Platone

PLATFORM FOR OPERATION
OF DISTRIBUTION NETWORKS

Platone

PLATform for Operation of distribution NEtworks

D2.6 v1.0

Platone DSO Technical Platform (v1)

The project PLATform for Operation of distribution NEtworks (Platone) receives funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no 864300.

Project name	Platone
Contractual delivery date:	28.02.2021
Actual delivery date:	28.02.2021
Main responsible:	Siemens SPA
Work package:	WP2 – Platform Implementation and Data Handling
Security:	P
Nature:	DEM
Version:	V1.0
Total number of pages:	24

Abstract

The Platone Open Framework aims to create an open, flexible and secure system that enables distribution grid flexibility/congestion management mechanisms, through innovative energy market models involving all the possible actors at many levels (DSOs, TSOs, customers, aggregators). The Platone Framework is an open source framework based on blockchain technology that enables a secure and shared data management system, allows standard and flexible integration of external solutions (e.g. legacy solutions), and is open to integration of external services through standardized open application program interfaces (APIs).

This document accompanies the software delivery of the Platone DSO Technical Platform and extends it with an architecture overview and the discussion of a demonstration setup.

The Platone DSO Technical Platform is part of the Platone Open Framework that will be integrated, tested and evaluated in three different demo sites in: Greece, Germany and Italy. Each of these demo sites will integrate different parts of the framework.

In particular, this version of the DSO Technical Platform will be integrated and validated within the Greek and German demo sites and their according architectures.

Keyword list

Platone DSO Technical Platform, Platone Framework, Open Source, micro-service, control centre, kubernetes

Executive Summary

The energy system is facing an incredible revolution whose end target is the creation of a new energy scenario widely dominated by renewable energy sources and mostly based on distributed energy generation. At the centre of this process is the distribution network where the majority of the new energy sources are and will be connected. Flexibility is a key resource in a scenario in which the grid is more and more changing from being a load-driven system to a generation-driven system, given the partial control on energy intake from renewable energy sources. This process implies also that the changes are not only related to the operational aspects but also to the market element. Digitalization is a key enabler of this process, opening the way to smart and efficient management of data sources in a secure way and making the separation between market and operation less and less meaningful.

The Platone project proposes an innovative approach for supporting the DSOs and other involved stakeholders in the energy transition phase. Platone aims to support the observability of the network and the exploitation of the flexibility to solve both the volatility of renewable energy sources and the less predictable consumption patterns.

The Platone solution consists of a layered set of platforms to meet the needs of system operators, aggregators and end users, named Platone Open Framework.

The Platone **DSO Technical Platform** is part of that framework and aims at enabling DSOs to fulfil market requests by evaluating the current grid state and activating local flexibility requests while ensuring the reliability and operational quality of service. Therefore, a micro service based platform architecture is presented that allows the deployment of services state-estimation and load prediction. Furthermore, the platform aims at an enlarged grid observability by providing a visualization of measured and predicted data.

Authors and Reviewers

Main responsible		
Partner	**Name**	**E-mail**
SIEM		
	Brunella Conte	brunella.conte@siemens,com
	Carlo Arrigoni	carlo.arrigoni@siemens.com
Author(s)/contributor(s)		
Partner	**Name**	
RWTH		
	Jonas Baude	
Reviewer(s)		
Partner	**Name**	
APIO		
	Alessandro Chelli	
ARETI		
	Gabriele Fedele	
Approver(s)		
Partner	**Name**	
RWTH		
	Padraic McKeever	

Table of Contents

1 Introduction

The project “PLATform for Operation of distribution Networks – Platone - aims to develop an architecture for testing and implementing a data acquisitions system based on a two-layer approach that will allow greater stakeholder involvement and will enable an efficient and smart network management. The tools used for this purpose will be based on platforms able to receive data from different sources, such as weather forecasting systems or distributed smart devices spread all over the urban area. These platforms, by talking to each other and exchanging data, will allow collecting and elaborating information useful for DSOs, transmission system operators (TSOs), customers and aggregators. In particular, the DSO will invest in a standard, open, non-discriminating, economic dispute settlement blockchain-based infrastructure, to give to both the customers and to the aggregator the possibility to more easily become flexibility market players. This solution will see the DSO evolve into a new form: a market enabler for end users and a smarter observer of the distribution network. By defining this innovative two-layer architecture, Platone removes technical barriers to the achievement of a carbon-free society by 2050 [1], creating the ecosystem for new market mechanisms for a rapid roll out among DSOs and for a large involvement of customers in the active management of grids and in the flexibility markets. The Platone platform will be tested in three European trials in Greece, Germany and Italy and within the Distributed Energy Management Initiative (DEMI) in Canada. The Platone consortium aims to go for a commercial exploitation of the results after the project is finished. Within the H2020 programme “A single, smart European electricity grid” Platone addresses the topic “Flexibility and retail market options for the distribution grid”.

