



# Docker für Dotnedders

Aber nicht nur.



# Themenüberblick

- Motivation: Container im Windpark und im Webshop
- Virtualisierung: Technologischer Überblick
- Docker: Infrastruktur
- Docker: Einfache Beispiele
- Docker: Befehlsübersicht
- Images bauen mit Docker-Files
- Mit Docker-Compose Container vernetzen
- Ausblick: Orchestrierung komplexer Container-Landschaften Ks8 und KubeEdge



# Motivation

## Was ist Docker?

- ▶ Mit Docker könnte die Firma Docker oder deren Produkt die Docker-Engine gemeint sein. Wir verstehen hier unter Docker die Docker-Engine.
- ▶ Docker ist ein Open-Source Tool, das das automatisierte Deployment von Applikationen in Software Containern realisiert.
- ▶ Neben der Community Variante docker-ce gibt es noch die enterprise Variante docker-ee mit kommerziellem Support und erweitertem Tooling (Bedienoberfläche)

# Motivation

## Was kann ich mit Docker machen?

- Schnelle Erstellung Entwicklungs- u. Testumgebungen mit Datenbanken, Webservern usw.
- Starke Kapselung ohne „Verunreinigung“ des Entwicklungsrechners.
- Mehrere unterschiedliche Versionen mit unterschiedlichen Konfigurationen, Datenbanken u.a. gleichzeitig installieren.
- Deployment-Szenarien schnell erstellen und testen.
- Gleiches Verhalten auf lokalen / dev / staging and production Servern.
- Deployment: Schnelles Update und schnelle Wiederherstellung von Vorgängerversionen.
- Einfaches klares Monitoring.
  
- Komplexe verteilte Applikationen (Microservice)
- Edge Computing (mit Kubernetes)



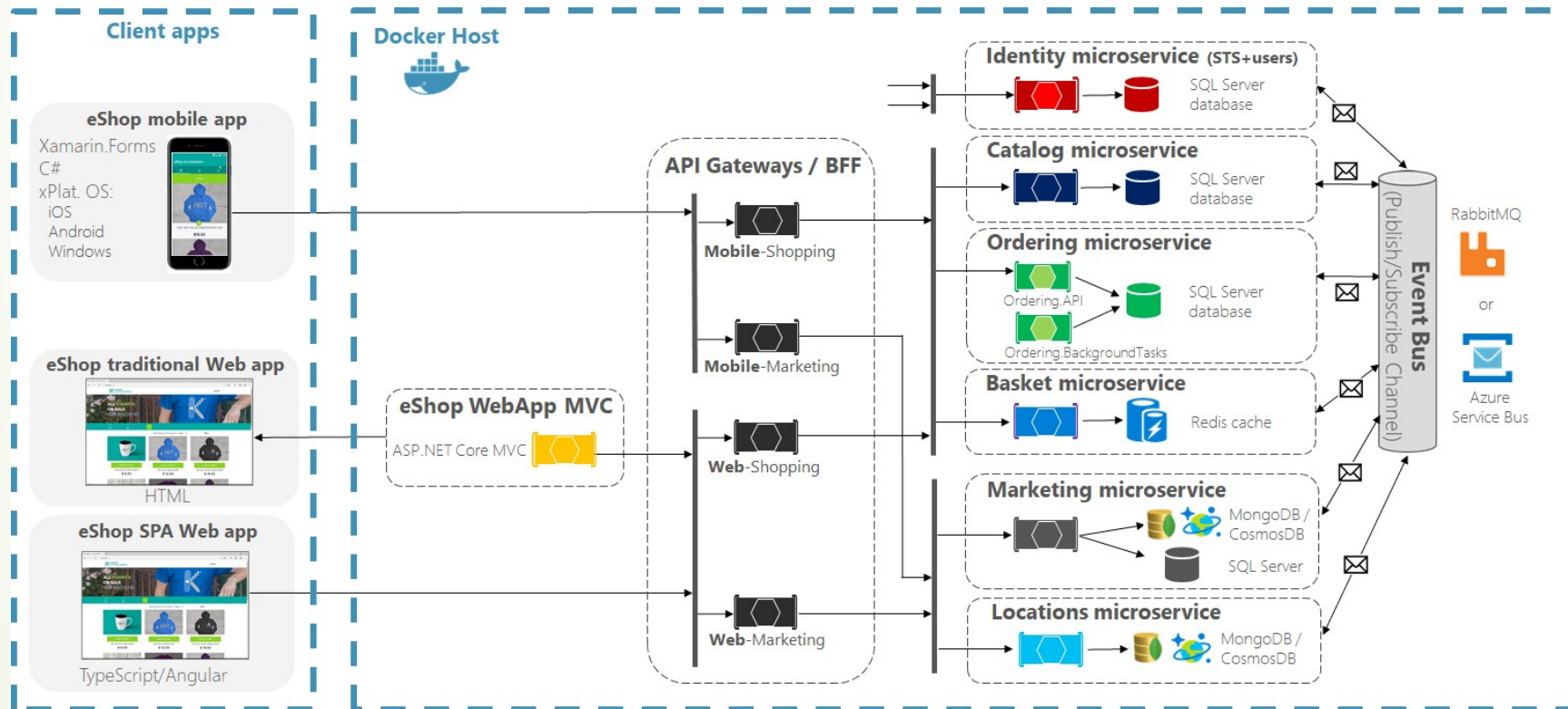
# Motivation Plattform

- Die native Plattform für Docker ist Linux.
- Aber die Implementationen für Mac und Windows stehen dem nicht mehr weit nach.
- Docker auf Windows-Rechner ist ab Windows 10 prof. oder Windows Server 16 möglich.
- Unter Windows kann man zwischen Linux und Windows-Containern wählen.
  
