



# MDM Container & Cloud

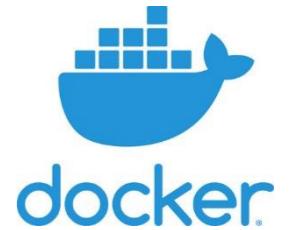


Building Competence. Crossing Borders.

# Container und Virtualisierung

# Docker

Docker ist eine Open-Source-Software zur Isolierung von Anwendungen mithilfe von Containervirtualisierung.



**Vereinfachte Bereitstellung:** Container enthalten alle notwendigen Pakete und lassen sich leicht transportieren und installieren.

**Konsistenz:** Anwendungen funktionieren in jeder Umgebung gleich.

**Ressourceneffizienz:** Container teilen sich das Betriebssystem des Hosts.

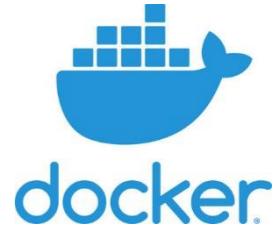
**Schnelle Skalierbarkeit:** Zusätzliche Container können schnell instanziert werden.

**Isolation:** Container bieten eine isolierte und sichere Umgebung.

# Docker Konzepte

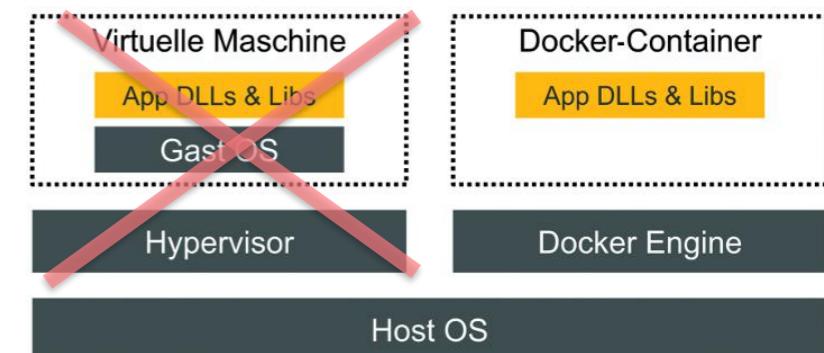
## Docker Konzepte

- Umgebung für Prozesse: Docker bietet eine isolierte Umgebung, um Prozesse zu starten.
- Trennung von Umgebungen: Saubere Trennung der Anwendungen und ihrer Abhängigkeiten.
- Sandbox: Container fungieren als Sandbox mit Zugriff auf definierte Ressourcen.
- Einfaches Interface: Einfache Schnittstelle zum Starten und Verwalten von Anwendungen.



## Was Docker NICHT ist

- Keine vollständige Virtualisierung: Docker virtualisiert nicht den gesamten Prozessor (z.B. x86, ARM) – Emulation mit QEMU oder Buildx möglich.
- Kein eigener Kernel: Container teilen sich den Kernel des Host-Betriebssystems.
- Kein Hypervisor: Docker verwendet keinen Hypervisor (Virtual-Machine-Monitor) und läuft nicht als Gast-OS auf einem fremden OS.

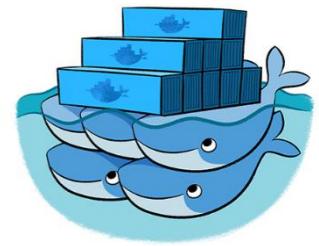


# Kubernetes und Docker Swarm

Docker ist eine Plattform zur Containerisierung von Anwendungen.

Kubernetes (k8s) und Docker Swarm:

Diese sind Orchestrierungstools, die Container-Cluster verwalten und skalieren.



## Kubernetes (k8s)

- Kubernetes, auch k8s oder Kube genannt, ist eine Open-Source-Plattform zur Automatisierung von Linux-Container-Operationen.
- Gruppierter und verwaltet Container, um Anwendungen effizient zu betreiben und zu skalieren.
- Vorteile: Automatisierte Bereitstellung, Skalierung, Selbstheilung und Updates.

## Docker Swarm

- Alternative: Docker Swarm ist eine einfachere Alternative zu Kubernetes.
- Nahtlose Integration mit Docker, ideal für kleinere Cluster und einfache Anwendungsfälle.

# Begriffe: Image, Container, Volume

## Docker Image

- Ein unveränderbares Template für Container
- Kann von einer Registry geladen oder dort gespeichert werden (z.B. Docker Hub)
- Image-Namen haben die Form [registry/] [user/] name[:tag]. Der Standard-Tag ist latest.
- Architektur-Support beachten: amd64 (PC, Intel Mac), arm64 (M1 Mac)

## Docker Container

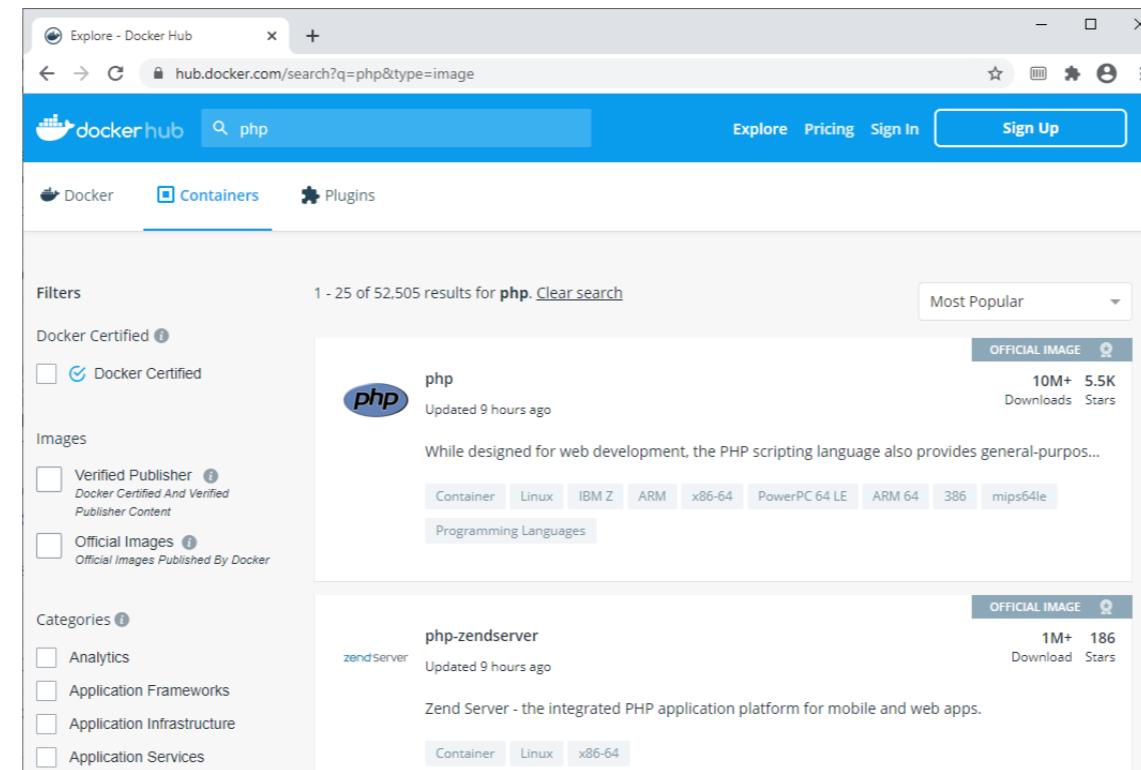
- Eine Instanz eines Image, kann gestartet, gestoppt, neu gestartet usw. werden
- Verwaltet Änderungen auf dem File System (idealerweise Volume verwenden)
- Neues Image kann vom aktuellen Container-Stand gemacht werden (nicht empfohlen, besser Dockerfile)

## Docker Volume

- Ein Docker Volume ist ein externer Speicherbereich, der Daten unabhängig vom Container speichert.
- So bleiben Daten erhalten, auch wenn der Container gelöscht wird.

# Docker Hub

Docker Hub ist ein Verzeichnis von Docker Images. Mittlerweile gibt es für praktisch jede Software ein Docker Image.



<https://hub.docker.com/>

# Ausgewählte Docker Befehle (Terminal)

## Images verwalten

`docker pull [image]`

Lädt ein Image aus einer Registry.

`docker images`

Listet alle lokalen Images auf.

`docker rmi [image]`

Entfernt ein lokales Image.

`docker push [image]`

Lädt ein lokales Image in eine Registry hoch.

## Container verwalten

`docker run [image]`

Startet einen neuen Container aus einem Image.

`docker ps`

Listet alle laufenden Container auf.

