EMS^2aaS: A Dockerized framework for remote EMS deployment

1st Giancarlo Marafioti
Dept. of Mathematics and Cybernetics
SINTEF Digital
Trondheim, Norway
giancarlo.marafioti@sintef.no

2nd Synne Fossøy
Dept. of Mathematics and Cybernetics
SINTEF Digital
Trondheim, Norway
synne.fossoy@sintef.no

3rd Johannes Philippus Maree
Dept. of Mathematics and Cybernetics
SINTEF Digital
Trondheim, Norway
phillip.maree@sintef.no

4th Iver Bakken Sperstad
Energy Systems
SINTEF Energy Research
Trondheim, Norway
iver.bakken.sperstad@sintef.no

***Abstract*—Energy Management Systems (EMSs) are software tools that support the grid operators to monitor, control and optimize generation, transmission/distribution, and loads in a electrical grid. Microgrids ("small scale" grids) have the potential to facilitate the energy management of a local grid with distributed energy resources (DER). In this paper, a Dockerized framework for remote EMS deployment is proposed and illustrated. The deployment of an EMS on a real system is time-consuming and may be hard to implement. To overcome this challenge, this work draws inspiration from the Software as a Service (SaaS) business model to propose an *EMS^2aaS* framework. This paper thus aims at providing a starting point to researchers and developers that would need to implement and test an EMS on a real pilot facility.**

***Index Terms*—energy management system, Docker, framework, Software as a Service, energy microgrid, model predictive control**

NOMENCLATURE

BESS	Battery Energy Storage System
CIL	Computer-in-the-loop
DER	Distributed Energy Resource
EMS	Energy Management System
EMS^2aaS	EMS Software as a Service
GPU	Graphics Processing Unit
HWIL	Hardware-in-the-loop
LAN	Local Area Network
MOSIOP	MOdelling, SImulation, and OPtimization
POC	Point Of Connection
PV	Photovoltaic
SDK	Software Development Kit
(S)MPC	(Stochastic) Model Predictive Control
SoC	State of Charge
RES	Renewable Energy Source
SaaS	Software as a Service

This work is co-funded by CINELDI - Centre for intelligent electricity distribution, an 8 year Research Centre under the FME-scheme (Centre for Environment-friendly Energy Research, 257626/E20), and by ROME - Resilient and Optimal Micro-Energy-grid (RCN project number 280797). The authors gratefully acknowledge the financial support from the Research Council of Norway (RCN), the CINELDI partners, and thank Lede (previously Skagerak Nett) in particular for discussions and cooperation. Venkatachalam Lakshmanan is acknowledged for his contributions to the framework design.

I. INTRODUCTION

A. Motivation and Background

The Centre for Intelligent Electricity Distribution (CINELDI) is one of the centres for environmental-friendly energy research in Norway (FME) [1]. CINELDI works towards digitalizing and modernising the electricity distribution grid for higher efficiency, flexibility, and resilience. The research centre is led by SINTEF, one of Europe's largest independent research organisations. Part of SINTEF's work in CINELDI, and the ROME project [2], focuses on research and development efforts for developing an intelligent Energy Management System Software as a Service (EMS^2aaS) framework. The main idea behind the software as a service (SaaS) business model is to avoid installation and maintenance of hardware and software at the customer premises. Instead, the software (application) runs on the cloud and is delivered over the Internet. Important requirements for the customer are to have an appropriate Internet connection and a client (PC) to access the cloud application. In the context of this paper, the goal is to offer a cloud application (software delivered over the Internet) as a service to optimize the energy management of a particular microgrid. The EMS^2aaS framework illustrated in this paper consists of three main parts: (i) A cloud-based EMS solution that scales for microgrid hardware-in-the-loop (HWIL), and computer-in-the-loop (CIL) simulations; (ii) Probabilistic load forecast modelling using non-linear Bayesian regression [3]; and (iii) Stochastic Model Predictive Control (SMPC) as

978-1-5386-5541-2/18/$31.00 ©2018 IEEE

DOI of published version: 10.1109/SEST50973.2021.9543194

EMS [4]. This paper addresses the work related to (i), providing a foundation for the implementation and the deployment of the parts at (ii) and (iii).