- Anwendungen in Docker-Containern sind reine Server-Applikationen, für die es eine Unterstützung im eingebetteten OS geben gibt.
- Das ist z.B. DotNet Core, Java, Python, C/C++ Rust usw.



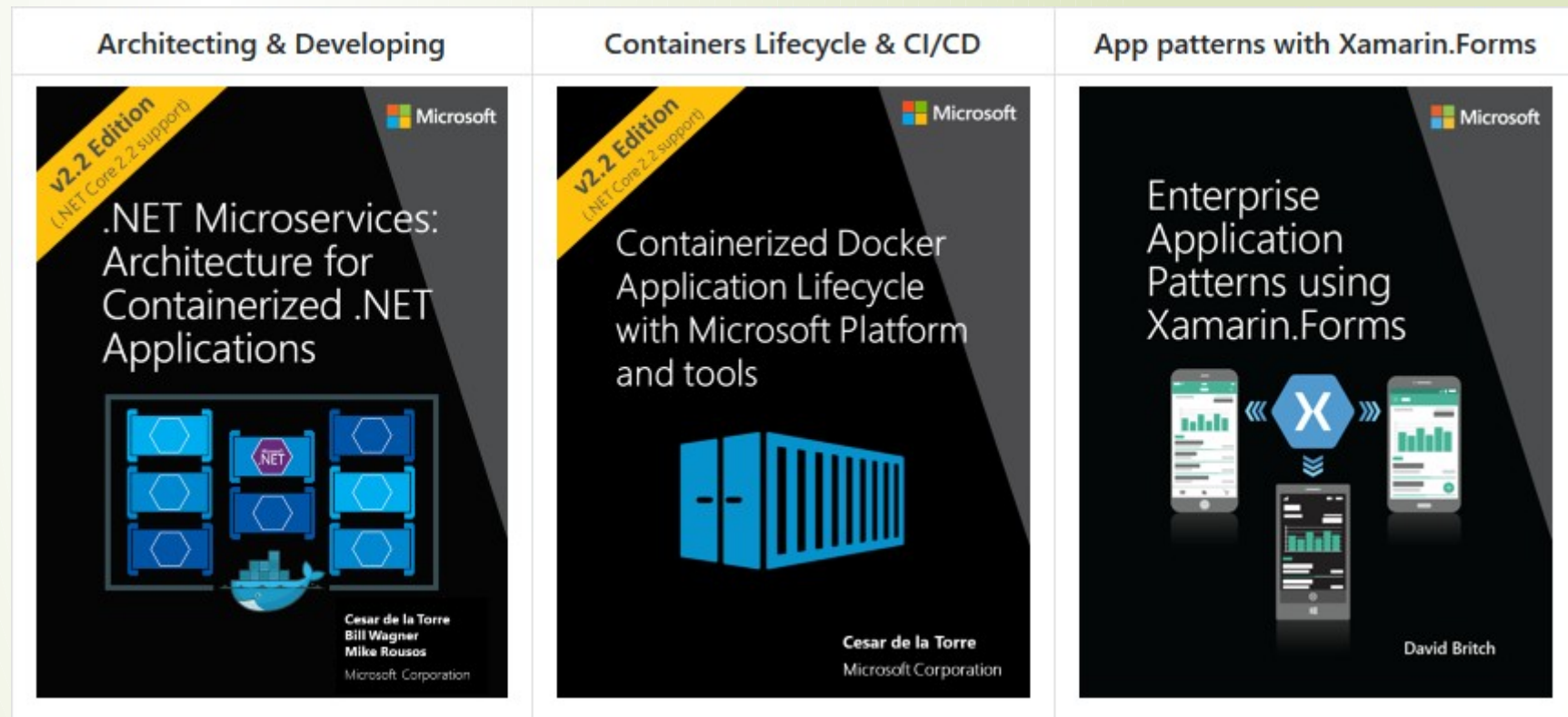
# Motivation eShop Microservices

## eShopOnContainers reference application (Development environment architecture)



# eShop

➔ <https://github.com/dotnet-architecture/eShopOnContainers>



# Windfarm

Windfarm 1



Windfarm 2



Windfarm 3



Internet

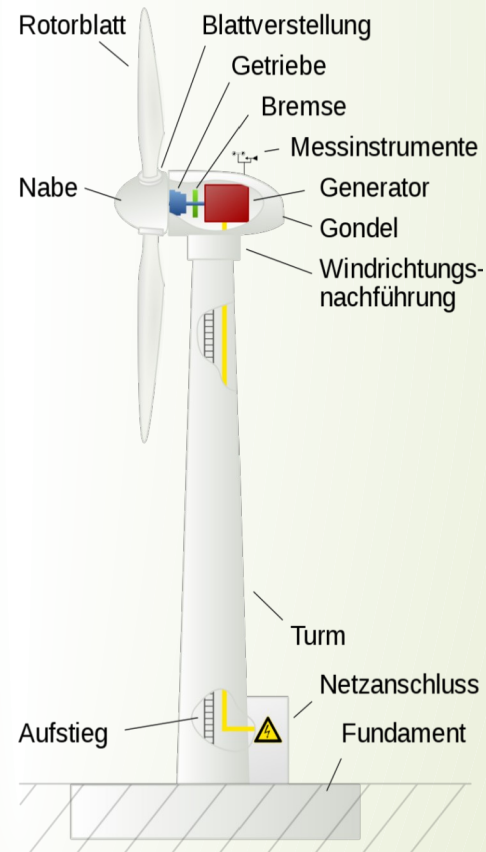


Rechenzentrum oder Cloud

- Monitoring
- Steuerung
- Messdaten
- Parametrierung
- Programm-Updates