The Platone solution consists of a two-layer architecture named **Platone Open Framework** (cf. Figure 1) The Platone Open Framework includes the following three layers:

Blockchain Service Layer**:** this layer enables the deployment of different blockchain-based components, providing a blockchain infrastructure and Smart Contracts services. In the context of Platone, the Platone Market platform is an example of blockchain-based platform deployed on the Blockchain Service Layer.

Blockchain Access Layer**:** this layer adds a further level of security and trustworthiness to the framework. It is an extension of the physical infrastructure and performs multiple tasks, among which are data certification and automated flexibility execution through Smart Contracts. It includes the Blockchain Access Platform and the Shared Customer Database.

Platone DSO Technical Platform: it allows DSOs to manage the distribution grid in a secure, efficient and stable manner. It is based on an open-source extensible microservices platform and allows to deploy, as Docker containers, specific services for the DSOs and execute them on kubernetes. The Data Bus layer, included on the DSO Technical Platform, allows integration both of other components of the Platone framework and of external components (e.g. DSO Management System) with a direct connection to the classical supervisory control and data acquisition (SCADA) system adopted by the DSO and served by standard communication protocols.

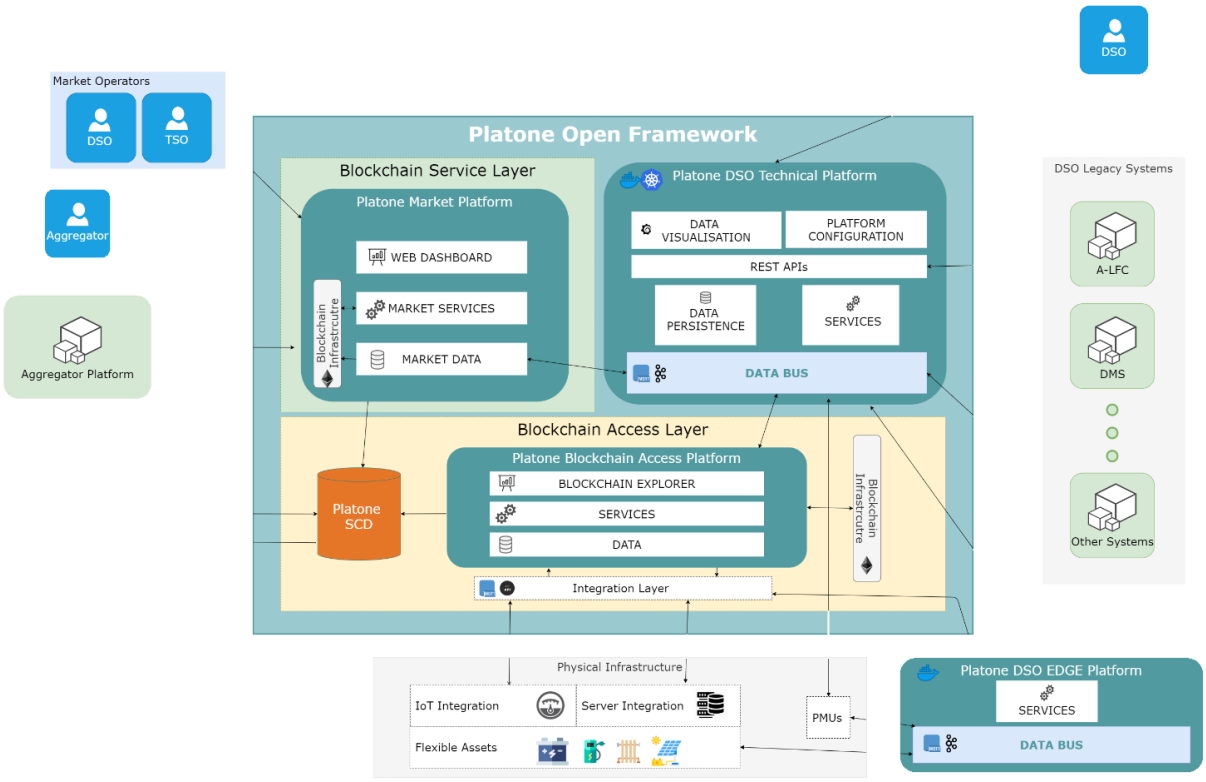

Figure 1 Platone Open Framework

1.1 Task 2.3

This deliverable is related to the Task 2.3 that aims at the implementation of a DSO Technical Platform, which allows a DSO to fulfil market requests by evaluating the current grid state and activating local flexibility requests while ensuring the reliability and operational quality of service. Therefore, a micro service based platform architecture is presented that allows the deployment of services state-estimation and load prediction. Furthermore, the platform aims at an enlarged grid observability by providing a visualization of measured and predicted data.

1.2 Objectives of the Work Reported in this Deliverable

The objective of this deliverable is to present the architecture of the Platone DSO Technical Platform and its realization by standard components that allows for the integration of custom DSO system services. The Platone description of action defines this deliverable as a demonstrator. This document accompanies the code repository with a more detailed architecture description as well as some extended deployment instructions for deploying, testing and integrating the platform.

1.3 Outline of the Deliverable

The second Chapter of this document describe the first realization of the Platone DSO Technical Platform according to the specification provided in Deliverable D2.1 [1] and discusses a data visualization stack in more detail, as well as the functionalities expected for the first release of the platform. Chapter 3 provides a brief overview of Interfaces and Communication Mechanisms. Chapter 4 delivers a compilation of Languages, Technologies and External Tools used throughout the platform. Chapter 5 is closely linked to the software delivery and provides detailed installation, setup and configuration instructions. Finally, Chapter 6 concludes this deliverable.

