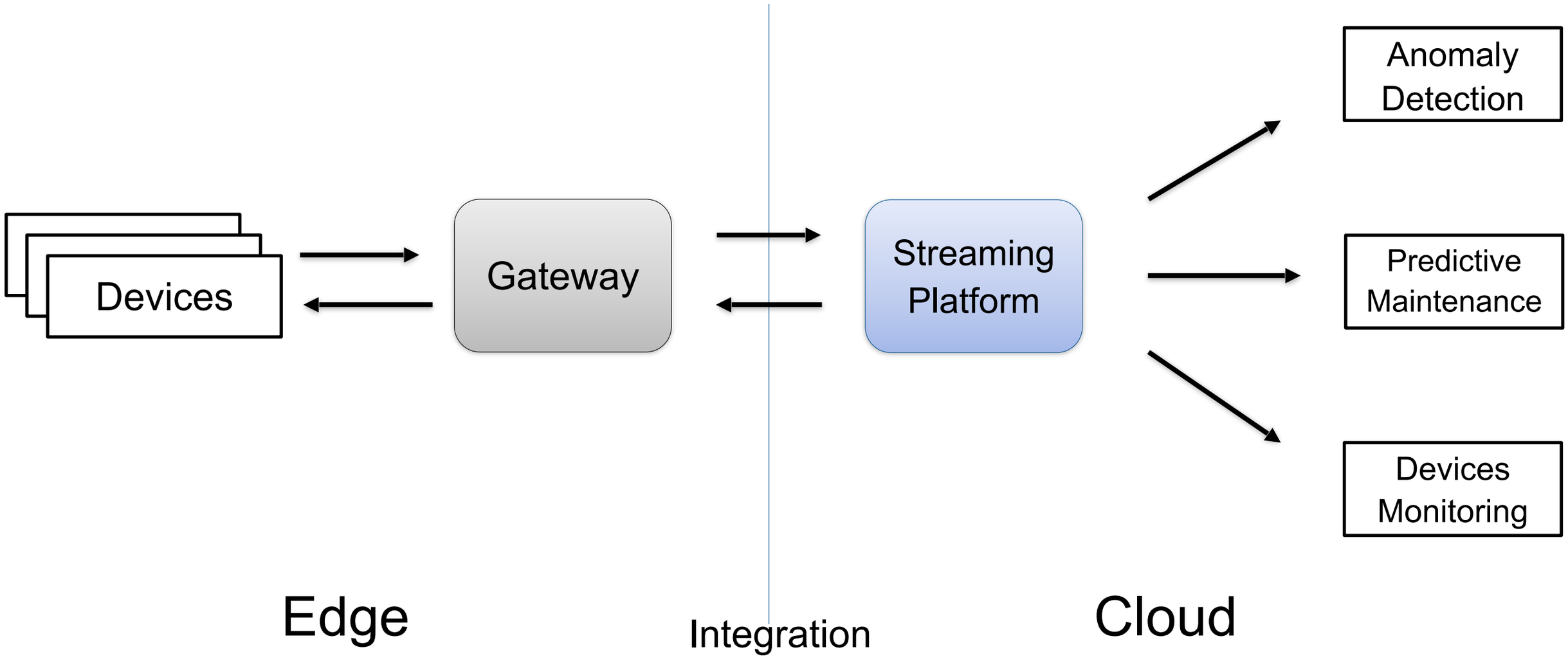
IoT Platform

Topics

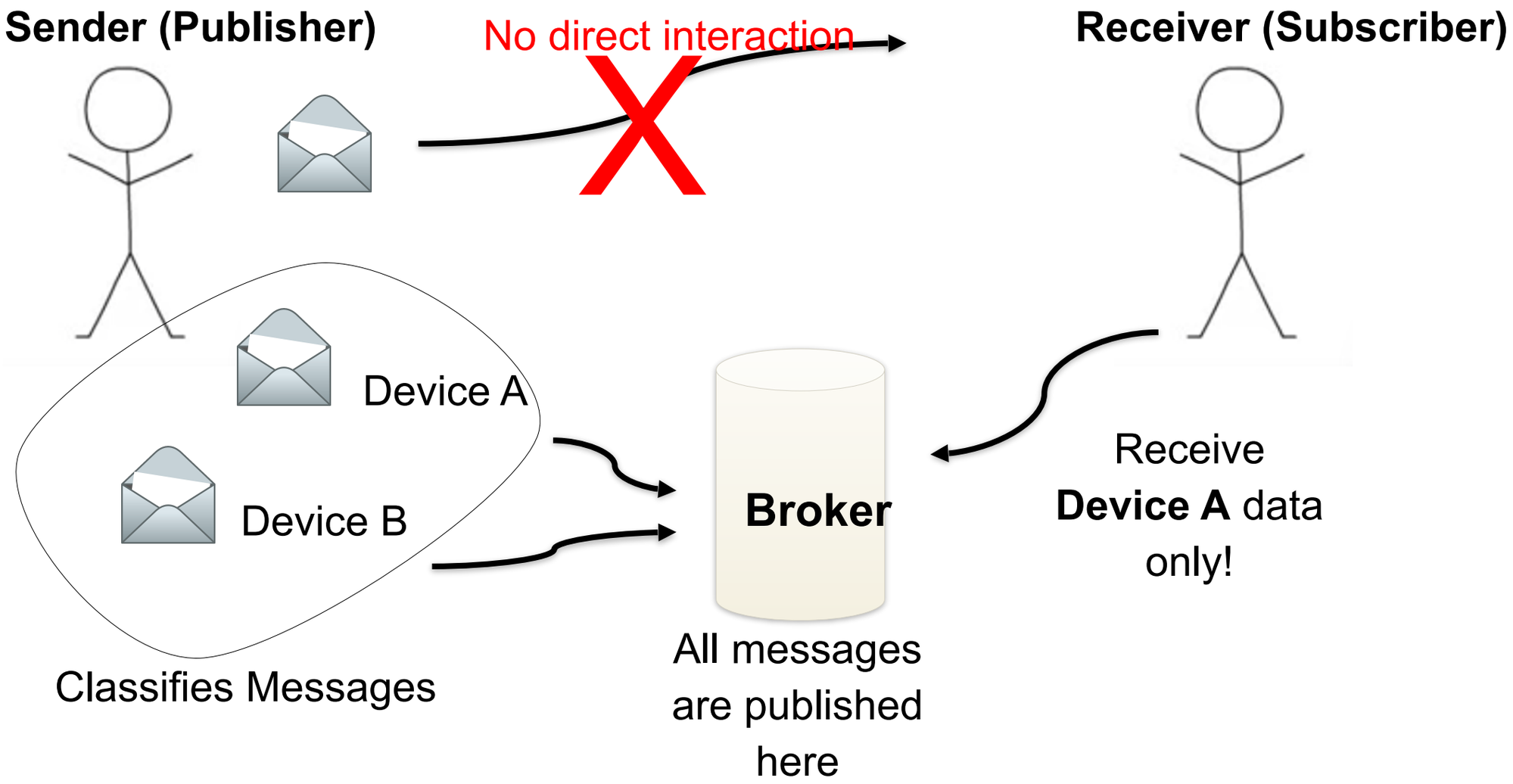
- MQTT
- Enabling Wifi
- IoT Platform
- Kibana
- Assignment

MQTT

IoT Usecase

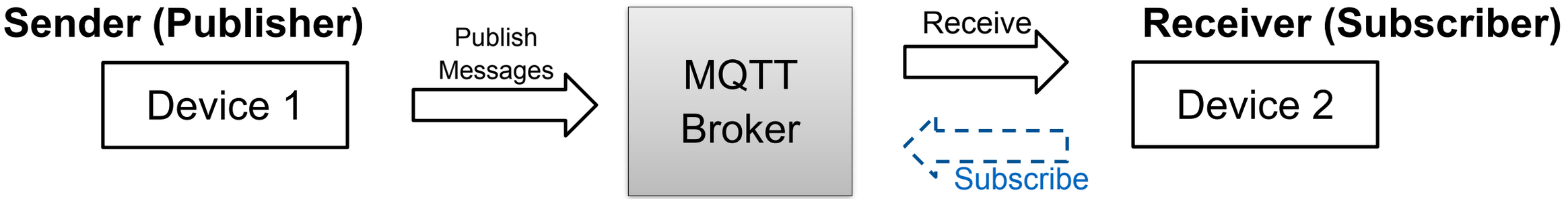

Devices
Gateway
Streaming Platform
Anomaly Detection
Predictive Maintenance
Devices Monitoring
Edge
Integration
Cloud

Publish / Subscribe



MQTT

- MQ Telemetry Transport or Message Queuing Telemetry Transport
- Lightweight, publish-subscribe network protocol that transports messages between devices.