`docker ps -a`

Listet alle Container auf, einschließlich gestoppter.

`docker stop [container]`

Stoppt einen laufenden Container.

`docker start [container]`

Startet einen gestoppten Container.

`docker restart [container]`

Startet einen Container neu.

`docker rm [container]`

Entfernt einen gestoppten Container.

# Docker Dashboard

Anstelle des Terminals kann auch das Dashboard verwendet werden.

## Empfehlung

- Start mit Terminal
- Stopp / Restart / Delete mit Dashboard
- URL öffnen mit Dashboard

The screenshot shows the Docker Desktop interface. The left sidebar has options: Containers, Images (which is selected), Volumes, Builds, Docker Scout, and Extensions. The main area is titled 'Images' with a search bar containing 'jenkins'. It shows 18 images in the 'Local' tab, with a progress bar indicating 8.33 GB / 14.83 GB in use. A table lists three Jenkins-related images:

Name	Tag	Image ID	Created	Size	Actions
mosazhaw/jenkins	2.479.	b9a5e18a979a	2 days ago	1.51 GB	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
jenkins/agent	<none>	bc75a6236720	7 months ago	295.83 MB	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
mosazhaw/jenkins	2.426.	d26efd8a4340	11 months ago	1.14 GB	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

At the bottom, it says 'Showing 3 items'. The footer includes 'Resource Saver mode', system stats (RAM 2.39 GB, CPU 0.75%, Disk 216.63 GB avail. of 269.43 GB), a 'New version available' notification, and a bell icon with '2' notifications.

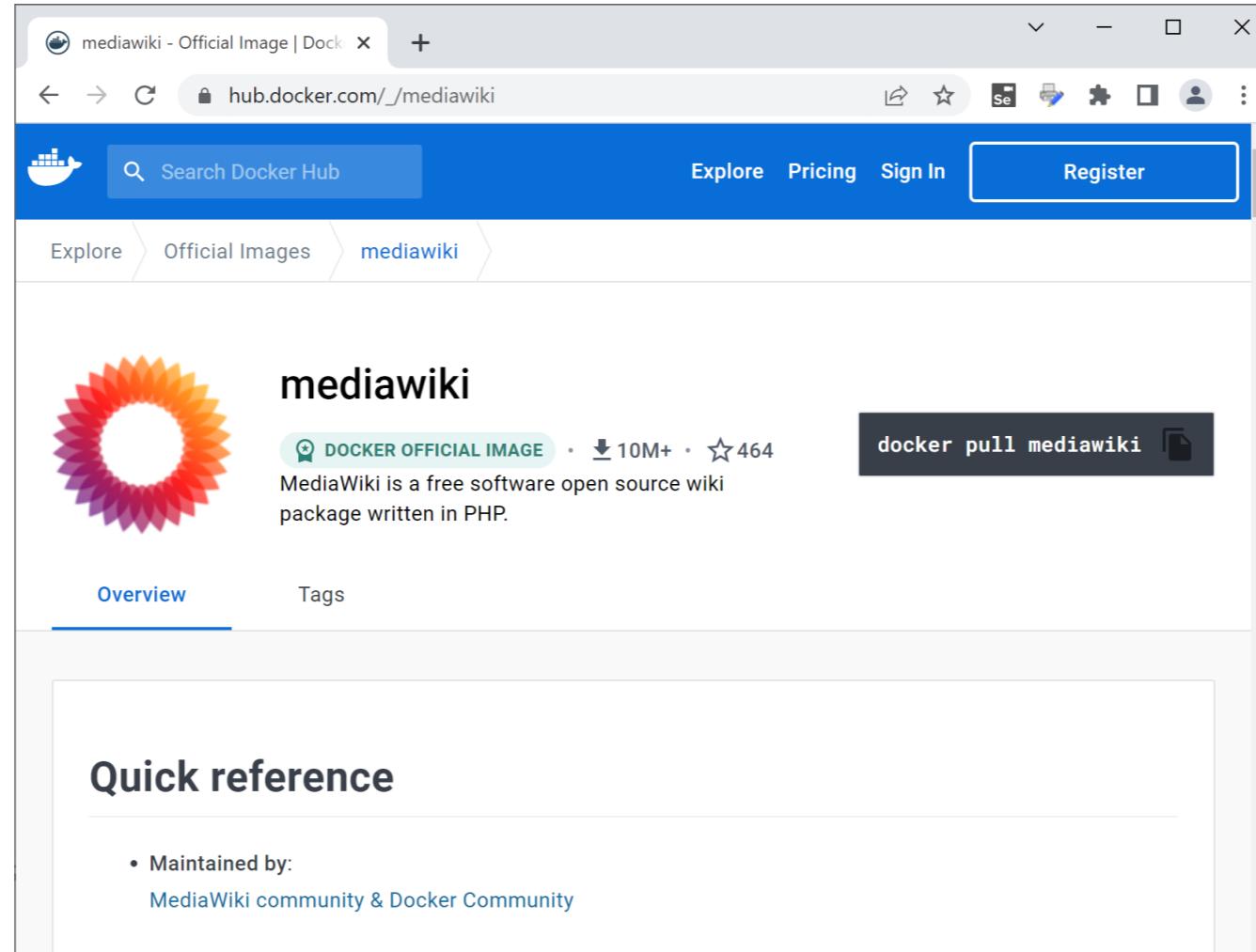
# Docker «Hello World»

Mit mediawiki

# Passendes Image suchen

Tutorial

[https://hub.docker.com/\\_/mediawiki](https://hub.docker.com/_/mediawiki): Docker Pull Command für das gewünschte Image kopieren



# docker pull und run

Tutorial

## pull

```
docker pull mediawiki:1.42.3
```

1. Pull ausführen (mit Tag)

-d detached (nicht im Vordergrund)

Name für Volume

Image:Tag

## run

```
docker run --name some-mediawiki -p 8081:80 -v mediawiki:/var/www/html -d mediawiki:1.42.3
```

2. Starten

Port des Docker Host

Interner Port innerhalb Container

Für **internen Port**, **Volume-Pfad** und **Image-Name:Tag** die Dokumentation des Images auf Docker Hub beachten.

**Container-Name**, **Host-Port** und **Volume-Name** dürfen frei gewählt werden (sofern nicht bereits belegt).

Existiert ein **Volume-Name** bereits, so wird dieses Volume verwendet.

# Dashboard

Tutorial

Nun erscheint der Container im Dashboard:

The screenshot shows the Docker Desktop interface. On the left, a sidebar menu includes 'Containers' (selected), 'Images', 'Volumes', 'Builds', 'Docker Scout', and 'Extensions'. The main area is titled 'Containers' with a 'Give feedback' link. It displays resource usage: 'Container CPU usage' (0.01% / 800%) and 'Container memory usage' (14.14MB / 15.06GB). A yellow callout box points to the 'Port(s)' entry for the 'some-mediawiki' container, which is listed with 'Container ID' d8f8b9ee8792 and 'Image' mediawiki:1.42.3. Below the table is a search bar and a 'Only show running containers' toggle. At the bottom, system status is shown: 'Engine running', 'RAM 4.33 GB CPU 1.51%', and 'Disk 216.52 GB avail. of 269.43 GB'. To the right, a browser window shows the MediaWiki 1.42.3 setup page at localhost:8081/mw-config/index.php, featuring the MediaWiki logo and the text 'MediaWiki 1.42.3 LocalSettings.php not found. Please [set up the wiki](#) first.' A yellow callout box points to this browser window with the text 'Mediawiki in Browser öffnen'. Further right, a panel titled 'Sprache' (Language) lists steps for installation: Sprache, Vorhandenes Wiki, Willkommen bei MediaWiki!, Mit der Datenbank verbinden, Eine vorhandene MediaWiki-Installation aktualisieren, Einstellungen zur Datenbank, Name, Optionen, Installieren, and Fertig!. The 'zh aw School of Management and Law' logo is in the bottom right corner.

Mediawiki in Browser öffnen

Containers

Container CPU usage  
0.01% / 800% (8 CPUs available)

Container memory usage  
14.14MB / 15.06GB

Show charts

Search

Only show running containers

Name	Container ID	Image	Port(s)
some-mediawiki	d8f8b9ee8792	mediawiki:1.42.3	8081:80

Engine running

RAM 4.33 GB CPU 1.51% Disk 216.52 GB avail. of 269.43 GB

Mediawiki 1.42.3

LocalSettings.php not found.  
Please [set up the wiki](#) first.

Sprache während des Installierens:

- Hilfe

- Sprache
- Vorhandenes Wiki
- Willkommen bei MediaWiki!
- Mit der Datenbank verbinden
- Eine vorhandene MediaWiki-Installation aktualisieren
- Einstellungen zur Datenbank
- Name
- Optionen
- Installieren
- Fertig!

