

DevOps Container Virtualisierung



Building Competence. Crossing Borders.

Docker und Docker Compose

Wir verlassen das Thema «Dev» und bewegen uns Richtung «Ops».

Ziel

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

Idee

Docker vereinfacht die Bereitstellung von Anwendungen, weil sich Container, die alle nötigen Pakete enthalten, leicht als Dateien transportieren und installieren lassen.



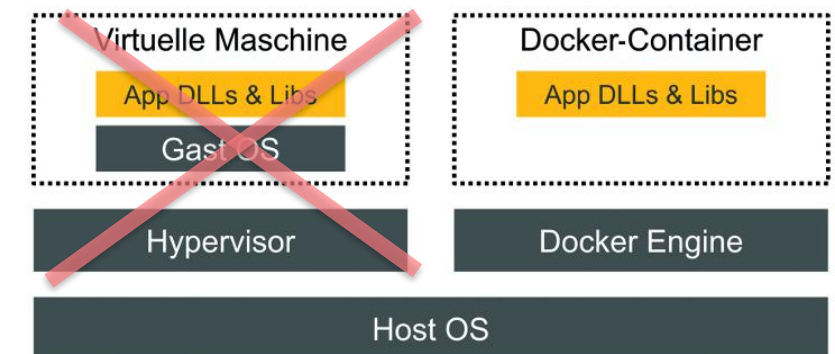
Docker Konzepte

Docker Konzepte

- Umgebung um Prozesse zu starten
- Saubere Trennung von Umgebungen
- Sandbox mit Zugriff auf definierte Ressourcen
- Einfaches Interface um Applikationen zu starten

Was Docker NICHT ist

- (Fast) keine Virtualisierung
(von Prozessoren, z.B. x86, ARM)
- Keine getrennte Kernel
- Kein Hypervisor (Virtual-Machine-Monitor, Gast-OS auf fremdem OS)



Kubernetes und Docker Swarm

Kubernetes, oder k8s (k - 8 Buchstaben – s, alles klar?), oder “Kube” als Abkürzung, ist eine Open-Source Plattform die Linux Container Operationen automatisieren kann.

Anders gesagt, man kann Gruppen von Linux Containern zusammenfassen und **Kubernetes** hilft diese zu verwalten.

Docker Swarm ist eine Alternative zu Kubernetes. Im Unterricht brauchen wir nur Docker und weder k8s noch Swarm.



Begriffe: Image und Container

Docker Image

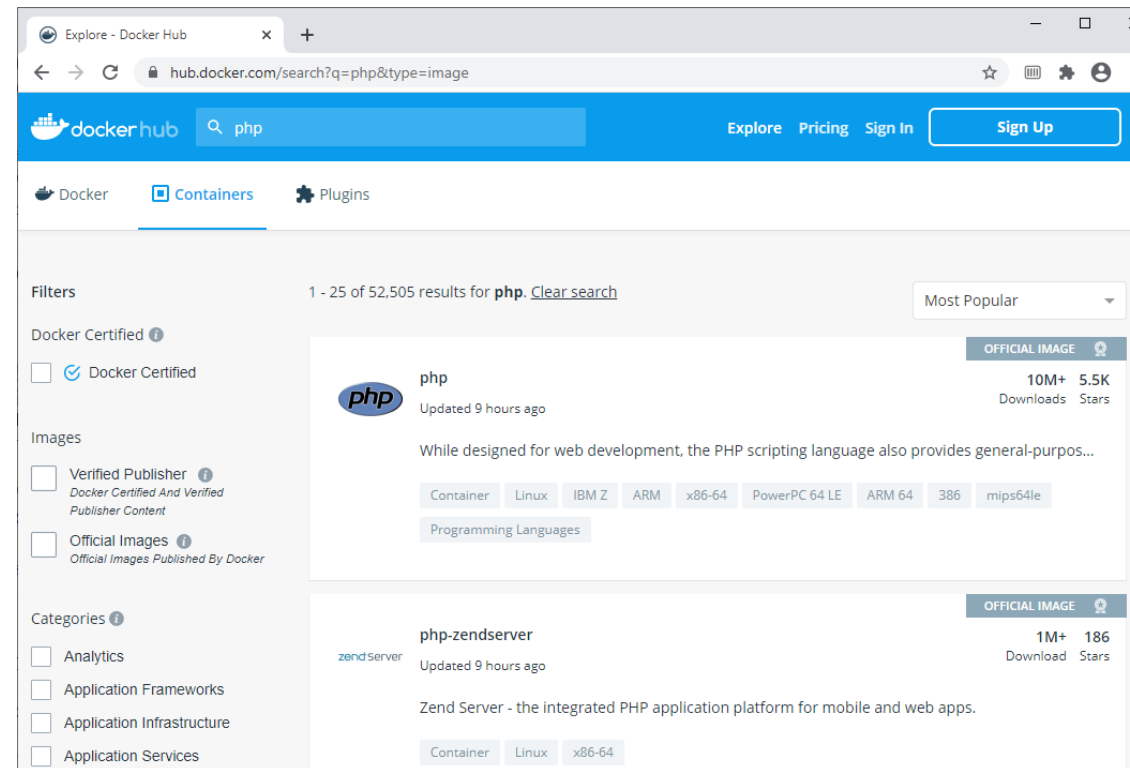
- Ein unveränderbares Template für Container
- Kann von einer Registry geladen bzw. dort gespeichert werden (z.B. Docker Hub)
- Image Namen haben die Form [registry/][user/]name[:tag]
- Der Standard-Tag ist latest
- Architektur-Support beachten: amd64 (PC, Intel Mac), arm64 (M1 Mac)

