Maximum Synergy Under One Roof

ju:niz REAL ESTATE

ju:niz

ju:niz ENERGY

GREENROCK

smart POWER

Green H2

How to Improve Renewable Energy Storage with MQTT, Modbus, and InfluxDB Cloud Dedicated

ju:niz ENERGY

Ricardo Kissinger | München, 26.09.2023

Agenda – Introduction to Company

01 **Introduction of ju:niz Energy GmbH**
Who we are and what we do

02 **Showcase of Generation 1.0 BESS**
References from our first Generation BESS

03 **Showcase of Generation 1.5 BESS**
A few examples from our current “SMAREG“ projects

ju:niz

01
Introduction of ju:niz Energy GmbH

ju:niz REAL ESTATE

ju:niz

ju:niz ENERGY

GREENROCK

smart POWER

Green H2

ju:niz Energy

Core business

- Intelligent large-scale storage systems that are operated in a grid-serving and economical manner.
- Decentral energy supply of renewable energies, battery storage and hydrogen for district areas
- intelligent energy management systems that control both the battery storage systems and the decentral energy system for optimal use.

Founded	Employees2023	Installed Capacity
2014	**66**	**145 MWh**

BVES

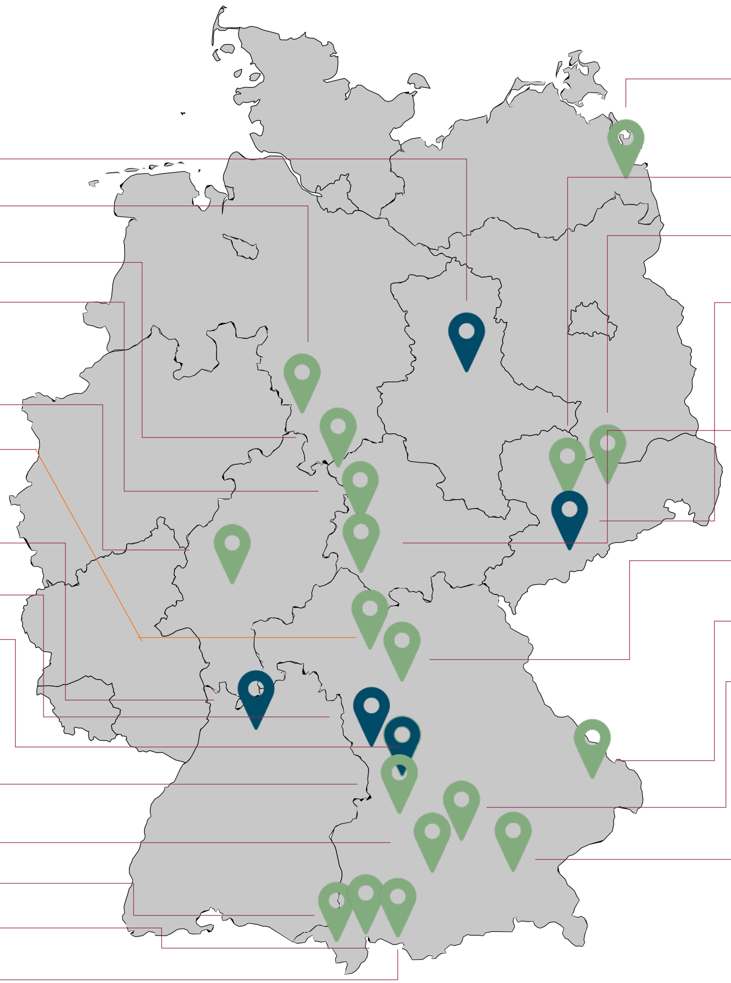
H2.B ZENTRUM WASSERSTOFF. BAYERN

bayern innovativ

ju:niz ENERGY

Our projects within Germany

In Betrieb
Im Bau



ju:niz

02

Showcase of Generation 1.0 BESS

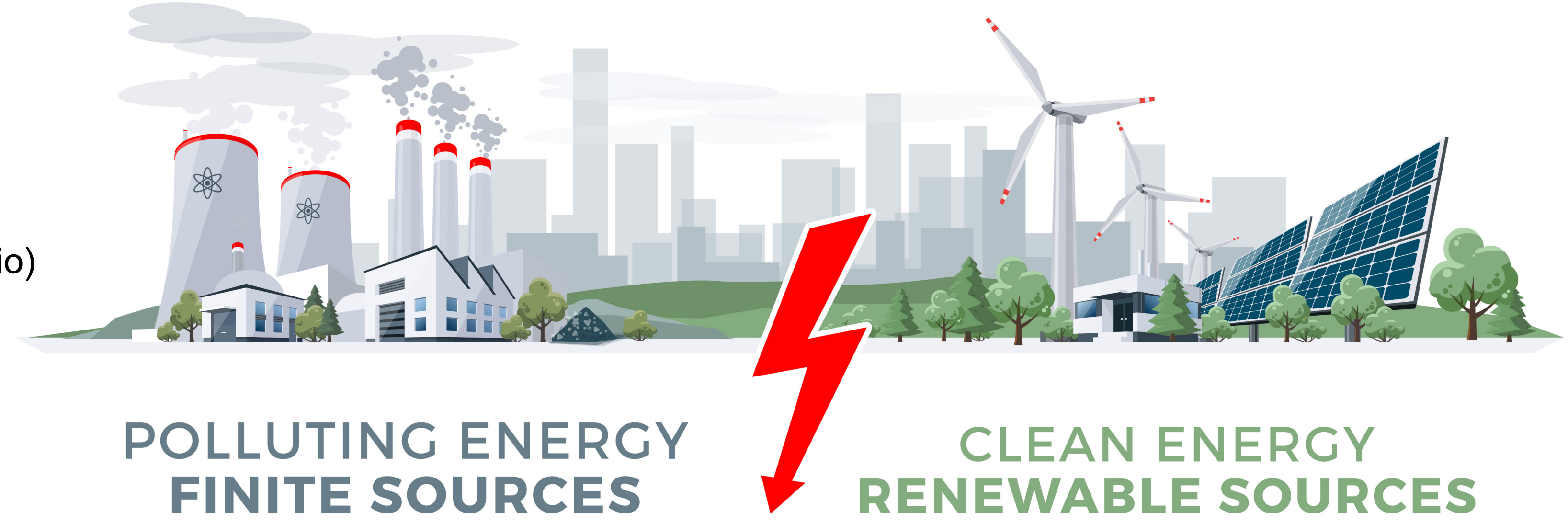
Introduction to Generation 1.0 BESS

Generation 1.0 BESS

- 25 Energy Storage plants across Germany
  - Starting capacity 150kwh
  - Average 1MWh capacity
  - Up to 11MWh capacity
- Using mainly Samsung NMC Batteries and "Second Life" Batteries with LMO

