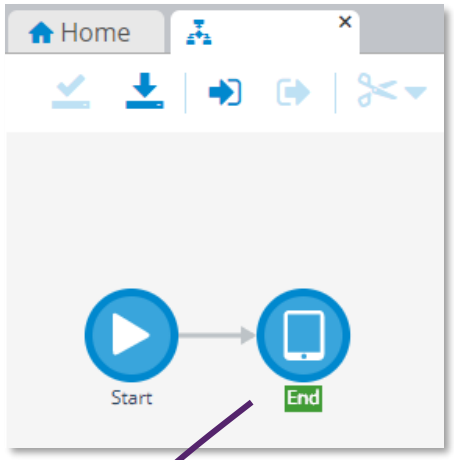
1. Enter the label that should be shown
2. Set the checkbox to *Display in Menu*
3. Save the changes
4. Switch to the *Runtime*
   - Your dashboard should be listed here



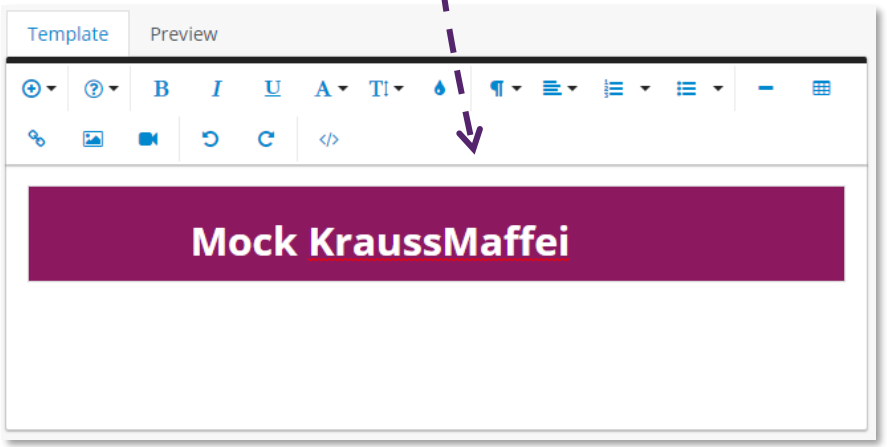


Add a Heading

1. Switch to the *Design* view
2. Select your project in the dropdown
3. Add a new *Flow*
4. Enter a name
5. Click *Ok*
   - A new tab will be opened
6. Select the *End* node
   - The *Properties* view will change and show you an editor
   - Move on to the next slide