Docker Container

- Eine Instanz eines Image
- Lässt sich starten, stoppen, neustarten, ...
- Verwaltet Änderungen auf dem File System
- Neues Image könnte vom aktuellen Container-Stand gemacht werden (nicht empfohlen, stattdessen Dockerfile verwenden)

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

Docker

docker version

Docker Version

docker ps

Zeigt alle laufenden Container

docker ps -a

Zeigt alle Container

Docker Hub / Images

docker pull XYZ

Ein Image von DockerHub laden

docker image ls

Alle geladenen Images auflisten

docker image rm ID

Image löschen

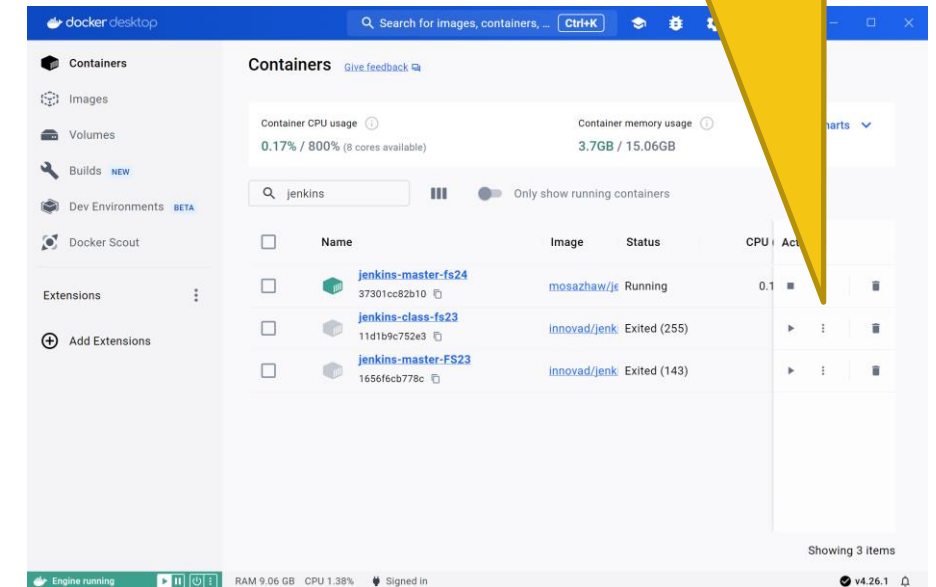
docker image prune -a

alle Images aufräumen

Docker Container

docker container ls

Auch über das Docker Dashboard lassen sich immer mehr Befehle ausführen



Docker Tutorial

Mediawiki installieren (Wikipedia-Software)

Docker Bedienung

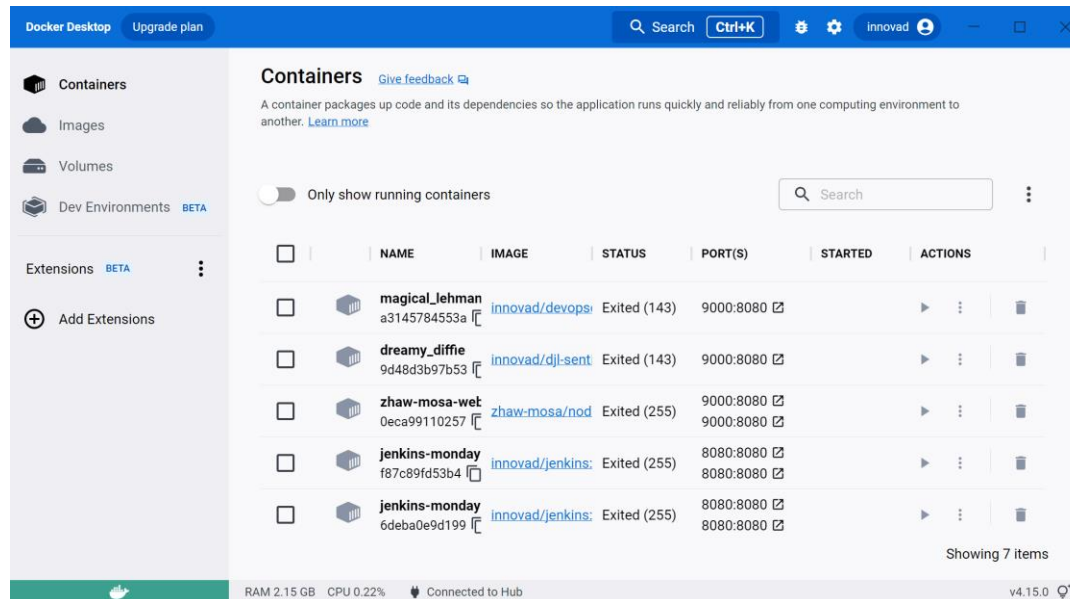
Tutorial

Docker

Die Bedienung erfolgt grundsätzlich über die Konsole (Windows cmd) oder Terminal (macOS).

