Deployment

Ziel dieser Aufgabe ist des das CRUD-Assignment in eine einizge Docker-Compose Datei zu packen um diese von überall aus ausführbar zu machen (deployment). Damit wäre es im weiteren auch möglich diese Datei auf einer Azure-Instanz o.ä hochzuladen und von dort aus auszuführen. Auch Interessant ist diese Art des Deployments für IT-Abteilungen von Firmen mit denen man zusammenarbeitet. Die freuen sich meistens wenn man Ihnen durch so etwas die Arbeit etwas abnimmt und man ist bei Updates nicht auf sie angewiesen, vorausgesetzt es wird immerwieder die neueste Version aus dem Registry gepullt (`docker pull <imagename>`).

Dazu braucht es folgende Voraussetzungen:

- Dockerfiles
- Images
- Container-Registry
- CI/CD Pipelines
- Je ein Repository für Frontend und Backend (bereits exisiterende können ggf. weiterverwendet werden)

Füge (falls nicht schon vorhanden) sowohl im Frontend als auch im Backend ein Dockerfile hinzu. Im übergeordneten Ordner kann sobald beide Dateien Images builden ein docker-compose mit folgender Struktur angelegt werden:

```
version: "xy"

services:
   frontend:
       build: ./frontend
   backend:
       build: ./backend
```

Das ist schon ein guter Start, jedoch setzt dieses Dockerfile voraus das der Programmcode in den Unterordnern vorliegt. Für einen Entwickler ist das kein Problem, aber wenn man das Programm dann beim Auftraggeber installieren möchte, möchte man dem im Normalfall nicht den Code geben. Deswegen ist es besser wenn man die Images in einer `Container-Registry` hochlädt um von überall aus auf diese zugreifen zu können. Dann schaut das Compose ca. so aus:

```
version: "xy"

services:
   frontend:
       image: ghcr.io/<username>/<CRUD_frontend_imagename>

   backend:
       image: ghcr.io/<username>/<CRUD_backend_imagename>
```

Aufbau des Image-Feldes

- ghcr.io Github container registry
- Dein Username auf Github
- Der Name des Images der beim CI/CD wird meistens über einen Tag in Git festgelegt

Zum Vergleich: Ein Image weit verbreiteter Software sieht oft so aus `mysql`. Dabei wurden für einige der obigen Felder Standardannahmen getroffen. In voller Länge würde es so aussehen: `docker.io/libraries/mysql:latest`. Alles bis auf `mysql` sind Standardparameter.

Einrichten der CI/CD

Die Containerimages kann man entweder Lokal builden und ins Registry hochladen oder sich auf den Github-Servern bauen lassen. Bei den Github-Servern ist der Vorteil dass man sicherstellen kann das vor dem Buildprozess Unittests, Sicherheitschecks etc. durchgeführt werden.

Um die Github Server anzuweisen ist folgende Ordnerstruktur im Repository zu erstellen: `.github/workflows/<docker_build_actionname.yml>`

Die Datei könnte ca. so aussehen:

```
name: Create and Push a Dockerfile inte the registry

on:
  push:
    tags:
      - '*' # jobs ausführen wenn ein git tag gepusht wird
      # branches: [ main ] # jobs ausführen wenn etwas auf den main branch gepusht
wird

permissions:
  contents: write

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2 # schritt eins repo am server klonen

    - name: Build with kaniko
    - uses: ... # mit kaniko (o.ä) das dockerfile bauen und dann ins repository
hochladen


    - name: Release with Notes # freiwilliges extra, release im repo erzeugen
      uses: softprops/action-gh-release@v1
      with:
        # todo .. pfad zur jar-datei gezippter dist ordner etc.
```

```
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

> Anleitung zum Containerimages bauen mit Kaniko: https://cloud.google.com/build/docs/optimize-builds/kaniko-cache?hl=de

Bewertungsgrundlagen bei der Abgabe

Füge jene docker-compose.yml Datei in das Repository der aktuellen Abgabe hinzu welche auf die von dir erstellten Images verweist.

Abzugeben ist:

- eine Docker-Compose Datei mit Images
- ggf. Logindaten um Zugriff auf die Lokalen Container zu erhalten
    - NICHT das eigene Passwort sondern nur einen Zugangstoken für die Registries

Punkteverteilung

- Nach einem Docker-Login von meiner Seite mit eurem Token kann ich die Images pullen 25P
- Die Images können ausgeführt werden ohne sofort abzustürzen 25P
- Bei den Tags gibt es sowohl eine aktuelle Version vX.Y als auch einen `latest`-Tag für die neueste Version 25P
- Beim Build eines neuen Images gibt es eine Schritt (Step) für das Ausführen von Unittests 25P