1.4 How to Read this Document

This document reports the software delivery of the Platone DSO Technical Platform, which is part of the Platone Open Framework that is implemented within WP2 of Platone. For a better understanding of the open framework of platform and requirements for the platform, we recommend reading D2.1 [1].

Besides that, this document provides a condensed version of the platform documentation and relates it to the formal design of the platform.

2 Platform Architecture

This chapter presents the architecture of the Platone DSO Technical Platform. The first section provides a formal description of the architecture and the components that are already available in the first release version. The second section provides a list of functions that are available as services and additional services that Platone will develop on top of the first release version. Furthermore, a visualization stack for PMU data visualization is presented in more detail as an exemplary basic use case of the platform and as a tutorial.

2.1 Overview

The Platone DSO Technical Platform enables distribution system operators to fulfil market requests by evaluating the current grid state and activating local flexibility requests while ensuring the reliability and operational quality of service by enlarged grid observability. The platform design builds on previous work done in the Horizon 2020 project *SOGNO* [10] and relies massively on a micro-service architecture.

The presented platform architecture aims at facilitating the transition to modular, micro-services based control centre software solution for distribution system operators. This allows for faster adjustment and independent development of components. The goal is to provide system operators and automation software developers with an open source framework that exposes open APIs to plug in new automation functions and supports industry standards such as CIM IEC61970 and IEC61850.

To address requirements such as high availability, scalability and modularity from the very beginning, the DSO Technical Platform is designed for deployment on kubernetes [11] clusters. Kubernetes, also known as K8s, is an open-source system for automation deployment, scaling and management of containerized applications. As all microservices of the platform are per requirement containerized in Docker [16] containers, they can easily be deployed on a kubernetes cluster. Kubernetes also simplifies different deployment approaches: from edge- and public-cloud to on-premises installation. However, the on-premises installation is considered the most relevant for a control centre platform. In order to minimize initial hurdles, Platone provides detailed installation manuals for a local installation based on the lightweight kubernetes distribution k3s [3].

Figure 2 illustrates the architecture of the DSO Technical Platform. The Databus is one of its core components and is implemented by means of a message broker to which all services can publish and / or subscribe in order to exchange data with other services, with field devices, or with external systems. Field devices or external systems can be made available in the data bus either directly or through the Platone Blockchain Access Layer [2].