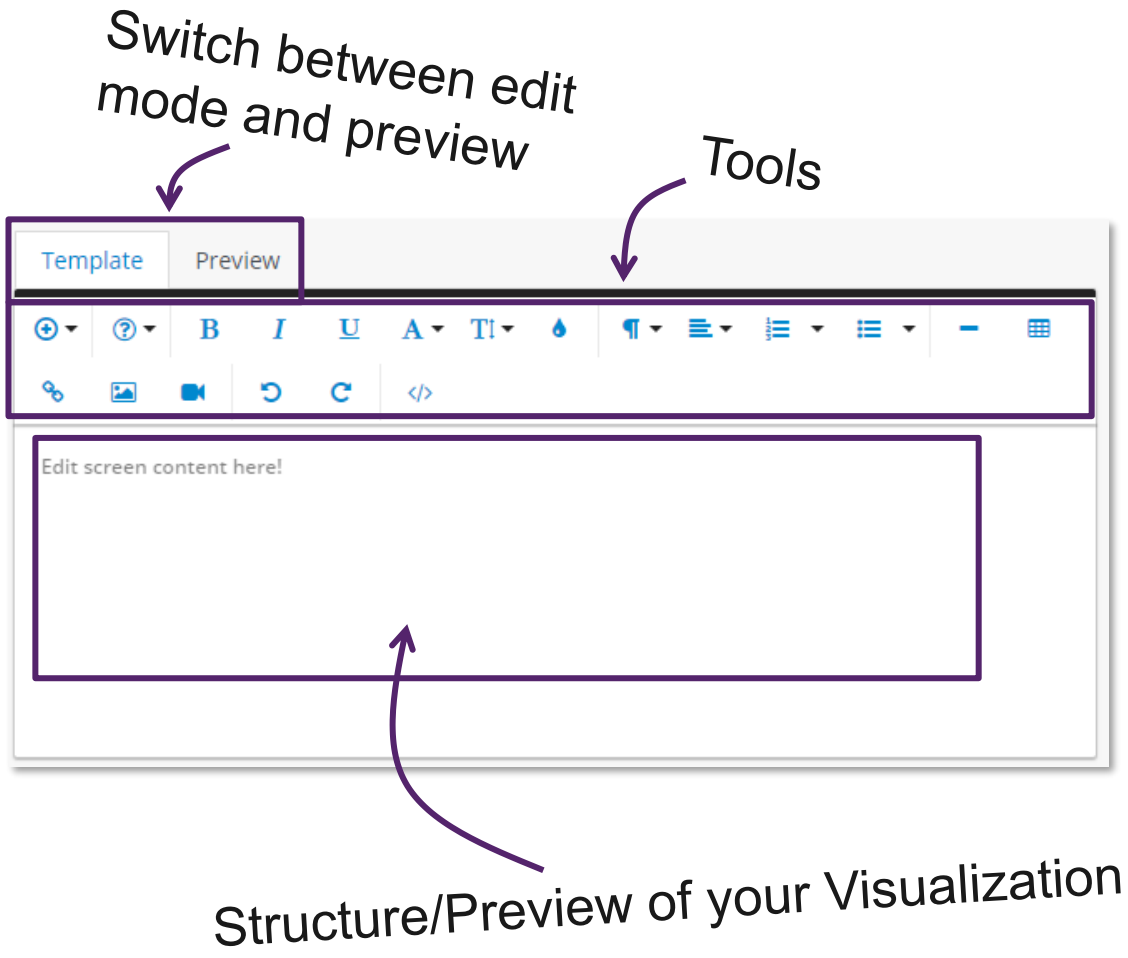


At the end, it will look something like that

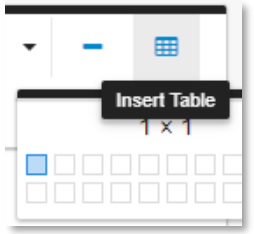


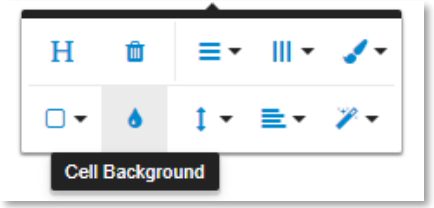
OST

Editor



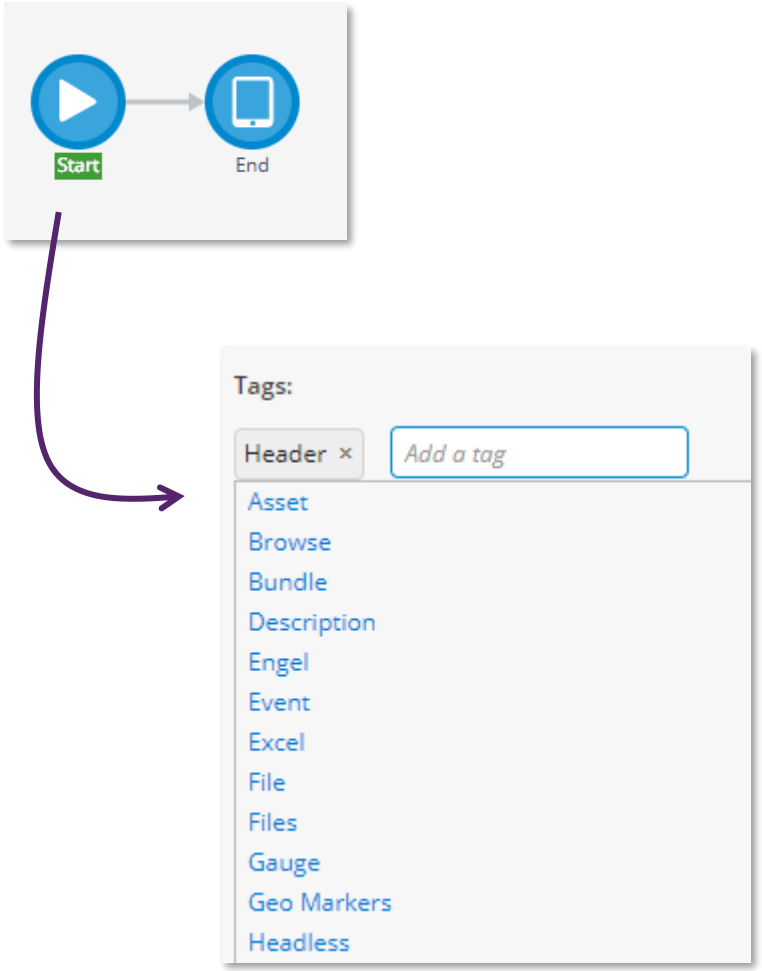
- You are completely free in defining your heading
  - The tools should be self explanatory
- You can get the heading from the previous slide by applying the following steps
  1. Create a table with a single cell
  2. Click into the cell to open the context menu
  3. Choose a background color
  4. Enter the text
  5. Format the text as bold, centered, white and change the size
     - You can also change the type to *Heading 1* if you want
  6. Save your changes





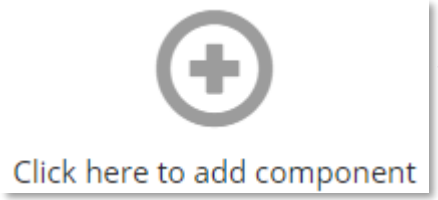
Tag your Flows

- To find your components easier later, you can set *tags* on your flows
- You can do this with the following steps
  1. Select the *Start* node
  2. Click into the *Tags* filed in the properties view
  3. Select an existing tag or enter a new one

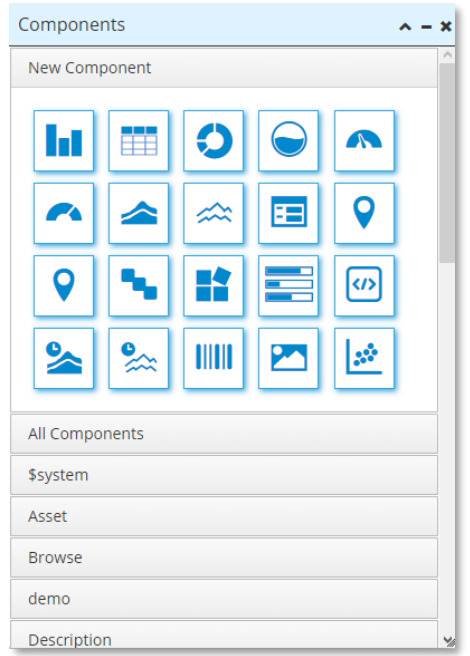


Add a Component to your Page

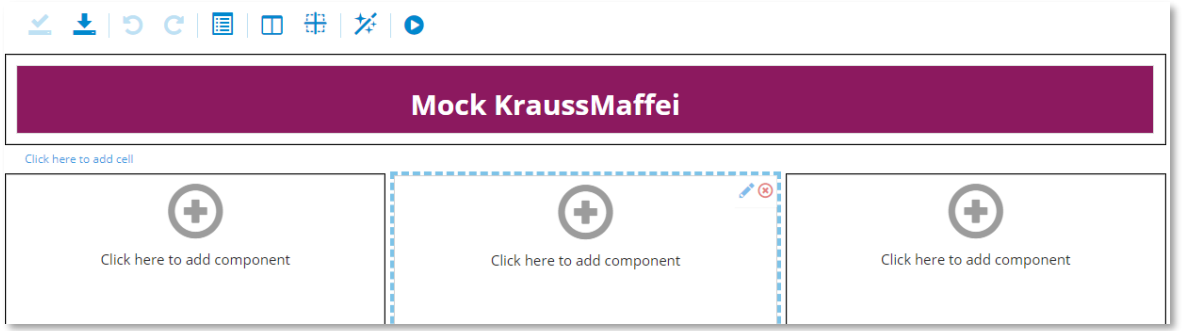
1. Switch back to your dashboard
2. Click on the cell you want to add your heading to



3. A menu will open where you can add a new component or select an existing one
   - If you have tagged your flow, you can open the tag from the menu and look for your flow
   - Otherwise, you can go to *All Components* and search for your flow



4. Drag & drop your component to the cell you want to have it
5. Save your changes



Visualize the Temperature

Let's turn up the Heat!

Create the Flow

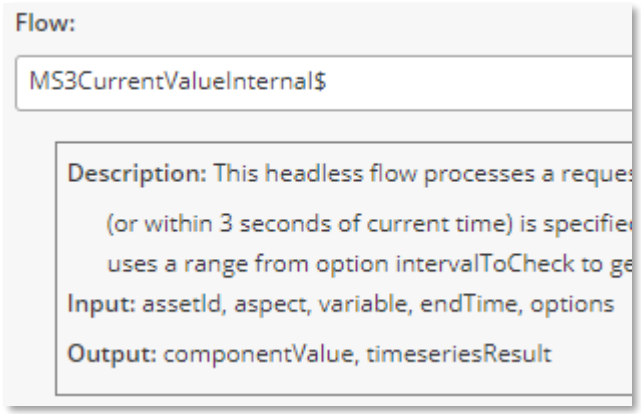
- In this step, you'll visualize a single value

1. Create a new flow
2. Click on the *Start* step
3. Click on *Append Step*  to insert a new step
   - An orange step will be added in between the *Start* and *End* steps
4. In the properties view, change the name to something meaningful (e.g. Get Temperature)
5. Change the *Step Type* to *Call*
6. Select *MS3CurrentValueInternal$* as *Flow*

➢ Go to the next slide

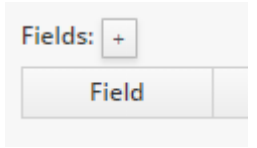
Add Parameters to your Flow

- You see that the selected flow expects some input parameters and will return some results



- You now have to define these variables in our flow so you can provide their values to the flow

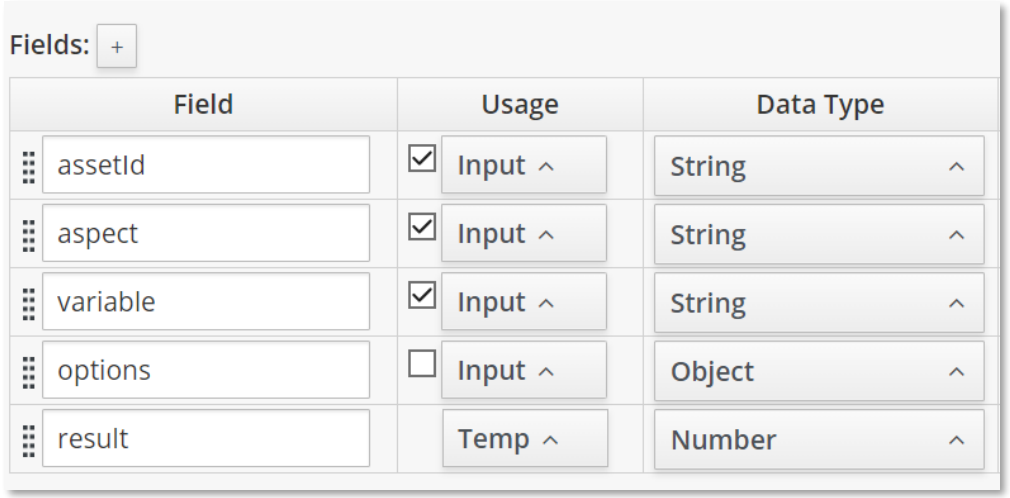
1. Select the *Start* node
2. Select the + next to *Fields*
   - A new entry gets populated