Main Usage cases:

- Primary Frequency Control (Main Usage scenario)
- Peak Shaving
- Self-Supply for Electric Energy
- Reduction of Distribution Grid Fees



Large-scale storage increases flexibility in the energy supply for industry/commerce and in the electricity grid

Frequency reserve and trading

Peak Shaving, Optimisation of own comsumption, 7000h for industry and district areas

> 10 MWh

< 5 MWh

iEMS

Operational Services

ju:niz ENERGY

PRL-Speicher Garching near by Munich

Key Facts

Customer	Forschungsprojekt
Location	Campus der Technischen Universität München
Completion	2016

Technical Design

Power	1,4 MW
Capacity	1,2 MWh
System	Samsung SDI / Daimler
Design	40 ft Battery container Rectifier Transformer
EMS	Smart Power iEMS

ju:niz ENERGY

JUWI Schmölln - First German Innovation project “Wind Energy and Energy Storage”

Key facts

Customer	Juwi
Location	Schmölln
Completion	2022

Technical Design

Power	2,9 MW
Capacity	3 MWh
System	SAMSUNG SDI
Design	40 ft Battery container Rectifier
EMS	Smart Power iEMS

SMAREG 1 - Smartbooster

Key Facts

Customer	Investor project
Location	Gotha
Completion	2020

Technical Design

Power	11 MW
Capacity	11 MWh
System	SAMSUNG SDI
Design	40 ft Battery container Rectifier Transformer Transfer station
EMS	Smart Power iEMS

ju:niz ENERGY

VWEW Leinau

Key Facts

Customer	Industrial Customer
Location	Kaufbeuren
Completion	2021

Technical Design

Power	7,3 MW
Capacity	3,7 MWh
System	SAMSUNG SDI
Design	20ft Battery container Rectifier Transformer
EMS	Smart Power iEMS

ju:niz ENERGY

03

Showcase of Generation 1.5 BESS

ju:niz

Introduction to Generation 1.5 BESS

Generation 1.5 BESS

- 7 Energy Storage plants across Germany
  - Starting capacity 11MWh
  - Average 23.7MWh capacity
  - Up to 67MWh capacity
  - Sum of 187.6MWh
- Using mainly Samsung NMC Batteries

Main Usage cases:

- Primary Frequency Control (Main Usage scenario)
- Reduction of Distribution Grid Fees

SMAREG4 – „Wartburgspeicher“

Key Facts

Customer	Investor project
Location	Eisenach
Completion	2022

Technical Design

Power	60 MW
Capacity	60 MWh
System	SAMSUNG SDI
Design	20x“Langhaus“ Design Rectifier Transformer
EMS	Smart Power iEMS

SMAREG6 – „Gunzenhausen“

Key Facts

Customer	Investor project
Location	Gunzenhausen
Completion	2023

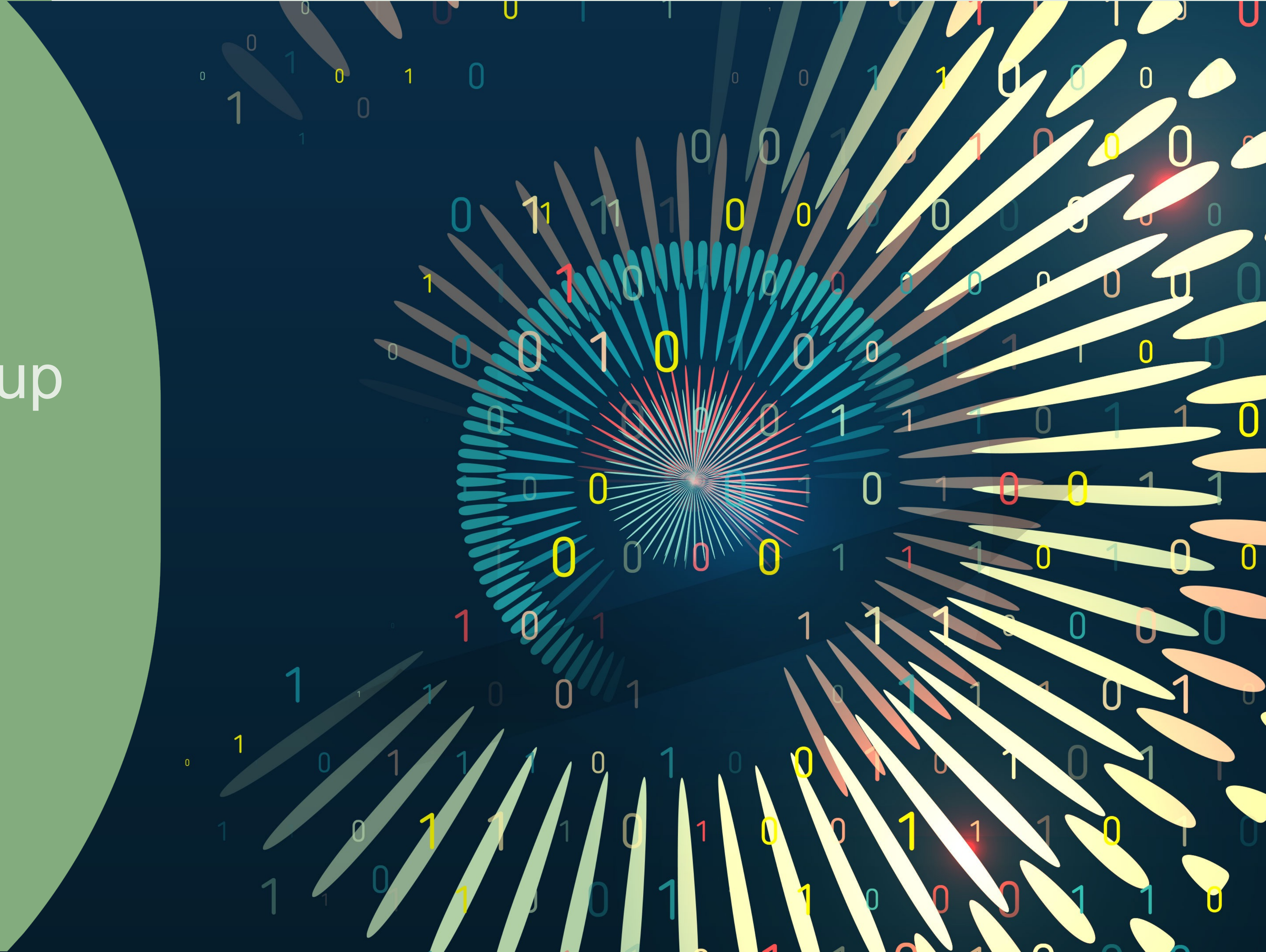
Technical Design

Power	24 MW
Capacity	24 MWh
System	SAMSUNG SDI
Design	2x Tripple „Langhaus“ Design Rectifier Transformer
EMS	Smart Power iEMS