1. Device 1 publishes to a **topic**

2. Device 2 subscribes to **same topic**
3. Device 2 receives messages.

- Topics are represented with strings and separated by slashes. Slashes indicate topic level.
- Example: office/room123/lamp (for lamp in room 123 in office)
- Broker is responsible for receiving the messages, filtering them and publishing to subscribed clients (For example Mosquitto broker).

MQTT QoS

- QoS levels in MQTT:
  - At most once (0)
  - At least once (1)
  - Exactly once (2).
- When you talk about QoS in MQTT, you need to consider the two sides of message delivery:
  - Message delivery form the publishing client to the broker.
  - Message delivery from the broker to the subscribing client.
- Higher QoS level requires more overhead
  - e.g. QoS 1

Quality of Service level 1: delivery at least once

MQTT Sessions

- Clean session
  - When subscriber disconnects, subscriptions and not received messages are not stored on the broker.
- Persistent session
  - In case of disconnect, subscriptions and messages with QoS 1 and QoS 2 are stored.

MQTT: Pros and Cons

- Pros
  - Lightweight
  - Simple APIs
  - Built for poor connectivity and high latency scenarios.
  - Many client connections (around 10s of thousands per MQTT server)
- Cons
  - Limited scalability

WIFI on ESP32

1. Connect to Wifi

- Run and understand the station example at .../esp-idf/examples/wifi/getting_started/station

```
/* The examples use WiFi configuration that you can set
   via project configuration menu

   If you'd rather not, just change the below entries to strings with
   the config you want - ie #define EXAMPLE_WIFI_SSID "mywifissid"
*/
#define EXAMPLE_ESP_WIFI_SSID      CONFIG_ESP_WIFI_SSID
#define EXAMPLE_ESP_WIFI_PASS      CONFIG_ESP_WIFI_PASSWORD
#define EXAMPLE_ESP_MAXIMUM_RETRY  CONFIG_ESP_MAXIMUM_RETRY

/* FreeRTOS event group to signal when we are connected*/
static EventGroupHandle_t s_wifi_event_group;

/* The event group allows multiple bits for each event,
   but we only care about two events:
 * - we are connected to the AP with an IP
 * - we failed to connect after the maximum amount of retries */
#define WIFI_CONNECTED_BIT BIT0
#define WIFI_FAIL_BIT      BIT1
```

```c
void wifi_init_sta(void)
{
    //Creates an event group with 24 bits
    //Stores event flags in individual bits
    //Tasks can wait for the flag to be raised and are then released
    s_wifi_event_group = xEventGroupCreate();
    //Initialize the network interface
    ESP_ERROR_CHECK(esp_netif_init());
    //Creates the system event loop which handles all events
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    //Creates a station network interface object
    esp_netif_create_default_wifi_sta();
    //Obtain a default configuration
    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    //Create a Wifi driver task and initialize the driver
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));
```

```
esp_event_handler_instance_t instance_any_id;
esp_event_handler_instance_t instance_got_ip;
//Register the event handler to the default event loop
//Events of the specified base id WIFI_EVENT are forwarded
//to the given event handler.
//Event handler instance is only needed to deregister the handler
ESP_ERROR_CHECK(esp_event_handler_instance_register(
        WIFI_EVENT,
        ESP_EVENT_ANY_ID,
        &event_handler,
        NULL,
        &instance_any_id));

ESP_ERROR_CHECK(esp_event_handler_instance_register(
        IP_EVENT,
        IP_EVENT_STA_GOT_IP,
        &event_handler,
        NULL,
        &instance_got_ip));
```

```
//Create a wifi configuration object
//.threshold.authmode define the minimum authentication mode
//pmf_cfg: extend privacy protection to network management frames
wifi_config_t wifi_config = {
    .sta = {
        .ssid = EXAMPLE_ESP_WIFI_SSID,
        .password = EXAMPLE_ESP_WIFI_PASS,
        .threshold.authmode = WIFI_AUTH_WPA2_PSK,
        .pmf_cfg = {
            .capable = true,
            .required = false
        },
    },
};
//Determines station mode
ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA) );
//Sets the configuration of the station
ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_STA, &wifi_config) );
//Starts the station
ESP_ERROR_CHECK(esp_wifi_start() );
ESP_LOGI(TAG, "wifi_init_sta finished.");
```

```
//Blocking until connected or failed
EventBits_t bits = xEventGroupWaitBits(
        s_wifi_event_group,
        WIFI_CONNECTED_BIT | WIFI_FAIL_BIT,
        pdFALSE,
        pdFALSE,
        portMAX_DELAY);

    if (bits & WIFI_CONNECTED_BIT) {
    ESP_LOGI(TAG, "connected to ap SSID:%s password:%s",
    EXAMPLE_ESP_WIFI_SSID, EXAMPLE_ESP_WIFI_PASS);
} else if (bits & WIFI_FAIL_BIT) {
    ESP_LOGI(TAG, "Failed to connect to SSID:%s, password:%s",
    EXAMPLE_ESP_WIFI_SSID, EXAMPLE_ESP_WIFI_PASS);
} else {
    ESP_LOGE(TAG, "UNEXPECTED EVENT");
}
```

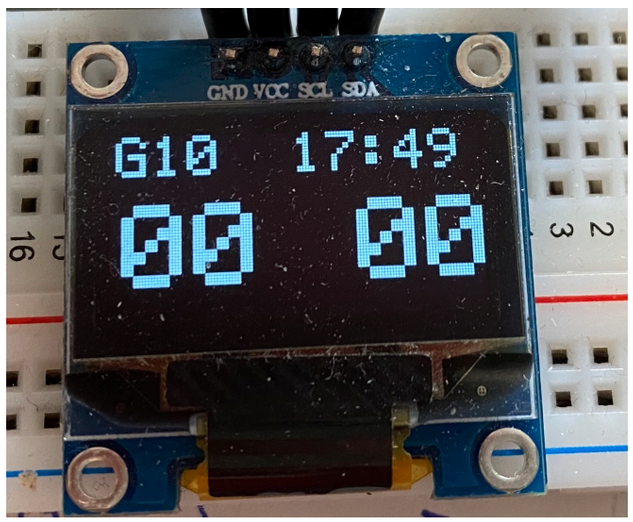
```c
static int s_retry_num = 0;

static void event_handler(void* arg, esp_event_base_t event_base,
int32_t event_id, void* event_data){
    // START and DISCONNECTED events
    if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START) {
        esp_wifi_connect();
    } else if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_DISCONNECTED) {
        if (s_retry_num < EXAMPLE_ESP_MAXIMUM_RETRY) {
            esp_wifi_connect();
            s_retry_num++;
            ESP_LOGI(TAG, "retry to connect to the AP");
        } else {
            //Signal failed
            xEventGroupSetBits(s_wifi_event_group, WIFI_FAIL_BIT);
        }
        ESP_LOGI(TAG,"connect to the AP fail");
    } else if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP) {
        ip_event_got_ip_t* event = (ip_event_got_ip_t*) event_data;
        ESP_LOGI(TAG, "got ip:" IPSTR, IP2STR(&event->ip_info.ip));
        s_retry_num = 0;
        //Signal connection
        xEventGroupSetBits(s_wifi_event_group, WIFI_CONNECTED_BIT);
    }
}
```

