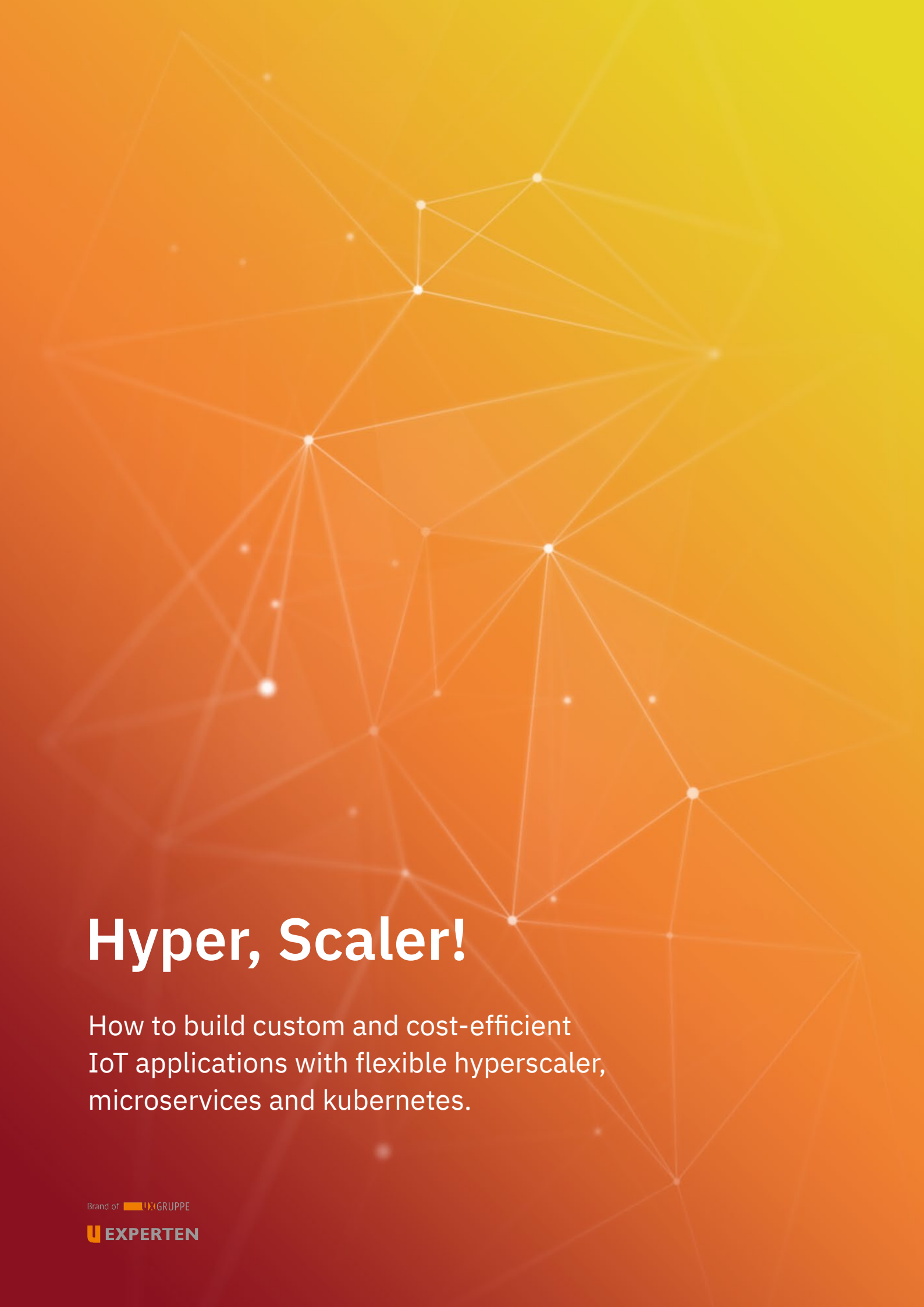


# Hyper, Scaler!

How to build custom and cost-efficient IoT applications with flexible hyperscaler, microservices and kubernetes.

Brand of UX GRUPPE
U EXPERTEN

# Content

# Executive Summary

Cloud-based services and the use of innovative technologies of the "Internet of Things" (IoT) offer new and attractive business opportunities for almost any market segment and application.

When vision and product roadmap of an IoT application are clear, technical decision-makers will have to choose a suitable technology at the very beginning of a development.

