Anwendungen von Semantik MediaWiki

Thema Docker

Raphael Manke, Patrick Eisele

Institut für Angewandte Informatik und Formale Beschreibungslehre (AIFB)

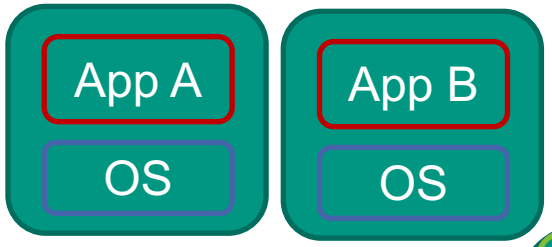
Motivation

- Sehr zeitaufwändig MediaWiki, Semantic MediaWiki jeweils auf verschiedenen Servern zu installieren, sowie die benötigten Entwicklungsumgebungen aktuell zu halten
- Problem: … funktioniert aber auf meiner Maschine, gelöst

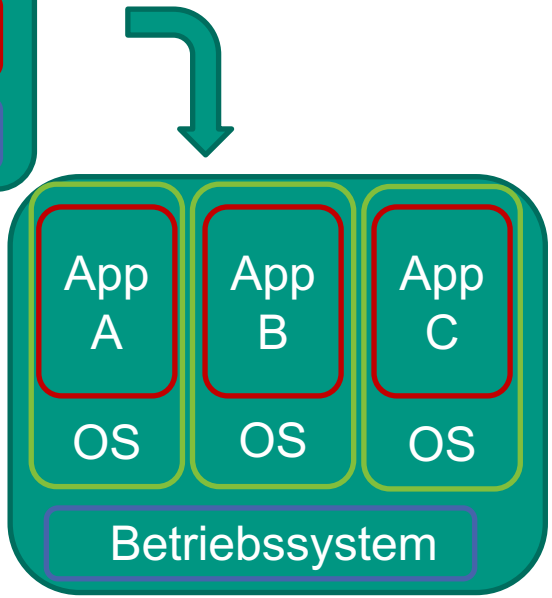
Ziele

- einfache Möglichkeit ein Semantik MediaWiki aufzusetzen
  - zum Testen / Ausprobieren
  - Demonstrationen
  - Entwicklungsumgebung
  - Deployment
- „Baukastensystem“ für den Benutzer
  - Oberfläche zum konfigurieren
  - click & run
- Migration „alter“ Semantik Media Wikis
  - einspielen einer Datenbank
  - Update auf neuste Version

Bare Metal vs. VM vs. Docker

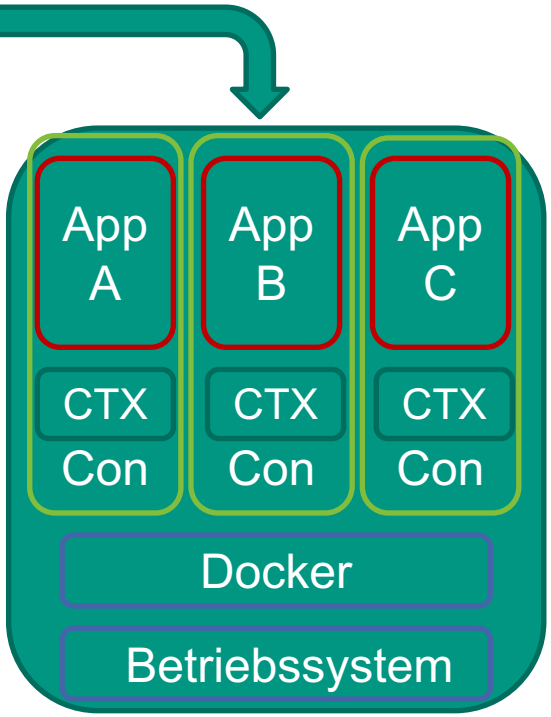


physischer Server
- hohe Betriebskosten
- ineffiziente Ressourecen-auslastung
- schlecht zu skalieren