Agenda – Technical Portion

04

Overview – Historical Setup



ju:niz

The Good Ol' Days

Centralised Monitoring Server:

- Rented vServer as the main influxDB Cluster
- No working backup solution
- Limitation of resources
- No upgrade of hardware/performance possible (dead end)
- Ubuntu Server installations close to end of life

Limitations of the legacy solution

- Unreliable syncing of data between Edge locations and centralised monitoring
- Due to Storage concerns, usage of retention policies to throw data out
- Due to throwing out data, analysis of the data is based on less precise information

Limitations of the used InfluxDB OSS 1.x

Features which were introduced with later influxDB OSS Versions

- Edge Data Replication: reliable way to sync data from Edge locations
- Flux Language: More flexible and powerful queries
- Integrated UI: Built-in dashboarding and alert management
- Tasks: Automated data processing without Kapacitor
- Token-based Auth: Enhanced security and access control
- Export/Import: Easier backup and restore options

05

InfluxDB Cloud V1

ju:niz

InfluxDB Cloud V1 – The begin of the journey

Requirements

- More reliable server setup (simple failover functionality at least!)
- Implementation of snapshots and backups
- Prevention to delete data due to storage issues
- Getting a more reliable data sync into the cloud
- Migrate all existing data from self-hosted influxDB instances into Cloud instance

Solution

- InfluxDB Cloud V1 Cluster – multiple node setup
- Implementation of Kapacitor Tasks to send data through Kapacitor to the influxDB Cloud
- Getting rid of storage issues or concerns

The problems and bottlenecks

Data Migration Headaches

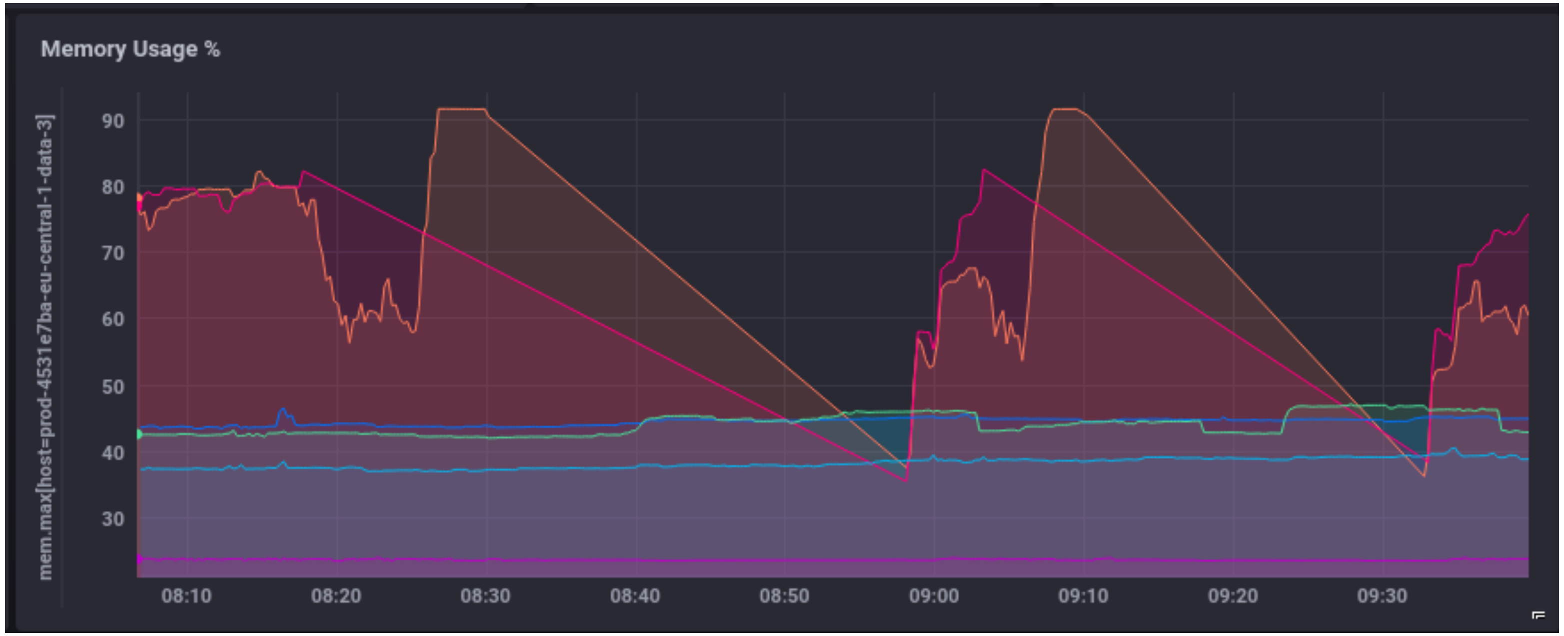
- Due to the nature of our legacy infrastructure, using the influxDB OSS 2.x tools was not an option
- Importing the compressed line protocol exports took forever and failed a lot of times

Getting data into the Cloud

- Kapacitor tasks are running like a cronjob – no “real time” data ingests
- Kapacitor is not sending data multiple times if it failed the first time

Cluster load always spikes

- Cluster is having a hard time to keep up with the workload
- Even though **not** all legacy plants are ingesting data