3. Name the field *assetId*, change the usage to *Input* and select the checkbox to mark it as required
4. Repeat the previous steps for the following fields as well
   - aspect (Input, String, required)
   - variable (Input, String, required)
   - options (Input, Object)
   - result (Temp, Number)

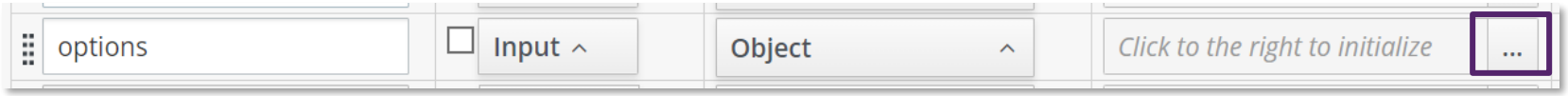
The Flow Parameters

- Your fields should now look like

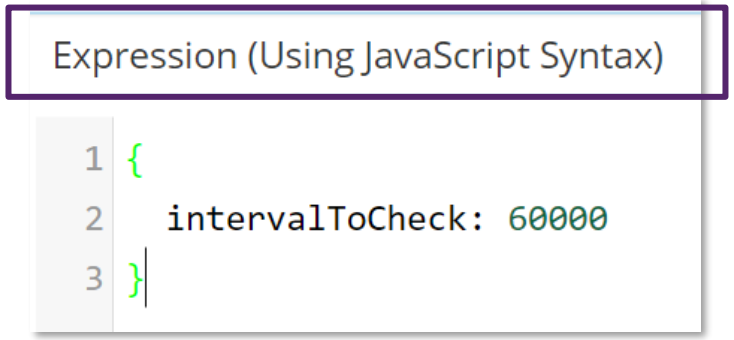


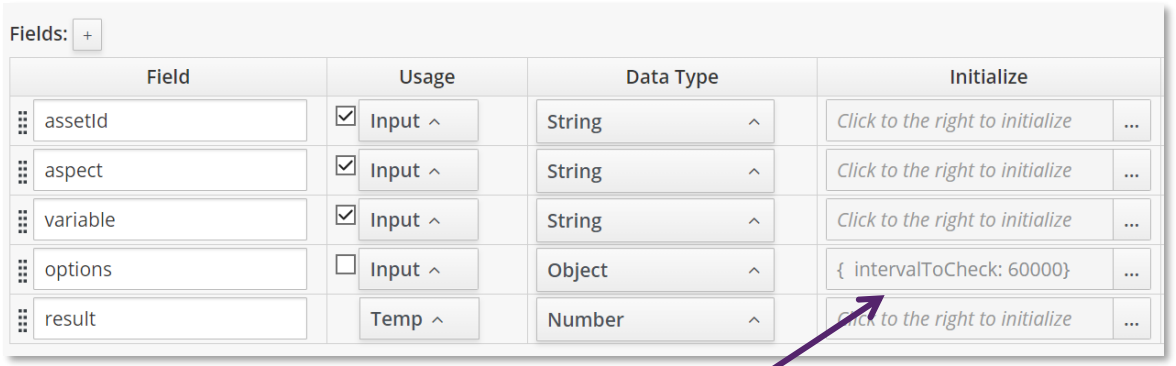
Set default Options

1. Click on the … to open the *options* dialog



2. In the dialog, define the following object
   - Ensure that you select *Expression* on the top

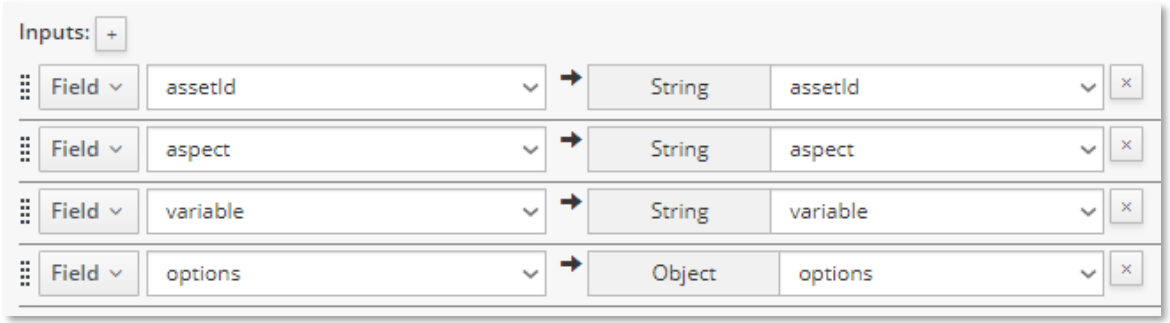




3. Close the dialog
   - You should see your defined object in the *Initialize* column

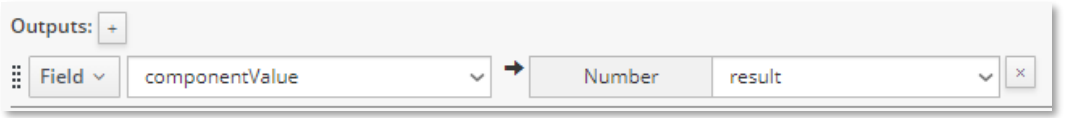
Configure the Call Step

1. Select the *Call* step you added previously
2. Click the + next to *Inputs*
3. Define the mappings as follows



- On the left-hand side are your input parameters you defined in the previous step
- On the right-hand side are the input parameters the *Flow* of the *Call* step expects