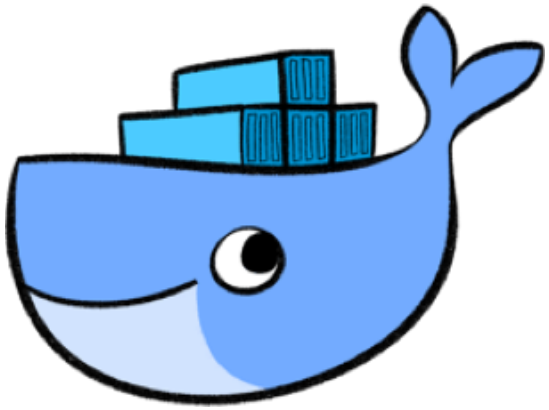
VM Server
- mehrere Anwendungen parallel
- leichtere Verteilung
- fixe Ressourcen-Verteilung
- Gast OS notwendig

Docker Server
- gemeinsame Ressourcen
- kein OS zu starten
- starke Portabilität



Docker

- Docker ist eine Containerplattform die es erlaubt Anwendungen in Containern auszuführen
  - Container = Anwendung + minimaler Ausführungskontext
  - Verteilung der Anwendung in viele kleine „Container“
- Kapselt alle für Anwendung benötigten Abhängigkeiten in einem virtuellem Container



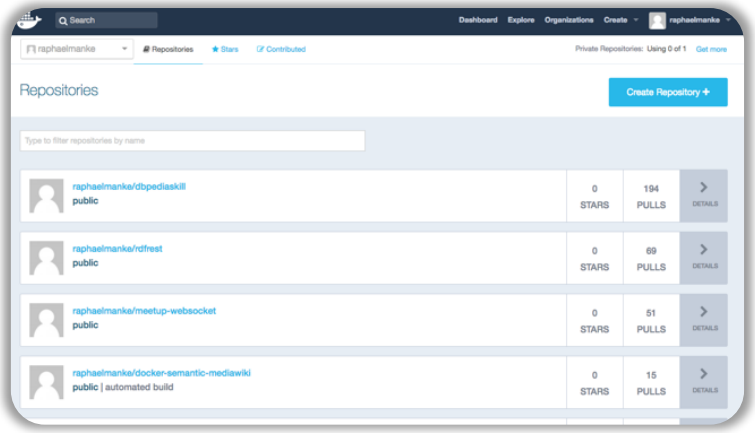
Docker Grundlagen

- Image
  - „unverändlicher“ Zustand einer virtuellen Umgebung
- Container
  - Instanz eines Image
  - Zusätliche Schicht auf Image
- Dockerfile
  - Spezifiziert ein Image
  - Basis für Container
- Docker Build
  - Baut ein Image aus einem Dockerfile
    - Bsp. `docker build -t smw .`
- Docker Run
  - Erstellt einen Container aus Image
    - Bsp. `docker run -p 8080:8080 smw`

```
ARG tag
FROM mediawiki:${tag:-1.30}
MAINTAINER Raphael Manke, Patrick Eisele

ENV COMPOSER_ALLOW_SUPERUSER = 1

RUN apt-get update && apt-get install -y \
        nano \
        wget \
        jq \
        unzip \
        libpng-dev

# install needed php extension
RUN docker-php-ext-install gd

# Copy scripts
COPY scripts ./scripts/check_db.sh ./
# Run the composer installation
RUN sh "install_composer.sh"

# Install semantic mediawiki
RUN php composer.phar require mediawiki/semantic-media-wiki "~2.5" --update-no-dev

# At containerstart run check script and then http server
CMD /var/www/html/check_db.sh && apache2-foreground
```