The problems and bottlenecks

06

InfluxDB Cloud Dedicated

ju:niz

The problems and bottlenecks

Data Migration Headaches

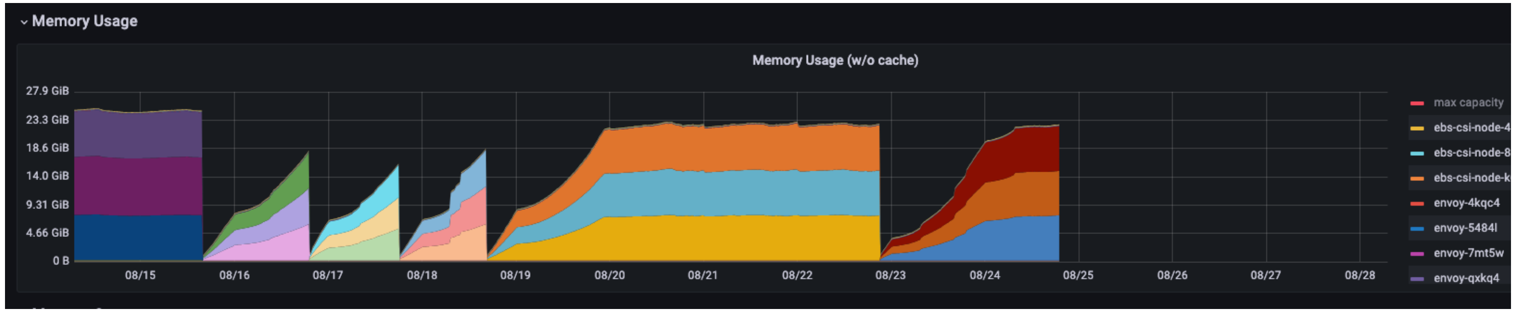
- Normalization of inconsistent data since InfluxDB V3 doesn't support different data types per field
- Getting rid of legacy "retention" policies since storage isn't a concern anymore

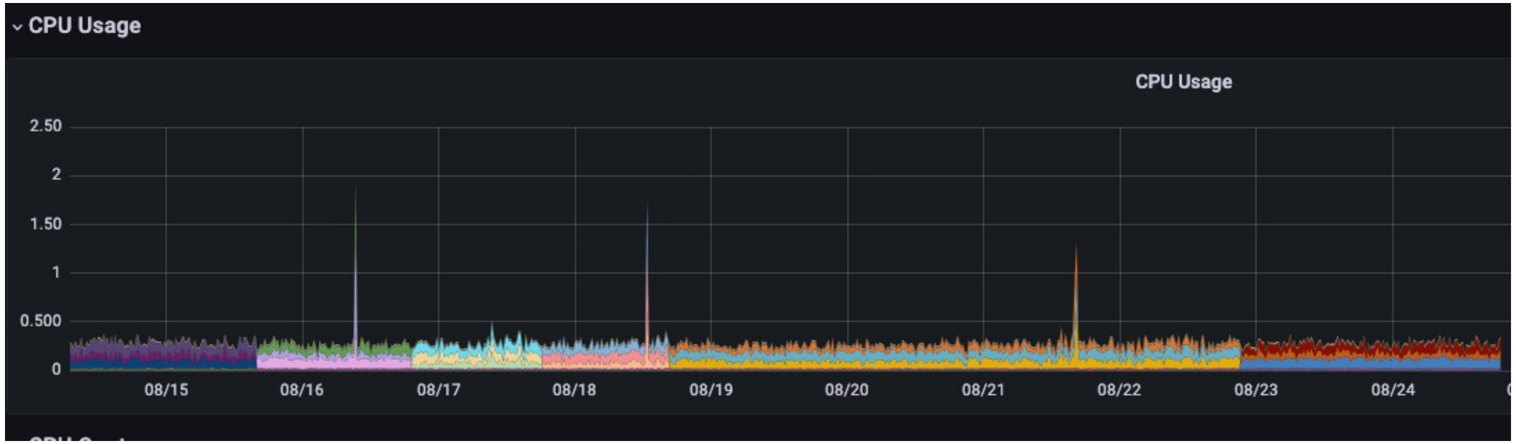
Getting data into the Cloud

- Using influxDB V2 write commands to send databases directly
- Running "inconsistent" databases through Telegraf to normalize the data
- Implementation of EDR for legacy plants

Cluster load

- Cluster is **not** having a hard time to keep up with the workload anymore
- **All legacy plants are ingesting data now**