Dashboard

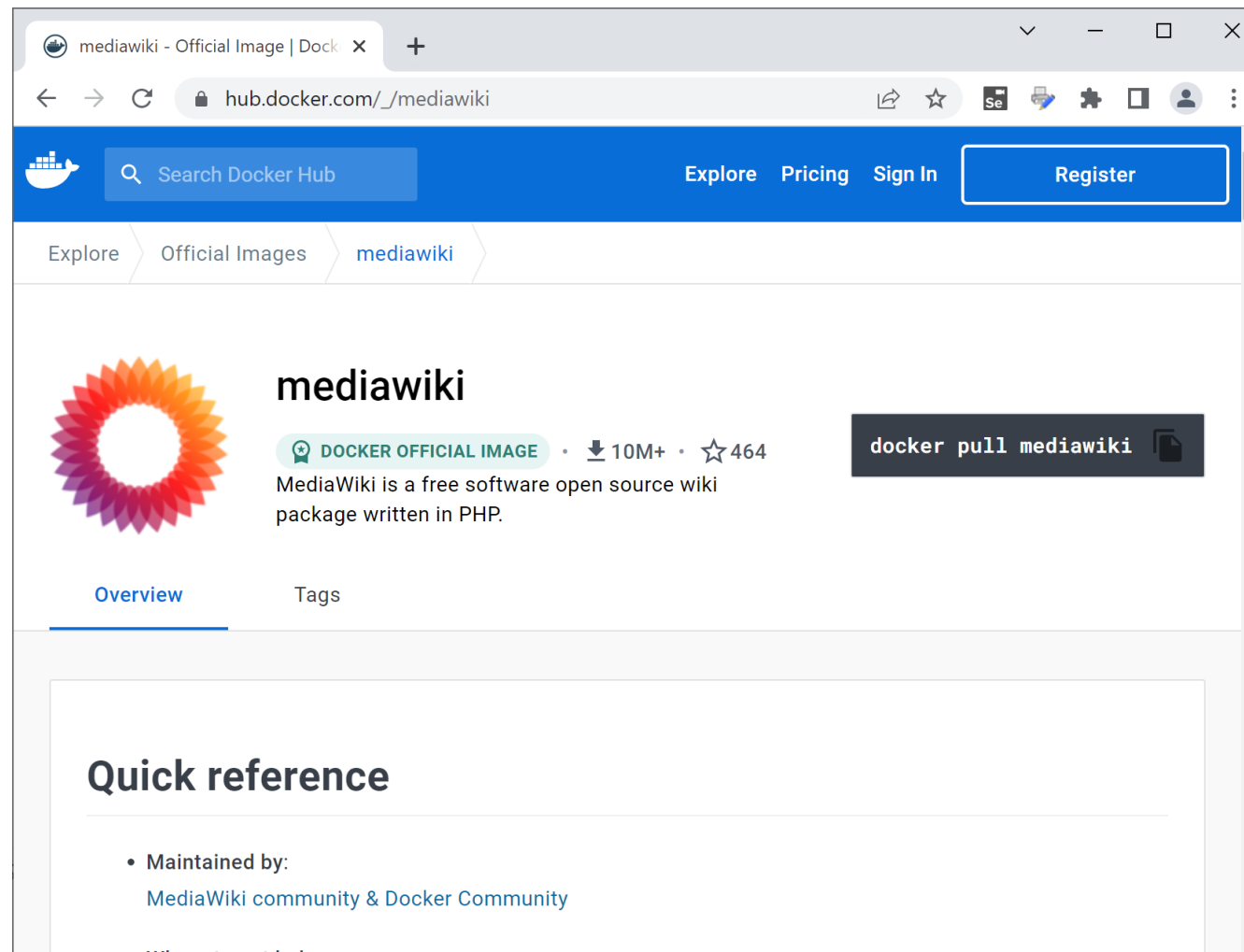
Das Dashboard bietet eine Übersicht sowie die Ausführung wichtiger Befehle.



Passendes Image suchen

Tutorial

https://hub.docker.com/_/mediawiki: Docker Pull Command für das gewünschte Image kopieren



docker pull und run

pull

`docker pull mediawiki`

Mit **tag** kann gewünschte Version geladen werden
z.B. `docker pull mediawiki:1.41`

1. Pull ausführen (mit Tag)

run

`docker run --name some-mediawiki -p 8080:80 -d mediawiki:1.41`

-d detached (nicht im Vordergrund)