## Docker Tutorial «mehrere Container»

Mediawiki nutzt integrierte Datenbank

Typischerweise aber mehrere «Server», z.B. Applikation und Server

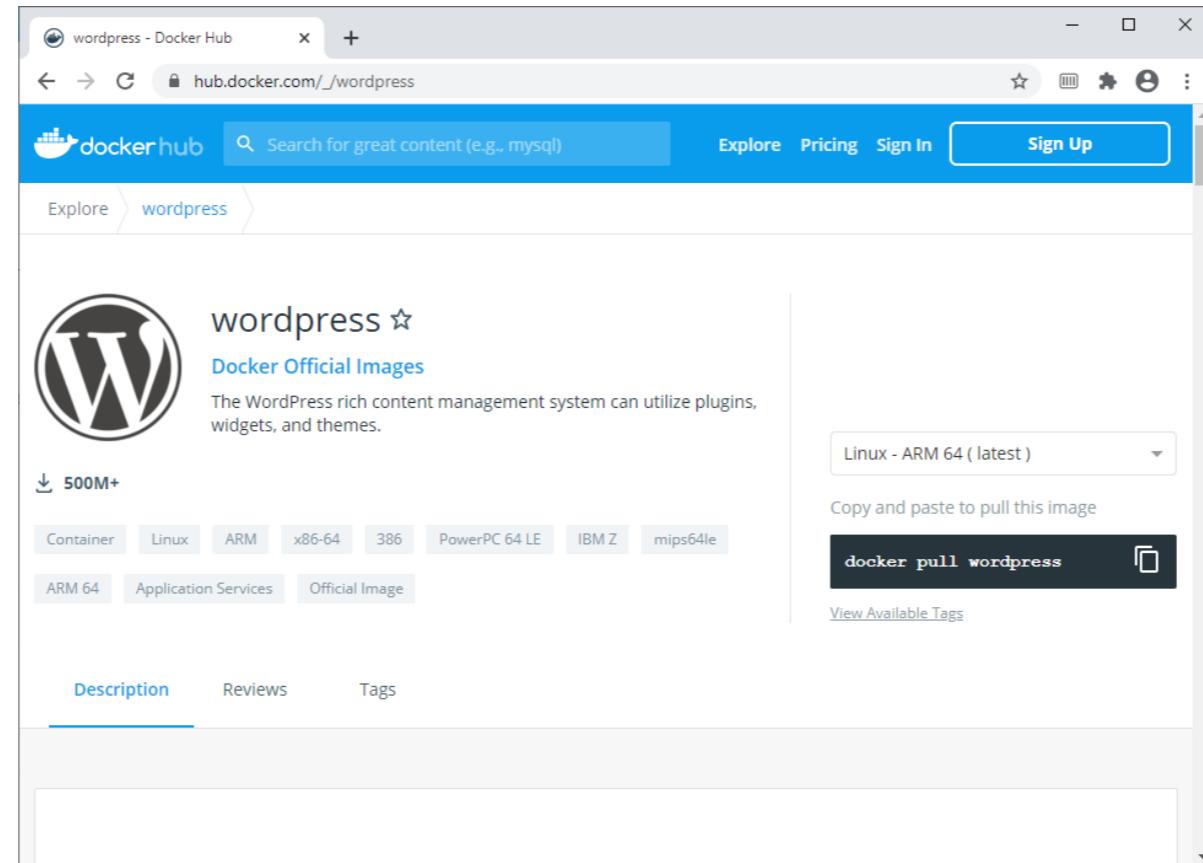
Mehrere Docker Instances

Netzwerke

# Wordpress Image

Tutorial

Wordpress (**wordpress:6.7.0-php8.3**) mit Datenbank (**mysql:9.1.0**) installieren.



# MySQL Datenbank installieren & starten

Tutorial

## MySQL

```
docker run --name local-mysql \
-e MYSQL_ROOT_PASSWORD=12345 \
-e MYSQL_DATABASE=wordpress \
-e MYSQL_USER=wordpress \
-e MYSQL_PASSWORD=wordpress \
-v mysql-data:/var/lib/mysql \
-d mysql:9.1.0
```

(Auf Windows) mit «git bash» Terminal ausführen oder alternativ in einer Zeile (ohne Backslash)

**Umgebungsvariablen (e):** Der Container legt beim Starten automatisch eine Datenbank mit dem Namen wordpress und dem Benutzer/Passwort wordpress/wordpress an. Diese Werte können konfiguriert werden.

**Pull:** Wenn image nicht verfügbar ist, dann erfolgt **pull** automatisch

# Wordpress installieren & starten

Tutorial

## Wordpress

```
docker run --name local-wordpress -p 8082:80 -v wordpress-data:/var/www/html -d wordpress:6.7.0-php8.3
```

Docker Desktop PERSONAL

Containers [Give feedback](#)

Container CPU usage: 0.60% / 800% (8 CPUs available)

Container memory usage: 466.26MB / 15.06GB

Show charts

Search  Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Actions
local-mysql	ccffdca0d1cc	mysql:9.1.0		0.57%	[⋮]
local-wordpress	a4e341acd2de	wordpress:6.7.0-php8.3	8082:80	0.01%	[⋮]

Showing 2 items

Engine running | RAM 6.40 GB CPU 0.00% Disk 214.89 GB avail. of 269.43 GB

Terminal [New version available](#) [2]

WordPress > Setup Configuration

localhost:8082/wp-admin

English (United States) [Afrikaans](#) [አማርኛ](#) [Aragonés](#) [العربية](#) [العربية المغربية](#) [অসমীয়া](#) [گۈزىنى اۇز بىلەن](#) [Azerbaijan dili](#) [Беларуская мова](#) [Български](#) [বাংলা](#) [ଓଡ଼ିଆ](#) [Bosanski](#) [Català](#) [Cebuano](#) [Čeština](#) [Cymraeg](#) [Dansk](#) [Deutsch \(Schweiz, Du\)](#) [Deutsch \(Schweiz\)](#)

Continue

# Wordpress im Browser öffnen

Tutorial

http://localhost:8082 öffnen und die Datenbank konfigurieren, aber ...

The image displays three sequential screenshots of the WordPress setup configuration process:

- Screenshot 1:** Shows the language selection step. "English (United States)" is selected from a list of many languages. A "Continue" button is at the bottom.
- Screenshot 2:** Shows the welcome screen with the WordPress logo. It lists five steps: 1. Datenbank-Name, 2. Datenbank-Benutzername, 3. Datenbank-Passwort, 4. Datenbank-Host, and 5. Tabellen-Präfix. Below this, it provides instructions for generating a wp-config.php file and contacting hosting if needed. A "Los geht's!" button is at the bottom.
- Screenshot 3:** Shows the database configuration step. It has four input fields:
  - Datenbank-Name:** wordpress
  - Benutzername:** wordpress
  - Passwort:** wordpress
  - Datenbank-Host:** local-mysqlA "Senden" button is at the bottom. Below the fields, there is explanatory text about the wp-config.php file and support for multiple installations.