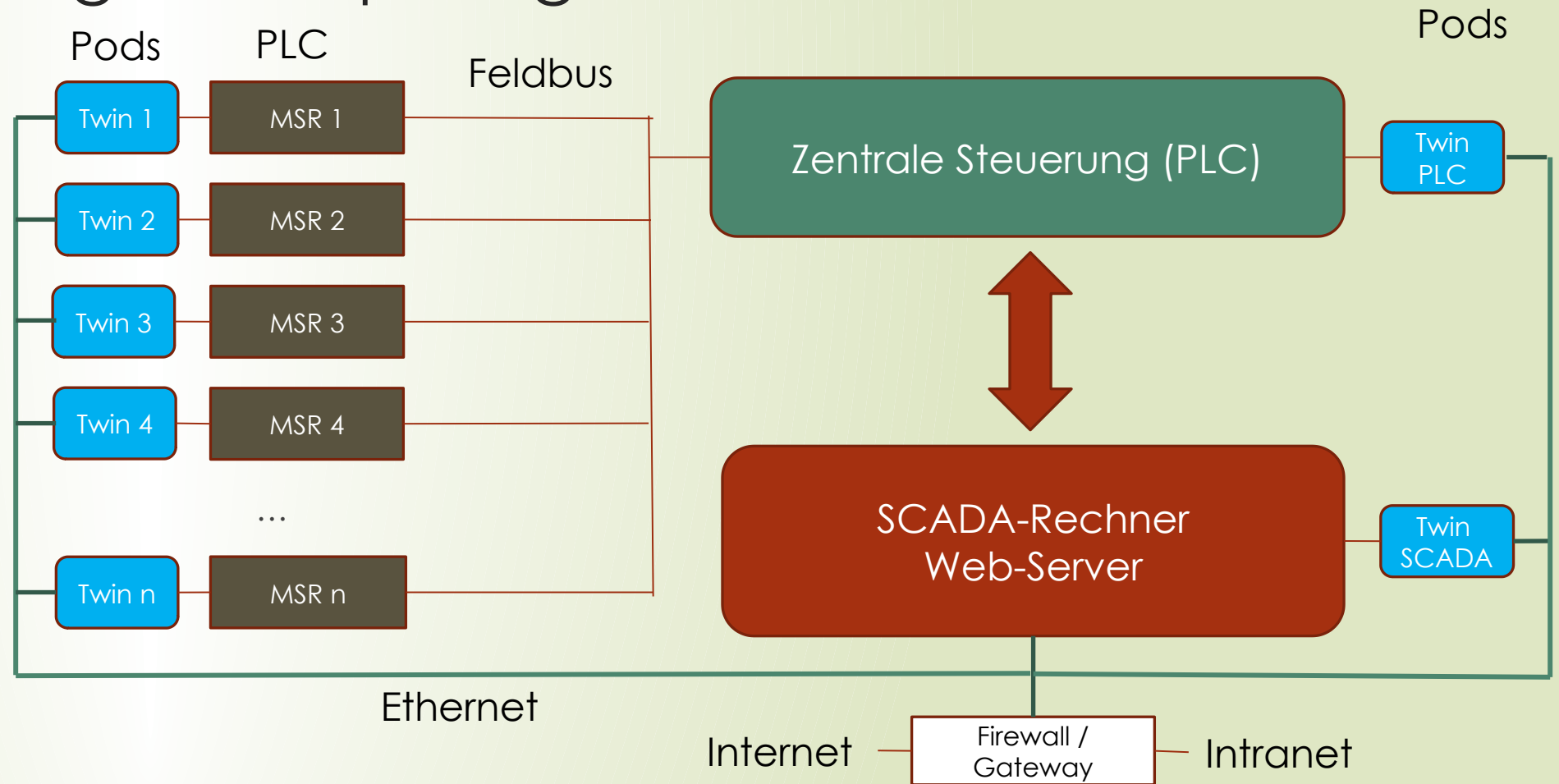
# Windkraftanlage



Messen, Steuern und Regeln der einzelnen Anlagenteile geschieht durch mehrere autonome Steuereinheiten, die mit einer zentralen Steuereinheit kommunizieren.

- Azimut-Steuerung (Windrichtung)
- Pitch-Steuerung (Blattverstellung),
- Umwandlungsregelung (Generator),
- Spannungsqualitätsüberwachung,
- Blitzeinschlag-Erkennung,
- Anti-Icing
- usw.

# Steuerung Windkraftwerk Edge-Computing



# Virtualisierung

- ▶ MVS/TSO z/370 IBM
- ▶ UNIX-Virtualisierte App
- ▶ Hyper-V, VmWare und VirtualBox
- ▶ LXC – Linux Container
- ▶ Docker-Container

OS-Virtualisierung  
Hyper-V, WmWare ...