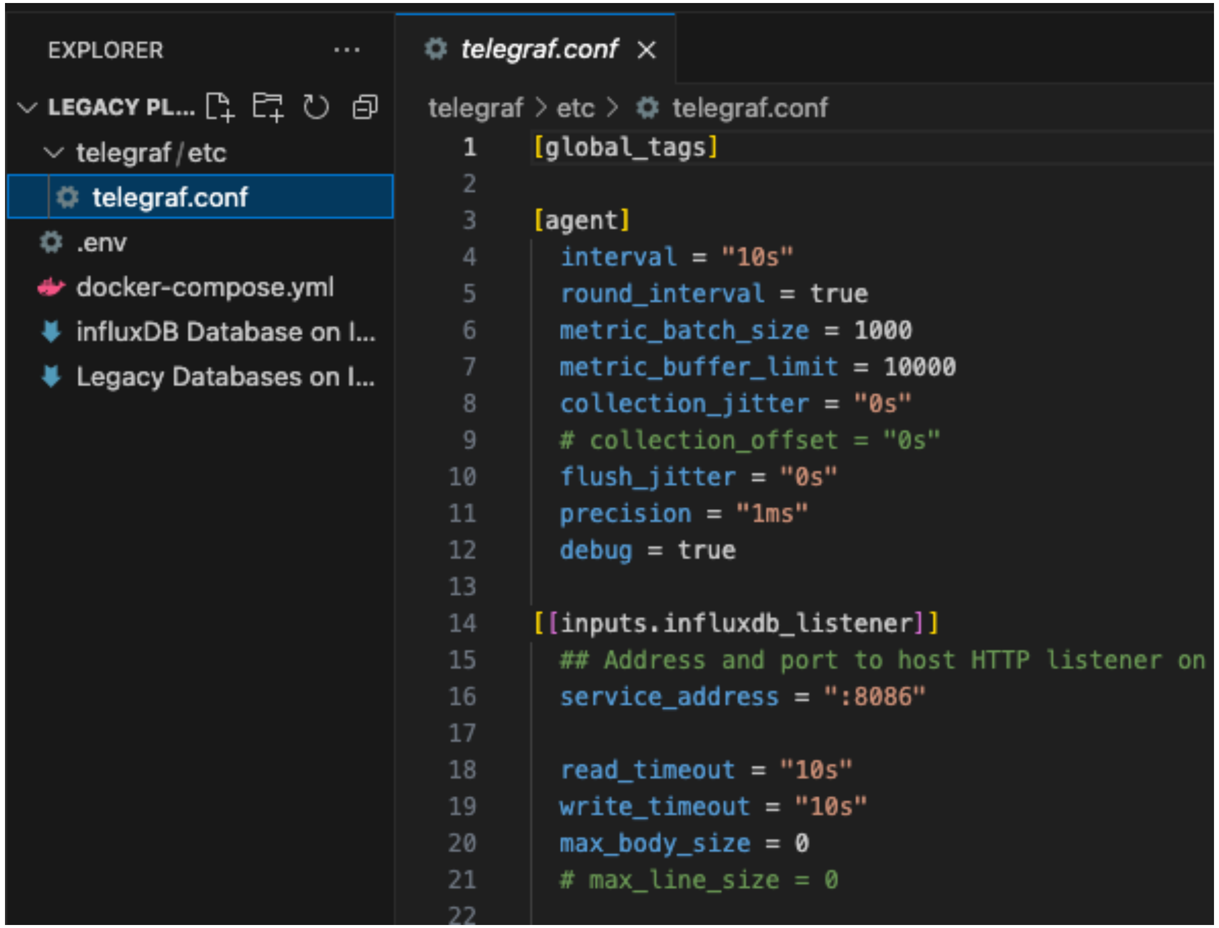
Introduction of EDR to Gen 1.0 Plants

Requirements

- More reliable transport from edge device data into cloud
- Avoiding to touch the actual plant control units, since touching the legacy scripts or SPS controls directly has a high risk potential of downtimes and interruptions
- Continuously data sending rather than getting data every 10mins
- Reliable mechanism to send back failed data

Solution

- Docker Compose “Swiss Army Knife” Container Setup which allows parallel installation to current local influxDB OSS
- Telegraf plays an influxDB v1 Listener
- Transfers data into influxDB OSS 2.4 Container
- Using EDR from the new influxDB Database
- Allows to keep all legacy scripts in place and no need to touch SPS
- Old Monitoring Server still gets data until all customers are migrated to the new Grafana instance



ju:niz

07

Gathering data with Telegraf from MQTT devices

ju:niz

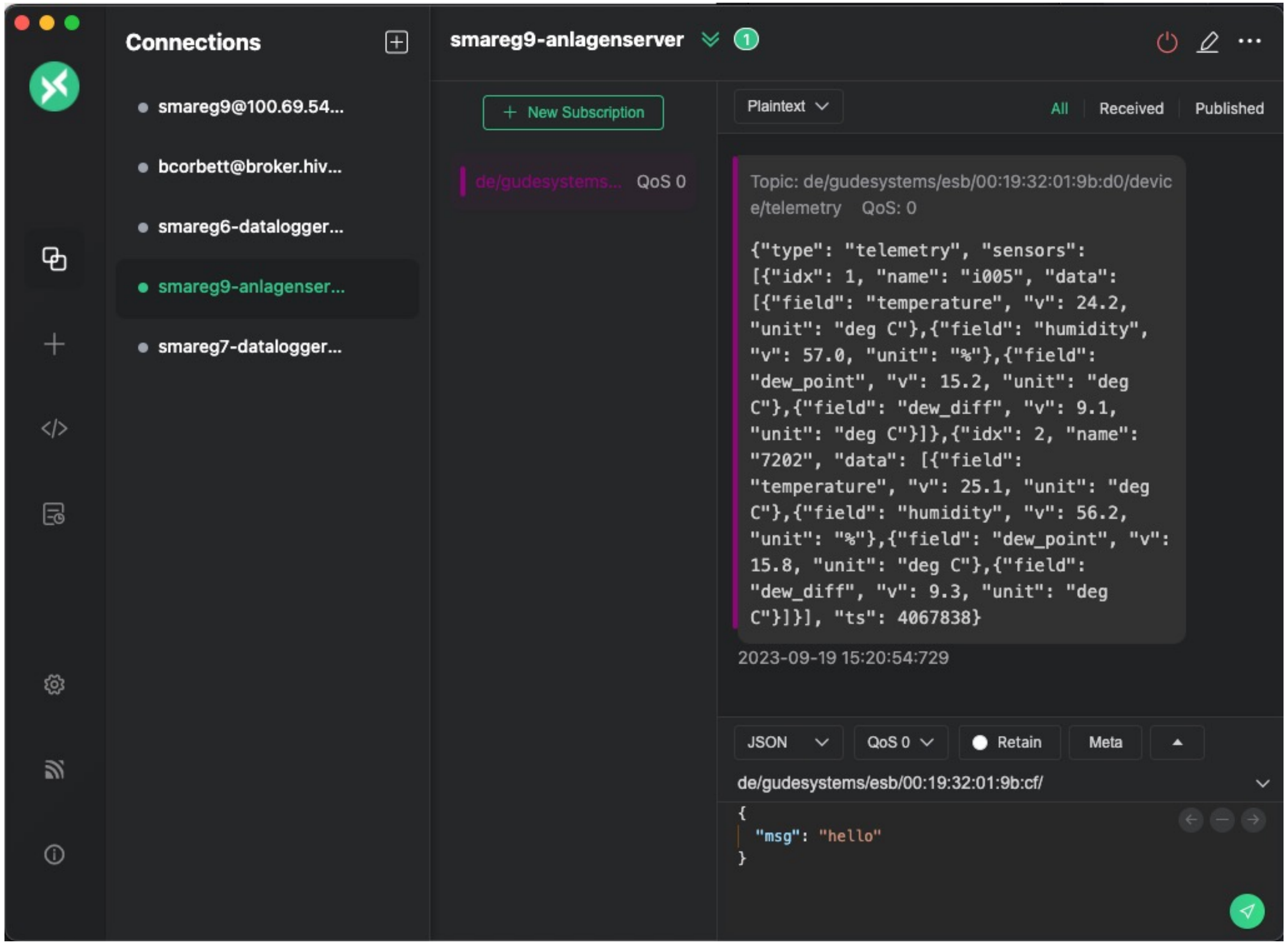
Temporary temperature logging

Requirements

- During construction time, final climate incl. all sensors are not yet working
- We need to have logging information about the temperatures and humidity for batteries
- Due to this being just temporary, the systems need to be implemented quickly with low costs