Entsprechend  
vorheriger Folie

Der DB-  
Container

# Docker Networks

Tutorial

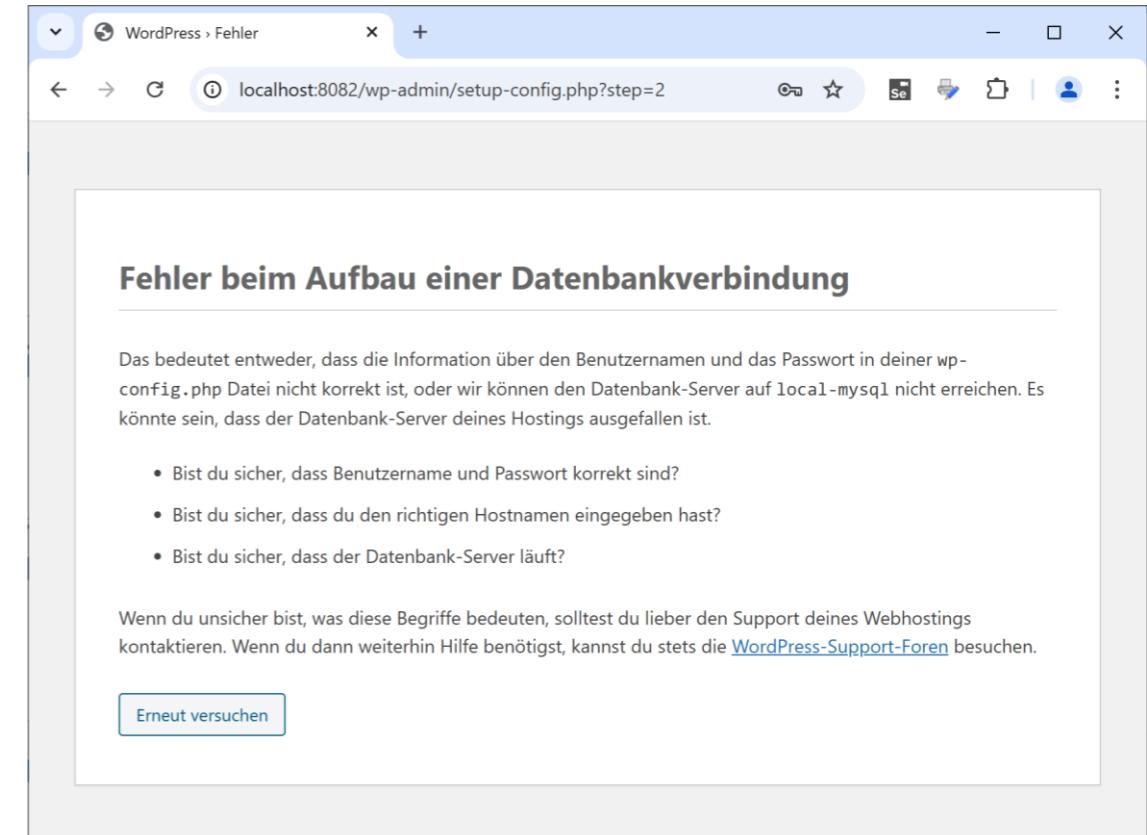
Die beiden Container können nicht miteinander kommunizieren → Docker Container sind isoliert.

## Virtuelles Netzwerk erstellen

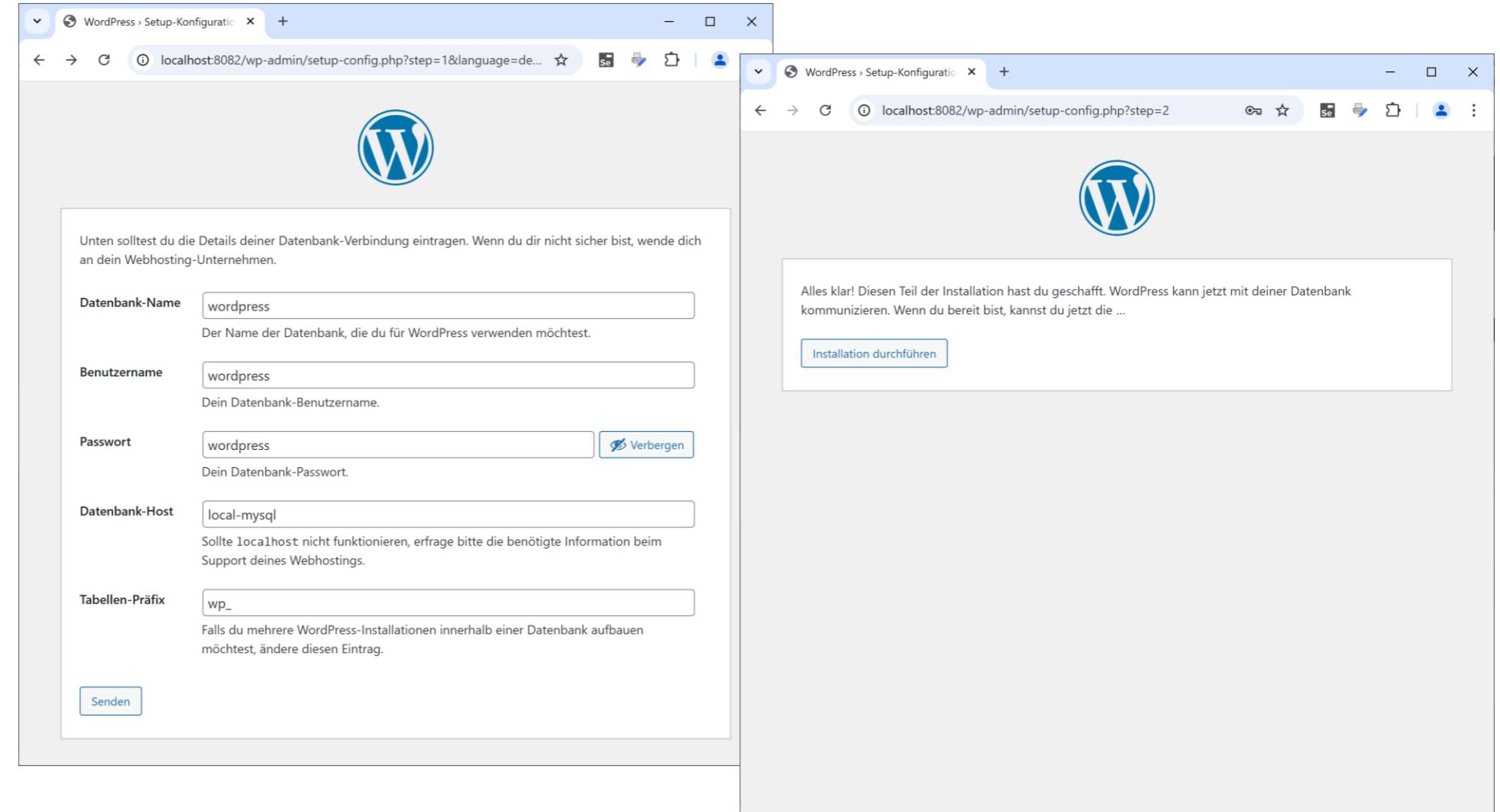
```
docker network create --attachable wordpress-network
```

## Container zu Netzwerk hinzufügen

```
docker network connect wordpress-network local-mysql  
docker network connect wordpress-network local-wordpress
```



Erneuter Versuch ist  
erfolgreich



# Fazit

## Mehrere Container

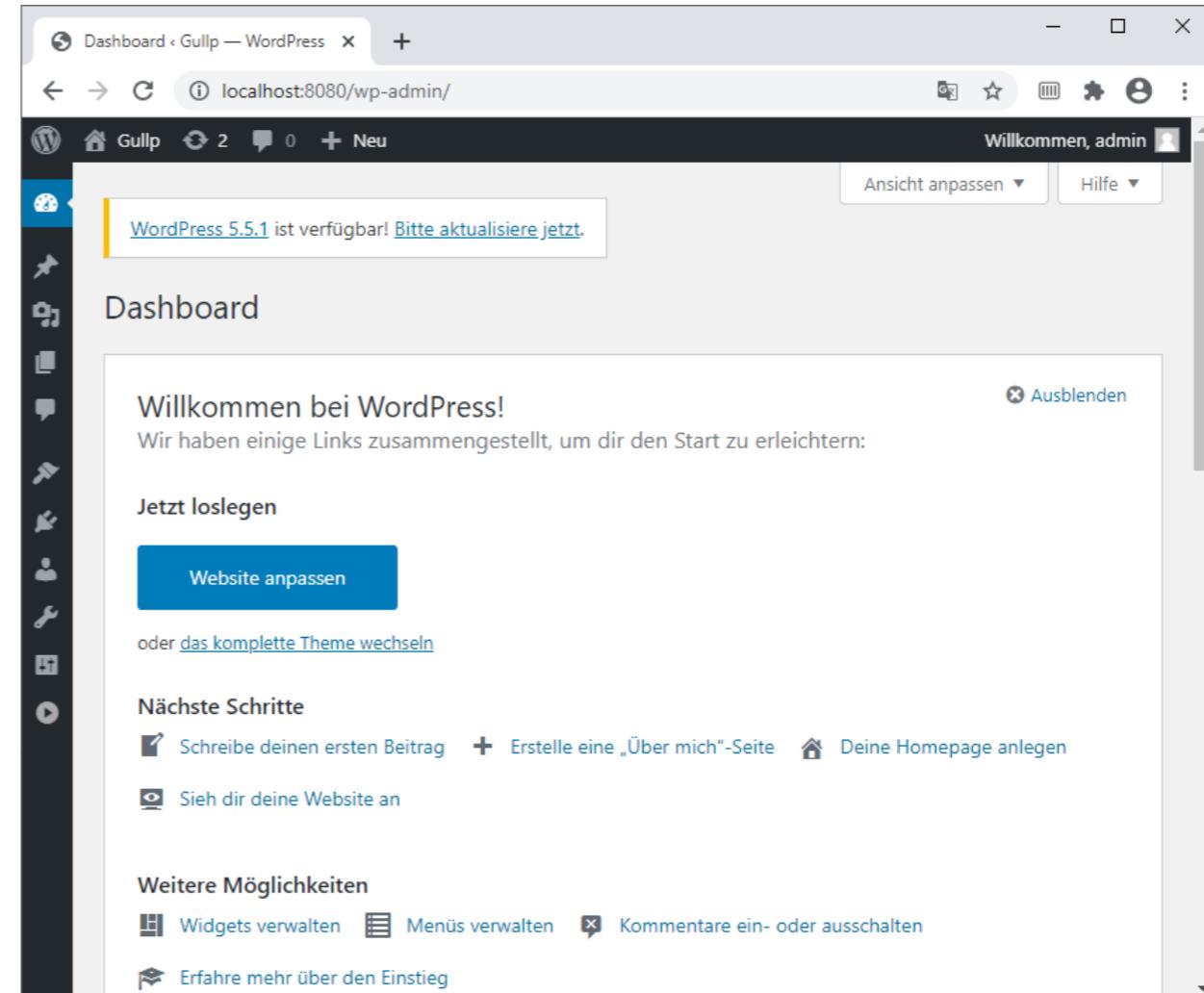
Die meisten Webapps bestehen aus mehreren Containern (z.B. Datenbank und Applikation)