Vollständiges OS

Host-OS  
ermöglicht Zugriff  
auf Hardware

Hardware

- Aufwändig
- relativ langsam
- kapselt stärker

Container-Virtualisierung  
LXC, Docker-Container

OS-Stub  
nutzt Host-OS

Host-OS  
kapselt Hardware

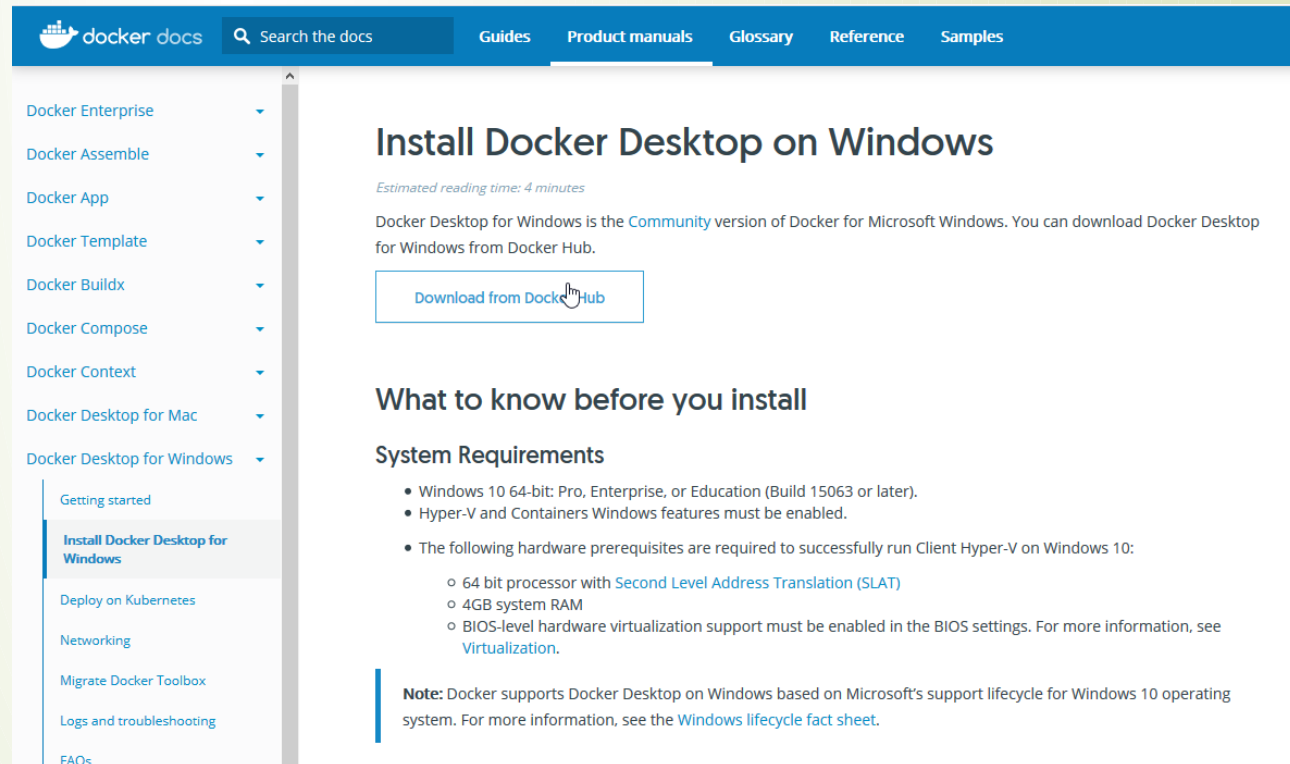
Hardware

- Kleiner Footprint
- Schnell
- Schwächer gekapselt auf OS-Ebene

# Docker Infrastruktur

## Docker Desktop Windows

- <https://docs.docker.com/docker-for-windows/install/>
- Achtung: Docker Desktop aktiviert Hyper-V (VMware und Virtual Box funktionieren nicht mehr.)



The screenshot shows the Docker documentation website. The navigation bar includes 'docker docs', a search bar, and links for 'Guides', 'Product manuals', 'Glossary', 'Reference', and 'Samples'. The left sidebar lists various Docker products, with 'Docker Desktop for Windows' selected. The main content area is titled 'Install Docker Desktop on Windows' and includes a 'Download from Docker Hub' button. Below this, there is a section 'What to know before you install' with 'System Requirements' listed as bullet points. A 'Note' at the bottom states that Docker supports Docker Desktop on Windows based on Microsoft's support lifecycle for Windows 10.

docker docs Search the docs Guides Product manuals Glossary Reference Samples

Docker Enterprise  
Docker Assemble  
Docker App  
Docker Template  
Docker Buildx  
Docker Compose  
Docker Context  
Docker Desktop for Mac  
Docker Desktop for Windows

### Install Docker Desktop on Windows

Estimated reading time: 4 minutes

Docker Desktop for Windows is the [Community](#) version of Docker for Microsoft Windows. You can download Docker Desktop for Windows from Docker Hub.

[Download from Docker Hub](#)