SNTP

Obtain the current time via the Network Time Protocol

- ESP32 provides a realtime clock (RTC) based on an internal 150 kHz oscillator or an external 32kHz crystal. (ESP WROOM board)
- RTC needs to be initialized from SNTP time server. It keeps time in sleep modes
- ESP32 SNTP library automatically updates RTC
- Implement the display showing the current time.



```c
void time_sync_notification_cb(struct timeval *tv)
{
    ESP_LOGI(TAG, "Notification of a time synchronization event");
}
```

```c
    ESP_LOGI(TAG, "Initializing SNTP");
    //Poll SNTP server via unicast. Default every hour
    sntp_setoperatingmode(SNTP_OPMODE_POLL);
    //Define SNTP server outside and in the MI building
    sntp_setservername(1, "pool.ntp.org");
    sntp_setservername(0, "ntp1.in.tum.de");
    //Define update notification function.
    sntp_set_time_sync_notification_cb(time_sync_notification_cb);
#ifdef CONFIG_SNTP_TIME_SYNC_METHOD_SMOOTH
    //if update necessary RTC is updated smoothly to reduce error
    sntp_set_sync_mode(SNTP_SYNC_MODE_SMOOTH);
#endif
    //Fetches time and keeps updating RTC
    sntp_init();
```

```
//Datatype for time in seconds since 1.1.1970
time_t now = 0;
//Linux data structure with calendar date and time broken down into its components.
struct tm timeinfo = { 0 };
//Number of retries waiting for synchronization
int retry = 0;
const int retry_count = 10;
while (sntp_get_sync_status() == SNTP_SYNC_STATUS_RESET && ++retry < retry_count) {
  ESP_LOGI(TAG, "Waiting for system time to be set... (%d/%d)", retry, retry_count);
  vTaskDelay(2000 / portTICK_PERIOD_MS);
}
if (retry==retry_count){
  ESP_LOGE(TAG,"Could not retrieve time.!\n");
  esp_restart();
}
//Retrieve time in seconds
time(&now);
```

CAPS IoT Platform

The need for an IoT Platform

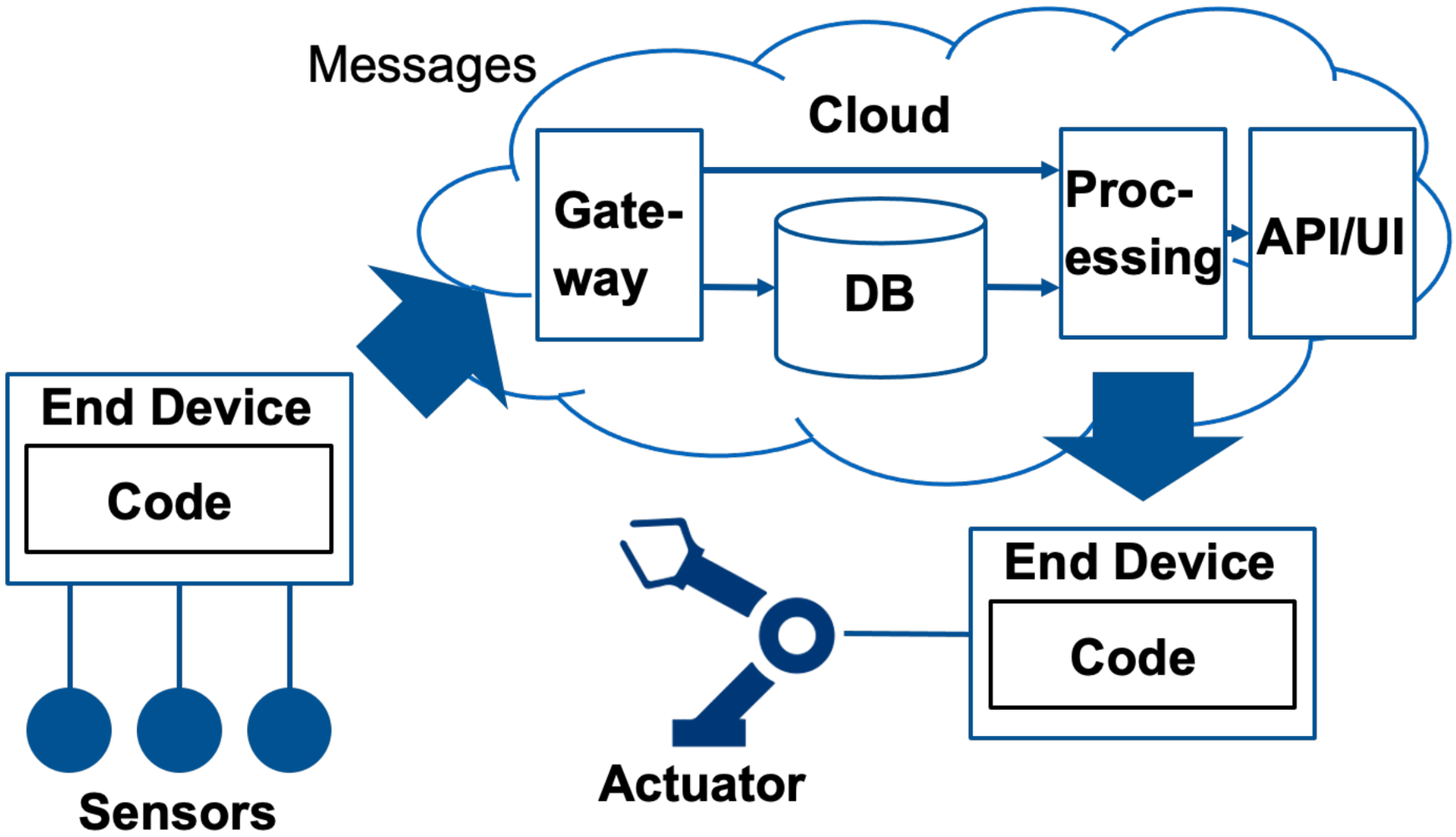
- The sensor data is scattered and too low-level.
- Usually, it is not possible to make meaningful conclusions based on it.
- Hence, one needs to centralize the data from many sensors in order to construct the higher-level representations.

- IoT Platform is the software solution where the centralization happens.