4. Click on the + next to *Outputs*
5. Define the mappings as follows

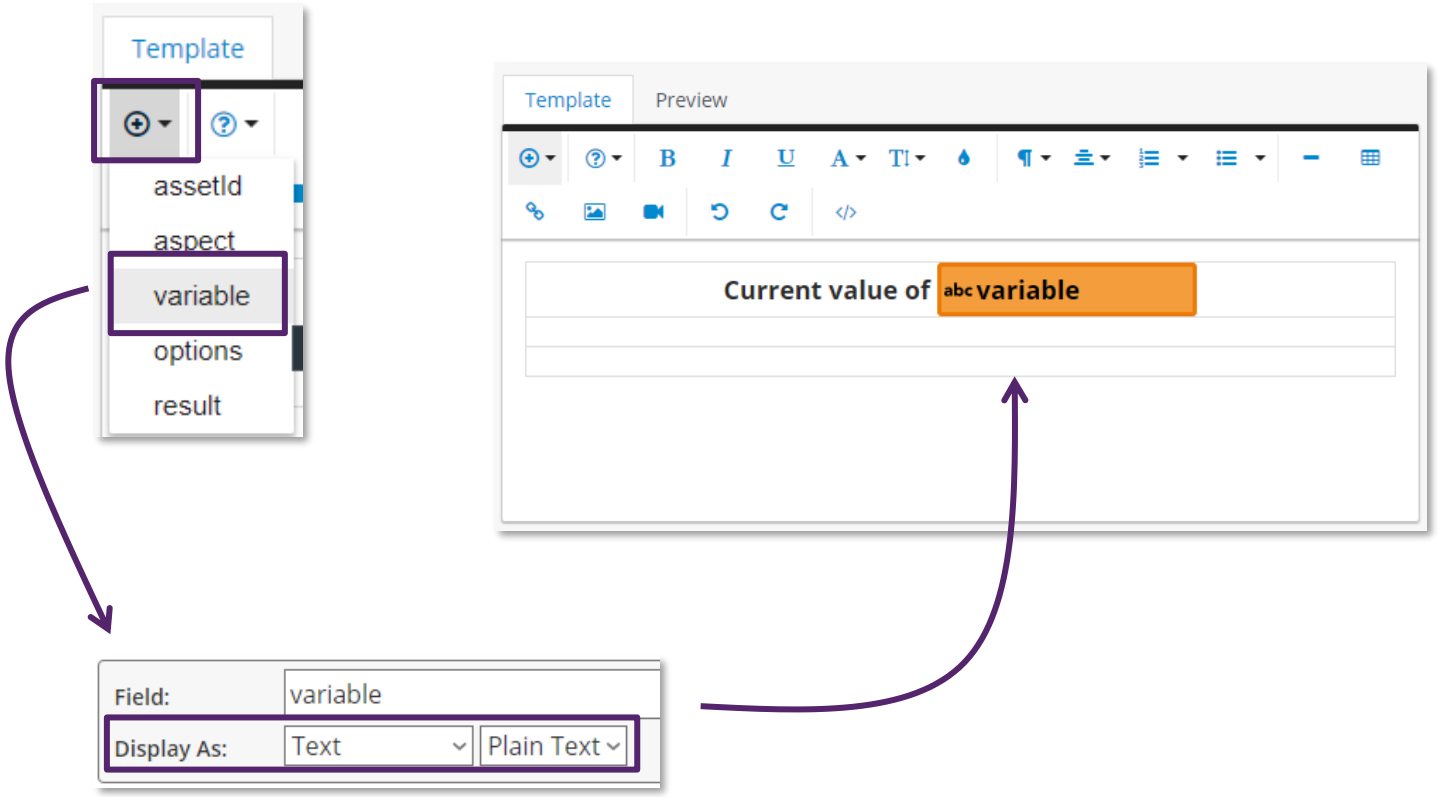


- On the left-hand side are the output variables of the *Call* step
- On the right-hand side are your fields you specified in the previous step

- Note on *options*
  - The object we defined in the previous step defines that the value returned by the *Call* step must be within an interval of the last 60 seconds
  - Otherwise, the returned value will be empty

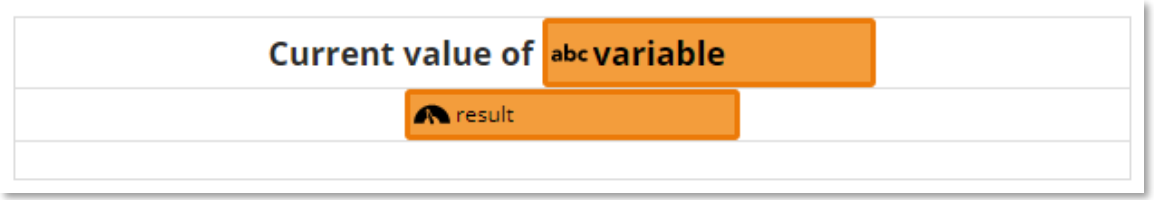
Defining the Visualization

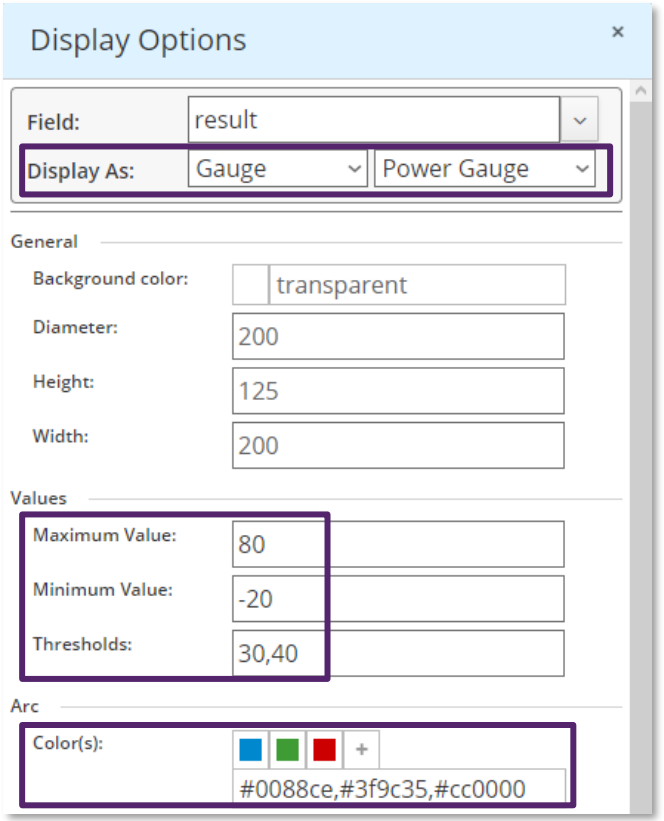
1. Select the *End* step
2. Add a table with 3 rows
3. Select the first row
4. Enter *Current value of*
5. Click on *Insert Field*
6. Select *variable*
   - A dialog will open
7. Change *Display As* to *Text*
8. Close the dialog

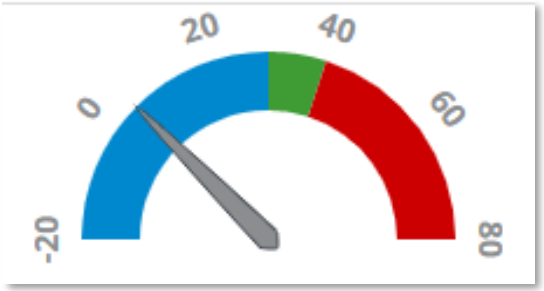


Visualize the current Temperature

1. Select the second row
2. Add *Insert Field* with *result*
3. Configure the field as shown on the right
   - If you specify multiple colors, you always have to define thresholds where the colors should change
   - As a consequence, you have to define one threshold less that your number of colors
4. Your *Template* should look like the following