This paper assists in choosing the right technology for cutting-edge cloud projects to embrace flexibility and independence (without vendor lock-in).

We describe how to build an IoT cloud application, running on hyperscaler platforms like **AWS**, **Microsoft Azure**, or **Google Cloud Platform**, from scratch, illustrated by a successful case study of U-Experten.

The concept addresses typical and frequent needs of technology decision-makers:

- **Flexibility:** Keep the door open to switch the hyperscaler platform but be free to mix your services with hyperscaler services.
- **Customizability:** Build an individual application where ready-to-use solutions do not fit.
- **Extendability:** Start with a minimum viable product (MVP) and extend step by step.
- **Scalability:** Adopt the compute power for individual needs.

A major focus in the case study is **flexibility**.

Hyperscaler[1] offer a wide set of services designed to build IoT applications. However, applications may be bound to the hyperscaler platform (vendor lock-in) in the future. Updates and product terminations can eventually result in unforeseen and expensive maintenance tasks.

On the other side, developing on one specific hyperscaler platform reveals also advantages: The infrastructure is available and ready to use and - compared to multiple platform implementations - it requires less effort to maintain it.

Based on our case study, this paper describes a concept of how to start development using open source software components running on a cloud infrastructure, to achieve flexibility and independence (without vendor lock-in), while in parallel embracing the advantages of scalability of a "ready-to-use cloud platform", mentioned above.

1 The term „hyperscaler" is used synonymously for AWS, Microsoft Azure and Google Cloud Platform in this article.

# Let’s Start with a Case Study

## How Does Kubernetes & Microservices Match With the Perfect Day at the Lake?

Stand-up paddling (SUP) has become a huge trend in outdoor activities all over the world. However, pumping up such a board can be an exhausting endeavour as well as disturbing in public recreation spaces because of the noise exposed.

Based on these observations the start-up company AIR4SUP has designed a smart connected pumping station for SUP-boards together with U-Experten.

The challenge: The business case of AIR4SUP requires highly efficient fleet device management to deliver excellent service at reasonable prices for their users.

Therefore, several use cases and user stories have been developed around fleet management, e.g: “Collect operating data, collect weather data, send them to the cloud, store them in a database and make them accessible to the user.”

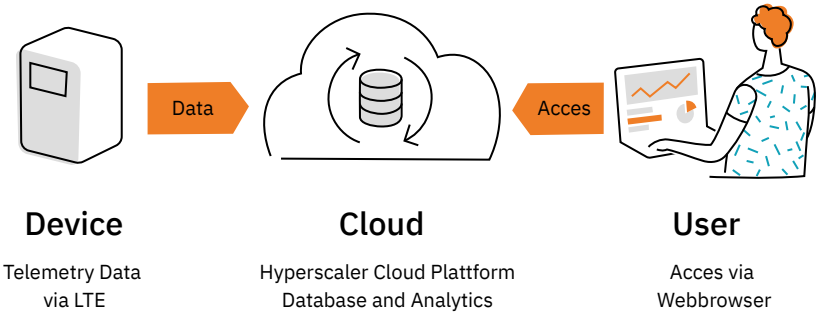



*Figure 1 offers an overview of a use case derived from the user story:*

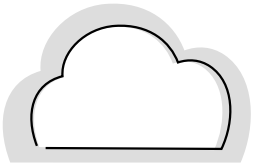
If you like to learn more about this case study, **please click here**

In the following, this case study will be used as a reference to illustrate our approach.

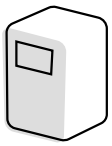
# Our Concept

## A Perfect Solution for a Project That Starts Small, but Already Thinks Big!

- data visualization
- data provisioning
- data post-processing
- data storage
- data pre-processing
- data reception



**Cloud Part**



**Device Part**

- data acquisition
- data pre-processing
- data pre-transmission

Typically, each IoT application requires customization over time. For future extensions or modifications the foundation needs to be flexible. At the same time, infrastructure costs should be predictable to provide operational cost-efficiency. For hassle free future growth, there also should be a strong focus on scalability.

To achieve these objectives sufficiently, the underlying concept requires open source software components both on the device side and cloud side to build up an IoT application on hyperscaler infrastructure from scratch.