Databus

The Databus is the core component of the DSO Technical Platform and is implemented as an MQTT message broker. The first release of the platform uses the RabbitMQ [5] message broker with an MQTT plugin. The Databus is accessible for all services within the platform and can also be exposed to outside the cluster for integration of external devices, platforms or services. It shall be used for asynchronous transfer of streaming data from field devices (e.g. PMUs) or setpoints.

Services

All services are able to connect to the Databus and may provide individual RESTful APIs as an interface for other services or to be exposed to external clients (users or systems). All services for the platform are deployed in Docker containers and restricted to the usage of the specified communication protocols. The programming languages are not specified to allow a wide flexibility for service developers.

API Gateway

The API Gateway is responsible for routing API requests from clients (i.e. users or external systems) to the related services while ensuring authentication, authorization, and security policies. Besides, data

transformations on the transfer. The first release of the DSO Technical Platform uses the basic functionality of the kubernetes ingress reverse proxy.

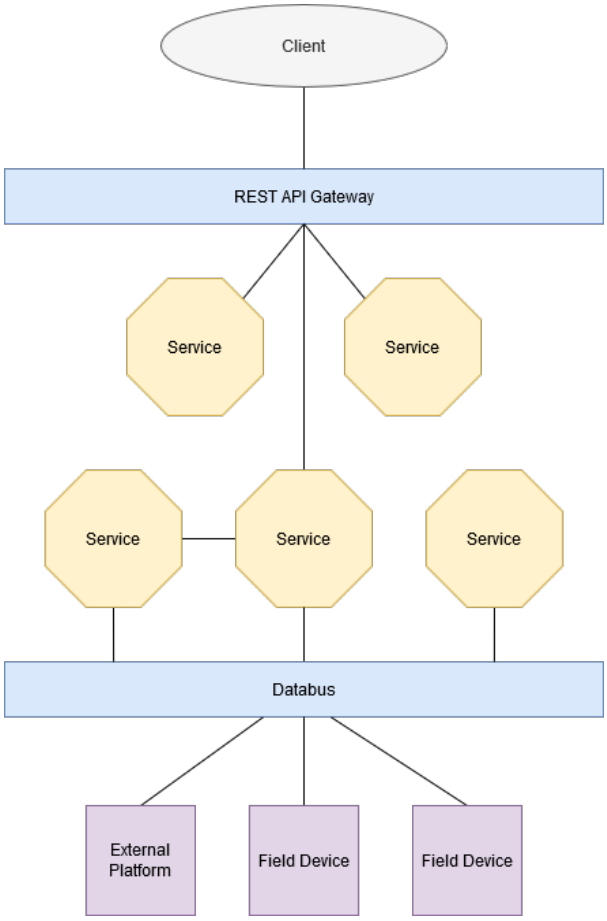

Figure 2 Platone DSO Technical Platform Architecture

2.2 Functionalities

The DSO Technical Platform supports a preliminary set of automation functionalities will be provided together with the framework based on the outcome of the SOGNO project:

- A **Load/generation prediction service**, which can provide detailed information about the future power demands and generated power to support planning and operation the power network.
- A **state-estimation service** that can calculate power loads at additional network nodes without installing measurement devices at that location.
- A FLISR (**Fault Location Isolation and Service Restoration**) service for autonomous self-healing of the grid.
- A data **visualization** stack.

The integration of the following services is foreseen by the Platone project:

- **Probabilistic load forecasting**: we foresee the integration of a machine-learning based forecasting module that provides more advanced load predications.

- **balancing of local energy community:** we foresee the integration of a balancing service that is capable of either minimizing the total energy consumption of an energy community or to handle energy supply and exchange in bulk packages.

PMU Data Visualization Stack

As a demonstrator and validation for the platform architecture, we present a PMU data visualization stack on the DSO Technical Platform. Figure 4 shows the concrete realization of the PMU data visualization stack. Besides the default Databus and gateway components three services are used:

- time series database
- database adaptor
- visualization service

In this example scenario, the PMU collects measurements from the grid and publishes them to the data bus. The database adaptor reads the measurements from the Databus and stores them in a time series database. The third service visualizes the data in a customizable dashboard. Optionally, the API of the database can also be exposed (dashed line) for external clients. Figure 3 shows an example of a Grafana dashboard plotting three different measurements over time. (The values are randomly generated by an emulated dummy PMU but traverse fully through the DSO technical platform as shown in Figure 4).