### What to know before you install

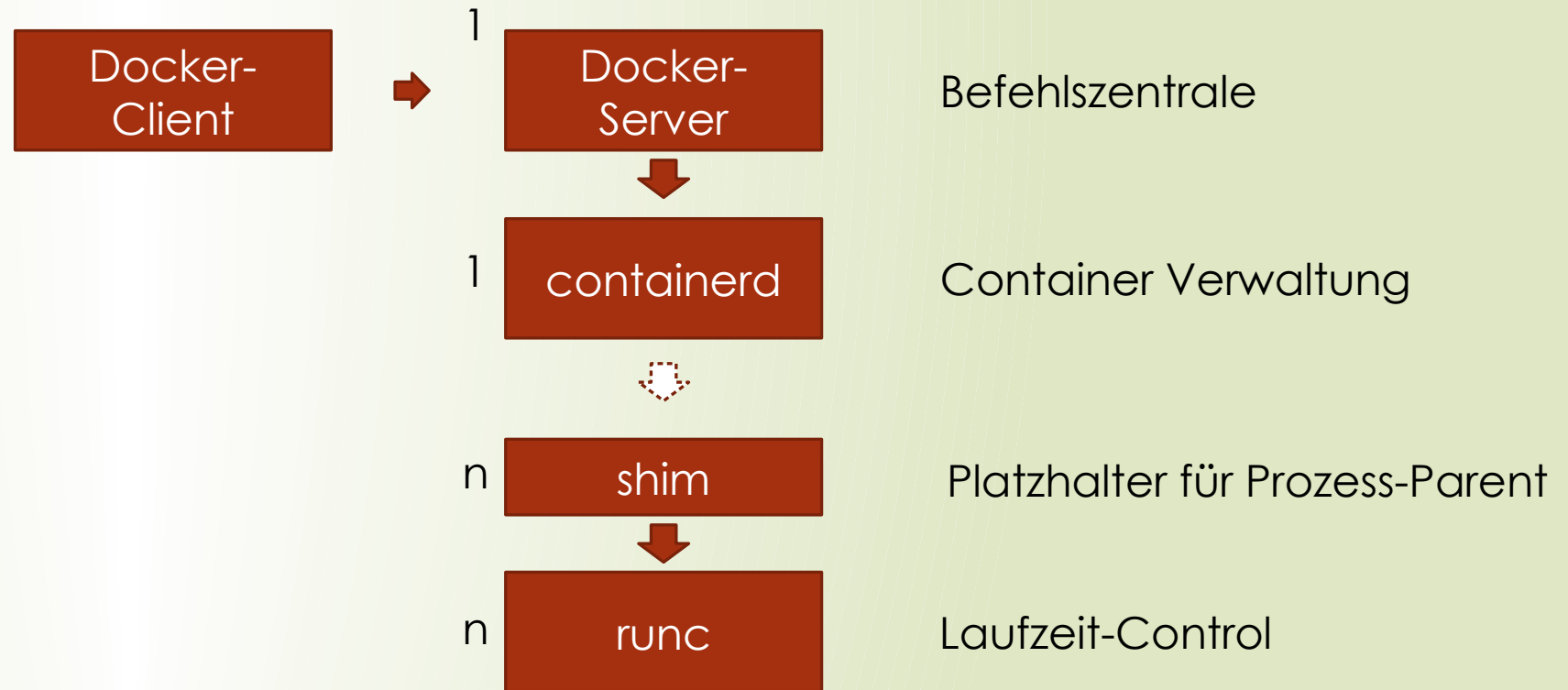
#### System Requirements

- Windows 10 64-bit: Pro, Enterprise, or Education (Build 15063 or later).
- Hyper-V and Containers Windows features must be enabled.
- The following hardware prerequisites are required to successfully run Client Hyper-V on Windows 10:
  - 64 bit processor with [Second Level Address Translation \(SLAT\)](#)
  - 4GB system RAM
  - BIOS-level hardware virtualization support must be enabled in the BIOS settings. For more information, see [Virtualization](#).

**Note:** Docker supports Docker Desktop on Windows based on Microsoft's support lifecycle for Windows 10 operating system. For more information, see the [Windows lifecycle fact sheet](#).

# Docker Infrastruktur

## Docker-Engine







# Docker Infrastruktur Image

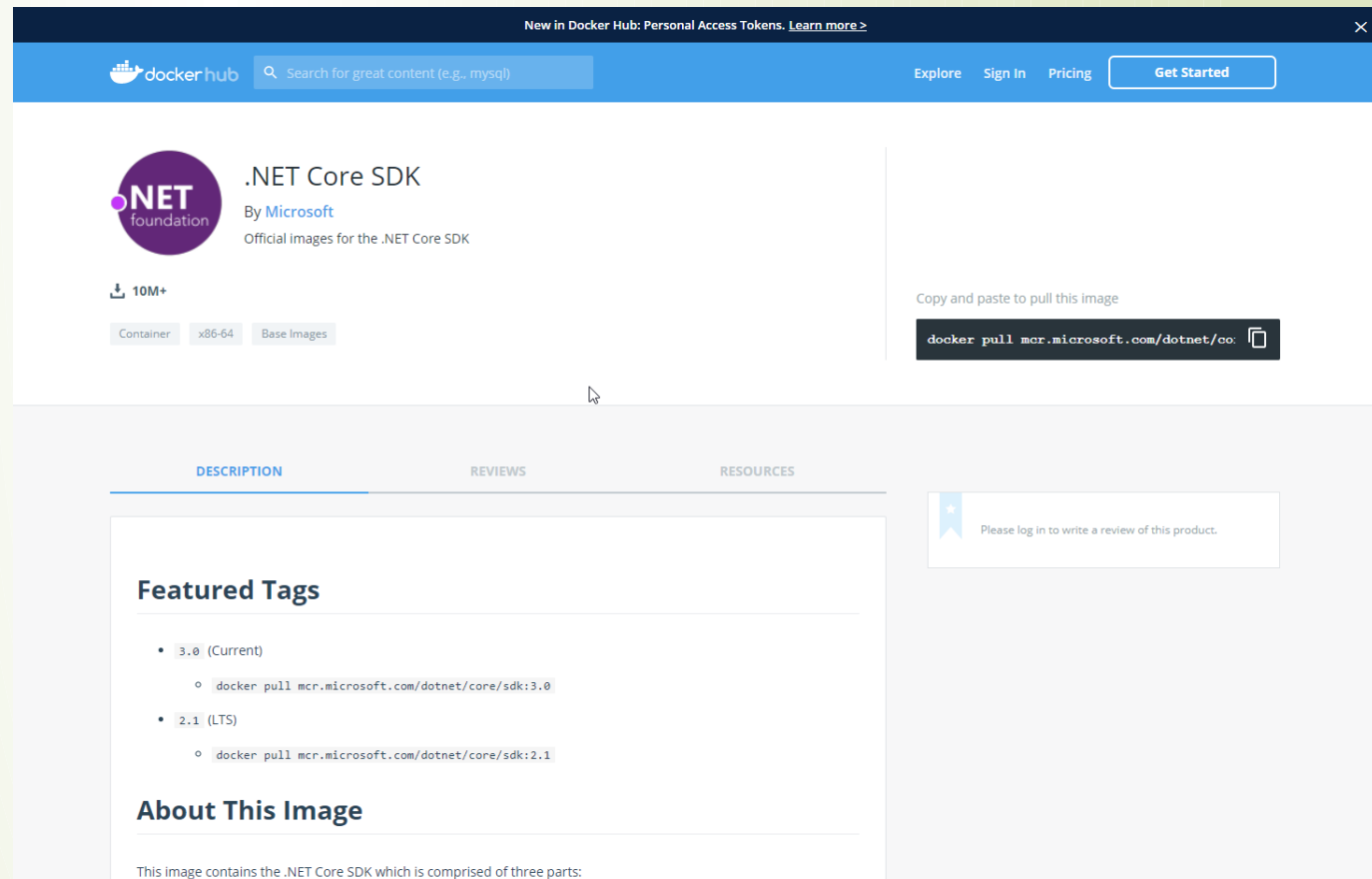
- ▶ Ein Image enthält den Inhalt eines Containers.
- ▶ Es enthält ein OS und optional darauf schichtweise aufbauend Runtimeumgebungen für Programmiersprachen oder Datenbanken o.ä. und Applikationen.
- ▶ Übliche Vorgehensweise für Applikation:
  - ▶ Wahl eines Basis Images
  - ▶ Erweiterung des Basis Images um die eigene Applikation (siehe Docker File)