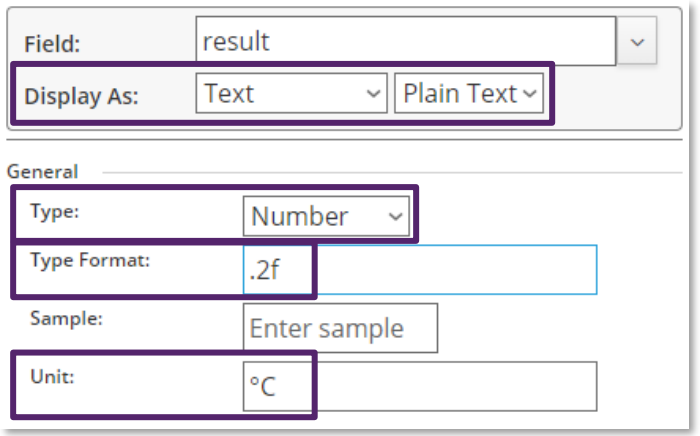


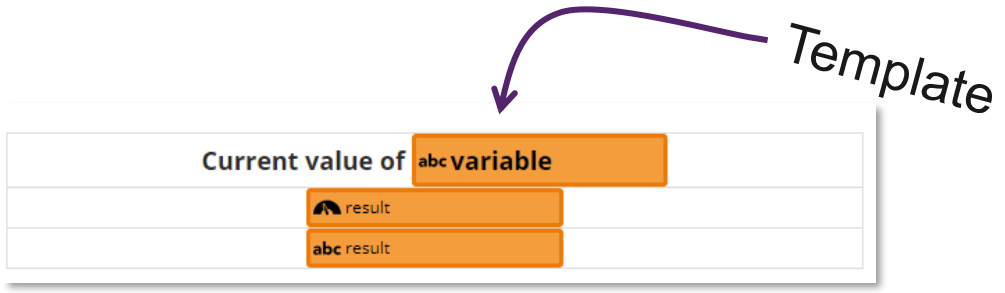


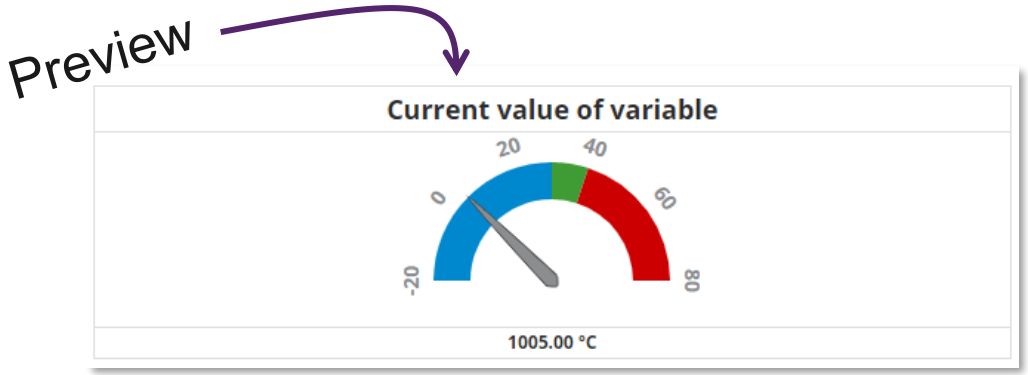


Showing the current Temperature

1. Select the last row
2. Add *Insert Field* with *result*, again
3. The value should be shown as *Text*
4. Change the *Type* to *Number*, and change the *Type Format* and *Unit* to the appropriate format
5. Your control should now look like the following

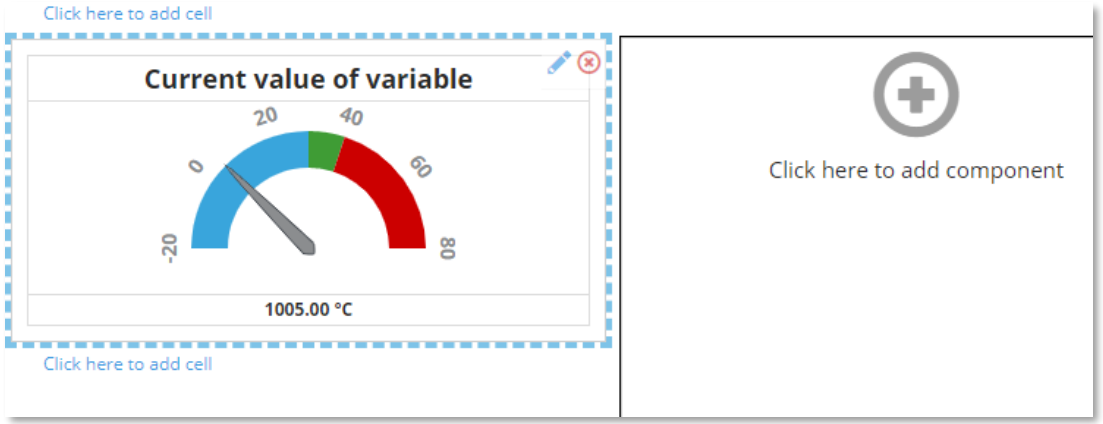






Adding Temperature to the Dashboard

1. Add your flow to a cell



2. Open the *Properties* view (if not already opened on the right-side of your screen)

3. Select your component
   - The *Page Properties* become *Cell Properties*

4. Change the *Component Inputs* of *assetId, aspect* and *variable* from *Component* to *Cell*
   - Doing this, we can configure the flow individually

5. Set the *assetId* to the id of your asset that receives the temperature data from the Raspberry Pi

6. Set the *aspect* to *Temperatures*, since this is the name of the used aspect

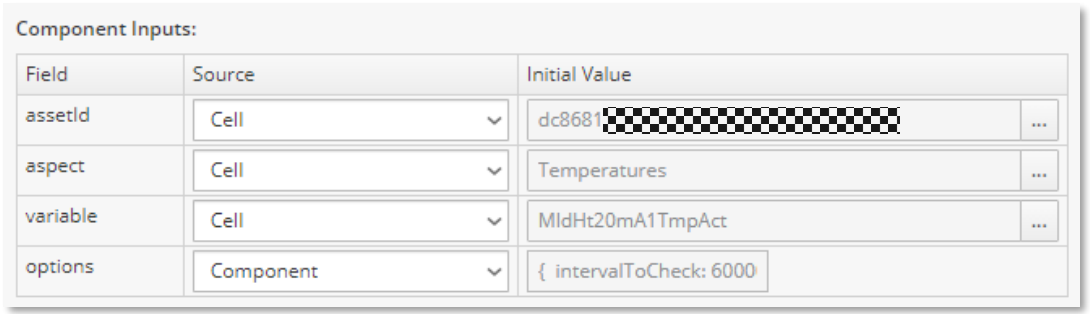
7. Set the *variable* to *MldHt20mA1TmpAct*

➢ Go to the next slide

OST

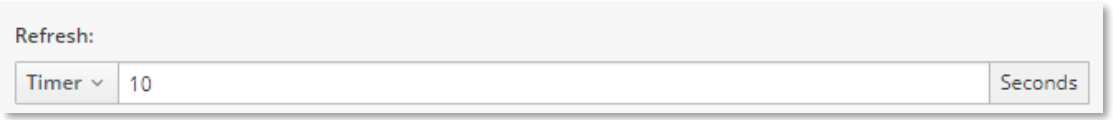
Test your Dashboard

- Your cell should be configured like the following



1. Switch to the *Page Properties* back 

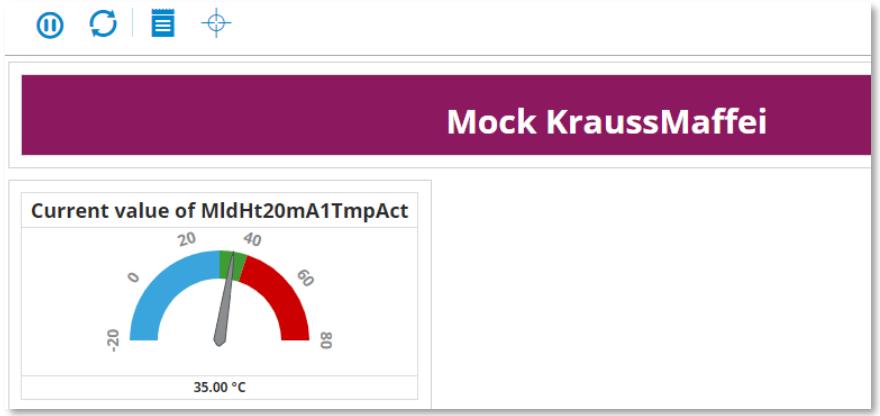
2. Set a *Refresh Timer*



3. Save your changes

4. Test your dashboard by clicking *Test Page* in the toolbar

   - A new tab will open, showing a preview of your dashboard



- Congratulations, you are able to visualize data in Edge2Web that was sent from your Raspberry Pi to the Cloud