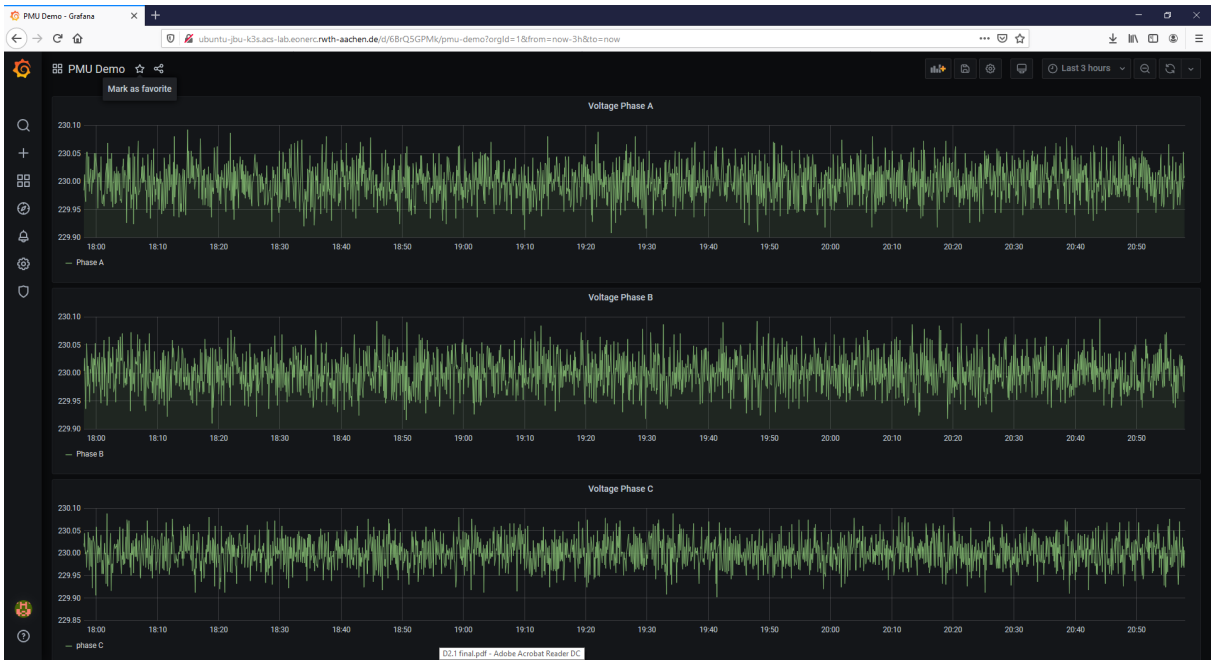

Figure 3 PMU Data Visualization Dashboard

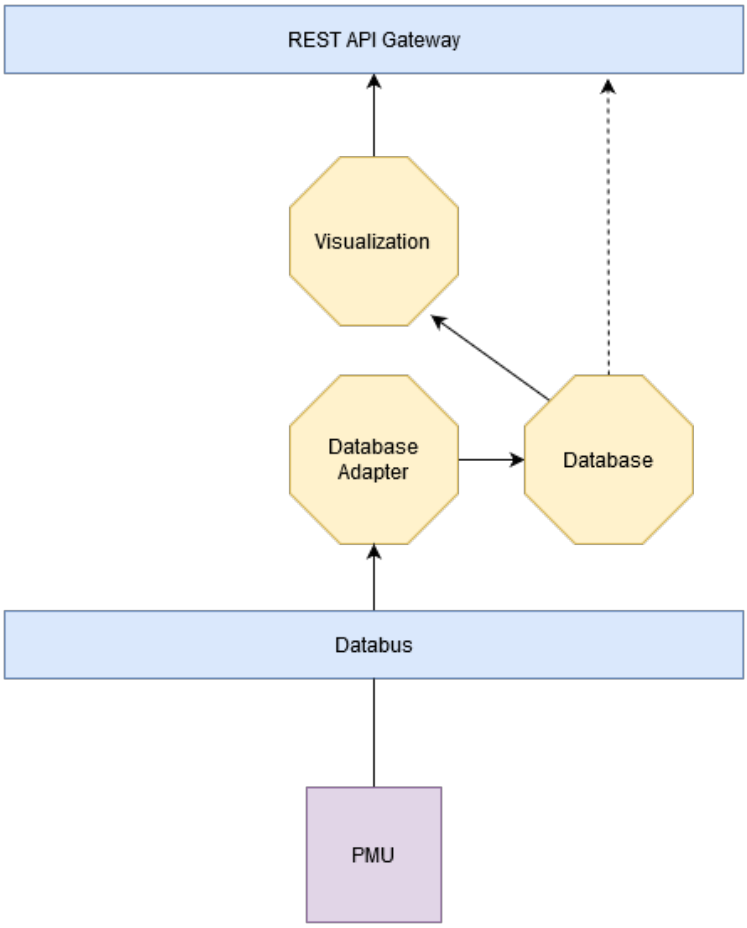

Figure 4 PMU Data visualization stack

2.3 Data Models

Work on definition and selection of data models is ongoing within the project consortium and the different trial sites. The PMU visualization example uses a custom *json* format for PMU data that has successfully been used for PMU data in previous research projects (cf. Figure 5). Besides that, we foresee the integration of the DSO Data Server [2] in the Greek Demo (WP 4) that provides meter readings according to the CIM industry standard CIM IEC 61968-9.

```
{
    "device": "pmu-0123",
    "timestamp": "TIMESTAMP",
    "readings": [
        {
            "component": "BUS1",
            "measurand": "voltmagnitude",
            "phase": "A",
            "data": "RANDOM"
        },
        {
            "component": "BUS1",
            "measurand": "voltmagnitude",
            "phase": "B",
            "data": "RANDOM"
        },
        {
            "component": "BUS1",
            "measurand": "voltmagnitude",
            "phase": "C",
            "data": "RANDOM"
        },

        /* ... more readings can go here */

    ]
 }
```

Figure 5 PMU Device Data

3 Interfaces and Communication Mechanisms

This chapter provides an overview of the interfaces to the DSOTP and between its microservices (cf. Chapter 2.1).

3.1 API Interfaces

This first release of the DSO Technical Platform does not expose custom REST APIs yet. For first integrations and development setups, the standard REST API of the underlying database could be exposed to external systems or platforms (not yet decided).

Services that we currently foresee to expose REST APIs:

- Forecasting
- State-estimation
- Energy Community Balancing
- Timeseries DB

The API documentation of the timeseries DB is available at [17]. All coming custom service APIs in the DSO technical platform will be specified using OpenAPI [18] specification to ensure maintainability and reduce development overhead. A more detailed discussion will follow in D2.9 (cf. [20]).

3.2 Databus

