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 Backend	Pipeline Datenbank

`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 \`

Beispiel-Deployment

Alle Stages für das Deployment von https://github.com/AlexanderPeter/cicd auf der AWS-Umgebung ist hier aufgeführt.

```
pipeline {
    agent any

    environment {
        PROJECT_NAME       = "cicd"
        BRANCH_NAME        = "develop"
        REPO_URL           =
"https://github.com/AlexanderPeter/${PROJECT_NAME}.git"
        TARGET_DIR         =
"/var/jenkins_home/projects/${PROJECT_NAME}/${BRANCH_NAME}"
        SONAR_SCANNER_OPTS = "-Xmx512m"
        BACKEND_CONTAINER  = "${PROJECT_NAME}_${BRANCH_NAME}_backend"
        DB_CONTAINER       = "${PROJECT_NAME}_${BRANCH_NAME}_db"
        DB_VOLUME          = "${PROJECT_NAME}_${BRANCH_NAME}_db_data"
        DB_USER            = "appuser"
        DB_PASSWORD        = "apppassword"
        DB_NAME            = "appdb"
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: BRANCH_NAME,
                    url: REPO_URL
            }
        }

        stage('Start Database') {
            when {
```

```
                expression { false }
            }
            steps {
                sh '''
                    echo "Starting DB with workspace $WORKSPACE"

                    docker stop $DB_CONTAINER || true
                    docker rm $DB_CONTAINER || true

                    docker run -d \
                      --name $DB_CONTAINER \
                      --network infra-net \
                      -e POSTGRES_USER=$DB_USER \
                      -e POSTGRES_PASSWORD=$DB_PASSWORD \
                      -e POSTGRES_DB=$DB_NAME \
                      -v $DB_VOLUME:/var/lib/postgresql/data \
                      -v $WORKSPACE/database:/docker-entrypoint-initdb.d \
                      postgres:17
                '''
            }
        }

        stage('Initialize Database') {
            when {
                expression { false }
            }
            steps {
                sh '''
                    TABLE_EXISTS=$(docker exec $DB_CONTAINER psql -U
$DB_USER -d $DB_NAME -tAc "SELECT to_regclass('public.poll')")
                    if [ "$TABLE_EXISTS" = "" ]; then
                        echo "Initializing schema..."
                        docker exec -i $DB_CONTAINER psql -U $DB_USER -d
$DB_NAME < database/schema.sql
                    else
                        echo "Database already initialized."
                    fi
                '''
            }
        }

        stage('Build Frontend') {
            when {
                expression { false }
            }
            steps {
                dir('frontend') {
                    sh '''
                        npm install
                        GENERATE_SOURCEMAP=false \
                        NODE_OPTIONS="--max-old-space-size=1024" \
```

```
                        PUBLIC_URL=/projects/${PROJECT_NAME}/${BRANCH_NAME}
\
REACT_APP_API_BASE=/api/${PROJECT_NAME}/${BRANCH_NAME}/api \
                        npm run build
                    '''
                }
            }
        }

        stage('Deploy Frontend') {
            when {
                expression { false }
            }
            steps {
                sh '''
                    echo "Deploying frontend to $TARGET_DIR"

                    mkdir -p "$TARGET_DIR"
                    rm -rf "$TARGET_DIR"/*

                    cp -r frontend/build/* "$TARGET_DIR"/
                '''
            }
        }

        stage('Deploy Backend') {
            when {
                expression { false }
            }
            steps {
                sh '''
                    docker build -t $BACKEND_CONTAINER backend/

                    docker stop $BACKEND_CONTAINER || true
                    docker rm $BACKEND_CONTAINER || true

                    docker run -d \
                        --name $BACKEND_CONTAINER \
                        --network infra-net \
                        -e
DATABASE_URL="postgresql://$DB_USER:$DB_PASSWORD@$DB_CONTAINER:5432/$DB_NAME
" \
                        $BACKEND_CONTAINER
                '''
            }
        }
    }
}
```