B. Relevant Literature

Microgrids are seen as means to facilitate the management of a grid contingency and the local optimization of energy supply by controlling distributed energy resources (DER) [5]. In microgrids, one can broadly differentiate between generation and load DERs. In the former, DERs may include photovoltaic (PV) panels, wind turbines, diesel generators and energy storage systems. Loads, on the other hand, are classified as either critical or non-critical, with the latter being perceived as flexible loads. A generalization of microgrids is presented in Figure 1. Microgrids can either be operated in island mode with no electrical connection to a larger grid; or, grid-connected mode with a direct connection to the utility grid via a Point of Connection (POC). Note that an islanded microgrid can be illustrated as the non-islanded microgrid in Figure 1 without the POC to the utility grid.

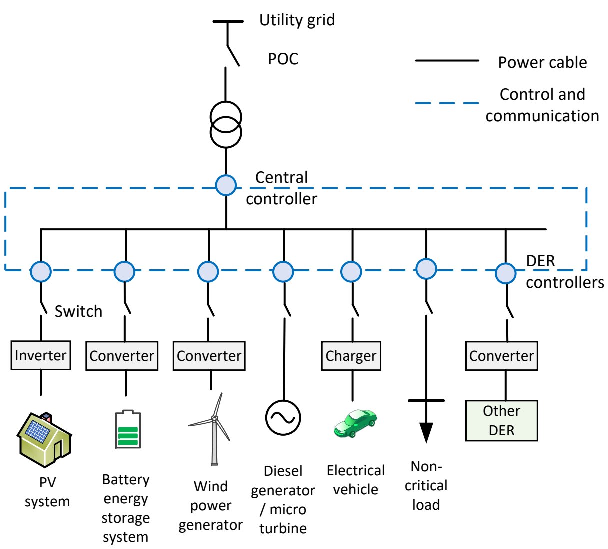

Fig. 1: Grid-connected microgrid (based on [5]).

In a microgrid, an energy management system (EMS) is essential for optimal use of the distributed energy resources in an intelligent, reliable, and coordinated way [6] [7] [8]. In CINELDI and the ROME project, Stochastic Model Predictive Control (SMPC) is proposed as an EMS for microgrids. An example of MPC-based EMS for microgrid is given in [9]. MPC is a receding horizon control approach, where an online optimization problem is solved sequentially over a receding horizon to find the optimal control input relative to some chosen cost function. A model is used to make predictions of the system's future behavior. MPC is useful for systems with constraints like for example upper and lower bounds on state of charge (SoC) in a battery energy storage system (BESS). Furthermore, the cost function allows defining a self-chosen metric to optimize i.e., one can minimize the cost of purchased energy from the utility grid. As a receding horizon control approach, MPC is useful for improving the controller robustness concerning model and prediction uncertainties and disturbances.

Researchers want to test and validate their developed EMS solutions (for instance MPC-based) on real systems (such as microgrids). However, the deployment on a real system is time-consuming and may be hard to implement. This is a likely reason why, to the authors' best knowledge, real-world deployment and demonstration of research-based EMS solutions [10] relatively rarely is reported in the research literature. The concept of Software as a Service (SaaS) is discussed in [11] which is a survey on different cloud computing applications for smart grids. Further, the challenges and opportunities of such a system are available in the study [12].

C. Contributions and Organization

An EMS^{2}aaS framework, as suggested in this paper, contributes to speed up deployment time and to easily switch between hardware-in-the-loop (HWIL), and computer-in-the-loop (CIL) simulations. This divides research-based implementations from the real system with the advantage to keep the two parts separated, as long as a clear interface is defined. Further, a general framework, as suggested in this paper, allows for easy adaptation to other systems/microgrids, and therefore, it may speed up the development time for future research.

This paper is organized as follows. The target system for the EMS^{2}aaS validation, Skagerak Energilab is presented in Section II. The software as a service framework is described in Section III and its basic setup is described in Section IV. The conclusion and further work of the paper are given in Section V.

II. The Skagerak Energilab pilot facility

The Skagerak Energilab [13] is a pilot facility used in CINELDI and, in this work, it is considered as the target system for the EMS^{2}aaS validation. More in detail, the Skagerak Energilab is a fully functional (virtual re-configurable) microgrid built around the Skagerak Arena soccer stadium in Skien, Norway. Previous research work based on the same physical system is available in [14] where a cost-benefit analysis for the operational strategies was carried out. In this previous work, however, system operation was simulated assuming perfect foresight for a full year of historic data, and load forecasts and uncertainties were not taken into account.