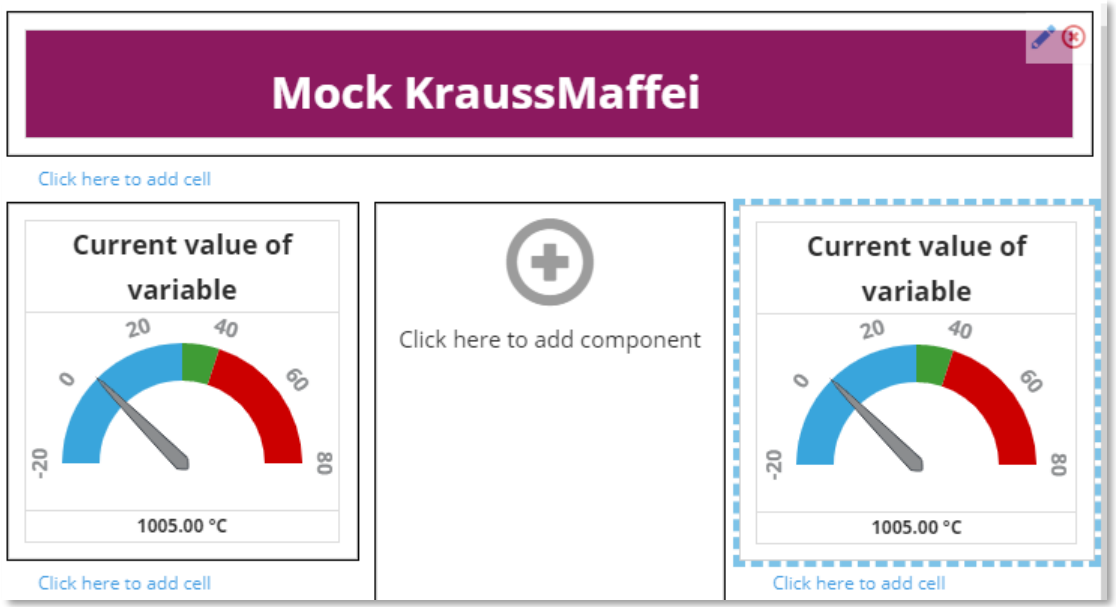
Reuse your Component

- In the previous step, you just configure the component to read a specific variable of your asset
- Now, use the same component, but for the other temperature your Raspberry Pi is sending

Component Inputs:

Field	Source	Initial Value
assetId	Cell	dc8681c[redacted]
aspect	Cell	Temperatures
variable	Cell	MldHt20mA2TmpAct
options	Component	{ intervalToCheck: 6000

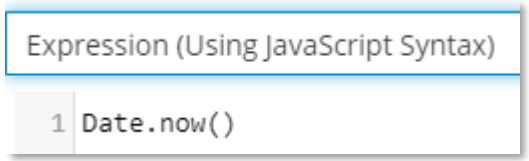
- Your dashboard now might look like



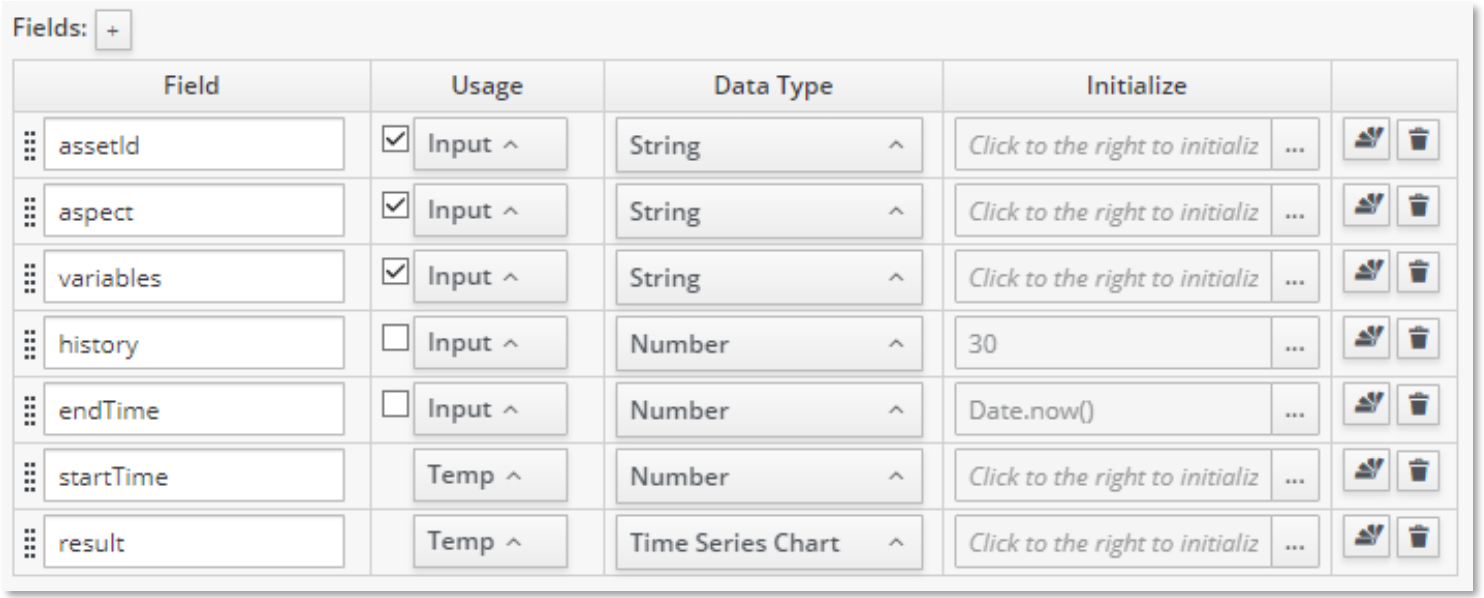
- It's up to you where you want to position the other temperature gauge
- When you test your dashboard, the values should change over time

Create a new Flow for the Time Series

- In the previous example, you just visualized a single value
- Now, you are going to show the history to see the changes over time
- Note that you should change the type of the initialization value of *endTime* to *Expression*



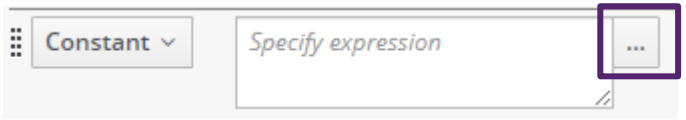
1. Create a new *Flow*
2. Define the fields as follows



OST

Calculate the Start Time

1. Add a new step to the flow
2. Define its *Step Type* as *Assign*
   - *Assign* is a generic step that you can use to execute JavaScript
3. Open the editor for the logic

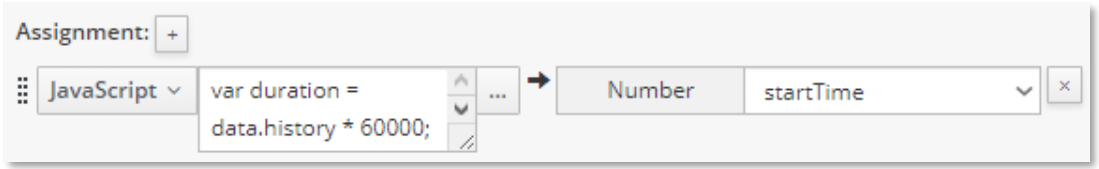


4. Change the type to *JavaScript Function*
5. Enter the following code

JavaScript Function

```
function(context, data) {
1 var duration = data.history * 60000;
2 return data.endTime - duration;
```

