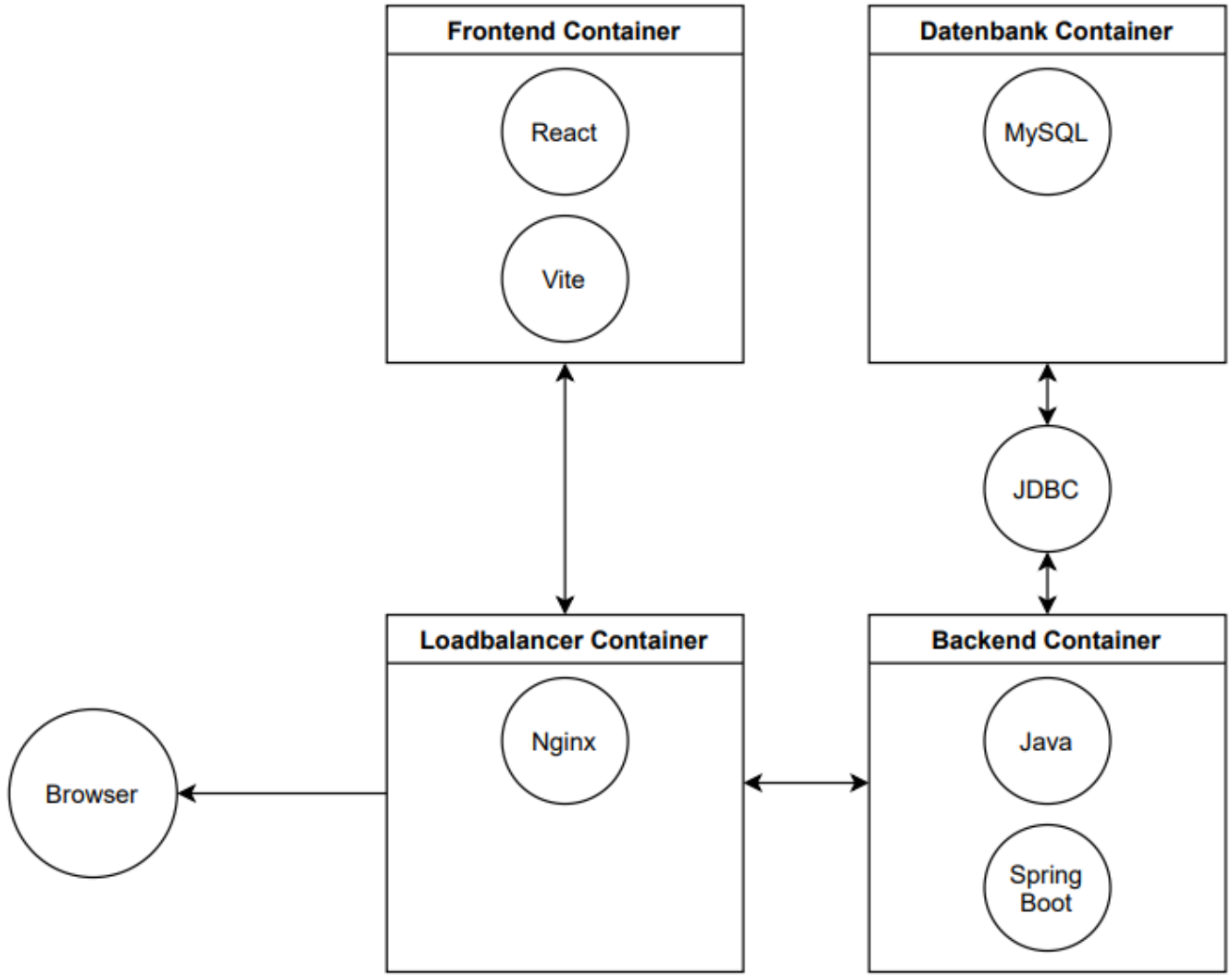
Sudocker

Von Philipp Schmid, Thomas Schneider, Delia Fasel

Infrastruktur

Infrastrucktur der Sudoker-App



Softwarekomponenten

- GitHub Actions
- Docker und Docker Compose
- Pandoc und LaTex
- GitHub Secrets
- ZIP-Tools

Workflow Einrichtung

```yaml
on:
  push:
    branches:
      - main

jobs:
  build-and-package:
    runs-on: ubuntu-latest
    env:
      SPRING_DATASOURCE_DRIVER_CLASS_NAME: com.mysql.cj.jdbc.Driver
      SPRING_JPA_HIBERNATE_DDL_AUTO: update
      SPRING_JPA_SHOW_SQL: true
```

```yaml
steps:
- name: Checkout Repository
  uses: actions/checkout@v2

- name: Install Pandoc and LaTeX
  run: |
    sudo apt-get update
    sudo apt-get install -y pandoc texlive-latex-base texlive-fonts-recommended texlive-fonts-extra texlive-latex-extra

- name: Convert Markdown to PDF
  run: pandoc ./Projektdokumentation.md -o ./Projektdokumentation.pdf

- name: Set up Docker Buildx
  uses: docker/setup-buildx-action@v1
```

