Way2Cloud: Container-Workshop

Voraussetzungen:

- Du hast bereits ein Maven-Projekt und das JAR-File wurde erfolgreich erstellt.
- Docker ist auf deinem lokalen Rechner installiert.

Schritt 1: JAR-File mit Maven erstellen

Falls du das JAR noch nicht erstellt hast, führe den folgenden Befehl aus, um es zu erstellen:

```
bash
Code kopieren
mvn clean package
```

Das JAR-File wird standardmäßig im Ordner `target/` deines Projekts erstellt.

Schritt 2: Dockerfile erstellen

Erstelle in deinem Projektverzeichnis eine Datei namens `Dockerfile` mit folgendem Inhalt:

```
Dockerfile
Code kopieren
# Verwende ein schlankes OpenJDK-Basisimage auf Alpine
FROM openjdk:17-alpine

# Setze ein Arbeitsverzeichnis im Container
WORKDIR /app

# Kopiere das erstellte JAR-File in das Image
COPY target/<dein-jar-file>.jar app.jar

# Definiere den Befehl, um die Anwendung zu starten
CMD ["java", "-jar", "app.jar"]
```

Erklärungen:

- **FROM openjdk:17-alpine**: Dies verwendet ein leichtgewichtiges OpenJDK-Basisimage, das auf Alpine Linux basiert.
- **WORKDIR /app**: Setzt das Arbeitsverzeichnis innerhalb des Containers auf `/app`.
- **COPY target/<dein-jar-file>.jar app.jar**: Kopiert dein JAR-File aus dem `target`-Ordner in das Arbeitsverzeichnis `/app` des Containers und benennt es `app.jar`.
- **CMD ["java", "-jar", "app.jar"]**: Führt das JAR-File aus, wenn der Container gestartet wird.

Schritt 3: Docker-Image bauen

Wechsle in das Verzeichnis, in dem dein `Dockerfile` liegt, und führe folgenden Befehl aus, um das Docker-Image zu erstellen:

```
bash
Code kopieren
docker build -t mein-java-app .
```

Das `-t mein-java-app` taggt das Image mit dem Namen `mein-java-app`.

Schritt 4: Docker-Container starten

Nachdem das Image gebaut wurde, kannst du den Container starten:

```
bash
Code kopieren
docker run -d -p 8080:8080 mein-java-app
```

Hierbei:

- `-d`: startet den Container im Hintergrund (detached mode).
- `-p 8080:8080`: mappt den Port 8080 des Containers auf den Port 8080 deines Hosts. Passe dies an, falls deine Anwendung auf einem anderen Port läuft.

Schritt 5: Container überprüfen

Überprüfe, ob dein Container läuft:

```
bash
Code kopieren
docker ps
```

Du solltest deinen Container mit dem Namen `mein-java-app` in der Liste sehen. Nun kannst du auf deine Anwendung zugreifen, indem du die entsprechende URL aufrufst (z. B. `http://localhost:8080`, je nach Anwendung).

Optional: Container stoppen

Falls du den Container stoppen möchtest:

```
bash
Code kopieren
docker stop <container-id>
```

Die `container-id` kannst du mit dem `docker ps`-Befehl herausfinden.

Zusammenfassung der wichtigsten Docker-Befehle

1. **Image bauen**: `docker build -t way2millionapp .`
2. **Container starten**: `docker run -p 8080:8080 way2millionapp`
3. **Logs ansehen**: `docker logs -f <container-id>`
4. **Container stoppen**: `docker stop <container-id>`
5. **Image pushen**: `docker push <username>/way2millionapp:latest`