## Netzwerke

Docker unterstützt Netzwerke

## Konfiguration

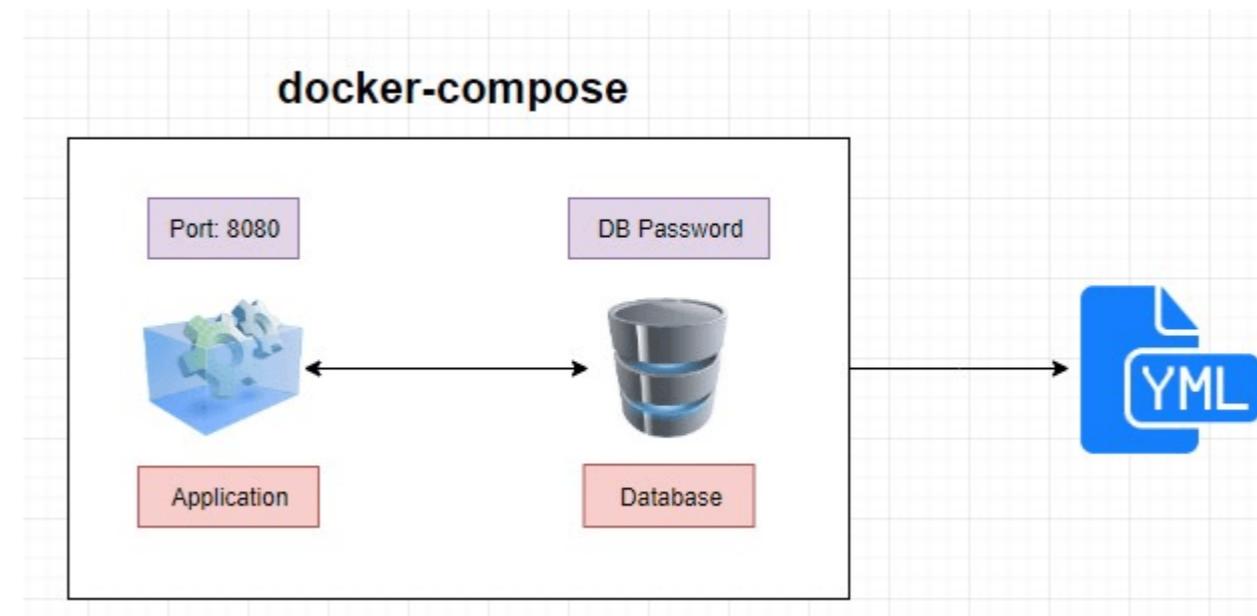
Die Konfiguration des Netzwerkes und der beiden Container kann aufwändig und fehleranfällig sein



# Docker Compose

Für Multi-Container Applications

Konfiguration mehrerer Container zusammenfassen



# Docker Compose File

## Docker Compose

Compose ist ein Tool, das Multi-Container Docker-Anwendungen definiert und ausführt.

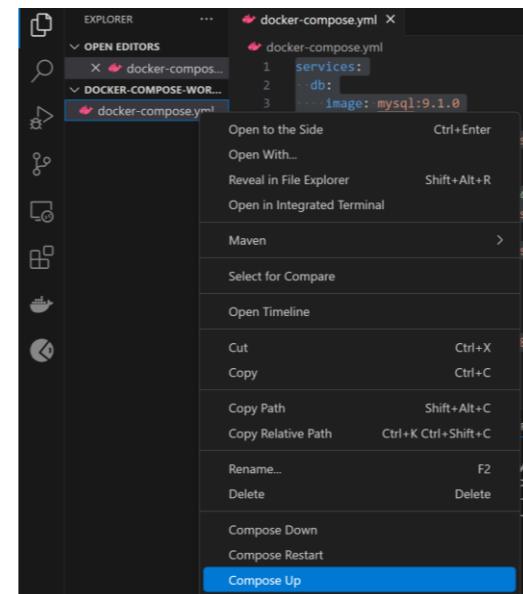
## YAML

Die Services werden in einem YAML-File beschrieben (**docker-compose.yml**). Mit einem einzigen Befehl können alle Services gestartet werden.

## Starten (Terminal)

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

## Starten (VS Code)



```
services:
  db:
    image: mysql:9.1.0
    volumes:
      - mysql-data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: 12345
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:6.7.0-php8.3
    ports:
      - "8082:80"
    restart: always
    volumes:
      - wordpress-data:/var/www/html
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress

volumes:
  mysql-data: {}
  wordpress-data: {}
```

# Docker Compose File

Tutorial

The screenshot shows the Docker Desktop interface. On the left, the sidebar has 'Containers' selected. The main area displays the 'Containers' page with three running containers listed:

Name	Container ID	Image	Port(s)	Actions
docker-compose-wordpress	-	-	-	[...]
wordpress-1	2e36cfec93f8	wordpress:6.7.0-php8.3	8082:80	[...]
db-1	27ab454b95b5	mysql:9.1.0	-	[...]

Resource usage statistics are shown at the top: Container CPU usage (0.59% / 800%) and Container memory usage (525.34MB / 15.06GB). A 'Show charts' button is available for monitoring.

At the bottom, system status is shown: 'Engine running', RAM usage (6.12 GB), CPU usage (0.00%), and Disk usage (218.31 GB avail. of 269.43 GB).

A separate browser window shows the WordPress installation wizard at [localhost:8082/wp-admin/install.php?language=de\\_CH\\_informal](http://localhost:8082/wp-admin/install.php?language=de_CH_informal). The page displays the 'Willkommen' (Welcome) screen with the WordPress logo, followed by the 'Benötigte Informationen' (Required Information) section where fields for website title, username, password, and email are filled out.

# Fazit Docker Compose

## Definition

Docker Compose ist ein Tool zur Definition und Ausführung von Multi-Container Docker-Anwendungen.

## YAML-Datei

Die Konfiguration der Services erfolgt in einer YAML-Datei («docker-compose.yml»).

## Einfacher Start

Mit dem Befehl «docker-compose up -d» können alle definierten Services gestartet werden.

# Dockerfile

Wie entstehen die Images auf Docker Hub?

# Image aus Container erstellen

## Inkonsistenz

Der Zustand des Containers kann variieren, was zu unterschiedlichen Images führt.

## Fehlende Dokumentation

Es gibt keine klare Dokumentation der Schritte, die zur Erstellung des Images geführt haben.

## Schwierige Reproduzierbarkeit

Es ist schwieriger, das Image in einer anderen Umgebung oder zu einem späteren Zeitpunkt exakt zu reproduzieren.

# Dockerfile

Die Lösung sind **Dockerfiles**. Dockerfiles lösen das Problem der konsistenten und wiederholbaren Erstellung von Docker-Images.