image

Port des Docker Host

Interner Port
innerhalb Container

2. Starten

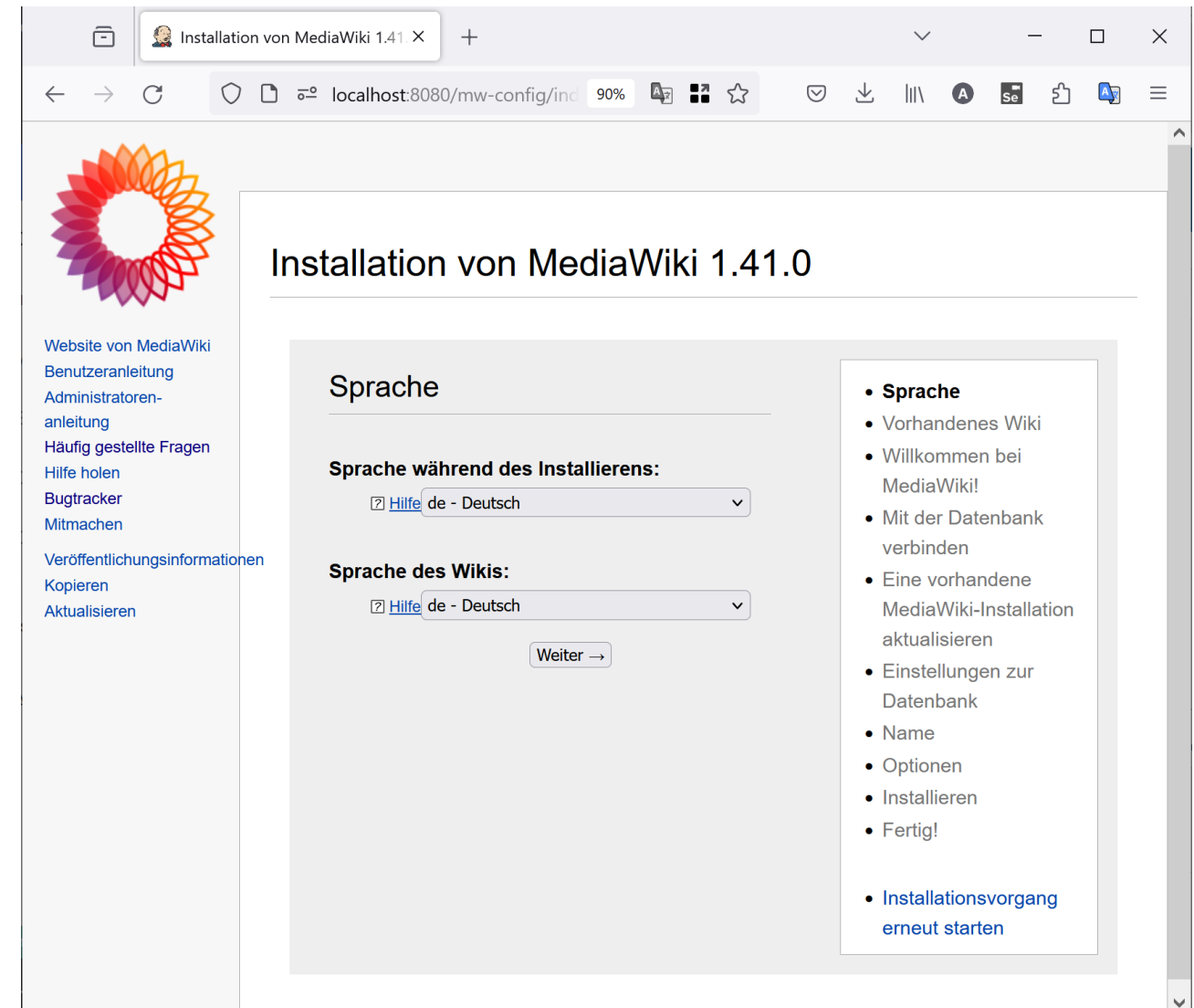
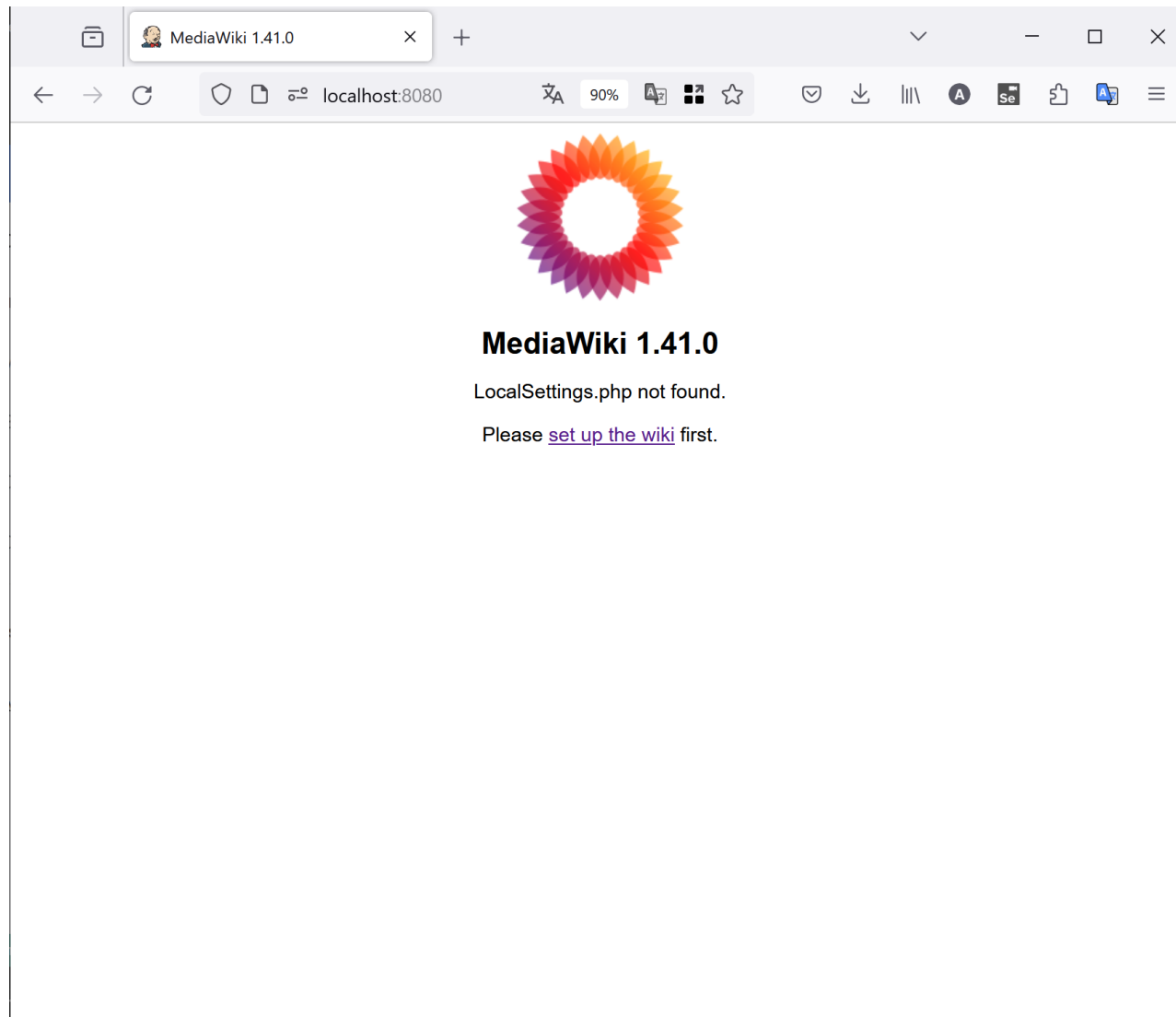
Nun sollte der Container im Dashboard erscheinen