Solution

- LAN Temperature Sensors which report data via MQTT
- Gathering the data from MQTT with Telegraf
- Sending the Telegraf data to local influxDB OSS
- Using EDR to send to InfluxDB Cloud Dedicated V3
- Add the database in Grafana for visualization and alerts



Telegraf.conf file for getting the job done

Telegraf Setup

- Define MQTT Consumer Input Plugin
- Define InfluxDB V2 Output Plugin
- Configure MQTT input plugin

```
telegraf.conf

# Read metrics from MQTT topic(s)
[[inputs.mqtt_consumer]]
  ## Broker URLs for the MQTT server or cluster.  To connect to multiple
  ## clusters or standalone servers, use a separate plugin instance.
  ##   example: servers = ["tcp://localhost:1883"]
  ##            servers = ["ssl://localhost:1883"]
  ##            servers = ["ws://localhost:1883"]
  servers = ["tcp://127.0.0.1:1883"]

  ## Topics that will be subscribed to.
  topics = [
   "de/gudesystems/esb/+/device/#",
    "sensors/#",
  ]
```

```
## https://github.com/influxdata/telegraf/blob/master/

data_format = "xpath_json"
xpath_native_types = true

[[inputs.mqtt_consumer.xpath]]
  metric_name = "'LanghausData'"
  metric_selection = "sensors/*"

  # timestamp = "/ts"
  # timestamp_format = "unix"

  field_selection = "data/*"
  field_name = "field"
  field_value = "v"

  [inputs.mqtt_consumer.xpath.tags]
    sensor = "name"
    type = "/type"

  [inputs.mqtt_consumer.xpath.fields]
    idx = "idx"
```

MQTT Input Plugin

- Set processing data format, i.e. ”xpath_json”
- Define the Metrics name and selection
- Define your field names and values

ju:niz

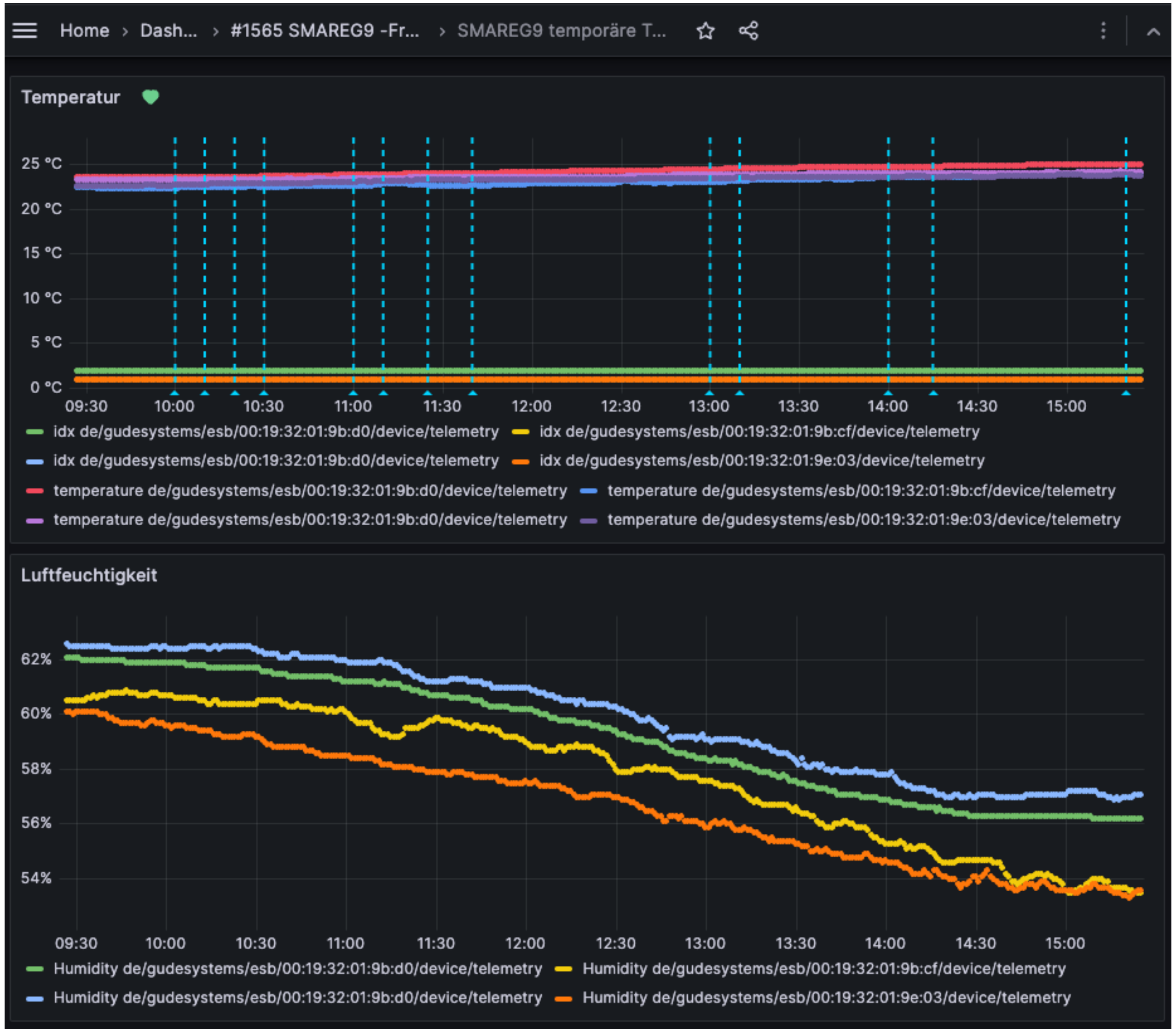
Grafana Visualization of temporary temperature logging

Grafana Visualization

- Showing information about each device and sensor
- Adding the information to Dashboards
- Allows historical information checking

Grafana Alerts

- Since plants in construction phase, are not actively in monitoring like “production plants” the alerts are important to notify about temperature/humidity issues