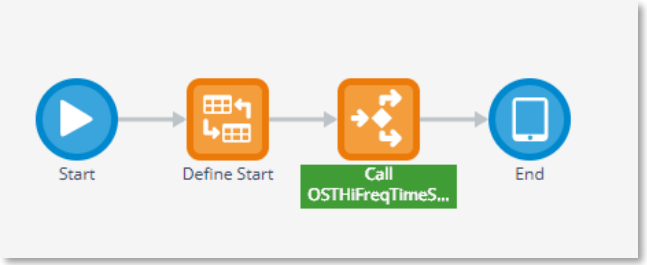
6. Close the editor
7. Assign the value to the *startTime* field

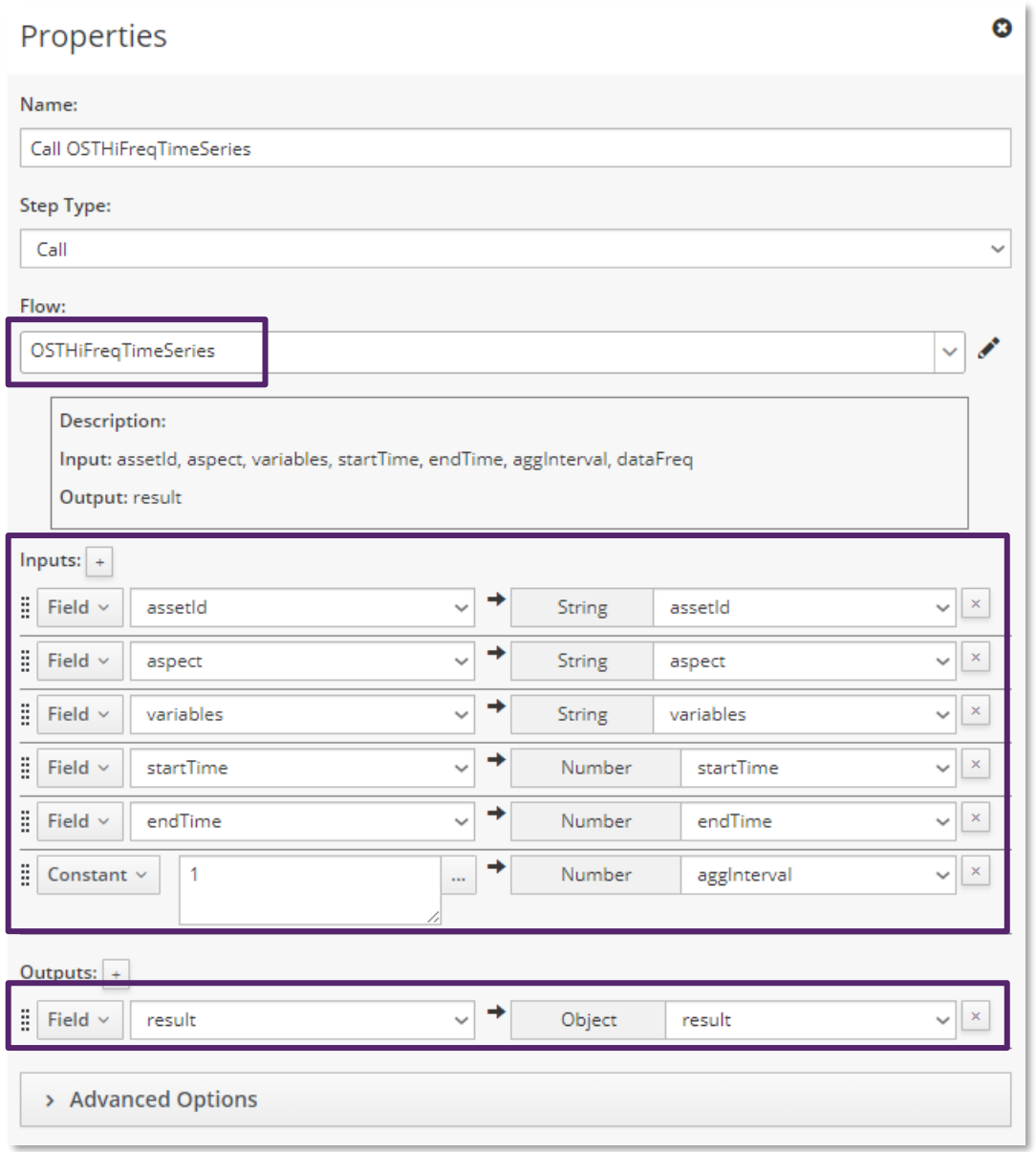


- We assume that the history is provided in minutes (default is 30)
- Since we need microseconds, we calculate the duration and subtract it from the end time

Get the Time Series

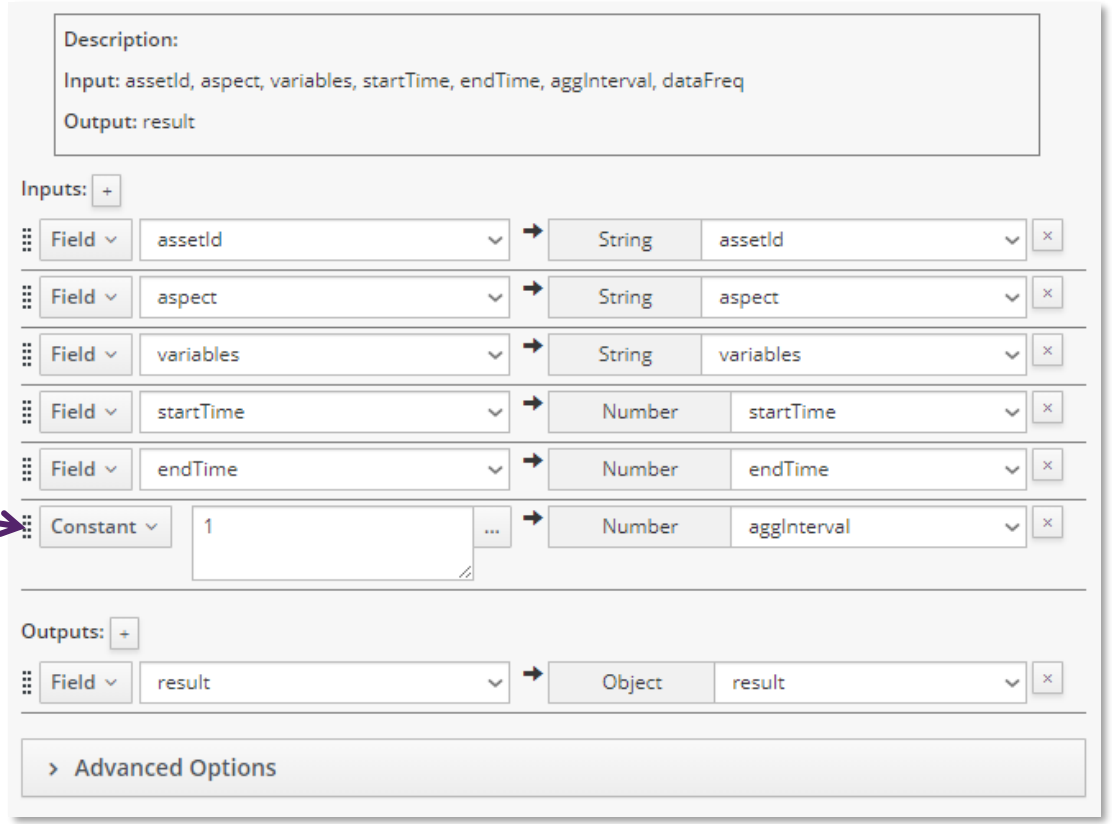
1. Add a new step
2. Change its *Step Type* to *Call*
3. Configure it as shown on the right
   - You are free to choose another *Name*
4. Your flow should now look like the one below