The screenshot shows the Docker Desktop interface. On the left is a sidebar with navigation options: Containers, Images, Volumes, Builds (NEW), Dev Environments (BETA), Docker Scout, Extensions, and Add Extensions. The main area is titled 'Containers' and displays system metrics: 'Container CPU usage' at 0.23% / 800% (8 cores available) and 'Container memory usage' at 3.69GB / 15.06GB. A search bar contains the text 'media'. Below the search bar is a table of containers. One container is listed: 'some-mediawiki' (ID: 3ebb10602efa), based on the 'mediawiki:1.4' image, in a 'Running' state, using 0% CPU, with port '8080:80' mapped, and started '1 minute ago'. A yellow callout bubble points to the '8080:80' port mapping with the text 'Mediawiki in Browser öffnen'. The bottom status bar shows 'Engine running', system resources (RAM 10.39 GB, CPU 0.38%), and a 'Signed in' user.

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
some-mediawiki 3ebb10602efa	mediawiki:1.4	Running	0%	8080:80	1 minute ago	[Stop] [Refresh] [Delete]

Mediawiki: Up and running

Tutorial



Docker Tutorial «mehrere Container»

Mediawiki nutzt integrierte Datenbank

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

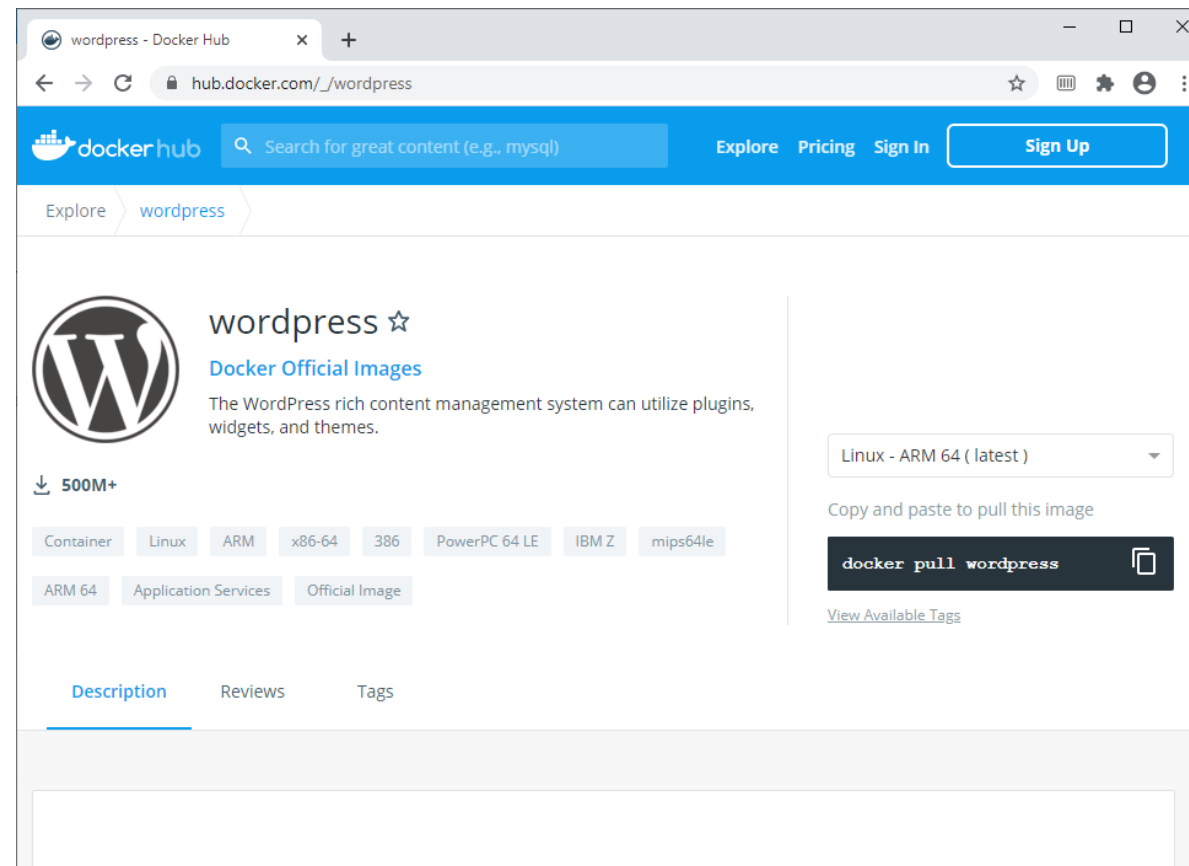
Mehrere Docker Instances

Netzwerke

Wordpress Image

Tutorial

Wir möchten Wordpress (**wordpress:6.4.2-php8.3**) installieren und brauchen dazu eine Datenbank (**mysql:8.2.0**)