Dockerfile



docker build



Docker Image

# Dockerfile zusammenbauen

## Verzeichnis

Name	Date modified	Type	Size
Dockerfile	02.09.2020 16:09	File	1 KB
package.json	02.09.2020 16:07	JSON File	1 KB
server.js	02.09.2020 16:08	JavaScript File	1 KB

## Build-Command

```
docker build -t mosazhaw/image-name .
```

Befehl in  
Terminal/Konsole im  
Verzeichnis ausführen  
(Punkt = aktuelles  
Verzeichnis)

# Dockerize Python Examples

# Sentiment Prediction mit Python

```
docker build -t mosazhaw/onnx-sentiment-analysis .
```

```
FROM python:3.12.7
```

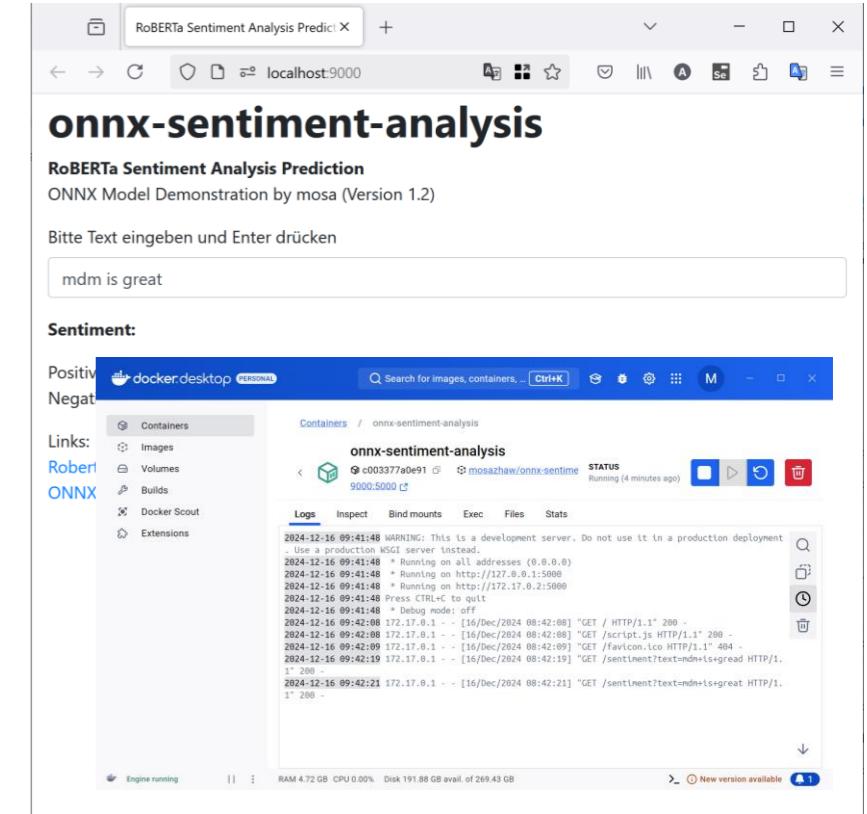
```
# Copy Files  
WORKDIR /usr/src/app  
COPY . .
```

```
# Install  
RUN pip install -r requirements.txt
```

```
# Docker Run Command  
EXPOSE 5000  
WORKDIR /usr/src/app/app  
CMD [ "python", "-m" , "flask", "run", "--host=0.0.0.0" ]
```

```
docker run --name onnx-sentiment-analysis -p 9000:5000 -d mosazhaw/onnx-sentiment-analysis
```

Problem: Alles wird kopiert, auch `.venv`



# .dockerignore

Die Idee von .dockerignore entspricht dem Konzept von .gitignore. Docker (insbesondere das COPY Command) ignoriert alle Dateien / Verzeichnisse in .dockerignore

.dockerignore um **/.venv** inklusive Inhalt zu ignorieren:

/ .venv

# comment	Kommentare werden ignoriert
* /temp *	Alle Dateien/Ordner, welche mit temp beginnen, in beliebigem direkten Unterverzeichnis ignorieren: /somedirA/temporary.txt oder /somedirB/temp
* !app.py	Alles ignorieren, ausser app.py (! für Ausnahmen)
temp?	Dateien im Root-Ordner wie /tempa oder /tempb ausschliessen

# Image Classification mit Python

```
docker build -t mosazhaw/onnx-image-classification .
```

```
FROM python:3.13.0
```

```
# Copy Files
```

```
WORKDIR /usr/src/app
```

```
COPY app.py *.onnx labels_map.txt requirements.txt ./
```

```
COPY web web
```

```
# Install
```

```
RUN apt-get update && apt-get install -y libgl1
```

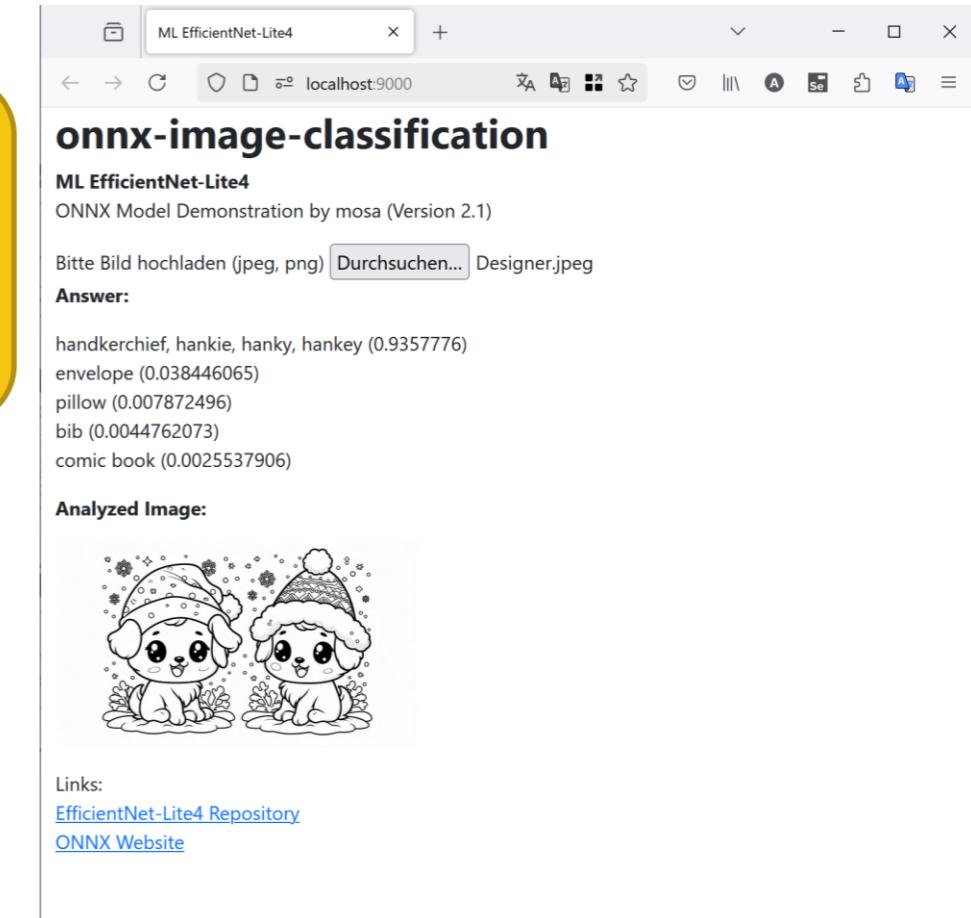
```
RUN pip install -r requirements.txt
```

```
# Docker Run Command
```

```
EXPOSE 5000
```

```
CMD [ "python", "-m" , "flask", "run", "--host=0.0.0.0" ]
```

Falls Software bei FROM nicht enthalten, zusätzlich installieren



```
docker run --name onnx-image-classification -p 9000:5000 -d mosazhaw/onnx-image-classification
```

## Docker Hub Push

Docker Konto erforderlich

# Docker Hub: Repository erstellen

The screenshot shows the Docker Hub interface. At the top, there is a navigation bar with links for 'Explore', 'Repositories', 'Organizations', 'Help', and a user profile for 'innovad'. Below the navigation bar, there is a search bar and a dropdown menu set to 'Content'. A prominent blue button labeled 'Create repository' is visible. A green callout bubble with the text 'Repository erstellen' points to this button. In the main content area, there is a list of repositories under the heading 'innovad'. One repository, 'innovad / dlj-footwear-classification', is shown with details: 'Contains: Image | Last pushed: 5 days ago', 'Not Scanned', 0 stars, 23 downloads, and 'Public'. Below this, another green callout bubble with the text 'Name wählen und Public Repo erstellen' points to the 'Create repository' form. The form itself has fields for 'Name' (set to 'innovad') and 'Description', and a 'Visibility' section where the 'Public' option is selected.

# Docker Hub: Push

Sentiment Analysis bauen und pushen:

```
docker build -t mosazhaw/onnx-sentiment-analysis .
docker push mosazhaw/onnx-sentiment-analysis:latest
```

Image Classification bauen und pushen:

```
docker build -t mosazhaw/onnx-image-classification .
docker push mosazhaw/onnx-image-classification:latest
```

## Azure und Docker

Azure App Service with Container

Azure Container Instances (ACI)

Azure Container Apps (ACA)

# App Service vs. Instances vs. Apps

Azure App Service

Azure Container Instances (ACI)

Azure Container Apps (ACA)

Azure App Service lässt sich alternativ auch mit Containern verwenden.