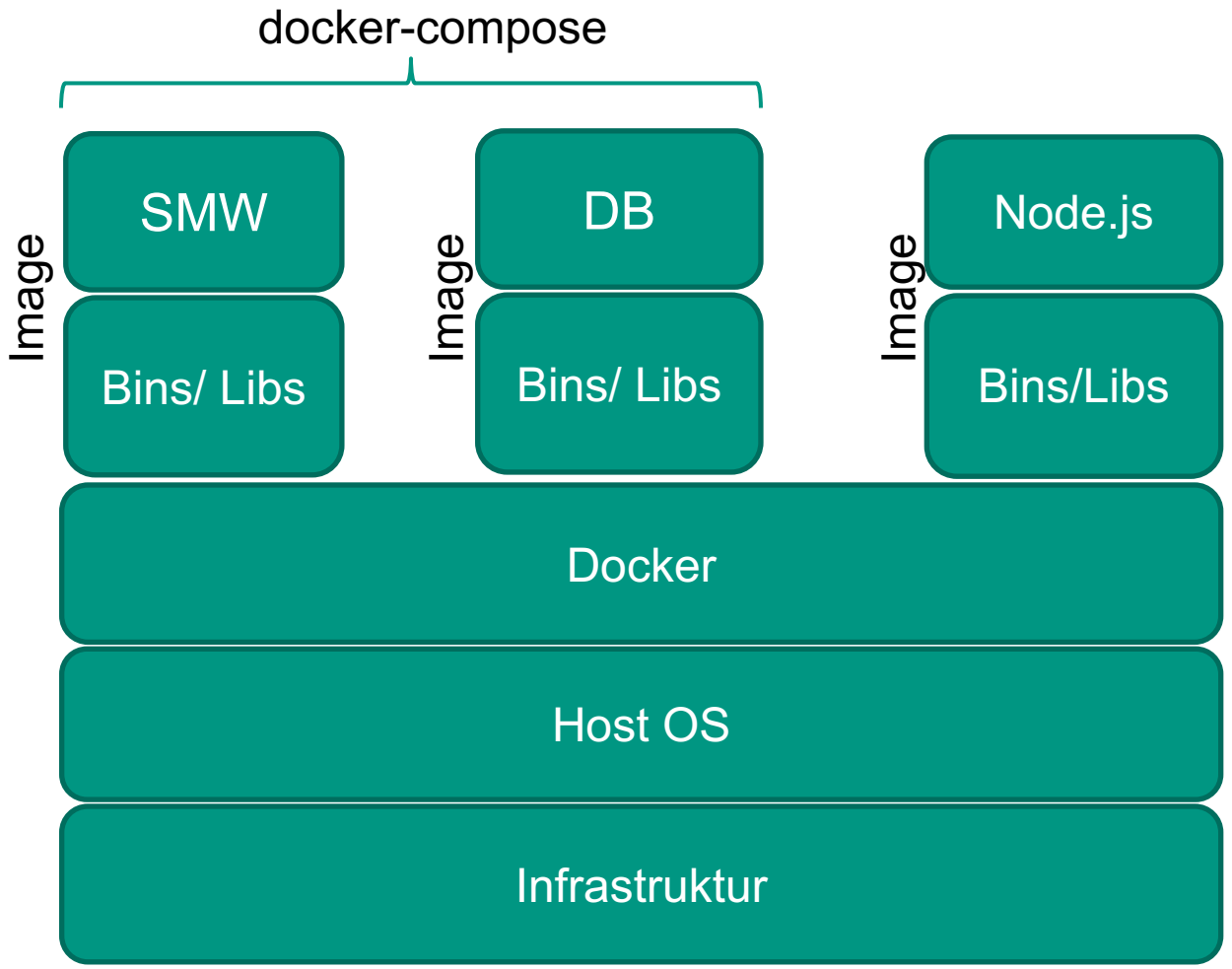
Docker Grundlagen

- Docker compose
  - Komposition von Containern
  - Spezifiziert Konfiguration
    - Bsp docker.compose.yml
  - Startet mehrere Container
    - Bsp. `docker-compose up`
- Docker Hub
  - Repository für Images
  - Autobuild Funktion