MySQL

```
docker run --name local-mysql -e MYSQL_ROOT_PASSWORD=12345 -e MYSQL_DATABASE=wordpress -e  
MYSQL_USER=wordpress -e MYSQL_PASSWORD=wordpress -d mysql:8.2.0
```

Pull wird
automatisch
gemacht

Der Container legt beim Starten automatisch eine Datenbank mit dem Namen wordpress und dem Benutzer/Passwort wordpress/wordpress an. Diese Werte können konfiguriert werden.

- wenn image nicht verfügbar ist, dann erfolgt **pull** automatisch
- Image verlangt das Setzen diverser Parameter

Wordpress installieren & starten

Tutorial

Wordpress

```
docker run --name local-wordpress -p 8080:80 -d wordpress:6.4.2-php8.3
```

Pull wird
automatisch
gemacht

Wordpress im Browser öffnen

Tutorial

<http://localhost:8080> öffnen und die Datenbank konfigurieren, aber ...

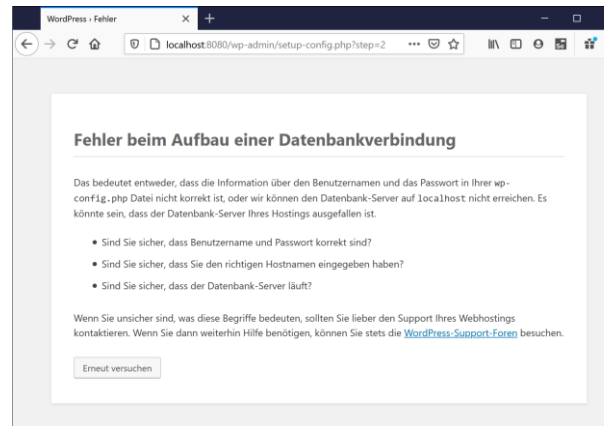
The image shows two screenshots of the WordPress Setup Configuration page. The left screenshot shows the language selection dropdown menu with 'English (United States)' selected. The right screenshot shows the database configuration form with the following fields:

- Datenbank-Name: wordpress
- Benutzername: wordpress
- Passwort: wordpress
- Datenbank-Host: local-mysql
- Tabellen-Präfix: wp_

Annotations in yellow callouts point to the 'Database Host' field and the 'Senden' button.

- Entsprechend vorheriger Folie (pointing to the 'Database Host' field)
- Der DB-Container (pointing to the 'Senden' button)

Die beiden Container können nicht miteinander kommunizieren.



Virtuelles Netzwerk erstellen

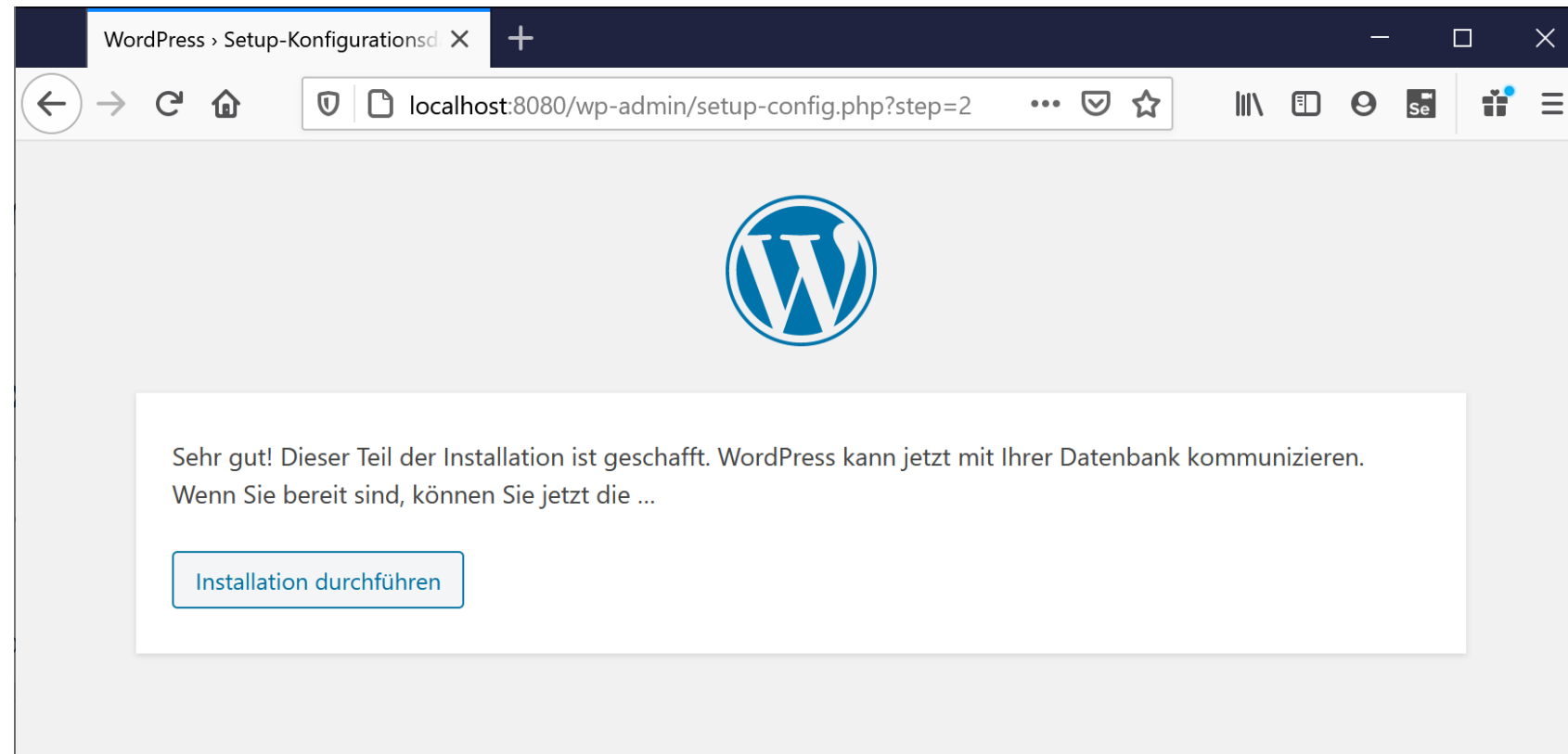
`docker network create --attachable wordpress-network`