Such applications can be deployed on most popular hyperscalers as dependencies to vendor-specific services are avoided. Concerning the underlying case study, the following requirements and objectives have been derived:

| | |
|---|---|
| **Flexibility** | • communication with the device shall be independent from the data / data format and shall be adoptable to individual use cases<br>• device data should be stored separately from other content or format<br>• the dependencies to hyperscalers should be kept at a minimum<br>• subsystems (e.g. database) shall be replaceable with alternatives<br>• an on-premise solution shall be foreseen<br>• an API for data provisioning shall be extendable |
| **Efficiency (OPEX)** | • data transmission shall be optimized<br>• time series data storage shall be optimized<br>• infrastructure costs for operation shall be foreseen and controllable |
| **Security by-design** | • communication between device & cloud shall be secured with state-of-the-art best practice methods<br>• communication between user & cloud shall be secured with state-of-the-art best practice methods. |
| **User Experience** | • the web-UI is individual and customizable to particular use cases for the end-user |

*Table 1: Basic requirements and objectives.*

# Technology Considerations

## Which Technology Components Do I Need to Launch a “Smart SUP Pump Station”?


### Device Components

The device-part includes electronic controls that mostly consist of two parts:

1. Baseboard with a dedicated CPU for handling pump operations
2. Linux-based Plug-On-Board for collecting operating data/ weather conditions, internet connectivity via LTE, and cloud communication.

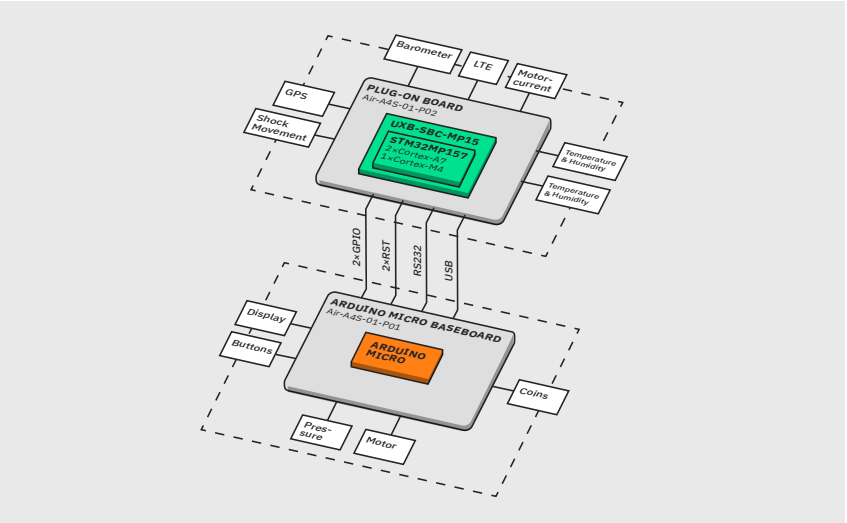



*Figure 1 offers an overview of a use case derived from the user story:*

Data acquisition, pre-processing, and transmission are tasks processed on the device (Device-part). Such data shall be transferred in a cost-optimized way to the cloud. This can be done by an IoT gateway (where devices are connected to) or by the device itself, depending on hardware/software capabilities.

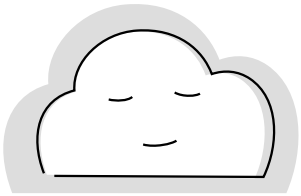
Authorization and data transmission are executed using an interface provided by a hyperscaler cloud service, often called an “IoT-Service”. Examples of these are Microsoft Azure IoT Hub or AWS IoT Core. These services provide a secure way of communication and are characterized by high availability. They also provide a bidirectional way of communication using protocols like e.g. MQTT[1].

### Summary: Chosen base technology for the device-part:

- Linux (OS, network stack, LTE connectivity, SSL, etc.)
- Qt (application framework)
- Azure IoT Hub Device SDK (connectivity to Azure IoT Hub)