The Databus is the core component of the DSO Technical Platform and is implemented as an MQTT message broker. For services running within the DSO Technical platform, the Databus exposes an MQTT 3.1.1 compliant MQTT broker on the default port 1883.

For external clients (devices, platforms, and development tools) the port can be exposed as a node port.

A description of the message topic structure etc. is planned for D2.9 (cf. [20]).

3.3 UI Interfaces

The first release of the DSO Technical Platform includes a Grafana [8] integration for visualization of the stored time series data.

4 Languages, Technologies and External Tools

The architecture of the DSO Technical platform consists of different open-source tools. The following table provides an overview over the core components.

Table 1 Languages, Technologies and External Tools

Layer/Component	Technologies/Framework	Deployment	Languages
Infrastructure	kubernetes (K8s, K3s) Helm Docker	bare-metal	
Databus	RabbitMQ	Helm Chart	
Timeseries Database	InfluxDB	Helm Chart	
Database Adapter	Telegraf	Docker image kubernetes deployment	Go
Visualization Service	Grafana	Helm Chart	

5 Packaging and Deployment

In order to ensure high availability, scalability and modularity, the DSO Technical Platform is designed for deployment on a kubernetes [11] cluster. Kubernetes, also known as K8s, is an open-source system for automation deployment, scaling and management of containerized applications. As all microservices of the platform are per requirement containerized in docker [16] containers, they can be easily deployed on a kubernetes cluster. Users that are already experienced with kubernetes and have access to a full-fledged kubernetes cluster (either on-premises, hybrid or on public cloud infrastructure) can deploy the DSO Technical Platform there. For other users, for development setups, or for edge cloud deployments we provide instructions for setting up a minimal kubernetes cluster based on k3s [3], a lightweight kubernetes distribution in the next section. All mentioned configuration files as well as the entire documentation are available at [9].