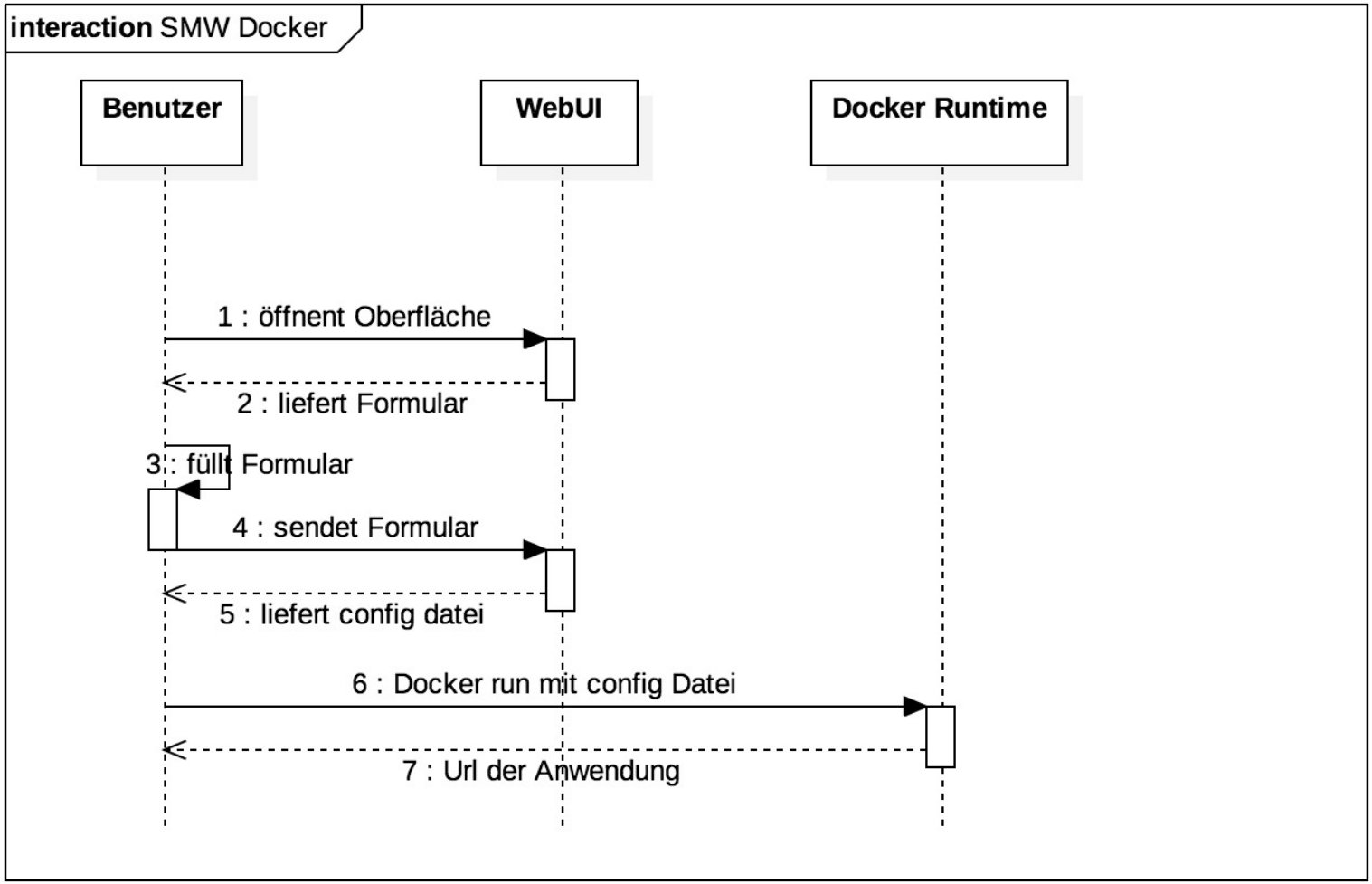



```yaml
version: '3'
services:
  mediawiki:
    image: raphaelmanke/docker-semantic-mediawiki
    restart: always
    ports:
      - ${HTTP_PORT:-8080}:80
    links:
      - database
    volumes:
      # After initial setup, download LocalSettings.php to the same directory as
      # this yaml and uncomment the following line and use compose to restart
      # the mediawiki service
      # - ./LocalSettings.php:/var/www/html/LocalSettings.php
      - ./extensions.json:/var/www/html/extensions.json
    depends_on:
      - database
  database:
    image: mariadb
    restart: always
    ports:
      - 3306:3306
```