ACA ist der neuere Service und bietet bessere Integration in Azure.

ACI ist einfacher zu verwenden und besser in Docker integriert.

Achtung: Je nach Preisplan und Region sind nicht alle Services verfügbar!

Weiterführende Informationen:

<https://medium.com/microsoftazure/app-services-aci-container-apps-aks-which-one-is-right-for-my-app-e77fc3501fe>

## Azure App Service

Azure App Service benötigt folgende Schritte

- Ressourcengruppe erstellen (sofern nicht bereits vorhanden)
- App Service Plan erstellen
- Webapp basierend auf Container erstellen

# Azure Ressourcengruppe

Eine Ressourcengruppe ist ein Container, der zugehörige Ressourcen für eine Azure-Lösung enthält.

```
az group create --name mdm-appservice --location switzerlandnorth
```

Die Ressourcengruppe kann alle Ressourcen für die Lösung enthalten oder nur die Ressourcen, die Sie als Gruppe verwalten möchten. Sie entscheiden, wie Sie Ressourcen Ressourcengruppen zuweisen möchten, basierend darauf, was für Ihre Organisation am sinnvollsten ist.

# Azure App Service: Web App Container

App Service Plan erstellen

```
az appservice plan create --name onnx-image-classification --resource-group mdm-appservice --sku F1 --is-linux
```

Vorher erzeugte Ressourcengruppe

App erstellen

```
az webapp create --resource-group mdm-appservice --plan onnx-image-classification --name onnx-image-classification --container-image mosazhaw/onnx-image-classification:latest
```

--container-image: eigenen Container verwenden

Vorher erzeugter Plan

-- name: Wird als Subdomain verwendet, daher mit eigenem Kürzel ergänzen

# Deployment

App ist online  
Bereitstellungscenter  
Protokollstream

**ML EfficientNet-Lite4**

ONNX Model Demonstration by mosa (Version 2.1)

Bitte Bild hochladen (jpeg, png)  tanne.png

**Answer:**

- pinwheel (0.50537634)
- hair slide (0.30795935)
- hatchet (0.026247688)
- lampshade, lamp shade (0.017697498)
- table lamp (0.017074354)

**Analyzed Image:**



Links:

- [EfficientNet-Lite4 Repository](#)
- [ONNX Website](#)

onnx-image-classification - Microsoft Edge

https://portal.azure.com/#@zhaw.onmicrosoft.com/resource/subscript... 90%

Microsoft Azure

Home > onnx-image-classification

onnx-image-classification | Protokollstream

Web-App

Suche

Übersicht

Aktivitätsprotokoll

Zugriffssteuerung (IAM)

Tags

Diagnose und Problembehandlung

Microsoft Defender for Cloud

Ereignisse (Vorschau)

Empfohlene Dienste (Vorschau)

Protokollstream

Bereitstellung

Bereitstellungsslots

Bereitstellungscenter

Einstellungen

Leistung

App Service-Plan

Entwicklungstools

API

Überwachung

Automation

Protokollebene Beenden Kopieren Löschen

Verbinden!

```
2024-12-16T08:30:06 Welcome, you are now connected to log-streaming service. Starting Log Tail -n 10 of existing logs ----/appsvtmp/volatile/logs/runtime/container.log
2024-12-16T08:30:03:87547172 Configure Services : 08.30.03.875237
2024-12-16T08:30:04.12531342 Configure : 08.30.04.125111
2024-12-16T08:30:04.27139882 Setting Up Routes : 08.30.04.271231
2024-12-16T08:30:04.37931482 Existing Configure : 08.30.04.379180
2024-12-16T08:30:04.41836982 [40m1m[33mwarn[39m22m49m:
Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
2024-12-16T08:30:04.41848072 No XML encryptor configured. Key {087fbaa2-d7d3-4547-a5cd-98ee0f4bac99} may be persisted to storage in unencrypted form.
2024-12-16T08:30:04.47444342 Hosting environment: Production
2024-12-16T08:30:04.47444292 Content root path: /opt/Kudu
2024-12-16T08:30:04.47444320 Now listening on: http://0.0.0.0:8181
2024-12-16T08:30:04.47443682 Application started. Press Ctrl+C to shut down. Ending Log Tail of existing logs ---Starting Live Log Stream ---
2024-12-16T08:30:51.55826072 WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
2024-12-16T08:30:51.55825282 * Debug mode: off
2024-12-16T08:30:51.60594052 * Running on all addresses (0.0.0.0)
2024-12-16T08:30:51.60597692 * Running on http://127.0.0.1:5000
2024-12-16T08:30:51.60598692 * Running on http://169.254.129.3:5000
2024-12-16T08:30:51.60601482 Press Ctrl+C to quit
2024-12-16T08:30:51.77609417 169.254.129.1 - [16/Dec/2024 08:30:51] "GET /robots933456.txt HTTP/1.1" 404 -
2024-12-16T08:30:52.13542652 169.254.129.1 - [16/Dec/2024 08:30:52] "GET / HTTP/1.1" 200 -
2024-12-16T08:30:52.13642712 169.254.129.1 - [16/Dec/2024 08:30:52] "GET / HTTP/1.1" 200 -
2024-12-16T08:30:52.17395722 169.254.129.1 - [16/Dec/2024 08:30:52] "GET / HTTP/1.1" 200 -
2024-12-16T08:30:52.22751702 169.254.129.1 - [16/Dec/2024 08:30:52] "GET / HTTP/1.1" 304 -
2024-12-16T08:30:52.25796292 169.254.129.1 - [16/Dec/2024 08:30:52] "GET /script.js HTTP/1.1" 200 -
2024-12-16T08:30:52.29106742 169.254.129.1 - [16/Dec/2024 08:30:52] "GET /favicon.ico HTTP/1.1" 404 -
```

# ACA

Azure Container Apps (ACA) benötigt folgende Schritte

- Ressourcengruppe erstellen (sofern nicht bereits vorhanden)
- Container App Environment erstellen
- Container App erstellen

Anleitung: <https://learn.microsoft.com/en-us/azure/container-apps/get-started>

# Azure Ressourcengruppe (für ACA)

Eine Ressourcengruppe ist ein Container, der zugehörige Ressourcen für eine Azure-Lösung enthält.

```
az group create --location switzerlandnorth --name mdm-aca
```

# ACA: Container App Environment

## Environment

```
az containerapp env create --name onnx-image-classification --resource-group mdm-aca --location westeurope
```

App erstellen (auch hier wieder eigenen Namen/Image und passende Gruppe/Plan/Environment verwenden)

```
az containerapp create --name onnx-image-classification --resource-group mdm-aca --environment onnx-image-classification --image mosazhaw/onnx-image-classification:latest --target-port 5000 --ingress external --query properties.configuration.ingress.fqdn
```

## Output und URL

```
Container app created. Access your app at https://efficientnet-lite4-onnx.calmgrass-7c479a9f.westeurope.azurecontainerapps.io/
```

# ACA: Troubleshooting: Log

The screenshot displays two windows related to troubleshooting an Azure container app.

**Left Window:** A browser window titled "ML EfficientNet-Lite4" showing the URL "onnx-image-classification.nicewave-5ac7cf9.westeurope.azurecontainerapps.io". The page title is "onnx-image-classification" and the sub-section is "ML EfficientNet-Lite4". It shows a file input field with the placeholder "Bitte Bild hochladen (jpeg, png)" and a button "Datei auswählen". Below the input field, there is a link "Keine ausgewählt". At the bottom, under "Links:", are two links: "EfficientNet-Lite4 Repository" and "ONNX Website".

