LU09b - Deployment von Datenbanken & Backends

AWS-Umgebung (Docker)

Der einfachste Weg, um Datenbank und Backend zu hosten, ist ein entsprechender Docker-Container zu erstellen. In der AWS-Buildumgebung ist der Zugriff auf die zugrundeliegende Docker-Instanz gewährt.

Damit man von ausserhalb der AWS-Umgebung auf diesen Backend-Container zugriefen kann wurde in der BZZ-AWS-Build-Umgebung ein Forwarding aktiviert, welches HTTP-Anfragen, die dem erwarteten Muster entsprechend an den Port 5000 des entsprechenden Containers weiterleitet.

Allgemein: `${BASE_URL}/api/${PROJECT_NAME}/${BRANCH_NAME}/${PATH}` → `${PROJECT_NAME}_${BRANCH_NAME}_backend:5000/${PATH}`

Konkrete Beispiele:

- „http://54.80.83.95/api/cicd/develop/api/polls“ → „cicd_develop_backend:5000/api/polls“
- „http://54.80.83.95/api/cicd/develop/api/votes“ → „cicd_develop_backend:5000/api/votes“

```
    location ~ ^/api/([^/]+)/([^/]+)/(.*)$ {
        proxy_pass http://$1_$2_backend:5000/$3;
    }
```

Das bedeutet, dass im Frontend sichergestellt werden muss, dass Aufrufe des Backends die richtige URL verwenden.

Pipeline	**Frontend**

`npm install`
`REACT_APP_API_BASE=/api/${PROJECT_NAME}/${BRANCH_NAME}/api \`
`npm run build`
	`const API_BASE = process.env.REACT_APP_API_BASE;`
	`...`
	`fetch(`${API_BASE}/votes?poll_id=${pollId}`)`
	`fetch(`${API_BASE}/votes`,... )`
	`fetch(`${API_BASE}/polls/${code}`)`
	`fetch(`${API_BASE}/polls`,... )`

Dasselbe gilt natürlich auch für das Backend, dessen Container richtig benannt und dessen Port korrekt gesetzt werden muss.

Pipeline	**Backend**

`BACKEND_CONTAINER = "${PROJECT_NAME}_${BRANCH_NAME}_backend"`
`...`
`docker build -t $BACKEND_CONTAINER backend/`
`docker run -d \`
`  --name $BACKEND_CONTAINER \`
`  ...`
`  $BACKEND_CONTAINER`
	`flask_app = create_app()`
	`flask_app.run(debug=True, host="0.0.0.0", port=5000)`

Die Datenbank muss wiederum so konfiguriert werden, dass sie vom Backend angesprochen werden

kann.

Die Credentials werden hier im Klartext verwendet, damit die Funktionsweise klarer ist. In der Praxis werden natürlich Secrets dafür verwendet.

Pipeline	**Pipeline**

`DB_CONTAINER = "${PROJECT_NAME}_${BRANCH_NAME}_db"`
`DB_USER = "appuser"`
`DB_PASSWORD = "apppassword"`
`DB_NAME = "appdb"`
`...`
`-e DATABASE_URL="postgresql://$DB_USER:$DB_PASSWORD@$DB_CONTAINER:5432/$DB_NAME" \`	

```
docker run -d \
    --name $DB_CONTAINER \
    -e POSTGRES_USER=$DB_USER \
    -e POSTGRES_PASSWORD=$DB_PASSWORD \
    -e POSTGRES_DB=$DB_NAME \
```

<WRAP> |

TODO