Container zu Netzwerk hinzufügen

`docker network connect wordpress-network local-mysql`

`docker network connect wordpress-network local-wordpress`

Erneut Passwort eingeben und der Datenbank-Zugang sollte funktionieren:



Fazit

Mehrere Container

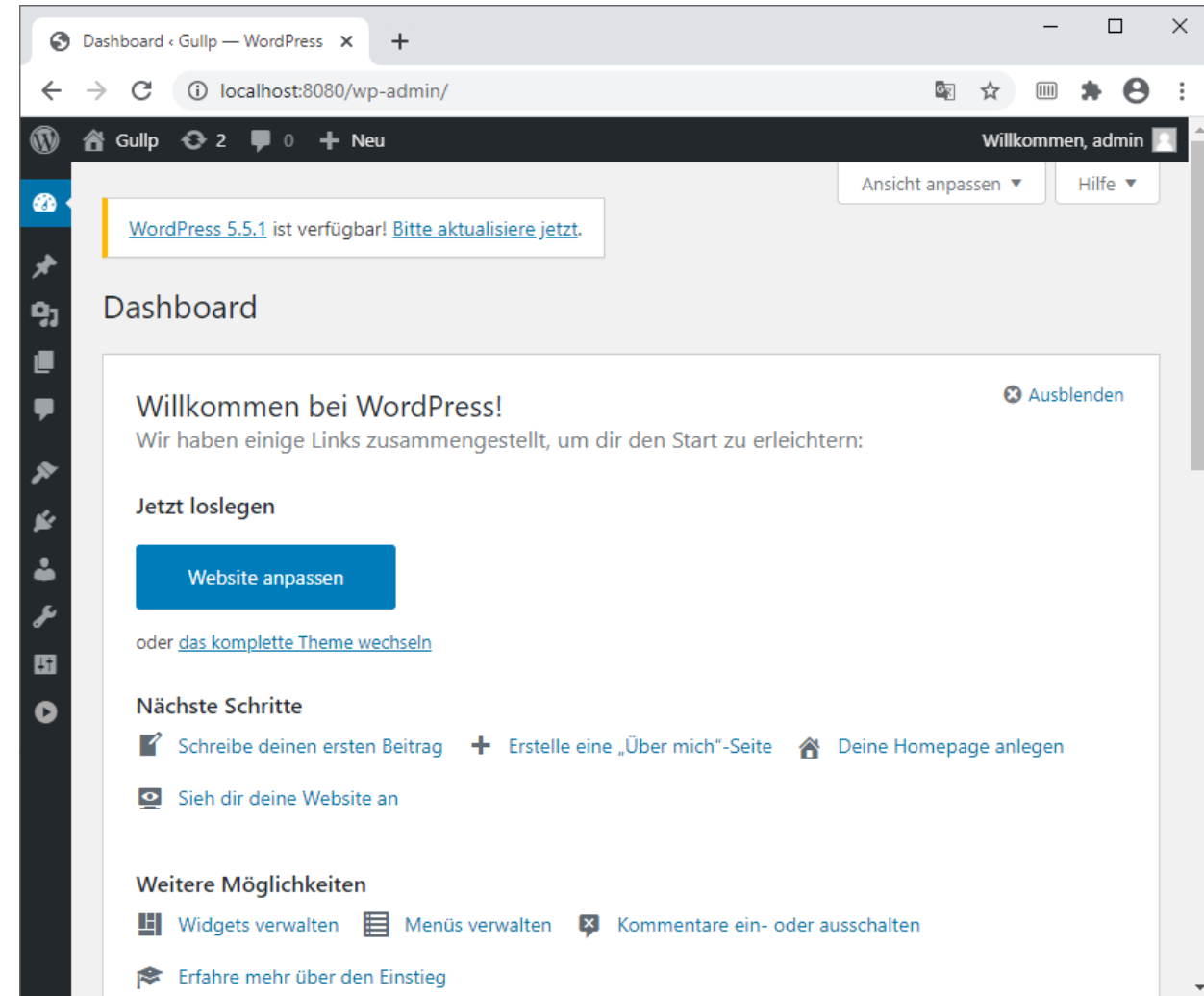
Die meisten Webapps bestehen aus mehreren Containern (z.B. Datenbank und Applikation)