Das Vorratslager für bereitgestellte Images befindet auf Docker Hub.

Dort können auch eigene Images abgelegt werden. <https://hub.docker.com/>

# Docker Infrastruktur

## Docker-Hub



The screenshot shows the Docker Hub interface for the .NET Core SDK image. At the top, there is a navigation bar with the Docker Hub logo, a search bar, and links for Explore, Sign In, Pricing, and a Get Started button. Below the navigation bar, the main content area displays the image details for ".NET Core SDK" by Microsoft. The image is categorized as a "Container" and "Base Images". It has over 10 million downloads. A code block shows the command to pull the image: `docker pull mcr.microsoft.com/dotnet/core/sdk`. Below this, there are tabs for "DESCRIPTION", "REVIEWS", and "RESOURCES". The "DESCRIPTION" tab is active, showing "Featured Tags" for versions 3.0 (Current) and 2.1 (LTS), each with a corresponding pull command. There is also a section for "About This Image" which states that the image contains the .NET Core SDK, which is comprised of three parts. A login prompt is visible on the right side of the page.

Warenlager (repository) für images

# Docker Infrastruktur

## Docker Hub

- dotnet/core-nightly: .NET Core (Preview)
- .NET Framework:
- dotnet/framework: .NET Framework, ASP.NET and WCF
  - dotnet/framework/samples: .NET Framework, ASP.NET and WCF Samples

### Full Tag Listing

Target OS

#### Linux amd64 Tags

Tags	Dockerfile	OS Version	Last Modified
3.0.100-buster, 3.0-buster, 3.0.100, 3.0, latest	Dockerfile	Debian 10	09/23/2019
3.0.100-alpine3.9, 3.0-alpine3.9, 3.0.100-alpine, 3.0-alpine	Dockerfile	Alpine 3.9	09/23/2019
3.0.100-disco, 3.0-disco	Dockerfile	Ubuntu 19.04	09/23/2019
3.0.100-bionic, 3.0-bionic	Dockerfile	Ubuntu 18.04	09/23/2019
2.2.402-stretch, 2.2-stretch, 2.2.402, 2.2	Dockerfile	Debian 9	09/12/2019
2.2.402-alpine3.9, 2.2-alpine3.9, 2.2.402-alpine, 2.2-alpine	Dockerfile	Alpine 3.9	09/10/2019
2.2.402-alpine3.8, 2.2-alpine3.8	Dockerfile	Alpine 3.8	09/10/2019
2.2.402-bionic, 2.2-bionic	Dockerfile	Ubuntu 18.04	09/19/2019
2.1.802-stretch, 2.1-stretch, 2.1.802, 2.1	Dockerfile	Debian 9	09/12/2019
2.1.802-alpine3.9, 2.1-alpine3.9, 2.1.802-alpine, 2.1-alpine	Dockerfile	Alpine 3.9	09/10/2019
2.1.802-alpine3.7, 2.1-alpine3.7	Dockerfile	Alpine 3.7	09/10/2019
2.1.802-bionic, 2.1-bionic	Dockerfile	Ubuntu 18.04	09/19/2019


Container OS

#### Linux arm64 Tags

Tags	Dockerfile	OS Version	Last Modified
3.0.100-buster-arm64v8, 3.0-buster-arm64v8, 3.0.100, 3.0, latest	Dockerfile	Debian 10	09/23/2019
3.0.100-disco-arm64v8, 3.0-disco-arm64v8	Dockerfile	Ubuntu 19.04	09/23/2019

Befehl:  
docker image pull <image name : tag>

Hinweis:  
Nicht für jedes Target OS muss ein Image existieren.



# Docker Container Infrastruktur

## Container Ressourcen

- Images
- Ports (Mapping)
- Volumes (Speicher Mapping; Links)
- Networks (Kommunikation zwischen mehreren Containern)

# Befehlsübersicht

## Docker Cheat Sheet

### ORCHESTRATE

Initialize swarm mode and listen on a specific interface  
`docker swarm init --advertise-addr 10.1.0.2`

Join an existing swarm as a manager node  
`docker swarm join --token <manager-token> 10.1.0.2:2377`

Join an existing swarm as a worker node  
`docker swarm join --token <worker-token> 10.1.0.2:2377`

List the nodes participating in a swarm  
`docker node ls`

Create a service from an image exposed on a specific port and deploy 3 instances  
`docker service create --replicas 3 -p 80:80 --name web nginx`