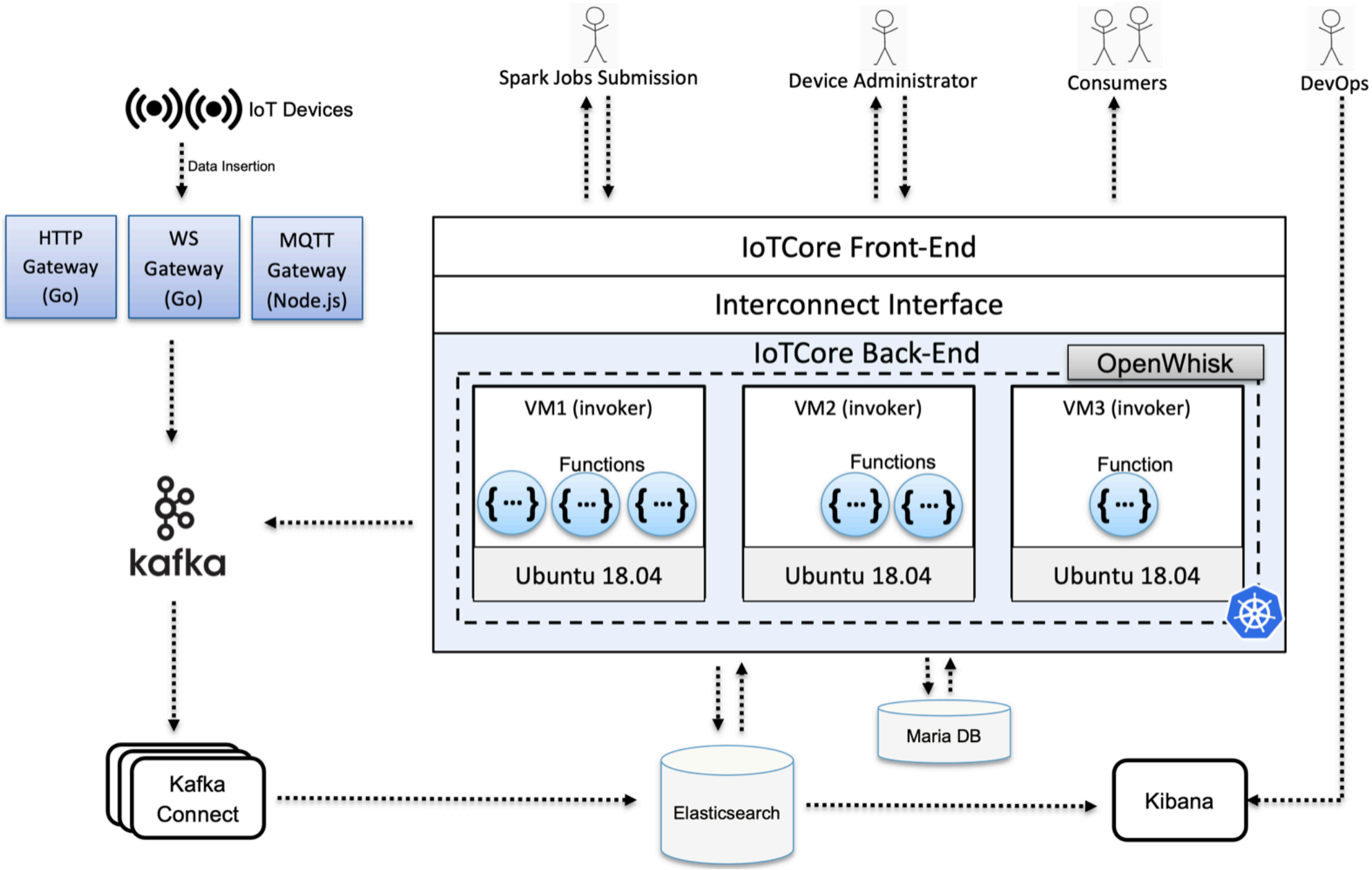
Functions of the IoT Platform

- IoT Platforms implements the following functions:
  - Storing the data
  - Getting the data
  - Providing access to the data / access control
  - Data processing and visualization
  - Notification and actuation in the environment

I10 IoT Platform

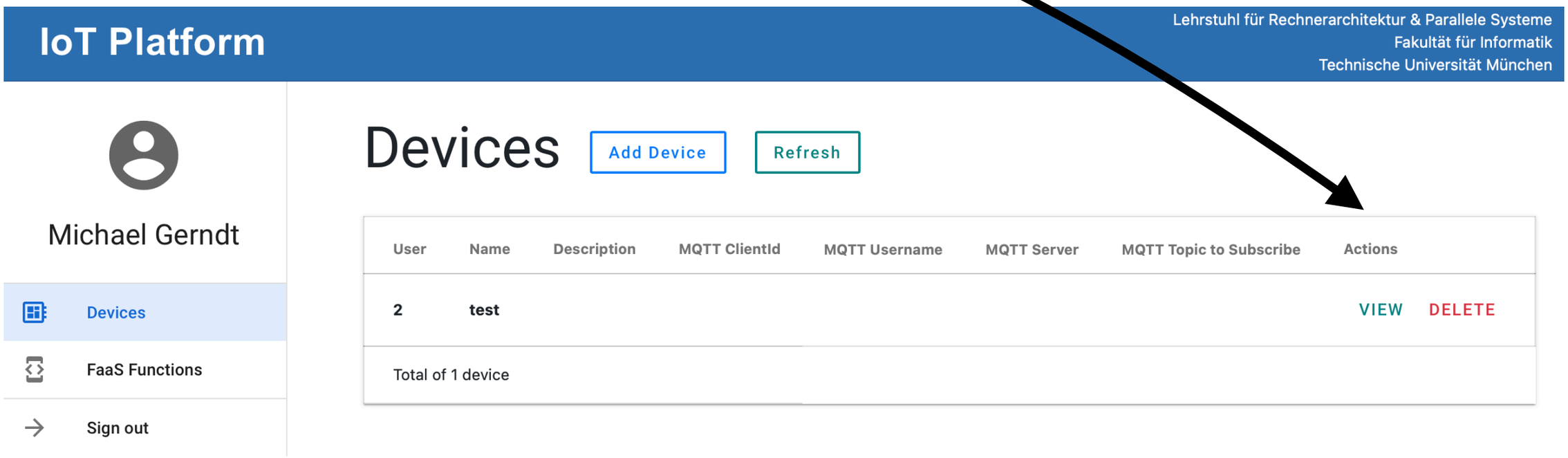


Architecture



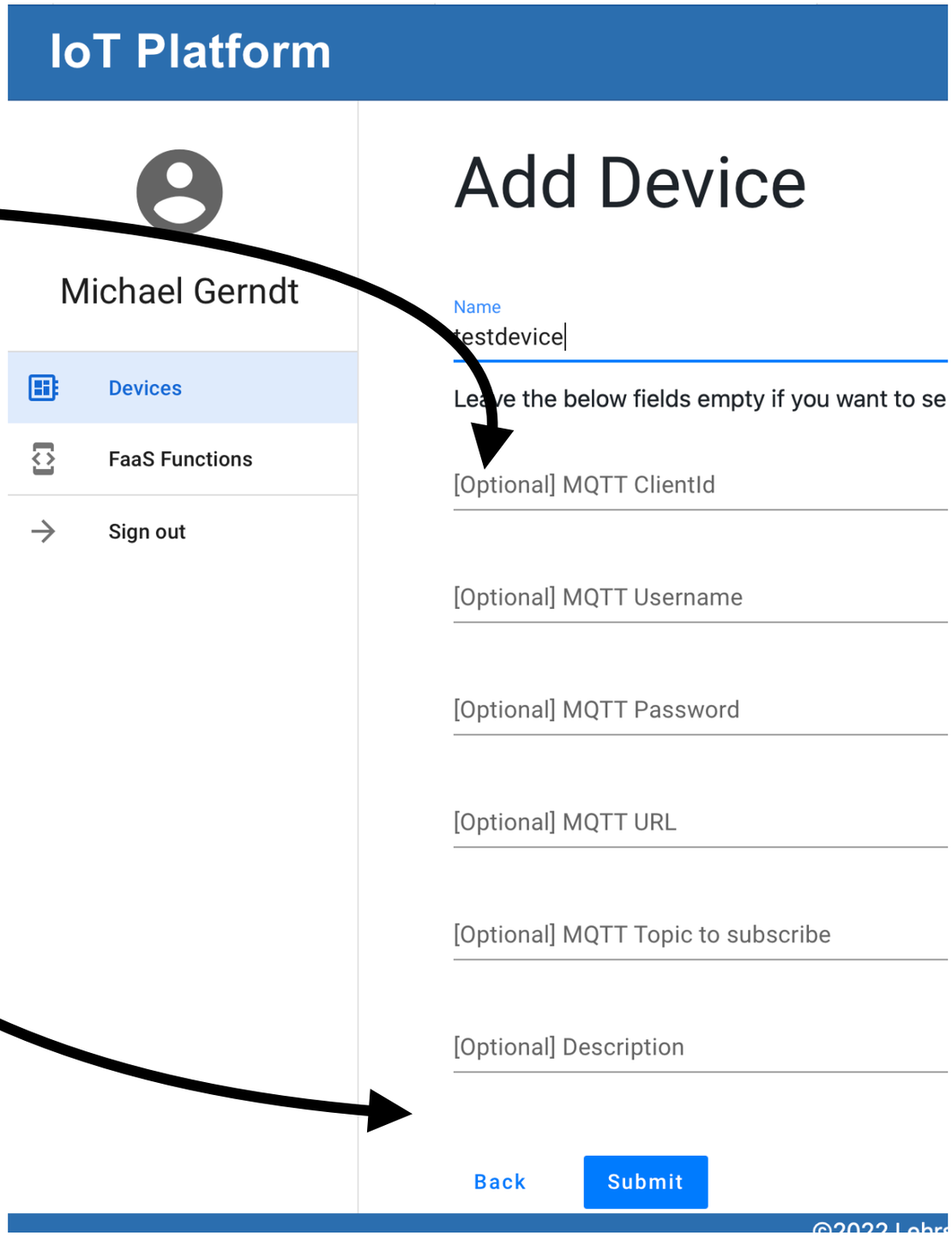
IoT Platform: Device Management

- http://138.246.236.107:3000
- Devices
  - View and manage your devices
  - Check the sensors.