Remarks: OSTHiFreqTimeSeries

- **Please note:** OSTHiFreqTimeSeries is an experimental implementation
  - The algorithm might not work properly in every scenario
- The given time series algorithm cannot handle data with a frequency of 100 Hz
  - Thus, we implemented our own algorithm to collect and aggregate this data
  - Use *aggInterval* to define how many values should be aggregated
    - The value's unit is in s (seconds)
      e.g. *aggInterval = 1* means 100 data points get reduced to 1 data point

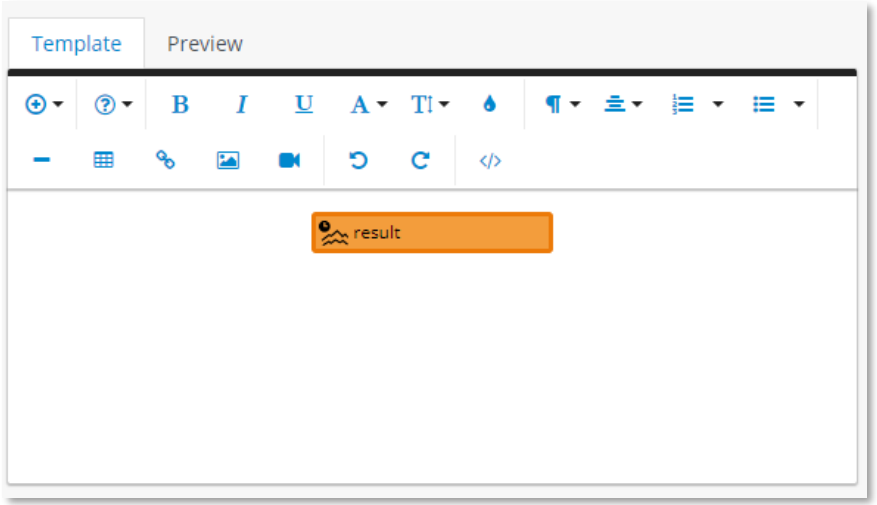


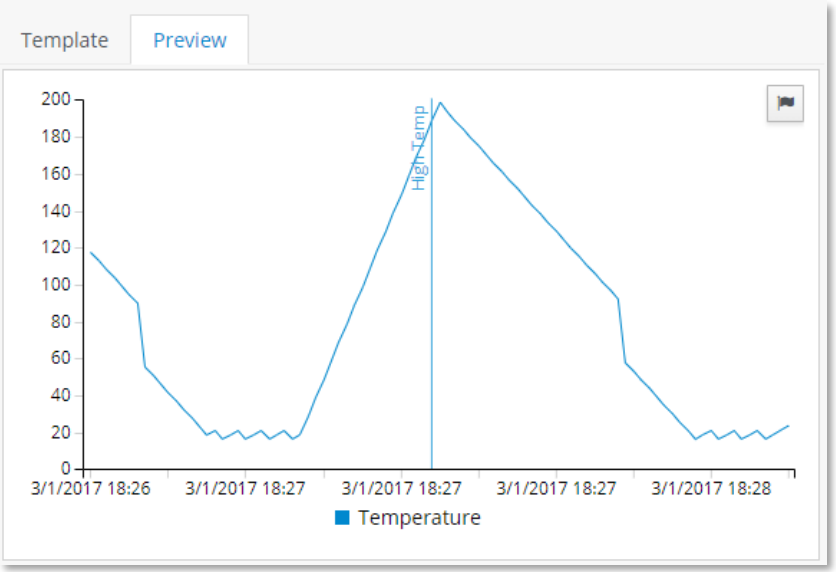
OST

Define the Visualization of the Time Series

- We'll keep it pretty simple for this one

1. Select the *End* node
2. Call *Insert Field* for *result*
   - For this example, we just keep the defaults
   - But feel free to change them as you want



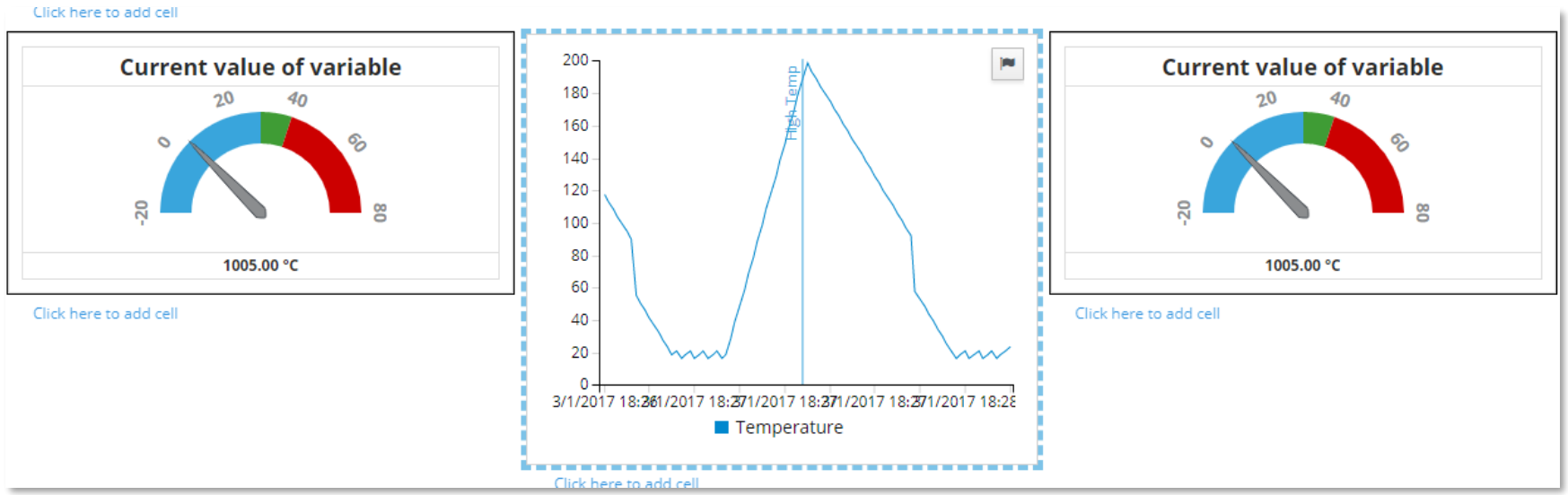


Extend your Dashboard with the Time Series

1. Switch to your dashboard
2. Add the newly created component
3. Configure the component to show two temperatures used before

Component Inputs:

Field	Source	Initial Value
assetId	Cell	dc8681cece[redacted] ...
aspect	Cell	Temperatures ...
variables	Cell	MldHt20mA1TmpAct,MldHt20mA2TmpAct ...
history	Component	30
endTime	Component	Date.now()



Today's Objectives

- ✓ Create physical and virtual Assets
- ✓ Send Sensor Data to the Cloud
- ✓ Receive Machine Data on your Asset
- ✓ Create a Dashboard for your Mock Machine (Raspberry Pi)
  - ✓ Visualize the Temperatures
  - ✓ Layout your Tiles