List the services running in a swarm  
`docker service ls`

Scale a service  
`docker service scale web=5`

List the tasks of a service  
`docker service ps web`

### BUILD

Build an image from the Dockerfile in the current directory and tag the image  
`docker build -t myapp:1.0 .`

List all images that are locally stored with the Docker engine  
`docker images`

Delete an image from the local image store  
`docker rmi alpine:3.4`

### SHIP

Pull an image from a registry  
`docker pull alpine:3.4`

Retag a local image with a new image name and tag  
`docker tag alpine:3.4 myrepo/myalpine:3.4`

Log in to a registry (the Docker Hub by default)  
`docker login my.registry.com:8000`

Push an image to a registry  
`docker push myrepo/myalpine:3.4`



### RUN

`docker run`  
--rm remove container automatically after it exits  
-it connect the container to terminal  
--name web name the container  
-p 5000:80 expose port 5000 externally and map to port 80  
-v ~/dev:/code create a host mapped volume inside the container  
alpine:3.4 the image from which the container is instantiated  
/bin/sh the command to run inside the container

Stop a running container through SIGTERM  
`docker stop web`

Stop a running container through SIGKILL  
`docker kill web`

Create an overlay network and specify a subnet  
`docker network create --subnet 10.1.0.0/24 --gateway 10.1.0.1 -d overlay mynet`

List the networks  
`docker network ls`

List the running containers  
`docker ps`

Delete all running and stopped containers  
`docker rm -f $(docker ps -aq)`

Create a new bash process inside the container and connect it to the terminal  
`docker exec -it web bash`

Print the last 100 lines of a container's logs  
`docker logs --tail 100 web`





# Docker File

- ▶ Deklarative Anweisungen in einem YAML-File

Einige Anweisungen:

- ▶ FROM – legt das Basis-Image fest
- ▶ RUN – führt Kommandos im Container aus
- ▶ ENV – Setzt Umgebungsvariablen
- ▶ WORKDIR – Setzt das Arbeitsverzeichnis
- ▶ Volume – Moint-Points für persistenten Speicher
- ▶ CMD – Definiert Startkommando
- ▶ ENTRYPOINT - Legt den Startpunkt fest
  
- ▶ [https://docs.docker.com/develop/develop-images/dockerfile\\_best-practices/](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/)



# Docker File

## Einfaches Beispiel

```
FROM ubuntu:16.04
```

```
RUN apt-get update \  
    && apt-get install -y apache2
```

```
COPY index.html /var/www/html/
```

```
WORKDIR /var/www/html
```

```
CMD ["apachectl", "-D", "FOREGROUND"]
```

```
EXPOSE 80
```



# Docker Compose

- Deklarative Anweisungen in einem YAML-File
- Beschreibt eine Folge von Containern, die auf einem Node zusammen arbeiten sollen.

Version # docker-compose version

Services # containers as services

Build # build section

Image # reference to an image

Command

Ports # Port Mapping

Network # references or declares an volumes mapping

Volumes # references an network

Networks # declares networks

Volumes # declares volumes

Secrets

Configs

- <https://docs.docker.com/compose/>



# Docker Compose Beispiel

version: '3'

services:

db:

image: mysql

container\_name: mysql\_db

restart: always

environment:

- MYSQL\_ROOT\_PASSWORD="secret"

web:

image: apache

build: ./webapp

depends\_on:

- db

container\_name: apache\_web

restart: always

ports:

- "8088:80"



# Docker Compose Links

```
version: "3"  
services:  
  web:  
    build: .  
    links:  
      - "db:database"  
  db:  
    image: postgres
```





# Docker-Compose CLI

- ▶ **version** `docker-compose -v # show version`
- ▶ **list** `docker-compose ps`
- ▶ **build** `docker-compose build`
- ▶ **Run** `docker-compose up`
- ▶ **run as a daemon** `docker-compose up -d`
- ▶ **stop** `docker-compose stop`
- ▶ **remove** `docker-compose rm`
- ▶ **stop + remove** `docker-compose down`
- ▶ **restart** `docker-compose restart`