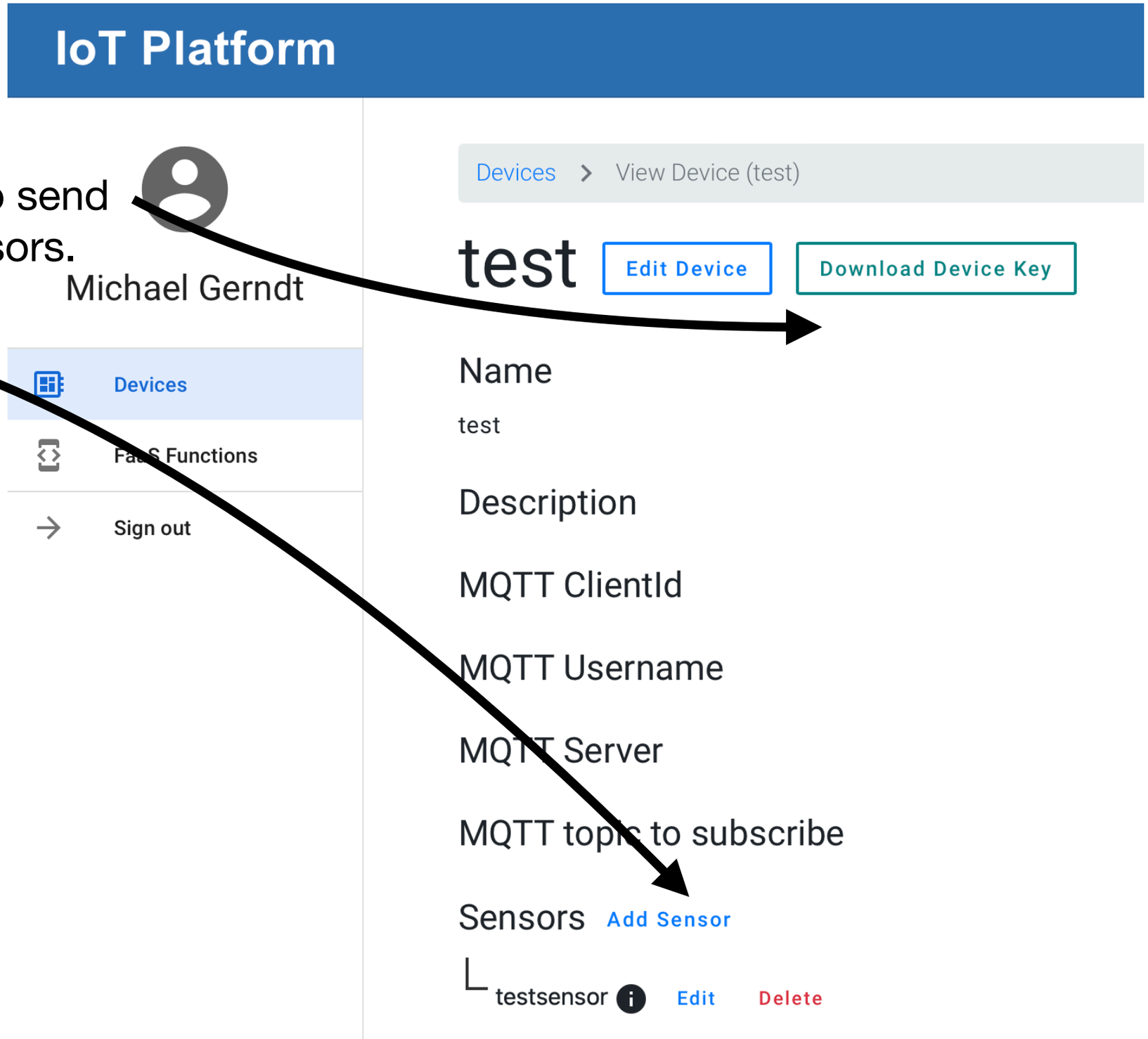
IoT Platform: Device Management

- Add a new device: name
- MQTT
  - External server
  - Internal server



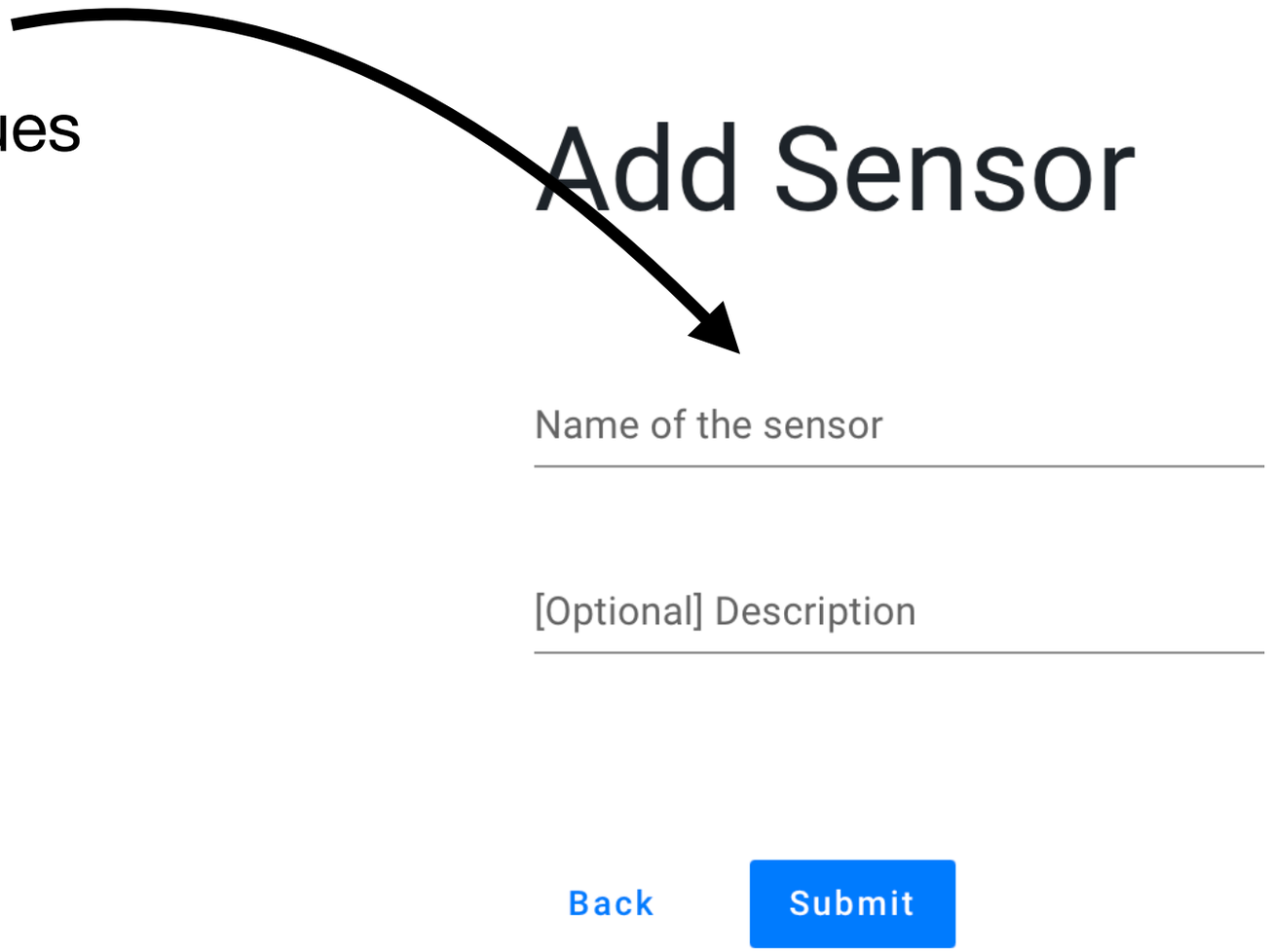
IoT Platform: Sensor Management

- View device
- Manage sensors
  - Download device key to send data for the device sensors.
  - Add sensor



IoT Platform: Sensor Management

- Add sensor name
- Currently only integer values are supported



Insert Sensor Data

```
mqtt_event_group = xEventGroupCreate();
const esp_mqtt_client_config_t mqtt_cfg = {
    .event_handle = mqtt_event_handler,
    .host = "131.159.35.132",
    .username = "JWT",
    .password = "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpYXQiOjE2M
    .port = 1883
    //.cert_pem = (const char *)mqtt_eclipse_org_pem_start,
    //mqtt://username:password@mqtt.eclipse.org:1884
};
```