The microgrid of Skagerak Energilab is illustrated in Figure 2. The microgrid has an 800 kW peak power PV plant, covering an area of 4400 m^2 and with an expected production of around 650 000 kWh yearly, installed as a renewable energy source (RES). Further, the microgrid consists of a 1100 kWh capacity battery energy storage system with 800 kW peak charging/discharging power, several residential loads, and the Skagerak Arena loads. As seen in Figure 2, the battery and the floodlights are connected to a different substation than the PV plant, Skagerak Arena loads, and the residential loads. The residential loads consist of forty-two private buildings, nine businesses, and two cooperative residential buildings [15]. The Skagerak Energilab is by default connected to the main grid for supplying power in times when the battery and PV plant does not deliver sufficient power. However, the microgrid enables island mode for use-cases, where the connection to the main grid is disconnected.

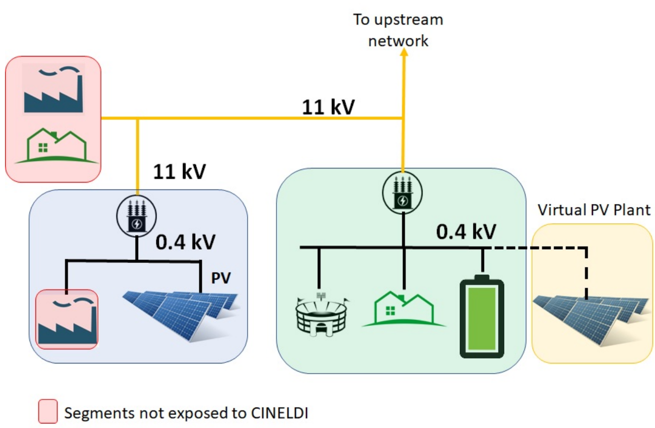

Fig. 2: An overview of the Skagerak Energilab microgrid, in Skien Norway.

III. Dockerized framework

Docker is selected as the main technology to build our software as a service framework. Docker is the de-facto standard to build and share containerized applications [16]. A container is a standardized unit packaging software for development, shipment, and deployment. A Docker container image (aka Docker image) is a standalone package including everything needed to run an application, i.e. code, tools, libraries, and settings. Docker images can be found in the collection of existing Docker images at Docker Hub or they can be designed for the specified application using a Docker file. Docker images running on a Docker engine become Docker containers, where a Docker container is an instance of the Docker image. An application/software deployed in a Docker container will have the same behavior independently of the operating system and hardware layer where the Docker engine is running. For example, a Docker container can run seamlessly on a Windows or a Linux machine, given that the appropriate Docker engine is installed.

Figure 3 illustrates the suggested architecture to apply the EMS^{2}aaS framework to the Skagerak Energilab pilot facility. The main idea is to use Docker to build a micro-service-oriented architecture [17] to achieve a reliable and easy-to-scale architecture, such as discussed in [18]. The SINTEF server runs the Modelling, Simulation, and Optimization (MOSIOP[1]) container and the processing container, in addition to a database with historical data. As several applications are designed and implemented to run on the SINTEF server, Docker Compose is used to define and run the containers for these applications. Docker Compose is a tool for multi-container applications, which simplifies the configuration of the containers and enables all containers to be started with one single command.

Note that in the Figure 3, there are two geographical regions, one at SINTEF premises in Trondheim (left side of the figure), and one at Skagerak Energilab in Skien (right side), both in Norway. Both SINTEF and Skagerak Energilab have local area networks (LANs), protected behind a firewall, and are connected through the cloud. As seen in Figure 3, the SINTEF developers and researchers can either test their algorithms on the actual pilot, a hardware-in-the-loop (HWIL) simulation, or on an emulator, a computer-in-the-loop (CIL) simulation. The Skagerak Energilab emulator is in short a virtual representation of the actual Skagerak Energilab pilot. An emulator enables researchers and developers to test and verify their work without the need to deploy them on the actual pilot.