1 Message Queuing Telemetry Transport (MQTT)

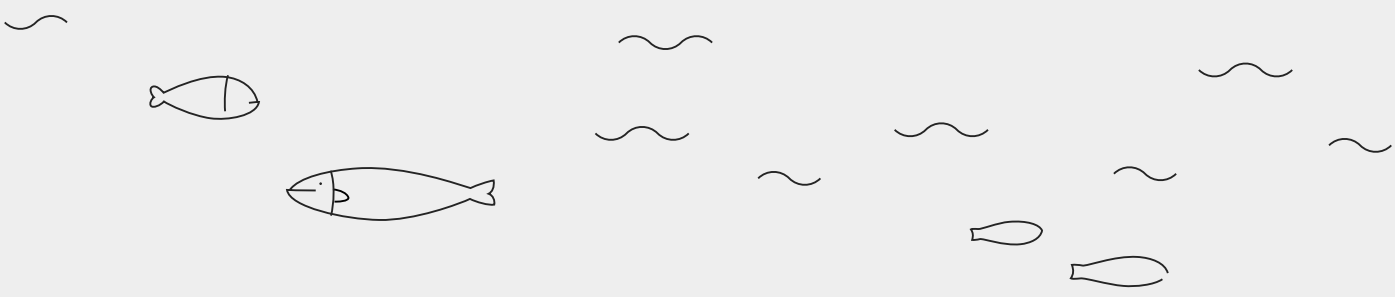


## Cloud Components

On the cloud side, pre- and postprocessing, as well as data provisioning tasks, are mostly use case driven: Data is received, processed (e.g. e-mail notification), stored in a database, post-processed (e.g. average calculations), provided to any other system or user (API) and optionally visualized (e.g. Web UI or local software using the API). Users need to be authenticated and the data transfer between cloud and users shall meet state-of-the-art encryption techniques like HTTPS.

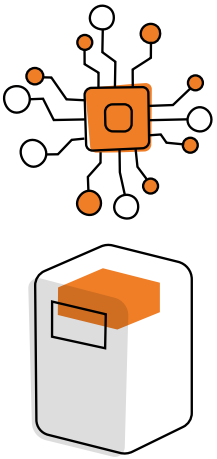
The underlying case study features the following base technologies:

- Azure Cloud Platform (virtual machines)
- Azure Kubernetes Service – AKS (container orchestration)
- Azure IoT Hub (device communication)
- MongoDB as document database (time series data)
- Apollo GraphQL (API)
- Angular/Angular Material (WebUI)
- Node.JS (microservices)
- OpenFaas (microservice orchestration)
- RabbitMQ as message broker (microservice communication)
- Auth0 (user authentication)
- Let’s Encrypt (ssl certificates)



# Hands-on Architecture

## How to Match the Technology Components in a Smoothly Running Architecture?



### Device Architecture

Since Linux is running on the Plug-On Board, it is suitable to use QT as application framework.

> **Qt is a cross-platform application development framework for desktop, embedded and mobile.**
> *Source: https://wiki.qt.io/About_Qt*

Azure IoT Hub Device SDK is used to connect to Azure IoT Hub.

> **IoT Hub Device SDKs enable you to build apps that run on your IoT devices using device client or module client**
> *Source: https://docs.microsoft.com/de-de/azure/iot-hub/iot-hub-devgui-de-sdks*

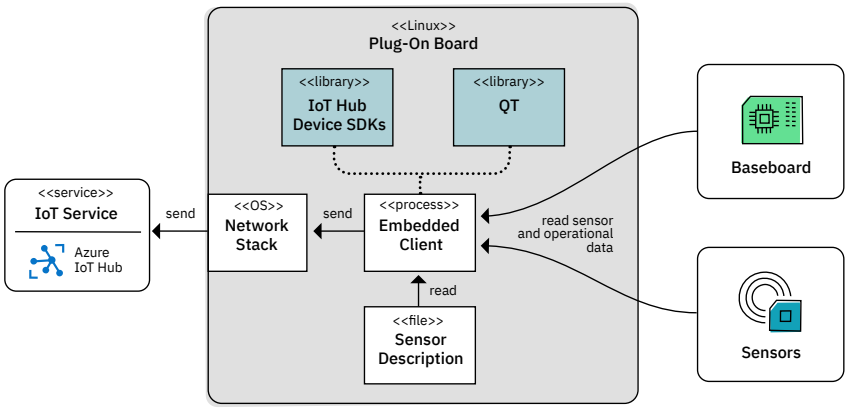
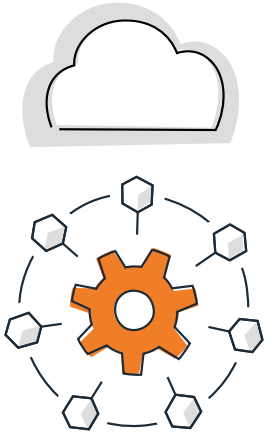



*Figure 4: Architecture Device-Part*

Figure 4 shows the device architecture. Data acquisition, data pre-processing, and data transmission is done by an application named “Embedded Client”. It reads sensor interface descriptions from a sensor description file which includes all information about which sensors are available and how they can be accessed.