Insert Sensor Data

```
time_t now = 0;
time(&now);
char msg[256];

sprintf(msg, "{\"username\":\"gerndt\",\"device_id\":\"54\",\"temperature\":%d,\"timestamp\":%lu000}",
        vDHT11data.temperature, now);
ESP_LOGI(tag, "Topic %s: %s\n", MQTT_TOPIC, msg);
int msg_id = esp_mqtt_client_publish(mqtt_client,MQTT_TOPIC, msg, strlen(msg), qos_test, 0);
```

Kibana: Visualizing the Data

Introduction to Elasticsearch

- Elasticsearch is an open source persistent data storage
- Elasticsearch consists of
  - Indices
  - Index consists of types
  - Types consist of documents with properties
- The minimal addressable element in Elasticsearch is the document

Kibana

- Kibana is an interface to manage the Elastic Stack
- It provides
  - Analytics: Visualization and analysis of data
  - Ready to start visualizations
    - Elastic Security Solution
    - Enterprise Search
    - Observability

- The IoT platform Kibana instance
  - http://138.246.236.107:5602
  - Login credentials are in Moodle.

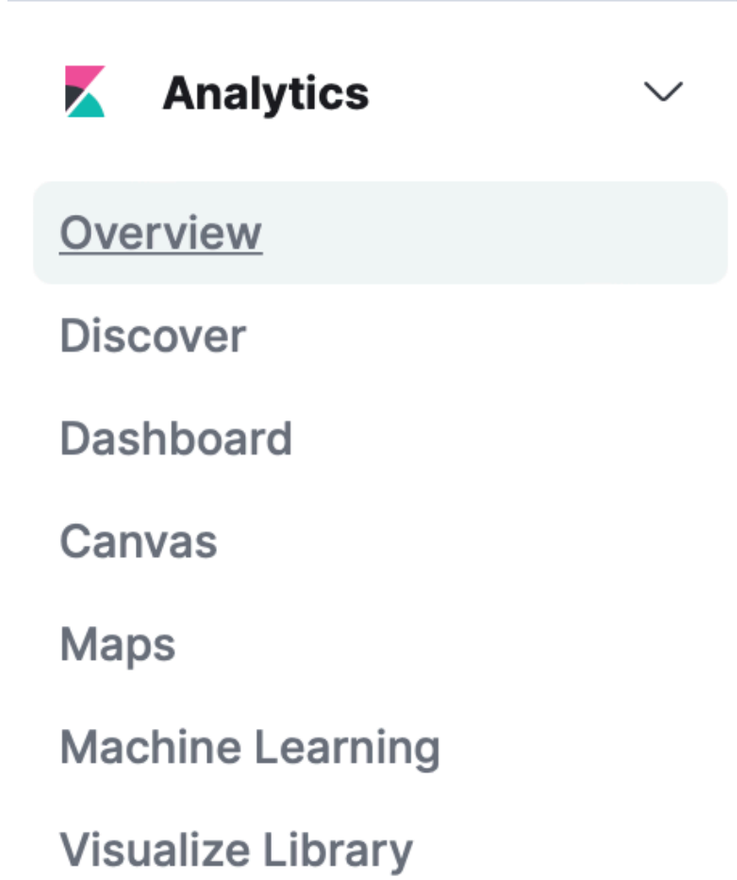
Visualization in Kibana

- Kibana is a free native visualization platform for the data stored in Elasticsearch.

- Key terms:
  - **Visualization** is a graph that is built based on the data in Elasticsearch
  - **Dashboard** is a collection of visualizations arranged in some order to provide a comprehensive description of some problem area
  - **Panel** is a region on the dashboard which contains a visualization (graph) and which can be moved around and resized
  - **Time window** is a selected time frame for which the visualization is built, e.g. between 13th of April and 1st of May; changing the time window allows to zoom in/out into graphs

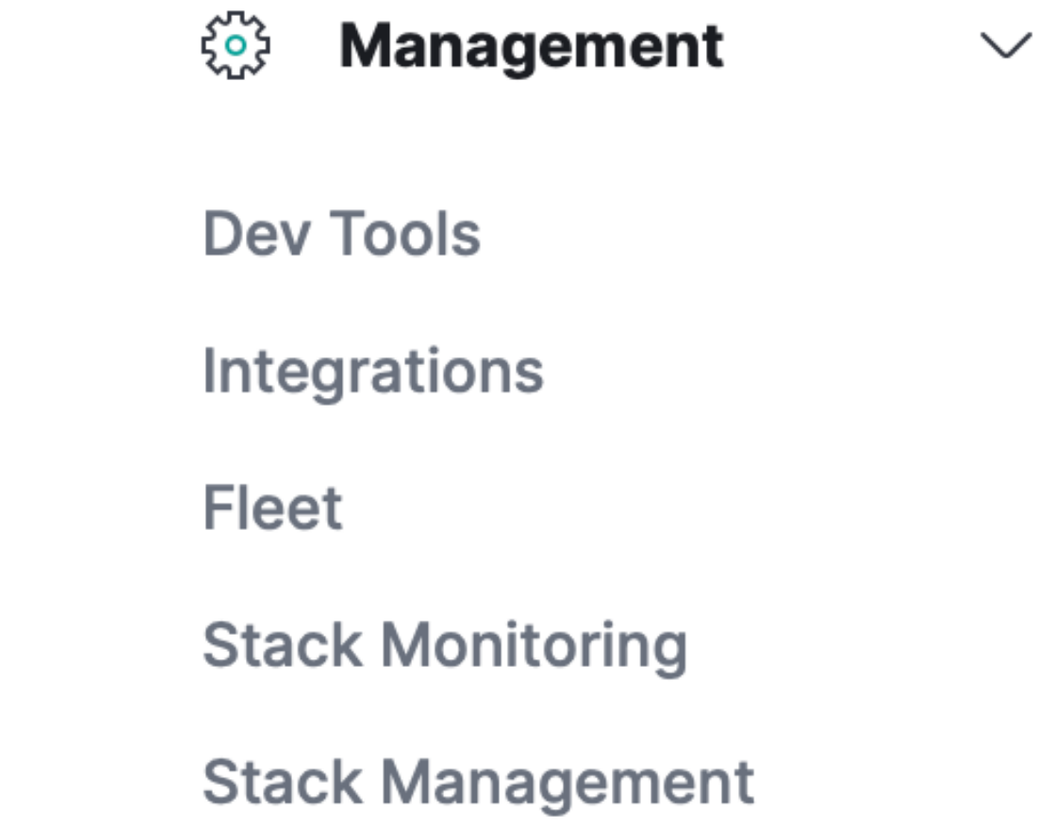
Kibana Analytics

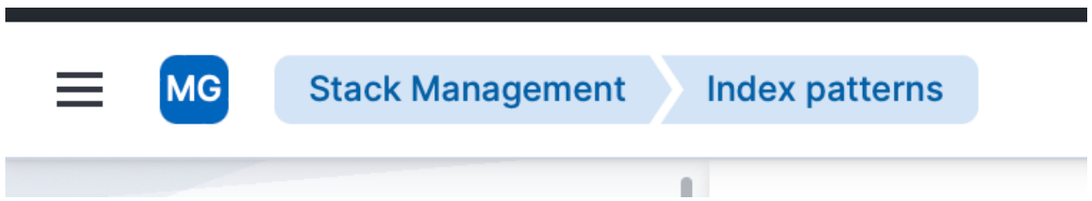
- **Overview** provides access to visualizations
- **Discover** page provides insights into the data contained in elastic search indices
- **Dashboards** page allows to create/manage dashboards containing multiple graphs
- **Canvas** dashboard creation
- **Maps** build maps with multiple layers
- **Machine Learning** for anomaly detection
- **Visualize Library** to store visualizations



Creating Index Pattern to Retrieve the Data

- Create first an index pattern
- Index pattern
  - Defines the data in elasticsearch
  - Defines properties of the fields





Creating Index Pattern to Retrieve the Data

Create index pattern

Name

2_2*

Use an asterisk (*) to match multiple characters. Spaces and the characters , /, ?, ", <, >, | are not allowed.