**Right Window:** A Microsoft Azure portal window titled "onnx-image-classification | Protokollstream". The URL is "https://portal.azure.com/#@zhaw.onmicrosoft.com/resource/subscript...". The left sidebar shows navigation options like "Übersicht", "Aktivitätsprotokoll", "Zugriffssteuerung (IAM)", "Tags", "Diagnose und Problembehandlung", "Anwendung", "Einstellungen", "Überwachung", "Warnungen", "Metriken", "Protokolle", "Protokollstream" (which is selected), "Konsole", and "Ratgeberempfehlungen". The main area shows log entries:

```
Verbindung wird hergestellt...
2024-12-16T08:25:20.20987 Connecting to the container 'onnx-image-classification'...
2024-12-16T08:25:20.27574 Successfully Connected to container: 'onnx-image-classification' [Revision: 'onnx-image-classification--vkqv578-585d67f784-66ztr', Replica: 'onnx-image-classification--vkqv578']
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://100.100.197.61:5000
Press CTRL+C to quit
```

# ACI

## Azure Container Instances

# ACI Deployment

Ressourcengruppe erstellen

```
az group create --location switzerlandnorth --name mdm-aci
```

App erstellen (auch hier wieder eigenen Namen/Image und passende Gruppe/Plan/Label verwenden)

```
az container create --resource-group mdm-aci --name onnx-image-classification --image mosazhaw/onnx-image-classification:latest --dns-name-label onnx-image-classification --ports 5000
```

URL: <http://onnx-image-classification.switzerlandnorth.azurecontainer.io:5000/>

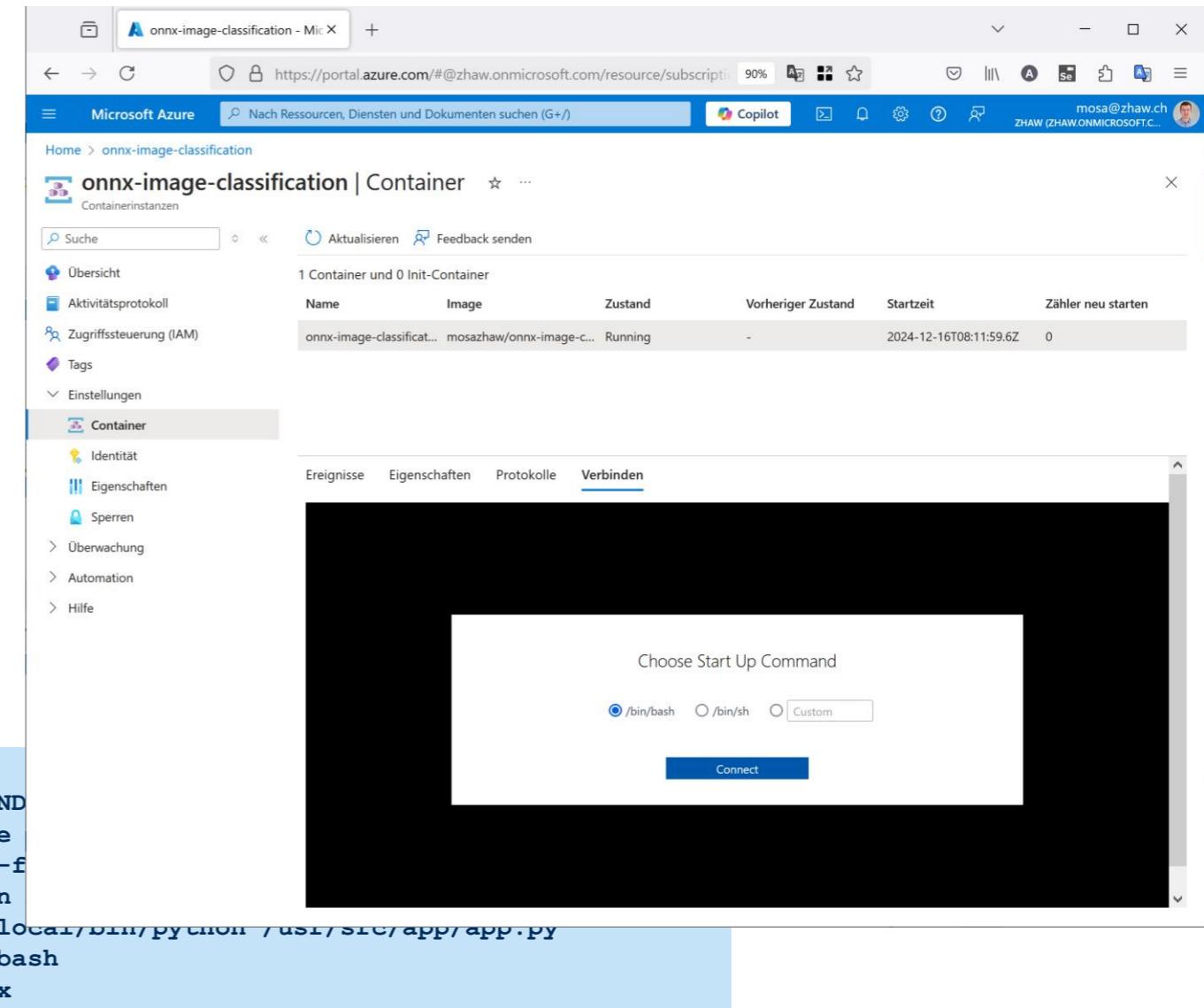
# ACI Troubleshooting: Deployment

The screenshot shows the Microsoft Azure portal interface for managing a container named 'onnx-image-classification'. The left sidebar contains navigation links such as Home, onnx-image-classification, Übersicht, Aktivitätsprotokoll, Zugriffssteuerung (IAM), Tags, Einstellungen, Container, Identität, Eigenschaften, Sperren, Überwachung, Automation, and Hilfe. The main content area displays the container's status with 1 Container and 0 Init-Container. A table lists the container's details: Name (onnx-image-classification), Image (mosazhaw/onnx-image-c...), Zustand (Running), Vorheriger Zustand (-), Startzeit (2024-12-16T08:11:59.6Z), and Zähler neu starten (0). Below this, the 'Ereignisse' tab of the activity log is selected, showing three entries: Started (Normal, 16.12.2024, 09:11:59 M..., 16.12.2024, 09:11:59 M..., Started container, 1), Pulled (Normal, 16.12.2024, 09:11:50 M..., 16.12.2024, 09:11:50 M..., Successfully pulled ima..., 1), and Pulling (Normal, 16.12.2024, 09:11:06 M..., 16.12.2024, 09:11:06 M..., pulling image "mosazh..., 1).

Name	Image	Zustand	Vorheriger Zustand	Startzeit	Zähler neu starten
onnx-image-classificat...	mosazhaw/onnx-image-c...	Running	-	2024-12-16T08:11:59.6Z	0

Ereignisse	Eigenschaften	Protokolle	Verbinden		
Zeitzone anzeigen	<input checked="" type="radio"/> Ortszeit	<input type="radio"/> UTC			
Name	↑↓ Typ	↑↓ Erster Zeitstempel	↑↓ Letzter Zeitstempel	↑↓ Meldung	↑↓ Anzahl
Started	Normal	16.12.2024, 09:11:59 M...	16.12.2024, 09:11:59 M...	Started container	1
Pulled	Normal	16.12.2024, 09:11:50 M...	16.12.2024, 09:11:50 M...	Successfully pulled ima...	1
Pulling	Normal	16.12.2024, 09:11:06 M...	16.12.2024, 09:11:06 M...	pulling image "mosazh...	1

# ACI Debug: Mit Container verbinden



## Prozesse und/oder Logs anschauen

```
root@sandboxhost-638055009988707785:/usr/src/app# ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root        1  0.0  0.0   1024     4 ?       Ss   14:17  0:00 /pause
root      255  0.0  0.0   1584     4 ?       Ss   14:31  0:00 tail -f
root      260  0.5 20.0 484936 201928 ?       Ss   14:31  0:01 python
root      269  0.9 20.2 484996 203972 ?       S1   14:31  0:01 /usr/local/bin/python /usr/src/app/app.py
root      307  0.0  0.3   6052  3876 pts/0    Ss   14:33  0:00 /bin/bash
root      329  0.0  0.3   8648  3360 pts/0    R+   14:34  0:00 ps aux
```

# GitHub Repositories

<https://github.com/mosazhaw/onnx-sentiment-analysis>

<https://github.com/mosazhaw/onnx-image-classification>

# Lernjournal 2 «Container»

## Ziele

- Docker-Konzept verstehen (Images, Container, Networks, Compose, Dockerfile)
- Deployment mit Azure Web Apps, ACA und ACI durchführen können

## Checkliste (Docker)

- ✓ Existierende Web-Applikation auf Docker Hub recherchiert und installiert, umfasst mind. zwei Container
  - ✓ Manuelles, lokales Deployment dokumentiert
  - ✓ Lokales Deployment mit Docker Compose dokumentiert (GitHub-Repo)
- ✓ Wahl einer der in der Vorlesung gezeigten Beispiele (onnx-sentiment-analysis, onnx-image-classification)
  - ✓ Lokaler Build des Containers mit Dockerfile
  - ✓ Lokales Deployment mit Docker
  - ✓ Push des eigenen Images auf Docker Hub
  - ✓ Azure Web App Deployment, ACA Deployment, ACI Deployment
  - ✓ Jeweils Browser-Ansicht, lokales Log und/oder Azure Portal und Deployment Log dokumentieren