08

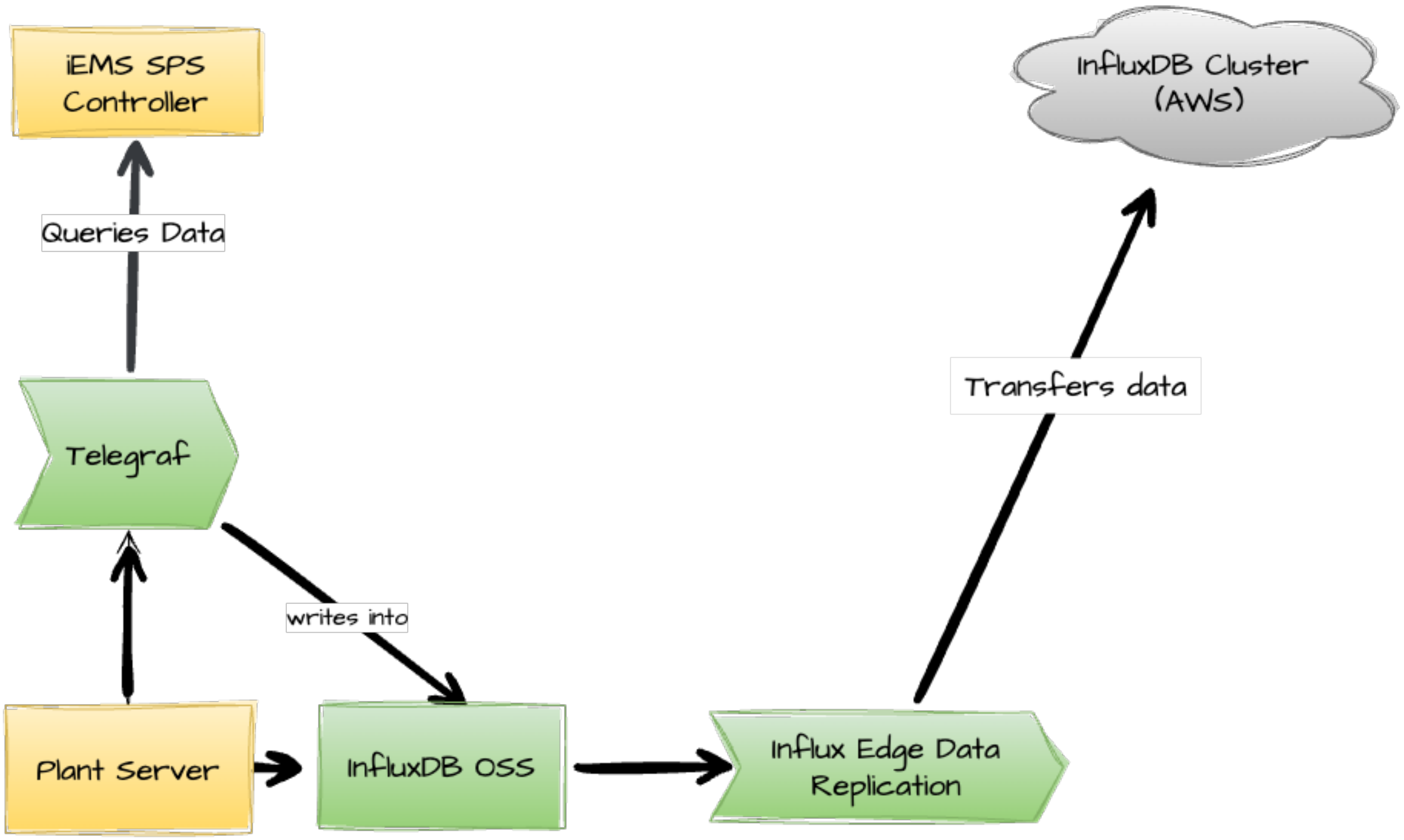
Gathering data with Telegraf from Modbus devices

ju:niz

Architecture Overview

Local Plant setup

- Each plant has local servers running influxDB OSS infrastructure
- Implementation of Telegraf to read Modbus data from devices directly
- Telegraf ”no code” solution allows agile development and deployments
- InfluxDB EDR takes care of reliable sending of data to the cloud



InfluxDB Solution Partner – B1-Systems

About B1-Systems

B1 Systems is a worldwide operating provider of Consulting, Training, Managed Service & Support around Linux/Open Source with more than 150 employees

Areas of expertise

- Container and Virtualization
- Public & Private Cloud
- High Availability Cluster
- Configuration & System Management
- Monitoring & Log Management

Products

- Client, Display & Server Management

Locations

- Rockolding
- Cologne
- Berlin
- Dresden
- Jena

Telegraf.conf file for getting the job done

Telegraf Setup

- Define Modbus Input Plugins
- Define Modbus registers for each sector of the plant
- Configure InfluxDB Output Plugin
- Configure – depending on plant size – multiple telegraf.conf files to allow parallel data querying
- **“No Code” solution without dependencies for modifications unlike SPS**

smareg_conf > SMAREG4 > telegraf.conf

```
  configuration_type = "request"

  [[inputs.modbus.request]]
    slave_id = 1
    optimization = "aggressive"
    #optimization_max_register_fill = 50
    byte_order = "ABCD"
    register = "input"

    fields = [
      { measurement = "Anlagendaten", name = "p_plant", type = "UINT32", scale=0.00000001, address = 10},
      { measurement = "Anlagendaten", name = "u1_2", type = "FLOAT32", scale=1.0, address = 12},
      { measurement = "Anlagendaten", name = "u1_3", type = "FLOAT32", scale=1.0, address = 14},
      { measurement = "Anlagendaten", name = "u2_3", type = "FLOAT32", scale=1.0, address = 16},
      { measurement = "Anlagendaten", name = "i1",type = "FLOAT32", scale=1.0, address = 18},
      { measurement = "Anlagendaten", name = "i2", type = "FLOAT32", scale=1.0, address = 20},
      { measurement = "Anlagendaten", name = "i3", type = "FLOAT32", scale=1.0, address = 22},
      { measurement = "Anlagendaten", name = "f_grid", type = "FLOAT32", scale=1.0, address = 24},
      { measurement = "Anlagendaten", name = "s_plant", type = "FLOAT32", scale=1.0, address = 26},
      { measurement = "Anlagendaten", name = "q_plant", type = "FLOAT32", scale=1.0, address = 28},
      { measurement = "Anlagendaten", name = "p_grid", type = "UINT32", scale=0.00000001, address = 30},
      { measurement = "Anlagendaten", name = "soc_plant", type = "FLOAT32", scale=0.1, address = 32},
      { measurement = "Anlagendaten", name = "t_amb_ems", type = "UINT8L", scale=0.1, address = 34},
      { measurement = "Anlagendaten", name = "t_amb_outside", type = "UINT8L", scale=0.1, address = 36},
      { measurement = "Anlagendaten", name = "t_it_cab", type = "UINT8L", scale=0.1, address = 38},
```