Communication between SINTEF and either the Skagerak Energilab or the emulator is sent through the big data streaming platform, Azure Event Hub [19]. Examples of communication are the current state of charge (SoC) data transfer from the BESS, or current production from the PV plant, or optimal rate of charge or discharge reference signals from the EMS to the BESS. The Azure Event Hub is a cloud-based service and it provides client software development kits (SDKs) for several programming languages like Java, C++, Python, and Go. As a cloud-based platform, Azure Event hub allows events to be streamed and received to the pilot facility, Skagerak Energilab, with minimal implementation demands required by the EM^{2}SaaS framework. This is valid also for the Skagerak Energilab site, therefore with minimal implementation demands required, the possible violation of the pilot facility's security policies is minimized. Further, the cloud-based approach makes it indifferent for the SINTEF server whether it communicates with the emulator or the actual plant.

The Azure Event hub platform can receive and process millions of events each second and provides backup solutions to avoid information loss. Each event contains a measurement of a single signal at a given time instance. All the received measurements from the Event Hub are processed by the processing container running on the SINTEF server. After that, all processed measurements are saved in the historical database running on another container in the same SINTEF server.

The SINTEF server runs the modelling, simulation, and optimization (MOSIOP) container. The container is used to model the assets at Skagerak Energilab, i.e., the PV plant or the BESS. These models are used by the SMPC-based EMS,

[1] In this container we run the SMPC-based EMS algorithm; more information about the SMPC formulation is available in an accompanying paper also submitted for dissemination [4].

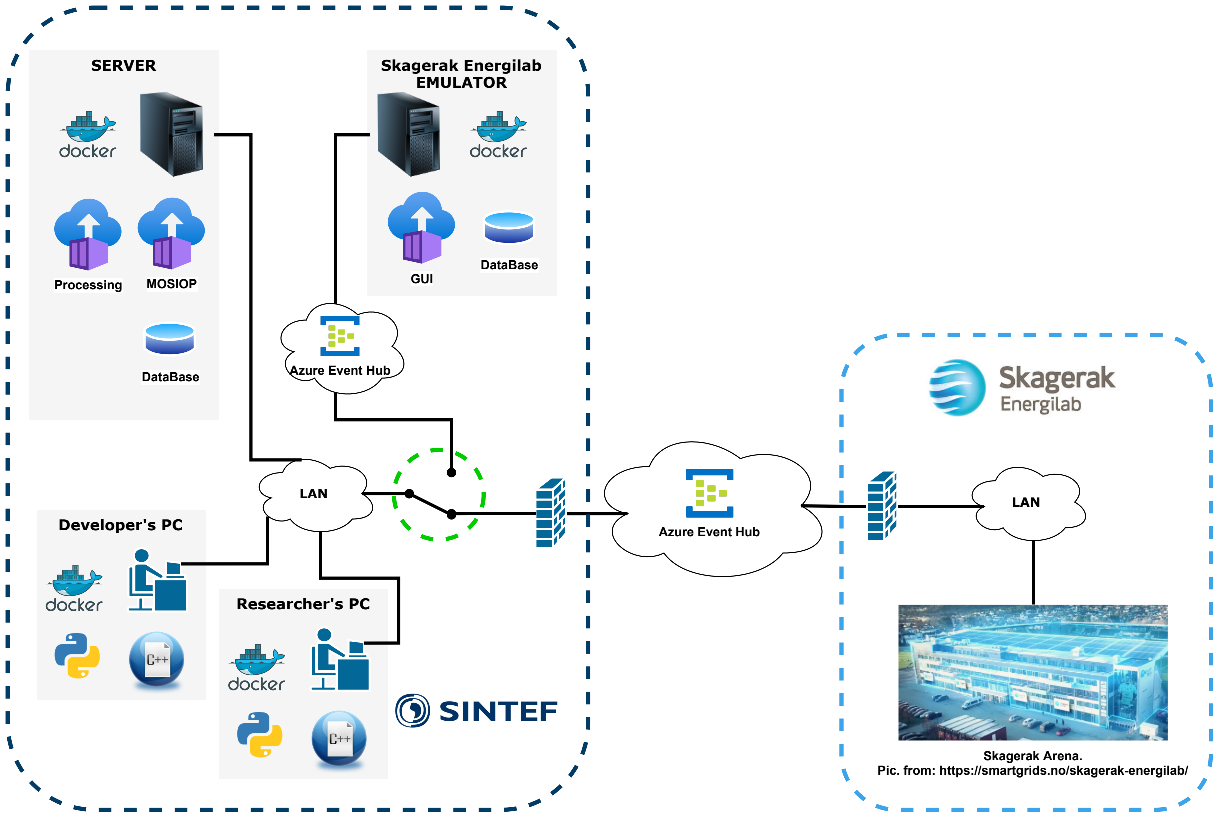

Fig. 3: Suggested high-level architecture for the EMS^{2}aaS when applying it to the Skagerak Energilab pilot facility. The green dashed circle illustrates the possibility to switch between the CIL and HWIL mode.