This file also includes a description of the operational data sent by the baseboard.

The Azure IoT Hub Device SDK uses the standard network stack provided by Linux. If it is necessary to change the hyperscaler in future, it is possible to replace the Azure IoT Hub Device SDK with the SDK of another hyperscaler.



Microservices

## Cloud Architecture

Data pre-processing and post-processing are omitted here since this is not part of the use case. Data reception, storage, provisioning, and visualization are split up into several components and result in a microservice architecture.

A microservice architecture is a state-of-the-art technique to decompose an application into discrete small services with dedicated tasks. These services can be built and scaled independently.

Within this design pattern, microservices implement a huge part of the IoT-cloud-application‘s business logic and can be seen as kind of distinct single-function modules. Together with other services for persistence and interfacing, they build the actual application.

Well-managed microservice architectures allow agile software development in multiple decentralized teams and extremely accelerate the delivery of new features to deliver and change features quickly.

Using microservices in a cloud environment can enable precise scaling and save the cost of unused server capacity. This supports application scalability.

Applications with microservice architecture consist of individual parts, therefore, the app as a whole is less vulnerable. Data and access can be separated better with microservices than with a monolithic approach where all information goes in and out through the same component. In this regard, microservices make applications more secure.

## Types of Services

The term “service” is very overloaded in the world wide web. Therefore, for a better understanding, we distinguish between several types of services.

| | |
|---|---|
| Service | External services we use from the hyperscaler or any other provider. |
| Subsystem | A service which is deployed “as is” and which is part of the application - e.g. database. |
| Microservice | Distinct single-function module we implement and deploy inside the application landscape. |
| Webservice | Service which is accessible via world wide web and provides some kind of functionality. |

Kubernetes

## Orchestration System

Subsystems, microservices, and web services need an environment where they run in and where orchestration is possible. Kubernetes is used here since Kubernetes is available on many hyperscaler platforms and can be installed in on-premise infrastructures, too.

> **"Kubernetes (commonly stylized as k8s) is an open-source container-orchestration system for automating computer application deployment, scaling, and management.**
>
> **(...)It aims to provide a „platform for automating deployment, scaling, and operations of application containers across clusters of hosts". It works with a range of container tools, including Docker.**
>
> *Source: https://en.wikipedia.org/wiki/Kubernetes*