5.1 Infrastructure requirements

5.1.1 Prerequisites

The single node setup was tested on the following operating systems:

- Ubuntu Server 20.04 LTS
- Ubuntu Desktop 20.04 LTS

Hardware recommendations:

- Memory: 8 GB
- Disc: 32 GB

In order to run a single node setup, we need to install kubernetes and the helm package manager. k3s [3] is a powerful lightweight kubernetes installation that suits well for single node or edge deployments. Helm is a package manager for kubernetes. We aim at providing helm charts for simple deployment of the platform and all our services to make the installation as easy as possible. For alternative deployment options we also recommend documentation provided by [15].

5.1.2 K3s

First, we need to install a k3s single node kubernetes cluster. Please refer to the official website [3] for alternative installation methods. The simplest way is to use the official install script. Run the following command to download and execute the official installation script.

```
$ curl -sfL https://get.k3s.io | sh -
```

Afterwards, we can verify our installation was successful by running

```
# k3s kubectl get node
```

If the installation was successful, this should return a list of kubernetes nodes containing solely your host as a master node.

5.1.3 Helm

Helm [4] is a package manager for kubernetes and provides detailed installation instructions on the website. Again, the fastest way is to use the provided install script:

```
$ curl -sfL https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
| bash
```

Now, we can verify the helm installation.

```
$ helm --kubeconfig /etc/rancher/k3s/k3s.yaml list
```

Eventually, you might need to change the ownership of that file:

```
# chown -R $USER /etc/rancher/k3s/
```

Finally, we can simplify the usage by setting the environment variable KUBECONFIG to our k3s.yaml file. By default (following our tutorial), it should be located in /etc/rancher/k3s/.

```
$ export KUBECONFIG=/etc/rancher/k3s/k3s.yaml
```

5.2 Deployment

Once a kubernetes cluster and helm are available, the core components can be deployed. In all following examples, we will use the namespace "*dsotp*" for deploying platform components.

5.2.1 Databus

For deploying the Databus via helm, the respective helm charts must be added to the local repository. Afterwards, the RabbitMQ broker can be installed. The file rabbitmq_values.yaml contains specific helm values for configuring the broker.

```
$ helm repo add bitnami https://charts.bitnami.com/bitnami
$ helm repo update
$ helm install -n dsotp --create-namespace -f Databus/rabbitmq_values.yaml
   rabbitmq bitnami/rabbitmq
```

This proposed deployment exposes the Databus externally on port 31883 and can be used by devices and external services or platforms.

A simple `mosquitto_pub` and `mosquitto_sub` [19] are useful for testing the external MQTT connectivity. Simply run them **outside** the cluster. In case of doubt, the **`-v`** might come in handy.

```
$ mosquitto_sub -h [IP or FQDN] -p 31883 -t /dev/test -u admin -P admin
```

```
$ mosquitto_pub -h [IP or FQDN] -p 31883 -u admin -P admin -t /dev/test -f
test_data.json
```

5.2.2 Timeseries Database

For deploying the time series database, the respective helm charts must be added to the local repository. Afterwards, influxdb [6] can be deployed using the provided values file.

```
$ helm repo add influxdata https://influxdata.github.io/helm-charts
$ helm repo update
$ helm install influxdb influxdata/influxdb -n dsotp -f ts-database/influxdb-helm-
values.yaml
```

Once the service is deployed, it needs additional preparation in order to initialize the database. The name of the pod executing the influx container can be obtained by running:

```
$ kubectl --namespace dsotp get pods
```

With the pod name, it is possible to log in and run the influxdb cli

```
# kubectl --namespace dsotp exec -i -t [pod name] /bin/sh
$ influx
```

Create database and user telegraf [7] and grant access

```
> CREATE DATABASE telegraf
> SHOW Databases
> CREATE USER telegraf WITH PASSWORD 'telegraf'
> GRANT ALL ON "telegraf" TO "telegraf"
```

5.2.3 Database Adapter

Since telegraf does not support nested json messages very well at the moment, the telegraf GitHub repository was forked and extended [12] but while writing this report, there is no helm chart yet.