Netzwerke

Docker unterstützt Netzwerke

Konfiguration

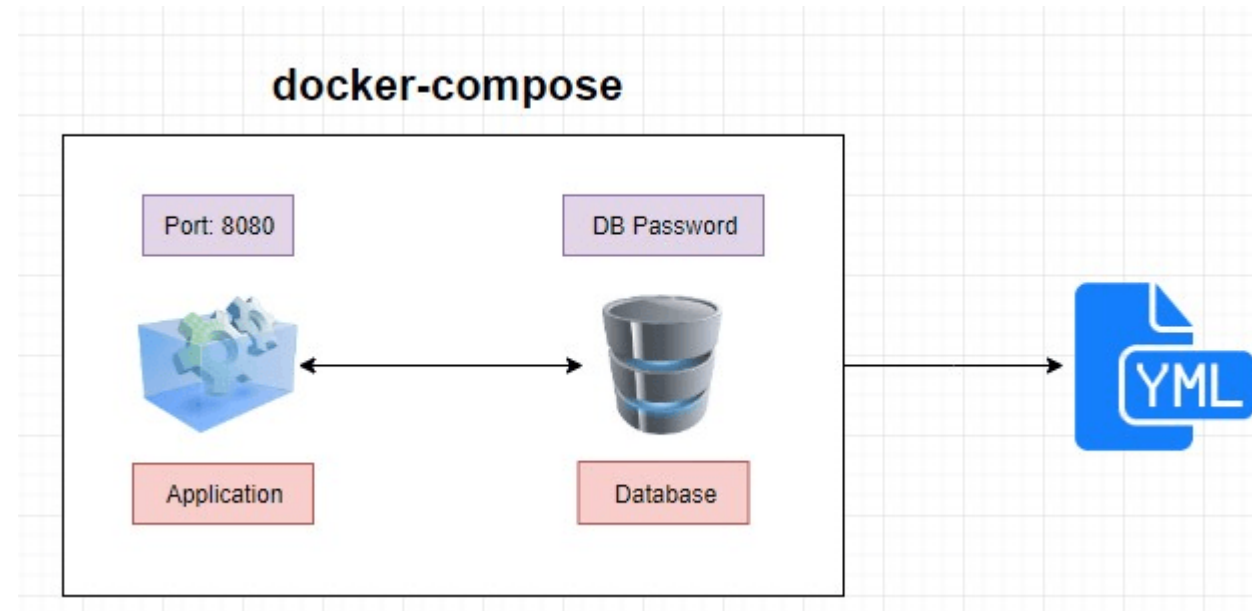
Die Konfiguration des Netzwerkes und der beiden Container kann aufwändig und fehleranfällig sein



Docker Compose

Für Multi-Container Applications

Konfiguration mehrerer Container zusammenfassen



Docker Compose File

Tutorial

Docker Compose

Compose ist ein Tool welches Multi-Container Docker Applikationen definieren und ausführen kann.

YAML

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

Starten (im Verzeichnis)

`docker-compose up -d`

```
version: '3.8'

services:
  db:
    image: mysql:8.2.0
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:6.4.2-php8.3
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    volumes:
      db_data: {}
```

Docker Compose File

Tutorial

The screenshot shows the Docker Desktop application window. The top bar is blue with the Docker logo and a search bar. The left sidebar contains navigation icons for Containers, Images, Volumes, Builds, Dev Environments, and Docker Scout. The main area is titled 'Containers' and displays system usage statistics (CPU and memory) and a table of running containers. The status bar at the bottom shows 'Engine running' and system resources.

Containers [Give feedback](#)

Container CPU usage *i* **0.12% / 800%** (8 cores available) Container memory usage *i* **3.7GB / 15.06GB** [Show charts](#) *v*

Search: word ☒ Only show running containers

<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	docker-compose-wordpress		Running (2/2)	0%		2 minutes ago	
<input type="checkbox"/>	wordpress-1 cd9308cf51c0	wordpress:6.	Running	0%	8000:80	2 minutes ago	
<input type="checkbox"/>	db-1 673e88d98bf6	mysql:8.2.0	Running	0%		2 minutes ago	

Showing 3 items

Engine running RAM 12.25 GB CPU 0.25% Signed in v4.26.1

Docker Compose

Mehrere Container auf einen Docker Host deployen

Docker Swarm (kein Thema dieser Vorlesung)

Container auf mehrere Docker Hosts deployen

Lernjournal

Ziele

- Docker-Konzept verstehen (Images und Container)
- Docker Networks anwenden können
- Docker Compose anwenden können

Checkliste

- ✓ Suche nach Applikation auf Docker Hub (Dokumentation), Prozessor-Architektur beachten
- ✓ Gefundene Webapplikation von Docker Hub installieren
- ✓ Applikation muss mindestens zwei Container umfassen
- ✓ Manuelles Deployment mit Docker testen, nachvollziehbare Dokumentation der lauffähigen Applikation
- ✓ Vorgehen dokumentieren
- ✓ Docker Compose File erstellen, auf GitHub verfügbar machen und dieses testen