Vorbereitung und Docker

Docker-Build und Verpackung

```
- name: Create Application Properties
  run: |
    mkdir -p ./Projekt/M223_Sudoku-main/src/main/resources/
    echo "spring.datasource.url=${{ secrets.SPRING_DATASOURCE_URL }}" >> ./Projekt/M223_Sudoku-main/src/main/resources/application.properties
    echo "spring.datasource.username=${{ secrets.SPRING_DATASOURCE_USERNAME }}" >> ./Projekt/M223_Sudoku-main/src/main/resources/application.properties
    echo "spring.datasource.password=${{ secrets.SPRING_DATASOURCE_PASSWORD }}" >> ./Projekt/M223_Sudoku-main/src/main/resources/application.properties
    echo "spring.datasource.driver-class-name=${{ env.SPRING_DATASOURCE_DRIVER_CLASS_NAME }}" >> ./Projekt/M223_Sudoku-main/src/main/resources/application.properties
    echo "spring.jpa.hibernate.ddl-auto=${{ env.SPRING_JPA_HIBERNATE_DDL_AUTO }}" >> ./Projekt/M223_Sudoku-main/src/main/resources/application.properties
    echo "spring.jpa.show-sql=${{ env.SPRING_JPA_SHOW_SQL }}" >> ./Projekt/M223_Sudoku-main/src/main/resources/application.properties
    echo "spring.security.user.name=${{ secrets.SPRING_SECURITY_USER_NAME }}" >> ./Projekt/M223_Sudoku-main/src/main/resources/application.properties
    echo "spring.security.user.password=${{ secrets.SPRING_SECURITY_USER_PASSWORD }}" >> ./Projekt/M223_Sudoku-main/src/main/resources/application.properties
    echo "wissquiz.app.jwtSecret=${{ secrets.WISSQUIZ_APP_JWTSECRET }}" >> ./Projekt/M223_Sudoku-main/src/main/resources/application.properties
    echo "wissquiz.app.jwtExpirationMs=${{ secrets.WISSQUIZ_APP_JWTEXPIRATIONMS }}" >> ./Projekt/M223_Sudoku-main/src/main/resources/application.properties

- name: Build Docker Images
  run: |
    docker-compose -f ./Projekt/docker-compose.yml build

- name: List Docker Images
  run: docker images

- name: Save Docker Images
  run: |
    docker save -o docker_images.tar projekt_frontend:latest projekt_backend:latest

- name: Create Zip File
  run: zip -r LB-Projekt-M210_Delia-Thomas-Philipp.zip ./Projekt docker_images.tar ./Projektdokumentation.pdf ./Presentation.pdf
```

Release und Deployment

```yaml
- name: Upload Artifact
  uses: actions/upload-artifact@v2
  with:
    name: LB-Projekt-M210_Delia-Thomas-Philipp
    path: LB-Projekt-M210_Delia-Thomas-Philipp.zip

- name: Create Release
  id: create_release
  uses: actions/create-release@v1
  env:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
  with:
      tag_name: ${{ github.run_id }}
      release_name: Release ${{ github.run_id }}
      draft: false
      prerelease: false

- name: Upload Release Asset
  uses: actions/upload-release-asset@v1
  env:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
  with:
      upload_url: ${{ steps.create_release.outputs.upload_url }}
      asset_path: ./LB-Projekt-M210_Delia-Thomas-Philipp.zip
      asset_name: LB-Projekt-M210_Delia-Thomas-Philipp.zip
      asset_content_type: application/zip
```

Docker Compose

- Erstellt Container für DB, Frontend, Backend
- Nginx-Loadbalancer für die orchestrierung
- Verbunden durch vordefinierte Ports

```yaml
db:
  image: mysql:latest
  environment:
    MYSQL_DATABASE: 'sudoku'
    MYSQL_USER: 'sudoku'
    MYSQL_PASSWORD: 'sudoku'
    MYSQL_ROOT_PASSWORD: 'sudoku'  # Consider using a different password for root
  ports:
    - "3307:3306"
  command: --default-authentication-plugin=mysql_native_password

  volumes:
    - ./init.sql:/docker-entrypoint-initdb.d/init.sql
    - db_data:/var/lib/mysql
```

Docker Compose - DB

```yaml
backend:
  build: ./M223_Sudoku-main
  ports:
    - "8080:8080"
  environment:
    - SPRING_DATASOURCE_URL=jdbc:mysql://db:3306/sudoku
    - SPRING_DATASOURCE_USERNAME=sudoku
    - SPRING_DATASOURCE_PASSWORD=sudoku
  depends_on:
    - db
```

Docker Compose - Backend

```yaml
frontend:
  build: ./M223_Sudoku_Frontend2-main
  ports:
    - "5174:5174"
loadbalancer:
  image: nginx:latest
  ports:
    - "8081:80"
  depends_on:
    - backend
    - frontend
```

Docker Compose - Frontend & Loadbalancer

Docker Backend

```
# Stage 1: Build the application
# Use a Maven image that includes OpenJDK 21
FROM maven:latest as build

# Copy the project files to the container
COPY ./pom.xml /usr/src/myapp/pom.xml
COPY ./src /usr/src/myapp/src

# Set the working directory
WORKDIR /usr/src/myapp

# Package the application
RUN mvn clean package -DskipTests

# Stage 2: Create the Docker container
# Use an OpenJDK 21 image for the runtime
FROM openjdk:latest

# Copy the JAR file from the build stage
COPY --from=build /usr/src/myapp/target/*.jar /usr/app/app.jar

# Expose the port the application runs on
EXPOSE 8080

# Run the JAR file
ENTRYPOINT ["java","-jar","/usr/app/app.jar"]
```

Docker Frontend

```
# Stage 1: Build the application
# Use a Node.js image to build the application
FROM node:latest as build

# Set the working directory
WORKDIR /app

# Copy package.json and package-lock.json (or yarn.lock if you use Yarn)
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application source code
COPY . .

# Build the application
RUN npm run build

# Stage 2: Serve the application
# Use a lightweight server, like nginx, to serve the built files
FROM nginx:alpine

# Copy the build directory from the build stage to the server's root
COPY --from=build /app/dist /usr/share/nginx/html

# Expose the port the server listens on
EXPOSE 80

# Start nginx
CMD ["nginx", "-g", "daemon off;"]
```