Timestamp field

timestamp

Select a timestamp field for use with the global time filter.

Show advanced settings

✓ Your index pattern matches 1 source.

2_2_3	Index

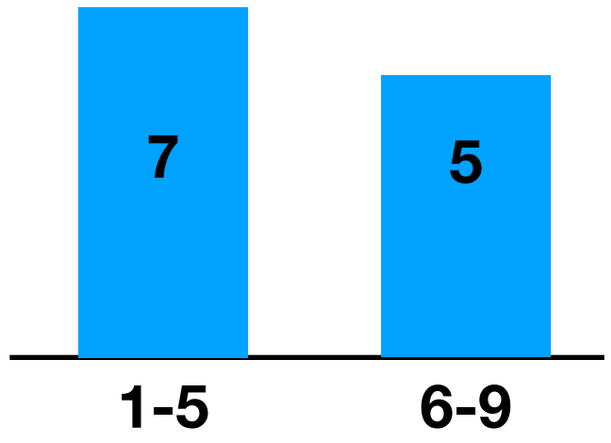
Rows per page: 10

× Close

Create index pattern

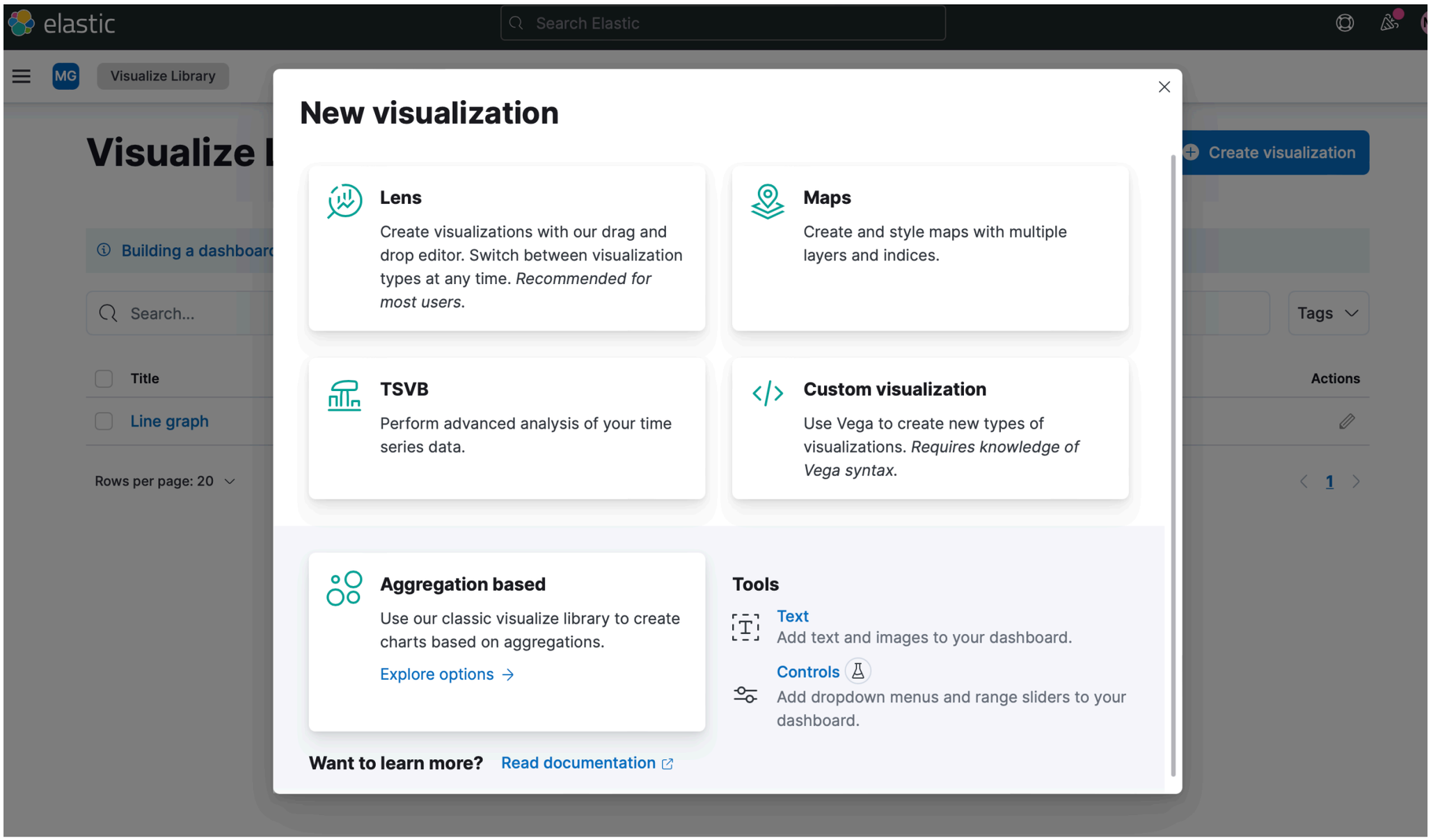
Visualization in Kibana: Metrics and Buckets

- Two abstractions:
  - Metrics represent the data values that will be central to the visualization.
  - Buckets is a mean to aggregate or group the data.
- Example: Observations 1, 2, 2, 2, 3, 4, 5, 6, 7, 7, 8, 9