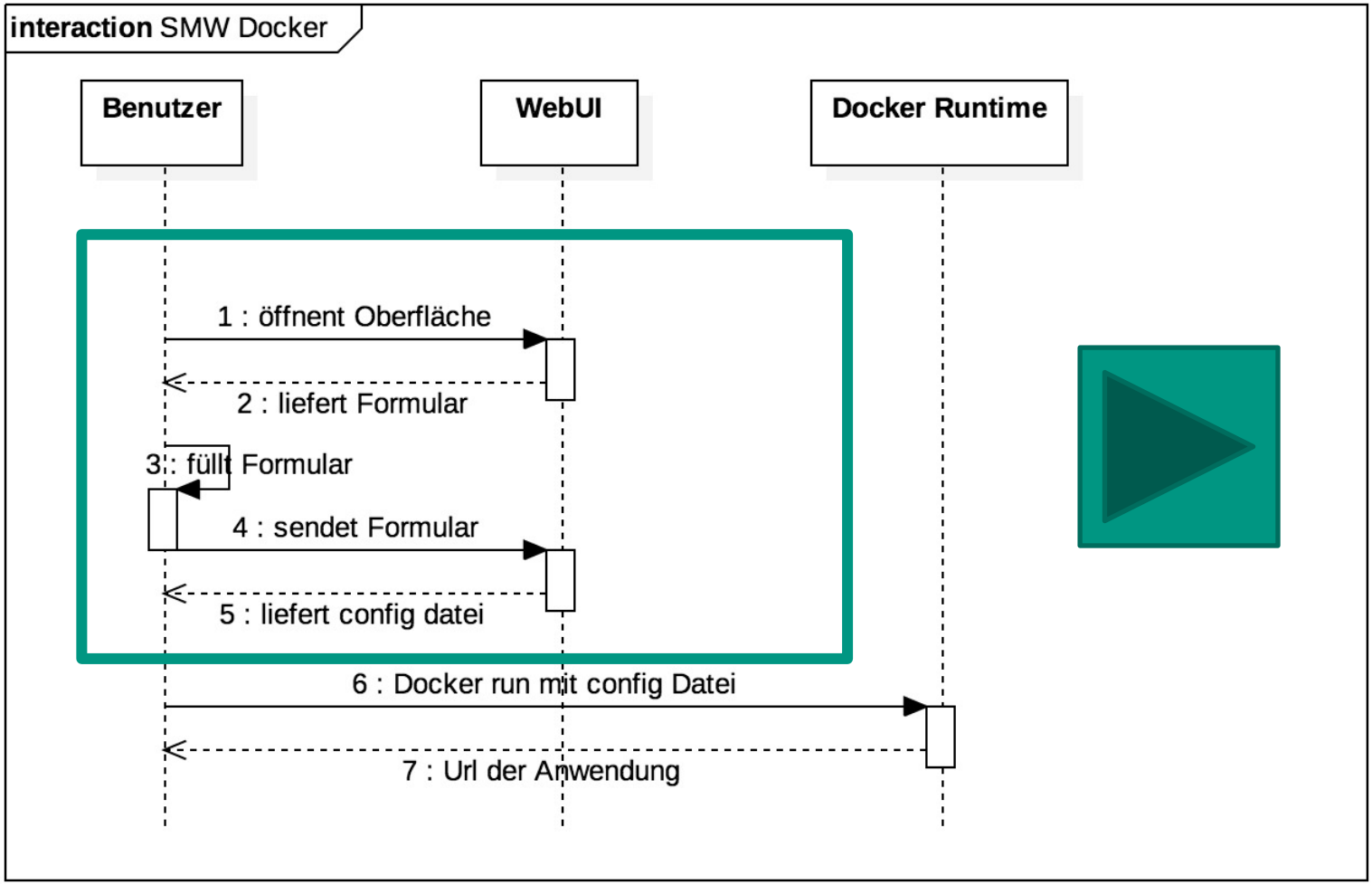
Lösungsansatz schematisch



Lösungsansatz schematisch



Weboberfläche Demo



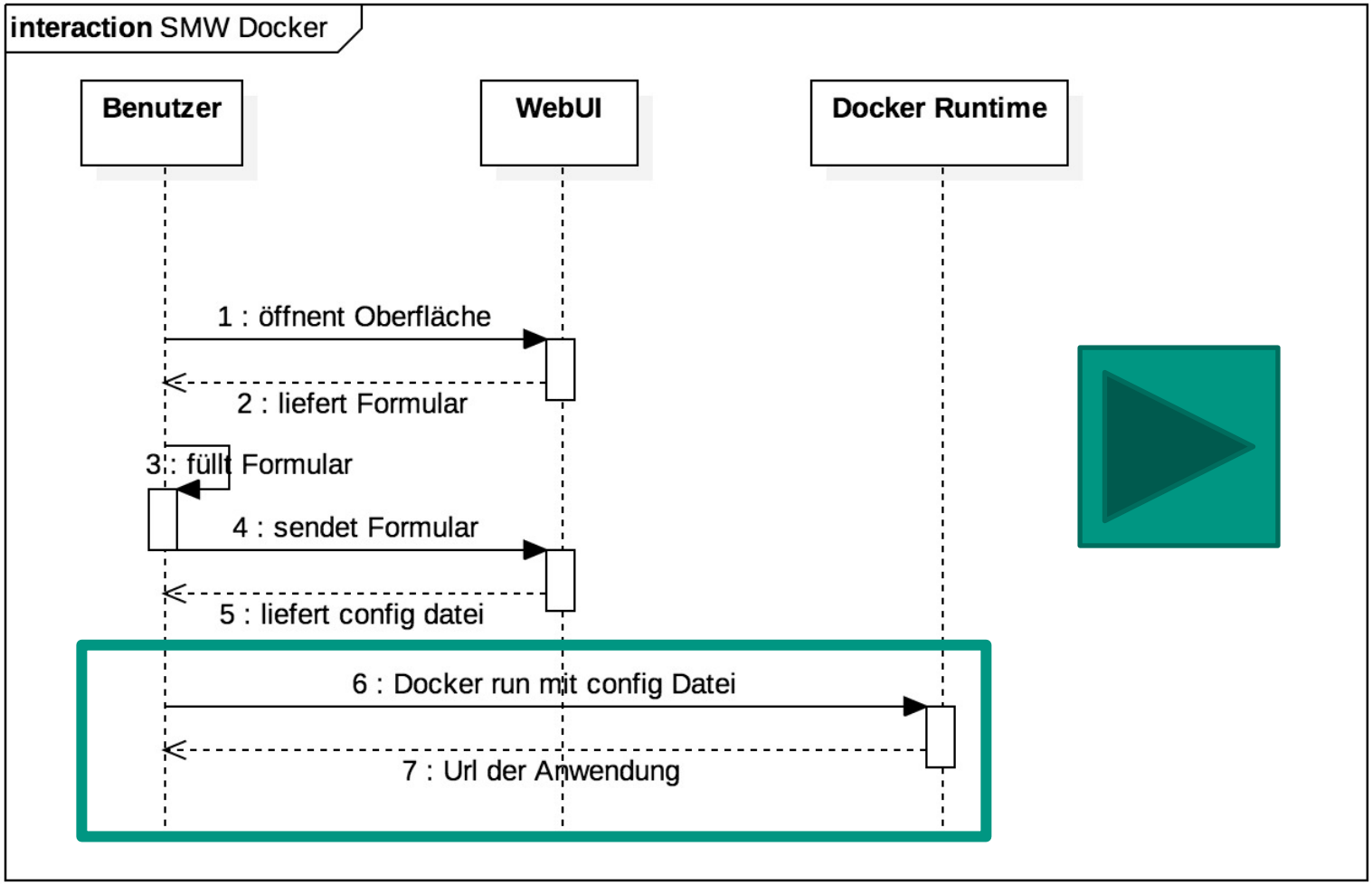
Angular

- Open-Source Webapplikationsframework entwickelt von großer Community angeführt von Google
- Vielzahl an vordefinierten Packages, hier Material Design Components
  - → User findet schnell zurecht
- Kombination von HTML5 und Typescript erlaubt eine Codebasis für jede Plattform
  - TypeScript ⊇ ECMAScript-6 und Rückwertskompatibel zu ECMAScript-5
  - → moderne, mächtige Webanwendungen

Mediawiki + Extensions mit Docker

- Mediawiki als offizielle Image im Dockerhub
  - Wird von der Foundation/Community verwaltet
  - Verwendung als Basisimage
- Erweitern durch Composer und SemanticMediawiki
  - Installation des Composer
  - Installation SemanticMediawiki mithilfe des composer
  - Kopieren eines initialisierungs-Skript
  - Speichern als neues Docker Image
- Zur Laufzeit
  - Initialisieren der Datenbank
  - Installation der individuellen Extensions
    - Download + registrieren in LocalSettings.php
    - Composer optional

Docker Demo


interaction SMW Docker
Benutzer
WebUI
Docker Runtime
1 : öffnent Oberfläche
2 : liefert Formular
3 : füllt Formular
4 : sendet Formular
5 : liefert config datei
6 : Docker run mit config Datei
7 : Url der Anwendung

Probleme

Installation und Auswahl von Erweiterungen

- Kein einheitlicher Marktplatz

→ Begrenzung auf offizielles MediaWiki Github-Repository

- Keine einheitliche Installationsmethode und nicht ohne weiteres ersichtlich, welche verwendet wird

