Docker Compose

Overview

In this lab you'll use Docker Compose to build two images and run two linked containers:

- ActiveMQ message broker
- Spring Boot application that sends/receives messages to a queue

IntelliJ projects

Demo project: **`demos\demo-docker-compose-mq`**

Roadmap

There are 4 exercises in this lab, of which the last exercise is "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Familiarization with the application code
2. Familiarization with Docker-related files
3. Using Docker Compose
4. (If Time Permits) Additional suggestions

Exercise 1: Familiarization with the application code

In IntelliJ, open the **`demo-docker-compose-mq`** project. This project contains Java code to send and receive messages to an ActiveMQ message queue. The application code is complete, and we've already compiled it and packaged it into a JAR file.

Take a look at the Java source code to see what it does:

`Application.java`	- Application entry-point
`EmployeeSender.java`	- Spring Boot component, sends messages to a queue
`EmployeeReceiver.java`	- Spring Boot component, receives messages from queue

Exercise 2: Familiarization with Docker-related files

We've already written all the Docker-related files for this lab. Take a look, and make sure you understand what's going on:

`Dockerfile-app`	- Builds a Docker image containing the application
`Dockerfile-activemq`	- Builds a Docker image containing ActiveMQ
`docker-compose.yaml`	- Orchestrates building of images and running of containers

Exercise 3: Using Docker Compose

Open a terminal window and run the following command, to build the images and run the containers for the application and for ActiveMQ:

```
docker-compose up
```

All being well, ActiveMQ should start up, and then your Spring Boot application should send and receive messages to a queue in ActiveMQ.

Exercise 4 (If time permits): Additional suggestions

Docker Compose is capable of linking any number of containers together. For example, if you're feeling brave, integrate the code for this lab (which features messaging) with the code from the demo (which features MySQL persistence) into a single application comprising 3 linked containers:

- Spring Boot application
- ActiveMQ message broker
- MySQL database engine