Grafana Visualization of BESS

Grafana Visualization

- Adding variables which allows flexible investigation of data
- No performance issues – if you check one battery or twenty
- Dashboards can be used for automated reporting
- Standardized dashboards allow quicker deployments and connecting new projects into the monitoring

ju:niz

Improving the Alert mechanism for both Generation BESS Systems

Grafana Multi-Dimensional Alerts

- Allows complex alert conditions based on multiple metrics
- Supports time-based conditions (e.g., alert if SOC < 20 for 15 mins)
- Provides greater context, reducing false positives

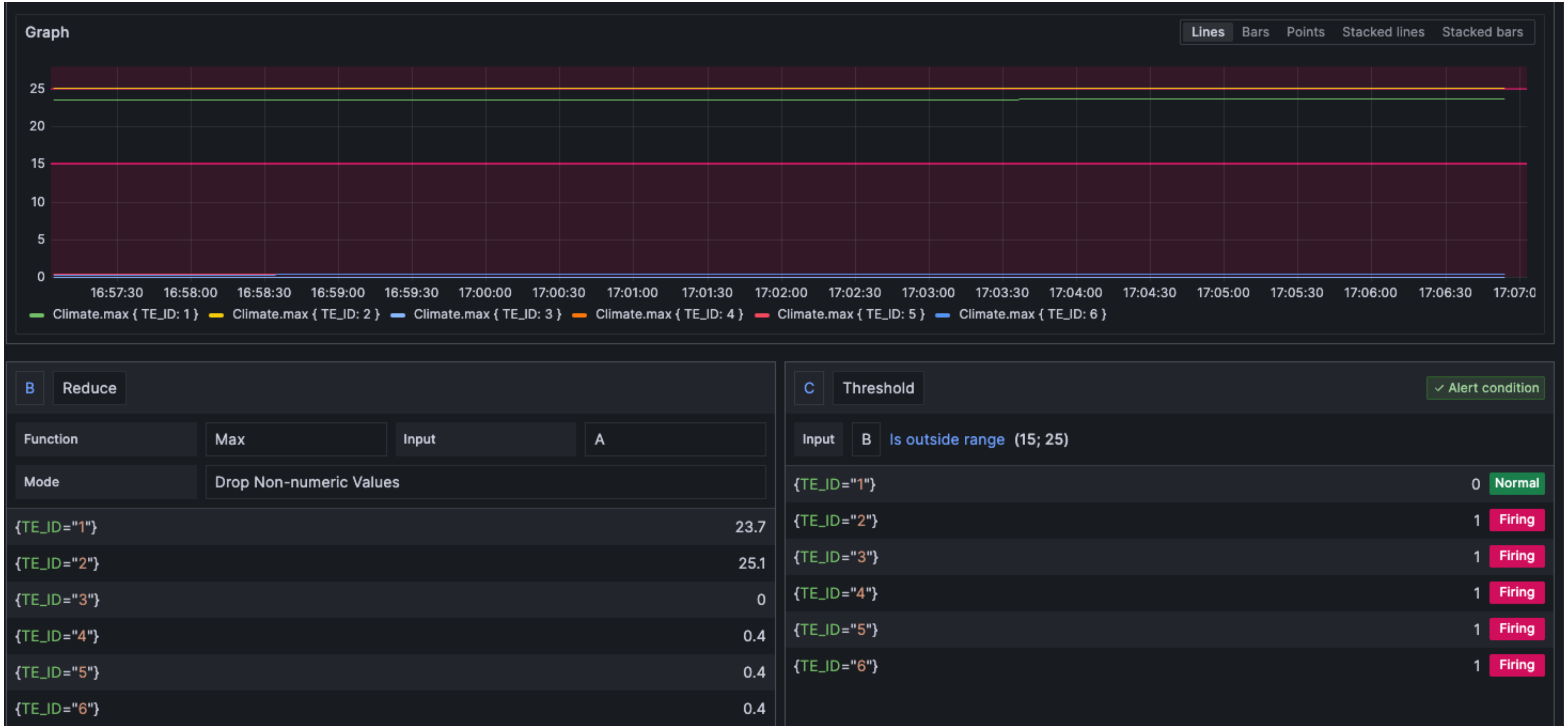
Sending Grafana Alerts Through Opsgenie

- Direct Integration of Grafana Alerts with Jira Ticket System
- Allows on-call scheduling and routing rules
- Allows additional notification in case of emergency, i.e. via SMS and phone call

Creating Jira Tickets via Opsgenie

- Auto-generate Jira tickets for specific alert conditions
- Streamlines incident management workflow

Grafana Alerts



ju:niz

Grafana Visualization of BESS

Grafana Visualization

- LIVE DEMONSTRATION – SWITCH TO BROWSER AND CONTINUE HERE LATER

Summary and next projects/steps

Getting Data with Telegraf

- Telegraf supports wide range of devices and communication protocols
- Allows easy integration – like influxDB Database – “no need to think about it twice”
- Supported and integrated in a wide range of products, i.e. our firewalls have a Telegraf plugin

InfluxDB Edge Data Replication

- Makes sending data stable even with unreliable network connections
- Reliable re-sending of failed metrics

InfluxDB Cloud Dedicated

- (Nearly) no need to think about storage costs and usage anymore
- Way better compression compared to influxDB Cloud V1 or OSS

Next projects and steps

- Evaluating communication between our SPS controllers and Telegraf with OPA-UA
- Evaluating communication between our SPS controllers and Telegraf with MQTT
- Integration of EPEX SPOT data
- Centralised cluster for analytics between batteries and market data

Wanna shape the

energy transition

We take responsibility for our actions, ask questions, even the uncomfortable ones, and courageously try out new approaches.

Where you can

get involved

Interested in pushing our shiny new InfluxDB Cloud Cluster to the max? Let's sit down for a coffee and discuss your path to joining the ju:niz team.😁

We're hiring like crazy – not limited to IT jobs! Find out about your next big move at ju:niz. 🚀

www.juniz.com

LinkedIn

ju:niz