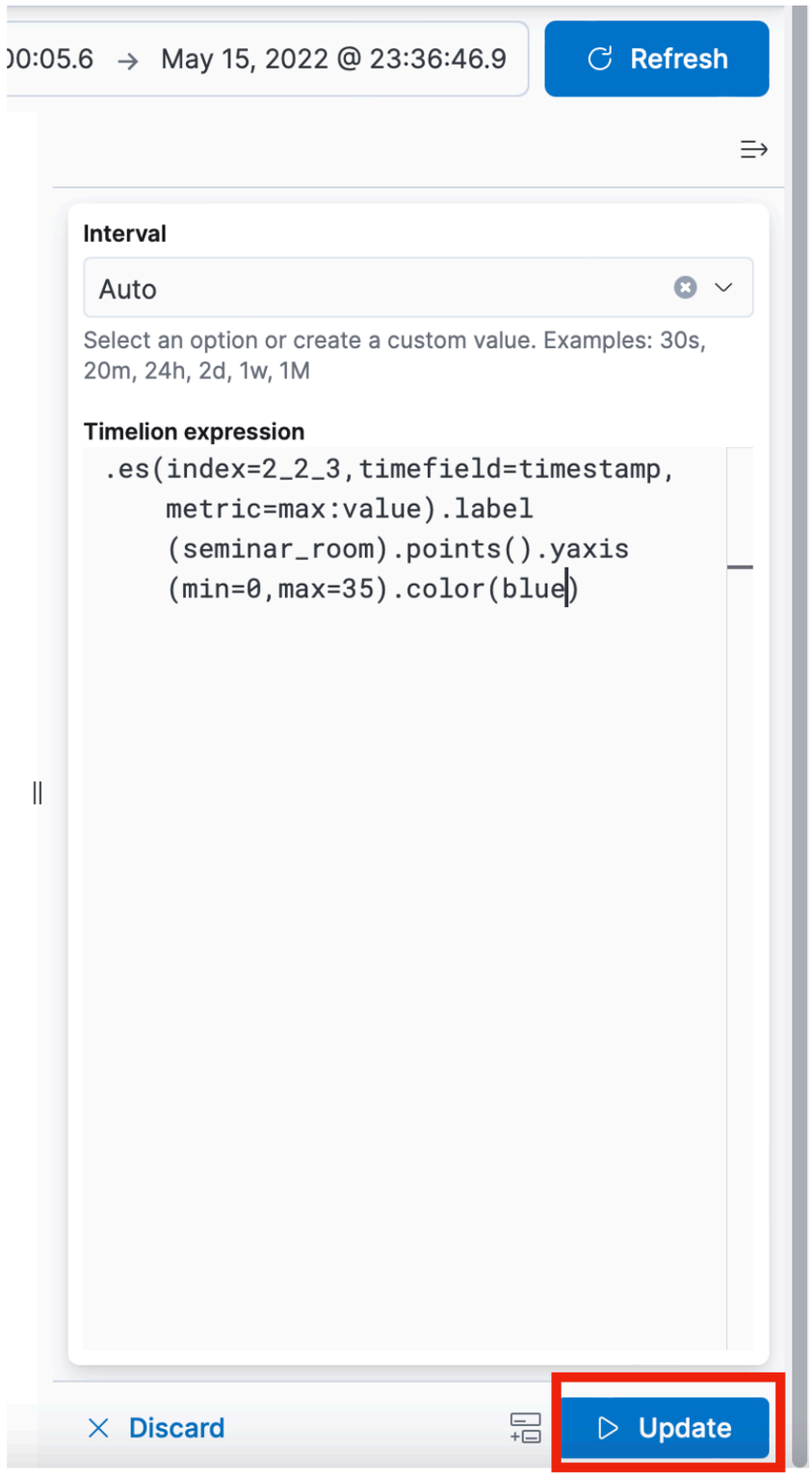
Create Visualization

- Visualize library - Aggregation based - timelion



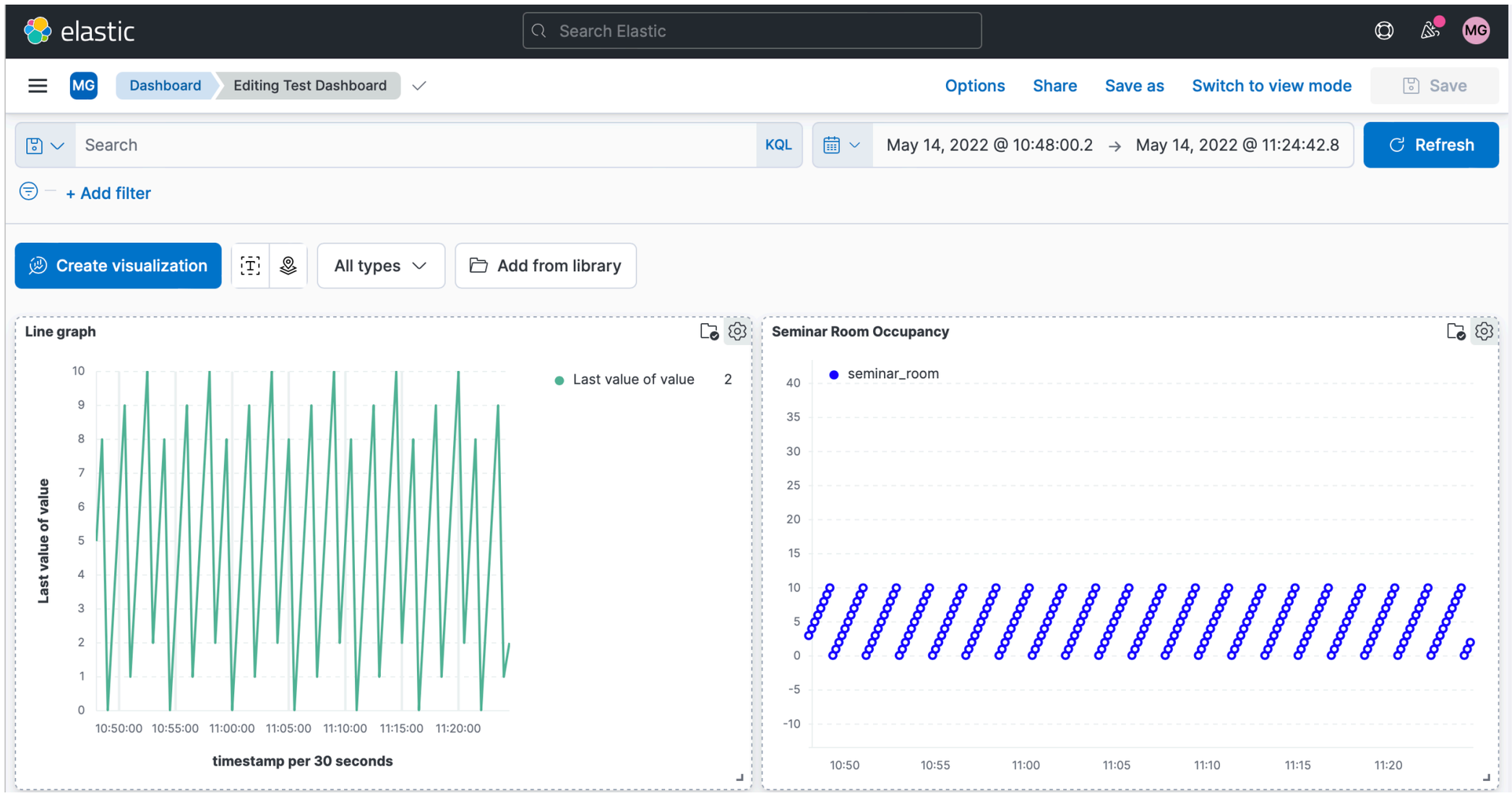
Time Series Graph with Timelion (1)

- Timelion is a time series data visualizer that uses the Timelion expression to make the graphs.
- The simplest expression requires to specify the
  - index pattern,
  - the time field,
  - the aggregation format with the metric to aggregate,
  - all followed by the graphical primitive used for the visualization, e.g. bars.
- .es(index=3_13_29,timefield=timestamp,metric=avg:value).bars()
- You need to press the Update button (in red frame) to see the graph. If empty, check time interval above the graph area. If resulting bars are too fine or too coarse, play with the interval.
- You can learn Timelion's syntax with a tutorial.



Create Dashboard

- Create a dashboard
- Add visualization from Visualize library



Potential Pitfall: Interpreting the Timestamp

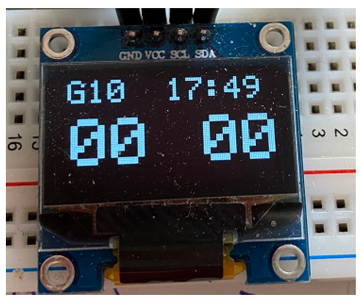
- Be careful when assigning a timestamp to your observation. Kibana supports the resolution down to seconds and milliseconds as of now.
- The problem with the correct format of timestamps becomes apparent when you use epoch time in your timestamp, i.e. seconds elapsed since midnight of 01.01.1970.
- Kibana assumes a particular unit of time, e.g. milliseconds. If you provide it with nanoseconds epoch time, then it will be treated incorrectly. Feel the difference:
  - The epoch time in microseconds is: 1586231735000000
  - Converter that assumes that the timestamp is in microseconds:
  - Tuesday, 7 April 2020, 3:55:35
  - Converter that assumes that the timestamp is in seconds:
  - 08/29/50267652 @ 12:53pm (UTC)
- Note: epoch time is ALWAYS interpreted! Knowing only the timestamp is NOT ENOUGH.

IoT Platform requires ms

```
time_t now = 0;
time(&now);
char msg[256];

sprintf(msg, "{\"username\":\"gerndt\",\"device_id\":\"54\",\"temperature\":%d,\"timestamp\":%lu000}",
        vDHT11data.temperature, now);
ESP_LOGI(tag, "Topic %s: %s\n", MQTT_TOPIC, msg);
int msg_id = esp_mqtt_client_publish(mqtt_client,MQTT_TOPIC, msg, strlen(msg), qos_test, 0);
```

1st Milestone

- Create device and sensor in Kibana
  - Download the device key
- Create an index_pattern
- Create a timelion visualization in Visualize Library
  - .es(index=2_2_3,timefield=timestamp,metric=max:value).label(seminar_room).points().yaxis(min=-5,max=35).color(blue)
  - Save with name "Group<n>_SeminarRoom"
- Merge the code from the test sensor (Moodle) with your code.
- Add display layout with Group, Time, Count, and Prediction
  - 

- Send measurements via MQTT to the platform every 5 minutes
- Check the visualization

1st Milestone

- Please provide your graphs and dashboards by Saturday 12:00.
- By then the boards should be setup in the shelf
- First data should be in the platform
- Thus, start immediately

- We will ensure that the shelve is always accessible.
- Wifi
  - Access to WLAN in seminar room
  - SSID: CAPS-Seminar-Room
  - Access Code: caps-schulz-seminar-room-wifi

Questions