In order to deploy the database adapter, a kubernetes *configmap* [13] has to be created and afterwards the adapter can be deployed:

```
```

```
$ kubectl apply -k ts-adapter/
$ kubectl apply -f ts-adapter/telegraf-deployment.yaml
```

5.2.4 Visualization

Deploying the Grafana [8] visualization services completes the PMU Visualization stack.

```
$ helm install grafana stable/grafana -f visualization/grafana_values.yaml
```

After deploying the Grafana service, the dashboard password can be obtained from the service pod by:

```
$ kubectl get secret -n dsotp grafana -o jsonpath="{.data.admin-password}" |
base64
    --decode ; echo
```

The Grafana web dashboard will be accessible via the IP or FQDN of the Platform node or the load balancer of your full-fledged kubernetes.

6 Conclusion

The work done at this stage provided a first release of the Platone DSO Technical Platform that allows for the implementation of flexibility control and grid observability tools based on a micro-service oriented architecture.

The first release addresses the minimal requirements for integrating with other platforms of the Platone Open Framework and for the development of demo site dependent services. A detailed description for installation and configuration of platform components is provided to ensure the usability and the impact. Moreover, continuous collaboration with the open-source SOGNO [14] community will facilitate the interoperability with and integration of other services such as a real-time simulator for testing new services.

The use of cloud-native technologies such as kubernetes allows for a scalable and flexible deployment of the platform from a single node on-premises control-room software to a distributed, cloud-based solution that utilizes advantages of local edge-cloud deployments.

Furthermore, the open source approach will ensure a better outreach and re-use of the results as well as an increment of the impact of the Platone project on the scientific community and in particular on the energy stakeholders.

7 List of Tables

8 List of Figures

9 List of References

[1] European Commission, “2050 long-term strategy”, [Online]. Available: https://ec.europa.eu/clima/policies/strategies/2050_en

[2] Platone “D2.1 Platone Platform requirements and reference architecture (v1)” [Online]. Available: https://www.platone-h2020.eu/data/deliverables/864300_M12_D2.1.pdf

[3] “K3s” [Online]. Available https://k3s.io/

[4] “Helm – The package manager for Kubernetes” [Online]. Available https://helm.sh/

[5] “RabbitMQ” [Online]. Available https://www.rabbitmq.com/

[6] “InfluxDB Platform” [Online]. Available https://www.influxdata.com/products/influxdb-overview/

[7] “Telegraf Open Source” [Online]. Available https://www.influxdata.com/time-series-platform/telegraf/

[8] “Grafana Cloud” [Online]. Available https://grafana.com/oss/grafana/

[9] “GitLab of the RWTH Aachen University” [Online]. Available https://git.rwth-aachen.de/acs/public/deliverables/platone

[10] “SOGNO energy: Service Oriented Grid for the Network of the Future” [Online]. Available https://www.sogno-energy.eu/

[11] “Kubernetes open source” [Online]. Available https://kubernetes.io/

[12] “Telegraf-sogno” [Online]. Available https://git.rwth-aachen.de/acs/public/third-party/telegraf-sogno/

[13] “Kubernetes ConfigMaps” [Online]. Available https://kubernetes.io/docs/concepts/configuration/configmap/

[14] “Sogno documentation” [Online]. Available https://sogno-platform.github.io/docs/

[15] “LF Energy – Sogno” [Online]. Available https://www.lfenergy.org/projects/sogno/

[16] “Docker: Empowering App Development for Developers” [Online]. Available https://www.docker.com/

[17] “InfluxDB API reference - InfluxData Documentation” [Online]. Available https://docs.influxdata.com/influxdb/v1.8/tools/api/

[18] “OpenAPI Initiative” [Online]. Available https://www.openapis.org/

[19] “Eclipse Mosquitto” [Online]. Available https://mosquitto.org/man/mosquitto_sub-1.html

[20] Grant Agreement No. 864300 – PLATONE

10 List of Abbreviations

Abbreviation	Term
DSO	Distribution System Operator
DSOTP	DSO Technical Platform
BLA	Blockchain Access Layer
MQTT	Message Queuing Telemetry Transport
REST	REpresentational State Transfer
API	Application Programming Interface
DB	Database
PMU	Phasor Measurement Unit
CIM	Common Information Model