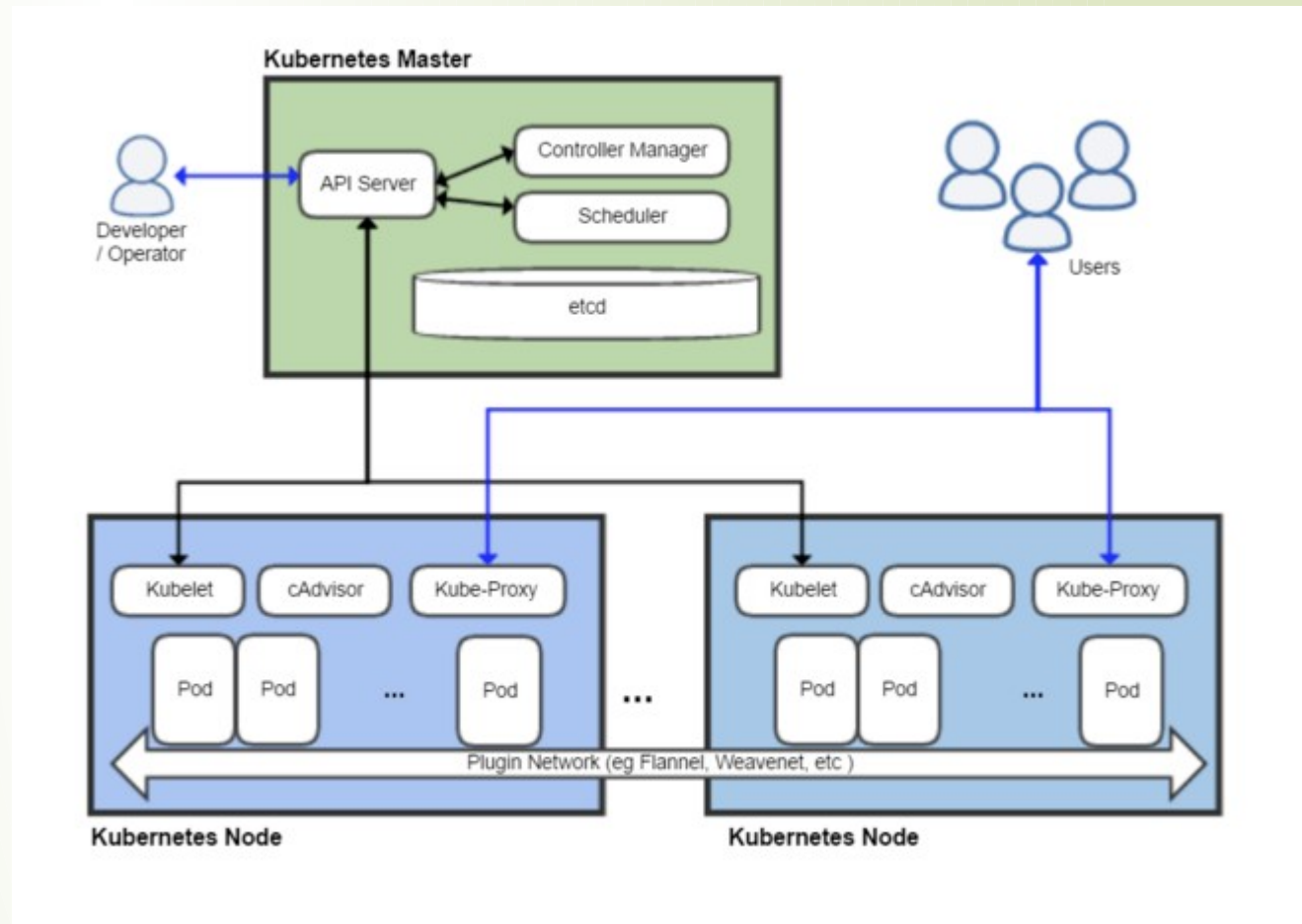


# Beispiele

## Jenkins Pipeline mit Docker

- ▶ Jenkins Pipeline
- ▶ Connect Container mit VS Code
- ▶ eShop mit Docker-Compose

# Ausblick Kubernetes





# Ausblick

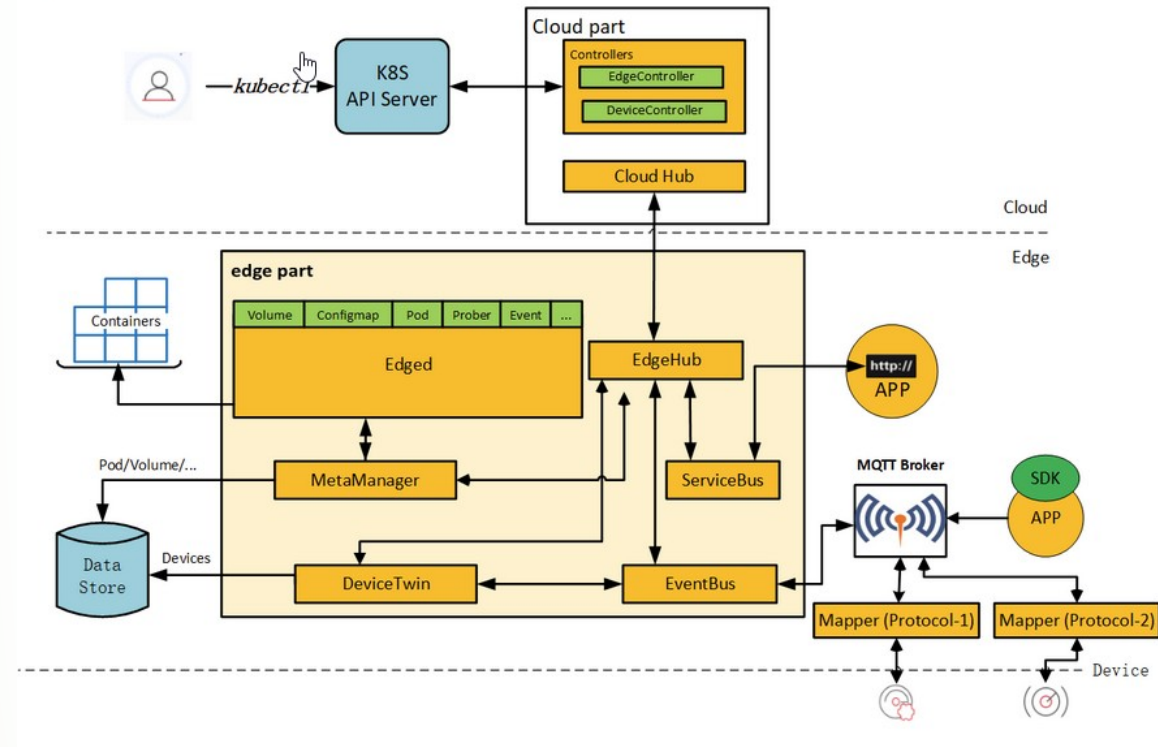
## Private Cloud mit Kubernetes

- Mehrere Nodes (reale oder virtuelle Maschinen)
- Ein Manager Node und mehrere Worker Nodes
- Installation von Docker und Kubernetes auf jeden Node
  
- z.B. für Ubuntu
- Docker: `sudo apt install docker-ce docker-cli containerd.io`
- Ks8: `sudo apt install kubelet kubeadm kubectl`

# Ausblick KubeEdge



## Architecture



<https://github.com/kubeedge/kubeedge>



Vielen Dank!