therefore simulation and optimization along the SMPC receding horizon are running here. The SMPC can interface with the probabilistic load forecasting models[2], trained on data from the historical database, that samples a set of scenarios (time series) of load demand and PV generation for the operational planning horizon of the SMPC. Figure 4 schematically illustrates the data flow and interfaces between the models and algorithms that are incorporated in the EM^{2}SaaS framework.

The SMPC could potentially benefit from the use of Graphics Processing Units (GPUs) for performing Monte Carlo simulations for sample-based SMPC approaches [20]. It is envisioned that the proposed framework could easily be deployed on available GPU cloud computing services [21], or on proprietary servers with dedicated GPU resources. Finally, note that with small configuration changes one could run these three (or even more) Docker containers on two or more different servers.

The database for historical data on the SINTEF server is a TimescaleDB database [22], where each communicated signal is assigned a separate data table. TimeScaleDB is a PostgreSQL database for time-series, where PostgreSQL is an open-source, object-relational database [23]. An object-relational database includes the benefits of object-oriented approaches into a relational database [24]. This makes for a more powerful and advanced database with more possibilities for in-build objects and complex procedures, compared to a relational database like MySQL. MySQL is the most popular relational database [25] and it can be useful to be used for comparison. The TimeScaleDB database includes features such as specialized functions related to time, date, and timezone and faster queries. The database is deployed as a container using the TimescaleDB Docker image. Deploying the database as a container makes deployment simple and attainable for inexperienced users and it contributes to the containerized approach of the whole architecture.

[2]For more information about the probabilistic load forecasting we refer to an accompanying paper also submitted for dissemination [3].

IV. EMS^{2}AAS FRAMEWORK SETUP

As mentioned in the previous section, Docker Compose is used to orchestrate the respective containers, which are processing, MOSIOP, and DataBase, on the SINTEF server. The configuration of the required services is defined in a YAML file, where a service can consist of one or more of the same container. The YAML file used for deployment of the EMS^{2}aaS framework is given below as:

```
version: '3'
services:
  postgresql:
    container_name: cineldi_postgresql
    image: timescale/timescaledb-postgis:latest-pg11
    ports:
      - 5433:5432
    volumes:
```