→ Verwendung der „alten“ Installationsmethode

Probleme

Migration alter MediaWikis

Vorgehen

- Datenbank einspielen, Zugangsdaten in LocalSettings.php anpassen
- Bau einer minimalen LocalSettings.php

Erkenntnisse

- Liste mit installierten Extension, die Installationsmethode angibt ist unumgänglich
- Änderungen an der LocalSettings.php müssen bekannt sein und für jeden Docker Container nachgeholt werden
- Die referenzierten Repositories müssen zugänglich sein

➔ Migration möglich, aber mit Anpassungen verbunden

Weiterführende Arbeit

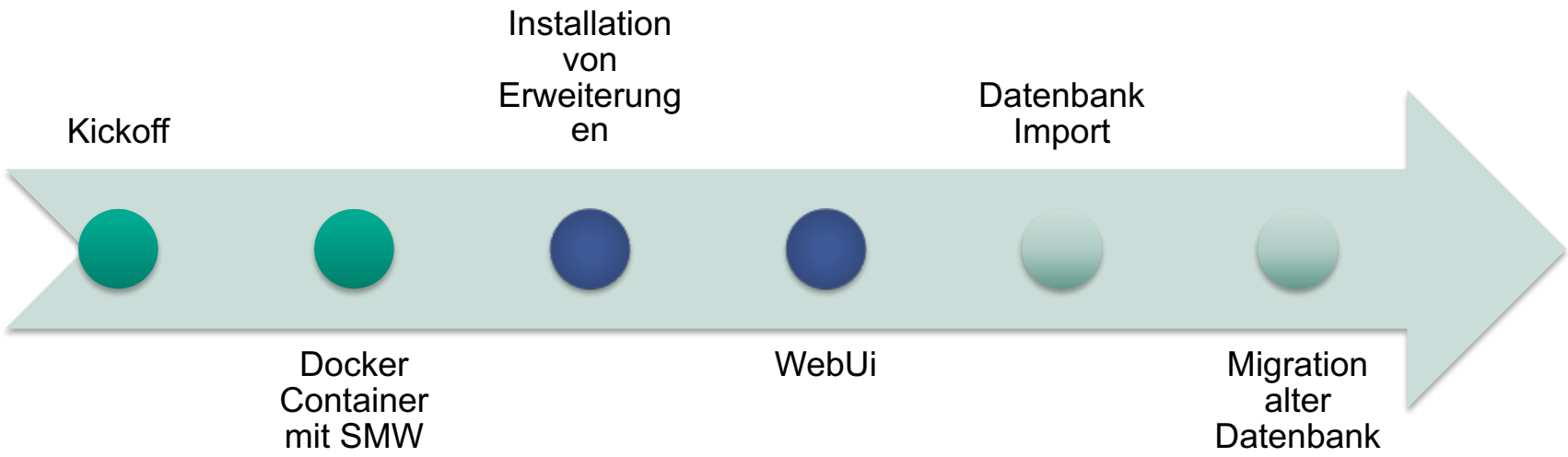
- Weiterentwicklung zu Produktivsystem mittels Docker Swarm, Kubernetes
  - → Gute Skalierbarkeit
  - → Ausfallsicherheit durch Redundanzen
- Einheitliche Verwaltung von Erweiterungen (und Skins)
  - →Konfigurationsdateien automatisch mit der Auswahl generieren
- Mittelfristig einheitliche Installation mittels Composer
- Erstellen eines Marketplaces

Vielen Dank für ihre Aufmerksamkeit

Fragen?

https://github.com/RaphaelManke/docker-smw-all

Roadmap


Kickoff
Docker Container mit SMW
Installation von Erweiterungen
WebUi
Datenbank Import
Migration alter Datenbank

Aktuelle Probleme

- Welche MediaWiki Erweiterungen?
  - Installationsmethode nicht immer gleich
  - Kompatibilität mit Versionen
    - insbesondere beim Update
- Skins werden nicht in zentralem Repository verwaltet
  - Webcrawler? Händische Selektion (speichern in einer Datenbank)?
- Extensions zur build- oder runtime installieren

- Einfache und Intuitive Weboberfläche gestalten
  - Welche Extensions anzeigen? Wie sortiert?
  - Extension-index cachen? (Datenbank)

Was ist noch geplant?

- Node.js Container für UI ist noch nicht eingebunden
- Momentan muss
- Schwierig herauszufinden, welche Erweiterungen unterstützt werden können