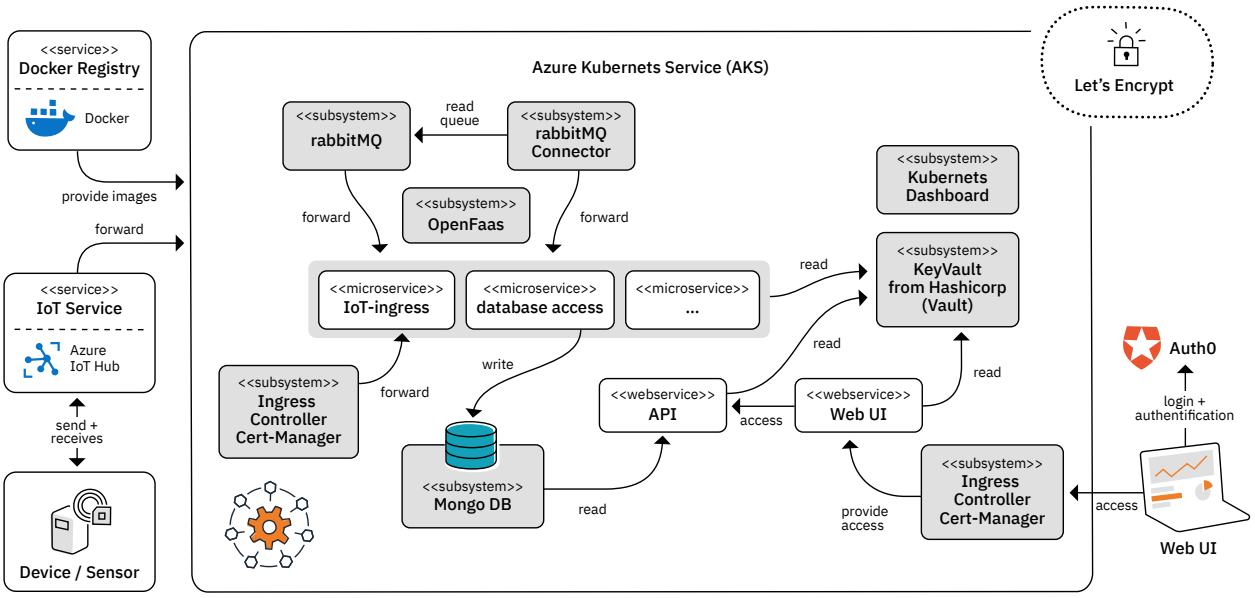



*Figure 3: Architecture Cloud-Part.*

Figure 3 shows the cloud-part's architecture. Some services come from Microsoft Azure and are illustrated on the left side, (Azure Container Registry, Azure IoT Hub). The Kubernetes cluster (Azure Kubernetes Service) in the middle comes from Azure too and hosts microservices, subsystems and webservices.

The two services on the right side are not part of Microsoft Azure: Free certificate authority for SSL certificates is provided by letsencrypt.org and an identity provider for user authentication comes from auth0.com.

# Conclusion

Open-source software components can be used to build an IoT application on hyperscaler infrastructure from scratch. Such an application can be deployed on most popular hyperscalers and major dependencies to vendor-specific services can be avoided. In terms of investment security, such flexibility may be crucial.

Customizability is given by the fact that the individual microservices and web services within the Kubernetes cluster can be written fully individually according to the application‘s requirements. Furthermore, the WebUI can be deployed individually to create a unique user experience.

The chosen microservice architecture assures easy extendibility of application features that fits into the agile development concept of a minimum viable product (MVP).

Scalability is achieved by using a serverless computing framework and hyperscalers autoscaling mechanisms. These autoscaling mechanisms are available on all popular hyperscaler platforms and are to be considered in an on-premise scenario individually.

# About U-Experten

*Heike Ziegler*
*Managing Director Software*

*Robert Stumpe*
*IoT Team Lead*

*Jeremias Bosch*
*Software Engineering &*
*Technical Project Management*

U-Experten are leading specialists for the realization, industrialization, and production of individual embedded HMI panels and smart connected devices. For the consumer and capital goods sectors, our interdisciplinary teams of UX design, engineering, and production are developing, designing, and producing connected products and product lines with perfect user experience.