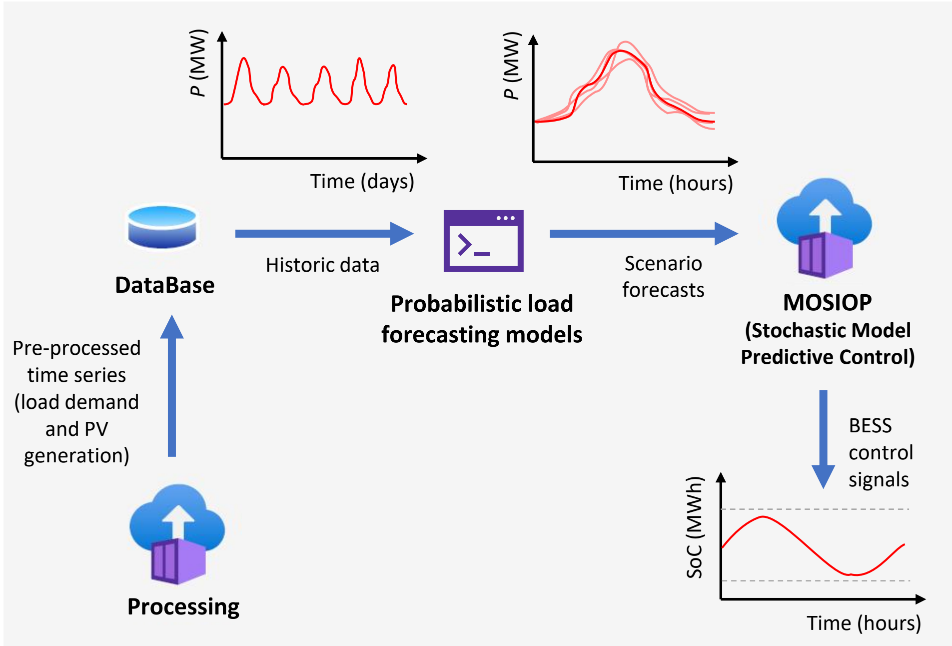

Fig. 4: Schematic of data flow and interfaces between models and algorithms incorporated in the EM^{2}SaaS framework.

```yaml
      - /etc/localtime:/etc/localtime:ro
      - /<location>/data:/var/lib/postgresql/data
      - /<location>/etc:/etc/postgresql
      - /<location>/log:/var/log/postgresql
    env_file:
      - database.env # configure postgres
    restart: unless-stopped

  azure:
    container_name: azure_client
    image: emssaas/azure_client:latest
    depends_on:
      - postgresql
    environment:
      - TZ=Norway/Trondheim
    restart: unless-stopped

  mosiop:
    build:
        context: .
        args:
         COINHSL_SRC_PATH: ./coinhsl-2019.05.21.tar.gz
         CONAN_SRC_PATH: ./conan
    privileged: true
    ports:
        - 'SERVER_IP_ADDRESS:2222:22'
    cap_add:
        - SYS_PTRACE
    volumes:
        - '$HOME/.Xauthority:/root/.Xauthority:rw'
        - /<location>:/<location>
    environment:
        - DISPLAY
    image: 'cineldi/mosiop:latest'
    tty: true
```

Listing 1: The YAML file used by Docker compose to deploy the energy management framework. <location >is any chosen file location on the SINTEF server.

The YAML file consists of three different services, PostgreSQL for the database container, azure for the processing container, and mosiop for the MOSIOP container. The images of the database and processing container are found at Docker Hub, while the MOSIOP container image is built by using a Dockerfile. All containers deployed using Docker Compose are automatically connected through a default network, which ensures that establishing communication between the launched containers is a non-issue. Services/Container, like for instance PostgreSQL, can be equipped with ports. Ports are useful for communication with software and standalone containers not connected to the Docker Compose network. By defining ports, one can easily determine which services can be reached from the outside or not, which is useful for shielding safety-critical procedures and for having control of the communication flow. As seen in the YAML file, services can have associated volumes. A Docker volume is a standalone ”memory location” hosted and managed by the Docker engine, independently of the local PC. Volumes can contain information and data needed and/or saved by containers. Several containers may access and manage the same volume. In the YAML file, the volumes are assigned to a local location on the SINTEF server to ensure that the volumes' content is preserved, when and if containers or volumes are stopped or deleted. To deploy all the services defined in a YAML file, only the following command is required:

```
$ docker-compose up -d
```

Note that Docker Compose only deploys changed, deleted, or stopped containers to minimize the efforts if some of the containers are already running.

V. Conclusion and Further Work

This paper has proposed an EMS^{2}aaS framework being developed at SINTEF with the ultimate goal of deploying and demonstrating an EMS remotely. This work contributes to the CINELDI efforts towards digitalizing and modernising electricity distribution. The short-term goals that have been defined were: i) to allow researchers and developers to easily access data from a microgrid pilot facility; ii) to use these data to build prediction models; iii) to develop and test EMS algorithms and prototypes against the Skagerak Energilab emulator. These three goals are achieved at the time of submission for this paper, and the paper presents the current status of the demonstration. Further work is needed to close the loop with the actual microgrid pilot facility (Skagerak Energilab). This is essential to achieve the ultimate goal and to validate the EMS. Validation will involve sending the control/reference signals to the actual pilot facility instead of the emulator, as illustrated in the green dashed circle in Figure 3.

Finally, the application of relevant technologies (Docker, Azure Event Hub, and PostgreSQL) for implementing the framework was illustrated. This illustration was not meant to be a thorough review (or tutorial) of these technologies; rather, the objective was to provide a starting point, as well as simplifying the work (by giving simple setup guidelines), to other researchers and developers that would need to implement and test an EMS on a real pilot.

REFERENCES

[1] SINTEF. (2020) Centre for Intelligent Electricity Distribution. Accessed: 2020-12-11. [Online]. Available: https://www.sintef.no/projectweb/cineldi/

[2] RCN. (2018) Resilient and Optimal Micro-Energy-grid (ROME). Accessed: 2020-12-11. [Online]. Available: https://prosjektbanken.forskningsradet.no/#/project/NFR/280797/Sprak=en

[3] M. Löschenbrand, S. Gros, and V. Lakshmanan, "Generating scenarios from probabilistic load forecasts via non-linear Bayesian regression," in *4th International Conference on Smart Energy Systems and Technologies (SEST 2021)*, (Paper submitted).

[4] J. P. Maree, S. Gros, and V. Lakshmanan, "Low-complexity Risk-averse MPC for EMS using reinforcement learning," in *4th International Conference on Smart Energy Systems and Technologies (SEST 2021)*, (Abstract submitted).

[5] *Microgrids Part 2: Guidelines for Operation*, https://webstore.iec.ch/publication/29578, 9 2018.

[6] M. F. Zia, E. Elbouchikhi, and M. Benbouzid, "Microgrids energy management systems: A critical review on methods, solutions, and prospects," *Applied Energy*, vol. 222, pp. 1033–1055, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261918306676

[7] L. Eun-Kyu, W. Shi, R. Gadh, and W. Kim, "Design and implementation of a microgrid energy management systems," *Sustainability*, vol. 8(11), p. 1143, 2016.

[8] W. Su and J. Wang, "Energy management systems in microgrid operations," *The Electricity Journal*, vol. 25, no. 8, pp. 45–60, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S104061901200214X

[9] A. Parisio, E. Rikos, and L. Glielmo, "A model predictive control approach to microgrid operation optimization," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 5, pp. 1813–1827, 2014.

[10] A. Parisio, E. Rikos, G. Tzamalis, and L. Glielmo, "Use of model predictive control for experimental microgrid optimization," *Applied Energy*, vol. 115, p. 37–46, 2014.

[11] S. Bera, S. Misra, and J. J. P. C. Rodrigues, "Cloud computing applications for smart grid: A survey," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1477–1494, 2015.

[12] J. L. Schaefer, J. C. M. Siluk, P. S. d. Carvalho, J. Renes Pinheiro, and P. S. Schneider, "Management Challenges and Opportunities for Energy Cloud Development and Diffusion," *Energies*, vol. 13, no. 16, p. 4048, Aug. 2020. [Online]. Available: https://www.mdpi.com/1996-1073/13/16/4048

[13] Skagerak Nett. (2020) Skagerak Energilab. Accessed: 2020-12-08. [Online]. Available: https://www.skagerakenergilab.no/forside/

[14] K. Berg, M. Resch, T. Weniger, and S. Simonsen, "Economic evaluation of operation strategies for battery systems in football stadiums: A norwegian case study," *Journal of Energy Storage*, vol. 34, p. 102190, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352152X20320119

[15] E. K. Skarholt and A. Vornicov, "Empirical Comparison of Load Forecasting Methods for Skagerak Energilab," Master's thesis, Norwegian school of Economics, 2020.

[16] Docker. (2020) Why Docker. [Online]. Available: https://www.docker.com/why-docker

[17] J. Turnbull, *The Docker Book*, 18092nd ed. James Turnbull, 2014.

[18] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Microservice Based architecture: Towards High-Availability for Stateful Applications with Kubernetes," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, 2019, pp. 176–185.

[19] Microsoft Azure. (2021) Event Hubs: Real-time data ingestion. [Online]. Available: https://azure.microsoft.com/en-gb/services/event-hubs/

[20] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, 2016.

[21] Nvidia. (2020) GPU cloud computing. [Online]. Available: https://www.nvidia.com/en-us/data-center/gpu-cloud-computing/

[22] Timescale. (2021) Time-series data simplified. [Online]. Available: https://www.timescale.com/

[23] The PostgreSQL Global Development Group. (2021) PostgreSQL: The world's most advanced open source relational database. [Online]. Available: https://www.postgresql.org/

[24] R. Elmasr and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Pearson, 2016.

[25] Oracle. (2021) MySQL Documentation. [Online]. Available: https://dev.mysql.com/doc/