**For more information, visit our website:**
**U-Experten | Smart Connected Products**

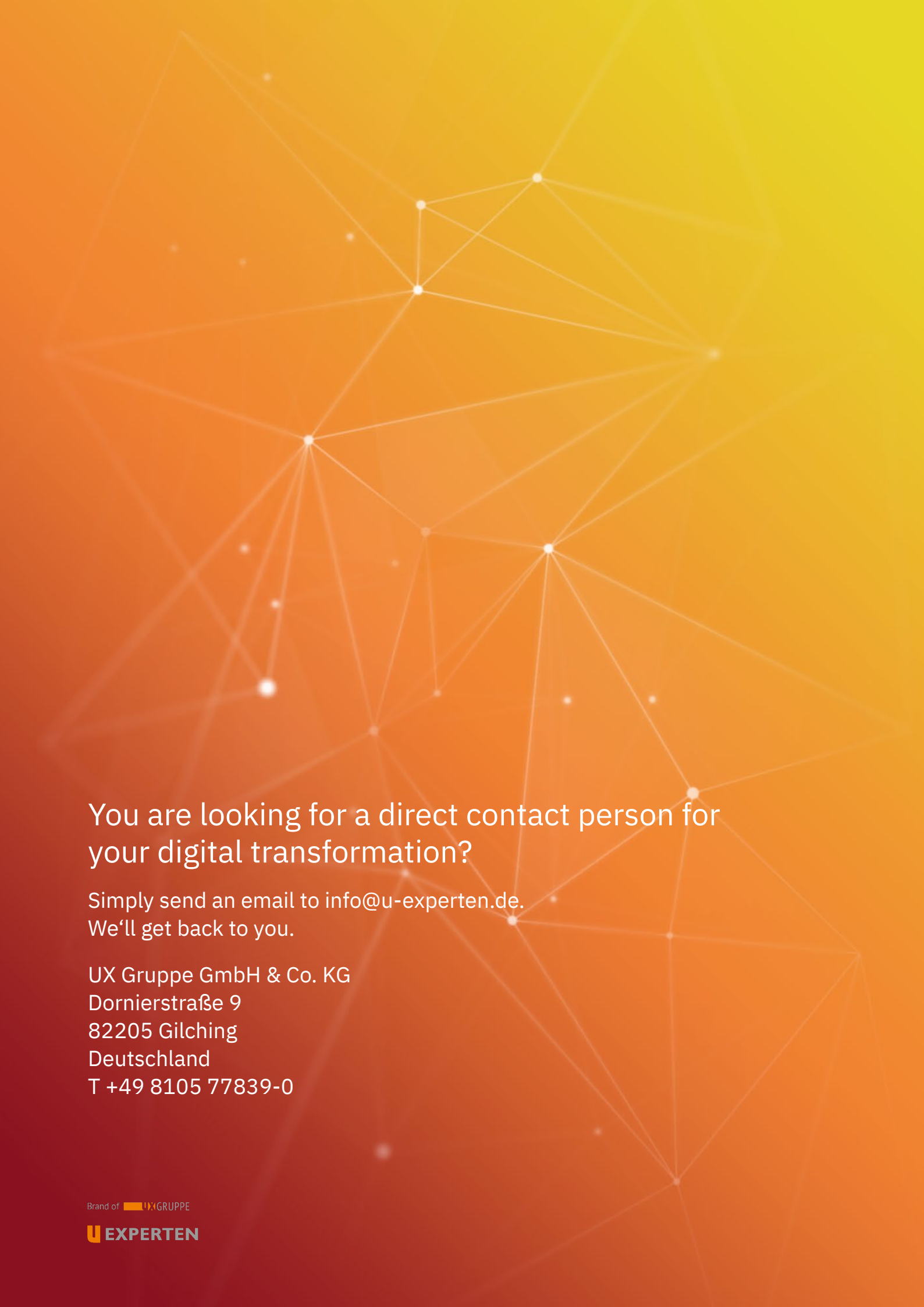
All companies and brands of the UX Gruppe stand for interdisciplinary cooperation in the areas of design, development, production, and distribution of industrial products. Excellent user experience is always our focus. From technological and design ideas to series production, UX Gruppe offers a broad spectrum of innovative services, components, and individual overall solutions.

UX Gruppe unites Ultratronik GmbH, Ultratronik Vertriebs GmbH, Imago Design GmbH, basysKom GmbH and the brand U-Experten under one roof. The group is located in Gilching, near Munich.

# Exhibit

Summary of used Components of the above-mentioned case study ”AIR4SUP”:

| Component | Type | Host System | Used for | Product/Name |
|---|---|---|---|---|
| Kubernetes | service | Azure | deployment, scaling, orchestration | Azure Kubernetes Service (AKS) |
| Docker Registry | service | Azure | storing binary images | Azure Container Registry |
| IoT-Service | service | Azure | device communication | Azure IoT Hub |
| Certificate Authority | service | letsencrypt.com | SSL certificatates | Let’s Encrypt |
| Identity Provider | service | Auth0 | user authentication | Auth0 |
| Serverless Computing Framework | subsystem | Kubernetes | orchestration microservices | OpenFaas |
| Message Broker | subsystem | Kubernetes | communication between microservices | Rabbit MQ, Rabbit MQ connector |
| Database | subsystem | Kubernetes | time series data | MongoDB |
| Certificate Management Controller | subsystem | Kubernetes | managing ssl certificates | Certmanager |
| Ingress Controller | subsystem | Kubernetes | routing traffic | NGINX |
| Secret Store | subsystem | Kubernetes | store secrets and configurations | HashiCorp Vault |
| API | webservice | Kubernetes | public API | "customized (typeScript)“ |
| Web UI | webservice | Kubernetes | portal with user interface | "customized (typescript)“ |
| IoT-Ingress | microservice | Kubernetes | Adaptor IoT Service to microservices | "customized (javaScript)“ |
| Database Access | microservice | Kubernetes | writing time series data to database | "customized (javaScript)“ |



## You are looking for a direct contact person for your digital transformation?

Simply send an email to info@u-experten.de.
We‘ll get back to you.

UX Gruppe GmbH & Co. KG
Dornierstraße 9
82205 Gilching
Deutschland
T +49 8105 77839-0

Brand of UX GRUPPE
U EXPERTEN